XRA700 Programmer's Guide

Amptek Inc.
14 Deangelo Drive, Bedford, MA 01730
PH: +1 781 275 2242 FAX: +1 781 275 3470
amptek.sales@ametek.com www.amptek.com

Table of Contents

1 Change List 3
2 Software 3
2.1 Uploading new firmware 3
2.2 Software Resources 3
2.2.1 Advanced VB Demo 3
2.2.2 SDK 3
3 Interfaces 4
3.1 USB 4
3.1.1 Installing the WinUSB Driver 4
3.1.2 WinUSB GUIDs 5
3.2 Ethernet 5
3.2.1 Ping 5
3.2.2 Ethernet Ports 5
3.2.3 General UDP Port 6
3.2.4 Netfinder UDP port 3040 7
3.3 RS232 9
4 Packet Format 10
4.1 Request Packets: 12
4.1.1 Request packet: "Request Status Packet" 14
4.1.2 Request packet: "Request Misc Data" 15
4.1.3 Request packet: "Request Ethernet Settings" 16
4.1.4 Request packet: "Request Diagnostic Data" 17
4.1.5 Request packet: "Request Netfinder Packet" 18
4.1.6 Request packet: "I^2C Transfer" 19
4.1.7 Request packet: "Text Configuration" 20
4.1.8 Request packet: "Text Configuration Readback" 21
4.1.9 Request packet: "Text Configuration (without saving to nonvolatile memory)" 22
4.1.10 Request packet: "Set Ch [1 through 7] DCAL" 23
4.1.11 Request packet: "Set HVS1, HVS2, HVS3 Calibration" 24
4.1.12 Request packet: "Write 512-byte Misc Data" 25
4.1.13 Request packet: "Set Ethernet Settings" 26
4.1.14 Request packet: "Select RS232 Baud Rate" 27
4.1.15 Request packet: "Interface Keep-alive – Allow Sharing" 28
4.1.16 Request packet: "Interface Keep-alive – No Sharing" 29
4.1.17 Request packet: "Interface Keep-alive – Lock" 30
4.2 Response Packets 31
4.2.1 Response packet: "Status Packet" 32
4.2.2 Response packet: "512-byte Misc Data Packet" 36
4.2.3 Response packet: "Ethernet Settings" 37
4.2.4 Response packet: "Diagnostic Data" 38

4.2.5 Response packet: "Configuration Readback" 39
4.2.6 Response packet: "Netfinder Packet" 40
4.2.7 Response packet: "I2C Read Data" 41
4.2.8 Response packet: "Comm test - Echo packet" 42
4.3 Acknowledge Packets 43
4.3.1 Acknowledge packet: "OK" 43
4.3.2 Acknowledge packet: "OK, with Interface Sharing Request" 44
4.3.3 Acknowledge packet: "Sync Error" 44
4.3.4 Acknowledge packet: "PID Error" 44
4.3.5 Acknowledge packet: "LEN Error" 44
4.3.6 Acknowledge packet: "Checksum Error" 45
4.3.7 Acknowledge packet: "Bad Parameter" 45
4.3.8 Acknowledge packet: "Unrecognized Command" 45
4.3.9 Acknowledge packet: "PC5 Not Present" 45
4.3.10 Acknowledge packet: "Bad Hex Record" 46
4.3.11 Acknowledge packet: "CP2201 Not Found" 46
4.3.12 Acknowledge packet: "Scope Data Not Available" 46
4.3.13 Acknowledge packet: "I^{2}C Error" 46
4.3.14 Acknowledge packet: "Calibration data not present" 46
5 ASCII Commands 48
5.1 Table 4 – ASCII Command Summary 49
5.1.1 BTDL – Determine Default Load status on configuration load 51
5.1.2 BTEC – Determine TEC status on configuration load 52
5.1.3 BTFN – Determine fan status on configuration load 53
5.1.4 BTHV – Determine high voltage power supply status on configuration load 54
5.1.5 BTMD – Boot Mode. Set the behavior of the device on power up. 55
5.1.6 BTPA – Determine preamplifier power supply status on configuration load. 56
5.1.7 C1EN, C2EN, C3EN, C4EN, C5EN, C6EN, C7EN – Channel Enable 57
5.1.8 ENDL – Default Load Enable 58
5.1.9 FANE – Fan Enable 59
5.1.10 HVS1, HVS2, HVS3 – High Voltage Supply Settings 60
5.1.11 HVSE – High Voltage Supply Enable 61
5.1.12 PAVE – Preamp Voltage Enable 62
5.1.13 RESC – Reset Configuration 63
5.1.14 TEC1, TEC2, TEC3, TEC4, TEC5, TEC6, TEC7 - Turn On/Off the Thermoelectric Cooler (TEC) Supply and Set the Temperature for detector on specified channel # 64
5.1.15 TECE – Thermoelectric Cooler Supply Enable 65
5.1.16 TECS - Turn On/Off the Thermoelectric Cooler (TEC) Supply and Set the Temperature for all detector channels 66
6 Select Topics 67
6.1 Important Configuration Considerations 67
6.1.1 Differences in Use of Non-Volatile Configuration at Power-up 67
6.1.2 Strategy to Avoid Degradation of Flash Memory 67

1 Change List

Reserved for future change notices.

2 Software

2.1 Uploading new firmware

The Firmware Manager application can be downloaded from the Amptek website for use in loading new firmware, changing various calibration settings, or changing the TCP/IP configuration of the device. Firmware Manager can be downloaded from here:

https://www.amptek.com/-/media/ametekamptek/documents/softwares/firmwaremanager.zip?la=en

The ZIP file download includes all current firmware files, so they can be manually installed as needed. Firmware Manager replaces the Firmware Loader and DP5 Loader applications. Firmware Manager can upgrade Amptek digital devices via USB, Ethernet or RS232.

2.2 Software Resources

There are a number of resources available to assist in developing an application:

2.2.1 Advanced VB Demo

The Advanced VB Demo can be downloaded from the Amptek website. It is written in Visual Basic 5, and shows how to use WinUSB, Winsock, and the VBComm control to communicate with a Amptek digital device. It handles nearly all packet types; it can upload new firmware and set various calibration values; it supports List Mode, though not at the highest count rates.

https://www.amptek.com/-/media/ametekamptek/documents/softwares/advanced_vb.zip?la=en

2.2.2 SDK

A Software Development Kit (SDK) is available from the Amptek website. It has C++ and VB examples. LabVIEW and other language examples may also be found on the website. Contact Amptek for more information.

https://www.amptek.com/software/software-downloads/dp5-digital-pulse-processor-software

3 Interfaces

The XRA700 supports three communication interfaces: USB, Ethernet and RS232. Details about the specific interfaces are in the following sections.

Note: Attempting to use multiple interfaces simultaneously can result in unexpected behavior. The UDP port is always open, regardless of whether USB is connected. The 'Netfinder' socket remains functional when USB is connected, and the contents of the Netfinder packet indicate if the UDP socket is unavailable because of USB.

3.1 USB

Amptek digital devices support 'full-speed' (12Mbps) USB 2.0. Three endpoints are used: the 'control' endpoint (EP0); EP1 IN (for response packets from the XRA700 to host PC); and EP2 OUT (for request packets from the host PC to XRA700.)

Transfers on EP1 and EP2 are terminated by a 'short packet' (a packet smaller than the max packet size of 64 bytes), or a zero-length packet ('ZLP').

For the XRA700, the USB Vendor ID (VID) and Product ID (PID) is:
VID: 0x10C4
PID: 0x842A

The DPPMCA application (and Advanced Visual Basic demo) use the WinUSB driver. This is a Microsoft product that is native to Vista and Windows 7 (32- and 64-bit), and installable on WinXP. The Visual Basic demo includes source code, and VC++ classes are available for those wishing to write their own communication software.

3.1.1 Installing the WinUSB Driver

The WinUSB driver has been tested by WinQual and signed for WinXP 32/64, Vista 32/64, and Win7 32/64. It is available via Windows Update, which means that a PC with an Internet connection should be able to install the driver automatically. Directions for manually installing the driver are in the 'WinUSB Driver Installation' folder in the ZIP file.

The ZIP file includes the document 'Manual WINUSB Driver Installation Instructions.pdf', which describes the complex process of using Windows' SetupAPI to find the device path, etc. [The VB demo demonstrates this technique, as well as setting pipe policies, opening pipes, sending and receiving data, etc.]

USB.h and USB100.h contain many structures, enums, etc. used by Windows' USB implementation – these files (from DDK6001) are in the ZIP file.

Microsoft's website gives information on how to call the specific WinUSB functions:
https://docs.microsoft.com/en-us/previous-versions/windows/hardware/drivers/ff540046(v=vs.85)

Windows 8 USB descriptors have been added, which may allow Win8 to automatically load WinUSB without having to install it. Microsoft's documentation is a bit sketchy on this; it hasn't been tested at Amptek.

Win8 descriptors added:

1. An 'OS String Descriptor', to tell Windows that extended feature descriptors are present;
2. An 'Extended Compat ID OS feature descriptor', to tell Windows to use WinUSB; and
3. An 'Extended properties OS feature descriptor', to specify the device's interface GUID.

3.1.2 WinUSB GUIDs

The WinUSB driver for the Amptek digital devices utilize two GUIDs:

FW6 GUIDs:
Setup Class GUID: {6A4E9A2D-9368-4f01-8E60-B3F9CDBAB5E8}
Interface Class GUID: {5A8ED6A1-7FC3-4b6a-A536-95DF35D03448}

3.2 Ethernet

The XRA700 incorporates a 10base-T Ethernet controller and embedded TCP/IP stack to provide Ethernet support via a standard RJ45 jack. The XRA700 supports auto-negotiation for half- and full-duplex. Only 10Mbit/s is supported.

PING and ARP are supported. A fixed IP address can be specified, or a dynamic address can be obtained from a DHCP server. (If 'dynamic' is selected, and the XRA700 is unsuccessful in obtaining a lease from a DHCP server, it will fall back to its previously-programmed static address.) The IP configuration (fixed/dynamic, IP address, netmask, gateway, and port) can be programmed via any of the XRA700 communication interfaces (Ethernet, USB or RS232).

