RainShadow Technology LLC

HDMI-CEC to USB Bridge

HDMI-CEC to RS-232 Bridge

Summary

The HDMI-CEC to USB and RS-232 Bridge is designed to interface a Home Theater to a PC using the CEC bus present in HDMI cables. This peripheral represents the PC as a CEC device to the rest of the HDMI connected Home Theater.

This device can *monitor* all the CEC bus traffic amongst the CEC devices and *transmit* CEC frames onto the CEC bus. It can communicate to a PC or other device through a USB port or RS-232 protocol.

Table of Contents

Background

The "Consumer Electronics Control" (CEC) is an evolving industry standard protocol whose purpose is to simplify the operation and control of home A/V products. Recent products implementing the latest CEC specification can now start talking to each other allowing them to operate as a single integrated device. Instead of using a remote to power on the DVD player, a second remote to set the input on the TV, and a third remote to control the volume on the A/V receiver, the vision is that the devices are smart enough to configure signal routing and control commands to allow a single remote to work everything.

Admittedly, this vision hasn't been achieved yet in that the current implementations are still manufacturer dependent. Manufacturer interoperation is not realized as manufacturers are currently branding their own version of the protocol with their custom vendor extension (i.e. Anynet (Samsung), Aquos Link (Sharp), BRAVIA Theatre Sync (Sony), Kuro Link (Pioneer), CE_Link and Regza Link (Toshiba) and others) . Although slowly, they are starting to implement a complete specification that will eventually allow them to work together.

Motivation

Until now, there has not been a way for the PC to jump into the HDMI-CEC party. Although a PC with a DVI connection or HDMI connection can be easily viewed on the TV, it is unable to present itself as CEC device to the rest of the home theater or interact with the other CEC devices.

This device, though, bridges that gap by allowing the PC to monitor and control the system of connected A/V products. The PC can now present itself as a CEC device, monitor the rest of the home theater components, and send out CEC instructions.

The HDMI-CEC Bridge implements the electrical and low-level CEC bus protocols and basic higher level CEC device protocols. Not only does the device act like a CEC device unto itself, it also allows the PC to contribute or take over these high-level protocols and system control by using scripting or other inexpensive and readily available development tools.

Device & Device Drivers

The HDMI-CEC to USB Bridge is implemented using a Microchip PIC18F87J50 microcontroller and USB software stack. It samples the CEC bus every .1ms, reconstructs any data present on the bus, and forwards the data (CEC frames) to the PC. It is also capable of transmitting CEC frames onto the CEC bus that have been passed from the PC. The PC to CEC communication is done using existing OS com port (tty) communication facilities and existing OS resident drivers.

In addition to presenting itself to the PC as a com port, the composite USB device can also be configured to present the PC as a CEC component to the Home Theater system. When this feature is enabled, the device implements some basic higher level CEC protocols including:

- determining the CEC device's logical address by polling other devices
- responding to queries regarding power state, physical address, CEC version, and Vendor ID
- presenting a programmable On Screen Display (OSD) device name to the TV

Because Windows uses the device's VID & PID (Vendor ID and Product ID) to identify a driver for the plugged in device, a Window ".inf" file is required to associate this device with the existing Windows serial USB driver ("usbser"). This ".inf" file can be downloaded from the website (http://www.rainshadowtech.com). Linux, on the other hand, will correctly detect the CDC device and engage the proper drivers (requires "kernel-module-usbserial" and "kernel-module-cdc-acm" kernel packages to be loaded). OS-X was demonstrated to work with this device by using the open source PL2023 driver available from Sourceforge (http://sourceforge.net/projects/osx-pl2303/)

The device has one red LED on the circuit board. Typically the LED will flash on/off once a second. If the device detects activity on the CEC bus (a valid start bit is detected), the LED will stop flashing and stay on for over 1 second.

Logical and Physical Addresses

The device maintains a CEC logical address between 0 and 0xF. This will allow it to correctly acknowledge commands that are sent to it.

