

Chumbot: Introduction to the Chumby

By Bob Smith

Chumby One Overview

This article describes how to make a Chumby One into a great robot computer. You'll see how to ssh into the Chumby, how its boot sequence works, and how to install a custom boot sequence to control your robot. This article covers how to install a native ARM GCC toolchain and how to install both Java and Python. You'll also see a hands-on introduction to the peripherals built into a Chumby. Much of the information in this article is taken from the Chumby forums, wikis, and web site which are all referenced at the end of the article.



Photo 1: The Default Clock Application

FEATURES: The Chumby One is basically an Internet radio and alarm clock that uses Flash for all user interfaces and applications. Photo 1 shows a Chumby One running the default clock application. The Flash applications are called “widgets”, and are downloaded from the Chumby web site after you create an account to specify which widgets you want to run. The Chumby is a pretty good alarm clock

- Full 32-bit Linux with all source code
- \$80 to \$100
- Great documentation: forums, wiki, web site with source code and schematics
- Non-brickable design with root on Micro-SD
- Programming languages include Shell, Perl, Flash, GCC, Java
- User interfaces include touchscreen/LCD or web pages
- Input/Output via USB 2.0 host or 3.3 Volt serial (internal)
- 5 volt external supply or internal 3.7 volt Li-ion battery

Table 1: Chumby One Features

so one of our goals is to make it easy to switch from “Chumby mode” to “robot mode” and back again.

As Table 1 shows the Chumby One has an impressive set of features for a robot computer that costs about \$100. It has full 32-bit Linux with all source code and it is exceptionally well documented on the Chumby forum, wiki, and web site. The Linux kernel and root file system are on a Micro-SD that is easily copied, modified or replaced. (One of the stated design goals of the Chumby is that it be easy to hack by people like us.) You can program the Chumby with Flash, Perl, C/C++, Java, or with shell commands. User interfaces are relatively easy to build since the Chumby has both a built-in web server and an LCD display with a touchscreen. You can connect to your robot using the external USB 2.0 host port or using an internal 3.3 volt serial port. The Chumby One runs on 5 volts making it easy to build your robot with a single battery pack. It also has a charger and a compartment for a NP-120 Lithium-ion battery.

SPECIFICATIONS: The Chumby One is an ARM based Linux computer with 64 MB of RAM and a one or two gigabyte Micro-SD flash disk for the root file system. It has a 320 by 240 LCD display with 16 bit color, a dimmable backlight, and a touchscreen. Audio components include a microphone, 2 Watt mono amplifier and speaker, and a stereo headphones jack. There is a rotary quadrature encoder with a large knob on the side and a crash-bar style push button on top. There is a 3-axis accelerometer

- ARM @ 464 MHz, 64 MB RAM, (i.MX233, no FPU)
- 2GB Micro-SD (1 GB works too)
- 320x240 @ 16 bits LCD with touchscreen and backlight
- WiFi (Ralink RT73, internal USB)
- 2W Speaker, stereo headphones, microphone
- 3-axis accelerometer (MMA7455)
- Quadrature knob and crash-bar top switch
- Video out (internal) 720x480

Table 2: Chumby One Specifications

that is a MMA74553 from Freescale Semiconductor. Table 2 is a summary of the Chumby One specifications. See the Chumby web site for a more complete list of specifications.

POWER BUDGET: The power requirement of a Chumby One are low enough to make it possible to convert a small AVR or PIC based robot to Linux with a minor upgrade to the battery system. Table 3 shows the current requirements (at 5 volts) for the core of a Chumby One and several common configurations.

- +260 ma CPU Board at maximum CPU load
- +110 ma BaseBoard4 w/9 peripherals
- +290 ma RA73 WiFi at minimum load
- + 30 ma LCD backlight at 50% (110 ma @ 100%)

Table 3: Chumby One Power Budget

Turning a Chumby One into a Robot Computer

This section of the article is intended to be “hands-on”, so you should have your Chumby One powered up and connected to the Internet. You should also have access to a Linux computer that is on the same network as the Chumby. You're about to void the warranty on your Chumby. That's not a problem for you, right?

BACKUP: The first step in turning a Chumby into a robot computer is to back up the Chumby software that came with the Chumby. You can skip this step but you do so at your own peril.

