HypnoGadget Manual

Version 1.1, Released June 2011.

By

Chris Lomont

www.hypnocube.com

HypnoGadget User Manual

Contents

Version

Manual Version 1.1, Released June 2011

Updated to include Microchip CDC USB driver information.

Manual Version 1.0, Released March 2008.

Introduction

Welcome, and thanks for purchasing your HypnoCube or HypnoSquare! This manual will show some of the features of your device, collectively called HypnoGadgets.

Version 4.0 of the internal software supports the following features:

- Previous and Next visualization selection,
- Removal of undesired visualizations from the playlist,
- Changing speed setting on a per item and entire gadget basis,
- Connection with HyperTerminal using a USB cable to access advanced features,
- Connection through our HypnoCOMM program, accessing advanced features,
- Drawing on the device through external programs,
- Custom programming allowing unlimited drawing on the device through a USB cable (you must be able to program and use our supplied C++ code).

Kit Instructions

If you bought a kit, building instructions are in a separate document available from http://hypnocube.com/instructions/. If you purchased a completed device, then you avoid the pain and the fun of building one yourself ☺

Terminology

Here is some terminology useful for understanding how the gadget plays images.

- **Visualization** – A visualization is one of the main effects, of which there are over 50, tailored for the gadget. A visualization often has many random variations in color, symmetry, and subeffects, giving each visualization a huge range of actual imagery. Visualizations are played back in a predefined sequence by default, but this is user changeable to random order.
- **Transition** – A transition is a visual blending between visualizations, with such effects as sweep, alpha blending, flip, and more. By default they are played in random order, but this can be user changed to sequential.
- **Timing** – The playback speed of each visualization and transition varies randomly, with the time stretching back and forth between preset limits. Each

visualization has a range of preferred speeds, and these ranges are user selectable with the console mode access or through the HypnoCOMM program.

Gadget Commands

The gadget has 2 buttons and a power switch. The buttons are called button A and button B, and which is which can be determined during powerup as shown in Table 1.

Upon powering up the gadget, there is by default a power on sequence that allows checking each LED for power and color. The default sequence is a white image, followed by a single lit LED that scrolls through the LEDs.

Pressing any button while the single LED is scrolling skips the scroll.

LED testing

There is a power switch on the gadget, and two buttons called button A and button B. Holding down different combinations of the buttons results in different colors on power up, allowing testing of each LED as well as remembering which button is which. Table 1 shows the button effects.

Reset

Holding down both buttons during power on results in a red image. Holding both causes the device to blink. After a few blinks all internal settings will be **Reset**. This allows a reset in case changes you made to the settings are undesired.

Button(s) Down	Effect
Neither	White screen
Button A	Blue screen
Button B	Green Screen
Buttons A and B	Red Screen
Buttons A and B, hold	Red Screen, then blinking, then **Reset**

Table 1 – Power-on Button Effects

Command sequences

There are a few command sequences that are available while the device is running. This information may be superseded by the device itself, which, using the console program, will list the commands it knows. Currently at a minimum the commands in Table 2 are available in version 4.0 of the code.

In Table 2 a lowercase 'a' denotes pushing down the A button, and an uppercase 'A' denotes releasing the A button. Meanings are similar for 'b' and 'B'. For example, **Next**

is executed by pressing the A button down and releasing it, and this jumps to the next visualization.

Sequence	Command	Description
aA	**Next**	Jump to next visualization.
bB	**Prev**	Jump to previous visualization.
abAB	**Lock**	Lock to current visualization. Gadget shows a brief pause denoting command received, then continues running , never changing. Executing **Next** or **Prev** releases lock
baBA	**Pause**	Pauses current image, resulting in a still image. Executing **Next** or **Prev** releases pause.
abBbBA	**Remove**	Removes currently playing visualization from the playlist. Visualization can be reinstated through the console editor or through a hardware reset. It is recommended to **Lock** the visualization prior to removal to prevent accidently removing the wrong visualization.

Table 2 - Button Command Sequences

Communications

To connect the gadget to your PC to use HyperTerminal, HypnoCOMM, or to play back custom images, you need to install a driver.

