User Manual for

NETWORK 3000

AMI 700 Slave

Custom Module

Document No:AMSUSMAN **Date:20-Feb-1997**

Issue: **Preliminary 1**

1. Introduction

The AM/I (AMI) 700 Slave Custom Module interface allows a Bristol Babcock 33XX series controller to communicate using the AMI 700 network protocol with the 33XX acting as a Slave device.

This document assumes some familiarity with the protocol and the structure of the devices being emulated. Familiarity with Accol II tools, data structures and language is also assumed.

This document does not give details of the message structures. Full details of the protocol are given in Reference 1, which can be obtained from BBI.

2. Overview

2.1. Operational Overview

The protocol is basically a master/slave protocol, with the master originating each transaction, and the slaves only generating replies when asked for by the master. The 33xx acts as a Slave. The definition of the protocol does support a Report by Exception(RBE) mode, with the Slaves being able to act as temporary Masters. This version of the Custom Module does NOT support RBE operation. The RTU can only respond to requests from a Master, and cannot generate unsolicited messages.

The Custom Module can, with certain restricitions, be configured to simulate more than one AMI 700 RTU (i.e. to respond to more than one RTU address). It will receive all messages generated on the link, and then try to match the Slave Address to its configuration map. If a match is found, then the message is processed, and a reply generated.

Only a selection of commands are directly implemented by the Custom Module (see below). Other commands from the Host can be handled via the 'intercept' facility, described below.

2.2. Summary of Limitations of this implementation

2.2.1. General Limitations

1. No Unsolicited messages generated by Slave. No support for 'Report by Exeception'.
2. No 'PAD' support implemented.
3. No support for Media Access Protocol (CSMA)
4. There is no PORTSTATUS Module support for this Custom Module.
5. Some AI-Net commands not directly supported, but available via 'Intecept' facility.
6. No support for 'Global Addressing' except for 'Clear Reset Status', 'Cold Start', 'Warm Start' and 'Set Time and Date', which accept an address of '0' (no reply is generated to the host). (no 'intercept processing' allowed for address of 0).
7. No support for 'Unconfigured RTU Reply' .
8. When more than one Slave address is specified, commands such as 'Clear Reset Status', 'Cold Start', 'Warm Start' and 'Set Time and Date' to a single slave will affect all slaves specified.
9. The 33xx can not forward (repeat) AI-Net messages to other nodes. It must be an 'end node'. All messages addressed to a slave address must be for that slave address.

2.2.2. Trends - Limitations

1. Accol Programmer must set up code to maintain the trend arrays used.

2. Only 4 seconds resolution available with Julian Timestamps

3. The Time Channel number in a 'Retrieve trended data by time' requests will not be used. The timestamps of the time channel configured for the requested trend block number will be used.

2.2.3. Audit / Event Logs - Limitations

1. Only 'Alarm' and 'Event' log messages are supported. 'Read Comment record' requests and 'Read Error log' requests will respond with empty messages (no records). '

2. For audit requests by time, an extra record prior to the start time will not be sent.

2.3. Accol AMI Slave Implementation Overview

2.3.1. General Description

This implementation uses the Comms Card ports configured in RS423 or RS485 Mode, to talk to the comms link used. In general, a NULL MODEM cable can be used to connect the Master to the Slave on a point-to-point basis. RTS/CTS modem controls are used. Extra hardware may be needed for Multidrop operation.

As this is a slave implementation, to some extent an 'emulation' is made of the slave RTU. Some of this is done within the Custom Module itself, whilst some is left to the Accol Programmer to do with Accol Code.

The main bulk of the processing is done by specifying a Task 0, non-executing Custom Module. Using its Custom List, the Custom Module is given a MAP of the slave being emulated. This takes the form of various Accol Signal Lists and Data Arrays. The Accol User must form this map as part of the design process. The map is read only by the Custom Module on Cold Start. The user should not change the values in the map after this, as unpredictable operation may occur.

As well as the automatic processing by the Custom Module, a facility (called 'Intercept Processing') is given to allow certain input messages to be passed back to the Accol, so that a response can be generated by the Accol. This facility provides a certain level of flexibility, in that processing of any AI-NET message can be handled, and normal Custom Module operation can be bypassed. Once the Accol has been notified of the message, it must then tell the Custom Module the response to send back to the Master, via a 'Rate Task' Custom Module call. See 'Task 2' in the example for a simple usage.

The Custom Module implementation gives powerful access to the protocol. This can mean that there is lot of information for the user to understand, but defaults are used to make basic use easier. It is highly recommended that the user uses as a base one of the example .ACC files (e.g. AMSLVEX.ACC). This gives an example implementation which can then be modified as required by the user (See EXAMPLE section at end of this document).

The following describes how each different type of the directly supported AI-Net commands and data type is handled by the Custom Module.

2.3.2. Read/Write Scan Tables (commands 00h, 21h)

A scan table is effectively just a list of AMI registers. This list of registers can be changed via a Write Configuration Table (type 4). On a cold start, a default list of registers can be specified. Note: all registers used must be present in the Register Map definition.

In the configuration of the Scan Table map, a 'default' (RO) array or 'config' (RW) array, or both must be specified. These arrays contain the list of registers to be used for a Read/Write scan table request. If the 'config' array is not specified, or is not set up (first element is zero), then the 'default' array will be used for any Read/Write scan table requests; otherwise the 'config' array will be used.

2.3.3. Read/Write Configuration Tables (commands 01h, 02h)

For all table types other than '4', Read/Writes will simply be used to read/write Accol data structures, under normal list / format control. It is then up to the Accol programmer to write code as required to handle the data.

For table type 4 (Scan table configuration), the Custom Module will read/write from the scan table 'config' array, as descibed above.

2.3.4. Read/Write Registers or Register Ranges (commands 05h, 06h, 2Ch, 2Dh)

AMI registers are mapped to either RW Data Arrays or Accol signals within Accol signal lists. The Custom Module converts the data to/from Accol internal format (32 bit Float, or Logical representation). Note: there may be some loss of precision in this conversion.

2.3.5. Set/Read Date and Time (commands 0Dh, 0Eh)

The Custom Module allows the system time (corresponding to Accol #TIME signals) to be modified. Note that if more than one slave is configured, this will affect all slaves.

