

Yocto Build for Meta-Tegra Device.

Readme made by Deepak Kumar Beniya
Embedded Software Developer
deepakbeniya64@gmail.com

=> At First install all the required Packages.

```
sudo apt-get install gawk wget git git-core diffstat  
unzip texinfo gcc-multilib build-essential \  
chrpath socat cpio python python3 python3-pip  
python3-pexpect \  
python3-git python3-jinja2 libegl1-mesa pylint3  
rsync bc bison \  
xz-utils debianutils iputils-ping libssl1.2-dev  
xterm \  
language-pack-en coreutils texi2html file  
docbook-utils \  
python-pysqlite2 help2man desktop-file-utils \  
libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf  
automake \  
groff curl lzop asciidoc u-boot-tools  
libreoffice-writer \  
sshpas ssh-askpass zip xz-utils kpartx vim screen
```

=> Follow The steps in your local PC to build YOCTO

=> Make sure that your system should have around
250GB of free space.

1. \$ git clone <https://github.com/OE4T/tegra-demo-distro.git>
2. \$ cd tegra-demo-distro
3. \$ git branch -a
4. \$ git checkout remotes/origin/kirkstone
5. \$ git submodule update --init
6. \$. ./setup-env --machine jetson-orin-nano-devkit --distro
tegrademo build_tegra_demo
7. \$ bitbake demo-image-full (it will create full images)

Targets for building Tegra demo images with test applications:

core-image-minimal - minimally bootable image (no demo apps, no graphics)

demo-image-base - basic image with no graphics

demo-image-egl - (basic image with DRM/EGL, no window manager)

demo-image-sato - X11 image with 'sato' UI

demo-image-weston - Wayland with Weston compositor

demo-image-full - (X11/sato UI plus docker, openCV, VPI, TensorRT and multimedia API samples)

=> After successfully compilation go to

build_tegra_demo/tmp/deploy/images/jetson-orin-nano-devkit/

You will get all the images.

8. \$ cd build_tegra_demo/tmp/deploy/images/jetson-orin-nano-devkit/
9. \$ mkdir tegra_flash
10. \$ tar -xvf
demo-image-full-jetson-orin-nano-devkit.tegraflash.tar.gz -C
tegra_flash/
11. \$ cd tegra_flash

=> Now, you can flash the images through doflash.sh script

=> check this link for more information.

<https://github.com/OE4T/meta-tegra/wiki/Flashing-the-Jetson-Dev-Kit>

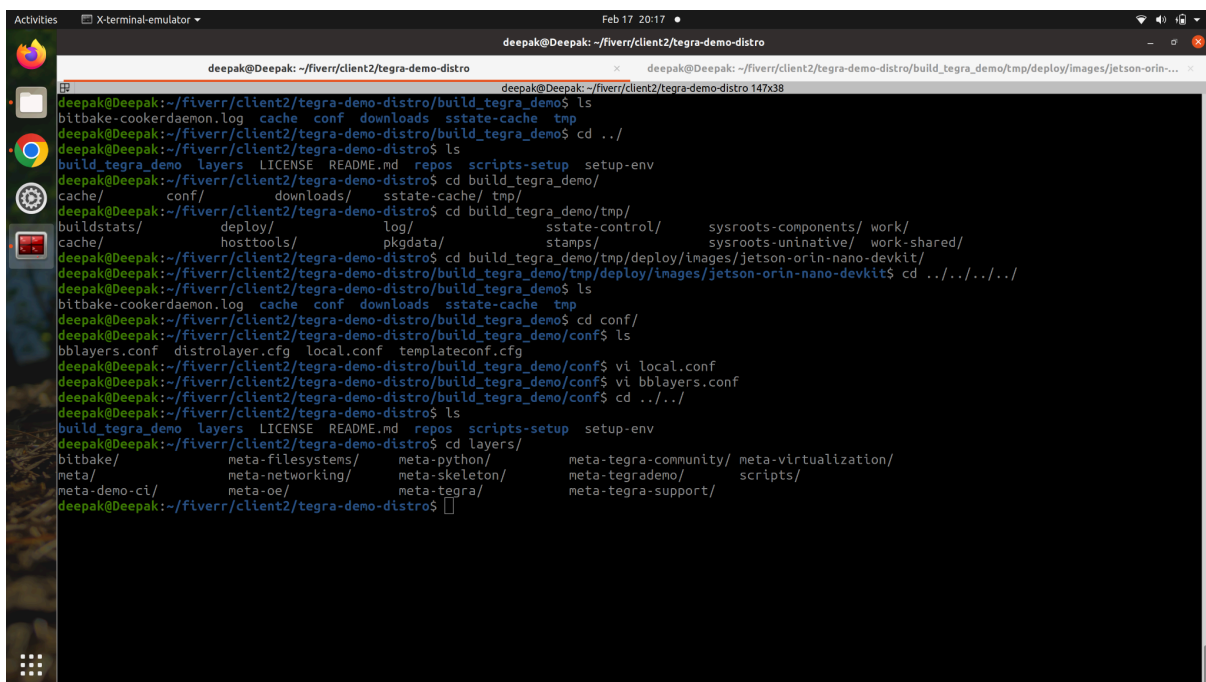
=> use different **bitbake demo-image-base** - basic image with no graphics

