Idn Objects

Introduction

An **Idn** object represents an individual SERCOS IDN. An Idn object is a container for data that is sent to and received from a Node. Methods are provided to get and set the data associated with an Idn object, but these methods do not actually transmit an Idn object to or from a Node.

Methods

Create, Delete, Validate Methods

mpiIdn**Create**	Create Idn object
mpiIdn**Delete**	Delete Idn object
mpiIdn**Validate**	Validate Idn object

Configuration and Information Methods

mpiIdn**Copy**	Copy all element fields of one IDN object to another.
mpiIdn**FieldGet**	Get the value of an IDN (or all fields) from the drive.
mpiIdn**FieldSet**	Set the value of an IDN field. The inverse of mpiIdnFieldGet.
mpiIdn**NumberName**	Fill the buffer with the text form of the IDN number.
mpiIdn**NumberText**	Get the ASCII representation of an Idn objects IDNumber (S-X-XXXX or P-X-XXXX)

Data Types

MPIIdn**Data**
MPIIdn**DataType**
MPIIdn**Element**
MPIIdn**Field**
MPIIdn**Message**
MPIIdn**Number**
MPIIdn**Procedure**
MPIIdn**Value**
MPIIdn**VarLength**

Macros

mpiIdn**DataGET**	Get data type (Element 3) and operation data (Element 7)
mpiIdn**DataSET**	
mpiIdn**ElementGET**	
mpiIdn**FieldATTR**	
mpiIdn**FieldAttrDATA_TYPE**	
mpiIdn**FieldAttrGET**	
mpiIdn**FieldAttrMASK**	
mpiIdn**FieldAttrSET**	
mpiIdn**FieldAttrSHIFT**	
mpiIdn**NUMBER**	
mpiIdn**NumberDATA_BLOCK**	
mpiIdn**NumberGET**	
mpiIdn**NumberPARAMETER_SET**	

mpiIdn**NumberPRODUCT**
mpiIdn**NumberSET**
mpiIdn**NumberSTANDARD**
mpiIdn**NumberTYPE**

Constants

MPIIdn**DATA_SIZE_MAX**

MPIIdn**NAME_SIZE_MAX**

MPIIdn**NUMBER_SIZE_MAX**

MPIIdn**UNIT_SIZE_MAX**

Copyright © 2002
Motion Engineering

mpiIdnCreate

Declaration

```
const MPIIdn mpiIdnCreate(MPIIdnNumber number)
```

Required Header stdmpi.h

Description

IdnCreate creates an Idn object, where ***number*** argument is the initial SERCOS IDNumber to be assigned to this object.
IdnCreate is the equivalent of a C++ constructor.

Return Values

handle	to an Idn object
MPIHandleVOID	if the object could not be created

See Also mpiIdnDelete | mpiIdnValidate

mpiIdnDelete

Declaration `long mpiIdnDelete(MPIIdn idn)`

Required Header stdmpi.h

Description **IdnDelete** deletes an Idn object and invalidates its handle (***idn***).
IdnDelete is the equivalent of a C++ destructor.

Return Values	
MPIMessageOK	if *IdnDelete* successfully deletes the Idn object and invalidates its handle

See Also mpiIdnCreate | mpiIdnValidate

mpiIdnValidate

Declaration `long mpiIdnValidate(MPIIdn idn)`

Required Header stdmpi.h

Description **IdnValidate** validates the Idn object and its handle (***idn***).

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

See Also mpiIdnCreate | mpiIdnDelete

mpiIdnCopy

Declaration

```
long mpiIdnCopy(MPIIdn idn,
                MPIIdn dst)
```

Required Header

stdmpi.h

Description

IdnCopy copies all element fields of one IDN object to another.

Return Values

MPIMessageOK	if *IdnCopy* successfully copies all element fields to on IDN object to another

See Also

mpiIdnFieldGet

Declaration

```
long mpiIdnFieldGet(MPIIdn      idn,
                    MPIIdnField field,
                    void        *value)
```

Required Header

stdmpi.h

Description

IdnFieldGet gets the value of an IDN (or all fields) from the drive. The pointer "value" must point to enough memory to hold the value of the requested field. See MPIIdnElement for the type of each field.

Return Values

MPIMessageOK	if successful

See Also

MPIIdnElement | mpiIdnFieldSet | MPIIdnField

mpiIdnFieldSet

Declaration

```
long mpiIdnFieldSet(MPIIdn      idn,
                    MPIIdnField field,
                    void        *value)
```

Required Header

stdmpi.h

Description

IdnFieldSet sets the value of an IDN field. The inverse of mpiIdnFieldGet.

