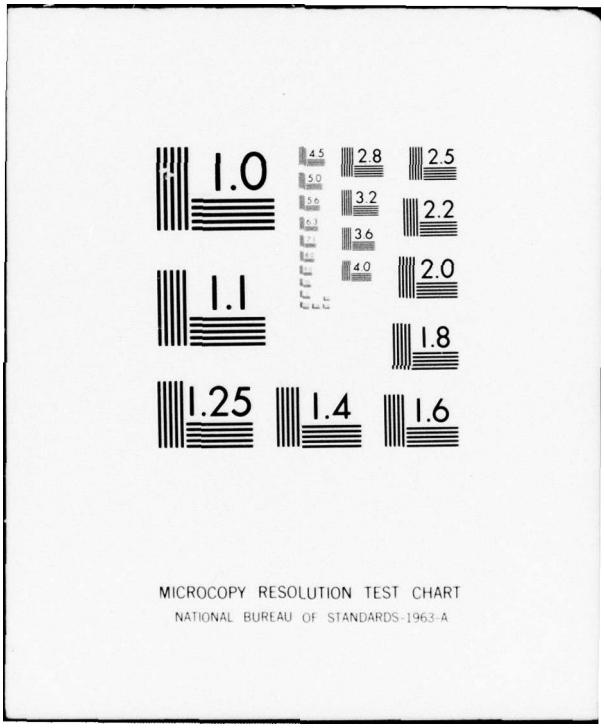
AD-A039 076 FOREIGN TECHNOLOGY DIV WRIGHT-PATTERSON AFB OHIO F/G 9/2
THE PL/1 COMPILER FOR THE ROBOTRON 300, (U)
NOV 76 W SCHILLER
UNCLASSIFIED FTD-ID(RS)I-1369-76 NL

| OF |
AD
A039076

END

DATE
FILMED
5-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A039076

FTD-ID(RS)I-1369-76

FOREIGN TECHNOLOGY DIVISION

THE PL/1 COMPILER FOR THE ROBOTRON 300

by

W. Schiller

DDC
RECEIVED
MAY 9 1977
D

Approved for public release;
distribution unlimited.

EDITED TRANSLATION

FTD-ID(RS)I-1369-76 11 November 1976

FTD-76-C-001143

CSI76054962

THE PL/1 COMPILER FOR THE ROBOTRON 300

By: W. Schiller

English pages: 25

Source: Rechentechnik Datenverarbeitung, Vol. 12, Nr. 8, September 1975, pp 19-26

Country of origin: East Germany
Translated by: LINGUISTIC SYSTEMS INC.
F33657-76-D-0389
Charles Funk
Requester: FTD/ETCK
Approved for public release; distribution unlimited.

THIS TRANSLATION IS A RENDITION OF THE ORIGINAL FOREIGN TEXT WITHOUT ANY ANALYTICAL OR EDITORIAL COMMENT. STATEMENTS OR THEORIES ADVOCATED OR IMPLIED ARE THOSE OF THE SOURCE AND DO NOT NECESSARILY REFLECT THE POSITION OR OPINION OF THE FOREIGN TECHNOLOGY DIVISION.

PREPARED BY:

TRANSLATION DIVISION
FOREIGN TECHNOLOGY DIVISION
WP-AFB, OHIO.

THE PL/1 COMPILER FOR THE ROBOTRON 300

W. Schiller

The PL/1 compiler conceived for the R300 data processing system is described herein. The foundations for its development are related, and the language range realized for the present version is indicated.

1 Why PL/1 for the R300?

Now that restricting influences have been overcome, a PL/1 compiler for the Robotron 300 data processing system is a possibility. The question arises, whether it is reasonable to develop a PL/1 compiler for a second generation installation, and furthermore whether its time has already passed.

To question the reasonableness is not entirely justified. A language like PL/1 is basically always reasonable for any computer with a capability to implement a PL/1 language range which can support a relatively wide scope of applications. The question must be more properly whether the R300 is suitable to implement PL/1 reasonably.

The answer to this question devolves from the existence of the compiler and its implementation in industrial practice.

The question of timing remains. It is true that the compiler has arrived somewhat late, however not too late. The reason is, in part, the language range implemented. Although the R300 PL/1 compiler is a subset compiler, it still includes significantly more possibilities than the existing subset compiler for the IBM 360 or Robotron 21 installations. The entire listing and its control, as well as error handling were completely adapted to OS-PL/1. PL/1 programs can be partially written and processed in the OS form. This applies, for example, to file mapping, sequential input and output, all error handling

during the object run (implementation of the ON instruction as in full PL/1), use of the INITIAL attribute, dynamic field mapping, etc. Thus means are placed in the hands of the programming user and student to become prepared to use the full PL/1 compiler available in the ES-1040 installation.