Note: By default, the XRA700 is configured to use a dynamic IP address. If no DHCP server is present, then the device falls back to the programmed static IP address. The Firmware Manager application is the preferred way to assign an IP address, although the Advanced VB Demo can also perform this function. If no DHCP server is present and a static IP address hasn't been programmed, the device will default to 192.168.1.10.

The XRA700 may work poorly via Ethernet when directly connected to a PC's Ethernet port. The XRA700 works fine if connected to the PC via an Ethernet hub or switch. This is how we recommend using our devices if Ethernet connectivity is required.

3.2.1 Ping

XRA700 Ethernet implementation has a 56-byte PING buffer.

3.2.2 Ethernet Ports

The XRA700 supports two UDP ports for communication: one port is for general communications (fixed at UDP port 10001, and possibly programmable in later firmware releases), and the other is for device discovery via the Silicon Labs Netfinder protocol (UDP port 3040.)

3.2.3 General UDP Port

By default, the XRA700 uses UDP port 10001 for communications. Once a packet exchange has taken place on this port, the socket is 'bound' to the IP address **and source port** of the host it exchanged packets with. Once the socket is bound, packets from other IP addresses will be ignored. After approx. 15 seconds of inactivity on the socket, the socket is reset so that it can once again connect to any IP.

To not allow other hosts to seize control of the XRA700, the host can continue exchanging packets so that this timeout does not occur. "Interface keep-alive" Request Packets can be sent periodically, if there is no other periodic traffic, to keep the socket from resetting. Or an "Interface – lock" Request Packet can be sent, in which case the socket will not be reset until power is cycled on the XRA700, or the host removes the lock by sending an "Interface keep-alive - allow sharing" or "Interface keep-alive - no sharing" Request Packet.

Broadcast packets can be received by the General UDP Port, which may be useful if multiple XRA700s are on a subnet, and synchronization or centralized control is desired. (Request packets such as 'Enable MCA' and 'Clear Spectrum' could be useful as broadcasts.) All Request Packets received by the XRA700 will generate a Response Packet, which isn't desirable for broadcasts. Future versions of the firmware may use the most significant bit of the PID2 packet ID to indicate a broadcast packet, so that XRA700s will be able to suppress Response Packets. Also, a command or Request Packet will be added so that a XRA700 can be configured whether to process or ignore broadcast packets. (This will allow some XRA700s on a subnet to not participate in broadcasts, if this is desired.)

The existing protocol works over the Internet, but possibly not very well. As it uses raw UDP packets with no sequence numbers, and spectrum packets are large enough that they're split into multiple UDP packets, theoretically the spectrum UDP packets could be received out of sequence, with no way to tell. Eventually, the firmware will have the option to encapsulate each UDP packet within the XRA700 packet structure, and use the XRA700 packet PIDs as sequence numbers. This should allow the protocol to be more robust when routed over the Internet – the packets will be able to be reordered if they arrive out of sequence, and it will be easier to detect dropped packets.

Timeouts

Generally, the XRA700 sends the Response or ACK packet after it has completed the operation specified by the Request Packet. This is usually happens quickly, so that the response is sent with little delay. Since the XRA700 always sends a Response or ACK Packet in response to receiving a Request Packet, it may be desirable to use a timeout interval to identify missed packets.

The Visual Basic demonstration software uses a timeout interval of 1000mS by default. However, there are a few packet types which require a longer timeout interval because they may take longer than 1s to execute:

Description	PID1	PID2	LEN MSB	LEN LSB	Recommended Timeout
Request diagnostic data	3	5	0	0	2500mS
Erase uC image #1	0x30	5	0	2	2000mS
Write 512-byte Misc Data	0xF0	0x09	2	0	1000mS

3.2.4 Netfinder UDP port 3040

The XRA700 supports the Silicon Labs 'Netfinder' protocol on UDP port 3040. This allows XRA700s to be 'discovered' – this is particularly useful if DHCP is used to assign dynamic IP addresses, as the XRA700 may have an unknown address.

By broadcasting a Netfinder 'Broadcast Identity Request' to UDP port 3040, the host directs all XRA700s that receive the broadcast to respond with an 'Identity Reply', which includes the XRA700 serial number, IP address, MAC address, a text description (if one was programmed), the time the device has been powered, and the status of the interface. The host collects the responses from all the XRA700s on the subnet, and can then determine which IP address to access based on XRA700 serial number, description, etc.

The 'Broadcast Identity Request' is 6 bytes in size and is defined below. The format of the 'Identity Reply' is also listed below – its size is variable because it contains several variable-length null-terminated strings. Note that the 2nd string, the description, comes from the XRA700's 'Misc Data' buffer. If the firmware finds a null-terminated string of 40 characters or less, it will send that as the description string. Otherwise, '(no description)' is used. [Note: the XRA700 will not respond to the Netfinder 'standard' 4-byte version of the 'Broadcast Identity Request' as defined by Silicon Labs.]

Note that because UDP packet reception is not guaranteed (packets aren't automatically resent), occasionally not all responses will be received. It may be necessary to send multiple Identity Request packets to receive responses to all units. (This depends on how busy the network is, and how many XRA700s are present.) Each subsequent Identity Request should use a different sequence ID – a XRA700 will not respond to a repeated Identity Request with the same sequence ID.

Note that while the 'Broadcast Identity Request' packet is normally sent as a UDP broadcast packet (i.e. a destination IP of xxx.xxx.xxx.255), it can also be sent to a specific IP address. Also note that UDP broadcast packets generally are limited to the local subnet, and not routed beyond that.

The 'Interface Sharing Request' format is also listed below. If a Netfinder Identity Reply has indicated that a XRA700 is in use, but that sharing is allowed, the host can send the 'Interface Sharing Request' to the Netfinder port of that XRA700. The XRA700 will use an ACK packet to indicate to its current host that a Sharing Request has been received. If that host wishes to grant the request, it does so by stopping transfers to the XRA700's UDP socket, which will allow the socket to reset itself and the new host to gain control of it. [No acknowledgement of the 'Interface Sharing Request' is sent by the XRA700 – the

host can resend it periodically, and either check the Interface Status in the Netfinder 'Identity Reply', or try sending packets to the XRA700 UDP port to see if it responds.]

Broadcast Identity Request – 6 bytes

Offset	0	1	2	3	4	5
Value	0x00	0x00	Random 16-bit sequence ID		0xF4	0xFA

Identity Reply – variable length

Data Offset	Value
0	0x01
1	Ethernet General UDP Port Status: 0 = Interface is open (unconnected); 1 = Interface is connected (sharing is allowed); 2 = Interface is connected (sharing is not allowed); 3 = Interface is locked 4 = Interface is unavailable because USB is connected
2-3	Replication of Random 16-bit sequence ID received from 'Identity Request'
4-5	Event 1 Days
6	Event 1 Hours
7	Event 1 Minutes
8-9	Event 2 Days
10	Event 2 Hours
11	Event 2 Minutes
12	Event 1 Seconds
13	Event 2 Seconds
14-19	MAC address (MSB…LSB)
20-23	IP Address (MSB…LSB)
24-27	Subnet Mask (MSB…LSB)
28-31	Default Gateway (MSB…LSB)
variable	Null-terminated string: 'Amptek XRA700 – S/N ' + serial number
variable	Null-terminated string – Description/misc text: "(no description)" if Misc Text isn't programmed;

	first 41 bytes of Misc Text if programmed
variable	Null-terminated string – Event 1 description: "Time Powered"
variable	Null-terminated string – Event 2 description: "Time on Network"

Interface Sharing Request – 6 bytes

Offset	0	1	2	3	4	5
Value	0x04	0x00	Random 16-bit sequence ID		0xF4	0xFA

3.3 RS232

The RS232 port on the Amptek digital devices operate at 115,200 baud, with 1 start bit, 8 data bits, no parity bit, and 1 stop bit. Neither hardware nor software handshaking is used. 57,600 or 19,200 baud operation can be selected; however, selecting these rates will disable the Ethernet port.

The RS232 interface employs a 'gap timer' on its receiver, to aid in synchronization – if the time between consecutive received bytes exceeds 100mS, all buffered bytes are discarded, and the XRA700 will resume searching for the sync characters that signal the start of a new packet. No acknowledge packet will be sent to indicate this has occurred.

4 Packet Format

All communications with the XRA700 follow a request/response format: the host sends a Request Packet to the XRA700, and the XRA700 returns a Response Packet. The Response Packet sent by the XRA700 to the host will be:

a. a packet containing the requested data;
b. an acknowledge (ACK) packet, indicating the request packet was received and interpreted without error (if the request packet is not one for which data is returned);
c. an ACK packet, indicating the request packet contains an error in structure or content;
d. an ACK packet, indicating why the request can't be executed;
e. an ACK packet, indicating that an ASCII command had a unrecognized command field or an invalid parameter field – the command in question will be echoed in the data field of the ACK packet.
f. an ACK packet, indicating the request packet was received and interpreted without error, and a sharing request was received from another computer via Ethernet;

All packets sent to and from the XRA700 use the same basic packet format: a 6-byte header, which defines the type and length of the packet, an optional data field, and a 16-bit checksum. For packets sent to the XRA700, the optional data field can be up to 512 bytes, for a maximum packet size of 520 bytes. For packets sent by the XRA700 to the host PC, the maximum data field size is 32767 for a maximum packet size of 32775 bytes.

Packet Format

Offset	0	1	2	3	4	5	6...5+LEN	6+LEN	7+LEN
Value	0xF5	0xFA	PID1	PID2	LEN_MSB	LEN_LSB	Data (optional, 0-512 bytes)	CHKSUM MSB	CHKSUM LSB

Fields:

Offset 0 & 1: Sync bytes – these fields have fixed values of 0xF5 and 0xFA.

Offset 2 & 3: Packet ID fields 1 & 2 (PID1 & PID2) – these define the meaning of the packet, as summarized in Table 1, Table 2 and Table 3.

Offset 4 & 5: 16-bit length field (LEN) of the optional data field. If data field is not present, LEN=0.

