Event Objects

Introduction

An **Event** object contains information about an asynchronous event. Typically, events are generated by the controller, but in some special cases it is possible to generate events from the host computer.

The Event object is retrieved through the EventMgr, via the Notify object. The Event object contains data about the type of event, its source, and other information. The user Event fields can be configured to collect data at the time when the event occurs in the controller.

Methods

Configuration and Information Methods

mpiEvent**StatusGet**	Get Event status
mpiEvent**StatusSet**	Set Event status

Data Types

MPIEvent**Message**
MEIEvent**NotifyData**
MPIEvent**Status**
MEIEvent**StatusInfo**
MPIEvent**Type** / MEIEvent**Type**

Constants

MPIEvent**StatusINFO_COUNT_MAX**	defines the size of the MPIEventStatus.info[] array.

mpiEventStatusGet

Declaration

```
long mpiEventStatusGet(MPIEvent       event,
                       MPIEventStatus *status)
```

Required Header

event.h

Description

EventStatusGet gets the status of an Event object (***event***) and writes it into the structure pointed to by ***status***. Event status includes the event type, type-specific codes and the event source.

Return Values

MPIMessageOK	if *EventStatusGet* successfully gets the status of an Event object and writes it into the structure

See Also

mpiEventStatusSet | meiEventStatusInfo

mpiEventStatusSet

Declaration

```
long mpiEventStatusSet(MPIEvent        event,
                       MPIEventStatus *status)
```

Required Header

event.h

Description

EventStatusSet sets (writes) the status of ***event*** using data from the structure pointed to by ***status***. Event status includes the event type, type-specific codes and the event source.

Return Values

MPIMessageOK	if *EventStatusSet* successfully sets (writes) the status of event using data from the structure

See Also

mpiEventStatusGet | meiEventStatusInfo

MPIEventMessage

MPIEventMessage

```
typedef enum {
    MPIEventMessageEVENT_INVALID,
} MPIEventMessage;
```

Description

MPIEventMessageEVENT_INVALID	
Meaning	The MPIEvent handle passed to an MPIEvent method is invalid.
Possible Causes	Either the handle was never initialized or the mpiEventCreate method failed.
Recommendations	MPIEvent objects are used internally. **Do not create your own.** If you have not and you are receiving this error message, then please contact an applications engineer at Motion Engineering, Inc.

See Also

MEIEventNotifyData

MPIEventNotifyData

```
typedef struct MEIEventNotifyData {
    void        *address[MEIXmpSignalUserData];
} MEIEventNotifyData;
```

Description The ***address*** of an **EventNotifyData** structure is passed as the third (void *external) argument to mpi'Object'EventNotify[GS]et(). The address array contains host-based XMP addresses, the contents of which are returned in MEIEventStatusInfo{}.data.

See Also MEIEventStatusInfo

MPIEventStatus

MPIEventStatus

```
typedef struct MPIEventStatus {
    MPIEventType        type;
    void                *source;

    long                info[MPIEventStatusINFO_COUNT_MAX];
} MPIEventStatus;
```

Description

EventStatus holds information about a particular event that was generated by the XMP.

type	identifies the type of event that was generated.
***source**	identifies what the source of the event was. source will either be a handle to an MPI object or a host pointer. Use mpiObjectModuleId() to identify what source points to.

See Also

mpiObjectModuleId | MPIEventType | MPIEventMgr | MPINotify | MEIEventStatusInfo | MPIEventStatusINFO_COUNT_MAX

MEIEventStatusInfo

MEIEventStatusInfo

```
typedef struct MEIEventStatusInfo {
    union {
        MPIHandle    handle;    /* generic */
        MPIAxis      axis;      /* MEIEventTypeAXIS_FIRST ... MEIEventTypeAXIS_LAST -
1 */
        long         number;    /* MPIEventTypeMOTION
                    MPIEventTypeMOTOR_FIRST ... MPIEventTypeMOTOR_LAST - 1
                    MEIEventTypeMOTOR_FIRST ... MEIEventTypeMOTOR_LAST - 1 */
        long         value;     /* MPIEventTypeEXTERNAL */
            } type;

            MEIXmpSignalID signalID;

            /* Contents of addresses specified by MEIEventNotifyData{} */
            union {
                  long sampleCounter;
                  struct {
                      long sampleCounter;
                  } motion;
                  struct {
                       long sampleCounter;
                       long actualPosition;
                  } axis;
                  struct {
                       long sampleCounter;
                       long encoderPosition;
                  } motor;
                  long word[MEIXmpSignalUserData];
             } data;
} MEIEventStatusInfo;
```

Description

EventStatusInfo is an information structure that tells the XMP what the data in MPIEventStatus.info holds.

