# QTPRO
# COMMAND LINE INTERFACE

## Revision 0.14

SEPTEMBER 15, 2019

BOB HALLIDAY
MEIR MECHTINGER
IGWE IGWE-KALU

CAMBRIDGE SOUND MANAGEMENT

# Table of Contents

**1 Revision History ..... 5**
**2 Data Communications ..... 7**
2.1 The Serial Command Packet ..... 7
2.1.1 Command Header ..... 7
2.1.2 Opcodes ..... 7
2.1.3 Data ..... 8
2.2 Get Commands Serial Response Packet ..... 8
2.2.1 ACK Data Packet ..... 8
2.2.2 NAK Data Packet ..... 9
**3 Telnet Login ..... 9**
**4 Summary of CS Opcode Commands ..... 11**
**5 Description of Individual CS Commands ..... 12**
5.1 Get MAC Address ..... 12
5.2 Get IP Mode ..... 12
5.3 Set IP Mode ..... 12
5.4 Get IP Address ..... 12
5.5 Set IP Address ..... 13
5.6 Other IP Address Settings ..... 13
5.7 Get Host Name ..... 14
5.8 Set Host Name ..... 14
5.9 Get User Name ..... 14
5.10 Get User Password ..... 14
5.11 Get Unit Name ..... 15
5.12 Set Unit Name ..... 15
5.13 Get Location ..... 15
5.14 Set Location ..... 15
5.15 Get Date ..... 16
5.16 Set Date ..... 16
5.17 Get Day of the Week ..... 16
5.18 Set Day of the Week ..... 17
5.19 Get Time ..... 17
5.20 Set Time ..... 17
5.21 Get Time Zone ..... 17
5.22 Set Time Zone ..... 18
5.23 Get DST Begin Time ..... 18

5.24 Set DST Begin Time ..... 19
5.25 Get DST End Time ..... 19
5.26 Set DST End Time ..... 19
5.27 Get DST Inversion Status ..... 20
5.28 Set DST Inversion Status ..... 20
5.29 Get Selftest Diagnostic Results ..... 21
5.30 Set Selftest Diagnostic Results ..... 21
5.31 Clear Selftest Diagnostic Results ..... 22
5.32 Get Destination for Errors Reporting ..... 23
5.33 Set Destination for Errors Reporting ..... 23
5.34 Get Firmware Version ..... 24
5.35 Get Front Panel Lock ..... 24
5.36 Get Audio A Sensitivity ..... 24
5.37 Set Audio A Sensitivity ..... 25
5.38 Get Audio B Sensitivity ..... 25
5.39 Set Audio B Sensitivity ..... 26
**6 Summary of ZO Opcode Commands ..... 27**
6.1 Get Masking Max ..... 29
6.2 Set Masking Max ..... 29
6.3 Get Masking Min ..... 29
6.4 Set Masking Min ..... 30
6.5 Get Masking Current ..... 30
6.6 Get Mute ..... 30
6.7 Set Mute ..... 31
6.8 Increment Masking Max Volume ..... 31
6.9 Decrement Masking Max Volume ..... 31
6.10 Increment Masking Min Volume ..... 32
6.11 Decrement Masking Min Volume ..... 32
6.12 Get Input A Max Volume ..... 32
6.13 Set Input A Max Volume ..... 33
6.14 Get Input A Min Volume ..... 33
6.15 Set Input A Min Volume ..... 33
6.16 Get Input B Max Volume ..... 34
6.17 Set Input B Max Volume ..... 34
6.18 Get Input B Min Volume ..... 34
6.19 Set Input B Min Volume ..... 35
6.20 Get Input A Current Volume ..... 35

6.21 Get Input B Current Volume ..... 35
6.22 Increment Audio A Max Volume ..... 36
6.23 Decrement Audio A Max Volume ..... 36
6.24 Increment Audio A Min Volume ..... 37
6.25 Decrement Audio A Min Volume ..... 37
6.26 Increment Audio B Max Volume ..... 38
6.27 Decrement Audio B Max Volume ..... 38
6.28 Increment Audio B Min Volume ..... 39
6.29 Decrement Audio B Min Volume ..... 39
6.30 Get Auto Ramping Enable ..... 40
6.31 Set Auto Ramping Enable ..... 40
6.32 Get Auto Ramping Levels ..... 41
6.33 Set Auto Ramping Levels ..... 41
6.34 Get Auto Ramping Dates ..... 42
6.35 Set Auto Ramping Dates ..... 42
6.36 Get Auto Ramp Exclusion ..... 43
6.37 Set Auto Ramp Exclusion ..... 44
6.38 Get Masking Begin Ramp to Max Time ..... 44
6.39 Set Masking Begin Ramp to Max Time ..... 44
6.40 Get Masking Begin Ramp to Min Time ..... 45
6.41 Set Masking Begin Ramp to Min Time ..... 45
6.42 Get Audio A Begin Ramp to Max Time ..... 45
6.43 Set Audio A Begin Ramp to Max Time ..... 46
6.44 Get Audio A Begin Ramp to Min Time ..... 46
6.45 Set Audio A Begin Ramp to Min Time ..... 46
6.46 Get Audio B Begin Ramp to Max Time ..... 47
6.47 Set Audio B Begin Ramp to Max Time ..... 47
6.48 Get Audio B Begin Ramp to Min Time ..... 47
6.49 Set Audio B Begin Ramp to Min Time ..... 48
6.50 Get Masking Ramp Interval ..... 48
6.51 Set Masking Ramp Interval ..... 48
6.52 Get Audio A Ramp Interval ..... 49
6.53 Set Audio A Ramp Interval ..... 49
6.54 Get Audio B Ramp Interval ..... 49
6.55 Set Audio B Ramp Interval ..... 50
6.56 Get Emitter Type ..... 50
6.57 Set Emitter Type ..... 50

6.58 Get Spectrum Type ..... 51
6.59 Set Spectrum Type ..... 51
6.60 Get Emitter Configuration ..... 51
6.61 Set Emitter Configuration ..... 52
6.62 Get Weekend Schedule ..... 52
6.63 Set Weekend Schedule ..... 52
6.64 Get Masking Equalizer ..... 53
6.65 Set Masking Equalizer ..... 53
6.66 Get Input A Equalizer ..... 54
6.67 Set Input A Equalizer ..... 55
6.68 Get Input B Equalizer ..... 55
6.69 Set Input B Equalizer ..... 56
6.70 Get Zone Name ..... 56
6.71 Set Zone Name ..... 56
**7 Summary of AL Read Only Batch Commands ..... 58**
7.1 Read All System Variables ..... 58
7.2 Read All Zone Variables ..... 60
7.3 Read All Masking Equalizers ..... 62
7.4 Read All Input A and B Equalizers ..... 63
**8 Special ZY Command Header ..... 65**

# 1 Revision History

| Date | Revision | Author | Change Description |
|---|---|---|---|
| 09/05/14 | 0.01 | bh | First cut |
| 09/10/14 | 0.02 | bh | Additional Read Only commands added |
| 10/22/14 | 0.03 | bh | Most Commands have been reconstituted as 4-byte (acronymic) commands |
| 11/12/14 | 0.04 | bh | Added Set capabilities for Gateway, Net Mask, Primary DNS and Secondary DNS |
| 02/26/15 | 0.05 | bh | Added Day of Week Command. Expanded the “AL” Command descriptions. |
| 03/03/15 | 0.06 | bh | INCM and DECM Commands added. |
| 04/27/15 | 0.07 | bh | Username and Password Commands are now Read-Only |
| 05/05/17 | 0.10 | MM | Updated document, adding Active Emitter commands |
| 10/17/17 | 0.11 | MM | Added Sensitivity commands for Audio A and Audio B, added Masking, Audio A and Audio B Min/Max commands and modified Auto Ramping functionality. |

| | | | |
|---|---|---|---|
| 1/29/18 | 0.12 | MM | Various updates: |
| 1/16/19 | 0.13 | IIK | Daylight Savings Inversion Update. GUI Updates |

# 2 Data Communications

The QtPro has a Command Line Interface (CLI) which enables the ability of remote devices to issue a variety of SET and GET Commands to the QtPro.

There are two distinct communications channels over which CLI Commands can be transmitted. These include:

1. Telnet via TCP/IP
2. USB or RS-232 (not yet available in the hardware)

These communications channels can operate either separately or simultaneously in the exchange of data with the QTPRO.

## *2.1 The Serial Command Packet*

The standard SET Command data packet transmitted by the PC Server to the QtPro will be defined as:

| **Bytes** | 2 | 3 | 1 | 4 | 1 | 0 – n | 1 |
|---|---|---|---|---|---|---|---|
| **Field** | Command Header | “SET” | ‘,’ | Opcode | ‘=’ | Data | <CR> |

The standard GET Command data packet transmitted by the PC Server to the QtPro will be defined as:

| **Bytes** | 2 | 3 | 1 | 4 | 1 |
|---|---|---|---|---|---|
| **Field** | Command Header | “GET” | ‘,’ | Opcode | <CR> |

All data is transmitted in standard ASCII format. “<CR>” = 0x0D.

### 2.1.1 Command Header

The Command Header is composed of **one** of the following three possible byte-pairs:

| Command | Value | Set | Get |
|---|---|---|---|
| Single Global Parameters | “CS” | y | y |
| Single Zone Parameters | “ZO” | y | y |
| Read-Only Batch Commands | “AL” | n | y |

In addition, a special function Command Header, “ZY”, can be used to trigger a Software Reset of the Device.

### 2.1.2 Opcodes

All Opcodes must be 4-bytes long and uppercase.

The CS Opcodes are defined in Section 4 below.

The ZO Opcodes are defined in Section 5 below.

The AL Opcodes are defined in Section 6 below.

### 2.1.3 Data

Some Command Packets require no data. In this case, any data transmitted by the pc will be ignored by the QTPRO.

## *2.2 Get Commands Serial Response Packet*

The Response data packet transmitted by the QtPro to the PC application will be defined as:

| **Bytes** | 1 | 2 | 1 | 1 | (0 – n) | 1 | 1 |
|---|---|---|---|---|---|---|---|
| **Field** | ‘{‘ | Command Header | Opcode | ‘=’ | Data | ‘}’ | <CR> |

For Example:

The server issues the Command to Get the Device IP Address:

**“CSGET,IPAD<CR>”**

Then the QtPro Serial Response Packet shall be:

**“{CSGET,IPAD=192.168.1.10}<CR>”**

Where 192.168.1.10 is the Device IP Address.

The Response Packet is composed of variable length Ascii Data. The content of the response is a function of the Opcode received.

### 2.2.1 ACK Data Packet

When data is transmitted in a SET Command, the normal response to a successfully received packet is “{ACK,echoed opcode and data}”.

| Byte | Value |
|---|---|
| 0 | ‘{‘ |
| 1 | ‘A’ |
| 2 | ‘C’ |
| 3 | ‘K’ |
| 4 | ‘,’ |
| 5- n | Echoed opcode and data, as in “IPAD=192.168.1.10” |
| n + 1 | ‘}’ |
| n + 2 | <CR> |

When a GET Command is issued from the Server, the returned data serves as an implied ACK, transmitted in place of the ACK.

