IBM

Series/1

GC34-0084-0

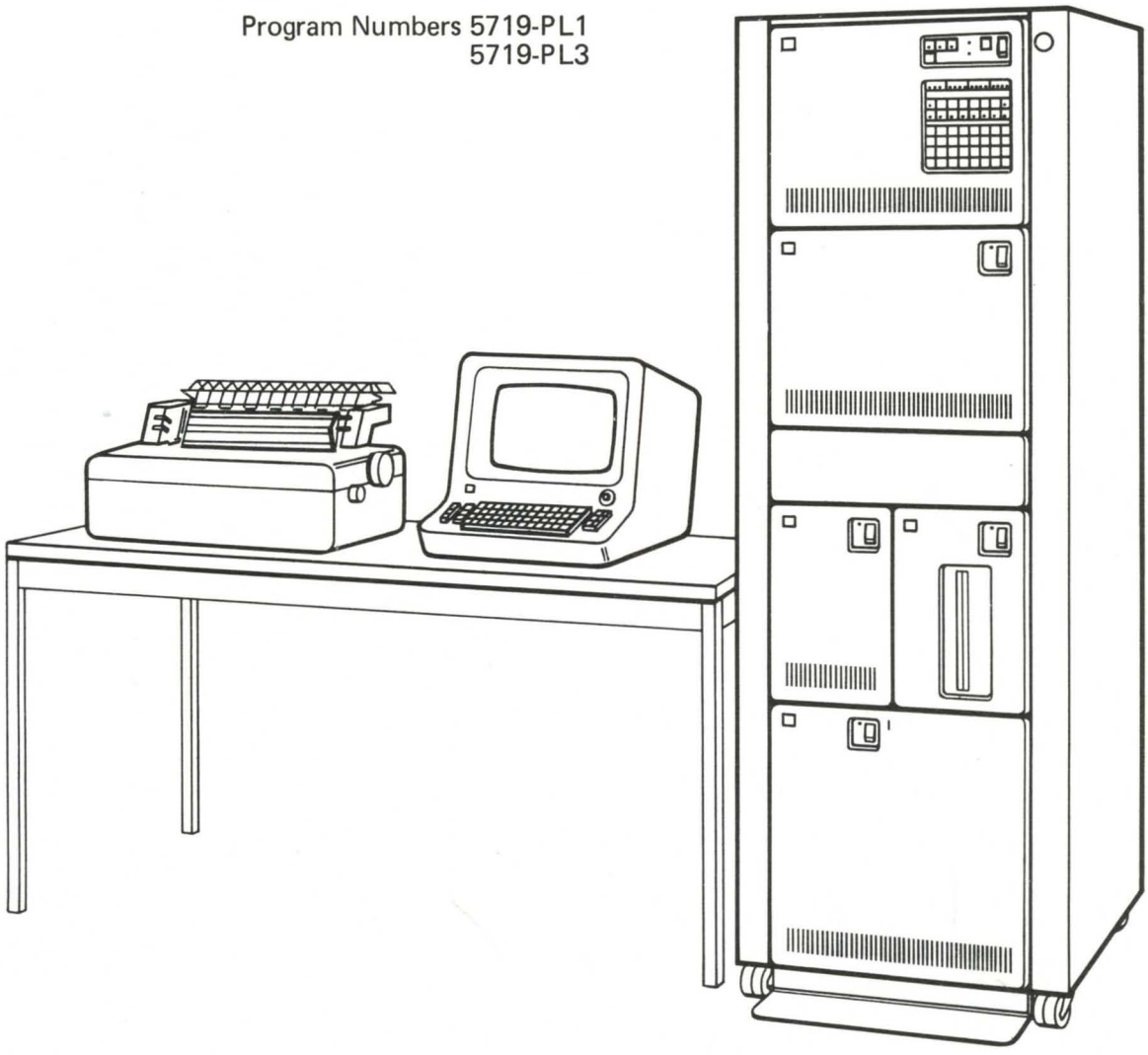
S1-29

PROGRAM
PRODUCT

PL/I INTRODUCTION

IBM Series/1
PL/I
Introduction

Program Numbers 5719-PL1
5719-PL3



IBM

Series/1

GC34-0084-0

S1-29

PROGRAM
PRODUCT

PL/I INTRODUCTION

IBM Series/1
PL/I
Introduction

Program Numbers 5719-PL1
5719-PL3

This publication is for planning purposes only. The information herein is subject to change before the products described become available.

First Edition (February 1977)

This manual applies to the IBM Series/1 PL/I Compiler and Resident Library (Program Number 5719-PL1), and the IBM Series/1 PL/I Transient Library (Program Number 5719-PL3).

Significant changes or additions to the contents of this publication will be reported in subsequent revisions or Technical Newsletters. Requests for copies of IBM publications should be made to your IBM representative or the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, send your comments to IBM Corporation, Systems Publications, Department 27T, P. O. Box 1328, Boca Raton, Florida 33432. Comments become the property of IBM.

Contents

This publication gives general information about the Series/1 PL/I Compiler.

The information provided is to be used as a planning aid only and is intended to introduce installation managers, systems analysts, and programmers to the facilities available with this compiler. The reader is assumed to know American National Standard (ANS) Programming Language PL/I (ANSI X3.53-1976).

"The Language Extensions Implemented by Series/1 PL/I" discusses those features which are not included in ANS PL/I, such as language extensions and sensor I/O.

"Productivity Features of Series/1 PL/I" discusses debugging and usability features.

"System Requirements" briefly describes both the machine and system requirements.

Appendix A is an alphabetical list of the PL/I and implementation-defined keywords.

Appendix B is a comparison, by function, of Series/1 PL/I and ANS PL/I.

Prerequisite Publications

- *IBM Series/1 System Summary*, GA34-0035, which contains an overview of the entire Series/1 hardware and software offering.

Related Publications

For information regarding the current availability of all related Series/1 publications, consult your IBM representative.

- *IBM Series/1 Realtime Programming System: Introduction and Planning Guide*, GC34-0102
- *IBM Series/1 Program Preparation Subsystem: Introduction*, GC34-0121
- *IBM Series/1 PL/I: Language Reference Manual.*

The Series/1 PL/I is a subset of ANS PL/I plus extensions. It requires two additional program products for its operation: IBM Series/1 Realtime Programming System (Program Number 5719-PC1) and IBM Series/1 Program Preparation Subsystem (Program Number 5719-AS1).

The Realtime Programming System is a multiprogramming, multitasking, event-driven, disk-based control system. It manages all physical resources—processor, storage, and devices. Its supervisor and data management services are the interface between your application programs and the Series/1 hardware. It supplies the environment for both realtime and batch programs.

The Program Preparation Subsystem is a set of programs that offers:

- A powerful program preparation tool for creating realtime and batch applications.
- A general purpose batch processing facility.

It consists of a Job Stream Processor, Text Editor, Macro Assembler, and Application Builder (which produces executable load modules from object modules produced by the language translators).

