YOCTO

Manual for yocto project:

https://docs.yoctoproject.org/1.8/dev-manual/dev-manual.html

Definitions:

https://automotive-4-dummies.blogspot.com/2021/01/learning-yocto.html

Terminology in the Yocto Project can be a little confusing. These definitions should help you along the way:

- OpenEmbedded: build system and community
- The Yocto Project: umbrella project and community
- Metadata: files containing information about how to build an image
- Recipe: file with instructions to build one or more packages
- Layer: directory containing grouped metadata (start with “meta-”)
- Board support package (BSP): layer that defines how to build for board (usually maintained by vendor)
- Distribution: specific implementation of Linux (kernel version, rootfs, etc.)
- Machine: defines the architecture, pins, buses, BSP, etc.
- Image: output of build process (bootable and executable Linux OS)

Youtube reference for build image:

- https://www.digikey.in/en/maker/projects/intro-to-embedded-linux-part-2-yocto-project/2c08a1ad09d74f20b9844e566d332da4

- https://www.youtube.com/watch?v=ygzKiIgycE4

Ppt reference :

https://e-labworks.com/training/en/ypr/slides.pdf

Build poky and qemu demo:

https://wiki.yoctoproject.org/wiki/Transcript:_from_git_checkout_to_meta-intel_BSP – documentation.

https://automotive-4-dummies.blogspot.com/2021/01/learning-yocto-basics-part-2-getting.html – demo video

Steps to build an image for x86:

- sudo apt update
- sudo apt upgrade
- sudo apt install -y bc build-essential chrpath cpio diffstat gawk git texinfo wget gdisk python3 python3-pip
- sudo apt install -y libssl-dev
- sudo apt-get install openssl
- vi ~/.bashrc [within this file we have to add a line "alias python=python3"]

- source ~/.bashrc
- python --version
- mkdir yocto
- cd yocto
- git clone git://git.yoctoproject.org/poky.git
- cd poky
- git checkout dunfell [we can replace codename dunfell as kirkstone,zeus etc..]
- git status
- git branch
- cd ../ -->(it will go to previous yocto dir)
- source poky/oe-init-build-env build
- bitbake-layers show-layers
- bitbake core-image-minimal
- runqemu or runqemu qemux86
- image will display, login- root

IMX6 -> steps to build an image on imx6 using repo

https://www.nxp.com/docs/en/user-guideIMX_YOCTO_PROJECT_USERS_GUIDE.pdf

- sudo apt-get install gawk wget git diffstat unzip texinfo gcc-multilib build-essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev xterm rsync curl zstd lz4 libssl-dev

- mkdir ~/bin (this step may not be needed if the bin folder already exists)
- curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
- chmod a+x ~/bin/repo
- ls -la ~/ | more (to open bashrc file and add "export PATH=~/bin:$PATH")
- git config --global user.name "Your Name"
- git config --global user.email "Your Email"
- git config --list
- sudo apt-get install python-is-python3
- mkdir imx-yocto-bs
- cd imx-yocto-bsp
- repo init -u https://github.com/nxp-imx/imx-manifest
  -b imx-linux-kirkstone -m imx-5.15.71-2.2.0.xml
- repo sync
- DISTRO=poky MACHINE=imx6sxsabreauto source imx-setup-release.sh -b yocto
- bitbake core-image-minimal

video for flashing an image using balena software

https://www.youtube.com/watch?v=kWRx40Q8B_A

IMX6 -> steps to build an image on imx6 using git (its working)

https://www.youtube.com/watch?v=ygzKiIgycE4

step1: cd yocto
step2:

- git clone -b dunfell https://github.com/freescale/meta-freescale.git

- git clone -b dunfell https://github.com/freescale/meta-freescale-distro.git
- git clone -b dunfell https://github.com/freescale/meta-freescale-3rdparty.git
- git clone git://git.yoctoproject.org/poky.git

step3: source poky/oe-init-built-env

[after giving above cmd the current dir will be -> yocto/build]

step3: cd conf and vi bblayers.conf
[pwd-> yocto/build/conf/bblayers.conf]
add below dir to bblayers.conf file:

```
"
  /home/tejaswini/yocto/meta-freescale \
  /home/tejaswini/yocto/meta-freescale-3rdparty \
  /home/tejaswini/yocto/meta-freescale-distro \
"
```

step4: vi local.conf [pwd-> yocto/build/conf/local.conf]

add machine name as "MACHINE = "imx6qdlsabreauto""

step5: cd ../ [pwd -> yocto/build]
bitbake core-image-minimal

step6: booting and flashing using balena software ,

- before install balena software, have to install 2 packages ->
  1. sudo add-apt-repository universe
  2. sudo apt install libfuse2
- install balena software,here is the link for software -> https://www.balena.io/etcher#download-etcher
- after downloading,
  1. go to file manager -> downloads here balena should be there.
  2. Right click on balena -> go to properties -> go to permissions -> give permission for execute check box.
- How to use balena software
  1. first connect sd card to pc.
  2. Run balena software.
  3. Select image which is in this directory - */home/tejaswini/yocto/build/tmp/deploy/images/sama5d2-icp-sd/*
  4. *select this image -> core-image-minimal-sama5d27-som1-ek-sd-20230421120543.rootfs.wic*

5. *next select target which means usb stick or sd card.*
6. *Finally click on flash.*

step7:

- after flashing, remove sd card from pc and connect it to board.
- Do this cmd for installing picocom,

