meiDeprecated Objects

Introduction

In an effort to provide the most comprehensive library of functions, the MPI library is constantly being updated and improved. In every software release, there may be a variety of additions, revisions, and deletions to the header files. However, in order to maintain backwards compatibility, selected data types and functions have been preserved in meiDeprecated.h and are still supported. Although the data types and methods listed below are still supported, their use in future applications is discouraged.

Methods

meiCanFlashNodeConfigGet
meiCanFlashNodeConfigSet
meiCanNodeDigitalInputGet
meiCanNodeDigitalInputsGet
meiCanNodeDigitalOutputGet
meiCanNodeDigitalOutputsGet
meiCanNodeDigitalOutputSet
meiCanNodeDigitalOutputsSet
meiControlEnabledLmbGet
meiControlEnabledLmbSet
mpiControlIoGet
mpiControlIoSet
mpiControlIoBitGet
mpiControlIoBitSet
meiControlVersionGet
meiControlVersionSet
mpiMotorIoGet
mpiMotorIoSet
mpiMotorFeedbackGet
meiPlatformCreate
meiPlatformDelete
meiSqNodeAnalogInput

Data Types

MEICanDigitalIO

MPIControlIoWords

MPIControlIo

MEIControlVersion

MEIMotorDedicatedIn

MEIMotorInput

MEIMotorDedicatedOut

MEIMotorDedicatedOVERTRAVEL_POS

MEIMotorDedicatedOVERTRAVEL_NEG

MEIMotorDedicatedInINDEX

MEIMotorGeneralIo

MEIMotorInput

MEIMotorIoMask

MPIMotorIo

MEIFilterDataType

MEISynqNetMessageNODE_UNAVAILABLE

MEISynqNetMessageRESPONSE_TIMEOUT

MEISynqNetMessageREADY_TIMEOUT

MEISynqNetMessageSRVC_ERROR

MEISynqNetMessageSRVC_UNSUPPORTED

MEISynqNetMaxMotorENCODER_COUNT

Constants

MEIMotorDedicatedOVERTRAVEL_POS

MEIMotorDedicatedOVERTRAVEL_NEG

meiCanFlashNodeConfigGet (Deprecated)

NOTE:
meiCanFlashNodeConfigGet was moved to meiDeprecated.h in the 03.01.00 MPI software release.

Declaration

```
long  meiCanFlashNodeConfigGet(MEICan            can,
                               void*             flash,
                               long              node,
                               MEICanNodeConfig* nodeConfig);
```

Required Header: stdmei.h

Description

meiCanFlashNodeConfigGet returns a copy of the current flash configuration of the CAN controller.

can	a handle to the Can object
flash	normally NULL
node	the node number of the CANOpen node
nodeConfig	a pointer to the CAN node configuration structure that will be filled in by this function

Return Values	
MPIMessageOK	

See Also

meiCanNodeFlashConfigSet

meiCanFlashNodeConfigSet (Deprecated)

NOTE:
meiCanFlashNodeConfigSet was moved to meiDeprecated.h in the 03.01.00 MPI software release.

Declaration

```
long  meiCanFlashNodeConfigSet(MEICan             can,
                               void*              flash,
                               long               node,
                               MEICanNodeConfig*  nodeConfig);
```

Required Header: stdmei.h

Description

meiCanFlashNodeConfigSet updates the current flash configuration for the node.

can	a handle to the Can object
flash	normally NULL
node	the node number of the CANOpen node
nodeConfig	a pointer to the CAN node configuration structure containing the new configuration

Return Values	
MPIMessageOK	

See Also

meiCanNodeFlashConfigGet

meiCanNodeDigitalInputGet (Deprecated)

NOTE:
meiCanNodeDigitalInputGet was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
long meiCanNodeDigitalInputGet(MEICan     can,
                               long       node,
                               long       bit,
                               long*      data);
```

Required Header: stdmei.h

Description

meiCanNodeDigitalInputGet gets the current state of the digital input bit on the specified CAN node.

(Not to be confused with meiCanNodeDigitalInputsGet.)

can	Handle to the CAN object.
node	The node number of the CANOpen node.
bit	Which bit on this node. This value should be between 0 and 63.
data	A pointer to where the current digital bit is returned. The value returned will be either 0 or 1.

Return Values	
MPIMessageOK	

Sample Code

The following sample code shows how to interrogate the current state of a single digital input bit on a controller. The variable **data** will contain either one or zero depending on the electrical signal being applied to the input pin on the CANOpen node. See meiCanCreate on how to create the CANHandle.

```
long data;
long Result;
Result = meiCanNodeDigitalInputGet(CANHandle,
                                   3, /*node*/
                                   0, /*bit*/
                                   &data );
```

See Also

meiCanCreate

meiCanNodeDigitalInputsGet (Deprecated)

NOTE:
meiCanNodeDigitalInputsGet was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
long meiCanNodeDigitalInputsGet(MEICan              can,
                                long                node,
                                MEICanDigitalIO*    data);
```

Required Header: stdmei.h

Description

meiCanNodeDigitalInputsGet gets the current state of all the digital input bits on the specified CAN node.

(Not to be confused with meiCanNodeDigitalInputGet.)

can	handle to the CAN object
node	the node number of the CANOpen node.
data	a pointer to where the current digital bits are returned.

