Main page
About CM
Devices
FAQ/Support
Community
Dev Center
Report a Wiki Problem
Recent Changes

Toolbox
What links here
Related changes
Special pages
Printable version
Permanent link

Build for d2tmo

How to Build CyanogenMod 10 for Galaxy S III (TMO) (codename: d2tmo)

Note:

This build walkthrough is auto-generated specifically for the Galaxy S III (TMO) based on the device template at Template:device_d2tmo.

Contents [hide]

Introduction

> **Note:**
> github is having issues with the CM repos, as such you will not be able to follow this build guide until the issues have been resolved. This warning will be removed as soon as the issues have been resolved

These instructions will hopefully assist you to start with a stock Galaxy S III (TMO), unlock the bootloader (if necessary), and then download the required tools as well as the very latest source code for CyanogenMod (based on Google's Android operating system). Using these, you can build both CyangenMod and ClockworkMod recovery image from source code, and then install them both to your device.

It is difficult to say how much experience is necessary to follow these instructions. While this guide is certainly not for the very very very uninitiated, these steps shouldn't require a PhD in software development either. Some readers will have no difficulty and breeze through the steps easily. Others may struggle over the most basic operation. Because people's experiences, backgrounds, and intuitions differ, it may be a good idea to read through just to ascertain whether you feel comfortable or are getting over your head.

Remember, you assume all risk of trying this, but you will reap the rewards! It's pretty satisfying to boot into a fresh operating system you baked at home :) And once you're an Android-building ninja, there will be no more need to wait for "nightly" builds from anyone. You will have at your fingertips the skills to build a full operating system from code to a running device, whenever you want. Where you go from there-- maybe you'll add a feature, fix a bug, add a translation, or use what you've learned to build a new app or port to a new device-- or maybe you'll never build again-- it's all really up to you.

What you'll need

- A Galaxy S III (TMO)
- A relatively recent computer (Linux, OS X, or Windows) w/a reasonable amount of RAM and storage. The less RAM you have, the longer the build will take. Using SSDs results in faster builds than traditional hard drives.
- A micro USB cable
- A decent Internet connection & reliable electricity :)
- Some familiarity with basic Android operation and terminology. It would help if you've installed custom roms on other devices and are familiar with what a recovery image such as ClockworkMod is, for example. It may also be useful to know some basic command line concepts such as `cd` for "change directory", the concept of directory hierarchies, that in Linux they are separated by `/`, etc.

If you are not accustomed to using Linux-- this is an excellent chance to learn. It's free-- just download and run a virtual machine (VM) such as Virtualbox, then install a Linux distribution such as Ubuntu. Any recent 64-bit version should work great, but the latest is recommended.

> **Note:**
> You want to use a 64-bit version of Linux. According to Google, 32-bit Linux environment will only work if you are building older versions prior to Gingerbread (2.3.x)/CyanogenMod 7.

Using a VM allows Linux to run as a guest inside your host computer-- a computer in a computer, if you will. If you hate Linux for whatever reason, you can always just uninstall and delete the whole thing. (There are plenty of places to find instructions for setting up Virtualbox with Ubuntu, so I'll leave it to you to do that.)

So let's begin!

Build CyanogenMod and ClockworkMod Recovery

Prepare the Build Environment

Add a comment to this section

> **Note:**
> You only need to do these steps the first time you build. If you previously prepared your build environment and have downloaded the CyanogenMod source code for another device, skip to **Prepare the device-specific code**.

Install the SDK

Add a comment to this section

If you have not previously installed adb and fastboot, install the Android SDK. "SDK" stands for Software Developer Kit, and it includes useful tools that you will can use to flash software, look at the system logs in real time, grab screenshots, and more-- all from your computer.

> **Helpful Tip!**
> While the SDK contains lots of different things-- the two tools you are most interested in for building Android are adb and fastboot, located in the `/platform-tools` directory.

Install the Build Packages

Add a comment to this section

Several "build packages" are needed to build CyanogenMod. You can install these using the package manager of your choice.

> **Helpful Tip!**
> A package manager in Linux is a system used to install or remove software (usually originating from the Internet) on your computer. With Ubuntu, you can use the Ubuntu Software Center. Even better, you may also use the `apt-get install` command directly in the Terminal.

(Learn more about the apt packaging tool system from Wikipedia.)

For 32-bit & 64-bit systems, you'll need:

```
git-core gnupg flex bison gperf libsdl1.2-dev libesd0-dev libwxgtk2.8-dev squashfs-tools build-essential zip curl
libncurses5-dev zlib1g-dev openjdk-6-jre openjdk-6-jdk pngcrush schedtool
```

For 64-bit only systems, get these:

```
g++-multilib lib32z1-dev lib32ncurses5-dev lib32readline-gplv2-dev gcc-4.7-multilib g++-4.5-multilib
```

Also see http://source.android.com/source/initializing.html which lists needed packages.

Create the directories

Add a comment to this section

You will need to set up some directories in your build environment.