If LEN =0:

Offset 6 & 7: 16-bit checksum: MSB, then LSB. This is a two's-complement of the 16-bit sum of all bytes prior to the checksum. (i.e. the 16-bit sum of the checksum and all other bytes in the packet is 0.)

If LEN > 0:

Offset 6: The start of the data field, whose length is given by the 16-bit LEN field. The last byte is offset 5+LEN.

Offset 6+LEN, 7+LEN: Checksum, as described above.

Checksum Visual Basic 5 example code:

```
‘ Note that a VB Long (32 bit) integer is used for the sum, as VB integers are signed,
‘ and a 16-bit signed INT wouldn’t work
‘ P() is the raw packet buffer


    Dim CSum As Long, X as Integer, LEN as Integer

    CSum = 0
    LEN = P(4) * 256 + P(5)  ‘ get the LEN field

    For X = 0 To LEN + 5    ' add up all the bytes except checksum
        CSum = CSum + P(X)
    Next X

    CSum = CSum + 256 * CLng(P(LEN + 6)) + CLng(P(LEN + 7))          ‘ add the 16-bit checksum

    If (CSum And &HFFFF&) = 0 Then
        ' packet is OK
        ‘ process the packet
    Else
        ' checksum error
        ‘ handle the error
    End If
```

4.1 Request Packets:

Table 1 – Request Packets

Description	Support*	PID1	PID2	LEN MSB	LEN LSB	Data (optional)	Response
Request status packet	ALL	1	1	0	0	XRA7-specific STATUS	Data packet
Request 512-byte misc data	ALL	3	2	0	0		Data packet
Request Ethernet settings	ALL	3	4	0	0		Data packet
Request diagnostic data	ALL	3	5	0	0		Data packet
Request Netfinder packet	ALL	3	7	0	0		Data packet
Perform I2C transfer	ALL	3	8	varies		I2C transfer description	ACK or Data
Text configuration (to XRA)	ALL	0x20	2	varies		ASCII configuration	ACK packet
Text configuration Readback (from XRA)	ALL	0x20	3	varies		ASCII configuration	Data packet
Text configuration (to XRA) without saving to nonvolatile memory	ALL	0x20	4	varies		ASCII configuration	ACK packet
Erase uC image #1	ALL	0x30	5	0	2	0x12 0x34	ACK packet
Set Ch 1 DCAL	*ALL*	*0xCA*	*0xD0*	*0*	*2*	*LSB, MSB*	*ACK packet*
Set Ch 2 DCAL	*ALL*	*0xCA*	*0xD1*	*0*	*2*	*LSB, MSB*	*ACK packet*
Set Ch 3 DCAL	*ALL*	*0xCA*	*0xD2*	*0*	*2*	*LSB, MSB*	*ACK packet*
Set Ch 4 DCAL	*ALL*	*0xCA*	*0xD3*	*0*	*2*	*LSB, MSB*	*ACK packet*
Set Ch 5 DCAL	*ALL*	*0xCA*	*0xD4*	*0*	*2*	*LSB, MSB*	*ACK packet*
Set Ch 6 DCAL	*ALL*	*0xCA*	*0xD5*	*0*	*2*	*LSB, MSB*	*ACK packet*
Set Ch 7 DCAL	*ALL*	*0xCA*	*0xD6*	*0*	*2*	*LSB, MSB*	*ACK packet*
Set HVS1 Cal	*ALL*	*0xCA*	*0xE0*	*0*	*3*	*Gain, offset LSB, offset MSB*	*ACK packet*
Set HVS2 Cal	*ALL*	*0xCA*	*0xE1*	*0*	*3*	*Gain, offset LSB, offset MSB*	*ACK packet*
Set HVS3 Cal	*ALL*	*0xCA*	*0xE2*	*0*	*3*	*Gain, offset LSB, offset MSB*	*ACK packet*
Upload packet (uC)	ALL	0x30	7	varies			ACK packet
Switch to uC image #1	ALL	0x30	9	0	4	0xA5 0xF1 CS MSB, LSB	ACK packet
Write 512-byte Misc Data	ALL	0xF0	0x09	2	0		ACK packet
Set Ethernet settings	ALL	0xF0	0x11	0	0x13	Fixed/Dyn, IP[4], Mask[4], Gateway[4], Dest. IP[4], Port[2]	ACK packet
Select RS232 baud rate	ALL	0xF0	0x13	0	1	Baud rate	ACK packet
Interface keep-alive - allow sharing	ALL	0xF0	0x20	0	0		ACK packet
Interface keep-alive - no sharing	ALL	0xF0	0x21	0	0		ACK packet
Interface keep-alive - lock	ALL	0xF0	0x22	0	0		ACK packet

Comm test - Request ACK packet	ALL	0xF1	0…15	0	0		ACK packet
Comm test - Streaming test mode	ALL	0xF1	0x7E	0	0 or 8	MINA MSB, LSB; MAXA MSB, LSB; INCR MSB, LSB; PERIOD MSB, LSB	ACK packet
Comm test - Echo packet	ALL	0xF1	0x7F	varies		Data to be echoed	Data packet

Items in italics are considered calibration values and don't normally need to be changed
* Initial firmware release which supports this packet type;

4.1.1 Request packet: "Request Status Packet"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	**0xF5**	**0xFA**	**1**	**1**	**0**	**0**	**0xFE**	**0x0F**

Purpose: This request directs the device to construct a status packet and send it to the host PC. The status packet can be requested with this request packet, or with one of the 'Request Spectrum plus Status' variants, in which case the status packet is included in the packet data field along with the spectrum data.

The status packet contains information about the current state of the spectrum, and of the device. Its format is listed with the "Status Packet" Response Packet.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of the requested data.

4.1.2 Request packet: "Request Misc Data"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	**0xF5**	**0xFA**	**3**	**2**	**0**	**0**	**0xFE**	**0x0C**

Purpose: This requests that the XRA700 send the 512-bytes of 'miscellaneous data' that is stored in the XRA700's non-volatile memory. (See also the "Write Misc Data" Request Packet for information on to write this data to the XRA700.)

Response: If no errors are detected, the XRA700 will respond with the "512-byte Misc Data" Response Packet (PID1=0x82, PID2=2).

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of the requested data.

4.1.3 Request packet: "Request Ethernet Settings"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	**0xF5**	**0xFA**	**3**	**4**	**0**	**0**	**0xFE**	**0x0A**

Purpose: This requests that the XRA700 send the Ethernet settings that are stored in the XRA700's non-volatile memory. (See also the "Set Ethernet Settings" Request Packet for information on how to write these settings.)

Response: If no errors are detected, the XRA700 will respond with the "Ethernet Settings" Response Packet (PID1=0x82, PID2=4).

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of the requested data. Additionally, if no Ethernet controller is detected on the XRA700, the "CP2201 not found" ACK packet will be returned instead of the requested data.

4.1.4 Request packet: "Request Diagnostic Data"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	**0xF5**	**0xFA**	**3**	**5**	**0**	**0**	**0xFE**	**0x09**

Purpose: This requests that the XRA700 perform some diagnostic tests, and return a suite of diagnostic data.

Response: If no errors are detected, the XRA700 will respond with the "Diagnostic Data" Response Packet (PID1=0x82, PID2=5). Note: the XRA700 runs a full memory test on the 512KB SRAM, which takes some time. The XRA700 will respond with the Response Packet a maximum of 2.5s after receiving the Request Packet.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of the requested data.

4.1.5 Request packet: "Request Netfinder Packet"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	**0xF5**	**0xFA**	**3**	**7**	**0**	**0**	**0xFE**	**0x07**

Purpose: This requests a Netfinder packet, but through the active communications port. [The 'Netfinder packet' is generally obtained via a broadcast on the TCP/IP Netfinder socket. See section 3.2.4 for details.]

Response: If no errors are detected, the XRA700 will respond with the "Netfinder Packet" Response Packet (PID1=0x82, PID2=8).

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of the requested data.

4.1.6 Request packet: "I²C Transfer"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6…5+LEN	6+LEN	7+LEN
Value	**0xF5**	**0xFA**	**3**	**8**	**varies**		**varies**	**varies**	

Purpose: The data field of this Request Packet contains the description for an I²C transfer to be performed by the XRA700. Read, Write and Combined I²C transfers are supported.

The format of the data field is listed below. The first 3 bytes are mandatory; following those are the bytes to be written, which can number from 0 to a maximum of 32. Therefore, the size of the data field (LEN) has a minimum of 3 and a maximum of 35.

Data Offset	Allowed Values	Value
0	1-127	7-bit I²C slave address ('SA')
1	0-32	Number of bytes to read ('NOBR')
2	0-32	Number of bytes to write ('NOBW')
3-34	0-255	Bytes to be written (optional)

Note: The XRA700 uses Slave Addresses of 0x20 and 0x4C. Transfers should not be attempted using these Slave Addresses. Also, the value in offset 0 is the 7-bit address – this value will be shifted left one bit, and the appropriate R/W bit merged in, before it's transmitted on the I²C bus.

Response: If no errors are detected, the XRA700 will respond either with an ACK OK packet (if NOBR=0), or with an I²C Response Packet with the bytes read from the I²C slave (if NOBR > 0).

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of the requested data or the ACK OK packet. If the I²C master didn't detect the I²C ACK at the proper time, an 'I²C ERROR' ACK packet will be returned. A BAD PARAMETER ACK packet will be returned if there are structure problems with the request. (NOBR=NOBW=0; LEN not equal to NOBW+3; NOBR > 32; NOBW > 32; or SA = 0).

