

## MSXE331x soap api functions

Generated by Doxygen 1.7.1

Fri Oct 24 2014 17:58:53



# 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	MSX-E331x functions	3
2.2	Common functions	3
2.3	Common general functions	4
2.3.1	Function Documentation	5
2.3.1.1	MXCommon__GetModuleType	5
2.3.1.2	MXCommon__GetHostname	5
2.3.1.3	MXCommon__SetHostname	6
2.3.1.4	MXCommon__GetClientConnections	6
2.3.1.5	MXCommon__Sterror	6
2.3.1.6	MXCommon__Reboot	8
2.3.1.7	MXCommon__ResetAllIOFunctionalities	8
2.3.1.8	MXCommon__DataseverRestart	8
2.3.1.9	MXCommon__GetEthernetLinksStates	9
2.4	Common temperature functions	10
2.4.1	Detailed Description	10
2.4.2	Function Documentation	10
2.4.2.1	MXCommon__GetModuleTemperatureValueAndStatus	10
2.4.2.2	MXCommon__SetModuleTemperatureWarningLevels	11
2.5	Common hardware trigger functions	11
2.5.1	Function Documentation	12
2.5.1.1	MXCommon__SetHardwareTriggerFilterTime	12
2.5.1.2	MXCommon__GetHardwareTriggerFilterTime	12
2.5.1.3	MXCommon__GetHardwareTriggerState	13

2.6	Common security functions . . . . .	13
2.6.1	Detailed Description . . . . .	14
2.6.2	Function Documentation . . . . .	15
2.6.2.1	MXCommon__SetCustomerKey . . . . .	15
2.6.2.2	MXCommon__TestCustomerID . . . . .	15
2.7	Common time functions . . . . .	15
2.7.1	Detailed Description . . . . .	16
2.7.2	Function Documentation . . . . .	16
2.7.2.1	MXCommon__SetTime . . . . .	16
2.7.2.2	MXCommon__SysToHardwareClock . . . . .	17
2.7.2.3	MXCommon__HardwareClockToSys . . . . .	17
2.7.2.4	MXCommon__GetTime . . . . .	17
2.7.2.5	MXCommon__GetUpTime . . . . .	18
2.8	Common I/O auto configuration functions . . . . .	18
2.8.1	Detailed Description . . . . .	19
2.8.2	Function Documentation . . . . .	19
2.8.2.1	MXCommon__GetAutoConfigurationFile . . . . .	19
2.8.2.2	MXCommon__SetAutoConfigurationFile . . . . .	19
2.8.2.3	MXCommon__StartAutoConfiguration . . . . .	20
2.9	Common synchronisation timer functions . . . . .	20
2.9.1	Function Documentation . . . . .	21
2.9.1.1	MXCommon__InitAndStartSynchroTimer . . . . .	21
2.9.1.2	MXCommon__StopAndReleaseSynchroTimer . . . . .	22
2.10	Set/Backup/Restore general system configuration . . . . .	22
2.10.1	Detailed Description . . . . .	22
2.10.2	Function Documentation . . . . .	23
2.10.2.1	MXCommon__GetConfigurationBackupFile . . . . .	23
2.10.2.2	MXCommon__ApplyConfigurationBackupFile . . . . .	23
2.10.2.3	MXCommon__ChangePassword . . . . .	24
2.11	System state management . . . . .	24
2.11.1	Detailed Description . . . . .	25
2.11.2	Function Documentation . . . . .	25
2.11.2.1	MXCommon__GetSubSystemState . . . . .	25
2.11.2.2	MXCommon__GetSubsystemIDFromName . . . . .	26
2.11.2.3	MXCommon__GetStateIDFromName . . . . .	26
2.11.2.4	MXCommon__GetSubsystemNameFromID . . . . .	27

2.11.2.5	MXCommon__GetStateNameFromID	27
2.12	Customer option management	27
2.12.1	Function Documentation	28
2.12.1.1	MXCommon__GetOptionInformation	28
2.13	Synchronisation management	28
2.13.1	Function Documentation	28
2.13.1.1	MXCommon__SetToMaster	28
2.13.1.2	MXCommon__GetSynchronizationStatus	29
2.14	MSX-E331x Acquisition functions	29
2.15	MSX-E331x Acquisition information functions	30
2.15.1	Function Documentation	30
2.15.1.1	MSXExxxx__AcquisitionGetNumberOfChannels	30
2.15.1.2	MSXExxxx__AcquisitionGetChannelInfo	31
2.16	MSX-E331x Autorefresh functions	31
2.16.1	Detailed Description	32
2.16.2	Function Documentation	33
2.16.2.1	MSXExxxx__AcquisitionAutoRefreshInitAndStart	33
2.16.2.2	MSXExxxx__AcquisitionAutoRefreshGetValues	35
2.16.2.3	MSXExxxx__AcquisitionAutoRefreshStopAndRelease	35
2.17	MSX-E331x Sequence functions	36
2.17.1	Detailed Description	36
2.17.2	Function Documentation	37
2.17.2.1	MSXExxxx__AcquisitionSequenceInitAndStart	37
2.17.2.2	MSXExxxx__AcquisitionSequenceStopAndRelease	39
2.18	MSX-E331x Pressure functions	40
2.19	MSX-E331x Pressure initialisation/information functions	40
2.19.1	Function Documentation	41
2.19.1.1	MSXExxxx__PressureGetNumberOfChannels	41
2.19.1.2	MSXExxxx__PressureSetChannelConfiguration	41
2.19.1.3	MSXExxxx__PressureSetSamplingRate	42
2.19.1.4	MSXExxxx__PressureGetConfiguration	43
<b>3</b>	<b>Data Structure Documentation</b>	<b>45</b>
3.1	ByteArray Struct Reference	45
3.1.1	Field Documentation	45
3.1.1.1	__ptr	45
3.1.1.2	__size	45

3.1.1.3	<code>__offset</code>	45
3.2	DefaultResponse Struct Reference	45
3.2.1	Field Documentation	46
3.2.1.1	<code>iReturnValue</code>	46
3.2.1.2	<code>syserrno</code>	46
3.3	MSXExxxx__AcquisitionAutoRefreshGetValuesResponse Struct Reference	46
3.3.1	Field Documentation	47
3.3.1.1	<code>sResponse</code>	47
3.3.1.2	<code>ulTimeStampLow</code>	47
3.3.1.3	<code>ulTimeStampHigh</code>	47
3.3.1.4	<code>ulCounterValue</code>	47
3.3.1.5	<code>dValue</code>	47
3.4	MSXExxxx__AcquisitionGetChannelInfoResponse Struct Reference	47
3.4.1	Field Documentation	47
3.4.1.1	<code>sResponse</code>	47
3.4.1.2	<code>ulType</code>	47
3.4.1.3	<code>ulHwPosition</code>	48
3.4.1.4	<code>ulChannelIndex</code>	48
3.5	MSXExxxx__AcquisitionSequenceInitAndStartChannelListParam Struct Reference	48
3.5.1	Field Documentation	48
3.5.1.1	<code>ulChannelList</code>	48
3.6	MSXExxxx__FileResponse Struct Reference	48
3.6.1	Field Documentation	49
3.6.1.1	<code>sResponse</code>	49
3.6.1.2	<code>sArray</code>	49
3.6.1.3	<code>ulEOF</code>	49
3.7	MSXExxxx__PressureGetConfigurationResponse Struct Reference	49
3.7.1	Field Documentation	49
3.7.1.1	<code>sResponse</code>	49
3.7.1.2	<code>dSensorSensibility</code>	49
3.7.1.3	<code>dSensorOffset</code>	49
3.7.1.4	<code>ulBaseSamplingRate</code>	49
3.8	MSXExxxx__Response Struct Reference	49
3.8.1	Field Documentation	50
3.8.1.1	<code>iReturnValue</code>	50
3.8.1.2	<code>syserrno</code>	50

3.9	MSXExxxx__unsignedLongResponse Struct Reference	50
3.9.1	Field Documentation	50
3.9.1.1	sResponse	50
3.9.1.2	ulValue	50
3.10	MSXExxxx__unsignedLongTimeStampResponse Struct Reference	50
3.10.1	Field Documentation	51
3.10.1.1	sResponse	51
3.10.1.2	ulValue	51
3.10.1.3	ulTimeStampLow	51
3.10.1.4	ulTimeStampHigh	51
3.11	MXCommon__ByteArrayResponse Struct Reference	51
3.11.1	Field Documentation	51
3.11.1.1	sResponse	51
3.11.1.2	sArray	51
3.12	MXCommon__FileResponse Struct Reference	51
3.12.1	Field Documentation	52
3.12.1.1	sResponse	52
3.12.1.2	sArray	52
3.12.1.3	ulEOF	52
3.13	MXCommon__GetAutoConfigurationFileResponse Struct Reference	52
3.13.1	Field Documentation	52
3.13.1.1	sResponse	52
3.13.1.2	bArray	52
3.13.1.3	ulEOF	52
3.14	MXCommon__GetEthernetLinksStatesResponse Struct Reference	52
3.14.1	Field Documentation	53
3.14.1.1	sResponse	53
3.14.1.2	sPort0	53
3.14.1.3	sPort1	53
3.15	MXCommon__GetHardwareTriggerFilterTimeResponse Struct Reference	53
3.15.1	Field Documentation	53
3.15.1.1	sResponse	53
3.15.1.2	ulFilterTime	53
3.15.1.3	ulInfo01	53
3.15.1.4	ulInfo02	53
3.16	MXCommon__GetHardwareTriggerStateResponse Struct Reference	53

3.16.1	Field Documentation	54
3.16.1.1	sResponse	54
3.16.1.2	ulState	54
3.16.1.3	ulInfo01	54
3.16.1.4	ulInfo02	54
3.17	MXCommon__GetModuleTemperatureValueAndStatusResponse Struct Reference	54
3.17.1	Field Documentation	55
3.17.1.1	sResponse	55
3.17.1.2	dTemperatureValue	55
3.17.1.3	ulTemperatureStatus	55
3.17.1.4	ulInfo	55
3.18	MXCommon__GetTimeResponse Struct Reference	55
3.18.1	Field Documentation	55
3.18.1.1	sResponse	55
3.18.1.2	ulLowTime	55
3.18.1.3	ulHighTime	55
3.19	MXCommon__GetUpTimeResponse Struct Reference	55
3.19.1	Field Documentation	56
3.19.1.1	sResponse	56
3.19.1.2	ulUpTime	56
3.20	MXCommon__Response Struct Reference	56
3.20.1	Field Documentation	56
3.20.1.1	iReturnValue	56
3.20.1.2	syserrno	56
3.21	MXCommon__TestCustomerIDResponse Struct Reference	56
3.21.1	Field Documentation	57
3.21.1.1	sResponse	57
3.21.1.2	bValueArray	57
3.21.1.3	bCryptedValueArray	57
3.22	MXCommon__unsignedLongResponse Struct Reference	57
3.22.1	Field Documentation	57
3.22.1.1	sResponse	57
3.22.1.2	ulValue	57
3.23	sGetEthernetLinksStatesPort Struct Reference	57
3.23.1	Field Documentation	58
3.23.1.1	ulState	58



3.23.1.2	ulSpeed	58
3.23.1.3	ulDuplex	58
3.23.1.4	ulInfo1	58
3.23.1.5	ulInfo2	58
3.24	UnsignedLongArray Struct Reference	58
3.24.1	Field Documentation	58
3.24.1.1	__ptr	58
3.24.1.2	__size	58
3.24.1.3	__offset	58
3.25	UnsignedShortArray Struct Reference	58
3.25.1	Field Documentation	59
3.25.1.1	__ptr	59
3.25.1.2	__size	59
3.25.1.3	__offset	59
3.26	xsd__base64Binary Struct Reference	59
3.26.1	Field Documentation	59
3.26.1.1	__ptr	59
3.26.1.2	__size	59
<b>4</b>	<b>File Documentation</b>	<b>61</b>
4.1	MSXE331x_public_doc.h File Reference	61
4.1.1	Typedef Documentation	67
4.1.1.1	xsd__string	67
4.1.1.2	xsd__char	67
4.1.1.3	xsd__float	67
4.1.1.4	xsd__double	67
4.1.1.5	xsd__int	67
4.1.1.6	xsd__long	67
4.1.1.7	xsd__unsignedByte	67
4.1.1.8	xsd__unsignedInt	67
4.1.1.9	xsd__unsignedShort	67
4.1.1.10	xsd__unsignedLong	67
4.1.2	Function Documentation	67
4.1.2.1	MXCommon__GetModuleType	67
4.1.2.2	MXCommon__GetHostname	67
4.1.2.3	MXCommon__SetHostname	68
4.1.2.4	MXCommon__GetClientConnections	68

4.1.2.5	MXCommon__Strerror	69
4.1.2.6	MXCommon__Reboot	70
4.1.2.7	MXCommon__ResetAllIOFunctionalities	70
4.1.2.8	MXCommon__DataseverRestart	71
4.1.2.9	MXCommon__GetEthernetLinksStates	71
4.1.2.10	MXCommon__GetModuleTemperatureValueAndStatus	72
4.1.2.11	MXCommon__SetModuleTemperatureWarningLevels	73
4.1.2.12	MXCommon__SetHardwareTriggerFilterTime	73
4.1.2.13	MXCommon__GetHardwareTriggerFilterTime	74
4.1.2.14	MXCommon__GetHardwareTriggerState	74
4.1.2.15	MXCommon__SetCustomerKey	75
4.1.2.16	MXCommon__TestCustomerID	75
4.1.2.17	MXCommon__SetTime	75
4.1.2.18	MXCommon__SysToHardwareClock	76
4.1.2.19	MXCommon__HardwareClockToSys	76
4.1.2.20	MXCommon__GetTime	77
4.1.2.21	MXCommon__GetUpTime	77
4.1.2.22	MXCommon__GetAutoConfigurationFile	77
4.1.2.23	MXCommon__SetAutoConfigurationFile	78
4.1.2.24	MXCommon__StartAutoConfiguration	78
4.1.2.25	MXCommon__InitAndStartSynchroTimer	78
4.1.2.26	MXCommon__StopAndReleaseSynchroTimer	79
4.1.2.27	MXCommon__GetConfigurationBackupFile	80
4.1.2.28	MXCommon__ApplyConfigurationBackupFile	81
4.1.2.29	MXCommon__ChangePassword	81
4.1.2.30	MXCommon__GetSubSystemState	82
4.1.2.31	MXCommon__GetSubsystemIDFromName	82
4.1.2.32	MXCommon__GetStateIDFromName	82
4.1.2.33	MXCommon__GetSubsystemNameFromID	83
4.1.2.34	MXCommon__GetStateNameFromID	83
4.1.2.35	MXCommon__GetOptionInformation	84
4.1.2.36	MXCommon__SetToMaster	84
4.1.2.37	MXCommon__GetSynchronizationStatus	84
4.1.2.38	MSXExxxx__AcquisitionGetNumberOfChannels	85
4.1.2.39	MSXExxxx__AcquisitionGetChannelInfo	85
4.1.2.40	MSXExxxx__AcquisitionAutoRefreshInitAndStart	86

