CAVLI wireless

Re-imagining Cellular IoT Solutions

Cavli C10QM/C20QM Smart Module

Cavli SDK User Manual

External Release Version 1.7

www.cavliwireless.com

Connect to our website and feel free to contact our technical support team for any assistance.

Cavli Inc.,

99 South Almaden Blvd., Suite 600, San Jose, California, 95113

Phone: 1-650-535-1150

Web: www.cavliwireless.com

IoT Connectivity Platform: www.cavlihubble.io

Support Centre

https://www.cavliwireless.com/support-center.html

e-Mail: support@cavliwireless.com

For sales enquiries

https://www.cavliwireless.com/contact-us.html

e-Mail: sales@cavliwireless.com

More IoT Modules

https://www.cavliwireless.com/iot-modules/cellular-modules.html

Table of Contents

Table of Figures

VERSION HISTORY

Version	Edit	Date
1.0	• Initial Version	27/06/2023
1.1	• Changed the order of Toolchain creation & Image building • Changed source setting command under Toolchain • Added note under Toolchain and re-arranged the command flow	28/06/2023
1.2	• Added dependencies for Native Linux System	29/06/2023
1.3	• New Chapter is added (Environment Setup) • Overall Execution flow updated	07/07/2023
1.4	• Added the steps for kernel custom config	12/07/2023
1.5	• Added firmware flashing commands • Removed kernel configurations (menuconfig) • Updated the examples • Steps to compile application programs	12/10/2023
1.6	• Updated the steps for downloading docker image	18/12/2023
1.7	• Added two new commands in Environment setup • Added one more dependency “boxes”	22/12/2023

1 Introduction

1.1 Overview

Cavli has made the SDK for our C10QM/C20QM products open for customers. Using our SDK, a customer is able to write their program onto our device, configure the interfaces to support their conditions, setting the device into Master-Slave mode among others. This document is meant to familiarize the customer with the SDK installation into the device from GitHub and its subsequent application with C10QM/C20QM.

1.2 References

The present document is based on the following document:
C10QM/C20QM OpenSDK Architecture

NOTE

- CAN via SPI supported
- Master mode supported
- Please contact support team for any queries.

2 System Requirements

2.1 Ubuntu System

Ubuntu System	8GB RAM or more (16GB Ideal)
4 Core CPU or more with Ubuntu 16.04	

Install Ubuntu from **https://ubuntu.com/download/desktop**

Install docker for working on SDK.

We are using docker inside a Linux machine. the memory allocated for it should not be less than 8GB

NOTE

- Ensure that all the assets mentioned in Chapter 3 are installed
- User must be in ***sudo su*** mode before setting configurations/compiling the files

3 Assets Needed

- apt-get install vim git build-essential diffstat texi2html texinfo subversion gawk chrpath wget

  To download needed support libraries.

NOTE

- The sections given in color are the commands to get the needed assets. Inputting them to the terminal will automatically download the intended asset

4 Steps to Initialize the SDK Environment

4.1 Docker Installation

1. Use the below link for proper docker installation.
   https://docs.docker.com/engine/install/ubuntu/
2. sudo apt-get update
   To download necessary updates for the Docker files.
3. sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
   To install Docker.
4. service docker start
   To start the Docker.
5. systemctl docker status
   To check the Docker, whether it is successfully running.

NOTE

- The sections given in color are the commands to get the needed assets. Inputting them to the terminal will automatically download the intended asset

4.2 Cavli Docker image download

1. Download the ubuntu image from:
   https://drive.google.com/file/d/1PS6EKtRsF5ZK3Ii1SHAfT_DG2_pKYGso/view

2. docker load --input cavli_sdk.tar
   To load the docker image. (Run this command after adding the downloaded tar file under user directory)

3. docker images

   To check if the image is loaded successfully (you should be able to see the image with an ID)

4. docker run -d --hostname 9607 --name cavli_9607_sdk -it -p 2227:22 -p 8893:80 -v /mnt/d/:/share/ -w /home/cavli/ cavli_sdk:1.0 /bin/bash

   To run the docker as container.

