*MINVAL/*MAXVAL - Evaluate the Minimum/Maximum

$$\left\{ \begin{array}{l} \textbf{*MINVAL} \\ \textbf{*MAXVAL} \end{array} \right\} \quad \textbf{([(IR}=\textit{result-format/length}\textbf{)]}\; \textit{operand,...}\textbf{)}$$

Format/length: Format and length may be specified explicitly using the `IR` clause or evaluated automatically using the *Format/Length Conversion Rule Tables* below.

This chapter covers the following topics:

- Function
- Restrictions
- Syntax Description
- Resulting Format/Length Conversion Rule Tables
- Evaluating the result-format-length

Function

The Natural system function `*MINVAL`/`*MAXVAL` evaluates the minimum/maximum value of all given operand values. The result is always a scalar value. If an array is specified as operand, the minimum/maximum of all array fields is evaluated.

When using alphanumerical or binary data as an argument, if the data is the same (for example, `*MINVAL('AB','AB')`), then the result is the argument with the smallest/largest length value.

Restrictions

When using the system function `*MINVAL`/`*MAXVAL`, the following restrictions apply:

- `*MINVAL`/`*MAXVAL` must not be used where a target variable is expected.
- You may not nest `*MINVAL`/`*MAXVAL` in a system function.

Syntax Description

Operand Definition Table:

Operand	Possible Structure				Possible Formats												Referencing Permitted	Dynamic Definition
operand	C	S	A	G		A	U	N	P	I	F	B	D	T				yes	no

Syntax Element Description:

Keyword	Description
`*MINVAL`	Evaluates the minimum value of all given operand values.
`*MAXVAL`	Evaluates the maximum value of all given operand values.
`operand`	The operand(s) whose minimum/maximum values are to be evaluated by the `*MINVAL`/`*MAXVAL` system function.
`result-format-length`	Intermediate Result clause for explicit specification of the resulting format/length. See *IR Clause* below.

IR Clause

The `IR` (Intermediate Result) clause may be used in order to specify explicitly the *result-format/length* of the whole `*MINVAL`/`*MAXVAL` system function.

```
IR=result-format/length
```

```
IR=  ⎡ format-length          ⎤
     ⎢                        ⎥
     ⎢    ⎡ A ⎤               ⎥
     ⎢ (  ⎢ U ⎥ ) DYNAMIC     ⎥
     ⎢    ⎣ B ⎦               ⎥
     ⎣                        ⎦
```
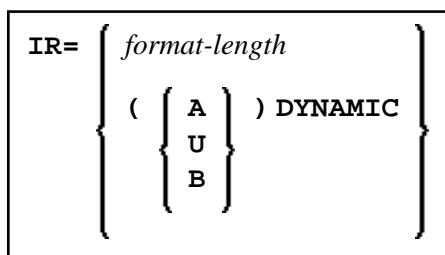
For an assortment of valid result-format/lengths, refer to the *Format/Length Conversion Rule Tables* below.

Syntax Element Description:

Keyword	Description
`format-length`	The compiler tries to determine the resulting format/length of the whole function. If the compiler cannot determine a format/length in a way that no loss of precision is guaranteed, the *format-length* must be set by the programmer using the `IR` operand extension.
A, U or B	Format: Alphanumeric, Unicode or Binary for dynamic variable.
DYNAMIC	Instead of specifying a fixed format/length, you may specify an alphanumeric, Unicode or binary format with dynamic length.

Example:

```
DEFINE DATA LOCAL
1 #RESULTI        (I4)
1 #RESULTA        (A20)
1 #RESULTADYN     (A) DYNAMIC
1 #A(I4)          CONST <1234>
1 #B(A20)         CONST <H'30313233'> /* '0123' stored
1 #C(I2/1:3)      CONST <2000, 2100, 2200>
END-DEFINE
*
#RESULTA     := *MAXVAL((IR=A20)          #A, #B)     /*no error, I4->A20 is allowed!
#RESULTADYN  := *MAXVAL((IR=(A)DYNAMIC)   #A, #B)     /*result is (A) dynamic
/* #RESULTI  := *MAXVAL((IR=I4)           #A, #B)     /*compiler error, because conv. A20->I4 is not allowed!
#RESULTI     := *MAXVAL((IR=I4)           #A, #C(*))  /*maximum of the array is evaluated
DISPLAY #RESULTA #RESULTADYN (AL=10) #RESULTI
END
```

Resulting Format/Length Conversion Rule Tables

There are different ways to define the resulting format/length of the whole `*MINVAL/*MAXVAL` system function.