--> sudo apt-get install picocom

- Then Connect pc and board by usb, after give below command in terminal.
- sudo picocom -b 115200 *dev/*ttyACM0
- login: root
  https://www.youtube.com/watch?v=3lR6frxLxCc

wayland reference

https://github.com/eisslec/Accelerated-Wayland-for-IMX6/blob/master/Documentation/manual-build-documentation.txt

EGT (ensemble graphical toolkit):

https://github.com/linux4sam/egt -> git repository

STEPS TO BUILD AN IMAGE FOR SAMA5D27-SOM1-EK-SD

step 1.1

- sudo apt update
- sudo apt upgrade
- sudo apt-get install gawk wget git diffstat unzip texinfo gcc-multilib build-essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev xterm rsync curl zstd lz4 libssl-dev

- sudo apt install -y libssl-dev
- sudo apt-get install openssl

step 1.2:

- open bashrc file in home directory -> vi ~/.bashrc
- within bashrc file we need to add this line at the end and save file ->
  1. alias python=python3
  2. export PATH=~/bin:$PATH
- source ~/.bashrc
- python –-version ->this cmd is for checking python version.

step 1.3:

- mkdir yocto
- cd yocto

step2:

1. Clone yocto/poky git repository with the proper branch ready
git clone https://git.yoctoproject.org/poky

2. Clone meta-openembedded git repository with the proper branch ready
git clone https://git.openembedded.org/meta-openembedded -b kirkstone

3. Clone meta-atmel layer with the proper branch ready
git clone https://github.com/linux4sam/meta-atmel.git -b kirkstone

4. Clone meta-arm layer with the proper branch ready
git clone https://git.yoctoproject.org/meta-arm -b kirkstone

step3: source poky/oe-init-built-env (for setting up built environment for building an image)

[after giving above cmd the current dir will be -> yocto/built]

step3: cd conf and vi bblayers.conf
[pwd-> yocto/built/conf/bblayers.conf]
add below dir to bblayers.conf file:
“

```
 /home/tejaswini/yocto/poky/meta-skeleton \
 /home/tejaswini/yocto/meta-arm/meta-arm \
 /home/tejaswini/yocto/meta-arm/meta-arm-toolchain \
 /home/tejaswini/yocto/meta-atmel \
 /home/tejaswini/yocto/meta-openembedded/meta-oe \
 /home/tejaswini/yocto/meta-openembedded/meta-networking \
 /home/tejaswini/yocto/meta-openembedded/meta-python \
```

“

step4: vi local.conf [pwd-> yocto/build/conf/local.conf]

add below lines to local.conf file

- add machine name as “MACHINE = "sama5d27-som1-ek-sd"”
- CORE_IMAGE_EXTRA_INSTALL+="can-utils"
- MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS+="can-utils"
- CORE_IMAGE_EXTRA_INSTALL+="kernel-modules"
- MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS+="kernel-modules"

step5: cd ../ [pwd -> yocto/build]

- bitbake core-image-minimal

step6: booting and flashing using balena software ,

- before install balena software, have to install 2 packages ->
  1. sudo add-apt-repository universe
  2. sudo apt install libfuse2
- install balena software,here is the link for software -> https://www.balena.io/etcher#download-etcher
- after downloading,
  1. go to file manager -> downloads ,here balena should be there.
  2. Right click on balena -> go to properties -> go to permissions -> give permission for execute check box.
- How to use balena software
  1. first connect sd card to pc.
  2. Run balena software.
  3. Select image which is in this directory - */home/tejaswini/yocto/build/tmp/deploy/images/sama5d2-icp-sd/*
  4. *select this image -> core-image-minimal-sama5d27-som1-ek-sd-20230421120543.rootfs.wic*
  5. *next select target which means usb stick or sd card.*
  6. *Finally click on flash.*

step7:

- after flashing, remove sd card from pc and connect it to board.
- Do this cmd for installing picocom,

--> sudo apt-get install picocom

- Then Connect pc and board by usb, after give below command in terminal.

- sudo picocom -b 115200 *dev/*ttyACM0
- login: root

Documentation on sama5d2-icp

https://www.linux4sam.org/bin/view/Linux4SAM/Sama5d27Som1EKMainPage

STEPS TO BUILD AN IMAGE FOR SAMA5D2-icp

step 1.1

- sudo apt update
- sudo apt upgrade
- sudo apt-get install gawk wget git diffstat unzip texinfo gcc-multilib build-essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev xterm rsync curl zstd lz4 libssl-dev
- sudo apt install -y libssl-dev
- sudo apt-get install openssl

step 1.2:

- open bashrc file in home directory -> vi ~/.bashrc
- within bashrc file we need to add this line at the end and save file ->
  1. alias python=python3
  2. export PATH=~/bin:$PATH
- source ~/.bashrc -> this cmd is used to re-run the .bashrc script to update our shell
- python –-version ->this cmd is for checking python version.

step 1.3:

- mkdir yocto -> creating yocto directory in the home directory
- cd yocto -> change to yocto directory

step2: These are the required dependenices to be cloned.