By means of the present implementation, it is possible for firms possessing an R300 and wishing to convert to an ESER series installation to rewrite parts of their established projects in PL/1, test them on their own installation, perhaps with modifications, and then on the day if installation of the new computer type to reinterpret the programs prepared in PL/1 with the PL/1 compiler of the new system and then make further modifications. No changes then need to be made in the text of the PL/1 source program. Many firms have already undergone an unbroken transition from an R300 type computer to an ES-1040 system through this capability.

An important point for the development of the PL/1 compiler for the Robotron 300 was the availability of a suitable compiler enabling students to receive advance training in the use of third generation computer systems on the numerous R300 systems installed in technical and secondary schools.

2 Language range implemented

2.1 Introduction

As already indicated in the introduction of section 1, the PL/1 compiler for the R300 was developed to make possible for R300 users a transition to a system in the ES-1040 series which is as free of complications as possible, and can occur in-house without an interruption in operations. The scope of the PL/1 implementation also results from this same consideration.

With regard to the OS/ES-PL/1 compiler expected, the actual compilation phase is designed and effected nearly analogously (with respect to the results) to that of the OS/ES compiler. This applies primarily to the error handling during compilation. Far-reaching corrections of incorrect PL/1 source programs, as sensible as possible, are undertaken by the compiler. The nature of the error correction corresponds to the seriousness of the error.

2.2 Rules for writing a PL/1 program

For formulation of a PL/1 program, only one character set can be used. It is composed of 57 characters. Compared to the ususl 60 character set used in PL/1 for other computer types, the dollar sign ($), the commercial "at" sign (@), and the underscore character (_) are missing in the "reduced 60 character set" for the R300. The characters not contained in the R300 machine code, &, |, and ¬ , are replaced by the characters [, !, and] respectively.

The 48 character set can not be employed. There are no further restrictions regarding the choice of designators.

The source text of the PL/1 program is to be prepared for an 80 column card format, within which:
Column 1 must contain a null character, or one of the four characters A, B, C or 4. These last four characters are carriage control information evaluated during echo printing of the PL/1 source text. They have the effect that the source text information is separated by one, two or three blank lines, or is printed on a new page.
Columns 2-72 contain the PL/1 source text.
Columns 73-80 can contain any desired information. They are ignored by the compiler.

All intervening spaces are to be punched with null characters. The final information contained on a punched card is to be terminated with an x-shift. The source text can also be punched on paper tape. The same punching specifications apply in this case as for cards. It is, of course, necessary that the entire "card file" be enclosed between a prefix and a suffix. The prefix consists of the two words, AAA and the 12 character name PL1QUELLTEXT (PL/1 source text), separated by a tabulation or by one null character. The suffix consists of the word ZZZ. Each paper tape line is to be terminated with a carriage return and line feed.

For R300-Pl/1, all codewords may be used in abbreviated form as well. Designators can be at most 31 characters long; external names are restricted to 7 characters.

2.3 Data types

For R300-PL/1, all real arithmetic data are permissible. Arithmetic constants can be entered as decimal or binary numbers in fixed or floating point representation. Attributes are derived by the compiler from the representation of the constants and associated with them.

Arithmetic variables are declared by adding to the name the arithmetic attributes for the kind of point (FIXED or FLOAT), number base (DECIMAL or BINARY), and precision (w[,d]). If the point is not specified, FIXED is assumed; if no number base is declared, DECIMAL is added, and if no precision is declared, a standard precision is used. This precision depends on the number base and point type attributes. The following summary shows the maximum and standard precisions for R300-PL/1:

Source attribute	Target attribute	Precision
BIN FIXED (p,q)	DEC FIXED	(1+INT(p/3.32,INT(q/3.32))

Number base	Point type	max. Prec.	Standard
DECIMAL	FIXED	(15,15)	(5,0)
DECIMAL	FLOAT	(15)	(8)
BINARY	FIXED	(31,31)	(15,0)
BINARY	FLOAT	(49)	(21)

The internal representation of binary fixed point data as well as its processing occurs in decimal. All binary fixed point constants contained in a PL/1 program are converted to their decimal equivalents during compilation and stored as decimals. Declared binary fixed point variables are correspondingly converted to decimal form and processed and processed in that form.

Conversion of procision occurs according to the following rule: The internal processing of fixed point quantities is done by means of fixed point arithmetic. For representation of a fixed point quantity, only as many digits are used as are specified in the precision.