### 2.2.2 NAK Data Packet

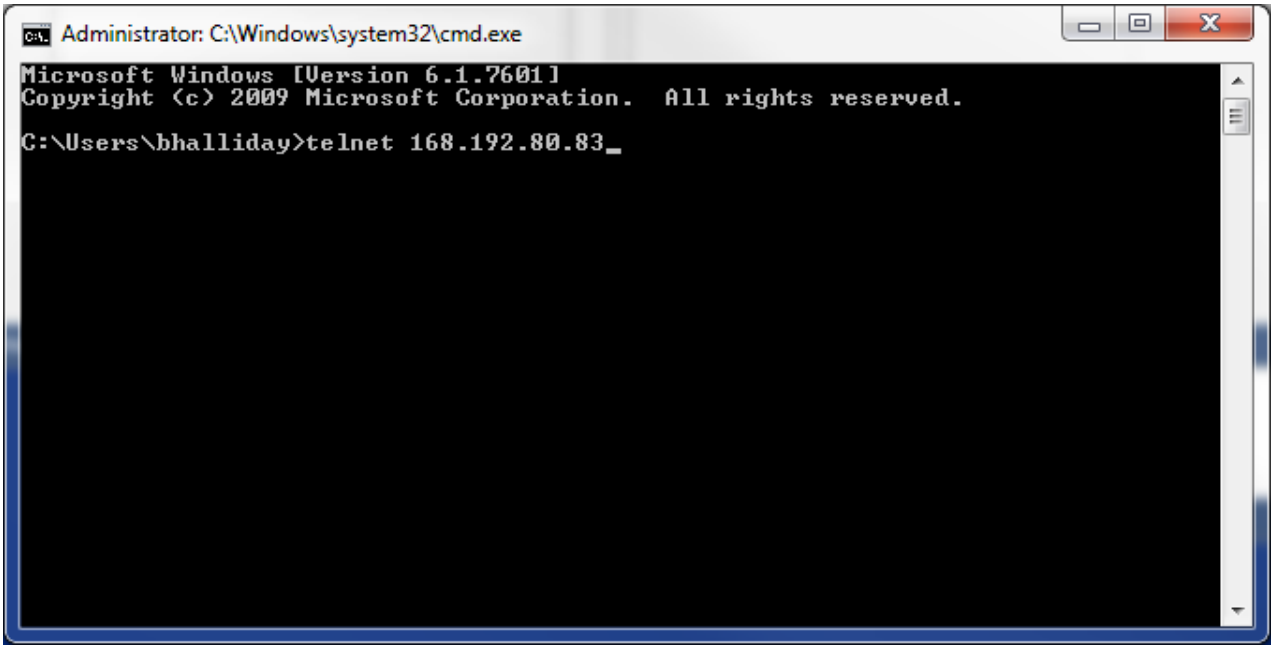
In the event that the PC application attempts to write illegal data to the QTPRO, the QtPro will respond with a NAK Data Packet:

| Byte | Value |
|---|---|
| 0 | ‘{‘ |
| 1 | ‘N’ |
| 2 | ‘A’ |
| 3 | ‘K’ |
| 4 | ‘}’ |
| 5 | <CR> |

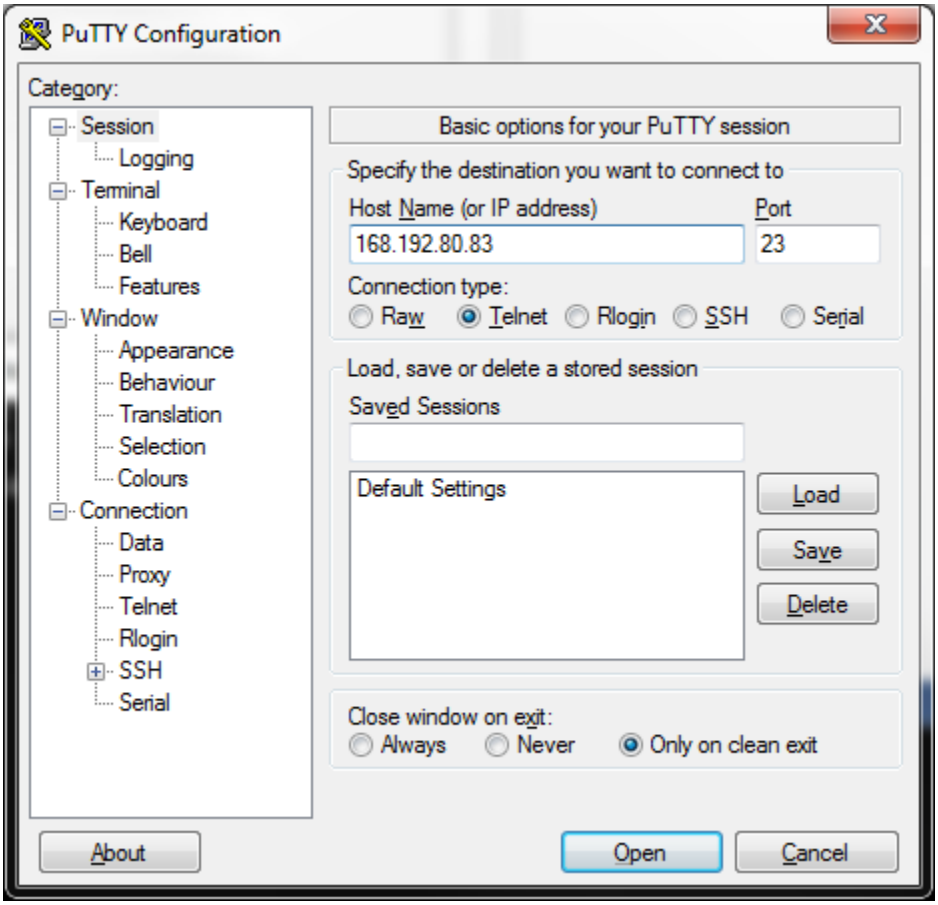
An unrecognized SET or GET Command will also trigger a NAK response.

# 3 Telnet Login

From a Command Line Window, the user should type “Telnet ip address”, as in:




If Telnet functionality is not native on a user’s machine, the user can download a free copy of Putty and initiate the Telnet session from within the Putty Window:




When the Telnet session has been successfully established, the Qt device will transmit the following opening greeting:

"CSM Telnet Window, v1.0"

Now the user can begin transmitting SET and GET commands as defined below.

To terminate the session, the user must transmit "BYE<CR>".

# 4 Summary of CS Opcode Commands

The following table summarizes the CS Command Packets transmitted by a PC application to the Microcontroller in the QTPRO:

| CS Function | Opcode | GET | SET |
|---|---|---|---|
| MAC Address | “MACA” | x | |
| IP Mode | “IPMO” | x | x |
| IP Address | “IPAD’’ | x | x |
| Gateway | “GATE” | x | x |
| Net Mask | “MASK” | x | x |
| Primary DNS | “PDNS” | x | x |
| Secondary DNS | “SDNS” | x | x |
| Host Name | “HNAM” | x | x |
| User Name | “UNAM” | x | |
| User Password | “UPWD” | x | |
| Unit Name | “UNIT” | x | x |
| Location | “LOCA” | x | x |
| Date | “DATE” | x | x |
| Day of the Week | “DAYW” | x | x |
| Time | “TIME” | x | x |
| Time Zone | “TZON” | x | x |
| DST Begin Time | “DSTB” | x | x |
| DST End Time | “DSTE” | x | x |
| DST Inversion Status | “IDST | X | X |
| Self-Test Diagnostic Results | “DIAG” | x | x |
| Destination for Errors Reporting | “EADD” | x | x |
| Clear Errors | “CLER” | | x |
| Firmware Version | “FIRM” | x | |
| Front Panel Lock | “LOCK” | x | |
| Audio A Sensitivity | “SENA” | x | X |
| Audio B Sensitivity | “SENB” | x | X |

For example,

1) A “Set Host Name” command would be represented by the Command Packet: “CSSET,HNAM=QT-600<CR>”

2) A “Get Host Name” command would be represented by the Command Packet: “CSGET,HNAM<CR>”

# 5 Description of Individual CS Commands

## 5.1 Get MAC Address

| | |
|---|---|
| **Opcode** | "MACA" |
| **Description** | The PC application gets the MAC Address from the XT-X00. |
| **Command Packet** | "CSGET,MACA<CR>" |
| **Data in Response Message** | 12 bytes of MAC address.<br>For example:<br>"{CSGET,MACA=123456789ABC}<CR>" |

The MAC Address is a fixed, immutable parameter, previously initialized inside the microcontroller. Therefore, no corresponding "SET MAC Address" command can be defined.

## 5.2 Get IP Mode

| | |
|---|---|
| **Opcode** | "IPMO" |
| **Description** | The PC application reads the IP Mode from the QTPRO. |
| **Command Packet** | "CSGET,IPMO<CR>" |
| **Data in Response Message** | A single Ascii Byte, where '1' = Dynamic and '0' = Static<br>For example:<br>"{CSGET,IPMO=1}<CR>" signifies that the IP Address is dynamic. |

## 5.3 Set IP Mode

| | |
|---|---|
| **Opcode** | "IPMO" |
| **Description** | The PC application writes the IP Mode to the QTPRO. |
| **Command Packet** | A single Ascii Byte, where '1' = Dynamic and '0' = Static. Note that any change in the IP Mode does not go into effect until the Device is reset.<br>For example:<br>"CSSET,IPMO=1<CR>" signifies that the IP Address is dynamic. |
| **Standard Response** | "{ACK,IPMO=1}<CR>" |

A change in the IP Mode does not go into effect immediately in the receiving device. The device must be reset, either manually or by software via the "ZY" Command. Refer to Section 8.

## 5.4 Get IP Address

| | |
|---|---|
| **Opcode** | "IPAD'' |

| | |
|---|---|
| **Description** | The PC application reads the IP Address from the QTPRO. |
| **Command Packet** | “CSGET,IPAD<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s IP address.<br>For example:<br>“{CSGET,IPAD=192.168.14.67}<CR>” |

## 5.5 Set IP Address

| | |
|---|---|
| **Opcode** | “IPAD’’ |
| **Description** | The PC application writes the IP Address to the QTPRO. |
| **Command Packet** | A variable Ascii string representing the device’s IP address.<br>This Opcode only has validity when the unit is operating in Static Mode. Furthermore, any new IP Address setting will not take effect until the device has been reset.<br>For example:<br>“CSSET,IPAD=192.168.14.67<CR>” |
| **Standard Response** | “{ACK,IPAD=192.168.14.67}<CR>” |

A change in the IP Address does not go into effect immediately in the receiving device. The device must be reset, either manually or by software via the “ZY” Command. Refer to Section 8.

Furthermore, this Opcode will have no validity if the device continues to operate in Dynamic Mode.

## 5.6 Other IP Address Settings

The following IP Address commands follow the format and behavior of the Set IP Address, except that no reset of the device is necessary for the new settings to stick.

| CS Function | Opcode | GET | SET |
|---|---|---|---|
| Gateway | “GATE” | x | x |
| Net Mask | “MASK” | x | x |
| Primary DNS | “PDNS” | x | x |
| Secondary DNS | “SDNS” | x | x |

## 5.7 Get Host Name

| | |
|---|---|
| **Opcode** | “HNAM” |
| **Description** | The PC application reads the Host Name from the QTPRO. |
| **Command Packet** | “CSGET,HNAM<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s Host Name.<br>For example:<br>“{CSGET,HNAM=QT-39}<CR>” |

## 5.8 Set Host Name

| | |
|---|---|
| **Opcode** | “HNAM” |
| **Description** | The PC application writes the Host Name to the QTPRO. |
| **Data** | A variable Ascii string representing the device’s Host Name.<br>For example:<br>“CSSET,HNAM=QT-39<CR>” sets the Host Name to “QT-39” |
| **Standard Response** | “{ACK,HNAM=QT-39}<CR>” |

## 5.9 Get User Name

| | |
|---|---|
| **Opcode** | “UNAM” |
| **Description** | The PC application reads the User Name from the QTPRO. |
| **Command Packet** | “CSGET,UNAM<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s User Name.<br>For example:<br>“{CSGET,UNAM=Terry}<CR>” |

## 5.10 Get User Password

| | |
|---|---|
| **Opcode** | “UPWD” |
| **Description** | The PC application reads the User Password from the QTPRO. |
| **Command Packet** | “CSGET,UPWD<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s User Password.<br>For example:<br>“{CSGET.UPWD=bob134}<CR>” |

## 5.11 Get Unit Name

| | |
|---|---|
| **Opcode** | "UNIT" |
| **Description** | The PC application reads the Unit Name from the QTPRO. |
| **Command Packet** | "CSGET,UNIT<CR>" |
| **Data in Response Message** | A variable Ascii string representing the device's Unit Name.<br>For example:<br>"{CSGET,UNIT=Acme-1}<CR>" |

## 5.12 Set Unit Name

| | |
|---|---|
| **Opcode** | "UNIT" |
| **Description** | The PC application writes the Unit Name to the QTPRO. |
| **Data** | A variable Ascii string representing the device's Unit Name.<br>For example:<br>"CSSET,UNIT=Acme-1<CR>" sets the Unit Name to "Acme-1" |
| **Standard Response** | "{ACK,UNIT=Acme-1}<CR>" |

