ABB Drives

Installation and Start-up Guide

Modbus Adapter Module
NMBA-01

Modbus Adapter Module
NMBA-01

Installation and Start-up Guide

3AFY 58919772 R0225

EFFECTIVE: 1.2.1999
SUPERSEDES: 1997-01-10

Safety Instructions

Overview

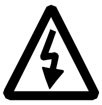
This chapter states the safety instructions that must be followed when installing and operating the NMBA-01 Modbus Adapter Module. If neglected, physical injury and death may follow, or damage may occur to the frequency converter, the motor and driven equipment. The material in this chapter must be studied before attempting any work on, or with, the unit.

Warnings and Notes

This manual distinguishes two sorts of safety instructions. Warnings are used to inform of conditions which can, if proper steps are not taken, lead to a serious fault condition, physical injury and death. Notes are used when the reader is required to pay special attention or when there is additional information available on the subject. Notes are less crucial than Warnings, but should not be disregarded.

Warnings

Readers are informed of situations that can result in serious physical injury and/or serious damage to equipment with the following symbols:



Dangerous Voltage Warning: warns of situations in which a high voltage can cause physical injury and/or damage equipment. The text next to this symbol describes ways to avoid the danger.

General Warning: warns of situations which can cause physical injury and/or damage equipment by means other than electrical. The text next to this symbol describes ways to avoid the danger.

Electrostatic Discharge Warning: warns of situations in which an electrostatic discharge can damage equipment. The text next to this symbol describes ways to avoid the danger.

Notes

Readers are notified of the need for special attention or additional information available on the subject with the following symbols:

CAUTION! **Caution** aims to draw special attention to a particular issue.

Note: **Note** gives additional information or points out more information available on the subject.

General Safety Instructions

WARNING! All electrical installation and maintenance work on the drive should be carried out by qualified electricians.

The drive and adjoining equipment must be properly earthed.

Do not attempt any work on a powered drive. After switching off the mains, always allow the intermediate circuit capacitors 5 minutes to discharge before working on the frequency converter, the motor or the motor cable. It is good practice to check (with a voltage indicating instrument) that the frequency converter is in fact discharged before beginning work.

The motor cable terminals of the drive are at a dangerously high voltage when mains power is applied, regardless of motor operation.

There can be dangerous voltages inside the drive from external control circuits when the drive mains power is shut off. Exercise appropriate care when working with the unit. Neglecting these instructions can cause physical injury and death.

WARNING! There are several automatic reset functions in the drive. If selected, they reset the unit and resume operation after a fault. These functions should not be selected if other equipment is not compatible with this kind of operation, or dangerous situations can be caused by such action.

More Warnings and Notes are printed at appropriate instances along the text.

Table of Contents

Chapter 7 – Fault Tracing

Appendix A – Technical Data

Appendix B – Modbus Protocol

Appendix C – Ambient Conditions

Chapter 1 – Introduction to This Guide

Overview

This chapter contains a description of the *Installation and Start-up Guide* for the NMBA-01 Modbus Adapter Module.

Intended Audience

The Guide is intended for the people who are responsible for installing, commissioning and using a Modbus Adapter Module with an ABB drive. The reader is expected to have a basic knowledge of electrical fundamentals, electrical wiring practices, the drive, the use of the drive control panel, and the Modbus protocol family.

What This Guide Contains

The installation and start-up of the NMBA-01 Modbus Adapter Module are introduced in this Guide.

It is assumed that the drive is installed and ready to operate before starting the installation of the adapter module. For more information on the installation and start-up procedures of the drive, please refer to its user documentation.

Safety Instructions are featured in the first few pages of this Guide. Safety Instructions describe the formats for various warnings and notations used within this Guide. This chapter also states the safety instructions which apply to the installation and operation of the NMBA-01 Module.

Chapter 1 – Introduction to This Guide contains a short description of the Guide.

Chapter 2 – Overview contains a short description of the NMBA-01 Modbus Adapter Module, a delivery checklist, and information on the manufacturer's warranty.

Chapter 3 – Mechanical Installation contains placing and mounting instructions for the module.

Chapter 4 – Electrical Installation contains wiring, bus termination and earthing instructions.

Chapter 5 – Programming explains how to program the master station and the drive before the communication through the adapter module can be started.

Chapter 6 – Communication contains a description of how data is transmitted through the NMBA-01 Module.

Chapter 7 – Fault Tracing explains how to trace faults with the Status LEDs on the NMBA-01 Module.

Appendix A – Technical Data contains the technical information on the Modbus module.

Appendix B – Modbus Protocol describes the Modbus protocol.

Appendix C – Ambient Conditions contains a specification of the ambient conditions allowed during the transportation, storing and use of the Modbus module.

Terms Used in This Guide

Parameter A parameter is an operating instruction for the drive. Parameters can be read and programmed with the drive control panel, or through the NMBA-01 Module.

Communication Module Communication Module is a parameter name/parameter selection name for a device (e.g. a fieldbus adapter) through which the drive is connected to an external serial communication network (e.g. a fieldbus). The communication with the communication module is activated with a drive parameter.

NMBA-01 Modbus Adapter Module The NMBA-01 Adapter Module is one of the optional fieldbus adapter modules available for ABB drives. The NMBA-01 is a device through which an ABB drive is connected to a Modbus serial communication bus.

Data Sets and Data Words Data sets are clusters of data sent through the DDCS link between the NMBA-01 Adapter Module and the drive. Each data set consists of three 16-bit words, i.e. data words. The Control Word (sometimes called the Command Word) and the Status Word, References and Actual Values (see Chapter 6) are types of data words; the contents of some data words are user-definable.

Broadcast Write Modbus network allows the Modbus master to perform a write to every slave station at the same time. This write is called Broadcast. This service does not give verification back to the master that the value has been received by each one of the slaves properly.

4XXXX Register Area Modicon PLCs have a signed integer data table area, which is used for Analogue Output modules and for storing temporary or set-point values. These registers are in the address area starting from 40001. The last register address available on PLCs depends on the available memory, but is less than 49999. The drive simulates this area by providing a read and write access to its parameters through this register address area.

Chapter 2 – Overview

Overview

This chapter describes the NMBA-01 Modbus Adapter module, and gives the Warranty and Liability information of the Manufacturer. (Information about the Modbus protocol is available in Appendix B.)

Introduction to Modbus

Modbus is a serial, asynchronous protocol. The Modbus protocol does not specify the physical interface. The typical physical interfaces are RS-232 and RS-485. The NMBA-01 uses the RS-485 interface.

Modbus is designed for integration with Modicon PLCs or other automation devices, and the services closely correspond to the PLC architecture. The NMBA-01 ‘looks like’ a Modicon PLC on the network.

The NMBA-01 Modbus Adapter Module

The NMBA-01 Modbus Adapter Module is an optional device for ABB drives which enables the connection of the drive to a Modbus system. The drive is considered as a slave in the Modbus network. Through the NMBA-01 Modbus Adapter Module it is possible to:

- Give control commands to the drive (Start, Stop, Run enable, etc.)
- Feed a motor speed or torque reference to the drive
- Give a process actual value or a process reference to the PID controller of the drive
- Read status information and actual values from the drive
- Change drive parameter values
- Reset a drive fault
- Use Multiple Drive Control.

The Modbus commands and services supported by the NMBA-01 Modbus Adapter Module are discussed in Chapter 6. Please refer to the user documentation of the drive as to which commands are supported by the drive.

The adapter module is mounted onto a standard mounting rail inside or outside the drive unit, depending on drive type and configuration. See the user’s manual of the drive for module placement options.

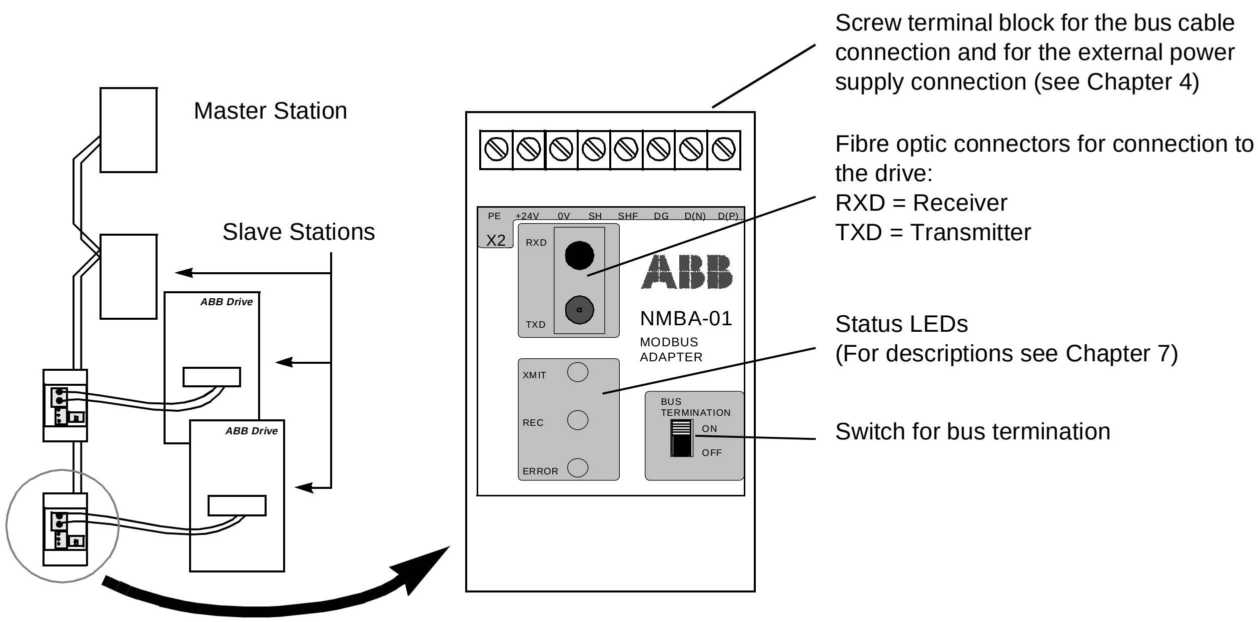


Figure 2-1   The construction of the Modbus link and the NMBA-01 Adapter Module.