Install USB Driver for 4Cube USB

To install the USB driver for the 8Square, see below.

As of December 2010, the 4Cube uses a second Microchip PIC to implement the USB functionality, replacing the expensive FTDI module.

Microchip's implementation makes use of Window's standard CDC drivers, but it still needs an .inf to identify it. It is also available from Microchip's website as part of their USB Framework but this is a rather large package (~100MB). You can download just the relevant bits from our site at http://hypnocube.com/Software/MicrochipCDC.zip.

Unzip the contents of the .zip file to a temporary directory. When you first plug in the 4Cube USB, Windows will prompt you with the "found new hardware" dialog box; point it to the mchpcdc.inf file to install.

I believe there are some driver implementations for Linux and OSX that will work, but that is outside the scope of this document.

Install USB Driver for 8Square USB

To install the USB driver for the 4Cube, see above.

The USB plug works through a USB to UART chip from FTDI. As a result, to connect to the device you need to install a driver for your computer. At the time of this writing the drivers were at http://www.ftdichip.com/Drivers/VCP.htm. The device is a FT232R, and supported operating systems include

- Windows XP, Vista, 7, Server 2003, Server 2008.
- Mac OS X
- Linux

NOTE: We have only tested on Windows XP/Vista/7 at this time!

Download the drivers from the FTDI site and install them. *Be sure to get the VCP driver, not the D2XX drivers!*

Once you have the USB drivers installed, plug in your gadget to a USB port on your computer, and power everything on. **Error! Reference source not found.** shows the new device added (in Windows XP Device Manager) when you have the HypnoGadget plugged in.
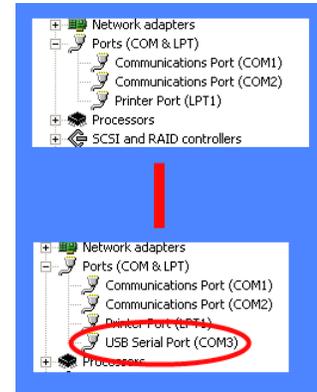

Figure 1 – New USB COM Port

Basic Communication

You can connect to the gadget through the COM port using standard serial terminals. HyperTerminal comes with Windows up to XP (but was removed in Vista). For Vista and 7, you can use Putty, which is a free, open-source telnet client[1]. Simply download the most recent version for Windows Vista or 7, and place it on your desktop.

HyperTerminal on Windows XP

To test communications with the gadget, open up HyperTerminal, under All Programs -> Accessories -> Communications -> HyperTerminal. If you are asked to make it your default Telnet program click No.

Then

1. If you are asked for Location Information, enter anything you desire.
2. If you are shown Phone and Modem options, press OK
3. On connection description, enter a name such as HypnoSquare. Press OK
4. On Connect To, select the highest numbered COM port. Press OK
5. Fill out COM Properties as in Figure 2. Be sure to enter Flow control None. Press OK.
6. Press the g key (lowercase). You should see something similar to Figure 3.
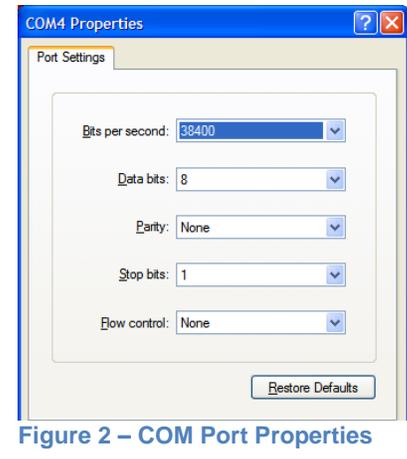

Figure 2 – COM Port Properties

If not, check you followed all the steps above correctly, and note any errors. If it does not work repeat the steps above using different COM ports.