To create them:

```
$ mkdir -p ~/bin
$ mkdir -p ~/android/system
```

Install the `repo` command

Add a comment to this section

Enter the following to download the "repo" binary and make it executable (runnable):

```
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

Put the `~/bin` directory in your path of execution

Make sure that the `~/bin` directory you just created is in your path of execution so that you can easily run the `repo` command even when you're not in `~/bin`. Assuming you are using the BASH shell, the default in recent versions of Ubuntu, you can set it like this:

```
$ export PATH=${PATH}:~/bin
```

Helpful Tip!

You can make this change to the path permanent for all future Terminal sessions:

```
$ gedit ~/.bashrc
```

This will launch a graphical text editor. Enter

```
$ export PATH=${PATH}:~/bin
```

on its own line, then save the file.

Initialize the CyanogenMod source repository

Add a comment to this section

Enter the following to initialize the repository:

```
$ cd ~/android/system/
$ repo init -u git://github.com/CyanogenMod/android.git -b cm-10.1
```

Download the source code

Add a comment to this section

To start the download of all the source code to your computer:

```
$ repo sync
```

The CM manifests include a sensible default configuration for repo, which we strongly suggest you use (i.e., don't pass any options to *sync*). For reference, our current values are -j4, and -c. The “-j4” part means that there will be 4 simultaneous threads/connections. If you experience problems syncing, you can lower this to -j3 or -j2 or even just repo sync by itself. “-c” will ask repo to pull in **only** the current branch, instead of the entire CM history

Prepare to wait a long time while the source code downloads.

Helpful Tip!

The `repo sync` command is used to update the latest source code from CyanogenMod and Google. Remember it, as you can do it every few days to keep your code base fresh and up-to-date.

Get prebuilt apps

Add a comment to this section

Next,

```
$ cd ~/android/system/vendor/cm
```

then enter:

```
$ ./get-prebuilts
```

You won't see any confirmation- just another prompt. But this should cause some prebuilt apps to be loaded and installed into the source code. Once completed, this does not need to be done again.

Prepare the device-specific code

Add a comment to this section

After the source downloads, type: `$ . build/envsetup.sh`

```
$ breakfast d2tmo
```

If all goes well, you should see that d2tmo-specific directories are downloaded automatically.

Helpful Tip!

If you get a **command not found** error for lunch, be sure you've done the "`. build/envsetup.sh`" command from `~/android/system`. Notice there is a period and space ("`. `") in that command.

Helpful Tip!

If you want to know more about what "`$ . build/envsetup.sh`" does or simply want to know more about the `breakfast`, `brunch` and `lunch` commands, you can head over to the Envsetup_help page

Extract proprietary blobs

Add a comment to this section

Now ensure that your Galaxy S III (TMO) is connected to your computer via the USB cable and that you are in the `~/android/system/device/samsung/d2tmo` directory (you can `cd ~/android/system/device/samsung/d2tmo` if necessary). Then run the `extract-files.sh` script:

```
$ ./extract-files.sh
```

You should see the proprietary files (aka “blobs”) get pulled from the device and moved to the right place in the `vendor` directory. If you see errors about adb being unable to pull the files, adb may not be in the path of execution. If this is the case, see the adb page for suggestions for dealing with "command not found" errors.

Note:

It’s important that these proprietary files are properly extracted and moved to the `vendor` directory. Without them, CyanogenMod will build without error, but you’ll be missing important functionality, such as the ability to see anything!

Turn on caching to speed up build

Add a comment to this section

If you want to speed up subsequent builds after this one, type:

```
$ export USE_CCACHE=1
```

Helpful Tip!

Instead of typing `cd ~/android/system` every time you want to return

back to the root of the source code, here's a short command that will do it for you: `croot` . To use this command, as with `brunch`, you must first do "`. build/envsetup.sh`" from `~/android/system`. Notice there is a period and space ("`. `") in that command.

Start the build

Add a comment to this section

Time to start building! So now type:

```
$ croot
$ brunch d2tmo
```

The build should begin.

Helpful Tip!

If the build doesn't start, try `lunch` and choose your device from the menu. If *that* doesn't work, try `breakfast` and choose from the menu. The command `make d2tmo` should then work.

Helpful Tip!

A second, bonus tip! If you get a **command not found** error for `croot` or `brunch` or `lunch`, be sure you've done the "`. build/envsetup.sh`" command in this Terminal session from the `~/android/system` directory.

If the build breaks...

Add a comment to this section

- If you experience this not-enough-memory-related error...

```
ERROR: signapk.jar failed: return code 1make: *** [out/target/product/d2tmo/cm_d2tmo-ota-eng.root.zip] Error 1
```

…you may want to make the following change to:

```
$ system/build/tools/releasetools/common.py
```

Change: `java -Xmx2048m` to `java -Xmx1024m` or `java -Xmx512m`

Then start the build again (with brunch).

- If you see a message about things suddenly being “killed” for no reason, your (virtual) machine may have run out of memory or storage space. Assign it more resources and try again.

Install the build

Assuming the build completed without error (it will be obvious when it finishes), type:

```
# cd $OUT
```

in the same terminal window that you did the build. Here you’ll find all the files that were created. The stuff that will go in `/system` is in a folder called `system`. The stuff that will become your ramdisk is in a folder called `root`. And your kernel is called... `kernel`.

But that’s all just background info. The two files we are interested in are (1) `recovery.img`, which contains ClockworkMod recovery, and (2) `cm-[something].zip`, which contains CyanogenMod.

Install CyanogenMod

Back to the `$OUT` directory on your computer-- you should see a file that looks something like:

```
cm-10-20120718-UNOFFICIAL-d2tmo.zip
```

> **Note:**
> The above file name may vary depending on the version of CM you are building. Your build may not include a version number or may identify itself as a "`KANG`" rather than `UNOFFICIAL` version. Regardless, the file name will end in `.zip` and should be titled similarly to official builds.

Now you can flash the `cm...zip` file above as usual via recovery mode. (Be sure you have backed up any previous installation before trying your new build.)

To get assistance

Add a comment to this section

- #cyanogenmod-dev - A helpful, real-time chat room (or “channel”) on IRC- the Internet Relay Chat.