Return Values	
MPIMessageOK	if successful

See Also

mpiIdnFieldGet | MPIIdnField

mpiIdnNumberName

Declaration

```
char * mpiIdnNumberName(MPIIdnNumber number,
                        char *buffer)
```

Required Header stdmpi.h

Description Given a pointer to a char buffer of length MPIIdnNUMBER_SIZE_MAX + 1, **IdnNumberName** fills the buffer with the text form of the IDN number.

Return Values	
MPIMessageOK	if successful

See Also

mpiIdnNumberText

Declaration

```
long mpiIdnNumberText(MPIIdn      idn,
                      const char **text)
```

Required Header stdmpi.h

Description **IdnNumberText** sets “text” to point to the internal memory storing the text form of the IDN number.

Return Values	
MPIMessageOK	if successful

See Also

MPIIdnData

MPIIdnData

```
typedef union {
    unsigned long       binary;
    MPIIdnNumber        idn;
    MPIIdnProcedure     procedure;
    char                *text;
    MPIIdnValue         value;
    MPIIdnVarLength     varLength;
} MPIIdnData;
```

Description

binary	0 or 1
idn	an IDN number
procedure	procedure IDN's data is a word that is a bit mask.
***text**	a char pointer that points to a text string.
value	a long or unsigned long.
valLength	IDN data whose length is not defined, and it's data can be of any number of types.

See Also

MPIIdnDATA_SIZE_MAX | MPIIdnDataType

MPIIdnDataType

MPIIdnDataType

```
typedef enum {
    MPIIdnDataTypeINVALID,
        /*
         * These data types correspond to the
         * SERCOS attribute TYPE field.
         * They should not be changed,
         * though new types may be appended
         */

    MPIIdnDataTypeBINARY,       /* 0x0 */
    MPIIdnDataTypeUNSIGNED,     /* 0x1 */
    MPIIdnDataTypeLONG,         /* 0x2 */
    MPIIdnDataTypeUNSIGNED_HEX, /* 0x4 */
    MPIIdnDataTypeTEXT,         /* 0x5 */
    MPIIdnDataTypeIDN,          /* 0x6 */

} MPIIdnDataType;
```

Description

IdnDataType corresponds to the SERCOS attribute TYPE field. It specifies the type of data that the IDN contains.

See Also

MPIIdnDATA_SIZE_MAX | MPIIdnData

MPIIdnElement

MPIIdnElement

```
typedef struct MPIIdnElement {
    MPIIdnNumber        number;         /* 0    MPIIdnFieldNUMBER       */
    const char          *name;          /* 1    MPIIdnFieldNAME         */
    unsigned long       attributes;     /* 2    MPIIdnFieldATTRIBUTES   */
    const char          *unit;          /* 3    MPIIdnFieldUNIT         */
    MPIIdnValue         minimum;        /* 4    MPIIdnFieldMINIMUM      */
    MPIIdnValue         maximum;        /* 5    MPIIdnFieldMAXIMUM      */
    MPIIdnData          data;           /* 6    MPIIdnFieldDATA         */
} MPIIdnElement;
```

Description

number	The IDN number field.
***name**	A pointer to the IDN name field.
attributes	The IDN attributes field.
***unit**	A pointer to the IDN unit field.
minimum	The IDN minimum field which is the smallest value the IDN data can be (not valid with certain data types).
maximum	The IDN maximum field which is the largest value the IDN data can be (not valid with certain data types).
data	The IDN data.

See Also

mpiIdnElementGET

MPIIdnField

MPIIdnField

```
typedef enum {
    MPIIdnFieldINVALID,

    MPIIdnFieldNUMBER,
    MPIIdnFieldNAME,
    MPIIdnFieldATTRIBUTES,
    MPIIdnFieldUNIT,
    MPIIdnFieldMINIMUM,
    MPIIdnFieldMAXIMUM,
    MPIIdnFieldDATA,

    MPIIdnFieldALL,
} MPIIdnField;
```

Description The **IdnField** enumeration is a list of all the different fields that comprises an IDN.

See Also mpiIdnFieldGet | mpiIdnFieldSet

MPIIdnMessage

MPIIdnMessage

```
typedef enum {

    MPIIdnMessageIDN_INVALID,
    MPIIdnMessageATTRIBUTES_INVALID,
} MPIIdnMessage;
```

Description

The **IdnMessage** is an enumeration that contains possible return values for IDN method calls.

See Also

MPIIdnNumber

MPIIdnNumber

```
typedef enum {
    MPIIdnNumberSTANDARD,
    MPIIdnNumberPRODUCT,

    MPIIdnNumberINVALID,

} MPIIdnNumber;
```

Description

IdnNumber is an IDN number, and also is used to specify whether the IDN is a standard or proprietary IDN.