---

[1]Putty is free from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html . Or Google it.
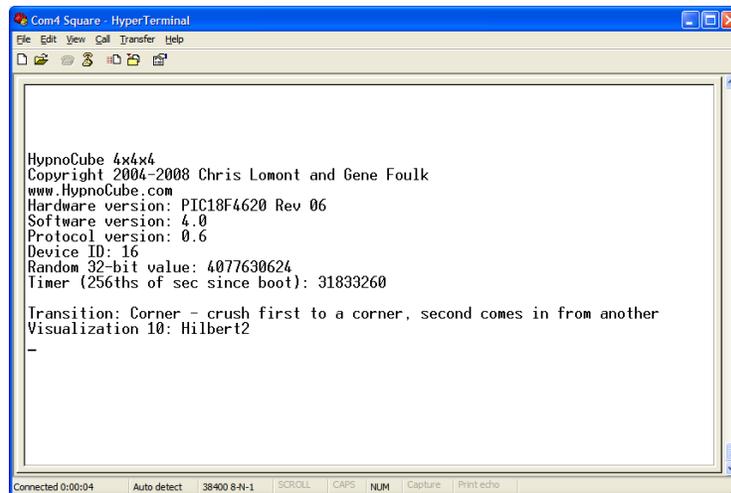
Figure 3 – HypnoGadget Terminal Info

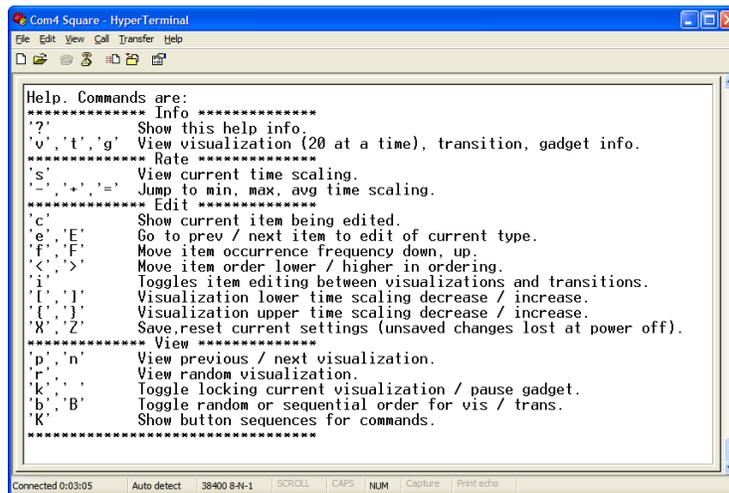## Putty on Windows Vista and Windows 7

When you run Putty, you get a configuration screen.

1. Create a new session, named HypnoGadget.
2. Select Serial. Set the parameters as in Figure 2. Also set a COM port (usually the highest numbered one on the system).
3. Return to session, select serial as the connection type.
4. Save your settings.
5. Select your session, and click Open.
6. Press the 'g' key to get Gadget Info. You should see a screen similar to Figure 3.

Console Usage

Once you have a connection, as the gadget runs, it prints out information to the terminal. It shows the names of the visualizations and transitions as they are selected for playback. You can get more information using keyboard commands.

Press the '?' key to get a list of commands recognized by the gadget. You will see something similar to Figure 4.

Figure 4 – HypnoGadget Commands

Try some. For example, pressing 'v' for visualizations will show info about the supported visualizations. Press it several times to see all the information.

All the commands in Table 3 – Console Mode CommandsTable 3 are supported, and possibly more. Some terminology:

- **Item** – either a transition or a visualization.
- **Count** – how many times this item has been played since last **Reset.**
- **Frequency** – how often this item is selected during random playback versus others of its type. For example, if all visualizations have frequency 0 except three with frequencies 1, 2, and 3, then there are 1+2+3 = 6 total "slots", so the first will play with frequency 1/6, the second with frequency 2/6 = 1/3, and the last with frequency 3/6 = ½. Thus one will play half the time, the other two not so often.
- **Speed Range** – each visualization has a speed range, which is the minimum and maximum desired speed for its playback.
- **Speed** – how fast a visualization moves is determined by its speed setting. This value is in 512ths of default speed, so if a visualization has a range of 256 to 1024, this means it is to run at 256/512 = ½ to 1024/512 = 2 times the default speed, giving it a range of half speed to double speed.

