Yocto开发手册

ADC615

Exported on 12/04/2023

Table of Contents

1 Yocto所有git仓库 4
2 Yocto基本介绍 6
3 Yocto 构建过程 8
4 Yocto官方文档 10
5 ADC615 Yocto项目简介 11
5.1 adc615-yoctohttps://github-am.geo.conti.de/ADAS/adc615-yocto 11
5.2 meta-conti-adashttps://github-am.geo.conti.de/ADAS/meta-conti-adas 11
5.3 meta-adc615-aragohttps://github-am.geo.conti.de/ADAS/meta-adc615-arago 13
5.4 meta-adc615-userspacehttps://github-am.geo.conti.de/ADAS/meta-adc615-userspace 14
6 如何进行Yocto构建 15
7 如何使用Yocto进行开发&维护 16
7.1 修改u-boot和kernel 16
7.1.1 1.直接修改u-boot仓库的代码 16
7.1.2 2.打补丁方式修改u-boot 16
7.1.3 补充：添加kernel自动加载模块 17
7.2 为rootfs中添加内容 17
7.2.1 添加package 17
7.2.2 添加系统配置及文件 18
7.2.2.1 修改以太网配置 18
7.2.2.2 添加.profile 18
7.3 如何添加开源库 19
7.4 注意事项 20
7.5 有用的小窍门 20

- Yocto所有git仓库(see page 4)
- Yocto基本介绍(see page 6)
- Yocto 构建过程(see page 8)
- Yocto官方文档(see page 10)
- ADC615 Yocto项目简介(see page 11)
    - adc615-yoctohttps://github-am.geo.conti.de/ADAS/adc615-yocto(see page 11)
    - meta-conti-adashttps://github-am.geo.conti.de/ADAS/meta-conti-adas(see page 11)
    - meta-adc615-aragohttps://github-am.geo.conti.de/ADAS/meta-adc615-arago(see page 13)
    - meta-adc615-userspacehttps://github-am.geo.conti.de/ADAS/meta-adc615-userspace(see page 14)
- 如何进行Yocto构建(see page 15)
- 如何使用Yocto进行开发&维护(see page 16)
    - 修改u-boot和kernel(see page 16)
        - 1.直接修改u-boot仓库的代码(see page 16)
        - 2.打补丁方式修改u-boot(see page 16)
        - 补充：添加kernel自动加载模块(see page 17)
    - 为rootfs中添加内容(see page 17)
        - 添加package(see page 17)
        - 添加系统配置及文件(see page 18)
            - 修改以太网配置(see page 18)
            - 添加.profile(see page 18)
    - 如何添加开源库(see page 19)
    - 注意事项(see page 20)
    - 有用的小窍门(see page 20)