Compatibility

The NMBA-01 is compatible with:

- ACS 300
- ACS 400
- ACS 600 SingleDrive
- ACS 600 MultiDrive
- ACS 600 MotionControl (ACP 600)
- ACS 600 CraneDrive (ACC 600)
- ACS 600 Pump and Fan Drive (ACF 600)
- DCS 500
- All master stations that support the Modicon-defined Modbus serial communication protocol.

Delivery Check

The option package for the NMBA-01 Modbus Adapter Module contains:

- Modbus Adapter Module, Type NMBA-01
- Two pairs (four pieces) of fibre optic cables
- Mounting rail
- This manual, the *NMBA-01 Installation and Start-up Guide.*

Warranty and Liability Information

The warranty for your ABB drive and options covers manufacturing defects. The manufacturer carries no responsibility for damage due to transport or unpacking.

In no event and under no circumstances shall the manufacturer be liable for damages and failures due to misuse, abuse, improper installation, or abnormal conditions of temperature, dust, or corrosives, or failures due to operation above rated capacities. Nor shall the manufacturer ever be liable for consequential and incidental damages.

The period of manufacturer’s warranty is 12 months, and not more than 18 months, from the date of delivery.

Extended warranty may be available with certified start-up. Contact your local distributor for details.

Your local ABB Drives company or distributor may have a different warranty period, which is specified in their sales terms, conditions, and warranty terms.

If you have any questions concerning your ABB drive, contact your local distributor or ABB Drives office.

The technical data and specifications are valid at the time of printing. ABB reserves the right to subsequent alterations.

Chapter 3 – Mechanical Installation

Overview

This chapter contains module mounting instructions. Depending on the drive, the module can be installed either inside or outside the drive housing or cabinet. See the user's manual of the drive for module placement options.

Mounting Outside the Drive

Choose the location for the module. Note the following:

- The cabling instructions must be followed (see Chapter 4). Also, the length of the fibre optic cables included in the option package restrict the distance between the module and the drive.
- Observe the free space requirements for the module (min. 10 mm from adjoining equipment or wall) and the drive (see the drive documentation).
- The ambient conditions should be taken into account (see Appendix C). The degree of protection of the module is IP 20.
- Module earth is connected to the mounting rail by means of an earthing clip (see *Figure 3-1*). The mounting rail onto which the option module is to be mounted must be earthed to a noiseless earth. If the rail is not mounted on a properly earthed base, a separate earthing conductor must be used. The conductor must be as short as possible and the cross-sectional area must be 6 mm^2 at least. ***Note:*** No solid copper conductor may be used (stranded wire allowed only).

Mounting instructions:

1. Switch off all dangerous voltages in the enclosure that the module is to be mounted in.
2. Fasten the rail and ensure the proper earthing as described above.
3. Push the module onto the rail. The module can be released by pulling the locking spring with a screwdriver (see *Figure 3-1*).

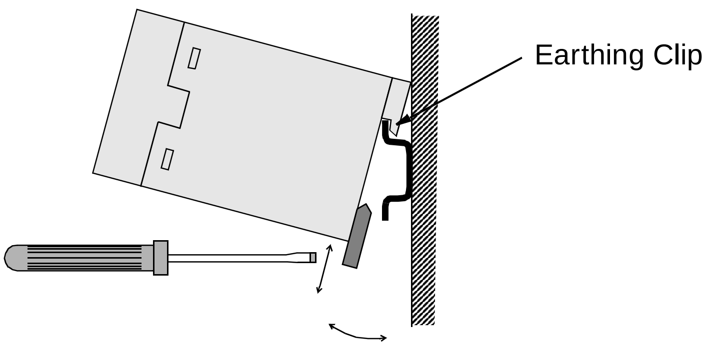


Figure 3-1 Mounting and removing the module.

Mounting Inside the Drive

The work inside the drive should be carried out by a qualified electrician only.

WARNING! Pay attention to the slowly discharging voltage of the capacitor bank and the voltages that are connected from external control circuits to the inputs and outputs of the drive.

WARNING! Do not touch the printed circuit boards. The integrated circuits are extremely sensitive to electrostatic discharge.

Mounting instructions:

1. Stop the drive.
2. Switch off the power supply of the drive and all dangerous voltages connected to the inputs and outputs.
3. Wait for five minutes to ensure that the capacitors in the intermediate circuit have discharged.
4. Remove the front cover of the drive.
5. Ensure that the mains cable, motor cable and capacitor bank (UDC+ and UDC-) are not powered.
6. Locate the position for the module (see the user's manual of the drive). Fasten the mounting rail to its place if not already installed. Observe the free space requirements for the module (min. 10 mm from adjoining equipment/wall).
7. Push the module onto the rail. The module can be released by pulling the locking spring with a screwdriver (see *Figure 3-1*).

Chapter 4 – Electrical Installation

Overview

This chapter contains:

- Cabling instructions
- Instructions for bus termination
- Connection and earthing instructions for the NMBA-01 Module and earthing instructions for the bus cable.

WARNING! Before installation, switch off the drive power supply. Wait for five minutes to ensure that the capacitor bank of the drive is discharged. Switch off all dangerous voltages connected from external control circuits to the inputs and outputs of the drive.

Cabling

Arrange the bus cables as far away from the motor cables as possible. Avoid parallel runs. Use bushings at cable entries.

Handle the fibre optic cables with care. When unplugging optic cables, always grab the connector, not the cable itself. Do not touch the ends of the fibres with bare hands as the fibre is extremely sensitive to dirt.

The maximum long term tensile load for the fibre optic cable is 1 N. The minimum short term bend radius is 25 mm.

Bus Termination

The built-in terminating resistors must be switched on if the NMBA-01 module is installed at the end of the bus. Otherwise the resistors must be switched off. Terminating resistors prevent signal reflections from the bus cable ends.

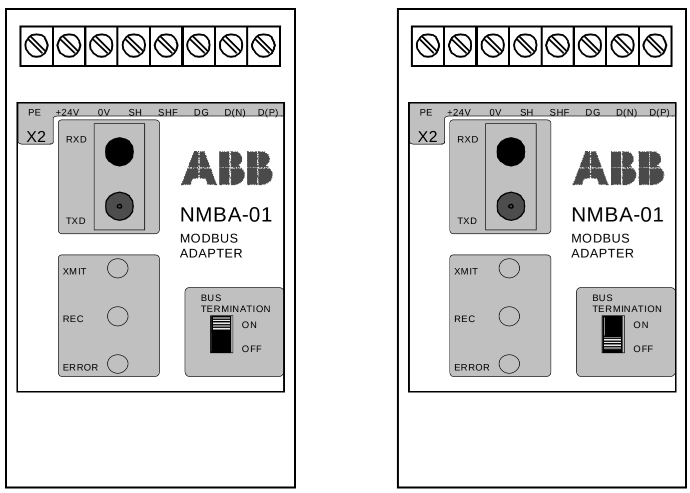


Figure 4-1 Terminating resistors on (left) and off (right).

NMBA-01 Connections

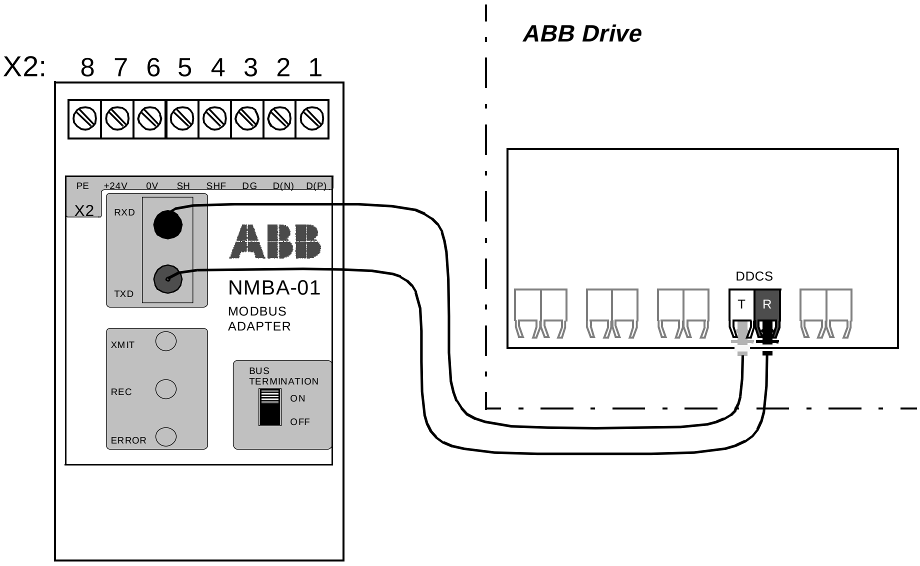


Figure 4-2 Fibre optic link connecting the NMBA-01 adapter to the drive.

The NMBA-01 module is connected to the drive using a fibre optic cable link. Consult the drive documentation as to the corresponding terminals inside the drive.

