NAKED MINI® 4

FORTRAN IV

OPERATIONS MANUAL

93560-80

APRIL 1981



ComputerAutomation®

NAKED MINI® Division

Engineering Reference Number
90-93580-01F1

HERTFORD HOUSE
DENHAM WAY, MAPLE CROSS
RICKMANSWORTH, HERTFORDSHIRE WD3 2XD

Telephone Rickmansworth 71211 Telex 922654

Printed in U.S.A.

REVISION HISTORY

Revision	Date	Comments
A0	Feb. 1979	Original issue.
B0-C0		Skipped for configuration control.
D0	Aug. 1979	Adds I/O status messages; corrects minor documentation errors.
E0	Feb. 1980	Changes form in which Emulator module is delivered; incorporates other revisions.
F0	Oct. 1980	Documents FORTRAN.OPT version of the runtime library; re-instates example Assembly language program calling a FORTRAN subroutine; corrects minor errors.
F1	Apr. 1981	Omits F: subroutines restricted to internal use.

Computer Automation, Inc., reserves the right to make changes in specifications at any time and without notice. The information furnished by Computer Automation, Inc., in this publication is believed to be accurate and reliable. However, no responsibility is assumed by Computer Automation, Inc., for its use; nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Computer Automation, Inc.

Computer Automation, Inc., welcomes comments and suggestions for improvements to our manuals. Should difficulty be encountered with the information in the manual or if you have a suggestion for manual improvement, please send the comments or suggestions to Computer Automation, Inc., Technical Publications, Department 11-0770, 18651 Von Karman, Irvine, California, 92713. Please provide as much information as possible to help evaluate the comment or suggestion.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

PREFACE

This manual is intended to aid the FORTRAN programmer in compiling, linking, loading, and executing programs on the NAKED MINI® 4 Family computers.

The manual is organized in a generally chronological order, according to the normal sequence of operations involved in developing and executing a FORTRAN program. The FORTRAN operating environment is described first, followed by compiler operation, library structure and linking procedures, and runtime procedures. The generation of compatible assembly language routines and system generation procedures are described at the end of the manual.

This manual refers frequently to procedures performed in the OS4 system and/or in the RTX4/IOS4 system; the reader should be thoroughly familiar with these systems.

The following Computer Automation, Incorporated, documents provide related information:

	Model Number
• NAKED MINI® 4 FORTRAN IV Language Manual	93560-90
• OS4 System Reference Manual	93460-90
• Real-Time Executive RTX4 Reference Manual	93410-90
• Input/Output Subsystem IOS4 Reference Manual	93405-90
• NAKED MINI 4 Assembler Reference Manual	93500-80
• NAKED MINI 4 Optional Instructions Emulator Manual	93063-90

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

©1980 COMPUTER AUTOMATION, INC.

CONTENTS

SECTION 1 OVERVIEW OF FORTRAN OPERATIONS 1-1

1.1 INTRODUCTION 1-1

1.2 OPERATING ENVIRONMENT 1-3

SECTION 2 THE FORTRAN COMPILER 2-1

2.1 INTRODUCTION 2-1

2.2 COMPILER ORGANIZATION AND OPERATION 2-2
2.2.1 Scan Phase 2-2
2.2.2 Allocate Phase 2-2
2.2.3 Generation Phase 2-2

2.3 COMPILER LOGICAL UNIT ASSIGNMENTS 2-3

2.4 INVOKING THE COMPILER: THE /FORTRAN COMMAND 2-4
2.4.1 Command Line Conventions 2-4
2.4.2 Command Options 2-5
2.4.3 Command Formats 2-9

2.5 COMPILER LISTINGS 2-11
2.5.1 Source Listing 2-11
2.5.2 Variable Storage Allocation Listing 2-13
2.5.3 Cross-Reference Listing 2-15
2.5.4 Object Listing 2-17
2.5.5 Summary 2-25

SECTION 3 LIBRARY STRUCTURE AND LINKING 3-1

3.1 INTRODUCTION 3-1

3.2 LIBRARY STRUCTURE AND USE 3-1

3.3 LINKING FOR EXECUTION UNDER OS4 3-2

3.4 LINKING FOR EXECUTION UNDER RTX. 3-2

SECTION 4 RUNTIME 4-1

4.1 INTRODUCTION 4-1

4.2 RUNTIME LOGICAL UNIT ASSIGNMENTS 4-2
4.2.1 Device Assignments for Execution under OS4 . . 4-2
4.2.2 Device Assignments for Execution under RTX4 . . 4-3

4.3 PROGRAM LOADING AND EXECUTION 4-3
4.3.1 Loading and Execution under OS4 4-4
4.3.2 Loading and Execution under RTX4 4-4

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

4.4 PROGRAM/USER INTERACTION 4-4
4.4.1 PAUSE Messages 4-4
4.4.2 STOP Messages 4-5

SECTION 5 INTERFACING TO ASSEMBLY LANGUAGE 5-1

5.1 INTRODUCTION 5-1

5.2 ASSEMBLY LANGUAGE SUBPROGRAMS CALLED FROM FORTRAN . . 5-1
5.2.1 Calling Program Requirements 5-2
5.2.2 Requirements of the Subprogram 5-2

5.3 FORTRAN SUBPROGRAMS CALLED FROM AN ASSEMBLY LANGUAGE PROGRAM 5-17

SECTION 6 SYSTEM DISKETTES 6-1

6.1 INTRODUCTION 6-1

6.2 FORTRAN PRODUCT DISKETTES 6-1

6.3 INSTRUCTION EMULATOR MODULE 6-2

APPENDIXES

APPENDIX A FORTRAN OPERATIONS SUMMARY A-1

APPENDIX B FORTRAN IV ERROR MESSAGES B-1

B.2 COMPILER DIAGNOSTICS DURING ALLOCATE PHASE B-6

B.3 FORTRAN RUNTIME ERROR MESSAGES B-7

B.4 FORTRAN RUNTIME I/O STATUS (IOSTAT=) VALUES B-9

APPENDIX C FORTRAN IV RUNTIME LIBRARY SUBROUTINES C-1

APPENDIX D RETURNING COMPLEX AND DOUBLE PRECISION VALUES TO FORTRAN FROM AN ASSEMBLY LANGUAGE FUNCTION SUBPROGRAM D-1

D.1 FORMAT OF FUNCTION RESULT FOR COMPLEX VALUES D-1

D.2 FORMAT OF FUNCTION RESULT FOR DOUBLE PRECISION VALUES D-2

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

FIGURES

Figure		Page
1-1	FORTRAN Program Development Sequence	1-2
2-1(1)	Sample Compiler Listing (Source Listing)	2-12
2-1(2)	Sample Compiler Listing (Variable Storage Allocation Listing) .	2-14
2-1(3)	Sample Compiler Listing (Cross-Reference Listing)	2-16
2-1(4)	Sample Compiler Listing (Object Listing)	2-20
2-1(5)	Sample Compiler Listing (Summary)	2-26
5-1	Sample Calling Program	5-6
5-2	Sample Assembly Language Subroutine	5-8
5-3	Sample Assembly Language Function	5-10
5-4	LINK Memory Map of Sample FORTRAN Program	5-12
5-5	Execution Printout of Sample FORTRAN Program	5-15
5-6	Sample Assembly Language Program Calling a FORTRAN Subprogram .	5-19
5-7	FORTRAN Subprogram Called by Sample Assembly Language Program .	5-22

TABLES

Table		
6-1	Emulator Routines Needed for FORTRAN Compilation/Execution . .	6-3

Revision F1

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

SECTION 1

OVERVIEW OF FORTRAN OPERATIONS

1.1 INTRODUCTION

Developing a FORTRAN program involves a series of sequential procedures:

1) The programmer designs and codes the FORTRAN program and transfers the code to a suitable input medium.

2) The OS4 FORTRAN IV compiler translates the FORTRAN source input to linkable object code.

3) The OS4 LINK utility links the object code output by the compiler to the needed library files and to any other required files.

4) The linked FORTRAN program is loaded by the OS4 /AUTOLOAD command or the hardware AutoLoad facility (if execution is to occur under RTX4) or by the OS4 loader (if execution is to occur under OS4).

This sequence, portrayed in Figure 1-1, prepares the program for execution under OS4 or RTX4.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1978 COMPUTER AUTOMATION, INC.

The FORTRAN program is coded, then stored on a suitable input medium (cards, paper tape, or magnetic file).

Source is input to the compiler, which manipulates and converts it to object format using an intermediate scratch file.

After converting the source, the compiler generates listings (source and object) and maps (allocation and subroutine usage) to the list device, and the compiled binary code to the output device

The compiler-generated program is input to the OS4 link program, which links it to the required library routines.

The linked binary code is output in standard loadable format.

The linked program is loaded (using a standard loader) and executed.

2-1

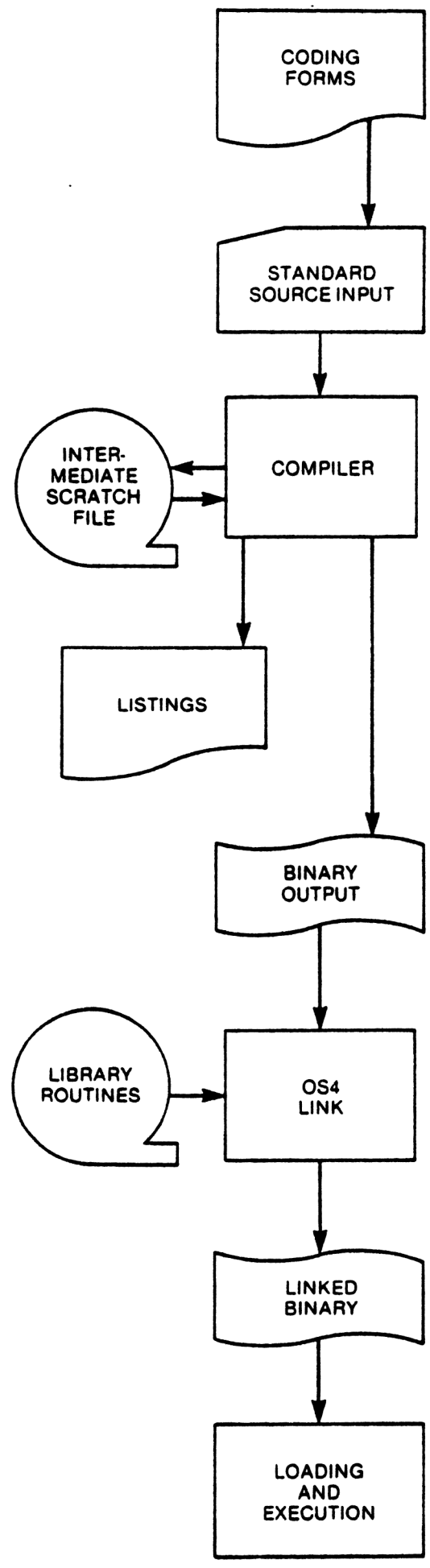


Figure 1-1. FORTRAN Program Development Sequence

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1978 COMPUTER AUTOMATION, INC.

1.2 OPERATING ENVIRONMENT

The following hardware represents the minimum required to compile, link, load, or execute a FORTRAN program:

- NAKED MINI 4 Family processor with 64K words of read/write memory.
- Dual diskette drives for loading the FORTRAN product and OS4 diskettes.
- A disk or diskette labeled using the OS4 LABEL utility.
- I/O Distributor capacity as required by input/output (I/O) channels used.

This minimum configuration can be augmented for improved performance. The system assumed for examples in this manual includes these items, plus a Teletype or CRT, card reader, and line printer.

The software required to compile a FORTRAN program is:

	Diskette ID Number(s)	Model Number
• OS4 system diskette:	F46001	93460-01
• NAKED MINI 4 FORTRAN system diskette	F58002	45066-00

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

SECTION 2

THE FORTRAN COMPILER

2.1 INTRODUCTION

When a FORTRAN source program is created on a suitable medium, it is converted to object code by the FORTRAN compiler. To accomplish this conversion, the compiler:

1) reads each source record (FORTRAN statement) from the input device,

2) converts each source record into intermediate program code using a system scratch file, and

3) writes the linkable object code to the output device.

The compiled program is not directly executable. Before it can be executed, the program must be linked to the needed library routines, converted into loadable format, and loaded by a suitable loader. These procedures are described in subsequent sections of this manual.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1978 COMPUTER AUTOMATION, INC.

2.2 COMPILER ORGANIZATION AND OPERATION

In reading source input, the FORTRAN compiler accepts each series of FORTRAN statements terminated by an END statement as a distinct program. Programs are processed one at a time, in batch mode; processing continues until an end-of-file condition is detected on the input medium. In processing programs, the compiler uses all of the memory available to it and dynamically allocates its working storage tables.

The FORTRAN compiler is comprised of four modules. One of these modules resides in memory from the time the compiler is invoked until the compilation is complete. The other three modules are brought into memory as overlays as they are required. Each of these overlayed modules corresponds to a distinct phase of compiler operation: the scan phase, the allocate phase, and the generation phase, described below. For each program it reads from the input device, the compiler invokes these three phases.

The compiler listings mentioned in this subsection are described in Subsection 2.5.

2.2.1 Scan Phase

During the scan phase, the compiler reads each record of the FORTRAN source program. It builds symbol tables in its working storage area, then writes the source program listing and any syntax error messages to the listing device and the intermediate program code to the system scratch device. The scan phase is terminated when a FORTRAN END statement is encountered.

2.2.2 Allocate Phase

During the allocate phase, the system allocates storage for program variables, using the symbol tables created during the scan phase. An allocation map is output to the listing device, as are error messages for undefined labels and conflicting COMMON or EQUIVALENCE statements. The compiler then generates a cross-reference listing, indicating where each statement label and variable is referenced.

2.2.3 Generation Phase

During the generation phase, the system processes the intermediate program code stored on the system scratch device during the scan phase. It uses the information gathered during the allocate phase to generate object code to the output device. The object listing (if requested), subprogram usage map, statement location list, and program size information are output to the listing device.

The compiler does not generate object code one statement at a time. It "remembers" computations and register contents from previous statements in the current block. A block ends when the compiler encounters a statement label which receives control via a program jump, and in other situations under the compiler's control (e.g., when the allocated memory space is full).

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1978 COMPUTER AUTOMATION, INC

2.3 COMPILER LOGICAL UNIT ASSIGNMENTS

The FORTRAN compiler makes use of OS4 logical units as follows:

- The System File (SF) unit supplies the compiler. All four compiler modules must reside on a single file-type device.

- The Standard Input (SI) unit supplies the FORTRAN source code to be input to the compiler. The Standard Output (SO) unit receives the object code output by the compiler. Normally, SI and SO are assigned in the command which invokes the compiler. (See Subsection 2.4.) If the command line explicitly specifies that object code is not to be produced, the SO unit assignment is not required.

- The System Scratch (XX) unit is used to store the intermediate code generated during the compiler's scan phase. Files created on the XX unit are erased automatically when the compilation is completed. XX should be assigned to the device having the most available space.

- The List Output (LO) unit receives the source listing and other compiler listings.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1978 COMPUTER AUTOMATION, INC.

2.4 INVOKING THE COMPILER: THE /FORTRAN COMMAND

The /FORTRAN command instructs OS4 to load and execute the FORTRAN compiler. The command:

1) identifies the source medium or media, i.e., the file(s) and/or device(s) containing the text to be compiled;

2) specifies any compiler options to be invoked; and

3) defines the destination media, i.e., the file or device to which the object code is to be written and the file or device to which the compiler listing is to be written.

2.4.1 Command Line Conventions

The programmer has considerable flexibility in how he supplies this information. Various ways of issuing the /FORTRAN command are described individually on subsequent pages. The following rules apply to all forms of the command:

1) Normally, the compiler does not prompt the user for input. After the requested operation is performed, control returns to OS4. If the compiler is unable to perform the requested operation, it generates an appropriate message to the OC unit and then returns control to OS4. However, if the programmer wishes to enter symbolic text to be compiled from his terminal rather than from a file or storage device, he can do so by specifying that terminal as the source medium. In this case, the compiler generates the prompt:

    <FOR>

    to the terminal when it is ready to accept input. The programmer can then enter the symbolic text. To signal the end of the text, he enters a /* command. (See the OS4 System User's Manual.)

