REVISIONS					
REV #	**ECN #**	**Description of Change**	**Approved**	**Quality**	**Date**
1	NDI	Initial Release	SJS	RV	3/18/17

Relay Controller

Interface Design Description

1 Introduction 3
2 References 3
3 Physical Layer 3
4 Application Layer Protocol 4
4.1 Message Format 4
4.2 Classes 5
4.3 Operations 5
4.4 Functions 6
4.4.1 Diagnostic Class 0x01 Functions 6
4.4.1.1 DeviceDetailedStatus (Class 0x01, Function 0x10) 7
4.4.1.2 ConfigDataParameter (Class 0x01, Function 0x40) 9
4.4.1.3 ManufacturingInformation (Class 0x01, Function 0x70) 10
4.4.1.4 PrepareToRestart (Class 0x01, Function 0x09) 12
4.4.1.5 DiagRestart (Class 0x01, Function 0x63) 12
4.4.2 Clu Class 0x0B Functions 13
4.4.2.1 CluInpInfo (Class 0x0B, Function 0x10) 13
4.4.2.2 CluInpOverride (Class 0x0B, Function 0x12) 14
4.4.2.3 CluMasterOff (Class 0x0B, Function 0x33) 15
4.5 AltoNET™ - Multinet Interface Details 16
4.5.1 Physical Layer Details 16
4.5.2 Message Format 16
4.6 Configuration Table Editing 17

1 Introduction

This document describes the protocol that is used for all system inter-board communication for the Alto line of Digital Audio amplifiers.

Data Dictionary

BCC	Block Check Character
CLU	Configurable Logic Unit
Clu	Configurable Logic Unit
Cfig	Configuration
Config	Configuration
Config Table	Configuration Table
Configuration	A collection of user settable parameters
DOC	Document
IDD	Interface Design Description
Ind	Clu Indicators
Inp	Clu Inputs
Ovr	Override
PRS	Product Requirements Specification
SRS	Software Requirements Specification
RC	Relay Controller
Rly	Clu Relays
RlyInd	Clu Relays and Indicators

2 References

3 Physical Layer

See Appendix Sections for Details regarding the Physical Layers.

4 Application Layer Protocol

4.1 Message Format

The external message format may vary depending upon the physical interface used. See the Appendix section for details regarding the message format.

The Internal System Message format is common to all physical interfaces, and it is described as follows (See Appendix for any Prefix or Suffix added to the External Message)**:**

Class	Oper	Func	Seq	Length	Data Payload	BCC
8 bits	8 bits	8 bits	8 bits	8 bits	<Data 1>…<Data 26>	8 bits

- The first five bytes of the message are the header which are **mandatory**
- Data payload is optional so a data **length** of ZERO is valid
- Unused payload bytes beyond the last byte of valid data are functionally ignored.
- **Seq**uence **Id**entifier is a numeric value that uniquely identifies a data transport protocol. It is a rolling number starting from 0 through 255. In the case of a parsing error or a missing number in the Seq series; the generated error message echoes the Seq of the offending message. When the Host is transmitting a message, it should increment the Seq for the next message. When the Client receives a message it will respond with the same Seq. If a message is missing, the Host may resend the message based on the importance of the message.
- Client responds with an AckNak message, for every “Set” command and a Response message for every “Get” command.
- There is a BCC-8 byte appended to the end of message.

o The Message Acknowledge system returns the following data in the Ack/Nak byte of all messag0x00 (0) Good Status

o 0x40 Busy, Waiting to Respond
o 0xFF (-1) Invalid BCC
o 0xFE (-2) Invalid Argument
o 0xFD (-3) Function Not Executed
o 0xFC (-4) Invalid CRC
o 0xFB (-5) - 0xF0 (-15) N/A
o 0xDF – 0xD4 N/A
o 0xD4 Hardware Fault
o 0xD3 – 0xCE N/A
o 0xCD Diag Session Not Active
o 0xCC – 0xB8 N/A
o 0xB7 – 0xB0 assigned to Relay Controller
o 0xAF – 0x80 N/A

- The Message Acknowledge system returns the following data in the Seq byte of all messages:

o Next Sequence Number for Ack
o Echo the ID received for Nak

- Data (Endianess)

o By convention, all scalar data types are presented in big Endian format.
o In bit fields, bits are numbered from right to left:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

o In Byte fields, bytes are number from right to left:

Byte 1	Byte 0

- **Errors**: In the event of errors, for example, Invalid Argument, Function Not Executed and Invalid BCC, the Error message is returned using the same sequence ID as the offending message.
- BCC is calculated as a running XOR of each byte in the message, excluding the prefix (32 bytes).

