MODBUS interface description

Table of Contents

General description **1**
Introduction 1
Why a MODBUS Server on the MSX-E modules? 1
Technical details 1

FC3 (read multiple register) Functions **3**
Function GetLastCommandStatus 3
For new application(s) or automate communication it is recommended to use the function GetLastCommandStatusEx. 3
Description 3
Query frame layout 4
Response frame layout 4
Exception frame layout 5
Function GetLastCommandStatusEx 5
Description 5
Query frame layout 6
Response frame layout 6
Exception frame layout 7
Function MXCommon__GetModuleType 7
For new application(s) or automate communication it is recommended to use the function MXCommon__GetModuleTypeEx. 7
Description 7
Query frame layout 7
Response frame layout 8
Exception frame layout 9
Function MXCommon__GetModuleTypeEx 9
Description 9
Query frame layout 9
Response frame layout 10
Exception frame layout 10
Function MXCommon__GetTime 11
For new application(s) or automate communication it is recommended to use the function MXCommon__GetTimeEx. 11
Description 11
Query frame layout 11
Response frame layout 12
Exception frame layout 12
Function MXCommon__GetTimeEx 13
Description 13
Query frame layout 13
Response frame layout 13
Exception frame layout 14
Function MXCommon__TestCustomerID 14
For new application(s) or automate communication it is recommended to use the function MXCommon__TestCustomerIDEx. 15
Description 15
Query frame layout 15
Response frame layout 15

Table of Contents

FC3 (read multiple register) Functions
Exception frame layout....16
Function MXCommon__TestCustomerIDEx....16
Description....16
Query frame layout....17
Response frame layout....17
Exception frame layout....18
Function MSXE360X__AnalogInputGetSequenceStatus....18
For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputGetSequenceStatusEx.....18
Description....18
Query frame layout....19
Response frame layout....19
Exception frame layout....20
Function MSXE360X__AnalogInputGetSequenceStatusEx....20
Description....20
Query frame layout....21
Response frame layout....22
Exception frame layout....22
Function MSXE360X__AnalogInputGetSequenceConfiguration....23
For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputGetSequenceConfigurationEx.....23
Description....23
Query frame layout....25
Response frame layout....26
Exception frame layout....27
Function MSXE360X__AnalogInputGetSequenceConfigurationEx....27
Description....27
Query frame layout....30
Response frame layout....30
Exception frame layout....32

FC16 (write multiple register) Functions....33
Function MXCommon__SetHardwareTriggerFilterTime....33
For new application(s) or automate communication it is recommended to use the function MXCommon__SetHardwareTriggerFilterTimeEx.....33
Description....33
Query frame layout....34
Response frame layout....35
Exception frame layout....35
Function MXCommon__SetHardwareTriggerFilterTimeEx....36
Description....36
Query frame layout....36
Response frame layout....37
Exception frame layout....37
Function MXCommon__InitAndStartSynchroTimer....38
For new application(s) or automate communication it is recommended to use the function MXCommon__InitAndStartSynchroTimerEx.....38

Table of Contents

FC16 (write multiple register) Functions
Description....38
Query frame layout....39
Response frame layout....40
Exception frame layout....40
Function MXCommon__InitAndStartSynchroTimerEx....41
Description....41
Query frame layout....42
Response frame layout....43
Exception frame layout....44
Function MXCommon__StopAndReleaseSynchroTimer....44
For new application(s) or automate communication it is recommended to use the function MXCommon__StopAndReleaseSynchroTimerEx.....44
Description....44
Query frame layout....45
Response frame layout....45
Exception frame layout....46
Function MXCommon__StopAndReleaseSynchroTimerEx....46
Description....46
Query frame layout....47
Response frame layout....47
Exception frame layout....48
Function MXCommon__Reboot....48
For new application(s) or automate communication it is recommended to use the function MXCommon__RebootEx.....48
Description....48
Query frame layout....49
Response frame layout....49
Exception frame layout....50
Function MXCommon__RebootEx....50
Description....50
Query frame layout....51
Response frame layout....51
Exception frame layout....52
Function MXCommon__SetCustomerKey....52
For new application(s) or automate communication it is recommended to use the function MXCommon__SetCustomerKeyEx.....52
Description....52
Query frame layout....53
Response frame layout....53
Exception frame layout....54
Function MXCommon__SetCustomerKeyEx....54
Description....54
Query frame layout....55
Response frame layout....55
Exception frame layout....56
Function MXCommon__SetFilterChannels....56
For new application(s) or automate communication it is recommended to use the function

Table of Contents

FC16 (write multiple register) Functions
MXCommon_SetFilterChannelsEx.....56
Description.....57
Query frame layout.....57
Response frame layout.....58
Exception frame layout.....58
Function MXCommon_SetFilterChannelsEx.....59
Description.....59
Query frame layout.....59
Response frame layout.....60
Exception frame layout.....60
Function MSXE360X_AnalogInputInitSequence.....61
For new application(s) or automate communication it is recommended to use the function MSXE360X_AnalogInputInitSequenceEx.....61
Description.....61
Query frame layout.....64
Response frame layout.....65
Exception frame layout.....66
Function MSXE360X_AnalogInputInitSequenceEx.....66
Description.....66
Query frame layout.....70
Response frame layout.....71
Exception frame layout.....72
Function MSXE360X_AnalogInputStartSequence.....72
For new application(s) or automate communication it is recommended to use the function MSXE360X_AnalogInputStartSequenceEx.....72
Description.....72
Query frame layout.....73
Response frame layout.....74
Exception frame layout.....74
Function MSXE360X_AnalogInputStartSequenceEx.....75
Description.....75
Query frame layout.....76
Response frame layout.....77
Exception frame layout.....77
Function MSXE360X_AnalogInputInitAndStartSequence.....78
For new application(s) or automate communication it is recommended to use the function MSXE360X_AnalogInputInitAndStartSequenceEx.....78
Description.....78
Query frame layout.....81
Response frame layout.....82
Exception frame layout.....83
Function MSXE360X_AnalogInputInitAndStartSequenceEx.....83
Description.....84
Query frame layout.....87
Response frame layout.....88
Exception frame layout.....89
Function MSXE360X_AnalogInputStopSequence.....89

Table of Contents

FC16 (write multiple register) Functions
For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputStopSequenceEx. 89
Description 89
Query frame layout 90
Response frame layout 91
Exception frame layout 91
Function MSXE360X__AnalogInputStopSequenceEx 92
Description 92
Query frame layout 92
Response frame layout 93
Exception frame layout 93
Function MSXE360X__AnalogInputReleaseSequence 94
For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputReleaseSequenceEx. 94
Description 94
Query frame layout 94
Response frame layout 95
Exception frame layout 96
Function MSXE360X__AnalogInputReleaseSequenceEx 96
Description 96
Query frame layout 97
Response frame layout 97
Exception frame layout 98
Function MSXE360X__AnalogInputStopAndReleaseSequence 98
For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputStopAndReleaseSequenceEx. 98
Description 98
Query frame layout 99
Response frame layout 100
Exception frame layout 100
Function MSXE360X__AnalogInputStopAndReleaseSequenceEx 101
Description 101
Query frame layout 101
Response frame layout 102
Exception frame layout 102

FC23 (read/write registers) Functions 104
Query frame layout 104
Response frame layout 105
Exception frame layout 105

Exception code description 106

Siemens Step 7 compatibility information (AWL/SDF code) 107

General description

Top

Introduction

This document describes the protocol used by the MODBUS server of the module.
The OPEN MODBUS protocol is based on the widely known MODBUS protocol.
OPEN MODBUS is an open protocol and is not manufacturer dependent.
It is mainly used to connect PLC and I/O devices.

Why a MODBUS Server on the MSX-E modules?

Thanks to the MODBUS server, it is possible to manage an MSX-E module with e.g.: a Siemens S7 PLC.
The S7 PLC can start acquisitions and read data from the MSX-E module!

Technical details

Please note that only MODBUS over TCP is standardized. Nonetheless in this present
version the server implements OPEN MODBUS/TCP class 0 and one function of the class 2 even on UDP sockets.

The MODBUS/TCP class 0 defines two types of query: FC3 and FC16.

- **FC3 functions** read register content from the memory of the remote system
- **FC16 functions** write new register content on the memory of the remote system

The MODBUS/TCP server implement the following query of the class 2 : FC23.

- **FC23 functions** read/write registers content from/to the memory of the remote system

The MODBUS server offer a virtual memory organisation: registers (functions)
are mapped to be equivalent to SOAP functions.

Characteristics of this communication channel as the standardisation document describes it are:

- The default port used by the server is **512** in both UDP/IP and TCP/IP. You can change this via the web server.
- Data are sent in network order, i.e. **big endian (Motorola formata)**. Use the standard C functions atons/atonl and ntohs/ntohl to convert values bigger than 1 bytes.
- Datastructures used to describe parameters that are embedded in on-wire frames **must** be packed. How to do that is compiler-dependant.

The ADDI-DATA MSX-E Modbus server offers the following extension to the standard:

- It is possible to configure the server to accept data sent in **little endian (Intel format)** (native order)
- In this case, the default port used is **215**. You can change this via the web server.

As answer to query a client may receive an acknowledgement (named *standard response* onward) or an exception.
If an exception or an error occured, you can use the GetLastCommandStatus command to get the real error number (from the remote server).
Real error numbers are described for each command in the "Returns" field.

The chapter below describes the available functions and their parameters.
It also contains the precise description of all frames implied in a given action.

FC3 (read multiple register) Functions

Top

Functions in this group are used to read values on the module.

- GetLastCommandStatus — Register: **0**
- GetLastCommandStatusEx — Register: **10000**
- MXCommon__GetModuleType — Register: **1**
- MXCommon__GetModuleTypeEx — Register: **10200**
- MXCommon__GetTime — Register: **2**
- MXCommon__GetTimeEx — Register: **10500**
- MXCommon__TestCustomerID — Register: **3**
- MXCommon__TestCustomerIDEx — Register: **10550**
- MSXE360X__AnalogInputGetSequenceStatus — Register: **100**
- MSXE360X__AnalogInputGetSequenceStatusEx — Register: **1000**
- MSXE360X__AnalogInputGetSequenceConfiguration — Register: **101**
- MSXE360X__AnalogInputGetSequenceConfigurationEx — Register: **1050**

Function GetLastCommandStatus

For new application(s) or automate communication it is recommended to use the function GetLastCommandStatusEx.

