Table of Contents

The PL/I Sort is a new feature of Iron Spring PL/I 0.9.6. The sort is a flexible program capable of sorting data on a number of fields in a variety of data formats in a mix of ascending and descending sequence. It is similar to, but not completely compatible with the IBM PLISRTx routines as documented in *PL/I for MVS and VM Programming Guide*, Release 1.1 (SC26-3113-01) In this document features not implemented in the current release are shown with a green background for information only. They may be available in a future release. Differences from the IBM version are highlighted on a red background.

PLISRTx consists of four builtin functions, depending on the source and destination of the data:

Function	Source of input	Destination of output
PLISRTA	File	File
PLISRTB	Procedure	File
PLISRTC	File	Procedure
PLISRTD	Procedure	Procedure

Invoking PLISRTD

The following code invokes PLISRTD:

```
CALL PLISRTD( Sort Statement, Record Statement,
              storage, return code,
              input procedure, output procedure
              [,...] );
```

Any arguments that follow "output procedure" are allowed for compatibility with IBM PLISRTD, and are ignored,

The arguments are:

- **Sort Statement**: A character expression that defines the fields to be sorted ("sort keys").

- **Record Statement**: A character expression that describes the records to be sorted.

- **Storage**: A FIXED value specifying the amount of storage to be used for sorting. This argument is included for compatibility with IBM PLISRTD, and is currently ignored.

- **Return code**: A reference to a FIXED BINARY(31) variable where the sort will store its return code when done. 0=sort successful, 16 = sort failed. This is also set as the value of the builtin subroutine PLIRETC.

- **Input procedure**: A reference to an internal or external user-written procedure which will supply the records to be sorted.

- **Output procedure:** A reference to a user-written procedure which will retrieve the sorted records one at a time.

Sort Statement

The format of the SORT Statement is:

```
"□SORT□FIELDS=(start1,length1,format1,seq1[,...]),
                other options□"
```

□ represents one or more blanks, all alphabetic data must be upper-case

Starting position and length are in bytes.

"*startn*" is the starting position in the record of the *n*th sort key, relative to 1.
"*lengthn*" is the length of the nth key.
"*formatn*" is the format of this key:

- CH: Character data, ASCII encoding, one byte per character.

- BI: Binary data (currently the same as CH)

- ZD: Zoned decimal, "Overpunched" sign in the last byte

- PD: Fixed decimal, 10 bytes per field.

- FI: Signed FIXED BINARY, length=1,2, or 4 bytes

- FL: Floating point

- FS: Numeric characters with optional leading floating sign (e.g."□□-3)

"*seqn*" is the sequence for sorting this key ("A", or "D")

"*other options*": "EQUALS" specifies that the input sequence of records with identical sort keys will be retained, "NOEQUALS" indicates that the order of

these records will be essentially random. Other keywords are allowed for compatibility; they are currently parsed but ignored.

For variable length records, if any part of the specified key is located beyond the end of any record, '00'x will be substituted for the missing data.

Record Statement

The format of the RECORD Statement is:

"□RECORD□TYPE=type,LENGTH=(length[,...])□"

"*type*" is F for fixed-length (nonvarying) records, V for varying

"length" is the maximum record length, not including the length prefix, for TYPE=V, or the actual record length for TYPE=F. The IBM sort allows additional length fields following the first which are ignored by this routine.

Input Procedure

The input procedure is a procedure you provide to supply records to be sorted. This is often called E15 for historical reasons.

SORT calls the input procedure to return one record at a time. If the current record is not the last the input procedure should set the PL/I return code to 12 with the statement `CALL PLIRETC(12);` The sort will continue to call the input procedure until it signals that the last record has been passed via `CALL PLIRETC(8);` Note that the return code of 8 should be set on the <u>next</u> call after the last record has been passed. Setting any other return code is an error.

The input procedure should be declared as returning CHARACTER(n), where *n* is the length of the record to be sorted. If the RECORD statement specifies TYPE=V the input procedure <u>must</u> return a VARYING character string, otherwise it must return nonvarying.

Output Procedure

The output procedure is a procedure you supply to receive sorted records one at a time. This is often called E35 for historical reasons.

SORT calls the output procedure passing it one record at a time. To continue processing the output procedure should set the PL/I return code to 4 before

returning with the statement CALL PLIRETC(4); If it wishes to terminate the sort it should set the return code to 16. Setting any other return code is an error. When the last record has been received the Sort returns to the statement following the call to PLISRTx; the output procedure will not receive any notification of the last record.