The bus cable and the external power supply are connected to terminal block X2 on the NMBA-01.

Table 4-1 Description of terminal block X2.

X2		Description
1	D(P)	D(P) = B = Data Positive (Conductor 1 in twisted pair) D(N) = A = Data Negative (Conductor 2 in twisted pair) DG = Data Ground
2	D(N)	D(P) = B = Data Positive (Conductor 1 in twisted pair) D(N) = A = Data Negative (Conductor 2 in twisted pair) DG = Data Ground
3	DG	D(P) = B = Data Positive (Conductor 1 in twisted pair) D(N) = A = Data Negative (Conductor 2 in twisted pair) DG = Data Ground
4	SHF	Cable screen AC earthing (via an RC filter)
5	SH	Cable screen earthing (directly earthed)
6	0V	Power supply for the module (24 V d.c. ± 10 %); screened cable.
7	+24 V	Power supply for the module (24 V d.c. ± 10 %); screened cable.
8	PE	Earth

Earthing

The NMBA-01 module earth is connected to the rail onto which the module is mounted. If the rail is fastened to an earthed metallic assembly plate, the module is automatically earthed, and no external earthing wire is needed. If the rail is fastened to a base that is not earthed, the rail must be connected to the nearest earthing terminal. However, the earthing wire should not be connected to the same terminal as the power cable screens. (See page 3-1.)

In the NMBA-01 module there are several built-in earthing terminals (see *Figure 4-3* and *Figure 4-4*):

- The **PE** terminal is internally connected to the NMBA-01 module earth. Normally, no external wires need to be connected to this terminal.
- The **SH** terminal is internally connected to the NMBA-01 module earth. The SH terminal is normally used for earthing the Modbus cable shield if there is no other station at which the cable shield is directly earthed.
- The **SHF** terminal is internally connected to the NMBA-01 module earth via an RC filter. The SHF terminal is typically used for earthing the Modbus cable shield.
- The **DG** terminal is isolated from the NMBA-01 module earth. This terminal is used for connecting the third conductor of the bus cable. The third conductor – Data Ground – offers a common reference or comparison potential to all modules on the bus.

Note: The use of Data Ground is recommended as it improves noise immunity. See *Figure 4-3* and *Figure 4-4*.

Earthing the Modbus Cable Shields

The Modbus cable shield may be directly earthed at one station only. At other stations the cable shield should be earthed via an RC filter.

There are two wiring examples in the figures below. The three-wire connection is the preferred one because of its the better noise immunity.

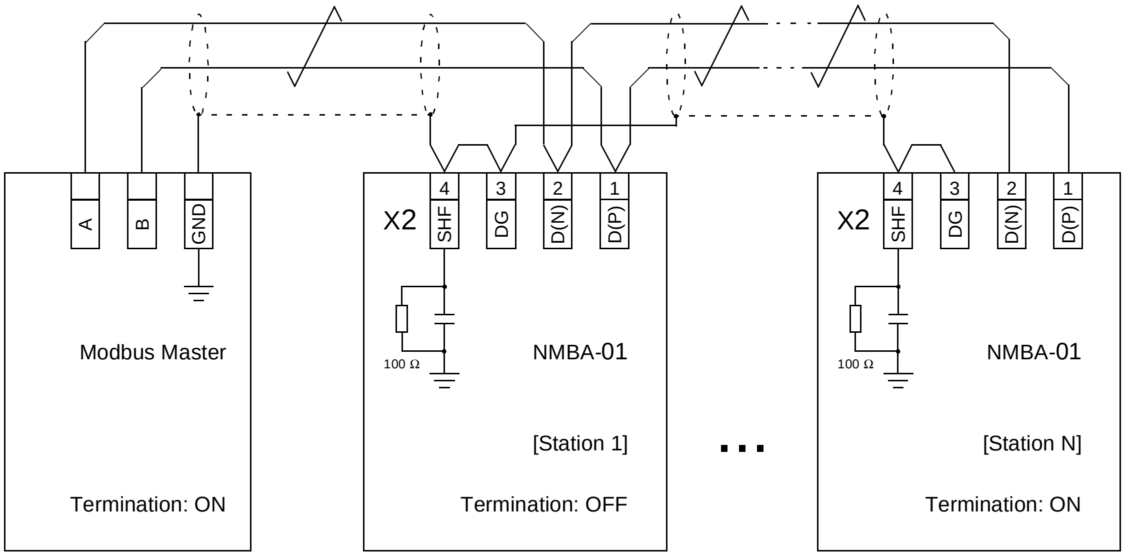


Figure 4-3 Two-wire Connection.

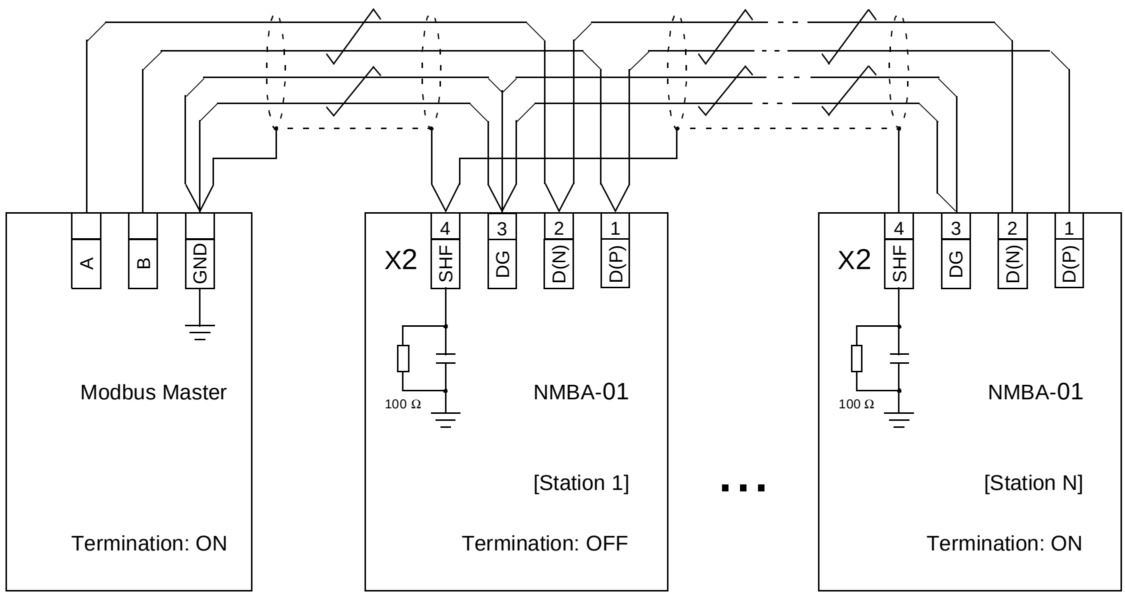


Figure 4-4 Three-wire Connection. (Preferred Practice.)

Chapter 5 – Programming

Overview

This chapter gives information on configuring the Modbus master station and the drive for communication through the NMBA-01 Modbus Adapter Module.

Configuring the System

After the NMBA-01 Modbus Adapter Module has been mechanically and electrically installed according to the instructions in Chapters 3 and 4, the master station and the drive must be prepared for communication with the module.

Please refer to the master station documentation for information on configuring the system for communication with the NMBA-01.

Modbus Connection Configuration

The detailed procedure of activating the module for communication with the drive is dependent on drive type. (Normally, a parameter must be adjusted to activate the communication. See the drive documentation.)

As communication between the drive and the NMBA-01 is established, several configuration parameters are copied to the drive. These parameters – shown in Table 5-1 – must be checked first and adjusted if necessary. The alternative selections for these parameters are discussed in more detail below the table. (Note that the new settings take effect only when the module is powered up for the next time.)

Note: The grouping, numbering, and adjustment procedure of parameters vary from drive to drive. See the drive documentation for information.

Table 5-1 The NMBA-01 configuration parameters.

Fieldbus Par. No.	Parameter Name	Alternative Settings	Default Setting
1	MODULE TYPE		NMBA-01 Vx.x
2	MODBUS MODE	**(0)** RTU wdg:flt; **(1)** RTU wdg:rst	**(0)** RTU wdg:flt
3	STATION NUMBER	1 – 247	1
4	BAUD RATE	**(0)** 1200; **(1)** 2400; **(2)** 4800; **(3)** 9600; **(4)** 19200	**(3)** 9600
5	PARITY	**(0)** EVEN; **(1)** ODD; **(2)** NONE 2 S.BIT; **(3)** NONE 1 S.BIT	**(2)** NONE 2 S.BIT
6	GOOD MESSAGES	0 – 32767	0
7	BAD MESSAGES	0 – 32767	0
8	DDCS CHANNEL	**(0)** CH0; **(1)** CH3	**(0)** CH0

MODULE TYPE
This parameter shows the module type and SW version number as detected by the drive. The value cannot be adjusted by the user. (If this parameter is undefined, the communication between the drive and the module has not been established.)

MODBUS MODE
Shows the logical protocol on the hardware communication option module (the NMBA-01 supports only the RTU mode) and the operating mode of the watch-dog reset. There are two selections:

RTU wdg:flt
Remote Terminal Unit (RTU) mode. On a watch-dog error, the module will indicate a watch-dog error. The module has to be reset (power off) manually.

RTU wdg:rst
Remote Terminal Unit (RTU) mode. On a watch-dog error, the module will reset itself.

STATION NUMBER
Each device on the Modbus link must have a unique station number. This parameter is used to define a station number for the drive it is connected to.

