File No. S360-25
Form C28-6560-0

IBM Systems Reference Library

IBM System/360 Transition Aids:
FORTRAN II Language Conversion Program
For the IBM 1401
Preliminary Specifications

This publication contains preliminary information about the IBM FORTRAN II Language Conversion Program (FORTRAN LCP). The FORTRAN LCP facilitates transition to IBM System/360 by detecting statements in current-systems FORTRAN II source programs that are incompatible with System/360 FORTRAN IV, by converting these statements to the proper System/360 format when possible, and by flagging statements that cannot be converted. This publication is intended to assist programmers and other installation personnel in planning for use of this conversion program.

PREFACE

The FORTRAN Language Conversion Program (FORTRAN LCP) facilitates transition to IBM System/360 by converting FORTRAN II source programs written for IBM current-systems FORTRAN II compilers into source programs for an IBM System/360 FORTRAN compiler.

The preliminary information in this publication will aid the user in planning for use of the FORTRAN LCP as part of his complete plan for conversion to System/360.

PREREQUISITE LITERATURE

The reader of this publication should be familiar with a current IBM FORTRAN II language and with the FORTRAN IV language as it is used with System/360.

The FORTRAN II language for a current IBM system is described in the publication FORTRAN II General Information Manual, Form F28-8074, and one of the following publications:

- FORTRAN Specifications and Operating Procedures; IBM 1401, Form C24-1455
- IBM 1410 FORTRAN, Form J24-1468
- IBM 1620 GOTRAN Interpretive Programming System; Reference Manual, Form C26-5594
- IBM 1620 FORTRAN with FORMAT, Form C26-5619
- IBM 1620 FORTRAN II Programming System Reference Manual, Form C26-5876
- IBM 7070-Series Programming Systems; FORTRAN, Form C28-6170
- User's and Operator's Guide, IBM 7070-Series Programming System; FORTRAN Operating Systems (FOS), Form C28-6369, which contains a listing of features and restrictions added to 7070-Series FORTRAN to form 7070 FOS FORTRAN
- IBM 705 FORTRAN Programming System, Form J28-6122
- IBM 7080 Programming Systems; 7080 Processor; FORTRAN Language, Form J28-6247
- IBM 7090/7094 Programming Systems; FORTRAN II Programming, Form C28-6054

The FORTRAN LCP permits the user to specify that his source programs are to be converted to one of three design levels of System/360 FORTRAN IV -- Basic Programming Support FORTRAN, Level E FORTRAN, or Level H FORTRAN. The reader of this manual should be familiar with the publication that describes the level of System/360 FORTRAN to which he is converting his source programs. The publications that describe System/360 FORTRAN are:

- IBM System/360 Basic Programming Support FORTRAN, Form C28-6504
- IBM Operating System/360 FORTRAN IV (E Level Subset), Form C28-6513
- IBM Operating System/360 FORTRAN IV, Form C28-6515 (This is referred to as Level H FORTRAN.)

The information in this manual is based on System/360 FORTRAN IV specifications as published in the following versions of the above manuals:

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A form for readers' comments appears at the back of this publication. Address any additional comments concerning the contents of this publication to: IBM Corporation, Programming Systems Publications, Department 637, Neighborhood Road, Kingston, New York 12401

Basic Programming Support FORTRAN, Form C28-6504-1, with no technical newsletters (Note that the name Special Support FORTRAN has been changed to Basic Support FORTRAN.)
Level E FORTRAN, Form C28-6513-0, with no technical newsletters
Level H FORTRAN, Form C28-6515-2, with no technical newsletters

ORGANIZATION OF THIS MANUAL

The body of the manual is divided into six major sections with each section devoted to a specific type of information.

The "Introduction" provides basic information on the characteristics of the program.

The section "General Description of the Program" provides the reader with an overall view of the program and serves as an introduction to the detailed sections that follow.

The section "General Problems in Converting to System/360 FORTRAN" contains discussions of some of the major differences between current FORTRAN II and System/360 FORTRAN IV. The section also indicates control card options that can be used to designate the manner in which the FORTRAN LCP is to resolve some of these differences.

The "Conversion Actions" section contains a series of lists that show the manner in which the LCP processes specific source statements. The first list, entitled "Conversion Actions for Current FORTRAN II Compilers," contains discussions of items that must be converted for all current FORTRAN II compilers. Subsequent lists contain items that apply specifically to one current FORTRAN II compiler such as 1401 FORTRAN, 1410 FORTRAN, etc. A user interested in a particular item need look in only two lists -- the all-compilers list and a specific-compiler list -- to obtain information on that item.

The "Conversion Output and Messages" section describes the output listing produced by the FORTRAN LCP and explains the messages that appear in the listing.

The "Control Cards and Operating Procedures" section lists the control information that the user specifies in control cards and describes the IBM 1401 configurations on which the FORTRAN LCP can be used.

Two appendixes are provided. Appendix A contains a sample converted program. The sample shows the original source program and the System/360 source program generated by the FORTRAN LCP. Appendix B shows the System/360 function names that are equivalent to the function names used in current FORTRAN II compilers.

NATURE OF FORTRAN INFORMATION IN THIS MANUAL

The discussion of each FORTRAN item in this manual is intended primarily to indicate how that item is acted upon by the FORTRAN LCP. To explain conversion actions, many major differences between current FORTRAN II and System/360 FORTRAN IV are described. However, this manual does not delineate all of the differences between a current FORTRAN II compiler and System/360 FORTRAN, nor does the manual list all of the requirements of System/360 FORTRAN.

CONTENTS

The FORTRAN II Language Conversion Program (FORTRAN LCP) is one of a series of programs provided by IBM to assist in transition from a current IBM data processing system to System/360.

The FORTRAN language provided for System/360 is the more powerful FORTRAN IV language. The FORTRAN II languages are sufficiently different from System/360 FORTRAN IV to require some recoding to prepare the earlier programs for compilation on System/360.

The purpose of the FORTRAN LCP is to reduce the time and effort required to convert existing FORTRAN II programs into System/360 FORTRAN IV programs.

To accomplish this, the FORTRAN LCP does the following:

- Recognizes FORTRAN II statements that are incompatible with System/360 FORTRAN.
- When possible, translates incompatible FORTRAN II statements into System/360 FORTRAN statements that have the same meaning and effect.
- Detects and flags FORTRAN II statements that have no System/360 FORTRAN equivalent or cannot be translated into a meaningful sequence of System/360 FORTRAN statements.
- Issues messages indicating incompatibilities that were found and specifying whether conversion action was taken.
- Produces an output listing that always contains the converted FORTRAN II statements and messages that document the conversion actions. When specified, the listing also contains the original source statements.
- Produces a punched deck, when specified, that contains the converted program.

Guided by the messages in the listing, a programmer can make any hand changes required to make his program compatible with System/360 FORTRAN IV.

ACCEPTABLE LANGUAGES

The FORTRAN LCP processes programs written in any of the following current IBM FORTRAN II languages:

IBM 1401 FORTRAN
IBM 1410 FORTRAN II
IBM 1620 GOTRAN
IBM 1620 FORTRAN With FORMAT
IBM 1620 FORTRAN II
IBM 7070-Series FORTRAN
IBM 7070 FOS FORTRAN
IBM 705 FORTRAN
IBM 7080 Processor FORTRAN
IBM 7090/7094 FORTRAN II

Each of these languages is described in one of the publications listed in the Preface under "Prerequisite Literature."

In the remainder of this manual, the phrases "current FORTRAN II compilers" and "current FORTRAN II" are defined to mean the languages listed above.

OUTPUT LANGUAGES

The FORTRAN LCP converts source programs into any of the following System/360 FORTRAN IV languages:

IBM System/360 Basic Programming Support FORTRAN (Level D)
IBM Operating System/360 FORTRAN IV (E Level Subset)
IBM Operating System/360 FORTRAN IV (Level H)

In the remainder of this manual, the term System/360 FORTRAN is used to refer jointly to all three levels of System/360 FORTRAN IV. When referring to a specific level of System/360 FORTRAN IV, one of the following terms is used: Basic Support FORTRAN, Level E FORTRAN, or Level H FORTRAN.

MINIMUM MACHINE REQUIREMENTS

The FORTRAN LCP can be executed on an IBM 1401 Data Processing System or an IBM System/360 with the 1401 Compatibility Feature.

The minimum 1401 configuration required for use of the program is as follows:

1. An IBM 1401 Data Processing System with:
   - 8,000 positions of core storage
   - Advanced Programming Feature
   - High-Low-Equal Compare Feature

2. One IBM 1402 Card Read-Punch
3. One IBM 1403 Printer, Model 2
4. Three IBM 729 Magnetic Tape Units (Models II, IV, V, or VI) or three IBM 7330 Magnetic Tape Units

This minimum configuration allows processing of one source program at a time from the card reader or from tape.

An additional magnetic tape unit is required for each of the following options:

1. Processing stacked source program input from tape.
2. Processing with the FORTRAN LCP system on tape instead of cards.
3. Recording of printed and/or punched output on tape.

The minimum System/360 configuration required for use of this program must be equivalent to the 1401 configuration described above. Information on equivalent System/360 configurations can be found in the publications IBM System/360 Model 30; 1401/1440/1460 Compatibility Feature, Form A24-3255, and IBM System/360, Model 40; Emulation of the IBM 1401/1460 Data Processing System, Form C28-6561.

Details on 1401 processing configurations are provided in the section "Control Cards and Operating Procedures."

TERMINOLOGY

The following discussions concern terms that are used throughout this manual.

INTEGER AND REAL

Many IBM FORTRAN II manuals use the terms fixed-point and floating-point to describe the types of constants, variables, and functions. System/360 FORTRAN uses the terms integer and real in place of fixed-point and floating-point, respectively.

The terms integer and real are used throughout this manual.

SINGLE-PRECISION AND DOUBLE-PRECISION

The terms single-precision and double-precision are used in 7090/7094 FORTRAN II to describe the precision of constants and variables and to indicate the precision of calculations.

In System/360 FORTRAN, length specification and type (integer, real, complex, or logical) determine the precision of a value or a calculation.

The terms single-precision and double-precision have been retained in this manual to aid the programmer who is already familiar with them. As applied to System/360, the term single-precision is used to describe a real value that occupies four bytes, and the term double-precision is used to describe a real value that occupies eight bytes. The terms are also used to describe calculations that result in such values.

CATEGORIES OF FUNCTIONS AND SUBPROGRAMS

This manual uses the following categories for functions and subprograms:

- Arithmetic Statement Functions
- Built-In Functions
- Library Functions
- Function Subprograms
- Subroutine Subprograms

In some current FORTRAN II compilers, there is no distinction between built-in and library functions. In these compilers, only the term library function is used. In 7090/7094 FORTRAN II, built-in functions are compiled as "open" subroutines and appear in the object program each time they are referred to in the source program. Library functions are compiled as "closed" subroutines and appear only once in the compiled program.

The reader should note that the terms built-in function and library function do not appear in System/360 FORTRAN manuals. These functions have been incorporated into System/360 FORTRAN as function subprograms. In addition, the reader should note that the term arithmetic statement function has been shortened to statement function in System/360 FORTRAN.

Thus, System/360 FORTRAN uses only the following three categories for functions and subprograms:

- Statement Functions
- Function Subprograms
- Subroutine Subprograms

The FORTRAN LCP is designed to accomplish a maximum amount of conversion. The program includes a variety of options that permit the user to apply it effectively to a wide range of FORTRAN II conversion needs.

Flexibility has been built into the program in the following major areas:

- Ability to translate from any of ten current IBM FORTRAN II languages into any of three levels of System/360 FORTRAN.
- Provision for use of the program on a variety of 1401 machine configurations.
- Construction of the program so that it will bypass certain processing phases when the functions performed by those phases are not needed.
- Inclusion of messages that indicate all conversion actions and clearly identify statements that could not be converted.
- Provisions for both card and listing output with the ability to suppress card output.

These features of the program are summarized in the paragraphs that follow. When details on a topic are provided in other sections of the manual, a reference is provided to the appropriate section.

CHARACTERISTICS OF SOURCE PROGRAMS TO BE CONVERTED

Each source program submitted to the FORTRAN LCP must have been compiled successfully on the user's current FORTRAN II compiler.

The source program may consist of punched cards or of 80-character unblocked card images on tape.

It is strongly recommended that the source programs be subjected to LCP conversion before any hand changes are made. This takes maximum advantage of the FORTRAN LCP capabilities and precludes hand coding errors that might cause erroneous conversion.

If any hand changes are made prior to conversion, these changes must not make the program unsuitable for compilation on the current IBM compiler for which the program was written originally.

The source program may contain current-monitor or FORTRAN-compiler control cards and non-FORTRAN statements. The manner in which the FORTRAN LCP treats these control cards and statements is explained in the following paragraphs.

- Current-Monitor and Current-Compiler Control Cards: No attempt is made to convert these cards into System/360 control cards. These control cards are merely reproduced in the source program listing. The cards are not included in the punched output.
- Non-FORTRAN Statements: The FORTRAN LCP makes no attempt to convert portions of a FORTRAN program that are written in another language. These statements are reproduced in the source program listing. If specified, non-FORTRAN statements are also punched. The user is responsible for recoding and reassembling non-FORTRAN routines.

The user is cautioned that each FORTRAN program and subprogram is converted separately. This requires that the user provide a separate LCP control card deck for each program and subprogram. Because each program is converted separately, there is no cross-checking between a subprogram and a calling program.

PROCESSING OPTIONS

Conversion can be executed on a variety of 1401 configurations, depending on available equipment and the user's preferences. Details on processing configurations are provided in the section "Control Cards and Operating Procedures."

The input options are as follows:

- System Input Options: The FORTRAN LCP system can be loaded from the card reader or from a magnetic tape unit.
- Source-Program Input Options: The source program to be converted can be read from either the card reader or a magnetic tape unit.

When both the FORTRAN LCP and the source program are on cards, only one source program at a time can be converted. However, when a fourth tape unit is available, stacked programs can be processed from tape or from the card reader.

The output options are as follows:

- Listing Output: The FORTRAN LCP always provides a listing of the converted program. A listing of the original source program is optional and is provided only if specified by the user.
- Punched Card Output: The user can specify that the converted program is to be punched into cards. If desired, non-FORTRAN statements will be punched and separated from converted FORTRAN statements.

Options are available to record punched and/or printed output on magnetic tape. If both forms of output are to be on tape, they will be recorded on the same tape.

PROGRAM ACTIONS

The FORTRAN LCP takes one of four types of action when it recognizes an incompatible source statement. Each type of action is discussed in the following paragraphs.

1. Full Conversion: When possible, the FORTRAN LCP converts an incompatible statement into the appropriate form for System/360 FORTRAN. A message in the output listing indicates that the statement has been converted.

For example, System/360 FORTRAN requires that READ and WRITE statements appear in the FORTRAN IV format.

Thus, a FORTRAN II statement such as:

```
READ INPUT TAPE IN, 45, A, B, C
```

is converted to:

```
READ (IN,45)A,B,C
```

2. Conversion With Warning: In some instances, the FORTRAN LCP converts an incompatible statement to an acceptable System/360 form, but issues a warning message to indicate that the conversion might produce undesirable execution results.

For example, in System/360 FORTRAN, a single-precision real constant may contain no more than seven digits.

When the FORTRAN LCP encounters a single-precision real constant containing more than seven digits, it adds E-exponent notation, but does not truncate the constant.

Thus, the statement:

```
X = 12345678.
```

is converted to:

```
X = 12345678.E0
```

In the output listing, the converted statement is accompanied by a message indicating that the statement may produce undesirable results during execution of the converted program. The warning is issued because the constant may exceed the precision that can be accommodated in a single System/360 machine word.

3. Warning of Possible Incompatibility: The format of a FORTRAN II statement may be entirely acceptable to System/360 FORTRAN, but the effect of the statement in System/360 FORTRAN may differ from its effect on the computer for which it was written.

For example, a FORMAT statement containing A-conversion may be acceptable to System/360 but may not produce the same results it did on a current IBM computer.

Specifically, assume the following coding is used in a 7090 FORTRAN II program to read BCD information from tape records:

```
DIMENSION NAME(2)
      .
      .
      .
10 FORMAT (2A6)
      .
      .
      .
READ INPUT TAPE IN, 10, NAME
```

When the READ statement is executed on a 7090, 12 BCD characters from symbolic tape unit IN are read into two 7090 machine words. (Each 7090 word holds six BCD characters.)

However, on a System/360, the two machine words reserved by the DIMENSION statement will hold only 8 of the 12 characters. (Each System/360 word holds four EBCDIC characters.) If the READ statement were converted and executed on a System/360, the first 2 characters on tape would be skipped, 4 characters would be read into NAME(1), 2 more characters would be skipped, and 4 characters would be read into NAME(2).

The FORTRAN LCP issues a warning message for a statement (such as the FORMAT statement above) when the statement might produce undesirable results on a System/360.

4. Incompatible Item, No Conversion: In some instances, an incompatible statement cannot be translated to an acceptable System/360 form.

For example, Boolean statements in 7090 FORTRAN II cannot be converted because System/360 FORTRAN does not manipulate Boo-

lean functions; instead, it implements logical variables and expressions.

Because of this, the FORTRAN LCP flags each Boolean statement with a message indicating that the statement was not converted.

USE OF THE OUTPUT LISTING

The output listing produced by the FORTRAN LCP always contains the converted program statements and messages generated during conversion. The listing may also contain tables that indicate changes in function names and variable names (see the section "Conversion Output and Messages"). At the user's option, the listing can also contain the original source statements.

Two types of messages appear in the converted program listing -- text messages and numeric message codes. A text message appears beside each statement that is converted by the LCP, generated by the LCP, or recognized as incompatible with System/360 FORTRAN. The numeric message codes (each four digits long) are provided to indicate the exact nature of a conversion action or to flag incompatible items. These message codes will be explained in a later version of this reference manual.

Using the output listing, the programmer can analyze conversion actions and determine whether any hand changes are required. If hand changes are required, they can be inserted into the converted output deck. The deck is then ready for compilation on a System/360 FORTRAN compiler.

GENERAL PROBLEMS IN CONVERTING TO SYSTEM/360 FORTRAN

Conversion to System/360 FORTRAN involves consideration of several general problems caused by differences in machine characteristics and differences among individual FORTRAN II compilers.

These general problems include differences in data file requirements, card codes, precision of calculations, arrangement of arrays, treatment of double-precision and complex values, and variances in the manner in which related COMMON and EQUIVALENCE problems are handled.

The FORTRAN LCP provides a solution for most of these differences. However, some problems either exceed the scope of language conversion, or are not subject to independent logical analysis. In these cases, the LCP signals the user that he must resolve the problem by making hand coding changes after conversion.

The discussions in this section are intended to call attention to these general problems and to indicate those problems for which FORTRAN LCP provides a solution. For certain problems, the user can specify in a control card the manner in which the FORTRAN LCP is to handle the problem. These options are explained within the appropriate discussions.

DATA FILE CONVERSION PROBLEM

Users who are converting programs for System/360 will also be concerned with the problem of data file conversion. Information on this subject will be provided at a later time.

DUAL-CHARACTER-CODE PROBLEM AND OPTIONS

Two basic sets of graphics are associated with the Binary Coded Decimal Interchange Code (BCDIC) used on current IBM computers. The two sets are the H set and the A set. The H set is used in FORTRAN programs.

In five cases, the card code used for a character in the H set is the same as the card code used for a different character in the A set. The conflicting characters, called dual characters in this manual, are:

H-Set Character	A-Set Character	Card Code
+	&	12
=	#	3-8
'	@	4-8
)	¤	12-4-8
(	%	0-4-8

The card code used in System/360 is the Extended Binary Coded Decimal Interchange Code (EBCDIC). In EBCDIC, the codes for the H-set characters have been changed so that each character has a separate and distinct code.

The EBCDIC character and card codes are:

Character	Card Code
+	12-6-8
&	12
=	6-8
#	3-8
'	5-8
@	4-8
)	11-5-8
< (replaces ¤)	12-4-8
(	12-5-8
%	0-4-8

System/360 requires that the proper EBCDIC card code be used for the former H-set dual characters.

FORTRAN LCP Options: The FORTRAN LCP provides control card options that allow the user to designate the type of code contained in his source deck and the type of code he wants punched in his output deck. The code that is present in the source program is designated in one control card, and the desired output code is designated in another control card. (See the section "Control Cards and Operating Procedures.")

By using these control cards, the user can specify one of four input/output combinations for treatment of dual-character codes, as follows:

- BCDIC input with BCDIC output
- BCDIC input with EBCDIC output
- EBCDIC input with BCDIC output
- EBCDIC input with EBCDIC output

MAGNITUDE AND PRECISION PROBLEMS

Differences in machine characteristics and compiler specifications contribute to potential conversion problems in the following FORTRAN areas:

- The magnitude of a constant or variable in a FORTRAN II program may exceed the limits of System/360 FORTRAN.
- The precision of System/360 FORTRAN calculations may be either greater than or less than precision of current FORTRAN II calculations.

These problems are discussed in the following paragraphs.

MAGNITUDE OF CONSTANTS AND VARIABLES

Each FORTRAN II compiler contains restrictions on the magnitude of constants and variables.

In some cases, the permissible magnitude for a current FORTRAN II compiler exceeds the limits of System/360 FORTRAN. This may cause differences in execution results. If a constant or variable in a source program exceeds System/360 limits, the user may want to change the constant or modify his program to make the magnitude compatible with System/360.

The System/360 limits on magnitude are as follows:

Type of Constant or Variable	Limits on Magnitude
Integer	$2^{31}-1$ (or approximately 10^9)
Real	16^{-63} through 16^{63} or 0 (or approximately 10^{75})
Double-Precision	16^{-63} through 16^{63} or 0 (or approximately 10^{75})

PRECISION OF CALCULATIONS

The precision of a FORTRAN calculation depends on the amount of core storage provided to hold the fraction portion of a real value. The greater the number of machine character-positions or binary bits provided for the fraction, the greater the precision of the computed value will be.

The maximum precision that can be achieved in a System/360 is less than the precision that can be achieved in some current IBM computers. This may cause differences in results when the converted program is executed on a System/360.

The System/360 limits on the precision of constants are as follows:

Type of Constant	Limits on Precision
Integer	$2^{31}-1$
Real	1 through 7 decimal digits
Double-Precision	1 through 16 decimal digits

PRECISION OF FUNCTIONS

Some System/360 functions are more precise than the same FORTRAN II functions because new algorithms have been developed for real arguments. This may also cause differences in results when a converted program is executed on a System/360.

FUNCTION-NAME CONFLICTS

The names of FORTRAN functions provided in System/360 FORTRAN differ from the names for the same functions in current FORTRAN II compilers.

Appendix B shows the System/360 function names that are equivalent to function names used in current FORTRAN II compilers. Note that IBM-provided functions are classified as function subprograms in System/360 FORTRAN, whereas these functions were classified as built-in and library functions in current FORTRAN II.

