

PaddlePi-K210 开发环境搭建指南

1、概述

本文介绍如何在 *Ubuntu* 或 *Windows* 下为 K210 搭建开发环境，用户可根据个人喜好选择开发环境。

2、Windows 命令行开发环境搭建

2.1 安装工具链

1. 下载 [cmake](#) V3.0 之后的 Windows 版本。这里以 V3.12.4 版本为例。

把 *cmake* 安装到 `D:\cmake` 目录，并把 `D:\cmake\bin` 目录添加到 *PATH* 环境变量。

2. 打开一个新的 cmd 窗口，输入 `cmake -version` 命令，若看到如下信息说明设置正确。

```
C:\Users\mbd2>cmake -version
cmake version 3.12.4

CMake suite maintained and supported by Kitware (kitware.com/cmake).
C:\Users\mbd2>
```











3. 从 [Kendryte Github](https://github.com/kendryte/kendryte-gnu-toolchain/releases) (<https://github.com/kendryte/kendryte-gnu-toolchain/releases>) 下载 Windows 版本工具链。打开网页后展开 *Assets* 可看到下载链接。

Extract

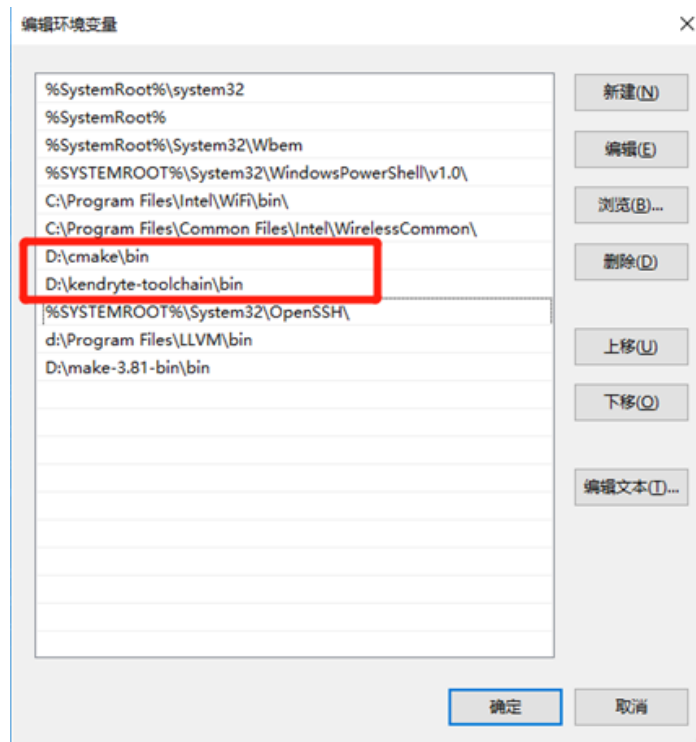
tar.xz is for fast download
tar.bz2 and zip is for arduino supported format

```
tar -Jxvf kendryte-toolchain.tar.xz
tar -jxvf kendryte-toolchain.tar.bz2
unzip kendryte-toolchain.zip
```

▼ Assets 10

 kendryte-toolchain-osx-mojave-8.2.0-20190409.tar.bz2	46.3 MB
 kendryte-toolchain-osx-mojave-8.2.0-20190409.tar.xz	21.4 MB
 kendryte-toolchain-ubuntu-amd64-8.2.0-20190409.tar.bz2	45.6 MB
 kendryte-toolchain-ubuntu-amd64-8.2.0-20190409.tar.xz	19.2 MB
 kendryte-toolchain-win-amd64-8.2.0-20190409.tar.xz	18.2 MB
 kendryte-toolchain-win-amd64-8.2.0-20190409.zip	51.4 MB
 kendryte-toolchain-win-i386-8.2.0-20190409.tar.xz	16.8 MB
 kendryte-toolchain-win-i386-8.2.0-20190409.zip	48.6 MB
 Source code (zip)	
 Source code (tar.gz)	

