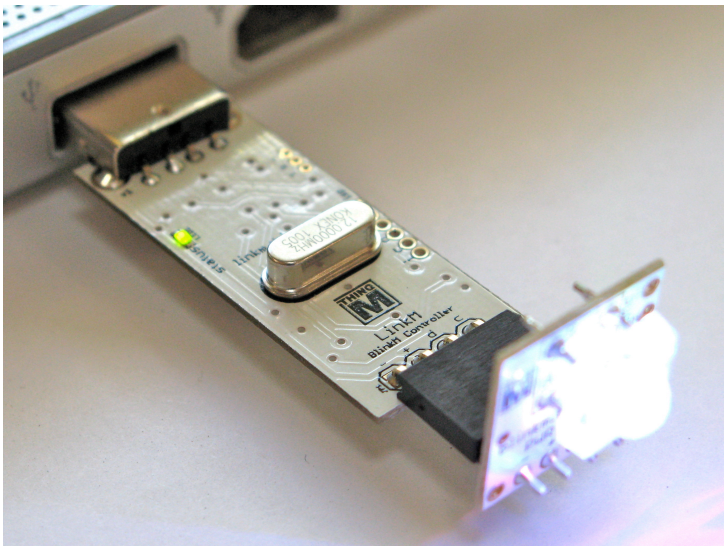
linkm.thingm.com

LINKM DATASHEET

Description

LinkM is a compact, inexpensive I2C interface designed for controlling and programming BlinkM Smart LEDs.



PRELIMINARY

Features

- USB HID class device / no devices drivers to install
- Directly power and control up to 8 BlinkMs or BlinkM MinMs from USB, or one BlinkM MaxM
- Hot-plug support of I2C devices
- Control up to 100 I2C devices with proper cabling
- Metronome to synchronize BlinkMs
- Entirely Open Source

Application Ideas

- Arduino-less programming of BlinkMs
- Computer-controlled RGB lighting
- Dynamic lighting for case mods
- Internet-triggered ambient colors
- Smart Christmas lights
- Embeddable control of I2C sensors

Table of Contents

Online Resources

The LinkM homepage is http://linkm.thingm.com/. You will find ready-to-run downloads, example code, and links to support forums.

The source code, example applications, firmware, and schematics to LinkM are hosted on Google Code at http://code.google.com/p/linkm/

The primary site for all things BlinkM is the BlinkM homepage: http://blinkm.thingm.com/. Additional, more DIY-oriented BlinkM information can be found at http://labs.thingm.com/.

1. Introduction

LinkM is a USB device that is designed to make it easy to control and program the BlinkM line of Smart LEDs. Since BlinkMs are just devices that speak the I2C serial protocol, LinkM is also a USB-to-I2C adapter and thus can be used with the many other I2C-based sensors, actuators, and other devices.

The USB protocol LinkM speaks is in the HID (Human Interface Device) class. This is the same protocol used by mice and keyboards and as such requires no special driver to be installed onto (and possibly crash) the host PC. And HID is supported by all OSes that support USB. LinkM includes examples that work with Mac OS X, Windows XP/7, and Ubuntu Linux, but can be made to work with other modern OSes with minimal effort.

LinkM is small enough to enable BlinkMs or other i2C devices to be used in space-constrained locations, making it useful for casemodding or embedded applications.

LinkM is designed to be low-cost and easy-to-use. It includes an I2C driver chip to enable longer I2C cable runs than what is normally achievable. But it is not designed to be a general solution for all I2C applications. Specifically, it does not support many of the various extensions and improvements to the base I2C protocol.

Every aspect of LinkM, from PC-side host software, to firmware, schematics, and end-user applications are open source and available for inspection at http://linkm.googlecode.com/.

