Rockchip RK2118 RKNN SDK Quick Start

ID: RK-KF-JC-414

Release Version: V2.1.0

Release Date: 2024-07-30

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document describes how to quickly get started using RK2118 NPU SDK.

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	Author	Date	Change Log	Reviewer
V2.1.0	Randall	2024-07-30	Support RK2118	Vincent

Table of Contents

---

1 Introduction

This document provides a detailed introduction for beginners on how to quickly use RKNN-Toolkit2 on a computer to perform model conversion and deploy it to a Rockchip development board.

Supported platforms: RK2118

2 Prepare Development Board

This chapter will explain how to connect the development board to a computer, divided into two parts:

- Introduction to Development Board and Connection Tools
- Connect the Development Board to the Computer

2.1 Introduction to Development Board

1. Development board

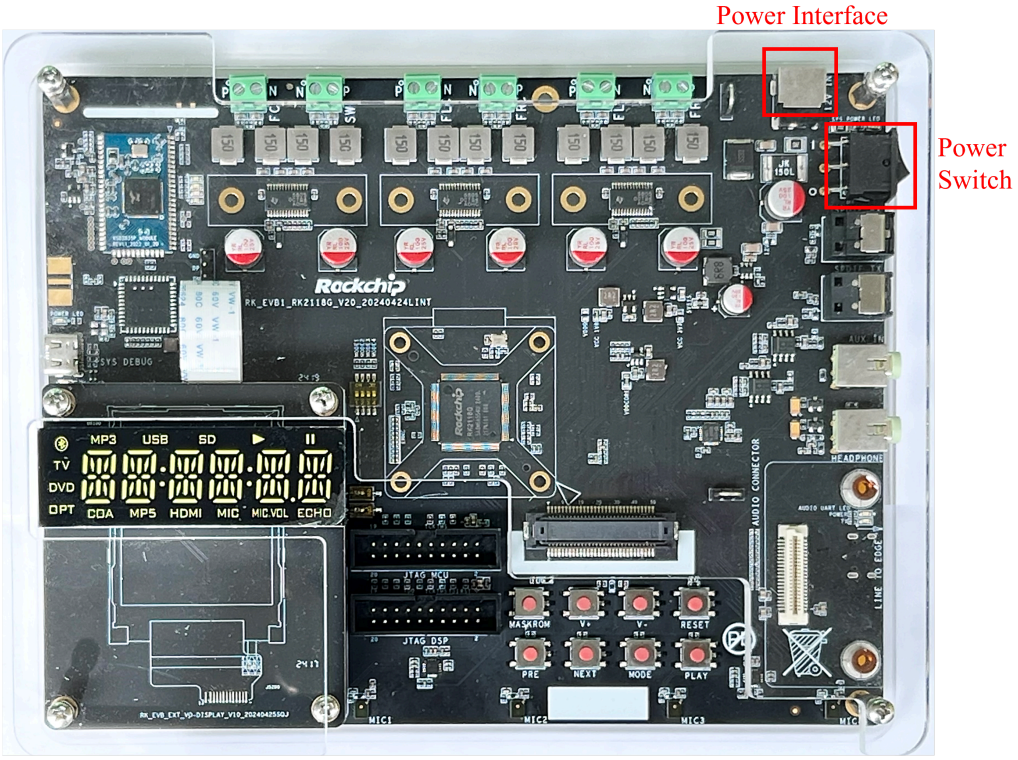

Figure 2-1 RK2118 development board

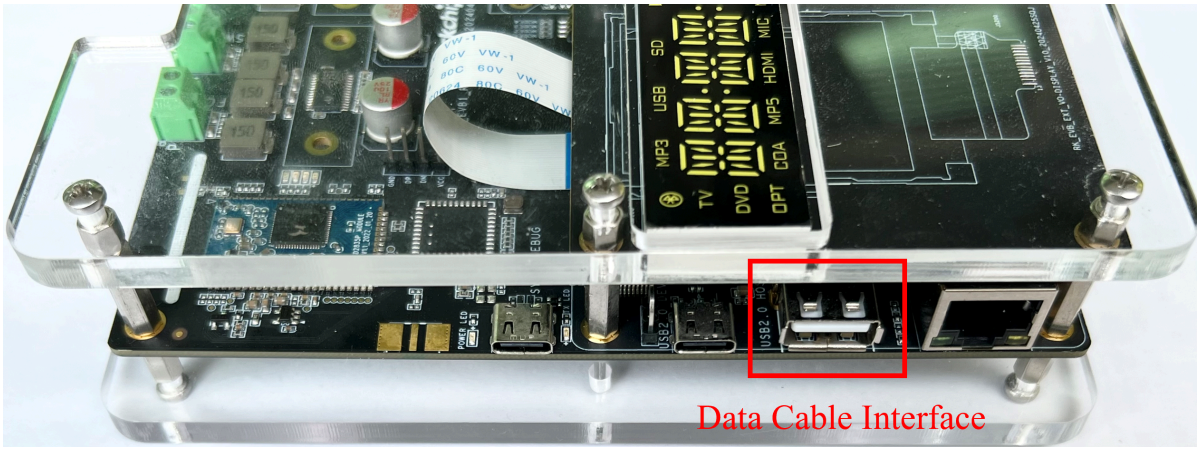

Figure 2-2 RK2118 development board

2. Data cable for connecting development board and computer

Figure2-3 USB-C USB data cable

Figure2-4 USB-A USB data cable

3. Power adapter

Figure 2-5 Power adapter with 12V-2A output

2.2 Connect the Development Board to the Computer

This section illustrates the connection process using the RK2118 development board as an example.

- Prepare a computer with Ubuntu 18.04, Ubuntu 20.04, or Ubuntu 22.04 operating system.
- Please note that the type and location of the data cable interface on the development board may vary between production batches. Refer to the provided schematic or documentation to locate the power interface and connect the development board's power adapter.
- Activate the power switch and wait for the development board's system to boot up.
- Use a USB Type-C cable to connect the 'SYSTEM DEBUG' port of the development board to the computer.
