Bristol Babcock Inc
Watertown, Connecticut

BBI NETWORK 3000:A/MI PROTOCOLS INTERFACE

FUNCTIONAL SPECIFICATION FOR 700 / 500 SERIES MASTER CUSTOM MODULES

Third Issue
14 October, 1997

1. INTRODUCTION

This document describes the implementation of an interface between BBI's Network 3000 3330/3335/3310 system (asynchronous ports only) and the Amocams/Modular, Inc. (AMI) RTU protocols (MicroRemote ASC-500 series and AI-Net), with the BBI 33xx device acting as a Master. The interface is achieved via a 'Custom Module' implemented in firmware for use with 33xx 'AK' version firmware and later (i.e. it can only be used on versions of 33xx software developed on the 'PC' platform). 386 Protected Mode Version is also available.

2. FUNCTIONAL OVERVIEW

The Bristol Babcock 33xx controller functions as the master station on a master/slave communication link. A port on the 33xx is connected via a standard RS232 point-to-point or Multidrop link to the AMI RTUs. The commands available over the link are described in references 1 and 2.

An ACCOL 'load' will need to run within the 33xx. This load will need to contain CUSTOM calls to send data to or request data from the RTUs. This data will be held in the 33xx in the form of standard ACCOL signals. Translation between the ACCOL signals and the messages to and from the RTUs will be handled by the Custom Module firmware. The provision of ACCOL loads for any target application is not part of this project, though test loads, used during the testing of this implementation, are available as part of the BBI 'release' documentation, and should be used to provide examples of the functionality provided. See Appendix B.

Each port of the 33xx can be associated with a separate communication link to an AMI RTU. This gives a maximum of 4 links per 3330/35. An area of memory needs to be allocated within the Accol load for use as buffer space.

All read/write commands available will be initiated by the 33xx. All operations will be under control of an ACCOL program. No automatic polling is performed by the custom module - any periodic polling for input must be performed from the ACCOL load, by appropriate CUSTOM calls within an ACCOL task running at an appropriate rate. No support is provided for 700 series unsolicited 'Report by Exception' commands sent by a slave.

In theory, both 500 series devices and 700 series devices can be connected to the same link. For this Custom module, in this case, the line speed must be the same for all the devices on one link.

Note that a 33xx master cannot co-exist with any other 'master' on the same link.

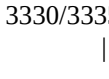
3. COMMUNICATIONS INTERFACE

3.1 General Details

Full details of the available cable configurations are described in references 1 and 2. It is assumed that data sent from the 33xx will not appear at the 33xx RX line. RTS/CTS should either be looped back at the 33xx end, or looped via a Modem connection. These are important timing considerations which must be taken into account - these can be tuned via the Custom Module parameters. The 33xx makes no use of other Modem controls, so these will have to be wired as required locally at a Modem (if used).

Each AMI RTU used has a unique address. When the 33xx sends a command, only the RTU addressed will respond. Thus the 33xx treats the link to each RTU as a logical 'point-to-point' link.

The 500 series and AI-Net protocols are defined such that both may exist on the same link so that the two RTU types may be mixed on the same communications line. However they must both operate at the same Baud Rate.



(Modem Interface Unit)

|

|-- ASC-500 series

|

|-- AI-Net (700 series devices)

ASC500 series

- Line Parameters - normally 600 bits per second (but configurable within Custom Module to 300/600/1200/2400/4800/9600 bits per second). 8 data bits (LSB first), 1 start bit, one stop bit, parity bit controlled by protocol
- Asynchronous RS232. Should be compatible with 33xx RS423, otherwise an interface converter may be needed (see below).
- Some message exchanges are of form 'message' while others are of the form 'message and response' with each transaction initiated by 33xx.
- A timeout on receiving a valid response message, where applicable, will be configurable on a per-message basis via Accol. The number of 'retry' re-transmissions is configurable. A command will not be sent until the response to a previous command sent is received (or retries exhausted). It should be borne in mind that if Timeouts are set too long, and too many retries are made, then the whole link (to ALL RTUs on the link) may be held up by the delay in completing an individual transaction .

AI-Net.

- Line Parameters - 300/600/1200/2400/4800/9600/19200 bits per second. 8 data bits (LSB first), 1 start bit, one stop bit, even or odd parity
- Asynchronous RS232. Should be compatible with 33xx RS423, otherwise an interface converter may be needed (see below).
- All message exchanges are of form 'message and response' with each transaction initiated by 33xx.
- A timeout on receiving a valid response message will be configurable on a per-message basis via Accol. The number of 'retry' re-transmissions is configurable. A command will not be sent until the response to a previous command sent is received (or retries exhausted). It should be borne in mind that if Timeouts are set too long, and too many retries are made, then the whole link (to ALL RTUs on the link) may be held up by the delay in completing an individual transaction. Some 700 series commands (e.g. Writing Configuration tables) can involve a long delay (maybe up to 20 seconds) before the RTU provides the final 'ack'. In these cases, the timeouts must allow for this.

Note: 33xx actually transmits using RS423. It has been found in the past that some RS232 implementations are not RS423 compatible. Whether any extra 'level matching' hardware is required should be established during testing. Provision of any such hardware is not covered by this project.