4.2 Classes

Table of possible Class IDs is shown below:

Class ID	Class Name
0x00	N/A
0x01	Diagnostic
0x02	N/A
0x03	N/A
0x04	N/A
0x08	N/A
0x09	N/A
0x0B	Clu

4.3 Operations

Table of possible Operation IDs is shown below:

Operation ID	Name	Description
0x00	Set	Host Writes data
0x01	Get	Host Reads data
0x05	Unsolicited	N/A RC Unsolocited
0x06	Response	Data returned by Client from a Get Operation
0x07	ACKNAK	Response to a message that was received
0x08	Status	N/A RC lStatus like heartbeat
0x80	MethodStart	Start a Method like DiagRestart
0x82	MethodAction	N/A RC Action like load, step, pause,continue &c
0x84	MethodStop	N/A RC Stop a Method
0x86	MethodResult	N/A RC Result of a Method

4.4 Functions

Each of the following sections lists all the possible Function tokens associated with a particular Class.

4.4.1 Diagnostic Class 0x01 Functions

The Diagnostic Class provides an interface to monitor and manage devices in the system.

Function ID	Function Name
0x09	PrepareForRestart
0x10	DeviceDetailedStatus
0x20	N/A
0x40	ConfigDataParameter
0x63	DiagRestart
0x70	ManufacturingInformation
0x74	N/A
0x75	N/A
0x78	N/A

4.4.1.1 DeviceDetailedStatus (Class 0x01, Function 0x10)

If the Overall byte is ZERO then initialization is complete and there are no faults. If the Fault bit is set then examine the Fault byte for details. Note that the three 28V power services are AND-ed together to determine a fault.

The Config Table/EEPROM fault indicates that either the Config Table is invalid or an EEPROM Read or Write failed. Reloading the Config Table should correct the problem otherwise there is an EEPROM hardware fault.

The six voltages are unsigned scaled integer shorts with a resolution of 0.01 volts. The normal range of the dual supply Power 28B5V to the Clu Inputs depends upon the selection in the Config Table.

Both the 8 relay and 8 indicator ON/OFF states are adjusted for the three 28V power supply masks in the Config Table. In addition, the states are also adjusted for overrides via applicable Clu Class messages described later in this document. The state of the algorithm output is logical but the adjusted state is the actual physical.

Note that there are 27 Clu inputs thus the fourth of these bytes contains only three inputs. Because the input states are adjusted for overrides their logical state represented in this message may not be the phsical pin state. The logical state is what us used by the control algorithms.