See Also

MPIIdnNUMBER_SIZE_MAX

MPIIdnProcedure

MPIIdnProcedure

```
typedef enum {
    MPIIdnProcedureSET,
    MPIIdnProcedureENABLED,

    MPIIdnProcedureCANCEL,
    MPIIdnProcedureEXECUTE,

    MPIIdnProcedureRUNNING,
    MPIIdnProcedureERROR,
    MPIIdnProcedureDATA_VALID,
} MPIIdnProcedure;
```

Description

The data portion of a Procedure IDN is a bit map that represents the state of the procedure, this enumeration is a bit mask and can be used to decipher the status of the procedure.

See Also

MPIIdnValue

MPIIdnValue

```
typedef union {
    long                l;
    unsigned long       u;
} MPIIdnValue;
```

Description

IdnValue represents either an unsigned long or long.

See Also

MEIIdnVarLength

MEIIdnVarLength

```
typedef struct MPIIdnVarLength{
    long                    count;

    MPIIdnDataType          type;

    union {
        MPIIdnNumber        *idn;
        char                *text;
        MPIIdnValue         *value;
        } as;
} MPIIdnVarLength;
```

Description

count	The size of the data buffer pointed to by **as**.
type	The type of data in the buffer.
as	A union of different pointer types, where the data is located.

See Also

mpiIdnDataGET

Declaration

```
#define mpiIdnDataGET(idn,idnData) \
        mpiIdnFieldGet((idn), MPIIdnFieldDATA, (idnData))
```

Required Header

stdmpi.h

Description

IdnDataGET reads the MPIIdnFieldDATA from the specified ***idn*** and writes it to the location pointed to by ***idnData***.

See Also

mpiIdnDataSET

mpiIdnDataSET

Declaration

```
#define mpiIdnDataSET(idn,idnData) \
        mpiIdnFieldSet((idn), MPIIdnFieldDATA, (idnData))
```

Required Header

stdmpi.h

Description

IdnDataSET writes the MPIIdnFieldDATA to the specified ***idn***, from the location pointed to by ***idnData***.

See Also

mpiIdnDataGET

mpiIdnElementGET

Declaration

```
#define mpiIdnElementGET(idn,idnElement) \
        mpiIdnFieldGet((idn), MPIIdnFieldALL, (idnElement))
```

Required Header

stdmpi.h

Description

IdnElementGET reads the MPIIdnFieldALL from the specified ***idn*** and writes it to the location pointed to by ***idnElement***.

See Also

MPIIdnElement

mpiIdnFieldATTR

Declaration

```
#define mpiIdnFieldATTR(mask,shift)
        (((mask) << 16) | (shift))
```

Required Header

stdmpi.h

Description

IdnFieldATTR ***shifts*** the ***mask*** up by 16 bits and bitwise ORs in the shift value.

See Also

mpiIdnFieldAttrGET | mpiIdnFieldAttrSET

mpiIdnFieldAttrDATA_TYPE

Declaration

```
#define mpiIdnFieldAttrDATA_TYPE(attr)
        ((MPIIdnDataType)mpiIdnFieldAttrGET  MPIIdnField AttrTYPE, (attr)))
```

Required Header

stdmpi.h

Description

IdnFieldAttrDATA_TYPE reads the ***MPIIdnDataType*** from the specified ***attr***.

See Also

mpiIdnFieldAttrGET

Declaration

```
#define mpiIdnFieldAttrGET(attr,word)
        (((word) >> iIdnFieldAttrSHIFT(attr)) &
        mpiIdnFieldAttrMASK(attr))
```

Required Header

stdmpi.h

Description

IdnFieldAttrGET reads the value of the specified ***attr*** (MPIIdnFieldAttr) from the ***word***.

See Also

mpiIdnFieldATTR

mpiIdnFieldAttrMASK

Declaration

```
#define     mpiIdnFieldAttrMASK(attr)
            ((( attr) >> 16) & 0xFFFF)
```

Required Header

stdmpi.h

Description

IdnFieldAttrMASK masks the value of the specified ***attr*** (MPIIdnFieldAttr). Used internally by mpiIdnFieldAttrGET/SET(…).

See Also

mpiIdnFieldAttrGET | mpiIdnFieldAttrSET

mpiIdnFieldAttrSET

Declaration

```
#define mpiIdnFieldAttrSET(attr,bits)
        (((bits) & mpiIdnFieldAttrMASK(attr)) <<
        mpiIdnFieldAttrSHIFT(attr))
```

Required Header

stdmpi.h

Description

IdnFieldAttrSET sets the value of the specified ***attr*** (MPIIdnFieldAttr) to ***bits***.