type	A union that specifies the object handle, motion number, or external ID value that generated the event
type.handle	A generic object handle. Used by MPIRecorder and MPIMotor events
type.axis	An axis object handle. Used by MPIAxis events
type.number	The motion number of the MPIMotion object that generated the event
type.value	An ID value used to identify what external source or MPISequence event was generated
signalID	Specifies what type of object actually generated the event
data	A union that contains extra data about the event that was generated
data.sampleCounter	The value of the sampleCounter when the event was generated
data.motion	A union that contains extra data about the motion event that was generated
data.motion.sampleCounter	The value of the sampleCounter when the motion event was generated
data.axis	A union that contains extra data about the axis event that was generated
data.axis.sampleCounter	The value of the sampleCounter when the axis.event was generated
data.axis.actualPosition	The value of the axis' actual position when the event was generated
data.motor	A union that contains extra data about the motor event that was generated

data.motor.sampleCounter	The value of the sampleCounter when the motor event was generated
data.motor.encoderPosition	The value of the motor's ecoder position when the event was generated
data.word[]	The extra data about the event that was generated formatted as an array of long values

Sample Code

```
MPINotify      notify
MPIEventStatus eventStatus;

. . .

/* Wait for event */
returnValue =
    mpiNotifyEventWait(notify,
                             &eventStatus,
                              MPIWaitFOREVER);
msgCHECK(returnValue);

if (eventStatus.type == MPIEventTypeMOTION_DONE) {
    MEIEventStatusInfo *info;

    info = (MEIEventStatusInfo *)eventStatus.info;

    . . .
}
```

See Also

MPIEventStatus | MPIAxis

MPIEventType / MEIEventType

MPIEventType

```
typedef enum {
    MPIEventTypeINVALID,

    MPIEventTypeNONE,           /*  0 */

    /* Motor events */
    MPIEventTypeAMP_FAULT,      /*  1 */
    MPIEventTypeHOME,           /*  2 */
    MPIEventTypeLIMIT_ERROR,    /*  3 */
    MPIEventTypeLIMIT_HW_NEG,   /*  4 */
    MPIEventTypeLIMIT_HW_POS,   /*  5 */
    MPIEventTypeLIMIT_SW_NEG,   /*  6 */
    MPIEventTypeLIMIT_SW_POS,   /*  7 */
    MPIEventTypeENCODER_FAULT,  /*  8 */

    /* Motion events */
    MPIEventTypeMOTION_DONE,            /*  9 */
    MPIEventTypeMOTION_AT_VELOCITY,     /* 10 */

    /* Recorder events */
    MPIEventTypeRECORDER_FULL,  /* 11 */
    MPIEventTypeRECORDER_DONE,  /* 12 */

    /* External events */
    MPIEventTypeEXTERNAL,       /* 13 */
} MPIEventType;
```

Description

EventType is used by the MPIEventMask macros to help generate event masks.

MPIEventTypeNONE	This event type indicates no event was generated.
MPIEventTypeAMP_FAULT	This event type indicates an Amp Fault event was generated from a Motor object.
MPIEventTypeHOME	This event type indicates a Home event was generated from a Motor object.
MPIEventTypeLIMIT_ERROR	This event type indicates a position Error Limit was generated from a Motor object.
MPIEventTypeLIMIT_HW_NEG	This event type indicates a Negative Hardware Limit event was generated from a Motor object.
MPIEventTypeLIMIT_HW_POS	This event type indicates a Positive Hardware Limit event was generated from a Motor object.
MPIEventTypeLIMIT_SW_NEG	This event type indicates a Negative Software Limit event was generated from a Motor object.
MPIEventTypeLIMIT_SW_POS	This event type indicates a Positive Software Limit event was generated from a Motor object.
MPIEventTypeENCODER_FAULT	This event type indicates an Encoder Fault event was generated from a Motor object.

MPIEventTypeMOTION_DONE	This event type indicates a Motion Done event was generated from a Motion Supervisor object.
MPIEventTypeMOTION_AT_VELOCITY	This event type indicates an At Velocity event was generated from a Motion Supervisor object.
MPIEventTypeRECORDER_FULL	This event type indicates a Recorder Full event was generated from a Recorder object.
MPIEventTypeRECORDER_DONE	This event type indicates a Recorder Done event was generated from a Recorder object.
MPIEventTypeEXTERNAL	This event type indicates an External event was generated from an external source.