2.3.6. Warm/Cold Boot (commands 0Fh, 10H)

A 'Warm Boot' command is acknowledged, and the STS bit is set in responses until reset by the Host.
A 'Cold Boot' command is acknowledged, and the STS bit is set in responses until reset by the Host. This will also clear the Scan Table configurations - the defaults will be used again.

2.3.7. Clear Reset Status (command 2Eh)

Clears the STS byte Cold/Warm boot bits.

2.3.8. Log/Audit processing (commands 31h, 3Ah, 2Bh, 3Bh, 28H, 3Ch, 43h)

The following AMI Function Codes are supported by this implementation of the AMI slave Custom Module. They request 'log' information from the RTU:

0x3A	Read Event Log Messages by Time
0x3B	Read Alarm Log Messages by Time
0x3C	Read Error Log Messages by Time
0x31	Read All Event Log Messages
0x2B	Read All Alarm Log Messages
0x28	Read All Error Log Messages
0x43	Read All Comment Records

The 'Event' Log and 'Alarm' Log requests will return data processed from the Accol Audit trail module. This can be configured to log 'events' (ie signal changes for a given list of signals), and also 'alarms' (for any alarm signals configured). There is no obvious equivalent information available for the 'Error' Log and 'Comment Records' function codes. The Custom Module will return empty data for these.

How Accol Audit Trail information is stored

The Accol Audit Trail information consists of a circular buffer of up to 4096 records (the actual number of records available is defined when memory is configured for the Accol Load). Each record can be an 'event' record or an 'alarm' record. As it is a circular buffer of a fixed size, it is possible that records may be overwritten before they have been sent to the AMI Host. This could happen in either 'wrap-mode' or 'stop on full' mode. In addition to a timestamp, the information saved in an audit trail record for each possible type of event/alarm is as follows:

Analog signal event	Signal pointer Old value (4-byte float) New value (4-byte float)
Logical signal event	Signal pointer Old value (boolean) New value (boolean)
Analog alarm	Signal pointer Value (4-byte float) Alarm type - one of: HIHI, HIGH, LOW, LOLO or Analog return to normal
Logical alarm	Signal pointer Value (boolean) Alarm type - one of: Logical alarm Logical return to normal

How this is used for AMI Log messages

The 'Event' and 'Alarm' Logs consist of a maximum of 256 entries. If more records exist in the audit buffer, then only the first 256 will be sent (and the Custom status will be set to a 'warning' value). The log data that is returned following a request for log information is in one of eight possible message formats. The Custom Module uses message format number 5 for 'event' records, and message format number 1 for 'alarm' entries. These contain the following information:

Format 5	Timestamp	4 bytes
	Format (fixed at 5)	Byte
	Error number	Integer
	Tag number	Integer
	Data register number	Integer
	Data value (before)	Variable (1-8 bytes)
	Data value (after)	Variable (1-8 bytes)
Format 1	Timestamp	4 bytes
	Format (fixed at 5)	Byte
	Error number	Integer
	Tag number	Integer
	Data register number	Integer
	Data value	Variable (1-8 bytes)

The AMI 'Data register number' to be included in the Log messages, and also the type of register data will need to be configured for any signals which require log messages. This is done via the 'Register' map, which needs to be configured for the 'Read Register' and 'Write Register' and Scan Table functions. Thus signals to be included in the Log messages will also need to be configured in the

'Register map' configuration (in a signal list, not an array). Any signals in the Audit Trail records which are not configured as AMI Registers will be ignored. The 'Register type' configured in the Register map will be used to determine the format of any data values returned. The types of data supported will be the same as those for the 'Read register' and 'Write register' functions. Registers in the register map definition could be duplicated - for example defined in a data array first, and then in a signal list in order to associate the register number with the signal for the audit logs (see example Accol load). The first definition will normally be used for any read/write value requests.

The value of the AMI 'Error number' to be included in the Log messages for each type of alarm/event is as follows:

Accol description	**AMI Error**	**AMI description**
Logical value change (event)	114	Status changed
Logical alarm	114	Status changed
Logical alarm, return to normal	114	Status changed
Analog value change (event)	176	Value is:
Analog alarm (HIHI)	173	Hi-Hi Limit Exceeded
Analog alarm (HI)	171	Hi Limit Exceeded
Analog alarm (LO)	169	Lo Limit Exceeded
Analog alarm (LOLO)	167	Lo-Lo Limit Exceeded
Analog alarm, return from HIHI	174	HI-Hi Limit return
Analog alarm, return from HI	172	Hi Limit return
Analog alarm, return from LO	170	Lo Limit return
Analog alarm, return from LOLO	168	Lo-Lo Limit return
Analog alarm, return from unknown	114	Return from unknown state

The AMI 'Tag number' included in the Log messages is not relevant. For this implementation, this is fixed at 0.

Note that the Accol audit record for an Analog alarm returning to normal does not contain information about the previous state. If the previous alarm state is known, then the specific AMI error code will be used, otherwise the 'return from unknown' error code will be used.

For the 'Read all events/alarms' functions, any relevant records (ie signals also configured as AMI register) that are in the Accol audit buffer will be returned. For the 'Read events/alarms by time' functions, then any records in the Accol audit buffer within the times specified will normally be returned. If, however, the start time of the request is the same as the end time of the previous request, then the processing of the Accol audit buffer will start from the previous end position. This could result in records being sent with timestamps not within the times requested. The processing of the Accol audit buffer assumes that the timestamps are in order - any changes of system time could produce unpredictable results.

2.3.9. Trends (commands 1Dh, 20h, 2Fh, 39h)

The following four AMI Function Codes are supported by this implementation of the AMI slave Custom Module. They request 'trend' information from the RTU:

0x1D	Read Block Trend
0x20	Freeze Block Trends
0x2F	Send Block Trend Information
0x39	Retrieve Trended Data by Time Frame

The AMI 'trend' samples will be held in Accol Read-Write Data Arrays, which are updated by the Accol program. This could be done via the Accol STORAGE module (see example). The Custom module will use these data arrays to process the above requests from the master.

As part of the Custom Module configuration information, two Accol Signal Lists/Data Arrays are used to contain definition information on the mapping of AMI Trend Data and AMI Trend Timestamps to Accol Data arrays. The first list configures each AMI 'trend block', and the second configures each AMI 'Time Channel'. Each AMI trend block must reference an AMI time channel number from the second list, and the size (number of rows) of the Data Array used for Trend data samples should be the same as that of the corresponding Time channel timestamps Data Array.