Name	Data Type	Size Bytes	Range	Description
Overall	Bit field	1	Bit 0: 0 – 1 Bit 1: 0 – 1	Fault: 0:normal, 1:fault Init: 0:normal, 1:initializing
Fault	Bit	1	Bit 0: 0 – 1 Bit 1: 0 – 1 Bit 2: 0 – 1 Bit 3: 0 – 1 Bit 4: 0 – 1	1 = fault 28V (AND of the 3 28V power services) 1 = fault 5V 1 = fault 8V 1 = fault 28V5V (dual power supply for Clu inputs) 1 = fault Config Table / EEPROM
Power 28V5V	Short 0.01V	2	0 – 2800 + overrange 0 – 500 +overrange	Dual Supply selected in Config Table
Power5V	Short 0.01V	2	0 – 500 + overrange	5V Supply
Power 8V	Short 0.01V	2	0 – 800 + overrange	8V Supply
Power 28V 1	Short 0.01V	2	0 – 2800 + overrange	1st of three 28V Services
Power 28V 2	Short 0.01V	2	0 – 2800 + overrange	2nd of three 28V Services
Power 28V 3	Short 0.01V	2	0 – 2800 + overrange	3rd of three 28V Services
Relay Results	Bit field	1	8 one bit fields	Physical state 1=ON, 0=OFF of the 8 relays after inhibits and overrides
Indicator Results	Bit field	1	8 one bit fields	Physical state 1=ON, 0=OFF of the 8 indicators after adjustment for inhibits and overrides
Input 0 – 7 Results	Bit field	1	8 one bit fields	Logical state 1=ON, 0=OFF of 8 inputs after adjustment for overrides. May not be Pin State.
Input 8– 15 Results	Bit field	1	8 one bit fields	Logical state 1=ON, 0=OFF of 8 inputs after adjustment for overrides. May not be Pin State.
Input 16 – 23 Results	Bit field	1	8 one bit fields	Logical state 1=ON, 0=OFF of 8 inputs after adjustment for overrides. May not be Pin State.
Input 24– 26 Results	Bit field	1	3 one bit fields	Logical state 1=ON, 0=OFF of 3 inputs after adjustment for overrides. May not be Pin State.
Input 0 – 7 Pin State	Bit field	1	8 one bit fields	Physical state 1=ON, 0=OFF of 8 input pins. May not match Logical State.
Input 8– 15 Pin State	Bit field	1	8 one bit fields	Physical state 1=ON, 0=OFF of 8 input pins. May not match Logical State.
Input 16 – 23 Pin	Bit field	1	8 one bit fields	Physical state 1=ON, 0=OFF of 8 input pins. May

Name	Data Type	Size Bytes	Range	Description
State				not match Logical State.
Input 24– 26 Pin State	Bit field	1	3 one bit fields	Physical state 1=ON, 0=OFF of 3 input pins. May not match Logical State.

Table of Detailed Status Data Fields

Operations Sent: ***Get, Unsolicited***

DeviceDetailedStatus Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x01	0x01	0x10	N	0	26 bytes x 0x00	0xyy

DeviceDetailedStatus Unsolicited (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8
0x01	0x05	0x10	N	24	Overall	Fault	28V5V	28V5V	5V	5V	8V	8V

Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17
28V 1	28V 1	28V 2	28V 2	28V 3	28V 3	Rly	Ind	Inp 0 -7

Data 18	Data 19	Data 20	Data 21	Data 22	Data 23	Data 24	Padding	BCC
Inp 8 - 15	Inp 16 - 23	Inp 24 - 26	Pin 0 - 7	Pin 6 - 15	Pin 16 - 23	Pin 24 - 26	2 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

DeviceDetailedStatus Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8
0x01	0x06	0x10	N	24	Overall	Fault	28V5V	28V5V	5V	5V	8V	8V

Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17
28V 1	28V 1	28V 2	28V 2	28V 3	28V 3	Rly	Ind	Inp 0 -7

Data 18	Data 19	Data 20	Data 21	Data 22	Data 23	Data 24	Padding	BCC
Inp 8 - 15	Inp 16 - 23	Inp 24 - 26	Pin 0 - 7	Pin 6 - 15	Pin 16 - 23	Pin 24 - 26	2 bytes x 0x00	0xyy

DeviceDetailedStatus AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x10	N	1	Ack Nak	25 bytes x 0x00	0xyy

DeviceDetailedStatus AckNak (Host to Client):

4.4.1.2 ConfigDataParameter (Class 0x01, Function 0x40)

This command dynamically updates the Configuration Parameter Values stored both in RAM and permanently in EEPROM. This command also retrieves other Configuration data like preset factory limits, data types, descriptions, etc., so that a list may be generated, viewed and/or printed. Parameters that are multiple bytes require special handling in software. For example, a 32-bit integer would require four Config Table lines. The Parameter Values, Limits for range check and Default Values would be assembled in the endian order required to be useful. Only single byte Parameters are handled in a general purpose manner.

