

Operation & Installation Manual

44xx & 48xx Series Multi-Channel Ethernet Attenuator

EAR NOTICE

THIS DOCUMENT CONTAINS CONTROLLED TECHNICAL DATA UNDER THE JURISDICTION OF THE EXPORT ADMINISTRATION REGULATIONS (EAR), 15 CFR 730-774. IT CANNOT BE TRANSFERRED TO ANY FOREIGN THIRD PARTY WITHOUT THE SPECIFIC PRIOR APPROVAL OF THE U.S. DEPARTMENT OF COMMERCE BUREAU OF INDUSTRY AND SECURITY (BIS). VIOLATIONS OF THESE REGULATIONS ARE PUNISHABLE BY FINE, IMPRISONMENT, OR BOTH.

This documentation may not be reproduced in any form, for any purpose unless authorized in writing by Weinschel, a part of API Technologies Corp.



Table of Contents

Table of Contents.....	2
1. Definitions	5
2. Safety Summary.....	5
2.1. General Precautions	5
2.2. Detailed Precautions.....	6
2.3. Safety Symbols	6
2.4. Electrostatic Discharge Sensitive.....	7
3. General Information	7
3.1. Document Overview	7
3.2. Related Manuals	7
3.3. Equipment Overview	7
4. Installation	10
4.1. Initial Setup.....	10
4.2. I/O Control Connections	11
4.3. RF Port Connectors	11
5. Remote Operation	12
5.1. Serial Console Boot Display	12
5.2. 10/100BaseT Ethernet.....	12
5.3. USB	14
6. Command Operation	14
6.1. 488.2 Status Reporting.....	15
6.2. Command Access Restrictions	16
6.3. Command Execution and Buffering	16
6.4. Command Reference	17
7. Application Specific Commands.....	17
7.1. Attenuation Commands	17
ATTN.....	18
ATTN?	18
STEPSIZE.....	19
STEPSIZE?	19
INCR	19
DECRL.....	19
7.2. General RF Configuration and Status Commands.....	20
SHOW STAT.....	20
SET RFCONFIG.....	20
SHOW RFCONFIG	Error! Bookmark not defined.
RFCONFIG?.....	20
7.3. 488.2 Common Commands	21
*CLS	21
*IDN?.....	21

*OPC?	21
*ESR?.....	21
*RST	22
*TST?.....	22
ERR?	22
7.4. Setup and Configuration Commands	22
FACTORY PRESET.....	23
SET ACCESS.....	23
SET EOS	23
7.4.1 Network	24
SET IPADDR.....	24
SET DHCP.....	24
SET AUTOIP.....	25
SET NETMASK.....	25
SET GATEWAY	25
SET ETH SPEED.....	25
SET TCP CONNECT.....	25
SET TCP SERVER.....	26
SET TCP KEEPALIVE	26
SET TCP TIMEOUT.....	26
SET TCP ECHO	26
SET UDP SERVER	26
SET TELNET SERVER	27
SET TELNET	27
SET HTTP SERVER.....	27
SET ANNOUNCE SERVER.....	27
SET NETSTAT.....	28
IPCONFIG?.....	28
MACADDR?	28
PING	28
7.5. Show Commands	29
SHOW EOS	29
SHOW SET.....	29
SHOW USB	30
SHOW NET	30
SHOW IPADDR.....	30
SHOW NET TCP.....	31
SHOW NET UDP	31
SHOW NET TELNET	32
SHOW NET HTTP.....	32
SHOW NET ANNOUNCE.....	32

7.6. MISC Commands	32
CONSOLE.....	32
CONSOLE?	33
USB CONSOLE.....	33
DSA.....	33
DELAY	33
REBOOT	34
RUN	34
TEMP?	34
TIME?.....	34
TIMESTAMP	34
TIMESTAMP?.....	34
REPEAT	35
SYSTEST.....	35
HELP	35
8. Firmware and Drivers	36
8.1. Installing Weinschel USB CDC Driver.....	36
9. Error Code and Messages.....	39
10. Instrument Security Procedures for Secure Environments.....	40
10.1. Clearing and Sanitization.....	40
10.2. Clearing Internal Memory.....	40
10.3. Sanitizing Internal Memory.....	40
11. Maintenance	42
11.1. Inspection	42
11.2. Preventative Maintenance	42
11.3. Machined Surfaces and Hardware	42
11.4. Chassis Cleaning	43
11.5. Connector Cleaning	43
12. Replacement Parts List and Drawings.....	44
12.1. Factory Service and Repairs	44
13. Contacting Weinschel.....	45
13.1. Manufacturer Warranty	45
14. Appendix	46
14.1. Revision History	46

1. Definitions

The following definitions apply to WARNINGS, CAUTIONS, and NOTICES found throughout this manual.



WARNING: An operating or maintenance procedure, practice, statement, condition, etc., which, if not strictly observed, could result in injury and/or death of personnel. Do not proceed beyond a WARNING symbol until all the indicated conditions have been fully understood and/or met.



CAUTION: An operating or maintenance procedure, practice, statement, condition, etc., which, if not strictly observed, could result in damage or destruction of the equipment or long-term health hazards to personnel. Do not proceed beyond a CAUTION symbol until all the indicated conditions have been fully understood and/or met.



NOTICE: An essential operating or maintenance procedure, condition, or statement that must be highlighted.

2. Safety Summary

2.1. General Precautions

The following are general precautions that are not related to any specific procedure and, therefore, do not appear elsewhere in this publication. These are precautions that personnel must understand and apply during various phases of instrument operation or service.

Potentially lethal voltages are present in this instrument. Serious shock hazards from voltages above 70 volts may exist in any connector, chassis, or circuit board. Observe the following precautions.



To minimize shock hazard, the instrument chassis must be connected to an electrical ground. Using the supplied three-conductor power cable ensures that the instrument can be firmly connected to the ac power source and electrical ground at a grounded power outlet. If using a 3-wire adapter be sure to connect the ground lead to earth ground.

Use the buddy system any time work involving active high voltage components is required. Turn OFF the power before making/breaking any electrical connection. Regard any exposed connector, terminal board, or circuit board as a possible shock hazard. DO NOT replace any component or module with power applied.

If test conditions to live equipment are required, ground the test equipment before probing the voltage or signal to be tested.

Personnel working with or near high voltage should be familiar with modern methods of resuscitation.

DO NOT wear jewelry (rings, bracelets, metal watches, and/or neck chains) while working on exposed equipment. Be very cautious about using hand tools near exposed backplanes, bus bars, and/or power supply terminals. Use properly insulated tools. When making test connections to the power supply terminals and bus bars, use only insulated probe tips.

Verify that the instrument is set to match the available line voltage and the correct fuse is installed.

DO NOT install substitute parts or perform any unauthorized modification to this instrument. Contact Weinschel to acquire any information on replacement parts or returning the instrument for repair. Unauthorized modification can cause injury to personnel and/or destruction of the instrument.

Operating personnel must not remove instrument covers. Component replacement or adjustments must be performed by qualified service personnel.

DO NOT operate the instrument near or in the presence of flammable gases or fumes.

2.2. Detailed Precautions

The following WARNINGS, CAUTIONS and NOTES appear throughout the text of this manual and are repeated here for emphasis.



CAUTION

All procedures and/or steps identified as must be followed exactly as written and according to industry accepted ESDS device handling procedures. Failure to comply WILL result in ESD damage.

- DO NOT use a nylon bristle brush in any solvent as the bristles may dissolve and cause damage to the circuit card or component.
- DO NOT use ultrasonic cleaning on parts or assemblies containing electrical or electronic components.
- DO NOT bend pins of electrical connectors when using fiber-bristle brush.

Compressed air used for cleaning and/or drying can create airborne particles that may enter the eye. Goggles/face shields should be worn. DO NOT direct air stream towards self or other personnel. Pressure should be restricted to a maximum of 15 psi to avoid personal injury.

Under no circumstances should a wire brush, steel wool, or abrasive compound be used on any surface. Using these items will cause extensive damage to the instrument's surface.

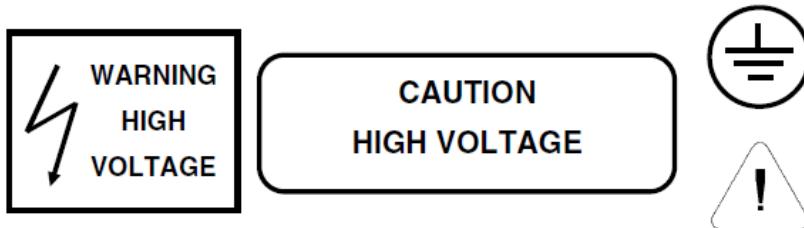


NOTICE

DO NOT return any instrument or component to Weinschel without receiving prior factory authorization.

2.3. Safety Symbols

The following symbols are used to identify safety hazards found throughout this publication and/or located on the instrument.



2.4. Electrostatic Discharge Sensitive

The equipment documented in this manual contains certain Electrostatic Discharge Sensitive (ESDS) components or parts. Therefore, certain procedures/steps are identified by the use of the symbol . This symbol is used in two ways:



- When the ESDS symbol is placed between a paragraph and title, that paragraph, including all subparagraphs, is considered ESDS device handling procedure.
- When the ESDS symbol is placed between a procedure/step number and the text, all of that procedure is considered an ESDS device handling procedure.

All procedures and/or steps identified as ESDS must be followed exactly as written and according to accepted ESDS device handling procedures. Failure to comply WILL RESULT in ESDS damage.

3. General Information

3.1. Document Overview

This manual is intended to support the operation of the Weinschel Model 44xx & 48xx Multi-Channel Programmable Ethernet Attenuators.

A general description of the product, information regarding installation and user operation, and maintenance instructions are included in this manual. Assembly drawings, wiring diagrams, and bill of materials are located in the document appendix.

3.2. Related Manuals

The following manuals contain information that may be used in conjunction with this manual to operate, service, or calibrate this device.

Manual	Title
IMxxx	Manual, Operating Instructions, Attenuation Control Software (ACCS)
IMxxx-1	Manual, Drawings, Replacement Parts Lists, Model 44xx & 48xx Series

3.3. Equipment Overview

The Model 44xx & 48xx are binary programmable attenuator designed to be controlled both via Ethernet and USB.

The 44xx & 48xx Series are multi-channel configurations where RF signal is routed through the left or right mounted Ports. This series can be configured for up to 4 or 8 independent channels of attenuation dependent on model.

The diagram below outlines the basic RF layout within the Model 44xx & 48xx series attenuators.

