

## MSXE312x soap api functions

Generated by Doxygen 1.7.1

Tue Aug 20 2013 16:20:41



# 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-E312x functions	3
2.2	Software hints	3
2.3	SOAP function calls in C/C++ language	3
2.3.1	C/C++ language SOAP function prototypes	4
2.3.2	Rules and use of gSOAP calls in C/C++	4
2.3.3	SOAP calls sample	6
2.4	MSX-E systems servers	10
2.4.1	SOAP server	10
2.4.2	Event server	10
2.4.3	Modbus server	10
2.5	Common functions	11
2.6	Common general functions	11
2.6.1	Function Documentation	12
2.6.1.1	MXCommon__GetModuleType	12
2.6.1.2	MXCommon__GetHostname	13
2.6.1.3	MXCommon__SetHostname	13
2.6.1.4	MXCommon__GetClientConnections	13
2.6.1.5	MXCommon__Strerror	14
2.6.1.6	MXCommon__Reboot	15
2.6.1.7	MXCommon__ResetAllIOFunctionalities	15
2.6.1.8	MXCommon__DataseverRestart	16
2.6.1.9	MXCommon__GetEthernetLinksStates	16
2.7	Common temperature functions	17

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

2.13.2.2	MXCommon__ApplyConfigurationBackupFile	31
2.13.2.3	MXCommon__ChangePassword	31
2.14	System state management	32
2.14.1	Detailed Description	32
2.14.2	Function Documentation	32
2.14.2.1	MXCommon__GetSubSystemState	32
2.14.2.2	MXCommon__GetSubsystemIDFromName	33
2.14.2.3	MXCommon__GetStateIDFromName	33
2.14.2.4	MXCommon__GetSubsystemNameFromID	34
2.14.2.5	MXCommon__GetStateNameFromID	34
2.15	Customer option management	34
2.15.1	Function Documentation	35
2.15.1.1	MXCommon__GetOptionInformation	35
2.16	Synchronisation management	35
2.16.1	Function Documentation	35
2.16.1.1	MXCommon__SetToMaster	35
2.16.1.2	MXCommon__GetSynchronizationStatus	36
2.17	MSX-E312x Acquisition functions	36
2.18	MSX-E312x Acquisition information functions	37
2.18.1	Function Documentation	37
2.18.1.1	MSXExxxx__AcquisitionGetNumberOfChannels	37
2.18.1.2	MSXExxxx__AcquisitionGetChannelInfo	38
2.19	MSX-E312x Autorefresh functions	38
2.19.1	Detailed Description	39
2.19.2	Function Documentation	40
2.19.2.1	MSXExxxx__AcquisitionAutoRefreshInitAndStart	40
2.19.2.2	MSXExxxx__AcquisitionAutoRefreshGetValues	42
2.19.2.3	MSXExxxx__AcquisitionAutoRefreshStopAndRelease	42
2.20	MSX-E312x Sequence functions	43
2.20.1	Detailed Description	43
2.20.2	Function Documentation	44
2.20.2.1	MSXExxxx__AcquisitionSequenceInitAndStart	44
2.20.2.2	MSXExxxx__AcquisitionSequenceStopAndRelease	46
2.21	MSX-E312x Analog input functions	47
2.22	MSX-E312x Analog input initialisation functions	47
2.22.1	Function Documentation	48

2.22.1.1	MSXExxxx__AnalogInputGetNumberOfChannels . . . . .	48
2.22.1.2	MSXExxxx__AnalogInputGetChannelsType . . . . .	48
2.22.1.3	MSXExxxx__AnalogInputSetChannelConfiguration . . . . .	49
2.22.1.4	MSXExxxx__AnalogInputSetSamplingRate . . . . .	50
2.22.1.5	MSXExxxx__AnalogInputGetSamplingRate . . . . .	52
2.22.1.6	MSXExxxx__AnalogInputGetConfiguration . . . . .	52
2.23	MSX-E312x Analog input calibration functions . . . . .	53
2.23.1	Function Documentation . . . . .	54
2.23.1.1	MSXExxxx__AnalogInputCalibrationStart . . . . .	54
2.23.1.2	MSXExxxx__AnalogInputCalibrationGetCurrentStatus . . . . .	55
2.23.1.3	MSXExxxx__AnalogInputCalibrationNextStep . . . . .	55
2.23.1.4	MSXExxxx__AnalogInputCalibrationBreak . . . . .	56
2.24	MSX-E312x Analog output functions . . . . .	56
2.24.1	Detailed Description . . . . .	56
2.24.2	Function Documentation . . . . .	57
2.24.2.1	MSXExxxx__AnalogOutputGetNumberOfChannels . . . . .	57
2.24.2.2	MSXExxxx__AnalogOutputWrite1Value . . . . .	57
2.24.2.3	MSXExxxx__AnalogOutputGetChannelValue . . . . .	59
2.24.2.4	MSXExxxx__AnalogOutputTriggerOutputs . . . . .	59
2.25	MSX-E312x Digital I/O functions . . . . .	60
2.26	MSX-E312x Digital I/O information, configuration functions . . . . .	60
2.26.1	Function Documentation . . . . .	61
2.26.1.1	MSXExxxx__DigitalIOGetNumberOfChannels . . . . .	61
2.26.1.2	MSXExxxx__DigitalIOGetNumberOfPorts . . . . .	61
2.26.1.3	MSXExxxx__DigitalIOGetPortAvailableDirections . . . . .	62
2.26.1.4	MSXExxxx__DigitalIOSetPortDirections . . . . .	62
2.26.1.5	MSXExxxx__DigitalIOGetPortDirections . . . . .	63
2.27	MSX-E312x Digital I/O filter functions . . . . .	63
2.27.1	Function Documentation . . . . .	64
2.27.1.1	MSXExxxx__DigitalIOSetInputsFilterTime . . . . .	64
2.27.1.2	MSXExxxx__DigitalIOEnableDisableInputsFilter . . . . .	65
2.27.1.3	MSXExxxx__DigitalIOGetInputsFilterConfiguration . . . . .	65
2.28	MSX-E312x Digital I/O diagnostic functions . . . . .	66
2.28.1	Function Documentation . . . . .	66
2.28.1.1	MSXExxxx__DigitalIOTestOutputsShortCircuit . . . . .	66
2.28.1.2	MSXExxxx__DigitalIORearmOutputsShortCircuit . . . . .	67

2.28.1.3	MSXExxxx__DigitalIOTestOutputsPowerSupply	67
2.29	MSX-E312x Digital I/O read/write functions	68
2.29.1	Function Documentation	68
2.29.1.1	MSXExxxx__DigitalIOReadChannel	68
2.29.1.2	MSXExxxx__DigitalIOReadPort	69
2.29.1.3	MSXExxxx__DigitalIOWriteChannel	70
2.29.1.4	MSXExxxx__DigitalIOWritePort	70
<b>3</b>	<b>Data Structure Documentation</b>	<b>71</b>
3.1	ByteArray Struct Reference	71
3.1.1	Field Documentation	71
3.1.1.1	__ptr	71
3.1.1.2	__size	71
3.1.1.3	__offset	71
3.2	DefaultResponse Struct Reference	71
3.2.1	Field Documentation	72
3.2.1.1	iReturnValue	72
3.2.1.2	syserrno	72
3.3	MSXExxxx__AcquisitionAutoRefreshGetValuesResponse Struct Reference	72
3.3.1	Field Documentation	73
3.3.1.1	sResponse	73
3.3.1.2	ulTimeStampLow	73
3.3.1.3	ulTimeStampHigh	73
3.3.1.4	ulCounterValue	73
3.3.1.5	dValue	73
3.4	MSXExxxx__AcquisitionGetChannelInfoResponse Struct Reference	73
3.4.1	Field Documentation	73
3.4.1.1	sResponse	73
3.4.1.2	ulType	73
3.4.1.3	ulHwPosition	74
3.4.1.4	ulChannelIndex	74
3.5	MSXExxxx__AcquisitionSequenceInitAndStartChannelListParam Struct Reference	74
3.5.1	Field Documentation	74
3.5.1.1	ulChannelList	74
3.6	MSXExxxx__AnalogInputCalibrationGetCurrentStatusResponse Struct Reference	74
3.6.1	Field Documentation	75
3.6.1.1	sResponse	75

3.6.1.2	ulStatus	75
3.6.1.3	ulChannel	75
3.6.1.4	dRefValue	75
3.6.1.5	dCurrentValue	75
3.7	MSXExxxx__AnalogInputGetChannelsTypeResponse Struct Reference	75
3.7.1	Field Documentation	75
3.7.1.1	sResponse	75
3.7.1.2	ulType	75
3.7.1.3	ulICP	75
3.8	MSXExxxx__AnalogInputGetConfigurationResponse Struct Reference	75
3.8.1	Field Documentation	77
3.8.1.1	sResponse	77
3.8.1.2	ulType	77
3.8.1.3	ulICPAvailable	77
3.8.1.4	ulAcDc	77
3.8.1.5	ulSingleDiff	77
3.8.1.6	ulPolarity	77
3.8.1.7	ulGain	77
3.8.1.8	ulICP	77
3.8.1.9	ulTimeBase	77
3.8.1.10	ulTimeBaseUnity	77
3.8.1.11	ulSamplingRate	77
3.9	MSXExxxx__AnalogInputGetSamplingRateResponse Struct Reference	77
3.9.1	Field Documentation	77
3.9.1.1	sResponse	77
3.9.1.2	ulClk	77
3.9.1.3	ulDivisionFactor	78
3.10	MSXExxxx__DigitalIOGetInputsFilterConfigurationResponse Struct Reference	78
3.10.1	Field Documentation	78
3.10.1.1	sResponse	78
3.10.1.2	ulFilterTime	78
3.10.1.3	ulFilter	78
3.11	MSXExxxx__DigitalIOGetPortAvailableDirectionsResponse Struct Reference	78
3.11.1	Field Documentation	79
3.11.1.1	sResponse	79
3.11.1.2	ulInputs	79



3.11.1.3	ulOutputs	79
3.12	MSXExxxx__FileResponse Struct Reference	79
3.12.1	Field Documentation	80
3.12.1.1	sResponse	80
3.12.1.2	sArray	80
3.12.1.3	ulEOF	80
3.13	MSXExxxx__Response Struct Reference	80
3.13.1	Field Documentation	80
3.13.1.1	iReturnValue	80
3.13.1.2	syserrno	80
3.14	MSXExxxx__unsignedLongResponse Struct Reference	80
3.14.1	Field Documentation	81
3.14.1.1	sResponse	81
3.14.1.2	ulValue	81
3.15	MSXExxxx__unsignedLongTimeStampResponse Struct Reference	81
3.15.1	Field Documentation	81
3.15.1.1	sResponse	81
3.15.1.2	ulValue	81
3.15.1.3	ulTimeStampLow	81
3.15.1.4	ulTimeStampHigh	81
3.16	MXCommon__ByteArrayResponse Struct Reference	81
3.16.1	Field Documentation	82
3.16.1.1	sResponse	82
3.16.1.2	sArray	82
3.17	MXCommon__FileResponse Struct Reference	82
3.17.1	Field Documentation	82
3.17.1.1	sResponse	82
3.17.1.2	sArray	82
3.17.1.3	ulEOF	82
3.18	MXCommon__GetAutoConfigurationFileResponse Struct Reference	82
3.18.1	Field Documentation	83
3.18.1.1	sResponse	83
3.18.1.2	bArray	83
3.18.1.3	ulEOF	83
3.19	MXCommon__GetEthernetLinksStatesResponse Struct Reference	83
3.19.1	Field Documentation	83

3.19.1.1	sResponse	83
3.19.1.2	sPort0	83
3.19.1.3	sPort1	83
3.20	MXCommon__GetHardwareTriggerFilterTimeResponse Struct Reference	83
3.20.1	Field Documentation	84
3.20.1.1	sResponse	84
3.20.1.2	ulFilterTime	84
3.20.1.3	ulInfo01	84
3.20.1.4	ulInfo02	84
3.21	MXCommon__GetHardwareTriggerStateResponse Struct Reference	84
3.21.1	Field Documentation	84
3.21.1.1	sResponse	84
3.21.1.2	ulState	84
3.21.1.3	ulInfo01	84
3.21.1.4	ulInfo02	84
3.22	MXCommon__GetModuleTemperatureValueAndStatusResponse Struct Reference	84
3.22.1	Field Documentation	85
3.22.1.1	sResponse	85
3.22.1.2	dTemperatureValue	85
3.22.1.3	ulTemperatureStatus	85
3.22.1.4	ulInfo	85
3.23	MXCommon__GetTimeResponse Struct Reference	85
3.23.1	Field Documentation	85
3.23.1.1	sResponse	85
3.23.1.2	ulLowTime	85
3.23.1.3	ulHighTime	85
3.24	MXCommon__GetUpTimeResponse Struct Reference	85
3.24.1	Field Documentation	86
3.24.1.1	sResponse	86
3.24.1.2	ulUpTime	86
3.25	MXCommon__Response Struct Reference	86
3.25.1	Field Documentation	86
3.25.1.1	iReturnValue	86
3.25.1.2	syserrno	86
3.26	MXCommon__TestCustomerIDResponse Struct Reference	86
3.26.1	Field Documentation	87

3.26.1.1	sResponse	87
3.26.1.2	bValueArray	87
3.26.1.3	bCryptedValueArray	87
3.27	MXCommon__unsignedLongResponse Struct Reference	87
3.27.1	Field Documentation	87
3.27.1.1	sResponse	87
3.27.1.2	ulValue	87
3.28	sGetEthernetLinksStatesPort Struct Reference	87
3.28.1	Field Documentation	88
3.28.1.1	ulState	88
3.28.1.2	ulSpeed	88
3.28.1.3	ulDuplex	88
3.28.1.4	ulInfo1	88
3.28.1.5	ulInfo2	88
3.29	UnsignedLongArray Struct Reference	88
3.29.1	Field Documentation	88
3.29.1.1	__ptr	88
3.29.1.2	__size	88
3.29.1.3	__offset	88
3.30	UnsignedShortArray Struct Reference	88
3.30.1	Field Documentation	89
3.30.1.1	__ptr	89
3.30.1.2	__size	89
3.30.1.3	__offset	89
3.31	xsd__base64Binary Struct Reference	89
3.31.1	Field Documentation	89
3.31.1.1	__ptr	89
3.31.1.2	__size	89
<b>4</b>	<b>File Documentation</b>	<b>91</b>
4.1	MSXE312x_public_doc.h File Reference	91
4.1.1	Typedef Documentation	99
4.1.1.1	xsd__string	99
4.1.1.2	xsd__char	99
4.1.1.3	xsd__float	99
4.1.1.4	xsd__double	99
4.1.1.5	xsd__int	99

4.1.1.6	<a href="#">xsd__long</a>	99
4.1.1.7	<a href="#">xsd__unsignedByte</a>	99
4.1.1.8	<a href="#">xsd__unsignedInt</a>	99
4.1.1.9	<a href="#">xsd__unsignedShort</a>	99
4.1.1.10	<a href="#">xsd__unsignedLong</a>	99
4.1.2	<a href="#">Function Documentation</a>	99
4.1.2.1	<a href="#">MXCommon__GetModuleType</a>	99
4.1.2.2	<a href="#">MXCommon__GetHostname</a>	100
4.1.2.3	<a href="#">MXCommon__SetHostname</a>	100
4.1.2.4	<a href="#">MXCommon__GetClientConnections</a>	100
4.1.2.5	<a href="#">MXCommon__Strerror</a>	101
4.1.2.6	<a href="#">MXCommon__Reboot</a>	102
4.1.2.7	<a href="#">MXCommon__ResetAllIOFunctionalities</a>	102
4.1.2.8	<a href="#">MXCommon__DataseverRestart</a>	103
4.1.2.9	<a href="#">MXCommon__GetEthernetLinksStates</a>	103
4.1.2.10	<a href="#">MXCommon__GetModuleTemperatureValueAndStatus</a>	104
4.1.2.11	<a href="#">MXCommon__SetModuleTemperatureWarningLevels</a>	105
4.1.2.12	<a href="#">MXCommon__SetHardwareTriggerFilterTime</a>	105
4.1.2.13	<a href="#">MXCommon__GetHardwareTriggerFilterTime</a>	106
4.1.2.14	<a href="#">MXCommon__GetHardwareTriggerState</a>	106
4.1.2.15	<a href="#">MXCommon__SetCustomerKey</a>	107
4.1.2.16	<a href="#">MXCommon__TestCustomerID</a>	107
4.1.2.17	<a href="#">MXCommon__SetTime</a>	107
4.1.2.18	<a href="#">MXCommon__SysToHardwareClock</a>	108
4.1.2.19	<a href="#">MXCommon__HardwareClockToSys</a>	108
4.1.2.20	<a href="#">MXCommon__GetTime</a>	109
4.1.2.21	<a href="#">MXCommon__GetUpTime</a>	109
4.1.2.22	<a href="#">MXCommon__GetAutoConfigurationFile</a>	109
4.1.2.23	<a href="#">MXCommon__SetAutoConfigurationFile</a>	110
4.1.2.24	<a href="#">MXCommon__StartAutoConfiguration</a>	110
4.1.2.25	<a href="#">MXCommon__InitAndStartSynchroTimer</a>	110
4.1.2.26	<a href="#">MXCommon__StopAndReleaseSynchroTimer</a>	111
4.1.2.27	<a href="#">MXCommon__GetConfigurationBackupFile</a>	112
4.1.2.28	<a href="#">MXCommon__ApplyConfigurationBackupFile</a>	113
4.1.2.29	<a href="#">MXCommon__ChangePassword</a>	113
4.1.2.30	<a href="#">MXCommon__GetSubSystemState</a>	114

4.1.2.31	MXCommon__GetSubsystemIDFromName	114
4.1.2.32	MXCommon__GetStateIDFromName	114
4.1.2.33	MXCommon__GetSubsystemNameFromID	115
4.1.2.34	MXCommon__GetStateNameFromID	115
4.1.2.35	MXCommon__GetOptionInformation	116
4.1.2.36	MXCommon__SetToMaster	116
4.1.2.37	MXCommon__GetSynchronizationStatus	116
4.1.2.38	MSXExxxx__AcquisitionGetNumberOfChannels	117
4.1.2.39	MSXExxxx__AcquisitionGetChannelInfo	117
4.1.2.40	MSXExxxx__AcquisitionAutoRefreshInitAndStart	118
4.1.2.41	MSXExxxx__AcquisitionAutoRefreshGetValues	120
4.1.2.42	MSXExxxx__AcquisitionAutoRefreshStopAndRelease	121
4.1.2.43	MSXExxxx__AcquisitionSequenceInitAndStart	121
4.1.2.44	MSXExxxx__AcquisitionSequenceStopAndRelease	123
4.1.2.45	MSXExxxx__AnalogInputGetNumberOfChannels	124
4.1.2.46	MSXExxxx__AnalogInputGetChannelsType	124
4.1.2.47	MSXExxxx__AnalogInputSetChannelConfiguration	125
4.1.2.48	MSXExxxx__AnalogInputSetSamplingRate	126
4.1.2.49	MSXExxxx__AnalogInputGetSamplingRate	128
4.1.2.50	MSXExxxx__AnalogInputGetConfiguration	128
4.1.2.51	MSXExxxx__AnalogInputCalibrationStart	129
4.1.2.52	MSXExxxx__AnalogInputCalibrationGetCurrentStatus	130
4.1.2.53	MSXExxxx__AnalogInputCalibrationNextStep	131
4.1.2.54	MSXExxxx__AnalogInputCalibrationBreak	131
4.1.2.55	MSXExxxx__AnalogOutputGetNumberOfChannels	131
4.1.2.56	MSXExxxx__AnalogOutputWrite1Value	132
4.1.2.57	MSXExxxx__AnalogOutputGetChannelValue	133
4.1.2.58	MSXExxxx__AnalogOutputTriggerOutputs	133
4.1.2.59	MSXExxxx__DigitalIOGetNumberOfChannels	134
4.1.2.60	MSXExxxx__DigitalIOGetNumberOfPorts	134
4.1.2.61	MSXExxxx__DigitalIOGetPortAvailableDirections	135
4.1.2.62	MSXExxxx__DigitalIOSetPortDirections	136
4.1.2.63	MSXExxxx__DigitalIOGetPortDirections	136
4.1.2.64	MSXExxxx__DigitalIOSetInputsFilterTime	137
4.1.2.65	MSXExxxx__DigitalIOEnableDisableInputsFilter	137
4.1.2.66	MSXExxxx__DigitalIOGetInputsFilterConfiguration	138

4.1.2.67	MSXExxxx__DigitalIOTestOutputsShortCircuit . . . . .	139
4.1.2.68	MSXExxxx__DigitalIORearmOutputsShortCircuit . . . . .	139
4.1.2.69	MSXExxxx__DigitalIOTestOutputsPowerSupply . . . . .	140
4.1.2.70	MSXExxxx__DigitalIOReadChannel . . . . .	140
4.1.2.71	MSXExxxx__DigitalIOReadPort . . . . .	141
4.1.2.72	MSXExxxx__DigitalIOWriteChannel . . . . .	141
4.1.2.73	MSXExxxx__DigitalIOWritePort . . . . .	142

# 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-E312x functions

#### Modules

- [MSX-E312x Acquisition functions](#)  
*Contain the acquisition information and initialisation functions.*
- [MSX-E312x Analog input functions](#)  
*Contain the analog input initialisation and calibration functions.*
- [MSX-E312x Analog output functions](#)  
*Contain the analog output initialisation and read/write functions.*
- [MSX-E312x Digital I/O functions](#)  
*Contain the digital I/O functions.*

### 2.2 Software hints

#### Modules

- [SOAP function calls in C/C++ language](#)  
*Some hints on how to use the SOAP functions in a C/C++ program.*
- [MSX-E systems servers](#)  
*The MSX-E embeds servers to provide configuration, management and monitoring over Ethernet.*

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

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

### 2.3.1 C/C++ language SOAP function prototypes

In this documentation, functions are described as follows.

