Global Objects

Introduction

Data types that are used by more than one module are defined in the mpidef.h header file.The definitions listed in this section are available to all modules, and are defined in mpidef.h.

The object.h header file contains the data type definition for a generic object handle (MPIHandle) and an invalid object handle (MPIHandleVOID). The object.h header file also contains declarations of generic mpiObject functions, the definitions for which are in the object.c source file. These functions take a generic object handle (of type MPIHandle) as their first argument. Any MPI object handle may be treated as an MPIHandle and passed as the first argument to these functions.

Data Types

MPI**Action**

MPI**IoSource**

MPI**IoType**

MPI**IoTrigger**

MPI**ModuleId** / MEI**ModuleId**

MPI**State**

MPI**Status**

MPI**StatusFlag**

MPI**StatusMask**

MPI**Trajectory**

MPI**Wait**

Macros

mpi**StatusMaskBIT**

MPIAction

MPIAction

```
typedef enum {
    MPIActionINVALID,

    MPIActionNONE,
    MPIActionSTOP,
    MPIActionE_STOP,
    MPIActionE_STOP_ABORT,
    MPIActionABORT,

    MPIActionDONE,
    MPIActionSTART,
    MPIActionRESUME,
    MPIActionRESET,
} MPIAction;
```

Description

MPIActionNONE	Performs no action. Use with MPIMotorEventConfig to prevent a motor event from performing an action.
MPIActionSTOP	Makes a motion supervisor perform a stop. This action can be commanded with mpiMotionAction(…) or by a motor event on the controller. Please see MPIMotionDecelTime for more information about stop actions.
MPIActionE_STOP	Makes a motion supervisor perform an e-stop. This action can be commanded with mpiMotionAction(…) or by a motor event on the controller. Please see MPIMotionDecelTime for more information about e-stop actions.
MPIActionE_STOP_ABORT	Makes a motion supervisor perform an e-stop and then an abort. This action can be commanded with mpiMotionAction(…) or by a motor event on the controller. Please see MPIMotionDecelTime for more information about e-stop actions.
MPIActionABORT	Makes a motion supervisor perform an abort. This action can be commanded with mpiMotionAction(…) or by a motor event on the controller.
MPIActionDONE	is currently not supported and is reserved for future use.
MPIActionSTART	Intended to force a motion supervisor to start when it is waiting for some event (a delay or hold) before starting. This action is currently not supported.
MPIActionRESUME	Makes a motion supervisor to resume motion after a stop action has occurred. A motion supervisor can only resume a motion after a stop event, not an e-stop event. This action can be commanded with mpiMotionAction(…).

MPIActionRESET	Makes a motion supervisor return to an idle state after an error has occurred or after a stop, e-stop, abort, or e-stop/abort action has occurred. While abort actions and certain errors cause all associated motors to turn off their amp-enable lines, this action does not change the state of any amp-enable lines. One will have to call the method mpiMotorAmpEnableSet(…) to re-enable the amplifiers. This action can be commanded with mpiMotionAction(…).

Remarks

MPIAction enumerations are used to perform some sort of action on an MPI object. Currently, only MPIMotion and MPIMotor use the MPIAction enumerations. One can command an MPIMotion object to perform some action with the mpiMotionAction(…) method, while one can get and set the types of actions that will be performed when certain motor events occur with the MPIMotorEventConfig structure with the mpiMotorEventConfigGet(…) and mpiMotorEventConfigSet(…) methods.

An MPIAction can be generated from the host or the firmware. Below is a table where MPIActions originate (start):

MPIAction	Originating from Host	Originating from XMP Firmware
Start	mpiMotionAction(…) (currently unsupported)	NEVER
Resume	mpiMotionAction(…)	NEVER
Reset	mpiMotionAction(…)	NEVER
Stop	mpiMotionAction(…)	Event
E_Stop	mpiMotionAction(…)	Event
ABORT	mpiMotionAction(…)	Event
DONE	NEVER	NEVER

See Also

mpiMotionAction | MPIMotionDecelTime | MPIMotorEventConfig
mpiMotorEventConfigGet | mpiMotorEventConfigSet | MPIEvent

MPIIoSource

MPIIoSource

```
typedef union {
    MPIHandle   motor;  /* MOTOR */
    long        index;  /* USER */
} MPIIoSource;
```

Description

motor	Handle to a motor object that is the source for the IO.
index	Value of the index for a user input. User IO's are no longer supported by the xmp (user IO's are handled through the motor object).