## 5.13 Get Location

| | |
|---|---|
| **Opcode** | "LOCA" |
| **Description** | The PC application reads the Location from the QTPRO. |
| **Command Packet** | "CSGET,LOCA<CR>" |
| **Data in Response Message** | A variable Ascii string representing the device's Location.<br>For example:<br>"{CSGET,LOCA=Waco, TX}<CR>" |

## 5.14 Set Location

| | |
|---|---|
| **Opcode** | "LOCA" |
| **Description** | The PC application writes the Location to the QTPRO. |
| **Data** | A variable Ascii string representing the device's Location.<br>For example:<br>"CSSET,LOCA=1st Floor<CR>" sets the Location to "1st Floor" |
| **Standard Response** | "{ACK,LOCA=1st Floor}<CR>" |

## 5.15 Get Date

| | |
|---|---|
| **Opcode** | "DATE" |
| **Description** | The PC application reads the Date from the QTPRO. |
| **Command Packet** | "CSGET,DATE<CR>" |
| **Data in Response Message** | An 8-byte Ascii string (mmddyyyy) representing the device's Date.<br>For example:<br>"{CSGET,DATE=08102015}<CR>" |

## 5.16 Set Date

| | |
|---|---|
| **Opcode** | "DATE" |
| **Description** | The PC application writes the Date to the QTPRO. |
| **Data** | An 8-byte Ascii string (mmddyyyy) representing the device's Date.<br>For example:<br>"CSSET,DATE=08102015<CR>" sets the Date to August 10, 2015 |
| **Standard Response** | "{ACK,DATE=081015}<CR>" |

## 5.17 Get Day of the Week

| | |
|---|---|
| **Opcode** | "DAYW" |
| **Description** | The PC application reads the Day of the Week from the QTPRO. |
| **Command Packet** | "CSGET,DAYW<CR>" |
| **Data in Response Message** | A single 16sci byte, representing the day of the week, where:<br>'0' = Sunday<br>'1' = Monday<br>'2' = Tuesday<br>'3' = Wednesday<br>'4' = Thursday<br>'5' = Friday<br>'6' = Saturday<br>For example:<br>"{CSGET,DAYW=1}<CR>" gets the Day of the Week = Monday |

## 5.18 Set Day of the Week

| | |
|---|---|
| **Opcode** | "DAYW" |
| **Description** | The PC application writes the Day of the Week to the QTPRO. |
| **Data** | A single 17sci byte in the range of '0' – '6', representing the day of the week.<br>For example:<br>"CSSET,DAYW=1<CR>" sets the Day of the Week to Monday |
| **Standard Response** | "{ACK,DAYW=1}<CR>" |

## 5.19 Get Time

| | |
|---|---|
| **Opcode** | "TIME" |
| **Description** | The PC application reads the Time from the QtPro in 24-hour clock format. |
| **Command Packet** | "CSGET,TIME<CR>" |
| **Data in Response Message** | A 6-byte Ascii string (hhmmss) representing the device's Time.<br>For example:<br>"{CSGET,TIME=130000}<CR>" |

## 5.20 Set Time

| | |
|---|---|
| **Opcode** | "TIME" |
| **Description** | The PC application writes the Time to the QtPro in 24-hour clock format. |
| **Data** | A 6-byte Ascii string (hhmmss) representing the device's Time.<br>For example:<br>"CSSET,TIME=130000<CR>" sets the Time to 1 PM. |
| **Standard Response** | "{ACK,TIME= 130000}<CR>" |

## 5.21 Get Time Zone

| | |
|---|---|
| **Opcode** | "TZON" |
| **Description** | The PC application reads the Time Zone from the QTPRO. |
| **Command Packet** | "CSGET,TZON<CR>" |
| **Data in Response Message** | A variable Ascii string representing the device's Time Zone. The valid range is -12 to +12. See the Time Zone Table below. |

| | |
|---|---|
| | For example:<br>“{CSGET,TZON=-8}<CR>” |

## 5.22 Set Time Zone

| | |
|---|---|
| **Opcode** | “TZON” |
| **Description** | The PC application writes the Time Zone to the QTPRO. |
| **Data** | A variable Ascii string representing the device’s Time Zone. The valid range is -12 to +12. See the Time Zone Table below.<br>For example:<br>“CSSET,TZON=-8<CR>” sets the Time Zone to Pacific Time |
| **Standard Response** | “{ACK,TZON=-8}<CR>” |

**TIME ZONE SETTINGS**

| Offset from GMT | Time Zone |
|---|---|
| -12 | |
| -11 | |
| -10 | |
| -9 | |
| -8 | Pacific Time |
| -7 | Mountain Time |
| -6 | Central Time |
| -5 | Eastern Time |
| -4 | Atlantic Time |
| -3 | |
| -2 | |
| -1 | |
| 0 | Western European Time |
| +1 | Central European Time |
| +2 | Eastern European Time |
| +3 | |
| +4 | |
| +5 | |
| +6 | |
| +7 | |
| +8 | |
| +9 | |
| +10 | |
| +11 | |
| +12 | |

## 5.23 Get DST Begin Time

| | |
|---|---|
| **Opcode** | “DSTB” |
| **Description** | The PC application reads the Daylight Savings Time Begin Time from the QTPRO. |
| **Command Packet** | “CSGET,DSTB<CR>” |
| **Data in Response Message** | A Single Ascii Byte representing the device’s DST Begin Time. See the DST Begin Table below. |

| | |
|---|---|
| | For example:<br>"{CSGET,DSTB=3<CR>" |

## 5.24 Set DST Begin Time

| | |
|---|---|
| **Opcode** | "DSTB" |
| **Description** | The PC application writes the Daylight Savings Time Begin Time to the QTPRO. |
| **Data** | A Single Ascii Byte representing the device's DST Begin Time. See the DST Begin Table below.<br>For example:<br>"CSSET.DSTB=3<CR>" sets the DST Begin Time to the third Sunday in March |
| **Standard Response** | "{ACK,DSTB=3}<CR>" |

**DAYLIGHT SAVINGS TIME BEGIN TIME**

| Hex Value | Ascii Value | DST Begin Time |
|---|---|---|
| 0x30 | '0' | Not used in this region |
| 0x31 | '1' | First Sunday in March |
| 0x32 | '2' | Second Sunday in March |
| 0x33 | '3' | Third Sunday in March |
| 0x34 | '4' | Fourth Sunday in March |
| 0x35 | '5' | First Sunday in April |

## 5.25 Get DST End Time

| | |
|---|---|
| **Opcode** | "DSTE" |
| **Description** | The PC application reads the Daylight Savings Time End Time from the QTPRO. |
| **Command Packet** | "CSGET,DSTE<CR>" |
| **Data in Response Message** | A Single Ascii Byte representing the device's DST End Time. See the DST End Table below.<br>For example:<br>"{CSGET,DSTE=4<CR>" |

## 5.26 Set DST End Time

| | |
|---|---|
| **Opcode** | "DSTE" |
| **Description** | The PC application writes the Daylight Savings Time End Time to the QTPRO. |

| | |
|---|---|
| **Data** | A Single Ascii Byte representing the device's DST End Time. See the DST End Table below.<br>For example:<br>"CSSET,DSTE=4<CR>" sets the DST End Time to the fourth Sunday in October. |
| **Standard Response** | "{ACK,DSTE=4}<CR>" |

**DAYLIGHT SAVINGS TIME END TIME**

| Hex Value | Ascii Value | DST Begin Time |
|---|---|---|
| 0x30 | '0' | Not used in this region |
| 0x31 | '1' | First Sunday in October |
| 0x32 | '2' | Second Sunday in October |
| 0x33 | '3' | Third Sunday in October |
| 0x34 | '4' | Fourth Sunday in October |
| 0x35 | '5' | First Sunday in November |

## 5.27 Get DST Inversion Status

| | |
|---|---|
| **Opcode** | "IDST" |
| **Description** | The PC application reads whether Daylight Savings Time has been inverted for the Southern Hemisphere, from the QTPRO. |
| **Command Packet** | "CSGET,IDST<CR>" |
| **Data in Response Message** | A Single Ascii Byte representing the device's DST Inversion Status. See the DST Inversion Tables below.<br>For example:<br>"{CSGET,IDST=1<CR>" |

## 5.28 Set DST Inversion Status

| | |
|---|---|
| **Opcode** | "IDST" |
| **Description** | The PC application writes whether Daylight Savings Time should be inverted for the Southern Hemisphere, to the QTPRO. |
| **Data** | A Single Ascii Byte representing the device's DST Inversion Status. See the DST Inversion Tables below.<br>***Note: Can only be set to "0" or "2"**<br>For example:<br>"CSSET.IDST=2<CR>" sets the DST Inversion Status to "Locked". Northern Hemisphere DST options in a primarily Southern Hemisphere Time Zone. |
| **Standard Response** | "{ACK,IDST=2}<CR>" |

**DAYLIGHT SAVINGS TIME INVERSION STATUS**

| Ascii Value | DST INVERSION STATUS |
|---|---|
| ‘0’ | Northern Hemisphere DST options |
| ‘1’ | Southern Hemisphere DST options |
| ‘2’ | “Locked”: Northern Hemisphere DST options in a primarily Southern Hemisphere Time Zone. |

**INVERTED DAYLIGHT SAVINGS TIME BEGIN TIME**

| Hex Value | Ascii Value | DST Begin Time |
|---|---|---|
| 0x30 | ‘0’ | Not used in this region |
| 0x31 | ‘1’ | Second Sunday in August |
| 0x32 | ‘2’ | Last Sunday in September |
| 0x33 | ‘3’ | First Sunday in October |
| 0x34 | ‘4’ | First Sunday in November (12am) |
| 0x35 | ‘5’ | First Sunday in November |

**INVERTED DAYLIGHT SAVINGS TIME END TIME**

| Hex Value | Ascii Value | DST End Time |
|---|---|---|
| 0x30 | ‘0’ | Not used in this region |
| 0x31 | ‘1’ | Third Sunday in January |
| 0x32 | ‘2’ | Third Sunday in February (12am) |
| 0x33 | ‘3’ | Fourth Sunday in March |
| 0x34 | ‘4’ | First Sunday in April |
| 0x35 | ‘5’ | Second Sunday in May |

## 5.29 Get Selftest Diagnostic Results

| | |
|---|---|
| **Opcode** | “DIAG” |
| **Description** | The PC application reads the Selftest Diagnostic Results from the QTPRO. |
| **Command Packet** | “CSGET,DIAG<CR>” |
| **Data in Response Message** | A 7-byte Ascii string representing the device’s Selftest Diagnostic Results. See the table below for a description of the error codes.<br>For example:<br>“{CSGET,DIAG=8412011}<CR>” |

## 5.30 Set Selftest Diagnostic Results

| | |
|---|---|
| **Opcode** | “DIAG” |
| **Description** | The PC application sets the Selftest Diagnostic Results for the QTPRO. |
| **Command Packet** | “CSSET,DIAG<CR>” |

| | |
|---|---|
| **Data in Response Message** | A 7-byte Ascii string representing the device's Selftest Diagnostic Results. Opcode should be used for testing only. See the table below for a description of the error codes.<br>For example:<br>"{CSSET,DIAG=8412011}<CR>" |

## 5.31 Clear Selftest Diagnostic Results

| | |
|---|---|
| **Opcode** | "CLER" |
| **Description** | The PC application sets the Selftest Diagnostic Results in the QTPRO. |
| **Data** | There is no data associated with this Opcode. All error codes are cleared to zeroes when this Opcode is received. Note that this Opcode only clears the error status codes. It cannot clear any persistent errors themselves. Therefore, if an error persists, then an active error status will be reported back to the server, even after a Clear Opcode has been executed.<br>For example:<br>"CSSET,CLER=1<CR>" clears all error status codes to zeros. |
| **Standard Response** | "{ACK,CLER=1}<CR>" |

**SELFTEST DIAGNOSTIC RESULTS**

This 11-byte array is composed of seven bytes of emitter errors, three bytes of I2c errors, and one byte of CPU errors.