Description

Return the result of the last remote function call

Parameters:

[Response frame layout] ***ReturnValue:*** The return value of the remote function.

- ♦ 0 Always means success
- ♦ -100 means you should check Syserrno;

♦ for other values, check the documentation of the function

[Response frame layout] ***Syserrno:*** the value of the libc errno after the call to the remote function

[Response frame layout] ***Errstr:*** A nul-terminated string describing the error code Syserrno

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	0	0x0000	0x0000
word count	2	16-bit integer	54	0x3600	0x0036

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	112	0x7000	0x0070
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function	1	8-bit integer	0x03	0x03	0x03

code					
Byte count	2	16-bit integer	108	0x6C00	0x006C
ReturnValue	4	32-bit integer	See the description above	0x????????	0x????????
Syserrno	4	32-bit integer	See the description above	0x????????	0x????????
Errstr	100	8-bit integer array	See the description above	0x??[100]	0x??[100]

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function GetLastCommandStatusEx

Description

Return the result of the last remote function call

Parameters:

[Response frame layout] ***ReturnValue:*** The return value of the remote function.

- 0 Always means success
- -100 means you should check Syserrno;
- for other values, check the documentation of the function

[Response frame layout] ***Syserrno:*** the value of the libc errno after the call to the remote function

[Response frame layout] ***Errstr:*** A nul-terminated string describing the error code Syserrno

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	10000	0x1027	0x2710
word count	2	16-bit integer	54	0x3600	0x0036

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	111	0x6F00	0x006F
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	1	8-bit integer	108	0x6C	0x6C

ReturnValue	4	32-bit integer	See the description above	0x????????	0x????????
Syserrno	4	32-bit integer	See the description above	0x????????	0x????????
Errstr	100	8-bit integer array	See the description above	0x??[100]	0x??[100]

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function MXCommon__GetModuleType

For new application(s) or automate communication it is recommended to use the function MXCommon__GetModuleTypeEx.

Description

Returns the type of the MSX-E Module

Parameters:

[Response frame layout] ***str:*** A 200-characters string

Query frame layout

Field	Size (Bytes)	Type	Value	little endian	big endian (Motorola)

				(Intel)	
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	1	0x0100	0x0001
word count	2	16-bit integer	100	0x6400	0x0064

Response frame layout

Field	**Size (Bytes)**	**Type**	**Value**	**little endian (Intel)**	**big endian (Motorola)**
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	204	0xCC00	0x00CC
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	2	16-bit integer	200	0xC800	0x00C8
str	200	8-bit integer array	See the description above	0x??[200]	0x??[200]

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function MXCommon__GetModuleTypeEx

Description

Returns the type of the MSX-E Module

Parameters:

[Response frame layout] ***str:*** A 200-characters string

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006

unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	10200	0xD827	0x27D8
word count	2	16-bit integer	100	0x6400	0x0064

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	203	0xCB00	0x00CB
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	1	8-bit integer	200	0xC8	0xC8
str	200	8-bit integer array	See the description above	0x??[200]	0x??[200]

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
	1		0 or 1		

unit identifier		8-bit integer		0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function MXCommon__GetTime

For new application(s) or automate communication it is recommended to use the function MXCommon__GetTimeEx.

Description

Get the time on the module

Parameters:

[Response frame layout] ***tv_sec:*** Number of seconds since the Epoch

[Response frame layout] ***tv_usec:*** Number of microseconds since the begin of the second

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	2	0x0200	0x0002

word count	2	16-bit integer	4	0x0400	0x0004

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	12	0x0C00	0x000C
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	2	16-bit integer	8	0x0800	0x0008
tv_sec	4	32-bit integer	See the description above	0x????????	0x????????
tv_usec	4	32-bit integer	See the description above	0x????????	0x????????

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
	1			??	??

Exception code		8-bit integer	See corresponding chapter		

Function MXCommon__GetTimeEx

Description

Get the time on the module

Parameters:

[Response frame layout] ***tv_sec:*** Number of seconds since the Epoch

[Response frame layout] ***tv_usec:*** Number of microseconds since the begin of the second

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	10500	0x0429	0x2904
word count	2	16-bit integer	4	0x0400	0x0004

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
	2			0x0000	0x0000

transaction identifier		16-bit integer	User defined - copied by server - usually 0		
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	11	0x0B00	0x000B
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	1	8-bit integer	8	0x08	0x08
tv_sec	4	32-bit integer	See the description above	0x????????	0x????????
tv_usec	4	32-bit integer	See the description above	0x????????	0x????????

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function MXCommon__TestCustomerID

For new application(s) or automate communication it is recommended to use the function MXCommon__TestCustomerIDEx.

Description

Permit to test the Customer ID (if the module has the right customer Key)

Parameters:

[Response frame layout] ***bValueArray:*** non crypted value array [16 bytes of random data]

[Response frame layout] ***bCryptedValueArray:*** Crypted value array [16 bytes of the crypted random data]

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	3	0x0300	0x0003
word count	2	16-bit integer	16	0x1000	0x0010

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by	0x0000	0x0000

			server - usually 0		
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	36	0x2400	0x0024
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	2	16-bit integer	32	0x2000	0x0020
bValueArray	16	8-bit integer array	See the description above	0x??[16]	0x??[16]
bCryptedValueArray	16	8-bit integer array	See the description above	0x??[16]	0x??[16]

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function MXCommon__TestCustomerIDEx

Description

Permit to test the Customer ID (if the module has the right customer Key)

Parameters:

[Response frame layout] ***bValueArray:*** non crypted value array [16 bytes of random data]

[Response frame layout] ***bCryptedValueArray:*** Crypted value array [16 bytes of the crypted random data]

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	10550	0x3629	0x2936
word count	2	16-bit integer	16	0x1000	0x0010

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	35	0x2300	0x0023
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03

Byte count	1	8-bit integer	32	0x20	0x20
bValueArray	16	8-bit integer array	See the description above	0x??[16]	0x??[16]
bCryptedValueArray	16	8-bit integer array	See the description above	0x??[16]	0x??[16]

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function MSXE360X__AnalogInputGetSequenceStatus

For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputGetSequenceStatusEx.

Description

Returns the sequence status.

Parameters:

- [Query frame layout] _: No input parameters
- [Response frame layout] ***ReturnValue:*** The return value of the remote function.
  - 0: means the remote function performed OK.
  - -1: Means an system error occured.
  - -2: PLD is not working.
  - -3: Error, system is in calibration.
  - -4: Driver status is wrong.

- -100: Internal system error occurs. See value of syserrno.

- [Response frame layout] ***ulStatus:*** Sequence status
  - 0: Disabled
  - 1: Enabled (in progress)
  - 2: End of the sequence
  - 3: Enabled, waiting for trigger
  - 4: PLD overflow
  - 5: Internal FIFO overflow

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus):

◊ ***syserrno :*** system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	100	0x6400	0x0064
word count	2	16-bit integer	2	0x0200	0x0002

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server -	0x0000	0x0000

			usually 0		
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	8	0x0800	0x0008
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	2	16-bit integer	4	0x0400	0x0004
ulStatus	4	32-bit integer	See the description above	0x????????	0x????????

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function MSXE360X__AnalogInputGetSequenceStatusEx

Description

Returns the sequence status.

Parameters:

- [Query frame layout] _: No input parameters
- [Response frame layout] ***ReturnValue:*** The return value of the remote function.

- ◊ 0: means the remote function performed OK.
- ◊ -1: Means an system error occured.
- ◊ -2: PLD is not working.
- ◊ -3: Error, system is in calibration.
- ◊ -4: Driver status is wrong.
- ◊ -100: Internal system error occurs. See value of syserrno.

♦ [Response frame layout] ***ulStatus:*** Sequence status

- ◊ 0: Disabled
- ◊ 1: Enabled (in progress)
- ◊ 2: End of the sequence
- ◊ 3: Enabled, waiting for trigger
- ◊ 4: PLD overflow
- ◊ 5: Internal FIFO overflow

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx):

· ***syserrno :*** system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	1000	0xE803	0x03E8
word count	2	16-bit integer	2	0x0200	0x0002

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	7	0x0700	0x0007
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	1	8-bit integer	4	0x04	0x04
ulStatus	4	32-bit integer	See the description above	0x????????	0x????????

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function MSXE360X__AnalogInputGetSequenceConfiguration

For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputGetSequenceConfigurationEx.

Description

Returns the sequence configuration.

Parameters:

◊ [Query frame layout] _: No input parameters
◊ [Response frame layout] ***ReturnValue:*** The return value of the remote function.
- · 0: means the remote function performed OK.
- · -1: Means an system error occured.
- · -2: PLD is not working.
- · -3: Error, system is in calibration.
- · -4: Driver status is wrong.
- · -100: Internal system error occurs. See value of syserrno.

◊ [Response frame layout] ***ulChannelMask***:

8 bits mask (0->0xFF) which defines channels used for the acquisition (each bit corresponds to one channel)

- · 0: not used
- · 1: used

◊ [Response frame layout] ***ulNbrOfSequence***:

Number of sequence to acquire

- · 0: Continuous mode
- · >0: Number of sequences (1 -> 4294967295 (0xFFFFFFFF))

◊ [Response frame layout] ***ulNbrMaxSequenceToTransfer***: Not used, must be 0
◊ [Response frame layout] ***dFrequencySelection***: Select the frequency of the acquisition:
- · 1000.00 Hz
- · 1280.00 Hz
- · 1562.50 Hz
- · 1600.00 Hz
- · 1666.67 Hz
- · 12000.00 Hz
- · 12500.00 Hz
- · 13125.00 Hz
- · 13200.00 Hz
- · 13333.33 Hz
- · 14000.00 Hz
- · 15000.00 Hz

- 16250.00 Hz
- 16400.00 Hz
- 16666.67 Hz
- 18000.00 Hz
- 110000.00 Hz
- 112500.00 Hz
- 112800.00 Hz
- 113333.33 Hz
- 116000.00 Hz
- 116666.67 Hz
- 120000.00 Hz
- 125000.00 Hz
- 132000.00 Hz
- 133333.33 Hz
- 140000.00 Hz
- 150000.00 Hz
- 164000.00 Hz
- 166666.67 Hz
- 180000.00 Hz
- 1100000.00 Hz
- 1128000.00 Hz

◊ [Response frame layout] ***pulGainArray***:

Define the gain (1, 10 or 100) to use for each channel.
Each index of the array corresponds to the corresponding channel.
Example:

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

◊ [Response frame layout] ***ulICPMask***:

8 bits mask (0 -> 0xFF) which defines if the ICP is activated or not (each bit correspond to one channel).
When the ICP is activated, the chanel must be configured with AC and SE.

