

SMEIL Language Reference

Grammar

$\langle module \rangle$	$::= \{ \langle import-stm \rangle \}$ $\{ \langle module-decl \rangle \}$ $\{ \langle entity \rangle \}$
$\langle import-stm \rangle$	$::= \text{'import' } \langle import-name \rangle [\langle qualified-specifier \rangle] \text{' ;'}$ $ \text{'from' } \langle import-name \rangle$ $\text{'import' } \langle ident \rangle \{ \text{' , ' } \langle ident \rangle \} [\langle qualified-specifier \rangle]$ ' ;'
$\langle import-name \rangle$	$::= \langle ident \rangle \{ \text{' . ' } \langle ident \rangle \}$
$\langle qualified-specifier \rangle$	$::= \text{'as' } \langle ident \rangle$
$\langle module-decl \rangle$	$::= \langle type-def \rangle$ $ \langle const-decl \rangle$ $ \langle enum-decl \rangle$ $ \langle func-decl \rangle$
$\langle type-def \rangle$	$::= \text{'type' } \langle ident \rangle \text{' : '}$ $\langle type-name \rangle \text{ (type alias)}$ $ \langle bus-signal-decls \rangle \text{ (bus definition)}$ ' ;'
$\langle entity \rangle$	$::= \langle network \rangle$ $ \langle process \rangle$
$\langle network \rangle$	$::= \text{'network' } \langle ident \rangle \text{' (' } [\langle params \rangle] \text{') '}$ $\text{' { ' } \{ \langle network-decl \rangle \} \text{' }'$
$\langle process \rangle$	$::= [\text{'clocked' }] \text{'proc' } \langle ident \rangle$ $\text{' (' } [\langle params \rangle] \text{') ' } \{ \langle process-decl \rangle \}$ $\text{' { ' } \{ \langle statement \rangle \} \text{' }'$
$\langle network-decl \rangle$	$::= \langle inst-decl \rangle$ $ \langle bus-decl \rangle$

	$\langle \text{const-decl} \rangle$ $\langle \text{gen-decl} \rangle$ $\langle \text{connect-decl} \rangle$
$\langle \text{process-decl} \rangle$	$::=$ $\langle \text{var-decl} \rangle$ $\langle \text{const-decl} \rangle$ $\langle \text{bus-decl} \rangle$ $\langle \text{enum-decl} \rangle$ $\langle \text{func-decl} \rangle$ $\langle \text{inst-decl} \rangle$ $\langle \text{gen-decl} \rangle$
$\langle \text{func-decls} \rangle$	$::=$ $\langle \text{var-decl} \rangle$ $\langle \text{const-decl} \rangle$ $\langle \text{enum-decl} \rangle$
$\langle \text{params} \rangle$	$::=$ $\langle \text{param} \rangle$ { , $\langle \text{param} \rangle$ }
$\langle \text{param} \rangle$	$::=$ [' [' [$\langle \text{integer} \rangle$] ']] $\langle \text{direction} \rangle$ $\langle \text{ident} \rangle$ [':' $\langle \text{type-name} \rangle$]
$\langle \text{direction} \rangle$	$::=$ 'in' (input signal) 'out' (output signal) 'const' (constant input value)
$\langle \text{var-decl} \rangle$	$::=$ 'var' $\langle \text{ident} \rangle$ ':' $\langle \text{type-name} \rangle$ ['=' $\langle \text{expression} \rangle$] [$\langle \text{range} \rangle$] ';'
$\langle \text{range} \rangle$	$::=$ 'range' $\langle \text{expression} \rangle$ 'to' $\langle \text{expression} \rangle$
$\langle \text{enum-decl} \rangle$	$::=$ 'enum' $\langle \text{ident} \rangle$ '{' $\langle \text{enum-field} \rangle$ { ',' $\langle \text{enum-field} \rangle$ } '}' ';'
$\langle \text{enum-field} \rangle$	$::=$ $\langle \text{ident} \rangle$ ['=' $\langle \text{integer} \rangle$]
$\langle \text{const-decl} \rangle$	$::=$ 'const' $\langle \text{ident} \rangle$ ':' $\langle \text{type-name} \rangle$ '=' $\langle \text{expression} \rangle$ ';'
$\langle \text{bus-decl} \rangle$	$::=$ ['clocked'] 'bus' $\langle \text{ident} \rangle$ $\langle \text{bus-decl-content} \rangle$ ';'
$\langle \text{func-decl} \rangle$	$::=$ 'function' $\langle \text{ident} \rangle$ '(' $\langle \text{params} \rangle$ ')' { $\langle \text{func-decls} \rangle$ } '{' { $\langle \text{statement} \rangle$ } '}'
$\langle \text{bus-decl-content} \rangle$	$::=$ '{' $\langle \text{bus-signal-decls} \rangle$ '}' $\langle \text{type-name} \rangle$
$\langle \text{bus-signal-decls} \rangle$	$::=$ $\langle \text{bus-signal-decl} \rangle$ { $\langle \text{bus-signal-decl} \rangle$ }