1 Yocto所有git仓库

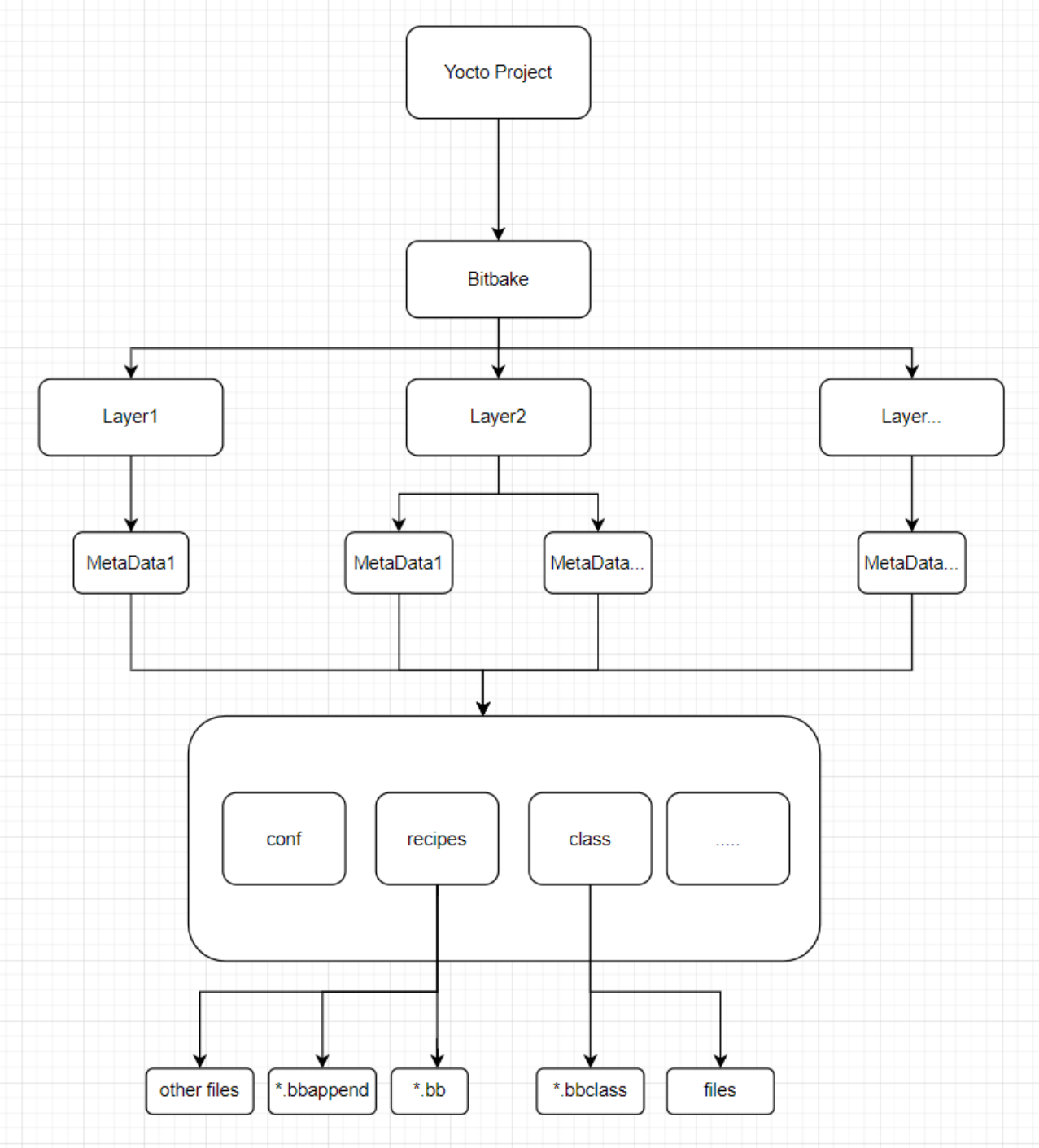
仓库地址	默认主分支
Yocto主仓库 https://github-am.geo.conti.de/ADAS/adc615-yocto	master
Layer仓库 https://github-am.geo.conti.de/ADAS/meta-conti-adas	master
Layer仓库 https://github-am.geo.conti.de/ADAS/meta-adc615-userspace	master
Layer仓库 https://github-am.geo.conti.de/ADAS/meta-adc615-arago	master

镜像仓库

镜像仓库地址
https://github-am.geo.conti.de/ADAS/bitbake
https://github-am.geo.conti.de/ADAS/meta-arago
https://github-am.geo.conti.de/ADAS/meta-psdkla
https://github-am.geo.conti.de/ADAS/meta-qt5
https://github-am.geo.conti.de/ADAS/meta-virtualization
https://github-am.geo.conti.de/ADAS/meta-openembedded
https://github-am.geo.conti.de/ADAS/meta-ti
https://github-am.geo.conti.de/ADAS/meta-arm
https://github-am.geo.conti.de/ADAS/oe-core
https://github-am.geo.conti.de/ADAS/meta-aws-mirror

镜像仓库地址
https://github-am.geo.conti.de/ADAS/meta-jupyter
https://github-am.geo.conti.de/ADAS/core-secdev-k3

2 Yocto基本介绍


Yocto Project
Bitbake
Layer1
Layer2
Layer...
MetaData1
MetaData1
MetaData...
MetaData...
conf
recipes
class
.....
other files
*.bbappend
*.bb
*.bbclass
files

Layer

Yocto是由多个layer构成，一个layer是很多metadata的集合。layer的划分实现了target分离，比如ADC615中有由最基础的target: adc615-platform-image，同时也有adc615-platform-image-deployment等等，而这些不同的target需要添加的组件是不同的，因此可以让不同的target在不同的layer中进行定制。