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

Two main differences can be remarked in the function prototypes:

- The prefix:

```
soap_call__
```

- The first three parameters of the SOAP stub:

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

The C function prototype has the following syntax:

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

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

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

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

**Note:** Software code pieces in this chapter comes from [SOAP calls sample](#)

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

```
struct soap *soapContext;

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

char soap_endpoint[] = IP_ADDRESS":5555";

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

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

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

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

### • Calling the MSX-E SOAP functions

#### Important

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

See [C/C++ language SOAP function prototypes](#) to call the MSX-E SOAP functions.

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

    ...

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

    ...
}
```

### • Error management

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

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

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

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

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

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

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

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

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

```
struct MXCommon__ByteArrayResponse byteArrayResponse;

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

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

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

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

- **Close the SOAP connection**

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

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

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

### 2.3.3 SOAP calls sample

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

To compile it, use *make IP\_ADDRESS=YOUR\_MSX-E\_IP\_ADDRESS* don't forget to set the **INTERFACE\_COMMON\_LIBRARY\_DIR** to point on the MSX-E Common Interface\_Library directory.

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

temperature.c file

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

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

/** kbhit for linux.
```

```

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

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

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

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

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

//-----

/** getch for linux.

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

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

    return (ch);
}

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

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

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

```

```

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

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

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

        return 1;
    }

    return 0;
}

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

int main (void)
{
    struct soap *soapContext;
    unsigned long option = 0;
    struct MXCommon__GetModuleTemperatureValueAndStatusResponse tempResponse
;
    int soap_error = 0;
    char *tempStatus[] = {"NOT AVAILABLE", "TOO LOW", "LOW", "NOMINAL", "HIG
H", "TOO HIGH"};
    // IP address of the MSX-E and after the ':' is the MSX-E SOAP server po
rt number
    char soap_endpoint[] = IP_ADDRESS":5555";

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

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

```

```

        return EXIT_FAILURE;

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

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

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

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

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

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

        printf ("\n");

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

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

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

    return EXIT_SUCCESS;
}

```

## Makefile

```

TOPDIR                                :=$(shell pwd)

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

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

```

```

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

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

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

TEMPERATURE_SRC                         := temperature.o
BINS                                    := temperature

all: $(BINS)

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

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

```

## 2.4 MSX-E systems servers

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

### 2.4.1 SOAP server

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

### 2.4.2 Event server

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

### 2.4.3 Modbus server

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



## 2.5 Common functions

### Modules

- [Common general functions](#)

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

- [Common temperature functions](#)

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

- [Common hardware trigger functions](#)

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

- [Common security functions](#)

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

- [Common time functions](#)

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

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

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

- [Common synchronisation timer functions](#)

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

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

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

- [System state management](#)

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

- [Customer option management](#)

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

- [Synchronisation management](#)

*Manage the synchronisation state of the system.*

## 2.6 Common general functions

Various utility functions, mainly to identify a remote system.

## Functions

- `int MXCommon__GetModuleType (void *_ , struct MXCommon__ByteArrayResponse *Response)`  
*This function return the type of the MSX-E Module.*
- `int MXCommon__GetHostname (void *_ , struct MXCommon__ByteArrayResponse *Response)`  
*This function return the hostname of the MSX-E Module.*
- `int MXCommon__SetHostname (struct xsd__base64Binary *bHostname, struct MXCommon__Response *Response)`  
*This function allows to set the hostname of the MSX-E Module.*
- `int MXCommon__GetClientConnections (void *_ , struct MXCommon__ByteArrayResponse *Response)`  
*This function return the client connection list.*
- `int MXCommon__Strerror (xsd__int ernum, struct MXCommon__ByteArrayResponse *Response)`  
*Call the libc strerror() on the remote device (actually this is a call to strerror\_r() ).*
- `int MXCommon__Reboot (void *_ , struct MXCommon__Response *Response)`  
*Ask the MSX-E module to reboot.*
- `int MXCommon__ResetAllIOFunctionalities (xsd__unsignedLong ulOption, struct MXCommon__Response *Response)`  
*Reset the I/O functionalities of the MSX-E system.*
- `int MXCommon__DataseverRestart (xsd__unsignedLong ulAction, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)`  
*Restart the data-server service.*
- `int MXCommon__GetEthernetLinksStates (void *_ , struct MXCommon__GetEthernetLinksStatesResponse *Response)`  
*Get MSX-E Ethernet links states.*

### 2.6.1 Function Documentation

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

##### Parameters

- [in] \_ : no input parameter
- [out] **Response** • sArray : Module type string  
 • sResponse Composed of iReturnValue and syserrno

##### Return values

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

### 2.6.1.2 int MXCommon\_\_GetHostname ( void \* \_\_, struct MXCommon\_\_ByteArrayResponse \* Response )

#### Parameters

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

#### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

### 2.6.1.3 int MXCommon\_\_SetHostname ( struct xsd\_\_base64Binary \* bHostname, struct MXCommon\_\_Response \* Response )

#### Parameters

- [in] **bHostname** : Hostname
- [out] **Response**    • iReturnValue : Return value
- 0 : success
  - -1: system error (see syserrno)
  - syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

#### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

### 2.6.1.4 int MXCommon\_\_GetClientConnections ( void \* \_\_, struct MXCommon\_\_ByteArrayResponse \* Response )

#### Parameters

- [in] *\_\_* : no input parameter
- [out] **Response**    • sArray : string containing the list of connected clients.
- sResponse Composed of iReturnValue and syserrno

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

Sample: 172.16.3.43:8989-5555 172.16.3.200:8989

#### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

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

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

#### Parameters

- [in] **errnum** : Error number
- [out] **Response** • sArray : See the description below.
- sResponse.iReturnValue : Return value
    - 0 : success
    - -1: system error (see `syserrno`).
  - sResponse.syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

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

#### NAME

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

#### SYNOPSIS

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

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

```
#define _XOPEN_SOURCE 600
#include <string.h>
```

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

#### DESCRIPTION

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

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

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

#### RETURN VALUE

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

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

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

#### ERRORS

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

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

#### CONFORMING TO

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

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

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

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

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

### Return values

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

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

### Parameters

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

### Return values

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

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

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

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

### Parameters

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

### Return values

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

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

#### Parameters

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

#### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

#### Note

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

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

#### Parameters

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

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

*sPort1: Second port informations*

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

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

## 2.7 Common temperature functions

These functions deals with the internal temperature sub-system.

### Data Structures

- struct [MXCommon\\_\\_GetModuleTemperatureValueAndStatusResponse](#)

### Functions

- int [MXCommon\\_\\_GetModuleTemperatureValueAndStatus](#) (xsd\_\_unsignedLong ulOption, struct [MXCommon\\_\\_GetModuleTemperatureValueAndStatusResponse](#) \*Response)

*Read the temperature on the module.*

- int [MXCommon\\_\\_SetModuleTemperatureWarningLevels](#) (xsd\_\_double dMinimalWarningLevel, xsd\_\_double dMaximalWarningLevel, xsd\_\_unsignedLong ulOption, struct [MXCommon\\_\\_Response](#) \*Response)

*Set the temperature warning level on the module.*

### 2.7.1 Detailed Description

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

## 2.7.2 Function Documentation

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

### Parameters

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

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

### Parameters

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

### Return values

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



## 2.8 Common hardware trigger functions

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

### Data Structures

- struct [MXCommon\\_\\_GetHardwareTriggerFilterTimeResponse](#)
- struct [MXCommon\\_\\_GetHardwareTriggerStateResponse](#)

### Functions

- int [MXCommon\\_\\_SetHardwareTriggerFilterTime](#) (xsd\_\_unsignedLong ulFilterTime, xsd\_\_unsignedLong ulOption, struct [MXCommon\\_\\_Response](#) \*Response)  
*Sets the filter time for the hardware trigger input in steps of 250 ns (max value: 65535).*
- int [MXCommon\\_\\_GetHardwareTriggerFilterTime](#) (xsd\_\_unsignedLong ulOption, struct [MXCommon\\_\\_GetHardwareTriggerFilterTimeResponse](#) \*Response)  
*Get the filter time for the hardware trigger input.*
- int [MXCommon\\_\\_GetHardwareTriggerState](#) (xsd\_\_unsignedLong ulOption, struct [MXCommon\\_\\_GetHardwareTriggerStateResponse](#) \*Response)  
*Get the hardware trigger state after the filter.*

### 2.8.1 Function Documentation

**2.8.1.1** int [MXCommon\\_\\_SetHardwareTriggerFilterTime](#) ( xsd\_\_unsignedLong *ulFilterTime*, xsd\_\_unsignedLong *ulOption*, struct [MXCommon\\_\\_Response](#) \* *Response* )

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

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

#### Parameters

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

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

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

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

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

**Return values***0* SOAP\_OK*Others* See SOAP error**2.8.1.2 int MXCommon\_\_GetHardwareTriggerFilterTime ( xsd\_\_unsignedLong ulOption, struct MXCommon\_\_GetHardwareTriggerFilterTimeResponse \* Response )**

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

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

**Parameters**

[in] *ulOption* Reserved. Set to 0

[out] *Response* Response of the system

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

**Return values***0* SOAP\_OK*Others* See SOAP error**2.8.1.3 int MXCommon\_\_GetHardwareTriggerState ( xsd\_\_unsignedLong ulOption, struct MXCommon\_\_GetHardwareTriggerStateResponse \* Response )****Parameters**

[in] *ulOption* : Reserved

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

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

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

## 2.9 Common security functions

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

### Data Structures

- struct [MXCommon\\_\\_TestCustomerIDResponse](#)

### Functions

- int [MXCommon\\_\\_SetCustomerKey](#) (struct [xsd\\_\\_base64Binary](#) \*bKey, struct [xsd\\_\\_base64Binary](#) \*bPublicKey, struct [MXCommon\\_\\_Response](#) \*Response)

*Set the Customer key.*

- int [MXCommon\\_\\_TestCustomerID](#) (void \*, struct [MXCommon\\_\\_TestCustomerIDResponse](#) \*Response)

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

### 2.9.1 Detailed Description

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

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

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

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

Detailed presentation of operations:

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

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

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

The user receives then two arrays of 16 bytes :

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

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

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

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

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

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

## 2.9.2 Function Documentation

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

### Parameters

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

### Return values

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

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

### Parameters

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

### Return values

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

## 2.10 Common time functions

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

### Data Structures

- struct [MXCommon\\_\\_GetTimeResponse](#)
- struct [MXCommon\\_\\_GetUpTimeResponse](#)

### Functions

- int [MXCommon\\_\\_SetTime](#) (xsd\_\_unsignedLong ulLowTime, xsd\_\_unsignedLong ulHighTime, struct [MXCommon\\_\\_Response](#) \*Response)  
*Set the time on the module.*
- int [MXCommon\\_\\_SysToHardwareClock](#) (void \*\_, struct [MXCommon\\_\\_Response](#) \*Response)  
*Set the hardware clock (if present) to the current system time.*
- int [MXCommon\\_\\_HardwareClockToSys](#) (void \*\_, struct [MXCommon\\_\\_Response](#) \*Response)  
*Set the system time from the hardware clock (if present).*
- int [MXCommon\\_\\_GetTime](#) (void \*\_, struct [MXCommon\\_\\_GetTimeResponse](#) \*Response)  
*Get the time on the module.*
- int [MXCommon\\_\\_GetUpTime](#) (void \*\_, struct [MXCommon\\_\\_GetUpTimeResponse](#) \*Response)  
*Ask the MSX-E module uptime (number of seconds since the last boot).*

#### 2.10.1 Detailed Description

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

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

#### 2.10.2 Function Documentation

##### 2.10.2.1 int [MXCommon\\_\\_SetTime](#) ( xsd\_\_unsignedLong ulLowTime, xsd\_\_unsignedLong ulHighTime, struct [MXCommon\\_\\_Response](#) \* Response )

##### Parameters

- [in] **ulLowTime** : Number of microseconds since the begin of the second
- [in] **ulHighTime** : Number of seconds since the Epoch (1st January,1970)
- [out] **Response**
  - sResponse.iReturnValue : Return value
  - 0 : success

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

#### Return values

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

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

##### Parameters

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

#### Return values

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

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

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

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

##### Parameters

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

#### Return values

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

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

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

#### Parameters

[in] \_\_ : No input parameter

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

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

#### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

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

#### Parameters

[in] \_\_ : no input parameter

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

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

#### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

## 2.11 Common I/O auto configuration functions

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

### Data Structures

- struct [MXCommon\\_\\_GetAutoConfigurationFileResponse](#)

## Functions

- `int MXCommon__GetAutoConfigurationFile (void *__, struct MXCommon__GetAutoConfigurationFileResponse *Response)`  
*Get the auto configuration file of the module.*
- `int MXCommon__SetAutoConfigurationFile (struct xsd__base64Binary *ByteArrayInput, xsd__unsignedLong ulEOF, struct MXCommon__Response *Response)`  
*Set the auto configuration file of the module.*
- `int MXCommon__StartAutoConfiguration (void *__, struct MXCommon__ByteArrayResponse *Response)`  
*start/Restart the auto configuration*

### 2.11.1 Detailed Description

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

This set of functions allows to:

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

### 2.11.2 Function Documentation

#### 2.11.2.1 `int MXCommon__GetAutoConfigurationFile ( void * ___, struct MXCommon__GetAutoConfigurationFileResponse * Response )`

##### Parameters

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

##### Return values

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



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

#### Parameters

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

#### Return values

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

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

#### Parameters

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

#### Return values

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

## 2.12 Common synchronisation timer functions

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

### Functions

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

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

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

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

## 2.12.1 Function Documentation

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

### Parameters

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

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

**Return values**

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

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

**Parameters**

- [in] *ulOption01* : Reserved
- [out] *Response* • `sResponse.iReturnValue` : Return value
- 0 : success
  - -1: system error (see `syserrno`)
  - -100: Start/Stop timer error
- `sResponse.syserrno` : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Sterror\(\)](#). May be ENOSYS : Function not implemented.

**Return values**

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

## 2.13 Set/Backup/Restore general system configuration

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

**Functions**

- `int MXCommon__GetConfigurationBackupFile (void ___, struct MXCommon__FileResponse *Response)`  
*Download a configuration backup file from the module.*
- `int MXCommon__ApplyConfigurationBackupFile (struct xsd__base64Binary *ByteArrayInput, xsd__unsignedLong ulEOF, struct MXCommon__Response *Response)`  
*Upload a new configuration on the module.*
- `int MXCommon__ChangePassword (struct xsd__base64Binary *PreviousUser, struct xsd__base64Binary *PreviousPassword, struct xsd__base64Binary *NewUser, struct xsd__base64Binary *NewPassword, struct MXCommon__Response *Response)`  
*Set a new id/password.*

### 2.13.1 Detailed Description

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

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

### 2.13.2 Function Documentation

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

##### Parameters

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

##### Return values

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

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

Below is an example in pseudo-C.

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

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

#### Parameters

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

#### Return values

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

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

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

The changes are immediately active.

#### Parameters

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

#### Return values

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

#### Warning

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

## 2.14 System state management

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

### Functions

- `int MXCommon__GetSubSystemState (xsd__unsignedLong SubsystemID, struct MXCommon__unsignedLongResponse *Response)`  
*Returns the current state of the specified sub-system.*
- `int MXCommon__GetSubsystemIDFromName (struct xsd__base64Binary *SubsystemName, struct MXCommon__unsignedLongResponse *Response)`  
*Returns the ID of the sub-system of symbolic name "SubsystemName".*
- `int MXCommon__GetStateIDFromName (xsd__unsignedLong SubsystemID, struct xsd__base64Binary *StateName, struct MXCommon__unsignedLongResponse *Response)`  
*Returns the ID of the state of symbolic name "StateName" of the sub-system of ID "SubsystemID".*
- `int MXCommon__GetSubsystemNameFromID (xsd__unsignedLong SubsystemID, struct MXCommon__ByteArrayResponse *Response)`  
*Returns the symbolic name of the sub-system of numerical ID "SubsystemName".*
- `int MXCommon__GetStateNameFromID (xsd__unsignedLong SubsystemID, xsd__unsignedLong StateID, struct MXCommon__ByteArrayResponse *Response)`  
*Returns the symbolic name of the state of numerical ID "StateID" of the sub-system of ID "SubsystemID".*

### 2.14.1 Detailed Description

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

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

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

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

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

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

### 2.14.2 Function Documentation

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

##### Parameters

[in] *SubsystemID* sub-system numerical ID

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

**Return values**

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

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

**Parameters**

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

**Return values**

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

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

**Parameters**

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

**Return values**

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

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

##### Parameters

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

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

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

##### Parameters

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

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

## 2.15 Customer option management

Enable to get informations about the options of the system.

### Functions

- int [MXCommon\\_\\_GetOptionInformation](#) (void \*, xsd\_\_unsignedLong ulOption01, xsd\_\_unsignedLong ulOption02, struct [MXCommon\\_\\_ByteArrayResponse](#) \*Response)
- Enables to get information about the options available on the system.*



### 2.15.1 Function Documentation

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

#### Parameters

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

#### Return values

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

## 2.16 Synchronisation management

Manage the synchronisation state of the system.

### Functions

- `int MXCommon__SetToMaster (void * __, xsd__unsignedLong ulState, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__Response *Response)`  
*Writes if the MSXE has to be always set to master The master mode (when enabled) make the system always detected as master.*
- `int MXCommon__GetSynchronizationStatus (void * __, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__unsignedLongResponse *Response)`  
*Reads the status of the synchronization for the corresponding MSXE The master mode (when enabled) make the system always detected as master.*

### 2.16.1 Function Documentation

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

#### Parameters

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

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

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

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

#### Parameters

[in] *ulOption01* Reserved. Set to 0

[in] *ulOption02* Reserved. Set to 0

[out] *Response sResponse.iReturnValue*

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

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

*ulValue* State of the supermaster mode

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

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

## 2.17 MSX-E312x Acquisition functions

Contain the acquisition information and initialisation functions.

#### Modules

- [MSX-E312x Acquisition information functions](#)

*Contain the acquisition information functions.*

- [MSX-E312x Autorefresh functions](#)

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

- [MSX-E312x Sequence functions](#)

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

## 2.18 MSX-E312x 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.*

### 2.18.1 Function Documentation

**2.18.1.1** 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

**2.18.1.2** `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

## 2.19 MSX-E312x 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.*

### 2.19.1 Detailed Description

The acquisition is initialised and the values of each channels are stored in memory on the Ethernet E/A module MSX-E312x.

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.

## 2.19.2 Function Documentation

**2.19.2.1** `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 10 to 65535 when the unit is the microsecond
  - range from min 1 to 65535 when the unit is the millisecond
  - range from min 1 to 65535 when the unit is the second
- [in] ***ulTriggerMask*** Define the source of the trigger
  - 0 : trigger disabled
  - 1 : Enable Hardware Digital Input Trigger
  - 2 : Enable Synchro Trigger
  - 3 : Enable Compare Trigger (only useful if your system has incremental counter 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 : Hardware trigger information

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

### 2.19.2.2 `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

### 2.19.2.3 `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`.



## 2.20 MSX-E312x 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.*

### 2.20.1 Detailed Description

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

There are different sequence modes:

- Certain number of sequences / continuous

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.

For examples, and more explanation concerning the sequence modes, please refer to:

- MSXE312x\_Sequence\_Software\_Start : Sequence without trigger
- MSXE312x\_Sequence\_Trigger\_One\_Shot : Sequence trigger one shot
- MSXE312x\_Sequence\_Trigger\_Only\_1\_Sequence : Sequence trigger sequence mode (only one sequence)
- MSXE312x\_Sequence\_Trigger\_Several\_Sequence : Sequence trigger sequence mode (several sequences)

## 2.20.2 Function Documentation

**2.20.2.1** `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 10 to 65535 when the unit is the microsecond
  - range from min 1 to 65535 when the unit is the millisecond
  - range from min 1 to 65535 when the unit is the second
- [in] *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
  - 3 : Enable Compare Trigger (only useful if your system has incremental counter 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 : Hardware trigger information
- 0 : No hardware trigger information required
  - 1 : Hardware trigger 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

#### 2.20.2.2 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

## 2.21 MSX-E312x Analog input functions

Contain the analog input initialisation and calibration functions.

### Modules

- [MSX-E312x Analog input initialisation functions](#)

*Contain the analog input initialisation functions.*

- [MSX-E312x Analog input calibration functions](#)

*Contain the analog input calibration functions.*

## 2.22 MSX-E312x Analog input initialisation functions

Contain the analog input initialisation functions.

### Data Structures

- struct [MSXExxxx\\_\\_AnalogInputGetChannelsTypeResponse](#)
- struct [MSXExxxx\\_\\_AnalogInputGetSamplingRateResponse](#)
- struct [MSXExxxx\\_\\_AnalogInputGetConfigurationResponse](#)

### Functions

- int [MSXExxxx\\_\\_AnalogInputGetNumberOfChannels](#) (xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)  
*Return the number of analog input channels.*
- int [MSXExxxx\\_\\_AnalogInputGetChannelsType](#) (xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_AnalogInputGetChannelsTypeResponse](#) \*Response)  
*Return the type of the analog input channels.*
- int [MSXExxxx\\_\\_AnalogInputSetChannelConfiguration](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulAcDc, xsd\_\_unsignedLong ulSingleDiff, xsd\_\_unsignedLong ulPolarity, xsd\_\_unsignedLong ulGain, xsd\_\_unsignedLong ulICP, xsd\_\_unsignedLong ulOption1, xsd\_\_unsignedLong ulOption2, struct [MSXExxxx\\_\\_Response](#) \*Response)  
*Analog input channel configuration.*
- int [MSXExxxx\\_\\_AnalogInputSetSamplingRate](#) (xsd\_\_unsignedLong ulTimeBase, xsd\_\_unsignedLong ulTimeBaseUnity, xsd\_\_unsignedLong ulSamplingRate, xsd\_\_unsignedLong ulOption1, xsd\_\_unsignedLong ulOption2, struct [MSXExxxx\\_\\_Response](#) \*Response)  
*Sets the sampling rate (or the sampling period) for the analog inputs.*
- int [MSXExxxx\\_\\_AnalogInputGetSamplingRate](#) (xsd\_\_unsignedLong ulOption1, xsd\_\_unsignedLong ulOption2, struct [MSXExxxx\\_\\_AnalogInputGetSamplingRateResponse](#) \*Response)  
*Reads the current sampling rate configuration.*

- `int MSXExxxx__AnalogInputGetConfiguration (xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MSXExxxx__AnalogInputGetConfigurationResponse *Response)`

*Get the current analog input channel configuration.*

## 2.22.1 Function Documentation

- 2.22.1.1** `int MSXExxxx__AnalogInputGetNumberOfChannels ( 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 analog input channels

### Returns

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

- 2.22.1.2** `int MSXExxxx__AnalogInputGetChannelsType ( xsd__unsignedLong ulOption1, struct MSXExxxx__AnalogInputGetChannelsTypeResponse * 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)

*ulType* : Array that contain the channels type (0 : Voltage, 1 : Current)

- *ulType* [0] : Channel 0 type

• ...

- *ulType* [7] : Channel 7 type

*ulICP* : Array that contain if ICP available (0 : not available, 1 : available)

- *ulType* [0] : Channel 0 ICP information

• ...

