Tables from *The* Designer's *Guide to VHDL*, 3rd *Edition* by Peter J. Ashenden

Morgan Kaufmann Publishers (an imprint of Elsevier)

TABLE 2.2 *VHDL operators in order of precedence, from most binding to least binding*

Operator	*Operation*	*Left operand type*	*Right operand type*	*Result type*
**	exponentiation	integer or floating-point	integer	same as left operand
abs	absolute value		numeric	same as operand
not	logical negation		boolean, bit, std_ulogic, 1-D array of boolean or bit, std_ulogic_vector	same as operand
and	logical and reduction		1-D array of boolean or bit, std_ulogic_vector	element type of operand
or	logical or reduction			
nand	negated logical and reduction			
nor	negated logical or reduction			
xor	exclusive or reduction			
xnor	negated exclusive or reduction			
*	multiplication	integer or floating-point	same as left operand	same as operands
		physical	integer or real	same as left operand
		integer or real	physical	same as right operand
/	division	integer or floating-point	same as left operand	same as operands
		physical	integer or real	same as left operand
		physical	same as left operand	universal integer
mod	modulo	integer or physical	same as left operand	same as operands
rem	remainder			
+	identity		numeric	same as operand
–	negation			
+	addition	numeric	same as left operand	same as operands
–	subtraction			
&	concatenation	1-D array	same as left operand	same as operands
		1-D array	element type of left operand	same as left operand
		element type of right operand	1-D array	same as right operand
		element type of result	element type of result	1-D array

Operator	*Operation*	*Left operand type*	*Right operand type*	*Result type*
sll **srl** **rol** **ror**	shift-left logical shift-right logical rotate left rotate right	1-D array of boolean or bit, std_ulogic_vector	integer	same as left operand
sla **sra**	shift-left arithmetic shift-right arithmetic	1-D array of boolean or bit	integer	same as left operand
= /=	equality inequality	any except file or protected type	same as left operand	boolean
< <= > >=	less than less than or equal to greater than greater than or equal to	scalar or 1-D array of any discrete type	same as left operand	boolean
?= ?/=	matching equality matching inequality	bit, std_ulogic or 1-D array of bit or std_ulogic	same as left operand	bit or std_ulogic
?< ?<= ?> ?>=	matching less than matching less than or equal to matching greater than matching greater than or equal to	bit or std_ulogic	same as left operand	bit or std_ulogic
and **or** **nand** **nor** **xor** **xnor**	logical and logical or negated logical and negated logical or exclusive or negated exclusive or	boolean, bit, std_ulogic, 1-D array of boolean or bit, std_ulogic_vector	same as left operand	same as operands
??	condition conversion		bit or std_ulogic	boolean

TABLE 9.1 *Constants defined in the package* math_real

Constant	*Value*	*Constant*	*Value*
math_e	e	math_log_of_2	$\ln 2$
math_1_over_e	$1/e$	math_log_of_10	$\ln 10$
math_pi	π	math_log2_of_e	$\log_2 e$
math_2_pi	2π	math_log10_of_e	$\log_{10} e$
math_1_over_pi	$1/\pi$	math_sqrt_2	$\sqrt{2}$
math_pi_over_2	$\pi/2$	math_1_over_sqrt_2	$1/\sqrt{2}$
math_pi_over_3	$\pi/3$	math_sqrt_pi	$\sqrt{\pi}$
math_pi_over_4	$\pi/4$	math_deg_to_rad	$2\pi/360$
math_3_pi_over_2	$3\pi/2$	math_rad_to_deg	$360/2\pi$