Floating point quantities can be placed in short or long representation. The short representation includes 10 characters (8 place mantissa and 2 place exponent); the long floating point representation includes 17 characters (15 place mantissa and 2 place exponent). The internal coded arithmetic representation of binary floating point data is the unnormalized decimal floating point form. If the declared precision is less than or equal to 26, the single floating point format is used; if it is greater than 26, the double floating point format is used. For decimal floating point data, if the precision $w \leq 8$, the short floating point format is used; if $w > 8$, the double floating point format.

Arithmetic data belongs to the class of problem data. Also within the class of problem data is string data, within which character strings and bit strings are to be distinguished. A character string is a sequence of arbitrary characters representable in the R300 (excluding identifiers and extraneous word marks). A character string of length w occupies w characters in memory.

The R300 uses no binary arithmetic. Bit strings in internal representation are symbols consisting of sequences of the digits 0 and 1. A bit string of length w occupies w characters in memory or in an external medium.

Also among arithmetic data are numeric strings. They can occur as variables, and must be declared with the PICTURE attribute. In general they contain processing information in addition to their numerical value. In the R300 implementation, the numerical value of a PICTURE variable is treated as a decimal variable. Only when the PICTURE variable is used as a string in some relationship is there any processing of the actual numerical value according to the programmed mapping instructions, and the variable is processed as a character string.

All problem data types may be converted into one another, as long as they conform to the rules of conversion. It is thus also possible to convert arithmetic variables to character strings and vice versa.

Should designators arise in a PL/1 source text, which are declared neither explicitly nor from context, they stand as implicitly declared. The standard attributes DECIMAL FLOAT (8) are assigned to them, insofar as the designator does not begin with the letters I, J, K, L, M or N, in which case the standard attributes BINARY FIXED (15,0) are assigned.

For arithmetic and string variables, R300-PL/1 permits starting values to be assigned to them in their declaration with the INITIAL attribute. Examples:

```
DCL PI DEC FIXED(5,4) INIT (3.1415),
STRING CHAR(20) INIT ('STARTINGVALUE'),
NULL1 CHAR (25) INIT ((25)' '),
NULL2 CHAR (25) INIT (' ');
```

The two variables NULL1 and NULL2 are both assigned 25 null characters. In the present initial version of the compiler, the use of the INITIAL attribute is restricted to only simple AUTOMATIC variables.

2.4 Data element grouping

In PL/1 there are two possible ways to assemble individual data elements into larger units:

1. variables of a single type into fields, and
2. variables of different types into structures.

2.4.1 Fields

In the present version of the compiler, the number of dimensions is limited to 5. Expressions for lower as well as upper boundaries are permitted, so that dynamic field mapping is possible. Individual field elements are referenced by indexed variables. The indices can be arbitrary expressions. The declaration of converging boundaries, i.e. the use of partial fields, is presently not possible. The use of field expressions is not permitted in the first version, nor is the use of the INITIAL attribute for initialization of field elements. STATIC fields may not be used. The use of field names in input and output lists is permitted, however.

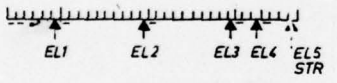
2.4.2 Structures

A grouping of variables of arbitrary types and fields is possible by means of structures. The depth of structuring is limited to twelve levels, and the maximum number of stages is 255. All structures are stored internally as a line, in that they are terminated with a line mark. Elements of structures are stored internally wordwise from left to right.

Example:

```
DCL
1 STR,
2 EL1 DEC FIXED (5,2)
2 EL2 FLOAT DEC (6)
2 EL3 CHAR (10),
2 EL4 BIT (3)
2 EL5 POINTER;
```

Storage requirements for this structure: 33 characters including line mark.



This type of storage is to be observed in the use of existing magnetic tape data sequences and their description as a structure. The usual first word of a line for the R300 user must also be declared as the last element in the structure describing the line.

The last element of a structure must not be a field.

2.5 Combination of problem data

All problem data types can be combined with each other in expressions. An arithmetic expression is formed when an arithmetic prefix operator (+ or -) is affixed to a variable,

or when two variables are combined with one of the five arithmetic operators +, -, *, /, **. Further, the formation of comparitive expressions is possible by means of the eight comparitive operators (>, >=, <, <=, =,]>,]<,]=).

The evaluation of a comparitive expression produces a bit string of length 1 in the internal representation.

The logical operators available are:] (negation), [(conjunction) and ! (disjunction). The concatenation operator can be represented by two "or" signs (!!).

The combination of different data types automatically effects conversion of the operands involved. The sequence of processing of an expression with many operands is controlled according to the familiar precedence rules for operators. The programmer can make alterations to the sequence through the use of parentheses.

