

RS-232/USB to DMX512 Converter/Controller
Model 4201a
Firmware Revision 1.4x
PC Board Revision 1.0
Instruction Manual

Durand Interstellar, Inc.
219 Oak Wood Way
Los Gatos, California 95032-2523
www.interstellar.com
tel: +1 408 356-3886, fax: +1 408 356-4659

Intended Use

This product will allow computers of almost any make and speed to control lighting in situations where a full lighting board is impractical. While this product does support all 512 channels per USITT DMX512/1990 (8 μ S MARK AFTER BREAK), the number of level changes sent by a host computer per second is limited by the speed of the RS-232 or USB port and the number of channels in use. A slow computer ('286 laptop) running MS-DOS is able to perform smooth fades on 24 channels, so you're not as limited as you might think. For most intended applications (trade shows, hotel events, store windows, museums, amusement park displays, etc.) and when using the USB port and/or the Fade command, this limitation will never show up. Using the Fade command, you can do smooth fades on all 512 channels no matter how slowly you enter the command. You may store one static level for each of the 512 possible lighting channels in each of the 128 internal presets (current active and 127 non-volatile). The non-volatile presets (NV-RAM) will be held until you change them, even with the power off. You may choose to have one of the NV-RAM presets sent out to the lighting dimmers automatically on power-up. There is also Script support, where you save a list of commands in NV-RAM and can execute them from a host command and/or automatically on power up. You may even connect this converter to an external modem (set to auto-answer) and control it over the phone. Wireless modems work well, also.

Please note that the Windows demo program and DMXC demo are intended as examples and do not have complete error checking. They have not been tested on every combination of computer and version of Windows and so they may crash unexpectedly.

Since the DMX512 protocol itself has no error checking, you should never use it to control anything that might be damaged or cause injury if the wrong data is received.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation. As defined by FCC rules (§15.3h), this is a "Class A digital device". A digital device that is marketed for use in a commercial, industrial or business environment, exclusive of a device which is marketed for use by the general public or is intended to be used in the home.

There are no user serviceable parts inside besides the jumpers and digital input connections. Be sure to always re-attach the cover securely and never apply power with the cover removed.

The latest version of this manual and firmware are available from our support site:
<http://www.interstellar.com/support.html>

All of our documentation is in Adobe Acrobat format. You can download a free reader for PCs and Macs at:

<http://www.adobe.com>

All trademarks are the property of their owners and are used for reference only.

Basic Hardware Setup

If you are using lighting control software supplied by one of our partners, please refer to their manual for setup and use.

First, you will need a computer or PLC that has an available serial port (COM Port) that supports RS-232 (see USB section if you wish to use the USB port instead of the serial port). You will also need a cable to connect your computer to the converter. If you have a PC, a simple 9-pin “straight-through” male to female cable should work. If you have a Mac or other computer, see your local computer dealer for any adaptors or special cables that you might need (the computer and cable MUST support RTS/CTS handshaking). Connect the serial cable to the 9-pin female connector on the end of the converter.

When the USB port is connected to a host, the converter will ignore the serial port (commands can then ONLY come from the USB port). It will also draw power from the computer if the AC adapter is not attached or plugged in. To use the serial port again, unplug the USB and AC adapter cables, and then only plug in the AC adapter and serial cables.

The following setup assumes you’re using the RS-232 connection.

The converters are shipped set for 9600 baud, if you wish to use a different baud rate, see the Advanced Hardware section of this manual.

Now, connect the output of the converter (5 pin XLR connector on end of converter) to your dimmers (they must support the DMX512 protocol, see your dimmer instructions for connection and setup information). If possible, set your dimmers to start on the first channel (1 or 0, depending on the dimmer). In any case, note the channel numbers of the dimmers you wish to control and whether they consider 1 or 0 the first channel.

Attach some lights to the dimmers and disconnect anything that may be damaged or cause injury if the dimmer receives bad control information.

If not already on, turn on the power to the dimmers, the computer, and plug in the converter using the supplied AC adapter (use of an improper adaptor or modifications to the supplied adaptor may void the warranty and may cause damage and/or injury, see the Power Notes below). The red POWER LED on the converter should come on.

If the converter was set to auto-playback mode (factory default), the green OUT LED will light (it may appear to be constantly on or blink) and your lights will be set to whatever levels were stored in the NV-RAM (the levels will probably be 0% on a new converter). If auto-playback was not selected, no DMX data will be sent and the lights attached to the dimmers should remain off. If script #0 is valid, it will be executed as soon as the unit powers up.

The factory default for the number of channels is 64 (40 hex), but you may change this using the “N” command.

Windows Serial Demo Software

The DEMO.EXE program was written for our original converter. It works fine with the new models using COM ports 1-4. For a more advanced, USB-based demo program, see the Two Scene Preset demo below. You can also check here for the latest software:

<http://www.interstellar.com/support.html>

Note: The old Windows demo program only controls lights on the first four channels. If you do not have any dimmers or lights on these channels you will not be able to use the demo program as supplied. This program is intended to demonstrate one way of using this converter, it is not intended to be used in actual applications.

The converter is shipped with the baud rate set to 9600, if you have changed it, you must return it to 9600 to use the demo program.

If you have an auto-execute script (#0) active, the demo program will not be able to control the converter until that script stops executing.

If you have Windows 95, 98, NT, or 2000, or XP on your computer, run the “demo.exe” program found on the enclosed disk. If the converter is connected to COM:1, you do not need to change the port settings. If it is connected to another port, click on the ComPort button and select the correct port (see your computer manual if you are unsure of which port you are using). The other settings should always be “9600”, “8”, “1”, “none”, and “RTS/CTS”. Select OK to return to the main window.

If the green OUT light is on or blinking, click on the button labeled “Playback Mode 0”, this will instruct the converter to stop sending data from the active memory (Preset 0). You may see the yellow IN light flash momentarily and then the green light should stay out. A short time later, your dimmers should turn off any attached lights (and possibly indicate a DMX no-data error). Now, press the button labeled “Slider Data in Loop” and the yellow IN light should be on (you may notice a flicker, this is fine), this means the computer is sending commands to the converter. The green OUT light should also be on or flickering, this indicates that data is being sent to the dimmers.

The 4 sliders are now controlling dimmer channels 1-4 (or 0-3 in some dimmers). Use your mouse to drag the sliders to new positions and note that you are controlling the lights. Also, any error indication on your dimmers should now be off. If you have changed the number of channels from the default, you may notice a delay between changing a level in the demo program and the lights actually changing. This is a result of the “D” command being sent out without waiting for the prompt. See the “D” command for more information. If you set “N” to a small number (less than about 170, 4 would be good since that’s all the channels the demo supports anyway) you will not see any delay.

Select the button labeled “None” to stop sending data to the dimmers. The yellow and green LEDs will go out and your dimmers will detect the loss of data and should shut off all attached lights.

Click on the button labeled “Playback Mode 1”, the yellow LED will blink once and the green LED will light. The converter is now sending 64 (or whatever “N” is set to) channels of levels from NV-RAM Preset 1. Your dimmers should clear any error indication and set all attached lights to whatever levels were stored in the NV-RAM.

Set the sliders to four different positions and then click the button labeled “Slider Data To Memory”. The yellow LED will blink once and the level of the sliders will be stored in the first four channels of the active memory (channels 5-512 will not be changed by this demo program). The new levels will be sent out to the dimmers immediately (no delay as there was with the loop mode, this mode is using “W”). Try setting the sliders to various positions and clicking the “Slider Data To Memory” button to see how you can change levels in the active memory. These levels will be maintained until changed, but will be lost when the power is turned off. The demo program does not write to the NV-RAM presets.

If you click on “Playback Mode 1” again, the levels from NV-RAM Preset 1 will be copied back to the active memory, overwriting any changes that you made.

You may leave the program by pressing the “Exit” button.

Two Scene Preset Demo

This program requires that the converter be connected to Windows by the USB port. If you've installed the HIDCOM driver, you'll need to uninstall it before this program will work (see USB Notes below for instructions).

This program is a fully functional lighting control program that emulates a 12 channel two scene preset board. This may be all you need for your application. More sophisticated control software called SmartLighttm is available from Edgeview Software, see our web page for links to try out and purchase this software. Mac software will be available soon.

First, plug the converter into your Windows computer using the USB cable only. You may optionally use the AC adapter, but it is not needed in most applications. The first time you plug the converter in, Windows will report finding new hardware. It should NOT ask you for any drivers. Once it is done installing the converter, there will be no further notices from Windows.

Next, run the Two Scene Demo program found on the CD. After it opens, make sure it does not say "DMX Converter not found!" at the top right. If this message appears, there was some problem communicating with the converter. Exit the demo, unplug the converter and AC adapter (if used). Reboot the computer and try again. It should work fine now.

Use the demo just like you would a manual lighting board. The MASTER control decides which of the scenes (A or B) you are sending to the lights. During a show you would set up one scene on A and then cross-fade to it using the MASTER control. You would then set up the next scene on B and cross-faded back to it when you are ready. You can do timed cross-fades by entering a time (in seconds) in the Fade Time box and then pressing GO.

The source files are provided for those programmers who would like to customize this or just know how it works.

USB HID Demo Software

We also supply a simple demonstration program called DMXC Demo. This program will communicate with the converter using the USB interface. The program will open a small window and allow you to type commands into the converter similar to using a terminal program. It is mainly intended to show programmers how to communicate with the converter without using the serial port driver, but can also be used to set up, test, or use a converter.

To exit the DMXC Demo program, simply click on the X in the upper right corner of the window.

The DMXC Demo program should work with all versions of Windows that support USB (Windows 98SE and later). It does not need any installation and can be simply deleted if you no longer wish it on your system. You may also run it directly from CD or floppy.