TABLE 9.2 *Functions defined in the package* math_real

Function	*Meaning*	*Function*	*Meaning*
ceil(x)	Ceiling of x (least integer $\geq x$)	sign(x)	Sign of x (–1.0, 0.0 or +1.0)
floor(x)	Floor of x (greatest integer $\leq x$)	"mod"(x, y)	Floating-point modulus of x / y
round(x)	x rounded to nearest integer value (ties rounded away from 0.0)	realmax(x, y)	Greater of x and y
trunc(x)	x truncated toward 0.0	realmin(x, y)	Lesser of x and y
sqrt(x)	$\sqrt{x}$	log(x)	$\ln x$
cbrt(x)	$\sqrt[3]{x}$	log2(x)	$\log_2 x$
"**"(n, y)	n^y	log10(x)	$\log_{10} x$
"**"(x, y)	x^y	log(x, y)	$\log_y x$
exp(x)	e^x		
sin(x)	sin x (x in radians)	arcsin(x)	arcsin x
cos(x)	cos x (x in radians)	arccos(x)	arccos x
tan(x)	tan x (x in radians)	arctan(x)	arctan x
		arctan(y, x)	arctan of point (x, y)
sinh(x)	sinh x	arcsinh(x)	arcsinh x
cosh(x)	cosh x	arccosh(x)	arccosh x
tanh(x)	tanh x	arctanh(x)	arctanh x

TABLE 9.3 *Overloaded operators defined in* math_complex

Operator	*Operation*		*Left operand*	*Right operand*	*Result*
=	equality		complex_polar	complex_polar	boolean
/=	inequality		complex_polar	complex_polar	boolean
abs	magnitude			complex	positive_real
				complex_polar	positive_real
–	negation			complex	complex
				complex_polar	complex_polar
+	addition	{	complex	complex	complex
–	subtraction		real	complex	complex
*	multiplication		complex	real	complex
/	division		complex_polar	complex_polar	complex_polar
			real	complex_polar	complex_polar
			complex_polar	real	complex_polar

TABLE 9.4 *Functions defined in the package* math_complex

Function	*Result type*	*Meaning*
cmplx(x, y)	complex	$x + jy$
get_principal_value(x)	principal_value	$x + 2\pi k$ for some k, such that $-\pi < \text{result} \leq \pi$
complex_to_polar(c)	complex_polar	c in polar form
polar_to_complex(p)	complex	p in Cartesian form
arg(z)	principal_value	$\arg(z)$ in radians
conj(z)	same as z	complex conjugate of z
sqrt(z)	same as z	$\sqrt{z}$
exp(z)	same as z	e^z
log(z)	same as z	$\ln z$
log2(z)	same as z	$\log_2 z$
log10(z)	same as z	$\log_{10} z$
log(z, y)	same as z	$\log_y z$
sin(z)	same as z	$\sin z$
cos(z)	same as z	$\cos z$
sinh(z)	same as z	$\sinh z$
cosh(z)	same as z	$\cosh z$

TABLE 9.5 *Operators defined in the IEEE standard synthesis packages*

Operator	*Operation*	*Left operand*	*Right operand*	*Result*
abs	absolute value		signed	signed
–	negation			
+	addition	unsigned	unsigned	unsigned
–	subtraction	unsigned	natural	unsigned
*	multiplication	natural	unsigned	unsigned
/	division	signed	signed	signed
rem	remainder	signed	integer	signed
mod	modulo	integer	signed	signed
+	addition	unsigned	*element type*	unsigned
–	subtraction	*element type*	unsigned	unsigned
		signed	*element type*	signed
		element type	signed	signed
=	equality	unsigned	unsigned	boolean
/=	inequality	unsigned	natural	boolean
<	less than	natural	unsigned	boolean
<=	less than or equal to	signed	signed	boolean
>	greater than	signed	integer	boolean
>=	greater than or equal to	integer	signed	boolean
?=	matching equality	unsigned	unsigned	*element type*
?/=	matching inequality	unsigned	natural	*element type*
?<	matching less than	natural	unsigned	*element type*
?<=	matching less than or equal to	signed	signed	*element type*
?>	matching greater than	signed	integer	*element type*
?>=	matching greater than or equal to	integer	signed	*element type*
sll	shift-left logical	unsigned	integer	unsigned
srl	shift-right logical	signed	integer	signed
sla	shift-left arithmetic			
sra	shift-right arithmetic			
rol	rotate left			
ror	rotate right			

