

SMEIL Language Reference

Grammar

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

$\langle params \rangle$	$::= \langle param \rangle \{ , \langle param \rangle \}$
$\langle param \rangle$	$::= [\text{'['} [\langle integer \rangle] \text{'}] \langle direction \rangle \langle ident \rangle [\text{'::'} \langle type \rangle]$
$\langle direction \rangle$	$::= \text{'in'}$ (input signal) 'out' (output signal) 'const' (constant input value)
$\langle var-decl \rangle$	$::= \text{'var'} \langle ident \rangle \text{'::'}$ $\langle type-name \rangle [\text{'='} \langle expression \rangle] [\langle range \rangle] \text{';'}$
$\langle range \rangle$	$::= \text{'range'} \langle expression \rangle \text{'to'} \langle expression \rangle$
$\langle enum \rangle$	$::= \text{'enum'} \langle ident \rangle$ $\text{'{' } \langle enum-field \rangle \{ \text{' ,' } \langle enum-field \rangle \} \text{'}} \text{';'}$
$\langle enum-field \rangle$	$::= \langle ident \rangle [\text{'='} \langle integer \rangle]$
$\langle const-decl \rangle$	$::= \text{'const'} \langle ident \rangle \text{'::'} \langle type-name \rangle \text{'='} \langle expression \rangle \text{';'}$
$\langle bus-decl \rangle$	$::= [\text{'exposed'}] \text{'bus'} \langle ident \rangle$ $\text{'{' } \langle bus-signal-decls \rangle \text{'}} \text{';'}$
$\langle func-decl \rangle$	$::= \text{'function'} \langle ident \rangle \text{'(' } \langle params \rangle \text{')' '{' } \{ \langle statement \rangle \}$ $\text{'}} \text{';'}$
$\langle bus-signal-decls \rangle$	$::= \langle bus-signal-decl \rangle \{ \langle bus-signal-decl \rangle \}$
$\langle bus-signal-decl \rangle$	$::= \langle ident \rangle \text{'::'} \langle type \rangle [\text{'='} \langle expression \rangle] [\langle range \rangle] \text{';'}$
$\langle inst-decl \rangle$	$::= \text{'instance'} \langle instance-name \rangle \text{'of'} \langle ident \rangle$ $\text{'(' } [\langle param-map \rangle \{ \text{' ,' } \langle param-map \rangle \}] \text{')' '}' \text{';'}$
$\langle instance-name \rangle$	$::= \langle ident \rangle \text{'[' } \langle expression \rangle \text{']'}$ (indexed instance) $\langle ident \rangle$ (named instance) '_' (anonymous instance)
$\langle param-map \rangle$	$::= [\langle ident \rangle \text{'::'}] \langle expression \rangle$
$\langle gen-decl \rangle$	$::= \text{'generate'} \langle ident \rangle \text{'='} \langle expression \rangle \text{'to'} \langle expression \rangle$ $\text{'{' } \{ \langle network-decl \rangle \} \text{'}}$
$\langle statement \rangle$	$::= \langle name \rangle \text{'='} \langle expression \rangle \text{';'}$ (assignment) $\langle ident \rangle \text{'(' } \langle param-map \rangle \text{')' '}' \text{';'}$ (function call) $\text{'if'} \text{'(' } \langle expression \rangle \text{')' '{' } \{ \langle statement \rangle \} \text{'}}$ $\{ \langle elif-block \rangle \} [\langle else-block \rangle]$ $\text{'for'} \langle ident \rangle \text{'='} \langle expression \rangle \text{'to'} \langle expression \rangle$

	‘{’ { <i>statement</i> } ‘}’ ‘switch’ <i>expression</i> ‘{’ <i>switch-case</i> { <i>switch-case</i> } [‘default’ ‘{’ <i>statement</i> { <i>statement</i> } ‘}’] ‘}’ ‘trace’ ‘(’ <i>format-string</i> { ‘,’ <i>expression</i> } ‘)’ ‘;’ ‘assert’ ‘(’ <i>expression</i> [‘,’ <i>string-literal</i>] ‘)’ ‘;’ ‘break’ ‘;’
<i>switch-case</i>	::= ‘case’ <i>expression</i> ‘{’ { <i>statement</i> } ‘}’
<i>elif-block</i>	::= ‘elif’ ‘(’ <i>expression</i> ‘)’ ‘{’ { <i>statement</i> } ‘}’
<i>else-block</i>	::= ‘else’ ‘{’ { <i>statement</i> } ‘}’
<i>format-string</i>	::= ‘”’ { <i>format-string-part</i> } ‘”’
<i>format-string-part</i>	::= ‘{’ (placeholder string) <i>string-char</i>
<i>expression</i>	::= <i>name</i> <i>literal</i> <i>expression</i> <i>bin-op</i> <i>expression</i> <i>un-op</i> <i>expression</i> ‘(’ <i>expression</i> ‘)’
<i>bin-op</i>	::= ‘+’ (addition) ‘-’ (subtraction) ‘*’ (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)
<i>un-op</i>	::= ‘-’ (negation) ‘+’ (identity)

	'!' (logical negation)
	'~' (bitwise-not)
$\langle literal \rangle$	$::=$ $\langle integer \rangle$ $\langle floating \rangle$ $\langle string-literal \rangle$ '[' $\langle integer \rangle$ { ',' $\langle integer \rangle$ } ']' (Array literal) 'true' 'false' 'U' (Undefined value)
$\langle string-literal \rangle$	$::=$ '"' { $\langle string-char \rangle$ } '"'
$\langle intrinsic-type \rangle$	$::=$ 'i' $\langle integer \rangle$ (signed integer) 'int' (arbitrary-width signed integer) 'u' $\langle integer \rangle$ (unsigned integer) 'uint' (arbitrary-width unsigned integer) 'f32' (single-precision floating point) 'f64' (double-precision floating point) 'bool' (boolean value) '[' [$\langle expression \rangle$] ']' $\langle type \rangle$ (array of type)
$\langle type \rangle$	$::=$ $\langle intrinsic-type \rangle$ $\langle ident \rangle$ (type definition)
$\langle ident \rangle$	$::=$ $\langle letter \rangle$ { $\langle letter \rangle$ $\langle number \rangle$ '_' '-' } (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'

$\langle hex-digit \rangle ::= \langle number \rangle$
 $\quad \quad \quad | \text{ 'a' - 'f' }$
 $\quad \quad \quad | \text{ 'A' - 'F' }$
 $\langle octal-digit \rangle ::= \text{ '0' - '8' }$
 $\langle string-char \rangle ::= (\text{ISO-8859-1 char with value } > 26)$

Operator precedence

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

Keywords

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