=> **bitbake demo-image-egl** - basic image with DRM/EGL, no window manager

=> **bitbake demo-image-sato** - X11 image with 'sato' UI

=> **bitbake demo-image-weston** - Wayland with Weston compositor

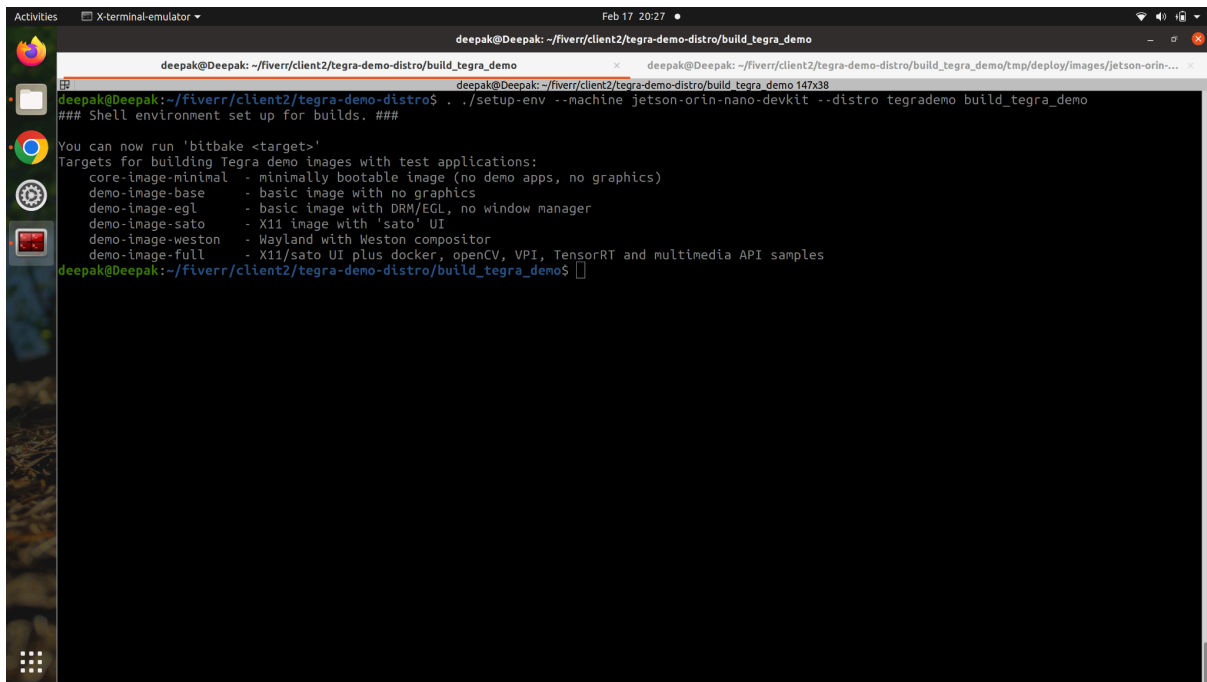
=> **bitbake demo-image-full** - X11/sato UI plus docker, openCV, VPI, TensorRT



```
deepak@Deepak: ~/fiverr/client2/tegra-demo-dist
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo$ ls
bitbake-cookerdaemon.log cache conf downloads sstate-cache tmp
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo$ cd ../
deepak@Deepak:~/fiverr/client2/tegra-demo-distro$ ls
build_tegra_demo layers LICENSE README.md repos scripts-setup setup-env
deepak@Deepak:~/fiverr/client2/tegra-demo-distro$ cd build_tegra_demo/
cache/ conf/ downloads/ sstate-cache/ tmp/
deepak@Deepak:~/fiverr/client2/tegra-demo-distro$ cd build_tegra_demo/tmp/
buildstats/ deploy/ log/ sstate-control/ sysroots-components/ work/
hosttools/ pkgdata/ stamps/ sysroots-uninative/ work-shared/
deepak@Deepak:~/fiverr/client2/tegra-demo-distro$ cd build_tegra_demo/tmp/deploy/images/jetson-orin-nano-devkit/
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/deploy/images/jetson-orin-nano-devkit$ cd ../../../../
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo$ ls
bitbake-cookerdaemon.log cache conf downloads sstate-cache tmp
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo$ cd conf/
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo/conf$ ls
bblayers.conf distrolayer.cfg local.conf templateconf.cfg
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo/conf$ vi local.conf
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo/conf$ vi bblayers.conf
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo/conf$ cd ../../
deepak@Deepak:~/fiverr/client2/tegra-demo-distro$ ls
build_tegra_demo layers LICENSE README.md repos scripts-setup setup-env
deepak@Deepak:~/fiverr/client2/tegra-demo-distro$ cd layers/
bitbake/ meta-filestystems/ meta-python/ meta-tegra-community/ meta-virtualization/
meta/ meta-networking/ meta-skeleton/ meta-tegradem0/ scripts/
meta-demo-ci/ meta-oe/ meta-tegra/ meta-tegra-support/
deepak@Deepak:~/fiverr/client2/tegra-demo-distro$
```