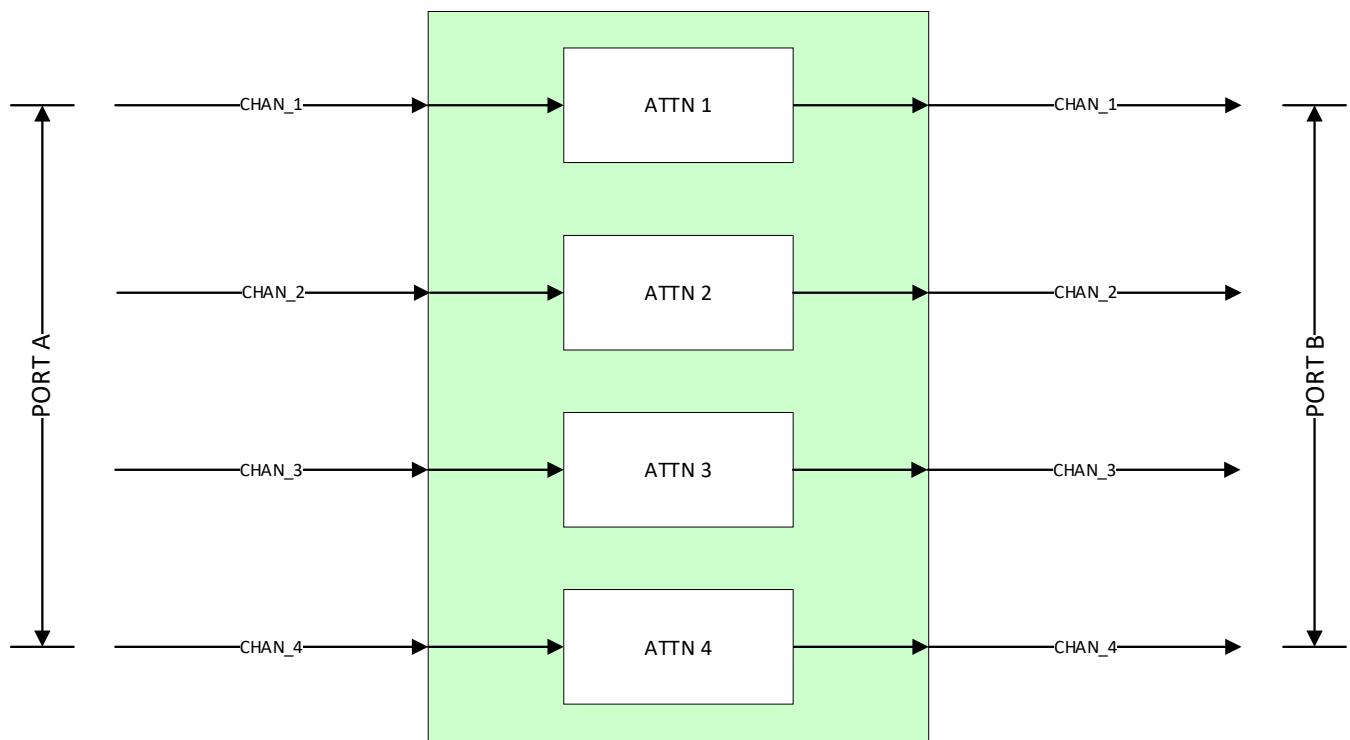


Figure 1: Functional Block Diagram of 4-channel 44xx series module

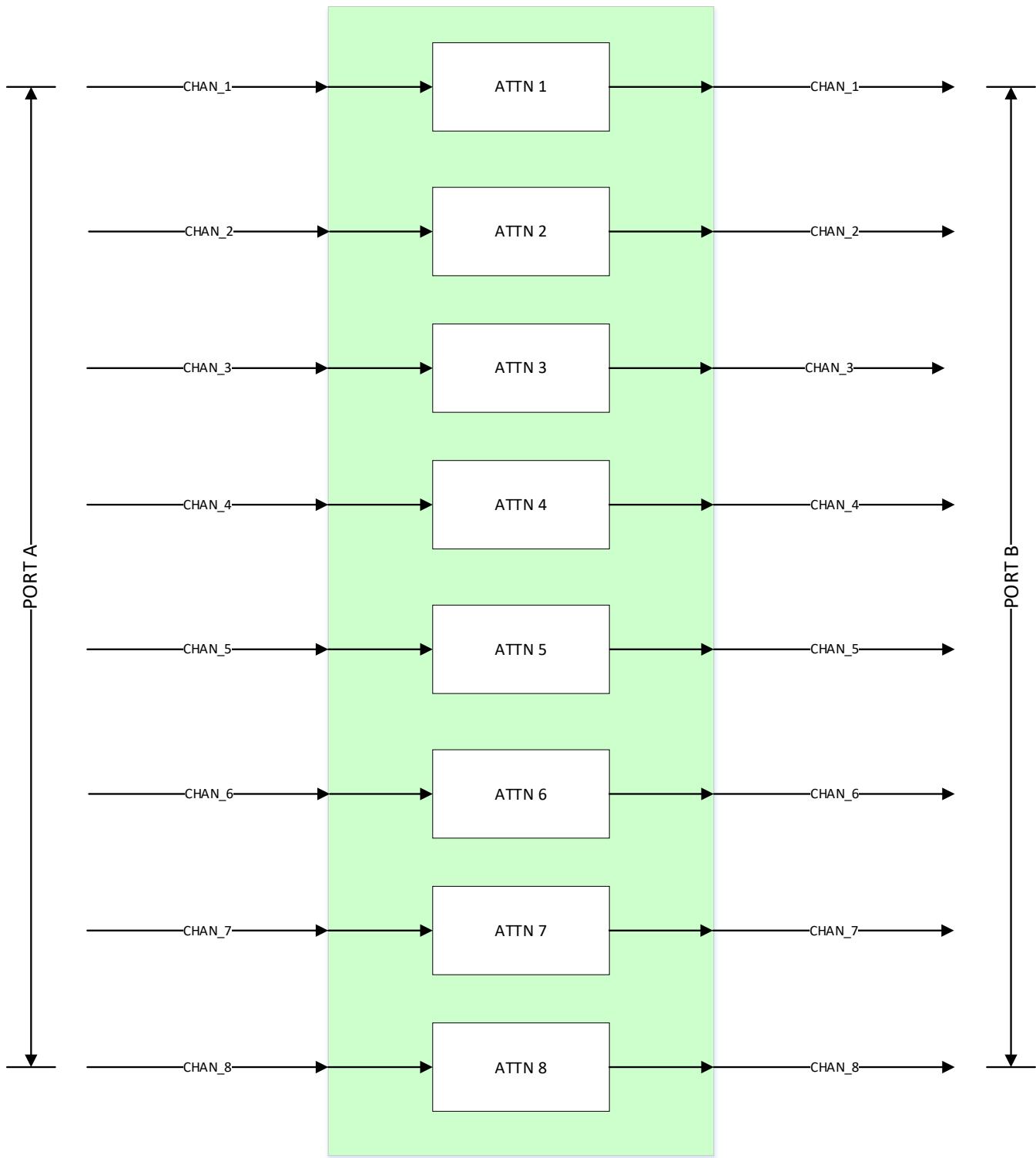


Figure 2: Functional Block Diagram of 4-channel 48xx series module

4. Installation

Figure 2 below shows the front and rear panels of a 440xx 4-Channel Ethernet Attenuator.

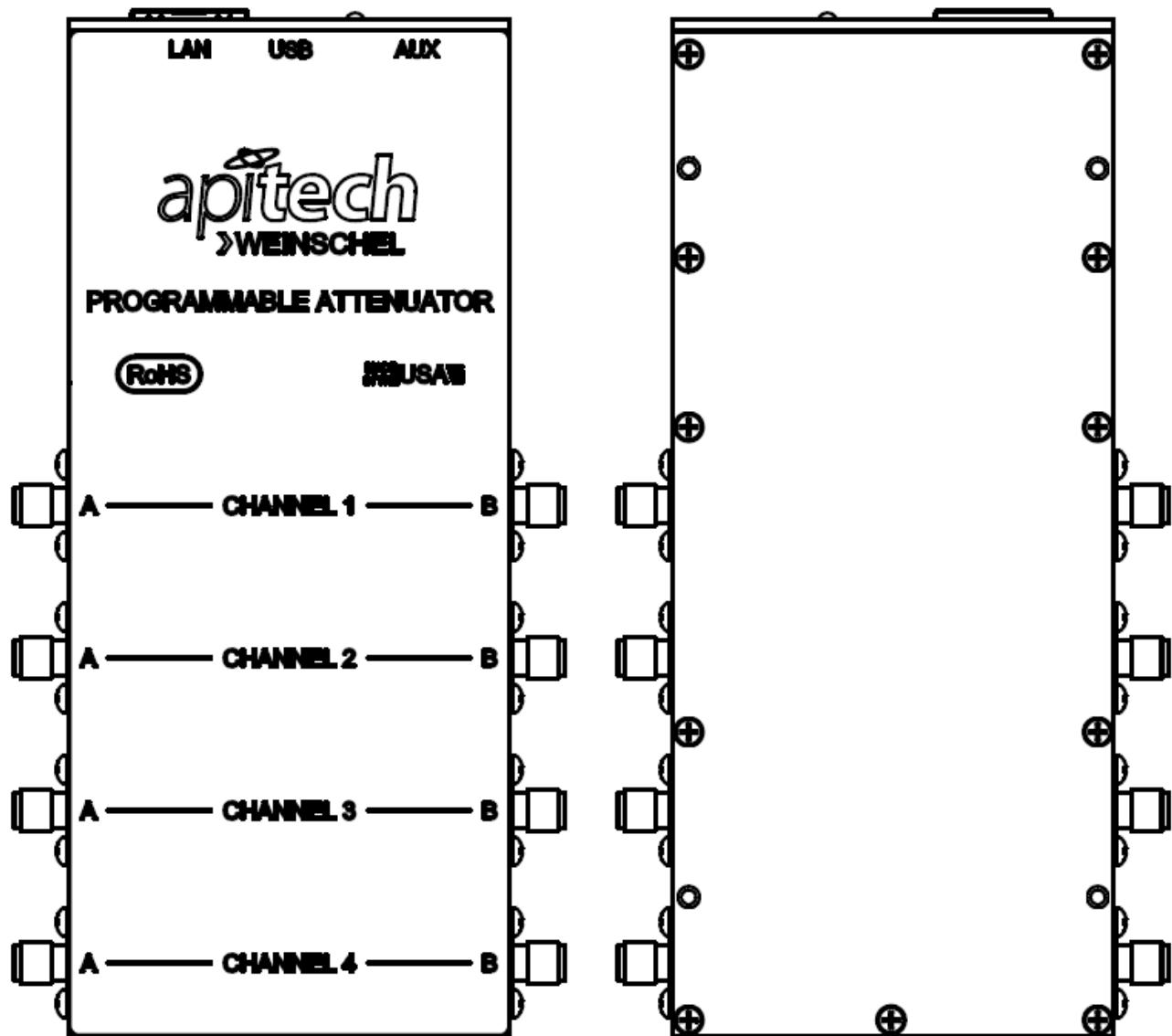


Figure 3: Front and Rear Panels of 44xx 4-Channel Ethernet Attenuator

4.1. Initial Setup

The following initial setup procedures should be performed prior to operating the Model 44xx & 48xx attenuator.

1. Perform an initial inspection of the unit.
 2. Verify the power connection is configured to the correct voltage.
 3. Connect all power, RF and communication cables as required to the unit.

4.2. I/O Control Connections

All external control connectors are located on the rear of the unit. Figure 2 show the location of each connector. Additional details about each connection can be found in Appendix section of this manual.

