MSX-E370x soap api functions

Generated by Doxygen 1.7.1

Wed May 4 2016 14:10:01

Contents

Chapter 1

Introduction

MainRevision:

1.1 Introduction

This documentation describes the SOAP functions and gives software hints to work with the MSX-E systems. Following documentations can be found under **Modules**.

SOAP means Simple Object Access Protocol. This protocol enables to use the MSX-E software functions over Ethernet. It is providing **Web Services** that can easily be consumed in many programming languages like C, C++, C#, VB.Net... With the SOAP functions, all functionalities of the MSX-E system can be managed / configured / monitored.

1.2 Remark: SOAP functions prototypes

In some programming languages, SOAP functions names and parameters could be different as those described in this documentation. Please see to Software hints

Chapter 2

Module Documentation

2.1 MX370x functions

Modules

- MX370x Get informations functions
- MX370x Auto refresh functions

  In the auto refresh mode the measurement value is updated automatically after each acquisition.

- MX370x Sequence functions

  A sequence is a list of channels (max 16) that are acquired.

- MX370x Transducer functions
- MX370x Min/Max acquisition functions

  In the Min/Max-mode an acquisition of certain channels is executed (adjustable by a mask) and the Min-/Max values of each channel are saved.

- MX370x diagnostic functions

  The module MSX-E370x disposes of a diagnostic function which, under certain circumstances, can detect a short circuit or line break on the primary circuit as well as on the secondary circuit.

- MX370x calibration functions

  The offset and amplitude error of the MSX-E370x is corrected through digital potentiometers.

- MX370x transducer database management functions

  These functions allows to manage the database of transducer types on the module.

- MX370x External Digital I/O functions

  Contain the digital I/O configuration, filter, diagnostic and read/write functions for the external digital I/O.

2.2 Software hints

Modules

- SOAP function calls in C/C++ language

Some hints on how to use the SOAP functions in a C/C++ program.

- MSX-E systems servers

  The MSX-E embeds servers to provide configuration, management and monitoring over Ethernet.

2.3 SOAP function calls in C/C++ language

Some hints on how to use the SOAP functions in a C/C++ program.

2.3.1 C/C++ language SOAP function prototypes

In this documentation, functions are described as follows.

```
  int MXCommon__GetModuleType(void * _, struct MXCommon__ByteArrayResponse
* Response);
```

Two main differences can be remarked in the function prototypes:

- The prefix:

  ```
  soap_call_
  ```

- The first three parameters of the SOAP stub:

  ```
  struct soap *soap
  const char *soap_endpoint
  const char *soap_action
  ```

The C function prototype has the following syntax:

```
int soap_call_MXCommon__GetModuleType(struct soap *soap, const char *soap_endpoin
    t, const char *soap_action, void *_, struct MXCommon__ByteArrayResponse *Response
    );
```

Functions to call in C/C++ are called **stubs** and can be found in the **MSXEXXXXStub.h** file.

2.3.2 Rules and use of gSOAP calls in C/C++

When programming with gSOAP in C/C++ language, some rules have to be followed to avoid memory leaks, slow socket disconnections and multi-threading compliancy.

Note: Software code pieces in this chapter comes from SOAP calls sample

- **Opening and initializing an SOAP connection (using the gSOAP functions)**

  ```
  struct soap *soapContext;

  // IP address of the MSX-E and after the ':' is the MSX-E SOAP server port number

  char soap_endpoint[] = IP_ADDRESS":5555";
  ```

```
// Allocates and initialize a new runtime context
if ((soapContext = soap_new ()) == NULL)
        return EXIT_FAILURE;

// Initializes the SOAP context with options
soap_init2 (soapContext, SOAP_IO_KEEPALIVE, SOAP_IO_KEEPALIVE);

// Sets timeouts in seconds
soapContext->send_timeout = 1;
soapContext->recv_timeout = 1;
soapContext->accept_timeout = 1;
```

Remark: A new runtime context is required in each new thread!

- **Calling the MSX-E SOAP functions**

  Important

  - In C/C++, each MSX-E SOAP function call is allocating memory (to manage deserialized data) and is not deallocating it automatically. The allocated memory has to be deallocated explicitly by calling, in this order, the soap_destroy and soap_end functions. These functions can be called after each MSX-E SOAP function call or after several calls. It is important to call these functions regularly to prevents memory leaks.
  - If the SOAP function is returning pointer, call the soap_destroy and soap_end functions only after working with the pointers. If soap_destroy and soap_end are called before working the pointers, the values pointed by the pointer will not be available. All dynamically allocated values are destroyed by soap_destroy and soap_end.

  See C/C++ language SOAP function prototypes to call the MSX-E SOAP functions.

```
for ( ; ; )
{
        soap_error = soap_call_MXCommon__GetModuleTemperatureValueAndStatus (soap
      Context, soap_endpoint, NULL, option, &tempResponse);

        ...

        // As gSOAP doesn't released automatically the memory that it allocates t
      o
        // deserialize SOAP data we have to released it explicitly.
        // There is no need to call the function in each loop cycle,
        // it depends on the application, and desired performance and
        // available memory.
        soap_destroy (soapContext);
        soap_end (soapContext);

        ...
}
```

- **Error management**

  There are 3 types of errors that can be generated by the MSX-E SOAP functions:

  - SOAP errors: It is the SOAP function return value. More details about the error can be obtained by using the soap_faultstring function.

    ```
    soap_error = soap_call_MXCommon__GetModuleTemperatureValueAndStatus (soapContext,
           soap_endpoint, NULL, option, &tempResponse);

    if (soap_error)
            printf ("message: %s\n", *soap_faultstring (soapContext));
    ```

    Remark: soap_faultstring has to be called just after the SOAP function call that generates the SOAP error and before the call of soap_destroy and soap_end.

– The MSX-E error (iReturnValue): Returns errors that are not linux specific. The iReturnValue variable is located in the response structure. These errors are described in this documentation.

```
soap_error = soap_call_MXCommon__GetModuleTemperatureValueAndStatus (soapContext,
       soap_endpoint, NULL, option, &tempResponse);

// Check for errors, if there are, stop the loop
if (checkErrors (soapContext, soap_endpoint, NULL, soap_error, tempResponse.sResp
      onse.iReturnValue, tempResponse.sResponse.syserrno))
        break;
```

– The MSX-E system error (syserrno): Returns linux specific errors. This variable has to be checked only if iReturnValue = -1 or iReturnValue <= -100. It returns the linux errno error. More details about the error can be obtained by using the soap_call_MXCommon__Strerror function.

```
struct MXCommon__ByteArrayResponse byteArrayResponse;

memset (&byteArrayResponse, 0, sizeof (struct MXCommon__ByteArrayResponse));

soap_error = soap_call_MXCommon__GetModuleTemperatureValueAndStatus (soapContext,
       soap_endpoint, NULL, option, &tempResponse);

if ((tempResponse.sResponse.iReturnValue == -1) || ((tempResponse.sResponse.iRetu
      rnValue <= -100) && tempResponse.sResponse.syserrno))
{
        soap_call_MXCommon__Strerror (soapContext, soap_endpoint, soap_action, te
      mpResponse.sResponse.syserrno, &byteArrayResponse);

        // Display the system error
        printf ("\nSystem error: %s\n", byteArrayResponse.sArray.__ptr);
}
```

- **Close the SOAP connection**

  Before to quit the application or when no MSX-E SOAP function calls are necessary, the SOAP connection has to be closed and the context has to be released. Call following functions in this order: soap_destroy, soap_end, soap_free.

```
// Release SOAP
soap_destroy (soapContext);
soap_end (soapContext);

// Close the socket an release the SOAP context
soap_free (soapContext);
```

2.3.3 SOAP calls sample

Here a sample code, and it's makefile, to read the MSX-E internal temperature.

To compile it, use *make* ***IP_ADDRESS=****YOUR_MSX-E_IP_ADDRESS* don't forget to set the **INTERFACE_COMMON_LIBRARY_DIR** to point on the MSX-E Common Interface_Library directory.

Remark: Don't use blank spaces in the directories and file names.

temperature.c file

```
#include <unistd.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <errno.h>
```

```
#include <MXCommon.nsmap> // this file must be included only once in the project
#include <assert.h>
#include <sys/stat.h>
#include <termios.h>

//------------------------------------------------------------------------------
      ----------
//------------------------------------------------------------------------------
      ----------

/** kbhit for linux.

 @retval 0: No key pressed.
 @retval <>0: Key pressed.
 */
int kbhit (void)
{
        struct termios oldt, newt;
        struct timeval tv;
        fd_set read_fd;
        int status;

        tcgetattr (STDIN_FILENO, &oldt);
        memcpy (&newt, &oldt, sizeof (newt));
        newt.c_lflag &= ~(ICANON | ECHO);
        tcsetattr (STDIN_FILENO, TCSANOW, &newt);

        tv.tv_sec  = 0;
        tv.tv_usec = 0;
        FD_ZERO (&read_fd);
        FD_SET (0, &read_fd);

        status = select (1, &read_fd, NULL, NULL, &tv);
        tcsetattr (STDIN_FILENO, TCSANOW, &oldt);

        if (status < 0)
                return 0;
        else
                return (status);
}

//------------------------------------------------------------------------------
      ----------

/** getch for linux.

 @retval key: Key number.
 */
int getch (void)
 {
        struct termios oldt, newt;
        int ch;

        tcgetattr (STDIN_FILENO, &oldt);
        memcpy (&newt, &oldt, sizeof (newt));
        newt.c_lflag &= ~(ICANON | ECHO);
        tcsetattr (STDIN_FILENO, TCSANOW, &newt);
        ch = getchar();
        tcsetattr (STDIN_FILENO, TCSANOW, &oldt);

        return (ch);
}

//------------------------------------------------------------------------------
      ----------
//------------------------------------------------------------------------------
      ----------
```

```
/** Check if the SOAP call returns an error, display it.

 @param[in] soapContext : SOAP context structure.
 @param[in] soap_endpoint : IP and port number of the system to access.
 @param[in] soap_action : SOAP action required by the Web service.
 @param[in] soap_error: The SOAP function return value.
 @param[in] msxe_error: The MSX-E system return value contained in the response s
      tructure (iReturnValue).
 @param[in] msxe_syserrno: The MSX-E system errno value contained in the response
       structure (syserrno).

 @retval 0: No error.
 @retval 1: Error.
 */
int checkErrors (struct soap *soapContext, const char *soap_endpoint, const char
      *soap_action, int soap_error, int msxe_error, int msxe_syserrno)
{
        if ((soap_error != 0) || (msxe_error != 0))
        {
                printf ("MSX-E function response error: %d\n", msxe_error);
                printf ("soap error: %d ", soap_error);

                // In case of an SOAP error, display it
                if (soap_error)
                        printf ("message: %s\n", *soap_faultstring (soapContext))
      ;
                else
                {
                        // It's not an SOAP error but a system error
                        if ((msxe_error == -1) || ((msxe_error <= -100) && msxe_s
      yserrno))
                        {
                                struct MXCommon__ByteArrayResponse byteArrayRespo
      nse;
                                memset (&byteArrayResponse, 0, sizeof (struct
      MXCommon__ByteArrayResponse));

                                // Get the system error
                                if (soap_call_MXCommon__Strerror (soapContext, so
      ap_endpoint, soap_action, msxe_syserrno, &byteArrayResponse))
                                        printf ("\nsoap_call_MXCommon__Strerror e
      rror: %s\n", *soap_faultstring (soapContext));
                                else // Display the system error
                                        printf ("\nSystem error: %s\n", byteArray
      Response.sArray.__ptr);
                        }
                }

                return 1;
        }

        return 0;
}

//--------------------------------------------------------------------------------
      ----------
//--------------------------------------------------------------------------------
      ----------

int main (void)
{
         struct soap *soapContext;
         unsigned long option = 0;
         struct MXCommon__GetModuleTemperatureValueAndStatusResponse tempResponse
      ;
         int soap_error = 0;
```

```
   char *tempStatus[] = {"NOT AVAILABLE", "TOO LOW", "LOW", "NOMINAL", "HIG
H", "TOO HIGH"};
   // IP address of the MSX-E and after the ':' is the MSX-E SOAP server po
rt number
   char soap_endpoint[] = IP_ADDRESS":5555";

   memset (&tempResponse, 0, sizeof (struct
MXCommon__GetModuleTemperatureValueAndStatusResponse));

   // Allocates and initialize a new runtime context
   if ((soapContext = soap_new ()) == NULL)
      return EXIT_FAILURE;

   // Initializes the SOAP context with options
   soap_init2 (soapContext, SOAP_IO_KEEPALIVE, SOAP_IO_KEEPALIVE);

   // Sets timeouts in seconds
   soapContext->send_timeout = 1;
   soapContext->recv_timeout = 1;
   soapContext->accept_timeout = 1;

   for ( ; ; )
   {
           soap_error = soap_call_MXCommon__GetModuleTemperatureValueAndSta
tus (soapContext, soap_endpoint, NULL, option, &tempResponse);

           // Check for errors, if there are, stop the loop
           if (checkErrors (soapContext, soap_endpoint, NULL, soap_error, t
empResponse.sResponse.iReturnValue, tempResponse.sResponse.syserrno))
                   break;

          // There is no error
          printf ("MSX-E Temperature: %.2lf °C status: ", tempResponse.dTem
peratureValue);

          if (tempResponse.ulTemperatureStatus < 6)
                  printf ("%s ", tempStatus[tempResponse.ulTemperatureStatu
s]);
          else
                  printf ("unknown ");

          printf ("(ESC to quit)\n");

           // As gSOAP doesn't released automatically the memory that it al
locates to
           // deserialize SOAP data we have to released it explicitly.
           // There is no need to call the function in each loop cycle,
           // it depends on the application, and desired performance and
           // available memory.
           soap_destroy (soapContext);
           soap_end (soapContext);

           // Listen on keyboard actions
           if (kbhit ())
                   if (getch () == 27) // Check if ESC key has been used
                           break;
  }

   // Release SOAP
   soap_destroy (soapContext);
   soap_end (soapContext);
   // Close the socket an release the SOAP context
   soap_free (soapContext);

   return EXIT_SUCCESS;
}
```

Makefile

```
TOPDIR                          :=$(shell pwd)

CC                                      :=$(CROSS)$(CC)
LD                                      :=$(CROSS)ld
STRIP                           :=$(CROSS)strip

ifneq ($(INTERFACE_COMMON_LIBRARY_DIR),"")
INTERFACE_COMMON_LIBRARY_DIR    =$(TOPDIR)/../../Interface_Library
endif

# The MSX-E IP address
ifneq ($(IP_ADDRESS),"")
IP_ADDRESS                                              =192.168.99.99
endif

# File implementing SOAP client
SOAPSOURCES                     := $(INTERFACE_COMMON_LIBRARY_DIR)/MSXEClient.o $
      (INTERFACE_COMMON_LIBRARY_DIR)/MSXEC.o $(INTERFACE_COMMON_LIBRARY_DIR)/stdsoap2.o


CFLAGS                          += -pipe -Wall -O0 -Winline -I$(INTERFACE_COMMON_
      LIBRARY_DIR) -DIP_ADDRESS=\"$(IP_ADDRESS)\"

TEMPERATURE_SRC         := temperature.o
BINS                            := temperature

all: $(BINS)

temperature: $(SOAPSOURCES) $(TEMPERATURE_SRC)
        $(CC) $(CFLAGS) -o $@ $^
        $(STRIP) $@

clean:
        -rm -f $(BINS)
        -rm -f $(INTERFACE_COMMON_LIBRARY_DIR)/*.o
        -rm -f *.o
```

2.4 MSX-E systems servers

The MSX-E embeds servers to provide configuration, management and monitoring over Ethernet.

2.4.1 SOAP server

SOAP means Simple Object Access Protocol. This protocol allows you to use the MSX-E software functions over Ethernet. It is providing **Web Services** that can easily be consumed in many programming languages like C, C++, C#, VB.Net... With the SOAP functions, all functionalities of the MSX-E system can be managed / configured / monitored. This server can be accessed on port 5555. All SOAP functions are described under Modules -> MSX-EXXXX functions.

2.4.2 Event server

MSX-E systems embed an event server that can be used to monitor the current MSX-E state. An MSX-E system is logically divided in subsystem states. A subsystem is uniquely identified by a number, as well as each possible state associated to it. These numerical identifiers can be monitored by using the event server, which signals when the state of a subsystem has changed. The MSX-E will send a new event frame as soon as a subsystem state changes on the MSX-E.

2.4.3 Modbus server

A Modbus server, accessible on port 512 permits, for example, to manage MSX-E systems from a PLC. The port number, endianness (Intel / Motorola) and protocol (TCP/UDP) can be configured through the MSX-E web interface.

2.5 Common functions

Modules

- Common general functions

  Various utility functions, mainly to identify a remote system.

- Common temperature functions

  These functions deals with the internal temperature sub-system.

- Common hardware trigger functions

  These functions allow to set and request the current value of the hardware trigger.

- Common security functions

  The "customer key" feature may for instance be used by a customer to be sure that his application communicates only with certified MSX-E modules.

- Common time functions

  A MSX-E module provides a "system clock" that may be in the simplest case set by the function MXCommon__SetTime().

- Common I/O auto configuration functions

  On the web site of some MSX-E module, there is the possibility to define an auto-configuration and auto start of the I/O.

- Common synchronisation timer functions

  When modules are linked through a "synchronisation bus", the master can run a timer that generate a "synchro signal" on the slaves when overrun.

- Set/Backup/Restore general system configuration

  Distinct of the I/O auto-configuration/auto-start functionality, these functions allows to manipulate the general system configuration.

- System state management

  Every MSX-E modules are composed of several sub-systems that work together to provide the system functionalities.

- Customer option management

  Enable to get informations about the options of the system.

- Synchronisation management

  Manage the synchronisation state of the system.

- input filter Filter management

Manages the analog input filters in the system.

2.6 Common general functions

Various utility functions, mainly to identify a remote system.

Functions

- int MXCommon__GetModuleType (void *_, struct MXCommon__ByteArrayResponse *Response)

  This function return the type of the MSX-E Module.

- int MXCommon__GetHostname (void *_, struct MXCommon__ByteArrayResponse *Response)

  This function return the hostname of the MSX-E Module.

- int MXCommon__SetHostname (struct xsd__base64Binary *bHostname, struct MXCommon__-Response *Response)

  This function allows to set the hostname of the MSX-E Module.

- int MXCommon__GetClientConnections (void *_, struct MXCommon__ByteArrayResponse *Response)

  This function return the client connection list.

- int MXCommon__Strerror (xsd__int errnum, struct MXCommon__ByteArrayResponse *Response)

  Call the libc strerror() on the remote device (actually this is a call to strerror_r()).

- int MXCommon__Reboot (void *_, struct MXCommon__Response *Response)

  Ask the MSX-E module to reboot.

- int MXCommon__ResetAllIOFunctionalities (xsd__unsignedLong ulOption, struct MXCommon_-_Response *Response)

  Reset the I/O functionalities of the MSX-E system.

- int MXCommon__DataserverRestart (xsd__unsignedLong ulAction, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)

  Restart the data-server service.

- int MXCommon__GetEthernetLinksStates (void *_, struct MXCommon__-GetEthernetLinksStatesResponse *Response)

  Get MSX-E Ethernet links states.

2.6.1 Function Documentation

2.6.1.1 int MXCommon__GetModuleType (void * _, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** • sArray : Module type string
- sResponse Composed of iReturnValue and syserrno

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.6.1.2 int MXCommon__GetHostname (void * _, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** • sArray : Hostname of the module
- iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.6.1.3 int MXCommon__SetHostname (struct xsd__base64Binary * *bHostname*, struct MXCommon__Response * *Response*)

Parameters

`[in]` ***bHostname*** : Hostname

`[out]` ***Response*** • iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.6.1.4 int MXCommon__GetClientConnections (void * _, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** • sArray : string containing the list of connected clients.
- sResponse Composed of iReturnValue and syserrno

The sArray string is of the form IP-Address:first connection-second connection---- IP-Address:first connection-second connection----

Sample: 172.16.3.43:8989-5555 172.16.3.200:8989

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.6.1.5 int MXCommon__Strerror (xsd__int *errnum,* struct MXCommon__ByteArrayResponse * *Response*)

Usually SOAP functions return this value in a variable named syserror, which is meaningful only when the function return value, usually called iReturnValue, indicate an error (that is, have a value of -1 or -100, depending of the case).

Parameters

[in] ***errnum*** : Error number

[out] ***Response*** • sArray : See the description below.
- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno).
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

```
STRERROR(3)                                                  Linux Programmer's Manual
STRERROR(3)

NAME
strerror, strerror_r - return string describing error code

SYNOPSIS
#include <string.h>

char *strerror(int errnum);

#define _XOPEN_SOURCE 600
#include <string.h>

int strerror_r(int errnum, char *buf, size_t n);

DESCRIPTION
The  strerror() function returns a string describing the error code passed
in the argument errnum,possibly using the LC_MESSAGES part of the current
locale to select the appropriate language.
```

```
This string must not be modified by the application, but may be modified
by a subsequent call to perror() or strerror().  No library  function will
modify this string.

The strerror_r() function is similar to strerror(), but is thread safe.
It returns the string in the user-supplied buffer buf of length n.

RETURN VALUE
The strerror() function returns the appropriate error description string,
or an unknown error message if the error code is unknown.
The value of errno is not changed for a successful call, and is set to a non-zero
value upon error.
The strerror_r() function returns 0 on success and -1 on failure, setting errno.

ERRORS
EINVAL The value of errnum is not a valid error number.

ERANGE Insufficient storage was supplied to contain the error description string.

CONFORMING TO
SVID 3, POSIX, 4.3BSD, ISO/IEC 9899:1990 (C89).
strerror_r() with prototype as given above is specified by SUSv3,
and was in use under Digital Unix and HP Unix. An incompatible function,
with prototype

char *strerror_r(int errnum, char *buf, size_t n);

is a GNU extension used by glibc (since 2.0), and must be regarded as obsolete
in view of SUSv3.
The GNU version may, but need not, use the user-supplied buffer.
If it does, the result may be truncated in case the supplied buffer is too small.
The result is always NUL-terminated.

SEE ALSO
errno(3), perror(3), strsignal(3)
```

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.6.1.6 int MXCommon__Reboot (void * _, struct MXCommon__Response * *Response*)

Parameters

`[in]` `_` : no input parameter

`[out]` ***Response***

- iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.6.1.7 int MXCommon__ResetAllIOFunctionalities (xsd__unsignedLong *ulOption*, struct MXCommon__Response * *Response*)

The behavior of the function depends on the MSX-E system that is used.

```
On MSX-E3511: Stop the watchdogs and stop the generators
On MSX-E3601: Stop the sequence acquisition and stop the calibration
On MSX-E3701: Stop the acquisition
```

Parameters

[in] ***ulOption*** Reserved. Set to 0

[out] ***Response*** ***iReturnValue***

- **0** The remote function performed OK
- **-1** Internal system error occurred. See value of syserrno
- **-100** Function not supported by the system

syserrno system error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

2.6.1.8 int MXCommon__DataserverRestart (xsd__unsignedLong *ulAction*, xsd__unsignedLong *ulOption*, struct MXCommon__Response * *Response*)

Parameters

[in] ***ulAction*** : action

- 0: normal restart
- 1: with cache file reset
- 2: with cache file deletion

[in] ***ulOption*** : Reserved

[out] ***Response*** • iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

Note

(revision>6386) Depending on the system type, can be used to restart the data-recv service as well. In this case, parameter action is ignored.

2.6.1.9 int MXCommon__GetEthernetLinksStates (void * _, struct MXCommon__GetEthernetLinksStatesResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** Structure that contains the MSX-E Ethernet links states and errors:

sResponse.iReturnValue

- **0** The remote function performed OK
- **-1** System error occurred
- **-2** Fail to get Ethernet links states
- **-100** Internal system error occurred. See value of syserrno

sResponse.syserrno system error code (the value of the libc "errno" code)

sPort0: Fisrt port informations

- **ulState**
  - **0** Link down
  - **1** Link up
- **ulSpeed**
  - **10** 10 Mb/s
  - **100** 100 Mb/s
- **ulDuplex**
  - **0** Half duplex
  - **1** Full duplex
- **ulInfo1** Reserverd
- **ulInfo2** Reserverd

sPort1: Second port informations

- **ulState**
  - **0** Link down
  - **1** Link up
- **ulSpeed**
  - **10** 10 Mb/s
  - **100** 100 Mb/s
- **ulDuplex**
  - **0** Half duplex
  - **1** Full duplex
- **ulInfo1** Reserverd
- **ulInfo2** Reserverd

Return values

0 SOAP_OK

Others See SOAP error

2.7 Common temperature functions

These functions deals with the internal temperature sub-system.