4.1.7 Request packet: "Text Configuration"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6…5+LEN	6+LEN	7+LEN
Value	**0xF5**	**0xFA**	**0x20**	**2**	**varies**		**varies**	**varies**	

Purpose: This sends a Text Configuration packet to the XRA700, which the XRA700 interprets and executes. (See the section on ASCII Commands for information on individual commands.) The commands are packed into the Data field, with the following rules:

1. Alphabetic characters must be uppercase.
2. Each command consists of a 4-character command, followed by "=", followed by the parameter (max of 10 characters), and terminated by a semicolon (Ex: "TPEA=10;")
3. Units may be included with the parameter, but they are ignored, and count towards the 10-character maximum length for the parameter. (Ex: "TPEA=10US;")
4. The commands are packed together with no whitespace of any kind (tabs, spaces, CR/LF, etc.)
5. **For some commands, the ordering is important**. See the 'ASCII Commands' section for more discussion of this.
6. If all commands fit in the max 512 byte data field, then the configuration can be sent as a single packet. If they don't fit, then additional Text Configuration packets can be sent. Generally, the ASCII Command "RESC=Y;" would be the first command of the first packet, and it would not appear in later packets. If additional packets are needed, ensure that commands are not chopped between packets – only complete command/parameter strings are allowed.

   Here's an example of the data field:
   RESC=YES;BOOT=ON;PAVE=0;HVS1=700;HVS2=-130;HVS3=OFF;TECS=OFF;
   TEC1=220;TEC2=220;TEC3=230;TEC4=220;TEC5=220;TEC6=OFF;TEC7=OFF;

Note: After the corresponding ACK Packet is sent, the configuration is written into Flash memory. This process takes anywhere from 80-400mS, during which time further packet processing is stalled. So, if a Request Packet is received immediately after the Text Configuration Request Packet is acknowledged, the corresponding ACK packet may be delayed by up to 400mS.

Response: If no errors are detected, the XRA700 will respond with an ACK OK packet. This indicates that all commands were recognized and executed without error.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of the requested data. If one or more errors are detected while parsing the Text Configuration, an 'Unknown/Bad Parameter', 'Unrecognized Command', or 'PC5 Not Present' ACK packet will be returned, and the data field will contain the ASCII command and parameter which caused the error. If more than one error was detected, only the last error & command will be returned.

4.1.8 Request packet: "Text Configuration Readback"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6…5+LEN	6+LEN	7+LEN
Value	**0xF5**	**0xFA**	**0x20**	**3**	**varies**		**varies**	**varies**	

Purpose: This sends a Text Configuration Readback Request Packet to the XRA700, which the XRA700 uses as a 'template' to read back the current configuration. The data field is the same format as for the Text Configuration Request Packet – a list of ASCII command and parameters, separated by semicolons. The XRA700 strips off any parameters and replaces them with the current settings for each command specified. The result is returned in the Configuration Readback Response Packet. It is possible to send a text configuration packet via the Text Configuration Request Packet (PID1=0x30 / PID2=2), and then reuse the same packet data field for the Text Configuration Readback Request Packet (PID1=0x30 / PID2=3) to read back the actual settings that the XRA700 derived from the Text Configuration packet.

The "=" and parameter are optional for each command listed; the minimum required is the 4-character commands, separated by semicolons.

Also note that the 'RESC' (Reset Configuration) has no meaning in the context of Configuration Readback – if included, it will return 'RESC=?;'.

Any number of commands may be read back, as long as they fit within the maximum data field of 512 bytes. There is no error handling for the condition where the generated Response Packet exceeds the maximum data field size, so limiting the size of the Request Packet data field is a good idea, to avoid this issue.

Response: If no errors are detected, the XRA700 will respond with a Configuration Readback Response Packet (PID1=0x82 / PID2 = 7). Any command which is not recognized in the data field will return "CMDA=??;", where 'CMDA' is the unknown command. [This is returned in the Response Packet with the rest of the commands.]

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of the requested data. If one or more errors are detected while parsing the Text Configuration, an 'Unknown/Bad Parameter' or 'Unrecognized Command' ACK packet will be returned, and the data field will contain the ASCII command and parameter which caused the error. If more than one error was detected, only the last error & command will be returned.

4.1.9 Request packet: "Text Configuration (without saving to nonvolatile memory)"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6…5+LEN	6+LEN	7+LEN
Value	**0xF5**	**0xFA**	**0x20**	**4**	**varies**		**varies**	**varies**	

Purpose: This packet is identical to the 'Text Configuration' Request Packet (PID1/PID2=0x20/0x02), with one exception: the received configuration data is not written to non-volatile memory, so this configuration will not be the default when the device is next powered on.

Under certain circumstances, the delay in writing the data to Flash memory can be an issue, so this Request Packet allows that issue to be avoided. Section 4.1.8 discusses this in more detail.

Response: If no errors are detected, the XRA700 will respond with an ACK OK packet. This indicates that all commands were recognized and executed without error.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of the requested data. If one or more errors are detected while parsing the Text Configuration, an 'Unknown/Bad Parameter', 'Unrecognized Command', or 'PC5 Not Present' ACK packet will be returned, and the data field will contain the ASCII command and parameter which caused the error. If more than one error was detected, only the last error & command will be returned.

4.1.10 Request packet: "Set Ch [1 through 7] DCAL"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data	CHKSUM MSB	CHKSUM LSB	
Offset	0	1	2	3	4	5	6-7	8	9	Notes
Value	**0xF5**	**0xFA**	**0xCA**	**0xD0**	**0**	**2**	**DCAL LSB, MSB**	**varies**		**Ch 1**
	0xF5	**0xFA**	**0xCA**	**0xD1**	**0**	**2**	**DCAL LSB, MSB**	**varies**		**Ch 2**
	0xF5	**0xFA**	**0xCA**	**0xD2**	**0**	**2**	**DCAL LSB, MSB**	**varies**		**Ch 3**
	0xF5	**0xFA**	**0xCA**	**0xD3**	**0**	**2**	**DCAL LSB, MSB**	**varies**		**Ch 4**
	0xF5	**0xFA**	**0xCA**	**0xD4**	**0**	**2**	**DCAL LSB, MSB**	**varies**		**Ch 5**
	0xF5	**0xFA**	**0xCA**	**0xD5**	**0**	**2**	**DCAL LSB, MSB**	**varies**		**Ch 6**
	0xF5	**0xFA**	**0xCA**	**0xD6**	**0**	**2**	**DCAL LSB, MSB**	**varies**		**Ch 7**

Purpose: This commands the XRA700 to set the non-volatile calibration value for the detector temperature measurement diode. The setting is an unsigned integer in the range of 0-4095, in the format LSB followed by MSB. The nominal value is 1.000V, which corresponds to a setting of 0x555. (The conversion factor is 732uV/count.) This value can be read back via the "Request Diagnostic Packet" Request Packet. The setting is stored in the XRA700 non-volatile memory. If a PC5 is present, it is also stored in the PC5 non-volatile memory.

Response: If no errors are detected, the XRA700 will respond with ACK OK packet.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of ACK OK.

4.1.11 Request packet: "Set HVS1, HVS2, HVS3 Calibration"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data	CHKSUM MSB	CHKSUM LSB	NOTES
Offset	0	1	2	3	4	5	6-7	8	9	
Value	**0xF5**	**0xFA**	**0xCA**	**0xE0**	**0**	**2**	**Gain, Offset**	**varies**		**HVS1**
Value	**0xF5**	**0xFA**	**0xCA**	**0xE1**	**0**	**2**	**Gain, Offset**	**varies**		**HVS2**
Value	**0xF5**	**0xFA**	**0xCA**	**0xE2**	**0**	**2**	**Gain, Offset**	**varies**		**HVS3**

Purpose: This commands the XRA700 to set the non-volatile gain and offset calibration values for the HV DAC. These values can be read back via the "Request Diagnostic Packet" Request Packet.

Gain: If gain setting is 0-127: Correction = (1024+setting)/1024 (i.e. 100% to 112.4%)
If gain setting is 128-255: Correction = (768+setting)/1024 (i.e. 87.5% to 99.9%)
Offset: The offset setting is a signed byte, with a scale factor of 366mV/LSB.
Calibrated DAC Value = (Raw DAC Value * Gain) + Offset.

Response: If no errors are detected, the XRA700 will respond with ACK OK packet.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of ACK OK.

4.1.12 Request packet: "Write 512-byte Misc Data"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6-517	518	519
Value	**0xF5**	**0xFA**	**0xF0**	**9**	**2**	**0**	**Misc Data**	**varies**	

Purpose: This commands the XRA700 to store the 512-byte data field in non-volatile memory. The data can be binary, ASCII, etc. (It could contain the description or location of the unit as an ASCII string, for example.) It is read back with the "Request Misc Data" Request Packet.

Response: If no errors are detected, the XRA700 will respond with ACK OK packet. Note this command requires a FLASH memory erase operation – the ACK packet could be delayed as much as 400mS from the request packet.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of ACK OK.

4.1.13 Request packet: "Set Ethernet Settings"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6-24	25	26
Value	**0xF5**	**0xFA**	**0xF0**	**0x11**	**0**	**0x13**	**Ethernet settings**	**varies**	

Purpose: This commands the XRA700 to set the IP address, etc. for the Ethernet controller. The current values can be read back via the "Request Ethernet Settings" Request Packet.

The format of the data field is:

Data Offset	Value
0	0 = Use fixed IP address; 0xFF = Get IP address from DHCP server
1-4	IP Address (MSB…LSB)
5-8	IP Subnet Mask (MSB…LSB)
9-12	IP Gateway Address (MSB…LSB)
13-16	Reserved
17-18	Port (MSB,LSB: fixed at 10001)

Response: If no errors are detected, the XRA700 will respond with ACK OK packet.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of ACK OK. Additionally, if no Ethernet controller is detected on the XRA700, the "CP2201 not found" ACK packet will be returned instead of ACK OK.

4.1.14 Request packet: "Select RS232 Baud Rate"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7	8
Value	**0xF5**	**0xFA**	**0xF0**	**0x13**	**0**	**1**	**Baud Rate**	**varies**	

Purpose: This selects the RS232 baud rate. The default is 115,200 baud. 57,600 baud or 19,200 baud can be selected; however, these setting are incompatible with the Ethernet stack. Therefore, **selecting 57,600 baud or 19,200 baud will disable the Ethernet interface**. When this value is changed, the new setting will not take effect until the next power cycle. The interpretation of the baud rate value ('setting') is:

0 => 115,200 baud;
1 => 57,600 baud;
2 => 19,200 baud;

Response: If no errors are detected, the XRA700 will respond with ACK OK packet. If the data byte is outside the range of 0-2, then a BAD PARAMETER error is returned.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of ACK OK.