MEIEventType

```
typedef enum {
        /* Motor events */
        MEIEventTypeLIMIT_USER0 = MPIEventTypeLAST,     /* 14 */
        MEIEventTypeLIMIT_USER1,        /* 15 */
        MEIEventTypeLIMIT_USER2,        /* 16 */
        MEIEventTypeLIMIT_USER3,        /* 17 */
        MEIEventTypeLIMIT_USER4,        /* 18 */
        MEIEventTypeLIMIT_USER5,        /* 19 */
        MEIEventTypeLIMIT_USER6,        /* 20 */
        MEIEventTypeLIMIT_USER7,        /* 21 */
        MEIEventTypeLIMIT_TORQUE,       /* 22 */

        /* Motion events */
        MEIEventTypeMOTION_OUT_OF_FRAMES,       /* 23 */
        MEIEventTypeMOTION_RESERVED0,           /* 24 */

        /* Axis events */
        MEIEventTypeIN_POSITION_COARSE,         /* 25 */
        MEIEventTypeIN_POSITION_FINE,           /* 26 */
        MEIEventTypeAT_TARGET,                  /* 27 */
        MEIEventTypeFRAME,                      /* 28 */
        MEIEventTypeAXIS_RESERVED0,             /* 29 */
        MEIEventTypeAXIS_RESERVED1,             /* 30 */
        MEIEventTypePOWER_FAILURE,              /* 31 */

} MEIEventType;
```

Description

EventType is used by the MPIEventMask macros to help generate event masks.

MEIEventTypeLIMIT_USER0	This event type indicates a User Limit event was generated from a Motor object.
MEIEventTypeLIMIT_USER1	This event type indicates a User Limit event was generated from a Motor object.
MEIEventTypeLIMIT_USER2	This event type indicates a User Limit event was generated from a Motor object.
MEIEventTypeLIMIT_USER3	This event type indicates a User Limit event was generated from a Motor object.
MEIEventTypeLIMIT_USER4	This event type indicates a User Limit event was generated from a Motor object.
MEIEventTypeLIMIT_USER5	This event type indicates a User Limit event was generated from a Motor object.
MEIEventTypeLIMIT_USER6	This event type indicates a User Limit event was generated from a Motor object.
MEIEventTypeLIMIT_USER7	This event type indicates a User Limit event was generated from a Motor object.
MEIEventTypeLIMIT_TORQUE	This event type indicates a Torque event was generated from a Motor object.
MEIEventTypeMOTION_OUT_OF_FRAMES	This event type indicates a Motion Done event was generated from a Motion Supervisor object.
MEIEventTypeMOTION_RESERVED0	This event type indicates a Reserved Motion event was generated from a Motion Supervisor object. This event type is reserved for future use or custom motion events.
MEIEventTypeIN_POSITION_COARSE	This event type indicates an In Coarse Position event was generated from an Axis object.
MEIEventTypeIN_POSITION_FINE	This event type indicates an In Fine Position event was generated from an Axis object.
MEIEventTypeAT_TARGET	Rserved Frame Event.
MEIEventTypeFRAME	This is currently not supported and is reserved for future use.
MEIEventTypeAXIS_RESERVED0	This event type indicates a Reserved Axis event was generated from an Axis object. This event type is reserved for future use or custom axis events.
MEIEventTypeAXIS_RESERVED1	This event type indicates a Reserved Axis event was generated from an Axis object. This event type is reserved for future use or custom axis events.
MEIEventTypePOWER_FAILURE	This event type indicates a Power Failure event was generated from the controller.

See Also

MPIEventMask | MPIEventMgr | MPINotify | MPIEventStatus
Special Note on the use of MPIEventTypeENCODER_FAULT

MPIEventStatusINFO_COUNT_MAX

MPIEventStatusINFO_COUNT_MAX

```
#define MPIEventStatusINFO_COUNT_MAX    (16)
```

Description **EventStatusINFO_COUNT_MAX** defines the size of the MPIEventStatus.info[] array.

See Also MPIEventStatus | MPIEventMgr | MPINotify

Special Note: *Use of MPIEventTypeENCODER_FAULT*

This event type is used to detect three types of encoder faults:

- **Broken wire errors**
- **Illegal state errors**
- **Absolute encoder initialization errors**
    - Timeout errors
    - Protocol errors

Broken wire errors are detected for either incremental or absolute encoders whenever both differential inputs of any encoder receiver (A, B, or Index) are at the same voltage level (i. e., whenever one or both inputs is disconnected from the encoders differential transmitter). The EncoderTermination configuration of the encoder input must be TRUE for correct detection of broken wires.

Illegal state errors occur whenever transitions are seen on both A and B phases of an encoder input at the same time (e.g. noise spikes).

There are two types of **absolute encoder initialization errors**: Timeout errors and Protocol errors. **Timeout errors** occur when an absolute encoder does not transmit absolute encoder data within the timeout period starting at the transition of the interrogation line (SEN line). **Protocol errors** are detected when serial absolute data is sent during the timeout, but the data cannot be interpreted by the XMP. Both error types result in an ENCODER_FAULT event.

Return to MPIEventType