  However, suppose a visualization with a speed of 1024 is followed by one with a speed of 256. The second one does not immediately run at speed 256. The internal speed is slowly decreased to 256, ensuring smooth movement of items. This is by design and cannot be changed.

- **Random/Sequential Order** – both Visualizations and Transitions can be run in sequential (pre-programmed) or random order.
- **Save/Reset** – Changes to the settings **must** be saved to remain in effect after a power off. So after making the changes you want, be sure to save the settings!
- **Editor** – there is an editor allowing editing of these parameters from the HyperTerminal console. Pressing '?' for help, and experimenting with the commands is the easiest way to learn how to use it.

A brief list of the commands and what they do is in Table 3. An example from pressing 'v' to see visualization info is in Figure 5.

Figure 5 – Visualization Info

Command (keypress)	Explanation
?	Show a help screen with all supported commands
v	Show visualization info (20 or so at a time). This shows a **count** of how many times the visualization has been played since power up, its **frequency** (only used if in **random order**), and the minimum and maximum **speed range** if not on **global speed control**.
t	Show information on the transitions such as **frequency** and **count.**
g	Display some info about the gadget such as hardware, software, and protocol version, time since last power on, and copyright info.
s	Show current time scaling.
-	Set the **speed** to the minimum speed in the current speed range.
+	Set the **speed** to the maximum speed in the current range.
=	Set the **speed** to the middle of the current speed range.
c	Show current **item** being edited.
e	Go to the next **item** of the current type (visualization/transition) for editing.
E	Go to the previous **item** of the current type (visualization/transition) for editing.
f	Increase the current item **frequency** for playback (needs random ordering to have an effect).
F	Decrease the current item **frequency** for playback (needs random ordering to have an effect).
<	Move the current item higher in the ordering (UNSUPPORTED!).
>	Move the current item lower in the ordering (UNSUPPORTED!).
i	Toggle **item** editing between **visualizations** and **transitions**.
[	Decrease the lower range of the **speed** of the currently editing visualization.
]	Increase the lower range of the **speed** of the currently editing visualization.
{	Decrease the upper range of the **speed** of the currently editing visualization.
}	Increase the upper range of the **speed** of the currently editing visualization.
X	**Save** the current setting. This is needed to save settings past a power off/on cycle. The gadget will blink as the settings are saved back to EEPROM.
Z	**Reset** the settings to the factory defaults. Be sure to save 'X' after this.
p	Go to the previous visualization.
n	Go to the next visualization.
r	Go to a random visualization.
k	Toggle **locking** to the current visualization. When locked only the current visualization will play until unlocked.
(space)	Toggle **pausing** the current image.
b	Toggle random ordering for visualizations. Default is sequential (i.e., not random).
B	Toggle random ordering for transitions. Default is random (i.e., not sequential).
K	Show the button sequences for the button commands the gadget knows.

Table 3 – Console Mode Commands

HypnoCOMM

A much more powerful program, one we wrote and distribute with source code, is HypnoCOMM, a tool designed to interact with your gadget. It allows setting many internal options, saving your options to disk for easy restoring or copying to more gadgets, and incorporates the console mode from HyperTerminal. It even includes a simple drawing section so you can draw on your gadget using the mouse on your PC.

Currently HypnoCOMM only runs on Microsoft Windows, and requires the Microsoft .NET 2.0 runtime and Microsoft Visual C++ 2005 SP1 Redistributable Package installed. We do not plan to port it to any other platforms but do make the source code available. *NOTE that Windows Vista and 7 already has .NET 2.0 installed.*

Before running HypnoCOMM, be sure to close your terminal session if you still have it open. Otherwise there will be a conflict on the COM port when talking to the device.

To run HypnoCOMM, obtain it from our website (http://hypnocube.com/info/docs/). Extract the HypnoCOMM.exe file and place it anywhere. Your desktop is fine. Clicking on the program should run it.