4.1.15 Request packet: "Interface Keep-alive – Allow Sharing"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	**0xF5**	**0xFA**	**0xF0**	**0x20**	**0**	**0**	**0xFD**	**0x01**

Purpose: This Request Packet is used to keep the XRA700's UDP socket bound to the IP address and source port of the computer connected to it. The UDP socket resets itself after 15 seconds of inactivity (after which any IP address can connect to it), so one of the "Interface Keep-alive" Request Packets should be sent periodically if there is no other periodic traffic, or if the periodic traffic is less frequent than ~10 seconds. [If the host wishes to change the Keep-alive status (sharing/no sharing/locked), then a Keep-alive packet should be sent with the new status, even if there is periodic traffic keeping the socket bound.]

The "Allow Sharing" variant of the "Interface Keep-alive" Request Packet tells the XRA700 to return the "Sharing Allowed" status in the Netfinder device discovery packet. It also commands the XRA700 to accept "Interface Sharing Request" packets on the Netfinder socket, and to return an "Ethernet Sharing Request" ACK packet via the normal UDP socket if such a packet is received. [This ACK packet will replace an OK ACK packet at the next opportunity.]

[This Request Packet only applies to Ethernet. In future firmware, it may be expanded to include the other interfaces.]

Response: If no errors are detected, the XRA700 will respond with ACK OK packet.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of ACK OK.

4.1.16 Request packet: "Interface Keep-alive – No Sharing"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	**0xF5**	**0xFA**	**0xF0**	**0x21**	**0**	**0**	**0xFD**	**0x00**

Purpose: This Request Packet is used to keep the XRA700's UDP socket bound to the IP address and source port of the computer connected to it. The UDP socket resets itself after 15 seconds of inactivity (after which any IP address can connect to it), so one of the "Interface Keep-alive" Request Packets should be sent periodically if there is no other periodic traffic, or if the periodic traffic is less frequent than ~10 seconds. [If the host wishes to change the Keep-alive status (sharing/no sharing/locked), then a Keep-alive packet should be sent with the new status, even if there is periodic traffic keeping the socket bound.]

The "No Sharing" variant of the "Interface Keep-alive" Request Packet tells the XRA700 to return the "No Sharing" status in the Netfinder device discovery packet. It also commands the XRA700 to ignore "Interface Sharing Request" packets on the Netfinder socket.

[This Request Packet only applies to Ethernet. In future firmware, it may be expanded to include the other interfaces.]

Response: If no errors are detected, the XRA700 will respond with ACK OK packet.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of ACK OK.

4.1.17 Request packet: "Interface Keep-alive – Lock"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	**0xF5**	**0xFA**	**0xF0**	**0x22**	**0**	**0**	**0xFC**	**0xFF**

Purpose: This Request Packet is used to keep the XRA700's UDP socket bound to the IP address and source port of the computer connected to it. Normally, the UDP socket resets itself after 15 seconds of inactivity (after which any IP address can connect to it), but it will not automatically reset itself after receiving this Request Packet (so periodic transfers are no longer required to keep the socket bound.) [If the host wishes to change the Keep-alive status (sharing/no sharing/locked), then a Keep-alive packet should be sent with the new status]

The "Lock" variant of the "Interface Keep-alive" Request Packet tells the XRA700 to return the "Lock" status in the Netfinder device discovery packet. It also commands the XRA700 to ignore "Interface Sharing Request" packets on the Netfinder socket, and to not automatically reset the socket. Care must be taken in using this; the UDP socket binds to the IP Address and source port of the host, so the XRA700 will then only communicate with the same IP & source port. The host IP address may change if the host uses a dynamic IP address; the source port may change if the host wishes to communicate with multiple XRA700s simultaneously, or doesn't set the source port explicitly. If the host loses the ability to communicate with the socket, then the power will have to be cycled on the XRA700 to reset the socket.

[This Request Packet only applies to Ethernet. In future firmware, it may be expanded to include the other interfaces.]

Response: If no errors are detected, the XRA700 will respond with ACK OK packet.

Possible errors: If the request packet is malformed or corrupted: CHKSUM, LEN, or PID error ACK packets can be returned instead of ACK OK.

4.2 Response Packets

Table 2– Text highlighted in Blue is new/modified since Rev A6 of the Programmer's Guide

Description	Support*	PID1	PID2	LEN MSB	LEN LSB	Data (optional)
Status Packet	ALL	0x80	3	0	0x64	See description
512-byte misc data packet	ALL	0x82	2	2	0	See description
Ethernet settings	ALL	0x82	4	0	0x17	See description
Diagnostic data	ALL	0x82	5	1	0	See description
Configuration readback packet	ALL	0x82	7	varies		
Netfinder packet	ALL	0x82	8	varies		
I²C Read Data	ALL	0x82	9	varies		See description
Comm test - Echo packet	ALL	0x8F	0x7F	varies		Echo data

* Initial firmware release which supports this packet type

4.2.1 Response packet: "Status Packet"

Offset	Data bits	Allowed value	Description	Notes
0	D7-D0	0xA7	Device Type	from "XRA7"
1	D7-D0	0x06	Firmware Major_version	BCD
2	D7-D0	0-255	Firmware Minor_version	0.0 BCD
3	D7-D0		Serial Number LSB	
4	D7-D0		Serial Number byte 2	
5	D7-D0		Serial Number byte 3	
6	D7-D0	0-255	Serial Number MSB	LSB first (to match XRA700)
7	D7-D0	0-255	Date BCD month digits (MM)	Not implemented
8	D7-D0	0-255	Date BCD day digits (DD)	Not implemented
9	D7-D0	0-255	Date BCD Year Digits (YY)	Not implemented
10	D7-D0	0-255	Time BCD Hour Digits (HH)	Not implemented
11	D7-D0	0-255	Time BCD Minute Digits (MM)	Not implemented
12	D7-D0	0-255	Time BCD Seconds Digits (SS)	Not implemented
13	D7-D0		Feature Set	TBD
14	D7		Global XRA_STATUS Autoboot	0 = off, 1 = on
	D6		Global XRA_STATUS HV Enable	0 = off, 1 = on
	D5		Global XRA_STATUS TEC Enable	0 = off, 1 = on
	D4		Global XRA_STATUS Preamp Power Enable	0 = off, 1 = on
	D3		Global XRA_STATUS Fan Enable	0 = off, 1 = on
	D2-D0		Unused	
15	D7-D0	0-255	Uptime	Not implemented
16	D7-D0	0-255	SYSTEM (chan 0) LED_STATE	Green = 1, Red = 2
17	D7-D0	0-255	Channel 1 status	Channel States: {0: INIT 1: COOLING 2: PREP 3: READY 4: FAULT 5: DISABLED}
18	D7-D0	0-255	Channel 2 status	
19	D7-D0	0-255	Channel 3 status	
20	D7-D0	0-255	Channel 4 status	
21	D7-D0	0-255	Channel 5 status	
22	D7-D0	0-255	Channel 6 status	
23	D7-D0	0-255	Channel 7 status	
24	D7-D0		Channel 8 status (not used)	
25	D7-D0	0-255	House Keeping MON_5N MSB	(MSB*256)+LSB = mV
26	D7-D0	0-255	House Keeping MON_5N LSB	
27	D7-D0	0-255	House Keeping MON_3.3V MSB	
28	D7-D0	0-255	House Keeping MON_3.3V LSB	
29	D7-D0	0-255	House Keeping MON_5P MSB	
30	D7-D0	0-255	House Keeping MON_5P LSB	
31	D7-D0	0-255	House Keeping MON_TEC MSB	
32	D7-D0	0-255	House Keeping MON_TEC LSB	
33	D7-D0	0-255	CH 1 HV Monitor, MSB	((MSB*256)+LSB)/2 = volts

34	D7-D0	0-255	CH 1 HV Monitor, LSB	
35	D7-D0	0-255	CH 2 HV Monitor, MSB	
36	D7-D0	0-255	CH 2 HV Monitor, LSB	
37	D7-D0	0-255	CH 3 HV Monitor, MSB	
38	D7-D0	0-255	CH 3 HV Monitor, LSB	
39	D7-D0	0-255	CH 4 HV Monitor, MSB	
40	D7-D0	0-255	CH 4 HV Monitor, LSB	
41	D7-D0	0-255	CH 5 HV Monitor, MSB	
42	D7-D0	0-255	CH 5 HV Monitor, LSB	
43	D7-D0	0-255	CH 6 HV Monitor, MSB	
44	D7-D0	0-255	CH 6 HV Monitor, LSB	
45	D7-D0	0-255	CH 7 HV Monitor, MSB	
46	D7-D0	0-255	CH 7 HV Monitor, LSB	
47	D7-D0	0-255	CH 8 HV Monitor, MSB	Not used
48	D7-D0	0-255	CH 8 HV Monitor, LSB	Not used
49	D7-D4			Not used
	D3-D0	0-255	CH 1 detector temperature MSB	0.1 degree Kelvin/cnt
50	D7-D0	0-255	CH 1 detector temperature LSB	
51	D7-D4			Not used
	D3-D0	0-255	CH 2 detector temperature MSB	
52	D7-D0	0-255	CH 2 detector temperature LSB	
53	D7-D4			Not used
	D3-D0	0-255	CH 3 detector temperature MSB	
54	D7-D0	0-255	CH 3 detector temperature LSB	
55	D7-D4			Not used
	D3-D0	0-255	CH 4 detector temperature MSB	
56	D7-D0	0-255	CH 4 detector temperature LSB	
57	D7-D4			Not used
	D3-D0	0-255	CH 5 detector temperature MSB	
58	D7-D0	0-255	CH 5 detector temperature LSB	
59	D7-D4			Not used
	D3-D0	0-255	CH 6 detector temperature MSB	
60	D7-D0	0-255	CH 6 detector temperature LSB	
61	D7-D4			Not used
	D3-D0	0-255	CH 7 detector temperature MSB	
62	D7-D0	0-255	CH 7 detector temperature LSB	
63	D7-D4			Not used
	D3-D0	0-255	CH 8 detector temperature MSB	Not used
64	D7-D0	0-255	CH 8 detector temperature LSB	Not used
65	D7-D4			Not used
	D3-D0	0-15	CH 1 TEC V Monitor, MSB	value in mV
66	D7-D0	0-255	CH 1 TEC V Monitor, LSB	
67	D7-D4			Not used
	D3-D0	0-15	CH 2 TEC V Monitor, MSB	