2.6 Instructions

Any executable instruction or any block can be preceded by a condition prefix list. The computation conditions:

CONVERSION
FIXED OVERFLOW
OVERFLOW
SIZE
STRINGRANGE
SUBSCRIPTRANGE
UNDERFLOW
ZERODIVIDE

may be used as prefix elements. In contrast to PL/1, the SUBSCRIPTRANGE condition is in effect as a standard.

The following instructions are implemented:
multiple result instruction, DO, GO, TO, IF, PROCEDURE, BEGIN, END, RETURN, STOP, CALL, DECLARE, GET, PUT, DISPLAY, READ, WRITE, LOCATE, ON, REVERT, SIGNAL, OPEN, CLOSE, and FORMAT.

2.6.1 The ON instruction

With the ON instruction the PL/1 programmer has access to a means to recognize errors arising in the objec run, and to handle them accordingly. Of the computation conditions named in 2.6, the conditions SIZE and STRINGRANGE are not in effect at the start of a program, but the other six conditions are effective. Different effects can be achieved with condition prefixes before instrictions. For example:

```
(SIZE, STRINGRANGE):
(NOZERODIVIDE):
BEG: BEGIN;
   .
   .
   .
```

means that during the operations within the BEGIN block BEG, insofar as nothing to the contrary is declared in a superior block, all computation conditions except ZERODIVIDE are in effect. At the same time the notation cited here shows that in contrast to subset PL/1, more condition prefixes can precede an instruction.

Besides the computation conditions there are also available the permanently effective conditions

```
ENDFILE (file name)
ENDPAGE (file name)
ERROR
KEY (file name)
RECORD (file name)
UNDEFINEDFILE (file name)
```

which, up to ERROR, belong to the class of I/O conditions. For the total of 14 conditions available in R300-PL/1, handlers can be established in an ON instriction. As handlers, all instructions except the DO instruction and the PROCEDURE instruction are permissible. The most frequent form of the handler will probably be the BEGIN block, since it is most convenient and clearest. If during run time, the specified condition arises within the domain of an ON instruction, an automatic branching to the measure established in the ON instruction occurs. After conclusion of the measure, control returns to the faulty instruction or the following instruction. An ON instruction in R300-PL/1, in contrast to full PL/1, requires no block of its own.

If a condition arises for which no handler has been established by the programmer, the default handler takes effect. This consists of printing of an error message, which includes the cause of the interruption and, when given in the EXEC instruction, the instruction number at which the error occurred, as well as information on all blocks active at the time of the interruption. The program is then terminated.

With the REVERT instruction, the conditions established in ON instructions within the same block as the REVERT instruction can be cancelled. In other words, execution of a REVERT instruction places again into effect those measures which were effective at the time of activation of the block. With the SIGNAL instruction, the occurrence of an error can be simulated.

2.6.2 I/O instructions

Input and putput are defined as the transfer of data from an external storage medium to primary memory, or from primary memory to an external storage medium. The external medium to or from which transfer is to occur must be declared as a file.

The file declaration must be made in a DECLARE instruction, in which certain file attributes are associated with the name of the file. In contrast to subset PL/1, the codeword FILE need not always be given. The type of transfer is to be indicated with the codeword STREAM or RECORD, depending on whether there is to be transfer of the data in the file as a series or in sets. Furthermore it must in general be specified whether the file is provided for input or output, or whether it is to be accessed sequentially or directly. In the present version, only files with the SEQUENTIAL attribute are permissible. All physical characteristics of the file, such as blicking, I/O device, characteristic sets of MT files, location of the index of INDEXED files, etc. are to be specified in the DD instruction associated with the file, outside of the PL/1 program. The dd name, i.e. the file name, must therein be identical with the name established in the DECLARE instruction, since the TITLE option has not been implemented in R300-PL/1. In the ENVIRONMENT attribute, only the method of organization of the data sequence can be stated. That is, only the codewords CONSECUTIVE or INDEXED are permitted. If no ENV attribute is given, CONSECUTIVE organization is assumed as a standard.

Before there can be access to a file from the program, it must first be opened. Opening can be explicit, by means of an OPEN instruction for the corresponding file, or implicit with the first access of a previously unopened file. Implicit opening is also possible for data transfer in sets, in contrast to subset PL/1. In the case of explicit opening of a file, there is a consolidation of the attributes specified in the OPEN instruction with the attributes established in the corresponding DECLARE instruction. If a contradiction thus arises, the UNDEFINEDFILE condition takes effect. For PRINT files, the number of lines per page can be established in the OPEN instruction with PAGESIZE (n) and the number of characters per

line by means of LINESIZE (n), and thus the format of a printed page can be specified.