Series/1 PL/I consists of two products: (1) a compiler with a resident library (Program Number 5719-PL1) and (2) a transient library (Program Number 5719-PL3). The resident library is made part of your task set at execution of the Application Builder. The transient library is installed into the shared task set on the execution system to support your application object program.

The Series/1 PL/I language is extensive in function, aimed at allowing you to quickly develop efficient application programs that can easily be extended or changed. Highlights of the PL/I offering include:

- Language extensions
- Extensive Input/Output (I/O) capability
- Multiple data types and organizations
- Easy data manipulation features
- Productivity features
- Additional features.

Language Extensions

Language extensions allow starting of asynchronous tasks and programs, and synchronization of their execution. Powerful event handling and resource control statements allow you to easily code applications that involve response to realtime events and resolution of resource contention. The ability to handle sensor I/O is also provided.

Extensive I/O Capability

Series/1 PL/I supports both stream and record I/O. Stream I/O statements read and write data with a minimum of programming effort, because automatic formatting and conversion are provided. The following specific options are available:

- List-directed I/O. This facility allows you to input or output data with automatic formatting and conversion.

- Edit-directed I/O. A full range of format items, including picture qualifications and control, allows you to generate complex reports with a minimum of programming effort.

Record I/O statements allow you to have more control over your I/O. The following options are available:

- Consecutive synchronous I/O. This facility is available through the use of the READ, WRITE, and REWRITE statements. You can improve your execution-time performance by using the EVENT options for asynchronous I/O.
- Direct I/O. This facility is available through the use of the READ, WRITE, DELETE, and REWRITE statements with the KEY option. Asynchronous direct I/O is also permitted.
- Sensor I/O. The facility for handling both sequential and random sampling of analog and digital I/O is available through the use of the READ and REWRITE statements.
- Transient files. This form of file organization allows you to communicate data between Realtime Programming System queues using PL/I READ and WRITE statements. The PL/I program can detect and handle the empty queue situation by coding an ON-unit for the PENDING ON-condition.

Data Types and Organizations

Series/1 PL/I supports arithmetic data, string data, and program control data. Arithmetic data can be represented in either binary or decimal radix and can be either fixed or floating point. Fixed point word and doubleword precisions are supported. Decimal fixed point data can have up to 15 digit positions, with up to 15 fractional positions. String data can be either bit or character, with fixed or varying length attributes. Program control data can be label, event, activation, lock, or pointer. Entry and file parameters are also supported.

PL/I data may be organized into arrays of up to 15 dimensions or in structures (hierarchical collections of data, not necessarily of the same type). A structure can also be dimensioned.

This wide variety of data types and organizations allows you to operate on data in the manner that most naturally matches your conception of the problem.

Data Manipulation Features

Series/1 PL/I provides both ease of expression and programmer productivity, since it supports all major PL/I operators, data types, and statements. Of particular interest are

- Powerful string operations, including substrings, concatenation, and general boolean operations.
- Full set of language built-in functions, including mathematical functions, string functions, and array functions.
- Structure assignment.
- Automatic data conversions in expressions.
- Generalized subscripting.
- Full support for internal and external procedures.
- Control structures including IF—THEN, IF—THEN—ELSE, DO, and DO—WHILE.

Productivity Features

Included in this category are such features as

- Extensive compile-time diagnostic messages
- Compile-time listing aids
- Execution-time diagnostic messages
- User programming and control of error conditions via the PL/I ON-handling language.

Additional Features

Included in this category are those aspects of Series/1 PL/I that make it uniquely suitable as a general application development tool. Of particular interest are

- Storage efficiency gained by the generation of reentrant code and support for automatic storage allocation.
- Program modularity and interface checking provided by the PL/I block structure and scope rules and the ENTRY attribute.
- The ability to build and manipulate chained data lists, rings, and plexes using the PL/I list processing support; that is, the pointer data type and based storage.

Summary

Series/1 PL/I is aimed at decreasing application development time in areas such as realtime, scientific or problem solving, as well as traditional data processing. These features also make it useful when implementing advanced applications such as transaction processing and data base handling.

The approach taken for the language extensions in Series/1 PL/I is to extend the PL/I language to allow easy development of realtime applications, while simultaneously retaining the basic structure and philosophy of the PL/I language. To achieve this goal, extensions are provided in the following areas:

- Ability to schedule, execute, and control external procedures as independent parallel tasks.
- Ability to schedule and execute programs (task sets).
- Support for synchronization and control of program data and flow by using EVENT variables, LOCK variables, and deadlock avoidance.
- Extension of event concepts to recognize Time-of-Day events, events triggered by external causes (i.e., process interrupts), repetitive events, and resetting events.
- Extension of PL/I record I/O to handle digital and analog I/O.

The last extension is termed *sensor I/O*, while the remaining four are referred to as *multitasking*. Refer to Appendix B for a comparison of Series/1 PL/I to ANS PL/I.

Multitasking

The use of a computing system to execute a number of operations concurrently is broadly termed multiprogramming. You can make use of the multiprogramming capability of the system by using the multitasking facilities discussed in the following sections.

A PL/I program is a set of one or more procedures, each of which consists of one or more blocks of PL/I statements. The execution of these procedures constitutes one or more *tasks*, each of which can be identified by a different *activation name*. A task is dynamic; it exists only while the procedure is being executed. A task is *not* a set of instructions, but an execution of a set of instructions. The instructions themselves, as written by you, may in fact be executed several times in different tasks.

It is necessary for at least one task to exist when a PL/I program is executed. Thus, when a PL/I program is first entered, its execution is part of a task. This particular task is called the *major task*; it is created by the operating environment.

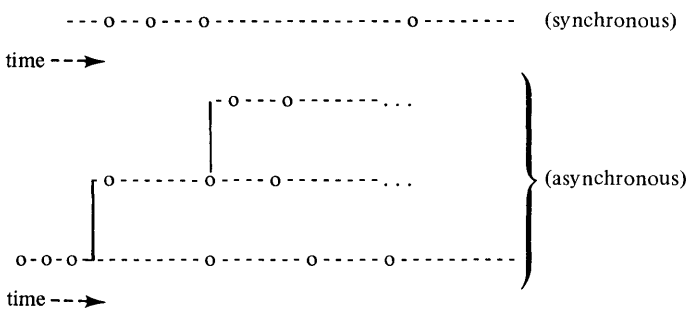
Synchronous and Asynchronous Operations

Unless the program specifies the creation of tasks, the execution of the statements of the program will proceed serially in time, according to the sequence designed by the order of the statements and the control statements. Such an operation is said to be *synchronous*. In addition to full facilities for conventional synchronous processing, means are provided for performing operations *asynchronously*.

Some reasons for considering the use of asynchronous operations are

- You may wish to make use of computer facilities which can operate simultaneously.
- A program may be written in which I/O units initiate or complete transmission at unpredictable times, such as disk operations, terminals.
- The asynchronous nature of realtime, event-driven applications.