- Utilize a serial communication tool on the computer to monitor the status of the RK2118. For example, you can use the 'minicom' application to establish a serial connection. The serial port baud rate should be set to 1500000 8N1.

3 Prepare Development Environment

This chapter introduces how to install the development environment directly on the computer. The subsequent sample program running process will also be explained using direct installation as an example. If you need to run the sample program in a Docker environment, you can refer to Chapter 5 to prepare the development environment.

This chapter is divided into three parts:

- Download RKNN Related Repositories
- Install the RKNN-Toolkit2 Environment on the Computer
- Prepare RK2118 SDK

3.1 Download RKNN Related Repositories

It is recommended to create a new directory to store the RKNN repositories. For example, create a folder named "Projects" and place the RKNN-Toolkit2 repositories in that directory. Refer to the following commands:

```
# Create the 'Projects' folder
mkdir Projects

# Switch to this directory
cd Projects

# Download the RKNN-Toolkit2 repository
git clone https://github.com/airockchip/rknn-toolkit2.git --depth 1

# Notice:
# 1. Parameter --depth 1 means to clone only the latest commit
# 2. If the 'git clone' command fails, you can also download the compressed file directly from GitHub and unzip it locally in this directory.
```

Overall directory structure:

```
Projects
├── rknn-toolkit2
│   ├── doc
│   ├── rknn-toolkit2
│   │   ├── packages
│   │   ├── docker
│   │   └── ...
│   ├── rknpu2
│   │   ├── runtime
│   │   └── ...
│   └── ...
```

3.2 Install the RKNN-Toolkit2 Environment on the Computer

3.2.1 Install Python

If the Python 3.8 environment is not installed on your system, or if there are multiple versions of Python installed, it is recommended to use Miniforge to create a new Python 3.8 environment.