=> When ever you need to compile then this command should be executed at first so that the terminal will be ready . It's basically called enableing toolchain.

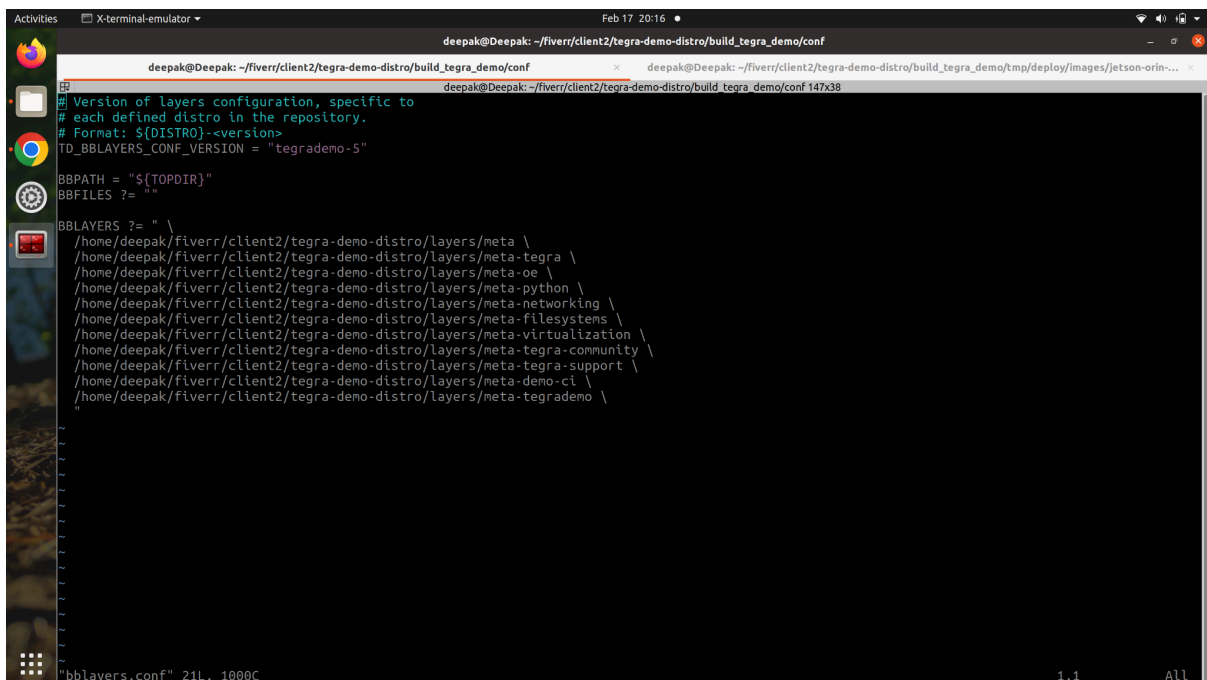
./setup-env --machine jetson-orin-nano-devkit --distro tegrademo build_tegra_demo



```
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo$ ./setup-env --machine jetson-orin-nano-devkit --distro tegrademo build_tegra_demo
### Shell environment set up for builds. ###

You can now run 'bitbake <target>'
Targets for building Tegra demo images with test applications:
core-image-minimal - minimally bootable image (no demo apps, no graphics)
demo-image-base   - basic image with no graphics
demo-image-egl    - basic image with DRM/EGL, no window manager
demo-image-sato   - X11 image with 'sato' UI
demo-image-weston - Wayland with Weston compositor
demo-image-full   - X11/sato UI plus docker, openCV, VPI, TensorRT and multimedia API samples
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo$
```