BAUD RATE
Defines the communication speed in Baud. There are five selections: **1200, 2400, 4800, 9600** and **19200** Baud.

PARITY
Defines the parity to be used with the Modbus communication. This parameter also defines the number of stop bits in use. Typically with Modbus communication the number of stop bits is 2 with no parity bit, and 1 with even or odd parity. There are four selections: **EVEN, ODD, NONE 1 S.BIT** and **NONE 2 S.BIT.**

GOOD MESSAGES This diagnostics counter increases by one every time a valid Modbus message has been received by NMBA-01 Module. This counter will roll over from 32767 back to 0. During normal operation, this counter is increasing constantly.

BAD MESSAGES This diagnostics counter increases by one every time the NMBA-01 Module finds any kind of communication error. This counter will roll over from 32767 back to 0. During normal operation, this counter hardly ever increases.

DDCS CHANNEL Selects the DDCS CHANNEL of the drive to which the fibre optics are connected. This parameter is only used with ACS 600.

Note: The changes to the parameters do not take effect immediately. To change the module settings, the power must be disconnected from the NMBA-01.

Control Locations

ABB drives can receive control information from multiple sources including digital inputs, analogue inputs, the drive control panel and a communication module (e.g. NMBA-01). ABB drives allow the user to separately determine the source for each type of control information (Start, Stop, Direction, Reference, Fault Reset, etc.). In order to give the fieldbus master station the most complete control over the drive, the communication module must be selected as source for this information. See the user documentation of the drive for information on the selection parameters.

Chapter 6 – Communication

Overview

This chapter describes the Modbus communication with the drive.

Register Read and Write

The drive parameter and data set information is mapped into a 4xxxx register area. This holding register area can be read from an external device, and an external device can modify the register values by writing to them.

There are no setup parameters for mapping the data to the 4xxxx register. The mapping is pre-defined and corresponds directly to the drive parameter grouping which is being used by the local drive panel.

All parameters are available for both reading and writing. The parameter writes are verified for correct value, and for valid register addresses. Some parameters never allow writes (including actual values), some parameters allow write only when the drive is stopped (including setup variables), and some can be modified at any time (including actual reference values).

Register Mapping

The drive parameters are mapped to the 4xxxx area so that:

- 40001 – 40096 are reserved for data sets.
- 40101 – 49999 are reserved for parameters.

In this mapping, the thousands and hundreds correspond to the group number, while the tens and ones correspond to the parameter number within a group. Register addresses 4GGPP are shown in *Table 6-1 Parameter Mapping*. In this table GG is the group number, and PP is the parameter number within the group.

Table 6-1 Parameter Mapping

	4GGPP	GG	PP
Data sets	4**0001** – 4**0096**	**00** Data sets	**01** Data word 1.1 **02** Data word 1.2 **03** Data word 1.3 **04** Data word 2.1 **05** Data word 2.2 **06** Data word 2.3 **07** Data word 3.1 ... **94** Data word 32.1 **95** Data word 32.2 **96** Data word 32.3
Parameters	4**0101** – 4**0199**	**01** Group 01	**01** Parameter 01 **02** Parameter 02 ... **99** Parameter 99
	4**0201** – 4**0299**	**02** Group 02	**01** Parameter 01 **02** Parameter 02 ... **99** Parameter 99
	…	...	...
	4**9901** – 4**9999**	**99** Group 99	**01** Parameter 01 … **99** Parameter 99

The register addresses which are not allocated to any drive parameter or data set are invalid. No reads or writes are allowed for these addresses. If there is an attempt to read or write outside the parameter addresses, the Modbus interface will return an exception code to the controller.

Refer to the drive manuals for its data sets, group and parameter numbers supported.

Exception Codes

The NMBA-01 supports the Modbus exception codes shown in *Table 6-2*.

Table 6-2 Exception Codes.

Code	Name	Reason
01	ILLEGAL FUNCTION	Unsupported Command.
02	ILLEGAL DATA ADDRESS	Address does not exist or is read/write protected.
03	ILLEGAL DATA VALUE	Value is outside min-max limits. Parameter is read-only.
04	SLAVE DEVICE FAILURE	Reading a multiple parameter register, which includes one or more read protected parameters.
06	SLAVE DEVICE BUSY	DDCS channel timeout.

Data Update

The NMBA-01 module has been designed for time-optimised, reliable data transfer between the Modbus network and the drive.

Table 6-3 Function Codes.

Code	Name	Meaning
03	Read Holding Registers	Reads the binary contents of holding registers (4X references) in the slave.
06	Preset Single Register	Presets a value into a single holding register (4X reference). When broadcast, the function presets the same register reference in all attached slaves.
16 (10 Hex)	Preset Multiple Regs	Presets values into a sequence of holding registers (4X references). When broadcast, the function presets the same register references in all attached slaves.

Data set registers are updated in a fast cyclic interval. Updating of parameter registers happens in a slower interval. In case of a single parameter register read or write request, the parameter value is updated directly from the drive’s internal memory. In case of a multiple parameter register read or write request, the parameter values are updated from the internal memory of the NMBA-01.

Multiple Drive Control

It is possible to connect multiple (theoretical maximum 247) drives to one NMBA-01. Multiple Drive Control has the same control principle as the point-to-point connection. The NMBA-01 behaves like a multiple-address node on the Modbus network.

The Modbus network can contain several Multiple Drive Control segments. However, the total number of drives cannot exceed 247.

Note: Only one type of drive (e.g. ACS 300 or ACS 600) can be connected to one Multiple Drive Control segment.

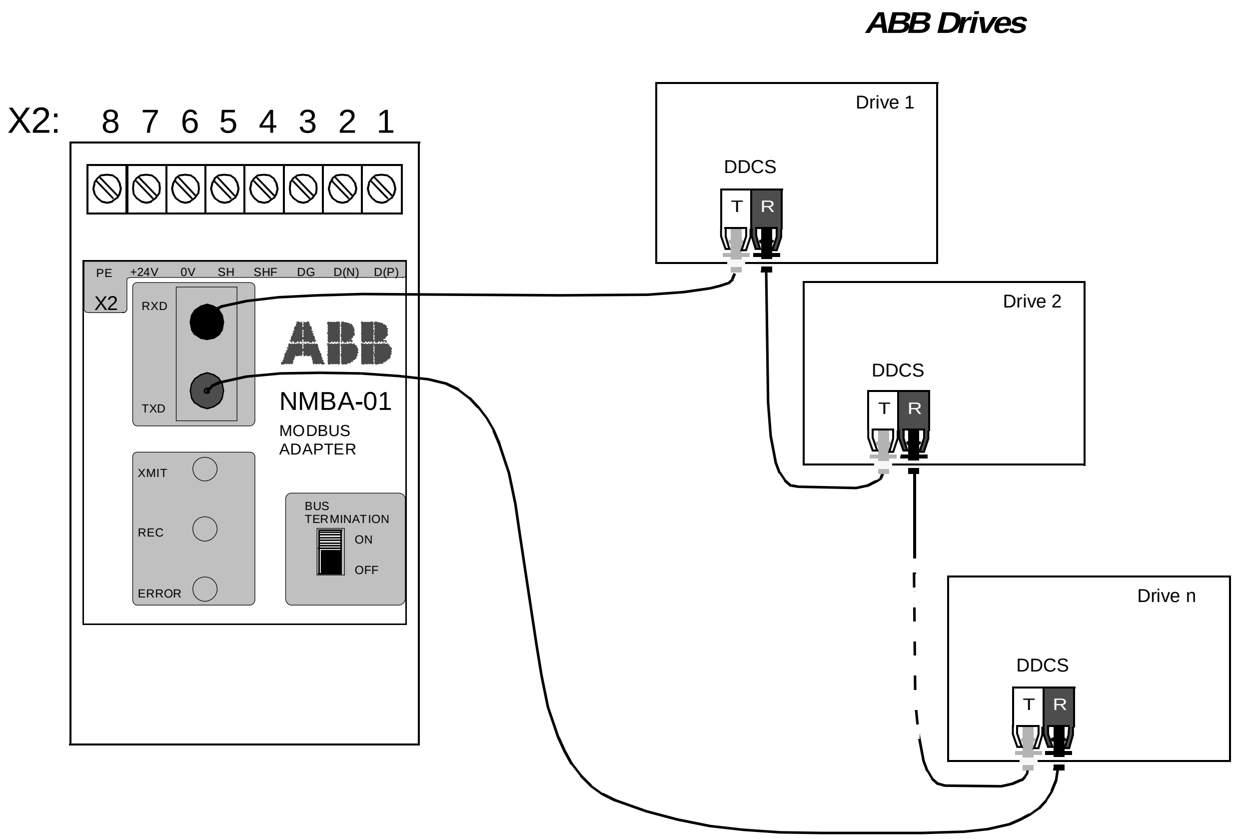


Figure 6-1 Multiple Drive Control Connection.

The Multiple Drive Control connection does not significantly add to the DDCS communication delay. The update interval of the data in the parameter registers is longer than in point-to-point connection, due to the fact that each parameter value is updated from the drive's internal memory. The delay is most noticeable when multiple parameter register read or write is requested.

Multiple Drive Control Set-up

Multiple Drive Control needs only parametrisation of one single drive, the first drive in the ring. To configure the Modbus adapter, the fieldbus parameters of the first drive are set by means of its control panel. (See Chapter 5 for more specific information.) The drive in question is at that time connected point-to-point with the NMBA-01(see *Figure 4-2*). After the Modbus adapter has been configured, the drives can be reconnected for Multiple Drive Control.

