# Sequence Objects

## Introduction

A **Sequence** object manages a set of Commands. The sequence is constructed on the host from a list of commands, then downloaded and executed in the controller. Typically, applications only use Sequences for very small or simple autonomous tasks that require execution in the controller. Due to their embedded execution, debugging can be difficult. It is best to use the host application to execute MPI methods directly for optimum flexibility and performance.

If you are considering using a program Sequencer or Command objects, please contact your support engineer. We recommend that you do NOT implement complex Sequences on your own.

Commands are implemented using MPICommand objects. Information about the different types of commands can be found on MPICommandType and MPICommandParams. Sample applications for using sequencers can be found in the Sample Applications section. Search for application names starting with ***seq***. Seqkill.c is a good place to start.

| Error Messages |

## Methods

### Create, Delete, Validate Methods

| | |
|---|---|
| mpiSequence**Create** | Create Sequence object |
| mpiSequence**Delete** | Delete Sequence object |
| mpiSequence**Validate** | Validate Sequence object |

### Configuration and Information Methods

| | |
|---|---|
| mpiSequence**ConfigGet** | Get sequence config |
| mpiSequence**ConfigSet** | Set sequence config |
| mpiSequence**FlashConfigGet** | Get sequence flash config |
| mpiSequence**FlashConfigSet** | Set sequence flash config |
| mpiSequence**PageSize** | Set pageSize to number of command slots used by sequence |
| mpiSequence**Status** | Return sequence status |

### Event Methods

| | |
|---|---|
| mpiSequence**EventNotifyGet** | Select an event mask for host notification of events |
| mpiSequence**EventNotifySet** | Enable host notification of sequence events |
| mpiSequence**EventReset** | Reset sequence events |

### Action Methods

| | |
|---|---|
| meiSequence**Compile** | |

| | |
|---|---|
| mpiSequence**Load** | Load sequence commands into firmware |
| mpiSequence**Resume** | Resume execution of sequence |
| mpiSequence**Start** | Start execution of sequence |
| mpiSequence**Step** | Execute count steps of a stopped sequence |
| mpiSequence**Stop** | Stop sequence |

### Memory Methods

| | |
|---|---|
| mpiSequence**Memory** | Set address used to access sequence memory |
| mpiSequence**MemoryGet** | Get bytes of sequence memory and put into application memory |
| mpiSequence**MemorySet** | Put (set) bytes of application memory into sequence memory |

### Relational Methods

| | |
|---|---|
| mpiSequence**Control** | Get handle to Control |
| mpiSequence**Number** | Get index number of sequence |

### Command Methods

| | |
|---|---|
| mpiSequence**Command** | Return handle to indexed command of sequence |
| mpiSequence**CommandAppend** | Append command to sequence |
| mpiSequence**CommandCount** | Count the number of commands in sequence |
| mpiSequence**CommandFirst** | Return handle to first command in sequence |
| mpiSequence**CommandIndex** | Return the index of a command in sequence |
| mpiSequence**CommandInsert** | Insert command into sequence |
| mpiSequence**CommandLast** | Return handle of last command in sequence |
| mpiSequence**CommandListGet** | Get list of commands in sequence |
| mpiSequence**CommandListSet** | Set list of commands in sequence |
| mpiSequence**CommandNext** | Get handle to next command in list |
| mpiSequence**CommandPrevious** | Get handle to previous command in list |
| mpiSequence**CommandRemove** | Remove command from list |

## Data Types

MPISequence**Config** / MEISequence**Config**

MPISequence**Message**

MPISequence**State**

MPISequence**Status**

MEISequence**Trace**

## See Also

MPICommand

MPICommand**Type**

MPICommand**Params**

seqKill.c (sample application)

# mpiSequenceCreate

## Declaration

```
MPISequence mpiSequenceCreate(MPIControl control,
                              long       number,
                              long       pageSize)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCreate** creates a Sequence object associated with the program sequencer identified by ***number*** located on motion controller (control). SequenceCreate is the equivalent of a C++ constructor.

| If | Then |
|---|---|
| **number** is **-1** | *SequenceCreate* selects the next unused program sequencer. If this is the first use of the program sequencer, then SequenceCreate will attempt to allocate pageSize firmware command slots. |
| **pageSize** is **-1** | *SequenceCreate* will allocate all remaining firmware command slots, which may prevent any more Sequence objects from being created. |

| Return Values | |
|---|---|
| **handle** | to a Sequence object |
| **MPIHandleVOID** | if the object could not be created |

## See Also

mpiSequenceDelete | mpiSequenceValidate

# mpiSequenceDelete

## Declaration

```
long mpiSequenceDelete(MPISequence  sequence)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceDelete** deletes a Sequence object and invalidates its handle (***sequence***). *SequenceDelete* is the equivalent of a C++ destructor.