Simply plug in the converter, and wait for Windows to say it's found a new HID (the message only appears the first time you plug it in). Windows should install the device and close the notification window without any input from you. Now you should be able to run the DMXC Demo.

Note that DMXC Demo will not work if you've installed the serial port driver. If you have already installed the serial port driver, see the HID Installation section at the end of the manual for information on removing the driver.

Advanced Hardware

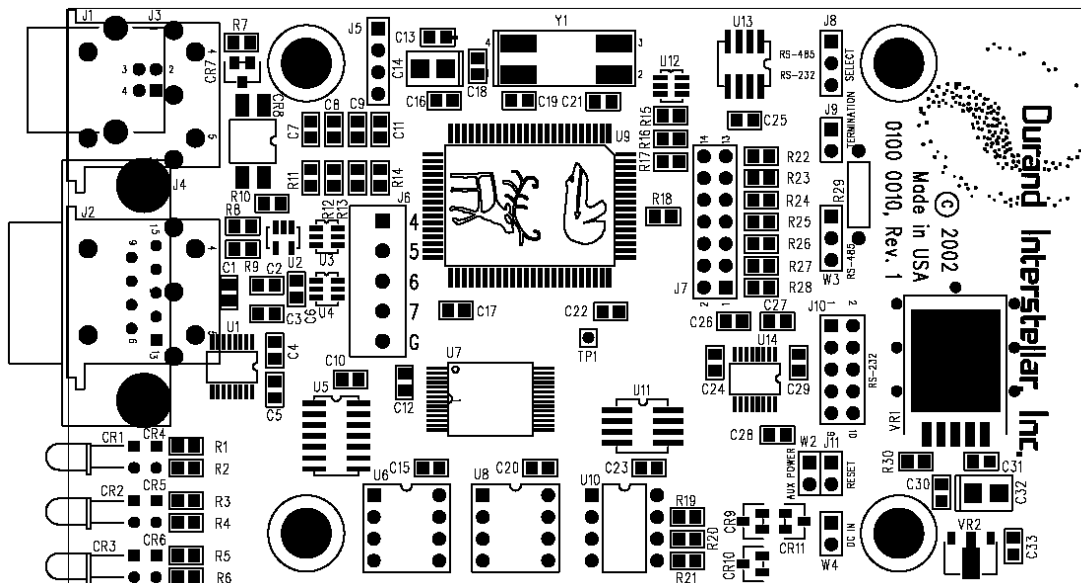
This section only applies to PC Board Revision 1. If you have Revision 0 (shipped before January 2003) you should refer to the 1.0 instruction manual for hardware settings.

There are jumpers and digital (switch) inputs inside the converter. To access these jumpers/connections, unplug the power and USB cables and remove the cover from the converter. You need to remove the 4 screws on the bottom of the converter and then the cover will lift off. Inside you will see a PC board with dual-pin post type jumpers and a screw terminal block (see below). If you remove a shunt you should replace it so that it only covers one post. This way you won't lose it (spares available from any electronics store).

J7 is the large rectangular jumper with 14 pins to the right of the center of the board. The first 6 pins are for baud rate selection and a digital input (see the Jump command), the rest are not used in this model. In the picture below, pin one is to the bottom right of J7, pin 2 is the bottom left, pin 3 is second from the bottom on the right, and pin 4 is second from the bottom on the left. When pins 5 & 6 are shorted, Flag bit 2 will read "1".

Baud rate selection (not applicable to USB use)

- 9600 baud: shunt between pins 1 & 2, shunt between pins 3 & 4.
- 19200 baud: shunt between pins 3 & 4 only.
- 28800 baud: shunt between pins 1 & 2 only.
- 38400 baud: no shunts installed.



J9 is to the right of J7 and is used to enable the DMX termination resistor. This is not needed (but doesn't hurt anything) when this board is the master (transmitting DMX). When the board is a slave or monitoring DMX signals (not supported in this version of firmware), then the last device on the cable **MUST** have a termination resistor. Install the shunt to enable the resistor.

J6 is a screw terminal block to the left of center, it is for the other four digital (or contact closure) inputs. The numbers to the right of the block indicate the bit positions of the inputs in the Flags byte. These are tested with the JUMP instruction in scripts. Shorting any pin to the “G” (ground) pin will cause that input to be tested as a “1” (with a 4-pole rotary switch you can read up to 16 positions). You may also drive these inputs with 3.3V or 5V digital logic (inputs may be driven between ground and 5V max). Each pin has a 2.2K resistor to 3.3V so your output must be able to sink 1.5mA. These inputs have reasonable surge protection on them, but if you are driving them from a long distance or other conditions that could cause damage or ground loops, it would be best to drive them with an optical isolator board or relays placed close to the converter. If you have a need for any specialized input boards (switches, sensors, isolation, “up-down” push buttons, keypad, etc.), please contact us for a quote.

Other jumpers/connectors on the board are reserved for factory test and/or other models. No connections should be made to any of these and spare shunts should not be stored on any reserved jumpers.

Anyone working with DMX512 systems should have the booklet “Recommended Practice for DMX512” available from PLASA Ltd. in the UK and USITT in the USA.

If you’d like to know more about how the DMX512 signal is actually sent to dimmers, you will want to read the specification from USITT titled “DMX512/1990”.

Advanced Software

If the demo program or scripting doesn't do everything you need, you will need to either write your own control software or purchase a package from one of our partners. Of course, for just setting static lighting levels, you can also use any terminal emulator program or the DMXC Demo program and type in the values by hand (slow, but you only have to do it once). You can also enter Scripts using a terminal emulator or DMXC Demo and then set the unit to run them automatically on power up. This would be good for trade shows, museums, store windows, etc. An external device like a PLC can also send Events and external switches can be read to control branching in scripts.

If you'd like to see how the demo programs work, the source files are on the CD and also available at:

<http://www.interstellar.com/support.html>

If you haven't yet, read the booklet "Recommended Practice for DMX512" published by PLASA and USITT.

If using a serial port on the host computer, it **MUST** be set up as:

- The baud rate must match the speed selected by the jumpers in the converter.
- 8 data bits
- 1 stop bit
- no parity
- RTS/CTS handshaking

If you are using the USB connection with the serial port driver (HID-aware software does NOT use the driver), the converter will show up in Windows as another serial port. You will set up your software just like it was talking to the converter over a "real" serial port, only the baud rate setting is ignored (the converter will always transfer data at the maximum USB rate). When the converter detects a USB connection, the RS-232 port is disabled until the next time you cycle the power (also, any script or fade running is aborted).

HyperTerm seems to have problems at times remembering to tell the converter it's ok to reply. If, when you press RETURN in HyperTerm the yellow light on the converter blinks, but nothing prints on the screen, type:

H0

Followed by a RETURN. This tells the converter to ignore the handshaking and you should now receive characters just fine. It is normally safe to leave handshaking turned off in the converter. There is no real need to use handshaking with the USB interface, it is ok to set HANDSHAKING to NONE in HyperTerm as long as you set it to OFF in the converter first.

If you are writing your own software, it is highly recommended that you use the HID interface available through your operating system (Windows, Mac, Linux, etc.). Any system that supports USB should not need any drivers to use the HID interface. See a

book such as “USB Complete, Second Edition” by Jan Axelson for a discussion of this and sample programs in VB and C. Also refer to the source code supplied with our demo programs.

If you aren’t comfortable with hexadecimal numbers, see pages 72-74 of the above book or any introductory computer programming text. All numbers sent to and returned from the DMX converter are in hex. There are also hexadecimal conversion application notes on our web page under Support.

Leading zeros are not required, for the number zero you must send at least one “0”. Numbers returned from the converter may or may not have leading zeros. If a number is received by the converter with more digits than it is looking for, only the last digits are kept (“12345678” is read as “5678” or “78”). Do NOT send the “0x” prefix in front of any number.

All commands and hex numbers may be in upper case, lower case, or a mixture of both.

White space is defined as Horizontal Tab (09 hex), Line Feed (0A hex), Vertical Tab (0B hex), Form Feed (0C hex), and Space (20 hex). Any amount/combination of white space may precede a command or number. White space may also follow the last data on a line and precede the carriage return.

All commands execute as they are received but they MUST be terminated with a carriage return (0D hex)...that’s ‘\r’ for you C programmers, NOT ‘\n’. This makes it much easier to use a terminal emulator. If you want to send “\r\n” or “\n\r”, that’s OK.

Any of these commands (except the “D” command) may be typed in from a terminal emulator program (such as Hyper Terminal in Windows). There are no timeouts on commands (except “D”), you could start a command today and finish it tomorrow and the converter wouldn’t care (unless there was a power failure). Aside from debugging, the main use of manual command entry would be for setting up a Script for applications where you set the levels and time or event-based changes once, enable auto-playback, unplug the computer, and go home.

If your computer allows you to copy text files to the serial port, you could make up a text file containing anything from a single cue to a long series. Using the time and delay commands, you could create fairly complicated lighting effects with nothing more than a text editor (like Notepad in Windows). You could even insert timer commands to lock the effects to the internal clock with 1/10 second resolution. If using the RS-232 connection, handshaking MUST be enabled on the host computer end.

You could also use text files to load Scripts and Presets into memory, that way you could edit the copy on your computer and re-load it whenever you needed to change something. This would also allow you to load the same Scripts and Presets in a number of controllers. You can even cut and paste from many e-mail programs directly to your terminal emulator (HyperTerm on a PC, ZTerm on a Mac).

When you power up the converter, you will get a message that it is starting the Auto-Exec Script. The factory default is for this Script to be empty, so you will get a message to that effect and then the normal prompt will appear.