- *ulType* [7] : Channel 7 ICP information

### Returns

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

**2.22.1.3** `int MSXExxxx_AnalogInputSetChannelConfiguration ( xsd_unsignedLong ulChannel, xsd_unsignedLong ulAcDc, xsd_unsignedLong ulSingleDiff, xsd_unsignedLong ulPolarity, xsd_unsignedLong ulGain, xsd_unsignedLong ulICP, xsd_unsignedLong ulOption1, xsd_unsignedLong ulOption2, struct MSXExxxx_Response * Response )`

#### Parameters

[in] *ulChannel* : Channel selection (0 to 7 or 255 for all channels)

[in] *ulAcDc* : AC/DC coupling selection

- 0 : DC
- 1 : AC

[in] *ulSingleDiff* : Single/differential selection

- 0 : Single
- 1 : Differential

[in] *ulPolarity* : Polarity selection

- 0 : Bipolar
- 1 : Unipolar

[in] *ulGain* : Gain selection

- 1 : Gain 1x
- 10 : Gain 10x
- 100 : Gain 100x
- 1000 : Gain 1000x

[in] *ulICP* : ICP source

- 0 : Disable the ICP source
- 1 : Enable the ICP source

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

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

[out] *Response* :

*iReturnValue* :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Channel selection wrong
- -3: AC/DC coupling selection wrong
- -4: Single/differential selection wrong
- -5: Polarity selection wrong
- -6: Gain selection wrong
- -7: ICP selection wrong
- -8: ICP not available
- -9: Acquisition in progress. Can not change the configuration

#### Returns

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

**2.22.1.4** `int MSXExxxx_AnalogInputSetSamplingRate ( xsd__unsignedLong ulTimeBase, xsd__unsignedLong ulTimeBaseUnity, xsd__unsignedLong ulSamplingRate, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx_Response * Response )`

Sets the sampling rate (or the sampling period) for the analog inputs.

- If you select **5** for the parameter **ulTimeBaseUnity**, then this function sets the **sampling rate**, i.e. the number of samples per second.
- If you select **0** for the parameter **ulTimeBaseUnity**, then this function sets the **sampling period**, i.e. the time between two samples.

#### Parameters

[in] **ulTimeBase** Time base selection

- 0 for 1MHz
- 1 for 6.4MHz

[in] **ulTimeBaseUnity** Unity

- 0 for  $\mu$ s
- 5 for Hz

[in] **ulSamplingRate** Sampling rate or sampling period according to the selected unity

Constraints:

- **Time base of 1MHz with  $\mu$ s unity (ulTimeBase = 0 and ulTimeBaseUnity = 0)**  
The value of **ulSamplingRate** must be between 8 $\mu$ s and 65535 $\mu$ s.
- **Time base of 1MHz with Hz unity (ulTimeBase = 0 and ulTimeBaseUnity = 5)**  
The value of **ulSamplingRate** must be between 16Hz and 125000Hz.  
The result of the operation 1000000/ulSamplingRate must be an integer number (1000000 % ulSamplingRate = 0).

Typical values:

- 125000Hz
- 100000Hz
- 62500Hz
- 50000Hz
- 40000Hz
- 31250Hz
- 25000Hz
- 20000Hz
- 15625Hz
- 12500Hz
- 10000Hz
- 8000Hz
- 6250Hz
- 5000Hz
- 4000Hz
- 3125Hz
- 2500Hz
- 2000Hz
- 1600Hz



- 1250Hz
- 1000Hz
- 100Hz
- 16Hz

**Time base of 6.4MHz with  $\mu$ s unity (`ulTimeBase = 1` and `ulTimeBaseUnity = 0`)**

The value of `ulSamplingRate` must be between 10 $\mu$ s and 10235 $\mu$ s.

The result of the operation  $6.4 * ulSamplingRate$  must be an integer number.

- **Time base of 6.4MHz with Hz unity (`ulTimeBase = 1` and `ulTimeBaseUnity = 5`)**

The value of `ulSamplingRate` must be between 100Hz and 128000Hz.

The result of the operation  $64000000 / ulSamplingRate$  must be an integer number ( $64000000 \% ulSamplingRate = 0$ ).

Typical values:

- 128000Hz
- 100000Hz
- 80000Hz
- 64000Hz
- 50000Hz
- 40000Hz
- 32000Hz
- 25000Hz
- 20000Hz
- 16000Hz
- 12800Hz
- 12500Hz
- 10000Hz
- 8000Hz
- 6400Hz
- 6250Hz
- 5000Hz
- 4000Hz
- 3200Hz
- 3125Hz
- 2500Hz
- 2000Hz
- 1600Hz
- 1280Hz
- 1000Hz
- 100Hz

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

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

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

- ***iReturnValue***

- **0**: The remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Sampling rate time base selection wrong
- **-3**: Sampling rate time base unity selection wrong
- **-4**: Sampling rate selection wrong

- **-5**: Acquisition in progress. Can not change the sampling rate
- **-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

**2.22.1.5** `int MSXExxxx__AnalogInputGetSamplingRate ( xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx__AnalogInputGetSamplingRateResponse * Response )`

Reads the current sampling rate configuration.

#### Parameters

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

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

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

- ***sResponse.iReturnValue***
  - **0**: The remote function performed OK
  - **-1**: System error occurred
  - **-100**: Internal system error occurred. See value of `syserrno`
- ***sResponse.syserrno*** system error code (the value of the libc "errno" code)
- ***ulClk*** Base clock used to define the sampling rate. 0: 1MHz, 1: 6.4MHz
- ***ulDivisionFactor*** The division factor (used with the `ulClk`) that defines the sampling rate. The sampling rate is always a multiple of the selected clock.

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

**2.22.1.6** `int MSXExxxx__AnalogInputGetConfiguration ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MSXExxxx__AnalogInputGetConfigurationResponse * Response )`

#### Parameters

[in] ***ulChannel*** : Channel selection (0 to 7)

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

[out] ***Response*** :

***sResponse.iReturnValue*** :

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

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

***ulType*** : Analog input type

- 0: Voltage
- 1: Current

***ulICPAvailable*** : ICP available?

- 0: ICP source not available
- 1: ICP source available

***ulAcDc*** : AC/DC coupling selection

- 0 : DC
- 1 : AC

***ulSingleDiff*** : Single/differential selection

- 0 : Single
- 1 : Differential

***ulPolarity*** : Polarity selection

- 0 : Bipolar
- 1 : Unipolar

***ulGain*** : Gain selection

- 1 : Gain 1x
- 10 : Gain 10x
- 100 : Gain 100x
- 1000 : Gain 1000x

***ulICP*** : ICP source

- 0 : ICP source disabled
- 1 : ICP source enabled

***ulTimeBase*** : Sampling rate time base selection

- 0 for 1MHz
- 1 for 6.4MHz

***ulTimeBaseUnity*** : Sampling rate time base unity

- 0 for  $\mu$ s
- 5 for Hz

***ulSamplingRate*** : Sampling rate (or sampling period is ulTimeBaseUnity is 0)

#### Returns

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

## 2.23 MSX-E312x Analog input calibration functions

Contain the analog input calibration functions.

#### Data Structures

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

## Functions

- `int MSXExxxx__AnalogInputCalibrationStart (xsd__unsignedLong ulChannel, xsd__unsignedLong ulGainSelection, xsd__unsignedLong ulRefSource, xsd__double dRefValue, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx__Response *Response)`

*This function start the analog input calibration.*

- `int MSXExxxx__AnalogInputCalibrationGetCurrentStatus (xsd__unsignedLong ulOption1, struct MSXExxxx__AnalogInputCalibrationGetCurrentStatusResponse *Response)`

*This function return the current analog input calibration calibration status.*

- `int MSXExxxx__AnalogInputCalibrationNextStep (xsd__unsignedLong ulOption1, struct MSXExxxx__Response *Response)`

*This function start the next analog input calibration step.*

- `int MSXExxxx__AnalogInputCalibrationBreak (xsd__unsignedLong ulOption1, struct MSXExxxx__Response *Response)`

*This function break the current analog input calibration.*

### 2.23.1 Function Documentation

- 2.23.1.1** `int MSXExxxx__AnalogInputCalibrationStart ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulGainSelection, xsd__unsignedLong ulRefSource, xsd__double dRefValue, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx__Response * Response )`

#### Parameters

[in] **ulChannel** : Calibration channel selection (0 to 15).

[in] **ulGainSelection** : Selected gain (1, 10, 100,)

[in] **ulRefSource** : Calibration reference source

- 0 : External source via the first function module channel
- 1 : System internal reference voltage

[in] **dRefValue** : Reference voltage value for the calibration (V or mA)

[in] **ulOption1** : Tolerance voltage for the calibration. If not 0, then defines the value of the tolerance (the unit is micro Volt).

If the calibration does not reach the required tolerance, an error is thrown.

Leaving 0 for this value will not check the tolerance of the calibration value and will try to get the best potentiometer value

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

[out] **Response** :

**sResponse.iReturnValue** :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Channel selection error
- -3: Any acquisition already started
- -4: Any calibration already started

- -5: Gain selection error
- -6: Reference source selection error
- -7: Ref value to high/low for the selected gain
- -8: Calibration start acquisition error

**Returns**

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

**2.23.1.2** `int MSXExxxx__AnalogInputCalibrationGetCurrentStatus ( xsd__unsignedLong ulOption1, struct MSXExxxx__AnalogInputCalibrationGetCurrentStatusResponse * 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
  - -2: No calibration in progress
  - 0: No calibration in progress
  - 1: Wait to setting the reference value
  - 2: Selected reference value calibration in progress
  - 3: Calibration finished
  - 4: User break occur
  - 5: Can not calibrate the selected reference value
- ulChannel* : Current calibration channel  
*dRefValue* : Current selected calibration reference value  
*dCurrentValue* : Last measured calibration value

**Returns**

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

**2.23.1.3** `int MSXExxxx__AnalogInputCalibrationNextStep ( xsd__unsignedLong ulOption1, struct MSXExxxx__Response * Response )`

**Parameters**

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

[out] *Response* :

*iReturnValue* :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: No calibration in progress

**Returns**

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

### 2.23.1.4 int MSXExxxx\_\_AnalogInputCalibrationBreak ( xsd\_\_unsignedLong ulOption1, struct MSXExxxx\_\_Response \* Response )

#### Parameters

- [in] *ulOption1* : Reserved. Set to 0
- [out] *Response* :
- iReturnValue* :
- 0: Means the remote function performed OK
  - -1: Means an system error occurred
  - -2: No calibration in progress

#### Returns

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

## 2.24 MSX-E312x Analog output functions

Contain the analog output initialisation and read/write functions.

### Functions

- int [MSXExxxx\\_\\_AnalogOutputGetNumberOfChannels](#) (xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)  
*Return the number of analog output channels.*
- int [MSXExxxx\\_\\_AnalogOutputWrite1Value](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulOutputType, xsd\_\_unsignedLong ulPolarity, xsd\_\_unsignedLong ulTriggerMask, xsd\_\_unsignedLong ulTriggerEdgeSelection, xsd\_\_unsignedLong ulTriggerCount, xsd\_\_unsignedLong ulValue, xsd\_\_unsignedLong ulOption01, xsd\_\_unsignedLong ulOption02, xsd\_\_unsignedLong ulOption03, struct [MSXExxxx\\_\\_Response](#) \*Response)  
*Set the value of an analog output.*
- int [MSXExxxx\\_\\_AnalogOutputGetChannelValue](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)  
*Return the current value of the selected channel.*
- int [MSXExxxx\\_\\_AnalogOutputTriggerOutputs](#) (xsd\_\_unsignedLong ulOption01, struct [MSXExxxx\\_\\_Response](#) \*Response)  
*Trigger the analog outputs.*

### 2.24.1 Detailed Description

#### Conversion of a digital value into an analog value

The functions [MSXExxxx\\_\\_AnalogOutputWrite1Value](#) and [MSXExxxx\\_\\_AnalogOutputGetChannelValue](#) both take or return a digital value.

You will probably want to use Volts or Amperes, so you will have to convert that digital value into a real Volt (or ampere) value.

In bipolar mode, the digital range is from 0 to 0xffff (16bits), and corresponds to the range -10V to +10V.

In unipolar mode, the digital range is from 0 to 0x7fff (8bits), and corresponds to the range 0V to +10V.

#### From digital to analog

**In bipolar mode:**  $\text{anavalue} = ((\text{digvalue} - 32767.0) / 32767.0) * 10.0$

**In unipolar mode:**  $\text{anavalue} = (\text{digvalue} * 10.0) / 32767.0$

#### From analog to digital

**In bipolar mode:**  $\text{digvalue} = (32767.0 * \text{anavalue} / 10.0) + 32768.0;$

**In unipolar mode:**  $\text{digvalue} = 32767.0 * \text{anavalue} / 10.0;$

## 2.24.2 Function Documentation

**2.24.2.1** `int MSXExxxx__AnalogOutputGetNumberOfChannels ( 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)

*Response.ulValue* Number of available analog output channels

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

**2.24.2.2** `int MSXExxxx__AnalogOutputWrite1Value ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulOutputType, xsd__unsignedLong ulPolarity, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerEdgeSelection, xsd__unsignedLong ulTriggerCount, xsd__unsignedLong ulValue, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, xsd__unsignedLong ulOption03, struct MSXExxxx__Response * Response )`

#### Parameters

[in] *ulChannel* Index of the channel (0 to MSXExxxx\_\_AnalogOutputGetNumberOfChannels - 1)

[in] *ulOutputType* Type of the output

- **0** voltage

- **1** current
  - **2** system default
- [in] **ulPolarity** Polarity
- **0** unipolar
  - **1** bipolar
- [in] **ulTriggerMask** Trigger to use to set the outputs (Mask of bits)
- **0** no trigger: output directly updated after a call to this function
  - **Bit 0 = 1**: Hardware trigger used
  - **Bit 1 = 1**: Synchro trigger used
  - **Bit 2 = 1**: Software trigger used: output updated after a call to MSXExxxx\_\_-AnalogOutputTriggerOutputs You can give a combination of bits. Examples:
  - **1**: Only hardware trigger
  - **2**: Only synchro trigger
  - **3**: Hardware trigger and synchro trigger
  - **7**: Hardware trigger, synchro trigger and software trigger
- [in] **ulTriggerEdgeSelection** Only used if you select the hardware trigger. Defines the detected edges (Mask of bits). The value cannot be null (if hardware trigger is selected).
- **Bit 0 = 1**: Rising edges are detected
  - **Bit 1 = 1**: Falling edges are detected
- To detect both edges (rising and falling), the value is also 3.
- [in] **ulTriggerCount** Only used if you select the hardware trigger. Defines the number of hardware trigger to wait before updating the analog outputs. The value cannot be null (if hardware trigger is selected).
- [in] **ulValue** Value that you want to set (digital format)
- If unipolar, range is from 0 to 0x7fff ( from 0V to +10V)
  - If bipolar, range is from 0 to 0xffff ( from -10V to +10V)
- [in] **ulOption01** Reserved. Set to 0
- [in] **ulOption02** Reserved. Set to 0
- [in] **ulOption03** Reserved. Set to 0
- [out] **Response iReturnValue**
- **0**: Means the remote function performed OK
  - **-1**: Means an system error occurred
  - **-2**: The PLD is not working
  - **-3**: The ulChannel parameter is wrong
  - **-4**: The ulOutputType parameter is wrong
  - **-5**: The ulPolarity parameter is wrong
  - **-6**: The ulTriggerMask parameter is wrong
  - **-7**: The ulValue parameter is wrong
  - **-8**: The ulTriggerEdgeSelection parameter is wrong
  - **-9**: The ulTriggerCount parameter is wrong
  - **-10**: Timeout while setting the value
  - **-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



**2.24.2.3** `int MSXExxxx__AnalogOutputGetChannelValue ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse * Response )`

#### Parameters

[in] *ulChannel* index of the channel (0 to MSXExxxx\_\_AnalogOutputGetNumberOfChannels - 1)

[in] *ulOption1* Reserved. Set to 0

[out] *Response sResponse.iReturnValue*

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: The PLD is not working
- **-3**: Invalid parameter channel
- **-4**: Timeout while asking channel value
- **-100**: Internal system error occurred. See value of syserrno

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

*ulValue* Current value of the channel (0 corresponds to -10V, 0x7fff corresponds to 0V, 0xffff corresponds to +10V)

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

**2.24.2.4** `int MSXExxxx__AnalogOutputTriggerOutputs ( xsd__unsignedLong ulOption01, struct MSXExxxx__Response * Response )`

Set the voltage of the outputs to the last value written with MSXExxxx\_\_AnalogOutputWrite1Value

Only useful if you called MSXExxxx\_\_AnalogOutputWrite1Value with the bit 2 of the trigger mask set to 1 (software trigger)

All the output values will be updated at the same time

#### Parameters

[in] *ulOption01* Reserved. Set to 0

[out] *Response iReturnValue*

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: The PLD is not working
- **-3**: Timeout while triggering outputs
- **-100**: Internal system error occurred. See value of syserrno

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

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

## 2.25 MSX-E312x Digital I/O functions

Contain the digital I/O functions.

### Modules

- [MSX-E312x Digital I/O information, configuration functions](#)  
*Contain the digital I/O information, configuration functions.*
- [MSX-E312x Digital I/O filter functions](#)  
*Contain the digital I/O filter functions.*
- [MSX-E312x Digital I/O diagnostic functions](#)  
*Contain the digital I/O diagnostic functions.*
- [MSX-E312x Digital I/O read/write functions](#)  
*Contain the digital I/O read/write functions.*

## 2.26 MSX-E312x Digital I/O information, configuration functions

Contain the digital I/O information, configuration functions.

### Data Structures

- struct [MSXExxxx\\_DigitalIOGetPortAvailableDirectionsResponse](#)

### Functions

- int [MSXExxxx\\_DigitalIOGetNumberOfChannels](#) (xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)  
*Returns the number of digital I/O channels.*
- int [MSXExxxx\\_DigitalIOGetNumberOfPorts](#) (xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)  
*Returns the number of digital I/O ports.*
- int [MSXExxxx\\_DigitalIOGetPortAvailableDirections](#) (xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_DigitalIOGetPortAvailableDirectionsResponse](#) \*Response)  
*Returns the available directions for the selected port (input or output).*
- int [MSXExxxx\\_DigitalIOSetPortDirections](#) (xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulDirection, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_Response](#) \*Response)  
*Write the current digital I/O direction for the selected port.*
- int [MSXExxxx\\_DigitalIOGetPortDirections](#) (xsd\_\_unsignedLong ulPort, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Reads the current digital I/O direction for the selected port.*

## 2.26.1 Function Documentation

### 2.26.1.1 `int MSXExxxx_DigitalIOGetNumberOfChannels ( xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse * Response )`

Returns the number of digital I/O channels.

#### Parameters

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

[out] *Response sResponse.iReturnValue*

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-100**: Internal system error occurred. See value of `syserrno`

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

*ulValue* Number of available digital I/O channels

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

### 2.26.1.2 `int MSXExxxx_DigitalIOGetNumberOfPorts ( xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse * Response )`

Returns the number of digital I/O ports.

A port is a set of consecutive digital I/O channels, whose status can be written or read at the same time.

#### Parameters

[in] *ulOption1* Reserved. Set to 0

[out] *Response sResponse.iReturnValue*

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-100**: Internal system error occurred. See value of `syserrno`

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

*ulValue* Number of available digital I/O ports

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

**2.26.1.3** `int MSXExxxx_DigitalIOGetPortAvailableDirections ( xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MSXExxxx_DigitalIOGetPortAvailableDirectionsResponse * Response )`

Returns the available directions for the selected port (input or output).

#### Parameters

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response** *sResponse.iReturnValue*
- **0**: The remote function performed OK
  - **-1**: System error occurred
  - **-2**: The ulPort parameter is wrong
  - **-100**: Internal system error occurred. See value of syserrno
- sResponse.syserrno* system error code (the value of the libc "errno" code)
- ulInputs** Digital inputs availability. Each bit indicates if the channel can be used as an input.  
Example:
- **1**: I/O 0 of the selected port can be used as an input
  - **4**: I/O 2 of the selected port can be used as an input
  - **3**: I/Os 0 and 1 of the selected port can be used as input
- ulOutputs** Digital outputs availability. Each bit indicates if the channel can be used as an output.  
Example:
- **1**: I/O 0 of the selected port can be used as an output
  - **4**: I/O 2 of the selected port can be used as an output
  - **3**: I/Os 0 and 1 of the selected port can be used as output

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

**2.26.1.4** `int MSXExxxx_DigitalIOSetPortDirections ( xsd__unsignedLong ulPort, xsd__unsignedLong ulDirection, xsd__unsignedLong ulOption1, struct MSXExxxx_Response * Response )`

Write the current digital I/O direction for the selected port.

#### Parameters

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulDirection** Digital I/O direction. Each bit indicates if the channel is used as an input or an output. Example:
- **1**: I/O 0 of the selected port is configured as output, all the other I/Os of the selected port are configured as input

- **3:** I/Os 0 and 1 of the selected port are configured as output, all the other I/Os of the selected port are configured as input

[in] *ulOption1* Reserved. Set to 0

[out] *Response iReturnValue*

- **0:** The remote function performed OK
- **-1:** System error occurred
- **-2:** The ulPort parameter is wrong
- **-3:** The ulDirection 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

#### 2.26.1.5 int MSXExxxx\_\_DigitalIOGetPortDirections ( xsd\_\_unsignedLong ulPort, struct MSXExxxx\_\_unsignedLongResponse \* Response )

Reads the current digital I/O direction for the selected port.

#### Parameters

[in] *ulPort* Selected digital I/O port (0 to MSXExxxx\_\_DigitalIOGetNumberOfPorts)

Please read the documentation of the MSXExxxx\_\_DigitalIOGetNumberOfPorts for the description of a port.

[in] *ulOption1* Reserved. Set to 0

[out] *Response sResponse.iReturnValue*

- **0:** The remote function performed OK
- **-1:** System error occurred
- **-2:** The ulPort parameter is wrong
- **-100:** Internal system error occurred. See value of syserrno

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

*ulValue* Current digital I/O direction. Each bit indicates if the channel is used as an input or an output. Example:

- **1:** I/O 0 of the selected port is configured as output, all the other I/Os of the selected port are configured as input
- **3:** I/Os 0 and 1 of the selected port are configured as output, all the other I/Os of the selected port are configured as input

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

## 2.27 MSX-E312x Digital I/O filter functions

Contain the digital I/O filter functions.

