CHAPTER 2

Fortran 95 Intrinsic Functions

This chapter lists the intrinsic function names recognized by the `f95` compiler.

2.1 Standard Fortran 95 Generic Intrinsic Functions

The generic Fortran 95 intrinsic functions are grouped in this section by functionality as they appear in the Fortran 95 standard.

The arguments shown are the names that can be used as argument keywords when using the keyword form, as in `cmplx(Y=B, KIND=M, X=A)`.

Consult the Fortran 95 standard for the detailed specifications of these generic intrinsic procedures.

2.1.1 Argument Presence Inquiry Function

Generic Intrinsic Name	Description
`PRESENT`	Argument presence

2.1.2 Numeric Functions

Generic Intrinsic Name	Description
`ABS (A)`	Absolute value
`AIMAG (Z)`	Imaginary part of a complex number
`AINT (A [, KIND])`	Truncation to whole number
`ANINT (A [, KIND])`	Nearest whole number
`CEILING (A [, KIND])`	Least integer greater than or equal to number
`CMPLX (X [, Y, KIND])`	Conversion to complex type
`CONJG (Z)`	Conjugate of a complex number
`DBLE (A)`	Conversion to double precision real type
`DIM (X, Y)`	Positive difference
`DPROD (X, Y)`	Double precision real product
`FLOOR (A [, KIND])`	Greatest integer less than or equal to number
`INT (A [, KIND])`	Conversion to integer type
`MAX (A1, A2 [, A3,...])`	Maximum value
`MIN (A1, A2 [, A3,...])`	Minimum value

`MOD (A, P)`	Remainder function
`MODULO (A, P)`	Modulo function
`NINT (A [, KIND])`	Nearest integer
`REAL (A [, KIND])`	Conversion to real type
`SIGN (A, B)`	Transfer of sign

2.1.3 Mathematical Functions

Generic Intrinsic Name	Description
`ACOS (X)`	Arccosine
`ASIN (X)`	Arcsine
`ATAN (X)`	Arctangent
`ATAN2 (Y, X)`	Arctangent
`COS (X)`	Cosine
`COSH (X)`	Hyperbolic cosine
`EXP (X)`	Exponential

LOG (X)	Natural logarithm
LOG10 (X)	Common logarithm (base 10)
SIN (X)	Sine
SINH (X)	Hyperbolic sine
SQRT (X)	Square root
TAN (X)	Tangent
TANH (X)	Hyperbolic tangent

2.1.4 Character Functions

Generic Intrinsic Name	Description
ACHAR (I)	Character in given position in ASCII collating sequence
ADJUSTL (STRING)	Adjust left
ADJUSTR (STRING)	Adjust right
CHAR (I [, KIND])	Character in given position in processor

	collating sequence
IACHAR (C)	Position of a character in ASCII collating sequence
ICHAR (C)	Position of a character in processor collating sequence
INDEX (STRING, SUBSTRING [, BACK])	Starting position of a substring
LEN_TRIM (STRING)	Length without trailing blank characters
LGE (STRING_A, STRING_B)	Lexically greater than or equal
LGT (STRING_A, STRING_B)	Lexically greater than
LLE (STRING_A, STRING_B)	Lexically less than or equal
LLT (STRING_A, STRING_B)	Lexically less than
REPEAT (STRING, NCOPIES)	Repeated concatenation
SCAN (STRING, SET [, BACK])	Scan a string for a character in a set
TRIM (STRING)	Remove trailing blank characters
VERIFY (STRING, SET [, BACK])	Verify the set of characters in a string

2.1.5 Character Inquiry Function

Generic Intrinsic Name	Description
`LEN (STRING)`	Length of a character entity

2.1.6 Kind Functions

Generic Intrinsic Name	Description
`KIND (X)`	Kind type parameter value
`SELECTED_INT_KIND (R)`	Integer kind type parameter value, given range
`SELECTED_REAL_KIND ([P, R])`	Real kind type parameter value, given precision and range

