Motion Control Products

Application note
Integrated Modbus support

AN00198-002

Ready to use PLC function blocks, combine with pre-written Mint application for simple control of e100 drives via Modbus

Introduction

Integrated Modbus support has been introduced on the following ABB motion control products:

NextMove ES/ESB-2 (Modbus RTU) – firmware version 5424 onwards
NextMove e100 (Modbus RTU, Modbus TCP) – firmware version 5633 onwards
MicroFlex e100 (Modbus RTU, Modbus TCP) – firmware version 5633 onwards
MotiFlex e100 (Modbus RTU, Modbus TCP) – firmware version 5633 onwards

New Mint keywords to enable and configure Modbus server operation have been added. Download firmware from www.abbmotion.com or contact motionsupport.uk@abb.com to obtain the relevant Mint System File (.msx) to update a drive / motion controller. Note that Modbus support is only offered on Revision B e100 products (e.g. MFE230A003**B**).

MicroFlex e150 drives support Modbus RTU and TCP as standard (with fixed mappings to Netdata) – please refer to the MicroFlex e150 user manual LT0291A01 for further details.

Support is provided for the following Modbus functions:

03 – Read holding registers
04 – Read input registers
06 – Preset single register
16 – Preset multiple registers
23 – Read / write 4x registers

A new Mint keyword (ModbusParameter) allows configuration of:

Modbus enable
Register mapping
Byte order
Word order
Diagnostics

The inclusion of integrated Modbus protocols on the listed controllers / drives allows connectivity between a wide range of ABB products including AC500 PLCs and CP600 HMIs as well as third party Modbus client devices.

Configuration

Modbus RTU and TCP operation on e100 products is included with firmware update 5633 (or later). Modbus RTU operation on NextMove ES/ESB-2 requires firmware version 5424 (or later) or firmware version 5454 (or later) to be installed (depending on whether the user wishes to use Mint compiler target format 13 or 14 respectively).

All Modbus parameters are configured via a new Mint keyword – ModbusParameter.
Before enabling Modbus operation it is necessary to set the correct byte and word order to suit the connected Modbus client (master) and to configure how Modbus registers in the received data packets are mapped to internal data areas in the Mint controllers.

value = ModbusParameter (bus, index)
ModbusParameter (bus, index) = value

Mint pre-defined constant values for bus are:

_busETHERNET	5	For Modbus TCP parameters
_busSERIAL1	6	For Modbus RTU parameters

Index		**Value**
_mpENABLE	0	0 = Disabled, 1 = Enabled
_mpREGISTER_MAPPING	1	0 = NetData array (e100 default), 1 = Comms array (ESB-2 default)
_mpBYTE_ORDER	2	0 = Big Endian (default), 1 = Little Endian
_mpWORD_ORDER	3	0 = Big Endian, 1 = Little Endian (default)
_mpDROPPED_FRAMES	6	Read only counter of number of invalid packets received
_mpDEBUG	7	Parameter used to display Modbus diagnostics

Mint pre-defined constant values for _mpREGISTER_MAPPING are:

_rmNET_DATA	0	To access netinteger / netfloat data
_rmCOMMS_ARRAY	1	To access commsinteger / comms data

NextMove ESB-2 is only provided with Comms array data so attempting to configure a Netdata register mapping on this controller will result in a 'Data Out of Range' error.

The default byte and word orders are configured to match the majority of Modbus devices on the market today. However, ABB PLCs and CP600 HMIs use 'Big Endian' word order so when connecting Mint controllers to ABB Modbus clients it is necessary to include…ModbusParameter (bus, _mpWORD_ORDER) = 0…in the Mint startup block code.

Modbus RTU is a serial based protocol and operation is supported using 8 data bits, 1 stop bit and no parity. The following baud rates are supported on all controllers:
19200
38400
57600
115200

Note that, due to timing constraints, 9600 baud is not supported. The baudrate to be used is set using the Mint SERIALBAUD(_Term1) keyword in the Mint program (or via the Mint Workbench 'Connectivity' page). As Modbus RTU packets include a node address it is also important to set the correct value for BUSNODE(_busSERIAL1) in the Mint program (or via the Mint Workbench 'Connectivity' page). Support is included for broadcast write functions (i.e. to Node address 0) so the user should avoid configuring a controller as Node 0 unless they intend to use broadcast functions.

When using Modbus TCP the controllers / drives use the standard Modbus port 502. At present this port cannot be changed but provision to allow this port to be modified may be added in a future firmware release. The IP address used by Modbus TCP is set by the rotary address switches on the front of the e100 product; 192.168.100.x (where x is the switch setting).