4.1.2.41	MSXExxxx__AcquisitionAutoRefreshGetValues . . . . .	88
4.1.2.42	MSXExxxx__AcquisitionAutoRefreshStopAndRelease . . . . .	89
4.1.2.43	MSXExxxx__AcquisitionSequenceInitAndStart . . . . .	89
4.1.2.44	MSXExxxx__AcquisitionSequenceStopAndRelease . . . . .	92
4.1.2.45	MSXExxxx__PressureGetNumberOfChannels . . . . .	92
4.1.2.46	MSXExxxx__PressureSetChannelConfiguration . . . . .	92
4.1.2.47	MSXExxxx__PressureSetSamplingRate . . . . .	93
4.1.2.48	MSXExxxx__PressureGetConfiguration . . . . .	94



# Chapter 1

## Introduction

**MainRevision:**

### 1.1 Introduction

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

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

### 1.2 Remark: SOAP functions prototypes

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



# Chapter 2

## Module Documentation

### 2.1 MSX-E331x functions

#### Modules

- [MSX-E331x Acquisition functions](#)  
*Contain the acquisition information and initialisation functions.*
- [MSX-E331x Pressure functions](#)  
*Contain the Pressure initialisation and diagnostic functions.*

### 2.2 Common functions

#### Modules

- [Common general functions](#)  
*Various utility functions, mainly to identify a remote system.*
- [Common temperature functions](#)  
*These functions deals with the internal temperature sub-system.*
- [Common hardware trigger functions](#)  
*These functions allow to set and request the current value of the hardware trigger.*
- [Common security functions](#)  
*The "customer key" feature may for instance be used by a customer to be sure that his application communicates only with certified MSX-E modules.*
- [Common time functions](#)  
*A MSX-E module provides a "system clock" that may be in the simplest case set by the function `MXCommon__SetTime()`.*
- [Common I/O auto configuration functions](#)

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

- [Common synchronisation timer functions](#)

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

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

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

- [System state management](#)

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

- [Customer option management](#)

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

- [Synchronisation management](#)

*Manage the synchronisation state of the system.*

## 2.3 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 errnum, struct MXCommon\_\_ByteArrayResponse *Response)`

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

- `int MXCommon\_\_Reboot (void *__, struct MXCommon\_\_Response *Response)`

*Ask the MSX-E module to reboot.*



- `int MXCommon__ResetAllIOFunctionalities (xsd__unsignedLong ulOption, struct MXCommon__Response *Response)`  
*Reset the I/O functionalities of the MSX-E system.*
- `int MXCommon__DataseverRestart (xsd__unsignedLong ulAction, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)`  
*Restart the data-server service.*
- `int MXCommon__GetEthernetLinksStates (void *_ , struct MXCommon__GetEthernetLinksStatesResponse *Response)`  
*Get MSX-E Ethernet links states.*

### 2.3.1 Function Documentation

#### 2.3.1.1 `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

#### 2.3.1.2 `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

### 2.3.1.3 `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

### 2.3.1.4 `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

### 2.3.1.5 `int MXCommon__Strerror ( xsd__int errnum, struct MXCommon__ByteArrayResponse * Response )`

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

#### Parameters

- [in] *errnum* : Error number
- [out] *Response*
  - *sArray* : See the description below.
  - *sResponse.iReturnValue* : Return value
    - 0 : success
    - -1: system error (see `syserrno`).

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

STRError(3) Linux Programmer's Manual  
STRError(3)

#### NAME

`strerror`, `strerror_r` - return string describing error code

#### SYNOPSIS

```
#include <string.h>
```

```
char *strerror(int errnum);
```

```
#define _XOPEN_SOURCE 600
```

```
#include <string.h>
```

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

#### DESCRIPTION

The `strerror()` function returns a string describing the error code passed in the argument `errnum`, possibly using the `LC_MESSAGES` part of the current locale to select the appropriate language. This string must not be modified by the application, but may be modified by a subsequent call to `perror()` or `strerror()`. No library function will modify this string.

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

#### RETURN VALUE

The `strerror()` function returns the appropriate error description string, or an unknown error message if the error code is unknown.

The value of `errno` is not changed for a successful call, and is set to a non-zero value upon error.

The `strerror_r()` function returns 0 on success and -1 on failure, setting `errno`.

#### ERRORS

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

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

#### CONFORMING TO

SVID 3, POSIX, 4.3BSD, ISO/IEC 9899:1990 (C89).

`strerror_r()` with prototype as given above is specified by SUSv3, and was in use under Digital Unix and HP Unix. An incompatible function, with prototype

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

is a GNU extension used by glibc (since 2.0), and must be regarded as obsolete in view of SUSv3.

The GNU version may, but need not, use the user-supplied buffer.

If it does, the result may be truncated in case the supplied buffer is too small. The result is always NUL-terminated.

#### SEE ALSO

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

### Return values

***SOAP\_OK*** SOAP call success

***otherwise*** SOAP protocol error

### 2.3.1.6 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

### 2.3.1.7 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

### 2.3.1.8 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.

### 2.3.1.9 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.iReturn Value***

- **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

## 2.4 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.*

### 2.4.1 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.

### 2.4.2 Function Documentation

**2.4.2.1** 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_TOLOW = 1` : Temperature too low !
  - `TEMPERATURE_LOW = 2` : Temperature under the min warning value
  - `TEMPERATURE_NOMINAL = 3` : Temperature in the nominal range
  - `TEMPERATURE_HIGH = 4` : Temperature over the max warning value
  - `TEMPERATURE_TOOHIGH = 5` : Temperature too high !
- `ulInfo` : Reserved

**Return values**

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

**2.4.2.2** `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

## 2.5 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.*

### 2.5.1 Function Documentation

#### 2.5.1.1 `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

#### 2.5.1.2 `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

**2.5.1.3** `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

## 2.6 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 ).*

### 2.6.1 Detailed Description

A "customer key" consists of two strings of data stored on the certified MSX-E module, to be used by the function [MXCommon\\_\\_TestCustomerID\(\)](#) to encrypt data.

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

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

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

Detailed presentation of operations:

The user generates and stores on the module two keys (thanks to the software function : [MXCommon\\_\\_SetCustomerKey\(\)](#)). This needs only to be done once:

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

When requested (with the software function : [MXCommon\\_\\_TestCustomerID\(\)](#) ), the module generates a 16 bytes random value and do an encryption of this value using the two saved keys and the AES algorithm (Rijndael).

The user receives then two arrays of 16 bytes :

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

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

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

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

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

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

## 2.6.2 Function Documentation

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

### Parameters

- [in] *bKey* : Customer key (only writable on the module) [32 bytes containing a AES key]
- [in] *bPublicKey* : IV (Initialisation vector) for the AES cryptography [16 bytes containing a AES key]
- [out] *Response*
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

### Return values

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

**2.6.2.2** `int MXCommon__TestCustomerID ( void * _, struct MXCommon__TestCustomerIDResponse * Response )`

### Parameters

- [in] \_ : No Input
- [out] *Response*
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).
  - bValueArray : non encrypted value array [16 bytes of random data]
  - bCryptedValueArray : Encrypted value array [16 bytes of the encrypted random data]