2.1.7 Logical Function

Generic Intrinsic Name	Description
`LOGICAL (L [, KIND])`	Convert between objects of type logical with different kind type parameters

2.1.8 Numeric Inquiry Functions

Generic Intrinsic Name	Description
`DIGITS (X)`	Number of significant digits of the model
`EPSILON (X)`	Number that is almost negligible compared to one
`HUGE (X)`	Largest number of the model
`MAXEXPONENT (X)`	Maximum exponent of the model
`MINEXPONENT (X)`	Minimum exponent of the model
`PRECISION (X)`	Decimal precision
`RADIX (X)`	Base of the model
`RANGE (X)`	Decimal exponent range
`TINY (X)`	Smallest positive number of the model

2.1.9 Bit Inquiry Function

Generic Intrinsic Name	Description

BIT_SIZE (I)	Number of bits of the model

2.1.10 Bit Manipulation Functions

Generic Intrinsic Name	Description
BTEST (I, POS)	Bit testing
IAND (I, J)	Logical AND
IBCLR (I, POS)	Clear bit
IBITS (I, POS, LEN)	Bit extraction
IBSET (I, POS)	Set bit
IEOR (I, J)	Exclusive OR
IOR (I, J)	Inclusive OR
ISHFT (I, SHIFT)	Logical shift
ISHFTC (I, SHIFT [, SIZE])	Circular shift
NOT (I)	Logical complement

2.1.11 Transfer Function

Generic Intrinsic Name	Description
`TRANSFER (SOURCE, MOLD [, SIZE])`	Treat first argument as if of type of second argument

2.1.12 Floating-Point Manipulation Functions

Generic Intrinsic Name	Description
`EXPONENT (X)`	Exponent part of a model number
`FRACTION (X)`	Fractional part of a number
`NEAREST (X, S)`	Nearest different processor number in given direction
`RRSPACING (X)`	Reciprocal of the relative spacing of model numbers near given number
`SCALE (X, I)`	Multiply a real by its base to an integer power
`SET_EXPONENT (X, I)`	Set exponent part of a number
`SPACING (X)`	Absolute spacing of model numbers near given number

2.1.13 Vector and Matrix Multiply Functions

Generic Intrinsic Name	Description
`DOT_PRODUCT (VECTOR_A, VECTOR_B)`	Dot product of two rank-one arrays
`MATMUL (MATRIX_A, MATRIX_B)`	Matrix multiplication

2.1.14 Array Reduction Functions

Generic Intrinsic Name	Description
`ALL (MASK [, DIM])`	True if all values are true
`ANY (MASK [, DIM])`	True if any value is true
`COUNT (MASK [, DIM])`	Number of true elements in an array
`MAXVAL (ARRAY, DIM [, MASK])` or `MAXVAL (ARRAY [, MASK])`	Maximum value in an array
`MINVAL (ARRAY, DIM [, MASK])` or `MINVAL (ARRAY [, MASK])`	Minimum value in an array

`PRODUCT (ARRAY, DIM [, MASK])` or `PRODUCT (ARRAY [, MASK])`	Product of array elements
`SUM (ARRAY, DIM [, MASK])` or `SUM (ARRAY [, MASK])`	Sum of array elements

2.1.15 Array Inquiry Functions

Generic Intrinsic Name	Description
`ALLOCATED (ARRAY)`	Array allocation status
`LBOUND (ARRAY [, DIM])`	Lower dimension bounds of an array
`SHAPE (SOURCE)`	Shape of an array or scalar
`SIZE (ARRAY [, DIM])`	Total number of elements in an array
`UBOUND (ARRAY [, DIM])`	Upper dimension bounds of an array

2.1.16 Array Construction Functions

Generic Intrinsic Name	Description

`MERGE (TSOURCE, FSOURCE, MASK)`	Merge under mask
`PACK (ARRAY, MASK [, VECTOR])`	Pack an array into an array of rank one under a mask
`SPREAD (SOURCE, DIM, NCOPIES)`	Replicates array by adding a dimension
`UNPACK (VECTOR, MASK, FIELD)`	Unpack an array of rank one into an array under a mask

