VS680 Software Overview

Andrew Duggan

22 August 2023

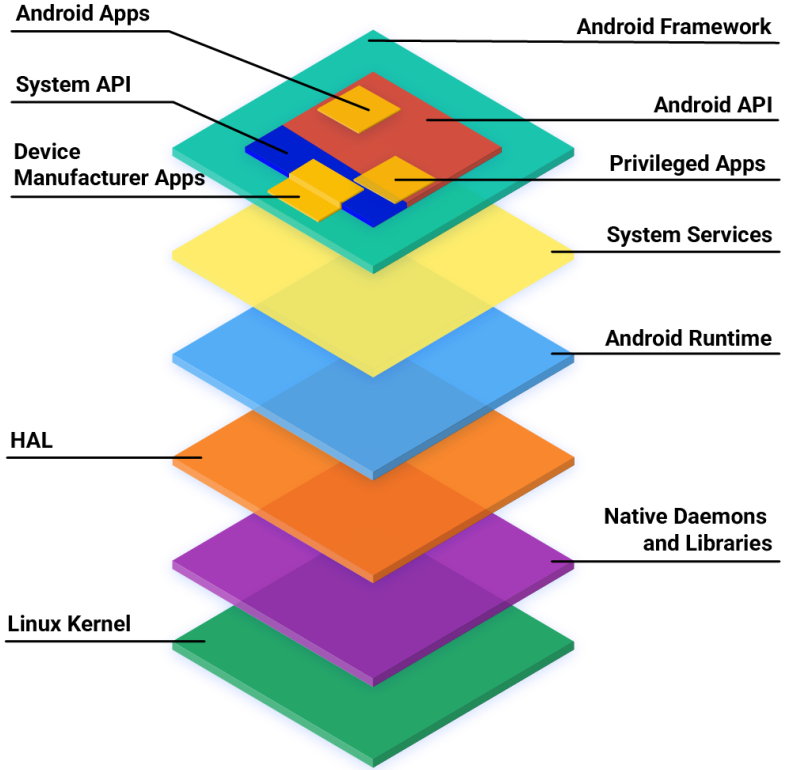
synaptics®

Overview

- VS680 runs the open source version of Android 12 (AOSP).
- We provide an SDK called VSSDK which contains the board specific components which make Android run.
    1. Kernel
    2. Bootloader
    3. Native processes for supporting media
    4. Trustzone OS and apps
    5. Android HAL implementations

© 2021 Synaptics Incorporated

Architectural Overview



VSSDK contains the code for the Linux Kernel, Native Daemons and Libraries, and HAL implementations.

© 2021 Synaptics Incorporated

Development on VS680

Android provides well defined APIs for interfacing with the Camera, NPU, and GPU.

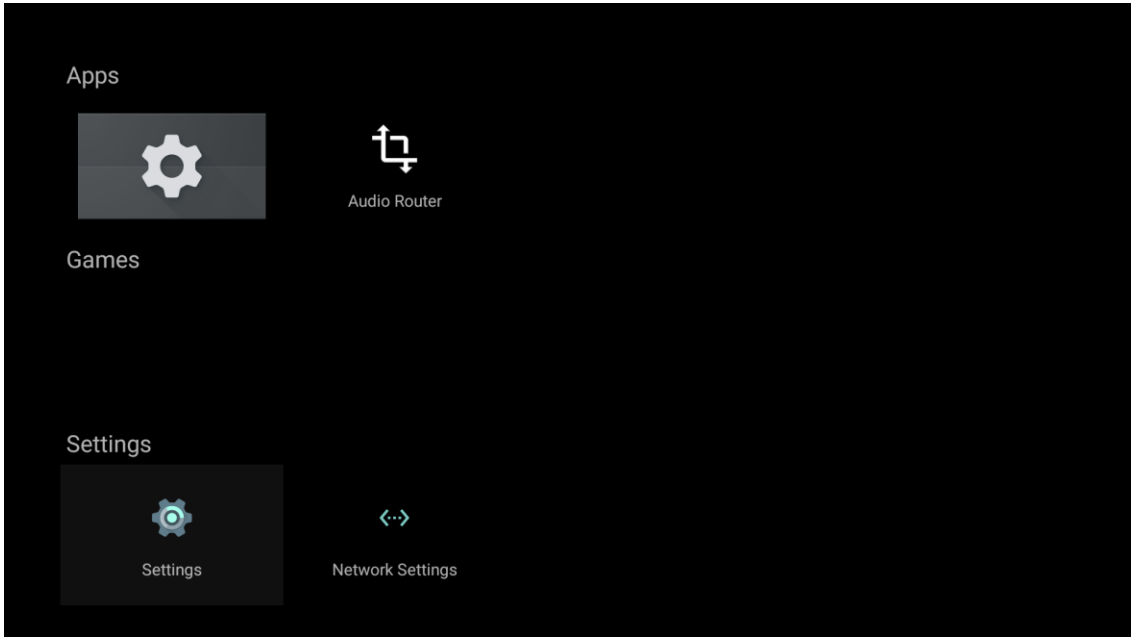
Applications can be developed using Android Studio the Android NDK can be used to develop the native components.