Operator	*Operation*		*Left operand*	*Right operand*	*Result*
not	negation			unsigned	unsigned
				signed	signed
and	logical and		unsigned	unsigned	unsigned
or	logical or		unsigned	*element type*	unsigned
nand	negated logical and		*element type*	unsigned	unsigned
nor	negated logical or		signed	signed	signed
xor	exclusive or		signed	*element type*	signed
xnor	negated exclusive or		*element type*	signed	signed
and	logical and reduction			unsigned	*element type*
or	logical or reduction			signed	*element type*
nand	negated logical and reduction				
nor	negated logical or reduction				
xor	exclusive or reduction				
xnor	negated exclusive or reduction				

TABLE 9.6 *Functions defined in the IEEE standard synthesis packages*

Function		*First parameter*	*Second parameter*	*Result*
minimum		unsigned	unsigned	unsigned
maximum		unsigned	natural	unsigned
		natural	unsigned	unsigned
		signed	signed	signed
		signed	integer	signed
		integer	signed	signed
shift_left		unsigned	natural	unsigned
shift_right		signed	natural	signed
rotate_left				
rotate_right				
find_leftmost		unsigned	*element type*	integer
find_rightmost		signed	*element type*	integer
resize		unsigned	natural	unsigned
		unsigned	unsigned	unsigned
		signed	natural	signed
		signed	signed	signed
to_integer		unsigned		natural
		signed		integer
to_unsigned		natural	natural	unsigned
		natural	unsigned	unsigned
to_signed		integer	natural	signed
		integer	signed	signed
rising_edge[a]		**signal** bit		boolean
falling_edge[a]				
std_match[b]		unsigned	unsigned	boolean
		signed	signed	boolean
		std_ulogic	std_ulogic	boolean
		std_ulogic_vector	std_ulogic_vector	boolean

Function		*First parameter*	*Second parameter*	*Result*
to_01[b] to_X01[b] to_X01Z[b] to_UX01[b]	{	unsigned signed	⟦ std_ulogic ⟧ ⟦ std_ulogic ⟧	unsigned signed
to_string to_bstring to_binary_string to_ostring to_octal_string to_hstring to_hex_string	{	unsigned signed		string string

a. Provided in numeric_bit only.
b. Provided in numeric_std only.

TABLE 9.7 *Operand and result types*

Operators	*Left*	*Right*	*Result*
Binary **and**, **or**, **nand**, **nor**, **xor**, **xnor**	std_ulogic	std_ulogic	std_ulogic
	LogicArrayType	*LogicArrayType*	*LogicArrayType*
	LogicArrayType	std_ulogic	*LogicArrayType*
	std_ulogic	*LogicArrayType*	*LogicArrayType*
not		std_ulogic	std_ulogic
		LogicArrayType	*LogicArrayType*
Unary reduction **and**, **or**, **nand**, **nor**, **xor**, **xnor**		*LogicArrayType*	std_ulogic
=, /=, <, <=, >, >=	*NumericArrayType*	*NumericArrayType*	boolean
	NumericArrayType	integer	boolean
	integer	*NumericArrayType*	boolean
	RealArrayType	real	boolean
	real	*RealArrayType*	boolean
?=, ?/=, ?<, ?<=, ?>, ?>=	*NumericArrayType*	*NumericArrayType*	*ArrayElementType*
	NumericArrayType	integer	*ArrayElementType*
	integer	*NumericArrayType*	*ArrayElementType*
	RealArrayType	real	*ArrayElementType*
	real	*RealArrayType*	*ArrayElementType*
rol, **ror**, **sll**, **srl**	*LogicArrayType*	integer	*LogicArrayType*
sla, **sra**	*NumericArrayType*	integer	*NumericArrayType*
Binary +, –, *, /, **mod**, **rem**	*NumericArrayType*	*NumericArrayType*	*NumericArrayType*
	NumericArrayType	integer	*NumericArrayType*
	integer	*NumericArrayType*	*NumericArrayType*
	RealArrayType	real	*RealArrayType*
	real	*RealArrayType*	*RealArrayType*
Binary +, –	*NumericArrayType*	std_ulogic	*NumericArrayType*
	std_ulogic	*NumericArrayType*	*NumericArrayType*