If you get an error "The application has failed to start because the application configuration is incorrect…" or the error "the application failed to initialize properly" then you need to obtain the "Microsoft .NET Framework Version 2.0 Redistributable Package" and/or the "Microsoft Visual C++ 2005 SP1 Redistributable Package" from www.microsoft.com. Download it (usually dotnetfx.exe for .NET and vcredist_x86.exe for the 2005 runtime) and install it. *NOTE: you must get the SP1 version; the non-SP1 version will not work!*

Select the COM port (usually the highest numbered one), and select connect. Note you must have your gadget plugged in before running HypnoCOMM in order for it to detect the COM port. To see if it is connected, click your cursor into the large console pane, and press ? to get help. You should see some information scroll by. If there is no information, try clicking the Login button a few times, leaving it in the Logged **Out** state. The gadget may still think it is logged in from a previous session.

In short there are three sections to HypnoCOMM. The first, console, is identical to the HyperTerminal session, and only works when you are connected, but not logged in to the device. The other two modes (Drawing, Options) require logging in. Press the Login button now, and you should see the LOGIN message turn green. If not, try a few more times.

Errors

Due to the nature of the communications, sometimes there are errors in transmitting data to and from the gadget. When an error happens, it is logged in red in the errors

tab. If you get an error, redo the operation until there is none. In rare cases you might have to reboot the gadget and/or restart HypnoCOMM. There is a Clear button to clear the error log, making it easier to detect new errors.

Drawing

If logged in, the Draw tab becomes active. Items:

- Follow Image – check this to have the screen follow the image of the gadget. For the Cube it is currently not very clear to see what the cube is doing.
- Show Demo – click this to send some patterns from HypnoCOMM to the gadget.
- Drawing – Click a palette color on the right, and draw in the large black screen. This draws on the gadget. Note the colors do not match exactly due to much different color response between LEDs and monitors.

To restore the gadget playing its own images, you must Logout. Of course you can Login again.

Options

If logged in, the Options tab becomes active. Initially the only option active is the "Get Options" button, which gets the option setting from the device. This takes a few seconds. Make sure there were no Errors on obtaining the information.

Most of the option meanings are explained earlier, in the HyperTerminal section. New features here are

- Get Options – obtains options from the gadget.
- Save – saves options back to the gadget.
- Reset – Resets the options in the gadget.
- Export – save options (must be loaded from gadget first) to a file on your PC
- Import – load options from your PC to HypnoCOMM. You must save them to the gadget if you want them there.

When Save-ing the options to the gadget, make sure no Errors occurred, or the changes did not take effect. For most changes it is best to logout, and return to the console tab and check the settings changed using the keyboard commands. Remember to use the console ou must click the cursor into the pane.

Finally, see the programmer section for more information on technical details of HypnoCOMM. You can obtain the C++ .NET sourcecode for your own use and modification.

Programmer Information

This section is for programmers and other technical people who want to extend the capabilities of the HypnoGadget.

We supply C++ sourcecode to the HypnoCOMM program as well a demo program, HypnoDemo, showing how to programmatically draw on the gadgets. ***These are unsupported*** in the sense we are not going to teach you how to program C++, but by looking through the code anyone familiar with C++ should be able to program images on their gadget. If you want to develop code on your gadget look through these programs and be sure you understand what is involved. Our library makes it as easy as possible, but no easier than that ☺

Technical details

For fun, here are some statistics about the HypnoGadgets:

- The codesize on the gadget is about 28,000 lines of C and assembler.
- The codesize of HypnoCOMM is about 5,500 lines of C/C++.
- The PIC used on the gadgets is currently the microchip PIC18F4620.
- We use about 32500 bytes of program space out of 32768 on the PIC
- We use about 3.3K of the 3.9K of the RAM on the PIC.
- We use a UART to USB chip to talk USB to the gadget.
- The COM protocol is 38400 bps, 8N1, flow control None.
- The random number generator saves state between runs, and has period 2^512, making it very unlikely to ever repeat during the cube lifetime. Even using 1000 random numbers a second would take much longer than the age of the universe to repeat the sequence.