MetaData

Yocto Project在构建Linux发行版本时，构建系统会进行解析的文件（数据），Metadata是Yocto Project中很关键的一个元素。通常来说，元数据包含recipe，配置conf文件，以及控制构建什么/如何构建（what and how）的数据。

conf

包含全局定义变量，用户定义变量和硬件配置信息。

recipes

元数据最常见的形式。Recipe可以包含一列用来构建二进制镜像文件的设定和任务（指令）。Recipe描述了你从哪获取代码，需要应用哪个（代码）补丁。同时Recipe还描述了对于其他Recipe或库的依赖，以及配置和编译选项。Recipe存放在Layer中

class

类似C++的类概念，包含不同recipe之间的通用操作，需要被bb或bbappend文件include, inherit, require才可以执行其内的操作

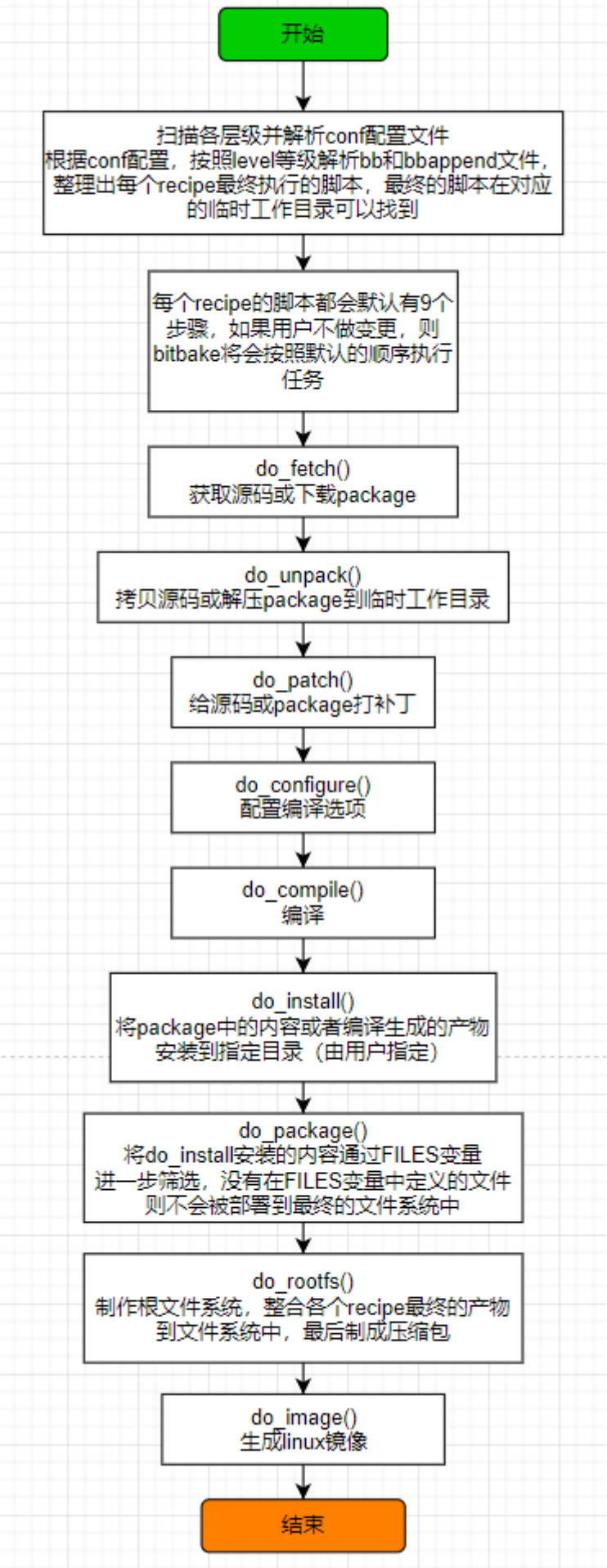
bb

recipe的描述文件，编写语法类似shell脚本，但是yocto添加了一些特殊语法，比如后缀加_append或者 remove可以添加或移除变量列表中的内容

bbappend

追加到同名的bb文件中，如果有同名的变量或者函数将会覆盖到bb文件中

3 Yocto 构建过程


开始
扫描各层级并解析conf配置文件
根据conf配置，按照level等级解析bb和bbappend文件，
整理出每个recipe最终执行的脚本，最终的脚本在对应
的临时工作目录可以找到
每个recipe的脚本都会默认有9个
步骤，如果用户不做变更，则
bitbake将会按照默认的顺序执行
任务
do_fetch()
获取源码或下载package
do_unpack()
拷贝源码或解压package到临时工作目录
do_patch()
给源码或package打补丁
do_configure()
配置编译选项
do_compile()
编译
do_install()
将package中的内容或者编译生成的产物
安装到指定目录（由用户指定）
do_package()
将do_install安装的内容通过FILES变量
进一步筛选，没有在FILES变量中定义的文件
则不会被部署到最终的文件系统中
do_rootfs()
制作根文件系统，整合各个recipe最终的产物
到文件系统中，最后制成压缩包
do_image()
生成linux镜像
结束

4 Yocto官方文档

https://docs.yoctoproject.org/

5 ADC615 Yocto项目简介

5.1 adc615-yocto[1]

ADC615 Yocto工程的构建入口，在README中可以看到构建指令以及选项

- **scripts/oe-layertool-setup.sh**：用于创建yocto构建的环境，比如bitbake构建环境
- **scripts/git_retry.sh**: 主要用于git拉取代码失败后重试。在oe-layertool-setup.sh里调用
- **configs/***：主要存放layer的repo信息，告诉yocto需要拉取哪些layer仓库

5.2 meta-conti-adas[2]

基于Yocto的层级概念创建的meta layer，是ADC615平台最主要的层级，包含以下内容

Dir/ Recipes	Components	Purpose
classes	fdu-scripts	fdu-container.bbclass中会把该目录的update_rootfs.sh脚本打包到最终生成的包中，flashtool将会调用该脚本进行rootfs的刷写
