Notify Objects

Introduction

A thread uses a **Notify** object to wait for event notification. For each thread intended to wait for events from an object (or objects), your application must create a Notify object. The source of firmware events are Motion, Sequence, and Recorder objects.

When it is desired to wait for event notifications from a single source, that source (i.e., object handle) can be passed as the second argument to mpiNotifyCreate(...). After a Notify object is appended to the EventMgr list of Notify objects, make a call to mpiNotifyEventWait(...) to instruct the Notify object to wait for event notification.

Implementation

Notify objects maintain a FIFO ("First In, First Out") buffer of events that have occurred. Each call to mpiNotifyEventWait(...) removes one event from the buffer. If the event buffer is empty, mpiNotifyEventWait(...) will wait for an event to be sent to the **Notify** object. This ensures that events will not be missed in cases where multiple events have occurred between calls to mpiNotifyEventWait(...). However, the danger is that an application may not call mpiNotifyEventWait(...) for a long time. In this case, the event buffer may grow rather quickly and use a large amount of system memory. In order to prevent this problem from occurring, one should use mpiNotifyEventMaskSet(...) to enable and disable event notification at the proper times.

Methods

Create, Delete, Validate Methods

mpiNotify**Create**	Create a Notify object
mpiNotify**Delete**	Delete a Notify object
mpiNotify**Validate**	Validate a Notify object

Event Methods

mpiNotify**Event**	Check to see if events have occurred
mpiNotify**EventFlush**	Flush pending events from event queue
mpiNotify**EventMaskGet**	Get event mask
mpiNotify**EventMaskSet**	Set event mask
mpiNotify**EventWait**	Get next event from queue or wait timeout msec for it to arrive
mpiNotify**EventWake**	Wake up thread waiting for notify object

Relational Methods

List Methods for Event Sources

mpiNotify**Source**	Get the indexth event source in list
mpiNotify**SourceAppend**	Append an event source to list
mpiNotify**SourceCount**	Count number of event sources in list
mpiNotify**SourceFirst**	Get first event source in list
mpiNotify**SourceIndex**	Get index value for event source in list
mpiNotify**SourceInsert**	Place event source after source in list
mpiNotify**SourceLast**	Get last event source in list
mpiNotify**SourceListGet**	Get list of event sources
mpiNotify**SourceListSet**	Create a list of event sources
mpiNotify**SourceNext**	Get next event source after source in list
mpiNotify**SourcePrevious**	Get the event source before source in list
mpiNotify**SourceRemove**	Remove event source from list

Data Types

MPINotify**Message**
MEINotify**Trace**

mpiNotifyCreate

Declaration

```
MPINotify mpiNotifyCreate(MPIEventMask mask,
                          void         *source)
```

Required Header

stdmpi.h

Description

NotifyCreate creates a Notify object that will accept event notifications for the events that are specified in ***mask***. The ***source*** argument specifies the initial element in the list of event sources, from which event notification will be accepted. If ***source*** is NULL, then event notification will be accepted from all event sources. *NotifyCreate* is the equivalent of a C++ constructor.

Return Values

handle	to a Notify object
MPIHandleVOID	if the object could not be created

See Also

mpiNotifyDelete | mpiNotifyValidate

mpiNotifyDelete

Declaration	`long mpiNotifyDelete(MPINotify notify)`
Required Header	stdmpi.h
Description	**NotifyDelete** deletes a Notify object and invalidates its handle (***notify***). *NotifyDelete* is the equivalent of a C++ destructor.

Return Values

MPIMessageOK	if *NotifyDelete* successfully deletes a Notify object and invalidates its handle

See Also mpiNotifyCreate | mpiNotifyValidate

mpiNotifyValidate

Declaration	`long mpiNotifyValidate(MPINotify notify)`
Required Header	stdmpi.h
Description	**NotifyValidate** validates a Notify object and its handle (***notify***).

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

See Also mpiNotifyCreate | mpiNotifyDelete

mpiNotifyEvent

Declaration

```
long mpiNotifyEvent(MPINotify       notify,
                    MPIEventStatus *status)
```

Required Header

stdmpi.h

Description

NotifyEvent first checks to see if the type field of ***status*** matches a bit in the event mask maintained by a Notify object (***notify***).

IF
the type field of ***status*** matches a bit in the event mask,

AND
the event source list maintained by the Notify object (***notify***) is empty or the ***source*** field of ***status*** matches a source in the event source list,

THEN
a Notify object (notify) will place the event in a FIFO event queue and signal that an event has been received.