3.2 Test Configuration

This implementation of the 500 series protocol was tested on the following configuration:

3330 (Port B)

|

(Modem Interface Unit AI 830 MIU)

|
|-- ASC-558 series RTU
|

In addition, the Custom Module was tested using a PC program to simulate other 500 series devices.

3330 to MIU cable details

	3330 (15 pin)		MIU (25 pin)
2	----------		2
4	----------		3
8	----------		4
	|		
6	----|		20-----|
			|
12	-----|		8 ----|
	|		
14	-----|		
9	----------		7

4. DESCRIPTION OF MESSAGE FORMATS AND USAGE

4.1 ASC-500 series protocol

Some commands are of the form ‘Command’ and some are of the form ‘Command/Answer’, with all commands being sent by the 33xx, with any corresponding answers returned by the RTU

The meaning of each Command/Answer type is governed by the ‘command’ field of the Command message.

The following describes the format of all available commands and answers. Mnemonics used are described subsequently. More details of the available messages are given in reference 1.

4.1.1 Command Message Format

All commands to the ASC-500 have the following format:

Byte	Value	Parity	Description
0	7F	Odd	Reset word
1,2	FF,0F	Even	Series 300 lockout code
3		Even	Address
4		Even	Address multiplier
5		Even	Command

Some commands have a few bytes of following data from byte 6.

4.1.2 Answer Message Format

The response to a command message varies by MicroRemote type. The various formats are defined below.

4.1.2.1 ASC-551 status monitor

Command Word - one byte	Meaning
F0	Normal scan

Response:

Byte	Value	Parity	Description
0		Odd	Address
1		Odd	Address Multiplier
2		Odd	Startus Word 1
3		Odd	Status Word 2
4	FF	Even	Set Word
5		Odd	CheckWord

4.1.2.2 ASC-554 relay card

Command Word - one byte	Bit	Meaning
	0	Relay 1
	1	Relay 2
A “1” selects the relay	2	Relay 3
A “0” deselects the relay	3	Relay 4
	4	Relay 5
	5	Relay 6
	6	Relay 7
	7	Relay 8

Response:
None

4.1.2.3 ASC-555 analog/digital converter

Command Word - one byte	Meaning
F0	Normal data scan

Response:

Byte	Value	Parity	Description
0		Odd	Address
1		Odd	Address Multiplier
		Odd	(2 to 16 data words)
x	FF	Even	Setword
x		Odd	Checkword

4.1.2.4 ASC-556 accumulator

Command Word - one byte	Meaning
F0	Normal data scan
E1	Reset 16-bit channels 1 and 2
D2	Reset 16-bit channels 3 and 4
C3	Reset all channels

Response:

Byte	Value	Parity	Description
0		Odd	Address
1		Odd	Address Multiplier
		Odd	(4 or 8 data words)
x	FF	Even	Setword
x		Odd	Checkword

4.1.2.5 ASC-557 digital/analog converter

Command:

Byte	Value	Parity	Description
0	7F	Odd	Reset word
1,2	FF,0F	Even	Series 300 lockout code
3		Even	Address
4		Even	Address multiplier
5		Even	Command
7,8,9		Even	Data
10	F0	Even	Load command

Command Word - one byte	Meaning
E1	Load channel 1 setpoint
D2	Load channel 2 setpoint
B4	Load channel 3 setpoint
C3	Load channels 1 and 2 setpoints
A5	Load channels 1 and 3 setpoints
96	Load channels 2 and 3 setpoints
87	Load channels 1, 2 , and 3 setpoints

Response:
None

4.1.2.6 ASC-558 combination

Command Word - one byte	Meaning
F0	Normal data scan
E1	Relay 1
D2	Relay 2
C3	Relay 3
B4	Relay 4
A5	Relay 5
96	Relay 6
87	Relay 7
78	Relay 8
69	Reset all accumulators

Response:

Byte	Value	Parity	Description
0		Odd	Address
1		Odd	Address Multiplier
		Odd	(2 status words, 8 or 16 analog data words plus 4 accumulator words)
x	FF	Even	Setword
x		Odd	Checkword

4.1.2.7 ASC-574 Combination

Command Word - one byte	Meaning
F0	Normal data scan
E1	Reset all accumulators
D2	Function Control, followed by 2 control bytes ,the first for relays 1 4, the second for relays 5 to 8, each either 96,E1,5A,0F for relays 2 and 3, relay 1, relays 1 and 3, or relays 1,2,3,4 respectively.

Response (to data scan only):

Byte	Value	Parity	Description
0		Odd	Address
1		Odd	Address Multiplier
		Odd	(2 status words,8 or 16 analog data words plus 4 accumulator words)
x	FF	Even	Setword
x		Odd	Checkword

4.1.3 Protocol Considerations

4.1.3.1 Calculation of checkword field

This is a 8 bit binary count of the number of “1” to “0” transmissions in the data reply message. See ref 1.