5. docker container ps -a

   List the containers

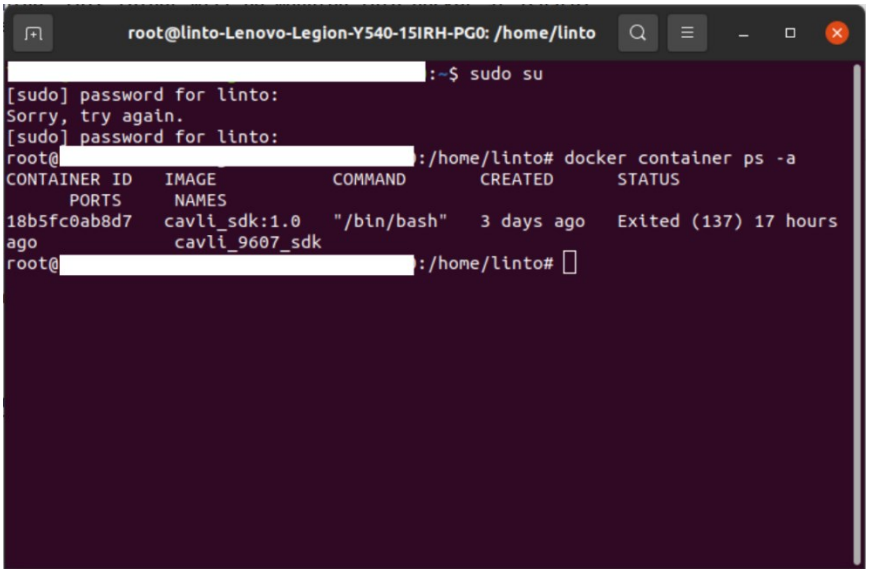

Figure 1: Listing containerID

6. docker start containerID (cavli_9607_sdk)

   docker attach containerID

   To start the containerID. (Now you are in the docker)

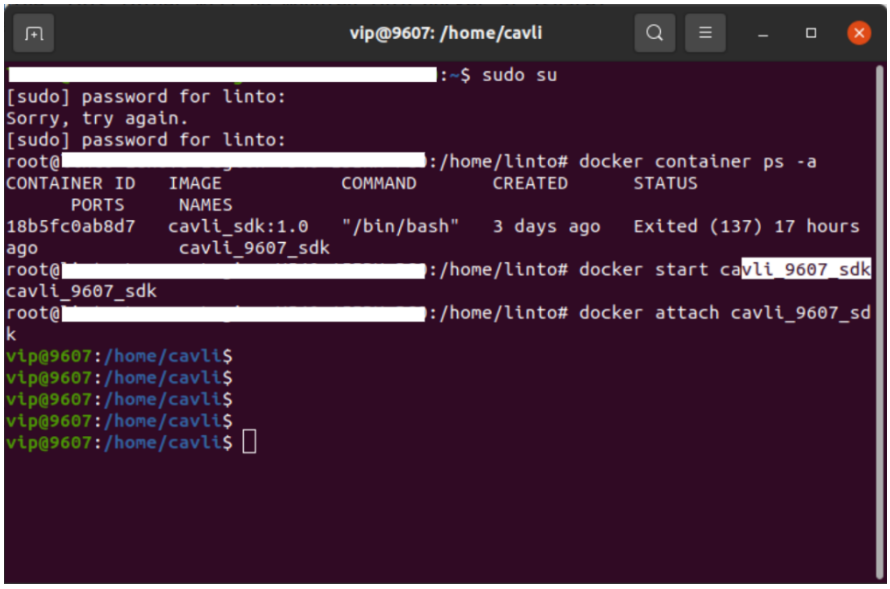

Figure 2: Starting the containerID

7. sudo apt-get update
   sudo apt-get install wget unzip zip boxes
   To install wget and zip, unzip.