3.2.1.1 Install miniforge

In the terminal window on the computer, execute the following command to check whether miniforge is installed. If already installed, you can skip this step.

```
conda -V
# Reference output: conda 24.3.0, indicating that Miniforge Conda version is 24.3.0
# If it shows 'conda: command not found', it means Miniforge Conda is not installed.
```

If Miniforge is not installed, you can download the Miniforge installer from the following link:

```
wget -c https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Linux-x86_64.sh
```

Then, install miniforge using the following command:

```
chmod 777 Miniforge3-Linux-x86_64.sh
bash Miniforge3-Linux-x86_64.sh
```

3.2.1.2 Create Python Environment Using miniforge

In the terminal window on the computer, execute the following command to switch to the conda base environment:

```
source /opt/miniforge3/bin/activate
# Upon successful activation, the command prompt will change to the following form:
# (base) xxx@xxx:~$
```

Create a Python 3.8 environment named 'toolkit2' using the following command:

```
conda create -n toolkit2 python=3.8
```

Activate the 'toolkit2' environment. Subsequently, RKNN-Toolkit2 will be installed in this environment.

```
conda activate toolkit2
# (toolkit2) xxx@xxx:~$
```

3.2.2 Install Dependencies and RKNN-Toolkit2

After activating the 'toolkit2' environment, switch to the rknn-toolkit2 directory, install the required libraries based on requirements_cpxx.txt, and install RKNN-Toolkit2 using the wheel package. Refer to the following commands:

```
# Switch to the rknn-toolkit2 directory
cd Projects/rknn-toolkit2/rknn-toolkit2

# Choose the appropriate requirements file based on your Python version
# For example, for Python 3.8, use requirements_cp38.txt
pip install -r packages/requirements_cpxx.txt

# Install RKNN-Toolkit2
# Choose the appropriate wheel package based on your Python version and processor architecture:
# Where x.x.x is the version number of RKNN-Toolkit2, xxxxxxxx is the submission number, and cpxx is the python version number. Please replace with the actual values accordingly.
pip install packages/rknn_toolkit2-x.x.x+xxxxxxxx-cpxx-cpxx-linux_x86_64.whl
```

3.2.3 Check if the RKNN-Toolkit2 Environment is Installed Successfully

Execute the following command. if no errors are reported, it indicates that the RKNN-Toolkit2 environment has been successfully installed.

```
# Switch to Python interactive mode
python

# Import the RKNN class
from rknn.api import RKNN
```

If the installation fails, please refer to Chapter 10.2 'RKNN-Toolkit2 Installation Issue' in the 'Rockchip_RKNPU_User_Guide_RKNN_SDK_V2.1.0_EN.pdf' document. It provides detailed solutions for resolving RKNN-Toolkit2 environment installation failures.

3.3 Prepare RK2118 SDK

The example of NPU is included in the RK2118 SDK, so you need to prepare the RK2118 SDK source code first.

3.3.1 Repository Download

- Download repo tool. If repo is already installed, please skip this step.

```
git clone ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -b stable
export PATH=/path/to/repo:$PATH
```

- repository download

```
mkdir rt-thread
cd rt-thread
repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u
ssh://git@www.rockchip.com.cn/rtos/rt-thread/rk/platform/release/manifests -b master -m rk2118.xml
.repo/repo/repo sync
```

Note: This project requires authorization to access, please contact Rockchip to apply.

3.3.2 Compilation environment preparation

Please refer to <Rockchip_Developer_Guide_RK2118> in the "rt-thread/RKDocs/manuals/rk2118" directory to build the development environment.

3.3.3 RK2118 firmware compilation

3.3.3.1 Enable RKNPU support

Refer to <Rockchip_Developer_Guide_RK2118> Chapter 4 "CPU Firmware Compilation" to compile firmware and enable RKNPU support. Return to the main directory of the rt-thread project and execute the following command:

```
cd bsp/rockchip/rk2118
cp board/evb/defconfig .config
scons --menuconfig
```

Please make sure the following options are turned on:

```
RT-Thread Components  --->
 [*] DFS: device virtual file system  --->
     [*]  Using posix-like functions, open/read/write/close
     [*]  Using working directory
     (4)  The maximal number of mounted file system
     (4)  The maximal number of file system type
     (16)  The maximal number of opened files
     [*]  Using mount table for file system
     [*]  Enable elm-chan fatfs
RT-Thread rockchip RK2118 drivers  --->
 [*] Enable RKNPU
```

3.3.3.2 Compile firmware and package

```
cd bsp/rockchip/rk2118
scons -j32
./mkimage.sh board/evb/setting.ini
```

After the compilation is completed, files such as Image/Firmware.img will be generated.

