

## MSXE301x soap api functions

Generated by Doxygen 1.7.1

Wed Aug 19 2015 17:12:57



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Introduction	1
1.2	Remark: SOAP functions prototypes	1
<b>2</b>	<b>Module Documentation</b>	<b>3</b>
2.1	Software hints	3
2.2	SOAP function calls in C/C++ language	3
2.2.1	C/C++ language SOAP function prototypes	3
2.2.2	Rules and use of gSOAP calls in C/C++	4
2.2.3	SOAP calls sample	6
2.3	MSX-E systems servers	10
2.3.1	SOAP server	10
2.3.2	Event server	10
2.3.3	Modbus server	10
2.3.4	Data server	10
2.4	Common functions	10
2.5	Common general functions	11
2.5.1	Function Documentation	12
2.5.1.1	MXCommon__GetModuleType	12
2.5.1.2	MXCommon__GetHostname	12
2.5.1.3	MXCommon__SetHostname	13
2.5.1.4	MXCommon__GetClientConnections	13
2.5.1.5	MXCommon__Strerror	13
2.5.1.6	MXCommon__Reboot	15
2.5.1.7	MXCommon__ResetAllIOFunctionalities	15
2.5.1.8	MXCommon__DataseverRestart	15
2.5.1.9	MXCommon__GetEthernetLinksStates	16
2.6	Common temperature functions	17

2.6.1	Detailed Description	17
2.6.2	Function Documentation	17
2.6.2.1	MXCommon__GetModuleTemperatureValueAndStatus	17
2.6.2.2	MXCommon__SetModuleTemperatureWarningLevels	18
2.7	Common hardware trigger functions	18
2.7.1	Function Documentation	19
2.7.1.1	MXCommon__SetHardwareTriggerFilterTime	19
2.7.1.2	MXCommon__GetHardwareTriggerFilterTime	20
2.7.1.3	MXCommon__GetHardwareTriggerState	20
2.8	Common security functions	20
2.8.1	Detailed Description	21
2.8.2	Function Documentation	22
2.8.2.1	MXCommon__SetCustomerKey	22
2.8.2.2	MXCommon__TestCustomerID	22
2.9	Common time functions	22
2.9.1	Detailed Description	23
2.9.2	Function Documentation	23
2.9.2.1	MXCommon__SetTime	23
2.9.2.2	MXCommon__SysToHardwareClock	24
2.9.2.3	MXCommon__HardwareClockToSys	24
2.9.2.4	MXCommon__GetTime	24
2.9.2.5	MXCommon__GetUpTime	25
2.10	Common I/O auto configuration functions	25
2.10.1	Detailed Description	26
2.10.2	Function Documentation	26
2.10.2.1	MXCommon__GetAutoConfigurationFile	26
2.10.2.2	MXCommon__SetAutoConfigurationFile	26
2.10.2.3	MXCommon__StartAutoConfiguration	27
2.11	Common synchronisation timer functions	27
2.11.1	Function Documentation	28
2.11.1.1	MXCommon__InitAndStartSynchroTimer	28
2.11.1.2	MXCommon__StopAndReleaseSynchroTimer	29
2.12	Set/Backup/Restore general system configuration	29
2.12.1	Detailed Description	29
2.12.2	Function Documentation	30
2.12.2.1	MXCommon__GetConfigurationBackupFile	30

2.12.2.2	MXCommon__ApplyConfigurationBackupFile	30
2.12.2.3	MXCommon__ChangePassword	31
2.13	System state management	31
2.13.1	Detailed Description	32
2.13.2	Function Documentation	32
2.13.2.1	MXCommon__GetSubSystemState	32
2.13.2.2	MXCommon__GetSubsystemIDFromName	33
2.13.2.3	MXCommon__GetStateIDFromName	33
2.13.2.4	MXCommon__GetSubsystemNameFromID	34
2.13.2.5	MXCommon__GetStateNameFromID	34
2.14	Customer option management	34
2.14.1	Function Documentation	35
2.14.1.1	MXCommon__GetOptionInformation	35
2.15	Synchronisation management	35
2.15.1	Function Documentation	35
2.15.1.1	MXCommon__SetToMaster	35
2.15.1.2	MXCommon__GetSynchronizationStatus	36
2.16	input filter Filter management	36
2.16.1	Function Documentation	37
2.16.1.1	MXCommon__SetFilterChannels	37
2.17	MSX-E301x functions	37
2.18	MSX-E301x Information functions	37
2.18.1	Function Documentation	38
2.18.1.1	MSXE301x__AnalogInputGetChannelType	38
2.18.1.2	MSXE301x__AnalogMeasureGetConfiguration	38
2.19	MSX-E301x Autorefresh functions	39
2.19.1	Detailed Description	40
2.19.2	Function Documentation	40
2.19.2.1	MSXE301x__AnalogInputInitAndStartAutoRefresh	40
2.19.2.2	MSXE301x__AnalogInputGetAutoRefreshValues	43
2.19.2.3	MSXE301x__AnalogInputStopAndReleaseAutoRefresh	43
2.20	MSX-E301x Sequence functions	44
2.20.1	Detailed Description	44
2.20.2	Function Documentation	45
2.20.2.1	MSXE301x__AnalogInputInitAndStartSequence	45
2.20.2.2	MSXE301x__AnalogInputStopAndReleaseSequence	48

2.21	MSX-E301x calibration functions	48
2.21.1	Detailed Description	48
2.21.2	Function Documentation	49
2.21.2.1	MSXE301x__CalibrationStart	49
2.21.2.2	MSXE301x__CalibrationGetCurrentStatus	50
2.21.2.3	MSXE301x__CalibrationNextStep	50
2.21.2.4	MSXE301x__CalibrationBreak	51
<b>3</b>	<b>Data Structure Documentation</b>	<b>53</b>
3.1	ByteArray Struct Reference	53
3.1.1	Field Documentation	53
3.1.1.1	__ptr	53
3.1.1.2	__size	53
3.1.1.3	__offset	53
3.2	DefaultResponse Struct Reference	53
3.2.1	Field Documentation	54
3.2.1.1	iReturnValue	54
3.2.1.2	syserrno	54
3.3	MSXE301x__AnalogInputGetAutoRefreshValuesResponse Struct Reference	54
3.3.1	Field Documentation	55
3.3.1.1	sResponse	55
3.3.1.2	ulTimeStampLow	55
3.3.1.3	ulTimeStampHigh	55
3.3.1.4	ulCounterValue	55
3.3.1.5	ulValue	55
3.4	MSXE301x__AnalogInputGetChannelTypeResponse Struct Reference	55
3.4.1	Field Documentation	55
3.4.1.1	sResponse	55
3.4.1.2	ulType	55
3.5	MSXE301x__AnalogMeasureGetConfigurationResponse Struct Reference	55
3.5.1	Field Documentation	57
3.5.1.1	iReturnValue	57
3.5.1.2	ulRunningMode	57
3.5.1.3	ulConvertTimeUnit	57
3.5.1.4	ulConvertTime	57
3.5.1.5	ulChannelInitialization	57
3.5.1.6	sAcquisition	57

3.5.1.7	ulChannelMask	57
3.5.1.8	ulAverageMode	57
3.5.1.9	ulAverageValue	57
3.5.1.10	ulDataFormat	57
3.5.1.11	sAutoRefresh	58
3.5.1.12	ulSequenceSize	58
3.5.1.13	ulSequence	58
3.5.1.14	ulSequenceInterrupt	58
3.5.1.15	ulNbrOfSequence	58
3.5.1.16	ulDelayFlag	59
3.5.1.17	ulDelayMode	59
3.5.1.18	ulDelayTimeUnit	59
3.5.1.19	ulDelayTime	59
3.5.1.20	sSequence	59
3.5.1.21	ulSequenceTriggerCount	59
3.5.1.22	ulFlag	59
3.5.1.23	ulLevel	59
3.5.1.24	ulAction	59
3.5.1.25	ulCount	59
3.5.1.26	sHardware	59
3.5.1.27	sSoftware	59
3.5.1.28	sSynchro	59
3.5.1.29	sTrigger	59
3.6	MSXE301x__CalibrationGetCurrentStatusResponse Struct Reference	59
3.6.1	Field Documentation	60
3.6.1.1	sResponse	60
3.6.1.2	ulStatus	60
3.6.1.3	dCurrentValue	60
3.6.1.4	ulError	60
3.7	MSXE301x__Response Struct Reference	61
3.7.1	Field Documentation	61
3.7.1.1	iReturnValue	61
3.7.1.2	syserrno	62
3.8	MSXE301x__unsignedLong16FixedArrayParam Struct Reference	62
3.8.1	Field Documentation	62
3.8.1.1	ulValue	62

3.9	MSXE301x__unsignedLongResponse Struct Reference	62
3.9.1	Field Documentation	62
3.9.1.1	sResponse	62
3.9.1.2	ulValue	62
3.10	MXCommon__ByteArrayResponse Struct Reference	62
3.10.1	Field Documentation	63
3.10.1.1	sResponse	63
3.10.1.2	sArray	63
3.11	MXCommon__FileResponse Struct Reference	63
3.11.1	Field Documentation	63
3.11.1.1	sResponse	63
3.11.1.2	sArray	63
3.11.1.3	ulEOF	63
3.12	MXCommon__GetAutoConfigurationFileResponse Struct Reference	63
3.12.1	Field Documentation	64
3.12.1.1	sResponse	64
3.12.1.2	bArray	64
3.12.1.3	ulEOF	64
3.13	MXCommon__GetEthernetLinksStatesResponse Struct Reference	64
3.13.1	Field Documentation	64
3.13.1.1	sResponse	64
3.13.1.2	sPort0	64
3.13.1.3	sPort1	64
3.14	MXCommon__GetHardwareTriggerFilterTimeResponse Struct Reference	64
3.14.1	Field Documentation	65
3.14.1.1	sResponse	65
3.14.1.2	ulFilterTime	65
3.14.1.3	ulInfo01	65
3.14.1.4	ulInfo02	65
3.15	MXCommon__GetHardwareTriggerStateResponse Struct Reference	65
3.15.1	Field Documentation	65
3.15.1.1	sResponse	65
3.15.1.2	ulState	65
3.15.1.3	ulInfo01	65
3.15.1.4	ulInfo02	65
3.16	MXCommon__GetModuleTemperatureValueAndStatusResponse Struct Reference	65



3.16.1	Field Documentation	66
3.16.1.1	sResponse	66
3.16.1.2	dTemperatureValue	66
3.16.1.3	ulTemperatureStatus	66
3.16.1.4	ulInfo	66
3.17	MXCommon__GetTimeResponse Struct Reference	66
3.17.1	Field Documentation	66
3.17.1.1	sResponse	66
3.17.1.2	ulLowTime	66
3.17.1.3	ulHighTime	66
3.18	MXCommon__GetUpTimeResponse Struct Reference	66
3.18.1	Field Documentation	67
3.18.1.1	sResponse	67
3.18.1.2	ulUpTime	67
3.19	MXCommon__Response Struct Reference	67
3.19.1	Field Documentation	67
3.19.1.1	iReturnValue	67
3.19.1.2	syserrno	67
3.20	MXCommon__TestCustomerIDResponse Struct Reference	67
3.20.1	Field Documentation	68
3.20.1.1	sResponse	68
3.20.1.2	bValueArray	68
3.20.1.3	bCryptedValueArray	68
3.21	MXCommon__unsignedLongResponse Struct Reference	68
3.21.1	Field Documentation	68
3.21.1.1	sResponse	68
3.21.1.2	ulValue	68
3.22	sGetEthernetLinksStatesPort Struct Reference	68
3.22.1	Field Documentation	69
3.22.1.1	ulState	69
3.22.1.2	ulSpeed	69
3.22.1.3	ulDuplex	69
3.22.1.4	ulInfo1	69
3.22.1.5	ulInfo2	69
3.23	UnsignedLongArray Struct Reference	69
3.23.1	Field Documentation	69

3.23.1.1	<code>__ptr</code>	69
3.23.1.2	<code>__size</code>	69
3.23.1.3	<code>__offset</code>	69
3.24	UnsignedShortArray Struct Reference	69
3.24.1	Field Documentation	70
3.24.1.1	<code>__ptr</code>	70
3.24.1.2	<code>__size</code>	70
3.24.1.3	<code>__offset</code>	70
3.25	xsd_base64Binary Struct Reference	70
3.25.1	Field Documentation	70
3.25.1.1	<code>__ptr</code>	70
3.25.1.2	<code>__size</code>	70
<b>4</b>	<b>File Documentation</b>	<b>71</b>
4.1	MSXE301x_public_doc.h File Reference	71
4.1.1	Typedef Documentation	77
4.1.1.1	<code>xsd_string</code>	77
4.1.1.2	<code>xsd_char</code>	77
4.1.1.3	<code>xsd_float</code>	77
4.1.1.4	<code>xsd_double</code>	77
4.1.1.5	<code>xsd_int</code>	77
4.1.1.6	<code>xsd_long</code>	77
4.1.1.7	<code>xsd_unsignedByte</code>	77
4.1.1.8	<code>xsd_unsignedInt</code>	77
4.1.1.9	<code>xsd_unsignedShort</code>	77
4.1.1.10	<code>xsd_unsignedLong</code>	77
4.1.2	Function Documentation	77
4.1.2.1	<code>MXCommon__GetModuleType</code>	77
4.1.2.2	<code>MXCommon__GetHostname</code>	77
4.1.2.3	<code>MXCommon__SetHostname</code>	78
4.1.2.4	<code>MXCommon__GetClientConnections</code>	78
4.1.2.5	<code>MXCommon__Strerror</code>	79
4.1.2.6	<code>MXCommon__Reboot</code>	80
4.1.2.7	<code>MXCommon__ResetAllIOFunctionalities</code>	80
4.1.2.8	<code>MXCommon__DataseverRestart</code>	81
4.1.2.9	<code>MXCommon__GetEthernetLinksStates</code>	81
4.1.2.10	<code>MXCommon__GetModuleTemperatureValueAndStatus</code>	82