### Return values

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

## 2.7 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).*

### 2.7.1 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.

### 2.7.2 Function Documentation

#### 2.7.2.1 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\\_\\_Sterror\(\)](#).

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

### 2.7.2.2 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.

### 2.7.2.3 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.

### 2.7.2.4 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

#### 2.7.2.5 `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

## 2.8 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*

### 2.8.1 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.

### 2.8.2 Function Documentation

#### 2.8.2.1 `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

#### 2.8.2.2 `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

### 2.8.2.3 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

## 2.9 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)*



## 2.9.1 Function Documentation

**2.9.1.1** `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

### 2.9.1.2 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

## 2.10 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.*

### 2.10.1 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"

## 2.10.2 Function Documentation

### 2.10.2.1 `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\\_\\_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;
}
*
```

### 2.10.2.2 `int MXCommon__ApplyConfigurationBackupFile ( struct xsd__base64Binary * ByteArrayInput, xsd__unsignedLong ulEOF, struct MXCommon__Response * Response )`

#### Parameters

- [in] ***ByteArrayInput*** : Array of Bytes of the file
- [in] ***ulEOF*** : End of file flag
- [out] ***Response*** • `sResponse.iReturnValue` : Return value

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

#### Return values

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

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

**2.10.2.3** `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.

## 2.11 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".

### 2.11.1 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.

### 2.11.2 Function Documentation

#### 2.11.2.1 `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

**2.11.2.2** `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

**2.11.2.3** `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

#### 2.11.2.4 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

#### 2.11.2.5 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

## 2.12 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.*

## 2.12.1 Function Documentation

**2.12.1.1** `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

## 2.13 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.*

## 2.13.1 Function Documentation

**2.13.1.1** `int MXCommon__SetToMaster ( void * __, xsd__unsignedLong ulState, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__Response * Response )`

### Parameters

- [in] *ulState* State of the supermaster mode
  - **0** automatic mode (default). The state of the system (master or slave) will be automatically detected by the system
  - **1** Set to master mode at all time. The system will always be detected as master
- [in] *ulOption01* Reserved. Set to 0
- [in] *ulOption02* Reserved. Set to 0
- [out] *Response* *iReturnValue*



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

**Return values**

**0** SOAP\_OK

*Others* See SOAP error

**2.13.1.2** `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

## 2.14 MSX-E331x Acquisition functions

Contain the acquisition information and initialisation functions.

**Modules**

- [MSX-E331x Acquisition information functions](#)

*Contain the acquisition information functions.*

- [MSX-E331x Autorefresh functions](#)

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

- [MSX-E331x Sequence functions](#)

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

## 2.15 MSX-E331x Acquisition information functions

Contain the acquisition information functions.

### Data Structures

- struct [MSXExxxx\\_\\_AcquisitionGetChannelInfoResponse](#)

### Functions

- int [MSXExxxx\\_\\_AcquisitionGetNumberOfChannels](#) (xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Return the number of acquisition channels.*

- int [MSXExxxx\\_\\_AcquisitionGetChannelInfo](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_AcquisitionGetChannelInfoResponse](#) \*Response)

*Return the selected acquisition channel type and hardware position.*

### 2.15.1 Function Documentation

**2.15.1.1** int [MSXExxxx\\_\\_AcquisitionGetNumberOfChannels](#) ( xsd\_\_unsignedLong *ulOption1*, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \* *Response* )

#### Parameters

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

[out] *Response* :

*sResponse.iReturnValue* :

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

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

*ulValue* : Number of available acquisition channels

#### Returns

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

**2.15.1.2** `int MSXExxxx__AcquisitionGetChannelInfo ( xsd__unsignedLong ulChannel,  
xsd__unsignedLong ulOption1, struct MSXExxxx__AcquisitionGetChannelInfoResponse  
* Response )`

#### Parameters

[in] *ulChannel* : Selected acquisition channel

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

[out] *Response* :

*sResponse.iReturnValue* :

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

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

*ulType* : Acquisition channel type

- 0 : Not available
- 1 : Temperature channel
- 2 : Pressure channel
- 3 : Transducer channel
- 4 : Analog input channel
- 5 : Analog input ICP channel
- 6 : Digital I/O port

*ulHwPosition* : Hardware position index (0 to 7) *ulChannelIndex* : Return the functionality channel index

#### Returns

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

## 2.16 MSX-E331x Autorefresh functions

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

### Data Structures

- struct [MSXExxxx\\_\\_AcquisitionAutoRefreshGetValuesResponse](#)

### Functions

- `int MSXExxxx__AcquisitionAutoRefreshInitAndStart (xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulAverageValue, xsd__unsignedLong ulRefreshTime, xsd__unsignedLong ulRefreshTimeUnit, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulForceStart, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, struct MSXExxxx__Response *Response)`

*Starts an autorefresh acquisition using provided configuration.*

- int `MSXExxxx__AcquisitionAutoRefreshGetValues` (xsd\_\_unsignedLong ulBlocking, struct `MSXExxxx__AcquisitionAutoRefreshGetValuesResponse` \*Response)

*Reads the values acquired in auto refresh mode.*

- int `MSXExxxx__AcquisitionAutoRefreshStopAndRelease` (void \*\_ , struct `MSXExxxx__Response` \*Response)

*Stops the current auto refresh acquisition.*

## 2.16.1 Detailed Description

The acquisition is initialised and the values of each channels are stored in memory on the Ethernet E/A module MSX-E331x.

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

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.

## 2.16.2 Function Documentation

**2.16.2.1** `int MSXExxxx_AcquisitionAutoRefreshInitAndStart ( xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulAverageValue, xsd__unsignedLong ulRefreshTime, xsd__unsignedLong ulRefreshTimeUnit, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulForceStart, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, struct MSXExxxx_Response * Response )`

### Parameters

- [in] **ulChannelMask** Mask of the channel to acquire by the auto refresh (1 bit = 1 Channel). 0 for all available acquisition channels
- [in] **ulAverageValue** Set the average value :
  - 1 : not used
  - max value : 255
- [in] **ulRefreshTimeUnit** Refresh Time Unit
  - 0 : microsecond
  - 1 : millisecond
  - 2 : second
- [in] **ulRefreshTime** Refresh Time
  - range from min 1000 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] **ulTriggerMask** Define the source of the trigger
  - 0 : trigger disabled
  - 1 : Enable Hardware Digital Input Trigger
  - 2 : Enable Synchro Trigger
  - 4 : Enable Compare Trigger (only useful if your system has incremental counter or Sine/-Cosine input)
  - 8 : Enable Index Trigger (only useful if your system has Sine/Cosine input)
- [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
  - 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)

D0 : Absolute Time stamp information

[in] ***ulDataFormat*** • 0 : no time stamp information

- 1 : time stamp information

D1 : Relative Time stamp information

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

D2 : Auto refresh counter information

- 0 : No auto refresh counter information
- 1 : Auto refresh counter information

D3 : System status information

- 0 : No system status information required
- 1 : System status information required

D4 : Data format

- 0: Digital value (see technical description)
- 1: Analog value (see technical description)

You can not select both absolute and relative time stamp simultaneously

[in] ***ulForceStart*** :

- 0 : Function return a error if any acquisition already in progress
- 1 : If any acquisition in progress then stop this

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

[in] ***ulOption2*** Reserved. Set to 0

[in] ***ulOption3*** Reserved. Set to 0

[out] ***Response*** :

***iReturnValue*** :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Any acquisition already in progress
- -3: Any selected channel not OK, call the diagnostic function for more information
- -4: Channel Mask error
- -5: Not available average value
- -6: Not available refresh time unit
- -7: The minimal refresh time is 1000 us !
- -8: The maximal refresh time is 65535 !
- -9: Trigger mask not available
- -10: Trigger mask : 2 different trigger source cannot be simultaneously be activated
- -11: Trigger mode not available
- -12: Trigger mask : 2 trigger mode cannot be simultaneously activated
- -13: Hardware trigger : front definition error
- -14: Hardware trigger count value not available
- -15: Nbr of sequence to acquire by trigger mode not available
- -16: Data format not available
- -17: Selected channels combination not available
- -100: Kernel function error

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

### Returns

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

#### 2.16.2.2 int MSXExxxx\_\_AcquisitionAutoRefreshGetValues ( xsd\_\_unsignedLong *ulBlocking*, struct MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse \* *Response* )

Reads the values acquired in auto refresh mode.

### Parameters

[in] *ulBlocking* Wait a new value or read the actual value

- 0: Get the current auto refresh values
- 1: Wait a new auto refresh value cycle

[out] *Response* Response of the system

- *iReturnValue*
  - 0: The remote function performed OK
  - -1: Means an system error occurred
  - -2: No Acquisition in progress
  - -3: 2s timeout occurred. This if you have enabled the blocking mode.
  - -4: 2s timeout occurred. This if you do not have enabled the blocking mode, and if the first value is not available.
  - -100: Internal system error occurred. See value of *syserrno*
- *syserrno* system error code (the value of the libc "errno" code)
- *ulTimeStampLow* number of microseconds since the Epoch
- *ulTimeStampHigh* number of seconds since the Epoch
- *ulCounterValue* counter value
- *dValue* Array that contains the channels values
  - *dValue[0]* Value of channel 0
  - ...
  - *dValue[15]* Value of channel 15

### Return values

0 SOAP\_OK

*Others* See SOAP error

#### 2.16.2.3 int MSXExxxx\_\_AcquisitionAutoRefreshStopAndRelease ( void \* \_\_, struct MSXExxxx\_\_Response \* *Response* )

### Parameters

[in] \_\_ Dummy parameter

[out] *Response* :

*iReturnValue* :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -100: 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 MSXExxxx\_\_AcquisitionAutoRefreshInitAndStart.

## 2.17 MSX-E331x Sequence functions

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

### Data Structures

- struct [MSXExxxx\\_\\_AcquisitionSequenceInitAndStartChannelListParam](#)

### Functions

- int [MSXExxxx\\_\\_AcquisitionSequenceInitAndStart](#) (xsd\_unsignedLong ulNbrOfChannel, struct [MSXExxxx\\_\\_AcquisitionSequenceInitAndStartChannelListParam](#) \*psChannelList, xsd\_unsignedLong ulAcquisitionTime, xsd\_unsignedLong ulAcquisitionTimeUnit, xsd\_unsignedLong ulNbrOfSequence, xsd\_unsignedLong ulNbrMaxSequenceToTransfer, xsd\_unsignedLong ulTriggerMask, xsd\_unsignedLong ulTriggerMode, xsd\_unsignedLong ulHardwareTriggerEdge, xsd\_unsignedLong ulHardwareTriggerCount, xsd\_unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd\_unsignedLong ulDataFormat, xsd\_unsignedLong ulForceStart, xsd\_unsignedLong ulOption1, xsd\_unsignedLong ulOption2, xsd\_unsignedLong ulOption3, struct [MSXExxxx\\_\\_Response](#) \*Response)

*Initialises and starts the sequence acquisition mode.*

- int [MSXExxxx\\_\\_AcquisitionSequenceStopAndRelease](#) (void \*\_ , struct [MSXExxxx\\_\\_Response](#) \*Response)

*Stop and release the sequence acquisition mode.*

### 2.17.1 Detailed Description

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

There are different sequence modes:

- Certain number of sequences / continuous

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.

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.

## 2.17.2 Function Documentation

**2.17.2.1** `int MSXExxxx_AcquisitionSequenceInitAndStart ( xsd__unsignedLong ulNbrOfChannel, struct MSXExxxx_AcquisitionSequenceInitAndStartChannelListParam * psChannelList, xsd__unsignedLong ulAcquisitionTime, xsd__unsignedLong ulAcquisitionTimeUnit, xsd__unsignedLong ulNbrOfSequence, xsd__unsignedLong ulNbrMaxSequenceToTransfer, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulForceStart, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, struct MSXExxxx_Response * Response )`

Initialises and starts the sequence acquisition mode.

### Parameters

- [in] *ulNbrOfChannel* : Nbr of channel in the sequence
- [in] *psChannelList* : List of the channel who compose the sequence.
- [in] *ulAcquisitionTime* : Acquisition Time
  - range from min 1000 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] *ulAcquisitionTimeUnit* : Acquisition Time Unit
  - 0 : us
  - 1 : ms
  - 2 : s
- [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 : (1,4096)
- [in] ***ulTriggerMask*** : Define the source of the trigger
- 0 : trigger disabled
  - 1 : Enable Hardware Digital Input Trigger
