SB52 Development Platform for Yocto Linux

User Guide

June. 2020

Version 0.1

Document Revisions

Date	Revision Number	Document Changes
06/2020	0.1	Draft version

Table of Contents

1 Overview

This tutorial guides new developers how to build Yocto Linux with the MTK i500 based development platform – SB52 board. It provides manuals for:

- Setting up a Linux® OS build machine.
- Building SB52 Yocto Linux images.
- Flashing the images to SB52 development board.

2 Setting up your computer

To build the Yocto Linux source files, you need a 64-bit version of Ubuntu (18.04 is recommended. But compatible with 16.04).

After installing the computer running Linux OS, check whether all the necessary packages are installed.

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-
multilib build-essential chrpath socat cpio python python3 python3-pip
python3-pexpect xz-utils debianutils iputils-ping python3-git python3-
jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm g++ libstdc++6
lib32stdc++6 libpulse-dev libevent-dev ninja-build rpm2cpio libswitch-
perl
```

gn tools install

```
$ sudo apt-get install libssl-dev
$ wget -O gn http://storage.googleapis.com/chromium-
gn/3fd43e5e0dcc674f0a0c004ec290d04bb2e1c60e
$ sudo chmod 777 gn
```

Put the **gn** in build server /usr/bin/

Install adb and Fastboot

```
$ sudo apt-get install android-tools-adb android-tools-fastboot
```

Note: If your Ubuntu use default dash shell, please install bash shell to build image.

3 Building the Yocto Linux for SB52

3.1 Downloading the SB52 source

Please contact your Innocomm contact window to download the SB52 source code.

3.2 Extracting the SB52 source

After you have set up a computer running Linux OS, extract the SB52 source tar by using the following commands:

```
$ cd ~ (or any other directory you like)
$ unzip sb52-yocto.33.tar.xz.zip (need password)
$ tar -xJvf sb52-yocto.33.tar.xz
```

3.3 Building Yocto Linux images

- **Full build**

```
$ export TEMPLATECONF=${PWD}/meta/meta-innocomm/conf/base/sb52

$ source meta/poky/oe-init-build-env
$ bitbake innocomm-image-openmm-aiv
```

Note:

If you want a clean build, remove **build** and **sstate-cache** folder, and start from beginning.

- **Partial build**

Partial compile can generate "boot","lk" image separately, also can verity build fail quickly.But can't generate "rootfs""userdata" images.

```
#Kernel
$ bitbake virtual/kernel -c cleansstate    //clean
$ bitbake virtual/kernel -f                //rebuild

#LK
$ bitbake lk -c cleansstate
$ bitbake lk -f

#Appmainprog and other target bb
$ bitbake appmainprog -c cleansstate
$ bitbake appmainprog
```

The images will be located in the below folder, if build is successful.

```
build/tmp/deploy/images/sb52
```

Note: All files under this folder are needed to flash the board

3.4 Commercial License

Although Yocto Linux is open source, some licenses of modules/packages are commercial and need to buy the license for your product.

Take AAC decoder plugin of gstreamer, gstfaad, as example, following modification should be done to include it in the image.

1) Add **gstreamer1.0-plugins-bad_%.bbappend**

```
$ mkdir -p meta/meta-innocomm/recipes-multimedia/gstreamer
$ touch meta/meta-innocomm/recipes-multimedia/gstreamer/gstreamer1.0-plugins-bad_%.bbappend
$ echo "PACKAGECONFIG += \" faad \\\"" > meta/meta-innocomm/recipes-multimedia/gstreamer/gstreamer1.0-plugins-bad_%.bbappend
```

2) Fix wayland depend

In **src/multimedia/gst-mtkwaylandsink/src/wayland.c**

Change

```
#include <gst/wayland/wanland.h>
```

To

```
#include <wayland.h>
```

3) Add whitelist for fdd2 (**please be NOTICED, the modification is for test only, if you want to enable modules/packages for your product finally, please buy licenses**)

In meta/meta-innocomm/conf/base/sb52/local.conf.sample, add following text.