All Command objects in a Sequence are deleted when the Sequence object is deleted.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |

| **Return Values** | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceCreate | mpiSequenceValidate

# mpiSequenceValidate

## Declaration

```
long mpiSequenceValidate(MPISequence sequence)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceValidate** validates the Sequence object and its handle (***sequence***).

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceCreate | mpiSequenceDelete

# mpiSequenceConfigGet

## Declaration

```
long mpiSequenceConfigGet(MPISequence          sequence,
                          MPISequenceConfig    *config,
                          void                 *external)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceConfigGet** gets the configuration of a Sequence object (***sequence***) and writes it in the structure pointed to by ***config***, and also writes it into the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The Sequence's configuration information in ***external*** is in addition to the Sequence's configuration information in ***config***, i.e, the configuration information in ***config*** and in ***external*** is not the same information. Note that ***config*** or ***external*** can be NULL (but not both NULL).

## Remarks

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

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceConfigSet | MEISequenceConfig

# mpiSequenceConfigSet

## Declaration

```
long mpiSequenceConfigSet(MPISequence          sequence,
                          MPISequenceConfig    *config,
                          void                 *external)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceConfigSet** sets the configuration of a Sequence (***sequence***) using data from the structure pointed to by ***config***, and also using data from the implementation- specific structure pointed to by ***external*** (if ***external*** is not NULL).

The Sequence's configuration information in ***external*** is in addition to the Sequence's configuration information in ***config***, i.e, the configuration information in ***config*** and in ***external*** is not the same information. Note that ***config*** or ***external*** can be NULL (but not both NULL).

## Remarks

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

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceConfigGet | MEISequenceConfig

# mpiSequenceFlashConfigGet

## Declaration

```
long mpiSequenceFlashConfigGet(MPISequence       sequence,
                               void              *flash,
                               MPISequenceConfig *config,
                               void              *external)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceFlashConfigGet** gets a Sequence's (***sequence***) flash configuration and writes it into the structure pointed to by ***config***, and also writes it into the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The Sequence's flash configuration information in ***external*** is in addition to the Sequence's flash configuration information in ***config***, i.e., the flash configuration information in ***config*** and in ***external*** is not the same information. Note that ***config*** or ***external*** can be NULL (but not both NULL). The implementation-specific ***flash*** argument is used to access flash memory.

## Remarks

***external*** either points to a structure of type MEISequenceConfig{} or is NULL. ***flash*** is either an MEIFlash handle or MPIHandleVOID. If ***flash*** is MPIHandleVOID, an MEIFlash object will be created and deleted internally.

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceFlashConfigSet

# mpiSequenceFlashConfigSet

## Declaration

```
long mpiSequenceFlashConfigSet(MPISequence        sequence,
                               void               *flash,
                               MPISequenceConfig  *config,
                               void               *external)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceFlashConfigSet** sets a Sequence's (***sequence***) flash configuration using data from the structure pointed to by ***config***, and also using data from the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The Sequence's flash configuration information in ***external*** is in addition to the Sequence's flash configuration information in ***config***, i.e., the flash configuration information in ***config*** and in ***external*** is not the same information. Note that ***config*** or ***external*** can be NULL (but not both NULL). The implementation-specific ***flash*** argument is used to access flash memory.

## Remarks

***external*** either points to a structure of type MEISequenceConfig{} or is NULL. ***flash*** is either an MEIFlash handle or MPIHandleVOID. If ***flash*** is MPIHandleVOID, an MEIFlash object will be created and deleted internally.

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

MEISequenceConfig | mpiSequenceFlashConfigGet

# mpiSequencePageSize

## Declaration