Data Structures

- struct MXCommon__GetModuleTemperatureValueAndStatusResponse

Functions

- int MXCommon__GetModuleTemperatureValueAndStatus (xsd__unsignedLong ulOption, struct MXCommon__GetModuleTemperatureValueAndStatusResponse *Response)

  Read the temperature on the module.

- int MXCommon__SetModuleTemperatureWarningLevels (xsd__double dMinimalWarningLevel, xsd__double dMaximalWarningLevel, xsd__unsignedLong ulOption, struct MXCommon__-Response *Response)

  Set the temperature warning level on the module.

2.7.1 Detailed Description

The role of this sub-system is to monitor the internal temperature of a module and issue a warning if it is below or above a threshold. If the internal temperature reaches a domain where the system is endangered, it switches automatically in a degraded working mode.

2.7.2 Function Documentation

2.7.2.1 int MXCommon__GetModuleTemperatureValueAndStatus (xsd__unsignedLong *ulOption*, struct MXCommon__GetModuleTemperatureValueAndStatusResponse * *Response*)

Parameters

`[in]` ***ulOption*** : Reserved

`[out]` ***Response***

- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
  - dValue : Temperature value in Degree Celsius
- ulTemperatureStatus : Temperature Status :
  - TEMPERATURE_INITIAL = 0 : Temperature not ready
  - TEMPERATURE_TOOLOW = 1 : Temperature too low !
  - TEMPERATURE_LOW = 2 : Temperature under the min warning value
  - TEMPERATURE_NOMINAL = 3 : Temperature in the nominal range
  - TEMPERATURE_HIGH = 4 : Temperature over the max warning value
  - TEMPERATURE_TOOHIGH = 5 : Temperature too high !

- ulInfo : Reserved

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.7.2.2 int MXCommon__SetModuleTemperatureWarningLevels (xsd__double *dMinimalWarningLevel,* xsd__double *dMaximalWarningLevel,* xsd__unsignedLong *ulOption,* struct MXCommon__Response * *Response*)

Parameters

[in] ***dMinimalWarningLevel*** : Minimal temperature warning level in Degree : 5 to 60 Degree Celsius

[in] ***dMaximalWarningLevel*** : Maximal temperature warning level in Degree : 5 to 60 Degree Celsius

[in] ***ulOption*** : Reserved

[out] ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.8 Common hardware trigger functions

These functions allow to set and request the current value of the hardware trigger.

Data Structures

- struct MXCommon__GetHardwareTriggerFilterTimeResponse
- struct MXCommon__GetHardwareTriggerStateResponse

Functions

- int MXCommon__SetHardwareTriggerFilterTime (xsd__unsignedLong ulFilterTime, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)

  Sets the filter time for the hardware trigger input in steps of 250 ns (max value: 65535).

- int MXCommon__GetHardwareTriggerFilterTime (xsd__unsignedLong ulOption, struct MXCommon__GetHardwareTriggerFilterTimeResponse *Response)

  Get the filter time for the hardware trigger input.

- int MXCommon__GetHardwareTriggerState (xsd__unsignedLong ulOption, struct MXCommon__GetHardwareTriggerStateResponse *Response)

  Get the hardware trigger state after the filter.

2.8.1 Function Documentation

2.8.1.1 int MXCommon__SetHardwareTriggerFilterTime (xsd__unsignedLong *ulFilterTime,* xsd__unsignedLong *ulOption,* struct MXCommon__Response * *Response*)

Sets the filter time for the hardware trigger input in steps of 250 ns (max value: 65535).

On the MSX-E3011 system, the step of the hardware trigger filter is **622ns**.

Parameters

`[in]` ***ulFilterTime*** Filter time for the hardware trigger input in steps of 250ns (max value : 65535).

- **0**: Disable the filter
- **1**: Sets the filter time to 250 ns
- **2**: Sets the filter time to 500 ns
- ...
- **65535**: Sets the filter time to 16 ms

`[in]` ***ulOption*** Reserved. Set to 0

`[out]` ***Response*** Response of the system

- ***sResponse.iReturnValue***
  - **0**: The remote function performed OK
  - **-1**: Internal system error occurred. See value of syserrno
- ***sResponse.syserrno*** system error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

2.8.1.2 int MXCommon__GetHardwareTriggerFilterTime (xsd__unsignedLong *ulOption,* struct MXCommon__GetHardwareTriggerFilterTimeResponse * *Response*)

Get the filter time for the hardware trigger input in **250ns** step (max value : 65535).

On the MSX-E3011 system, the step of the hardware trigger filter is **622ns**.

Parameters

`[in]` ***ulOption*** Reserved. Set to 0

`[out]` ***Response*** Response of the system

- ***ulFilterTime*** filter time for the hardware trigger input
  - **0**: filter disabled
  - **1**: filter of 250ns
  - **2**: filter of 500ns
  - ...
  - **65535**: filter of 16ms
- ***sResponse.iReturnValue***
  - **0**: The remote function performed OK
  - **-1**: Internal system error occurred. See value of syserrno
- ***sResponse.syserrno*** system error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

2.8.1.3 int MXCommon__GetHardwareTriggerState (xsd__unsignedLong *ulOption,* struct MXCommon__GetHardwareTriggerStateResponse * *Response*)

Parameters

`[in]` ***ulOption*** : Reserved

`[out]` ***Response*** • ulState : Hardware trigger input state.
  – 0: Hardware trigger input is low
  – 1: Hardware trigger input is high.
• sResponse.iReturnValue : Return value
  – 0 : success
  – -1: system error (see syserrno)
• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.9 Common security functions

The "customer key" feature may for instance be used by a customer to be sure that his application communicates only with certified MSX-E modules.

Data Structures

- struct MXCommon__TestCustomerIDResponse

Functions

- int MXCommon__SetCustomerKey (struct xsd__base64Binary *bKey, struct xsd__base64Binary *bPublicKey, struct MXCommon__Response *Response)

  Set the Customer key.

- int MXCommon__TestCustomerID (void *_, struct MXCommon__TestCustomerIDResponse *Response)

  Test the Customer ID (if the module has the right customer Key).

2.9.1 Detailed Description

A "customer key" consists of two strings of data stored on the certified MSX-E module, to be used by the function MXCommon__TestCustomerID() to encrypt data.

These strings can not be read back. They are supposed to be kept secret by the user of this functionality.

To test if the MSX-E module you use is certified, you can request the MSX-E module to provide a set of randomly generated data and the result of the encryption (through the use of the stored "customer key") of the same data. Then your application must encrypt the delivered random data with its own "customer key" and compare it with the encrypted data delivered by the MSX-E module.

If the results are matching, the MSX-E module is certified for this application.

Detailed presentation of operations:

The user generates and stores on the module two keys (thanks to the software function : MXCommon__SetCustomerKey()). This needs only to be done once:

- A public Key K1 (16 Bytes)
- A private Key K2 (32 Bytes)

When requested (with the software function : MXCommon__TestCustomerID()), the module generates a 16 bytes random value and do an encryption of this value using the two saved keys and the AES algorithm (Rijndael).

The user receives then two arrays of 16 bytes :

- one with a random value [A]
- the second with encrypted random value [B]

[B]=AES([A], K1, K2)

The user performs then the same computation from [A],K1,K2 and compares his result with [B]. If it is the same, it means that the module he is using was already configured with the correct identification token.

The security of the method comes from that even knowing [A] and [B] no one can deduce K1 and K2 back in practical times. ADDI-DATA is not aware of a practical way to remotely retrieve the value of the key stored on a module.

It is the responsibility of the developer of the application to ensure that these tokens are suitably protected. The authorisation of the change of the "customer key" on the MSX-E module can be managed with the web interface.

The use of the "customer key" don't have an impact of the other functionalities of the MSX-E module.

2.9.2 Function Documentation

2.9.2.1 int MXCommon__SetCustomerKey (struct xsd__base64Binary * *bKey*, struct xsd__base64Binary * *bPublicKey*, struct MXCommon__Response * *Response*)

Parameters

`[in]` ***bKey*** : Customer key (only writable on the module) [32 bytes containing a AES key]

`[in]` ***bPublicKey*** : IV (Initialisation vector) for the AES cryptography [16 bytes containing a AES key]

`[out]` ***Response*** • sResponse.iReturnValue : Return value

  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.9.2.2 int MXCommon__TestCustomerID (void * _, struct MXCommon__TestCustomerIDResponse * *Response*)

Parameters

`[in]` _ : No Input

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- bValueArray : non encrypted value array [16 bytes of random data]
- bCryptedValueArray : Encrypted value array [16 bytes of the encrypted random data]

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.10 Common time functions

A MSX-E module provides a "system clock" that may be in the simplest case set by the function MXCommon__SetTime().

Data Structures

- struct MXCommon__GetTimeResponse
- struct MXCommon__GetUpTimeResponse

Functions

- int MXCommon__SetTime (xsd__unsignedLong ulLowTime, xsd__unsignedLong ulHighTime, struct MXCommon__Response *Response)

  Set the time on the module.

- int MXCommon__SysToHardwareClock (void *_, struct MXCommon__Response *Response)

  Set the hardware clock (if present) to the current system time.

- int MXCommon__HardwareClockToSys (void *_, struct MXCommon__Response *Response)
  Set the system time from the hardware clock (if present).
- int MXCommon__GetTime (void *_, struct MXCommon__GetTimeResponse *Response)
  Get the time on the module.
- int MXCommon__GetUpTime (void *_, struct MXCommon__GetUpTimeResponse *Response)
  Ask the MSX-E module uptime (number of seconds since the last boot).

2.10.1 Detailed Description

If the module is configured to use NTP, the time received by the NTP server will have a greater priority. If the module is linked to another through a "synchronization bus" and is slave, then the time received from the master is the one taken into account.

Recent models also provide a "hardware clock", a component whose role is to track the time between reboots.

2.10.2 Function Documentation

2.10.2.1 int MXCommon__SetTime (xsd__unsignedLong *ulLowTime*, xsd__unsignedLong *ulHighTime*, struct MXCommon__Response * *Response*)

Parameters

`[in]` ***ulLowTime*** : Number of microseconds since the begin of the second

`[in]` ***ulHighTime*** : Number of seconds since the Epoch (1st January,1970)

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.10.2.2 int MXCommon__SysToHardwareClock (void * _, struct MXCommon__Response * *Response*)

Parameters

`[in]` _ No input parameter

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

If this function fails, it means the module does not have a hardware RTC, or the hardware is not functional. Check the "hwclock" subsystem status.

2.10.2.3 int MXCommon__HardwareClockToSys (void * _, struct MXCommon__Response * *Response*)

When the hardware clock is present, the system time is automatically set to it when the module becomes master on the inter-module synchronisation bus.

Parameters

`[in]` _ No input parameter

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

If this function fails, it means the module does not have a hardware RTC, or the hardware is not functional. Check the "hwclock" subsystem status.

2.10.2.4 int MXCommon__GetTime (void * _, struct MXCommon__GetTimeResponse * *Response*)

Parameters

`[in]` _ : No input parameter

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

• ulLowTime : Number of microseconds since the begin of the second

• ulHighTime : Number of seconds since the Epoch (1st January,1970)

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.10.2.5 int MXCommon__GetUpTime (void * _, struct MXCommon__GetUpTimeResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

• ulUpTime : Number of seconds since the last boot of the system.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.11 Common I/O auto configuration functions

On the web site of some MSX-E module, there is the possibility to define an auto-configuration and auto start of the I/O.

Data Structures

- struct MXCommon__GetAutoConfigurationFileResponse

Functions

- int MXCommon__GetAutoConfigurationFile (void *_, struct MXCommon__GetAutoConfigurationFileResponse *Response)

  Get the auto configuration file of the module.

- int MXCommon__SetAutoConfigurationFile (struct xsd__base64Binary *ByteArrayInput, xsd__unsignedLong ulEOF, struct MXCommon__Response *Response)

  Set the auto configuration file of the module.

- int MXCommon__StartAutoConfiguration (void *_, struct MXCommon__ByteArrayResponse *Response)

  start/Restart the auto configuration

2.11.1 Detailed Description

- Auto-configuration means the system configures the I/O automatically at boot time.
- Auto-start means the system starts an acquisition automatically at boot time (this may no make sense for some systems). It implies auto-configuration.

This set of functions allows to:

- get the auto-configuration/start currently set on module, as a read-only binary file.
- set a auto-configuration/start on the module, using a previously saved file.
- start or restart the auto-configuration/start on the module, using the current configuration saved on the module.

2.11.2 Function Documentation

2.11.2.1 int MXCommon__GetAutoConfigurationFile (void * _, struct MXCommon__GetAutoConfigurationFileResponse * *Response*)

Parameters

`[in]` _ : No input parameter

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
  - -100 : Error of the read of the auto configuration file

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
• bArray : Array of Bytes of the file
• ulEOF : End of file flag

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.11.2.2 int MXCommon__SetAutoConfigurationFile (struct xsd__base64Binary * *ByteArrayInput,* xsd__unsignedLong *ulEOF,* struct MXCommon__Response * *Response*)

Parameters

`[in]` ***ByteArrayInput*** : Array of Bytes of the file

`[in]` ***ulEOF*** : End of file flag

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.11.2.3 int MXCommon__StartAutoConfiguration (void * _, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

[in] _ : No input parameter

[out] ***Response***
- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
  - sArray : message returned by the auto configuration start

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.12 Common synchronisation timer functions

When modules are linked through a "synchronisation bus", the master can run a timer that generate a "synchro signal" on the slaves when overrun.

Functions

- int MXCommon__InitAndStartSynchroTimer (xsd__unsignedLong ulTimeBase, xsd__unsignedLong ulReloadValue, xsd__unsignedLong ulNbrOfCycle, xsd__unsignedLong ulGenerateTriggerMode, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, xsd__unsignedLong ulOption03, xsd__unsignedLong ulOption04, struct MXCommon__Response *Response)

  Initialises and starts the synchronisation timer of the module (not already available on all module).

- int MXCommon__StopAndReleaseSynchroTimer (xsd__unsignedLong ulOption01, struct MXCommon__Response *Response)

  start/Restart the synchronisation timer (not already available on all module)

2.12.1 Function Documentation

2.12.1.1 int MXCommon__InitAndStartSynchroTimer (xsd__unsignedLong *ulTimeBase*, xsd__unsignedLong *ulReloadValue*, xsd__unsignedLong *ulNbrOfCycle*, xsd__unsignedLong *ulGenerateTriggerMode*, xsd__unsignedLong *ulOption01*, xsd__unsignedLong *ulOption02*, xsd__unsignedLong *ulOption03*, xsd__unsignedLong *ulOption04*, struct MXCommon__Response * *Response*)

Parameters

[in] ***ulTimeBase*** : Time base of the timer (0 for us, 1 for ms, 2 for s)

[in] ***ulReloadValue*** : Timer reload value (0 to 0xFFFF), minimum reload time is 5 us

[in] ***ulNbrOfCycle*** : Number of timer cycle

- 0: continuous
- > 0: defined number of cycle

[in] ***ulGenerateTriggerMode*** :

- 0: Wait the time overflow to set the synchronisation trigger
- 1: Set the synchronisation trigger by the start of the timer and after each time overflow

[in] ***ulOption01*** : Define the source of the trigger

- 0 : Trigger disabled
- 1 : Enable the hardware digital input trigger

[in] ***ulOption02*** : Define the edge of the hardware trigger who generates a trigger action

- 1 : rising edge (Only if hardware trigger selected)
- 2 : falling edge (Only if hardware trigger selected)
- 3 : Both front (Only if hardware trigger selected)

[in] ***ulOption03*** : Define the number of trigger events before the action occur

- 1 : all trigger event start the action
- max value : 65535

[in] ***ulOption04*** : Reserved

[out] ***Response*** • sResponse.iReturnValue : Return value

  - 0 : success
  - -1: system error (see syserrno)
  - -2: not available time base
  - -3: timer reload value can not be greater than 65535
  - -4: minimum time reload is 5 us
  - -5: Number of cycle can not be greater than 65535
  - -6: Generate trigger mode error
  - -100: Init timer error
  - -101: Start timer error

- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror(). May be ENOSYS : Function not implemented.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.12.1.2 int MXCommon__StopAndReleaseSynchroTimer (xsd__unsignedLong *ulOption01*, struct MXCommon__Response * *Response*)

Parameters

[in] ***ulOption01*** : Reserved

[out] ***Response*** • sResponse.iReturnValue : Return value

  - 0 : success
  - -1: system error (see syserrno)
  - -100: Start/Stop timer error

- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror(). May be ENOSYS : Function not implemented.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.13 Set/Backup/Restore general system configuration

Distinct of the I/O auto-configuration/auto-start functionality, these functions allows to manipulate the general system configuration.

Functions

- int MXCommon__GetConfigurationBackupFile (void *_, struct MXCommon__FileResponse *Response)

  Download a configuration backup file from the module.

- int MXCommon__ApplyConfigurationBackupFile (struct xsd__base64Binary *ByteArrayInput, xsd__unsignedLong ulEOF, struct MXCommon__Response *Response)

  Upload a new configuration on the module.

- int MXCommon__ChangePassword (struct xsd__base64Binary *PreviousUser, struct xsd__base64Binary *PreviousPassword, struct xsd__base64Binary *NewUser, struct xsd__base64Binary *NewPassword, struct MXCommon__Response *Response)

  Set a new id/password.

2.13.1 Detailed Description

It includes the network configuration, and generally everything that can be set up through the web interface.

These functions have been included to ease the automation of module customisation. They may be disabled using the web interface, in "Security/Remote general system configuration authorisation/remote sysconf changes"

2.13.2 Function Documentation

2.13.2.1 int MXCommon__GetConfigurationBackupFile (void * _, struct MXCommon__FileResponse * *Response*)

Parameters

`[in]` _ : No input parameter

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  – 0 : success
  – -1: system error (see syserrno) (see syserrno)
  • sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
  • bArray : Array of Bytes of the file
  • ulEOF : End of file flag

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

This function is designed to be called repeatedly until no more data is available. At this point the flag ulEOF is set.

Below is an example in pseudo-C.

```
int dummy;
struct MXCommon__FileResponse Response;
while(1)
{
if ( MXCommon__GetConfigurationBackupFile(&dummy, &Response) != SOAP_OK)
{
// handle soap error
}
if (Response.iReturnValue)
{
// handle remote error (Response.syserrno contains more information)
}
// do something with the data, for example save it in a file
write(fd, Response.bArray.__ptr, Response.bArray.__size);
// if this is the end of the file, quit the loop
if(Response.ulEOF)
break;
}
*
```

2.13.2.2 int MXCommon__ApplyConfigurationBackupFile (struct xsd__base64Binary * *ByteArrayInput,* xsd__unsignedLong *ulEOF,* struct MXCommon__Response * *Response*)

Parameters

[in] ***ByteArrayInput*** : Array of Bytes of the file

[in] ***ulEOF*** : End of file flag

[out] ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

This function is designed to be called repeatedly until all data is transfered. At this point the flag ulEOF must be set to 1. The new configuration is then applied.

2.13.2.3 int MXCommon__ChangePassword (struct xsd__base64Binary * *PreviousUser,* struct xsd__base64Binary * *PreviousPassword,* struct xsd__base64Binary * *NewUser,* struct xsd__base64Binary * *NewPassword,* struct MXCommon__Response * *Response*)

The changes are immediately active.

Parameters

[in] _ : No input parameter

[out] ***Response***
- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: string PreviousUser is invalid
  - -2: string PreviousPassword is invalid
  - -3: string NewUser is invalid
  - -4: string NewPassword is invalid
  - -5: authentification failed
  - -100: system error while saving tokens (use syserrno for more information)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- sArray : message returned by the auto configuration start

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

Warning

The parameters transit in clear text. Use this functionality only on trusted networks.
Given that ADDI-DATA GmbH takes security seriously, there is no way to change the password without knowing it. No "hidden back-door". This function makes it all too easy to lock a module, if you don't remember the password you set on it.

2.14 System state management

Every MSX-E modules are composed of several sub-systems that work together to provide the system functionalities.

Functions

- int MXCommon__GetSubSystemState (xsd__unsignedLong SubsystemID, struct MXCommon__-unsignedLongResponse *Response)

  Returns the current state of the specified sub-system.

- int MXCommon__GetSubsystemIDFromName (struct xsd__base64Binary *SubsystemName, struct MXCommon__unsignedLongResponse *Response)

  Returns the ID of the sub-system of symbolic name "SubsystemName".

- int MXCommon__GetStateIDFromName (xsd__unsignedLong SubsystemID, struct xsd__-base64Binary *StateName, struct MXCommon__unsignedLongResponse *Response)

  Returns the ID of the state of symbolic name "StateName" of the sub-system of ID "SubsystemID".

- int MXCommon__GetSubsystemNameFromID (xsd__unsignedLong SubsystemID, struct MXCommon__ByteArrayResponse *Response)

  Returns the symbolic name of the sub-system of numerical ID "SubsystemName".

- int MXCommon__GetStateNameFromID (xsd__unsignedLong SubsystemID, xsd__unsignedLong StateID, struct MXCommon__ByteArrayResponse *Response)

  Returns the symbolic name of the state of numerical ID "StateID" of the sub-system of ID "SubsystemID".

2.14.1 Detailed Description

These sub-systems have a state that, for example, indicate if it functions nominally.

A sub-system is identified by its ID (a positive integer) and its symbolic name. Each state in the set of possible states for a given sub-system has also an ID and a symbolic name.

Names are less likely to change between releases of the MSX-E operating system. That is why manipulating names should be preferred against indexes in an application. Still, manipulating ID is more efficient.

The functions in this section provide a way to retrieve the association between names and indexes. MXCommon__GetSubSystemState() requests the state of a given sub-system.

Notice that the event manager is the recommended way to be warned of a change of state.

The list of sub-systems and their ID and associated name can be consulted on the web site of the module.

2.14.2 Function Documentation

2.14.2.1 int MXCommon__GetSubSystemState (xsd__unsignedLong *SubsystemID,* struct MXCommon__unsignedLongResponse * *Response*)

Parameters

`[in]` ***SubsystemID*** sub-system numerical ID

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameter SubsystemID

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

• Value The state of the sub-system "Id" at the moment of the execution of the request.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.14.2.2 int MXCommon__GetSubsystemIDFromName (struct xsd__base64Binary * *SubsystemName,* struct MXCommon__unsignedLongResponse * *Response*)

Parameters

`[in]` ***SubsystemName*** sub-system symbolic name.

`[out]` ***Response*** • sResponse.iReturnValue :Return value
  - 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameter SubsystemName

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

• Value The numerical ID of the sub-system "SubsystemName".

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.14.2.3 int MXCommon__GetStateIDFromName (xsd__unsignedLong *SubsystemID*, struct xsd__base64Binary * *StateName*, struct MXCommon__unsignedLongResponse * *Response*)

Parameters

`[in]` ***SubsystemID*** sub-system numerical ID

`[in]` ***StateName*** state symbolic name.

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameters SubsystemID or StateName

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

• Value The numerical ID of the state "StateName".

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.14.2.4 int MXCommon__GetSubsystemNameFromID (xsd__unsignedLong *SubsystemID*, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

`[in]` ***SubsystemID*** sub-system numerical ID.

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameter SubsystemName

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

• sArray : The symbolic name associated with the ID.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.14.2.5 int MXCommon__GetStateNameFromID (xsd__unsignedLong *SubsystemID*, xsd__unsignedLong *StateID*, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

`[in]` ***SubsystemID*** sub-system numerical ID.

`[in]` ***StateID*** sub-system numerical ID.

[out] ***Response*** • sResponse.iReturnValue : Return value
  – 0 success
  – -1 system error while executing the request (see syserrno)
  – -2 invalid parameters SubsystemID or StateID
• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
• sArray The symbolic name associated with the state numerical ID.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.15 Customer option management

Enable to get informations about the options of the system.

Functions

- int MXCommon__GetOptionInformation (void *_, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__ByteArrayResponse *Response)

  Enables to get information about the options available on the system.

2.15.1 Function Documentation

2.15.1.1 int MXCommon__GetOptionInformation (void * _, xsd__unsignedLong *ulOption01*, xsd__unsignedLong *ulOption02*, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

[in] ***ulOption01,:*** not used, set it to 0

[in] ***ulOption02,:*** not used, set it to 0

[out] ***Response*** • sArray : Option information string
  • sResponse Composed of iReturnValue and syserrno

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.16 Synchronisation management

Manage the synchronisation state of the system.