At power-up, the device will automatically look for an unused logical address that is consistent with its purpose. Per the CEC spec, logical addresses 4, 8, and 11 are reserved for playback devices. In its default configuration this device will present itself as a playback device. As a result, it will consecutively poll those 3 addresses. The first address polled that does not have a component acknowledging the poll will become this device's address. If all 3 addresses are in use, this device will remain at address 0xF. NOTE: this automatic logical address determination can be disabled with the 'C' command.

The command 'A' will allow the PC to override the device's automatically acquired address and set it to anything between 0 and 0xF. The overridden address is not maintained in flash and is not persistent between power cycles. The next time the device power's up, it will poll for the preferred addresses described above (or be left at 0xF if automatic logical address determination is disabled).

An extension to the 'A' command can further override the logical address and allow the device to be configured to acknowledge multiple addresses. For instance, the device could be set up to acknowledge logical addresses 2 and 3 with a bit-field setting of 0x000C ('C' being bits 2 and 3 of the address bit field).

The device must also maintain a physical address that indicates *where the PC is physically connected* into the system; the device's physical address *should not be* specified by where *this device* is connected into the Home Theater system. This physical address is used all other connected components for automatic source routing.

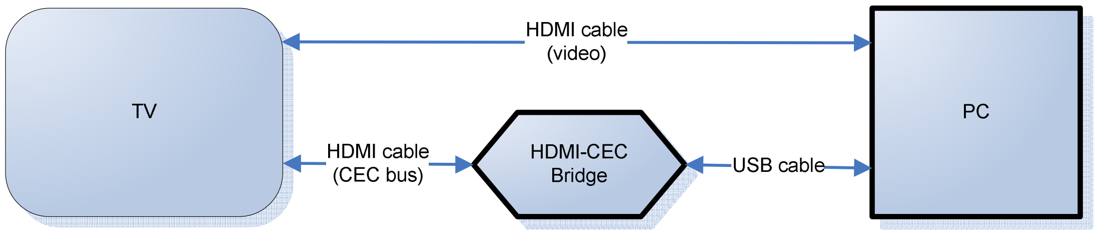
Along with the physical address, the device maintains the 'device type' of the device that the bridge is emulating. In the default configuration, the device type is set to 'Playback Device' (4).

A standard CEC device can detect the physical address using the DDC channel of the HDMI. Since this device does not monitor this bus (and my not even be plugged into the same HDMI port as the PC), the physical address must be manually set once (the value is maintained in flash).

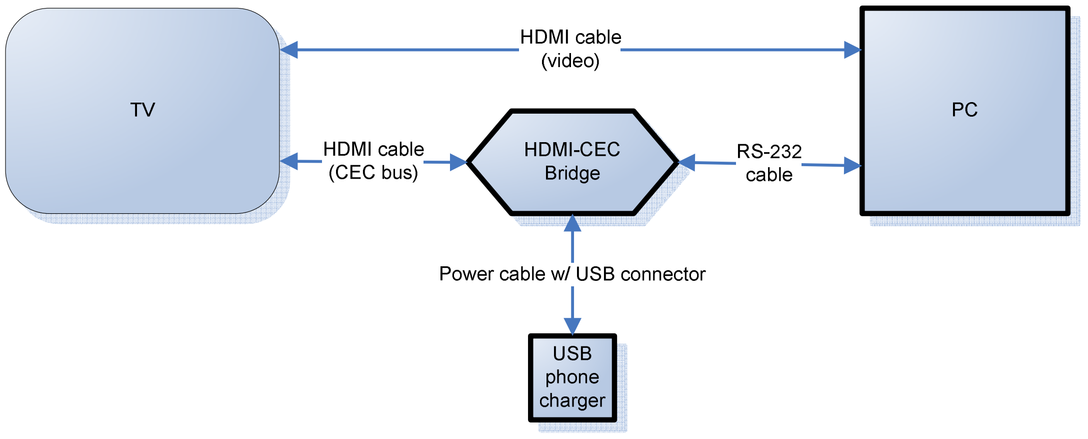
This physical address implicitly determines the CEC network topology. 0.0.0.0 is the root device of the TV. The address 2.0.0.0 is most likely a device plugged into the second HDMI port of the TV. The address 3.1.0.0 is most likely a device plugged into the first HDMI port of a switch which is then plugged into the third HDMI port of the TV. The default address of this device is 4.0.0.0 (which assumes that the PC will be plugged into the 4th HDMI port of the TV if the address is not updated).