2.1.17 Array Reshape Function

Generic Intrinsic Name	Description
`RESHAPE (SOURCE, SHAPE[, PAD, ORDER])`	Reshape an array

2.1.18 Array Manipulation Functions

Generic Intrinsic Name	Description
`CSHIFT (ARRAY, SHIFT [, DIM])`	Circular shift
`EOSHIFT (ARRAY, SHIFT [, BOUNDARY, DIM]`	End-off shift
`TRANSPOSE (MATRIX)`	Transpose of an array of rank two

2.1.19 Array Location Functions

Generic Intrinsic Name	Description
`MAXLOC (ARRAY, DIM [, MASK])` or `MAXLOC (ARRAY [, MASK])`	Location of a maximum value in an array
`MINLOC (ARRAY, DIM [, MASK])` or `MINLOC (ARRAY [, MASK])`	Location of a minimum value in an array

2.1.20 Pointer Association Status Functions

Generic Intrinsic Name	Description
`ASSOCIATED (POINTER [, TARGET])`	Association status inquiry or comparison
`NULL ([MOLD])`	Returns disassociated pointer

2.1.21 System Environment Procedures

Generic Intrinsic Name	Description

COMMAND_ARGUMENT_COUNT ()	Returns number of command arguments
GET_COMMAND ([COMMAND, LENGTH, STATUS])	Returns entire command that invoked the program
GET_COMMAND_ARGUMENT (NUMBER [, VALUE, LENGTH, STATUS])	Returns a command argument
GET_ENVIRONMENT_VARIABLE (NAME [, VALUE, LENGTH, STATUS, TRIM_NAME])	Obtain the value of an environment variable.

2.1.22 Intrinsic Subroutines

Generic Intrinsic Name	Description
CPU_TIME (TIME)	Obtain processor time
DATE_AND_TIME ([DATE, TIME, ZONE, VALUES])	Obtain date and time
MVBITS (FROM, FROMPOS, LEN, TO, TOPOS)	Copies bits from one integer to another
RANDOM_NUMBER (HARVEST)	Returns pseudorandom number
RANDOM_SEED ([SIZE, PUT, GET])	Initializes or restarts the pseudorandom number generator

SYSTEM_CLOCK ([COUNT, COUNT_RATE, COUNT_MAX])	Obtain data from the system clock

2.1.23 Specific Names for Intrinsic Functions

TABLE 2-1 Specific and Generic Names for Fortran 95 Intrinsic Functions

	Specific Name	Generic Name	Argument Type
	ABS (A)	ABS (A)	default real
	ACOS (X)	ACOS (X)	default real
	AIMAG (Z)	AIMAG (Z)	default complex
	AINT (A)	AINT (A)	default real
	ALOG (X)	LOG (X)	default real
	ALOG10 (X)	LOG10 (X)	default real
#	AMAX0 (A1, A2 [, A3,...])	REAL (MAX (A1, A2 [, A3,...]))	default integer
#	AMAX1 (A1, A2 [, A3,...])	MAX (A1, A2 [, A3,...])	default real

#	AMIN0 (A1, A2 [, A3,...])	REAL (MIN (A1, A2 [, A3,...]))	default integer
#	AMIN1 (A1, A2 [, A3,...])	MIN (A1, A2 [, A3,...])	default real
	AMOD (A, P)	MOD (A, P)	default real
	ANINT (A)	ANINT (A)	default real
	ASIN (X)	ASIN (X)	default real
	ATAN (X)	ATAN (X)	default real
	ATAN2 (Y, X)	ATAN2 (Y, X)	default real
	CABS (A)	ABS (A)	default complex
	CCOS (X)	COS (X)	default complex
	CEXP (X)	EXP (X)	default complex
#	CHAR (I)	CHAR (I)	default integer
	CLOG (X)	LOG (X)	default complex
	CONJG (Z)	CONJG (Z)	default complex
	COS (X)	COS (X)	default real