Conversion to System/360 Function Names: The name of each IBM-provided function that appears in a FORTRAN II source program is changed to its proper System/360 name. For example, the function name XABSF is converted to IABS wherever it appears in the source program.

Because the System/360 function names are new, it is possible that a FORTRAN II programmer used one of these names for a function or subprogram that he wrote himself. This could produce a situation in which a user-created function or subprogram name would be the same as a System/360 function name. The FORTRAN LCP takes action to prevent such name conflicts in the converted program.

The possibility also exists that a user-written function or subprogram name is the same as a System/360 reserved word. The FORTRAN LCP also prevents this type of

conflict in a converted program. (Note that there are no reserved words in System/360 Level H FORTRAN. When converting to this level, the FORTRAN LCP does not have to check for conflicts with reserved words.)

Prevention of Name Conflicts: To avoid the above possibilities, the FORTRAN LCP checks each user-written function or subprogram name against a list of System/360 function names and reserved words. If a match occurs, the user-written name is replaced with an LCP substitution name. A unique LCP substitution name is provided for each source program function name that could possibly conflict with a System/360 function name or reserved word. Therefore, the same name is used to replace a particular conflicting name in all programs (and subprograms) processed by the FORTRAN LCP.

Form of LCP Substitution Names: An LCP substitution name is in one of the following three forms:

FCxxP	(Used to replace specific real function or subprogram names)
LCxxP	(Used to replace specific integer function or subprogram names)
SCxxP	(Used to replace specific real function or subprogram names)

The xx portion of the name consists of two digits ranging from 01 to 99. (The fact that two forms are provided for replacement of real function and subprogram names has no significance for the programmer; the two forms are provided merely to facilitate internal LCP processing.)

The user should note that the letter D or C is added to the front of the LCP substitution name when the function or subprogram is double-precision or complex, respectively.

The insertion of an LCP substitution name is illustrated in the example shown below. Note that in the original coding, AIMAG is the name of a user-written function subprogram. The problem results from the fact that AIMAG is also the name of a System/360 function.

Original:

```
Y=AIMAG(X)
```

Converted:

```
Y=FC01P(X)
```

The name FC01P is the LCP substitution name that is used to replace AIMAG wherever it appears in a source program.

Possible Substitution Name Problem: User-created function and subprogram names are also checked against the list of LCP substitution names. If a match is detected, the LCP issues a message to indicate that the user-created name may have to be changed by hand.

For example, assume the source program contains the following statement in which FC01P is a user-created function subprogram name:

```
A=FC01P(B)
```

The function name FC01P need not be changed if that name (which is also an LCP substitution name) has not been inserted into the program to replace a reference to the function AIMAG, and the user's library does not contain a function named AIMAG.

In extreme cases, the program might contain a double conflict. For example, assume that a source program contains the following statement in which AIMAG and FC01P are the names of user-created function subprograms:

```
X=AIMAG(Y)+FC01P(Z)
```

During conversion, AIMAG is replaced by the LCP substitution name FC01P. The statement than contains the same name for two different function subprograms. The converted statement appears as follows:

```
X=FC01P(Y)+FC01P(Z)
```

The FORTRAN LCP detects that the second function name matches an LCP substitution name and issues a warning message. The user would have to change the second function name by hand to remove the conflict.

Note: Besides changing the name in the source program, the user must also change the name of the function in his library.

VARIABLE-NAME CONFLICTS AND OPTIONS

Two conditions can cause a variable name in a source program to be invalid. Those conditions are:

- The variable name is the same as a System/360 function name or reserved word, or
- The variable name is the same as an LCP substitution name.

In either case, the FORTRAN LCP replaces the conflicting variable name with a six-character insert variable.

Form of Insert Variables: There are two forms of insert variables, as follows:

LCPxxx	(Used to replace integer variables)
FCPxxx	(Used to replace real variables)

The xxx portion of both forms consists of three digits. The first insert variable required for a program contains the digits 000, and the count is increased by 1 for each additional variable that is needed.

If a variable name already present in a source program coincidentally matches a potential insert variable, the source program variable is left unchanged and the conflicting insert variable is not used during conversion of the program. For example, if a source program coincidentally contains the integer variable LCP000, the first integer insert variable used during the conversion will be LCP001.

As many as 15 integer variables and 15 real variables can conflict with potential insert variables without affecting conversion. If more than 15 of one type conflict with potential insert variables, the conversion is terminated. A message is issued to indicate that the alphabetic portion of the LCP insert variable must be changed.

Option to Change Insert Variables: The FORTRAN LCP allows the user to change the three alphabetic characters that will appear in each type of insert variable. To change the alphabetic characters to be used in insert variables, the user enters three alphabetic characters in a control card. (See "Control Card Entries" in the section "Control Cards and Operating Procedures.")

Option to Replace Tape References: The FORTRAN LCP also allows the user to specify that a tape reference constant is to be replaced by a variable name or another tape constant. A tape constant is replaced by entering the following two items in a control card: (1) the tape constant as it appears in the source program, and (2) the variable name or tape constant that is to replace that reference whenever it appears in an input/output statement in the program. The FORTRAN LCP stipulates that when a tape constant is to be replaced by a variable name, the variable name must begin with one of the letters I through N.

For example, assume that the user wants to replace references to Tape 6 with the variable IOUT in a program he is converting to System/360 Level H FORTRAN. By specifying the replace option, the user could cause the replacement action shown in the following conversion.

Original:

```
WRITE OUTPUT TAPE 6, 120, G, H, I
```

Converted:

```
DATA IOUT /6/
WRITE (IOUT, 120) G, H, I
```

In specifying replacement of a tape constant, the user causes the constant to be changed every time it is encountered in an input/output statement. The user can specify that five different tape constants are to be modified throughout the program.

ARRANGEMENT OF ARRAYS IN STORAGE

The manner in which arrays are placed in storage varies among the current FORTRAN II compilers. As a result, there are significant differences between the arrangements used by some current compilers and the arrangement used by System/360 FORTRAN compiler.

These differences are of little or no importance to a programmer who uses only FORTRAN coding to manipulate elements in an array. However, the differences are very significant to the programmer who uses non-FORTRAN coding (Autocoder, for example) to manipulate elements, or to the programmer who accesses only one part of a double-precision or complex number.

INTEGER AND REAL (SINGLE-PRECISION) VARIABLES AND ARRAYS

The various ways in which elements are arranged in integer or real (single-precision) arrays are illustrated in Figure 1.

Each compiler controls the placement and manipulation of the elements according to its own conventions. If a programmer uses FORTRAN subscript notation and adheres to compiler rules, the proper elements in an array are used in computations.

However, if a programmer has used non-FORTRAN coding to manipulate the elements, he must be aware of the System/360 array arrangement in order to properly recode the non-FORTRAN portions of his program.

The differences between 7090/7094 arrays and System/360 arrays are even more critical when manipulating elements in double-precision and complex arrays. The 7090/7094 FORTRAN II compiler is the only

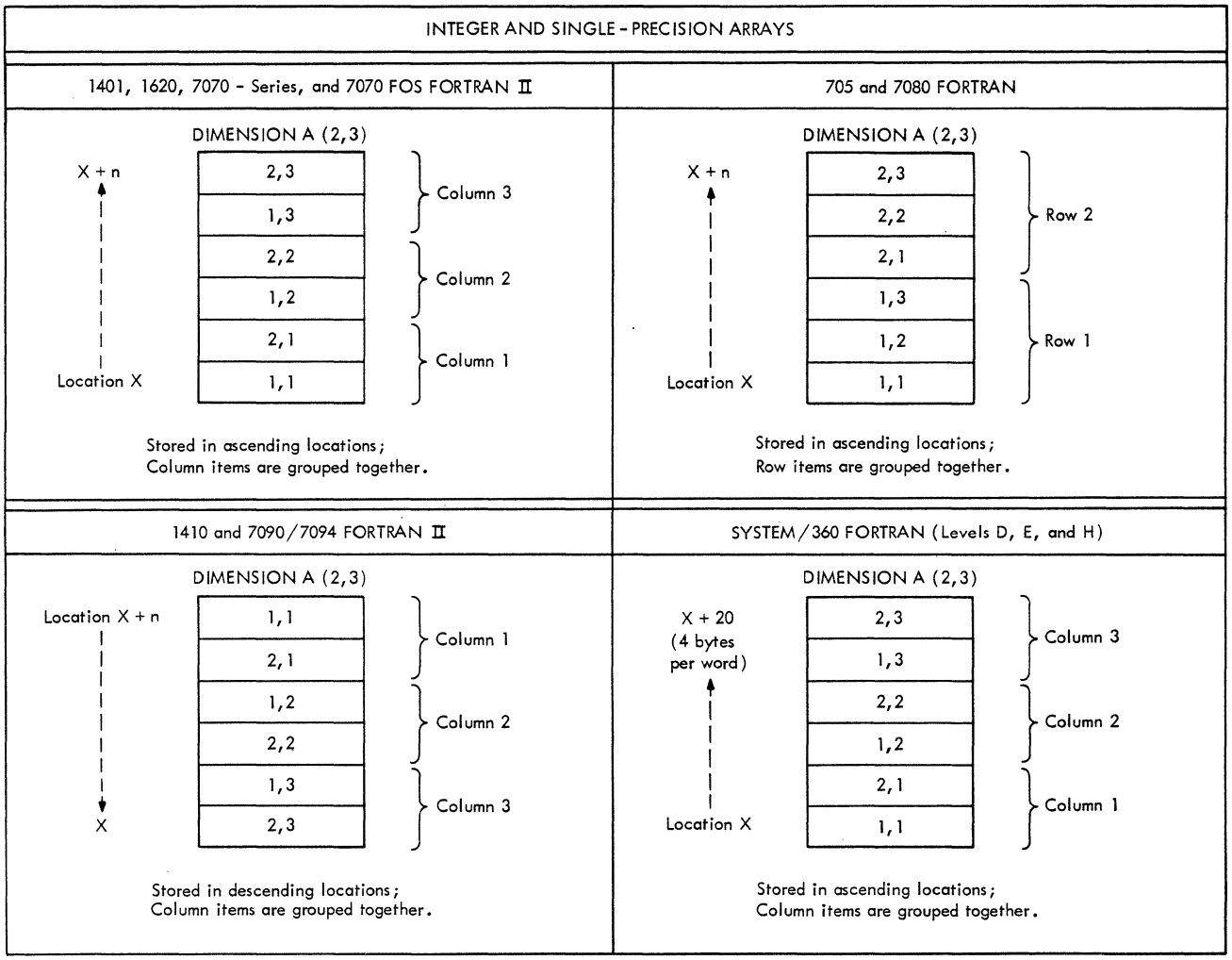

Figure 1. Arrangement of Integer and Single-Precision Arrays in Storage

current FORTRAN II compiler to provide double-precision and complex computations. Therefore, the following discussions apply only to that compiler and the System/360 Level E and Level H compilers.

DOUBLE-PRECISION VARIABLES AND ARRAYS

Because of a difference in the machine-word format of double-precision numbers, the System/360 compilers -- unlike the 7090/7094 FORTRAN II compiler -- do not allow direct access to the least significant portion of a double-precision number.

Differences in Machine-Word Format of Double-Precision Numbers: In the 7090/7094, a double-precision number is stored internally as a pair of real numbers. One 36-bit word contains the most-significant part, and another 36-bit word contains the least-significant part. Each part has its own characteristic and fraction. Both parts of the number are used in double-precision calculations, and it is possible to manipulate either part of the number independently.

In System/360, a double-precision number is stored as an entity in one machine double-word. The first byte of the double-word contains the sign and the characteristic; the other seven bytes contain the fraction. The function SNGL can be used to derive the most-significant part. If the programmer desires the least-significant part, he must code his own

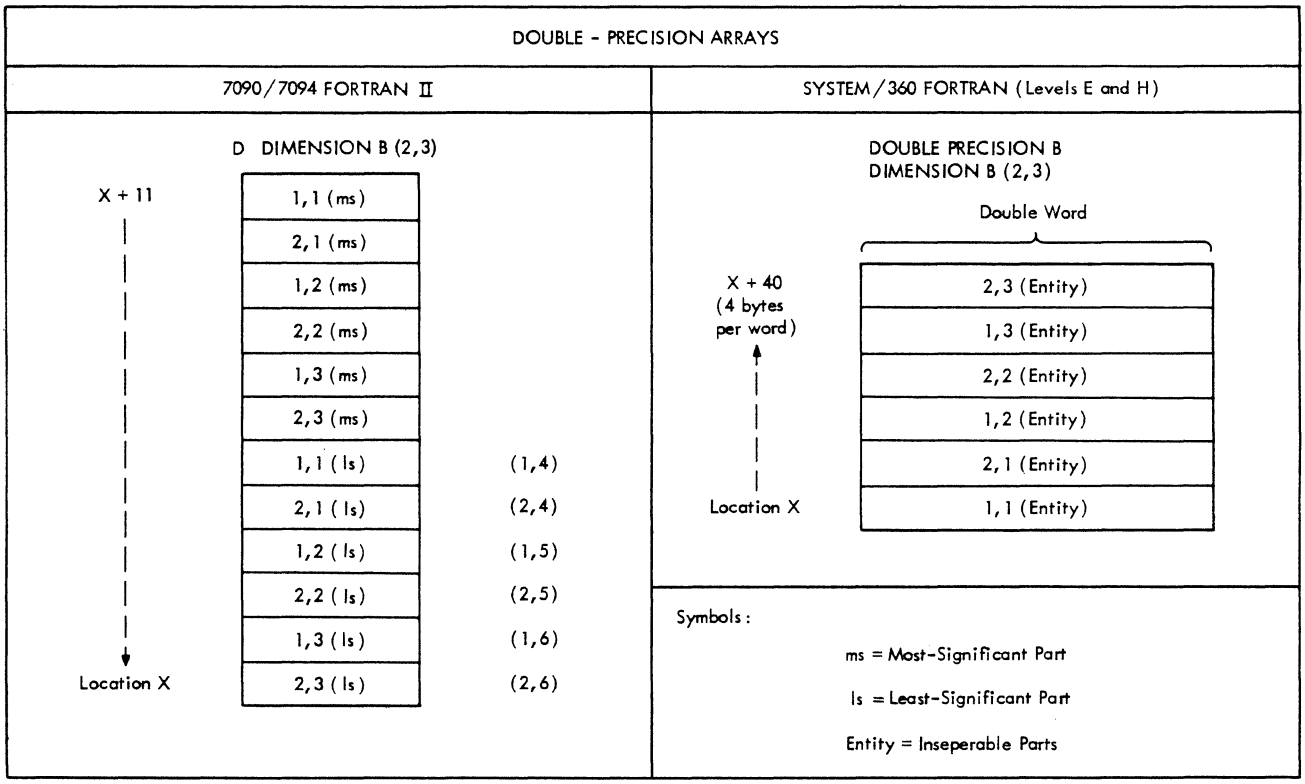

Figure 2. Arrangement of Double-Precision Arrays in Storage

routine to derive it from the double-word entity.

This difference in double-precision format means that a FORTRAN II statement containing a reference to the least-significant part of a double-precision number cannot be translated into a meaningful System/360 FORTRAN statement. Because conversion is not possible, the FORTRAN LCP generates a message to indicate the statement must be changed by hand.

Arrangement of Double-Precision Arrays: Figure 2 shows the arrangement of a double-precision array in 7090/7094 FORTRAN II and the arrangement of the same double-precision array in System/360.

Notice that in the example of the 7090/7094 array, the six double-precision numbers occupy 12 storage locations. The six most-significant parts are followed by the six least-significant parts.

This contrasts with the System/360 double-precision array in which the six double-precision numbers are stored consecutively as six double-word entities.

Input and Output of Double-Precision Numbers: The combined format of double-precision numbers in System/360 also affects the manner in which double-precision values are read into and written out of storage.

In 7090/7094 FORTRAN II, each half of a double-precision number must be read or written separately. There is no provision in the compiler for reading or writing a double-precision quantity as a single entity.

For example, to define and fill a double-precision array the 7090/7094 programmer could use the statements:

```
D       DIMENSION A(10)
        READ TAPE 5, (A(I), I=1, 20)
```

These statements assume that all 10 most-significant parts of the double-precision numbers are recorded on the input tape first, followed by all 10 least-significant parts of the numbers.

The READ statement would be invalid in System/360 FORTRAN because the parts of a double-precision number cannot be read

separately. System/360 FORTRAN requires that a double-precision number be read or written as a single entity.

Consequently, when converting a 7090/7094 program to System/360 FORTRAN, hand changes must be made to input-output items in three areas: (1) The format of double-precision input data must be changed on its external medium; (2) FORMAT statements must be changed; and (3) input/output statements must be changed.

When the FORTRAN LCP encounters an input/output statement that contains a reference to a double-precision variable or array, it issues a message to indicate the statement must be changed by hand. The user is responsible for making hand changes to FORMAT statements and for changing the format of the input data itself.

COMPLEX VARIABLES AND ARRAYS

Both System/360 FORTRAN and 7090/7094 FORTRAN II store the real and imaginary parts of complex variables as separate values.

Machine-Word Formats for Complex Numbers: In the 7090/7094, a complex number is stored internally as a pair of single-precision numbers. One 36-bit word contains the real part and another 36-bit word contains the imaginary part. Either part of the number can be manipulated independently.

In a System/360, a complex number is also stored as a pair of numbers. The real part may occupy either one word or a double-word. The imaginary part occupies the same amount of storage as the corresponding real part.

Nature of the Complex-Number Conversion Problem

Two factors contribute to the conversion problem associated with complex numbers.

One factor involves the different methods by which a 7090/7094 FORTRAN II programmer and a System/360 Level H FORTRAN programmer accesses separate parts of a complex number. The other factor is a significant difference in the sequence in which the compilers store the elements of a complex array.

Accessing Separate Parts of a Complex Number: In 7090/7094 FORTRAN II, entire statements (not individual variable names or arrays) are classified as complex. By using the appropriate subscripts in a non-complex statement, a FORTRAN programmer can access either part of a complex number.

In System/360 FORTRAN, individual variable names, arrays, and functions are typed as complex. To secure the real or imaginary part of a complex number, the FORTRAN programmer must use the functions REAL or AIMAG.

Arrangement of Complex Arrays: Figure 3 shows the arrangement of a complex array in 7090/7094 FORTRAN II and the arrangement of the same array in System/360 Level H FORTRAN.

The sequence of elements in a 7090/7094 complex array is similar to the sequence of elements in a 7090/7094 double-precision array, shown in Figure 2. In a complex array, all of the real parts precede all of the imaginary parts. This is similar to the double-precision array in which all of the most-significant parts precede all of the least-significant parts.

The real and imaginary parts are sequenced differently in a System/360 complex array. In a System/360 array, the real part of each complex number is followed immediately by the imaginary part.

Because of the two factors discussed above, references in a FORTRAN II program to the separate parts of a complex number would be invalid in System/360 Level H FORTRAN unless conversion actions are taken to make them valid.

FORTRAN LCP Actions for Complex Variables and Arrays

The FORTRAN LCP handles the complex-number conversion problem by doubling the dimension of each complex variable and array, and by treating each part of the converted variable or array as a real value.

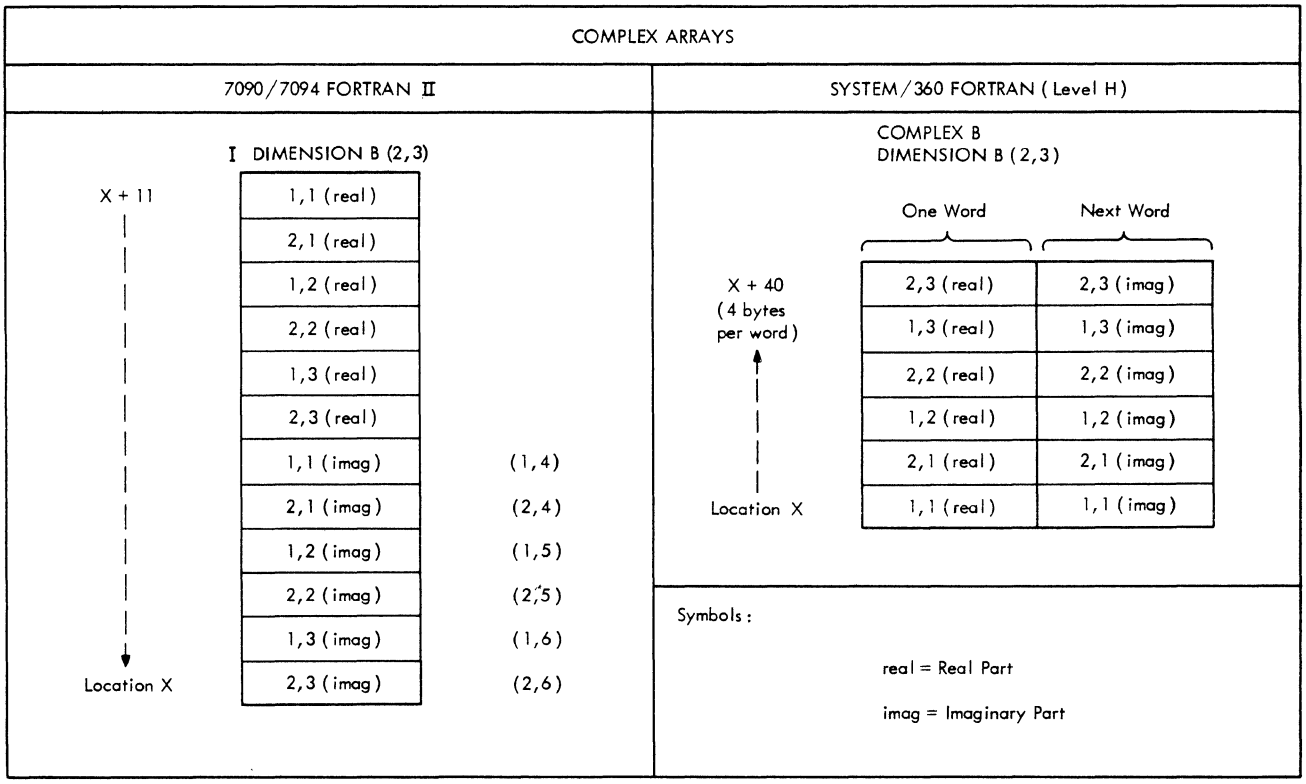

Figure 3. Arrangement of Complex Arrays in Storage

By doubling the dimension of the variable or array, the FORTRAN LCP causes the System/360 compiler to provide the same number of machine words for the array as were provided by the 7090/7094 compiler. This allows the elements in the array to be stored and manipulated in a manner similar to 7090/7094 FORTRAN II.