3.3.3.3 Flashing firmware

Refer to "4.1.3.4 Firmware Burning" section of "Rockchip_Developer_Guide_RK2118" to flash firmware on development board.

4 Run Example Programs

This chapter will introduce how to quickly run example programs on the development board. It is divided into three parts:

- Introduction to RKNN SDK
- Model Conversion
- How to Run RKNN C Demo

4.1 Introduction to RKNN SDK

RK2118 RKNN SDK is similar to the SDK for RV1103/RV1106. It is in the rt-thread/components/rknpu directory of the RK2118 project, the directory structure is as follows:

```
rknpu
├── examples     # Example code
├── include      # Header file
├── lib          # Runtime library
├── rknn_server  # Interactive service program with DSP
└── SConscript
```

The examples directory includes some usage examples, such as MobileNet. Taking rknn_benchmark as an example, its directory structure is as follows:

```
rknn_benchmark/
├── README.md
├── res
│   ├── convert.py       # RKNN Model Conversion Script
├── SConscript           # Compile Script
├── src
│   └── rknn_benchmark.c  # RKNN C Example
└── tools
    └── update_res.py     # Generate the RKNN model input file
```

4.2 Model Conversion

The following uses MobileNetV1 as an example to introduce how to convert RKNN model.

Enter the rt-thread/components/rknpu/examples/rknn_benchmark/res directory and run the convert.py script, which converts the original pb model into an RKNN model. The reference command is as follows:

```
cd rt-thread/components/rknpu/examples/rknn_benchmark/res
python convert.py
```

If the operation is successful, mobilenetv1.rknn and dog.bin will be generated. dog.bin is the model input file.

4.3 How to use RKNN C Demo

The following uses rknn_benchmark as an example to introduce how to use RKNN C Demo.

Note: Different RKNN C Demos have different usages. Please follow the steps in README.md in their respective directories.

4.3.1 Compile rknn_benchmark

The firmware compiled by referring to section 3.3.3 RK2118 firmware compilation already contains the rknn_benchmark program by default. The `rknn_benchmark` can be used directly after the system starts.

4.3.2 Push files to the board

Since the storage space of the board is limited, you can use a USB flash drive to store the RKNN model during testing (only FAT32 format is supported). You can also package the RKNN model into the firmware. Here we take a USB flash drive as an example. Copy mobilenetv1.rknn and dog.bin generated in "4.2 Model Conversion" to the root directory of the USB flash drive. And insert the USB flash drive into the "USB 2.0 HOST" port of the board. Execute the following command on the board to see the model file.

```
ls /
```

4.3.3 Running Demo on the board

Execute the following command to run rknn_benchmark on the board:

```
# Enter the board (serial port)

cd /

# Usage: ./rknn_benchmark <model_path> <input_path>
rknn_benchmark mobilenetv1.rknn dog.bin
```

results reference:

```
[156] 19.187500
[155] 16.234375
[205] 13.539062
[284] 12.226562
[260] 10.179687
```

Note: The running results only print the TOP5, and softmax is not performed by default.

5 RKNN Model Conversion in Docker (Optional)

If you need to run the RKNN Python Demo in a Docker environment, please refer to the content in this chapter to prepare the development environment.

Please note that here we provide a Docker image containing the RKNN-Toolkit2 environment, allowing users to directly run RKNN Python Demo without worrying about environment installation issues. However, this Docker image only includes a clean RKNN-Toolkit2 environment and is specifically designed for running RKNN Python Demo.

This chapter is divided into three parts:

- Install Docker
- Install the RKNN-Toolkit2 Environment in Docker
- How to Run RKNN Python Demo

5.1 Install Docker

If Docker is already installed, you can skip this step. If it is not installed, please follow the official documentation for installation.