The output procedure should be declared with one parameter, a character string, which should be nonvarying either the actual length of the record or an adjustable string [CHARACTER(*)]. If the record type is V an adjustable string should be used.

Sort messages

The sort will generate messages if requested The declaration of the message file "SORTMSG" in PLISRTx is:

```
       PRINT OUTPUT ENV( V CRLF )
```

If you want messages to be printed the calling program must declare a <u>varying</u> character string representing the file name of the message file, as would be specified in the TITLE option of the OPEN statement. It should also declare the external pointer "sortmsg_dd" and set it to the address of the title string. If

```
       DECLARE SORTMSG_DD POINTER EXTERNAL;
       SORTMSG_DD = ADDR(message_file_name);
```

"sortmsg_dd" is not declared or is set to SYSNULL() no messages will be generated. If the name of the title string is "message_file_name", for example, include the following code:

```
Iron Spring PLISRTD version 1.0, Nov, 2014.
 SORT FIELDS=(23,4,FI,D),EQUALS
 RECORD TYPE=F,LENGTH=(26)
Records passed to sort               9
Records passed from sort              9
Sort ended successfully
```

The messages for a successful sort will resemble the following:

Technical Details

Storage requirements; The current version of the sort routines operate on data entirely in memory. The storage requirements are N*(RL+KL+4+E)

- N is the number of records to be sorted
- RL is the record length. For variable length records only the actual lengths plus the two-byte length prefix are used
- KL is the sum of the lengths of all sort keys in the record
- E is four if "EQUALS" is specified, otherwise zero.

Character data: The Iron Spring sort procedure treats character data as ASCII, while the IBM sort routines assume EBCDIC data by default. Data containing a mix of alphabetic, numeric, and special characters will sort differently, so program logic may need to be reviewed. If necessary an EBCDIC sort sequence can be introduced in a future release.

Decimal data: FIXED DECIMAL data is assumed to be in x86 native format. IBM-format packed decimal data will be supported in a future release.

Sort algorithm: The sort procedure uses Hibbard's Modification of Shellsort, as presented in *Internal Sorting Methods Illustrated with PL/I Programs* by Robert P. Rich, Prentice-Hall, 1972; p.73.

Sample Code

```
/****************************************************************/
/* EXAMPLE:    Sample sort program using PLISRTD               */
/*             Iron Spring Software, Inc. 2014                 */
/****************************************************************/
examp: procedure options(main);
  dcl     sortin                record input
                                env( F TEXT  RECSIZE(64) );
  dcl     rc                    fixed bin(31);
  dcl   sortmsg_dd              ptr       external;
  dcl   sortmsg_title  char(64) varying static init( 'sortmsg' );
  /* Statement sorts on 'sale_amount' */
  dcl   srt char(64) varying  static init(
        ' SORT FIELDS=(3,10,PD,D) ' );
  dcl    rec char(64) varying static init(
        ' RECORD TYPE=V,LENGTH=(64) ' );
  dcl   PLISRTD                 builtin;

  /* This is the layout of the records to be sorted */
  dcl  1 sales_detail_rec    unaligned based,
         5 stock_number        fixed bin(15),
         5 sale_amount         fixed dec(7,2),
         5 customer_name,
           10 last             char(32),
           10 first            char(16),
         5 fil                 char(4);

  sortmsg_dd = addr(sortmsg_title);

  open file(sortin) title( 'sort.dat' );
  /* Call the sort */
  call PLISRTD( srt, rec, 0, rc, E15, E35 );

  E15: proc returns( char(64) varying );
    dcl   input_rec            char(64)  varying;
    on endfile(sortin) begin;
      close file(sortin);
      call pliretc(8);                    /* No more records to sort   */
      return( ' ' );
      end;
    read file(sortin) into(input_rec);
    call pliretc(12);                     /* Add this record to sort   */
    return( input_rec );
  end E15;

 E35: proc( inrec );
```

```
dcl     inrec                 char(*);
dcl     p                     ptr;
p = addr(inrec);
put skip edit( p->stock_number, p->sale_amount,
               p->customer_name.last, p->customer_name.first )
             ( p'zzz9B',p'$$$,$$$v.99B',a,x(1),a );
call pliretc(4);                        /* give me another record    */
end E35;

end examp;
```