	fdu-container.bbclass	用于制作flashtool支持的container包
	tispl_gen.bbclass	用于生成自定义的tispl.bin
conf	machine	定义machine相关的环境变量，如SOC类型、CPU架构等
	layer.conf	定义当前layer的基本环境变量，如BBFILES : 用于添加.bb文件的搜索路径
recipes-apps	conti-logdump	用于构建在A核中查看其他核心打印的log的程序
	fast-development-update	用于构建与部署flashtool在板子上的环境
recipes-bsp	PowerVR Rogue	Imagination Tech公司的图形处理器，TDA4中集成的SDK就是这个GPU（用于图像计算），所以会用到它的驱动

1 https://github-am.geo.conti.de/ADAS/adc615-yocto
2 https://github-am.geo.conti.de/ADAS/meta-conti-adas

	ti-sci-fw	TI system control interface, 用于构建控制TI跨处理器架构的硬件资源的接口，tiboot3.bin在这里生成
	ti-syscfg	用于进行TDA4多核系统资源配置，这里会影响tiboot.bin的编译
	u-boot	指定了uboot的仓库、分支、补丁等信息，定义了uboot的编译、打包流程 uboot.img和tispl.bin是由该recipe生成的，如果配置了meta-adc615-userspace这个layer，tispl.bin将会被它修改
recipes-connectivity	mobile-broadband-provider-info	这个package包含移动宽带提供商的信息，以及各种网络类型的配置文件，它可以用来配置移动宽带网络的连接
recipes-core	base-files	用于在根文件系统添加文件，启动logo以及分区表等
	eth-if-config	用于配置eth相关配置，如指定系统中的网络接口名称
	images	用于定义构建的所需镜像，以及构建该镜像所需的组件
	packagegroups	定义了需要安装到rootfs的软件包
	systemd	用于定义systemd相关组件构建规则的目录，systemd是linux的系统和服务管理器，目前在这个目录中只额外添加了eth相关的配置（eth0 ip 192.168.0.199等)
	systemd-service	配置systemd服务，目前添加了多核启动的服务，用于在linux系统中启动R核等其他核心的firmware
recipes-devtool	dtbocfg	Device tree overlay 该工具可以在不重新编译内核或重新启动系统的情况下添加新的设备树配置或修改设备树
	input-utils	用于开发、调试输入设备相关的程序
	python	用于部署构建过程中使用到的python工具，比如container tool, 用于最后打包完整软件包时使用。
recipes-graphics	libgles	OpenGL ES和EGL标准库，帮助应用程序利用GPU进行图形处理
recipes-kernel	linux-ti-staging	存放kernel需要添加的defconfig配置项、设备树、以及补丁等 kernel的Image和dtb在这里生成的