2) If the command line ends with a plus sign (+) or a comma (,), the compiler generates the prompt:

    FOR..

    to the CI unit to indicate that it is ready to accept additional input. The programmer can then enter a continuation of the command from the CI unit. When the programmer enters a line which does not end with a plus sign or comma, the compiler does not prompt for further command input. Any number of continuation lines may be entered in this manner.

3) The file name extension is optional in a file identifier specified as a source medium. If the extension is omitted, the compiler searches for the .FTN version. If that version is not found, an error occurs.

4) The destination file name is optional. If omitted, the compiler writes the object code to a file having the same name as the first source medium, but using the extension .OBJ.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1978 COMPUTER AUTOMATION, INC.

5) If a destination medium is a file, the command overwrites any current contents of that file. The command has no effect on source media.

6) If no END statement appears in the source text upon end-of-file, the compiler generates the message *** NO END STATEMENT *** to the OC unit and continues processing as if an END statement were found.

7) The compiler must reside on the SF unit.

2.4.2 Command Options

The programmer can specify any of the following options, separated by commas and enclosed in parentheses, in the /FORTRAN command. Options may be abbreviated to the minimum number of characters required to uniquely identify the word (underlined in the following discussion). If an option is not specified, the default option applies.

LISTING[=[file]]	generates a compiler listing. If only the word "LISTING" appears, the listing is generated to the LO unit (default). If an equals sign also appears, the listing is generated to the .LST version of the first source medium, which must be a file. If a file identifier also appears, the listing is generated to that file. (In this case, the file name extension is optional; by default, the listing is generated to the .LST version of the file.)
NOLISTING	does not generate a compiler listing.
ERRORSONLY	includes only error lines in the compiler listing.
NOERRORSONLY	includes statement lines and error lines in the compiler listing (default).
OBJECT	produces object code (default).
NOOBJECT	does not produce object code.
LISTOBJECT	generates a pseudo-assembler listing of the object code to the LO unit.
NOLISTOBJECT	does not produce a listing of the object code (default).

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

XON — ignores an X appearing in column 1 and includes the line in the compilation. This option is useful during debugging. For example, the programmer can include output statements such as:

```
X 200 WRITE (*,*) 'REACHED LINE 200'
```

as an aid in tracing the program's execution. When the program is completely debugged, and a final compilation is performed without the XON option, these statements remain as historical references.

NOXON — treats any line containing an X in column 1 as a comment (default).

XREF — generates a cross-reference listing during the allocate phase. (See Topic 2.5.3.)

NOXREF — does not generate a cross-reference listing (default).

ANSI — allocates two words (instead of one) for each integer and logical variable in the program. The second word of each variable is not used; thus, INTEGER*2 with the ANSI option is not the same as INTEGER*4.

This option is used when a program requires that storage allocation be ANSI-compatible; the ANSI standard specifies that integer, logical, and real quantities be the same size. In most instances, this option has no adverse effect on program operation, with two exceptions. It is recommended that any operation which steps through each word of memory (e.g., internal READ or internal WRITE) not be used on an integer or logical buffer or array when the ANSI option is used. It is also recommended that any programs interfacing to an ANSI program also be ANSI-compatible to avoid any conflicting COMMON variables which are integers or logicals.

NOANSI — allocates one word each for all integer and logical variables (default).

PAGESIZE=nn — overrides the default lines per page for the listing, where nn is the desired decimal number of lines and nn is smaller than the default value. The default value is specified at system generation time.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

LINEWIDTH=nnn overrides the default number of characters per line for the listing (default is 120), where nnn is the desired decimal number of characters per line. If a user specifies a number in access of this limit, the system defaults to 133.

The printer line size for compile:

1. If output is going to a non-directory device (printer, CRT), the definition block of the device (specified at SYSGEN) is inspected and the actual physical size is used.

2. If output is going to a dirctory device (disk) the actual size cannot be determined. The size to be used is determined in the following sequence:

   a. The line size specified on the command line (LINEWIDTH=nnn) or 133, whichever is smaller, if any.

   b. The line size is 120 if LINEWIDTH=nnn is not specified.

The printer line size for runtime:

1. If output is going to a non-directory device (printer, CRT), the definition block for the device (specified at SYSGEN) is inspected and the actual physical size is used.

2. If output is going to a directory device (disk) the actual size cannot be determined. The size to be used is determined in the following sequence:

   a. The record size as specified on the OPEN command, if any.

   b. A default value of 120. If the user wants to change this value, it has a global symbol called F:RDLW, which can be patched. Values larger than 133 will change to 133.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

TRACE — generates extra runtime calls in the compiled program, causing trace information to be output to the SO unit.

Whenever a labeled statement is reached, the message:

```
program LINE line
```

is generated, where "program" is the name of the program containing the statement and "line" is the source line number of the statement about to be executed. The program name is not generated if it is the same as the program name in the previous trace line.

Whenever a subroutine or function is entered, the message:

```
subprogram ENTRY
```

is generated, where "subprogram" is the name of the subroutine or function. Tracing is invoked on entry, not on the call to the subprogram. Thus, only subprograms compiled in TRACE mode are traced.

Whenever a RETURN statement is encountered, the message:

```
program RETURN LINE line
```

is generated, where "program" and "line" are as defined above.

This information is sufficient to follow the flow of the program, since it causes the tracing of all jumps (the transfer point is labeled), all calls except those to library routines (which are assumed to operate correctly), and to subprograms not compiled in TRACE mode (which are also assumed to operate correctly). It is not necessary that all of the programs loaded be compiled in TRACE mode.

NOTRACE — does not generate extra runtime calls to produce tracing information (default).

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

2.4.3 Command Formats

The /FORTRAN command may be followed by any number of parameters specifying the medium from which FORTRAN source code is to be taken, the medium to which the binary object code is to be output, and the compiler options to be invoked. The following paragraphs detail some of the parameter combinations that may be used in the /FORTRAN command.

Form 1: /FORTRAN medium(options)

This form of the /FORTRAN command causes FORTRAN source code to be input from, and object code output to, the same device. The compiler options and associated parentheses may be omitted.

Example: /FORTRAN PROGRAM(XREF)

This command causes source code to be read from the file named PROGRAM. Since no unit or device is specified, and since the file extension is not given, the system searches the UF device for the file with an extension of .FTN. (If none is found, an error occurs.) Object code is output to file UF.PROGRAM.OBJ. Source code and cross-reference listings are output to the LO device.

Form 2: /FORTRAN medium1=medium2(options)

This form of the command causes FORTRAN source code to be input from one device or file (medium2) and the object code to be output to another (medium1).

Example: /FORTRAN PP00=PROGRAM(XREF)

This example is like the Form 1 example except that the object code is output to the paper tape punch rather than to the default output file.

Form 3: /FORTRAN [medium1=]medium2(options)...+mediumn(options)

This form of the /FORTRAN command is the same as Form 2 or 3, except that the additional medium3 and medium4 constructions cause further input of FORTRAN source text to be taken from an additional medium when the preceding input medium is exhausted.

Example: /FORTRAN PP00=PROGRAM(XREF)+DF1.FORT(ANSI)

In this example, FORTRAN source code is read from UF.PROGRAM.FTN as in the preceding example. When an end-of-file is found, additional source code is input from DF1.FORT.FTN with the ANSI option in effect. The object code is output to the paper tape punch and a source listing is output to the LO device.

Form 4: /FORTRAN [medium=]medium(options)[+medium(options)]...},...

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

This command form allows multiple compilations to be made using only one /FORTRAN command. Several parameter groupings may be combined as shown in the following example.

```
Example:  /FORTRAN DM1.TEST1=INPUT1.FTN(XREF,XON),
          DM1.TEST2=INPUT2A(LISTOB,XR)+
          INPUT2B(XR)
```

In this example, source is input from UF.INPUT1.FTN and object generated on DM1.TEST1.OBJ, with a cross-reference listing and lines beginning with X compiled. A second object file is generated to DM1.TEST2.OBJ from two source files: UF.INPUT2A.FTN, for which an object listing and cross-reference listing are printed, and UF.INPUT2B.FTN, for which a cross-reference listing (but no object listing) is printed.

Form 5: /FORTRAN

The FORTRAN command with no input or output media specified causes source code to be input from the SI device and object code to be output to the SO device.

```
Example:  /ASSIGN SI=DF.PROG.FTN
          /ASSIGN SO=DF1.PROG.OBJ,LO=LP
          /FORTRAN
```

In the above example sequence, the programmer enters a /FORTRAN command without parameters after making required unit assignments. This sequence causes FORTRAN statements to be input from DF.PROG.FTN. Object code is output to DF1.PROG.OBJ. A source listing is output to the line printer.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1978 COMPUTER AUTOMATION, INC.

2.5 COMPILER LISTINGS

A complete listing of a compiled FORTRAN program consists of five parts:

- source listing
- variable storage allocation listing
- cross-reference listing
- object listing
- summary

If no listing options appear in the /FORTRAN command, only the source listing, variable storage allocation listing, and summary are produced. The LISTOBJECT option causes the object listing to be produced, and the XREF option causes the cross-reference listing to be produced. If the ERRORSONLY option is specified, the source listing is suppressed except for the first line and any lines containing errors.

Figure 2-1 provides an example of a complete FORTRAN compiler listing. To allow the associated text to appear adjacent to each of the five parts, the figure is presented in sections: 2-1(1) through 2-1(5). In actual practice, the listing would appear on contiguous pages.

2.5.1 Source Listing

The source listing shows each source line, preceded by a decimal line number beginning with 0001. One space separates the line number and the first column of the source line. Every line is numbered, including continuation lines and comments. If ERRORSONLY is requested, the first source line is automatically output, and the correct line number is shown for any error source lines. Error messages may be interspersed, as shown after line 0022 of the sample listing in Figure 2-1(1). Note that each such message is followed by a string of E's (or W's if the message is a warning) and asterisks. See Appendix B for a list of the FORTRAN error/warning messages.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

```
PAGE    1   1981/03/12 10:46:58.75    C
FORTRAN(F000)     FTNEXAMPL  .FTN

 0001 C
 0002 C       FORTRAN LISTING EXAMPLE PROGRAM
 0003 C
 0004         INTEGER NN(25),LL(10)
 0005         LOGICAL LOG
 0006         DOUBLE PRECISION DX,DY
 0007         COMMON MM(100),M/BLK/Y
 0008         EQUIVALENCE (L,LL)
 0009         ISF(KD)=KD*8
 0010 10      K=(L+300)*M-74
 0011         MM(I)=K
 0012         X=ABS(Y+4)
 0013         DX=DABS(DY/4.3)
 0014         IF(DX.LT.0) GO TO 70
 0015         CALL SUB(L+300,7HABCDE  ,Y+4)
 0016 20      WRITE (6,30) Y
 0017 30      FORMAT (5X,I5,' VALUES.')
 0018         IF (K.EQ.M) GO TO 10
 0019 50      DO 60 I=1,10
 0020 60      MM(I)=-1
 0021         ASSIGN 40 TO K
 0022         MN(3)=0
               $
01) UNDIMENSIONE) E*E*E*E*F*E*E*E*E*E*E*E*F*E*E*E*E*E*E*F*E*E*F*F*E*E*E*E*E  (0000)

 0023 70      STOP
 0024         END
```

Figure 2-1(1). Sample Compiler Listing (Source Listing)

REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

2.5.2 Variable Storage Allocation Listing

Several types of tables can appear in this listing, depending on the variables used in the program and their allocation. A table heading appears only if there are items to appear in the table. Variables in each table are listed in storage order, except those in EQUIVALENCE, which may appear out of order. Local arrays and scalars are always listed in order.

If any variables are allocated in COMMON, a storage map appears for each COMMON block, including blank COMMON, labeled BLANK:. Each COMMON storage map includes the name of the block and its size (i.e., the number of words occupied) in hexadecimal. Next, the location (in hexadecimal), name, type, and size (in decimal), of each variable in the block are listed.

After the COMMON storage maps (if any), a table of all local scalar variables (i.e., scalar variables not in COMMON and not EQUIVALENCEd) is listed, followed by a table of any local variables (arrays or scalars) in EQUIVALENCE.

Finally, any local (i.e., non-COMMON) arrays not in EQUIVALENCE are listed. This table includes the same items as a COMMON storage map.

The compiler always generates at least two rel areas, one for instructions and one for data. (In this example, two additional areas are generated, one for blank COMMON and one for labeled COMMON.) Thus, it may appear that variable locations overlap with each other and with instructions. For example, location :000F appears to be used for three different purposes:

- The 16th element of the COMMON array MM (see page 2-13)
- The variable K (see page 2-13)
- The third word of the object code generated for the FORTRAN statement on line 9 (see page 2-20)

Actually, however, these represent relative location :000F in three different rel areas; each rel area is listed separately in the map generated by the linker.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

```
PAGE    2   1981/03/12 10:47:05.50    C
FORTRAN(F000)     FTNEXAMPL  .FTN


 UNDEFINED LABELS E*E*E*E*E*E*E*E*E*E*E*E*F*E*E*E*E*E*F*E*E*E*E*E*E*E*E*E*E*  (0022)

    40 FIRST REF AT LINE     21

 COMMON BLOCK/BLANK:/ ALLOCATION  0065 WORDS

 LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MOD

 0000 MM         100 ARRAY     INTEGER*2  0064 M            1 SCALAR    INTEGER*2

 COMMON BLOCK/BLK    / ALLOCATION  0002 WORDS

 LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MOD

 0000 Y            2 SCALAR    REAL


 SCALAR ALLOCATION

 LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MOD

 000F K            1 SCALAR    INTEGER*2  0010 I            1 SCALAR    INTEGER*2  0011 X            2 SCALAR    REA
 0013 DX           4 SCALAR    DOUBLE     0017 DY           4 SCALAR    DOUBLE


 EQUIVALENCE ALLOCATION

 LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MOD

 0018 L            1 SCALAR    INTEGER*2  001B LL          10 ARRAY     INTEGER*2


PAGE    3   1981/03/12 10:47:09.25    C
FORTRAN(F000)     FTNEXAMPL  .FTN


 ARRAY ALLOCATION

 LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MODE       LOCN NAME     WORDS TYPE      MOD

 0025 NN          25 ARRAY     INTEGER*2
```

Figure 2-1(2). Sample Compiler Listing (Variable Storage Allocation Listing)

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

©1980 COMPUTER AUTOMATION, INC.

2.5.3 Cross-Reference Listing

A cross-reference listing is generated only when the XREF option is specified in the /FORTRAN command. This listing includes each statement label number, variable name, function name, and subroutine in the program, and where that item is used. Statement label numbers are listed first, in numeric order, followed by the names, listed alphabetically.

Next to each item is a list of line numbers; these lines contain references to the item. The number of the line in which the item is defined is enclosed in slashes (/); a line number at which the item's value changes is enclosed in asterisks (*).

Following the cross-reference is a list of variable names which are referenced, but which previously have not been stored into or initialized (e.g., a variable name on the righthand side of an assignment statement).

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

```
PAGE    4   1981/03/12 10:47:10.25   C
FORTRAN(F000)    FTNEXAMPL  .FTN

 CROSS REFERENCE

    10     /0010/    0018
    20     /0016/
    30      0016    /0017/
    40      0021
    50     /0019/
    60      0019    /0020/
    70      0014    /0023/
 ABS        0012
 BLK       *0007*
 DABS       0013
 DX        /0006/   *0013*    0014
 DY        /0006/    0013
 I          0011    *0019*    0020
 ISF       *0009*
 K         *0010*    0011     0018    *0021*
 KD         0009     0009
 L         *0008*    0010     0015
 LL        /0004/   *0008*
 LOG       /0005/
 M         *0007*    0010     0018
 MM        *0007*   *0011*   *0020*
 MN        *0022*
 NN        /0004/
 SUB        0015
 X         *0012*
 Y         *0007*    0012     0015     0016

UNSET VARIABLES

DY
NN
```

Figure 2-1(3). Sample Compiler Listing (Cross-Reference Listing)

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

©1980 COMPUTER AUTOMATION, INC.

2.5.4 Object Listing

An object listing is generated only when the LISTOBJECT option is specified in the /FORTRAN command. An object listing includes all source lines, even those suppressed in the source listing by the ERRORSONLY option, if specified.