4.3. RF Port Connectors



To prevent damage to the device do NOT exceed the maximum allowable power level specifications of the unit at any port. Use a calibrated torque wrench to tighten all RF connections.

A typical Model 44xx or 44xx Multi-Channel Programmable Attenuator contains 4 or 8 left and right side SMA connectors for each of the channels respectively. Connectors are per MILSTD-348 interface dimensions and mate nondestructively per MIL-STD-212.

5. Remote Operation

5.1. Serial Console Boot Display

When the system powers up with serial Console mode enabled a series of messages are sent via the serial port showing various installed features, settings, and system status. A typical boot up screen captured with a terminal emulator is shown below.

```
Tuesday, January 20, 2015 1:11:02 PM

API Weinschel 4400 USB-LAN Attn V1.03
firmware: 1012633301B
serialno: 001

RF config: 4, DSA-94P5, 94.5, 0.5 8000MHz
MAC id: 04:91:62:E7:06:A9
IP addr: 0.0.0.0
netmask: 255.255.255.0
gateway: 0.0.0.0

netstat: enabled

DHCP: enabled
AutoIP: enabled

>
**netstat: link down
```

5.2. 10/100BaseT Ethernet

The Ethernet port supports 10/100BaseT operation, with auto-negotiation of the interface speed and duplex mode. LED indicators are provided to indicate network LINK status (green) and TX/RX activity (YELLOW). Supported network protocols include: IP, UDP, TCP, ICMP (ARP and PING), DHCP, AUTOIP, TELNET, and HTTP. The TCP and UDP servers allow connections to be established for general programming purposes. A TELNET server is provided for a command-line interface that implements many of the functions of the serial console CLI, and an HTTP server that allows control via a browser. Additionally, the Microchip Announce protocol is implemented to provide support for the Microchip Ethernet Discovery tool, which is a UDP-based protocol used to detect ethernet devices on the network.

Supported network application protocols and their default ports are summarized below. Any undesired server protocols can be disabled by setting their respective port to 0. For example, SET TELNET SERVER 0 will disable the telnet protocol.

PROTOCOL	DEFAULT PORT
TCP server	10001 (TCP)
UDP server	20000 (UDP)
TELNET server	23 (TCP)
HTTP server	80 (TCP)
ANNOUNCE server	30303 (UDP)
DHCP client	68 (UDP)

IP addressing modes supported include the use of a statically assigned fixed address, or dynamic address assignment using either DHCP or AutoIP. The use of DHCP requires a DHCP server to reside on the network. AutoIP is an address mode that can be used when no DHCP server is available. It automatically allocates an address from the special block of addresses 169.254.1.0 to 169.254.254.255 reserved for link-local addressing. These addresses are only valid on the link that the host is connected to, such as a local network segment or point-to-point connection, and are unrouteable. See the SET IPADDR, SET DHCP, SET AUTOIP, and SHOW IPADDR commands for more information.

The unit provides a TCP server that can be used for control and status of the unit using the same text-based messages used by the serial port. By default, the server is configured to support a single connection and listen on a single port (port 10001), however the number of allowable connections can be changed to support up to 4 simultaneous users. Server settings, such as the number of connections, port number, keepalive timeout, inactivity close timer, and character echoing are programmable by the user (see SET TCP in the command reference). The status of the server(s) can be seen using the SHOW NET TCP command.

A UDP server is also provided that will accept command messages sent via UDP protocol using the same text-based messaging. UDP is a connection-less based protocol that is simpler and has less overhead than TCP. By default, the internal UDP server listens on port 20000, but this can be changed via the SET UDP SERVER command.

The TELNET server communicates using the standard port 23 typically used by the TELNET protocol. The implementation is a reduced-functionality version and does not support the full protocol, but it should function properly with many clients. The server only supports a single connection, does not require any login, and does not support options negotiation. The server does support an inactivity timeout, and allows the use of TELNET NOP commands sent by the client to keep a session open. By default, the timeout is set for 300 seconds (5 minutes), after which the server will automatically close the session if no activity has occurred. Many of these features are configurable by the user (see SET TELNET in the command reference). The status of the server can be seen using the SHOW NET TELNET command. An active TELNET connection may be closed from the telnet application on the client using the 'QUIT' command.

HTTP protocol support allows the user to send commands using HTTP GET. Query responses are returned using plain text. An Internet browser can be used as a console for HTTP control by typing the commands/queries directly into the browser address bar. The basic format of an HTTP command is:

```
http://ADDRESS:PORT/TEXT_COMMAND
```

where ADDRESS is the IP address, PORT is the TCP port number (can be omitted if port 80 is used), and TEXT_COMMAND can be any of the device ASCII text commands. As an example, the following will send an IDN query message to IP address 192.168.1.100 port 80 and display the results in the browser window:

http://192.168.1.100/*IDN?

Various network events generate status messages shown on the serial Console port. These events include TCP and TELNET server connect/disconnect messages and DHCP/AUTOIP address assignment changes. The status messages may be disabled if desired (see SET NETSTAT), but are enabled by default.

For simple device discovery, the Microchip Announce protocol is used. The Microchip Announce protocol is a UDP-based scheme used to detect devices supporting the protocol. The protocol broadcasts UDP packets to port 30303 containing the message, "Discovery: Who is out there?", and supporting devices respond with a UDP packet which provides the device IP and MAC address, as well as other info such as the firmware version. A copy of the Ethernet Device Discoverer application for MS Windows-based systems is provided on the CD supplied with the unit, or can be downloaded from the Microchip website.

5.3. USB

The USB port provides a USB Communications Device Class device (CDC) interface that allows programming via a virtual serial COM port using the same text-based commands as the serial port. For interaction with a terminal emulator, a console mode command-line interface (CLI) is provided for ease of use, and is user-configurable via the USB CONSOLE command. Refer to Firmware and Driver section of this manual for information on installing the USB CDC driver file.

6. Command Operation

Commands are comprised of text-based ASCII strings. The command parser is case-insensitive, so either upper or lower case characters are acceptable. Command parameters may be separated with either an ASCII SPACE char (0x20) or an ASCII COMMA char (0x2E), but the separator character used must be the same within an individual command string. Additional SPACE characters are ignored. Typically, input program messages may be terminated using either an ASCII CR character (0x0D) or an ASCII LF character (0x0A), however this can be changed by using the SET EOS command. Command message strings are limited to 128-characters total, including the terminator. Multiple commands can be included in one message by separating the individual commands with an ASCII SEMICOLON character ';' (0x3B), up to the 128-character message limit. Response messages are terminated differently depending on the source of the command. Response messages sent over the serial and USB CDC ports default to using both a CR (0x0D) and LF (0x0A) to terminate the line, while messages sent via a network TCP or UDP connection default to using a single CR (0x0D) terminator, and messages via GPIB use a single LF (0x0A). The output terminator sequence may be changed using the SET EOS command. A list of supported commands can be seen by typing 'HELP' or '?' at the Console prompt.

The command structure/operation is similar to that used in IEEE 488.2, and includes some of the 488.2 Common Commands such as *IDN?, *RST, *CLS, and *OPC?, in addition to device specific commands. In 488.2, programming commands take one of two forms: a Program message or a Query message. Program messages are used to send commands to the device, while Query messages are used to elicit a response. Query commands are those that contain a '?' character. In general, the device does not generate any response to a program message unless the message contains a valid Query command. (Note that this does not apply when operating in Console mode, or when using some commands such as SHOW which are designed to provide the user general information). You can use this feature to provide a method to synchronize command execution with the controller by appending a Query to the desired command, and waiting for the response. For example, sending "*CLS; *OPC?" will place a "1" in the output queue when the *CLS command has been executed. Query commands that return multiple values will have the values separated by an ASCII COMMA character (0x2E). If multiple Query commands are included in the same message, the individual query responses will be separated with an ASCII SEMICOLON character (0x3B).

An Error Queue is provided that logs the results of command/execution errors in a FIFO fashion. The queue entries can be read using the ERR? command, which returns both an error code and a descriptive text message, such as

101, "invalid command"

When the queue is empty, ERR? returns the message **0, "no error"**. The queue can be emptied by repeatedly sending ERR? until all entries are read from the queue, or via sending the *CLS message. Note: There is a single Error Queue shared by all the command interfaces, such as the network socket connections, TELNET, USB, and the serial Console. Since the TELNET, USB and serial Console interfaces operate in an interactive fashion, if you are using multiple interfaces simultaneously the error messages may not appear on the expected interface. For example, errors generated by messages sent to the TCP server port may be shown on the serial console if it is in active use. Refer to Appendix D for a list of typical error codes and their corresponding messages.

Unless otherwise specified, commands revert to their default setting at system reset/poweron, with the exception of the system setup and configuration commands (see SET). The various SET commands are used to update the settings in non-volatile memory (NVM), and do not typically take effect until the next system reset event unless otherwise noted.

6.1. 488.2 Status Reporting

The unit implements the 488.2 Status Reporting Structure, which for GPIB utilizes the IEEE488.1 status byte with additional data structures and rules. The Status Byte Register can be read with either a serial poll (GPIB operation only) or the *STB? common query command. The Service Request Enable Register (SRE) allows the user to select which bits in the Status Byte Register may cause service requests. A bit value of one indicates that the corresponding event is enabled, while a bit value of zero disables an event. The Service Request Enable Register may be accessed with the *SRE and *SRE? common commands. The Status Byte Register may be cleared with the *CLS common command, with the exception of the MAV bit, which is controlled by the operation of the Output Queue. The SRE Register is set to 0 at power-on, disabling all events.

Status Byte Register/ Service Request Enable Register Formats

D7	D6	D5	D4	D3	D2	D1	D0
	RQS	ESB	MAV				

Bit	Mnemonic	Description
6	RQS	Request Service: This bit, if set, indicates that the device is asserting the SRQ signal.
5	ESB	Event Status Bit: This bit is true when an enabled event in the Event Status Register is true.
4	MAV	Message Available: This bit is true when there is valid data available in the output queue.

The Standard Event Status Register is used to report various IEEE 488.2-defined events. The register contents may be accessed with the *ESR? command. An Event Status Enable Register allows the user to select which bits in the ESR that will be reflected in the ESB summary message bit of the Status Byte Register. The Event Status Enable Register may be accessed with the *ESE and *ESE? common commands. The Event Status Register is cleared by an *ESR? query or *CLS common command. The ESE Register is set to 0 at power-on, disabling all events.

Standard Event Status Register/ Standard Event Status Enable Register Formats							
D7	D6	D5	D4	D3	D2	D1	D0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Bit	Mnemonic	Description
7	PON	Power On This bit indicates that the device has powered-on.
6	URQ	User Request This event bit indicates that a local control is causing a User Request.
5	CME	Command Error The parser has detected a syntax error in the current command.
4	EXE	Execution Error The command could not be properly executed due to an illegal input range, or other inconsistent data.
3	DDE	Device Dependent Error A command could not properly complete due to some device specific error.
2	QYE	Query Error This bit indicates that either an attempt has been made to read data when there was none present, or that data in the Output Queue has been lost.
1	RQC	Request Control The device is requesting control (not implemented).
0	OPC	Operation Complete This bit is generated in response to an *OPC command. It indicates that the unit has completed all pending operations.

