LML33 Manual

Linux Media Labs

2002-08-03

Contents

1 Notational Conventions

You input is designated with $, for the command shell input prompt, and with # for superuser mode input. Your input and system response is presented in **bold face**.

2 System Requirements

- RedHat Linux 5.2, 6.0, 6.1,6.2, 7.1, 7.2 (other modern distributions should work as well)
- Kernel 2.0.34, 2.0.36, 2.2.X, 2.4.X
- CPU 200MHz or more
- RAM 32M or more
- EIDE or SCSI-2
- Any Video card
- Video overlay mode was tested with Matrox Millenium AGP G200 and other AGP cards may work

3 Miscelleneus

Because of the variations in Linux distributions, as well as because of quick rate of change in the shipping kernels Linux Media Labs provides drivers in source form, under the terms of Gnu General Public License (GNU GPL).

LML33 driver is being developed as part of generic Linux motion Video Streams infrastructure (LINVS). Therefore, we refer to LINVS driver and LINVS device when topics are not specific to LML33.

4 LML33 card

Install the LML33 card and reboot the system. Select Ignore when the new hardware added message is dispalyed while rebooting.

5 Linux System Configuraion

The system configuration can only be changed by the root (superuser), so you need to enter superuser mode:

$su -

After that you need to allocate memory for DMA RAM used by the board to store compressed JPEG video frames during video capture and playback. **Your LML33 card requires 2MB of memory to b‘e allocated for its use.**

If you have more then one LML33 card then you need 2Mb of RAM reserved per LML33 card.

Depending on the boot loader you use - LILO or GRUB (which is used by RedHat since revision 7.1) do the following:

5.1 LILO

For example if your system has 256Mb RAM then you need to edit the /etc/lilo.conf file by adding:

append = "mem=254M"

to the image you're booting from. If you have 96Mb you need to add 'mem=94M' line etc. Make certain if you have smp setup, that you use the smp section of lilo.conf, not the linux-up section.

Here is an example of /etc/lilo.conf file:

```
image=/boot/vmlinuz-2.0.34-0.6
label=linux
append = "mem=62M"
root=/dev/hda3 initrd=/boot/initrd-2.0.34-0.6.img read-only
```

*Please note, if you have more then one append string in your lilo.conf file, you must add your memory specifications to

the existing append string in order to avoid complications.

After that you **must** run 'lilo' command (to activate changes to your /etc/lilo.conf).

5.2 GRUB

Change the line in the /boot/grub.conf file that looks like :

kernel /boot/vmlinuz-2.4.7-10 ro root=/dev/hda1,ro

to

kernel /boot/vmlinuz-2.4.7-10 ro root=/dev/hda1,ro mem=250M

after reboot, that will reduce the memory size used by the kernel, leaving free memory needed by LinVS driver.

6 Linux Video Streaming Driver (LINVS v1.1x)

6.1 Obtaining the Software

Mount the CDROM.

 # mount /mnt/cdrom

#cd /tmp

Untar this file:

$ tar zxvf /mnt/cdrom/LML33/LML33driver1.1.*.tgz

This would create subdirecotry **LinVS** that contains low level video streaming driver.

Before using the driver for the first time you need to create compression video device nodes in the /dev directory, using script "mkvidnodes", which is located together with your driver in the LinVS directory.

 # cd /tmp/LinVS

#./mkvidnodes

That will make nodes:

- /dev/mvideo/status0
- /dev/mvideo/stream0
- /dev/mvideo/frame0
- /dev/mvideo/rawframe0
- /dev/mvideo/codec0
- /dev/mvideo/video4linux0

that correspond to the first card installed on the system. Nodes with numeric suffixes (like /dev/mvideo/stream2) corresopond to other LINVS cards installed on your system. Numeric suffix of 0 has aliases without defined suffixes (i.e. /dev/mvideo/status == /dev/mvideo/status0)

In order for the changes in the **conf** file to take effect **you must reboot** your system before proceding.