In examining the object code produced for an individual statement, remember from Section 2.2 that the compiler does not generate object code one statement at a time: it "remembers" computations and the contents of the registers from previous statements within a block. Thus, the code for one statement may appear to be incomplete, since it is making use of results from previous statements. See, for example, source line 0011, which "remembers" that the value of K is left in the A register by statement 0010.

The allocation of variables is shown in the allocation maps, so it is not reproduced in the object listing. The FORMAT statements, although generated apart at the head of the program, are listed where they appear in the source program. Here the program is not listed in strict forward order (i.e., the memory locations are not listed sequentially). Another place is the backward origins, as described in subheading 3 below. For the most part, however, the program is listed in forward order, beginning at the entry point and ending at the last temp or constant.

Each line of object code listed consists of seven parts. From left to right, the first three are:

1) The hexadecimal location counter.

2) The hexadecimal representation of the generated word, which may be an instruction or a data value. In many cases, this is only a skeleton word, since the actual address is not known at the time it is listed. These situations include references to COMMON, externals, and most forward locations. Also, an instruction may turn out to be indirect through a literal pool pointer, even though it is not listed that way. A second word appears on lines which contain double word instructions.

   Note that instructions which reference FORTRAN variables are normally Y-relative instructions. For example, line 10, shown on page 2-20, causes the value of an expression to be stored into the integer variable K. Under that statement, location :001A shows the actual COPY instruction which accomplishes this. The instruction translates into machine language as :844F, which is equivalent to

```
COPY    A,:F(Y)
```

   In other words, the variable K is located at relative location :F within the data rel area (as shown on page 2-13), and the Y register always points to the start of that data rel area. Note that this is a different location than location :F in the program rel area, which is within the instruction sequence generated for source statement 9 (page 2-20).

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

3) The optional letter F or B, which indicates operand addressing manipulation. When an instruction references an operand whose location is indeterminate because it has not yet been encountered (called a forward reference), the letter F is printed, and the machine language instruction is shown as a skeleton only. (See location :000D, the JMP #M7 instruction.) Further down in the listing, at location :0013, #M7 is defined. A backward origin then appears (see B, next line) to location :000D again, and the final contents of the instruction (:9E85) are shown, with the JMP #M7 instruction repeated.

The next four items in the line are parts of a simulated assembly language listing of the instruction. It is not always possible to list the instruction exactly as it would appear in assembly language, but in most cases the representation is very close and makes it clear what the compiler is doing.

4) Label field. For normal instructions (i.e., not temps, constants, or literal pools), two types of labels can appear:

#n	Statement number from the source program. (For example, see location :0047,label 20).
#Mn	"Made" label, an internal transfer point generated by the compiler (e.g., location :000E).

Other types of labels can appear in special places:

#ICn	Integer constant (e.g., location :0060).
#RCn	Real (or double precision or complex) constant (e.g., location :0061).
#HCn	Hollerith constant (e.g., location :0076).
#L	Literal pool (not shown).

5) Op-code field. All op-codes are either standard assembler mnemonics or slightly modified versions, such as SHIFTL, SHIFTR.

6) Operand field. Where appropriate, this field may begin with * (indirect).

a. Blank. For op-codes like RSK that have no operand (e.g., location :0012).

b. Decimal value, optionally preceded by minus sign (e.g., location :0060), or shown as a literal (location :0019).

c. Hexadecimal value, always preceded by a colon (e.g., location :0030).

d. Alphanumeric string, enclosed in quotes (e.g., location :0076).

e. #n statement label (e.g., location :004A).

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1978 COMPUTER AUTOMATION, INC.

f. #Tn (Temp, e.g., location :002A); #Mn ("made" label, e.g., location :000D); #ICn (integer constant, e.g., location :0036); #RCn (real constant, e.g., location :001D); or #HCn (Hollerith constant, e.g., location :0045).

g. Hardware register (e.g., A, AQ, X, as in location :001D).

h. FORTRAN variable name; normally shown indexed by the Y register since FORTRAN always generates data in a separate REL area which must be accessed through Y (e.g., location :001A). Sometimes the variable name is preceded by a "@" character (e.g., location :001C). This signifies that an element of an array (MM in this case) is being accessed, and that the operand being referenced is not really the start of the array itself (MM), but rather a pointer to that array minus an addressing offset (@MM).

i. Special system (or runtime) name, which always contains a colon (e.g., location :000E).

7) Comment field. When a numeric constant is referenced, its hexadecimal value is shown in the comment field (e.g., location :0013).

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

```
                 000D  FE00          F          JSK        F:RSTU
 * 0001 C
 * 0002 C       FORTRAN LISTING EXAMPLE PROGRAM
 * 0003 C
 * 0004         INTEGER NN(25),LL(10)
 * 0005         LOGICAL LOG
 * 0006         DOUBLE PRECISION DX,DY
 * 0007         COMMON MM(100),M/BLK/Y
 * 0008         EQUIVALENCE (L,LL)
 * 0009         ISF(KD)=KD*8
                 000E  9E00          F          JMP        #M7
                 000F  FE00          F #MM      JSK        F:RDMY
                 0010  0007                     WORD       KD
                 0011  8147                     COPY       *KD(Y),A
                 0012  0E21                     SHIFTL     A,3
                 0013  2309                     RSK
 * 0010 10      K=(L+300)*M-74
                 0014                  #M7      EQU        :0014
                 000E  9E85          B          JMP        #M7
                 0014  8200          F #10      COPY       #IC1,A          :012C
                 0015  8858                     ADD        L(Y),A
                 0016  4FF9                     SHIFTR     A0,16
                 0017  4750  0000               MUL        *M(Y),A0
                 0019  0040                     COPY       0,A
                 001A  0A96                     SUB        =74,A
                 0018  844F                     COPY       A,K(Y)
 * 0011         MM(I)=K
                 001C  A050                     COPY       I(Y),X
                 001D  85CA                     COPY       A,*@MM(X,Y)
 * 0012         X=ABS(Y+4)
                 001E  4786  0000    F          COPYE      #RC1,A0         :4180 :0000
                 0020  5C50  000E               ADDF       *Y(Y)
                 0022  00FD                     RBIT       15,A
                 0023  4717  0011               COPYE      A0,X(Y)
```

Figure 2-1(4). Sample Compiler Listing (Object Listing, Page 1 of 5)

ComputerAutomation®

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

Revision F1

```
 * 0013          DX=DABS(DY/4.3)
                  0025  2917                        COPY    =23,X
                  0026  2062                        ADD     Y,X
                  0027  FE00          F             JSK     F:RDLD
                  0028  2784  0000    F             CEA     #RC2,X          :4189 :4999 :9999 :999A
                  002A  FE00          F             JSK     F:RDDV
                  002B  2714  003E                  CEA     #T0(Y),X
                  002D  FE00          F             JSK     F:RDST
                  002E  FE00          F             JSK     DABS
                  002F  0001                        WORD    1
                  0030  8000                        WORD    :8000
                  0031  003E                        WORD    #T0
                  0032  2913                        COPY    =19,X
                  0033  2062                        ADD     Y,X
                  0034  FE00          F             JSK     F:RDST
 * 0014          IF(DX.LT.0) GO TO 70
                  0035  1380          F             JLT     A,#M9
 * 0015          CALL SUB(L+300,7HABCDEF   ,Y+4)
                  0036  8200          F             COPY    #IC1,A          :012C
                  0037  8858                        ADD     L(Y),A
                  0038  847E                        COPY    A,#T0(Y)
                  0039  4786  0000    F             COPYE   #RC1,AQ         :4180 :0000
                  003B  5C50  000E                  ADDF    *Y(Y)
                  003D  4717  003F                  COPYE   AQ,#T1(Y)
                  003F  A04C                        COPY    SUB(Y),X
                  0040  FE00          F             JSK     SUB
                  0041  0005                        WORD    5
                  0042  8800                        WORD    :8800
                  0043  003E                        WORD    #T0
                  0044  0000          F             WORD    #HC0
                  0045  003F                        WORD    #T1
 * 0016 20       WRITE (6,30) Y
                  0046  0906              #20       COPY    =6,A
                  0047  FE00          F             JSK     F:RWEF
```

Figure 2-1(4). Sample Compiler Listing (Object Listing, Page 2 of 5)

ComputerAutomation®

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
PAGE    7   1981/03/12 10:47:27.50    C
FORTRAN(F000)     FTNEXAMPL  .FTN

                 0048  1000                                WORD      :1000
                 0049  0004                                WORD      #30
                 004A  804E                                COPY      MY(Y),A
                 004B  FE00          F                     JSR       F:RROL
                 004C  FE00          F                     JSR       F:WETF
 * 0017 30      FORMAT (5X,I5,' VALUES.')
                 0004  2835                      #30       BYTE      '(5X,I5,' VALUES.')'
 * 0018         IF (K.EQ.M) GO TO 10
                 004D  804F                                COPY      K(Y),A
                 004E  8D4D                                SUB       *M(Y),A
                 004F  1100          F                     JEQ       A,#M10
                 0014                            #M10      EQU       :0014
                 004F  1104          3                     JEQ       A,#M10
 * 0019 50      DO 60 I=1,10
                 0050  2901                      #50       COPY      =1,X
                 0051  A450                      #M11      COPY      X,I(Y)
 * 0020 60      MM(I)=-1
                 0052  08FF                      #60       COPY      =-1,A
                 0053  85CA                                COPY      A,*@MM(X,Y)
                 0054  2801                                ADD       =1,X
                 0055  0020                                COPY      X,A
                 0056  0AF6                                SUB       =10,A
                 0057  1289                                JLE       A,#M11
 * 0021         ASSIGN 40 TO K
                 0058  8200          F                     COPY      #40,A
                 0059  844F                                COPY      A,K(Y)
 * 0022         MN(3)=0
 *               $
 * 01) UNDIMENSIONED E*E*E*F*E*E*E*E*E*E*E*F*E*E*F*F*F*F*F*F*F*E*E*E*F*E*F*F*F*E*E*F   (0000)
                 005A  FE00          F                     JSR       F:RERR
                 005B  0016                                WORD      22
 * 0023 70      STOP
                 005C                            #70       EQU       :005C
                 005C                            #M9       EQU       :005C
```

Figure 2-1(4). Sample Compiler Listing (Object Listing, Page 3 of 5)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
                0035  13E6          B        JLT       A,#M9
                005C  0900                     COPY      =0,A
                005D  FE00          F        JSK       F:RSTO
 * 0024         END
                0014  82C9          B        COPY      #IC1,A          :012C
                0036  82A7          B        COPY      #IC1,A          :012C
                005E  012C            #IC1   WORD      300
                001E  4786  003F    B        COPYE     #RC1,AQ         :4180 :0000
                0039  4786  0024    B        COPYE     #RC1,AQ         :4180 :0000
                005F  4180            #RC1   WORD      16768
                0060  0000                     WORD      0
                0028  2784  0037    B        CEA       #RC2,X          :4189 :9999 :9999 :999A
                0061  4189            #RC2   WORD      16777
                0062  9999                     WORD      -26215
                0063  9999                     WORD      -26215
                0064  999A                     WORD      -26214
                000D  FF07          B        JSK       *F:RSTO
                0065  0000                     WORD      F:RSTO
                000F  FF06          B        JSK       *F:RDMY
                0066  0000                     WORD      F:RDMY
                0027  FFBF          B        JSK       *F:RDLD
                0067  0000                     WORD      F:RDLD
                002A  FFBD          B        JSK       *F:RDDV
                0068  0000                     WORD      F:RDDV
                002D  FFBB          B        JSK       *F:RDST
                0034  FFB4          B        JSK       *F:RDST
                0069  0000                     WORD      F:RDST
                002E  FFB3          B        JSK       *DABS
                006A  0000                     WORD      DABS
                0040  FFAA          B        JSK       *SUB
                006B  0000                     WORD      SUB
                0047  FFA4          B        JSK       *F:RWEF
                006C  0000                     WORD      F:RWEF
                0048  FFA1          B        JSK       *F:RROL
```

Figure 2-1(4). Sample Compiler Listing (Object Listing, Page 4 of 5)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.
© 1980 COMPUTER AUTOMATION, INC.

```
006D  0000                    WORD    F:RROL
004C  FFA1          B         JSK     *F:RETF
006E  0000                    WORD    F:RFIF
0058  8296          B         COPY    #40,A
006F  0000          F         WORD    #40
005A  FF95          B         JSK     *F:RERR
0070  0000                    WORD    F:RERR
005D  FF93          B         JSK     *F:RSTO
0071  0000                    WORD    F:RSTO
006F  0072          B         WORD    #40
0072  FF7D            #40     JSK     *F:RFRR
0073  0000          F         WORD    #RC4                :0000 :0028
0074  0007                    WORD    7
0044  0075          B         WORD    #HC0
0075  4142            #HC0    WORD    'AB'
0076  4344                    WORD    'CD'
0077  4520                    WORD    'E '
0078  2020                    WORD    '  '
0073  0079          B         WORD    #RC4                :0000 :0028
0079  0000            #RC4    WORD    0
007A  0028                    WORD    40
```

Figure 2-1(4). Sample Compiler Listing (Object Listing, Page 5 of 5)

REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

ComputerAutomation®

2.5.5 Summary

The last element of a compiler listing is the summary. Appearing first in the summary are the subprograms called by the program, including functions and subroutines referenced explicitly by the program, as well as runtime routines referenced by the generated object code (e.g., for input/output). Names referenced by the program are FORTRAN names, i.e., beginning with a letter and containing only letters and digits. Runtime routines are non-FORTRAN names because they always contain a colon (e.g., F:RWF, F:RREL).

The table shows first the name of the subprogram. Next is the type (e.g., REAL, INTEGER) if it is a FORTRAN-referenced subprogram; the number of arguments in the subroutine call appears next. If the number of arguments is variable or unknown (name declared external but not directly called), the number of arguments is shown as zero.

With the exception of intrinsic functions, this list of subprograms called represents the names that must be found during loading, either from the library or from user-supplied programs. Intrinsic functions (e.g., ABS) are listed here but are not actually referenced externally. They are generated in-line.

Second in the summary is a map of the statement labels. This map includes the statement numbers used in the source program and also the "made" labels generated by the compiler (#Mn). They appear in the order defined or referenced in the object program, which is not necessarily storage order. Each entry contains the hexadecimal location, the label, and in certain cases, one of the following usage indicators:

FORMAT - denotes the label of a FORMAT statement.
DO END - denotes a label used only as the terminus of a DO.
UNUSED - denotes a label defined on a statement but never referenced.

Finally, five pieces of information are given about the program:

- The location (in hexadecimal) of the entry point.
- The total sizes (in hexadecimal) of the program Rel area and the data Rel area, including local variables but not COMMON.
- The number (in decimal) of long (extended memory reference) instructions used.
- The message:

  COMPILATION COMPLETE

  followed by the number of errors and warnings (even if zero).

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

```
 SUBPROGRAMS CALLED

 NAME     TYPE     ARGS    NAME     TYPE     ARGS    NAME     TYPE     ARGS    NAME     TYPE     ARGS

 ABS      INLINE           DABS     KNOWN    1       SUB      USER     5       F:RWFF   RUNTIME
 F:RROL   RUNTIME          F:RFTF   RUNTIME          F:RERR   RUNTIME          F:RSTO   RUNTIME
 F:RSTU   RUNTIME          F:RDMY   RUNTIME          F:RDLO   RUNTIME          F:RDDV   RUNTIME
 F:RDST   RUNTIME

PAGE   10   1981/03/12 10:47:41.25   C
FORTRAN(F000)    FTNEXAMPL  .FTN

 STATEMENT LABELS

 LOCN  LABEL  USE         LOCN  LABEL  USE         LOCN  LABEL  USE         LOCN  LABEL  USE

 0014  #10               005C  #70               0046  #20    UNUSED      0004  #30    FORMAT
 0050  #50    UNUSED      0052  #60    DO END      FFFF  #40               0014  #M7
 000F  #M8                005C  #M9                0014  #M10              0051  #M11

 STATEMENT LOCATION MAP

 LINE   LOCN  LINE  LOCN  LINE  LOCN  LINE  LOCN  LINE  LOCN  LINE  LOCN  LINE  LOCN  LINE  LOCN

 4      000E  5     000E  6     000E  7     000E  8     000E  9     000E  10    0014  11    001C
 12     001E  13    0025  14    0035  15    0036  16    0046  17    004D  18    004D  19    0050
 20     0052  21    0058  22    005A  23    005C  24    005E

 ENTRY=0000
 PROGRAM SIZE=0073 WORDS
 DATA SIZE=0042 WORDS
 LONG INSTRUCTIONS USED 9
 COMPILATION COMPLETE  2 ERRORS (9999) 0 WARNINGS (0000)
```

Figure 2-1(5). Sample Compiler Listing (Summary)

© 1980 COMPUTER AUTOMATION, INC.