  - 2 : Enable Synchro Trigger
  - 4 : Enable Compare Trigger (only useful if your system has incremental counter or Sine/- Cosine input)
  - 8 : Enable Index Trigger (only useful if your system has Sine/Cosine input)
- [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
- 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 : Absolute Time stamp information
- 0 : no time stamp information
  - 1 : time stamp information
- D1 : Relative Time stamp information
- 0 : no time stamp information
  - 1 : time stamp information
- D2 : Sequence counter information
- 0 : No sequence counter information
  - 1 : Sequence counter information
- D3 : System status information
- 0 : No system status information required
  - 1 : System status information required
- D4 : Data format
- 0: Digital value (see technical description)
  - 1: Analog value (see technical description)

You can not select both absolute and relative time stamp simultaneously

[in] **ulForceStart** :

- 0 : Function return a error if any acquisition already in progress
- 1 : If any acquisition in progress then stop this

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

[in] **ulOption2** : Reserved. Set to 0

[in] **ulOption3** : Reserved. Set to 0

[out] **Response** :

**iReturnValue** :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Any acquisition already in progress
- -3: The nbr of channel in the sequence is null or too high
- -4: Channel index selection error
- -5: Channel already selected
- -6: Any selected channel not OK, call the diagnostic function for more information
- -7: Not available acquisition time unit
- -8: The minimal acquisition time is 1000 us !
- -9: The maximal acquisition time is 65535 !
- -10: Transfer sequence size error (1 to 4096) !
- -11: The total number of sequences is not a multiple from number of sequences to transfer
- -12: Trigger mask not available
- -13: Trigger mask : 2 different trigger source cannot be simultaneously be activated
- -14: Trigger mode not available
- -15: Trigger mask : 2 trigger mode cannot be simultaneously be activated
- -16: Hardware trigger : front definition error
- -17: Hardware trigger count value not available
- -18: Nbr of sequence to acquire by trigger mode not available
- -19: Data format not available
- -20: Selected channels combination not available
- -100: Start sequence kernel function error

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

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

#### 2.17.2.2 int MSXExxxx\_\_AcquisitionSequenceStopAndRelease ( void \* \_\_, struct MSXExxxx\_\_Response \* *Response* )

##### Parameters

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

[out] **Response** :

**iReturnValue** :

- 0: Means the remote function performed OK

- -1: Means an system error occurred
- -2: No sequence acquisition in progress
- -100: Kernel function error

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

### Returns

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

## 2.18 MSX-E331x Pressure functions

Contain the Pressure initialisation and diagnostic functions.

### Modules

- [MSX-E331x Pressure initialisation/information functions](#)  
*Contain the Pressure initialisation/information functions.*

## 2.19 MSX-E331x Pressure initialisation/information functions

Contain the Pressure initialisation/information functions.

### Data Structures

- struct [MSXExxxx\\_\\_PressureGetConfigurationResponse](#)

### Functions

- int [MSXExxxx\\_\\_PressureGetNumberOfChannels](#) (xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)  
*Return the number of pressure channels.*
- int [MSXExxxx\\_\\_PressureSetChannelConfiguration](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_double dSensorSensibility, xsd\_\_double dSensorOffset, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_Response](#) \*Response)  
*Pressure sensor configuration for the selected channel.*
- int [MSXExxxx\\_\\_PressureSetSamplingRate](#) (xsd\_\_unsignedLong ulChannelGroup, xsd\_\_unsignedLong ulBaseSamplingRate, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_Response](#) \*Response)  
*Pressure acquisition sampling rate selection.*
- int [MSXExxxx\\_\\_PressureGetConfiguration](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_PressureGetConfigurationResponse](#) \*Response)  
*Get the selected pressure channel current configuration.*

## 2.19.1 Function Documentation

### 2.19.1.1 `int MSXExxxx_PressureGetNumberOfChannels ( xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse * Response )`

#### Parameters

- [in] *ulOption1* : Reserved. Set to 0
- [out] *Response* :
- sResponse.iReturnValue* :
    - 0: Means the remote function performed OK
    - -1: Means an system error occurred
  - sResponse.syserrno* : system-error code (the value of the libc "errno" code)
  - ulValue* : Number of available pressure channels

#### Returns

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

### 2.19.1.2 `int MSXExxxx_PressureSetChannelConfiguration ( xsd__unsignedLong ulChannel, xsd__double dSensorSensibility, xsd__double dSensorOffset, xsd__unsignedLong ulOption1, struct MSXExxxx__Response * Response )`

#### Remarks

For MSXE with pressure functionality :

- Before revision 6982 the dSensorOffset parameter is given in mV.  

$$\text{Pressure value (in Unit)} = (\text{AnalogValue (mV)} - \text{dSensorOffset (mV)}) / (\text{BridgeSupply (V)} * \text{dSensorSensibility(mV / V / Unit)})$$
- Since revision 6982 the dSensorOffset parameter is given in Unit instead of mV.  

$$\text{Pressure value (in Unit)} = (\text{AnalogValue (mV)}) / (\text{BridgeSupply (V)} * \text{dSensorSensibility(mV / V / Unit)}) - \text{dSensorOffset (Unit)}$$

#### Parameters

- [in] *ulChannel* : Channel selection (0 to 15 or 255 for all channels)
- [in] *dSensorSensibility* : Sensor sensibility (mV/V/bar or mV/V/mbar or mV/V/Pa, ...). Refer to sensor documentation
- [in] *dSensorOffset* : Sensor offset in unit (bar/Newton/Pa/Psi...) that depends on the sensor Refer to sensor documentation
- [out] *Response* :
- iReturnValue* :
    - 0: Means the remote function performed OK
    - -1: Means an system error occurred
    - -2: Channel selection wrong
    - -3: Sensor sensibility selection wrong
    - -4: Acquisition in progress. Can not change the configuration

#### Returns

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

**2.19.1.3** `int MSXExxxx_PressureSetSamplingRate ( xsd__unsignedLong ulChannelGroup, xsd__unsignedLong ulBaseSamplingRate, xsd__unsignedLong ulOption1, struct MSXExxxx_Response * Response )`

#### Parameters

[in] *ulChannelGroup* : Channel group selection.

- 0 for channels 0 and 1
- 1 for channels 2 and 3
- 2 for channels 4 and 5
- 3 for channels 6 and 7
- 4 for channels 8 and 9
- 5 for channels 10 and 11
- 6 for channels 12 and 13
- 7 for channels 14 and 15
- ...
- 255 for all channels

[in] *ulBaseSamplingRate* : Sampling rate selection

- 5 for 5Hz
- 10 for 10Hz
- 20 for 20Hz
- 40 for 40Hz
- 80 for 80Hz
- 160 for 160Hz
- 320 for 320Hz
- 640 for 640Hz
- 1000 for 1000Hz
- 2000 for 2000Hz

**If only one channel is used then the real sampling rate is  $ulBaseSamplingRate / 2$**

**If all 2 channels are used then the real sampling rate is  $ulBaseSamplingRate / 3$**

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

[out] *Response* :

*iReturnValue* :

- 0 : Means the remote function performed OK
- -1 : Means an system error occurred
- -2 : Channel group selection error
- -3 : Sampling rate selection error
- -4 : Acquisition in progress. Can not change the sampling rate
- -100 : IOCTL system call error

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

#### Returns

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

**2.19.1.4** `int MSXExxxx_PressureGetConfiguration ( xsd_unsignedLong ulChannel,  
xsd_unsignedLong ulOption1, struct MSXExxxx_PressureGetConfigurationResponse  
* Response )`

#### Parameters

[in] *ulChannel* : Channel selection (0 to 15)

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

[out] *Response* :

*sResponse.iReturnValue* :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Channel selection wrong

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

*dSensorSensibility* : Sensor sensibility (mV/V/bar or mV/V/mbar or mV/V/Pa, ...). Refer to sensor documentation

*dSensorOffset* : Sensor V offset for 0 mV/V/bar, 0 mV/V/mbar, ... Refer to sensor documentation

*ulBaseSamplingRate* : Sampling rate selection

- 5 for 5Hz
- 10 for 10Hz
- 20 for 20Hz
- 40 for 40Hz
- 80 for 80Hz
- 160 for 160Hz
- 320 for 320Hz
- 640 for 640Hz
- 1000 for 1000Hz
- 2000 for 2000Hz

#### Returns

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Channel selection wrong





## 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 MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse 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\\_\\_double](#) dValue [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 MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse::sResponse**
- 3.3.1.2 **xsd\_\_unsignedLong MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse::ulTimeStampLow**
- 3.3.1.3 **xsd\_\_unsignedLong MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse::ulTimeStampHigh**
- 3.3.1.4 **xsd\_\_unsignedLong MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse::ulCounterValue**
- 3.3.1.5 **xsd\_\_double MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse::dValue[16]**

## 3.4 MSXExxxx\_\_AcquisitionGetChannelInfoResponse Struct Reference

### Data Fields

- struct [DefaultResponse sResponse](#)  
*Default return values.*
- [xsd\\_\\_unsignedLong ulType](#)  
*Acquisition channel type*  
.
- [xsd\\_\\_unsignedLong ulHwPosition](#)  
*Hardware position index (0 to 7).*
- [xsd\\_\\_unsignedLong ulChannelIndex](#)  
*Return the functionality channel index.*

### 3.4.1 Field Documentation

- 3.4.1.1 **struct DefaultResponse MSXExxxx\_\_AcquisitionGetChannelInfoResponse::sResponse**
- 3.4.1.2 **xsd\_\_unsignedLong MSXExxxx\_\_AcquisitionGetChannelInfoResponse::ulType**
  - 0 : Not available
  - 1 : Temperature channel
  - 2 : Pressure channel
  - 3 : Transducer channel
  - 4 : Analog input channel
  - 5 : Analog input ICP channel

- 6 : Digital I/O port
- 7 : Incremental counter channel

3.4.1.3 `xsd__unsignedLong MSXExxxx__AcquisitionGetChannelInfoResponse::ulHwPosition`

3.4.1.4 `xsd__unsignedLong MSXExxxx__AcquisitionGetChannelInfoResponse::ulChannelIndex`

## 3.5 MSXExxxx\_\_AcquisitionSequenceInitAndStartChannelListParam Struct Reference

### Data Fields

- `xsd__unsignedLong ulChannelList [16]`

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

### 3.5.1 Field Documentation

3.5.1.1 `xsd__unsignedLong MSXExxxx__AcquisitionSequenceInitAndStartChannelListParam::ulChannelList[16]`

## 3.6 MSXExxxx\_\_FileResponse Struct Reference

### Data Fields

- struct `DefaultResponse sResponse`

*return values.*

- struct `ByteArray sArray`

*Dynamic Array of byte.*

- `xsd__unsignedLong ulEOF`

*flag indicating end of file.*

### 3.6.1 Field Documentation

3.6.1.1 struct `DefaultResponse` `MSXExxxx__FileResponse::sResponse`

3.6.1.2 struct `ByteArray` `MSXExxxx__FileResponse::sArray`

3.6.1.3 `xsd__unsignedLong` `MSXExxxx__FileResponse::ulEOF`

## 3.7 MSXExxxx\_\_PressureGetConfigurationResponse Struct Reference

### Data Fields

- struct `DefaultResponse` `sResponse`

*Default return values.*

- `xsd__double` `dSensorSensibility`

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

- `xsd__double` `dSensorOffset`

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

- `xsd__unsignedLong` `ulBaseSamplingRate`

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

### 3.7.1 Field Documentation

3.7.1.1 struct `DefaultResponse` `MSXExxxx__PressureGetConfigurationResponse::sResponse`

3.7.1.2 `xsd__double` `MSXExxxx__PressureGetConfigurationResponse::dSensorSensibility`

3.7.1.3 `xsd__double` `MSXExxxx__PressureGetConfigurationResponse::dSensorOffset`

3.7.1.4 `xsd__unsignedLong` `MSXExxxx__PressureGetConfigurationResponse::ulBaseSamplingRate`

## 3.8 MSXExxxx\_\_Response Struct Reference

### Data Fields

- `xsd__int` `iReturnValue`

*return value of the call :*

- `xsd__int` `syserrno`

*System-error code (the value of the libc "errno" code).*

### 3.8.1 Field Documentation

#### 3.8.1.1 xsd\_\_int MSXExxxx\_\_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

#### 3.8.1.2 xsd\_\_int MSXExxxx\_\_Response::syserrno

## 3.9 MSXExxxx\_\_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 MSXExxxx\_\_unsignedLongResponse::sResponse

#### 3.9.1.2 xsd\_\_unsignedLong MSXExxxx\_\_unsignedLongResponse::ulValue

## 3.10 MSXExxxx\_\_unsignedLongTimeStampResponse 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*
- [xsd\\_\\_unsignedLong ulTimeStampLow](#)  
*the meaning of this value is defined in the related header of the function who use this type*
- [xsd\\_\\_unsignedLong ulTimeStampHigh](#)  
*the meaning of this value is defined in the related header of the function who use this type*

### 3.10.1 Field Documentation

- 3.10.1.1 struct DefaultResponse MSXExxxx\_\_unsignedLongTimeStampResponse::sResponse
- 3.10.1.2 xsd\_\_unsignedLong MSXExxxx\_\_unsignedLongTimeStampResponse::ulValue
- 3.10.1.3 xsd\_\_unsignedLong MSXExxxx\_\_unsignedLongTimeStampResponse::ulTimeStampLow
- 3.10.1.4 xsd\_\_unsignedLong MSXExxxx\_\_ -  
unsignedLongTimeStampResponse::ulTimeStampHigh

## 3.11 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.*

### 3.11.1 Field Documentation

- 3.11.1.1 struct DefaultResponse MXCommon\_\_ByteArrayResponse::sResponse
- 3.11.1.2 struct ByteArray MXCommon\_\_ByteArrayResponse::sArray

## 3.12 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.*

### 3.12.1 Field Documentation

3.12.1.1 struct `DefaultResponse` `MXCommon__FileResponse::sResponse`

3.12.1.2 struct `ByteArray` `MXCommon__FileResponse::sArray`

3.12.1.3 `xsd__unsignedLong` `MXCommon__FileResponse::ulEOF`

## 3.13 `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.*

### 3.13.1 Field Documentation

3.13.1.1 struct `DefaultResponse` `MXCommon__GetAutoConfigurationFileResponse::sResponse`

3.13.1.2 struct `ByteArray` `MXCommon__GetAutoConfigurationFileResponse::bArray`

3.13.1.3 `xsd__unsignedLong` `MXCommon__GetAutoConfigurationFileResponse::ulEOF`

## 3.14 `MXCommon__GetEthernetLinksStatesResponse` Struct Reference

### Data Fields

- struct `DefaultResponse` `sResponse`

*Default return values.*

- struct `sGetEthernetLinksStatesPort` `sPort0`

- struct `sGetEthernetLinksStatesPort` `sPort1`



### 3.14.1 Field Documentation

3.14.1.1 struct DefaultResponse MXCommon\_\_GetEthernetLinksStatesResponse::sResponse

3.14.1.2 struct sGetEthernetLinksStatesPort MXCommon\_\_ -  
