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.

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

List Methods for Event Sources

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

SequenceCreate 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 **SequenceDelete** 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.

Return Values

MPIMessageOK	if *SequenceDelete* successfully a Sequence object and invalidates its handle

See Also mpiSequenceCreate | mpiSequenceValidate

mpiSequenceValidate

Declaration

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

Required Header stdmpi.h

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

Return Values	
MPIMessageOK	if Sequence is a handle to a valid object.

See Also mpiSequenceCreate | mpiSequenceDelete

mpiSequenceConfigGet

Declaration

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

Required Header

stdmpi.h

Description

SequenceConfigGet 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).

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

Return Values

MPIMessageOK	if *SequenceConfigGet* successfully gets and writes the configuration of a Sequence object into the structure(s)

See Also

mpiSequenceConfigSet | MEISequenceConfig

mpiSequenceConfigSet

Declaration

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

Required Header

stdmpi.h

Description

SequenceConfigSet 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).

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

Return Values

MPIMessageOK	if *SequenceConfigSet* successfully sets a Sequence's configuration using data from the structure(s)

See Also

mpiSequenceConfigGet | MEISequenceConfig

mpiSequenceFlashConfigGet

Declaration

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

Required Header

stdmpi.h

Description

SequenceFlashConfigGet 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.

XMP Only	***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	if *SequenceFlashConfigGet* successfully writes the Sequence's flash configuration to the structure(s)

See Also

mpiSequenceFlashConfigSet

mpiSequenceFlashConfigSet

Declaration

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

Required Header

stdmpi.h

Description

SequenceFlashConfigSet 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.

XMP Only	***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	if *SequenceFlashConfigSet* successfully sets the Sequence's flash configuration using data from the structure(s)

See Also

MEISequenceConfig | mpiSequenceFlashConfigGet

mpiSequencePageSize

Declaration

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

Required Header

stdmpi.h

Description

SequencePageSize 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	if *SequencePageSize* successfully writes the number of command slots (available to the Sequence) to the contents of ***pageSize***

See Also

mpiSequenceStatus

Declaration

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

Required Header stdmpi.h

Description

SequenceStatus 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).

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

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

Return Values

MPIMessageOK	if *SequenceStatus* successfully returns the Sequence's status and writes the status to the structure(s)
MPIMessageARG_INVALID	if the *status* pointer is NULL.

See Also MPISequenceStatus

mpiSequenceEventNotifyGet

Declaration

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

Required Header

stdmpi.h

Description

SequenceEventNotifyGet 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).

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

Return Values

MPIMessageOK	if *SequenceEventNotifyGet* successfully writes the event mask to the structure(s)

See Also

MEIEventMask | mpiSequenceEventNotifySet

mpiSequenceEventNotifySet

Declaration

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

Required Header

stdmpi.h

Description

SequenceEventNotifySet 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.

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

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

Return Values

MPIMessageOK	if *SequenceEventNotifySet* successfully requests host notification of the events in the event mask(s)

See Also

MPIEventMaskEXTERNAL | MEIEventMask | mpiSequenceEventNotifyGet

mpiSequenceEventReset

Declaration

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

Required Header

stdmpi.h

Description

SequenceEventReset 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	if *SequenceEventReset* successfully resets the event(s) that are specified in ***eventMask*** and generated by a Sequence object

See Also

meiSequenceCompile

Declaration	`long meiSequenceCompile(MPISequence sequence)`
Required Header	stdmpi.h
Description	**SequenceCompile** "compiles" a ***sequence*** object by reading its list of Command objects and then creating an equivalent list of XMP commands.

Return Values

MPIMessageOK	if *SequenceCompile* successfully reads a Sequence object's list of Command objects and creates an equivalent list of XMP commands

See Also

mpiSequenceLoad

Declaration

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

Required Header

stdmpi.h

Description

SequenceLoad 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	if *SequenceLoad* successfully loads the firmware command slots of a Sequence

See Also

mpiSequenceStart | mpiEventMgrService | mpiSequenceStep

mpiSequenceResume

Declaration

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

Required Header

stdmpi.h

Description

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

Return Values

MPIMessageOK	if *SequenceResume* successfully resumes a Sequence from the point where the Sequence has stopped

See Also

mpiSequenceStart

Declaration

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

Required Header

stdmpi.h

Description

SequenceStart 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	if *SequenceStart* successfully begins the execution of a Command Sequence

See Also

mpiSequenceStop

mpiSequenceStep

Declaration

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

Required Header

stdmpi.h

Description

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

Return Values

MPIMessageOK	if *SequenceStep* successfully executes ***count*** steps (Commands) of a stopped Sequence

See Also

mpiSequenceStop

Declaration `long mpiSequenceStop(MPISequence sequence)`

Required Header stdmpi.h

Description **SequenceStop** 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	if *SequenceStop* successfully stops a Sequence (while it is executing)

