File No. S360-29
Form GC28-2050-0

TSS

IBM Systems Reference Library

IBM System/360 Time Sharing System

Planning for PL/I

This planning aid, intended for use prior to the availability of the IBM System/360 Time Sharing System PL/I processor, will be replaced by reference documentation to be issued when TSS/360 PL/I is released.

The user will be able to write PL/I programs to be compiled under TSS/ 360 PL/I if he refers to this publication in conjunction with the manuals listed in the preface.

PREFACE

This publication is intended for systems analysts and programmers who will use it as a planning aid for the TSS/360 PL/I processor, before that becomes available. The PL/I language features described in IBM System/360 Time Sharing System: PL/I Reference Manual, Form C28-2045, and in IBM System/360 Time Sharing System: PL/I Library Computational Subroutines, Form C28-2046, correspond to the fourth version of the PL/I (F) compiler of IBM System/360 Operating System (OS/360). The initial release of the TSS/360 PL/I compiler will correspond to the fifth version of the OS/360 PL/I (F) compiler. This publication indicates the features that differ from those currently described for TSS/360.

Section 1 describes functional modifications to the PL/I language and compiler; Section 2 describes improvements in performance and optimization. The appendix provides an alphabetical list of the changed language features.

Several features have been added to the PL/I language so that users of the IBM System/360 Operating System can write teleprocessing applications programs. These new language features will be accepted by the compiler, but an attempt to execute statements containing these features will result in task termination in TSS/360.

PREREQUISITE PUBLICATIONS

The reader should understand the information in:

IBM System/360 Time Sharing System: PL/I Reference Manual, Form C28-2045

IBM System/360 Time Sharing System: PL/I Library Computational Subroutines, Form C28-2046

First Edition (February 1970)

Significant changes or additions to this publication will be provided in new editions or Technical Newsletters.

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.

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

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Time Sharing System/360 Programming Publications, Department 643, Neighborhood Road, Kingston, New York. 12401

CONTENTS

Functional additions to the TSS/360 PL/I compiler consist of: two string built-in functions (TRANSLATE and VERIFY); two optimization options (ORDER and REORDER); and the adoption of System/360 halfword binary facilities for fixed binary variables with precision less than 16.

STRING-HANDLING ADDITIONS

TRANSLATE returns a translation of a given string to the point of reference, according to a translation table defined by two other strings. Example: Enables items in any user-specified character set to be read in, translated into internal notation, and processed by the PL/I application program; also, retranslation into the same or another character set can be performed on output.

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

The TRANSLATE String Built-in Function

Description: 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: "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; a character-for-character map from "r" onto "p" defines the translation table. If "p" is not specified, an implementation-defined character string is provided; for this compiler, this string consists of the 256 EBCDIC characters arranged in ascending order, hexadecimal 00 through FF.

If any argument is arithmetic, it is converted to string; a character string if the argument is DECIMAL, a bit string if the argument is BINARY. If, after any arithmetic-to-string conversion, all arguments are bit strings, or all are character strings, no further conversion takes place; otherwise, bit-string arguments are converted to character strings.

