SqNode Objects

Introduction

A **SqNode** object manages a single SynqNet network node connected to a SynqNet network. It represents the physical network node. It contains information about the node, as well as its status and configuration. It provides read/write access to the node via network cyclic data and service commands. It also provides an interface to any drives connected to the node.

During network initialization, the SynqNet nodes are discovered and mapped to the SynqNet object. The number of motors per SqNode is determined and mapped to the controller's motor objects. Each node connected to a controller is assigned a number (0, 1, 2, etc) in the order it is discovered. The node number is used to index the SqNode objects.

Methods

Create, Delete, Validate Methods

meiSqNode**Create**

meiSqNode**Delete**

meiSqNode**Validate**

Configuration and Information Methods

meiSqNode**Command**

meiSqNode**ConfigGet**

meiSqNode**ConfigSet**

meiSqNode**FlashConfigGet**

meiSqNode**FlashConfigSet**

meiSqNode**FpgaDefaultFileName**

meiSqNode**Info**

meiSqNode**Status**

meiSqNode**StatusClear**

meiSqNode**UserDataGet**

meiSqNode**UserDataSet**

Drive Interface Methods

meiSqNode**DriveConfigGet**

meiSqNode**DriveConfigSet**

meiSqNode**DriveInfo**
meiSqNode**DriveMapParamCount**
meiSqNode**DriveMapParamList**
meiSqNode**DriveMapConfigCount**
meiSqNode**DriveMapConfigList**
meiSqNode**DriveMapParamFileGet**
meiSqNode**DriveMapParamFileSet**
meiSqNode**DriveMonitor**
meiSqNode**DriveMonitorConfigGet**
meiSqNode**DriveMonitorConfigSet**
meiSqNode**DriveParamCalculate**
meiSqNode**DriveParamClear**
meiSqNode**DriveParamGet**
meiSqNode**DriveParamListGet**
meiSqNode**DriveParamListSet**
meiSqNode**DriveParamReload**
meiSqNode**DriveParamRestore**
meiSqNode**DriveParamSet**
meiSqNode**DriveParamStore**

I/O Methods

meiSqNode**AnalogInput**
meiSqNode**DigitalIn**
meiSqNode**DigitalInBit**
meiSqNode**DigitalOutBitGet**
meiSqNode**DigitalOutBitSet**
meiSqNode**DigitalOutGet**
meiSqNode**DigitalOutSet**

Action Methods

meiSqNode**Download**
meiSqNode**FlashErase**
meiSqNode**FpgaFileNameVerify**
meiSqNode**Verify**

Event Methods

meiSqNode**EventNotifyGet**
meiSqNode**EventNotifySet**
meiSqNode**EventReset**

Memory Methods

meiSqNode**Memory**

meiSqNode**MemoryGet**

meiSqNode**MemorySet**

Relational Methods

meiSqNode**Control**

meiSqNode**Number**

Data Types

MEISqNode**Callback**

MEISqNode**Channel**

MEISqNode**CmdType**

MEISqNode**CmdHeader**

MEISqNode**Command**

MEISqNode**Config**

MEISqNode**ConfigIoAbort**

MEISqNode**ConfigAlarm**

MEISqNode**ConfigPacketError**

MEISqNode**ConfigTrigger**

MEISqNode**ConfigUserFault**

MEISqNode**DataSize**

MEISqNode**DownloadParams**

MEISqNode**DriveInfo**

MEISqNode**DriveMonitor**

MEISqNode**DriveMonitorConfig**

MEISqNode**DriveMonitorData**

MEISqNode**DriveMonitorDataType**

MEISqNode**DriveParamCallback**

MEISqNode**DriveParamCallbackType**

MEISqNode**FeedbackSecondary**

MEISqNode**FileName**

MEISqNode**FpgaType**

MEISqNode**Info**

MEISqNode**InfoId**

MEISqNode**InfoIo**

MEISqNode**InfoFpga**

MEISqNode**InfoNetwork**

MEISqNode**Memory**

MEISqNode**Message**

MEISqNode**MonitorValue**

MEISqNode**MonitorValueIndex**

MEISqNode**Response**

MEISqNode**Status**

MEISqNode**StatusCrcError**

MEISqNode**StatusPacketError**

MEISqNode**UserData**

Constants

MEISqNode**ID_CHAR_MAX**

MEISqNode**FILENAME_MAX**

MEISqNode**ManufacturerDATA_CHAR_MAX**

MEISqNode**MaxFEEDBACK_SECONDARY**

MEISqNode**MaxMOTORS**

MEISqNode**NOT_AVAILABLE**

MEISqNode**STATUS_NOT_AVAILABLE**

MEISqNode**UserDATA_CHAR_MAX**

MEI**DriveMapParamMAX_STRING_LENGTH**

MEI**FPGARINCONREV**

MEI**FpgaSqMACVersionDEFAULT**

MEI**FpgaSqMACVersionMIN**

MEI**FpgaSqMACVersionMAX**

MEI**FpgaSqNodeVersionDEFAULT**

MEI**FpgaSqNodeVersionMIN**

MEI**FpgaSqNodeVersionMAX**

meiSqNodeCreate

Declaration

```
meiSqNodeCreate(MPIControl  control,
                long        number)
```

Required Header

stdmei.h

Description

SqNodeCreate creates a SqNode object identified by ***number***, which is associated with a control object.

SqNodeCreate is the equivalent of a C++ constructor.

control	a handle to a Control object
number	an index to the SqNode. The first node number is 0, the second is 1, etc.

Return Values

handle	to a SqNode object. After creating a SqNode object it must be validated using meiSqNodeValidate().
MPIHandleVOID	if the object could not be created

See Also

meiSqNodeDelete | meiSqNodeValidate

meiSqNodeDelete

Declaration

```
long meiSqNodeDelete(MEISqNode  node);
```

Required Header

stdmei.h

Description

SqNodeDelete deletes a SqNode object and invalidates its handle.

SqNodeDelete is the equivalent of a C++ destructor.

node	a handle of the SqNode object to delete in the reverse order to avoid memory leaks.

Return Values

MPIMessageOK	if *SqNode* successfully deleted the object.

See Also

meiSqNodeCreate | meiSqNodeValidate

meiSqNodeValidate

Declaration

```
long meiSqNodeValidate(MEISqNode  node);
```

Required Header

stdmei.h

Description

SqNodeValidate validates a SqNode object and its handle.

SqNodeValidate is the equivalent of a C++ constructor.

node	a handle to SqNode object.

Return Values

MPIMessageOK	if *SqNode* is a handle to a valid object.

See Also

meiSqNodeCreate | meiSqNodeDelete

meiSqNodeCommand

Declaration

```
long meiSqNodeCommand(MEISqNode          node,
                      MEISqNodeCommand   *command,
                      MEISqNodeResponse  *response);
```

Required Header

stdmei.h

Description

SqNodeCommand sends a service command to a SynqNet node using the data from the structure pointed to by command and writes the response into the structure pointed to by response. Service commands occur across the SynqNet network through a service channel. In SYNQ mode there is one service channel for each node. In ASYNQ mode there is one service channel for all nodes. The controller sends the command and waits for a response using a 4 state handshake. In SYNQ mode the service command data is sent through the cyclic packets, but due to the handshaking, it is not considered a cyclic operation.

For SynqNet nodes that have drive memory interfaces, service commands can be sent to drives. Also, the service commands supports access to drive data memory, program memory, I/O memory, and direct commands. Please see the MEISqNodeCommand{…} and MEISqNodeResponse{…} structures for more information. And be sure to consult the drive's header file in the (C:\MEI)\XMP\sqNodeLib\include directory, as well as, the drive manufacturer's manual for valid drive addresses.

node	a handle to a SynqNet node object
***command**	a pointer to a SynqNet node command structure
***response**	a pointer to a SynqNet node response structure

Return Values

MPIMessageOK	if *SqNodeCommand* successfully sends a service command and receives a response.
MPIMessageARG_INVALID	if the command pointer is NULL or the response pointer is NULL
MEISynqNetMessageRESPONSE_TIMEOUT	if the node does not respond to the service command.
MEISynqNetMessageREADY_TIMEOUT	if the node is busy and does not acknowledge the service command.

See Also

MEISqNodeCommand | MEISqNodeResponse | MEISqNodeCmdHeader | MEISqNodeCmdType | MEISqNodeDataSize | MEISqNodeMemory | MEISqNodeChannel

meiSqNodeConfigGet

Declaration

```
long meiSqNodeConfigGet(MEISqNode       node,
                        MEISqNodeConfig *config);
```

Required Header

stdmei.h

Description

SqNodeConfigGet reads a SynqNet node's (***node***) configuration and writes it into the structure pointed to by ***config***.

node	a handle to a SynqNet node object.
***config**	a pointer to a SynqNet node config structure.

Return Values

MPIMessageOK	if *SqNodeConfigGet* successfully reads the node configuration.

See Also

meiSqNodeInfo | meiSqNodeDriveConfigGet

meiSqNodeConfigSet

Declaration

```
long meiSqNodeConfigSet(MEISqNode       node,
                        MEISqNodeConfig *config);
```

Required Header

stdmei.h

Description

SqNodeConfigSet writes a SynqNet node's (***node***) configuration using data from the structure pointed to by ***config***.

node	a handle to a SynqNet node object
***config**	a pointer to a SynqNet node config structure

Return Values

MPIMessageOK	if *SqNodeConfigGet* successfully writes the node configuration.
MEIMessageARG_INVALID	if the upStreamError or downStreamError fault/fail limits are less than 0 or greater than 255.
MEISqNodeMessageFEEDBACK_MAP_INVALID	if secondary encoder (n) is not mappable to the motor on the node specified by MEISqNodeFeedbackSecondary[n].motorIndex..

See Also

meiSqNodeInfo | meiSqNodeDriveConfigSet | meiSynqNetFlashTopologySave

meiSqNodeFlashConfigGet

Declaration

```
long meiSqNodeFlashConfigGet(MEISqNode          node,
                             void               *flash,
                             MEISqNodeConfig    *config);
```

Required Header

stdmei.h

Description

SqNodeFlashConfigGet reads a SynqNet node's flash configuration and writes it into the structure pointed to by ***config***.

node	a handle to a SynqNet node object.
***flash**	***flash*** is either an MEIFlash handle or MPIHandleVOID. If ***flash*** is MPIHandleVOID, an MEIFlash object will be created and deleted internally.
***config**	a pointer to a SynqNet node config structure.

Return Values

MPIMessageOK	if *SqNodeFlashConfigGet* successfully reads a SynqNet node's flash configuration and writes it into the structure pointed to by *config*.

See Also

meiSqNodeFlashConfigSet

meiSqNodeFlashConfigSet

Declaration

```
long meiSqNodeFlashConfigSet(MEISqNode          node,
                             void               *flash,
                             MEISqNodeConfig    *config);
```

Required Header

stdmei.h

Description

SqNodeFlashConfigSet sets a SynqNet Node (***node***) flash configuration using data from the structure pointed to by ***config***.

NOTE: The network topology must first be saved before changing node config values in Flash memory. These values will also be cleared when network topology is cleared using meiSynqNetFlashTopologyClear(…).

node	a handle to a SynqNet node object.
***flash**	***flash*** is either an MEIFlash handle or MPIHandleVOID. If ***flash*** is MPIHandleVOID, an MEIFlash object will be created and deleted internally. If ***flash*** is a valid MEIFlash handle, then the MEIFlash object cache will be updated, but the actual write to controller flash will not occur. Use meiFlashMemoryFromFileType(...) to prompt the actual write to ***flash***.
***config**	a pointer to a SynqNet node config structure.

Return Values

MPIMessageOK	if *SqNodeFlashConfigSet* successfully sets a SynqNet Node (*node*) flash configuration using data from the structure pointed to by *config*.
MEISqNodeMessageFEEDBACK_MAP_INVAILD	given secondary encoder (*n*) is not mappable to the motor on the node specified by MEISqNodeFeedbackSecondary[n].motorIndex.
MPIMessageARG_INVALID	If the upStreamError or downStreamError fault/fail limits are less than 0 or greater than 255.
MEIFlashMessageNETWORK_TOPOLOGY_ERROR	if a valid flash handle was supplied and SynqNet topology had not yet been saved to flash using meiSynqNetFlashTopologySave(…).

meiSqNodeFpgaDefaultFileName

Declaration

```
long meiSqNodeFpgaDefaultFileName(MEISqNode           sqNode,
                                  MEISqNodeFileName   *fileName);
```

Required Header

stdmei.h

Description

SqNodeFpgaFileDefaultFileName provides the default image filename for an sqNode.

sqNode	handle to a SqNode object.
***fileName**	a pointer to a structure that has space allocated to hold an FPGA filename.