Install the lfs before cloning the source. Following this guideline

Make sure you follow these steps diligently.

8. sudo apt-get install software-properties-common
9. sudo curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
10. sudo apt-get install git-lfs
11. git lfs install
    (This is required to download the ARM clang library dependency while cloning the SDK).

NOTE

- /mnt/d/ is the directory on your host machine, this folder will be mounted into docker at /share/. Change /mnt/d/ to your need. The purpose is file sharing between docker and the host.

4.3 With Native Linux System

Currently supported Linux version 16.04:

1. sudo apt-get install wget unzip zip automake gcc curl
   To install wget and zip,unzip
2. $ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
   build-essential chrpath socat cpio python python3 python3-pip python3-pexpect
   Packages need to build an image with a headless system.

4.4 Key Generation

1. Paste the ssh-keygen given: ssh-keygen -t ecdsa -b 521
   Press Enter 3 times (i.e., Passphrase etc are entered null)

(Make sure you are in root mode for gitclone and downloads, password :vip)

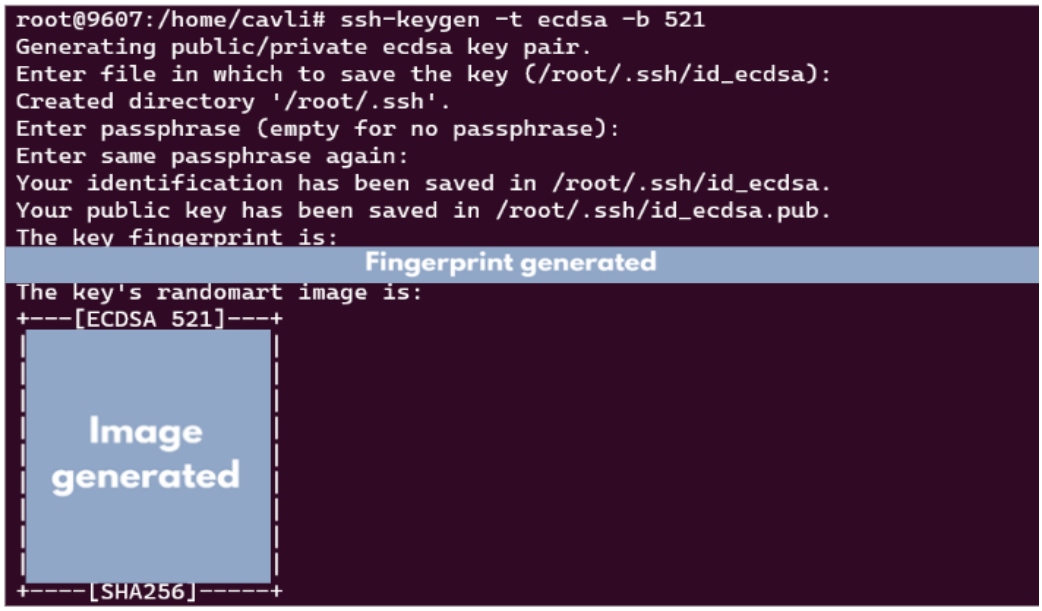

Figure 3: Fingerprint Generation of the device

2. Paste ***cat ~/.ssh/id_*.pub***

   Press *Enter* once

   This will return the SSH ID of your device

Figure 4: SSH Key Generated

3. Pass the SSH ID hence generated to the Cavli Service Desk
4. The Service Desk will return a URL to download the git. It must be used as ***git clone URL.*** Wait for the download to complete.