4.1.3.2 Transmission protocol timings

The following should be considered :

a) Note that there are parity changes embedded within the message formats for 500 series. It will be necessary for the 3330/3335 to sense the parity of incomming data to properly sense the end of data. Similarly, on Transmit from the 3330, parity needs to be changed affter the Reset Word is transmitted. The Custom Module needs to wait until the character has cleared the USART before changing the parity. As there is an ‘all clear’ status on the USARTS used, but no asscociated interrupt, this can be checked after a time delay. This is tunable via the Custom Module Signal List.
b) Response timeouts should allow for the normal and worst case response times. This can be specified in the Custom Module signal list for each request. This must allow for ‘turn around delay’ and ‘System timeout’ periods (see ref 3).
c) In the event of no valid response being received within the timeout period, retries may be performed, specified by parameter P1 (see below).
d) A ‘transmit timeout’ may be specified. This is effectively a timeout on CTS not appearing at the 33xx USART.

4.2 AI-Net protocol

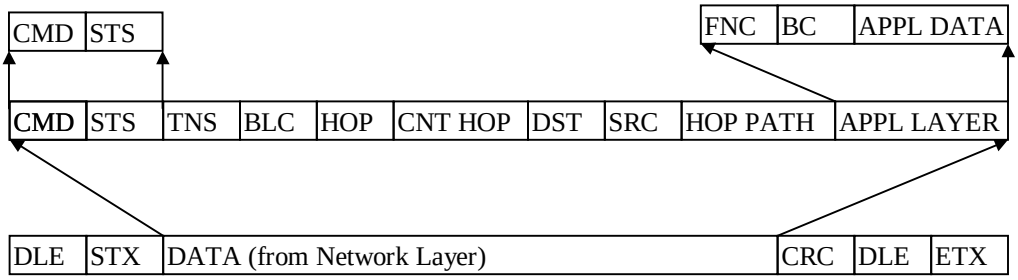
All commands are of the form 1 ‘Command/Answer’, with all commands being sent by the 33xx, with corresponding answers returned by the RTU

The meaning of each Command/Answer type is governed by the ‘command’ field (CMD/FNC) of the Command message.

The following describes the format of all available commands and answers. More details of the available messages are given in reference 2.

4.2.1 General Message Format

All messages in the AI-Net protocol have the following format:



Linker layer fields:

DLE STX: Start of message header.

DATA: All datra that the packet carries starts here.

CRC: Cyclic Redundancy Code.

DLE ETX: End of message tail.

Network layer fields:

CMD: Indicates message type being sent.

STS: Status byte.

TNS: The two TNS bytes conatin a unique 16 bit transaction identifier field. Note: during Multiblock operations, TNS remains constant.

BLC: Byte field that shows the remaining number of blocks to be sent.

HOP: Nibble field that indicates the number of addresses (words) in the hop map.

CNT HOP: Nibble field that indicates how many hops are left until the ultimate destination is reached.

DST: 16 bit value which is the destination address of the packet.

SRC: 16 bit value which is the destination source of the packet

HOP PATH: The list of all RTUs the message must pass through to reach its ultimate destination.

APPL LAYER: Application layer command and data fields.

Application layer fields:

FNC: Application Layer function or command code. The following subset of commands will be supported by this driver:

00	Read scan list
01	Read configured table
02	Write configured table
05	Read register
06	Write register
21	Write scan table
2C	Read register range
2D	Write register range
2E	Clear reset status

BC: 16 bit value indicating the byte count of all the Application Layer data to be returned.

DATA: the application data returned. The format of the data varies by FNC.

4.2.2 Answer Message Format

The response to a command message varies by function code. The various formats supported are defined below.

4.2.2.1 Read scan table

Host to RTU

80	00	00	0001	05		
CMD	STS	Function code				
			Byte count of application layer			
				Table number		

RTU to Host

40	00	00	0006	05	7f	(Data)
CMD	STS	Function code				
			Byte count of application layer			
				Table number		
					Right justified packed Booleans	
						First byte of non-Boolean data

4.2.2.2 Read configured table

Host to RTU

80	00	01	0003	NN	NN	NN
CMD	STS	Function code				
			Byte count of application layer			
				Table type		
					Table number	
						Data byte channel number when used

RTU to Host

40	00	01	NNNN	NN	NN	(Data)
CMD	STS	Function code				
			Byte count of application layer			
				Table type		
					Table number	
						Data (varies by table type)

4.2.2.3 Write configured table

Host to RTU

80	00	02	NNNN	NN	NN	(Data)
CMD	STS	Function code				
		Byte count of application layer				
			Table type			
				Table number		
					Data	

RTU to Host

40	00	02	0003	NN	NN	(Data)
CMD	STS	Function code				
		Byte count of application layer				
			Table type			
				Table number		
					Data	

4.2.2.4 Read register

Host to RTU

80	00	05	NNNN	(Data)	
CMD	STS	Function code			
		Byte count of application layer			
			List of registers to be read		

RTU to Host

40	00	05	NNNN	(Data)	
CMD	STS	Function code			
		Byte count of application layer			
			Register data		

4.2.2.5 Write register

Host to RTU

80	00	06	NNNN	NN	(Data)	
CMD	STS	Function code				
		Byte count of application layer				
			Register number			
				Register data		