- 0: Not activated
- 1: Activated

◊ [Response frame layout] ***ulTriggerMask***: Define the source of the trigger

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

◊ [Response frame layout] ***ulTriggerMode***: Not used, must be 0

◊ [Response frame layout] **ulHardwareTriggerEdge**: Defines the edge of the trigger

- 1: Hardware Trigger Rising Edge
- 2: Hardware Trigger Falling Edge
- 3: Enable both rising edge and falling edge

◊ [Response frame layout] ***ulHardwareTriggerCount***: Defines the number of external trigger ignored before taking account (1 -> 65535)

◊ [Response frame layout] ***ulByTriggerNbrOfSeqToAcquire***: Not used, must be 0

◊ [Response frame layout] ***ulDataFormat***: Dataformat of the frame, see remarks and examples for more information:

· D0: Absolute time stamp information (232 bits data)
- 0: No time stamp information
- 1: Time stamp information

· D1: Not used, must be 0

· D2: Sequence counter (32 bits data)
- 0: No sequence counter information required
- 1: Sequence counter information required

· D3: Hardware Trigger information (32 bits data)
- 0: No Hardware Trigger information required
- 1: Hardware Trigger information required

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

· 0: AC

· 1: DC

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

· 0: SE

· 1: DIFF

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus):

· ***syserrno :*** system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number	2	16-bit integer	101	0x6500	0x0065

(=register)					
word count	2	16-bit integer	42	0x2A00	0x002A

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	88	0x5800	0x0058
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	2	16-bit integer	84	0x5400	0x0054
ulChannelMask	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrOfSequence	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrMaxSequenceToTransfer	4	32-bit integer	See the description above	0x????????	0x????????
dFrequencySelection	4	32-bit floating point	See the description above	0x????????	0x????????
pulGainArray	32	32-bit integer array	See the description above	0x????????[8]	0x????????[8]
ulICPMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMode	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerEdge	4	32-bit integer	See the description	0x????????	0x????????

			above		
ulHardwareTriggerCount	4	32-bit integer	See the description above	0x????????	0x????????
ulByTriggerNbrOfSeqToAcquire	4	32-bit integer	See the description above	0x????????	0x????????
ulDataFormat	4	32-bit integer	See the description above	0x????????	0x????????
ulCouplingSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????
ulSeDiffSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

Function MSXE360X__AnalogInputGetSequenceConfigurationEx

Description

Returns the sequence configuration.

Parameters:

◊ [Query frame layout] _: No input parameters

◊ [Response frame layout] ***ReturnValue:*** The return value of the remote function.
- · 0: means the remote function performed OK.
- · -1: Means an system error occured.
- · -2: PLD is not working.
- · -3: Error, system is in calibration.
- · -4: Driver status is wrong.
- · -100: Internal system error occurs. See value of syserrno.

◊ [Response frame layout] ***ulChannelMask***:

8 bits mask (0->0xFF) which defines channels used for the acquisition (each bit corresponds to one channel)

- · 0: not used
- · 1: used

◊ [Response frame layout] ***ulNbrOfSequence***:

Number of sequence to acquire

- · 0: Continuous mode
- · >0: Number of sequences (1 -> 4294967295 (0xFFFFFFFF))

◊ [Response frame layout] ***ulNbrMaxSequenceToTransfer***: Not used, must be 0

◊ [Response frame layout] ***dFrequencySelection***: Select the frequency of the acquisition:
- · 1000.00 Hz
- · 1280.00 Hz
- · 1562.50 Hz
- · 1600.00 Hz
- · 1666.67 Hz
- · 12000.00 Hz
- · 12500.00 Hz
- · 13125.00 Hz
- · 13200.00 Hz
- · 13333.33 Hz
- · 14000.00 Hz
- · 15000.00 Hz
- · 16250.00 Hz
- · 16400.00 Hz
- · 16666.67 Hz
- · 18000.00 Hz
- · 110000.00 Hz
- · 112500.00 Hz
- · 112800.00 Hz
- · 113333.33 Hz
- · 116000.00 Hz
- · 116666.67 Hz
- · 120000.00 Hz
- · 125000.00 Hz
- · 132000.00 Hz
- · 133333.33 Hz
- · 140000.00 Hz
- · 150000.00 Hz

- 164000.00 Hz
- 166666.67 Hz
- 180000.00 Hz
- 1100000.00 Hz
- 1128000.00 Hz

◊ [Response frame layout] ***pulGainArray***:

Define the gain (1, 10 or 100) to use for each channel.
Each index of the array corresponds to the corresponding channel.
Example:

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

◊ [Response frame layout] ***ulICPMask***:

8 bits mask (0 -> 0xFF) which defines if the ICP is activated or not (each bit correspond to one channel).
When the ICP is activated, the chanel must be configured with AC and SE.

- 0: Not activated
- 1: Activated

◊ [Response frame layout] ***ulTriggerMask***: Define the source of the trigger
- 0: Trigger disabled
- 1: Enable Hardware Digital Input Trigger
- 2: Enable Synchro Trigger
- 3: Enable both Hardware and Synchro Trigger

◊ [Response frame layout] ***ulTriggerMode***: Not used, must be 0
◊ [Response frame layout] **ulHardwareTriggerEdge**: Defines the edge of the trigger
- 1: Hardware Trigger Rising Edge
- 2: Hardware Trigger Falling Edge
- 3: Enable both rising edge and falling edge

◊ [Response frame layout] ***ulHardwareTriggerCount***: Defines the number of external trigger ignored before taking account (1 -> 65535)
◊ [Response frame layout] ***ulByTriggerNbrOfSeqToAcquire***: Not used, must be 0
◊ [Response frame layout] ***ulDataFormat***: Dataformat of the frame, see remarks and examples for more information:
- D0: Absolute time stamp information (232 bits data)
  - 0: No time stamp information
  - 1: Time stamp information
- D1: Not used, must be 0
- D2: Sequence counter (32 bits data)
  - 0: No sequence counter information required
  - 1: Sequence counter information required
- D3: Hardware Trigger information (32 bits data)
  - 0: No Hardware Trigger information required
  - 1: Hardware Trigger information required

◊ [Response frame layout] ***ulCouplingSelectionMask***: 8 bits mask (0 -> 0xFF) which defines the coupling for each channel (each bit corresponds to one channel)
- 0: AC
- 1: DC

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

· 0: SE

· 1: DIFF

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx):

· ***syserrno :*** system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Reference number (=register)	2	16-bit integer	1050	0x1A04	0x041A
word count	2	16-bit integer	42	0x2A00	0x002A

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2		87	0x5700	0x0057

		16-bit integer			
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x03	0x03	0x03
Byte count	1	8-bit integer	84	0x54	0x54
ulChannelMask	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrOfSequence	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrMaxSequenceToTransfer	4	32-bit integer	See the description above	0x????????	0x????????
dFrequencySelection	4	32-bit floating point	See the description above	0x????????	0x????????
pulGainArray	32	32-bit integer array	See the description above	0x????????[8]	0x????????[8]
ulICPMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMode	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerEdge	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerCount	4	32-bit integer	See the description above	0x????????	0x????????
ulByTriggerNbrOfSeqToAcquire	4	32-bit integer	See the description above	0x????????	0x????????
ulDataFormat	4	32-bit integer	See the description above	0x????????	0x????????
ulCouplingSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????
ulSeDiffSelectionMask	4	32-bit integer	See the description	0x????????	0x????????

			above		

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x83	0x83	0x83
Exception code	1	8-bit integer	See corresponding chapter	??	??

FC16 (write multiple register) Functions

Top

Functions in this group are used to set value on the module.

◊ MXCommon__SetHardwareTriggerFilterTime Register: **100**
◊ MXCommon__SetHardwareTriggerFilterTimeEx Register: **11000**
◊ MXCommon__InitAndStartSynchroTimer Register: **101**
◊ MXCommon__InitAndStartSynchroTimerEx Register: **11050**
◊ MXCommon__StopAndReleaseSynchroTimer Register: **102**
◊ MXCommon__StopAndReleaseSynchroTimerEx Register: **11100**
◊ MXCommon__Reboot Register: **103**
◊ MXCommon__RebootEx Register: **11150**
◊ MXCommon__SetCustomerKey Register: **104**
◊ MXCommon__SetCustomerKeyEx Register: **11200**
◊ MXCommon__SetFilterChannels Register: **105**
◊ MXCommon__SetFilterChannelsEx Register: **11250**
◊ MSXE360X__AnalogInputInitSequence Register: **1**
◊ MSXE360X__AnalogInputInitSequenceEx Register: **1100**
◊ MSXE360X__AnalogInputStartSequence Register: **2**
◊ MSXE360X__AnalogInputStartSequenceEx Register: **1150**
◊ MSXE360X__AnalogInputInitAndStartSequence Register: **3**
◊ MSXE360X__AnalogInputInitAndStartSequenceEx Register: **1200**
◊ MSXE360X__AnalogInputStopSequence Register: **4**
◊ MSXE360X__AnalogInputStopSequenceEx Register: **1250**
◊ MSXE360X__AnalogInputReleaseSequence Register: **5**
◊ MSXE360X__AnalogInputReleaseSequenceEx Register: **1300**
◊ MSXE360X__AnalogInputStopAndReleaseSequence Register: **6**
◊ MSXE360X__AnalogInputStopAndReleaseSequenceEx Register: **1350**

Function MXCommon__SetHardwareTriggerFilterTime

For new application(s) or automate communication it is recommended to use the function MXCommon__SetHardwareTriggerFilterTimeEx.

Description

Sets 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

◊ [Query frame layout] ***ulFilterTime*** Filter time for the hardware trigger input in 250ns step (max value : 65535).
- · **0**: disable the filter
- · **1**: filter of 250ns
- · **2**: filter of 500ns
- · ...
- · **65535**: filter of 16ms

◊ [Query frame layout] ***ulOption*** Reserved. Set to 0

Returns

Possible return value on the remote system (read them with GetLastCommandStatus).