When you plug in the USB cable and any other time the host computer sends a RESET command over the USB, any timer commands, fades and/or scripts will be aborted.

See below for more information on Scripts.

When using a terminal emulator, do NOT use the backspace key. The converter executes commands as they are entered and can not backspace. If you make a mistake while typing, press RETURN and reenter the entire command again.

Command Descriptions

There aren't that many commands to master and you really only need a couple in most applications. Make sure you have the correct version manual for the firmware in your converter. Use the "I" command and check the version number matches the cover of this manual.

Don't forget the carriage return (0D hex) after every line!

If an error is detected while processing a command, a '!' character is sent to the host. When the converter is ready for a command, it will send a prompt to the host. The prompt consists of a return (0D) and linefeed/newline (0A) followed by the character '>'. In most applications you can safely ignore error checking (you shouldn't ever get any errors), however you **MUST** set up your software to deal with incoming characters or set handshaking to OFF. The demo program discards all replies, if you turn off that section of code everything will be fine for about 10 seconds and then the receive buffer will fill up. When this happens Windows sets RTS to FALSE (the host saying "Stop sending characters"). If handshaking is ON, the converter will then wait for RTS to go back to TRUE before it sends out the next character. Since the converter can't execute the next command, its receive buffer will fill up and it will set CTS to FALSE telling the host to stop sending characters. Both computers will now wait a very long time for something to happen.

Setting handshaking to OFF tells the converter to ignore any errors while sending data to the host. This **MAY** cause lost characters on long replies, such as the "I" and "?" commands. As long as the host doesn't mind lost characters, this will not affect the operation of the converter.

If you don't care about receiving responses from the converter, you can use the "H0" command to turn off the checks for RTS, then save the settings with the "Z" command. The converter will still send the responses, but will ignore RTS and always send the data. If the host doesn't accept data fast enough, the converter will simply discard characters until the host catches up. The converter will always use the CTS line to tell the host to stop, this isn't needed for USB, but for RS-232 connections the host should always be set to obey this.

There is no harm in issuing the next command before the current one has completed. The new command(s) will be buffered and executed as soon as the previous one has completed unless you issue an abort.

To get started, look at the W command below, depending on your application, it may be the only command you need.

Be sure to see the note below on NV-RAM use.

Commands:

C COPY MEMORY

This will copy any preset to any other. This command ALWAYS copies all 512 channels, no matter what “N” is set to. Simply enter the source preset number and the destination preset number. If the destination is in NV-RAM, this command will be somewhat slow as writing to this memory is not fast.

The active memory is considered to be Preset 0. If the Playback Mode is not 0 and you copy from NV-RAM to Preset 0, the new levels will be sent out as they are copied (possibly over several DMX packets). This could be a problem if you are controlling moving lights or other specialty items. It is recommended to use the Playback Mode to copy to active memory as it is optimized for speed and is synchronized with the outgoing DMX packets. Using the copy command does NOT interfere with the DMX output as long as you are writing to NV-RAM.

To copy from active memory to NV-RAM Preset 16 (10 hex), enter:

C 0 10

To copy from Preset 1 to Preset 63, enter:

C 1 3F

D DIRECT TO DMX COMMAND (not recommended for new applications, this command is to maintain compatibility with older versions)

This is the only command that you shouldn't type in at a terminal emulator. The DMX specification has a 1-second timeout for loss of data detection, so unless you type VERY fast, your dimmers will time out and shut down.

The simple version of this command is just a D followed by a string of hex numbers separated by white space. The first number is channel 1, the next is 2, and so on. The converter will hold the data as it's received until you send a return, then it will all be sent out as one packet. If you send more than the number of channels set with “N”, the extra channels will be ignored.

Sample:

D 1 2 3 4 5 6 7 8 9 a b c d e f 10 11 12 13

Since the DMX format doesn't let you skip channels before you get to the one you're interested in, this can get tedious (and slow) if you have a device on channel 500. To use the simple format you'd have to insert 499 zeros plus white space. Enter the Skip subcommand (a form of RLL coding). DMX data is only 8 bits, so each value on the line must be between 00 and FF. If a number is between F00 and FFF it is interpreted as a skip command and will insert a number of zeros in the DMX data stream at high speed relieving both the host computer and the programmer of the task. The number of channels “skipped” is determined by the lower 8 bits (the right two digits) of the 3-

digit number. F01 means “insert one channel of 0”, F02 means insert 2 zeros, etc. with F00 a special case that inserts 256 zeros. So to send the data 55 to channel 500, you could enter:

D F00 FF3 55

Note that we had to use two skip commands in a row, as long as they’re separated by white space, this works fine. Simply repeat the skip command as many times as you need.

You can also intersperse skip commands with data as in:

D F12 55 F22 AA

Please note that skipped channels are set to a level of 0. If you have any lights on those channels, they’ll be dark. If you have 500 channels of lights and want to change any one of them, you **MUST** send out all 500 channels of data every time. If this is too much trouble for you, then see the “W” command.

With the above commands the host has to send a minimum of 2 characters and normally 3 characters per channel (white space + one or two digits). This isn’t slow but it isn’t fast, either. For those of you who want the fastest updates possible, there’s the binary subcommand. It works something like the skip command, only the first digit is ‘B’ and the lower 8 bits are the number of binary channels to follow (“B12 ” means the next 12 hex bytes that come from the host are 8-bit binary levels). Since binary data doesn’t print well, we’ll use ‘x’ in the example to represent the 8-bit binary data.

D B04 xxxx F05 AA 55

In the above example the “B04 ” means that the next 4 characters, whatever they are, are treated as a level with no checks performed. Note there **MUST** be a single space (or other white space) after the “B04” or the command won’t work. The rest of the command string just shows that you can mix these subcommands with normal simple data. **BE CAREFUL**, if one of the ‘x’s was missing, the white space in front of the “F05” would be sent as a level and then the “F05” would generate an error. So, if you had 8 channels in binary mode starting at channel 1, the command would look like:

D B08 xxxxxxxx

A note on the “D” command. As soon as the “D” is received, the Playback Mode is switched to 0. As each channel is received it is stored in active memory. When the Carriage Return at the end of the line is received, one (and only one) DMX packet is sent out and then the DMX output is halted again. The last data sent out will remain in active memory until changed by the “D” or some other command (including a “P” command). You can use the Copy command to place this data in some other Preset, if you wish.

Also, due to buffering delays in the host computer and the converter, if you

are sending out “D” commands rapidly and “N” is set to a large number, you will notice a delay between issuing a “D” command with new levels and the lights actually changing. There are several ways around this, send the “D” commands slower than the packet rate (depends on the setting of “N”), wait for the prompt before sending the next “D” command, use a smaller value of “N”, or use the “W” command.

When the “D” command completes, the DMX output is turned OFF (the same as entering the command “P 0”). If you’re going to issue another “D” command within one second, there’s no problem. If you’re not, use the command “P FF” to turn the output back on so your dimmers don’t time out.

ER READ EVENT

This command returns the current value of the Event data as well as the Flags byte. An example of the data returned may be:

1435

The first digit is the status of the digital inputs. The “1” indicates that digital input #4 is low (switch closed) and the other 3 are high (switches open).

The next digit is the rest of the flag bits. The 4 indicated that Zero, Carry, and Sign are zero and the Jumper (on J8) is installed. If only Zero were set, this would read “1”. If only Carry were set it would read “2”. Sign would read “8”. Any combination of flag bits may be on at one time.

EW EVENT & FLAGS WRITE

This allows another way for the host to update the Event data. The “~” command always works and requires binary data (and does not update the flags). EW only works when a script is not running and works with normal hex numbers. EW 156 will load the number 0x56 into the Event memory while setting the Zero flag to 1 and the Carry and Sign flags to zero. No other flag bits are changed. EW 56 would still load 0x56 into the Event memory and also clear the flags Zero, Carry, and Sign. The flags and Event are cleared to zero on powerup.

F FADE COMMAND WITH MASKING

This command will execute a cross fade from whatever levels are currently being output (in Active Memory) to any other preset (1-127). Masking allows you to only fade selected channels while leaving the others untouched. Fading to preset #0 is a special case for “fade to black”.

The command format is an ‘F’ followed by the preset number to fade to, the time that the fade should take, and optionally the preset to use for a mask (see below). The fade time is given in 0.1 second steps, so 12.3 seconds would be 123 (7B hex). The valid range of time is 0-FFFF (a little over 109 minutes max). If you issue the abort command ‘*’, the fade will be terminated and the levels will jump to the final value.

If you wanted to fade from the current levels to preset #4 over 12.3 seconds, you would issue the command:

F 4 7B

If you only want to fade certain channels, you need to set up a preset as a mask (any preset other than zero will do). In this special preset, any channel that is set to zero will NOT be updated during the fade. For example, if you only want to fade channels 2 and 5 you could setup a mask preset like this:

F0 0

W2 1 0 0 1

C 0 10

In this example the Fade to 0 sets all channels in active memory to zero, and then we write a one (could be any non-zero number) into channels 2 and 5. After that we copy preset #0 (Active Memory) to preset #10 (hex).

This example assumes you are doing this without your lights turned off as part of your setup. If you need to do this while your lights are on, you could write directly to the EEPROM using:

W 2000 0 0 1 0 0 1 0 0 0 0 0

Here 2000 is the starting address of preset #10 in the EEPROM (200 hex times the preset number). Note that in this case you will have to write a zero to every channel that you don't want changed during the fade, there's no quick way to clear an entire preset in EEPROM. At least you should only have to do this once and you only have to clear the channels in use (if you only have 12 dimmers, there's no need to worry about the rest of the channels). If you have a few different channels you wish to mask at different times, simply set up a different mask preset for each case.