=> These are meta layers that add into this bblayers.conf file



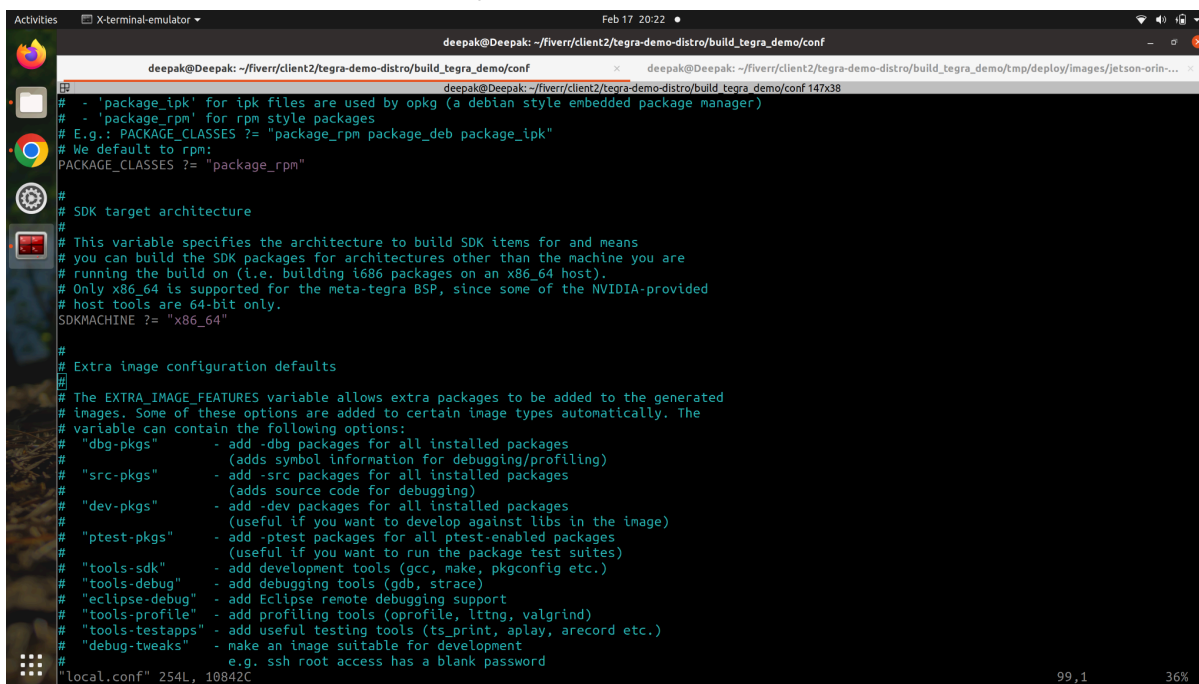
```
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/conf
deepak@Deepak:~/fiverr/client2/tegra-demo-distro/build_tegra_demo/conf$ cat bblayers.conf
###
# Version of layers configuration, specific to
# each defined DISTRO in the repository.
# Format: ${DISTRO}-<version>
TD_BBLAYERS_CONF_VERSION = "tegrademo-5"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= "
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-tegra \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-oe \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-python \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-networking \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-filesystems \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-virtualization \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-tegra-community \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-tegra-support \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-demo-ci \
/home/deepak/fiverr/client2/tegra-demo-distro/layers/meta-tegrademo \
"

"bblayers.conf" 21L, 1000C
1,1 ALL
```