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

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	16	0x1000	0x0010
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	100	0x6400	0x0064
word count	2	16-bit integer	4	0x0400	0x0004
byte count	2	16-bit integer	8	0x0800	0x0008
ulFilterTime	4	32-bit integer	See the description above	0x????????	0x????????
Reserved	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	100	0x6400	0x0064
word count	2	16-bit integer	4	0x0400	0x0004

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__SetHardwareTriggerFilterTimeEx

Description

Sets 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

◊ [Query frame layout] ***ulFilterTime*** Filter time for the hardware trigger input in 250ns step (max value : 65535).
- · **0**: disable the filter
- · **1**: filter of 250ns
- · **2**: filter of 500ns
- · ...
- · **65535**: filter of 16ms

◊ [Query frame layout] ***ulOption*** Reserved. Set to 0

Returns

Possible return value on the remote system (read them with GetLastCommandStatusEx).

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

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	15	0x0F00	0x000F
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11000	0xF82A	0x2AF8
word count	2		4	0x0400	0x0004

		16-bit integer			
byte count	1	8-bit integer	8	0x08	0x08
ulFilterTime	4	32-bit integer	See the description above	0x????????	0x????????
Reserved	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11000	0xF82A	0x2AF8
word count	2	16-bit integer	4	0x0400	0x0004

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit	3	0x0300	0x0003

		integer			
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__InitAndStartSynchroTimer

For new application(s) or automate communication it is recommended to use the function MXCommon__InitAndStartSynchroTimerEx.

Description

Init and start the synchronisation timer of the module (not already available on all module)

Parameters:

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

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

[Query frame layout] ***ulNbrOfCycle:*** Number of timer cycle

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

[Query frame layout] ***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

[Query frame layout] ***ulOption01:***Define the source of the trigger

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

[Query frame layout] ***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)

[Query frame layout] ***ulOption03:***Define the number of trigger events before the action occur

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

[Query frame layout] ***ulOption04:*** Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus)

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

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	40	0x2800	0x0028
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	101	0x6500	0x0065
word count	2	16-bit integer	16	0x1000	0x0010
byte count	2	16-bit integer	32	0x2000	0x0020
ulTimeBase	4	32-bit integer	See the description above	0x????????	0x????????
ulReloadValue	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrOfCycle	4			0x????????	0x????????

		32-bit integer	See the description above		
ulGenerateTriggerMode	4	32-bit integer	See the description above	0x????????	0x????????
ulOption01	4	32-bit integer	See the description above	0x????????	0x????????
ulOption02	4	32-bit integer	See the description above	0x????????	0x????????
ulOption03	4	32-bit integer	See the description above	0x????????	0x????????
ulOption04	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	101	0x6500	0x0065
word count	2	16-bit integer	16	0x1000	0x0010

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)

transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__InitAndStartSynchroTimerEx

Description

Init and start the synchronisation timer of the module (not already available on all module)

Parameters:

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

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

[Query frame layout] ***ulNbrOfCycle:*** Number of timer cycle

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

[Query frame layout] ***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

[Query frame layout] ***ulOption01:***Define the source of the trigger

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

[Query frame layout] ***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)

[Query frame layout] ***ulOption03:*** Define the number of trigger events before the action occur

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

[Query frame layout] ***ulOption04:*** Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx)

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

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	39	0x2700	0x0027
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11050	0x2A2B	0x2B2A
word count	2	16-bit integer	16	0x1000	0x0010
byte count	1	8-bit integer	32	0x20	0x20
ulTimeBase	4	32-bit integer	See the description	0x????????	0x????????

			above		
ulReloadValue	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrOfCycle	4	32-bit integer	See the description above	0x????????	0x????????
ulGenerateTriggerMode	4	32-bit integer	See the description above	0x????????	0x????????
ulOption01	4	32-bit integer	See the description above	0x????????	0x????????
ulOption02	4	32-bit integer	See the description above	0x????????	0x????????
ulOption03	4	32-bit integer	See the description above	0x????????	0x????????
ulOption04	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11050	0x2A2B	0x2B2A
word count	2	16-bit integer	16	0x1000	0x0010

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__StopAndReleaseSynchroTimer

For new application(s) or automate communication it is recommended to use the function MXCommon__StopAndReleaseSynchroTimerEx.

Description

stop the synchronisation timer (not already available on all module)

Parameters:

[Query frame layout] ***ulOption01*** : Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus)

· 0 : means the remote function performed OK
· -1: means an system error occured
· -100: Start/Stop timer error

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	12	0x0C00	0x000C
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	102	0x6600	0x0066
word count	2	16-bit integer	2	0x0200	0x0002
byte count	2	16-bit integer	4	0x0400	0x0004
ulOption01	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS	1	8-bit	0x10	0x10	0x10

Function code		integer			
Reference number (=register)	2	16-bit integer	102	0x6600	0x0066
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__StopAndReleaseSynchroTimerEx

Description

stop the synchronisation timer (not already available on all module)

Parameters:

[Query frame layout] ***ulOption01*** : Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx)

· 0 : means the remote function performed OK
· -1: means an system error occured
· -100: Start/Stop timer error

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	11	0x0B00	0x000B
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11100	0x5C2B	0x2B5C
word count	2	16-bit integer	2	0x0200	0x0002
byte count	1	8-bit integer	4	0x04	0x04
ulOption01	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS	1	8-bit	0x10	0x10	0x10

Function code		integer			
Reference number (=register)	2	16-bit integer	11100	0x5C2B	0x2B5C
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__Reboot

For new application(s) or automate communication it is recommended to use the function MXCommon__RebootEx.

Description

Ask the MSX-E module to reboot

Parameters:

[Query frame layout] ***Dummy*** : Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus)

· 0 : means the remote function performed OK

· -1: means an system error occured (probably EPERM)

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	12	0x0C00	0x000C
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	103	0x6700	0x0067
word count	2	16-bit integer	2	0x0200	0x0002
byte count	2	16-bit integer	4	0x0400	0x0004
Dummy	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01

MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	103	0x6700	0x0067
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__RebootEx

Description

Ask the MSX-E module to reboot

Parameters:

[Query frame layout] ***Dummy*** : Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx)

· 0 : means the remote function performed OK
· -1: means an system error occured (probably EPERM)

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	11	0x0B00	0x000B
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11150	0x8E2B	0x2B8E
word count	2	16-bit integer	2	0x0200	0x0002
byte count	1	8-bit integer	4	0x04	0x04
Dummy	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS	1	8-bit	0x10	0x10	0x10

Function code		integer			
Reference number (=register)	2	16-bit integer	11150	0x8E2B	0x2B8E
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__SetCustomerKey

For new application(s) or automate communication it is recommended to use the function MXCommon__SetCustomerKeyEx.

Description

Permit to set the Customer key

Parameters:

[Query frame layout] ***bKey*** : Customer key (only writable on the module) [32 bytes containing a AES key]

[Query frame layout] ***bPublicKey*** : IV (Initialisation vector) for the AES cryptography [16 bytes containing a AES key]

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus)

- 0 : means the remote function performed OK
- -1: means an system error occured (probably EPERM)

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	56	0x3800	0x0038
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	104	0x6800	0x0068
word count	2	16-bit integer	24	0x1800	0x0018
byte count	2	16-bit integer	48	0x3000	0x0030
bKey	32	8-bit integer array	See the description above	0x??[32]	0x??[32]
bPublicKey	16	8-bit integer array	See the description above	0x??[16]	0x??[16]

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000

protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	104	0x6800	0x0068
word count	2	16-bit integer	24	0x1800	0x0018

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__SetCustomerKeyEx

Description

Permit to set the Customer key

Parameters:

[Query frame layout] ***bKey*** : Customer key (only writable on the module) [32 bytes containing a AES key]

[Query frame layout] ***bPublicKey*** : IV (Initialisation vector) for the AES cryptography [16 bytes containing a AES key]

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx)

· 0 : means the remote function performed OK
· -1: means an system error occured (probably EPERM)

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	55	0x3700	0x0037
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11200	0xC02B	0x2BC0
word count	2	16-bit integer	24	0x1800	0x0018
byte count	1	8-bit integer	48	0x30	0x30
bKey	32	8-bit integer array	See the description above	0x??[32]	0x??[32]
bPublicKey	16	8-bit integer array	See the description above	0x??[16]	0x??[16]

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined	0x0000	0x0000

			- copied by server - usually 0		
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11200	0xC02B	0x2BC0
word count	2	16-bit integer	24	0x1800	0x0018

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__SetFilterChannels

For new application(s) or automate communication it is recommended to use the function MXCommon__SetFilterChannelsEx.

Description

Permit to set a filter per channel

Parameters:

[Query frame layout] ***ChannelList*** : Each index of the array is representing a channel. To set a filter on a channel, enter the filter ID. By default the value ist 0 (No filter).

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus)

· 0 : means the remote function performed OK
· -1: means a system error occurred (probably EPERM)

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	24	0x1800	0x0018
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	105	0x6900	0x0069
word count	2	16-bit integer	8	0x0800	0x0008
byte count	2	16-bit integer	16	0x1000	0x0010
ChannelList	16	8-bit integer array	See the description above	0x??[16]	0x??[16]

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	105	0x6900	0x0069
word count	2	16-bit integer	8	0x0800	0x0008

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MXCommon__SetFilterChannelsEx

Description

Permit to set a filter per channel

Parameters:

[Query frame layout] ***ChannelList*** : Each index of the array is representing a channel. To set a filter on a channel, enter the filter ID. By default the value ist 0 (No filter).

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx)

· 0 : means the remote function performed OK
· -1: means a system error occurred (probably EPERM)

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	23	0x1700	0x0017
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11250	0xF22B	0x2BF2
word count	2	16-bit integer	8	0x0800	0x0008
byte count	1	8-bit integer	16	0x10	0x10
ChannelList	16	8-bit integer array	See the description above	0x??[16]	0x??[16]

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	11250	0xF22B	0x2BF2
word count	2	16-bit integer	8	0x0800	0x0008

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputInitSequence

For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputInitSequenceEx.

Description

Initialize the analog input sequence acquisition mode.
Do not call when a sequence is already started.

