File No. S360-29
Form C33-0002-0

OS

IBM Systems Reference Library

IBM System/360 Operating System

PL/I (F) Version 5

Planning Guide

This publication is a planning aid only. It is intended for use prior to the availability of the fifth version of the PL/I (F) Compiler and is to be replaced by reference documentation when that compiler becomes available.

Used in conjunction with the publication IBM System/360 PL/I Reference Manual, Form C28-8201, this publication enables the user to write PL/I programs that are to be compiled using the fifth version of the F Compiler under the IBM System/360 Operating System.

PREFACE

This publication is intended for use as a planning aid by systems analysts and programmers who will be using the fifth version of the F Compiler when it becomes available. It is designed for the reader who already has a knowledge of the language and who requires an additional source of reference for planning purposes.

The book contains four sections related to the four major areas of improvement in the compiler: Section 1 describes functional additions (and therefore contains most of the language changes and additions implemented by the new version of the compiler); Section 2 describes improvements in performance and optimization; Section 3 details improvements in the use of storage; Section 4 describes other improvements in the usability of the compiler. Additional information, including compatibility with previous versions of the compiler, and system requirements, is given in Appendixes A and B; and Appendix C provides an alphabetical list of changed language features related to topics covered in the text.

REQUISITE PUBLICATIONS

To make full use of this publication, the reader must have the following:

IBM System/360 PL/I Reference Manual, Form C28-8201.

For additional information necessary to compile, link-edit, and execute a program, the reader should refer to the following publication:

IBM System/360 Operating System: PL/I (F) Programmer's Guide, Form C28-6594.

RECOMMENDED PUBLICATIONS

The following publications contain other information that will be valuable to users intending to take advantage of the teleprocessing support implemented by the fifth version of the compiler:

IBM System/360 Operating System: QTAM Message Processing Services, Form C30-2003

IBM System/360 Operating System: QTAM Message Control Program, Form C30-2005

First Edition (March 1969)

This edition applies to Version 5 of the PL/I (F) Compiler which operates under IBM System/360 Operating System until otherwise indicated in new editions or technical Newsletters. Changes are periodically made to the specifications herein; before using this publication in connection with the operating of IBM systems (or equipment), refer to the latest SRL Newsletter, Form N20-0360, for the editions that are applicable and current.

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.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM United Kingdom Laboratories Ltd., Programming Publications, Hursley Park, Winchester, Hampshire, England.

Requests for copies of IBM publications should be made to your IBM representative or the IBM branch office serving your locality.

FIGURES

TABLES

1. FUNCTIONAL ADDITIONS

Functional additions for the fifth version of the F Compiler consist of the following: teleprocessing support by means of TRANSIENT files; two new string built-in functions (TRANSLATE and VERIFY); two new optimization options, ORDER and REORDER; and the adoption of System/360 halfword binary facilities for fixed binary variables of precision less than 16.

TELEPROCESSING SUPPORT

This subsection describes the new teleprocessing features of PL/I (i.e., the TRANSIENT file attribute and the PENDING condition) together with the associated teleprocessing format options of the ENVIRONMENT attribute. A simple programming example is given at the end of the subsection to illustrate the interrelationship of the new features.

INTRODUCTION

The TRANSIENT attribute and an associated ON-condition, PENDING, have been introduced into the language to allow teleprocessing applications programs to be written in PL/I. The fifth version of the F Compiler provides a communicating link between PL/I message processing programs using these features and the QTAM (Queued Telecommunications Access Method) facilities of the operating system.

The user must provide a QTAM message control program (MCP) suitable for the particular installation; QTAM macros can be used for this purpose. The message control program handles messages originating from and destined for a number of remote terminals, each of which is identified by a terminal name carried with the message. These messages are transmitted to and from the PL/I message processing program via queues in main storage. (These queues are supported by corresponding intermediate queues in a disk data set. The PL/I program has access only to the main storage queues, by means of a single intermediate buffer for each file.) The exact message format depends on the MCP, but each message will carry the terminal name with it. A message may be a complete unit, or may consist of a number of records so that it can be split up for processing; the ENVIRONMENT attribute is used to inform the compiler of the message format. The PL/I programmer must have this message format information to enable him to write the message processing program.

The PL/I features are simply an extension of the existing record-oriented input/output facilities of the language. The "data set" associated with each TRANSIENT file is in fact an input or output message queue set up by the MCP. A READ statement for the file will take the next message (or the next record from the current message) from the associated queue, assign the data part to the variable named in the INTO option (or set a pointer to point to the data in the buffer), and save the terminal name by assigning it to the variable named in the KEYTO option. (The PENDING condition is raised if the input queue is empty when the READ statement is executed.) A WRITE or LOCATE statement will transmit the processed message or record to the output queue, using the element expression specified in the KEYFROM option to identify the destination terminal.

LANGUAGE ADDITIONS

The teleprocessing extension to the language comprises the TRANSIENT attribute, which is an alternative to DIRECT and SEQUENTIAL, and the PENDING condition. Some of the rules for input/output statements are altered to accommodate the TRANSIENT attribute. In addition, the fifth version of the compiler requires a teleprocessing format option to be specified in the ENVIRONMENT attribute.

The TRANSIENT Attribute

The TRANSIENT attribute indicates that the contents of the data set associated with the file are re-established each time the data set is accessed. In effect, this means that records can be continually added to the data set by one program during the execution of another program that continually removes records from the data set. Thus the data set can be considered to be a continuous queue through which the records pass in transit between the message control program and the message processing program.

The data set associated with a TRANSIENT file differs from those associated with DIRECT and SEQUENTIAL files in that its contents are dynamic; reading a record removes it from the data set. Such a data set can never be created or accessed by a DIRECT or SEQUENTIAL file.

The use of TRANSIENT files is almost totally dependent on the implementation; for this reason, a list of rules for the use of TRANSIENT with the F Compiler is given below the general format.

General format:

DIRECT|SEQUENTIAL|TRANSIENT

The following rules apply specifically to the use of TRANSIENT with the F Compiler:

1. The TRANSIENT attribute can be specified only for RECORD KEYED BUFFERED files with either the INPUT or the OUTPUT attribute.

2. The ENVIRONMENT attribute with one of the two teleprocessing format options (G and R) must be declared for TRANSIENT files.

3. Input can be specified only by a READ statement with the KEYTO option and either the INTO option or the SET option.

4. Output can be specified only by a WRITE statement or a LOCATE statement, either of which must have the KEYFROM option.

5. The EVENT option is not permitted, since TRANSIENT files are always BUFFERED.

6. The "data set" associated with a TRANSIENT file is in fact a queue of messages maintained automatically in main storage by a separate message control program using the QTAM (Queued Telecommunications Access Method) facilities of the operating system. The queue is always accessed sequentially.

7. The name or title of a TRANSIENT INPUT file must be the name of a recognized queue set up by the message control program. For TRANSIENT OUTPUT files, any name can be declared, since the file is re-associated for each output operation with a queue determined by the terminal name.

8. The element expression specified in the KEYFROM option must have as its value a recognized terminal identification.

The PENDING Condition

General Format: PENDING (file-name)

Description: Except when signaled, the PENDING condition can be raised only during execution of a READ statement for a TRANSIENT file. It is raised when an attempt is made to read a record that is temporarily unavailable (i.e., for the F Compiler, when the message queue associated with the file contains no messages at the time the READ statement is executed).