| Byte | Error | Results Range |
|---|---|---|
| 0 | Control Module Error - lower 4 bits<br>bit 0 :<br>bit 1: System Thermal Fault<br>bit 2:<br>bit 3: System Hardware Fault | The emitter error bytes fall in the range of Ascii '0' to '?' (0x30 – 0x3F), where '0' = no error. |
| 1 | Zone 6 Emitter Errors - lower 4 bits<br>bit 0 : Amp Short Fault<br>bit 1: Amp Thermal Fault<br>bit 2: Emitter Network Fault<br>bit 3: Hardware Fault | |
| 2 | Zone 5 Emitter Errors - lower 4 bits | |
| 3 | Zone 4 Emitter Errors - lower 4 bits | |
| 4 | Zone 3 Emitter Errors - lower 4 bits | |
| 5 | Zone 2 Emitter Errors - lower 4 bits | |
| 6 | Zone 1 Emitter Errors - lower 4 bits | |
| 7 | Bit 0 – I2c Error – Zone 1 DSP<br>Bit 1 – I2c Error – Zone 2 DSP<br>Bit 2 – I2c Error – Zone 3 DSP<br>Bit 3 - | The I2c error bytes fall into the range of Ascii '0' to '?' (0x30 – 0x3F), where '0' = no error. |
| 8 | Bit 0 – I2c Error – Zone 4 DSP<br>Bit 1 – I2c Error – Zone 5 DSP<br>Bit 2 – I2c Error – Zone 6 DSP<br>Bit 3 - | |
| 9 | Bit 0 – I2c Error – Temperature Sensor<br>Bit 1 – I2c Error – Eeprom<br>Bit 2 – I2c Error – Real Time Clock (RTC) | |

| | |
|---|---|
| 10 | Bit 3 -<br>Bit 0 – RAM Error<br>Bit 1 –<br>Bit 2 –<br>Bit 3 - |

## 5.32 Get Destination for Errors Reporting

| | |
|---|---|
| **Opcode** | "EADD" |
| **Description** | The PC application reads the Destination for Errors Reporting from the QTPRO. |
| **Command Packet** | "CSGET, EADD <CR>" |
| **Data in Response Message** | A variable Ascii string representing the device's Destination for Errors Reporting. Up to five comma separated 23sci string subfields may be included in the response message. Each subfield is assigned a prefix ID value.<br><br>**Subfield ID** / **Definition**<br>A – Email Address<br>B – Server Name<br>C – Port Number<br>D – User Name<br>E – Password<br><br>For example:<br>"{CSGET, EADD=Asmtp.mail.com,C25}<CR>" |

## 5.33 Set Destination for Errors Reporting

| | |
|---|---|
| **Opcode** | "EADD" |
| **Description** | The PC application writes the Destination for Errors Reporting to the QTPRO. |
| **Data** | A variable Ascii string representing the device's Destination for Errors Reporting. Up to five comma separated 23sci string fields may be included in the Set Command. See Table above<br><br>For example:<br><br>"CSSET, EADD=Asmtp.mail.com,C25" sets the Email Address = "smtp.mail.com" and the Port Number = 0x25.<br><br>If the software needs to clear all subfields back to NULLs, then the command would look like this:<br><br>"CSSET, EADD =A,B,C,D,E<CR>" |

| | |
|---|---|
| **Standard Response** | "{ACK, EADD=Asmtp.mail.com,C25}<CR>" |

## 5.34 Get Firmware Version

| | |
|---|---|
| **Opcode** | "FIRM" |
| **Description** | The PC application reads the Firmware Version from the QTPRO. |
| **Command Packet** | "CSGET,FIRM<CR>" |
| **Data in Response Message** | A variable-lengtrh Ascii string representing the device's Firmware Version.<br>For example:<br>"{CSGET,FIRM=6.2.0}<CR>" returns firmware version 6.2.0 |

## 5.35 Get Front Panel Lock

| | |
|---|---|
| **Opcode** | "LOCK" |
| **Description** | The PC application reads the state of the Front Panel Lock from the QTPRO. |
| **Command Packet** | "CSGET,LOCK<CR>" |
| **Data in Response Message** | A single Ascii Byte, where '0' = Locked and '1' = Unlocked<br>For example:<br>"{CSGET,LOCK=1}<CR>" signifies that the Front Panel is unlocked |

## 5.36 Get Audio A Sensitivity

| | |
|---|---|
| **Opcode** | "SENA" |
| **Description** | The PC application reads the state of the Audio A Sensitivity from the QTPRO. |
| **Command Packet** | "CSGET,SENA<CR>" |
| **Data in Response Message** | A single Ascii Byte, where<br>'0' = Normal Signal, no gain<br>'1' = High Signal, negative gain<br>'2' = Low Signal, positive gain<br>For example:<br>"{CSGET,SENA=1}<CR>" signifies that the Audio A sensitivity is set to High Signal |

## 5.37 Set Audio A Sensitivity

| | |
|---|---|
| **Opcode** | “SENA” |
| **Description** | The PC application writes the state of the Audio A Sensitivity to the QTPRO. |
| **Command Packet** | “CSSET,SENA=0<CR>” |
| **Data in Response Message** | A single Ascii Byte, where<br>‘0’ = Normal Signal, no gain<br>‘1’ = High Signal, negative gain<br>‘2’ = Low Signal, positive gain<br>For example:<br>“{CSSET,SENA=0}<CR>” sets the Audio A sensitivity to Normal Signal |

## 5.38 Get Audio B Sensitivity

| | |
|---|---|
| **Opcode** | “SENB” |
| **Description** | The PC application reads the state of the Audio B Sensitivity from the QTPRO. |
| **Command Packet** | “CSGET,SENB<CR>” |
| **Data in Response Message** | A single Ascii Byte, where<br>‘0’ = Normal Signal, no gain<br>‘1’ = High Signal, negative gain<br>‘2’ = Low Signal, positive gain<br>For example:<br>“{CSGET,SENB=1}<CR>” signifies that the Audio B sensitivity is set to High Signal |

## 5.39 Set Audio B Sensitivity

| | |
|---|---|
| **Opcode** | “SENB” |
| **Description** | The PC application writes the state of the Audio B Sensitivity to the QTPRO. |
| **Command Packet** | “CSSET,SENB=0<CR>” |
| **Data in Response Message** | A single Ascii Byte, where<br>‘0’ = Normal Signal, no gain<br>‘1’ = High Signal, negative gain<br>‘2’ = Low Signal, positive gain<br>For example:<br>“{CSSET,SENA=0}<CR>” sets the Audio A sensitivity to Normal Signal |

# 6 Summary of ZO Opcode Commands

ZO Commands have two subfields associated with them:

1. The Opcode value itself
2. Zone Number – 0-5 for the QT-600 or 0-2 for the QT-300

The Zone Number always follows the Opcode value in the Command Sequence Order.

The following table summarizes the ZO Command Packets transmitted by a PC application to the Microcontroller in the QTPRO:

| ZO Function | Opcode | SET | GET |
|---|---|---|---|
| Masking Max | “MMAX” | x | x |
| Masking Min | “MMIN” | x | x |
| Masking Current | “MCUR” | | x |
| Mute | “MUTE” | x | x |
| Increment Masking Max | “INCM” | x | |
| Decrement Masking Max | “DECM” | x | |
| Increment Masking Min | “INCI” | x | |
| Decrement Masking Min | “DECI” | x | |
| Input A Max | “INAM” | x | x |
| Input B Max | “INBM” | x | x |
| Input A Min | “INAI” | x | x |
| Input B Min | “INBI” | x | x |
| Input A Current | “INAC” | | x |
| Input B Current | “INBC” | | x |
| Increment Audio A Max | “ICAM” | x | |
| Increment Audio B Max | “ICBM” | x | |
| Increment Audio A Min | “ICAI” | x | |
| Increment Audio B Min | “ICBI” | x | |
| Decrement Audio A Max | “DCAM” | x | |
| Decrement Audio B Max | “DCBM” | x | |
| Decrement Audio A Min | “DCAI” | x | |
| Decrement Audio B Min | “DCBI” | x | |
| Auto Ramping Enable | “ENAR” | x | x |
| Auto Ramping Levels | “ARLV” | x | x |
| Auto Ramping Dates | “ARDT” | x | x |
| Auto Ramping Exclusion | “AREX” | x | x |
| Begin Masking Ramp to Max Time | “RMAX” | x | x |
| Begin Masking Ramp to Min Time | “RMIN” | x | x |

| | | | |
|---|---|---|---|
| Begin Audio A Ramp to Max Time | “RAMX” | x | x |
| Begin Audio A Ramp to Min Time | “RAMN” | x | x |
| Begin Audio B Ramp to Max Time | “RBMX” | x | x |
| Begin Audio B Ramp to Min Time | “RBMN” | x | x |
| Masking Ramp Interval | “RINT” | x | x |
| Audio A Ramp Interval | “RAIT” | x | x |
| Audio B Ramp Interval | “RBIT” | x | x |
| Emitter Type | “EMIT” | x | x |
| Spectrum Type | “SPEC” | x | x |
| Automatic Emitter Configuration | “EMCF” | x | x |
| Weekend Schedule | “WEEK” | x | x |
| Masking Equalizers | “MEQU” | x | x |
| Input A Equalizers | “AEQU” | x | x |
| Input B Equalizers | “BEQU” | x | x |
| Zone Name | “ZNAM” | x | x |

For example,

1) A “Set Masking Max” command of Volume Level 10 to Zone 3 would be represented by the Command Packet: “ZOSET,MMAX2=20<CR>”

## 6.1 Get Masking Max

| | |
|---|---|
| **Opcode** | “MMAX” |
| **Description** | The PC application reads the Zone 6 maximum sound volume level from the QTPRO. Each unit of MMAX represents ½ dB. |
| **Command Packet** | “ZOGET,MMAX5<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s Zone 6 maximum sound volume level.<br>For example:<br>“{ZOGET,MMAX5=11}<CR>” means that volume = 5.5dB |

## 6.2 Set Masking Max

| | |
|---|---|
| **Opcode** | “MMAX” |
| **Description** | The PC application writes the Zone 6 maximum sound volume level to the QTPRO. Each unit of MMAX represents ½ dB. |
| **Data** | A variable Ascii string representing the device’s maximum sound volume level. The valid volume level range is ‘0’ to ‘60’ where is unit is equal to ½ dB.<br>For example:<br>“ZOSET,MMAX5=11<CR>” sets Zone 6 max sound volume level to 5.5dB. |
| **Standard Response** | “{ACK, MMAX5=11}<CR>” |

## 6.3 Get Masking Min

| | |
|---|---|
| **Opcode** | “MMIN” |
| **Description** | The PC application reads the Zone 6 minimum sound volume level from the QTPRO. Each unit of MMIN represents ½ dB. |
| **Command Packet** | “ZOGET. MMIN5<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s minimum sound volume level.<br>For example:<br>“{ZOGET, MMIN5=11}<CR>”<br>Gets a volume value level of 5.5 dB. |

## 6.4 Set Masking Min

| | |
|---|---|
| **Opcode** | "MMIN" |
| **Description** | The PC application writes the Zone 6 minimum sound volume level to the QTPRO. |
| **Data** | A variable Ascii string representing the device's minimum sound volume level. The valid volume level range is '0' to '60' where each unit = ½ dB.<br>For example:<br>"ZOSET, MMIN5=11<CR>" sets Zone 6 min sound volume level to 5.5dB. |
| **Standard Response** | "{ACK, MMIN5=11}<CR>" |

## 6.5 Get Masking Current

| | |
|---|---|
| **Opcode** | "MCUR" |
| **Description** | The PC application reads the Zone 6 current sound volume level from the QTPRO. |
| **Command Packet** | "ZOGET,MCUR5<CR>" |
| **Data in Response Message** | A variable Ascii string representing the device's minimum sound volume level.<br>For example:<br>"{ZOGET, MCUR5=11}<CR>" |

## 6.6 Get Mute

| | |
|---|---|
| **Opcode** | "MUTE" |
| **Description** | The PC application reads the Zone 6 mute state from the QTPRO. |
| **Command Packet** | "ZOGET, MUTE5<CR>" |
| **Data in Response Message** | A single Ascii byte representing the device's mute state, where '0' = unmuted, and '1' = muted<br>For example:<br>"{ZOGET, MUTE5=1}<CR>" |