Connecting

This device could be connected into the HDMI cable between the TV and the PC, but that is not necessary. Because the CEC bus is a single wire bus between all components, the device can be plugged into any HDMI input in the Home Theater system. The advantage of this approach is that the integrity of the cable (and therefore the quality of the signal) between the TV and the PC does not need to compromised by adding this device to it.



Example connection (1) --
connecting the Bridge through a USB cable



Example connection (2) --
connecting the Bridge through a RS-232 cable and using a 'phone charger' with a USB connector as the power supply

On both Windows and Linux, the HDMI-CEC to USB bridge will show up as a serial port after plugging in the USB port. If Windows asks for a driver disk, point it at the ".inf" file described above to instruct the OS to associate the device with the existing windows driver ("usbser"). Though not officially supported, this device has been shown to work with OS-X.

To manually identify the Windows com port, the device manager can be used. Look under "Ports (COM & LPT)". For programmatic identification, the PID (0xFF59) and VID (0x04D8) will be useful. To monitor the CEC bus, Windows HyperTerminal program is sufficient. HyperTerimal settings that work include 9600 bps, 8 data bits, no parity, 1 stop bit, and no flow control. On the settings tab select "TTY" emulation (Telnet terminal ID TELETYPE-33), the option "Send line ends with line feeds", and the option "Echo typed characters locally".

To manually identify the Linux port, the command "ls /dev/ttyACM*" will often list the specific tty port being used. If this is ambiguous, the command "lsusb –v" (using administrative privileges) will give additional information. To monitor the CEC bus, it is as easy as:

```
cat </dev/ttyACM0
```

Similarly, to send CEC frames from a different shell:

```
cat > /dev/ttyACM0
```

Using the RS-232 protocol instead of USB

The RS-232 serial interface uses a Phoenix Contact 3-pin terminal block (manufacturer part #1840379). The left pin is 'GND', the center pin is the device's 'RX', and the right pin is the device's 'TX'.

All commands and syntax are identical to the USB com-port interface.

The simplest way to power the device when in this configuration is to use a 'wall-wart' charger that has a USB connector.

Commands and Responses (summary)

The Device listens for several plain text single characters commands. A summary of the commands are as follows:

- x : transmit a CEC frame on the bus
- a : display/update the device's current logical address and address 'bit-field' (also see 'b')
- o : display/update the device's OSD (on screen display) name
- p : display/update the device's physical address and 'device type'
- q : display/update the device's retry count
- r : report the device firmware revision level
- c : display/update the configuration bits
- m : mirror a text string back to the host

The HDMI-CEC to USB Bridge will buffer all incoming characters until it receives an EOF (end-of-frame) marker. The tilde "`~`", carriage return "`\r`", and line feed "`\n`" are all interpreted as EOF characters. Once it receives the EOF, the device assumes it has received a complete command and attempts to act on it.

The "!" character is used to throw out (abandon) any characters previously buffered that have not yet been acted on (no EOF marker received yet). It is generally recommended that all commands sent to the device are preceded with an "!".