When "r" is shorter than "p," it is right-padded (with blanks or 0's, 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." A change is made to the source string only when a 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;
```

That 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 SYSOUT. Thus, if the string S='$12,345.50' were read in, the string T'D12.345,50' would be written out. (In TSS/360, the same result can be achieved by omitting P and making R consist of the EBCDIC sequence, except for the replacement of the comma, period, and dollar sign by the period, comma, and 'D'.)

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

Description: VERIFY examines two given strings and returns a fixed binary 0 if each character or bit in the first string is represented in the second string; 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: "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 or both character 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).

Each character or bit, c, of the source string is examined to see if it is represented 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 a character or bit is not represented in the verification string, the return is the index of that character or bit in the source string. If each character or bit in the source string is represented in the verification string, the returned value is 0.

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

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

will then return a value of 0 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 (if A = 'P GT X', the returned value is 0; if A = 'P > X', the 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; these keywords will not be accepted by this compiler. (Previously, these keywords were accepted without being acted upon.) Two options (ORDER and REORDER) for PROCEDURE and BEGIN statements have been added. These options in the PL/I language stipulate the rules that any compiler must observe during optimization. The way in which this compiler ensures that these rules are observed is described in Section 2. The REDUCIBLE and IRREDUCIBLE attributes are retained in the language; this compiler will continue to accept them without taking action.

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 to produce the same result more efficiently. The order of execution is sequential, except where modified by a control statement such as GO TO.

However, the user can vary the degree of language stringency imposed on the compiler by using the ORDER and REORDER options in the PROCEDURE and BEGIN statements. REORDER specifies a partial relaxation of the rules to allow the compiler more freedom in optimization. Whether the compiler takes advantage of this relaxation depends on other factors than the option specified:

- The compiler will optimize code only where it can recognize that it is safe to do so.
- The user can prevent the compiler from introducing extra phases by avoiding optimization.

These two considerations are also described in Section 2.

LANGUAGE MODIFICATIONS.

The syntax of the PROCEDURE and BEGIN statements has been changed to allow the inclusion of the keywords ORDER and REORDER. The general format of the PROCEDURE statement:

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

For the BEGIN statement, similar format change has been made.

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 be compiled for execution in the sequence of the source program's statements, even if the code could be reordered to produce the same result more efficiently. The relaxation allowed by REORDER is such that if computa-

tional or system-action interruptions occur during execution of the block, the result might not be the same as it would be under the strict rules.

The selected option, ORDER or REORDER, 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, the ORDER option will be assumed, unless REORDER is explicitly specified.

ORDER

The ORDER option specifies that the language rules are to be maintained; any optimization must be such that execution of a block produces a result that is in accordance with the strict definition in PL/I. The values of variables set by execution of all statements prior to computational or system-action interruptions are guaranteed to be valid in an on-unit entered as a result of an interruption, or anywhere in the program afterwards. The strict definition allows the compiler to optimize common expressions, where recognizably safe, by evaluating them once and saving the result, rather than re-evaluating them for each reference.

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

REORDER

The REORDER option specifies that execution of the block must produce a result that is in accordance with the strict definition in PL/I unless a computational or system-action interruption occurs during execution of the block. When there is such an interruption , the result is allowed to deviate:

1. 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 that are modified, allocated, or freed in an on-unit for a computational or system-action interruption (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 interruption occurs during the execution of a block and this interruption is followed by a reference to a variable whose value is not guaranteed to be valid.

INDEX STRING BUILT-IN FUNCTION

The language has been changed to reflect these rules: When both arguments have either the binary or bit-string attribute, both are expressed as bit strings; in all other cases, both are expressed as character strings.

ADOPTION OF HALFWORD BINARY FACILITIES

Previously, with PL/I, fixed binary variables of any precision were stored as fullwords (four bytes). The compiler will now store fixed binary variables with precision less than 16 as halfwords (two bytes), and will use System/360 halfword instructions to process them. Variables of default precision will be stored as halfwords.

The change does not apply to fixed binary constants or fixed binary intermediate targets (i.e., compiler-created temporaries for holding intermediate results); these will be stored as fullwords. However, for more efficient execution time, store fixed binary variables as fullwords (i.e., specify precision 16 or greater).