The following diagram distinguishes between synchronous and asynchronous operations. (The circles represent statements.)



In an asynchronous operation, once a new line has been started, the statements on that line are executed in sequence and independently of the statements on any other line.

Creation of Tasks and Programs

Tasks or programs are created by execution of a RUN statement. The scheduled task or program will then be executed asynchronously with the scheduling task or program. The RUN statement itself is not part of the newly-created task or program.

Scheduled tasks must be external procedures. All tasks must have OPTIONS (TASK) on their PROCEDURE statement.

The options of the RUN statement have the following meaning:

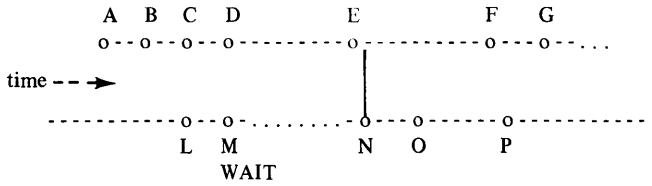
- ACTIVATION — Specifies a name for the activation variable created by the execution of the RUN statement. Specifying the name in this way is necessary if you wish to issue a STOP or UNSCHEDULE for the scheduled task or program in any other task.
- EVENT — Indicates that you wish to issue a WAIT statement which will wait on the completion of the task created by the RUN statement.
- PRIORITY — Indicates the priority of the task created by the RUN statement. When two or more tasks are able to continue, but external constraints prevent simultaneous execution of them all, then those with priorities of a higher numeric value will proceed first.
- AT — Indicates the time of day the task or program should start.
- AFTER — Gives a time interval after which the task or program should start.
- EVERY — Gives a time interval such that after the program is started, it should be repeatedly started at the specified interval.
- SOURCE — Indicates that the task or program is to start as the result of a process interrupt. This process interrupt is specified in an ENVIRONMENT option.
- ENVIRONMENT — Is used to define implementation characteristics of the event.

Coordination and Synchronization of Tasks

Synchronizing Two Asynchronous Operations

In order that the result of an asynchronous operation may be made available to other tasks, a WAIT statement can be used to synchronize two or more asynchronous operations.

The following diagram illustrates this



Assume that before statement N can be executed, both L and E must have been executed. M, therefore, issues a WAIT statement which will suspend operation on that line until E has been completed. After N, the statements O, P, ... are executed synchronously, as are the statements F, G, ...

Dynamic Descendance of Tasks and Programs

If, within the execution of a task, a block B is activated and control for that task stays at points internal to B until B is terminated, no other blocks can be activated within that task while B is active.

It is possible, however, for control of that task to pass outside B and cause activation of other blocks while B is still active for single tasking applications. It is also possible for a new task or program to be initiated during the activation of B by a RUN statement. All additional blocks activated in the orginal task and program are dynamic descendants of B, but all blocks in the new task or program are not dynamic descendants of B and of the blocks of which B is a descendant. None of the rules associated with dynamic descendance apply across task or program boundaries, for example, ON-units established prior to the scheduling of a task or program are not inherited by the scheduled task or program.

Sharing of Data Between Tasks

A program can share data between tasks only by declaring the data to be shared with either the EXTERNAL attribute, or by using BASED variables with a pointer which has external scope. In order to ensure predictable results when accessing such shared data, you should use the PL/I synchronization language, i.e., LOCKs, to force temporary serialization of data access. Naturally, such serialization is not necessary unless contention can exist.

The following rules should be considered when sharing data across task boundaries:

- Any generation of any variable of any storage class can be referred to in any task by means of an appropriate BASED variable reference. It is your responsibility to ensure the required variable is in an allocated state at the time of reference.
- Static variables may be referred to in any task in which they are known.
- Automatic variables can be referred to by any block dynamically descendant from the block which allocates them; however, not across task boundaries.

Sharing Data Between Programs

There is no data sharing defined, in the PL/I language, between programs. However, STATIC EXTERNAL data can be placed in the shared task set area

during application build. This can give the effect of data shared between programs.

Sharing Files Between Tasks

Files that are known between tasks can be opened by any task. Subsequent opens are ignored. Any task that knows a file can close it. When a task is exited, all internal files opened by that task are closed. Any task that knows a file can read from or write to that file. More than one file may be associated with the same data set.

Termination of Tasks and Programs

A task may be terminated in one of the following ways:

- Control for the task reaches a RETURN or END statement for the procedure invoked with a TASK option.
- Control for any task reaches a STOP statement.
- Another task issues a STOP statement naming the activation variable of the program or task to be terminated.

Notice that a task cannot be terminated by a GOTO statement which transfers control out of the task. The execution of such a GOTO statement is an error.

A program can be terminated by issuing a STOP statement from within the program. A program can terminate another program by issuing a STOP naming the activation variable associated with the other program.

When a task is terminated, the following actions take place:

- All I/O events, which were initiated in that task and which are not yet complete, are set complete. Their results are not defined.
- All locks held by the task are unlocked.
- All internal files, which were opened during that task and are not yet closed, are closed.
- All active blocks in the task are terminated.
- If the task is terminated by any statement other than a RETURN or an END statement in this task, the status value of the event variable associated with the task is set, as is the completion value of the event.

Variables which were being assigned at the time of task termination, or data sets associated with OUTPUT or UPDATE files which were being created or updated at the time of task termination, may not have defined values after termination. It is your responsibility to ensure that assignment to variables or transmission to files is properly completed before the task performing these operations terminates.

Attributes

The following is a list of the attributes associated with multitasking and their functions:

- EVENT — Specifies that the associated identifier being declared is used as an event name. Event names are used to investigate the current state of tasks or of asynchronous I/O operations. They can also be used as program switches to control synchronization of parts of a program.
- LOCK — Specifies that the associated identifier being declared is used as a lock name. Lock names are used as logical gates for task synchronization purposes (primarily to data).
- ENVIRONMENT — Is allowed to be specified with the EVENT attribute. It is used to describe implementation

defined characteristics of the event. These characteristics become associated with the event when it becomes connected. This can then cause the completion and status of the event to be set in an implementation way. For example, process-interrupt association can be specified in the ENVIRONMENT option.

- AT — Specifies a time of day. When the event is connected, it should be posted at that time of day.
- AFTER — Specifies a time interval after which an event should be posted. The interval starts when the event is connected.
- SOURCE — Indicates that the event will be posted by a process interrupt described in the ENVIRONMENT option of the event declaration.
- ACTIVATION — Describes a variable that may be used as task or program identifier to test priority, UNSCHEDULE, or STOP the task or program.

Built-In Functions

The multitasking built-in functions are used during multitasking and during asynchronous I/O operations. They allow you to investigate the current state of execution of a task or asynchronous I/O operation.

The function names and definitions are

- COMPLETION — Returns the completion value of a given event expression. The event can be associated with completion of a task; completion of DISPLAY, AT, AFTER, SOURCE; completion of an I/O operation; or it can be user-defined. It can be connected or not connected.
- STATUS — Returns the status value of a given event expression. The event expression represents the event whose status value is to be returned. It can be connected or not connected.
- PRIORITY — Returns the priority of a task relative to the current task.
- DAYNO — Returns the number of the current day of the year.

