

# Object Interface

## Introduction

Each **Object** shares some common functionality with other objects. The Object module encapsulates the common object methods and macros into a single module. This makes object handling consistent and more efficient.

## Methods

### Create, Delete, Validate Methods

[mpiObjectValidate](#)

### Configuration and Information Methods

[mpiObjectModuleId](#)

[mpiObjectTimeoutGet](#)

[mpiObjectTimeoutSet](#)

[meiObjectTraceGet](#)

[meiObjectTraceSet](#)

## Data Types

[MPIObjectMap](#)

## Macros

[mpiObjectMapAND\\_ASSIGN](#)

[mpiObjectMapASSIGN](#)

[mpiObjectMapBitCountMAX](#)

[mpiObjectMapBitGET](#)

[mpiObjectMapBitSET](#)

[mpiObjectMapCLEAR](#)

[mpiObjectMapCOMPLEMENT](#)

[mpiObjectMapIS\\_CLEAR](#)

[mpiObjectMapIS\\_EQUAL](#)

[mpiObjectMapIS\\_VALID](#)

[mpiObjectMapMAX](#)

[mpiObjectMapOR\\_ASSIGN](#)

[meiObjectTraceGET](#)

[meiObjectTraceSET](#)

# mpiObjectValidate

## Declaration

```
long mpiObjectValidate(MPIHandle handle,  
                       MPIModuleId moduleId)
```

**Required Header:** stdmpi.h

## Description

**mpiObjectValidate** verifies that the object (*handle*) is of the type *moduleId*.

### Return Values

<b>MPIMessageOK</b>	if <i>handle</i> is a handle to a valid object of type <i>moduleId</i> .
---------------------	--

## See Also

# mpiObjectModuleId

## Declaration

```
long mpiObjectModuleId(MPIHandle handle,  
                        MPIModuleId *moduleId)
```

**Required Header:** stdmpi.h

## Description

**mpiObjectModuleId** function writes the MPIModuleId of the object (***handle***) to the location pointed to by ***moduleId***.

### Return Values

<b>MPIMessageOK</b>	if <i>ObjectModuleId</i> successfully writes the MPIModuleId to the location
---------------------	--

## See Also

# mpiObjectTimeoutGet

## Declaration

```
long mpiObjectTimeoutGet (MPIHandle handle,  
                          MPIWait *timeout)
```

**Required Header:** stdmpi.h

## Description

**mpiObjectTimeoutGet** method gets the timeout value for the object (***handle***), and writes it to the location pointed to by ***timeout***.

### Return Values

<b>MPIMessageOK</b>	if <i>ObjectTimeoutGet</i> successfully writes the timeout value to the location
---------------------	--

## See Also

[mpiObjectTimeoutSet](#)

# mpiObjectTimeoutSet

## Declaration

```
long mpiObjectTimeoutSet(MPIHandle handle,
                        MPIWait timeout)
```

**Required Header:** stdmpi.h

## Description

**mpiObjectTimeoutSet** sets the timeout value for an object (*handle*) to *timeout*.

The *timeout* value is used in a multi-threaded environment when an MPI function must block to wait for a shared resource to become available. The default *timeout* value is MPIWaitFOREVER.

<i>If "timeout" is</i>	<b>Then</b>
<b>MPIWaitFOREVER</b>	<i>ObjectTimeoutSet</i> will wait until the resource becomes available
<b>MPIWaitPOLL</b>	<i>ObjectTimeoutSet</i> will not wait for the resource to become available. If the shared resource is not available, a timeout will be considered to have occurred.
<b>a value</b>	<i>ObjectTimeoutSet</i> will wait timeout milliseconds for the resource to become available

### Return Values

<b>MPIMessageOK</b>	if ObjectTimeoutSet successfully sets the timeout value of the object
<b>MPIMessageTIMEOUT</b>	if a timeout occurs before the shared resource becomes available

## See Also

[mpiObjectTimeoutGet](#)

# meiObjectTraceGet / meiObjectTraceGET

## Declaration: meiObjectTraceGet

```
long meiObjectTraceGet(MPIHandle    handle,
                       MEITraceMask *traceMask)
```

**Required Header:** stdmei.h

## Description

**meiObjectTraceGet** gets an Object's trace mask and writes it to the value pointed to by traceMask.

<b>handle</b>	a handle to an object
<b>*traceMask</b>	a pointer to the value of an object's trace mask

### Return Values

**MPIMessageOK** if *ObjectTraceGet* successfully gets the trace mask for an object.

## Declaration: meiObjectTraceGET

```
#define meiObjectTraceGET(object) (((MEIObject)(object))->trace)
```

**Required Header:** stdmei.h

## Description

**meiObjectTraceGET** gets the object's global trace mask.

## See Also

[meiObjectTraceSET](#) | [meiObjectTraceSet](#)

# meiObjectTraceSet / meiObjectTraceSET

## Declaration: meiObjectTraceSet

```
long meiObjectTraceSet(MPIHandle    handle,
                       MEITraceMask traceMask)
```

**Required Header:** stdmei.h

## Description

**meiObjectTraceSet** sets an Object's trace mask to the value specified by *traceMask*.

<b>handle</b>	a handle to an object
<b>traceMask</b>	the value of an object's trace mask

### Return Values

<b>MPIMessageOK</b>	if <i>ObjectTraceSet</i> successfully sets the trace mask for an object.
---------------------	--

## Declaration: meiObjectTraceSET

```
#define meiObjectTraceSET(object,mask)
        (((MEIObject)(object))->trace = (mask))
```

**Required Header:** stdmei.h

## Description

**meiObjectTraceSET** sets the object's global trace mask.

## See Also

[meiObjectTraceGET](#) | [meiObjectTraceGet](#)



# MPIObjectMap

## Definition

```
typedef unsigned long MPIObjectMap;
```

## Description

**MPIObjectMap** contains the amp fault information from a SynqNet drive. The amp fault messages are drive specific. Not all drives support amp fault messages. For details, please see the SqNodeLib header files and the drive manufacturer's documentation.