GetEthernetLinksStatesResponse::sPort0

3.14.1.3 struct sGetEthernetLinksStatesPort MXCommon\_\_ -  
GetEthernetLinksStatesResponse::sPort1

## 3.15 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.*

### 3.15.1 Field Documentation

3.15.1.1 struct DefaultResponse MXCommon\_\_ -  
GetHardwareTriggerFilterTimeResponse::sResponse

3.15.1.2 [xsd\\_\\_unsignedLong](#) MXCommon\_\_ -  
GetHardwareTriggerFilterTimeResponse::ulFilterTime

3.15.1.3 [xsd\\_\\_unsignedLong](#) MXCommon\_\_GetHardwareTriggerFilterTimeResponse::ulInfo01

3.15.1.4 [xsd\\_\\_unsignedLong](#) MXCommon\_\_GetHardwareTriggerFilterTimeResponse::ulInfo02

## 3.16 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.*

### 3.16.1 Field Documentation

3.16.1.1 struct [DefaultResponse](#) [MXCommon\\_\\_GetHardwareTriggerStateResponse::sResponse](#)

3.16.1.2 [xsd\\_\\_unsignedLong](#) [MXCommon\\_\\_GetHardwareTriggerStateResponse::ulState](#)

3.16.1.3 [xsd\\_\\_unsignedLong](#) [MXCommon\\_\\_GetHardwareTriggerStateResponse::ulInfo01](#)

3.16.1.4 [xsd\\_\\_unsignedLong](#) [MXCommon\\_\\_GetHardwareTriggerStateResponse::ulInfo02](#)

## 3.17 [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.*

### 3.17.1 Field Documentation

- 3.17.1.1 struct `DefaultResponse` `MXCommon__GetModuleTemperatureValueAndStatusResponse::sResponse`
- 3.17.1.2 `xsd__double` `MXCommon__GetModuleTemperatureValueAndStatusResponse::dTemperatureValue`
- 3.17.1.3 `xsd__unsignedLong` `MXCommon__GetModuleTemperatureValueAndStatusResponse::ulTemperatureStatus`
- 3.17.1.4 `xsd__unsignedLong` `MXCommon__GetModuleTemperatureValueAndStatusResponse::ulInfo`

## 3.18 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).*

### 3.18.1 Field Documentation

- 3.18.1.1 struct `DefaultResponse` `MXCommon__GetTimeResponse::sResponse`
- 3.18.1.2 `xsd__unsignedLong` `MXCommon__GetTimeResponse::ulLowTime`
- 3.18.1.3 `xsd__unsignedLong` `MXCommon__GetTimeResponse::ulHighTime`

## 3.19 MXCommon\_\_GetUpTimeResponse Struct Reference

### Data Fields

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

### 3.19.1 Field Documentation

3.19.1.1 struct `DefaultResponse` `MXCommon__GetUpTimeResponse::sResponse`

3.19.1.2 `xsd__unsignedLong` `MXCommon__GetUpTimeResponse::ulUpTime`

## 3.20 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()`).*

### 3.20.1 Field Documentation

3.20.1.1 `xsd__int` `MXCommon__Response::iReturnValue`

3.20.1.2 `xsd__int` `MXCommon__Response::syserrno`

## 3.21 MXCommon\_\_TestCustomerIDResponse Struct Reference

### Data Fields

- struct `DefaultResponse` `sResponse`

*Default return values.*

- struct `ByteArray` `bValueArray`

*non encrypted value*

- struct `ByteArray` `bCryptedValueArray`

*encrypted value*

### 3.21.1 Field Documentation

3.21.1.1 struct DefaultResponse MXCommon\_\_TestCustomerIDResponse::sResponse

3.21.1.2 struct ByteArray MXCommon\_\_TestCustomerIDResponse::bValueArray

3.21.1.3 struct ByteArray MXCommon\_\_TestCustomerIDResponse::bCryptedValueArray

## 3.22 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.*

### 3.22.1 Field Documentation

3.22.1.1 struct DefaultResponse MXCommon\_\_unsignedLongResponse::sResponse

3.22.1.2 [xsd\\_\\_unsignedLong](#) MXCommon\_\_unsignedLongResponse::ulValue

## 3.23 sGetEthernetLinksStatesPort Struct Reference

### Data Fields

- [xsd\\_\\_unsignedLong](#) ulState
- [xsd\\_\\_unsignedLong](#) ulSpeed
- [xsd\\_\\_unsignedLong](#) ulDuplex
- [xsd\\_\\_unsignedLong](#) ulInfo1
- [xsd\\_\\_unsignedLong](#) ulInfo2

### 3.23.1 Field Documentation

3.23.1.1 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulState`

3.23.1.2 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulSpeed`

3.23.1.3 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulDuplex`

3.23.1.4 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo1`

3.23.1.5 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo2`

## 3.24 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*

### 3.24.1 Field Documentation

3.24.1.1 `xsd__unsignedLong* UnsignedLongArray::__ptr`

3.24.1.2 `int UnsignedLongArray::__size`

3.24.1.3 `int UnsignedLongArray::__offset`

## 3.25 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*

### 3.25.1 Field Documentation

3.25.1.1 `xsd__unsignedShort* UnsignedShortArray::__ptr`

3.25.1.2 `int UnsignedShortArray::__size`

3.25.1.3 `int UnsignedShortArray::__offset`

## 3.26 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*

### 3.26.1 Field Documentation

3.26.1.1 `unsigned char* xsd__base64Binary::__ptr`

3.26.1.2 `int xsd__base64Binary::__size`





# Chapter 4

## File Documentation

### 4.1 MSXE331x\_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 [MSXExxxx\\_\\_Response](#)
- struct [MSXExxxx\\_\\_unsignedLongResponse](#)
- struct [MSXExxxx\\_\\_FileResponse](#)
- struct [MSXExxxx\\_\\_unsignedLongTimeStampResponse](#)
- struct [MSXExxxx\\_\\_AcquisitionGetChannelInfoResponse](#)
- struct [MSXExxxx\\_\\_AcquisitionAutoRefreshGetValuesResponse](#)
- struct [MSXExxxx\\_\\_AcquisitionSequenceInitAndStartChannelListParam](#)
- struct [MSXExxxx\\_\\_PressureGetConfigurationResponse](#)

## 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__Sterror (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 [MSXExxxx\\_\\_AcquisitionGetNumberOfChannels](#) ([xsd\\_\\_unsignedLong](#) ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Return the number of acquisition channels.*

- int [MSXExxxx\\_\\_AcquisitionGetChannelInfo](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_AcquisitionGetChannelInfoResponse](#) \*Response)

*Return the selected acquisition channel type and hardware position.*

- int [MSXExxxx\\_\\_AcquisitionAutoRefreshInitAndStart](#) (xsd\_\_unsignedLong ulChannelMask, xsd\_\_unsignedLong ulAverageValue, xsd\_\_unsignedLong ulRefreshTime, xsd\_\_unsignedLong ulRefreshTimeUnit, xsd\_\_unsignedLong ulTriggerMask, xsd\_\_unsignedLong ulTriggerMode, xsd\_\_unsignedLong ulHardwareTriggerEdge, xsd\_\_unsignedLong ulHardwareTriggerCount, xsd\_\_unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd\_\_unsignedLong ulDataFormat, xsd\_\_unsignedLong ulForceStart, xsd\_\_unsignedLong ulOption1, xsd\_\_unsignedLong ulOption2, xsd\_\_unsignedLong ulOption3, struct [MSXExxxx\\_\\_Response](#) \*Response)

*Starts an autorefresh acquisition using provided configuration.*

- int [MSXExxxx\\_\\_AcquisitionAutoRefreshGetValues](#) (xsd\_\_unsignedLong ulBlocking, struct [MSXExxxx\\_\\_AcquisitionAutoRefreshGetValuesResponse](#) \*Response)

*Reads the values acquired in auto refresh mode.*

- int [MSXExxxx\\_\\_AcquisitionAutoRefreshStopAndRelease](#) (void \*\_ , struct [MSXExxxx\\_\\_Response](#) \*Response)

*Stops the current auto refresh acquisition.*

- int [MSXExxxx\\_\\_AcquisitionSequenceInitAndStart](#) (xsd\_\_unsignedLong ulNbrOfChannel, struct [MSXExxxx\\_\\_AcquisitionSequenceInitAndStartChannelListParam](#) \*psChannelList, xsd\_\_unsignedLong ulAcquisitionTime, xsd\_\_unsignedLong ulAcquisitionTimeUnit, xsd\_\_unsignedLong ulNbrOfSequence, xsd\_\_unsignedLong ulNbrMaxSequenceToTransfer, xsd\_\_unsignedLong ulTriggerMask, xsd\_\_unsignedLong ulTriggerMode, xsd\_\_unsignedLong ulHardwareTriggerEdge, xsd\_\_unsignedLong ulHardwareTriggerCount, xsd\_\_unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd\_\_unsignedLong ulDataFormat, xsd\_\_unsignedLong ulForceStart, xsd\_\_unsignedLong ulOption1, xsd\_\_unsignedLong ulOption2, xsd\_\_unsignedLong ulOption3, struct [MSXExxxx\\_\\_Response](#) \*Response)

*Initialises and starts the sequence acquisition mode.*

- int [MSXExxxx\\_\\_AcquisitionSequenceStopAndRelease](#) (void \*\_ , struct [MSXExxxx\\_\\_Response](#) \*Response)

*Stop and release the sequence acquisition mode.*

- int [MSXExxxx\\_\\_PressureGetNumberOfChannels](#) (xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Return the number of pressure channels.*

- int [MSXExxxx\\_\\_PressureSetChannelConfiguration](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_double dSensorSensitivity, xsd\_\_double dSensorOffset, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_Response](#) \*Response)

*Pressure sensor configuration for the selected channel.*

- int [MSXExxxx\\_\\_PressureSetSamplingRate](#) (xsd\_\_unsignedLong ulChannelGroup, xsd\_\_unsignedLong ulBaseSamplingRate, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_Response](#) \*Response)

*Pressure acquisition sampling rate selection.*

- int [MSXExxx\\_PressureGetConfiguration](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulOption1, struct [MSXExxx\\_PressureGetConfigurationResponse](#) \*Response)

*Get the selected pressure channel current configuration.*

### 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)  
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

#### 4.1.2.6 int MXCommon\_\_Reboot ( void \* \_, struct MXCommon\_\_Response \* *Response* )

### Parameters

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

### Return values

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

#### 4.1.2.7 int MXCommon\_\_ResetAllIOFunctionalities ( xsd\_\_unsignedLong *ulOption*, struct MXCommon\_\_Response \* *Response* )

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

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

### Parameters

[in] **ulOption** Reserved. Set to 0  
 [out] **Response** **iReturnValue**  
     • **0** The remote function performed OK  
     • **-1** Internal system error occurred. See value of syserrno  
     • **-100** Function not supported by the system  
     **syserrno** system error code (the value of the libc "errno" code)

### Return values

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

#### 4.1.2.8 int MXCommon\_\_DataserverRestart ( xsd\_\_unsignedLong ulAction, xsd\_\_unsignedLong ulOption, struct MXCommon\_\_Response \* Response )

##### Parameters

- [in] **ulAction** : action
- 0: normal restart
  - 1: with cache file reset
  - 2: with cache file deletion
- [in] **ulOption** : Reserved
- [out] **Response**     • iReturnValue : Return value
- 0 : success
  - -1: system error (see syserrno)
  - syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

##### Note

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

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

##### Parameters

- [in] **\_** : no input parameter
- [out] **Response** Structure that contains the MSX-E Ethernet links states and errors:
- sResponse.iReturnValue**
- **0** The remote function performed OK
  - **-1** System error occurred
  - **-2** Fail to get Ethernet links states
  - **-100** Internal system error occurred. See value of syserrno
- sResponse.syserrno** system error code (the value of the libc "errno" code)
- sPort0: Fisrt port informations**
- **ulState**
    - **0** Link down
    - **1** Link up
  - **ulSpeed**
    - **10** 10 Mb/s
    - **100** 100 Mb/s
  - **ulDuplex**
    - **0** Half duplex
    - **1** Full duplex

- **ulInfo1** Reserved
- **ulInfo2** Reserved

*sPort1: Second port informations*

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

**Return values**

**0** SOAP\_OK

*Others* See SOAP error

**4.1.2.10 int MXCommon\_\_GetModuleTemperatureValueAndStatus ( xsd\_\_unsignedLong ulOption, struct MXCommon\_\_GetModuleTemperatureValueAndStatusResponse \* Response )**

**Parameters**

[in] *ulOption* : Reserved

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

- **0** : success
- **-1** : system error (see syserrno)
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Sterror\(\)](#).
  - dValue : Temperature value in Degree Celsius
- ulTemperatureStatus : Temperature Status :
  - TEMPERATURE\_INITIAL = 0 : Temperature not ready
  - TEMPERATURE\_TOLOW = 1 : Temperature too low !
  - TEMPERATURE\_LOW = 2 : Temperature under the min warning value
  - TEMPERATURE\_NOMINAL = 3 : Temperature in the nominal range
  - TEMPERATURE\_HIGH = 4 : Temperature over the max warning value
  - TEMPERATURE\_TOOHIGH = 5 : Temperature too high !

- ulInfo : Reserved

**Return values**

**SOAP\_OK** SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

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

#### Return values

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

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

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

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

#### Parameters

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

#### Return values

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

#### 4.1.2.13 int MXCommon\_\_GetHardwareTriggerFilterTime ( xsd\_\_unsignedLong *ulOption*, struct MXCommon\_\_GetHardwareTriggerFilterTimeResponse \* *Response* )

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

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

##### Parameters

[in] *ulOption* Reserved. Set to 0

[out] *Response* Response of the system

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

##### Return values

0 SOAP\_OK

*Others* See SOAP error

#### 4.1.2.14 int MXCommon\_\_GetHardwareTriggerState ( xsd\_\_unsignedLong *ulOption*, struct MXCommon\_\_GetHardwareTriggerStateResponse \* *Response* )

##### Parameters

[in] *ulOption* : Reserved

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

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

##### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

- [in] *bKey* : Customer key (only writable on the module) [32 bytes containing a AES key]
- [in] *bPublicKey* : IV (Initialisation vector) for the AES cryptography [16 bytes containing a AES key]
- [out] *Response*
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

#### Return values

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

**4.1.2.16** `int MXCommon__TestCustomerID ( void * _, struct MXCommon__TestCustomerIDResponse * Response )`

#### Parameters

- [in] *\_* : No Input
- [out] *Response*
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).
  - bValueArray : non encrypted value array [16 bytes of random data]
  - bCryptedValueArray : Encrypted value array [16 bytes of the encrypted random data]

#### Return values

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

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

#### Parameters

- [in] *ulLowTime* : Number of microseconds since the begin of the second
- [in] *ulHighTime* : Number of seconds since the Epoch (1st January,1970)
- [out] *Response*
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

**Return values**

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

#### 4.1.2.18 int MXCommon\_\_SysToHardwareClock ( void \* \_, struct MXCommon\_\_Response \* Response )

**Parameters**

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

