APPENDIX A. IEEE-488 INTERFACE PROTOCOL

A.1 Introduction

The R-110 receiver can use its IEEE-488 interface in either of two modes. In one mode it uses the interface to control an external R-1180 microwave downconverter. In the other mode it may be remotely controlled by a host computer. These two modes are mutually exclusive -- only one mode may be operative at a time. Thus if both an MDC and a host computer are present on the interface at the same time, the host computer must then control the MDC directly, not through the receiver.

This document deals with the command codes used by the host computer to command the receiver, and the formats of any required responses. An attempt is being made to conform to the IEEE-488.2 standard, in a limited way. While all of the "mandatory" features should eventually be implemented, few of the "optional" ones are anticipated. In addition, of the many data formats supported by the spec, only a few will be recognized by the radio, typically one specific format per command.

A.2 Receiver Addressing

The address of the receiver is set by a dipswitch on the processor PCB. It may be changed either by resetting the dipswitch and cycling power or reset, or may be changed (in local mode) using the front panel controls. The default DSI-standard address setting for the receiver is 16.

A.3 General Command Formats

Two general command formats are supported: the commands relating to the radio's settings, and a list of common commands identified in the '488.2 spec. The commands in the first list have in common a header consisting of a number of purely alphabetic characters (plus digits and underscores after the first character), whereas those in the second list begin with an asterisk followed by three alphabetic characters. Not included in this list are the low-level "bus messages" of the '488.1 spec (e.g., address-to-listen). Command headers are case-insensitive.

Following the header, the setting commands will typically contain a parameter field or a question mark. A specified parameter is the setting to be applied to the receiver, while a question mark indicates a query by the host for the receiver's current value of the setting specified by the header. In response to a query the receiver will respond with its current value of the desired setting, using a more precise form of the command's data format, minus the header.

Embedded within commands there may be "whitespace", defined as non-printing and blank characters. Length of whitespace in commands is arbitrary, since it is ignored by the receiver other than where at least one whitespace character is required by the '488.2 spec.

A.4 Data Formats

Decimal numeric data can be supplied in integer, decimal, or exponential format, as specified by '488.2. Units identifiers, such as "MHz", while made optional by '488.2, are not recognized by the radio. Hexadecimal data is in hex integer format, preceded by "#H". Both upper and lower case are acceptable for A - F and H. String data is mostly "mnemonic" format, which means upper or lower case letters, numbers, and the underscore, with the first character required to be alphabetical. There is also an instance of arbitrary length ASCII data, terminated with newline in conjunction with EOI.

A.5 Whitespace

'488.2 defines whitespace as the ASCII characters in the ranges 00 - 09 and 0B - 20 hex. They may be present in groups of arbitrary length. These characters are not processed by the receiver, but rather are skipped over, except where at least one whitespace character is required in a particular place by a particular command form.

A.6 Message Separator

The semicolon (";") is specified as the message separator by '488.2. Messages containing more than one command or query use this character to separate the pieces. Response messages also use the character to delimit responses from different queries.

A.7 Data Separator

The comma (",") is specified as the data separator by '488.2. It is used to delimit individual parameters or data within a single command, query, or response. Commas can be preceded or followed by whitespace in commands, but not in responses.

A.8 Compound Command Headers

The use of multiple command header mnemonics, separated by ":" characters, is not implemented in the receiver.

A.9 Message Terminator

The EOI line may be used without ATN to indicate the end of a message. (Used with ATN it forces a parallel poll.) So can a newline character (ASCII linefeed). '488.2 requires the receiver to recognize any combination of newline and EOI as a message terminator. The terminator may be preceded by whitespace. In order to conserve bytes the receiver will simply assert EOI with the last byte of data in a response message, foregoing the newline character entirely. The exception is when the response is in "arbitrary ASCII", which requires that the string be terminated with both the newline and EOI.

A.10 Low Level Interface Functions

The IEEE-488.1 interface spec lists a number of available low-level functions and capabilities. Many of these have associated with them a "capability level" which can range from no capability to full implementation, and may in some cases provide for partial capability in which some but not all of the function is available. A description of the '488.1 functions and the R-110's implementation of them is provided in the following paragraphs.

A.10.1 Source Handshake

This allows the radio to send data on the interface. The radio is fully capable of doing this. The capability code is SH1.

A.10.2 Acceptor Handshake

This allows the radio to receive data on the interface. The radio is fully capable of doing this. The capability code is AH1.