6.2 Compiling the Driver

Before proceding with compiling of the LML33 Driver Module, make certain that you have a symlink "linux" for your running kernel in /usr/src if not created already create the symlink by using the following command:

su -
#ln -s /usr/src/linux-... /usr/src/linux

Change directory to LinVS and make the loadable driver

 # cd /tmp/LinVS
 # make

The 'make' command would build the loadable driver object file: **lml33.o**

6.3 Loading the Driver Module

You may load LML33 driver after each reboot manually, or modify the /etc/conf.modules to let this happen automatically.

To manually load the driver:

$ su -

/sbin/insmod /tmp/LinVS/lml33.o *[optional insmod line parameters]*

Don't forget to exit superuser mode:

exit

At this point you should check the /dev/mvideo/status for installed driver information that should look simular to:

$cat /dev/mvideo/status0

LML33 driver version 1.1.*
Copyright (C) 1987-2000 Vassili Leonov and others.
LML33 driver comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under
certain conditions.
Check http://www.linuxmedialabs.com/lml33src for latest release.
high_memory=0x03f00000 nBoards=1 screen_width=1024 screen_height=768
vmem_base=0 bitrate=80000
board=0
bus=128
devFn=104
pciPhysBaseAddr=0xdd800000
lml33PHighMemory=0x03f00000
pciIrq= 9
bufferSize=0x1f000

Double check that **lml33PHighMemory** corresponds to your RAM size properly (i.e. to 62M is you have a 64M system etc.)

6.4 Configuring System to Load LML33 Driver Module Automatically

In order to load lml33.o module automatically you need to do the following:

su -

Add the line:

alias char-major-194 lml33

to file **/etc/conf.modules or /etc/modules.conf** in recent distributions.

For better video capture quality, add options line to the above said conf file

options lml33 video_stream=1 video_input=1 color_encoding=1 time_decimation=1 bitrate=250000

Copy file lml33.o to the location where it would be autoloadable:

cp /tmp/LinVS/lml33.o /lib/modules/<kernel>/misc

(substitute <kernel> with the directory corresponding to the kernel version you're running, or 'smp' if you're using dual CPU system, if there is no /misc directory, copy the file to /lib/modules/<kernel>/)

Note : if misc subdirectory is not available under **/lib/modules/<kernel> it should be created.**

To check the version of your current kernel, use:

#uname -r

Then procede to execute the following command:

 # depmod -a

now, you should be able to autoload the driver by just trying to read any of the /dev/mvideo/* devices:

$ cat /dev/mvideo/status

LML33 driver version 1.1.16

...

6.5 Testing the LML33 card

To capture a single jpeg frame in LML33 format:
$ dd if=/dev/mvideo/frame of=file.jpeg

To put a still image to LML33 analog video output**:**
$dd if=file.jpg of=/dev/mvideo/frame

To capture a video stream**:**
$ dd if=/dev/mvideo/stream of=filename.mjpg bs=50k

To playback a video stream
$ dd if=filename.mjpg of=/dev/mvideo/stream bs=50k

You can also use **cp** command instead of **dd**

6.6 Troubleshooting

If you run into any problems with the driver is helps to rebuild (and reinstall it) with debugging support turned on. Change to the driver directory and do:

$ make clean

$ make "CFLAGS = -DDEBUG=1 -O -Wall"

Change to superuser mode and enable generation of debug log by editing file /etc/syslog.conf to have destination for debug level messages, for example: *.=debug /var/log/debug. After that restart syslogd with the following command:

$ /etc/rc.d/init.d/syslog restart

Unload lml33 module, move lml33.o to the location you load this module from and reload lml33 module. You can now use:

$ tail -f /var/log/debug

to view debugging messages from the driver.

The above procedure is especially recommended is you're getting mysterious -EBUSY ("Device or resource busy") status from driver calls, /var/log/debug would contain extended explanation of the error.