Operators	*Left*	*Right*	*Result*
Unary –, **abs**		signed, sfixed, float	signed, sfixed, float
maximum, minimum	*NumericArrayType*	*NumericArrayType*	*NumericArrayType*
	NumericArrayType	integer	*NumericArrayType*
	integer	*NumericArrayType*	*NumericArrayType*
	RealArrayType	real	*RealArrayType*
	real	*RealArrayType*	*RealArrayType*

TABLE 9.8 *Result sizes and index ranges*

Operator	*Result type*	*Result size and/or range*
Array/array **and**, **or**, **nand**, **nor**, **xor**, **xnor**	*ArrayOfBits*	*Result*'length = *L*'length = *R*'length Fixed, Float: *Result*'range = *L*'range
Array/scalar **and**, **or**, **nand**, **nor**, **xor**, **xnor**	*ArrayOfBits*	*Result*'length = *A*'length Fixed, Float: *Result*'range = *A*'range
not	*ArrayOfBits*	*Result*'length = *R*'length Fixed, Float: *Result*'range = *R*'range
rol, **ror**, **sll**, **srl**, **sla**, **sra**	*ArrayOfBits*	*Result*'length = *A*'length Fixed, Float: *Result*'range = *A*'range
+, –, *, /, **rem**, **mod**	float	maximum(*L*'left, *R*'left) down to minimum(*L*'right, *R*'right)
Binary +, –	unsigned, signed	maximum(*L*'length, *R*'length) – 1 down to 0
	ufixed, sfixed	maximum(*L*'left, *R*'left) + 1 down to minimum(*L*'right, *R*'right)
*	unsigned, signed	*L*'length + *R*'length – 1 down to 0
	ufixed, sfixed	*L*'left + *R*'left + 1 down to *L*'right + *R*'right
/	unsigned, signed	*L*'length – 1 down to 0
	ufixed	*L*'left – *R*'right down to *L*'right – *R*'left – 1
	sfixed	*L*'left – *R*'right + 1 down to *L*'right – *R*'left
rem	unsigned, signed	*R*'length – 1 down to 0
	ufixed, sfixed	minimum(*L*'left, *R*'left) down to minimum(*L*'right, *R*'right)
mod	unsigned, signed	*R*'length – 1 down to 0
	ufixed	minimum(*L*'left, *R*'left) down to minimum(*L*'right, *R*'right)
	sfixed	*R*'left down to minimum(*L*'right, *R*'right)
Unary –, **abs**	signed	*R*'length – 1 down to 0
	sfixed	*R*'left + 1 down to *R*'right
minimum, maximum	*DiscreteArrayType*	*Result*'length = *A*'length Fixed, Float: *Result*'range = *A*'range
	unsigned, signed	maximum(*L*'length, *R*'length) – 1 down to 0
	ufixed, sfixed, float	minimum(*L*'left, *R*'left) down to minimum(*L*'right, *R*'right)

TABLE 9.9 *Conversions between bit and standard-logic types*

Function	*Result type*	*Parameter 1 type*	*Parameter 2*	*Package*
to_bit	bit	std_ulogic	xmap	1164
to_std_ulogic	std_ulogic	bit		1164
to_bv	bit_vector	sulv	xmap	1164
		natural	size	nbu
		natural	size_res	nbu
to_sulv	sulv	bv		1164
		slv		1164
		natural	size	nsu
		natural	size_res	nsu
		ufixed		fixed
		sfixed		fixed
		float		float
to_slv	slv	bv		1164
		sulv		1164
		natural	natural	nsu
		natural	size_res	nsu
		ufixed		fixed
		sfixed		fixed
		float		float

TABLE 9.10 *Conversion functions yielding* unsigned *and* signed *values*

Function	*Result type*	*Param 1 type*	*Param 2*	*Param 3*	*Param 4*	*Package*
to_unsigned	unsigned	natural	size			ns/b
			size_res			
		ufixed	size	overflow	round	fixed
			size_res	overflow	round	
		float	size	round	chk_err	float
			size_res	round	chk_err	
to_signed	signed	integer	size			ns/b
			size_res			
		sfixed	size	overflow	round	fixed
			size_res	overflow	round	
		float	size	round	chk_err	float
			size_res	round	chk_err	