1. Clone yocto/poky git repository with the proper branch ready
git clone https://git.yoctoproject.org/poky -b kirkstone

2. Clone meta-openembedded git repository with the proper branch ready
git clone https://git.openembedded.org/meta-openembedded -b kirkstone

3. Clone meta-atmel layer with the proper branch ready
git clone https://github.com/linux4sam/meta-atmel.git -b kirkstone

4. Clone meta-arm layer with the proper branch ready
git clone https://git.yoctoproject.org/meta-arm -b kirkstone

__step3__: source poky/oe-init-built-env (for setting up built environment for building an image)

[after giving above cmd, it will take us to the build directory
-> yocto/build]

__step4__:

1. cd conf
2. vi bblayers.conf (pwd will be -> /yocto/build/conf/bblayers.conf)

add below meta-layers path to bblayers.conf file:

"

/home/tejaswini/yocto/poky/meta-skeleton \
/home/tejaswini/yocto/meta-arm/meta-arm \
/home/tejaswini/yocto/meta-arm/meta-arm-toolchain \
/home/tejaswini/yocto/meta-atmel \
/home/tejaswini/yocto/meta-openembedded/meta-oe \
/home/tejaswini/yocto/meta-openembedded/meta-networking \
/home/tejaswini/yocto/meta-openembedded/meta-python \
/home/tejaswini/yocto/meta-openembedded/meta-initramfs \
/home/tejaswini/yocto/meta-openembedded/meta-webserver \
/home/tejaswini/yocto/meta-openembedded/meta-multimedia \

"

__step5__: vi local.conf [pwd-> yocto/build/conf/local.conf]

- Comment the existing machine name which is MACHINE = "qemux86-64" and add machine name as __MACHINE = "sama5d2-icp-sd"__
- add below lines to local.conf file anywhere we wish to add.

1. CORE_IMAGE_EXTRA_INSTALL+="can-utils"
2. MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS+="can-utils"
3. CORE_IMAGE_EXTRA_INSTALL+="kernel-modules"
4. MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS+="kernel-modules"

- above 4 lines are added for can configuration and kernel modules.

step6: cd ../ [pwd -> yocto/build]

- bitbake core-image-minimal -> this in-built cmd is used to build an image.
- While doing bitbake ,if you get any locale related error then just use this cmd -> LC_ALL=en_US.utf8 bitbake core-image-minimal

step7:booting and flashing using balena software ,

- before installing balena software, have to install 2 packages ->
  1. sudo add-apt-repository universe
  2. sudo apt install libfuse2
- install balena software,here is the link for software -> https://www.balena.io/etcher#download-etcher
- after downloading,
  1. go to file manager -> go to downloads, here balena should be there.
  2. Right click on balena -> go to properties -> go to permissions -> click on execute check box.
- **How to use balena software**
  1. first insert sd card to pc.
  2. Run balena software.
  3. Select the image which will be in below path, ->

/home/tejaswini/yocto/build/tmp/deploy/images/sama5d2-icp-sd/

  4. *select this image -> core-image-minimal-sama5d27-icp-sd-20230421120543.rootfs.wic*
  5. *next select target which means usb stick or sd card.*
  6. *Finally click on flash.*

step8:

- after flashing, remove sd card from pc and connect it to board.
- Do this cmd for installing picocom,

--> sudo apt-get install picocom

- Then Connect pc and board by usb, then give below command in terminal.
- sudo picocom -b 115200 /*dev/*ttyACM0
  login: root

gcc procedure : (cross-compilation)

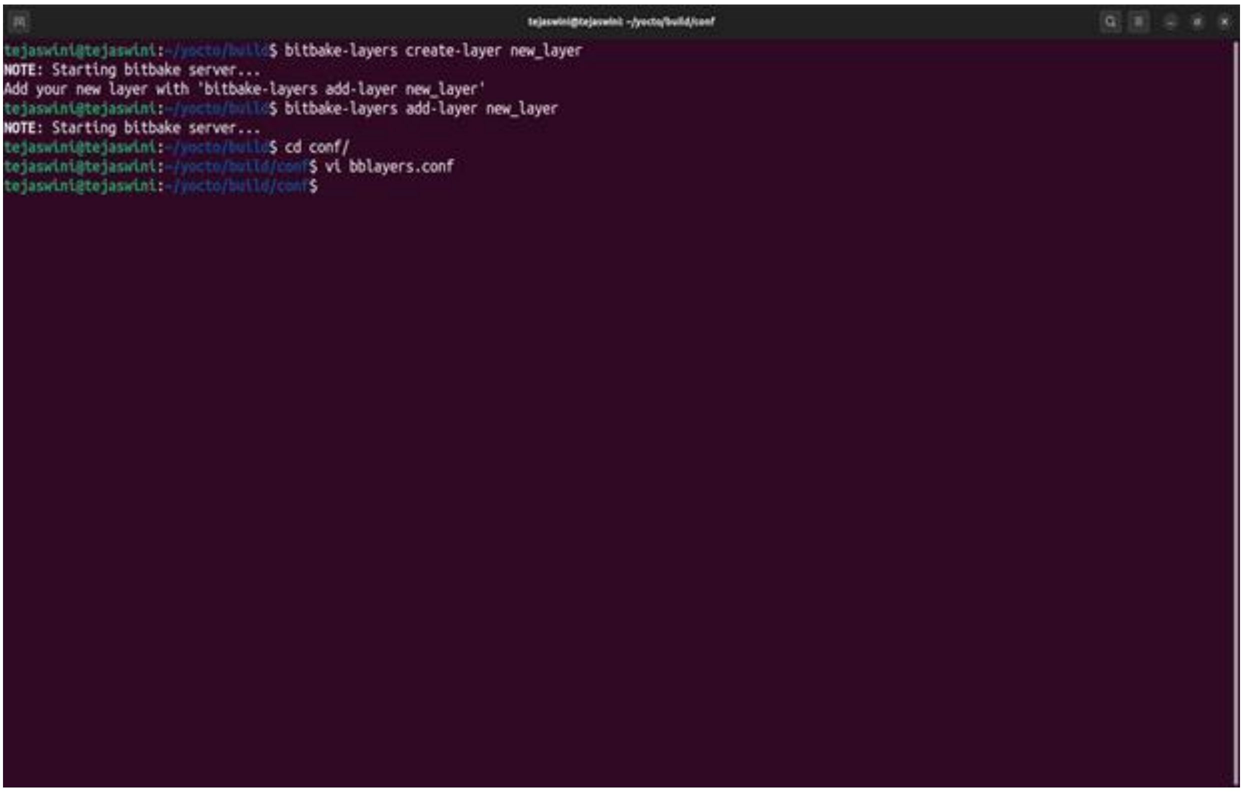
- https://armkeil.blob.core.windows.net/developer/Files/downloads/gnu-rm/10.3-2021.10/gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2