Standard System Action: In the absence of an on-unit, the action is as described for normal return.

Normal Return: Upon the normal completion of the on-unit for this condition, control returns to the point of interrupt (unless the condition was signaled), where execution is suspended until an appropriate record becomes available. If the condition was signaled, execution continues with the statement immediately following the SIGNAL statement that caused the interrupt.

Note: The value of the ONKEY built-in function when the PENDING condition is raised is a null string.

NEW ENVIRONMENT OPTIONS (G AND R)

Two new options of the ENVIRONMENT attribute are associated with the teleprocessing extension. These are:

G(maximum-message-size)

R(maximum-record-size)

One of these options must be specified for TRANSIENT files; they cannot be specified for DIRECT, SEQUENTIAL, or STREAM files; and they cannot appear in conjunction with any other option of the ENVIRONMENT attribute.

The maximum message size and maximum record size are specified by decimal integer constants.

G(maximum-message-size) specifies that execution of an input/output statement will result in the movement of a complete message to or from the message queue.

R(maximum-record-size) specifies that execution of an input/output statement will result in the movement of one record of a message to or from the message queue.

For both G and R formats, a buffer is always used, and its length will depend on the value of the specified decimal integer constant. The value that must be specified will depend on the message format as set up by the separate message control program. The PL/I programmer must have details of the message format in order to write a message processing program. In general, the messages and records are treated as if they were V-format records.

SPECIAL REQUIREMENTS

Execution of a program using TRANSIENT files will require that the user's generated system contains QTAM modules, and that a QTAM MCP (Message Control Program) is provided by the user.

QTAM Message Control Program

A QTAM message control program is required to direct incoming and outgoing messages to and from the main storage queues via a message data set on disk storage. Simple QTAM MCP macros are available to aid the user in generating the message control program.

Termination of the Teleprocessing System

In order to terminate the message control program, and thus the teleprocessing system, it is necessary to write an assembly language message processing program that will perform the following functions:

1. Make sure that all of the other processing programs in the system have either closed their QTAM data sets or that they are about to do so. One method could be to send to all the input queues a message which could be recognized by the processing programs as a message asking them to close the data sets.
2. Issue a QTAM CLOSEMC macro-instruction to terminate the entire system. (This macro-instruction passes control to QTAM termination routines.) It is not possible to issue this macro in PL/I processing programs, since it causes the execution of a program to be delayed until all other users of the QTAM data sets have closed them.
3. Return.

The action of the QTAM termination routines when given control by CLOSEMC is as follows:

1. The message control program will continue to receive the message currently being transmitted, but it will poll no further terminals after this transmission is complete.
2. As each terminal stops transmitting messages, the MCP will start sending messages waiting in the output queues. As the incoming traffic declines to nothing, the outgoing traffic continues until all outstanding messages have been sent.
3. Once the traffic has ceased, a check is made to ensure that all the QTAM data sets opened in the various message processing programs have been closed.
4. Finally, the entire system is terminated by closing all its lines and queue data sets.

Reactivation of the Teleprocessing System

The system is reactivated by executing the message control program in the normal way. All the processing programs could be started as required. Any messages that were still in the input queues when the system was terminated will still be available to the processing programs. Further information can be found in the publications IBM System/360 Operating System: QTAM Message Processing Services, Form C30-2003, and IBM System/360 Operating System : QTAM Message Control Program, Form C30-2005.

PROGRAMMING EXAMPLE

The programming example in Figure 1 is designed to illustrate that the PL/I teleprocessing extension is a simple extension of the existing record-oriented

```
                /*  AN INQUIRY HANDLING PROGRAM  */

CLOCK: PROCEDURE OPTIONS (MAIN);
    DECLARE (IN,OUT) FILE TRANSIENT KEYED ENV (G(50));
    DECLARE (INPUT_MESSAGE,TERMINAL) CHAR (50) VARYING;

       /* SET UP PENDING ON UNIT */
    ON PENDING (IN) DISPLAY ('CLOCK ROUTINE WAITING FOR WORK');

 GET_MESSAGE: READ FILE (IN) INTO (INPUT_MESSAGE) KEYTO (TERMINAL);

       /* IS IT END OF SESSION ? */

    IF INDEX(INPUT_MESSAGE,'END') ¬= 0
       THEN RETURN;

       /* VALIDATE INPUT */

    IF VERIFY(INPUT_MESSAGE,'TD ') ¬= 0
       THEN CALL ERROR(INPUT_MESSAGE,TERMINAL);

       /* NOW DETERMINE WHETHER TIME OR DATE
          IS REQUIRED                    */

    IF INDEX(INPUT_MESSAGE,'T') ¬= 0
       THEN CALL TIME_ROUTINE(TERMINAL);
       ELSE CALL DATE_ROUTINE(TERMINAL);
    GO TO GET_MESSAGE;


       /* SUBROUTINE TO RETURN RESPONSE */

TIME_ROUTINE: PROCEDURE (TERM);
    DECLARE (MSG,TERM) CHAR (50) VARYING;

       /* INVOKE PL/I BUILT IN FUNCTION TIME */

    MSG = TIME;
    GO TO WRITE_MESSAGE;
DATE_ROUTINE: ENTRY (TERM);

       /* INVOKE PL/I BUILT IN FUNCTION DATE */

    MSG = DATE;
    WRITE_MESSAGE: WRITE FILE (OUT) FROM (MSG) KEYFROM (TERM);
END TIME_ROUTINE;

       /* ERROR SUBROUTINE */

ERROR: PROCEDURE (MSG,TERM);
    DECLARE (MSG,TERM) CHAR (50) VARYING;
    DECLARE REPLY CHAR (23) INITIAL
                         ('YOUR MESSAGE IS INVALID');

       /* THIS ROUTINE LOGS THE ERROR AT THE SYSTEM
          CONSOLE AND INFORMS THE TERMINAL USER OF
          THE ERROR                             */

    DISPLAY ('BAD MESSAGE ''' ||MSG|| ''' FROM TERMINAL '||TERM);
    WRITE FILE (OUT) FROM (REPLY) KEYFROM (TERM);
    GO TO GET_MESSAGE;
END ERROR;

END CLOCK;
```

Figure 1. Teleprocessing Programming Example

transmission facilities, rather than to represent a typical user's message processing program.

The program in the example handles two types of message. The format of the messages is a single character, 'D' or 'T' surrounded by blanks. The 'D' character causes the date to be returned to the inquiring terminal, and 'T' causes the time to be returned. Any other character is an error. Note that the DD statements for the files IN and OUT would be DD DUMMY statements.

In the example, each inquiry is fully serviced before control returns to GET_MESSAGE to obtain the next message. Response could be greatly improved in an MVT system by use of the multitasking facilities of PL/I. If the routines DATE_ROUTINE, TIME_ROUTINE, and ERROR were invoked as subtasks of CLOCK, control would return to GET_MESSAGE as soon as the appropriate handling routine had been invoked, rather than after it had finished.

STRING-HANDLING ADDITIONS

Two new string built-in functions are implemented. These are the TRANSLATE and VERIFY built-in functions.

TRANSLATE returns a translation of a given string to the point of reference according to a translation table defined by two other strings. One example of its use is to enable items specified in character sets other than EBCDIC to be read in, translated into internal notation, and processed by the PL/I application program. Retranslation into character sets other than EBCDIC can be performed on output.