The significance of the dimension action is illustrated in the following conversion example:

Original:

```
I       DIMENSION B(2,3)
        .
        .
        .
        READ INPUT TAPE 5,10,((B(I,J),
                              I=1,2) J=1,6)
```

Converted:

```
        DIMENSION B(2,6)
        .
        .
        .
        READ (5,10) ((B(I,J),I=1,2) J=1,6)
```

In System/360, the converted DIMENSION statement will create an array consisting of 12 machine words, the same number of words required for the complex array generated from the original statement in 7090/7094 FORTRAN II.

Except for conversion to FORTRAN IV format, the READ statement is unchanged. The implied DO in the statement remains valid. When the READ statement is executed on a System/360, the elements are stored in the sequence shown in Figure 4, which is the same sequence as in the equivalent 7090/7094 array (see Figure 3). Note also that, by doubling the dimension of the array, the FORTRAN LCP has made it possible to read the data without changing the format of the data on its external medium.

Thus, the LCP action ensures that references to one part (real or imaginary) of a complex variable within noncomplex statements in the program remain valid. Such references require no conversion.

Further details on conversion of 7090/7094 complex statements are provided under the heading "Complex Operations" in the section "Conversion Actions for the 7090/7094 FORTRAN II Compiler."

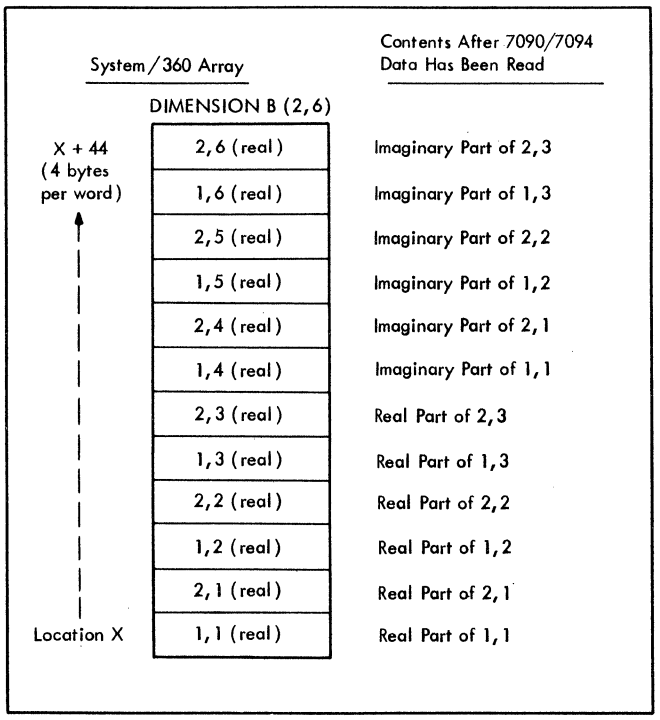

Figure 4. System/360 Array Resulting From Converted DIMENSION Statement

COMMON-EQUIVALENCE INTERACTION PROBLEM AND OPTION

Conversion to System/360 FORTRAN may require resolution of the COMMON-EQUIVALENCE interaction problem. This problem results from the fact that some FORTRAN II compilers give EQUIVALENCE statements precedence over COMMON statements in allocating COMMON storage. In contrast, System/360 FORTRAN gives precedence to the COMMON statements.

This difference can have serious effects on the allocation of COMMON storage by System/360 FORTRAN unless adjustments are made to ensure that the COMMON area is structured in the same way it was structured in FORTRAN II.

The FORTRAN II compilers that give precedence to EQUIVALENCE statements are:

- 1410 FORTRAN II
- 7070-Series FORTRAN
- 7070 FOS FORTRAN
- 7090/7094 FORTRAN II

In converting from one of these FORTRAN languages to System/360 FORTRAN, the user must determine whether or not the equivalencies specified in the FORTRAN II program are to be preserved in the System/360 program.

The COMMON-EQUIVALENCE interaction problem is not involved in conversion from any of the other FORTRAN II languages. The 1620 FORTRAN II compiler gives precedence to COMMON statements. None of the remaining FORTRAN II compilers provide both the COMMON statement and the EQUIVALENCE statement, and some provide neither.

The nature of the COMMON-EQUIVALENCE interaction problem and the manner in which the FORTRAN LCP resolves the problem are discussed in the following text.

NATURE OF THE PROBLEM

In 1410, 7070-Series, 7070 FOS, and 7090/7094 FORTRAN II, EQUIVALENCE statements take precedence over COMMON statements in allocating COMMON storage. When variables named in COMMON statements also appear in EQUIVALENCE statements, the ordinary sequence of the variables in COMMON storage is changed. Priority is given to the variables in EQUIVALENCE statements in the order in which they appear in the EQUIVALENCE statements.

For example, the statements shown below result in the indicated storage allocations:

FORTRAN II	SYSTEM/360 FORTRAN
COMMON A,B EQUIVALENCE (B,C)	COMMON X,Y EQUIVALENCE Y,Z
Storage Allocation	Storage Allocation
1. B.....C 2. A	1. X 2. Y.....Z

Notice that although the variable A is mentioned first in the COMMON statement, FORTRAN II relegates that variable to the second position in COMMON storage. Precedence is given to the EQUIVALENCE statement and the variable B and its equivalent C are located in the first area.

In System/360 FORTRAN, however, the order in which the variables are assigned to COMMON storage is not affected by the equivalencing. System/360 assigns COMMON storage locations in the sequence in which variables are encountered in COMMON statements. Consequently, in the example above, the variable X is assigned first and the equivalenced variables Y and Z are assigned second.

The problem is further complicated when an equivalenced variable also appears in a DIMENSION statement.

FORTRAN II permits the first variable mentioned in a COMMON statement to be equivalenced to a dimensioned variable even though the equivalencing forces repositioning of the first COMMON variable. An example of this follows:

FORTRAN II

```
COMMON D
DIMENSION E(2)
EQUIVALENCE (D,E(2))
```

Storage Allocation

1. E(1)
2. E(2).....D

In giving precedence to the EQUIVALENCE statement, FORTRAN II assigns the first COMMON location to E(1), thus allowing D and E(2) to share the second location.

This usage is not permitted in System/360 FORTRAN. Because System/360 FORTRAN requires that arrays be stored in consecutive forward locations, a variable may not be made equivalent to an element in an array in such a way as to cause the proposed array to extend beyond the beginning of COMMON storage.

A third phase of the COMMON-EQUIVALENCE interaction problem is shown in the following example:

FORTRAN II	SYSTEM/360 FORTRAN
COMMON F,G	COMMON U,V
DIMENSION F(2),H(3)	DIMENSION U(2),W(3)
EQUIVALENCE (F(2),H)	EQUIVALENCE (U(2),W)
Storage Allocation	Storage Allocation
1. F(1)	1. U(1)
2. F(2).....H(1)	2. U(2).....W(1)
3. H(2)	3. V.....W(2)
4. H(3)	4. W(3)
5. G	

The statements in this example cause FORTRAN II to leave a gap in COMMON storage between the locations assigned to the variables F(2) and G. The EQUIVALENCE statement causes H(1), the first element in array H, to share the second location in COMMON storage with the variable F(2). Two locations must be provided for the elements H(2) and H(3). FORTRAN II assigns the variable G to the fifth location, thus avoiding any implied equivalence.

This treatment contrasts with System/360 FORTRAN in which the variable V is assigned to the third location, thus creating an implied equivalence between the variable V and the element W(2).

FORTRAN LCP RESOLUTION OF THE PROBLEM

The user can specify that the FORTRAN LCP is to resolve the COMMON-EQUIVALENCE interaction problem. If the user specifies the reorder COMMON option, the FORTRAN LCP generates new COMMON statements that will cause the System/360 compiler to allocate COMMON storage in the same way it was assigned in FORTRAN II.

Optionally, the user can designate that the LCP is to disregard the interaction problem and allow the COMMON statement to remain unchanged.

When the user specifies that the problem is to be resolved, the three phases of the problem are handled as shown in the examples below. The first three examples show the manner in which the FORTRAN II coding in the preceding examples would be converted.

EXAMPLE 1

Original:	Converted:
COMMON A,B	EQUIVALENCE (B,C)
EQUIVALENCE (B,C)	COMMON B,A
Storage Allocation	Storage Allocation
1. B.....C	1. B.....C
2. A	2. A

EXAMPLE 2

Original:	Converted:
COMMON D	DIMENSION E(2)
DIMENSION E(2)	EQUIVALENCE (D,E(2))
EQUIVALENCE (D,E(2))	COMMON E
Storage Allocation	Storage Allocation
1. E(1)	1. E(1)
2. E(2).....D	2. E(2).....D

EXAMPLE 3

Original:	Converted:
COMMON F,G	DIMENSION F(2),H(3)
DIMENSION F(2),H(3)	EQUIVALENCE (F(2),H)
EQUIVALENCE (F(2),H)	COMMON F,FCP000(2),G
Storage Allocation	Storage Allocation
1. F(1)	1. F(1)
2. F(2).....H(1)	2. F(2).....H(1)
3. H(2)	3. FCP000(1).....H(2)
4. H(3)	4. FCP000(2).....H(3)
5. G	5. G

In Example 3, a dimensioned insert variable is created by the FORTRAN LCP and added to the COMMON statement to ensure that the System/360 compiler places the variable G in the fifth location of COMMON storage.

Note that internal FORTRAN LCP processing normally causes the sequence of COMMON, DIMENSION, and EQUIVALENCE statements to be changed. In the converted program the sequence is: DIMENSION, EQUIVALENCE, and COMMON. This change has no effect upon execution of the converted program. All specification statements (except FORMAT statements) are moved to the beginning of the converted program.

Example 4 shows the resolution of a combination of these problems.

In Example 4, the COMMON-EQUIVALENCE interaction problem is resolved as follows:

1. The array name L is inserted as the first variable in the COMMON statement so that L(1) will occupy the first location in COMMON storage. This permits C(1) to be equivalent to L(2).
2. The positioning of array C now relies on the position of array L and is determined by the EQUIVALENCE statement. Therefore, the array name C is removed from the COMMON statement.
3. The insert variable FCP000 is created and placed in the COMMON statement to force element K(1) into equivalency with C(3).
4. The array name K is inserted into the COMMON statement to provide for consecutive assignment of COMMON storage locations.
5. The variables B and A are placed at the end of the COMMON statement because they do not involve any equivalencies.

EXAMPLE 4

Original FORTRAN II Coding:

```
COMMON B,C,A
DIMENSION C(4),K(3),L(2)
EQUIVALENCE (C,L(2) ),(C(3),K)
```

FORTRAN II Storage Allocation	System/360 Storage Allocation
1. L(1)	1. B.....L(1)
2. L(2)..... C(1)	2. C(1)..... L(2)
3. C(2)	3. C(2)
4. K(1).....C(3)	4. C(3)..... K(1)
5. K(2).....C(4)	5. C(4)..... K(2)
6. K(3)	6. A..... K(3)
7. B	
8. A	

Converted LCP Coding:

```
DIMENSION C(4),K(3),L(2)
EQUIVALENCE (C,L(2) ),(C(3),K)
COMMON L,FCP000,K,B,A
```

System/360 Storage Allocation

1. L(1)
2. L(2) ..... C(1)
3. FCP000....C(2)
4. K(1) .....C(3)
5. K(2) .....C(4)
6. K(3)
7. B
8. A

The Reorder COMMON Option: The FORTRAN LCP provides a control-card option that enables the user to specify whether or not the COMMON-EQUIVALENCE problem is to be resolved. This option is the reorder COMMON option. (See the section "Control Cards and Operating Procedures.")

The lists in this section contain discussions of FORTRAN items that require modification for System/360 FORTRAN.

The lists include items that are converted by the FORTRAN LCP as well as items that require hand conversion. For an item that is converted by the FORTRAN LCP, the discussion explains how the statement is converted and provides a coding example to illustrate the conversion. For an item that cannot be converted, the discussion indicates the reason the conversion cannot be accomplished.

ORGANIZATION OF THE LISTS

The first conversion action list in this section contains discussions of items and statements that must be converted for all current FORTRAN II compilers, or for all compilers except those specifically noted. When the discussion does not apply to a compiler, the exception is noted by the phrase "Not Applicable to..." at the beginning of the text.

Subsequent lists in this section contain discussions of items and statements that apply to a specific FORTRAN II compiler, i,e., 1401 FORTRAN, 1410 FORTRAN II, 1620 GOTRAN, etc.

Thus a FORTRAN II item that must be converted is either discussed in the all-compilers list or a specific-compiler list. A user interested in the conversion action for a particular FORTRAN II item should scan the all-compilers list first. If he fails to find the item there, he should look in the list that applies to his FORTRAN II compiler.

The FORTRAN items in each list are arranged in the same sequence in which they are discussed in the System/360 Level H FORTRAN manual, Form C28-6515.

Within each list, individual items are grouped in the following categories:

1. General Considerations
2. Elements of the Language (constants, variables, and arrays)
3. Arithmetic and Logical Expressions
4. Control Statements (DO, GO TO, IF, etc.)
5. Input/Output Statements (READ, WRITE, FORMAT, etc.)
6. Specification Statements (COMMON, DIMENSION, and EQUIVALENCE)
7. Functions and Subprograms

The FORTRAN LCP does not recognize certain items and statements that are incompatible with System/360 FORTRAN. These items are discussed within the conversion action lists. However, as an aid to the user, these items are also summarized in a separate list entitled "Incompatibilities That Are Not Recognized."

DATA SET TERMINOLOGY

The terms data set and data set reference number are used in the discussion of input/output statements. In System/360 programming, the term data set refers to a named collection of data. A data set may reside on one or more input/output units. A data set reference number refers to the data set itself without regard to the input/output unit (or units) on which the data set resides.

FORM OF CODING EXAMPLES

Coding examples are provided to illustrate the manner in which the FORTRAN LCP modifies a statement.

The format of each coding example is:

Original:

```
FORTRAN CODING AS IT WOULD
APPEAR IN THE SOURCE PROGRAM
```

Converted:

```
CODING AS IT WOULD APPEAR
IN OUTPUT FROM THE FORTRAN LCP
```

The printing format of this manual requires that some coding statements be continued on one or more subsequent lines. These continuations are right-justified, and no continuation indicators are used. The reader should note that the continuations as shown in the examples do not reflect continuations produced by the FORTRAN LCP. All continuation lines generated by the FORTRAN LCP meet System/360 FORTRAN continuation requirements.

CONVERSION ACTIONS FOR CURRENT FORTRAN II COMPILERS

This list contains items that must be converted for all current FORTRAN II compilers, or for all compilers except those cited in a "Not Applicable" statement.

General Considerations

CONTROL CARDS: The FORTRAN LCP does not attempt to translate current-monitor or FORTRAN-II-compiler control cards. These cards are printed in the original source program listing with messages to indicate that they must be either removed or replaced. These cards are not included in converted punched output.

NON-FORTRAN STATEMENTS: The FORTRAN LCP recognizes non-FORTRAN coding within source programs being converted. Some current FORTRAN II compilers permit only subprograms to be written in a non-FORTRAN language while other current compilers permit individual non-FORTRAN instructions to be inserted among FORTRAN statements.

All non-FORTRAN statements are listed in the source program listing. A message appears with a single, non-FORTRAN statement or with the first in a series of non-FORTRAN statements to indicate that the statements must be converted by hand.

The user can specify in a control card that the non-FORTRAN statements are to be punched. If punching of these statements is specified, cards containing non-FORTRAN statements are separated from converted FORTRAN cards.

CONTINUATION CARDS: A converted FORTRAN statement may be longer than the original statement. If necessary, the FORTRAN LCP generates continuation cards to complete the statement.

System/360 FORTRAN permits a maximum of 19 continuation cards. If a converted source statement requires more than 19 continuation cards, the FORTRAN LCP completes conversion of the statement but issues a warning message in the output listing.

BLANKS WITHIN WORDS: Some current FORTRAN II compilers permit embedded blanks within FORTRAN key words (such as READ, WRITE, COMMON, GO TO, etc.) and within names (such as variable names, arrays names, subprogram names, etc.).

The requirements differ for the three levels of System/360 FORTRAN, as follows:

Basic Support and Level E: Embedded blanks are not permitted.

Level H: Embedded blanks are permitted.

When converting to any level of System/360 FORTRAN, the FORTRAN LCP removes embedded blanks from FORTRAN key words and from names that appear in the program. No messages are generated.

Original:

```
TO TAL=A+B
```

Converted:

```
TOTAL=A+B
```

CONDENSING OF OUTPUT STATEMENTS: The FORTRAN LCP removes extraneous blanks from all FORTRAN statements in the converted output program. (Note, however, that blanks are not removed from the literal portion of an H-code specification.)

In condensing the statements, the LCP follows the following conventions:

- One blank follows each FORTRAN key word.
- If the key word that begins a statement starts later than column 7, the key word is moved to column 7.
- Blanks following a delimiter are removed. The delimiters are:
  `+ - * / ) , . (`
- A blank appears before a delimiter only when the delimiter follows a FORTRAN key word.
- Column 73 is also a delimiter. Blanks are inserted so that a name or constant is never split between the end of one coding line and the beginning of the next continuation line.

One exception to these rules is the DO statement in which a blank appears following the statement number that specifies the last statement in the range of the DO loop. The b in the following DO statement shows the location of that blank:

```
DO 22bI=1,9,2
```

The FORTRAN LCP does not generate messages when statements are merely condensed by removing extraneous blanks. The following example illustrates the condensing of a statement.

Original:

```
ROOT1 = (-B + SQRT ((B * B) - 4.0 *
                    A * C)) / ( 2.0 * A)
```

Converted:

```
ROOT1=(-B+SQRT((B*B)-4.0*A*C))/(2.0*A)
```

Elements of the Language

REAL CONSTANTS: Some current FORTRAN II compilers allow a single-precision real constant to contain more than seven decimal digits.

In System/360 FORTRAN, a single-precision real constant is limited to seven decimal digits.

When a single-precision real constant contains more than seven significant digits, the FORTRAN LCP adds a decimal exponent of E0, but does not truncate the constant. A message in the output listing indicates that the statement has been converted.

Original:

```
X = 1234567891.
```

Converted:

```
X=1234567891.E0
```

Addition of the exponent notation of .E0 ensures that the System/360 FORTRAN compiler will compile the constant as a single-precision constant (the E overrides the number of digits). The System/360 compiler retains the magnitude of the constant. However, the System/360 compiler modifies the constant if the number of digits exceeds the precision that can be contained in one System/360 machine word.

EXPONENT NOTATION: Some current FORTRAN II compilers allow the digit 0 to be omitted from decimal exponent notation.

System/360 FORTRAN requires that the zero be included when the exponent is zero.

The FORTRAN LCP adds a 0 to the E or D-exponent notation when the zero is absent. A message in the output listing indicates that the statement has been converted.

Original:

```
D      Y = 2.4E
       X = 2.4E
```

Converted:

```
       DOUBLE PRECISION Y
       Y=2.4D0
       X=2.4E0
```

VARIABLE NAMES: The FORTRAN LCP replaces any variable name that is the same as (1) a System/360 FORTRAN reserved word, (2) a System/360 FORTRAN function name, or (3) an LCP substitution name. The conflicting variable name is replaced by an insert variable created by the FORTRAN LCP.

The form of an insert variable is as follows:

LCPxxx (for replacing integer variables)
FCPxxx (for replacing real variables)

The xxx portion represents three digits varying from 000 through 999. The first insert variable of either form contains 000 and the number is increased by 1 for each additional variable of that form.

Note: In the coding examples in this section, the names LCP000 and FCP000 are used as insert variables for integer variables and real variables, respectively.

Arithmetic Expressions

HIERARCHY OF ARITHMETIC OPERATIONS: The order in which arithmetic operations are executed is important when computing integer values.

All current FORTRAN II compilers specify that operations proceed from left to right in an established order of precedence. However, there are slight variations in the manner in which a series of non-parenthesized operations is implemented.

In System/360 FORTRAN, arithmetic operations at the same hierarchy (except exponentiation) are performed on a strict left to right basis when parentheses are not included to control the order of computation.

The FORTRAN LCP assumes that parentheses were used in arithmetic expressions when the order of computation was important. If the parentheses were omitted, the user must insert them by hand. The FORTRAN LCP does not attempt to provide parentheses and does not generate messages to indicate they are lacking.

Control Statements

ASSIGN STATEMENT: (Not Applicable to 1401 FORTRAN, 1620 GOTRAN, 1620 FORTRAN With FORMAT, or 1620 FORTRAN II) This statement is not provided in System/360 Basic Support or Level E FORTRAN.

The FORTRAN LCP issues a message when it encounters this statement in any source program being converted to Basic Support or Level E FORTRAN. The statement must be converted by hand.

The ASSIGN statement is accepted by Level H FORTRAN.

ASSIGNED GO TO STATEMENT: (Not Applicable to 1401 FORTRAN, 1620 GOTRAN, 1620 FORTRAN With FORMAT, or 1620 FORTRAN II) This statement is not provided in System/360 Basic Support or Level E FORTRAN.

The FORTRAN LCP issues a message when it encounters this statement in any program being converted to Basic Support or Level E FORTRAN.

The Assigned GO TO statement is accepted by Level H FORTRAN.

DO STATEMENT: Some current FORTRAN II compilers permit transfers into the range of a DO loop. The restrictions on such transfers vary among the compilers.

System/360 FORTRAN allows transfer into the range of a DO loop under only one condition. If a transfer was made out of the range of the innermost DO loop in a nest of DOs, transfer can be made back into that innermost DO providing none of the indexing parameters have been changed by statements outside the range of the DO.

If the source program contains a transfer into a DO loop under any condition except that cited above, the user is responsible for changing the program. The FORTRAN LCP does not generate a message when the source program contains an illegal transfer into a DO loop.

MACHINE INDICATOR STATEMENTS: (Not Applicable to 1620 GOTRAN) All current FORTRAN II compilers except 1620 GOTRAN contain statements that set or test machine indicators such as sense lights, overflow indicators, and divide-check indicators.

System/360 FORTRAN provides subroutine subprograms that simulate these tests. These subroutines are referred to by using CALL statements.

The FORTRAN LCP converts an instruction to turn a sense light on or off into a CALL SLITE(i) statement.

Original:

```
100 SENSE LIGHT 3
```

Converted:

```
100 CALL SLITE(3)
```

A source program statement that tests a sense light or indicator is converted to a CALL statement followed by a computed GO TO statement. Insert variables are generated to provide a location to which the subroutine can return a value to reflect the status of the indicator.

Original:

```
110 IF (SENSE LIGHT 2) 111,112
120 IF DIVIDE CHECK 121,122
130 IF ACCUMULATOR OVERFLOW 131,132
140 IF QUOTIENT OVERFLOW 141,142
```

Converted:

```
110 CALL SLITET(2,LCP000)
    GO TO (111,112),LCP000
120 CALL DVCHK(LCP000)
    GO TO (121,122),LCP000
130 CALL OVERFL(LCP000)
    GO TO (131,132,132),LCP000
140 CALL OVERFL(LCP000)
    GO TO (141,142,142),LCP000
```

Note that a third statement number is generated in the GO TO statements related to the accumulator overflow and quotient overflow tests. In System/360 FORTRAN, this is the statement to which transfer is made if an underflow condition is detected. Statements were not available in FORTRAN II to test underflow conditions. The second statement number in these GO TO statements is the statement to which transfer is made if no overflow has occurred. By duplicating the no-overflow address, the FORTRAN LCP will cause underflow conditions to be ignored. If the user desires corrective action to be taken for underflow conditions, he must change the third statement number and provide an appropriate correction routine.