## Data Structures

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

## Functions

- int [MSXExxxx\\_\\_DigitalIOSetInputsFilterTime](#) (xsd\_\_unsignedLong ulFilterTime, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_Response](#) \*Response)  
*Sets the filter time for the digital inputs in steps of 250 ns (max value: 16777215).*
- int [MSXExxxx\\_\\_DigitalIOEnableDisableInputsFilter](#) (xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulFilter, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_Response](#) \*Response)  
*Enables/disables the digital input filter for the selected port.*
- int [MSXExxxx\\_\\_DigitalIOGetInputsFilterConfiguration](#) (xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_DigitalIOGetInputsFilterConfigurationResponse](#) \*Response)  
*Reads the digital inputs filter configuration for the selected port.*

### 2.27.1 Function Documentation

**2.27.1.1** int [MSXExxxx\\_\\_DigitalIOSetInputsFilterTime](#) ( xsd\_\_unsignedLong *ulFilterTime*, xsd\_\_unsignedLong *ulOption1*, struct [MSXExxxx\\_\\_Response](#) \* *Response* )

Sets the filter time for the digital inputs in steps of 250 ns (max value: 16777215)

#### Parameters

[in] ***ulFilterTime*** Filter time for the digital inputs in steps of 250 ns (max value: 16777215)

- **0**: Disable the filter
- **1**: Sets the filter time to 250 ns
- **2**: Sets the filter time to 500 ns
- ...
- **16777215**: Sets the filter time to 4 s

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

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

- ***iReturnValue***
  - **0**: The remote function performed OK
  - **-1**: System error occurred
  - **-2**: 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

### 2.27.1.2 `int MSXExxxx_DigitalIOEnableDisableInputsFilter ( xsd__unsignedLong ulPort, xsd__unsignedLong ulFilter, xsd__unsignedLong ulOption1, struct MSXExxxx_Response * Response )`

Enables/disables the digital input filter for the selected port.

#### Parameters

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulFilter** Digital input filter selection. Each bit indicates if the filter is enabled on the input.  
Example:
- **1**: Filter only enabled on input 0
  - **3**: Filter enabled on inputs 0 and 1
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response iReturnValue**
- **0**: The remote function performed OK
  - **-1**: System error occurred
  - **-2**: The ulPort parameter is wrong
  - **-3**: The ulFilter parameter is wrong
  - **-4**: Any selected input is not an input or a bidirectional channel
  - **-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

### 2.27.1.3 `int MSXExxxx_DigitalIOGetInputsFilterConfiguration ( xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MSXExxxx_DigitalIOGetInputsFilterConfigurationResponse * Response )`

Reads the digital inputs filter configuration for the selected port.

#### Parameters

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response sResponse.iReturnValue**
- **0**: The remote function performed OK
  - **-1**: System error occurred
  - **-2**: The ulPort parameter is wrong
  - **-100**: Internal system error occurred. See value of syserrno

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

*ulFilterTime* Filter time value (from 1 to 16777215) 1 corresponds to 250 ns, 2 corresponds to 500 ns, ...

*ulFilter* Digital inputs filter selection. Each bit indicate the filter state for one digital input channel.

#### Return values

*0* SOAP\_OK

*Others* See SOAP error

## 2.28 MSX-E312x Digital I/O diagnostic functions

Contain the digital I/O diagnostic functions.

### Functions

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

*Get the short-circuit status of the outputs of the selected port.*

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

*Rearm the digital outputs short circuit.*

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

*Reads the current power supply status of the selected port.*

### 2.28.1 Function Documentation

**2.28.1.1** int [MSXExxxx\\_\\_DigitalIOTestOutputsShortCircuit](#) ( xsd\_\_unsignedLong *ulPort*, xsd\_\_unsignedLong *ulOption1*, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \* *Response* )

Get the short-circuit status of the outputs of the selected port.

The function returns a mask of bits (32 bits). Each bit represents the short-circuit state of an output.

If you detect a short circuit, first solve it, and then, call the [MSXExxxx\\_\\_DigitalIORearmOutputsShortCircuit](#) function.

#### Parameters

- [in] *ulPort* Selected digital I/O port (0 to [MSXExxxx\\_\\_DigitalIOGetNumberOfPorts](#))  
Please read the documentation of the [MSXExxxx\\_\\_DigitalIOGetNumberOfPorts](#) for the description of a port.
- [in] *ulOption1* Reserved. Set to 0
- [out] *Response* *sResponse.iReturnValue* :



- **0**: The remote function performed OK
- **-1**: System error occurred
- **-2**: The ulPort parameter is wrong
- **-100**: Internal system error occurred. See value of syserrno

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

*ulValue* : Digital outputs short circuit state. Each bit represents the short-circuit state of one digital output channel.

- B0 : 0: Digital I/O 0/32 no short circuit. 1: Digital I/O 0/32 short circuit
- ...
- D31 : 0: Digital I/O 31/63 no short circuit. 1: Digital I/O 31/63 short circuit

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

#### 2.28.1.2 int MSXExxxx\_DigitalIORearmOutputsShortCircuit ( xsd\_\_unsignedLong ulOption1, struct MSXExxxx\_Response \* Response )

Rearm the digital outputs short circuit.

Please use only this function if you detected a short circuit using the function MSXExxxx\_DigitalIOTestOutputsShortCircuit.

#### Parameters

[in] *ulOption1* Reserved. Set to 0

[out] *Response iReturnValue*

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-100**: Internal system error occurred. See value of syserrno

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

#### Returns

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

#### 2.28.1.3 int MSXExxxx\_DigitalIOTestOutputsPowerSupply ( xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulOption1, struct MSXExxxx\_unsignedLongResponse \* Response )

Reads the current power supply status of the selected port.

The digital outputs need an external power supply. This function checks the state of the power supply.

#### Parameters

[in] *ulPort* Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)

Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.

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

[out] **Response sResponse.iReturnValue**

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-2**: The PLD is not working
- **-3**: The ulPort parameter is wrong
- **-100**: Internal system error occurred. See value of syserrno

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

**ulValue** Current digital I/O power supply state. Each bit indicates the power supply state of the output. Example:

- **1**: No external supply voltage for the output 0
- **3**: No external supply voltage for the outputs 0 and 1

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

## 2.29 MSX-E312x Digital I/O read/write functions

Contain the digital I/O read/write functions.

### Functions

- int [MSXExxxx\\_DigitalIOReadChannel](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_unsignedLongResponse](#) \*Response)  
*Read the selected digital I/O channel.*
- int [MSXExxxx\\_DigitalIOReadPort](#) (xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_unsignedLongResponse](#) \*Response)  
*Read the selected digital I/O port.*
- int [MSXExxxx\\_DigitalIOWriteChannel](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulState, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_Response](#) \*Response)  
*Set the selected digital output channel to on or off.*
- int [MSXExxxx\\_DigitalIOWritePort](#) (xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulState, xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_Response](#) \*Response)  
*Write a value to the selected digital I/O port.*

### 2.29.1 Function Documentation

**2.29.1.1** int [MSXExxxx\\_DigitalIOReadChannel](#) ( xsd\_\_unsignedLong *ulChannel*, xsd\_\_unsignedLong *ulOption1*, struct [MSXExxxx\\_unsignedLongResponse](#) \* *Response* )

Read the selected digital I/O channel.

If the selected channel is an output, then this function returns the current output state.

**Parameters**

[in] **ulChannel** Selected digital I/O channel (0 to MSXExxxxDigitalIOGetNumberOfChannels)

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

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

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-2**: The ulChannel parameter is wrong
- **-100**: Internal system error occurred. See value of syserrno

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

**ulValue :** Digital I/O channel state

- 0: Digital I/O channel is low
- 1: Digital I/O channel is high

**Return values**

**0** SOAP\_OK

**Others** See SOAP error

### 2.29.1.2 int MSXExxxxDigitalIOReadPort ( xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulOption1, struct MSXExxxxDigitalIOResponse \* Response )

Read the selected digital I/O port.

**Parameters**

[in] **ulPort** Selected digital I/O port (0 to MSXExxxxDigitalIOGetNumberOfPorts)

Please read the documentation of the MSXExxxxDigitalIOGetNumberOfPorts for the description of a port.

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

[out] **Response sResponse.iReturnValue**

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-2**: The ulPort parameter is wrong
- **-100**: Internal system error occurred. See value of syserrno

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

**ulValue** Digital I/O state. Each bit indicates the state of one digital I/O channel.

- D0 : 0: Digital I/O 0/32 is low. 1: Digital I/O 0/32 is high
- ...
- D31 : 0: Digital I/O 31/63 is low. 1: Digital I/O 31/63 is high

**Return values**

**0** SOAP\_OK

**Others** See SOAP error

### 2.29.1.3 `int MSXExxxx_DigitalIOWriteChannel ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulState, xsd__unsignedLong ulOption1, struct MSXExxxx_Response * Response )`

Set the selected digital output channel to on or off.

#### Parameters

- [in] ***ulChannel*** Selected digital I/O channel (0 to MSXExxxx\_DigitalIOGetNumberOfChannels)
  - [in] ***ulState*** Digital I/O channel state
    - 0: Set the digital I/O output channel to low
    - 1: Set the digital I/O output channel to high
  - [in] ***ulOption1*** Reserved. Set to 0
  - [out] ***Response iReturnValue***
    - 0: The remote function performed OK
    - -1: System error occurred
    - -2: The ulChannel parameter is wrong
    - -3: The ulState parameter is wrong
    - -4: The selected digital I/O is not an output
    - -100: Internal system error occurred. See value of syserrno
- syserrno*** system error code (the value of the libc "errno" code)

#### Return values

- 0 SOAP\_OK
- Others* See SOAP error

### 2.29.1.4 `int MSXExxxx_DigitalIOWritePort ( xsd__unsignedLong ulPort, xsd__unsignedLong ulState, xsd__unsignedLong ulOption1, struct MSXExxxx_Response * Response )`

Write a value to the selected digital I/O port.

#### Parameters

- [in] ***ulPort*** Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.
  - [in] ***ulState*** Digital I/O state. Each bit set the state for one digital I/O channel (0: off, 1: on).
  - [in] ***ulOption1*** Reserved. Set to 0
  - [out] ***Response iReturnValue***
    - 0: The remote function performed OK
    - -1: System error occurred
    - -2: The ulPort parameter is wrong
    - -3: The ulState parameter is wrong
    - -4: Any digital I/O set to 1 is not an output channel
    - -100: Internal system error occurred. See value of syserrno
- syserrno*** system error code (the value of the libc "errno" code)

#### Return values

- 0 SOAP\_OK
- Others* See SOAP error

## Chapter 3

# Data Structure Documentation

### 3.1 ByteArray Struct Reference

Dynamic Array of byte - encapsulates C-type strings.

#### Data Fields

- `xsd__unsignedByte * __ptr`  
*pointer of byte*
- `int __size`  
*size of the byte array in bytes*
- `int __offset`  
*not used*

#### 3.1.1 Field Documentation

3.1.1.1 `xsd__unsignedByte* ByteArray::__ptr`

3.1.1.2 `int ByteArray::__size`

3.1.1.3 `int ByteArray::__offset`

### 3.2 DefaultResponse Struct Reference

#### Data Fields

- `xsd__int iReturnValue`  
*return value of the call :*
- `xsd__int syserrno`  
*system-error code (the value of the libc "errno" code)*

### 3.2.1 Field Documentation

#### 3.2.1.1 xsd\_\_int DefaultResponse::iReturnValue

- 0 means the remote function performed OK
- -1 means a system error 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\_\_AnalogInputCalibrationGetCurrentStatusResponse Struct Reference

#### Data Fields

- struct `DefaultResponse sResponse`

*Default return values.*

- `xsd__unsignedLong ulStatus`

*Status.*

- `xsd__unsignedLong ulChannel`

*Current calibration channel.*

- `xsd__double dRefValue`

*Current selected calibration reference value.*

- `xsd__double dCurrentValue`

*Last measured calibration value.*



### 3.6.1 Field Documentation

- 3.6.1.1 **struct DefaultResponse MSXExxxx\_\_AnalogInputCalibrationGetCurrentStatusResponse::sResponse**
- 3.6.1.2 **xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputCalibrationGetCurrentStatusResponse::ulStatus**
- 3.6.1.3 **xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputCalibrationGetCurrentStatusResponse::ulChannel**
- 3.6.1.4 **xsd\_\_double MSXExxxx\_\_AnalogInputCalibrationGetCurrentStatusResponse::dRefValue**
- 3.6.1.5 **xsd\_\_double MSXExxxx\_\_AnalogInputCalibrationGetCurrentStatusResponse::dCurrentValue**

## 3.7 MSXExxxx\_\_AnalogInputGetChannelsTypeResponse Struct Reference

### Data Fields

- struct [DefaultResponse sResponse](#)  
*Default return values.*
- [xsd\\_\\_unsignedLong ulType](#) [8]  
*The meaning of this field is defined in the related header of the function who use this type.*
- [xsd\\_\\_unsignedLong ulICP](#) [8]  
*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\_\_AnalogInputGetChannelsTypeResponse::sResponse**
- 3.7.1.2 **xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetChannelsTypeResponse::ulType[8]**
- 3.7.1.3 **xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetChannelsTypeResponse::ulICP[8]**

## 3.8 MSXExxxx\_\_AnalogInputGetConfigurationResponse Struct Reference

### Data Fields

- struct [DefaultResponse sResponse](#)  
*Default return values.*
- [xsd\\_\\_unsignedLong ulType](#)  
*The meaning of this field is defined in the related header of the function who use this type.*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 3.8.1 Field Documentation

- 3.8.1.1 struct DefaultResponse MSXExxxx\_\_AnalogInputGetConfigurationResponse::sResponse
- 3.8.1.2 xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetConfigurationResponse::ulType
- 3.8.1.3 xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetConfigurationResponse::ulICPAvailable
- 3.8.1.4 xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetConfigurationResponse::ulAcDc
- 3.8.1.5 xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetConfigurationResponse::ulSingleDiff
- 3.8.1.6 xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetConfigurationResponse::ulPolarity
- 3.8.1.7 xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetConfigurationResponse::ulGain
- 3.8.1.8 xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetConfigurationResponse::ulICP
- 3.8.1.9 xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetConfigurationResponse::ulTimeBase
- 3.8.1.10 xsd\_\_unsignedLong MSXExxxx\_\_ -  
AnalogInputGetConfigurationResponse::ulTimeBaseUnity
- 3.8.1.11 xsd\_\_unsignedLong MSXExxxx\_\_ -  
AnalogInputGetConfigurationResponse::ulSamplingRate

## 3.9 MSXExxxx\_\_AnalogInputGetSamplingRateResponse Struct Reference

### Data Fields

- struct [DefaultResponse](#) sResponse  
*Default return values.*
- [xsd\\_\\_unsignedLong](#) ulClk  
*Base clock used to define the sampling rate.*
- [xsd\\_\\_unsignedLong](#) ulDivisionFactor  
*The division factor (used with the ulClk) that defines the sampling rate.*

### 3.9.1 Field Documentation

- 3.9.1.1 struct DefaultResponse MSXExxxx\_\_AnalogInputGetSamplingRateResponse::sResponse
- 3.9.1.2 xsd\_\_unsignedLong MSXExxxx\_\_AnalogInputGetSamplingRateResponse::ulClk

0: 1MHz, 1: 6.4MHz

### 3.9.1.3 xsd\_\_unsignedLong MSXExxxx\_\_ - AnalogInputGetSamplingRateResponse::ulDivisionFactor

The sampling rate is always a multiple of the selected clock.

## 3.10 MSXExxxx\_\_DigitalIOGetInputsFilterConfigurationResponse Struct Reference

### Data Fields

- struct [DefaultResponse](#) sResponse  
*Default return values.*
- [xsd\\_\\_unsignedLong](#) ulFilterTime  
*Filter time value (from 1 to 16777215) 1 corresponds to 250 ns, 2 corresponds to 500 ns, ...*
- [xsd\\_\\_unsignedLong](#) ulFilter  
*Digital inputs filter selection.*

### 3.10.1 Field Documentation

#### 3.10.1.1 struct DefaultResponse MSXExxxx\_\_ - DigitalIOGetInputsFilterConfigurationResponse::sResponse

#### 3.10.1.2 xsd\_\_unsignedLong MSXExxxx\_\_ - DigitalIOGetInputsFilterConfigurationResponse::ulFilterTime

#### 3.10.1.3 xsd\_\_unsignedLong MSXExxxx\_\_ - DigitalIOGetInputsFilterConfigurationResponse::ulFilter

Each bit indicate the filter state for one digital input channel.

- D0 : 0: Digital I/O 0/32 filter disabled. 1: Digital I/O 0/32 filter enabled
- ...
- D31 : 0: Digital I/O 31/63 filter disabled. 1: Digital I/O 31/63 filter enabled

## 3.11 MSXExxxx\_\_DigitalIOGetPortAvailableDirectionsResponse Struct Reference

### Data Fields

- struct [DefaultResponse](#) sResponse  
*Default return values.*
- [xsd\\_\\_unsignedLong](#) ulInputs

*Digital input directions availability.*

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

*Digital output directions availability.*

### 3.11.1 Field Documentation

#### 3.11.1.1 struct DefaultResponse MSXExxxx\_\_ - DigitalIOGetPortAvailableDirectionsResponse::sResponse

#### 3.11.1.2 xsd\_\_unsignedLong MSXExxxx\_\_ - DigitalIOGetPortAvailableDirectionsResponse::ulInputs

Each bit indicates the availability for one channel.

- D0 : 0: Digital I/O 0/32 can not be used for input. 1: Digital I/O 0/32 can be used for input
- ...
- D31 : 0: Digital I/O 31/63 can not be used for input. 1: Digital I/O 31/63 can be used for input

#### 3.11.1.3 xsd\_\_unsignedLong MSXExxxx\_\_ - DigitalIOGetPortAvailableDirectionsResponse::ulOutputs

Each bit indicates the availability for one channel.

- D0 : 0: Digital I/O 0/32 can not be used for output. 1: Digital I/O 0/32 can be used for output
- ...
- D31 : 0: Digital I/O 31/63 can not be used for output. 1: Digital I/O 31/63 can be used for output

## 3.12 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.*

### 3.12.1 Field Documentation

3.12.1.1 struct DefaultResponse MSXExxxx\_\_FileResponse::sResponse

3.12.1.2 struct ByteArray MSXExxxx\_\_FileResponse::sArray

3.12.1.3 xsd\_\_unsignedLong MSXExxxx\_\_FileResponse::ulEOF

## 3.13 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).*

### 3.13.1 Field Documentation

3.13.1.1 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

3.13.1.2 xsd\_\_int MSXExxxx\_\_Response::syserrno

## 3.14 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.*

### 3.14.1 Field Documentation

3.14.1.1 struct [DefaultResponse](#) MSXExxxx\_\_unsignedLongResponse::sResponse

3.14.1.2 xsd\_\_unsignedLong MSXExxxx\_\_unsignedLongResponse::ulValue

## 3.15 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*

### 3.15.1 Field Documentation

3.15.1.1 struct [DefaultResponse](#) MSXExxxx\_\_unsignedLongTimeStampResponse::sResponse

3.15.1.2 xsd\_\_unsignedLong MSXExxxx\_\_unsignedLongTimeStampResponse::ulValue

3.15.1.3 xsd\_\_unsignedLong MSXExxxx\_\_unsignedLongTimeStampResponse::ulTimeStampLow

3.15.1.4 xsd\_\_unsignedLong MSXExxxx\_\_-  
unsignedLongTimeStampResponse::ulTimeStampHigh

## 3.16 MXCommon\_\_ByteArrayResponse Struct Reference

Response containing a C-type string.

### Data Fields

- struct [DefaultResponse](#) sResponse  
*Default return values.*
- struct [ByteArray](#) sArray  
*Dynamic Array of byte - encapsulates C-type strings.*

### 3.16.1 Field Documentation

3.16.1.1 struct `DefaultResponse MXCommon__ByteArrayResponse::sResponse`

3.16.1.2 struct `ByteArray MXCommon__ByteArrayResponse::sArray`

## 3.17 MXCommon\_\_FileResponse Struct Reference

Response containing a chunk of a file.

### Data Fields

- struct `DefaultResponse sResponse`  
*return values.*
- struct `ByteArray sArray`  
*Dynamic Array of byte.*
- `xsd__unsignedLong ulEOF`  
*flag indicating end of file.*

### 3.17.1 Field Documentation

3.17.1.1 struct `DefaultResponse MXCommon__FileResponse::sResponse`

3.17.1.2 struct `ByteArray MXCommon__FileResponse::sArray`

3.17.1.3 `xsd__unsignedLong MXCommon__FileResponse::ulEOF`

## 3.18 MXCommon\_\_GetAutoConfigurationFileResponse Struct Reference

### Data Fields

- struct `DefaultResponse sResponse`  
*Default return values.*
- struct `ByteArray bArray`  
*Array of byte of the file.*
- `xsd__unsignedLong ulEOF`  
*End of file flag.*



### 3.18.1 Field Documentation

3.18.1.1 struct DefaultResponse MXCommon\_\_GetAutoConfigurationFileResponse::sResponse

3.18.1.2 struct ByteArray MXCommon\_\_GetAutoConfigurationFileResponse::bArray

3.18.1.3 xsd\_\_unsignedLong MXCommon\_\_GetAutoConfigurationFileResponse::ulEOF

## 3.19 MXCommon\_\_GetEthernetLinksStatesResponse Struct Reference

### Data Fields

- struct [DefaultResponse](#) sResponse  
*Default return values.*
- struct [sGetEthernetLinksStatesPort](#) sPort0
- struct [sGetEthernetLinksStatesPort](#) sPort1

### 3.19.1 Field Documentation

3.19.1.1 struct DefaultResponse MXCommon\_\_GetEthernetLinksStatesResponse::sResponse

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

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

## 3.20 MXCommon\_\_GetHardwareTriggerFilterTimeResponse Struct Reference

### Data Fields

- struct [DefaultResponse](#) sResponse  
*Default return values.*
- [xsd\\_\\_unsignedLong](#) ulFilterTime  
*Hardware filter time (step of 250ns).*
- [xsd\\_\\_unsignedLong](#) ulInfo01  
*Reserved.*
- [xsd\\_\\_unsignedLong](#) ulInfo02  
*Reserved.*

### 3.20.1 Field Documentation

- 3.20.1.1 struct `DefaultResponse MXCommon__GetHardwareTriggerFilterTimeResponse::sResponse`
- 3.20.1.2 `xsd__unsignedLong MXCommon__GetHardwareTriggerFilterTimeResponse::ulFilterTime`
- 3.20.1.3 `xsd__unsignedLong MXCommon__GetHardwareTriggerFilterTimeResponse::ulInfo01`
- 3.20.1.4 `xsd__unsignedLong MXCommon__GetHardwareTriggerFilterTimeResponse::ulInfo02`

## 3.21 MXCommon\_\_GetHardwareTriggerStateResponse Struct Reference

### Data Fields

- struct `DefaultResponse sResponse`  
*Default return values.*
- `xsd__unsignedLong ulState`  
*0 : Trigger input is low / 1 : Trigger input is high*
- `xsd__unsignedLong ulInfo01`  
*Reserved.*
- `xsd__unsignedLong ulInfo02`  
*Reserved.*

### 3.21.1 Field Documentation

- 3.21.1.1 struct `DefaultResponse MXCommon__GetHardwareTriggerStateResponse::sResponse`
- 3.21.1.2 `xsd__unsignedLong MXCommon__GetHardwareTriggerStateResponse::ulState`
- 3.21.1.3 `xsd__unsignedLong MXCommon__GetHardwareTriggerStateResponse::ulInfo01`
- 3.21.1.4 `xsd__unsignedLong MXCommon__GetHardwareTriggerStateResponse::ulInfo02`

## 3.22 MXCommon\_\_GetModuleTemperatureValueAndStatusResponse Struct Reference

### Data Fields

- struct `DefaultResponse sResponse`  
*Default return value.*
- `xsd__double dTemperatureValue`

*Temperature value.*

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

*Temperature status.*

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

*Reserved.*

### 3.22.1 Field Documentation

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

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

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

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

## 3.23 MXCommon\_\_GetTimeResponse Struct Reference

### Data Fields

- `struct DefaultResponse sResponse`

*Default return values.*

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

*Number of microseconds since the begin of the second.*

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

*Number of seconds since the Epoch (1st January,1970).*

### 3.23.1 Field Documentation

**3.23.1.1** `struct DefaultResponse MXCommon__GetTimeResponse::sResponse`

**3.23.1.2** `xsd__unsignedLong MXCommon__GetTimeResponse::ulLowTime`

**3.23.1.3** `xsd__unsignedLong MXCommon__GetTimeResponse::ulHighTime`

## 3.24 MXCommon\_\_GetUpTimeResponse Struct Reference

### Data Fields

- `struct DefaultResponse sResponse`

*Default return value.*

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

*Reserved.*

### 3.24.1 Field Documentation

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

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

## 3.25 MXCommon\_\_Response Struct Reference

contains return values

### Data Fields

- [xsd\\_\\_int iReturnValue](#)

*return value of the call :*

- 0 success
- -1 a system error occurred, the meaning of other values is function dependent and should be defined in the related header.