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

SECTION 3

LIBRARY STRUCTURE AND LINKING

3.1 INTRODUCTION

All compiled FORTRAN programs contain references to external subprograms:

1) compiler-generated references to mathematical or input/output (I/O) routines;

2) references to user-generated subroutines, or to basic external or user-generated FUNCTION subprograms; and/or

3) references to RTX4/IOS4 services.

Some of these references can be satisfied by compiling referenced subprograms with the main program in a batch. Other references must be satisfied by linking the compiled FORTRAN program with the appropriate user files and/or library files.

The specific linking procedures to be followed depend on whether the program is to be executed under OS4 or RTX4. This section first describes library structure and use, then outlines the procedures for linking a program to be executed under OS4 and under RTX4.

3.2 LIBRARY STRUCTURE AND USE

The FORTRAN IV product diskette contains two versions of the FORTRAN runtime library, FORTRAN.LIB and FORTRAN.OPT. FORTRAN.LIB is a library of all runtime routines needed to link a FORTRAN program. FORTRAN.OPT is a version of the complete runtime library that uses the floating point firmware available on optioned processors.

FORTRAN.OPT was created for routines that use floating point arithmetic and run on optioned processors. Execution time for these routines is greatly reduced by the use of FORTRAN.OPT rather than FORTRAN.LIB. However, there is a cost of one digit of precision on most calculations.

The FORTRAN.OPT library should not be used on processors that do not have the optional instructions. To do so increases execution time, because the NM4 optional Instruction Emulator must then be used.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

3.3 LINKING FOR EXECUTION UNDER OS4

To link a FORTRAN program for execution under OS4, enter the OS4 /LINK command in the following manner:

1) Assign the unit containing the compiled FORTRAN object program(s) as the primary input file.

2) Include any files containing user-generated FORTRAN or assembly language subprograms as secondary input files.

3) Include FORTRAN.LIB, or FORTRAN.OPT, and OS4.LIB as the final input files, in Library Link (LIBRARY) mode.

The resulting linked program may be loaded under any OS4 system; relinking is not necessary.

For example, the following /LINK command could be used to link a FORTRAN program for execution under OS4:

```
/LINK OS4FORTPROG=FORTPROG+SUB1+FORTRAN+OS4
```

This command produces an executable FORTRAN program on UF called OS4FORTPROG.BIN, from FORTRAN main program FORTPROG.OBJ, FORTRAN subroutine SUB1.OBJ, and FORTRAN library routines in FORTRAN.LIB.

3.4 LINKING FOR EXECUTION UNDER RTX4

To run a FORTRAN program under RTX4, it must be a task. One or more independent FORTRAN programs may be run in this manner. A FORTRAN main program may be converted to a task by adding the TASK statement before the first service line in the main program. Linking a FORTRAN task for execution under RTX4 is similar to linking for execution under OS4 except that more modules are required. See the Multi-Tasking example in the Real-Time Executive User's Manual (93410-90). A single assembly language module is also required which defines the INIT:A, UAT:AA, and ECB:A RTX4 tables.

For RTX4 execution, the FORTRAN and assembly language modules must be linked with the following libraries in the order listed:

FORTRAN library	FORTRAN.LIB
-or-	
Optional FORTRAN library with floating point routines	FORTRAN.OPT
NAKED MINI® 4 Optional Instruction Emulator library (if needed)	EMLTR.LIB
SFM library	SFM.LIB
IOS4 library	IOS.LIB
RTX4	RTX.LIB

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

The following example demonstrates a LINK command line that could be used to link a FORTRAN for execution under RTX4. Note that linking with SFM.LIB is required even if the FORTRAN programs do not use the services of the Standard File Manager (SFM).

```
LINK PROG=STARTUP(AB=100)+FORTTASK(NOLIB)+SUB1+SUB2+FORTRAN.OPT(SYS)+
EMLTR(SYS)+SFM(SYS)+IOS(SYS)+RTX(SYS)
```

This command creates from the following input files an AutoLoadable module called PROG.BIN. It is created on the UF unit.

- FORTTASK.OBJ — FORTRAN task.
- SUB1.OBJ, SUB2.OBJ — FORTRAN or assembly language subprogram object modules.
- STARTUP.OBJ — Assembled start-up routine containing the ECB, UAT and INIT:A
- FORTRAN.OPT — FORTRAN library routines, using the optional hardware floating point instructions.
- EMLTR.LIB — NM4 Instruction Emulator routines.
- IOS.LIB — IOS4 library routines.
- SFM.LIB — Standard File Manager library routines.
- RTX.LIB — RTX4 library routines.

Refer to the OS4 System User's Manual (93460-90) for more detailed information on LINK usage.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

SECTION 4

RUNTIME

4.1 INTRODUCTION

When the FORTRAN program is successfully compiled and linked, it is ready to be loaded and executed. Program execution, however, requires that assignments first be made for input/output (I/O) devices to be used.

During program execution, situations may arise which call for a response from the user.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1978 COMPUTER AUTOMATION, INC.

4.2 RUNTIME LOGICAL UNIT ASSIGNMENTS

All I/O operations specified in the FORTRAN source program make use of logical unit assignments. A FORTRAN I/O statement includes a logical unit number, expressed as an integer constant or a simple integer variable, with two exceptions: the free-form READ and free-form WRITE statements use asterisks to imply input from logical unit 5 and output to logical unit 6.

Prior to program execution, any FORTRAN logical unit numbers used in the program must have specific I/O devices assigned to them, unless specific OPEN and CLOSE statements are included in the program. (See the NAKED MINI 4 FORTRAN IV Language Manual.)

4.2.1 Device Assignments for Execution under OS4

The OS4 /ASSIGN command is used to make device assignments when execution under OS4 is desired. (Assignment is not necessary if the OPEN statement is used.) The standard delivered OS4 contains the following UAT entries:

FORTRAN Unit Number	Logical Unit Name	Physical Device
1	X1	
2	X2	
3	X3	
4	X4	
5	SI	CR00
6	LO	LP00
7	SF	DF00
8	UF	DF01
9	OC	TV00
10	CI	TV00
11	SL	TV00
12	SO	TV00
13	SA	TV00
14	LD	
15	OV	
16	XX	DF01
17	PO	LP00
18	Z1	
19	Z2	
20	Z3	
21	Z4	
22	Z5	
23	Z6	
24	ZP	

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1978 COMPUTER AUTOMATION, INC.

The first column is the FORTRAN unit number, which always corresponds to the order within the table of the logical units shown in the second column. The third column lists the default physical unit assignments to logical units in the second column. Thus, device assignment via the /ASSIGN command consists solely of changing the physical device in the third column. A reference to a FORTRAN unit number is the same as a reference to a logical unit name; that is,

```
/ASSIGN 6 = TV
```

is exactly equivalent to

```
/ASSIGN LO = TV
```

and vice versa. The relationship between corresponding entries in columns 1 and 2 does not change. Since there are only 24 logical units defined in the standard UAT, reference to a FORTRAN unit greater than 24 will be meaningless to OS4 in an /ASSIGN statement, although FORTRAN itself recognizes units 1-99 (e.g., in a WRITE statement).

Note that the command

```
/ASSIGN 4 = LO
```

has meaning; it is effectively the same as

```
/ASSIGN X4 = LP00
```

assuming LO was assigned to LP00 prior to the command.

The FORTRAN runtime outputs error messages to the OC and SL logical units, therefore, the user must ensure these are assigned properly prior to execution.

4.2.2 Device Assignments for Execution under RTX4

When preparing a FORTRAN program for execution under RTX4, device assignments are determined by the RTX4 Unit Assignment Table (UAT), which must be supplied by the user in the start-up module. UAT format is described in the Input/Output Subsystem User's Manual. The sample program listed in Section 5 includes an example UAT.

The FORTRAN runtime outputs error messages to the OC and SL logical units, therefore, the user must provide these two units in the UAT and ensure they are assigned prior to execution.

4.3 PROGRAM LOADING AND EXECUTION

The procedures used to load a linked FORTRAN program depend on whether the program is intended for execution under OS4 or RTX4.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

4.3.1 Loading and Execution under OS4

After making any required device assignments, the user can load and execute the program with an OS4 /file command, where "file" is the name of the linked user's program. Use of the /file command is described in the OS4 System User's Manual. A FORTRAN program linked on an OS4 system may be loaded and executed successfully on any other OS4 system.

4.3.2 Loading and Execution under RTX4

A program prepared for execution under RTX4 is linked to be an absolute, stand-alone program, and is therefore not loadable by the OS4 /file command. Loading for RTX4 execution is performed by either:

Using the OS4 /AUTOLOAD command to load and execute the program, or

Renaming or copying the file to OS:.SYS on a disk device, then using the hardware AutoLoad feature to load and execute the program.

The OS4 System User's Manual contains the detailed information needed to follow these procedures.

4.4 PROGRAM/USER INTERACTION

When the linked FORTRAN program is executed, various conditions may occur to which the user must respond. These conditions are discussed below, except for runtime error conditions. Appendix B lists error messages and recommended actions.

4.4.1 PAUSE Messages

When a PAUSE statement is included in the FORTRAN program, its execution causes the message:

```
PAUSE xxxxx
```

to be written to the Operator Console (OC) and System Log (SL) devices. (The xxxxx parameter is a decimal number which assumes any meaning the programmer desires.)

If the FORTRAN program is running under OS4, program execution is suspended after the message is written. OS4 then generates the prompt:

```
CX>
```

At this point the user may input the character "C" to continue or "X" to abort program execution.

If the FORTRAN program is running under RTX4, program execution is not suspended; the task continues execution after generating the PAUSE message.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

©1979 COMPUTER AUTOMATION, INC.

4.4.2 STOP Messages

When a STOP statement is included in the FORTRAN program, its execution causes the message:

```
STOP xxxxx
```

to be output to the Operator Console (OC) and System Log (SL) devices. (The xxxxx parameter is a decimal number which assumes any meaning the programmer desires.)

If the FORTRAN program is running under OS4, the program is aborted and control reverts to the OS4 Executive.

If the FORTRAN program is running under RTX4, the task in progress is terminated and any resources used by that task are freed. If a STOP statement with a parameter of zero is executed while RTX4 is running, the message is not output.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1978 COMPUTER AUTOMATION, INC.

SECTION 5

INTERFACING TO ASSEMBLY LANGUAGE

5.1 INTRODUCTION

In certain instances the programmer may want to combine the capabilities of both FORTRAN and assembly language. He may do so by writing FORTRAN subprograms or functions in assembly language and calling them from FORTRAN mainline programs, or by calling FORTRAN subprograms from assembly language programs.

This section provides information on the interface between FORTRAN and assembly language in each of these situations.

5.2 ASSEMBLY LANGUAGE SUBPROGRAMS CALLED FROM FORTRAN

This subsection describes the procedures for calling an assembly language subprogram from a FORTRAN main program. A sample calling program (Figure 5-1) and the subprogram it calls (Figure 5-2) are presented for illustration. As in the preceding subsection, references to individual lines are underlined in the text.

The requirements of the calling program are discussed first, followed by the subprogram requirements.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

5.2.1 Calling Program Requirements

The FORTRAN program listed in Figure 5-1 illustrates the call to the assembly language function subprogram on line 11. Note that there is no difference in the FORTRAN module between calling a FORTRAN subprogram and calling one written in assembly language. In this example, the reference is to the assembly language function called MYFUNC, to which are passed two parameters, J and K; the purpose of the function is to add J and K together and return the sum. In this example, the sum is returned in the A Register, because it is an integer result. FORTRAN requires that the result of a function subprogram be returned in specific registers, depending on their data type, as follows:

Function Type	Result in
Integer	A Register
Logical	A Register
Long Integer	AQ Registers (high-order byte in A Register)
Real	AQ Registers (high-order byte in A Register)

Note that Double Precision and Complex types are not shown in the table. This is because results for these types of functions are not returned in registers, but rather in specific memory locations within the FORTRAN library subroutine F:XFDA. The rules for passing these values are somewhat complicated (e.g., double precision values must be passed in an unpacked six-word format).

Procedures for writing a function of double precision or complex type, are included in Appendix D. These procedures apply only to functions, and not to subroutines, which simply return any values through the arguments.

5.2.2 Requirements of the Subprogram

In order to generate the proper object code for the linker, the compiler must know the name of the subprogram's non-shareable data segment. For this reason, a convention has been adopted: the name of a subprogram's data segment must be the same as the subprogram name, padded to eight characters with colons. Thus, MYFUNC'S data segment's name is MYFUNC::, as illustrated in Figure 5-3 (line 34). Even though the assembler allows names up to eight characters in length, a subprogram name should be no longer than six characters. Besides allowing room for the addition of at least two colons to specify the associated data segment, the main reason for this is that FORTRAN, in referencing a subprogram in its object code, will automatically truncate the name to six characters. Thus the FORTRAN statement

```
CALL SUB12345
```

will generate references to the name

```
SUB123
```

and if the subprogram really is NAM'ed SUB12345, the linker will not find it.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

Once the subroutine is entered, it must, as its first executable instruction, call the F:RGMY FORTRAN library subroutine to process the arguments passed from the calling program. As shown on lines 16 and 17, a one-word parameter is passed to F:RGMY, containing the number of arguments (2 in this case).

The subprogram's data segment (MYFUNC:: in this case) must contain a reserve area as its first seven words. These cells are used by F:RGMY. The second and third words of this reserve area must contain the following information:

- The second word must contain a pointer to the FORTRAN name F:XRDA, which is the standard data rel area used by all FORTRAN library routines.

- The third word must contain a pointer to an ASCII message which represents the name of the user's subprogram. The message must begin with the length in bytes of the message, followed by the message itself. The length may be either in the first byte, in which case the name in ASCII starts in the second byte; or the length may be in the second byte (as illustrated), in which case the first byte must be zero and the ASCII text starts in the third byte.

The purpose of this information is to permit F:RGMY to call the error routine if necessary.

Immediately following these seven words, one word must be reserved for each argument, in the order they appear in the CALL statement.

The functional flow through F:RGMY is as follows:

1) F:RGMY is entered with the Y register pointing to the FORTRAN data area, and the X register pointing to the subprogram's data segment. (These registers are loaded prior to entering the function.)

2) F:RGMY checks to see that the number of arguments passed from the FORTRAN program is the same as the count passed in the call to F:RGMY. If not, it outputs an error message, using the data supplied by the user in the second and third words of the work area. To find the count passed from FORTRAN, it makes use of the P register stacked on entrance to the subroutine. This is why the call to F:RGMY must be the first item in the subprogram.

3) F:RGMY computes the address of each argument in the call, and stores it in consecutive words of the subprogram's data area beginning with the eighth word, since the first seven words are required for FORTRAN temp cells. (The addresses of the arguments, not the arguments themselves, are stored.)

4) F:RGMY then saves the pointer to the FORTRAN data area in the first location of the working cells in the subprogram's data area, via the instruction:

```
COPY Y,0(X)
```


® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

The two arguments in the example would have been stored as follows:

address of J → 7(X)
address of K → 8(X)

5) Finally, F:RGMY transfers the address of the subprogram's data area into the Y register:

COPY X,Y

and returns to the subprogram.

In the example, the purpose of the subprogram is to add the contents of J and K. Thus, to load the contents of J into the A register, the instruction:

COPY *JAY(Y),A

is used. This instruction is equivalent to:

COPY *7(Y),A

since JAY has been equated to 7 because F:RGMY has stored the address of J in 7(Y). (Remember that 0(Y) through 6(Y) are used by FORTRAN.)

Then K (i.e., *8(Y)) is added to the J value in the A Register, where FORTRAN expects to find the result on return.

Recall that 0(Y) contains the address of FORTRAN's data area. This value must be restored prior to returning from the subprogram. Thus:

COPY 0(Y),Y

is the last instruction before the RSK.