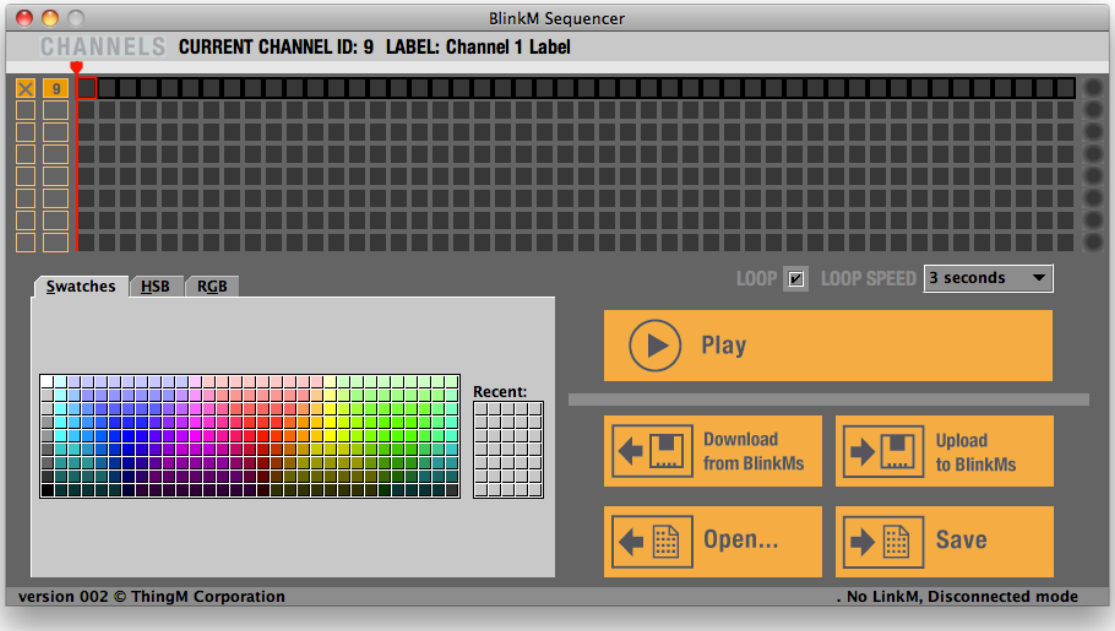
2. Getting Started

It's easy to get LinkM, a BlinkM, and your computer all working together.

2.1. Run BlinkMSequencer2

Download BlinkMSequencer2 for your OS from the sidebar at http://linkm.thingm.com/. Unzip it from its bundle and double-click on it. On start up, you'll see the main sequencer window as in Figure 2.1.

Figure 2.1: BlinkMSequencer2

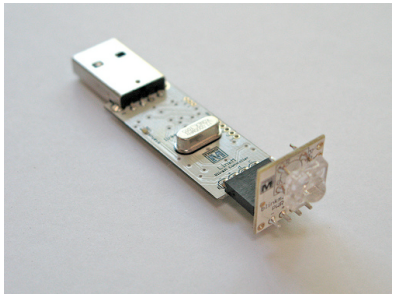


There are two halves to the interface. The top half is a grid where rows indicate different channels of BlinkMs and columns indicate different moments in time. You can change the color of each grid cell using the color picker in the lower left. To change multiple cells, click and drag then choose the color you want.

2.2. Connecting a BlinkM to a LinkM

Plug a BlinkM in as in Photo 2.1. The LinkM board is designed so that if a BlinkM is plugged correctly into it, the LinkM will lay flat with its top face (the one with the ThingM logo and the “linkm.thingm.com” URL facing up.

Photo 2.1: Plugging in a BlinkM

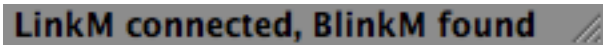


With a BlinkM plugged correctly into LinkM, plug LinkM into your computer.

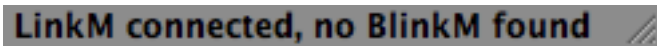
2.3 BlinkMSequencer2 Detects LinkM

Upon startup, BlinkMSequencer will automatically look for and connect to LinkM if it is plugged in. You do not have to explicitly connect and disconnect a BlinkM. While BlinkMSequencer is running, it will auto-detect a LinkM when it is plugged in.

