**1.1 Build Agema Yocto**

Let’s focus on Agema Yocto project. This is a very practical approach. There

are a number of high-level steps involved in building Yocto Project for Agema.

Most steps are implemented as a bash script in build\_yocto\_step\_\*.sh file.

Let’s overview what these steps are below.

\_ Create a new work directory for Yocto Project.

*Rationale*: Maintain a clean separation between Yocto builds.

\_ Clone init-kconfig git project

*Rationale*: It contains build\_yocto\_step\_\*.sh bash script.

\_ Choose a desired Yocto high-level configuration.

*Rationale*: Easily choose Yocto-2.5.2 with BCM-SDK 6.5.15.

\_ Create and customize Yocto Project.

*Rationale*: Add meta layers and update local.conf.

\_ Build Yocto Project.

*Rationale*: Run bitbake core-image-full-cmdline.

\_ Upload Yocto Project build artifacts.

*Rationale*: Share Yocto eSDK with other teams.

Let’s do these steps one-by-one.

**1.1.1 Create a new work directory for Yocto Project**

$ cd $HOME

$ mkdir -p workspace/yocto

$ cd workspace/yocto

It is highly advised to keep a separate directory for every Yocto Project build!

**1.1.2 Clone initial git project**

1.1. BUILD AGEMA YOCTO 5

$ cd $HOME/workspace/yocto

$ git clone ssh://git@bitbucket.mrv.co.il:7999/yoc/init-kconfig.git

The init-kconfig is a tiny project that contains build scripts and *high-level*

configuration for Yocto Project build.

**1.1.3 Choose a desired Yocto high-level configuration**

The init-kconfig project contains high-level configuration files – they are

found under defconfigs directory.

$ cd init-kconfig

$ ls -1 defconfigs

$ agema\_yocto-2.5.1\_bcm-sdk-6.5.15

$ agema\_yocto-2.5.1\_bcm-sdk-6.5.7

$ agema\_yocto-2.5.2\_bcm-sdk-6.5.15

$ agema\_yocto-2.6\_bcm-sdk-6.5.15

Every one of these configuration files contains a *high-level* configuration for

Yocto Agema build *that may change* relatively frequently. Let’s take a look

at the meaninful part of this configuration:

$ cd init-kconfig

$ cat defconfigs/agema\_yocto-2.5.2\_bcm-sdk-6.5.15 | grep -v "^#"

CONFIG\_BCM\_SDK\_6\_5\_15=y

CONFIG\_KEEPALIVED\_2\_0\_12=y

CONFIG\_YOCTO\_2\_5\_2=y

That’s simple – this file defines the version of BCM-SDK package, keepalived

package and the version of Yocto Project itself. This is the gist of it. We need

to choose one of these configuration files and copy it to .config file to create

the effective configuration for Yocto Project build. Note, that this follows how

Kbuild build system that is used in Linux kernel and other projects. 1 The

kbuild\_skeleton project was used here as a reference. 2

So, let’s “activate” the chosen configuration.

$ cd init-kconfig

$ cp defconfigs/agema\_yocto-2.5.2\_bcm-sdk-6.5.15 .config

The .config file will be used by the following build steps.

1https://github.com/embedded-it/kbuild-template

2https://github.com/masahir0y/kbuild\_skeleton

6 1. YOCTO

**1.1.4 Create and customize Yocto Project**

Now we run the scripts that will create and customize a classical Yocto Project

environment.

**— IMPORTANT NOTE —** Use with care the build step that removes a

previous Yocto Project build – build\_yocto\_step\_02\_remove\_previous.sh

script!

Here, all the needed git sub-projects will be cloned and organized in a file

hierarchy expected by Yocto Project build process.

$ cd init-kconfig

$ ./build\_yocto\_step\_01\_configure.sh

$ ./build\_yocto\_step\_03\_clone.sh

$ ./build\_yocto\_step\_04\_checkout\_branches.sh

$ ./build\_yocto\_step\_05\_customize\_yocto\_build\_conf.sh

$ ./build\_yocto\_step\_06\_add\_layers.sh

Let’s comment the scripts above.

