RTL/2
Standard stream I/O

RTL/2
Standard stream I/O

RTL/2 Reference : 5
Version : 2

Contents

0 Introduction

This manual describes the specifications of the standard stream procedures for input and output that should always be available in all RTL/2 systems which allow information in character form to be transferred between tasks and channels (in both directions). These facilities assume that the standard RTL/2 streaming and error recovery mechanisms are available in the systems. (A summary of these mechanisms is given in Section 2 and full details are given in the RTL/2 System Standards manual). System designers and writers are strongly recommended to implement these procedures and likewise users are urged to use them, thereby bringing about a uniformity in this area in all programs and consequently aiding their transportability, understanding and maintenance.

The procedures described in detail in Sections 3 and 4 should always be readily available in RTL/2 systems. This does not mean that they have to be an integral part of the system (although some almost certainly will be); those which are not part of the system should be available in a form which makes them easily accessible to a user at the point of loading his program (e.g. RTL/2 source on cards or paper tape, "object" code which can be "linked" into his program).

The information in this manual is relevant to both system designers and writers and users.

In Section 3 the reader will encounter various specifications described by a syntax which is defined in Section 0.2 of the RTL/2 Language Specification manual.

REFERENCES

RTL/2 Language Specification — RTL/2 Reference : 1
RTL/2 System Standards — RTL/2 Reference : 4

1 Reserved stream I/O names

The following names are reserved for the standard stream I/O facilities specified in Sections 3 and 4 and must not be used as external names for any other facilities.

FREAD
FWRT
FWRTF
IREAD
IWRT
IWRTF
NLS
RREAD
RWRT
RWRTF
SPS
TREAD
TWRT

2 Summary of streamed input and output and error recovery mechanisms in RTL/2 systems

This section summarises the streamed input and output and error recovery mechanisms which are assumed to be available in an RTL/2 system by the facilities described in Sections 3 and 4. Full details of these mechanisms are given in the RTL/2 System Standards manual.

2.1 Streamed Input and Output

Each task has two SVC data bricks associated with it, namely:

i)
```
DATA RRSIO;
    PROC( )BYTE IN;
    PROC(BYTE) OUT;
ENDDATA
```

The procedure in IN will remove the next character from the current input stream and return it as result. The procedure in OUT will send the character passed as parameter to the current output stream. All streaming of individual characters will be via IN and OUT as appropriate.

ii)
```
DATA RRSED;
    BYTE TERMCH,
         IOFLAG;
ENDDATA
```

TERMCH and IOFLAG are concerned with the standard stream input procedures (see Section 3). On returning to a task which has called one of the procedures TERMCH will contain the last character read (and removed) from the current input stream. IOFLAG is used as one of the ways of indicating that an input format error has occurred (although it may be used for other I/O error indications as well in some systems).

Streaming provides a convenient method of transferring characters between a task and a channel – a channel being either a receiver or sender of characters (or both) – in a manner which is independent of the channel. Often a channel will be a physical device (e.g. teletype, paper tape reader), but it can also be a logical device (such as an internal array).

There must be a procedure which transfers characters to or from a task for each channel (there need not be a separate procedure for each channel since some procedures may handle several channels). The system will normally provide such procedures for channels which are physical devices; those for logical devices will normally be provided by the task. For input channels they must have the same description as IN in RRSIO, for output channels they must have the same description as OUT.

In order to specify the input or output channel to which streaming is to be applied, the value of the appropriate procedure must be placed in IN or OUT respectively, thereby defining the current stream in that direction. The way in which this is actually done will be dependent on the system in which the task is running – for example, if the system provides the procedure for transferring the characters to or from the required channel then this may be achieved by calling another system procedure, specifying the channel by number; if the channel is a logical device a direct assignment of the procedure value to IN or OUT may be all that is necessary.

Having provided a current stream, a task then has access to it via IN or OUT (for single characters) and the standard stream procedures described below (for numbers and text etc). There may of course be additional facilities in a system which also use the current streams.