Now, let's fade to preset #4 over 12.3 seconds again, only using masking so that only channels 2 & 5 actually change:

F 4 7B 10

The smoothness of the fade is dependent on the number of channels ("N").

With all 512 channels enabled, the lights are only updated about 30 times a second. With 19 or fewer channels the update rate is over 700 times per second!

When the command is finished the Active Memory will contain the final levels and the DMX output will be on, regardless of what the Playback Mode was originally set to.

H HANDSHAKING COMMAND

In order to control the flow of information over a serial connection, handshaking is used. This converter uses Hardware Handshaking, meaning that it will not send any data to the host computer when the RTS line is false and it will signal the host to stop sending by setting the CTS line false. If your software does not keep the RTS line true most of the

time (for example, you don't even care about the responses), responses to commands will back up in the converter until it runs out of memory, and then it will not execute any more commands. To prevent this from happening, send the command "H 0". This will prevent the converter from looking at the RTS line, it will just send out responses whenever it has them, even if the host computer isn't hooked up. This is recommended if you are executing Scripts with no host attached. Sending "H 1" turns handshaking back on.

The converter will ALWAYS generate CTS signals, no matter what H is set to. The host computer should obey this line.

The Z command will save the current setting of H and set it at the next power cycle.

Setting handshaking off MAY cause loss of characters on long replies (such as from the "I" and "?" commands). This does not affect the DMX output in any way and if software on the host doesn't care about the lost characters, then no harm is done.

I INFORMATION COMMAND

When the converter receives this command it will respond with several lines of text describing the software and what the initial conditions are. Here's an example of what the converter will send:

```
RS-232/USB to DMX Converter
Copyright 1999-2004, Durand Interstellar, Inc.
Version: 4201 1.41 20041009
Serial Number: 00000000
Playback mode: 01
Start Code: 00
Channels: 040
Serial Handshaking: ON
USB Connection: OFF
```

You may wish to set up your software to check that you're talking to the right converter (our other products will also have an 'I' command). The first number after "Version:" is the model number ("4201"), the next number is the firmware version ("1.41") and the last number is the firmware date ("9 October 2004"). The playback mode reflects the current setting (00 = off, 01-7F = which NV-RAM preset was last copied to active memory). Serial Handshaking indicates whether the converter pays attention to the RTS signal from the host. The USB Connection indicates if a USB connection has been detected since power up. Other versions of this product may have higher playback mode numbers and more lines of information.

N NUMBER OF CHANNELS

This command lets you set the maximum number of channels. If you are using a small number of channels, this GREATLY increases the number of packets sent out per second and the speed that scripts execute. Note that if you change Playback Mode it will ONLY copy the number of channels set by this command. The higher channel numbers in active memory will not be changed (and can be used for data storage by scripts).

If you set N to 0, DMX packets of zero length will be sent. Be careful, as this may confuse some devices. If your device doesn't mind this, you could use this as a way to pause the DMX transmission while you update active memory with the "W" command. A better way would be to issue the "P 0" command to turn off the output, make your changes, then issue the "P FF" command to turn it back on.

The maximum number of channels is 512 (200 hex), the factory default is 64 (40 hex).

P SET PLAYBACK MODE

Valid modes are 00 for off and 01-7F for auto-playback. As soon as this command is entered, the DMX output is stopped. If the new mode is not equal to zero, the selected preset is copied from NV-RAM to the active memory (Preset 0) and the DMX output is re-started. This all happens VERY quickly.

There's a special case where you set the mode to FF. This does NOT change the playback mode setting, it only turns the DMX output back on if it was off. Whatever data is in Active Memory will now be sent out.

On powerup, the auto-playback preset is loaded before the auto-execute script (if any) starts.

Samples:

P0

P4

PFF

R READ MEMORY

This command will cause 16 bytes to be read back to the host from memory. The memory is accessed as a single 64K block, so you supply the starting address as a 16 bit hex number where the upper 7 bits are the preset number. Addresses 0000-01FF are channels 1-512 in the active memory (these are what are sent out the DMX port). The rest of the memory block is the NV-RAM. This command always sets the lower 4 bits of the address low before starting the read, so the last digit of the address you send doesn't matter (but is required). So if you send 123, the first byte returned will be from Preset 0, channel 120 (hex). The first line below shows a read back command for

channels 120 through 12F, the second line shows a typical response:

R123

11 22 33 44 55 66 77 88 99 00 AA BB CC DD EE FF

Each channel is returned as a 2-digit hex number with a space between channels.

To read back the first 16 channels of NV-RAM Preset 1, you would send “R200” (200 hex times the preset number). To read back NV-RAM Preset 2 you would send “R400”. The very last channel in the last NV-RAM Preset is at address “FFFF”.

S START CODE

This command lets you change the DMX Start Code to some other number. The only valid number defined by the DMX Standard is “00”, but other devices may require a different start code. There’s also no reason you can’t change the start code between “D” commands. This way you could have some non-standard item on the same DMX cable as well as normal lighting dimmers. Your computer would have to keep track of what it was sending to which device.

Example:

S0	(control regular lighting channels)
D 1 2 3 4 5	(send lighting data)
S55	(special start code for non-standard item)
D A B C D	(send commands to non-standard item)
S0	(back to beginning of loop)

TD TIME DELAY

This command will allow you to pause processing any further commands until the delay has passed. The format is “TD” followed by the delay in hours, minutes, seconds, and tenths of seconds. It may be aborted with the “*” command. To delay for 1 hour, 2 minutes, and 3.4 seconds, send:

TD 1 2 3 4

You may truncate the line, the missing numbers will be assumed to be zero. To delay for 2 minutes exactly, you would send:

TD 0 2

See note below on Timer Range

TR TIMER READ

This command will give you the current timer value in hours, minutes, seconds, and tenths of seconds. The hours are returned as a 4-digit hex number, all other numbers are 2-digit hex.

See note below on Timer Range

TS TIMER SET

This command will set the timer to whatever value you wish. It is useful

for setting a known time for the start of a show (often 0h, 0m, 0.0s). To set the timer to all zeros, simply enter:

TS

To set it to 1 hour, 2 minutes, 3.4 seconds enter:

TS 1 2 3 4

See note below on Timer Range

TW TIMER WAIT

This command will wait until the given timer value (like an alarm clock). It may be aborted with the '*' command. If the time requested has already passed, you will get an error. To wait until the time is 1 hour, 2 minutes, enter:

TD 1 2

See note below on Timer Range

U UPDATE FIRMWARE

This command is ONLY to be used when there is a firmware update available for your model number (firmware updates will NOT work on different models). If you have custom firmware, then the updates may only work for one specific serial number. Use the "I" command to check your model and serial numbers (these should also be on the bottom of the converter).

Assuming you have a new update that we sent you or you downloaded from our web site, then the first thing you should do is a test run. This will check the update file for errors, but won't actually change the existing firmware. To do this with a terminal emulator, you should make sure you get a ">" prompt every time you press return. Then, type "UT" but do NOT press return. Open your Send File menu and select ASCII TEXT. Now send the update file. This will take a few minutes and the yellow light will be on or flashing. If everything worked, you'll receive a message telling you that there were no errors and to re-run it with "UW" instead of "UT". "UW" will write the new firmware to memory.

DO NOT INTERRUPT THIS PROCESS!

If you get an error message (which you shouldn't since you ran the "UT" command with no problems), then you may try again (you are still running on the old firmware until you cycle the power). If you can't get the update to complete properly, you probably have a dead unit that must be returned to the factory for repair.

When the update completes normally, you will receive a message that everything went ok. At this point the processor halts and you should cycle the power to load the new firmware. Issue an "I" command and check that the version has changed to the latest.

You will also notice that the setup information saved by the "Z" command is erased. Presets 2-63 are unchanged, but Preset 1 has been cleared to all zeros

(since the default Playback Mode is 01 you don't want to turn on all the lights full by accident), the Start Code is set to 0, and the Number of Channels to 64 (40 hex). The Auto-Exec Script (Script #0) will also be cleared.

W WRITE TO MEMORY

This command allows you to update locations in the memory (active or NV-RAM). You supply a 16 bit starting address followed by 1 or more bytes of data to be written. To write the data (1, 2, 3, 4) to channels 5-8 of active memory, you would use this command:

W 4 1 2 3 4

Note that the memory address is always 1 less than the channel number (assuming channel numbers for your dimmer start at 1). If you are writing to addresses in the active memory and the DMX output is on (green light lit), every change you make will be sent out to the DMX port immediately. This is the preferred way to dynamically control lights from your software; you only need to send the changes and they take effect immediately. If you are writing to NV-RAM (addresses 0200-FFFF), each channel is stored as you enter it but it does not affect the current DMX output. Presets start every 200 hex addresses (512 bytes), so preset #3 would start at 600 hex and run through 7FF.

This command will allow you to write past the end of one preset and into the next. This could be handy for writing several presets all at once.

XE EXECUTE SCRIPT

This command will start execution of a script. You enter the command as:

XE 0 1

Where 0 is the script number (0 = Auto-Exec Script) and the 1 is the Label you wish to jump to. If the label is left off, execution will start at label 0.

If you specify a preset that isn't a valid script or the label is not found, you will get an error message. Also, if during execution an error occurs, a message will be sent to the host. All errors result in the script aborting.

Be sure to see the note below on Scripts.

XR READ SCRIPT

This command will copy a script from memory to the host. You enter the command as:

XR 0

Where 0 is the script number (0 = Auto-Exec Script).

Be sure to see the note below on Scripts.