**The gist of build step scripts**

\_ build\_yocto\_step\_01\_configure.sh

Converts .config to more handy configuration format.

\_ build\_yocto\_step\_03\_clone.sh

Clones the meta layers using Google repo tool.

\_ build\_yocto\_step\_04\_checkout\_branches.sh

Checks out git branch according to .config configuration.

\_ build\_yocto\_step\_05\_customize\_yocto\_build\_conf.sh

Applies customizations to local.conf build configuration.

\_ build\_yocto\_step\_06\_add\_layers.sh

Adds the needed meta layers via bitbake-layers add-layer.

1.1. BUILD AGEMA YOCTO 7

**1.1.5 Build Yocto Project**

All the previous build steps may be thought of as a preparation (setup and

customization) for the actual Yocto Project build. These steps take about

10-15 minutes of running time on a strong machine. The next script actually

builds the whole Yocto Project – it supposedly takes 4 to 5 hours on the same

strong machine.

Most of the build time is spent on building GCC toolchain and generic Linux

packages. Currently, Yocto image that is used for Agema products contains

around 300 packages – see the command below.

**How many packages in Agema?**

$ cd init-kconfig/ybld/poky

$ source ./oe-init-build-env ../build

$ bitbake -g core-image-full-cmdline

$ cat recipe-depends.dot \

| grep -v -e '-native' | grep -v digraph | grep -v -e '-image' \

| awk '{print $1}' | sort | uniq | wc -l

The last command gives us the answer – image contains 292 packages.

**Only two bitbake commands**

The actually important Yocto commands that build the image are run by

bitbake utility. So, there are only two commands that we *actually* need to

build Yocto for Agema:

\_ bitbake core-image-full-cmdline

\_ bitbake core-image-full-cmdline -c populate\_sdk\_ext

The first command builds all the packages and creates the Rootfs image. The

second command creates Yocto eSDK self-installable file. Most of the build

time is spent on bitbake core-image-full-cmdline command.

8 1. YOCTO

**Zoom in to run bitbake commands**

Sometimes, we need to circumvent the high-level build scripts and to run the

bitbake commands by ourselves in Linux terminal.

To run bitbake commands separately (not inside the script) open **new terminal**

**window** and run the below commands:

$ cd init-kconfig/ybld/poky

$ source ./oe-init-build-env ../build

$ bitbake core-image-full-cmdline

$ bitbake core-image-full-cmdline -c populate\_sdk\_ext

**Run the build script**

So, let the build begin.

$ cd init-kconfig

$ ./build\_yocto\_step\_07\_yocto\_build.sh

**1.1.6 Upload Yocto Project build artifacts**

There is only one Yocto build artifact that needs to be shared with the outside

world – Yocto eSDK self-installable file. Yocto eSDK needs to be installed on

the development machine of every application developer before building and

creating the self-installable *bundle* file for Agema device.

Yocto eSDK self-installable file is found in the Yocto deploy directory as

shown below.

$ cd init-kconfig/ybld

$ cd build/tmp/deploy/sdk

$ ls poky-glibc-x86\_64-core-image-full-cmdline-i586-toolchain-ext-2.5.2.sh

Currently the size of Yocto eSDK self-installable image is around 2 GB. The

upload is done via simple scp tmp/deploy/sdk/$name.\* <user>@<IP>:...

commands.

1.1. BUILD AGEMA YOCTO 9

**Run the upload script**

So, let’s do the actual upload of Yocto eSDK:

$ cd init-kconfig

$ ./build\_yocto\_step\_08\_upload.sh

*— NOTE —* The upload step should be run *only* by CI/CD (TeamCity) or

*only* when you know what you are doing. Otherwise, Yocto eSDK image that

is used by all application developers may be overwritten by not-yet-tested

image.

**1.1.7 Concluding notes**

**Run all build scripts in one shot**

To wrap it up – all the Yocto Agema build script may be executed locally

on development machine in one shot as it is done in CI/CD approach (on

TeamCity).