recipes-support	devmem-conti	kernel module，用于在用户空间访问内存，由conti-logdump使用
	devpcietp-ep	kernel module， 用于用户空间调试PCIE
	xxhash	为了解决旧版本分支不对的问题而添加的
recipes-ti	ti-rtos-bin	用于部署rtos固件
wic		该文件夹是yocto中创建image文件的工具，这些image可以用于将文件系统和软件包部署到物理设备上，wic工具使用.wks的文件作为模板，该类文件描述了image的大小、分区、文件系统、软件包等信息。wic可以自动将所需的文件系统和软件包根据wks的描述配置到image中

5.3 meta-adc615-arago[3]

仿照TI 的meta-arago构建的发行版layer，用于添加发行版所需的组件

Dir/Recipes	Components	Purpose
classes	certs	存放根证书文件，Conan用这个文件来验证远程服务器的SSL证书是否有效
	profiles	存放conan配置文件
	conan-conti.bbclass	定义了conan部署package的函数
conf		和meta-conti-adas的conf类似
recipes-apps	dwarves	用于程序调试，添加debug symbol
	libepf	用于处理eBPF的库
recipes-devtools	gdb	添加gdb
	python	用于部署构建过程中使用的python库
recipes-rootfs	set-environment	向rootfs中添加.profile文件

3 https://github-am.geo.conti.de/ADAS/meta-adc615-arago

	procps	用于添加linux系统配置，目前添加了coredump存放路径
	sytemd-services	向rootfs中添加linux服务，目前有loopback-multicast-setup、phc2sys、ptp4l
scripts	python3-bitbakery.py	BitBake Recipe Generator for PyPI packages

5.4 meta-adc615-userspace[4]

目前该layer仅用于从artifactory上通过conan拉取IU的elf文件，用来重新构建tispl.bin

后续会添加其他核心的固件

4 https://github-am.geo.conti.de/ADAS/meta-adc615-userspace

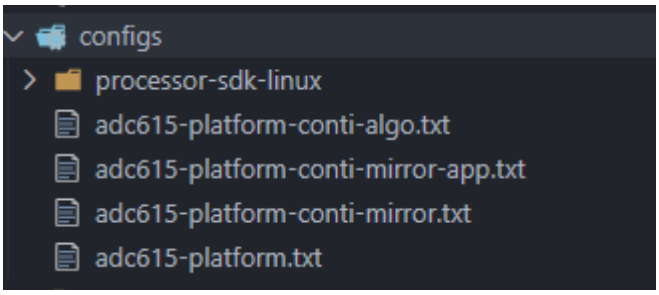
6 如何进行Yocto构建

```
## 第一步，进入adc615-yocto目录执行以下指令，其中config_file是如图1的configs目录下的任意txt文件
./build.sh -c configs/<config_file>
## 第一步如果遇到meta-conan仓库拉取失败的报错，则需要手动去掉proxy，如下
## unset http_proxy https_proxy ALL_PROXY all_proxy

## 第二步，进入第一步生成的build目录执行以下指令，点后面需要有空格（注意这一步建议重新开终端恢复proxy，因为这一步之后的构建又需要proxy了）
. conf/setenv

## 第三步，进行构建，<machine>表示当前机器名，目前只有adc615；<-k>表示遇到报错也接着往下执行，这个option可以不加，<target>表示构建的目标镜像名，详细列表参考git仓库的README
MACHINE=<machine> bitbake -k <target>
```

