PR1ME Technical Update

Subject: Rev. 19 PL/I Subset G **Number:** 84

Revision: Rev. 19.0

Date: July 1982

Applicable Hardware: 50 Series CPUs

Applicable Software: PRIMOS

Documentation Impact: PL/I Subset G Reference Guide (IDR4031)

Abstract: At Master Disk Revision Level 19 (Rev. 19.0) many new features and documentation corrections have been implemented. Some of them include:

- PL/I-G supports four additional options: -MAP/-NO_MAP, -ERRTTY/-NOERRTTY, -ERRLIST, and -FRN.

- The OPTIONS(SHORTCALL) is useful for calling procedures with the PMA instruction JSXB instead of the more common PCL instruction.

- Constant -(2**15) is supported but constant -(2**31) is not.

- PL/I-G now supports large linesize I/O up to 2056 characters.

PR1ME **Computer, Inc.** 145 Pennsylvania Avenue, Framingham, Mass. 01701
(617) 879-2960

PTU 84

REV. 19 PL/I, Subset G

INTRODUCTION

This PTU updates the PL/I Subset G Reference Guide (IDR4031) at Master Disk Revision Level 19 (Rev. 19.0). It includes new features and documentation corrections from Revs. 18.2, 18.3, and 19.0. It also includes documentation corrections from the Rev. 18.4 Software Release Document.

FOUR NEW OPTIONS

The PL/I-G compiler has four new options:

- -MAP/-NO_MAP
- -ERRTTY/-NOERRTTY
- -ERRLIST
- -FRN

The -MAP option is the default and will not make any user visible changes. The -NO_MAP option will turn off the variable reference map at the end of the listing.

-ERRTTY, the default, lists errors on the terminal. -NOERRTTY suppresses error listing on the terminal.

-ERRLIST produces an errors-only listing file.

-FRN rounds the floating accumulator before storing a float bin (23).

THE OPTIONS (SHORTCALL) DECLARATION

The OPTIONS(SHORTCALL) declaration is useful for calling PMA procedures with the PMA instruction JSXB instead of the more common PCL instruction. A procedure call of this type is faster than one using PCL. However, the called procedure must be written to expect this kind of call. In Rev. 18 and Rev. 19, the only system subroutine that can (and must) be declared in this way is MKONU$.

The format of this declaration is:

```
DECLARE procedure-name ENTRY OPTIONS(SHORTCALL [stack-size] );
```

stack-size specifies the extra space needed for the calling procedure's stack. The default size is 8.

The call does not generate a new stack for storage, as does PCL. The calling procedure's stack space is used. Thus it may be necessary to specify stack size in the declaration in order to enlarge the calling stack. For example, MKONU$ requires a 28-word stack, so the user's stack must be large enough to accommodate this requirement. If stack size is not large enough, the return from the subroutine will cause unpredictable error messages.

Arguments may be used with the SHORTCALL option. The computer will set up the L register to point to a vector containing the addresses of the arguments, or, in the case of one argument, to the address of the argument itself. No type checking is done. For Rev. 19, there are no standard subroutine calls that require both SHORTCALL and argument passing.

OTHER NEW FEATURES

- Constant -(2**15) is supported but constant -(2**31) is not supported.
- PL/I-G now supports large linesize I/O up to 2056 characters.
- The TRANSLATE built-in function works for a shorter second argument.
- The first digit of a value greater than dec(14) has been dropped. The largest decimal digit that can be printed is f(14,*). If a program attempts to print a larger decimal digit, a size error will occur and the value will not be printed.
- The STRING built-in function now accepts a character string argument.
- An unconditional branch will be generated for a 'DO WHILE('1'b)' statement.

DOCUMENTATION CORRECTIONS AND CLARIFICATIONS

PICTURE Attribute

A pictured value of all 9's cannot contain blank characters. It can contain only numeric digits. Please add this information to the discussion of PICTURED DATA, pages 3-4 to 3-7.

Device Files

A device file may be opened with either of the following:

```
OPEN FILE(name) TITLE('device-name -DEVICE')
OPEN FILE(name) TITLE('@device-name')
```

However, the qualifier -DEVICE and the @ sign should not be used with SYSPRINT and SYSIN. This information should be added to the discussion of -DEVICE on page 11-4.

On page 11-5, the first item in the bullet list reads:

> If the name is a device name, -DEVICE is used.

This sentence should read:

> If the name is SYSIN or SYSPRINT, -DEVICE is used automatically, by default.

A-Format Input Conversion

The first paragraph of page 8-12 should read as follows:

> A-Format Input Conversion: For input conversion, when *w* is specified, the result is a character string containing the next *w* characters from the input-stream file. If *w* is not specified, characters are read up to the end-of-line or until the character string is filled, regardless of whether the string is character-varying or not.

PL/I Subset G Program Structure

On page F-2, the third item under Program Structure should read as follows:

> The following options of the PROCEDURE and ENTRY statements are not supported:
>
> ORDER*, IRREDUCIBLE*, REDUCIBLE*, REORDER*

Iterative-do Statement

The Note at the bottom of page 9-10 should be changed to read as follows:

> If both the TO option and the WHILE option are omitted, the do-group is executed once.

Character Data Alignment

The last three lines of page 3-7 should be replaced with the following line:

```
CHARACTER(N) [VARYING] [ALIGNED]
```

On page 3-8, the paragraph beginning "Nonvarying character string variables..." should be replaced with the following two paragraphs:

> The ALIGNED attribute aligns nonvarying character data long word boundaries. If ALIGNED is not specified, nonvarying character data is byte-aligned. The ALIGNED attribute has no effect on varying character data, which is always byte-aligned.
>
> Nonvarying character-string variables always occupy exactly n bytes of storage. As elements of arrays or members of a structure, they begin on the next available byte if they are byte-aligned. In this case, an array of nonvarying characters can be stored and accessed as if it were a single string. See Section 4 for a discussion of storage sharing.

On page 5-7, the paragraph on ALIGNED should read as follows:

> ALIGNED is an optional part of the bit-string and character string specifications. Its presence allows an implementation to align data on a convenient storage boundary (word boundary, for nonvarying character data). Specifying ALIGNED may cause a bit string to use more bits of storage than are specified by its declared length. See Section 3 for discussions of bit-string and character data.

The following two entries should be added to the list on page 11-2:

Character(n) Aligned	word	n bytes
Character(n) Varying Aligned	word	(n+3)/2 words

Also on page 11-2, the last sentence before the section on INPUT/OUTPUT ON TTY should read as follows:

> The ALIGNED attribute applied to character(n) varying data has no effect on the alignment of the data.

On page C-7, the entry labelled Alignment should read as follows:

> CHARACTER VARYING data is always byte-aligned. Nonvarying CHARACTER data is byte-aligned unless ALIGNED is specified, in which case it is aligned along word boundaries.

ROUND Function

On page 10-12, the second sentence in the section on ROUND(X,K) should read as follows:

> The result is always fixed decimal, and is the value of X rounded such that the K-th position of X is expressed to its nearest integer.