

20. Preprocessor directives 1

This clause contains the following sections.

- a) **#ifdef**, **#ifndef** — Use these preprocessor directives to control *e* processing. The preprocessor directives check for the existence of a **#define** for a given name. 5
 - 1) **#ifdef** — If a given name is defined, use the attached code; otherwise, use different code.
 - 2) **#ifndef** — If a given name is not defined, use the attached code; otherwise, use different code.

The **#ifdef** and **#ifndef** directives can be used as statements, struct members, or actions. 10
- b) **#define** — Defines a name macro, also called a *replacement macro*.
- c) **#undef** — Removes the definition of a name macro.

See also: Clause 13 and Clause 21. 15

20.1 #ifdef, #ifndef 20

Purpose	Define a preprocessor directive	20
Category	Statement, struct member, action	
Syntax	#if[n]def [<i>name</i>] then { <i>e-code</i> } [#else { <i>e-code</i> }]	25
Parameters	<i>name</i> Without a backtick (`), this is a name previously defined by a define statement. With a backtick (` <i>name</i>), this is a name defined by a Verilog <code>`define</code> directive or by a define statement, where the macro is defined in Verilog style. See also: Clause 13. 25	
	<i>e-code</i> <i>e</i> code to be included, based on whether the name macro has been defined. <ul style="list-style-type: none"> — For an #ifdef or #ifndef statement, only <i>e</i> statements can be used in <i>e-code</i>. — For an #ifdef or #ifndef struct member, only struct members can be used in <i>e-code</i>. — For an #ifdef or #ifndef action, only actions can be used in <i>e-code</i>. 30 35	

The **#ifdef** and **#ifndef** preprocessor directives can be used along with **define** name macros to cause the *e* parser to process particular code or ignore it, depending on whether a given name macro has been defined. 40

- The **#ifdef** syntax checks whether the name macro has been defined and, if it has, includes the code following the **then** construct.
- The **#ifndef** syntax checks whether the name macro has been defined and if it has not, includes the code following the **then** construct. 45

The optional **#else** syntax provides an alternative statement when the **#ifdef** or **#ifndef** is not true.

- For **#ifdef**, if the name macro has not been defined, the **#else** code is included.
- For **#ifndef**, if the name macro has been defined, the **#else** text is included. 50

Except when it is within an **#else** block, the **#ifdef** or **#ifndef** keyword shall be the first keyword on the line. 55

1 Syntax example:

```

2     #ifdef MEM_LG then {
3         import mml.e;
4     };
5 
```

NOTE—The import statement in the syntax example above needs to be on a line by itself. The syntax “**#ifdef** MEM_LG **then {import mml.e};**”, where the **import** statement is part of the **#ifdef** statement line, shall cause an error.

10 20.2 #define

Purpose	Define a name macro	
Category	Statement	
Syntax	[#]define [^]name [replacement]	
Parameters	<i>name</i>	Any e name. This is used with no replacement parameter for conditional code processing. An #ifdef preprocessor directive later in the e code that has the name as its argument evaluates to TRUE (see 20.1). When a <i>replacement</i> is given, the parser substitutes the replacement for the macro name everywhere name appears, except inside strings. The name can be preceded with a backtick (`). This makes the name look like a Verilog <code>`define name</code> , but it is treated the same as a name without a backtick. The name is case sensitive: LEN is not the same as len.
	<i>replacement</i>	Any syntactic element, for example, an expression or an HDL variable. This replaces the name wherever the name appears in the e code that follows the define statement.

With a *replacement*, this defines a macro that replaces the *name* wherever it occurs in the e code. With no *replacement*, this specifies a name that can be used in **#ifdef** preprocessor directives for conditional code. A subsequent evaluation of an **#ifdef** that has the *name* as its argument returns TRUE.

A **#define** statement shall be on a line by itself and it only applies to e code that is loaded after the **#define**. To use a **#define** macro, refer to the *name*.

The *replacement* shall not contain the *name*; if it does, this shall result in a runtime error.

Use parentheses around the *replacement* when they are needed to ensure proper associativity, e.g.,

```
define LX 2*len+m;
```

45 is different than:

```
define LX 2*(len+m);
```

In an expression like `lenx = LX`, the first case becomes `lenx = 2*len + m`, while the second case becomes `lenx = 2*len + 2*m`.

NOTE—The leading # is shown as optional in the syntax, in order to support the **define** statement syntax from previous e releases, in which the # does not appear.

****Keep this Note??**

Syntax example:

```
define PLIST_SIZE 100;
```

20.3 #undef

Purpose	Undefine a name macro
Category	Statement
Syntax	undef [`] <i>name</i>
Parameters	<p><i>name</i> Any <i>e</i> name. This is used with no replacement parameter for conditional code processing. An #ifdef preprocessor directive later in the <i>e</i> code that has the name as its argument evaluates to TRUE (see 20.1).</p> <p>When a <i>replacement</i> is given, the parser substitutes the replacement for the macro name everywhere name appears, except inside strings.</p> <p>The name can be preceded with a backtick (`). This makes the name look like a Verilog <code>`define name</code>, but it is treated the same as a name without a backtick.</p> <p>The name is case sensitive: LEN is not the same as len.</p>

This removes a name macro that was defined using the **#define** statement. The **#undef** statement can appear anywhere in the *e* code. The name macro is not recognized from the point where the **#undef** statement appears onward. The effect is propagated to all files that are loaded after the **#undef** statement is encountered. The following rules also apply.

- A name macro that is undefined in a compiled *e* module is not accessible to the C interface.
- A name macro that has been undefined can be redefined later, with any value. The last value is accessible to the C interface.
- If the undefined name macro was not previously defined, **#undef** has no effect.

Syntax example:

```
#undef PLIST_SIZE;
```