Functions

- int MXCommon__SetToMaster (void *_, xsd__unsignedLong ulState, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__Response *Response)

  Writes if the MSXE has to be always set to master The master mode (when enabled) make the system always detected as master.

- int MXCommon__GetSynchronizationStatus (void *_, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__unsignedLongResponse *Response)

  Reads the status of the synchronization for the corresponding MSXE The master mode (when enabled) make the system always detected as master.

2.16.1 Function Documentation

2.16.1.1 int MXCommon__SetToMaster (void * _, xsd__unsignedLong *ulState*, xsd__unsignedLong *ulOption01*, xsd__unsignedLong *ulOption02*, struct MXCommon__Response * *Response*)

Parameters

[in] ***ulState*** State of the supermaster mode

- **0** automatic mode (default). The state of the system (master or slave) will be automatically detected by the system
- **1** Set to master mode at all time. The system will always be detected as master

[in] ***ulOption01*** Reserved. Set to 0

[in] ***ulOption02*** Reserved. Set to 0

[out] ***Response iReturnValue***

- **0** The remote function performed OK
- **-1** System error occurred
- **-2** The PLD is not working
- **-3** The ulFilterTime parameter is wrong
- **-100** Internal system error occurred. See value of syserrno ***syserrno*** system error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

2.16.1.2 int MXCommon__GetSynchronizationStatus (void * _, xsd__unsignedLong *ulOption01*, xsd__unsignedLong *ulOption02*, struct MXCommon__unsignedLongResponse * *Response*)

Parameters

[in] ***ulOption01*** Reserved. Set to 0

[in] ***ulOption02*** Reserved. Set to 0

[out] ***Response sResponse.iReturnValue***

- **0** The remote function performed OK
- **-1** System error occurred
- **-2** The PLD is not working
- **-100** Internal system error occurred. See value of syserrno

sResponse.syserrno system error code (the value of the libc "errno" code)

ulValue State of the supermaster mode

- **0** Automatic mode (default). The state of the system (master or slave) will be automatically detected by the system
- **1** MSXE is always set as a master. The system will always be detected as master

Return values

0 SOAP_OK

Others See SOAP error

2.17 input filter Filter management

Manages the analog input filters in the system.

Functions

- int MXCommon__SetFilterChannels (struct xsd__base64Binary *ChannelList, struct MXCommon__Response *Response)

  This function sets or resets a filter to a channel.

2.17.1 Function Documentation

2.17.1.1 int MXCommon__SetFilterChannels (struct xsd__base64Binary * *ChannelList,* struct MXCommon__Response * *Response*)

Parameters

`[in]` ***ChannelList*** Each index of the array represents a channel. A filter can be affected to each channel. If FilterID = 0, no filter is set (the filter is disabled on the corresponding channel). e.g.: ChannelList[0] = FilterID // Set FilterID on channel 0.

`[out]` ***Response***

- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

2.18 MX370x Get informations functions

Data Structures

- struct MX370x__TransducerGetTypeInformationResponse
- struct MSXE370x__doubleArrayParam

Functions

- int MX370x__TransducerGetNbrOfType (void *_, struct MX370x__unsignedlongResponse *Response)

  Returns the number of transducer types currently defined in the database.

2.18.1 Function Documentation

2.18.1.1 int MX370x__TransducerGetNbrOfType (void * _, struct MX370x__unsignedlongResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue : Error value

- 0: success
- <> 0: error
- -100: kernel function error

ulValue : number of transducers type.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.19 MX370x Auto refresh functions

In the auto refresh mode the measurement value is updated automatically after each acquisition.

Functions

- int MX370x__TransducerInitAndStartAutoRefresh (xsd__unsignedLong ulTransducerSelection, xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulAverageMode, xsd__unsignedLong ulAverageValue, xsd__unsignedLong ulDivisionFactor, xsd__unsignedLong ulTriggerAction, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulHardwareTriggerFilterTime, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, struct MX370x__Response *Response)

  Initialise and start the transducer auto refresh acquisition mode.

- int MX370x__TransducerGetAutoRefreshValues (void *_, struct MX370x__-unsignedlong17ArrayResponse *Response)

  This function get the auto refresh counter value an the channels values.

- int MX370x__TransducerStopAndReleaseAutoRefresh (void *_, struct MX370x__Response *Response)

  Stop and release the transducer auto refresh acquisition mode.

2.19.1 Detailed Description

The analog acquisition is initialised and the values of each channels are stored in memory on the Ethernet module MSX-E370x.

The PC reads the data asynchronously to the acquisition via the data socket or a SOAP function.

You can define a mask of all channels that should be acquired.

In the auto refresh mode you can activate the channel average value computation on the module:

- Average value calculation per channel : Each channel is acquired x times to compute an average value for the channel.
- Average value calculation per sequence : All sequences are acquired x times to compute a average value per channel.

You can start the acquisition by a hardware trigger in the auto refresh and sequence mode.

The hardware trigger can react to a rising edge, falling edge or both edges.

You have the following possibilities:

- Initialising a filter on the trigger input to avoid errors
- Defining a number of edges before a trigger action is generated

There are two trigger modes:

a) One shot

b) Sequence

a) One shot:

After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.

b) Sequence:

After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

There is also the possibility to stop the acquisition with the hardware trigger.

This can only be do when :

- the hardware trigger is not used to start the acquisition.

- the hardware trigger is used in one shot mode to start the acquisition.

In the case of the hardware trigger is used to start the acquisition in one shot mode, then the hardware trigger start condition can restart the acquisition !

2.19.2 Function Documentation

2.19.2.1 int MX370x__TransducerInitAndStartAutoRefresh (xsd__unsignedLong *ulTransducerSelection,* xsd__unsignedLong *ulChannelMask,* xsd__unsignedLong *ulAverageMode,* xsd__unsignedLong *ulAverageValue,* xsd__unsignedLong *ulDivisionFactor,* xsd__unsignedLong *ulTriggerAction,* xsd__unsignedLong *ulHardwareTriggerCount,* xsd__unsignedLong *ulHardwareTriggerFilterTime,* xsd__unsignedLong *ulByTriggerNbrOfSeqToAcquire,* xsd__unsignedLong *ulOption1,* xsd__unsignedLong *ulOption2,* xsd__unsignedLong *ulOption3,* xsd__unsignedLong *ulOption4,* struct MX370x__Response ∗ *Response*)

Parameters

[in] ***ulTransducerSelection*** : Transducer type selection

[in] ***ulChannelMask*** : Mask of the channel to acquire by the auto refresh (1 bit = 1 Channel) for example :

- 0x3 : Channel 0, channel 1
- 0xFF : Channel 0 to 7
- 0xF0 : Channel 3 to 7

[in] ***ulAverageMode*** : Set the average mode :

- 0 : not used
- 1 : average per Sequence : All sequences are acquired x times to compute an average value per channel.
- 2 : average per channel : Each channel is acquired x times to compute an average value for the channel.

[in] ***ulAverageValue*** : Set the average value (only used, when average is used)

- 0 : average not used
- max value : 255

[in] ***ulDivisionFactor*** : Division factor (min: 5, max: 255)

[in] ***ulTriggerAction*** : Trigger action :

Hardware Trigger Start D0 - D7

Bit 3,2,1,0 : Define the trigger mode

- 0000 : Trigger disabled
- 0001 : One shot trigger : After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.
- 0010 : Sequence trigger : After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

Bit 7,6 : define the active front (Only if hardware trigger selected)

- 01 : rising front (Only if hardware trigger selected)
- 10 : falling front (Only if hardware trigger selected)
- 11 : Both front (Only if hardware trigger selected)

Synchronisation Trigger Start : D8-D15

Bit 11,10,9,8 : Define the trigger mode

- 0000 : trigger disabled
- 0001 : One shot trigger : After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.
- 0010 : Sequence trigger : After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

Hardware Trigger Stop D16 - D19

The hardware trigger stop can only be activated when :

- The hardware trigger start is not used.
- The hardware trigger start is used in one shot mode.

The stop of the acquisition is really do at the end of a sequence acquisition(to avoid that the acquisition is stop in the middle of a sequence).

Bit 16 : Define the trigger stop is enable or not

- 0 : Stop trigger disabled
- 1 : Stop trigger enabled.

Bit 18,17 : define the active front (Only if hardware trigger stop selected)

- 01 : rising front (Only if hardware trigger stop selected)
- 10 : falling front (Only if hardware trigger stop selected)
- 11 : Both front (Only if hardware trigger stop selected)

Bit 19 : define if the hardware trigger stop use the ulHardwareTriggerCount (Only if hardware trigger stop selected)

- 0 : ulHardwareTriggerCount not used : First hardware trigger stop will stop the acquisition
- 1 : ulHardwareTriggerCount is used : The ulHardwareTriggerCount hardware trigger will stop the acquisition

[in] ***ulHardwareTriggerCount*** : Define the number of trigger events before the trigger action occur

0 or 1 : all trigger event start the trigger action

max value : 65535

[in] ***ulHardwareTriggerFilterTime*** : Filter time for the hardware trigger (= multiplier from 250 ns step)

max value : 65535

[in] ***ulByTriggerNbrOfSeqToAcquire*** : Define the number of sequence to acquire by each trigger event

[in] ***ulOption1*** : Data format option

D0 : Time stamp information

- 0 : no time stamp information
- 1 : time stamp information

D1 : Data format

- 0 : Digital value
- 1 : Analog value (in mm)

D2 : invert value

- 0 : don't invert the channel value
- 1 : invert the channel value (-2 mm -> + 2mm)

[in] ***ulOption2*** : Reserved

[in] ***ulOption3*** : Reserved

[in] ***ulOption4*** : Reserved

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -2: Transducer selection error
- -3: Channel mask error : can not be null
- -4: Channel mask error
- -5: Average mode error
- -6: Average value error
- -7: Division factor error
- -8: Incorrect value for Hardware Trigger Mode
- -9: Incorrect value for Hardware Trigger front
- -10: Incorrect value for Synchro Trigger Mode
- -11: Incorrect value for Hardware Trigger count
- -12: Incorrect value for Hardware Trigger filter time
- -13: Incorrect value for "trigger number of sequence to acquire"
- -14: Wrong data format parameter (ulOption1)
- -15: A value for Hardware Trigger front was defined but Hardware Trigger Mode is not set
- -16: Cannot use both triggers at the same time
- -17: Incorrect value for the hardware trigger stop front
- -18: Hardware trigger stop can not be used by this configuration of hardware trigger start
- -100: TransducerInit kernel function error
- -101: InitConvertTimeDivisionFactor kernel function error
- -102: SetAutoRefreshAverageValue kernel function error
- -103: InitDigitalInputFilter kernel function error
- -104: InitEnableDisableHardwareTrigger kernel function error
- -105: SynchroTrigger Init/Enable/Disable kernel function error
- -106: SetTriggerSequenceCount kernel function error
- -107: StartAutoRefresh kernel function error

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.19.2.2 int MX370x__TransducerGetAutoRefreshValues (void * _, struct MX370x__unsignedlong17ArrayResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue :

- 0: success

- -100: GetAutoRefreshAllValues kernel function error

ulValue : Array that contain the counter and channels values

- ulValues [0] : Auto refresh counter value
- ulValues [1] : Channel 0 value
- ...
- ulValues [16] : Channel 15 value

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.19.2.3 int MX370x__TransducerStopAndReleaseAutoRefresh (void * _, struct MX370x__Response * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0 : success
- -1: means an system error occurred
- -100: "StopAutoRefresh" kernel function error

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.20 MX370x Sequence functions

A sequence is a list of channels (max 16) that are acquired.

Functions

- int MX370x__TransducerInitAndStartSequence (xsd__unsignedLong ulTransducerSelection, xsd__unsignedLong ulNbrOfChannel, struct MX370x__unsignedLong16FixedArray *pulChannelList, xsd__unsignedLong ulDivisionFactor, xsd__unsignedLong ulNbrOfSequence, xsd__unsignedLong ulNbrMaxSequenceToTransfer, xsd__unsignedLong ulDelayMode, xsd__unsignedLong ulDelayTimeUnit, xsd__unsignedLong ulDelayValue, xsd__unsignedLong ulTriggerAction, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulHardwareTriggerFilterTime, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, struct MX370x__Response *Response)

  Initialise and start the transducer sequence acquisition mode.

- int MX370x__TransducerStopAndReleaseSequence (void *_, struct MX370x__Response *Response)

  Stop and release the transducer sequence acquisition mode.

2.20.1 Detailed Description

A sequence is a list of channels (max 16) that are acquired. It can be any order of the channels in this list.

There are different sequence modes:

- Certain number of sequences / continuous
- With/Without delay

a) Certain number of sequences:

After the acquisition of the defined number of sequences, the acquisition is stopped automatically.

b) Continuous:

The sequences are acquired continuously until a software-stop-command occurs.

c) Without delay:

There is no waiting time between the acquisitions of 2 sequences.

d) With delay:

A delay between 2 sequences can be configured:

For this there are 2 delay types:

> Mode 1: The delay time defines the time between 2 sequence beginnings.

> Mode 2: The delay time defines the time between the end of a sequence until the beginning of the next sequence.

You can start the acquisition by a hardware trigger in the auto refresh and sequence mode.

The hardware trigger can react to a rising, falling or both edges.

You have the following possibilities:

- Initialising a filter on the trigger input to avoid errors
- Defining a number of edges before a trigger action is generated

There are two trigger modes:

a) One shot

b) Sequence

a) One shot:

After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.

b) Sequence:

After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

There is also the possibility to stop the acquisition with the hardware trigger.

This can only be do when :

- the hardware trigger is not used to start the acquisition.
- the hardware trigger is used in one shot mode to start the acquisition.

In the case of the hardware trigger is used to start the acquisition in one shot mode, and by the stop of the trigger

the number of sequence to acquire is not reach, then the hardware trigger start condition can restart the acquisition !

2.20.2 Function Documentation

2.20.2.1 int MX370x__TransducerInitAndStartSequence (xsd__unsignedLong *ulTransducerSelection,* xsd__unsignedLong *ulNbrOfChannel,* struct MX370x__unsignedLong16FixedArray * *pulChannelList,* xsd__unsignedLong *ulDivisionFactor,* xsd__unsignedLong *ulNbrOfSequence,* xsd__unsignedLong *ulNbrMaxSequenceToTransfer,* xsd__unsignedLong *ulDelayMode,* xsd__unsignedLong *ulDelayTimeUnit,* xsd__unsignedLong *ulDelayValue,* xsd__unsignedLong *ulTriggerAction,* xsd__unsignedLong *ulHardwareTriggerCount,* xsd__unsignedLong *ulHardwareTriggerFilterTime,* xsd__unsignedLong *ulByTriggerNbrOfSeqToAcquire,* xsd__unsignedLong *ulOption1,* xsd__unsignedLong *ulOption2,* xsd__unsignedLong *ulOption3,* xsd__unsignedLong *ulOption4,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerSelection*** : Transducer type selection

[in] ***ulNbrOfChannel*** : Number of channel in the sequence

[in] ***pulChannelList*** : List of the channel index (0 to MaxChannel-1) who compose the sequence. This parameter is an array.

For a sequence that contains two channels (let say channel 0 and channel 1), you will have:

- pulChannelList[0] = 0
- pulChannelList[1] = 1

[in] ***ulDivisionFactor*** : Division factor (min: 5, max: 255)

The division factor sets the switching time from one channel to another (the channels of the system are multiplexed). When the multiplexer switches from one channel to the next one, you need to wait for a certain time (settling time) before acquiring the measurement value of the transducer. If the division factor is too low (< 10), the measurement can be distorted.

The switching time between two channels equals to the product of the division factor and the exciting signal period of the transducer .

Example: If a transducer connected to channel 0 uses a 10 kHz nominal frequency and the division factor is set to 12, the switching time from channel 0 to the next one is: $12 * (1 / 10000) = 1.2$ ms.

[in] ***ulNbrOfSequence*** : Number of sequence to acquire :

- 0 : continuous mode
- > 0 : number of sequence

[in] ***ulNbrMaxSequenceToTransfer*** : This parameter defined the minimal number of sequences to acquired between each send of data by the system.

Warning : They are two possibilities that the number of sequences sent doesn't reach the minimal number:

- By the end of the acquisition.
- If the memory capacity is not big enough.

[in] ***ulDelayMode*** : Delay Mode :

- ADDIDATA_DELAY_NOT_USED 0 : Delay is not used.
- ADDIDATA_DELAY_MODE1_USED 1 : The delay time defines the time between 2 sequence beginnings.
- ADDIDATA_DELAY_MODE2_USED 2 : The delay time defines the time between the end of a sequence until the beginning of the next sequence.

[in] ***ulDelayTimeUnit*** : Selection of the unit of ulDelayValue

- 0: ms
- 1: s

[in] ***ulDelayValue*** : Delay Value (max value: 65535)

[in] ***ulTriggerAction*** : Trigger action :

Hardware Trigger Start D0 - D7

Bit 3,2,1,0 : Define the trigger mode

- 0000 : Trigger disabled
- 0001 : One shot trigger : After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.
- 0010 : Sequence trigger : After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

Bit 7,6 : define the active front (Only if hardware trigger selected)

- 01 : rising front (Only if hardware trigger selected)
- 10 : falling front (Only if hardware trigger selected)
- 11 : Both front (Only if hardware trigger selected)

Synchronisation Trigger Start : D8-D15

Bit 11,10,9,8 : Define the trigger mode

- 0000 : trigger disabled
- 0001 : One shot trigger : After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.
- 0010 : Sequence trigger : After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

Hardware Trigger Stop D16 - D19

The hardware trigger stop can only be activated when :

- The hardware trigger start is not used.
- The hardware trigger start is used in one shot mode.

The stop of the acquisition is really do at the end of a sequence acquisition(to avoid that the acquisition is stop in the middle of a sequence).

Bit 16 : Define the trigger stop is enable or not

- 0 : Stop trigger disabled
- 1 : Stop trigger enabled.

Bit 18,17 : define the active front (Only if hardware trigger stop selected)

- 01 : rising front (Only if hardware trigger stop selected)
- 10 : falling front (Only if hardware trigger stop selected)
- 11 : Both front (Only if hardware trigger stop selected)

Bit 19 : define if the hardware trigger stop use the ulHardwareTriggerCount (Only if hardware trigger stop selected)

- 0 : ulHardwareTriggerCount not used : First hardware trigger stop will stop the acquisition
- 1 : ulHardwareTriggerCount is used : The ulHardwareTriggerCount hardware trigger will stop the acquisition

[in] ***ulHardwareTriggerCount*** : Define the number of trigger events before the trigger action occur

- 0 or 1 : all trigger event start the trigger action
- max value : 65535

[in] ***ulHardwareTriggerFilterTime*** : Filter time for the hardware trigger (= multiplier from 250 ns step)

- max value : 65535

[in] ***ulByTriggerNbrOfSeqToAcquire*** : define the number of sequence to acquire by each trigger event

[in] ***ulOption1*** : Data format option

D0 : Time stamp information

- 0 : no time stamp information
- 1 : timestamp information

D1 : Sequence counter information

- 0 : No sequence counter information
- 1 : Sequence counter information

D2 : Data format

- 0 : Digital value
- 1 : Analog value (in mm)

D3 : invert value

- 0 : don't invert the channel value
- 1 : invert the channel value (-2 mm -> + 2mm)

D4 : receive a relative Time Stamp (first acquisition => time stamp=0) instead of absolute time stamp

- 0 : No relative time stamp information
- 1 : Relative time stamp information

D5 : receive the hardware trigger information

- 0 : no hardware trigger information
- 1 : hardware trigger information

[in] ***ulOption2*** : Reserved

[in] ***ulOption3*** : Reserved

[in] ***ulOption4*** : Reserved

[out] ***Response*** :

iReturnValue :

- 0 : success

- -1: means an system error occurred
- -2: Tranducer selection error
- -3: Number of channel error
- -4: Channel array selection error
- -5: Division factor error
- -6: Incorrect value for Hardware Trigger Mode
- -7: Incorrect value for Hardware Trigger Front
- -8: Incorrect value for Synchro Trigger Mode
- -9: Incorrect value for Hardware Trigger Count
- -10: Incorrect value for Hardware Trigger filter time
- -11: Incorrect value for "trigger number of sequence to acquire"
- -12: Delay Mode selection error
- -13: Delay time unit selection error
- -14: Delay value
- -15: Wrong data format parameter (ulOption1)
- -16: A value for Hardware Trigger front was defined but Hardware Trigger Mode is not set
- -17: Cannot use both triggers at the same time
- -18: Incorrect value for the hardware trigger stop front
- -19: Hardware trigger stop can not be used by this configuration of hardware trigger start
- -100: TransducerInit kernel function error
- -101: InitConvertTimeDivisionFactor kernel function error
- -102: InitEnableDisableSequenceDelay kernel function error
- -103: InitDigitalInputFilter kernel function error
- -104: InitEnableDisableHardwareTrigger kernel function error
- -105: InitEnableSynchroTrigger kernel function error
- -106: DisableSynchroTrigger kernel function error
- -107: SetTriggerSequenceCount kernel function error
- -108: InitSequence kernel function error
- -109: StartStopSequence kernel function error

  syserrno : System-error code (the value of the libc "errno" code)
  Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
  Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.20.2.2 int MX370x__TransducerStopAndReleaseSequence (void * _, struct MX370x__Response * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0: success

- -1: means an system error occurred
- -100: StartStopSequence kernel function error

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.21 MX370x Transducer functions

Functions

- int MX370x__TransducerSetOffset (struct MSXE370x__doubleArrayParam *pdOffsetArray, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, struct MX370x__Response *Response)

  Set / Reset an offset on transducer channels.

- int MX370x__TransducerGetTypeInformation (xsd__unsignedLong ulIndex, struct MX370x__TransducerGetTypeInformationResponse *Response)

  Returns the information stored in the database about the type.

2.21.1 Function Documentation

2.21.1.1 int MX370x__TransducerSetOffset (struct MSXE370x__doubleArrayParam * *pdOffsetArray*, xsd__unsignedLong *ulOption1*, xsd__unsignedLong *ulOption2*, xsd__unsignedLong *ulOption3*, xsd__unsignedLong *ulOption4*, struct MX370x__Response * *Response*)

This function permits to set an offset (reference point) to the measured value.

To disable (reset) a channel offset, set the corresponding channel value to 0.0.

Example: To set a reference point to a transducer in a particular position:

- Reset the offset by setting all channel offset to 0 (pdOffsetArray).
- Run a sequence with the transducer at the position you want to be 0 (reference point). Save the acquired values to put them into pdOffsetArray.
- Stop the acquisition.
- Run MX370x__TransducerSetOffset function to set the offset with the pdOffsetArray previously saved.
- In the next sequence, position will be 0.

Remark : This function cannot be used when an acquisition is running

For more information see SetOffset sample.

Parameters

[in] ***pdOffsetArray*** : array with each offsets for transducers (channel 0 to channel 15)

[in] ***ulOption1*** : Reserved

[in] ***ulOption2*** : Reserved

[in] ***ulOption3*** : Reserved

[in] ***ulOption4*** : Reserved

[out] ***Response*** :

iReturnValue : Error value

- 0: success
- -1: means an system error occurred
- -2: driver status error, acquisition is running
- -100: transducerSetOffset kernel function error
- <> 0: error

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.21.1.2 int MX370x__TransducerGetTypeInformation (xsd__unsignedLong *ulIndex,* struct MX370x__TransducerGetTypeInformationResponse * *Response*)

Parameters

[in] ***ulIndex*** : index of the transducer

[out] ***Response*** :

iReturnValue : Error value

- 0: success
- <> 0: error
- -1: index is invalid
- -100: failure of kernel function "GetTransducerInformation"
- -101: failure of kernel function "GetTransducerType"

ulTransducerSelectionIndex : Selection value. Value to write for the transducer type selection

pcName : Name of the transducer type

ulCalibrationStatus : Calibration status

- 0 : Transducer type is not calibrated
- 1 : Transducer type is calibrated

ulType : Type (0: HB 1: LVDT 2:Knaebel 3:HB-Mahr 4:LVDT-Mahr)

ulFrequency : Frequency (Hz)

ulImpedance : Impedance (Ohm)

dVeff : Nominal voltage (Vrms)

dSensibility : Sensibility (mv/V/mm)

dRange : Range (mm)

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.22 MX370x Min/Max acquisition functions

In the Min/Max-mode an acquisition of certain channels is executed (adjustable by a mask) and the Min-/Max values of each channel are saved.

