

# 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$ $\text{'import' } \langle ident \rangle \{ \text{' , ' } \langle ident \rangle \} [ \langle qualified-specifier \rangle ]$ $\text{' ;'}$
$\langle import-name \rangle$	$::= \langle ident \rangle \{ \text{' . ' } \langle ident \rangle \}$
$\langle qualified-specifier \rangle$	$::= \text{'as' } \langle ident \rangle$
$\langle module-decl \rangle$	$::= \langle type-def \rangle$ $  \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)}$ $\text{' ;'}$
$\langle entity \rangle$	$::= \langle network \rangle$ $  \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$

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

$\langle \text{bus-signal-decl} \rangle$	$::= \langle \text{ident} \rangle \text{'.'} \langle \text{type-name} \rangle [ \text{'='} \langle \text{expression} \rangle ] [ \langle \text{range} \rangle ] \text{';'}$
$\langle \text{connect-entry} \rangle$	$::= \langle \text{name} \rangle \text{'->'} \langle \text{name} \rangle$
$\langle \text{connect-decl} \rangle$	$::= \text{connect} \langle \text{connect-entry} \rangle \{ \text{' ,' } \langle \text{connect-entry} \rangle \} \text{';'}$
$\langle \text{inst-decl} \rangle$	$::= \text{'instance'} \langle \text{instance-name} \rangle \text{'of'} \langle \text{ident} \rangle \text{'('} [ \langle \text{param-map} \rangle \{ \text{' ,' } \langle \text{param-map} \rangle \} ] \text{' )' ' ;'}$
$\langle \text{instance-name} \rangle$	$::= \langle \text{ident} \rangle \text{'['} \langle \text{expression} \rangle \text{' ]'}$ (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{ident} \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 \} [ \text{'default'} \text{'{' } \langle \text{statement} \rangle \text{'}' } \{ \langle \text{statement} \rangle \} \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)
$\langle bin-op \rangle$	$::=$ $\text{'+'}$ (addition) $ $ $\text{'-'}$ (subtraction) $ $ $\text{'*}'$ (multiplication) $ $ $\text{'/'}$ (division) $ $ $\text{'%'}$ (modulo) $ $ $\text{'=='}$ (equal) $ $ $\text{'!='}$ (not equal) $ $ $\text{'<<'}$ (shift left) $ $ $\text{'>>'}$ (shift right) $ $ $\text{'<'}$ (less than) $ $ $\text{'>'}$ (greater than) $ $ $\text{'>='}$ (greater than or equal) $ $ $\text{'<='}$ (less than or equal) $ $ $\text{'\&'}$ (bitwise-and) $ $ $\text{' '}$ (bitwise-or) $ $ $\text{'^'}$ (bitwise-xor) $ $ $\text{'\&\&'}$ (logical conjunction) $ $ $\text{'  '}$ (logical disjunction)
$\langle un-op \rangle$	$::=$ $\text{'-'}$ (negation) $ $ $\text{'+'}$ (identity) $ $ $\text{'!'}$ (logical negation) $ $ $\text{'\sim'}$ (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) $ $ $\text{'true'}$ $ $ $\text{'false'}$ $ $ $\text{'U'}$ (Undefined value)
$\langle string-literal \rangle$	$::=$ $\text{' ' } \{ \langle string-char \rangle \} \text{' '}$
$\langle intrinsic-type \rangle$	$::=$ $\text{'i'}$ $\langle integer \rangle$ (signed integer) $ $ $\text{'int'}$ (arbitrary-width signed integer) $ $ $\text{'u'}$ $\langle integer \rangle$ (unsigned integer) $ $ $\text{'uint'}$ (arbitrary-width unsigned integer) $ $ $\text{'f32'}$ (single-precision floating point) $ $ $\text{'f64'}$ (double-precision floating point) $ $ $\text{'bool'}$ (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)
$\langle ident \rangle$	$::= \langle letter \rangle \{ \langle letter \rangle   \langle number \rangle   '_'   '-' \}$ (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	$+ - ! \sim$ (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
- func
- generate
- if
- import
- in
- instance
- network
- of
- out
- proc
- range
- return
- switch
- sync
- to
- unique
- var
- wait
- where