To enable Modbus server (slave) operation the Mint program should issue….
ModbusParameter (bus, _mpENABLE) = 1 (where bus = _busETHERNET for Modbus TCP or _busSERIAL1 for Modbus RTU).

Enabling Modbus RTU server operation automatically disables both Host Comms Protocol (HCP1/2) and ABB Binary Protocol (BBP) functionality on the serial port. Enabling Modbus RTU also prevents the controller from directing data from Mint PRINT statements to the serial port to avoid corruption of Modbus data packets.

Register Mappings

All supported Modbus functions target a common data area in the Mint controller as set by the Mint keyword ModbusParameter (_mpREGISTER_MAPPING). These data areas have a fixed mapping with respect to the Modbus registers on the server as shown by the table below (equivalent AC500 addresses are also shown for reference):

Server Modbus register	AC500 address		Mint Comms array (Comms=Real, Commsinteger = DWord)		Mint Netdata array (Netfloat = Real, Netinteger = DWord)	
0	%MW0.0	%MD0.0	Invalid	Invalid	Element 0 LSW	Element 0
1	%MW0.1		Invalid		Element 0 MSW	
2	%MW0.2	%MD0.1	Element 1 LSW	Element 1	Element 1 LSW	Element 1
3	%MW0.3		Element 1 MSW		Element 1 MSW	
4	%MW0.4	%MD0.2	Element 2 LSW	Element 2	Element 2 LSW	Element 2
5	%MW0.5		Element 2 MSW		Element 2 MSW	
…	---	---	---	---	---	---
198	%MW0.198	%MD0.99	Element 99 LSW	Element 99	Element 99 LSW	Element 99
199	%MW0.199		Element 99 MSW		Element 99 MSW	
200	%MW0.200	%MD0.100	Invalid	Invalid	Element 100 LSW	Element 100
201	%MW0.201		Invalid		Element 100 MSW	
202	%MW0.202	%MD0.101	Invalid	Invalid	Element 101 LSW	Element 101
203	%MW0.203		Invalid		Element 101 MSW	
…	---	---	---	---	---	---
1996	%MW0.1996	%MD0.998	Invalid	Invalid	Element 998 LSW	Element 998
1997	%MW0.1997		Invalid		Element 998 MSW	
1998	%MW0.1998	%MD0.999	Invalid	Invalid	Element 999 LSW	Element 999
1999	%MW0.1999		Invalid		Element 999 MSW	

LSW – Least Significant Word : MSW – Most Significant Word

The Mint Comms array provides 99 elements (1-99). If the client attempts to access Comms element 0 or Comms elements greater than 99 the Mint controller will return the 'Invalid Data Address' Modbus exception packet.

The Mint Netdata array (only supported on e100 products) provides 1000 elements (0-999). If the client attempts to access Netdata elements greater than 999 the Mint controller will return the 'Invalid Data Address' Modbus exception packet.

Example Mint Code

The following code snippets show typical Mint code that may be included in a controller / drive Startup block:

Example Mint code – Mint Modbus RTU slave connected to AC500 or CP600 using Comms array

```
BUSNODE(_busSERIAL1) = 2          'Mint controller is node 2 on RTU network
SERIALBAUD(_Term1) = 57600        'Running at 57.6kbaud
ModbusParameter (_busSERIAL1, _mpBYTE_ORDER)  = 0  'Use big endian byte order
ModbusParameter (_busSERIAL1, _mpWORD_ORDER) = 0  'Use big endian word order
ModbusParameter (_busSERIAL1, _mpREGISTER_MAPPING) = _rmCOMMS_ARRAY
ModbusParameter (_busSERIAL1, _mpENABLE) = 1
```

Example Mint code – Mint Modbus TCP slave connected to AC500 or CP600 using Netdata array

```
'IP address is set by rotary switches….192.168.100.x
ModbusParameter (_busETHERNET, _mpBYTE_ORDER)  = 0  'Use big endian byte order
ModbusParameter (_busETHERNET, _mpWORD_ORDER) = 0  'Use big endian word order
ModbusParameter (_busETHERNET, _mpREGISTER_MAPPING) = _rmNET_DATA
ModbusParameter (_busETHERNET, _mpENABLE) = 1
```