Pseudo-Variables

In general, pseudo-variables are certain built-in function names that can appear wherever other variables can appear in order to receive values. In short, they are built-in function names used as receiving fields.

STATUS is a pseudo-variable that resets the status value of an event-expression.

Statements

The following is a list of the statements associated with multitasking and their functions:

- RUN — Invokes a task or program.
- STOP — Causes termination of a program or task.
- UNSCHEDULE — Eliminates the scheduling of a task or program that has already been scheduled with an AT,

	AFTER, EVERY, or SOURCE that has not yet been satisfied.
• TRANSFER TO	Stops the current program and "simultaneously" starts the specified program.
• WAIT	Retains control within the activated block until certain specified events have completed.
• LOCK	Seizes a lock or tests whether a lock is held by another task. The purpose of the statement is to seize a logical gate which allows synchronization for access to data, control flow, etc.
• UNLOCK	Removes the specified lock from its locked state. The purpose is to free a logical gate which is used for synchronization of access to data or control flow.
• CONNECT	Causes the connection of the event to the posting options described in the AT, AFTER, or SOURCE and ENVIRONMENT attributes of the event declaration.
• DISCONNECT	Makes an event not connected. This is used, primarily, to cause the disconnection of the event from the AT, AFTER, or SOURCE attributes, or the EVENT option of I/O, DISPLAY, or RUN.
• POST	Sets the completion value to '1'B and, optionally, the status value of disconnected event variables. This is done in one uninterruptable sequence.
• CLEAR	Clears the completion and status values of event variables.

Sensor I/O

The general approach taken for sensor I/O on the Series/1 is to use existing PL/I record I/O statements to access analog and digital data. Slight restrictions in the functions supported by record I/O are necessary to accommodate the implementation model of sensor I/O.

General Description

Prior to execution of a task, sensor I/O points are defined in a table which is external to the PL/I program. All of the physical sensor I/O points need not be defined in the table; sensor I/O may be physically accessed sequentially, in which case only the first point needs to be defined. However, if the points are to be accessed directly, that is randomly, each point must be defined. Each point so defined is accessed using an external reference to its address in the table.

One or more sensor I/O points may be accessed as one record. Sensor I/O records, which are described in the ENVIRONMENT attribute of a file declaration, may be either type S (sequential sampling) or type R (random sampling). All of the records in a sensor I/O file declaration must be the same type, and there must be a type S or R description for each sensor I/O record to be accessed.

Sensor I/O records are accessed by their relative position in the ENVIRONMENT attribute. If SEQUENTIAL is specified, the first READ/WRITE accesses the first record; the second READ/WRITE accesses the second record; etc. If KEYED SEQUENTIAL or DIRECT is specified, the KEY option is used to access any record specified in the ENVIRONMENT attribute.

The type S record may be used to access sensor I/O points which are physically consecutive. Access begins at the point specified by you and continues for the specified number of points. This method of accessing sensor I/O points is called sequential sampling.

The type R record may be used to access sensor I/O points which may not be physically consecutive. Access begins at the point specified by you and continues to the next point specified by you. This method of accessing sensor I/O points is called random sampling.

Sensor output points can be files declared as UPDATE or INPUT files—*not* OUTPUT. This restriction is required because the WRITE statement required for an OUTPUT file indicates the *addition* of a record, yet the number of records in a sensor I/O file is fixed before execution. Declaring the file UPDATE also allows reading an output point to determine the last value written to that point. If it is desirable to only read from the output points, the file can be declared INPUT.

The DELETE statement cannot be used, because the number of sensor I/O records is fixed before execution. REWRITE must be used to output a record to sensor output files, because such files can be declared only as UPDATE files.

Accessing Sensor I/O Records

Input Files

PL/I access to records in an ANALOG or DIGITAL INPUT file can be SEQUENTIAL, KEYED SEQUENTIAL, or DIRECT.

- If SEQUENTIAL is declared, the first READ statement accesses the first record specified in the ENVIRONMENT attribute, the second READ statement accesses the second record, etc.
- If KEYED SEQUENTIAL is declared, the KEY option may be used on a READ statement to access a particular record. If the KEY option is not used, the records are accessed sequentially.
- If DIRECT is declared, the KEY option *must* be used on all READ statements associated with the file.
- The KEYTO option may be used on READ statements associated with KEYED SEQUENTIAL files. When the KEYTO option is used, the relative record number of the record is assigned to the KEYTO character string variable.

Update Files

PL/I access to records in an ANALOG or DIGITAL UPDATE file can be SEQUENTIAL, KEYED SEQUENTIAL, or DIRECT.

- If SEQUENTIAL is declared, access to the records proceeds as described above for SEQUENTIAL input files. The REWRITE statement is used to output the record because sensor output files *must* be declared UPDATE.
- If KEYED SEQUENTIAL is declared, access to records proceeds as described above for KEYED SEQUENTIAL input files.
- If DIRECT is declared, access to the records proceeds as described above for DIRECT input files. The KEY option *must* also be used.

If KEYED SEQUENTIAL or DIRECT is specified, the KEY option is used to access any record specified in the ENVIRONMENT attribute. Therefore, the KEY option expression *must* evaluate to a relative record number; for example, 3 for the third record.

The following chart lists the allowable transmission statements and options for sensor I/O files. An X under a file attribute and accessing method indicates that the statement and option is allowable.

INPUT			UPDATE			Transmission
DIRECT	*SEQL.*	*SEQL. KYD*	*DIRECT*	*SEQL.*	*SEQL. KYD*	*Statements and options*
	X	X		X	X	READ INTO
		X			X	READ INTO KEYTO
X		X	X		X	READ INTO KEY
	X	X		X	X	READ IGNORE
	X	X		X	X	READ INTO EVENT
		X		X	X	READ INTO KEYTO EVENT
X		X	X		X	READ INTO KEY EVENT
	X	X		X	X	READ IGNORE EVENT
				X	X	REWRITE FROM
			X		X	REWRITE FROM KEY
			X		X	REWRITE FROM KEY EVENT

Analog Input

When an Analog Input (AI) record is accessed, the Analog to Digital Converter (ADC) returns a coded representation of the point value, having the attributes BIT(16). Truncation and/or loss of precision may occur if the target variable does not have the correct attributes.

When an AI point is defined external to PL/I, a range code is specified which indicates the amplifier gain to be used; this is the code used when the point is read, unless it is overridden by the GAIN option.

Without the GAIN option, the AI points are read as follows:

- Type S record — Each point in the record is read with the defined range code of the point indicated by you in the "address" operand of the record declaration.
- Type R record — Each point in the record is read with the defined range code for that point.

With the GAIN option and with either type S or R records, each point in the record is read with the range code specified by you; or, if you did not specify one, each point is read with the highest valid range code possible for that point.