To summarize the subprogram requirements (refer to Figure 5-2):

1. Separate segments for procedure and data are required for re-entrancy, if any arguments are passed and/or if the subprogram uses any variable data. The data segment area must be the subprogram name filled out to eight characters with colons.

2. The first seven words of the data segment area are reserved for FORTRAN use, specifically for F:RGMY, the argument passing routine. Because the Y register upon return from F:RGMY points to this area, this area may be referenced as 0(Y) through 6(Y). 1(Y) should point to F:XRDA, which must also be declared EXTERNAL. 2(Y) should point to the name of the subprogram in ASCII, preceded by its byte length.

3. The call to F:RGMY must be the first instruction. F:RGMY must also be declared EXTERNAL.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

4. Because F:RGMY saves the address of the FORTRAN calling program's data area (found in the Y register on entry) in 0(Y), the instruction

```
COPY   0(Y),Y
```

must be performed before exiting the subprogram.

5. While it is not recommended, the user may ignore F:RGMY, especially if no arguments are passed. In that case it is not necessary to follow points 2,3, and 4 above. If this is done, and there are no arguments, the following should be noted:

   - The Y register contents on entry to the subprogram must be either kept intact, or saved and restored upon return.
   - The X register contents on entry will point to the data area, if any. If no data area exists, the X register value will be meaningless.
   - The word containing the number of arguments resides immediately following the FORTRAN call to the subprogram. If this count is zero, the stacked P register for return need not be incremented before return.

6. Transfer addresses should not be used on END statements. If this is done, an incorrect transfer address will be generated for FORTRAN startup.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

```
PAGE    1   1981/03/12 11:11:51.00   C  FORTRAN MAIN PROGRAM
FORTRAN(F000)    FTNDEMO    .FTN

 0001 C  FORTRAN MAIN PROGRAM
 0002 C
 0003 C  THIS FORTRAN MAIN PROGRAM CALLS AN ASSEMBLY
 0004 C  LANGUAGE SUBROUTINE (MYSUBR) AND AN ASSEMBLY
 0005 C  LANGUAGE FUNCTION (MYFUNC).
 0006 C
 0007         DO 10 J=0,5
 0008         DO 10 K=0,5
 0009         CALL MYSUBR(J,K,L)
 0010         WRITE(6,100) J,K,L
 0011         L=MYFUNC(J,K)
 0012         WRITE(6,200) J,K,L
 0013 10      CONTINUE
 0014 100     FORMAT(' MYSUBR ',I1,'+',I1,' = ',I2)
 0015 200     FORMAT(' MYFUNC ',I1,'+',I1,' = ',I2)
 0016         STOP
 0017         END


PAGE    2   1981/03/12 11:11:55.50   C  FORTRAN MAIN PROGRAM
FORTRAN(F000)    FTNDEMO    .FTN


SCALAR ALLOCATION

LOCN NAME     WORDS TYPE     MODE       LOCN NAME     WORDS TYPE     MODE       LOCN NAME     WORDS TYPE     MOD
000A J            1 SCALAR   INTEGER*2  000B K            1 SCALAR   INTEGER*2  000C L            1 SCALAR   INT
```

Figure 5-1. Sample Calling Program (Page 1 of 2)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
PAGE    3   1981/03/12 11:12:01.25   C  FORTRAN MAIN PROGRAM
FORTRAN(F000)    FTNDEMO    .FTN


 SUBPROGRAMS CALLED

 NAME    TYPE    ARGS   NAME    TYPE    ARGS   NAME    TYPE    ARGS   NAME    TYPE    ARGS

 MYSUBR  USER    3      F:RWEF  RUNTIME        F:RIOL  RUNTIME        F:RETF  RUNTIME
 MYFUNC  USER    2      F:RSTO  RUNTIME        F:RSTU  RUNTIME

 STATEMENT LABELS

 LOCN  LABEL  USE        LOCN  LABEL  USE        LOCN  LABEL  USE        LOCN  LABEL  USE

 0053  #10    DO END     0004  #100   FORMAT     0014  #200   FORMAT     0026  #M3
 0028  #M4

 STATEMENT LOCATION MAP

 LINE   LOCN  LINE   LOCN  LINE   LOCN  LINE   LOCN  LINE   LOCN  LINE   LOCN  LINE   LOCN  LINE   LOCN

 7      0025  8      0027  9      0029  10     0030  11     003E  12     0045  13     0053  14     005D
 15     005D  16     005D  17     005F

 ENTRY=0024
 PROGRAM SIZE=0066 WORDS
 DATA SIZE=000D WORDS
 LONG INSTRUCTIONS USED 0
 COMPILATION COMPLETE  0 ERRORS (0000) 0 WARNINGS (0000)
```

Figure 5-1. Sample Calling Program (Page 2 of 2)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
PAGE  0001  MACRO (F100)
1981/03/12  10:57:39.25
SI = DM01.MYSUBR.ASM    SO = DM01.MYSUBR.OBJ
                    0001  *
                    0002  *  THIS ASSEMBLY LANGUAGE SUBROUTINE ADDS THE FIRST
                    0003  *  TWO ARGUMENTS AND RETURNS THE RESULT IN THE
                    0004  *  THIRD ARGUMENT.
                    0005  *
     0000           0006          NAM       MYSUBR
                    0007          EXTR      F:RGMY,F:XRDA
                    0008  *
                    0009  *  MYSUBR SHARABLE PROGRAM SEGMENT
                    0010  *
     0000 0002      0011  MYSUBR  REL
                    0012          SHARABLE MYSUBR
                    0013  *
     00000000       0014  MYSUBR  EQU       $
0000 FF8A 0008      0015          JSK       F:RGMY          CALL ARG VERIFY ROUTINE IN FTN LIB
0001 0003           0016          WORD      3               NUMBER OF ARGUMENTS
                    0017  *
     00000007       0018  JAY     EQU       7
     00000008       0019  KAY     EQU       8
     00000009       0020  ELL     EQU       9
                    0021  *
0002 8147 0007      0022          COPY      *JAY(Y),A       GET J
0003 8948 0008      0023          ADD       *KAY(Y),A       ADD K
0004 8549 0009      0024          COPY      A,*ELL(Y)       RETURN IN L
0005 E040 0000      0025          COPY      0(Y),Y          RESTORE FORTRAN DATA AREA ADDR INTO Y
0006 2309           0026          RSK
                    0027  *
                    0028  *  NAME IN TEXT, FOR ERROR MESSAGE
                    0029  *
0007 0006           0030  SUBNAME WORD      6
0008 4D59           0031          BYTE      'MYSUBR'
0009 5355
000A 4252
                    0032  *
                    0033  *  MYSUBR NON-SHARABLE DATA SEGMENT
                    0034  *
```

Figure 5-2. Sample Assembly Language Subroutine (Page 1 of 2)

CA Computer Automation

REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

©1980 COMPUTER AUTOMATION, INC.

```
                1,0
                F:XRDA
                SUBNAME
                  4,0

                OF J,K AND L:
                     3,0

                     0(Y)
                     1(Y)
                     2(Y)
                     3(Y)  THRU 6(Y)

PAGE  0002  MACRO (F100)
1981/03/12  10:57:43.75

0008 0001 F:RGMY
     0000 0003   0035  MYSUBR:: REL
                 0036  *
                 0037           RES
0000 0000        0038           WORD
0001 0000        0039           WORD
0002 0007        0040           RES
0003 0000        0041  *
                 0042  *  ADDRESSES
                 0043  *
0007 0000        0044           RES
                 0045           END

0000  ERRORS   (0000/0000)
0000  WARNINGS (0000/0000)
```

Figure 5-2. Sample Assembly Language Subroutine (Page 2 of 2)

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
PAGE  0001  MACRO (F100)
1981/03/12  10:58:40.75
SI = DM01.MYFUNC.ASM    SO = DM01.MYFUNC.OBJ
                    0001  *
                    0002  *  THIS ASSEMBLY LANGUAGE FUNCTION ADDS THE TWO
                    0003  *  ARGUMENTS AND RETURNS THE RESULT.
                    0004  *
     0000           0005          NAM       MYFUNC
                    0006          EXTR      F:RGMY,F:XRDA
                    0007  *
                    0008  *  MYFUNC SHARABLE PROGRAM SEGMENT
                    0009  *
     0000 0002      0010  MYFUNC   REL
                    0011          SHARABLE  MYFUNC
                    0012  *
     00000000       0013  MYFUNC   EQU       $
0000 FF89 000A      0014          JSK       F:RGMY          CALL ARG VERIFY ROUTINE IN FTN LIB
0001 0002           0015          WORD      2               NUMBER OF ARGUMENTS
                    0016  *
     00000007       0017  JAY       EQU       7
     00000008       0018  KAY       EQU       8
                    0019  *
0002 8147 0007      0020          COPY      *JAY(Y),A       GET J
0003 8948 0008      0021          ADD       *KAY(Y),A       ADD K
0004 E040 0000      0022          COPY      0(Y),Y          RESTORE FORTRAN DATA AREA ADDR INTO Y
0005 2309           0023          RSK
                    0024  *
                    0025  *  NAME IN TEXT, FOR ERROR MESSAGE
                    0026  *
0006 0006           0027  SUBNAME   WORD      6
0007 4D59           0028          BYTE      'MYFUNC'
0008 4655
0009 4E43
                    0029  *
                    0030  *  MYFUNC NON-SHARABLE DATA SEGMENT
                    0031  *
000A 0001 F:RGMY
     0000 0003      0032  MYFUNC:: REL
                    0033  *
```

Figure 5-3. Sample Assembly Language Function (Page 1 of 2)

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

CA ComputerAutomation

```
PAGE  0002  MACRO (F100)
1981/03/12  10:58:45.00

0000 0000        0034            RES       1,0             0(Y)
0001 0000        0035            WORD      F:XRDA          1(Y)
0002 0006        0036            WORD      SUBNAME         2(Y)
0003 0000        0037            RES       4,0             3(Y) THRU 6(Y)
                 0038  *
                 0039  *  ADDRESSES OF J AND K:
                 0040  *
0007 0000        0041            RES       2,0
                 0042            END

0000  ERRORS     (0000/0000)
0000  WARNINGS   (0000/0000)
```

Figure 5-3. Sample Assembly Language Function (Page 2 of 2)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
PAGE 0001                                          MULTI-TASKING LINKER (EC)
1981/03/12 11:07:56                                    FILE USAGE SUMMARY

DM1.FTNDEMO.BIN=DM1.FTNDEMO.OBJ+
DM1.MYSUBR.OBJ+
DM1.MYFUNC.OBJ+
DM.FORTRAN.LIB+
     NOTE: ***  FORTRAN.LIB  REV E200 ***
DM.OS4.LIB
     NOTE: ***  OS4.LIB - REV E400  ***
```

Figure 5-4. LINK Memory Map of Sample FORTRAN Program (Page 1 of 3)

CA ComputerAutomation

REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

©1980 COMPUTER AUTOMATION, INC.

MULTI-TASKING LINKER (EC,E8,E5,E3,--,E2,E1)
ALL SYMBOLS

```
THE TRANSFER ADDRESS IS 0024(FORTRAN)

ABSOLUTE SYMBOLS     (0000 - FFFE)
  0001-I:IO       0104-E:ERR      0109-E:SF       010E-E:ISM      0113-E:PRINT    011F-E:LL2      0126-E:BUFSEM
  0100-E:MON      0105-E:HIGH     010A-E:UF       010F-E:ICH      0114-E:CIFP     0123-F:CIHHD    0127-E:ERRTAB
  0101-E:MONE     0106-E:LOW      010B-E:JCL      0110-E:GFN      0115-E:SLFP     0124-E:ZBSEM    0128-E:SYSFLG
  0102-E:LOAD     0107-E:UAT      010C-E:IJCL     0111-E:RCI      0116-E:EXEC     0125-E:BASEM    FFF6-F:XNFU
  0103-E:LDR      0108-E:UATP     010D-E:MSG      0112-E:SLU      0118-E:LL1

 MEMORY  RELOCATABLE       (0000 - FFFF)  0000-R:LOW      1721-R:HIGH

 SEGMENT FORTRAN  (0000 - 0065) RELOCATABLE - ROM, SHAREABLE      (0000 - 0065)
  0024-F:MAIN

 SEGMENT MYSUBR   (0066 - 0071) RELOCATABLE - RAM, SHAREABLE      (0066 - 0071)
  0066-MYSUBR

 SEGMENT MYFUNC   (0072 - 007C) RELOCATABLE - RAM, SHAREABLE      (0072 - 007C)
  0072-MYFUNC

SEGMENT F:LIBR    (007D - 1439) RELOCATABLE - ROM, SHAREABLE      (007D - 1439)
  007D-F:RREF     010A-F:RLUS     0314-F:IOER     05F2-F:RFE      0A88-F:XMPP     0E46-F:ERCS     10C7-F:XCLS
  008C-F:RWEF     010E-F:RCUS     0316-F:RFFD     05F8-F:RFD      0AAC-F:XVPP     0EFA-F:XSTP     1108-F:XEND
  00C1-F:RDLW     0121-F:RAOL     0345-F:RFWD     0990-F:RFTS     0AB9-F:XGNP     0EFC-F:XCAF     112E-F:XBCK
  00C2-F:RRIF     0121-F:RIOL     034A-F:RFSW     099A-F:RFAD     0AD1-F:XLYA     0F13-F:XGLS     118D-F:XREW
  00CD-F:RWIF     0125-F:R4OL     0354-F:RFSO     09AD-F:RFDA     0AE1-F:XTRM     0F37-F:XASW     11AD-F:XSRN
  00E0-F:RETF     0129-F:RROL     0455-F:RFI      09C8-F:RFFD     0B24-F:XGIU     0F65-F:XASR     11C8-F:XENQ
  00F5-F:RIUS     012D-F:RDOL     0486-F:RFL      09D3-F:RFWH     0B28-F:XGOU     0FAA-F:XSHW     11DE-F:XPSE
  00F9-F:R4US     0131-F:RCOL     04E2-F:RFZ      09EC-F:RPAU     0B43-F:XECT     0FC2-F:XSHR     11FF-F:XINT
  00FD-F:RAUS     0133-F:RLOL     055E-F:RFG      09F7-F:RSTO     0E3E-F:ERRS     0FD7-F:XHTR     1431-F:RSTU
  0102-F:RRUS     013C-F:RHUS     05B0-F:RFIR     0A2C-F:RGMY     0E3F-F:XSNE     0FFE-F:XEBT     1437-F:XSTF
  0106-F:RDUS     02FC-F:RFRW     05D9-F:RFF      0A62-F:X1PP     0E45-F:ERRC     1024-F:XOPN

SEGMENT F:FORT:: (143A - 1446) RELOCATABLE - RAM
SEGMENT MYSUBR:: (1447 - 1450) RELOCATABLE - RAM
```

Figure 5-4. LINK Memory Map of Sample FORTRAN Program (Page 2 of 3)

CA ComputerAutomation®
® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.
© 1980 COMPUTER AUTOMATION, INC.

```
PAGE 0003                                MULTI-TASKING LINKER (EC,EB,E5,E3,--,E2,E1)
1981/03/12 11:10:33                                    ALL SYMBOLS

 SEGMENT MYFUNC:: (1451 - 1459) RELOCATABLE - RAM
 SEGMENT F:LIBR:: (145A - 1721) RELOCATABLE - RAM      (145A - 1721)
  1458-F:XRDA     157E-F:XFDS     157F-F:XFDE     1580-F:XFDM     1582-F:XFCR     1588-F:XFCI     160A-F:XFUT
  1575-F:XFDA


MISSING: SRF:ER

NO WARNINGS
```

Figure 5-4. LINK Memory Map of Sample FORTRAN Program (Page 3 of 3)