	COSH (X)	COSH (X)	default real
	CSIN (X)	SIN (X)	default complex
	CSQRT (X)	SQRT (X)	default complex
	DABS (A)	ABS (A)	double precision
	DACOS (X)	ACOS (X)	double precision
	DASIN (X)	ASIN (X)	double precision
	DATAN (X)	ATAN (X)	double precision
	DATAN2 (Y, X)	ATAN2 (Y, X)	double precision
	DCOS (X)	COS (X)	double precision
	DCOSH (X)	COSH (X)	double precision
	DDIM (X, Y)	DIM (X, Y)	double precision

	DEXP (X)	EXP (X)	double precision
	DIM (X, Y)	DIM (X, Y)	default real
	DINT (A)	AINT (A)	double precision
	DLOG (X)	LOG (X)	double precision
	DLOG10 (X)	LOG10 (X)	double precision
#	DMAX1 (A1, A2 [, A3,...])	MAX (A1, A2 [, A3,...])	double precision
#	DMIN1 (A1, A2 [, A3,...])	MIN (A1, A2 [, A3,...])	double precision
	DMOD (A, P)	MOD (A, P)	double precision
	DNINT (A)	ANINT (A)	double precision
	DPROD (X, Y)	DPROD (X, Y)	default real
	DSIGN (A, B)	SIGN (A, B)	double precision
	DSIN (X)	SIN (X)	double precision

	DSINH (X)	SINH (X)	double precision
	DSQRT (X)	SQRT (X)	double precision
	DTAN (X)	TAN (X)	double precision
	DTANH (X)	TANH (X)	double precision
	EXP (X)	EXP (X)	default real
#	FLOAT (A)	REAL (A)	default integer
	IABS (A)	ABS (A)	default integer
#	ICHAR (C)	ICHAR (C)	default character
	IDIM (X, Y)	DIM (X, Y)	default integer
#	IDINT (A)	INT (A)	double precision
	IDNINT (A)	NINT (A)	double precision
#	IFIX (A)	INT (A)	default real
	INDEX (STRING, SUBSTRING)	INDEX (STRING, SUBSTRING)	default character

#	INT (A)	INT (A)	default real
	ISIGN (A, B)	SIGN (A, B)	default integer
	LEN (STRING)	LEN (STRING)	default character
#	LGE (STRING_A, STRING_B)	LGE (STRING_A, STRING_B)	default character
#	LGT (STRING_A, STRING_B)	LGT (STRING_A, STRING_B)	default character
#	LLE (STRING_A, STRING_B)	LLE (STRING_A, STRING_B)	default character
#	LLT (STRING_A, STRING_B)	LLT (STRING_A, STRING_B)	default character
#	MAX0 (A1, A2 [, A3,...])	MAX (A1, A2 [, A3,...])	default integer
#	MAX1 (A1, A2 [, A3,...])	INT (MAX (A1, A2 [, A3,...]))	default real
#	MIN0 (A1, A2 [, A3,...])	MIN (A1, A2 [, A3,...])	default integer
#	MIN1 (A1, A2 [, A3,...])	INT (MIN (A1, A2 [, A3,...]))	default real
	MOD (A, P)	MOD (A, P)	default integer
	NINT (A)	NINT (A)	default real

#	REAL (A)	REAL (A)	default integer
	SIGN (A, B)	SIGN (A, B)	default real
	SIN (X)	SIN (X)	default real
	SINH (X)	SINH (X)	default real
#	SNGL (A)	REAL (A)	double precision
	SQRT (X)	SQRT (X)	default real
	TAN (X)	TAN (X)	default real
	TANH (X)	TANH (X)	default real

Functions marked with # cannot be used as an actual argument.
"double precision" means double-precision real.

2.2 Fortran 2000 Module Routines

The Fortran 2000 draft standard provides a set of intrinsic modules that define features to support IEEE arithmetic and interoperability with the C language. These modules define new functions and subroutines, and are implemented in the Sun Studio 8 Fortran 95 compiler.