Data Structures

- struct MX370x__TransducerGetMinMaxStatusResponse

Functions

- int MX370x__TransducerInitAndStartMinMaxAcquisition (xsd__unsignedLong ulTransducerSelection, xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulDivisionFactor, xsd__unsignedLong ulStopChannelMask, xsd__unsignedLong ulStopCondition, xsd__unsignedLong ulStopValue, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, struct MX370x__Response *Response)

  Initialise and start the transducer min/max acquisition.

- int MX370x__TransducerGetMinMaxStatus (void *_, struct MX370x__TransducerGetMinMaxStatusResponse *Response)

  This function get the min/max acquisition status.

- int MX370x__TransducerStopAndReleaseMinMaxAcquisition (void *_, struct MX370x__Response *Response)

  Stop and release the transducer min/max acquisition mode.

2.22.1 Detailed Description

The acquisition runs until a stop-command (synchronisation or hardware; see below) occurs.

In this acquisition mode you have the possibility for a hardware-stop:

A compare-value can be set as well as a condition can be defined for one or more channels.

Example:

If the values of channels 0 are greater than 0x1000, the acquisition is stopped.

In this acquisition mode no values are sent to the data server.

To get the Min-/Max values and the acquisition status, the adequate SOAP function must be used.

2.22.2 Function Documentation

2.22.2.1 int MX370x__TransducerInitAndStartMinMaxAcquisition (xsd__unsignedLong *ulTransducerSelection,* xsd__unsignedLong *ulChannelMask,* xsd__unsignedLong *ulDivisionFactor,* xsd__unsignedLong *ulStopChannelMask,* xsd__unsignedLong *ulStopCondition,* xsd__unsignedLong *ulStopValue,* xsd__unsignedLong *ulOption1,* xsd__unsignedLong *ulOption2,* xsd__unsignedLong *ulOption3,* xsd__unsignedLong *ulOption4,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerSelection*** : Transducer type selection

[in] ***ulChannelMask*** : Mask of the channel for the min/max evaluation (1 bit = 1 channel)

[in] ***ulDivisionFactor*** : Division factor (min: 5, max: 255)

[in] ***ulStopChannelMask*** : Stop channel mask (1 bit = 1 channel)

[in] ***ulStopCondition*** : Define the condition to stop the min/max acquisition :

- 0 : disabled
- 1 : <
- 2 : >

[in] ***ulStopValue*** : Stop value (24 bits : 0 to 0xFFFFFF)

[in] ***ulOption1*** : Reserved

[in] ***ulOption2*** : Reserved

[in] ***ulOption3*** : Reserved

[in] ***ulOption4*** : Reserved

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -2: Transducer selection error
- -3: Channel mask can not be null
- -4: channel mask error
- -5: Division factor error
- -6: Stop condition selection error
- -7: Stop channel mask can not be null
- -8: Stop channel mask error
- -9: Stop value error
- -100: TransducerInit kernel function error
- -101: InitConvertTimeDivisionFactor kernel function error
- -102: InitMinMaxAcquisition kernel function error
- -103: StartStopMinMaxAcquisition kernel function error

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.22.2.2 int MX370x__TransducerGetMinMaxStatus (void * _, struct MX370x__TransducerGetMinMaxStatusResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue :

- 0 : success
- -100: GetMinMaxAcquisitionStatus kernel function error

ulFlag : Min/Max acquisition status :

- 0 : Disable
- 1 : Enable (in progress)
- 2 : End of sequence

ulOverflow : Overflow status

- 0 : No overflow
- 1 : PLD overflow

pulMinValues : Array with the minimale values

pulMaxValues : Array with the maximale values

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.22.2.3 int MX370x__TransducerStopAndReleaseMinMaxAcquisition (void * _, struct MX370x__Response * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: StartStopMinMaxAcquisition kernel function error

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.23 MX370x diagnostic functions

The module MSX-E370x disposes of a diagnostic function which, under certain circumstances, can detect a short circuit or line break on the primary circuit as well as on the secondary circuit.

Functions

- int MX370x__TransducerInitPrimaryConnectionTest (void *_, struct MX370x__Response *Response)

  Initialise the primary connection test.

- int MX370x__TransducerTestPrimaryConnection (void *_, struct MX370x__unsignedlongDefaultResponse *Response)

  Test the primary connection.

- int MX370x__TransducerTestPrimaryShortCircuit (void *_, struct MX370x__unsignedlongDefaultResponse *Response)

  Test primary short circuit status.

- int MX370x__TransducerRearmPrimary (void *_, struct MX370x__unsignedlongDefaultResponse *Response)

  The rearm function permits to switch the outputs on, after the resolution of a primary short-circuit.

- int MX370x__TransducerTestSecondaryConnection (xsd__unsignedLong ulChannel, struct MX370x__unsignedlongDefaultResponse *Response)

  Test the secondary connection For MSX-E370x HB or MSX-E370x LVDT modules, this function test if the secondary line is open or not.

- int MX370x__TransducerTestSecondaryShortCircuit (xsd__unsignedLong ulChannel, struct MX370x__unsignedlongDefaultResponse *Response)

  Test the secondary short circuit status (between transducer measurement signal against mass) of the selected channel
  Important !!
  This function can not be used for the MSX-E370x Mahr modules.

2.23.1 Detailed Description

The short-circuit detection on the primary circuit is activated continuously.

The other diagnostic functions are activated by software functions.

Short-circuit

On the primary circuit the supply voltage of the power buffer is controlled. If a short circuit occurs

(between OSC+ and OSC- or OSC- against mass or OSC+ against mass), a voltage drop is detected.

This information is returned by software diagnostic function.

In case of short circuit the power buffer disposes of internal fuses which switch the outputs off.

On the secondary circuit the status of the channel which caused a short circuit (between transducer measurement signal against mass)

is returned by the software function MX370x__TransducerTestSecondaryShortCircuit.

Important !!

The secondary short circuit can not be tested via the MX370x__TransducerTestSecondaryShortCircuit function for the MSX-E370x Mahr modules.

Use the function MX370x__TransducerTestSecondaryConnection to detect if the transducer are OK or not.

Line break

In case of a line break (OSC+ or OSC-) on the primary circuit, the sofware function controls if at least one of the n connected transducers

is not correctly connected.

To do that, the n transducers must be connected on the module and the MX370x__-TransducerInitPrimaryConnectionTest must be called to saved the status of the input.

This status is then compared to a new status by the call of the MX370x__-TransducerTestPrimaryConnection function.

On the secondary circuit (transducer measurement signal), the status of the channel with a line break is returned by

the software function MX370x__TransducerTestSecondaryConnection.

Important !!

For the MSX-E370x Mahr the function MX370x__TransducerTestSecondaryConnection test if the transducer are OK or not.

2.23.2 Function Documentation

2.23.2.1 int MX370x__TransducerInitPrimaryConnectionTest (void * _, struct MX370x__Response * *Response*)

Can not be used for the MSX-E370x Mahr This function save the number of plugged transducer. This value will then be used when calling the MX370x__TransducerTestPrimaryConnection function. You must call this function at least one time after boot, and then, each time you change the plugged transducer.

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: Primary short circuit occur
- -101: No transducer connected
- -103: Functionality not available

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.23.2.2 int MX370x__TransducerTestPrimaryConnection (void * _, struct MX370x__unsignedlongDefaultResponse * *Response*)

. Can not be used for the MSX-E370x Mahr The saved status input from the MX370x__-TransducerInitPrimaryConnectionTest function is compared to a new status by the call of this function.

Important !!

This function can not be used for the MSX-E370x Mahr modules. Refer you to the MX370x__-TransducerTestSecondaryConnection function.

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: Primary short circuit occur
- -101: No transducers connected
- -102: Test primary connection but no initialisation occur.
- -103: Functionality not available.

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.
ulValue : Connection status:

- 0: connection error
- 1: connection ok

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.23.2.3 int MX370x__TransducerTestPrimaryShortCircuit (void * _, struct MX370x__unsignedlongDefaultResponse * *Response*)

On the primary circuit the supply voltage of the power buffer is controlled. If a short circuit occurs

(between OSC+ and OSC- or OSC- against mass or OSC+ against mass), a voltage drop is detected.

This information is returned by this function.

In case of short circuit the power buffer disposes of internal fuses which switch the outputs off.

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.
ulValue : Short circuit status:

- 0: short circuit
- 1: no short circuit

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.23.2.4 int MX370x__TransducerRearmPrimary (void * _, struct MX370x__unsignedlongDefaultResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.
ulValue : Rearm status:

- 0: Rearm not ok
- 1: Rearm ok

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.23.2.5 int MX370x__TransducerTestSecondaryConnection (xsd__unsignedLong *ulChannel,* struct MX370x__unsignedlongDefaultResponse * *Response*)

For the MSX-E370x Mahr modules, this function test if the connected transducer is OK or not.

Parameters

[in] ***ulChannel,:*** Channel selection (0 to MaxChannel-1)

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: Primary short circuit occur

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.
ulValue : Connection status:

- 0: connection error

- 1: connection ok

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.23.2.6 int MX370x__TransducerTestSecondaryShortCircuit (xsd__unsignedLong *ulChannel,* struct MX370x__unsignedlongDefaultResponse * *Response*)

Refer you to the MX370x__TransducerTestSecondaryConnection function.

Parameters

`[in]` ***ulChannel,:*** Channel selection (0 to MaxChannel-1)

`[out]` ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: Primary short circuit occur
- -101: Functionality not available

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.
ulValue : Short circuit status:

- 0: short circuit
- 1: no short circuit

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.24 MX370x calibration functions

The offset and amplitude error of the MSX-E370x is corrected through digital potentiometers.

Data Structures

- struct MX370x__CalibrationGetCurrentStatusResponse

Functions

- int MX370x__CalibrationStart (xsd__unsignedLong ulTransducerIndex, xsd__unsignedLong ulChannel, xsd__double dPosition, struct MX370x__Response *Response)

  This function start the calibration thread.

- int MX370x__CalibrationStartWithPrimaryConnection (xsd__unsignedLong ulTransducerIndex, xsd__unsignedLong ulChannel, xsd__double dPosition, xsd__unsignedLong ulReserved, struct MX370x__Response *Response)

  This function start the calibration thread with the primary connection line test.

- int MX370x__CalibrationGetCurrentStatus (void *_, struct MX370x__CalibrationGetCurrentStatusResponse *Response)

  This function return the current calibration status.

- int MX370x__CalibrationNextStep (void *_, struct MX370x__Response *Response)

  This function start the next calibration step.

- int MX370x__CalibrationBreak (void *_, struct MX370x__Response *Response)

  This function break the current calibration.

2.24.1 Detailed Description

The user can realize this calibration with the help of the software functions.

After the calibration has been finished, the values of the digital potentiometer are stored

in the flash (by the call of the MX370x__DataBaseSaveTransducers function).

At power up or by transducer selection for an acquisition, the calibration parameters are read from the flash and are sent to the digital potentiometers.

2.24.2 Function Documentation

2.24.2.1 int MX370x__CalibrationStart (xsd__unsignedLong *ulTransducerIndex*, xsd__unsignedLong *ulChannel*, xsd__double *dPosition*, struct MX370x__Response * *Response*)

Parameters

`[in]` ***ulTransducerIndex*** : Selected transducer type to calibrate

`[in]` ***ulChannel*** : Selected the channel to use for the calibration (0 to MaxChannel-1)

`[in]` ***dPosition*** : Selected user calibration position in mm (-transducer range to +transducer range)

`[out]` ***Response*** :

iReturnValue : Error value :

- 0 : means the remote function performed OK
- -1 : means an system error occured

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.24.2.2 int MX370x__CalibrationStartWithPrimaryConnection (xsd__unsignedLong *ulTransducerIndex,* xsd__unsignedLong *ulChannel,* xsd__double *dPosition,* xsd__unsignedLong *ulReserved,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerIndex*** : Selected transducer type to calibrate

[in] ***ulChannel*** : Selected the channel to use for the calibration (0 to MaxChannel-1)

[in] ***dPosition*** : Selected user calibration position in mm (-transducer range to +transducer range)

[in] ***ulReserved*** : Reserved muss be set to 0

[out] ***Response*** :

iReturnValue : Error value :

- 0 : means the remote function performed OK
- -1 : means an system error occured

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.24.2.3 int MX370x__CalibrationGetCurrentStatus (void * _, struct MX370x__CalibrationGetCurrentStatusResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue : Error value

- 0 : No Error
- <> 0 : Error

ulStatus : Status

- 0: No calibration in progress
- 1: Primary calibration in progress
- 2: Wait user access null position setting
- 3: Null position calibration thread in progress
- 4: Wait user access user position setting
- 5: User position calibration thread in progress
- 6: Calibration finiched
- 7: Wait user connect only one transducer for the primary open line diagnostic
- 8: Primary open line diagnostic thread in progress

ulDigitalValue : Last measured digital value

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.24.2.4 int MX370x__CalibrationNextStep (void * _, struct MX370x__Response * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue : Error value :

- 0 : means the remote function performed OK
- -1 : means an system error occured

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.24.2.5 int MX370x__CalibrationBreak (void * _, struct MX370x__Response * *Response*)

The values of the digital potentiometer will be lost.

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue : Error value :

- 0 : means the remote function performed OK
- -1 : means an system error occured

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.25 MX370x transducer database management functions

These functions allows to manage the database of transducer types on the module.

Data Structures

- struct MX370x__DataBaseGetTransducerInformationResponse

Functions

- int MX370x__DataBaseGetNumberOfTransducers (void *_, struct MX370x__-unsignedlongResponse *Response)

  Returns the number of transducer types currently defined in the database.

- int MX370x__DataBaseGetTransducerType (xsd__unsignedLong ulTransducerIndex, struct MX370x__unsignedlongResponse *Response)

  Returns the transducer identifier of the selected transducer.

- int MX370x__DataBaseGetTransducerInformation (xsd__unsignedLong ulTransducerIndex, struct MX370x__DataBaseGetTransducerInformationResponse *Response)

  Returns the information stored in the database about the type.

- int MX370x__DataBaseAddTransducer (xsd__unsignedLong ulTransducerIndex, xsd__string cName, xsd__unsignedLong ulType, xsd__unsignedLong ulFrequency, xsd__unsignedLong ulImpedance, xsd__double dVeff, xsd__double dSensitivity, xsd__double dRange, struct MX370x_-_Response *Response)

  Adds a new transducer type definition into the database of the module.

- int MX370x__DataBaseDelTransducer (xsd__unsignedLong ulTransducerIndex, struct MX370x__-Response *Response)

  Deletes the selected transducer from the transducer database.

- int MX370x__DataBaseSaveTransducers (void *_, struct MX370x__ByteArrayResponse *Response)

  Commits the current changes in the transducer database, including the calibration values.

2.25.1 Function Documentation

2.25.1.1 int MX370x__DataBaseGetNumberOfTransducers (void * _, struct MX370x__unsignedlongResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue : Error code :

- 0 : success
- <> 0 : error
- -100 : kernel function error

ulValue : number of transducer types.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.25.1.2 int MX370x__DataBaseGetTransducerType (xsd__unsignedLong *ulTransducerIndex,* struct MX370x__unsignedlongResponse * *Response*)

Parameters

[in] ***ulTransducerIndex*** : Transducer index (0 to ulTransducerNumbers - 1)

[out] ***Response*** :

iReturnValue : Error code :

- 0 : success
- <> 0 : error
- -100 : kernel function error

ulValue : transducer identifier. Use this value as the "Index" parameter given to DataBaseGetTransducerInformationResponse().

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.25.1.3 int MX370x__DataBaseGetTransducerInformation (xsd__unsignedLong *ulTransducerIndex,* struct MX370x__DataBaseGetTransducerInformationResponse * *Response*)

Parameters

[in] ***ulTransducerIndex*** : transducer identifier, as returned by DataBaseGetTransducerType().

[out] ***Response*** :

iReturnValue : Error code :

- 0 : success
- <> 0 : error
- -100: kernel function error

cName : Name
ulCalibrate : Calibration state (0 : not calibrated 1 : calibrated)
ulType : Type (0: HB 1: LVDT 2:Knaebel 3:HB-Mahr 4:LVDT-Mahr)
ulFrequency : Nominal frequency (Hz)
ulImpedance : Impedance (Ohm)
dVeff : Nominal voltage (Vrms)
dSensitivity : Sensitivity (mV/V/mm)
dRange : Range (mm)

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.25.1.4 int MX370x__DataBaseAddTransducer (xsd__unsignedLong *ulTransducerIndex,* xsd__string *cName,* xsd__unsignedLong *ulType,* xsd__unsignedLong *ulFrequency,* xsd__unsignedLong *ulImpedance,* xsd__double *dVeff,* xsd__double *dSensitivity,* xsd__double *dRange,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerIndex*** : Identifier of the new type, user-defined value in the range 200 .. 255

[in] ***cName*** : Name

[in] ***ulType*** : Type (0: HB 1: LVDT 2:Knaebel 3:HB-Mahr 4:LVDT-Mahr)

[in] ***ulFrequency*** : Nominal frequency (Hz)

[in] ***ulImpedance*** : Impedance (Ohm)

[in] ***dVeff*** : Nominal voltage (Vrms)

[in] ***dSensitivity*** : Sensitivity (mV/V/mm)

[in] ***dRange*** : Range (mm)

[out] ***Response*** :

iReturnValue : Error code :

- 0 : success
- <> 0 : error
- -100: kernel function error
  syserrno : system-error code (the value of the libc "errno" code) Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
  Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

Note

This function returns an error if a transducer with the same identifier already exists in the database.

2.25.1.5 int MX370x__DataBaseDelTransducer (xsd__unsignedLong *ulTransducerIndex,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerIndex*** : identifier, as returned by DataBaseGetTransducerType().

[out] ***Response*** :

iReturnValue : Error value :

- 0 : success
- <> 0 : error
- -100: kernel function error
  syserrno : system-error code (the value of the libc "errno" code) Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
  Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

Note

This function returns an error if the identifier does not map to an existing transducer type.

2.25.1.6 int MX370x__DataBaseSaveTransducers (void * _, struct MX370x__ByteArrayResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** : ***sResponse.iReturnValue :*** Error code

- 0 success
- <> 0 : error

sResponse.syserrno : system-error code (the value of the libc "errno" code) Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

2.26 MX370x External Digital I/O functions

Contain the digital I/O configuration, filter, diagnostic and read/write functions for the external digital I/O.

Data Structures

- struct MX370x__unsignedLongResponse

Modules

- MX370x External Digital I/O information, configuration functions

  Contain the external digital I/O information, configuration functions.

- MX370x External Digital I/O filter functions

  Contain the external digital I/O filter functions.

- MX370x External Digital I/O diagnostic functions

  Contain the external digital I/O diagnostic functions.

- MX370x External Digital I/O read/write functions

  Contain the external digital I/O read/write functions.

2.27 MX370x External Digital I/O information, configuration functions

Contain the external digital I/O information, configuration functions.

Functions

- int MX370x__ExtDigitalIOGetNumberOfChannels (xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Return the number of digital I/O channels.

- int MX370x__ExtDigitalIOGetNumberOfPorts (xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Return the number of digital I/O ports.

- int MX370x__ExtDigitalIOGetNumberOfChannelsPerPort (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Return the number of digital I/O channels for a given port.

- int MX370x__ExtDigitalIOGetPortDirections (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Get the digital I/O direction for the selected port.

2.27.1 Function Documentation

2.27.1.1 int MX370x__ExtDigitalIOGetNumberOfChannels (xsd__unsignedLong *ulOption1*, struct MX370x__unsignedLongResponse * *Response*)

Parameters

`[in]` ***ulOption1*** Reserved. Set to 0

`[out]` ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Number of available digital I/O channels

Return values

0 SOAP_OK

Others See SOAP error

2.27.1.2 int MX370x__ExtDigitalIOGetNumberOfPorts (xsd__unsignedLong *ulOption1*, struct MX370x__unsignedLongResponse * *Response*)

A port is a set of consecutive digital I/O channels which states can be get or set at the same time.

Parameters

`[in]` ***ulOption1*** Reserved. Set to 0

`[out]` ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK

- **-1**: Means an system error occurred
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Number of available digital I/O ports

Return values

0 SOAP_OK

Others See SOAP error

2.27.1.3 int MX370x__ExtDigitalIOGetNumberOfChannelsPerPort (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse ∗ *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)
Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response sResponse.iReturnValue :***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Number of digital I/O channels for the selected port

Return values

0 SOAP_OK

Others See SOAP error

2.27.1.4 int MX370x__ExtDigitalIOGetPortDirections (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse ∗ *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)
Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Digital directions selection. Each bit indicates the direction for one channel.

- D0 : 0: Digital I/O 0 of selected port used as input. 1: Digital I/O 0 of selected port used as output
- ...
- D31 : 0: Digital I/O 31 of selected port used as input. 1: Digital I/O 31 of selected port used as output

Return values

0 SOAP_OK

Others See SOAP error

2.28 MX370x External Digital I/O filter functions

Contain the external digital I/O filter functions.

Data Structures

- struct MX370x__ExtDigitalIOGetInputsFilterConfigurationResponse

Functions

- int MX370x__ExtDigitalIOSetInputsFilterTime (xsd__unsignedLong ulFilterTime, xsd__unsignedLong ulOption1, struct MX370x__Response *Response)

  Set the filter time for the digital inputs.

- int MX370x__ExtDigitalIOEnableDisableInputsFilter (xsd__unsignedLong ulPort, xsd__unsignedLong ulFilter, xsd__unsignedLong ulOption1, struct MX370x__Response *Response)

  Enable/disable the digital inputs filter for the selected port.

- int MX370x__ExtDigitalIOGetInputsFilterConfiguration (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__ExtDigitalIOGetInputsFilterConfigurationResponse *Response)

  Get the digital inputs filter configuration for the selected port.

2.28.1 Function Documentation

2.28.1.1 int MX370x__ExtDigitalIOSetInputsFilterTime (xsd__unsignedLong *ulFilterTime*, xsd__unsignedLong *ulOption1*, struct MX370x__Response * *Response*)

Parameters

`[in]` ***ulFilterTime*** Filter time, maximum 511 (unit 20 micro s) (1 corresponds to 20 micro s, 2 corresponds to 40 micro s, ...)

`[in]` ***ulOption1*** Reserved. Set to 0

`[out]` ***Response iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred

- **-2**: Filter time selection wrong
- **-3**: Error when writing the new filter time
- **-100**: Internal system error occurred. See value of syserrno

syserrno system-error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

2.28.1.2 int MX370x__ExtDigitalIOEnableDisableInputsFilter (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulFilter,* xsd__unsignedLong *ulOption1,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)
Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulFilter*** 1 to enable, 0 to disable

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Filter selection wrong
- **-4**: Error when writing new filter state
- **-100**: Internal system error occurred. See value of syserrno

syserrno system-error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

2.28.1.3 int MX370x__ExtDigitalIOGetInputsFilterConfiguration (xsd_-_unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__ExtDigitalIOGetInputsFilterConfigurationResponse * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)
Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred

- **-2**: Port selection wrong
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulFilterTime Filter time, maximum 511 (unit 20 micro s) (1 corresponds to 20 micro s, 2 corresponds to 40 micro s, ...)

ulFilter 1 filter is enabled, 0 filter is disabled

Return values

0 SOAP_OK

Others See SOAP error

2.29 MX370x External Digital I/O diagnostic functions

Contain the external digital I/O diagnostic functions.

Functions

- int MX370x__ExtDigitalIOTestOutputsShortCircuit (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Get the short circuit status from selected port.

- int MX370x__ExtDigitalIOTestOutputsPowerSupply (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Get the output power supply status from selected port.

2.29.1 Function Documentation

2.29.1.1 int MX370x__ExtDigitalIOTestOutputsShortCircuit (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)

Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Error when getting the diagnosis
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Digital outputs short-circuit state.

- 0: No short-circuit.

- 1: Short-circuit detected

Return values

0 SOAP_OK

Others See SOAP error

2.29.1.2 int MX370x__ExtDigitalIOTestOutputsPowerSupply (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse * *Response*)

The digital outputs need an external power supply. This function checks the state of the power supply. If the power supply is ok, the function returns 0 (in ulValue).

Parameters

`[in]` ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1).

Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

`[in]` ***ulOption1*** Reserved. Set to 0

`[out]` ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Error when getting the diagnosis
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Digital outputs power supply state.

- 0: power supply state is ok.
- 1: no power supply detected on the outputs

Return values

0 SOAP_OK

Others See SOAP error

2.30 MX370x External Digital I/O read/write functions

Contain the external digital I/O read/write functions.

Functions

- int MX370x__ExtDigitalIOReadChannel (xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Read the selected digital I/O channel.

- int MX370x__ExtDigitalIOReadPort (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

Read the selected digital I/O port.