In all cases illustrated above, the FORTRAN LCP generates messages to indicate the statements have been converted.

System/360 FORTRAN has no equivalent for a sense-switch statement. Thus, a statement such as:

```
IF (SENSE SWITCH 5) 121,122
```

cannot be translated into a meaningful System/360 FORTRAN statement. Whenever a

Table 1. Summary of Conversion Actions for Input/Output Statements

Format of FORTRAN II Statement	Format After LCP Conversion to:		
	Basic Support FORTRAN	Level E FORTRAN	Level H FORTRAN
PRINT n, list	WRITE (3,n)list	WRITE (3,n)list	No conversion; Level H accepts this FORTRAN II statement.
PUNCH n, list	WRITE (2,n)list	WRITE (2,n)list	No conversion; Level H accepts this FORTRAN II statement.
READ n, list	READ (1,n)list	READ (1,n)list	No conversion; Level H accepts this FORTRAN II statement.
READ INPUT TAPE i, n, list	READ (i,n)list	READ (i,n)list	READ (i,n)list
READ TAPE i, list	READ (i)list	READ (i)list	READ (i)list
TYPE n, list	WRITE (3,n)list	WRITE (3,n)list	PRINT n,list
WRITE OUTPUT TAPE i, n, list	WRITE (i,n)list	WRITE (i,n)list	WRITE (i,n)list
WRITE TAPE i, list	WRITE (i)list	WRITE (i)list	WRITE (i)list

Symbols:
i = a logical unit number in FORTRAN II and a data set reference number in System/360 FORTRAN.
n = a statement number of a FORMAT statement.
list = a list of variables to which data is to be read or from which data is to be written.
1,2,3 = data set reference numbers.

Note that data set reference numbers are inserted by the LCP during conversion of the PRINT, PUNCH, READ n, and TYPE statements.

sense switch statement is encountered in a source program, the FORTRAN LCP generates a message to indicate that the statement must be changed by hand.

Input/Output Statements

Table 1 summarizes LCP conversion actions for input/output statements that are available in all or most FORTRAN II compilers. The manner in which each FORTRAN II statement is converted is discussed in detail in the paragraphs that follow. For discussions of input/output statements that do not appear in the table, the reader should refer to the conversion action list that contains items that apply specifically to the compiler in which he is interested.

DATA SET REFERENCE CONVENTIONS: Operating System/360 and Basic Programming Support programs contain conventions pertaining to standard input and output units.

Under these conventions, the following data set reference numbers are associated with the indicated system units:

Data Set Reference Number	Associated Unit
1	System input unit
2	System output unit (Punch output)
3	System output unit (Print output)

The unit assignments to specific data set reference numbers can be modified through the use of control cards.

A source program submitted to the FORTRAN LCP may already contain READ and WRITE statements that refer to logical units 1, 2, and 3. In the converted program, these become references to data sets 1, 2, and 3, respectively.

In addition, during conversion of certain incompatible input/output statements (such as PRINT, PUNCH, and TYPE), the FORTRAN LCP creates new input/output statements that also contain references to the data sets 1, 2, and 3.

Thus, conversion may produce situations in which input/output statements that referred to different units in a FORTRAN II program will refer to the same unit in the converted program.

If the user anticipates this problem, he can use the replace tape reference option to change tape references in READ and WRITE statements, or he can check his converted program and change any data set references that are in conflict.

PRINT STATEMENT: The following FORTRAN II format of the PRINT statement is not provided in System/360 Basic Support or Level E FORTRAN:

```
PRINT n, list
```

where n is the number of a FORMAT statement and list is a list of variables from which the data is to be printed.

When converting to Basic Support or Level E FORTRAN, the FORTRAN LCP converts a PRINT statement into a WRITE statement. The number 3 is inserted as the data set reference number. A message in the output listing indicates the statement has been converted.

Original:

```
PRINT 10, A
```

Converted:

```
WRITE (3,10) A
```

The format of the PRINT statement shown above is valid in System/360 LEVEL H FORTRAN and is not modified during conversion to that level. On a System/360, the compiled PRINT statement causes output to be written in the data set associated with system output.

PUNCH STATEMENT: The following FORTRAN II format of the PUNCH statement is not provided in System/360 Basic Support for Level E FORTRAN:

```
PUNCH n, list
```

where n is the number of a FORMAT statement and list is a list of variables from which the data is to be punched.

When converting to Basic Support or Level E FORTRAN, the FORTRAN LCP converts a PUNCH statement into a WRITE statement. The number 2 is inserted as the data set reference number. A message in the output listing indicates the statement has been converted.

Original:

```
PUNCH 10, A
```

Converted:

```
WRITE (2,10) A
```

The format of the PUNCH statement shown above is valid in System/360 Level H FORTRAN and is not modified during conversion to that level. On a System/360, the compiled PUNCH statement causes output to be written in the data set associated with system output.

READ STATEMENT: Current FORTRAN II compilers permit use of the following form of the READ statement when input is to be read from the card reader:

```
READ n, list
```

where n is the number of a FORMAT statement and list is a list of varibles into which the input data is to be read.

System/360 Basic Support and Level E FORTRAN do not recognize the above form of the READ statement. When converting to either of these levels, the FORTRAN LCP changes the format of the statement and inserts the number 1 as the data set reference number. A message in the output listing indicates that the statement has been converted.

Original:

```
READ 10, A
```

Converted:

```
READ (1,10) A
```

System/360 Level H FORTRAN accepts a READ statement in the form READ n, list. On a System/360, the compiled statement causes input to be read from the data set associated with system input.

READ INPUT TAPE STATEMENT: (Not Applicable to 1620 GOTRAN, 1620 FORTRAN With FORMAT, or 1620 FORTRAN II) This statement is used in current FORTRAN II to read data that was

recorded on magnetic tape in external notation. The format of the statement is:

READ INPUT TAPE i, n, list

where i designates the tape unit, n is the number of a FORMAT statement, and list is a list of variables into which the data is to be read.

This form of the READ statement is not provided in System/360 FORTRAN.

The FORTRAN LCP converts a READ INPUT TAPE statement into the following form which is accepted by System/360 FORTRAN:

READ (i,n) list

Note that in this form, i represents a data set reference number instead of designating a tape unit. A message in the output listing indicates that the statement has been converted.

Original:

```
READ INPUT TAPE 7, 100, A, B, C
```

Converted:

```
READ (7,100) A,B,C
```

READ TAPE STATEMENT: (Not Applicable to 1620 GOTRAN, 1620 FORTRAN With FORMAT, or 1620 FORTRAN II) This statement is used in current FORTRAN II to read data that was recorded on magnetic tape in internal notation. The format of the statement is:

READ TAPE i, list

where i designates the tape unit and list is a list of variables into which the data is to be read.

In System/360 FORTRAN, the format of this statement has been changed to:

READ (i) list

Note that in this form, i represents a data set reference number instead of designating a tape unit.

The FORTRAN LCP converts a READ TAPE statement into the proper System/360 format. A message in the output listing indicates that the statement has been converted.

Original:

```
READ TAPE 4, D, E, F
```

Converted:

```
READ (4) D,E,F
```

Data files written in internal notation will require conversion before records in those files can be processed on System/360. Information on data file conversion will be provided at a later time.

TYPE STATEMENT: (Not Applicable to 1401 FORTRAN, 1620 GOTRAN, 705 FORTRAN, or 7090/7094 FORTRAN II) The TYPE statement is used in current FORTRAN II to write output on the console typewriter. The format of the statement is:

TYPE n, list

where n is the number of a FORMAT statement and list is a list of variables from which the data is to be typed.

This statement is not provided in System/360 FORTRAN.

When converting to System/360 Basic Support or Level E FORTRAN, the FORTRAN LCP converts a TYPE statement into a WRITE statement. The number 3 is inserted as the data set reference number.

Original:

```
TYPE 10, X, Y, Z
```

Converted:

```
WRITE (3,10) X,Y,Z
```

When converting to Level H FORTRAN, the FORTRAN LCP converts the TYPE statement into a PRINT statement.

Original:

```
TYPE 10, X, Y, Z
```

Converted:

```
PRINT 10,X,Y,Z
```

In all cases, a message in the output listing indicates that the statement has been changed.

WRITE OUTPUT TAPE STATEMENT: (Not Applicable to 1620 GOTRAN, 1620 FORTRAN With FORMAT, or 1620 FORTRAN II) This statement is used in current FORTRAN II to write data on magnetic tape in external notation. The format of the statement is:

WRITE OUTPUT TAPE i, n, list

where i designates the tape unit, n is the number of a FORMAT statement, and list is a list of variables from which the data is to be written.

This form of the WRITE statement is not provided in System/360 FORTRAN.

The FORTRAN LCP converts a WRITE OUTPUT TAPE statement into the following form which is accepted by System/360 FORTRAN:

```
WRITE (i,n) list
```

Note that in this form, i represents a data set reference number instead of designating a tape unit. A message in the output listing indicates that the statement has been converted.

Original:

```
WRITE OUTPUT TAPE 5, 100, A, B, C
```

Converted:

```
WRITE (5,100) A,B,C
```

WRITE TAPE STATEMENT: (Not Applicable to 1620 GOTRAN, 1620 FORTRAN With FORMAT, or 1620 FORTRAN II) This statement is used in current FORTRAN II to write data on tape in internal notation. The format of the statement is:

```
WRITE TAPE i, list
```

where i designates the tape unit and list is a list of variables from which the data is to be written.

In System/360 FORTRAN, the format of this statement has been changed to:

```
WRITE (i) list
```

Note that in this form i represents a data set reference number instead of designating a tape unit.

The FORTRAN LCP converts a WRITE TAPE statement into the proper System/360 format. A message in the output listing indicates that the statement has been converted.

Original:

```
WRITE TAPE 6, D, E, F
```

Converted:

```
WRITE (6) D,E,F
```

INPUT AND OUTPUT OF ARRAYS: (Not Applicable to 1620 GOTRAN or 1620 FORTRAN With FORMAT) In a current FORTRAN II program, the name of an array can appear in an input/output statement before the array is defined in a DIMENSION statement.

When the array name is used in an input/output statement before it appears in a DIMENSION statement, only the first element in the array is read or written. When the DIMENSION statement precedes use of the array name in an input/output statement, the entire array is read or written.

The FORTRAN LCP reorders the sequence of statements in the source program so that all specification statements (except FORMAT statements) appear before the first executable statement of the program. Thus, all DIMENSION statements are moved to the beginning of the program.

This change may produce a coding sequence that specifies input or output of an entire array when only the first element is desired.

In reordering the specification statements, the FORTRAN LCP does not check to determine whether the reordering will affect input/output references to arrays. The user must check the converted output to determine whether any hand coding changes are required.

REPLACING TAPE REFERENCES: (Not Applicable to 1620 GOTRAN, 1620 FORTRAN With FORMAT, or 1620 FORTRAN II) The FORTRAN LCP provides a control-card option that enables the user to replace a tape constant with a variable name or another tape constant. The tape constant is replaced wherever it appears in an input/output statement.

To specify replacement, the tape constant that appears in the source program and the variable or constant that is to replace it are entered in a control card. Any variable name to be used as a tape reference must begin with one of the letters I through N.

The following paragraphs indicate the manner in which replacement with a variable name is implemented. In the examples, the user has specified that each reference to Tape 6 is to be replaced by the variable IOUT.

When converting to Basic Support or Level E FORTRAN, the input/output statements associated with the tape unit are modified and an arithmetic statement is constructed to assign the proper value to the variable name. The arithmetic statement will precede the first executable statement from the source program.

Original:

```
WRITE OUTPUT TAPE 6, 120, J, K, L
```

Converted:

```
IOUT=6
.
.
.
WRITE (IOUT,120) J,K,L
```

When converting to System/360 Level H FORTRAN, replacement is implemented by modifying each input/output statement associated with Tape 6 and by creating a DATA statement to assign the proper value to the variable name.

Original:

```
WRITE OUTPUT TAPE 6, 120, G, H, I
```

Converted:

```
DATA IOUT /6/
.
.
.
WRITE (IOUT,120) G,H,I
```

Messages in the output listing indicate that the statements have been converted.

Details on specifying replacement of tape references are provided in the section "Control Cards and Operating Procedures."

FORMAT Statements

A-CONVERSION: (Not Applicable to 1620 GOTRAN or 1620 FORTRAN With FORMAT) FORMAT statements containing A-conversion may produce undesirable results in System/360 if the A-conversion specifications are left unchanged.

In current FORTRAN II, the basic field length (that is, the amount of core storage to be provided for a certain type of variable) is either specified by the programmer, pre-established by the compiler, or determined by the fixed word-length of the computer.

In System/360 FORTRAN, a length specification (either implicit or explicit) determines the amount of core storage reserved for each type of variable.

If the basic field length for a variable in current FORTRAN II differs from the length specification for that type of variable in System/360 FORTRAN, the field may be either too large or too small for the number of characters to be read or written using A-conversion specifications.

If the w in the specification nAw exceeds the System/360 length specification, characters will be lost when the data is read. If the w is less than the System/360 length specification, undesired blanks may be created.

Because A-conversion specifications are closely related to data format, the FORTRAN LCP does not attempt to convert such specifications. However, when the FORTRAN LCP encounters an A-conversion specification in the source program, it generates a message to indicate that the statement should be reviewed and hand changes made where necessary.

CARRIAGE CONTROL CHARACTERS: (Not Applicable to 1620 GOTRAN or 1620 FORTRAN II) System/360 FORTRAN does not permit use of the characters 2 through 9 or J through R to control the printer carriage. However, it does permit use of the following characters:

Character	Carriage Advance Before Printing
Blank	Advance one line.
0	Advance two lines.
1	First line of next page.
+	No advance.

The FORTRAN LCP does not check the validity of carriage control characters in source-program statements. The user must check his converted program to ensure that the proper carriage control characters are specified.

READING FORMAT SPECIFICATIONS: System/360 Basic Support and Level E FORTRAN do not permit FORMAT specifications to be read into storage during execution of the FORTRAN program.

When converting to Basic Support or Level E FORTRAN, the FORTRAN LCP issues a warning message whenever it encounters an input/output statement in which a variable appears in the position that normally contains a FORMAT statement number. A variable in this position indicates that the input/output statement utilizes object-time FORMAT information.

Object-time reading of FORMAT information is allowed in System/360 Level H FORTRAN.

Specification Statements

ORDER OF SPECIFICATION STATEMENTS: System/360 Basic Support and Level E FORTRAN require that specification statements (COMMON, DIMENSION, EQUIVALENCE, and Type statements) precede the first executable statement in the program.

To meet this requirement, the FORTRAN LCP reorders statements so that specification statements precede the first executable statement in the program.

In addition, Basic Support and Level E FORTRAN require that the dimension of an array be provided the first time the array name is used in the source program. The FORTRAN LCP provides dimension information for an array in the first specification statement (Type, DIMENSION, or COMMON statement) in which the array name appears in the converted program.

Messages are provided in the output listing.

Functions and Subprograms

TERMINAL F IN FUNCTION NAMES: System/360 FORTRAN does not require that the name of a function end with an F.

The FORTRAN LCP removes the terminal F from each library, built-in, and arithmetic statement function name encountered in a source program. This eliminates the possibility of a seven-character name and also makes the function name in the source program match the name as it appears in the user's library. A message in the output listing indicates the statement has been modified.

In the following conversion example, SINF is the name of an IBM-provided library function and ADFUNCF is the name of a user-written function.

Original:

```
X = Y + SINF (A) * ADFUNCF (Z**2)
```

Converted:

```
X=Y+SIN(A)*ADFUNC(Z**2)
```

CONFLICTING FUNCTION AND SUBPROGRAM NAMES: The FORTRAN LCP detects any user-created function or subprogram name that is the same as a System/360 function name and replaces it with an LCP substitution name. (See the information under the heading "Function-Name Conflicts" in the section "General Problems in Converting to System/360 FORTRAN.")

Whenever the FORTRAN LCP replaces a function or subprogram name, it generates a message to indicate that the statement has been converted.

Original:

```
CALL AIMAG (A,B)
```

Converted:

```
CALL FC01P(A,B)
```

Because the names of IBM-provided functions are different in System/360 FORTRAN from the names used for the same functions in FORTRAN II, the FORTRAN LCP replaces a FORTRAN II function name with the proper System/360 name. A message in the output listing indicates the name has been replaced.

Original:

```
X = Y + LOGF (Z)
```

Converted:

```
X=Y+ALOG(Z)
```

TYPE STATEMENTS FOR FUNCTION NAMES: (Not applicable to 1620 GOTRAN, 1620 FORTRAN II, and 7080 FORTRAN) The initial-letter conventions for establishing the type (integer or real) for user-created built-in, library, and arithmetic statment functions have been changed in System/360 FORTRAN.

Current FORTRAN II compilers designate X as the only initial character that can be used to establish that a function is an integer function (I through N indicate real functions). In System/360 FORTRAN, the initial characters I through N imply that the function is an integer function; all other initial letters (including X) imply the function is real. (Note that System/360 FORTRAN provides implicit and explicit specification statements that may be used to override the initial-letter conventions in establishing the type of a function.)

As indicated earlier, the FORTRAN LCP removes the terminal F from all user-created function names. In addition, the LCP creates an explicit type statement for user-created function names that start with an X, or with I through N. The explicit statement establishes the function as integer or real, respectively.

Original:

```
J = XTRAF (A,B)
Y = INTGRF (C)
```

Converted:

```
INTEGER XTRA
REAL INTGR
J=XTRA(A,B)
Y=INTGR(C)
```

Note: The rules for naming arithmetic statement functions in 1620 FORTRAN II are the same as those in System/360 FORTRAN. Therefore, when converting from this language, the FORTRAN LCP does not generate an explicit type statement for arithmetic statement functions. The same is true for

all functions in 7080 FORTRAN programs that are named according to FORTRAN IV rules.

ARGUMENTS IN ARITHMETIC STATEMENT FUNCTIONS: (Not Applicable to 1401 FORTRAN, 1620 GOTRAN, 1620 FORTRAN With FORMAT, 7070-Series Basic FORTRAN, and 7070 FOS FORTRAN) In current FORTRAN II compilers, there is no specified limit on the number of variables that can appear as dummy arguments in an arithmetic statement function.

In System/360 FORTRAN, a maximum of 15 variables that appear in the expression of an arithmetic statement function can be used as arguments of the function.

The FORTRAN LCP issues a warning message when it encounters an arithmetic statement function in which more than 15 variables are used as dummy arguments. The statement must be converted by hand.

CONVERSION ACTIONS FOR THE 1401 FORTRAN II COMPILER

This list contains conversion actions that apply specifically to the 1401 FORTRAN compiler. Items that apply to all current compilers, including the 1401 compiler, can be found in the list entitled "Conversion Actions for Current FORTRAN II Compilers."

STATEMENTS CONTAINING VARIABLES WITH INSUFFICIENT SUBSCRIPTS: In 1401 FORTRAN, a programmer can use a singly-subscripted reference to a multiply-subscripted variable. An example of this is:

```
DIMENSION J(10,10),K(5,5)
.
.
.
J(1)=Y(1)
```

Such references are invalid in System/360 FORTRAN (except in EQUIVALENCE statements).

When the FORTRAN LCP encounters a singly-subscripted reference in a 1401 FORTRAN program, it checks to determine whether the value of the single subscript is greater than the value of the first subscript for the variable in the DIMENSION statement.

If the single subscript is less than or equal to the first subscript in the DIMENSION statement, the FORTRAN LCP appends sufficient subscripts to make the number of subscripts in the reference equal to the number of subscripts in the DIMENSION statement. Each appended subscript consists of a 1. A message in the output listing indicates that the statement has been converted.

Original:

```
DIMENSION Y(3,3)
X=Y(2)
```

Converted:

```
DIMENSION Y(3,3)
X=Y(2,1)
```

However, if the single subscript exceeds the initial subscript in the DIMENSION statement, the reference is left unchanged. A warning message is issued to indicate that the statement must be changed by hand.

For example, the following coding is left unchanged:

```
DIMENSION Y(3,3)
X=Y(8)
```

Subscripts are also appended to a nonsubscripted reference to a dimensioned variable in an arithmetic or IF statement when the variable is used as an expression in that statement. Note, however, that subscripts are not appended to a nonsubscripted array name that is passed as an argument to a function or subprogram. Messages in the output listing indicate the statements have been converted.

Original:

```
DIMENSION X(10,10), Y(10,10), Z(10,10)
EQUIVALENCE (X(3),XX)
X(I) = Y(I) * Z - SUBF(Y) - COSF(Y(3))
                                * SINF(Y**2)
```

Converted:

```
DIMENSION X(10,10),Y(10,10),Z(10,10)
EQUIVALENCE (X(3),XX)
X(I,1)=Y(I,1)*Z(1,1)-SUB(Y)-COS(Y(3,1))
                          *SIN(Y(1,1)**2)
```

Note that in the third line of converted coding, a subscript has not been appended to the nonsubscripted array name Y where it is passed as an argument to the function SUB. No subscript has been appended to that reference because the function needs only the starting location of the array.

INDEX IN A DO STATEMENT: In 1401 FORTRAN, the index name in a DO statement can be the same as a dimensioned integer variable that appears in a statement outside the DO loop. The index name is the i in the following DO statement format:

```
DO n i = m1, m2, m3
```

The following coding illustrates this usage:

```
    DIMENSION J(10,2)
    .
    .
    .
    DO 10 J=5, 100, 5
    LL = J
10  PRINT 35, LL
    .
    .
    .
    END
```

System/360 FORTRAN does not permit the index name in a DO statement to be the same as a dimensioned variable that appears outside the DO loop.

The FORTRAN LCP detects this violation and issues a message to indicate the statement must be changed by hand.

CONVERSION ACTIONS FOR THE 1620 GOTRAN, 1620 FORTRAN WITH FORMAT, AND 1620 FORTRAN II COMPILERS

This list contains conversion items that apply specifically to 1620 GOTRAN, 1620 FORTRAN With FORMAT, and 1620 FORTRAN II. Items that apply to all current compilers, including the 1620 compilers, can be found in the list entitled "Conversion Actions for Current FORTRAN II Compilers."

An item that applies to only one of the 1620 compilers is marked by the statement "1620 FOTRAN Only", "1620 FORTRAN With FORMAT Only", or " 1620 FORTRAN II Only" at the beginning of the discussion.