Return Values

MPIMessageOK	if *SqNodeFpgaDefaultFileName* successfully returns the name of an FPGA image file.
MPIMessageARG_INVALID	if the pointer to filename is NULL

See Also

meiSqNodeInfo

Declaration

```
long meiSqNodeInfo(MEISqNode      node,
                   MEISqNodeInfo  *info);
```

Required Header

stdmei.h

Description

SqNodeInfo reads a SynqNet node's information and writes it into a structure pointed to by ***info***. The info structure contains read only data about the node.

The RMB-10V, RMB-10V2 and some Trust nodes support analog inputs. MPI support has been added to support the reading of node-based analog inputs. The number of analog inputs a node supports can be determined with meiSqNodeInfo(...). An analog input value can be read with meiSqNodeAnalogIn(...). The analog to digital converted value is scaled from -1.0 to +1.0, where +1.0 is a full-scale positive voltage. The input range of the ADC is hardware-specific.

node	a handle to a SynqNet node object.
***info**	a pointer to a drive specific information structure.

Return Values

MPIMessageOK	if *SqNodeInfo* successfully reads the node information.
MPIMessageARG_INVALID	if the info pointer is NULL.

See Also

meiSqNodeDriveInfo | meiSqNodeConfigGet | meiSqNodeDriveConfigGet | meiSqNodeAnalogIn

meiSqNodeStatus

Declaration

```
long meiSqNodeStatus(MEISqNode          node,
                     MEISqNodeStatus    *status);
```

Required Header

stdmei.h

Description

SqNodeStatus reads status from the ***node*** associated with the SynqNet object and writes it into the structure pointed to by ***status***. The SynqNet node status structure contains error counters and event mask data.

NOTE: This data requires service commands to acces the data on the node. As a result, it may take up to 9 servo cycles to read the data. At the default sample rate of 2kHz, this would translate to 4.5ms.

node	a handle to a SynqNet node object.
***status**	pointer to a SynqNet status structure.

Return Values

MPIMessageOK	if *SynqNetStatus* successfully reads the node status and writes it into the structure.
MPIMessageARG_INVALID	if the status pointer is NULL.

See Also

meiSynqNetStatus | meiSqNodeInfo

meiSqNodeStatusClear

Declaration

```
long meiSqNodeConfigSet(MEISqNode      node);
```

Required Header stdmei.h

Description **SqNodeStatusClear** clears node CRC errors on all ports, clears node Packet errors, clears node ioAbort state, and resets SqNode events.

node	a handle to a SynqNet node object

Return Values

MPIMessageOK	if *SqNodeStatusClear* successfully ...

See Also MEIEventTypeSQNODE

meiSqNodeUserDataGet

Declaration

```
long meiSqNodeUserDataGet(MEISqNode           node,
                          MEISqNodeUserData   *data);
```

Required Header stdmei.h

Description **SqNodeUserDataGet** reads the user data from the node.

node	a handle to a SynqNet node object.
***data**	a pointer to a MEISqNodeUserData structure, allocated on the host.

Return Values	
MPIMessageOK	if *SqNodeUserDataGet* successfully gets the user data from the node.

See Also meiSqNodeUserDataSet

meiSqNodeUserDataSet

Declaration

```
long meiSqNodeUserDataSet(MEISqNode          node,
                          MEISqNodeUserData  *data);
```

Required Header

stdmei.h

Description

SqNodeUserDataSet writes the user data to the node.

node	a handle to a SynqNet node object.
***data**	a pointer to a MEISqNodeUserData structure, allocated on the host.

Return Values

MPIMessageOK	if *SqNodeUserDataSet* successfully sets the user data to the node.

See Also

meiSqNodeUserDataGet | MEISqNodeUserData

meiSqNodeDriveConfigGet

Declaration

```
long meiSqNodeDriveConfigGet(MEISqNode  node,
                             long       driveIndex, /* relative to the node */
                             void       *config);   /* node specific       */
```

Required Header

stdmei.h

Description

SqNodeDriveConfigGet reads a SynqNet node's drive configuration and writes it into a drive specific structure pointed to by ***config***. SynqNet nodes may support one or more drive interfaces. The drive configuration can be read if the drive interface hardware supports a communication channel to the drive processor. The drive interface(s) for a SynqNet node are indexed by a number (0, 1, 2, etc.).

The drive configuration structure is drive specific. The SqNodeLib includes the drive specific structures and methods. Please see the drive's header file in the (C:\MEI)\XMP\sqNodeLib\include directory, as well as, the drive manufacturer's documentation for details. Use meiSqNodeInfo(…), to determine if the SynqNet node supports a drive interface and it's type.

node	a handle to a SynqNet node object
driveIndex	an index to a drive interface on a SynqNet node. The first drive interface is 0, the second is 1, etc.
***config**	a pointer to a drive specific configuration structure.

Return Values

MPIMessageOK	if *SqNodeDriveConfigGet* successfully reads the drive configuration.

See Also

meiSqNodeInfo | meiSqNodeDriveInfo | meiSqNodeConfigGet |

See Also

meiSqNodeFlashConfigGet | Flash Objects | meiSynqNetFlashTopologySave

meiSqNodeDriveConfigSet

Declaration

```
long meiSqNodeDriveConfigSet(MEISqNode  node,
                             long       driveIndex, /* relative to the node */
                             void       *config);   /* node specific       */
```

Required Header

stdmei.h

Description

SqNodeDriveConfigSet writes a SynqNet node's drive configuration from a drive specific structure pointed to by ***config***. SynqNet nodes may support one or more drive interfaces. The drive configuration can be written if the drive interface hardware supports a communication channel to the drive processor. The drive interface(s) for a SynqNet node are indexed by a number (0, 1, 2, etc.).

The drive configuration structure is drive specific. The SqNodeLib includes the drive specific structures and methods. Please see the drive manufacturer's documentation for details. Use meiSqNodeInfo(…), to determine if the SynqNet node supports a drive interface and its type.

node	a handle to a SynqNet node object
driveIndex	an index to a drive interface on a SynqNet node. The first drive interface is 0, the second is 1, etc.
***config**	a pointer to a drive specific configuration structure.

Return Values

MPIMessageOK	if *SqNodeDriveConfigSet* successfully writes the drive configuration.

See Also

meiSqNodeInfo | meiSqNodeDriveInfo | meiSqNodeConfigSet |

meiSqNodeDriveInfo

Declaration

```
long meiSqNodeDriveInfo(MEISqNode           node,
                        long                driveIndex, /* relative to the node */
                        MEISqNodeDriveInfo  *info,
                        void                *external);  /* node specific */
```

Required Header

stdmei.h

Description

SqNodeDriveInfo reads a SynqNet node's drive information and writes it into a drive specific structure pointed to by ***info***. The drive info structure contains read only data. SynqNet nodes may support one or more drive interfaces. The drive information can be read if the drive interface hardware supports a communication channel to the drive processor. The drive interface(s) for a SynqNet node are indexed by a number (0, 1, 2, etc.).

The drive information structure is drive specific. The SqNodeLib includes the drive specific structures and methods. Please see the drive's header file for the drive specific information structures, as well as, the drive manufacturer's documentation for details. All supported drive header files are located in the (C:\MEI)\XMP\sqNodeLib\include directory. Use meiSqNodeInfo(…), to determine if the SynqNet node supports a drive interface and it's type.

node	a handle to a SynqNet node object.
driveIndex	an index to a drive interface on a SynqNet node. The first drive interface is 0, the second is 1, etc.
***info**	a pointer to a structure that contains general drive information.
***external**	a pointer to a drive specific information structure. See the appropriate drive vendor *.h for definition. (NOTE: it can be NULL)

Return Values

MPIMessageOK	if *SqNodeDriveInfo* successfully reads the drive information.

See Also

meiSqNodeInfo | meiSqNodeConfigGet | meiSqNodeDriveConfigGet |

meiSqNodeDriveMapParamCount

Declaration

```
long meiSqNodeDriveMapParamCount (MEISqNode     sqNode,
                                  MEIDriveMap   driveMap,
                                  long          driveIndex,
                                  long          *paramsCount);
```

Required Header

stdmei.h

Description

SqNodeDriveMapParamCount scans the drive map file for a drive entry that matches this node on the network. If an entry is found, then this function returns the number of drive parameters that need to be preserved for the configuration of the drive.

This function is normally used with the meiSqNodeDriveMapParamList function. First, this function is called in order to get the size of the drive parameter list. Then the user can use this size to allocate enough memory to hold the complete parameter list before calling meiSqNodeDriveMapParamList to fill in the list.

sqNode	a handle to a SynqNet node object.
driveMap	a handle to a DriveMap object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
***paramsCount**	pointer to the variable that will be set by this function.

Return Values

MPIMessageOK	if *meiSqNodeDriveMapParamCount* successfully scans the drive map file for a drive entry that matches this node on the network and returns the number of drive parameters that need to be preserved for the configuration of the drive.

See Also

meiSqNodeDriveMapParamList

meiSqNodeDriveMapParamList

Declaration

```
long meiSqNodeDriveMapParamList (MEISqNode      sqNode,
                                 MEIDriveMap    driveMap,
                                 long           driveIndex,
                                 long           paramsCount,
                                 MEIDriveParamInfo *driveParamInfo);
```

Required Header

stdmei.h

Description

SqNodeDriveMapParamList scans the drive map file for an entry that matches the node on the network. If a drive entry is found, this function writes the drive parameter information about each of the drive parameters to the ***driveParamInfo*** list.

This function is normally used with the meiSqNodeDriveMapParamCount function. The meiSqNodeDriveMapParamCount function is called first to get the size of the parameter list, the user can then use this size to allocate enough memory to hold the complete parameter list before calling this function to fill in the parameter list.

sqNode	a handle to a SynqNet node object.
driveMap	a handle to a DriveMap object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
paramsCount	the number of drive parameter information records that can be written to the *driveParamInfo* list.
***driveParamInfo**	pointer to the list of drive parameter information records that will be filled in by this function.

Return Values

MPIMessageOK	if *meiSqNodeDriveMapParamList* successfully scans the drive map file for an entry that matches the node on the network and writes the information about each of the drive parameters to the *driveParamInfo* list.

See Also

meiSqNodeDriveMapParamCount

meiSqNodeDriveMapConfigCount

Declaration

```
long meiSqNodeDriveMapConfigCount (MEISqNode     sqNode,
                                   MEIDriveMap   driveMap,
                                   long          driveIndex,
                                   long          *configCount);
```

Required Header

stdmei.h

Description

SqNodeDriveMapScanConfigCount scans the drive map file for a drive entry that matches this node on the network. If an entry is found, this function returns the number of drive parameters that need to be preserved for the configuration of the drive.

This function is normally used with the meiSqNodeDriveMapConfigList function. This function is first called to get the size of the drive configuration list. Then the user can use this size to allocate enough memory to hold the complete configuration list before calling meiSqNodeDriveMapConfigList to fill in the list.

sqNode	a handle to a SynqNet node object.
driveMap	a handle to a DriveMap object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
***configCount**	pointer to the variable that will be set by this function.

Return Values

MPIMessageOK	if *meiSqNodeDriveMapConfigCount* successfully scans the drive map file for a drive entry that matches this node on the network and returns the number of drive parameters that need to be preserved for the configuration of the drive.

See Also

meiSqNodeDriveMapConfigList

meiSqNodeDriveMapConfigList

Declaration

```
long meiSqNodeDriveMapConfigList (MEISqNode     sqNode,
                                  MEIDriveMap   driveMap,
                                  long          driveIndex,
                                  long          configCount,
                                  long          *configList);
```

Required Header

stdmei.h

Description

SqNodeDriveMapScanConfigList scans the drive map file for a drive entry that matches this node on the network. If an entry is found, this function returns the list of drive parameters that need to be preserved for the configuration of the drive.