- int MX370x__ExtDigitalIOWriteChannel (xsd__unsignedLong ulChannel, xsd__unsignedLong ulState, xsd__unsignedLong ulOption1, struct MX370x__Response *Response)

  Set to low/high the selected digital output channel.

- int MX370x__ExtDigitalIOWritePort (xsd__unsignedLong ulPort, xsd__unsignedLong ulState, xsd__unsignedLong ulOption1, struct MX370x__Response *Response)

  Write a value to the selected digital I/O port.

2.30.1 Function Documentation

2.30.1.1 int MX370x__ExtDigitalIOReadChannel (xsd__unsignedLong *ulChannel*, xsd__unsignedLong *ulOption1*, struct MX370x__unsignedLongResponse * *Response*)

If selected channel is an output, then this function returns the current output state.

Parameters

[in] ***ulChannel*** Selected digital I/O channel (0 to MX370x__ExtDigitalIOGetNumberOfChannels - 1)

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Channel selection wrong
- **-3**: Error when reading the channel state
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Digital I/O channel state

- 0: Digital I/O channel is low
- 1: Digital I/O channel is high

Return values

0 SOAP_OK

Others See SOAP error

2.30.1.2 int MX370x__ExtDigitalIOReadPort (xsd__unsignedLong *ulPort*, xsd__unsignedLong *ulOption1*, struct MX370x__unsignedLongResponse * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)
Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Error when reading the port state
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Digital I/O state. Each bit represent the state for one digital I/O channel.

- D0 : 0: Digital I/O 0 of selected port is low. 1: Digital I/O 0 of selected port is high
- ...
- D31 : 0: Digital I/O 31 of selected port is low. 1: Digital I/O 31 of selected port is high

Return values

0 SOAP_OK

Others See SOAP error

2.30.1.3 int MX370x__ExtDigitalIOWriteChannel (xsd__unsignedLong *ulChannel*, xsd__unsignedLong *ulState*, xsd__unsignedLong *ulOption1*, struct MX370x__Response * *Response*)

Parameters

[in] ***ulChannel*** Selected digital I/O channel (0 to MX370x__ExtDigitalIOGetNumberOfChannels - 1)

[in] ***ulState*** Digital I/O channel state

- 0: Set the digital I/O output channel to low
- 1: Set the digital I/O output channel to high

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Channel selection wrong or selected channel is an input
- **-3**: State selection wrong
- **-4**: Error when setting digital output
- **-100**: Internal system error occurred. See value of syserrno

syserrno system-error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

2.30.1.4 int MX370x__ExtDigitalIOWritePort (xsd__unsignedLong *ulPort*, xsd__unsignedLong *ulState*, xsd__unsignedLong *ulOption1*, struct MX370x__Response * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)
Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulState*** Digital I/O state. Each bit set the state for one digital I/O channel.

- D0 : 0: Set the digital I/O output channel 0 of the selected port to low. 1: Set the digital I/O output channel 0 of the selected port to high
- ...
- D31 : 0: Set the digital I/O output channel 31 of the selected port to low. 1: Set the digital I/O output channel 31 of the selected port to high

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Any selected digital I/O is not a output channel
- **-4**: Error when setting digital outputs
- **-100**: Internal system error occurred. See value of syserrno

syserrno system-error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

Chapter 3

Data Structure Documentation

3.1 ByteArray Struct Reference

Dynamic Array of byte - encapsulates C-type strings.

Data Fields

- xsd__unsignedByte * __ptr
  pointer of byte
- int __size
  size of the byte array in bytes
- int __offset
  not used

3.1.1 Field Documentation

3.1.1.1 xsd__unsignedByte* ByteArray::__ptr

3.1.1.2 int ByteArray::__size

3.1.1.3 int ByteArray::__offset

3.2 DefaultResponse Struct Reference

Data Fields

- xsd__int iReturnValue
  return value of the call :
- xsd__int syserrno
  system-error code (the value of the libc "errno" code)

3.2.1 Field Documentation

3.2.1.1 xsd__int DefaultResponse::iReturnValue

- 0 means the remote function performed OK
- -1 means a system error occured, the meaning of other values is function dependant and should be defined in the related header

3.2.1.2 xsd__int DefaultResponse::syserrno

3.3 MSXE370x__doubleArrayParam Struct Reference

Data Fields

- xsd__double dOffset [16]
  the meaning of this value is defined in the related header of the function who use this type

3.3.1 Field Documentation

3.3.1.1 xsd__double MSXE370x__doubleArrayParam::dOffset[16]

3.4 MX370x__ByteArrayResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse
  Default return values.
- struct ByteArray sArray
  Dynamic Array of byte - encapsulates C-type strings.

3.4.1 Field Documentation

3.4.1.1 struct DefaultResponse MX370x__ByteArrayResponse::sResponse

3.4.1.2 struct ByteArray MX370x__ByteArrayResponse::sArray

3.5 MX370x__CalibrationGetCurrentStatusResponse Struct Reference

Data Fields

- xsd__int iReturnValue
  return value of the call :

- xsd__unsignedLong ulStatus

  Calibration status :

- xsd__unsignedLong ulDigitalValue

  Last measured digital value.

3.5.1 Field Documentation

3.5.1.1 xsd__int MX370x__CalibrationGetCurrentStatusResponse::iReturnValue

- 0 means the remote function performed OK
- -1 means a system error occured, the meaning of other values is function dependant and should be defined in the related header

3.5.1.2 xsd__unsignedLong MX370x__CalibrationGetCurrentStatusResponse::ulStatus

- 0: No calibration in progress
- 1: Primary calibration in progress
- 2: Wait user access null position setting
- 3: Null position calibration thread in progress
- 4: Wait user access user position setting
- 5: User position calibration thread in progress
- 6: Calibration finiched
- 7: Wait user connect only one transducer for the primary open line diagnostic
- 8: Primary open line diagnostic thread in progress

3.5.1.3 xsd__unsignedLong MX370x__CalibrationGetCurrentStatusResponse::ulDigitalValue

3.6 MX370x__DataBaseGetTransducerInformationResponse Struct Reference

Data Fields

- xsd__int iReturnValue
- xsd__unsignedByte cName [100]

  Name.

- xsd__unsignedLong ulCalibrate

  Calibration state (0 : not calibrated 1 : calibrated).

- xsd__unsignedLong ulType
  Type (0: HB 1: LVDT 2:Knaebel 3:HB-Mahr 4:LVDT-Mahr).
- xsd__unsignedLong ulFrequency
  Nominal frequency (Hz).
- xsd__unsignedLong ulImpedance
  Load impedance (ohm).
- xsd__double dVeff
  Nominal voltage (Vrms).
- xsd__double dSensitivity
  Sensibility (mV/V/mm).
- xsd__double dRange
  Range (mm).

3.6.1 Field Documentation

3.6.1.1 xsd__int MX370x__DataBaseGetTransducerInformationResponse::iReturnValue

3.6.1.2 xsd__unsignedByte MX370x__DataBaseGetTransducerInformationResponse::cName[100]

3.6.1.3 xsd__unsignedLong MX370x__DataBaseGetTransducerInformationResponse::ulCalibrate

3.6.1.4 xsd__unsignedLong MX370x__DataBaseGetTransducerInformationResponse::ulType

3.6.1.5 xsd__unsignedLong MX370x__-DataBaseGetTransducerInformationResponse::ulFrequency

3.6.1.6 xsd__unsignedLong MX370x__-DataBaseGetTransducerInformationResponse::ulImpedance

3.6.1.7 xsd__double MX370x__DataBaseGetTransducerInformationResponse::dVeff

3.6.1.8 xsd__double MX370x__DataBaseGetTransducerInformationResponse::dSensitivity

3.6.1.9 xsd__double MX370x__DataBaseGetTransducerInformationResponse::dRange

3.7 MX370x__ExtDigitalIOGetInputsFilterConfigurationResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse
  Default return values.
- xsd__unsignedLong ulFilterTime

Filter time, maximum 511 (unit 20 micro s) (1 corresponds to 20 micro s, 2 corresponds to 40 micro s, ...).

- xsd__unsignedLong ulFilter

  Digital inputs filter selection.

3.7.1 Field Documentation

3.7.1.1 struct DefaultResponse MX370x__-ExtDigitalIOGetInputsFilterConfigurationResponse::sResponse

3.7.1.2 xsd__unsignedLong MX370x__-ExtDigitalIOGetInputsFilterConfigurationResponse::ulFilterTime

3.7.1.3 xsd__unsignedLong MX370x__-ExtDigitalIOGetInputsFilterConfigurationResponse::ulFilter

1 filter is enabled, 0 filter is disabled

3.8 MX370x__Response Struct Reference

Data Fields

- xsd__int iReturnValue

  return value of the call :

- xsd__int syserrno

  system-error code (the value of the libc "errno" code)

3.8.1 Field Documentation

3.8.1.1 xsd__int MX370x__Response::iReturnValue

- 0 success
- -1 means a system error occurred, the meaning of other values is function dependant and should be defined in the related header

3.8.1.2 xsd__int MX370x__Response::syserrno

3.9 MX370x__TransducerGetMinMaxStatusResponse Struct Reference

Data Fields

- xsd__int iReturnValue

return value of the call :

- xsd__unsignedLong ulFlag
  status of the Min/Max Mode :
- xsd__unsignedLong ulOverFlow
  Overflow status :
- xsd__unsignedLong pulMinValues [16]
  Array with the minimal values.
- xsd__unsignedLong pulMaxValues [16]
  Array with the maximal values.

3.9.1 Field Documentation

3.9.1.1 xsd__int MX370x__TransducerGetMinMaxStatusResponse::iReturnValue

- 0 success
- -1 means a system error occurred, the meaning of other values is function dependant and should be defined in the related header

3.9.1.2 xsd__unsignedLong MX370x__TransducerGetMinMaxStatusResponse::ulFlag

- 0 : Disable
- 1 : Enable (in progress)
- 2 : End of sequence

3.9.1.3 xsd__unsignedLong MX370x__TransducerGetMinMaxStatusResponse::ulOverFlow

- 0 : No overflow
- 1 : PLD overflow

3.9.1.4 xsd__unsignedLong MX370x__TransducerGetMinMaxStatusResponse::pulMinValues[16]

3.9.1.5 xsd__unsignedLong MX370x__TransducerGetMinMaxStatusResponse::pulMaxValues[16]

3.10 MX370x__TransducerGetTypeInformationResponse Struct Reference

Data Fields

- xsd__int iReturnValue

return value of the call :

- xsd__unsignedLong ulTransducerSelectionIndex
  Identifier.
- xsd__unsignedByte pcName [100]
  Name of the transducer type.
- xsd__unsignedLong ulCalibrationStatus
  Calibration status.
- xsd__unsignedLong ulType
  Type (0: HB 1: LVDT 2:Knaebel 3:HB-Mahr 4:LVDT-Mahr).
- xsd__unsignedLong ulFrequency
  Frequency (Hz).
- xsd__unsignedLong ulImpedance
  Impedance (Ohm).
- xsd__double dVeff
  Nominal voltage (Vrms).
- xsd__double dSensibility
  Sensibility (mv/V/mm).
- xsd__double dRange
  Range (mm).

3.10.1 Field Documentation

3.10.1.1 xsd__int MX370x__TransducerGetTypeInformationResponse::iReturnValue

- 0 success
- -1 means a system error occurred, the meaning of other values is function dependant and should be defined in the related header

3.10.1.2 xsd__unsignedLong MX370x__TransducerGetTypeInformationResponse::ulTransducerSelectionIndex

Value to use for the transducer type selection in the other SOAP functions.

3.10.1.3 xsd__unsignedByte MX370x__TransducerGetTypeInformationResponse::pcName[100]

3.10.1.4 xsd__unsignedLong MX370x__TransducerGetTypeInformationResponse::ulCalibrationStatus

- 0 : Transducer type is not calibrated

- 1 : Transducer type is calibrated

3.10.1.5 xsd__unsignedLong MX370x__TransducerGetTypeInformationResponse::ulType

3.10.1.6 xsd__unsignedLong MX370x__TransducerGetTypeInformationResponse::ulFrequency

3.10.1.7 xsd__unsignedLong MX370x__TransducerGetTypeInformationResponse::ulImpedance

3.10.1.8 xsd__double MX370x__TransducerGetTypeInformationResponse::dVeff

3.10.1.9 xsd__double MX370x__TransducerGetTypeInformationResponse::dSensibility

3.10.1.10 xsd__double MX370x__TransducerGetTypeInformationResponse::dRange

3.11 MX370x__unsignedLong16FixedArray Struct Reference

Data Fields

- xsd__int iReturnValue

  return value of the call :

- xsd__unsignedLong ulValue [16]

  the meaning of this value is defined in the related header of the function who use this type

3.11.1 Field Documentation

3.11.1.1 xsd__int MX370x__unsignedLong16FixedArray::iReturnValue

- 0 success
- -1 means a system error occurred, the meaning of other values is function dependant and should be defined in the related header

3.11.1.2 xsd__unsignedLong MX370x__unsignedLong16FixedArray::ulValue[16]

3.12 MX370x__unsignedlong17ArrayResponse Struct Reference

Data Fields

- xsd__int iReturnValue

  return value of the call :

- xsd__unsignedLong ulValue [17]

  the meaning of this value is defined in the related header of the function who use this type

3.12.1 Field Documentation

3.12.1.1 xsd__int MX370x__unsignedlong17ArrayResponse::iReturnValue

- 0 success
- -1 means a system error occurred, the meaning of other values is function dependant and should be defined in the related header

3.12.1.2 xsd__unsignedLong MX370x__unsignedlong17ArrayResponse::ulValue[17]

3.13 MX370x__unsignedlongDefaultResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse

  Default return values.

- xsd__unsignedLong ulValue

  the meaning of this value is defined in the related header of the function who use this type

3.13.1 Field Documentation

3.13.1.1 struct DefaultResponse MX370x__unsignedlongDefaultResponse::sResponse

3.13.1.2 xsd__unsignedLong MX370x__unsignedlongDefaultResponse::ulValue

3.14 MX370x__unsignedlongResponse Struct Reference

Data Fields

- xsd__int iReturnValue

  return value of the call :

- xsd__unsignedLong ulValue

  the meaning of this value is defined in the related header of the function who use this type

3.14.1 Field Documentation

3.14.1.1 xsd__int MX370x__unsignedlongResponse::iReturnValue

- 0 success
- -1 means a system error occurred, the meaning of other values is function dependant and should be defined in the related header

3.14.1.2 xsd__unsignedLong MX370x__unsignedlongResponse::ulValue

3.15 MX370x__unsignedLongResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse
  Default return values.
- xsd__unsignedLong ulValue
  The meaning of this value is defined in the related header of the function who use this type.

3.15.1 Field Documentation

3.15.1.1 struct DefaultResponse MX370x__unsignedLongResponse::sResponse

3.15.1.2 xsd__unsignedLong MX370x__unsignedLongResponse::ulValue

3.16 MXCommon__ByteArrayResponse Struct Reference

Response containing a C-type string.

Data Fields

- struct DefaultResponse sResponse
  Default return values.
- struct ByteArray sArray
  Dynamic Array of byte - encapsulates C-type strings.

3.16.1 Field Documentation

3.16.1.1 struct DefaultResponse MXCommon__ByteArrayResponse::sResponse

3.16.1.2 struct ByteArray MXCommon__ByteArrayResponse::sArray

3.17 MXCommon__FileResponse Struct Reference

Response containing a chunk of a file.

Data Fields

- struct DefaultResponse sResponse
  return values.

- struct ByteArray sArray
  Dynamic Array of byte.
- xsd__unsignedLong ulEOF
  flag indicating end of file.

3.17.1 Field Documentation

3.17.1.1 struct DefaultResponse MXCommon__FileResponse::sResponse

3.17.1.2 struct ByteArray MXCommon__FileResponse::sArray

3.17.1.3 xsd__unsignedLong MXCommon__FileResponse::ulEOF

3.18 MXCommon__GetAutoConfigurationFileResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse
  Default return values.
- struct ByteArray bArray
  Array of byte of the file.
- xsd__unsignedLong ulEOF
  End of file flag.

3.18.1 Field Documentation

3.18.1.1 struct DefaultResponse MXCommon__GetAutoConfigurationFileResponse::sResponse

3.18.1.2 struct ByteArray MXCommon__GetAutoConfigurationFileResponse::bArray

3.18.1.3 xsd__unsignedLong MXCommon__GetAutoConfigurationFileResponse::ulEOF

3.19 MXCommon__GetEthernetLinksStatesResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse
  Default return values.
- struct sGetEthernetLinksStatesPort sPort0
- struct sGetEthernetLinksStatesPort sPort1

3.19.1 Field Documentation

3.19.1.1 struct DefaultResponse MXCommon__GetEthernetLinksStatesResponse::sResponse

3.19.1.2 struct sGetEthernetLinksStatesPort MXCommon__-GetEthernetLinksStatesResponse::sPort0

3.19.1.3 struct sGetEthernetLinksStatesPort MXCommon__-GetEthernetLinksStatesResponse::sPort1

3.20 MXCommon__GetHardwareTriggerFilterTimeResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse
  Default return values.
- xsd__unsignedLong ulFilterTime
  Hardware filter time (step of 250ns).
- xsd__unsignedLong ulInfo01
  Reserved.
- xsd__unsignedLong ulInfo02
  Reserved.

3.20.1 Field Documentation

3.20.1.1 struct DefaultResponse MXCommon__-GetHardwareTriggerFilterTimeResponse::sResponse

3.20.1.2 xsd__unsignedLong MXCommon__-GetHardwareTriggerFilterTimeResponse::ulFilterTime

3.20.1.3 xsd__unsignedLong MXCommon__GetHardwareTriggerFilterTimeResponse::ulInfo01

3.20.1.4 xsd__unsignedLong MXCommon__GetHardwareTriggerFilterTimeResponse::ulInfo02

3.21 MXCommon__GetHardwareTriggerStateResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse
  Default return values.
- xsd__unsignedLong ulState

0 : Trigger input is low / 1 : Trigger input is high

- xsd__unsignedLong ulInfo01

  Reserved.

- xsd__unsignedLong ulInfo02

  Reserved.

3.21.1 Field Documentation

3.21.1.1 struct DefaultResponse MXCommon__GetHardwareTriggerStateResponse::sResponse

3.21.1.2 xsd__unsignedLong MXCommon__GetHardwareTriggerStateResponse::ulState

3.21.1.3 xsd__unsignedLong MXCommon__GetHardwareTriggerStateResponse::ulInfo01

3.21.1.4 xsd__unsignedLong MXCommon__GetHardwareTriggerStateResponse::ulInfo02

3.22 MXCommon__GetModuleTemperatureValueAndStatusResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse

  Default return value.

- xsd__double dTemperatureValue

  Temperature value.

- xsd__unsignedLong ulTemperatureStatus

  Temperature status.

- xsd__unsignedLong ulInfo

  Reserved.

3.22.1 Field Documentation

3.22.1.1 struct DefaultResponse MXCommon__-GetModuleTemperatureValueAndStatusResponse::sResponse

3.22.1.2 xsd__double MXCommon__-GetModuleTemperatureValueAndStatusResponse::dTemperatureValue

3.22.1.3 xsd__unsignedLong MXCommon__-GetModuleTemperatureValueAndStatusResponse::ulTemperatureStatus

3.22.1.4 xsd__unsignedLong MXCommon__-GetModuleTemperatureValueAndStatusResponse::ulInfo

3.23 MXCommon__GetTimeResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse

  Default return values.

- xsd__unsignedLong ulLowTime

  Number of microseconds since the begin of the second.

- xsd__unsignedLong ulHighTime

  Number of seconds since the Epoch (1st January,1970).

3.23.1 Field Documentation

3.23.1.1 struct DefaultResponse MXCommon__GetTimeResponse::sResponse

3.23.1.2 xsd__unsignedLong MXCommon__GetTimeResponse::ulLowTime

3.23.1.3 xsd__unsignedLong MXCommon__GetTimeResponse::ulHighTime

3.24 MXCommon__GetUpTimeResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse

  Default return value.

- xsd__unsignedLong ulUpTime

  Reserved.

3.24.1 Field Documentation

3.24.1.1 struct DefaultResponse MXCommon__GetUpTimeResponse::sResponse

3.24.1.2 xsd__unsignedLong MXCommon__GetUpTimeResponse::ulUpTime

3.25 MXCommon__Response Struct Reference

contains return values

Data Fields

- xsd__int iReturnValue

  return value of the call :
  - *0 success*
  - *-1 a system error occurred, the meaning of other values is function dependent and should be defined in the related header.*

- xsd__int syserrno

  system-error code (the value of the libc "errno" code, see MXCommon__Strerror()).

3.25.1 Field Documentation

3.25.1.1 xsd__int MXCommon__Response::iReturnValue

3.25.1.2 xsd__int MXCommon__Response::syserrno

3.26 MXCommon__TestCustomerIDResponse Struct Reference

Data Fields

- struct DefaultResponse sResponse

  Default return values.

- struct ByteArray bValueArray

  non encrypted value

- struct ByteArray bCryptedValueArray

  encrypted value

3.26.1 Field Documentation

3.26.1.1 struct DefaultResponse MXCommon__TestCustomerIDResponse::sResponse

3.26.1.2 struct ByteArray MXCommon__TestCustomerIDResponse::bValueArray

3.26.1.3 struct ByteArray MXCommon__TestCustomerIDResponse::bCryptedValueArray

3.27 MXCommon__unsignedLongResponse Struct Reference

Response containing a numerical value (ex: return code).

Data Fields

- struct DefaultResponse sResponse

  Default return values.

- xsd__unsignedLong ulValue

  The meaning of this value is defined in the related header of the function who use this type.

3.27.1 Field Documentation

3.27.1.1 struct DefaultResponse MXCommon__unsignedLongResponse::sResponse

3.27.1.2 xsd__unsignedLong MXCommon__unsignedLongResponse::ulValue

3.28 sGetEthernetLinksStatesPort Struct Reference

Data Fields

- xsd__unsignedLong ulState
- xsd__unsignedLong ulSpeed
- xsd__unsignedLong ulDuplex
- xsd__unsignedLong ulInfo1
- xsd__unsignedLong ulInfo2

3.28.1 Field Documentation

3.28.1.1 xsd__unsignedLong sGetEthernetLinksStatesPort::ulState

3.28.1.2 xsd__unsignedLong sGetEthernetLinksStatesPort::ulSpeed

3.28.1.3 xsd__unsignedLong sGetEthernetLinksStatesPort::ulDuplex

3.28.1.4 xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo1

3.28.1.5 xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo2

3.29 UnsignedLongArray Struct Reference

Dynamic Array of unsigned long.

Data Fields

- xsd__unsignedLong * __ptr
  pointer of unsigned Long
- int __size
  size of the unsigned Long array in Bytes
- int __offset
  not used

3.29.1 Field Documentation

3.29.1.1 xsd__unsignedLong* UnsignedLongArray::__ptr

3.29.1.2 int UnsignedLongArray::__size

3.29.1.3 int UnsignedLongArray::__offset

3.30 UnsignedShortArray Struct Reference

Dynamic Array of unsigned short.

Data Fields

- xsd__unsignedShort * __ptr
  pointer of unsigned short
- int __size
  size of the unsigned short array in Bytes

- int __offset
  not used

3.30.1 Field Documentation

3.30.1.1 xsd__unsignedShort* UnsignedShortArray::__ptr

3.30.1.2 int UnsignedShortArray::__size

3.30.1.3 int UnsignedShortArray::__offset

3.31 xsd__base64Binary Struct Reference

Dynamic Array of byte for input use.

Data Fields

- unsigned char * __ptr
  pointer of byte
- int __size
  size of the byte array

3.31.1 Field Documentation

3.31.1.1 unsigned char* xsd__base64Binary::__ptr

3.31.1.2 int xsd__base64Binary::__size

Chapter 4

File Documentation

4.1 MSXE370x_public_doc.h File Reference

Data Structures

- struct xsd__base64Binary

  Dynamic Array of byte for input use.

- struct UnsignedShortArray

  Dynamic Array of unsigned short.

- struct UnsignedLongArray

  Dynamic Array of unsigned long.

- struct ByteArray

  Dynamic Array of byte - encapsulates C-type strings.

- struct DefaultResponse
- struct MXCommon__Response

  contains return values

- struct MXCommon__ByteArrayResponse

  Response containing a C-type string.

- struct MXCommon__FileResponse

  Response containing a chunk of a file.

- struct MXCommon__unsignedLongResponse

  Response containing a numerical value (ex: return code).