The device will generate several message types, each with a distinctive prefix. A summary of the message types are as follows:

- ?REC : indicates a CEC frame read from the CEC bus
- ?STA : indicates the status of the last CEC frame that was attempted to be written to bus
- ?ADR : indicates the logical address of the device
- ?BDR: like 'ADR', this also indicates the logical address of the device
- ?PHY : indicates the physical address of the device
- ?QTY : indicates the retry count
- ?REV : indicates the firmware revision level
- ?OSD : indicates the current OSD name of the device
- ?CFG: indicates the current configuration bits of the device
- ?MIR : indicates that a message is simply being mirrored back from the device to the host

Makeup of a CEC frame

This document does not go into detail of the HDMI-CEC specification. For more information, visit www.hdmi.org

Superficially, a CEC frame on the bus is formatted in the following manner:

- a CEC frame (message) consists of a string of 10-bit words
- the start of a CEC frame is indicated with a start bit
- each 10-bit word is 8-bits of data, an acknowledge bit, and an EOM (end of message bit)
- the last word of a frame will have the EOM bit set

The HDMI-CEC to USB bridge implements the low-level protocol by watching for and measuring the start bit, constructing/assembling the 10-bit words from 8-bit data, setting/monitoring the acknowledge bit appropriately, and (when sending) doing retries if directly-addressed messages are not acknowledged properly.

Specification discrepancies and device shortcomings

The prototype has shown to be very reliable in monitoring and sending information on a CEC bus. The current implementation, though, is not true to the specification in every aspect. Known discrepancies include:

1. The device has no way of determining its physical address.
2. If the device detects incorrect bit timing, it will simply stop listening to the message and not acknowledge the remainder of the frame. It will not perform CEC Line Error Handling as specified in CEC 7.4.

Device change history

V1 – original version

V2 – added isolating diode, disabled acknowledgement when address is 0xF

V3 – added mouse functionality, upgraded protocol, logical address detection, and CEC auto-response at firmware level

V4 – added configuration bits to enable/disable automatic logical address determination and mouse capabilities. Enabled the RS-232 protocol (TTL level signals).

V5 – removed mouse functionality

V6 – put on new dedicated board, changed LED logic to use only 1 LED.

V7 -- a) relaxed measured bit timing constraints during read, b) corrected 10ms offset when the bridge is reading the data it is writing, c) added the ability to set the 'device type' with the physical address, d) added the ability for the device to acknowledge communication to more than a single address (spoof multiple addresses simultaneously), e) corrected a '!~' anomaly that would repeat prior command, f) added the 'R' command to report firmware revision

-- correction to this guide regarding the 'C' command

V8 -- added the ability for the device to enter 'suspend' (note: it does not go into low power mode). Once in suspend, the device will be configured to wake the host when it detects *any* CEC traffic. Also added the command to adjust the 'retry count' – that is, the number of times the device will automatically try sending a command if it doesn't receive an acknowledgement.

V2.0- added the 'b' command to set the logical address and have it stored in flash memory

-- correction to this guide, replaced "logical address" with "physical address" on page 15

Response '?REC'

Every CEC frame that the device detects will be forwarded from the device to the PC, whether or not the message was directed at or broadcast to the device's logical address. In this way, the PC can monitor all CEC activity.

At a layer above the low-level CEC frame, the HDMI-CEC bridge reports a CEC frame back to the host in the following format:

```
?REC hh hh hh ..... s\r\n
```

- where "`?REC`" is the lead-in text to indicate start-of-frame
- where "`hh`" is 2 hexadecimal digits representing 8 bits of data on the CEC bus
- where "s" is a status code (one hex digit) that indicates end-of-frame
  - `s=1` – frame was complete and acknowledged by some CEC component
  - `s=2` – frame was complete but there was not an acknowledgement by any component
  - `s='other'` – there was a bit error when reading the frame, therefore the frame is not complete
- where "`\r\n`" is a "carriage return – line feed" character combination

Examples of frames received by the HDMI-CEC device (logical address 4) from a Samsung television (logical address 0) include:

`?REC 0F 84 00 00 00 1`	tv broadcasts its physical address
`?REC 0F 85 1`	tv broadcasts a request asking for who is the active source
`?REC 0F 82 00 00 1`	tv broadcasts that it is the active source
`?REC 0F 87 00 00 F0 1`	tv responds to request for Vendor ID with 0000F0 (Samsung)
`?REC 04 9F 1`	tv asks device for its vendor ID
`?REC 04 83 1`	tv asks device for phys address
`?REC 04 46 1`	tv asks device for its OSD (on screen display) name
`?REC 03 2`	tv probes logical address 3 to see if it exists

All broadcast messages listed above are terminated with a 1. This indicates that no device rejected the broadcast.