When the sequencer detects both a LinkM and a BlinkM, it displays updates the lower-right status message area to show:



If LinkM is connected correctly, but the LinkM cannot detect any BlinkM, BlinkMSequencer2 will display the status message:



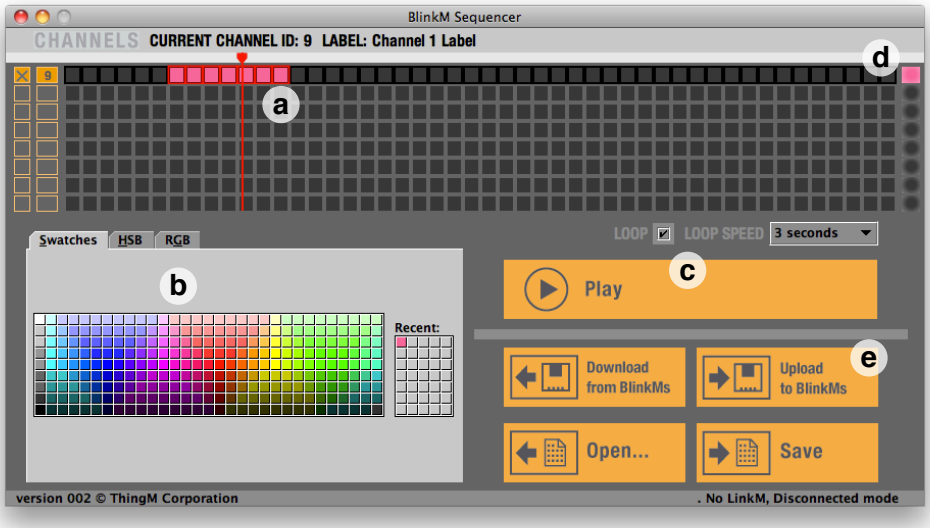
And if no LinkM is detected, it will display:

In Disconnected Mode, BlinkMSequencer2 is still fully usable for playing around with light sequences on the computer.

2.4. Making Color Sequences

If you have one BlinkM, the select cells only from the top row and pick colors for them with the color picker on the lower right, as in (a) & (b) in Figure 2.2. Press the “Play” button (c) to stop/start light playback. You’ll see your BlinkM’s color match the color preview spot on the right (d). Once you like your composition, press the “Upload to BlinkMs” (e) button to write it to your BlinkM. The BlinkM will now play back that pattern whenever the BlinkM is connected to power.

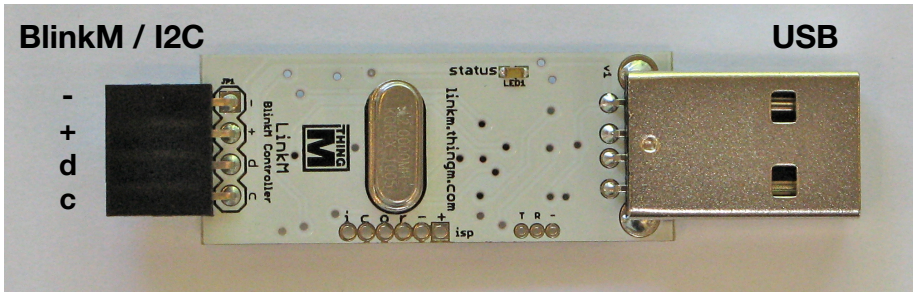
Figure 2.2: BlinkMSequencer2 color picking



3. LinkM Connections

As seen in Photo 3.1, LinkM only has two external connections: USB and BlinkM/ I2C. In normal use, these are the only two connectors that needed.

Photo 3.1: LinkM external connections



USB

The USB connection is a standard USB/1.0 Type A male connector. Make sure to plug LinkM into a powered hub that can supply up to 500mA.

I2C

The BlinkM / I2C connector is a 4-pin socket with the connections as shown in Table 3.1. This connector is designed to mate with the BlinkM line of devices so they can be plugged straight in with no wiring necessary.

Table 3.1: BlinkM connector

–	Ground, aka “Gnd”
+	+5VDC regulated power, 400mA max, aka “Vcc”
d	I2C data input/output, aka “SDA”
c	I2C clock input/output, aka “SCL”

4. Using LinkM

tbd

4.1. Connecting BlinkMs

When plugging BlinkMs, make sure the BlinkM is oriented as in the photos below. Plugging it in backwards could damage the BlinkM

1. Plug BlinkM into LinkM
    a. Make sure BlinkM is oriented as in the photos. Plugging it in backwards could damage the BlinkM
2. Plug LinkM into your computer
3. Run LinkM-enabled software

Always unplug LinkM before plugging in or unplugging BlinkM devices. The LinkM hardware supports hotplugging of I2C devices but the current host software doesn’t take advantage of it.