- struct sGetEthernetLinksStatesPort
- struct MXCommon__GetEthernetLinksStatesResponse
- struct MXCommon__GetModuleTemperatureValueAndStatusResponse
- struct MXCommon__GetHardwareTriggerFilterTimeResponse
- struct MXCommon__GetHardwareTriggerStateResponse

- struct MXCommon__TestCustomerIDResponse
- struct MXCommon__GetTimeResponse
- struct MXCommon__GetUpTimeResponse
- struct MXCommon__GetAutoConfigurationFileResponse
- struct MX370x__unsignedlongResponse
- struct MX370x__unsignedLong16FixedArray
- struct MX370x__unsignedlong17ArrayResponse
- struct MX370x__Response
- struct MX370x__ByteArrayResponse
- struct MX370x__unsignedlongDefaultResponse
- struct MX370x__TransducerGetTypeInformationResponse
- struct MSXE370x__doubleArrayParam
- struct MX370x__TransducerGetMinMaxStatusResponse
- struct MX370x__CalibrationGetCurrentStatusResponse
- struct MX370x__DataBaseGetTransducerInformationResponse
- struct MX370x__unsignedLongResponse
- struct MX370x__ExtDigitalIOGetInputsFilterConfigurationResponse

Typedefs

- typedef char * xsd__string

  encode xsd__string value as the xsd:string schema type

- typedef char xsd__char

  encode xsd__string value as the xsd:char schema type

- typedef float xsd__float

  encode xsd__float value as the xsd:float schema type

- typedef double xsd__double

  encode xsd__double value as the xsd:double schema type

- typedef int xsd__int

  encode xsd__int value as the xsd:int schema type

- typedef long xsd__long

  encode xsd__long value as the xsd:long schema type

- typedef unsigned char xsd__unsignedByte

  encode xsd__unsignedByte value as the xsd:unsignedByte schema type

- typedef unsigned int xsd__unsignedInt

  encode xsd__unsignedInt value as the xsd:unsignedInt schema type

- typedef unsigned short int xsd__unsignedShort

  encode xsd__unsignedShort value as the xsd:unsignedShort schema type

- typedef unsigned long xsd__unsignedLong

  encode xsd__unsignedLong value as the xsd:unsignedLong schema type

Functions

- int MXCommon__GetModuleType (void *_, struct MXCommon__ByteArrayResponse *Response)

  This function return the type of the MSX-E Module.

- int MXCommon__GetHostname (void *_, struct MXCommon__ByteArrayResponse *Response)

  This function return the hostname of the MSX-E Module.

- int MXCommon__SetHostname (struct xsd__base64Binary *bHostname, struct MXCommon__Response *Response)

  This function allows to set the hostname of the MSX-E Module.

- int MXCommon__GetClientConnections (void *_, struct MXCommon__ByteArrayResponse *Response)

  This function return the client connection list.

- int MXCommon__Strerror (xsd__int errnum, struct MXCommon__ByteArrayResponse *Response)

  Call the libc strerror() on the remote device (actually this is a call to strerror_r()).

- int MXCommon__Reboot (void *_, struct MXCommon__Response *Response)

  Ask the MSX-E module to reboot.

- int MXCommon__ResetAllIOFunctionalities (xsd__unsignedLong ulOption, struct MXCommon__Response *Response)

  Reset the I/O functionalities of the MSX-E system.

- int MXCommon__DataserverRestart (xsd__unsignedLong ulAction, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)

  Restart the data-server service.

- int MXCommon__GetEthernetLinksStates (void *_, struct MXCommon__GetEthernetLinksStatesResponse *Response)

  Get MSX-E Ethernet links states.

- int MXCommon__GetModuleTemperatureValueAndStatus (xsd__unsignedLong ulOption, struct MXCommon__GetModuleTemperatureValueAndStatusResponse *Response)

  Read the temperature on the module.

- int MXCommon__SetModuleTemperatureWarningLevels (xsd__double dMinimalWarningLevel, xsd__double dMaximalWarningLevel, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)

  Set the temperature warning level on the module.

- int MXCommon__SetHardwareTriggerFilterTime (xsd__unsignedLong ulFilterTime, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)

  Sets the filter time for the hardware trigger input in steps of 250 ns (max value: 65535).

- int MXCommon__GetHardwareTriggerFilterTime (xsd__unsignedLong ulOption, struct MXCommon__GetHardwareTriggerFilterTimeResponse *Response)

Get the filter time for the hardware trigger input.

- int MXCommon__GetHardwareTriggerState (xsd__unsignedLong ulOption, struct MXCommon__GetHardwareTriggerStateResponse *Response)

  Get the hardware trigger state after the filter.

- int MXCommon__SetCustomerKey (struct xsd__base64Binary *bKey, struct xsd__base64Binary *bPublicKey, struct MXCommon__Response *Response)

  Set the Customer key.

- int MXCommon__TestCustomerID (void *_, struct MXCommon__TestCustomerIDResponse *Response)

  Test the Customer ID (if the module has the right customer Key).

- int MXCommon__SetTime (xsd__unsignedLong ulLowTime, xsd__unsignedLong ulHighTime, struct MXCommon__Response *Response)

  Set the time on the module.

- int MXCommon__SysToHardwareClock (void *_, struct MXCommon__Response *Response)

  Set the hardware clock (if present) to the current system time.

- int MXCommon__HardwareClockToSys (void *_, struct MXCommon__Response *Response)

  Set the system time from the hardware clock (if present).

- int MXCommon__GetTime (void *_, struct MXCommon__GetTimeResponse *Response)

  Get the time on the module.

- int MXCommon__GetUpTime (void *_, struct MXCommon__GetUpTimeResponse *Response)

  Ask the MSX-E module uptime (number of seconds since the last boot).

- int MXCommon__GetAutoConfigurationFile (void *_, struct MXCommon__GetAutoConfigurationFileResponse *Response)

  Get the auto configuration file of the module.

- int MXCommon__SetAutoConfigurationFile (struct xsd__base64Binary *ByteArrayInput, xsd__unsignedLong ulEOF, struct MXCommon__Response *Response)

  Set the auto configuration file of the module.

- int MXCommon__StartAutoConfiguration (void *_, struct MXCommon__ByteArrayResponse *Response)

  start/Restart the auto configuration

- int MXCommon__InitAndStartSynchroTimer (xsd__unsignedLong ulTimeBase, xsd__unsignedLong ulReloadValue, xsd__unsignedLong ulNbrOfCycle, xsd__unsignedLong ulGenerateTriggerMode, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, xsd__unsignedLong ulOption03, xsd__unsignedLong ulOption04, struct MXCommon__Response *Response)

  Initialises and starts the synchronisation timer of the module (not already available on all module).

- int MXCommon__StopAndReleaseSynchroTimer (xsd__unsignedLong ulOption01, struct MXCommon__Response *Response)

start/Restart the synchronisation timer (not already available on all module)

- int MXCommon__GetConfigurationBackupFile (void *_, struct MXCommon__FileResponse *Response)

  Download a configuration backup file from the module.

- int MXCommon__ApplyConfigurationBackupFile (struct xsd__base64Binary *ByteArrayInput, xsd__unsignedLong ulEOF, struct MXCommon__Response *Response)

  Upload a new configuration on the module.

- int MXCommon__ChangePassword (struct xsd__base64Binary *PreviousUser, struct xsd__base64Binary *PreviousPassword, struct xsd__base64Binary *NewUser, struct xsd__base64Binary *NewPassword, struct MXCommon__Response *Response)

  Set a new id/password.

- int MXCommon__GetSubSystemState (xsd__unsignedLong SubsystemID, struct MXCommon__unsignedLongResponse *Response)

  Returns the current state of the specified sub-system.

- int MXCommon__GetSubsystemIDFromName (struct xsd__base64Binary *SubsystemName, struct MXCommon__unsignedLongResponse *Response)

  Returns the ID of the sub-system of symbolic name "SubsystemName".

- int MXCommon__GetStateIDFromName (xsd__unsignedLong SubsystemID, struct xsd__base64Binary *StateName, struct MXCommon__unsignedLongResponse *Response)

  Returns the ID of the state of symbolic name "StateName" of the sub-system of ID "SubsystemID".

- int MXCommon__GetSubsystemNameFromID (xsd__unsignedLong SubsystemID, struct MXCommon__ByteArrayResponse *Response)

  Returns the symbolic name of the sub-system of numerical ID "SubsystemName".

- int MXCommon__GetStateNameFromID (xsd__unsignedLong SubsystemID, xsd__unsignedLong StateID, struct MXCommon__ByteArrayResponse *Response)

  Returns the symbolic name of the state of numerical ID "StateID" of the sub-system of ID "SubsystemID".

- int MXCommon__GetOptionInformation (void *_, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__ByteArrayResponse *Response)

  Enables to get information about the options available on the system.

- int MXCommon__SetToMaster (void *_, xsd__unsignedLong ulState, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__Response *Response)

  Writes if the MSXE has to be always set to master The master mode (when enabled) make the system always detected as master.

- int MXCommon__GetSynchronizationStatus (void *_, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__unsignedLongResponse *Response)

  Reads the status of the synchronization for the corresponding MSXE The master mode (when enabled) make the system always detected as master.

- int MXCommon__SetFilterChannels (struct xsd__base64Binary *ChannelList, struct MXCommon__Response *Response)

This function sets or resets a filter to a channel.

- int MX370x__TransducerGetNbrOfType (void *_, struct MX370x__unsignedlongResponse *Response)

  Returns the number of transducer types currently defined in the database.

- int MX370x__TransducerInitAndStartAutoRefresh (xsd__unsignedLong ulTransducerSelection, xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulAverageMode, xsd__unsignedLong ulAverageValue, xsd__unsignedLong ulDivisionFactor, xsd__unsignedLong ulTriggerAction, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulHardwareTriggerFilterTime, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, struct MX370x__Response *Response)

  Initialise and start the transducer auto refresh acquisition mode.

- int MX370x__TransducerGetAutoRefreshValues (void *_, struct MX370x__unsignedlong17ArrayResponse *Response)

  This function get the auto refresh counter value an the channels values.

- int MX370x__TransducerStopAndReleaseAutoRefresh (void *_, struct MX370x__Response *Response)

  Stop and release the transducer auto refresh acquisition mode.

- int MX370x__TransducerInitAndStartSequence (xsd__unsignedLong ulTransducerSelection, xsd__unsignedLong ulNbrOfChannel, struct MX370x__unsignedLong16FixedArray *pulChannelList, xsd__unsignedLong ulDivisionFactor, xsd__unsignedLong ulNbrOfSequence, xsd__unsignedLong ulNbrMaxSequenceToTransfer, xsd__unsignedLong ulDelayMode, xsd__unsignedLong ulDelayTimeUnit, xsd__unsignedLong ulDelayValue, xsd__unsignedLong ulTriggerAction, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulHardwareTriggerFilterTime, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, struct MX370x__Response *Response)

  Initialise and start the transducer sequence acquisition mode.

- int MX370x__TransducerStopAndReleaseSequence (void *_, struct MX370x__Response *Response)

  Stop and release the transducer sequence acquisition mode.

- int MX370x__TransducerSetOffset (struct MSXE370x__doubleArrayParam *pdOffsetArray, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, struct MX370x__Response *Response)

  Set / Reset an offset on transducer channels.

- int MX370x__TransducerGetTypeInformation (xsd__unsignedLong ulIndex, struct MX370x__TransducerGetTypeInformationResponse *Response)

  Returns the information stored in the database about the type.

- int MX370x__TransducerInitAndStartMinMaxAcquisition (xsd__unsignedLong ulTransducerSelection, xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulDivisionFactor, xsd__unsignedLong ulStopChannelMask, xsd__unsignedLong ulStopCondition, xsd__unsignedLong ulStopValue, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, struct MX370x__Response *Response)

Initialise and start the transducer min/max acquisition.

- int MX370x__TransducerGetMinMaxStatus (void *_, struct MX370x__TransducerGetMinMaxStatusResponse *Response)

  This function get the min/max acquisition status.

- int MX370x__TransducerStopAndReleaseMinMaxAcquisition (void *_, struct MX370x__Response *Response)

  Stop and release the transducer min/max acquisition mode.

- int MX370x__TransducerInitPrimaryConnectionTest (void *_, struct MX370x__Response *Response)

  Initialise the primary connection test.

- int MX370x__TransducerTestPrimaryConnection (void *_, struct MX370x__unsignedlongDefaultResponse *Response)

  Test the primary connection.

- int MX370x__TransducerTestPrimaryShortCircuit (void *_, struct MX370x__unsignedlongDefaultResponse *Response)

  Test primary short circuit status.

- int MX370x__TransducerRearmPrimary (void *_, struct MX370x__unsignedlongDefaultResponse *Response)

  The rearm function permits to switch the outputs on, after the resolution of a primary short-circuit.

- int MX370x__TransducerTestSecondaryConnection (xsd__unsignedLong ulChannel, struct MX370x__unsignedlongDefaultResponse *Response)

  Test the secondary connection For MSX-E370x HB or MSX-E370x LVDT modules, this function test if the secondary line is open or not.

- int MX370x__TransducerTestSecondaryShortCircuit (xsd__unsignedLong ulChannel, struct MX370x__unsignedlongDefaultResponse *Response)

  Test the secondary short circuit status (between transducer measurement signal against mass) of the selected channel
  Important !!
  This function can not be used for the MSX-E370x Mahr modules.

- int MX370x__CalibrationStart (xsd__unsignedLong ulTransducerIndex, xsd__unsignedLong ulChannel, xsd__double dPosition, struct MX370x__Response *Response)

  This function start the calibration thread.

- int MX370x__CalibrationStartWithPrimaryConnection (xsd__unsignedLong ulTransducerIndex, xsd__unsignedLong ulChannel, xsd__double dPosition, xsd__unsignedLong ulReserved, struct MX370x__Response *Response)

  This function start the calibration thread with the primary connection line test.

- int MX370x__CalibrationGetCurrentStatus (void *_, struct MX370x__CalibrationGetCurrentStatusResponse *Response)

  This function return the current calibration status.

- int MX370x__CalibrationNextStep (void *_, struct MX370x__Response *Response)

This function start the next calibration step.

- int MX370x__CalibrationBreak (void *_, struct MX370x__Response *Response)

  This function break the current calibration.

- int MX370x__DataBaseGetNumberOfTransducers (void *_, struct MX370x__unsignedlongResponse *Response)

  Returns the number of transducer types currently defined in the database.

- int MX370x__DataBaseGetTransducerType (xsd__unsignedLong ulTransducerIndex, struct MX370x__unsignedlongResponse *Response)

  Returns the transducer identifier of the selected transducer.

- int MX370x__DataBaseGetTransducerInformation (xsd__unsignedLong ulTransducerIndex, struct MX370x__DataBaseGetTransducerInformationResponse *Response)

  Returns the information stored in the database about the type.

- int MX370x__DataBaseAddTransducer (xsd__unsignedLong ulTransducerIndex, xsd__string cName, xsd__unsignedLong ulType, xsd__unsignedLong ulFrequency, xsd__unsignedLong ulImpedance, xsd__double dVeff, xsd__double dSensitivity, xsd__double dRange, struct MX370x__Response *Response)

  Adds a new transducer type definition into the database of the module.

- int MX370x__DataBaseDelTransducer (xsd__unsignedLong ulTransducerIndex, struct MX370x__Response *Response)

  Deletes the selected transducer from the transducer database.

- int MX370x__DataBaseSaveTransducers (void *_, struct MX370x__ByteArrayResponse *Response)

  Commits the current changes in the transducer database, including the calibration values.

- int MX370x__ExtDigitalIOGetNumberOfChannels (xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Return the number of digital I/O channels.

- int MX370x__ExtDigitalIOGetNumberOfPorts (xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Return the number of digital I/O ports.

- int MX370x__ExtDigitalIOGetNumberOfChannelsPerPort (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Return the number of digital I/O channels for a given port.

- int MX370x__ExtDigitalIOGetPortDirections (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Get the digital I/O direction for the selected port.

- int MX370x__ExtDigitalIOSetInputsFilterTime (xsd__unsignedLong ulFilterTime, xsd__unsignedLong ulOption1, struct MX370x__Response *Response)

  Set the filter time for the digital inputs.

- int MX370x__ExtDigitalIOEnableDisableInputsFilter (xsd__unsignedLong ulPort, xsd__unsignedLong ulFilter, xsd__unsignedLong ulOption1, struct MX370x__Response *Response)

  Enable/disable the digital inputs filter for the selected port.

- int MX370x__ExtDigitalIOGetInputsFilterConfiguration (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__ExtDigitalIOGetInputsFilterConfigurationResponse *Response)

  Get the digital inputs filter configuration for the selected port.

- int MX370x__ExtDigitalIOTestOutputsShortCircuit (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Get the short circuit status from selected port.

- int MX370x__ExtDigitalIOTestOutputsPowerSupply (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Get the output power supply status from selected port.

- int MX370x__ExtDigitalIOReadChannel (xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Read the selected digital I/O channel.

- int MX370x__ExtDigitalIOReadPort (xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MX370x__unsignedLongResponse *Response)

  Read the selected digital I/O port.

- int MX370x__ExtDigitalIOWriteChannel (xsd__unsignedLong ulChannel, xsd__unsignedLong ulState, xsd__unsignedLong ulOption1, struct MX370x__Response *Response)

  Set to low/high the selected digital output channel.

- int MX370x__ExtDigitalIOWritePort (xsd__unsignedLong ulPort, xsd__unsignedLong ulState, xsd__unsignedLong ulOption1, struct MX370x__Response *Response)

  Write a value to the selected digital I/O port.

4.1.1 Typedef Documentation

4.1.1.1 typedef char* xsd__string

4.1.1.2 typedef char xsd__char

4.1.1.3 typedef float xsd__float

4.1.1.4 typedef double xsd__double

4.1.1.5 typedef int xsd__int

4.1.1.6 typedef long xsd__long

4.1.1.7 typedef unsigned char xsd__unsignedByte

4.1.1.8 typedef unsigned int xsd__unsignedInt

4.1.1.9 typedef unsigned short int xsd__unsignedShort

4.1.1.10 typedef unsigned long xsd__unsignedLong

4.1.2 Function Documentation

4.1.2.1 int MXCommon__GetModuleType (void * _, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** • sArray : Module type string
- sResponse Composed of iReturnValue and syserrno

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.2 int MXCommon__GetHostname (void * _, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** • sArray : Hostname of the module
- iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- syserrno : System-error code. The value of the libc "errno" code, see MXCommon__-Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.3 int MXCommon__SetHostname (struct xsd__base64Binary * *bHostname*, struct MXCommon__Response * *Response*)

Parameters

`[in]` ***bHostname*** : Hostname

`[out]` ***Response*** • iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

• syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.4 int MXCommon__GetClientConnections (void * _, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** • sArray : string containing the list of connected clients.
• sResponse Composed of iReturnValue and syserrno

The sArray string is of the form IP-Address:first connection-second connection---- IP-Address:first connection-second connection----

Sample: 172.16.3.43:8989-5555 172.16.3.200:8989

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.5 int MXCommon__Strerror (xsd__int *errnum*, struct MXCommon__ByteArrayResponse * *Response*)

Usually SOAP functions return this value in a variable named syserror, which is meaningful only when the function return value, usually called iReturnValue, indicate an error (that is, have a value of -1 or -100, depending of the case).

Parameters

`[in]` ***errnum*** : Error number

[out] ***Response*** • sArray : See the description below.
- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno).
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

```
STRERROR(3)                                                  Linux Programmer's Manual
STRERROR(3)

NAME
strerror, strerror_r - return string describing error code

SYNOPSIS
#include <string.h>

char *strerror(int errnum);

#define _XOPEN_SOURCE 600
#include <string.h>

int strerror_r(int errnum, char *buf, size_t n);

DESCRIPTION
The  strerror() function returns a string describing the error code passed
in the argument errnum,possibly using the LC_MESSAGES part of the current
locale to select the appropriate language.
This string must not be modified by the application, but may be modified
by a subsequent call to perror() or strerror().  No library  function will
modify this string.

The strerror_r() function is similar to strerror(), but is thread safe.
It returns the string in the user-supplied buffer buf of length n.

RETURN VALUE
The strerror() function returns the appropriate error description string,
or an unknown error message if the error code is unknown.
The value of errno is not changed for a successful call, and is set to a non-zero
value upon error.
The strerror_r() function returns 0 on success and -1 on failure, setting errno.

ERRORS
EINVAL The value of errnum is not a valid error number.

ERANGE Insufficient storage was supplied to contain the error description string.

CONFORMING TO
SVID 3, POSIX, 4.3BSD, ISO/IEC 9899:1990 (C89).
strerror_r() with prototype as given above is specified by SUSv3,
and was in use under Digital Unix and HP Unix. An incompatible function,
with prototype

char *strerror_r(int errnum, char *buf, size_t n);

is a GNU extension used by glibc (since 2.0), and must be regarded as obsolete
in view of SUSv3.
The GNU version may, but need not, use the user-supplied buffer.
If it does, the result may be truncated in case the supplied buffer is too small.
The result is always NUL-terminated.

SEE ALSO
errno(3), perror(3), strsignal(3)
```

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.6 int MXCommon__Reboot (void * _, struct MXCommon__Response * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** • iReturnValue : Return value
  – 0 : success
  – -1: system error (see syserrno)
• syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.7 int MXCommon__ResetAllIOFunctionalities (xsd__unsignedLong *ulOption*, struct MXCommon__Response * *Response*)

The behavior of the function depends on the MSX-E system that is used.

```
On MSX-E3511: Stop the watchdogs and stop the generators
On MSX-E3601: Stop the sequence acquisition and stop the calibration
On MSX-E3701: Stop the acquisition
```

Parameters

`[in]` ***ulOption*** Reserved. Set to 0

`[out]` ***Response iReturnValue***

- **0** The remote function performed OK
- **-1** Internal system error occurred. See value of syserrno
- **-100** Function not supported by the system

syserrno system error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.8 int MXCommon__DataserverRestart (xsd__unsignedLong *ulAction*, xsd__unsignedLong *ulOption*, struct MXCommon__Response * *Response*)

Parameters

`[in]` ***ulAction*** : action

- 0: normal restart
- 1: with cache file reset

- 2: with cache file deletion

[in] ***ulOption*** : Reserved

[out] ***Response*** • iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- syserrno : System-error code. The value of the libc "errno" code, see MXCommon__-Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

Note

(revision>6386) Depending on the system type, can be used to restart the data-recv service as well. In this case, parameter action is ignored.

4.1.2.9 int MXCommon__GetEthernetLinksStates (void * _, struct MXCommon__GetEthernetLinksStatesResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** Structure that contains the MSX-E Ethernet links states and errors:

sResponse.iReturnValue

- **0** The remote function performed OK
- **-1** System error occurred
- **-2** Fail to get Ethernet links states
- **-100** Internal system error occurred. See value of syserrno

sResponse.syserrno system error code (the value of the libc "errno" code)

sPort0: Fisrt port informations

- **ulState**
  - **0** Link down
  - **1** Link up
- **ulSpeed**
  - **10** 10 Mb/s
  - **100** 100 Mb/s
- **ulDuplex**
  - **0** Half duplex
  - **1** Full duplex
- **ulInfo1** Reserverd
- **ulInfo2** Reserverd

sPort1: Second port informations

- **ulState**
  - **0** Link down
  - **1** Link up
- **ulSpeed**

  - **10** 10 Mb/s
  - **100** 100 Mb/s
- **ulDuplex**
  - **0** Half duplex
  - **1** Full duplex
- **ulInfo1** Reserverd
- **ulInfo2** Reserverd

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.10 int MXCommon__GetModuleTemperatureValueAndStatus (xsd__unsignedLong *ulOption*, struct MXCommon__GetModuleTemperatureValueAndStatusResponse * *Response*)

Parameters

[in] ***ulOption*** : Reserved

[out] ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
  - dValue : Temperature value in Degree Celsius
- ulTemperatureStatus : Temperature Status :
  - TEMPERATURE_INITIAL = 0 : Temperature not ready
  - TEMPERATURE_TOOLOW = 1 : Temperature too low !
  - TEMPERATURE_LOW = 2 : Temperature under the min warning value
  - TEMPERATURE_NOMINAL = 3 : Temperature in the nominal range
  - TEMPERATURE_HIGH = 4 : Temperature over the max warning value
  - TEMPERATURE_TOOHIGH = 5 : Temperature too high !

