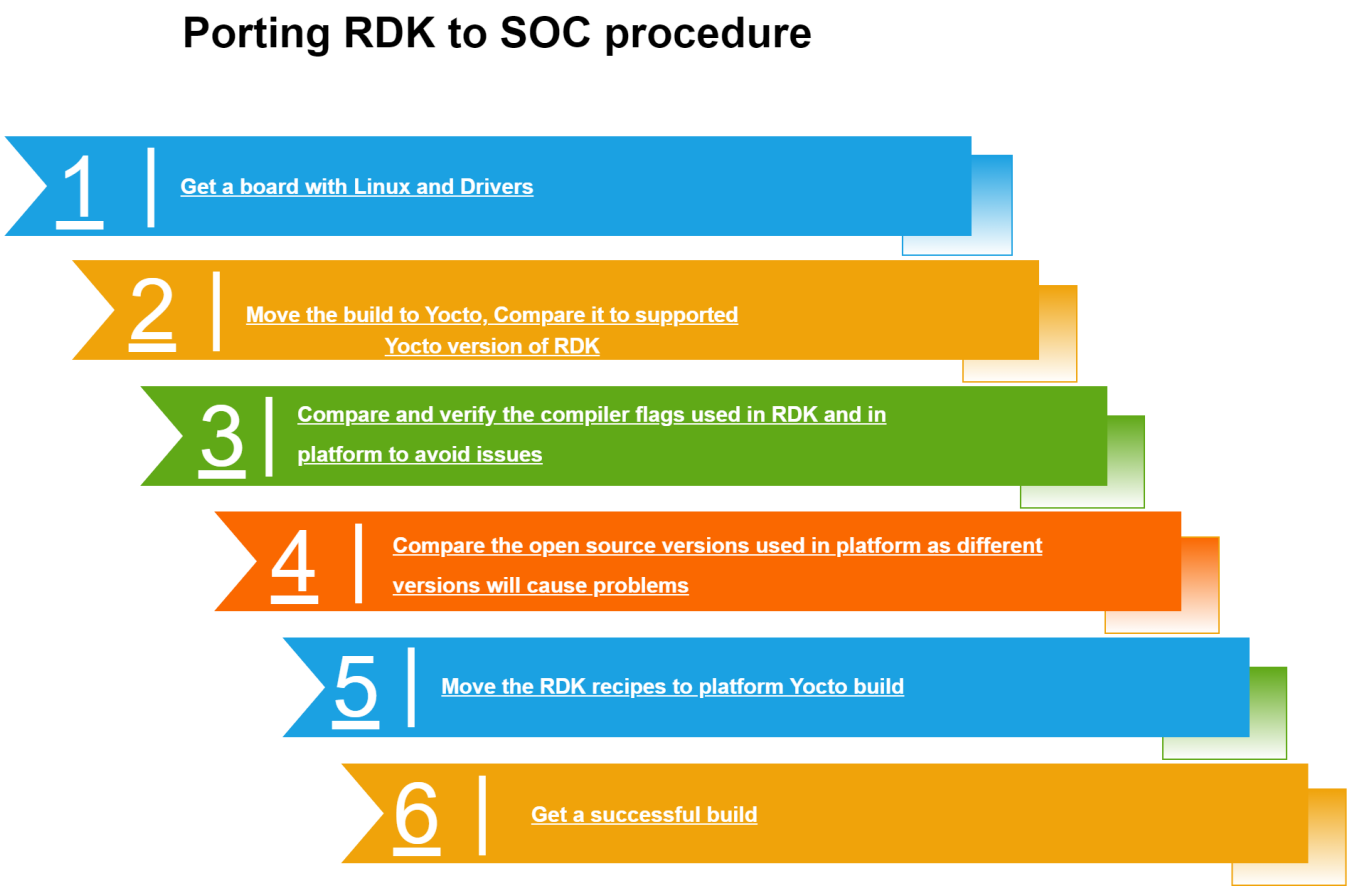
RDK-V 4.0 - SOC Porting Guide

The aim of this SoC porting guide is to guide SoC Vendors on how to port RDK to their platforms.



Step 1 : Get a board with Linux and drivers

RDK is based on Yocto Linux. Prior to porting RDK on a SoC , the precondition is to have the SoC platform running on Linux. The Linux version can be a SoC specific one with kernal hardening and other OS optimizations specific to SoC, as RDK could easily run on top of vendor specific Linux. SoC should also provide drivers for the other peripherals in SoC platform, like WiFi, so that the unit can work independently and completely from a SoC point of view.

Step 2 : Move the build to Yocto, compare it to supported Yocto version of RDK

Once SoC vendor is ready with an own port of Linux + drivers for the SoC platform, it is time to migrate the platform to Yocto based builds. If the SoC is already having a Yocto based Linux, this step can be skipped. The current Yocto versions supported are Yocto 3.1- Dunfell (preferred) as well as Yocto 2.2- Morty (soon to be deprecated)

Yocto 3.1 Upgradation support the following:

- Version upgrades for bitbake, GStreamer and other OE components
- Linux kernel 5.4 or above
- Extensible SDK

Each component in RDK is a standalone repository with its own individual build tools producing a library or set of binaries. When OE layers are upgraded to the newer versions, necessary changes need to be made in the RDK Yocto meta layers which use these components, to avoid build failures.