This function is normally used with the meiSqNodeDriveMapConfigCount function. The meiSqNodeDriveMapConfigCount function is first called to get the size of the drive configuration list. Then the user can use this size to allocate enough memory to hold the complete configuration list before calling this function to fill in the list.

sqNode	a handle to a SynqNet node object.
driveMap	a handle to a DriveMap object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
configCount	the number of drive parameter information records that can be written to the *configList* list.
***configList**	pointer to the list of drive parameters that make up the drive configuration that will be filled in by this function.

Return Values

MPIMessageOK	if *meiSqNodeDriveMapConfigList* successfully scans the drive map file for a drive entry that matches this node on the network and returns the list of drive parameters that need to be preserved for the configuration of the drive.

See Also

meiSqNodeDriveMapConfigCount

meiSqNodeDriveMapParamFileGet

Declaration

```
long meiSqNodeDriveMapParamFileGet(MEISqNode     sqNode,
                                   MEIDriveMap   driveMap,
                                   long          driveIndex,
                                   char          *driveConfigFilename,
                                   long          append);
```

Required Header

stdmei.h

Description

SqNodeDriveMapParamFileGet saves the current set of drive parameters in the drive to the ***driveConfigFilename***.

sqNode	a handle to the SynqNet node object.
driveMap	a handle to the DriveMap object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
***driveConfigFilename**	the name of the file that holds the stored drive configuration file.
append	1 = The new data is appended to the existing drive configuration file if it exists. 0 = A new drive configuration file is created to hold the drive parameters. If a file already exists, it will be overwritten.

Return Values

MPIMessageOK	if *SqNodeDriveMapParamFileGet* successfully saves the current set of drive parameters in the drive to the ***driveConfigFilename***.

See Also

meiSqNodeDriveMapParamFileSet

meiSqNodeDriveMapParamFileSet

Declaration

```
long meiSqNodeDriveMapParamFileSet(MEISqNode                  sqNode,
                                   MEIDriveMap                driveMap,
                                   long                       driveIndex,
                                   char                       *driveConfigFilename,
                                   MEISqNodeDriveParamCallback callback,
                                   long                       warning);
```

Required Header

stdmei.h

Description

SqNodeDriveMapParamFileSet loads the drive parameters stored in the file named ***driveConfigFilename*** into the drive.

sqNode	a handle to the SynqNet node object.
driveMap	a handle to the DriveMap object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
***driveConfigFilename**	the name of the file that holds the stored drive configuration file.
callback	A callback function that this function calls to indicate if the function is changing the value of a drive parameter or setting a new drive parameter that has failed. Passing NULL for this parameter will disable the callback feature.
warning	0 = if setting a drive parameter fails, this function will fail immediately. 1 = if setting a drive parameter fails, then the function will continue with the remaining drive parameters and generate a warning by calling the callback function.

Return Values

MPIMessageOK	if *SqNodeDriveMapParamFileSet* successfully loads the drive parameters stored in the file named ***driveConfigFilename*** into the drive.

See Also

meiSqNodeDriveMapParamFileGet

meiSqNodeDriveMonitor

Declaration

```
long meiSqNodeDriveMonitor(MEISqNode             node,
                           long                  driveIndex, /* relative to the node
*/
                           MEISqNodeMonitorValue *value);
```

Required Header

stdmei.h

Description

SqNodeDriveMonitor reads the monitor fields from the drive and writes them into the structure pointed to by ***value***.

node	a handle to a SynqNet node object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
***value**	pointer to a structure of monitor values

Return Values

MPIMessageOK	if *SqNodeDriveMonitor* successfully reads the monitor information.

See Also

MEISqNodeMonitorValue

meiSqNodeDriveMonitorConfigGet

Declaration

```
long meiSqNodeDriveMonitorConfigGet(MEISqNode     node,
                                    long          driveIndex, /* relative to the node
*/
                                    MEISqNodeDriveMonitorConfig   *config);
```

Required Header

stdmei.h

Description

SqNodeDriveMonitorConfigGet reads a SynqNet node's drive monitor configuration and writes it into a structure pointed to by ***config***. SynqNet nodes may support one or more drive interfaces. The drive monitor configuration can be read if the drive interface hardware supports a communication channel to the drive processor. The drive interface(s) for a SynqNet node are indexed by a number (0, 1, 2, etc.).

The SynqNet network packets have some extra fields that can be configured to read drive data every sample. Each monitor field is 32 bits. SynqNet nodes with drive interfaces that support drive monitoring can be configured to transmit the data. The drive manufacturer determines what data is available for monitoring. The monitor data can be specified by a predetermined index or memory address. Please see the drive's header file for the drive specific configuration structures, as well as, the drive manufacturer's documentation for details. All supported drive header files are located in the (C:\MEI)\XMP\sqNodeLib\include directory.

node	a handle to a SynqNet node object.
driveIndex	an index to a drive interface on a SynqNet node. The first drive interface is 0, the second is 1, etc.
***config**	a pointer to a drive monitor configuration structure.

Return Values

MPIMessageOK	if *SqNodeDriveMonitorConfigGet* successfully reads the drive monitor configuration.

See Also

meiSqNodeInfo | meiSqNodeDriveInfo

meiSqNodeDriveMonitorConfigSet

Declaration

```
long meiSqNodeDriveMonitorConfigSet(MEISqNode  node,
            long                       driveIndex,   /* relative to the node */
            MEISqNodeDriveMonitorConfig  *config);
```

Required Header

stdmei.h

Description

SqNodeDriveMonitorConfigSet writes a SynqNet node's drive monitor configuration from a structure pointed to by ***config***. SynqNet nodes may support one or more drive interfaces. The drive monitor configuration can be written if the drive interface hardware supports a communication channel to the drive processor. The drive interface(s) for a SynqNet node are indexed by a number (0, 1, 2, etc.).

The SynqNet network packets have some extra fields that can be configured to read drive data every sample. Each monitor field is 32 bits. SynqNet nodes with drive interfaces that support drive monitoring can be configured to transmit the data. The drive manufacturer determines what data is available for monitoring. The monitor data can be specified by a predetermined index or memory address. Please see the drive's header file for the drive specific configuration structures, as well as, the drive manufacturer's documentation for details. All supported drive header files are located in the (C:\MEI)\XMP\sqNodeLib\include directory.

node	a handle to a SynqNet node object.
driveIndex	an index to a drive interface on a SynqNet node. The first drive interface is 0, the second is 1, etc.
***config**	a pointer to a drive monitor configuration structure.

Return Values

MPIMessageOK	if *SqNodeDriveMonitorConfigSet* successfully writes the drive monitor configuration.

See Also

meiSqNodeInfo | meiSqNodeDriveInfo

meiSqNodeDriveParamCalculate

Declaration

```
long meiSqNodeDriveParamCalculate(MEISqNode sqNode,
                                  long      driveIndex);
```

Required Header

stdmei.h

Description

Some drives need to calculate some internal quantities after a drive parameter has been changed. The **SqNodeDriveParamCalculate** function will instruct the drive to calculate its internal quantities. This feature is not supported or required by all drives.

sqNode	a handle to a SynqNet node object
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.

Return Values

MPIMessageOK	if *SqNodeDriveParamCalculate* successfully has the drive to calculate its internal quantities.

See Also

meiSqNodeDriveParamClear

Declaration

```
long meiSqNodeDriveParamClear(MEISqNode sqNode,
                              long      driveIndex);
```

Required Header

stdmei.h

Description

SqNodeDriveParamClear clears the previously saved drive by loading the default set of drive parameters into the current and non-volatile storage on the drive. These drive parameters will be used each time this drive is subsequently started (after a power-on or network reset). The default drive parameters will take effect immediately.

NOTE: This function may not be supported by all drives. The default set of drive parameters may be different between different drive types and different drive manufactures.

sqNode	a handle to a SynqNet node object
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.

Return Values

MPIMessageOK	if *SqNodeDriveParamClear* successfully clears the previously saved drive by loading the default set of drive parameters into the current and non-volatile storage on the drive.

See Also

meiSqNodeDriveParamReload

meiSqNodeDriveParamGet

Declaration

```
long meiSqNodeDriveParamGet(MEISqNode                node,
                            long                     driveIndex,
                            long                     param,
                            MEIDriveMapParamType     paramType,
                            MEIDriveMapParamValue    *value);
```

Required Header

stdmei.h

Description

SqNodeDriveParamGet reads a drive parameter from the drive and fills in the appropriate field of the union pointed to by ***value***. The ***paramType*** defines the type of data that is read from the drive and also defines which field will be used in the ***value*** union.

node	a handle to the SynqNet node object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
param	an index for the drive parameter that is being accessed.
paramType	the type of the data read from the drive and which field will be used in the *value* union.
***value**	a pointer to the union that will be filled in.

Return Values

MPIMessageOK	if *SqNodeDriveParamGet* successfully reads a drive parameter from the drive and fills in the appropriate field of the union pointed to by ***value***.

See Also

meiSqNodeDriveParamSet

meiSqNodeDriveParamListGet

Declaration

```
long meiSqNodeDriveParamListGet(MEISqNode                node,
                                long                     driveIndex,
                                long                     size,
                                long                     *paramList,
                                MEIDriveMapParamType     *paramTypes,
                                MEIDriveMapParamValue    *paramValues);
```

Required Header

stdmei.h

Description

SqNodeDriveParamListGet reads a series of drive parameters from the drive and fills in the appropriate fields of the unions pointed to by ***paramValues***. The ***paramTypes*** defines the type of data that is read from the drive and also defines which fields in the ***paramValues*** unions are going to be used.

node	a handle to the SynqNet node object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
size	the number of drive parameters to be read.
***paramList**	a pointer to a list of the drive parameter indexes that are being accessed.
***paramTypes**	a pointer to a list of drive parameter types to be read from the drive and which field in the paramValues union is going to be used.
***paramValues**	a pointer to a list of unions that will be filled in by this function.

Return Values

MPIMessageOK	if *SqNodeDriveParamListGet* successfully reads a series of drive parameters from the drive and fills in the appropriate fields of the unions pointed to by ***paramValues***.

See Also

meiSqNodeDriveParamListSet

meiSqNodeDriveParamListSet

Declaration

```
long meiSqNodeDriveParamListSet(MEISqNode                node,
                                long                     driveIndex,
                                long                     size,
                                long                     *paramList,
                                MEIDriveMapParamType     *paramTypes,
                                MEIDriveMapParamValue    *paramValues);
```

Required Header

stdmei.h

Description

SqNodeDriveParamListSet writes a series of drive parameters pointed to by ***value*** to the drive. The ***paramTypes*** defines each type of data item that is written to the drive and also defines which field in the ***paramValues*** unions are going to be used.

node	a handle to the SynqNet node object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
size	the number of drive parameters to be written.
***paramList**	a pointer to a list of drive parameter types to be written to the drive and which field in the paramValues union is going to be used.
***paramTypes**	a pointer to a list of the drive parameter indexes that are being accessed.
***paramValues**	a pointer to a list of unions that will be written to the drive.

Return Values

MPIMessageOK	if *SqNodeDriveParamListSet* successfully writes a series of drive parameters pointed to by ***value*** to the drive.

See Also

meiSqNodeDriveParamListGet

meiSqNodeDriveParamReload

Declaration

```
long meiSqNodeDriveParamReload(MEISqNode sqNode,
                               long      driveIndex);
```

Required Header stdmei.h

Description **SqNodeDriveParamReload** overwrites the current set of drive parameters with the set from the non-volatile storage on the drive. These new drive parameters will take effect immediately.

sqNode	a handle to a SynqNet node object
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.

Return Values

MPIMessageOK	if *SqNodeDriveParamReload* successfully overwrites the current set of drive parameters with the set from the non-volatile storage on the drive.

See Also meiSqNodeDriveParamClear

meiSqNodeDriveParamRestore

Declaration

```
long meiSqNodeDriveParamRestore(MEISqNode sqNode,
                                long      driveIndex);
```

Required Header stdmei.h

Description **SqNodeDriveParamRestore** loads the default set of drive parameters into current set of drive parameters on the drive. The default drive parameters will take effect immediately.

sqNode	a handle to a SynqNet node object
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.

Return Values

MPIMessageOK	if *SqNodeDriveParamRestore* successfully loads the default set of drive parameters into current set of drive parameters on the drive.