RTU to Host

40	00	06	0000		
CMD	STS	Function code			
		Byte count of application layer			

4.2.2.6 Write scan table

Host to RTU

80	00	21	NNNN	NN	(Data)	
CMD	STS	Function code				
		Byte count of application layer				
			Table number			
				Register data for specified scan table		

RTU to Host

40	00	21	0001	NN	
CMD	STS	Function code			
		Byte count of application layer			
			Table number		

4.2.2.7 Read register range

Host to RTU

80	00	2C	0004	NN	NN
CMD	STS	Function code			
		Byte count of application layer			
			Starting register number		
				Number of registers	

RTU to Host

40	00	2c	NNNN	(Data)
CMD	STS	Function code		
		Byte count of application layer		
			Register data packed according to type	

4.2.2.8 Write register range

Host to RTU

80	00	2D	NNNN	NNNN	NNNN	(Data)
CMD	STS	Function code				
		Byte count of application layer				
			Starting register number			
				Number of registers		
					Register data	

RTU to Host

40	00	2D	0000
CMD	STS	Function code	
		Byte count of application layer	

4.2.2.9 Clear reset status

Host to RTU

80	00	2E	0000
CMD	STS	Function code	
		Byte count of application layer	

RTU to Host

40	00	2E	0000
CMD	STS	Function code	
		Byte count of application layer	

4.2.3 Protocol Considerations

4.2.3.1 Link layer protocol

This driver will not support the link layer line access protocol and therefore will not support unsolicited report by exception.

4.2.3.2 Calculation of CRC field

The CRC value is calculated based on all data bytes using the polynomial x16 + x15 + x2 + x0. The CRC is not calculated on the STX or ETX sequences. The CRC is calculated prior to DLE stuffing. See ref. 2.

4.2.3.3 Transmission protocol timings

The following should be considered :

a) Note that the AI-Net protocol use the non-transparent data link escape, DLE, sequences to distinguish data from control characters. Therefore, the messages must be DLE stuffed on send and DLE stripped on receipt.
b) Response time-outs should allow for the normal and worst case response times. This can be specified in the Custom Module signal list for each request. This must allow for 'turn around delay' and 'System time-out' periods .
c) In the event of no valid response being received within the time-out period, retries may be performed, specified by parameter P1.
d) A transmit timeout needs to be specified. This is effectively a timeout on CTS not appearing at the 33xx USART.

5. ACCOL USER INTERFACE

5.1 General Description

Communication is achieved in the normal way, by using the CUSTOM module statement within an ACCOL executing task (not TASK 0). The CUSTOM call needs a signal list to be specified, which gives details of the requested operation.

The characteristics of the port used must be set up (see below).

The following sections describe the format of the Custom Module call from an ACCOL program. Please refer to the examples in Appendix A.

5.2 Memory allocation

On 33xx 186 devices, and 386 cards operating in 'Real' mode, Memory must be allocated for use by the Custom Module for buffer space. 3K bytes is needed for each port used. This memory is allocated by including a statement of the following form within the *MEMORY section of the Accol load: E.g. for 3 ports used:

CUSTOM_SIZE 9216

For 386 Protected Mode, the memory is allocated via the MTOS memory pool. Enought memory needs to be allocated via the 'GLOBAL_STORE' statement in the *MEMORY section of the Accol load. Again, 3K bytes is required for each port used. E.g.

GLOBAL_STORE 10

Note that the allocation needs to be enough for ALL uses (AMI and non-AMI).

5.3 Custom Port Configured for AMI Mode

The following is a list of the parameter field values for a Custom port when configured for AMI mode.

- MODE: Set this field to 24 to indicate AMI mode.
- BAUD: Set this field to 110, 300, 600, 1200, 2400, 4800, or 9600 to indicate the communication baud rate.
- CHARACTER LENGTH: This field is not used. The number of bits per character is fixed at 8 internally.
- STOP BITS: This field is not used. The number of stop bits is fixed at 1 internally.
- PARITY: Not used for the ASC-500 series, set this field to Odd, Even or None to indicate the type of character parity to use for AI-Net.
- P1:. No of communications attempts (retries+1). Normally 1 or 2. Enforced within driver to a maximum of 3
- P2: The host address to be used by the AI-Net protocol.

5.4 ACCOL Terminal Assignments

The following is a list of the terminal values for the custom module when configured for AMI mode.

- MODE: A value of 24 indicates AMI mode.
- LIST: The number of the signal list that contains the signals used by this module to control the interface. This signal list is referred to as the AMI signal list and is described later.
- STATUS: The value of this terminal is a status code representing the module's status. The status code is used to indicate various communication states and error conditions. Communication and processing of reply messages are aborted when the status code is negative. The following is a list of possible status code values and their definition.