7 File Conversion from LVMJPEG into AVI JPEG Format

We provide a script to convert LML33 MJPEG file to JPEG-AVI format. The script can be downloaded from http://linuxmedialabs.com/lml33src, the file name is 'mjpg2avi'.

There are quite a few viewing programs (like xanim) for AVI under Linux and other OSs (you need an add-on JPEG codec for AVI under MS WindowsXX, for example from MainConcepts).

In order to convert a file video.jpeg to video.avi use the command:

$ mjpg2avi video.jpeg video.avi

8 Obtaining and Installing Main Actor Software

This tool is bundled with LML33 and is not developed by Linux Media Labs.

Download the software from http://www.mainactor.com ("MainActor Video Editor for Linux") file name: MainActor-3.5.1-1.i386.rpm

$ su

to root and install it using rpm:

$ rpm -i MainActor-3.5.1-1.i386.rpm

Start the program using the command:

$ maseq&

Open top pull-down menu: "Help"/"Perform Registration" which brings a dialog box asking for a serial number.

Your serial number is: _____

(Please e-mail ma@linuxmedialabs.com if you did not receive the serial number).

Enter it and you now have a full version of MainActor installed on your system.

Please contact MainConcepts with problems and issues, their Web site is: http://www.mainactor.com

9 Linux Video MJPEG Stream Format

Linux Media Labs has defined a format for streaming motion JPEG video data named Linux Video MJPEG Stream (LMJPEG). The format is based on a sequence of JPEG frames (each JPEG frame contains one compressed image)(see Figure 1)

JPEG frames are placed into hardware JPEG Codec Frames (same as Code Buffers per Zoran terminology) and passed to hardware for decoding - or received from hardware on encoding.

For the interlaced video stream two JPEG frames are grouped together - so two images can be put into a Codec Frame. Resulting video signal has the **first** (odd) image transmitted first during the first field that gets painted **without vertical offset** on TV monitor, the **second** (even) image is transmitted as the second field and is painted with **half line offset down**(see Figure 2).

When hardware codec runs out of new Codec Frames during playback, it repeats the last one. This allows still intralaced picture to be displayed on TV monitor, with effective resolution up to 720x480 (for NTSC).