```
long mpiSequencePageSize(MPISequence sequence,
                         long        *pageSize)
```

**Required Header:** stdmpi.h

## Description

**mpiSequencePageSize** writes the *number* of command slots that are available to a Sequence (***sequence***, on its associated motion controller) to the contents of ***pageSize***.

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

# mpiSequenceStatus

## Declaration

```
long mpiSequenceStatus(MPISequence       sequence,
                       MPISequenceStatus *status,
                       void              *external)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceStatus** returns the status of a Sequence (***sequence***), and writes it into the structure pointed to by ***status***, and also writes it into the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

## Remarks

***external*** should always be set to NULL.

| | |
|---|---|
| **sequence** | a handle to a Sequence object |
| ***status** | a pointer to Sequence's status structure |
| ***external** | a pointer to an implementation-specific structure |

| Return Values | |
|---|---|
| MPIMessageOK | |
| MPIMessageARG_INVALID | |

## See Also

MPISequenceStatus

# mpiSequenceEventNotifyGet

## Declaration

```
long mpiSequenceEventNotifyGet(MPISequence   sequence,
                               MPIEventMask  *eventMask,
                               void          *external)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceEventNotifyGet** writes an event mask [that specifies the event types (generated by the Sequence ***sequence***, for which host notification has been requested] to the structure pointed to by ***eventMask***, and also writes it into the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The event mask information in ***external*** is *in addition* to the event mask information in ***eventMask***, i.e, the event mask information in ***eventMask*** and in ***external*** is not the same information. Note that ***eventMask*** or ***external*** can be NULL (but not both NULL).

## Remarks

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

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

MEIEventMask | mpiSequenceEventNotifySet

# mpiSequenceEventNotifySet

## Declaration

```
long mpiSequenceEventNotifySet(MPISequence   sequence,
                               MPIEventMask  eventMask,
                               void          *external)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceEventNotifySet** requests host notification of the event(s) specified by ***eventMask*** and generated by a Sequence (***sequence***), and also using data from the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The event mask information in ***external*** is in addition to the event mask information in ***eventMask***, i.e, the event mask information in ***eventMask*** and in ***external*** is not the same information. Note that ***eventMask*** or external can be NULL (but not both NULL).

The mask of event types generated by a Sequence object consists of MPIEventMaskEXTERNAL. When a Sequence issues a Command of type MPICommandTypeEVENT, an event of type MPIEventTypeEXTERNAL is generated. The only event generated by a Sequence is MPIEventTypeEXTERNAL, which is generated when a Sequence issues a Command of type MPICommandTypeEVENT.

## Remarks

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

| To | Use "eventMask" |
|---|---|
| Disable host notification of all Sequence events | MPIEventTypeNONE |
| Enable host notification of all Sequence events | MPIEventMaskALL |

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

MPIEventMaskEXTERNAL | MEIEventMask | mpiSequenceEventNotifyGet

# mpiSequenceEventReset

## Declaration

```
long mpiSequenceEventReset(MPISequence  sequence,
                           MPIEventMask eventMask)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceEventReset** resets the event(s) that are specified in ***eventMask*** and generated by a Sequence (***sequence***). Your application should not call SequenceEventReset *until* one or more latchable events have occurred.

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiControlEventReset | mpiMotionEventReset | mpiMotorEventReset | mpiRecorderEventReset | mpiSequenceEventReset | meiSynqNetEventReset | meiSqNodeEventReset | mpiAxisEventReset

Event Notification Methods

# meiSequenceCompile

## Declaration