The VERIFY function verifies that each character or bit in a given source string is represented in a given verification string; in other words, it tests the validity of each character or bit according to user-specified criteria.

The TRANSLATE String Built-in Function

Definition: TRANSLATE returns the translated value of a specified string to the point of invocation. The translation is performed in accordance with a translation table supplied in the form of two arguments to the function.

Reference: TRANSLATE(s,r[,p])

Arguments: The argument "s" represents the source string, i.e., the string that supplies the value to be translated. Arguments "r" and "p" represent the replacement and position strings respectively; these strings correspond to each other to provide the translation table. If "p" is not specified, an implementation-defined character string is provided; for the F Compiler, this string consists of the 256 possible EBCDIC characters arranged in ascending order (i.e., from hexadecimal 00 through FF).

If any argument is arithmetic, it is converted to string (a character string if the argument is DECIMAL, or a bit string if the argument is BINARY). If, after any arithmetic-to-string conversion has been performed, all arguments are bit strings, no further conversion takes place; otherwise, bit-string arguments are converted to character strings. If "r" is shorter than "p," it is padded on the right (with blanks or zeros, depending on the string type) to the length of "p."

Result: The value returned by this function is a string identical in length and value to the source string "s," except that if any character/bit position of "s" contains a character or bit that has been specified for replacement (by inclusion of that value in the position string "p"), that value will be replaced by the corresponding value from the replacement string "r." The correspondence is by position: character/bit positions 1,2,3, ... n of "p" correspond respectively to character/bit positions 1,2,3, ... n of "r."

Example:

```
     DECLARE (S,T) CHAR(10),
             (P,R) CHAR(3);
     .
     .
     .
     P=',.$';
     R='.,D';
  A: GET DATA(S);
     T=TRANSLATE(S,R,P);
     PUT DATA(T);
     GO TO A;
```

The above sequence reads in data from SYSIN, translates commas to periods, periods to commas, and dollar signs to the character 'D', and writes out the result on SYSPRINT. Thus, if the string S='$12,345.50' were read in, the string T='D12.345,50' would be written out. (For the F Compiler, precisely the same result could be achieved by omitting P altogether and making R consist of the

EBCDIC sequence except for the replacement of the comma, period, and dollar-sign characters by the period, comma, and 'D' characters respectively.)

Note: Use of this function will in many cases result in the in-line use of the TR machine instruction.

The VERIFY String Built-in Function

Definition: VERIFY examines two given strings to verify that each character or bit in the first string is represented in the second string, returning a fixed binary value of 0 if this is the case; otherwise, the value returned is the index of the first character (in the first string) that is not represented in the second string.

Reference: VERIFY(expr-1,expr-2)

Arguments: The arguments "expr-1" and "expr-2" represent the source and verification strings respectively. If either argument is arithmetic, it is converted to string (a character string if the argument is DECIMAL, or a bit string if the argument is BINARY). If, after any arithmetic-to-string conversion has been performed, both arguments are bit strings, no further conversion takes place; otherwise, the bit-string argument is converted to a character string.

Result: The value returned by this function is a fixed binary integer of default precision ((15,0) for the F Compiler), determined as follows:

Each character or bit, c, of the source string is examined to see if it is represented anywhere in the verification string, i.e., to determine if

```
INDEX(expr-2,c)¬=0
```

The characters or bits of the source string are examined from left to right. If an examined character or bit is not represented in the verification string, the index of that character or bit in the source string is returned. If each character or bit in the source string is represented in the verification string, the returned value is zero.

Example: Assume that B is a character string of length 48, containing the 48 characters permitted in the 48-character set. The expression

```
VERIFY(A,B)
```

will then return a value of zero for any value of A that consists solely of characters from the 48-character set, but will index the first character in a value of A that does not conform to the 48 character set (e.g., if A = 'P GT X', the returned value is zero; if A = 'P > X', the returned value is 3).

Note: Use of this function will in many cases result in the in-line use of the TRT machine instruction.

OPTIMIZATION EXTENSIONS

The NORMAL, ABNORMAL, USES, and SETS attributes have been removed from the PL/I language, and these keywords will no longer be accepted by the F Compiler. (Previous versions of the compiler accepted the keywords without acting on them. For details of the effect on compatibility, see Appendix A of this publication.) Two new options (ORDER and REORDER) for PROCEDURE and BEGIN statements have been added, and are implemented in the fifth version of the compiler. This subsection describes these options in terms of the PL/I language, since they stipulate the rules that any compiler must observe during optimization. The way in which the F Compiler ensures that these rules are observed is discussed in Section 2, "Performance Improvements and Optimization." The REDUCIBLE and IRREDUCIBLE attributes are retained in the language, and the F Compiler will continue to accept them without taking action.

INTRODUCTION

Strictly speaking, the order in which the statements of a PL/I source program are to be executed is specified by the order in which they appear in the source program, even if the code could be reordered so as to produce the same result more efficiently. The order of execution is normally sequential except where modified by a control statement such as GO TO.

The programmer can vary the degree of language stringency imposed on the compiler by using the ORDER and REORDER options on the PROCEDURE and BEGIN statements. REORDER specifies a partial relaxation of the rules to allow the compiler more freedom in optimization. Whether the F

Compiler takes advantage of this relaxation (described in more detail below) depends on other factors than the option specified. Firstly, it will not try to optimize code unless it is obviously safe to do so; secondly, optimization of object code generally means an increase in compilation time, and so provision is made for the user to prevent the compiler from introducing the necessary extra phases. These two factors are discussed in more detail in Section 2, "Performance Improvements and Optimization."

LANGUAGE ADDITIONS

The syntax of the PROCEDURE and BEGIN statements is changed to allow the inclusion of the keyword ORDER or the keyword REORDER. The general format of the PROCEDURE statement will be as follows:

```
  entry-name:  [entry-name:]...
     PROCEDURE[(parameter[,parameter]...)]
     [OPTIONS (option-list)]
     [RECURSIVE] [RETURNS (attribute...)]
     [ORDER|REORDER];
```

A similar change is made to the general format of the BEGIN statement. (The inclusion of the RETURNS keyword in the PROCEDURE statement is another language change and is discussed in Appendix A, "Compatibility with Previous Versions").

ORDER and REORDER specify, for optimization purposes, the degree of language stringency to be observed during compilation of the block. The strict rules require that the source program should be compiled so as to be executed in the order specified by the sequence of the statements in the source program, even if the code could be reordered so as to produce the same result more efficiently. The relaxation allowed by REORDER is such that if computational or system action interrupts occur during execution of the block, the result is not necessarily the same as it would be under the strict rules.

The selected option applies to all nested blocks unless overridden; if neither option is specified, the option that applies to the containing block will be assumed. If the block is an external procedure, it will be assumed to have the ORDER option unless REORDER is explicitly specified.

The ORDER Option

The ORDER option specifies that the normal language rules are not to be relaxed; i.e., any optimization must be such that the execution of a block always produces a result that is in accordance with the strict definition of PL/I. This means that the values of variables set by execution of all statements prior to computational or system action interrupts are guaranteed in an on-unit entered as a result of the interrupt, or anywhere in the program afterwards. Note that the strict definition now allows the compiler to optimize common expressions (see note below), where safely possible, by evaluating them once only and saving the result, rather than re-evaluating for each reference. Consequently, object programs produced by the fifth version of the compiler may differ from those produced by the fourth version in respect of the number of computational or system action conditions raised during execution.