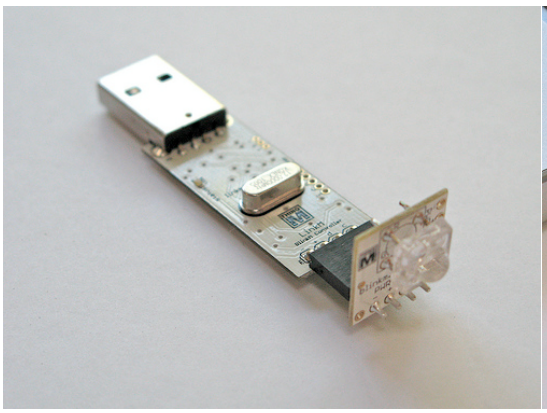


Figure 4.1: Plugging in BlinkM

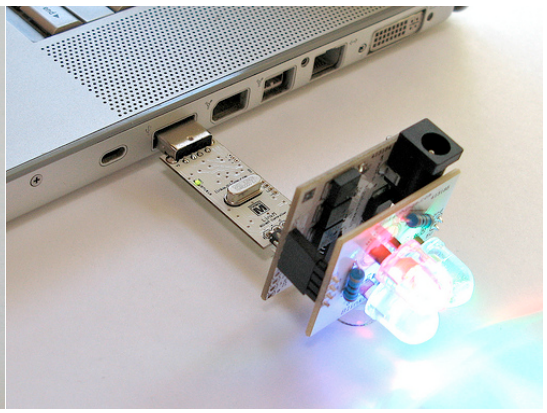


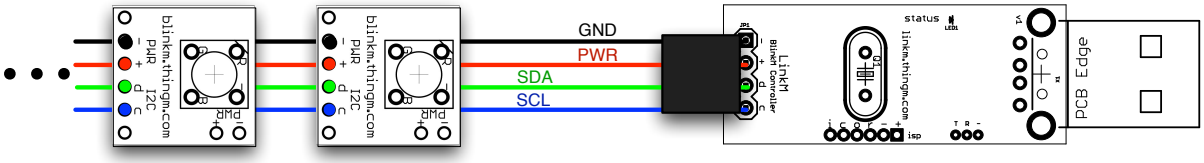
Figure 4.2: Plugging in BlinkM MaxM

4.2. Connecting Multiple BlinkMs

LinkM can control multiple BlinkMs simultaneously. The I2C protocol spoken by BlinkMs is inherently multi-device. Only four wires are needed to connect many devices. Connecting multiple devices just means wiring the same four wires to each device, as in Figure 4.2.

Controlling is not the same as powering the devices, however. LinkM can directly power up to 8 BlinkMs but can control many more than that.

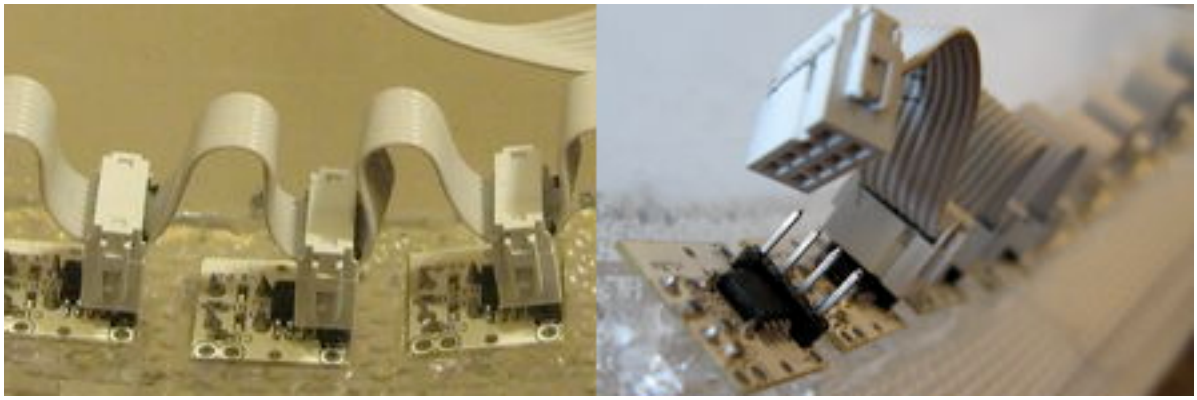
Figure 4.2. Connecting Multiple BlinkMs to LinkM



An easy way to make a multi-BlinkM cable is to use ribbon cable and 2x4-pin "IDC" crimp connectors as in Photo 4.2. These connectors require no soldering, just some pliers to crimp the connector around the cable.