Return Values	
MPIMessageOK	

See Also

meiCanNodeDigitalInputGet

meiCanNodeDigitalOutputGet (Deprecated)

NOTE:
meiCanNodeDigitalOutputGet was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
long meiCanNodeDigitalOutputGet(MEICan    can,
                                long      node,
                                long      bit,
                                long*     data);
```

Required Header: stdmei.h

Description

meiCanNodeDigitalOutputGet gets the current state of the digital output bit on the specified CAN Node.

(Not to be confused with meiCanNodeDigitalOutputsGet.)

can	handle to the CAN object
node	the node number of the CANOpen node.
bit	which bit on this node.
data	a pointer to where the current digital bit is returned.

Return Values	
MPIMessageOK	

See Also

meiCanNodeDigitalOutputSet | meiCanNodeDigitalOutputsGet | meiCanNodeDigitalOutputsSet

meiCanNodeDigitalOutputsGet (Deprecated)

NOTE:
meiCanNodeDigitalOutputsGet was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
long meiCanNodeDigitalOutputsGet(MEICan              can,
                                 long                node,
                                 MEICanDigitalIO*    data);
```

Required Header: stdmei.h

Description

meiCanNodeDigitalOutputsGet gets the current state of all the digital output bits on the specified CAN node.

(Not to be confused with meiCanNodeDigitalOutputGet.)

can	handle to the CAN object
node	the node number of the CANOpen node.
data	a pointer to where the current digital bit is returned.

Return Values	
MPIMessageOK	

See Also

meiCanNodeDigitalOutputGet | meiCanNodeDigitalOutputSet | meiCanNodeDigitalOutputsSet

meiCanNodeDigitalOutputSet (Deprecated)

NOTE:
meiCanNodeDigitalOutputSet was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
long meiCanNodeDigitalOutputSet(MEICan    can,
                                long      node,
                                long      bit,
                                long      data);
```

Required Header: stdmei.h

Description

meiCanNodeDigitalOutputSet changes the state of the digital output bit on the specified CAN Node.

(Not to be confused with meiCanNodeDigitalOutputsSet.)

can	Handle to the CAN object
node	The node number of the CANOpen node.
bit	Which bit on this node. This value should be between 0 and 63.
data	The new state of the digital bit. The value returned will be either 0 or 1.

Return Values	
MPIMessageOK	

See Also

meiCanNodeDigitalOutputGet | meiCanNodeDigitalOutputsGet | meiCanNodeDigitalOutputsSet

meiCanNodeDigitalOutputsSet (Deprecated)

NOTE:
meiCanNodeDigitalOutputsSet was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
long meiCanNodeDigitalOutputsSet(MEICan                can,
                                 long                  node,
                                 MEICanDigitalIO*      data);
```

Required Header: stdmei.h

Description

meiCanNodeDigitalOutputsSet changes the current state of all the digital output bits on the specified CAN node.

(Not to be confused with meiCanNodeDigitalOutputSet.)

can	handle to the CAN object
node	the node number of the CANOpen node.
data	the new data of the digital bits.

Return Values	
MPIMessageOK	

See Also

meiCanNodeDigitalOutputGet | meiCanNodeDigitalOutputSet | meiCanNodeDigitalOutputsGet

meiControlEnabledLmbGet (Deprecated)

NOTE:
meiControlEnabledLmbGet was moved to meiDeprecated.h in the 20030420 MPI software release.

Declaration

```
long  meiControlEnabledLmbGet  (MPIControl   control,
                                long         *count);
```

Required Header: stdmei.h

Description

meiControlEnabledLmbGet is for backwards compatability only (always returns zero).

control	a handle to the Control object.
***count**	a pointer to the location at which to write the number of enabled Local Motion Blocks (LMBs).

Return Values	
MPIMessageOK	

See Also

meiControlEnabledLmbSet

meiControlEnabledLmbSet (Deprecated)

NOTE:
meiControlEnabledLmbSet was moved to meiDeprecated.h in the 20030420 MPI software release.

Declaration

```
/* for backwards compatability only (does nothing) */
long  meiControlEnabledLmbSet  (MPIControl   control,
                                 long         *count);
```

Required Header: stdmei.h

Description

meiControlEnabledLmbSet is for backwards compatability only. It does nothing.

control	a handle to the Control object.
***count**	a pointer to the location that stores the number of Local Motion Blocks (LMBs) to enable.

Return Values	
MPIMessageOK	

See Also

meiControlEnabledLmbGet

mpiControlIoGet (Deprecated)

NOTE:
mpiControlIoGet was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
long mpiControlIoGet(MPIControl        control,
                     MPIControlIo      *io);
```

Required Header: stdmei.h

Description

mpiControlIoGet reads the states of a controller's digital inputs and writes them into a structure pointed to by ***io***. Some controller models have local digital I/O. Please see the controller hardware documentation for details.

control	a handle to a Control object
***io**	a pointer to a structure containing the digital input and output values.

Return Values	
MPIMessageOK	

See Also

mpiControlIoSet | MPIControlInput | MPIControlOutput

mpiControlIoSet (Deprecated)

NOTE:
mpiControlIoSet was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
long mpiControlIoSet(MPIControl        control,
                     MPIControlIo      *io);
```

Required Header: stdmei.h

Description

mpiControlIoSet writes the states of a controller's digital I/O using data from a structure pointed to by ***io***. Some controller models have local digital I/O. Please see the controller hardware documentation for details.

control	a handle to a Control object
***io**	a pointer to a structure containing the digital input and output values.