Example:
https://github.com/android/camera-samples

© 2021 Synaptics Incorporated

AOSP vs Android

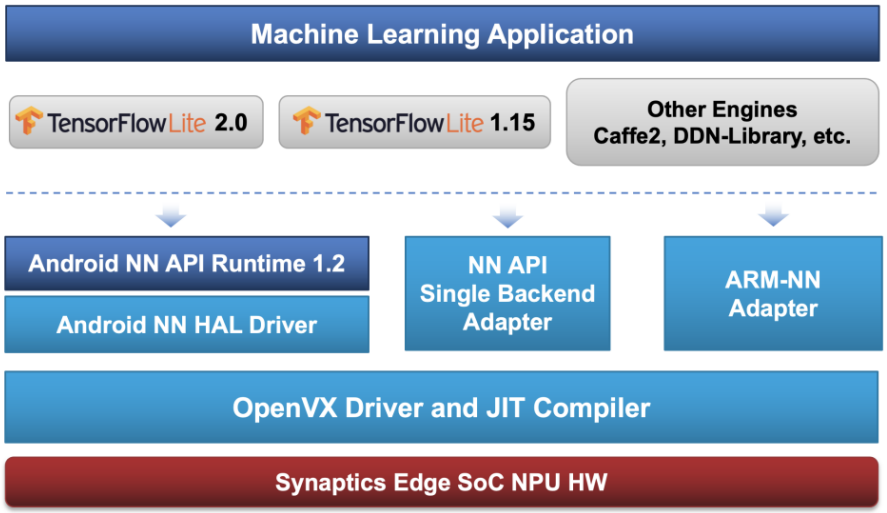
AOSP is the purely open source version of Android and it lacks Google Play Services. Developing Android apps for AOSP cannot depend on these services.



© 2021 Synaptics Incorporated

Interacting with the NPU

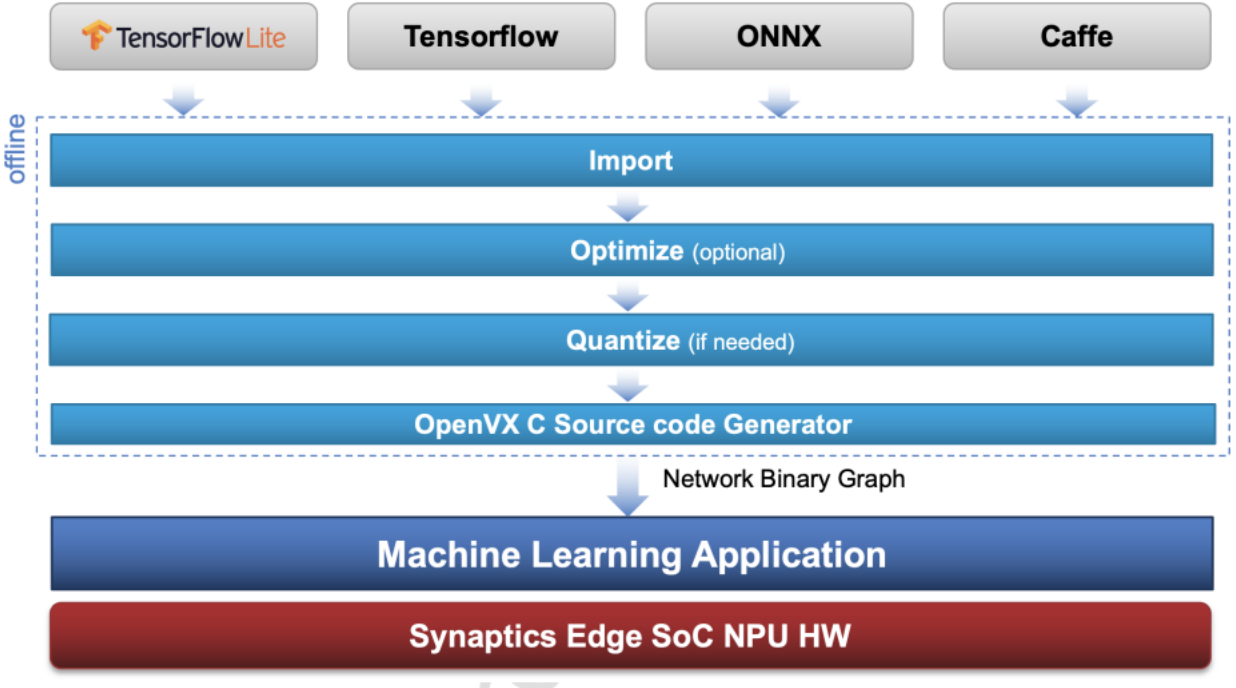
It can be accessed through the Android NNAPI interface. The NNAPI uses "online" model conversion which optimizes and converts the model at runtime into a native format understood by the NPU hardware.