See Also

MPIIoType

MPIIoType

```
typedef enum {
    MPIIoTypeINVALID,

    MPIIoTypeMOTOR,
    MPIIoTypeUSER,

} MPIIoType;
```

Description

MPIIoTypeMotor	Value specifies the IO type as motor (the IO source is a motor object).
MPIIoTypeUSER	Value specifies the IO type as user (User IO types are currently not supported. User IO is available through the motor objects).

See Also

MPIIoTrigger

MPIIoTrigger

```
typedef struct MPIIoTrigger {
    MPIIoType        type;
    MPIIoSource      source;
    unsigned long    mask;
    unsigned long    pattern;
} MPIIoTrigger;
```

Description

type	see MPIIoType.
source	see MPIIoSource.
mask	Value that specifies the mask to be applied to the IO.
pattern	Value that specifies the pattern to be compared to the masked IO.

See Also

MPIModuleId / MEIModuleId

MPIModuleId

```
typedef enum {
    MPIModuleIdINVALID,

    MPIModuleIdMESSAGE,
    MPIModuleIdADC,
    MPIModuleIdAXIS,
    MPIModuleIdCAPTURE,
    MPIModuleIdCOMMAND,
    MPIModuleIdCOMPARE,
    MPIModuleIdCONTROL,

    MPIModuleIdEVENT,
    MPIModuleIdEVENTMGR,
    MPIModuleIdFILTER,
    MPIModuleIdIDN,
    MPIModuleIdIDNLIST,
    MPIModuleIdMOTION,
    MPIModuleIdMOTOR,
    MPIModuleIdNODE,
    MPIModuleIdNOTIFY,
    MPIModuleIdPATH,
    MPIModuleIdRECORDER,
    MPIModuleIdSEQUENCE,
    MPIModuleIdSERCOS,
    MPIModuleIdSYNQNET,  /* This is a duplicate of MEIModuleIdSYNQNET
                         only for use in calling functions in xmp.c */
    MPIModuleIdBLOCK,    /* This is a duplicate of MEIModuleIdBLOCK only
                         for use in calling functions in xmp.c */

    MPIModuleIdEXTERNAL,

    MPIModuleIdMAX = 0xFF
} MPIModuleId;
```

Description

ModuleId is used to identify what module a particular MPIHandle belongs to. If the handle is an external memory pointer instead of an MPI object handle, MPIModuleIdEXTERNAL will be returned by MPI methods.

MEIModuleId

```
typedef enum {
    MEIModuleIdPLATFORM = MPIModuleIdEXTERNAL,

    MEIModuleIdCAN,

    MEIModuleIdCLIENT,
    MEIModuleIdELEMENT,
    MEIModuleIdFLASH,
    MEIModuleIdLIST,
    MEIModuleIdMAP,
    MEIModuleIdPACKET,
    MEIModuleIdSERVER,
    MEIModuleIdSYNQNET,
    MEIModuleIdBLOCK,
    MEIModuleIdSNDRIVE,
```

```
}MEIModuleId;
```

Description

ModuleId is used to identify what module a particular MPIHandle belongs to. If the handle is an external memory pointer instead of an MPI object handle, MPIModuleIdEXTERNAL will be returned by MPI methods.

See Also

mpiObjectModuleId | mpiObjectValidate

MPIState

MPIState

```
typedef enum {
    MPIStateINVALID,

    MPIStateINIT,
    MPIStateIDLE,
    MPIStateMOVING,
    MPIStateSTOPPING,
    MPIStateSTOPPING_ERROR,
    MPIStateERROR,
} MPIState;
```

Description

State enumerations define basic states the motion is in. MPIState resides in the MPIStatus structure. Currently MPIState is only used with motion module.

MPIStateINIT	The of the motion is performing an initialization.
MPIStateIDLE	The state of motion is idle and waiting to resume motion.
MPIStateMOVING	The state of the motion is moving.
MPIStateSTOPPING	The state of the motion is stopping. This occurs from a Stop event, but not an E_Stop, E_Stop Abort, or Abort events.
MPIStateSTOPPING_ERROR	The state of the motion is performing an emergency stop and/or abort on all axes.
MPIStateERROR	The state of the motion is in error. The error state is generated from an E_Stop or Abort event.