## 6.7 Set Mute

| | |
|---|---|
| **Opcode** | “MUTE” |
| **Description** | The PC application writes the Zone 6 mute state to the QTPRO. |
| **Data** | A single Ascii byte representing the device’s mute state, where ‘0’ = unmuted, and ‘1’ = muted.<br>For example:<br>“ZOSET, MUTE5=1<CR>” sets Zone 6 to Mute Active. |

## 6.8 Increment Masking Max Volume

| | |
|---|---|
| **Opcode** | “INCM” |
| **Description** | The PC application commands the Qt-Pro to increment the masking max volume level. Volume will be increased by ½ dB. |
| **Data** | No data.<br>If the Masking Max value is already at its maximum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,INCM5<CR>” increments the Zone 6 max sound volume level. |
| **Standard Response** | “{ACK, INCM5}<CR>” |

## 6.9 Decrement Masking Max Volume

| | |
|---|---|
| **Opcode** | “DECM” |
| **Description** | The PC application commands the Qt-Pro to decrement the masking max volume level. Volume will be decreased by ½ dB. |
| **Data** | No data.<br>If the Masking Max value is already at its minimum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,DECM5<CR>” decrements the Zone 6 max sound volume level. |
| **Standard Response** | “{ACK, DECM5}<CR>” |

## 6.10 Increment Masking Min Volume

| | |
|---|---|
| **Opcode** | “INCI” |
| **Description** | The PC application commands the Qt-Pro to increment the masking min volume level. Volume will be increased by ½ dB. |
| **Data** | No data.<br>If the Masking Min value is already at its maximum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,INCI5<CR>” increments the Zone 6 min masking volume level. |
| **Standard Response** | “{ACK, INCI5}<CR>” |

## 6.11 Decrement Masking Min Volume

| | |
|---|---|
| **Opcode** | “DECI” |
| **Description** | The PC application commands the Qt-Pro to decrement the masking min volume level. Volume will be decreased by ½ dB. |
| **Data** | No data.<br>If the Masking Min value is already at its minimum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,DECI5<CR>” decrements the Zone 6 min masking volume level. |
| **Standard Response** | “{ACK, DECI5}<CR>” |

## 6.12 Get Input A Max Volume

| | |
|---|---|
| **Opcode** | “INAM” |
| **Description** | The PC application reads the Zone 6 Auxiliary audio Input A maximum volume level from the Qt-Pro. |
| **Command Packet** | “ZOGET, INAM5<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s Auxiliary audio Input A Max volume level.<br>For example:<br>“{ZOGET, INAM5=9}<CR>” |

## 6.13 Set Input A Max Volume

| | |
|---|---|
| **Opcode** | “INAM” |
| **Description** | The PC application writes the Zone 6 Auxiliary audio Input A music maximum volume level to the Qt-Pro. |
| **Data** | A variable Ascii string representing the device’s Auxiliary audio Input A volume level. The valid volume level range is ‘0’ to ‘30’.<br>For example:<br>“ZOSET, INAM5=9<CR>” sets Zone 6 Input A Max volume level to 9. |
| **Standard Response** | “{ACK, INAM5=9}<CR>” |

## 6.14 Get Input A Min Volume

| | |
|---|---|
| **Opcode** | “INAI” |
| **Description** | The PC application reads the Zone 6 Auxiliary audio Input A minimum volume level from the Qt-Pro. |
| **Command Packet** | “ZOGET, INAI5<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s Auxiliary audio Input A Min volume level.<br>For example:<br>“{ZOGET, INAI5=9}<CR>” |

## 6.15 Set Input A Min Volume

| | |
|---|---|
| **Opcode** | “INAI” |
| **Description** | The PC application writes the Zone 6 Auxiliary audio Input A music minimum volume level to the Qt-Pro. |
| **Data** | A variable Ascii string representing the device’s Auxiliary audio Input A Min volume level. The valid volume level range is ‘0’ to ‘30’.<br>For example:<br>“ZOSET, INAI5=9<CR>” sets Zone 6 Input A Min volume level to 9. |
| **Standard Response** | “{ACK, INAI5=9}<CR>” |

## 6.16 Get Input B Max Volume

| | |
|---|---|
| **Opcode** | “INBM” |
| **Description** | The PC application reads the Zone 6 Auxiliary audio Input B maximum volume level from the Qt-Pro. |
| **Command Packet** | “ZOGET, INBM5<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s Auxiliary audio Input B Max volume level.<br>For example:<br>“{ZOGET, INBM5=9}<CR>” |

## 6.17 Set Input B Max Volume

| | |
|---|---|
| **Opcode** | “INBM” |
| **Description** | The PC application writes the Zone 6 Auxiliary audio Input B music maximum volume level to the Qt-Pro. |
| **Data** | A variable Ascii string representing the device’s Auxiliary audio Input B Max volume level. The valid volume level range is ‘0’ to ‘30’.<br>For example:<br>“ZOSET, INBM5=9<CR>” sets Zone 6 Input B Max volume level to 9. |
| **Standard Response** | “{ACK, INBM5=9}<CR>” |

## 6.18 Get Input B Min Volume

| | |
|---|---|
| **Opcode** | “INBI” |
| **Description** | The PC application reads the Zone 6 Auxiliary audio Input B minimum volume level from the Qt-Pro. |
| **Command Packet** | “ZOGET, INBI5<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s Auxiliary audio Input B Max volume level.<br>For example:<br>“{ZOGET, INBI5=9}<CR>” |

## 6.19 Set Input B Min Volume

| | |
|---|---|
| **Opcode** | “INBI” |
| **Description** | The PC application writes the Zone 6 Auxiliary audio Input B music minimum volume level to the Qt-Pro. |
| **Data** | A variable Ascii string representing the device’s Auxiliary audio Input B Min volume level. The valid volume level range is ‘0’ to ‘30’.<br>For example:<br>“ZOSET, INBI5=9<CR>” sets Zone 6 Input B Min volume level to 9. |
| **Standard Response** | “{ACK, INBI5=9}<CR>” |

## 6.20 Get Input A Current Volume

| | |
|---|---|
| **Opcode** | “INAC” |
| **Description** | The PC application reads the Zone 6 Auxiliary audio Input A volume level from the Qt-Pro. |
| **Command Packet** | “ZOGET, INAC5<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s Auxiliary audio Input A volume level.<br>For example:<br>“{ZOGET, INAC5}<CR>” |
| **Standard Response** | “{ZOGET, INAC5=9}<CR>” |

## 6.21 Get Input B Current Volume

| | |
|---|---|
| **Opcode** | “INBC” |
| **Description** | The PC application reads the Zone 6 Auxiliary audio Input B volume level from the Qt-Pro. |
| **Command Packet** | “ZOGET, INAC5<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s Auxiliary audio Input B volume level.<br>For example:<br>“{ZOGET, INBC5}<CR>” |
| **Standard Response** | “{ZOGET, INBC5=9}<CR>” |

## 6.22 Increment Audio A Max Volume

| | |
|---|---|
| **Opcode** | "ICAM" |
| **Description** | The PC application commands the Qt-Pro to increment the Audio A max volume level. Volume will be increased by 1 dB. |
| **Data** | No data.<br>If the Audio A Max value is already at its maximum level, then the Qt-Pro will respond with a "NAK".<br>For example:<br>"ZOSET,ICAM5<CR>" increments the Zone 6 max Audio A volume level. |
| **Standard Response** | "{ACK, ICAM5}<CR>" |

## 6.23 Decrement Audio A Max Volume

| | |
|---|---|
| **Opcode** | "DCAM" |
| **Description** | The PC application commands the Qt-Pro to decrement the Audio A max volume level. Volume will be decreased by 1 dB. |
| **Data** | No data.<br>If the Audio A Max value is already at its minimum level, then the Qt-Pro will respond with a "NAK".<br>For example:<br>"ZOSET,DCAM5<CR>" decrements the Zone 6 max Audio A volume level. |
| **Standard Response** | "{ACK, DCAM5}<CR>" |

## 6.24 Increment Audio A Min Volume

| | |
|---|---|
| **Opcode** | “ICAI” |
| **Description** | The PC application commands the Qt-Pro to increment the Audio A min volume level. Volume will be increased by 1 dB. |
| **Data** | No data.<br>If the Audio A Min value is already at its maximum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,ICAI5<CR>” increments the Zone 6 min Audio A volume level. |
| **Standard Response** | “{ACK, ICAI5}<CR>” |

## 6.25 Decrement Audio A Min Volume

| | |
|---|---|
| **Opcode** | “DCAI” |
| **Description** | The PC application commands the Qt-Pro to decrement the Audio A min volume level. Volume will be decreased by 1 dB. |
| **Data** | No data.<br>If the Audio A Min value is already at its minimum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,DCAI5<CR>” decrements the Zone 6 min Audio A volume level. |
| **Standard Response** | “{ACK, DCAI5}<CR>” |

## 6.26 Increment Audio B Max Volume

| | |
|---|---|
| **Opcode** | “ICBM” |
| **Description** | The PC application commands the Qt-Pro to increment the Audio B max volume level. Volume will be increased by 1 dB. |
| **Data** | No data.<br>If the Audio B Max value is already at its maximum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,ICBM5<CR>” increments the Zone 6 max Audio A volume level. |
| **Standard Response** | “{ACK, ICBM5}<CR>” |

## 6.27 Decrement Audio B Max Volume

| | |
|---|---|
| **Opcode** | “DCBM” |
| **Description** | The PC application commands the Qt-Pro to decrement the Audio B max volume level. Volume will be decreased by 1 dB. |
| **Data** | No data.<br>If the Audio B Max value is already at its minimum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,DCBM5<CR>” decrements the Zone 6 max Audio B volume level. |
| **Standard Response** | “{ACK, DCBM5}<CR>” |

## 6.28 Increment Audio B Min Volume

| | |
|---|---|
| **Opcode** | “ICBI” |
| **Description** | The PC application commands the Qt-Pro to increment the Audio B min volume level. Volume will be increased by 1 dB. |
| **Data** | No data.<br>If the Audio B Min value is already at its maximum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,ICBI5<CR>” increments the Zone 6 min Audio B volume level. |
| **Standard Response** | “{ACK, ICBI5}<CR>” |

## 6.29 Decrement Audio B Min Volume

| | |
|---|---|
| **Opcode** | “DCBI” |
| **Description** | The PC application commands the Qt-Pro to decrement the Audio B min volume level. Volume will be decreased by 1 dB. |
| **Data** | No data.<br>If the Audio B Min value is already at its minimum level, then the Qt-Pro will respond with a “NAK”.<br>For example:<br>“ZOSET,DCBI5<CR>” decrements the Zone 6 min Audio B volume level. |
| **Standard Response** | “{ACK, DCBI5}<CR>” |

## 6.30 Get Auto Ramping Enable

| | |
|---|---|
| **Opcode** | "ENAR" |
| **Description** | The PC application reads the Zone 6 Auto Ramping Enable setting from the Qt-Pro. |
| **Command Packet** | "ZOGET,ENAR5<CR>" |
| **Data in Response Message** | A variable Ascii string representing the device's Auto Ramping Enable setting. The valid values are:<br>0 – Auto Ramping Disabled<br>1 – Auto Ramping Enabled<br>For example:<br>"{ZOGET,ENAR5=0}<CR>" |

## 6.31 Set Auto Ramping Enable

| | |
|---|---|
| **Opcode** | "ENAR" |
| **Description** | The PC application writes the Auto Ramping Enable setting to the Qt-Pro. |
| **Data** | A variable Ascii string representing the device's Auto Ramping Enable Setting. The valid values are:<br>0 – Auto Ramping Disabled<br>1 – Auto Ramping Enabled<br>Note: Enabling Auto Ramping on a zone checks that all dates have been set correctly. The command will respond with a NAK if dates are missing or invalid. The dates need to be progressing so 2nd date cannot be earlier than the 1st one.<br>For example:<br>"ZOSET,ENAR5=1<CR>" Enables Zone 6 Auto Ramping. |
| **Standard Response** | "{ACK, ENAR5=1}<CR>" |