XW WRITE SCRIPT

This command will copy a script from the host to memory. You enter the command as:

XW 0 :0; TS; F1 100; TW 0 1; F0 200; J 0

Where 0 is the script number (0 = Auto-Exec Script) followed by ONE white space character. The rest of the line will be written to NV-RAM as the script.

The example script will clear the timer, fade to preset 1, wait until the timer gets to 1 minute, fade to black, then jump to label 0 (loop forever).

The Auto-Exec Script can hold just under 400 characters (INCLUDING white space), while the others can hold up to 511 (an entire preset terminated with '\0').

To erase a script, enter the XW command followed by the script number, then press RETURN.

Be sure to see the note below on Scripts.

X? SCRIPT HELP

This will print out a list of the currently available Script Commands. The letters in parenthesis to the right of a command are the Flag Bits that are updated by this command.

Be sure to see the note below on Scripts.

Z WRITE STARTUP DATA

This command will copy the Playback Mode, Start Code, Number of Channels, and Handshaking to NV-RAM. These settings will be recalled the next time power is cycled. So if you don't want anything to transmit on power up, clear the Auto-Exec Script, set Playback Mode to 0 and then issue the "Z" command. Note that you only have to set this up once and it will work every time power is cycled.

*** ABORT TIMER COMMAND**

This will abort any current timer or script-based command (in case you told it to wait 20 hours instead of 20 seconds). Currently the commands affected are TD, TW, F, and all the Script commands. Everything that may have been sent between the timer/script command and the '*' will be ignored. If the USB interface is being used and the host computer sends a RESET, SETUP, or SLEEP command, it will also cause an abort.

~ EXTERNAL EVENT

This is a special command for controlling the execution of Scripts. The controller keeps an 8-bit Event variable in memory. This is set to 0 upon

power up, and is updated with this command as well as through scripts. The format of this command varies slightly depending on if a Script is running or not. Also see the “ER” and “EW” commands.

If a Script is not running and you want to send the event 0x55, you would send the command as:

~U

followed by a carriage return. The “U” is the ASCII character 0x55. You may send any 8-bit value, but it **MUST** be the very next character after the ‘~’. Also be careful with sending ‘*’ as it will be interpreted as an abort (abort has a higher priority when scripts are not running). The last event sent will be held and will be immediately available to any Scripts running. Also, see the EW command.

If a Script IS running, you can send the “~U” at any time, even in the middle of entering another command and even if CTS has been set false (do not send carriage returns or you could fill up the buffer). Note that **EVERY** character sent to the converter while a script is running is checked to see if it’s a ‘~’ **BEFORE** any other checks are performed. The character following the ‘~’ will **ALWAYS** be interpreted as an event and both characters will be removed from any further processing. The converter will not give any response to receiving an event other than anything you may have programmed your Script to output.

+ **INCREMENT CHANNEL**

This command allows you to increment one or more channels by set amounts. It is intended for applications where a simple controller like a PLC or architectural lighting control needs to fade one or more channels up but doesn’t have the processing power to calculate the values itself.

You enter it by typing a ‘+’, followed by the starting channel number, followed by white space and a list of amounts to add to the existing channel values. For example, the following command:

+ 5 1 2 0 3

Will start with channel 5 and add 1 to whatever value was stored there. It will then add 2 to channel 6, 0 to channel 7, and 3 to channel 8. All channel data is clamped meaning that if the channel was set to the value 0xFE and you increment it by 1 or more, the resulting value will not exceed 0xFF. The amount you can increment by can be any value from 0 (do nothing to this channel) to 0xFF (go to full on).

If your list of values causes the channel number to go past the last channel (0x1FF), it will wrap back to channel zero. This could be handy, or dangerous, depending on your view.

- **DECREMENT CHANNEL**

This command is identical to the INCREMENT CHANNEL command, only the values you supply are subtracted from the channel data. The channel data will clamp at zero, so you don't need to worry about decrementing too much.

? **HELP**

This will print a list of the currently available commands.

NV-RAM Notes

The original version of this controller used FRAM for all the memory. This had good and bad points. FRAM is non-volatile (maintains the contents with the power off) and is fast enough to keep up with the DMX output speed, but until recently would burn out too soon. This forced us to set the DMX packet rate (number of times per second that one set of “N” channels is sent out) very slow. This made it hard to control certain devices or get smooth fades without sending constant data using the “D” command.

Now, the processor used in this version has lots of internal static RAM (SRAM) and several other nice features. This is something you just couldn’t find when we built our first converters in 1999. The DMX data is now sent from this SRAM (Preset “O”), so the number of reads/writes does not matter. This allows a control program to use the “W” command to just send changes to the active memory. Presets 01-127 are now stored in EEPROM. EEPROM isn’t all that fast, but is only accessed when you change presets or edit it (neither of which should mind a few milliseconds delay). Now here’s the warning: an EEPROM allows unlimited reads but is limited to about 1 million writes before it is burned out. This limit is per memory location, so if you constantly wrote to Preset 1 it would eventually burn out (return invalid data) but the rest of the memory would still work fine. So, if you have a need to change presets a lot, change which one you use from time to time.

If you were to change Preset 1 every hour non-stop, it would burn out in 114 years. If you changed it every minute, it would burn out in just under 2 years. Still not bad.

If you DO burn out the memory, you can easily replace it. They’re the only chips in sockets on the board (“U6” & “U8”, Microchip 24LC256IP) and retail for under \$2 each. Presets 1-63, the “Z” data, and the auto-execute script are in “U8”, Presets 64-127 are in “U6”).

There’s another possible alternative. Ramtron, the maker of the FRAM we used in the past, has announced a part that can replace the EEPROM in the converter. It will still burn out, but it takes 100 billion cycles, and that’s a long time. The new boards support the FRAM chip, but large chips are not yet available, you would only have 15 presets. If you’re interested in swapping out your EEPROM for the new FRAM, contact us for more information.

The first time you turn on the power after replacing the EEPROM, the firmware will not recognize the EEPROM and will clear the information saved with the “Z” command as well as all of Preset 1 and the Auto-Exec Script. This also happens if a new version of firmware is loaded. This prevents strange things from happening if unknown data is read from the EEPROM upon power up. The defaults are P=1, S=0, N=040 (hex), H=1, Preset 01 and the auto-exec Script all zero.

Timer Notes

In order to keep track of real-time events, this converter maintains a real-time clock. Most commands don't care what the absolute value of the clock is, only how much time has passed since the command was issued. If you never use the "TW" or "TT" commands (which use an absolute time), then you also probably will never need to set the clock with the "TS" command or read it with the "TR" command.

If you DO wish to use absolute time, there are a couple of things to keep in mind. First, the clock can count up to about 6 months before rolling back over to all zeros, but the timer commands will only accept times up to about 10 days. If you have a need to time longer than that or you want the clock synchronized to your watch, you're going to have to set it at least every 10 days with the "TS" command. This shouldn't be a problem as long as a host computer is connected (just issue a "TS" command at the beginning of each show).

Next is accuracy. This clock isn't intended to be super accurate and may be off by as much as a minute or two a day. Again, if you need more accuracy, your host computer can reset the clock as often as you wish. Most shows that use absolute time will want to start at zero for every show anyway.

When you are issuing any command that requires you to supply a time value (such as "TS"), the format is:

TS hh mm ss tt

hh = hours, range is 00-FF (0-255 hours)

mm = minutes, range is 00-FF (0-255 minutes)

ss = seconds, range is 00-FF (0-255 seconds)

tt = tenths of seconds, range is 00-FF (0-25.5 seconds)

You may leave off tt, if you wish.

You may leave off ss if you also leave off tt.

You may leave off mm if you also leave off ss and tt.

You may leave off hh if you leave off mm, ss, and tt.

Any values you leave off will be set to zero. The command "TS" with no numbers following it will set the clock to all zeros.

You'll notice that other than hours, you are allowed to enter values larger than what's normal in that place. This allows you to enter 78 (hex) in the seconds position and get the same result as if you entered 2 minutes. All of the following will get you a 2 minute delay:

TD 0 2

TD 0 0 78

TD 0 1 3A 14

When reading back the time, you will see FOUR digits for the hour, and 2 for the rest of the numbers.

Script Notes

The Model 4201 Converter has a powerful scripting language built in. More commands may be added in the future, but the current commands allow a lot of options. Using the Script feature requires a little care, as is the case whenever you are programming a computer or PLC.

A little explanation about how Scripts work. The Auto-Execute Script (#0) is stored in a reserved part of NV-RAM but Scripts 1-127 use the SAME NV-RAM locations as Presets 1-127. This means you can read/write Scripts 1-127 with the R and W commands and also read/write/execute Presets 1-127 with the X commands. The Scripts are stored in straight ASCII Text with a NULL (0x00) to terminate the string. The difference between the XW command and the W command is that XW accepts a text string and adds the NULL at the end for you. Feel free to use either method of editing Scripts, just be careful. You **MUST** use the XW and XR commands to write/read the Auto-Exec Script (Script 0).

Since Scripts and Presets use the same memory, this means you can confuse things. If you write a Script into location 0x10 and then issue the “P 10” command, you will send the Script out as lighting levels. Probably not what you had in mind. We suggest either sticking with only Script 0 which is in a reserved place in NV-RAM or placing all the scripts at the end of NV-RAM while placing all your Presets at the beginning of NV-RAM. Something like 1-100 = Presets, 101-127 = Scripts. In almost all cases, Script 0 should be enough for you.

There is one other thing about Script 0, it can only hold just under 400 characters while the rest of the Scripts can use the entire Preset they are in (511 characters plus one NULL).

When a Script is started, it is copied to RAM and scanned to find all labels. A script **MUST** have at least one label before the first executable command. After all the labels have been located, the Script is executed from whatever label was specified when it was called (if no label was specified, 0 is assumed).