$\langle \text{bus-signal-decl} \rangle$	$::= \langle \text{ident} \rangle \text{'.'} \langle \text{type-name} \rangle [\text{'='} \langle \text{expression} \rangle] [\langle \text{range} \rangle] \text{';'}$
$\langle \text{connect-entry} \rangle$	$::= \langle \text{name} \rangle \text{'->'} \langle \text{name} \rangle$
$\langle \text{connect-decl} \rangle$	$::= \text{connect} \langle \text{connect-entry} \rangle \{ \text{' ,' } \langle \text{connect-entry} \rangle \} \text{' ;'}$
$\langle \text{inst-decl} \rangle$	$::= \text{'instance'} \langle \text{instance-name} \rangle \text{'of'} \langle \text{ident} \rangle \text{'('} [\langle \text{param-map} \rangle \{ \text{' ,' } \langle \text{param-map} \rangle \}] \text{')' ' ;'}$
$\langle \text{instance-name} \rangle$	$::= \langle \text{ident} \rangle \text{'['} \langle \text{expression} \rangle \text{']'}$ (indexed instance) $ \langle \text{ident} \rangle$ (named instance) $ \text{'_ '}$ (anonymous instance)
$\langle \text{param-map} \rangle$	$::= [\langle \text{ident} \rangle \text{'.'}] \langle \text{expression} \rangle$
$\langle \text{gen-decl} \rangle$	$::= \text{'generate'} \langle \text{ident} \rangle \text{'='} \langle \text{expression} \rangle \text{'to'} \langle \text{expression} \rangle \text{'{' } \{ \langle \text{network-decl} \rangle \} \text{'}'}$
$\langle \text{statement} \rangle$	$::= \langle \text{name} \rangle \text{'='} \langle \text{expression} \rangle \text{' ;'}$ (assignment) $ \langle \text{name} \rangle \text{'('} \langle \text{param-map} \rangle \text{')' ' ;'}$ (function call) $ \text{'if'} \text{'('} \langle \text{expression} \rangle \text{')' '{' } \{ \langle \text{statement} \rangle \} \text{'}'}$ $\{ \langle \text{elif-block} \rangle \} [\langle \text{else-block} \rangle]$ $ \text{'for'} \langle \text{ident} \rangle \text{'='} \langle \text{expression} \rangle \text{'to'} \langle \text{expression} \rangle \text{'{' } \{ \langle \text{statement} \rangle \} \text{'}'}$ $ \text{'switch'} \langle \text{simple-expression} \rangle \text{'{' } \{ \langle \text{switch-case} \rangle \{ \langle \text{switch-case} \rangle \} [\text{'default'} \text{'{' } \langle \text{statement} \rangle \} \text{'}' } \{ \langle \text{statement} \rangle \} \text{'}' } [\text{'}'] \text{'}'}$ $ \text{'trace'} \text{'('} \langle \text{format-string} \rangle \{ \text{' ,' } \langle \text{expression} \rangle \} \text{')' ' ;'}$ $ \text{'assert'} \text{'('} \langle \text{expression} \rangle [\text{' ,' } \langle \text{string-literal} \rangle] \text{')' ' ;'}$ $ \text{'break'} \text{' ;'}$
$\langle \text{switch-case} \rangle$	$::= \text{'case'} \langle \text{simple-expression} \rangle \text{'{' } \{ \langle \text{statement} \rangle \} \text{'}'}$
$\langle \text{elif-block} \rangle$	$::= \text{'elif'} \text{'('} \langle \text{expression} \rangle \text{')' '{' } \{ \langle \text{statement} \rangle \} \text{'}'}$
$\langle \text{else-block} \rangle$	$::= \text{'else'} \text{'{' } \{ \langle \text{statement} \rangle \} \text{'}'}$
$\langle \text{format-string} \rangle$	$::= \text{'"'} \{ \langle \text{format-string-part} \rangle \} \text{'" '}$
$\langle \text{format-string-part} \rangle$	$::= \text{'{'}}$ (placeholder string) $ \langle \text{string-char} \rangle$
$\langle \text{simple-expression} \rangle$	$::= \langle \text{literal} \rangle$ $ \langle \text{name} \rangle$