- ulInfo : Reserved

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.11 int MXCommon__SetModuleTemperatureWarningLevels (xsd__double *dMinimalWarningLevel*, xsd__double *dMaximalWarningLevel*, xsd__unsignedLong *ulOption*, struct MXCommon__Response * *Response*)

Parameters

[in] ***dMinimalWarningLevel*** : Minimal temperature warning level in Degree : 5 to 60 Degree Celsius

[in] ***dMaximalWarningLevel*** : Maximal temperature warning level in Degree : 5 to 60 Degree Celsius

[in] ***ulOption*** : Reserved

[out] ***Response*** • sResponse.iReturnValue : Return value
  – 0 : success
  – -1: system error (see syserrno)
• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.12 int MXCommon__SetHardwareTriggerFilterTime (xsd__unsignedLong *ulFilterTime,* xsd__unsignedLong *ulOption,* struct MXCommon__Response * *Response*)

Sets the filter time for the hardware trigger input in steps of 250 ns (max value: 65535).

On the MSX-E3011 system, the step of the hardware trigger filter is **622ns**.

Parameters

[in] ***ulFilterTime*** Filter time for the hardware trigger input in steps of 250ns (max value : 65535).
- **0**: Disable the filter
- **1**: Sets the filter time to 250 ns
- **2**: Sets the filter time to 500 ns
- ...
- **65535**: Sets the filter time to 16 ms

[in] ***ulOption*** Reserved. Set to 0

[out] ***Response*** Response of the system
- ***sResponse.iReturnValue***
  – **0**: The remote function performed OK
  – **-1**: Internal system error occurred. See value of syserrno
- ***sResponse.syserrno*** system error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.13 int MXCommon__GetHardwareTriggerFilterTime (xsd__unsignedLong *ulOption,* struct MXCommon__GetHardwareTriggerFilterTimeResponse * *Response*)

Get the filter time for the hardware trigger input in **250ns** step (max value : 65535).

On the MSX-E3011 system, the step of the hardware trigger filter is **622ns**.

Parameters

[in] ***ulOption*** Reserved. Set to 0

[out] ***Response*** Response of the system

- ***ulFilterTime*** filter time for the hardware trigger input
  - **0**: filter disabled
  - **1**: filter of 250ns
  - **2**: filter of 500ns
  - ...
  - **65535**: filter of 16ms
- ***sResponse.iReturnValue***
  - **0**: The remote function performed OK
  - **-1**: Internal system error occurred. See value of syserrno
- ***sResponse.syserrno*** system error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.14 int MXCommon__GetHardwareTriggerState (xsd__unsignedLong *ulOption*, struct MXCommon__GetHardwareTriggerStateResponse * *Response*)

Parameters

[in] ***ulOption*** : Reserved

[out] ***Response*** • ulState : Hardware trigger input state.
  - 0: Hardware trigger input is low
  - 1: Hardware trigger input is high.
- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.15 int MXCommon__SetCustomerKey (struct xsd__base64Binary * *bKey*, struct xsd__base64Binary * *bPublicKey*, struct MXCommon__Response * *Response*)

Parameters

[in] ***bKey*** : Customer key (only writable on the module) [32 bytes containing a AES key]

[in] ***bPublicKey*** : IV (Initialisation vector) for the AES cryptography [16 bytes containing a AES key]

[out] ***Response*** • sResponse.iReturnValue : Return value

  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.16 int MXCommon__TestCustomerID (void * _, struct MXCommon__TestCustomerIDResponse * *Response*)

Parameters

`[in]` _ : No Input

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- bValueArray : non encrypted value array [16 bytes of random data]
- bCryptedValueArray : Encrypted value array [16 bytes of the encrypted random data]

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.17 int MXCommon__SetTime (xsd__unsignedLong *ulLowTime,* xsd__unsignedLong *ulHighTime,* struct MXCommon__Response * *Response*)

Parameters

`[in]` ***ulLowTime*** : Number of microseconds since the begin of the second

`[in]` ***ulHighTime*** : Number of seconds since the Epoch (1st January,1970)

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.18 int MXCommon__SysToHardwareClock (void * _, struct MXCommon__Response * *Response*)

Parameters

[in] _ No input parameter

[out] ***Response*** • sResponse.iReturnValue : Return value
  – 0 : success
  – -1: system error (see syserrno)
  • sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

If this function fails, it means the module does not have a hardware RTC, or the hardware is not functional. Check the "hwclock" subsystem status.

4.1.2.19 int MXCommon__HardwareClockToSys (void * _, struct MXCommon__Response * *Response*)

When the hardware clock is present, the system time is automatically set to it when the module becomes master on the inter-module synchronisation bus.

Parameters

[in] _ No input parameter

[out] ***Response*** • sResponse.iReturnValue : Return value
  – 0 : success
  – -1: system error (see syserrno)
  • sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

If this function fails, it means the module does not have a hardware RTC, or the hardware is not functional. Check the "hwclock" subsystem status.

4.1.2.20 int MXCommon__GetTime (void * _, struct MXCommon__GetTimeResponse * *Response*)

Parameters

[in] _ : No input parameter

[out] ***Response*** • sResponse.iReturnValue : Return value
  – 0 : success

  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- ulLowTime : Number of microseconds since the begin of the second
- ulHighTime : Number of seconds since the Epoch (1st January,1970)

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.21 int MXCommon__GetUpTime (void * _, struct MXCommon__GetUpTimeResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- ulUpTime : Number of seconds since the last boot of the system.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.22 int MXCommon__GetAutoConfigurationFile (void * _, struct MXCommon__GetAutoConfigurationFileResponse * *Response*)

Parameters

`[in]` _ : No input parameter

`[out]` ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
  - -100 : Error of the read of the auto configuration file
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- bArray : Array of Bytes of the file
- ulEOF : End of file flag

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.23 int MXCommon__SetAutoConfigurationFile (struct xsd__base64Binary * *ByteArrayInput,* xsd__unsignedLong *ulEOF,* struct MXCommon__Response * *Response*)

Parameters

[in] ***ByteArrayInput*** : Array of Bytes of the file

[in] ***ulEOF*** : End of file flag

[out] ***Response*** • sResponse.iReturnValue : Return value
- 0 : success
- -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.24 int MXCommon__StartAutoConfiguration (void * _, struct MXCommon__ByteArrayResponse * *Response*)

Parameters

[in] _ : No input parameter

[out] ***Response*** • sResponse.iReturnValue : Return value
- 0 : success
- -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- sArray : message returned by the auto configuration start

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.25 int MXCommon__InitAndStartSynchroTimer (xsd__unsignedLong *ulTimeBase,* xsd__unsignedLong *ulReloadValue,* xsd__unsignedLong *ulNbrOfCycle,* xsd__unsignedLong *ulGenerateTriggerMode,* xsd__unsignedLong *ulOption01,* xsd__unsignedLong *ulOption02,* xsd__unsignedLong *ulOption03,* xsd__unsignedLong *ulOption04,* struct MXCommon__Response * *Response*)

Parameters

[in] ***ulTimeBase*** : Time base of the timer (0 for us, 1 for ms, 2 for s)

[in] ***ulReloadValue*** : Timer reload value (0 to 0xFFFF), minimum reload time is 5 us

[in] ***ulNbrOfCycle*** : Number of timer cycle
- 0: continuous
- > 0: defined number of cycle

[in] ***ulGenerateTriggerMode*** :
- 0: Wait the time overflow to set the synchronisation trigger
- 1: Set the synchronisation trigger by the start of the timer and after each time overflow

[in] ***ulOption01*** : Define the source of the trigger
- 0 : Trigger disabled
- 1 : Enable the hardware digital input trigger

[in] ***ulOption02*** : Define the edge of the hardware trigger who generates a trigger action
- 1 : rising edge (Only if hardware trigger selected)
- 2 : falling edge (Only if hardware trigger selected)
- 3 : Both front (Only if hardware trigger selected)

[in] ***ulOption03*** : Define the number of trigger events before the action occur
- 1 : all trigger event start the action
- max value : 65535

[in] ***ulOption04*** : Reserved

[out] ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
  - -2: not available time base
  - -3: timer reload value can not be greater than 65535
  - -4: minimum time reload is 5 us
  - -5: Number of cycle can not be greater than 65535
  - -6: Generate trigger mode error
  - -100: Init timer error
  - -101: Start timer error
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror(). May be ENOSYS : Function not implemented.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.26 int MXCommon__StopAndReleaseSynchroTimer (xsd__unsignedLong *ulOption01*, struct MXCommon__Response * *Response*)

Parameters

[in] ***ulOption01*** : Reserved

[out] ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
  - -100: Start/Stop timer error
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror(). May be ENOSYS : Function not implemented.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.27 int MXCommon__GetConfigurationBackupFile (void * _, struct MXCommon__FileResponse * *Response*)

Parameters

[in] _ : No input parameter

[out] ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno) (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- bArray : Array of Bytes of the file
- ulEOF : End of file flag

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

This function is designed to be called repeatedly until no more data is available. At this point the flag ulEOF is set.

Below is an example in pseudo-C.

```
int dummy;
struct MXCommon__FileResponse Response;
while(1)
{
if ( MXCommon__GetConfigurationBackupFile(&dummy, &Response) != SOAP_OK)
{
// handle soap error
}
if (Response.iReturnValue)
{
// handle remote error (Response.syserrno contains more information)
}
// do something with the data, for example save it in a file
write(fd, Response.bArray.__ptr, Response.bArray.__size);
// if this is the end of the file, quit the loop
if(Response.ulEOF)
break;
}
*
```

4.1.2.28 int MXCommon__ApplyConfigurationBackupFile (struct xsd__base64Binary * *ByteArrayInput*, xsd__unsignedLong *ulEOF*, struct MXCommon__Response * *Response*)

Parameters

[in] ***ByteArrayInput*** : Array of Bytes of the file

[in] ***ulEOF*** : End of file flag

[out] ***Response*** • sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)

- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

This function is designed to be called repeatedly until all data is transfered. At this point the flag ulEOF must be set to 1. The new configuration is then applied.

4.1.2.29 int MXCommon__ChangePassword (struct xsd__base64Binary * *PreviousUser*, struct xsd__base64Binary * *PreviousPassword*, struct xsd__base64Binary * *NewUser*, struct xsd__base64Binary * *NewPassword*, struct MXCommon__Response * *Response*)

The changes are immediately active.

Parameters

`[in]` _ : No input parameter

`[out]` ***Response***
- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: string PreviousUser is invalid
  - -2: string PreviousPassword is invalid
  - -3: string NewUser is invalid
  - -4: string NewPassword is invalid
  - -5: authentification failed
  - -100: system error while saving tokens (use syserrno for more information)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- sArray : message returned by the auto configuration start

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

Warning

The parameters transit in clear text. Use this functionality only on trusted networks.
Given that ADDI-DATA GmbH takes security seriously, there is no way to change the password without knowing it. No "hidden back-door". This function makes it all too easy to lock a module, if you don't remember the password you set on it.

4.1.2.30 int MXCommon__GetSubSystemState (xsd__unsignedLong *SubsystemID*, struct MXCommon__unsignedLongResponse * *Response*)

Parameters

`[in]` ***SubsystemID*** sub-system numerical ID

`[out]` ***Response***
- sResponse.iReturnValue : Return value

  - 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameter SubsystemID
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- Value The state of the sub-system "Id" at the moment of the execution of the request.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.31 int MXCommon__GetSubsystemIDFromName (struct xsd__base64Binary * *SubsystemName,* struct MXCommon__unsignedLongResponse * *Response*)

Parameters

`[in]` ***SubsystemName*** sub-system symbolic name.

`[out]` ***Response***
- sResponse.iReturnValue :Return value
  - 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameter SubsystemName
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- Value The numerical ID of the sub-system "SubsystemName".

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.32 int MXCommon__GetStateIDFromName (xsd__unsignedLong *SubsystemID,* struct xsd__base64Binary * *StateName,* struct MXCommon__unsignedLongResponse * *Response*)

Parameters

`[in]` ***SubsystemID*** sub-system numerical ID

`[in]` ***StateName*** state symbolic name.

`[out]` ***Response***
- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameters SubsystemID or StateName
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().
- Value The numerical ID of the state "StateName".

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.33 int MXCommon__GetSubsystemNameFromID (xsd__unsignedLong *SubsystemID,* struct MXCommon__ByteArrayResponse * *Response*)

Parameters

[in] ***SubsystemID*** sub-system numerical ID.

[out] ***Response*** • sResponse.iReturnValue : Return value
- 0 : success
- -1: system error while executing the request (see syserrno)
- -2: invalid parameter SubsystemName

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

• sArray : The symbolic name associated with the ID.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.34 int MXCommon__GetStateNameFromID (xsd__unsignedLong *SubsystemID,* xsd__unsignedLong *StateID,* struct MXCommon__ByteArrayResponse * *Response*)

Parameters

[in] ***SubsystemID*** sub-system numerical ID.

[in] ***StateID*** sub-system numerical ID.

[out] ***Response*** • sResponse.iReturnValue : Return value
- 0 success
- -1 system error while executing the request (see syserrno)
- -2 invalid parameters SubsystemID or StateID

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

• sArray The symbolic name associated with the state numerical ID.

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.35 int MXCommon__GetOptionInformation (void * _, xsd__unsignedLong *ulOption01,* xsd__unsignedLong *ulOption02,* struct MXCommon__ByteArrayResponse * *Response*)

Parameters

[in] ***ulOption01,:*** not used, set it to 0

[in] ***ulOption02,:*** not used, set it to 0

[out] ***Response*** • sArray : Option information string

• sResponse Composed of iReturnValue and syserrno

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.36 int MXCommon__SetToMaster (void * _, xsd__unsignedLong *ulState*, xsd__unsignedLong *ulOption01*, xsd__unsignedLong *ulOption02*, struct MXCommon__Response * *Response*)

Parameters

[in] ***ulState*** State of the supermaster mode

- **0** automatic mode (default). The state of the system (master or slave) will be automatically detected by the system
- **1** Set to master mode at all time. The system will always be detected as master

[in] ***ulOption01*** Reserved. Set to 0

[in] ***ulOption02*** Reserved. Set to 0

[out] ***Response iReturnValue***

- **0** The remote function performed OK
- **-1** System error occurred
- **-2** The PLD is not working
- **-3** The ulFilterTime parameter is wrong
- **-100** Internal system error occurred. See value of syserrno ***syserrno*** system error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.37 int MXCommon__GetSynchronizationStatus (void * _, xsd__unsignedLong *ulOption01*, xsd__unsignedLong *ulOption02*, struct MXCommon__unsignedLongResponse * *Response*)

Parameters

[in] ***ulOption01*** Reserved. Set to 0

[in] ***ulOption02*** Reserved. Set to 0

[out] ***Response sResponse.iReturnValue***

- **0** The remote function performed OK
- **-1** System error occurred
- **-2** The PLD is not working
- **-100** Internal system error occurred. See value of syserrno

sResponse.syserrno system error code (the value of the libc "errno" code)

ulValue State of the supermaster mode

- **0** Automatic mode (default). The state of the system (master or slave) will be automatically detected by the system
- **1** MSXE is always set as a master. The system will always be detected as master

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.38 int MXCommon__SetFilterChannels (struct xsd__base64Binary * *ChannelList*, struct MXCommon__Response * *Response*)

Parameters

[in] ***ChannelList*** Each index of the array represents a channel. A filter can be affected to each channel. If FilterID = 0, no filter is set (the filter is disabled on the corresponding channel). e.g.: ChannelList[0] = FilterID // Set FilterID on channel 0.

[out] ***Response*** • sResponse.iReturnValue : Return value
- 0 : success
- -1: system error (see syserrno)

• sResponse.syserrno : System-error code. The value of the libc "errno" code, see MXCommon__Strerror().

Return values

SOAP_OK SOAP call success

otherwise SOAP protocol error

4.1.2.39 int MX370x__TransducerGetNbrOfType (void * _, struct MX370x__unsignedlongResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue : Error value
- 0: success
- <> 0: error
- -100: kernel function error

ulValue : number of transducers type.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.40 int MX370x__TransducerInitAndStartAutoRefresh (xsd__unsignedLong *ulTransducerSelection*, xsd__unsignedLong *ulChannelMask*, xsd__unsignedLong *ulAverageMode*, xsd__unsignedLong *ulAverageValue*, xsd__unsignedLong *ulDivisionFactor*, xsd__unsignedLong *ulTriggerAction*, xsd__unsignedLong *ulHardwareTriggerCount*, xsd__unsignedLong *ulHardwareTriggerFilterTime*, xsd__unsignedLong *ulByTriggerNbrOfSeqToAcquire*, xsd__unsignedLong *ulOption1*, xsd__unsignedLong *ulOption2*, xsd__unsignedLong *ulOption3*, xsd__unsignedLong *ulOption4*, struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerSelection*** : Transducer type selection

[in] ***ulChannelMask*** : Mask of the channel to acquire by the auto refresh (1 bit = 1 Channel) for example :

- 0x3 : Channel 0, channel 1
- 0xFF : Channel 0 to 7
- 0xF0 : Channel 3 to 7

[in] ***ulAverageMode*** : Set the average mode :

- 0 : not used
- 1 : average per Sequence : All sequences are acquired x times to compute an average value per channel.
- 2 : average per channel : Each channel is acquired x times to compute an average value for the channel.

[in] ***ulAverageValue*** : Set the average value (only used, when average is used)

- 0 : average not used
- max value : 255

[in] ***ulDivisionFactor*** : Division factor (min: 5, max: 255)

[in] ***ulTriggerAction*** : Trigger action :

Hardware Trigger Start D0 - D7

Bit 3,2,1,0 : Define the trigger mode

- 0000 : Trigger disabled
- 0001 : One shot trigger : After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.
- 0010 : Sequence trigger : After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

Bit 7,6 : define the active front (Only if hardware trigger selected)

- 01 : rising front (Only if hardware trigger selected)
- 10 : falling front (Only if hardware trigger selected)
- 11 : Both front (Only if hardware trigger selected)

Synchronisation Trigger Start : D8-D15

Bit 11,10,9,8 : Define the trigger mode

- 0000 : trigger disabled
- 0001 : One shot trigger : After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.
- 0010 : Sequence trigger : After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

Hardware Trigger Stop D16 - D19

The hardware trigger stop can only be activated when :

- The hardware trigger start is not used.
- The hardware trigger start is used in one shot mode.

The stop of the acquisition is really do at the end of a sequence acquisition(to avoid that the acquisition is stop in the middle of a sequence).

Bit 16 : Define the trigger stop is enable or not

- 0 : Stop trigger disabled
- 1 : Stop trigger enabled.

Bit 18,17 : define the active front (Only if hardware trigger stop selected)

- 01 : rising front (Only if hardware trigger stop selected)
- 10 : falling front (Only if hardware trigger stop selected)

- 11 : Both front (Only if hardware trigger stop selected)

Bit 19 : define if the hardware trigger stop use the ulHardwareTriggerCount (Only if hardware trigger stop selected)

- 0 : ulHardwareTriggerCount not used : First hardware trigger stop will stop the acquisition
- 1 : ulHardwareTriggerCount is used : The ulHardwareTriggerCount hardware trigger will stop the acquisition

[in] ***ulHardwareTriggerCount*** : Define the number of trigger events before the trigger action occur

0 or 1 : all trigger event start the trigger action

max value : 65535

[in] ***ulHardwareTriggerFilterTime*** : Filter time for the hardware trigger (= multiplier from 250 ns step)

max value : 65535

[in] ***ulByTriggerNbrOfSeqToAcquire*** : Define the number of sequence to acquire by each trigger event

[in] ***ulOption1*** : Data format option

D0 : Time stamp information

- 0 : no time stamp information
- 1 : time stamp information

D1 : Data format

- 0 : Digital value
- 1 : Analog value (in mm)

D2 : invert value

- 0 : don't invert the channel value
- 1 : invert the channel value (-2 mm -> + 2mm)

[in] ***ulOption2*** : Reserved

[in] ***ulOption3*** : Reserved

[in] ***ulOption4*** : Reserved

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -2: Transducer selection error
- -3: Channel mask error : can not be null
- -4: Channel mask error
- -5: Average mode error
- -6: Average value error
- -7: Division factor error
- -8: Incorrect value for Hardware Trigger Mode
- -9: Incorrect value for Hardware Trigger front
- -10: Incorrect value for Synchro Trigger Mode
- -11: Incorrect value for Hardware Trigger count
- -12: Incorrect value for Hardware Trigger filter time
- -13: Incorrect value for "trigger number of sequence to acquire"
- -14: Wrong data format parameter (ulOption1)

- -15: A value for Hardware Trigger front was defined but Hardware Trigger Mode is not set
- -16: Cannot use both triggers at the same time
- -17: Incorrect value for the hardware trigger stop front
- -18: Hardware trigger stop can not be used by this configuration of hardware trigger start
- -100: TransducerInit kernel function error
- -101: InitConvertTimeDivisionFactor kernel function error
- -102: SetAutoRefreshAverageValue kernel function error
- -103: InitDigitalInputFilter kernel function error
- -104: InitEnableDisableHardwareTrigger kernel function error
- -105: SynchroTrigger Init/Enable/Disable kernel function error
- -106: SetTriggerSequenceCount kernel function error
- -107: StartAutoRefresh kernel function error

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.41 int MX370x__TransducerGetAutoRefreshValues (void * _, struct MX370x__unsignedlong17ArrayResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0: success
- -100: GetAutoRefreshAllValues kernel function error

ulValue : Array that contain the counter and channels values

- ulValues [0] : Auto refresh counter value
- ulValues [1] : Channel 0 value
- ...
- ulValues [16] : Channel 15 value

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.42 int MX370x__TransducerStopAndReleaseAutoRefresh (void * _, struct MX370x__Response * *Response*)

Parameters

`[in]` _ : no input parameter

[out] ***Response*** :

iReturnValue :

- 0 : success
- -1: means an system error occurred
- -100: "StopAutoRefresh" kernel function error

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.43 int MX370x__TransducerInitAndStartSequence (xsd__unsignedLong *ulTransducerSelection,* xsd__unsignedLong *ulNbrOfChannel,* struct MX370x__unsignedLong16FixedArray * *pulChannelList,* xsd__unsignedLong *ulDivisionFactor,* xsd__unsignedLong *ulNbrOfSequence,* xsd__unsignedLong *ulNbrMaxSequenceToTransfer,* xsd__unsignedLong *ulDelayMode,* xsd__unsignedLong *ulDelayTimeUnit,* xsd__unsignedLong *ulDelayValue,* xsd__unsignedLong *ulTriggerAction,* xsd__unsignedLong *ulHardwareTriggerCount,* xsd__unsignedLong *ulHardwareTriggerFilterTime,* xsd__unsignedLong *ulByTriggerNbrOfSeqToAcquire,* xsd__unsignedLong *ulOption1,* xsd__unsignedLong *ulOption2,* xsd__unsignedLong *ulOption3,* xsd__unsignedLong *ulOption4,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerSelection*** : Transducer type selection

[in] ***ulNbrOfChannel*** : Number of channel in the sequence

[in] ***pulChannelList*** : List of the channel index (0 to MaxChannel-1) who compose the sequence. This parameter is an array.

For a sequence that contains two channels (let say channel 0 and channel 1), you will have:

- pulChannelList[0] = 0
- pulChannelList[1] = 1

[in] ***ulDivisionFactor*** : Division factor (min: 5, max: 255)

The division factor sets the switching time from one channel to another (the channels of the system are multiplexed). When the multiplexer switches from one channel to the next one, you need to wait for a certain time (settling time) before acquiring the measurement value of the transducer. If the division factor is too low (< 10), the measurement can be distorted.

The switching time between two channels equals to the product of the division factor and the exciting signal period of the transducer .

Example: If a transducer connected to channel 0 uses a 10 kHz nominal frequency and the division factor is set to 12, the switching time from channel 0 to the next one is: 12 * (1 / 10000) = 1.2 ms.

[in] ***ulNbrOfSequence*** : Number of sequence to acquire :

- 0 : continuous mode
- > 0 : number of sequence

[in] ***ulNbrMaxSequenceToTransfer*** : This parameter defined the minimal number of sequences to acquired between each send of data by the system.

Warning : They are two possibilities that the number of sequences sent doesn't reach the minimal number:

- By the end of the acquisition.
- If the memory capacity is not big enough.

[in] ***ulDelayMode*** : Delay Mode :

- ADDIDATA_DELAY_NOT_USED 0 : Delay is not used.
- ADDIDATA_DELAY_MODE1_USED 1 : The delay time defines the time between 2 sequence beginnings.
- ADDIDATA_DELAY_MODE2_USED 2 : The delay time defines the time between the end of a sequence until the beginning of the next sequence.