Return Values	
MPIMessageOK	

See Also

mpiControlIoGet | MPIControlInput | MPIControlOutput

meiControlIoBitGet (Deprecated)

NOTE:
meiControlIoBitGet was moved to meiDeprecated.h in the 03.03.00 MPI software release.
Please see Transitioning to the New Motor I/O Functions.

Declaration

```
long  meiControlIoBitGet(MPIControl        control,
                         MEIControlIoBit   bit,
                         MPI_BOOL          *value);
```

Required Header: stdmei.h
Change History: Modified in the 03.03.00

Description

meiControlIoBitGet reads the state of a controller digital input bit and writes it into a long pointed to by ***value***. Some controller models have local digital I/O. Please see the controller hardware documentation for details.

control	a handle to the Control object
bit	an enumerated bit number
***value**	a pointer to a long. The value contains the state of the bit.

Return Values	
MPIMessageOK	

See Also

meiControlIoBitSet | MEIControlIoBit

meiControlIoBitSet (Deprecated)

NOTE:
meiControlIoBitSet was moved to meiDeprecated.h in the 03.03.00 MPI software release.
Please see Transitioning to the New Motor I/O Functions.

Declaration

```
long  meiControlIoBitSet(MPIControl          control,
                         MEIControlIoBit     bit,
                         MPI_BOOL            *value);
```

Required Header: stdmei.h
Change History: Modified in the 03.03.00

Description

meiControlIoBitSet writes the state of a controller digital output bit using data from a value pointed to by a long. Some controller models have local digital I/O. Please see the controller hardware documentation for details.

control	a handle to the Control object
bit	an enumerated bit number
***value**	a pointer to a long. The value contains the state of the bit.

Return Values	
MPIMessageOK	

See Also

meiControlIoBitGet | MEIControlIoBit

meiControlVersionGet (Deprecated)

NOTE:
meiControlVersionGet was moved to meiDeprecated.h in the 03.01.00 MPI software release.

Declaration

```
long  meiControlVersionGet(MPIControl          control,
                           MEIControlVersion   *version);
```

Required Header: stdmei.h

Description

meiControlVersionGet writes the the version numbers of the XMP firmware, hardware, and the MPI library to the structure pointed to by ***version***.

Return Values	
MPIMessageOK	

See Also

meiControlVersionSet

meiControlVersionSet (Deprecated)

NOTE:
meiControlVersionSet was moved to meiDeprecated.h in the 03.01.00 MPI software release.

Declaration

```
long  meiControlVersionSet(MPIControl         control,
                           MEIControlVersion  *version);
```

Required Header: stdmei.h

Description

meiControlVersionSet sets the version numbers of the XMP firmware, hardware, and the MPI library using data from the structure pointed to by version.

Normally, the MPI library is compatible only with the XMP firmware for which the library is specifically built; i.e., only when

version -> mpi.firmware.version == version -> xmp.firmware.version

However, there are times when it is desirable to have the MPI library ignore incompatible firmware and continue to operate. As an example, the flash utility instructs the MPI library to ignore firmware incompatibility when new firmware is being loaded. Of course, this new firmware should also be compatible with the MPI library. In such cases, the version -> xmp.firmware structure will be copied into ***control***.

Return Values	
MPIMessageOK	

See Also

meiControlVersionGet

mpiMotorIoGet (Deprecated)

NOTE:
mpiMotorIoGet was moved to meiDeprecated.h in the 03.02.00 MPI software release.

Declaration

```
long  mpiMotorIoGet(MPIMotor    motor,
                    MPIMotorIo  *io);
```

Required Header: stdmei.h

Description

mpiMotorIoGet gets a Motor's (***motor***) dedicated I/O bits and writes them into the structure pointed to by ***io***.

NOTE: When using I/O on SynqNet nodes, use the motor I/O masks in the drive's header file for cleaner code. Most SynqNet nodes have a header file that defines things that are specific to that drive. The header files are found in C:\mei\XMP\sqNodeLib\include. An example is shown below that reads the hall sensors from the Trust TA802.

```
/*
        Shows the hall sensors.
        Make sure you include trust_ta800.h. Ex:
        #include "C:\mei\XMP\sqNodeLib\include\trust_ta800.h"

        The hall sensor masks are found in the enum TA800MotorIoMask.
*/

void showHalls(MPIMotor motor)
{
    MPIMotorIo io;
      long returnValue;
      long a, b, c;

      while (meiPlatformKey(MPIWaitPOLL) <= 0)
      {
         returnValue = mpiMotorIoGet(motor, &io);
         msgCHECK(returnValue);

         a = ((io.input & TA800MotorIoMaskHALL_A) == TA800MotorIoMaskHALL_A);
         b = ((io.input & TA800MotorIoMaskHALL_B) == TA800MotorIoMaskHALL_B);
         c = ((io.input & TA800MotorIoMaskHALL_C) == TA800MotorIoMaskHALL_C);

         /* Prints a 1 or 0 indicating the hall state */
         printf("Hall A %d B %d C %d\t\t\r", a, b, c);

         meiPlatformSleep(100);
      }
}
```

Sample Code

```
/* Poll io inputs for a motor and print to the screen */
void readIo(MPIMotor motor)
{
    MPIMotorIo io;
    long returnValue;

    while (meiPlatformKey(MPIWaitPOLL) <= 0)
    {
        returnValue = mpiMotorIoGet(motor, &io);
        msgCHECK(returnValue);

        printf("\rIO %x", io.input);

        meiPlatformSleep(100); /* Wait 100 mSec */
    }
}
```

Return Values	
MPIMessageOK	

See Also

mpiMotorIoSet

mpiMotorIoSet (Deprecated)

NOTE:
mpiMotorIoSet was moved to meiDeprecated.h in the 03.02.00 MPI software release.