A sequence is a list of channels (max 8) that are acquired.

Parameters:

◊ [Query frame layout] ***ulChannelMask***:

8 bits mask (0->0xFF) which defines channels used for the acquisition (each bit corresponds to one channel)

- 0: not used
- 1: used

◊ [Query frame layout] ***ulNbrOfSequence***:

Number of sequence to acquire

- 0: Continuous mode
- >0: Number of sequences (1 -> 4294967295 (0xFFFFFFFF))

◊ [Query frame layout] ***ulNbrMaxSequenceToTransfer***: Not used, must be 0

◊ [Query frame layout] ***dFrequencySelection***: Select the frequency of the acquisition:

- 1000.00 Hz
- 1280.00 Hz
- 1562.50 Hz
- 1600.00 Hz
- 1666.67 Hz
- 12000.00 Hz
- 12500.00 Hz
- 13125.00 Hz
- 13200.00 Hz
- 13333.33 Hz
- 14000.00 Hz
- 15000.00 Hz
- 16250.00 Hz
- 16400.00 Hz
- 16666.67 Hz
- 18000.00 Hz
- 110000.00 Hz
- 112500.00 Hz

- 112800.00 Hz
- 113333.33 Hz
- 116000.00 Hz
- 116666.67 Hz
- 120000.00 Hz
- 125000.00 Hz
- 132000.00 Hz
- 133333.33 Hz
- 140000.00 Hz
- 150000.00 Hz
- 164000.00 Hz
- 166666.67 Hz
- 180000.00 Hz
- 1100000.00 Hz
- 1128000.00 Hz

◊ [Query frame layout] ***pulGainArray***:

Define the gain (1, 10 or 100) to use for each channel.
Each index of the array corresponds to the corresponding channel.
Example:

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

◊ [Query frame layout] ***ulICPMask***:

8 bits mask (0 -> 0xFF) which defines if the ICP is activated or not (each bit correspond to one channel).
When the ICP is activated, the chanel must be configured with AC and SE.

- 0: Not activated
- 1: Activated

◊ [Query frame layout] ***ulTriggerMask***: Define the source of the trigger
- 0: Trigger disabled
- 1: Enable Hardware Digital Input Trigger
- 2: Enable Synchro Trigger
- 3: Enable both Hardware and Synchro Trigger

◊ [Query frame layout] ***ulTriggerMode***: Not used, must be 0
◊ [Query frame layout] **ulHardwareTriggerEdge**: Defines the edge of the trigger
- 1: Hardware Trigger Rising Edge
- 2: Hardware Trigger Falling Edge
- 3: Enable both rising edge and falling edge

◊ [Query frame layout] ***ulHardwareTriggerCount***: Defines the number of external trigger ignored before taking account (1 -> 65535)
◊ [Query frame layout] ***ulByTriggerNbrOfSeqToAcquire***: Not used, must be 0
◊ [Query frame layout] ***ulDataFormat***: Dataformat of the frame, see remarks and examples for more information:
- D0: Absolute time stamp information (232 bits data)
  - 0: No time stamp information
  - 1: Time stamp information
- D1: Not used, must be 0

- D2: Sequence counter (32 bits data)
  - 0: No sequence counter information required
  - 1: Sequence counter information required
- D3: Hardware Trigger information (32 bits data)
  - 0: No Hardware Trigger information required
  - 1: Hardware Trigger information required

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

- 0: AC
- 1: DC

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

- 0: SE
- 1: DIFF

Remarks:

◊ data packets depends on the (number of sequence asked) (sequence size in 32 bits words) 4

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

◊ The data order is

- Timestamp seconds (optional)
- Timestamp microseconds (optional)
- Sequence counter (optional)
- Hardware trigger information (optional)
- Selected channels in ascending order

◊ Sequence size in bytes: [(timestamp (s) + timestamp (us)) (optional) + sequence counter (optional) + hardware trigger information (optional) + number of channels] 4

Examples:

◊ 4 channels in continuous mode: packets of 8192 bytes are sent -> 512 sequences of 4 32 bits channels per packet

◊ 10 sequences of 4 channels with timestamp and hardware trigger information: 1 packet of 280 bytes is sent by the MSXE -> 10 sequences of : timestamp (232 bits word) + sequence counter size (32 bits) + 4 32 bits channels)

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus):

- 0: means the remote function performed OK
- -1: means an system error occured
- -2: PLD is not working
- -3: error, system is in calibration
- -4: channel action is wrong
- -5: gain selection error
- -6: channel coupling selection error
- -7: SE / Diff selection error
- -8: ICP selection error

· -9: ICP can only be used with AC and SE
· -10: driver is not in idle state
· -11: PLD is not working
· -12: error, system is in calibration
· -13: Frequency selection error
· -14: driver is not in idle state
· -19: channel mask can not be null
· -20: channel mask selection error
· -21: number of sequence selection error
· -22: sequence interrupt selection error (must be 0)
· -23: the ulTriggerMode parameter is wrong (must be 0)
· -24: the ulHardwareTriggerEdge parameter is wrong
· -25: the ulHardwareTriggerCount parameter is wrong
· -26: the ulByTriggerNbrOfSeqToAcquire parameter is wrong (must be 0)
· -27: the ulDataFormat parameter is wrong
· -28: the ulTriggerMask parameter is wrong
· -29: the ulICPMask is wrong (0->0xFF)
· -30: the ulCouplingSelectionMask is wrong (0->0xFF)
· -31: the ulSeDiffSelectionMask is wrong (0->0xFF)
· -40: PLD is not working
· -41: error, system is in calibration
· -42: driver status is wrong
· -100 internal system error occurs see value of syserrno

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	92	0x5C00	0x005C
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1	0x0100	0x0001
word count	2	16-bit integer	42	0x2A00	0x002A
byte count	2	16-bit integer	84	0x5400	0x0054

ulChannelMask	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrOfSequence	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrMaxSequenceToTransfer	4	32-bit integer	See the description above	0x????????	0x????????
dFrequencySelection	4	32-bit floating point	See the description above	0x????????	0x????????
pulGainArray	32	32-bit integer array	See the description above	0x????????[8]	0x????????[8]
ulICPMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMode	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerEdge	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerCount	4	32-bit integer	See the description above	0x????????	0x????????
ulByTriggerNbrOfSeqToAcquire	4	32-bit integer	See the description above	0x????????	0x????????
ulDataFormat	4	32-bit integer	See the description above	0x????????	0x????????
ulCouplingSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????
ulSeDiffSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined	0x0000	0x0000

			- copied by server - usually 0		
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1	0x0100	0x0001
word count	2	16-bit integer	42	0x2A00	0x002A

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputInitSequenceEx

Description

Initialize the analog input sequence acquisition mode.
Do not call when a sequence is already started.

A sequence is a list of channels (max 8) that are acquired.

Parameters:

◊ [Query frame layout] ***ulChannelMask***:

8 bits mask (0->0xFF) which defines channels used for the acquisition (each bit corresponds to one channel)

· 0: not used
· 1: used

◊ [Query frame layout] ***ulNbrOfSequence***:

Number of sequence to acquire

· 0: Continuous mode
· >0: Number of sequences (1 -> 4294967295 (0xFFFFFFFF))

◊ [Query frame layout] ***ulNbrMaxSequenceToTransfer***: Not used, must be 0

◊ [Query frame layout] ***dFrequencySelection***: Select the frequency of the acquisition:

· 1000.00 Hz
· 1280.00 Hz
· 1562.50 Hz
· 1600.00 Hz
· 1666.67 Hz
· 12000.00 Hz
· 12500.00 Hz
· 13125.00 Hz
· 13200.00 Hz
· 13333.33 Hz
· 14000.00 Hz
· 15000.00 Hz
· 16250.00 Hz
· 16400.00 Hz
· 16666.67 Hz
· 18000.00 Hz
· 110000.00 Hz
· 112500.00 Hz
· 112800.00 Hz
· 113333.33 Hz
· 116000.00 Hz
· 116666.67 Hz
· 120000.00 Hz
· 125000.00 Hz
· 132000.00 Hz
· 133333.33 Hz
· 140000.00 Hz
· 150000.00 Hz
· 164000.00 Hz
· 166666.67 Hz
· 180000.00 Hz

- 1100000.00 Hz
- 1128000.00 Hz

◊ [Query frame layout] ***pulGainArray***:

Define the gain (1, 10 or 100) to use for each channel.
Each index of the array corresponds to the corresponding channel.
Example:

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

◊ [Query frame layout] ***ulICPMask***:

8 bits mask (0 -> 0xFF) which defines if the ICP is activated or not (each bit correspond to one channel).
When the ICP is activated, the chanel must be configured with AC and SE.

- 0: Not activated
- 1: Activated

◊ [Query frame layout] ***ulTriggerMask***: Define the source of the trigger
- 0: Trigger disabled
- 1: Enable Hardware Digital Input Trigger
- 2: Enable Synchro Trigger
- 3: Enable both Hardware and Synchro Trigger

◊ [Query frame layout] ***ulTriggerMode***: Not used, must be 0
◊ [Query frame layout] **ulHardwareTriggerEdge**: Defines the edge of the trigger
- 1: Hardware Trigger Rising Edge
- 2: Hardware Trigger Falling Edge
- 3: Enable both rising edge and falling edge

◊ [Query frame layout] ***ulHardwareTriggerCount***: Defines the number of external trigger ignored before taking account (1 -> 65535)
◊ [Query frame layout] ***ulByTriggerNbrOfSeqToAcquire***: Not used, must be 0
◊ [Query frame layout] ***ulDataFormat***: Dataformat of the frame, see remarks and examples for more information:
- D0: Absolute time stamp information (232 bits data)
  - 0: No time stamp information
  - 1: Time stamp information
- D1: Not used, must be 0
- D2: Sequence counter (32 bits data)
  - 0: No sequence counter information required
  - 1: Sequence counter information required
- D3: Hardware Trigger information (32 bits data)
  - 0: No Hardware Trigger information required
  - 1: Hardware Trigger information required

◊ [Query frame layout] ***ulCouplingSelectionMask***: 8 bits mask (0 -> 0xFF) which defines the coupling for each channel (each bit corresponds to one channel)
- 0: AC
- 1: DC

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

- 1: DIFF

Remarks:

◊ data packets depends on the (number of sequence asked) (sequence size in 32 bits words) 4
- bytes, Only one packet of the corresponding size is sent
- 0 (continuous) or > 8192 bytes : packet of 8192 bytes containing the sequences are sent by the MSXE, for the last sequence, the last packet is sent with the rest of the size

◊ The data order is
- Timestamp seconds (optional)
- Timestamp microseconds (optional)
- Sequence counter (optional)
- Hardware trigger information (optional)
- Selected channels in ascending order

◊ Sequence size in bytes: [(timestamp (s) + timestamp (us)) (optional) + sequence counter (optional) + hardware trigger information (optional) + number of channels] 4

Examples:

◊ 4 channels in continuous mode: packets of 8192 bytes are sent -> 512 sequences of 4 32 bits channels per packet

◊ 10 sequences of 4 channels with timestamp and hardware trigger information: 1 packet of 280 bytes is sent by the MSXE -> 10 sequences of : timestamp (232 bits word) + sequence counter size (32 bits) + 4 32 bits channels)

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx):

- 0: means the remote function performed OK
- -1: means an system error occured
- -2: PLD is not working
- -3: error, system is in calibration
- -4: channel action is wrong
- -5: gain selection error
- -6: channel coupling selection error
- -7: SE / Diff selection error
- -8: ICP selection error
- -9: ICP can only be used with AC and SE
- -10: driver is not in idle state
- -11: PLD is not working
- -12: error, system is in calibration
- -13: Frequency selection error
- -14: driver is not in idle state
- -19: channel mask can not be null
- -20: channel mask selection error
- -21: number of sequence selection error
- -22: sequence interrupt selection error (must be 0)
- -23: the ulTriggerMode parameter is wrong (must be 0)
- -24: the ulHardwareTriggerEdge parameter is wrong
- -25: the ulHardwareTriggerCount parameter is wrong

· -26: the ulByTriggerNbrOfSeqToAcquire parameter is wrong (must be 0)
· -27: the ulDataFormat parameter is wrong
· -28: the ulTriggerMask parameter is wrong
· -29: the ulICPMask is wrong (0->0xFF)
· -30: the ulCouplingSelectionMask is wrong (0->0xFF)
· -31: the ulSeDiffSelectionMask is wrong (0->0xFF)
· -40: PLD is not working
· -41: error, system is in calibration
· -42: driver status is wrong
· -100 internal system error occurs see value of syserrno

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	91	0x5B00	0x005B
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1100	0x4C04	0x044C
word count	2	16-bit integer	42	0x2A00	0x002A
byte count	1	8-bit integer	84	0x54	0x54
ulChannelMask	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrOfSequence	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrMaxSequenceToTransfer	4	32-bit integer	See the description above	0x????????	0x????????
dFrequencySelection	4	32-bit floating point	See the description above	0x????????	0x????????
pulGainArray	32			0x????????[8]	0x????????[8]

		32-bit integer array	See the description above		
ulICPMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMode	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerEdge	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerCount	4	32-bit integer	See the description above	0x????????	0x????????
ulByTriggerNbrOfSeqToAcquire	4	32-bit integer	See the description above	0x????????	0x????????
ulDataFormat	4	32-bit integer	See the description above	0x????????	0x????????
ulCouplingSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????
ulSeDiffSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
	1		0x10	0x10	0x10

MODBUS Function code		8-bit integer			
Reference number (=register)	2	16-bit integer	1100	0x4C04	0x044C
word count	2	16-bit integer	42	0x2A00	0x002A

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputStartSequence

For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputStartSequenceEx.

Description

Start the analog input sequence acquisition mode.
Do not call when a sequence is already started ot not initalized.

A sequence is a list of channels (max 8) that are acquired.

Parameters:

◊ [Query frame layout] ***Dummy***: Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus):

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

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000

length	2	16-bit integer	12	0x0C00	0x000C
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	2	0x0200	0x0002
word count	2	16-bit integer	2	0x0200	0x0002
byte count	2	16-bit integer	4	0x0400	0x0004
Dummy	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	2	0x0200	0x0002
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)

transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputStartSequenceEx

Description

Start the analog input sequence acquisition mode.
Do not call when a sequence is already started ot not initalized.

A sequence is a list of channels (max 8) that are acquired.

Parameters:

◊ [Query frame layout] ***Dummy***: Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx):

· 0: means the remote function performed OK
· -1: means an system error occured
· -2: PLD is not working
· -3: error, system is in calibration
· -4: channel action is wrong
· -5: gain selection error
· -6: channel coupling selection error
· -7: SE / Diff selection error
· -8: ICP selection error
· -9: ICP can only be used with AC and SE
· -10: driver is not in idle state
· -11: PLD is not working
· -12: error, system is in calibration
· -13: Frequency selection error

- -14: driver is not in idle state
- -19: channel mask can not be null
- -20: channel mask selection error
- -21: number of sequence selection error
- -22: sequence interrupt selection error (must be 0)
- -23: the ulTriggerMode parameter is wrong (must be 0)
- -24: the ulHardwareTriggerEdge parameter is wrong
- -25: the ulHardwareTriggerCount parameter is wrong
- -26: the ulByTriggerNbrOfSeqToAcquire parameter is wrong (must be 0)
- -27: the ulDataFormat parameter is wrong
- -28: the ulTriggerMask parameter is wrong
- -29: the ulICPMask is wrong (0->0xFF)
- -30: the ulCouplingSelectionMask is wrong (0->0xFF)
- -31: the ulSeDiffSelectionMask is wrong (0->0xFF)
- -40: PLD is not working
- -41: error, system is in calibration
- -42: driver status is wrong
- -100 internal system error occurs see value of syserrno

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	11	0x0B00	0x000B
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1150	0x7E04	0x047E
word count	2	16-bit integer	2	0x0200	0x0002
byte count	1	8-bit integer	4	0x04	0x04
Dummy	4	32-bit integer	See the description	0x????????	0x????????

			above		

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1150	0x7E04	0x047E
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputInitAndStartSequence

For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputInitAndStartSequenceEx.

Description

Initialize and start the analog input sequence acquisition mode.
Do not call when a sequence is already started.

A sequence is a list of channels (max 8) that are acquired.

Parameters:

◊ [Query frame layout] ***ulChannelMask***:

8 bits mask (0->0xFF) which defines channels used for the acquisition (each bit corresponds to one channel)

- 0: not used
- 1: used

◊ [Query frame layout] ***ulNbrOfSequence***:

Number of sequence to acquire

- 0: Continuous mode
- >0: Number of sequences (1 -> 4294967295 (0xFFFFFFFF))

◊ [Query frame layout] ***ulNbrMaxSequenceToTransfer***: Not used, must be 0

◊ [Query frame layout] ***dFrequencySelection***: Select the frequency of the acquisition:

- 1000.00 Hz
- 1280.00 Hz
- 1562.50 Hz
- 1600.00 Hz
- 1666.67 Hz
- 12000.00 Hz
- 12500.00 Hz
- 13125.00 Hz
- 13200.00 Hz
- 13333.33 Hz
- 14000.00 Hz
- 15000.00 Hz
- 16250.00 Hz
- 16400.00 Hz
- 16666.67 Hz
- 18000.00 Hz
- 110000.00 Hz

- 112500.00 Hz
- 112800.00 Hz
- 113333.33 Hz
- 116000.00 Hz
- 116666.67 Hz
- 120000.00 Hz
- 125000.00 Hz
- 132000.00 Hz
- 133333.33 Hz
- 140000.00 Hz
- 150000.00 Hz
- 164000.00 Hz
- 166666.67 Hz
- 180000.00 Hz
- 1100000.00 Hz
- 1128000.00 Hz

◊ [Query frame layout] ***pulGainArray***:

Define the gain (1, 10 or 100) to use for each channel.
Each index of the array corresponds to the corresponding channel.
Example:

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

◊ [Query frame layout] ***ulICPMask***:

8 bits mask (0 -> 0xFF) which defines if the ICP is activated or not (each bit correspond to one channel).
When the ICP is activated, the chanel must be configured with AC and SE.

- 0: Not activated
- 1: Activated

◊ [Query frame layout] ***ulTriggerMask***: Define the source of the trigger

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

◊ [Query frame layout] ***ulTriggerMode***: Not used, must be 0

◊ [Query frame layout] **ulHardwareTriggerEdge**: Defines the edge of the trigger

- 1: Hardware Trigger Rising Edge
- 2: Hardware Trigger Falling Edge
- 3: Enable both rising edge and falling edge

◊ [Query frame layout] ***ulHardwareTriggerCount***: Defines the number of external trigger ignored before taking account (1 -> 65535)

◊ [Query frame layout] ***ulByTriggerNbrOfSeqToAcquire***: Not used, must be 0

◊ [Query frame layout] ***ulDataFormat***: Dataformat of the frame, see remarks and examples for more information:

- D0: Absolute time stamp information (232 bits data)
  - 0: No time stamp information
  - 1: Time stamp information

- D1: Not used, must be 0
- D2: Sequence counter (32 bits data)
    - 0: No sequence counter information required
    - 1: Sequence counter information required
- D3: Hardware Trigger information (32 bits data)
    - 0: No Hardware Trigger information required
    - 1: Hardware Trigger information required

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

- 0: AC
- 1: DC

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

- 0: SE
- 1: DIFF

Remarks:

◊ data packets depends on the (number of sequence asked) (sequence size in 32 bits words) 4

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

◊ The data order is

- Timestamp seconds (optional)
- Timestamp microseconds (optional)
- Sequence counter (optional)
- Hardware trigger information (optional)
- Selected channels in ascending order

◊ Sequence size in bytes: [(timestamp (s) + timestamp (us)) (optional) + sequence counter (optional) + hardware trigger information (optional) + number of channels] 4

Examples:

◊ 4 channels in continuous mode: packets of 8192 bytes are sent -> 512 sequences of 4 32 bits channels per packet

◊ 10 sequences of 4 channels with timestamp and hardware trigger information: 1 packet of 280 bytes is sent by the MSXE -> 10 sequences of : timestamp (232 bits word) + sequence counter size (32 bits) + 4 32 bits channels)

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus):

- 0: means the remote function performed OK
- -1: means an system error occured
- -2: PLD is not working
- -3: error, system is in calibration
- -4: channel action is wrong
- -5: gain selection error
- -6: channel coupling selection error
- -7: SE / Diff selection error