The Relay Controller (RC) itself uses the actual Parameter (Value) and also the high / low limits and whether the Data Type is signed or not to range check the incoming Parameter Value. Some of tthe message and what is stored in the Config Table is ignored by the RC.

Name	Data Type	Size in Bytes	Range	Description
Board	Enum	1	Ignored	Ignored by RC
Index	Byte	1	0 - 255	Index of the Parameter Cell
Multiple	Byte		Ignored	Ignored by RC
Parameter	Byte	1	0 - 255	Parameter Value being set
ASCII description	Byte (ASCII)	20	Valid ASCII values	Up to 20 character description of the data field. Ignored by RC.
Value	Byte	1	0 - 255	The current value being used for the Parameter
Low limit	Byte	1	0 - 255	The factory lower limit for the parameter
High limit	Byte	1	0 - 255	The factory higher limit for the parameter
Default	Byte	1	0 - 255	The factory default value
DataType	Byte	1	0 - 255	0 -255 except 3 result in an unsigned range limit check. The value 3 results in a signed range limit check.

Table of ConfigDataParameter Data Fields

Operations Sent: ***Get, Set***

ConfigDataParameter Get (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x01	0x01	0x40	N	3	Board	Index	Multiple	23 bytes x 0x00	0xyy

ConfigDataParameter Set (Host to Client): (Weight on Wheels active, sets Active Config values)

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Padding	BCC
0x01	0x00	0x40	N	4	Board	Index	Multiple	Parameter	22 bytes x 0x00	0xyy

Operations Returned: **Response, *AckNak***

ConfigDataParameter Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x01	0x06	0x40	N	26	Index	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII

Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17	Data 18	Data 19
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII

Data 20	Data 21	Data 22	Data 23	Data 24	Data 25	Data 26	BCC
ASCII	ASCII	Value	Low	High	Default	Datatype	0xyy

ConfigDataParameter AckNak:

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x40	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.1.3 ManufacturingInformation (Class 0x01, Function 0x70)

This Code Version is embedded in Program Flash and the other Manufacturing Information (MI) is located in Serial EEPROM. The MI contains information about Software and Hardware, such as SW Code version, HW version and HW Serial Number. The Major and Minor version byte value can be 00 – 99. Reporting of version information should always be padded to 2 digit (ex: 07.02 not 7.2). There are 11 Bytes in both the Set and [Get] Response] reserved for use by manufacturing.

Note that location of this information in the messages matches the Amplifier. This leaves interleaved unused Data bytes.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
SW Version	Byte	2	(00-99) . (00-99)	Software Version (major.minor)
HW Version	Byte	1	00-255	Hardware Version - The integer value must match the Hardware BOM version
Reserved1	Byte	1	00-255	Reserved for Alto use
Unused	Byte	1	00-255	Returns 0
Reserved2	Byte	1	00-255	Reserved for Alto use
Unused	Byte	2	00-255	Return 0
Multinet Address	Byte	1	00-31	Address of unit 0-31 (gets added to Unit Type to make Addr)
Unused	Byte	1	00-255	Returns 0
Unit Serial Number	Unsigned Long	4	100000-999999	Unit serial number (6 decimal digits)
Unused	Byte	1	00-255	Returns 0
Unit Type	Byte	1	00-255	Code indicating unit type (30 +)
Unit Sub Type	Byte	1	00-255	Code indicating unit sub type
Unit ID Number	Byte	1	00-255	Code indicating unit ID number (1 to #units)
Config ID Number	Byte	1	00-255	Indicates ID code for loaded configuration (if applicable)
MP Rev	Byte	1	00-255	Indicates the MP Rev (MOD Level) used to build unit
Main SW Part#	Unsigned Double	3	100000-999999	Main Software Alto Part# (6 decimal digits). The MSB is not sent, it should be set to 0.

Table of Manufacturing Information Data Fields

<u>Operations Sent</u>: ***Get ManufacturingInformation Get (Host to Client):***

Class	Oper	Func	Seq	Length	Padding	BCC
0x01	0x01	0x70	N	0	26 bytes x 0x00	0xyy

ManufacturingInformation Set (Host to Client): (Factory Only!!!)