### **MPIObjectMap**

A map of MPI objects. An ObjectMap is a bitmap, where each numbered bit represents the presence or absence of the correspondingly numbered object. Valid maps are Axis/Filter and Filter/Motor. The Axis/Filter map can also be read, but setting this map must be done through the corresponding Filter objects.

## See Also

# mpiObjectMapAND\_ASSIGN

## Declaration

```
#define mpiObjectMapAND_ASSIGN(dst, src)          ((dst) &= (src))
```

**Required Header:** stdmpi.h

## Description

Bitwise ANDs the *dst* object map with the *src* object map and assigns the result to *dst*.

## See Also

[mpiObjectMapASSIGN](#)

# mpiObjectMapASSIGN

## Declaration

```
#define mpiObjectMapASSIGN(dst,src) ((dst) = (src))
```

**Required Header:** stdmpi.h

## Description

**mpiObjectMapASSIGN** assigns **src** object map to the **dst** object map.

## See Also

[mpiObjectMapAND\\_ASSIGN](#)

# mpiObjectMapBitCountMAX

## Declaration

```
#define mpiObjectMapBitCountMAX(objectMap)  
    (sizeof(objectMap) * 8)
```

**Required Header:** stdmpi.h

## Description

**mpiObjectMapBitCountMAX** calculates the maximum bit count for the specified object *objectMap*.

## See Also

# mpiObjectMapBitGET

## Declaration

```
#define mpiObjectMapBitGET(objectMap,bit)  
    (((objectMap) & (0x1 << (bit))) ? 1 : 0)
```

**Required Header:** stdmpi.h

## Description

**mpiObjectMapBitGET** gets the bit number's state for the specified object *map*.

## See Also

[mpiObjectMapBitSET](#)

# mpiObjectMapBitSET

## Declaration

```
#define mpiObjectMapBitSET(objectMap,bit,value) \  
    (((value) == 0) \  
     ? ((objectMap) &= ~(0x1 << (bit))) \  
     : ((objectMap) |= (0x1 << (bit))))
```

**Required Header:** stdmpi.h

## Description

`mpiObjectMapBitSET` sets object map's specified *bit* number to the specified *value*.

## See Also

[mpiObjectMapBitGET](#)

# mpiObjectMapCLEAR

## Declaration

```
#define mpiObjectMapCLEAR(objectMap) ((objectmap) = 0x0)
```

**Required Header:** stdmpi.h

## Description

**mpiObjectMapCLEAR** sets the object *map* to zero.

## See Also

# mpiObjectMapCOMPLEMENT

## Declaration

```
#define mpiObjectMapCOMPLEMENT(objectMap)  
    ((objectMap) = ~(objectMap))
```

**Required Header:** stdmpi.h

## Description

**mpiObjectMapCOMPLEMENT** performs bitwise complement to the object map and assigns the result to *objectMap*.

## See Also



# mpiObjectMapIS\_CLEAR

## Declaration

```
#define mpiObjectMapIS_CLEAR(objectMap) ((objectMap) == 0x0)
```

**Required Header:** stdmpi.h

## Description

**mpiObjectMapIS\_CLEAR** compares the map to 0x0. If the map is zero, it returns a non-zero value.

## See Also

# mpiObjectMapIS\_EQUAL

## Declaration

```
#define mpiObjectMapIS_EQUAL(map1, map2) ((map1) == (map2))
```

**Required Header:** stdmpi.h

## Description

**mpiObjectMapIS\_EQUAL** compares map1 to map2. If the maps are equal, it returns a non-zero value.

## See Also

# mpiObjectMapIS\_VALID

## Declaration

```
#define mpiObjectMapIS_VALID(objectMap, count) \  
    (((count) >= (sizeof(objectMap) * 8)) || \  
    (((~((0x1 << (count)) - 1)) & (objectMap)) == 0)))
```

**Required Header:** stdmpi.h

## Description

**mpiObjectMapIS\_VALID** checks if the map is within the count range. If the map is valid, it returns a non-zero value.

## See Also

# mpiObjectMapMAX

## Declaration

```
#define mpiObjectMapMAX(objectMap, count)  
    ((objectMap) = (0x1 << (count)) - 1)
```

**Required Header:** stdmpi.h

## Description

**mpiObjectMapMAX** calculates the maximum object ***objectMap*** with the specified ***count***.

## See Also

# mpiObjectMapOR\_ASSIGN

## Declaration

```
#define mpiObjectMapOR_ASSIGN(dst, src)    ((dst) |= (src))
```

**Required Header:** stdmpi.h

## Description

Bitwise ORs the *dst* object map with the *src* object map and assigns the result to *dst*.

## See Also

[mpiObjectMapASSIGN](#)