You may have any amount of white space and/or comments in the Script that you want, but excess space uses up memory and also slightly slows down execution of the Script.

Every command **MUST** be terminated with the ‘;’ character, except for the last command in the Script. If execution reaches the end of the Script, execution is aborted.

The math instructions treat the Event data as an accumulator in a microprocessor. All math is done through the accumulator (Event) and the results may effect the Flags and/or one of the lighting channels (0-0xFF). If the host sends new Event data with the “~” command while a script is executing, the new data will overwrite whatever a math instruction may have written to Event. It’s probably best to not use math instructions that change Event when the host is also changing it.

The Flags are stored in a single byte and are defined as follows:

bit 0	Zero	The last instruction caused Event to be zero
bit 1	Carry	The last instruction caused an over or underflow
bit 2	Jumper	Status of the third jumper on J7, 1 == shorted
bit 3	Sign	The last instruction caused bit 7 of Event to be high
bit 4	Digital 4	Status of a digital input, low == 1
bit 5	Digital 5	Status of a digital input, low == 1
bit 6	Digital 6	Status of a digital input, low == 1
bit 7	Digital 7	Status of a digital input, low == 1

The Carry bit is also set by the TT command. The Zero, Carry, and Sign flags may also be set using the MF command and “EW” from the host.

Flag bits can be tested with the J (jump) instruction.

The following commands are also available for use INSIDE a Script:

F, P, S, TD, TS, TW, TR, +, -

All the same as the regular versions. The TR command sends the time back to the host. Handy for debug and triggering other devices. The TW command does NOT generate an error if the time has already passed, it just continues execution of the script.

TT Same as the TW command, except instead of waiting for the time to pass, the Carry bit is set if the current time is greater or equal to the specified time, otherwise Carry is cleared. Execution does not pause.

Q Quit Script, return control to host.

: Set a label. Example “:0;” is label 0. Valid range of labels is 0-0xFF. Labels do not have to be in order (you can skip around);

‘ Comment. Anything following the apostrophe until the next ‘;’ will be ignored. Reserved characters like ‘~’ are not allowed in comments. Comments slow down execution slightly. Comments must start where a valid command would start (not before the ‘;’ of the previous command).

L Link to another Script (“long jump”). “L 10 5;” would start Script 0x10 at Label 0x05, the same as sending “XE 10 5” from the host. “L 10” would start Script 0x10 at label 0.

J Test Event and/or Flag Bits and jump to label. The format is:

J 1 2155 0150;

Where the 1 is the label to jump to, the 0x2155 will be ANDed with the most recent Flags and Event data, and the 0x0150 is what must be matched to perform the jump. So, in the case of the example,
 “if(((Flags & 0x21) == 0x01)) && ((Event & 0x55) == 0x50))) Jump to Label_1”.

If you aren't interested in the Flags, you can just use bytes to look at the event as in:

```
J1 55 50;
```

If you leave the last number off (the 0x3450), the result must equal zero. If you leave off both numbers, then this will be an unconditional jump.

```
J1;
```

Will jump to label one in all cases. If the label is not found, an error is generated.

Jumps are made to absolute addresses, so execution speed is not affected by the length or direction of the jump.

- O Output string to host. The string format is similar to a "C" string.
Example: O "String to host.\r\n";
Restricted characters are '*', ';', ':', '"', '~', '\', and NULL. Do NOT use these characters directly in a string. If you wish to use any of these, use the escape commands below. Note that to send the '\' from "C" you need two of them "\\".

```
\a = bell
```

```
\b = backspace
```

```
\f = formfeed
```

```
\n = newline
```

```
\r = return
```

```
\t = horizontal tab
```

```
\v = vertical tab
```

```
\` = ~ (tilde)
```

```
\8 = * (asterisk)
```

```
\. = : (colon)
```

```
\, = ; (semi-colon)
```

```
\# = Event data as a two digit hex number
```

```
\\ = backslash (from "C" this would look like "\\")
```

Math commands. All the following commands work with the Event data as if it were an accumulator and the Active Memory (Preset 0) as if it were 512 registers. Note that any channels above the number set by the "N" command will NOT be changed with the Fade or Preset commands and so may be used for general storage (data in these locations will be random when the script starts unless a Copy command was issued before the script started). In most applications, this will give you several hundred bytes of storage. Any changes to a channel that's currently being sent out will take effect on the next update (essentially instantly).

Some commands set/clear flag bits depending on the results of the operation. The host can force-load the Event data at any time with the "~" command, so use this carefully.

The first two commands ('+' & '-') do NOT start with 'M' to differentiate them from the "M+" and "M-" commands.

+ INCREMENT CHANNEL[Event]

This command allows you to increment a channel by a set amount. Note that this differs from the normal '+' command in that the channel number is taken from the Event register and you supply a fixed increment amount. Also, you can only increment one channel at a time from inside the script, but you could easily write a loop to increment the channel number and then the channel.

You enter it by typing a '+', followed by white space and the amount to add to the existing channel values (pointed to by the Event register). For example, the following command:

+ 1

Will add 1 to whatever value was stored in the channel pointed to by the Event register. Zero, Sign, and Carry are updated.

All channel data is clamped meaning that if the channel was set to the value 0xFE and you increment it by 1 or more, the resulting value will be 0xFF. The amount you can increment by can be any value from 0 (do nothing to this channel) to 0xFF (go to full on).

Note that since the Event register is only 8 bits, this can only be used on the first 256 channels (00-FF).

- DECREMENT CHANNEL[Event]

This command is identical to the INCREMENT CHANNEL[Event] command, only the value you supply is subtracted from the channel data. The channel data will clamp at zero, so you don't need to worry decrementing too much.

MA Bitwise AND channel into Event. MA 105 will read the data from channel number 0x105 and AND it into the Event data, with the result left in Event. The Zero and Sign flags will be updated.

MD Divide Event by a channel. MD 1 will divide the Event by the data in channel 1 and leave the results in Event. Zero and Sign are updated. Divide by zero results in 0xFF in Event.

MF Set the flag bits (z, c, s only). MF 0 will clear all three bits.

MIL Load Event indirectly. Assume channel 0x103 has the data 0x12 in it. MIL 103 will copy the data from channel 0x12 into Event, using channel 0x103 as a pointer. In order to indirectly access channels 0x100-0x1FF, set

the high bit of the pointer channel number to one.

MIL 8103 will copy the data from channel 0x112 into Event. No flags are changed.

- MIW Write Event indirectly. The same as MIL, except Event is copied TO the channel pointed to. If channel 0x103 has 0x12 in it, MIW 103 will copy Event to channel 0x12.
- ML Loads Event with the data from the channel. ML 5 copies the data in channel number 5 into Event. No flags are changed.
- MM Multiply Event by channel. MM 4 will multiply the Event by the data in channel 4, leaving the results in Event. Zero, Sign, and Carry are updated. Carry set on overflow.
- MN Bitwise compliment of the data in Event. If Event is 0xAA and you issue the MN command, Event will become 0x55. Zero and Sign are updated.
- MO Bitwise OR channel data into Event. Zero and Sign are updated.
- MR Load Event with a random number. The normal "C" rand() function is used. The seed is set from an interrupt counter the first time MR is called, so there will be a variation in the sequence of numbers every time you power up the unit. The numbers returned will fall between 0 and 0xFF with any number as likely as any other. Zero and Sign are updated.
- MS Swap the data in Event with a channel. Assume Event has 0x12 in it and channel 3 has 0x34 in it. When you execute the MS command, Event will now have 0x34 in it and channel 3 will have 0x12. No flags are updated.
- MW Write Event to a channel. MW 6 will copy the contents of Event to channel #6.
- MX Bitwise XOR Event by a channel with the results left in Event. Zero and Sign are updated.
- M# Load Event with a constant. M# 6 will place the number 0x06 in Event. No flags are updated.
- M+ Increment Event. Zero, Sign, and Carry are updated. Carry will be 1 if Event rolled over to zero.
- M- Decrement Event. Zero, Sign, and Carry are updated. Carry will be 1 if Event rolled over to 0xFF.
- M> Roll Event Right through Carry. Assume Carry is 1 and Event is 0x10. After the M> command, Event will contain 0x88 and Carry will be zero. Zero,

Sign, and Carry are updated.

M< Roll Event LEFT through Carry. Assume Carry is 1 and Event is 0x10. After the M< command, Event will contain 0x21 and Carry will be zero. Zero, Sign, and Carry are updated.

Let's walk through an example.

First, write the script to the Auto-Exec location

```
XW0 :0; P 1; TD 0 1; F 2 100; J 10 FF 1; F 1 200; J 0; :10; F 3 100; TD 0 1; J 0
```

We wrote this to the Auto-Exec position, so you can either manually start it with “XE0” or just cycle the power and it will start executing.

:0;	Label 0
P 1;	Copy Preset 1 to Active Memory
TD 0 1;	Delay for 1 minute
F 2 100;	Fade to preset 2 over 25.5 seconds
J 10 FF 1;	Conditional jump. AND Event with 0xFF and compare to 0x01
F 1 200;	Fade to preset 1 over 51.2 seconds
J 0;	Unconditional jump to Label 0
:10;	Label 10
F 3 100;	Fade to preset 3 over 25.5 seconds
TD 0 1;	Delay for 1 minute
J 0	Unconditional jump to Label 0
	(semi-colon not needed since it's the last command)

This will loop forever unless an Abort (“*”) is received.

It's ok to add comments after the last command, assuming the last command is either an Unconditional Jump or Quit. Be sure there's a semi-colon ‘;’ between the last command and the comments. You can also put comments BEFORE the first label, if you wish. Anything before the first label will not be executed (do NOT put colons in the comments!).