**Return values**

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

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

#### 4.1.2.19 int MXCommon\_\_HardwareClockToSys ( void \* \_, struct MXCommon\_\_Response \* Response )

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

**Parameters**

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

**Return values**

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

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



#### 4.1.2.20 int MXCommon\_\_GetTime ( void \* \_\_, struct MXCommon\_\_GetTimeResponse \* Response )

##### Parameters

- [in] \_\_ : No input parameter
- [out] **Response** • sResponse.iReturnValue : Return value
- 0 : success
  - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).
  - ulLowTime : Number of microseconds since the begin of the second
  - ulHighTime : Number of seconds since the Epoch (1st January,1970)

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

#### 4.1.2.21 int MXCommon\_\_GetUpTime ( void \* \_\_, struct MXCommon\_\_GetUpTimeResponse \* Response )

##### Parameters

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

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

#### 4.1.2.22 int MXCommon\_\_GetAutoConfigurationFile ( void \* \_\_, struct MXCommon\_\_GetAutoConfigurationFileResponse \* Response )

##### Parameters

- [in] \_\_ : No input parameter
- [out] **Response** • sResponse.iReturnValue : Return value
- 0 : success
  - -1: system error (see syserrno)
  - -100 : Error of the read of the auto configuration file
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).
  - bArray : Array of Bytes of the file

- *ulEOF* : End of file flag

#### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

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

[in] *ulEOF* : End of file flag

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

– 0 : success

– -1: system error (see syserrno)

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

#### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

**4.1.2.24** `int MXCommon__StartAutoConfiguration ( void * _, struct MXCommon__ByteArrayResponse * Response )`

#### Parameters

[in] *\_* : No input parameter

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

– 0 : success

– -1: system error (see syserrno)

- sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Sterror\(\)](#).
- sArray : message returned by the auto configuration start

#### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

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

- [in] ***ulReloadValue*** : Timer reload value (0 to 0xFFFF), minimum reload time is 5 us
- [in] ***ulNbrOfCycle*** : Number of timer cycle
  - 0: continuous
  - > 0: defined number of cycle
- [in] ***ulGenerateTriggerMode*** :
  - 0: Wait the time overflow to set the synchronisation trigger
  - 1: Set the synchronisation trigger by the start of the timer and after each time overflow
- [in] ***ulOption01*** : Define the source of the trigger
  - 0 : Trigger disabled
  - 1 : Enable the hardware digital input trigger
- [in] ***ulOption02*** : Define the edge of the hardware trigger who generates a trigger action
  - 1 : rising edge (Only if hardware trigger selected)
  - 2 : falling edge (Only if hardware trigger selected)
  - 3 : Both front (Only if hardware trigger selected)
- [in] ***ulOption03*** : Define the number of trigger events before the action occur
  - 1 : all trigger event start the action
  - max value : 65535
- [in] ***ulOption04*** : Reserved
- [out] ***Response***
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
    - -2: not available time base
    - -3: timer reload value can not be greater than 65535
    - -4: minimum time reload is 5 us
    - -5: Number of cycle can not be greater than 65535
    - -6: Generate trigger mode error
    - -100: Init timer error
    - -101: Start timer error
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#). May be ENOSYS : Function not implemented.

**Return values*****SOAP\_OK*** SOAP call success***otherwise*** SOAP protocol error
**4.1.2.26 int MXCommon\_\_StopAndReleaseSynchroTimer ( xsd\_\_unsignedLong ulOption01, struct MXCommon\_\_Response \* Response )**
**Parameters**

- [in] ***ulOption01*** : Reserved
- [out] ***Response***
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
    - -100: Start/Stop timer error

- `sResponse.syserrno` : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Sterror\(\)](#). May be `ENOSYS` : Function not implemented.

### Return values

***SOAP\_OK*** SOAP call success

***otherwise*** SOAP protocol error

#### 4.1.2.27 `int MXCommon__GetConfigurationBackupFile ( void * _, struct MXCommon__FileResponse * Response )`

### Parameters

- [in] `_` : No input parameter
- [out] ***Response*** • `sResponse.iReturnValue` : Return value
- 0 : success
  - -1: system error (see `syserrno`) (see `syserrno`)
  - `sResponse.syserrno` : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Sterror\(\)](#).
  - `bArray` : Array of Bytes of the file
  - `ulEOF` : End of file flag

### Return values

***SOAP\_OK*** SOAP call success

***otherwise*** SOAP protocol error

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

Below is an example in pseudo-C.

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

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

#### Parameters

- [in] *ByteArrayInput* : Array of Bytes of the file
- [in] *ulEOF* : End of file flag
- [out] *Response*
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see syserrno)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

#### Return values

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

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

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

The changes are immediately active.

#### Parameters

- [in] *\_* : No input parameter
- [out] *Response*
  - sResponse.iReturnValue : Return value
    - 0 : success
    - -1: string PreviousUser is invalid
    - -2: string PreviousPassword is invalid
    - -3: string NewUser is invalid
    - -4: string NewPassword is invalid
    - -5: authentication failed
    - -100: system error while saving tokens (use syserrno for more information)
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).
  - sArray : message returned by the auto configuration start

#### Return values

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

#### Warning

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

#### 4.1.2.30 int MXCommon\_\_GetSubSystemState ( xsd\_\_unsignedLong SubsystemID, struct MXCommon\_\_unsignedLongResponse \* Response )

##### Parameters

- [in] *SubsystemID* sub-system numerical ID
- [out] *Response* • sResponse.iReturnValue : Return value
- 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameter SubsystemID
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).
  - Value The state of the sub-system "Id" at the moment of the execution of the request.

##### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

#### 4.1.2.31 int MXCommon\_\_GetSubsystemIDFromName ( struct xsd\_\_base64Binary \* SubsystemName, struct MXCommon\_\_unsignedLongResponse \* Response )

##### Parameters

- [in] *SubsystemName* sub-system symbolic name.
- [out] *Response* • sResponse.iReturnValue :Return value
- 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameter SubsystemName
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).
  - Value The numerical ID of the sub-system "SubsystemName".

##### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

#### 4.1.2.32 int MXCommon\_\_GetStateIDFromName ( xsd\_\_unsignedLong SubsystemID, struct xsd\_\_base64Binary \* StateName, struct MXCommon\_\_unsignedLongResponse \* Response )

##### Parameters

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

- sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Sterror\(\)](#).
- Value The numerical ID of the state "StateName".

**Return values**

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

#### 4.1.2.33 int MXCommon\_\_GetSubsystemNameFromID ( xsd\_\_unsignedLong SubsystemID, struct MXCommon\_\_ByteArrayResponse \* Response )

**Parameters**

- [in] *SubsystemID* sub-system numerical ID.
- [out] *Response* • sResponse.iReturnValue : Return value
- 0 : success
  - -1: system error while executing the request (see syserrno)
  - -2: invalid parameter SubsystemName
- sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Sterror\(\)](#).
  - sArray : The symbolic name associated with the ID.

**Return values**

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

#### 4.1.2.34 int MXCommon\_\_GetStateNameFromID ( xsd\_\_unsignedLong SubsystemID, xsd\_\_unsignedLong StateID, struct MXCommon\_\_ByteArrayResponse \* Response )

**Parameters**

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

**Return values**

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

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

#### Parameters

- [in] *ulOption01*,: not used, set it to 0
- [in] *ulOption02*,: not used, set it to 0
- [out] *Response*
  - sArray : Option information string
  - sResponse Composed of iReturnValue and syserrno

#### Return values

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

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

#### Parameters

- [in] *ulState* State of the supermaster mode
  - **0** automatic mode (default). The state of the system (master or slave) will be automatically detected by the system
  - **1** Set to master mode at all time. The system will always be detected as master
- [in] *ulOption01* Reserved. Set to 0
- [in] *ulOption02* Reserved. Set to 0
- [out] *Response iReturnValue*
  - **0** The remote function performed OK
  - **-1** System error occurred
  - **-2** The PLD is not working
  - **-3** The ulFilterTime parameter is wrong
  - **-100** Internal system error occurred. See value of syserrno *syserrno* system error code (the value of the libc "errno" code)

#### Return values

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

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

#### Parameters

- [in] *ulOption01* Reserved. Set to 0
- [in] *ulOption02* Reserved. Set to 0
- [out] *Response sResponse.iReturnValue*



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

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

*ulValue* State of the supermaster mode

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

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

**4.1.2.38** `int MSXExxxx__AcquisitionGetNumberOfChannels ( xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse * Response )`

#### Parameters

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

[out] *Response* :

*sResponse.iReturnValue* :

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

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

*ulValue* : Number of available acquisition channels

#### Returns

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

**4.1.2.39** `int MSXExxxx__AcquisitionGetChannelInfo ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MSXExxxx__AcquisitionGetChannelInfoResponse * Response )`

#### Parameters

[in] *ulChannel* : Selected acquisition channel

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

[out] *Response* :

*sResponse.iReturnValue* :

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

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

***ulType*** : Acquisition channel type

- 0 : Not available
- 1 : Temperature channel
- 2 : Pressure channel
- 3 : Transducer channel
- 4 : Analog input channel
- 5 : Analog input ICP channel
- 6 : Digital I/O port

***ulHwPosition*** : Hardware position index (0 to 7) ***ulChannelIndex*** : Return the functionality channel index

### Returns

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

**4.1.2.40** `int MSXExxxx__AcquisitionAutoRefreshInitAndStart ( xsd__unsignedLong ulChannelMask, xsd__unsignedLong ulAverageValue, xsd__unsignedLong ulRefreshTime, xsd__unsignedLong ulRefreshTimeUnit, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerMode, xsd__unsignedLong ulHardwareTriggerEdge, xsd__unsignedLong ulHardwareTriggerCount, xsd__unsignedLong ulByTriggerNbrOfSeqToAcquire, xsd__unsignedLong ulDataFormat, xsd__unsignedLong ulForceStart, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, xsd__unsignedLong ulOption3, struct MSXExxxx__Response * Response )`

### Parameters

[in] ***ulChannelMask*** Mask of the channel to acquire by the auto refresh (1 bit = 1 Channel). 0 for all available acquisition channels

[in] ***ulAverageValue*** Set the average value :

- 1 : not used
- max value : 255

[in] ***ulRefreshTimeUnit*** Refresh Time Unit

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

[in] ***ulRefreshTime*** Refresh Time

- range from min 1000 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] ***ulTriggerMask*** Define the source of the trigger

- 0 : trigger disabled
- 1 : Enable Hardware Digital Input Trigger
- 2 : Enable Synchro Trigger
- 4 : Enable Compare Trigger (only useful if your system has incremental counter or Sine/-Cosine input)

- 8 : Enable Index Trigger (only useful if your system has Sine/Cosine input)
- [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
- 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)
- D0 : Absolute Time stamp information
- [in] ***ulDataFormat*** • 0 : no time stamp information
- 1 : time stamp information
- D1 : Relative Time stamp information
- 0 : no time stamp information
  - 1 : time stamp information
- D2 : Auto refresh counter information
- 0 : No auto refresh counter information
  - 1 : Auto refresh counter information
- D3 : System status information
- 0 : No system status information required
  - 1 : System status information required
- D4 : Data format
- 0: Digital value (see technical description)
  - 1: Analog value (see technical description)
- You can not select both absolute and relative time stamp simultaneously
- [in] ***ulForceStart*** :
- 0 : Function return a error if any acquisition already in progress
  - 1 : If any acquisition in progress then stop this
- [in] ***ulOption1*** Reserved. Set to 0
- [in] ***ulOption2*** Reserved. Set to 0
- [in] ***ulOption3*** Reserved. Set to 0
- [out] ***Response*** :
- iReturnValue*** :
- 0: Means the remote function performed OK
  - -1: Means an system error occurred
  - -2: Any acquisition already in progress
  - -3: Any selected channel not OK, call the diagnostic function for more information

- -4: Channel Mask error
- -5: Not available average value
- -6: Not available refresh time unit
- -7: The minimal refresh time is 1000 us !
- -8: The maximal refresh time is 65535 !
- -9: Trigger mask not available