- [xsd\\_\\_int syserrno](#)

*system-error code (the value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#)).*

### 3.25.1 Field Documentation

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

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

## 3.26 MXCommon\_\_TestCustomerIDResponse Struct Reference

### Data Fields

- [struct DefaultResponse sResponse](#)

*Default return values.*

- [struct ByteArray bValueArray](#)

*non encrypted value*

- [struct ByteArray bCryptedValueArray](#)

*encrypted value*

### 3.26.1 Field Documentation

3.26.1.1 struct DefaultResponse MXCommon\_\_TestCustomerIDResponse::sResponse

3.26.1.2 struct ByteArray MXCommon\_\_TestCustomerIDResponse::bValueArray

3.26.1.3 struct ByteArray MXCommon\_\_TestCustomerIDResponse::bCryptedValueArray

## 3.27 MXCommon\_\_unsignedLongResponse Struct Reference

Response containing a numerical value (ex: return code).

### Data Fields

- struct [DefaultResponse](#) sResponse

*Default return values.*

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

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

### 3.27.1 Field Documentation

3.27.1.1 struct DefaultResponse MXCommon\_\_unsignedLongResponse::sResponse

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

## 3.28 sGetEthernetLinksStatesPort Struct Reference

### Data Fields

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

### 3.28.1 Field Documentation

3.28.1.1 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulState`

3.28.1.2 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulSpeed`

3.28.1.3 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulDuplex`

3.28.1.4 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo1`

3.28.1.5 `xsd__unsignedLong sGetEthernetLinksStatesPort::ulInfo2`

## 3.29 UnsignedLongArray Struct Reference

Dynamic Array of unsigned long.

### Data Fields

- `xsd__unsignedLong * __ptr`  
*pointer of unsigned Long*
- `int __size`  
*size of the unsigned Long array in Bytes*
- `int __offset`  
*not used*

### 3.29.1 Field Documentation

3.29.1.1 `xsd__unsignedLong* UnsignedLongArray::__ptr`

3.29.1.2 `int UnsignedLongArray::__size`

3.29.1.3 `int UnsignedLongArray::__offset`

## 3.30 UnsignedShortArray Struct Reference

Dynamic Array of unsigned short.

### Data Fields

- `xsd__unsignedShort * __ptr`  
*pointer of unsigned short*
- `int __size`  
*size of the unsigned short array in Bytes*

- int [\\_\\_offset](#)  
*not used*

### 3.30.1 Field Documentation

3.30.1.1 `xsd__unsignedShort* UnsignedShortArray::__ptr`

3.30.1.2 `int UnsignedShortArray::__size`

3.30.1.3 `int UnsignedShortArray::__offset`

## 3.31 xsd\_\_base64Binary Struct Reference

Dynamic Array of byte for input use.

### Data Fields

- unsigned char \* [\\_\\_ptr](#)  
*pointer of byte*
- int [\\_\\_size](#)  
*size of the byte array*

### 3.31.1 Field Documentation

3.31.1.1 `unsigned char* xsd__base64Binary::__ptr`

3.31.1.2 `int xsd__base64Binary::__size`





# Chapter 4

## File Documentation

### 4.1 MSXE312x\_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\\_\\_AnalogInputGetChannelsTypeResponse](#)
- struct [MSXExxxx\\_\\_AnalogInputGetSamplingRateResponse](#)
- struct [MSXExxxx\\_\\_AnalogInputGetConfigurationResponse](#)
- struct [MSXExxxx\\_\\_AnalogInputCalibrationGetCurrentStatusResponse](#)
- struct [MSXExxxx\\_\\_DigitalIOGetPortAvailableDirectionsResponse](#)
- struct [MSXExxxx\\_\\_DigitalIOGetInputsFilterConfigurationResponse](#)

## Typedefs

- typedef char \* [xsd\\_\\_string](#)  
*encode xsd\_\_string value as the xsd:string schema type*
- typedef char [xsd\\_\\_char](#)  
*encode xsd\_\_string value as the xsd:char schema type*
- typedef float [xsd\\_\\_float](#)  
*encode xsd\_\_float value as the xsd:float schema type*
- typedef double [xsd\\_\\_double](#)  
*encode xsd\_\_double value as the xsd:double schema type*
- typedef int [xsd\\_\\_int](#)  
*encode xsd\_\_int value as the xsd:int schema type*
- typedef long [xsd\\_\\_long](#)  
*encode xsd\_\_long value as the xsd:long schema type*
- typedef unsigned char [xsd\\_\\_unsignedByte](#)  
*encode xsd\_\_unsignedByte value as the xsd:unsignedByte schema type*
- typedef unsigned int [xsd\\_\\_unsignedInt](#)  
*encode xsd\_\_unsignedInt value as the xsd:unsignedInt schema type*
- typedef unsigned short int [xsd\\_\\_unsignedShort](#)  
*encode xsd\_\_unsignedShort value as the xsd:unsignedShort schema type*
- typedef unsigned long [xsd\\_\\_unsignedLong](#)  
*encode xsd\_\_unsignedLong value as the xsd:unsignedLong schema type*

## Functions

- `int MXCommon__GetModuleType (void *__, struct MXCommon__ByteArrayResponse *Response)`  
*This function return the type of the MSX-E Module.*
- `int MXCommon__GetHostname (void *__, struct MXCommon__ByteArrayResponse *Response)`  
*This function return the hostname of the MSX-E Module.*
- `int MXCommon__SetHostname (struct xsd__base64Binary *bHostname, struct MXCommon__Response *Response)`  
*This function allows to set the hostname of the MSX-E Module.*
- `int MXCommon__GetClientConnections (void *__, struct MXCommon__ByteArrayResponse *Response)`  
*This function return the client connection list.*
- `int MXCommon__Strerror (xsd__int errnum, struct MXCommon__ByteArrayResponse *Response)`  
*Call the libc strerror() on the remote device (actually this is a call to strerror\_r() ).*
- `int MXCommon__Reboot (void *__, struct MXCommon__Response *Response)`  
*Ask the MSX-E module to reboot.*
- `int MXCommon__ResetAllIOFunctionalities (xsd__unsignedLong ulOption, struct MXCommon__Response *Response)`  
*Reset the I/O functionalities of the MSX-E system.*
- `int MXCommon__DataseverRestart (xsd__unsignedLong ulAction, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)`  
*Restart the data-server service.*
- `int MXCommon__GetEthernetLinksStates (void *__, struct MXCommon__GetEthernetLinksStatesResponse *Response)`  
*Get MSX-E Ethernet links states.*
- `int MXCommon__GetModuleTemperatureValueAndStatus (xsd__unsignedLong ulOption, struct MXCommon__GetModuleTemperatureValueAndStatusResponse *Response)`  
*Read the temperature on the module.*
- `int MXCommon__SetModuleTemperatureWarningLevels (xsd__double dMinimalWarningLevel, xsd__double dMaximalWarningLevel, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)`  
*Set the temperature warning level on the module.*
- `int MXCommon__SetHardwareTriggerFilterTime (xsd__unsignedLong ulFilterTime, xsd__unsignedLong ulOption, struct MXCommon__Response *Response)`  
*Sets the filter time for the hardware trigger input in steps of 250 ns (max value: 65535).*
- `int MXCommon__GetHardwareTriggerFilterTime (xsd__unsignedLong ulOption, struct MXCommon__GetHardwareTriggerFilterTimeResponse *Response)`

*Get the filter time for the hardware trigger input.*

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

*Get the hardware trigger state after the filter.*

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

*Set the Customer key.*

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

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

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

*Set the time on the module.*

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

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

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

*Set the system time from the hardware clock (if present).*

- `int MXCommon__GetTime (void *_ , struct MXCommon__GetTimeResponse *Response)`

*Get the time on the module.*

- `int MXCommon__GetUpTime (void *_ , struct MXCommon__GetUpTimeResponse *Response)`

*Ask the MSX-E module uptime (number of seconds since the last boot).*

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

*Get the auto configuration file of the module.*

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

*Set the auto configuration file of the module.*

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

*start/Restart the auto configuration*

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

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

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

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

- `int MXCommon__GetConfigurationBackupFile (void ___, struct MXCommon__FileResponse *Response)`  
*Download a configuration backup file from the module.*
- `int MXCommon__ApplyConfigurationBackupFile (struct xsd__base64Binary *ByteArrayInput, xsd__unsignedLong ulEOF, struct MXCommon__Response *Response)`  
*Upload a new configuration on the module.*
- `int MXCommon__ChangePassword (struct xsd__base64Binary *PreviousUser, struct xsd__base64Binary *PreviousPassword, struct xsd__base64Binary *NewUser, struct xsd__base64Binary *NewPassword, struct MXCommon__Response *Response)`  
*Set a new id/password.*
- `int MXCommon__GetSubSystemState (xsd__unsignedLong SubsystemID, struct MXCommon__unsignedLongResponse *Response)`  
*Returns the current state of the specified sub-system.*
- `int MXCommon__GetSubsystemIDFromName (struct xsd__base64Binary *SubsystemName, struct MXCommon__unsignedLongResponse *Response)`  
*Returns the ID of the sub-system of symbolic name "SubsystemName".*
- `int MXCommon__GetStateIDFromName (xsd__unsignedLong SubsystemID, struct xsd__base64Binary *StateName, struct MXCommon__unsignedLongResponse *Response)`  
*Returns the ID of the state of symbolic name "StateName" of the sub-system of ID "SubsystemID".*
- `int MXCommon__GetSubsystemNameFromID (xsd__unsignedLong SubsystemID, struct MXCommon__ByteArrayResponse *Response)`  
*Returns the symbolic name of the sub-system of numerical ID "SubsystemName".*
- `int MXCommon__GetStateNameFromID (xsd__unsignedLong SubsystemID, xsd__unsignedLong StateID, struct MXCommon__ByteArrayResponse *Response)`  
*Returns the symbolic name of the state of numerical ID "StateID" of the sub-system of ID "SubsystemID".*
- `int MXCommon__GetOptionInformation (void ___, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__ByteArrayResponse *Response)`  
*Enables to get information about the options available on the system.*
- `int MXCommon__SetToMaster (void ___, xsd__unsignedLong ulState, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__Response *Response)`  
*Writes if the MSXE has to be always set to master The master mode (when enabled) make the system always detected as master.*
- `int MXCommon__GetSynchronizationStatus (void ___, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, struct MXCommon__unsignedLongResponse *Response)`  
*Reads the status of the synchronization for the corresponding MSXE The master mode (when enabled) make the system always detected as master.*
- `int 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\\_\\_AnalogInputGetNumberOfChannels](#) (xsd\_\_unsignedLong ulOption1, struct [MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Return the number of analog input channels.*

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

*Return the type of the analog input channels.*

- int [MSXExxxx\\_\\_AnalogInputSetChannelConfiguration](#) (xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulAcDc, xsd\_\_unsignedLong ulSingleDiff, xsd\_\_unsignedLong ulPolarity, xsd\_\_unsignedLong ulGain, xsd\_\_unsignedLong ulICP, xsd\_\_unsignedLong ulOption1, xsd\_\_unsignedLong ulOption2, struct [MSXExxxx\\_\\_Response](#) \*Response)

*Analog input channel configuration.*

- `int MSXExxxx__AnalogInputSetSamplingRate (xsd__unsignedLong ulTimeBase, xsd__unsignedLong ulTimeBaseUnity, xsd__unsignedLong ulSamplingRate, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx__Response *Response)`  
*Sets the sampling rate (or the sampling period) for the analog inputs.*
- `int MSXExxxx__AnalogInputGetSamplingRate (xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx__AnalogInputGetSamplingRateResponse *Response)`  
*Reads the current sampling rate configuration.*
- `int MSXExxxx__AnalogInputGetConfiguration (xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MSXExxxx__AnalogInputGetConfigurationResponse *Response)`  
*Get the current analog input channel configuration.*
- `int MSXExxxx__AnalogInputCalibrationStart (xsd__unsignedLong ulChannel, xsd__unsignedLong ulGainSelection, xsd__unsignedLong ulRefSource, xsd__double dRefValue, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx__Response *Response)`  
*This function start the analog input calibration.*
- `int MSXExxxx__AnalogInputCalibrationGetCurrentStatus (xsd__unsignedLong ulOption1, struct MSXExxxx__AnalogInputCalibrationGetCurrentStatusResponse *Response)`  
*This function return the current analog input calibration calibration status.*
- `int MSXExxxx__AnalogInputCalibrationNextStep (xsd__unsignedLong ulOption1, struct MSXExxxx__Response *Response)`  
*This function start the next analog input calibration step.*
- `int MSXExxxx__AnalogInputCalibrationBreak (xsd__unsignedLong ulOption1, struct MSXExxxx__Response *Response)`  
*This function break the current analog input calibration.*
- `int MSXExxxx__AnalogOutputGetNumberOfChannels (xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse *Response)`  
*Return the number of analog output channels.*
- `int MSXExxxx__AnalogOutputWriteIValue (xsd__unsignedLong ulChannel, xsd__unsignedLong ulOutputType, xsd__unsignedLong ulPolarity, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerEdgeSelection, xsd__unsignedLong ulTriggerCount, xsd__unsignedLong ulValue, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, xsd__unsignedLong ulOption03, struct MSXExxxx__Response *Response)`  
*Set the value of an analog output.*
- `int MSXExxxx__AnalogOutputGetChannelValue (xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse *Response)`  
*Return the current value of the selected channel.*
- `int MSXExxxx__AnalogOutputTriggerOutputs (xsd__unsignedLong ulOption01, struct MSXExxxx__Response *Response)`  
*Trigger the analog outputs.*

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

*Returns the number of digital I/O channels.*
- [int MSXExxxx\\_DigitalIOGetNumberOfPorts](#) ([xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Returns the number of digital I/O ports.*
- [int MSXExxxx\\_DigitalIOGetPortAvailableDirections](#) ([xsd\\_\\_unsignedLong](#) ulPort, [xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_DigitalIOGetPortAvailableDirectionsResponse](#) \*Response)

*Returns the available directions for the selected port (input or output).*
- [int MSXExxxx\\_DigitalIOSetPortDirections](#) ([xsd\\_\\_unsignedLong](#) ulPort, [xsd\\_\\_unsignedLong](#) ulDirection, [xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_\\_Response](#) \*Response)

*Write the current digital I/O direction for the selected port.*
- [int MSXExxxx\\_DigitalIOGetPortDirections](#) ([xsd\\_\\_unsignedLong](#) ulPort, [struct MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Reads the current digital I/O direction for the selected port.*
- [int MSXExxxx\\_DigitalIOSetInputsFilterTime](#) ([xsd\\_\\_unsignedLong](#) ulFilterTime, [xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_\\_Response](#) \*Response)

*Sets the filter time for the digital inputs in steps of 250 ns (max value: 16777215).*
- [int MSXExxxx\\_DigitalIOEnableDisableInputsFilter](#) ([xsd\\_\\_unsignedLong](#) ulPort, [xsd\\_\\_unsignedLong](#) ulFilter, [xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_\\_Response](#) \*Response)

*Enables/disables the digital input filter for the selected port.*
- [int MSXExxxx\\_DigitalIOGetInputsFilterConfiguration](#) ([xsd\\_\\_unsignedLong](#) ulPort, [xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_DigitalIOGetInputsFilterConfigurationResponse](#) \*Response)

*Reads the digital inputs filter configuration for the selected port.*
- [int MSXExxxx\\_DigitalIOTestOutputsShortCircuit](#) ([xsd\\_\\_unsignedLong](#) ulPort, [xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Get the short-circuit status of the outputs of the selected port.*
- [int MSXExxxx\\_DigitalIORearmOutputsShortCircuit](#) ([xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_\\_Response](#) \*Response)

*Rearm the digital outputs short circuit.*
- [int MSXExxxx\\_DigitalIOTestOutputsPowerSupply](#) ([xsd\\_\\_unsignedLong](#) ulPort, [xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Reads the current power supply status of the selected port.*
- [int MSXExxxx\\_DigitalIOReadChannel](#) ([xsd\\_\\_unsignedLong](#) ulChannel, [xsd\\_\\_unsignedLong](#) ulOption1, [struct MSXExxxx\\_\\_unsignedLongResponse](#) \*Response)

*Read the selected digital I/O channel.*



- int `MSXExxxx__DigitalIOReadPort` (`xsd__unsignedLong` ulPort, `xsd__unsignedLong` ulOption1, struct `MSXExxxx__unsignedLongResponse` \*Response)

*Read the selected digital I/O port.*

- int `MSXExxxx__DigitalIOWriteChannel` (`xsd__unsignedLong` ulChannel, `xsd__unsignedLong` ulState, `xsd__unsignedLong` ulOption1, struct `MSXExxxx__Response` \*Response)

*Set the selected digital output channel to on or off.*

- int `MSXExxxx__DigitalIOWritePort` (`xsd__unsignedLong` ulPort, `xsd__unsignedLong` ulState, `xsd__unsignedLong` ulOption1, struct `MSXExxxx__Response` \*Response)

*Write a value to the selected digital I/O port.*

### 4.1.1 Typedef Documentation

4.1.1.1 typedef char\* `xsd__string`

4.1.1.2 typedef char `xsd__char`

4.1.1.3 typedef float `xsd__float`

4.1.1.4 typedef double `xsd__double`

4.1.1.5 typedef int `xsd__int`

4.1.1.6 typedef long `xsd__long`

4.1.1.7 typedef unsigned char `xsd__unsignedByte`

4.1.1.8 typedef unsigned int `xsd__unsignedInt`

4.1.1.9 typedef unsigned short int `xsd__unsignedShort`

4.1.1.10 typedef unsigned long `xsd__unsignedLong`

### 4.1.2 Function Documentation

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

#### Parameters

[in] `_` : no input parameter

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

#### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

#### 4.1.2.2 int MXCommon\_\_GetHostname ( void \* \_\_, struct MXCommon\_\_ByteArrayResponse \* Response )

##### Parameters

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

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

#### 4.1.2.3 int MXCommon\_\_SetHostname ( struct xsd\_\_base64Binary \* bHostname, struct MXCommon\_\_Response \* Response )

##### Parameters

- [in] *bHostname* : Hostname
- [out] **Response** • iReturnValue : Return value
- 0 : success
  - -1: system error (see syserrno)
  - syserrno : System-error code. The value of the libc "errno" code, see [MXCommon\\_\\_Strerror\(\)](#).

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

#### 4.1.2.4 int MXCommon\_\_GetClientConnections ( void \* \_\_, struct MXCommon\_\_ByteArrayResponse \* Response )

##### Parameters

- [in] *\_\_* : no input parameter
- [out] **Response** • sArray : string containing the list of connected clients.
- sResponse Composed of iReturnValue and syserrno

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

Sample: 172.16.3.43:8989-5555 172.16.3.200:8989

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

#### 4.1.2.5 int MXCommon\_\_Strerror ( xsd\_\_int *errnum*, struct MXCommon\_\_ByteArrayResponse \* *Response* )

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

##### Parameters

[in] **errnum** : Error number

[out] **Response** • sArray : See the description below.

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

STRERROR(3)  
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\_TOOLOW = 1 : Temperature too low !
  - TEMPERATURE\_LOW = 2 : Temperature under the min warning value
  - TEMPERATURE\_NOMINAL = 3 : Temperature in the nominal range
  - TEMPERATURE\_HIGH = 4 : Temperature over the max warning value
  - TEMPERATURE\_TOOHIGH = 5 : Temperature too high !

- ulInfo : Reserved

**Return values**

**SOAP\_OK** SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

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

#### Return values

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

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

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

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

#### Parameters

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

#### Return values

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

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

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

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

##### Parameters

[in] *ulOption* Reserved. Set to 0

[out] *Response* Response of the system

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

##### Return values

0 SOAP\_OK

*Others* See SOAP error

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

##### Parameters

[in] *ulOption* : Reserved

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

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

##### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error



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

#### Parameters

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

#### Return values

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

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

#### Parameters

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

#### Return values

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

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

#### Parameters

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

**Return values**

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

[in] \_ No input parameter

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

– 0 : success

– -1: system error (see syserrno)

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

**Return values**

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

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

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

**Parameters**

[in] \_ No input parameter

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

– 0 : success

– -1: system error (see syserrno)

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

**Return values**

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

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

##### Parameters

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

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

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

##### Parameters

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

##### Return values

**SOAP\_OK** SOAP call success

**otherwise** SOAP protocol error

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

##### Parameters

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

- `ulEOF` : End of file flag

#### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

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

[in] *ulEOF* : End of file flag

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

– 0 : success

– -1: system error (see `syserrno`)

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

#### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

[in] `_` : No input parameter

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

– 0 : success

– -1: system error (see `syserrno`)

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

#### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

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

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

#### Return values

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

***otherwise*** SOAP protocol error

#### 4.1.2.26 int MXCommon\_\_StopAndReleaseSynchroTimer ( xsd\_\_unsignedLong ulOption01, struct MXCommon\_\_Response \* Response )

##### Parameters

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

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

#### Return values

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

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

##### Parameters

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

#### Return values

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

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

Below is an example in pseudo-C.