A.10.3 Talker Function

This allows the radio to be placed in talker mode so that it may send data on the bus. Sixteen capability levels are defined by '488.1, but only four are acceptable to '488.2. The radio supports the T6 capability level, which means that it can handle serial polls and that it leaves talker mode automatically when addressed to listen. It does not support "talk only" in remote, since it must be commanded to talk by an external controller. (In MDC mode it supports "talk only" as part of the control sequence.) It does not support extended or dual primary addressing.

A.10.4 Listener Function

This allows the radio to be placed in listener mode so that it may receive data on the bus. Eight capability levels are defined by '488.1, but only four are acceptable to '488.2. The radio supports the L4 capability level, which means that it will automatically leave talker mode if addressed to listen. It does not support "listen only" in remote, since it must be commanded to listen by an external controller. (In MDC mode it supports "listen only" as part of the control sequence.) It does not support extended or dual primary addressing.

A.10.5 Service Request Function

This allows the radio to signal the host that it needs attention, using a dedicated control line on the bus which is shared by all connected devices. '488.1 requires that the host respond to a service request with a serial poll. '488.2 makes elaborate use of this interface and requires full capability, which the radio supports. The capability code is SR1.

A.10.6 Remote/Local Function

This allows the radio to be placed in remote and returned to local. There is also the option of recognizing commands to lock out either remote or local. '488.2 requires either no or full capability, including lockouts. The radio supports full capability, for which the code is RL1.

A.10.7 Parallel Poll Function

This allows several devices on the bus to respond to polling by the host simultaneously, each having been previously assigned one bus data line, with no sharing. It is not a response to the service request function. Line assignment can either be done in hardware in the responding device, or may be assigned by the host via a command. '488.2 requires either no capability or remote assignment. The radio does not support parallel polling, for which the capability code is PP0.

A.10.8 Device Clear Function

This causes the radio's interface to be cleared. Both selective and universal commands are provided by '488.1, but '488.2 requires full capability, which includes both. The capability code is DC1.

A.10.9 Device Trigger Function

This provides a synchronization signal from the host so that several devices on the bus can perform activities simultaneously. What those activities are depends on the particular device, but are expected to be associated with the devices function rather than with the interface. '488.2 does not require the capability for its basic set of operations, and so it is not implemented in the R-110. The capability code is DT0.

A.10.10 Controller Function

This allows a device to either control the bus from the outset or to have control passed to it later. Of the 28 capability levels defined by '488.1 only five are supported by '488.2. The radio is not a controller during remote operation, giving it no capability, for which the code is C0. (In MDC mode it is in control of the bus, however. This one reason why MDC mode and remote mode are mutually exclusive.)

A.10.11 Electrical Interface

Certain interface lines may be either three-state or open-collector. Both configurations are allowed by '488.2. The radio uses three-state drivers where possible, for which the capability code is E2.

A.10.12 Capability Level Summary

In summary, the radio's capability levels are:

- SH1
- AH1
- T6
- L4
- SR1
- RL1
- PP0
- DC1
- DT0
- C0
- E2

A.10.13 Interface Clear Function

The IFC signal is given a dedicated line on the bus. It is used by the system controller to gain control back from any other device which maybe temporarily in charge. Since we're not a controlling device here, it doesn't really concern us.

A.10.14 EOI Function

The EOI signal is given a dedicated line on the bus. It is used to indicate the end of a message or as part of a parallel poll. '488.2 requires that either EOI or the ASCII linefeed be acceptable as an incoming message terminator, and that both be used at the end of a response message.

A.10.15 Serial Poll

Serial polling is required by '488.1, but its use is much more explicitly defined by '488.2. It consists of polling devices on the bus one by one. When polled, a device returns a full byte of data (as opposed to parallel polling, in which all devices respond simultaneously, each being typically assigned a single bit). One bit of the return byte (bit 6) specifies if the particular device being polled is currently asserting a service request. Of the other bits, some have specific uses required by '488.2 See the status register and event status register commands and queries in the section on common commands.

A.11 Common Commands and Queries

A list of common commands is explicitly defined by the '488.2 spec. Some are required by all compliant devices, while others are optional, but if provided must still comply with the descrption given in the spec. Required commands include:

- Identification query
- Reset command
- Self-test query
- Operation complete command and query
- Wait to continue command
- Clear status command
- Standard event status enable command and query
- Standard event status query
- Service request enable command and query
- Status query

Optional common commands supported by the R-110 include:

- Save settings
- Recall settings

Command mnemonics, parameters and data, and all response data is in ASCII. In the following descriptions, strings enclosed in angle brackets (e.g. <*IDN?>) represent the string to be recognized or sent, minus the brackets. Alphabetical characters may be upper or lower case. Quote marks may be single or double as long as both starting and ending marks match. One sort may be enclosed in another sort without interference.

