

MSXE360x soap api functions

Generated by Doxygen 1.7.1

Thu Sep 5 2013 09:34:48

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Remark: SOAP functions prototypes	1
2	Module Documentation	3
2.1	MSX-E360x functions	3
2.2	Software hints	3
2.3	SOAP function calls in C/C++ language	3
2.3.1	C/C++ language SOAP function prototypes	3
2.3.2	Rules and use of gSOAP calls in C/C++	4
2.3.3	SOAP calls sample	6
2.4	MSX-E systems servers	10
2.4.1	SOAP server	10
2.4.2	Event server	10
2.4.3	Modbus server	10
2.5	Common functions	10
2.6	Common general functions	11
2.6.1	Function Documentation	12
2.6.1.1	MXCommon__GetModuleType	12
2.6.1.2	MXCommon__GetHostname	12
2.6.1.3	MXCommon__SetHostname	13
2.6.1.4	MXCommon__GetClientConnections	13
2.6.1.5	MXCommon__Strerror	13
2.6.1.6	MXCommon__Reboot	15
2.6.1.7	MXCommon__ResetAllIOFunctionalities	15
2.6.1.8	MXCommon__DataseverRestart	15
2.6.1.9	MXCommon__GetEthernetLinksStates	16
2.7	Common temperature functions	17

2.7.1	Detailed Description	17
2.7.2	Function Documentation	17
2.7.2.1	MXCommon__GetModuleTemperatureValueAndStatus	17
2.7.2.2	MXCommon__SetModuleTemperatureWarningLevels	18
2.8	Common hardware trigger functions	18
2.8.1	Function Documentation	19
2.8.1.1	MXCommon__SetHardwareTriggerFilterTime	19
2.8.1.2	MXCommon__GetHardwareTriggerFilterTime	20
2.8.1.3	MXCommon__GetHardwareTriggerState	20
2.9	Common security functions	20
2.9.1	Detailed Description	21
2.9.2	Function Documentation	22
2.9.2.1	MXCommon__SetCustomerKey	22
2.9.2.2	MXCommon__TestCustomerID	22
2.10	Common time functions	22
2.10.1	Detailed Description	23
2.10.2	Function Documentation	23
2.10.2.1	MXCommon__SetTime	23
2.10.2.2	MXCommon__SysToHardwareClock	24
2.10.2.3	MXCommon__HardwareClockToSys	24
2.10.2.4	MXCommon__GetTime	24
2.10.2.5	MXCommon__GetUpTime	25
2.11	Common I/O auto configuration functions	25
2.11.1	Detailed Description	26
2.11.2	Function Documentation	26
2.11.2.1	MXCommon__GetAutoConfigurationFile	26
2.11.2.2	MXCommon__SetAutoConfigurationFile	26
2.11.2.3	MXCommon__StartAutoConfiguration	27
2.12	Common synchronisation timer functions	27
2.12.1	Function Documentation	28
2.12.1.1	MXCommon__InitAndStartSynchroTimer	28
2.12.1.2	MXCommon__StopAndReleaseSynchroTimer	29
2.13	Set/Backup/Restore general system configuration	29
2.13.1	Detailed Description	29
2.13.2	Function Documentation	30
2.13.2.1	MXCommon__GetConfigurationBackupFile	30

2.13.2.2	MXCommon__ApplyConfigurationBackupFile	30
2.13.2.3	MXCommon__ChangePassword	31
2.14	System state management	31
2.14.1	Detailed Description	32
2.14.2	Function Documentation	32
2.14.2.1	MXCommon__GetSubSystemState	32
2.14.2.2	MXCommon__GetSubsystemIDFromName	33
2.14.2.3	MXCommon__GetStateIDFromName	33
2.14.2.4	MXCommon__GetSubsystemNameFromID	34
2.14.2.5	MXCommon__GetStateNameFromID	34
2.15	Customer option management	34
2.15.1	Function Documentation	35
2.15.1.1	MXCommon__GetOptionInformation	35
2.16	Synchronisation management	35
2.16.1	Function Documentation	35
2.16.1.1	MXCommon__SetToMaster	35
2.16.1.2	MXCommon__GetSynchronizationStatus	36
2.17	MSX-E360x Sequence functions	36
2.17.1	Function Documentation	37
2.17.1.1	MSXE360x__AnalogInputInitAndStartSequence	37
2.17.1.2	MSXE360x__AnalogInputStopAndReleaseSequence	41
2.17.1.3	MSXE360x__AnalogInputGetSequenceStatus	41
3	Data Structure Documentation	43
3.1	ByteArray Struct Reference	43
3.1.1	Field Documentation	43
3.1.1.1	__ptr	43
3.1.1.2	__size	43
3.1.1.3	__offset	43
3.2	DefaultResponse Struct Reference	43
3.2.1	Field Documentation	44
3.2.1.1	iReturnValue	44
3.2.1.2	syserrno	44
3.3	MSXE360x__Response Struct Reference	44
3.3.1	Field Documentation	44
3.3.1.1	iReturnValue	44
3.3.1.2	syserrno	44

3.4	MSXE360x__unsignedLong8FixedArrayParam Struct Reference	44
3.4.1	Field Documentation	45
3.4.1.1	ulValue	45
3.5	MSXE360x__unsignedlongResponse Struct Reference	45
3.5.1	Field Documentation	45
3.5.1.1	sResponse	45
3.5.1.2	ulValue	45
3.6	MXCommon__ByteArrayResponse Struct Reference	45
3.6.1	Field Documentation	45
3.6.1.1	sResponse	45
3.6.1.2	sArray	45
3.7	MXCommon__FileResponse Struct Reference	45
3.7.1	Field Documentation	46
3.7.1.1	sResponse	46
3.7.1.2	sArray	46
3.7.1.3	ulEOF	46
3.8	MXCommon__GetAutoConfigurationFileResponse Struct Reference	46
3.8.1	Field Documentation	46
3.8.1.1	sResponse	46
3.8.1.2	bArray	46
3.8.1.3	ulEOF	46
3.9	MXCommon__GetEthernetLinksStatesResponse Struct Reference	46
3.9.1	Field Documentation	47
3.9.1.1	sResponse	47
3.9.1.2	sPort0	47
3.9.1.3	sPort1	47
3.10	MXCommon__GetHardwareTriggerFilterTimeResponse Struct Reference	47
3.10.1	Field Documentation	47
3.10.1.1	sResponse	47
3.10.1.2	ulFilterTime	47
3.10.1.3	ulInfo01	47
3.10.1.4	ulInfo02	47
3.11	MXCommon__GetHardwareTriggerStateResponse Struct Reference	47
3.11.1	Field Documentation	48
3.11.1.1	sResponse	48
3.11.1.2	ulState	48

3.11.1.3	ulInfo01	48
3.11.1.4	ulInfo02	48
3.12	MXCommon__GetModuleTemperatureValueAndStatusResponse Struct Reference	48
3.12.1	Field Documentation	49
3.12.1.1	sResponse	49
3.12.1.2	dTemperatureValue	49
3.12.1.3	ulTemperatureStatus	49
3.12.1.4	ulInfo	49
3.13	MXCommon__GetTimeResponse Struct Reference	49
3.13.1	Field Documentation	49
3.13.1.1	sResponse	49
3.13.1.2	ulLowTime	49
3.13.1.3	ulHighTime	49
3.14	MXCommon__GetUpTimeResponse Struct Reference	49
3.14.1	Field Documentation	50
3.14.1.1	sResponse	50
3.14.1.2	ulUpTime	50
3.15	MXCommon__Response Struct Reference	50
3.15.1	Field Documentation	50
3.15.1.1	iReturnValue	50
3.15.1.2	syserrno	50
3.16	MXCommon__TestCustomerIDResponse Struct Reference	50
3.16.1	Field Documentation	51
3.16.1.1	sResponse	51
3.16.1.2	bValueArray	51
3.16.1.3	bCryptedValueArray	51
3.17	MXCommon__unsignedLongResponse Struct Reference	51
3.17.1	Field Documentation	51
3.17.1.1	sResponse	51
3.17.1.2	ulValue	51
3.18	sGetEthernetLinksStatesPort Struct Reference	51
3.18.1	Field Documentation	52
3.18.1.1	ulState	52
3.18.1.2	ulSpeed	52
3.18.1.3	ulDuplex	52
3.18.1.4	ulInfo1	52

3.18.1.5	ulInfo2	52