Notes*:*

- The first two rows of the Data Array containing the 'Trend samples' are reserved to keep the requested 'start' and 'end' pointers following a 'Freeze' request. These will be written to by the Custom module on receipt of a 'Freeze' request, and read by the Custom module on receipt of a 'Read request'. The Accol should update the 'freeze start' pointer if that row is about to be overwritten (see example). The rest of the column of the array should be used as a circular buffer to hold the trend samples.

- A pointer for the 'last row written' should be kept by the Accol in the first row of each 'Timestamp' column. Also, a flag should be kept by the Accol in the second row of the Timestamp column to indicate whether or not the Trend data has 'wrapped' (ie the first trend sample has been overwritten). Thus the row size of the Data Array should be at least two more than the number of samples required. The rest of the column of the array should be used as a circular buffer to hold the timestamps. The maximum row number used for the timestamps should be the same as that configured for the Time channel.

- Only Analog Trend data arrays will be supported. If Booleans are to be stored, then this could be done using an Analog array (0 = 'FALSE', any other value = 'TRUE')

- The number of samples saved should be at least one more than the 'maximum number of samples' configured for any of the trend blocks.

- The Timestamps should be written into the specified data array by the Accol program in 'Julian' format when the Trend data is saved. Julian format is described in the Accol manual - System signal #TIME.000 contains the current date/time in 'Julian' format. The resolution is every 4 seconds.

- The format of the trend sample data returned (following a 'Read Block Trend' or 'Retrieve Trended Data by Time' request) will be determined by the 'register type' of the register number (as specified the register map definition). If the register number is not in one of the register maps, then the register type will default to 'FLOAT'.

Custom Module implementation of Function Code requests:

- Function 0x20 (Freeze Block Trends):
  'Freezes' the specified trend blocks - expect to then get a 'Read' request for each block.

  On receipt of a 'freeze' request, the current row number, and the 'maximum number of samples' configured will be used to determine the freeze 'start' and freeze 'end' row required. These values will be saved into the first two rows of the 'Trend Block' data array. If it is likely that the 'freeze start' row is overwritten before the 'Read request' is sent, then larger size arrays could be used (for both the Time Channel data, and the Trend data). If, for example, the maximum number of samples is 1024, then the array could be size 1030. Thus 'frozen' data would only be overwritten if there were 5 or 6 more samples in the period between the 'freeze' and the 'read'. The Accol program should adjust the 'freeze' start pointer if data is due to be overwritten - thus ensuring that misleading data will not be returned (although fewer than expected samples will be returned).

- Function 0x1D (Read Block Trends):
  Returns sampled trend data for the trend block specified in the request.

  Returns data from the Trend Data array specified for the requested Trend block number, starting at the 'freeze start' row, and finishing with the 'freeze end' row. The Timestamp of the 'end' row is obtained from the Time Channel data array (for the corresponding row), and returned in the message. If the 'freeze start' and the 'freeze end' rows are both zero, then the maximum number of samples will be returned, finishing with the latest sample. If either the 'freeze start' or 'freeze end' rows are invalid, then no samples will be returned. It is therefore up to the Accol program to determine what should happen if 'frozen' trend samples start being overwritten.

- Function 0x2F (Send Block Trend Information):
  Returns fixed information about the Trend Block requested (or all the Trend blocks configured).

  If a zero byte count is used in the request, then information for all trend blocks configured will be returned. They will be returned in the same order that they are configured. Otherwise information for the trend block requested is returned. The Register number being trended, the Sample data type and the Trend block max. size are obtained from the Trend Block configuration signal list for the requested Trend block. The sample frequency is obtained from the corresponding Time Channel signal list.

- Function 0x39 (Retrieve Trended data by time)
  Returns trend data for a specified time period.

  It is assumed that the 'Trended Data Channel' requested specifiies either a Trend block number, or a Trend block number +25000. The Trended Time Channel requested will be ignored. The relevant Time channel (as specified in the Trend Block configuration) will be searched for the requested times, and the corresponding data (and timestamps if the 'Time Flag' is set) will be returned.

2.3.10.Other commands / Intercept facility

None of the other commands are automatically handled by the Custom Module, but they can be processed using the 'Intercept' facility. Those commands which the Accol User wishes to process from the Accol load can be given an entry in the 'Intercept Configuration Table'. For each command given, an Accol List and Format number is given where the contents of the message received are put. When the Custom Module receives one of these commands, it sets the command number in the signal

INTERCEPT_FUNCTION, sets the Accol format Status in the signal INTERCEPT_STATUS, sets signal INTERCEPT_COUNT to the length of the input message (not including the header), and sets the INTERCEPT_NOTIFY signal to tell the Accol that a message is recevied. The Accol can then look at the input data, and process as required, before generating a REPLY via a Rate-Task Custom Module call (compulsory!). The Custom Module will then continue normal operation. Refer to the example 'Task 2'.

This facility can be used to process any commands (including replacing the 'automatic' Custom Module processing if required). If the Accol doen't issue a reply wiithin a certain period (controlled by P1/P2 parameters, see below), then the Custom Module will continue normal operation. Whilst it is waiting for a respponse from the Accol, no other messages will be processed.

2.3.11. Basic Data Types

As part of the Custom Module configuration information, an Accol Signal List (or Data Array) is used to contain definition information on the mapping of AMI Registers to Accol signals. This list is formed from a consecutive number of entries, each of 5 signals, giving information for a 'block' of AMI registers of a particular type.

List of Register data types supported:

1	Digital Input (boolean)	(boolean)
2	Digital Output (boolean)	(boolean)
3	Analog Input	(2-byte unsigned integer)
4	Analog Output	(2-byte unsigned integer)
5	Accumulator	(4-byte unsigned integer)
6	Integer	(2-byte signed integer)
7	Float	(4-byte 'short' float)
8	Booleans	(boolean)
9	Unsigned 16 bit integer	
10	Unsigned 24 bit integer	
11	Unsigned 32 bit integer	(Accol can't store full precision)
12	Signed 16 bit integer	
13	Signed 24 bit integer	
14	Signed 32 bit integer	(Accol can't store full precision)
15	Short Real (4 bytes)	
16	Long Real (8 bytes)	(Accol can't store full precision)

The first 8 Data Types are allocated the same values as the 'Data Type' list given in the AI-Net Manual in the description of the 'Read Block Trend' function. The remainder are included for completeness.

Note that all Accol values are stored in 32 Bit IEEE Floating point form. Thus: 64 bit floats and 32 bit integers cannot be stored with full precision.