- Explicit Specification of the Resulting Format/Length

- Implicit Specification of the Resulting Format/Length

Explicit Specification of the Resulting Format/Length

The resulting format/length of the whole `*MINVAL/*MAXVAL` system function may be specified by the `IR` clause. All operands specified will be converted into this resulting format/length, if this is possible without any loss of precision. Afterwards the minimum/maximum of all the converted operands will be evaluated and one single scalar value with the evaluated format/length will be set as result of the whole system function.

Implicit Specification of the Resulting Format/Length

If no `IR` clause is used inside the `*MINVAL/*MAXVAL` system function, the resulting format/length will be evaluated regarding the format/length of all operands specified as arguments inside the `*MINVAL/*MAXVAL` system function. The format/length of each operand is taken and combined with the format/length of the next following operand of the argument list. The resulting format/length of two single operands are then evaluated using the Format/Length Conversion Rule Tables below.

The Format/Length Conversion Rule Table is separated into two different subtables. All combinations not shown in the two tables below are invalid and must not be applied inside the argument list of the `*MINVAL/*MAXVAL` system function. The keyword `FLF` indicates that the `IR` clause must be used in order to define the resulting format/length, because there otherwise may be a loss of precision.

Table 1

Covers all the numeric combinations of two different operands.

	Format-length	**Second Operand**				
		I1	**I2**	**I4**	**P$a.b$, N$a.b$**	**F4, F8**
First Operand	**I1**	I1	I2	I4	Pmax(3,a).b	F8
	I2	I2	I2	I4	Pmax(5,a).b	F8
	I4	I4	I4	I4	Pmax(10,a).b	F8
	P$x.y$, N$x.y$	Pmax(3,x).y	Pmax(5,x).y	Pmax(10,x).y	if max(x,a) + max(y,b) <= 29 Pmax(x,a).max(y,b) else FLF	if $y=0$ and $x <=15$; F8 else FLF
	F4, F8	F8	F8	F8	if $b=0$ and $a <=15$ F8 else FLF	F8

Legend:

FLF	Format-length declaration forced. The resulting format must be specified using the IR clause.
Ix	Format/length is Integer. x specifies the number of bytes which are used to store the Integer value.
Fx	Format/length is Float. x specifies the number of bytes which are used to store the Float value.
P$x.y$ **Pa,b**	Packed format with corresponding number of digits before the decimal point (x,a) and the precision (y,b).
N$x.y$ **Na,b**	Numeric format with corresponding number of digits before the decimal point (x,a) and the precision (y,b).
Pmax(c,d).e	The resulting format is packed. The length is evaluated by the information following. The number of digits before the decimal point is the maximum value of c and d. The precision value is e.
Pmax(c,d).max(e,f)	The resulting format is packed. The length is evaluated by the information following. The number of digits before the decimal point is the maximum value of c and d. The precision value is the maximum value of e and f.

Table 2

Covers all other formats and lengths which may be used for *MINVAL/*MAXVAL system function operands.

First Operand	Second Operand					
	Format-length	D	T	A*a*, A dynamic	B*a*, B dynamic	U*a*, U dynamic
	D	D	T	NA	NA	NA
	T	T	T	NA	NA	NA
	A*x*, A dynamic	NA	NA	A dynamic	A dynamic	U dynamic
	B*x*, B dynamic	NA	NA	A dynamic	B dynamic	U dynamic
	U*x*, U dynamic	NA	NA	U dynamic	U dynamic	U dynamic

Legend:

NA	This combination is not allowed.
D	Date format.
T	Time format.
B*x*, B*a*	Binary format with length *x, a*.
A*x*, A*a*	Alphanumeric format with length *x, a*.
U*x*, U*a*	Unicode format with length *x, a*.
B dynamic	Binary format with dynamic length.
A dynamic	Alphanumeric format with dynamic length.
U dynamic	Unicode format with dynamic length.