```
LICENSE_FLAGS_WHITELIST = "commercial_faad2"
```

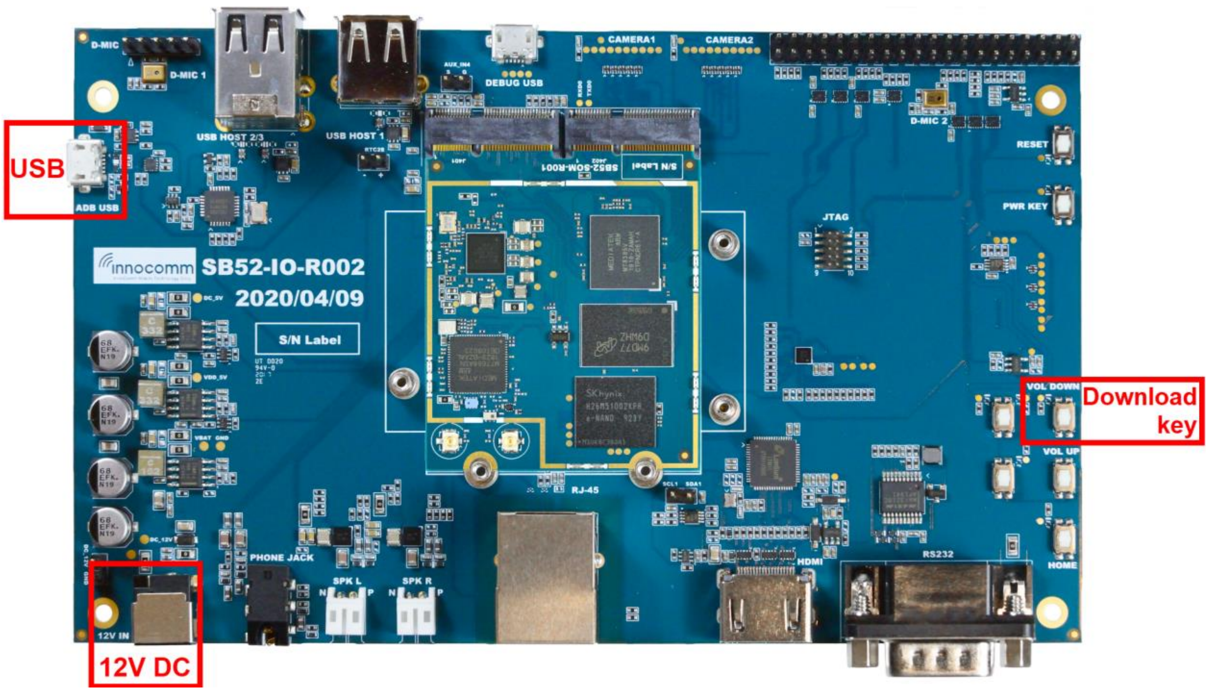
For more information about commercial license recipes, please refer to https://www.yoctoproject.org/docs/current/mega-manual/mega-manual.html#enabling-commercially-licensed-recipes

4 Flashing SB52 image

Use the command below to flash the board

```
$ sudo python flashimage.py
```

- Full flash steps:



1. Plug out SB52 power cable

2. Plug in USB cable

3. Run flashimage.py

```
$ sudo python flashimage.py
usage: flashimage.py [-h] [-d] [-s] [-u] [-b] [-t] [-v] [-n] [--toolsdir TOOLSDIR]
```

```
                     [--productdir PRODUCTDIR]
                     [partition]

********************************************************************************
   Running flasher on Linux-4.15.0-72-generic-x86_64-with-Ubuntu-18.04-bionic
********************************************************************************

********************************************************************************
* flash images under:                                                          *
*     /home/lin/sb52                                     *
********************************************************************************

                                 Checking image
--------------------------------------------------------------------------------
                                   MBR : PASS
                               bl2.img : PASS
                               tee.img : PASS
                              boot.img : PASS
                           system.ext4 : PASS
                         cam_vpu_a.img : PASS
                         cam_vpu_b.img : PASS
                         cam_vpu_c.img : PASS
                          sspm-fit.img : PASS
                             spmfw.img : PASS
                         userdata.ext4 : PASS

                                 Start flashing
--------------------------------------------------------------------------------
Waiting for DA mode
```

4. Press and hold **download key** (to enter download mode)

5. Plug in SB52 power cable. Image flash will start...