See Also meiSqNodeDriveParamStore

meiSqNodeDriveParamSet

Declaration

```
long meiSqNodeDriveParamSet(MEISqNode                node,
                            long                     driveIndex,
                            long                     param,
                            MEIDriveMapParamType     paramType,
                            MEIDriveMapParamValue    *value);
```

Required Header

stdmei.h

Description

SqNodeDriveParamSet writes the drive parameter that is pointed to by ***value*** to the drive. The ***paramType*** defines the type of data that is written to the drive and also defines which field will be used in the ***value*** union.

node	a handle to the SynqNet node object.
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.
param	an index for the drive parameter that is being accessed.
paramType	the type of data being written to the drive and which field will be used in the *value* union.
***value**	pointer to the union that will be written to the drive.

Return Values

MPIMessageOK	if *SqNodeDriveParamSet* successfully writes the drive parameter that is pointed to by ***value*** to the drive.

See Also

meiSqNodeDriveParamGet

meiSqNodeDriveParamStore

Declaration

```
long meiSqNodeDriveParamStore(MEISqNode sqNode,
                              long      driveIndex);
```

Required Header

stdmei.h

Description

SqNodeDriveParamStore saves the drive parameters into non-volatile storage on the drive. These drive parameters will be used each time the drive is subsequently started (after a power-on or network reset).

NOTE: This function may not be supported by all drives.

sqNode	a handle to a SynqNet node object
driveIndex	an index to the drive (0, 1, 2, etc), relative to the node.

Return Values

MPIMessageOK	if *SqNodeDriveParamStore* successfully saves the drive parameters into non-volatile storage on the drive.

See Also

meiSqNodeDriveParamRestore

meiSqNodeAnalogInput

Declaration

```
long meiSqNodeAnalogInput(MEISqNode     node,
                          long          channel,
                          double        *state);
```

Required Header

stdmei.h

Description

SqNodeAnalogInput gets the latest sample of the analog input channel on a SynqNet node. The analog value is between +1 and -1.

The RMB-10V, RMB-10V2 and some Trust nodes support analog inputs. MPI support has been added to support the reading of node-based analog inputs. The number of analog inputs a node supports can be determined with meiSqNodeInfo(...). An analog input value can be read with meiSqNodeAnalogIn(...). The analog to digital converted value is scaled from -1.0 to +1.0, where +1.0 is a full-scale positive voltage. The input range of the ADC is hardware-specific.

node	a handle to a SynqNet node object.
channel	the number of the analog input channel on this node.
***state**	the magnitude to the analog input.

Return Values

MPIMessageOK	if *SqNodeAnalogInput* successfully gets the latest sample of the analog input channel on a SynqNet node.

See Also

meiSqNodeDigitalIn

Declaration

```
long meiSqNodeDigitalIn(MEISqNode     node,
                        long          bitCount,
                        long          *state);
```

Required Header

stdmei.h

Description

SqNodeDigitalIn gets the state of a set of digital inputs on a SynqNet node. The ***bitCount*** argument defines how may input bits are to be copied into the memory pointed to by the ***state*** argument. You can find out the maximum number of digital input bits on each node by using the meiSqNodeInfo function.

For example:

```
long inputs[2];
meiSqNodeDigitalIn( node0, 32, inputs );
```

node	a handle to a SynqNet node object.
bitCount	how many bits are to be copied.
***state**	pointer to where the inputs state is copied.

Return Values

MPIMessageOK	if *SqNodeDigitalIn* successfully gets the state of all the digital inputs on a SynqNet node.

See Also

meiSqNodeDigitalInBit

Declaration

```
long meiSqNodeDigitalInBit(MEISqNode     node,
                           long          bit,
                           long          *state);
```

Required Header

stdmei.h

Description

The **SqNodeDigitalInBit** function gets the state of a digital input bit on a SynqNet node.

node	a handle to a SynqNet node object.
bit	the number of the input bit on the node.
***state**	The current state of the input on the node. This will either be one or zero.

Return Values

MPIMessageOK	if *SqNodeDigitalInBit* successfully gets the state of a digital input bit on a SynqNet node.

See Also

meiSqNodeDigitalOutBitGet | meiSqNodeDigitalOutBitSet

meiSqNodeDigitalOutBitGet

Declaration

```
long meiSqNodeDigitalOutBitGet(MEISqNode     node,
                               long          bit,
                               long          *state);
```

Required Header

stdmei.h

Description

SqNodeDigitalOutBitGet gets the state of a digital output bit on a SynqNet node.

node	a handle to a SynqNet node object
bit	the number of the output bit on the node.
***state**	the current state of the output on the node. This will either be one or zero.

Return Values

MPIMessageOK	if *SqNodeDigitalOutBitGet* successfully gets the state of a digital output bit on a SynqNet node.

See Also

meiSqNodeDigitalOutBitSet

meiSqNodeDigitalOutBitSet

Declaration

```
long meiSqNodeDigitalOutBitSet(MEISqNode node,
                               long      bit,
                               long      state);
```

Required Header

stdmei.h

Description

SqNodeDigitalOutBitSet sets the state of a digital output bit on a SynqNet node.

node	a handle to a SynqNet node object
bit	the number of the output bit on the node.
state	the desired state of the output on the node. This should either be one or zero.

Return Values

MPIMessageOK	if *SqNodeDigitalOutBitSet* successfully sets the state of a digital output bit on a SynqNet node.

See Also

meiSqNodeDigitalOutBitGet

meiSqNodeDigitalOutGet

Declaration

```
long meiSqNodeDigitalOutGet(MEISqNode     node,
                            long          bitCount,
                            long          *state);
```

Required Header

stdmei.h

Description

SqNodeDigitalOutSet gets the current state of a set of digital outputs on a SynqNet node. The ***bitCount*** argument defines how may output bits are to be copied into the memory pointed to by the ***state*** argument. You can find out the maximum number of digital output bits on each node using the meiSqNodeInfo function.

For example:

```
long outputs[2];
meiSqNodeDigitalOutGet( node0, 32, outputs );
```

node	a handle to a SynqNet node object.
bitCount	how many bits are to be copied.
***state**	points to where the data is held in the user's program.

Return Values	
MPIMessageOK	if *SqNodeDigitalOutGet* successfully gets the state of all the digital outputs on a SynqNet node.

See Also

meiSqNodeDigitalOutSet

meiSqNodeDigitalOutSet

Declaration

```
long meiSqNodeDigitalOutSet(MEISqNode     node,
                            long          bitCount,
                            long          *state);
```

Required Header

stdmei.h

Description

SqNodeDigitalOutSet sets a set of digital outputs on a SynqNet node. The ***bitCount*** argument defines how may output bits are to be set. The memory holding the new output states are pointed to by the ***state*** argument. You can find out the maximum number of digital output bits on each node using the meiSqNodeInfo function.

For example:

```
long outputs[2];
outputs[0] = 0x0100;
outputs[1] = 0x0080;
meiSqNodeDigitalOutSet( node0, 32, outputs );
```

node	a handle to a SynqNet node object.
bitCount	how many bits are to be set.
***state**	pointer to where the desired output bits are held.

Return Values

MPIMessageOK	if *SqNodeDigitalOutSet* successfully sets the state of all the digital outputs on a SynqNet node.

See Also

meiSqNodeDigitalOutGet

meiSqNodeDownload

Declaration

```
long meiSqNodeDownload(MEISqNode                    node,
                       MEISqNodeDownloadParams      *params);
```

Required Header

stdmei.h

Description

SqNodeDownload SqNodeDownload reads a binary image from a file and writes it into a SynqNet node's non-volatile storage. SynqNet nodes may support one or more drive interfaces. SqNodeDownload can also write binary images to a drives' non-volatile storage if the drive interface hardware supports a communication channel to the drive processor. The drive interface(s) for a SynqNet node are indexed by a number (0, 1, 2, etc.).

The SynqNet node binary files are node specific. Please see the Node Binary Files: Product Table.

The SynqNet drive binary files are drive specific. The SqNodeLib includes the drive specific code necessary to support various hardware download protocols. Please see the drive manufacturer's documentation for details. Use meiSqNodeInfo(…), to determine if the SynqNet node supports a drive interface and it's type.

The binary download process requires a significant amount of time, probably between 5 to 30 seconds, depending on the node/drive type and file size. A callback function pointer is provided in the MEISqNodeDownloadParams structure for the calling application to monitor the download progress.

node	a handle to a SynqNet node object
***params**	a pointer to the download parameters structure.

Return Values

MPIMessageOK	if *SqNodeNumber* successfully reads the network number

See Also

meiSqNodeInfo | meiSynqNetInfo | MEISqNodeDownloadParams | MEISqNodeChannel | MEISqNodeCallback

meiSqNodeFlashErase

Declaration

```
long meiSqNodeFlashErase(MEISqNode      sqNode);
```

Required Header

stdmei.h

Description

SqNodeFlashErase brings the SynqNet network down to discovery mode, sends a service command down to the node that erases its runtime flash, and leaves the network down. The next time the network is brought up to Synq mode the node will be running off its boot image.

node	a handle to a SynqNet node object

Return Values

MPIMessageOK	if *SqNodeFlashErase* successfully brings the SynqNet network down to discovery mode, sends a service command down to the node that erases its runtime flash, and leaves the network down.

See Also

meiSqNodeFpgaFileNameVerify

Declaration

```
long meiSqNodeFpgaFileNameVerify(MEISqNode   sqNode,
                                 char*       fileName);
```

Required Header

stdmei.h

Description

meiSqNodeFpgaFileNameVerify verifies that the filename provided is compatible with a given sqNode.

sqNode	a handle to an SqNode object.
fileName	a pointer to a string containing the name of an SqNode image file.

Return Values

MPIMessageOK	If the FPGA image file is compatible with the specified SqNode.
MEISqNodeMessageFILE_NODE_MISMATCH	If the FPGA image file is NOT compatible with the specified SqNode.

See Also

meiSqNodeVerify

Declaration

```
long meiSqNodeVerify(MEISqNode                  node,
                     MEISqNodeDownloadParams    *params);
```

Required Header

stdmei.h

Description

SqNodeVerify verifies that the runtime image on a sqNode matches the data contained in a provided image file.

node	a handle to SqNode object.
***params**	a pointer to parameters used in the verify routine.

Return Values

MPIMessageOK	If the *meisqNodeVerify* successfully verifies that there is a matching runtime image.
MEISqNodeMessageVERIFY_FAIL	If the file provided does not match the runtime image on the node.

See Also

meiSqNodeEventNotifyGet

Declaration

```
long meiSqNodeEventNotifyGet(MEISqNode        node,
                             MPIEventMask     *eventMask,
                             void             *external);
```

Required Header

stdmei.h

Description

SqNodeEventNotifyGet reads the event mask (that specifies the event types for which host notification has been requested) to the location pointed to by ***eventMask***, and also writes it into the implementation specific location pointed to by ***external***. (if external is not NULL).

Use the event mask macros mpiEventMaskGET(…), mpiEventMaskBitGET(…), etc. to decode the eventMask.

The event notification data in external is in addition to the event notification data in eventMask. If external is NULL, the event notification data will not be copied to the external pointer.

XMP Only

external either points to a structure of type **MEIEventNotifyData{…}** or is NULL.

node	a handle to a SynqNet node object
***eventMask**	pointer to an event mask, whose bits are defined by the MPI/MEIEventType enumerations.
***external**	pointer to external

Return Values

MPIMessageOK	if *SqNodeEventNotifyGet* successfully reads the event mask
MPIMessageARG_INVALID	if the eventMask pointer is NULL

See Also

MPI/MEIEventType | MEIEventNotifyData | MEIEventStatusInfo

meiSqNodeEventNotifySet

Declaration

```
long meiSqNodeEventNotifySet(MEISqNode      node,
                             MPIEventMask   eventMask,
                             void           *external);
```

Required Header

stdmei.h

Description

SqNodeEventNotifySet requests host notification of the event(s) that are generated by SqNode and specified by ***eventMask***, and also specified by the implementation specific location pointed to by ***external*** (if external is not NULL).

Use the event mask macros meiEventMaskSQNODE(…), mpiEventMaskSET(…), mpiEventMaskBitSET(…), mpiEventMaskCLEAR(…), etc. to create the eventMask.

The event notification data in external is in addition to the event notification data in eventMask. If external is NULL, the event notification data will not be copied to the external pointer.

