Freescale Yocto Project Tutorial

Leonardo Sandoval

2013

Menu

- Build & Boot your FSL Yocto Image in N Steps
- Folders
- Architecture
- Metadata/inputs
- Common Development tasks
  - Creating a new layer
  - Patching the Kernel
  - Building the Kernel with make
  - Contribute to the Freescale Yocto Project

Build and Boot your Freescale Yocto Image in n-steps

- Check required packages for your Linux Distribution and install them
- Install the repo utility following these steps

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

- Download the BSP metadata (recipes + configuration files + classes)

```
$ mkdir fsl-community-bsp
$ cd fsl-community-bsp
fsl-community-bsp $ repo init \
                        -u https://github.com/Freescale/fsl-community-bsp-platform \
                        -b dylan
fsl-community-bsp $ repo sync # Takes some minutes the first time
```

- Select your machine and prepare the bitbake's environment

```
# To list all FSL related machines, type
fsl-community-bsp $ find sources/meta-fsl* -name "*.conf" | grep "conf/machine"
fsl-community-bsp $ MACHINE=<selected machine> . ./setup-environment build
# if MACHINE is not set, the default machine is 'imx6qsabresd'
build $
```

- Choose an image and bake it!

```
build $ bitbake-layers show-recipes | grep image    # To list all possible images
build $ bitbake <selected image>                     # Bake! The first time can
                                                     # take several hours.
# e.g bitbake core-image-minimal
```

- Flash

```
# Insert your SD Card
# Type '£ dmesg | tail' to see the device node being used, e.g /dev/sdb)
# In case SD to be flash has already some partitions, the host system may have
# mounted these, so unmount them, e.g. '£ sudo umount /dev/sdb?'.
build $ ls -la 'tmp/deploy/images/*.sdcard'
```

```
# Flash the soft link one
build $ sudo dd \
            if=tmp/deploy/images/<selected image>-<select machine>.sdcard \
            of=/dev/sdX \
            bs=1M
build $ sync
```

- Place your SD Card in the correct board's slot and boot!

Found Errors? Subscribe and report it to meta-freescale list

Yocto Folders

- **fsl-community-bsp**: Base (BASE) directory where all Yocto data resides (recipes, source code, built packages, images, etc)
- **BASE/sources**: Source (SOURCE) directory where metadata (layers) resides
- **BASE/build**: Build (BUILD) directory where `bitbake` commands are executed
- **BASE/build/tmp**: Target (TMP) directory for all bitbake commands
- **BASE/build/tmp/work**: Working (WORKING) directory for recipes tasks
- **BASE/build/tmp/deploy**: Deploy (DEPLOY) directory where bitbake's output data is found
- **BASE/build/tmp/deploy/images**: Complete and partial images are found under this folder

Architecture

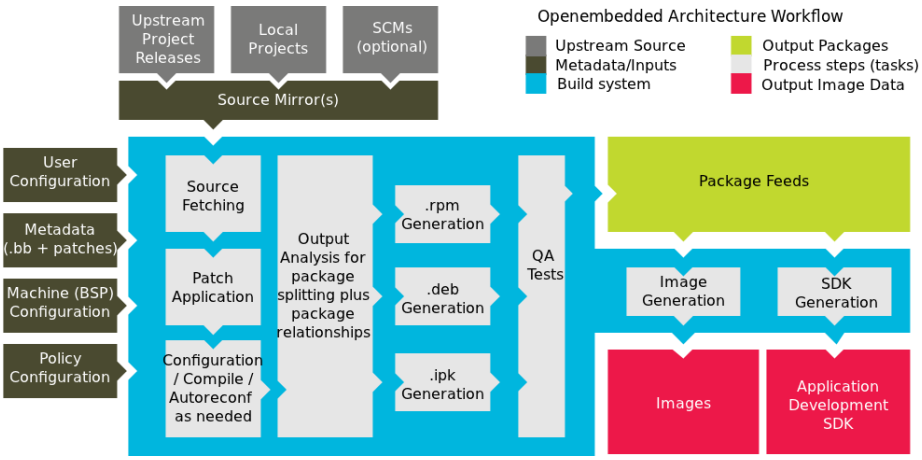

Figure 1:

Metadata

BitBake handles the parsing and execution of the data files. The data itself is of various types:

- Recipes: Provides details about particular pieces of software.
- Class Data: Abstracts common build information (e.g. how to build a Linux kernel).
- Configuration Data: Defines machine-specific settings, policy decisions, and so forth. Configuration data acts as the glue to bind everything together.