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

©1980 COMPUTER AUTOMATION, INC.

```
MYSUBR  0+0  =   0
MYFUNC  0+0  =   0
MYSUBR  0+1  =   1
MYFUNC  0+1  =   1
MYSUBR  0+2  =   2
MYFUNC  0+2  =   2
MYSUBR  0+3  =   3
MYFUNC  0+3  =   3
MYSUBR  0+4  =   4
MYFUNC  0+4  =   4
MYSUBR  0+5  =   5
MYFUNC  0+5  =   5
MYSUBR  1+0  =   1
MYFUNC  1+0  =   1
MYSUBR  1+1  =   2
MYFUNC  1+1  =   2
MYSUBR  1+2  =   3
MYFUNC  1+2  =   3
MYSUBR  1+3  =   4
MYFUNC  1+3  =   4
MYSUBR  1+4  =   5
MYFUNC  1+4  =   5
MYSUBR  1+5  =   6
MYFUNC  1+5  =   6
MYSUBR  2+0  =   2
MYFUNC  2+0  =   2
MYSUBR  2+1  =   3
MYFUNC  2+1  =   3
MYSUBR  2+2  =   4
MYFUNC  2+2  =   4
MYSUBR  2+3  =   5
MYFUNC  2+3  =   5
MYSUBR  2+4  =   6
MYFUNC  2+4  =   6
MYSUBR  2+5  =   7
MYFUNC  2+5  =   7
MYSUBR  3+0  =   3
MYFUNC  3+0  =   3
```

Figure 5-5. Execution Printout of Sample FORTRAN Program (Page 1 of 2)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
MYSUBR 3+1 =  4
MYFUNC 3+1 =  4
MYSUBR 3+2 =  5
MYFUNC 3+2 =  5
MYSUBR 3+3 =  6
MYFUNC 3+3 =  6
MYSUBR 3+4 =  7
MYFUNC 3+4 =  7
MYSUBR 3+5 =  8
MYFUNC 3+5 =  8
MYSUBR 4+0 =  4
MYFUNC 4+0 =  4
MYSUBR 4+1 =  5
MYFUNC 4+1 =  5
MYSUBR 4+2 =  6
MYFUNC 4+2 =  6
MYSUBR 4+3 =  7
MYFUNC 4+3 =  7
MYSUBR 4+4 =  8
MYFUNC 4+4 =  8
MYSUBR 4+5 =  9
MYFUNC 4+5 =  9
MYSUBR 5+0 =  5
MYFUNC 5+0 =  5
MYSUBR 5+1 =  6
MYFUNC 5+1 =  6
MYSUBR 5+2 =  7
MYFUNC 5+2 =  7
MYSUBR 5+3 =  8
MYFUNC 5+3 =  8
MYSUBR 5+4 =  9
MYFUNC 5+4 =  9
MYSUBR 5+5 = 10
MYFUNC 5+5 = 10
```

Figure 5-5. Execution Printout of Sample FORTAN Program
(Page 2 of 2)

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

5.3 FORTRAN SUBPROGRAMS CALLED FROM AN ASSEMBLY LANGUAGE PROGRAM

The preceding subsection presented an application under RTX4 in which a FORTRAN program initializes two variables and calls an assembly language subprogram to add them together.

This is the reverse situation, wherein an assembly language program calls a FORTRAN subprogram.

The calling sequence is in a fixed format as shown in Figure 5-8.

Prior to the call, the Y register must point to the calling program's data area and the X register must point to the subprogram's data area, as in the previous example. The actual call is structured as follows:

```
JSK       Subprogram
WORD      Argument Count
WORD      Code Word
WORD      Argument 1
WORD      Argument 2
            .
            .
            .
WORD      Argument 8
WORD      Code Word
WORD      Argument 9
            .
            .
            .
```

where: Argument Count is the number of arguments passed from the assembly language call to the FORTRAN subprogram. This Count is actually the total number of argument lines that appear below all code word lines in the call. It does not include the code words themselves. If more than eight arguments are passed by the program as shown above, a new code word line is placed under the first eight argument lines, followed by a new set of up to eight argument lines, and so on. If the Count is zero, no code word or argument lines are included in the call.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION INC

© 1979 COMPUTER AUTOMATION, INC.

code word

is a number defining the types of arguments contained in the next eight argument lines. This value is comprised of eight two-bit fields with bits 15-14 corresponding to the first following argument, bits 13-12 corresponding to the second, and so on. The meanings of the two-bit codes are as follows:

Binary Code	Meaning
00	Argument is an absolute address.
01	Argument is an absolute indirect address.
10	Argument is a Y-relative address.
11	Argument is a Y-relative indirect address.

The code word should be padded to the right with zeros if less than eight arguments are specified.

Arguments 1-n

are addresses of arguments to be passed from the assembly language call to the FORTRAN subprogram.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

```
PAGE  0001  MACRO (E100) FORTRAN DEMO3
1980/08/29  16:16:48.50  DEMO3A.ASM
SI = DM01.DEMO3A      .ASM   SO = DM01.DEMO3A      .OBJ   LI = DM01.DEMO3A      .LST

                    0003  * THIS DEMO SHOWS AN ASSEMBLY LANGUAGE MAIN PROGRAM
                    0004  * (F:MAIN)  CALLING A FORTRAN SUBROUTINE (FSUBR)
                    0005  *
     0000           0006          NAM     F:MAIN          MAIN PROGRAM ENTRY
                    0007          EXTR    FSUBR           FORTRAN SUBROUTINE
                    0008  *
                    0009  * FORTRAN MAIN PROGRAM SHARABLE SEGMENT
                    0010  *
     0000 0002      0011  F:MAIN  REL
                    0012          SHARABLE F:MAIN
                    0013  *
     00000000       0014  F:MAIN  EQU     $               ENTRY POINT
0000 0900           0015          COPY    =0,A
0001 8791 0013      0016          COPY    A,JAY
0002 8390 0013      0017  LOOP    COPY    JAY,A           SET UP J & K
0003 0B01           0018          ADD     =1,A
0004 878E 0013      0019          COPY    A,JAY
0005 0C15           0020          CLSN    A,=21           LOOP 20 TIMES
0006 9E8B 0012      0021          JMP     DONE            GO WHEN DONE
0007 0B01           0022          ADD     =1,A
0008 878B 0014      0023          COPY    A,KAY
                    0024  *  PREPARE TO CALL FORTRAN SUBROUTINE
0009 A28B 0015      0025          COPY    =SUBPTR,X       PUT SUBR DATA REL PTR IN X
000A E28B 0016      0026          COPY    =MAINPTR,Y      PUT MAIN DATA REL PTR IN Y
000B FF8B 0017      0027          JSK     FSUBR           CALL SUBROUTINE
000C 0003           0028          WORD    3               NUMBER OF ARGUMENTS
000D A800           0029          WORD    :A800           CODE WORD GIVING THE
                    0030  *   ADDRESS MODE OF THE ARGUMENTS (Y-REL IN THIS CASE)
000E 0007           0031          WORD    JAY-MAINPTR     REL ADDR OF J
000F 0008           0032          WORD    KAY-MAINPTR     REL ADDR OF K
0010 0009           0033          WORD    ELL-MAINPTR     REL ADDR OF L
                    0034  *
                    0035  * CONTROL RETURNS HERE AFTER FORTRAN SUBROUTINE
                    0036  * HAS ADDED J AND K INTO L, AND PRINTED ALL
                    0037  * THREE VALUES
```

Figure 5-6. Sample Assembly Language Program Calling a FORTRAN Subprogram (Page 1 of 3)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
PAGE  0002  MACRO (E100) FORTRAN DEMO3
1980/08/29  16:16:49.75  DEM03A.ASM
SI = DM01.DEM03A      .ASM   SO = DM01.DEM03A      .OBJ   LI = DM01.DEM03A      .LST

                    0038  *
0011 9E70 0002      0039            JMP       LOOP          INCREMENT J AND K
                    0040  *
     00000012       0041  DONE      EQU       $
0012 9F85 0018      0042            E:MON     EXIT          IO OS4 MONITOR
     0006           0043            LPOOL
0013 0007 JAY
0014 0008 KAY
0015 0000 SUBPTR
0016 0000 MAINPTR
0017 0001 FSUBR
0018 0000 E:MON
                    0044  *
                    0045  * SUBROUTINE NON-SHARABLE DATA SEGMENT
                    0046  *
     0000 0003      0047  FSUBR::: REL
                    0048  *
     00000000       0049  SUBPTR    EQU       $             PTR TO SUBR DATA REL AREA
                    0050  *
                    0051  * MAIN PROGRAM NON-SHARABLE DATA SEGMENT
                    0052  *
     0000 0004      0053  F:MAIN:: REL
                    0054  *
     00000000       0055  MAINPTR   EQU       $             *ADDR GOES IN Y FOR CALL)
0000                0056            RES       7             FORTRAN WORK AREA USED BY
                                                                      F:RGMY (MUST BE
0007 0000           0058  JAY       WORD      0             THIS IS J
0008 0000           0059  KAY       WORD      0             THIS IS K                  7 CELLS)
0009 0000           0060  ELL       WORD      0             THIS IS L
                    0061            END       F:MAIN

0000  ERRORS    (0000)
0000  WARNINGS  (0000)
```

Figure 5-6. Sample Assembly Language Program Calling a FORTRAN Subprogram (Page 2 of 3)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
SUBROUTINE OUTPUT: J= 1   K= 2   L= 3
SUBROUTINE OUTPUT: J= 2   K= 3   L= 5
SUBROUTINE OUTPUT: J= 3   K= 4   L= 7
SUBROUTINE OUTPUT: J= 4   K= 5   L= 9
SUBROUTINE OUTPUT: J= 5   K= 6   L=11
SUBROUTINE OUTPUT: J= 6   K= 7   L=13
SUBROUTINE OUTPUT: J= 7   K= 8   L=15
SUBROUTINE OUTPUT: J= 8   K= 9   L=17
SUBROUTINE OUTPUT: J= 9   K=10   L=19
SUBROUTINE OUTPUT: J=10   K=11   L=21
SUBROUTINE OUTPUT: J=11   K=12   L=23
SUBROUTINE OUTPUT: J=12   K=13   L=25
SUBROUTINE OUTPUT: J=13   K=14   L=27
SUBROUTINE OUTPUT: J=14   K=15   L=29
SUBROUTINE OUTPUT: J=15   K=16   L=31
SUBROUTINE OUTPUT: J=16   K=17   L=33
SUBROUTINE OUTPUT: J=17   K=18   L=35
SUBROUTINE OUTPUT: J=18   K=19   L=37
SUBROUTINE OUTPUT: J=19   K=20   L=39
SUBROUTINE OUTPUT: J=20   K=21   L=41
```

Figure 5-6. Sample Assembly Language Program Calling a FORTRAN Subprogram (Page 3 of 3)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

```
PAGE    1   1980/08/29 15:25:21.50   C      FORTRAN DEM03
FORTRAN(F000)     DEM03B      .FTN

 0001 C      FORTRAN DEM03
 0002 C
 0003 C      THIS FORTRAN SUBROUTINE (FSUBR) IS CALLED BY AN
 0004 C      ASSEMBLY LANGUAGE MAIN PROGRAM (F:MAIN) WITH THREE
 0005 C      ARGUMENTS, J, K, AND L.  THE SUBROUTINE ADDS
 0006 C      J AND K INTO L, AND OUTPUTS ALL THREE VALUES
 0007 C      TO THE PRINTER.
 0008 C
 0009        SUBROUTINE FSUBR (JDUM,KDUM,LDUM)
 0010 C
 0011        LDUM = JDUM + KDUM
 0012        WRITE (6,200) JDUM,KDUM,LDUM
 0013        RETURN
 0014 200    FORMAT(' SUBROUTINE OUTPUT: J=',I2,'  K=',I2,'  L=',I2)
 0015        END



PAGE    2   1980/08/29 15:25:24.75   C      FORTRAN DEM03
FORTRAN(F000)     DEM03B      .FTN


 DUMMY ARGUMENT ALLOCATION

 LOCN NAME     WORDS TYPE      MODE        LOCN NAME     WORDS TYPE      MODE        LOCN NAME
                                                                                     WORDS TYPE      MOD

 0007 JDUM          1 SCALAR    INTEGER*2 0008 KDUM          1 SCALAR    INTEGER*2
 0009 LDUM          1 SCALAR    INT
```

Figure 5-7. FORTRAN Subprogram Called by Sample Assembly Language Program (Page 1 of 4)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.
© 1979 COMPUTER AUTOMATION, INC.

```
PAGE    3   1980/08/29 15:25:29.25   C      FORTRAN DEMO3
FORTRAN(F000)     DEMO3B      .FTN

 * 0001 C       FORTRAN DEMO3
 * 0002 C
 * 0003 C       THIS FORTRAN SUBROUTINE (FSUBR) IS CALLED BY AN
 * 0004 C       ASSEMBLY LANGUAGE MAIN PROGRAM (F:MAIN) WITH THREE
 * 0005 C       ARGUMENTS, J, K, AND L.  THE SUBROUTINE ADDS
 * 0006 C       J AND K INTO L, AND OUTPUTS ALL THREE VALUES
 * 0007 C       TO THE PRINTER.
 * 0008 C
 * 0009         SUBROUTINE FSUBR (JDUM,KDUM,LDUM)
                  001D  FE00          F          JSK        F:RGMY
                  001E  0003                     WORD       3
 * 0010 C
 * 0011         LDUM = JDUM + KDUM
                  001F  8147                     COPY       *JDUM(Y),A
                  0020  8948                     ADD        *KDUM(Y),A
                  0021  8549                     COPY       A,*LDUM(Y)
 * 0012         WRITE (6,200) JDUM,KDUM,LDUM
                  0022  0906                     COPY       =6,A
                  0023  FF00          F          JSK        F:RWEF
                  0024  1000                     WORD       :1000
                  0025  0004                     WORD       #200
                  0026  8047                     COPY       @JDUM(Y),A
                  0027  FE00          F          JSK        F:RIOL
                  0028  8048                     COPY       @KDUM(Y),A
                  0029  FE00          F          JSK        F:RIOL
                  002A  8049                     COPY       @LDUM(Y),A
                  002B  FE00          F          JSK        F:RIOL
                  002C  FE00          F          JSK        F:RETF
 * 0013         RETURN
                  002D  E040                     COPY       0(Y),Y
                  002E  2309                     RSK
 * 0014 200     FORMAT(' SUBROUTINE OUTPUT: J=',I2,'  K=',I2,'  L=',I2)
                  0004  2827            #200     BYTE       '(' SUBROUTINE OUTPUT: J=',I2,'  '
                  0014  4B3D                     BYTE       'K=',I2,'  L=',I2) '
```

Figure 5-7. FORTRAN Subprogram Called by Sample Assembly Language Program (Page 2 of 4)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

FORTRAN(F000) DEMO3B .FTN

```
 * 0015        END
                 002F  E040                    COPY      0(Y),Y
                 0030  2309                    RSK
                 001D  FF93          B         JSK       *F:RGMY
                 0031  0000                    WORD      F:RGMY
                 0023  FF8E          B         JSK       *F:RWEF
                 0032  0000                    WORD      F:RWEF
                 0027  FF8B          B         JSK       *F:RIOL
                 0029  FF89          B         JSK       *F:RIOL
                 002B  FF87          B         JSK       *F:RIOL
                 0033  0000                    WORD      F:RIOL
                 002C  FF87          B         JSK       *F:RETF
                 0034  0000                    WORD      F:RETF

 SUBPROGRAMS CALLED

 NAME     TYPE     ARGS    NAME     TYPE     ARGS    NAME     TYPE     ARGS    NAME     TYPE

 F:RWEF   RUNTIME          F:RIOL   RUNTIME          F:RETF   RUNTIME          F:RGMY   RUNTIME

 STATEMENT LABELS

 LOCN  LABEL  USE          LOCN  LABEL  USE          LOCN  LABEL  USE          LOCN  LABEL

 0004  #200   FORMAT

 STATEMENT LOCATION MAP

 LINE   LOCN  LINE   LOCN  LINE   LOCN  LINE   LOCN  LINE   LOCN  LINE   LOCN  LINE

 9      0010  11     001F  12     0022  13     002D  14     002F  15     002F
```

Figure 5-7. Sample Assembly Language Program Calling a FORTRAN Subprogram (Page 3 of 4)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

CA ComputerAutomation®