© 2021 Synaptics Incorporated

Offline Models

Offline conversion is done beforehand and allows for greater optimization. The output is a native binary format compatible with the NPU hardware.



© 2021 Synaptics Incorporated

SynAP

Synaptics provides SynAP to support offline modes.

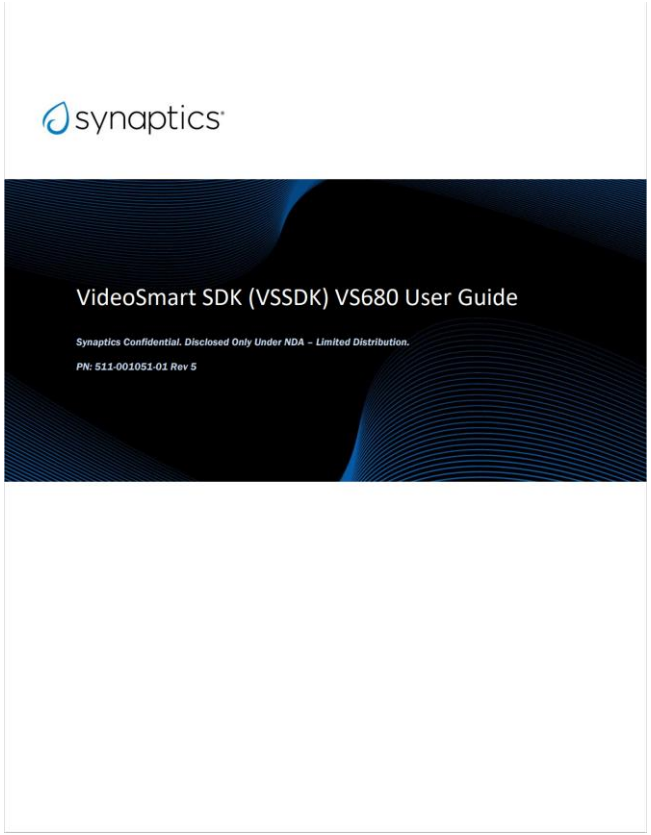
SynAP consists of a framework for running offline models and a toolkit for quantizing, optimizing, and profiling offline models.