In the present first version of the compiler, the declaration of a DCB parameter in the DD instruction is only possible for magnetic tape files. The record type (subparameter RECFM = F) can here only be F (i.e. fixed length records). The record length (specified in the LRECL subparameter of the DCB parameter) is arbitrary for files provided for data transfer in sets. For the files provided for sequential transfer to MT, only a specification of LRECL = 80, BLKSIZE = 800 is permissible. The instructions for transfer in sets always refer to a record. Records are terminated with a record mark in R300-PL/1.

By means of the instructions
READ FILE (MT) INTO (RECORD); or
WRITE FILE (OUT) FROM (VAR);
in the first case a record including record mark is transferred from the input region of the MT file into the memory region RECORD, and in the second case a complete record, including record mark is transferred from the memory region VAR into the output region of the OUT file. This means that the variables in the INTO or FROM option must be structures. Other variables inevitably lead to errors. If only one field or a single variable is to be transferred, it must formally be embedded in a structure for R300-PL/1 (so that the compiler can produce the record mark important for transfer by records.)

The READ instruction with the SET option and the LOCAT instruction enable processing in the buffer for fixed consecutive files.

Of the forms of sequential transfer provided in PL/1, LIST and EDIT controlled transfer have been implemented in R300-PL/1.

Sequential transfer is characterized by the fact that the data is viewed as an unbroken string of characters at the external medium.

In the case of LIST controlled transfer, the data at the external medium are valid arithmetic or string constants. For the case of input, the individual constants are to be separated by one or more null characters and/or a comma. On output, a null character is always used as a separator. In printed output, the string boundary characters (apostrophes) of character strings are not printed. The elements to be printed are in this case also output in column positions determined by the implementation. For R300-PL/1 these positions are 1, 25, 49, 73, 97, and 121.

In the case of EDIT controlled transfer, the number of characters to be drawn from the input medium or delivered to the output medium is to be declared by a format element in a format list. Here data and control format elements are to be distinguished. Possible data format elements are:

F(w[,d[,p]])	for decimal fixed point numbers
E(w,d[,s])	for decimal floating point numbers
A[(w)]	for character strings
B[(w)]	for bit strings.

The P format element has not been implemented in the present first version.

The control format elements are:

X (w)	for spacing
SKIP (w)	for relative line advance
LINE (w)	for absolute line advance
COLUMN (w)	for positioning symbols in a record
PAGE	for page advance
R (mark)	for reference to a remote format list.

The control format elements PAGE and LINE (w) can only be used for STREAM OUTPUT PRINT files. All other format elements can be used for any STREAM file.

PAGE, SKIP [(w)] and LINE (w) can also be specified as options in the PUT instruction. If no FILE option with file specification is present in a GET or PUT instruction, in the case of input the standard input file SYSIN is assumed, and in the case of output, the standard output file SYSPRINT. The standard input file is connected to punched card input by the LSE channel at IP1, the standard output file to the printer by OP1.

PUT LIST (U,V,W) SKIP (7); ; means that the values of the variables U, V, W will be printed via the standard output file SYSPRINT. Six blank lines are to be inserted.

If, instead of the FILE option, the STRING option is used in a GET or PUT instruction, the characters are drawn from or inserted in the character string specified after the codeword STRING.

2.6.3 PROCEDURE and CALL instruction

A block is a closed sequence of instructions introduced with a BEGIN or PROCEDURE instruction, and terminated with a corresponding END instruction. Blocks determine the domain of names, and control the storage allocation for variables. Entry into a block can occur only through the BEGIN or PROCEDURE instruction. For the present first version, the ENTRY instruction is not permitted as a secondary entry point. BEGIN and PROCEDURE blocks differ with regard to activation. A BEGIN instruction can be labelled. It can then be reached with a GOTO instruction. A PROCEDURE block can only be reached with a CALL instruction, unless by specification of the MAIN option

in the PROCEDURE instruction it is elevated to a main procedure. This can then be activated only through the operating system. A PROCEDURE instruction can contain several entry names.

Activated BEGIN and PROCEDURE blocks are terminated upon reaching their END instruction or by a jump to a superior block. Upon encountering a STOP instruction, the entire program is terminated. A PROCEDURE block can also be terminated by a RETURN instruction. END and RETURN instructions cause program control to return to the instruction following the calling point.

Blocks can be nested, in which case the outermost block must be a PROCEDURE block. It is called an external procedure. The depth of nesting for the R300-PL/1 compiler is limited to nine. A name declared in a DECLARE instruction in a block is valid in the entire block, including all subordinate blocks, as long as the same name is not declared again in one of the subordinate blocks.