<u>Note</u>: A common expression is an expression that occurs more than once in a program but is obviously intended to result in the same value each time that it is evaluated, i.e., if a later expression is identical to an earlier expression, with no intervening modification to any operand, the expressions are said to be common.

The REORDER Option

The REORDER option specifies that execution of the block must produce a result that is in accordance with the strict definition of PL/I unless a computational or system action interrupt occurs during execution of the block; the result is then allowed to deviate as follows:

1. After a computational or system action interrupt has occurred during execution of the block, the values of variables modified, allocated, or freed in the block are guaranteed only after normal return from an on-unit or when accessed by the ONCHAR and ONSOURCE condition built-in functions.

2. The values of variables modified, allocated, or freed in an on-unit for a computational or system action condition (or in a block activated by such an on-unit) are not guaranteed on return from the on-unit into the block, except for values modified by

the ONCHAR and ONSOURCE pseudo variables.

A program is in error if a computational or system action interrupt occurs during the execution of the block and this interrupt is followed by a reference to a variable whose value is not guaranteed in such circumstances.

ADOPTION OF HALFWORD BINARY FACILITIES

With previous versions of the compiler, fixed binary variables of any precision were always mapped as fullwords (requiring four bytes of storage). The fifth version of the compiler will map fixed binary variables of precision less than 16 as halfwords (requiring only two bytes of storage), and will use System/360 halfword instructions to process them. Note that variables of default precision will be mapped as halfwords. This subsection describes the effect of this change on storage, data interchange, and compatibility.

The change does not apply to fixed binary constants or fixed binary intermediate targets (i.e., compiler created temporaries for holding intermediate results of operations). These will continue to occupy fullwords.

Effect on Storage

The total amount of main storage and external storage required at execution time by fixed binary variables of precision less than 16 will be halved; this includes storage for arrays and elements of structures.

Effect on Data Interchange

Communication between PL/I and other languages is improved: the feature permits easier data interchange between programs written in PL/I and those written in COBOL and FORTRAN. (Use of the COBOL option in the ENVIRONMENT atrribute allows data interchange with COBOL programs; use of the UNALIGNED attribute in conjunction with VS- or VBS-format records allows data interchange with FORTRAN programs.)

Compatibility

This change will affect compatibility with programs written for previous versions of the compiler only in relation to the following:

1. External fixed binary variables
2. Fixed binary arguments and parameters
3. Variables defined on external fixed binary variables
4. Fixed binary variables used in record-oriented transmission
5. Based fixed binary variables.

Full details of circumstances in which such incompatibilities could arise and methods of avoiding them are given in Appendix A of this publication.

Relaxation of REFER Option Restriction

The restriction on the two variables in the REFER option of the BASED attribute has been eased to permit fixed binary integer variables of the same precision as each other. This will allow the user the choice of either continuing to use fullword binary or using halfword binary for the controlling fields in self-defining structures.

INTRODUCTION

This section gives details of the improvements that can be expected in the execution speed of object programs produced by the fifth version of the F compiler compared with the speed of those produced by the fourth version. The degree of object program optimization attempted by the compiler depends on the PL/I block options ORDER and REORDER, and on the compiler option OPT. The descriptions of the specific areas of improvement that follow this introduction indicate the block and compiler options that should be specified in order to obtain the benefits of each feature; they also include estimates of the local performance improvement in each case.

Where it is indicated that optimization will be effected for both ORDER and REORDER, it is probable that specification of REORDER will result in the greater degree of optimization. However, even where REORDER is stated to be necessary for a particular type of optimization to occur, there will usually be some optimization when ORDER is specified.

For the fifth version of the F Compiler, the option OPT can be specified with one of three values:

OPT=0	requests fast compilation and, as a secondary consideration, reduction of the storage space required by the object program at the expense of execution time.
OPT=1	requests fast compilation and, as a secondary consideration, reduction of object program execution time at the expense of storage space.
OPT=2	requests reduction of object program execution time at the expense of compilation time.

Note that the new optimization phases of the compiler will be invoked only when OPT=2 is specified. The resultant degradation in compilation time has been estimated to be not greater than 22 percent when 88K bytes of storage are available for the compilation.

The performance improvement figures given below are average local improvements; this data is not necessarily representative of all programs and is given for guidance only.

It is not possible to quote a specific overall percentage improvement figure since individual programs will vary considerably. Estimates indicate an execution speed improvement ratio in the range 1:1 to 2.6:1.

LOOP AND SUBSCRIPT OPTIMIZATION

LOOP CONTROL MECHANISM

The mechanism of loop control will be simplified wherever possible. In particular, BXLE or BXH machine instructions will be generated rather than the present five-instruction sequence.

Block option: ORDER/REORDER

Optimization level: OPT=2

Local performance improvement: between 2 to 1 and 3 to 1.

LOOP CONTROL VARIABLES

The use of control variables as subscripts will be optimized.

Block option: REORDER

Optimization level: OPT=2

Local performance improvement: between 2 to 1 and n to 1 for addressing code, where n = number of times the control variable appears in subscript expressions.

ARRAY EXPRESSIONS

A combination of the techniques used for optimization of loop control mechanisms and control variables will be employed.

Block option: ORDER/REORDER

Optimization level: OPT=2

Local performance improvement: between 2 to 1 and n to 1, where n = number of arrays in the expression.

SUBSCRIPT LISTS

Identical expressions that will represent the same value will be replaced by temporary variables to which the value will be assigned. Expressions whose values will not change will be moved out of loops.

Block option: REORDER

Optimization level: OPT=2

Local performance improvement: n to 1, where n = number of common subscript expressions.

ADDITIONAL IN-LINE CONVERSIONS

In addition to those data conversions for which in-line code is generated by the third and fourth versions of the F Compiler, in-line code will be generated for certain cases of conversion between the following data types:

Numeric character to FIXED BINARY

FIXED BINARY to numeric character

Numeric character to FIXED DECIMAL

FIXED DECIMAL to numeric character

In-line code will be generated for these conversions where the picture specification comprises a string of 9s and includes an implied decimal point (e.g., '99V9'). The picture specification may also include editing characters preceding the 9s (e.g., '$$$999V99'), or embedded periods (e.g., '99.99V9.9').

Block option: ORDER/REORDER

Optimization level: OPT=0, OPT=1, or OPT=2

Local performance improvement: beween 5 to 1 and 11 to 1.

IMPROVED CODE FOR ASSIGNMENTS

Optimized code that does not use temporary storage will be produced in the following cases when FIXEDOVERFLOW and SIZE are disabled or cannot be raised, and when the operands are of suitable scale and precision:

1. Simple fixed-decimal assignments (for example, A = A + constant; X = A + B; X = A * B + C;).
2. Simple expressions and assignments that involve only character-string variables and character-string constants (for example, X = A||B;).
3. Assignments between temporary variables such as occur in some subroutine or function references.

Block option: ORDER/REORDER

Optimization level: OPT=0, OPT=1, or OPT=2

Local performance improvement: between 2 to 1 and 3 to 1.

IMPROVED REGISTER USAGE

Improvements in the register-allocation stage of the compiler can result in better use of registers during execution of the object program, thereby eliminating some intermediate store and load instructions.