Automatic zero correction can be performed when reading AI points by specifying ZEROCOR with the sensor I/O record in the ENVIRONMENT attribute.

Analog Output

The coded value to be written to an Analog Output (AO) point should have the attributes BIT(10) because these are the attributes of the value presented to the AO hardware. No implicit conversion to BIT(10) is performed. Similarly, since AO points can be read from as well as written to, the coded value read from an AO point will have the attributes BIT(10).

As previously explained, AO files may be declared UPDATE or INPUT. This allows reading of an AO point to determine the last value written to that point. The REWRITE statement *must* be used to output the value to an AO point. WRITE and DELETE are not allowed.

Records in an AO file may be accessed sequentially or directly. If an AO file is declared SEQUENTIAL, a REWRITE statement causes output values to be written to the analog points indicated by the current record. If the file is declared DIRECT or KEYED SEQUENTIAL, a REWRITE statement causes output values to be written to the analog points indicated by the record specified by the KEY option expression.

Digital Input

When a Digital Input (DI) record is accessed, a logical DI group bit string is returned to you. A logical DI group consists of from 1 to 16 contiguous bits in a physical DI group, or spanning two groups. (Since the physical DI group is never accessed as such, the following references to "DI group" imply "logical DI group".)

The area to which the DI group bit string is assigned should be connected storage.

Digital Output

When a Digital Output (DO) record is accessed, a bit string value is written to a logical DO group. A logical DO group consists of from 1 to 16 contiguous bits in a physical DO group, or spanning two groups. (Since the physical DO group is never accessed as such, the following references to "DO group" imply "logical DO group".)

As previously explained, DO files may be declared UPDATE or INPUT. This allows reading of a DO group to determine the last value written to that group. The REWRITE statement must be used to output the value to a DO group. WRITE and DELETE are not allowed.

Records for a DO file may be accessed sequentially or directly. If the file is declared SEQUENTIAL, a REWRITE statement causes a bit string value to be written to the DO group indicated by the current record. If it is declared DIRECT or KEYED SEQUENTIAL, a REWRITE statement causes a bit string value to be written to the DO group indicated by the record specified by the KEY option expression.

Debugging Features

Series/1 PL/I provides an extensive range of debugging features to decrease the time and effort required for program writing and checkout. These features are provided at compile-time and at program execution time.

Compile-Time Debugging Features

Diagnostics

Comprehensive diagnostic messages are provided by the Compiler. They provide an explicit description of the error.

Each error message indicates the statement number of the erroneous statement, the source item within the statement involved (if applicable), an explanation of the error, and any assumptions made or actions taken by the Compiler. Messages are graded into 5 severity levels. Optionally, only error messages above a specified severity level are printed.

Listing Aids

The Compiler provides listings and maps which will assist in writing and checkout of your PL/I programs. These include:

- Source listing with Compiler generated statement numbers.
- Attribute and cross-reference listing for variables used in the PL/I program.
- Object code listing (in Assembler language type format).
- Lists of offsets of object code statements and variables.
- Storage and control block requirements.
- External Symbol Dictionary.

Execution-Time Debugging Features

Diagnostics

Comprehensive diagnostic messages are provided at execution time. Each message indicates the name of the procedure block in which the error occurred, the offset of the instruction in error, and an explanation of the error. The offset, together with the statement offset list generated by the Compiler allows you to determine the PL/I source statement associated with the error.

Implementation of PL/I Error Handling Features

Series/1 PL/I supports PL/I error handling features that enable you to detect errors at execution time. Significant of these features are

- ON-conditions, which enable you to specify what action is to be taken if a particular error condition occurs.
  The ON-conditions supported are specified in Appendix B.
- The SNAP option on the ON statement which gives you a calling trace of the path to the statement in error.
- The condition built-in functions which enable you to determine more information about the source of the error, and which can sometimes also be used to correct the error.
  The condition built-in functions supported are specified in Appendix B.

Usability Features

To aid programmer productivity there are Compiler features to enable you to control the operation of the Compiler to best suit your environment and mode of development.

Compiler Options

You can specify Compiler options which control the operation of the Compiler. These options enable you to:

- Control the format of the source program that the Compiler can accept (e.g., 48- or 60-character set, the position of the source on the input records).
- Control the format of the printed output.
- Control which of the Compiler listings and maps will be printed.
- Suppress the printing of error messages below a specified priority.
- Specify that, depending on the severity of errors found, compilation will be restricted to syntax checking.

The options used for the compilation of a program are specified in several ways. The Compiler assumes a set of default options. These can be modified by parameters on the procedure or control language statement which invokes the Compiler, or by options specified on a special control statement (*PROCESS statement).

Multiple Compilations

The Series/1 PL/I Compiler allows compilations of more than one PL/I source program during one invocation on the Compiler. The separate PL/I programs are separated by *PROCESS statements; these can be used to specify different options for each compilation. This feature reduces the job stream processing overhead.

The minimum system requirements for *compilation* are

- Series/1 Processor with 64KB
- Series/1 PL/I Compiler and Resident Library
- 28KB partition in primary storage for the PL/I Compiler
- Series/1 Realtime Programming System
- Series/1 Program Preparation Subsystem

The minimum system requirements for *application build* are

- Series/1 Processor with at least 48KB
- Series/1 PL/I Compiler and Resident Library
- 16KB partition in primary storage for the application builder
- Series/1 Realtime Programming System
- Series/1 Program Preparation Subsystem

The minimum *execution* requirements in a *realtime partition* are

- Series/1 Processor with at least 48KB
- Series/1 PL/I Transient Library
- Primary storage partition sufficient for the PL/I program and its library support
- Series/1 Realtime Programming System

The minimum *execution* requirements in a *batch partition* are

- Series/1 Processor with at least 48KB
- Series/1 PL/I Transient Library
- Primary storage partition sufficient for the PL/I program and its library support
- Series/1 Realtime Programming System
- Series/1 Program Preparation Subsystem

Note. Refer to the Related Publications in the Preface to determine the minimum system requirements for the Series/1 Realtime Programming System and the Program Preparation Subsystem.

Appendix A. Summary of Keywords

The following is a complete list of the PL/I and implementation-defined keywords implemented by the Series/1 PL/I compiler.