$\langle expression \rangle$	$::=$ $\langle simple-expression \rangle$ $ $ $\langle expression \rangle \langle bin-op \rangle \langle expression \rangle$ $ $ $\langle un-op \rangle \langle expression \rangle$ $ $ $\text{'('} \langle expression \rangle \text{'}'$ $ $ $\text{'('} \langle type-name \rangle \text{'}' \langle expression \rangle$ (type cast)
$\langle bin-op \rangle$	$::=$ '+' (addition) $ $ '-' (subtraction) $ $ $\text{'*}'$ (multiplication) $ $ '/' (division) $ $ '%' (modulo) $ $ '==' (equal) $ $ '!=' (not equal) $ $ '<<' (shift left) $ $ '>>' (shift right) $ $ '<' (less than) $ $ '>' (greater than) $ $ '>=' (greater than or equal) $ $ '<=' (less than or equal) $ $ '\&' (bitwise-and) $ $ ' ' (bitwise-or) $ $ '^' (bitwise-xor) $ $ '\&\&' (logical conjunction) $ $ ' ' (logical disjunction)
$\langle un-op \rangle$	$::=$ '-' (negation) $ $ '+' (identity) $ $ '!' (logical negation) $ $ '\sim' (bitwise-not)
$\langle literal \rangle$	$::=$ $\langle integer \rangle$ $ $ $\langle floating \rangle$ $ $ $\langle string-literal \rangle$ $ $ $\text{'['} \langle integer \rangle \{ \text{' ,' } \langle integer \rangle \} \text{'}'$ (Array literal) $ $ 'true' $ $ 'false' $ $ 'U' (Undefined value)
$\langle string-literal \rangle$	$::=$ $\text{'"'} \{ \langle string-char \rangle \} \text{'"}$
$\langle intrinsic-type \rangle$	$::=$ 'i' $\langle integer \rangle$ (signed integer) $ $ 'int' (arbitrary-width signed integer) $ $ 'u' $\langle integer \rangle$ (unsigned integer) $ $ 'uint' (arbitrary-width unsigned integer) $ $ 'float' (arbitrary-width floating point) $ $ 'f8' (8 bit floating point)

	'f16' (16 bit floating point) 'f32' (single-precision floating point) 'f64' (double-precision floating point) 'bool' (boolean value)
$\langle type-name \rangle$	$::= \langle intrinsic-type \rangle$ $\langle name \rangle$ (type definition) '[' [$\langle expression \rangle$] ']' $\langle type-name \rangle$ (array of type)
$\langle ident \rangle$	$::= \langle letter \rangle \{ \langle letter \rangle \mid \langle number \rangle \mid _ \mid - \}$ (identifier)
$\langle name \rangle$	$::= \langle ident \rangle$ $\langle name \rangle _ \langle name \rangle$ (hierarchical accessor) $\langle name \rangle _ '[' \langle array-index \rangle ']'$ (array element access)
$\langle array-index \rangle$	$::= _*$ (wildcard) $\langle expression \rangle$ (element index)
$\langle integer \rangle$	$::= \langle number \rangle \{ \langle number \rangle \}$ (decimal number) '0x' $\langle hex-digit \rangle \{ \langle hex-digit \rangle \}$ (hexadecimal number) '0o' $\langle octal-digit \rangle \{ \langle octal-digit \rangle \}$ (octal number)
$\langle floating \rangle$	$::= \{ \langle number \rangle \} _ _ \langle number \rangle \{ \langle number \rangle \}$
$\langle number \rangle$	$::= _0 _9$
$\langle letter \rangle$	$::= _a _z$ $_A _Z$
$\langle hex-digit \rangle$	$::= \langle number \rangle$ $_a _f$ $_A _F$
$\langle octal-digit \rangle$	$::= _0 _8$
$\langle string-char \rangle$	$::=$ (ISO-8859-1 char with value > 26)

Operator precedence

Precedence	Operators
0	+ - ! ~ (unary)
1	* / %
2	+ -
3	<< >>
4	< > <= >=
5	== !=
6	& ^
7	&&
8	

Keywords

- as
- async
- await
- barrier
- break
- bus
- case
- const
- connect
- clocked
- default
- elif
- else
- enum
- exposed
- for
- from
- function
- generate
- if
- import
- in
- instance
- network
- of
- out
- proc
- range
- return
- switch
- sync
- to
- unique
- var
- wait
- where