Table A-1: Error/Status messages - AMI

0	Communication completed successfully
1	Communication requested, waiting to send
2	Command message sent, waiting for reply
-2	invalid Custom (Modbus) List specified
-3	invalid Port specified
-4	invalid Slave Address specified
-5	invalid Function Type code specified
-6	invalid Address specified
-7	invalid Number specified
-8	invalid Response Timeout value specified
-9	invalid I/O List specified
-10	invalid Format number specified
-16	No expanded memory allocated by ACCOL load
-18	invalid RTS Time value specified
-19	invalid FUNC factor value specified
-20	invalid MESS Factor value specified
-25	input character overrun error detected
-26	input character parity error detected
-27	input character framing error detected
-28	input security check (CRC) error detected (or input message too long)
-29	timed out waiting for response
-31	transmit timed out waiting for CTS
-32	unexpected I/O failure
-33	invalid I/O List specified
-34	invalid Format number specified
-35	invalid TX Timeout value specified
-36	invalid hop array specified
-37	error in hop array (e.g. no -1 at end)
-38	bad apl byte count in list
-39	bad hop array number in list
-40	bad done list
-41	bad command signal in done list
-42	bad STS value signal in done list
-43	bad ‘valid’ signal in done list
-44	bad PFP signal in done list
-45	bad cold-start flag signal in done list
-46	bad warm-start flag signal in done list
-47	bad recevie count signal in done list
-48	read register array failure
-49	read register array overflow
-50	Tx Character not clear of USART before time elapsed

The following are format errors and warnings. Those with positive signs are warnings - data sent or returned is usually valid.:

101	An input signal is control inhibited (formats only)
102	Attempt to store signal into a constant (formats only)
103	An input string signal value was truncated (formats only)

104	Attempt to store into a Read Only data array
116	Format error - attempt to go beyond end of buffer on read
117	Format error - list/format specifies more than totalcount
118	Format error - overflow in write regs data array
-101	Format error - unsupported field descriptor
-102	Format error - attempt to use signal beyond end of list
-103	Format error - too many levels of parenthesis
-104	Format error - unmatched right parenthesis
-105	Format error - sub format number does not exist
-106	Format error - too many levels of sub formats
-107	Format error - invalid data array number selected
-108	Format error - data array has not been defined
-109	Format error - attempt to use cell beyond end of array
-110	Format error - signal or cell type must be analog
-111	Format error - signal type must be string
-112	Format error - sig or cell type must be analog or logical
-113	Format error - BCD input value is invalid
-114	Format error - bad analog value for BCD output
-115	Format error - unexpected input signal store failure
-116	Format error - attempt to go beyond end of buffer
-117	Format error - invalid floating point value
-119	Format error - AI_NET Multiblock transmit error
-120	Format error - AI_NET Multiblock receive error

5.5 AMI Signal List

The signal list specified via the Custom module's List terminal (AMI signal list) must be organized as follows. Some fields have differing meanings depending on the command code. Please refer to examples suggested in Appendix A.

The same list is used for both 500 series and 700 series commands. Fields not used for 500 series commands must be present, although they are not checked or used. Similarly, fields not used for 700 series commands must be present.

Note: the Accol load should not only check the status, but also check the STS.ERROR signal in the 'Done' list after each transaction. Other fields in the 'Done' list can be checked as required.

- Signal 1: Port no (Used for all commands)

  An analog signal whose value represents the 3330 port number to be used for communication with RTU. The port must be a Custom port configured for AMI mode. The values are assigned as follows:

Signal Value	Port used for Communication
1..0	A
2.0	B
3.0	C
4.0	D

  .....

- Signal 2: Address (Used for all commands)

  An analog signal whose value specifies the RTU address.

- Signal 3: Address multiplier (500 series commands only)

  This is an analog signal whose value specifies the address multiplier on the ASC-500.

- Signal 4: Hop Map Array Number (700 series commands only)

  This is an analog signal giving the number of an analog data array containing the hop-map, if required. If not required, then this signal may be set to 0. The hop map should consist of a list of addresses (terminated by -1) to which the local destination node forwards the message (to reach its ultimate destination).

- Signal 5: Function code (Used for all commands)

  This is a analog signal whose value specifies the the command function to the RTU. The following functions will be supported:

  1 Read AI-Net scan table
  2 Read AI-Net register
  3 Read AI-Net register range
  4 Read AI_Net configured table
  11 Write AI-Net scan table
  12 Write AI-Net register
  13 Write AI-Net register range
  14 Write AI-Net configured table
  21 Clear AI-Net Reset status
  30 AI-Net General Write/Read
  101 Scan ASC-551
  102 Scan ASC-555
  103 Scan ASC-556
  104 Scan ASC-558
  105 Scan ASC-571
  106 Scan ASC-574
  111 Select ASC-554 relay
  112 Reset ASC-556 accumulator
  113 Load ASC-557 output channel
  114 Select ASC-558 relay
  115 Reset ASC-558 accumulator
  116 Select ASC-574 relay
  117 Reset ASC-574 accumulator

- Signal 6: Point address/Table Number (Used various ASC-500 / AI-Net commands)

  This is a analog signal whose value specifies the initial address of an ASC-500 or AI-Net data point (register/table) for data output or input.

  For AI-Net register list operations, this is the number of an analog array that contains the list of registers.

  For AI-Net Register Range operations in contains the address of the first register in the range.

  For AI-Net Table Commands, it contains the table number.