Protocol

There is a byte level protocol for sending and receiving data from the Gadget. Currently this is done through the USB driver, which looks like a COM port at the programming end in Windows. There is a chip on the gadget that converts the USB signals to a UART on the circuit board, and this is then read into the PIC through its UART.

This is the specification we developed when we wrote the code. If you want to implement this yourself you can, or you can use the C++ library we provide to talk to the device. The library makes it much easier to communicate with, especially if all you want to do is to play back images of your design.

```
HypnoGadget Commands and Packet Protocol 0.6 - by Chris Lomont - 2008

Data streams are sent back and forth between the gadget and an external device.
```

A command is broken into one or more packets. For example, to display an image requiring N bytes, the N bytes may need spread across multiple packets.

The gadget currently uses the serial communications for two modes, Console mode and Packet mode. By default it is in Console mode, allowing connecting with any serial terminal with correct settings. A successful Login changes Console mode to Packet mode, and a Logout changes back. The only Commands recognized in Console mode are Login and Ping. A Ping changes briefly out of console mode to allow the gadget to return an Ack command, and then Console mode is restored. Console mode is described elsewhere.

Packet synchronizing wrapper:
Define the SYNC character as byte value 192 (0xC0). Define the ESC character value as 219 (0xDB). Packets begin and end with a SYNC character, which can occur nowhere else. If the SYNC value is in a packet then it is replaced with the two byte sequence ESC, ESC+1. If ESC needs to be transmitted then it is replaced with the two byte sequence ESC, ESC+2. This is done on the entire packet including header, data, and checksum. It is undone at the other end before the packet is decoded.

Packet:
Inside the synchronizing wrapper is the packet, which consists of a 3 byte header, a variable length payload, and 2 byte checksum. The maximum length of a payload is 50 bytes, giving a maximum packet length of 55 bytes (before possible SYNC and ESC expansion). This length may increase in later versions.

Packet Structure:
(Length is in bytes. Values are stored big endian)
Offset   Length   Meaning
  0   1   Packet type high 3 bits, packet sequence 5 low bits
  1   1   Length of data 0-50
  2   1   Destination (0 = broadcast, else id of device)
  3   variable   Data
end-2   2   Checksum (CRC-16 CCITT)

Packet types are:
 2 - Data packet that is not last in sequence.
 3 - Data packet that is last in sequence.

Data packets representing one command start with a sequence from zero. If all requested data fits in one packet, it has type 3 and sequence 0, hence byte 1 is 011 00000 = 0x60. If a command spans more than 32 packets then wrap the sequence counter mod 32.

Data packets contain commands and data, and are covered below.

Any packet failing a length check, a checksum check, or sequencing information is discarded, and an Error command is returned.

Checksum:
The checksum is a 16 bit CRC-16 CCITT (using polynomial 0x1021). This is the same algorithm provided in the C++ Boost libraries (www.boost.org).

This is a standard CRC-16 with the following properties:
    * Width = 16 bits
    * Truncated polynomial = 0x1021
    * Initial value = 0xFFFF
    * Input data is NOT reflected
    * Output CRC is NOT reflected
    * No XOR is performed on the output CRC

```
Example values are (ASCII character set):
Message         CRC-16    Message Length (bytes)
 -None-         0xFFFF      0
   "A"          0xB915      1
"123456789"     0x29B1      9
```

Data:
Commands start with a 1 byte command, followed by necessary data.

Login:
To get access to the gadget, you must call the Login command (command 0) with
the challenge value of the device. By default the device challenge value is
hex 0xABADC0DE. A successful login will generate an ACK; an unsuccessful login
will generate an Error packet.

Requiring a login is done to give some minor security in case you want it.
It also locks the device from line noise. The challenge value is stored in
the gadget options, allowing changing it if desired.

Logout:
Logout from the gadget when done, using the Logout command. If the gadget is
set to require Pings then after a few seconds with no packets sent to the
gadget it will logout, and send a Logout command.

Ping:
You can Ping the gadget without logging in to allow checking the connection.