```
$ sudo python flashimage.py
[sudo] password for lin:

usage: flashimage.py [-h] [-d] [-s] [-u] [-b] [-t] [-v] [-n] [--toolsdir TOOLSDIR]
                     [--productdir PRODUCTDIR]
                     [partition]

********************************************************************************
   Running flasher on Linux-4.15.0-72-generic-x86_64-with-Ubuntu-18.04-bionic
```

```
********************************************************************************

********************************************************************************
* flash images under:                                                          *
*     /home/lin/sb52                          *
********************************************************************************

                                 Checking image
--------------------------------------------------------------------------------
                                   MBR : PASS
                               bl2.img : PASS
                               tee.img : PASS
                              boot.img : PASS
                           system.ext4 : PASS
                         cam_vpu_a.img : PASS
                         cam_vpu_b.img : PASS
                         cam_vpu_c.img : PASS
                          sspm-fit.img : PASS
                             spmfw.img : PASS
                         userdata.ext4 : PASS

                                 Start flashing
--------------------------------------------------------------------------------
Waiting for DA mode
.
datool - device detected:

Waiting for fastboot mode
.
Fastboot - device detected:  EM7LVGI7SORO4995

erasing 'mmc0'...
(bootloader) request sz: 0x3ab400000, real erase len: 0x3ab400000
OKAY [  0.156s]
finished. total time: 0.156s
target reported max download size of 67108864 bytes
sending 'mmc0' (17 KB)...
OKAY [  0.007s]
writing 'mmc0'...
OKAY [  0.020s]
finished. total time: 0.027s
target reported max download size of 67108864 bytes
sending 'mmc0boot0' (274 KB)...
```

```
OKAY [  0.021s]
writing 'mmc0boot0'...
OKAY [  0.013s]
finished. total time: 0.034s
target reported max download size of 67108864 bytes
sending 'tee_a' (989 KB)...
OKAY [  0.047s]
writing 'tee_a'...
OKAY [  0.022s]
finished. total time: 0.069s
target reported max download size of 67108864 bytes
sending 'boot_a' (25376 KB)...
OKAY [  1.163s]
writing 'boot_a'...
OKAY [  0.552s]
finished. total time: 1.715s
target reported max download size of 67108864 bytes
Invalid sparse file format at header magic
sending sparse 'system_a' 1/9 (65532 KB)...
OKAY [  2.946s]
writing 'system_a' 1/9...
OKAY [  1.479s]
sending sparse 'system_a' 2/9 (64896 KB)...
OKAY [  2.925s]
writing 'system_a' 2/9...
OKAY [  1.354s]
sending sparse 'system_a' 3/9 (59772 KB)...
OKAY [  2.710s]
writing 'system_a' 3/9...
OKAY [  1.326s]
sending sparse 'system_a' 4/9 (63164 KB)...
OKAY [  2.812s]
writing 'system_a' 4/9...
OKAY [  1.287s]
sending sparse 'system_a' 5/9 (58976 KB)...
OKAY [  2.673s]
writing 'system_a' 5/9...
OKAY [  1.195s]
sending sparse 'system_a' 6/9 (65532 KB)...
OKAY [  2.930s]
writing 'system_a' 6/9...
OKAY [  1.337s]
sending sparse 'system_a' 7/9 (65532 KB)...
```

```
OKAY [  2.967s]
writing 'system_a' 7/9...
OKAY [  1.334s]
sending sparse 'system_a' 8/9 (65532 KB)...
OKAY [  2.930s]
writing 'system_a' 8/9...
OKAY [  1.329s]
sending sparse 'system_a' 9/9 (18476 KB)...
OKAY [  0.816s]
writing 'system_a' 9/9...
OKAY [  0.424s]
finished. total time: 34.775s
target reported max download size of 67108864 bytes
sending 'cam_vpu_a' (1614 KB)...
OKAY [  0.145s]
writing 'cam_vpu_a'...
OKAY [  0.035s]
finished. total time: 0.179s
target reported max download size of 67108864 bytes
sending 'cam_vpu_b' (9889 KB)...
OKAY [  0.747s]
writing 'cam_vpu_b'...
OKAY [  0.203s]
finished. total time: 0.950s
target reported max download size of 67108864 bytes
sending 'cam_vpu_c' (135 KB)...
OKAY [  0.030s]
writing 'cam_vpu_c'...
OKAY [  0.006s]
finished. total time: 0.037s
target reported max download size of 67108864 bytes
sending 'sspm_a' (486 KB)...
OKAY [  0.060s]
writing 'sspm_a'...
OKAY [  0.013s]
finished. total time: 0.072s
target reported max download size of 67108864 bytes
sending 'spmfw' (38 KB)...
OKAY [  0.041s]
writing 'spmfw'...
OKAY [  0.004s]
finished. total time: 0.045s
target reported max download size of 67108864 bytes
```