3.19	UnsignedLongArray Struct Reference	52
3.19.1	Field Documentation	52
3.19.1.1	__ptr	52
3.19.1.2	__size	52
3.19.1.3	__offset	52
3.20	UnsignedShortArray Struct Reference	52
3.20.1	Field Documentation	53
3.20.1.1	__ptr	53
3.20.1.2	__size	53
3.20.1.3	__offset	53
3.21	xsd__base64Binary Struct Reference	53
3.21.1	Field Documentation	53
3.21.1.1	__ptr	53
3.21.1.2	__size	53
4	File Documentation	55
4.1	MSXE360x_public_doc.h File Reference	55
4.1.1	Typedef Documentation	60
4.1.1.1	xsd__string	60
4.1.1.2	xsd__char	60
4.1.1.3	xsd__float	60
4.1.1.4	xsd__double	60
4.1.1.5	xsd__int	60
4.1.1.6	xsd__long	60
4.1.1.7	xsd__unsignedByte	60
4.1.1.8	xsd__unsignedInt	60
4.1.1.9	xsd__unsignedShort	60
4.1.1.10	xsd__unsignedLong	60
4.1.2	Function Documentation	60
4.1.2.1	MXCommon__GetModuleType	60
4.1.2.2	MXCommon__GetHostname	61
4.1.2.3	MXCommon__SetHostname	61
4.1.2.4	MXCommon__GetClientConnections	61
4.1.2.5	MXCommon__Strerror	62
4.1.2.6	MXCommon__Reboot	63
4.1.2.7	MXCommon__ResetAllIOFunctionalities	63

4.1.2.8	MXCommon__DataserverRestart	64
4.1.2.9	MXCommon__GetEthernetLinksStates	64
4.1.2.10	MXCommon__GetModuleTemperatureValueAndStatus	65
4.1.2.11	MXCommon__SetModuleTemperatureWarningLevels	66
4.1.2.12	MXCommon__SetHardwareTriggerFilterTime	66
4.1.2.13	MXCommon__GetHardwareTriggerFilterTime	67
4.1.2.14	MXCommon__GetHardwareTriggerState	67
4.1.2.15	MXCommon__SetCustomerKey	68
4.1.2.16	MXCommon__TestCustomerID	68
4.1.2.17	MXCommon__SetTime	68
4.1.2.18	MXCommon__SysToHardwareClock	69
4.1.2.19	MXCommon__HardwareClockToSys	69
4.1.2.20	MXCommon__GetTime	70
4.1.2.21	MXCommon__GetUpTime	70
4.1.2.22	MXCommon__GetAutoConfigurationFile	70
4.1.2.23	MXCommon__SetAutoConfigurationFile	71
4.1.2.24	MXCommon__StartAutoConfiguration	71
4.1.2.25	MXCommon__InitAndStartSynchroTimer	71
4.1.2.26	MXCommon__StopAndReleaseSynchroTimer	72
4.1.2.27	MXCommon__GetConfigurationBackupFile	73
4.1.2.28	MXCommon__ApplyConfigurationBackupFile	74
4.1.2.29	MXCommon__ChangePassword	74
4.1.2.30	MXCommon__GetSubSystemState	75
4.1.2.31	MXCommon__GetSubsystemIDFromName	75
4.1.2.32	MXCommon__GetStateIDFromName	75
4.1.2.33	MXCommon__GetSubsystemNameFromID	76
4.1.2.34	MXCommon__GetStateNameFromID	76
4.1.2.35	MXCommon__GetOptionInformation	77
4.1.2.36	MXCommon__SetToMaster	77
4.1.2.37	MXCommon__GetSynchronizationStatus	77
4.1.2.38	MSXE360x__AnalogInputInitAndStartSequence	78
4.1.2.39	MSXE360x__AnalogInputStopAndReleaseSequence	82
4.1.2.40	MSXE360x__AnalogInputGetSequenceStatus	82

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 MSX-E360x functions

Modules

- [MSX-E360x Sequence functions](#)

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_endpoint,
    const char *soap_action, void *_ , struct MXCommon__ByteArrayResponse *Response
);
```

Functions to call in C/C++ are called **stubs** and can be found in the **MSXEXXXStub.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_KEEPAIVE, SOAP_IO_KEEPAIVE);

// 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.nsmmap> // 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_KEEPAIVE, SOAP_IO_KEEPAIVE);

    // 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

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

# The MSX-E IP address
ifndef $(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.

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__DataseverRestart (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)
STRError(3)

Linux Programmer's Manual

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** Reserved
 - **ulInfo2** Reserved
- 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** Reserved
- **ulInfo2** Reserved

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__Sterror\(\)](#).
 - 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__Sterror\(\)](#).

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__Sterror\(\)](#).

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__Sterror\(\)](#).
 - 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__Sterror\(\)](#).

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__Sterror\(\)](#).

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__Sterror\(\)](#).
- `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__Sterror\(\)](#).
 - `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__Sterror\(\)](#). 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__Sterror\(\)](#).

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 uEOF 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: authentication 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__Sterror\(\)](#).
 - 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__Sterror\(\)](#).
- 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__Sterror\(\)](#).
- 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__Sterror\(\)](#).
 - 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__Sterror\(\)](#).
 - 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 MSX-E360x Sequence functions

Functions

- `int MSXE360x__AnalogInputInitAndStartSequence (xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulNbrOfSequence, xsd__unsignedLong ulNbrMaxSequenceToTransfer, xsd__double dFrequencySelection, struct MSXE360x__unsignedLong8FixedArrayParam *pulGainArray, xsd__unsignedLong ulICPMask, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulCouplingSelectionMask, xsd__unsignedLong ulSeDiffSelectionMask, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, xsd__unsignedLong ulOption5, xsd__unsignedLong ulOption6, struct MSXE360x__Response *Response)`

Initialise and start the analog input sequence acquisition mode.

- int [MSXE360x__AnalogInputStopAndReleaseSequence](#) (void *_ , struct [MSXE360x__Response](#) *Response)

Stop and Release the analog input sequence acquisition mode.

- int [MSXE360x__AnalogInputGetSequenceStatus](#) (void *_ , struct [MSXE360x__unsignedlongResponse](#) *Response)

Get the analog input sequence acquisition status.