The label that starts the Script does NOT have to be the first label. The Auto-Exec Script always starts at Label 0, but the label may appear anywhere in the script.

Here are some scripts that show a few more possibilities:

Use a 2 pole rotary switch to select one of four presets:

```
:0; J1 3000 0; J2 3000 1000; J3 3000 2000; J4 3000 3000; :1; 'Fade to Black; F0 0 0 5;
J0; :2; 'Light Door; F1 0 0 10; J0; :3; 'Light Window; F 2 0 0 20; J0; :4; 'Flash; TS; :5;
P3; TD 2; P4; TD 2; TT 0 0 2; J5 200 0; J0
```

The above would all be one script without the line breaks. Here's what's going on:

:0; Label 0 is the default starting label for a script

J1 3000 0; J2 3000 1000; J3 3000 2000; J4 3000 3000; This is a "jump table", it tests the digital inputs 4 & 5 (which are connected to external switches, or other device) and then depending on their levels, decides where to jump next. Note, the first test "J1 3000 0;" could be left out since the other tests would fail, letting the execution continue with label 1 anyway.

:1; 'Fade to Black; F0 0 0 5; J0; This label is jumped to if both switches are open (high). The comment will be ignored and the lights will fade to black in 0.5 second, then execution will jump back to testing the switches. Note that if the switches haven't changed, this will be executed over and over again. In this case nothing is hurt by this since fading from black to black doesn't change anything. A better version of this might be:

```
:1; 'better Fade to Black; F0 0 0 5; :11; J11 3000 0; J0
```

This will fade to black once, and then loop just testing the switches until they change to some other value. This same sort of thing can also be done for sections 2 & 3 below.

:2; 'Light Door; F1 0 0 10; J0; This will fade the lights to preset #1 over 5 seconds. Note the comment above about a better jump condition (otherwise the switches will only be tested every 5 seconds at the end of each fade).

:3; 'Light Window; F 2 0 0 20; J0; See "Light Door" above.

:4; 'Flash; TS; :5; P3; TD 2; P4; TD 2; TT 0 0 2; J5 200 0; J0 This one's a bit more interesting. It starts out by resetting the clock to all zero. Then it sets the lights to Preset #3 with no delay (let's say this is all lights on). It then delays 0.2 second and sets the lights to Preset #4 (lets say this is all lights off). It waits 0.2 second more, then it tests to see if 2 seconds have gone by since the TS command. If not (carry = 0), then we jump to label 5 and do one more blink cycle (without resetting the clock). Once 2 seconds have gone by, the TT command will set the carry high, causing the J5 test to fail. We then jump back to Label 0 to start all over again.

Use two switches to raise/lower a set of lights:

```
:0; TD 0 0 0 1; J1 3000 1000; J2 3000 2000; J0; :1; 'raise lights; M# 0; + 2; M+; + 3; J0;
:2; 'dim lights; M# 0; - 2; M+; - FF; J0
```

Here's what's going on:

:0; TD 0 0 0 1; The script starts at label 0, then delays for 1/10th second.

J1 3000 1000; J2 3000 2000; J0; This is a jump table. If one switch is pressed, it jumps to label 1 (fade up). If the other switch is pressed it jumps to label 2 (fade down). If both or neither switch is pressed, it jumps back to label 0 to start over (does nothing).

:1; 'raise lights; M# 0; + 2; M+; + 3; J0; Here we have a label, a comment, and then we load the value 0 into the Event register. We then increment channel zero (based on Event being zero) by 2. Then we increment Event and increment channel 1 by 3. Now we jump back to label 0 to start over.

:2; 'dim lights; M# 0; - 2; M+; - FF; J0 This is similar to the above, only we're decrementing. Note also that channel 1 will go full off at the first press of the switch and stay there.

Check the Support section of our web page for more examples.

Power Notes

This converter runs on 3.3V internally and has a voltage regulator that is specified to work from about 4.5V to 26V. To allow a little margin, we have specified the voltage requirements as 5-24VDC at about 150mA (a DMX short will draw more current but not hurt the converter). The voltage regulator has built-in protection for use in automotive environments (reverse battery, 60V spikes). The lower the voltage you run the converter at, the cooler it will run (24VDC at high ambient temperatures is not recommended). If you are using the USB connection to your computer, your computer will normally supply all the power needed. If you need external power, you may use your own AC adapter (we recommend between 5 and 9VDC at 200mA or more) or you can order an AC adapter and/or DC power cord (with or without a cigarette lighter plug on it) from us. Here are the specifications for the power plug:

Co-axial Power Plug
2.1mm internal diameter
5.5mm external diameter
Center Positive

You do not need to worry about external power conflicts when using our current USB based products. There are blocking diodes in line with the AC adapter and USB power so there is no interaction between the two (the adapter won't try to power the host computer). If the AC adapter is plugged in, then the converter will use that power and not the host computer's (saving a laptop's batteries). If the host is turned off or the USB cable is unplugged, the converter will continue to operate. If AC power is lost or the adapter unplugged, the converter will switch to using power from the host, if one is connected. It's perfectly ok to plug and unplug the AC adaptor while the USB cable is plugged in.

Due to the blocking diodes, voltage does not appear at the power jack when the AC adapter is unplugged. You may not use this as a power source for other devices (our earlier models would supply 5VDC at the power jack when plugged into a USB host).

USB Notes

USB = Universal Serial Bus. See <http://www.usb.org/> for more information.

There are several types of USB devices that all host computers are supposed to know how to talk to without any additional drivers. Unfortunately, serial ports aren't one of them. In order to avoid writing and maintaining a separate driver for each different version of Windows (not to mention Mac, Linux, and other operating systems), we have our converter appear to the host computer as one of the devices that do not need a driver. It appears to the host as a Human Interface Device (HID) of generic type. Visual Basic and other software can talk to HID ports with no other drivers needed (see the DMXC and other demo programs). For compatibility with other software, we also supply a Windows driver that makes our HID port appear as a regular COM port. While it will work, this driver is not recommended for Windows XP due to problems XP has with unsigned drivers (XP will try to avoid using the driver).

In order to reduce the load on our firmware when using the converter with the regular serial cable, the converter will only check for a new USB connection about every 5 seconds, so when it is plugged into a computer for the first time it may take up to 10 seconds for the computer to find and initialize it. Once the converter has been initialized as a USB device, the serial port will not work until the power is cycled without a USB cable attached.

Some things to be aware of with USB devices

- This converter is a "Full Speed" USB device (12Mb/s) and may be plugged into either a "Full Speed" or "High-Speed" port (USB versions 1.0, 1.1, 2.0)
- The USB is a shared resource. If you have a lot of devices (or a few data intensive ones like cameras) on the bus, the amount of data you can send to the converter will go down.
- With USB devices, if you plug and unplug them rapidly or several times in a row, some host computers may get confused and crash. Whenever attaching a USB device, plug the cable in quickly and don't unplug it again for several seconds. Also wait before plugging it back in.
- If the host computer or USB hub thinks a device is drawing too much power, that particular device will be turned off. There is no harm done (other than the DMX Converter shutting down), but this often means rebooting your computer to get the power back on. Some older hubs are much more sensitive than newer ones. If you have any trouble with this, use an AC adapter to supply power to the converter (this will also keep the DMX signal alive, even if the host or hub is turned off and save a laptop's batteries).
- Computers will try to place USB devices into SUSPEND (power down) mode when the computer isn't used for a while. This converter will refuse all SUSPEND requests to prevent any disruption of your show (but will abort any fade or script running when a SLEEP or RESET is received). This may prevent your computer from entering SUSPEND/SLEEP/HYBERNATE mode while the converter is plugged in.

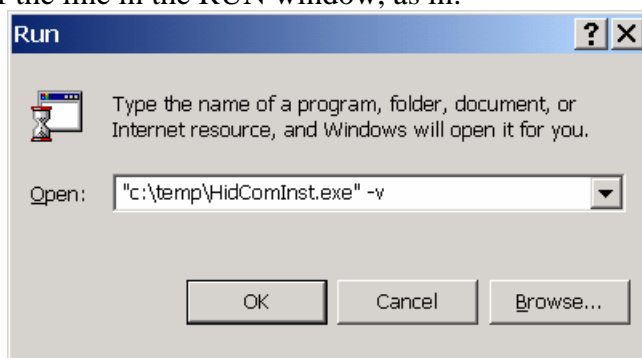
Serial Driver Installation

The serial driver should NOT be installed if you are using HID-aware software such as the DMXC Demo, Two Scene Demo, Edgeview Software's SmartLight, or MacFOH software.

To install the HID to Serial driver, create a scratch directory somewhere on your system (if you have our CD, you can just execute the install directly from the CD). Unzip the contents of the USB_Driver Zip file into the scratch directory or floppy (if you don't have an un-zip utility, go to <http://www.winzip.com/> to download one).

Click on START and then select RUN and BROWSE to where you unzipped the install files (or to the folder on the CD) and select HIDCOMINST.EXE. Click OK to go back to the RUN box and click OK again to run the install program. By default it will not give any indication that it is running, but you should get a Windows hour glass for less than a minute. Any previous versions of this driver will be deleted and replaced with the latest.

If you would like to see a progress report of what the install program is doing, add a space and -v at the end of the line in the RUN window, as in:



and a status window will open showing everything that is going on. You will see some messages about files not found or not able to be deleted as it is checking for old versions and other files that may have been previously installed.

Reboot your computer and then plug the USB cable into the converter. It may take up to 10 seconds for the computer to find and start initializing the converter. If it asks you to insert the HID install disk, BROWSE to where you saved the install files (or to the CD) and select OK.