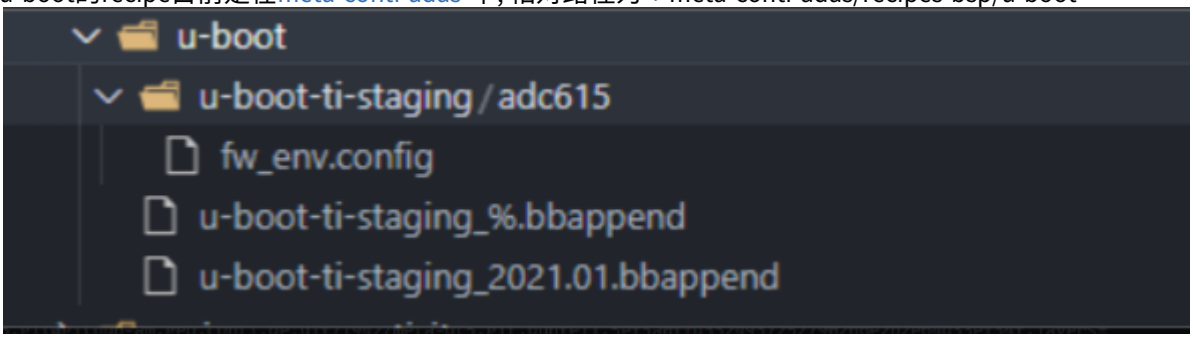
图1



7 如何使用Yocto进行开发&维护

7.1 修改u-boot和kernel

修改uboot和kernel的方式基本一致，这里以u-boot举例：
u-boot的recipe目前是在meta-conti-adas[5]中, 相对路径为：meta-conti-adas/recipes-bsp/u-boot



有两种方法可以修改u-boot代码：

7.1.1 1.直接修改u-boot仓库的代码

打开u-boot-ti-staging_2021.01.bbappend文件，修改两个参数即可：
BRANCH = "master"， 这里修改分支
SRCREV = "839661c28ab02ba1881127c1cdb73ecc4af68aea"， 这里修改BRANCH对应的commit
修改完bbapend文件后，提交改动到meta-conti-adc615仓库，同时需要把最新的meta-conti-adc615的commit更新到adc615_yocto_project中要使用的layer配置文件中，如configs/adc615-platform-conti-mirror.txt配置文件

```
configs > adc615-platform-conti-mirror.txt
1   # This file takes repo entries in the format
2   # repo name,repo uri,repo branch,repo commit[,layers=layer1:layer2...:layern]
3
4   bitbake,git@github-am.geo.conti.de:uif71942/bitbake.git,1.46,e3db9c2e9eded3c5cb6040714a6054b44f6b3880
5   meta-arago,git@github-am.geo.conti.de:uif71942/meta-arago.git,dunfell,08.05.00.005,layers=meta-arago-distro:meta-arago-extras
6   meta-psdkla,git@github-am.geo.conti.de:uif71942/meta-psdkla.git,master,06a6260f965ad7765581878c52b0bed894103847,layers=
7   meta-qt5,git@github-am.geo.conti.de:uif71942/meta-qt5.git,dunfell,5ef3a0ffd3324937252790266e2b2e64d33ef34f,layers=
8   meta-virtualization,git@github-am.geo.conti.de:uif71942/meta-virtualization.git,dunfell,beea119eb529b4a11f266004aee8b548427aea39,layers
9   meta-openembedded,git@github-am.geo.conti.de:uif71942/meta-openembedded.git,dunfell,7203130ed8b58c0df75cb72222ac2bcf546bce44,layers=met
10  meta-ti,git@github-am.geo.conti.de:uif71942/meta-ti.git,dunfell,08.05.00.005,layers=
11  meta-arm,git@github-am.geo.conti.de:uif71942/meta-arm.git,dunfell,c4f04f3fb66f8f4365b08b553af8206372e90a63,layers=meta-arm:meta-arm-toc
12  oe-core,git@github-am.geo.conti.de:uif71942/oe-core.git,dunfell,1ee082e979baaba871bbe1d91181bb04951faf3b,layers=meta
13  meta-aws,git@github-am.geo.conti.de:uif71942/meta-aws.git,dunfell,49e7a5ffb8799b9eee638a6a5425a13b00c33215,layers=
14  meta-jupyter,git@github-am.geo.conti.de:uif71942/meta-jupyter.git,master,82db248e2eca36e52cbeafe599b755dc87720383,layers=
15
16  meta-conti-adc615,git@github-am.geo.conti.de:uif71942/meta-conti-adc615.git,master,725dca1ffc7d35f9e2a49faa32efa696bf34c1b2,layers=
17  meta-adc615-arago,git@github-am.geo.conti.de:uif71942/meta-adc615-arago.git,master,4cd5e924e3ca1270f4a7b1d9c6078b246636455e,layers=
18  meta-conan,https://github.com/conan-io/meta-conan.git,master,0eaff4f40403f5fc0583cb43b86ba5610fdfecee,layers=
19
20  OECORELAYERCONF=./sample-files/bblayers.conf.sample
21  OECORELOCALCONF=./sample-files/local-arago64.conf.sample
22
```