4.1.2.11	MXCommon__SetModuleTemperatureWarningLevels . . . . .	83
4.1.2.12	MXCommon__SetHardwareTriggerFilterTime . . . . .	83
4.1.2.13	MXCommon__GetHardwareTriggerFilterTime . . . . .	84
4.1.2.14	MXCommon__GetHardwareTriggerState . . . . .	84
4.1.2.15	MXCommon__SetCustomerKey . . . . .	85
4.1.2.16	MXCommon__TestCustomerID . . . . .	85
4.1.2.17	MXCommon__SetTime . . . . .	85
4.1.2.18	MXCommon__SysToHardwareClock . . . . .	86
4.1.2.19	MXCommon__HardwareClockToSys . . . . .	86
4.1.2.20	MXCommon__GetTime . . . . .	87
4.1.2.21	MXCommon__GetUpTime . . . . .	87
4.1.2.22	MXCommon__GetAutoConfigurationFile . . . . .	87
4.1.2.23	MXCommon__SetAutoConfigurationFile . . . . .	88
4.1.2.24	MXCommon__StartAutoConfiguration . . . . .	88
4.1.2.25	MXCommon__InitAndStartSynchroTimer . . . . .	88
4.1.2.26	MXCommon__StopAndReleaseSynchroTimer . . . . .	89
4.1.2.27	MXCommon__GetConfigurationBackupFile . . . . .	90
4.1.2.28	MXCommon__ApplyConfigurationBackupFile . . . . .	91
4.1.2.29	MXCommon__ChangePassword . . . . .	91
4.1.2.30	MXCommon__GetSubSystemState . . . . .	92
4.1.2.31	MXCommon__GetSubsystemIDFromName . . . . .	92
4.1.2.32	MXCommon__GetStateIDFromName . . . . .	92
4.1.2.33	MXCommon__GetSubsystemNameFromID . . . . .	93
4.1.2.34	MXCommon__GetStateNameFromID . . . . .	93
4.1.2.35	MXCommon__GetOptionInformation . . . . .	94
4.1.2.36	MXCommon__SetToMaster . . . . .	94
4.1.2.37	MXCommon__GetSynchronizationStatus . . . . .	94
4.1.2.38	MXCommon__SetFilterChannels . . . . .	95
4.1.2.39	MSXE301x__AnalogInputGetChannelType . . . . .	95
4.1.2.40	MSXE301x__AnalogMeasureGetConfiguration . . . . .	96
4.1.2.41	MSXE301x__AnalogInputInitAndStartAutoRefresh . . . . .	97
4.1.2.42	MSXE301x__AnalogInputGetAutoRefreshValues . . . . .	99
4.1.2.43	MSXE301x__AnalogInputStopAndReleaseAutoRefresh . . . . .	100
4.1.2.44	MSXE301x__AnalogInputInitAndStartSequence . . . . .	100
4.1.2.45	MSXE301x__AnalogInputStopAndReleaseSequence . . . . .	103
4.1.2.46	MSXE301x__CalibrationStart . . . . .	103

4.1.2.47	<a href="#">MSXE301x__CalibrationGetCurrentStatus</a>	104
4.1.2.48	<a href="#">MSXE301x__CalibrationNextStep</a>	104
4.1.2.49	<a href="#">MSXE301x__CalibrationBreak</a>	105

# 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 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.*

### 2.2 SOAP function calls in C/C++ language

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

#### 2.2.1 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.

### 2.2.2 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.sResponse.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.iReturnValue <= -100) && tempResponse.sResponse.syserrno))
{
    soap_call_MXCommon__Strerror (soapContext, soap_endpoint, soap_action, tempResponse.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 and release the SOAP context
soap_free (soapContext);
```

### 2.2.3 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.

@returnval 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).

@returnval 0: No error.
@returnval 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

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

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

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

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

TEMPERATURE_SRC         := temperature.o
BINS                    := temperature

all: $(BINS)

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

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

```

## 2.3 MSX-E systems servers

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

### 2.3.1 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.

### 2.3.2 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.

### 2.3.3 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.

### 2.3.4 Data server

The MSX-E301x embeds a **data server** that can be used to read acquired values. The default port number is set on 8989 and can be changed by using the **Data server** MSX-E Web interface.

## 2.4 Common functions

### Modules

- [Common general functions](#)

*Various utility functions, mainly to identify a remote system.*

- [Common temperature functions](#)

*These functions deals with the internal temperature sub-system.*

- [Common hardware trigger functions](#)

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

- [Common security functions](#)

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

- [Common time functions](#)

*A MSX-E module provides a "system clock" that may be in the simplest case set by the function [MXCommon\\_\\_SetTime\(\)](#).*

- [Common I/O auto configuration functions](#)

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

- [Common synchronisation timer functions](#)

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

- [Set/Backup/Restore general system configuration](#)

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

- [System state management](#)

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

- [Customer option management](#)

*Enable to get informations about the options of the system.*

- [Synchronisation management](#)

*Manage the synchronisation state of the system.*

- [input filter Filter management](#)

*Manages the analog input filters in the system.*

## 2.5 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__Sterror (xsd__int ernum, 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.*

## 2.5.1 Function Documentation

### 2.5.1.1 `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

### 2.5.1.2 `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

### 2.5.1.3 `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

### 2.5.1.4 `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

### 2.5.1.5 `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

**2.5.1.6 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

**2.5.1.7 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

**2.5.1.8 int MXCommon\_\_DataseverRestart ( 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.

### 2.5.1.9 int MXCommon\_\_GetEthernetLinksStates ( void \* \_, struct MXCommon\_\_GetEthernetLinksStatesResponse \* Response )

#### Parameters

[in] *\_* : no input parameter

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

*sResponse.iReturnValue*

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

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

*sPort0: Fisrt port informations*

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

*sPort1: Second port informations*

- **ulState**
  - **0** Link down

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

**Return values**

0 SOAP\_OK

*Others* See SOAP error

## 2.6 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.*

### 2.6.1 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.

### 2.6.2 Function Documentation

- #### 2.6.2.1 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

**2.6.2.2 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

## 2.7 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.*

### 2.7.1 Function Documentation

#### 2.7.1.1 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

### 2.7.1.2 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

### 2.7.1.3 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

## 2.8 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 ).*

### 2.8.1 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]

$[B] = \text{AES}([A], K1, K2)$

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.

## 2.8.2 Function Documentation

### 2.8.2.1 `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

### 2.8.2.2 `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

## 2.9 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).*

### 2.9.1 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.

### 2.9.2 Function Documentation

#### 2.9.2.1 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

### 2.9.2.2 `int MXCommon__SysToHardwareClock ( void * _, struct MXCommon__Response * Response )`

#### Parameters

- [in] \_ No input parameter
- [out] **Response**
- sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

#### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

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

### 2.9.2.3 `int MXCommon__HardwareClockToSys ( void * _, struct MXCommon__Response * Response )`

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

#### Parameters

- [in] \_ No input parameter
- [out] **Response**
- sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

#### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

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

### 2.9.2.4 `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

### 2.9.2.5 `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

## 2.10 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*

### 2.10.1 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.

### 2.10.2 Function Documentation

#### 2.10.2.1 `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

#### 2.10.2.2 `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

### 2.10.2.3 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

## 2.11 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)*

## 2.11.1 Function Documentation

**2.11.1.1** `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



### 2.11.1.2 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

## 2.12 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.*

### 2.12.1 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"

## 2.12.2 Function Documentation

### 2.12.2.1 `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;
}
*
```

### 2.12.2.2 `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 uLEOF must be set to 1. The new configuration is then applied.

**2.12.2.3** `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.

## 2.13 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".

### 2.13.1 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.

### 2.13.2 Function Documentation

#### 2.13.2.1 `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

**2.13.2.2** `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

**2.13.2.3** `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

### 2.13.2.4 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

### 2.13.2.5 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

## 2.14 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.*

## 2.14.1 Function Documentation

**2.14.1.1** `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

## 2.15 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.*

## 2.15.1 Function Documentation

**2.15.1.1** `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` system error code (the value of the libc "errno" code)

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

**2.15.1.2** `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

## 2.16 input filter Filter management

Manages the analog input filters in the system.

#### Functions

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

*This function sets or resets a filter to a channel.*



## 2.16.1 Function Documentation

**2.16.1.1** `int MXCommon_SetFilterChannels ( struct xsd__base64Binary * ChannelList, struct MXCommon_Response * Response )`

### Parameters

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

[out] ***Response***

- sResponse.iReturnValue : Return value
  - 0 : success
  - -1: system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Sterror\(\)](#).

### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

## 2.17 MSX-E301x functions

### Modules

- [MSX-E301x Information functions](#)
- [MSX-E301x Autorefresh functions](#)

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

- [MSX-E301x Sequence functions](#)

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

- [MSX-E301x calibration functions](#)

*The calibration functions permit to calibrate the analog input of the module.*

## 2.18 MSX-E301x Information functions

### Data Structures

- struct [MSXE301x\\_\\_AnalogInputGetChannelTypeResponse](#)

### Functions

- int [MSXE301x\\_\\_AnalogInputGetChannelType](#) (xsd\_\_unsignedLong ulOption1, struct [MSXE301x\\_\\_AnalogInputGetChannelTypeResponse](#) \*Response)

*Return the type of the analog input channels.*

- `int MSXE301x__AnalogMeasureGetConfiguration (xsd__unsignedLong ulModule, struct MSXE301x__AnalogMeasureGetConfigurationResponse *Response)`

*Return the running configuration.*

## 2.18.1 Function Documentation

### 2.18.1.1 `int MSXE301x__AnalogInputGetChannelType (xsd__unsignedLong ulOption1, struct MSXE301x__AnalogInputGetChannelTypeResponse * Response )`

#### Parameters

[in] *ulOption1* : Reserved

[out] *Response* :

*iReturnValue* :

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

*syserrno* : system-error code (the value of the libc "errno" code) *ulType* : Array that contain the channels type (0 : voltage, 1 : current)

- *ulType* [0] : Channel 0 type
- ...
- *ulType* [15] : Channel 15 type

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

### 2.18.1.2 `int MSXE301x__AnalogMeasureGetConfiguration (xsd__unsignedLong ulModule, struct MSXE301x__AnalogMeasureGetConfigurationResponse * Response )`

#### Parameters

[in] *ulOption1* : Reserved

[out] *Response* :

*iReturnValue* :

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

*sAcquisition* :

- *ulConvertTimeUnit* 0: micro second 1: milli second 2: second
- *pul\_ConvertTimeValue* : (10,65535)
- *pul\_ChannelInitialisation* (Are channels initialised and how) pointer to unsigned long[16]. bit 31 = initialised : (0,1) bit 30= polarity : (0,1) - 0:bipolar, 1:unipolar bits 29-0 : gain : (1,2) - 0: 10 V , 1: 5V

*sAutoRefresh* :

- *ulChannelMask* Mask of channels used for the auto refresh mode
- *ulAverageMode* Determine average mode 0: Sequence 1 : Channel
- *ulAverageValue* Determine the number of sequence for a average value calculation

- *ulDataFormat* D0: Time stamp information. 0 without, 1 with. D1 : Data format 0: digital 1: analog (in V)

***sSequence :***

- *ulSequenceSize* Sequence array size
- *ulSequence* Sequence channel array
- *ulSequenceInterrupt* Determine the number of sequences for the interrupt
- *ulNbrOfSequence* Determine the number total of sequences. 0: Continuous
- *ulDelayFlag* 0 : Disabled 1 : Enabled
- *ulDelayMode* 0 : delay not used 1 : mode 1 2 : mode 2
- *ulDelayTimeUnit* Delay time unit 1: micros 2 : ms 3 : s
- *ulDelayTime* Delay time
- *ulDataFormat* D0: Time stamp information. 0 without, 1 with.  
D1 : Sequence counter information. 0 without, 1 with.  
D2 : Data format 0: digital 1: analog (in V)

**Returns**

- 0: SOAP\_OK
- <> 0: See SOAP error

## 2.19 MSX-E301x Autorefresh functions

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

### Data Structures

- struct [MSXE301x\\_\\_AnalogInputGetAutoRefreshValuesResponse](#)

### Functions

- int [MSXE301x\\_\\_AnalogInputInitAndStartAutoRefresh](#) (xsd\_\_unsignedLong ulChannelMask, struct [MSXE301x\\_\\_unsignedLong16FixedArrayParam](#) \*pulGainArray, struct [MSXE301x\\_\\_unsignedLong16FixedArrayParam](#) \*pulPolarityArray, xsd\_\_unsignedLong ulAverageMode, xsd\_\_unsignedLong ulAverageValue, xsd\_\_unsignedLong ulConversionTime, xsd\_\_unsignedLong ulConversionTimeUnit, xsd\_\_unsignedLong ulTriggerMask, xsd\_\_unsignedLong ulTriggerMode, xsd\_\_unsignedLong ulHardwareTriggerEdge, xsd\_\_unsignedLong ulHardwareTriggerCount, xsd\_\_unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd\_\_unsignedLong ulDataFormat, xsd\_\_unsignedLong ulOption1, xsd\_\_unsignedLong ulOption2, xsd\_\_unsignedLong ulOption3, struct [MSXE301x\\_\\_Response](#) \*Response)

*Starts an autorefresh acquisition using provided configuration.*

- int [MSXE301x\\_\\_AnalogInputGetAutoRefreshValues](#) (void \*\_ , struct [MSXE301x\\_\\_AnalogInputGetAutoRefreshValuesResponse](#) \*Response)

*Returns the values acquired in auto refresh mode.*

- int [MSXE301x\\_\\_AnalogInputStopAndReleaseAutoRefresh](#) (void \*\_ , struct [MSXE301x\\_\\_Response](#) \*Response)

*Stops the current autorefresh acquisition.*

### 2.19.1 Detailed Description

The analog acquisition is initialised and the values of each channels are stored in memory on the Ethernet E/A module MSX-E301x.

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

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

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

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

You can start the acquisition by a hardware trigger or a synchro trigger.

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

You have the following possibility:

- Defining a number of edges before a trigger action is generated

There are two trigger modes:(for the hardware or synchro trigger)

a) One shot

b) Sequence

a) One shot:

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

b) Sequence:

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

### 2.19.2 Function Documentation

**2.19.2.1** `int MSXE301x__AnalogInputInitAndStartAutoRefresh ( xsd__unsignedLong ulChannelMask, struct MSXE301x__unsignedLong16FixedArrayParam * pulGainArray, struct MSXE301x__unsignedLong16FixedArrayParam * pulPolarityArray, xsd__unsignedLong ulAverageMode, xsd__unsignedLong ulAverageValue, xsd__unsignedLong ulConversionTime, xsd__unsignedLong ulConversionTimeUnit, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, struct MSXE301x__Response * Response )`

#### Parameters

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

[in] ***pulGainArray*** Define the gain (1 or 2) to use for each channel.

Information : by the channel type "current", the gain 2 is recommended.

- 1 : +10 V or max 20mA when Unipolar, +/-10V or +/- max 20mA when Bipolar
- 2 : +5 V or 20mA when Unipolar, +/-5V or +/- 20mA when Bipolar Each index of the array correspond to the channel :

sample :

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

Remark: When using a current version of the MSX-E301x, gain 2 has to be used.

[in] ***pulPolarityArray*** Define the polarity (0:Unipolar or 1:Bipolar) to use for each channel Each index of the array correspond to the channel :

sample :

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

Remark: When using the unipolar mode, the raw value for 0 volt (or 0 ampere) will be 32767.

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

- 0 : not used
- 1 : average per Sequence
- 2 : average per channel

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

- 0 : not used
- max value : 255

[in] ***ulConversionTime*** Conversion Time

- range from min 10 to 65535 when the unit is the microsecond
- range from min 1 to 65535 when the unit is the millisecond
- range from min 1 to 65535 when the unit is the second

[in] ***ulConversionTimeUnit*** Conversion Time Unit

- 0 : microsecond
- 1 : millisecond
- 2 : second

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

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

[in] ***ulTriggerMode*** Define the trigger mode

- 1 : One shot trigger
- 2 : Sequence trigger

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

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

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

- 0 : continuous mode
- <> 0 : number of sequence : (1..0xFFFFFFFF)

[in] ***ulDataFormat*** D0 : Time stamp information

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

D1 : Data format

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

[in] ***ulOption1*** Reserved

[in] ***ulOption2*** Reserved

[in] ***ulOption3*** Reserved

[out] ***Response*** :

***iReturnValue*** :

- 0 : means the remote function performed OK
- -1: means an system error occurred
- -2: The channel mask cannot be null
- -3: Channel Mask error
- -4: Gain selection error
- -5: Polarity selection error
- -6: not available average mode
- -7: not available average value
- -8: The minimal converting time is 10 us !
- -9: Not available conversion time unit
- -10: Trigger mode : 2 different mode cannot be simultaneously be activated
- -11: Hardware trigger : front definition error
- -12: Hardware trigger count value not available
- -13: Nbr of sequence to acquire by trigger mode not available
- -14: Data format not available
- -100: Channel initialisation kernel function error
- -101: Conversion time initialisation kernel function error
- -102: Average mode initialisation kernel function error
- -103: hardware trigger initialisation/enable kernel function error
- -104: hardware trigger disable kernel function error
- -105: synchro trigger initialisation/enable kernel function error
- -106: synchro trigger disable kernel function error
- -107 Sequence trigger count initialisation error
- -108: Start auto refresh kernel function error

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

## Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

### 2.19.2.2 int MSXE301x\_\_AnalogInputGetAutoRefreshValues ( void \* \_\_, struct MSXE301x\_\_AnalogInputGetAutoRefreshValuesResponse \* *Response* )

#### Parameters

[in] \_\_ Dummy parameter

[out] *Response* :

*iReturnValue* :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: GetAutoRefreshAllValues kernel function error

*ulTimeStampLow* : number of microseconds since the begin of the second

*ulTimeStampHigh* : number of seconds since the Epoch

*ulCounterValue* : Array that contain the counter values

*ulValue* : Array that contain the channels values

- ulValues [0] : Channel 0 value
- ...
- ulValues [15] : Channel 15 value

Remark: When using the unipolar mode, the raw value for 0 volt (or 0 ampere) will be 32767.

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

### 2.19.2.3 int MSXE301x\_\_AnalogInputStopAndReleaseAutoRefresh ( void \* \_\_, struct MSXE301x\_\_Response \* *Response* )

#### Parameters

[in] \_\_ Dummy parameter

[out] *Response* :

*iReturnValue* :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: StopAutoRefresh kernel function error

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

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

Must be called before any another call to MSXE301x\_\_AnalogInputInitAndStartAutoRefresh.

## 2.20 MSX-E301x Sequence functions

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

### Functions

- `int MSXE301x__AnalogInputInitAndStartSequence (xsd__unsignedLong ulNbrOfChannel, struct MSXE301x__unsignedLong16FixedArrayParam *pulChannelList, struct MSXE301x__unsignedLong16FixedArrayParam *pulGainArray, struct MSXE301x__unsignedLong16FixedArrayParam *pulPolarityArray, xsd__unsignedLong ulConversionTime, xsd__unsignedLong ulConversionTimeUnit, xsd__unsignedLong ulNbrOfSequence, xsd__unsignedLong ulNbrMaxSequenceToTransfer, xsd__unsignedLong ulDelayMode, xsd__unsignedLong ulDelayTimeUnit, xsd__unsignedLong ulDelayValue, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, struct MSXE301x__Response *Response)`  
*Initialise and start the analog input sequence acquisition mode.*

- `int MSXE301x__AnalogInputStopAndReleaseSequence (void *_ , struct MSXE301x__Response *Response)`  
*Stop and release the analog input sequence acquisition mode.*

### 2.20.1 Detailed Description

It can be any order of the channels in this list.

There are different sequence modes:

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

a) Certain number of sequences:

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

b) Continuous:

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

c) Without delay:

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

d) With delay:

A delay between 2 sequences can be configured:

For this there are 2 delay types:

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

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

You can start the acquisition by a hardware or synchro trigger.



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

You have the following possibility:

- Defining a number of edges before a trigger action is generated

There are two trigger modes (for the hardware or synchro trigger):

a) One shot

b) Sequence

a) One shot:

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

b) Sequence:

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

## 2.20.2 Function Documentation

**2.20.2.1** `int MSXE301x__AnalogInputInitAndStartSequence ( xsd_unsignedLong ulNbrOfChannel, struct MSXE301x__unsignedLong16FixedArrayParam * pulChannelList, struct MSXE301x__unsignedLong16FixedArrayParam * pulGainArray, struct MSXE301x__unsignedLong16FixedArrayParam * pulPolarityArray, xsd_unsignedLong ulConversionTime, xsd_unsignedLong ulConversionTimeUnit, xsd_unsignedLong ulNbrOfSequence, xsd_unsignedLong ulNbrMaxSequenceToTransfer, xsd_unsignedLong ulDelayMode, xsd_unsignedLong ulDelayTimeUnit, xsd_unsignedLong ulDelayValue, xsd_unsignedLong ulTriggerMask, xsd_unsignedLong ulTriggerMode, xsd_unsignedLong ulHardwareTriggerEdge, xsd_unsignedLong ulHardwareTriggerCount, xsd_unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd_unsignedLong ulDataFormat, xsd_unsignedLong ulOption1, xsd_unsignedLong ulOption2, xsd_unsignedLong ulOption3, struct MSXE301x__Response * Response )`

### Parameters

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

[in] *pulChannelList* : list of the channel who compose the sequence

[in] *pulGainArray* Define the gain (1 or 2) to use for each channel.

Information : by the channel type "current", the gain 2 is recommended.

- 1 : +10 V or max 20mA when Unipolar, +/-10V or max +/-20mA when Bipolar
  - 2 : +5 V or 20 mA when Unipolar, +/-5V or +/- 20mA when Bipolar Each index of the array correspond to the channel :
- sample :
- [0] : Define the gain for the channel 0
  - [1] : Define the gain for the channel 1
  - ...

Remark: When using a current version of the MSX-E301x, gain 2 has to be used.

[in] *pulPolarityArray* Define the polarity (0:Unipolar or 1:Bipolar) to use for each channel

Each index of the array correspond to the channel :

sample :

- [0] : Define the polarity for the channel 0
  - [1] : Define the polarity for the channel 1
  - ...
- Remark: When using the unipolar mode, the raw value for 0 volt (or 0 ampere) will be 32767.

[in] ***ulConversionTime*** Conversion Time (min 10 us or 40 us)

[in] ***ulConversionTimeUnit*** Conversion Time Unit

- 0 : us
- 1 : ms

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

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

[in] ***ulNbrMaxSequenceToTransfer*** : Max nbr of sequence to acquire before a data transfer : (1,65535)

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

- 0 : delay not used
- 1 : mode 1
- 2 : mode 2

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

0: us  
1: ms  
2: s

[in] ***ulDelayValue*** : Delay Value : (1..0xFFFF)

[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*** Define the trigger mode

- 1 : One shot trigger
- 2 : Sequence trigger

[in] ***ulHardwareTriggerEdge*** Define the edge of the hardware trigger who generate a trigger action

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

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

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

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

- 0 : continuous mode
- <> 0 : number of sequence : (1..0xFFFFFFFF)

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

D0 : Time stamp information

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

D1 : Sequence counter information

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

D2 : Data format

- 0 : 32/16 bit digital value
- 1 : 32-bit analog value (in V)

D3 : 16-Bit digital value

- 0 : 32-bit digital/analog value
- 1 : 16-bit digital value.

Only a pair number of acquisition channels (ulNbrOfChannel) can be selected.

[in] **ulOption1** : Reserved

[in] **ulOption2** : Reserved

[in] **ulOption3** : Reserved

[out] **Response** :

**iReturnValue** :

- 0: means the remote function performed OK
- -1: means an system error occured
- -2: The nbr of channel in the sequence cannot be null
- -3: Channel index selection error
- -4: Gain selection error
- -5: Polarity selection error
- -6: The minimal converting time is 10 us !
- -7: Not available conversion time unit
- -8: Delay mode selection not available
- -9: Delay time unit selection not available
- -10: Delay value not available
- -11: Trigger mode : 2 different mode cannot be simultaneously be activated
- -12: Hardware trigger : edge definition error
- -13: Hardware trigger count value not available
- -14: Nbr of sequence to acquire by trigger mode not available
- -15: Data format not available
- -100: Channel initialisation kernel function error
- -101: Conversion time initialisation kernel function error
- -102: Delay initialisation kernel function error
- -103: hardware trigger initialisation/enable kernel function error
- -104: hardware trigger disable kernel function error
- -105: synchro trigger initialisation/enable kernel function error
- -106: synchro trigger disable kernel function error
- -107 Sequence trigger count initialisation error
- -108: Sequence initialisation kernel function error
- -109: Start sequence kernel function error

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

## Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

### 2.20.2.2 int MSXE301x\_\_AnalogInputStopAndReleaseSequence ( void \* \_\_, struct MSXE301x\_\_Response \* Response )

#### Parameters

[in] \_\_ : no input parameter

[out] Response :

*iReturnValue* :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: Stop sequence kernel function error
- -101: Release sequence kernel function error

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

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

## 2.21 MSX-E301x calibration functions

The calibration functions permit to calibrate the analog input of the module.

### Data Structures

- struct [MSXE301x\\_\\_CalibrationGetCurrentStatusResponse](#)

### Functions

- int [MSXE301x\\_\\_CalibrationStart](#) (xsd\_\_unsignedLong ulCalibrationMode, xsd\_\_unsignedLong ulGain, xsd\_\_double dCalibrationValue, struct [MSXE301x\\_\\_Response](#) \*Response)

*This function start the calibration thread.*

- int [MSXE301x\\_\\_CalibrationGetCurrentStatus](#) (void \_\_, struct [MSXE301x\\_\\_CalibrationGetCurrentStatusResponse](#) \*Response)

*This function return the current calibration status.*

- int [MSXE301x\\_\\_CalibrationNextStep](#) (void \_\_, struct [MSXE301x\\_\\_Response](#) \*Response)

*This function start the next calibration step.*

- int [MSXE301x\\_\\_CalibrationBreak](#) (void \_\_, struct [MSXE301x\\_\\_Response](#) \*Response)

*This function break the current calibration.*

### 2.21.1 Detailed Description

Please use the following algorithm :

```

MSXE301x__CalibrationStart(...)

do
    MSXE301x__CalibrationGetCurrentStatus
while (Status == 0)

MSXE301x__CalibrationNextStep(...)

do
    MSXE301x__CalibrationGetCurrentStatus
while (Status == 1)

MSXE301x__CalibrationNextStep(...)