If a thread is waiting for event notification (after having called mpiNotifyEventWait(notify)), the signal will awaken it. Otherwise, the next call to mpiNotifyEventWait(notify) will return immediately with ***status***.

Return Values

MPIMessageOK	if the Notify object (***notify***) successfully places the event in a FIFO event queue and signals that an event has been received

See Also

mpiNotifyEventWait

mpiNotifyEventFlush

Declaration	`long mpiNotifyEventFlush(MPINotify notify)`
Required Header	stdmpi.h
Description	**NotifyEventFlush** flushes any pending events from the internal FIFO event queue maintained by a Notify object (***notify***).

Return Values

MPIMessageOK	if *NotifyEventFlush* successfully flushes the pending events from the internal FIFO event queue maintained by the Notify object

See Also

mpiNotifyEventMaskGet

Declaration

```
long mpiNotifyEventMaskGet(MPINotify    notify,
                           MPIEventMask *mask)
```

Required Header stdmpi.h

Description

NotifyEventMaskGet writes an event mask (that specifies the event type(s) for which event notification is accepted by a Notify object (***notify***)) to the location pointed to by ***mask***.

Sample Code

```
/*
  Disables event notification and copies the previously used
  event mask to oldMask.  oldMask may then be used to re-enable
  event notification via another call to mpiNotifyEventMaskSet().
*/
void NotifyDisable(MPINotify notify, MPIEventMask* oldMask)
{
    MPIEventMask newMask;
    long returnValue;

    returnValue = mpiNotifyEventMaskGet(notify, oldMask);
    msgCheck(returnValue);

    mpiEventMaskCLEAR(newMask);

    returnValue = mpiNotifyEventMaskSet(notify, newMask);
    msgCheck(returnValue);
}
```

Return Values

MPIMessageOK	if *NotifyEventMaskGet* successfully writes the event mask to the location pointed to by mask

See Also mpiNotifyEventMaskSet

mpiNotifyEventMaskSet

Declaration

```
long mpiNotifyEventMaskSet(MPINotify    notify,
                           MPIEventMask mask)
```

Required Header

stdmpi.h

Description

NotifyEventMaskSet sets the event type(s) for which notification will be accepted by a Notify object (***notify***), as specified by ***mask***.

Event notification is accepted for event types specified in ***mask***, a bit mask of MPIEventMask bits (associated with the desired MPIEventType values). The MPIEventMask bits must be set or cleared using the MPIEventMask macros. Event notification is denied for event types not specified in ***mask***.

Sample Code

```
/*
  Disables event notification and copies the previously used
  event mask to oldMask.  oldMask may then be used to re-enable
  event notification via another call to mpiNotifyEventMaskSet().
*/
void NotifyDisable(MPINotify notify, MPIEventMask* oldMask)
{
    MPIEventMask newMask;
    long returnValue;

    returnValue = mpiNotifyEventMaskGet(notify, oldMask);
    msgCheck(returnValue);

    mpiEventMaskCLEAR(newMask);

    returnValue = mpiNotifyEventMaskSet(notify, newMask);
    msgCheck(returnValue);
}
```

Return Values

MPIMessageOK	if *NotifyEventMaskSet* successfully sets the event type(s) for which notification will be accepted by a Notify object

See Also

MPIEventType | mpiNotifyEventMaskGet

mpiNotifyEventWait

Declaration

```
long mpiNotifyEventWait(MPINotify      notify,
                        MPIEventStatus *status,
                        MPIWait        timeout)
```

Required Header

stdmpi.h

Description

NotifyEventWait sets the contents of the structure pointed to by status, using the status of the first event in the internal FIFO event queue (maintained by a Notify object (notify)), and then removes the first event from the queue. If no event is available in the internal FIFO event queue, NotifyEventWait will wait for timeout milliseconds.

If "timeout" is	***Then***
MPIWaitPOLL (**0**)	*NotifyEventWait* will not wait for an event to arrive
MPIWaitFOREVER (**-1**)	*NotifyEventWait* will wait forever for an event to arrive

Return Values

MPIMessageOK	if *NotifyEventWait* successfully sets the contents of the structure pointed to by status, and then removes the first event from the queue
MPIMessageTIMEOUT	if no event is present and the contents of status are undefined

See Also

mpiNotifyEventWake

mpiNotifyEventWake

Declaration

```
long mpiNotifyEventWake(MPINotify       notify,
                        MPIEventStatus *status)
```

Required Header

stdmpi.h

Description

NotifyEventWake wakes a thread that is waiting for an event notification from a Notify object (***notify***). The awakened thread will return from its call to mpiNotifyEventWait(notify, status, timeout) with the contents of ***status*** set to the contents of status. If status is NULL, ***status*** will indicate an event of type MPIEventTypeNONE.