2.17.1 Function Documentation

- 2.17.1.1** int [MSXE360x__AnalogInputInitAndStartSequence](#) (xsd__unsignedLong *ulChannelMask*, xsd__unsignedLong *ulNbrOfSequence*, xsd__unsignedLong *ulNbrMaxSequenceToTransfer*, xsd__double *dFrequencySelection*, struct [MSXE360x__unsignedLong8FixedArrayParam](#) * *pulGainArray*, xsd__unsignedLong *ulICPMask*, xsd__unsignedLong *ulTriggerMask*, xsd__unsignedLong *ulTriggerMode*, xsd__unsignedLong *ulHardwareTriggerEdge*, xsd__unsignedLong *ulHardwareTriggerCount*, xsd__unsignedLong *ulByTriggerNbrOfSeqToAcquire*, xsd__unsignedLong *ulDataFormat*, xsd__unsignedLong *ulCouplingSelectionMask*, xsd__unsignedLong *ulSeDiffSelectionMask*, xsd__unsignedLong *ulOption1*, xsd__unsignedLong *ulOption2*, xsd__unsignedLong *ulOption3*, xsd__unsignedLong *ulOption4*, xsd__unsignedLong *ulOption5*, xsd__unsignedLong *ulOption6*, struct [MSXE360x__Response](#) * *Response*)

Parameters

[in] ***ulChannelMask*** : 8 bits mask (0->0xFF) which defines channels used for the acquisition (each bit corresponds to one channel)

- 0 : not used
- 1 : used

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

- 0 : Continuous mode
- > 0 : number of sequence (1 -> 4294967295 (0xFFFFFFFF))

[in] ***ulNbrMaxSequenceToTransfer*** : Not used, must be 0

[in] ***dFrequencySelection*** : Select the frequency of the acquisition

- 1000.00 Hz
- 1280.00 Hz
- 1562.50 Hz
- 1600.00 Hz
- 1666.67 Hz
- 2000.00 Hz
- 2500.00 Hz
- 3125.00 Hz
- 3200.00 Hz
- 3333.33 Hz
- 4000.00 Hz
- 5000.00 Hz

- 6250.00 Hz
- 6400.00 Hz
- 6666.67 Hz
- 8000.00 Hz
- 10000.00 Hz
- 12500.00 Hz
- 12800.00 Hz
- 13333.33 Hz
- 16000.00 Hz
- 16666.67 Hz
- 20000.00 Hz
- 25000.00 Hz
- 32000.00 Hz
- 33333.33 Hz
- 40000.00 Hz
- 50000.00 Hz
- 64000.00 Hz
- 66666.67 Hz
- 80000.00 Hz
- 100000.00 Hz
- 128000.00 Hz

[in] **pulGainArray** : Define the gain (1,10 or 100) to use for each channel.

Each index of the array corresponds to the corresponding channel :

example :

- [0] : Define the gain for the channel 0
- [1] : Define the gain for the channel 1
- ...

[in] **ulICPMask** : 8 bits mask (0->0xFF) which defines if the ICP is activated or not. (each bit corresponds to one channel)

When the ICP is activated, the channel must be configured with AC and SE.

- 0 : not activated
- 1 : activated

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

- 0 : trigger disabled
- 1 : Enable Hardware Digital Input Trigger
- 2 : Enable Synchro Trigger
- 3 : Enable both Hardware and Synchro Trigger

[in] **ulTriggerMode** : Not used, must be 0

[in] **ulHardwareTriggerEdge** : Define the edge of the trigger

- 1 : Hardware trigger rising edge
- 2 : Hardware trigger falling edge
- 3 : Enable both rising edge and falling edge

[in] **ulHardwareTriggerCount** : Define the number of external trigger ignored before taking it account (1 -> 65535)

[in] **ulByTriggerNbrOfSeqToAcquire** : Not used, must be 0

[in] ***ulDataFormat*** : Dataformat of the frame, see remarks and examples for more informations :

D0 : Absolute time stamp information (2*32 bits data)

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

D1 : Not used must be 0

D2 : Sequence counter (32 bits data)

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

D3 : Hardware trigger information (32 bits data)

- 0 : No hardware trigger information required
- 1 : Hardware trigger information required

[in] ***ulCouplingSelectionMask*** : 8 bits mask (0->0xFF) which defines the coupling for each channel (each bit corresponds to one channel)

- 0 : AC
- 1 : DC

[in] ***ulSeDiffSelectionMask*** : 8 bits mask (0->0xFF) which defines SE/DIFF mode for each channel (each bit corresponds to one channel)

- 0 : SE
- 1 : DIFF

Remark 1 : data packets depends on the (number of sequence asked) * (sequence size in 32 bits words) * 4

- <4-8192> bytes, Only one packet of the corresponding size is sent
- 0 (continuous) or > 8192 bytes : packet of 8192 bytes containing the sequences are sent by the MSXE, for the last sequence, the last packet is sent with the rest of the size

Remark 2 : the data order is

- timestamp seconds (optional)
- timestamp microseconds (optional)
- sequence counter (optional)
- hardware trigger information (optional)
- selected channels in ascending order

Remark 3 : sequence size in bytes

- [(timestamp (s) + timestamp (us)) (optional) + sequence counter (optional) + hardware trigger information (optional) + number of channels] * 4

Example 1 : 4 channels in continuous sequence mode

- packets of 8192 bytes are sent -> 512 sequences of 4 32 bits channels per packet

Example 2 : 10 sequences of 4 channels with timestamp and hardware trigger information.