Example Mint code – Mint Modbus RTU slave connected to third party Modbus client using little endian word order and Netdata

```
BUSNODE(_busSERIAL1) = 4          'Mint controller is node 4 on RTU network
SERIALBAUD(_Term1) = 38400        'Running at 38.4kbaud
ModbusParameter (_busSERIAL1, _mpBYTE_ORDER)   = 0  'Use big endian byte order
ModbusParameter (_busSERIAL1, _mpWORD_ORDER) = 1  'Use little endian word order
ModbusParameter (_busSERIAL1, _mpREGISTER_MAPPING) = _rmNET_DATA
ModbusParameter (_busSERIAL1, _mpENABLE) = 1
```

For controllers / drives supporting both Ethernet and serial channels (e.g. NextMove e100, MicroFlex e100) it is possible to configure / enable Modbus TCP operation on Ethernet and at the same time configure / enable Modbus RTU operation on the serial port.

Mint Events

Mint events operate differently depending on the controller being used:
NextMove ESB-2: Comms events will be raised whenever the Modbus client writes to Comms locations 1 to 5 (i.e. data does not necessarily have to change for an event to be raised).

e100 products: Comms events will be raised whenever the Modbus client writes modified data to Comms locations 1 to 10(i.e. data must change for an event to be raised). Netdata events operate in the same manner and will only be raised whenever the Modbus client writes modified data to Netinteger / Netfloat locations 0 to 31.

Physical Connection

The table below shows the physical connection possibilities for Mint products supporting integrated Modbus protocols. AC500 and CP600 products are included for reference.

Connection Type	**NextMove e100**	**NextMove ESB-2**	**MicroFlex e100**	**MotiFlex e100**	**AC500**	**AC500 Eco**	**CP600**
RS232	Yes	Yes (by variant)	No	No	Yes	No	Yes
2 wire RS485	No	No	Yes	Yes	Yes	Yes	Yes
4 wire RS422	Yes	Yes (by variant)	No	No	No	No	Yes
Ethernet	Yes	No	Yes	Yes	Yes	Yes (by variant)	Yes

When using 2-wire RS485 be sure to include 120 ohm terminating resistors between data lines A and B at each end of the network to avoid data corruption.

When using 4-wire RS422 be sure to include 120 ohm terminating resistors between RX+ and RX- at each end of the network to avoid data corruption.

When using Modbus TCP / Ethernet, wire the network in a 'star' type configuration using an Ethernet switch between the client and the connected server(s). Straight-through or crossover cables may be used between the e100 device and the switch. If the e100 product is part of an Ethernet Powerlink (EPL) network then it is also necessary to include an EPL Router (Catalog No. OPT036-501) between the switch and the e100 device.

Important: AC500 and CP600 products use a non-standard RS232 pinout so be sure to check the relevant product manual before connecting other serial devices to these products. Failure to observe the correct pinout may result in damage to the connected device.

Diagnostics

The ModbusParameter keyword provides two types of diagnostics.

A. Reading ModbusParameter (bus, _mpDROPPED_FRAMES) will return a counter indicating how many Modbus packets have been rejected (e.g. invalid checksum, incomplete message……in summary, any message for which neither a valid nor exception response has been generated)

B Writing a terminal channel value to ModbusParameter (bus, _mpDEBUG) will reset diagnostic counters and setup which terminal channel is to be used to display diagnostic information. Reading ModbusParameter (bus, _mpDEBUG) will then return which terminal channel is being used for diagnostics and will display some summary information on the selected terminal channel (e.g. last packet received, last packet transmitted, error counters etc..)

Example:

```
ModbusParameter (_busETHERNET, _mpDEBUG) = _Term2
? ModbusParameter (_busETHERNET, _mpDEBUG)
```

If these commands were entered on a NextMove e100 the controller would display diagnostic information about Modbus TCP on the USB terminal (i.e. Workbench terminal window).

Example debug output:

```
Valid: 2
Dropped: 0.  Exception: NO_EXCEPTION
Rx: 8 bytes
Node: 2   Fn code (hex): 03
Data (hex): 00 02 00 01
CRC (hex): f9 25
Tx: 7 bytes
Node: 2   Fn code (hex): 03
Data (hex): 02 00 00
CRC (hex): 44 fc
```

Note: Due to issues with the UART when using the 2 wire RS485 port on MicroFlex e100, every time the drive responds to a Modbus frame an additional null character is received internally. This is then interpreted as a dropped frame (invalid length) and the diagnostic information will reflect this. It is therefore recommended to only use the debug parameter for RS232 and Ethernet connections.

Contact us

For more information please contact your local ABB representative or one of the following:

www.abbmotion.com
www.abb.com/drives
www.abb.com/drivespartners
www.abb.com/PLC