2.2.1 IEEE Arithmetic and Exceptions Modules

The Fortran 2000 draft standard intrinsic modules `IEEE_EXCEPTIONS`, `IEEE_ARITHMETIC`, and `IEEE_FEATURES` to support new features in the proposed language standard to support IEEE arithmetic and IEEE exception handling.

The draft standard defines a set of inquiry functions, elemental functions, kind functions, elemental subroutines, and nonelemental subroutines. These are listed in the tables that follow.

To access these functions and subroutines, the calling routine must include

```
USE, INTRINSIC :: IEEE_ARITHMETIC, IEEE_EXCEPTIONS
```

See Chapter 14 of the draft standard (http://www.j3-fortran.org) for details.

2.2.1.1 Inquiry Functions

The module `IEEE_EXCEPTIONS` contains the following inquiry functions.

Function	Descripton
`IEEE_SUPPORT_FLAG(FLAG[,X])`	Inquire whether the processor supports an exception.
`IEEE_SUPPORT_HALTING(FLAG)`	Inquire whether the processor supports control of halting after an exception.

The module `IEEE_ARITHMETIC` contains the following inquiry functions.

Function	Description
`IEEE_SUPPORT_DATATYPE([X])`	Inquire whether the processor supports IEEE arithmetic.
`IEEE_SUPPORT_DENORMAL([X])`	Inquire whether the processor supports denormalized numbers.
`IEEE_SUPPORT_DIVIDE([X])`	Inquire whether the processor supports divide with the accuracy specified by the IEEE standard.
`IEEE_SUPPORT_INF([X])`	Inquire whether the processor supports the IEEE infinity.

IEEE_SUPPORT_IO([X])	Inquire whether the processor supports IEEE base conversion rounding during formatted input/output.
IEEE_SUPPORT_NAN([X])	Inquire whether the processor supports the IEEE Not-a-Number.
IEEE_SUPPORT_ROUNDING(VAL[,X])	Inquire whether the processor supports a particular rounding mode.
IEEE_SUPPORT_SQRT([X])	Inquire whether the processor supports the IEEE square root.
IEEE_SUPPORT_STANDARD([X])	Inquire whether the processor supports all IEEE facilities.

2.2.1.2 Elemental Functions

The module IEEE_ARITHMETIC contains the following elemental functions for real X and Y for which IEEE_SUPPORT_DATATYPE(X) and IEEE_SUPPORT_DATATYPE(Y) are true.

Function	Description
IEEE_CLASS(X)	IEEE class.
IEEE_COPY_SIGN(X,Y)	IEEE copysign function.
IEEE_IS_FINITE(X)	Determine if value is finite.
IEEE_IS_NAN(X)	Determine if value is IEEE Not-a-Number

IEEE_IS_NORMAL(X)	Determine if a value is normal.
IEEE_IS_NEGATIVE(X)	Determine if value is negative.
IEEE_LOGB(X)	Unbiased exponent in the IEEE floating point format.
IEEE_NEXT_AFTER(X,Y)	Returns the next representable neighbor of X in the direction toward Y.
IEEE_REM(X,Y)	The IEEE REM remainder function, X - Y*N where N is the integer nearest to the exact value of X/Y.
IEEE_RINT(X)	Round to an integer value according to the current rounding mode.
IEEE_SCALB(X,I)	Returns X*2**I
IEEE_UNORDERED(X,Y)	IEEE unordered function. True if X or Y is a NaN and false otherwise.
IEEE_VALUE(X,CLASS)	Generate an IEEE value.

2.2.1.3 Kind Function

The module IEEE_ARITHMETIC contains the following transformational function:

Function	Description
IEEE_SELECTED_REAL_KIND([P,][R])	Kind type parameter value fo an IEEE real with given precision and range.

2.2.1.4 Elemental Subroutines

The module `IEEE_EXCEPTIONS` contains the following elemental subroutines.

Subroutine	Description
`IEEE_GET_FLAG(FLAG,FLAG_VALUE)`	Get an exception flag.
`IEEE_GET_HALTING_MODE(FLAG,HALTING)`	GEt halting mode for an exception.