The '488.2 spec assumes that both commands/queries and responses may be accumulated in input and output buffers or queues, separated by semicolons. When the output buffer is emptied an ASCII linefeed, accompanied by EOI, is sent to indicate the end of the response message.

A.11.1 Identification Query

The identification query format is <*IDN?>. The response from the receiver is single string in arbitrary ASCII format, comprised of the following fields, separated by commas:

Field 1:	manufacturer
Field 2:	model
Field 3:	serial number or <0>
Field 4:	firmware level or <0>

The string is terminated with a newline character in conjunction with EOI, and the whole thing must be fewer than 72 characters in length. The string termination requires that this query be the last operation before reading the output queue, or it and everything preceding it in the buffer will be wiped out by the next query.

A.11.2 Reset Command

The format is <*RST>. The action taken is to return the receiver to its powerup settings while maintaining it in remote mode.

A.11.3 Self Test Query

The format is <*TST?>. The action taken is to perform a self-test of the receiver. The response for a successful self-test is <0>. A not-so-successful self-test response is an NR1 optionally signed integer from -32767 to 32767, excluding 0. Currently there is no self-test implemented in the receiver, so the expected response is <0>.

A.11.4 Operation Complete Command and Query

The formats are <*OPC> and <*OPC?>. The command tells the receiver to set the "operation complete" bit in its standard event status register when the actions currently in process are completed. Since the receiver only does one thing at a time this should be the case immediately after the response to each incoming command.

The query causes the receiver to respond with an ASCII <1> when all current operations in process are completed. Since we only do one thing at a time the response is immediate.

A.11.5 Wait to Continue Command

The format is <*WAI>. This command causes the receiver to finish all operations currently in process before starting on anything new. It does this anyway, so the command is accepted but ignored.

A.11.6 Clear Status Command

The format is <*CLS>. Action taken is to clear all the status bits used for service requests and serial polls.

A.11.7 Standard Event Status Enable Command and Query

The formats are <*ESE data> and <*ESE?>, where the whitespace in the command is mandatory. The command provides an enable mask for the status register. Unmasked bits contribute to the generation of a summary bit which in turn forms part of the status register which generates service requests.

The format of the standard event status register is:

Bit 0: operation complete
Bit 1: request control
Bit 2: query error
Bit 3: device dependent error
Bit 4: execution error
Bit 5: command error
Bit 6: user request
Bit 7: power on

These bits are ANDed with the mask bits specified in the command and the results ORed to produce the ESB bit (bit 5) in the status byte register. Descriptions of the individual bits is as follows:

Operation Complete: normally cleared, set in response to the operation complete command after all in-process activities are completed.

Request Control: a request for transfer of control from the host. Not used.

Query Error: an attempt has been made to read response data from the receiver when none is available or pending, or response data has been lost.

Device Dependent Error: catch all for errors other than query errors, command errors, and execution errors. An example is a step command past the end of the available tuning range.

Execution Error: command data out of range or unavailable, or some current status of the receiver prevents the command from being implemented.

Command Error: command syntax error, or an unrecognized command, or a group execute trigger received in the middle of a message.

User Request: some special control available to the user is activated to signal the host. Not used.

Power On: indicates power has been cycled.

All of the foregoing bits are cleared after being read.

The command header <*ESE> is followed by whitespace, then by a decimal format number from 0 - 255, which when converted to binary forms the mask for the event status register bits.

The response to the query is an NR1 integer from 0 - 255, representing the mask byte currently in use.

A.11.8 Standard Event Status Query

The format is <*ESR?>. The response is an NR1 integer from 0 - 255 representing the current contents of the standard event status register defined above. The register is cleared after being read. The enable mask used for generation of the ESB bit is not applied to the response data.

A.11.9 Service Request Enable Command and Query

The formats are <*SRE data> and <*SRE?>. Function and formats are similar to the event status register command and query described above except that the mask is applied to the service request status byte and is used in the generation of the service request message. The service request status byte is returned in response to a serial poll.