The Status Reporting Registers may be used for serial and other asynchronous communications, with certain limitations. The Status Byte Register can only be read via the *STB? query command, as the serial port does not provide for a serial poll operation. Also, as data in the Output Queue is sent automatically during asynchronous operation, the MAV message available bit in the STB serves no purpose.

6.2. Command Access Restrictions

Commands that set or change the unit configuration such as IP address, subnet mask, etc, store their settings in non-volatile memory (NVM). These SET commands are normally available over all interfaces, but can be set to function only via the local serial port using the SET ACCESS command. Note that some commands (such as performing program updates) are allowed only on the serial port regardless of the ACCESS setting.

6.3. Command Execution and Buffering

Typically, simple commands execute in 1-2msecs, however certain commands such as switching an electromechanical relay-based attenuator may take significantly longer than this. During this time, input commands are buffered for later execution. Buffering typically provides space for approximately 20 commands, but this is command and interface dependent.

When operating at fast communication rates it is possible to exceed the buffering ability causing commands to be missed. For interfaces such as RS232, you can use hardware flow control to prevent this from occurring. For other interfaces, you can use a command/query scheme as a synchronization method. A simple example of this would be to append a query to the desired command, and waiting for the response. For example, sending "ATTN 1 10;*OPC?" will place a "1" in the output queue when the ATTN command has been executed.

6.4. Command Reference

In the command descriptions that follow, argument types are described using the following additional conventions to indicate the relative size of the parameter:

Argument Type	Relative Size
Byte	Used to indicate an 8-bit unsigned integer
Word	Used to indicate a 16-bit unsigned integer
Int8	8-bit integer
Int16	16-bit integer
Int32	32-bit integer
String	Character data, including the max number of characters allowable. (ie string8 has a max of 8 chars)

Numeric arguments default to decimal (base 10) notation, but may optionally be provided in hex or binary if appropriate by using a "0x" prefix for hex or "0b" for binary. In addition, commands that accept a '0' or '1' argument will also accept the text strings 'OFF' and 'ON' in place of the numeric parameter. For example, "CONSOLE 1" and "CONSOLE ON" are equivalent.

Required command keywords are shown in CAPITAL letters, and arguments are shown in *italics*. Square brackets '[]' may be used to indicate an optional parameter, for example [select]. Optional parameters, if not supplied by the user, assume the default setting specified in the text.

7. Application Specific Commands

7.1. Attenuation Commands

For the following attenuation control commands there are a number of methods for specifying the attenuator *select* parameter. In the simplest form, *select* specifies a single numeric channel number from 1 to *n*, where *n* is the maximum number of installed attenuation channels (ie 1-12). For users familiar with API 8210A-based controllers, the *select* parameter may also be specified using the string prefix 'AT' along with the channel number (ie AT1. See the ATTN command for more information). In addition to specifying single attenuators, multiple attenuators may be selected by placing them into a named group, and then using that group name as the *select* parameter. When using a group, all attenuators in the group will be set to the same dB value. Currently, the system allows for two groups: a user-defined group and a predefined group named ALL. See the GROUP command for more information.

Attenuation value settings are specified in dB, with up to 2 digits of precision after the decimal point for attenuators that support step sizes of < 1dB. When specifying integral dB values, usage of the decimal point format is strictly option (ie '10' is the same as '10.00'). The attenuation setting must be a multiple of the attenuators intrinsic step size or the command will generate an error. For example, an attenuator that has a stepsize of 0.25 dB will accept settings of 0.25 and 0.5, but will generate an error if set to 0.3. For responses, attenuation values will be formatted to the base precision of the attenuator

Certain attenuator types, such as relay-based models, have switching speed and cycle rate limitations which must be accounted for when programming. Switching speed requirements are built into the command execution time and are always enforced such that if you execute a command sequence such as "ATTN 1 10;*OPC?" the response will not be sent until the attenuator has been programmed and has changed state. Typical switching speeds for these types of attenuators are in the 5-20msec range, depending on the model. For relay-based attenuators, in addition to the switching speed there is also a maximum rate at which commands can be sent to an individual attenuator referred to as the cycling rate. This is a much longer period of time, and is typically in the 100-150msec

range (6-10 operations/second). Note that this delay is on a per attenuator basis, and will only be seen if an attempt is made to reprogram an attenuator before this time limit expires. Solid-state attenuators do not have these limitations, and can be switched at the maximum rate that commands are executed, which is typically in the 1-2ms range. You can use the RFCONFIG? ATTN command to view the parameters associated with the attenuator type installed in the system.

Attenuation commands can have their values cached, and the actual attenuation settings updated upon receipt of a single command. This caching can help reduce the attenuator to attenuator skew that occurs when setting multiple attenuator channels, and can also help eliminate many of the switching speed related execution delays seen when using relay-based attenuators. Attenuator caching is controlled via the TRIG ATTN command. When enabled, caching will typically reduce the channel to channel programming skew to <100usec per channel, which is significantly faster than what is normally obtained when sending individual ATTN commands. Some attenuator models such as the 4205A, 4204, and 4209 series support a hardware trigger function which provides <5usec skew in the attenuation setting across all channels when attenuator caching is enabled.

ATTN

Function: set attenuator

Syntax: ATTN *select setting*

Argument(s): *select* attenuator select 1-*n*, AT1-AT*n*, or a group name
 setting attenuator setting, in dB. *setting*=0-max attenuation value, or MAX

Remarks: This command sets the specified RF attenuator(s) to the dB value provided by *setting*. If *setting* is MAX, then the maximum dB value for that attenuator will be used.

Return Value: none

Example(s):

```
ATTN 1 10          // sets attn 1 to 10 dB
ATTN ALL MAX      // sets all attenuators to their max value
ATTN AT1 15.75    // sets attn 1 to 15.75 dB (8210A mode)
```

ATTN?

Function: read attenuator setting

Syntax: ATTN? *select*

Argument(s): *select* attenuator select 1-*n*, AT1-AT*n*, or ALL

Remarks: This command returns the current setting of the specified attenuator. Specifying ALL will return a comma-separated list of the settings for all attenuators.

Return Value: attenuator setting, in dB

Example(s):

```
ATTN 1 10      // sets attn 1 to 10 dB
ATTN? 1        // read attn 1 setting
10            // returns attn 1 setting (10 dB)
```

SYNC

Function: set programmed sync setting

Syntax: SYNC *n*

Argument(s): *n* 0-1 (OFF/ON)

Remarks: By default, each individual command will automatically generate a SYNC pulse (AUX J2-1). This can be controlled using the SYNC command to enable programmed sync instead of auto sync. Using programmed sync, data will be latched and a SYNC pulse generated when SYNC 0 is executed.

Return Value: none

Example(s):

```
SYNC 1          // enable programmed SYNC
```

```
SYNC 0      // disable programmed SYNC (asserts SYNC output if previously enabled)
```

STEPSIZE**Function:** set attenuator stepsize**Syntax:** STEPSIZE *select setting***Argument(s):** *select* attenuator select 1-*n*, AT1-AT*n*, or a group name
setting attenuator stepsize, in dB. *Setting*=0-max attenuation value**Remarks:** This command sets the attenuation stepsize for the specified attenuator(s). This command can be used with the INCR and DECR commands to change the step value. Specifying a *setting* of 0 sets the stepsize to the intrinsic step value for the attenuator, in effect removing any previous STEPSIZE command.**Return Value:** none**Example(s):**

```
STEPSIZE 1 10      // sets attn 1 stepsize to 10dB
STEPSIZE ALL 0     // sets the stepsize for all attn to their native value
```

STEPSIZE?**Function:** read attenuator stepsize setting**Syntax:** STEPSIZE? *attnno***Argument(s):** *attnno* attenuator select 1-*n* or AT1-AT*n***Remarks:** This command returns the current stepsize of the specified attenuator**Return Value:** attenuator setting, in dB**Example(s):**

```
STEPSIZE 1 10      // sets attn 1 stepsize to 10dB
STEPSIZE? 1         // read attn 1 stepsize setting
```

INCR**Function:** increment attenuator setting**Syntax:** INCR *select***Argument(s):** *select* attenuator select 1-*n*, AT1-AT*n*, or a group name**Remarks:** This command increments the current setting of the specified attenuator(s) by the attenuator's STEPSIZE setting.**Return Value:** none**Example(s):**

```
ATTN 1 5      // sets attn 1 to 5dB
STEPSIZE 1 10// sets attn 1 stepsize to 10dB
INCR 1        // increments attn 1 by the stepsize (5dB + 10dB = 15dB)
```

DECR**Function:** decrement attenuator setting**Syntax:** DECR *select***Argument(s):** *select* attenuator select 1-*n*, AT1-AT*n*, or a group name**Remarks:** This command decrements the current setting of the specified attenuator(s) by the attenuator's STEPSIZE setting.**Return Value:** none**Example(s):**

```
ATTN 1 15      // sets attn 1 to 15dB
STEPSIZE 1 10// sets attn 1 stepsize to 10dB
DECR 1        // decrements attn 1 by the stepsize (15dB - 10dB = 5dB)
```

7.2. General RF Configuration and Status Commands

SHOW STAT

Function: displays current settings

Syntax: SHOW STAT

Argument(s): none

Remarks: This command displays the current settings of the RF hardware. The list will change to reflect the type and number of devices installed in the system.

Example(s):

```
>show stat
ATTN 1: 95.25
ATTN 2: 95.25
ATTN 3: 95.25
ATTN 4: 95.25
```

RFCONFIG

Function: set rf hardware settings

Syntax: RFCONFIG DEFAULT ATTN *db* // set the default attenuator attenuation
RFCONFIG FREQ *Mhz* // set the default frequency

Argument(s): *db* Attenuation value in dB
Mhz Frequency in MHz

Remarks: This command can be used to change the RF configuration to set different default parameters.

Return Value: none

Example(s):

```
RFCONFIG DEFAULT ATTN 1 // sets default attenuation to 1dB
RFCONFIG FREQ 1 // sets default frequency to 1MHz
```

SET RFCONFIG

Function: set rf hardware installation

Syntax: SET RFCONFIG CHAN *numchan* [*numchan2*] // set the number of installed channels
SET RFCONFIG ATTN *type* // set the installed attenuator type

Remarks: This command can be used to change the RF configuration to support different chassis configurations. If the unit supports a single RF device type (ie attenuators only), then the SET RFCONFIG CHAN command only accepts a single parameter. If the unit supports multiple device types (ie both attenuators and switches), then SET RFCONFIG CHAN supports two parameters, with the first specifying the number of attenuators and the second specifying the number of switches.

Return Value: none

Example(s):

```
SET RFCONFIG CHAN 9 // sets number of installed channels to 9
SET RFCONFIG CHAN 5 2 // sets number of attns to 5 and switches to 2
```

RFCONFIG?