NotifyEventWake is different from *NotifyEvent*, because event notification is not accepted based on the event type or source (*NotifyEvent*); instead event notification is always accepted (*NotifyEventWake*).

Return Values

MPIMessageOK	if *NotifyEventWake* successfully wakes a thread that is waiting for an event notification from a Notify object

See Also

mpiNotifyEventWait | MPIEventType

mpiNotifySource

Declaration

```
void* mpiNotifySource(MPINotify notify,
                      long      index)
```

Required Header

stdmpi.h

Description

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

notify	a handle to the Notify object.
index	a position in the list.

Return Values

***index*th event source**	in the event source list maintained by a Notify object (***notify***)

NULL	if ***notify*** is invalid
	if ***index*** is less than 0
	if ***index*** is greater than or equal to mpiNotifySourceCount(notify)

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 ***notify*** is an invalid handle.

See Also

mpiNotifySourceAppend

Declaration

```
long mpiNotifySourceAppend(MPINotify notify,
                           void      *source)
```

Required Header

stdmpi.h

Description

NotifySourceAppend appends an event source (***source***) to the list of event sources maintained by a Notify object (***notify***).

notify	a handle to the Notify object.
source	a pointer with an arbitrary (but non-NULL) value.

Return Values

MPIMessageOK	if *NotifySourceAppend* successfully appends source to the list of event ***sources*** maintained by a Notify object
MPIMessageHANDLE_INVALID	Either ***source*** is NULL or ***notify*** is an invalid handle.
MPIMessageNO_MEMORY	Not enough memory was available.

See Also

mpiNotifySourceCount

Declaration

```
long mpiNotifySourceCount(MPINotify notify)
```

Required Header stdmpi.h

Description **NotifySourceCount** returns the number of elements on the list.

notify	a handle to the Notify object.

Return Values

number of event sources	in the event source list maintained by a Notify object (***notify***)
-1	if ***notify*** is invalid
0	if the event source list is empty

See Also

mpiNotifySourceFirst

Declaration

```
void* mpiNotifySourceFirst(MPINotify notify)
```

Required Header

stdmpi.h

Description

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

notify	a handle to the Notify object.

Return Values

first event source	in the event source list maintained by a Notify object (***notify***)
NULL	if ***notify*** is invalid or if the event source list is empty
MPIMessageHANDLE_INVALID	if ***notify*** is an invalid handle.

See Also

mpiNotifySourceNext | mpiNotifySourceLast

mpiNotifySourceIndex

Declaration

```
long mpiNotifySourceIndex(MPINotify notify,
                          void      *source)
```

Required Header

stdmpi.h

Description

NotifySourceIndex returns the position of ***source*** on the list.

notify	a handle to the Notify object.
source	a pointer with an arbitrary (but non-NULL) value.

Return Values

index of *source*	in the event source list maintained by a Notify object (***notify***)
-1	if ***notify*** is invalid if the event source (***source***) was not found in the event source list

See Also

mpiNotifySourceInsert

Declaration

```
long mpiNotifySourceInsert(MPINotify notify,
                           void      *source,
                           void      *insert)
```

Required Header

stdmpi.h

Description

NotifySourceInsert places the event source (pointed to by ***insert***) after a specified event source (***source***), in the list of event sources that are maintained by a Notify object (***notify***).

Return Values

MPIMessageOK	if *NotifySourceInsert* successfully places the event source after the specified event source, in the list of event sources that are maintained by a Notify object

See Also

mpiNotifySourceLast

Declaration

```
void* mpiNotifySourceLast(MPINotify notify)
```

Required Header

stdmpi.h

Description

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

notify	a handle to the Notify object.

Return Values

last event source	in the list maintained by a Notify object (***notify***)
NULL	if ***notify*** is invalid if the event source list is empty

See Also

mpiNotifySourcePrevious | mpiNotifySourceFirst

mpiNotifySourceListGet

Declaration

```
long mpiNotifySourceListGet(MPINotify notify,
                            long      *sourceCount,
                            void      **sourceList)
```

Required Header

stdmpi.h

Description

NotifySourceListGet returns the event source list for a Notify object (***notify***). *NotifySourceListGet* writes the number of event sources in the event source list to the location (pointed to by ***sourceCount***), and also writes an array of ***sourceCount*** event source pointers to the location (pointed to by ***sourceList***).

