# HowToMake

English | [简体中文](HowToMake.md)

* Note that compilation needs to be done in the sophon-stream directory.

## x86/arm PCIe Platform
```bash
mkdir build
cd build
cmake ..
make -j4
```

## SoC Platform
Usually, when cross-compiling programs on an x86 host, you need to set up a cross-compilation environment using the SOPHON SDK on the x86 host. You will package the required header files and library files into the `sophon_sdk_soc` directory. You can either download the SOPHON SDK and package it yourself or download our pre-packaged files (choose one based on your SOC environment).

The following three files correspond to the v23.03.01, v23.05.01, and v23.07.01 versions of the official SDK.

```bash
pip3 install dfss
python3 -m dfss --url=open@sophgo.com:/sophon-stream/soc-sdk/soc0301.tar.gz
python3 -m dfss --url=open@sophgo.com:/sophon-stream/soc-sdk/soc0501.tar.gz
python3 -m dfss --url=open@sophgo.com:/sophon-stream/soc-sdk/soc0701.tar.gz
```

When cross-compiling, make sure to provide the absolute path to `SOPHON_SDK_SOC`:

```bash
mkdir build
cd build
cmake ../ -DTARGET_ARCH=soc -DSOPHON_SDK_SOC=/path/to/sophon_sdk_soc
make -j4
```

If the required SDK version was not provided in the previous sections, you'll need to package the `soc-sdk` by following this process. Note that for SE9 devices, sophon-img in the following command needs to be changed to sophon-img-se9, and sophon-mw needs to be changed to sophon_media.

1. Unzip the contents of the `libsophon_soc_<x.y.z>_aarch64.tar.gz` file located in the `sophon-img` directory of the SDK. Copy all the contents of the `lib` and `include` folders to your `soc-sdk` directory.

```bash
cd sophon-img_<date>_<hash>
# Create a root directory for dependency files
mkdir -p soc-sdk
# Unzip the 'libsophon_soc_${x.y.z}_aarch64.tar.gz' from the 'sophon-img' release package, where x.y.z is the version number.
```

```bash
tar -zxf libsophon_soc_<x.y.z>_aarch64.tar.gz
# Copy the relevant library directories and header files to the root directory
of the dependency files
cp -rf libsophon_soc_<x.y.z>_aarch64/opt/sophon/libsophon-<x.y.z>/lib ${soc-sdk}
cp -rf libsophon_soc_<x.y.z>_aarch64/opt/sophon/libsophon-<x.y.z>/include ${soc-
sdk}
```

2. Unzip the contents of the `sophon-mw-soc_<x.y.z>_aarch64.tar.gz` file located in the `sophon-mw` directory of the SDK. Copy all the contents of the `lib` and `include` folders under `sophon-mw` to your `soc-sdk` directory.

```bash
cd sophon-mw_<date>_<hash>
# Unzip the 'sophon-mw-soc_<x.y.z>_aarch64.tar.gz' from the 'sophon-mw' package,
where x.y.z is the version number.
tar -zxf sophon-mw-soc_<x.y.z>_aarch64.tar.gz
# Copy the library directories and header files of ffmpeg and OpenCV to the root
directory of the dependency files.
cp -rf sophon-mw-soc_<x.y.z>_aarch64/opt/sophon/sophon-ffmpeg_<x.y.z>/lib ${soc-
sdk}
cp -rf sophon-mw-soc_<x.y.z>_aarch64/opt/sophon/sophon-ffmpeg_<x.y.z>/include $
{soc-sdk}
cp -rf sophon-mw-soc_<x.y.z>_aarch64/opt/sophon/sophon-opencv_<x.y.z>/lib ${soc-
sdk}
cp -rf sophon-mw-soc_<x.y.z>_aarch64/opt/sophon/sophon-opencv_<x.y.z>/include $
{soc-sdk}
```

## Building Using Development Docker Image

If your local environment is partially incompatible and it's inconvenient to change it, you can use the Docker image we provide for compiling.

Download the image through dfss:

```bash
pip3 install dfss
python3 -m dfss --url=open@sophgo.com:/sophon-stream/docker/stream_dev.tar
```

If you're using Docker for the first time, you can execute the following commands to install and configure it (only for the first-time setup):

```bash
sudo apt install docker.io
sudo systemctl start docker
sudo systemctl enable docker
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

Load the image from the downloaded image directory:

```bash
docker load -i stream_dev.tar
```

You can check the loaded image using `docker images`, which defaults to `stream_dev:latest`.

Create a container:

```bash
docker run --privileged --name stream_dev -v $PWD:/workspace -it
stream_dev:latest
# stream_dev is just an example name; please specify your own container name.
```

The `workspace` directory in the container will be mounted to the directory of the host machine where you ran `docker run`. You can use this container to compile the project.

## Compilation Results

1. `framework` and `element` will generate dynamic link libraries in `build/lib`.

2. `samples` will generate executable files in the `build` folder within the respective sample directories. For example, `samples/yolov5` will generate the `yolov5_demo` executable file under `samples/yolov5/build`.

For PCIe platforms, you can run tests directly on the PCIe platform. For SoC platforms, you'll need to copy the dynamically linked libraries and executable files generated by cross-compilation to the SoC platform for testing.

3. After cross-compilation, when copying the compiled results to your Micro Server, make sure to maintain the directory structure. You can use the following command to directly copy the `sophon-stream` directory from your host to your Micro Server:

```bash
scp -r ./sophon-stream linaro@<your ip>:<your path>
```

You can set the IP address and file directory of your Micro Server as per your specific situation.