The Android Build System

Introduction

Ron Munitz

Updated: February 2014

PSCG

PSCG

The Android Build System

- Host Tools
- Target Tools
- Platform tools
- Documentation Tools
- The Target's (Android Platform) code base
- **Build System**

The Android Build System

- Build Systems are a huge topic.
- A build system
  - Takes a series of *rules* or *recipes*
  - Knows how to generate an embedded platform images/artifacts from source code, configuration files, BLOBs, ...
  - Allows to easily select a premade configuration and build a ready to use code without working too hard
  - Allows customizing products and defining new ones

The Android Build System

- Consists of two essential folders:
  - ***build/*** Contains the definitions of the build system, along with some predefined devices.
  - **device/** Contains definitions for devices. The build system rules really parse them

- 🐵 An optional, out of source control folder may be added for vendor specific additions (mainly BLOBs). It is listed as ***vendor/***

build/

- buildspec.mk.default - template for remake
- CleanSpec.mk - Build cleanup definitions
- core - Build System rules
- envsetup.sh - environment preparation script
- libs - some host helper libs
- **target** - Target definitions.
- tools - Building, packaging, etc.

The Android Build System

- Main folder: **build/**
- Based on GNU make
  - Makefile on top directory says: #include build/core.mk
  - Then a lot of other files are included.
  - Heavily uses Python and bash.
- Heavily uses environment variables
  - @see build/envsetup.sh

The Android Build System

- Very easy to use for building:
  - . build/envsetup.sh #sets env vars/functions
  - lunch <config>-<variant> # selects configuration
  - make   # That's gnu make, no customizations.
- Very easy for flashing via Software tools
  - With emulator (no need to flash…)
  - With fastboot
  - With other custom bootloaders

build/target

- This is what you should probably care about.
- Contains two folders:
  - **board/** - board definition file
  - **product/** - product definition file
- Build recipes are defined in these folders.
- Device definitions in ***device/*** include files from these folders, and derive from recipes.
- Or include their own.

build/target/board

Where (some) board magic is defined

build/target/board

- Android.mk - automatically included by build
  - includes the ***no longer necessary*** **AndroidBoard.mk** file at TARGET_DEVICE_DIR
    - What **is** required is **BoardConfig.mk** .
    - @see build/core/config.mk
  - Populates the fastboot read android-info.txt file with the contents of the devices board.info.txt
    - Or with "board=$(TARGET_BOOTLOADER_BOARD_NAME)"
- Templates for predefined boards
  - @see next slide

build/target/board (cont.)

- Contains build templates for predefined boards
  - emulator, generic_<arch>, vbox_x86
- Each contains:
  - AndroidBoard.mk (obsolete)
  - **BoardConfig**.mk - Board definitions (see next slide)
  - device.mk - Some Board/Hardware related pacakges at device.mk and BoardConfig.mk

A bit about BoardConfig.mk

- TARGET_NO_BOOTLOADER - Self Explaining
- TARGET_NO_KERNEL - use prebuilt kernel
- TARGET_USE_CAMERA_STUB
- ...
- Architecture, ABIs, Partition layout, OpenGL config, Radio config…

BoardConfig.mk search path

- The build system searches for **BoardConfig.mk** at the following locations:
  - build/target/board/$TARGET_DEVICE/
  - device/*/$TARGET_DEVICE/
  - vendor/*/$TARGET_DEVICE/
- If there is no such file - the build fails.
- If there is more than one match - the build fails.