7.1.2 2.打补丁方式修改u-boot

这种方式仅需修改meta-conti-adc615和adc615_yocto_project仓库，步骤如下：

- 制作u-boot补丁

5 https://github-am.geo.conti.de/ADAS/meta-conti-adas

修改完代码后：

```
git add .
git commit -m "xxxx"
git format-patch -1
```

- 打补丁，在bbapend文件内加入补丁文件路径即可，如下：

```
#补丁文件存放路径：
# meta-conti-adc615/recipes-bsp/u-boot/u-boot-ti-staging/adc615/xxx.patch
# u-boot-ti-staging_2021.01.bbappend:
SRC_URI_append_adc615 = " \
  file://adc615/fw_env.config \
  file://adc615/xxx.patch \
"
```

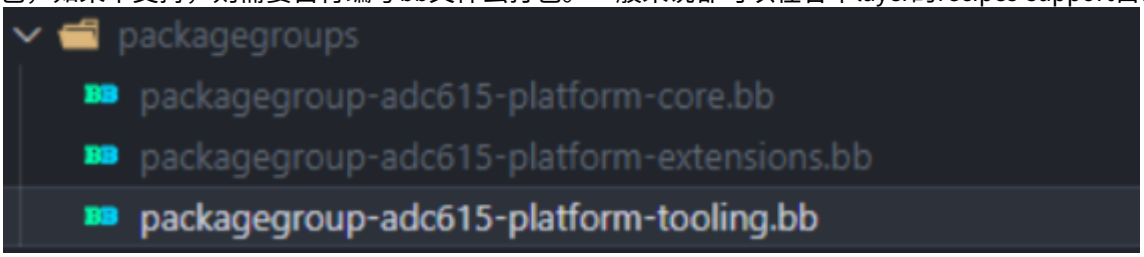
7.1.3 补充：添加kernel自动加载模块

```
# 在linux-ti-staging_5.10.bbappend文件内添加如下声明即可
KERNEL_MODULE_AUTOLOAD += "xxxmodules"
```

7.2 为rootfs中添加内容

7.2.1 添加package

目前添加package都统一在同一个地方meta-conti-adas/recipes-core/packagegroups，仅需在bb文件内添加包名，yocto会自动把包加入rootfs（不过添加前需要确认当前的yocto中的各个recipes是否支持当前想要添加的包，如果不支持，则需要自行编写bb文件去打包。一般来说都可以在各个layer的recipes-support目录下找到）



7.2.2 添加系统配置及文件

由于rootfs内容很多，这里只举两个例子，基本上想要添加文件或配置只需要添加bb文件，在bb文件中的do_install函数把想要添加的文件install到对应的目录即可

7.2.2.1 修改以太网配置

在meta-conti-adc615/recipes-core/systemd中添加了以太网的配置，这里可以修改eth0网卡相关的配置：

```
# 10-eth.network
[Match]
Name=eth0
KernelCommandLine=!root=/dev/nfs
[Network]
Address=192.168.0.199
```

7.2.2.2 添加.profile

```
# meta-adc615-arago/recipes-rootfs/environment/files/.profile
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
alias J5='ssh root@192.168.0.10'
alias j5='J5'

# meta-adc615-arago/recipes-rootfs/environment/set-environment.bb
DESCRIPTION = "Linux set user env"
LICENSE = "CLOSED"
# 将文件添加至SRC_URI列表
SRC_URI_append = "\
```

```
  file://.profile \
"
# 将文件拷贝至临时rootfs目录
do_install() {
  install -d ${D}${ROOT_HOME}/
  install -m 0644 ${WORKDIR}/.profile ${D}${ROOT_HOME}/
}
# 指定最终要安装到rootfs的文件列表
FILES_${PN} = "\
  ${ROOT_HOME}/.profile \
"
```