XMP Only

external either points to a structure of type **MEIEventNotifyData{…}** or is NULL.

The MEIEventNotifyData{…} structure is an array of controller addresses, whose contents are placed into the MEIEventStatusInfo{…} structure (of all events generated by this object).

node	a handle to a SynqNet node object
eventMask	pointer to an event mask, whose bits are defined by the MPI/MEIEventType enumerations.
***external**	pointer to external

Return Values

MPIMessageOK	if *SynqNetEventNotifySet* successfully writes the event mask
MPIMessageARG_INVALID	if the eventMask pointer is NULL

See Also

MEI/MPIEventType | MEIEventNotifyData | MEIEventStatusInfo

meiSqNodeEventReset

Declaration

```
long meiSqNodeEventReset(MEISqNode      sqNode,
                         MPIEventMask   eventMask);
```

Required Header

stdmei.h

Description

SqNodeEventReset is a method used to reset events that have been latched on a node. Events that can be reset by this method include:

See MEIEventType.

```
/* SqNode events */
MEIEventTypeSQNODE_IO_ABORT,
MEIEventTypeSQNODE_NODE_DISABLE,
MEIEventTypeSQNODE_NODE_ALARM,
MEIEventTypeSQNODE_ANALOG_POWER_FAULT,
MEIEventTypeSQNODE_USER_FAULT,
MEIEventTypeSQNODE_NODE_FAILURE,
```

sqNode	a handle to a SynqNet node object
eventMask	pointer to an event mask, whose bits are defined by the MPI/MEIEventType enumerations.

Return Values

MPIMessageOK	if *SynqNetEventReset* successfully writes the event mask
MPIMessageARG_INVALID	if the eventMask pointer is NULL

See Also

meiSqNodeMemory

Declaration

```
long meiSqNodeMemory(MEISqNode     node,
                     void          **memory);
```

Required Header

stdmei.h

Description

SqNodeMemory writes an address (that can be used to access SqNode memory) to the contents of memory. This address (or an address calculated from it) can be passed as the src argument to mpiSqNodeMemoryGet(…) or the dst argument to mpiSqNodeMemorySet(…).

node	a handle to a SynqNet node object
****memory**	a pointer to an SqNode memory address.

Return Values

MPIMesssageOK	if *SqNodeMemory* successfully writes the SqNode memory address to the contents of memory.

See Also

meiSqNodeMemoryGet | meiSqNodeMemorySet

meiSqNodeMemoryGet

Declaration

```
long meiSqNodeMemoryGet(MEISqNode    node,
                        void         *dst,
                        void         *src,
                        long         count);
```

Required Header

stdmei.h

Description

SqNodeMemoryGet reads count bytes of an SqNode's memory, starting from address ***src*** and writes it to application memory, starting at address ***dst***.

node	a handle to a SynqNet node object
***dst**	pointer to the destination address in application memory
***src**	pointer to the source address in SqNode memory
count	number of bytes to copy

Return Values

MPIMesssageOK	if *SqNodeMemoryGet* successfully copies data from SqNode memory to application memory.

See Also

meiSqNodeMemory | meiSqNodeMemorySet

meiSqNodeMemorySet

Declaration

```
long meiSqNodeMemorySet(MEISqNode    node,
                        void         *dst,
                        void         *src,
                        long         count);
```

Required Header

stdmei.h

Description

SqNodeMemorySet reads count bytes of application memory, starting from address ***src*** and writes it to an SqNode's memory, starting at address ***dst***.

node	a handle to a SynqNet node object
***dst**	pointer to the destination address in SqNode memory
***src**	pointer to the source address in application memory
count	number of bytes to copy

Return Values

MPIMesssageOK	if *SqNodeMemorySet* successfully copies data from application memory to SqNode memory.

See Also

meiSqNodeMemory | meiSqNodeMemoryGet

meiSqNodeControl

Declaration

```
meiSqNodeControl(MEISqNode node);
```

Required Header

stdmei.h

Description

SqNodeControl returns a handle to the control object associated with the SqNode object.

node	a handle to a SynqNet node object

Return Values

MPIControl	a handle to a control object
MPIHandleVOID	if node is not valid

See Also

meiSqNodeCreate | mpiControlCreate

meiSqNodeNumber

Declaration

```
long meiSqNodeNumber(MEISqNode     node,
                     long          *number);
```

Required Header

stdmei.h

Description

SqNodeNumber reads the index of a SynqNet ***node*** and writes it into the contents of a long pointed to by ***number***. Each SqNode associated with a controller is indexed by a identification number (0, 1, 2, etc.).

node	a handle to a SynqNet node object
***number**	a pointer to the index of a SynqNet node.

Return Values

MPIMessageOK	if *SqNodeNumber* successfully reads the network number

See Also

meiSynqNetInfo | meiSynqNetNumber

MEISqNodeCallback

MEISqNodeCallback

```
typedef long (*MEISqNodeCallback)(long percentage);
```

Description

SqNodeCallback is a pointer to a function, which can be used to monitor the meiSqNodeDownload(…) progress.

percentage	The portion (from 0% to 100%) of memory download that has been completed.

See Also

meiSqNodeDownload

MEISqNodeChannel

MEISqNodeChannel

```
typedef enum MEISqNodeChannel {
    MEISqNodeChannelDRIVE0,
    MEISqNodeChannelDRIVE1,
    MEISqNodeChannelDRIVE2,
    MEISqNodeChannelDRIVE3,
    MEISqNodeChannelDRIVE4,
    MEISqNodeChannelDRIVE5,
    MEISqNodeChannelDRIVE6,
    MEISqNodeChannelDRIVE7,
    MEISqNodeChannelNODE,

} MEISqNodeChannel;
```

Description

SqNodeChannel is an enumeration of communication interfaces to a node. All SynqNet nodes support a single NODE channel to the network interface device. SynqNet nodes may support one or more drive channels to a drive processor. DRIVE channels are indexed by an enumeration (DRIVE0, DRIVE1, DRIVE2, etc.).

MEISqNodeChannelDRIVE0	interface to drive number 0
MEISqNodeChannelDRIVE1	interface to drive number 1
MEISqNodeChannelDRIVE2	interface to drive number 2
MEISqNodeChannelDRIVE3	interface to drive number 3
MEISqNodeChannelDRIVE4	interface to drive number 4
MEISqNodeChannelDRIVE5	interface to drive number 5
MEISqNodeChannelDRIVE6	interface to drive number 6
MEISqNodeChannelDRIVE7	interface to drive number 7
MEISqNodeChannelNODE	interface to the node device

See Also

meiSqNodeCommand | meiSqNodeDownload

MEISqNodeCmdType

MEISqNodeCmdType

```
typedef enum MEISqNodeCmdType {
    MEISqNodeCmdTypeREAD,
    MEISqNodeCmdTypeWRITE,
} MEISqNodeCmdType;
```

Description

SqNodeCmdType is an enumeration of service command types to send to a node or drive.

MEISqNodeCmdTypeREAD	read data
MEISqNodeCmdTypeWRITE	write data

See Also

meiSqNodeCommand | MEISqNodeCmdHeader

MEISqNodeCmdHeader

MEISqNodeCmdHeader

```
typedef struct MEISqNodeCmdHeader {
    MEISqNodeChannel    channel;    /* internal node destination */
    MEISqNodeMemory     memory;
    MEISqNodeDataSize   size;
    MEISqNodeCmdType    type;       /* read/write command */
} MEISqNodeCmdHeader;
```

Description

The **SqNodeCmdHeader** structure specifies the service command communication interface to the device, the memory region on the device to access, the data size, and type.

channel	Communication interface to a device. See MEISqNodeChannel.
memory	The memory region to access. See MEISqNodeMemory.
size	The length of data to send or receive. See MEISqNodeDataSize.
type	The service command action (read or write). See MEISqNodeCmdType.

See Also

meiSqNodeCommand | MEISqNodeCommand

MEISqNodeCommand

MEISqNodeCommand

```
typedef struct MEISqNodeCommand {
    MEISqNodeCmdHeader  header;
    unsigned long       address;  /* command resgisters */
    unsigned long       data;     /* command data */
} MEISqNodeCommand;
```

Description

The **SqNodeCommand** structure specifies the service command. It includes a header structure (channel, memory, size, and type), a destination address, and the data.

header	A structure that specifies the channel, memory region, and data size. See MEISqNodeCmdHeader.
address	A memory location to read or write the data.
data	The command data to send.

See Also

meiSqNodeCommand | MEISqNodeResponse

MEISqNodeConfig

MEISqNodeConfig

```
typedef struct MEISqNodeConfig {
    MEISqNodeConfigAlarm          nodeAlarm;
    MEISqNodeConfigIoAbort        ioAbort;
    MEISqNodeConfigPacketError    upStreamError;
    MEISqNodeConfigPacketError    downStreamError;
    MEISqNodeConfigUserFault      userFault;
    MEISqNodeFeedbackSecondary    feedbackSecondary
                                  [MEISqNodeMaxFEEDBACK_SECONDARY];
} MEISqNodeConfig;
```

Description

The **SqNodeConfig** structure specifies the SynqNet node configurations.

nodeAlarm	A structure to configure a SynqNet node's trigger conditions for the Node Alarm output bit. The node alarm circuit is node specific, but is intended to notify users when the node has a problem. The nodeAlarm occurs on an ioAbort, DedicatedInAMP_FAULT (one per motor/drive) or an FPGA fails to operate with run-time code. See MEISqNodeConfigIoAbort and MEISqNodeConfigNodeAlarm for the trigger configurations.
ioAbort	A structure to configure a SynqNet node's trigger conditions for an I/O Abort action. When an ioAbort is triggered, the SynqNet node's outputs are disabled (set to the power-on condition). See MEISqNodeConfigIoAbort for the trigger configurations.
upStreamError	A structure used to configure the fault and failure limits for the upstream SynqNet packets. The controller keeps track of how many bad packets are received from the Node and performs the appropriate actions when the fault and fail limits are reached. See MEISqNodeConfigPacketError for appropriate ranges and resulting actions.
downStreamError	A structure used to configure the fault and failure limits for the downstream SynqNet packets. The node keeps track of how many bad packets are received from the controller and performs the appropriate actions when the fault and fail limits are reached. See MEISqNodeConfigPacketError for appropriate ranges and resulting actions.
userFault	A structure to configure the trigger conditions for a SynqNet node user fault. When a user fault is triggered, a node ioAbort and/or an action on each motor will occur. See MEISqNodeConfigUserFault for the trigger configurations.

feedbackSecondary	A structure to configure the secondary encoder resources on the node. See MEISqNodeFeedbackSecondary for more information.

See Also

meiSqNodeConfigGet | meiSqNodeConfigSet | MEISqNodeConfigPacketError

MEISqNodeConfigIoAbort

MEISqNodeConfigIoAbort

```
typedef struct MEISqNodeConfigIoAbort {
    MEISqNodeConfigTrigger   synqLost;     /* communication error */
    MEISqNodeConfigTrigger   nodeDisable;  /* external input */
    MEISqNodeConfigTrigger   powerFault;   /* analog power failure */
    long                     userFault;    /* TRUE = user fault causes ioabort */
} MEISqNodeConfigIoAbort;
```

Description

The **SqNodeConfigIoAbort** structure specifies the SynqNet node configurations to generate an I/O Abort action. When an ioAbort is triggered, the SynqNet node's outputs are disabled (set to the power-on condition). The IoAbort is triggered when any one or more of the following enabled configurations occur:

synqLost	Occurs when a SynqNet node drops out of SYNQ (cyclic) mode to SYNQ_LOST mode. See MEISqNodeConfigTrigger.
nodeDisable	An input bit to the SynqNet node. The node disable circuit is node specific, but is intended to shutdown the node via the IoAbort. See MEISqNodeConfigTrigger.
powerFault	An input bit to the SynqNet node. The power fault circuit is node specific, but is usually connected to an analog power monitor. Typically, when the DAC power or other analog component power is either too high or drops below a threshold, the power fault is triggered. Please see the node/drive manufacturer's documentation for details. See MEISqNodeConfigTrigger.
userFault	A user configurable trigger condition. A value of TRUE enables the trigger, FALSE disables the trigger.