TABLE 9.11 *Conversion functions yielding* ufixed *and* sfixed *values*

Function	*Result type*	*Param 1 type*	*Param 2*	*Param 3*	*Param 4*	*Param 5*	*Param 6*	*Param 7*	*Package*
to_ufixed	ufixed	sulv	L_index	R_index					fixed
			size_res						
		unsigned							
			L_index	R_index	overflow	round			
			size_res	overflow	round				
		natural	L_index	R_index	overflow	round			
			size_res	overflow	round				
		real	L_index	R_index	overflow	round	guard		
			size_res	overflow	round	guard			
		float	L_index	R_index	overflow	round	chk_err	denorm	float
			size_res	overflow	round	chk_err	denorm		

Function	*Result type*	*Param 1 type*	*Param 2*	*Param 3*	*Param 4*	*Param 5*	*Param 6*	*Param 7*	*Package*
to_sfixed	sfixed	ufixed							fixed
		sulv	L_index	R_index					
			size_res						
		signed							
			L_index	R_index	overflow	round			
			size_res	overflow	round				
		integer	L_index	R_index	overflow	round			
			size_res	overflow	round				
		real	L_index	R_index	overflow	round	guard		
			size_res	overflow	round	guard			
		float	L_index	R_index	overflow	round	chk_err	denorm	float
			size_res	overflow	round	chk_err	denorm		

TABLE 9.12 *Conversion functions yielding* float *values*

Function	*Result type*	*Param 1 type*	*Param 2*	*Param 3*	*Param 4*	*Param 5*	*Package*
to_float	float	sulv	exponent	fraction			float
			size_res				
		unsigned					
			exponent	fraction	round		
			size_res	round			
		signed	exponent	fraction	round		
			size_res	round			
		ufixed	exponent	fraction	round	denorm	
			size_res	round	denorm		
		sfixed	exponent	fraction	round	denorm	
			size_res	round	denorm		
		integer	exponent	fraction	round		
			size_res	round			
		real	exponent	fraction	round	denorm	
			size_res	round	denorm		

TABLE 9.13 *Conversion functions yielding* integer *and* real *values*

Function	*Result type*	*Param 1 type*	*Param 2*	*Param 3*	*Param 4*	*Package*
to_integer	natural	bv				nbu
	natural	sulv				nsu
	natural	unsigned				ns/b
	integer	signed				
	natural	ufixed	overflow	round		fixed
	integer	sfixed	overflow	round		
	integer	float	round	chk_err		float
to_real	real	ufixed				fixed
		sfixed				
		float	round	chk_err	denorm	float

TABLE 9.14 *Resizing functions*

Function	*Result type*	*Param 1 type*	*Param 2*	*Param 3*	*Param 4*	*Param 5*	*Param 6*	*Param 7*	*Package*
resize	bv	bv	new_size						nbu
			size_res						
	sulv	sulv	new_size						nsu
			size_res						
	unsigned	unsigned	new_size						ns/b
			size_res						
	signed	signed	new_size						
			size_res						
	ufixed	ufixed	L_index	R_index	overflow	round			fixed
			size_res	overflow	round				
	sfixed	sfixed	L_index	R_index	overflow	round			
			size_res	overflow	round				
	float	float	exponent	fraction	round	chk_err	den_in	den_out	float
			size_res	round	chk_err	den_in	den_out		

TABLE 9.15 *Strength reduction mappings*

Function	*'U'*	*'X'*	*'0'*	*'1'*	*'Z'*	*'W'*	*'L'*	*'H'*	*'–'*
to_01	xmap	xmap	'0'	'1'	xmap	xmap	'0'	'1'	xmap
to_X01	'X'	'X'	'0'	'1'	'X'	'X'	'0'	'1'	'X'
to_X01Z	'X'	'X'	'0'	'1'	'Z'	'X'	'0'	'1'	'X'
to_UX01	'U'	'X'	'0'	'1'	'X'	'X'	'0'	'1'	'X'