Just copy-paste the commands below.

$ cd init-kconfig

$ cp defconfigs/agema\_yocto-2.5.2\_bcm-sdk-6.5.15 .config

$ ./build\_yocto\_step\_01\_configure.sh

$ ./build\_yocto\_step\_03\_clone.sh

$ ./build\_yocto\_step\_04\_checkout\_branches.sh

$ ./build\_yocto\_step\_05\_customize\_yocto\_build\_conf.sh

$ ./build\_yocto\_step\_06\_add\_layers.sh

$ ./build\_yocto\_step\_07\_yocto\_build.sh

**Build configuration vs development iteration**

Build steps build\_yocto\_step\_[01..06]\*.sh should be done only once – this

is the initial configuration step! Build step build\_yocto\_step\_07\_yocto\_build.sh

- is the development iteration that may be done many times.

10 1. YOCTO

**Custom scripts or bitbake**

When working extensively with Yocto development - it is advised to work

with bitbake commands instead of custom build\_yocto\_step\_\*.sh scripts.

Try the below commands to begin working with bitbake commands.

*— NOTE —* Always use new fresh terminal windows when working with

Yocto commands (to keep bash environment clean).

Just copy-paste the commands below.

$ cd init-kconfig/ybld/poky

$ source ./oe-init-build-env ../build

$ bitbake core-image-full-cmdline

$ bitbake keepalived

**Yocto dynamic vs static configuration**

Linux kernel, U-Boot and many other packages use Kbuild configuration system

where the *total* project configuration is kept in one file – .config file.

Yocto, on the other hand, does not have a static configuration saved in a file.

Yocto creates the projects configuration (analogous to .config) every time

when bitbake command is executed. To observe this configuration file let’s

use the bitbake -e <recipe> command.

$ cd init-kconfig/ybld/poky

$ source ./oe-init-build-env ../build

$ bitbake -e keepalived

The configuration is huge – about 20,000 lines – and it contains all the needed

configuration to build a recipe.

**Log file for build script**

The full log of the build is saved in ylog file. It is advised to tail -f this log

file as the Yocto build is running in the background.

$ cd init-kconfig

$ tail -f ylog

1.2. AGEMA DEV INSTALL ENVIRONMENT 11

**1.2 Agema dev install environment**

\_ Two important variables on the script are yocto and bcm\_sdk.

# ~~~~~~~~~~~~~~~~~

# Script parameters

# ~~~~~~~~~~~~~~~~~

yocto="2.5.2" # default

bcm\_sdk="6.5.15" # default

\_ Clones OpenClovis on host development machine.

OpenClovis makefiles MUST be accessed during Agema build.

\_ Install Yocto eSDK on host development machine.

Re-create Rootfs image using devtool build-image command, because

Yocto eSDK installer does not contain Rootfs images.

\_ Clone metaswitch project (it is separate git repository).

It is advised to use Google repo tool to clone all the needed git projects:

$ git clone

*,*! ssh://git@bitbucket.mrv.co.il:7999/agema/agema\_build.git

$ cd agema\_build

$ ./setup\_build\_env.sh

**1.3 Agema build script build\_agema˙py**

\_ Reads the JSON environment file

\_ Sets version

\_ Build routing apps – actually calles SAFplus\_build.sh script

\_ The SAFplus\_build.sh script sources environment-setup-agema file

to activation Yocto eSDK environment and define location of OpenClovis

source tree.

\_ For incremental build – call build\_agema.py -q (calls to SAFplus\_build.sh).

\_ For full build – call build\_agema.py (calls to SAFplus\_build.sh all).

12 1. YOCTO

\_ Copy all components to temporary folder (e.g., Yocto Rootfs, selfextractor.sh,

rootfs\_extras, kernel image, SCNodeIO.tgz).

SCNodeIO.tgz contains all network applications (developed by ADVA)

in the form of OpenClovis agents.

Yocto Rootfs contains only “build-once” artifacts.

\_ Create tar.gz archive from above components and prepend (using cat

Linux utility) the selfextractor.sh script – bundle is ready!