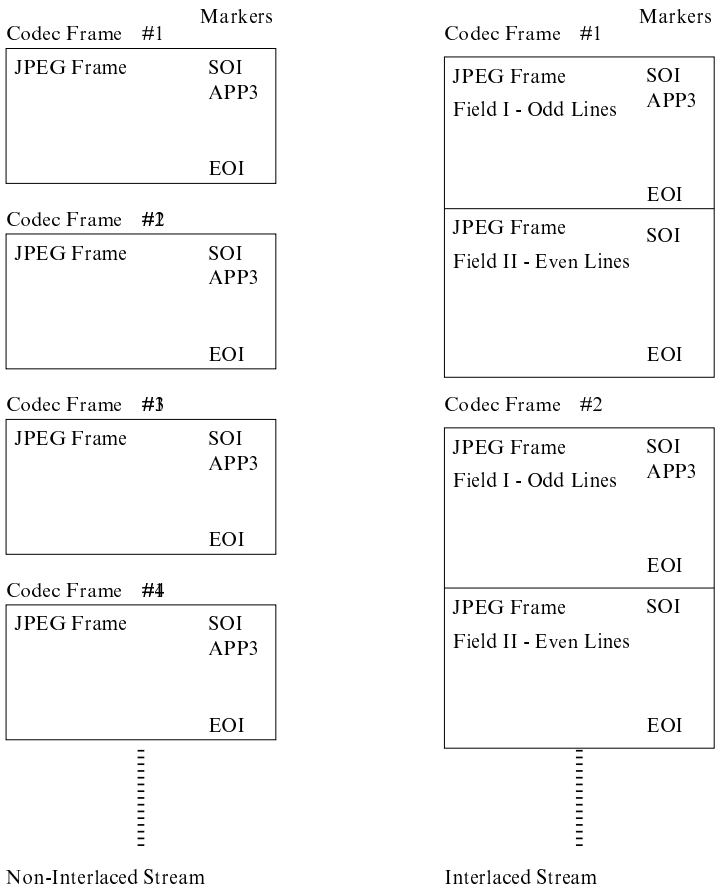

Figure 1: LMJPEG Stream Structure

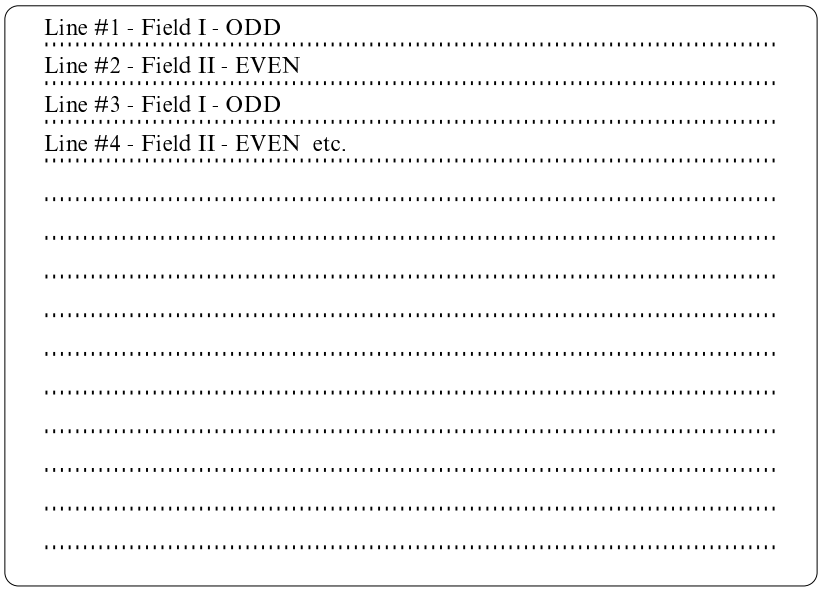

Figure 2: Interlaced TV raster

For non-interlaced LMJPEG stream one JPEG frame goes per Codec Frame. Each Image corresponding to JPEG frame is painted in the same vertical position on TV monitor, thus vertical resolution is half of the interlaced mode.

Every Codec Frame has an APP3 marker inside the first (odd) JPEG frame with information about the video stream parameters - or for non-interlaced stream inside every JPEG frame (since there is only one JPEG frame per Codec Frame)

The structure of the APP3 marker block (specific to LMJPEG format):

Data Size	Name	Explaination	typical value	Note
word	mrkAPP3	JPEG Application Comment #3	0xE3FF	
word	lenAPP3	Size of this block, without mrkAPP3	0x2C00	Big Endian
byte * 4	nm[]	Comment ID, must be set to "LML"	'LML',\0	
word	frameNo	Hardware generated frame seq. no		
dword	sec	Timestamp on the frame, seconds		
dword	usec	Timestamp on the frame, microsec.		
dword	frameSize	Size of Code Frame that follows		
dword	frameSeqNo	Driver generated CodeFrame seq.no		
dword	colorEncoding	1 - NTSC, 2 - PAL, 3 - SECAM	1	
dword	videoStream	1 - D1, 3-CIF, 4-QCIF	1	
word	timeDecimation	*See below*	1	
byte * 10	filler[]	Fill to total size of 0x30 (48)		

Note: byte is 8 bits in size, word is 16 bits, double word is 32 bits.

All data is Little Endian - unless specified as Big Endian.

Little Endian means less significant byte goes first, therefore is word value is 0xE3FF it's transmitted as 0xFF, 0xE3 at the byte sequence level - unless specified otherwise.

videoStream relates to the resolution in the captured JPEG frames. For NTSC D1 means 720x240, CIF means 360x240 and QCIF means 180x120.

timeDecimation works as follows - if it's set to 1, the stream goes at the incoming video signal rate (60fps for NTSC, 25fps for PAL). If it's set to 2 - only every second frame is left, i.e. 60fps turns into 30fps. If it's set to 3 then only every 3rd frame is left, i.e. 60fps turns into 20fps. If it's set to 4 we get 15 fps and so forth.