  For 558 write relay commands, it contains the relay number 1 to 8.

  For 556 write, it contains the Reset command code, E1, D2 or C3

  For 557 write it contains the D/A ‘Command Word’ E1 -- 87

- Signal 7: Point count or Table Type/DataByte (Some AI-Net commands)

  This is a analog signal whose value specifies the count of register to read in a read or write register list, or in a read or write register range command. Also, it has a special coding for Read/Write Configured Table commands.

  In a Write Configured Table command, it refers to the contents of the ‘Table Type’ field.

In a Read Configured Command, it should be considered as a Byte encoded field, with the low byte containing the data byte, and the high byte containing the Table Type.

- Signal 8: Application Data Count (Used for AI-Net output commands).

  This is an analog signal containing the total application data length for an AI-Net output command (it is placed after the function code in the message - see ref 2) . This enables Multiblock outputs to be generated where applicable.

- Signal 9: Input/Output List Number (Various 500 series /AI-Net commands)

  This is an analog signal giving the number of a list containing signals to contain the input/output data for the given command. The contents of the input list varies according to the command type and must be matched with the associated input format .

- Signal 10: Format Number for input list (Various 500 series /AI-Net commands)

  This is an analog signal giving the number of a format list to be used to translate between the data field of an answer message and the ACCOL signals used for input. Refer to examples in Appendix A for details. (also used for some output commands)

  Note: on 'General Write/Read' commands, the Data commences at the 'FC' field. The next 2 bytes are the application count, and will be overwritten by the contents of Signal 8 above. These 2 bytes must be allowed for in the Format (e.g. by using BYT,2X to skip the fields).

- Signal 11: Reply time-out (Used for all commands)

  An analog signal whose value is the amount of time to wait for a reply message. This value is in units of seconds with a resolution or 0.1sec., to a maximum of 25.5

- Signal 12: Transmit time--out (Used for all commands)

  An analog signal whose value is the amount of time to wait for the output part of a message to complete. This value is in units of seconds with a resolution or 0.1sec., to a maximum of 25.5

- Signal 13: RTS Assert Delay (Used for all commands)

  An analog signal whose value is the amount of time to wait after asserting RTS before transmitting the first character. This value is in units of seconds with a resolution or 1 msec. This allows tuning for potential different Modem devices. A value of 0.1s has been tested with the 830 MIU.

- Signal 14: Function code tx factor (Used for all 500 series commands)

  An analog signal whose value is a multiplier of the character transmit time (or a multiplier of 5ms minimum) to wait for the clearing of the USART after the Reset Word is transmitted.A Value of 2 has been tested . (Its use is to allow the parity to change to be made on the 33xx Usart without affecting the previous character.)

- Signal 15: Last char tx factor (Used for all commands)

  An analog signal whose value is a multiplier of the character transmit time (or a multiplier of 5ms minimum) to wait for the clearing of the USART after the Last character has been transmitted. A Value of 2 has been tested . RTS is de-asserted after this time.

- Signal 16: General Write/Read I/O List (Used for AINET General Write/Read commands)

  This is an analog signal giving the number of a list containing signals to contain the input data for the given command. The contents of the input list varies according to the command type and must be matched with the associated input format . Note - the output part is handled by Signal 9.

- Signal 17: General Write/Read Format No (Used for AINET General Write/Read commands)

This is an analog signal giving the number of a format list to be used to translate between the data field of an answer message and the ACCOL signals used for input.

- Signal 18: Done List (Used for all commands)

  This is an analog signal containing a list number specifying a list which gives completion information. For 500 Series commands, only the first signal in the list , the 'Done' signal, is relevant. The contents of this list are given below.

5.6 Done List contents

- Signal 1: Done Signal

  An analog or logical signal that indicates completion of a communication request. If a logical signal is used, it will be set False when the communication request is initiated and will be set TRUE when the communication request is completed. An analog signal will be incremented. The Custom module's STATUS terminal is updated at the same time as this signal.

- Signal 2: Command Byte application status

  An analog signal that contains the contens of the Command Byte Application dependent data (bottom 4 bits).

- Signal 3: Status Byte application status

  An analog signal that contains the contens of the Status Byte Application dependent data (bits 1 to 3 shifted right 1 place).

- Signal 4: Status Byte Error Flag

  An logical signal that gives the status of the status byte error flag (set if an application level error is detected by the RTU).

- Signal 5: Status Byte PFP Flag

  An logical signal that gives the status of the status byte PFP flag .

- Signal 6: Status Byte Cold Start Flag

  An logical signal that gives the status of the status byte Cold Start flag .

- Signal 7: Status Byte Warm Start Flag

  An logical signal that gives the status of the status byte Warm Start flag .

- Signal 8: Application Received Byte Count

  An analog signal that contains the overall Application Byte Count given in a received AINET message (contained on all single message responses, and in the first block only of a Multiblock response). This can be used to check the amount of data actually returened by an RTU.

6. FORMATS FOR THE AMI PROTOCOLS

Formats are defined in the same way as logger formats. Types of formats are the usual ACCOL AIC structures but are interpreted uniquely by the this interface. Only a subset of the possible format field descriptors are valid for AMI interfaces..