Input/Output Statements

Table 2 summarizes LCP conversion actions for specialized input/output statements available in the 1620 GOTRAN and FORTRAN compilers. The manner in which each statement is converted is discussed in detail in the paragraphs that follow. For discussions of other 1620 input/output statements, see "Input/Output Statements" in the section "Conversion Actions for Current FORTRAN II Compilers."

ACCEPT STATEMENT: (1620 FORTRAN With FORMAT and 1620 FORTRAN II Only) In the 1620 FORTRAN compilers, the ACCEPT statement is used to accept input from the console typewriter. The format of this statement is:

```
ACCEPT n, list
```

Table 2. Summary of Conversion Actions for Specialized 1620 Input/Output Statements

Format of 1620 Statements	Statement Available in: 1620 GOTRAN Only	1620 FORTRAN With Format	1620 FORTRAN II	Format After LCP Conversion to: Basic Support FORTRAN	Level E FORTRAN	Level H FORTRAN
ACCEPT n, list		X	X	READ (1,n)list	READ (1,n)list	READ n,list
ACCEPT TAPE n, list		X	X	READ (1,n)list	READ (1,n)list	READ n,list
PLOT (v,c)	X			No meaningful conversion possible	No meaningful conversion possible	No meaningful conversion possible
PRINT, list	X			WRITE (ivar)list	WRITE (ivar)list	WRITE (ivar)list
PUNCH, list	X			WRITE (ivar)list	WRITE (ivar)list	WRITE (ivar)list
PUNCH TAPE n, list		X	X	WRITE (2,n)list	WRITE (2,n)list	PUNCH n, list
READ, list	X			READ (ivar)list	READ (ivar)list	READ (ivar)list

Symbols:
n = a statement number of a FORMAT statement.
list = a list of variables to which data is to be read or from which data is to be written.
ivar = an insert variable that represents a data set reference number.
1,2 = data set reference numbers.

Note: See Table 1 for conversion of: PRINT n, list
PUNCH n, list
TYPE n, list

where n is the number of a FORMAT statement and list is a list of variables containing the data to be typed.

System/360 FORTRAN does not provide this statement.

When converting to System/360 Basic Support or Level E FORTRAN, the FORTRAN LCP converts an ACCEPT statement into a READ statement that contains a data set reference number. The number 1 is inserted as the data set reference number.

Original:

```
ACCEPT 10,A, X
```

Converted:

```
READ (1,10) A,X
```

When converting to System/360 Level H FORTRAN, the FORTRAN LCP converts the ACCEPT statement into a READ statement in which the input unit is implied. The input will be read from the data set associated with system input.

Original:

```
ACCEPT 10, A, X
```

Converted:

```
READ 10,A,X
```

In both cases, a message in the output listing indicates the statement has been converted.

ACCEPT TAPE STATEMENT: (1620 FORTRAN With FORMAT and 1620 FORTRAN II Only) In the 1620 FORTRAN compilers, the ACCEPT TAPE statement is used to read input from a paper tape reader. The format of this statement is:

```
ACCEPT TAPE n, list
```

where n is the number of a FORMAT statement and list is a list of variables into which the data is to be read.

System/360 FORTRAN does not provide this statement.

When converting to System/360 Basic Support or Level E FORTRAN, the FORTRAN LCP converts an ACCEPT TAPE statement into a READ statement that contains a data set reference number. The number 1 is inserted as the data set reference number.

Original:

```
ACCEPT TAPE 50, Q, R, S
```

Converted:

```
READ (1,50) Q,R,S
```

When converting to System/360 Level H FORTRAN, the FORTRAN LCP converts an ACCEPT TAPE statement to a READ statement in which the input unit is implied. The input will be read from the data set associated with system input.

Original:

```
ACCEPT TAPE 60, T, U, V
```

Converted:

```
READ 60,T,U,V
```

In both cases, a message in the output listing indicates the statement has been converted.

PLOT STATEMENT: (1620 GOTRAN Only) In 1620 GOTRAN, the PLOT statement is used to plot curves on the console typewriter. There is no equivalent statement in System/360 FORTRAN.

Whenever the FORTRAN LCP encounters this statement, it issues a message to indicate that the statement is invalid.

PRINT STATEMENT: (1620 GOTRAN Only) In 1620 GOTRAN, the PRINT statement is used to type output data on the console typewriter. The format of the statement is:

```
PRINT, list
```

where list is a list of variables from which the data is to be transmitted.

System/360 does not recognize this form of the PRINT statement.

The FORTRAN LCP converts the statement to a WRITE statement and creates an insert variable to represent the data set reference number. A message in the output listing indicates the statement has been converted.

Original:

```
PRINT, X, Y, Z
```

Converted:

```
WRITE (LCP000) X,Y,Z
```

PUNCH STATEMENT: (1620 GOTRAN Only) In 1620 GOTRAN, the PUNCH statement is used to punch data into cards or paper tape. The form of the statement is:

```
PUNCH, list
```

where list is a list of variables from which the data is to be transmitted.

System/360 FORTRAN does not recognize this form of the PUNCH statement.

The FORTRAN LCP converts the statement to a WRITE statement and creates an insert variable to represent the data set reference number. A message in the output listing indicates that the statement has been converted.

Original:

```
PUNCH, D, E, F
```

Converted:

```
WRITE (LCP000) D,E,F
```

PUNCH TAPE STATEMENT: (1620 FORTRAN With FORMAT and 1620 FORTRAN II Only) In the 1620 FORTRAN compilers the PUNCH TAPE statement is used to punch output on paper tape. The format of this statement is:

```
PUNCH TAPE n, list
```

where n is the number of a FORMAT statement and list is a list of variables from which the data is to be punched.

System/360 FORTRAN does not provide this statement.

When converting to System/360 Basic Support or Level E FORTRAN, the FORTRAN LCP converts a PUNCH TAPE statement to a WRITE statement that contains a data set reference number. The number 2 is inserted as the data set reference number.

Original:

```
PUNCH TAPE 10, X, Y
```

Converted:

```
WRITE (2,10) X,Y
```

When converting to System/360 Level H FORTRAN, the FORTRAN LCP converts a PUNCH TAPE statement to a PUNCH statement in which the output unit is implied. The output is written in the data set associated with system output.

Original:

```
PUNCH TAPE 10, X, Y
```

Converted:

```
PUNCH 10,X,Y
```

READ STATEMENT: (1620 GOTRAN Only) In 1620 GOTRAN, the READ statement is used to accept input from either punched cards, punched tape, or the console typewriter. The format of the statement is:

```
READ, list
```

where list is a list of variables into which the input data is to be read.

System/360 FORTRAN does not recognize the above form of the READ statement.

The FORTRAN LCP changes the form of the statement and creates an insert variable to represent the data set reference number. A message in the output listing indicates that the statement has been converted.

Original:

```
READ, A, B, C
```

Converted:

```
READ (LCP000) A,B,C
```

CONVERSION ACTIONS FOR THE 1410 FORTRAN II COMPILER

This list contains conversion items that apply specifically to the 1410 FORTRAN II compiler. Other items that apply to the 1410 compiler can be found in the list entitled "Conversion Actions for Current FORTRAN II Compilers."

Elements of the Language

ORDER OF ELEMENTS IN ARRAYS: In 1410 FORTRAN, the elements in an array are stored in descending storage locations. (See Figure 1 in the section "Arrangement of Arrays in Storage.") This contrasts with System/360 FORTRAN in which the elements are stored in ascending locations. (Note that in both 1410 FORTRAN and System/360 FORTRAN, the first subscript is varied most rapidly and the last subscript is varied least rapidly.)

No conversion is required because of the different ways in which the two compilers store arrays. Both compilers supply the proper element when subscripts are used in FORTRAN statements.

However, if the user has employed non-FORTRAN statements to manipulate individual elements in an array, the locations referred to in the non-FORTRAN statements may be incorrect. In recoding non-FORTRAN portions of his program, the user must be careful to refer to the elements of an array according to the locations in which they are stored by the System/360 FORTRAN compiler.

The FORTRAN LCP does not analyze non-FORTRAN coding. Therefore, no warning message concerning this difference is issued.

COMMON-EQUIVALENCE INTERACTION PROBLEM: In 1410 FORTRAN, EQUIVALENCE statements are given precedence over COMMON statements in determining the allocation of COMMON storage.

This contrasts with System/360 FORTRAN in which variables and arrays are assigned to COMMON storage in the order in which they appear in COMMON statements.

Because System/360 FORTRAN gives precedence to COMMON statements, the manner in which System/360 FORTRAN allocates COMMON storage for a given set of COMMON and EQUIVALENCE statements may differ significantly from the manner in which COMMON storage was allocated for those statements by the 1410 FORTRAN compiler.

If the user specifies the reorder COMMON option, the FORTRAN LCP resolves this problem by generating new COMMON statements that will cause the System/360 compiler to allocate storage in the same way it is assigned by the 1410 FORTRAN compiler. A message in the output listing indicates new COMMON statements have been generated.

Further information on the COMMON-EQUIVALENCE interaction problem is provided in the section "General Problems in Converting to System/360 FORTRAN."

Input/Output Statements

DIRECT ACCESS STATEMENTS: The FORTRAN LCP does not convert any of the following 1410 FORTRAN direct access statements:

```
DEFINE FILE Statement
FIND Statement
FETCH Statement
RECORD Statement
```

When the FORTRAN LCP encounters any of these statements in a source program, it issues a message to indicate that the statement must be changed by hand.

Functions and Subprograms

F CARDS: In 1410 FORTRAN, an F card is required when the name of a function subprogram or subroutine subprogram appears as an argument in a CALL statement.

F cards are not used in System/360 FORTRAN. Instead, a function subprogram name or a subroutine subprogram name that is used as an argument in a CALL statement must appear in an EXTERNAL statement in the calling program.

The FORTRAN LCP generates an EXTERNAL statement for each F card encountered in the source program and places the EXTERNAL statement among the specification statements at the beginning of the output program. A message in the output listing indicates the statement has been converted.

Original:

```
F       SIN, COS, FUNC
```

Converted:

```
        EXTERNAL SIN,COS,FUNC
```

CONVERSION ACTION FOR THE 7070-SERIES AND 7070 FOS FORTRAN COMPILERS

This list contains conversion actions that apply specifically to the 7070-Series and 7070 FOS FORTRAN compilers. Other items that apply to the 7070-Series and 7070 FOS FORTRAN compilers can be found in the list entitled "Conversion Actions for Current FORTRAN II Compilers."

H-LITERAL IN AN ARITHMETIC STATEMENT: In the 7070 FORTRAN compilers, it is possible to use an H-literal on the right side of the equal sign in an arithmetic statement. An example is:

```
WORD = 4H END
```

This usage is unacceptable in System/360 Basic Support and Level E FORTRAN. When converting to either of these levels, the FORTRAN LCP issues a message to indicate the statement must be changed by hand.

Level H FORTRAN does not accept an H-literal in an arithmetic statement. However, this level does accept an H-literal in a DATA statement.

When converting to Level H FORTRAN, the FORTRAN LCP replaces the H-literal with an insert variable and creates a DATA statement to establish the value of the insert variable. A warning message is issued to indicate that the user should check for incompatibilities between the size of the literal and the length specification for the variable to which the literal is equated.

Original:

```
WORD = 4H END
```

Converted:

```
DATA FCP000/4H END/
WORD=FCP000
```

COMMON-EQUIVALENCE INTERACTION PROBLEM: (Not Applicable to 7070-Series Basic FORTRAN) In 7070-Series Full FORTRAN and in 7070 FOS FORTRAN, EQUIVALENCE statements are given precedence over COMMON statements in determining the allocation of COMMON storage.

This contrasts to System/360 FORTRAN in which variables and arrays are assigned to COMMON storage in the order in which they appear in COMMON statements.

Because System/360 gives precedence to COMMON statements, the manner in which System/360 FORTRAN allocates COMMON storage for a given set of COMMON and EQUIVALENCE statements may differ significantly from the manner in which COMMON storage is allocated for these statements by either 7070 FORTRAN compiler.

If the user specifies the reorder COMMON option, the FORTRAN LCP resolves this problem by generating new COMMON statements that will cause the System/360 compiler to allocate storage in the same way it was assigned by the 7070 FORTRAN compiler. A message in the output listing indicates that new COMMON statements have been generated.

Further information on the COMMON-EQUIVALENCE interaction problem is provided in the section "General Problems in Converting to System/360 FORTRAN."

OVERFLOW INDICATOR TEST STATEMENTS: In the 7070 FORTRAN compilers, the IF ACCUMULATOR OVERFLOW and IF QUOTIENT OVERFLOW statements can be used to test integer and real operations.

In System/360 FORTRAN, the subroutine subprograms that simulate these tests apply only to real operations.

Whenever either of these statements is encountered in a 7070 program, the FORTRAN LCP completes the conversion as indicated under the heading "Machine Indicator Statements" in the section "Conversion Actions for Current FORTRAN II Compilers." However, in the case of the 7070 program, the FORTRAN LCP issues a message with the generated CALL statement. The statement must be changed by hand if it is used to test an integer operation.

CONVERSION ACTIONS FOR THE 705 AND 7080 FORTRAN COMPILERS

This list contains conversion items that apply specifically to the 705 FORTRAN and 7080 Processor FORTRAN compilers. Other items that apply to these compilers can be found in the list entitled "Conversion Actions for Current FORTRAN II Compilers."

An item that applies to the 705 compiler or the 7080 compiler only is marked by the statement "705 FORTRAN Only" or "7080 FORTRAN Only" at the beginning of the discussion.

Elements of the Language

LENGTH OF VARIABLES: (705 FORTRAN Only) In 705 FORTRAN, a variable name can contain as many as ten characters.

System/360 FORTRAN limits a variable name to six characters.

During FORTRAN LCP conversion, any variable name in a 705 FORTRAN program that exceeds six characters is truncated to six characters by removing characters from the right end of the name. A message in the output listing indicates the statement has been changed.

Original:

```
VARIABLE = A + CONSTANT
```

Converted:

```
VARIAB = A+CONSTA
```

The FORTRAN LCP does not check to determine whether a truncated variable name matches another name in the program. The user should check his converted program to ensure that truncation did not create name conflicts.

ARRANGEMENT OF ARRAYS: In 705 and 7080 FORTRAN, the elements in an array are stored so that elements in the same row are together; that is, the elements are stored by varying the last subscript most rapidly and varying the first subscript least rapidly. (See Figure 1 in the section "Arrangement of Arrays in Storage.")

This contrasts with System/360 FORTRAN in which the elements in an array are stored by varying the first subscript most rapidly and the last subscript least rapidly.

(Note that 705 FORTRAN, 7080 FORTRAN, and System/360 FORTRAN all store the elements in ascending storage locations.)

No LCP conversion is required because of the different ways in which the compilers store arrays. The compilers supply the proper elements when subscripts are used in FORTRAN statements.

However, if the user has employed non-FORTRAN statements to manipulate individual elements in an array, the locations referred to in the non-FORTRAN statements may be incorrect.

In recoding the non-FORTRAN portions of his program, the user must be careful to refer to elements of an array according to the locations in which they are stored by the System/360 FORTRAN compiler.

The FORTRAN LCP does not analyze non-FORTRAN coding. Therefore, no warning message concerning this difference is issued.

Control Statements

NEGATIVE INDEXING PARAMETERS IN A DO STATEMENT: (705 FORTRAN Only) In 705 FORTRAN, it is possible to use a negative value for the third indexing parameter. The third indexing parameter is m_3 in the following DO statement form:

```
DO n i = m1, m2, m3
```

Use of the negative parameter allowed the programmer to decrement the value *i* instead of incrementing it.

System/360 requires that the value of all indexing parameters be positive.

The FORTRAN LCP detects the use of a negative constant as the m_3 parameter. The parameter is left unchanged, but the LCP generates a message to indicate that the statement must be changed by hand.

When the m_3 parameter has been specified as a variable, the user should check to ensure that the variable contains a positive value each time the DO loop is entered.

Input/Output Statements

READ (0100) STATEMENT: (7080 FORTRAN Only) In 7080 FORTRAN, a programmer can refer to the card reader in a READ (i,n) list or

READ (i) list statement by designating 0100, either as a constant or a variable, as the input unit. For example, the following statements would result in input from the card reader:

```
J = 0100
READ (J) A, B
READ (0100,10) C, D
```

This usage is unacceptable to System/360 FORTRAN.

The FORTRAN LCP does not recognize this incompatibility. The statements can be changed by using the replace tape reference option provided by the LCP.

WRITE (0500) STATEMENT: (7080 FORTRAN Only) In 7080 FORTRAN, a programmer can refer to the console typewriter in a WRITE (i,n) list or WRITE (i) list statement by designating 0500, either as a constant or a variable, as the output unit. For example, the following statements would result in output to the console typewriter:

```
I = 0500
WRITE (I) W, X
WRITE (0500,10) Y, Z
```

This usage is unacceptable to System/360 FORTRAN.

The FORTRAN LCP does not recognize this incompatibility. The statements can be changed by using the replace tape reference option provided by the LCP.

Specification Statements

VARIABLE DIMENSIONS: (7080 FORTRAN Only) In 7080 FORTRAN, integer variables can be used in a DIMENSION statement to represent the dimensions of an array. For example, a programmer can specify an array as follows:

```
DIMENSION A(J,K/1000)
```

where 1000 specifies the maximum size of the array. By using integer variables as subscripts, the programmer can vary the dimensions of an array during execution of the program.

System/360 FORTRAN allows integer variables to be used as the dimensions of an array only when the DIMENSION statement is in a function subprogram or subroutine subprogram, and the dimensions of the array appear as arguments in a FUNCTION or SUBROUTINE statement.

When the FORTRAN LCP encounters a DIMENSION statement containing variable dimensions, it issues a message to indicate that the statement must be changed by hand.

CONVERSION ACTIONS FOR THE 7090/7094 FORTRAN II COMPILER

This list contains conversion items that apply specifically to the 7090/7094 FORTRAN II compiler. Other items that apply to the 7090/7094 FORTRAN II compiler can be found in the list entitled "Conversion Actions for Current FORTRAN II Compilers."

The organization of this action list varies somewhat from the organization of the preceding lists.

Items that are not specifically related to double-precision or complex number operations are discussed first. These items are arranged according to the type of statement to which they are related. These general items are followed by two separate groups of items: (1) a group of items related to double-precision operations, and (2) a group of items related to complex operations.

Background information on the problems of double-precision and complex operations can be found in the section "General Problems in Converting to System/360 FORTRAN."

Elements of the Language

ORDER OF ELEMENTS IN ARRAYS: In 7090/7094 FORTRAN II, the elements in an array are stored in descending storage locations. (See Figure 1 in the section "Arrangement of Arrays in Storage.)

This contrasts with System/360 FORTRAN in which the elements are stored in ascending locations.

(Note that in both 7090/7094 FORTRAN II and System/360 FORTRAN, the first subscript is varied most rapidly and the last subscript is varied least rapidly.)

No conversion is required because of the different ways in which the two compilers store arrays. Both compilers supply the proper element when subscripts are used in FORTRAN statements.

However, if the user has employed non-FORTRAN statements to manipulate individual elements in an array, the locations referred to in the non-FORTRAN statements may be incorrect.

In recoding non-FORTRAN portions of his program, the user must be careful to refer to the elements of an array according to the locations in which they are stored by the System/360 FORTRAN compiler.

No warning message concerning this difference is issued by the FORTRAN LCP.

Arithmetic and Logical Statements

BOOLEAN STATEMENTS: Boolean statements in 7090/7094 FORTRAN II cannot be converted because System/360 FORTRAN does not manipulate Boolean functions. Instead, System/360 FORTRAN provides for use of logical variables and expressions.

When the FORTRAN LCP encounters a Boolean statement, it generates a message to indicate that the statement must be changed by hand.

H-LITERAL IN AN ARITHMETIC OR IF STATEMENT: In 7090/7094 FORTRAN II, it is possible to use an H-literal in an IF statement or on the right side of the equal sign in an arithmetic statement. An example is:

```
WORD = 4H END
```

This usage is unacceptable in System/360 Basic Support and Level E FORTRAN. When converting to either of these levels, the FORTRAN LCP issues a message to indicate that the statement must be changed by hand.

Level H FORTRAN accepts an H-literal in a DATA statement.

When converting to Level H FORTRAN, the FORTRAN LCP replaces the H-literal with an insert variable and creates a DATA statement to establish the value of the insert variable. A warning message is issued to indicate that the user should check for length incompatibilities.

Original:

```
WORD = 4H END
```

Converted:

```
DATA FCP000/4H END/
WORD=FCP000
```

IMPLICIT MULTIPLICATION: In some cases, 7090/7094 FORTRAN II performs multiplication even though an asterisk has been omitted.

System/360 FORTRAN does not implement implicit multiplication. All desired computations must be specified explicitly.

The FORTRAN LCP inserts an asterisk into an expression at any point at which the asterisk is needed to make multiplication explicit. A message in the output listing indicates the statement has been converted.

Original:

```
X = (A + B) C + 3.D - (E + F) (G + H)
                                + 2.E0X
```

Converted:

```
X=(A+B)*C+3.*D-(E+F)*(G+H)+2.E0*X
```

STATEMENTS CONTAINING VARIABLES WITH INSUFFICIENT SUBSCRIPTS: In 7090/7094 FORTRAN II, a programmer can use a singly-subscripted reference to a multiply-subscripted variable. An example of this is:

```
DIMENSION J(10,10),K(5,5)
.
.
.
J(1)=Y(1)
```

Such references are invalid in System/360 FORTRAN (except in EQUIVALENCE statements).

When the FORTRAN LCP encounters an invalid singly-subscripted reference, it appends sufficient subscripts to make the number of subscripts in the reference equal to the dimensions of the array. Each appended subscript consists of a 1.

Subscripts are also added to a nonsubscripted reference to a dimensioned variable in an arithmetic, IF, or CALL statement when the variable is used as an expression in that statement. Note, however, that subscripts are not appended when a nonsubscripted array name is passed as an argument to a function or subprogram. Messages in the output listing indicate the statements have been converted.

Original:

```
DIMENSION X(10,10,5), Y(10,10), Z(10,10)
EQUIVALENCE (X(3),XX)
X(I) = Y(I) * Z-SUBF(Y) - COSF(Y(3)) *
                                SINF(Y**2)
CALL XYZ (X,Y(3),Z**2)
```

Converted:

```
DIMENSION X(10,10,5),Y(10,10),Z(10,10)
EQUIVALENCE (X(3),XX)
X(I,1,1)=Y(I,1)*Z(1,1)-SUB(Y)
              -COS(Y(3,1))*SIN(Y(1,1)**2)
CALL XYZ (X,Y(3,1),Z(1,1)**2)
```

Note that subscripts have not been appended to the nonsubscripted array name Y where it is passed as an argument to the function SUB or to the nonsubscripted array name X where it is passed as an argument to subprogram XYZ. No subscripts have been appended because the function and subprogram need only the starting location of array Y and array X, respectively.