build/target/product

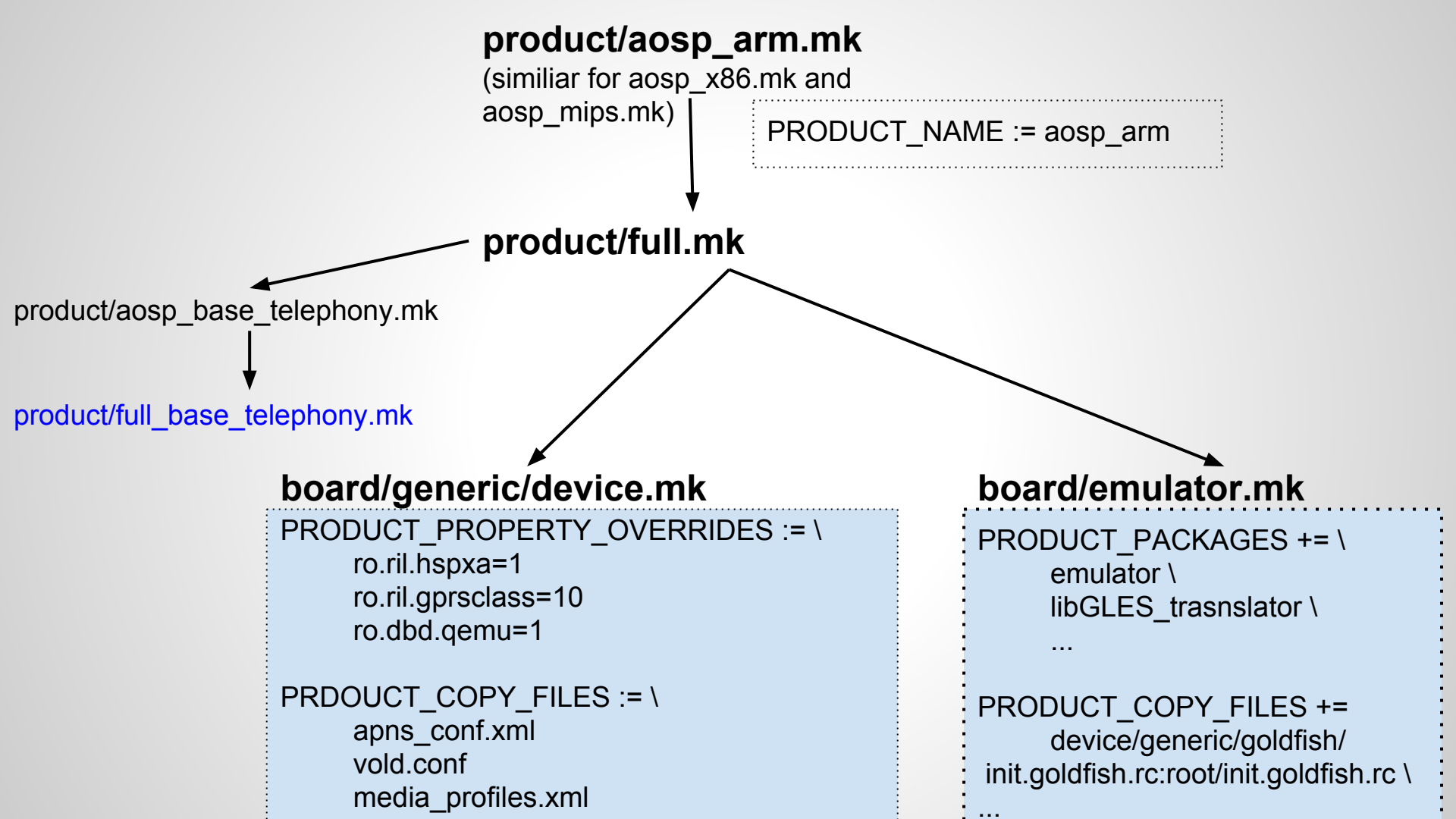
Where (product packages) dreams come true