Usage:
A scenario for playback of user images is then:
1. Login.
2. Send image frame using SetFrame.
3. Flip image to show frame using FlipFrame.
4. Go to 2 as desired to playback images.
5. Logout to enable device returning to standard running state.

Commands:
A command is denoted by the first byte of the data stream, followed by any
necessary parameters. A command is not executed until all packets representing
the entire command are decoded correctly.

-> means command can be sent to gadget, <- means command may originate
from from gadget, <-> means packet can go both ways.
All other commands are reserved.

Implemented commands by protocol version:
0.1 : NONE
0.2 : NONE
0.3 : Login, Logout, Version, Ping, Ack, Error
0.4 : SetFrame, FlipFrame
0.5 : Reset, Options, GetError, Info
0.6 : MaxVisIdx,SelectVis,MaxTranIdx,SelectTran,GetFrame

Future?
0.7 : SetPixel,GetPixel,CurrentItem
0.8 : SetRate,DrawLine,DrawBox,FillImage
0.9 : SetPFrame,ScrollText,LoadAnim

Command Reference:

dir   Name   data packets

```
--------------------------------------------------------------------------
<->0  - Login      - send empty data, get back 4 bytes data, xor with
                     challenge value, send back. A valid Login returns an Ack
                     command. An invalid login returns an Error command, and
                     causes a 1 second delay before another Login will be
                     accepted from the gadget.
<->1  - Logout     - disconnect, data is empty. Sent by gadget if idle-time
                     exceeded (use Pings to keep connection alive).

 ->10 - Reset      - Reset device to default settings. Data is 0 bytes.
<->11 - Info       - send Info command with data, 0 terminated string returned
                     in an Info packet.
                   - 1st data byte:
                       0 - get gadget info item # j.
                       1 - get visualization # j info.
                       2 - get transition # j info.
                   - 2nd byte - index j of what to get (bounds checked).
                   - index out of bounds returns enpty string
                   - info text indices are:
                       0 - gadget name
                       1 - device description
                       2 - copyright information
                       ? - others possible in future
<->12 - Version    - send 0 data, receive version #s as bytes:
                     major hardware, minor hardware,
                     major software, minor software,
                     major protocol, minor protocol.
                     Other pairs may follow, so check length.
<->15 - Options    - send 0 data section to get options block sent back, send
                     options block to write options. There is a version field
                     in the Options block, and if incorrect, an Error is
                     returned from the gadget to prevent corruption.
                     Currently the only description of the options block is
                     in supplied header code.

<- 20 - Error      - various error codes and associated data. First byte:
                       0 = no error
                       1 = timeout - too long of a delay between packets
                       2 = missing packet, followed by missing sequence number.
                       3 = invalid checksum
                       4 = invalid type (2 and 3 defined for now).
                       5 = invalid sequence counter
                       6 = missing SYNC - SYNC out of order (2 SYNC in a row, for
                           example)
                       7 = invalid packet length
                       8 = invalid command
                       9 = invalid data (valid command)
                      10 = invalid ESC sequence - illegal byte after ESC byte.
                      11 = overflow - too much data was fed in with the packets.
                      12 = command not implemented (in case command deliberately
                           not allowed)
                      13 = invalid login value.
<->21 - GetError   - read error byte if data = 0, reset error byte if data = 1,
                     otherwise set internal error to value-2.
                   - error returned as a byte in a GetError command

<- 25 - Ack        - Each packet the gadget gets that doesn't generate a
                     response by protocol requirements sends back an
                     acknowledge. The data is the command byte, the checksum
                     from the packet received, then the final sequence counter
                     as a byte.
```

```
<->30 - CurrentItem- send 0 data, get string back about current items running.

<->40 - MaxVisIdx  - send 0 data, get back 1 byte max visualization index.
 ->41 - SelectVis  - data is 1 byte vis index, gets bounds checked and clamped.
<->42 - MaxTranIdx - send 0 data, get back 1 byte max transition index.
 ->43 - SelectTran - data is 1 byte tran index, gets bounds checked and clamped.

 ->50 - SetRate    - Set the serial connection rate (TODO).

 ->60 - Ping       - no data bytes past command byte needed. More bytes may be
                     used to change checksum. Sending a ping is needed
                     periodically to prevent auto logout after a few seconds.
                     Sending the gadget a ping will get an Ack response. May
                     be used even if not logged in to the gadget.

 ->80 - FlipFrame  - Flips the buffers, showing what was in the draw buffer.
 ->81 - SetFrame   - Fills a buffer using 96 bytes. See below.
<->82 - GetFrame   - Send with 0 data, get back frame of image data.
 ->83 - SetPFrame  - Send compressed frame (TODO)
 ->84 - SetPixel   - r,g,b,x,y,(z) bytes, dimension determined by size
<->85 - GetPixel   - x,y,(z). returns r,g,b bytes.
 ->86 - DrawLine   - r,g,b,x1,y1,(z1),x2,y2,(z2). (r,g,b in 0-255)
 ->87 - DrawBox    - r,g,b,x1,y1,(z1),x2,y2,(z2),filled (0,1).  (r,g,b 0-255)
 ->88 - FillImage  - Fill image with color, data is 3 bytes R,G,B (0-255)
 ->89 - ScrollText - TODO (allow colors, text?)
 ->90 - LoadAnim   - TODO (scriptable frames as anim?)

Drawing:
SetFrame: the HypnoSquare8 and HypnoCube4 image frame is 64 pixels, each
pixel taking 4 bits each of Red, Green, and Blue color. This is packed, so
3 bytes store 2 pixels as RGBRGB (=6*4=24 bits). Thus the 64 pixels are
stored in 64/2*3=96 bytes of data, so SetFrame and GetFrame have a data field
(past the command byte) of 96 bytes. SetFrame stops the gadget from drawing
its own images for a small time, allowing an external program to drive the
images. Once external images stop being sent and pings stop for a small time,
then the gadget takes over again (in case the connection was lost). This
keeps images in one form or another running.

Drawing too fast (over 30 fps) could bog down the device and cause it to
flicker or stop.

When implemented, the Square uses x,y coordinates, and the Cube uses x,y,z
coordinates. Future devices will use the appropriate number of coordinate as
needed. Commands should send the correct number of coordinates to the gadget,
otherwise the command will be tagged as an error. Colors will be 0-255 per
r,g,b channel except for packed frames.

Notes:
1. Setting a pixel and then getting it may return different values, since the
   RGB bytes may be truncated in the buffer as the color depth dictates.
2. Example: a broadcast all Ping with no extra data is bytes (including SYNCs)
   // TODO check!
   0xC0, 0x60, 0x01, 0x00, 0x3C, 0x1D, 0xFD, 0xC0

END OF PROTOCOL
```

