Conditional Compilation

Available since firmware version 7.6

Table of Contents

Conditional Compilation allows boolean constants to determine whether a section of code should be compiled. Conditional compilation values are defined by the `#const` identifier and has the form:

```
#const <constant-name> = <expression>
```

> **Examples of valid expressions**
>
> ```
> #const someFlag = true
> #const anotherFlag = false
> #const someOtherFlag = someFlag
> ```

Criteria:

- This initial release only supports `boolean` constant values.
- Constant names must be composed of alphanumeric characters, and optionally, the underscore _ character. There is no limit to the length of a constant name.
- Constant names are case-insensitive.
- A constant name should not be redefined if it has already been defined prior.

Defining a Constant

A constant can be defined in 2 ways:

- A manifest attribute in the channel package
- A constant locally scoped to individual BrightScript files

Manifest Constant

Conditional compilation values can be specified in the manifest via the `bs_const` attribute:

```
bs_const=someFlag=false
```

More than one conditional compilation value can be specified via semi-colon separated key-value pairs:

```
bs_const=someFlag=false;anotherFlag=true
```

Uses

There are a variety of ways Conditional Compilation can be used:

Example demonstrating use of manifest constants

```
#if someFlag
 'code to execute when someFlag is true
#else if anotherFlag
 'code to execute when anotherFlag is true
#end if
```

Example demonstrating locally scoped constants

```
#const FeatureA = true
#const FeatureB = false

#if FeatureA
 'code for Feature A
#else if FeatureB
 'code for Feature B
#else
 'production code
#end if
```

An `#error` constant can also be used to force a compilation error with an error message:

Example demonstrating usage of #error constant

```
#const FeatureA = true

#if FeatureA AND deviceSupported
 'code for Feature A
#else
 #error Your device cannot support this feature.
#end if
```

Block Comments

Conditional Compilation can also be used to form block comments. Previously, each line of code needed to be commented out.

Example of code block comments

```
#if false
 This is a function that does nothing.
 This function takes no parameters.
 This function does not return anything.
#end if
Function foo() as void
 'do nothing
End Function
```

Example of commenting out code

```
#if false
Function Order66() as void
 'code for Order66
End Function
#end if
```

Calling this function would result in a compile time error. To re-enable this function, change `#if false` to `#if true`.