```
sending 'userdata' (22528 KB)...
OKAY [  1.532s]
writing 'userdata'...
OKAY [  0.470s]
finished. total time: 2.002s
rebooting...

finished. total time: 0.052s

Success
```

- Upgrade each partition separately

Partial download steps:

1. SB52 power off
2. Connect USB cable
3. Enter waiting for upgrade mode

```
$ cd build/tmp/deploy/images/sb52
$ sudo python fbtool.py -f dl_addr.ini
INFO: pySerial version: (3.0.1)
INFO: Use config file: dl_addr.ini
INFO: Waiting to connect platform...
```

4. Press and hold volume up key, then Power on SB52.

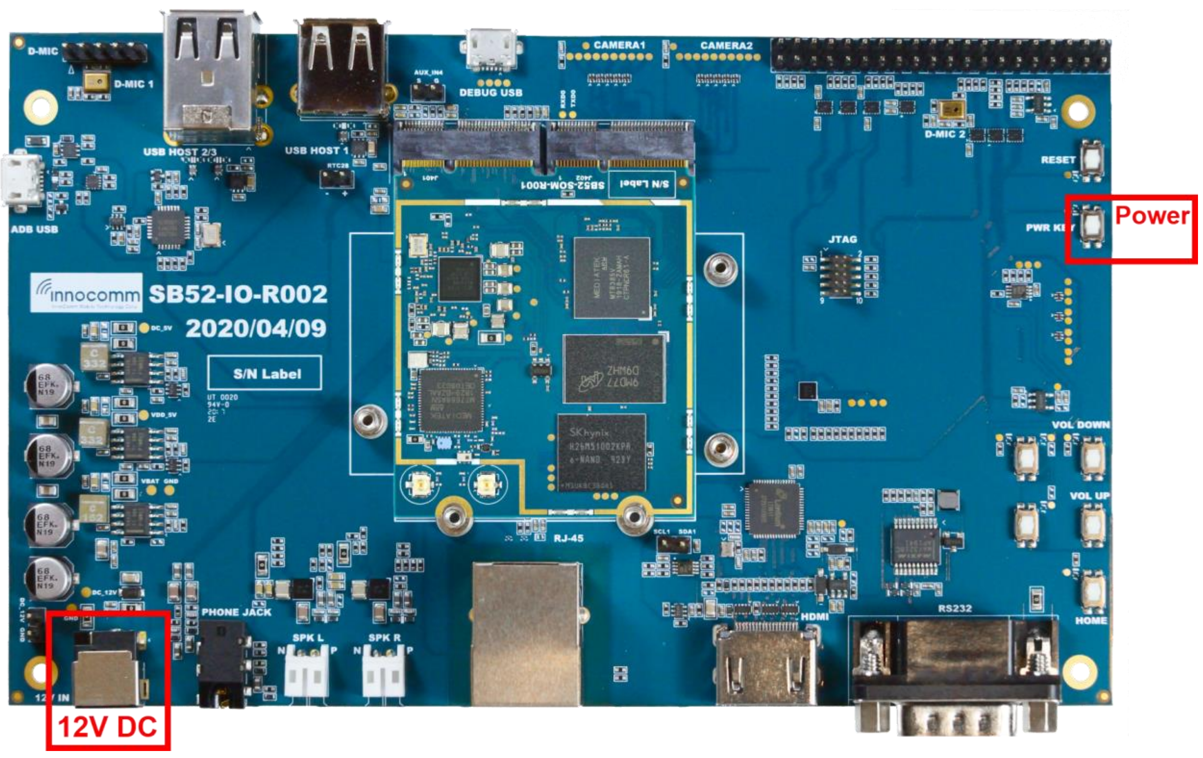
```
$ cd build/tmp/deploy/images/sb52
$ sudo python fbtool.py -f dl_addr.ini
INFO: pySerial version: (3.0.1)
INFO: Use config file: dl_addr.ini
INFO: Waiting to connect platform...
INFO: Got /dev/ttyACM0
INFO: Connect brom
INFO: Loading file: lk.bin
INFO: Send lk.bin
INFO: Jump da
$
```

Jump back to the command line. The fastboot mode is ready. You can continue to enter the next command to flash the specific partition.