See Also

meiSqNodeConfigGet | meiSqNodeConfigSet

MEISqNodeConfigAlarm

MEISqNodeConfigAlarm

```
typedef struct MEISqNodeConfigAlarm {
    unsigned long   mask;              /* One bit per drive/motor. Triggered by
                                          the MEIMotorDedicatedInAMP_FAULT input. */
    long            notCyclicEnable;   /* allow nodeAlarm to be asserted when the
node
                                          is not in cyclic mode */
    long            ioAbortEnable;     /* allow ioAbort to assert nodeAlarm     */
} MEISqNodeConfigAlarm;
```

Description

The **SqNodeConfigAlarm** structure specifies the input trigger for the SynqNet node alarm output. The input triggers are the MEIMotorDedicatedInAMP_FAULT bits for each motor/drive interface.

mask	Each bit in the mask represents a motor or drive interface. For example, a value of 0x3 will trigger the node alarm output when either motor 0's OR motor 1's MEIMotorDedicatedInAMP_FAULT bit is TRUE.
notCyclicEnable	This Boolean variable is used to specify whether or not a node can receive an alarm when it is not in cyclic mode. TRUE = node alarm can be asserted in any mode. FALSE = node alarm can only be asserted in cyclic mode.
ioAbortEnable	This Boolean variable is used to specify the effect an I/O abort will have on the node alarm output. TRUE = an I/O abort will trigger a node alarm. FALSE = an I/O abort will not necessarily trigger a node alarm.

See Also

meiSqNodeConfigGet | meiSqNodeConfigSet | MEIMotorDedicatedIn

MEISqNodeConfigPacketError

MEISqNodeConfigPacketError

```
typedef struct MEISqNodeConfigPacketError {
    long faultLimit;    /* 0 - 255 */
    long failLimit;     /* 0 - 255 */
} MEISqNodeConfigPacketError;
```

Description

The **SqNodeConfigPacketError** structure specifies the limit conditions for SynqNet node packet rate errors.

faultLimit	Packet error rate limit to generate a fault. When the faultLimit is reached, the node will attempt to recover by switching the port used for data transmission. Valid range is 0 to 255. The value saturates at 255.
failLimit	Packet error rate limit to generate a failure. When the failLimit is reached, the node will drop to the SYNQ_LOST state and disable its outputs. Valid range is 0 to 255. The value saturates at 255.

See Also

meiSqNodeConfigGet | meiSqNodeConfigSet

MEISqNodeConfigTrigger

MEISqNodeConfigTrigger

```
typedef struct MEISqNodeConfigTrigger {
    long    enable;
    long    invert;
} MEISqNodeConfigTrigger;
```

Description

The **SqNodeConfigTrigger** structure specifies trigger configurations.

enable	Enables or disables the trigger. A value of TRUE enables the trigger, FALSE disables the trigger.
invert	Normal or inverted trigger polarity. A value of FALSE indicates normal polarity, TRUE indicates inverted polarity.

See Also

MEISqNodeConfigIoAbort

MEISqNodeConfigUserFault

MEISqNodeConfigUserFault

```
typedef struct MEISqNodeConfigUserFault {
    long            *addr;    /* firmware addr */
    unsigned long   mask;
    unsigned long   pattern;
} MEISqNodeConfigUserFault;
```

Description

The **SqNodeConfigUserFault** structure specifies the trigger conditions for a user defined input. The trigger condition can be configured for any controller address. When the masked value at the specified addr matches the pattern, the user fault is active. The user fault triggers a SynqNet node IoAbort if the userFault flag in MEISqNodeConfigIoAbort{…} is enabled. The user fault also triggers an action for all the motors associated with the node. The userFaultAction is specified in the MEIMotorConfig{…} structure.

***addr**	A pointer to a controller address.
mask	A bit mask ANDed with the value at the controller address.
pattern	A bit pattern compared to the masked value at the controller address. When the masked value equals the pattern, the user trigger is TRUE.

See Also

MEISqNodeConfigIoAbort | MEIMotorConfig | meiSqNodeConfigGet | meiSqNodeConfigSet

MEISqNodeDataSize

MEISqNodeDataSize

```
typedef enum MEISqNodeDataSize {  /* read/write data width */
    MEISqNodeDataSize8BIT,
    MEISqNodeDataSize16BIT,
    MEISqNodeDataSize24BIT,
    MEISqNodeDataSize32BIT,
} MEISqNodeDataSize
```

Description

SqNodeDataSize is an enumeration of service command data lengths. The data length is in units of bits.

MEISqNodeDataSize8BIT	8 bit data length
MEISqNodeDataSize16BIT	16 bit data length
MEISqNodeDataSize24BIT	24 bit data length
MEISqNodeDataSize32BIT	32 bit data length

See Also

meiSqNodeCommand | MEISqNodeCmdHeader

MEISqNodeDownloadParams

MEISqNodeDownloadParams

```
typedef struct MEISqNodeDownloadParams {
    char                *filename;
    MEISqNodeChannel    channel;
    MEISqNodeCallback   callback;
} MEISqNodeDownloadParams;
```

Description

The **SqNodeDownloadParams** structure specifies the parameters for downloading a binary image to a SynqNet node.

***filename**	A pointer to a file name. The file contains a header and binary code/data. Files are node/drive specific. Please see the Node Binary Files: Product Table or the drive manufacturer's documentation for the drive binary files.
channel	A communication interface to a node's logic device or drive processor. See MEISqNodeChannel.
callback	A pointer to a callback function, to monitor the download progress. See MEISqNodeCallback.

See Also

meiSqNodeDownload

MEISqNodeDriveInfo

MEISqNodeDriveInfo

```
typedef struct MEISqNodeDriveInfo {
    char    firmwareVersion[MEISqNodeDriveParamMAX_STRING_LENGTH];
} MEISqNodeDriveInfo;
```

Description

SqNodeDriveInfo is a structure containing information about a specified drive.

firmwareVersion	A string containing drive firmware version information that is retrieved from the Drive Processor on the Node.

See Also

MEISqNodeDriveMonitor

MEISqNodeDriveMonitor

```
typedef struct MEISqNodeDriveMonitor {
    MEISqNodeDriveMonitorDataType    type;
    MEISqNodeDriveMonitorData        data;
} MEISqNodeDriveMonitor;
```

Description

The **SqNodeDriveMonitor** structure specifies the data to be placed in the monitor field by the drive.

type	The drive data is selected by its type. See MEISqNodeDriveMonitorDataType.
data	The location of the drive data. See MEISqNodeDriveMonitorData.

See Also

MEISqNodeMonitorValue | meiSqNodeDriveMonitorConfigGet | meiSqNodeDriveMonitorConfigSet

MEISqNodeDriveMonitorConfig

MEISqNodeDriveMonitorConfig

```
typedef struct MEISqNodeDriveMonitorConfig {
    MEISqNodeDriveMonitor    monitorA;
    MEISqNodeDriveMonitor    monitorB;
    MEISqNodeDriveMonitor    monitorC;
}MEISqNodeDriveMonitorConfig;
```

Description

The **SqNodeDriveMonitorConfig** structure specifies the configuration for the drive monitor fields.

monitorA	configuration for drive monitor A
monitorB	configuration for drive monitor B
monitorC	configuration for drive monitor C

See Also

MEISqNodeDriveMonitor | meiSqNodeDriveMonitorConfigGet | meiSqNodeDriveMonitorConfigSet |

MEISqNodeDriveMonitorData

MEISqNodeDriveMonitorData

```
typedef union {
    long    index;     /* the values for these parameters are drive specific    */
    long    address;   /* and can be found in the appropriate drive modules     */
} MEISqNodeDriveMonitorData;
```

Description

SqNodeDriveMonitorData specifies the location of the monitor data. Drive data can be specified by either an index or an address. The location is drive specific. Please see the drive manufacturer's documentation.

index	A drive specific value to select a monitor data field from a table.
address	A drive specific memory address to select the monitor data.

See Also

meiSqNodeDriveMonitorConfigGet | meiSqNodeDriveMonitorConfigSet

MEISqNodeDriveMonitorDataType

MEISqNodeDriveMonitorDataType

```
typedef enum MEISqNodeDriveMonitorDataType {
    MEISqNodeDriveMonitorDataTypeINDEX,
    MEISqNodeDriveMonitorDataTypeADDRESS,
} MEISqNodeDriveMonitorDataType;
```

Description

SqNodeDriveMonitorDataType is an enumeration of monitor data selection types.

MEISqNodeDriveMonitorDataTypeINDEX	Select monitor data using an index to a table.
MEISqNodeDriveMonitorDataTypeADDRESS	Select monitor data using an address.

See Also

meiSqNodeDriveMonitorConfigGet | meiSqNodeDriveMonitorConfigSet

MEISqNodeDriveParamCallback

MEISqNodeDriveParamCallback

```
typedef void (*MEISqNodeDriveParamCallback)
             (MEISqNodeDriveParamCallbackType  type,
              char                            *name,
              long                             number,
              char                            *value);
```

Description

In the **SqNodeDriveParamCallback** structure, the function's pointer type defines a function that can be passed to the meiSqNodeDriveParamFileSet function. The meiSqNodeDriveParamFileSet function will call this type of function to report progress or warnings. A NULL value for the callback pointer will disable the callback feature.

type	the type of event that caused the callback function to be called.
name	name of the drive parameter.
number	drive parameter index.
value	the value of the drive parameter.

See Also

meiSqNodeDriveParamFileSet

MEISqNodeDriveParamCallbackType

MEISqNodeDriveParamCallbackType

```
typedef enum {
    MEISqNodeDriveParamCallbackTypeCHANGED,
    MEISqNodeDriveParamCallbackTypeSET_FAILED,
} MEISqNodeDriveParamCallbackType;
```

Description

The **SqNodeDriveParamCallbackType** enumeration is used by the MEISqNodeDriveParamCallback function to describe the type of event that caused the callback function to be called.

MEISqNodeDriveParamCallbackTypeCHANGED	This indicates that the new drive parameter value is different to the current parameter value.
MEISqNodeDriveParamCallbackTypeSET_FAILED	Setting this drive parameter failed.

See Also

MEISqNodeDriveParamCallback

MEISqNodeFeedbackSecondary

MEISqNodeFeedbackSecondary

```
typedef struct MEISqNodeFeedbackSecondary {
    long    motorIndex;
} MEISqNodeFeedbackSecondary;
```

Description

The **SqNodeFeedbackSecondary** structure allows for configuration of the secondary feedback resources on a SynqNet node.

motorIndex	Indicates motorIndex on the node to which the secondary feedback resource is mapped. This value is MEISqNodeNOT_AVAILABLE if the secondary feedback resource does not exist on the node hardware

See Also

MEISqNodeConfig

MEISqNodeFileName

MEISqNodeFileName

```
typedef struct MEISqNodeFileName{
    char  fileName[MEISqNodeFILENAME_MAX];
}MEISqNodeFileName;
```

Description

FileName is used in methods that retrieve filenames from the MPI.

fileName	String containing the name of an SqNode image file.

See Also

MEISqNodeFpgaType

MEISqNodeFpgaType

```
typedef enum MEISqNodeFpgaType {
    MEISqNodeFpgaTypeBOOT,
    MEISqNodeFpgaTypeRUN_TIME,
} MEISqNodeFpgaType
```

Description

sqNodeFpgaType is an enumeration of FPGA types.

MEISqNodeFpgaTypeBOOT	The FPGA is operating with a boot image. The boot image only supports basic SynqNet communication. Use meiSqNodeDownload(…) to download the runtime image to the SynqNet node.
MEISqNodeFpgaTypeRUN_TIME	The FPGA is operating with a runtime image.

See Also

meiSqNodeInfo | MEISqNodeInfoFpga | meiSqNodeDownload

MEISqNodeInfo

MEISqNodeInfo

```
typedef struct MEISqNodeInfo {
    long                    motorCount;
    long                    driveCount;
    long                    motorOffset;
    long                    feedbackSecondaryCount;
    MEISqNodeInfoId         id;
    MEISqNodeInfoFpga       fpga;
    MEISqNodeInfoNetwork    network;
    MEISqNodeInfoIo         io;
} MEISqNodeInfo;
```

Description

The **SqNodeInfo** structure contains static data stored for the SynqNet node. The motor objects are indexed sequentially across all the SynqNet nodes associated with each network. Each motor on a controller has a unique number.

motorCount	The number of motors that the SynqNet node supports.
driveCount	The number of drives interfaces that the SynqNet node supports.
motorOffset	The starting number for the first motor on the SynqNet node.
feedbackSecondaryCount	The number of auxillary feedbacks on the node.
id	A structure that contains identification data for the SynqNet node. See MEISqNodeInfoId.
fpga	A structure that contains identification data for the SynqNet node FPGA. See MEISqNodeInfoFpga.
network	A structure that contains network interface information for the SynqNet node. See MEISqNodeInfoNetwork.
io	A structure that returns how many of each type of node I/O this node supports.