During initialisation of the NMBA-01, the number of drives connected to the ring is checked. The slaves are autonumbered starting from the first drive in the ring. For example if fieldbus parameter *03 STATION NUMBER* of the first drive is 5, then the station number of the second drive is 6 etc. In Multiple Drive Control, the NMBA-01 sends only two fieldbus parameters to all other drives in the ring. These parameters are *01 MODULE TYPE* and *03 STATION NUMBER* and they are read only.

Parameter Handling

The NMBA-01 can read and write up to 99 parameters in one frame if these are in the same parameter group. The user can read and write (Modbus function codes 0x03, 0x06 and 0x10 Hex) all parameters of each drive in Multiple Drive Control as if they were each connected to their own NMBA-01 module. However, the response time might be longer depending on the number of parameters to be handled.

Broadcast messages cannot be used for changing parameters.

Data Set Communication

Data sets 1, 3, 5,... (data from NMBA-01 to the drive) can be used in broadcast messages. However, data sets 2, 4, 6,... (data from the drive to the NMBA-01) cannot be used in broadcast messages.

Chapter 7 – Fault Tracing

Overview

This chapter gives step-by-step diagnostics information for finding out the root causes and corrections to the most common problems with the NMBA-01 module.

This section is divided into different sections, which lists first the symptoms, then possible causes, and remedies for them.

Installation Problems

Verify all the connections on the module. Check that

- the Modbus cable is connected correctly to terminal block X2.
- 24 V d.c. power is connected to the power connectors.
- the fibre cables between the drive and the NMBA-01 are correctly connected.
- the fibre link connector colours match the drive and NMBA-01 connector colours.

Drive Setup

Fieldbus parameter group is not visible on the control panel.

- Activate the fieldbus module according to the instructions in the drive manual.

Drive parameters can be read, but control commands (Start/Stop or Reference) do not work.

- Check that the drive has the fieldbus adapter selected as the source of these commands (see the drive manual).

PLC Programming

PLC program is beyond ABB support. Contact the manufacturer for assistance.

Status LEDs

The NMBA-01 has three status LEDs. These are from top to bottom:

- *XMIT* LED. This LED will flash each time the NMBA-01 transmits a response or an exception on the Modbus network.
- *REC* LED. This LED will flash each time the NMBA-01 receives a command from the Modbus network.
- *ERROR* LED for combined error and module status indication. This LED will flash for the following reasons:
  - If the received command had a parity error.
  - If the received command had a CRC error.
  - If the received command was not supported by the NMBA-01 module. In this case the *XMIT* LED will also flash.
  - If the NMBA-01 module has found an error. The error codes are described in Table 7-2 below.

Module Diagnostics

On module power-up, the NMBA-01 goes through a power-up self-test sequence. During this, the test state is indicated by the front three LEDs. The normal power-up procedure is as follows:

- All LEDs are turned on for the duration of the RAM/ROM test. If the test is passed, all LEDs will be turned off.
- The *ERROR* LED will flash rapidly for few seconds as the DDCS link between the NMBA-01 and the drive is initialised.
- *REC* and *XMIT* LEDs will flash according to the data transferred between the NMBA-01 and the drive.

Table 7-1   Modbus Error Codes during Initialization.

Flash Code	Status	Correction
XMIT, REC, and ERROR LEDs on	ROM checksum test failed	Hardware failure. Replace NMBA.
REC and ERROR LEDs on	RAM test failed	Hardware failure. Replace NMBA.
ERROR LED on	DDCS ASIC register access test failed	Hardware failure. Replace NMBA.

Table 7-2 Modbus Error Codes during Operation.

Flash Code	Status	Correction
Occasional flash on Error LED	Module on-line. Parity, CRC or unsupported command received.	Check the wiring and earthing of the module. Verify that only supported commands are sent to the module.
Error LED flashes continuously	NMBA is OK, no response from the drive on fibre link.	Power on the drive. Check the fibre link wiring.
All LEDs flash continuously at the same time.	Watchdog time out.	Hardware failure on NMBA module. Switch power off and wait a few seconds before switching the power on again; if the problem persists, check that fieldbus parameter *02 MODBUS MODE* is set to RTU wdg:rst.
Continuous reboot. Fast blinking of *ERROR* LED for approx. 2 sec. repeated every few sec.	Drive configuration write failed (DDCS link communication fault).	Check fibre optics, DDCS channel and SW version of the drive.

Appendix A – Technical Data

DDCS Link

Compatible Devices: All ABB Fieldbus Adapter modules,
ABB ACS 300, ACS 400, ACS/ACP/ACF 600, DCS 500 Drives

Size of the Link: 2 stations (3 to 248 in Multiple Drive Control)

Medium: Fibre optic cable

- Construction: Plastic core, diameter 1 mm, sheathed with plastic jacket
- Attenuation: 0.31 dB/m
- Maximum Length between Stations: 10 m
- Specifications:

Parameter	Minimum	Maximum	Unit
Storage Temperature	-55	+85	°C
Installation Temperature	-20	+70	°C
Short Term Tensile Force		50	N
Short Term Bend Radius	25		mm
Long Term Bend Radius	35		mm
Long Term Tensile Load		1	N
Flexing		1000	cycles

Topology: Point-to-point (Ring in Multiple Drive Control)

Serial Communication Type: Asynchronous, half Duplex

Transfer Rate: 4 Mbit/s

Protocol: Distributed Drives Communication System (DDCS)

Connectors: Blue – receiver; grey – transmitter

Fieldbus Link

Compatible Devices: Any Modbus device capable of Modbus communication as a master.

Size of the Link: 247 stations including repeaters (31 stations and 1 repeater per segment)

Medium: Shielded, twisted pair RS485 cable

- Termination: built in the NMBA-01 Module
- Modbus cable: Belden 9841 (typical)
- Maximum Bus Length: 1200 m

Topology: Multi-drop

Serial Communication Type: Asynchronous, half Duplex

Transfer Rate: 1200, 2400, 4800, 9600, 19200 bit/s

Protocol: Modbus

NMBA-01

Enclosure: Plastic, dimensions 45 × 75 × 105 mm; degree of protection IP 20

Mounting: Onto a standard mounting rail

Settings: Via drive interface (control panel)

Current Consumption: 65 mA at 24 V d.c.

Connectors:

- Light transmitter (grey) and receiver (blue) (Hewlett-Packard Versatile Link) for connection to the drive
- One Combicon MVSTBW 2,5/8-ST-5,08 (8-pole, cross-section 2.5 mm^2 max.) screw terminal block for the fieldbus and power supply:

X2		Description
1	D(P)	D(P) = B = Data Positive (Conductor 1 in twisted pair) D(N) = A = Data Negative (Conductor 2 in twisted pair) DG = Data Ground
2	D(N)	
3	DG	
4	SHF	Cable screen AC earthing (via an RC filter)
5	SH	Cable screen earthing (directly earthed)
6	0V	Power supply for the module (24 V d.c. ± 10 %); screened cable.
7	+24 V	
8	PE	Earth

General:

- All materials are UL/CSA approved
- Complies with EMC Standards EN 50081-2 and EN 50082-2

Appendix B – Modbus Protocol

This chapter describes the serial Modbus protocol. This chapter is intended for people who need to program a serial Modbus master for their own control system.

The material of this chapter is copyrighted by Modicon, and is used by permission of AEG Schneider Automation (Modicon). This material is included in the Modicon publication *Modicon Modbus Protocol Reference Guide* (PI-MBUS-300 Rev. E). The latest revision is available at Modicon internet homepage *http://www.modicon.com*.

Introducing Modbus Protocol

The Modbus protocol is a serial master–slave protocol. This appendix describes the Modbus protocol only to the level which is required to fully access the ABB drives. The Modbus protocol defines what is serially transmitted on the communication link. The physical interface to the NMBA-01 is half-duplex RS-485.

Transactions on Modbus Networks

Standard Modbus ports on Modicon controllers use an RS-232C compatible serial interface that defines connector pinouts, cabling, signal levels, transmission baud rates, and parity checking. Controllers can be networked directly or via modems.

Controllers communicate using a master – slave technique, in which only one device (the master) can initiate transactions (called 'queries'). The other devices (the slaves) respond by supplying the requested data to the master, or by taking the action requested in the query. Typical master devices include host processors and programming panels. Typical slaves include programmable controllers.

The master can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a message (called a 'response') to queries that are addressed to them individually. Responses are not returned to broadcast queries from the master.

The Modbus protocol establishes the format for the master's query by placing into it the device (or broadcast) address, a function code defining the requested action, any data to be sent, and an error-checking field. The slave's response message is also constructed using Modbus protocol. It contains fields confirming the action taken, any data to be returned, and an error-checking field. If an error occurred in receipt of the message, or if the slave is unable to perform the requested action, the slave will construct an error message and send it as its response.

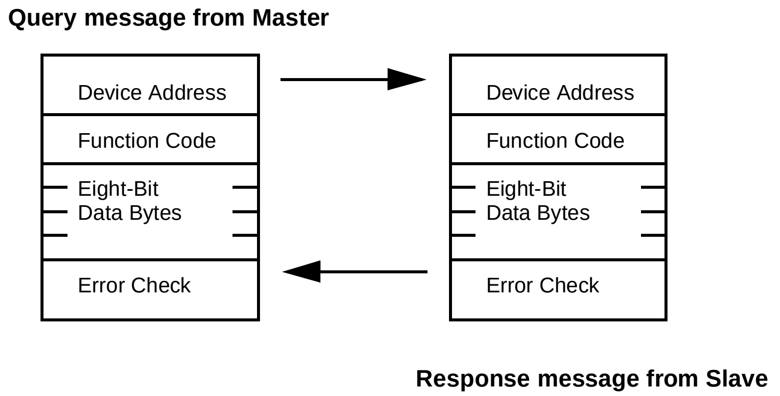


Figure B-1 Master-Slave Query-Response Cycle

The Query: The function code in the query tells the addressed slave device what kind of action to perform. The data bytes contain any additional information that the slave will need to perform the function. For example, function code 03 will query the slave to read holding registers and respond with their contents. The data field must contain the information telling the slave which register to start at and how many registers to read. The error check field provides a method for the slave to validate the integrity of the message contents.