```
long meiSequenceCompile(MPISequence sequence)
```

**Required Header:** stdmei.h

## Description

**meiSequenceCompile** "compiles" a ***sequence*** object by reading its list of Command objects and then creating an equivalent list of XMP commands.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |

| **Return Values** | |
|---|---|
| MPIMessageOK | |

## Compiling Program Sequencer Commands

An MPICommand will "compile" into one or more MEIXmpCommand{}s, each of which takes up a slot in the MEIXmpCommandBuffer{}. In general, an MPICommand will compile into a single MEIXmpCommand{}, but an MPICommand of type MPICommandTypeMOTION [with a motionCommand of MPICommandMotionSTART (i.e. mpiMotionStart(...))] will require several MEIXmpCommand{}s.

How many sequencer commands an MPI sequence command compiles to depends on the number of axes and number of positions in the move. The next table shows how many xmp sequencer commands it takes to do the equivalent of an mpiMotionStart(...).

**Number of Sequencer Commands to be equivalent to mpiMotionStart(...)**

| Number of required sequencer commands | To do this: |
|---|---|
| axisCount + | One MEIXmpCommand{} per axis to write the axis number to MEIXmpLinkBuffer{}.MSLink[].Axis[].AxisNumber |
| 1+ | 1 + One MEIXmpCommand{} to write axisCount to MEIXmpLinkBuffer{}.MSLink[].Axes |
| 1+ | One MEIXmpCommand{} to write the MEIXmpMotionType{} to MS[].Mode. |
| (((axisCount*pointCount) + 3) / 4) + | One MEIXmpCommand{} for every four MEIXmpPoint{}s written to PointBuffer.Point[] |

| | |
|---|---|
| axisCount + | One MEIXmpCommand{} per axis to load the MEIXmpPoint (s) |
| 1 | One MEIXmpCommand{} to start the motion |

## See Also

# mpiSequenceLoad

## Declaration

```
long mpiSequenceLoad(MPISequence sequence,
                     MPICommand  command,
                     MPI_BOOL    start);
```

**Required Header:** stdmpi.h
**Change History:** Modified in the 03.03.00

## Description

**mpiSequenceLoad** loads the firmware command slots of a Sequence (***sequence***) as necessary, starting with the Command (***command***).

*SequenceLoad* is intended to be called initially by mpiSequenceStart(...) and called thereafter by mpiEventMgrService(...) (in response to reception of an *internal page fault event notification* from the firmware). Except when you are debugging a sequence via mpiSequenceStep(...), your application should never need to directly call SequenceLoad.

| If | Then |
|---|---|
| ***command*** is MPIHandleVOID | *SequenceLoad* loads Commands starting with the first Command of the Sequence |
| ***start*** is not FALSE | *SequenceLoad* starts the sequence after the commands are loaded |

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceStart | mpiEventMgrService | mpiSequenceStep

# mpiSequenceResume

## Declaration

```
long mpiSequenceResume(MPISequence sequence)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceResume** resumes a Sequence (***sequence***) from the point where the Sequence has stopped (if execution has been stopped).

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

# mpiSequenceStart

## Declaration

```
long mpiSequenceStart(MPISequence sequence,
                      MPICommand  command)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceStart** begins the execution of a Sequence (***sequence***), starting with the Command (***command***). If ***command*** is MPIHandleVOID, execution starts with the first command of the Sequence.

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceStop

# mpiSequenceStep

## Declaration

```
long mpiSequenceStep(MPISequence sequence,
                     long        count)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceStep** executes ***count*** steps (Commands) of a stopped Sequence (***sequence***). After executing the Commands, the Sequence will be in the MPISequenceStateSTOPPED state.

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

# mpiSequenceStop

## Declaration

```
long mpiSequenceStop(MPISequence sequence)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceStop** stops a Sequence (***sequence***), if execution has been started. A stopped Sequence can be resumed from the point where it has stopped.

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceStart

# mpiSequenceMemory

## Declaration

```
long mpiSequenceMemory(MPISequence sequence,
                       void        **memory)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceMemory** writes an address [used to access a Sequence's (sequence) memory] to the contents of ***memory***. This address (or an address calculated from it) is passed as the ***src*** argument to mpiSequenceMemoryGet(...) and as the ***dst*** argument to mpiSequenceMemorySet(...).

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceMemoryGet | mpiSequenceMemorySet

# mpiSequenceMemoryGet

## Declaration

```
long mpiSequenceMemoryGet(MPISequence sequence,
                          void        *dst,
                          const void  *src,
                          long        count);
```

**Required Header:** stdmpi.h
**Change History:** Modified in the 03.03.00

## Description

**mpiSequenceMemoryGet** copies ***count*** bytes of a Sequence's (***sequence***) memory (starting at address ***src***) to application memory (starting at address ***dst***).