Layers

- Metadata is organized into multiple `layers`.
- Layers allow you to isolate different types of customizations from each other.
- DO NOT do your modifications in existing layers, instead create a layer and create recipes (.bb files) or modified existing ones (.bbappend files)

Configuration Data

- `build/conf/local.conf`: Local User Configuration for your build environment
- `build/conf/bblayers.conf`: Define layers, which are directory trees, traversed by BitBake.
- `sources/meta-*/conf/layer.conf`: Layer configuration file
- `sources/meta-*/conf/machine/*.conf`: Machine configuration files

Build's local configuration file `build/conf/local.conf`

```
MACHINE ??= 'wandboard-dual'
DISTRO ?= 'poky'
#PACKAGE_CLASSES ?= "package_rpm"
EXTRA_IMAGE_FEATURES = "debug-tweaks"
USER_CLASSES ?= "buildstats image-mklibs image-prelink"
```

```
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS = "\
    STOPTASKS,${TMPDIR},1G,100K \
    STOPTASKS,${DL_DIR},1G,100K \
    STOPTASKS,${SSTATE_DIR},1G,100K \
    ABORT,${TMPDIR},100M,1K \
    ABORT,${DL_DIR},100M,1K \
    ABORT,${SSTATE_DIR},100M,1K"
CONF_VERSION = "1"

BB_NUMBER_THREADS = '4'
PARALLEL_MAKE = '-j 4'
ACCEPT_FSL_EULA = ""
#added by bitbake
DL_DIR = "/home/b42214/fsl-local/yocto/fsl-community-bsp-dylan/downloads/"
#added by bitbake
SSTATE_MIRRORS = ""
#added by bitbake
PACKAGE_CLASSES = "package_rpm"
```

Important variables:

- `MACHINE`: Indicates the machine, `imx6qsabresd` is the default
- `BB_NUMBER_THREADS` and `PARALLEL_MAKE`: Indicate the max number of threads when baking and compiling
- `DL_DIR`: Tarball repository. Several users can share the same folder, so data can be reused.

Build's layer configuration file `build/conf/bblayers.conf`

- Automatically created by the `setup-environment` script (see section 'Build & Boot your FSL Yocto Image in N Steps')
- Only modified when adding a new layer

```
LCONF_VERSION = "6"

BBPATH = "${TOPDIR}"
BSPDIR := "${@os.path.abspath(os.path.dirname(d.getVar('FILE', True)) + '/../..')}"

BBFILES ?= ""
BBLAYERS = " \
  ${BSPDIR}/sources/poky/meta \
```

```
  ${BSPDIR}/sources/poky/meta-yocto \
  \
  ${BSPDIR}/sources/meta-openembedded/meta-oe \
  \
  ${BSPDIR}/sources/meta-fsl-arm \
  ${BSPDIR}/sources/meta-fsl-arm-extra \
  ${BSPDIR}/sources/meta-fsl-demos \
"
```

Layer configuration file `meta-fsl-arm/conf/layer.conf`

```
# We have a conf and classes directory, add to BBPATH
BBPATH .= ":${LAYERDIR}"

# We have a packages directory, add to BBFILES
BBFILES += "${LAYERDIR}/recipes-*/*/*.bb \
            ${LAYERDIR}/recipes-*/*/*.bbappend"

BBFILE_COLLECTIONS += "fsl-arm"
BBFILE_PATTERN_fsl-arm := "^${LAYERDIR}/"
BBFILE_PRIORITY_fsl-arm = "5"

FSL_EULA_FILE = "${LAYERDIR}/EULA"

FSL_MIRROR ?= "http://www.freescale.com/lgfiles/NMG/MAD/YOCTO/"

MIRRORS += " \
${FSL_MIRROR}   http://download.ossystems.com.br/bsp/freescale/source/ \n \
"
```

Important variables:

- `BBFILES`: Indicates where to look for `.bb*` files
- `BBFILE_PRIORITY_fsl-arm`: Indicates layer's priority
- `MIRRORS`: Indicates where to get the source code