See Also

MPIStatus

MPIStatus

MPIStatus

```
typedef struct MPIStatus {
    MPIState        state;
    MPIAction       action;
    MPIEventMask    eventMask;

    long    settled;
    long    atTarget;

    MPIStatusMask    statusMask;
} MPIStatus;
```

Description

state	Value that indicates the state of an xmp controller's motion supervisor.
action	Value that indicates the action to perform for a motion supervisor.
eventMask	Array that defines the event mask bits. The array is defined as typedef MPIEventMaskELEMENT_TYPE MPIEventMask[MPIEventMaskELEMENTS].
settled	Value that indicates if an axis associated with a motion supervisor has settled (is in fine position).
atTarget	Value that indicates if an axis associated with a motion supervisor has completed its command trajectory (i.e. the command position has reached the targeted end point of the move).

See Also

Note on status information if using a SERCOS controller.
MPIState | MPIStatusFlag | MPIStatusMask

MPIStatusFlag

MPIStatusFlag

```
typedef enum {
    MPIStatusFlagINVALID,

    MPIStatusFlagCOMM_ERROR,

} MPIStatusFlag;
```

Description

See Also

Special Note on status information when using a SERCOS controller.

MPIStatusMask

MPIStatusMask

```
typedef enum {
    MPIStatusMaskNONE   = 0x0,

    MPIStatusMaskCOMM_ERROR     = mpiStatusMaskBIT(MPIStatusFlagCOMM_ERROR),
                                        /* 0x00000001 */

    MPIStatusMaskMOTOR  = MPIStatusMaskCOMM_ERROR,
                                        /* 0x00000001 */

    MPIStatusMaskALL    = mpiStatusMaskBIT(MPIStatusFlagLAST) - 1
                                        /* 0x00000001 */
} MPIStatusMask;
```

Description

MPIStatusMaskCOMM_ERROR	Value for the status mask that indicates a commutation error has occurred.
MPIStatusMaskMOTOR	Value specifies the motor's status mask.
MPIStatusMaskALL	Value specifies the status mask that encompasses all the possible status flags.

See Also

Note on status information if using a SERCOS controller.
MPIStatus | MPIStatusFlag

MPITrajectory

MPITrajectory

```
typedef struct MPITrajectory {
    double    velocity;
    double    acceleration;
    double    deceleration;
    double    jerkPercent;
    double    accelerationJerk;
    double    decelerationJerk;
} MPITrajectory;
```

Description

The **Trajectory** structure is within the MPIMotionVelocity structure which in turn is within the MPIMotionParams structure. The data contained in MPITrajectory are the parameters used in certain motion profiles.

See Also

MPIMotionParams | MPIMotionSCurve | MPIMotionTrapezoidal | MPIMotionType
MPIMotionVelocity | mpiMotionTrajectory

MPIWait

MPIWait

```
typedef enum {
    MPIWaitFOREVER = -1,
    MPIWaitPOLL = 0,
    MPIWaitMSEC
} MPIWait;
```

Description

Wait enumerations define basic wait times for certain MPI methods.

MPIWaitFOREVER	Makes MPI methods wait forever for an event to occur before returning.
MPIWaitPOLL	Makes MPI methods see if a certain event has occurred. If an event has not occurred, then the MPI method will generally return immediately returning the value MPIMessageTIMEOUT.
MPIWaitMSEC	Defines a period of one millisecond. If used alone, this will make MPI methods wait for one millisecond for an event occurs before returning. One can pass an an agument a multiple of MPIWaitMSEC to make MPI methods wait longer periods of time. For example, the following statement will make mpiPlatformKey wait 5 milliseconds for a user keystroke: mpiPlatformKey(5 * MPIWaitMSEC); If an event does not occur within the specified time, MPI methods will generally return the value MPIMessageTIMEOUT.

WARNING

The MPI depends on the ability of the operating system it is running on to be able to activate threads or put threads to sleep for a specified period of time in order for these times to be accurate. Microsoft Windows platforms are not real-time operating systems and are known to be unable to activate threads any quicker than 10 milliseconds. If you encounter a timing problem, it is likely an operating system timing issue.

See Also

mpiControlInterruptWait | mpiNotifyEventWait | mpiObjectTimeoutGet
mpiObjectTimeoutSet | meiPlatformKey

mpiStatusMaskBIT

Declaration `#define mpiStatusMaskBIT(flag) (0x1 << (flag))`

Required Header stdmpi.h

Description **StatusMaskBIT** converts the status flag into the status mask.

See Also MPIStatusFlag | MPIStatusMask