Declaration

```
long  mpiMotorIoSet(MPIMotor    motor,
                    MPIMotorIo  *io);
```

Required Header: stdmei.h

Description

mpiMotorIoSet sets a Motor's (***motor***) dedicated I/O bits using data from the structure pointed to by ***io***.

NOTE: When using I/O on SynqNet nodes, use the motor I/O masks in the drive's header file for clearer code. Most SynqNet nodes have a header file that defines things that are specific to that drive. The header files are found in C:\mei\XMP\sqNodeLib\include. An example is shown below that reads the hall sensors from the Trust TA802.

```
/*
        Shows the hall sensors.
        Make sure you include trust_ta800.h. Ex:
        #include "C:\mei\XMP\sqNodeLib\include\trust_ta800.h"

        The hall sensor masks are found in the enum TA800MotorIoMask.
*/

void showHalls(MPIMotor motor)
{
    MPIMotorIo io;
      long returnValue;
      long a, b, c;

      while (meiPlatformKey(MPIWaitPOLL) <= 0)
      {
         returnValue = mpiMotorIoGet(motor, &io);
         msgCHECK(returnValue);

         a = ((io.input & TA800MotorIoMaskHALL_A) == TA800MotorIoMaskHALL_A);
         b = ((io.input & TA800MotorIoMaskHALL_B) == TA800MotorIoMaskHALL_B);
         c = ((io.input & TA800MotorIoMaskHALL_C) == TA800MotorIoMaskHALL_C);

         /* Prints a 1 or 0 indicating the hall state */
         printf("Hall A %d B %d C %d\t\t\r", a, b, c);

         meiPlatformSleep(100);
        }
}
```

Return Values	
MPIMessageOK	

See Also

mpiMotorIoGet

mpiMotorFeedbackGet (Deprecated)

NOTE:
mpiMotorFeedbackGet was moved to meiDeprecated.h in the 20030402 MPI software release.

Declaration

```
long  mpiMotorFeedbackGet(MPIMotor   motor,  /* use mpiMotorFeedback(...) */
                          double     feedback);
```

Required Header: stdmei.h

Description

mpiMotorFeedbackGet gets the feedback position of a Motor (***motor***) and writes it into the location pointed to by ***feedback***.

Return Values	
MPIMessageOK	

See Also

meiPlatformCreate (Deprecated)

NOTE:
meiPlatformCreate was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
MEIPlatform  meiPlatformCreate(MPIControl  control);
```

Required Header: stdmei.h

Description

meiPlatformCreate is for internal purposes only.

Customers should NOT use meiPlatformCreate in motion applications.

meiPlatformDelete (Deprecated)

NOTE:
meiPlatformDelete was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Declaration

```
long  meiPlatformDelete(MEIPlatform   platform);
```

Required Header: stdmei.h

Description

meiPlatformDelete is for internal purposes only.

Customers should NOT use meiPlatformCreate in motion applications.

meiSqNodeAnalogInput (Deprecated)

NOTE:
meiSqNodeAnalogInput was moved to meiDeprecated.h in the 03.02.00 MPI software release.

Declaration

```
long  meiSqNodeAnalogInput(MEISqNode   node,
                           long        channel,
                           double      *state);
```

Required Header: stdmei.h

Description

meiSqNodeAnalogInput gets the current state of an analog input on a SynqNet node.

node	a handle to a SynqNet node object.
channel	the index of the analog input channel (with respect to the node).
***state**	a pointer to where the current state of the input is written by this function.

Return Values	
MPIMessageOK	

See Also

MEICanDigitalIO (Deprecated)

NOTE:
MEICanDigitalIO was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Definition

```
typedef struct MEICanDigitalIO {
    unsigned   long    data[2];
} MEICanDigitalIO;
```

Description

MEICanDigitalIO holds the state of all the digital inputs or outputs on a CANOpen node. The maximum number of inputs or outputs on a single node supports is 64.

data	Data associated with the command.

See Also

MPIControlIo (Deprecated)

NOTE:
MPIControlIo was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Definition

```
typedef struct MPIControlIo {
    unsigned long   input[MPIControlIoWords];
    unsigned long   output[MPIControlIoWords]
} MPIControlIo;
```

Description

MPIControlIo contains the controller's local digital input and output states. The digital inputs can be read and the digital outputs can be read or written through this structure.

input	An array of digital input values. Each bit mask is defined by the MEIControlInput enumeration.
output	An array of digital output values. Each bit mask is defined by the MEIControlOutput enumeration.

See Also

MEIControlOutput | MEIControlInput | mpiControlIoGet | mpiControlIoSet

MEIControlVersion (Deprecated)

NOTE:
MEIControlVersion was moved to meiDeprecated.h in the 03.01.00 MPI software release. It was replaced by MEIControlInfo.

Definition