Example part numbers for this setup are:
- 2×4-pin 0.1"-spacing IDC connector, Part number "FCI 71600-308LF" available from Mouser – "649-71600-308LF"
- Ribbon cable – standard 0.05″ spacing cable for IDC connectors, like Amphenol Spectra-Strip from Mouser

Photo 4.2: BlinkM I2C bus with IDC connectors and ribbon cable



[better image tbd]

4.3. BlinkM Power Capability

LinkM can draw at most 500mA from a USB port. A regular BlinkM or BlinkM MinM can draw at most 60mA. So the LinkM can directly power up to 8 regular BlinkMs or BlinkM MinMs.

A BlinkM MaxM normally draws up to 250mA, so LinkM can safely power one MaxM.

Using external power and proper cabling, a LinkM can control up to 100 BlinkM or other I2C devices.

4.4. Stand-alone Operation

LinkM has the ability to control and synchronize BlinkMs without a computer control. It contains a metronome that will periodically issue a sync command to all BlinkMs that are connected to it. This feature is called a “playticker” in the command-line tools and is currently only available via the command line.

Every time the metronome fires, it toggles the status LED.

To enable the metronome, issue the command:

```
% ./linkm-tool --playset "1,0,4,48"
% ./linkm-tool --linkmeesave
```

To disable the metronome, issue the command:

```
% ./linkm-tool --playset "0,0,4,48"
% ./linkm-tool --linkmeesave
```

To play back the default startup light script, issue the commands:

```
% ./linkm-tool --playset "1,0,50,6,0"
% ./linkm-tool --linkmeesave
```

5. Using BlinkMSequencer2

BlinkMSequencer2 is a multi-BlinkM light sequencer designed for use with BlinkMs. You can control up to eight different sets of BlinkMs

The sequencer will autodetect when a LinkM is plugged in or unplugged. It will also autodetect any BlinkMs plugged into LinkM.

1. Installing BlinkMSequencer

5.1. Programming one or more BlinkMs

tbd

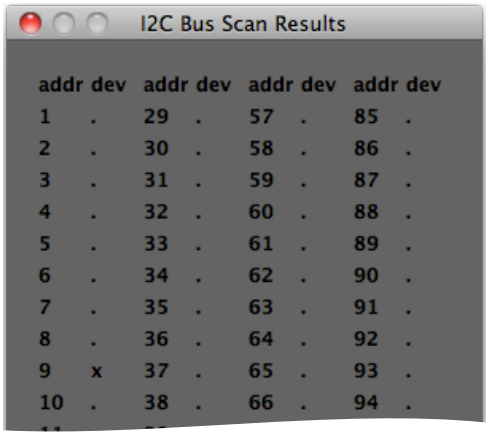
5.2. Setting I2C address of your BlinkM

tbd

5.3. Scanning I2C bus

tbd

Figure 5.3: Using LinkM as an I2C bus scanner



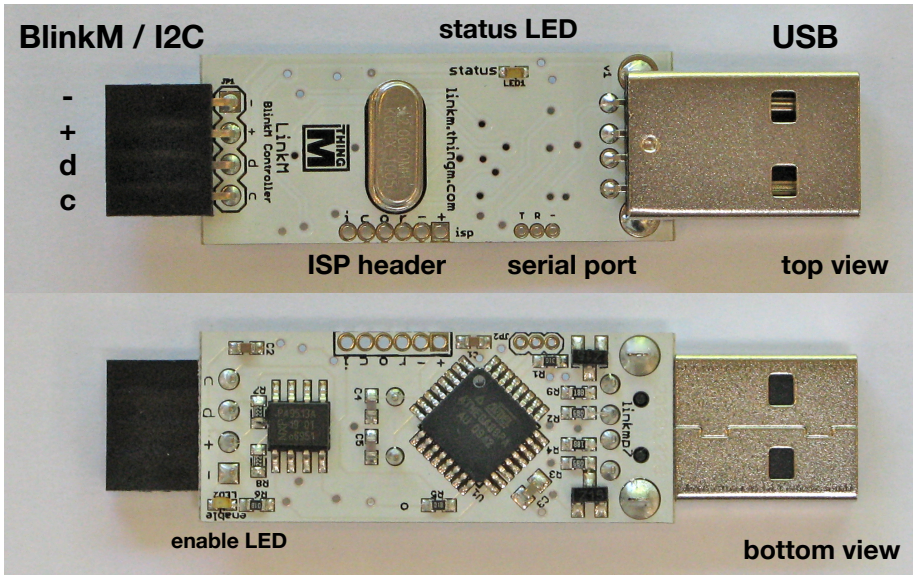
6. LinkM Hardware Internals

LinkM can be used as a general I2C adapter or an experimentation platform for AVR USB.