All messages sent to the device at logical address 4 are terminated with a 1. This indicates that the device at logical address 4 acknowledged the receipt of the message.

The message sent to device 3 is terminated with a 2. This indicates that there was not a device on the bus that acknowledged the frame.

Command ‘A’, Response ‘?ADR ’

<u>‘A’ : Command to display or update the device’s logical address and ‘address bit-field’.</u>

- !a~ // ask the device to report its current logical address
- !A E~ // set the device’s logical address to 0xE
- !a E 0080~ // set the device’s logical address to 0xE while also instructing it to acknowledge messages sent to logical address 0x7 (0x0080 is the 7th bit of the 15 bit bit-field)

The device maintains a CEC logical address between 0 and 0xF. This will allow it to correctly acknowledge commands that are sent to it.

The following four digits after the logical address create a 15 bit field. The device will acknowledge a message to any logical address that has its corresponding bit set in the bit-field. For instance, a bit-field of 0x0001 will instruct the device to acknowledge any messages sent to logical address 0 (in addition to acknowledging any messages sent to the bridge’s current logical address). A bit-field value of 0x7FFE will instruct the device to acknowledge messages sent from address 1 to 15. The default value of the bit-field is ‘0’ – as a result, the device will only acknowledge its current logical address.

At power-up, the device will automatically look for an unused logical address that is consistent with its purpose. Per the CEC spec, logical addresses 4, 8, and 11 are reserved for playback devices. Because (in its default configuration) this device will present itself as a playback device, it will consecutively poll those 3 addresses. The first address polled that does not have a component acknowledging the poll will become this device’s address. If all 3 addresses are in use, this device will remain at address 0xF.

The logical address and address bit-field is not persistent between power cycles.

<u>‘?ADR ’ : Response to the ‘A’ command</u>

- ?ADR 4 0000 // the device reports its logical address to be 0x4, and a bit-field of 0x0000 (indicating that ONLY commands to logical address 4 will be acknowledged)
- ?ADR E 0006 // the device reports its logical address to be 0xF, and an address bit-field of 0x0006. ‘6’ indicates bits ‘1’ and ‘2’ together, therefore the device will acknowledge logical addresses ‘1’, ‘2’, and ‘E’.

Command ‘B’, Response ‘?BDR ’

‘B’ : Almost identical to the ‘A’ command. This time, though, the logical address gets committed to flash memory.

- !b~ // ask the device to report its current logical address
- !B E~ // set the device’s logical address to 0xE

Please refer to ‘A’ for a general description of the logical address and this command.

The difference between the ‘B’ command and the ‘A’ command is that when using ‘B’, the logical address will get committed to flash.

The logical address committed to flash memory will be used when the device is powered on. If, and only if the device is not configured to poll for its address, the address committed to flash will be used as the device’s logical address. If the device is configured to poll for its address, this value will not be used and the device will instead poll for its logical address as described in the ‘A’ command.

The default logical address set in Flash is 0xF. Unless the device is being used in a static or ‘closed’ environment, it is not recommend that this default logical address is changed. This is because when the device is turned on, (per spec) it cannot assume that the logical address it had before is still available. Thus, the device should initially appear to the bus as address of 0xF and then poll for an available logical address, choosing ad address that is not in use.

If all the devices are statically configured, using a static logical address will provide a power-up time optimization because polling and device address determination can be skipped.

Last point to consider – if the device being used to spoof multiple devices, and as a result has to continually change its logical address, do not use this command. Each time the command is used, the new value is committed to flash generating unnecessary flash writes and wearing it down. Secondly, a flash write suspends the microcontroller; during the time of the flash write the device will not be processing CEC commands.