3. Using the Custom Module

3.1. Configuration Overview

Before an ACCOL load can be fully implemented, the following steps may need to be performed.

1. The physical configuration of the network may need to be designed or assessed. Extra hardware may be required for Multidrop operation.
2. The Network configuration and associated limitations needs to be assessed, particularly with reference to the required timing. Also, consideration must be made on the loading of the 33xx RTU. It may not be possible to support a large number of ports all at high speed. It may be necessary to perform 'benchmark' tests on the 33xx RTU's with a simulation of the target environment.

3. Information concerning all the communicating devices needs to be obtained (e.g. addresses, capabilities). If installing onto an existing network, the overall capacity of the network must be considered.

3.2. Accol Configuration overview

The following steps are required to configure the ACCOL load in order to make the 33XX controller act as an AMI Slave device.

1. Define a Custom Port. This is described in the section *'Defining the Custom Port'*.
2. Define the Task 0 Custom Modules /Signal Lists, known as the AMI Slave Custom Lists. Also define the lists and arrays forming the Slave Map. This is described in the section *'Defining the Task 0 Custom Lists'*.
3. Define any Rate-Task Custom Modules/Lists required. This is described in the section *'Defining the Rate Task / Reply Custom Lists'*.
4. Set up the required Output/Input lists and formats for the rate task Custom Modules, using formats described in the section '*Accol Data Format Codes for AMI Slave'*..
5. Write the supporting Accol code required, including mapping the signals to 33xx hardware and Accol modules as appropriate. Write any Accol code necessary for trend processing. Write Accol code for any 'intercept' processing.

3.3. Defining the Custom Port

The Custom Port is associated with a specific link. The following is a list of the parameter field values for a Custom port when configured AMI Slave mode.

- MODE: Set this field to 25 to indicate AMI Slave mode.
- BAUD: Set this field to the baud rate required (values up to 19200 supported)
- CHARACTER LENGTH: Set this field to 8 bits 'BIT_8'.
- STOP BITS: Set this field to 'SBIT_1' for 1 stop bit, 'SBIT_2' for 2 stop bits
- PARITY: Set this field to 'PARITY_O','PARITY_E'',PARITY_N',for odd, even or no parity as appropriate
- P1: This is used in association with the 'Intercept' facility. It gives a poll period in milliseconds that that the Custom Module looks for a response to an Intercept from the Accol. If 0 is specified, then a default of 20ms is used. If in doubt, use the default!
- P2:. This is used in association with the 'Intercept' facility. It gives the time allowed (in milliseconds) for the Accol to generate a response to the Intercept. If 0 is specified, then a default of 3000ms (3 seconds) is used. If in doubt, use the default!

3.4. Custom Memory Allocation

For this Custom Module, an amount of CUSTOM Memory needs to be allocated within the *MEMORY section of the Accol load for each port used. This is used to hold information obtained from the Slave Map, and also io buffers. This will depend on the Slave Map, but an allocation of 4K bytes per port should be ample. A Custom Status error is generated if there is not enough memory present, in which case, the Accol User should adjust the value until no memory errors are reported.

The allocation must take into account any other users of this memory.

For 186/386 'Real' Mode, use (for example)

*MEMORY
CUSTOM_SIZE 4000 per port

For 386 ‘Protected’ Mode, use (for example)

*MEMORY
GLOBAL_STORE 4 (1k pages) per port

3.5. Defining the Custom Modules

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

- MODE: A value of 25 indicates AMI Slave 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 Slave Custom list..
- 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. See section ‘Error and Status Codes’.

3.6. Defining the Task 0 Custom List

Task 0 Custom List Contents

Terminal name	Signal Type	Description
Signal 1 PORT	A	Analog signal with value defining communications port to use for this interface.
Signal 2 FUNCTION	A	Analog signal of value 1 = Process Slave Map (this Custom List). No other values allowed in Task 0 signal list.
Signal 3 DONE	A / L	Updated whenever the Custom Status signal is written to. A logical signal is turned ON on, whilst an analog signal is incremented.
Signal 4 RESPONSE TIMEOUT	A	Time allowed to receive an expected message, only applicable to Multiblock replies. Range allowed 0.1s to 25.5s (resolution of 0.1s) Default is 1sec (used if set to -1)
Signal 5 TRANSMIT TIMEOUT	A	Time allowed to transmit a message, from start of data timed to end of transmission of last character of transmitted message. (e.g. if can't transmit due to CTS not being set). Range allowed 0.1s to 25.5s (resolution of 0.1s) Default is 1sec (used if set to -1)
Signal 6 RTS_ASSERT_DELAY	A	Optional delay after asserting RTS on reply transmission, before transmitting first character. Range 0 to 10s (resolution of 4ms). Default value of 0. (used if set to -1).
Signal 7 MESS_CLEAR_DELAY	A	Optional delay after de-asserting RTS at end of reply transmission. This takes the form of a number of character times for which to delay before de-asserting RTS. Range 0 to 255*character time (resolution of 4 ms). Default value of 0. (used if set to -1).
Signal 8 AMS_STATS_ARRAY	A	The number of a RW Data Array which is used to contain diagnostic information. This must be a RW Data of at least 35 columns. The contents are described below. A value of 0 means facility is not used. The row size could be just 1 or 2 if only operational statistics are required, or 10 if the full set of statistics is required.
Signal 9 AUDIT_LIST_NUMBER	A	This is the list specified in the AUDIT TRAIL module, if used. It is used by the Audit module to log value changes. If signals in this list match any mapped to AMI registers (via the register map), then AMI event log reports will be generated for these signals. See general description above. If 0, then no event reports will be generated (though alarm reports could be).
Signal 10	L	This is a logical signal set by the Custom Module when a Intercepted command is received, to indicate that the Accol