- Remove the four screws that hold the Chumby One together by first removing the pads that cover the screw access holes. See Photo 2.
- Slide the bottom cover back and move it to the side to expose the Micro-SD cage. See Photo 3.
- Slide the cage cover down and out to unlock the cage and remove the Micro-SD card.
- Mount the Micro-SD card in a USB card reader and connect it to your Linux computer.
- If the card reader connects as `sdb` then back up the card with the command:

```
dd if=/dev/sdb of=Chumby_Disk.img
```
- Put the Chumby Micro-SD in a safe place and put a new Micro-SD (> 1GB) in the card reader and copy the Chumby image onto it. Assuming again that it connects as `sdb`, the command is:

```
dd if=Chumby_Disk.img of=/dev/sdb
```



Photo 2: Screw Access Holes



BOOT PROCESS: The boot process for the Chumby is pretty typical of many embedded Linux systems. The bootloader starts Linux which runs init which runs /etc/init.d/rcS which, on the Chumby, runs /etc/init.d/rcS.background. The Chumby developers really do want it to be hacked and they put in the following switch at exactly the right point in the boot process to decide whether to boot as a Flash player (“Chumby mode”) or as some other device (“robot mode”):

```
if [ -e /psp/rcS -a -x /psp/rcS ]
then
    exec /psp/rcS
fi
```

The root partition is mounted read-only but both /psp and /mnt/storage are separate partitions that are mounted read-write. To make your Chumby into a robot computer all you have to do is install a custom /psp/rcS file that runs your robot programs. A tarball that we provide (see below) has a version of /psp/rcS that switches one more time to decide whether to continue as a Flash player or to exec /mnt/storage/bin/rcS.robot.

One final note on the boot sequence is to note that if you turn on the Chumby while holding your finger on the touchscreen the Chumby enter a “recovery” mode. This is really useful when you move your robot to a new WiFi area and have to use Chumby mode to configure it for the new WiFi link.

SSH LOGIN: The Chumby ships with an ssh server installed but disabled. The steps to enable it the first time are as follows:

- Press the top bar to go to the Control Panel , then tap on Settings
- Tap on Chumby Info and note the IP address
- Tap on the tiny Pi symbol in the top right
- Tap on SSHD to activate SSH daemon (until reboot)
- ssh to the Chumby as user “root” with no password
- touch /psp/start_sshd # to have SSHD at start up

After logging in do a **ps** or a **top** on the Chumby to see what processes are running. Execute the following two commands to exit “Chumby mode” and have the Chumby become a small, simple, embedded Linux computer.

```
/usr/chumby/scripts/stop_control_panel
killall chumbradiod cpid ntpd crond chumbhowld chumbalarmd
```

Do another **ps** or **top** to see the difference. To make this change permanent you need to install a versions of /psp/rcS that does not run the Flash player.

We have prepared a tarball with some extra Busybox utilities, and a version of /psp/rcS that let's you easily switch between Chumby mode and robot mode. Issue the following commands at the Chumby shell prompt:

```
cd /mnt/storage
wget http://www.linuxtoys.org/chumbot/chumby_robot.tgz
tar -xzf chumby_robot.tgz
rm chumby_robot.tgz
mv rcS /psp
mount -o remount,rw /
passwd root
```

```
mount -o remount,ro /
touch /mnt/storage/be_a_robot # rm this to be a Flash player again
(sync ; sleep 5; reboot) &
exit
```

The above shows how to make the root files system read-write and how to set a root password. It is a good idea to remount the root file system as read-only as quickly as possible after making changes to it. The file `be_a_robot` is a switch used by `/psp/rcS` to indicate whether to finish booting as a Flash-playing Chumby or as a robot Chumby. Keep it to be a robot; remove it to be a Chumby.

PERIPHERALS: It is fairly easy to demonstrate the Chumby's peripherals. For example, the following command sets the LCD backlight intensity to full. Try this command with different values to see where the backlight first becomes visible.

```
echo 90 > /sys/devices/platform/stmp3xxx-bl/backlight/stmp3xxx-bl/brightness
```

The following two commands clears the LCD framebuffer and then write "Hello, World!" to it:

```
dd if=/dev/zero of=/dev/fb0 bs=640 count=240
echo -n "Hello, World!" | fbwrite --color=255,255,255 --pos=1,3 -
```

The framebuffer can display 22 lines of 53 characters each. One problem with `fbwrite` is that it does not turn pixels off. That is, it only turns pixels on and overwriting an area on the screen with new text usually looks garbled.

Three Chumby peripherals are linked to the Linux input subsystem. They are the top button, the touchscreen, and the rotary encoder. Try each of the following commands (`ctrl-C` to stop) to see each of these peripherals in action. The rotary knob is mapped to the joystick and the top button is mapped to the keyboard.

```
hexdump /dev/input/by-id/soc-noserial-event-joystick
hexdump /dev/input/by-id/soc-noserial-event-kbd
hexdump /dev/input/by-id/soc-noserial-event-ts
hexdump /dev/input/by-id/soc-noserial-ts
```

The Linux input subsystem is well documented and is easy to use. Do an Internet search on the term "Using the Input Subsystem" for more information.