## 6.32 Get Auto Ramping Levels

| | |
|---|---|
| **Opcode** | "ARLV" |
| **Description** | The PC application reads the Zone 6 Auto Ramping level settings from the Qt-Pro. |
| **Command Packet** | "ZOGET,ARLV5a<CR>" |
| **Data in Response Message** | A variable Ascii string representing the device's Auto Ramping Level setting. The byte order of the packet is:<br><br>**Byte** — **Value**<br>0 — 'Z'<br>1 — 'O'<br>2-9 — "GET,ARLV"<br>10 — Zone in the range of 0 – 5<br>11 — Level number (see table below)<br><br>For example:<br>"{ZOGET,ARLV5a=10}<CR>" |

## 6.33 Set Auto Ramping Levels

| | |
|---|---|
| **Opcode** | "ARLV" |
| **Description** | The PC application writes the Auto Ramping Level settings to the Qt-Pro. |
| **Data** | A variable Ascii string representing the device's Auto Ramping Level setting. The byte order of the packet is:<br><br>**Byte** — **Value**<br>0 — 'Z'<br>1 — 'O'<br>2-9 — "SET,ARLV"<br>10 — Zone in the range of 0 – 5<br>11 — Level number (see table below)<br>12 — Level Value<br><br>For example:<br>"ZOSET,ARLV5a=20<CR>" Sets Auto Ramp start level to 20. |

| | |
|---|---|
| **Standard Response** | "{ACK, ARLV5a=20}<CR>" |

**Auto-Ramping Levels Table**

| Level | Level number |
|---|---|
| 'a' | 0 (start) |
| 'b' | 1 |
| 'c' | 2 |

## 6.34 Get Auto Ramping Dates

| | |
|---|---|
| **Opcode** | "ARDT" |
| **Description** | The PC application reads the Auto Ramping Date settings from the Qt-Pro. |
| **Command Packet** | "ZOGET,ARDT5a<CR>" |
| **Data in Response Message** | A variable Ascii string representing the device's Auto Ramping Level setting. The byte order of the packet is:<br><br>**Byte** — **Value**<br>0 — 'Z'<br>1 — 'O'<br>2-9 — "GET,ARDT"<br>10 — Zone in the range of 0 – 5<br>11 — Date number (see table below)<br><br>For example:<br>"{ZOGET,ARDT5a="2017-08-10"}<CR>" |

## 6.35 Set Auto Ramping Dates

| | |
|---|---|
| **Opcode** | "ARDT" |
| **Description** | The PC application writes the Auto Ramping Date settings to the Qt-Pro. |
| **Data** | A variable Ascii string representing the device's Auto Ramping Level setting. The byte order of the packet is:<br><br>**Byte** — **Value** |

| | |
|---|---|
| | 0 ‘Z’<br>1 ‘O’<br>2-9 “SET,ARDT”<br>10 Zone in the range of 0 – 5<br>11 Date number (see table below)<br>12 Date string in “yyyy-mm-dd” format<br>For example:<br>“ZOSET,ARDT5a=2017-08-10<CR>” Sets Auto Ramp start date to Aug 10 2017. |
| **Standard Response** | “{ACK, ARDT5a=2017-08-10}<CR>” |

**Auto-Ramping Dates Table**

| Date | Date number |
|---|---|
| ‘a’ | 0 (start) |
| ‘b’ | 1 |
| ‘c’ | 2 |
| ‘d’ | 3 |
| ‘e’ | 4 |
| ‘f’ | 5 |

## *6.36 Get Auto Ramp Exclusion*

| | |
|---|---|
| **Opcode** | “AREX” |
| **Description** | The PC application reads the Auto Ramping Exclusion settings from the Qt-Pro. |
| **Command Packet** | “ZOGET,AREX5<CR>” |
| **Data in Response Message** | A fixed-size 3-byte Ascii string, where each byte represents:<br>‘0’ – Do Not Exclude<br>‘1’ – Exclude<br>and Byte 0 = Exclude Friday, byte 1 = Exclude Saturday, and byte 2 = Exclude Sunday.<br>For example:<br>“{ZOGET,AREX5=011}<CR>” |

## 6.37 Set Auto Ramp Exclusion

| | |
|---|---|
| **Opcode** | “AREX” |
| **Description** | The PC application writes the Auto Ramping Exclusion Setting to the Qt-Pro. |
| **Data** | A fixed-size 3-byte Ascii string, where each byte represents:<br>‘0’ – Do Not Exclude<br>‘1’ – Exclude<br>and Byte 0 = Exclude Friday, byte 1 = Exclude Saturday, and byte 2 = Exclude Sunday.<br>For example:<br>“ZOSET,AREX5=011<CR>” Sets the Zone 6 Auto Ramping to only exclude Saturday and Sunday |
| **Standard Response** | “{ACK,AREX5=011}<CR>” |

## 6.38 Get Masking Begin Ramp to Max Time

| | |
|---|---|
| **Opcode** | “RMAX” |
| **Description** | The PC application reads the Masking Begin Ramp to Max Time from the Qt-Pro. |
| **Command Packet** | “ZOGET,RMAX5<CR>” |
| **Data in Response Message** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to max time.<br>For example:<br>“{ZOGET,RMAX5=0700}<CR>” |

## 6.39 Set Masking Begin Ramp to Max Time

| | |
|---|---|
| **Opcode** | “RMAX” |
| **Description** | The PC application writes the Masking Begin Ramp to Max Time value to the Qt-Pro. |
| **Data** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to max time.<br>For example:<br>“ZOSET,RMAX5=0700<CR>” sets Zone 6 Begin Ramp to max Time to 7am. |
| **Standard Response** | “{ACK,RMAX5=0700}<CR>” |

## 6.40 Get Masking Begin Ramp to Min Time

| | |
|---|---|
| **Opcode** | “RMIN” |
| **Description** | The PC application reads the Masking Begin Ramp to Min Time from the QTPRO. |
| **Command Packet** | “ZOGET,RMIN5<CR>” |
| **Data in Response Message** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to min time.<br>For example:<br>“{ZOGET,RMIN5=0700}<CR>” |

## 6.41 Set Masking Begin Ramp to Min Time

| | |
|---|---|
| **Opcode** | “RMIN” |
| **Description** | The PC application writes the Begin Ramp to Min Time value to the QTPRO. |
| **Data** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to min time.<br>For example:<br>“ZOSET,RMIN5=1800<CR>” sets Zone 6 Begin Ramp to min Time to 6pm. |
| **Standard Response** | “{ACK,RMIN5=1800}<CR>” |

## 6.42 Get Audio A Begin Ramp to Max Time

| | |
|---|---|
| **Opcode** | “RAMX” |
| **Description** | The PC application reads the Audio A Begin Ramp to Max Time from the Qt-Pro. |
| **Command Packet** | “ZOGET,RAMX5<CR>” |

| | |
|---|---|
| **Data in Response Message** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to max time.<br>For example:<br>"{ZOGET,RAMX5=0700}<CR>" |

## 6.43 Set Audio A Begin Ramp to Max Time

| | |
|---|---|
| **Opcode** | "RAMX" |
| **Description** | The PC application writes the Audio A Begin Ramp to Max Time value to the Qt-Pro. |
| **Data** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to max time.<br>For example:<br>"ZOSET,RAMX5=0700<CR>" sets Zone 6 Audio A Begin Ramp to max Time to 7am. |
| **Standard Response** | "{ACK,RAMX5=0700}<CR>" |

## 6.44 Get Audio A Begin Ramp to Min Time

| | |
|---|---|
| **Opcode** | "RAMN" |
| **Description** | The PC application reads the Audio A Begin Ramp to Min Time from the Qt-Pro. |
| **Command Packet** | "ZOGET,RAMN5<CR>" |
| **Data in Response Message** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to min time.<br>For example:<br>"{ZOGET,RAMN5=0700}<CR>" |

## 6.45 Set Audio A Begin Ramp to Min Time

| | |
|---|---|
| **Opcode** | "RAMN" |
| **Description** | The PC application writes the Audio A Begin Ramp to Min Time value to the Qt-Pro. |
| **Data** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to min time.<br>For example: |

| | |
|---|---|
| | “ZOSET,RAMN5=1800<CR>” sets Zone 6 Audio A Begin Ramp to min Time to 6pm. |
| **Standard Response** | “{ACK,RAMN5=1800}<CR>” |

## 6.46 Get Audio B Begin Ramp to Max Time

| | |
|---|---|
| **Opcode** | “RBMX” |
| **Description** | The PC application reads the Audio B Begin Ramp to Max Time from the Qt-Pro. |
| **Command Packet** | “ZOGET,RBMX5<CR>” |
| **Data in Response Message** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to max time.<br>For example:<br>“{ZOGET,RBMX5=0700}<CR>” |

## 6.47 Set Audio B Begin Ramp to Max Time

| | |
|---|---|
| **Opcode** | “RBMX” |
| **Description** | The PC application writes the Audio B Begin Ramp to Max Time value to the Qt-Pro. |
| **Data** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to max time.<br>For example:<br>“ZOSET,RBMX5=0700<CR>” sets Zone 6 Audio B Begin Ramp to max Time to 7am. |
| **Standard Response** | “{ACK,RBMX5=0700}<CR>” |

## 6.48 Get Audio B Begin Ramp to Min Time

| | |
|---|---|
| **Opcode** | “RBMN” |
| **Description** | The PC application reads the Audio B Begin Ramp to Min Time from the Qt-Pro. |
| **Command Packet** | “ZOGET,RBMN5<CR>” |
| **Data in Response Message** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to min time. |

| | |
|---|---|
| | For example:<br>“{ZOGET,RBMN5=0700}<CR>” |

## 6.49 Set Audio B Begin Ramp to Min Time

| | |
|---|---|
| **Opcode** | “RBMN” |
| **Description** | The PC application writes the Audio B Begin Ramp to Min Time value to the Qt-Pro. |
| **Data** | A fixed-size 4-byte Ascii string representing the time in Hours and Minutes (HHMM) of when to begin ramp to min time.<br>For example:<br>“ZOSET,RBMN5=1800<CR>” sets Zone 6 Audio B Begin Ramp to min Time to 6pm. |
| **Standard Response** | “{ACK,RBMN5=1800}<CR>” |

## 6.50 Get Masking Ramp Interval

| | |
|---|---|
| **Opcode** | “RINT” |
| **Description** | The PC application reads the Masking Ramp Interval option from the Qt-Pro. |
| **Command Packet** | “ZOGET,RINT5<CR>” |
| **Data in Response Message** | An Ascii String representing the Masking Ramp Interval in minutes. The maximum allowable value is 120 minutes.<br>For example:<br>“{ZOGET,RINT5=98}<CR>” |

## 6.51 Set Masking Ramp Interval

| | |
|---|---|
| **Opcode** | “RINT” |
| **Description** | The PC application writes the Masking Ramp Interval option to the Qt-Pro. |
| **Data** | An Ascii String representing the Masking Ramp Interval in minutes. The maximum allowable value is 120 minutes.<br>For example:<br>“ZOSET,RINT5=98<CR>” sets Zone 6 Masking Ramp Interval to 98 minutes. |

| | |
|---|---|
| **Standard Response** | "{ACK,RINT5=98}<CR>" |

## 6.52 Get Audio A Ramp Interval

| | |
|---|---|
| **Opcode** | "RAIT" |
| **Description** | The PC application reads the Audio A Ramp Interval option from the Qt-Pro. |
| **Command Packet** | "ZOGET,RAIT5<CR>" |
| **Data in Response Message** | An Ascii String representing the Audio A Ramp Interval in minutes. The maximum allowable value is 120 minutes.<br>For example:<br>"{ZOGET,RAIT5=98}<CR>" |

## 6.53 Set Audio A Ramp Interval

| | |
|---|---|
| **Opcode** | "RAIT" |
| **Description** | The PC application writes the Audio A Ramp Interval option to the Qt-Pro. |
| **Data** | An Ascii String representing the Audio A Ramp Interval in minutes. The maximum allowable value is 120 minutes.<br>For example:<br>"ZOSET,RAIT5=98<CR>" sets Zone 6 Audio A Ramp Interval to 98 minutes. |
| **Standard Response** | "{ACK,RAIT5=98}<CR>" |