Control Statements

INDEX IN A DO STATEMENT: In 7090/7094 FORTRAN, the index in a DO statement can be the same as a dimensioned integer variable that appears in a statement outside the DO loop. The index is the i in the following DO statement format:

DO n i = m_1, m_2, m_3

The following coding illustrates this usage:

```
DIMENSION J(10,2)
.
.
.
DO 10 J=5, 100, 5
LL = J
10 PRINT 35, LL
.
.
.
END
```

System/360 FORTRAN does not permit a variable to be used as the index in a DO statement if the variable appears as a dimensioned variable in a statement outside the DO loop.

The FORTRAN LCP detects this violation and issues a message to indicate that the DO statement must be changed by hand.

END STATEMENT: In 7090/7094 FORTRAN II, the END statement can contain a series of program option parameters. These options are meaningless in System/360 FORTRAN.

The FORTRAN LCP blanks out the parameters and generates a message to indicate that the statement has been converted.

Original:

```
END (1,2,0,1,1)
```

Converted:

```
END
```

Input/Output Statements

READ DRUM STATEMENT: In 7090/7094 FORTRAN II, the READ DRUM statement is used to read binary information from a drum storage unit. The format of the statement is:

READ DRUM i, j, list

where i designates the drum unit, j is a drum address, and list is a list of variables into which the information is read.

The FORTRAN LCP converts a READ DRUM statement to a READ statement. An insert variable is created to represent the data set reference number. During conversion of the statement, the FORTRAN LCP also deletes the drum address (j). A message in the output listing indicates the statement has been converted.

Original:

```
READ DRUM K, J, A, B, C, D(3)
```

Converted:

```
READ (LCP000) A,B,C,D(3)
```

Data files written in internal notation will require conversion before records in those files can be processed on System/360. Information on data file conversion will be provided at a later time.

RIT STATEMENT: In 7090/7094 FORTRAN II, it is possible to use an RIT statement in place of a READ INPUT TAPE statement. The RIT statement is converted in the same manner as the READ INPUT TAPE statement (see "Conversion Actions for Current FORTRAN II Compilers"). An example follows.

Original:

```
RIT INTAPE, 60, D, E, F
```

Converted:

```
READ (INTAPE,60) D,E,F
```

Note that in the converted form, INTAPE is an integer variable that represents the data set reference number.

WRITE DRUM STATEMENT: In 7090/7094 FORTRAN II, the WRITE DRUM statement is used to write binary information on a drum storage unit. The format of the statement is:

WRITE DRUM i, j, list

where i designates the drum unit, j is a drum address, and list is a list of variables from which the information is to be written.

The FORTRAN LCP converts a WRITE DRUM statement into a WRITE statement. An integer insert variable is created to represent the data set reference number. During conversion of the statement, the FORTRAN LCP also deletes the drum address (j). A message in the output listing indicates the statement has been converted.

Original:

```
WRITE DRUM 2, 1000, D, E, F, G(6)
```

Converted:

```
WRITE (LCP000) D,E,F,G(6)
```

WOT STATEMENT: In 7090/7094 FORTRAN II, it is possible to use a WOT statement in place of a WRITE OUTPUT TAPE statement. The WOT statement is converted in the same manner as the WRITE OUTPUT TAPE statement (see "Conversion Actions for Current FORTRAN II Compilers"). An example follows.

Original:

```
WOT NOUTTP, 70, R, L, B
```

Converted:

```
WRITE (NOUTTP,70) R,L,B
```

Note that in the converted form, NOUTTP is an integer variable that represents the data set reference number.

OCTAL CONVERSION IN A FORMAT STATEMENT: The 7090/7094 FORTRAN II permits octal conversion to be specified in a FORMAT statement. Octal conversion is not meaningful in System/360 FORTRAN and is not permitted.

The FORTRAN LCP generates a message whenever it encounters a FORMAT statement containing octal-conversion notation. The message is issued to indicate that the statement must be changed by hand.

Specification Statements

MULTIPLE APPEARANCE OF THE SAME VARIABLE IN A COMMON STATEMENT: In 7090/7094 FORTRAN II, the same variable can appear more than once in a COMMON statement.

In System/360 FORTRAN, a COMMON statement is invalid if the same variable appears in it twice.

The FORTRAN LCP reconstructs the COMMON statement and deletes the second and all subsequent references to the variable. A message in the output listing indicates that a new COMMON statement has been generated.

Original:

```
COMMON A, B, C, A, D, E
```

Converted:

```
COMMON A,B,C,D,E,
```

COMMON-EQUIVALENCE INTERACTION PROBLEM: In 7090/7094 FORTRAN II, EQUIVALENCE statements are given precedence over COMMON statements in determining the allocation of COMMON storage.

This contrasts with System/360 FORTRAN in which variables and arrays are assigned to COMMON storage in the order in which they appear in COMMON statements. Consequently, the manner in which System/360 FORTRAN allocates COMMON storage for a given set of COMMON and EQUIVALENCE statements may differ significantly from the manner in which COMMON storage was allocated for those statements by the 7090/7094 FORTRAN II compiler.

If the user specifies the reorder COMMON option, the FORTRAN LCP resolves this problem by generating new COMMON statements that will cause the System/360 compiler to allocate storage in the same way it is assigned by the 7090/7094 FORTRAN II compiler. A message in the output listing indicates that new statements have been generated.

Further information on the COMMON-EQUIVALENCE interaction problem is provided in the section "General Problems in Converting to System/360 FORTRAN."

FREQUENCY STATEMENT: The FREQUENCY statement is used in 7090/7094 FORTRAN II programs to optimize object coding. The statement indicates the estimated frequency with which specific control branches will be taken.

The statement is not provided in System/360 FORTRAN.

The FORTRAN LCP converts a FREQUENCY statement into a comment card. A message in the output listing indicates the statement has been converted.

Original:

```
      FREQUENCY 30(1,2,1),40(11),50
                              (1,7,1,1)
```

Converted:

```
C     FREQUENCY 30(1,2,1),40(11),50
                              (1,7,1,1)
```

Functions and Subprograms

F CARDS: In 7090/7094 FORTRAN, an F card is required when a library function name, function subprogram name, or a subroutine subprogram name is used as an argument to another function subprogram or subroutine subprogram.

F cards are not used in System/360 FORTRAN. Instead, a function or subprogram name that is used as an argument to a function or subroutine subprogram must appear in an EXTERNAL statement in the calling program. (System/360 FORTRAN classifies an EXTERNAL statement as a specification statement. Basic Support and Level E FORTRAN stipulate that EXTERNAL statements must appear prior to the first executable statement in a program.)

The FORTRAN LCP generates an EXTERNAL statement for each F card encountered in the source program and places the EXTERNAL statement among the specification statements at the beginning of the output program. A message in the output listing indicates that EXTERNAL statements have been converted.

Original:

```
F     SIN, COS, FUNC
```

Converted:

```
      EXTERNAL SIN,COS,FUNC
```

REPLACING ARGUMENTS IN ARITHMETIC STATEMENT FUNCTIONS: In 7090/7094 FORTRAN II, arguments that appear in arithmetic statement functions are dummy variables and may be the same as names that appear elsewhere in the program.

However, conflicts may arise when the FORTRAN LCP generates Type statements for double-precision and complex variables. Consider the following 7090/7094 FORTRAN II coding:

```
D     DIMENSION X(10,10)
      FIRSTF(X,I)=A + X**I
```

The variable X in the DIMENSION statement is a double-precision variable. The argument X in the arithmetic statement function is a single-precision dummy argument. This causes no problem in 7090/7094 FORTRAN II.

However, during conversion, the FORTRAN LCP generates an explicit DOUBLE PRECISION statement that establishes the variable X as a double-precision variable throughout the program, even in its use as a dummy argument.

When a 7090/7094 program contains double-precision or complex operations, the FORTRAN LCP replaces all dummy arguments in arithmetic statement functions with insert variables. A message in the output listing indicates the statement has been converted.

Original:

```
D      DIMENSION X(10,10)
       FIRSTF (X,I) = A+X**I
```

Converted:

```
       DOUBLE PRECISION X
       DIMENSION X(10,10)
       FIRST (FCP000,LCP000)=A+FCP000
                             **LCP000
```

DOUBLE-PRECISION OPERATIONS

This list contains items that apply specifically to double-precision operations.

System/360 Basic Support FORTRAN does not permit use of double-precision variables, arrays, or statements.

SPECIFYING DOUBLE-PRECISION VARIABLES AND ARITHMETIC STATEMENT FUNCTIONS: In 7090/7094 FORTRAN, entire statements are designated as double-precision by inserting the letter D in column 1.

System/360 FORTRAN requires that variables, arithmetic statement function names, and function subprogram names be designated individually as double-precision. The DOUBLE PRECISION type statement provides one means of doing this.

The FORTRAN LCP generates DOUBLE PRECISION statements that include each variable name and arithmetic statement function name found in 7090/7094 double-precision statements.

Original:

```
D      ALPHA = BETA * GAMMA + FUNCTF(Y,Z)
```

Converted:

```
       DOUBLE PRECISION ALPHA, BETA,
                        GAMMA,FUNCT,Y,Z
       ALPHA=BETA*GAMMA+FUNCT(Y,Z)
```

Note that FUNCT is the name of an arithmetic statement function.

REAL CONSTANTS IN DOUBLE-PRECISION STATEMENTS: In 7090/7094, a real constant containing E-exponent notation can appear in a double-precision statement. The use of the letter D in column 1 causes the constant to be treated as double-precision.

By removing the D from column 1 of double-precision statements, the FORTRAN LCP creates the need for explicit specification of double-precision constants. This problem is solved by inserting D-exponent notation into each real constant found in a double-precision statement.

Original:

```
D      A = 1567.0E15 * X + 2.4
```

Converted:

```
       DOUBLE PRECISION A,X
       A=1567.0D15*X+2.4D0
```

REFERENCES TO THE MOST-SIGNIFICANT PART OF A DOUBLE-PRECISION VARIABLE: In 7090/7094 FORTRAN II, a programmer can refer to the most-significant part of a double-precision variable by referring to that variable in a single-precision statement.

In System/360 FORTRAN, a double-precision number is stored as an entity. A programmer cannot directly access the most-significant or least-significant part of the number. However, by using the function SNGL, the programmer can derive the most significant part from the double-precision entity.

The FORTRAN LCP uses the function SNGL to translate a reference to the most-significant part of a double-precision variable in a statement without a D in column 1.

When the reference in such a statement is used in any manner except as an argument to a function or subprogram, the FORTRAN LCP makes the variable an argument to the function SNGL. During execution of the converted statement, the desired part of the double-precision variable is returned to the object program.

The variable is also made an argument to the function SNGL when it is in an expression used as an argument to another function or subprogram (see example below). However, when the variable name stands alone as an argument to a function or subprogram, the reference is not altered.

Original:

```
D      B = C * D
       IF (B) 5,5,10
     5 CALL XYZ (B, C**2)
    10 AB = C * D - SINF(B)
```

Converted:

```
      DOUBLE PRECISION B,C,D
      B=C*D
      IF (SNGL(B)) 5,5,10
    5 CALL XYZ(B,SNGL(C)**2)
   10 AB=SNGL(C)*SNGL(D)-SIN(B)
```

REFERENCES TO THE LEAST-SIGNIFICANT PART OF A DOUBLE-PRECISION VARIABLE: In System/360 FORTRAN, it is not possible to refer to the least-significant portion of a double-precision variable.

When the FORTRAN LCP detects that a subscripted variable definitely refers to the least-significant part of a double-precision number, it issues a message to indicate that the statement must be changed by hand.

However, when variables are used as subscripts to a double-precision variable, the FORTRAN LCP cannot determine whether the values of the subscripts will refer to the most-significant or least-significant part of a number. In this case, LCP translates the statement by using the function SNGL.

Consider the following example:

```
D     DIMENSION A(5,5), C(3)
      DIMENSION B(10)
      DO 7 J=1, 10
    7 B(J) = A(I,J)
    8 D = C(I)
    9 E = C(4)
```

The FORTRAN LCP can determine immediately that the subscript of C in statement 9 is a reference to the least-significant part of a number. The statement is not translated and is flagged for hand conversion.

However, the FORTRAN LCP cannot determine whether the subscripts in statements 7 and 8 refer to the most-significant or least-significant part of the numbers. Those two statements are translated by using the function SNGL and warning messages are issued to indicate that the translations may be invalid. The conversion is shown below.

Original:

```
D     DIMENSION A(5,5), C(3)
      DIMENSION B(10)
      DO 7 J=1, 10
    7 B(J) = A(I,J)
    8 D = C(I)
    9 E = C(4)
```

Converted:

```
      DOUBLE PRECISION A(5,5),C(3)
      DIMENSION B(10)
      DO 7 J=1,10
    7 B(J)=SNGL(A(I,J))
    8 D=SNGL(C(I))
    9 E=C(4)
```

Warning messages appear with statements 7, 8, and 9 in the output listing.

If a reference containing variable subscripts appears on the left side of an arithmetic statement, the statement is not converted. This reference indicates that a quantity is to be stored in one part of a double-precision number. The FORTRAN LCP issues a message to indicate that the statement must be changed by hand.

Original:

```
D     DIMENSION A(5,5,5)
      A(I,J,K)=B**2
```

Converted:

```
      DOUBLE PRECISION A(5,5,5)
      A(I,J,K)=B**2
```

DOUBLE-PRECISION VARIABLES IN INPUT/OUTPUT LISTS: In 7090/7094 FORTRAN II, each part of a double-precision number is read or written separately.

In System/360 FORTRAN, a double-precision number is read or written as an entity.

In converting input/output statements, the FORTRAN LCP does not attempt to modify references to double-precision variables or double-precision arrays. When such references are encountered, the FORTRAN LCP generates messages to indicate that the statements must be changed by hand.

Consider the following conversion:

Original:

```
D     DIMENSION H(10)
D     A = B * C
   13 WRITE TAPE 6, A(1), A(2)
   14 WRITE TAPE 6, A, C(1)
   15 READ TAPE 5, (H(I), I=1, 20)
   16 WRITE TAPE 6, B(2), (H(I),
                      H(I+10), I=1, 10)
```

Converted:

```
      DOUBLE PRECISION A,B,C,H(10)
      A=B*C
   13 WRITE (6)A(1),A(2)
   14 WRITE (6)A,C(1)
   15 READ (5) (H(I),I=1,20)
   16 WRITE (6)B(2),(H(I),H(I+10),
                                 I=1,10)
```

Note that the DOUBLE PRECISION statement is generated and that the READ and WRITE portions of the statements are converted. However, all references to double-precision variables and arrays are unchanged. Warning messages are generated in the output listing for statements 13, 14, 15, and 16.

The difference in the manner in which double-precision variables and arrays are read and written also requires that FORMAT statements be changed. Consider the following coding:

```
D       A = B * C
        WRITE OUTPUT TAPE 5,10,A(1),A(2)
     10 FORMAT (2012)
```

The FORTRAN LCP cannot recognize that the conversion specified in the FORMAT statement pertains to double-precision variables and therefore cannot translate the statement. Whenever there is a double-precision variable or array in an input/output list, a message is generated to indicate that the FORMAT statement associated with the input/output statement must be changed by hand.

COMMON STATEMENTS CONTAINING DOUBLE-PRECISION VARIABLES: In 7090/7094 FORTRAN II, no boundary problems are involved when a COMMON statement contains a mixture of single-precision and double-precision variables.

This contrasts with System/360 FORTRAN in which boundary requirements may affect the validity of a COMMON statement that contains both single-precision and double-precision variables.

Consider the following coding in two subroutine subprograms:

Subprogram 1:

```
        SUBROUTINE SUB1
        COMMON R,S,T
D       A=S**2
```

Subprogram 2:

```
        SUBROUTINE SUB2
        COMMON X,Y,Z
        DIMENSION Y(2)
```

In 7090/7094 FORTRAN II, COMMON storage is allocated so that R is equivalent to X, S to Y, and T to Z.

In System/360 FORTRAN, the COMMON statement in Subprogram 1 would be invalid because the double-precision variable S would not begin on a double-word boundary. The System/360 FORTRAN compilers begin COMMON storage on a double-word boundary and assign variables consecutively. To remedy the boundary problem and to preserve the implied equivalencies between the subprograms, the programmer could change the order of the variables in the COMMON statements so that T appeared before S, and Z appeared before Y.

In making hand changes to compensate for boundary problems, the user must take care not to destroy implied equivalencies between one program and another. For example, if the user attempted to compensate for the boundary problem in Subprogram 1 by inserting a dummy variable between R and S, the implied equivalencies between Subprogram 1 and Subprogram 2 would be destroyed.

The boundary problem is also involved in use of the reorder COMMON option. In reordering variables in COMMON statements, the FORTRAN LCP does not ensure that a double-precision variable will begin on a System/360 double-word boundary.

When converting double-precision programs, the FORTRAN LCP issues a message to indicate that COMMON statements should be checked and hand changes made where necessary.

EQUIVALENCE STATEMENTS CONTAINING DOUBLE-PRECISION VARIABLES: In 7090/7094 FORTRAN, a single-precision variable can be equivalenced to the least-significant portion of a double-precision variable. Consider the following coding:

```
        EQUIVALENCE (A(2),E)
D       A = B**2
```

In 7090/7094 FORTRAN, the EQUIVALENCE statement specifies that the variable E is to share COMMON storage with the least-significant part of A (assuming that A is not an array). In System/360 FORTRAN, it is not possible to directly reference the least significant part of a double-precision variable.

The FORTRAN LCP generates a warning message when it encounters a statement that equivalences a variable to the least-significant portion of a double-precision variable. The statement must be changed by hand.

DOUBLE-PRECISION FUNCTION NAMES: The FORTRAN LCP adds the letter D to the beginning of the name of a library or built-in function wherever that name appears in a

double-precision statement. The letter F at the end of the name is removed.

These actions make the source program name the same as the name of the function on the library tape.

A message in the output listing indicates the statement has been converted.

Original:

```
D       Y = SINF(X)
```

Converted:

```
        DOUBLE PRECISION Y,X
        Y=DSIN(X)
```

DOUBLE-PRECISION FUNCTION SUBPROGRAMS: In converting a double-precision function subprogram, the FORTRAN LCP specifies the type of the function as part of the FUNCTION statement. A message in the output listing indicates that the FUNCTION statement has been converted.

Original:

```
        FUNCTION SUB1 (X,Y)
        .
        .
        .
D       SUB1=X+Y
        .
        .
        .
D       RETURN
        END
```

Converted:

```
        DOUBLE PRECISION FUNCTION SUB1
                                        (X,Y)
        DOUBLE PRECISION X,Y
        .
        .
        .
        SUB1=X+Y
        .
        .
        .
        RETURN
        END
```

COMPLEX OPERATIONS

This list contains conversion items that apply specifically to complex arithmetic operations.

Incompatibilities concerning complex arithmetic operations result from (1) differences in the manner in which 7090/7094 FORTRAN II and System/360 FORTRAN store the elements of a complex array, and (2) differences in the manner in which a programmer accesses parts of a complex number. These differences are discussed in the section "Arrangement of Arrays in Storage."

The FORTRAN LCP resolves most of these incompatibilities by doubling the dimension of each complex variable or array and by treating both parts of a complex number as real numbers.

When complex arithmetic statements are encountered during the conversion, the FORTRAN LCP uses the functions REAL, AIMAG, and CMPLX to accomplish the complex calculations. The function CMPLX is used to derive a complex value from the two real numbers that represent the FORTRAN II complex number. Once the complex value has been derived, the functions REAL and AIMAG are used to derive the parts. Each part, however, is stored as a real number. Insert variables are created where necessary to facilitate the conversion and to make the converted coding more efficient.

System/360 Level H FORTRAN is the only level that permits use of complex variables, arrays, or statements. When converting to Basic Support or Level E FORTRAN, the FORTRAN LCP generates a warning message whenever it encounters a complex statement in the source program. The statement must be converted by hand.

COMPLEX VARIABLES IN ARITHMETIC STATEMENTS: In 7090/7094 FORTRAN II, entire statements are designated as complex by inserting an I in column 1 of the statement.

The FORTRAN LCP removes the I from column 1 and treats both parts of a complex variable as real variables. References to a complex variable are translated by using the functions REAL, AIMAG, and CMPLX.

Original:

```
I       DIMENSION A(4,4)
I   10 A(I,J) = B**2
```

Converted:

```
        DIMENSION A(4,8),B(2)
        COMPLEX FCP000
    10  FCP000=CMPLX(B(1),B(2))**2
        A(I,J)=REAL(FCP000)
        A(I,J+4)=AIMAG(FCP000)
```

The dimensions of array A and variable B are doubled. An insert variable is created and established as a complex variable by using the explicit type statement COMPLEX. This insert variable is created to avoid duplicate computations.

Statement 10 causes a complex value to be computed and stored in the complex variable FCP000. The real part of that variable is then stored in A(I,J) and the imaginary part is stored in A(I,J+4).

Note: In the above example, statement 10 might be the last statement in the range of a DO loop. The FORTRAN LCP flags this statement to indicate that the user should check to determine whether the generation of additional statements has affected the range of a DO loop. See the discussion of "Statement Numbers With Complex Statements."

COMPLEX DATA IN IF STATEMENTS: In 7090/7094 FORTRAN II, complex constants and variables can be used in the expression of an arithmetic IF statement.

This is invalid in System/360 FORTRAN.

The FORTRAN LCP issues a warning message whenever it encounters an IF statement that contains an I in column 1. The user must change the statement by hand.

COMPLEX VARIABLES IN NONCOMPLEX STATEMENTS: If a reference to a complex variable appears in a 7090/7094 FORTRAN II statement that lacks an I in column 1, the reference is left unchanged.

Original:

```
I       DIMENSION A(5,5,5),C(3)
        BB = A(I,J,K) +C(4)
```

Converted:

```
        DIMENSION A(5,5,10),C(6)
        BB=A(I,J,K) +C(4)
```

The FORTRAN LCP treatment of complex variables and arrays is such that the subscripts of a complex variable in noncomplex statements and input/output statements remain valid in the converted program.

STATEMENT NUMBERS WITH COMPLEX STATEMENTS: In converting complex arithmetic statements, the FORTRAN LCP replaces the original statement with several new statements.

If a statement number is associated with the complex arithmetic statement, that number is assigned to the first of the new statements. This creates a problem if the original statement is the last statement in the range of a DO loop. Consider the following conversion:

Original:

```
I       DIMENSION A(10), B(10)
        DO 100 K=1, 10
        .
        .
        .
        IF (X-Y) 100,200,100
        .
        .
        .
I 100   A(K) = B(K)
```

Converted:

```
        DIMENSION A(20),B(20)
        COMPLEX FCP000
        DO 100 K=1,10
        .
        .
        .
        IF (X-Y) 100,200,100
        .
        .
        .
  100   FCP000=CMPLX(B(K),B(K+10))
        A(K)=REAL(FCP000)
        A(K+10)=AIMAG(FCP000)
```

In the converted coding above, the user will want the last two statements to be included in the range of the DO loop. One solution to the problem is to move the statement number 100 to the last of the generated statements. Note, however, that this action necessitates modification of two transfer numbers in the IF statement within the DO loop.