Block option: ORDER/REORDER

Optimization level: OPT=2

Local performance improvement: 1.25 to 1 when applicable.

IMPROVED CODE FOR MATHEMATICAL BUILT-IN FUNCTIONS

The mathematical built-in functions have been recoded to use new algorithms and to exploit recent changes in the floating-point hardware.

Block option: ORDER/REORDER

Optimization level: OPT=0, OPT=1, or OPT=2

Performance improvement: increased accuracy of results of many mathematical built-in functions, and shorter execution time for some functions.

In general the fifth version of the PL/I(F) Compiler will not create programs differing significantly in their use of core storage from those created by the fourth version. There are however two areas where considerable gains can be made through using the fifth version. These items are the adoption of halfword binary storage and the creation of a single PL/I library that can be used concurrently by PL/I programs executed in an MVT environment. As use of both of these items is highly program dependent, a general statement cannot be made on the overall impact on the use of storage.

HALFWORD BINARY FACILITIES

The halfword binary facilities are already descibed in Section 1 under the heading "Adoption of Halfword Binary Facilities."

SHARED LIBRARY

This feature permits the selective building of a 'shareable' PL/I Library by means of the PL1LIB System Generation macro-instruction. This library can then be made resident in the LINKPACK area of the Operating System/360 (MVT Option only) at the time of Initial Program Load (IPL). The library can then be accessed from one or more regions concurrently through a communications routine link-edited with the user program. PL/I Library routines not built into the 'shareable' library will be automatically link-edited into the user's load module. Execution of a load module requiring the shared library will proceed normally even if the shared library was not made resident at IPL-time. In this case the shared library will be loaded dynamically at the start of execution into the user region or partition on MVT or MFT, or into the user program area in PCP.

Table 1 shows the 35 groups that a user can select for inclusion in his resident PL/I library, and the System Generation options required to obtain them. The 35 groups make up 8 "packages" covering Housekeeping functions, Conversions, String Functions, Array Functions, Arithmetic Functions (ADD, MIN, MULTIPLY, etc.), Mathematical Functions (SIN, COS, SQRT, etc.), STREAM Input/Output, and RECORD Input/Output.

The breakdown of the PL/I Library into groups by modules within package is given in Tables 2 through 9. It should be noted that some modules appear in several groups within a package; selection of these groups will cause the inclusion of only a single copy of any one module. Group 1 (the non-shareable modules) is the only group that spans more than a single package. In addition, each PL/I library module appears in only one package.

Table 1. Shared Library Feature - Module Groups Selected by Options in SYSGEN PL1LIB Macro

Group No.	Main Functions of the Group	Approx. Size of Group	PL1LIB SYSGEN Options to Select Group	
1	Non-shared modules	5500	Modules not sharable	
2	Multi-tasking storage management	7100	MODES=TASK	STORG=BASIC
3	Non-tasking storage management	5100	MODES=NOTK	STORG=BASIC
4	Error handler (ON-units)	1300		STORG=ERR
5	List processing and structure mapping	2700		STORG=LISTP
6	Basic conversion package	3900	MODES=REAL	CONVS=BASIC
7	Edit conversions	4300		CONVS=EDIT
8	Complex conversions	4400	MODES=CMPX	CONVS=BASIC
9	Bit string conversions	2700		CONVS=BIT
10	Character string conversions	3900		CONVS=CHAR
11	Picture conversions	7900		CONVS=PICT
12	Sterling conversions	4400		CONVS=STERL
13	Optimization=1 special conversions	5000		CONVS=OPT1
14	Bit string functions	3200		STRGS=BIT
15	Character string functions	1800		STRGS=CHAR
16	'STRING' b-i-f and PV	2700		STRGS=STR
17	Real non-interleaved arrays	2000	MODES=REAL	ARRAY=BASIC
18	Real interleaved arrays	2500	MODES=REAL	ARRAY=LEAF
19	Complex non-interleaved arrays	2500	MODES=CMPX	ARRAY=BASIC
20	Complex interleaved arrays	3200	MODES=CMPX	ARRAY=LEAF
21	Real arithmetic operators	1800	MODES=REAL	MATHS=BASIC
22	Complex arithmetic operators	4300	MODES=CMPX	MATHS=BASIC
23	Real short arithmetic functions	2700	MODES=REAL	MATHS=SHORT
24	Real long arithmetic functions	3800	MODES=REAL	MATHS=LONG
25	Complex short arithmetic functions	4800	MODES=CMPX	MATHS=SHORT
26	Complex long arithmetic functions	5500	MODES=CMPX	MATHS=LONG
27	Non-tasking data-directed I/O	5100	MODES=NOTK	STRIO=DATA
28	Non-tasking list-directed I/O	5200	MODES=NOTK	STRIO=LIST
29	Non-tasking edit-directed I/O	3100	MODES=NOTK	STRIO=EDIT
30	Multi-tasking data-directed I/O	5300	MODES=TASK	STRIO=DATA
31	Multi-tasking list-directed I/O	5300	MODES=TASK	STRIO=LIST
32	Multi-tasking edit-directed I/O	3200	MODES=TASK	STRIO=EDIT
33	Non-tasking record I/O	1700	MODES=NOTK	RECIO=BASIC
34	Multi-tasking record I/O	2400	MODES=TASK	RECIO=BASIC
35	Non-tasking record I/O wait	1100	MODES=NOTK	RECIO=WAIT
36	Multi-tasking record I/O wait	1300	MODES=TASK	RECIO=WAIT

Notes: 1. The group sizes given above are rounded up to the nearest 100 bytes.
2. The non-shared modules (Group 1) comprise those modules from the Housekeeping, String Function, and STREAM I/O Packages which cannot reside in the shared library.
3. When several groups from the same package are selected, the size of the resulting group is not necessarily the sum of the individual group sizes.

Table 2. Housekeeping Package

Module	Description	Group Number				
		1	2	3	4	5
IHETCV	Control Variable		X			
IHETEA	Event Variable		X			
IHETER	ON Field		X			
IHETEV	COMPLETION		X			
IHETPB	PRIORITY		X			
IHETPR	PRIORITY		X			
IHETSA	Storage Manag.t		X			
IHETSE	FINISH		X			
IHETSS	FINISH		X			
IHESAP	Storage Manag.t			X		
IHEOSS	FINISH			X		
IHEOSE	EXIT			X		
IHECKP	Checkpoint	X				
IHEDSP	Display	X				
IHEDUM	Dump		X	X		
IHESRT	Sort	X				
IHEERR	Error		X	X		
IHECFA	ONLOC				X	
IHECFB	ONCODE				X	
IHECNT	ONLINE				X	
IHESRC	)CHAR ONSOURCE)DATA)FILE				X	
IHESRD	ONKEY				X	
IHELSP	List Processing					X
IHESTR	Structure Mapping					X
IHEBEG	Terminal Error	X				
IHECFC	Mod 91 interrupts	X				
IHEM91	Mod 91 errors	X				
IHEMAI	Main	X				
IHEMSI	No Timer	X				
IHEMST	No TIME	X				
IHEMSW	WAIT I/O Event	X				
IHEOSD	Date		X	X		
IHEOSI	Delay	X				
IHEOST	Time		X	X		
IHEPTT	COPY Tasking	X				
IHEPRT	COPY No-tasking	X				
IHERES	Restart	X				
IHESIZ	Length PRV		X	X		

