Guideline to SoC and OEM

This section is intended to help SoC/ OEM vendors to integrate RDK's Yocto framework to their platform.

A basic description about SoC and OEM layers can be found here: Yocto layers

SoC layer design guidelines

- There shall be separate configuration file (.inc) for each variant of chip-set which will be used by the OEM layer to build a specific device.
- There shall be an option to build an image for reference platforms
- There shall be only one layer for a particular SoC which can support multiple chip variants and the same layer can be integrated with different OEM layers
- All the binaries and libraries shall be installed in /usr/bin and /usr/lib of RFS ; It shall not be installed in any other directory. Remember /bin /sbin /lib are for system binaries and libraries which are common for all platforms.The /usr/local is used for overriding the existing one with recompiled binary/library which is not applicable in Set Top Box scenario.

OEM layer design guidelines

- There shall be separate device (machine) configuration file (.conf) for each device for the particular chip family for which the OEM layer is intended for
      For Eg : A layer "meta-rdk-oem-OEM-X-SOC-Y" means this layer shall be able to build any devices manufactured by OEM "X" with all variants of SoC "Y" like Y-1,Y-2 etc
- It is recommended to use small case alphanumerical characters to denote a machine configuration file. For example ; qemux86.conf. Avoid special characters, "_" etc. Remember an "_" denotes overriding in Yocto scenario and therefore using it for configuration file name can cause issues.
- If one OEM use two different SoCs for different devices, there shall be two separate layers such as "meta-rdk-oem-OEM-X-SOC-Y" , "meta-rdk-oem-OEM-X-SOC-Z" etc
- The device (machine) configuration file shall include corresponding include (.inc) file to get machine configuration details.

Guideline to create OEM specific images

All the work needs be done only in the meta-rdk-oem layer

Write a recipe <oem>-image-tools-native.bb under recipes-tools to copy all the tools, binaries and scripts needed to create custom oem image to the staging binary directory

- This recipe should fetch all the sources from the GIT repo
- Example name space for the GIT repo where you should place all the tools like permission files, authentication keys etc - **“rdk/devices/<OEM>/<OEM_device>/tools”**
- Example namespace for the GIT repo to keep all the binaries (crcsum etc) - **“rdk/devices/<OEM>/<OEM_device>/bin”**
- This recipe will have only one task – do_install, which copies all the necessary files to create oem images to staging binary directory
- **"**inherit native" this class would short-circuit all the target build and strip tasks.

Adding a new SoC/OEM to RDK

Adding a new machine to the Yocto Project is a straightforward process which involves following steps:

- Create a new layer which will hold all the recipes and machine configurations for the new SoC/OEM.
- Adding the Machine Configuration File for the new SoC/OEM.
- Adding a Kernel for the Machine.
- Adding Recipe for SoC/OEM
- Creating packages for building images

Creating the new SoC/OEM Layer

Use the yocto-layer create sub-command to create a new general layer.

- yocto-layer create mylayer

There shall be separate device (machine) configuration file (.conf) for each device for the particular chip family for which the layer is intended for

- For Eg : A layer "meta-rdk-oem-OEM-X-SOC-Y" means this layer shall be able to build any devices manufactured by OEM "X" with all variants of SoC "Y" like Y-1,Y-2 etc

The device (machine) configuration file shall include corresponding include (.inc) file to get machine configuration details.

Adding the Machine Configuration File

To add a machine configuration, you need to add a .conf file with details of the device being added to the conf/machine/ file.
The most important variables to set in this file are as follows:

- TARGET_ARCH (e.g. "arm")
- PREFERRED_PROVIDER_virtual/kernel
- MACHINE_FEATURES (e.g. "apm screen wifi")

You might also need these variables:

- KERNEL_IMAGETYPE (e.g. "zImage")
- IMAGE_FSTYPES (e.g. "tar.gz jffs2")

The default configuration are defined in meta-rdk/conf/distro/rdk.conf and it should be overwritten by the machine specific conf file.
For example, meta-rdk-oem-<>/meta-<>/conf/machine/include/<>.inc

- PREFERRED_PROVIDER_virtual/iarmmgrs-hal = "iarmmgrs-hal-broadcom"
- PREFERRED_PROVIDER_virtual/closedcaption-hal = "closedcaption-hal-broadcom"

Adding a Kernel for the Machine

The OpenEmbedded build system needs to be able to build a kernel for the machine. We need to either create a new kernel recipe for this machine, or extend an existing recipe. There are several kernel examples in the Source Directory at meta/recipes-kernel/linux that can be used as references.
If you are creating a new recipe, following steps need to be done:

- setting up a SRC_URI.
- Specify any necessary patches
- create a configure task that configures the unpacked kernel with a defconfig.

If you are extending an existing kernel, it is usually a matter of adding a suitable defconfig file. The file needs to be added into a location similar to defconfig files used for other machines in a given kernel.
A possible way to do this is by listing the file in the SRC_URI and adding the machine to the expression in COMPATIBLE_MACHINE:

- COMPATIBLE_MACHINE = '(qemux86|qemumips)'

Adding recipes for SoC/OEM

The following kind of recipes can be added to SoC/OEM layer. The recipes shall be grouped as described in slide “BSP Reference Layer”

- recipes (.bb) to build Kernel
- recipes(.bb) to build SDK
- Kernel patches (SoC/OEM specific - if any)
- SDK patches (SoC/OEM specific - if any)
- Any SoC/OEM specific scripts or files which need to be installed in RFS

Creating packages for building images

- Create a custom package-group for the SoC/OEM which shall list all the recipes that are required for the image.
- Create a custom image for generating RFS for required SoC/OEM.

Bitbake work-flow

- All the components are built using individual recipes. There shall be a main image recipe (example , rdk-generic-image) which includes all other required recipe and create the final RFS
- Package groups recipe is one support a image recipe to select the set of packages
- The recipes will be called in sequence
  (1) opensource components
  (2) Kernel
  (3) SDK
  (4) RDK
  (5) MSO
  (6) Packaging and create final image.

- The final linux and RFS image will be created under build_folder/tmp/deploy/images