- 1 packet of 280 bytes is sent by the MSXE -> 10 sequences of : timestamp (2*32 bits word) + sequence counter size (32 bits) + 4 * 32 bits channels)

Parameters

[out] **Response** :

iReturnValue :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -2: pld is not working
- -3: error, system is in calibration
- -4: channel action is wrong
- -5: gain selection error
- -6: channel coupling selection error
- -7: SE / Diff selection error
- -8: ICP selection error
- -9: ICP can only be used with AC and SE
- -10: driver is not in idle state
- -11: pld is not working
- -12: error, system is in calibration
- -13: Frequency selection error
- -14: driver is not in idle state
- -19: channel mask can not be null
- -20: channel mask selection error
- -21: number of sequence selection error
- -22: sequence interrupt selection error (must be 0)
- -23: the ulTriggerMode parameter is wrong (must be 0)
- -24: the ulHardwareTriggerEdge parameter is wrong
- -25: the ulHardwareTriggerCount parameter is wrong
- -26: the ulByTriggerNbrOfSeqToAcquire parameter is wrong (must be 0)
- -27: the ulDataFormat parameter is wrong
- -28: the ulTriggerMask parameter is wrong
- -29: the ulICPMask is wrong (0->0xFF)
- -30: the ulCouplingSelectionMask is wrong (0->0xFF)
- -31: the ulSeDiffSelectionMask is wrong (0->0xFF)
- -40: pld is not working
- -41: error, system is in calibration
- -42: driver status is wrong
- -100 internal system error occurs see value of syserrno

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

Returns

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

2.17.1.2 `int MSXE360x__AnalogInputStopAndReleaseSequence (void * __, struct MSXE360x__Response * Response)`

Parameters

[in] `__` : no input parameter

[out] *Response* :

iReturnValue :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -2: pld is not working
- -3: error, system is in calibration
- -4: driver status is wrong
- -40: pld is not working
- -41: error, system is in calibration
- -42: driver status is wrong
- -100 internal system error occurs see value of `syserrno`

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

Returns

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

2.17.1.3 `int MSXE360x__AnalogInputGetSequenceStatus (void * __, struct MSXE360x__unsignedlongResponse * Response)`

Parameters

[in] `__` : no input parameter

[out] *Response* :

sResponse.iReturnValue :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -2: pld is not working
- -3: error, system is in calibration
- -4: driver status is wrong
- -100 internal system error occurs see value of `syserrno`

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

ulValue : Sequence Status :

- 0 : disable
- 1 : enable (in progress)
- 2 : end of the sequence
- 3 : enable, but wait for trigger
- 4 : pld overflow
- 5 : internal FIFO overflow

Returns

- 0: SOAP_OK
- <> 0: 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 occurred, 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 MSXE360x__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)

3.3.1 Field Documentation

3.3.1.1 xsd__int MSXE360x__Response::iReturnValue

- 0 means the remote function performed OK
- -1 means a system error occurred, the meaning of other values is function dependant and should be defined in the related header

3.3.1.2 xsd__int MSXE360x__Response::syserrno

3.4 MSXE360x__unsignedLong8FixedArrayParam Struct Reference

Data Fields

- [xsd__unsignedLong ulValue](#) [8]
the meaning of this value is defined in the related header of the function who use this type

3.4.1 Field Documentation

3.4.1.1 `xsd__unsignedLong MSXE360x__unsignedLong8FixedArrayParam::ulValue[8]`

3.5 MSXE360x__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

3.5.1 Field Documentation

3.5.1.1 `struct DefaultResponse MSXE360x__unsignedlongResponse::sResponse`

3.5.1.2 `xsd__unsignedLong MSXE360x__unsignedlongResponse::ulValue`

3.6 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.

3.6.1 Field Documentation

3.6.1.1 `struct DefaultResponse MXCommon__ByteArrayResponse::sResponse`

3.6.1.2 `struct ByteArray MXCommon__ByteArrayResponse::sArray`

3.7 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.

3.7.1 Field Documentation

3.7.1.1 struct [DefaultResponse MXCommon__FileResponse::sResponse](#)

3.7.1.2 struct [ByteArray MXCommon__FileResponse::sArray](#)

3.7.1.3 [xsd__unsignedLong MXCommon__FileResponse::ulEOF](#)

3.8 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.

3.8.1 Field Documentation

3.8.1.1 struct [DefaultResponse MXCommon__GetAutoConfigurationFileResponse::sResponse](#)

3.8.1.2 struct [ByteArray MXCommon__GetAutoConfigurationFileResponse::bArray](#)

3.8.1.3 [xsd__unsignedLong MXCommon__GetAutoConfigurationFileResponse::ulEOF](#)

3.9 MXCommon__GetEthernetLinksStatesResponse Struct Reference

Data Fields

- struct [DefaultResponse sResponse](#)
Default return values.

- struct [sGetEthernetLinksStatesPort sPort0](#)
- struct [sGetEthernetLinksStatesPort sPort1](#)

3.9.1 Field Documentation

3.9.1.1 struct `DefaultResponse MXCommon__GetEthernetLinksStatesResponse::sResponse`

3.9.1.2 struct `sGetEthernetLinksStatesPort MXCommon__GetEthernetLinksStatesResponse::sPort0`

3.9.1.3 struct `sGetEthernetLinksStatesPort MXCommon__GetEthernetLinksStatesResponse::sPort1`

3.10 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.

3.10.1 Field Documentation

3.10.1.1 struct `DefaultResponse MXCommon__GetHardwareTriggerFilterTimeResponse::sResponse`

3.10.1.2 `xsd__unsignedLong MXCommon__GetHardwareTriggerFilterTimeResponse::ulFilterTime`

3.10.1.3 `xsd__unsignedLong MXCommon__GetHardwareTriggerFilterTimeResponse::ulInfo01`

3.10.1.4 `xsd__unsignedLong MXCommon__GetHardwareTriggerFilterTimeResponse::ulInfo02`

3.11 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.

3.11.1 Field Documentation

3.11.1.1 `struct DefaultResponse MXCommon__GetHardwareTriggerStateResponse::sResponse`

3.11.1.2 `xsd__unsignedLong MXCommon__GetHardwareTriggerStateResponse::ulState`

3.11.1.3 `xsd__unsignedLong MXCommon__GetHardwareTriggerStateResponse::ulInfo01`

3.11.1.4 `xsd__unsignedLong MXCommon__GetHardwareTriggerStateResponse::ulInfo02`

3.12 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.

3.12.1 Field Documentation

- 3.12.1.1 struct `DefaultResponse` `MXCommon__GetModuleTemperatureValueAndStatusResponse::sResponse`
- 3.12.1.2 `xsd__double` `MXCommon__GetModuleTemperatureValueAndStatusResponse::dTemperatureValue`
- 3.12.1.3 `xsd__unsignedLong` `MXCommon__GetModuleTemperatureValueAndStatusResponse::ulTemperatureStatus`
- 3.12.1.4 `xsd__unsignedLong` `MXCommon__GetModuleTemperatureValueAndStatusResponse::ulInfo`

3.13 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).

3.13.1 Field Documentation

- 3.13.1.1 struct `DefaultResponse` `MXCommon__GetTimeResponse::sResponse`
- 3.13.1.2 `xsd__unsignedLong` `MXCommon__GetTimeResponse::ulLowTime`
- 3.13.1.3 `xsd__unsignedLong` `MXCommon__GetTimeResponse::ulHighTime`

3.14 MXCommon__GetUpTimeResponse Struct Reference

Data Fields

- struct `DefaultResponse` `sResponse`
Default return value.
- `xsd__unsignedLong` `ulUpTime`
Reserved.

3.14.1 Field Documentation

3.14.1.1 struct `DefaultResponse` `MXCommon__GetUpTimeResponse::sResponse`

3.14.1.2 `xsd__unsignedLong` `MXCommon__GetUpTimeResponse::ulUpTime`

3.15 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()`).

3.15.1 Field Documentation

3.15.1.1 `xsd__int` `MXCommon__Response::iReturnValue`

3.15.1.2 `xsd__int` `MXCommon__Response::syserrno`

3.16 MXCommon__TestCustomerIDResponse Struct Reference

Data Fields

- struct `DefaultResponse` `sResponse`

Default return values.

- struct `ByteArray` `bValueArray`

non encrypted value

- struct `ByteArray` `bCryptedValueArray`

encrypted value

3.16.1 Field Documentation

3.16.1.1 struct DefaultResponse MXCommon__TestCustomerIDResponse::sResponse

3.16.1.2 struct ByteArray MXCommon__TestCustomerIDResponse::bValueArray

3.16.1.3 struct ByteArray MXCommon__TestCustomerIDResponse::bCryptedValueArray

3.17 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.