Table 3. Conversion Package

Module	Description	Group Number							
		6	7	8	9	10	11	12	13
IHE DIA	F format director	X	X						X
IHE DIB	E to Internal		X			X			
IHE DID	B to Internal		X		X				
IHE DIE	Picture to Internal		X				X	X	
IHE DIL	A/B error		X						
IHE DIM	C to Internal		X						
IHE DOA	Internal to F/E	X	X						X
IHE DOB	Internal to A		X						
IHE DOD	Internal to B		X						
IHE DOE	Internal to Picture		X				X	X	
IHE DOM	Internal to C		X						
IHE DMA	Conversion director	X	X	X	X	X	X	X	
IHE DNB	Arithmetic to Bit				X				
IHE DBN	Bit to Arithmetic				X				
IHE DCN	Bit to Character					X			
IHE DNC	Arith to Character					X			X
IHE KCA	Valid Dec. Picture						X		X
IHE KCB	Valid Sterling Picture							X	
IHE KCD	Valid Char. Picture					X			
IHE UPA	Address Real Complex			X					
IHE UPB	Address Imag. Complex			X			X		
IHE VCA	Arith. attributes			X					
IHE VCS	Complex to Internal			X					
IHE VFA	Binary to Decimal	X		X			X		
IHE VFB	Float to Fixed	X		X			X		
IHE VFC	Float to Float	X							
IHE VFD	Fixed to Float	X		X			X		
IHE VFE	Float to Float	X							
IHE VKB	Decimal to Packed						X		
IHE VKC	Sterling to Packed							X	
IHE VKF	Packed to Fixed						X		
IHE VKG	Packed to Sterling							X	
IHE VPA	Decimal to Binary	X		X			X		
IHE VPB	Decimal to F	X		X			X		
IHE VPC	Packed to E	X		X			X		
IHE VPD	Packed to Decimal	X		X			X		
IHE VPE	E/F to Packed	X		X			X		
IHE VPF	Decimal to Packed	X		X			X		
IHE VPG	Fixed to Float					X			
IHE VPH	Bit to Float				X				
IHE VSA	Varying Bit				X				
IHE VSB	Varying Bit/Character				X	X			
IHE VSC	Varying Character					X			
IHE VSD	Varying Bit/Character				X	X			
IHE VSE	Character to Picture					X			
IHE VSF	Bit to Picture				X	X			
IHE VQA	Float to Fixed								X
IHE VQB	Decimal to Arithmetic								X
IHE VQC	Arith to E/F/Char.								X

Table 4. STRING Function Package

Module	Description	Group Number			
		1	14	15	16
IHEBSA	And		X		
IHEBSO	Or		X		
IHEBSN	Not		X		
IHEBSC	Compare		X		
IHEBSM	Assign		X		
IHEBSK	Concat, REPEAT		X		
IHEBSD	Compare		X		
IHEBSS	Compare, SUBSTR		X		
IHEBSI	INDEX		X		
IHEBSF	BOOL		X		
IHEBSV	VERIFY	X			
IHEBST	TRANSLATE	X			
IHECSK	REPEAT			X	
IHECSC	Compare			X	
IHECSM	Assign, Fill HIGH/LOW			X	
IHECSS	SUBSTR			X	
IHECSI	INDEX			X	
IHESTG	STRING BIF				X
IHESTP	STRING PV				X
IHECSV	VERIFY			X	
IHECST	TRANSLATE			X	

Table 5. ARRAY Function Package

Module	Description	Group Number			
		17	18	19	20
IHEJXS	Indexer	X	X	X	X
IHEJXI	Indexer		X		X
IHENL1	ALL ANY	X	X	X	X
IHENL2	ALL ANY		X		X
IHESSF	SUM	X			
IHESMF	SUM		X		
IHESSG	SUM	X		X	
IHESMG	SUM		X		X
IHESSH	SUM	X		X	
IHESMH	SUM		X		X
IHEPSF	PROD	X			
IHEPDF	PROD		X		
IHEPSS	PROD	X			
IHEPDS	PROD		X		
IHEPSL	PROD	X			
IHEPDL	PROD		X		
IHEYGF	POLY	X	X		
IHEYGS	POLY	X	X		
IHEYGL	POLY	X	X		
IHESSX	SUM			X	
IHESMX	SUM				X
IHEPSX	PROD			X	
IHEPDX	PROD				X
IHEPSW	PROD			X	
IHEPDW	PROD				X
IHEPSZ	PROD			X	
IHEPDZ	PROD				X
IHEYGX	POLY			X	X
IHEYGW	POLY			X	X
IHEYGZ	POLY			X	X

Table 6. Arithmetic Function Package

Module	Description	Group Number	
		21	22
IHEXIB	X**N	X	
IHEXID	X**N	X	
IHEAPD	X**N	X	
IHEXIS	X**N	X	
IHEXXS	Shift	X	
IHEXIL	X**Y	X	
IHEXXL	X**Y	X	
IHEMZU	X*Y X/Y		X
IHEXIU	X**N		X
IHEMZV	X*Y X/Y		X
IHEXIV	X**N		X
IHEMZW	X*Y		X
IHEDZW	X/Y		X
IHEXIW	X**N		X
IHEXXW	X**Y		X
IHEMZZ	X*Y		X
IHEDZZ	X/Y		X
IHEXIZ	X**N		X
IHEXXZ	X**Y		X
IHEMXB	MAX MIN	X	
IHEMXD	MAX MIN	X	
IHEADD	ADD	X	
IHEMXS	MAX MIN	X	
IHEMXL	MAX MIN	X	
IHEMPU	MULTIPLY		X
IHEDVU	DIVIDE		X
IHEADV	ADD		X
IHEMPV	MULTIPLY		X
IHEDVV	DIVIDE		X

Table 7. Mathematical Function Package

Module	Description	Group Number			
		23	24	25	26
IHESQS	SQRT	X		X	
IHEEXS	EXP	X		X	
IHELNS	LOG	X		X	
IHESNS	SIN COS	X		X	
IHETNS	TAN	X		X	
IHEATS	ATAN	X		X	
IHESHS	SINH COSH	X		X	
IHETHS	TANH	X		X	
IHEHTS	ATANH	X		X	
IHEEFS	ERF	X		X	
IHESQL	SQRT		X		X
IHEEXL	EXP		X		X
IHELNL	LOG		X		X
IHESNL	SIN COS		X		X
IHETNL	TAN		X		X
IHEATL	ATAN		X		X
IHESHL	SINH COSH		X		X
IHETHL	TANH		X		X
IHEHTL	ATANH		X		X
IHEEFL	ERF		X		X
IHESQW	SQRT			X	
IHEEXW	EXP			X	
IHELNW	LOG			X	
IHESNW	SIN COS SINH COSH			X	
IHETNW	TAN TANH			X	
IHEATW	ATAN ATANH			X	
IHESQZ	SQRT				X
IHEEXZ	EXP				X
IHELNZ	LOG				X
IHESNZ	SIN COS SINH COSH				X
IHETNZ	TAN TANH				X
IHEATZ	ATAN ATANH				X
IHEABU	ABS			X	
IHEABV	ABS			X	
IHEABW	ABS				X
IHEABZ	ABS				X