4. 将下载后的文件解压缩，将解压后的 `D:\kendryte-toolchain\bin` 目录添加到 `PATH` 环境变量，如下图。



5. 重新打开一个 cmd 窗口，输入 `riscv64-unknown-elf-gcc -v` 命令，看到如下信息说明编译器设置正确。

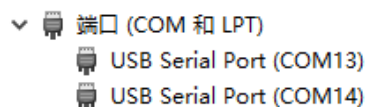
```
C:\Users\mbad2>riscv64-unknown-elf-gcc -v
Using built-in specs.
COLLECT_GCC=riscv64-unknown-elf-gcc
COLLECT_LTO_WRAPPER=e:/kendryte-toolchain/bin/./libexec/gcc/riscv64-unknown-elf/8.2.0/lto-wrapper.exe
Target: riscv64-unknown-elf
Configured with: /workdir/riscv-gcc/configure --target=riscv64-unknown-elf --host=x86_64-w64-mingw32 --prefix=/opt/kendryte-toolchain --disable-shared --enable-threads=posix --enable-languages=c,c++ --enable-libatomic --without-system-zlib --enable-tls --with-newlib --with-sysroot=/opt/kendryte-toolchain/riscv64-unknown-elf --with-native-system-header-dir=/include --disable-libmudflap --disable-libssp --disable-libquadmath --disable-libgomp --disable-nls --src=../riscv-gcc --enable-checking=yes --disable-multilib --with-abi=lp64f --with-arch=rv64imafc 'CFLAGS_FOR_TARGET=-Os -ffunction-sections -fdata-sections -mmodel=medany' 'CXXFLAGS_FOR_TARGET=-Os -ffunction-sections -fdata-sections -mmodel=medany'
Thread model: posix
gcc version 8.2.0 (GCC)
```

2.2 安装 OPENOCD

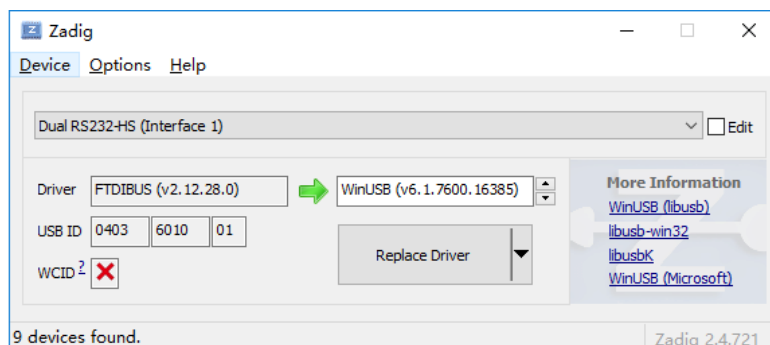
若不进行在线仿真器调试，可跳过这个步骤，参考 2.3 节开始编译程序。

1. 从 [Kendryte Github](https://github.com/kendryte/openocd) (<https://github.com/kendryte/openocd-kendryte/releases>) 下载 Windows 版本的 Openocd 并解压缩。
2. 安装 FT2232 驱动

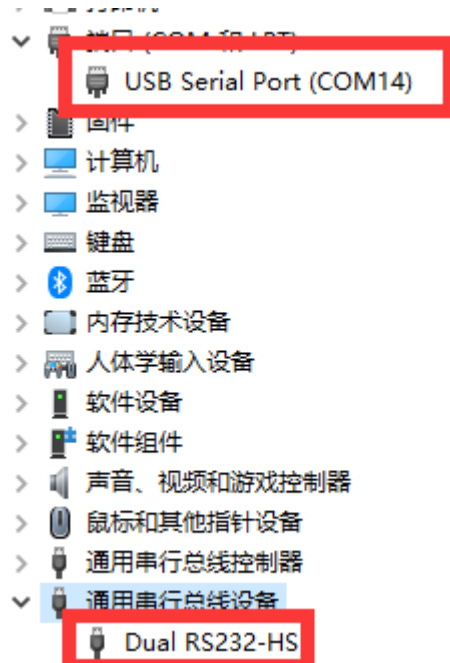
将 FT2232 通过 USB 连接后，默认设备管理器出现两个串口设备，如下图所示。



运行 `XXX\kendryte-openocd\tool` 下的 `zadig-2.4.exe` (XXX 为 `kendryte-openocd` 的保存路径)，点击 `Options->List All Devices`，在下拉列表选中 `Dual RS232-HS (Interface 1)`，点击【**Replace Driver**】即可将驱动替换为 WinUSB 驱动。



注意，替换 Dual RS232-HS(Interface 1)，***不要将 Dual RS232-HS (Interface 0) 转为 WinUSB**。替换过驱动后建议断开 FT2232 连接后重新连接。替换完成后，驱动将呈现如下状态：