Use MSXE301x__CalibrationBreak(...) to break the calibration sequence

```

## 2.21.2 Function Documentation

**2.21.2.1** `int MSXE301x__CalibrationStart ( xsd__unsignedLong ulCalibrationMode, xsd__unsignedLong ulGain, xsd__double dCalibrationValue, struct MSXE301x__Response * Response )`

### Parameters

[in] *ulCalibrationMode* : Calibration mode selection:

- 0 Hex : All channels calibration via a external calibration source (voltage/current). Not available for mixing input types.
- 1 Hex : All channels calibration via the internal voltage calibration source. Not available for current or mixing input types.
- 2 Hex : Channel group 0 (Channel 0 to 3) calibration via a external calibration source (voltage/current).
- 102 Hex : Channel group 1 (Channel 4 to 7) calibration via a external calibration source (voltage/current).
- 202 Hex : Channel group 2 (Channel 8 to 11) calibration via a external calibration source (voltage/current).
- 302 Hex : Channel group 3 (Channel 12 to 15) calibration via a external calibration source (voltage/current).
- 3 Hex : Channel group 0 (Channel 0 to 3) calibration via the internal voltage calibration source. Not available for current inputs.
- 103 Hex : Channel group 1 (Channel 4 to 7) calibration via the internal voltage calibration source. Not available for current inputs.
- 203 Hex : Channel group 2 (Channel 8 to 11) calibration via the internal voltage calibration source. Not available for current inputs.
- 303 Hex : Channel group 3 (Channel 12 to 15) calibration via the internal voltage calibration source. Not available for current inputs.

[in] *ulGain* : Gain selection 1 or 2

[in] *dCalibrationValue* : Calibration value

[out] *Response* :

*iReturnValue* : Error value :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -2: Calibration mode selection error
- -3: Calibration gain selection error

- -4: Calibration value selection error
- -100: Start calibration kernel function error

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

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

#### 2.21.2.2 int MSXE301x\_\_CalibrationGetCurrentStatus ( void \* \_\_, struct MSXE301x\_\_CalibrationGetCurrentStatusResponse \* Response )

##### Parameters

[in] \_\_ : no input parameter

[out] **Response** :

**iReturnValue** : Error value :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: Get status calibration kernel function error
- -101: Save calibration read source error
- -102: Save calibration write destination error

**ulStatus** : TODO

**ulDigitalValue** : Last measured digital value

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

#### 2.21.2.3 int MSXE301x\_\_CalibrationNextStep ( void \* \_\_, struct MSXE301x\_\_Response \* Response )

##### Parameters

[in] \_\_ : no input parameter

[out] **Response** :

**iReturnValue** : Error value :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: Calibration next step kernel function error

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

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

**2.21.2.4** `int MSXE301x__CalibrationBreak ( void * _, struct MSXE301x__Response * Response )`

#### Parameters

[in] `_` : no input parameter

[out] ***Response*** :

***iReturnValue*** : Error value :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: Calibration break kernel function error

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

#### 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 MSXE301x\_\_AnalogInputGetAutoRefreshValuesResponse Struct Reference

### Data Fields

- struct [DefaultResponse](#) sResponse

*Default return values.*

- [xsd\\_\\_unsignedLong](#) ulTimeStampLow

*The meaning of this field is defined in the related header of the function who use this type.*

- [xsd\\_\\_unsignedLong](#) ulTimeStampHigh

*The meaning of this field is defined in the related header of the function who use this type.*

- [xsd\\_\\_unsignedLong](#) ulCounterValue

*The meaning of this field is defined in the related header of the function who use this type.*

- [xsd\\_\\_unsignedLong](#) ulValue [16]

*The meaning of this field is defined in the related header of the function who use this type.*

### 3.3.1 Field Documentation

- 3.3.1.1 struct `DefaultResponse` `MSXE301x__AnalogInputGetAutoRefreshValuesResponse::sResponse`
- 3.3.1.2 `xsd__unsignedLong` `MSXE301x__AnalogInputGetAutoRefreshValuesResponse::ulTimeStampLow`
- 3.3.1.3 `xsd__unsignedLong` `MSXE301x__AnalogInputGetAutoRefreshValuesResponse::ulTimeStampHigh`
- 3.3.1.4 `xsd__unsignedLong` `MSXE301x__AnalogInputGetAutoRefreshValuesResponse::ulCounterValue`
- 3.3.1.5 `xsd__unsignedLong` `MSXE301x__AnalogInputGetAutoRefreshValuesResponse::ulValue[16]`

## 3.4 MSXE301x\_\_AnalogInputGetChannelTypeResponse Struct Reference

### Data Fields

- struct `DefaultResponse` `sResponse`  
*Default return values.*
- `xsd__unsignedLong` `ulType` [16]  
*The meaning of this field is defined in the related header of the function who use this type.*

### 3.4.1 Field Documentation

- 3.4.1.1 struct `DefaultResponse` `MSXE301x__AnalogInputGetChannelTypeResponse::sResponse`
- 3.4.1.2 `xsd__unsignedLong` `MSXE301x__AnalogInputGetChannelTypeResponse::ulType[16]`

## 3.5 MSXE301x\_\_AnalogMeasureGetConfigurationResponse Struct Reference

### Data Fields

- `xsd__int` `iReturnValue`
- `xsd__unsignedLong` `ulRunningMode`
- struct {
  - `xsd__unsignedLong` `ulConvertTimeUnit`  
*0: micro second 1: milli second 2: second*
  - `xsd__unsignedLong` `ulConvertTime`  
*The conversion time.*
  - `xsd__unsignedLong` `ulChannelInitialization`  
*Channel initialization ?*

} sAcquisition

0: Disabled 1: Auto refresh 2: Sequence 2: Calibtation

- struct {
    - xsd\_\_unsignedLong ulChannelMask  
Mask of channels used for the auto refresh mode.
    - xsd\_\_unsignedLong ulAverageMode  
Determine average mode 0: Sequence 1 : Channel.
    - xsd\_\_unsignedLong ulAverageValue  
Determine the number of sequence for a average value calculation.
    - xsd\_\_unsignedLong ulDataFormat  
D0: Time stamp information.
- } sAutoRefresh

- struct {
    - xsd\_\_unsignedLong ulSequenceSize  
Sequence array size.
    - xsd\_\_unsignedLong ulSequence [64]  
Sequence channel array.
    - xsd\_\_unsignedLong ulSequenceInterrupt  
Determine the number of sequences for the interrupt.
    - xsd\_\_unsignedLong ulNbrOfSequence  
Determine the number total of sequences.
    - xsd\_\_unsignedLong ulDelayFlag  
0 : Disabled 1 : Enabled
    - xsd\_\_unsignedLong ulDelayMode  
Delay mode.
    - xsd\_\_unsignedLong ulDelayTimeUnit  
Delay time unit 1: micros 2 : ms 3 : s.
    - xsd\_\_unsignedLong ulDelayTime  
Delay time.
    - xsd\_\_unsignedLong ulDataFormat  
D0: Time stamp information.
- } sSequence

- struct {
    - xsd\_\_unsignedLong ulSequenceTriggerCount  
Number of sequence to trigger.
    - struct {
      - xsd\_\_unsignedLong ulFlag  
0 : Disabled 1: Enabled
      - xsd\_\_unsignedLong ulLevel  
0: Low level 1: High level 2 : Low/High level
      - xsd\_\_unsignedLong ulAction  
0: One shot 1: Sequence trigger
      - xsd\_\_unsignedLong ulCount  
Number of trigger vor the action.
- } sHardware
- struct {
    - xsd\_\_unsignedLong ulFlag  
0 : Disabled 1: Enabled
    - xsd\_\_unsignedLong ulAction