- download the above file & extract file.
- nano ~/.bashrc
- we have to specify where is above tar file is located, like this "export PATH=/home/tejaswini/Downloads/gcc-arm-none-eabi-10.3-2021.10-x86_64-linux/bin:$PATH"
- sudo apt install gcc-arm-linux-gnueabihf
- vi hello.c -> implement C program
- arm-linux-gnueabihf-gcc -o hello hello.c -> it will give executable file
- copy executable file to sd card -> sudo cp hello /media/tejaswini/root/home/root
- sudo picocom -b 115200 /dev/ttyACM0
- ./hello (give this on target machine)

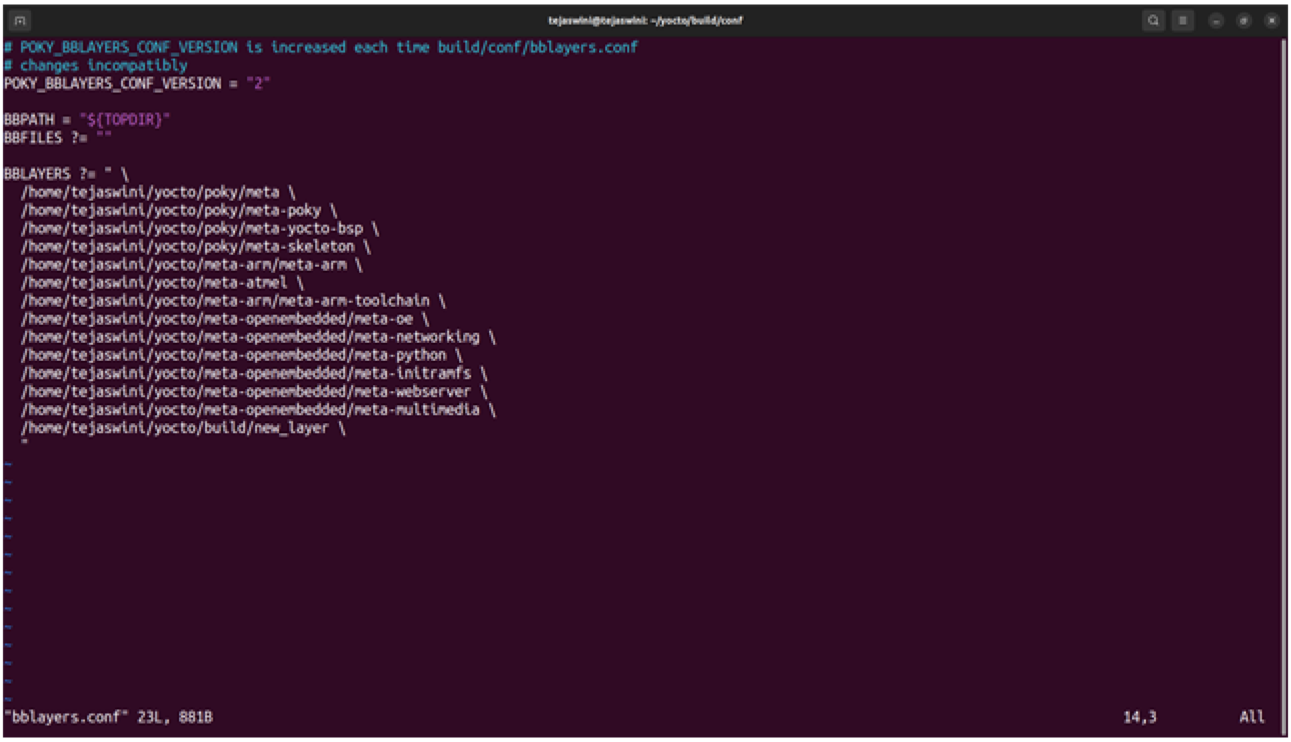
compiling c programs by adding recipes without cross compile

- refer this -> https://george-calin.medium.com/how-to-prepare-a-helloworld-c-recipe-with-yocto-project-1f74c296a777
- https://www.youtube.com/watch?v=3HsaoVqX7dg