The Response: If the slave makes a normal response, the function code in the response is an echo of the function code in the query. The data bytes contain the data collected by the slave, such as register values or status. If an error occurs, the function code is modified to indicate that the response is an error response, and the data bytes contain a code that describes the error. The error check field allows the master to confirm that the message contents are valid.

The Two Serial Transmission Modes

Controllers can be setup to communicate on standard Modbus networks using either of two transmission modes: ASCII or RTU. Users select the desired mode, along with the serial port communication parameters (baud rate, parity mode, etc.), during configuration of each controller. The mode and serial parameters *must be the same for all devices on a Modbus network*. The selection of ASCII or RTU mode pertains only to standard Modbus networks. It defines the bit contents of message fields transmitted serially on those networks. It determines how information will be packed into the message fields and decoded.

The NMBA-01 supports only the RTU mode. Only the RTU mode is described in this document.

RTU Mode

When controllers are setup to communicate on a Modbus network using RTU (Remote Terminal Unit) mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput than ASCII for the same baud rate. Each message must be transmitted in a continuous stream.

The format for each byte in RTU mode is:

Coding System:	8-bit binary, hexadecimal 0-9, A-F Two hexadecimal characters contained in each 8-bit field of the message
Bits per Byte:	1 start bit 8 data bits, least significant bit sent first 1 bit for even/odd parity; no bit for no parity 1 stop bit if parity is used; 2 bits if no parity
Error Check Field:	Cyclical Redundancy Check (CRC)

Modbus Message Framing

In either of the two serial transmission modes (ASCII or RTU), a Modbus message is placed by the transmitting device into a frame that has a known beginning and ending point. This allows receiving devices to begin at the start of the message, read the address portion and determine which device is addressed (or all devices, if the message is broadcast), and to know when the message is completed. Partial messages can be detected and errors can be set as a result.

RTU Framing

In RTU mode, messages start with a silent interval of at least 3.5 character times. This is most easily implemented as a multiple of character times at the baud rate that is being used on the network (shown as T1-T2-T3-T4 in the figure below). The first field then transmitted is the device address.

The allowable characters transmitted for all fields are hexadecimal 0-9, A-F. Networked devices monitor the network bus continuously, including during the ‘silent’ intervals. When the first field (the address field) is received, each device decodes it to find out if it is the addressed device.

Following the last transmitted character, a similar interval of at least 3.5 character times marks the end of the message. A new message can begin after this interval.

The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame, the receiving device flushes the incomplete message and assumes that the next byte will be the address field of a new message.

Similarly, if a new message begins earlier than 3.5 character times following a previous message, the receiving device will consider it a continuation of the previous message. This will set an error, as the value in the final CRC field will not be valid for the combined messages. A typical message frame is shown below.

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1–T2–T3–T4	8 BITS	8 BITS	n × 8 BITS	16 BITS	T1–T2–T3–T4

Figure B-2 Message Frame

How the Address Field is Handled

The address field of a message frame contains eight bits (RTU). Valid slave device addresses are in the range of 0 – 247 decimal. The individual slave devices are assigned addresses in the range of 1 – 247. A master addresses a slave by placing the slave address in the address field of the message. When the slave sends its response, it places its own address in this address field of the response to let the master know which slave is responding.

Address 0 is used for the broadcast address, which all slave devices recognise. When Modbus protocol is used on higher level networks, broadcasts may not be allowed, or may be replaced by other methods. For example, Modbus Plus uses a shared global database that can be updated with each token rotation.

How the Function Field is Handled

The function code field of a message frame contains eight bits (RTU). Valid codes are in the range of 1 – 255 decimal. Of these, some codes are applicable to all Modicon controllers, while some codes apply only to certain models, and others are reserved for future use. The NMBA-01 supports the function codes 3, 6, and 16 (0x03, 0x06, and 0x10 Hex) only.

When a message is sent from a master to a slave device the function code field tells the slave what kind of action to perform. Example is to read a group of outputs.

When the slave responds to the master, it uses the function code field to indicate either a normal (error-free) response or that some kind of error occurred (called an exception response). For a normal response, the slave simply echoes the original function code. For an exception response, the slave returns a code that is equivalent to the original function code with its most significant bit set to a logic 1.

For example, a message from master to slave to read a group of holding registers would have the following function code:

0000 0011   (Hexadecimal 03)

If the slave device takes the requested action without error, it returns the same code in its response. If an exception occurs, it returns:

1000 0011   (Hexadecimal 83)

In addition to its modification of the function code for an exception response, the slave places a unique code into the data field of the response message. This tells the master what kind of error occurred, or the reason for the exception.

The master device's application program has the responsibility of handling exception responses. Typical processes are to post subsequent retries of the message, to try diagnostic messages to the slave, and to notify operators.

Contents of the Data Field

The data field is constructed using sets of two hexadecimal digits, in the range of 00 to FF hexadecimal. These are made from one RTU character, according to the network's serial transmission mode.

The data field of messages sent from a master to slave devices contains additional information which the slave must use to take the action defined by the function code. This can include items like discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field.

For example, if the master requests a slave to read a group of holding registers (function code 03), the data field specifies the starting register and how many registers are to be read. If the master writes to a group of registers in the slave (function code 10 hexadecimal), the data field specifies the starting register, how many registers to write, the count of data bytes to follow in the data field, and the data to be written into the registers.

If no error occurs, the data field of a response from a slave to a master contains the data requested. If an error occurs, the field contains an exception code that the master application can use to determine the next action to be taken.

The data field can be nonexistent (of zero length) in certain kinds of messages. For example, in a request from a master device for a slave to respond with its communications event log (function code 0B hexadecimal), the slave does not require any additional information. The NMBA-01 does not support the function code 0B hexadecimal. The function code alone specifies the action.

Contents of the Error Checking Field

Two kinds of error-checking methods are used for standard Modbus networks. The error checking field contents depend upon the method that is being used.

ASCII

When ASCII mode is used for character framing, the error checking field contains two ASCII characters. The error check characters are the result of a Longitudinal Redundancy Check (LRC) calculation that is performed on the message contents, exclusive of the beginning 'colon' and terminating CRLF characters.

The LRC characters are appended to the message as the last field preceding the CRLF characters.

RTU

When RTU mode is used for character framing, the error checking field contains a 16-bit value implemented as two 8-bit bytes. The error check value is the result of a Cyclical Redundancy Check calculation performed on the message contents.

The CRC field is appended to the message as the last field in the message. When this is done, the low-order byte of the field is appended first, followed by the high-order byte. The CRC high-order byte is the last byte to be sent in the message.

Additional information about error checking is contained later in this appendix.

How Characters are Transmitted Serially

When messages are transmitted on standard Modbus serial networks, each character or byte is sent in this order (left to right):

Least Significant Bit (LSB) … Most Significant Bit (MSB)

With RTU character framing, the bit sequence is:

With Parity Checking

Start	1	2	3	4	5	6	7	8	Par	Stop

Without Parity Checking

Start	1	2	3	4	5	6	7	8	Stop	Stop

Error Checking Methods

Standard Modbus serial networks use two kinds of error checking. Parity checking (even or odd) can be optionally applied to each character. Frame checking (CRC) is applied to the entire message. Both the character check and message frame check are generated in the master device and applied to the message contents before transmission. The slave device checks each character and the entire message frame during receipt.

The master is configured by the user to wait for a predetermined time-out interval before aborting the transaction. This interval is set to be long enough for any slave to respond normally. If the slave detects a transmission error, the message will not be acted upon. The slave will not construct a response to the master. Thus the time-out will expire and allow the master's program to handle the error. Note that a message addressed to a nonexistent slave device will also cause a time-out. A proper time-out value for the NMBA-01 is 100 ms.

Parity Checking

Users can configure controllers for Even or Odd Parity checking, or for No Parity checking. This will determine how the parity bit will be set in each character.

If either Even or Odd Parity is specified, the quantity of 1 bits will be counted in the data portion of each character (eight for RTU). The parity bit will then be set to a 0 or 1 to result in an Even or Odd total of 1 bits.

For example, these eight data bits are contained in an RTU character frame:

1100 0101

The total quantity of 1 bits in the frame is four. If Even Parity is used, the frame's parity bit will be a 0, making the total quantity of 1 bits still an even number (four). If Odd Parity is used, the parity bit will be a 1, making an odd quantity (five).

When the message is transmitted, the parity bit is calculated and applied to the frame of each character. The receiving device counts the quantity of 1 bits and sets an error if they are not the same as configured for that device. All devices on the Modbus network must be configured to use the same parity check method.

Note that parity checking can only detect an error if an odd number of bits are picked up or dropped in a character frame during transmission. For example, if Odd Parity checking is employed, and two 1 bits are dropped from a character containing three 1 bits, the result is still an odd count of 1 bits.

If No Parity checking is specified, no parity bit is transmitted and no parity check can be made. An additional stop bit is transmitted to fill out the character frame.