The formats are used to translate between ACCOL analog/logical signals and the data field of command/answer messages.

The AMI field descriptors and their functions are as follows:

()	Parenthesis are used to group a section of the Format for repetition. Parenthesis may be nested up to five levels.
SFn	This descriptor invokes Format number n where n is any valid Format number. At the end of Format n, processing continues with the descriptor following SFn.
DA	The value of the current signal in the I/O list is used to define the number of an analog Data Array to be used. The signal's type must be analog. Array mode is set active which causes cells in the Data Array to be used by field descriptors for input and output. The first cell in the array is used first and all columns of a row are used before going to the next row. This descriptor causes an increment to the next signal in the I/O list.
DL	The value of the current signal in the I/O list is used to define the number of a logical Data Array to be used. The signal's type must be analog. Array mode is set active which causes cells in the Data Array to be used by field descriptors for input and output. The first cell in the array is used first and all columns of a row are used before going to the next row. This descriptor causes an increment to the next signal in the I/O list.
DE	Array mode is ended. Field descriptors resume using signals in the I/O list.
DC	Array mode is set active. A Data Array must have been previously defined via the DA or DL field descriptors. Field descriptors resume using cells in the data array.
BIT	Bit alignment mode is set active. The data in a message is processed in units of bits. Lower order bits of a byte or word are processed before higher order bits. If Word alignment mode was previously active, any remaining bits of the current word are used before using the next data byte. If Byte alignment mode was previously active, any remaining bits of the current byte are used before using the next data byte. It is intended that Bit alignment mode be used to access single bit logical values and subfields within a byte or word.
BYT	Byte alignment mode is set active. The data in a message is processed in units of bytes. Each field begins with the low order bit of the next byte. Values are treated as being right justified within the byte. If Word alignment mode was previously active and the high order byte of the current word was now used, the high byte is used first before using the next data byte.
WRD	Word alignment mode is set active. The data in a message is processed in units of words. Values are treated as being the combination of two bytes. Each field begins with the low order bit of the next word. Either the low order byte or the high order byte can occur first in the message. Values are right justified within the word.
LBF	Low Byte First mode is set active. Word alignment mode will treat the first of two bytes as being the low order byte of the word.
HBF	High Byte First mode is set active. Word alignment mode will treat the first of two bytes as being the high order byte of the word.
VL	This field descriptor is used for input or output of logical values. It operates on either bits, bytes, or words depending on the alignment mode. For input, the current bit, byte, or word value in the message is tested for zero. A value of zero is treated as false and a non-zero value is treated as true. The current signal in the I/O list or the current cell in the data array is set to reflect the true or false value. Analog signals or cells are set to 0.0 for false and 1.0 for true. String signals are invalid.

For output, the current signal or cell is tested for true or false. If true, a bit, byte, or word value of 1 is put in the message. If false, a bit, byte, or word value of 0 is put in the message. Analog signals or cells with values of 0.0 are treated as being false. String signals are invalid.

This descriptor causes an increment to the next signal in the I/O list or to the next cell in the data array depending on array modee being active. It also causes an increment to the next bit, byte, or word in the message depending on the alignment mode.

VSn This field descriptor is used for input or output of igned (2's complement) binary values with a field width of n bits., If Bit mode is active, the next n bits in the message are used. If Byte or Word mode is active, the field is right justified in the byte or word. If Byte mode is active and n is greater than 8, multiple bytes will be used. If Word mode is active and n is greater than 16, multiple words will be used.

The value of n may range from 2 to 32. The default value for n if not specified is; 2 for Bit mode, 8 for Byte mode, and 16 for Word mode.

For input, the current signal in the I/O list or the current cell in the data array is set to the value of this field. Logical signals or cells are set to false if the value is zero and set to true if the value is non zero. String signals are invalid.

For output, the value of the current signal or cell is in the message. Logical signal or cell values of false are equivalent to 0 and values of true are equivalent to 1. String signals are invalid. Values are rounded to the next integer value and values too large for the field are output as the largest possible field value.

This descriptor causes an increment to the next signal in the I/O list or to the next cell in the data array depending on array mode being active.

VUn This field descriptor is the same as VSn with the following exceptions. The binary value is unsigned and n may range from 1 to 32. Negative values are output as zero. The maximum value in a 32 bit field is limited to a 31 bit number for both input and output.

BCDn This field descriptor is used for input or output of Binary Coded Decimal (BCD) values with a field width of n digits. If Bit mode is active, the next n*4 bits in the messsage are used with the first digit treated as the highest order digit. If Byte or Word mode is active, the digits are right justified within the byte or word. If Byte mode is active and n is greater than 2, muultiple bytes will be used. If Word mode is active and n is greater than 2, multiple bytes will be used. If Word mode is active and n is greater than 4, multiple words will be used.

The value of n may range from 1 to 39. The default value for n if not specified is; 1 for Bit mode, 2 for Byte modek, and 4 for Word mode.

For input, the current signal in the I/O list or the current cell in the data array is set to the value of the field. Logical signals or cells are set to false if the value is zero and set to true if the value is non-zero. String signals are invalid.