Function: read chassis configuration items

Syntax: RFCONFIG? CHAN // returns the number of installed channels
RFCONFIG? ATTN *n* // returns configuration info for attn *n* (model, range, etc)
RFCONFIG? LIST TYPE // returns a list of supported models

Remarks: These commands can be used to retrieve various chassis configuration settings. RFCONFIG? CHAN returns the number of installed devices. If the unit is configured to support both attenuators and switches then this command returns two values (attenuators, switches). Otherwise it returns a single value.

RFCONFIG? ATTN returns the attn type, max attenuation, stepsize, switching speed (msec), cycle rate (msec), and a descriptive string.

RFCONFIG? LIST TYPE returns a list of the supported attn types

Example(s):

```
>RFCONFIG? ATTN 1
4205A-95.5, 95.5, 0.5, 0, 0, "95.5dB/0.5dB, 0.2-6GHz"
```

```
>RFCONFIG? CHAN
5, 2 // unit supports 5 attenuators and 2 switches
```

7.3. 488.2 Common Commands

*CLS

Function: clears the error status

Syntax: *CLS

Argument(s): none

Remarks: This function is used to clear the Error Queue

Return Value: none

Example(s):

```
*CLS
```

*IDN?

Function: Reads the system identification information

Syntax: *IDN?

Argument(s): none

Remarks: This function is used to read the system identification info, which is a string consisting of the following data: manufacturer, model, serial number, and firmware version.

Return Value: *idstr* string id info

Example(s):

```
*IDN?
API Weinschel, 4400, 001, V1.03
```

*OPC?

Function: Operation complete query

Syntax: *OPC?

Argument(s): none

Remarks: This function loads a '1' into the output queue when the Program Message Unit is executed. Its primary use is to provide an indication of command completion by including the command as the last one in a series of commands. It can be useful to synchronize operation and to prevent input buffer overflow.

Return Value: 1 integer constant command completed

Example(s):

```
CMD1 1; CMD2 2; *OPC?
1 // sends a '1' when the three commands have been executed
```

*ESR?

Function: Event Status Register query

Syntax: *ESR?

Argument(s): none

Remarks: This function reads the 488.2 Event Status Register. Reading the register also clears it.

Return Value: *int8* integer status register

Example(s):

```
*ESR?  
32          // indicates a Command Error
```

RST*Function:** Performs a device application level reset.**Syntax:** *RST**Argument(s):** none**Remarks:** This function is used to reset the device application settings. For a full device reset, see the REBOOT command.**Return Value:** none**Example(s):**

```
*RST
```

TST?*Function:** Self-test query**Syntax:** *TST?**Argument(s):** none**Remarks:** This function performs an internal self-test. Upon completion, the results of the test are loaded into the output queue.**Return Value:** *testresults* integer '0' indicates test passed. Non-zero indicates test failed.**Example(s):**

```
*TST?
```

```
0          // returns a '0' when the test completes successfully
```

ERR?**Function:** Read the Error Queue**Syntax:** ERR?

```
SYST:ERR?
```

Argument(s): none**Remarks:** This function returns the last entry in the error status queue, and a string description of the error code. Repeating the command will return the next entry, until the error queue is empty and returns a zero. The error queue may be cleared via the *CLS command. Note that when using the command-line interface the Error Queue contents are automatically displayed after each command prior to issuing the CLI prompt.**Return Value:** error number, "error description"**Example(s):**

```
ERR?
```

```
101, "invalid command"
```

```
ERR?
```

```
0, "no error"
```

7.4. Setup and Configuration Commands

NOTE: The SET commands are used to update settings which are stored in non-volatile memory (NVM), and do not typically take effect until the next system restart event (power cycle or REBOOT) unless otherwise noted. The settings listed here are dependent on the installed hardware, so not all settings are available on some implementations.

The FACTORY PRESET command may be used to set all NVM settings to their factory default values. The default settings will take effect on the next restart.

Access to SET commands may be restricted to the serial port only. This is controlled via the SET ACCESS command.

FACTORY RESET**Function:** Sets all NVM parameters to their factory default setting.**Syntax:** FACTORY RESET

FACTORY RESET VERIFY

Argument(s): none**Remarks:** The FACTORY RESET command restores all NVM parameters to the factory default. The default settings will take effect on the next restart. FACTORY RESET VERIFY may be used to verify the integrity of the preset memory. A return value of 0 indicates the integrity check passed, while any other result indicates a failure.**Example(s):**

```
FACTORY RESET VERIFY
> 0
```

SET ACCESS**Function:** restrict access to SET commands**Syntax:** SET ACCESS *onoff***Argument(s):** *onoff* 0=access disabled (serial port only), 1=access enabled on all interfaces**Remarks:** This function enables/disables access to the NVM SET commands. When access is disabled, the SET commands are only functional over the local serial port, and will generate an error if used over other interfaces. The default setting is 1, enabling access to SET commands over all interfaces.**SET EOS****Function:** sets the Program Message Terminator and/or Response Message Terminator end of string characters**Syntax:** SET EOS *interface inout val***Argument(s):** *interface* protocol selection SERIAL, USB, GPIB, TCP, UDP, or ALL*inout* PMT (input) or RMT (output)*val* word, eos characters**Remarks:** This function sets the input Program Message Terminator (PMT) or the output Response Message Terminator (RMT) sequences. Each communications port/protocol can have separate definitions. The *val* parameter specifies the character sequence used, and can specify up to two characters, typically as a hex word high byte-low byte pair. Common definitions for the terminators include the ASCII CR (0x0D) and LF (0x0A) characters. A single character may be specified either by using 0 for the high byte, such as 0x000D, or by only specifying a single character (ie 0x0D). On input, the message will terminate when either of the two character codes are received, while for output the characters are sent low byte then high byte, unless it is specified as 0. Note that the serial CONSOLE and network TELNET servers are excluded from this selection and always use a fixed CRLF (0x0A0D) sequence. The current settings may be viewed using the SHOW EOS command.**Return Value:** none**Example(s):**

```
SET EOS SERIAL PMT 0x0A0D // set serial input to terminate on either CR or LF
SET EOS SERIAL RMT 0x0A0D // set serial output sequence as CR-LF
SET EOS USB PMT 0x0A0D // set usb input to terminate on either CR or LF
SET EOS USB RMT 0x0A0D // set usb output sequence as CR-LF
SET EOS TCP PMT 0x0A0D // set tcp socket input to terminate on CR or LF
SET EOS TCP RMT 0x0D // set tcp output sequence as a single CR character
SET EOS UDP PMT 0x0A // set udp socket input to terminate on LF only
SET EOS UDP RMT 0x0D // set udp output sequence as a single LF character
```

7.4.1 Network

SET IPADDR

Function: Sets the network IP address/mode

Syntax: SET IPADDR [*ipaddr* | DHCP | AUTOIP]

Argument(s): *ipaddr* static IP address, in the form DDD.DDD.DDD.DDD

DHCP selects DHCP address mode (default)

AUTOIP selects AUTOIP mode

Remarks: This function sets the default IP address mode, allowing the choice between static or dynamic modes. There is some interaction between the various settings, but typically selecting one mode disables the others as follows:

Static IP

Setting a static fixed IP address automatically disables DHCP and AutoIP operation.

DHCP

Setting DHCP mode will enable both the DHCP and AutoIP modes. The existing static IP address (if any) will be erased. DHCP takes preference over any other selected mode. If the system is unable to obtain an address from a DHCP server on the network, it will switch over to AutoIP mode, where it will attempt to assign a link-local address.

AutoIP

Setting AUTOIP mode will enable the AutoIP function and disable DHCP operation for networks where a DHCP server is not available. The existing static IP address (if any) will be erased.

You can also use the SET DHCP and SET AUTOIP commands to combine modes and override the default addressing mode operation selected by this command. If doing so, you should use the SET IPADDR command prior to using SET DHCP or SET AUTOIP, as it has precedence. For example, you can use the SET IPADDR *ipaddr* to set a fixed IP, followed by SET DHCP ON to enable DHCP. The system would attempt to use DHCP, and if unable to obtain an address would use the static IP address *ipaddr*. Likewise, you can use SET IPADDR DHCP followed by SET AUTOIP OFF, in which case the system would only use DHCP and would never switch over to AutoIP mode. You can use the SHOW IPADDR command to view the current address in use, as well as the status of the DHCP and AUTOIP clients.

Return Value: none

Example(s):

```
SET IPADDR 10.0.0.2          // sets static IP, disables DHCP and AUTOIP
SET IPADDR DHCP              // enables DHCP (and AutoIP)
SET IPADDR AUTOIP            // enables AutoIP (disables DHCP)
```

SET DHCP

Function: DHCP client control

Syntax: SET DHCP *enable*

Argument(s): *enable* byte, 0-1 (or OFF/ON)

Remarks: This function can be used to selectively enable or disable the DHCP client. A value of 0 (or OFF) disables DHCP, while any other value (or ON) enables DHCP.

Return Value: none

Example(s):

```
SET DHCP 1                  // enable DHCP
SET DHCP ON                 // enable DHCP
SET DHCP OFF                // disables DHCP
```

SET AUTOIP**Function:** AutoIP client control**Syntax:** SET AUTOIP *enable***Argument(s):** *enable* byte, 0-1 (or OFF/ON)**Remarks:** This function can be used to selectively enable or disable the AutoIP client. A value of 0 (or OFF) disables AutoIP, while any other value (or ON) enables AutoIP.**Return Value:** none**Example(s):**

```
SET AUTOIP 1          // enable AutoIP
SET AUTOIP ON         // enable AutoIP
SET AUTOIP 0          // disables AutoIP
```

SET NETMASK**Function:** Sets the network IP address subnet mask**Syntax:** SET NETMASK *ipmask***Argument(s):** *ipmask* subnet mask, in the form DDD.DDD.DDD.DDD**Remarks:** This function sets the default IP subnet mask used when static IP addressing is selected. The default value is 255.255.255.0**Return Value:** none**Example(s):**

```
SET NETMASK 255.255.255.0
```

SET GATEWAY**Function:** Sets the network Gateway IP address**Syntax:** SET GATEWAY *ipaddr***Argument(s):** *ipaddr* IP address, in the form DDD.DDD.DDD.DDD**Remarks:** This function sets the default gateway/router IP address. Network packets that have a destination not reachable by the current IP configuration are sent to this address. The default value is 0.0.0.0, which disables the gateway function.**Return Value:** none**Example(s):**

```
SET GATEWAY 10.0.0.100
```

SET ETH SPEED**Function:** Sets the Ethernet PHY speed and duplex**Syntax:** SET ETH SPEED [auto|10|100] [DUPLEX half|full]**Remarks:** This function sets the Ethernet PHY auto-negotiation settings**Return Value:** none**Example(s):**

```
SET ETH SPEED AUTO          // enable auto-negotiation
SET ETH SPEED 10 DUPLEX HALF // set fixed 10Mb half-duplex
```

SET TCP CONNECT**Function:** Sets the number of TCP server connections**Syntax:** SET TCP CONNECT *numconnect***Argument(s):** *numconnect* max number of connections, 1-4**Remarks:** This function sets the maximum number of allowed simultaneous connections (users) that are supported by the TCP server. The default is 1, allowing a single user.**Return Value:** none**Example(s):**

```
SET TCP CONNECT 4          // sets the server to allow up to 4 users
```

SET TCP SERVER

Function: Sets the TCP server port number

Syntax: SET TCP SERVER *portno*

Argument(s): *portno* server port, 0-65535

Remarks: This function sets the port number used to communicate with the internal TCP server. Setting *portno* to 0 disables the server. The default TCP server port is 10001.