INTERCEPT_NOTIFY		must process the Intercept message. After the Accol has finished, it must set this to OFF, then issue an Intercept Response to allow the Custom Module to send a response.
Signal 11 INTERCEPT_RTU_ADDRESS	A	This is an analog signal which the Custom Module sets to the command code of a command just received, for 'intercept' processing.
Signal 12 INTERCEPT_FUNCTION	A	This is an analog signal which the Custom Module sets to the command code of a command just received, for 'intercept' processing.
Signal 13 INTERCEPT_STATUS	A	This is an analog signal which the Custom Module sets to the format status (not the status in the message received) resulting from the interpretation of the input message using the list/format given in the intercept table for this command. For list of status codes, see below
Signal 14 INTERCEPT_COUNT	A	This is an analog signal which the Custom Module sets to the length (minus the header) of a command message just received, for 'intercept' processing.
		The remainder of the list is taken up with Slave Map Information. As more than 1 Slave can be supported, a map is required for each separate Slave. Each group of 7 signals following applies to a single slave address.
Signal 15 RTU_ADDRESS	A	This is the address of a Slave.
Signal 16 SCAN_TABLE_MAP_LIST_NO	A	This is a Analog Signal giving the number of a signal list or data array which contains a description of the scan tables for this slave. The contents are described below. If a positive value, then it refers to a Signal List. If a negative value then the absolute value refers to an Analog Data Array. A value of 0 means facility is not used. (All commands then must be processed using 'Intercept' facility).
Signal 17 CONF_TABLE_MAP_LIST_NO	L	This is a Analog Signal giving the number of a signal list or data array which contains a description of the configuration tables for this slave. The contents are described below. If a positive value, then it refers to a Signal List. If a negative value then the absolute value refers to an Analog Data Array. A value of 0 means facility is not used. (All commands then must be processed using 'Intercept' facility)
Signal 18 REG_MAP_LIST_NO	A	This is a Analog Signal giving the number of a signal list or data array which contains a description of the registers for this slave. The contents are described below. If a positive value, then it refers to a Signal List. If a negative value then the absolute value refers to an Analog Data Array. A value of 0 means facility is not used.
Signal 19 INTERCEPT_MAP_LIST_NO	A	This is a Analog Signal giving the number of a signal list or data array which contains a description of the commands to be processed by the 'Intercept' facitily for this slave. The contents are described below. If a positive value, then it refers to a Signal List. If a negative value then the absolute value refers to an Analog Data Array. A value of 0 means

		facility is not used. (All commands then must be processed using 'Intercept' facility)
Signal 20 TREND_MAP_LIST_NO	L	This is a Analog Signal giving the number of a signal list or data array which contains a description of the configuration tables for this slave. The contents are described below. If a positive value, then it refers to a Signal List. If a negative value then the absolute value refers to an Analog Data Array. A value of 0 means facility is not used. (All commands then must be processed using 'Intercept' facility)
Signal 21 TIME_MAP_LIST_NO	A	This is a Analog Signal giving the number of a signal list or data array which contains a description of the configuration tables for this slave. The contents are described below. If a positive value, then it refers to a Signal List. If a negative value then the absolute value refers to an Analog Data Array. A value of 0 means facility is not used. (All commands then must be processed using 'Intercept' facility)

3.7. Defining the Rate Task (Reply) Custom List

NOTE: This is only required when replying to Intercept Commands passed back via the Custom Module (i.e. Binary Control, Analog Control and Other Intercepts). A reply MUST be generated before any other commands are processed by the Custom Module (even if no response is to be generated to the command, in which case Custom Module should be called with FUNCTION= 3).

Intercept Reply Custom List Contents

Terminal name	Signal Type	Description
Signal 1 PORT	A	Analog value defining communications port to use for this command.
Signal 2 FUNCTION	A	2 = Send a Reply according to the list / format below. 3 = Continue operation WITHOUT sending a reply (e.g. if the Accol decided that no
Signal 3 DONE	A / L	Updated when the Custom Task has completed this request. A logical signal is turned ON on, whilst an analog signal is incremented.
Signal 4 RP_STATUS	A	Set this up to contain the contents of the Status byte to be transmitted to the Master.
Signal 5 RP_DATA_COUNT	A	The total number of characters to be transmitted in the data portion of the message (not including the header, which is put on automatically by the Custom Module)
Signal 6 RP_OUT_LIST	A	The number of the Output list to contain the data to be output, interpreted by the format below
Signal 7 RP_OUT_FORMAT	A	The number of the Format to control the data output.

3.8. Contents of the Scan Table Map

This is a 5 column x n row analog array or a signal list (size must be multiple of 5). Each row contains information relating to the format of a particular Scan Table (for this Slave). The list can be finished with a '-1' in the first column. The format of each row is as follows:

Column number	Contents
1	Table number.
2	Default Array. This is the number of an analog array (probably a Read-Only array) which contains a list of registers to form this Scan Table. It is only used if no Scan Table Configuration has been received from the host for this table (e.g. on cold start).
3	Default Array Column. This is the column in the Default Array containing the default register map. (Its use allows all maps to be stored in a single analog array).
4	Configured Array. This is the number of a RW analog array. When a Write Configured Table command for this Scan Table is received from the host, this array is used to store the register map received. Subsequent Read/Write Scan table commands will then use this register map.
5	Configured Array Column. This is the column in the Configured Array to contain the register map. (Its use allows all maps to be stored in a single analog array).

3.9. Contents of the Configured Table Map

This is a 5 column x n row analog array or a signal list (size must be multiple of 5). Each row contains information relating to the processing of a 'Read/Write Configured Table' (for this Slave). The list can be finished with a '-1' in the first column.

Note: This is not used for Write Configured Table commands other than for configuring Scan Tables (see above). No special processing is done by the Custom Module for these - the contents of the received message are just decoded using the list/format The format of each row is as follows:

Column number	Contents
1	Table number.
2	Table type. Type 4 (Scan Table) not relevant here.
3	Size in bytes of Table.
4	List number, used to control the translation of the message contents (for Read/Write Table commands)
5	Format number, used to control the translation of the message contents (for Read/Write Table commands)

It is up to the Accol load to process this data as required.

3.10. Contents of the Register Map

This is a 5 column x n row analog array or a signal list (size must be multiple of 5). Each row contains information relating to the mapping of a range of AMI registers to Accol signals or Data Arrays (for this Slave). The list can be finished with a '-1' in the first column. The Custom Module uses these maps in processing Scan Tables, and Read/Write register commands.

Column/ element number	Contents
1	Start Register Number. An AMI register number corresponding to the first of a range of registers.
2	Number of Registers in this range.
3	Register Type. See list of register types above.
4	Array or List number. This is gives the number of a signal list or data array which contains the registers. If a positive value, then it refers to a Signal List. If a negative value then the absolute value refers to an Analog Data Array.
5	Position or Column. If an Array is specified in column 4, then this refers to the Column number in the specified array to be used. If a list is specified in column 4, then this refers to the start element number in the list to contain the first of the registers in this map (this allows a single list to contain a map of non-consecutive registers, or registers of different types.)