Table 8. STREAM I/O Package

Module	Description	Group Number						
		1	27	28	29	30	31	32
IHEDDI	Read Data		X			X		
IHEDDO	Write/Convert Data		X					
IHEDDJ	Array address		X			X		
IHEDDP	Array subscript		X			X		
IHEDDT	Write Data Tasking					X		
IHEIBT	Tasking PUT					X	X	X
IHEIOA	GET		X	X	X	X	X	X
IHEIOB	Non-tasking PUT		X	X	X			
IHEIOC	GET string	X						
IHEIOD	Datafield handler				X			X
IHEIOF	Logical records		X	X	X	X	X	X
IHEIOP	Page/Skip/Line		X	X	X	X	X	X
IHEIOX	Skip/Column				X			X
IHELDI	Read List			X			X	
IHELDO	Write/Convert List			X			X	
IHETAB	Page/Line default		X	X		X	X	

Table 9. RECORD I/O Package

Module	Description	Group Number			
		33	34	35	36
IHEION	I/O transmitter route	X			
IHEOSW	Wait I/O EVENT			X	
IHEOCL	OPEN/CLOSE	X			
IHEINT	I/O transmitter route		X		
IHETSW	Wait I/O EVENT				X
IHEOCT	OPEN/CLOSE		X		

4. USABILITY IMPROVEMENTS

Certain improvements have been made to the compiler that increase its convenience to the user. This section describes these improvements, which comprise the following:

- Two additions (NCP and TRKOFL) to the option list of the ENVIRONMENT attribute
- Interception of Operating System ABEND occurrences by means of the STAE feature
- Improvements to the multitasking implementation
- Implementation of additions to the list of acceptable abbreviations of PL/I keywords

NEW ENVIRONMENT OPTIONS

In addition to the teleprocessing format options, G and R, described in Section 1, there are two further additions to the option list of the ENVIRONMENT attribute, NCP and TRKOFL. Each can alternatively be specified in the DD statement.

Track Overflow (TRKOFL)

The track overflow option specifies that records transmitted to a direct access storage device can be written on overflow tracks if necessary. It is specified as follows:

```
TRKOFL
```

This option is equivalent to the specification of "T" in the RECFM subparameter of the DCB parameter of the DD statement.

Asynchronous Operations Limit (NCP)

The asynchronous operations limit specifies the number of incomplete input/output operations with the EVENT option that are allowed to exist for the file at one time. The specification is as follows:

```
NCP(decimal-integer-constant)
```

The decimal integer constant must have a value in the range 1 through 99; otherwise, 1 is assumed and an error message is issued. This option is equivalent to the NCP subparameter of the DCB parameter of the DD statement.

IMPROVED ABNORMAL TERMINATION (STAE FEATURE)

With previous versions of the compiler, an abnormal termination of a PL/I program resulted in an Operating System ABEND; this meant that the PL/I program was not cleanly terminated. The fifth version of the compiler will use the STAE feature to intercept abnormal termination and provide a cleaner program termination.

MULTITASKING IMPROVEMENTS

A problem encountered with multitasking as implemented by the fourth version of the compiler was preventing two or more tasks from concurrently executing code that modified control blocks and chains (i.e., "soft" code). The fifth version prevents this loss of control by executing "soft" code within the master control task. This improvement is particularly important to Multiprocessing System users.

This change is transparent to the PL/I programmer, but will require that object modules for PL/I multitasking programs compiled by the fourth version be link-edited with the fifth version library.

ADDITIONS TO THE LIST OF ACCEPTABLE ABBREVIATIONS

The following abbreviations for file attribute keywords will be accepted by the fifth version:

Keyword	Abbreviation
BUFFERED	BUF
EXCLUSIVE	EXCL
SEQUENTIAL	SEQL
UNBUFFERED	UNBUF

Owing to the changes and improvements in PL/I and the five versions of the compiler and library, certain unavoidable incompatibilities have arisen between library modules of the different versions and releases. These changes are detailed in the PL/I(F) Programmers Guide; however, two general rules can be stated:

1. Compiled code from any release of the compiler must always be executed using a library of the same or a later release.
2. Library modules of different releases can be mixed only in the following circumstances:

    a. All link-edited library modules must belong to the same release.

    b. All dynamically linked or loaded library modules must belong to the same release,and must be of at least as late a release as the link-edited library modules.

NEW LIBRARY MODULES AND HALFWORD BINARY ADOPTION

The implementation of teleprocessing support, improved mathematical built-in functions, multitasking improvements, and halfword binary has made a considerable impact on the PL/I library. The changes in the fifth version fall into two major groups -- those due to halfword binary adoption, and those due to the other factors.

EFFECT OF NEW LIBRARY MODULES AND RECOMMENDED ACTION

Most of the changes to the PL/I library have been made so as to be transparent to the user. Action need be taken only in the following two cases:

1. Programs link-edited with first or second version libraries will not run with fifth version dynamic library modules. Any users of programs link-edited with first or second version libraries will need to link-edit their program object modules with the fifth version library before attempting to execute the programs in a fifth version system.
2. Users of fourth version multitasking features will need to link-edit their program object modules with the fifth version library before attempting to execute the programs in a fifth version system.

EFFECT OF HALFWORD BINARY AND RECOMMENDED ACTION

In the first through fourth versions of the PL/I (F) Compiler, all fixed binary variables occupy a fullword (4 bytes). In the fifth version, fixed binary variables of precision less than 16 occupy a halfword (2 bytes). This change will affect users who have explicitly declared fixed binary variables with precision of less than 16 or who have allowed variables to acquire fixed binary default precision.

The change to halfword binary can <u>only</u> impact users of the following features:

1. External fixed binary variables
2. Fixed binary arguments and parameters
3. Variables defined on external fixed binary variables
4. Fixed binary variables used in record-oriented transmission
5. Based fixed binary variables

To aid the users of the above features to determine if program changes are necessary, the various combinations of cases are given below:

1. Previous version load module (link edited program and library) with fifth version dynamic library:

    Changes are <u>not</u> required.

2. Previous version program object module(s) with fifth version library:

    Changes are <u>not</u> required.

3. Previous version program object module(s) with fifth version program object modules(s) and library:

Errors will occur if the fifth version modules communicate with previous version modules by use of the above features.

The problems can be avoided if the user either:

a. Recompiles the previous version procedures, unless case 4 or 5 applies.

b. Declares all halfword binary variables in the features above with precision greater than 15 <u>before</u> compiling the fifth version procedures.

4. Previous version source program(s) compiled with fifth version:

   Changes are not required <u>unless</u> the user has violated any of the existing PL/I language rules applying to argument/parameter matching, use of pointers, base matching for defined items, or record descriptions containing halfword binary itmes.

5. Fifth version program (load module) with previous version data sets:

   Errors will occur if the data set records contain halfword binary items.

   The problem can be avoided by declaring all the halfword binary items in the structures used for input from the previous version data sets with precision greater than 15, <u>before</u> compiling the procedures with the fifth version.

6. Previous version program with fifth version data sets:

   Errors will occur if the data set records contain halfword binary items.

   The problem can be avoided if the user either:

   a. Recompiles the program with the fifth version.

   b. Ensures that all fixed binary variables used in RECORD output structures are declared with precision greater than 15.