68	D7-D0	0-255	CH 2 TEC V Monitor, LSB	
69	D7-D4			Not used
	D3-D0	0-15	CH 3 TEC V Monitor, MSB	
70	D7-D0	0-255	CH 3 TEC V Monitor, LSB	
71	D7-D4			Not used
	D3-D0	0-15	CH 4 TEC V Monitor, MSB	
72	D7-D0	0-255	CH 4 TEC V Monitor, LSB	
73	D7-D4			Not used
	D3-D0	0-15	CH 5 TEC V Monitor, MSB	
74	D7-D0	0-255	CH 5 TEC V Monitor, LSB	
75	D7-D4			Not used
	D3-D0	0-15	CH 6 TEC V Monitor, MSB	
76	D7-D0	0-255	CH 6 TEC V Monitor, LSB	
77	D7-D4			Not used
	D3-D0	0-15	CH 7 TEC V Monitor, MSB	
78	D7-D0	0-255	CH 7 TEC V Monitor, LSB	
79	D7-D4			Not used
	D3-D0		CH 8 TEC V Monitor, MSB	Not used
80	D7-D0		CH 8 TEC V Monitor, LSB	Not used
81	D7-D0	-128… +127	Control Board temp sensor (DS1775)	1 °C/count
82	D7-D0	_	Current Monitor, MSB	Not implemented
83	D7-D0	_	Current Monitor, LSB	Not implemented
84	D7-D4	_		
	D3-D0	0 to 800	HV1 Supply setting MSB	12-bit signed value in V
85	D7-D0		HV1 Supply setting LSB	
86	D7-D4	_		Not used
	D3-D0	-200 to 0	HV2 Supply setting MSB	12-bit signed value in V
87	D7-D0		HV2 Supply setting LSB	
88	D7-D4			Not used
	D3-D0	-500 to 0	HV3 Supply setting MSB	12-bit signed value in V
89	D7-D0		HV3 Supply setting LSB	
90	D7-D0	0	HV4 Supply setting MSB	not used
91	D7-D0	0 … 3	HV4 Supply setting LSB	not used
92	D7-D4	0, 1, 2, 15	HVSEL_2	Value of Nibble: 0: Channel to HVCube 1 1: Channel to HVCube 2 2: Channel to HVCube 3 15: Channel not connected to HV
	D3-D0	0, 1, 2, 15	HVSEL_1	
93	D7-D4	0, 1, 2, 15	HVSEL_4	
	D3-D0	0, 1, 2, 15	HVSEL_3	
94	D7-D4	0, 1, 2, 15	HVSEL_6	
	D3-D0	0, 1, 2, 15	HVSEL_5	
95	D7-D4	15	HVSEL_8 (Not Used)	

	D3-D0	0, 1, 2, 15	HVSEL_7	
96	D7-D0	-128… +127	Heat sink temp (1 °C/count, signed)	1 °C/count
97			Fan status	TBD
98	D7-D0	0-255	ECO Byte	TBD
99			Unused	TBD

4.2.2 Response packet: "512-byte Misc Data Packet"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data Field	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6-517	518	519
Value	0xF5	0xFA	**0x82**	**2**	2	0	Misc data	Varies	

Purpose: This response packet is returned in response to a 'Request Misc Data' packet (PID1=3, PID2=2). [See the 'Write Misc Data' Request Packet for information on writing the Misc Data.]

Data Format: The data field contains 512 bytes, which is the Misc Data programmed via a 'Write Misc Data' request packet. The data can be binary, ASCII, etc. This is stored in non-volatile memory, so it remains after the power is cycled.

4.2.3 Response packet: "Ethernet Settings"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data Field	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6-30	31	32
Value	0xF5	0xFA	**0x82**	**4**	0	0x19	Ethernet settings	Varies	

Purpose: This response packet is returned in response to a 'Request Ethernet Settings' packet (PID1=3, PID2=4).

Data Format: The data field contains 23 bytes of Ethernet Settings, in the following format:

Offset	Value
0	0 = Use fixed IP address; 0xFF = Get IP address from DHCP server
1-4	IP Address (MSB…LSB)
5-8	IP Subnet Mask (MSB…LSB)
9-12	IP Gateway Address (MSB…LSB)
13-16	Reserved
17-18	Port (MSB,LSB: fixed at 10001 for now)
19-24	XRA700 MAC Address

4.2.4 Response packet: "Diagnostic Data"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data Field	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6-261	262	263
Value	0xF5	0xFA	**0x82**	**5**	1	0	Diagnostic Data	Varies	

Purpose: This response packet is returned in response to a 'Request Diagnostic Data' Request Packet (PID1=3, PID2=5).

Data Format: The data field contains 256 bytes of various types of diagnostic data. This is not yet implemented.

4.2.5 Response packet: "Configuration Readback"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	Data Field	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6…5+LEN	6+LEN	7+LEN
Value	0xF5	0xFA	**0x82**	**7**	varies		Configuration readback (ASCII)	varies	

Purpose: This response packet is returned in response to a 'Configuration Readback' Request Packet (PID1=0x20, PID2=3). The data field contains ASCII commands and parameters, separated by semicolons.

Data Format: The data field will contain the current ASCII configuration – it will have all the commands that were sent with the "Configuration Readback" Request Packet, except the parameters will have been replaced with the actual settings in use by the XRA700. See the "Text Configuration" Request Packet, "Configuration Readback" Request packet, and the section on ASCII Commands for more details.

4.2.6 Response packet: "Netfinder Packet"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	DATA FIELD	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6…5+LEN	6+LEN	7+LEN
Value	**0xF5**	**0xFA**	**0x82**	**8**	**varies**		**Netfinder data**	**varies**	

Purpose: This is used to return the 'Netfinder' packet via the normal communications link. [Normally, the Netfinder data is returned by a dedicated TCP/IP socket.]

Data Format: The data field is formatted the same as an independent Netfinder packet, as documented in section 3.2.4.

4.2.7 Response packet: "I2C Read Data"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	DATA FIELD	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6…5+LEN	6+LEN	7+LEN
Value	**0xF5**	**0xFA**	**0x82**	**9**	**varies**		**I²C read data**	**varies**	

Purpose: This is used to return the data read by the 'I^{2}C Transfer' Request Packet. If the Number of Bytes to Read ('NOBR') in the 'I^{2}C Transfer' Request Packet is nonzero, and the I^{2}C transfer was successful, then the 'I^{2}C Read Data' Response Packet is returned, with the data field containing the bytes read by the transfer. [Note that if NOBR = 0, the XRA700 will return an ACK OK packet rather than an 'I^{2}C Read Data' packet with LEN=0.]

Data Format: The LEN value will be the same as NOBR, and the data field will contain the bytes read, in the order that they were read.

4.2.8 Response packet: "Comm test - Echo packet"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	DATA FIELD	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6…5+LEN	6+LEN	7+LEN
Value	**0xF5**	**0xFA**	**0x8F**	**0x7F**	**varies**		**Echo data**	**varies**	

Purpose: This Response Packet is sent in response to the Request Packet of the same name. The LEN & DATA fields are the same as in the Request Packet.

Data Format: Whatever data is in the data field of the Request Packet is copied, byte-for-byte, into the data field of the Response packet.

4.3 Acknowledge Packets

Table 3–

Description	Support*	PID1	PID2	LEN MSB	LEN LSB	Data (optional)
OK	ALL	0xFF	0	0	0	
Sync error	ALL	0xFF	1	0	0	
PID error	ALL	0xFF	2	0	0	
LEN error	ALL	0xFF	3	0	0	
Checksum error	ALL	0xFF	4	0	0	
Bad parameter	ALL	0xFF	5	varies		ASCII command
Bad hex record (structure/chksum)	ALL	0xFF	6	0	0	
Unrecognized command	ALL	0xFF	7	varies		ASCII command
CP2201 not found	ALL	0xFF	9	0	0	
Scope data not available (not triggered)	ALL	0xFF	0x0A	0	0	
PC5 not present	ALL	0xFF	0x0B	varies		ASCII command
OK + Interface sharing request	ALL	0xFF	0x0C	0	0	
Busy - another interface is in use		0xFF	0x0D	0	0	
I2C error	ALL	0xFF	0x0E	0	0	
Calibration data not present	ALL	0xFF	0x11	0	0	

* Initial firmware release which supports this packet type

4.3.1 Acknowledge packet: "OK"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**0**	0	0	0xFD	0x12

Purpose: This ACK packet indicates that the Request Packet was received without error, that its contents were recognized, and that the requested action was carried out. This ACK packet is only returned by Request Packets which do not request data.

4.3.2 Acknowledge packet: "OK, with Interface Sharing Request"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**0x0C**	0	0	0xFD	0x06

Purpose: This ACK packet has the same meaning as the "OK" ACK packet. In addition, another computer or interface is requesting that activity be paused so that it can gain access to the XRA700. To allow it access, the application should stop sending traffic to the XRA700. To maintain control (and deny it access), continue sending Request Packets. (Send 'Interface keep-alive' packets periodically, if no other periodic traffic is occurring. See the Interface section for more details.)

4.3.3 Acknowledge packet: "Sync Error"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**1**	0	0	0xFD	0x11

Purpose: This ACK packet indicates that the sync bytes in Request Packet were not correct, and therefore, the Request Packet was rejected.

Note: This error condition can only be generated via USB or Ethernet; RS232 uses the sync bytes to locate a packet in the RS232 data stream. In the absence of the sync byte pattern, no Request Packet is detected and thus no ACK packet can be returned.