Return Value: none

Example(s):

```
SET TCP SERVER 1024 // sets the server to listen for connections on port 1024
```

SET TCP KEEPALIVE

Function: Sets the TCP keepalive rate

Syntax: SET TCP KEEPALIVE *tout*

Argument(s): *tout* keepalive timer value, in seconds (0, 10-7200)

Remarks: This function sets the value of the TCP keepalive timeout parameter. The TCP server uses this setting in order to keep a socket open by periodically sending keepalive packets during periods of inactivity. The value can be set for 10 to 7200 seconds (2 hours), or 0 to disable the keepalive function. The default is 30 seconds.

Return Value: none

Example(s):

```
SET TCP KEEPALIVE 60 // sets the keepalive timer to 60 seconds
```

SET TCP TIMEOUT

Function: Sets the TCP server inactivity timeout

Syntax: SET TCP TIMEOUT *tout*

Argument(s): *tout* inactivity timer value, in seconds (0 - 60)

Remarks: This function sets the value of the TCP server inactivity timeout parameter. The TCP server uses this setting in order to automatically close a connection if the client is inactive for a period of time. The value can be set for 0 to 60 seconds, with 0 (or OFF) disabling the inactivity timeout function. The default is 0 (inactivity timeout disabled).

Return Value: none

Example(s):

```
SET TCP TIMEOUT 10 // close connection if host is inactive for 10 seconds
```

SET TCP ECHO

Function: Sets the TCP server character echoing

Syntax: SET TCP ECHO *onoff*

Argument(s): *onoff* byte, 0-1 (or OFF/ON)

Remarks: This function controls the setting of character echoing for the TCP server(s). With echo on, the server echos each received character back to the sender on a character by character basis, while with the setting off no such echoing occurs. This is useful for testing connectivity, but can result in a large number of packets transactions and degrade performance. By default, TCP echo is OFF.

Return Value: none

Example(s):

```
SET TCP ECHO OFF // disables character echoing  
SET TCP ECHO 1 // enables character echoing
```

SET UDP SERVER

Function: Sets the UDP server port number

Syntax: SET UDP SERVER *portno*

Argument(s): *portno* server port, server port, 0-65535

Remarks: This function sets the port number used to communicate with the internal UDP server. Setting *portno* to 0 disables the server. The default UDP server port is 20000.

Return Value: none

Example(s):

```
SET UDP SERVER 1024 // sets the server to listen for messages on port 1024
```

SET TELNET SERVER

Function: Sets the TELNET server port number

Syntax: SET TELNET SERVER *portno*

Argument(s): *portno* server port, server port, 0-65535

Remarks: This function sets the port number used to communicate with the internal TELNET server. Setting *portno* to 0 disables the server. The default TELNET port is 23.

Return Value: none

Example(s):

```
SET TELNET SERVER 23 // set the server to listen for messages on port 23
```

SET TELNET

Function: Set TELNET server controls

Syntax: SET TELNET ECHO *onoff* // local server echo
 SET TELNET OPTNEG *onoff* // TELNET options negotiation
 SET TELNET KEEPALIVE *onoff* // TELNET NOP keepalive
 SET TELNET LOGIN *onoff* // require login
 SET TELNET TIMEOUT *secs* // session inactivity timeout

Argument(s): *onoff* byte, 0-1 (or OFF/ON)

Remarks: This function controls various settings of the TELNET server operation. The current settings can be viewed using the SHOW NET TELNET command. Note that there are two TELNET server implementations available, a full and a reduced-functionality version, and not all parameters are supported by the reduced version. The default settings are: echo on, optneg off, NOP keepalive on, logon off, and an inactivity timeout of 300 seconds.

Return Value: none

SET HTTP SERVER

Function: Sets the HTTP server port number

Syntax: SET HTTP SERVER *portno*

Argument(s): *portno* server port, server port, 0-65535

Remarks: This function sets the port number used to communicate with the internal HTTP server. Setting *portno* to 0 disables the server. The default HTTP server port is 80.

Return Value: none

Example(s):

```
SET HTTP SERVER 80 // set the server to listen for messages on port 80
```

SET ANNOUNCE SERVER

Function: Sets the ANNOUNCE server port number

Syntax: SET ANNOUNCE SERVER *portno*

Argument(s): *portno* server port, server port, 0-65535

Remarks: This function sets the port number used to communicate with the UDP-based ANNOUNCE server. Setting *portno* to 0 disables the server. The default ANNOUNCE port is 30303.

Return Value: none

Example(s):

```
SET ANNOUNCE SERVER 30303
```

SET NETSTAT**Function:** network status message events control**Syntax:** SET NETSTAT *enable***Argument(s):** *enable* byte, 0-1 (or OFF/ON)**Remarks:** This function can be used to control the display of network status messages on the serial console, including link up/down and port connect/disconnect messages. A value of 0 (or OFF) disables messages, while any other value (or ON) enables them. The default setting is on.**Return Value:** none**Example(s):**

```
SET NETSTAT 1           // enables logging of network events to the console
```

Example NETSTAT messages:

```
**netstat: link up
**netstat: link down
**netstat: port 23: socket 2 connected to 10.0.0.101
**netstat: port 23: disconnected
```

IPCONFIG?**Function:** return network settings**Syntax:** IPCONFIG?**Argument(s):** none**Remarks:** This query command returns the current network settings, including the IP address, subnet mask, gateway address, DHCP enable, AutoIP enable, TCP server port, UDP server port, and the HTTP server port.**Example(s):**

```
IPCONFIG?
192.168.1.1, 255.255.255.0, 192.168.1.100, 1, 1, 10001, 20000, 80
```

MACADDR?**Function:** return MAC address**Syntax:** MACADDR?**Argument(s):** none**Remarks:** This command returns the MAC address of the unit.**Example(s):**

```
MACADDR?
04:91:62:E7:06:A9
```

PING**Function:** Sends ICMP ECHO packets**Syntax:** PING *ipaddr***Argument(s):** *ipaddr* destination IP address, in the form DDD.DDD.DDD.DDD**Remarks:** This function performs a ping of the specified network address. It is primarily for console usage**Example(s):**

```
>ping 10.100.103.113
pinging 10.100.103.113...
reply time: 1 ms
reply time: 1 ms
reply time: 1 ms
reply time: 1 ms
```

7.5. Show Commands

The SHOW commands provide a method to view a variety of system settings and information. They are primarily meant for CLI usage such as the console mode or telnet, as the contents are system dependent and may change depending on the current mode, settings, and installed implementation features.

SHOW EOS

- Function:** Shows a summary of the EOS PMT and RMT message terminator settings
Example(s):

```
>show eos
serial pmt: 0x0A0D
serial rmt: 0x0A0D
usb pmt: 0x0A0D
usb rmt: 0x0A0D
tcp pmt: 0x0A0D
tcp rmt: 0x000D
udp pmt: 0x0A0D
udp rmt: 0x000D
```

SHOW SET

- Function:** Shows a summary of various configuration SET command settings
Example(s):

```
>show set

modelno: 4400
serialno: 00

access: 1
console: 1
usb conf: 0x07
usb protocol: 1
usb pmt: 0x0A0D
usb rmt: 0x0A0D
usb delay: 100
net conf: 0x07
IP addr: 0.0.0.0
netmask: 255.255.255.0
gateway: 0.0.0.0
addr conf: 0x07 DHCP AutoIP
phy mode: 0x1F
tcp keepalive: 30
tcp server conf: 0x01
tcp port: 10001
tcp timeout: 0
tcp connections: 1
tcp pmt: 0x0A0D
tcp rmt: 0x000D
udp server conf: 0x01
udp port: 20000
udp pmt: 0x0A0D
udp rmt: 0x000D
telnet conf: 0x07
telnet port: 23
telnet timeout: 300
```

```
http conf: 0x01
http port: 80
http timeout: 10
announce conf: 0x03
announce port: 30303

attn type: 3
attn count: 4
attn delay: 65535
attn settle: 8000
```

SHOW USB

Function: Shows USB settings

Example(s):

```
>show usb
```

```
firmware: 19311459401C
protocol: CDC
VID: 0x25EA
PID: 0x206C
ver: 2.00
stat: 0xC1
```

SHOW NET

Function: Shows general network settings

Example(s):

```
>show net
```

```
link stat: up
phy speed: 100
phy duplex: half
autoneg mode: 0x1F auto, 100, 10, full, half

MAC id: 04:91:62:E7:06:A9
IP addr: 192.168.1.1
netmask: 255.255.255.0
gateway: 192.168.1.100

netstat: enabled

DHCP: enabled
AutoIP: enabled
```

SHOW IPADDR

Function: Shows IP address mode and status

Example(s):

```
// Example #1: SET IPADDR DHCP
// status: no DHCP server found, using AUTOIP
>show ipaddr
```

```
IP address : 169.254.127.57
subnet mask: 255.255.0.0
default IP : 0.0.0.0
```

```

DHCP client: enabled
server IP: none detected
addr stat: not bound

AutoIP client: enabled
addr stat: bound

// Example #2: SET IPADDR 10.0.0.2, SET DHCP ON
// status: no DHCP server found, using static IP
>show ipaddr

IP address : 10.0.0.2
subnet mask: 255.255.255.0
default IP : 10.0.0.2

DHCP client: enabled
server IP: none detected
addr stat: not bound

AutoIP client: disabled

// Example #3: SET IPADDR 10.0.0.2, SET DHCP ON
// status: DHCP server detected, using DHCP
>show ipaddr
IP address : 192.168.0.2
subnet mask: 255.255.255.0
default IP : 10.0.0.2
DHCP client: enabled
server IP: 192.168.0.1
addr stat: bound
AutoIP client: disabled

```

SHOW NET TCP**Function:** Shows TCP server settings/status**Example(s):**

```

>show net tcp

tcp server port: 10001
tcp keepalive: 30
tcp echo: off
tcp timeout: 0

server connections: 1 (4 max)
port 10001: socket 0 connected to 192.168.1.100

```

SHOW NET UDP**Function:** Shows UDP server settings/status**Example(s):**

```

>show net udp

udp server port: 20000

```

SHOW NET TELNET**Function:** Shows TELNET server settings/status**Example(s):**

```
>show net telnet

telnet server port: 23
timeout: 300
flags:
echo: 1
keepalive: 1
neg options: 0
login: 0

max connections: 1
port 23: no connection
```

SHOW NET HTTP**Function:** Shows HTTP server settings/status**Example(s):**

```
>show net http

http server port: 80
timeout: 10

max connections: 2
port 80: no connection
port 80: no connection
```

SHOW NET ANNOUNCE**Function:** Shows ANNOUNCE server settings/status**Example(s):**

```
>show net announce
announce port: 30303
connect: 1
```

7.6. MISC Commands**CONSOLE****Function:** Console mode enable**Syntax:** CONSOLE *mode***Argument(s):** *mode* byte 0, 1, 2, 3 or OFF, ON, ENABLE, DISABLE**Remarks:** This function enables/disables the serial port console mode command-line interface and optionally updates the nvm setting. Setting *mode*=0 turns console off, *mode*=1 turns console on, *mode*=2 enables the console, and *mode*=3 disables the console. Modes 0 and 1 (OFF and ON) update the nvm setting, while modes 2 and 3 (ENABLE and DISABLE) do not. Note: This setting may be overridden by a hardware DIP switch located on the controller assembly.**Return Value:** none**Example(s):**

CONSOLE ON	// turns on the console and updates nvm setting
CONSOLE ENABLE	// turns on console for this session only
CONSOLE 0	// turns off the console and updates nvm setting
CONSOLE DISABLE	// turns off console for this session only

CONSOLE?**Function:** Console mode query**Syntax:** CONSOLE?**Argument(s):** none**Remarks:** This function returns the serial console mode nvm and DIP switch settings**Return Value:** *nvm, dipsw* integer, integer**Example(s):**

CONSOLE?