3.11. Defining the Intercept Configuration Array

This is a 3 column x n row (as required) analog array or a signal list (size must be a multiple of 3). Each row contains a record which defines a Command Code for a message which may be received, and an Accol List/Format Number used to pass back the contents of the message.). The list can be finished with a '-1' in the first column. The format of each row is as follows:

Column number	Contents
1	Command Number to be processed via intercept facility
2	Accol List Number to be used for input message
3	Accol Format Number to be used for input message

3.12. Contents of the Trend Map

This is a 6 column x n row analog array or a signal list (size must be multiple of 6). Each row contains information relating to the mapping of an AMI Trend Block range of AMI registers to Accol signals or Data Arrays (for this Slave). The list can be finished with a '-1' in the first column. The Custom Module uses these maps in processing Trend commands

Column/ element number	Contents

1	Trend Number. An AMI trend block number (0 to 255).
2	Registers number to be trended. This is used for a 'Send Block Trend Information' request, and also to determine the type of data returned.
3	Maximum number of samples that are returned in either a Read Block Trend request, or a Retrieve Trended Data by Time Frame Request..
4	Sample Array number. This is the number of a RW Analog Array containing the sample values.... (e.g. this could be column 2 ...of the same array as that used to contain the timestamps). See example. Note: the first two rows of this array are used for control information - see above.
5	Column of above array used.
6	Time Channel Number - corresponds to the time map (see below).

3.13. Contents of the Time Map

This is a 5 column x n row analog array or a signal list (size must be multiple of 5). Each row contains information relating to the mapping of a range of AMI registers to Accol signals or Data Arrays (for this Slave).). The list can be finished with a '-1' in the first column. The Custom Module uses these maps in processing Scan Tables, and Read/Write register commands

Column/ element number	**Contents**
1	Time Channel number (0 to 255)
2	Sample frequency. (not directly used to generate samples, as that is contolled by the Accol. This is used only in the response to a Send Block Trend Information request.)
3	Time Channel Array number. The number of the RW analog array containing the timestamps. (e.g. this could be column 1 of the same array as that used to contain the data).
4	Column of above array used.
5	Max Row. (The maximum number of rows in the array that are used before the wrap around, See example. Note: the first two rows of this array are used for control information - see above.)

3.14. Contents of the Stats Array

Row 1: General stats (initialisation and in operation)

Column number	**Contents**
1	Response timeout configured (set on initialisation only)
2	Transmit timeout configured (set on initialisation only)
3	RTS assert delay value configured (set on initialisation only)
4	Message Clear delay value configured (set on initialisation only)

7	Function code of last request processed
8	Custom status of last request processed
9	Slave address of last request processed
10-15	Diagnostics (test purposes only)
18	Number of messages received by driver
19	Number of 'global address' (slave address 0) 'Set date/time' requests processed
20	Number of Accol module calls (replies to intercepts)
21	Number of 'global address' (slave address 0) 'Warm boot' requests processed
22	Number of 'global address' (slave address 0) 'Cold boot' requests processed
23	Number of 'global address' (slave address 0) 'Clear reset' requests processed

Row 2: For the first configured slave (in operation only)

7	Function code of last request processed for this slave
8	Custom status of last request processed for this slave
10	Number of 'Read scan table' requests processed
11	Number of 'Write scan table' requests processed
12	Number of 'Read config table' requests processed
13	Number of 'Write config table' requests processed
14	Number of 'Read register' requests processed
15	Number of 'Write register' requests processed
16	Number of 'Read register range' requests processed
17	Number of 'Write register range' requests processed
18	Number of 'Intercept' requests processed
19	Number of 'Set date/time' requests processed
20	Number of 'Read date/time' requests processed
21	Number of 'Warm boot' requests processed
22	Number of 'Cold boot' requests processed
23	Number of 'Clear reset' requests processed
24	Number of 'Event log' requests processed
25	Number of 'Alarm log' requests processed
26	Number of 'Error log' requests processed
27	Number of 'Comment record' requests processed
28	Number of 'Read block trend' requests processed
29	Number of 'Freeze block trend' requests processed
30	Number of 'Send trend information' requests processed
31	Number of 'Read trend by time' requests processed

Row 3: Configuration data for each slave address configured (initialisation only)

1	Slave address
2	Number AMI audit signals (alarm and event) configured
3	Number of time channels configured
4	Number of scan tables configured
5	Number of config tables configured
6	Number of Register maps configured
7	Number of Intercept functions configured
8	Number of Trends configured
	... repeated for all slaves configured (if enough columns)

Row 4: Configuration data for each scan table for the first slave (initialisation only)

1	Scan table number

2	Scan table default array number
3	Scan table default array column number
4	Scan table configuration array number
5	Scan table configuration array column number
	... repeated for all scan tables configured (if enough columns)

Row 5: Configuration data for each config table for the first slave (initialisation only)

1	Config table number
2	Config table type
3	Config table size (in bytes)
4	Config table Accol list number
5	Config table Accol format number
	... repeated for all config tables configured (if enough columns)

Row 6: Configuration data for each register map for the first slave (initialisation only)

1	Register start number
2	Register end number
3	Register type
4	Register Accol list/array number (array if -ve)
5	Register position in list/ column number of array
	... repeated for all register maps configured (if enough columns)

Row 7: Configuration data for each intercept for the first slave (initialisation only)

1	Intercept function code
2	Intercept Accol list number
3	Intercept Accol format number
	... repeated for all intercept function configured (if enough columns)

Row 8: Configuration data for each trend block for the first slave (initialisation only)

1	Trend block number
2	Trend block number of register trended
3	Trend block max. number of samples
4	Trend block Accol array number
5	Trend block Accol array column number
6	Trend block index to time channel (n.b. this is not the same as the time channel number)
7	Trend block register type
	... repeated for all trend blocks configured (if enough columns)

Row 9: Configuration data for each time channel for the first slave (initialisation only)

1	Time channel number
2	Time channel sample frequency
3	Time channel Accol array number
4	Time channel Accol array column number
5	Time channel max. row in Accol array used for trend data
	... repeated for all trend blocks configured (if enough columns)

Row 10: Configuration data for each audit signal for the first slave (initialisation only)

1	AMI Register number of signal configured for audit (alarm or event)
2	AMI Register type
	... repeated for all audit signals configured (if enough columns)

3.15. Accol Data Format Codes for AMI Slave

() 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.