4.3.4 Acknowledge packet: "PID Error"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**2**	0	0	0xFD	0x10

Purpose: This ACK packet indicates that PID1 & PID2 combination is not recognized as a valid Request Packet, and therefore, the Request Packet was rejected.

4.3.5 Acknowledge packet: "LEN Error"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**3**	0	0	0xFD	0x0F

Purpose: This ACK packet indicates that the LEN field of the Request Packet was not consistent with Request Packet type defined by the PID1 & PID2 combination. It is not recognized as a valid Request Packet, and therefore, the Request Packet was rejected.

4.3.6 Acknowledge packet: "Checksum Error"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**4**	0	0	0xFD	0x0E

Purpose: This ACK packet indicates that the checksum of the Request Packet was incorrect, and therefore, the Request Packet was rejected.

4.3.7 Acknowledge packet: "Bad Parameter"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	DATA	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6...5+LEN	6+LEN	7+LEN
Value	0xF5	0xFA	**0xFF**	**5**	0	varies	ASCII Command Echo	varies	

4.3.8 Acknowledge packet: "Unrecognized Command"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	DATA	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6...5+LEN	6+LEN	7+LEN
Value	0xF5	0xFA	**0xFF**	**7**	0	varies	ASCII Command Echo	varies	

4.3.9 Acknowledge packet: "PC5 Not Present"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	DATA	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6...5+LEN	6+LEN	7+LEN
Value	0xF5	0xFA	**0xFF**	**0x0B**	0	varies	ASCII Command Echo	varies	

Purpose: These ACK packets are in response to ASCII command errors – the data field will contain the ASCII command and parameter which caused the error. "Bad Parameter" means that the parameter isn't recognized or exceeds the range of the command. "Unrecognized Command" means that the 4-character command isn't recognized. "PC5 Not Present" is returned if a PC5 is not mated to the XRA700, and a command requiring a PC5 is sent. (i.e. "HVSE", Set High Voltage.) [A 'Bad Parameter' ACK packet may also be returned for a malformed I2C Request Packet, in which case LEN=0.]

If an incomplete or garbled command is returned in the data field, it may mean that the ASCII Configuration Packet has structural issues. (Disallowed whitespace, missing semicolon, etc.)

4.3.10 Acknowledge packet: "Bad Hex Record"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**6**	0	0	0xFD	0x0C

Purpose: This ACK packet applies to microcontroller upload packets (PID1=0x30, PID2=2 or 7.) It means that the hex record contained in the data field of the Request Packet had a checksum or other structural error.

4.3.11 Acknowledge packet: "CP2201 Not Found"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**9**	0	0	0xFD	0x09

Purpose: This ACK packet indicates that the "Set Ethernet Settings" (PID1=0xF0, PID2=0x11) was received, but an Ethernet controller was not detected on the XRA700. [Some early XRA700s were manufactured without an Ethernet controller.]

4.3.12 Acknowledge packet: "Scope Data Not Available"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**0x0A**	0	0	0xFD	0x08

Purpose: This ACK packet indicates that a "Send Scope Data" (PID1=3, PID2=1 or 3) was received, but the digital oscilloscope hasn't triggered, so no data is available. [The digital oscilloscope must be armed, and then a trigger must occur for data to be available.]

4.3.13 Acknowledge packet: "I²C Error"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**0x0E**	0	0	0xFD	0x04

Purpose: This ACK packet indicates that the I^2C Bus Master attempted a transfer as described by the 'I^2C Transfer' Request Packet, but no I^2C ACK was detected from an I^2C Slave.

4.3.14 Acknowledge packet: "Calibration data not present"

	SYNC1	SYNC2	PID1	PID2	LEN MSB	LEN LSB	CHKSUM MSB	CHKSUM LSB
Offset	0	1	2	3	4	5	6	7
Value	0xF5	0xFA	**0xFF**	**0x11**	0	0	0xFD	0x01

Purpose: This ACK packet indicates that a Request Packet has been recognized as valid by the firmware, but that the requested calibration data has not been previously programmed into the device.

5 ASCII Commands

In the following section, the format is specified for all the ASCII commands recognized by the XRA700. These commands are sent to the XRA700 in the 'Text Configuration' Request Packet (PID1=0x20, PID2=2). The commands can be sent singly, but most often, they are grouped together into a single Text Configuration packet so that the entire configuration can be sent at once.

Generally, when the entire configuration is sent at once, the first command should be "RESC=Y". This resets the configuration to a known state before applying the configuration commands. This is not automatically done because it may be possible that the required configuration is larger than can be sent in a single 'Text Configuration' Request Packet. If this is the case, then "RESC=Y" and as many commands as will fit should be sent in the first Text Configuration packet, with the remaining commands in later packets. Also, it may be desirable to send individual commands to fine-tune the configuration.

This is discussed in the 'Text Configuration' Request Packet definition, but it's worth mentioning again: all ASCII characters must be upper-case; no whitespace is allowed (i.e. no spaces, tabs, CR-LF, etc.); and the commands are separated by semicolons.

Some commands have dependencies; they require other commands to be listed before them. For example, at least one Thermoelectric Cooler command (TEC#) needs to be turned on before the high voltage power supplies (HVS#) are turned on for system stability. If a dependent command is specified, that command must occur prior to the command which lists the dependency. The table below specifies the 'Order' for each command – lower Order commands must be sent before higher Order commands.

5.1 Table 4 – ASCII Command Summary

Description	Support*	Command	Parameter	Units	Default	ORDER	Limits
Turn Default Load on when loading config at startup	ALL	BTDL	[ON|OF{F}]		ON		
Turn TEC on when loading config at startup	ALL	BTEC	[ON|OF{F}]		ON		
Turn Fan on when loading config at startup	ALL	BTFN	[ON|OF{F}]		OFF		
Turn on HVPS when loading config at startup	ALL	BTHV	[ON|OF{F}]		ON		
Boot mode select	ALL	BTMD	[AUTO|DEF{AULT}|DEL{AY}]		DEF		
Turn on Preamp Power Supplies when loading config at startup	ALL	BTPA	[ON|OF{F}]		ON		
Channel 1 enable	ALL	C1EN	[ON|OF{F}]		ON		
Channel 2 enable	ALL	C2EN	[ON|OF{F}]		ON		
Channel 3 enable	ALL	C3EN	[ON|OF{F}]		ON		
Channel 4 enable	ALL	C4EN	[ON|OF{F}]		ON		
Channel 5 enable	ALL	C5EN	[ON|OF{F}]		ON		
Channel 6 enable	ALL	C6EN	[ON|OF{F}]		ON		
Channel 7 enable	ALL	C7EN	[ON|OF{F}]		ON		
Default Load turn on	ALL	ENDL	[ON|OF{F}]		ON		
Turn on the fan	ALL	FANE	[ON|OF{F}]		OFF	3	
HV set, supply #1	ALL	HVS1	[{+|-}###|OF{F}]	[V]	OFF	2	0 to 800V (+HV)
HV set, supply #2	ALL	HVS2	[{+|-}###|OF{F}]	[V]	OFF	2	0 to -200V (-HV)
HV set, supply #3	ALL	HVS3	[{+|-}###|OF{F}]	[V]	OFF	2	0 to -500V (-HV)
HV FET turn on	ALL	HVSE	[ON|OF{F}]		OFF	2	
PreAmp Power Supply	ALL	PAVE	[ON|OF{F}]		OFF	2	
Reset Configuration	ALL	RESC	[Y{ES}|NO]				
TEC set,	ALL	TEC1	[###|OF{F}]	[K]	OFF	1	0-299K

channel 1					(300)		
TEC set, channel 2	ALL	TEC2	[###|OF{F}]	[K]	OFF	1	0-299K
TEC set, channel 3	ALL	TEC3	[###|OF{F}]	[K]	OFF	1	0-299K
TEC set, channel 4	ALL	TEC4	[###|OF{F}]	[K]	OFF	1	0-299K
TEC set, channel 5	ALL	TEC5	[###|OF{F}]	[K]	OFF	1	0-299K
TEC set, channel 6	ALL	TEC6	[###|OF{F}]	[K]	OFF	1	0-299K
TEC set, channel 7	ALL	TEC7	[###|OF{F}]	[K]	OFF	1	0-299K
TEC set, all channels	ALL	TECS	[###|OF{F}]	[K]	OFF	1	0-299K
TEC Enable	ALL	TECE	[ON|OF{F}]		OFF	1	

*Support: Minimum firmware (FW) version to support command, plus which devices support it;

5.1.1 BTDL – Determine Default Load status on configuration load

Command: BTDL
Parameter: [ON|OF{F}]
Units: none
Default: ON

Dependencies: none
Supported: All
Related: BTEC, BTFN, BTHV, BTMD, BTPA

Description: BTDL is used to determine whether the Default Load will be turned on during a boot up configuration load. It is recommended to leave the Default load on, but you may get slight power savings by turning it off.

Errors: An INVALID PARAMETER error will be returned if the parameter is not one of those listed.

Example: BTDL=ON; // Turn default load on during configuration loads

5.1.2 BTEC – Determine TEC status on configuration load

Command: BTEC
Parameter: [ON|OF{F}]
Units: none
Default: ON

Dependencies: none
Supported: All
Related: BTDL, BTFN, BTHV, BTMD, BTPA

Description: BTEC is used to determine whether the TEC will be turned on during a boot up configuration load. If it is turned on, each TEC channel will be loaded to the value that was saved to the configuration.

Errors: An INVALID PARAMETER error will be returned if the parameter is not one of those listed.

Example: BTEC=ON; // Turn TEC on during configuration loads

5.1.3 BTFN – Determine fan status on configuration load

Command: BTFN
Parameter: [ON|OF{F}]
Units: none
Default: OFF

Dependencies: none
Supported: All
Related: BTDL, BTEC, BTHV, BTMD, BTPA

Description: BTFN is used to determine whether the fan will be turned on during a boot up configuration load.

Errors: An INVALID PARAMETER error will be returned if the parameter is not one of those listed.