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

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

#### Parameters

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

#### Return values

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

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

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

The changes are immediately active.

#### Parameters

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

#### Return values

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

#### Warning

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

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

##### Parameters

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

##### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

##### Parameters

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

##### Return values

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

##### Parameters

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



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

**Return values**

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

**Parameters**

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

**Return values**

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

**Parameters**

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

**Return values**

*SOAP\_OK* SOAP call success

*otherwise* SOAP protocol error

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

#### Parameters

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

#### Return values

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

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

#### Parameters

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

#### Return values

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

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

#### Parameters

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

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

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

*ulValue* State of the supermaster mode

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

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

**4.1.2.38** `int 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 10 to 65535 when the unit is the microsecond
  - range from min 1 to 65535 when the unit is the millisecond
  - range from min 1 to 65535 when the unit is the second
- [in] ***ulTriggerMask*** Define the source of the trigger
- 0 : trigger disabled
  - 1 : Enable Hardware Digital Input Trigger
  - 2 : Enable Synchro Trigger
  - 3 : Enable Compare Trigger (only useful if your system has incremental counter 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
- $\langle \rangle$  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 : Hardware trigger information

- 0 : No hardware trigger information required
- 1 : Hardware trigger 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 10 to 65535 when the unit is the microsecond
- range from min 1 to 65535 when the unit is the millisecond
- range from min 1 to 65535 when the unit is the second

[in] *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
  - 3 : Enable Compare Trigger (only useful if your system has incremental counter 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 : Hardware trigger information
    - 0 : No hardware trigger information required
    - 1 : Hardware trigger 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\_\_AnalogInputGetNumberOfChannels ( 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 analog input channels

#### Returns

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

#### 4.1.2.46 int MSXExxxx\_\_AnalogInputGetChannelsType ( xsd\_\_unsignedLong *ulOption1*, struct MSXExxxx\_\_AnalogInputGetChannelsTypeResponse \* *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)

*ulType* : Array that contain the channels type (0 : Voltage, 1 : Current)

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

- *ulType* [7] : Channel 7 type

*ulICP* : Array that contain if ICP available (0 : not available, 1 : available)

- *ulType* [0] : Channel 0 ICP information
- ...
- *ulType* [7] : Channel 7 ICP information

#### Returns

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

**4.1.2.47** `int MSXExxxx_AnalogInputSetChannelConfiguration ( xsd_unsignedLong ulChannel, xsd_unsignedLong ulAcDc, xsd_unsignedLong ulSingleDiff, xsd_unsignedLong ulPolarity, xsd_unsignedLong ulGain, xsd_unsignedLong ulICP, xsd_unsignedLong ulOption1, xsd_unsignedLong ulOption2, struct MSXExxxx_Response * Response )`

#### Parameters

[in] *ulChannel* : Channel selection (0 to 7 or 255 for all channels)

[in] *ulAcDc* : AC/DC coupling selection

- 0 : DC
- 1 : AC

[in] *ulSingleDiff* : Single/differential selection

- 0 : Single
- 1 : Differential

[in] *ulPolarity* : Polarity selection

- 0 : Bipolar
- 1 : Unipolar

[in] *ulGain* : Gain selection

- 1 : Gain 1x
- 10 : Gain 10x
- 100 : Gain 100x
- 1000 : Gain 1000x

[in] *ulICP* : ICP source

- 0 : Disable the ICP source
- 1 : Enable the ICP source

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

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

[out] *Response* :

*iReturnValue* :

- 0: Means the remote function performed OK
- -1: Means an system error occurred
- -2: Channel selection wrong
- -3: AC/DC coupling selection wrong
- -4: Single/differential selection wrong
- -5: Polarity selection wrong
- -6: Gain selection wrong
- -7: ICP selection wrong
- -8: ICP not available
- -9: Acquisition in progress. Can not change the configuration

#### Returns

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

**4.1.2.48** `int MSXExxxx_AnalogInputSetSamplingRate ( xsd__unsignedLong ulTimeBase, xsd__unsignedLong ulTimeBaseUnity, xsd__unsignedLong ulSamplingRate, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx_Response * Response )`

Sets the sampling rate (or the sampling period) for the analog inputs.

- If you select **5** for the parameter **ulTimeBaseUnity**, then this function sets the **sampling rate**, i.e. the number of samples per second.
- If you select **0** for the parameter **ulTimeBaseUnity**, then this function sets the **sampling period**, i.e. the time between two samples.

#### Parameters

[in] **ulTimeBase** Time base selection

- 0 for 1MHz
- 1 for 6.4MHz

[in] **ulTimeBaseUnity** Unity

- 0 for  $\mu$ s
- 5 for Hz

[in] **ulSamplingRate** Sampling rate or sampling period according to the selected unity

Constraints:

- **Time base of 1MHz with  $\mu$ s unity (ulTimeBase = 0 and ulTimeBaseUnity = 0)**  
The value of **ulSamplingRate** must be between 8 $\mu$ s and 65535 $\mu$ s.
- **Time base of 1MHz with Hz unity (ulTimeBase = 0 and ulTimeBaseUnity = 5)**  
The value of **ulSamplingRate** must be between 16Hz and 125000Hz.  
The result of the operation 1000000/ulSamplingRate must be an integer number (1000000 % ulSamplingRate = 0).

Typical values:

- 125000Hz
- 100000Hz
- 62500Hz
- 50000Hz
- 40000Hz
- 31250Hz
- 25000Hz
- 20000Hz
- 15625Hz
- 12500Hz
- 10000Hz
- 8000Hz
- 6250Hz
- 5000Hz
- 4000Hz
- 3125Hz
- 2500Hz
- 2000Hz
- 1600Hz

- 1250Hz
- 1000Hz
- 100Hz
- 16Hz

**Time base of 6.4MHz with  $\mu$ s unity (`ulTimeBase = 1` and `ulTimeBaseUnity = 0`)**

The value of `ulSamplingRate` must be between 10 $\mu$ s and 10235 $\mu$ s.

The result of the operation  $6.4 * ulSamplingRate$  must be an integer number.

- **Time base of 6.4MHz with Hz unity (`ulTimeBase = 1` and `ulTimeBaseUnity = 5`)**

The value of `ulSamplingRate` must be between 100Hz and 128000Hz.

The result of the operation  $64000000 / ulSamplingRate$  must be an integer number ( $64000000 \% ulSamplingRate = 0$ ).

Typical values:

- 128000Hz
- 100000Hz
- 80000Hz
- 64000Hz
- 50000Hz
- 40000Hz
- 32000Hz
- 25000Hz
- 20000Hz
- 16000Hz
- 12800Hz
- 12500Hz
- 10000Hz
- 8000Hz
- 6400Hz
- 6250Hz
- 5000Hz
- 4000Hz
- 3200Hz
- 3125Hz
- 2500Hz
- 2000Hz
- 1600Hz
- 1280Hz
- 1000Hz
- 100Hz

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

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

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

- ***iReturnValue***

- **0**: The remote function performed OK
- **-1**: Means an system error occurred
- **-2**: Sampling rate time base selection wrong
- **-3**: Sampling rate time base unity selection wrong
- **-4**: Sampling rate selection wrong

- **-5**: Acquisition in progress. Can not change the sampling rate
- **-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

**4.1.2.49** `int MSXExxxx__AnalogInputGetSamplingRate ( xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx__AnalogInputGetSamplingRateResponse * Response )`

Reads the current sampling rate configuration.

#### Parameters

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

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

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

- ***sResponse.iReturnValue***
  - **0**: The remote function performed OK
  - **-1**: System error occurred
  - **-100**: Internal system error occurred. See value of `syserrno`
- ***sResponse.syserrno*** system error code (the value of the libc "errno" code)
- ***ulClk*** Base clock used to define the sampling rate. 0: 1MHz, 1: 6.4MHz
- ***ulDivisionFactor*** The division factor (used with the `ulClk`) that defines the sampling rate. The sampling rate is always a multiple of the selected clock.

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

**4.1.2.50** `int MSXExxxx__AnalogInputGetConfiguration ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MSXExxxx__AnalogInputGetConfigurationResponse * Response )`

#### Parameters

[in] ***ulChannel*** : Channel selection (0 to 7)

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

[out] ***Response*** :

***sResponse.iReturnValue*** :

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

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

***ulType*** : Analog input type

- 0: Voltage
- 1: Current

***ulICPAvailable*** : ICP available?

- 0: ICP source not available
- 1: ICP source available

***ulAcDc*** : AC/DC coupling selection

- 0 : DC
- 1 : AC

***ulSingleDiff*** : Single/differential selection

- 0 : Single
- 1 : Differential

***ulPolarity*** : Polarity selection

- 0 : Bipolar
- 1 : Unipolar

***ulGain*** : Gain selection

- 1 : Gain 1x
- 10 : Gain 10x
- 100 : Gain 100x
- 1000 : Gain 1000x

***ulICP*** : ICP source

- 0 : ICP source disabled
- 1 : ICP source enabled

***ulTimeBase*** : Sampling rate time base selection

- 0 for 1MHz
- 1 for 6.4MHz

***ulTimeBaseUnity*** : Sampling rate time base unity

- 0 for  $\mu$ s
- 5 for Hz

***ulSamplingRate*** : Sampling rate (or sampling period is ulTimeBaseUnity is 0)

## Returns

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

**4.1.2.51** `int MSXExxxx__AnalogInputCalibrationStart ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulGainSelection, xsd__unsignedLong ulRefSource, xsd__double dRefValue, xsd__unsignedLong ulOption1, xsd__unsignedLong ulOption2, struct MSXExxxx__Response * Response )`

## Parameters

- [in] ***ulChannel*** : Calibration channel selection (0 to 15).
- [in] ***ulGainSelection*** : Selected gain (1, 10, 100,)
- [in] ***ulRefSource*** : Calibration reference source

- 0 : External source via the first function module channel
  - 1 : System internal reference voltage
- [in] **dRefValue** : Reference voltage value for the calibration (V or mA)
- [in] **ulOption1** : Tolerance voltage for the calibration. If not 0, then defines the value of the tolerance (the unit is micro Volt).
- If the calibration does not reach the required tolerance, an error is thrown.
- Leaving 0 for this value will not check the tolerance of the calibration value and will try to get the best potentiometer value
- [in] **ulOption2** : Reserved. Set to 0
- [out] **Response** :
- sResponse.iReturnValue** :
- 0: Means the remote function performed OK
  - -1: Means an system error occurred
  - -2: Channel selection error
  - -3: Any acquisition already started
  - -4: Any calibration already started
  - -5: Gain selection error
  - -6: Reference source selection error
  - -7: Ref value to high/low for the selected gain
  - -8: Calibration start acquisition error

#### Returns

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

**4.1.2.52** `int MSXExxxx__AnalogInputCalibrationGetCurrentStatus ( xsd__unsignedLong ulOption1, struct MSXExxxx__AnalogInputCalibrationGetCurrentStatusResponse * 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
  - -2: No calibration in progress **ulStatus** : Status
  - 0: No calibration in progress
  - 1: Wait to setting the reference value
  - 2: Selected reference value calibration in progress
  - 3: Calibration finished
  - 4: User break occur
  - 5: Can not calibrate the selected reference value **ulChannel** : Current calibration channel
- dRefValue** : Current selected calibration reference value **dCurrentValue** : Last measured calibration value

#### Returns

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



#### 4.1.2.53 `int MSXExxxx__AnalogInputCalibrationNextStep ( xsd__unsignedLong ulOption1, struct MSXExxxx__Response * Response )`

##### Parameters

- [in] *ulOption1* : Reserved. Set to 0
- [out] *Response* :
- iReturnValue* :
- 0: Means the remote function performed OK
  - -1: Means an system error occurred
  - -2: No calibration in progress

##### Returns

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

#### 4.1.2.54 `int MSXExxxx__AnalogInputCalibrationBreak ( xsd__unsignedLong ulOption1, struct MSXExxxx__Response * Response )`

##### Parameters

- [in] *ulOption1* : Reserved. Set to 0
- [out] *Response* :
- iReturnValue* :
- 0: Means the remote function performed OK
  - -1: Means an system error occurred
  - -2: No calibration in progress

##### Returns

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

#### 4.1.2.55 `int MSXExxxx__AnalogOutputGetNumberOfChannels ( 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)
- Response.ulValue* Number of available analog output channels

##### Return values

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

**4.1.2.56** `int MSXExxxx__AnalogOutputWrite1Value ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulOutputType, xsd__unsignedLong ulPolarity, xsd__unsignedLong ulTriggerMask, xsd__unsignedLong ulTriggerEdgeSelection, xsd__unsignedLong ulTriggerCount, xsd__unsignedLong ulValue, xsd__unsignedLong ulOption01, xsd__unsignedLong ulOption02, xsd__unsignedLong ulOption03, struct MSXExxxx__Response * Response )`

#### Parameters

- [in] **ulChannel** Index of the channel (0 to MSXExxxx\_\_AnalogOutputGetNumberOfChannels - 1)
- [in] **ulOutputType** Type of the output
  - **0** voltage
  - **1** current
  - **2** system default
- [in] **ulPolarity** Polarity
  - **0** unipolar
  - **1** bipolar
- [in] **ulTriggerMask** Trigger to use to set the outputs (Mask of bits)
  - **0** no trigger: output directly updated after a call to this function
  - **Bit 0 = 1**: Hardware trigger used
  - **Bit 1 = 1**: Synchro trigger used
  - **Bit 2 = 1**: Software trigger used: output updated after a call to MSXExxxx\_\_AnalogOutputTriggerOutputs You can give a combination of bits. Examples:
  - **1**: Only hardware trigger
  - **2**: Only synchro trigger
  - **3**: Hardware trigger and synchro trigger
  - **7**: Hardware trigger, synchro trigger and software trigger
- [in] **ulTriggerEdgeSelection** Only used if you select the hardware trigger. Defines the detected edges (Mask of bits). The value cannot be null (if hardware trigger is selected).
  - **Bit 0 = 1**: Rising edges are detected
  - **Bit 1 = 1**: Falling edges are detected
  - To detect both edges (rising and falling), the value is also 3.
- [in] **ulTriggerCount** Only used if you select the hardware trigger. Defines the number of hardware trigger to wait before updating the analog outputs. The value cannot be null (if hardware trigger is selected).
- [in] **ulValue** Value that you want to set (digital format)
  - If unipolar, range is from 0 to 0x7fff ( from 0V to +10V)
  - If bipolar, range is from 0 to 0xffff ( from -10V to +10V)
- [in] **ulOption01** Reserved. Set to 0
- [in] **ulOption02** Reserved. Set to 0
- [in] **ulOption03** Reserved. Set to 0
- [out] **Response iReturnValue**
  - **0**: Means the remote function performed OK
  - **-1**: Means an system error occurred
  - **-2**: The PLD is not working
  - **-3**: The ulChannel parameter is wrong

- **-4:** The `ulOutputType` parameter is wrong
- **-5:** The `ulPolarity` parameter is wrong
- **-6:** The `ulTriggerMask` parameter is wrong
- **-7:** The `ulValue` parameter is wrong
- **-8:** The `ulTriggerEdgeSelection` parameter is wrong
- **-9:** The `ulTriggerCount` parameter is wrong
- **-10:** Timeout while setting the value
- **-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

**4.1.2.57** `int MSXExxxx__AnalogOutputGetChannelValue ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse * Response )`

#### Parameters

[in] *ulChannel* index of the channel (0 to `MSXExxxx__AnalogOutputGetNumberOfChannels - 1`)

[in] *ulOption1* Reserved. Set to 0

[out] *Response* *sResponse.iReturnValue*

- **0:** Means the remote function performed OK
- **-1:** Means an system error occurred
- **-2:** The PLD is not working
- **-3:** Invalid parameter channel
- **-4:** Timeout while asking channel value
- **-100:** Internal system error occurred. See value of `syserrno`

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

*ulValue* Current value of the channel (0 corresponds to -10V, 0x7fff corresponds to 0V, 0xffff corresponds to +10V)

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

**4.1.2.58** `int MSXExxxx__AnalogOutputTriggerOutputs ( xsd__unsignedLong ulOption01, struct MSXExxxx__Response * Response )`

Set the voltage of the outputs to the last value written with `MSXExxxx__AnalogOutputWrite1Value`

Only useful if you called `MSXExxxx__AnalogOutputWrite1Value` with the bit 2 of the trigger mask set to 1 (software trigger)

All the output values will be updated at the same time

**Parameters**

[in] *ulOption01* Reserved. Set to 0

[out] *Response iReturnValue*

- **0**: Means the remote function performed OK
- **-1**: Means an system error occurred
- **-2**: The PLD is not working
- **-3**: Timeout while triggering outputs
- **-100**: Internal system error occurred. See value of syserrno

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

**Return values**

**0** SOAP\_OK

*Others* See SOAP error

**4.1.2.59** `int MSXEXxxx__DigitalIOGetNumberOfChannels ( xsd__unsignedLong ulOption1, struct MSXEXxxx__unsignedLongResponse * Response )`

Returns the number of digital I/O channels.

**Parameters**

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

[out] *Response sResponse.iReturnValue*

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-100**: Internal system error occurred. See value of syserrno

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

*ulValue* Number of available digital I/O channels

**Return values**

**0** SOAP\_OK

*Others* See SOAP error

**4.1.2.60** `int MSXEXxxx__DigitalIOGetNumberOfPorts ( xsd__unsignedLong ulOption1, struct MSXEXxxx__unsignedLongResponse * Response )`

Returns the number of digital I/O ports.

A port is a set of consecutive digital I/O channels, whose status can be written or read at the same time.

**Parameters**

[in] *ulOption1* Reserved. Set to 0

[out] *Response sResponse.iReturnValue*

- **0**: The remote function performed OK
- **-1**: System error occurred

- **-100**: Internal system error occurred. See value of `syserrno`

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

*ulValue* Number of available digital I/O ports

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

**4.1.2.61** `int MSXExxxx_DigitalIOGetPortAvailableDirections ( xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MSXExxxx_DigitalIOGetPortAvailableDirectionsResponse * Response )`

Returns the available directions for the selected port (input or output).

#### Parameters

[in] *ulPort* Selected digital I/O port (0 to `MSXExxxx_DigitalIOGetNumberOfPorts`)

Please read the documentation of the `MSXExxxx_DigitalIOGetNumberOfPorts` for the description of a port.

[in] *ulOption1* Reserved. Set to 0

[out] *Response sResponse.iReturnValue*

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-2**: The *ulPort* parameter is wrong
- **-100**: Internal system error occurred. See value of `syserrno`

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

*ulInputs* Digital inputs availability. Each bit indicates if the channel can be used as an input.  
Example:

- **1**: I/O 0 of the selected port can be used as an input
- **4**: I/O 2 of the selected port can be used as an input
- **3**: I/Os 0 and 1 of the selected port can be used as input

*ulOutputs* Digital outputs availability. Each bit indicates if the channel can be used as an output.  
Example:

- **1**: I/O 0 of the selected port can be used as an output
- **4**: I/O 2 of the selected port can be used as an output
- **3**: I/Os 0 and 1 of the selected port can be used as output

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

#### 4.1.2.62 `int MSXExxxx_DigitalIOSetPortDirections ( xsd__unsignedLong ulPort, xsd__unsignedLong ulDirection, xsd__unsignedLong ulOption1, struct MSXExxxx_Response * Response )`

Write the current digital I/O direction for the selected port.

##### Parameters

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulDirection** Digital I/O direction. Each bit indicates if the channel is used as an input or an output. Example:
- **1:** I/O 0 of the selected port is configured as output, all the other I/Os of the selected port are configured as input
  - **3:** I/Os 0 and 1 of the selected port are configured as output, all the other I/Os of the selected port are configured as input
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response iReturnValue**
- **0:** The remote function performed OK
  - **-1:** System error occurred
  - **-2:** The ulPort parameter is wrong
  - **-3:** The ulDirection 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

#### 4.1.2.63 `int MSXExxxx_DigitalIOGetPortDirections ( xsd__unsignedLong ulPort, struct MSXExxxx_unsignedLongResponse * Response )`

Reads the current digital I/O direction for the selected port.

##### Parameters

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response sResponse.iReturnValue**
- **0:** The remote function performed OK
  - **-1:** System error occurred
  - **-2:** The ulPort parameter is wrong
  - **-100:** Internal system error occurred. See value of syserrno

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

*ulValue* Current digital I/O direction. Each bit indicates if the channel is used as an input or an output. Example:

- **1:** I/O 0 of the selected port is configured as output, all the other I/Os of the selected port are configured as input
- **3:** I/Os 0 and 1 of the selected port are configured as output, all the other I/Os of the selected port are configured as input

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

#### 4.1.2.64 int MSXExxxx\_DigitalIOSetInputsFilterTime ( xsd\_unsignedLong ulFilterTime, xsd\_unsignedLong ulOption1, struct MSXExxxx\_Response \* Response )

Sets the filter time for the digital inputs in steps of 250 ns (max value: 16777215)

#### Parameters

[in] *ulFilterTime* Filter time for the digital inputs in steps of 250 ns (max value: 16777215)

- **0:** Disable the filter
- **1:** Sets the filter time to 250 ns
- **2:** Sets the filter time to 500 ns
- ...
- **16777215:** Sets the filter time to 4 s

[in] *ulOption1* Reserved. Set to 0

[out] *Response* Response of the system

- *iReturnValue*
  - **0:** The remote function performed OK
  - **-1:** System error occurred
  - **-2:** 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

#### 4.1.2.65 int MSXExxxx\_DigitalIOEnableDisableInputsFilter ( xsd\_unsignedLong ulPort, xsd\_unsignedLong ulFilter, xsd\_unsignedLong ulOption1, struct MSXExxxx\_Response \* Response )

Enables/disables the digital input filter for the selected port.

**Parameters**

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulFilter** Digital input filter selection. Each bit indicates if the filter is enabled on the input.  
Example:
- **1**: Filter only enabled on input 0
  - **3**: Filter enabled on inputs 0 and 1
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response iReturnValue**
- **0**: The remote function performed OK
  - **-1**: System error occurred
  - **-2**: The ulPort parameter is wrong
  - **-3**: The ulFilter parameter is wrong
  - **-4**: Any selected input is not an input or a bidirectional channel
  - **-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

**4.1.2.66 int MSXExxxx\_\_DigitalIOGetInputsFilterConfiguration ( xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulOption1, struct MSXExxxx\_\_DigitalIOGetInputsFilterConfigurationResponse \* Response )**

Reads the digital inputs filter configuration for the selected port.