Return Values	
MPIMessageOK	if *NotifySourceListGet* successfully returns the event source list for a Notify object

See Also

mpiNotifySourceListSet | NotifySourceAppend | NotifySourceInsert | mpiNotifyCreate

mpiNotifySourceListSet

Declaration

```
long mpiNotifySourceListSet(MPINotify notify,
                            long      sourceCount,
                            void      **sourceList)
```

Required Header

stdmpi.h

Description

NotifySourceListSet creates an event source list of length ***sourceCount***, using the source pointers specified by ***sourceList***. The ***sourceList*** argument is the address of an array of ***sourceCount*** event source pointers, or is NULL (if ***sourceCount*** = 0). Any existing event source list is completely replaced after using *NotifySourceListSet*.

You can also create an event source list incrementally (i.e., created one source at a time) by using *NotifySourceAppend/Insert* methods. To specify the first event source of a list, use the ***source*** argument of mpiNotifyCreate(...). Use the *NotifySourceList* methods to examine and manipulate an event source list, regardless of how you created it.

Return Values

MPIMessageOK	if *NotifySourceListSet* successfully creates an *event source* list using the source pointers specified by ***sourceList***

See Also

NotifySourceAppend | NotifySourceInsert | mpiNotifyCreate | mpiNotifySourceListGet

mpiNotifySourceNext

Declaration

```
void * mpiNotifySourceNext(MPINotify  notify,
                           void       *source)
```

Required Header

stdmpi.h

Description

NotifySourceNext returns the next element following "source" on the list. This function can be used in conjuntion with mpiNotifySourceFirst() in order to iterate through the list.

notify	a handle to the Notify object.
source	a pointer with an arbitrary (but non-NULL) value.

Return Values

event source	before the event source (***source***) in the event source list maintained by a Notify object (***notify***)
NULL	if ***notify*** is invalid if the event source (***source***) is the first event source in the event source list
MPIMessageHANDLE_INVALID	Either ***source*** is NULL or ***notify*** is an invalid handle.

See Also

mpiNotifySourcePrevious

mpiNotifySourcePrevious

Declaration

```
void * mpiNotifySourcePrevious(MPINotify notify,
                               void      *source)
```

Required Header

stdmpi.h

Description

NotifySourcePrevious returns the previous element prior to "source" on the list. This function can be used in conjuntion with mpiNotifySourceLast() in order to iterate through the list backwards.

notify	a handle to the Notify object.
source	a pointer with an arbitrary (but non-NULL) value.

Return Values

event source	before the event source (***source***) in the event source list maintained by a Notify object (***notify***)
NULL	if ***notify*** is invalid if the event source (***source***) is the first event source in the event source list
MPIMessageHANDLE_INVALID	Either ***source*** is NULL or ***notify*** is an invalid handle.

See Also

mpiNotifySourceNext

mpiNotifySourceRemove

Declaration

```
long mpiNotifySourceRemove(MPINotify notify,
                           void      *source)
```

Required Header

stdmpi.h

Description

NotifySourceRemove removes an event source (***source***) from the list of event sources maintained by a Notify object (***notify***).

Return Values

MPIMessageOK	if *NotifySourceRemove* successfully removes the event source (***source***) from the list of event sources maintained by a Notify object

See Also

MPINotifyMessage

MPINotifyMessage

```
typedef enum {

    MPINotifyMessageNOTIFY_INVALID,
    MPINotifyMessageWAIT_IN_PROGRESS,
} MPINotifyMessage;
```

Description

MPINotifyMessageNOTIFY_INVALID

The notify object is not valid. This message code is returned by a notify method if the notify object handle is not valid. This problem can be caused by failed mpiNotifyCreate(...). To prevent this problem, check your notify objects after creation by using mpiNotifyValidate(...).

MPINotifyMessageWAIT_IN_PROGRESS

The notify object is waiting for an event. This message code is returned by mpiNotifyEventWait(...) if the notify object is already waiting for an event in another thread. To prevent this problem, make sure a thread does not share notify objects with other threads.

Sample Code

```
MPIControl    control;
MPINotify     notify;
long          returnValue;

...

notify =
        mpiNotifyCreate(control);
returnValue =
        mpiNotifyValidate(notify);
```

See Also

MPINotify | mpiNotifyCreate | mpiNotifyValidate

MEINotifyTrace

MEINotifyTrace

```
typedef enum {
    MEINotifyTraceTHREAD,
} MEINotifyTrace;
```

Description

MEINotifyTraceTHREAD	will display trace information when notify objects are set to wait, are finished waiting, and when they are signaled to wake.

See Also