Keyword	*Abbreviation*	*Use of Keyword*
A		format item
ABS		built-in function
ACOS		built-in function
ACTIVATION	ACTN	attribute, option of RUN, STOP, and UNSCHEDULE statements
ADDR		built-in function
AFTER		attribute, option of RUN statement
ALIGNED		attribute
ALL		option of UNLOCK statement
ANALOG		option of ENVIRONMENT attribute
ASCII		option of ENVIRONMENT attribute
ASIN		built-in function
AT		attribute, option of RUN statement
ATAN		built-in function
ATAND		built-in function
AUTOMATIC	AUTO	attribute
B		format item
BASED		attribute
BEGIN		statement
BINARY	BIN	attribute, built-in function
BIT		attribute, built-in function
BLKSIZE		option of ENVIRONMENT attribute
BOOL		built-in function
BUFFERS		option of ENVIRONMENT attribute
BUILTIN		attribute
BY		clause of DO statement
CALL		statement
CHAR		built-in function
CHARACTER	CHAR	attribute
CLEAR		statement
CLOSE		statement
COLUMN	COL	format item
COMPLETION	CPLN	built-in function
CONDITION	COND	attribute, condition
CONNECT		statement

Keyword	*Abbreviation*	*Use of Keyword*
CONNECTED	CONN	attribute
CONSECUTIVE		option of ENVIRONMENT attribute
CONVERSION	CONV	condition
COPY		built-in function
COS		built-in function
COSD		built-in function
COUNT		built-in function
CTLASA		option of ENVIRONMENT attribute
DATE		built-in function
DAYNO		built-in function
DECIMAL	DEC	attribute, built-in function
DECLARE	DCL	statement
DELAY		statement
DELETE		statement
DEVNBR		option of ENVIRONMENT attribute
DIGITAL		option of ENVIRONMENT attribute
DIM		built-in function
DIRECT		file description attribute
DISCONNECT		statement
DISK		option of ENVIRONMENT attribute
DISPLAY		statement
DO		statement, option of GET and PUT statements
E		format item
EDIT		option of GET and PUT statements
ELSE		clause of IF and LOCK statements
ELSIZ		option of ENVIRONMENT attribute
END		statement
ENDFILE		condition
ENDPAGE		condition
ENTRY		attribute
ENVIRONMENT	ENV	attribute, option of RUN and TRANSFER TO statements
ERROR		condition
EVENT		attribute, option of READ, WRITE, REWRITE, DISPLAY, DELETE, and RUN statements
EVERY		built-in function, option of RUN statement
EXP		built-in function
EXTERNAL	EXT	attribute

Keyword	*Abbreviation*	*Use of Keyword*
F		format item, option of ENVIRONMENT attribute
FB		option of ENVIRONMENT attribute
FBS		option of ENVIRONMENT attribute
FILE		attribute, option of I/O statement
FINISH		condition
FIXED		attribute, built-in function
FIXEDOVERFLOW	FOFL	condition
FLOAT		attribute, built-in function
FORMAT		attribute, statement
FROM		option of WRITE and REWRITE statements
GAIN		option of ENVIRONMENT attribute
GET		statement
GO TO	GOTO	statement
HBOUND		built-in function
HIGH		built-in function
IF		statement
IGNORE		option of READ statement
INCLUDE		macro statement
INITIAL	INIT	attribute
INPUT		file description attribute, option of OPEN statement
INTERNAL	INT	attribute
INTO		option of READ statement
KEY		option of READ, DELETE, and REWRITE statements, condition
KEYED		attribute
KEYFROM		option of WRITE statement
KEYTO		option of READ statement
LABEL		attribute
LBOUND		built-in function
LENGTH		built-in function
LINE		format item, option of PUT statement
LINENO		built-in function
LINESIZE		option of OPEN statement
LIST		option of GET and PUT statements
LOCK		attribute, statement
LOG		built-in function
LOG2		built-in function
LOG10		built-in function
LOW		built-in function
MAIN		option of PROCEDURE statement

Keyword	*Abbreviation*	*Use of Keyword*
NOCONVERSION	NOCONV	condition prefix (disables CONVERSION)
NOFIXEDOVERFLOW	NOFOFL	condition prefix (disables FIXEDOVERFLOW)
NOOVERFLOW	NOOFL	condition prefix (disables OVERFLOW)
NOSIZE		condition prefix (disables SIZE)
NOSTRINGRANGE	NOSTRG	condition prefix (disables STRINGRANGE)
NOSTRINGSIZE	NOSTRZ	condition prefix (disables STRINGSIZE)
NOSUBSCRIPTRANGE	NOSUBRG	condition prefix (disables SUBSCRIPTRANGE)
NOUNDERFLOW	NOUFL	condition prefix (disables UNDERFLOW)
NOZERODIVIDE	NOZDIV	condition prefix (disables ZERODIVIDE)
NULL		built-in function
ON		statement
ONCHAR		built-in function, pseudo-variable
ONCODE		built-in function
ONCOUNT		built-in function
ONFILE		built-in function
ONKEY		built-in function
ONLOC		built-in function
ONSOURCE		built-in function, pseudo-variable
OPEN		statement
OPTIONS		option of PROCEDURE and BEGIN statements
OUTPUT		file description attribute, option of OPEN statement
OVERFLOW	OFL	condition
P		format item
PAGE		format item, option of PUT statement, macro statement
PAGESIZE		option of OPEN statement
PARTITION		option of ENVIRONMENT attribute
PENDING		condition
PIBIT		option of ENVIRONMENT attribute
POINTER	PTR	attribute
POST		statement
PRECISION	PREC	built-in function
PRINT		file description attribute
PRIORITY		built-in function, option of RUN and TRANSFER TO statements

Keyword	*Abbreviation*	*Use of Keyword*
PRIVATE		option of ENVIRONMENT attribute
PROCEDURE	PROC	statement
PROCESS		control statement
PROGRAM		attribute
PUT		statement
QPRTY		option of ENVIRONMENT attribute
R		format item, option of ENVIRONMENT attribute
READ		statement
RECORD		file description attribute, condition
RECSIZE		option of ENVIRONMENT attribute
RECURSIVE		option of PROCEDURE statement
REGIONAL		option of ENVIRONMENT attribute
REPEATS		option of OPTIONS attribute
REPLY		option of DISPLAY statement
RESTART		option of ENVIRONMENT attribute
RETURN		statement
RETURNS		attribute, option of PROCEDURE statement
REVERT		statement
REWRITE		statement
RUN		statement
S		option of ENVIRONMENT attribute
SEQUENTIAL	SEQL	file description attribute
SIGN		built-in function
SIGNAL		statement
SIN		built-in function
SIND		built-in function
SIZE		condition
SKIP		format item, option of GET and PUT statements, macro statement
SNAP		option of ON statement
SOME		built-in function
SOURCE		attribute
SQRT		built-in function
SSTNDX		option of ENVIRONMENT attribute
STACK		option of ENVIRONMENT attribute
STACKSIZE		option of PROCEDURE and BEGIN statements
STATIC		attribute

Keyword	*Abbreviation*	*Use of Keyword*
STATUS		built-in function, pseudo-variable, option of POST statement
STG		option of ENVIRONMENT attribute
STOP		statement
STREAM		attribute
STRING		option of GET and PUT statements
STRINGRANGE	STRG	condition
STRINGSIZE	STRZ	condition
SUBSCRIPTRANGE	SUBRG	condition
SUBSTR		built-in function, pseudo-variable
SYSIN		standard input file
SYSPRINT		standard output file
SYSTEM		statement, option of ON statement
TAN		built-in function
TAND		built-in function
TASK		option of PROCEDURE and STOP statements
THEN		clause of IF statement
TIME		built-in function
TO		clause of DO statement
TRANSFER TO		statement
TRANSIENT		attribute
TRANSMIT		condition
UNALIGNED	UNAL	attribute
UNDEFINEDFILE	UNDF	condition
UNDERFLOW	UFL	condition
UNLOCK		statement
UNSCHEDULE		statement
UNSPEC		built-in function, pseudo-variable
UPDATE		attribute
V		option of
VARYING	VAR	string attribute
VB		option of ENVIRONMENT attribute
VBS		option of ENVIRONMENT attribute
WAIT		statement
WHILE		clause of DO statement
WRAP		option of ENVIRONMENT attribute
WRITE		statement