- -10: Trigger mask : 2 different trigger source cannot be simultaneously be activated
- -11: Trigger mode not available
- -12: Trigger mask : 2 trigger mode cannot be simultaneously activated
- -13: Hardware trigger : front definition error
- -14: Hardware trigger count value not available
- -15: Nbr of sequence to acquire by trigger mode not available
- -16: Data format not available
- -17: Selected channels combination not available
- -100: Kernel function error

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

#### Returns

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

#### 4.1.2.41 int MSXExxxx\_\_AcquisitionAutoRefreshGetValues ( xsd\_\_unsignedLong ulBlocking, struct MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse \* Response )

Reads the values acquired in auto refresh mode.

#### Parameters

[in] **ulBlocking** Wait a new value or read the actual value

- **0**: Get the current auto refresh values
- **1**: Wait a new auto refresh value cycle

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

- **iReturnValue**
  - **0**: The remote function performed OK
  - **-1**: Means an system error occurred
  - **-2**: No Acquisition in progress
  - **-3**: 2s timeout occurred. This if you have enabled the blocking mode.
  - **-4**: 2s timeout occurred. This if you do not have enabled the blocking mode, and if the first value is not available.
  - **-100**: Internal system error occurred. See value of syserrno
- **syserrno** system error code (the value of the libc "errno" code)
- **ulTimeStampLow** number of microseconds since the Epoch
- **ulTimeStampHigh** number of seconds since the Epoch
- **ulCounterValue** counter value
- **dValue** Array that contains the channels values
  - **dValue[0]** Value of channel 0

- ...
- *dValue[15]* Value of channel 15

**Return values**

0 SOAP\_OK

*Others* See SOAP error

#### 4.1.2.42 int MSXExxxx\_\_AcquisitionAutoRefreshStopAndRelease ( void \* \_, struct MSXExxxx\_\_Response \* *Response* )

**Parameters**

[in] \_ Dummy parameter

[out] *Response* :

*iReturnValue* :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -100: 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 MSXExxxx\_\_AcquisitionAutoRefreshInitAndStart.

#### 4.1.2.43 int MSXExxxx\_\_AcquisitionSequenceInitAndStart ( xsd\_\_unsignedLong *ulNbrOfChannel*, struct MSXExxxx\_\_AcquisitionSequenceInitAndStartChannelListParam \* *psChannelList*, xsd\_\_unsignedLong *ulAcquisitionTime*, xsd\_\_unsignedLong *ulAcquisitionTimeUnit*, xsd\_\_unsignedLong *ulNbrOfSequence*, xsd\_\_unsignedLong *ulNbrMaxSequenceToTransfer*, xsd\_\_unsignedLong *ulTriggerMask*, xsd\_\_unsignedLong *ulTriggerMode*, xsd\_\_unsignedLong *ulHardwareTriggerEdge*, xsd\_\_unsignedLong *ulHardwareTriggerCount*, xsd\_\_unsignedLong *ulByTriggerNbrOfSeqToAcquire*, xsd\_\_unsignedLong *ulDataFormat*, xsd\_\_unsignedLong *ulForceStart*, xsd\_\_unsignedLong *ulOption1*, xsd\_\_unsignedLong *ulOption2*, xsd\_\_unsignedLong *ulOption3*, struct MSXExxxx\_\_Response \* *Response* )

Initialises and starts the sequence acquisition mode.

**Parameters**

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

[in] *psChannelList* : List of the channel who compose the sequence.

[in] *ulAcquisitionTime* : Acquisition Time

- range from min 1000 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] ***ulAcquisitionTimeUnit*** : Acquisition Time Unit
- 0 : us
  - 1 : ms
  - 2 : s
- [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 : (1,4096)
- [in] ***ulTriggerMask*** : Define the source of the trigger
- 0 : trigger disabled
  - 1 : Enable Hardware Digital Input Trigger
  - 2 : Enable Synchro Trigger
  - 4 : Enable Compare Trigger (only useful if your system has incremental counter or Sine/-Cosine input)
  - 8 : Enable Index Trigger (only useful if your system has Sine/Cosine input)
- [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
- 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 : Absolute Time stamp information
- 0 : no time stamp information
  - 1 : time stamp information
- D1 : Relative Time stamp information
- 0 : no time stamp information
  - 1 : time stamp information
- D2 : Sequence counter information
- 0 : No sequence counter information
  - 1 : Sequence counter information
- D3 : System status information
- 0 : No system status information required

- 1 : System status information required

D4 : Data format

- 0: Digital value (see technical description)
- 1: Analog value (see technical description)

You can not select both absolute and relative time stamp simultaneously

[in] **ulForceStart** :

- 0 : Function return a error if any acquisition already in progress
- 1 : If any acquisition in progress then stop this

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

[in] **ulOption2** : Reserved. Set to 0

[in] **ulOption3** : Reserved. Set to 0

[out] **Response** :

**iReturnValue** :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Any acquisition already in progress
- -3: The nbr of channel in the sequence is null or too high
- -4: Channel index selection error
- -5: Channel already selected
- -6: Any selected channel not OK, call the diagnostic function for more information
- -7: Not available acquisition time unit
- -8: The minimal acquisition time is 1000 us !
- -9: The maximal acquisition time is 65535 !
- -10: Transfer sequence size error (1 to 4096) !
- -11: The total number of sequences is not a multiple from number of sequences to transfer
- -12: Trigger mask not available
- -13: Trigger mask : 2 different trigger source cannot be simultaneously be activated
- -14: Trigger mode not available
- -15: Trigger mask : 2 trigger mode cannot be simultaneously be activated
- -16: Hardware trigger : front definition error
- -17: Hardware trigger count value not available
- -18: Nbr of sequence to acquire by trigger mode not available
- -19: Data format not available
- -20: Selected channels combination not available
- -100: Start sequence kernel function error

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

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

#### 4.1.2.44 int MSXExxxx\_\_AcquisitionSequenceStopAndRelease ( void \* \_\_, struct MSXExxxx\_\_Response \* *Response* )

##### Parameters

[in] \_\_ : no input parameter

[out] *Response* :

*iReturnValue* :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: No sequence acquisition in progress
- -100: 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 MSXExxxx\_\_PressureGetNumberOfChannels ( xsd\_\_unsignedLong *ulOption1*, struct MSXExxxx\_\_unsignedLongResponse \* *Response* )

##### Parameters

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

[out] *Response* :

*sResponse.iReturnValue* :

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

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

*ulValue* : Number of available pressure channels

##### Returns

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

#### 4.1.2.46 int MSXExxxx\_\_PressureSetChannelConfiguration ( xsd\_\_unsignedLong *ulChannel*, xsd\_\_double *dSensorSensibility*, xsd\_\_double *dSensorOffset*, xsd\_\_unsignedLong *ulOption1*, struct MSXExxxx\_\_Response \* *Response* )

##### Remarks

For MSXE with pressure functionality :

- Before revision 6982 the *dSensorOffset* parameter is given in mV.  

$$\text{Pressure value (in Unit)} = (\text{AnalogValue (mV)} - \text{dSensorOffset (mV)}) / (\text{BridgeSupply (V)} * \text{dSensorSensibility(mV / V / Unit)})$$
- Since revision 6982 the *dSensorOffset* parameter is given in Unit instead of mV.  

$$\text{Pressure value (in Unit)} = (\text{AnalogValue (mV)}) / (\text{BridgeSupply (V)} * \text{dSensorSensibility(mV / V / Unit)}) - \text{dSensorOffset (Unit)}$$



**Parameters**

- [in] ***ulChannel*** : Channel selection (0 to 15 or 255 for all channels)
- [in] ***dSensorSensibility*** : Sensor sensibility (mV/V/bar or mV/V/mbar or mV/V/Pa, ...). Refer to sensor documentation
- [in] ***dSensorOffset*** : Sensor offset in unit (bar/Newton/Pa/Psi...) that depends on the sensor Refer to sensor documentation
- [out] ***Response*** :
  - iReturnValue*** :
    - 0: Means the remote function performed OK
    - -1: Means an system error occurred
    - -2: Channel selection wrong
    - -3: Sensor sensibility selection wrong
    - -4: Acquisition in progress. Can not change the configuration

**Returns**

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

**4.1.2.47** `int MSXExxxx_PressureSetSamplingRate ( xsd__unsignedLong ulChannelGroup, xsd__unsignedLong ulBaseSamplingRate, xsd__unsignedLong ulOption1, struct MSXExxxx_Response * Response )`

**Parameters**

- [in] ***ulChannelGroup*** : Channel group selection.
  - 0 for channels 0 and 1
  - 1 for channels 2 and 3
  - 2 for channels 4 and 5
  - 3 for channels 6 and 7
  - 4 for channels 8 and 9
  - 5 for channels 10 and 11
  - 6 for channels 12 and 13
  - 7 for channels 14 and 15
  - ...
  - 255 for all channels
- [in] ***ulBaseSamplingRate*** : Sampling rate selection
  - 5 for 5Hz
  - 10 for 10Hz
  - 20 for 20Hz
  - 40 for 40Hz
  - 80 for 80Hz
  - 160 for 160Hz
  - 320 for 320Hz
  - 640 for 640Hz
  - 1000 for 1000Hz
  - 2000 for 2000Hz

If only one channel is used then the real sampling rate is  $ulBaseSamplingRate / 2$

If all 2 channels are used then the real sampling rate is  $ulBaseSamplingRate / 3$

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

[out] *Response* :

*iReturnValue* :

- 0 : Means the remote function performed OK
- -1 : Means an system error occurred
- -2 : Channel group selection error
- -3 : Sampling rate selection error
- -4 : Acquisition in progress. Can not change the sampling rate
- -100 : IOCTL system call error

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

### Returns

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

**4.1.2.48** `int MSXExxxx__PressureGetConfiguration ( xsd__unsignedLong ulChannel,  
xsd__unsignedLong ulOption1, struct MSXExxxx__PressureGetConfigurationResponse  
* Response )`

### Parameters

[in] *ulChannel* : Channel selection (0 to 15)

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

[out] *Response* :

*sResponse.iReturnValue* :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Channel selection wrong

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

*dSensorSensibility* : Sensor sensibility (mV/V/bar or mV/V/mbar or mV/V/Pa, ...). Refer to sensor documentation

*dSensorOffset* : Sensor V offset for 0 mV/V/bar, 0 mV/V/mbar, ... Refer to sensor documentation

*ulBaseSamplingRate* : Sampling rate selection

- 5 for 5Hz
- 10 for 10Hz
- 20 for 20Hz
- 40 for 40Hz
- 80 for 80Hz
- 160 for 160Hz
- 320 for 320Hz
- 640 for 640Hz
- 1000 for 1000Hz
- 2000 for 2000Hz

**Returns**

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Channel selection wrong

# Index

- `__offset`
    - ByteArray, [45](#)
    - UnsignedLongArray, [58](#)
    - UnsignedShortArray, [59](#)
  - `__ptr`
    - ByteArray, [45](#)
    - UnsignedLongArray, [58](#)
    - UnsignedShortArray, [59](#)
    - `xsd__base64Binary`, [59](#)
  - `__size`
    - ByteArray, [45](#)
    - UnsignedLongArray, [58](#)
    - UnsignedShortArray, [59](#)
    - `xsd__base64Binary`, [59](#)
- `bArray`
  - `MXCommon__-`
    - `GetAutoConfigurationFileResponse`, [52](#)
- `bCryptedValueArray`
  - `MXCommon__TestCustomerIDResponse`, [57](#)
- `bValueArray`
  - `MXCommon__TestCustomerIDResponse`, [57](#)
- `ByteArray`, [45](#)
  - `__offset`, [45](#)
  - `__ptr`, [45](#)
  - `__size`, [45](#)
- Common functions, [3](#)
- Common general functions, [4](#)
- Common hardware trigger functions, [11](#)
- Common I/O auto configuration functions, [18](#)
- Common security functions, [13](#)
- Common synchronisation timer functions, [20](#)
- Common temperature functions, [10](#)
- Common time functions, [15](#)
- Common\_autoconf
  - `MXCommon__GetAutoConfigurationFile`, [19](#)
  - `MXCommon__SetAutoConfigurationFile`, [19](#)
  - `MXCommon__StartAutoConfiguration`, [20](#)
- Common\_configuration
  - `MXCommon__-`
    - `ApplyConfigurationBackupFile`, [23](#)
    - `MXCommon__ChangePassword`, [24](#)
    - `MXCommon__GetConfigurationBackupFile`, [23](#)
- Common\_general
  - `MXCommon__DataseverRestart`, [8](#)
  - `MXCommon__GetClientConnections`, [6](#)
  - `MXCommon__GetEthernetLinksStates`, [9](#)
  - `MXCommon__GetHostname`, [5](#)
  - `MXCommon__GetModuleType`, [5](#)
  - `MXCommon__Reboot`, [7](#)
  - `MXCommon__ResetAllIOFunctionalities`, [8](#)
  - `MXCommon__SetHostname`, [5](#)
  - `MXCommon__Sterror`, [6](#)
- Common\_hardware\_trigger
  - `MXCommon__-`
    - `GetHardwareTriggerFilterTime`, [12](#)
    - `MXCommon__GetHardwareTriggerState`, [13](#)
    - `MXCommon__-`
      - `SetHardwareTriggerFilterTime`, [12](#)
- Common\_security
  - `MXCommon__SetCustomerKey`, [15](#)
  - `MXCommon__TestCustomerID`, [15](#)
- Common\_synchrotimer
  - `MXCommon__InitAndStartSynchroTimer`, [21](#)
  - `MXCommon__-`
    - `StopAndReleaseSynchroTimer`, [21](#)
- Common\_temperature
  - `MXCommon__-`
    - `GetModuleTemperatureValueAndStatus`, [10](#)
    - `MXCommon__-`
      - `SetModuleTemperatureWarningLevels`, [11](#)
- Common\_time
  - `MXCommon__GetTime`, [17](#)
  - `MXCommon__GetUpTime`, [18](#)
  - `MXCommon__HardwareClockToSys`, [17](#)
  - `MXCommon__SetTime`, [16](#)
  - `MXCommon__SysToHardwareClock`, [16](#)
- Customer option management, [27](#)
- CustomerOption
  - `MXCommon__GetOptionInformation`, [28](#)
- DefaultResponse, [45](#)
  - `iReturnValue`, [46](#)
  - `syserrno`, [46](#)

- dSensorOffset
  - MSXExxxx\_\_-
    - PressureGetConfigurationResponse, 49
- dSensorSensibility
  - MSXExxxx\_\_-
    - PressureGetConfigurationResponse, 49
- dTemperatureValue
  - MXCommon\_\_-
    - GetModuleTemperatureValueAndStatusResponse, 55
- dValue
  - MSXExxxx\_\_-
    - AcquisitionAutoRefreshGetValuesResponse, 47
- iReturnValue
  - DefaultResponse, 46
  - MSXExxxx\_\_Response, 50
  - MXCommon\_\_Response, 56
- MSX-E331x Acquisition functions, 29
- MSX-E331x Acquisition information functions, 30
- MSX-E331x Autorefresh functions, 31
- MSX-E331x functions, 3
- MSX-E331x Pressure functions, 40
- MSX-E331x Pressure initialisation/information functions, 40
- MSX-E331x Sequence functions, 36
- MSXE331x\_Acquisition\_Info
  - MSXExxxx\_\_AcquisitionGetChannelInfo, 30
  - MSXExxxx\_\_-
    - AcquisitionGetNumberOfChannels, 30
- MSXE331x\_Autorefresh
  - MSXExxxx\_\_-
    - AcquisitionAutoRefreshGetValues, 35
  - MSXExxxx\_\_-
    - AcquisitionAutoRefreshInitAndStart, 33
  - MSXExxxx\_\_-
    - AcquisitionAutoRefreshStopAndRelease, 35
- MSXE331x\_Pressure\_Init
  - MSXExxxx\_\_PressureGetConfiguration, 42
  - MSXExxxx\_\_-
    - PressureGetNumberOfChannels, 41
  - MSXExxxx\_\_-