For output, the value of the current signal or cell is put in the message. Logical signal or cell values of false are equivalent to 0 and values of true are equivalent to 1. String signals are invalid.

This descriptor causes an increment to the next signal in the I/O list or to the next cell in the data array depending on array mode being active.

Tn This field descriptor is used for input or output of ASCII text strings with a length of n characters. Each character is 8 bits. If Bit mode is active, the next n*8 bits in the message are used. If Byte mode is active, the next n bytes are used. If Word mode is

active, the next n/2 words are used.

The value of n may range from 1 to 64. The value of n will default to the length of the String signal's value if it is not specified. Only string signals from the I/O list are valid.

Values too large will be truncated and values too small will be padded with space characters.

For input, the current string signal in the I/O list is set to the string value of the field. Space characters are suubstituted for non printable characters in the string.

For output, the value of the current string signal is put in the message.

This descriptor causes an increment to the next signal in the I/O list.

X This field descriptor is used to skip a bit, byte, or word depending on the alignment mode. For output, a value of 0 is put in the message for the current bit, byte, or word.

This descriptor causes an increment to the next bit, byte, or word in the message depending on the alignment mode.

It is possible for field descriptors VS, VU, BCD, and T to use a partial byte or word. If Byte alignment mode is active and there are unused bits in the current byte, switching to Bit alignment mode via the BIT field descriptor will allow the unused bits to be accessed. If Word alignment mode is active and there are unused bits or bytes in the current word, switching to Bit or Byte alignment mode will allow the unused bits or bytes to be accessed. This is useful when different data types are combined into the same byte or word.

For example, a word may contain a 3 digit BCD value in the low order 12 bits and 4 logical status values in the high order 4 bits. The Format sequence WRD BCD3 BIT 4VL will relate the BCD value with a signal or array cell and each of the four status bits with its own signal or array cell.

CST1:n This field specifies conversion is to be made using 32 bit floating point format (similar to IEEE) The low byte is output/input first.

CST3:n 500 Series analog input format. 12 bit input value, sign bit and overrange bit (mapped into questionable bit)

CST4:n 557 RTU analog output format. Output value as a 3 byte field, each byte consisting of a hex value in bottom 4 bits with its complement in upper 4 bits

CST5:n 64 bit analog format. Input/Output value is an 8 byte field,LSB first, in standard 'Intel' 64 bit double precision format.NOTE: use of this format is not precise, as ACCOL signals cannot hold the full resolution. Use this for AI-Net 64 bit registers.

CST6:n 32 Bit unsigned Long integer format.. Input/Output value as a 4 byte field, without sign bit. NOTE: use of this format is not precise, as ACCOL signals cannot hold the full precision. Use this for ASC500/AI-NET registers where full precision is not required. (see notes in section 9).

7. APPENDIX A : EXAMPLES

Please refer to the Accol load file AMIALL.ACC for examples of all types of 500/700 series commands. The user should use this as the basis for all implementations.

8. APPENDIX B : REFERENCES

1. AMOCAMS/MODULAR MicroRemote ASC-500 Series Reference Manual
2. AMOCAMS/MODULAR AI-Net Protocol Definition Reference Manual

9. APPENDIX C : SPECIFIC NOTES AND LIMITATIONS

The following should be borne in mind by users of this Custom Module:

a) A command is only sent when the previous command has been answered, or has timed out. This means that a device being addressed which is not responding correctly may hold up commands being sent to other devices.

b) Some 500/700 series devices support the use of 32 bit accumulators. As Accol signals do not support values with full 32 bit resolution, these values are not directly supported by formats with full precision. The Accol programmer who requries full precision should read the high and low words into separate signals(using VU formats) and then convert or use as required.

c) 500 series analog values should be read using format CST3. This maps the polarity bit into the sign of the Accol signal, and the Overrange bit is mapped into the questionable bit. Accumulators should be read with the VU format.

d) There are no formats to input Gray Codes. These should be read if required using VU format and translated via Accol code.

e) Note 500 series reference manual gives a command code of F0 for 554 Relay Output. This is incorrect.

f) The implementation allows generation of hop-maps to connect to a remote not directly connected, however the 33xx cannot itself route messages

g) No explicit support of operation with a PAD is supported

h) No Unsolicited Report by Exception mode is supported (the Media Access Protocol is not used). Slaves must not send any messages unless in response to a Message from the 33xx.

i) The ‘Application Count’ on AINET Output messages must be known. Although this may be a little tedious for the programmer, it sames time in the Custopm Module operation, and there is no need to pre-scan the List and Formats where Multiblock transmisions are used.

j) Please refer to release README.TXT for latest information.

10. Appendix D - Troubleshooting

In the event of problems, the first point of reference should be the examples provided. Check that the usage being made is based on one of the examples provided.

Should there still be problems, then have ready the following information before contacting BBI support:

- A copy of the actual Accol load being used.
- Evidence of serial line activity (it may be necessary to use a datascope to provide details of this)
- Evidence of the status signal values being returned, and the contents of the ‘done’ list.
- Details of the cables used.

Return to Application Notes Menu

Return to the List of Manuals