## 6.54 Get Audio B Ramp Interval

| | |
|---|---|
| **Opcode** | "RBIT" |
| **Description** | The PC application reads the Audio B Ramp Interval option from the Qt-Pro. |
| **Command Packet** | "ZOGET,RBIT5<CR>" |
| **Data in Response Message** | An Ascii String representing the Audio B Ramp Interval in minutes. The maximum allowable value is 120 minutes.<br>For example:<br>"{ZOGET,RBIT5=98}<CR>" |

## 6.55 Set Audio B Ramp Interval

| | |
|---|---|
| **Opcode** | "RBIT" |
| **Description** | The PC application writes the Audio B Ramp Interval option to the Qt-Pro. |
| **Data** | An Ascii String representing the Audio B Ramp Interval in minutes. The maximum allowable value is 120 minutes.<br>For example:<br>"ZOSET,RBIT5=98<CR>" sets Zone 6 Audio B Ramp Interval to 98 minutes. |
| **Standard Response** | "{ACK,RBIT5=98}<CR>" |

## 6.56 Get Emitter Type

| | |
|---|---|
| **Opcode** | "EMIT" |
| **Description** | The PC application reads the Emitter Type option from the QTPRO. |
| **Data** | An Ascii String representing the Emitter Type. Possible values are :<br>0 – Standard Emitter<br>1 – Active Emitter<br>For example:<br>"ZOGET,EMIT5<CR>" Gets the emitter type for zone 6 |
| **Standard Response** | "{ACK,EMIT5=0}<CR>" |

## 6.57 Set Emitter Type

| | |
|---|---|
| **Opcode** | "EMIT" |
| **Description** | The PC application writes the Emitter Type option from the QTPRO. |
| **Data** | An Ascii String representing the Emitter Type. Possible values are :<br>0 – Standard Emitter<br>1 – Active Emitter<br>For example:<br>"ZOSET,EMIT5=1<CR>" Gets the emitter type for zone 6 |
| **Standard Response** | "{ACK,EMIT5=1}<CR>" |

## 6.58 Get Spectrum Type

| | |
|---|---|
| **Opcode** | "SPEC" |
| **Description** | The PC application reads the Spectrum Type option from the QTPRO. |
| **Data** | An Ascii String representing the Spectrum Type. Possible values are:<br>0 – Standard Spectrum<br>1 – Active Emitter Spectrum<br>For example:<br>"ZOGET,SPEC5<CR>" Gets the spectrum type for zone 6 |
| **Standard Response** | "{ACK,SPEC5=0}<CR>" |

## 6.59 Set Spectrum Type

| | |
|---|---|
| **Opcode** | "SPEC" |
| **Description** | The PC application writes the Spectrum Type option from the QTPRO. |
| **Data** | An Ascii String representing the Spectrum Type. Possible values are:<br>0 – Standard Spectrum<br>1 – Active Emitter Spectrum<br>For example:<br>"ZOSET,SPEC5=1<CR>" Sets the spectrum type for zone 6 |
| **Standard Response** | "{ACK,SPEC5=1}<CR>" |

## 6.60 Get Emitter Configuration

| | |
|---|---|
| **Opcode** | "EMCF" |
| **Description** | The PC application gets the emitter configuration from the QTPRO. |
| **Data** | An Ascii String representing the configuration. Possible values are:<br>0 – Automatically sensing emitter on power cycle<br>1 – User set emitter settings<br>For example:<br>"ZOGET,EMCF5=1<CR>" Gets the emitter configuration for zone 6 |
| **Standard Response** | "{ACK,EMCF5=1}<CR>" |

## 6.61 Set Emitter Configuration

| | |
|---|---|
| **Opcode** | "EMCF" |
| **Description** | The PC application resets the emitter configuration from the QTPRO. |
| **Data** | An Ascii String representing the configuration. Possible values are:<br>1 – Reenable auto-sensing emitters on power cycle<br>For example:<br>"ZOSET,EMCF5=1<CR>" Sets the emitter configuration for zone 6 |
| **Standard Response** | "{ACK,EMCF5=1}<CR>" |

## 6.62 Get Weekend Schedule

| | |
|---|---|
| **Opcode** | "WEEK" |
| **Description** | The PC application reads the Weekend Schedule option from the QTPRO. |
| **Command Packet** | "ZOGET,WEEK5<CR>" |
| **Data in Response Message** | A fixed-size 3-byte Ascii string, where each byte represents:<br>'1' – Max/Min<br>'2' – Min<br>and Byte 0 = Friday schedule, byte 1 = Saturday schedule, and byte 2 = Sunday Schedule.<br>For example:<br>"{ZOGET,WEEK5=122}<CR>" |

## 6.63 Set Weekend Schedule

| | |
|---|---|
| **Opcode** | "WEEK" |
| **Description** | The PC application writes the Weekend Schedule option to the QTPRO. |
| **Data** | A fixed-size 3-byte Ascii string, where each byte represents:<br>'1' – Max/Min<br>'2' – Min<br>and Byte 0 = Friday schedule, byte 1 = Saturday schedule, and byte 2 = Sunday Schedule.<br>For example:<br>"ZOSET,WEEK5=011<CR>" sets the Zone 6 Friday Schedule to Max/Min, the Saturday Schedule to Min, and the Sunday Schedule to Min. |

| | |
|---|---|
| **Standard Response** | "{ACK,WEEK5=122}<CR>" |

## 6.64 Get Masking Equalizer

| | |
|---|---|
| **Opcode** | "MEQU" |
| **Description** | The PC application reads the Masking Equalizer from Frequency x and Zone z. The byte order of the packet is<br>**Byte** **Value**<br>0 'Z'<br>1 'O'<br>2-9 "GET,MEQU"<br>10 Zone in the range of 0 – 5<br>11 Frequency – see table below |
| **Command Packet** | "ZOGET,MEQU1a<CR>" Get Masking Equalizer from Frequency 125, Zone 2. |
| **Data in Response Message** | A variable Ascii string in the range of -12 to +12.<br>For example:<br>"{ZOGET,MEQU1a=-12}<CR>"--- Gets a value of -12 from Frequency 125, Zone 2. |

## 6.65 Set Masking Equalizer

| | |
|---|---|
| **Opcode** | "MEQU" |
| **Description** | The PC application writes the Masking Equalizer for Frequency x and Zone z. |
| **Data** | A variable Ascii string representing the Masking Equalizer for Frequency x and Zone z. For example,<br>"ZOSET,MEQU1a=10<CR> sets a value of +10 to Frequency 125, Zone 2. |
| **Standard Response** | "{ACK,MEQU1a=10}<CR>" |

**MASKING EQUALIZER TABLE**

| Value | Frequency |
|---|---|
| ‘a’ | 125 |
| ‘b’ | 160 |
| ‘c’ | 200 |
| ‘d’ | 250 |
| ‘e’ | 315 |
| ‘f’ | 400 |
| ‘g’ | 500 |
| ‘h’ | 630 |
| ‘i’ | 800 |
| ‘j’ | 1000 |
| ‘k’ | 1250 |
| ‘l’ | 1600 |
| ‘m’ | 2000 |
| ‘n’ | 2500 |
| ‘o’ | 3150 |
| ‘p’ | 4000 |
| ‘q’ | 5000 |
| ‘r’ | 6300 |

## *6.66 Get Input A Equalizer*

<table>
<tr><td>Opcode</td><td>“AEQU”</td></tr>
<tr><td>Description</td><td>The PC application reads the Input A Equalizer from Frequency x and Zone z. The byte order of the packet is
<table>
<tr><th>Byte</th><th>Value</th></tr>
<tr><td>0</td><td>‘Z’</td></tr>
<tr><td>1</td><td>‘O’</td></tr>
<tr><td>2-9</td><td>“GET,AEQU”</td></tr>
<tr><td>10</td><td>Zone in the range of 0 – 5</td></tr>
<tr><td>11</td><td>Frequency – see table below</td></tr>
</table></td></tr>
<tr><td>Command Packet</td><td>“ZOGET,AEQU1a<CR>” Get Input A Equalizer from Frequency 250, Zone 2.</td></tr>
<tr><td>Data in Response Message</td><td>A variable Ascii string in the range of -12 to +12.<br>For example:</td></tr>
</table>

| | |
|---|---|
| | “{ZOGET,AEQU1a=-12}<CR>”--- Gets a value of -12 from Frequency 250, Zone 2. |

## 6.67 Set Input A Equalizer

| | |
|---|---|
| **Opcode** | “AEQU” |
| **Description** | The PC application writes the Input A Equalizer for Frequency x and Zone z. |
| **Data** | A variable Ascii string representing the Input A Equalizer for Frequency x and Zone z. For example,<br>“ZOSET,AEQU1b=11<CR> sets a value of +11 to Frequency 500, Zone 2. |
| **Standard Response** | “{ACK,AEQU1b=11}<CR>” |

**INPUT ‘X’ EQUALIZER TABLE**

| Value | Frequency |
|---|---|
| ‘a’ | 250 |
| ‘b’ | 500 |
| ‘c’ | 1000 |
| ‘d’ | 2000 |
| ‘e’ | 4000 |
| ‘f’ | 8000 |

## 6.68 Get Input B Equalizer

| | |
|---|---|
| **Opcode** | “BEQU” |
| **Description** | The PC application reads the Input A Equalizer from Frequency x and Zone z. The byte order of the packet is |

| Byte | Value |
|---|---|
| 0 | ‘Z’ |
| 1 | ‘O’ |
| 2-9 | “GET,BEQU” |
| 10 | Zone in the range of 0 – 5 |
| 11 | Frequency – see table above |

| | |
|---|---|
| **Command Packet** | “ZOGET,BEQU1a<CR>” Get Input B Equalizer from Frequency 250, Zone 2. |
| **Data in Response Message** | A variable Ascii string in the range of -12 to +12.<br>For example:<br>“{ZOGET,BEQU1a=-12}<CR>”--- Gets a value of -12 from Frequency 250, Zone 2. |

## 6.69 Set Input B Equalizer

| | |
|---|---|
| **Opcode** | “BEQU” |
| **Description** | The PC application writes the Input B Equalizer for Frequency x and Zone z. |
| **Data** | A variable Ascii string representing the Input B Equalizer for Frequency x and Zone z. For example,<br>“ZOSET,BEQU1b=11<CR> sets a value of +11 to Frequency 500, Zone 2. |
| **Standard Response** | “{ACK,BEQU1b=11}<CR>” |

## 6.70 Get Zone Name

| | |
|---|---|
| **Opcode** | “ZNAM” |
| **Description** | The PC application reads the Zone 6 zone name from the QTPRO. |
| **Command Packet** | “ZOGET,ZNAM5<CR>” |
| **Data in Response Message** | A variable Ascii string representing the device’s Zone 6 zone name.<br>For example:<br>“{ZOGET,ZNAM5=SOUTH POD}<CR>” |

## 6.71 Set Zone Name

| | |
|---|---|
| **Opcode** | “ZNAM” |
| **Description** | The PC application writes the Zone 6 zone name to the QTPRO. |
| **Data** | A variable Ascii string representing the device’s Zone 6 zone name.<br>For example:<br>“ZOSET,ZNAM5=SOUTH PAD<CR>” sets Zone 6 zone name to SOUTH POD |

| | |
|---|---|
| **Standard Response** | “{ACK, ZNAM5=SOUTH POD}<CR>” |

# 7 Summary of AL Read Only Batch Commands

There are four general purpose Read Only Commands available in the system

| Opcode | | GET | |
|---|---|---|---|
| Read All System Variables | | “SYSV” | Reads all variables in the system that are not zone related |
| Read All Zone Variables | | “ZONE” | Reads all variables associated with a specific zone, except for the equalizer values |
| Read All Masking Equalizers | | “MEQU” | Read all Masking Equalizer values from a specific zone. |
| Read All Input A and Input B Equalizers | | ‘INAB” | Read all Input A and Input B Equalizer values from a specific zone. |