```
PAGE    5   1980/08/29 15:25:32.00    C      FORTRAN DEM03
FORTRAN(F000)     DEM03B      .FTN

 ENTRY=001D
 PROGRAM SIZE=0035 WORDS
 DATA SIZE=000B WORDS
 LONG INSTRUCTIONS USED 0
 COMPILATION COMPLETE  0 ERRORS (0000) 0 WARNINGS (0000)
```

Figure 5-7. Sample Assembly Language Program Calling a FORTRAN Subprogram (Page 4 of 4)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

SECTION 6

SYSTEM DISKETTES

6.1 INTRODUCTION

This section outlines the contents of the FORTRAN IV product diskette and describes how it is used during the program development process. It also discusses the use of the NM4 Instruction Emulator module which is delivered as part of the RTX4 Library diskette (product No. 84-93410-01).

6.2 FORTRAN PRODUCT DISKETTE

FORTRAN IV is provided on a diskette having the following:

CAI Part Number	93580-02
Diskette File Name	F58002
Model Number	45066-00

The FORTRAN files contained on this diskette are:

FORTRAN.SYS
FTNSCAN.SOL
FTNALLOCATE.SOL
FTNGEN.SOL
FORTRAN.LIB
FORTRAN.OPT

FORTRAN.SYS is the memory-resident portion of the compiler. It is this file which is loaded with the /FORTRAN command. The other three files with .SOL extensions are the scan, allocate, and generation FORTRAN system overlays, which are loaded as needed by FORTRAN.SYS.

FORTRAN.LIB consists of all runtime routines needed to link a FORTRAN program.

FORTRAN.OPT consists of all runtime routines, but employs the hardware floating point instructions where possible, for use on processors that contain those instructions.

Normally, no modification should be required to any of these files in order to compile a FORTRAN program, except to copy them from the delivered product diskette to the desired system device. Bear in mind that the FORTRAN.SYS module, during its execution, loads the three overlays from the SF logical unit, so all four files must always reside on the same device.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

6.3 INSTRUCTION EMULATOR MODULE

The FORTRAN compiler is designed to run on a standard LSI 4/30 processor; that is, it uses certain instructions such as CEA, CSM, and various SHIFT instructions which are optional on the LSI-4/10 processor. However, the object code generated by the compiler, and runtime library routines, use floating point and other instructions which require emulation on any un-optioned processor.

Therefore, although compilation of any FORTRAN program can be performed on an un-optioned 4/30 processor, execution of the compiled program cannot, in most cases. For this reason, the NM4 Instruction Emulator module is delivered as an additional part of the FORTRAN package.

The Emulator is fully described in the NAKED MINI® 4 Optional Instruction Emulator manual. The following discussion summarizes the procedures for using the Emulator.

Each non-standard LSI-4/10 instruction is emulated through the unimplemented instruction trap. (The List instructions are not emulated.) The Emulator library consists of the following modules:

- EM:410X — This module supports emulation of all instructions which are optional on the LSI-4/10 but standard on the LSI-4/30 and 4/90, e.g., the SUBC instruction.

- EM:430X — This module supports emulation of all instructions which are optional on the 4/10 and 4/30 but standard on the 4/90, less those instructions included in the EM:410X module above. That is, an ANDE instruction is included here, because it is optional on both the 4/10 and the 4/30, but a SUBC instruction is not. Note that there are no optional instructions on the 4/30 which are not also optional on the 4/10 (except the List instructions, which are not emulated).

- EM:FLPT — This module supports emulation of the floating point instructions ADDF, SUBF, MULF, DIVF, CSKF, FIX, and FLOAT. These instructions are optional on all three processors.

- EM:410 — This module consists of LOAD directives for the EM:410X, EM:430X, and EM:FLPT modules, which comprise the instruction set of a 4/10 option card.

- EM:430 — This module consists of LOAD directives for the EM:430X and EM:FLPT modules, which comprise the instruction set of a 4/30 option card.

Versions C0 and higher of the RTX4 library include an RTX4/Emulator interface module called R:EMLTR. This module, and selected modules from the above list, must be linked in during OS4 system generation in order to run the FORTRAN compiler on an un-optioned 4/10 processor, or to run the linked FORTRAN compiler output on any un-optioned processor. If a FORTRAN application is to run under RTX4, any needed emulation routines are included at link time.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

Including the Emulator in an OS4 system generation link requires inclusion of a dummy module as the first file of the link process, and unconditional linking (NOLIBRARY) of any needed Emulator routines during the link process. The dummy module contains the instruction:

```
LOAD    R:EMLTR
```

The Emulator is included in a FORTRAN/RTX4 application link by specifying the following directives in the assembly language start-up module:

```
LOAD    R:EMLTR
LOAD    EM:nnn
LOAD    EM:nnn
    etc.
```

where "EM:nnn" represents a required Emulator module. Table 6.1 outlines the Emulator routines required for the various link processes.

Table 6.1. Emulator Routines Needed for FORTRAN Compilation/Execution

PROCESSOR	FOR FORTRAN COMPILATION	FOR FORTRAN PROGRAM EXECUTION
Un-optioned 4/10	R:EMLTR EM:410X	R:EMLTR EM:410
4/10 with Option Card	None	None
Un-optioned 4/30	None	R:EMLTR EM:430
4/30 with Option Card	None	None
4/90 without Floating Point Option	None	R:EMLTR EM:FLPT
4/90 with Floating Point Option	None	None

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

APPENDIX A

FORTRAN OPERATIONS SUMMARY

The following general procedures are used to prepare a FORTRAN program for execution:

1) Design and code the FORTRAN program and subroutines using the NAKED MINI 4 FORTRAN IV Language Manual as a reference. If assembly language subprograms are referenced by the FORTRAN program, refer to Section 5 of this manual for special instructions relating to their use. If RTX4 execution is intended, an assembly language start-up module must be included. (See Section 5.) Input the source code to the desired file device.

2) Invoke the compiler using the /FORTRAN command described in Section 2 of this manual.

3) Link the program for either RTX4 or OS4 execution, following the procedures outlined in Section 3.

4) If OS4 execution is intended, make necessary runtime unit assignments. Assign all logical unit numbers used in FORTRAN program I/O statements to needed physical units.

5) Load the program using the OS4 /file command if it is to be run under OS4. Either use the OS4 /AUTOLOAD command to load and execute the program for RTX4 execution, or copy the user program to OS::.SYS on a disk and use the hardware AutoLoad facility to load and execute it.

6) Respond to PAUSE and STOP messages and error conditions as they occur during program execution.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1978 COMPUTER AUTOMATION, INC

APPENDIX B

ERROR/WARNING MESSAGES AND STATUS VALUES

B.1 COMPILER DIAGNOSTICS DURING SCAN PHASE

The messages listed below are displayed in response to certain conditions during the scan phase.

Message	Error/Warning (E/W)	Comments
ALLOCATION ERRORS	E	A name appearing in a declaration statement is invalid because of previous usage. For example: A COMMON name is already held in COMMON, or is not a scalar or array. An adjustable dimension is not a scalar dummy. A name was dimensioned or typed twice. A dummy was declared in COMMON, EQUIVALENCE, or EXTERNAL. An EQUIVALENCE or DATA array subscript is out of range.
ANSI ARRAY WARNING	W	A format stored in an integer or logical array probably will not work in ANSI mode.
ARGUMENT CONVERTED	W	A subprogram argument is of the wrong type and was converted to the right type. This can happen to a library function (proper type is known to the compiler), a statement function (type was determined at the definition), or an ordinary external function (if a previous call is made with different type arguments). An argument of logical type cannot be converted to numeric, or vice versa, CONFLICT error message.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1978 COMPUTER AUTOMATION, INC.

Message	Error/Warning (E/W)	Comments
ARGUMENT COUNT	E	The wrong number of arguments was passed to the subprogram. This can happen in the same cases as ARGUMENT CONVERTED.
ARRAY SIZE	E	An array is dimensioned greater than 64K.
BLOCK DATA ONLY	E	This statement may not appear in a BLOCK DATA subprogram.
CONSTANT SIZE	E	One of the following has occurred: • A floating point constant is Out of Range; -1.7×10^{38} to -1.5×10^{-39}, 0, and 2.9×10^{-39} to 1.7×10^{38}. • Exponent size on a floating point constant is too large. • A hexadecimal or Hollerith constant is too long for context, or more than 255 or less than 1. • A DATA repeat count is not an integer with a value greater than 1.
CONTROL VARIABLE REUSE	E	A nested DO loop is attempting to use the same control vaiable as an outer loop.
DATA COUNT	E	The number of constants is not the same as the number of variables. (Long Hollerith constants may act as several constants.) This message usually is followed by a SYNTAX error message.
DATA TYPE	E	A constant is not of the same type as the variable. This message does not apply to hexadecimal or alphanumeric constants.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

Message	Error/Warning (E/W)	Comments
DECLARATION CONFLICT	E	A name cannot be used in the context given, due to previous usage (e.g., the name following a call is not of a subroutine). See also MISUSED IDENTIFIER.
DIMENSION RANGE	E	A negative or zero dimension was specified, or an upper bound less than the lower bound was specified.
DUPLICATE DUMMY	E	The same name was used twice as a dummy in the definition of a FUNCTION, SUBROUTINE, or a statement function.
DUPLICATE OPTION	W	A keyword option appears more than once.
EXTRA COMMA	W	Two consecutive commas were encountered in a list of items.
FORMAT LABEL	E	A label previously was used to reference a FORMAT statement (e.g, in a READ/WRITE statement).
ILLEGAL DO CLOSE	W	A DO loop may not terminate on a GO TO, DO, arithmetic IF, RETURN, or STOP statement. If DO loops are also improperly nested, this message may not appear, but the label will appear under OPEN DO LOOPS.
ILLEGAL LABEL	E	The label is outside of the range 1-99999, a DO terminal label has already appeared; or the terminal label is used elsewhere in a construct, indicating it is a FORMAT label.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

Message	Error/Warning (E/W)	Comments
ILLEGAL NUMBER	E	FORTRAN has encountered one of the following: • An integer value outside the allowable range (i.e., -32768 to +32767) • A format repeat count value of zero • An integer value in a complex constant • A negated alphanumeric string • An illegal value in a "type" statement (e.g., INTEGER*3) • A hexadecimal value containing more than four digits
INCOMPATIBLE OPTIONS	E	The keyword option is invalid because of previous code.
INVALID UNIT ASSIGNMENT	E	The I/O unit specified is invalid.
JUMP LABEL	E	This label previously appeared on a statement that was not a FORMAT.
LABEL MISSING	W	An unlabeled FORMAT statement was encountered, or an unlabeled statement follows a jump and cannot be reached. Although this is a warning, an unlabeled format statement will not be generated.
MISSING COMMA	W	A comma is needed between two items.
MISSING UNIT	E	No I/O unit was specified.
MISUSED NAME	E	A name cannot be used in a specified way because of previous usage (e.g., a DO index is used as an array, or a name to the left of an equal sign is not used as a scalar or array).
MULTI DEFINED	E	A statement label was previously defined.

● REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

Message	Error/Warning (E/W)	Comments
NOT ARRAY	E	A FORMAT reference name is not an array.
NOT INTEGER	E	This expression must be integer (e.g., a subscript) but it contains at least one non-integer element. A $ marks the end of the expression, but the erroneous element may not be the last one in the expression.
NUMBER OF SUBSCRIPTS	E	Too many or too few subscripts were indicated for an array or an array appears at the beginning of a statement without subscripts.
RANGE	E	The letters in an IMPLICIT statement are not in ascending order or a constant subscript is outside the array. (This is valid in constructs where the compiler must evaluate the subscript, such as equivalence statements.)
RANGE	W	A (constant) subscript references an address beyond the end of the array.
STATEMENT NOT ALLOWED	E	Statements do not appear in the correct order. A logical IF may not control a DO or another logical IF. In general, declaration statements must come at the beginning of the program.
SYNTAX	E	Operands, operators, or punctuation was improperly sequenced. In a FORMAT statement, this error may be caused by incorrect Hollerith fields. This is by far the most common error message.
TYPE CONFLICT	E	A complex expression appears in an arithmetic IF; or it appears improperly in an assignment statement, relational expression, or exponentiation; or a logical operand or argument appears where a numeric should be, or vice versa.
UNDIMENSIONED	E	The name followed by a left parenthesis to the left of an equal sign was not dimensioned or a variable presented with subscripts is not an array.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

Message	Error/Warning (E/W)	Comments
UNRECOGNIZABLE	E	The compiler cannot determine what kind of statement this is supposed to be.
UNSUCCESSFUL INCLUDE	E	The file specified in an INCLUDE statement could not be found or opened.

B.2 COMPILER DIAGNOSTICS DURING ALLOCATE PHASE

The messages listed below are displayed in response to certain conditions during the allocate phase.

Message	Comments
ALLOCATION ERRORS	The list of variable names following this message were involved in illegal EQUIVALENCE, due to one of the following: • A conflict of storage assignment • An extension of COMMON • A working storage overflow
FUNCTION NAME NOT REFERENCED	The name of a FUNCTION, which is supposed to return the result, was never referenced. This message appears at the beginning of the allocation map.
OPEN DO LOOPS	This diagnostic is followed by lines of the form: 44 OPENED AT LINE 140 This example indicates a "DO 44" on line 140, but the terminal statement with a label 44 was not found. Sometimes the label may have actually appeared, but was not found due to incorrect nesting of DO loops. This message appears at the beginning of the allocation map.
STORAGE OVERFLOW	One of the storage areas (local, blank COMMON, labeled COMMON) has overflowed 64K. This message appears following the map of the corresponding storage area.
UNDEFINED LABELS	This diagnostic is followed by lines of the form: 17 FIRST REF AT LINE 9 In this example, the statement number 17 was never defined, and there is at least one reference to it, on line 9. There may be an overlap between this diagnostic and OPEN DO LOOPS. This message appears at the beginning of the allocation map.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

B.3 FORTRAN RUNTIME ERROR MESSAGES

The messages listed below may be displayed at runtime in response to error conditions. The error messages take the following general form:

(routine), (message) ERROR at :xxxx

where routine and message are as described below, and :xxxx represents in hexadecimal the error location.

Message	Routine	Comments
ARGUMENT TOO LARGE	COS, DCOS, DSIN, DTAN, SIN, TAN	All significance to the result was lost, and a zero is being returned.
ARGUMENT TOO LARGE	DEXP, EXP, IDINT, IFIX, INT, I**R, R**R, D**R, I**D, R**D, D**D	The result of the routine would cause an overflow, so the maximum value is returned.
BOTH ARGUMENTS ZERO	ATAN2, DATAN2	Zero is being returned as the result.
BOTH ARGUMENTS ZERO	CLOG	The real and imaginary parts are both zero, so the negative of the maximum value is being returned.
DIVISION BY ZERO	Many	This condition automatically is tested for many routines, but is not expected to occur.
END OF FILE	ENDFILE, FORMATTED, UNFORMATTED	If this is output during a READ, an end-of-file mark was encountered. If output during a WRITE or ENDFILE, this message means that the end-of-tape or end-of-media was reached, although the requested WRITE was performed. If the END= option was specified, this message does not appear.
ILLEGAL FORMAT CHAR	FORMATTED	A syntax error was encountered in a FORMAT stored in an array.
ILLEGAL INPUT CHAR	FORMATTED	An illegal character was encountered in a numeric input field.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1978 COMPUTER AUTOMATION, INC.

Message	Routine	Comments
ILLEGAL OPERATION	BACKSPACE, ENDFILE, FORMATTED, REWIND, UNFORMATTED	The requested operation cannot be performed on the requested device.
ILLEGAL REPEAT COUNT	FORMATTED	A repeat count of zero was specified in a FORMAT stored in an array.
ILLEGAL UNIT	BACKSPACE, ENDFILE, FORMATTED, REWIND, UNFORMATTED	The unit number is not in the logical unit table.
INCORRECT NUMBER OF ARGUMENTS	Many	A library routine was called with the wrong number of arguments.
INTEGER INPUT OVERFLOW	FORMATTED	The input value exceeds 32K, so the maximum value is being returned.
I/O	BACKSPACE, ENDFILE, FORMATTED, REWIND, UNFORMATTED	A hardware error occurred.
LINE dddd, COMPILATION	Program name	A statement on source line dddd has a compilation error. Line dddd always will be marked with an error message except in the case of an undefined label reference.
NEGATIVE ARGUMENT	ALOG, ALOG10, DLOG, DLOG10, DSQRT, SQRT	A negative argument was specified, so the absolute value was used instead.
NUMBER OF ARGUMENTS	FORMATTED	A FORTRAN-compiled subprogram was called with the wrong number of arguments.
NUMERIC MISMATCH	FORMATTED	A numeric value is associated with a logical format, or vice versa.
OUT OF RANGE	COMPUTED GO TO	The variable (v) is greater than n (the number of labels).
OVERFLOW	CABS, CCOS, CEXP, CSIN, CSQRT, DMOD, DTAN, DTANH, EXP, TAN, TANH	An overflow occurred, so the maximum value of the result was returned.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

Message	Routine	Comments
OVERFLOW	I**I, R**I, D**I, C**I, I**R, R**R, D**R, I**D, R**D, D**D	Exponentiation overflow or underflow occurred. The maximum value or zero was returned respectively, as the result.
PAREN NESTING	FORMATTED	More than eight levels of nesting were encountered. This many levels of nesting can only occur on FORMATs stored in arrays.
REAL INPUT OVERFLOW	FORMATTED	A floating point input value is too large, and a maximum value was returned.
SINGULARITY	STAN, TAN	The tangent of (n+1/2) cannot be expressed, and the maximum value was returned. The use of arguments near the singularity point may generate the message OVERFLOW.

B.4 FORTRAN RUNTIME I/O STATUS (IOSTAT=) VALUES

The I/O status value returned to the user when the IOSTAT= parameter is included in the I/O statement represents one of the following general conditions:

-1	End of file
0	Normal completion, no error
1-255	Error condition

When the end of a file is reached during a read operation and an end-of-file exit address is not included in the READ statement, the operation is terminated and the following message is output to unit 6:

END OF FILE

When an I/O operation ends abnormally and an error exit address was not specified in the I/O statement, an error message corresponding to the error status value is output to unit 6. The error status values, their meanings, and the corresponding error messages are listed below.

Status Value			
Decimal	Hexadecimal	Error Messages	Meaning
1	:01	I/O HALTED	The I/O operation was halted.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

Status Value Decimal	Status Value Hexadecimal	Error Message	Meaning
2	:02	NOT READY	The device is not ready (e.g., the disk drive door is open).
3	:03	WRITE PROTECT	The device to be written to is write-protected.
4	:04	TIME OUT	A watchdog timer has expired before the device has responded.
5	:05	POWER FAIL	The power failed and IOS4 could not recover.
6	:06		Unused
7	:07	VOLUME NOT ON UNIT	The volume name specified by the user is not on the unit specified by the user.
8	:08	VOLUME NOT FOUND	The volume specified by the user was not mounted on a disk drive on the system.
9	:09	LOGICAL UNIT NOT RELEASED	An open file was already connected to the logical unit during an attempted logical unit connection.
10-64	:0A-:40		Unused.
65	:41	BAD CHECK SUM	The paper tape reader encountered a bad check sum. This error generally is caused by problems relating to the paper tape punch operation.
66	:42	UNASSIGNED DEVICE	The requested device was not assigned to a logical unit.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

Status Value Decimal	Status Value Hexadecimal	Error Message	Meaning
67	:43	LUN NOT IN UAT	The requested logical unit name/number is not in the Unit Assignment Table.
68	:44	REACHED END OF MEDIA	The end-of-media was reached.
69	:45	HARDWARE	An unrecoverable hardware error occurred.
70	:46	ILLEGAL OPERATION FOR THIS DEVICE	An operation was requested that cannot be performed by the requested device.
71	:47	NON-EXISTENT MEMORY ADDRESS	Non-existent memory was referenced (e.g., uninstalled memory).
72	:48	MEMORY ADDRESS IN SYSTEM AREA	A referenced memory address is within an area of memory used by the system.
73	:49	DIRECTORY FULL	The volume directory contains all the file entries possible. See the Input/Output Subsystem IOS4 User's Manual (93405-90) for the number of entries allowed for different types of disks.
74	:4A	DISK NOT LABELED	The referenced disk is not labeled.
75	:4B	FILE ADDRESS OVERFLOW	The user's Current File Index (CFI) was addressed a byte count greater than $2^{31}-1$.
76	:4C	ILLEGAL AU	An AU calculation has produced an AU greater than $2^{16}-1$; a user Current File Index (CFI) is too large.
77	:4D	AU NOT FOUND	A user Current File Index (CFI) pointer pointed to an AU beyond the allocated extents.

Revised 8/79

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC

Status Value Decimal	Status Value Hexadecimal	Error Messages	Meaning
78	:4E	WRITE DIRECT HIT END OF FILE	A write direct operation has extended past the end of the file.
79	:4F	NOT EXTENDABLE	A file with a secondary extent size of zero has filled its initial extent.
80	:50	ILLEGAL FILE ADDRESS	An illegal Current File Index (CFI) value was specified.
81	:51	ALLOCATION TABLE FOULED UP	A power failure or system crash has occurred at a critical moment. The user should copy all files off this volume and relabel it.
82	:52	SYSTEM USAGE ONLY	The user has attempted to perform I/O on a system file.
83	:53	FCB CHAIN BAD	The File Control Block (FCB) chain has been destroyed.
84	:54	FILE OPEN	An illegal operation (probably delete) has been attempted on an open file.
85	:55	FILE NOT OPEN	An operation other than Open or Close was attempted on a file which was not open.
86	:56	ILLEGAL FILE POINTER	No Current File Index (CFI) pointer was specified in the user's I/O Block (IOB).
87	:57	UNIT NOT GIVEN	The user has omitted a required unit name.
88	:58	UNASSIGNED FCB	A File Control Block (FCB) is not assigned to a logical unit.
89	:59	BUFFER HAS WRITE PENDING	A buffer contains data which has not been written to the file.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

Status Value Decimal	Hexadecimal	Error Message	Meaning
90	:5A	NO MORE FREE SPACE	The entire disk has been allocated.
91	:5B	BAD TYPE CODE	A bad type code was encountered.
92	:5C	ABORT	The operation was aborted.
93-126	:5D-:7E		Unused
127	:7F	RTXER	An RTX4 error occurred during the I/O operation. The RTX4 error code is returned in register A. These codes are listed in the Real-Time Executive RTX4 User's Manual(93410-90).
128	:80	ACCESS	An access was declared both sequential and direct.
129	:81	FORMAT	Formatted and unformatted (ASCII and binary) data was found in the same file.
130	:82	INVALID TRAILING COUNT	An unformatted binary record has an incorrect trailing count (doesn't match leading count).
131	:83	UNABLE TO BACKSPACE	Too many backspaces were issued; the file pointer is lost.
132	:84	PAST EOF	An I/O operation occurred past the end of a file.
133	:85	TOO MANY UNITS	Too many units were referenced.
134	:86	READ AFTER WRITE	A read operation was attempted after a write during sequential I/O. (The user must first backspace or rewind to write after reading.)

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

Status Value Decimal	Status Value Hexadecimal	Error Message	Meaning
135	:87	CANNOT BACKSPACE NEW UNIT	Backspacing is not allowed as a first operation; rewinding is allowed.
136	:88	FILE ALREADY OPEN	A file was opened twice.
137	:89	INVALID FILE NAME	An invalid file name was specified.
138	:8A	FILE NOT FOUND	The file specified by the user was not on the volume specified by the user.
139	:8B	INVALID LEADING COUNT	An unformatted binary file has an invalid leading count found on backspace.
140	:8C	NON-EXISTENT UNIT	An inquire operation performed before any I/O to a specified unit showed the unit to be non-existent.
141	:8D	FILE ALREADY EXISTS	An existing file was opened as a new file.
142	:8E	CANNOT DELETE PHYSICAL UNIT	A close-with-delete operation was requested on a non-file-type device.
143	:8F	LOGICAL UNIT (UF) NOT ASSIGNED	A logical unit is not assigned (e.g., unit 6 is not assigned to the line printer for free form I/O).
144	:90	ILLEGAL INPUT	An illegal character was input.
145	:91	CANNOT DELETE	An attempt was made to delete an unnamed file.
146	:92	ILLEGAL CHARACTER IN FORMAT	A FORMAT statement contains an illegal character.
147	:93	TOO MANY NESTED PARENTHESES IN LOOP	Too many nested parentheses were encountered.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1979 COMPUTER AUTOMATION, INC.

Status Value Decimal	Status Value Hexadecimal	Error Message	Meaning
148	:94	UNMATCHED RIGHT PARENTHESIS IN FORMAT	A parenthesis is missing in a FORMAT statement.
149	:95	ILLEGAL REPEAT COUNT IN FORMAT	An illegal repeat count was encountered in a FORMAT statement.
150	:96	HOLLERITH INPUT	The Hollerith FORMAT statement cannot be used to order input.
151	:97	NUMERIC/LOGICAL TYPE MISMATCH	A data type statement does not match the type specification made in a FORMAT statement.
152	:98	ILLEGAL MAG. TAPE ACCESS OR FORM	Trying to read or write non-sequential data to Mag. Tape.
153	:99	INVALID DIRECT ACCESS RECORD NUMBER	Trying to direct access a record that doesn't exist.
154	:9A	LINE SIZE ERROR	Logical Unit unassigned or not in the UAT. This error routine is in the Get Line Size routine.

B.5 RTX4 SERVICE CALL ERROR STATUS VALUES

Decimal	Hexadecimal	Description
513	:201	Semaphore exception
514	:202	S-trap out of range
515	:203	Insufficient stack specification
516	:204	Cannot fill environment memory pool request
517	:205	Cannot fill system freepool request
518	:206	Negative activity priority
519	:207	Clock control block (CCB) exception
520	:208	Hardware trap
521	:209	Table ID check failure
522	:20A	System activity violation
523	:20B	Wall Clock CCB exception
524	:20C	Invalid mailbox ID
525	:20D	Template expansion error

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

APPENDIX C

FORTRAN IV RUNTIME LIBRARY SUBROUTINES

The following pages list the name and function of each subroutine and cell contained in the files FORTRAN.LIB and FORTRAN.OPT, which are accessible to user programs (written in assembler or FORTRAN code).

Subroutine Name	Function
AIMAG	Imaginary Part of Complex
AINT	Real Truncate
ALOG	Real Natural Logarithm
ALOG10	Real Logarithm Base 10
AMAXO	Real Maximum of Integer
AMAX1	Real Maximum of Real
AMAX4	Real Maximum of Long Integer
AMINO	Real Minimum of Integer
AMIN1	Real Minimum of Real
AMIN4	Real Minimum of Long Integer
AMOD	Real Modulus
ATAN	Real Arctangent
ATAN2	Real Arctangent 2
CABS	Complex Absolute Value
CCOS	Complex Cosine
CEXP	Complex Exponential
CLOG	Complex Log E
CMPLX	Convert 2 Real to Complex
CONJG	Complex Conjugate
COS	Real Cosine
COSH	Real Hyperbolic Cosine
CSIN	Complex Sine
CSQRT	Complex Square Root
DABS	Double Absolute Value
DATAN	Double Arctangent
DATAN2	Double Arctangent 2
DBLE	Real to Double
DCOS	Double Cosine
DDIM	Double Diminish
DEXP	Double Expontential
DFLOAT	Integer to Double
DIM	Real Diminish
DINT	Double Truncate
DLOG	Double Natural Logarithm
DLOG10	Double Logarithm Base 10
DMAXO	Double Maximum of Integer
DMAX1	Double Maximum of Double
DMAX4	Double Maximum of Long Integer
DMINO	Double Minimum of Integer
DMIN1	Double Minimum of Double
DMIN4	Double Minimum of Long Integer
DMOD	Double Precision Modulus
DSIGN	Double Sign

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

Subroutine Name	Function
DSIN	Double Sine
DSQRT	Double Square Root
DTAN	Double Tangent
DTANH	Double Hyperbolic Tangent
EXP	Real Exponential
F:RGMY	Parameter Passer
F:XFCI	Imaginary Part of Complex
F:XFCR	Real Part of Complex
F:XFDE	MPAC Exponent
F:XFDM	MPAC Mantissa Four Words
F:XFDS	MPAC Sign
F:XRDA	Runtime System Area
FLOAT	Integer to Real
IDINT	Double to Integer
IFIX	Real to Integer
INT	Real to Integer
ISIGN	Integer Sign
LABS	Long Integer Absolute Value
LDIM	Long Integer Diminish
LDINT	Double to Long Integer
LINT	Real to Long Integer
LMAX1	Long Integer Maximum of Real
LMAX4	Long Integer Maximum of Long Integer
LMIN1	Long Integer Minimum of Real
LMIN4	Long Integer Minimum of Long Integer
LMOD	Long Integer Modulus
LSIGN	Long Integer Sign
MAX0	Integer Maximum of Integer
MAX1	Integer Maximum of Real
MIN0	Integer Minimum of Integer
MIN1	Integer Minimum of Real
MOD	Integer Modulus
REAL	Real Part of Complex
SIGN	Real Sign
SIN	Real Sine
SINH	Real Hyperbolic Sine
SNGL	Double to Real
SQRT	Real Square Root
TAN	Tangent
TANH	Real Hyperbolic Tangent

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC.

© 1980 COMPUTER AUTOMATION, INC.

APPENDIX D

RETURNING COMPLEX AND DOUBLE PRECISION VALUES TO FORTRAN FROM AN ASSEMBLY LANGUAGE FUNCTION SUBPROGRAM

Topic 5.3.1 of this manual describes how function results from assembly language function subprograms are returned for integer, logical, long integer, and real data types. These results are returned in the A or AQ registers because they can be manipulated by FORTRAN using hardware integer or floating point math instructions. Complex and double precision values cannot be manipulated in the same way, however, so FORTRAN requires that they be returned in specific cells within the FORTRAN library subroutine F:XFDA. There are eight such cells; their names (which may be referenced via EXTR directives in the function subprogram) and expected contents are:

Name	Expected Contents
F:XFDS	Sign of double precision result
F:XFDE	Exponent of double precision result
F:XFDM	High-order word of four-word double precision mantissa. Words 2, 3, and 4 immediately follow.
F:XFCR	For complex functions, the high-order word of two-word real portion of the complex result
F:XFCI	High-order word of two-word imaginary portion of a complex result

D.1 FORMAT OF FUNCTION RESULT FOR COMPLEX VALUES

The format of a complex result is straightforward: it is simply two two-word floating point values in standard Computer Automation, Inc., floating point format, one value respresenting the real portion, and one representing the imaginary portion. Floating point number formats are described, for example, in the LSI 4/90 Computer Reference Manual, 91-20945-00. Once a complex result is calculated, it is simply stored into F:XFCR and F:XFCR+1 (real part), and F:XFCI and F:XFCI+1 (imaginary part).

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.

D.2 FORMAT OF FUNCTION RESULT FOR DOUBLE PRECISION VALUES

The format of a double precision result is returned in a six-word unpacked form, according to the following rules:

Sign: The sign of a double precision value must be one of two values: :0000 if positive, or :8000 if negative. Only bit 15 holds the sign, but bits 14 through 0 must all be set to zero. The sign is returned in F:XFDS.

Exponent: The exponent is returned unbiased; this is different than it appears in a standard floating point number, which is normally eight bits. A full 16 bits is returned instead, representing the range 2^{-128} to 2^{127}, where the exponent and actual cell contents are as follows:

Exponent	Actual Contents
-128	:FF80
-127	:FF81
-1	:FFFF
0	:0000
1	:0001
127	:007F

The exponent is returned in F:XFDE. Note that the values :0080 through :FF7F are illegal

Mantissa: The mantissa is returned as a 60-bit value, in four words of 15 bits each, where bit 15 of each word is not part of the value and must be zero. The high-order bit, which is normally implied but not present in standard format, must be present in bit 14 of the high-order word. The mantissa must be normalized to a fraction less than 1.0, which implies that bit 14 of the high-order word must be set to 1, unless the entire floating point value is zero. The binary point is assumed to be immediately to the left of bit 14 in the high-order word. The mantissa is returned in F:XFDM (high-order word), F:XFDM+1, F:XFDM+2, and F:XFDM+3 (low-order word).

Register Contents: In addition to locations F:XFDS through F:XFDM+3 being stored as described above, double precision functions must return with the A Register containing the inclusive OR of the sign and the high-order mantissa word; that is, with F:XFDS and F:XFDM ORed together.

® REGISTERED TRADEMARK OF COMPUTER AUTOMATION, INC

© 1979 COMPUTER AUTOMATION, INC.