1. mkdir yocto and cd yocto (soures directory)
2. clone all dependencies for specific board and add them all in bblayers.conf file(refer the steps in sama5d2-icp for cloning the depenndencies)
3. source poky/oe-init-build-env (for setting up built environment for building an image)
4. bitbake-layers create-layer <layer_name>
5. bitbake-layers add-layer <layer_name> -> it will add layer path in bblayer.conf
6. it will look like the below image after adding layers,(pwd -> /yocto/build/conf/bblayers.conf)



```
tejaswini@tejaswini:~/yocto/build$ bitbake-layers create-layer new_layer
NOTE: Starting bitbake server...
Add your new layer with 'bitbake-layers add-layer new_layer'
tejaswini@tejaswini:~/yocto/build$ bitbake-layers add-layer new_layer
NOTE: Starting bitbake server...
tejaswini@tejaswini:~/yocto/build$ cd conf/
tejaswini@tejaswini:~/yocto/build/conf$ vi bblayers.conf
tejaswini@tejaswini:~/yocto/build/conf$
```



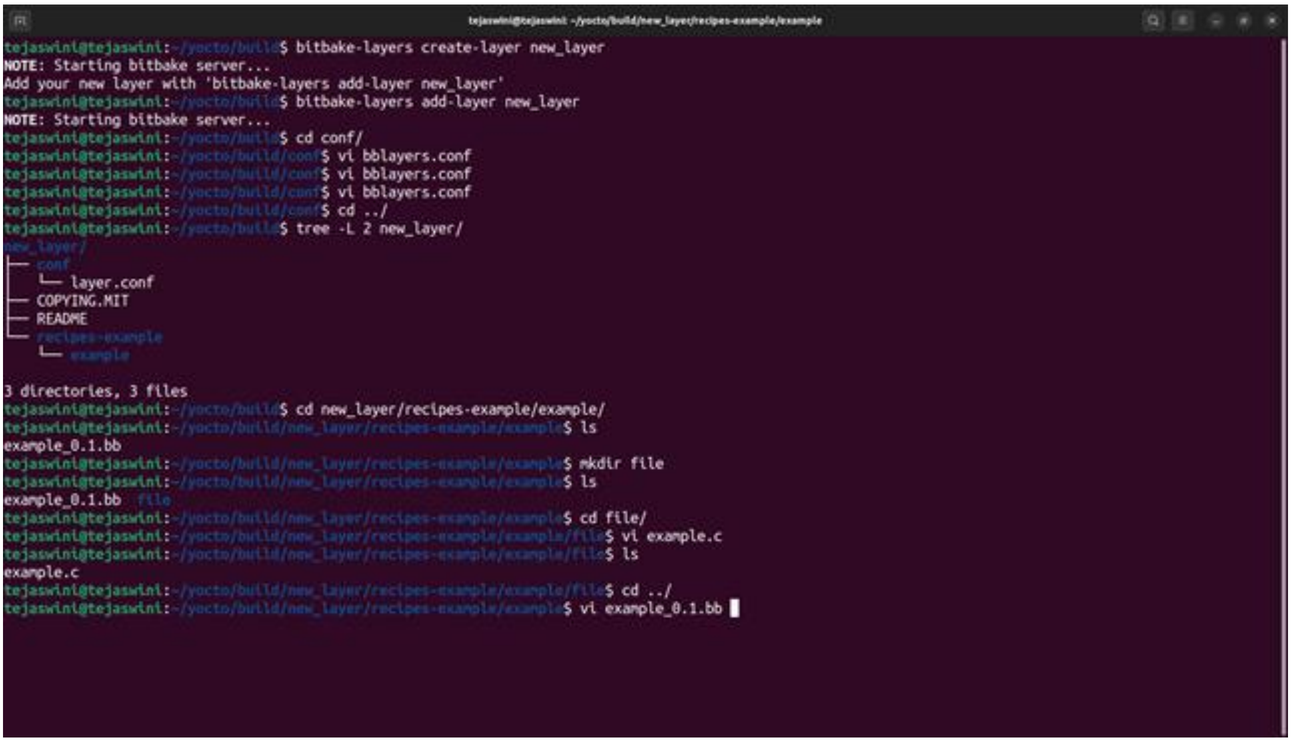
```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
  /home/tejaswini/yocto/poky/meta \
  /home/tejaswini/yocto/poky/meta-poky \
  /home/tejaswini/yocto/poky/meta-yocto-bsp \
  /home/tejaswini/yocto/poky/meta-skeleton \
  /home/tejaswini/yocto/meta-arm/meta-arm \
  /home/tejaswini/yocto/meta-atmel \
  /home/tejaswini/yocto/meta-arm/meta-arm-toolchain \
  /home/tejaswini/yocto/meta-openembedded/meta-oe \
  /home/tejaswini/yocto/meta-openembedded/meta-networking \
  /home/tejaswini/yocto/meta-openembedded/meta-python \
  /home/tejaswini/yocto/meta-openembedded/meta-initramfs \
  /home/tejaswini/yocto/meta-openembedded/meta-webserver \
  /home/tejaswini/yocto/meta-openembedded/meta-multimedia \
  /home/tejaswini/yocto/build/new_layer \
  "
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"bblayers.conf" 23L, 881B                                              14,3          All
```