NOTE

- Steps 1-4 once done for a user in a device subsystem it need not be repeated for that user and subsystem
  As in; if it has Ubuntu and Windows, you will need to do for both separately as they are different subsystems
- The sections which are given in *italics*, **boldened** and ***blue color*** as ***Section*** are to be entered into the terminal as is. They are command lines used to execute a said function
- The sections which are given in ***orange color*** as Section stand for folder/file names

5 Environment setup

You can access the files for the environment setup on Google Drive . Below are the commands that should be used for each section:

1. Download the download.zip from:
   https://drive.google.com/file/d/1pSeOTfsKSpMucJHbYEPUthZwPUTEiNKR/view
   The zip file contain these packages (glib, GCC, bluez, dbus....etc)
2. Copy the downloads.zip into C10QM-CS-SDK/apps_proc/poky/build and unzip it. cp downloads.zip C10QM-CS-SDK/apps_proc/poky/build
   unzip download.zip

   OR

   sudo docker cp downloads.zip :/home/cavli/C10QM-CS SDK/apps_proc/poky/build

Change the owner mode. Following this command (here assumption user is builder, change it if needed) sudo chown -R builder:root

Example: s**udo chown -R vip:root /home/cavli/C10QM-CS-SDK**

Follow the below commands after downloading the file

1. cd C10QM-CS-SDK/apps_proc/poky

   To change directory to C10QM-CS-SDK

2. export PRODUCT_SUBTYPE=c10qm *//Supported product: c10qm - v0 , c10qm - v1, c20qm - v1*

   export CAVLI_HW_VERSION="v1"

   To Select Whether it is C10QM or C20QM (Simply put it helps to execute the ".sh" file)

3. source ./build/conf/set_bb_env.sh

   Sets the source environment for build.

4. build-9607-image

   build command for environment build.
   (The build process may take half an hour or more depends on your system).

6 Steps to create SDK Toolchain

This need to run if you have not created a toolchain. Please do an environment build before toolchain build.

1. cd C10QM-CS-SDK/apps_proc/poky

   To change directory.

2. source ./build/conf/set_bb_env.sh

   To set the source environment.

3. bitbake -c populate_sdk machine-image

   To build the toolchain:d

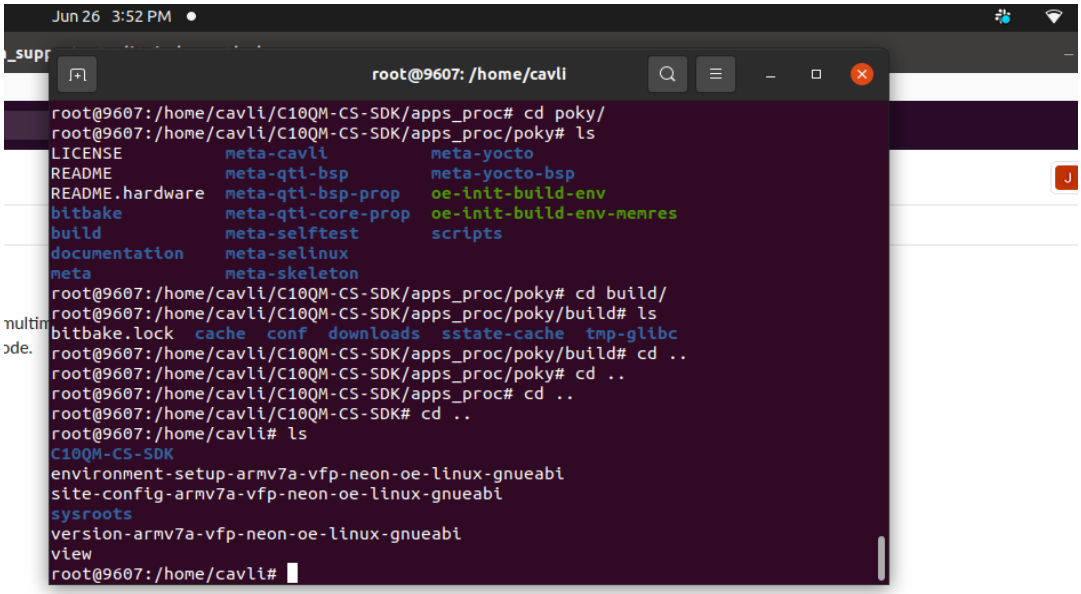

Figure 5: Cross toolchain build and copy

4. cd C10QM-CS-SDK/apps_proc/poky/build/tmp-glibc/deploy/sdk

   To change directory.