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceMemorySet | mpiSequenceMemory

# mpiSequenceMemorySet

## Declaration

```
long mpiSequenceMemorySet(MPISequence sequence,
                          void        *dst,
                          const void  *src,
                          long        count);
```

**Required Header:** stdmpi.h
**Change History:** Modified in the 03.03.00

## Description

**mpiSequenceMemorySet** copies ***count*** bytes of application memory (starting at address ***src***) to a Sequence's (***sequence***) memory (starting at address ***dst***).

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceMemory | mpiSequenceMemoryGet

# mpiSequenceControl

## Declaration

```
MPIControl mpiSequenceControl(MPISequence sequence)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceControl** returns a handle to the Control object with which the Sequence object is associated.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |

| **Return Values** | |
|---|---|
| **MPIControl** | a handle to the Sequence object |
| **MPIHandleVOID** | if ***sequence*** is invalid |

## See Also

mpiSequenceCreate | mpiControlCreate

# mpiSequenceNumber

## Declaration

```
long mpiSequenceNumber(MPISequence sequence,
                       long        *number)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceNumber** writes the index of a Sequence (***sequence***, on the motion controller that the Sequence object is associated with) to the contents of ***number***.

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

# mpiSequenceCommand

## Declaration

```
MPICommand mpiSequenceCommand(MPISequence sequence,
                              long        index)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommand** returns the element at the position on the list indicated by ***index***.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |
| **index** | a position in the list. |

| **Return Values** | |
|---|---|
| **handle** | to the ***index***th Command of a Sequence (***sequence***) |
| **MPIHandleVOID** | if ***sequence*** is invalid<br>if ***index*** is less than 0<br>if ***index*** is greater than or equal to **mpiSequenceCount(sequence)** |
| MPIMessageARG_INVALID | if ***index*** is a negative number. |
| MEIListMessageELEMENT_NOT_FOUND | if ***index*** is greater than or equal to the number of elements in the list. |
| MPIMessageHANDLE_INVALID | if ***sequence*** is an invalid handle. |

## See Also

# mpiSequenceCommandAppend

## Declaration

```
long mpiSequenceCommandAppend(MPISequence sequence,
                              MPICommand  command)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandAppend** appends a Command (***command***) to a Sequence (***sequence***).

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |
| **command** | a handle to a Command object. |

| **Return Values** | |
|---|---|
| MPIMessageOK | |
| MPIMessageHANDLE_INVALID | |
| MPIMessageNO_MEMORY | |

## See Also

# mpiSequenceCommandCount

## Declaration

```
long mpiSequenceCommandCount(MPISequence sequence)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandCount** returns the number of elements on the list.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |

| Return Values | |
|---|---|
| **number of Commands** | in a Sequence (***sequence***) |
| **-1** | if ***sequence*** is invalid |
| **0** | if ***sequence*** is empty |

## See Also

# mpiSequenceCommandFirst

## Declaration

```
MPICommand mpiSequenceCommandFirst(MPISequence  sequence)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandFirst** returns the first element in the list. This function can be used in conjuntion with mpiSequenceCommandNext() in order to iterate through the list.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |

| **Return Values** | |
|---|---|
| **handle** | to the first Command in a Sequence (***sequence***) |
| **MPIHandleVOID** | if ***sequence*** is invalid<br>if ***sequence*** is empty |
| MPIMessageHANDLE_INVALID | |

## See Also

mpiSequenceCommandNext | mpiSequenceCommandLast

# mpiSequenceCommandIndex

## Declaration

```
long mpiSequenceCommandIndex(MPISequence  sequence,
                             MPICommand   command)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandIndex** returns the position of "command" on the list.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |
| **command** | a handle to a Command object. |

| **Return Values** | |
|---|---|
| **index** | of a Command (***command***) in a Sequence (***sequence***) |
| **-1** | if ***sequence*** is invalid<br>if the Command (***command***) was not found in the Sequence (***sequence***) |

## See Also

# mpiSequenceCommandInsert

## Declaration

```
long mpiSequenceCommandInsert(MPISequence sequence,
                              MPICommand  command,
                              MPICommand  insert)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandInsert** inserts a Command (***insert***) in a Sequence (***sequence***) just after the specified Command (***command***).