7. In created recipe, there should be recipe-example directory, in that example directory should be there.
8. Inside example directory, we have to create directory called files, then write c program code in files directory (vi example.c), like this

yocto/built/<layer_name>/recipe-example/example/files/vi example.c\


tejaswini@tejaswini: ~/yocto/build/new_layer/recipes-example/example

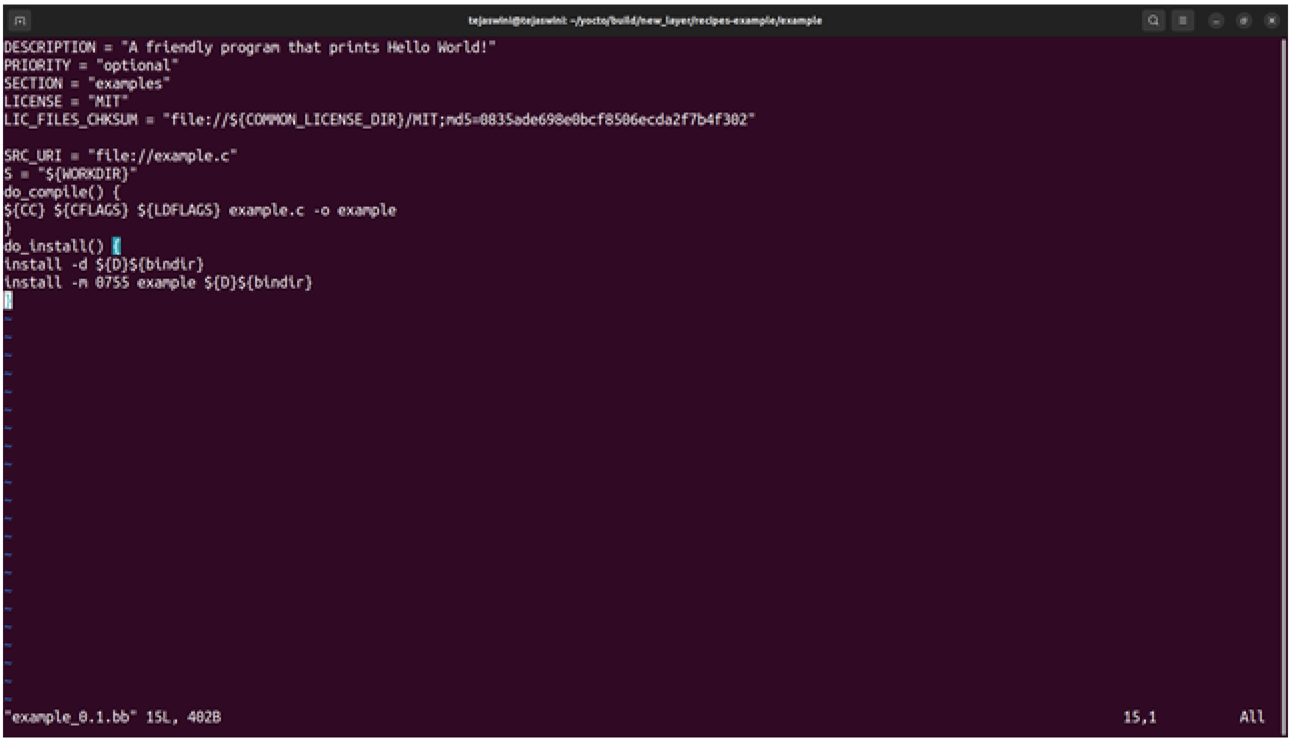
```
tejaswini@tejaswini:~/yocto/build$ bitbake-layers create-layer new_layer
NOTE: Starting bitbake server...
Add your new layer with 'bitbake-layers add-layer new_layer'
tejaswini@tejaswini:~/yocto/build$ bitbake-layers add-layer new_layer
NOTE: Starting bitbake server...
tejaswini@tejaswini:~/yocto/build$ cd conf/
tejaswini@tejaswini:~/yocto/build/conf$ vi bblayers.conf
tejaswini@tejaswini:~/yocto/build/conf$ vi bblayers.conf
tejaswini@tejaswini:~/yocto/build/conf$ vi bblayers.conf
tejaswini@tejaswini:~/yocto/build/conf$ cd ../
tejaswini@tejaswini:~/yocto/build$ tree -L 2 new_layer/
new_layer/
├── conf
│   └── layer.conf
├── COPYING.MIT
├── README
└── recipes-example
    └── example

3 directories, 3 files
tejaswini@tejaswini:~/yocto/build$ cd new_layer/recipes-example/example/
tejaswini@tejaswini:~/yocto/build/new_layer/recipes-example/example$ ls
example_0.1.bb
tejaswini@tejaswini:~/yocto/build/new_layer/recipes-example/example$ mkdir file
tejaswini@tejaswini:~/yocto/build/new_layer/recipes-example/example$ ls
example_0.1.bb  file
tejaswini@tejaswini:~/yocto/build/new_layer/recipes-example/example$ cd file/
tejaswini@tejaswini:~/yocto/build/new_layer/recipes-example/example/file$ vi example.c
tejaswini@tejaswini:~/yocto/build/new_layer/recipes-example/example/file$ ls
example.c
tejaswini@tejaswini:~/yocto/build/new_layer/recipes-example/example/file$ cd ../
tejaswini@tejaswini:~/yocto/build/new_layer/recipes-example/example$ vi example_0.1.bb
```