The following commands set the speaker volume and play a wave file. Try different volume settings in the range of 200 to 255.

```
amixer sset DAC 255
aplay /mnt/storage/sounds/bugsbunny1.wav
```

Most WiFi radio link information is available from the following command. You may need to pipe the output of this command to `grep` and `cut` in order to get the specific information you require.

```
iwconfig
```

The Chumby accelerometer is connected to an SPI port on the ARM processor. The `acceld` daemon reads the SPI port and makes the accelerometer data available on a Unix socket. The following command and its source code are included in the tarball to give you an idea of how to use the accelerometer.

```
gacc &
```

PROGRAMMING: The Chumby comes with the shell and Perl installed. The shell is ash and is part of Busybox. Ash might not have every feature that Bash does but it seems to do pretty well for script files. It is easy to add C/C++, Java, and Python to the Chumby. Both cross and native GCC compilers are available, and we cover installation of the native GCC compiler here.

Using the Chumby itself to compile programs means you don't have to scp your programs across and can make development faster. It also means you don't have to install anything on your main computer and that the development environment goes wherever your Chumby goes. The native toolchain even include make, bison, and flex. Only a few commands and a 20MB download are required. SSH to your Chumby and execute the following:

```
cd /mnt/storage
wget http://files.chumby.com/hacks/falconwing_toolchain.sh
sh falconwing_toolchain.sh # takes a couple of minutes
rm falconwing_toolchain.sh
```

Test your new native GCC compiler with the commands

```
gcc -o /mnt/storage/bin/hello /mnt/storage/bin/hello.c
hello
```

If you prefer to do your C/C++ compiles on your normal desktop machine you can install the cross toolchain. Do an Internet search for “Chumby” and “GNU_Toolchain” to get to the Chumby forum page that describes how to install the cross toolchain. It is pretty easy after the 77 MB download finishes.

This page, <http://wiki.chumby.com/mediawiki/index.php/Java>, offers a direct link to the compiled VM that we use in the installation directions below. SSH to the Chumby and execute the following:

```
cd /mnt/storage
wget http://files.chumby.com/languages/java/jamvm_chumby_1.7.tar.gz
tar -xzf jamvm_chumby_1.7.tar.gz
rm java javac # these reference a USB drive. Replace them.
cd /mnt/storage/bin
ln -s jamvm java
echo '#!/bin/sh' > javac
echo 'jikes -classpath /mnt/storage/lib/glibj.zip $@' >> javac
chmod 755 javac
java -version
javac -version
```

Prefer Python? Here's how to install it:

```
cd /mnt/storage
wget http://files.chumby.com/languages/python/python2.6-chumby.tgz
tar -xzf python2.6-chumby.tgz
python --help
```

WEB UI: A web server is a great way to get status from and commands into your robot. The Chumby web server is part of Busybox and supports CGI programs using shell scripts as the CGI programming language. The chumby_robot tarball you installed earlier has a web server running with a document root of /mnt/storage/www. This makes it easy for you create your own web pages and CGI programs. A sample program in /mnt/storage/www shows how to change the backlight brightness using a web form. Try it now by using your browser to display the Chumby robot mode web page. If

the IP address of your Chumby is MY_CHUMBY_IP, then display the following URL:

`http://MY_CHUMBY_IP/index.html`

Enter both valid (0 to 100) and invalid values into the backlight field. Hitting the Enter key should send the new value to the CGI program. Use the CGI programs in /mnt/storage/www/cgi-bin as an example for your own CGI programs.

LINKS: Use the following links to more information about the Chumby, a GREAT robot computer.

`http://www.bunniestudios.com/blog/?p=611` – Bunnie's Chumby One introduction

`http://wiki.chumby.com/mediawiki/index.php/Add_a_video_connector_to_the_chumby_One`

`http://wiki.chumby.com/mediawiki/index.php/Hacking_Linux_for_chumby`

`http://forum.chumby.com` - chumbysphere user forum

`http://wiki.chumby.com/mediawiki/index.php/Chumby_tricks` - list of tricks and tips

`http://files.chumby.com/source/` - Chumby source code

`http://bunniestudios.com/blog/images/chumby_one_backside.jpg`

`http://bunniestudios.com/blog/images/chumby_one_frontside.jpg`