2.2.1.5 Nonelemental Subroutines

The module `IEEE_EXCEPTIONS` contains the following nonelemental subroutines.

Subroutine	Description
`IEEE_GET_STATUS(STATUS_VALUE)`	Get the current state of the floating point environment.
`IEEE_SET_FLAG(FLAG,FLAG_VALUE)`	Set an exception flag.
`IEEE_SET_HALTING_MODE(FLAG,HALTING)`	Controls continuation or halting on exceptions.
`IEEE_SET_STATUS(STATUS_VALUE)`	Restore the state of the floating point environment.

The module `IEEE_ARITHMETIC` contains the following nonelemental subroutines.

Subroutine	Description
`IEEE_GET_ROUNDING_MODE(ROUND_VAL)`	Get the current IEEE rounding mode.
`IEEE_SET_ROUNDING_MODE(ROUND_VAL)`	Set the current IEEE rounding mode.

2.2.2 C Binding Module

The Fortran 2000 draft standard provides a means of referencing C language procedures. The `ISO_C_BINDING` module defines three support procedures as intrinsic module functions. Accessing these functions requires

```
USE, INTRINSIC :: ISO_C_BINDING, ONLY: C_LOC, C_PTR, C_ASSOCIATED
```

in the calling routine. The procedures defined in the module are

Function	Description
`C_LOC(X)`	Returns the C address of the argument
`C_ASSOCIATED(C_PTR_1 [, C_PTR_2])`	Indicates the association status of `C_PTR_1` or indicates whether `C_PTR_1` and `C_PTR_2` are associated with the same entity.
`C_F_POINTER(CPTR, FPTR [, SHAPE])`	Associates a pointer with the target of a C pointer and specifies its shape.

For details on the `ISO_C_BINDING` intrinsic module, see Chapter 15 of the Fortran 2000 draft standard at http://www.j3-fortran.org/.

2.3 Non-Standard Fortran 95 Intrinsic Functions

The following functions are considered intrinsics by the `f95` compiler, but are not part of the Fortran 95 standard.

2.3.1 Basic Linear Algebra Functions (BLAS)

When compiling with `-xknown_lib=blas`, the compiler will recognize calls to the following routines as intrinsics and will optimize for and link to the Sun Performance Library implementation. The compiler will ignore user-supplied versions of these routines.

TABLE 2-2 BLAS Intrinsics

Function	Description

`CAXPY`
`DAXPY`
`SAXPY`
`ZAXPY`
	Product of a scalar and a vector plus a vector

`CCOPY`
`DCOPY`
`SCOPY`
`ZCOPY`
	Copy a vector

`CDOTC`
`CDOTU`
`DDOT`
`SDOT`
`ZDOTC`
	Dot product (inner product)

ZDOTU	
CSCAL DSCAL SSCAL ZSCAL	Scale a vector

See the *Sun Performance Library User's Guide* for more information on these routines.

2.3.2 Interval Arithmetic Intrinsic Functions

The following table lists intrinsic functions that are recognized by the compiler when compiling for interval arithmetic (`-xia`). For details, see the *Fortran 95 Interval Arithmetic Programming Reference*.

DINTERVAL	DIVIX	INF	INTERVAL
ISEMPTY	MAG	MID	MIG
NDIGITS	QINTERVAL	SINTERVAL	SUP
VDABS	VDACOS	VDASIN	VDATAN
VDATAN2	VDCEILING	VDCOS	VDCOSH
VDEXP	VDFLOOR	VDINF	VDINT
VDISEMPTY	VDLOG	VDLOG10	VDMAG
VDMID	VDMIG	VDMOD	VDNINT