1, 0

// console nvm flag = 1, DIP switch = 0

CMDSTATS?**Function:** Returns command statistics**Syntax:** CMDSTATS?**Argument(s):** none**Remarks:** This command returns the command statistics of the unit**Return Value:** number <total cmds, errors>**Example(s):**

CMDSTATS? // returns cmd stats

8, 2

USB CONSOLE**Function:** USB Console mode enable**Syntax:** USB CONSOLE *mode***Argument(s):** *mode* byte 0, 1, 2, 3 or OFF, ON, ENABLE, DISABLE**Remarks:** This function enables/disables the USB CDC console mode command-line interface and optionally updates the nvm setting. Setting *mode*=0 turns console off, *mode*=1 turns console on, *mode*=2 enables the console, and *mode*=3 disables the console. Modes 0 and 1 (OFF and ON) update the nvm setting, while modes 2 and 3 (ENABLE and DISABLE) do not.**Return Value:** none**Example(s):**USB CONSOLE ON // turns on the USB console and updates nvm setting
USB CONSOLE ENABLE // turns on USB console for this session only
USB CONSOLE 0 // turns off the USB console and updates nvm setting
USB CONSOLE DISABLE // turns off USB console for this session only**DSA****Function:** Sets attenuation of individual attenuator ICs**Syntax:** DSA *chan b1 b2 b3***Argument(s):** *chan* channel select*b1 b2 b3* attenuator IC setting, in dB. *setting*=0-max attenuation value, or MAX**Remarks:** This command sets the attenuation of each individual IC on a channel.**Return Value:** none**Example(s):**

DSA 1 16 0 0 // sets the first channel, first attenuator IC to 16dB

DELAY**Function:** Delays execution (Pause)**Syntax:** DELAY *msecs***Argument(s):** *msecs* word, 0-65535 in msecs**Remarks:** This command pauses execution for the specified time in msecs.**Return Value:** none**Example(s):**

```
ATTN 1 0; DELAY 100; ATTN 1 10 // waits 100 msecs between attn commands
```

REBOOT**Function:** system reset**Syntax:** REBOOT**Argument(s):** none**Remarks:** This command performs a system reboot, similar to a poweron reset.**Return Value:** none**Example(s):**

REBOOT

RUN**Function:** run an auxiliary program function**Syntax:** RUN *cmd***Argument(s):** *cmd* command function

LOADER

Remarks: This command runs an external function, such as the Flash Bootloader for downloading program updates**Return Value:** none**Example(s):**

RUN LOADER // invoke the flash bootloader for update

TEMP?**Function:** reads internal temperature sensor**Syntax:** TEMP? [*sensor*]**Argument(s):** *sensor* byte optional sensor number, 0-2. default=0 (internal)**Remarks:** This function returns the current temperature (degrees C) and the max temperature.

Resolution is 0.5 degrees

Return Value: degC, max_degC**Example(s):**

TEMP?

30.0, 35.5

TIME?**Function:** reads execution time**Syntax:** TIME?**Argument(s):** none**Remarks:** This function returns the execution time from the start of the command message, in msecs.**Return Value:** msecs integer32**Example(s):**CMDSTATS 0; TIME?; DELAY 10; TIME
1;11**TIMESTAMP****TIMESTAMP?****Function:** sets/reads timestamp timer**Syntax:** TIMESTAMP [*clear*]

TIMESTAMP?

Argument(s): *clear* byte flag**Remarks:** These commands can be used to time multiple events spanning a long time period. TIMESTAMP records the current system tick counter, and TIMESTAMP? returns the number of ticks since the last TIMESTAMP command. The tick counter is a 1ms 32-bit value, and will rollover when the max

count is reached. The command **TIMESTAMP 0** allows you to reset any current timestamp, after which **TIMESTAMP?** will return the current system tick counter directly.

Return Value: ticks integer32

Example(s):

```
>TIMESTAMP?          // read current system ticks (sys uptime)
1193502
>TIMESTAMP; TIMESTAMP? // set a new timestamp and read the time
1
>TIMESTAMP 0          // reset timestamp
>TIMESTAMP?          // returns current system tick counter
1214141
```

REPEAT

Function: Enables command repetition/looping

Syntax: REPEAT *count*

Argument(s): *count* word, 1-65535

Remarks: This function causes the remainder of the current command message to be repeated *count* number of times. Any commands included prior to REPEAT are executed a single time.

Return Value: none

Example(s):

```
ATTN 1 0; REPEAT 50; INCR 1; DELAY 100 // repeats INCR and DELAY 50 times
```

SYSTEST

Function: performs a low-level system test

Syntax: SYSTEST [*select*]

Argument(s): *select* test select, optional (varies by platform)

WDT // test the watchdog timer function

STACK // test the stack over/underflow reset function

ALL // test aux hardware board

FP // front-panel (if installed)

Remarks: This command performs a low-level test on the selected hardware. NOTE: These tests should be used with great caution as it may exercise attached RF devices/hardware. You must cycle power/REBOOT after performing SYSTEST to restore normal operation.

Return Value: various status messages

HELP

Function: Displays a list of supported commands

Syntax: HELP [/level]

? [/level] (console mode enabled)

Argument(s): /*level* help level, ALL or 1-3

Remarks: This command will display a list of all the commands with a short description of their function. The list is divided into multiple levels, with each level including more commands. HELP ALL will display a list of all supported commands.

Return Value: none

Example(s):

```
HELP          // displays main application level commands
HELP ALL      // displays all commands
```

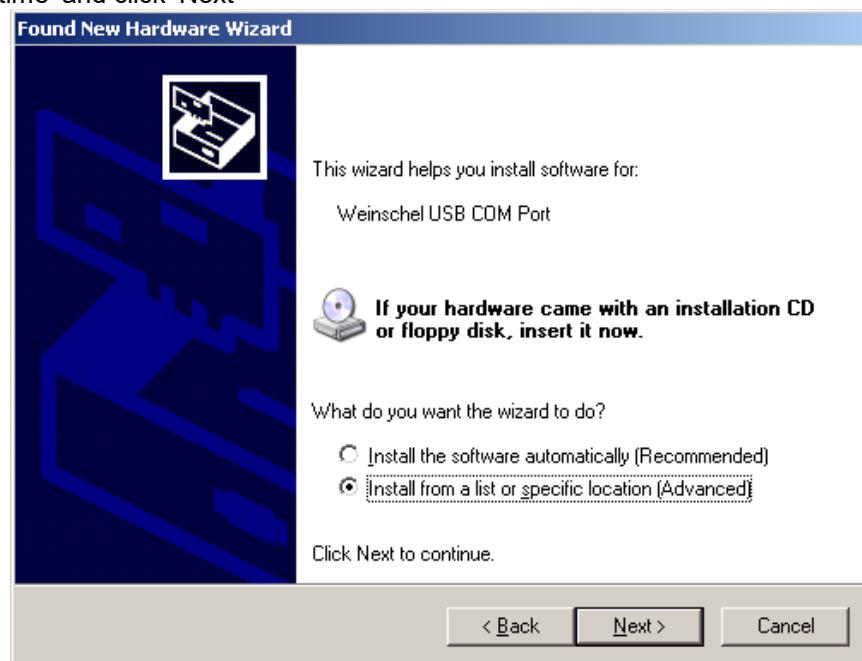
8. Firmware and Drivers

8.1. Installing Weinschel USB CDC Driver

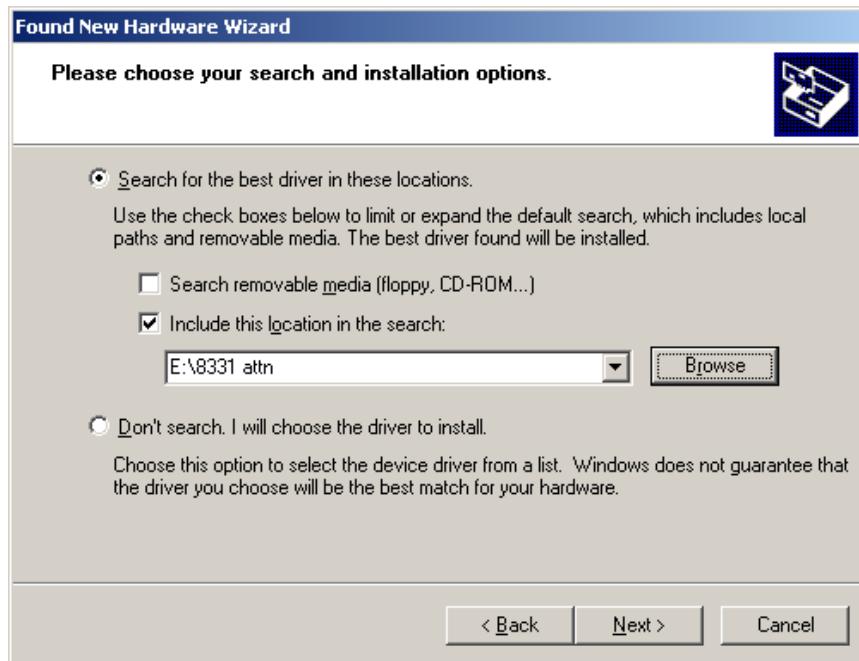
Connect a USB cable from the unit to a USB port on the PC. Windows should detect the device and the New Hardware wizard should run.



Select 'No, not this time' and click 'Next'



Select 'Install from a specific location' and click 'Next'



Using the 'Browse' button, navigate to the drive/folder containing the AW83xxCDC.inf file.