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 any precision, as long as both precisions are the same. This allows the user the choice of either continuing to use fullword binary or using halfword binary for the controlling fields in self-defining structures.

RETURNS KEYWORD IN 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 explicityly specified. If RETURNS is omitted, the omission will be diagnosed as an error and the keyword will be assumed to be present. The error will have severity-level "warning."

Example:

Previously:

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

Now required:

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

ADDITIONS TO THE LIST OF ACCEPTABLE ABBREVIATIONS

File Attribute Keyword	Abbreviation
BUFFERED	BUF
EXCLUSIVE	EXCL
SEQUENTIAL	SEQL
UNBUFFERED	UNBUF

OTHER ENVIRONMENT OPTIONS (TRKOFL AND NCP)

Track overflow (TRKOFL) specifies that overflow tracks on direct-access storage devices can be used if necessary; specified as:

TRKOFL

Asynchronous operations limit (NCP) specifies the number of incomplete input/output operations with the EVENT option that are allowed to exist for the file at one time; specified as:

NCP (decimal-integer-constant)

The allowable range of the argument is 1 to 99; if nothing is specified, the system assumes 1.

Although these options are compiled correctly, they do not affect execution since TSS/360 does not support these features (except for NCP in BSAM).

TELEPROCESSING LANGUAGE FEATURES

Several features have been added to the PL/I language so that users of the IBM System/360 Operating System can write teleprocessing applications programs. These new language features will be accepted by the compiler, but an attempt to execute statements containing these features will result in task termination in TSS/360.

The new language features:

1. TRANSIENT file attribute -- Indicates that the file is to be associated with a teleprocessing data set. TRANSIENT, an alternative to DIRECT and SEQUENTIAL, can be specified only for RECORD KEYED BUFFERED files that have either the INPUT or the OUTPUT attribute.

2. PENDING condition -- Except when signaled, can only be raised during execution of a READ statement for a TRANSIENT file. Its form:

   PENDING (file-name)

3. ENVIRONMENT format options (G and R) -- Applicable only to the teleprocessing extension; one of these options must be specified for TRANSIENT files. They cannot be specified for DIRECT, SEQUENTIAL, or STREAM files; they cannot appear in conjunction with any other option of the ENVIRONMENT attribute. Their formats:

   G (maximum-message size)
   R (maximum-record size)

The TSS/360 PL/I compiler will include the optimization improvements incorporated in the fifth version of the PL/I (F) compiler of OS/360. The degree of optimization attempted by the compiler depends on the PL/I block options ORDER and REORDER, and on the value specified by the user in 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 for each feature.

When optimization will be effected for both ORDER and RFORDER, it is probable that REORDER will produce the greater degree of optimization. However, even when REORDER is necessary for a particular type of optimization to occur, there will usually be some optimization if ORDER is specified.

OPT can be specified:

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.

The new optimization phases of the compiler will be invoked only when OPT=2 is specified.

LOOP AND SUBSCRIPT OPTIMIZATION

Loop Control Mechanism

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

Block option: ORDER|REORDER

Optimization level: OPT=2

Loop Control Variables

The use of control variables as subscripts will be optimized.

Block option: REORDER

Optimization level: OPT=2

ARRAY Expressions

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

Block option: ORDER|REORDER

Optimization level: OPT=2

Subscript Lists

Identical expressions, representing 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

IMPROVED CODE FOR ASSIGNMENTS

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

1. Simple fixed decimal assignments (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 (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

IMPROVED REGISTER USAGE

Improvements in the register-allocation stage of the compiler may 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

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

CHANGES TO THE LIBRARY COMPUTATIONAL SUBROUTINES

To take advantage of the improved floating-point engineering change (IFPEC), the TSS/360 PL/I library computational subroutines will include the changes incorporated in the fifth version of the PL/I (F) compiler of OS/360. These changes will consist mainly of removal of coding that became redundant with the engineering change, and the readjustment of constants to the new precision.

IMPROVEMENTS IN USE OF STORAGE

The extent of the required private storage has been reduced by the adoption of halfword binary storage and the creation of a single PL/I library that can be shared by all users. These improvements are program-dependent; no general statement can be made about the overall effect on the use of storage.

Additions	Associated Topic
ORDER option	Optimization extensions
PENDING condition	Teleprocessing
REORDER option	Optimization extensions
TRANSIENT attribute	Teleprocessing
TRANSLATE built-in function	String-handling additions
VERIFY built-in function	String-handling additions
Changes	
%PROCEDURE statement	Mandatory RETURNS keyword
ABNORMAL attribute	Removal from language
BASED attribute	Adoption of halfword binary facilities
BEGIN statement	Optimization extensions
BUFFERED attribute	Additions to list of abbreviations
ENTRY attribute	Removal of USES and SETS attributes
ENTRY statement	Mandatory RETURNS keyword
ENVIRONMENT attribute	Teleprocessing
EXCLUSIVE attribute	Additions to list of abbreviations
FIXED BINARY variables	Adoption of halfword binary facilities
NORMAL attribute	Removal from language
PROCEDURE statement	Optimization extensions; Mandatory RETURNS keyword
READ statement	Teleprocessing
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

Planning for PL/I Printed in U.S.A. GC28-2050-0

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)