Class	Op	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
0x01	0x00	0x70	N	22	HW Ver	Reserved 1	0x00	Reserved2	0x00	MN Address

Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17-21	Data 22	Padding	BCC
Unit S/N	Unit S/N	Unit S/N	Unit S/N	0x00	Unit Type (read only)	Unit Sub Type	Unit ID Number	CFG ID Number	MP Rev	Ignored, read only	0x00	4 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

ManufacturingInformation Response from Get (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8
0x01	0x06	0x70	N	26	SW Ver	SW Ver	HW Ver	Reserved1	0x00	Reserved2	0x00	0X00

Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15
0x00	MN Address	Unit S/N MSB	Unit S/N Byte 3	Unit S/N Byte 2	Unit S/N LSB	0x00

Data 16	Data 17	Data 18	Data 19	Data 20	Data 21	Data 22	Data 23
Unit Type	Unit Sub Type	Unit ID Number	CFG ID Number	MP Rev	Main SW PN Byte3	Main SW PN Byte2	Main SW PN LSB

Data 24	Data 25	Data 26	BCC
0x00	0x00	0x00	0xyy

ManufacturingInformation AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x70	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.1.4 PrepareToRestart (Class 0x01, Function 0x09)

PrepareToRestart and DiagRestart are paired messages of which the former allows a device to prepare for restart before receiving and executing the actual restart message. In the case of an Amplifier prepare means gracefully mute to avoid audio artifacts. In a Relay Controller, a discrete device, no application preparation is necessary. A watchdog timeout is induced to cause a warm restart. During this process an attempt is made to acknowledge that a DiagRestart is imminent.

Data Fields

Name	Data Type	Size in Bytes	Range	Description

Table of PrepareToRestart Data Fields

Operations Sent: S*et*
PrepareToRestart Set (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x01	0x00	0x09	N	0	26 bytes x 0x00	0xyy

Operations Returned: *AckNak*

PrepareToRestart AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x70	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.1.5 DiagRestart (Class 0x01, Function 0x63)

Refer to the description of PrepareToRestart and DiagRestart in PrepareToRestart.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
				Software Version

Table of Diag*Restart* Data Fields

Operations Sent: MethodStart
DiagRestart Set (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x01	0x80	0x63	N	0	26 bytes x 0x00	0xyy

Operations Returned: *AckNak*

DiagRestart AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x70	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2 Clu Class 0x0B Functions

Func	Name
0x10	CluInpInfo
0x12	CluInpOverride
0x33	CluMasterOff

4.4.2.1 CluInpInfo (Class 0x0B, Function 0x10)

CluInpInfo is subject to significant change during development. Information available from the Config Table and Diagnostic Class Function DetailedDeviceStatus is not repeated here. Ultimately, all 26 available Data Bytes will be used. Let's begin with just a few things then discover what else is really useful.

Debounce Data - when all all 8 samples are OFF the debounced state is OFF. Likewise with ON. There is no m of n processing, for example, when 6 of 8 are OFF then the debounced state is OFF. External hardware like buttons and switches are expected to be stable after settling time.

Status Data - Bit 0 is the ON/OFF state adjusted for overrides thus not necessarily the pin state.
Bit 1 signals a change of debounced state from OFF->ON or ON->OFF occurred. Bit 2 is set during a hold delay. Bit 3 set indicates the algorithm is running. Bit 4 set indicates the algorithm associated with the Input is running but has temporarily paused its normal functionality. Alternative processing is active.

Parm 1 and Parm 2 Data reveal information pertinent to Sequence, Binary Count and Hot Cup algorithms. These three functions plus One Shot also use Parrm Config. Be careful to distinguish between Config/Cfg and Data/Dat. Parm Config is static after the algorithm is initialized unlike Parm 1 Data and Parm 2 Data which may or may not be dynamic while in the run state.