Protocol Code

At the time this document was written, our library consists of the C++ files and headers shown in Table 4. Any other documentation will be with any sample programs we release. Also the HypnoCOMM program uses this library, and we are releasing the source for the HypnoCOMM.

File and header	Purpose
Gadget.cpp/Gadget.h	The main interface to the Hypnogadget and likely all you need to interface with.
Packet.cpp/Packet.h	The low level packet reading/writing code.
CRC16.cpp/CRC16.h	Checksum computation.
Command.h	Definitions of the commands for the gadget.
defines.h	Type defines to make a few files cross compile between the gadget and the library.
options.h	The current options layout for the gadget.

Table 4 – HypnoGadget Library C++ Files and Headers

We also offer a simple application with sourcecode, HypnoDemo, that demonstrates image playback.

Read through the sourcecode to the respective applications to understand the details of programming the HypnoGadget.

HypnoDemo

HypnoDemo is a small program demonstrating how to draw animations on the HypnoGadget. At the time of this writing code is shown for the 4x4x4 Cube and the 8x8 Square. Besides the files in Table 4, this program consists of HypnoDemo.h and HypnoDemo.cpp. The header file has a few classes implementing COM port access, and the cpp file demonstrates

1. Connect to the COM port,
2. Login into the gadget.
3. Draw animations using simple SetPixel calls to a buffer
4. Logout
5. Close the COM port.

END OF DOCUMENT