Data can be transmitted between a CALL instruction and the PROCEDURE block to be called by means of the argument-parameter relation. In the PROCEDURE instruction, a parenthesized list of variable names, the parameter list, is then to be given. These variable names are assigned standard attributes corresponding to the initial letter of the designators, insofar as the parameters were not also declared in a DECLARE instruction with different attributes. For the present version it is reasonable (though not absolutely necessary) to declare all parameters. At most 29 elements can be specified in the parameter list. In the CALL instructions the parameters can be assigned names or expressions in corresponding sequence. The assignment is performed according to address; that is, no storage is allocated for the parameters; they use the storage space pf the arguments. Since in the transmission there is no

conversion of type of the argument to that of the corresponding parameter (it is in this version impossible to specify the desired attributes of the parameters in a parameter attribute declaration of an ENTRY attribute declaration otherwise necessary for this purpose), the programmer must take care that the linking conditions with respect to types are satisfied. In the present first version, it is not possible to transmit structures with the parameter-argument relation.

If the procedure name arises within an expression, the procedure is employed as a function procedure; that is, it computes a value which is inserted in place of the function name. The value to be computed is returned by means of the RETURN instruction. The attributes of the value to be returned must be specified in the RETURNS option of the PROCEDURE instruction, are the attributes of the returned value determined in standard fashion from the initial letter of the ENTRY name.* The calling procedure must of course deal with the same determination of attributes in the case of declaration of the entry name with the RETURNS attribute.

The syntactic definition of the declaration of a procedure to be employed as a function is as follows:

entry name: [entry name:] . . .
PROCEDURE [(par],par]: . .)]
[RETURNS (attribute)]:

It is to be observed that, in contrast to subset PL/1, the attribute declaration for the returned value must follow the codeword RETURNS in parentheses.

---

*This sentence was anomalous in the original text, and has been translated as it appeared. At least one clause seems to have been omitted in the source.

In the first compiler version described herein, external procedures are initially not permitted. The insertion of assembler code is also not allowed.

2.6.4 Standard functions

All mathematical standard functions in R300-PL/1 can contain as arguments only simple variables or simple expressions, and field names or field expressions are not allowed.

The following standard functions are components of the compiler:

1. All standard mathematical functions
2. The arithmetic standard functions:

ABS
MAX
MIN

As a deviation from PL/1, in the case of the functions MAX and MIN by specification of a field name the largest or smallest element of the field is determined.

3. The string functions:

BIT
CHAR
HIGH
INDEX
LENGTH
LOW
REPEAT
SUBSTR
TRANSLATE
VERIFY

4. The standard functions for field arguments:

ALL
ANY
DIM

HBOUND
LBOUND
PROD
SUM
The standard function DATE can presently be used only in output data lists.

3 The translation process

The PL/1 compiler is a multi-stage and multi-step translator. It consists of 16 program sections executed in sequence. In the first step, the PL/1 source program is converted to a macrp-mops-assembler program. Subsequently a special assembler translator undertakes further processing.

As a result of the first step, the following listing is obtained:

First, the PL/1 source program is printed. The instructions are given in their original form, but are provided with numbering to the left (instruction numbers) and, as long as NONEST is not specified in the EXEC card, with the block level number and the depth of DO group nesting. The block level number indicates on which block level the corresponding instruction is located. The DO group nesting number similarly indicates on which level the instruction is located within a DO group. The depth of DO group nesting is limited to 10 for the present version, the depth of block nesting to 9.

Following the source text listing is the listing of attributes and references. Here all explicitly and implicitly declared quantities are listed in alphabetic order together with their complete quantity of attributes complemented by the compiler. To the left of the designator, in the case of an

explicit declaration, is the number of the instruction in which the designator was declared. Below the attribute set belonging to a designator, in increasing order, are the numbers of the instructions in which the designator is referenced.

Following this list there generally appears the list of error messages. This list is divided in four classes, according to the severity of the error. The sequence is:
Terminal errors
Severe errors
Errors
Warnings.
Within each error class, the errors are sorted in increasing instruction number order. This eases the programer's search for the errors. This sorting is the only difference in the listing compared to those produced in full PL/1 (OS/ES).

If the programmer has elected to specify LIST in the PARM parameter of the EXEC card, he receives a record of the assembler code produced with the associated object code produced with the associated object code. The finished translated program is temporarily located on MT5 and can be loaded from there into primary memory for running via MONS.

4 Example

To give a view of the listing described, we wish to undertake a simple example program. It was in part extracted from the "Programmer's Handbook of the IBM 360" by C.B.Germain.

4.1 Problem definition

A simple bill writer will be undertaken. Two card types will be used for this, address cards and article cards.