3. OpenOCD 配置

下载 kendryte-openocd-0.X.X-win32, 在 tcl 文件夹中, 添加 ft232.cfg 文件, ft232.cfg 填写如下内容:

```
interface ftdi

ftdi_vid_pid 0x0403 0x6010

ftdi_channel 1

ftdi_layout_init 0x00e8 0x00eb

transport select jtag

ftdi_tdo_sample_edge falling

adapter_khz 10000

gdb_port 3333

telnet_port 4444

set _CHIPNAME riscv

jtag newtap $_CHIPNAME cpu -irlen 5 -expected-id 0x04e4796b

set _TARGETNAME $_CHIPNAME.cpu
```

```
target create $_TARGETNAME riscv -chain-position $_TARGETNAME
```

```
init
```

```
halt
```

2.3 编译

1. 从 [Kendryte Github](https://github.com/kendryte/kendryte-standalone-sdk/releases) (<https://github.com/kendryte/kendryte-standalone-sdk/releases>) 下载 `kendryte-standalone-sdk`。
2. 在 `kendryte-standalone-sdk` 目录下创建 `build` 目录。
3. 打开 `cmd`，进入 `build` 目录，在 `build` 目录下运行 `cmake`。

```
> cmake .. -DPROJ=hello_world -G "MinGW Makefiles"
```

当输出以下结果表示 `cmake` 正常。

```
Makefile created.

-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/mbed2/Desktop/kendryte-standalone-sdk/build
```

4. 在 `build` 目录下编译。

```
> make -j
```

当编译输出以下结果表示 `make` 成功。

```
Scanning dependencies of target hello_world
[ 97%] Building C object CMakeFiles/hello_world.dir/src/hello_world/main.c.obj
[100%] Linking C executable hello_world
Generating .bin file ...
[100%] Built target hello_world
```

2.4 调试

若不进行在线仿真器调试，可跳过这个步骤，参考 2.5 节下载方法验证程序运行结果。

1. 运行 `openocd`。

从 `cmd` 进入 `kendryte-openocd/bin` 目录，运行 `openocd.exe -f ..\tcl\ft232.cfg`。

```
> .\openocd.exe -f ..\tcl\ft232.cfg
```

2. GDB 调试

从另外一个 `cmd` 窗口进入 `kendryte-standalone-sdk\build` 目录。

```
> riscv64-unknown-elf-gdb hello_world --eval-command="target remote:3333"
```

输入: `load`

输入: `c` 运行程序, 此时查看串口, 将看到 Core 0 Hello World 的字符输出。

其中:

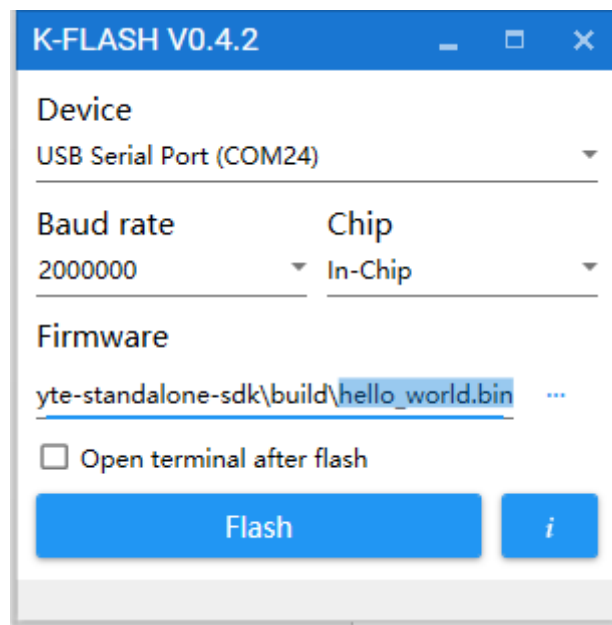
- `hello_world` 是编译过程生成的 *elf* 文件。
- `3333` 是 `openocd` 监听的端口号。

成功连接 `openocd` 后, 就可以使用 `load`、`break`、`continue` 等 `gdb` 命令来调试了。

注: `gdb` 手册参考 [gdb website](#)。

2.5 下载

1. 从 [Kendryte Github](https://github.com/kendryte/kendryte-flash-windows/releases) (<https://github.com/kendryte/kendryte-flash-windows/releases>) 下载 Windows 版本 `K-flash` 工具。
2. 开发板插上 Type-C, 上电进入 ISP 模式。运行 `K-flash.exe`, 如下图所示: 自动识别 COM 号及设置波特率, 选择 *Device* 和 *Firmware*, 例如 `hello_world.bin`, 点击 *Flash* 开始下载。