TABLE 20.1 *The predefined attributes giving information about values in a type*

Attribute	*Type of* T	*Result type*	*Result*
T'left	Any scalar type or subtype	Same as T	Leftmost value in T
T'right	"	"	Rightmost value in T
T'low	"	"	Least value in T
T'high	"	"	Greatest value in T
T'ascending	"	boolean	True if T is an ascending range, false otherwise
T'image(x)	"	string	A textual representation of the value x of type T
T'value(s)	"	base type of T	Value in T represented by the string s
T'pos(s)	Any discrete or physical type or subtype	universal integer	Position number of x in T
T'val(x)	"	Base type of T	Value at position x in T
T'succ(x)	"	"	Value at position one greater than x in T
T'pred(x)	"	"	Value at position one less than x in T
T'leftof(x)	"	"	Value at position one to the left of x in T
T'rightof(x)	"	"	Value at position one to the right of x in T

TABLE 20.2 *The predefined attributes giving information about the index range of an array*

Attribute	*Result*
A'left(n)	Leftmost value in index range of dimension n
A'right(n)	Rightmost value in index range of dimension n
A'low(n)	Least value in index range of dimension n
A'high(n)	Greatest value in index range of dimension n
A'range(n)	Index range of dimension n
A'reverse_range(n)	Index range of dimension n reversed in direction and bounds
A'length(n)	Length of index range of dimension n
A'ascending(n)	True if index range of dimension n is ascending, false otherwise

TABLE 20.3 *The predefined attributes giving type information*

Attribute	*Prefix*	*Result*
T'base	Any type or subtype	The base type of T, for use only as prefix of another attribute
O'subtype	Any object or alias of an object	The fully constrained subtype of O, including constraints defining index ranges (if O is an array or has elements that are arrays)
A'element	Any array type, subtype, or object	If A is an array type or subtype, the element subtype of A. If A is an arrray object, the element subtype of A including constraints defining all index ranges

TABLE 20.4 *The predefined attributes giving information about signals and values of signals*

Attribute	*Result type*	*Result*
S'delayed(t)	base type of S	Implicit signal, with the same value as S, but delayed by t time units (t ≥ 0 ns)
S'stable(t)	boolean	Implicit signal, true when no event has occurred on S for t time units, false otherwise (t ≥ 0 ns)
S'quiet(t)	boolean	Implicit signal, true when no transaction has occurred on S for t time units, false otherwise (t ≥ 0 ns)
S'transaction	bit	Implicit signal, changes value in simulation cycles in which a transaction occurs on S
S'event	boolean	True if an event has occurred on S in the current simulation cycle, false otherwise
S'active	boolean	True if a transaction has occurred on S in the current simulation cycle, false otherwise
S'last_event	time	Time since last event occurred on S, or time'high if no event has yet occurred
S'last_active	time	Time since last transaction occurred on S, or time'high if no transaction has yet occurred
S'last_value	base type of S	Value of S before last event occurred on it
S'driving	boolean	True if the containing process is driving S (or every element of a composite signal S), or false if the containing process has disconnected its driver for S (or any element of S) with a null transaction
S'driving_value	base type of S	Value contributed by driver for S in the containing process

TABLE 20.5 *The predefined attributes that provide names of declared items*

Attribute	*Result*
X'simple_name	A string representing the identifier, character or operator symbol defined in the declaration of the item X
X'path_name	A string describing the path through the elaborated design hierarchy, from the top-level entity or package to the item X
X'instance_name	A string similar to that produced by X'path_name, but including the names of the entity and architecture bound to each component instance in the path