```
/* replaced by MEIControlInfo */
typedef struct MEIControlVersion {
    struct {   /* control.c */
        char   *version;   /* MEIControlVersionMPI (YYYYMMDD) */


        struct {   /* xmp.h */
            long    version;    /* MEIXmpVERSION */
            long    option;     /* MEIXmpOPTION */
        } firmware;
    } mpi;

    struct {
        long    version; /* hardware version */

        struct {   /* MEIXmpData.SystemData{} */
            long    version;    /* MEIXmpVERSION_EXTRACT(SoftwareID) */
            char    revision;   /* ('A' - 1) + MEIXmpREVISION_EXTRACT(SoftwareID) */
            long    subRevision;    /* MEIXmpSUB_REV_EXTRACT(Option) */
            long    developmentId;  /* MEIXmpDEVELOPMENT_ID_EXTRACT(Option) */
            long    option;         /* MEIXmpOPTION_EXTRACT(Option) */
            long    userVersion;
            long    branchId;
     } firmware;

     struct {
        struct {
            long    version;
            long    option;
          } PLD;
          struct {
             char number[10];
             char rev[5];
          } model;
          struct {
             long    version;
          } FPGA;
       } board;
    } xmp;
    struct {
       char version[10];
    } driver;

} MEIControlVersion;
```

Description

MEIControlVersion is a structure that specifies the version information for the MPI and the controller's firmware, FPGAs, and the bus interface.

mpi	A structure that contains the version information of the MPI.
mpi.version	A string representing the version of the MPI. The version of the MPI is broken down by date, branch, and revision (MPIVersion.branch.revision). For ex: 20021220.1.2 means MPI version 20021220, branch 1, revision 2.
mpi.firmware	The firmware version information that the current version of the MPI will work with. A new field has been added to the XMP's firmware to identify and differentiate between intermediate branch software revisions. The branch value is represented as a hex number between 0x00000000 and 0xFFFFFFFF. Each digit represents an instance of a branch (0x1 to 0xF). A single digit represents a single branch from a specific version, two digits represent a branch of a branch, three digits represent a branch of a branch of a branch, etc.
xmp	A structure that contains the version information of the XMP controller.
xmp.firmware	The XMP's firmware version information.
xmp.motionBlock[]	An array of structures that contain version information about the motion blocks on the XMP.
xmp.board.PLD.version	A string that contains the controller's PLD version.
xmp.board.PLD.option	A string that contains the controller's PLD option.
xmp.board	An array of structures that contain version information about the XMP controller boards.
xmp.board.model.number	A string that contains the hardware model number for the controller.
xmp.board.model.rev	A string that contains the hardware revision for the controller.
xmp.board.FPGA.version	A string that contains the controller's FPGA version.
driver.version	A string containing device driver version information. This values is "N/A" if the driver version is not available from your operating system.

See Also

MPIControl | MEIControlInfo

MPIMotorIo (Deprecated)

NOTE:
MPIMotorIo was moved to meiDeprecated.h in the 03.02.00 MPI software release.

Definition

```
typedef struct MPIMotorIo {
    unsigned long       input;
    unsigned long       output;
} MPIMotorIo;
```

Description

MPIMotorIo is a structure used to read the Motor input and set Motor output values.

input	The **input** value reflects the 32 bits of general and dedicated input values. The dedicated input values (bits 0-15) are specified by MEIMotorDedicatedIn. The general purpose input values (bits 16-31) can be specified by MEIMotorIoMask.
output	The **output** value reflects the 32 bits of general and dedicated output values. The dedicated output values (bits 0-15) are specified by MEIMotorDedicatedOut. The general purpose output values (bits 16-31) can be specified by MEIMotorIoMask.

See Also

MEIMotorDedicatedIn | MEIMotorDedicatedOut | MEIMotorIoMask

MPIControlIoWords (Deprecated)

NOTE:
MPIControlIoWords was moved to meiDeprecated.h in the 03.03.00 MPI software release.

Definition

```
#define  MPIControlIoWords   (1)
```

Description

MPIControlIoWords defines the number of 32 bit Input and Output words on the controller.

See Also

MPIControlIo | MEIControlIoBit | MEIControlInput | MEIControlOutput

MEIMotorDedicatedIn (Deprecated)

NOTE:
MEIMotorDedicatedIn was moved to meiDeprecated.h in the 03.02.00 MPI software release.

Definition