Both are documented in the SDK in syna-release/synap/release/doc/SynAP.pdf

© 2021 Synaptics Incorporated

Working with VS680

User Guide



© 2021 Synaptics Incorporated

Accessing the SDK with Git

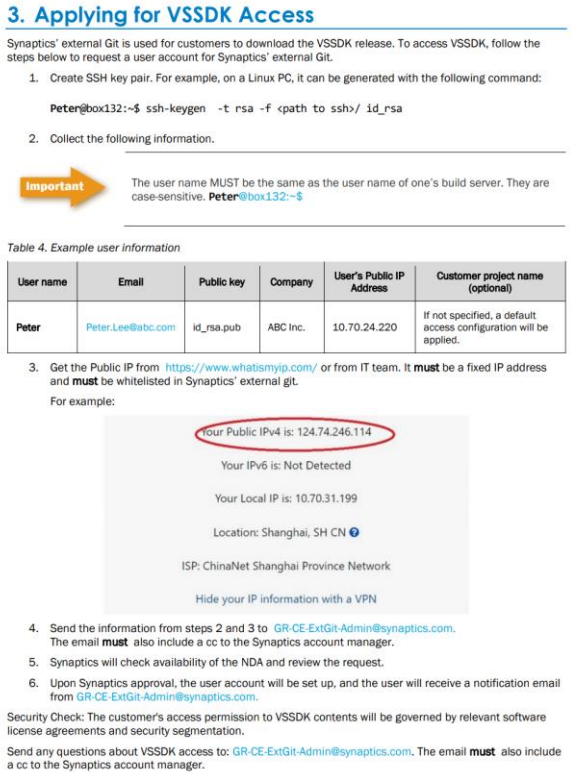
Accessing our external SDK is restricted. We will create one or two accounts which have access to it. Accounts should use an @ucf.edu email address.

Access requires:

- SSH RSA public / private key pair for the account.
- The external IP address of the server being used to download the code.

A local mirror can be setup to distribute the SDK throughout the class.

Instructions are on page 8 of the User Guide.



3. Applying for VSSDK Access

Synaptics' external Git is used for customers to download the VSSDK release. To access VSSDK, follow the steps below to request a user account for Synaptics' external Git.

1. Create SSH key pair. For example, on a Linux PC, it can be generated with the following command:

   `Peter@box132:~$ ssh-keygen -t rsa -f <path to ssh>/ id_rsa`

2. Collect the following information.

Important The user name MUST be the same as the user name of one's build server. They are case-sensitive. **Peter**@box132:~$

Table 4. Example user information

User name	Email	Public key	Company	User's Public IP Address	Customer project name (optional)
Peter	Peter.Lee@abc.com	id_rsa.pub	ABC Inc.	10.70.24.220	If not specified, a default access configuration will be applied.

3. Get the Public IP from https://www.whatismyip.com/ or from IT team. It **must** be a fixed IP address and **must** be whitelisted in Synaptics' external git.

   For example:

Your Public IPv4 is: 124.74.246.114

Your IPv6 is: Not Detected

Your Local IP is: 10.70.31.199

Location: Shanghai, SH CN

ISP: ChinaNet Shanghai Province Network

Hide your IP information with a VPN

4. Send the information from steps 2 and 3 to GR-CE-ExtGit-Admin@synaptics.com. The email **must** also include a cc to the Synaptics account manager.
5. Synaptics will check availability of the NDA and review the request.
6. Upon Synaptics approval, the user account will be set up, and the user will receive a notification email from GR-CE-ExtGit-Admin@synaptics.com.

Security Check: The customer's access permission to VSSDK contents will be governed by relevant software license agreements and security segmentation.

Send any questions about VSSDK access to: GR-CE-ExtGit-Admin@synaptics.com. The email **must** also include a cc to the Synaptics account manager.

© 2021 Synaptics Incorporated

Build Environment

A build server with 16GB RAM and 300 – 600GB storage.
We recommend Ubuntu 16.04 as the OS.

Build server can be a VM or docker container.

List of package dependencies is in the user guide.

© 2021 Synaptics Incorporated

Downloading the SDK

The SDK and Android both use repo to manage all of the Git repositories.

```
[aduggan@sjc1uvp-bld06 build]$ mkdir s_aosp_dolphin_sl_noip
[aduggan@sjc1uvp-bld06 build]$ cd s_aosp_dolphin_sl_noip/
[aduggan@sjc1uvp-bld06 s_aosp_dolphin_sl_noip]$ repo init -u ssh://debugithub.synaptics.com:29418/manifest -b m/cd/reference_auto_sync/sdk/rel_branch/vssdk/v1.10.1/202301101805/s_aosp_do
lphin_sl_noip/v/202305160610/202305171142 --repo-url=ssh://debugithub.synaptics.com:29418/common/git-repo-url --no-repo-verify
Get ssh://debugithub.synaptics.com:29418/common/git-repo-url
remote: Counting objects: 3389, done
remote: Finding sources: 100% (3389/3389)
remote: Total 3389 (delta 1845), reused 3389 (delta 1845)
Receiving objects: 100% (3389/3389), 3.16 MiB | 6.34 MiB/s, done.
Resolving deltas: 100% (1845/1845), done.
From ssh://debugithub.synaptics.com:29418/common/git-repo-url
 * [new branch]      maint      -> origin/maint
 * [new branch]      master     -> origin/master
 * [new branch]      stable     -> origin/stable
```