5. ./oecore-x86_64-armv7a-vfp-neon-toolchain-nodistro.0.sh

   Exports the SDK Toolchain.

   Please input the location where you want to start the SDK Toolchain

   (for example: /home/cavli)

6. source /home/cavli/environment-setup-armv7a-vfp-neon-oe-linux-gnueabi

   To source the toolchain.

7. Echo $PATH

To display the source path

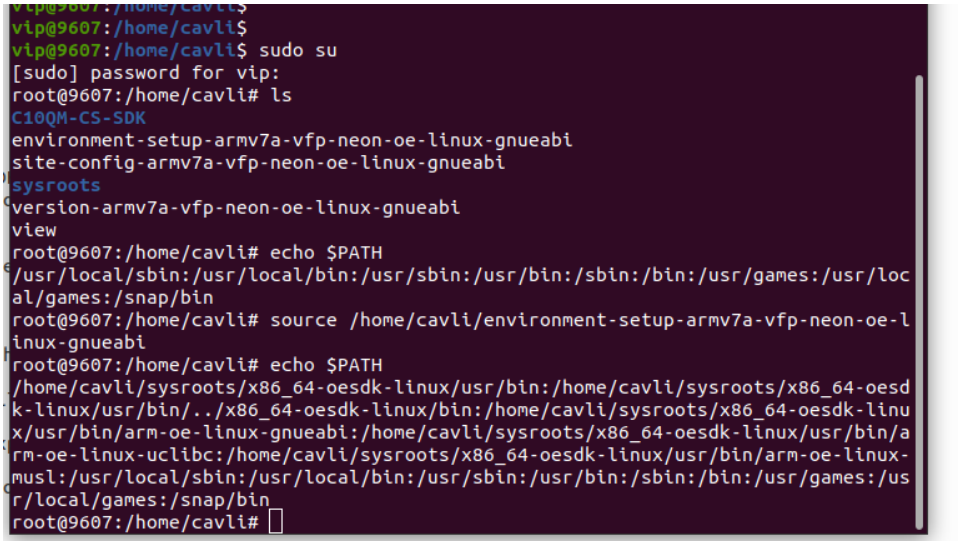

Figure 6 Source Path

NOTE

- Each time you want to use the SDK Toolchain, please run: **Step-6.**
- Building toolchain **without setting source** will leads to bricking of module.

7 Steps to Build Image for target device (C10QM)

1. cd C10QM-CS-SDK/apps_proc/poky

   To change directory to C10QM-CS-SDK
2. source ./build/conf/set_bb_env.sh

   Sets the source environment. (The build process may take half an hour or more depends on your system)
3. source /home/cavli/environment-setup-armv7a-vfp-neon-oe-linux-gnueabi

   To source the toolchain
4. build-9607-image

   To compile system image

7.1 To Compile little kernel

1. bitbake -c cleansstate lk

   To clean the build
2. bitbake lk

   To build the little kernel

7.2 To Compile linux kernel

1. bitbake -c cleansstate virtual/kernel

   To clean the build
2. bitbake virtual/kernel

   To build the little kernel

The images to be flashed are located at:

C10QM_SDK/apps_proc/poky/build/tmp-glibc/deploy/images/mdm9607

Cavli Example Projects:

There are 5 example projects in C10QM_SDK/apps_proc/cavli/examples

cpp

cavil_api

serial

mqtt

tcp

And their recipes in:

C10QM_SDK/apps_proc/poky/meta-cavli/recipes-cavli.

Figure 7 Shell view of the module

NOTE

- To improve Yocto build time as well as some issues occur when Yocto downloads packages, we prepare the data for Yocto build.

8 Compiling Application Programmes

There are 5 Application programs (examples) which can be compiled using bitbake or toolchain.