3.17.1 Field Documentation

3.17.1.1 struct DefaultResponse MXCommon__unsignedLongResponse::sResponse

3.17.1.2 [xsd__unsignedLong](#) MXCommon__unsignedLongResponse::ulValue

3.18 sGetEthernetLinksStatesPort Struct Reference

Data Fields

- [xsd__unsignedLong](#) ulState
- [xsd__unsignedLong](#) ulSpeed
- [xsd__unsignedLong](#) ulDuplex
- [xsd__unsignedLong](#) ulInfo1
- [xsd__unsignedLong](#) ulInfo2

3.18.1 Field Documentation

3.18.1.1 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulState`

3.18.1.2 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulSpeed`

3.18.1.3 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulDuplex`

3.18.1.4 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo1`

3.18.1.5 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo2`

3.19 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

3.19.1 Field Documentation

3.19.1.1 `xsd__unsignedLong* UnsignedLongArray::__ptr`

3.19.1.2 `int UnsignedLongArray::__size`

3.19.1.3 `int UnsignedLongArray::__offset`

3.20 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

3.20.1 Field Documentation

3.20.1.1 `xsd__unsignedShort* UnsignedShortArray::__ptr`

3.20.1.2 `int UnsignedShortArray::__size`

3.20.1.3 `int UnsignedShortArray::__offset`

3.21 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

3.21.1 Field Documentation

3.21.1.1 `unsigned char* xsd__base64Binary::__ptr`

3.21.1.2 `int xsd__base64Binary::__size`

Chapter 4

File Documentation

4.1 MSXE360x_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 [MSXE360x__Response](#)
- struct [MSXE360x__unsignedlongResponse](#)
- struct [MSXE360x__unsignedLong8FixedArrayParam](#)

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__DataseverRestart (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 [MSXE360x__AnalogInputInitAndStartSequence](#) (xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulNbrOfSequence, xsd__unsignedLong ulNbrMaxSequenceToTransfer, xsd__double dFrequencySelection, struct [MSXE360x__unsignedLong8FixedArrayParam](#) *pulGainArray, xsd__unsignedLong ulICPMask, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulCouplingSelectionMask, xsd__unsignedLong ulSeDiffSelectionMask, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, xsd__unsignedLong ulOption5, xsd__unsignedLong ulOption6, struct [MSXE360x__Response](#) *Response)

Initialise and start the analog input sequence acquisition mode.

- int [MSXE360x__AnalogInputStopAndReleaseSequence](#) (void *_ , struct [MSXE360x__Response](#) *Response)

Stop and Release the analog input sequence acquisition mode.

- int [MSXE360x__AnalogInputGetSequenceStatus](#) (void *_ , struct [MSXE360x__unsignedlongResponse](#) *Response)

Get the analog input sequence acquisition status.

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** Reserved
- **ulInfo2** Reserved

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** Reserved
- **ulInfo2** Reserved

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__Sterror\(\)](#).
 - dValue : Temperature value in Degree Celsius
- ulTemperatureStatus : Temperature Status :
 - TEMPERATURE_INITIAL = 0 : Temperature not ready
 - TEMPERATURE_TOLOW = 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__Sterror\(\)](#).

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__Sterror\(\)](#).

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__Sterror\(\)](#).
- `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__Sterror\(\)](#). 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__Sterror\(\)](#).
 - 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: authentication 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__Sterror\(\)](#).
- 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__Sterror\(\)](#).
 - `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__Sterror\(\)](#).
 - `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 MSXE360x__AnalogInputInitAndStartSequence (xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulNbrOfSequence, xsd__unsignedLong ulNbrMaxSequenceToTransfer, xsd__double dFrequencySelection, struct MSXE360x__unsignedLong8FixedArrayParam * pulGainArray, xsd__unsignedLong ulICPMask, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulCouplingSelectionMask, xsd__unsignedLong ulSeDiffSelectionMask, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, xsd__unsignedLong ulOption4, xsd__unsignedLong ulOption5, xsd__unsignedLong ulOption6, struct MSXE360x__Response * Response)`

Parameters

[in] *ulChannelMask* : 8 bits mask (0->0xFF) which defines channels used for the acquisition (each bit corresponds to one channel)

- 0 : not used
- 1 : used

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

- 0 : Continuous mode
- > 0 : number of sequence (1 -> 4294967295 (0xFFFFFFFF))

[in] *ulNbrMaxSequenceToTransfer* : Not used, must be 0

[in] *dFrequencySelection* : Select the frequency of the acquisition

- 1000.00 Hz
- 1280.00 Hz
- 1562.50 Hz
- 1600.00 Hz
- 1666.67 Hz
- 2000.00 Hz
- 2500.00 Hz
- 3125.00 Hz
- 3200.00 Hz

- 3333.33 Hz
- 4000.00 Hz
- 5000.00 Hz
- 6250.00 Hz
- 6400.00 Hz
- 6666.67 Hz
- 8000.00 Hz
- 10000.00 Hz
- 12500.00 Hz
- 12800.00 Hz
- 13333.33 Hz
- 16000.00 Hz
- 16666.67 Hz
- 20000.00 Hz
- 25000.00 Hz
- 32000.00 Hz
- 33333.33 Hz
- 40000.00 Hz
- 50000.00 Hz
- 64000.00 Hz
- 66666.67 Hz
- 80000.00 Hz
- 100000.00 Hz
- 128000.00 Hz

[in] **pulGainArray** : Define the gain (1,10 or 100) to use for each channel.

Each index of the array corresponds to the corresponding channel :

example :

- [0] : Define the gain for the channel 0
- [1] : Define the gain for the channel 1
- ...

[in] **ulICPMask** : 8 bits mask (0->0xFF) which defines if the ICP is activated or not. (each bit corresponds to one channel)

When the ICP is activated, the channel must be configured with AC and SE.

- 0 : not activated
- 1 : activated

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

- 0 : trigger disabled
- 1 : Enable Hardware Digital Input Trigger
- 2 : Enable Synchro Trigger
- 3 : Enable both Hardware and Synchro Trigger

[in] **ulTriggerMode** : Not used, must be 0

[in] **ulHardwareTriggerEdge** : Define the edge of the trigger

- 1 : Hardware trigger rising edge
- 2 : Hardware trigger falling edge
- 3 : Enable both rising edge and falling edge

- [in] ***ulHardwareTriggerCount*** : Define the number of external trigger ignored before taking it account (1 -> 65535)
- [in] ***ulByTriggerNbrOfSeqToAcquire*** : Not used, must be 0
- [in] ***ulDataFormat*** : Dataformat of the frame, see remarks and examples for more informations :
 - D0 : Absolute time stamp information (2*32 bits data)
 - 0 : no time stamp information
 - 1 : time stamp information
 - D1 : Not used must be 0
 - D2 : Sequence counter (32 bits data)
 - 0: No sequence counter information required
 - 1: Sequence counter information required
 - D3 : Hardware trigger information (32 bits data)
 - 0 : No hardware trigger information required
 - 1 : Hardware trigger information required
- [in] ***ulCouplingSelectionMask*** : 8 bits mask (0->0xFF) which defines the coupling for each channel (each bit corresponds to one channel)
 - 0 : AC
 - 1 : DC
- [in] ***ulSeDiffSelectionMask*** : 8 bits mask (0->0xFF) which defines SE/DIFF mode for each channel (each bit corresponds to one channel)
 - 0 : SE
 - 1 : DIFF

Remark 1 : data packets depends on the (number of sequence asked) * (sequence size in 32 bits words) * 4

- <4-8192> bytes, Only one packet of the corresponding size is sent
- 0 (continuous) or > 8192 bytes : packet of 8192 bytes containing the sequences are sent by the MSXE, for the last sequence, the last packet is sent with the rest of the size

Remark 2 : the data order is

- timestamp seconds (optional)
- timestamp microseconds (optional)
- sequence counter (optional)
- hardware trigger information (optional)
- selected channels in ascending order

Remark 3 : sequence size in bytes

- [(timestamp (s) + timestamp (us)) (optional) + sequence counter (optional) + hardware trigger information (optional) + number of channels] * 4

Example 1 : 4 channels in continuous sequence mode

- packets of 8192 bytes are sent -> 512 sequences of 4 32 bits channels per packet

Example 2 : 10 sequences of 4 channels with timestamp and hardware trigger information.