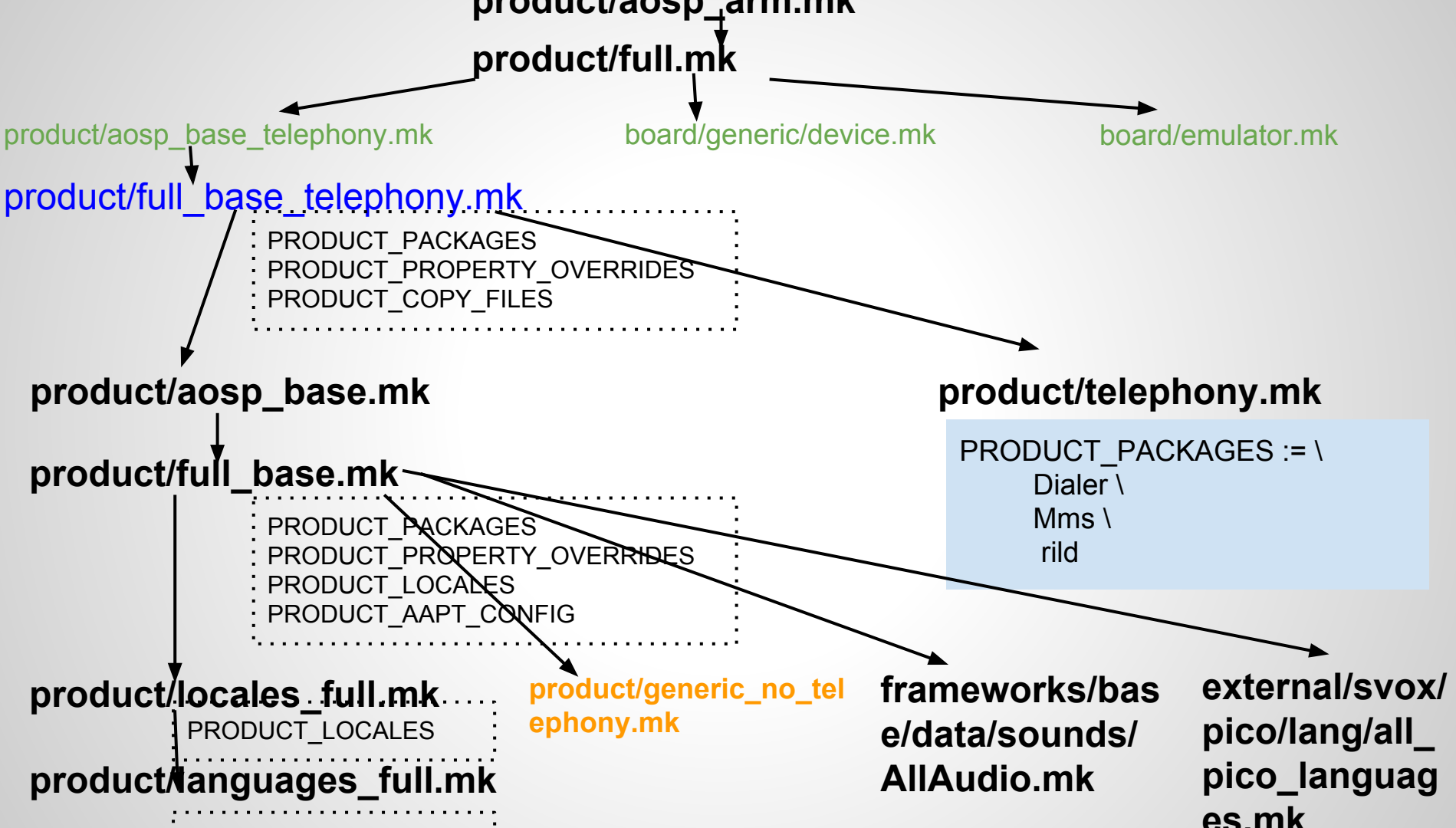
build/target/product

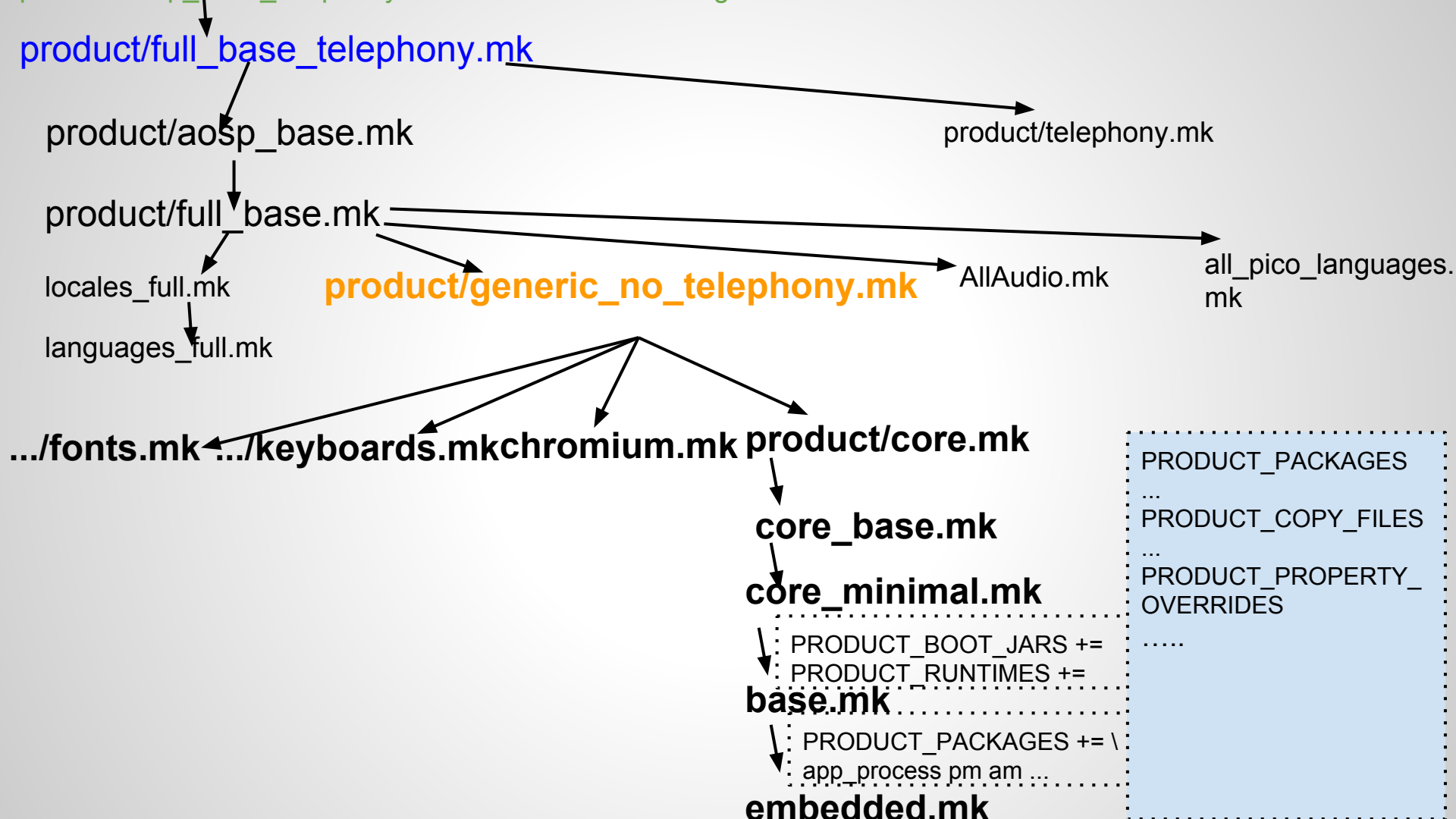
- **AndroidProducts.mk** defines a list of products to add to the build system.
- More products can be added by declaring additional AndroidProducts.mk files in either
  - device/.../
  - vendor/.../
- **security/** includes prebuilt certificates
- Quick Start flow:
  - generic_mips.mk INHERITS generic.mk INHERITS generic_no_telephony.mk AND telephony.mk ...