下载完成后程序将自动开始运行, 通过串口可看到有 “Core0 hello_world” 字符输出。

3、Ubuntu 18.04 命令行开发环境搭建

3.1 安装工具链

1. 安装 `build-essential` 以获取 `make` 工具。

```
$ sudo apt install build-essential
```

若如上命令执行不成功，执行如下命令更新一下软件源。









```
$ sudo apt update
```

2. 安装 `cmake`。

```
$ sudo apt install cmake
```

3. 从 Kendryte Github (<https://github.com/kendryte/kendryte-gnu-toolchain/releases>) 下载如下图所示的 Ubuntu 版本工具链，放到 `/opt` 目录并解压缩。

▼ Assets 10

 kendryte-toolchain-osx-mojave-8.2.0-20190409.tar.bz2	46.3 MB
 kendryte-toolchain-osx-mojave-8.2.0-20190409.tar.xz	21.4 MB
 kendryte-toolchain-ubuntu-amd64-8.2.0-20190409.tar.bz2	45.6 MB
 kendryte-toolchain-ubuntu-amd64-8.2.0-20190409.tar.xz	19.2 MB
 kendryte-toolchain-win-amd64-8.2.0-20190409.tar.xz	18.2 MB
 kendryte-toolchain-win-amd64-8.2.0-20190409.zip	51.4 MB
 kendryte-toolchain-win-i386-8.2.0-20190409.tar.xz	16.8 MB
 kendryte-toolchain-win-i386-8.2.0-20190409.zip	48.6 MB
 Source code (zip)	
 Source code (tar.gz)	

```
$ sudo mv kendryte-toolchain-ubuntu-amd64-8.2.0-20190409.tar.bz2 /opt
```

```
$ cd /opt
```

```
$ sudo tar -jxvf kendryte-toolchain-ubuntu-amd64-20190409.tar.bz2
```

打开 `~/.bashrc` 文件，在文件末尾添加如下一行，将 `/opt/kendryte-toolchain/bin` 目录添加到 `PATH` 环境变量。

```
export PATH=$PATH:/opt/kendryte-toolchain/bin
```

使修改生效。

```
$ source ~/.bashrc
```

3.2 安装 OPENOCD

若不进行在线仿真器调试，可跳过这节内容，参考 3.3 节开始编译。

1. 从 Kendryte Github (<https://github.com/kendryte/openocd-kendryte/releases>) 下载 Ubuntu 版本的 Openocd。这里以 kendryte-openocd-0.2.3-ubuntu64.tar.gz 为例。

```
$ sudo mv ken-openocd-0.2.3-ubuntu64.tar.gz /opt  
  
$ cd opt/  
  
$ sudo tar -zxvf kendryte-openocd-0.2.3-unbutu64.tar.gz
```

注：可能需要安装如下依赖包。

```
$ sudo apt install libusb-dev libftdi-dev libhidapi-dev
```

2. 确认 kendryte-openocd/tcl/ft232.cfg 文件是否存在，如果 ft232.cfg 不存在，需要创建 ft232.cfg 文件。

```
$ touch /opt/kendryte-openocd/tcl/ft232.cfg  
  
$ vi /opt/kendryte-openocd/tcl/ft232.cfg
```

添加如下内容：

```
interface ftdi  
  
ftdi_vid_pid 0x0403 0x6010  
  
ftdi_channel 1  
  
ftdi_layout_init 0x00e8 0x00eb  
  
transport select jtag  
  
ftdi_tdo_sample_edge falling  
  
adapter_khz 10000  
  
gdb_port 3333  
  
telnet_port 4444  
  
set _CHIPNAME riscv
```



```
jtag newtap $_CHIPNAME cpu -irlen 5 -expected-id 0x04e4796b

set _TARGETNAME $_CHIPNAME.cpu

target create $_TARGETNAME riscv -chain-position $_TARGETNAME

init

halt
```

3. 确认/etc/udev/rules.d/ftdi-usb.rules 文件是否存在，如果 ftdi-usb.rules 不存在，需要创建 ftdi-usb.rules 文件。

```
$sudo touch /etc/udev/rules.d/ftdi-usb.rules

$ sudo vi /etc/udev/rules.d/ftdi-usb.rules
```