7.3 如何添加开源库

```
# DESCRIPTION : 第三方开源库的描述，讲述这库是做什么的
DESCRIPTION = "The glog library implements application-level logging. This \
library provides logging APIs based on C++-style streams and various helper \
macros."

# HOMEPAGE ：第三方库的官网链接
HOMEPAGE = "https://github.com/google/glog"

# LICENSE : 第三方库使用的license类型，
LICENSE = "BSD-3-Clause"

# LIC_FILES_CHKSUM ：描述license文件类型,licenes文件的md5校验码
LIC_FILES_CHKSUM = "file://COPYING;md5=dc9db360e0bbd4e46672f3fd91dd6c4b"

# SRC_URI : 描述软件包的下载链接，下载分支，下载时使用的传输协议类型，在该链接中可以patch的形式添加软件包的更新
SRC_URI = " \
    git://github.com/google/glog.git;nobranch=1;protocol=https \
    file://0001-Rework-CMake-glog-VERSION-management.patch \
    file://0002-Find-Libunwind-during-configure.patch \
    file://0003-installation-path-fix.patch \
"

# SRCREV : 软件包的commit节点
SRCREV = "a6a166db069520dbbd653c97c2e5b12e08a8bb26"

# S : 声明clone下来的第三方软件的目录地址
S = "${WORKDIR}/git"

# inherit cmake : 第三方软件使用cmake构建部署时，在bb文件中声明此项，yocto会自动调用cmake将软件构建部署
inherit cmake

# PACKAGECONFIG ：声明该软件需要使能或禁止的属性
PACKAGECONFIG ?= "shared unwind"
```

```
# PACKAGECONFIG_remove_riscv32 = "unwind" : 移除 uwind对riscv32的支持
PACKAGECONFIG_remove_riscv64 = "unwind"
PACKAGECONFIG_remove_riscv32 = "unwind"

PACKAGECONFIG[unwind] = "-DWITH_UNWIND=ON,-DWITH_UNWIND=OFF,libunwind,libunwind"
PACKAGECONFIG[shared] = "-DBUILD_SHARED_LIBS=ON,-DBUILD_SHARED_LIBS=OFF,,"

# do_configure_append() ：在配置部署时,添加自定义的部署属性
do_configure_append() {
    # remove WORKDIR info to improve reproducibility
    if [ -f  "${B}/config.h" ] ; then
        sed -i 's/'$(echo ${WORKDIR} | sed 's_/_\\/_g')'/../g' ${B}/config.h
    fi
}
```

7.4 注意事项

1.添加完自定义的包之后，需要在meta-conti-adas/recipes-core/packagegroups中的bb文件加入自己定义的包名，否则不会生效

2.修改已有的bb或bbappend文件之后，必须使用该指令清除缓存后再重新编译 “ bitbake -c cleanall {recipename or target} “， 否则有可能出现本地测试通过，但别人拉取了你做的改动编译不通过的情况。

3.尽量不要在文件名中使用’_‘下划线作为连接符，使用’-‘减号作为连接符，因为在bitbake眼中，下划线是特殊的符号，代表它需要去特殊处理下划线后的内容

7.5 有用的小窍门

```
# 查找文件关联的package
oe-pkgdata-util find-path /lib/ld-2.24.so

# 打印出recipe可执行的task,
bitbake -c listtasks {recipename}

# 生成recipe构建过程的所有依赖
bitbake -g {recipename}

# 执行某个recipe的某个任务
bitbake -c {task} {recipename}

#cleanall
bitbake -c cleanall {recipename or target}

# 显示执行环境，常用于查找当前bitbake的包的源路径和目标路径
bitbake -e {recipename}
```

```
# 显示构建执行过程
bitbake -v

# 显示当前所有recipe
bitbake -s


# 直接执行.bb
bitbake -b xxx.bb

# 环境变量可以在这个文件找到./sources/oe-core/meta/conf/bitbake.conf
# 当遇到 QA Issue: Package version for package kernel-image-fitimage went backwards
which would break package feeds, 可以在conf中配置ERROR_QA_remove = "version-going-
backwards"

# 查找文件属于哪个package
oe-pkgdata-util find-path xxx.so

# 列出package会安装哪些文件
oe-pkgdata-util list-pkg-files xxxx

# build/buildhistory/images/adc615/glibc/adc615-image-platform/installed-package-
sizes.txt文件可以查看每个包占用的空间
```