1. Subdivision of the address cards (card type 01)

Columns 1 to 2: card type
Columns 3 to 8: form of address
Columns 9 to 20: title
Columns 21 to 40: name
Columns 41 to 45: postal zone number
Columns 46 to 60: town
Columns 61 to 80: street address

2. Subdivision of the article cards (card type 11)

Columns 1 to 2: card type
Columns 3 to 6: article number
Columns 7 to 26: article designation
Columns 27 to 32: quantity ordered (maximum 3 decimal places)
Columns 33 to 36: unit quantity
Columns 37 to 41: price of unit quantity (with 2 decimal places)
Columns 42 to 80: other information

For processing, a card deck consisting of address cards and article cards is used, and a number of article cards follow each address card. Accurate classification of the card deck will not be observed.

Bills are to be written on the printer.

3. The billing date is to be printed in columns 58 to 65 of the first printed line.

From line 3, beginning with print column 3, the address of the customer is to be printed.

Beginning with line 15, the individual billing lines are to be printed. The following line subdivision is to be maintained:

Position 3 to 22: article designation
Position 28 to 34: quantity
Position 36 to 39: unit quantity
Position 42 to 47: individual price of unit quantity
Position 58 to 67: total price

On the 59th line the billing amount is to be printed beginning in position 55.

For certain articles of the collection, care must be taken in the billing, that bills are not written for larger qyantities than are available in the inventory.

To be able to conduct such a control, at the outset the article numbers and the corresponding inventories in support of this control are to be read in.

If, during processing, it is determined that the quantity ordered cannot be delivered in full, in the corresponding billing line, instead of the total price, the information SOLD OUT is to be printed (positions 58-67).

At the end of the program, a list is to be printed, in which are indicated the articles to be verified, the original and the new inventory quantities.

4.2 Solution

The program was punched on paper tape. Figures 1, 2, and 3 show the listing and an extract from the assembler code produced. Figures 4 and 5 show the result of a computation.

To use as little storage as necessary, the fields for article numbers, old and new inventories (TAB#ART#NO, TAB#INVEN,

TAB#INVEN#NEW) are declared as dynamic fields. For that, the number of articles to be verified was used as the upper limit. In the main procedure, the number of articles whose inventory is to be controlled is read in under LIST control. The fields to be constructed thereafter must naturally be declared in a subordinate blick, which we have formulated as a BEGIN block.

The program to print the address heading was formulated as a subprogram with general validity. It demonstrates very clearly the possibilities offered by a language like PL/1 for the purposes of text processing. The form of the SUBSTR standard function used here is, of course, not possible in subset PL/1, since the third argument there must always represent a decimal integer. A portion of the EDIT controlled output used here could also not be used in subset PL/1, because there the specifications of the format elements can also only be whole numbers; that is, specifications of the form LINE (M) and COL (N + 6) or the like are not possible.

The PL/1 compiler in the version described herein is available for sale to the user. Inquiries should be addressed to the Ingenieurhochschule Dresden, Hans-Grundigstrasse 25.

```
R300 - PL/1 (VERSION 1.4)   PROCEDURE 01   RESR  THRUSDAY, 8 MAY 75

ANW[EISUNG] STEP SCHA
(?STATEMENT?)

          /+                SAMPLE PROGRAM          +/  00000006
          /+++++++++++++++++++++++++++++++++++++++++++/  00000007
                            /+ ZWICKAU, DEN 4.9.74  +/  00000008
                            /+ --------------------  +/  00000009
          /+-------------------------------------  +/  00000010
  1       RESR:  PROC OPTIONS(MAIN):                      00000011
                 /+---------------+/                      00000012
                 /+DECLARATIONS   +/                      00000013
                 /+---------------+/                      00000014
                 /+DECLARATION OF ADDRESS SETS            00000015
  2  1    /+     /+DCL 1 ADDRESSES                        00000016
                       2 ADDRESS CHAR(6)                  00000017
                       2 TITLE CHAR(12)                   00000018
                       2 NAME CHAR(20)                    00000019
                       2 ZIP CODE CHAR(5)                 00000020
                       2 LOCALITY CHARACTER(15)           00000021
                       2 APARTMENT CHAR(20)               00000022
                 / ------------- +/                       00000023

                 /+DECLARATION OF ARTICLE VARIABLES+/     00000024
  3  1           DCL ART*NR FIXED(4)                      00000025
                     ART*DESIGNATION CHAR(20)             00000026
                     ART*QUANTITY FIXED(6.3)              00000027
                     ART*QUANTITIES*UNIT CHAR(4)          00000028
                     ART*EPR FIXED(5.2)                   00000029
                 /+-----------------------------------+/  00000030

                 /+OTHER INPUT VARIABLES +/               00000031
  4  1           DECLARE CARD TYPE CHARACTER(2),          00000032
                 REMAINING CHARACTER(39)                  00000033
                 /+---------------------- +/              00000034

                 /+OUTPUT VARIABLES +/                    00000035
  5  1           DCL TOTAL PRICE FIXED(8.2)               00000036
                 INVOICE TOTAL(10.2)                      00000037
                 /+---------------------- +/              00000038
```

Figure 1. Listing of the PL/1 Program translated.