    - PressureSetChannelConfiguration, 41
  - MSXExxxx\_\_PressureSetSamplingRate, 41
- MSXE331x\_public\_doc.h, 61
- MSXExxxx\_\_-
  - AcquisitionAutoRefreshGetValues, 88
- MSXExxxx\_\_-
  - AcquisitionAutoRefreshInitAndStart, 86
- MSXExxxx\_\_-
  - AcquisitionAutoRefreshStopAndRelease, 89
- MSXExxxx\_\_AcquisitionGetChannelInfo, 85
- MSXExxxx\_\_-
  - AcquisitionGetNumberOfChannels, 85
- MSXExxxx\_\_-
  - AcquisitionSequenceInitAndStart, 89
- MSXExxxx\_\_-
  - AcquisitionSequenceStopAndRelease, 91
- MSXExxxx\_\_PressureGetConfiguration, 94
- MSXExxxx\_\_-
  - PressureGetNumberOfChannels, 92
- MSXExxxx\_\_-
  - PressureSetChannelConfiguration, 92
- MSXExxxx\_\_PressureSetSamplingRate, 93
- MXCommon\_\_-
  - ApplyConfigurationBackupFile, 80
- MXCommon\_\_ChangePassword, 81
- MXCommon\_\_DataseverRestart, 70
- MXCommon\_\_GetAutoConfigurationFile, 77
- MXCommon\_\_GetClientConnections, 68
- MXCommon\_\_GetConfigurationBackupFile, 80
- MXCommon\_\_GetEthernetLinksStates, 71
- MXCommon\_\_-
  - GetHardwareTriggerFilterTime, 73
- MXCommon\_\_GetHardwareTriggerState, 74
- MXCommon\_\_GetHostname, 67
- MXCommon\_\_-
  - GetModuleTemperatureValueAndStatus, 72
- MXCommon\_\_GetModuleType, 67
- MXCommon\_\_GetOptionInformation, 83
- MXCommon\_\_GetStateIDFromName, 82
- MXCommon\_\_GetStateNameFromID, 83
- MXCommon\_\_GetSubsystemIDFromName, 82
- MXCommon\_\_GetSubsystemNameFromID, 83
- MXCommon\_\_GetSubSystemState, 81
- MXCommon\_\_GetSynchronizationStatus, 84
- MXCommon\_\_GetTime, 76
- MXCommon\_\_GetUpTime, 77

- MXCommon\_\_HardwareClockToSys, [76](#)
- MXCommon\_\_InitAndStartSynchroTimer, [78](#)
- MXCommon\_\_Reboot, [70](#)
- MXCommon\_\_ResetAllIOFunctionalities, [70](#)
- MXCommon\_\_SetAutoConfigurationFile, [78](#)
- MXCommon\_\_SetCustomerKey, [74](#)
- MXCommon\_\_-
  - SetHardwareTriggerFilterTime, [73](#)
- MXCommon\_\_SetHostname, [68](#)
- MXCommon\_\_-
  - SetModuleTemperatureWarningLevels, [72](#)
- MXCommon\_\_SetTime, [75](#)
- MXCommon\_\_SetToMaster, [84](#)
- MXCommon\_\_StartAutoConfiguration, [78](#)
- MXCommon\_\_-
  - StopAndReleaseSynchroTimer, [79](#)
- MXCommon\_\_Strerror, [68](#)
- MXCommon\_\_SysToHardwareClock, [76](#)
- MXCommon\_\_TestCustomerID, [75](#)
- xsd\_\_char, [67](#)
- xsd\_\_double, [67](#)
- xsd\_\_float, [67](#)
- xsd\_\_int, [67](#)
- xsd\_\_long, [67](#)
- xsd\_\_string, [67](#)
- xsd\_\_unsignedByte, [67](#)
- xsd\_\_unsignedInt, [67](#)
- xsd\_\_unsignedLong, [67](#)
- xsd\_\_unsignedShort, [67](#)
- MSXE331x\_Sequence
  - MSXExxxx\_-
    - AcquisitionSequenceInitAndStart, [37](#)
  - MSXExxxx\_-
    - AcquisitionSequenceStopAndRelease, [39](#)
- MSXExxxx\_\_AcquisitionAutoRefreshGetValues
  - MSXE331x\_Autorefresh, [35](#)
  - MSXE331x\_public\_doc.h, [88](#)
- MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse,MSXE331x\_Pressure\_Init, [41](#)
  - MSXE331x\_public\_doc.h, [93](#)
- dValue, [47](#)
- sResponse, [47](#)
- ulCounterValue, [47](#)
- ulTimeStampHigh, [47](#)
- ulTimeStampLow, [47](#)
- MSXExxxx\_\_AcquisitionAutoRefreshInitAndStart
  - MSXE331x\_Autorefresh, [33](#)
  - MSXE331x\_public\_doc.h, [86](#)
- MSXExxxx\_\_AcquisitionAutoRefreshStopAndRelease
  - MSXE331x\_Autorefresh, [35](#)
  - MSXE331x\_public\_doc.h, [89](#)
- MSXExxxx\_\_AcquisitionGetChannelInfo
  - MSXE331x\_Acquisition\_Info, [30](#)
  - MSXE331x\_public\_doc.h, [85](#)
- MSXExxxx\_\_AcquisitionGetChannelInfoResponse, [47](#)
  - sResponse, [47](#)
  - ulChannelIndex, [48](#)
  - ulHwPosition, [48](#)
  - ulType, [47](#)
- MSXExxxx\_\_AcquisitionGetNumberOfChannels
  - MSXE331x\_Acquisition\_Info, [30](#)
  - MSXE331x\_public\_doc.h, [85](#)
- MSXExxxx\_\_AcquisitionSequenceInitAndStart
  - MSXE331x\_public\_doc.h, [89](#)
  - MSXE331x\_Sequence, [37](#)
- MSXExxxx\_\_AcquisitionSequenceInitAndStartChannelListParam, [48](#)
  - ulChannelList, [48](#)
- MSXExxxx\_\_AcquisitionSequenceStopAndRelease
  - MSXE331x\_public\_doc.h, [91](#)
  - MSXE331x\_Sequence, [39](#)
- MSXExxxx\_\_FileResponse, [48](#)
  - sArray, [49](#)
  - sResponse, [49](#)
  - ulEOF, [49](#)
- MSXExxxx\_\_PressureGetConfiguration
  - MSXE331x\_Pressure\_Init, [42](#)
  - MSXE331x\_public\_doc.h, [94](#)
- MSXExxxx\_\_PressureGetConfigurationResponse, [49](#)
  - dSensorOffset, [49](#)
  - dSensorSensibility, [49](#)
  - sResponse, [49](#)
  - ulBaseSamplingRate, [49](#)
- MSXExxxx\_\_PressureGetNumberOfChannels
  - MSXE331x\_Pressure\_Init, [41](#)
  - MSXE331x\_public\_doc.h, [92](#)
- MSXExxxx\_\_PressureSetChannelConfiguration
  - MSXE331x\_Pressure\_Init, [41](#)
  - MSXE331x\_public\_doc.h, [92](#)
- MSXExxxx\_\_PressureSetSamplingRate
  - MSXE331x\_Pressure\_Init, [41](#)
  - MSXE331x\_public\_doc.h, [93](#)
- MSXExxxx\_\_Response, [49](#)
  - iReturnValue, [50](#)
  - syserrno, [50](#)
- MSXExxxx\_\_unsignedLongResponse, [50](#)
  - sResponse, [50](#)
  - ulValue, [50](#)
- MSXExxxx\_\_unsignedLongTimeStampResponse, [50](#)
  - sResponse, [51](#)
  - ulTimeStampHigh, [51](#)
  - ulTimeStampLow, [51](#)
  - ulValue, [51](#)

- MXCommon\_\_ApplyConfigurationBackupFile
  - Common\_configuration, 23
  - MSXE331x\_public\_doc.h, 80
- MXCommon\_\_ByteArrayResponse, 51
  - sArray, 51
  - sResponse, 51
- MXCommon\_\_ChangePassword
  - Common\_configuration, 24
  - MSXE331x\_public\_doc.h, 81
- MXCommon\_\_DataseverRestart
  - Common\_general, 8
  - MSXE331x\_public\_doc.h, 70
- MXCommon\_\_FileResponse, 51
  - sArray, 52
  - sResponse, 52
  - ulEOF, 52
- MXCommon\_\_GetAutoConfigurationFile
  - Common\_autoconf, 19
  - MSXE331x\_public\_doc.h, 77
- MXCommon\_\_GetAutoConfigurationFileResponse, 52
  - bArray, 52
  - sResponse, 52
  - ulEOF, 52
- MXCommon\_\_GetClientConnections
  - Common\_general, 6
  - MSXE331x\_public\_doc.h, 68
- MXCommon\_\_GetConfigurationBackupFile
  - Common\_configuration, 23
  - MSXE331x\_public\_doc.h, 80
- MXCommon\_\_GetEthernetLinksStates
  - Common\_general, 9
  - MSXE331x\_public\_doc.h, 71
- MXCommon\_\_GetEthernetLinksStatesResponse, 52
  - sPort0, 53
  - sPort1, 53
  - sResponse, 53
- MXCommon\_\_GetHardwareTriggerFilterTime
  - Common\_hardware\_trigger, 12
  - MSXE331x\_public\_doc.h, 73
- MXCommon\_\_GetHardwareTriggerFilterTimeResponse, 53
  - sResponse, 53
  - ulFilterTime, 53
  - ulInfo01, 53
  - ulInfo02, 53
- MXCommon\_\_GetHardwareTriggerState
  - Common\_hardware\_trigger, 13
  - MSXE331x\_public\_doc.h, 74
- MXCommon\_\_GetHardwareTriggerStateResponse, 53
  - sResponse, 54
  - ulInfo01, 54
  - ulInfo02, 54
  - ulState, 54
- MXCommon\_\_GetHostname
  - Common\_general, 5
  - MSXE331x\_public\_doc.h, 67
- MXCommon\_\_GetModuleTemperatureValueAndStatus
  - Common\_temperature, 10
  - MSXE331x\_public\_doc.h, 72
- MXCommon\_\_GetModuleTemperatureValueAndStatusResponse, 54
  - dTemperatureValue, 55
  - sResponse, 55
  - ulInfo, 55
  - ulTemperatureStatus, 55
- MXCommon\_\_GetModuleType
  - Common\_general, 5
  - MSXE331x\_public\_doc.h, 67
- MXCommon\_\_GetOptionInformation
  - CustomerOption, 28
  - MSXE331x\_public\_doc.h, 83
- MXCommon\_\_GetStateIDFromName
  - MSXE331x\_public\_doc.h, 82
  - SystemStatemanagement, 26
- MXCommon\_\_GetStateNameFromID
  - MSXE331x\_public\_doc.h, 83
  - SystemStatemanagement, 27
- MXCommon\_\_GetSubsystemIDFromName
  - MSXE331x\_public\_doc.h, 82
  - SystemStatemanagement, 26
- MXCommon\_\_GetSubsystemNameFromID
  - MSXE331x\_public\_doc.h, 83
  - SystemStatemanagement, 26
- MXCommon\_\_GetSubSystemState
  - MSXE331x\_public\_doc.h, 81
  - SystemStatemanagement, 25
- MXCommon\_\_GetSynchronizationStatus
  - MSXE331x\_public\_doc.h, 84
  - Synchronisation, 29
- MXCommon\_\_GetTime
  - Common\_time, 17
  - MSXE331x\_public\_doc.h, 76
- MXCommon\_\_GetTimeResponse, 55
  - sResponse, 55
  - ulHighTime, 55
  - ulLowTime, 55
- MXCommon\_\_GetUpTime
  - Common\_time, 18
  - MSXE331x\_public\_doc.h, 77
- MXCommon\_\_GetUpTimeResponse, 55
  - sResponse, 56
  - ulUpTime, 56
- MXCommon\_\_HardwareClockToSys
  - Common\_time, 17
  - MSXE331x\_public\_doc.h, 76

- MXCommon\_\_InitAndStartSynchroTimer
  - Common\_synchrotimer, [21](#)
  - MSXE331x\_public\_doc.h, [78](#)
- MXCommon\_\_Reboot
  - Common\_general, [7](#)
  - MSXE331x\_public\_doc.h, [70](#)
- MXCommon\_\_ResetAllIOFunctionalities
  - Common\_general, [8](#)
  - MSXE331x\_public\_doc.h, [70](#)
- MXCommon\_\_Response, [56](#)
  - iReturnValue, [56](#)
  - syserrno, [56](#)
- MXCommon\_\_SetAutoConfigurationFile
  - Common\_autoconf, [19](#)
  - MSXE331x\_public\_doc.h, [78](#)
- MXCommon\_\_SetCustomerKey
  - Common\_security, [15](#)
  - MSXE331x\_public\_doc.h, [74](#)
- MXCommon\_\_SetHardwareTriggerFilterTime
  - Common\_hardware\_trigger, [12](#)
  - MSXE331x\_public\_doc.h, [73](#)
- MXCommon\_\_SetHostname
  - Common\_general, [5](#)
  - MSXE331x\_public\_doc.h, [68](#)
- MXCommon\_\_SetModuleTemperatureWarningLevels
  - Common\_temperature, [11](#)
  - MSXE331x\_public\_doc.h, [72](#)
- MXCommon\_\_SetTime
  - Common\_time, [16](#)
  - MSXE331x\_public\_doc.h, [75](#)
- MXCommon\_\_SetToMaster
  - MSXE331x\_public\_doc.h, [84](#)
  - Synchronisation, [28](#)
- MXCommon\_\_StartAutoConfiguration
  - Common\_autoconf, [20](#)
  - MSXE331x\_public\_doc.h, [78](#)
- MXCommon\_\_StopAndReleaseSynchroTimer
  - Common\_synchrotimer, [21](#)
  - MSXE331x\_public\_doc.h, [79](#)
- MXCommon\_\_Sterror
  - Common\_general, [6](#)
  - MSXE331x\_public\_doc.h, [68](#)
- MXCommon\_\_SysToHardwareClock
  - Common\_time, [16](#)
  - MSXE331x\_public\_doc.h, [76](#)
- MXCommon\_\_TestCustomerID
  - Common\_security, [15](#)
  - MSXE331x\_public\_doc.h, [75](#)
- MXCommon\_\_TestCustomerIDResponse, [56](#)
  - bCryptedValueArray, [57](#)
  - bValueArray, [57](#)
  - sResponse, [57](#)
- MXCommon\_\_unsignedLongResponse, [57](#)
  - sResponse, [57](#)
- ulValue, [57](#)
- sArray
  - MSXExxxx\_\_FileResponse, [49](#)
  - MXCommon\_\_ByteArrayResponse, [51](#)
  - MXCommon\_\_FileResponse, [52](#)
- Set/Backup/Restore general system configuration, [22](#)
- sGetEthernetLinksStatesPort, [57](#)
  - ulDuplex, [58](#)
  - ulInfo1, [58](#)
  - ulInfo2, [58](#)
  - ulSpeed, [58](#)
  - ulState, [58](#)
- sPort0
  - MXCommon\_\_ - GetEthernetLinksStatesResponse, [53](#)
- sPort1
  - MXCommon\_\_ - GetEthernetLinksStatesResponse, [53](#)
- sResponse
  - MSXExxxx\_\_ - AcquisitionAutoRefreshGetValuesResponse, [47](#)
  - MSXExxxx\_\_ - AcquisitionGetChannelInfoResponse, [47](#)
  - MSXExxxx\_\_FileResponse, [49](#)
  - MSXExxxx\_\_ - PressureGetConfigurationResponse, [49](#)
  - MSXExxxx\_\_unsignedLongResponse, [50](#)
  - MSXExxxx\_\_ - unsignedLongTimeStampResponse, [51](#)
  - MXCommon\_\_ByteArrayResponse, [51](#)
  - MXCommon\_\_FileResponse, [52](#)
  - MXCommon\_\_ - GetAutoConfigurationFileResponse, [52](#)
  - MXCommon\_\_ - GetEthernetLinksStatesResponse, [53](#)
  - MXCommon\_\_ - GetHardwareTriggerFilterTimeResponse, [53](#)
  - MXCommon\_\_ - GetHardwareTriggerStateResponse, [54](#)
  - MXCommon\_\_ - GetModuleTemperatureValueAndStatusResponse, [55](#)
  - MXCommon\_\_GetTimeResponse, [55](#)
  - MXCommon\_\_GetUpTimeResponse, [56](#)
  - MXCommon\_\_TestCustomerIDResponse, [57](#)



- MXCommon\_\_unsignedLongResponse, [57](#)
- Synchronisation
  - MXCommon\_\_GetSynchronizationStatus, [29](#)
  - MXCommon\_\_SetToMaster, [28](#)
- Synchronisation management, [28](#)
- syserrno
  - DefaultResponse, [46](#)
  - MSXExxxx\_\_Response, [50](#)
  - MXCommon\_\_Response, [56](#)
- System state management, [24](#)
- SystemStatemanagement
  - MXCommon\_\_GetStateIDFromName, [26](#)
  - MXCommon\_\_GetStateNameFromID, [27](#)
  - MXCommon\_\_GetSubsystemIDFromName, [26](#)
  - MXCommon\_\_GetSubsystemNameFromID, [26](#)
  - MXCommon\_\_GetSubSystemState, [25](#)
- ulBaseSamplingRate
  - MSXExxxx\_\_-
    - PressureGetConfigurationResponse, [49](#)
- ulChannelIndex
  - MSXExxxx\_\_-
    - AcquisitionGetChannelInfoResponse, [48](#)
- ulChannelList
  - MSXExxxx\_\_-
    - AcquisitionSequenceInitAndStartChannelListParameters, [48](#)
- ulCounterValue
  - MSXExxxx\_\_-
    - AcquisitionAutoRefreshGetValuesResponse, [47](#)
- ulDuplex
  - sGetEthernetLinksStatesPort, [58](#)
- ulEOF
  - MSXExxxx\_\_FileResponse, [49](#)
  - MXCommon\_\_FileResponse, [52](#)
  - MXCommon\_\_-
    - GetAutoConfigurationFileResponse, [52](#)
- ulFilterTime
  - MXCommon\_\_-
    - GetHardwareTriggerFilterTimeResponse, [53](#)
- ulHighTime
  - MXCommon\_\_GetTimeResponse, [55](#)
- ulHwPosition
  - MSXExxxx\_\_-
    - AcquisitionGetChannelInfoResponse, [48](#)
- ulInfo
  - MXCommon\_\_-
    - GetModuleTemperatureValueAndStatusResponse, [55](#)
  - ulInfo01
    - MXCommon\_\_-
      - GetHardwareTriggerFilterTimeResponse, [53](#)
    - MXCommon\_\_-
      - GetHardwareTriggerStateResponse, [54](#)
  - ulInfo02
    - MXCommon\_\_-
      - GetHardwareTriggerFilterTimeResponse, [53](#)
    - MXCommon\_\_-
      - GetHardwareTriggerStateResponse, [54](#)
  - ulInfo1
    - sGetEthernetLinksStatesPort, [58](#)
  - ulInfo2
    - sGetEthernetLinksStatesPort, [58](#)
  - ulLowTime
    - MXCommon\_\_GetTimeResponse, [55](#)
  - ulSpeed
    - sGetEthernetLinksStatesPort, [58](#)
  - ulState
    - MXCommon\_\_-
      - GetHardwareTriggerStateResponse, [54](#)
    - sGetEthernetLinksStatesPort, [58](#)
  - ulTemperatureStatus
    - MXCommon\_\_-
      - GetModuleTemperatureValueAndStatusResponse, [55](#)
  - ulTimeStampHigh
    - MSXExxxx\_\_-
      - AcquisitionAutoRefreshGetValuesResponse, [47](#)
    - MSXExxxx\_\_-
      - unsignedLongTimeStampResponse, [51](#)
  - ulTimeStampLow
    - MSXExxxx\_\_-
      - AcquisitionAutoRefreshGetValuesResponse, [47](#)
    - MSXExxxx\_\_-
      - unsignedLongTimeStampResponse, [51](#)
  - ulType
    - MSXExxxx\_\_-
      - AcquisitionGetChannelInfoResponse, [47](#)
  - ulUpTime
    - MXCommon\_\_GetUpTimeResponse, [56](#)

- ulValue
  - MSXExxxx\_\_unsignedLongResponse, [50](#)
  - MSXExxxx\_\_-
    - unsignedLongTimeStampResponse, [51](#)
  - MXCommon\_\_unsignedLongResponse, [57](#)
- UnsignedLongArray, [58](#)
  - \_\_offset, [58](#)
  - \_\_ptr, [58](#)
  - \_\_size, [58](#)
- UnsignedShortArray, [58](#)
  - \_\_offset, [59](#)
  - \_\_ptr, [59](#)
  - \_\_size, [59](#)
- xsd\_\_base64Binary, [59](#)
  - \_\_ptr, [59](#)
  - \_\_size, [59](#)
- xsd\_\_char
  - MSXE331x\_public\_doc.h, [67](#)
- xsd\_\_double
  - MSXE331x\_public\_doc.h, [67](#)
- xsd\_\_float
  - MSXE331x\_public\_doc.h, [67](#)
- xsd\_\_int
  - MSXE331x\_public\_doc.h, [67](#)
- xsd\_\_long
  - MSXE331x\_public\_doc.h, [67](#)
- xsd\_\_string
  - MSXE331x\_public\_doc.h, [67](#)
- xsd\_\_unsignedByte
  - MSXE331x\_public\_doc.h, [67](#)
- xsd\_\_unsignedInt
  - MSXE331x\_public\_doc.h, [67](#)
- xsd\_\_unsignedLong
  - MSXE331x\_public\_doc.h, [67](#)
- xsd\_\_unsignedShort
  - MSXE331x\_public\_doc.h, [67](#)