In this example the manifest branch name is set with the –b parameter and is:
m/cd/reference_auto_sync/sdk/rel_branch/vssdk/v1.10.1/202301101805/s_aosp_dolphin_sl_noip/v/202305160610/202305171142

This will varies based on the release being downloaded.

```
[aduggan@sjc1uvp-bld06 s_aosp_dolphin_sl_noip]$ repo sync
Fetching project DEBU/android/vendor/imagination
remote: Counting objects: 7109, done
remote: Finding sources: 100% (7109/7109)
```

© 2021 Synaptics Incorporated

SDK Directory Structure

The Android directory contains Android HAL components and will be copied into the Android
Tree.
The syna-release directory low level components for the system.

```
[aduggan@sjc1uvp-bld06 s_aosp_dolphin_sl_noip]$ ls
android  syna-release
[aduggan@sjc1uvp-bld06 s_aosp_dolphin_sl_noip]$ ls syna-release/
ampsdk  application  boot  build  configs  drm  external  factory  fw_enc  linux_5_4  Makefile  osal  security  synap  sysroot  ta_app  ta_enc  tee  toolchain
[aduggan@sjc1uvp-bld06 s_aosp_dolphin_sl_noip]$ ls android/
device  vendor
[aduggan@sjc1uvp-bld06 s_aosp_dolphin_sl_noip]$
```

© 2021 Synaptics Incorporated

Downloading Android

Downloads branch android-12.0.0_r32 of AOSP

```
[aduggan@sjc1uvp-bld06 build]$ cd aosp-s
[aduggan@sjc1uvp-bld06 aosp-s]$ repo init -u https://android.googlesource.com/platform/manifest -b android-12.0.0_r32
Get https://gerrit.googlesource.com/git-repo/clone.bundle
Get https://gerrit.googlesource.com/git-repo
Checking out files: 100% (137/137), done.
```

```
[aduggan@sjc1uvp-bld06 aosp-s]$ repo sync
remote: Counting objects: 464, done
remote: Finding sources: 100% (208/208)
remote: Total 208 (delta 62), reused 149 (delta 62)
Receiving objects: 100% (208/208), 690.91 KiB | 23.82 MiB/s, done.
Resolving deltas: 100% (62/62), completed with 9 local objects.
```

© 2021 Synaptics Incorporated

Android Directory

```
[aduggan@sjc1uvp-bld06 aosp-s]$ ls
Android.bp  bootable        BUILD          dalvik       device      hardware  libnativehelper  pdk              sdk     toolchain  WORKSPACE
art         bootstrap.bash  compatibility  developers   external    kernel    Makefile         platform_testing  system  tools
bionic      build           cts            development  frameworks  libcore   packages         prebuilts         test    vendor
```

Then copy SDK's android directory into the AOSP tree.

```
[aduggan@sjc1uvp-bld06 s_aosp_dolphin_sl_noip]$ cp -r android/* ../aosp-s
```

This integrates all of the Synaptics vendor specific files into Android.

© 2021 Synaptics Incorporated

Building the Android Image

The steps to build the SDK and an entire AOSP image are performed by the build_androidtv script.

```
[aduggan@sjc1uvp-bld06 aosp-s]$ ./vendor/synaptics/build/build_androidtv -p ./vendor/synaptics/dolphin/configs/aosp_dolphin_sl_noip -m /data/build/s_aosp_dolphin_sl_noip/syna-release
```

The script also allows only building specific steps. The command below will only build the SDK.

```
[aduggan@sjc1uvp-bld06 aosp-s]$ ./vendor/synaptics/build/build_androidtv -p ./vendor/synaptics/dolphin/configs/aosp_dolphin_sl_noip -m /data/build/s_aosp_dolphin_sl_noip/syna-release -s
prepare,sdk
```

© 2021 Synaptics Incorporated

The Android Image

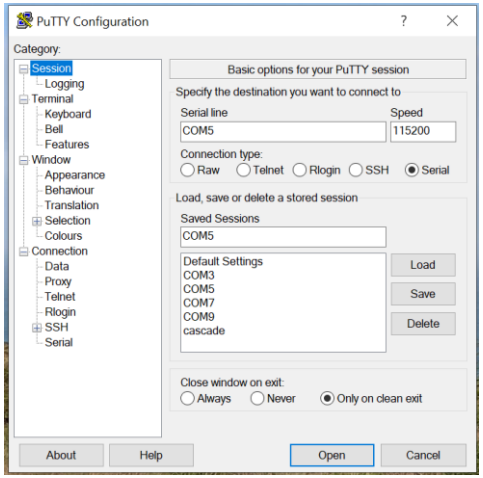
Once the build is complete the Android image will be located in ./vendor/synaptics/dolphin/build/aosp_dolphin_sl_noip/release/eMMCimg