Here is an example of LML33 stream header *(with every other field printed in bold)*:

```
00000000 ff d8 ff e3 00 2c 4c 4d - 4c 00 00 00 91 9d f3 38 .....,LML......8
00000010 77 16 0b 00 56 c9 00 00 - 87 00 00 00 01 00 00 00 w...V...........
00000020 01 00 00 00 01 00 00 00 - 00 00 00 00 00 00 00 00 ................
```

10 LML33/DC10 Zoran Driver (V4L API driver)

There is another driver implementation available for LML33 which is using Video4Linux API with MJPEG extensions. This driver is more suitable for doing synchronized audio/video capture and playback. You can not use LinVS driver and LML33/DC10 Zoran driver at the same time.

In order to install LML33/DC10 driver you need to mount LMLCD, then use following commands:

```
cd /mnt/cdrom/LML33
./lml33install.sh
```

this would install the driver files, as well as make sure the driver is loaded at system startup.

If you want to stop using LML33/DC10 driver use the following command:

```
./lml33uninstall.sh
```

11 Creating Video-CD / SVCD using mjpegtools and VCDimager

In order to install video processing tools for Linux, mount LMLCD, then use the following commands:

cd /mnt/cdrom/contrib
./softwareInstall.sh

First step is to capture video into MJPEG encapsulated into AVI file format:

lavrec -d1 -fa -in -t 30 -q 80 test.avi

where -t switch sets the size of clip to be captured in seconds, so adjust it to the amount of seconds you want to capture

After that you can edit the resolution AVI file with MainActor video editor (maseq command) or another NLE for Linux, like Broadcast2000, Kino etc.

Second step is to separate audio stream from video stream and encode it into MP2 audio stream:

lav2wav test.avi | mp2enc -V -o aud.mp2

Third step is to separate video stream and encode it into MPEG1 (for VCD) or MPEG2 (for SVCD). For VCD (352x240 resolution, 30 frames/sec) use the following command:

lav2yuv test.avi | yuvscaler -o VCD | mpeg2enc -s -r 16 -o video.m1v

For SVCD (480x480 resolution, 60 fields/sec interlaced) use the following command:

lav2yuv -x -s 2 test.avi | yuvscaler -O SVCD | mpeg2enc -M 2 -F 3 -s -b 2500 -V 400 -o video.m2v

Forth step is to combine audio and video streams, for VCD:

mplex -f 1 aud.mp2 video.m1v -o video.m1s

or for SVCD:

mplex -f 2 aud.mp2 video.m2v -o video.m2s

Fifth step is to create a VCD or SVCD image that can be written to CD-R or CD-RW disk, for VCD:

vcdimager video.m1s

or for SVCD

vcdimager -t svcd video.m2s

This command creates two files: videocd.bin and videocd.cue. In order to produce CD-R or CD-RW you need to use the following command:

cdrdao write -device 0,0,0 videocd.cue

at this point you will get a disk that can be played with any software VCD/SVCD player or standalone VCD/SVCD player as well as most DVD players would play VCDs.

In order to check your clip or use computer for watching the clip use the following command:

mplayer video.m2s

In order to playback captured AVI, with LML33 composite video output use the following command:

lavplay -p C test.avi

All tools mentioned have man pages with a lot of additional information on switches and modes. For example use 'man lavrec' command to find more about video capture utility.

12 Estimating Disk Drive Performance

In order to estimate the performance of the disk subsystem use the following command:

$ time dd if=/dev/zero of=/tmp/dummy.test bs=1024k count=100

Make sure you have at least 100Mb free on a disk drive mounted on /tmp. If you have more space available on /tmp, change to count=500 or more. Then calculate the transfer rate in Mb/sec.