Name	Data Type	Size in Bytes	Range	Description
Input Unit	Byte	1	0 - 26	The input unit's info requested
Debounce Data	Bit field	1	8 single bits	Each bit is a sample, Bit 0 newest, 7 oldest. 1=ON, 0=OFF.
Status Data	Bit field	1	Bit 0: 1=on Bit 0: 1=change Bit 0: 1=hold Bit 0: 1=run Bit 0: 1=pause	0=off 0=not change 0=not hold delay 0=not run 0=not pause
Parm 1 Data	Byte	1	0 - 255	Varies depending upon the Input function
Parm 2 Data	Byte	1	0 - 255	Varies depending upon the Input function
Parm 3 Data	Byte	1	0 - 255	Varies depending upon the Input function

Table of CluInpInfo Data Fields

<u>Operations Sent</u>: ***Get***
CluInpInfo Get (Host to Client):

Class	Op	Func	Seq	Length	Data 1	Padding	BCC
0x0B	0x01	0x10	N	1	Inp Unit	25 bytes x 0x00	0xyy

<u>Operations Returned</u>: ***Response, AckNak***
CluInpInfo Data Response from Get (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Padding	BCC
0x0B	0x06	0x10	N	6	Inp Unit	Debounce Data	Status Data	Parm 1 Data	Parm 2 Data	Parm 3 Data	20 bytes x 0x00	0xyy

CluInpInfo AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x0B	0x07	0x10	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2.2 CluInpOverride (Class 0x0B, Function 0x12)

CluInpOverride provides an individually selectable override of the Pin State for each of the Inputs. There is an enable/disable and on/off bit for each of the inputs. With this capability input conditions can be simulated that would be difficult or impossible to duplicate with external hardware. This allows rigorous testing of individual algorithms and overall system control strategy. Race conditions, other intermittent relationships and error conditions can be easily introduced into the CLU. It is also a quick and easy way to debug fundamental logic errors.
Note that there are 27 Clu inputs thus the fourth of these bytes contains only three inputs.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Inputs 0 - 7 Enable	Bit field	1	8 one bit fields	1=Enable Override, 0=Disable Override
Inputs 8 - 15 Enable	Bit field	1	8 one bit fields	1=Enable Override, 0=Disable Override
Inputs 16 - 23 Enable	Bit field	1	8 one bit fields	1=Enable Override, 0=Disable Override
Inputs 24 - 26 Enable	Bit field	1	3 one bit fields	1=Enable Override, 0=Disable Override
Inputs 0 - 7 OnOff	Bit field	1	8 one bit fields	1=ON, 0=OFF
Inputs 8 - 15 OnOff	Bit field	1	8 one bit fields	1=ON, 0=OFF
Inputs 16 - 23 OnOff	Bit field	1	8 one bit fields	1=ON, 0=OFF
Inputs 24 - 26 OnOff	Bit field	1	3 one bit fields	1=ON, 0=OFF

Table of CluInpOverride Data Fields

***Operations Sent*: Get, Set**

CluInpOverride Get (Host to Client):

Class	Op	Func	Seq	Length	Data 1	Padding	BCC
0x0B	0x01	0x12	N	1	Inp Unit	25 bytes x 0x00	0xyy

CluInpOverride Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3
0x0B	0x00	0x12	N	8	Inputs 0 - 7 Enable	Inputs 8 - 15 Enable	Inputs 16 - 23 Enable

Data 4	Data 5	Data 6	Data 7	Data 8	Padding	BCC
Inputs 24 - 26 Enable	Inputs 0 - 7 OnOff	Inputs 8 - 15 OnOff	Inputs 16 - 23 OnOff	Inputs 24 - 26 OnOff	18 bytes x 0x00	0xyy

CluInpOverride* Returned: *Response, AckNak

CluInpOverride Data Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3
0x0B	0x06	0x12	N	8	Inputs 0 - 7 Enable	Inputs 8 - 15 Enable	Inputs 16 - 23 Enable

Data 4	Data 6	Data 7	Data 8	Data 9	Padding	BCC
Inputs 24 - 26 Enable	Inputs 0 - 7 OnOff	Inputs 8 - 15 OnOff	Inputs 16 - 23 OnOff	Inputs 24 - 26 OnOff	17 bytes x 0x00	0xyy

CluInpOverride AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x0B	0x07	0x12	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2.3 CluMasterOff (Class 0x0B, Function 0x33)

CluMasterOff resets all relays, indicators and algorithms. If this occurs while any input is active then after debouncing (< 50 ms) the input will turns its algorithm ON.