VDSIGN	VDSIN	VDSINH	VDSQRT
VDSUP	VDTAN	VDTANH	VDWID
VQABS	VQCEILING	VQFLOOR	VQINF
VQINT	VQISEMPTY	VQMAG	VQMID
VQMIG	VQNINT	VQSUP	VQWID
VSABS	VSACOS	VSASIN	VSATAN
VSATAN2	VSCEILING	VSCOS	VSCOSH
VSEXP	VSFLOOR	VSINF	VSINT
VSISEMPTY	VSLOG	VSLOG10	VSMAG
VSMID	VSMIG	VSMOD	VSNINT
VSSIGN	VSSIN	VSSINH	VSSQRT
VSSUP	VSTAN	VSTANH	VSWID
WID			

2.3.3 Other Vendor Intrinsic Functions

The `f95` compiler recognizes a variety of legacy intrinsic functions that were defined by Fortran compilers from other vendors, including Cray Research, Inc. These are obsolete and their use should be avoided.

TABLE 2-3 Intrinsic Functions From Cray CF90 and Other Compilers

Function	Arguments	Description
`CLOC`	([C=]c)	Obtains the address of a character object
`COMPL`	([I=]i)	Bit-by-bit complement of a word. Use NOT(i) instead
`COT`	([X=]x)	Generic cotangent. (Also: DCOT, QCOT)
`CSMG`	([I=]i,[J=]j, [K=]k)	Conditional Scalar Merge
`DSHIFTL`	([I=]i,[J=]j, [K=]k)	Double-object left shift of i and j by k bits
`DSHIFTR`	([I=]i,[J=]j, [K=]k)	Double-object right shift of i and j by k bits
`EQV`	([I=]i,[J=]j)	Logical equivalence. Use IOER(i,j) instead.
`FCD`	([I=]i,[J=]j)	Constructs a character pointer
`GETPOS`	([I=]i)	Obtains file position
`IBCHNG`	([I=]i, [POS=]j)	Generic function to change specified bit in a word.

ISHA	([I=]i, [SHIFT=]j)	Generic arithmetic shift
ISHC	([I=]i, [SHIFT=]j)	Generic circular shift
ISHL	([I=]i, [SHIFT=]j)	Generic left shift
LEADZ	([I=]i)	Counts number of leading 0 bits
LENGTH	([I=]i)	Returns the number of Cray words successfully transferred
LOC	([I=]i)	Returns the address of a variable (See Section 1.4.32, loc: Return the Address of an Object)
NEQV	([I=]i,[J=]j)	Logical non-equivalence. Use IOER(i,j) instead.
POPCNT	([I=]i)	Counts number of bits set to 1
POPPAR	([I=]i)	Computes bit popluation parity
SHIFT	([I=]i,[J=]j)	Shift left circular. Use ISHFT(i,j) or ISHFTC(i,j,k) instead.
SHIFTA	([I=]i,[J=]j)	Arithmetic shift with sign extension.
SHIFTL	([I=]i,[J=]j)	Shift left with zero fill. Use ISHFT(i,j) or ISHFTC(i,j,k) instead.
SHIFTR	([I=]i,[J=]j)	Shift right with zero fill. Use ISHFT(i,j) or ISHFTC(i,j,k) instead.
TIMEF	()	Returns elapsed time since the first call

`UNIT`	([I=]i)	Returns status of BUFFERIN or BUFFEROUT
`XOR`	([I=]i,[J=]j)	Logical exclusive OR. Use IOER(i,j) instead.

See also Chapter 3 for a list of VMS Fortran 77 intrinsics.

2.3.4 Other Extensions

The Fortran 95 compiler recognizes the following additional intrinsic functions:

2.3.4.1 MPI_SIZEOF

`MPI_SIZEOF(` *x*`,` *size*`,` *error*`)` Returns the size in bytes of the machine representation of the given variable, *x*. If *x* is an array, it returns the size of the base element and not the size of the whole array	
x	input; variable or array of arbitrary type
size	output; integer; size in bytes of *x*
error	output; integer; set to an error code if an error detected, zero otherwise

2.3.4.2 Memory Functions

Memory allocation, reallocation, and deallocation functions `malloc()`, `realloc()`, and `free()` are implemented as `f95` intrinsics. See Section 1.4.35, malloc, malloc64, realloc, free: Allocate/Reallocate/Deallocate Memory for details.