See Also

meiSqNodeInfo

MEISqNodeInfoId

MEISqNodeInfoId

```
typedef struct MEISqNodeInfoId {
    unsigned  long   nodeType;   /* product/mfg code   */
    char             *nodeName;  /* product/mfg string */
    unsigned  long   option;     /* product option code*/
    unsigned  long   switchId;   /* rotary switch id   */
    unsigned  long   unique;     /* unique id code     */

    long             exactMatch; /* TRUE/FALSE         */

    char             serialNumber[MEISqNodeID_CHAR_MAX];
    char             modelNumber[MEISqNodeID_CHAR_MAX];
    char             manufacturerData[MEISqNodeManufacturerDATA_CHAR_MAX];
} MEISqNodeInfoId;
```

Description

The **SqNodeInfoId** structure contains identification data for the SynqNet node.

All nodes by all manufacturers will have ***nodeType*** and ***unique*** numbers that should generate a unique identification for each node on the SynqNet network.. Although some node manufacturers may opt to leave the ***serialNumber*** and ***modelNumber*** fields blank, you can still identify and distinguish a node by comparing the ***nodeType*** and ***unique*** numbers. The ***nodeType*** number is also represented by a unique text string *nodeName*.

nodeType	A 32 bit value that identifies the node hardware. The upper 16 bits represent the manufacturer of the SynqNet node hardware. Each manufacturer has a unique value. The lower 16 bits represent the SynqNet node product type. The SynqNet node manufacturer determines a unique value to track a product series. Typically, the node type value is displayed in hex.
***nodeName**	A string that represents the SynqNet nodeType. The nodeName string matches the name of the SqNodeLib node specific header file.
option	The product option code within a product series.
switchId	If a node/drive have an physical address switch on its faceplate, switchId will contain the value to which the switch is set. If an ID switch is not supported by a node, this value will be set to -1 (0xFFFFFFFF).

MEISqNodeInfoIo

MEISqNodeInfoIo

```
typedef struct MEISqNodeInfoIo {
   long  digitalInCount;
   long  digitalOutCount;
   long  analogInCount;
   long  analogOutCount;
} MEISqNodeInfoIo;
```

Description

The **SqNodeInfoIo** structure lists the number of digital and analog inputs that are supported by a SynqNet node.

digitalInCount	The number of digital inputs on a SynqNet node.
digitalOutCount	The number of digital outputs on a SynqNet node.
analogInCount	The number of analog inputs on a SynqNet node.
analogOutCount	The number of analog outputs on a SynqNet node.

See Also

meiSqNodeInfo

MEISqNodeInfoFpga

MEISqNodeInfoFpga

```
typedef struct MEISqNodeInfoFpga {
    MEISqNodeFpgaType          type;
    unsigned           long    vendorDevice;
    unsigned           long    version;
    long                       defaultVersion; /* TRUE/FALSE */
} MEISqNodeInfoFpga;
```

Description

The **SqNodeInfoFpga** structure contains identification data for the SynqNet node FPGA.

type	The FPGA type. See MEISqNodeFpgaType.
vendorDevice	A 32 bit value that identifies the FPGA image. The upper 16 bits represent the manufacturer of the SynqNet node network interface device. Each manufacturer has a unique vendor value. The lower 16 bits represent the SynqNet node network interface component. The device is typically an FPGA (could be an ASIC). If the device is an FPGA, the vendorDevice information is stored in the FPGA binary image. Each device for a particular vendor has a unique device value. Typically, the vendorDevice value is displayed in hexadecimal format.
version	A 32 bit value that represents the revision of the device. The upper 16 bits represent the SynqNet network interface revision. The lower 16 bits represent the device revision. Typically, the version value is displayed in hexadecimal format.
defaultVersion	Indicates if the default version of the SqNode FPGA image is loaded on this node. The defaultVersion defines the version of the SynqNet node FPGA image that was built and tested with the current version of the MPI.

See Also

meiSqNodeInfo | MEISqNodeInfoId | MPI/SynqNet FPGA Compatibility Check

unique	A 32 bit value that identifies the node. It is an unsigned long. The SynqNet node manufacturer determines this unique value to track a single product. This is useful to determine when individual nodes of the same type are switched or replaced on a SynqNet network. **NOTE**: It is possible for a manufacturer to use the same unique identification number for two nodes of different models. The combination of SqNode.Name (or nodeType) and SqNode.UniqueId will be unique for any given code.
exactMatch	A string that tells you if the node is running under a matched or unmatched classification. The value of meiSqNodeInfo.id.exactMatch is TRUE when all ID components have been matched to a supported configuration. The value is FALSE when running with a default (unmatched) configuration.
serialNumber	A string that represents the SynqNet node serial number. For a given node type, the serial number is unique. The SynqNet node manufacturer determines the serial number to track an individual unit.
modelNumber	A string that represents the SynqNet node model number. The SynqNet node manufacturer determines the model number.
manfacturerData	A string containing Manufacturer-specific data which is stored on the node at time of production.

See Also

meiSqNodeInfo | MEISqNodeInfoFpga

MEISqNodeInfoNetwork

MEISqNodeInfoNetwork

```
typedef struct MEISqNodeInfoNetwork {
    long    number;
    long    inPorts;
    long    outPorts;
} MEISqNodeInfoNetwork;
```

Description

The **SqNodeInfoNetwork** structure contains information about the SynqNet node's network interface.

number	An index to a SynqNet network associated with a controller.
inPorts	The number of SynqNet IN port network interfaces.
outPorts	The number of SynqNet OUT port network interfaces.

Remarks

The labeling convention for IN and OUT ports is for convenience. The hardware ports are identical. During SynqNet initialization, the node are discovered based on the OUT to IN port connections.

See Also

meiSqNodeInfo

MEISqNodeMemory

MEISqNodeMemory

```
typedef enum MEISqNodeMemory {
    MEISqNodeMemoryDATA,        /* node/drive processor RAM */
    MEISqNodeMemoryPROGRAM,     /* drive processor program memory */
    MEISqNodeMemoryIO,          /* drive I/O memory */
    MEISqNodeMemoryDRIVE,       /* direct command to drive */
} MEISqNodeMemory;
```

Description

SqNodeMemory is an enumeration of drive region types to access with a service command.

MEISqNodeMemoryDATA	node/drive processor data memory
MEISqNodeMemoryPROGRAM	drive processor program memory
MEISqNodeMemoryIO	drive I/O memory
MEISqNodeMemoryDRIVE	direct command to drive processor

See Also

MEISqNodeCmdHeader | meiSqNodeCommand

MEISqNodeMessage

MEISqNodeMessage

```
typedef enum {
    MEISqNodeMessageINVALID,
    MEISqNodeMessageCONFIG_NETWORK_MISMATCH,
    MEISqNodeMessageMAP_CONFIG_MISMATCH,
    MEISqNodeMessageNOT_IN_CONFIG_FILE,
    MEISqNodeMessageCONFIG_FILE_FORMAT_INVALID,

    MEISqNodeMessageRESPONSE_TIMEOUT,
    MEISqNodeMessageREADY_TIMEOUT,
    MEISqNodeMessageSRVC_ERROR,
    MEISqNodeMessageSRVC_UNSUPPORTED,
    MEISqNodeMessageSRVC_CHANNEL_INVALID,

    MEISqNodeMessageCMD_NOT_SUPPORTED,
    MEISqNodeMessageDISCOVERY_FAILURE,
    MEISqNodeMessageDISPATCH_ERROR,
    MEISqNodeMessageINIT_FAILURE,
    MEISqNodeMessageINTERFACE_ERROR1,
    MEISqNodeMessageFILE_NODE_MISMATCH,
    MEISqNodeMessageFILE_INVALID,
    MEISqNodeMessageINVALID_HEADER,
    MEISqNodeMessageDOWNLOAD_FAIL,
    MEISqNodeMessageVERIFY_FAIL,
    MEISqNodeMessageDOWNLOAD_NOT_SUPPORTED,
    MEISqNodeMessageVERIFY_NOT_SUPPORTED,
    MEISqNodeMessageBOOT_ROM_INVALID,
    MEISqNodeMessageINVALID_TABLE,
    MEISqNodeMessageINVALID_STR_LEN,
    MEISqNodeMessageFEEDBACK_MAP_INVAILD,
} MEISqNodeMessage;
```

Description

SqNodeMessage is an enumeration of SynqNet node error messages that can be returned by the MPI library.

MEISqNodeMessageINVALID

The SqNode type is out of range. This message code is returned by SynqNet node methods if the node type is not a member of the SQNodeLibNodeType enumeration.

MEISqNodeMessageCONFIG_NETWORK_MISMATCH

The type of map file specified in meiSqNodeDriveParamFileSet does not match the type of drive found on the network.

MEISqNodeMessageMAP_CONFIG_MISMATCH

The parameter name or number specified in meiSqNodeDriveMapParamFileSet was not valid for the specified drive.

MEISqNodeMessageNOT_IN_CONFIG_FILE

The parameter name or number specified in meiSqNodeDriveMapParamFileSet was not found.

MEISqNodeMessageCONFIG_FILE_FORMAT_INVALID

A file with an incorrect format was used in meiSqNodeDriveMapParamFileSet.

MEISqNodeMessageRESPONSE_TIMEOUT

Currently unused and is reserved for future use.

MEISqNodeMessageREADY_TIMEOUT

Currently unused and is reserved for future use.

MEISqNodeMessageSRVC_ERROR

Currently unused and is reserved for future use.

MEISqNodeMessageSRVC_UNSUPPORTED

Currently unused and is reserved for future use.

MEISqNodeMessageSRVC_CHANNEL_INVALID

Invalid service channel specified. See MEISqNodeCmdHeader.

MEISqNodeMessageCMD_NOT_SUPPORTED

The service command is not supported by the node.

MEISqNodeMessageDISCOVERY_FAILURE

Unable to discover node rescources.

MEISqNodeMessageDISPATCH_ERROR

Is the default error code returned when a node specific routine has failed. Check the node FPGA version to verify whether or not it is correct.

MEISqNodeMessageINIT_FAILURE

A node specific initialization routine was unable to successfully complete its routine. Verify that the node FPGA is the default version for your MPI version. See MEISqNodeInfoFpga.defaultVersion.

MEISqNodeMessageINTERFACE_ERROR1

This is an outdated node, which does not support the current discovery routine.

MEISqNodeMessageFILE_NODE_MISMATCH

Node type does not match the file provided for download.

MEISqNodeMessageFILE_INVALID

The file provided for download was not found or was corrupted.

MEISqNodeMessageINVALID_HEADER

The header information in the download image is invalid. Please verify firmware file to be correct and retry download. If firmware file is correct please contact firmware manufacturer.

MEISqNodeMessageDOWNLOAD_FAIL

Node firmware download failed. Verify that the firmware file is correct and retry the download.

NOTE: A network reset my be required.

MEISqNodeMessageVERIFY_FAIL

The node FPGA firmware does not match the FPGA image file.

MEISqNodeMessageDOWNLOAD_NOT_SUPPORTED

The downloading of the node firmware (FPGA) image is not supported for this node.

MEISqNodeMessageVERIFY_NOT_SUPPORTED

The Node specified for verification does not support the upload of the FPGA image. Therefore, the image cannot be verified.

MEISqNodeMessageBOOT_ROM_INVALID

The SqNode Boot Rom identification or version is not recognized by the MPI.

MEISqNodeMessageINVALID_TABLE

Invalid resource table in node module. This is a fatal error within the MPI. Please verify MPI and node FPGA versions to be correct and then contact MEI's Technical Support.

MEISqNodeMessageINVALID_STR_LEN

An attempt to write information to the node has failed due to an invalid string length.

MEISqNodeMessageFEEDBACK_MAP_INVAILD

Returned from MEISqNodeConfigSet(...) when the given secondary encoder (*n*) is not mappable to the motor on the node specified by MEISqNodeFeedbackSecondary[n].motorIndex.