Command ‘C’, Response ‘?CFG ’

<u>‘C’ : Command to display or update the device’s configuration bits.</u>

- !c~ // ask the device to report its current configuration
- !c 0003~ // set the device’s 32-bit configuration word to 0x3 to enable higher level // functionality and automatic logical address determination
- !C0000~ // disable both higher level functionality and automatic logical address //determination
- !c0002~ // enable automatic address determination but not higher level functionality
- !c000F~ // enable all functionality including the ability to wake the host from sleep

Two configuration bits are currently used by this device.

Bit 0, when set, will enable the higher level protocols. When cleared these operations are disabled and the device will not report its OSD name or automatically or respond to any other CEC commands sent from other devices. This mode allows the Host PC to be entirely responsible for talking on the CEC bus.

Bit 1, when set, will enable automatic logical address determination. When cleared, the logical address determination is disabled and the logical address will remain at ‘0xF’ until manually changed with the ‘A’ command.

Bit 2, when set, will attempt to wake the host up from sleep under the following conditions:
-- Bit 0 is set (higher level protocols enabled)
-- The device has been put into suspend mode by the host
-- The device has been armed by the host to allow it to wake the host
-- The device receives traffic on the CEC bus addressed to its primary logical address

Bit 3, when set, will modify bit-2’s behavior by ignoring whether or not the host has armed the device to allow it to wake up the host. This is a windows hack that may allow the device to wake up the host even though the native windows serial com port emulator driver does not support wakeup functionality. This hack has been demonstrated to work when another USB device that is allowed to wake up host happens to be on the same USB circuit/hub.

The 32-bit configuration word is persistent between power cycles.

<u>‘?CFG’ : Response to the ‘C’ command</u>

- ?CFG 0003 // the device has logical address determination and higher level functionality // enabled
- ?CFG 0000 // the device has logical address determination and higher level functionality // disabled

When higher level functionality is enabled, the device will automatically respond to the following commands:

- 0x1A, deck status
- 0x46, report OSD name
- 0x83, report physical address (and device type)
- 0x8C, report vendor id (answer is 'not supported')
- 0x8D, respond to 'menu' request saying menu is active. This allows some commands to be forwarded from the remote control to the device.
- 0x8F, report power status
- 0x9F, report CEC version

When 'host wakeup' is enabled in addition to higher level functionality, the device will attempt to wake the host up from sleep when receiving any command that is sent to its primary logical address.

Command ‘O’, Response ‘?OSD ’

‘O’ : Command to display or update the device’s on screen display name

- !O~ // ask the device to report its current on screen display name
- !OMyDevice~ // set the device’s on screen display name to “MyDevice”

The device maintains a 14 character string as its OSD name. This string will be returned to any component (such as the TV) that asks this device for its display name.

This command can update the display string. Because it is stored in flash, the updated string is persistent between power cycles.

‘?OSD ’ : Response to the ‘O’ command

- ?OSD MyDevice // the device reports its logical address to be “MyDevice”

Command ‘P’, Response ‘?PHY ’

‘P’ : Command to display or update the device’s current physical address

- !P~ // ask the device to report its current physical address
- !P 3000~ // set the device’s physical address to 3.0.0.0
- !p 2000 5~ // set the device’s physical address to 2.0.0.0 and its device type to ‘5’ (audio system)

The device must also maintain a physical address that indicates *where the PC is physically connected* into the system; the device’s physical address *should not be* specified by where this device is connected into the Home Theater system. This physical address is used by all other connected components for automatic source routing.

A standard CEC device can detect the physical address using the DDC channel of the HDMI. Since this device does not monitor this bus (and may not even be plugged into the same HDMI port as the PC), the physical address must be manually set once (the value is maintained in flash) to the address of where the PC is plugged in.