```
$ sudo ./fastboot-linux-x86_64 devices
$ sudo ./fastboot-linux-x86_64 flash mmc0boot0 bl2.img
$ sudo ./fastboot-linux-x86_64 flash tee_a tee.img
$ sudo ./fastboot-linux-x86_64 flash boot_a boot.img
$ sudo ./fastboot-linux-x86_64 flash system_a system.ext4
$ sudo ./fastboot-linux-x86_64 flash cam_vpu_a cam_vpu_a.img
$ sudo ./fastboot-linux-x86_64 flash cam_vpu_b cam_vpu_b.img
$ sudo ./fastboot-linux-x86_64 flash cam_vpu_c cam_vpu_c.img
$ sudo ./fastboot-linux-x86_64 flash sspm_a sspm-fit.img
$ sudo ./fastboot-linux-x86_64 flash spmfw spmfw.img
$ sudo ./fastboot-linux-x86_64 flash userdata userdata.ext4
$ sudo ./fastboot-linux-x86_64 reboot
```

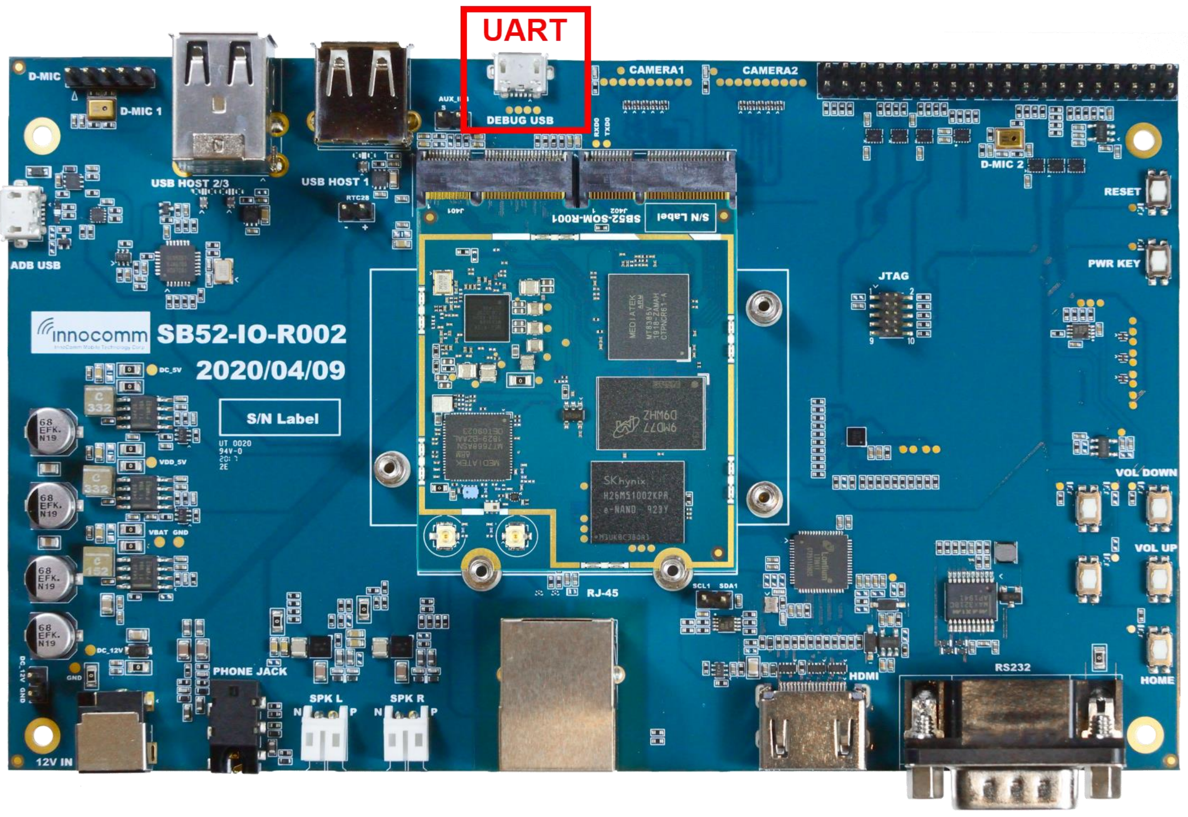
5 Booting SB52

SB52 development board is powered by 12V DC from DC Jack. Long press power key to boot SB52.



6 Serial console terminal

The serial console is a helpful tool for debugging your board and reviewing system log information. The console is the default output location for kernel log messages (i.e. dmesg), and it also provides access to a full shell prompt that you can use to access commands such as logcat.



Recommended tools for serial communication terminal:

- **Putty** for Windows.
- **Minicom** for Ubuntu. ($ sudo apt-get install minicom)