It should be noted that the characters presented to a channel will not necessarily be identical to those processed by a task. For instance, a teletype needs two characters (carriage return and line feed) to produce a new line whereas the task will only consider one character (the newline, value HEX 0A) when wishing to achieve the same effect.

Users should consult the documentation for the specific systems which they are using for details on setting up current streams and the external and internal representation of characters.

2.2 Error Recovery

Each task has one SVC data brick associated with it, namely

```
DATA RRERR;
    LABEL ERL;
    INT ERN;
    PROC(INT) ERP;
ENDDATA
```

ERL contains the unrecoverable error label, ERN the unrecoverable error number and ERP the recoverable error procedure. The parameter passed to ERP is the recoverable error number.

A procedure to enable an unrecoverable error to be simulated and monitored is also available:

PROC RRGEL(INT N)

This sets ERN to N, invokes any error monitoring facilities and then passes control to ERL.

In fact only the standard stream input procedures are directly concerned with this error mechanism. They will call the recoverable error procedure with an appropriate recoverable error number as parameter if input in an illegal format is encountered (see Section 3).

3 Stream input facilities

This section contains detailed specifications of the stream input facilities that should be provided with all RTL/2 systems.

These facilities all interface to the system through the mechanism described in Section 2.

All the procedures described also have two actions in common:

i) the last character read, and therefore removed, from the current input stream (otherwise known as the terminating character) will always be placed in TERMCH in RRSED (see Section 2) regardless of whether the input action requested was carried out successfully or not.

ii) an error encountered in the format of the input (see sections below) is always reported to the calling task in the following manner:

a) the value of IOFLAG in RRSED is set to 1;

b) the recoverable error procedure contained in ERP in RRERR is called. The recoverable error number, which is passed as parameter to this procedure, will identify this fact and also the procedure which was being called — the values it can take are given in Section 3.6.

Note that since the error procedure may not return to the calling procedure TERMCH must be set up before ERP is called.

3.1 Character Input

Individual characters are obtained from the current input stream by use of the procedure variable IN in data brick RRSIO (see Section 2).

3.2 PROC FREAD( )FRAC

This reads a decimal number from the current input stream and returns a truncated fraction value as result. The format of the number will be:

[layout] [+ | –] digit.digitlist termch

where layout ::= lchar . . .
lchar ::=space | tab |newline
space::=character-of-value-HEX 20
tab::=character-of-value-HEX 09
newline::=character-of-value-HEX 0A
digitlist::=digit . . .
digit::=0|1|2|3|4|5|6|7|8|9
termch::=character-which-cannot-be-part-of-the-number

Possible errors are illegal format or fraction (fixed-point) overflow (see 3 (ii) above and 3.6). Note that the result is then undefined.

However a representation of the value 1, e.g. 1.0, may be read in without causing overflow, the result being the largest possible value for a positive fraction in this case (thus any value output by FWRTF, see 4.5, will be legal input to FREAD).

3.3 PROC IREAD( )INT

This reads a decimal integer from the current input stream and returns its value as result. The format of the integer will be:

[layout] [+ | –] digitlist termch

where layout::= lchar . . .
lchar ::=space | tab |newline
space::=character-of-value-HEX 20
tab::=character-of-value-HEX 09
newline::=character-of-value-HEX 0A
digitlist::=digit . . .
digit::=0|1|2|3|4|5|6|7|8|9
termch::=character-which-cannot-be-part-of-the-integer

Possible errors are illegal format and integer (fixed-point) overflow (see 3 (ii) above and 3.6). Note that the result is then undefined.

3.4 PROC RREAD( )REAL

This reads a decimal number from the current input stream and returns its value as result. The format of the number will be:

[layout] [+ | −] digitlist[.digitlist] [E[+ | −] digitlist] termch

where layout::=lchar . . .
lchar ::=space | tab |newline
space::=character-of-value-HEX 20
tab::=character-of-value-HEX 09
newline::=character-of-value-HEX 0A
digitlist::=digit . . .
digit::=0|1|2|3|4|5|6|7|8|9
termch::=character-which-cannot-be-part-of-the-number

Possible errors are illegal format or real (floating-point) overflow (see 3(ii) above and 3.6). Note that the result is then undefined.