```
typedef enum {
    MEIMotorDedicatedInAMP_FAULT        =  MEIXmpMotorDedicatedFlagsMaskAMP_FAULT,   /*
bit 0  */
    MEIMotorDedicatedInBRAKE_APPLIED    =  MEIXmpMotorDedicatedFlagsMaskBRAKE_ON,    /*
bit 1  */
    MEIMotorDedicatedInHOME             =  MEIXmpMotorDedicatedFlagsMaskHOME,        /*
bit 2  */
    MEIMotorDedicatedInLIMIT_HW_POS     =  MEIXmpMotorDedicatedFlagsMaskPOS_LIMIT,   /*
bit 3  */
    MEIMotorDedicatedInLIMIT_HW_NEG     =  MEIXmpMotorDedicatedFlagsMaskNEG_LIMIT,   /*
bit 4  */
    MEIMotorDedicatedInINDEX_PRIMARY    =  MEIXmpMotorDedicatedFlagsMaskENC_INDEX0, /*
bit 5  */
    MEIMotorDedicatedInFEEDBACK_FAULT   =  MEIXmpMotorDedicatedFlagsMaskENC_FAULT,   /*
bit 6  */
    MEIMotorDedicatedInCAPTURED         =  MEIXmpMotorDedicatedFlagsMaskCAPTURED,    /*
bit 7  */
    MEIMotorDedicatedInHALL_A           =  MEIXmpMotorDedicatedFlagsMaskHALL_A,      /*
bit 8  */
    MEIMotorDedicatedInHALL_B           =  MEIXmpMotorDedicatedFlagsMaskHALL_B,      /*
bit 9  */
    MEIMotorDedicatedInHALL_C           =  MEIXmpMotorDedicatedFlagsMaskHALL_C,      /*
bit 10 */
    MEIMotorDedicatedInAMP_ACTIVE       =  MEIXmpMotorDedicatedFlagsMaskAMP_ACTIVE, /*
bit 11 */
    MEIMotorDedicatedInINDEX_SECONDARY =  MEIXmpMotorDedicatedFlagsMaskENC_INDEX1,
/* bit 12 */
    MEIMotorDedicatedInWARNING          =  MEIXmpMotorDedicatedFlagsMaskWARNING,
/* bit 13 */
    MEIMotorDedicatedInDRIVE_STATUS_9   =  MEIXmpMotorDedicatedFlagsMaskDRIVE_STATUS9,
/* bit 14 */
    MEIMotorDedicatedInDRIVE_STATUS_10 =
MEIXmpMotorDedicatedFlagsMaskDRIVE_STATUS10, /* bit 15 */
} MEIMotorDedicatedIn;
```

Description

MEIMotorDedicatedIn is an enumeration of bit masks for the motor's dedicated inputs. The support for dedicated inputs is node/drive specific. See the node/drive manufacturer's documentation for details.

MEIMotorDedicatedInAMP_FAULT	Generated by the masked motor fault bits. Active when one or more masked motor faults bits are active. See MEIMotorFaultConfig.
MEIMotorDedicatedInBRAKE_APPLIED	Mechanical brake state.
MEIMotorDedicatedInHOME	Position calibration sensor.
MEIMotorDedicatedInLIMIT_HW_POS	Hardware limit for the positive direction.
MEIMotorDedicatedInLIMIT_HW_NEG	Hardware limit for the negative direction.
MEIMotorDedicatedInINDEX_PRIMARY	Primary encoder index input signal.
MEIMotorDedicatedInFEEDBACK_FAULT	Position feedback status. TRUE when position feedback fails, FALSE when operating properly.
MEIMotorDedicatedInCAPTURED	Currently not supported.
MEIMotorDedicatedInHALL_A	Reflects the state of Hall Sensor A
MEIMotorDedicatedInHALL_B	Reflects the state of Hall Sensor B
MEIMotorDedicatedInHALL_C	Reflects the state of Hall Sensor C
MEIMotorDedicatedInAMP_ACTIVE	A bit set by the drive that indicates the amplifier's state. 1 = Amplifier is closing the current loop and the motor winding are energized. 0 = Amplifier is not closing the current loop and the motor windings are not energized. Support for this bit varies depending on the drive type.
MEIMotorDedicatedInINDEX_SECONDARY	Secondary encoder index input signal.
MEIMotorDedicatedInWARNING	Drive warning state. 1 = drive warning status bit is active and warning message is available 0 = drive warning status bit is not active. Support for this bit varies depending on the drive type.
MEIMotorDedicatedInDRIVE_STATUS_9	State of bit 9 in the SynqNet drive specific status register.
MEIMotorDedicatedInDRIVE_STATUS_10	State of bit 10 in the SynqNet drive specific status register.

See Also

mpiMotorIoGet

MEIMotorInput (Deprecated)

NOTE:
MEIMototInput was moved to meiDeprecated.h in the 03.02.00 MPI software release.
It was replaced by MEIMotorDedicatedIn.

Definition

```
typedef enum {
    MEIMotorInputOVERTRAVEL_POS = MEIMotorDedicatedInLIMIT_HW_POS,
    MEIMotorInputOVERTRAVEL_NEG = MEIMotorDedicatedInLIMIT_HW_NEG,
    MEIMotorInputHOME           = MEIMotorDedicatedInHOME,
    MEIMotorInputAMP_FAULT      = MEIMotorDedicatedInAMP_FAULT,
    MEIMotorInputINDEX          = MEIMotorDedicatedInINDEX_PRIMARY,
} MEIMotorInput;
```

Description

MEIMotorInputOVERTRAVEL_POS	See MEIMotorDedicatedInLIMIT_HW_POS.
MEIMotorInputOVERTRAVEL_NEG	See MEIMotorDedicatedInLIMIT_HW_NEG.
MEIMotorInputHOME	See MEIMotorDedicatedInHOME.
MEIMotorInputAMP_FAULT	See MEIMotorDedicatedInAMP_FAULT.
MEIMotorInputINDEX	See MEIMotorDedicatedInINDEX_PRIMARY.

See Also

MEIMotorDedicatedIn

MEIMotorDedicatedOut (Deprecated)

NOTE:
MEIMotorDedicatedOut was moved to meiDeprecated.h in the 03.02.00 MPI software release.

Definition

```
typedef enum {
    MEIMotorDedicatedOutAMP_ENABLE     = MEIXmpMotorDedicatedFlagsMaskAMP_ENABLE,
/* bit 0 */
    MEIMotorDedicatedOutBRAKE_RELEASE  = MEIXmpMotorDedicatedFlagsMaskBRAKE_RELEASE,
/* bit 1 */
} MEIMotorDedicatedOut;
```

Description

MEIMotorDedicatedOut is an enumeration of bit masks for the motor's dedicated outputs. The support for dedicated outputs is node/drive specific. See the node/drive manufacturer's documentation for details.

MEIMotorDedicatedOutAMP_ENABLE	Enable/disable drive or amplifier. Drive is enabled when TRUE, disabled when FALSE.
MEIMotorDedicatedOutBRAKE_RELEASE	Enable/disable mechanical brake. Brake is released (motor shaft is free) when TRUE, engaged when FALSE.

See Also

mpiMotorIoGet | mpiMotorIoSet

MEIMotorDedicatedInOVERTRAVEL_POS (Deprecated)

NOTE:
MEIMotorDedicatedInOVERTRAVEL_POS was moved to meiDeprecated.h in the 20030421 MPI software release.

Definition