8.1 Compilation using bitbake

1. source ./build/conf/set_bb_env.sh

   To source the environment
2. bitbake <example>

   To compile the recipe using bitbake
3. After compilation the binary file of the compiled application will be in this location

   tmp-glibc/work/mdm9607-oe-linux-gnueabi/cavlicpp/1.0.0-r0/image/usr/bin$

```
vip@lintodock:/home/cavli/C10QM-CS-SDK/apps_proc/poky/build/tmp-glibc/work/mdm9607-oe-linux-gnueabi/cavlicpp/1.0.0-r0/im
age/usr/bin$ ls
cpp_example
```

Figure 8: Using Bitbake

8.2 Compilation using toolchain

1. source /home/cavli/environment-setup-armv7a-vfp-neon-oe-linux-gnueabi

   To source toolchain
2. Go to examples in C10QM-CS-SDK/apps_proc/cavli/examples
3. Compile the required application using make command

```
vip@lintodock:/home/cavli/C10QM-CS-SDK/apps_proc/cavli/examples/cpp$ ls
Makefile  atomic_ex.cpp  hello.c  inc  main.cpp
vip@lintodock:/home/cavli/C10QM-CS-SDK/apps_proc/cavli/examples/cpp$ make
arm-oe-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=softfp -mfpu=neon --sysroot=/home/cavli/sysroots/armv7a-vfp-neon-oe
-linux-gnueabi -g -O0 -Iinc -c hello.c -o hello.o
arm-oe-linux-gnueabi-g++  -march=armv7-a -mfloat-abi=softfp -mfpu=neon --sysroot=/home/cavli/sysroots/armv7a-vfp-neon-oe
-linux-gnueabi -Iinc -c main.cpp -o main.o
arm-oe-linux-gnueabi-g++  -march=armv7-a -mfloat-abi=softfp -mfpu=neon --sysroot=/home/cavli/sysroots/armv7a-vfp-neon-oe
-linux-gnueabi -g -O0 -Iinc -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed -Wl,-z,relro,-z,now,-z,noexecstack -o cpp_examp
le hello.o main.o -latomic -lstdc++ -Iinc
vip@lintodock:/home/cavli/C10QM-CS-SDK/apps_proc/cavli/examples/cpp$ ls
Makefile  atomic_ex.cpp  cpp_example  hello.c  hello.o  inc  main.cpp  main.o
```

Figure 9 Using Toolchain

8.3 To push the binary to the modem

After compiling the application programs binary files generated need to be pushed into the module using the command: adb.exe push <binary path> <destination path>
Change the permissions using the command chmod +x binary name then the customer can execute the application inside the modem.

9 Steps for Flashing the images

To Flash the images platform tools (adb and fastboot) need to be installed.
The link to the Android webpage is attached here. You may download the platform-tools from the website.

For linux operating system, use the command

```
sudo apt install android-tools-adb android-tools-fastboot-y
```

For windows the drivers needed can be downloaded from here.

Run the following commands to flash the images

1. adb devices

   To check if the modem interface has been detected by the platform tools.

2. adb reboot bootloader

   Reboot the bootloader to enter into booting state.

3. fastboot devices

   To check the fastbooting of devices

- For 256MB variant use these commands

  1. ***.\fastboot.exe flash aboot*** **"<enter_file_path>/appsboot.mbn"**

     To flash the little kernel

  2. ***.\fastboot.exe flash boot "<enter_file_path>\mdm9607-boot-2K.img"***

     To flash the linux kernel

  3. ***.\fastboot.exe flash system "<enter_file_path>\mdm9607-sysfs-2K.ubi"***

     To flash the linux system

- For 512MB variant use these commands

  1. ***.\fastboot.exe flash aboot*** **"<enter_file_path>/appsboot.mbn"**

     To flash the little kernel

  2. ***.\fastboot.exe flash boot "<enter_file_path>\mdm9607-boot-.img"***

     To flash the linux kernel

  3. ***.\fastboot.exe flash system "<enter_file_path>\mdm9607-sysfs-.ubi"***

     To flash the linux system