Configure the serial port as follows:

- Baud rate: 921600
- Data bits: 8
- Parity: None
- Stop bits: 1
- Hardware Flow Control : No
- Software Flow Control : No

7 Switch USB Mode

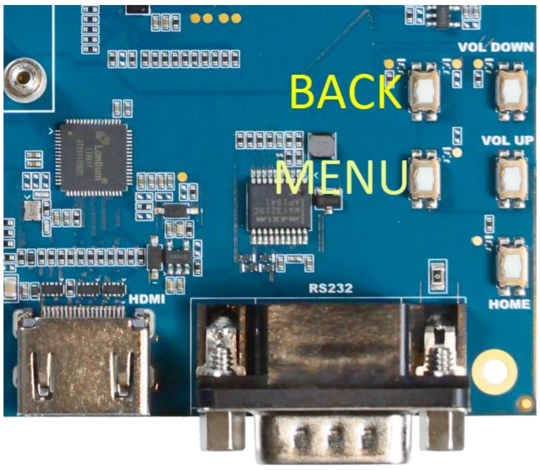
To use Ethernet, or any USB device connected to USB type-A, it requires to switch the USB mode of SB52 from device to host.

The SB52 provides function of switching the USB mode via key(BACK or MENU).

BACK: switch to USB device mode

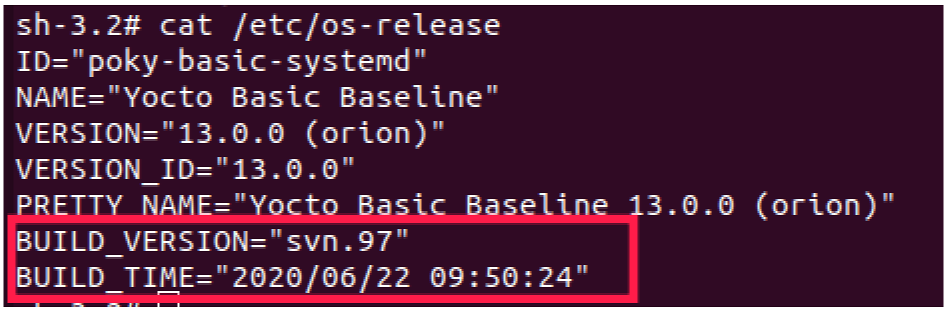
MENU: switch to USB host mode

If user press neither BACK nor MENU key when boot up the SB52, the USB mode keeps same as previous boot-on.



8 Versioning of Released FW

The versioning of FW image is used to clarify which FW released by Innocomm is flashed on the device and the information could be checked with **/etc/os-release**. Following is an example of ‘cat /etc/os-release’, the **BUILD_VERSION** and **BUILD_TIME** represent the subversion revision and build time respectively.



```
sh-3.2# cat /etc/os-release
ID="poky-basic-systemd"
NAME="Yocto Basic Baseline"
VERSION="13.0.0 (orion)"
VERSION_ID="13.0.0"
PRETTY_NAME="Yocto Basic Baseline 13.0.0 (orion)"
BUILD_VERSION="svn.97"
BUILD_TIME="2020/06/22 09:50:24"
```

9 Reference

SB52_Yocto_Linux_Verification_Guide