· -8: ICP selection error
· -9: ICP can only be used with AC and SE
· -10: driver is not in idle state
· -11: PLD is not working
· -12: error, system is in calibration
· -13: Frequency selection error
· -14: driver is not in idle state
· -19: channel mask can not be null
· -20: channel mask selection error
· -21: number of sequence selection error
· -22: sequence interrupt selection error (must be 0)
· -23: the ulTriggerMode parameter is wrong (must be 0)
· -24: the ulHardwareTriggerEdge parameter is wrong
· -25: the ulHardwareTriggerCount parameter is wrong
· -26: the ulByTriggerNbrOfSeqToAcquire parameter is wrong (must be 0)
· -27: the ulDataFormat parameter is wrong
· -28: the ulTriggerMask parameter is wrong
· -29: the ulICPMask is wrong (0->0xFF)
· -30: the ulCouplingSelectionMask is wrong (0->0xFF)
· -31: the ulSeDiffSelectionMask is wrong (0->0xFF)
· -40: PLD is not working
· -41: error, system is in calibration
· -42: driver status is wrong
· -100 internal system error occurs see value of syserrno

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	92	0x5C00	0x005C
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	3	0x0300	0x0003
word count	2	16-bit integer	42	0x2A00	0x002A
byte count	2	16-bit	84	0x5400	0x0054

		integer			
ulChannelMask	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrOfSequence	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrMaxSequenceToTransfer	4	32-bit integer	See the description above	0x????????	0x????????
dFrequencySelection	4	32-bit floating point	See the description above	0x????????	0x????????
pulGainArray	32	32-bit integer array	See the description above	0x????????[8]	0x????????[8]
ulICPMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMode	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerEdge	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerCount	4	32-bit integer	See the description above	0x????????	0x????????
ulByTriggerNbrOfSeqToAcquire	4	32-bit integer	See the description above	0x????????	0x????????
ulDataFormat	4	32-bit integer	See the description above	0x????????	0x????????
ulCouplingSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????
ulSeDiffSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
	2			0x0000	0x0000

transaction identifier		16-bit integer	User defined - copied by server - usually 0		
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	3	0x0300	0x0003
word count	2	16-bit integer	42	0x2A00	0x002A

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputInitAndStartSequenceEx

Description

Initialize and start the analog input sequence acquisition mode.
Do not call when a sequence is already started.

A sequence is a list of channels (max 8) that are acquired.

Parameters:

◊ [Query frame layout] ***ulChannelMask***:

8 bits mask (0->0xFF) which defines channels used for the acquisition (each bit corresponds to one channel)

- 0: not used
- 1: used

◊ [Query frame layout] ***ulNbrOfSequence***:

Number of sequence to acquire

- 0: Continuous mode
- >0: Number of sequences (1 -> 4294967295 (0xFFFFFFFF))

◊ [Query frame layout] ***ulNbrMaxSequenceToTransfer***: Not used, must be 0

◊ [Query frame layout] ***dFrequencySelection***: Select the frequency of the acquisition:

- 1000.00 Hz
- 1280.00 Hz
- 1562.50 Hz
- 1600.00 Hz
- 1666.67 Hz
- 12000.00 Hz
- 12500.00 Hz
- 13125.00 Hz
- 13200.00 Hz
- 13333.33 Hz
- 14000.00 Hz
- 15000.00 Hz
- 16250.00 Hz
- 16400.00 Hz
- 16666.67 Hz
- 18000.00 Hz
- 110000.00 Hz
- 112500.00 Hz
- 112800.00 Hz
- 113333.33 Hz
- 116000.00 Hz
- 116666.67 Hz
- 120000.00 Hz
- 125000.00 Hz
- 132000.00 Hz

- 133333.33 Hz
- 140000.00 Hz
- 150000.00 Hz
- 164000.00 Hz
- 166666.67 Hz
- 180000.00 Hz
- 1100000.00 Hz
- 1128000.00 Hz

◊ [Query frame layout] ***pulGainArray***:

Define the gain (1, 10 or 100) to use for each channel.
Each index of the array corresponds to the corresponding channel.
Example:

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

◊ [Query frame layout] ***ulICPMask***:

8 bits mask (0 -> 0xFF) which defines if the ICP is activated or not (each bit correspond to one channel).
When the ICP is activated, the chanel must be configured with AC and SE.

- 0: Not activated
- 1: Activated

◊ [Query frame layout] ***ulTriggerMask***: Define the source of the trigger
- 0: Trigger disabled
- 1: Enable Hardware Digital Input Trigger
- 2: Enable Synchro Trigger
- 3: Enable both Hardware and Synchro Trigger

◊ [Query frame layout] ***ulTriggerMode***: Not used, must be 0

◊ [Query frame layout] **ulHardwareTriggerEdge**: Defines the edge of the trigger
- 1: Hardware Trigger Rising Edge
- 2: Hardware Trigger Falling Edge
- 3: Enable both rising edge and falling edge

◊ [Query frame layout] ***ulHardwareTriggerCount***: Defines the number of external trigger ignored before taking account (1 -> 65535)

◊ [Query frame layout] ***ulByTriggerNbrOfSeqToAcquire***: Not used, must be 0

◊ [Query frame layout] ***ulDataFormat***: Dataformat of the frame, see remarks and examples for more information:
- D0: Absolute time stamp information (232 bits data)
  - 0: No time stamp information
  - 1: Time stamp information
- D1: Not used, must be 0
- D2: Sequence counter (32 bits data)
  - 0: No sequence counter information required
  - 1: Sequence counter information required
- D3: Hardware Trigger information (32 bits data)
  - 0: No Hardware Trigger information required
  - 1: Hardware Trigger information required

◊ [Query frame layout] ***ulCouplingSelectionMask***: 8 bits mask (0 -> 0xFF) which defines the coupling for each channel (each bit corresponds to one channel)
- 0: AC
- 1: DC

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

Remarks:

◊ data packets depends on the (number of sequence asked) (sequence size in 32 bits words) 4
- bytes, Only one packet of the corresponding size is sent
- 0 (continuous) or > 8192 bytes : packet of 8192 bytes containing the sequences are sent by the MSXE, for the last sequence, the last packet is sent with the rest of the size

◊ The data order is
- Timestamp seconds (optional)
- Timestamp microseconds (optional)
- Sequence counter (optional)
- Hardware trigger information (optional)
- Selected channels in ascending order

◊ Sequence size in bytes: [(timestamp (s) + timestamp (us)) (optional) + sequence counter (optional) + hardware trigger information (optional) + number of channels] 4

Examples:

◊ 4 channels in continuous mode: packets of 8192 bytes are sent -> 512 sequences of 4 32 bits channels per packet

◊ 10 sequences of 4 channels with timestamp and hardware trigger information: 1 packet of 280 bytes is sent by the MSXE -> 10 sequences of : timestamp (232 bits word) + sequence counter size (32 bits) + 4 32 bits channels)

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx):

- 0: means the remote function performed OK
- -1: means an system error occured
- -2: PLD is not working
- -3: error, system is in calibration
- -4: channel action is wrong
- -5: gain selection error
- -6: channel coupling selection error
- -7: SE / Diff selection error
- -8: ICP selection error
- -9: ICP can only be used with AC and SE
- -10: driver is not in idle state
- -11: PLD is not working
- -12: error, system is in calibration
- -13: Frequency selection error
- -14: driver is not in idle state

- -19: channel mask can not be null
- -20: channel mask selection error
- -21: number of sequence selection error
- -22: sequence interrupt selection error (must be 0)
- -23: the ulTriggerMode parameter is wrong (must be 0)
- -24: the ulHardwareTriggerEdge parameter is wrong
- -25: the ulHardwareTriggerCount parameter is wrong
- -26: the ulByTriggerNbrOfSeqToAcquire parameter is wrong (must be 0)
- -27: the ulDataFormat parameter is wrong
- -28: the ulTriggerMask parameter is wrong
- -29: the ulICPMask is wrong (0->0xFF)
- -30: the ulCouplingSelectionMask is wrong (0->0xFF)
- -31: the ulSeDiffSelectionMask is wrong (0->0xFF)
- -40: PLD is not working
- -41: error, system is in calibration
- -42: driver status is wrong
- -100 internal system error occurs see value of syserrno

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	91	0x5B00	0x005B
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1200	0xB004	0x04B0
word count	2	16-bit integer	42	0x2A00	0x002A
byte count	1	8-bit integer	84	0x54	0x54
ulChannelMask	4	32-bit integer	See the description above	0x????????	0x????????
ulNbrOfSequence	4	32-bit integer	See the description above	0x????????	0x????????

ulNbrMaxSequenceToTransfer	4	32-bit integer	See the description above	0x????????	0x????????
dFrequencySelection	4	32-bit floating point	See the description above	0x????????	0x????????
pulGainArray	32	32-bit integer array	See the description above	0x????????[8]	0x????????[8]
ulICPMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMask	4	32-bit integer	See the description above	0x????????	0x????????
ulTriggerMode	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerEdge	4	32-bit integer	See the description above	0x????????	0x????????
ulHardwareTriggerCount	4	32-bit integer	See the description above	0x????????	0x????????
ulByTriggerNbrOfSeqToAcquire	4	32-bit integer	See the description above	0x????????	0x????????
ulDataFormat	4	32-bit integer	See the description above	0x????????	0x????????
ulCouplingSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????
ulSeDiffSelectionMask	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
	2		0	0x0000	0x0000

protocol identifier		16-bit integer			
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1200	0xB004	0x04B0
word count	2	16-bit integer	42	0x2A00	0x002A

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputStopSequence

For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputStopSequenceEx.

Description

Stop the analog input sequence acquisition mode.
Must be called before any another call to MSXE360x__AnalogInputInitAndStartSequence.

Parameters:

◊ [Query frame layout] ***Dummy***: Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus):

· 0: means the remote function performed OK.
· -1: Means an system error occured.
· -2: PLD is not working.
· -3: Error, system is in calibration.
· -4: Driver status is wrong.
· -40: PLD is not working.
· -41: Error, system is in calibration.
· -42: Driver status is wrong.
· -100: Stop sequence kernel function error.
· -101: Release sequence kernel function error.