The bits of the status byte, given above, are repeated here:

Bit 0: power supply fault
Bit 1: lock status fault
Bit 2: front overload
Bit 3: back overload
Bit 4: message available ('488.2)
Bit 5: event status ('488.2)
Bit 6: service requested ('488.1)
Bit 7: (unassigned)

Program data is a decimal number from 0 - 255 which forms the current mask. Note that bit 6 of the enable mask is ignored, since bit 6 of the status byte is effectively the ORed combination of the other 7 bits. The response data is an NR1 integer from 0 - 63 or 128 - 191, indicating that bit 6 is always returned clear.

A.11.10 Status Query

The format is <*STB?>. The response is an NR1 integer from 0 - 255 representing the status byte defined in the preceding paragraph. The status register is not necessarily cleared after reading, and the mask used to enable service requests is not applied.

A.11.11 Store Settings Command

The format is <*SAV data>, where the whitespace is mandatory and <data> is the storage location number in decimal format. It can range from 0 - 99.

Currently the list of what is saved is as follows:

- tuned frequency
- scan start frequency
- scan stop frequency
- scan step size
- scan rate
- scan repeat mode (one-shot, unidirectional, bidirectional)
- tune step size (from selected tuning digit)
- tune mode (step or ramp)
- input attenuation
- reference input attenuation (for absolute gain calculation)
- input selection
- MDC attenuation
- MDC input selection
- input attenuation when exiting MDC mode
- input selection when exiting MDC mode
- bandwidth
- bandwidth normal/extended selection
- reference bandwidth (for absolute gain calculation)
- wideband selection
- gain
- gain distribution in BFO
- gain distribution otherwise
- gain reference (for absolute gain calculation)
- absolute gain mode select
- delta gain mode select
- AGC select
- autorange select
- log/lin select
- Z axis select
- Z axis invert select
- BFO select
- pulse stretch enable (for future implementation)
- slideback enable (for future implementation)

- tune stored step size select
- scan pause resume select (continue or revert)
- GPIB enable
- MDC enable
- limit beep enable
- error beep enable
- fault beep enable

The state is saved to temporary storage in the radio, so that it is lost when power is removed. Note that the radio treats the store and recall bus commands just like the front panel operations, using the same storage locations. This means that settings stored in local may be recalled in remote and vice versa. It also means that some settings which are always disabled in remote are nevertheless stored and recalled by the interface commands.

A.11.12 Recall Settings Command

The format is <*RCL data>, where the whitespace is mandatory and <data> is the storage location number in decimal format. It can range from 0 to 99 for accessing temporary storage and -0 to -99 for permanent storage.

What gets recalled and loaded into active storage (and hardware) is listed in the description of the store command above.

A.12 Device-Specific Commands and Queries

Commands and queries specific to the receiver include:

- tuned frequency
- tuning step size
- tuning step up/down
- signal input select
- input attenuation
- IF bandwidth
- wide bandwidth
- gain
- gain distribution
- IF attenuation
- lin/log detection
- query all settings
- gain table initialization
- end-to-end gain initialization
- DCIF fixed gain initialization

Except for "query all settings" and the step up and down commands, each item provides both a command and a query. Descriptions of each are given in the following paragraphs.

A.12.1 Frequency Tuning Command and Query

Tuned frequency can range from 1 kHz to 1 GHz. The format of the command is <FREQ data>, where the whitespace is mandatory and <data> is in decimal format with resolution as needed down to 0.1 Hz. The number specified will be interpreted as Hz.

The query has the format <FREQ?>. The response is returned in NR3 exponential format, in Hz, with no header.

Note: the minimum legal tuned frequency in wideband mode is 15 MHz.

A.12.2 Frequency Tuning Step Size Command and Query

Step size can range from 0.1 Hz to 1 GHz, although the upper bound is provided for compatibility rather than for practical use. The receiver can tune to 0.1 Hz accuracy all the way to 1 GHz although the display on its front panel loses digits on the right at higher frequencies.

The format of the command is <STEP data>, where the data is in decimal format, in Hz.

The format of the query is <STEP?>. The response is returned in NR3 exponential format, in Hz, with no header.

Note: the minimum step size in wideband mode is 5 MHz, and must be a multiple of 5 MHz. Step size commands will be accepted bearing other values, and steps will be implemented to the nearest 5 MHz value. Queries will be responded to with the value commanded.

A.12.3 Frequency Tuning Step Up and Down Commands

This command adds or subtracts the current value of tuning step size to/from the currently tuned frequency. There are two commands, <STEPUP> and <STEPDN>.