Keyword	*Abbreviation*	*Use of Keyword*
X		format item
ZEROCOR		option of ENVIRONMENT attribute
ZERODIVIDE	ZDIV	condition ENVIRONMENT attribute

Appendix B. Comparison of Series/1 PL/I Subset to ANS PL/I

Comparison of Built-In Functions and Pseudo-Variables

Function	*Series/1*		*ANS PL/I*
ABS	YES		YES
ACOS	YES		YES
ADD	NO		YES
ADDR	YES		YES
AFTER	NO		YES
ALLOCATION	NO		YES
ASIN	YES		YES
ATAN	YES		YES
ATAND	YES		YES
ATANH	NO		YES
BEFORE	NO		YES
BINARY	YES	1	YES
BIT	YES		YES
BOOL	YES		YES
CEIL	NO		YES
CHAR	YES		YES
COLLATE	NO		YES
COMPLETION	YES		NO
COMPLEX	NO		YES
CONJG	NO		YES
COPY	YES	18	YES
COS	YES		YES
COSD	YES		YES
COSH	NO		YES
COUNT	YES		NO
DATE	YES		YES
DAYNO	YES		NO
DECAT	NO		YES
DECIMAL	YES	2	YES
DIM	YES		YES
DIVIDE	NO		YES
DOT	NO		YES
EMPTY	NO		YES
ERF	NO		YES
ERFC	NO		YES
EVERY	YES		YES
EXP	YES		YES
FIXED	YES		YES
FLOAT	YES		YES
FLOOR	NO		YES

Function	*Series/1*		*ANS PL/I*
HBOUND	YES		YES
HIGH	YES		YES
IMAG	NO		YES
IMAG PV	NO		YES
INDEX	NO		YES
LBOUND	YES		YES
LENGTH	YES		YES
LINENO	YES		YES
LOG	YES		YES
LOG2	YES		YES
LOG10	YES		YES
LOW	YES		YES
MAX	NO		YES
MIN	NO		YES
MOD	NO		YES
MULTIPLY	NO		YES
NULL	YES		YES
OFFSET	NO		YES
ONCHAR	YES		YES
ONCHAR PV	YES		YES
ONCOUNT	YES		NO
ONCODE	YES		YES
ONFIELD	NO	3	YES
ONFILE	YES		YES
ONKEY	YES		YES
ONLOC	YES		YES
ONSOURCE	YES		YES
ONSOURCE PV	YES		YES
PAGENO	NO		YES
PAGENO PV	NO		YES
PRECISION	YES	1, 2	YES
POINTER	NO		YES
PRIORITY	YES		NO
PROD	NO		YES
REAL	NO		YES
REAL PV	NO		YES
REVERSE	NO		YES
ROUND	NO		YES
SIGN	YES		YES
SIN	YES		YES
SIND	YES		YES
SINH	NO		YES
SOME	YES		YES
SQRT	YES		YES
STATUS	YES		NO
STATUS PV	YES		NO
STRING	NO		YES

Function	*Series/1*		*ANS PL/I*
STRING PV	NO		YES
SUBSTR	YES		YES
SUBSTR PV	YES		YES
SUBTRACT	NO		YES
SUM	NO		YES
TAN	YES		YES
TAND	YES		YES
TANH	NO		YES
TIME	YES		YES
TRANSLATE	NO		YES
TRUNC	NO		YES
UNSPEC	YES	20	YES
UNSPEC PV	YES	20	YES
VALID	NO		YES
VERIFY	NO		YES

Comparison of Attributes

Attribute	*Series/1*		*ANS PL/I*
ACTIVATION	YES	21	NO
AFTER	YES		NO
ALIGNED	YES		YES
AREA	NO		YES
AT	YES		NO
AUTOMATIC	YES	4	YES
BASED	YES	5	YES
BINARY	YES	1	YES
BIT	YES	6	YES
BUILTIN	YES		YES
CHARACTER	YES	7	YES
COMPLEX	NO		YES
CONDITION	YES		YES
CONNECTED	YES		NO
CONSTANT	NO		YES
CONTROLLED	NO		YES
DECIMAL	YES	2	YES
DEFINED			
—simple	NO		YES
—iSUB	NO		YES
—string overlay	(ADDR BIF)		YES
dimension	YES	4	YES
DIMENSION	NO		YES
DIRECT	YES		YES
ENTRY	YES		YES
ENVIRONMENT	YES		YES
EVENT	YES	21	NO
EXTERNAL	YES		YES
FILE	YES		YES
FIXED	YES		YES
FLOAT	YES	8	YES
FORMAT	YES	19	YES
GENERIC	NO		YES

Attribute	*Series/1*		*ANS PL/I*
INITIAL	YES	9	YES
INPUT	YES		YES
INTERNAL	YES		YES
KEYED	YES		YES
LABEL	YES		YES
length	YES		YES
LIKE	NO		YES
LOCAL	NO		YES
LOCK	YES		NO
MEMBER	NO		YES
NONVARYING	NO		YES
OFFSET	NO		YES
OPTIONS	NO		YES
OUTPUT	YES		YES
parameter	YES	10	YES
PARAMETER	NO		YES
PICTURE	NO		YES
POINTER	YES		YES
POSITION	NO		YES
precision	YES	1, 2	YES
PRECISION	NO		YES
PRINT	YES		YES
PROGRAM	YES		NO
REAL	NO		YES
RECORD	YES		YES
RETURNS	YES	12	YES
SEQUENTIAL	YES		YES
size	NO		YES
SOURCE	YES		NO
STATIC	YES		YES
STREAM	YES		YES
STRUCTURE	NO		YES
TRANSIENT	YES		NO
UNALIGNED	YES	12	YES
UPDATE	YES		YES
VARIABLE	NO		YES
VARYING	YES	6, 7	YES