- 1 packet of 280 bytes is sent by the MSXE -> 10 sequences of : timestamp (2*32 bits word) + sequence counter size (32 bits) + 4 * 32 bits channels)

Parameters

[out] **Response** :

iReturnValue :

- 0: means the remote function performed OK
- -1: means an system error occured
- -2: pld is not working
- -3: error, system is in calibration
- -4: channel action is wrong
- -5: gain selection error
- -6: channel coupling selection error
- -7: SE / Diff selection error
- -8: ICP selection error
- -9: ICP can only be used with AC and SE
- -10: driver is not in idle state
- -11: pld is not working
- -12: error, system is in calibration
- -13: Frequency selection error
- -14: driver is not in idle state
- -19: channel mask can not be null
- -20: channel mask selection error
- -21: number of sequence selection error
- -22: sequence interrupt selection error (must be 0)
- -23: the ulTriggerMode parameter is wrong (must be 0)
- -24: the ulHardwareTriggerEdge parameter is wrong
- -25: the ulHardwareTriggerCount parameter is wrong
- -26: the ulByTriggerNbrOfSeqToAcquire parameter is wrong (must be 0)
- -27: the ulDataFormat parameter is wrong
- -28: the ulTriggerMask parameter is wrong
- -29: the ulICPMask is wrong (0->0xFF)
- -30: the ulCouplingSelectionMask is wrong (0->0xFF)
- -31: the ulSeDiffSelectionMask is wrong (0->0xFF)
- -40: pld is not working
- -41: error, system is in calibration
- -42: driver status is wrong
- -100 internal system error occurs see value of syserrno

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

Returns

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

4.1.2.39 int MSXE360x__AnalogInputStopAndReleaseSequence (void * __, struct MSXE360x__Response * *Response*)

Parameters

[in] __ : no input parameter

[out] *Response* :

iReturnValue :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -2: pld is not working
- -3: error, system is in calibration
- -4: driver status is wrong
- -40: pld is not working
- -41: error, system is in calibration
- -42: driver status is wrong
- -100 internal system error occurs see value of syserrno

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

Returns

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

4.1.2.40 int MSXE360x__AnalogInputGetSequenceStatus (void * __, struct MSXE360x__unsignedlongResponse * *Response*)

Parameters

[in] __ : no input parameter

[out] *Response* :

sResponse.iReturnValue :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -2: pld is not working
- -3: error, system is in calibration
- -4: driver status is wrong
- -100 internal system error occurs see value of syserrno

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

ulValue : Sequence Status :

- 0 : disable
- 1 : enable (in progress)
- 2 : end of the sequence
- 3 : enable, but wait for trigger
- 4 : pld overflow
- 5 : internal FIFO overflow

Returns

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

Index

- __offset
 - ByteArray, [43](#)
 - UnsignedLongArray, [52](#)
 - UnsignedShortArray, [53](#)
 - __ptr
 - ByteArray, [43](#)
 - UnsignedLongArray, [52](#)
 - UnsignedShortArray, [53](#)
 - xsd__base64Binary, [53](#)
 - __size
 - ByteArray, [43](#)
 - UnsignedLongArray, [52](#)
 - UnsignedShortArray, [53](#)
 - xsd__base64Binary, [53](#)
- bArray
 - MXCommon__-
 - GetAutoConfigurationFileResponse, [46](#)
- bCryptedValueArray
 - MXCommon__TestCustomerIDResponse, [51](#)
- bValueArray
 - MXCommon__TestCustomerIDResponse, [51](#)
- ByteArray, [43](#)
 - __offset, [43](#)
 - __ptr, [43](#)
 - __size, [43](#)
- Common functions, [10](#)
- Common general functions, [11](#)
- Common hardware trigger functions, [18](#)
- Common I/O auto configuration functions, [25](#)
- Common security functions, [20](#)
- Common synchronisation timer functions, [27](#)
- Common temperature functions, [17](#)
- Common time functions, [22](#)
- Common_autoconf
 - MXCommon__GetAutoConfigurationFile, [26](#)
 - MXCommon__SetAutoConfigurationFile, [26](#)
 - MXCommon__StartAutoConfiguration, [27](#)
- Common_configuration
 - MXCommon__-
 - ApplyConfigurationBackupFile, [30](#)
 - MXCommon__ChangePassword, [31](#)
 - MXCommon__GetConfigurationBackupFile, [30](#)
 - Common_general
 - MXCommon__DataseverRestart, [15](#)
 - MXCommon__GetClientConnections, [13](#)
 - MXCommon__GetEthernetLinksStates, [16](#)
 - MXCommon__GetHostname, [12](#)
 - MXCommon__GetModuleType, [12](#)
 - MXCommon__Reboot, [15](#)
 - MXCommon__ResetAllIOFunctionalities, [15](#)
 - MXCommon__SetHostname, [13](#)
 - MXCommon__Strerror, [13](#)
 - Common_hardware_trigger
 - MXCommon__-
 - GetHardwareTriggerFilterTime, [19](#)
 - MXCommon__GetHardwareTriggerState, [20](#)
 - MXCommon__-
 - SetHardwareTriggerFilterTime, [19](#)
 - Common_security
 - MXCommon__SetCustomerKey, [22](#)
 - MXCommon__TestCustomerID, [22](#)
 - Common_synchrotimer
 - MXCommon__InitAndStartSynchroTimer, [28](#)
 - MXCommon__-
 - StopAndReleaseSynchroTimer, [28](#)
 - Common_temperature
 - MXCommon__-
 - GetModuleTemperatureValueAndStatus, [17](#)
 - MXCommon__-
 - SetModuleTemperatureWarningLevels, [18](#)
 - Common_time
 - MXCommon__GetTime, [24](#)
 - MXCommon__GetUpTime, [25](#)
 - MXCommon__HardwareClockToSys, [24](#)