```

    0: One shot 1: Sequence trigger
} sSoftware
struct {
    xsd__unsignedLong ulFlag
    0 : Disabled 1: Enabled
    xsd__unsignedLong ulAction
    0: One shot 1: Sequence trigger
} sSynchro
} sTrigger

```

### 3.5.1 Field Documentation

- 3.5.1.1 `xsd__int MSXE301x__AnalogMeasureGetConfigurationResponse::iReturnValue`
- 3.5.1.2 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulRunningMode`
- 3.5.1.3 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulConvertTimeUnit`
- 3.5.1.4 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulConvertTime`
- 3.5.1.5 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulChannelInitialization`
- 3.5.1.6 `struct { ... } MSXE301x__AnalogMeasureGetConfigurationResponse::sAcquisition`
- 3.5.1.7 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulChannelMask`
- 3.5.1.8 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulAverageMode`
- 3.5.1.9 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulAverageValue`
- 3.5.1.10 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulDataFormat`

0 without, 1 with. D1 : Data format 0: digital 1: analog (in V)

0 without, 1 with. D1 : Sequence counter information. 0 without, 1 with. D2 : Data format 0: digital 1: analog (in V)

**3.5.1.11**   **struct { ... } MSXE301x\_\_AnalogMeasureGetConfigurationResponse::sAutoRefresh**

**3.5.1.12**   **xsd\_\_unsignedLong MSXE301x\_\_-**  
**AnalogMeasureGetConfigurationResponse::ulSequenceSize**

**3.5.1.13**   **xsd\_\_unsignedLong MSXE301x\_\_-**  
**AnalogMeasureGetConfigurationResponse::ulSequence[64]**

**3.5.1.14**   **xsd\_\_unsignedLong MSXE301x\_\_-**  
**AnalogMeasureGetConfigurationResponse::ulSequenceInterrupt**

**3.5.1.15**   **xsd\_\_unsignedLong MSXE301x\_\_-**  
**AnalogMeasureGetConfigurationResponse::ulNbrOfSequence**

0: Continuous

- 3.5.1.16 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulDelayFlag`
- 3.5.1.17 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulDelayMode`
- 3.5.1.18 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulDelayTimeUnit`
- 3.5.1.19 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulDelayTime`
- 3.5.1.20 `struct { ... } MSXE301x__AnalogMeasureGetConfigurationResponse::sSequence`
- 3.5.1.21 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulSequenceTriggerCount`
- 3.5.1.22 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulFlag`
- 3.5.1.23 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulLevel`
- 3.5.1.24 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulAction`
- 3.5.1.25 `xsd__unsignedLong MSXE301x__AnalogMeasureGetConfigurationResponse::ulCount`
- 3.5.1.26 `struct { ... } MSXE301x__AnalogMeasureGetConfigurationResponse::sHardware`
- 3.5.1.27 `struct { ... } MSXE301x__AnalogMeasureGetConfigurationResponse::sSoftware`
- 3.5.1.28 `struct { ... } MSXE301x__AnalogMeasureGetConfigurationResponse::sSynchro`
- 3.5.1.29 `struct { ... } MSXE301x__AnalogMeasureGetConfigurationResponse::sTrigger`

## 3.6 MSXE301x\_\_CalibrationGetCurrentStatusResponse Struct Reference

### Data Fields

- `struct DefaultResponse sResponse`  
*Default return values.*
- `xsd__unsignedLong ulStatus`  
*Calibration status :*
- `xsd__double dCurrentValue`  
*Last measured value.*
- `xsd__unsignedLong ulError`  
*Calibration error :*

### 3.6.1 Field Documentation

#### 3.6.1.1 struct DefaultResponse MSXE301x\_\_CalibrationGetCurrentStatusResponse::sResponse

#### 3.6.1.2 xsd\_\_unsignedLong MSXE301x\_\_CalibrationGetCurrentStatusResponse::ulStatus

- 0 Hex: No calibration in progress
- 1 Hex: Wait that the user inject 0V to channel 0
- 101 Hex: Wait that the user inject 0V to channel 4
- 201 Hex: Wait that the user inject 0V to channel 8
- 301 Hex: Wait that the user inject 0V to channel 12
- 2 Hex: Offset group 0 (channel 0 to 3) calibration in progress
- 102 Hex: Offset group 1 (channel 4 to 7) calibration in progress
- 202 Hex: Offset group 2 (channel 8 to 11) calibration in progress
- 302 Hex: Offset group 3 (channel 12 to 15) calibration in progress
- 3 Hex: Wait that the user inject the selected calibration value to channel 0
- 103 Hex: Wait that the user inject the selected calibration value to channel 4
- 203 Hex: Wait that the user inject the selected calibration value to channel 8
- 303 Hex: Wait that the user inject the selected calibration value to channel 12
- 4 Hex: Selected group 0 (channel 0 to 3) calibration value in progress
- 104 Hex: Selected group 1 (channel 4 to 7) calibration value in progress
- 204 Hex: Selected group 2 (channel 7 to 11) calibration value in progress
- 304 Hex: Selected group 3 (channel 12 to 15) calibration value in progress
- 6 Hex: Wait that the user inject the selected inverted calibration value to channel 0
- 106 Hex: Wait that the user inject the selected inverted calibration value to channel 4
- 206 Hex: Wait that the user inject the selected inverted calibration value to channel 8
- 306 Hex: Wait that the user inject the selected inverted calibration value to channel 12
- 5 Hex: Calibration finished

#### 3.6.1.3 xsd\_\_double MSXE301x\_\_CalibrationGetCurrentStatusResponse::dCurrentValue

#### 3.6.1.4 xsd\_\_unsignedLong MSXE301x\_\_CalibrationGetCurrentStatusResponse::ulError

- 0 : No error
- 1 : User break occur
- 2 : Min set 0 offset potentiometer value reached
- 3 : Max set 0 offset potentiometer value reached



- 4 : Min set 1 offset potentiometer value reached
- 5 : Max set 1 offset potentiometer value reached
- 6 : Min set 2 offset potentiometer value reached
- 7 : Max set 2 offset potentiometer value reached
- 8 : Min set 3 offset potentiometer value reached
- 9 : Max set 3 offset potentiometer value reached
- 10 : Min set 0 gain potentiometer value reached
- 11 : Max set 0 gain potentiometer value reached
- 12 : Min set 1 gain potentiometer value reached
- 13 : Max set 1 gain potentiometer value reached
- 14 : Min set 2 gain potentiometer value reached
- 15 : Max set 2 gain potentiometer value reached
- 16 : Min set 3 gain potentiometer value reached
- 17 : Max set 3 gain potentiometer value reached

## 3.7 MSXE301x\_\_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).*

### 3.7.1 Field Documentation

#### 3.7.1.1 [xsd\\_\\_int MSXE301x\\_\\_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

3.7.1.2 `xsd__int MSXE301x__Response::syserrno`

## 3.8 MSXE301x\_\_unsignedLong16FixedArrayParam Struct Reference

### Data Fields

- `xsd__unsignedLong ulValue` [16]

*The meaning of this field is defined in the related header of the function who use this type.*

### 3.8.1 Field Documentation

3.8.1.1 `xsd__unsignedLong MSXE301x__unsignedLong16FixedArrayParam::ulValue`[16]

## 3.9 MSXE301x\_\_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.*

### 3.9.1 Field Documentation

3.9.1.1 `struct DefaultResponse MSXE301x__unsignedLongResponse::sResponse`

3.9.1.2 `xsd__unsignedLong MSXE301x__unsignedLongResponse::ulValue`

## 3.10 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.*

### 3.10.1 Field Documentation

3.10.1.1 struct DefaultResponse MXCommon\_\_ByteArrayResponse::sResponse

3.10.1.2 struct ByteArray MXCommon\_\_ByteArrayResponse::sArray

## 3.11 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.*

### 3.11.1 Field Documentation

3.11.1.1 struct DefaultResponse MXCommon\_\_FileResponse::sResponse

3.11.1.2 struct ByteArray MXCommon\_\_FileResponse::sArray

3.11.1.3 [xsd\\_\\_unsignedLong](#) MXCommon\_\_FileResponse::ulEOF

## 3.12 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.*

### 3.12.1 Field Documentation

3.12.1.1 struct `DefaultResponse` `MXCommon__GetAutoConfigurationFileResponse::sResponse`

3.12.1.2 struct `ByteArray` `MXCommon__GetAutoConfigurationFileResponse::bArray`

3.12.1.3 `xsd__unsignedLong` `MXCommon__GetAutoConfigurationFileResponse::ulEOF`

## 3.13 `MXCommon__GetEthernetLinksStatesResponse` Struct Reference

### Data Fields

- struct `DefaultResponse` `sResponse`  
*Default return values.*
- struct `sGetEthernetLinksStatesPort` `sPort0`
- struct `sGetEthernetLinksStatesPort` `sPort1`

### 3.13.1 Field Documentation

3.13.1.1 struct `DefaultResponse` `MXCommon__GetEthernetLinksStatesResponse::sResponse`

3.13.1.2 struct `sGetEthernetLinksStatesPort` `MXCommon__GetEthernetLinksStatesResponse::sPort0`

3.13.1.3 struct `sGetEthernetLinksStatesPort` `MXCommon__GetEthernetLinksStatesResponse::sPort1`

## 3.14 `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.*

### 3.14.1 Field Documentation

- 3.14.1.1 struct `DefaultResponse` `MXCommon__GetHardwareTriggerFilterTimeResponse::sResponse`
- 3.14.1.2 `xsd__unsignedLong` `MXCommon__GetHardwareTriggerFilterTimeResponse::ulFilterTime`
- 3.14.1.3 `xsd__unsignedLong` `MXCommon__GetHardwareTriggerFilterTimeResponse::ulInfo01`
- 3.14.1.4 `xsd__unsignedLong` `MXCommon__GetHardwareTriggerFilterTimeResponse::ulInfo02`

## 3.15 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.*

### 3.15.1 Field Documentation

- 3.15.1.1 struct `DefaultResponse` `MXCommon__GetHardwareTriggerStateResponse::sResponse`
- 3.15.1.2 `xsd__unsignedLong` `MXCommon__GetHardwareTriggerStateResponse::ulState`
- 3.15.1.3 `xsd__unsignedLong` `MXCommon__GetHardwareTriggerStateResponse::ulInfo01`
- 3.15.1.4 `xsd__unsignedLong` `MXCommon__GetHardwareTriggerStateResponse::ulInfo02`

## 3.16 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.*

### 3.16.1 Field Documentation

**3.16.1.1** `struct DefaultResponse MXCommon__ -  
GetModuleTemperatureValueAndStatusResponse::sResponse`

**3.16.1.2** `xsd__double MXCommon__ -  
GetModuleTemperatureValueAndStatusResponse::dTemperatureValue`

**3.16.1.3** `xsd__unsignedLong MXCommon__ -  
GetModuleTemperatureValueAndStatusResponse::ulTemperatureStatus`

**3.16.1.4** `xsd__unsignedLong MXCommon__ -  
GetModuleTemperatureValueAndStatusResponse::ulInfo`

## 3.17 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).*

### 3.17.1 Field Documentation

**3.17.1.1** `struct DefaultResponse MXCommon__GetTimeResponse::sResponse`

**3.17.1.2** `xsd__unsignedLong MXCommon__GetTimeResponse::ulLowTime`

**3.17.1.3** `xsd__unsignedLong MXCommon__GetTimeResponse::ulHighTime`

## 3.18 MXCommon\_\_GetUpTimeResponse Struct Reference

### Data Fields

- `struct DefaultResponse sResponse`

*Default return value.*

- [xsd\\_\\_unsignedLong ulUpTime](#)

*Reserved.*

### 3.18.1 Field Documentation

3.18.1.1 [struct DefaultResponse MXCommon\\_\\_GetUpTimeResponse::sResponse](#)

3.18.1.2 [xsd\\_\\_unsignedLong MXCommon\\_\\_GetUpTimeResponse::ulUpTime](#)

## 3.19 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\(\)](#)).*

### 3.19.1 Field Documentation

3.19.1.1 [xsd\\_\\_int MXCommon\\_\\_Response::iReturnValue](#)

3.19.1.2 [xsd\\_\\_int MXCommon\\_\\_Response::syserrno](#)

## 3.20 MXCommon\_\_TestCustomerIDResponse Struct Reference

### Data Fields

- [struct DefaultResponse sResponse](#)

*Default return values.*

- [struct ByteArray bValueArray](#)

*non encrypted value*

- [struct ByteArray bCryptedValueArray](#)

*encrypted value*

### 3.20.1 Field Documentation

3.20.1.1 struct `DefaultResponse` `MXCommon__TestCustomerIDResponse::sResponse`

3.20.1.2 struct `ByteArray` `MXCommon__TestCustomerIDResponse::bValueArray`

3.20.1.3 struct `ByteArray` `MXCommon__TestCustomerIDResponse::bCryptedValueArray`

## 3.21 `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.*

### 3.21.1 Field Documentation

3.21.1.1 struct `DefaultResponse` `MXCommon__unsignedLongResponse::sResponse`

3.21.1.2 `xsd__unsignedLong` `MXCommon__unsignedLongResponse::ulValue`

## 3.22 `sGetEthernetLinksStatesPort` Struct Reference

### Data Fields

- `xsd__unsignedLong` `ulState`
- `xsd__unsignedLong` `ulSpeed`
- `xsd__unsignedLong` `ulDuplex`
- `xsd__unsignedLong` `ulInfo1`
- `xsd__unsignedLong` `ulInfo2`



### 3.22.1 Field Documentation

3.22.1.1 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulState`

3.22.1.2 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulSpeed`

3.22.1.3 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulDuplex`

3.22.1.4 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo1`

3.22.1.5 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo2`

## 3.23 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*

### 3.23.1 Field Documentation

3.23.1.1 `xsd__unsignedLong* UnsignedLongArray::__ptr`

3.23.1.2 `int UnsignedLongArray::__size`

3.23.1.3 `int UnsignedLongArray::__offset`

## 3.24 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*

### 3.24.1 Field Documentation

3.24.1.1 `xsd__unsignedShort* UnsignedShortArray::__ptr`

3.24.1.2 `int UnsignedShortArray::__size`

3.24.1.3 `int UnsignedShortArray::__offset`

## 3.25 `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*

### 3.25.1 Field Documentation

3.25.1.1 `unsigned char* xsd__base64Binary::__ptr`

3.25.1.2 `int xsd__base64Binary::__size`

# Chapter 4

## File Documentation

### 4.1 MSXE301x\_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 [MSXE301x\\_\\_Response](#)
- struct [MSXE301x\\_\\_unsignedLongResponse](#)
- struct [MSXE301x\\_\\_unsignedLong16FixedArrayParam](#)
- struct [MSXE301x\\_\\_AnalogMeasureGetConfigurationResponse](#)
- struct [MSXE301x\\_\\_AnalogInputGetChannelTypeResponse](#)
- struct [MSXE301x\\_\\_AnalogInputGetAutoRefreshValuesResponse](#)
- struct [MSXE301x\\_\\_CalibrationGetCurrentStatusResponse](#)

## 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 [MXCommon\\_\\_SetFilterChannels](#) (struct [xsd\\_\\_base64Binary](#) \*ChannelList, struct [MXCommon\\_\\_Response](#) \*Response)

*This function sets or resets a filter to a channel.*

- `int MSXE301x__AnalogInputGetChannelType (xsd__unsignedLong ulOption1, struct MSXE301x__AnalogInputGetChannelTypeResponse *Response)`

*Return the type of the analog input channels.*

- `int MSXE301x__AnalogMeasureGetConfiguration (xsd__unsignedLong ulModule, struct MSXE301x__AnalogMeasureGetConfigurationResponse *Response)`