Step 3 : Compare and verify the compiler flags used in RDK and in platform to avoid issues

This is an important step while porting RDK to a new SoC platform. RDK Linux is built with considering particular build flags/features in target platform(For example, RDK considers hardware floating point in platform where as some platforms are on software based floating point). SoC vendor need to analyze such flags in RDK and then make a comparison with the existing SoC platform Linux to ensure compatibility or to understand the required modifications in RDK code so as to house the compatibility changes

Specific DISTRO_FEATURES can be added to support build time flag for specific platforms. For example : DISTRO_FEATURES_append = " referencepltfm "

Step 4 : Compare the open source versions used in platform as different versions will cause problems

Depending on the Yocto version, the RDK build will be working with some particular version of Open source components. This might either be a dependency with the Yocto version compatibility as such or with RDK (functionality or license issues). If the SoC Linux has some version dependency on particular open source software and, if it conflicts with the version in RDK, vendor needs to make required changes to make the open source version to match the RDK requirements as best as possible, by adding required patches in SoC platform

Check below layers for all opensourced version packages recipes used in RDK. If multiple recipes with different versions are available, then check for the value of **PREFFERED_VERSION_<recipe name>** set.

Meta layers : - meta-openembedded, openembedded-core, meta-rdk-ext, meta-rdk, meta-cmf

Step 5 : Move the RDK recipes to platform Yocto build

For platform specific recipes, keep them in SoC meta layer. While it is a good practice to start afresh with a new manifest for the target platform, manifest file for a similar featured platform can be used as a starting point too. Check all device specific repos in the reference manifest, and ensure corresponding device repos are created for this new device as needed, and update the manifest with these updated repos

- Create artifactory repo for the project with appropriate permissions for vendors. This is used for hosting any binaries required for the project
- Check-in SoC components - tool chain, SDK, kernel, drivers, etc.. to corresponding SoC gerrit repos/ artifactory as applicable
- Populate meta SoC layer with initial set of changes needed to build kernel, SoC components, etc..
- If there is any SoC reference board exists, create corresponding machine configuration in SoC layer, and create a image target to be able to build final image for the reference board. This layer should be separated out with in meta SoC layer from other common SoC, common chip sub layers which will be usually used by meta oem layer as well.
- Populate meta oem layer with machine configuration and other bare minimum changes required to generate a target image for OEM board.
- machine configuration can be updated with "NEEDED_BSPLAYERS" field to include required SoC, OEM layers in the build
- Any unwanted recipes during early stage of bring up can be masked using BBMASK, if needed.

Step 6 : Get a successful build

Add platform specific main recipe to create image.

HAL implementation by SoC

Device Settings

Device settings component is having a HAL interface to control device specific peripherals such as video port, audio port and display and front panel.

More details on HAL interface can be found here: Device Settings HAL Types & Public API DTCP HAL Interfaces.

DTCP

Integrates the SoC provided DTCP library with DTCP/IP manager Interface implementation which manages source/sink DTCP/IP sessions and performs the encryption/decryption.

HAL Interface specification: DTCP HAL Interfaces

tr69hostif

Contains SoC specific MOCA libraries, headers and MOCA profile codes.

API Details: TR69 Host Interface Handler

For more details on tr69hostif, please refer: tr69hostif

gst-plugins-rdk /qamtunersrc

qamtunersrc is a push based gstreamer source plugin which tunes to the given service and provides the SPTS data.

- Manages tuner and demux
- Filters PIDs required for SPTS
- Output SPTS as gstreamer buffer

Properties

1. Modulation:
2. Frequency: frequency to tune in KHZ
3. Tunerid: Tuner Handle

Depends on platform specific libraries for tune, filtering, and pod functionalities.

gst-plugins-SoC /playersinkbin

Playersinkbin is a gstreamer bin element consisting of demux, decoder and sink elements. A template file gstplayersinkbin.c.template and gstplayersinkbin.h.template are provided as a reference for SoC implementation. SoC has to add details of platform specific plugins and implement the required properties expected out of them.

hdmicec

SDK Vendors should implement CEC driver interface API as specified in hdmi_cec_driver.h

HAL Interface Specificcation: HDMI-CEC HAL APIs specification

iarmmgrs

Power, IR and DeepSleep modules are having SoC dependency. APIs are specified in plat_power.h, plat_ir.h and deepSleepMgr.h

More details about APIs can be found here

westeros-SoC

Contains functions for creating and handling native eglwindow. HAL APIs are specified in westeros-gl.h

Wi-Fi (Client)

Wi-Fi Client HAL provides an interface (data structures and API) to interact with underlying Wi-Fi driver and enabling the client to be connected with an Access Point.

HAL APIs are specified in wifi_client_hal.h. Doxygen Link: Wifi HAL API Specification

Provisioning - iCrypto

Set of Cryptographic APIs Implementation, can be used to achieve all type of Cryptographic requirements from Premium App.

Graphics - westeros backend

Media Pipeline Backend layer is a module in device layer which glues the media pipeline capabilities of the device to that of the IgnitionX.

Input key handling

As part of integrating Premium Apps as a native application on the RDK stack, the Premium Apps Video Porting (PVP) Layer needs to be implemented. The PVP Layer consists of the following modules:

- Ignition Device Layer
  - Common
  - Display
  - Input
  - TTS
  - Secure Storage
  - Message Bus
- Device Properties Provider
- Media Pipeline Backend
- PlayReady DRM

Media Playback - gstreamer

Abstraction layer for third-party applications to get specific GStreamer values from the SoC pipeline.

DRM (playready, widevine, TEE, SVP etc.)

Amazon assets are encrypted using Microsoft PlayReady. A robust implementation of the PlayReady must be provided for porting kit to decrypt assets.

TTS

Text-to-speech converts text into spoken voice output to help customers navigate the Prime Video application without seeing the screen. Text-to-speech is mandatory for the US region and optional in other regions.