When the FORTRAN LCP converts a complex arithmetic statement, it generates a warning message whenever the statement has a statement number.

COMPLEX VARIABLES IN INPUT/OUTPUT LISTS: References to complex variables in input/output lists are left unchanged. The original references are valid because of the manner in which the FORTRAN LCP treats complex variables and arrays.

Original:

```
I       DIMENSION D(3,4),H(10)
I   10  E = F * G / H(J) + D(1,1)
        WRITE OUTPUT TAPE 6, 13, D(I,J),
                    E(2),G(1), (H(I),I=1,20)
        READ INPUT TAPE 5,20, (H(I),
                                I=1,20)
```

Converted:

```
      COMPLEX FCP000
      DIMENSION D(3,8),H(20),E(2),G(2),
                                    F(2)
   10 FCP000=CMPLX(F(1),F(2))*CMPL(G(1),
               G(2))/CMPLX(H(J),H(J+10))
                     +CMPLX(D(1,1),D(1,5))
      E(1)=REAL(FCP000)
      E(2)=AIMAG(FCP000)
      WRITE(6,13) D(1,J),E(2),G(1),(H(I),
                                    I=1,20)
      READ (5,20) (H(I),I=1,20)
```

COMPLEX FUNCTION NAMES: The FORTRAN LCP adds the letter C to the beginning of the name of a library or built-in function wherever the name appears in a complex statement. The letter F at the end of the name is removed.

These actions make the source program name the same as the name of the function on the library tape.

A message in the output listing indicates the statement has been converted.

Original:

```
I     A = COSF(B)
```

Converted:

```
      COMPLEX FCP000
      DIMENSION A(2),B(2)
      FCP000=CCOS(CMPLX(B(1),B(2)))
      A(1)=REAL(FCP000)
      A(2)=AIMAG(FCP000)
```

REFERENCES TO COMPLEX FUNCTIONS AND SUBPROGRAMS: The FORTRAN LCP takes the following actions in converting complex functions or subprograms and in converting statements that contain references to a complex function or subprogram.

Arithmetic Statement Function: The function name and insert variables that replace the dummy arguments used in the definition of the function are included in COMPLEX type statements. A variable that appears in the expression part of the definition but is not a dummy argument will result in incompatibilities. The converted definition statement is flagged to indicate that such variables are in the statement.

When a reference to a arithmetic statement function appears in a complex statement, the arguments in the reference are converted by using the function CMPLX to derive complex values from the two real parts of each argument.

Built-In and Library Functions: Complex arguments must be transmitted to the actual functions that exist in the library. The statements are converted so that complex arguments are provided.

Subprograms: Because subprograms are also converted by the FORTRAN LCP, variables and array names that appear as arguments are converted to real variables and arrays. However, the FORTRAN LCP is unable to convert a complex expression used as an argument in the calling program because the converted subprogram expects to receive real arguments. The statement containing the complex expression as an argument is flagged to indicate the need for hand conversion.

Consider the following conversion in which ALIBF is a library or built-in function, FNCSUB is a function subprogram, ARITHF is an arithmetic statement function, and SUBRTN is a subroutine subprogram:

Original:

```
I     DIMENSION A(10),P(5)
I     ARITHF(C,D) = C + D + X
I     Y = ALIBF(B) + FNCSUB(E) + ARITHF
                                   (H,G)
I  20 CALL SUBRTN (P,A(I)+(3.0,2.1),Y)
```

Converted:

```
      COMPLEX ARITH,FCP001,FCP101,FCP102
      DIMENSION A(20),P(10),X(2),Y(2),
                B(2),E(2),H(2),G(2)
      ARITH (FCP101,FCP102)=FCP101
                           +FCP102+X
      FCP001=CALIB(CMPLX(B(1),B(2))
                  +FNCSUB(E)+ARITH
                 (CMPLX(H(1),H(2)),
                  CMPLX(G(1),G(2)))
      Y(1)=REAL(FCP001)
      Y(2)=AIMAG(FCP001)
   20 CALL SUBRTN (P,A(I)+(3.0,2.1),Y)
```

A message in the output listing indicates that the variable X in the arithmetic statement function could not be converted and must be changed by hand. A message will also be issued to indicate that the second argument in statement 20 (the argument A(I)+(3.0,2.1)) could not be converted.

The following example illustrates the conversion action within a subprogram, using the subroutine name SUBRTN from the example above:

Original:

```
      SUBROUTINE SUBRTN (R,S,T)
I     DIMENSION R(5), S(10)
      .
      .
      .
I     T = SINF(R(K))
      .
      .
      .
I     RETURN
      END
```

Converted:

```
      SUBROUTINE SUBRTN(R,S,T)
      DIMENSION R(10),S(20),T(2)
      COMPLEX FCP000
      .
      .
      .
      FCP000=CSIN(CMPLX(R(K),R(K+5)))
      T(1)=REAL(FCP000)
      T(2)=AIMAG(FCP000)
      .
      .
      .
      RETURN
      END
```

The complex variables and arrays have been converted so they can be treated as real variables and arrays. The conversion of the subprogram arguments in the calling program is effective only when the arguments are arrays, unsubscripted variables, and other function or subprogram names.

The FORTRAN LCP issues a message whenever an argument in a calling program cannot be converted

COMPLEX FUNCTION SUBPROGRAMS: In converting a complex function subprogram, the FORTRAN LCP specifies the type of the function as part of the FUNCTION statement. A message in the output listing indicates the FUNCTION statement has been converted.

Original:

```
      FUNCTION SUB2(R,S)
      .
      .
      .
I     SUB2=R+S
      .
      .
      .
I     RETURN
      END
```

Converted:

```
      COMPLEX FUNCTION SUB2(R,S)
      DIMENSION R(2),S(2)
      .
      .
      .
      SUB2=CMPLX(R(1),R(2))+CMPLX(S(1),
                                  S(2))
      .
      .
      .
      RETURN
      END
```

INCOMPATIBILITIES THAT ARE NOT RECOGNIZED

This list summarizes the FORTRAN II items and statements that are incompatible with System/360 FORTRAN and are not recognized by the FORTRAN LCP. All items in this list are also discussed in the conversion action lists. The user must check his converted program for occurrence of these items and make hand changes where necessary. The current FORTRAN compilers to which an item or statement refers are listed beside the title of the item or statement.

HIERARCHY OF ARITHMETIC OPERATIONS: (All current FORTRAN II compilers) If the user has failed to include parentheses to control the order of computations in arithmetic expressions, the FORTRAN LCP neither provides parentheses nor generates a message. If the order of computations is important, the user must check his program and insert the parentheses where necessary.

DO STATEMENT: (All current FORTRAN II compilers) If a source program contains a transfer into a DO loop under any condition except that cited under the heading "DO Statement" in the section "Conversion Actions for Current FORTRAN II Compilers," the user must change his program by hand.

INPUT AND OUTPUT OF ARRAYS: (All current FORTRAN II compilers except 1620 GOTRAN and 1620 FORTRAN With FORMAT) The FORTRAN LCP changes the sequence of specification statements in a source program so that all specification statements (except FORMAT statements) appear before the first executable statement of the program. Because this reordering may affect input/output references to arrays, the user must check the converted output to determine whether any hand coding changes are required.

CARRIAGE CONTROL CHARACTERS: (All current FORTRAN II compilers) The user must check his program to ensure that the characters 2 through 9 or J through R are not used as carriage control characters.

ORDERING OF ELEMENTS IN ARRAYS: (1410, 705/7080, 7090/7094) If the user employs non-FORTRAN statements to manipulate individual elements in an array, the locations referred to in the non-FORTRAN statements may be incorrect in System/360 FORTRAN. Care should be taken to refer to the elements of an array according to the locations in which they are stored by the System/360 FORTRAN compiler.

READ (0100) AND WRITE (0500) STATEMENTS: (7080) The FORTRAN LCP does not recognize use of the values 0100 or 0500 in READ or WRITE statements, respectively, to designate the card reader or console typewriter in a 7080 FORTRAN program. The replace tape reference option can be used to correct this incompatibility.

COMMON STATEMENTS CONTAINING DOUBLE-PRECISION VARIABLES: (7090/7094) Two problems are associated with COMMON statements that contain double-precision variables, as follows:

1. Boundary Problem: System/360 FORTRAN requires that a double-precision variable begin on a double-word boundary. Neither converted COMMON statements nor COMMON statements generated in response to the reorder COMMON option are checked by the LCP to determine that each double-precision variable starts on a double-word boundary. The user should check COMMON statements in the converted program and make hand changes where necessary.

2. Implied Equivalencies: In some instances, COMMON statements in a calling program and a subprogram are specified so that implied equivalencies exist between variables in the two programs. In making hand changes to meet System/360 boundary requirements, the user should take care to preserve these implied equivalencies.

CONVERSION OUTPUT AND MESSAGES

This section describes the output generated by the FORTRAN LCP and explains the messages created during conversion.

CONVERSION OUTPUT

The FORTRAN LCP can produce two forms of output: punched output and printed output.

Punched output is optional. When punched output is specified, the FORTRAN LCP produces a punched deck that contains the converted FORTRAN program. The user can also specify that non-FORTRAN statements are to be punched into cards.

A listing of the converted program is always provided. This listing contains the converted program with messages generated during conversion, and also contains tables showing changes in variable and function names. In addition, the user can specify that a listing of the original source statements is to be included in the listing output.

Both forms of output are discussed in the following text.

PUNCHED CARD OUTPUT

The punched card deck produced by the FORTRAN LCP contains converted statements and source statements that did not require conversion. In all cases, extraneous blanks have been removed from the statements. The form of the coding in the punched cards matches the coding in the listing of the converted program.

Cards containing incompatibilities should be corrected before the new source deck is submitted for initial System/360 FORTRAN compilation.

If the user has specified that non-FORTRAN statements are to be punched, cards containing these statements are separated from cards containing converted FORTRAN statements. The manner in which the cards are separated differs according to the device to which the punched output is sent.

When punched output is sent directly to the 1402 card punch, cards containing non-FORTRAN statements drop into a different punched-card pocket from the one that contains converted FORTRAN statements. When punched output is recorded on tape and later punched, the non-FORTRAN cards are punched before non-FORTRAN cards, but all cards fall into the same punched-card pocket.

LISTING OUTPUT

The listing is divided into two major sections, as follows:

- Source program listing (which is optional)
- Converted program listing (which is always provided)

The converted program listing is subdivided into four parts, as follows:

- Listing of the specification statements in the converted program.
- Function name table showing the System/360 function names and LCP substitution names that have replaced source program function names.
- Insert variable table showing the LCP insert variables that have replaced variable names in the source program.
- Listing of the remainder of the converted program with messages.

Each part of the output listing is described more fully in the following paragraphs.

Source Program Listing

This listing contains the original source statements exactly as they appeared in the input cards. The statements are listed consecutively.

This portion of the listing is optional. The user specifies in a control card whether or not the source program is to be listed.

Function Name Table

This table contains each function name in the source program that has either (1)

been replaced with an LCP substitution name, or (2) been changed to an equivalent System/360 function name. (However, FORTRAN II function names that have been converted merely by removing the terminal F are not included in the table.) The changed name or the LCP substitution name appears beside the function name it has replaced.

This table does not appear in the output listing if there are no function name changes.

The conditions that cause a function name to be changed or replaced are described in the section "General Problems in Converting to System/360 FORTRAN."

Appendix B contains a listing of System/360 function names that are equivalent to FORTRAN II function names.

Insert Variable Table

This table shows each source program variable name that has been replaced and the LCP insert variable that has replaced it. The same insert variable is used to replace the same variable name wherever that name appears in a source program.

This table does not appear in the output listing if no variable name has been replaced by an insert variable.

The conditions that cause insert variables to be placed in a program are described in the section "General Problems in Converting to System/360 FORTRAN."

Converted Program Listing

This listing contains the converted source program and the messages generated during the conversion.

The converted program contains converted statements, statements generated by the LCP, incompatible statements that could not be converted, and source statements that did not require conversion. Extraneous blanks have been removed from statements in the converted program.

In the converted program listing, all specification statements (except FORMAT statements) are printed in a section that precedes the function name table and the insert variable table. The remainder of the converted program, starting with the first executable statement, is printed after the two tables.

The listing of the converted program coding is divided generally into two parts. The left portion of the listing contains the converted FORTRAN statements. The right portion of the listing contains text messages that indicate whether the statement has been modified by the FORTRAN LCP, generated by the FORTRAN LCP, or is incompatible with System/360 FORTRAN. Also provided are four-digit message codes that aid the user in determining the exact nature of each conversion action or incompatibility. These message codes will be explained in detail in the reference manual provided when the program is completed.

MESSAGES

The messages in the converted program listing provide the user with two methods of identifying actions taken by the FORTRAN LCP.

The text messages facilitate quick visual identification of the status of each statement in the converted program. This enables the user to scan the program to find individual statements or groups of statements that require hand changes.

A text message appears with each output statement that fits into one of the following categories:

- The statement was modified by the FORTRAN LCP.
- The statement was generated by the FORTRAN LCP.
- The statement, or a portion of the statement, is definitely invalid in System/360 FORTRAN but could not be converted by the FORTRAN LCP.
- The statement, or a portion of the statement, may be invalid in System/360 FORTRAN.

In addition to text messages, the FORTRAN LCP produces four-digit message codes to assist the user in determining the exact nature of each conversion action and each unresolved conversion problem. A distinct message code will be related to each conversion action or unresolved problem. The meaning of each code will be explained in a later version of this reference manual.

Using the output listing, the FORTRAN LCP manual, and the proper System/360 FORTRAN manual, the user can determine the hand changes that are required to make his program suitable for compilation by a System/360 FORTRAN compiler.

CONTROL CARDS AND OPERATING PROCEDURES

This section contains information on control specifications and the various machine configurations and operating options available to the user.

CONTROL SPECIFICATIONS

Certain control information is required for the FORTRAN LCP to accomplish a conversion. This control information can be provided in either of two ways.

The information can be preset in the communications areas of the LCP Control Phase or the information can be provided in control cards.

PRESET CONTROL INFORMATION

In the distributed LCP tape, control entries have already been preset to a standard pattern.

The user might not need to change the pattern. However, to modify an entry, the user punches specified characters into cards within the LCP Control Phase. In this way, the user replaces a standard option with his own preset option.

Whenever the LCP Control Phase is loaded at the beginning of a conversion, control indicators are set to reflect the preset options.

To override a preset option the user must include a control card for that option in the control card deck for the source program. The control card overrides a preset option only for the program for which it is used.

By presetting control options, the user can reduce the number of control cards required for each conversion. This procedure is useful whenever the same options are desired for many consecutive programs.

USE OF CONTROL CARDS

When all control options have been preset, the only control card required with a source program is a specific terminal control card.

However, if the user wants to change options for a particular conversion, additional control cards must be included in the control card deck.

The terminal control card or a set of control cards must be provided for each source program processed by the FORTRAN LCP.

When more than one control card is used, the cards related to one source program (a main FORTRAN program or a subprogram) must appear in a single control card deck. The individual cards may be in any order within this deck except that the last card must be the terminal card.

The position of this deck within the card input stream depends upon the data processing equipment being used. Positioning of control cards is discussed later in this section.

CONTROL CARD ENTRIES

To override preset options for one conversion, the user must provide one or more of the following entries in control cards used with a source program:

1. Base Computer: This entry indicates the current IBM computer for which the source program was written.
2. Target Level of System/360 FORTRAN: This entry indicates the level of System/360 FORTRAN to which the source program is to be converted. The user can specify Basic Support, Level E, or Level H FORTRAN.
3. Input Unit: This entry indicates the input unit on which the source program is located. The user can specify the card reader, an LCP work tape (single program only), or an additional tape unit.
4. Input Card Code: This entry indicates the card code that is present in the source program. The user specifies either BCDIC input or EBCDIC input.
5. Output Card Code: This entry indicates the card code to be used in punching an output deck. The user specifies either BCDIC output or EBCDIC output.
6. Listing Output Specifications: Two entries are specified for listing output, as follows:
   a. Source Program Entry. This entry

indicates whether or not a listing of the original source program is to be included in listing output.

b. Listing Output Unit Entry. This entry indicates whether listing output is to be sent to the on-line 1403 printer or to magnetic tape.

7. Punched Output Specification: An entry is required to indicate whether or not the FORTRAN LCP is to provide punched output. Two other entries are specified, as follows:
   a. Punched Output Content Entry. This entry indicates whether FORTRAN statements only are to be punched, or whether converted FORTRAN statements and non-FORTRAN statements are to be punched. If the user designates that non-FORTRAN statements are to be punched, these cards are separated from converted FORTRAN cards.
   b. Punched Output Unit Entry. This entry indicates whether punched output is to be sent to the on-line 1402 card punch or to magnetic tape.
8. Reorder COMMON Specification: This entry indicates whether or not the variables in COMMON statements are to be reordered. The options are:
   a. Variables are to be left in the same order in which they appear in the source program COMMON statements.
   b. Variables are to be reordered so that the System/360 FORTRAN compiler will allocate COMMON storage in the same manner in which it was allocated by the FORTRAN II compiler. (See the discussion of the COMMON-EQUIVALENCE interaction problem in the section "General Problems in Converting to System/360 FORTRAN.")
9. Date: This entry can be used to provide the date on which the conversion is being performed. If provided, the date appears in the output listing.
10. Integer Variable Change Entry: This entry can be used to change the three alphabetic characters that appear at the beginning of all integer insert variables generated by the FORTRAN LCP. The form of the integer insert variable is normally LCPxxx where the xxx represents a three-digit number. If the user desires alphabetic characters other than LCP, the desired alphabetic characters are entered in a control card.
11. Real Variable Change Entry: This entry can be used to change the three alphabetic characters that appear at the beginning of all real insert variables generated by the FORTRAN LCP. The form of the real insert variable is normally FCPxxx where the xxx represents a three-digit number. If the user desires alphabetic characters other than FCP, the desired alphabetic characters are entered in a control card.
12. Replace Tape Reference Entry: This entry can be used to specify that constants used as tape references in input/output statements are to be modified. The user can specify that a particular tape constant is to be replaced with a variable name or another tape constant wherever that tape constant appears in the source program. If a tape constant is to be replaced by a variable name, the name must be six or fewer characters and must begin with one of the letters I through N. For each tape constant to be changed, the user is required to provide the constant as it appears in the source program and followed by the variable or constant that is to replace it. The LCP allows references to as many as five different tape units to be changed wherever they appear in input/output statements in the program.

SYSTEM CREATION

The FORTRAN LCP is delivered to the customer on a reel of magnetic tape. This tape is used to create an LCP system either in the form of a system card deck or in the form of a system tape.

The term LCP system is used in this manual to refer to the control and processing phases of the FORTRAN LCP. The term system card deck refers to a deck of cards that contain these phases. The term system tape refers to a tape that contains the phases.

The manner in which the user creates the system card deck and the system tape is discussed in the following paragraphs.

CREATING A SYSTEM CARD DECK

The delivered FORTRAN LCP tape is converted to a system card deck by using a utility program to punch the contents of the delivered tape.

The first program on the delivered tape is the Tape Create Program. Consequently, this program is the first to be punched out. The user must remove the Tape Create Program cards from the front of the punched

deck. The remainder of the deck is then ready for use as a system card deck.

Preset options can be punched into Control Phase cards if the deck is used.

CREATING A SYSTEM TAPE

The Tape Create Program enables the user to automatically convert the system card deck into a system tape.

The procedure for creating a system tape is as follows:

1. Punch the contents of the delivered FORTRAN LCP tape as described above.
2. At this point, the user can punch control options into Control Phase cards.
3. Place the entire punched deck into the read hopper of the 1402 Card-Read-Punch. (Do not remove the Tape Create Program from the front of the punch deck.)
4. Mount a work tape on an available tape unit.

The pressing of the Load key initiates loading and execution of the Tape Create Program. When execution is completed, the LCP system has been created on the work tape.

PROCESSING CONFIGURATIONS

The FORTRAN LCP is designed to operate on a minimum machine configuration consisting of an 8,000-character 1401 with a card-reader, a printer, three tape units, the Advanced Programming Feature, and the High-Low-Equal Compare Feature (or a System/360 with the 1401 Compatibility Feature and equivalent input/output devices and features.)

The three tape units in the minimum configuration are LCP work tapes. One of these units can be used for an input function. If the user desires, he can convert a single program mounted on one of the work tape units. When the source program has been read, the tape is rewound and unloaded.

Greater efficiency can be achieved by using more than three tape units. Each additional tape unit either speeds processing or provides the user with more options. The 1401 system permits as many as six tape units to be used. If available, the fourth, fifth, and sixth tape units can be used for the LCP system input, source program input, or punched and/or print output, depending on the user's preferences (see Figure 5).

CARD-ORIENTED SYSTEM

If the LCP phases are on cards, the system is called a card-oriented system. The source programs may be either on cards or on tape. Note that the LCP control cards must always be introduced through the card reader.

If a source program is on cards, the source program is inserted into the LCP system card deck. Figure 6 shows the sequence in which portions of the combined deck are placed in the 1402 read hopper.

The deck begins with the LCP Control Phase. This phase is followed by the control card deck for this conversion. The control cards are followed by the first LCP processing phase and then by the source program itself. The rest of the deck consists of the other LCP processing phases in sequence.

When the minimum three tape units are available, only a single source program can be processed from tape. This tape is mounted on an LCP work tape unit. The control card deck and the system card deck are placed in the card reader. The sequence of the deck is: (1) LCP Control Phase, (2) control card deck, and (3) LCP processing phases in sequence. An entry in a control card indicates that the source program is on the work tape.

When a fourth tape unit is available, the user can process single or stacked programs from tape. However, the operator must reload the system card deck and a control card deck for each program to be converted. When processing stacked programs, the LCP issues a message after the last program has been converted.

TAPE-ORIENTED SYSTEM

When the LCP phases are on tape, the system is called a tape-oriented system. The source programs may be either on cards or on tape.

In a tape-oriented system, single or stacked programs can be processed through the card reader. For stacked programs on cards, the format of the input deck is as shown in Figure 7. A control card deck is

required for each source program, and the source program immediately follows its control card deck.

The minimum tape-oriented system (four tape units) also allows a single source program to be processed from an LCP work tape.

When a fifth tape unit is available, stacked source programs can be processed from the additional unit. When the LCP phases and the source programs are on tape, the only input from the card reader is the control card decks in the sequence in which the programs are to be converted.

In processing stacked source programs on tape, transition from one job to the next is accomplished by the LCP. When conversion of one program is completed, the LCP Control Phase is automatically reloaded and processing of the next program begins.

A special entry in the terminal control card for the last source program signals the FORTRAN LCP that it is processing the last in a series of stacked source programs.