Comparison of Statements and Options

Statement	*Series/1*		*ANS PL/I*	
ALLOCATE	NO		YES	
Assignment				
—element	YES		YES	
—array	YES		YES	
—structure	YES	13	YES	
—BY NAME	NO		YES	
BEGIN	YES		YES	
OPTIONS	YES		YES	
STACKSIZE	YES		YES	
CALL	YES	14	YES	
CLEAR	YES		NO	
CLOSE	YES		YES	
CONNECT	YES		NO	
DECLARE	YES		YES	
DEFAULT	NO		YES	
DELAY	YES		NO	
DELETE	YES		YES	
DISCONNECT	YES		NO	
DISPLAY	YES		NO	
DO				
—arithmetic	YES		YES	
—char ctl var	YES		YES	
—WHILE	YES		YES	
—REPEAT	NO		YES	
END	YES		YES	
ENTRY	NO		YES	
FORMAT	YES		YES	
FREE	NO		YES	
GET				
LIST	YES		YES	
DATA	NO	3	YES	
EDIT	YES		YES	
STRING	YES		YES	
expressions	YES	17	YES	
DO	YES		YES	
SKIP	YES		YES	
LINE	YES		YES	
PAGE	YES		YES	
COPY	NO		YES	
GO TO	YES		YES	
IF	YES		YES	
LOCATE	NO		YES	
LOCK	YES		NO	
null	YES		YES	
ON (with SNAP)	YES		YES	
OPEN	YES	15	YES	
POST	YES		NO	
PROCEDURE	YES	23	YES	23
OPTIONS	YES		YES	
MAIN	YES		NO	
STACKSIZE	YES		NO	
TASK	YES		NO	

Statement	*Series/1*	*ANS PL/I*
REPEATS	YES	NO
RECURSIVE	YES	YES
RETURNS	YES	YES
PUT	SEE GET	
READ		
INTO	YES	YES
SET	NO	YES
IGNORE	YES	YES
KEY	YES	YES
KEYTO	YES	YES
EVENT	YES	NO
RETURN	YES 16	YES
—aggregate	NO	YES
—extents	NO	YES
—entry	NO	YES
REVERT	YES	YES
REWRITE		
FROM	YES	YES
KEY	YES	YES
EVENT	YES	NO
buffer	NO	YES
RUN	YES	NO
SIGNAL	YES	YES
STOP	YES	YES
SYSTEM	YES	YES
TRANSFER TO	YES	NO
UNLOCK	YES	NO
UNSCHEDULE	YES	NO
WAIT		
—I/O event	YES	NO
—time limit	YES	NO
—task event	YES	NO
—process interrupt	YES	NO
WRITE		
FROM	YES	YES
KEYFROM	YES	YES
EVENT	YES	NO

Comparison of ON Conditions

Condition	*Series/1*	*ANS PL/I*
AREA	NO	YES
CHECK	NO 3	YES
CONDITION	YES	YES
CONVERSION	YES	YES
ENDFILE	YES	YES
ENDPAGE	YES	YES
ERROR	YES	YES
FINISH	YES	YES
FIXEDOVERFLOW	YES	YES
KEY	YES	YES
NAME	NO 3	YES
OVERFLOW	YES	YES
PENDING	YES	NO

Condition	*Series/1*	*ANS PL/I*
RECORD	YES	YES
SIZE	YES	YES
STORAGE	NO	YES
STRINGRANGE	YES	YES
STRINGSIZE	YES	YES
SUBSCRIPTRANGE	YES	YES
TRANSMIT	YES	YES
UNDEFINEDFILE	YES	YES
UNDERFLOW	YES	YES
ZERODIVIDE	YES	YES

Comparison of Format Items

Format Item	*Series/1*	*ANS PL/I*
A	YES	YES
B	YES	YES
C	NO	YES
COLUMN	YES	YES
E	YES	YES
expressions	NO	YES
F	YES	YES
LINE	YES	YES
P	YES 22	YES
PAGE	YES	YES
R	YES	YES
SKIP	YES	YES
TAB	NO	YES
X	YES	YES

Comparison of Macro Facility

Statement	*Series/1*	*ANS PL/I*
INCLUDE	YES	YES
PAGE	YES	NO
PROCESS	YES	NO
SKIP	YES	NO

Comparison of ENVIRONMENT Options

These options are allowed by ANS PL/I, but are implementation defined.

Option	*Series/1*
ANALOG	YES 24
ASCII	YES 24
BLKSIZE	YES 24
BUFFERS	YES 24
CONSECUTIVE	YES 24
CTLASA	YES 24
DEVNBR	YES 24
DIGITAL	YES 24
DISK	YES 24
ELSIZ	YES 24
F	YES 24
FB	YES 24
FBS	YES 24
GAIN	YES 24

Option	*Series/1*	
PARTITION	YES	24
PIBIT	YES	24
PRIVATE	YES	24
PROGRAM	YES	24
QPRTY	YES	24
R	YES	24
RECSIZE	YES	24
REGIONAL	YES	24
RESTART	YES	24
S	YES	24
SSTNDX	YES	24
STACK	YES	24
STG	YES	24
V	YES	24
VB	YES	24
VBS	YES	24
WRAP	YES	24
ZEROCOR	YES	24

Notes:

1. Binary fixed precision P, where P = 15 or P = 31, scale factor q must be 0.
2. Decimal fixed precision P, where P ≤ 15, scale factor q must be ≤p.
3. Data-directed I/O is not supported in this subset because of extensive use of address space. This implies that CHECK is not supported.
4. No adjustable string lengths or array extents.
5. No REFER option or adjustable extents or lengths.
6. Maximum length is 255 bits.
7. Maximum length is 255 characters.
8. No support for extended precision floating point.
9. No INITIAL values allowed for AUTOMATIC data.
10. String length must be a constant, constant extents for array of structures, simple array can have asterisks.
11. Cannot return an array, required for the declaration of external function reference.
12. Arithmetic items may not be unaligned.
13. Assignment only; only like attributes.
14. Simple scalar expressions as arguments.
15. Only INPUT and OUTPUT options permitted.
16. Returned string must be constant length.
17. Simple scalar expression permitted in PUT.
18. COPY built-in function replaces the IBM REPEAT function.
19. Remote FORMATS must be in same block.
20. Scalar only, not array or structure.
21. Cannot be AUTOMATIC.
22. No floating point Pictures.
23. No parameters are allowed on MAIN procedure.
24. See *Language Reference Manual* for detailed syntax.

Index

PL/I:
Introduction
GC34-0084-0

READER'S
COMMENT
FORM

YOUR COMMENTS, PLEASE . . .

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. All comments and suggestions become the property of IBM.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or to the IBM branch office serving your locality.

Corrections or clarifications needed:

Page Comment

Cut or Fold Along Line

What is your occupation?_____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate your name and address in the space below if you wish a reply.

_____

_____

_____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GC34-0084-0

Your comments, please . . .

This manual is part of a library that serves as a reference source for IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold

Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

IBM Corporation
Systems Publications, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold

Fold

International Business Machines Corporation
General Systems Division
5775D Glenridge Drive N.E.
P.O. Box 2150, Atlanta, Georgia 30301
(U.S.A. only)

Cut Along Line

IBM Series/1 PL/1: Introduction Printed in U.S.A. GC34-0084-0

International Business Machines Corporation

General Systems Division
5775D Glenridge Drive N.E.
P. O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

IBM Series/1 PL/I: Introduction Printed in U.S.A. GC34-0084-0

GC34-0084-0