build/target/product example

- aosp_x86.mk
  INHERITS
  - full_x86.mk
    - INHERITS
      - aosp_base_telephony.mk
      - board/generic_x86/device.mk
    - INCLUDES
      - $(SRC_TARGET_DIR)/product/emulator.mk
- Another example + a full walkthrough are given in the next slides


product/aosp_arm.mk
(similiar for aosp_x86.mk and
aosp_mips.mk)
PRODUCT_NAME := aosp_arm
product/full.mk
product/aosp_base_telephony.mk
product/full_base_telephony.mk
board/generic/device.mk
PRODUCT_PROPERTY_OVERRIDES := \
ro.ril.hspxa=1
ro.ril.gprsclass=10
ro.dbd.qemu=1
PRDOUCT_COPY_FILES := \
apns_conf.xml
vold.conf
media_profiles.xml
board/emulator.mk
PRODUCT_PACKAGES += \
emulator \
libGLES_trasnslator \
...
PRODUCT_COPY_FILES +=
device/generic/goldfish/
init.goldfish.rc:root/init.goldfish.rc \
...


product/aosp_arm.mk
product/full.mk
product/aosp_base_telephony.mk
board/generic/device.mk
board/emulator.mk
product/full_base_telephony.mk
PRODUCT_PACKAGES
PRODUCT_PROPERTY_OVERRIDES
PRODUCT_COPY_FILES
product/aosp_base.mk
product/telephony.mk
PRODUCT_PACKAGES := \
Dialer \
Mms \
rild
product/full_base.mk
PRODUCT_PACKAGES
PRODUCT_PROPERTY_OVERRIDES
PRODUCT_LOCALES
PRODUCT_AAPT_CONFIG
product/locales_full.mk
PRODUCT_LOCALES
product/languages_full.mk
product/generic_no_telephony.mk
frameworks/base/data/sounds/AllAudio.mk
external/svox/pico/lang/all_pico_languages.mk


product/full_base_telephony.mk
product/aosp_base.mk
product/telephony.mk
product/full_base.mk
locales_full.mk
product/generic_no_telephony.mk
AllAudio.mk
all_pico_languages.mk
languages_full.mk
.../fonts.mk
.../keyboards.mk
chromium.mk
product/core.mk
core_base.mk
core_minimal.mk
PRODUCT_BOOT_JARS +=
PRODUCT_RUNTIMES +=
base.mk
PRODUCT_PACKAGES += \
app_process pm am ...
embedded.mk
PRODUCT_PACKAGES
...
PRODUCT_COPY_FILES
...
PRODUCT_PROPERTY_OVERRIDES
.....

build/core/build-system.html

The definitive build system document!
Most recent news!
Draft version
From 2006...
Does give a good overview of the design criteria.

build/core

The nails and hammers of the printed, framed dream

build/core

- Processes the definitions we have discussed of the product and board
- defines the language for them
- Also applies a lot of other configuration, compiler rules, and what not...
- @see config.mk ,main.mk, product.mk for info

Thank You