```
/* Backwards compatible dedicated I/O defines */
#define MEIMotorDedicatedInOVERTRAVEL_POS MEIMotorDedicatedInLIMIT_HW_POS
```

Description

MEIMotorDedicatedInOVERTRAVEL_POS is equivalent to MPIMotorDedicatedInLIMIT_HW_POS.

See MPIMotorDedicatedIn for a description.

See Also

MPIMotorDedicatedIn

MEIMotorDedicatedInOVERTRAVEL_NEG (Deprecated)

NOTE:
MEIMotorDedicatedInOVERTRAVEL_NEG was moved to meiDeprecated.h in the 20030421 MPI software release.

Definition

```
/* Backwards compatible dedicated I/O defines */
#define MEIMotorDedicatedInOVERTRAVEL_NEG MEIMotorDedicatedInLIMIT_HW_NEG
```

Description

MEIMotorDedicatedInOVERTRAVEL_NEG is equivalent to MPIMotorDedicatedInLIMIT_HW_NEG.

See MPIMotorDedicatedIn for a description.

See Also

MPIMotorDedicatedIn

MEIMotorGeneralIo (Deprecated)

NOTE:
MEIMotorGeneralIo was moved to meiDeprecated.h in the 03.02.00 MPI software release. See MPIMotorGeneralIo.

Definition

```
typedef enum MEIMotorGeneralIo {
  MEIMotorGeneralIoINVALID = -1
  MEIMotorGeneralIo0,
  MEIMotorGeneralIo1,
  MEIMotorGeneralIo2,
  MEIMotorGeneralIo3,
  MEIMotorGeneralIo4,
  MEIMotorGeneralIo5,
  MEIMotorGeneralIo6,
  MEIMotorGeneralIo7,
  MEIMotorGeneralIo8,
  MEIMotorGeneralIo9,
  MEIMotorGeneralIo10,
  MEIMotorGeneralIo11,
  MEIMotorGeneralIo12,
  MEIMotorGeneralIo13,
  MEIMotorGeneralIo14,
  MEIMotorGeneralIo15,

} MEIMotorGeneralIo;
```

Description

MEIMotorGeneralIo enumeration gives labels for each of the general purpose outputs that a motor can support.

See Also

MEIMotorIoMask (Deprecated)

NOTE:
MEIMotorIoMask was moved to meiDeprecated.h in the 03.02.00 MPI software release.

Definition

```
typedef enum {
    MEIMotorIoMask0  =  MEIXmpMotorIOMaskConfigurable0,  /* bit 16 */
    MEIMotorIoMask1  =  MEIXmpMotorIOMaskConfigurable1,  /* bit 17 */
    MEIMotorIoMask2  =  MEIXmpMotorIOMaskConfigurable2,  /* bit 18 */
    MEIMotorIoMask3  =  MEIXmpMotorIOMaskConfigurable3,  /* bit 19 */
    MEIMotorIoMask4  =  MEIXmpMotorIOMaskConfigurable4,  /* bit 20 */
    MEIMotorIoMask5  =  MEIXmpMotorIOMaskConfigurable5,  /* bit 21 */
    MEIMotorIoMask6  =  MEIXmpMotorIOMaskConfigurable6,  /* bit 22 */
    MEIMotorIoMask7  =  MEIXmpMotorIOMaskConfigurable7,  /* bit 23 */
    MEIMotorIoMask8  =  MEIXmpMotorIOMaskConfigurable8,  /* bit 24 */
    MEIMotorIoMask9  =  MEIXmpMotorIOMaskConfigurable9,  /* bit 25 */
    MEIMotorIoMask10 =  MEIXmpMotorIOMaskConfigurable10, /* bit 26 */
    MEIMotorIoMask11 =  MEIXmpMotorIOMaskConfigurable11, /* bit 27 */
    MEIMotorIoMask12 =  MEIXmpMotorIOMaskConfigurable12, /* bit 28 */
    MEIMotorIoMask13 =  MEIXmpMotorIOMaskConfigurable13, /* bit 29 */
    MEIMotorIoMask14 =  MEIXmpMotorIOMaskConfigurable14, /* bit 30 */
    MEIMotorIoMask15 =  MEIXmpMotorIOMaskConfigurable15, /* bit 31 */
} MEIMotorIoMask;
```

Description

MEIMotorIoMask is an enumeration of bit masks for the motor's configurable I/O. The support for configurable I/O is node/drive specific. See the node/drive manufacturer's documentation for details.

MEIMotorIoMask0	motor I/O mask for bit number 0
MEIMotorIoMask1	motor I/O mask for bit number 1
MEIMotorIoMask2	motor I/O mask for bit number 2
MEIMotorIoMask3	motor I/O mask for bit number 3
MEIMotorIoMask4	motor I/O mask for bit number 4
MEIMotorIoMask5	motor I/O mask for bit number 5
MEIMotorIoMask6	motor I/O mask for bit number 6

MEIMotorIoMask7	motor I/O mask for bit number 7
MEIMotorIoMask8	motor I/O mask for bit number 8
MEIMotorIoMask9	motor I/O mask for bit number 9
MEIMotorIoMask10	motor I/O mask for bit number 10
MEIMotorIoMask11	motor I/O mask for bit number 11
MEIMotorIoMask12	motor I/O mask for bit number 12
MEIMotorIoMask13	motor I/O mask for bit number 13
MEIMotorIoMask14	motor I/O mask for bit number 14
MEIMotorIoMask15	motor I/O mask for bit number 15

See Also

mpiMotorIoGet | mpiMotorIoSet | MEIMotorIoType

MEIFilterDataType (Deprecated)

NOTE:
MEIFilterDataType was moved to meiDeprecated.h in the 20030516 MPI software release.

Definition