MANDATORY RETURNS KEYWORD ON PROCEDURE, ENTRY, AND %PROCEDURE STATEMENTS

The RETURNS keyword is now mandatory in PROCEDURE, %PROCEDURE, and ENTRY statements of function procedures when the function value attributes are explicitly specified. If omitted it will be diagnosed as an error and will be assumed to be present. In the first release of the fifth version the error will be of severity level "Warning"; in subsequent maintenance releases, the severity level of the diagnostic will be raised to "Error."

Example:

Before fifth version:

```
     P:PROC(A) FIXED BINARY;
```

Required for fifth version:

```
P:PROC(A) RETURNS(FIXED BINARY);
```

REMOVAL OF ABNORMAL, NORMAL, USES, AND SETS ATTRIBUTES

The attributes ABNORMAL, NORMAL, USES, and SETS have been removed from the PL/I Language. Any appearance of these attributes in programs is now invalid and will be diagnosed as an error and ignored. Prior to the fifth version they were recognized but not utilized.

The severity level of the diagnostic message in the first release of the fifth version of the compiler will be "Warning." In subsequent maintenance releases of the fifth version the severity level will be raised to "Error."

CORRECTION TO THE IMPLEMENTATION OF INDEX BUILT-IN FUNCTION

The INDEX built-in function is now correctly implemented as described in the PL/I Reference Manual. The only incompatibility that can arise is when the function has been incorrectly used in compilations prior to fifth version. The case in which error would occur is if one or both of the arguments were fixed binary variables or expressions. Prior to the fifth version these would have been converted directly to CHARACTER; in the fifth version, they are now correctly converted to BIT and then, if necessary, to CHARACTER.

PUBLICATIONS

Details of all the fifth version features described in this planning guide will be provided in revisions of the following publications, available with the fifth version:

IBM System/360 PL/I Reference Manual, Form C28-8201

IBM System/360 Operating System: PL/I (F) Programmer's Guide, Form C28-6594

IBM System/360 Operating System: PL/I Subroutine Library Computational Subroutines , Form C28-6590

SUPPORTING PROGRAMS FOR TELEPROCESSING

The following are required for the execution of a program that uses TRANSIENT files:

1. A generated operating system that includes the QTAM modules.

2. A QTAM message control program to direct the incoming and outgoing messages to and from main storage message queues via a disk data set. Simple QTAM macro instructions are available for use in generating this program. A QTAM message processing program to terminate the QTAM message control program is also required.

The following publications contain relevant information:

IBM System/360 Operating System: QTAM Message Processing Services, Form C30-2003

IBM System/360 Operating System: QTAM Message Control Program, Form C30-2005

MACHINE REQUIREMENTS

COMPILATION

The minimum requirement for the F compiler is a 64K System/360 machine. The compiler itself requires at least 44K bytes of main storage. The machine must include the floating-point and decimal instruction sets; it must also include the timer feature if the time taken for compilation is to be listed.

Direct-access storage space is required in the systems residence device, and the compiler data sets require devices as listed in Table 10.

EXECUTION

The main storage requirements of the object program are a function of the PL/I facilities used.

The machine on which the object program is executed must include the floating point and decimal instruction sets; it must also include the timer feature if the TIME built-in function or the DELAY statement is to be used.

OPERATING SYSTEM REQUIREMENTS

COMPILATION

The compiler is designed to operate under System/360 Operating System with a minimal control program. Use of input/output access methods is restricted to BSAM and QSAM. Option 6A (Time) is required if the time taken for compilation is to be listed.

EXECUTION

The object program must be executed under System/360 Operating System. Advantage can be taken of the shared library feature only with Option 4(MVT);

however, a program compiled and link edited for execution with a shared library can still be executed under PCP or MFT.

All object programs require the presence of the BSAM and QSAM modules in the generated operating system; a program that uses INDEXED or REGIONAL data set organization requires the ISAM or BDAM modules, respectively.

Option 6A (Time) is required if the TIME built-in function or the DELAY statement is to be used.

Users who wish to reassemble modules of the PL/I Subroutine Library require an assembler program with pseudo-register support (for example, the System/360 F-Level Assembler).

Table 10. Compiler Data Sets

Name	Function	Permissible devices	When required
SYSIN	System input	DASD, magnetic tape, card reader, paper-tape reader	Always
SYSPRINT	System printer	DASD, magnetic tape, printer	Always
SYSLIN	System linkage-editor input	DASD, magnetic tape	When LOAD option specified
SYSPUNCH	System card punch	DASD, magnetic tape, card punch	When DECK or MACDECK option specified
SYSUT1	Spill	DASD	When space required by compiler exceeds main storage allocated for compilation
SYSUT3	Preprocessor or translated 48-character set output	DASD, magnetic tape	When preprocessor or 48-character set used
SYSLIB	Preprocessor %INCLUDE	DASD, magnetic tape	Only when %INCLUDE is used

The following is a complete list of language changes implemented by the fifth version of the F Compiler, together with the topics in this publication that give details of the changes.

ADDITIONS

Feature	Associated Topic
ORDER option	Optimization extensions
PENDING condition	Teleprocessing support
REORDER option	Optimization extensions
TRANSIENT attribute	Teleprocessing support
TRANSLATE built-in function	String-handling additions
VERIFY built-in function	String-handling additions

CHANGES

Changed Feature	Associated Topic
%PROCEDURE statement	Mandatory RETURNS keyword
ABNORMAL attribute	Removal from language
BASED attribute	(See REFER option, below)
BEGIN statement	Optimization extensions
BUFFERED attribute	Teleprocessing support Additions to list of abbreviations
ENTRY attribute	Removal of USES and SETS attributes
ENTRY statement	Mandatory RETURNS keyword
ENVIRONMENT attribute	Teleprocessing support Usability improvements
EXCLUSIVE attribute	Additions to list of abbreviations
FIXED BINARY variables	Adoption of halfword binary facilities
INDEX built-in function	Correction to implementation (see Appendix A)
KEYED attribute	Teleprocessing support
LOCATE statement	Teleprocessing support
NORMAL attribute	Removal from language

Changed Feature	Associated Topic
ONKEY built-in function	Teleprocessing support (see PENDING condition)
PROCEDURE statement	Optimization extensions; Mandatory RETURNS keyword
READ statement	Teleprocessing support
REFER option	Adoption of halfword binary facilities
SEQUENTIAL attribute	Additions to list of abbreviations
SETS attribute	Removal from language
UNBUFFERED attribute	Additions to list of abbreviations
USES attribute	Removal from language
WRITE statement	Teleprocessing support

READER'S COMMENT FORM

IBM System/360 Operating System
PL/I (F) Version 5
Planning Guide

Form C33-0002-0

How did you use this publication?

As a reference source ☐

As a class-room text ☐

As a self-study text ☐

Based on your own experience, rate this publication:

As a reference source—Very Good ☐ Good ☐ Fair ☐ Poor ☐ Very Poor ☐

As a text—Very Good ☐ Good ☐ Fair ☐ Poor ☐ Very Poor ☐

What is your occupation?

We would appreciate your specific comments; please give page and line references where appropriate. If you wish a reply, be sure to include your name and address.

● Thank you for your cooperation. No postage necessary if mailed in U.S.A.

YOUR COMMENTS PLEASE . . . .

This SRL bulletin is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM CORPORATION

112 EAST POST ROAD,
WHITE PLAINS, N.Y. 10601.

Attention: Department 813 (HP)

fold fold

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM System/360 Printed in U.S.A. C33-0002-0

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]