CRC Checking

In RTU mode, messages include an error-checking field that is based on a Cyclical Redundancy Check (CRC) method. The CRC field checks the contents of the entire message. It is applied regardless of any parity check method used for the individual characters of the message.

The CRC field is two bytes, containing a 16-bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The CRC is started by first pre-loading a 16-bit register to all 1's. Then a process begins of applying successive 8-bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit if one is used, do not apply to the CRC.

During generation of the CRC, each 8-bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8-bit byte is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the bytes of the message have been applied, is the CRC value.

In ladder logic, the CKSM function calculates a CRC from the message contents. For applications using host computers, a detailed example of CRC generation is contained later in this appendix.

Modbus Function Formats

This chapter describes in detail the data content on every Modbus message supported by the NMBA-01.

How Numerical Values are Expressed

Unless specified otherwise, numerical values (such as addresses, codes, or data) are expressed as *decimal values in the text of this section*. They are expressed as *hexadecimal values in the message fields of the figures*.

Data Addresses in Modbus Messages

All data addresses in Modbus messages are referenced to zero. The first occurrence of a data item is addressed as item number zero. For example:

- The coil known as 'coil 1' in a programmable controller is addressed as coil 0000 in the data address field of a Modbus message.
- Coil 127 decimal is addressed as coil 007E hex (126 decimal).
- Holding register 40001 is addressed as register 0000 in the data address field of the message. The function code field already specifies a 'holding register' operation. Therefore the '4XXXX' reference is implicit.
- Holding register 40108 is addressed as register 006B hex (107 decimal).

Field Contents in Modbus Messages

Figure B-3 'Master Query with RTU Framing' shows an example of a Modbus query message. *Figure B-4 'Slave Response with RTU Framing'* is an example of a normal response. Both examples show the field contents in hexadecimal, and also show how a message is be framed in RTU mode.

The master query is a Read Holding Registers request to slave device address 06. The message requests data from three holding registers, 40108 through 40110. Note that the message specifies the starting register address as 0107 (006B hex).

The slave response echoes the function code, indicating this is a normal response. The ‘Byte Count’ field specifies how many *8-Bit data items* are being returned. It shows the count of 8-bit bytes to follow in the data for RTU.

For example, the value 63 hex is sent as one 8-bit byte in RTU mode (01100011). The 'Byte Count' field counts this data as one 8-bit item, regardless of the character framing method (ASCII or RTU).

How to Use the Byte Count Field: When you construct responses in buffers, use a Byte Count value that equals the count of 8-bit bytes in your message data. The value is exclusive of all other field contents, including the Byte Count field. *Figure B-4 ‘Slave Response with RTU Framing’* shows how the byte count field is implemented in a typical response.

QUERY

Field Name	Example (Hex)	RTU 8-Bit Field
Header		None
Slave Address	06	0000 0110
Function	03	0000 0011
Starting Address Hi	00	0000 0000
Starting Address Lo	6B	0110 1011
No. of Registers Hi	00	0000 0000
No. of Registers Lo	03	0000 0011
Error Check		CRC (16 bits)
Trailer		None
	Total Bytes:	8

Figure B-3 Master Query with RTU Framing

RESPONSE

Field Name	Example (Hex)	RTU 8-Bit Field
Header		None
Slave Address	06	0000 0110
Function	03	0000 0011
Byte Count	06	0000 0110
Data Hi	02	0000 0010
Data Lo	2B	0010 1011
Data Hi	00	0000 0000
Data Lo	00	0000 0000
Data Hi	00	0000 0000
Data Lo	00	0000 0000
Error Check		CRC (16 bits)
Trailer		None
	Total Bytes:	11

Figure B-4 Slave Response with RTU Framing

Function Codes

The NMBA-01 supports three Modbus function codes. These allow the Master to read and write 16-bit integer values to the drive.

03 Read Holding Registers

Reads the binary contents of holding registers (4X references) in the slave. Broadcast is not supported.

Query

The query message specifies the starting register and quantity of registers to be read. Registers are addressed starting at zero: registers 1–16 are addressed as 0–15.

Here is an example of a request to read registers 40108–40110 from slave device 17:

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	03
Starting Address Hi	00
Starting Address Lo	6B
No. of Points Hi	00
No. of Points Lo	03
Error Check CRC	CRC (16-Bits)

Figure B-5 Read Holding Registers - Query

Response The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits.

Data is scanned in the slave at the rate of 125 registers per scan for 984-X8X controllers (984–685, etc.), and at the rate of 32 registers per scan for all other controllers. The response is returned when the data is completely assembled.

Here is an example of a response to the previous query:

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	03
Byte Count	06
Data Hi (Register 40108)	02
Data Lo (Register 40108)	2B
Data Hi (Register 40109)	00
Data Lo (Register 40109)	00
Data Hi (Register 40110)	00
Data Lo (Register 40110)	64
Error Check CRC	CRC (16-Bits)

Figure B-6 Read Holding Registers - Response

The contents of register 40108 are shown as the two byte values of 02 2B hex, or 555 decimal. The contents of registers 40109–40110 are 00 00 and 00 64 hex, or 0 and 100 decimal.

06 Preset Single Register

Presets a value into a single holding register (4X reference). When broadcast, the function presets the same register reference in all attached slaves.

Query

The query message specifies the register reference to be preset. Registers are addressed starting at zero: register 1 is addressed as 0.

The requested preset value is specified in the query data field. The NMBA-01 uses 16-bit values.

Here is an example of a request to preset register 40002 to 00 03 hex in slave device 17:

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	06
Register Address Hi	00
Register Address Lo	01
Preset Data Hi	00
Preset Data Lo	03
Error Check CRC	CRC (16-Bits)

Figure B-7 Preset Single Register - Query

Response

The normal response is an echo of the query, returned after the register contents have been preset.

Here is an example of a response to the query on the opposite page:

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	06
Register Address Hi	00
Register Address Lo	01
Preset Data Hi	00
Preset Data Lo	03
Error Check CRC	CRC (16-Bits)

Figure B-8 Preset Single Register - Response

16 (10 Hex) Preset Multiple Regs

Presets values into a sequence of holding registers (4X references). When broadcast, the function presets the same register references in all attached slaves.

The NMBA-01 allows one or multiple registers to be written at one time using one Preset Multiple Regs function. Only registers within one group can be written at one time. If a write to one of the registers fails, the module will try to write to other registers, but the response will contain a corresponding exception message.

Query

The query message specifies the register references to be preset. Registers are addressed starting at zero: register 1 is addressed as 0.

The requested preset values are specified in the query data field. The NMBA-01 uses 16-bit values. Data is packed as two bytes per register.

Here is an example of a request to preset one register starting at 40002 to 00 0A in slave device 17:

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	10
Starting Address Hi	00
Starting Address Lo	01
No. of Registers Hi	00
No. of Registers Lo	01
Byte Count	02
Data Hi	00
Data Lo	0A
Error Check CRC	CRC (16-Bits)

Figure B-9 Preset Multiple Registers - Query

Response

The normal response returns the slave address, function code, starting address, and quantity of registers preset.

Here is an example of a response to the query shown above.

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	10
Starting Address Hi	00
Starting Address Lo	01
No. of Registers Hi	00
No. of Registers Lo	01
Error Check CRC	CRC (16-Bits)

Figure B-10 Preset Multiple Registers - Response

Exception Responses

Except for broadcast messages, when a master device sends a query to a slave device it expects a normal response. One of four possible events can occur from the master's query:

1. If the slave device receives the query without a communication error, and can handle the query normally, it returns a normal response.
2. If the slave does not receive the query due to a communication error, no response is returned. The master program will eventually process a time-out condition for the query.
3. If the slave receives the query, but detects a communication error (parity, LRC, or CRC), no response is returned. The master program will eventually process a time-out condition for the query.
4. If the slave receives the query without a communication error, but cannot handle it (for example, if the request is to read a non-existent coil or register), the slave will return an exception response informing the master of the nature of the error. The exception response message has two fields that differentiate it from a normal response:

Function Code Field: In a normal response, the slave echoes the function code of the original query in the function code field of the response. All function codes have a most-significant bit (MSB) of 0 (their values are all below 80 hexadecimal). In an exception response, the slave sets the MSB of the function code to 1. This makes the function code value in an exception response exactly 80 hexadecimal higher than the value would be for a normal response. With the function code's MSB set, the master's application program can recognise the exception response and can examine the data field for the exception code.

Data Field: In a normal response, the slave may return data or statistics in the data field (any information that was requested in the query). In an exception response, the slave returns an exception code in the data field. This defines the slave condition that caused the exception. *Figure B-11 'Master Query and Slave Exception Response'* shows an example of a master query and slave exception response.

The field examples are shown in hexadecimal.

QUERY		
Byte	Contents	Example
1	Slave Address	0A
2	Function	01
3	Starting Address Hi	04
4	Starting Address Lo	A1
5	No. of Coils Hi	00
6	No. of Coils Lo	01
7	LRC	4F
EXCEPTION RESPONSE		
1	Slave Address	0A
2	Function	81
3	Exception Code	02
4	LRC	73

Figure B-11 Master Query and Slave Exception Response

In this example, the master addresses a query to slave device 10 (0A hex). The function code (01) is for a Read Coil Status operation. It requests the status of the coil at address 1245 (04A1 hex). Note that only that one coil is to be read, as specified by the number of coils field (0001).

If the coil address is non-existent in the slave device, the slave will return the exception response with the exception code shown (02). This specifies an illegal data address for the slave. For example, if the slave is a 984-385 with 512 coils, this code would be returned.