添加如下内容：

```
ACTION=="ADD", ATTR{IDVENDOR}=="0403", MODE:="666"
```

3.3 编译

1. 从 [Kendryte Github](https://github.com/kendryte/kendryte-standalone-sdk/releases) (<https://github.com/kendryte/kendryte-standalone-sdk/releases>) 下载 [kendryte-standalone-sdk](#)，目前最新版本 V0.5.6。

V0.5.6

zzxcanaan released this 7 days ago

- Major change
 - Add irda rs485
 - Add rtc tick interrupt handler
 - Add rtc alarm interrupt handler
 - Modify system default print uart
 - Update KPU driver for latest NNCASE
 - Delete freertos

Assets 2

- Source code (zip)
- Source code (tar.gz)

2. 在 [kendryte-standalone-sdk](#) 目录下创建 [build](#) 目录。
3. 进入 [build](#) 目录后运行 *cmake*。

```
$ cmake .. -DPROJ=hello_world -DTOOLCHAIN=/opt/kendryte-toolchain/bin
```

5. 编译。

```
$ make -j
```

3.4 调试

若不进行在线仿真器调试，可跳过这个步骤，参考 3.5 节下载以验证程序运行结果。

1. 运行 openocd。

```
$ cd /opt/kendry-openocd
$ sudo ./bin/openocd -f ./tcl/ft232.cfg
```

正常将显示如下内容：

Kendryte Open On-Chip Debugger For RISC-V v0.1.3 (20180912)

Licensed under GNU GPL v2

ftdi samples TDO on falling edge of TCK

adapter speed: 10000 kHz

Info : clock speed 10000 kHz

Info : JTAG tap: riscv.cpu tap/device found: 0x04e4796b (mfg: 0x4b5
(<unknown>), part: 0x4e47, ver: 0x0)

Info : [0] Found 4 triggers

Info : [1] Found 4 triggers

[1] halted at 0x8000af8a due to debug interrupt

Info : Examined RISC-V core; found 2 harts, XLEN=64,
misa=0x800000000014112d

Info : Listening on port 3333 for gdb connections

[1] halted at 0x8000af8a due to debug interrupt

[0] halted at 0x800015f0 due to debug interrupt

Info : Listening on port 6666 for tcl connections

Info : Listening on port 4444 for telnet connections

2. 运行 gdb。

进入 `kendryte-standalone-sdk/build` 目录，打开另一个 *terminal*。

```
$ riscv64-unknown-elf-gdb hello_world --eval-command="target remote:3333"
```

其中：

- `hello_world` 是编译过程生成的 *elf* 文件。
- `3333` 是 openocd 监听的端口号。

成功连接 openocd 后，就可以使用 *load*、*break*、*continue* 等 gdb 命令来调试了。

注：gdb 手册参考 [gdb website](#)。

3.5 下载

1. 从 [Kendryte Github](https://github.com/kendryte/kflash.py/releases) (<https://github.com/kendryte/kflash.py/releases>) 下载 `kflash.py` 脚本，最新版本 V0.8.2。
2. 下载 bin 文件到 K210。

```
$ sudo python3 kflash.py kendryte-standalone-sdk/build/hello_world.bin
```