Docker installation official documentation link: https://docs.docker.com/install/linux/docker-ce/ubuntu/

Note: It is necessary to add the user to the 'docker' group.

```
# Create the docker group
sudo groupadd docker
# Add the current user to the docker group
sudo usermod -aG docker $USER
# Update to activate the docker group
newgrp docker

# Verify that docker commands can be executed without sudo
docker run hello-world
```

The reference output for successful installation is as follows:

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:88ec0acaa3ec199d3b7eaf73588f4518c25f9d34f58ce9a0df68429c5af48e8d
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
```

5.2 install the RKNN-Toolkit2 Environment in Docker

5.2.1 Prepare the RKNN-Toolkit2 Image

This section introduces two methods for creating the RKNN-Toolkit2 image environment. You can choose either method.

1. Create the RKNN Toolkit2 image using a Dockerfile

   In the RKNN-Toolkit2 project, the 'docker/docker_file' folder provides a Dockerfile for building the RKNN-Toolkit2 development environment. Users can create an image using the 'docker build' command, as shown below:

```
# Note: Replace 'xx' and 'x.x.x' with the actual version numbers
cd Projects/rknn-toolkit2/rknn-toolkit2/docker/docker_file/ubuntu_xx_xx_cpxx
docker build -f Dockerfile_ubuntu_xx_xx_for_cpxx -t rknn-toolkit2:x.x.x-cpxx .
```

2. Create the RKNN Toolkit2 image by loading a pre-packaged Docker image file

   Download the RKNN-Toolkit2 project files for the corresponding version from the following link. After extracting the files, the 'docker/docker_image' folder provides a pre-packaged Docker image with all the development environments.

   Download link from the cloud drive: https://console.zbox.filez.com/l/I00fc3 (Extraction code: rknn)

   Execute the following command to load the corresponding Python version of the image file.

```
# Note: Replace 'x.x.x' with the actual version number of RKNN-Toolkit2, and 'cpxx' with the Python version
docker load --input rknn-toolkit2-x.x.x-cpxx-docker.tar.gz
```

5.2.2 Query Image Information

After successfully creating or loading an image, you can view Docker image information using the following command:

```
docker images
```

The corresponding RKNN-Toolkit2 image information is displayed as follows.

```
REPOSITORY      TAG          IMAGE ID       CREATED       SIZE
rknn-toolkit2   x.x.x-cpxx   xxxxxxxxxxxx   1 hours ago   5.89GB
```

5.3 Model Conversion

Note: Please ensure that the board has the RKNPU2 environment installed. If not, refer to the content in Chapter 3.4 for installation instructions.

- Map the files and run the container

  Please refer to Chapter 3.1 for downloading the rt-thread project locally. Then, map it into the container and run the container using the 'docker run' command. After running, it will switch to the container's bash environment. Refer to the following commands:

```
# Use the docker run command to create and run the RKNN Toolkit2 container
# Map local files into the container by attaching the -v <host src folder>:<image dst folder> parameter
docker run -t -i --privileged \
   -v /dev/bus/usb:/dev/bus/usb \
   -v /Projects/rt-thread:/rt-thread \
   rknn-toolkit2:x.x.x-cpxx \
   /bin/bash
```

- Model Conversion

  Switch to the rt-thread/components/rknpu/examples/rknn_benchmark/res directory and run the convert.py script.

```
cd rt-thread/components/rknpu/examples/rknn_benchmark/res
python convert.py
```

6 FAQ

6.1 No adb Device Visible

RK2118 currently does not support debugging via adb, so you will not see any adb devices when using this command.

6.2 What is the role of the rknn_server service?

The rknn_server service on RK2118 is not used for board debugging with RKNN-Toolkit2. Instead, it facilitates the communication service between the DSP and the MCU.

7 Reference Documents

- For more detailed usage and interface descriptions of RKNN-Toolkit2 and RKNPU2, please refer to the manual *"Rockchip_RKNPU_User_Guide_RKNN_SDK_V2.1.0_CN.pdf"*.
- For relevant documents of RK2118, please refer to *"Rockchip_Developer_Guide_RK2118.pdf"*.