**Parameters**

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response sResponse.iReturnValue**
- **0**: The remote function performed OK
  - **-1**: System error occurred
  - **-2**: The ulPort parameter is wrong
  - **-100**: Internal system error occurred. See value of syserrno
- sResponse.syserrno** system error code (the value of the libc "errno" code)
- ulFilterTime** Filter time value (from 1 to 16777215) 1 corresponds to 250 ns, 2 corresponds to 500 ns, ...
- ulFilter** Digital inputs filter selection. Each bit indicate the filter state for one digital input channel.

**Return values**

**0** SOAP\_OK

**Others** See SOAP error



**4.1.2.67** `int MSXExxxx_DigitalIOTestOutputsShortCircuit ( xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse * Response )`

Get the short-circuit status of the outputs of the selected port.

The function returns a mask of bits (32 bits). Each bit represents the short-circuit state of an output.

If you detect a short circuit, first solve it, and then, call the MSXExxxx\_DigitalIORearmOutputsShortCircuit function.

#### Parameters

[in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)

Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.

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

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

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-2**: The ulPort parameter is wrong
- **-100**: Internal system error occurred. See value of syserrno

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

**ulValue :** Digital outputs short circuit state. Each bit represents the short-circuit state of one digital output channel.

- B0 : 0: Digital I/O 0/32 no short circuit. 1: Digital I/O 0/32 short circuit
- ...
- D31 : 0: Digital I/O 31/63 no short circuit. 1: Digital I/O 31/63 short circuit

#### Return values

**0** SOAP\_OK

**Others** See SOAP error

**4.1.2.68** `int MSXExxxx_DigitalIORearmOutputsShortCircuit ( xsd__unsignedLong ulOption1, struct MSXExxxx__Response * Response )`

Rearm the digital outputs short circuit.

Please use only this function if you detected a short circuit using the function MSXExxxx\_DigitalIOTestOutputsShortCircuit.

#### Parameters

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

[out] **Response iReturnValue**

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-100**: Internal system error occurred. See value of syserrno

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

**Returns**

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

**4.1.2.69** `int MSXExxxx__DigitalIOTestOutputsPowerSupply ( xsd__unsignedLong ulPort, xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse * Response )`

Reads the current power supply status of the selected port.

The digital outputs need an external power supply. This function checks the state of the power supply.

**Parameters**

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_\_DigitalIOGetNumberOfPorts)  
Please read the documentation of the MSXExxxx\_\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response sResponse.iReturnValue**
- **0**: The remote function performed OK
  - **-1**: System error occurred
  - **-2**: The PLD is not working
  - **-3**: The ulPort parameter is wrong
  - **-100**: Internal system error occurred. See value of syserrno
- sResponse.syserrno** system error code (the value of the libc "errno" code)
- ulValue** Current digital I/O power supply state. Each bit indicates the power supply state of the output. Example:
- **1**: No external supply voltage for the output 0
  - **3**: No external supply voltage for the outputs 0 and 1

**Return values**

**0** SOAP\_OK

**Others** See SOAP error

**4.1.2.70** `int MSXExxxx__DigitalIOReadChannel ( xsd__unsignedLong ulChannel, xsd__unsignedLong ulOption1, struct MSXExxxx__unsignedLongResponse * Response )`

Read the selected digital I/O channel.

If the selected channel is an output, then this function returns the current output state.

**Parameters**

- [in] **ulChannel** Selected digital I/O channel (0 to MSXExxxx\_\_DigitalIOGetNumberOfChannels)
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response sResponse.iReturnValue :**

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-2**: The ulChannel parameter is wrong
- **-100**: Internal system error occurred. See value of syserrno

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

*ulValue* : Digital I/O channel state

- 0: Digital I/O channel is low
- 1: Digital I/O channel is high

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

#### 4.1.2.71 int MSXExxxx\_DigitalIOReadPort ( xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulOption1, struct MSXExxxx\_\_unsignedLongResponse \* Response )

Read the selected digital I/O port.

#### Parameters

[in] *ulPort* Selected digital I/O port (0 to MSXExxxx\_DigitalIOGetNumberOfPorts)

Please read the documentation of the MSXExxxx\_DigitalIOGetNumberOfPorts for the description of a port.

[in] *ulOption1* Reserved. Set to 0

[out] *Response sResponse.iReturnValue*

- **0**: The remote function performed OK
- **-1**: System error occurred
- **-2**: The ulPort parameter is wrong
- **-100**: Internal system error occurred. See value of syserrno

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

*ulValue* Digital I/O state. Each bit indicates the state of one digital I/O channel.

- D0 : 0: Digital I/O 0/32 is low. 1: Digital I/O 0/32 is high
- ...
- D31 : 0: Digital I/O 31/63 is low. 1: Digital I/O 31/63 is high

#### Return values

**0** SOAP\_OK

*Others* See SOAP error

#### 4.1.2.72 int MSXExxxx\_DigitalIOWriteChannel ( xsd\_\_unsignedLong ulChannel, xsd\_\_unsignedLong ulState, xsd\_\_unsignedLong ulOption1, struct MSXExxxx\_\_Response \* Response )

Set the selected digital output channel to on or off.

**Parameters**

- [in] **ulChannel** Selected digital I/O channel (0 to MSXExxxx\_\_DigitalIOGetNumberOfChannels)
- [in] **ulState** Digital I/O channel state
- 0: Set the digital I/O output channel to low
  - 1: Set the digital I/O output channel to high
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response iReturnValue**
- 0: The remote function performed OK
  - -1: System error occurred
  - -2: The ulChannel parameter is wrong
  - -3: The ulState parameter is wrong
  - -4: The selected digital I/O is not an output
  - -100: Internal system error occurred. See value of syserrno
- syserrno** system error code (the value of the libc "errno" code)

**Return values**

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

#### 4.1.2.73 int MSXExxxx\_\_DigitalIOWritePort ( xsd\_\_unsignedLong ulPort, xsd\_\_unsignedLong ulState, xsd\_\_unsignedLong ulOption1, struct MSXExxxx\_\_Response \* Response )

Write a value to the selected digital I/O port.

**Parameters**

- [in] **ulPort** Selected digital I/O port (0 to MSXExxxx\_\_DigitalIOGetNumberOfPorts)
- Please read the documentation of the MSXExxxx\_\_DigitalIOGetNumberOfPorts for the description of a port.
- [in] **ulState** Digital I/O state. Each bit set the state for one digital I/O channel (0: off, 1: on).
- [in] **ulOption1** Reserved. Set to 0
- [out] **Response iReturnValue**
- 0: The remote function performed OK
  - -1: System error occurred
  - -2: The ulPort parameter is wrong
  - -3: The ulState parameter is wrong
  - -4: Any digital I/O set to 1 is not an output channel
  - -100: Internal system error occurred. See value of syserrno
- syserrno** system error code (the value of the libc "errno" code)

**Return values**

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

# Index

- \_\_offset
    - ByteArray, [71](#)
    - UnsignedLongArray, [88](#)
    - UnsignedShortArray, [89](#)
  - \_\_ptr
    - ByteArray, [71](#)
    - UnsignedLongArray, [88](#)
    - UnsignedShortArray, [89](#)
    - xsd\_\_base64Binary, [89](#)
  - \_\_size
    - ByteArray, [71](#)
    - UnsignedLongArray, [88](#)
    - UnsignedShortArray, [89](#)
    - xsd\_\_base64Binary, [89](#)
- bArray
  - MXCommon\_\_-
    - GetAutoConfigurationFileResponse, [83](#)
- bCryptedValueArray
  - MXCommon\_\_TestCustomerIDResponse, [87](#)
- bValueArray
  - MXCommon\_\_TestCustomerIDResponse, [87](#)
- ByteArray, [71](#)
  - \_\_offset, [71](#)
  - \_\_ptr, [71](#)
  - \_\_size, [71](#)
- Common functions, [11](#)
- Common general functions, [11](#)
- Common hardware trigger functions, [19](#)
- Common I/O auto configuration functions, [25](#)
- Common security functions, [21](#)
- Common synchronisation timer functions, [27](#)
- Common temperature functions, [17](#)
- Common time functions, [23](#)
- Common\_autoconf
  - MXCommon\_\_GetAutoConfigurationFile, [26](#)
  - MXCommon\_\_SetAutoConfigurationFile, [26](#)
  - MXCommon\_\_StartAutoConfiguration, [27](#)
- Common\_configuration
  - MXCommon\_\_-
    - ApplyConfigurationBackupFile, [30](#)
  - MXCommon\_\_ChangePassword, [31](#)
  - MXCommon\_\_GetConfigurationBackupFile, [30](#)
- Common\_general
  - MXCommon\_\_DataseverRestart, [15](#)
  - MXCommon\_\_GetClientConnections, [13](#)
  - MXCommon\_\_GetEthernetLinksStates, [16](#)
  - MXCommon\_\_GetHostname, [12](#)
  - MXCommon\_\_GetModuleType, [12](#)
  - MXCommon\_\_Reboot, [15](#)
  - MXCommon\_\_ResetAllIOFunctionalities, [15](#)
  - MXCommon\_\_SetHostname, [13](#)
  - MXCommon\_\_Sterror, [13](#)
- Common\_hardware\_trigger
  - MXCommon\_\_-
    - GetHardwareTriggerFilterTime, [20](#)
    - MXCommon\_\_GetHardwareTriggerState, [20](#)
    - MXCommon\_\_-
      - SetHardwareTriggerFilterTime, [19](#)
- Common\_security
  - MXCommon\_\_SetCustomerKey, [22](#)
  - MXCommon\_\_TestCustomerID, [22](#)
- Common\_synchrotimer
  - MXCommon\_\_InitAndStartSynchroTimer, [28](#)
  - MXCommon\_\_-
    - StopAndReleaseSynchroTimer, [29](#)
- Common\_temperature
  - MXCommon\_\_-
    - GetModuleTemperatureValueAndStatus, [18](#)
    - MXCommon\_\_-
      - SetModuleTemperatureWarningLevels, [18](#)
- Common\_time
  - MXCommon\_\_GetTime, [24](#)
  - MXCommon\_\_GetUpTime, [25](#)
  - MXCommon\_\_HardwareClockToSys, [24](#)
  - MXCommon\_\_SetTime, [23](#)
  - MXCommon\_\_SysToHardwareClock, [24](#)
- Customer option management, [34](#)
- CustomerOption
  - MXCommon\_\_GetOptionInformation, [35](#)
- dCurrentValue
  - MSXExxxx\_\_-
    - AnalogInputCalibrationGetCurrentStatusResponse,

- 75
- DefaultResponse, 71
  - iReturnValue, 72
  - syserrno, 72
- dRefValue
  - MSXExxxx\_-
    - AnalogInputCalibrationGetCurrentStatusResponse, 75
- dTemperatureValue
  - MXCommon\_-
    - GetModuleTemperatureValueAndStatusResponse, 85
- dValue
  - MSXExxxx\_-
    - AcquisitionAutoRefreshGetValuesResponse, 73
- iReturnValue
  - DefaultResponse, 72
  - MSXExxxx\_\_Response, 80
  - MXCommon\_\_Response, 86
- MSX-E systems servers, 10
- MSX-E312x Acquisition functions, 36
- MSX-E312x Acquisition information functions, 37
- MSX-E312x Analog input calibration functions, 53
- MSX-E312x Analog input functions, 47
- MSX-E312x Analog input initialisation functions, 47
- MSX-E312x Analog output functions, 56
- MSX-E312x Autorefresh functions, 38
- MSX-E312x Digital I/O diagnostic functions, 66
- MSX-E312x Digital I/O filter functions, 63
- MSX-E312x Digital I/O functions, 60
- MSX-E312x Digital I/O information, configuration functions, 60
- MSX-E312x Digital I/O read/write functions, 68
- MSX-E312x functions, 3
- MSX-E312x Sequence functions, 43
- MSXE312x\_Acquisition\_Info
  - MSXExxxx\_\_AcquisitionGetChannelInfo, 37
  - MSXExxxx\_-
    - AcquisitionGetNumberOfChannels, 37
- MSXE312x\_AnaInp\_Calibration
  - MSXExxxx\_\_AnalogInputCalibrationBreak, 55
  - MSXExxxx\_-
    - AnalogInputCalibrationGetCurrentStatus, 55
  - MSXExxxx\_-
    - AnalogInputCalibrationNextStep, 55
  - MSXExxxx\_\_AnalogInputCalibrationStart, 54
- MSXE312x\_AnaInp\_Init
  - MSXExxxx\_\_AnalogInputGetChannelsType, 48
  - MSXExxxx\_\_AnalogInputGetConfiguration, 52
  - MSXExxxx\_-
    - AnalogInputGetNumberOfChannels, 48
  - MSXExxxx\_\_AnalogInputGetSamplingRate, 52
  - MSXExxxx\_-
    - AnalogInputSetChannelConfiguration, 48
  - MSXExxxx\_\_AnalogInputSetSamplingRate, 49
  - MSXE312x\_AnaOut
    - MSXExxxx\_-
      - AnalogOutputGetChannelValue, 58
    - MSXExxxx\_-
      - AnalogOutputGetNumberOfChannels, 57
    - MSXExxxx\_\_AnalogOutputTriggerOutputs, 59
    - MSXExxxx\_\_AnalogOutputWriteI Value, 57
  - MSXE312x\_Autorefresh
    - MSXExxxx\_-
      - AcquisitionAutoRefreshGetValues, 41
    - MSXExxxx\_-
      - AcquisitionAutoRefreshInitAndStart, 40
    - MSXExxxx\_-
      - AcquisitionAutoRefreshStopAndRelease, 42
  - MSXE312x\_DigIO\_Access
    - MSXExxxx\_\_DigitalIOReadChannel, 68
    - MSXExxxx\_\_DigitalIOReadPort, 69
    - MSXExxxx\_\_DigitalIOWriteChannel, 69
    - MSXExxxx\_\_DigitalIOWritePort, 70
  - MSXE312x\_DigIO\_Diagnostic
    - MSXExxxx\_-
      - DigitalIORearmOutputsShortCircuit, 67
    - MSXExxxx\_-
      - DigitalIOTestOutputsPowerSupply, 67
    - MSXExxxx\_-
      - DigitalIOTestOutputsShortCircuit, 66
  - MSXE312x\_DigIO\_Filter
    - MSXExxxx\_-
      - DigitalIOEnableDisableInputsFilter, 64
    - MSXExxxx\_-
      - DigitalIOGetInputsFilterConfiguration,

- 65
- MSXExxxx\_\_DigitalIOSetInputsFilterTime, 64
- MSXE312x\_DigIO\_Info
  - MSXExxxx\_\_-
    - DigitalIOGetNumberOfChannels, 61
  - MSXExxxx\_\_DigitalIOGetNumberOfPorts, 61
  - MSXExxxx\_\_-
    - DigitalIOGetPortAvailableDirections, 61
  - MSXExxxx\_\_DigitalIOGetPortDirections, 63
  - MSXExxxx\_\_DigitalIOSetPortDirections, 62
- MSXE312x\_public\_doc.h, 91
- MSXExxxx\_\_-
  - AcquisitionAutoRefreshGetValues, 120
- MSXExxxx\_\_-
  - AcquisitionAutoRefreshInitAndStart, 118
- MSXExxxx\_\_-
  - AcquisitionAutoRefreshStopAndRelease, 121
- MSXExxxx\_\_AcquisitionGetChannelInfo, 117
- MSXExxxx\_\_-
  - AcquisitionGetNumberOfChannels, 117
- MSXExxxx\_\_-
  - AcquisitionSequenceInitAndStart, 121
- MSXExxxx\_\_-
  - AcquisitionSequenceStopAndRelease, 123
- MSXExxxx\_\_AnalogInputCalibrationBreak, 131
- MSXExxxx\_\_-
  - AnalogInputCalibrationGetCurrentStatus, 130
- MSXExxxx\_\_-
  - AnalogInputCalibrationNextStep, 130
- MSXExxxx\_\_AnalogInputCalibrationStart, 129
- MSXExxxx\_\_AnalogInputGetChannelsType, 124
- MSXExxxx\_\_AnalogInputGetConfiguration, 128
- MSXExxxx\_\_-
  - AnalogInputGetNumberOfChannels, 124
- MSXExxxx\_\_AnalogInputGetSamplingRate, 128
- MSXExxxx\_\_-
  - AnalogInputSetChannelConfiguration, 124
- MSXExxxx\_\_AnalogInputSetSamplingRate, 125
- MSXExxxx\_\_-
  - AnalogOutputGetChannelValue, 133
- MSXExxxx\_\_-
  - AnalogOutputGetNumberOfChannels, 131
- MSXExxxx\_\_AnalogOutputTriggerOutputs, 133
- MSXExxxx\_\_AnalogOutputWriteI Value, 131
- MSXExxxx\_\_-
  - DigitalIOEnableDisableInputsFilter, 137
- MSXExxxx\_\_-
  - DigitalIOGetInputsFilterConfiguration, 138
- MSXExxxx\_\_-
  - DigitalIOGetNumberOfChannels, 134
- MSXExxxx\_\_DigitalIOGetNumberOfPorts, 134
- MSXExxxx\_\_-
  - DigitalIOGetPortAvailableDirections, 135
- MSXExxxx\_\_DigitalIOGetPortDirections, 136
- MSXExxxx\_\_DigitalIOReadChannel, 140
- MSXExxxx\_\_DigitalIOReadPort, 141
- MSXExxxx\_\_-
  - DigitalIORearmOutputsShortCircuit, 139
- MSXExxxx\_\_DigitalIOSetInputsFilterTime, 137
- MSXExxxx\_\_DigitalIOSetPortDirections, 135
- MSXExxxx\_\_-
  - DigitalIOTestOutputsPowerSupply, 140
- MSXExxxx\_\_-
  - DigitalIOTestOutputsShortCircuit, 138
- MSXExxxx\_\_DigitalIOWriteChannel, 141
- MSXExxxx\_\_DigitalIOWritePort, 142
- MXCommon\_\_-
  - ApplyConfigurationBackupFile, 112
- MXCommon\_\_ChangePassword, 113
- MXCommon\_\_DataseverRestart, 102
- MXCommon\_\_GetAutoConfigurationFile, 109
- MXCommon\_\_GetClientConnections, 100
- MXCommon\_\_GetConfigurationBackupFile, 112
- MXCommon\_\_GetEthernetLinksStates, 103
- MXCommon\_\_-
  - GetHardwareTriggerFilterTime, 105

- MXCommon\_\_GetHardwareTriggerState, 106
- MXCommon\_\_GetHostname, 99
- MXCommon\_\_-
  - GetModuleTemperatureValueAndStatus, 104
- MXCommon\_\_GetModuleType, 99
- MXCommon\_\_GetOptionInformation, 115
- MXCommon\_\_GetStateIDFromName, 114
- MXCommon\_\_GetStateNameFromID, 115
- MXCommon\_\_GetSubsystemIDFromName, 114
- MXCommon\_\_GetSubsystemNameFromID, 115
- MXCommon\_\_GetSubSystemState, 113
- MXCommon\_\_GetSynchronizationStatus, 116
- MXCommon\_\_GetTime, 108
- MXCommon\_\_GetUpTime, 109
- MXCommon\_\_HardwareClockToSys, 108
- MXCommon\_\_InitAndStartSynchroTimer, 110
- MXCommon\_\_Reboot, 102
- MXCommon\_\_ResetAllIOFunctionalities, 102
- MXCommon\_\_SetAutoConfigurationFile, 110
- MXCommon\_\_SetCustomerKey, 106
- MXCommon\_\_-
  - SetHardwareTriggerFilterTime, 105
- MXCommon\_\_SetHostname, 100
- MXCommon\_\_-
  - SetModuleTemperatureWarningLevels, 104
- MXCommon\_\_SetTime, 107
- MXCommon\_\_SetToMaster, 116
- MXCommon\_\_StartAutoConfiguration, 110
- MXCommon\_\_-
  - StopAndReleaseSynchroTimer, 111
- MXCommon\_\_Sterror, 100
- MXCommon\_\_SysToHardwareClock, 108
- MXCommon\_\_TestCustomerID, 107
- xsd\_\_char, 99
- xsd\_\_double, 99
- xsd\_\_float, 99
- xsd\_\_int, 99
- xsd\_\_long, 99
- xsd\_\_string, 99
- xsd\_\_unsignedByte, 99
- xsd\_\_unsignedInt, 99
- xsd\_\_unsignedLong, 99
- xsd\_\_unsignedShort, 99
- MSXE312x\_Sequence
  - MSXExxxx\_\_-
    - AcquisitionSequenceInitAndStart, 44
  - MSXExxxx\_\_-
    - AcquisitionSequenceStopAndRelease, 46
  - MSXExxxx\_\_AcquisitionAutoRefreshGetValues
    - MSXE312x\_Autorefresh, 41
    - MSXE312x\_public\_doc.h, 120
  - MSXExxxx\_\_AcquisitionAutoRefreshGetValuesResponse, 72
    - dValue, 73
    - sResponse, 73
    - ulCounterValue, 73
    - ulTimeStampHigh, 73
    - ulTimeStampLow, 73
  - MSXExxxx\_\_AcquisitionAutoRefreshInitAndStart
    - MSXE312x\_Autorefresh, 40
    - MSXE312x\_public\_doc.h, 118
  - MSXExxxx\_\_AcquisitionAutoRefreshStopAndRelease
    - MSXE312x\_Autorefresh, 42
    - MSXE312x\_public\_doc.h, 121
  - MSXExxxx\_\_AcquisitionGetChannelInfo
    - MSXE312x\_Acquisition\_Info, 37
    - MSXE312x\_public\_doc.h, 117
  - MSXExxxx\_\_AcquisitionGetChannelInfoResponse, 73
    - sResponse, 73
    - ulChannelIndex, 74
    - ulHwPosition, 74
    - ulType, 73
  - MSXExxxx\_\_AcquisitionGetNumberOfChannels
    - MSXE312x\_Acquisition\_Info, 37
    - MSXE312x\_public\_doc.h, 117
  - MSXExxxx\_\_AcquisitionSequenceInitAndStart
    - MSXE312x\_public\_doc.h, 121
    - MSXE312x\_Sequence, 44
  - MSXExxxx\_\_AcquisitionSequenceInitAndStartChannelListParam, 74
    - ulChannelList, 74
  - MSXExxxx\_\_AcquisitionSequenceStopAndRelease
    - MSXE312x\_public\_doc.h, 123
    - MSXE312x\_Sequence, 46
  - MSXExxxx\_\_AnalogInputCalibrationBreak
    - MSXE312x\_AnaInp\_Calibration, 55
    - MSXE312x\_public\_doc.h, 131
  - MSXExxxx\_\_AnalogInputCalibrationGetCurrentStatus
    - MSXE312x\_AnaInp\_Calibration, 55
    - MSXE312x\_public\_doc.h, 130
  - MSXExxxx\_\_AnalogInputCalibrationGetCurrentStatusResponse, 74
    - dCurrentValue, 75
    - dRefValue, 75
    - sResponse, 75
    - ulChannel, 75
    - ulStatus, 75
  - MSXExxxx\_\_AnalogInputCalibrationNextStep