In wideband mode the lower tuning limit becomes 15 MHz and the step size granularity is 5 MHz. If a step size which is not an integer multiple of 5 MHz is used then stepping will not proceed smoothly, since the step size provided will be added to/subtracted from the current tuned frequency and stored, but will be truncated to the next lower 5 MHz value before being applied to hardware.

A.12.4 RF Input Select Command and Query

The format of the command is <INP data>, where the whitespace is mandatory and the data is in decimal format, and must evaluate to 1 or 2. The query is <INP?>. The response is <1> or <2>, with no header. Input 1 is the upper RF input connector and input 2 is the lower.

A.12.5 RF Input Attenuation Command and Query

The format of the command is <ATTN data>, where the whitespace is mandatory and the data is in decimal format, ranging from 0 to 70 in steps of 10. The query is <ATTN?>. The response is an NR1 integer 0 to 70 in steps of 10, with no header.

A.12.6 IF Bandwidth Command and Query

The format of the command is <BW data>, where the whitespace is mandatory and the data is in decimal format. The command for wideband is the specific form <BW WIDE> where the whitespace is mandatory and the string is case-insensitive. The query is <BW?>. The response is in NR3 exponential format or <WIDE>, with no header.

Command data must evaluate to one of the legal bandwidths:

15 MHz	4 MHz
1 MHz	300 kHz
80 kHz	20 kHz
16 kHz	12.5 kHz
10 kHz	8 kHz
6.4 kHz	5 kHz
4 kHz	3.2 kHz
2.5 kHz	2 kHz
1.6 kHz	1.25 kHz
1 kHz	800 Hz
640 Hz	500 Hz
400 Hz	320 Hz
250 Hz	200 Hz

A.12.7 IF Gain Command and Query

The format of the IF gain command is <GAIN data>, where the whitespace is mandatory and the data is in decimal format, representing a number of dB from 0 - 50, to 0.1 dB resolution. The command can also take the specific form <GAIN AGC> in which the whitespace is mandatory and the string is case-insensitive. The query is <GAIN?>. The response is in NR2 decimal format or <AGC>, with no header.

Note: the IF gain stages are unused in wideband mode. It is the responsibility of the host to deal with this. Gain commands will be accepted, and queries responded to, even though the signal path bypasses the stage. If a switch is then made to narrowband then the commanded gain will be applied.

A.12.8 IF Gain Distribution Command and Query

The formats of the gain distribution commmand are <DIST IMP> and <DIST CW> where the whitespace is mandatory and the strings are case-insensitive. The query is <DIST?>. The response is <IMP> or <CW>.

A.12.9 Detector Selection Command and Query

The formats of the command are <DET LIN> and <DET LOG> where the whitespace is mandatory and the strings are case-insensitive. The query is <DET?>. The response is <LIN> or <LOG>.

A.12.10 Query All Settings

The query is <INFO?>, while the response is formatted as follows, with commas between fields:

Field 1:	tuned frequency, in NR3 exponential format
Field 2:	tuning step size, in NR3 exponential format
Field 3:	input selection, <1> or <2>
Field 4:	input attenuation, NR1 integer multiple of 10 from 0 - 70
Field 5:	IF gain, NR2 decimal from 0 - 50 with 0.1 resolution, or <AGC>
Field 6:	IF gain distribution, <IMP> or <CW>
Field 7:	IF bandwidth, NR3 exponential format or <WIDE>
Field 8:	detector selection, <LIN> or <LOG>

There are no headers in the response fields.

A.12.11 IF Attenuation Command and Query

Programmable gain in the IF is normally handled by the IF gain command described above. That command includes a gain value which indexes a gain distribution table (which may be selected using the gain distribution command) which contains dB values for the first two attenuator circuits in the IF. Each of these dB values is used as an index into separate linearization tables which in turn provide the attenuator settings as binary codes. Meanwhile, the third attenuator circuit is handled separately as end-to-end gain compensation. It also passes through a linearization table, but the setting is determined by tuning band rather than gain setting. The IF attenuation command described here, in contrast to the gain command and end-to-end compensation, bypasses all of the tables and applies data passed with the command directly to the driver circuits. Note that in doing so the receiver will lose knowledge of its actual gain setting, since it bases that value on the value used to address the distribution table and the end-to-end linearization table.