=> If you want to add some packages like (vim, mqtt, tensor flow) then it can be added in this file bblayers.conf



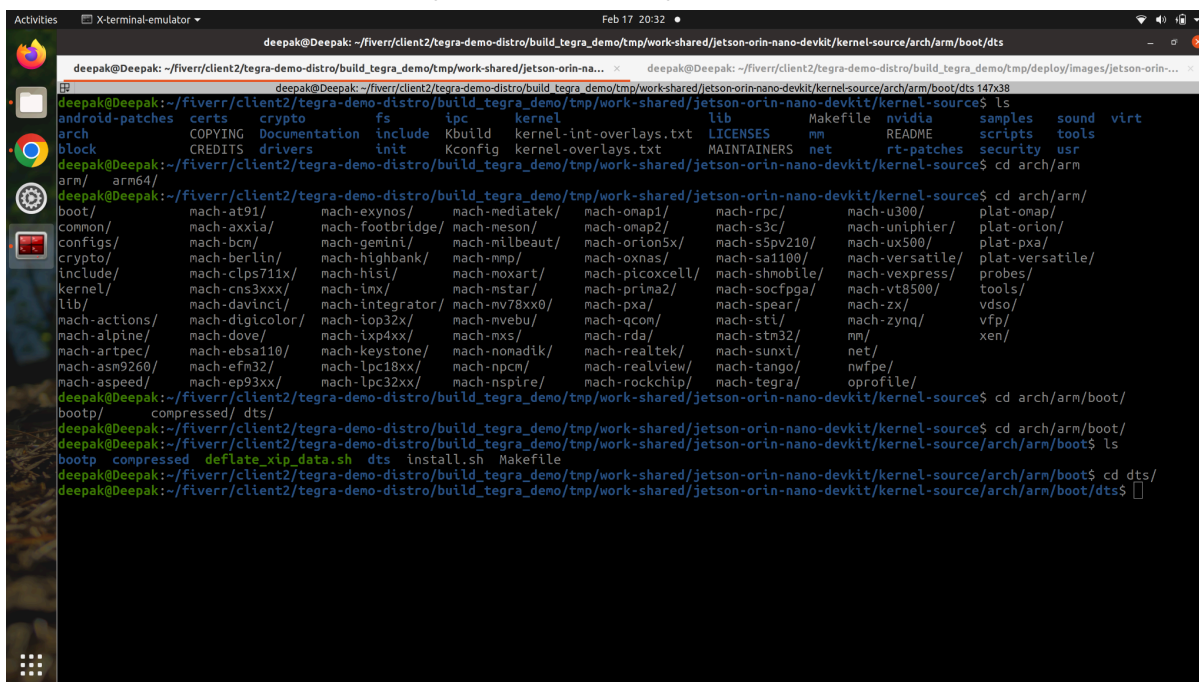
```
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/conf
# - 'package_ipk' for ipk files are used by opkg (a debian style embedded package manager)
# - 'package_rpm' for rpm style packages
# E.g.: PACKAGE_CLASSES ?= "package_rpm package_deb package_ipk"
# We default to rpm:
PACKAGE_CLASSES ?= "package_rpm"

#
# SDK target architecture
#
# This variable specifies the architecture to build SDK items for and means
# you can build the SDK packages for architectures other than the machine you are
# running the build on (i.e. building i686 packages on an x86_64 host).
# Only x86_64 is supported for the meta-tegra BSP, since some of the NVIDIA-provided
# host tools are 64-bit only.
SDKMACHINE ?= "x86_64"

#
# Extra image configuration defaults
#
# The EXTRA_IMAGE_FEATURES variable allows extra packages to be added to the generated
# images. Some of these options are added to certain image types automatically. The
# variable can contain the following options:
# "dbg-pkgs" - add -dbg packages for all installed packages
#             (adds symbol information for debugging/profiling)
# "src-pkgs" - add -src packages for all installed packages
#             (adds source code for debugging)
# "dev-pkgs" - add -dev packages for all installed packages
#             (useful if you want to develop against libs in the image)
# "ptest-pkgs" - add -ptest packages for all ptest-enabled packages
#              (useful if you want to run the package test suites)
# "tools-sdk" - add development tools (gcc, make, pkgconfig etc.)
# "tools-debug" - add debugging tools (gdb, strace)
# "eclipse-debug" - add Eclipse remote debugging support
# "tools-profile" - add profiling tools (oprofile, lttng, valgrind)
# "tools-testapps" - add useful testing tools (ts_print, aplay, arecord etc.)
# "debug-tweaks" - make an image suitable for development
#                 e.g. ssh root access has a blank password

"local.conf" 254L, 10842C 99,1 36%
```