 - MXCommon__SetTime, [23](#)
 - MXCommon__SysToHardwareClock, [23](#)
 - Customer option management, [34](#)
 - CustomerOption
 - MXCommon__GetOptionInformation, [35](#)
 - DefaultResponse, [43](#)
 - iReturn Value, [44](#)
 - syserrno, [44](#)

- dTemperatureValue
 - MXCommon__-
 - GetModuleTemperatureValueAndStatusResponse, 49
- iReturnValue
 - DefaultResponse, 44
 - MSXE360x__Response, 44
 - MXCommon__Response, 50
- MSX-E systems servers, 10
- MSX-E360x functions, 3
- MSX-E360x Sequence functions, 36
- MSXE360x__AnalogInputGetSequenceStatus
 - MSXE360x_public_doc.h, 82
 - MSXE360x__Sequence, 41
- MSXE360x__AnalogInputInitAndStartSequence
 - MSXE360x_public_doc.h, 78
 - MSXE360x__Sequence, 37
- MSXE360x__AnalogInputStopAndReleaseSequence
 - MSXE360x_public_doc.h, 81
 - MSXE360x__Sequence, 40
- MSXE360x__Response, 44
 - iReturnValue, 44
 - syserrno, 44
- MSXE360x__unsignedLong8FixedArrayParam, 44
 - ulValue, 45
- MSXE360x__unsignedlongResponse, 45
 - sResponse, 45
 - ulValue, 45
- MSXE360x_public_doc.h, 55
 - MSXE360x__-
 - AnalogInputGetSequenceStatus, 82
 - MSXE360x__-
 - AnalogInputInitAndStartSequence, 78
 - MSXE360x__-
 - AnalogInputStopAndReleaseSequence, 81
 - MXCommon__-
 - ApplyConfigurationBackupFile, 73
 - MXCommon__ChangePassword, 74
 - MXCommon__DataseverRestart, 63
 - MXCommon__GetAutoConfigurationFile, 70
 - MXCommon__GetClientConnections, 61
 - MXCommon__GetConfigurationBackupFile, 73
 - MXCommon__GetEthernetLinksStates, 64
 - MXCommon__-
 - GetHardwareTriggerFilterTime, 66
 - MXCommon__GetHardwareTriggerState, 67
 - MXCommon__GetHostname, 60
 - MXCommon__-
 - GetModuleTemperatureValueAndStatus, 65
 - MXCommon__GetModuleType, 60
 - MXCommon__GetOptionInformation, 76
 - MXCommon__GetStateIDFromName, 75
 - MXCommon__GetStateNameFromID, 76
 - MXCommon__GetSubsystemIDFromName, 75
 - MXCommon__GetSubsystemNameFromID, 76
 - MXCommon__GetSubSystemState, 74
 - MXCommon__GetSynchronizationStatus, 77
 - MXCommon__GetTime, 69
 - MXCommon__GetUpTime, 70
 - MXCommon__HardwareClockToSys, 69
 - MXCommon__InitAndStartSynchroTimer, 71
 - MXCommon__Reboot, 63
 - MXCommon__ResetAllIOFunctionalities, 63
 - MXCommon__SetAutoConfigurationFile, 71
 - MXCommon__SetCustomerKey, 67
 - MXCommon__-
 - SetHardwareTriggerFilterTime, 66
 - MXCommon__SetHostname, 61
 - MXCommon__-
 - SetModuleTemperatureWarningLevels, 65
 - MXCommon__SetTime, 68
 - MXCommon__SetToMaster, 77
 - MXCommon__StartAutoConfiguration, 71
 - MXCommon__-
 - StopAndReleaseSynchroTimer, 72
 - MXCommon__Sterror, 61
 - MXCommon__SysToHardwareClock, 69
 - MXCommon__TestCustomerID, 68
 - xsd__char, 60
 - xsd__double, 60
 - xsd__float, 60
 - xsd__int, 60
 - xsd__long, 60
 - xsd__string, 60
 - xsd__unsignedByte, 60
 - xsd__unsignedInt, 60
 - xsd__unsignedLong, 60
 - xsd__unsignedShort, 60
 - MSXE360x__Sequence
 - MSXE360x__-
 - AnalogInputGetSequenceStatus, 41
 - MSXE360x__-
 - AnalogInputInitAndStartSequence, 37
 - MSXE360x__-
 - AnalogInputStopAndReleaseSequence, 40
 - MXCommon__ApplyConfigurationBackupFile
 - Common_configuration, 30

- MSXE360x_public_doc.h, 73
- MXCommon__ByteArrayResponse, 45
 - sArray, 45
 - sResponse, 45
- MXCommon__ChangePassword
 - Common_configuration, 31
 - MSXE360x_public_doc.h, 74
- MXCommon__DataseverRestart
 - Common_general, 15
 - MSXE360x_public_doc.h, 63
- MXCommon__FileResponse, 45
 - sArray, 46
 - sResponse, 46
 - ulEOF, 46
- MXCommon__GetAutoConfigurationFile
 - Common_autoconf, 26
 - MSXE360x_public_doc.h, 70
- MXCommon__GetAutoConfigurationFileResponse, 46
 - bArray, 46
 - sResponse, 46
 - ulEOF, 46
- MXCommon__GetClientConnections
 - Common_general, 13
 - MSXE360x_public_doc.h, 61
- MXCommon__GetConfigurationBackupFile
 - Common_configuration, 30
 - MSXE360x_public_doc.h, 73
- MXCommon__GetEthernetLinksStates
 - Common_general, 16
 - MSXE360x_public_doc.h, 64
- MXCommon__GetEthernetLinksStatesResponse, 46
 - sPort0, 47
 - sPort1, 47
 - sResponse, 47
- MXCommon__GetHardwareTriggerFilterTime
 - Common_hardware_trigger, 19
 - MSXE360x_public_doc.h, 66
- MXCommon__GetHardwareTriggerFilterTimeResponse, 47
 - sResponse, 47
 - ulFilterTime, 47
 - ulInfo01, 47
 - ulInfo02, 47
- MXCommon__GetHardwareTriggerState
 - Common_hardware_trigger, 20
 - MSXE360x_public_doc.h, 67
- MXCommon__GetHardwareTriggerStateResponse, 47
 - sResponse, 48
 - ulInfo01, 48
 - ulInfo02, 48
 - ulState, 48
- MXCommon__GetHostname
 - Common_general, 12
 - MSXE360x_public_doc.h, 60
- MXCommon__GetModuleTemperatureValueAndStatus
 - Common_temperature, 17
 - MSXE360x_public_doc.h, 65
- MXCommon__GetModuleTemperatureValueAndStatusResponse, 48
 - dTemperatureValue, 49
 - sResponse, 49
 - ulInfo, 49
 - ulTemperatureStatus, 49
- MXCommon__GetModuleType
 - Common_general, 12
 - MSXE360x_public_doc.h, 60
- MXCommon__GetOptionInformation
 - CustomerOption, 35
 - MSXE360x_public_doc.h, 76
- MXCommon__GetStateIDFromName
 - MSXE360x_public_doc.h, 75
 - SystemStatemanagement, 33
- MXCommon__GetStateNameFromID
 - MSXE360x_public_doc.h, 76
 - SystemStatemanagement, 34
- MXCommon__GetSubsystemIDFromName
 - MSXE360x_public_doc.h, 75
 - SystemStatemanagement, 33
- MXCommon__GetSubsystemNameFromID
 - MSXE360x_public_doc.h, 76
 - SystemStatemanagement, 33