For example:

```
time dd if=/dev/zero of=/tmp/dummy.test bs=1024k count=120
120+0 records in
120+0 records out
0.01user 8.99system 0:40.16elapsed 22%CPU (0avgtext+0avgdata
0maxresident)
0inputs+0outputs (199major+271minor)pagefaults 6swaps
```

That gives 120Mb / 40sec = 3 Mb/sec

If you keep in mind that a video stream of 720x480@30fps in 4:2:2 color with 1:15 compression is about 1.4 Mb/sec you can estimate if your disk subsystem has enough performance to record/playback video without frames dropped.

13 Frequently Asked Questions (FAQ)

Q: Problems inserting module with SMP kernels:

A:

1. Add #include <linux/modversions.h> to std_module.h.

2. Re-compile the kernel from source! This is necessary because the RedHat kernel/module 'include' files are _not_ suitable for compiling SMP driver/modules... (there should really be a "kernel-headers-SMP. ..rpm") You need to make sure that the option for "set version information on all symbols for modules" is set to YES.. In fact all the three options in the section "loadable module support" should be set to YES... (using make xconfig).

Q: Can not fine modversions.h file during LINVS driver compilatons

A: Make sure you have kernel sources installed. Run 'make dep' from /usr/src/linux (or another location where kernel sources are installd on a non RedHat setup).

14 Depreciated

14.1 Tcl based drivers (v0.x)

14.1.1 Obtaining the Software

From http://linuxmedialabs.com/lml33src download the files lml33driver.tgz, lml33tcl.tgz. (or lml33tcl_pal.tgz for the PAL version). For example you can do it with the 'lynx' browser (assuming you have you computer connected to the Internet) :

$lynx -dump http://linuxmedialabs.com/lml33src/lml33driver.tgz > lml33driver.tgz

$lynx -dump http://linuxmedialabs.com/lml33src/lml33tcl.tgz > lml33tcl.tgz

Untar these files:

$ tar zxvf lml33driver.tgz

$ tar zxvf lml33tcl.tgz

That would create following subdirectories :

- lml/lml33driver

This directory contains a set of source files to build a loadable software driver for LML33 video capture board. After finishing the installation you would get a set of devices in the /dev directory for various functions associated with the board.

- lml/lml33tcl
- lml/lml33tcl_PAL (for the PAL version)

This directory contains a set of source files to build an extension to Tcl/Tk interpreter (named lml33tcl and lml33tk). Commands are added to manipulate LML33 registers, virtual memory and bus mastering (DMA) data transfers from LML33 board to the system RAM.

14.1.2 Compiling the Driver

Because of the variations in Linux distributions, as well as because of quick rate of change in the shipping kernels Linux Media Labs provides drivers in source form, under the terms of Gnu General Public License (GNU GPL).

Change directory to lml/driver and make the loadable driver:

$ cd lml/driver

$ make

The 'make' command would build the loadable driver object file: **lml33.o**

14.1.3 Loading the Driver Module

You may load LML33 driver after each reboot manually, or modify the /etc/conf.modules to let this happen automatically.

To manually load the driver:

$ su

/sbin/insmod lml33.o

Don't forget to exit superuser mode:

exit

At this point you should check the /dev/jvideostat for installed driver information that should look simular to:

$cat /dev/videostat

```
LML33 driver version LML33 v0.2
Copyright (C) 1987-1988 Vassili Leonov and others.
LML33 driver comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under
certain conditions.
pciPhysBaseAddr=0xe4000000
bus=0 devFn=72
pciIrq=12
high_memory=0x1f00000
lml33PHighMemory=0x1f00000
```

Double check that **high_memory** corresponds to your RAM size properly (i.e. to 31M is you have a 32M system)

14.1.4 Tcl/Tk LML33 Extension and Configuration Scripts Insallation

Change directory to lml/lml33tcl and make the interpreter:

$ cd lml/lml33tcl

$ make

The 'make' command would build two executable files: **lml33tcl** and **lml33tk**

Then su to root and run ./suid script to make interpreters suid root, so that they can do IO Memory mapping :

$ su

./suid

don't forget to exit superuser mode:

exit

the script 'suid' performs the following:

```
chown root lml33tcl lml33tk
chmod +s lml33tcl lml33tk
```

Please be aware of the security implications! These scripts give root access to anybody who has execute permissions for them.