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 AI-NET registers where full precision is not required.

3.16. Error and Status Codes

Note: Some errors (e.g. memory and Initialisation errors) mean that the Custom module operation is disabled.

Error/Status messages

0	communication completed successfully
1	communication requested, waiting to send
2	command message sent, waiting for reply
4	audit request warning - more than 256 audit records to send (max of 256 sent).
5	read trend warning - Accol array 'start freeze' and 'end freeze' pointers invalid
7	register value was out of range (read register or read range function)
10	Slave 'Task 0 ' configuration accepted.
11	Valid intercept passed to Accol
12	Valid broadcast message (address 0) received
-2	invalid AMI Slave Custom List specified
-3	invalid Port specified ('Reply' Custom List)
-4	invalid Function Type code specified ('Task 0' or 'Reply' Custom List)
-5	invalid reply STS low nibble specified
-6	invalid reply data count specified
-7	invalid I/O List specified
-8	invalid Format number specified
-16	No or insufficent expanded memory allocated by ACCOL load
-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
-29	timed out waiting for response
-30	invalid msg framing characters received
-31	transmit timed out waiting for CTS
-32	unexpected I/O failure
-33	all clear time not enough -USART not clear
-35	Broadcast (address 0) with unsupported function code received
-36	Invalid Header in multiblock from master
-37	Invalid BLC in block from master
-39	Invalid BLC in Ack from Master
-40	extra block received after error in rx multiblock
-41	Not the final destination in the hop path
-42	No matching slave address found
-43	intercept reply from Accol, but none expected
-44	function code in received request not supported
-45	internal error
-46	last block of multiblock not BLC=0
-47	size of received message invalid
-48	read register request with multiblock receive - not supported
-49	invalid data in set date/time request
-50	read scan request - scan table not configured
-51	write scan request - scan table not configured
-52	read config request (type 4) - scan table not configured
-53	write config request (type 4) - scan table not configured
-54	read scan request - scan table config array not set up
-55	write scan request - scan table config array not set up
-56	read config request (type 4) - scan config array not set

-57	write config request (type 4) - no scan config array
-58	read config request - config table not configured
-59	write config request - config table not configured
-60	write config request (type 4) - data not multiple of 3
-61	write config request (type 4) - scan conf array too small
-62	write scan request - incorrect size (bytes) in request
-70	read reg. request - register not found
-71	read reg. range request - register not found
-72	read scan request - register not found
-73	write reg. request - register not found
-74	write reg. range request - register not found
-75	write scan request - register not found
-76	write config request (type 4) - register not found
-77	read reg. range request - different size of registers
-78	write reg. range request - different size of registers
-79	read reg. range request - none requested
-80	write reg. range request - none requested
-81	write reg. range request - invalid size of data
-82	trend time request - if 'current' pointer is invalid
-83	trend info request - invalid trend block number
-84	trend info request - trend block not configured
-85	trend time request - invalid data channel number
-86	trend read request - trend block not configured
-87	trend time request - trend block not configured
-88	trend time request - sample overwritten while processing
-89	trend time request - start is after latest sample time
-90	audit log request - inconsistent timestamps in buffer
-101	task 0 initialisation - invalid Slave (general error)
-102	task 0 initialisation - unable to allocate sufficient local Custom memory
-103	task 0 initialisation - unable to allocate sufficient Custom memory
-111	task 0 initialisation - invalid Response Timeout value specified
-112	task 0 initialisation - invalid TX Timeout value specified
-113	task 0 initialisation - invalid RTS Time value specified
-114	task 0 initialisation - invalid MESS Factor value specified
-117	task 0 initialisation - invalid Stats Array number specified
-118	task 0 initialisation - Stats Array size (columns) is too small
-119	task 0 initialisation - invalid audit list specified
-120	task 0 initialisation - invalid intercept notify signal (must be logical)
-121	task 0 initialisation - invalid intercept RTU address signal (must be analog)
-122	task 0 initialisation - invalid intercept function signal (must be analog)
-123	task 0 initialisation - invalid intercept status signal (must be analog)
-124	task 0 initialisation - invalid intercept byte count signal (must be analog)
-125	task 0 initialisation - invalid size of scan table map
-126	task 0 initialisation - invalid size of config table map
-127	task 0 initialisation - invalid size of register table map
-128	task 0 initialisation - invalid size of intercept table map
-129	task 0 initialisation - invalid size of trend table map
-130	task 0 initialisation - invalid size of time channel table map
-131	task 0 initialisation - invalid RTU slave address
-132	task 0 initialisation - invalid scan table map list/array
-133	task 0 initialisation - invalid config table map list/array
-134	task 0 initialisation - invalid register table map list/array
-135	task 0 initialisation - invalid intercept table map list/array
-136	task 0 initialisation - invalid trend table map list/array
-137	task 0 initialisation - invalid time channel table map list/array
-141	task 0 initialisation - invalid scan table number

-142 task 0 initialisation - invalid scan table 'default' array
-143 task 0 initialisation - invalid scan table 'default' array column number
-144 task 0 initialisation - invalid scan table 'config' array
-145 task 0 initialisation - invalid scan table 'config' array column number
-151 task 0 initialisation - invalid config table number
-152 task 0 initialisation - invalid config table type
-153 task 0 initialisation - invalid config table size
-154 task 0 initialisation - invalid config table list
-155 task 0 initialisation - invalid config table format
-161 task 0 initialisation - invalid start register number
-162 task 0 initialisation - invalid number of registers
-163 task 0 initialisation - invalid register type
-164 task 0 initialisation - invalid register list/array
-165 task 0 initialisation - invalid register position in list/column in array
-166 task 0 initialisation - invalid register list/array size
-171 task 0 initialisation - invalid intercept function code number
-172 task 0 initialisation - invalid intercept list number
-173 task 0 initialisation - invalid intercept format number
-181 task 0 initialisation - invalid trend block number
-182 task 0 initialisation - invalid trend register number
-183 task 0 initialisation - invalid trend max. number of samples
-184 task 0 initialisation - invalid trend array number
-185 task 0 initialisation - invalid trend array column number
-186 task 0 initialisation - invalid time channel number
-187 task 0 initialisation - time channel number not configured
-188 task 0 initialisation - time channel array/trend array incompatible sizes
-191 task 0 initialisation - invalid time channel number
-192 task 0 initialisation - invalid time channel sample frequency
-193 task 0 initialisation - invalid time channel array number
-194 task 0 initialisation - invalid time channel array column number
-195 task 0 initialisation - invalid time channel max. row number