*Return the running configuration.*

- `int MSXE301x__AnalogInputInitAndStartAutoRefresh (xsd__unsignedLong ulChannelMask, struct MSXE301x__unsignedLong16FixedArrayParam *pulGainArray, struct MSXE301x__unsignedLong16FixedArrayParam *pulPolarityArray, xsd__unsignedLong ulAverageMode, xsd__unsignedLong ulAverageValue, xsd__unsignedLong ulConversionTime, xsd__unsignedLong ulConversionTimeUnit, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, struct MSXE301x__Response *Response)`

*Starts an autorefresh acquisition using provided configuration.*

- `int MSXE301x__AnalogInputGetAutoRefreshValues (void *_ , struct MSXE301x__AnalogInputGetAutoRefreshValuesResponse *Response)`

*Returns the values acquired in auto refresh mode.*

- `int MSXE301x__AnalogInputStopAndReleaseAutoRefresh (void *_ , struct MSXE301x__Response *Response)`

*Stops the current autorefresh acquisition.*

- `int MSXE301x__AnalogInputInitAndStartSequence (xsd__unsignedLong ulNbrOfChannel, struct MSXE301x__unsignedLong16FixedArrayParam *pulChannelList, struct MSXE301x__unsignedLong16FixedArrayParam *pulGainArray, struct MSXE301x__unsignedLong16FixedArrayParam *pulPolarityArray, xsd__unsignedLong ulConversionTime, xsd__unsignedLong ulConversionTimeUnit, xsd__unsignedLong ulNbrOfSequence, xsd__unsignedLong ulNbrMaxSequenceToTransfer, xsd__unsignedLong ulDelayMode, xsd__unsignedLong ulDelayTimeUnit, xsd__unsignedLong ulDelayValue, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, struct MSXE301x__Response *Response)`

*Initialise and start the analog input sequence acquisition mode.*

- `int MSXE301x__AnalogInputStopAndReleaseSequence (void *_ , struct MSXE301x__Response *Response)`

*Stop and release the analog input sequence acquisition mode.*

- `int MSXE301x__CalibrationStart (xsd__unsignedLong ulCalibrationMode, xsd__unsignedLong ulGain, xsd__double dCalibrationValue, struct MSXE301x__Response *Response)`

*This function start the calibration thread.*

- `int MSXE301x__CalibrationGetCurrentStatus (void *_ , struct MSXE301x__CalibrationGetCurrentStatusResponse *Response)`



*This function return the current calibration status.*

- int [MSXE301x\\_\\_CalibrationNextStep](#) (void \*\_ , struct [MSXE301x\\_\\_Response](#) \*Response)  
*This function start the next calibration step.*
- int [MSXE301x\\_\\_CalibrationBreak](#) (void \*\_ , struct [MSXE301x\\_\\_Response](#) \*Response)  
*This function break the current calibration.*

### 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\\_\\_Strerror\(\)](#).
  - dValue : Temperature value in Degree Celsius
- ulTemperatureStatus : Temperature Status :
  - TEMPERATURE\_INITIAL = 0 : Temperature not ready
  - TEMPERATURE\_TOOLOW = 1 : Temperature too low !
  - TEMPERATURE\_LOW = 2 : Temperature under the min warning value
  - TEMPERATURE\_NOMINAL = 3 : Temperature in the nominal range
  - TEMPERATURE\_HIGH = 4 : Temperature over the max warning value
  - TEMPERATURE\_TOOHIGH = 5 : Temperature too high !

- ulInfo : Reserved

**Return values**

**SOAP\_OK** SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

- [in] *dMinimalWarningLevel* : Minimal temperature warning level in Degree : 5 to 60 Degree Celsius
- [in] *dMaximalWarningLevel* : Maximal temperature warning level in Degree : 5 to 60 Degree Celsius
- [in] *ulOption* : Reserved
- [out] *Response*
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

#### Return values

- SOAP\_OK* SOAP call success
- otherwise* SOAP protocol error

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

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

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

#### Parameters

- [in] *ulFilterTime* Filter time for the hardware trigger input in steps of 250ns (max value : 65535 ).
  - **0**: Disable the filter
  - **1**: Sets the filter time to 250 ns
  - **2**: Sets the filter time to 500 ns
  - ...
  - **65535**: Sets the filter time to 16 ms
- [in] *ulOption* Reserved. Set to 0
- [out] *Response* Response of the system
  - *sResponse.iReturnValue*
    - **0**: The remote function performed OK
    - **-1**: Internal system error occurred. See value of syserrno
  - *sResponse.syserrno* system error code (the value of the libc "errno" code)

#### Return values

- 0** SOAP\_OK
- Others* See SOAP error

#### 4.1.2.13 `int MXCommon__GetHardwareTriggerFilterTime ( xsd__unsignedLong ulOption, struct MXCommon__GetHardwareTriggerFilterTimeResponse * Response )`

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

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

##### Parameters

[in] *ulOption* Reserved. Set to 0

[out] *Response* Response of the system

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

##### Return values

0 SOAP\_OK

*Others* See SOAP error

#### 4.1.2.14 `int MXCommon__GetHardwareTriggerState ( xsd__unsignedLong ulOption, struct MXCommon__GetHardwareTriggerStateResponse * Response )`

##### Parameters

[in] *ulOption* : Reserved

[out] *Response* • *ulState* : Hardware trigger input state.

- 0: Hardware trigger input is low
- 1: Hardware trigger input is high.
- *sResponse.iReturnValue* : Return value
  - 0 : success
  - -1: system error (see syserrno)
- *sResponse.syserrno* : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_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\\_\\_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 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\\_\\_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.

#### 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 MXCommon\_SetFilterChannels ( struct xsd\_\_base64Binary \* ChannelList, struct MXCommon\_Response \* Response )

##### Parameters

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

[out] **Response**

- `sResponse.iReturnValue` : Return value
  - 0 : success
  - -1: system error (see `syserrno`)
- `sResponse.syserrno` : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

#### Return values

**SOAP\_OK** SOAP call success

*otherwise* SOAP protocol error

#### 4.1.2.39 int MSXE301x\_AnalogInputGetChannelType ( xsd\_\_unsignedLong ulOption1, struct MSXE301x\_AnalogInputGetChannelTypeResponse \* Response )

##### Parameters

[in] **ulOption1** : Reserved

[out] **Response** :

*iReturnValue* :

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

*syserrno* : system-error code (the value of the libc "errno" code) *ulType* : Array that contain the channels type (0 : voltage, 1 : current)

- `ulType [0]` : Channel 0 type
- ...

- `ulType [15]` : Channel 15 type

### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

**4.1.2.40** `int MSXE301x__AnalogMeasureGetConfiguration ( xsd__unsignedLong ulModule,  
struct MSXE301x__AnalogMeasureGetConfigurationResponse * Response )`

### Parameters

[in] *ulOption1* : Reserved

[out] *Response* :

*iReturnValue* :

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

*sAcquisition* :

- *ulConvertTimeUnit* 0: micro second 1: milli second 2: second
- *pul\_ConvertTimeValue* : (10,65535)
- *pul\_ChannelInitialisation* (Are channels initialised and how) pointer to unsigned long[16].  
bit 31 = initialised : (0,1) bit 30= polarity : (0,1) - 0:bipolar, 1:unipolar bits 29-0 : gain :  
(1,2) - 0: 10 V , 1: 5V

*sAutoRefresh* :

- *ulChannelMask* Mask of channels used for the auto refresh mode
- *ulAverageMode* Determine average mode 0: Sequence 1 : Channel
- *ulAverageValue* Determine the number of sequence for a average value calculation
- *ulDataFormat* D0: Time stamp information. 0 without, 1 with. D1 : Data format 0: digital  
1: analog (in V)

*sSequence* :

- *ulSequenceSize* Sequence array size
- *ulSequence* Sequence channel array
- *ulSequenceInterrupt* Determine the number of sequences for the interrupt
- *ulNbrOfSequence* Determine the number total of sequences. 0: Continuous
- *ulDelayFlag* 0 : Disabled 1 : Enabled
- *ulDelayMode* 0 : delay not used 1 : mode 1 2 : mode 2
- *ulDelayTimeUnit* Delay time unit 1: micros 2 : ms 3 : s
- *ulDelayTime* Delay time
- *ulDataFormat* D0: Time stamp information. 0 without, 1 with.  
D1 : Sequence counter information. 0 without, 1 with.  
D2 : Data format 0: digital 1: analog (in V)

### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

**4.1.2.41** `int MSXE301x__AnalogInputInitAndStartAutoRefresh ( xsd__unsignedLong ulChannelMask, struct MSXE301x__unsignedLong16FixedArrayParam * pulGainArray, struct MSXE301x__unsignedLong16FixedArrayParam * pulPolarityArray, xsd__unsignedLong ulAverageMode, xsd__unsignedLong ulAverageValue, xsd__unsignedLong ulConversionTime, xsd__unsignedLong ulConversionTimeUnit, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, struct MSXE301x__Response * Response )`

#### Parameters

- [in] **ulChannelMask** Mask of the channel to acquire by the auto refresh (1 bit = 1 Channel)
- [in] **pulGainArray** Define the gain (1 or 2) to use for each channel.  
Information : by the channel type "current", the gain 2 is recommended.
- 1 : +10 V or max 20mA when Unipolar, +/-10V or +/- max 20mA when Bipolar
  - 2 : +5 V or 20mA when Unipolar, +/-5V or +/- 20mA when Bipolar Each index of the array correspond to the channel :
- sample :
- [0] : Define the gain for the channel 0
  - [1] : Define the gain for the channel 1
  - ...
- Remark: When using a current version of the MSX-E301x, gain 2 has to be used.
- [in] **pulPolarityArray** Define the polarity (0:Unipolar or 1:Bipolar) to use for each channel Each index of the array correspond to the channel :
- sample :
- [0] : Define the polarity for the channel 0
  - [1] : Define the polarity for the channel 1
  - ...
- Remark: When using the unipolar mode, the raw value for 0 volt (or 0 ampere) will be 32767.
- [in] **ulAverageMode** Set the average mode :
- 0 : not used
  - 1 : average per Sequence
  - 2 : average per channel
- [in] **ulAverageValue** Set the average value (only used, when average is used) :
- 0 : not used
  - max value : 255
- [in] **ulConversionTime** Conversion Time
- range from min 10 to 65535 when the unit is the microsecond
  - range from min 1 to 65535 when the unit is the millisecond
  - range from min 1 to 65535 when the unit is the second
- [in] **ulConversionTimeUnit** Conversion Time Unit
- 0 : microsecond
  - 1 : millisecond
  - 2 : second

- [in] ***ulTriggerMask*** Define the source of the trigger
- 0 : trigger disabled
  - 1 : Enable Hardware Digital Input Trigger
  - 2 : Enable Synchro Trigger
- [in] ***ulTriggerMode*** Define the trigger mode
- 1 : One shot trigger
  - 2 : Sequence trigger
- [in] ***ulHardwareTriggerEdge*** 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] ***ulHardwareTriggerCount*** Define the number of trigger events before the action occur
- 0 or 1 : all trigger event start the action
  - max value : 65535
- [in] ***ulByTriggerNbrOfSeqToAcquire*** Define the number of sequence to acquire by each trigger event
- 0 : continuous mode
  - <> 0 : number of sequence : (1..0xFFFFFFFF)
- [in] ***ulDataFormat*** D0 : Time stamp information
- 0: no time stamp information
  - 1: time stamp information
- D1 : Data format
- 0: Digital value
  - 1: Analog value (in V)
- [in] ***ulOption1*** Reserved
- [in] ***ulOption2*** Reserved
- [in] ***ulOption3*** Reserved
- [out] ***Response*** :
- iReturnValue*** :
- 0 : means the remote function performed OK
  - -1: means an system error occurred
  - -2: The channel mask cannot be null
  - -3: Channel Mask error
  - -4: Gain selection error
  - -5: Polarity selection error
  - -6: not available average mode
  - -7: not available average value
  - -8: The minimal converting time is 10 us !
  - -9: Not available conversion time unit
  - -10: Trigger mode : 2 different mode cannot be simultaneously be activated
  - -11: Hardware trigger : front definition error
  - -12: Hardware trigger count value not available
  - -13: Nbr of sequence to acquire by trigger mode not available
  - -14: Data format not available

- -100: Channel initialisation kernel function error
- -101: Conversion time initialisation kernel function error
- -102: Average mode initialisation kernel function error
- -103: hardware trigger initialisation/enable kernel function error
- -104: hardware trigger disable kernel function error
- -105: synchro trigger initialisation/enable kernel function error
- -106: synchro trigger disable kernel function error
- -107 Sequence trigger count initialisation error
- -108: Start auto refresh kernel function error

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

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

#### 4.1.2.42 int MSXE301x\_\_AnalogInputGetAutoRefreshValues ( void \* \_\_, struct MSXE301x\_\_AnalogInputGetAutoRefreshValuesResponse \* *Response* )

#### Parameters

[in] *\_\_* Dummy parameter

[out] *Response* :

**iReturnValue** :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: GetAutoRefreshAllValues kernel function error

**ulTimeStampLow** : number of microseconds since the begin of the second

**ulTimeStampHigh** : number of seconds since the Epoch

**ulCounterValue** : Array that contain the counter values

**ulValue** : Array that contain the channels values

- ulValues [0] : Channel 0 value
- ...
- ulValues [15] : Channel 15 value

Remark: When using the unipolar mode, the raw value for 0 volt (or 0 ampere) will be 32767.

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

#### 4.1.2.43 int MSXE301x\_\_AnalogInputStopAndReleaseAutoRefresh ( void \* \_\_, struct MSXE301x\_\_Response \* *Response* )

##### Parameters

[in] *\_\_* Dummy parameter

[out] *Response* :

*iReturnValue* :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: StopAutoRefresh kernel function error

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

##### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

Must be called before any another call to MSXE301x\_\_AnalogInputInitAndStartAutoRefresh.