A listing of Modicon exception codes is in *Table C-1 Standard Exception Codes.*

Table C-1 Standard Exception Codes

Code	Name	Meaning
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the slave. If a Poll Program Complete command was issued, this code indicates that no program function preceded it.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the slave.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for the slave.
04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the slave was attempting to perform the requested action.
05	ACKNOWLEDGE	The slave has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a time-out error from occurring in the master. The master can next issue a Poll Program Complete message to determine if processing is completed.
06	SLAVE DEVICE BUSY	The slave is engaged in processing a long duration program command. The master should retransmit the message later when the slave is free.
07	NEGATIVE ACKNOWLEDGE	The slave cannot perform the program function received in the query. This code is returned for an unsuccessful programming request using function code 13 or 14 decimal. The master should request diagnostic or error information from the slave.
08	MEMORY PARITY ERROR	The slave attempted to read extended memory, but detected a parity error in the memory. The master can retry the request, but service may be required on the slave device.

CRC Generation

The Cyclical Redundancy Check (CRC) field is two bytes, containing a 16-bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The CRC is started by first preloading a 16-bit register to all 1's. Then a process begins of applying successive 8-bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit if one is used, do not apply to the CRC.

During generation of the CRC, each 8-bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. if the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8-bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the characters of the message have been applied, is the CRC value.

A procedure for generating a CRC is:

1. Load a 1 6-bit register with FFFF hex (all 1's). Call this the CRC register.
2. Exclusive OR the first 8-bit byte of the message with the low-order byte of the 16-bit CRC register, putting the result in the CRC register.
3. Shift the CRC register one bit to the right (toward the LSB), zero-filling the MSB. Extract and examine the LSB.
4. (If the LSB was 0): Repeat Step 3 (another shift).
(if the LSB was 1): Exclusive OR the CRC register with the polynomial value A001 hex (101 0 0000 0000 0001).
5. Repeat Steps 3 and 4 until 8 shifts have been performed. When this is done, a complete 8-bit byte will have been processed.
6. Repeat Steps 2 through 5 for the next 8-bit byte of the message. Continue doing this until all bytes have been processed.
7. The final contents of the CRC register is the CRC value.

Placing the CRC into the Message

When the 16-bit CRC (2 8-bit bytes) is transmitted in the message, the low-order byte will be transmitted first, followed by the high-order byte. For example, if the CRC value is 1241 hex (0001 0010 0100 0001):

Addr	Func	Data Count	Data	Data	Data	Data	CRC Lo 41	CRC Hi 12

Figure B-12 CRC Byte Sequence

Example

An example of a C language function performing CRC generation is shown on the following pages. All of the possible CRC values are pre-loaded into two arrays, which are simply indexed as the function increments through the message buffer. One array contains all of the 256 possible CRC values for the high byte of the 16-bit CRC field, and the other contains all of the values for the low byte. Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

The function takes two arguments:

`unsigned char *puchMsg`
A pointer to the message buffer containing binary data to be used for generating the CRC

`unsigned short usDataLen`
The quantity of bytes in the message buffer.

The function returns the CRC as a type unsigned short.

```
/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi [ ] = {
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xCl,0x81,
0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x0l,0xC0,
0x80,0x41,0x0l,0xC0,0x80,0x41,0x00,0xCl,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,
0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,
0x00,0xCl,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,
0x40,0x0l,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,
0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,
0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,
0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,
0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x00,0xCl,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,
0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,
0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40
};
/* Table of CRC values for low-order byte*/
static char auchCRCLo [ ] = {
0x00,0xC0,0xC1,0x01,0xC3,0x03,0x02,0xC2,0xC6,0x06,0x07,0xC7,0x05,0xC5,0xC4,
0x04,0xCC,0x0C,0x0D,0xCD,0x0F,0xCF,0xCE,0x0E,0x0A,0xCA,0xCB,0x0B,0xC9,0x09,
0x08,0xC8,0xD8,0x18,0xl9,0xD9,0xlB,0xDB,0xDA,0xlA,0xlE,0xDE,0xDF,0xlF,0xDD,
0xlD,0xlC,0xDC,0x14,0xD4,0xD5,0x15,0xD7,0x17,0x16,0xD6,0xD2,0x12,0x13,0xD3,
0xll,0xDl,0xD0,0xl0,0xF0,0x30,0x31,0xF1,0x33,0xF3,0xF2,0x32,0x36,0xF6,0xF7,
0x37,0xF5,0x35,0x34,0xF4,0x3C,0xFC,0xFD,0x3D,0xFF,0x3F,0x3E,0xFE,0xFA,0x3A,
0x3B,0xFB,0x39,0xF9,0xF8,0x38,0x28,0xE8,0xE9,0x29,0xEB,0x2B,0x2A,0xEA,0xEE,
0x2E,0x2F,0xEF,0x2D,0xED,0xEC,0x2C,0xE4,0x24,0x25,0xE5,0x27,0xE7,0xE6,0x26,
0x22,0xE2,0xE3,0x23,0xE1,0x21,0x20,0xE0,0xA0,0x60,0x61,0xA1,0x63,0xA3,0xA2,
0x62,0x66,0xA6,0xA7,0x67,0xA5,0x65,0x64,0xA4,0x6C,0xAC,0xAD,0x6D,0xAF,0x6F,
0x6E,0xAE,0xAA,0x6A,0x6B,0xAB,0x69,0xA9,0xA8,0x68,0x78,0xB8,0xB9,0x79,0xBB,
0x7B,0x7A,0xBA,0xBE,0x7E,0x7F,0xBF,0x7D,0xBD,0xBC,0x7C,0xB4,0x74,0x75,0xB5,
0x77,0xB7,0xB6,0x76,0x72,0xB2,0xB3,0x73,0xB1,0x71,0x70,0xB0,0x50,0x90,0x91,
0x51,0x93,0x53,0x52,0x92,0x96,0x56,0x57,0x97,0x55,0x95,0x94,0x54,0x9C,0x5C,
0x5D,0x9D,0x5F,0x9F,0x9E,0x5E,0x5A,0x9A,0x9B,0x5B,0x99,0x59,0x58,0x98,0x88,
0x48,0x49,0x89,0x4B,0x8B,0x8A,0x4A,0x4E,0x8E,0x8F,0x4F,0x8D,0x4D,0x4C,0x8C,
0x44,0x84,0x85,0x45,0x87,0x47,0x46,0x86,0x82,0x42,0x43,0x83,0x41,0x81,0x80,
0x40
};
```

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ;                  / * message to calculate CRC upon* /
unsigned short usDataLen;                 /* quantity of bytes in message*/
{
    unsigned char uchCRCHi = 0xFF;        /* high byte of CRC initialized*/
    unsigned char uchCRCLo = 0xFF;        /* low byte of CRC initialized*/
    unsigned uIndex;                      /* will index into CRC lookup table*/

    while (usDataLen--)                   /* pass through message buffer*/
    {
        uIndex = uchCRCHi ^ *puchMsg++;/* calculate the CRC*/
        uchCRCHi = uchCRCLo ^ auchCRCHi [uIndex] ;
        uchCRCLo = auchCRCLo [uIndex] ;
    }

    return (uchCRCHi << 8 | uchCRCLo);
```

Appendix C – Ambient Conditions

Ambient Conditions, Operation

Ambient operating conditions refer to the conditions the option module is subjected to when installed for stationary use.

Air Temperature: 0 to +50 °C

Relative Humidity: 5 to 95 %, no condensation allowed. Maximum allowed relative humidity is 60 % in the presence of corrosive gases.

Contamination Levels:
Chemical gases: IEC 721-3-3, Class 3C2
Solid particles: IEC 721-3-3, Class 3S2

Installation Site Altitude: 0 to 2000 m above sea level. If the installation site is higer than 2000 m above sea level, please contact your local ABB distributor or office for further information.

Vibration: Max 0.3 mm (2 to 9 Hz), max 1 m/s^2 (9 to 200 Hz) sinusoidal (IEC 68-2-6)

Shock: Max 70 m/s^2, 22 ms (IEC 68-2-27)

Ambient Conditions, Storage

Ambient storage conditions refer to the conditions the option module is subjected to during storage in the protective package.

Temperature: -40 to +70 °C.

Relative Humidity: Less than 95 %, no condensation allowed

Atmospheric Pressure: 70 to 106 kPa

Vibration: Max 0.3 mm (2 to 9 Hz), max 1 m/s^2 (9 to 200 Hz) sinusoidal (IEC 68-2-6)

Shock: Max 100 m/s^2, 11 ms (IEC 68-2-27)

Ambient Conditions, Transportation

Ambient transportation conditions refer to the conditions the option module is subjected to during transportation in the protective package.

Temperature: -40 to +70 °C

Relative Humidity: Less than 95 %, no condensation allowed.

Atmospheric Pressure: 60 to 106 kPa

Vibration: Max 3.5 mm (2 to 9 Hz), max 15 m/s^2 (9 to 200 Hz) sinusoidal (IEC 68-2-6)

Shock: Max 100 m/s^2, 11 ms (IEC 68-2-27)

Bump: Max 300 m/s^2, 6 ms (IEC 68-2-29)

Free Fall: 250 mm

NMBA-01/EN
3AFY 58919772 R0225
EFFECTIVE: 1.2.1999
SUPERSEDES: 1997-01-10

ABB Industry Oy
Drives Group
P.O.Box 184
FIN-00381 Helsinki
FINLAND
Telephone: +358 10 222 000
Telefax: +358 10 222 2681