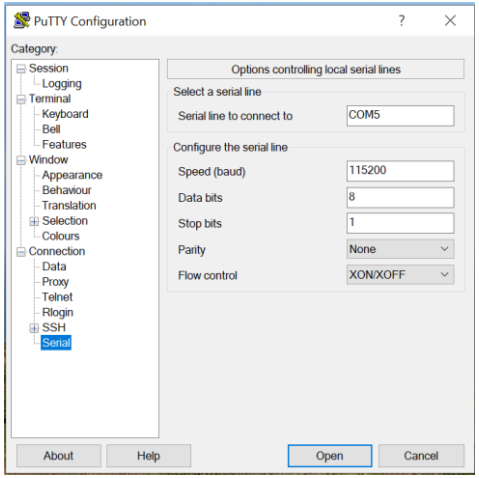
```
[aduggan@sjc1uvp-bld06 release]$ ls
eMMCimg  factory_img  over_air_img
[aduggan@sjc1uvp-bld06 release]$ cd eMMCimg/
[aduggan@sjc1uvp-bld06 eMMCimg]$ ls
dolphin-syna-image.zip  factory_setting.subimg
[aduggan@sjc1uvp-bld06 eMMCimg]$ unzip dolphin-syna-image.zip
Archive:  dolphin-syna-image.zip
   creating: eMMCimg/
  inflating: eMMCimg/bl.subimg
  inflating: eMMCimg/emmc_image_list
  inflating: eMMCimg/emmc_part_list
  inflating: eMMCimg/fastboot.subimg
  inflating: eMMCimg/fastlogo.subimg
  inflating: eMMCimg/key.subimg
  inflating: eMMCimg/preboot.subimg
  inflating: eMMCimg/tzk.subimg
  inflating: eMMCimg/boot.subimg
  inflating: eMMCimg/userdata.subimg
  inflating: eMMCimg/vendor_boot.subimg
  inflating: eMMCimg/emmc_image_list_full
  inflating: eMMCimg/vbmeta.subimg
  inflating: eMMCimg/vbmeta_system.subimg
  inflating: eMMCimg/dtbo.subimg
  inflating: eMMCimg/boot-debug.subimg
  inflating: eMMCimg/vendor_boot-debug.subimg
  inflating: eMMCimg/super.subimg.0
  inflating: eMMCimg/super.subimg.1
  inflating: eMMCimg/super.subimg.2
  inflating: eMMCimg/super.subimg.3
  inflating: eMMCimg/super.subimg.4
[aduggan@sjc1uvp-bld06 eMMCimg]$ cp factory_setting.subimg eMMCimg/
[aduggan@sjc1uvp-bld06 eMMCimg]$ ls
dolphin-syna-image.zip  eMMCimg  factory_setting.subimg
[aduggan@sjc1uvp-bld06 eMMCimg]$
```

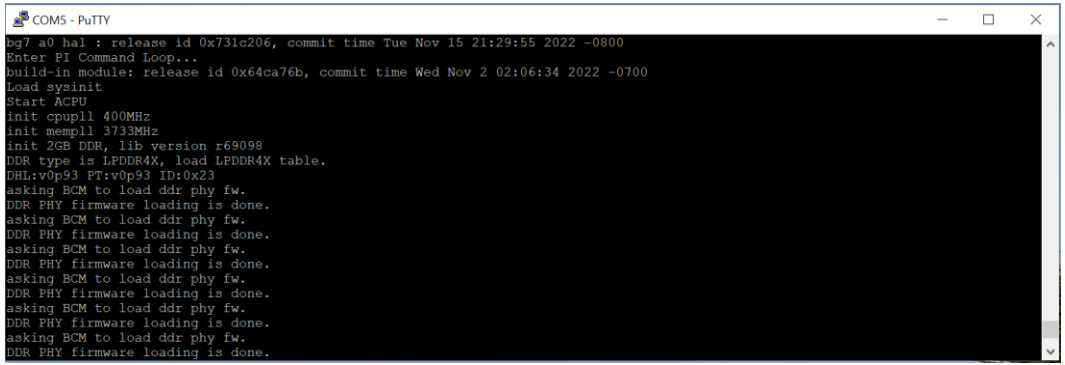
Unzip the dolphin-syna-image.zip file. The images will be extracted into a new eMMCimg directory. Then copy the factory_setting.subimg to the new eMMCimg. Now the resulting eMMCimg directory can be copied to a flash drive for installation onto the board.