Example: BTFN=OFF; // Turn Fan off during configuration loads

5.1.4 BTHV – Determine high voltage power supply status on configuration load

Command: BTHV
Parameter: [ON|OF{F}]
Units: none
Default: ON

Dependencies: none
Supported: All
Related: BTDL, BTEC, BTFN, BTMD, BTPA, HVS{X}, HVSE

Description: BTHV is used to determine whether the high voltage power supplies will be turned on during a boot up configuration load. If it is turned on, each high voltage channel will be loaded to the value that was saved in the configuration.

Errors: An INVALID PARAMETER error will be returned if the parameter is not one of those listed.

Example: BTHV=ON; // Turn HV power supplies on during configuration loads

5.1.5 BTMD – Boot Mode. Set the behavior of the device on power up.

Command: BTMD
Parameter: [DEF{AULT}|DEL{AY}|AUT{O}]
Units: none
Default: DEFAULT

Dependencies: none
Supported: All
Related: BTDL, BTEC, BTFN, BTHV, BTPA

Description: BTMD is used to set the behavior of the XRA700 unit on power up. The XRA700 Pleiades Array has 3 boot modes. These modes are described in the table below:

Mode	Command	Description
Default	BTMD=DEF;	Turning on the array, no configuration is loaded. However, if the "MODE" ring LED button is pressed during power up, the previous configuration will be loaded
Delay	BTMD=DEL;	Turning on the array, no configuration is loaded. If the "MODE" Ring LED button is pressed at any time, the previous configuration will be loaded
Automatic	BTMD=AUTO;	Turning on the array, the last used configuration will be loaded automatically. If the "MODE" ring LED button is held during power up, no configuration will be loaded.

The "MODE" button on the device is a 2 color LED. If a previous configuration was loaded, the LED ring will be green. If no previous configuration was loaded, the LED ring will be red.

Errors: An INVALID PARAMETER error will be returned if the parameter is not one of those listed.

Example: BTMD=AUTO // Change boot mode to automatically load the previously written configuration

5.1.6 BTPA – Determine preamplifier power supply status on configuration load.

Command: BTPA
Parameter: [ON|OF{F}]
Units: none
Default: ON

Dependencies: none
Supported: All
Related: BTDL, BTEC, BTFN, BTHV, BTMD, PAVE

Description: BTPA is used to determine whether the preamplifier power supplies will be turned on during a boot up configuration load.

Errors: An INVALID PARAMETER error will be returned if the parameter is not one of those listed.

Example: BTPA=ON; // Turn PA power supplies on during configuration loads

5.1.7 C1EN, C2EN, C3EN, C4EN, C5EN, C6EN, C7EN – Channel Enable

Command: C{X}EN; X= {1,2,3,4,5,6,7}
Parameter: [ON|OF{F}]
Units: none
Default: ON

Dependencies: none
Supported: All
Related: TEC{X}

Description: CXEN is the channel enable setting. In the event of a known detector problem, a user may want to disable that channel to prevent power loss while cooling a faulty detector and to prevent the channel LED from blinking while the device tries to cool the detector. A disabled channel will have an Amber LED, will read back DISABLE as its state, and will be unable to have a temperature setting until it is enabled again via this ASCII command.

Errors: An INVALID PARAMETER error will be returned if the parameter is not one of those listed.

Example: C2EN=OFF; // Disable detector in channel 2

5.1.8 ENDL – Default Load Enable

Command: ENDL
Parameter: [ON|OF{F}]
Units: none
Default: ON

Dependencies: none
Supported: All
Related: BTDL

Description: The default load is a feature for stability of the system before TECs turn on. It will take care of itself, but ENDL acts as a manual on/off switch.

Errors: An INVALID PARAMETER error will be returned if the parameter is not one of those listed.

Example: ENDL=OFF; // Disable default load

5.1.9 FANE – Fan Enable

Command: FANE
Parameter: [ON|OF{F}]
Units: none
Default: OFF

Dependencies: none
Supported: All
Related: BTFN

Description: Turns the fan on or off.

Errors: An INVALID PARAMETER error will be returned if the parameter is not one of those listed.

Example: FANE=ON; // Turn the fan on

5.1.10 HVS1, HVS2, HVS3 – High Voltage Supply Settings

Command: HVS#. #= {1,2,3}
Parameter: [{+|-}####|OFF]
Units: none
Default: 0

Dependencies: none
Supported: All
Related: BTHV, HVSE

Description: HVS1, HVS2, and HVS3 control the settings for the 3 possible high voltage power supplies. These supplies may or may not be installed depending on the configuration of detectors in your particular XRA700 unit. HV1 is used for CdTe detectors, HVS2 is used for standard Fast SDDs, and HVS3 is used for 1mm^3 thick Fast SDDs. The limits for each channel are listed in the table below:

HV Supply	Target Detector	Lower Limit	Upper Limit
1	CdTe	0	800
2	Fast SDD	-200	0
3	1mm^3 Fast SDD	-500	0

Note for system stability, the TEC supplies should be turned on before the high voltage supplies.

Errors: An INVALID PARAMETER error will be returned if the parameter is outside of the upper and lower limits of each channel.

Example: HVS1=700; // Set HV Supply 1 to 700V
Example: HVS2=-135; // Set HV Supply 2 to -135V
Example: HVS3=-400; // Set HV Supply 3 to -400V

5.1.11 HVSE – High Voltage Supply Enable

Command: HVSE
Parameter: [ON|OF{F}]
Units: none
Default: 0

Dependencies: none
Supported: All
Related: BTHV, HVS1, HVS2, HVS3

Description: Master enable switch, turns all high voltage supplies on or off if they have a setting written to their control DACs.

Errors: An INVALID PARAMETER error will be returned if the parameter is outside of the upper and lower limits of each channel.

Example: HVSE=ON; // Enable high voltage channels

5.1.12 PAVE – Preamp Voltage Enable

Command: PAVE
Parameter: [ON|OF{F}]
Units: none
Default: 0

Dependencies: none
Supported: All
Related: BTPA

Description: Switch to turn preamplifier power supplies on or off.

Errors: An INVALID PARAMETER error will be returned if the parameter is outside of the upper and lower limits of each channel.

Example: PAVE=ON; // Enable preamplifier power

5.1.13 RESC – Reset Configuration

Command: RESC
Parameter: [YES|NO]
Units: none
Default: 0

Dependencies: none
Supported: All
Related:

Description: RESC is used to reset the XRA700's configuration to default settings. It resets each setting to the default listed for each command and in table XX. If it is used, it should be the first command in the ACII configuration packet.

RESC is useful because the XRA700 stores the settings from previous configuration commands in non-volatile memory

Errors: An INVALID PARAMETER error will be returned if the parameter is outside of the upper and lower limits of each channel.

Example: RESC=Y; // Reset the configuration to default settings

5.1.14 TEC1, TEC2, TEC3, TEC4, TEC5, TEC6, TEC7 - Turn On/Off the Thermoelectric Cooler (TEC) Supply and Set the Temperature for detector on specified channel

Command: TEC#, #= {1, 2, 3, 4, 5, 6, 7}
Parameter: [###|OF{F}]
Units: K
Default: OFF
Range: 0-299K; ~0.1K precision

Dependencies: none
Supported: A
Related: BTEC, TECE, TECS

Description: TEC# turns the PC5 TEC supply on or off, and sets the TEC temperature setpoint for channel #. Note for system stability, the TEC supplies should be turned on before the high voltage supplies.

Errors: An INVALID PARAMETER error will be returned if the specified temperature is outside the range listed. A PC5 NOT PRESENT error will be returned if a PC5 is not detected.

Example:
```
TEC1=220K;      // Turn on the TEC supply for detector 1 and cool to 220K
                // or as cool as possible
```

5.1.15 TECE – Thermoelectric Cooler Supply Enable

Command: TECE
Parameter: [ON|OF{F}]
Units: none
Default: 0

Dependencies: none
Supported: All
Related: BTEC, TEC#, TECS

Description: Master enable switch, turns all TEC supplies on or off if they have a setting written to their control DACs.

Errors: An INVALID PARAMETER error will be returned if the parameter is outside of the upper and lower limits of each channel.

Example: TECE=ON; // Enable TEC supply channels

5.1.16 TECS - Turn On/Off the Thermoelectric Cooler (TEC) Supply and Set the Temperature for all detector channels

Command: TECS
Parameter: [###|OF{F}]
Units: K
Default: OFF
Range: 0-299K; ~0.1K precision

Dependencies: none
Supported: All
Related: BTEC, TECE, TEC#

Description: TECS turns the XRA700 TEC supplies on or off, and sets the TEC temperature setpoint for all channels.

Errors: An INVALID PARAMETER error will be returned if the specified temperature is outside the range listed. A PC5 NOT PRESENT error will be returned if a PC5 is not detected.

Example:
```
TECS=220K;      // Turn on the TEC supply and cool to 220K
                // or as cool as possible
```

6 Select Topics

6.1 Important Configuration Considerations

All Amptek digital products use a flash memory chip to store calibration data, serial number, etc. They also store the most recently used configuration, so that the device can return to the same configuration the next time it is powered on. [See the next section of a discussion of this behavior.]

Specifically, each time the 'Text Configuration' request packet (section 4.1.8) is received, the packet is parsed, the configuration is executed, and the resulting configuration is written to redundant records in the flash memory.

While this is good for most applications, it has the ability to cause serious issues when used in a 24/7 setting, where thousands of acquisitions per month (or in some cases, many more) are possible. Section 6.1.2 below gives more details of the potential issues, and discusses ways to avoid problems.

6.1.1 Differences in Use of Non-Volatile Configuration at Power-up

To Be Determined

6.1.2 Strategy to Avoid Degradation of Flash Memory

The simplest way to avoid degrading the flash memory due to a high number of configuration packets is to use the 'Text configuration (to XRA700) without saving to nonvolatile memory' form of the Text Configuration request packet. This packet type will not save the configuration packet to the non-volatile memory. This has two impacts:

1. This configuration will not be used at power-up. Whatever configuration was last stored will be used.
2. There is typically a ~100mS time during which communications may be stalled while the configuration is written to redundant flash records. This delay does not occur if the configuration is not written to flash.