14.1.5 Testing the LML33 Card and Software Setup

From the lml33tcl directory you can run:

- ./inii22.tcl <- to initialize the board.
- ./checki2c.tcl <- this should test chip id's and print TEST OK
- ./bars.tcl <- this should produce color bars on the video output
- ./playback.tcl <- still plays the included test bars captured image back
- ./capture.tcl <- make sure there is some video input
- ./save.tcl <- this should produce 4 files, and should print their sizes
- ./playback.tcl <- still plays your captured image

14.1.6 Using the v0.x Driver

Run capture.tcl and get to 'compression active' message. Then you can use:
$ dd if=/dev/jvideo of=filename.mjpg bs=50k
to capture a video stream

To capture a single jpeg frame in LML33 format:
$ dd if=/dev/jframe of=file.jpeg

Run playback.tcl and get to 'decompression active' message. After that you can use:
$ dd if=filename.mjpg of=/dev/jvideo
to playback a video stream

And use:
$dd if=file.jpg of=/dev/jframe
to put a still image to LML33 analog video output.

You can also use **cp** command instead of **dd**

14.1.7 Tcl/Tk LML33 Extension Commands

In addition to all usual Tcl commands the following are added, with the capability to read hex numbers if they are prefixed with 0x (as in C), i.e. you can say for example **readl 0x1F00000**

iopl gives you access to all I/O ports

inb <portAddr> read byte from I/O port

readx <memAddr> read byte, word or long (32bit) (readb,readv,readl) from a virtual memory location mapped to zr36067 registers.

writex <memAddr> <value> same for write operation (as in readx)

setbit067 <portAddr> <bitNo> <value 0 or 1> set zr36067 register bit to a value

mmap <memAddr> <size> <vmemAddr> mmap certain physical memory address region of <size> to a desired virtual address, specify 0 if you don't care. Sometimes that's the only way to map - since it would fail otherwise.

mmapLML33 maps physical range of addressed belonging to LML33 to a virtual memory address and maps the area at 31M (for code buffers). These addreses are returned by this command as a list.

gpio <bit no> <bit value> set gpio bit to a value

i2c_probe <addr> probe i2c bus address for a device present. Only 0x88 and 0x8a are used be LML33

i2c_read856 read BT856 chip ID register. Nothing else can be read from BT856.

i2c_read819 <sub addr> read value from BT819 subaddress

i2c_writeXXX <sub addr> <value> i2c_write856 or i2c_write819, read register value from a given subaddress.

i2c_setbitXXX <sub addr> <bitNo> <bit value 0 or 1> i2c_setbit856, i2c_setbit819, set a bit to a value. For 856 only registers at 0xDA, 0xDC and 0xDE are supported.

po_read <guest ID> <guest subaddr> reads a guest register

i2c_probe <addr> probe i2c bus address for a device present. Only 0x88 and 0x8a are used be LML33

i2c_read856 read BT856 chip ID register. Nothing else can be read from BT856.

i2c_read819 <sub addr> read value from BT819 subaddress

i2c_writeXXX <sub addr> <value> i2c_write856 or i2c_write819, read register value from a given subaddress.

i2c_setbitXXX <sub addr> <bitNo> <bit value 0 or 1> i2c_setbit856, i2c_setbit819, set a bit to a value. For 856 only registers at 0xDA, 0xDC and 0xDE are supported.

po_read <guest ID> <guest subaddr> reads a guest register