TABLE 22.1 *The Gumnut instruction set*

Arithmetic and logical instructions	
add *rd*, *rs*, *op2*	Add *rs* and *op2*, result in *rd*
addc *rd*, *rs*, *op2*	Add *rs* and *op2* with carry, result in *rd*
sub *rd*, *rs*, *op2*	Subtract *op2* from *rs*, result in *rd*
subc *rd*, *rs*, *op2*	Subtract *op2* from *rs* with carry, result in *rd*
and *rd*, *rs*, *op2*	Logical AND of *rs* and *op2*, result in *rd*
or *rd*, *rs*, *op2*	Logical OR of *rs* and *op2*, result in *rd*
xor *rd*, *rs*, *op2*	Logical XOR of *rs* and *op2*, result in *rd*
mask *rd*, *rs*, *op2*	Logical AND of *rs* and NOT *op2*, result in *rd*
Shift instructions	
shl *rd*, *rs*, *count*	Shift *rs* value left *count* places, result in *rd*
shr *rd*, *rs*, *count*	Shift *rs* value right *count* places, result in *rd*
rol *rd*, *rs*, *count*	Rotate *rs* value left *count* places, result in *rd*
ror *rd*, *rs*, *count*	Rotate *rs* value right *count* places, result in *rd*
Memory and I/O instructions	
ldm *rd*, (*rs*)±*offset*	Load to *rd* from memory
stm *rd*, (*rs*)±*offset*	Store to memory from *rd*
inp *rd*, (*rs*)±*offset*	Input to *rd* from input controller register
out *rd*, (*rs*)±*offset*	Output to output controller register from *rd*
Branch instructions	
bz ±*disp*	Branch if Z is set
bnz ±*disp*	Branch is Z is not set
bc ±*disp*	Branch if C is set
bnc ±*disp*	Branch if C is not set
Jump instructions	
jmp *addr*	Jump to *addr*
jsb *addr*	Jump to subroutine at *addr*
Miscellaneous instructions	

ret	Return from subroutine
reti	Return from interrupt
enai	Enable interrupts
disi	Disable interrupts
wait	Wait for interrupts
stby	Enter low-power standby mode

TABLE 22.2 *Function code values*

add	000	addc	001	sub	010	subc	011
and	100	or	101	xor	110	mask	111
shl	00	shr	01	rol	10	ror	11
ldm	00	stm	01	inp	10	out	11
bz	00	bnz	01	bc	10	bnc	11
jmp	0	jsb	1				
ret	000	reti	001	enai	010	disi	011
wait	100	stby	101				

TABLE 23.1 *Strings for specifying ciphers*

String	*Cipher*	*Cipher type*
"des-cbc"	DES in CBC mode.	Symmetric
"3des-cbc"	Triple DES in CBC mode.	Symmetric
"aes128-cbc"	AES in CBC mode with 128-bit key.	Symmetric
"aes192-cbc"	AES in CBC mode with 192-bit key.	Symmetric
"aes256-cbc"	AES in CBC mode with 256-bit key.	Symmetric
"blowfish-cbc"	Blowfish in CBC mode.	Symmetric
"twofish128-cbc"	Twofish in CBC mode with 128-bit key.	Symmetric
"twofish192-cbc"	Twofish in CBC mode with 192-bit key.	Symmetric
"twofish256-cbc"	Twofish in CBC mode with 256-bit key.	Symmetric
"serpent128-cbc"	Serpent in CBC mode with 128-bit key.	Symmetric
"serpent192-cbc"	Serpent in CBC mode with 192-bit key.	Symmetric
"serpent256-cbc"	Serpent in CBC mode with 256-bit key.	Symmetric
"cast128-cbc"	CAST-128 in CBC mode.	Symmetric
"rsa"	RSA.	Asymmetric
"elgamal"	ElGamal.	Asymmetric
"pgp-rsa"	OpenPGP RSA key.	Asymmetric

TABLE 23.2 *Strings for specifying encodings*

String	*Encoding methods*
"uuencode"	IEEE Std 1003.1™-2001 (uuencode Historical Algorithm)
"base64"	IETF RFC 2045, also IEEE Std 1003.1 (uuencode -m)
"quoted-printable"	IETF RFC 2045
"raw"	Identity transformation; no encoding is performed, and the data may contain non-printing characters.

TABLE 23.3 *Strings for specifying hash functions*

Digest method string	*Required/optional*	*Hash function*
"sha1"	Required	Secure Hash Algorithm 1 (SHA-1).
"md5"	Required	Message Digest Algorithm 5.
"md2"	Optional	Message Digest Algorithm 2.
"ripemd-160"	Optional	RIPEMD-160.