See Also

meiSqNodeDriveMapParamFileSet | meiSqNodeConfigSet

MEISqNodeMonitorValue

MEISqNodeMonitorValue

```
typedef struct MEISqNodeMonitorValue {
    long    count;
    long    monitor[MEISqNodeMonitorValueIndexLAST];
} MEISqNodeMonitorValue;
```

Description

The **SqNodeMonitorValue** structure contains the data for the monitor fields read by the meiSqNodeDriveMonitor(…) method.

count	The number of monitor fields read. This specifies the size of the monitor array.
monitor	An array of monitor data fields. Each field is indexed by the MEISqNodeMonitorValueIndex enumeration.

See Also

meiSqNodeDriveMonitor | meiSqNodeDriveMonitorConfigGet | meiSqNodeDriveMonitorConfigSet

MEISqNodeMonitorValueIndex

MEISqNodeMonitorValueIndex

```
typedef enum MEISqNodeMonitorValueIndex {
    MEISqNodeMonitorValueIndexA,
    MEISqNodeMonitorValueIndexB,
    MEISqNodeMonitorValueIndexC,
    MEISqNodeMonitorValueIndexD,
} MEISqNodeMonitorValueIndex;
```

Description

SqNodeMonitorValueIndex is an enumeration of indices to node monitor values.

MEISqNodeMonitorValueIndexA	Index to node monitor value A.
MEISqNodeMonitorValueIndexB	Index to node monitor value B.
MEISqNodeMonitorValueIndexC	Index to node monitor value C.
MEISqNodeMonitorValueIndexD	Index to node monitor value D.

See Also

meiSqNodeDriveMonitor | meiSqNodeDriveMonitorConfigGet | meiSqNodeDriveMonitorConfigSet

MEISqNodeResponse

MEISqNodeResponse

```
typedef struct MEISqNodeResponse {
    unsigned long    data;    /* response data */
} MEISqNodeResponse;
```

Description

The **SqNodeResponse** structure contains the service command response data.

data	The response information from a service command. The data field is only valid for MEISqNodeCmdTypeREAD command types.

See Also

meiSqNodeCommand

MEISqNodeStatus

MEISqNodeStatus

```
typedef struct MEISqNodeStatus {
    MEISqNodeStatusPacketError   upStreamError;
    MEISqNodeStatusPacketError   downStreamError;
    MEISqNodeStatusCrcError      crcError;
    MPIEventMask                 eventMask;
} MEISqNodeStatus;
```

Description

The **SqNodeStatus** structure contains error counters and the ***eventMask*** for a SynqNet node.

upStreamError	The rate and count of bad synqNet messages received by the controller from the Node. See MEISqNodeStatusPacketError.
downStreamError	The rate and count of bad synqNet messages received by the Node from the controller. See MEISqNodeStatusPacketError.
crcError	Counters for the CRC errors. See MEISqNodeStatusCrcError.
eventMask	Array that defines the event mask bits. The array is defined as: `typedef MPIEventMaskELEMENT_TYPE MPIEEventMask[MPIEventMaskELEMENTS]` The bits are defined by the MPI/MEIEventType enumerations.

See Also

meiSqNodeStatus | meiSynqNetStatus | MEISqNodeConfig

MEISqNodeStatusCrcError

MEISqNodeStatusCrcError

```
typedef struct MEISqNodeStatusCrcError {
    long    port[MEINetworkPortLAST];
} MEISqNodeStatusCrcError;
```

Description

The **SqNodeStatusCrcError** structure contains CRC error counters for each network port. The CRC error counters are helpful for diagnosing data integrity problems. The counter increments for any CRC error on any packet received at that port (whether the packet is addressed to the node or not). The CRC error counters are cleared during network initialization.

port	An array of CRC error counters. Each network port has one CRC error counter. The valid range is 0 to 255. The value saturates at 255.

See Also

meiSqNodeStatus | MEINetworkPort

MEISqNodeStatusPacketError

MEISqNodeStatusPacketError

```
typedef struct MEISqNodeStatusPacketError {
    long    rate;
    long    count;
} MEISqNodeStatusPacketError;
```

Description

The **SqNodeStatusPacketError** structure contains packet error counters and rate counters. Each SynqNet node has a packet error counter and a packet error rate counter. Packets addressed to a node are checked for integrity.

The packet error counters are used to monitor long-term data integrity. These counters do not trigger any fault or fail actions. The packet error counter is incremented once for each missing or invalid packet. Typically, an application will periodically read the packet error counters and store the values in a log.

The packet error rate counters are used to trigger fault recovery and/or failure shutdown. The packet error rate counter is incremented for each missing or invalid packet and is decremented for 16 consecutive valid packets. Thus, the packet error rate counters can detect large errors over short periods of time or small errors over long periods of time.

rate	The packet error rate counter. The valid range is 0 to 255. The value saturates at 255.
count	The packet error counter. The valid range is 0 to 255. The value saturates at 255.

See Also

meiSqNodeStatus | MEISqNodeConfigPacketError

MEISqNodeUserData

MEISqNodeUserData

```
typedef struct MEISqNodeUserData {
    char    data[MEISqNodeUserDATA_CHAR_MAX];
}MEISqNodeUserData ;
```

Description

The **SqNodeUserData** structure is used to store the user information that is located on the SqNode.

data	User information on the SqNode used for storing SqNode identification or any other useful data. Programmable string to be used by a customer to store identification-specific information. This data is not used by the MPI and is stored in the SqNode's EEPROM.

See Also

meiSqNodeUserDataGet | meiSqNodeUserDataSet

MEISqNodeID_CHAR_MAX

MEISqNodeID_CHAR_MAX

```
#define MEISqNodeID_CHAR_MAX  (30)
```

Description

SqNodeID_CHAR_MAX defines the maximum length (number of characters) in a sqNode identification string.

See Also

MEISqNodeFILENAME_MAX

MEISqNodeFILENAME_MAX

```
#define MEISqNodeeFILENAME_MAX  (18)
```

Description

FILENAME_MAX defines the maximum size allowed for SqNode filenames.

See Also

MEISqNodeManufacturerDATA_CHAR_MAX

MEISqNodeManufacturerDATA_CHAR_MAX

```
#define MEISqNodeManufacturerDATA_CHAR_MAX  (0x10)
```

Description

SqNodeManufacturerDATA_CHAR_MAX defines the maximum number of characters stored in the Manufacturer's Data field.

See Also

MEISqNodeMaxFEEDBACK_SECONDARY

MEISqNodeMaxFEEDBACK_SECONDARY

```
#define MEISqNodeMaxFEEDBACK_SECONDARY  (MEISqNodeMaxMOTORS)
```

Description

SqNodeMaxFEEDBACK_SECONDARY defines the maximum number of secondary feedback devices per SynqNet node.

See Also

MEISqNodeConfig | MEISqNodeFeedbackSecondary

MEISqNodeMaxMOTORS

MEISqNodeMaxMOTORS

```
#define MEISqNodeMaxMOTORS  (MEIXmpMotorsPerBlock)
```

Description

SqNodeMaxMOTORS defines the maximum number of motor objects supported on a single SynqNet node. This define should be used instead of the MEIXmpMotorsPerBlock definition in xmp.h. It is recommended that applications avoid programming to defines or structures in xmp.h

See Also

SqNode Objects

MEISqNodeNOT_AVAILABLE

MEISqNodeNOT_AVAILABLE

```
#define MEISqNodeNOT_AVAILABLE  (-1)
```

Description

SqNodeNOT_AVAILABLE defines a possible value for MEISqNodeConfig.feedbackSecondary[n].motorIndex.
If motorIndex = MEISqNodeNOT_AVAILABLE, then a secondary feedback device does not exist on the hardware.

See Also

MEISqNodeConfig | MEISqNodeFeedbackSecondary

MEISqNodeSTATUS_NOT_AVAILABLE

MEISqNodeSTATUS_NOT_AVAILABLE

```
#define  MEISqNodeSTATUS_NOT_AVAILABLE  (-1)
```

Description

With exception to MEISqNodeStatus.eventMaskValue, the **SqNodeSTATUS_NOT_AVAILABLE** value is assigned to all Node status variables when the Status is not available as a result of lost communication with the node.

See Also

MEISqNodeStatus

MEISqNodeUserDATA_CHAR_MAX

MEISqNodeUserDATA_CHAR_MAX

```
#define MEISqNodeUserDATA_CHAR_MAX  (0x10)
```

Description

SqNodeUserDATA_CHAR_MAX defines the maximum number of characters in the User defined string, stored on the SqNode.

See Also

MEIDriveMapParamMAX_STRING_LENGTH

MEIDriveMapParamMAX_STRING_LENGTH

```
#define  MEIDriveMapParamMAX_STRING_LENGTH  (256)
```

Description

The **DriveMapParamMAX_STRING_LENGTH** macro defines the maximum length of a string that can be read from or written to a drive parameter.

See Also

MEIFPGARINCONREV

MPIFPGARINCONREV

```
#define MEIFPGARINCONREV    (0x0221)
```

Description

MEIFPGARINCONREV defines the version of the SynqNet controller FPGA image that was built and tested with the current version of the MPI.

See Also

MEIFpgaSqMACVersionDEFAULT

MEIFpgaSqMACVersionDEFAULT

```
#define MEIFpgaSqMACVersionDEFAULT    (0x020E)
```

Description

MEIFpgaMACVersionDEFAULT defines the version of the SqMAC FPGA image that was built and tested with the current version of the MPI. The sqMAC FPGA image is built into the SynqNet node FPGA image. This version applies to all node types.

See Also

MEIFpgaSqNodeVersionDEFAULT | MPI/SynqNet FPGA Compatibility Check

MEIFpgaSqMACVersionMIN

MEIFpgaSqMACVersionMIN

```
#define MEIFpgaSqMACVersionMIN    (0x0207)
```

Description

MEIFpgaMACVersionMIN defines the minimum version of the SqMAC FPGA image that is compatible with the current version of the MPI. The sqMAC FPGA image is built into the SynqNet node FPGA image.

This version applies to all node types.

See Also

MEIFpgaSqNodeVersionDEFAULT | MPI/SynqNet FPGA Compatibility Check

MEIFpgaSqMACVersionMAX

MEIFpgaSqMACVersionMAX

```
#define MEIFpgaSqMACVersionMAX    (0x02FF)
```

Description

MEIFpgaMACVersionMAX defines the maximum version of the SqMAC FPGA image that is compatible with the current version of the MPI. The sqMAC FPGA image is built into the SynqNet node FPGA image.

This version applies to all node types.

See Also

MEIFpgaSqNodeVersionDEFAULT | MPI/SynqNet FPGA Compatibility Check

MEIFpgaSqNodeVersionDEFAULT

MEIFpgaSqNodeVersionDEFAULT

```
#define MEIFpgaSqNodeVersionDEFAULT    (0x0311)
```

Description

MEIFpgaSqNodeVersionDEFAULT defines the version of the SynqNet node FPGA image that was built and tested with the current version of the MPI. This is the recommended version to have loaded on all SynqNet nodes.

This version may not apply to all node types.

See Also

MEIFpgaSqNodeVersionMIN | MEIFpgaSqNodeVersionMAX | MPI/SynqNet FPGA Compatibility Check

MEIFpgaSqNodeVersionMIN

MEIFpgaSqNodeVersionMIN

```
#define MEIFpgaSqNodeVersionMIN    (0x0303)
```

Description

MEIFpgaSqNodeVersionMIN defines the minimum version of the SynqNet node FPGA image that is compatible with the current version of the MPI.

This version may not apply to all node types.

See Also

MEIFpgaSqNodeVersionMAX | MEIFpgaSqNodeVersionDEFAULT | MPI/SynqNet FPGA Compatibility Check

MEIFpgaSqNodeVersionMAX

MEIFpgaSqNodeVersionMAX

```
#define MEIFpgaSqNodeVersionMAX    (0x03FF)
```

Description

MEIFpgaSqNodeVersionMAX defines the maximum version of the SynqNet node FPGA image that is compatible with the current version of the MPI.

This version may not apply to all node types.

See Also

MEIFpgaSqNodeVersionMIN | MEIFpgaSqNodeVersionDEFAULT | MPI/SynqNet FPGA Compatibility Check