Machine configuration file: `meta-fsl-arm/conf/imx6qsabresd.conf`

```
#@TYPE: Machine
#@NAME: i.MX6Q SABRE SD
#@DESCRIPTION: Machine configuration for Freescale i.MX6Q SABRE SD
```

```
include conf/machine/include/imx-base.inc
include conf/machine/include/tune-cortexa9.inc

SOC_FAMILY = "mx6:mx6q"

KERNEL_DEVICETREE = "${S}/arch/arm/boot/dts/imx6q-sabresd.dts"

UBOOT_MACHINE = "mx6qsabresd_config"

SERIAL_CONSOLE = "115200 ttymxc0"

MACHINE_FEATURES += " pci wifi bluetooth"
```

[conf/machine/include/imx-base.inc]

Important variables:

- IMAGE_FSTYPES: Located on imx-base.inc. Defines the type of outputs for the Root Filesystem. Default is: "tar.bz2 ext3 sdcard"
- UBOOT_ENTRYPOINT_*: Located on imx-base.inc. Defines where the Kernel is loaded by U-boot
- SOC_FAMILY: Defines machine's family. Only recipes with the same SOC_FAMILY (defined with the recipe's variable COMPATIBLE_MACHINE) are taken into account when baking for a particular machine.
- UBOOT_MACHINE: Define the u-boot configuration file

Creating a new Layer

It is suggested to create a layer when creating or modifying any metadata file (recipe, configuration file or class). The main reason is simple: modularity. In the other hand, make sure your new metadata has not already be implemented (layer, recipe or machine), so before proceeding check this link.

- To have access to Yocto scripts, setup the enviroment from the BASE folder

```
fsl-community-bsp $ . setup-environment build
```

- Move to the place you want to create your layer and choose a name (e.g. `fsl-custom`)

```
sources $ yocto-layer create fsl-custom
# Answer the questions. Make sure the priority is set correctly (higher numbers,
# higher priorities). Set the priority equal to the lowest already present, except
# when you have introduce a new recipe with the same name as other and want to shadow
# the original one.
```

- Add any metadata contect. Suggestion: Version the layer with Git and upload your local git repo to a server
- Edit and add the layer to the `build/conf/bblayers.conf` file
- To verify that your layer is *seen* by BitBake, run the following command under the BUILD folder

```
build $ bitbake-layers show-layers
```

Patching the Linux Kernel

The Linux Kernel is just another recipe for Yocto, so learning to patch it you learn to patch any other package. In the other hand, Yocto **should not** be used for package development, but in those rare cases follow the below steps. It is assumed that you have already build the package you want to patch.

- Create the patch or patches. In this example we are patching the Linux kernel for wandboard-dual machine; in other words, the value of MACHINE on the `build/conf/local.conf` is `MACHINE ??= 'wandboard-dual'`. In case you already have the patches, make sure these can be nicely applied with the commands `git apply --check` <PATCH_NAME>, and jump this step

```
build $ cd tmp/work/wandboard_dual-poky-linux-gnueabi/linux-wandboard/3.0.35-r0/git
build $ # Edit any files you want to change
build $ git add <modified file 1> <modified file 2> ..  # Select the files you
                                                        # want to commit
build $ git commit -s -m '<your commit's title>'        # Create the commit
build $ git format-patch -1                             # Create the patch
```

- Create a new layer (see section 'Creating a new Layer')
- On the new layer (e.g `meta-fsl-custom`) , create the corresponding sub-folders and the `.bbfile`

```
sources $ mkdir -p \
            meta-fsl-custom/recipes-kernel/linux/linux-wandboard-3.0.35/
sources $ cat > meta-fsl-custom/recipes-kernel/linux/linux-wandboard_3.0.35.bbappend
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}-${PV}:"
SRC_URI += "file://0001-calibrate-Add-printk-example.patch"
PRINC := "${@int(PRINC) + 1}"
^d
```

- Move the patch to the new layer

```
sources $ cp \
../build/tmp/work/wandboard_dual-poky-linux-gnueabi/linux-wandboard/3.0.35-r0/\
    git/0001-calibrate-Add-printk-example.patch \
    meta-fsl-custom/recipes-kernel/linux/linux-wandboard-3.0.35
```

- Setup the enviroment and clean previous package's build data (sstate)

```
fsl-community-bsp $ . setup-environment build
build $ bitbake -c cleansstate linux-wandboard
```

- Compile and Deploy

```
build $ bitbake -f -c compile linux-wandboard
build $ bitbake -c deploy linux-wandboard
```

- Insert the SD into your Host and copy the `uImage` into the first partition. Do not forget to unmount the partition before removing the card!

```
build $ sudo cp tmp/deploy/images/uImage /media/Boot\ wandbo/
```

- Insert the SD into your board and test your change.