9. yocto/built/<layer_name>/recipe-example/example/ -> in this directory example_0.1.bb file should be there.
10. Add below lines to example_0.1.bb file

```
DESCRIPTION = "A friendly program that prints Hello World!"
PRIORITY = "optional"
SECTION = "examples"
LICENSE = "MIT"
LIC_FILES_CHKSUM =
"file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"

SRC_URI = "file://example.c"
S = "${WORKDIR}"
do_compile() {
${CC} ${CFLAGS} ${LDFLAGS} example.c -o example
}
do_install() {
install -d ${D}${bindir}
install -m 0755 example ${D}${bindir}
}
```


tejaswini@tejaswini: ~/yocto/build/new_layer/recipes-example/example

```
DESCRIPTION = "A friendly program that prints Hello World!"
PRIORITY = "optional"
SECTION = "examples"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"

SRC_URI = "file://example.c"
S = "${WORKDIR}"
do_compile() {
${CC} ${CFLAGS} ${LDFLAGS} example.c -o example
}
do_install() {
install -d ${D}${bindir}
install -m 0755 example ${D}${bindir}
}

"example_0.1.bb" 15L, 402B                                                                  15,1          All
```

11. Add below lines to local.conf file (pwd-> /yocto/build/conf/local.conf)

CORE_IMAGE_EXTRA_INSTALL+="example"
MACHINE = “sama5d2-icp-sd”

12. In built directory, give command “bitbake example” (pwd -> /yocto/build)
13. Rebuild an image by giving “bitbake core-image-minimal” in same directory.
14. After that, flash by using balena software and boot by giving this cmd “sudo picocom -b 115200 /*dev*/ttyACM0”
15. After login, just give this name we will get output

-> example

CAN driver:

https://www.youtube.com/watch?v=Qb1PS0KQUQA&list=PLERTijJOmYrApVZqiI6gtA8hr1_6QS-cs&index=4 -> concept reference video

Enabling CAN interface on sama5d2-icp

- **If kernel modules is not there in target machine then add this 2 lines to local.conf file (pwd -> /yocto/build/conf/local.conf).**

1. CORE_IMAGE_EXTRA_INSTALL+="kernel-modules"
2. MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS+="kernel-modules"

- Give modprobe <device name> -> to load kernel module.

Ex: modprobe can0

- **Add follwing lines to the local.conf file to enable can interface (pwd -> /yocto/build/conf/local.conf)**

1. IMAGE_INSTALL.append = "linux-canutils"
2. CORE_IMAGE_EXTRA_INSTALL+="can-utils"
3. MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS+="can-utils"

- **CAN cmds for executing can programs**

1. create recipe of can prgm (create executable file, refer -> compiling c programs by adding recipes without cross compile)
2. sudo picocom -b 115200 */dev/ttyACM0*
3. setting up the can interface by below commands
   - ip link set can0 type can bitrate 500000 triple-sampling on
   - ifconfig can0 up
4. give program name.

Enabling USB as HID Gadget(temporary):

- For enabling usb as hid gadget, we need to enable few thing in kernel configuration.
- To open kernel configuration give this cmd in host terminal

-> bitbake -c menuconfig virtual/kernel

- need to follow given path :- click on device drivers -> click on USB support -> click on USB Gadget Support -> disable everything (by pressing n) and enable HID function, USB Gadget functions configurable through configfs, mass storage and function filesystem (by pressing y and save it by pressing on save button) -> click on USB Gadget precomposed configurations -> disable everything (by pressing n) and enable mass storage and function filesystem(by pressing y and save it) -> come out from kernel configuration by clicking on exit button.
- Rebuilt an image by this cmd > bitbake core-image-minimal
- Flash image to the sd card.
- Connect sd card to board and connect usb from board(j16 port) to pc
- In host terminal give this cmd ->

sudo picocom -b 115200 /*dev*/ttyACM0

- login to the board terminal by giving root as login

- After that follow below cmds(board terminal),
    1. modprobe libcomposite
    2. cd /sys/kernel/config
    3. mkdir usb_gadget/g1
    4. cd usb_gadget/g1
    5. mkdir configs/c.1
    6. mkdir functions/hid.usb0
    7. echo 1 > functions/hid.usb0/protocol
    8. echo 1 > functions/hid.usb0/subclass
    9. echo 8 > functions/hid.usb0/report_length
    10. cd functions/hid.usb0
    11. vi report_desc (need to add below descriptor into report_desc file)

```
#!/bin/bash

OFILE=hidreport.bin

echo -ne \\x05\\x01\\x09\\x06\\xa1\\x01\\x05\\x07\\x19\\xe0\\x29\\xe7\\x15\\x00\\x25\\x01 > $OFILE
echo -ne \\x75\\x01\\x95\\x08\\x81\\x02\\x95\\x01\\x75\\x08\\x81\\x03\\x95\\x05\\x75\\x01 >> $OFILE
echo -ne \\x05\\x08\\x19\\x01\\x29\\x05\\x91\\x02\\x95\\x01\\x75\\x03\\x91\\x03\\x95\\x06 >> $OFILE
echo -ne \\x75\\x08\\x15\\x00\\x25\\x65\\x05\\x07\\x19\\x00\\x29\\x65\\x81\\x00\\xc0 >> $OFILE
```

12. cd ../../ (pwd -> /sys/kernel/config/usb_gadget/g1)
13. mkdir strings/0x409
14. mkdir configs/c.1/strings/0x409
15. echo 0xa4ac > idProduct
16. echo 0x0525 > idVendor
17. echo serial > strings/0x409/serialnumber
18. echo capgemini > strings/0x409/manufacturer
19. echo "HID Gadget" > strings/0x409/product
20. echo "Conf 1" > configs/c.1/strings/0x409/configuration
21. echo 120 > configs/c.1/MaxPower
22. ln -s functions/hid.usb0 configs/c.1
23. ls /sys/class/udc
24. echo 300000.gadget > UDC