This physical address implicitly determines the CEC network topology. 0.0.0.0 is the root device of the TV. The address 2.0.0.0 is most likely a device plugged into the second HDMI port of the TV. The address 3.1.0.0 is most likely a device plugged into the first HDMI port of a switch which is then plugged into the third HDMI port of the TV. The default address of this device is 4.0.0.0 (which assumes that the PC will be plugged into the 4th HDMI port of the TV if the address is not updated).

The physical address is stored in flash and therefore any updates are persistent between power cycles.

‘?PHY ’ : Response to the ‘P’ command

- ?PHY 3000 4 // the device reports its physical address to be 3.0.0.0 and its device type to be 4 (Playback device)

Command ‘Q’, Response ‘?QTY ’

<u>‘Q’ : Command to display or update the command retry count</u>

- !Q~ // ask the device to report its current retry count
- !Q 2~ // configure the device to attempt sending command a maximum of 2 times
- !Q E~ // configure the device to attempt sending a command a maximum of 14 times

The default ‘retry’ count per the HDMI specification is ‘5’. That is, if a target device does not respond to a message, or if an error is detected during the send of a message, the bridge will automatically try to send the message again (each time backing off on the retry timing).

The ‘Q’ command can be used to determine how many times the bridge will attempt to send the same message in the case it is not successful.

The ‘Q’ command can also be used to modify this count. Valid numbers are between ‘1’ and ‘F’ (1-15). Attempting to set the value to ‘0’ will result in the default value of ‘5’ being used.

This value is not stored in persistent storage and will be reset back to the default value of ‘5’ at each power cycle.

<u>‘?QTY’ : Response to the ‘Q’ command</u>

- ?QTY 5 //the device reports that it will attempt to send a message up to 5 times

Command ‘R’, Response ‘?REV ’

‘R’ : Command to display or update the device firmware revision

- !R~ // ask the device to report its current firmware revision level

The device will return its current revision level.

If the revision level is suffixed with “_HTPC”, the firmware is compatible with the HTPC windows software.

‘?REV’ : Response to the ‘R’ command

- ?REV 1.8_HTPC //the device reports its firmware level to be “1.8”. A suffix of “_HTPC” on the revision number indicates compatibility with the HTPC windows software.

Command ‘X’, Response ‘?STA ’

<u>‘X’ : Command to send a CEC frame onto the CEC bus</u>

- !x5~ // send an empty message from device’s logical address to device at 0x5
- !x0 46 ~ // ask address ‘0’ (TV) for its on screen display name
- !x0 04 ~ // send “image view on” to TV to turn it on
- !xF 84 4000004~// broadcast a physical address of 4.0.0.0 and device type 4

The device will place its logical address at the very beginning of the CEC frame to be sent (the source). The very first digit after the ‘x’ command becomes the destination address. The remaining hex digits make up the remainder of the CEC command to be sent.

Since a frame consists of 8-bit bytes on the bus, the device expects an even number of hexadecimal digits to precede an EOF marker. If an odd-number is encountered, the device will assume the most significant nibble of the last byte is 0. Thus, a message of “`x23~`” would be interpreted to mean “`x203~`”.

Any frames that are sent by the device to the bus will simultaneously be read by the receiving portion of the device. Because of automatic retries, a directly addressed message that isn’t acknowledged by the receiving device may appear (be read) as 5 consecutive unacknowledged messages.

<u>‘?STA’ : Response to the ‘X’ command</u>

After the device completes its attempt to send a message, it will report the sending status back to the host. A sending status has the following format:

```
?STA s\r\n
```

- where “`?STA`” is the lead-in text to indicate a status message for the last frame sent
- where “s” is a status code (one hex digit) that indicates final sending status
  - `s=1` – frame was complete and acknowledged by a destination device correctly
  - `s=2` – frame was complete but there was not an acknowledgement by any device
  - `s='other'` – there was a bit error when sending the frame, therefore the frame is not complete
- where “`\r\n`” is a “carriage return – line feed” character combination