3.5 PROC TREAD(REF ARRAY BYTE X,T)INT

This reads characters from the current input stream and places them into successive elements of the array referenced by X beginning at element 1. The values in the elements of the array referenced by T contain a selection of terminating characters. The input is terminated as soon as one of these terminating characters is encountered. The terminating character is not put into X. The number of characters placed in X is returned as result.

An error will have occurred if a terminating character has not been encountered after LENGTH X + 1 characters have been read (see 3 (ii) above and 3.6). Note that in this case the result will then be LENGTH X.

3.6 Input Errors

3.6.1 Illegal Format

The recoverable error numbers reported by the above procedures if illegal format in the input is encountered are:

Number	Procedure
100	FREAD
101	IREAD
102	RREAD
103	TREAD

3.6.2 Overflow

Arithmetic overflow can occur in FREAD, IREAD and RREAD. The way in which this is reported to the calling task will depend on the system in which it is running, and this is governed principally by the overflow hardware facilities made available by the host computer.

4 Stream output facilities

This section contains detailed specifications of the stream output facilities that will be provided with all RTL/2 systems.

These facilities all interface to the system through the mechanisms described in Section 2.

4.1 Character Output

Individual characters are sent to the current output stream by use of the procedure variable OUT in data brick RRSIO (see Section 2).

4.2 PROC NLS(INT N)

This sends N newline characters (value HEX 0A) to the current output stream.

4.3 PROC SPS(INT N)

This sends N space characters (value HEX 20) to the current output stream.

4.4 PROC FWRT(FRAC X)

This sends the unrounded fraction value X to the current output stream as a decimal number in a fixed format. This format is:

a) minus sign if X<0.0B0
b) digit
c) decimal point
d) f digits

The number of digits after the decimal point (f) may vary between implementations, and will be dependent on specific representations of fractions.

4.5 PROC FWRTF(FRAC X,INT N)

This sends the rounded fraction value X to the current output stream as a decimal number in a field of width N+3 in the following format:

a) minus sign if X<0.0B0, otherwise a space
b) digit
c) decimal point
d) N digits

Note that at least one digit will always be output after the decimal point, even if N≤0.

4.6 PROC IWRT(INT X)

This sends the integer value X to the current output stream as a decimal integer in a fixed format. This format is:

a) minus sign if X<0
b) decimal digits of the integer with leading zeros suppressed

4.7 PROC IWRTF(INT X,M)

This sends the integer value X to the current output stream as a decimal integer in a field of width M+1. The integer will be right-justified within this field, leading zeros will be suppressed and it will be preceded by a minus sign (X<0) or a space (X≥0). If the integer overflows the field then the field will be expanded as necessary.

4.8 PROC RWRT(REAL X)

This sends the unrounded real value X to the current output stream as a decimal number in a fixed format. This format will be:

a) minus sign if X<0.0
b) digit
c) decimal point
d) r digits
e) E
f) decimal exponent as an integer in the format given by IWRT (see 4.6)

The number of digits (r) after the decimal point may vary between implementations, and will be dependent on specific representations of reals.

4.9 PROC RWRTF(REAL X,INT M,N)

This sends the rounded real value X to the current output stream as a decimal number in a format determined by M and N.

a) M≠0, N=0
the integer part of X as for IWRTF (see 4.7)
Field width is M+1

b) M≠0, N≠0
i) the integer part of X as for IWRTF (field width M+1)
ii) decimal point
iii) N fractional places
Field width is M+N+2

c) M=0, N>0
i) minus sign (X<0.0) or space (X≥0.0)
ii) digit
iii) decimal point
iv) N digits
v) E
vi) 2 digit decimal exponent preceded by a + or − sign as appropriate

Field width is N+7

If the decimal exponent exceeds two digits (i.e. > 99) then its digit field will be expanded as necessary.

If X overflows the format requested it will be output in the format given by the values 0 and M+N for M and N respectively. The field width will then be M+N+7.

4.10 PROC TWRT(REF ARRAY BYTE X)

This sends LENGTH X characters from successive elements of the array referenced by X, starting at element 1, to the current output stream.