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

201 An input signal is control inhibited (formats only)
202 Attempt to store signal into a constant (formats only)
203 An input string signal value was truncated (formats only)
204 Attempt to store into a Read Only data array
216 Format error - attempt to go beyond end of buffer on read
217 Format error - list/format specifies more than totalcount
218 Format error - overflow in write regs data array
-201 Format error - unsupported field descriptor
-202 Format error - attempt to use signal beyond end of list
-203 Format error - too many levels of parenthesis
-204 Format error - unmatched right parenthesis
-205 Format error - sub format number does not exist
-206 Format error - too many levels of sub formats
-207 Format error - invalid data array number selected
-208 Format error - data array has not been defined
-209 Format error - attempt to use cell beyond end of array
-210 Format error - signal or cell type must be analog
-211 Format error - signal type must be string
-212 Format error - sig or cell type must be analog or logical
-213 Format error - BCD input value is invalid

-214	Format error - bad analog value for BCD output
-215	Format error - unexpected input signal store failure
-216	Format error - attempt to go beyond end of buffer
-217	Format error - invalid floating point value
-219	Format error - Floating point field overflow on output
-221	Format error - field length too big, or decimal point position error
-222	Format error - rep count not 1

4. Problem solving

4.1. Use of Serial Line Analyser

If errors are occuring on the link, or there is diffculty in establishing communications, then it may be desirable to monitor the communications activity by using a Serial Line Analyser. When installing onto a network (e.g. replacing an AMI Slave), it may be a good idea to use the Serial Line Analyser to make a recording of the dialog going between the host and the old Slave. This may be required by BBI staff in accessing any problems.

4.2. Stats array

If problems occur, investigating the stats array may give more information. BBI may ask for the contents of this when assessing problems.

5. Example

The Accol Load AMSLVEX.ACC (available from BBI) contains a basic configuration, where Data Arrays are used rather than Signal Lists (where selectable). It is suitable for use as a basis for an actual implementation. This load has a single slave at address 1.

Basic Information

The 33xx RTU is connected in this case using Port 2. This may be changed as required (the PORT signal within the Custom Lists should also be changed to reflect the Port used.

Task 0

This contains the AMISLV Custom Module to define the slave configuration. It also contains an example AUDIT module, which can be adapted as required. Signals in the load with basename AUDIT are associated with the AUDIT module.

Task 2

This gives an example of of 'Intercept Processing'. The Accol code waits for the Intercept Notify signal to be set, indicating that a message is ready to be processed. This example performs simple processing on a command message of type 50 (SBO Normal Select), before generating the response. The processing required here can be extended as required.

Task 3

This is an example of how trend data can be set up . It uses this STORAGE module to trend signals. Delete this task if trends are not required, or modifiy as appropriate. The Accol code increments the three signals being trended. It then updates the 'current row' pointer to the next row (wrapping if required). Before the row is updated, the 'freeze start' pointer is checked. If the 'freeze start' position is about to be overwritten then either the 'freeze start' position is moved by one row, or, if the 'freeze start' position is equal to the 'freeze end' position, then both positions are set to 1 (otherwise the 'start' position would overtake the 'end'). If both positions are set to 1 (invalid row pointers), then no samples will be sent by the slave in response to a 'Read Block Trend' request. If the positions are reset to 0, and no further 'freeze' requests are made, then the maximum number of samples will be sent in response to a 'Read Block Trend' request.

Task 4

This task is purely for test purposes, and initialises the 'registers' to known values. Delete this task once no longer required.

Task 7

This task is purely for test purposes. It is used to test the audit processing by creating alarms/events in the audit buffer. Delete this task once no longer required.

Scan Table map (data array 151)

Four scan tables (numbers 1,2,6 and 7) are configured, with default arrays for tables 1,2 and 6 (contained in arrays 109, 107 and 107 respectively). Tables 1,2 and 7 can be configured via a 'Write configuration table' function.

Configuration Table maps (data array 161)

Two configuration tables (numbers 1 and 2, both type 1) are configured.

Register Map (data array 171)

The slave configuration contains a register map as follows:

Start register	No. of registers	Register type	Values list/array	Values pos. in list/ column number
1	1999	1 (digin)	logical array 171	1
2000	2000	3 (anin)	analog array 172	1
4000	1000	5 (accumulator)	analog array 173	1
5000	2000	6 (integer)	analog array 174	1
9000	2000	7 (float)	analog array 175	1
11000	1000	2 (digout)	logical array 176	1
2000	2	3 (anin)	signal list 130	0
1	4	1 (digin)	signal list 130	2
2010	2	3 (anin)	signal list 31	1
10	3	1 (digin)	signal list 31	3

Note that the last four entries are to associate Accol signals with AMI registers for the Audit processing. They duplicate earlier register numbers; any register values (for read/write register and scan table requests) will use the values from the data arrays, not the signal lists.

Intercept Map (data array 181)

One intercept function code (0x32, SBO Normal Select) is configured.

Trend Map (data array 191)

Three trend blocks (numbers 201, 202, 203) are configured, all using the same Time channel number and the same Accol Data array.

Time channel Map (data array 192)

One Time channel (number 1) is configured, using the first column of the Trend samples Data array. The maximum row number used is 15. Therefore 13 trend/time samples should be put into a circular buffer from row 3 to row 15 inclusive. The maximum number of samples sent in a 'read' trend request is configured in the Trend map as 10 - therefore there are three 'spare' samples (a minumum of 1 'spare' sample is required).

Audit signals

The following 6 alarm signals are in the register map (from signal list 130), and will therefore be relevant for any alarm log requests:

ALMSIG.A001.U16
ALMSIG.A002.U16
ALMSIG.L001.
ALMSIG.L002.
ALMSIG.L003.
ALMSIG.L004.

The following 5 signals are in the Audit list and in the register map (from signal list 31), and will therefore be relevant for any event log requests:

EVTSIG.A001.U16
EVTSIG.A002.U16
EVTSIG.L001.
EVTSIG.L002.
EVTSIG.L003.

6. References

1. AMOCAMS/MODULAR AI-Net Protocol Definition Reference Manual. March 26 1993

Return to Application Notes Menu

Return to the List of Manuals