6.1. Hardware Overview

Besides the two external connectors, LinkM has some internal capabilities as well.

Photo 6.1: Internals



6.2. ISP header

The 6-pin ISP header is the standard AVR ISP programming connections, but in a 2.0mm pitch straight header. The pinout is described in Table 6.1. These pins can also be used to inspect the functioning of LinkM or be used for various hacking purposes.

Table 6.1: ISP header

+	+5VDC / “Vcc”
-	Ground / “Gnd”
r	RESET
o	MOSI / ATmega8 PB3 / bootloader enable
c	SCK / ATmega8 PB5 / enable LED
i	MISO / ATmega8 PB4 / status LED

6.3 Serial port

The UART TX and RX pins of the LinkM microcontroller are brought out to a separate header for use in diagnostic and hacking purposes. The pinout is descrbed in Table 6.2.

Table 6.2: Serial header

-	Ground / “Gnd”
R	Serial receive

T	Serial transmit

6.3. LEDs

There are two LEDs on LinkM. These LEDs are not easy to see if the LinkM board is in an enclosure. In normal operation with no commands being sent to LinkM, the “status” LED is off and the “enable” LED is on. In bootloader mode, the “status” LED is on and the “enable” LED is off.

Table 6.3: LEDs on LinkM

status	On top of LinkM board. Normally off. Lit when LinkM is processing a command. Can also be set programmatically with the statusLED() function. In bootloader mode, this LED stays lit.
enable	On bottom of LinkM board. Normally on. Indicates that I2C buffer chip is enabled. The LED (and the state of the I2C buffer chip) can be set with the i2cEnable() function.

6.2. LinkM Hardware Limitations

tbd
- no high-speed i2c, no 10-bit addrs
-

7. LinkM API

There are ThingM-provided APIs to communicate with LinkM in the following languages:

These APIs work on Mac OS X, Windows and Ubuntu Linux. These are all open-source and available from http://linkm.googlecode.com/.

- C
- Java
- Processing (which is really the Java API)

7.1 General Concepts

Both APIs are fundamentally very similar. At their core, each provides three kinds of functions:
- *linkm_open*
- *linkm_close*

- *linkm_command*

The *linkm_open* function searches for a plugged in LinkM and returns with an error code (in the C API) or an IOException (in the Java API) if no LinkM is found. At the moment, the API only supports a single LinkM.

The *linkm_close* function closes any resources used from a *linkm_open*.

The *linkm_command* function is the main entry point into LinkM. It is used to send data to LinkM, receive data from LinkM, or do a send/receive transaction with LinkM (the most common case). The first argument to this function is a command code that indicates the type of command to perform. The command codes are divided into two types: those that interact only with the LinkM hardware and those that perform an I2C operation.

From the C API file “linkm-lib.h”, those command codes are:

```
enum {
    LINKM_CMD_NONE     = 0,  // no command, do not use
    // I2C commands
    LINKM_CMD_I2CTRANS = 1,  // i2c read & write (N args: addr + other)
    LINKM_CMD_I2CWRITE = 2,  // i2c write to dev (N args: addr + other)
    LINKM_CMD_I2CREAD  = 3,  // i2c read         (1 args: addr)
    LINKM_CMD_I2CSCAN  = 4,  // i2c bus scan     (2 args: start,end)
    LINKM_CMD_I2CCONN  = 5,  // i2c connect/disc (1 args: 1/0)
    LINKM_CMD_I2CINIT  = 6,  // i2c init         (0 args: )

    // linkm board commands
    LINKM_CMD_VERSIONGET = 100,  // return linkm version
    LINKM_CMD_STATLEDSET = 101,  // status LED set   (1 args: 1/0)
    LINKM_CMD_STATLEDGET = 102,  // status LED get   (0 args)
    LINKM_CMD_PLAYSET    = 103,  // set params of player state machine
    LINKM_CMD_PLAYGET    = 104,  // get params of  player state machine
    LINKM_CMD_EESAVE     = 105,  // save linkm state to EEPROM
    LINKM_CMD_EELOAD     = 106,  // load linkm state from EEPROM
    LINKM_CMD_GOBOOTLOAD = 107,  // trigger USB bootload
};
```

The most common I2C-class command is LINKM_CMD_I2CTRANS.