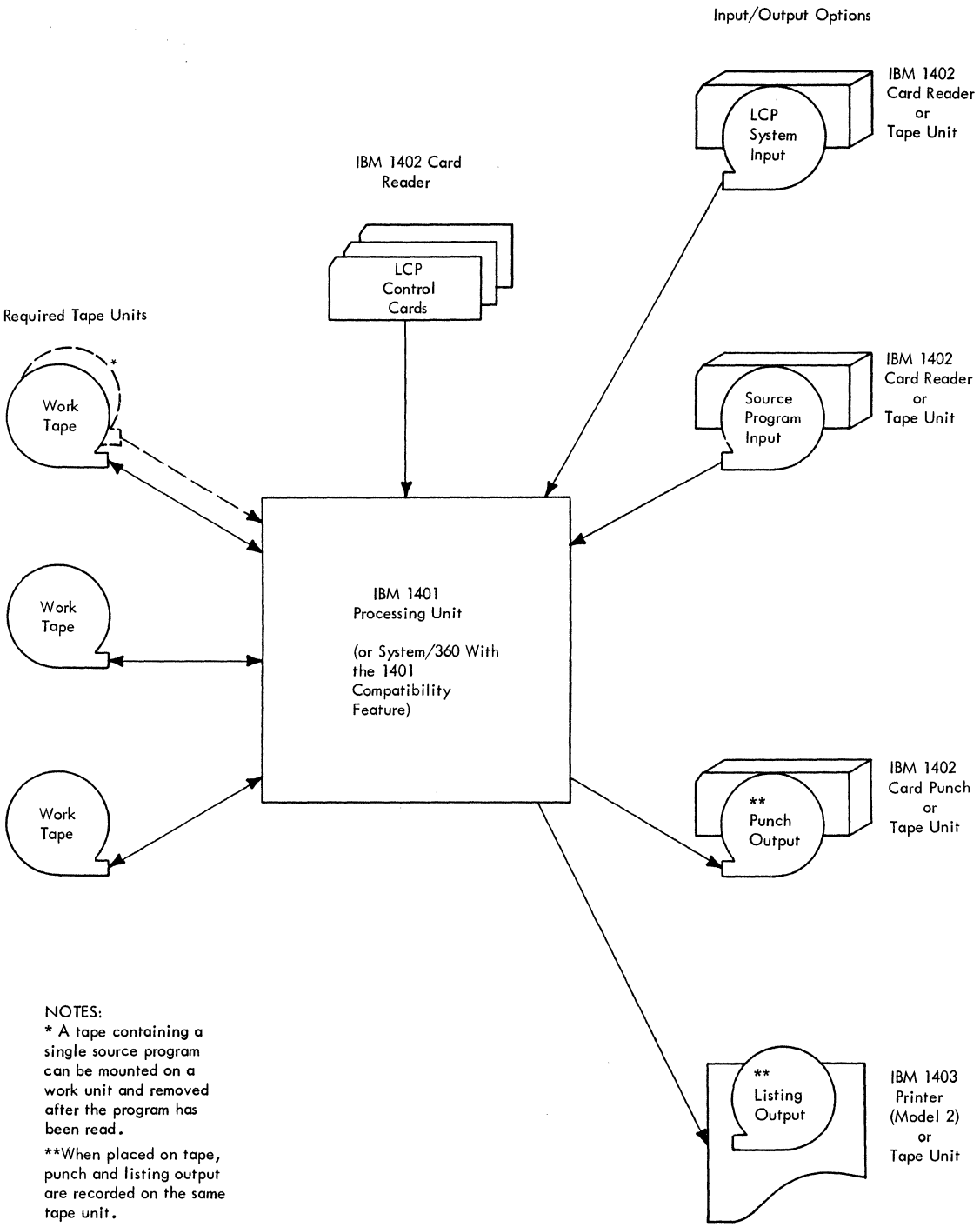

Figure 5. Input/Output Configuration Options for the FORTRAN LCP

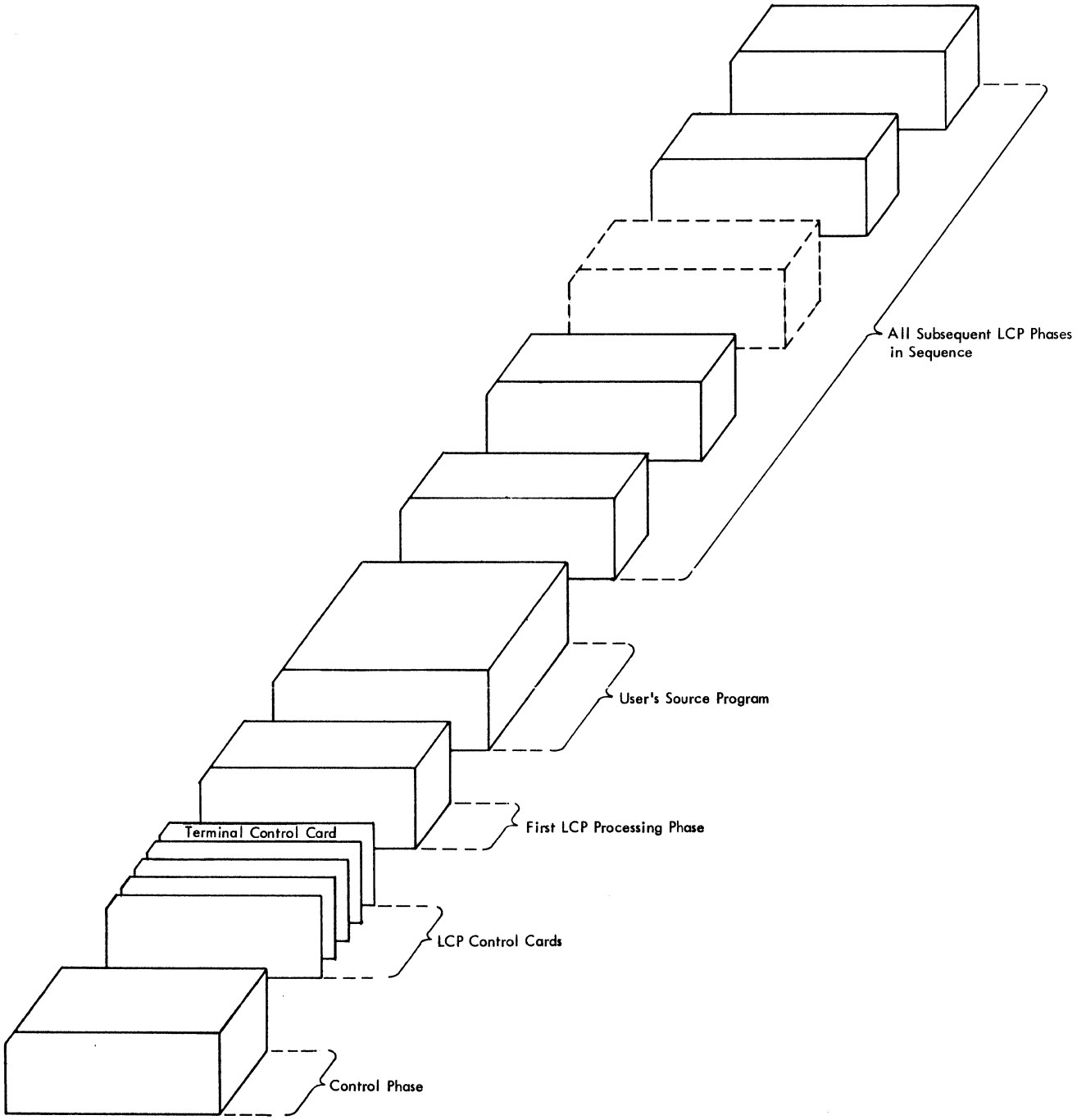

Figure 6. Input Deck When Both the FORTRAN LCP and Source Program Are on Cards

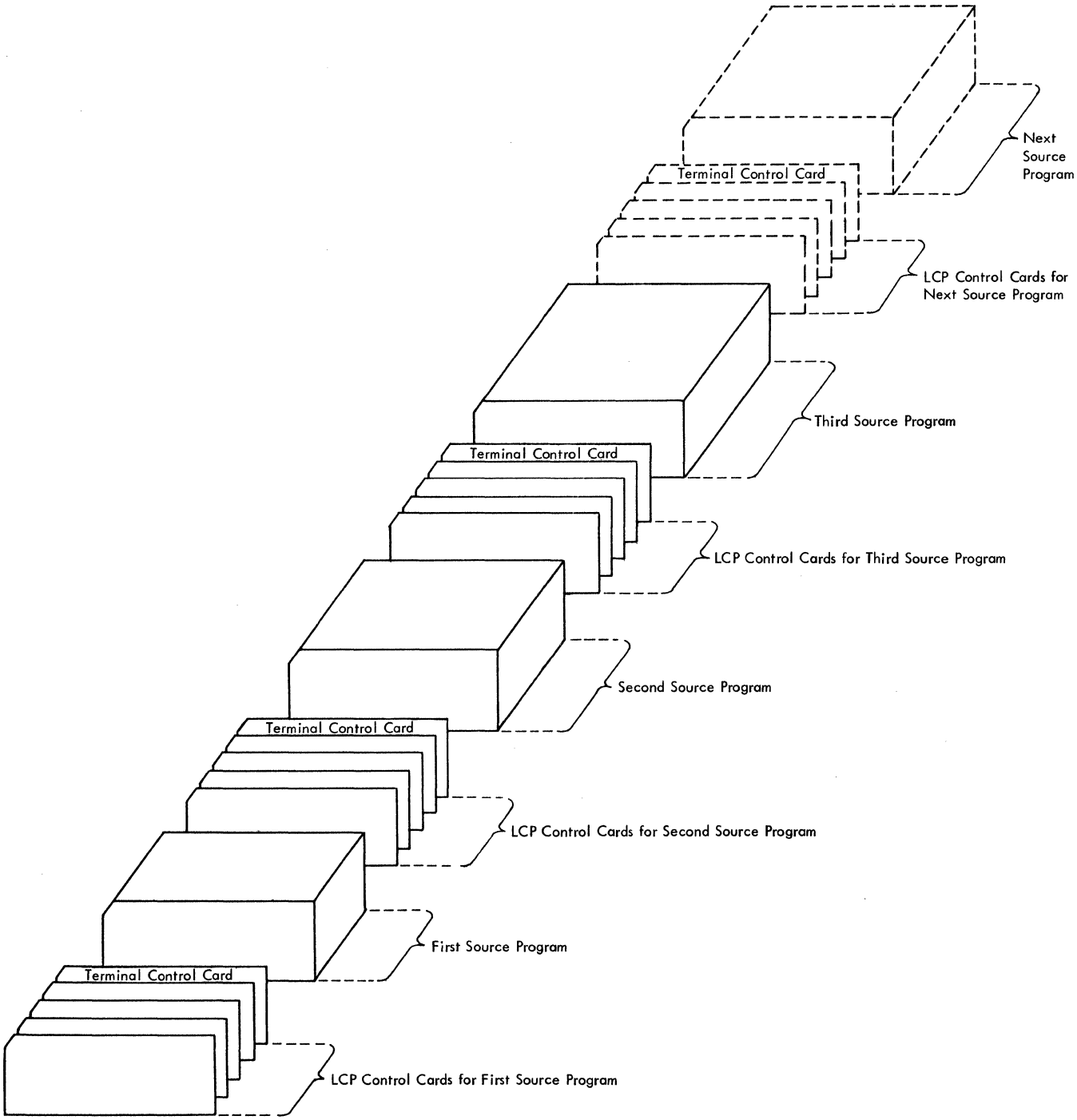

Figure 7. Input Deck When the FORTRAN LCP Is on Tape and Source Programs Are Read from the Card Reader

This appendix illustrates LCP conversion of a FORTRAN program. Figure 8 is a listing of the program as it was written for a FORTRAN II compiler. Figure 9 is a listing of the same program after conversion by the FORTRAN LCP.

The program reads ten real numbers from tape, sorts the numbers into ascending sequence, takes the square root of each number, multiplies it by pi, and truncates the result to an integer. Finally, the program writes the results onto tape. If any product causes an accumulator overflow, the program prints out the corresponding input number with a message and halts.

Conversion actions and the reasons for them are explained fully under "Conversion Action Lists" in the body of this manual. Besides condensing the coding, the LCP has:

1. Placed specification statements before the first executable statement in the program.
2. Converted the READ and WRITE statements into acceptable System/360 format.
3. Created insert variables where required.
4. Supplied a 0 for the exponent of the real constant PI.
5. Replaced the function names SQRTF and XINTF with the proper System/360 names (see Appendix B).
6. Converted the accumulator overflow test into a CALL to a System/360 FORTRAN subroutine.

```
      COMMON ARRAY, IFIX                                                INPUT
      DIMENSION ARRAY (10), IFIX (10)                                   INPUT
      READ INPUT TAPE 2, 100, (ARRAY (I), I = 1, 10)                    INPUT
  100 FORMAT (10F12.4)                                                  INPUT
      DO 200 INDEX = 1, 10                                              INPUT
      NCOMP = 10-INDEX                                                  INPUT
      DO 200 ISUB = 1, NCOMP                                            INPUT
      IF (ARRAY (ISUB) - ARRAY (ISUB + 1)) 200, 200, 150                INPUT
  150 TEMP= ARRAY (ISUB + 1)                                            INPUT
      ARRAY (ISUB + 1) = ARRAY (ISUB)                                   INPUT
      ARRAY (ISUB) = TEMP                                               INPUT
  200 CONTINUE                                                          INPUT
      PI = 3.1415927E                                                   INPUT
      DO 300 INDEX = 1, 10                                              INPUT
      ARRAY (INDEX) = SQRTF (ARRAY (INDEX)) * PI                        INPUT
      IF ACCUMULATOR OVERFLOW 400, 300                                  INPUT
  300 IFIX (INDEX) = XINTF (ARRAY (INDEX))                              INPUT
      GO TO 900                                                         INPUT
  400 PRINT 500, ARRAY (INDEX)                                          INPUT
  500 FORMAT (1H0, F12.4, 20HACCUMULATOR OVERFLOW)                      INPUT
      STOP                                                              INPUT
  900 WRITE OUTPUT TAPE 6, 1000, (IFIX (INDEX), INDEX = 1, 10)          INPUT
 1000 FORMAT (10I6)                                                     INPUT
      END                                                               INPUT
```

Figure 8. FORTRAN II Source Program

```
      DIMENSION ARRAY(10),LCP001(10)                                    OUTPUT
      COMMON ARRAY,LCP001                                               OUTPUT
      READ (2,100) (ARRAY(I),I=1,10)                                    OUTPUT
  100 FORMAT (10F12.4)                                                  OUTPUT
      DO 200 INDEX=1,10                                                 OUTPUT
      NCOMP=10-INDEX                                                    OUTPUT
      DO 200 ISUB=1,NCOMP                                               OUTPUT
      IF (ARRAY(ISUB)-ARRAY(ISUB+1)) 200,200,150                        OUTPUT
  150 TEMP=ARRAY(ISUB+1)                                                OUTPUT
      ARRAY(ISUB+1)=ARRAY(ISUB)                                         OUTPUT
      ARRAY(ISUB)=TEMP                                                  OUTPUT
  200 CONTINUE                                                          OUTPUT
      PI=3.1415927E0                                                    OUTPUT
      DO 300 INDEX=1,10                                                 OUTPUT
      ARRAY(INDEX)=SQRT(ARRAY(INDEX))*PI                                OUTPUT
      CALL OVERFL(LCP002)                                               OUTPUT
      GO TO (400,300,300),LCP002                                        OUTPUT
  300 LCP001(INDEX)=INT(ARRAY(INDEX))                                   OUTPUT
      GO TO 900                                                         OUTPUT
  400 PRINT 500,ARRAY(INDEX)                                            OUTPUT
  500 FORMAT (1H0,F12.4,20HACCUMULATOR OVERFLOW)                        OUTPUT
      STOP                                                              OUTPUT
  900 WRITE (6,1000) (LCP001(INDEX),INDEX=1,10)                         OUTPUT
 1000 FORMAT (10I6)                                                     OUTPUT
      END                                                               OUTPUT
```

Figure 9. Converted Source Program

When the FORTRAN LCP encounters the name of an IBM-provided built-in or library function, it converts the FORTRAN II name to the appropriate System/360 FORTRAN name.

Table 3 shows the function names used in the various FORTRAN II compilers and the equivalent function names in System/360 FORTRAN.

To use the table, the reader should scan down a column until he finds the name of the FORTRAN II function in which he is interested. The reader then scans across to find the equivalent System/360 name. Except for certain 705 and 7080 FORTRAN function names, the FORTRAN II names are in alphabetical order.

A blank box on the FORTRAN II side of the table indicates that the function was not provided in that FORTRAN II compiler.

A blank box on the System/360 side of the table indicates that no equivalent function is provided in System/360 FORTRAN. If there is no System/360 equivalent for a FORTRAN II function, the FORTRAN LCP issues a warning message in the output listing.

Certain functions provided in System/360 Level H FORTRAN are not provided in Level E or Basic Support FORTRAN. Asterisk notation is used in the table to indicate those function that are not available in either or both of the lower level System/360 FORTRAN compilers.

Note 1: The 7080 FORTRAN compiler accepts either the FORTRAN II names listed in the table or FORTRAN IV equivalents. Except for two cases noted in the table, the FORTRAN IV function name is the same as the System/360 function name. The 7080 FORTRAN IV names are not included in the table. When the 7080 FORTRAN IV name is the same as a System/360 name, no name change is required.

Note 2: The name of each of the six functions provided in 1620 GOTRAN is included in the table in the same box as the equivalent 1620 FORTRAN function name. Each GOTRAN function name is converted to the equivalent System/360 single-precision function name.

Table 3. System/360 Names for FORTRAN II Functions

FORTRAN II Name							System/360 Name		
1401	1620	1410	7070/7074	705	7080	7090/7094	Single-Precision	Double-Precision	Complex
ABSF	ABSF	ABSF	ABSF	ABSF	ABSF	ABSF	ABS	DABS*	CABS**
			ASINF						
ATANF	ATN tt ATANF	ATANF	ATANF	ATANF	ATANF	ATANF	ATAN	DATAN*	
						ATAN2F		DATAN2*	
COSF	COS tt COSF	COSF	COSF	COSF	COSF	COSF	COS	DCOS*	CCOS**
		DIMF	DIMF	DIMF	DIMF	DIMF	DIM		
EXPF	EXP tt EXPF	EXPF	EXPF	EXPF	EXPF	EXPF	EXP	DEXP*	CEXP**
			EXPXF	EXPBF	EXPBF EXPB t				
FLOATF		FLOATF	FLOATF	FLOATF	FLOATF	FLOATF	FLOAT	DFLOAT*	
	INTF	INTF	INTF	INTF	INTF	INTF	AINT		
LOGF	LOG tt LOGF	LOGF	LOGF	LOGF	LOGF	LOGF	ALOG	DLOG*	CLOG**
			LOGXF	LOGBF	LOGBF ALOGB t		ALOG10	DLOG10*	CLOG10**
		MAX0F	MAX0F	MAX0F	MAX0F	MAX0F	AMAX0		
		MAX1F	MAX1F	MAX1F	MAX1F	MAX1F	AMAX1		
		MIN0F	MIN0F	MIN0F	MIN0F	MIN0F	AMIN0		
		MIN1F	MIN1F	MIN1F	MIN1F	MIN1F	AMIN1		
		MODF	MODF	MODF	MODF	MODF	AMOD	DMOD*	
		SIGNF	SIGNF	SIGNF	SIGNF	SIGNF	SIGN	DSIGN*	
SINF	SIN tt SINF	SINF	SINF	SINF	SINF	SINF	SIN	DSIN*	CSIN**
SQRTF	SQR tt SQRTF	SQRTF	SQRTF	SQRTF	SQRTF	SQRTF	SQRT	DSQRT*	CSQRT**
			TANHF			TANHF	TANH		
XABSF		XABSF	XABSF	XABSF	XABSF	XABSF	IABS		
		XDIMF	XDIMF	XDIMF	XDIMF	XDIMF	IDIM		
XFIXF		XFIXF	XFIXF	XFIXF	XFIXF	XFIXF	IFIX		
		XINTF	XINTF	XINTF	XINTF	XINTF	INT		
		XMAX0F	XMAX0F	XMAX0F	XMAX0F	XMAX0F	MAX0		
		XMAX1F	XMAX1F	XMAX1F	XMAX1F	XMAX1F	MAX1	DMAX1*	
		XMIN0F	XMIN0F	XMIN0F	XMIN0F	XMIN0F	MIN0		
		XMIN1F	XMIN1F	XMIN1F	XMIN1F	XMIN1F	MIN1	DMIN1*	
		XMODF	XMODF	XMODF	XMODF	XMODF	MOD		
		XSIGNF	XSIGNF	XSIGNF	XSINGF	XSIGNF	ISIGN		

Notes:
* Function not available in System/360 Basic Support FORTRAN.
** Function not available in System/360 Basic Support or Level E FORTRAN.
t This is the 7080 FORTRAN IV name for the function. ALOGB is converted to ALOG10.
tt This is a 1620 GOTRAN function name. It is converted to the equivalent System/360 single-precision function name.

Where more than one page reference is provided, the first page number indicates the major reference.

To differentiate between entries for some input/output statements, a number in parentheses is provided to indicate the compiler(s) to which the entry applies. For example, (1620) indicates that the statement is related to one or more 1620 compilers.

READER'S COMMENTS

IBM System/360 Transition Aids: FORTRAN II Language Conversion Program
For the IBM 1401; Preliminary Specifications
C28-6560-0

Your comments will help us to produce better publications for your use. Please check or fill in the items below and add explanations and other comments in the space provided.

Name: _____

Address: _____

Which of the following terms best describes your job?

¤ Programmer	¤ Systems Analyst	¤ Customer Engineer
¤ Manager	¤ Engineer	¤ Systems Engineer
¤ Operator	¤ Mathematician	¤ Sales Representative
¤ Instructor	¤ Student/Trainee	¤ Other (explain) _____

Does your installation subscribe to the SRL Revision Service? ¤ Yes ¤ No

How did you use this publication?

¤ As an introduction
¤ As a reference manual
¤ As a text (student)
¤ As a text (instructor)
¤ For another purpose (explain) _____

Did you find the material easy to read and understand? ¤ Yes ¤ No (explain below)

Did you find the material organized for convenient use? ¤ Yes ¤ No (explain below)

Specific Criticisms (explain below)

Clarifications on pages
Additions on pages
Deletions on pages
Errors on pages

Explanations and Other Comments

No postage necessary if mailed in U.S.A.

FOLD FOLD

FIRST CLASS
PERMIT NO. 116
KINGSTON, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY
IBM CORPORATION
NEIGHBORHOOD ROAD
KINGSTON, N. Y. 12401

ATTN: PROGRAMMING SYSTEMS PUBLICATIONS
DEPARTMENT 637

FOLD FOLD

Printed in U.S.A. C28-6560-0 465/08 P

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601