- MXCommon__GetSubSystemState
 - MSXE360x_public_doc.h, 74
 - SystemStatemanagement, 32
- MXCommon__GetSynchronizationStatus
 - MSXE360x_public_doc.h, 77
 - Synchronisation, 36
- MXCommon__GetTime
 - Common_time, 24
 - MSXE360x_public_doc.h, 69
- MXCommon__GetTimeResponse, 49
 - sResponse, 49
 - ulHighTime, 49
 - ulLowTime, 49
- MXCommon__GetUpTime
 - Common_time, 25
 - MSXE360x_public_doc.h, 70
- MXCommon__GetUpTimeResponse, 49
 - sResponse, 50
 - ulUpTime, 50
- MXCommon__HardwareClockToSys
 - Common_time, 24
 - MSXE360x_public_doc.h, 69
- MXCommon__InitAndStartSynchroTimer
 - Common_synchrotimer, 28

- MSXE360x_public_doc.h, 71
- MXCommon__Reboot
 - Common_general, 15
 - MSXE360x_public_doc.h, 63
- MXCommon__ResetAllIOFunctionalities
 - Common_general, 15
 - MSXE360x_public_doc.h, 63
- MXCommon__Response, 50
 - iReturnValue, 50
 - syserrno, 50
- MXCommon__SetAutoConfigurationFile
 - Common_autoconf, 26
 - MSXE360x_public_doc.h, 71
- MXCommon__SetCustomerKey
 - Common_security, 22
 - MSXE360x_public_doc.h, 67
- MXCommon__SetHardwareTriggerFilterTime
 - Common_hardware_trigger, 19
 - MSXE360x_public_doc.h, 66
- MXCommon__SetHostname
 - Common_general, 13
 - MSXE360x_public_doc.h, 61
- MXCommon__SetModuleTemperatureWarningLevels
 - Common_temperature, 18
 - MSXE360x_public_doc.h, 65
- MXCommon__SetTime
 - Common_time, 23
 - MSXE360x_public_doc.h, 68
- MXCommon__SetToMaster
 - MSXE360x_public_doc.h, 77
 - Synchronisation, 35
- MXCommon__StartAutoConfiguration
 - Common_autoconf, 27
 - MSXE360x_public_doc.h, 71
- MXCommon__StopAndReleaseSynchroTimer
 - Common_synchrotimer, 28
 - MSXE360x_public_doc.h, 72
- MXCommon__Sterror
 - Common_general, 13
 - MSXE360x_public_doc.h, 61
- MXCommon__SysToHardwareClock
 - Common_time, 23
 - MSXE360x_public_doc.h, 69
- MXCommon__TestCustomerID
 - Common_security, 22
 - MSXE360x_public_doc.h, 68
- MXCommon__TestCustomerIDResponse, 50
 - bCryptedValueArray, 51
 - bValueArray, 51
 - sResponse, 51
- MXCommon__unsignedLongResponse, 51
 - sResponse, 51
 - ulValue, 51
- sArray
 - MXCommon__ByteArrayResponse, 45
 - MXCommon__FileResponse, 46
- Set/Backup/Restore general system configuration, 29
- sGetEthernetLinksStatesPort, 51
 - ulDuplex, 52
 - ulInfo1, 52
 - ulInfo2, 52
 - ulSpeed, 52
 - ulState, 52
- SOAP function calls in C/C++ language, 3
- Software hints, 3
- sPort0
 - MXCommon__ -
 - GetEthernetLinksStatesResponse, 47
- sPort1
 - MXCommon__ -
 - GetEthernetLinksStatesResponse, 47
- sResponse
 - MSXE360x__unsignedlongResponse, 45
 - MXCommon__ByteArrayResponse, 45
 - MXCommon__FileResponse, 46
 - MXCommon__ -
 - GetAutoConfigurationFileResponse, 46
 - MXCommon__ -
 - GetEthernetLinksStatesResponse, 47
 - MXCommon__ -
 - GetHardwareTriggerFilterTimeResponse, 47
 - MXCommon__ -
 - GetHardwareTriggerStateResponse, 48
 - MXCommon__ -
 - GetModuleTemperatureValueAndStatusResponse, 49
 - MXCommon__GetTimeResponse, 49
 - MXCommon__GetUpTimeResponse, 50
 - MXCommon__TestCustomerIDResponse, 51
 - MXCommon__unsignedLongResponse, 51
- Synchronisation
 - MXCommon__GetSynchronizationStatus, 36
 - MXCommon__SetToMaster, 35
- Synchronisation management, 35
- syserrno
 - DefaultResponse, 44
 - MSXE360x__Response, 44
 - MXCommon__Response, 50
- System state management, 31
- SystemStatemanagement
 - MXCommon__GetStateIDFromName, 33
 - MXCommon__GetStateNameFromID, 34

- MXCommon__GetSubsystemIDFromName, 33
- MXCommon__GetSubsystemNameFromID, 33
- MXCommon__GetSubSystemState, 32
- ulDuplex
 - sGetEthernetLinksStatesPort, 52
- ulEOF
 - MXCommon__FileResponse, 46
 - MXCommon__-
 - GetAutoConfigurationFileResponse, 46
- ulFilterTime
 - MXCommon__-
 - GetHardwareTriggerFilterTimeResponse, 47
- ulHighTime
 - MXCommon__GetTimeResponse, 49
- ulInfo
 - MXCommon__-
 - GetModuleTemperatureValueAndStatusResponse, 49
- ulInfo01
 - MXCommon__-
 - GetHardwareTriggerFilterTimeResponse, 47
 - MXCommon__-
 - GetHardwareTriggerStateResponse, 48
- ulInfo02
 - MXCommon__-
 - GetHardwareTriggerFilterTimeResponse, 47
 - MXCommon__-
 - GetHardwareTriggerStateResponse, 48
- ulInfo1
 - sGetEthernetLinksStatesPort, 52
- ulInfo2
 - sGetEthernetLinksStatesPort, 52
- ulLowTime
 - MXCommon__GetTimeResponse, 49
- ulSpeed
 - sGetEthernetLinksStatesPort, 52
- ulState
 - MXCommon__-
 - GetHardwareTriggerStateResponse, 48
 - sGetEthernetLinksStatesPort, 52
- ulTemperatureStatus
 - MXCommon__-
 - GetModuleTemperatureValueAndStatusResponse, 49
- ulUpTime
 - MXCommon__GetUpTimeResponse, 50
- ulValue
 - MSXE360x__-
 - unsignedLong8FixedArrayParam, 45
 - MSXE360x__unsignedLongResponse, 45
 - MXCommon__unsignedLongResponse, 51
- UnsignedLongArray, 52
 - __offset, 52
 - __ptr, 52
 - __size, 52
- UnsignedShortArray, 52
 - __offset, 53
 - __ptr, 53
 - __size, 53
- xsd__base64Binary, 53
 - __ptr, 53
 - __size, 53
- xsd__char
 - MSXE360x_public_doc.h, 60
- xsd__double
 - MSXE360x_public_doc.h, 60
- xsd__float
 - MSXE360x_public_doc.h, 60
- xsd__int
 - MSXE360x_public_doc.h, 60
- xsd__long
 - MSXE360x_public_doc.h, 60
- xsd__string
 - MSXE360x_public_doc.h, 60
- xsd__unsignedByte
 - MSXE360x_public_doc.h, 60
- xsd__unsignedInt
 - MSXE360x_public_doc.h, 60
- xsd__unsignedLong
 - MSXE360x_public_doc.h, 60
- xsd__unsignedShort
 - MSXE360x_public_doc.h, 60