Name	Data Type	Size in Bytes	Range	Description
Dummy	Byte	1	0xCF	Just a Dummy

Table of CluMasterOff Control Data Fields

Operations Sent: ***Set***

CluMasterOff Set (Host to Client):

Class	Oper	Func	Seq	Len	Data 1	Data Padding	BCC
0x0B	0x00	0x33	N	1	Dummy	25 bytes x 0x00	0xyy

CluMasterOff* Returned: *AckNak

CluMasterOff AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x0B	0x07	0x33	N	1	AckNak	25 bytes x 0x00	0xyy

4.5 AltoNET™ - Multinet Interface Details

AltoNET™ supports RS-485 4-wire physical and data link layers and provides a proprietary applications layer message protocol for communicating with Alto Products like Amplifiers and Relay Controllers.

4.5.1 Physical Layer Details

The RS485 interface is differential with up to -10 to +10 Volt input range.
Differential signal should be at least +/- 2V fully loaded.
The protocol although full duplex supports effectively half duplex and expects a baud rate of 115,200, 8 bits of data, no parity, 1 stop bit and no flow control.

The physical connections required implementing this interface as Inputs and Outputs are:

Signal	PA700 BD	Main BD	Description
Tx+	J5-27	J2-9	Transmit +
Tx-	J5-26	J2-8	Transmit -
Rx+	J5-25	J2-7	Receive +
Rx-	J5-24	J2-6	Receive -

4.5.2 Message Format

The protocol is effectively Master-Slave Ping-Pong. Slaves do not speak unless first spoken to because only one Slave can talk at a time. When not talking a Slave must disable physically its transmitter not enabling it until it receives a message with an Address that matches the Slave's or the wildcard address 255.

The Address byte is a combination of the Unit Type code and the Unit Logical Address. For the 107800 (RC) the unit type is 64 (0x40). The unit type is hard coded into the Manufacturing Info field. The Logical address is entered into the Manufacturing Info field. The Address byte should be the sum of these 2 values.

An address of 0xFF (255) is interpreted as a wildcard address. Any Alto Multinet device will respond to this address so it should only be used when a device is alone on a network. It is intended to be used only for manufacturing and maintenance.

All messages sent via AltoNET will have the same basic format shown below:

	Prefix	Class	Operation	Function	Addr	Length	Data	BCC	CR/LF
Actual bits	16 bits	8 bits	8 bits	8 bits	8 bits	8 bits	<Data 1>…<Data N>	8 bits	16 bits
ASCII Xmitted bits	32 bits	16 bits	16 bits	16 bits	16 bits	16 bits	<Data 1>…<Data N>	16 bits	16 bits

- The messages are sent and received as an ASCII string, so ascii "AA550008" is actually sent and received as 0x41 0x41 0x35 0x35 0x30 0x30 0x30 0x38 ….
- ASCII string must be terminated with CR/LF (0x0D 0x0A)
- The prefix is 0xAA 0x55
- The next Five bytes form the header.

- Unused bytes beyond the last byte of valid data are functionally ignored.
- BCC is calculated as a running XOR of each HEX byte in the message before conversion to ASCII, excluding the prefix (32 bytes).
- Message header, data length, and BCC are checked for validity in returned ACK/NACK.
- Each Data Field is 8 bits unless otherwise indicated throughout the document.
- There must be a minimum delay between sent messages of 10mS.

The BCC Block Check Character is an 8-bit Character created by performing consecutive exclusive OR (XOR) operation of the entire message byte-by-byte. The BCC is appended to the last byte of the message as showed in the next diagram. It is always placed as the last byte (position 31). Any unused bytes between the payload and BCC should be padded to 0x00

4.6 Configuration Table Editing

Configuration table data Parameter Byte Values can be edited using the Alto Forte Manager.