- MSXE312x\_AnaInp\_Calibration, [55](#)
- MSXE312x\_public\_doc.h, [130](#)
- MSXExxxx\_\_AnalogInputCalibrationStart
  - MSXE312x\_AnaInp\_Calibration, [54](#)
  - MSXE312x\_public\_doc.h, [129](#)
- MSXExxxx\_\_AnalogInputGetChannelsType
  - MSXE312x\_AnaInp\_Init, [48](#)
  - MSXE312x\_public\_doc.h, [124](#)
- MSXExxxx\_\_AnalogInputGetChannelsTypeResponse,
  - [75](#)
  - sResponse, [75](#)
  - ulICP, [75](#)
  - ulType, [75](#)
- MSXExxxx\_\_AnalogInputGetConfiguration
  - MSXE312x\_AnaInp\_Init, [52](#)
  - MSXE312x\_public\_doc.h, [128](#)
- MSXExxxx\_\_AnalogInputGetConfigurationResponse,
  - [75](#)
  - sResponse, [77](#)
  - ulAcDc, [77](#)
  - ulGain, [77](#)
  - ulICP, [77](#)
  - ulICPAvailable, [77](#)
  - ulPolarity, [77](#)
  - ulSamplingRate, [77](#)
  - ulSingleDiff, [77](#)
  - ulTimeBase, [77](#)
  - ulTimeBaseUnity, [77](#)
  - ulType, [77](#)
- MSXExxxx\_\_AnalogInputGetNumberOfChannels
  - MSXE312x\_AnaInp\_Init, [48](#)
  - MSXE312x\_public\_doc.h, [124](#)
- MSXExxxx\_\_AnalogInputGetSamplingRate
  - MSXE312x\_AnaInp\_Init, [52](#)
  - MSXE312x\_public\_doc.h, [128](#)
- MSXExxxx\_\_AnalogInputGetSamplingRateResponse
  - [77](#)
  - sResponse, [77](#)
  - ulClk, [77](#)
  - ulDivisionFactor, [77](#)
- MSXExxxx\_\_AnalogInputSetChannelConfiguration
  - MSXE312x\_AnaInp\_Init, [48](#)
  - MSXE312x\_public\_doc.h, [124](#)
- MSXExxxx\_\_AnalogInputSetSamplingRate
  - MSXE312x\_AnaInp\_Init, [49](#)
  - MSXE312x\_public\_doc.h, [125](#)
- MSXExxxx\_\_AnalogOutputGetChannelValue
  - MSXE312x\_AnaOut, [58](#)
  - MSXE312x\_public\_doc.h, [133](#)
- MSXExxxx\_\_AnalogOutputGetNumberOfChannels
  - MSXE312x\_AnaOut, [57](#)
  - MSXE312x\_public\_doc.h, [131](#)
- MSXExxxx\_\_AnalogOutputTriggerOutputs
  - MSXE312x\_AnaOut, [59](#)
  - MSXE312x\_public\_doc.h, [133](#)
- MSXExxxx\_\_AnalogOutputWriteIValue
  - MSXE312x\_AnaOut, [57](#)
  - MSXE312x\_public\_doc.h, [131](#)
- MSXExxxx\_\_DigitalIOEnableDisableInputsFilter
  - MSXE312x\_DigIO\_Filter, [64](#)
  - MSXE312x\_public\_doc.h, [137](#)
- MSXExxxx\_\_DigitalIOGetInputsFilterConfiguration
  - MSXE312x\_DigIO\_Filter, [65](#)
  - MSXE312x\_public\_doc.h, [138](#)
- MSXExxxx\_\_DigitalIOGetInputsFilterConfigurationResponse,
  - [78](#)
  - sResponse, [78](#)
  - ulFilter, [78](#)
  - ulFilterTime, [78](#)
- MSXExxxx\_\_DigitalIOGetNumberOfChannels
  - MSXE312x\_DigIO\_Info, [61](#)
  - MSXE312x\_public\_doc.h, [134](#)
- MSXExxxx\_\_DigitalIOGetNumberOfPorts
  - MSXE312x\_DigIO\_Info, [61](#)
  - MSXE312x\_public\_doc.h, [134](#)
- MSXExxxx\_\_DigitalIOGetPortAvailableDirections
  - MSXE312x\_DigIO\_Info, [61](#)
  - MSXE312x\_public\_doc.h, [135](#)
- MSXExxxx\_\_DigitalIOGetPortAvailableDirectionsResponse,
  - [78](#)
  - sResponse, [79](#)
  - ulInputs, [79](#)
  - ulOutputs, [79](#)
- MSXExxxx\_\_DigitalIOGetPortDirections
  - MSXE312x\_DigIO\_Info, [63](#)
  - MSXE312x\_public\_doc.h, [136](#)
- MSXExxxx\_\_DigitalIOReadChannel
  - MSXE312x\_DigIO\_Access, [68](#)
  - MSXE312x\_public\_doc.h, [140](#)
- MSXExxxx\_\_DigitalIOReadPort
  - MSXE312x\_DigIO\_Access, [69](#)
  - MSXE312x\_public\_doc.h, [141](#)
- MSXExxxx\_\_DigitalIORearmOutputsShortCircuit
  - MSXE312x\_DigIO\_Diagnostic, [67](#)
  - MSXE312x\_public\_doc.h, [139](#)
- MSXExxxx\_\_DigitalIOSetInputsFilterTime
  - MSXE312x\_DigIO\_Filter, [64](#)
  - MSXE312x\_public\_doc.h, [137](#)
- MSXExxxx\_\_DigitalIOSetPortDirections
  - MSXE312x\_DigIO\_Info, [62](#)
  - MSXE312x\_public\_doc.h, [135](#)
- MSXExxxx\_\_DigitalIOTestOutputsPowerSupply
  - MSXE312x\_DigIO\_Diagnostic, [67](#)
  - MSXE312x\_public\_doc.h, [140](#)
- MSXExxxx\_\_DigitalIOTestOutputsShortCircuit
  - MSXE312x\_DigIO\_Diagnostic, [66](#)
  - MSXE312x\_public\_doc.h, [138](#)
- MSXExxxx\_\_DigitalIOWriteChannel

- MSXE312x\_DigIO\_Access, 69
- MSXE312x\_public\_doc.h, 141
- MSXExxxx\_DigitalIOWritePort
  - MSXE312x\_DigIO\_Access, 70
  - MSXE312x\_public\_doc.h, 142
- MSXExxxx\_FileResponse, 79
  - sArray, 80
  - sResponse, 80
  - ulEOF, 80
- MSXExxxx\_Response, 80
  - iReturnValue, 80
  - syserrno, 80
- MSXExxxx\_unsignedLongResponse, 80
  - sResponse, 81
  - ulValue, 81
- MSXExxxx\_unsignedLongTimeStampResponse, 81
  - sResponse, 81
  - ulTimeStampHigh, 81
  - ulTimeStampLow, 81
  - ulValue, 81
- MXCommon\_\_ApplyConfigurationBackupFile
  - Common\_configuration, 30
  - MSXE312x\_public\_doc.h, 112
- MXCommon\_\_ByteArrayResponse, 81
  - sArray, 82
  - sResponse, 82
- MXCommon\_\_ChangePassword
  - Common\_configuration, 31
  - MSXE312x\_public\_doc.h, 113
- MXCommon\_\_DataseverRestart
  - Common\_general, 15
  - MSXE312x\_public\_doc.h, 102
- MXCommon\_\_FileResponse, 82
  - sArray, 82
  - sResponse, 82
  - ulEOF, 82
- MXCommon\_\_GetAutoConfigurationFile
  - Common\_autoconf, 26
  - MSXE312x\_public\_doc.h, 109
- MXCommon\_\_GetAutoConfigurationFileResponse, 82
  - bArray, 83
  - sResponse, 83
  - ulEOF, 83
- MXCommon\_\_GetClientConnections
  - Common\_general, 13
  - MSXE312x\_public\_doc.h, 100
- MXCommon\_\_GetConfigurationBackupFile
  - Common\_configuration, 30
  - MSXE312x\_public\_doc.h, 112
- MXCommon\_\_GetEthernetLinksStates
  - Common\_general, 16
  - MSXE312x\_public\_doc.h, 103
- MXCommon\_\_GetEthernetLinksStatesResponse, 83
  - sPort0, 83
  - sPort1, 83
  - sResponse, 83
- MXCommon\_\_GetHardwareTriggerFilterTime
  - Common\_hardware\_trigger, 20
  - MSXE312x\_public\_doc.h, 105
- MXCommon\_\_GetHardwareTriggerFilterTimeResponse, 83
  - sResponse, 84
  - ulFilterTime, 84
  - ulInfo01, 84
  - ulInfo02, 84
- MXCommon\_\_GetHardwareTriggerState
  - Common\_hardware\_trigger, 20
  - MSXE312x\_public\_doc.h, 106
- MXCommon\_\_GetHardwareTriggerStateResponse, 84
  - sResponse, 84
  - ulInfo01, 84
  - ulInfo02, 84
  - ulState, 84
- MXCommon\_\_GetHostname
  - Common\_general, 12
  - MSXE312x\_public\_doc.h, 99
- MXCommon\_\_GetModuleTemperatureValueAndStatus
  - Common\_temperature, 18
  - MSXE312x\_public\_doc.h, 104
- MXCommon\_\_GetModuleTemperatureValueAndStatusResponse, 84
  - dTemperatureValue, 85
  - sResponse, 85
  - ulInfo, 85
  - ulTemperatureStatus, 85
- MXCommon\_\_GetModuleType
  - Common\_general, 12
  - MSXE312x\_public\_doc.h, 99
- MXCommon\_\_GetOptionInformation
  - CustomerOption, 35
  - MSXE312x\_public\_doc.h, 115
- MXCommon\_\_GetStateIDFromName
  - MSXE312x\_public\_doc.h, 114
  - SystemStatemanagement, 33
- MXCommon\_\_GetStateNameFromID
  - MSXE312x\_public\_doc.h, 115
  - SystemStatemanagement, 34
- MXCommon\_\_GetSubsystemIDFromName
  - MSXE312x\_public\_doc.h, 114
  - SystemStatemanagement, 33
- MXCommon\_\_GetSubsystemNameFromID
  - MSXE312x\_public\_doc.h, 115
  - SystemStatemanagement, 33
- MXCommon\_\_GetSubSystemState

- MSXE312x\_public\_doc.h, [113](#)
- SystemStatemanagement, [32](#)
- MXCommon\_\_GetSynchronizationStatus
  - MSXE312x\_public\_doc.h, [116](#)
  - Synchronisation, [36](#)
- MXCommon\_\_GetTime
  - Common\_time, [24](#)
  - MSXE312x\_public\_doc.h, [108](#)
- MXCommon\_\_GetTimeResponse, [85](#)
  - sResponse, [85](#)
  - ulHighTime, [85](#)
  - ulLowTime, [85](#)
- MXCommon\_\_GetUpTime
  - Common\_time, [25](#)
  - MSXE312x\_public\_doc.h, [109](#)
- MXCommon\_\_GetUpTimeResponse, [85](#)
  - sResponse, [86](#)
  - ulUpTime, [86](#)
- MXCommon\_\_HardwareClockToSys
  - Common\_time, [24](#)
  - MSXE312x\_public\_doc.h, [108](#)
- MXCommon\_\_InitAndStartSynchroTimer
  - Common\_synchrotimer, [28](#)
  - MSXE312x\_public\_doc.h, [110](#)
- MXCommon\_\_Reboot
  - Common\_general, [15](#)
  - MSXE312x\_public\_doc.h, [102](#)
- MXCommon\_\_ResetAllIOFunctionalities
  - Common\_general, [15](#)
  - MSXE312x\_public\_doc.h, [102](#)
- MXCommon\_\_Response, [86](#)
  - iReturnValue, [86](#)
  - syserrno, [86](#)
- MXCommon\_\_SetAutoConfigurationFile
  - Common\_autoconf, [26](#)
  - MSXE312x\_public\_doc.h, [110](#)
- MXCommon\_\_SetCustomerKey
  - Common\_security, [22](#)
  - MSXE312x\_public\_doc.h, [106](#)
- MXCommon\_\_SetHardwareTriggerFilterTime
  - Common\_hardware\_trigger, [19](#)
  - MSXE312x\_public\_doc.h, [105](#)
- MXCommon\_\_SetHostname
  - Common\_general, [13](#)
  - MSXE312x\_public\_doc.h, [100](#)
- MXCommon\_\_SetModuleTemperatureWarningLevels
  - Common\_temperature, [18](#)
  - MSXE312x\_public\_doc.h, [104](#)
- MXCommon\_\_SetTime
  - Common\_time, [23](#)
  - MSXE312x\_public\_doc.h, [107](#)
- MXCommon\_\_SetToMaster
  - MSXE312x\_public\_doc.h, [116](#)
  - Synchronisation, [35](#)
- MXCommon\_\_StartAutoConfiguration
  - Common\_autoconf, [27](#)
  - MSXE312x\_public\_doc.h, [110](#)
- MXCommon\_\_StopAndReleaseSynchroTimer
  - Common\_synchrotimer, [29](#)
  - MSXE312x\_public\_doc.h, [111](#)
- MXCommon\_\_Sterror
  - Common\_general, [13](#)
  - MSXE312x\_public\_doc.h, [100](#)
- MXCommon\_\_SysToHardwareClock
  - Common\_time, [24](#)
  - MSXE312x\_public\_doc.h, [108](#)
- MXCommon\_\_TestCustomerID
  - Common\_security, [22](#)
  - MSXE312x\_public\_doc.h, [107](#)
- MXCommon\_\_TestCustomerIDResponse, [86](#)
  - bCryptedValueArray, [87](#)
  - bValueArray, [87](#)
  - sResponse, [87](#)
- MXCommon\_\_unsignedLongResponse, [87](#)
  - sResponse, [87](#)
  - ulValue, [87](#)
- sArray
  - MSXExxxx\_\_FileResponse, [80](#)
  - MXCommon\_\_ByteArrayResponse, [82](#)
  - MXCommon\_\_FileResponse, [82](#)
- Set/Backup/Restore general system configuration, [29](#)
- sGetEthernetLinksStatesPort, [87](#)
  - ulDuplex, [88](#)
  - ulInfo1, [88](#)
  - ulInfo2, [88](#)
  - ulSpeed, [88](#)
  - ulState, [88](#)
- SOAP function calls in C/C++ language, [3](#)
- Software hints, [3](#)
- sPort0
  - MXCommon\_\_-GetEthernetLinksStatesResponse, [83](#)
- sPort1
  - MXCommon\_\_-GetEthernetLinksStatesResponse, [83](#)
- sResponse
  - MSXExxxx\_\_-AcquisitionAutoRefreshGetValuesResponse, [73](#)
  - MSXExxxx\_\_-AcquisitionGetChannelInfoResponse, [73](#)
  - MSXExxxx\_\_-AnalogInputCalibrationGetCurrentStatusResponse, [75](#)

- MSXExxxx\_\_-
  - AnalogInputGetChannelsTypeResponse, 75
- MSXExxxx\_\_-
  - AnalogInputGetConfigurationResponse, 77
- MSXExxxx\_\_-
  - AnalogInputGetSamplingRateResponse, 77
- MSXExxxx\_\_-
  - DigitalIOGetInputsFilterConfigurationResponse, 78
- MSXExxxx\_\_-
  - DigitalIOGetPortAvailableDirectionsResponse, 79
- MSXExxxx\_\_FileResponse, 80
- MSXExxxx\_\_unsignedLongResponse, 81
- MSXExxxx\_\_-
  - unsignedLongTimeStampResponse, 81
- MXCommon\_\_ByteArrayResponse, 82
- MXCommon\_\_FileResponse, 82
- MXCommon\_\_-
  - GetAutoConfigurationFileResponse, 83
- MXCommon\_\_-
  - GetEthernetLinksStatesResponse, 83
- MXCommon\_\_-
  - GetHardwareTriggerFilterTimeResponse, 84
- MXCommon\_\_-
  - GetHardwareTriggerStateResponse, 84
- MXCommon\_\_-
  - GetModuleTemperatureValueAndStatusResponse, 85
- MXCommon\_\_GetTimeResponse, 85
- MXCommon\_\_GetUpTimeResponse, 86
- MXCommon\_\_TestCustomerIDResponse, 87
- MXCommon\_\_unsignedLongResponse, 87
- Synchronisation
  - MXCommon\_\_GetSynchronizationStatus, 36
  - MXCommon\_\_SetToMaster, 35
- Synchronisation management, 35
- syserrno
  - DefaultResponse, 72
  - MSXExxxx\_\_Response, 80
  - MXCommon\_\_Response, 86
- System state management, 32
- SystemStatemanagement
  - MXCommon\_\_GetStateIDFromName, 33
  - MXCommon\_\_GetStateNameFromID, 34
  - MXCommon\_\_GetSubsystemIDFromName, 33
- MXCommon\_\_GetSubsystemNameFromID, 33
- MXCommon\_\_GetSubSystemState, 32
- ulAcDc
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse, 77
- ulChannel
  - MSXExxxx\_\_-
    - AnalogInputCalibrationGetCurrentStatusResponse, 75
- ulChannelIndex
  - MSXExxxx\_\_-
    - AcquisitionGetChannelInfoResponse, 74
- ulChannelList
  - MSXExxxx\_\_-
    - AcquisitionSequenceInitAndStartChannelListParam, 74
- ulClk
  - MSXExxxx\_\_-
    - AnalogInputGetSamplingRateResponse, 77
- ulCounterValue
  - MSXExxxx\_\_-
    - AcquisitionAutoRefreshGetValuesResponse, 73
- ulDivisionFactor
  - MSXExxxx\_\_-
    - AnalogInputGetSamplingRateResponse, 77
- ulDuplex
  - sGetEthernetLinksStatesPort, 88
- ulFilter
  - MSXExxxx\_\_-
    - DigitalIOGetInputsFilterConfigurationResponse, 78
- ulFilterTime
  - MSXExxxx\_\_-
    - DigitalIOGetInputsFilterConfigurationResponse, 78
- MXCommon\_\_-
  - GetHardwareTriggerFilterTimeResponse, 84
- ulGain
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse,

- 77
- ulHighTime
  - MXCommon\_\_GetTimeResponse, 85
- ulHwPosition
  - MSXExxxx\_\_-
    - AcquisitionGetChannelInfoResponse, 74
- ulICP
  - MSXExxxx\_\_-
    - AnalogInputGetChannelsTypeResponse, 75
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse, 77
- ulICPAvailable
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse, 77
- ulInfo
  - MXCommon\_\_-
    - GetModuleTemperatureValueAndStatusResponse, 85
- ulInfo01
  - MXCommon\_\_-
    - GetHardwareTriggerFilterTimeResponse, 84
  - MXCommon\_\_-
    - GetHardwareTriggerStateResponse, 84
- ulInfo02
  - MXCommon\_\_-
    - GetHardwareTriggerFilterTimeResponse, 84
  - MXCommon\_\_-
    - GetHardwareTriggerStateResponse, 84
- ulInfo1
  - sGetEthernetLinksStatesPort, 88
- ulInfo2
  - sGetEthernetLinksStatesPort, 88
- ulInputs
  - MSXExxxx\_\_-
    - DigitalIOGetPortAvailableDirectionsResponse, 79
- ulLowTime
  - MXCommon\_\_GetTimeResponse, 85
- ulOutputs
  - MSXExxxx\_\_-
    - DigitalIOGetPortAvailableDirectionsResponse, 79
- ulPolarity
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse, 77
- ulSamplingRate
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse, 77
- ulSingleDiff
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse, 77
- ulSpeed
  - sGetEthernetLinksStatesPort, 88
- ulState
  - MXCommon\_\_-
    - GetHardwareTriggerStateResponse, 84
  - sGetEthernetLinksStatesPort, 88
- ulStatus
  - MSXExxxx\_\_-
    - AnalogInputCalibrationGetCurrentStatusResponse, 75
- ulTemperatureStatus
  - MXCommon\_\_-
    - GetModuleTemperatureValueAndStatusResponse, 85
- ulTimeBase
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse, 77
- ulTimeBaseUnity
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse, 77
- ulTimeStampHigh
  - MSXExxxx\_\_-
    - AcquisitionAutoRefreshGetValuesResponse, 73
  - MSXExxxx\_\_-
    - unsignedLongTimeStampResponse, 81
- ulTimeStampLow
  - MSXExxxx\_\_-
    - AcquisitionAutoRefreshGetValuesResponse, 73
  - MSXExxxx\_\_-
    - unsignedLongTimeStampResponse, 81
- ulType
  - MSXExxxx\_\_-
    - AcquisitionGetChannelInfoResponse, 73
  - MSXExxxx\_\_-
    - AnalogInputGetChannelsTypeResponse, 75
  - MSXExxxx\_\_-
    - AnalogInputGetConfigurationResponse,

77

ulUpTime  
    MXCommon\_\_GetUpTimeResponse, 86

ulValue  
    MSXExxxx\_\_unsignedLongResponse, 81  
    MSXExxxx\_\_-  
        unsignedLongTimeStampResponse,  
        81  
    MXCommon\_\_unsignedLongResponse, 87

UnsignedLongArray, 88  
    \_\_offset, 88  
    \_\_ptr, 88  
    \_\_size, 88

UnsignedShortArray, 88  
    \_\_offset, 89  
    \_\_ptr, 89  
    \_\_size, 89

xsd\_\_base64Binary, 89  
    \_\_ptr, 89  
    \_\_size, 89

xsd\_\_char  
    MSXE312x\_public\_doc.h, 99

xsd\_\_double  
    MSXE312x\_public\_doc.h, 99

xsd\_\_float  
    MSXE312x\_public\_doc.h, 99

xsd\_\_int  
    MSXE312x\_public\_doc.h, 99

xsd\_\_long  
    MSXE312x\_public\_doc.h, 99

xsd\_\_string  
    MSXE312x\_public\_doc.h, 99

xsd\_\_unsignedByte  
    MSXE312x\_public\_doc.h, 99

xsd\_\_unsignedInt  
    MSXE312x\_public\_doc.h, 99

xsd\_\_unsignedLong  
    MSXE312x\_public\_doc.h, 99

xsd\_\_unsignedShort  
    MSXE312x\_public\_doc.h, 99