The format of the command is <IATN data , data , data>, where the first whitespace is mandatory and the rest optional, and the data is 12 bits each of binary in hexadecimal format (for example <#H123,#H456,#H789>). Alphabetical characters in hexadecimal format are case-insensitive. The query is <IATN?>. The response is <data,data,data>, again in hexadecimal format. For both the command and the query the first value given is what is applied to the first attenuator in the IF signal path.

A.12.12 End-to-End Gain Command and Query

End-to-end gain is established by the last attenuator circuit in the IF. It is a fixed value for each of the three main tuning bands (1 - 250 KHz, 250 KHz - 15 MHz, 15 MHz - 1 GHz). The value is in dB to 0.1 dB resolution and can range from 0 - 17.5 dB. The intent is to establish a standard end-to-end gain for all receivers, given the same IF gain, input attenuation, bandwidth, detector, and AGC settings.

The format of the command is <EATN data1 , data2 , data3>, where the first whitespace is mandatory and the rest optional, and data is in decimal format for bands 1, 2, and 3 respectively. The query is <EATN?>. The response is <data1,data2,data3> in NR2 decimal format. Command data is stored in nonvolatile memory and becomes part of the gain calibration of the radio. The data which pertains to the current tuning band is also applied to hardware.

A.12.13 DCIF Fixed Gain Command and Query

The end-to-end gain compensation circuit described in the preceding paragraph may be bypassed when the DCIF is in use. The DCIF, in addition to its own fixed gain and bandwidth-compensating gain requirements, must also handle its own end-to-end gain compensation. The required value depends both on the current tuning band, as it did in the IF circuit, but also upon whether log or linear detection is selected. Note that the DCIF is controlled by programmable amplifiers rather than attenuators, so it uses gain values rather than attenuation.

The format of the command is <DCGNdb data , data> where the first whitespace is mandatory and the rest optional, the <d> is <LN> for linear or <LG> for log, <b> is <1>, <2>, or <3> for band, and the data is in decimal format, in dB of gain to 0.1 dB resolution, for the first and second DCIF programmable amplifiers, respectively. The values passed are for combined fixed and end-to-end compensating gain. The query is <DCGNdb?> with the same use of <d> and <b> as before. The response is <data,data> in NR2 decimal format, representing combined fixed and end-to-end gain for the first and second amplifiers respectively.

Command data is stored in nonvolatile memory and becomes part of the receiver's calibration. If the command specification matches the current tuning band and detector selection then the data is also applied to hardware.

A.12.14 IF Attenuation Linearization Table Command and Query

IF gain and end-to-end gain are set by means of PIN diode attenuator circuits. There are three of these in series, with intervening amplifiers and filters. The first is a double circuit connected to a single driving function, so the first circuit affords twice the attenuation range of either of the other two. Drive requirements for these circuits are fairly nonlinear, and vary significantly from unit to unit as well, so tables have been implemented which convert a desired attenuation in dB into a digital code which is used to drive a given attenuator circuit. The tables give values for every 0.2 dB from 0 - 35 dB for the first circuit, and every 0.1 dB from 0 - 17.5 dB for the other two. The first table value is for baseline zero attenuation, ranging up to maximum attenuation at the other end. Table values are 12 bits long each (although the prototype receivers only use the first eight).

Note that baseline zero attenuation does not necessarily need to measure zero for any particular attenuator circuit. In fact it may be preferable to give up a dB or so initially in order to avoid the very nonlinear knee at turnon, and then make up the difference in the end-to-end compensation. The eventual distribution of gain is the determined by a combination of the impulsive and CW gain distribution tables, the linearization tables, and the end-to-end compensating gains for both the IF and the DCIF. While the distribution tables are not currently modifyable, the rest must be set up by a properly written calibration algorithm and test procedure, which is not described in this document.

The command is <ATBLx index , data[, ...]>, where <x> can be <1>, <2>, or <3> to indicate which table, <index> is a decimal number from 0 - 175 indicating the table record number, and <data> is 12 bits in hexadecimal format, for example <#H123>. The alphabetical characters in hexadecimal format are case-insensitive. Data supplied after the first value will be set into subsequent consecutive entries in the table. In this manner, starting with 0, the entire table may be initialized with a single command. The query is <ATBLx? index>, where the whitespace is again mandatory and <x> and <index> are the same as for the command. The response is the same hexadecimal data. A special case of the query is <ATBLx? ALL>, where the whitespace is mandatory and the string is case-insensitive. The response to this query is the entire contents of the table starting with 0, with the entries separated by commas.