7.2. Command-line Programs

tbd
There are two command-line programs available that test the Java and C APIs.
- c_host/linkm-tool
- java_host/linkm.sh

7.3. Java API

The Java API has a thin wrapper around the C API for the low-level methods for doing basic connecting and commanding. The Java API also contains mid-level methods for doing I2C and BlinkM inspection and communication. Additionally it contains high-level methods for loading and saving BlinkM light scripts. For full documentation on the Java API, see the Javadoc on the LinkM Google Code site.

A complete example of the Java API that fades any connected BlinkM to full-on white is:

```
import thingm.linkm.*;
LinkM linkm = new LinkM();
try {
  linkm.open();
  linkm.fadeToRGB( blinkmaddr, 255,255,255 );
  linkm.close();
} catch( IOException ioe ) {  println("Could not connect: " +ioe); }
```

Example low-level Java API methods:

```
public void open()
public native synchronized void command(int cmd,
                                        byte[] buf_send,
                                        byte[] buf_recv)
public native void close()
```

Example I2C-/BlinkM-level Java API methods:

```
public void commandi2c( byte[] buf_send, byte[] buf_recv )
public byte[] i2cScan(int start_addr, int end_addr)
public void i2cEnable(boolean state)
public void fadeToRGB(int addr, int r, int g, int b)
public void setFadeSpeed(int addr, int fadespeed)
public void playScript(int addr, int script_id, int reps, int pos)
```

Example high-level BlinkM API methods:

```
static final public String[] loadFile( File filename )
static final public boolean saveFile( File file, String scriptstr )
static final public BlinkMScript parseScript( String scriptstr )
```

7.4. C API

The C API is located in the “c_host” directory of the linkm project. It offers the minimum three functions: linkm_open(), linkm_close(), and linkm_command(). These functions are defined in the “linkm-lib.h” file.

The command-line application “linkm-tool” is a demonstration of the C API and is also useful inclusion in shell scripts and CGI scripts.

A complete example of the API that fades any connected BlinkM to full-on white is:

```
#include “linkm-lib.h”
usbDevice_t *dev;
int err;
if( (err = linkm_open( &dev )) ) {
  fprintf(stderr, "Error opening LinkM: %s\n", linkm_error_msg(err));
  exit(1);
}
uint8_t cmdbuf[8] = {blinkmaddr, ‘c’, 0xff,0xff,0xff};
if( (err = linkm_command(dev, LINKM_CMD_I2CTRANS, 5,0, cmdbuf, NULL)) {
  fprintf(stderr,"error on color cmd: %s\n",linkm_error_msg(err));
}
linkm_close(dev);
```