© 2021 Synaptics Incorporated

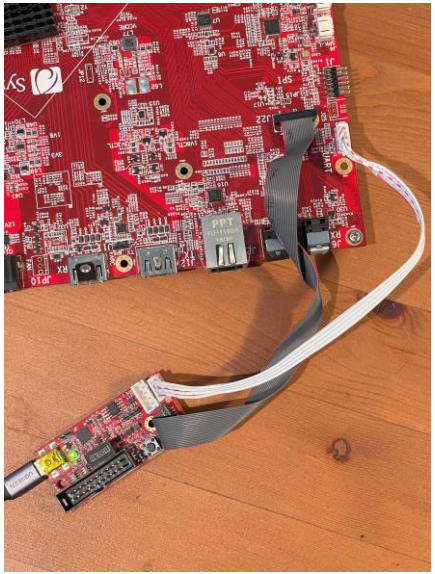
Setting up the console







Debug board connects to the board with the white cable into the UART connector. (Gray cable not provided)

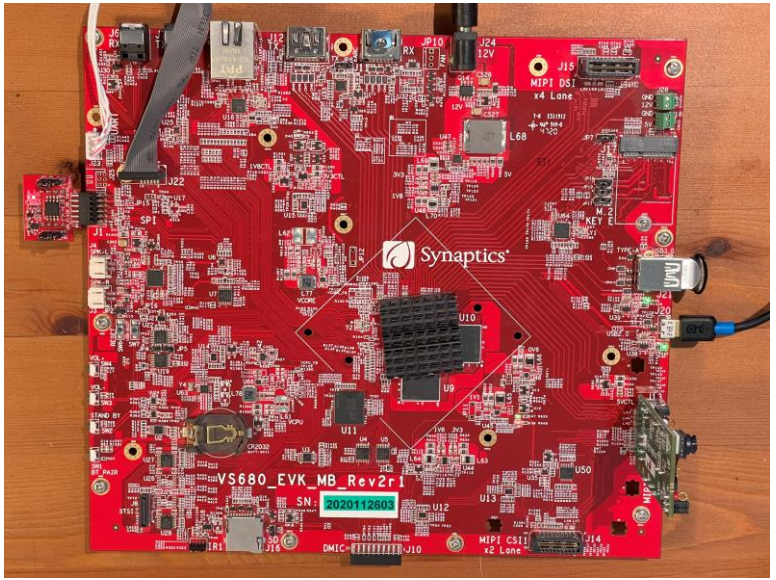


© 2021 Synaptics Incorporated

Flashing the Android Image

Flashing requires booting from an external SPI flash board. It contains a version of u-boot with support for reading the image from USB and writing it to the internal eMMC.






```
COM5 - PuTTY
bg7 a0 erom: release id 0x6b857a7, commit time Mon Dec 7 14:18:24 2020 -0800
bg7 a0 hal : release id 0xfa3f39a, commit time Thu Dec 17 19:30:11 2020 -0800
Enter PI Command Loop...
build-in module: release id 0x697d993, commit time Mon Dec 7 17:01:33 2020 -0800
Load sysinit
Start ACPU
init cpupll 400MHz
init mempll 3733MHz
init 2GB DDR, lib version r67382
asking BCM to load ddr phy fw.
DDR PHY firmware loading is done.
asking BCM to load ddr phy fw.
DDR PHY firmware loading is done.
asking BCM to load ddr phy fw.
DDR PHY firmware loading is done.
asking BCM to load ddr phy fw.
DDR PHY firmware loading is done.
asking BCM to load ddr phy fw.
DDR PHY firmware loading is done.
asking BCM to load ddr phy fw.
DDR PHY firmware loading is done.
hw_init done
custom table done
asking BCM to load miniloader.
try to verify flash miniloader
fail to verify miniloader, error 0xffe00101
init syspll 800MHz


U-Boot 2019.10-18897-ge8f2888 (Dec 22 2020 - 10:49:43 +0800)

Model: Synaptics VS680 EVK
       Watchdog enabled
DRAM:  2 GiB
Flash: SF: Detected w25q128fw with page size 256 Bytes, erase size 64 KiB, total 16 MiB
16 MiB
MMC:   sdhci@aa0000: 0
Loading Environment from SPI Flash... *** Warning - bad CRC, using default environment

In:    smuart@d000
Out:   smuart@d000
Err:   smuart@d000
Net:
Warning: ethernet@b60000 (eth0) using random MAC address - 7a:f7:bf:7c:7d:be
eth0: ethernet@b60000