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceCommandNext | mpiSequenceCommandLast

# mpiSequenceCommandLast

## Declaration

```
MPICommand mpiSequenceCommandLast(MPISequence sequence)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandLast** returns the last element in the list. This function can be used in conjuntion with mpiSequenceCommandPrevious(...) in order to iterate through the list backwards.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |

| **Return Values** | |
|---|---|
| MPIMessageOK | |

| **Return Values** | |
|---|---|
| **handle** | to the last Command in a Sequence (***sequence***) |
| **MPIHandleVOID** | if ***sequence*** is invalid<br>if ***sequence*** is empty |
| MPIMessageHANDLE_INVALID | |

## See Also

mpiSequenceCommandFirst | mpiSequenceCommandPrevious | mpiSequenceCommandNext

# mpiSequenceCommandListGet

## Declaration

```
long mpiSequenceCommandListGet(MPISequence  sequence,
                               long         *commandCount,
                               MPICommand   *commandList)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandListGet** gets the Commands in a Sequence (***sequence***). *SequenceCommandListGet* writes the number of Commands [in a Sequence (***sequence***)] to the location (pointed to by ***commandCount***), and also writes an array (of ***commandCount*** Command handles) to the location (pointed to by ***commandList***).

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceCommandListSet

# mpiSequenceCommandListSet

## Declaration

```
long mpiSequenceCommandListSet(MPISequence  sequence,
                               long         commandCount,
                               MPICommand   *commandList)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandListSet** creates a Sequence (***sequence***) of ***commandCount*** Commands using the Command handles specified by ***commandList***. Any existing command Sequence is completely replaced.

The ***commandList*** parameter is the address of an array of ***commandCount*** Command handles, or is NULL (if ***commandCount*** is equal to zero).

You can also create a command Sequence incrementally (i.e., one command at a time), by using the Append and/or Insert methods. Use the List methods to examine and manipulate a command Sequence, regardless of how it was created.

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

mpiSequenceCommandListGet

# mpiSequenceCommandNext

## Declaration

```
MPICommand mpiSequenceCommandNext(MPISequence sequence,
                                  MPICommand  command)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandNext** returns the next element following "command" on the list. This function can be used in conjuntion with mpiSequenceCommandFirst(...) in order to iterate through the list.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |
| **command** | a handle to a Command object. |

| **Return Values** | |
|---|---|
| **handle** | to the Command following the Command (***command***) in a Sequence (***sequence***) |
| **MPIHandleVOID** | if ***sequence*** is invalid<br>if ***command*** is the last command in a Sequence (***sequence***) |
| MPIMessageHANDLE_INVALID | |

## See Also

mpiSequenceCommandFirst | mpiSequenceCommandPrevious

# mpiSequenceCommandPrevious

## Declaration

```
MPICommand mpiSequenceCommandPrevious(MPISequence sequence,
                                      MPICommand  command)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandPrevious** returns the previous element prior to "command" on the list. This function can be used in conjuntion with mpiSequenceCommandLast(...) in order to iterate through the list backwards.

| | |
|---|---|
| **sequence** | a handle to the Sequence object. |
| **command** | a handle to a Command object. |

| **Return Values** | |
|---|---|
| **handle** | to the Command preceding the Command (***command***) in a Sequence (***sequence***) |
| **MPIHandleVOID** | if ***sequence*** is invalid<br>if ***command*** is the first command in a Sequence (***sequence***) |
| MPIMessageHANDLE_INVALID | |

## See Also

mpiSequenceCommandLast | mpiSequenceCommandNext

# mpiSequenceCommandRemove

## Declaration

```
long mpiSequenceCommandRemove(MPISequence sequence,
                              MPICommand  command)
```

**Required Header:** stdmpi.h

## Description

**mpiSequenceCommandRemove** removes a Command (***command***) from a Sequence (***sequence***).

| Return Values | |
|---|---|
| MPIMessageOK | |

## See Also

# MPISequenceConfig / MEISequenceConfig

## Definition: MPISequenceConfig