See Also

mpiIdnFieldAttrGET | mpiIdnFieldATTR

mpiIdnFieldAttrSHIFT

Declaration

```
#define mpiIdnFieldAttrSHIFT(attr)
        ((attr) & 0xFFFF)
```

Required Header

stdmpi.h

Description

IdnFieldAttrSHIFT shifts the value of the specified ***attr*** (MPIIdnFieldAttr). Used internally by mpiIdnFieldAttrGET/SET(…).

See Also

mpiIdnFieldAttrGET | mpiIdnFieldAttrSET

mpiIdnNUMBER

Declaration

```
#define mpiIdnNUMBER(type,param,data) \
((((type) & 0x1) << 15) | (((param) & 0x7) << 12) | ((data) & 0xFFF))
```

Required Header

stdmpi.h

Description

IdnNUMBER converts the idn type, param, and data into a number. Used internally by mpiIdnNumberPRODUCT/STANDARD(…).

See Also

mpiIdnNumberPRODUCT | mpiIdnNumberSTANDARD

mpiIdnNumberDATA_BLOCK

Declaration

```
#define     mpiIdnNumberDATA_BLOCK(number)
            ((number) & 0xFFF)
```

Required Header

stdmpi.h

Description

IdnNumberDATA_BLOCK determines the idn data block from a SERCOS identification ***number***.

See Also

mpiIdnNumberGET

Declaration

```
#define     mpiIdnNumberGET(idn,idnNumber) \
            mpiIdnFieldGet((idn), MPIIdnFieldNUMBER, (idnNumber))
```

Required Header

stdmpi.h

Description

IdnNumberGET gets the SERCOS identification number from the ***idn*** and writes it into the location pointed to by ***idnNumber***.

See Also

mpiIdnNumberSET

mpiIdnNumberPARAMETER_SET

Declaration

```
#define    mpiIdnNumberPARAMETER_SET(number)
           (((number) >> 12) & 0x7)
```

Required Header

stdmpi.h

Description

IdnNumberPARAMETER_SET determines the idn parameters from a SERCOS identification ***number***.

See Also

mpiIdnNumberPRODUCT

Declaration

```
#define  mpiIdnNumberPRODUCT(data)
         mpiIdnNUMBER(MPIIdnNumberPRODUCT, 0, (data))
```

Required Header

stdmpi.h

Description

IdnNumberPRODUCT converts an idn from a product specific value to the SERCOS identification number format.

See Also

mpiIdnNumberSET

Declaration

```
#define     mpiIdnNumberSET(idn,idnNumber) \
            mpiIdnFieldSet((idn), MPIIdnFieldNUMBER, (idnNumber))
```

Required Header

stdmpi.h

Description

IdnNumberSET sets the SERCOS identification number in the ***idn***, to the value from ***idnNumber***.

See Also

mpiIdnNumberGET

mpiIdnNumberSTANDARD

Declaration

```
#define  mpiIdnNumberSTANDARD(data)
         mpiIdnNUMBER(MPIIdnNumberSTANDARD, 0, (data))
```

Required Header stdmpi.h

Description **IdnNumberSTANDARD** converts an idn from a standard value to the SERCOS identification number format.

See Also

mpiIdnNumberTYPE

Declaration

```
#define mpiIdnNumberTYPE(number)
        (((number) >> 15) & 0x1)
```

Required Header

stdmpi.h

Description

IdnNumberTYPE determines the idn type (STANDARD or PRODUCT) from a SERCOS identification ***number***.

See Also

MPIIdnNumber

MPIIdnDATA_SIZE_MAX

MPIIdnDATA_SIZE_MAX

```
#define   MPIIdnDATA_SIZE_MAX   (65532)
```

Description

IdnDATA_SIZE_MAX defines the maximum data size an Idn object supports.

See Also

mpiIdnDataGET | MPIIdnData

MPIIdnNAME_SIZE_MAX

MPIIdnNAME_SIZE_MAX

```
#define MPIIdnNAME_SIZE_MAX     (60)
```

Description

IdnNAME_SIZE_MAX defines the maximum number of characters that can be used to represent the name of an IDN.

See Also

MPIIdnNUMBER_SIZE_MAX

MPIIdnNUMBER_SIZE_MAX

```
#define MPIIdnNUMBER_SIZE_MAX   (8)
```

Description

IdnNUMBER_SIZE_MAX defines the maximum number of characters that can be used in an IDN number string.

See Also

MPIIdnNumber

MPIIdnUNIT_SIZE_MAX

MPIIdnUNIT_SIZE_MAX

```
#define  MPIIdnUNIT_SIZE_MAX    (12)
```

Description

IdnUNIT_SIZE_MAX defines the maximum number of characters that can be used to represent the units used by an IDN.

See Also