Evaluating the *result-format-length*

Using the rules described above, the compiler is able to process the source operands by regarding pairs of operands and calculating an intermediate result for each pair. The first pair consists of the first and the second operand, the second pair of the intermediate result and the third operand, etc. After all operands have been processed, the last result shows the comparison of format and length which will be used to compare all operands in order to evaluate the minimum/maximum. When you use this method of format-length evaluation, the operand *format-lengths* can appear in any order.

Example:

```
DEFINE DATA LOCAL
1 A (I2)      INIT <34>
1 B (P4.2)    INIT <1234.56>
1 C (N4.4)    INIT <12.6789>
1 D (I1)      INIT <100>
1 E (I4/1:3)  INIT <32, 6745, 456>
1 #RES-MIN (P10.7)
1 #RES-MAX (P10.7)
END-DEFINE
*
MOVE *MINVAL(A, B, C, D, E(*)) TO #RES-MIN
MOVE *MAXVAL(A, B, C, D, E(*)) TO #RES-MAX
DISPLAY #RES-MIN #RES-MAX
END
```

Output:

```
          #RES-MIN                          #RES-MAX
--------------------------    --------------------------

              12.6789000                    6745.0000000
```

The following table shows the single steps evaluating the format/length of the example automatically. It shows the intermediate result (ir) of all steps and the comparison format/length (cf) which is used as *result-format/length*.

Evaluation Order	Name of First Operand	Format/Length of First Operand or Intermediate Result	Name of Second Operand	Format/Length of Second Operand or Intermediate Result	Format/Length of the Intermediate Result (ir)
1.	A	I2	B	P4.2	ir1 = P5.2
2.	ir1	P5.2	C	N4.4	ir2 = P5.4
3.	ir2	P5.4	D	I1	ir3 = P5.4
4.	ir3	P5.4	E	I4	**cf = P10.4**

During runtime, all operands are converted into the cf format/length; then all converted values are compared, and the corresponding minimum/maximum is evaluated.

Format/Length Evaluation Order

The following graphic represents the order in which format and length are evaluated:
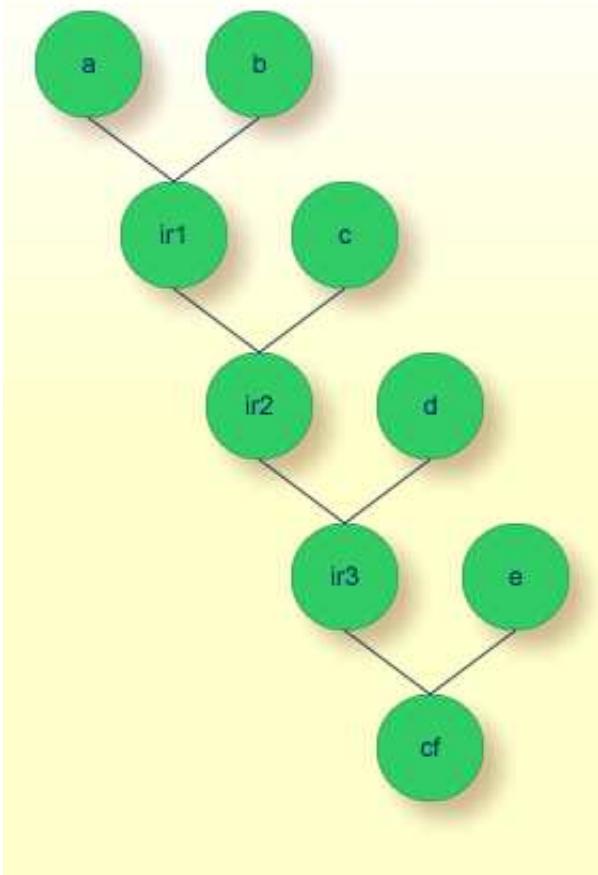
Legend:

ir1, ir2, ir3	Intermediate result 1, 2, 3.
cf	Comparison of format and length.