ethernet@b60000 Waiting for PHY auto negotiation to complete..... TIMEOUT !
Could not initialize PHY ethernet@b60000
=> usb2emmc eMMCimg
starting USB...
Bus dwc2@c00000: Bus dwc3@d00000: Register 2000140 NbrPorts 2
Starting the controller
USB XHCI 1.10
scanning bus dwc2@c00000 for devices... 1 USB Device(s) found
scanning bus dwc3@d00000 for devices... 3 USB Device(s) found
       scanning usb for storage devices... 1 Storage Device(s) found
fatload usb 0:1 0x23752e00 eMMCimg/emmc_part_list
1062 bytes read in 36 ms (28.3 KiB/s)
switch to partitions #0, OK
mmc0(part 0) is current device


Caution! Your devices Erase group is 0x400
The erase range would be change to 0x0~0x3ff

start to create partiton factory_setting 16 16...
start to create partiton key_a 32 1...
```

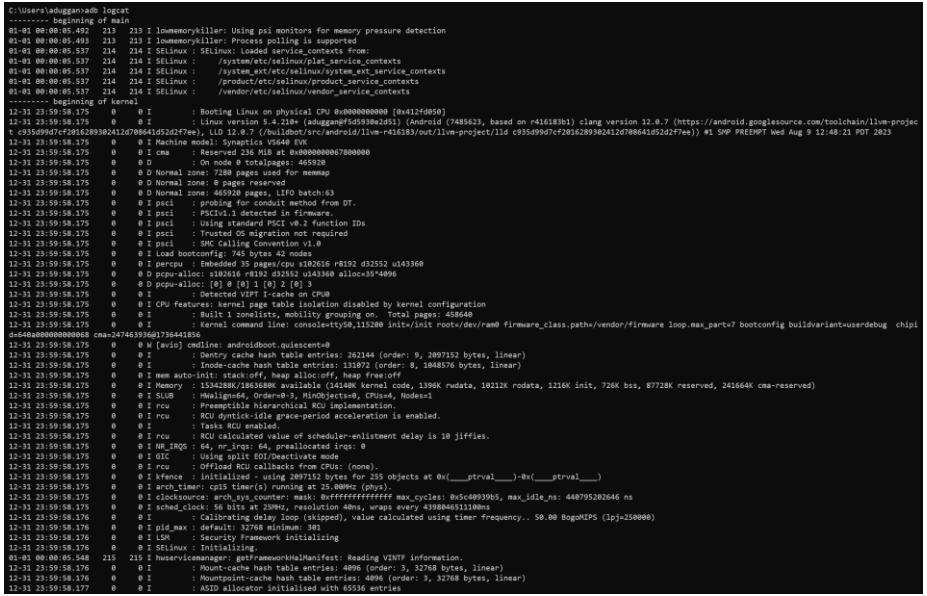
Run usb2emmc eMMCimg at the prompt.

© 2021 Synaptics Incorporated

Debugging

Android Studio and the Android NDK support debugging of apps.

To debug system level issues Android's logcat is very useful.
https://developer.android.com/tools/logcat



© 2021 Synaptics Incorporated

Working with Cameras

The MIPI-CSI camera module and external USB cameras will be detected by the Android Camera HAL and will be accessible to Android applications.

The MIPI-CSI camera module plugs into the MIPI-CSI0 x4 Lane connector.





© 2021 Synaptics Incorporated

Support Requests using Jira

We will use Jira to handle support requests and questions.

One or two accounts associated with an @ucf.edu email will have access to Jira. When filing Jiras please provide as much information as possible. For example, please include console output, logcat output, sample code or binaries, and any other information which could be relevant.

Meet Patel will be the main support contact.

https://synacsm.atlassian.net/servicedesk/customer/user/login?destination=portals

© 2021 Synaptics Incorporated