Building the Kernel Manually

- To setup the Yocto environment, from the BASE folder run

```
fsl-community-bsp $ . setup-environment build
```

- Build the toolchain

```
build $ bitbake meta-toolchain
# Other toolchains:
#   Qt Embedded toolchain build: bitbake meta-toolchain-qte
#   Qt X11 toolchain build: bitbake meta-toolchain-qt
```

- Install it on your PC

```
build $ sudo sh \
    tmp/deploy/sdk/poky-eglibc-x86_64-arm-toolchain-<version>.sh
```

- Setup the toolchain environment

```
build $ source \
    /opt/poky/<version>/environment-setup-armv7a-vfp-neon-poky-linux-gnueabi
```

- Get the Linux Kernel's source code.

```
$ git clone git://git.freescale.com/imx/linux-2.6-imx.git linux-imx
$ cd linux-imx
```

- Create a local branch

```
linux-imx $ BRANCH=imx_3.0.35_4.0.0 # Change to any branch you want,
                                    # Use 'git branch -a' to list all
linux-imx $ git checkout -b ${BRANCH} origin/${BRANCH}
```

- Export ARCH and CROSS_COMPILE

```
linux-imx $ export ARCH=arm
linux-imx $ export CROSS_COMPILE=arm-poky-linux-gnueabi-
linux-imx $ unset LDFLAGS
```

- Choose configuration and compile

```
linux-imx $ make imx6_defconfig
linux-imx $ make uImage
```

- To Test your changes, copy the `uImage` into your SD Card

```
linux-imx $ sudo cp arch/arm/boot/uImage /media/boot
```

- If case you want your changes to be reflected on your Yocto Framework, create the patches following the section 'Patching the kernel'.

Contributing to the Freescale Yocto Project

The Yocto Project is open-source, so anyone can contribute. No matter what your contribution is (bug fixing or new metadata), contributions are sent through patches to a community list. Many eyes will look into your patch and at some point it is either rejected or accepted. Follow these steps to contribute:

- Make sure you have previously configured your personal info

```
$ git config --global user.name "Your Name Here"
$ git config --global user.email "your_email@example.com"
```

- Subscribed to the Freescale Yocto Project Mailing List
- Download `master` branches

```
fsl-community-bsp $  repo init \
    -u https://github.com/Freescale/fsl-community-bsp-platform \
    -b master
```

- Update

```
fsl-community-bsp $ repo sync
```

- Create local branches so your work is *not* done on master

```
fsl-community-bsp $ repo start <branch name> --all
```

Where <`branch name`> is any name you want to give to your local branch (e.g. `fix_uboot_recipe`, `new_gstreamer_recipe`, etc.)

- Make your changes in any Freescale related folder (e.g. sources/meta-fsl-arm). In case you modified a recipe (.bb) or include (.inc) file, do not forget to *bump* (increase the value by one) either the `PR` or `INC_PR` value
- Commit your changes using `git`. In this example we assume your change is on `meta-fsl-arm` folder

```
sources/meta-fsl-arm $ git add <file 1> <file 2>
sources/meta-fsl-arm $ git commit
```

On the commit's log, the title must start with the filename change or introduced, then a brief description of the patch's goal, following with a long description. Make sure you follow the standards (type `git log --pretty=oneline` to see previous commits)

- Create a patch

```
sources/meta-fsl-arm $ git format-patch -s \
    --subject-prefix='<meta-fsl-arm][PATCH' -1
```

Where the last parameter (`-1`) indicate to patch last commit. In case you want to create patches for older commits, just indicate the correct index. If your patch is done in other folder, just make sure you change the `--subject-prefix` value.

- Send your patch or patches with

```
git send-email --to meta-freescale@yoctoproject.org <patch>
```

where <patch> is the file created by `git format-patch`.

- Keep track of patch's responses on the mailing list. In case you need to rework your patch, repeat the steps but this time the patch's subject changes to `--subject-prefix='<meta-fsl-*][PATCH v2'`
- Once your patch has been approved, you can delete your working branches

```
fsl-community-bsp $ repo abandon <branch name>
```