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	12	0x0C00	0x000C
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	4	0x0400	0x0004
word count	2	16-bit integer	2	0x0200	0x0002
byte count	2	16-bit integer	4	0x0400	0x0004
Dummy	4	32-bit integer	See the description	0x????????	0x????????

			above		

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	4	0x0400	0x0004
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputStopSequenceEx

Description

Stop the analog input sequence acquisition mode.
Must be called before any another call to MSXE360x__AnalogInputInitAndStartSequence.

Parameters:

◊ [Query frame layout] ***Dummy***: Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx):

· 0: means the remote function performed OK.
· -1: Means an system error occured.
· -2: PLD is not working.
· -3: Error, system is in calibration.
· -4: Driver status is wrong.
· -40: PLD is not working.
· -41: Error, system is in calibration.
· -42: Driver status is wrong.
· -100: Stop sequence kernel function error.
· -101: Release sequence kernel function error.

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	11	0x0B00	0x000B
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10

Reference number (=register)	2	16-bit integer	1250	0xE204	0x04E2
word count	2	16-bit integer	2	0x0200	0x0002
byte count	1	8-bit integer	4	0x04	0x04
Dummy	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1250	0xE204	0x04E2
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit	3	0x0300	0x0003

		integer			
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputReleaseSequence

For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputReleaseSequenceEx.

Description

Release the analog input sequence acquisition mode.
Sequence has to be stopped before released.
Must be called before any another call to MSXE360x__AnalogInputInitAndStartSequence.

Parameters:

◊ [Query frame layout] ***Dummy***: Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus):

· 0: means the remote function performed OK.
· -1: Means an system error occured.
· -2: PLD is not working.
· -3: Error, system is in calibration.
· -4: Driver status is wrong.
· -40: PLD is not working.
· -41: Error, system is in calibration.
· -42: Driver status is wrong.
· -100: Stop sequence kernel function error.
· -101: Release sequence kernel function error.

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian	big endian (Motorola)

				(Intel)	
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	12	0x0C00	0x000C
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	5	0x0500	0x0005
word count	2	16-bit integer	2	0x0200	0x0002
byte count	2	16-bit integer	4	0x0400	0x0004
Dummy	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	5	0x0500	0x0005

word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputReleaseSequenceEx

Description

Release the analog input sequence acquisition mode.
Sequence has to be stopped before released.
Must be called before any another call to MSXE360x__AnalogInputInitAndStartSequence.

Parameters:

◊ [Query frame layout] ***Dummy***: Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx):

· 0: means the remote function performed OK.
· -1: Means an system error occured.
· -2: PLD is not working.
· -3: Error, system is in calibration.
· -4: Driver status is wrong.
· -40: PLD is not working.
· -41: Error, system is in calibration.

· -42: Driver status is wrong.
· -100: Stop sequence kernel function error.
· -101: Release sequence kernel function error.

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	11	0x0B00	0x000B
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1300	0x1405	0x0514
word count	2	16-bit integer	2	0x0200	0x0002
byte count	1	8-bit integer	4	0x04	0x04
Dummy	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000

length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1300	0x1405	0x0514
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputStopAndReleaseSequence

For new application(s) or automate communication it is recommended to use the function MSXE360X__AnalogInputStopAndReleaseSequenceEx.

Description

Stop and release the analog input sequence acquisition mode.
Must be called before any another call to MSXE360x__AnalogInputInitAndStartSequence.

Parameters:

◊ [Query frame layout] ***Dummy***: Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus):

· 0: means the remote function performed OK.
· -1: Means an system error occured.
· -2: PLD is not working.
· -3: Error, system is in calibration.
· -4: Driver status is wrong.
· -40: PLD is not working.
· -41: Error, system is in calibration.
· -42: Driver status is wrong.
· -100: Stop sequence kernel function error.
· -101: Release sequence kernel function error.

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	12	0x0C00	0x000C
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	6	0x0600	0x0006
word count	2	16-bit integer	2	0x0200	0x0002
byte count	2	16-bit integer	4	0x0400	0x0004
Dummy	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	6	0x0600	0x0006
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

Function MSXE360X__AnalogInputStopAndReleaseSequenceEx

Description

Stop and release the analog input sequence acquisition mode.
Must be called before any another call to MSXE360x__AnalogInputInitAndStartSequence.

Parameters:

◊ [Query frame layout] ***Dummy***: Reserved

Returns:

Possible return value on the remote system (read them with GetLastCommandStatusEx):

· 0: means the remote function performed OK.
· -1: Means an system error occured.
· -2: PLD is not working.
· -3: Error, system is in calibration.
· -4: Driver status is wrong.
· -40: PLD is not working.
· -41: Error, system is in calibration.
· -42: Driver status is wrong.
· -100: Stop sequence kernel function error.
· -101: Release sequence kernel function error.

syserrno : system-error code (the value of the libc "errno" code) EPERM means a sequence acquisition was not started

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	11	0x0B00	0x000B
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10

Reference number (=register)	2	16-bit integer	1350	0x4605	0x0546
word count	2	16-bit integer	2	0x0200	0x0002
byte count	1	8-bit integer	4	0x04	0x04
Dummy	4	32-bit integer	See the description above	0x????????	0x????????

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	6	0x0600	0x0006
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x10	0x10	0x10
Reference number (=register)	2	16-bit integer	1350	0x4605	0x0546
word count	2	16-bit integer	2	0x0200	0x0002

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Intel)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit	3	0x0300	0x0003

		integer			
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x90	0x90	0x90
Exception code	1	8-bit integer	See corresponding chapter	0x??	0x??

FC23 (read/write registers) Functions

Top

Functions in this group are used to read/write values on the module.
This functions permits to call a write (FC16) and then a read(FC3) function in one command.

Query frame layout

Field	Size (Bytes)	Type	Value	little endian (Motorola)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	Depends to the FC16 function called	?	?
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x17	0x17	0x17
Reference number for read (=register)	2	16-bit integer	FC3 reference	?	?
Word count for read	2	16-bit integer	See the corresponding FC3 function	?	?
Reference number for write (=register)	2	16-bit integer	FC16 reference	?	?
Word count for write	2	16-bit integer	See the corresponding FC16 function	?	?
Byte count	1	8-bit integer	(= 2xWord count for write)	?	?
Register values	?	?	See the corresponding FC16 function	?	?

Response frame layout

Field	Size (Bytes)	Type	Value	little endian (Motorola)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	Depends to the FC3 function called	?	?
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x17	0x17	0x17
Byte count	1	8-bit integer	(= 2x word count for read)	?	?
Register values	?	?	See the corresponding FC3 function	?	?

Exception frame layout

Field	Size (Bytes)	Type	Value	little endian (Motorola)	big endian (Motorola)
transaction identifier	2	16-bit integer	User defined - copied by server - usually 0	0x0000	0x0000
protocol identifier	2	16-bit integer	0	0x0000	0x0000
length	2	16-bit integer	3	0x0300	0x0003
unit identifier	1	8-bit integer	0 or 1	0x00 or 0x01	0x00 or 0x01
MODBUS Function code	1	8-bit integer	0x97	0x97	0x97
Exception code	1	8-bit integer	See corresponding chapter	??	??

Exception code description

Top

Name	Value	Description
MODBUS_ILLEGAL_FUNCTION	0x1	function code is not allowable action for t slave
MODBUS_ILLEGAL_DATA_ADDRESS	0x2	data address received query is not allowabl
MODBUS_ILLEGAL_DATA_VALUE	0x3	incorrect value int th query data field or th length is incorrect
MODBUS_ILLEGAL_DATA_RESPONSE_LENGTH	0x4	the request as framed would generate a response whose size exceeds the available MODBUS datasize.
MODBUS_ACKNOWLEDGE	0x5	specialized use in conjunction with programming comma
MODBUS_DSLAVE_DEVICE_BUSY	0x6	specialized use in conjunction with programming comma
MODBUS_NEGATIVE_ACKNOWLEDGE	0x07	specialized use in conjunction with programming comma
MODBUS_MEMORY_PARITY_ERROR	0x08	the extended file area failed to pass a consistency check
MODBUS_REMOTE_EXECUTION_ERROR	0x09	the remote function performed incorrectly (use function GetLastCommandSta to know why)
MODBUS_GATEWAY_PATH_UNAVAILABLE	0x0A	used with modbus pl gateway
MODBUS_GATEWAY_TARGET_DEVICE_FAILED_TO_RESPOND	0x0B	used with modbus pl gateway

Siemens Step 7 compatibility information (AWL/SDF code)

Top

Due to limitations of the S7 platform, some names of function and parameter have been shortened in the AWL and S7 code. This table summarizes the changes against the standard version as described above.

Function/Parameter	Renamed as
MXCommon__GetModuleType	GetModuleType
MXCommon__GetTime	GetTime
MXCommon__TestCustomerID	TestCustomerID
MSXE360X__AnalogInputGetSequenceStatus	360x_AIGetSeqStat
MSXE360X__AnalogInputGetSequenceConfiguration	360x_AIGetSeqConf
ulNbrMaxSequenceToTransfer	ulNbrMaxSeqToTransf
ulByTriggerNbrOfSeqToAcquire	ulByTrigNbrOfSeqToAcq
MXCommon__SetHardwareTriggerFilterTime	SetHwTrigFiltTime
MXCommon__InitAndStartSynchroTimer	InitStartSyncTimer
MXCommon__StopAndReleaseSynchroTimer	StopRelSyncTimer
MXCommon__Reboot	Reboot
MXCommon__SetCustomerKey	SetCustomerKey
MXCommon__SetFilterChannels	SetFilterChannels
MSXE360X__AnalogInputInitSequence	360x_AIInitSeq
ulNbrMaxSequenceToTransfer	ulNbrMaxSeqToTransf
ulByTriggerNbrOfSeqToAcquire	ulByTrigNbrOfSeqToAcq
MSXE360X__AnalogInputStartSequence	360x_AIStartSeq
MSXE360X__AnalogInputInitAndStartSequence	360x_AIInitStartSeq
ulNbrMaxSequenceToTransfer	ulNbrMaxSeqToTransf
ulByTriggerNbrOfSeqToAcquire	ulByTrigNbrOfSeqToAcq
MSXE360X__AnalogInputStopSequence	360x_AIStopSeq
MSXE360X__AnalogInputReleaseSequence	360x_AIRelSeq
MSXE360X__AnalogInputStopAndReleaseSequence	360x_AIStopRelSeq