## *7.1 Read All System Variables*

| | |
|---|---|
| **Opcode** | “SYSV” |
| **Description** | The PC application reads all variables in the system that are not zone related.. |
| **Command Packet** | “ALSYSV<CR>” |
| **Data in Response Message** | There are 20 variables in the system that can be read all at once by the server via the “ALSYSV” Command. Therefore, 20 comma-separated 58sci string subfields will be included in the response message. Each subfield is assigned a prefix ID value, defined in the following table: |

| Subfield | ID |
|---|---|
| MAC Address | “MACA” |
| IP Mode | “IPMO” |
| IP Address | “IPAD’’ |
| Gateway | “GATE” |
| Net Mask | “MASK” |
| Primary DNS | “PDNS” |
| Secondary DNS | “SDNS” |
| Host Name | “HNAM” |
| User Name | “UNAM” |
| User Password | “UPWD” |
| Unit Name | “UNIT” |
| Location | “LOCA” |
| Date | “DATE” |
| Day of the Week | “DAYW” |
| Time | “TIME” |
| Time Zone | “TZON” |

| | | | | |
|---|---|---|---|---|
| | | DST Begin Time | “DSTB” | |
| | | DST End Time | “DSTE” | |
| | | Self-Test Diagnostic Results | “DIAG” | |
| | | Destination for Errors Reporting | “EADD” | |
| | | Firmware Version | “FIRM” | |
| | | Front Panel Lock Status | “LOCK” | |
| | | Audio A Sensitivity | “SENA” | |
| | | Audio B Sensitivity | “SENB” | |

An example response message would look like this:

| | |
|---|---|
| “{ALSYSV=MACA=123456789ABC, | MAC Address = 123456789ABC |
| IPMO=0, | IP Mode = Dynamic |
| IPAD=192.168.9.76, | IP Address = 192.168.9.76 |
| GATE=192.168.1.1, | Gateway = 192.168.1.1 |
| MASK=255.255.255.0, | Net Mask = 255.255.255.0 |
| PDNS=Cono_555, | Primary DNS = Cono_555 |
| SDNS=Texas97, | Secondary DNS = Texas97 |
| HNAM=QT-39, | Host Name = QT-39 |
| UNAM=Terry, | User Name = Terry |
| UPWD=Dinkle-33!, | Password = Dinkle-33! |
| UNIT=Acme_1, | Unit Name = Acme_1 |
| LOCA=1st Floor, | Location = 1st Floor |
| DATE=09052014, | Date = 09/05/2014 |
| DAYW=1, | Day of the Week = Monday |
| TIME=133536, | Time = 13:35:36 |
| TZON=-8, | Time Zone = Pacific Time |
| DSTB=0, | DST Begin Time = 2nd Sunday in March |
| DSTE=0, | DST End Time = 2nd Sunday in October |
| DIAG=00000000000, | Self-Test Diagnostic Results = no errors |
| EADD=A,B,C25,D,E” | Destination Port for Errors Reporting = 0x25 |
| FIRM=6.3.0” | Firmware version = v6.3.0 |
| LOCK=1, | Keypad Lock is disabled |
| SENA=1, | Audio A Sensitivity is High Signal |
| SENB=0}<CR>” | Audio B Sensitivity is Normal Signal |

## 7.2 Read All Zone Variables

| | |
|---|---|
| **Opcode** | "ZONE" |
| **Description** | The PC application reads all variables associated with a specific zone, except for the equalizer values. The "ALZONE" Read Zone command takes as its seventh byte the Zone Number ID, which is '0' – '5' for the QT-600 or '0' – '2' for the QT-300. For example, the Read Zone Command "ALZONE4" requests a dump of all Zone 5 variables |
| **Command Packet** | "ALZONEx<CR>" |
| **Data in Response Message** | There are 12 variables defined for each zone in the system that can be read all at once by the server via the "ALZONE" Command. Therefore, 12 comma-separated 60sci string subfields will be included in the response message. Each subfield is assigned a prefix ID value, defined in the following table. |

| Subfield | ID |
|---|---|
| Masking Max | "MMAX" |
| Masking Min | "MMIN" |
| Masking Current | "MCUR" |
| Mute State | "MUTE" |
| Input A Max | "INAM" |
| Input B Max | "INBM" |
| Input A Min | "INAI" |
| Input B Min | "INBI" |
| Input A Current Level | "INAC" |
| Input B Current Level | "INBC" |
| Auto Ramping Enable | "ENAR" |
| Auto Ramping Exclude | "AREX" |
| Begin Masking Ramp to Max Time | "RMAX" |
| Begin Masking Ramp to Min Time | "RMIN" |
| Begin Audio A Ramp To Max Time | "RAMX" |
| Begin Audio A Ramp To Min Time | "RAMN" |
| Begin Audio B Ramp To Max Time | "RBMX" |
| Begin Audio B Ramp To Min Time | "RBMN" |
| Masking Ramp Interval | "RINT" |
| Audio A Ramp Interval | "RAIT" |

| | | | | |
|---|---|---|---|---|
| | | Audio B Ramp Interval | “RBIT” | |
| | | Emitter Type | “EMIT” | |
| | | Spectrum Type | “SPEC” | |
| | | Weekend Schedule | “WEEK” | |
| | | Zone Name | “ZNAM” | |

An example response message would look like this

| | |
|---|---|
| | Read variables from Zone 5. |
| “{ALZONE4=MMAX=11, | Masking Max Volume = 11 |
| MMIN=11, | Masking Min Volume = 11 |
| MCUR=11, | Masking Current Volume = 11 |
| MUTE=1, | Muting is active |
| INAM=13, | Input A Max Volume = 13 |
| INBM=16, | Input B Max Volume = 16 |
| INAI=8, | Input A Max Volume = 8 |
| INBI=10, | Input B Max Volume = 10 |
| INAC=12, | Input A Current Volume = 12 |
| INBC=12, | Input B Current Volume = 12 |
| ENAR=0, | Auto Ramping Enable = Off |
| AREX=011, | Friday = Include |
| | Saturday = Exclude |
| | Sunday = Exclude |
| RMAX=0420, | Begin Masking Ramp to Max Time = 07:00am |
| RMIN=1080, | Begin Masking Ramp to Min Time = 06:00pm |
| RAMX=0420, | Begin Audio A Ramp to Max Time = 07:00am |
| RAMN=1080, | Begin Audio A Ramp to Min Time = 06:00pm |
| RBMX=0420, | Begin Audio B Ramp to Max Time = 07:00am |
| RBMN=1080, | Begin Audio B Ramp to Min Time = 06:00pm |
| RINT=98, | Masking Ramp Interval = 98 minutes |
| RAIT=60, | Audio A Ramp Interval = 60 minutes |
| RBIT=45 | Audio B Ramp Interval = 45 minutes |
| EMIT=0 | Emitter Type = Standard emitter |
| SPEC=0 | Spectrum Type = Standard Spectrum |
| WEEK=011, | Friday Schedule = Min/Max |
| | Saturday Schedule = Min |
| | Sunday Schedule = Min |
| ZNAM=SOUTH POD}<CR>” | Zone Name = SOUTH POD |

## 7.3 Read All Masking Equalizers

| | |
|---|---|
| **Opcode** | "MEQU" |
| **Description** | Read all Masking Equalizer values from a specific zone. The "ALMEQU" Read Masking Equalizers command takes as its seventh byte the Zone Number ID, which is '0' – '5' for the QT-600 or '0' – '2' for the QT-300. For example, the Read Command "ALMEQU4" requests a dump of all Zone 5 Masking Equalizer values. |
| **Command Packet** | "ALMEQUx<CR>" |
| **Data in Response Message** | There are 15 Masking Equalizer values per zone that can be read all at once by the server via the "ALMEQU" Command. Therefore, 15 comma-separated ascii string subfields will be included in the response message. Each subfield is assigned a prefix ID value, defined in the following table: |

| Masking Equalizer | ID |
|---|---|
| 200 | 'a' |
| 250 | 'b' |
| 315 | 'c' |
| 400 | 'd' |
| 500 | 'e' |
| 630 | 'f' |
| 800 | 'g' |
| 1000 | 'h' |
| 1250 | 'i' |
| 1600 | 'j' |
| 2000 | 'k' |
| 2500 | 'l' |
| 3150 | 'm' |
| 4000 | 'n' |
| 5000 | 'o' |

An example response message would look like this

| | |
|---|---|
| | Read variables from Zone 5. |
| "{ALMEQU4=a=11, | Masking Equalizer 200 = 11 |
| b=10, | Masking Equalizer 250 = 10 |
| c=9, | Masking Equalizer 315 = 9 |
| d=8, | Masking Equalizer 400 = 8 |
| e=7, | Masking Equalizer 500 = 7 |
| f=6, | Masking Equalizer 630 = 6 |
| g=5, | Masking Equalizer 800 = 5 |

| | |
|---|---|
| h=4, | Masking Equalizer 1000 = 4 |
| i=3, | Masking Equalizer 1250 = 3 |
| j=2, | Masking Equalizer 1600 = 2 |
| k=1, | Masking Equalizer 2000 = 1 |
| l=0, | Masking Equalizer 2500 = 0 |
| m=-1, | Masking Equalizer 3150 = -1 |
| n=-2, | Masking Equalizer 4000 = -2 |
| o=-3}<CR> | Masking Equalizer 5000 = -3 |

## 7.4 Read All Input A and B Equalizers

<table>
<tr><td>Opcode</td><td>“INAB”</td></tr>
<tr><td>Description</td><td>The PC application reads all Input A and Input B equalizers from a specific zone. The “ALINAB” Read Input X Equalizers command takes as its sevent byte the Zone Number ID, which is ‘0’ – ‘5’ for the QT-600 or ‘0’ – ‘2’ for the QT-300. For example, the Read Command “ALINAB4” requests a dump of all Zone 5 Input X Equalizer values.</td></tr>
<tr><td>Command Packet</td><td>“ALINABx<CR>”</td></tr>
<tr><td>Data in Response Message</td><td>There are 12 Input X Equalizer values per zone, six for Input A and six for Input B, that can be read all at once by the server via the “ALINAB” Command. Therefore, 12 comma-separated ascii string subfields will be included in the response message. Each subfield is assigned a prefix ID value, defined in the following table:
<table>
<tr><th>Input A Equalizer</th><th>ID</th></tr>
<tr><td>250</td><td>‘a’</td></tr>
<tr><td>500</td><td>‘b’</td></tr>
<tr><td>1000</td><td>‘c’</td></tr>
<tr><td>2000</td><td>‘d’</td></tr>
<tr><td>4000</td><td>‘e’</td></tr>
<tr><td>8000</td><td>‘f’</td></tr>
<tr><th>Input B Equalizer</th><th>ID</th></tr>
<tr><td>250</td><td>‘g’</td></tr>
<tr><td>500</td><td>‘h’</td></tr>
<tr><td>1000</td><td>‘i’</td></tr>
<tr><td>2000</td><td>‘j’</td></tr>
<tr><td>4000</td><td>‘k’</td></tr>
<tr><td>8000</td><td>‘l’</td></tr>
</table></td></tr>
</table>

An example response message would look like this

Read variables from Zone 5.

| | |
|---|---|
| “{ALINAB4=a=11, | Input A Equalizer 250 = 11 |
| b=10, | Input A Equalizer 500 = 10 |
| c=9, | Input A Equalizer 1000 = 9 |
| d=8, | Input A Equalizer 2000 = 8 |
| e=7, | Input A Equalizer 4000 = 7 |
| f=6, | Input A Equalizer 8000 = 6 |
| g=5, | Input B Equalizer 250 = 5 |
| h=4, | Input B Equalizer 500 = 4 |
| i=3, | Input B Equalizer 1000 = 3 |
| j=2, | Input B Equalizer 2000 = 2 |
| k=1, | Input B Equalizer 4000 = 1 |
| l=0}<CR> | Input B Equalizer 8000 = 0 |

# 8 Special ZY Command Header

If the Server has chosen to modify the (fixed) IP address of the device, this change will not go into effect until the unit has been reset. Therefore, the special ZY Command has been defined to serve this purpose.

In order to guard against the possibility of an accidental reset triggered from the network, a fixed 4-byte parameter string is defined to accompany the ZY Command Header. The value of this string is “XWvU”. Therefore, the entire command packet defined to trigger a device reset shall be:

**“ZYXWvU”**

When the device receives the Reset Command, it will:

1. Respond with “{ACK,ZYXWvU}<CR>”
2. Wait 50ms
3. Trigger a software reset of the unit.

If the unit was previously communicating over a Telnet session, Step #3 will cause such Telnet session to be terminated.