#### 4.1.2.44 int MSXE301x\_\_AnalogInputInitAndStartSequence ( xsd\_\_unsignedLong *ulNbrOfChannel*, struct MSXE301x\_\_unsignedLong16FixedArrayParam \* *pulChannelList*, struct MSXE301x\_\_unsignedLong16FixedArrayParam \* *pulGainArray*, struct MSXE301x\_\_unsignedLong16FixedArrayParam \* *pulPolarityArray*, xsd\_\_unsignedLong *ulConversionTime*, xsd\_\_unsignedLong *ulConversionTimeUnit*, xsd\_\_unsignedLong *ulNbrOfSequence*, xsd\_\_unsignedLong *ulNbrMaxSequenceToTransfer*, xsd\_\_unsignedLong *ulDelayMode*, xsd\_\_unsignedLong *ulDelayTimeUnit*, xsd\_\_unsignedLong *ulDelayValue*, xsd\_\_unsignedLong *ulTriggerMask*, xsd\_\_unsignedLong *ulTriggerMode*, xsd\_\_unsignedLong *ulHardwareTriggerEdge*, xsd\_\_unsignedLong *ulHardwareTriggerCount*, xsd\_\_unsignedLong *ulByTriggerNbrOfSeqToAcquire*, xsd\_\_unsignedLong *ulDataFormat*, xsd\_\_unsignedLong *ulOption1*, xsd\_\_unsignedLong *ulOption2*, xsd\_\_unsignedLong *ulOption3*, struct MSXE301x\_\_Response \* *Response* )

##### Parameters

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

[in] *pulChannelList* : list of the channel who compose the sequence

[in] *pulGainArray* Define the gain (1 or 2) to use for each channel.

Information : by the channel type "current", the gain 2 is recommended.

- 1 : +10 V or max 20mA when Unipolar, +/-10V or max +/-20mA when Bipolar
  - 2 : +5 V or 20 mA when Unipolar, +/-5V or +/- 20mA when Bipolar Each index of the array correspond to the channel :
- sample :
- [0] : Define the gain for the channel 0
  - [1] : Define the gain for the channel 1
  - ...

Remark: When using a current version of the MSX-E301x, gain 2 has to be used.

[in] *pulPolarityArray* Define the polarity (0:Unipolar or 1:Bipolar) to use for each channel

Each index of the array correspond to the channel :

sample :



- [0] : Define the polarity for the channel 0
  - [1] : Define the polarity for the channel 1
  - ...
- Remark: When using the unipolar mode, the raw value for 0 volt (or 0 ampere) will be 32767.
- [in] ***ulConversionTime*** Conversion Time (min 10 us or 40 us)
- [in] ***ulConversionTimeUnit*** Conversion Time Unit
- 0 : us
  - 1 : ms
- [in] ***ulNbrOfSequence*** : Number of sequence to acquire :
- 0 : continuous mode
  - <> 0 : number of sequence
- [in] ***ulNbrMaxSequenceToTransfer*** : Max nbr of sequence to acquire before a data transfer : (1,65535)
- [in] ***ulDelayMode*** : Delay Mode :
- 0 : delay not used
  - 1 : mode 1
  - 2 : mode 2
- [in] ***ulDelayTimeUnit*** : Selection of the time unit
- 0: us
- 1: ms
- 2: s
- [in] ***ulDelayValue*** : Delay Value : (1..0xFFFF)
- [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*** Define the trigger mode
- 1 : One shot trigger
  - 2 : Sequence trigger
- [in] ***ulHardwareTriggerEdge*** Define the edge of the hardware trigger who generate a trigger action
- 1 : rising front (Only if hardware trigger selected)
  - 2 : falling front (Only if hardware trigger selected)
  - 3 : Both front (Only if hardware trigger selected)
- [in] ***ulHardwareTriggerCount*** Define the number of trigger events before the action occur
- 0 or 1 : all trigger event start the action
  - max value : 65535
- [in] ***ulByTriggerNbrOfSeqToAcquire*** : define the number of sequence to acquire by each trigger event
- 0 : continuous mode
  - <> 0 : number of sequence : (1..0xFFFFFFFF)
- [in] ***ulDataFormat*** : Data format option
- D0 : Time stamp information

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

D1 : Sequence counter information

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

D2 : Data format

- 0 : 32/16 bit digital value
- 1 : 32-bit analog value (in V)

D3 : 16-Bit digital value

- 0 : 32-bit digital/analog value
- 1 : 16-bit digital value.

Only a pair number of acquisition channels (ulNbrOfChannel) can be selected.

[in] **ulOption1** : Reserved

[in] **ulOption2** : Reserved

[in] **ulOption3** : Reserved

[out] **Response** :

**iReturnValue** :

- 0: means the remote function performed OK
- -1: means an system error occured
- -2: The nbr of channel in the sequence cannot be null
- -3: Channel index selection error
- -4: Gain selection error
- -5: Polarity selection error
- -6: The minimal converting time is 10 us !
- -7: Not available conversion time unit
- -8: Delay mode selection not available
- -9: Delay time unit selection not available
- -10: Delay value not available
- -11: Trigger mode : 2 different mode cannot be simultaneously be activated
- -12: Hardware trigger : edge definition error
- -13: Hardware trigger count value not available
- -14: Nbr of sequence to acquire by trigger mode not available
- -15: Data format not available
- -100: Channel initialisation kernel function error
- -101: Conversion time initialisation kernel function error
- -102: Delay initialisation kernel function error
- -103: hardware trigger initialisation/enable kernel function error
- -104: hardware trigger disable kernel function error
- -105: synchro trigger initialisation/enable kernel function error
- -106: synchro trigger disable kernel function error
- -107 Sequence trigger count initialisation error
- -108: Sequence initialisation kernel function error
- -109: Start sequence kernel function error

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

## Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

#### 4.1.2.45 int MSXE301x\_\_AnalogInputStopAndReleaseSequence ( void \* \_\_, struct MSXE301x\_\_Response \* *Response* )

##### Parameters

[in] \_\_ : no input parameter

[out] *Response* :

*iReturnValue* :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: Stop sequence kernel function error
- -101: Release sequence kernel function error

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

##### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

#### 4.1.2.46 int MSXE301x\_\_CalibrationStart ( xsd\_\_unsignedLong *ulCalibrationMode*, xsd\_\_unsignedLong *ulGain*, xsd\_\_double *dCalibrationValue*, struct MSXE301x\_\_Response \* *Response* )

##### Parameters

[in] *ulCalibrationMode* : Calibration mode selection:

- 0 Hex : All channels calibration via a external calibration source (voltage/current). Not available for mixing input types.
- 1 Hex : All channels calibration via the internal voltage calibration source. Not available for current or mixing input types.
- 2 Hex : Channel group 0 (Channel 0 to 3) calibration via a external calibration source (voltage/current).
- 102 Hex : Channel group 1 (Channel 4 to 7) calibration via a external calibration source (voltage/current).
- 202 Hex : Channel group 2 (Channel 8 to 11) calibration via a external calibration source (voltage/current).
- 302 Hex : Channel group 3 (Channel 12 to 15) calibration via a external calibration source (voltage/current).
- 3 Hex : Channel group 0 (Channel 0 to 3) calibration via the internal voltage calibration source. Not available for current inputs.
- 103 Hex : Channel group 1 (Channel 4 to 7) calibration via the internal voltage calibration source. Not available for current inputs.
- 203 Hex : Channel group 2 (Channel 8 to 11) calibration via the internal voltage calibration source. Not available for current inputs.
- 303 Hex : Channel group 3 (Channel 12 to 15) calibration via the internal voltage calibration source. Not available for current inputs.

[in] *ulGain* : Gain selection 1 or 2

[in] *dCalibrationValue* : Calibration value

[out] *Response* :

*iReturnValue* : Error value :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -2: Calibration mode selection error
- -3: Calibration gain selection error
- -4: Calibration value selection error
- -100: Start calibration kernel function error

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

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

**4.1.2.47** `int MSXE301x__CalibrationGetCurrentStatus ( void * __, struct MSXE301x__CalibrationGetCurrentStatusResponse * Response )`

#### Parameters

[in] **\_** : no input parameter

[out] **Response** :

**iReturnValue** : Error value :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: Get status calibration kernel function error
- -101: Save calibration read source error
- -102: Save calibration write destination error

**ulStatus** : TODO

**ulDigitalValue** : Last measured digital value

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

**4.1.2.48** `int MSXE301x__CalibrationNextStep ( void * __, struct MSXE301x__Response * Response )`

#### Parameters

[in] **\_** : no input parameter

[out] **Response** :

**iReturnValue** : Error value :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: Calibration next step kernel function error

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

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

**4.1.2.49** `int MSXE301x__CalibrationBreak ( void * _, struct MSXE301x__Response * Response )`

#### Parameters

[in] `_` : no input parameter

[out] *Response* :

*iReturnValue* : Error value :

- 0: means the remote function performed OK
- -1: means an system error occurred
- -100: Calibration break kernel function error

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

#### Returns

- 0: SOAP\_OK
- <> 0: See SOAP error

# Index

- `__offset`
    - ByteArray, [53](#)
    - UnsignedLongArray, [69](#)
    - UnsignedShortArray, [70](#)
  - `__ptr`
    - ByteArray, [53](#)
    - UnsignedLongArray, [69](#)
    - UnsignedShortArray, [70](#)
    - `xsd__base64Binary`, [70](#)
  - `__size`
    - ByteArray, [53](#)
    - UnsignedLongArray, [69](#)
    - UnsignedShortArray, [70](#)
    - `xsd__base64Binary`, [70](#)
- Analog
  - `MXCommon__SetFilterChannels`, [37](#)
- bArray
  - `MXCommon__-`
    - `GetAutoConfigurationFileResponse`, [64](#)
- bCryptedValueArray
  - `MXCommon__TestCustomerIDResponse`, [68](#)
- bValueArray
  - `MXCommon__TestCustomerIDResponse`, [68](#)
- ByteArray, [53](#)
  - `__offset`, [53](#)
  - `__ptr`, [53](#)
  - `__size`, [53](#)
- 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__Sterror`, [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](#)
- dCurrentValue

- MSXE301x\_\_-
  - CalibrationGetCurrentStatusResponse, 60
- DefaultResponse, 53
- iReturnValue, 54
- syserrno, 54
- dTemperatureValue
  - MXCommon\_\_-
    - GetModuleTemperatureValueAndStatusResponse, 66
- input filter Filter management, 36
- iReturnValue
  - DefaultResponse, 54
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, 57
  - MSXE301x\_\_Response, 61
  - MXCommon\_\_Response, 67
- MSX-E systems servers, 10
- MSX-E301x Autorefresh functions, 39
- MSX-E301x calibration functions, 48
- MSX-E301x functions, 37
- MSX-E301x Information functions, 37
- MSX-E301x Sequence functions, 44
- MSXE301x\_\_AnalogInputGetAutoRefreshValues
  - MSXE301x\_Autorefresh, 42
  - MSXE301x\_public\_doc.h, 99
- MSXE301x\_\_AnalogInputGetAutoRefreshValuesResponse, 54
  - sResponse, 55
  - ulCounterValue, 55
  - ulTimeStampHigh, 55
  - ulTimeStampLow, 55
  - ulValue, 55
- MSXE301x\_\_AnalogInputGetChannelType
  - MSXE301x\_Information, 38
  - MSXE301x\_public\_doc.h, 95
- MSXE301x\_\_AnalogInputGetChannelTypeResponse, 55
  - sResponse, 55
  - ulType, 55
- MSXE301x\_\_AnalogInputInitAndStartAutoRefresh
  - MSXE301x\_Autorefresh, 40
  - MSXE301x\_public\_doc.h, 96
- MSXE301x\_\_AnalogInputInitAndStartSequence
  - MSXE301x\_public\_doc.h, 100
  - MSXE301x\_Sequence, 45
- MSXE301x\_\_AnalogInputStopAndReleaseAutoRefresh
  - MSXE301x\_Autorefresh, 43
  - MSXE301x\_public\_doc.h, 99
- MSXE301x\_\_AnalogInputStopAndReleaseSequence
  - MSXE301x\_public\_doc.h, 102
- MSXE301x\_Sequence, 47
- MSXE301x\_\_AnalogMeasureGetConfiguration
  - MSXE301x\_Information, 38
  - MSXE301x\_public\_doc.h, 96
- MSXE301x\_\_AnalogMeasureGetConfigurationResponse, 55
  - iReturnValue, 57
  - sAcquisition, 57
  - sAutoRefresh, 57
  - sHardware, 59
  - sSequence, 59
  - sSoftware, 59
  - sSynchro, 59
  - sTrigger, 59
  - ulAction, 59
  - ulAverageMode, 57
  - ulAverageValue, 57
  - ulChannelInitialization, 57
  - ulChannelMask, 57
  - ulConvertTime, 57
  - ulConvertTimeUnit, 57
  - ulCount, 59
  - ulDataFormat, 57
  - ulDelayFlag, 58
  - ulDelayMode, 59
  - ulDelayTime, 59
  - ulDelayTimeUnit, 59
  - ulFlag, 59
  - ulLevel, 59
  - ulNbrOfSequence, 58
  - ulRunningMode, 57
  - ulSequence, 58
  - ulSequenceInterrupt, 58
  - ulSequenceSize, 58
  - ulSequenceTriggerCount, 59
- MSXE301x\_\_CalibrationBreak
  - MSXE301x\_Calib, 50
  - MSXE301x\_public\_doc.h, 104
- MSXE301x\_\_CalibrationGetCurrentStatus
  - MSXE301x\_Calib, 50
  - MSXE301x\_public\_doc.h, 104
- MSXE301x\_\_CalibrationGetCurrentStatusResponse, 59
  - dCurrentValue, 60
  - sResponse, 60
  - ulError, 60
  - ulStatus, 60
- MSXE301x\_\_CalibrationNextStep
  - MSXE301x\_Calib, 50
  - MSXE301x\_public\_doc.h, 104
- MSXE301x\_\_CalibrationStart
  - MSXE301x\_Calib, 49
  - MSXE301x\_public\_doc.h, 103
- MSXE301x\_\_Response, 61

- iReturnValue, [61](#)
- syserrno, [61](#)
- MSXE301x\_\_unsignedLong16FixedArrayParam, [62](#)
  - ulValue, [62](#)
- MSXE301x\_\_unsignedLongResponse, [62](#)
  - sResponse, [62](#)
  - ulValue, [62](#)
- MSXE301x\_Autorefresh
  - MSXE301x\_\_-
    - AnalogInputGetAutoRefreshValues, [42](#)
  - MSXE301x\_\_-
    - AnalogInputInitAndStartAutoRefresh, [40](#)
  - MSXE301x\_\_-
    - AnalogInputStopAndReleaseAutoRefresh, [43](#)
- MSXE301x\_Calib
  - MSXE301x\_\_CalibrationBreak, [50](#)
  - MSXE301x\_\_CalibrationGetCurrentStatus, [50](#)
  - MSXE301x\_\_CalibrationNextStep, [50](#)
  - MSXE301x\_\_CalibrationStart, [49](#)
- MSXE301x\_Information
  - MSXE301x\_\_AnalogInputGetChannelType, [38](#)
  - MSXE301x\_\_-
    - AnalogMeasureGetConfiguration, [38](#)
- MSXE301x\_public\_doc.h, [71](#)
  - MSXE301x\_\_-
    - AnalogInputGetAutoRefreshValues, [99](#)
  - MSXE301x\_\_AnalogInputGetChannelType, [95](#)
  - MSXE301x\_\_-
    - AnalogInputInitAndStartAutoRefresh, [96](#)
  - MSXE301x\_\_-