See Also mpiSequenceStart

mpiSequenceMemory

Declaration

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

Required Header

stdmpi.h

Description

SequenceMemory 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	if *SequenceMemory* successfully writes the address (used to access Sequence memory) to the contents of memory

See Also

mpiSequenceMemoryGet | mpiSequenceMemorySet

mpiSequenceMemoryGet

Declaration

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

Required Header

stdmpi.h

Description

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

Return Values

MPIMessageOK	if *SequenceMemoryGet* successfully copies count bytes of Sequence memory to application memory

See Also

mpiSequenceMemorySet | mpiSequenceMemory

mpiSequenceMemorySet

Declaration

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

Required Header

stdmpi.h

Description

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

Return Values

MPIMessageOK	if *SequenceMemorySet* successfully copies ***count*** bytes of application memory to a Sequence object's memory

See Also

mpiSequenceMemory | mpiSequenceMemoryGet

mpiSequenceControl

Declaration

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

Required Header stdmpi.h

Description **SequenceControl** 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

SequenceNumber 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	if *SequenceNumber* successfully writes the Sequence's index to the contents of ***number***

See Also

mpiSequenceCommand

Declaration

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

Required Header

stdmpi.h

Description

SequenceCommand 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
	if ***index*** is less than 0
	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

SequenceCommandAppend 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	if *SequenceCommandAppend* successfully appends a Command to a Sequence
MPIMessageHANDLE_INVALID	Either ***sequence*** or ***command*** is an invalid handle.
MPIMessageNO_MEMORY	Not enough memory was available.

See Also

mpiSequenceCommandCount

Declaration

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

Required Header

stdmpi.h

Description

SequenceCommandCount 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

SequenceCommandFirst 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
	if ***sequence*** is empty

MPIMessageHANDLE_INVALID	if ***sequence*** is an invalid handle.

See Also

mpiSequenceCommandNext | mpiSequenceCommandLast

mpiSequenceCommandIndex

Declaration

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

Required Header

stdmpi.h

Description

SequenceCommandIndex 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 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

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

Return Values	
MPIMessageOK	if *SequenceCommandInsert* successfully inserts the Command (***insert***) in a Sequence following the specified Command (***command***)

See Also

mpiSequenceCommandLast

Declaration

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

Required Header

stdmpi.h

Description

SequenceCommandLast 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

handle	to the last Command in a Sequence (***sequence***)

MPIHandleVOID	if ***sequence*** is invalid
	if ***sequence*** is empty

MPIMessageHANDLE_INVALID	if ***sequence*** is an invalid handle.

See Also

mpiSequenceCommandFirst | mpiSequenceCommandPrevious | mpiSequenceCommandNext

mpiSequenceCommandListGet

Declaration

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

Required Header

stdmpi.h

Description

SequenceCommandListGet 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	if *SequenceCommandListGet* successfully gets the list of Commands in a Sequence

See Also

mpiSequenceCommandListSet

mpiSequenceCommandListSet

Declaration

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

Required Header

stdmpi.h

Description

SequenceCommandListSet 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	if *SequenceCommandListGet* successfully creates a Sequence of Commands using the Command handles specified by ***commandList***

See Also

mpiSequenceCommandListGet

mpiSequenceCommandNext

Declaration

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

Required Header

stdmpi.h

Description

SequenceCommandNext 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 if ***command*** is the last command in a Sequence (***sequence***)
MPIMessageHANDLE_INVALID	Either ***sequence*** or ***command*** is an invalid handle.

See Also

mpiSequenceCommandFirst | mpiSequenceCommandPrevious

mpiSequenceCommandPrevious

Declaration

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

Required Header

stdmpi.h

Description

SequenceCommandPrevious 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 if ***command*** is the first command in a Sequence (***sequence***)
MPIMessageHANDLE_INVALID	Either ***sequence*** or ***command*** is an invalid handle.

See Also

mpiSequenceCommandLast | mpiSequenceCommandNext

mpiSequenceCommandRemove

Declaration

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

Required Header

stdmpi.h

Description

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

Return Values	
MPIMessageOK	if *SequenceCommandRemove* successfully removes the Command from a Sequence

See Also

MPISequenceConfig / MEISequenceConfig

MPISequenceConfig

```
typedef MPIEmpty    MPISequenceConfig;
```

Description **SequenceConfig** is currently not supported and is reserved for future use.

MEISequenceConfig

```
typedef MPIEmpty    MEISequenceConfig;
```

Description **SequenceConfig** is currently not supported and is reserved for future use.

See Also mpiSequenceConfigGet | mpiSequenceConfigSet

MPISequenceMessage

MPISequenceMessage

```
typedef enum {

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

Description

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

MPISequenceState

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

Description

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

MPISequenceStatus

```
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

MEISequenceTrace

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

Description

MPISequenceTrace sets tracing on for the mpiSequenceLoad() method.

See Also

MPISequence | MEITrace | mpiSequenceLoad