#define

#define *symbol*

The #define statement is a conditional compilation directive that defines a *symbol* for use only with #ifdef or #ifndef expressions. #undef removes the definition.

Details

The defined *symbol* exists only in the file where it is defined; the only exception is in the main procedure window where the scope covers all other procedures except independent modules. See **Conditional Compilation** on page IV-108 for information on defining a global *symbol*.

#define cannot be combined inline with other conditional compilation directives.

See Also

The #**undef**, #**ifdef-**#**endif**, and #**ifndef-**#**endif** statements.

Conditional Compilation on page IV-108.

#if-#elif-#endif

```
#if expression1
   <TRUE part 1>
#elif expression2
   <TRUE part 2>
[…]
[#else
   <FALSE part>]
#endif
```

In a #if-#elif-#endif conditional compilation statement, when an expression evaluates as TRUE (absolute value > 0.5), then only code corresponding to the TRUE part of that expression is compiled, and then the conditional statement is exited. If all expressions evaluate as FALSE (zero) then *FALSE part* is compiled when present.

Details

Conditional compiler directives must be either entirely outside or inside function definitions; they cannot straddle a function fragment. Conditionals cannot be used within Macros.

See Also

Conditional Compilation on page IV-108 for more usage details.

#if-#endif

```
#if expression
   <TRUE part>
[#else
   <FALSE part>]
#endif
```

A #if-#endif conditional compilation statement evaluates *expression*. If *expression* is TRUE (absolute value > 0.5) then the code in *TRUE part* is compiled, or if FALSE (zero) then the optional *FALSE part* is compiled.

Details

Conditional compiler directives must be either entirely outside or inside function definitions; they cannot straddle a function fragment. Conditionals cannot be used within Macros.

See Also

Conditional Compilation on page IV-108 for more usage details.