=> for kernel modification you can modify here.



```
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/work-shared/jetson-orin-nano-devkit/kernel-source/arch/arm/boot/dts
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/work-shared/jetson-orin-nano-devkit/kernel-source/arch/arm/boot/dts$ ls
android-patches  certs  crypto  fs  ipc  kernel  lib  Makefile  nvidia  samples  sound  virt
arch             COPYING  Documentation  include  Kbuild  kernel-int-overlays.txt  LICENSES  mm  README  scripts  tools
block           CREDITS  drivers  init  Kconfig  kernel-overlays.txt  MAINTAINERS  net  rt-patches  security  usr
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/work-shared/jetson-orin-nano-devkit/kernel-source$ cd arch/arm
arm/  arm64/
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/work-shared/jetson-orin-nano-devkit/kernel-source$ cd arch/arm/
boot/          mach-at91/      mach-exynos/    mach-mediatek/  mach-omap1/    mach-rpc/      mach-u300/    plat-omap/
common/       mach-axxia/    mach-footbridge/ mach-meson/     mach-omap2/    mach-s3c/      mach-uniphier/ plat-orion/
configs/     mach-bcm/      mach-gemini/    mach-milbeaut/  mach-orion5x/  mach-s5pv210/  mach-ux500/    plat-pxa/
crypto/      mach-berlin/   mach-highbank/  mach-mmp/       mach-oxnas/    mach-sa1100/   mach-versatile/ plat-versatile/
include/     mach-clps711x/ mach-hist/      mach-moxart/    mach-plcoxcell/ mach-shmobile/ mach-vexpress/ probes/
kernel/      mach-cns3xxx/  mach-imx/       mach-mstar/     mach-prima2/   mach-socfpga/  mach-vt8500/  tools/
lib/         mach-davinci/  mach-integrator/ mach-mv78xx0/   mach-pxa/      mach-spear/    mach-zx/      vdsso/
mach-actions/ mach-digicolor/ mach-iop32x/    mach-mvebu/     mach-qcom/     mach-sti/      mach-zynq/    vfp/
mach-alpine/  mach-dove/     mach-lxp4xx/    mach-mxc/       mach-rida/     mach-stn32/    mm/           xen/
mach-artpec/  mach-ehs110/   mach-keystone/  mach-nomadik/   mach-realtek/  mach-sunxi/    net/
mach-asmp9260/ mach-efm32/    mach-lpc18xx/   mach-npcm/      mach-realview/ mach-tango/    nwfp/
mach-aspeed/  mach-ep93xx/   mach-lpc32xx/   mach-nspire/    mach-rockchip/ mach-tegra/    oprofile/
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/work-shared/jetson-orin-nano-devkit/kernel-source$ cd arch/arm/boot/
boot/  compressed/  dts/
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/work-shared/jetson-orin-nano-devkit/kernel-source$ cd arch/arm/boot/
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/work-shared/jetson-orin-nano-devkit/kernel-source/arch/arm/boot$ ls
bootp  compressed  deflate_xip_data.sh  dts  install.sh  Makefile
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/work-shared/jetson-orin-nano-devkit/kernel-source/arch/arm/boot$ cd dts/
deepak@Deepak: ~/fiverr/client2/tegra-demo-distro/build_tegra_demo/tmp/work-shared/jetson-orin-nano-devkit/kernel-source/arch/arm/boot/dts$
```