Select 'Next'

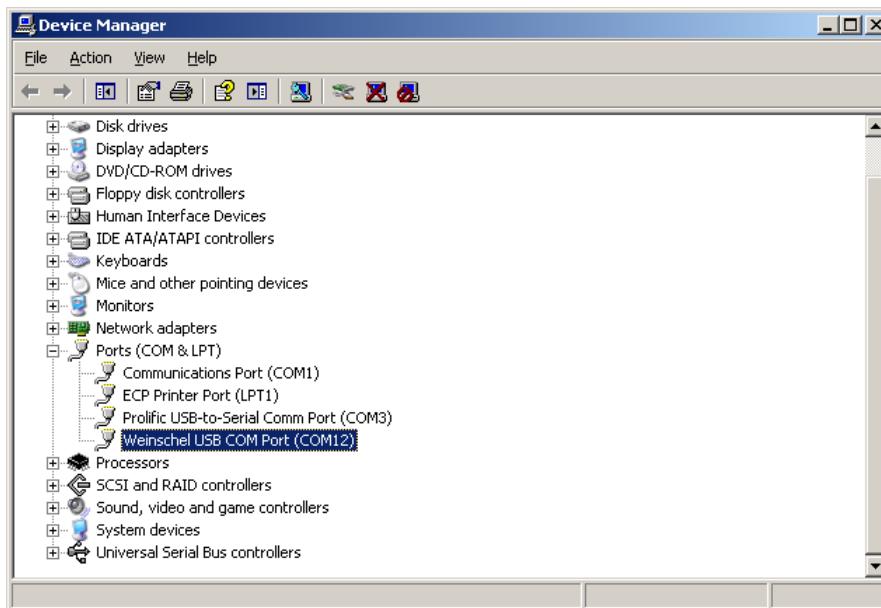


You will get a message stating that the driver is not signed. Select 'Continue Anyway'



The hardware wizard should complete. Select 'Finish', and you should get a message that your new hardware is installed and ready to be used.

To verify that the driver is installed properly, go to Device Manager and you should see an entry under Ports (COM & LPT) for Weinschel USB COM Port, and the assigned COM port number



9. Error Code and Messages

Error Code	Error Descriptions
General Hardware/System Errors: 0-99	
0	No Error
1	General Error
2	Comm Error
3	Reset Event
4	Timer Allocation Error
5	SMBUS Error
6	Temperature Sensor Error
7	MCP IOX Error
8	SPI NVM Error
9	XSMBus Error
10	XSMBus Fault
11	Network Hardware Error
12	MAC Address Error
13	LCD Initialization Error
14	MCP2515 Hardware Error
15	CANBus RXB Error
16	CANBus TXB Error
17	USB Hardware Error
18	GPIB Hardware Error
Parser Errors: 1XX	
100	Parser Error
101	Invalid Command
102	Argument Error
103	Command Unsupported
104	Input Command Length
105	Command Not Found
106	Syntax Error
Execution Errors: 2XX	
200	Execution Error
201	Input Buffer Error
NVM and MPFS Storage Errors: 3XX	
300	NVM Error
301	NVM Format Error
302	NVM Defaults Set
303	MPFS Header Error
304	File Open Error
305	SPI NVM Format Error
Application Level Errors: 4XX	
400	Application Error
401	Hardware Failure
402	Not Installed
403	Hardware Lockout
Protocol Errors: 5XX	
500	Protocol Error
501	TELNET Timeout

10. Instrument Security Procedures for Secure Environments

10.1. Clearing and Sanitization

Clearing is the process of eradicating the data so that it can no longer be retrieved using the standard instrument interfaces. Clearing is typically used when the instrument is to remain in an environment with an acceptable level of protection.

Sanitization is the process of eradicating stored data so that the data cannot be recovered using a reasonable level of effort using any known technology. Sanitization is appropriate when an instrument is to be removed from a secure environment.

10.2. Clearing Internal Memory

Clearing can be performed using the RS232 serial interface. Connect to the unit as in normal operation and send the command FACTORY PRESET. This will set all user-data to factory defaults. Turn off power to clear the volatile memory.

10.3. Sanitizing Internal Memory

Internal memory sanitization follows the guidelines set in NIST SP 800-88 for flash and EEPROM memories. This includes:

1. Performing a sector erase (if applicable). As some devices do not support this feature, for these devices the entire chip is written with all 1's (0xFF) which is the normal erased state for most flash/EEPROM memory.
2. Write a pattern to all locations. Verify the pattern.
3. Write the inverse pattern to all locations. Verify the inverse pattern.
4. Write all 0's (0x00) to all locations. Verify the locations contain 0.
5. Write all 1's (0xFF) to all locations. Verify the contents contain 0xFF.

Sanitization can be performed using the RS232 serial interface. Connect to the unit as in normal operation and send the command FACTORY SANITIZE. You will see the "sanitizing" message along with a progress indicator as the internal firmware performs the sanitization procedure. If the sanitization is successful you will then see a message stating "*TURN OFF POWER*" to complete the clearing of volatile memory. The instrument may now be removed from the secure environment.

```
>FACTORY SANITIZE
sanitizing.....
*TURN OFF POWER*
```

After performing the sanitization you can also verify that the procedure was successful. Prior to turning off power send the command FACTORY SANITIZE VERIFY. You should see the following, indicating that the memory is sanitized:

```
>FACTORY SANITIZE VERIFY
verifying...clear
```

Any other response to the above commands indicates that the sanitization was unsuccessful.

During the first reboot after performing a sanitization you will see a variety of Error 3xx "failure" messages as the unit detects the erased memory areas and performs a factory preset action. This is normal. Reboot the unit and the factory presets should be in effect.

11. Maintenance

The following paragraphs provide general inspection and maintenance guidelines for the unit.

11.1. Inspection

Perform a visual inspection in conjunction with the maintenance activities schedule when a malfunction is suspected, or whenever an assembly is removed or replaced.

11.2. Preventative Maintenance

While the unit requires very little preventative maintenance, it should not be subjected to physical abuse, severe mechanical shock, high humidity, or operating temperatures outside the specification range. The instrument should be kept free of excessive dirt and dust, since these can interfere with connector functions and with normal heat dissipation. The following paragraphs provide the preventive maintenance that is to be performed on the Unit.

Care should be taken to prevent strain on the interconnecting cables, since damage here may not always be apparent. Occasionally check the external cables and connectors for signs of cracked insulation and/or bent or worn pins. Tests show that connectors must be clean for accuracy and stability. This requires an inspection and cleaning of each connector immediately before use. When cleaning precautions are observed regularly, connectors can maintain their stability for over several thousand connection cycles. Refer to Appendix for more information about cables and connectors.

11.3. Machined Surfaces and Hardware

To remove light dirt and dust from mechanical parts such as castings, covers and other hardware proceed as follows:



WARNING

Compressed air used for cleaning and/or drying can create airborne particles that may enter the eye. Goggles/ face-shields should be worn. DO NOT direct air stream towards self or other personnel. Pressure should be restricted to a maximum 15 psi to avoid personal injury.



CAUTION

Under no circumstances use a wire brush, steel wool, or abrasive compound. Using these items will cause extensive damage to the instrument's surface. DO NOT use a nylon bristle brush in solvent as the bristles may dissolve and cause damage to the circuit card or component.

- Use 5 psi of clean, moisture-free compressed air or preferably dry nitrogen to blow loose dirt and dust from surface of item.
- Briskly brush isopropyl alcohol onto area to be cleaned with a fiber-bristle brush.
- Remove residue with lint-free cloth and repeat previous step as a rinse.
- When parts are thoroughly clean, dry parts using 5 psi of clean, moisture-free compressed air or preferably dry nitrogen.
- Clean smaller mechanical parts or hardware by dipping into a container of isopropyl alcohol. Remove dirt by brushing with fiber-bristle brush after parts have been immersed for several hours.
- Remove parts from isopropyl alcohol and rinse by immersing into a different container of isopropyl alcohol.
- When parts are thoroughly cleaned, dry parts using 5 psi of clean, moisture-free compressed air or preferably dry nitrogen.

11.4. Chassis Cleaning

Clean chassis using a lint-free cloth moistened with water and mild detergent. For harder to clean areas, such as inside corners of chassis, use a vacuum cleaner.

11.5. Connector Cleaning

Where small amounts of rust, corrosion, and/or oxide deposits are present on connectors, clean externally with a soft-bristle brush, aluminum wool, or internally with an acid brush; then wash with a non-corrosive solvent. Exercise care to ensure no metal filing or residue remains inside the connector and the connector is thoroughly dry. Where rust, corrosion, and/or oxide deposits are present in large quantities, replace the connector.

12. Replacement Parts List and Drawings

The assembly/component locations and schematic diagrams for the Model 44xx & 48xx are located in the Appendix by the drawing number. Drawing find numbers have also been included in the manuals RPL to help locate components or hardware.

12.1. Factory Service and Repairs

DO NOT send products back to the factory without prior authorization.

Please contact the Weinschel Customer Service Department to discuss your product and resolve any issues that may be corrected without returning the product to the factory. If the issue cannot be corrected, you may be issued an RMA number and instructed to return the product. Additionally, you may be requested to submit additional information regarding the product failure to help verify your complaint.

When contacting customer service, please provide the following information:

1. Product Model Number
2. Product Serial Number
3. Date of Original Purchase
4. Company Name
5. Name
6. Phone Number

If a product has been approved to be returned to the factory, follow these instructions to ensure timely service.

1. If possible, use the original packing container and cushioning material. If the original materials are not available, use a strong shipping container and protect the product with shock absorbing material.
2. Shock absorbing material should be 3/4 inch thickness or greater and should protect all sides of the unit, as well as prevent movement.
3. Attach a tag to the product with the following information:
 - Model and serial numbers of all returned products
 - Service being requested
 - Description of malfunction
 - Return address
 - Authorization to conduct repairs
 - Return authorization number (RMA #)
4. Seal the packaging and mark it as FRAGILE.
5. Ship the product to the listed addressor or to an authorized sales representative. This information will be supplied by Weinschel.

13. Contacting Weinschel

Please use the general information below to contact Weinschel for any inquiries.

Mail	Weinschel 5305 Spectrum Drive Frederick, MD 21703-7362 U.S.A.
Telefax	1-301-846-9116
Phone	Toll Free: 1-800-638-2048 Toll call: 1-301-846-9222
Website	http://weinschel.apitech.com/
E-mail	weinschel-sales@apitech.com

13.1. Manufacturer Warranty

PRODUCTS - Weinschel, a part of API Technologies Corp., warrants each product it manufactures to be free from defects in material and workmanship under normal use and service anywhere in the world. Weinschel's only obligation under this Warranty is to repair or replace, at its plant, any product or part thereof that is returned with transportation charges prepaid to Weinschel by the original purchaser within ONE YEAR from the date of shipment.

The foregoing Warranty does not apply Weinschel's sole opinion to products that have been subject to improper or inadequate maintenance, unauthorized modifications, misuse, or operation outside the environmental specifications for the product.

SOFTWARE PRODUCTS - Weinschel software products are supplied without representation or Warranty of any kind. Weinschel, therefore, assume no responsibility and will not accept liability (consequential or otherwise) arising from the use of program materials, disk, or tape.

The Warranty period is controlled by the Warranty document furnished with each product and begins on the date of shipment. All Warranty returns must be authorized by Weinschel prior to their return.

Weinschel's Quality System Certified to:



14. Appendix

14.1. Revision History

Revision	Date	Description of Changes
X	10/4/2021	Initial R

The remaining pages of this user manual are reserved for supporting documentation and drawings.