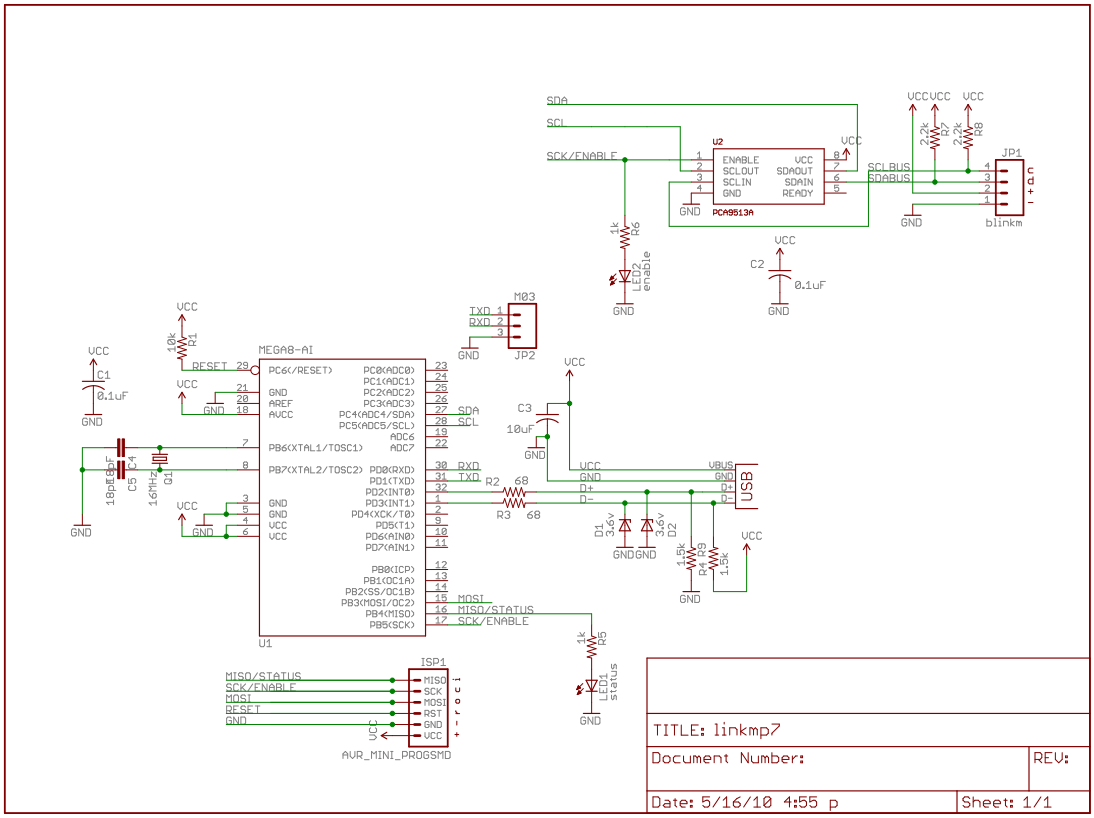
Specifically the API is:

```
int linkm_open(usbDevice_t** dev);
void linkm_close(usbDevice_t* dev);
int linkm_command(usbDevice_t* dev,
                  int cmd,
                  int bytes_send,
                  int bytes_recv,
                  uint8_t* buf_send,
                  uint8_t* buf_recv);
char* linkm_error_msg(int errCode);
```

The linkm_command() function is used to both send, receive, or both send and receive data to the LinkM.

8. Schematics

Figure 8.1: LinkM schematic



9. Packaging Information

Figure 9.1: LinkM PCB Packaging Information

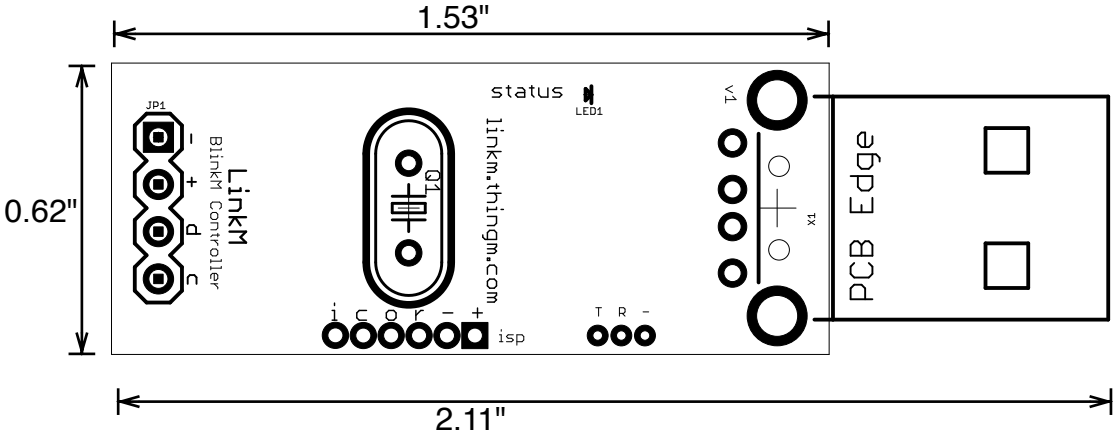


Figure 9.2: LinkM Enclosure Packing Information

[tbd]

10. Updates to this Datasheet

20100505 – initial release

Disclaimer

THINGM LABS

http://thingm.com/

1126 Palm Terrace
Pasadena, CA 91104

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

SparkFun Electronics:
COM-09903