Once Windows has assigned a COM port number to a DMX converter, it will always use the same number as long as you plug the same converter into the same USB port (often, it doesn't matter which port you use). However, each converter has a different serial number so if you plug in a different converter, Windows will see it as a new device and assign a different port to it. If you start getting too many ports assigned, simply re-install the software and all the ports will be deleted.

Installations to Windows 2000 go very smoothly. If you open your Device Manager you should find you have a new DMX PORT with a COM number assigned to it. You are now up and running.

Windows XP will complain that the driver has not been certified by Microsoft, install it anyway. The DMX converter will probably appear as an HID device. Pick the 4201 driver icon from the HID group and select update driver, pick the manual option, then pick the non-virtual Durand Interstellar driver. The icon disappears from the HID group and reappears in the port group as a COM port and it remains this way through plugging and unplugging the USB port.

The drivers are supposed to work on Windows Me, but have not been tested. Installation should be similar to Windows 2000.

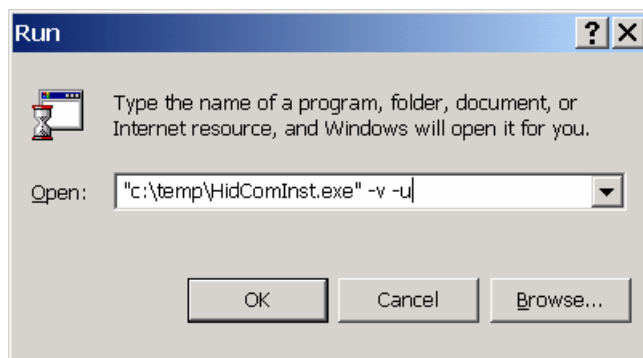
Windows 98 should be upgraded to the Second Edition (Win-98SE) before installing the serial driver. Win-98SE isn't as stable as Windows 2000 and there seem to be a lot of versions of it in use. Some systems seem to install just fine, and then suddenly decide that they've found a modem. The problem with this is the converter may not be assigned a COM port number, preventing you from talking to it. Win-98SE may also ask you for files that are not supplied with our normal install disk, these are Microsoft files that can usually be found somewhere on your hard disk (use FIND FILES) or on the original INSTALL disks that came with your computer. Windows 98 will also show TWO DMX converter ports, do not use the Virtual one. You may have to experiment to find what COM port number has been assigned to the converter, the port number will not change as long as you always plug the converter into the same USB port.

MAC, Linux, and Others

ANY system that supports USB 1.1 or later will install this converter as an HID without any extra driver files. Many times (such as with an iBook) there won't even be a notice that the device was installed. If you look through your device manager (whatever it's called on your system) you'll find it listed with various information about the device. You will need special HID-aware software in order to use our converter on a non-Windows system.

HID Installation

If you wish to communicate with the converter using the HID-aware software, do NOT install the driver as that will hide the HID port from your software. If you have already installed the driver, you may un-install it by clicking on START and then select RUN and BROWSE to where you unzipped the install files and select HIDCOMINST.EXE. Add a space, then -v, then another space, then -u to the end of the command.



You will probably see messages about files not found or not able to be deleted, the software is checking for various files that may have been installed. When it completes, it will prompt you to “press any key” and will then close the status window. Reboot your computer and plug in the converter (if it isn’t already plugged in) and you should now see the converter listed under “Human Interface Devices” in your Device Manager. If it still shows up as a COM port, right click on it and select UNINSTALL. When it is done, reboot your computer and it should NOW be listed as HID. You can test it using our DMXC Demo or Two Scene Preset programs. These programs require no installation and can be run off a CD or floppy if you wish.

Some information you may need to talk to our unit as a Plug-n-Play HID if you’re writing your own software:

```

Vendor ID:      0x0F48 (Durand Interstellar, Inc.)
Product ID:     0x4201
Product Version: 0x0141 (current firmware version)
Packet Size:    0x40
Device Class:   0
Subclass:       0
Sub-subclass:   0
Interfaces:     1
Type:           0x21 (HID)
Release:        0x0110 (USB version 1.1)
Country:        0 (USA)
Polling:        1 (1 mS)

ReportDscr:
    db 06h, 0A0h, 0FFh    ;; Usage Page (FFA0H = vendor defined)
    db 09h, 01h           ;; Usage (vendor defined)
    db 0A1h, 01h          ;; Collection (Application)
;; The input report
    db 09h, 01h           ;; Usage (vendor defined)
    db 15h, 00h           ;; Logical minimum (0)
    db 26h, 0FFh, 00h     ;; Logical maximum (255)
    db 75h, 08h           ;; Report size (8 bits)
    db 95h, 40h           ;; Report count (64 fields)
    db 81h, 02h           ;; Input (Data, variable, absolute)
;; The output report
    db 09h, 02h           ;; Usage (vendor defined)
    db 75h, 08h           ;; Report size (8 bits)
    db 95h, 40h           ;; Report count (64 fields)
    db 91h, 02h           ;; Output (Data, Variable, Absolute)
;; The feature report
    db 09h, 03h           ;; Usage Page (vendor defined)
    db 75h, 08h           ;; Report size (8 bits)
    db 95h, 05h           ;; Report count (5 fields)
    db 0B1h, 02h          ;; Output (Data, variable, absolute)
    db 0C0h              ;; End Collection

```


Data is passed to and from the host in packets. The packets FROM the host can be from 3 to 64 bytes long. Typically, they are fixed at 64 bytes. The format is:

Byte 0 Flags
 bit 0-3 reserved, always set to 0
 bit 3 1 = RESET (flush buffers)
 bit 4 1 = RTS true
 bit 5 1 = DTR true
 bit 6-7 reserved, always set to 0

Byte 1 Length (in bytes) of data to follow. Range 0-62.

Bytes 2-64 Data (standard ASCII as you would send out the serial port)

USB has its own handshaking built in. If the input buffer in the controller is full, an incoming packet will be automatically refused until there is room for it. This means that if the buffer is full the host can't even send a RESET command. A SETUP command will cause the converter to clear all buffers and abort and fade and/or script that was running. It's best to mind the CTS bit below.

Packets TO the host are ALWAYS 64 bytes long to keep Windows happy. They are sent whenever there is data in the output buffer and/or the host has changed any flag bits. Packets will NOT be sent after a SETUP or RESET until at least one packet has been received from the host. This prevents problems with the BIOS in some systems.

Byte 0 Flags
 bit 0-3 reserved (set to zero)
 bit 4 1 = CTS true
 bit 5 1 = DSR true (always 1)
 bit 6 1 = CD true (always 1)
 bit 7 1 = RI true (always 0)

Byte 1 Length (in bytes) of valid data to follow. Range 0-62.

Bytes 2-64 Data (standard ASCII as you would receive from the serial port)

Packets to the host are only transmitted after a polling command from the host. If, for some reason, the host stops polling (like it went into SLEEP mode), no packets will go out. If a script is trying to send data to the host, the buffer will eventually fill up and hang the converter unless handshaking in the converter has been turned off ("H 0"). With handshaking off, any time the output buffer fills up, additional characters are discarded until the host catches up. This might cause a loss of characters from commands with long responses (such as "I", "?", or scripts that print a lot).

Warranty

Durand Interstellar, Inc. warrants this product to be free from manufacturing defects in original material, including original parts, and workmanship under normal use and conditions (“manufacturing defect”) for a period of one (1) year from date of original purchase. A charge will be made for repairs not covered by the warranty.

Should service become necessary, contact Durand Interstellar, Inc. for return authorization and then:

- Pack the unit in a well-padded corrugated box
- Enclose a copy of your proof of purchase, if you are not the original purchaser
- Ship the unit prepaid via an insured carrier

NOTE: This warranty is void if the product is:

1. Damaged through negligence, misuse, abuse, or accident
2. Modified or repaired by anyone other than Durand Interstellar, Inc.
3. Damaged because it is improperly connected to other equipment of other manufacturers
4. Connected to any power source other than the adaptor supplied with the product

NOTE: This warranty does not cover:

5. Damage to equipment not properly connected to the product
6. Cost incurred in the shipping of the product to Durand Interstellar, Inc. to perform warranty repairs
7. Damage or improper operation of unit caused by customer abuse, misuse, negligence, or failure to follow operating instructions provided with the product
8. Ordinary adjustments to the product which can be performed by the customer as outlined in the instruction manual
9. EEPROM burnout after the first 60 days
10. Improper operation of the unit caused by software written by any third party

ANY APPLICABLE IMPLIED WARRANTIES, INCLUDING THE WARRANTY OF MERCHANTABILITY, ARE LIMITED IN DURATION TO THE PERIOD OF THE EXPRESSED WARRANTY AS PROVIDED HEREIN BEGINNING WITH THE DATE OF ORIGINAL PURCHASE AT RETAIL, AND NO WARRANTIES, WHETHER EXPRESS OR IMPLIED, SHALL APPLY TO THE PRODUCT THEREAFTER. DURAND INTERSTELLAR, INC. MAKES NO WARRANTY AS TO THE FITNESS OF THE PRODUCT FOR ANY PARTICULAR PURPOSE OR USE.

UNDER NO CIRCUMSTANCES SHALL DURAND INTERSTELLAR, INC. BE LIABLE FOR ANY LOSS, DIRECT, INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGE ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS PRODUCT.

THIS WARRANTY IS ONLY VALID IN THE UNITED STATES OF AMERICA. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. HOWEVER, YOU MAY HAVE OTHER RIGHTS WHICH MAY VARY FROM STATE TO STATE. SOME STATES DO NOT ALLOW LIMITATION ON IMPLIED WARRANTIES OR EXCLUSION OF CONSEQUENTIAL DAMAGE, THEREFORE THESE RESTRICTIONS MAY NOT APPLY TO YOU.