```
typedef MPIEmpty    MPISequenceConfig;
```

## Description

**MPISequenceConfig** is currently not supported and is reserved for future use.

## Definition: MEISequenceConfig

```
typedef MPIEmpty    MEISequenceConfig;
```

## Description

**MEISequenceConfig** is currently not supported and is reserved for future use.

## See Also

mpiSequenceConfigGet | mpiSequenceConfigSet

# MPISequenceMessage

## Definition

```
typedef enum {

    MPISequenceMessageSEQUENCE_INVALID,
    MPISequenceMessageCOMMAND_COUNT,
    MPISequenceMessageCOMMAND_NOT_FOUND,
    MPISequenceMessageSTARTED,
    MPISequenceMessageSTOPPED,
} MPISequenceMessage;
```

## Description

**MPISequenceMessage** is an enumeration of Sequence error messages that can be returned by the MPI library.

### MPISequenceMessageSEQUENCE_INVALID

The sequence number is out of range. This message code is returned by mpiSequenceCreate(...) if the sequence number is less than zero or greater than or equal to MEIXmpMAX_PSs. This message code is also returned if the specified sequence number is not active in the controller. To correct this problem, use mpiControlConfigSet(...) to enable the sequence object, by setting the sequenceCount to greater than the sequence number. For example, to enable sequence 0 to 3, set sequenceCount to 4. This message code is returned by mpiSequenceLoad(...) if the sequence buffer size and the sequence page size are not equal. This indicates an internal MPI Library problem.

### MPISequenceMessageCOMMAND_COUNT

The sequence command count is out of range. This message code is returned by mpiSequenceStart(...) or meiSequenceCompile(...) if the sequence command count is less than or equal to zero. To correct this problem, set the command count to a value greater than zero.

### MPISequenceMessageCOMMAND_NOT_FOUND

The sequence command is not found. This message code is returned by mpiSequenceLoad(...), mpiSequenceStart(...), or meiSequenceCompile(...) if the specified command is not a member of the sequence. To correct this problem, specify a command that is a member of the sequence.

### MPISequenceMessageSTARTED

The program sequencer is already running. This message code is returned by mpiSequenceResume(...), mpiSequenceStart(...), or mpiSequenceStep(...) if the program sequencer has already been started. If this is a problem, call mpiSequenceStop(...) to stop the program sequencer or monitor the sequence status and wait for the state to equal STOPPED.

### MPISequenceMessageSTOPPED

The program sequencer is not running. This message code is returned by mpiSequenceStop(...) if the program sequencer has already been stopped. If this is a problem, call mpiSequenceStart(...) to start the program sequencer.

## See Also

# MPISequenceState

## Definition

```
typedef enum {
    MPISequenceStateSTOPPED = 0,
    MPISequenceStateSTARTED,
} MPISequenceState;
```

## Description

**MPISequenceState** is an enumeration of fan status bit for use in the MPIControlFanStatusMask. The status bits represent the present status condition(s) for the fan controller on a given Control object.

| | |
|---|---|
| **MPISequenceStateSTOPPED** | Means that the XMP's on-board program sequencer state is stopped. The program sequencer is in this state after it is created, and is not running. If the program sequencer has already been started, then a call to the MPI method mpiSequenceStop will stop the sequencer, and the sequencer state will be MPISequenceStateSTOPPED. |
| **MPISequenceStateSTARTED** | Means that the XMP's on-board program sequencer state is running. The program sequencer is in this state after it has been created, and successfully started with a call to the MPI method mpiSequenceStart. |

## See Also

# MPISequenceStatus

## Definition

```
typedef struct MPISequenceStatus {
    MPICommand          command;
    MPISequenceState    state;
} MPISequenceStatus;
```

## Description

**MPISequenceStatus** is a status structure for MPISequence objects.

| | |
|---|---|
| **command** | The current command of the MPISequence object |
| **state** | The current state of the MPISequence object |

## See Also

MPISequence | mpiSequenceStatus

# MEISequenceTrace

## Definition

```
typedef enum {
    MEISequenceTraceLOAD,
} MEISequenceTrace;
```

## Description

**MEISequenceTrace** sets tracing on for the mpiSequenceLoad(...) method.

## See Also

MPISequence | MEITrace | mpiSequenceLoad