- after all these steps,connect b-type usb cable from board(j9 port) to pc
- open new terminal and give this cmd -> lsusb(it will display usb hid gadget)

Enabling USB as HID Gadget(permanent):

- For enabling usb as hid gadget, we need to enable few thing in kernel configuration.
- To open kernel configuration give this cmd in host terminal

-> bitbake -c menuconfig virtual/kernel

- need to follow given path :- click on device drivers -> click on USB support -> click on USB Gadget Support -> disable everything (by pressing n) and enable HID function, USB Gadget functions configurable through configfs, mass storage and function filesystem (by pressing y and save it by pressing on save button) -> click on USB Gadget precomposed configurations -> disable everything (by pressing n) and enable mass storage and function filesystem(by pressing y) -> come out from kernel configuration by clicking on exit button.
- Rebuilt an image by this cmd > bitbake core-image-minimal
- Flash image to the sd card.
- Connect sd card to board and connect usb from board(j16 port) to pc
- In host terminal give this cmd ->

sudo picocom -b 115200 /*dev*/ttyACM0

- login to the board terminal by giving root as login
- in borad terminal,
- cd /*etc*/init.d
- create new script file on this dir (pwd->/etc/init.d),add below content in that file and save it as filename.sh file

```
#!/bin/bash
HIDREPORTBIN=/tmp/hidreport.bin

echo -ne \\x05\\x01\\x09\\x06\\xa1\\x01\\x05\\x07\\x19\\xe0\\x29\\xe7\\x15\\x00\\x25\\x01 > $HIDREPORTBIN
echo -ne \\x75\\x01\\x95\\x08\\x81\\x02\\x95\\x01\\x75\\x08\\x81\\x03\\x95\\x05\\x75\\x01 >> $HIDREPORTBIN
echo -ne \\x05\\x08\\x19\\x01\\x29\\x05\\x91\\x02\\x95\\x01\\x75\\x03\\x91\\x03\\x95\\x06 >> $HIDREPORTBIN
echo -ne \\x75\\x08\\x15\\x00\\x25\\x65\\x05\\x07\\x19\\x00\\x29\\x65\\x81\\x00\\xc0 >> $HIDREPORTBIN


modprobe libcomposite
#cd /sys/kernel/config

if [ -d "/sys/kernel/config/usb_gadget/g1" ]; then
   echo "g1 Directory already exists."
else
   mkdir "/sys/kernel/config/usb_gadget/g1"
   echo "Directory created."

fi

#cd usb_gadget/g1

if [ -d "/sys/kernel/config/usb_gadget/g1/configs/c.1" ]; then
   echo "configs/c.1 Directory already exists."
else
   mkdir "/sys/kernel/config/usb_gadget/g1/configs/c.1"
   echo "Directory created."
```

```
fi

#cd usb_gadget/g1

if [ -d "/sys/kernel/config/usb_gadget/g1/functions/hid.usb0" ]; then
   echo "functions/hid.usb0 Directory already exists."
else
   mkdir "/sys/kernel/config/usb_gadget/g1/functions/hid.usb0"
   echo "Directory created."

fi
#cd usb_gadget/g1

cd /sys/kernel/config/usb_gadget/g1

echo 1 > functions/hid.usb0/protocol
echo 1 > functions/hid.usb0/subclass
echo 8 > functions/hid.usb0/report_length
cat $HIDREPORTBIN > functions/hid.usb0/report_desc

#cd usb_gadget/g1

if [ -d "/sys/kernel/config/usb_gadget/g1/strings/0x409" ]; then
   echo "strings/0x409 Directory already exists."
else
   mkdir "/sys/kernel/config/usb_gadget/g1/strings/0x409"
   echo "Directory created."

fi

if [ -d "/sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409" ]; then
   echo "configs/c.1/strings/0x409 Directory already exists."
else
   mkdir "/sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409"
   echo "Directory created."

fi

#mkdir configs/c.1/strings/0x409

echo 0xa4ac > idProduct
echo 0x0525 > idVendor
```

```
cd /sys/kernel/config/usb_gadget/g1
echo serial > strings/0x409/serialnumber
echo capgemini > strings/0x409/manu0facturer
echo "HID Gadget" > strings/0x409/product

cd /sys/kernel/config/usb_gadget/g1

echo "Conf 1" > configs/c.1/strings/0x409/configuration
echo 120 > configs/c.1/MaxPower
ln -s functions/hid.usb0 configs/c.1
echo 300000.gadget > UDC
```

- save above file in .sh format
- give this cmd -> update-rc.d <filename.sh> defaults
- then reboot
- open new terminal and give this cmd -> lsusb(it will display usb hid gadget)

Communication through USB as HID:

- after enabling usb as hid gadget, using c program we can communicate.
- First need to install some packages,
    1. sudo apt-get install libhidapi-dev
    2. sudo apt-get install libhidapi-libusb0
- then, need to implement c program in whichever directory we want.

1. Vi filename.c
2. gcc -o filename filename.c -lhidapi-libusb
3. connect board (j16 & j9) to pc
4. run program in host terminal itself by giving this cmd

-> ./filename