[in] ***ulDelayTimeUnit*** : Selection of the unit of ulDelayValue

- 0: ms
- 1: s

[in] ***ulDelayValue*** : Delay Value (max value: 65535)

[in] ***ulTriggerAction*** : Trigger action :

Hardware Trigger Start D0 - D7

Bit 3,2,1,0 : Define the trigger mode

- 0000 : Trigger disabled
- 0001 : One shot trigger : After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.
- 0010 : Sequence trigger : After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

Bit 7,6 : define the active front (Only if hardware trigger selected)

- 01 : rising front (Only if hardware trigger selected)
- 10 : falling front (Only if hardware trigger selected)
- 11 : Both front (Only if hardware trigger selected)

Synchronisation Trigger Start : D8-D15

Bit 11,10,9,8 : Define the trigger mode

- 0000 : trigger disabled
- 0001 : One shot trigger : After the software start, the module is waiting for a trigger signal to start the acquisition. After this the trigger signal is ignored.
- 0010 : Sequence trigger : After the software start the module is waiting for the trigger signal and acquires x sequences (also adjustable) and then wait again.

Hardware Trigger Stop D16 - D19

The hardware trigger stop can only be activated when :

- The hardware trigger start is not used.
- The hardware trigger start is used in one shot mode.

The stop of the acquisition is really do at the end of a sequence acquisition(to avoid that the acquisition is stop in the middle of a sequence).

Bit 16 : Define the trigger stop is enable or not

- 0 : Stop trigger disabled
- 1 : Stop trigger enabled.

Bit 18,17 : define the active front (Only if hardware trigger stop selected)

- 01 : rising front (Only if hardware trigger stop selected)
- 10 : falling front (Only if hardware trigger stop selected)
- 11 : Both front (Only if hardware trigger stop selected)

Bit 19 : define if the hardware trigger stop use the ulHardwareTriggerCount (Only if hardware trigger stop selected)

- 0 : ulHardwareTriggerCount not used : First hardware trigger stop will stop the acquisition
- 1 : ulHardwareTriggerCount is used : The ulHardwareTriggerCount hardware trigger will stop the acquisition

[in] ***ulHardwareTriggerCount*** : Define the number of trigger events before the trigger action occur

- 0 or 1 : all trigger event start the trigger action
- max value : 65535

[in] ***ulHardwareTriggerFilterTime*** : Filter time for the hardware trigger (= multiplier from 250 ns step)

- max value : 65535

[in] ***ulByTriggerNbrOfSeqToAcquire*** : define the number of sequence to acquire by each trigger event

[in] ***ulOption1*** : Data format option

D0 : Time stamp information

- 0 : no time stamp information
- 1 : timestamp information

D1 : Sequence counter information

- 0 : No sequence counter information
- 1 : Sequence counter information

D2 : Data format

- 0 : Digital value
- 1 : Analog value (in mm)

D3 : invert value

- 0 : don't invert the channel value
- 1 : invert the channel value (-2 mm -> + 2mm)

D4 : receive a relative Time Stamp (first acquisition => time stamp=0) instead of absolute time stamp

- 0 : No relative time stamp information
- 1 : Relative time stamp information

D5 : receive the hardware trigger information

- 0 : no hardware trigger information
- 1 : hardware trigger information

[in] ***ulOption2*** : Reserved

[in] ***ulOption3*** : Reserved

[in] ***ulOption4*** : Reserved

[out] ***Response*** :

iReturnValue :

- 0 : success

- -1: means an system error occurred
- -2: Tranducer selection error
- -3: Number of channel error
- -4: Channel array selection error
- -5: Division factor error
- -6: Incorrect value for Hardware Trigger Mode
- -7: Incorrect value for Hardware Trigger Front
- -8: Incorrect value for Synchro Trigger Mode
- -9: Incorrect value for Hardware Trigger Count
- -10: Incorrect value for Hardware Trigger filter time
- -11: Incorrect value for "trigger number of sequence to acquire"
- -12: Delay Mode selection error
- -13: Delay time unit selection error
- -14: Delay value
- -15: Wrong data format parameter (ulOption1)
- -16: A value for Hardware Trigger front was defined but Hardware Trigger Mode is not set
- -17: Cannot use both triggers at the same time
- -18: Incorrect value for the hardware trigger stop front
- -19: Hardware trigger stop can not be used by this configuration of hardware trigger start
- -100: TransducerInit kernel function error
- -101: InitConvertTimeDivisionFactor kernel function error
- -102: InitEnableDisableSequenceDelay kernel function error
- -103: InitDigitalInputFilter kernel function error
- -104: InitEnableDisableHardwareTrigger kernel function error
- -105: InitEnableSynchroTrigger kernel function error
- -106: DisableSynchroTrigger kernel function error
- -107: SetTriggerSequenceCount kernel function error
- -108: InitSequence kernel function error
- -109: StartStopSequence kernel function error
  syserrno : System-error code (the value of the libc "errno" code)
  Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
  Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.44 int MX370x__TransducerStopAndReleaseSequence (void * _, struct MX370x__Response * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0: success

- -1: means an system error occurred
- -100: StartStopSequence kernel function error

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.45 int MX370x__TransducerSetOffset (struct MSXE370x__doubleArrayParam * *pdOffsetArray,* xsd__unsignedLong *ulOption1,* xsd__unsignedLong *ulOption2,* xsd__unsignedLong *ulOption3,* xsd__unsignedLong *ulOption4,* struct MX370x__Response * *Response*)

This function permits to set an offset (reference point) to the measured value.

To disable (reset) a channel offset, set the corresponding channel value to 0.0.

Example: To set a reference point to a transducer in a particular position:

- Reset the offset by setting all channel offset to 0 (pdOffsetArray).
- Run a sequence with the transducer at the position you want to be 0 (reference point). Save the acquired values to put them into pdOffsetArray.
- Stop the acquisition.
- Run MX370x__TransducerSetOffset function to set the offset with the pdOffsetArray previously saved.
- In the next sequence, position will be 0.

Remark : This function cannot be used when an acquisition is running

For more information see SetOffset sample.

Parameters

`[in]` ***pdOffsetArray*** : array with each offsets for transducers (channel 0 to channel 15)

`[in]` ***ulOption1*** : Reserved

`[in]` ***ulOption2*** : Reserved

`[in]` ***ulOption3*** : Reserved

`[in]` ***ulOption4*** : Reserved

`[out]` ***Response*** :

iReturnValue : Error value

- 0: success
- -1: means an system error occurred
- -2: driver status error, acquisition is running
- -100: transducerSetOffset kernel function error
- <> 0: error

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.46 int MX370x__TransducerGetTypeInformation (xsd__unsignedLong *ulIndex,* struct MX370x__TransducerGetTypeInformationResponse * *Response*)

Parameters

[in] ***ulIndex*** : index of the transducer

[out] ***Response*** :

iReturnValue : Error value

- 0: success
- <> 0: error
- -1: index is invalid
- -100: failure of kernel function "GetTransducerInformation"
- -101: failure of kernel function "GetTransducerType"

ulTransducerSelectionIndex : Selection value. Value to write for the transducer type selection

pcName : Name of the transducer type

ulCalibrationStatus : Calibration status

- 0 : Transducer type is not calibrated
- 1 : Transducer type is calibrated

ulType : Type (0: HB 1: LVDT 2:Knaebel 3:HB-Mahr 4:LVDT-Mahr)

ulFrequency : Frequency (Hz)

ulImpedance : Impedance (Ohm)

dVeff : Nominal voltage (Vrms)

dSensibility : Sensibility (mv/V/mm)

dRange : Range (mm)

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.47 int MX370x__TransducerInitAndStartMinMaxAcquisition (xsd__unsignedLong *ulTransducerSelection,* xsd__unsignedLong *ulChannelMask,* xsd__unsignedLong *ulDivisionFactor,* xsd__unsignedLong *ulStopChannelMask,* xsd__unsignedLong *ulStopCondition,* xsd__unsignedLong *ulStopValue,* xsd__unsignedLong *ulOption1,* xsd__unsignedLong *ulOption2,* xsd__unsignedLong *ulOption3,* xsd__unsignedLong *ulOption4,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerSelection*** : Transducer type selection

[in] ***ulChannelMask*** : Mask of the channel for the min/max evaluation (1 bit = 1 channel)

[in] ***ulDivisionFactor*** : Division factor (min: 5, max: 255)

[in] ***ulStopChannelMask*** : Stop channel mask (1 bit = 1 channel)

[in] ***ulStopCondition*** : Define the condition to stop the min/max acquisition :

- 0 : disabled
- 1 : <
- 2 : >

[in] ***ulStopValue*** : Stop value (24 bits : 0 to 0xFFFFFF)

[in] ***ulOption1*** : Reserved

[in] ***ulOption2*** : Reserved

[in] ***ulOption3*** : Reserved

[in] ***ulOption4*** : Reserved

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -2: Transducer selection error
- -3: Channel mask can not be null
- -4: channel mask error
- -5: Division factor error
- -6: Stop condition selection error
- -7: Stop channel mask can not be null
- -8: Stop channel mask error
- -9: Stop value error
- -100: TransducerInit kernel function error
- -101: InitConvertTimeDivisionFactor kernel function error
- -102: InitMinMaxAcquisition kernel function error
- -103: StartStopMinMaxAcquisition kernel function error

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.48 int MX370x__TransducerGetMinMaxStatus (void * _, struct MX370x__TransducerGetMinMaxStatusResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue :

- 0 : success
- -100: GetMinMaxAcquisitionStatus kernel function error

ulFlag : Min/Max acquisition status :

- 0 : Disable
- 1 : Enable (in progress)
- 2 : End of sequence

ulOverflow : Overflow status

- 0 : No overflow
- 1 : PLD overflow

pulMinValues : Array with the minimale values

pulMaxValues : Array with the maximale values

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.49 int MX370x__TransducerStopAndReleaseMinMaxAcquisition (void * _, struct MX370x__Response * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: StartStopMinMaxAcquisition kernel function error

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.50 int MX370x__TransducerInitPrimaryConnectionTest (void * _, struct MX370x__Response * *Response*)

Can not be used for the MSX-E370x Mahr This function save the number of plugged transducer. This value will then be used when calling the MX370x__TransducerTestPrimaryConnection function. You must call this function at least one time after boot, and then, each time you change the plugged transducer.

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: Primary short circuit occur
- -101: No transducer connected
- -103: Functionality not available

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.51 int MX370x__TransducerTestPrimaryConnection (void * _, struct MX370x__unsignedlongDefaultResponse * *Response*)

. Can not be used for the MSX-E370x Mahr The saved status input from the MX370x__-TransducerInitPrimaryConnectionTest function is compared to a new status by the call of this function.

Important !!

This function can not be used for the MSX-E370x Mahr modules. Refer you to the MX370x__-TransducerTestSecondaryConnection function.

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: Primary short circuit occur
- -101: No transducers connected
- -102: Test primary connection but no initialisation occur.
- -103: Functionality not available.

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

ulValue : Connection status:

- 0: connection error
- 1: connection ok

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.52 int MX370x__TransducerTestPrimaryShortCircuit (void * _, struct MX370x__unsignedlongDefaultResponse * *Response*)

On the primary circuit the supply voltage of the power buffer is controlled. If a short circuit occurs (between OSC+ and OSC- or OSC- against mass or OSC+ against mass), a voltage drop is detected.

This information is returned by this function.

In case of short circuit the power buffer disposes of internal fuses which switch the outputs off.

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.
ulValue : Short circuit status:

- 0: short circuit
- 1: no short circuit

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.53 int MX370x__TransducerRearmPrimary (void * _, struct MX370x__unsignedlongDefaultResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.
ulValue : Rearm status:

- 0: Rearm not ok
- 1: Rearm ok

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.54 int MX370x__TransducerTestSecondaryConnection (xsd__unsignedLong *ulChannel,* struct MX370x__unsignedlongDefaultResponse * *Response*)

For the MSX-E370x Mahr modules, this function test if the connected transducer is OK or not.

Parameters

[in] ***ulChannel,:*** Channel selection (0 to MaxChannel-1)

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: Primary short circuit occur

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.
ulValue : Connection status:

- 0: connection error
- 1: connection ok

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.55 int MX370x__TransducerTestSecondaryShortCircuit (xsd__unsignedLong *ulChannel,* struct MX370x__unsignedlongDefaultResponse * *Response*)

Refer you to the MX370x__TransducerTestSecondaryConnection function.

Parameters

[in] ***ulChannel,:*** Channel selection (0 to MaxChannel-1)

[out] ***Response*** :

iReturnValue :

- 0: success
- -1: means an system error occurred
- -100: Primary short circuit occur
- -101: Functionality not available

syserrno : System-error code (the value of the libc "errno" code)
Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
Give this value to the MXCommon_Strerror to get the string describing the error number.
ulValue : Short circuit status:

- 0: short circuit
- 1: no short circuit

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.56 int MX370x__CalibrationStart (xsd__unsignedLong *ulTransducerIndex,* xsd__unsignedLong *ulChannel,* xsd__double *dPosition,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerIndex*** : Selected transducer type to calibrate

[in] ***ulChannel*** : Selected the channel to use for the calibration (0 to MaxChannel-1)

[in] ***dPosition*** : Selected user calibration position in mm (-transducer range to +transducer range)

[out] ***Response*** :

iReturnValue : Error value :

- 0 : means the remote function performed OK
- -1 : means an system error occured

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.57 int MX370x__CalibrationStartWithPrimaryConnection (xsd__unsignedLong *ulTransducerIndex,* xsd__unsignedLong *ulChannel,* xsd__double *dPosition,* xsd__unsignedLong *ulReserved,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerIndex*** : Selected transducer type to calibrate

[in] ***ulChannel*** : Selected the channel to use for the calibration (0 to MaxChannel-1)

[in] ***dPosition*** : Selected user calibration position in mm (-transducer range to +transducer range)

[in] ***ulReserved*** : Reserved muss be set to 0

[out] ***Response*** :

iReturnValue : Error value :

- 0 : means the remote function performed OK
- -1 : means an system error occured

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.58 int MX370x__CalibrationGetCurrentStatus (void * _, struct MX370x__CalibrationGetCurrentStatusResponse * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue : Error value

- 0 : No Error
- <> 0 : Error

ulStatus : Status

- 0: No calibration in progress
- 1: Primary calibration in progress
- 2: Wait user access null position setting
- 3: Null position calibration thread in progress
- 4: Wait user access user position setting
- 5: User position calibration thread in progress
- 6: Calibration finiched
- 7: Wait user connect only one transducer for the primary open line diagnostic
- 8: Primary open line diagnostic thread in progress

ulDigitalValue : Last measured digital value

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.59 int MX370x__CalibrationNextStep (void * _, struct MX370x__Response * *Response*)

Parameters

`[in]` _ : no input parameter

`[out]` ***Response*** :

iReturnValue : Error value :

- 0 : means the remote function performed OK
- -1 : means an system error occured

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.60 int MX370x__CalibrationBreak (void * _, struct MX370x__Response * *Response*)

The values of the digital potentiometer will be lost.

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue : Error value :

- 0 : means the remote function performed OK
- -1 : means an system error occured

syserrno : System-error code (the value of the libc "errno" code)

Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.61 int MX370x__DataBaseGetNumberOfTransducers (void * _, struct MX370x__unsignedlongResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** :

iReturnValue : Error code :

- 0 : success
- <> 0 : error
- -100 : kernel function error

ulValue : number of transducer types.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.62 int MX370x__DataBaseGetTransducerType (xsd__unsignedLong *ulTransducerIndex*, struct MX370x__unsignedlongResponse * *Response*)

Parameters

[in] ***ulTransducerIndex*** : Transducer index (0 to ulTransducerNumbers - 1)

[out] ***Response*** :

iReturnValue : Error code :

- 0 : success
- <> 0 : error
- -100 : kernel function error

ulValue : transducer identifier. Use this value as the "Index" parameter given to DataBaseGetTransducerInformationResponse().

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.63 int MX370x__DataBaseGetTransducerInformation (xsd__unsignedLong *ulTransducerIndex,* struct MX370x__DataBaseGetTransducerInformationResponse * *Response*)

Parameters

[in] ***ulTransducerIndex*** : transducer identifier, as returned by DataBaseGetTransducerType().

[out] ***Response*** :

iReturnValue : Error code :

- 0 : success
- <> 0 : error
- -100: kernel function error

cName : Name
ulCalibrate : Calibration state (0 : not calibrated 1 : calibrated)
ulType : Type (0: HB 1: LVDT 2:Knaebel 3:HB-Mahr 4:LVDT-Mahr)
ulFrequency : Nominal frequency (Hz)
ulImpedance : Impedance (Ohm)
dVeff : Nominal voltage (Vrms)
dSensitivity : Sensitivity (mV/V/mm)
dRange : Range (mm)

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.64 int MX370x__DataBaseAddTransducer (xsd__unsignedLong *ulTransducerIndex,* xsd__string *cName,* xsd__unsignedLong *ulType,* xsd__unsignedLong *ulFrequency,* xsd__unsignedLong *ulImpedance,* xsd__double *dVeff,* xsd__double *dSensitivity,* xsd__double *dRange,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerIndex*** : Identifier of the new type, user-defined value in the range 200 .. 255

[in] ***cName*** : Name

[in] ***ulType*** : Type (0: HB 1: LVDT 2:Knaebel 3:HB-Mahr 4:LVDT-Mahr)

[in] ***ulFrequency*** : Nominal frequency (Hz)

[in] ***ulImpedance*** : Impedance (Ohm)

[in] ***dVeff*** : Nominal voltage (Vrms)

[in] ***dSensitivity*** : Sensitivity (mV/V/mm)

[in] ***dRange*** : Range (mm)

[out] ***Response*** :

iReturnValue : Error code :

- 0 : success
- <> 0 : error
- -100: kernel function error
  syserrno : system-error code (the value of the libc "errno" code) Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
  Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

Note

This function returns an error if a transducer with the same identifier already exists in the database.

4.1.2.65 int MX370x__DataBaseDelTransducer (xsd__unsignedLong *ulTransducerIndex,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulTransducerIndex*** : identifier, as returned by DataBaseGetTransducerType().

[out] ***Response*** :

iReturnValue : Error value :

- 0 : success
- <> 0 : error
- -100: kernel function error
  syserrno : system-error code (the value of the libc "errno" code) Its value is significant only when the iReturnValue returned an error (-1 or <= -100)
  Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

Note

This function returns an error if the identifier does not map to an existing transducer type.

4.1.2.66 int MX370x__DataBaseSaveTransducers (void * _, struct MX370x__ByteArrayResponse * *Response*)

Parameters

[in] _ : no input parameter

[out] ***Response*** : ***sResponse.iReturnValue :*** Error code

- 0 success
- <> 0 : error

sResponse.syserrno : system-error code (the value of the libc "errno" code) Its value is significant only when the iReturnValue returned an error (-1 or <= -100)

Give this value to the MXCommon_Strerror to get the string describing the error number.

Returns

- 0: SOAP_OK
- <> 0: See SOAP error

4.1.2.67 int MX370x__ExtDigitalIOGetNumberOfChannels (xsd__unsignedLong *ulOption1*, struct MX370x__unsignedLongResponse * *Response*)

Parameters

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Number of available digital I/O channels

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.68 int MX370x__ExtDigitalIOGetNumberOfPorts (xsd__unsignedLong *ulOption1*, struct MX370x__unsignedLongResponse * *Response*)

A port is a set of consecutive digital I/O channels which states can be get or set at the same time.

Parameters

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Number of available digital I/O ports

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.69 int MX370x__ExtDigitalIOGetNumberOfChannelsPerPort (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)
Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response*** ***sResponse.iReturnValue :***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Number of digital I/O channels for the selected port

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.70 int MX370x__ExtDigitalIOGetPortDirections (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)
Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Digital directions selection. Each bit indicates the direction for one channel.

- D0 : 0: Digital I/O 0 of selected port used as input. 1: Digital I/O 0 of selected port used as output
- ...
- D31 : 0: Digital I/O 31 of selected port used as input. 1: Digital I/O 31 of selected port used as output

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.71 int MX370x__ExtDigitalIOSetInputsFilterTime (xsd__unsignedLong *ulFilterTime,* xsd__unsignedLong *ulOption1,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulFilterTime*** Filter time, maximum 511 (unit 20 micro s) (1 corresponds to 20 micro s, 2 corresponds to 40 micro s, ...)

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Filter time selection wrong
- **-3**: Error when writing the new filter time
- **-100**: Internal system error occurred. See value of syserrno

syserrno system-error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.72 int MX370x__ExtDigitalIOEnableDisableInputsFilter (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulFilter,* xsd__unsignedLong *ulOption1,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)

Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulFilter*** 1 to enable, 0 to disable

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Filter selection wrong
- **-4**: Error when writing new filter state
- **-100**: Internal system error occurred. See value of syserrno

syserrno system-error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.73 int MX370x__ExtDigitalIOGetInputsFilterConfiguration (xsd_-_unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__ExtDigitalIOGetInputsFilterConfigurationResponse * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)

Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulFilterTime Filter time, maximum 511 (unit 20 micro s) (1 corresponds to 20 micro s, 2 corresponds to 40 micro s, ...)

ulFilter 1 filter is enabled, 0 filter is disabled

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.74 int MX370x__ExtDigitalIOTestOutputsShortCircuit (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)

Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response*** ***sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Error when getting the diagnosis
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Digital outputs short-circuit state.

- 0: No short-circuit.
- 1: Short-circuit detected

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.75 int MX370x__ExtDigitalIOTestOutputsPowerSupply (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse * *Response*)

The digital outputs need an external power supply. This function checks the state of the power supply. If the power supply is ok, the function returns 0 (in ulValue).

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1).
Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Error when getting the diagnosis
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)
ulValue Digital outputs power supply state.

- 0: power supply state is ok.
- 1: no power supply detected on the outputs

Return values

0 SOAP_OK
Others See SOAP error

4.1.2.76 int MX370x__ExtDigitalIOReadChannel (xsd__unsignedLong *ulChannel,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse * *Response*)

If selected channel is an output, then this function returns the current output state.

Parameters

[in] ***ulChannel*** Selected digital I/O channel (0 to MX370x__ExtDigitalIOGetNumberOfChannels - 1)

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Channel selection wrong
- **-3**: Error when reading the channel state
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)
ulValue Digital I/O channel state

- 0: Digital I/O channel is low
- 1: Digital I/O channel is high

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.77 int MX370x__ExtDigitalIOReadPort (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulOption1,* struct MX370x__unsignedLongResponse * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)

Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response sResponse.iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Error when reading the port state
- **-100**: Internal system error occurred. See value of syserrno

sResponse.syserrno system-error code (the value of the libc "errno" code)

ulValue Digital I/O state. Each bit represent the state for one digital I/O channel.

- D0 : 0: Digital I/O 0 of selected port is low. 1: Digital I/O 0 of selected port is high
- ...
- D31 : 0: Digital I/O 31 of selected port is low. 1: Digital I/O 31 of selected port is high

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.78 int MX370x__ExtDigitalIOWriteChannel (xsd__unsignedLong *ulChannel,* xsd__unsignedLong *ulState,* xsd__unsignedLong *ulOption1,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulChannel*** Selected digital I/O channel (0 to MX370x__ExtDigitalIOGetNumberOfChannels - 1)

[in] ***ulState*** Digital I/O channel state

- 0: Set the digital I/O output channel to low
- 1: Set the digital I/O output channel to high

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Channel selection wrong or selected channel is an input

- **-3**: State selection wrong
- **-4**: Error when setting digital output
- **-100**: Internal system error occurred. See value of syserrno

syserrno system-error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

4.1.2.79 int MX370x__ExtDigitalIOWritePort (xsd__unsignedLong *ulPort,* xsd__unsignedLong *ulState,* xsd__unsignedLong *ulOption1,* struct MX370x__Response * *Response*)

Parameters

[in] ***ulPort*** Selected digital I/O port (0 to MX370x__ExtDigitalIOGetNumberOfPorts -1)

Have a look on the documentation of MX370x__ExtDigitalIOGetNumberOfPorts for the description of a port.

[in] ***ulState*** Digital I/O state. Each bit set the state for one digital I/O channel.

- D0 : 0: Set the digital I/O output channel 0 of the selected port to low. 1: Set the digital I/O output channel 0 of the selected port to high
- ...
- D31 : 0: Set the digital I/O output channel 31 of the selected port to low. 1: Set the digital I/O output channel 31 of the selected port to high

[in] ***ulOption1*** Reserved. Set to 0

[out] ***Response iReturnValue***

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Port selection wrong
- **-3**: Any selected digital I/O is not a output channel
- **-4**: Error when setting digital outputs
- **-100**: Internal system error occurred. See value of syserrno

syserrno system-error code (the value of the libc "errno" code)

Return values

0 SOAP_OK

Others See SOAP error

Index