    - AnalogInputInitAndStartSequence, [100](#)
  - MSXE301x\_\_-
    - AnalogInputStopAndReleaseAutoRefresh, [99](#)
  - MSXE301x\_\_-
    - AnalogInputStopAndReleaseSequence, [102](#)
  - MSXE301x\_\_-
    - AnalogMeasureGetConfiguration, [96](#)
  - MSXE301x\_\_CalibrationBreak, [104](#)
  - MSXE301x\_\_CalibrationGetCurrentStatus, [104](#)
  - MSXE301x\_\_CalibrationNextStep, [104](#)
  - MSXE301x\_\_CalibrationStart, [103](#)
  - MXCommon\_\_-
    - ApplyConfigurationBackupFile, [90](#)
  - MXCommon\_\_ChangePassword, [91](#)
  - MXCommon\_\_DataserverRestart, [80](#)
  - MXCommon\_\_GetAutoConfigurationFile, [87](#)
  - MXCommon\_\_GetClientConnections, [78](#)
  - MXCommon\_\_GetConfigurationBackupFile, [90](#)
  - MXCommon\_\_GetEthernetLinksStates, [81](#)
  - MXCommon\_\_-
    - GetHardwareTriggerFilterTime, [83](#)
  - MXCommon\_\_GetHardwareTriggerState, [84](#)
  - MXCommon\_\_GetHostname, [77](#)
  - MXCommon\_\_-
    - GetModuleTemperatureValueAndStatus, [82](#)
  - MXCommon\_\_GetModuleType, [77](#)
  - MXCommon\_\_GetOptionInformation, [93](#)
  - MXCommon\_\_GetStateIDFromName, [92](#)
  - MXCommon\_\_GetStateNameFromID, [93](#)
  - MXCommon\_\_GetSubsystemIDFromName, [92](#)
  - MXCommon\_\_GetSubsystemNameFromID, [93](#)
  - MXCommon\_\_GetSubSystemState, [91](#)
  - MXCommon\_\_GetSynchronizationStatus, [94](#)
  - MXCommon\_\_GetTime, [86](#)
  - MXCommon\_\_GetUpTime, [87](#)
  - MXCommon\_\_HardwareClockToSys, [86](#)
  - MXCommon\_\_InitAndStartSynchroTimer, [88](#)
  - MXCommon\_\_Reboot, [80](#)
  - MXCommon\_\_ResetAllIOFunctionalities, [80](#)
  - MXCommon\_\_SetAutoConfigurationFile, [88](#)
  - MXCommon\_\_SetCustomerKey, [84](#)
  - MXCommon\_\_SetFilterChannels, [95](#)
  - MXCommon\_\_-
    - SetHardwareTriggerFilterTime, [83](#)
  - MXCommon\_\_SetHostname, [78](#)
  - MXCommon\_\_-
    - SetModuleTemperatureWarningLevels, [82](#)
  - MXCommon\_\_SetTime, [85](#)
  - MXCommon\_\_SetToMaster, [94](#)
  - MXCommon\_\_StartAutoConfiguration, [88](#)
  - MXCommon\_\_-
    - StopAndReleaseSynchroTimer, [89](#)
  - MXCommon\_\_Sterror, [78](#)
  - MXCommon\_\_SysToHardwareClock, [86](#)
  - MXCommon\_\_TestCustomerID, [85](#)
  - xsd\_\_char, [77](#)
  - xsd\_\_double, [77](#)
  - xsd\_\_float, [77](#)
  - xsd\_\_int, [77](#)
  - xsd\_\_long, [77](#)



- xsd\_\_string, 77
- xsd\_\_unsignedByte, 77
- xsd\_\_unsignedInt, 77
- xsd\_\_unsignedLong, 77
- xsd\_\_unsignedShort, 77
- MSXE301x\_Sequence
  - MSXE301x\_\_-
    - AnalogInputInitAndStartSequence, 45
  - MSXE301x\_\_-
    - AnalogInputStopAndReleaseSequence, 47
- MXCommon\_\_ApplyConfigurationBackupFile
  - Common\_configuration, 30
  - MSXE301x\_public\_doc.h, 90
- MXCommon\_\_ByteArrayResponse, 62
  - sArray, 63
  - sResponse, 63
- MXCommon\_\_ChangePassword
  - Common\_configuration, 31
  - MSXE301x\_public\_doc.h, 91
- MXCommon\_\_DataseverRestart
  - Common\_general, 15
  - MSXE301x\_public\_doc.h, 80
- MXCommon\_\_FileResponse, 63
  - sArray, 63
  - sResponse, 63
  - ulEOF, 63
- MXCommon\_\_GetAutoConfigurationFile
  - Common\_autoconf, 26
  - MSXE301x\_public\_doc.h, 87
- MXCommon\_\_GetAutoConfigurationFileResponse, 63
  - bArray, 64
  - sResponse, 64
  - ulEOF, 64
- MXCommon\_\_GetClientConnections
  - Common\_general, 13
  - MSXE301x\_public\_doc.h, 78
- MXCommon\_\_GetConfigurationBackupFile
  - Common\_configuration, 30
  - MSXE301x\_public\_doc.h, 90
- MXCommon\_\_GetEthernetLinksStates
  - Common\_general, 16
  - MSXE301x\_public\_doc.h, 81
- MXCommon\_\_GetEthernetLinksStatesResponse, 64
  - sPort0, 64
  - sPort1, 64
  - sResponse, 64
- MXCommon\_\_GetHardwareTriggerFilterTime
  - Common\_hardware\_trigger, 19
  - MSXE301x\_public\_doc.h, 83
- MXCommon\_\_GetHardwareTriggerFilterTimeResponse, 64
  - sResponse, 65
  - ulFilterTime, 65
  - ulInfo01, 65
  - ulInfo02, 65
- MXCommon\_\_GetHardwareTriggerState
  - Common\_hardware\_trigger, 20
  - MSXE301x\_public\_doc.h, 84
- MXCommon\_\_GetHardwareTriggerStateResponse, 65
  - sResponse, 65
  - ulInfo01, 65
  - ulInfo02, 65
  - ulState, 65
- MXCommon\_\_GetHostname
  - Common\_general, 12
  - MSXE301x\_public\_doc.h, 77
- MXCommon\_\_GetModuleTemperatureValueAndStatus
  - Common\_temperature, 17
  - MSXE301x\_public\_doc.h, 82
- MXCommon\_\_GetModuleTemperatureValueAndStatusResponse, 65
  - dTemperatureValue, 66
  - sResponse, 66
  - ulInfo, 66
  - ulTemperatureStatus, 66
- MXCommon\_\_GetModuleType
  - Common\_general, 12
  - MSXE301x\_public\_doc.h, 77
- MXCommon\_\_GetOptionInformation
  - CustomerOption, 35
  - MSXE301x\_public\_doc.h, 93
- MXCommon\_\_GetStateIDFromName
  - MSXE301x\_public\_doc.h, 92
  - SystemStatemanagement, 33
- MXCommon\_\_GetStateNameFromID
  - MSXE301x\_public\_doc.h, 93
  - SystemStatemanagement, 34
- MXCommon\_\_GetSubsystemIDFromName
  - MSXE301x\_public\_doc.h, 92
  - SystemStatemanagement, 33
- MXCommon\_\_GetSubsystemNameFromID
  - MSXE301x\_public\_doc.h, 93
  - SystemStatemanagement, 33
- MXCommon\_\_GetSubSystemState
  - MSXE301x\_public\_doc.h, 91
  - SystemStatemanagement, 32
- MXCommon\_\_GetSynchronizationStatus
  - MSXE301x\_public\_doc.h, 94
  - Synchronisation, 36
- MXCommon\_\_GetTime
  - Common\_time, 24
  - MSXE301x\_public\_doc.h, 86

- MXCommon\_\_GetTimeResponse, 66
  - sResponse, 66
  - ulHighTime, 66
  - ulLowTime, 66
- MXCommon\_\_GetUpTime
  - Common\_time, 25
  - MSXE301x\_public\_doc.h, 87
- MXCommon\_\_GetUpTimeResponse, 66
  - sResponse, 67
  - ulUpTime, 67
- MXCommon\_\_HardwareClockToSys
  - Common\_time, 24
  - MSXE301x\_public\_doc.h, 86
- MXCommon\_\_InitAndStartSynchroTimer
  - Common\_synchrotimer, 28
  - MSXE301x\_public\_doc.h, 88
- MXCommon\_\_Reboot
  - Common\_general, 15
  - MSXE301x\_public\_doc.h, 80
- MXCommon\_\_ResetAllIOFunctionalities
  - Common\_general, 15
  - MSXE301x\_public\_doc.h, 80
- MXCommon\_\_Response, 67
  - iReturnValue, 67
  - syserrno, 67
- MXCommon\_\_SetAutoConfigurationFile
  - Common\_autoconf, 26
  - MSXE301x\_public\_doc.h, 88
- MXCommon\_\_SetCustomerKey
  - Common\_security, 22
  - MSXE301x\_public\_doc.h, 84
- MXCommon\_\_SetFilterChannels
  - Analog, 37
  - MSXE301x\_public\_doc.h, 95
- MXCommon\_\_SetHardwareTriggerFilterTime
  - Common\_hardware\_trigger, 19
  - MSXE301x\_public\_doc.h, 83
- MXCommon\_\_SetHostname
  - Common\_general, 13
  - MSXE301x\_public\_doc.h, 78
- MXCommon\_\_SetModuleTemperatureWarningLevels
  - Common\_temperature, 18
  - MSXE301x\_public\_doc.h, 82
- MXCommon\_\_SetTime
  - Common\_time, 23
  - MSXE301x\_public\_doc.h, 85
- MXCommon\_\_SetToMaster
  - MSXE301x\_public\_doc.h, 94
  - Synchronisation, 35
- MXCommon\_\_StartAutoConfiguration
  - Common\_autoconf, 27
  - MSXE301x\_public\_doc.h, 88
- MXCommon\_\_StopAndReleaseSynchroTimer
  - Common\_synchrotimer, 28
  - MSXE301x\_public\_doc.h, 89
- MXCommon\_\_Sterror
  - Common\_general, 13
  - MSXE301x\_public\_doc.h, 78
- MXCommon\_\_SysToHardwareClock
  - Common\_time, 23
  - MSXE301x\_public\_doc.h, 86
- MXCommon\_\_TestCustomerID
  - Common\_security, 22
  - MSXE301x\_public\_doc.h, 85
- MXCommon\_\_TestCustomerIDResponse, 67
  - bCryptedValueArray, 68
  - bValueArray, 68
  - sResponse, 68
- MXCommon\_\_unsignedLongResponse, 68
  - sResponse, 68
  - ulValue, 68
- sAcquisition
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, 57
- sArray
  - MXCommon\_\_ByteArrayResponse, 63
  - MXCommon\_\_FileResponse, 63
- sAutoRefresh
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, 57
- Set/Backup/Restore general system configuration, 29
- sGetEthernetLinksStatesPort, 68
  - ulDuplex, 69
  - ulInfo1, 69
  - ulInfo2, 69
  - ulSpeed, 69
  - ulState, 69
- sHardware
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, 59
- SOAP function calls in C/C++ language, 3
- Software hints, 3
- sPort0
  - MXCommon\_\_-
    - GetEthernetLinksStatesResponse, 64
- sPort1
  - MXCommon\_\_-
    - GetEthernetLinksStatesResponse, 64
- sResponse
  - MSXE301x\_\_-
    - AnalogInputGetAutoRefreshValuesResponse, 55

- MSXE301x\_\_-
  - AnalogInputGetChannelTypeResponse, [55](#)
- MSXE301x\_\_-
  - CalibrationGetCurrentStatusResponse, [60](#)
- MSXE301x\_\_unsignedLongResponse, [62](#)
- MXCommon\_\_ByteArrayResponse, [63](#)
- MXCommon\_\_FileResponse, [63](#)
- MXCommon\_\_-
  - GetAutoConfigurationFileResponse, [64](#)
- MXCommon\_\_-
  - GetEthernetLinksStatesResponse, [64](#)
- MXCommon\_\_-
  - GetHardwareTriggerFilterTimeResponse, [65](#)
- MXCommon\_\_-
  - GetHardwareTriggerStateResponse, [65](#)
- MXCommon\_\_-
  - GetModuleTemperatureValueAndStatusResponse, [66](#)
- MXCommon\_\_GetTimeResponse, [66](#)
- MXCommon\_\_GetUpTimeResponse, [67](#)
- MXCommon\_\_TestCustomerIDResponse, [68](#)
- MXCommon\_\_unsignedLongResponse, [68](#)
- sSequence
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [59](#)
- sSoftware
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [59](#)
- sSynchro
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [59](#)
- sTrigger
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [59](#)
- Synchronisation
  - MXCommon\_\_GetSynchronizationStatus, [36](#)
  - MXCommon\_\_SetToMaster, [35](#)
- Synchronisation management, [35](#)
- syserrno
  - DefaultResponse, [54](#)
  - MSXE301x\_\_Response, [61](#)
  - MXCommon\_\_Response, [67](#)
- System state management, [31](#)
- SystemStatemanagement
  - MXCommon\_\_GetStateIDFromName, [33](#)
  - MXCommon\_\_GetStateNameFromID, [34](#)
  - MXCommon\_\_GetSubsystemIDFromName, [33](#)
  - MXCommon\_\_GetSubsystemNameFromID, [33](#)
  - MXCommon\_\_GetSubSystemState, [32](#)
- ulAction
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [59](#)
- ulAverageMode
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [57](#)
- ulAverageValue
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [57](#)
- ulChannelInitialization
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [57](#)
- ulChannelMask
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [57](#)
- ulConvertTime
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [57](#)
- ulConvertTimeUnit
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [57](#)
- ulCount
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [59](#)
- ulCounterValue
  - MSXE301x\_\_-
    - AnalogInputGetAutoRefreshValuesResponse, [55](#)
- ulDataFormat
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [57](#)
- ulDelayFlag
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [58](#)
- ulDelayMode
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, [58](#)

- 59
- ulDelayTime
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, 59
    - ulLowTime
      - MXCommon\_\_GetTimeResponse, 66
- ulDelayTimeUnit
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, 59
- ulDuplex
  - sGetEthernetLinksStatesPort, 69
- ulEOF
  - MXCommon\_\_FileResponse, 63
  - MXCommon\_\_-
    - GetAutoConfigurationFileResponse, 64
- ulError
  - MSXE301x\_\_-
    - CalibrationGetCurrentStatusResponse, 60
- ulFilterTime
  - MXCommon\_\_-
    - GetHardwareTriggerFilterTimeResponse, 65
- ulFlag
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, 59
- ulHighTime
  - MXCommon\_\_GetTimeResponse, 66
- ulInfo
  - MXCommon\_\_-
    - GetModuleTemperatureValueAndStatusResponse, 66
- ulInfo01
  - MXCommon\_\_-
    - GetHardwareTriggerFilterTimeResponse, 65
  - MXCommon\_\_-
    - GetHardwareTriggerStateResponse, 65
- ulInfo02
  - MXCommon\_\_-
    - GetHardwareTriggerFilterTimeResponse, 65
  - MXCommon\_\_-
    - GetHardwareTriggerStateResponse, 65
- ulInfo1
  - sGetEthernetLinksStatesPort, 69
- ulInfo2
  - sGetEthernetLinksStatesPort, 69
- ulLevel
  - MSXE301x\_\_-
    - AnalogMeasureGetConfigurationResponse, 59
  - ulNbrOfSequence
    - MSXE301x\_\_-
      - AnalogMeasureGetConfigurationResponse, 58
  - ulRunningMode
    - MSXE301x\_\_-
      - AnalogMeasureGetConfigurationResponse, 57
  - ulSequence
    - MSXE301x\_\_-
      - AnalogMeasureGetConfigurationResponse, 58
  - ulSequenceInterrupt
    - MSXE301x\_\_-
      - AnalogMeasureGetConfigurationResponse, 58
  - ulSequenceSize
    - MSXE301x\_\_-
      - AnalogMeasureGetConfigurationResponse, 58
  - ulSequenceTriggerCount
    - MSXE301x\_\_-
      - AnalogMeasureGetConfigurationResponse, 59
  - ulSpeed
    - sGetEthernetLinksStatesPort, 69
  - ulState
    - MXCommon\_\_-
      - GetHardwareTriggerStateResponse, 65
    - sGetEthernetLinksStatesPort, 69
  - ulStatus
    - MSXE301x\_\_-
      - CalibrationGetCurrentStatusResponse, 60
  - ulTemperatureStatus
    - MXCommon\_\_-
      - GetModuleTemperatureValueAndStatusResponse, 66
  - ulTimeStampHigh
    - MSXE301x\_\_-
      - AnalogInputGetAutoRefreshValuesResponse, 55
  - ulTimeStampLow
    - MSXE301x\_\_-
      - AnalogInputGetAutoRefreshValuesResponse, 55
  - ulType

- MSXE301x\_\_-
  - AnalogInputGetChannelTypeResponse, [55](#)
- ulUpTime
  - MXCommon\_\_GetUpTimeResponse, [67](#)
- ulValue
  - MSXE301x\_\_-
    - AnalogInputGetAutoRefreshValuesResponse, [55](#)
  - MSXE301x\_\_-
    - unsignedLong16FixedArrayParam, [62](#)
  - MSXE301x\_\_unsignedLongResponse, [62](#)
  - MXCommon\_\_unsignedLongResponse, [68](#)
- UnsignedLongArray, [69](#)
  - \_\_offset, [69](#)
  - \_\_ptr, [69](#)
  - \_\_size, [69](#)
- UnsignedShortArray, [69](#)
  - \_\_offset, [70](#)
  - \_\_ptr, [70](#)
  - \_\_size, [70](#)
- xsd\_\_base64Binary, [70](#)
  - \_\_ptr, [70](#)
  - \_\_size, [70](#)
- xsd\_\_char
  - MSXE301x\_public\_doc.h, [77](#)
- xsd\_\_double
  - MSXE301x\_public\_doc.h, [77](#)
- xsd\_\_float
  - MSXE301x\_public\_doc.h, [77](#)
- xsd\_\_int
  - MSXE301x\_public\_doc.h, [77](#)
- xsd\_\_long
  - MSXE301x\_public\_doc.h, [77](#)
- xsd\_\_string
  - MSXE301x\_public\_doc.h, [77](#)
- xsd\_\_unsignedByte
  - MSXE301x\_public\_doc.h, [77](#)
- xsd\_\_unsignedInt
  - MSXE301x\_public\_doc.h, [77](#)
- xsd\_\_unsignedLong
  - MSXE301x\_public\_doc.h, [77](#)
- xsd\_\_unsignedShort
  - MSXE301x\_public\_doc.h, [77](#)