R300 - PL/1 (VERSION 1.4) PROCEDURE 01 RESR THURSDAY 8 MAY 75

DCL.NR	DESIGNATOR	ATTRIBUTE AND REFERENCES
44	A	MARKER CONSTANTS 48
18	ADRE (not found)	ENTRY, DECIMAL, FLOAT(SIMPLE) 85
2	ADRESSE	AUTOMATIC, ALIGNED, STRUCTURE, 83
2	ADDRESS	IN ADDRESS, AUTOMATIC, UNALIGNED, STRING(6), CHARACTER 21
8	NUMBER	AUTOMATIC, ALIGNED, DECIMAL, FIXED(2.0) 11, 14, 55, 92
3	ART*DESIGNA- TION	AUTOMATIC, UNALIGNED, STRING(20), CHARACTER 54, 62, 69
3	ART*EPR	AUTOMATIC, ALIGNED, DECIMAL, FIXED(5.2), 54, 67, 69
3	ART*QUANTITY	AUTOMATIC, ALIGNED, DECIMAL, FIXED(6.3) 54, 58, 61, 67, 69

Figure 2. List of Designators used in the program and error messages.

```
IR SEL   ADR2  ADR   INSTRUCTION  PRBL BL/Z   FEATURE  OPERATION    OPERAND      WEDNESDAY,
                                                                                 AM 7 MAY 75

               5352  -01049       .    031140          AD         1564..A      S
               5358  A01056       0002 031150          TM         1577..A      S

                     ++++++       .    031160          STATEMENT NUMBER 79

 6             5364  CF0079       .    031170          RT IRGA    0079
               5370  C000M8       .    031180          RT IRGN    48         8 1
               5376  +01063       .    031190          TV         1590..A      S
               5382  -00092       .    031200          AD         77590.C      S
               5388  A01063       .    031210          TN         1590..A      S

                     ++++++       .    031220          STATEMENT NUMBER 80
```

Figure 3. Extract from the assembler code produced 07.05.75

HERRN
ARNOLD HATNICHTS

99 NIRGENDS

ZELT NR. 171 F

Rubber gloves	6.000 pair	2.75	16.50
Auto paint			SOLD OUT
Raincoat, size 92	1.000 pieces	68. 0	68.50
Trouser clamps	2.000 pieces	0.25	0.50
Red exterior paint	2.450 KG	10.30	25.23

INVENTORY CONTROL NUMBER 110.73

ARTICLE NUMBER	OLD INVENTORY STOCK	NEW INVENTORY STOCK
3456	234	227
3120	3526	3523
2431	10	1
8743	0	0
2581	24500	24164
4703	600300	600286
6925	753	741
4208	126	123
6420	20	17
1357	300	295
9135	2043	1982
6789	344	342

Figure 4, 5. Result of a computation.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER FTD-ID(RS)I-1369-76	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* THE PL/1 COMPILER FOR THE ROBOTRON 300		5. TYPE OF REPORT & PERIOD COVERED Translation
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) W. Schiller		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Foreign Technology Division Air Force Systems Command U. S. Air Force		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE 1975
		13. NUMBER OF PAGES 25
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

09

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DISTRIBUTION LIST

DISTRIBUTION DIRECT TO RECIPIENT

ORGANIZATION		MICROFICHE
A205	DMATC	1
A210	DMAAC	2
B344	DIA/DS-4C	8
C043	USAMIIA	1
C509	BALLISTIC RES LABS	1
C510	AIR MOBILITY R&D LAB/FIO	1
C513	PICATINNY ARSENAL	1
C535	AVIATION SYS COMD	1
C557	USAIIC	1
C591	FSTC	5
C619	MIA REDSTONE	1
D008	NISC	1
H300	USAICE (USAREUR)	1
P005	ERDA	2
P055	CIA/CRS/ADD/SD	1
NAVORDSTA (50L)		1
NAVWPNSCEN (Code 121)		1
NASA/KSI		1
544 IES/RDPO		1
AFIT/LD		1

ORGANIZATION		MICROFICHE
E053	AF/INAKA	1
E017	AF/RDQLR-W	1
E404	AEDC	1
E408	AFWL	1
E410	ADTC	1
E413	ESD	2
	FTD	
	CCN	1
	ETID	3
	NIA/PHS	1
	NICD	5