```
/*
 * MEIFilterDataType changed to simply MEIDataType
 * so that other MPI methods can use the data type enum
 */
#define MEIFilterDataTypeINVALID        (MEIDataTypeINVALID)
#define MEIFilterDataTypeCHAR           (MEIDataTypeCHAR)
#define MEIFilterDataTypeSHORT          (MEIDataTypeSHORT)
#define MEIFilterDataTypeUSHORT         (MEIDataTypeUSHORT)
#define MEIFilterDataTypeLONG           (MEIDataTypeLONG)
#define MEIFilterDataTypeULONG          (MEIDataTypeULONG)
#define MEIFilterDataTypeFLOAT          (MEIDataTypeFLOAT)
#define MEIFilterDataTypeDOUBLE         (MEIDataTypeDOUBLE)
#define MEIFilterDataTypeLAST           (MEIDataTypeLAST)
#define MEIFilterDataTypeFIRST          (MEIDataTypeFIRST)
```

Description

MEIFilterDataType is an enumeration of data types for the filter coefficients.

MEIFilterDataTypeCHAR	character filter data type
MEIFilterDataTypeSHORT	short integer filter data type
MEIFilterDataTypeUSHORT	unsigned short integer filter data type
MEIFilterDataTypeLONG	long integer filter data type
MEIFilterDataTypeULONG	unsigned long filter data type
MEIFilterDataTypeFLOAT	floating point filter data type
MEIFilterDataTypeDOUBLE	double precision floating point filter data type

See Also

MEIDataType

MEISynqNetMessage (Deprecated)

NOTE:
MEISynqNetMessage was moved to meiDeprecated.h in the 20030715 MPI software release.

Definition

```
#define MEISynqNetMessageNODE_UNAVAILABLE (MEISqNodeMessageNODE_INVALID)
#define MEISynqNetMessageRESPONSE_TIMEOUT (MEISqNodeMessageRESPONSE_TIMEOUT)
#define MEISynqNetMessageREADY_TIMEOUT    (MEISqNodeMessageREADY_TIMEOUT)
#define MEISynqNetMessageSRVC_ERROR       (MEISqNodeMessageSRVC_ERROR)
#define MEISynqNetMessageSRVC_UNSUPPORTED (MEISqNodeMessageSRVC_UNSUPPORTED)
```

Description

MEISynqNetMessage is an enumeration of SynqNet error messages that can be returned by the MPI library.

MEISynqNetMessageNODE_UNAVAILABLE	The node number is not available on the network. This message code is returned by MPI methods that fail a service command transaction due to the specified node number is greater than or equal to the total number of nodes discovered during network initialization. To correct this problem, check the discovered node count with meiSynqNetInfo(…). If the node count is not what you expected check your network wiring, node condition, and re-initialize the network with mpiControlReset(…).
MEISynqNetMessageRESPONSE_TIMEOUT	The node failed to respond to a service command within the timeout. This message code is returned by MPI methods that fail a service command transaction because the node failed to respond within the allotted amount of time. To correct this problem, check your node hardware. There are 32 possible message codes for this error. Each message code specifies a different node, from node number 0 to 31.

MEISynqNetMessageREADY_TIMEOUT	The node failed to be ready for a service command within the timeout. This message code is returned by MPI methods that fail a service command transaction because the node is not ready to accept service commands. To correct this problem, check your node hardware. There are 32 possible message codes for this error. Each message code specifies a different node, from node number 0 to 31.
MEISynqNetMessageSRVC_ERROR	The service command failed. This message code is returned by MPI methods that fail a service command transaction. To correct this problem, check your node hardware. There are 32 possible message codes for this error. Each message code specifies a different node, from node number 0 to 31.
MEISynqNetMessageSRVC_UNSUPPORTED	The node does not support the service command. This message code is returned by MPI methods that fail a service command transaction because the node does not support it.

See Also

meiSynqNetInfo | mpiControlReset

MEISynqNetMaxMotorENCODER_COUNT (Deprecated)

NOTE:
MEISynqNetMaxMotorENCODER_COUNT was moved to meiDeprecated.h in the 20030722 MPI software release.

Definition

```
#define MEISynqNetMaxMotorENCODER_COUNT (MEIXmpMotorEncoders) /*  2 */
```

Description

MEISynqNetMaxMotorENCODER_COUNT defines the maximum number of encoder resources per motor.

NOTE: The encoder count may be further limited by the available resources on the node.

This define should be used instead of the MEIXmpMotorEncoders definition in xmp.h. It is recommended that applications avoid programming to defines or structures in xmp.h.

See Also