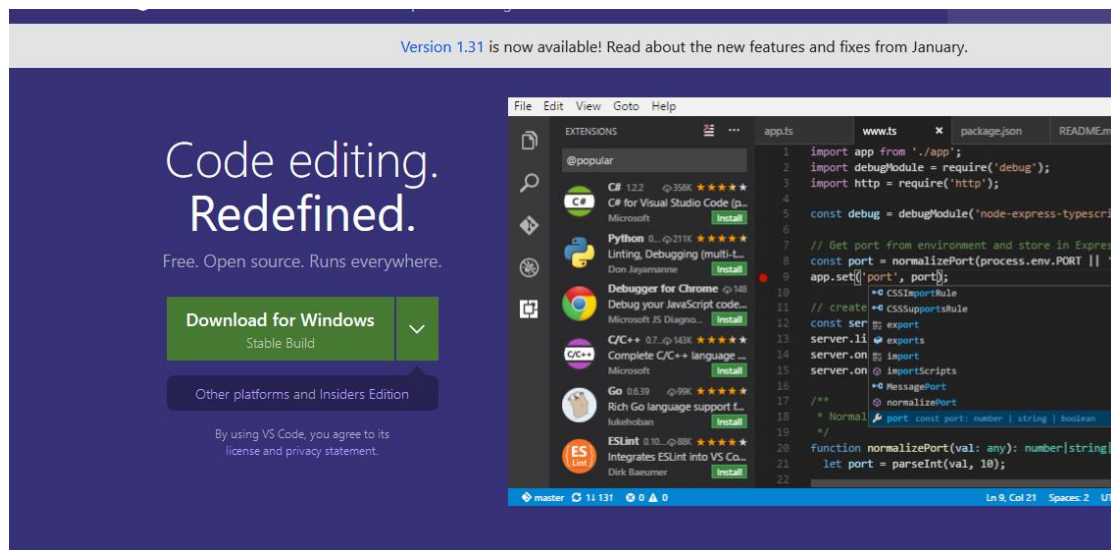
注：可以使用 `-p` 选项指定下载端口。

下载完成后程序将自动开始运行，通过串口可看到有 “Core 0 hello_world” 字符输出。

4、Visual Studio Code 开发环境搭建

4.1 下载并安装 VSCODE

下载地址：<https://code.visualstudio.com/>



4.2 下载安装 CMAKE

下载地址：<https://cmake.org/download/>

The release was packaged with CPack which is included as part of the release. The .sh files are self extracting gzipped tar files. To install a .sh file, run it with /bin/sh and follow the directions. The OS-machine.tar.gz files are gzipped tar files of the install tree. The OS-machine.tar.Z files are compressed tar files of the install tree. The tar file distributions can be untared in any directory. They are prefixed by the version of CMake. For example, the Linux-x86_64 tar file is all under the directory cmake-Linux-x86_64. This prefix can be removed as long as the share, bin, man and doc directories are moved relative to each other. To build the source distributions, unpack them with zip or tar and follow the instructions in Readme.txt at the top of the source tree. See also the [CMake 3.14 Release Notes](#). Source distributions:

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.14.0-rc2.tar.gz cmake-3.14.0-rc2.tar.Z
Windows Source (has \r\n line feeds)	cmake-3.14.0-rc2.zip

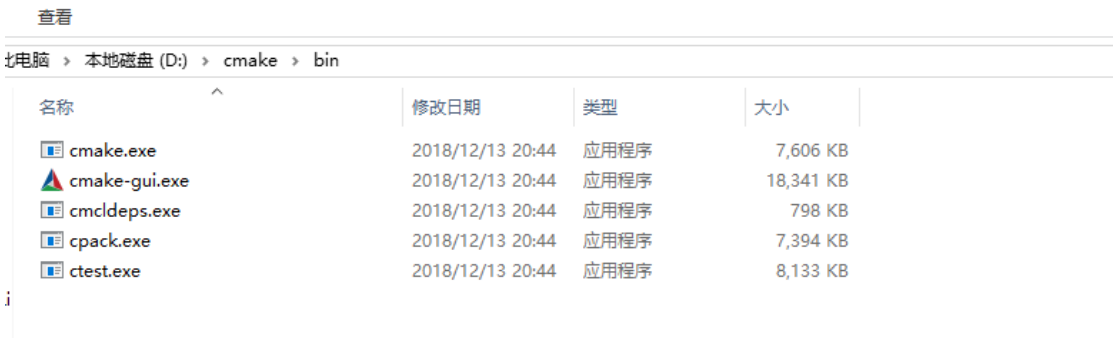
Binary distributions:

Platform	Files
Windows win64-x64 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.14.0-rc2-win64-x64.msi
Windows win64-x64 ZIP	cmake-3.14.0-rc2-win64-x64.zip
Windows win32-x86 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.14.0-rc2-win32-x86.msi
Windows win32-x86 ZIP	cmake-3.14.0-rc2-win32-x86.zip
Mac OS X 10.7 or later	cmake-3.14.0-rc2-Darwin-x86_64.dmg cmake-3.14.0-rc2-Darwin-x86_64.tar.gz
Linux x86_64	cmake-3.14.0-rc2-Linux-x86_64.sh cmake-3.14.0-rc2-Linux-x86_64.tar.gz

Download verification:

Role	Files
------	-------

安装 cmake，本示例安装到 D 盘根目录下。



4.3 下载 K210 的 TOOLCHAIN

下载地址：<https://github.com/kendryte/kendryte-gnu-toolchain/releases>

Pull requests Issues Marketplace Explore

kendryte / kendryte-gnu-toolchain Watch 17 Star 16 Fork 4

Code Issues 2 Pull requests 0 Projects 0 Wiki Security Insights

Releases Tags

Kendryte GNU Toolchain v8.2.0-20190409
vowstar released this on 10 Apr · 4 commits to develop since this release

Version

- Binutils 2.31.51
- GDB 8.2
- GCC 8.2.0

Build

- Linux

```
./configure --prefix=/opt/kendryte-toolchain --with-cmodel=medany --with-arch=rv64imafc --with-a
make -j8
```
- OSX

```
./configure --prefix=/usr/local/opt/kendryte-toolchain --with-cmodel=medany --with-arch=rv64imaf
make -j8
```
- Windows

Assets 10

kendryte-toolchain-osx-mojave-8.2.0-20190409.tar.bz2	46.3 MB
kendryte-toolchain-osx-mojave-8.2.0-20190409.tar.xz	21.4 MB
kendryte-toolchain-ubuntu-amd64-8.2.0-20190409.tar.bz2	45.6 MB
kendryte-toolchain-ubuntu-amd64-8.2.0-20190409.tar.xz	19.2 MB
kendryte-toolchain-win-amd64-8.2.0-20190409.tar.xz	18.2 MB
kendryte-toolchain-win-amd64-8.2.0-20190409.zip	51.4 MB
kendryte-toolchain-win-i386-8.2.0-20190409.tar.xz	16.8 MB
kendryte-toolchain-win-i386-8.2.0-20190409.zip	48.6 MB
Source code (zip)	
Source code (tar.gz)	

下载完成后，解压到任意目录，本示例是放到 D 盘根目录下。

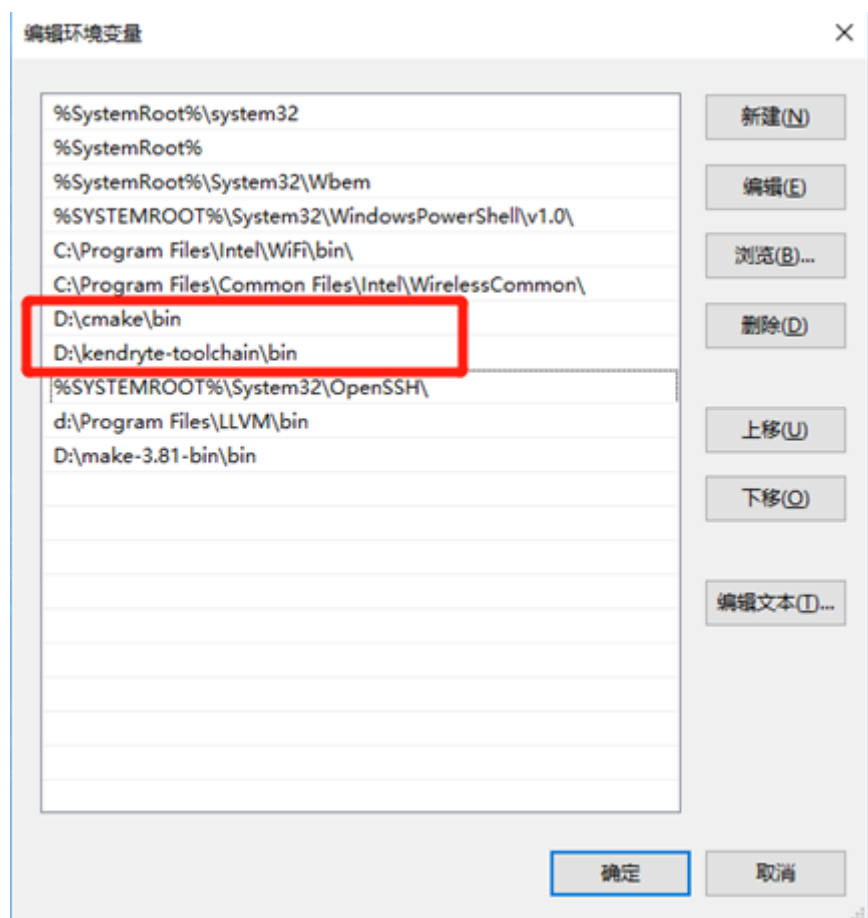
共享 查看

D:\kendryte-toolchain\bin

搜索"bin"

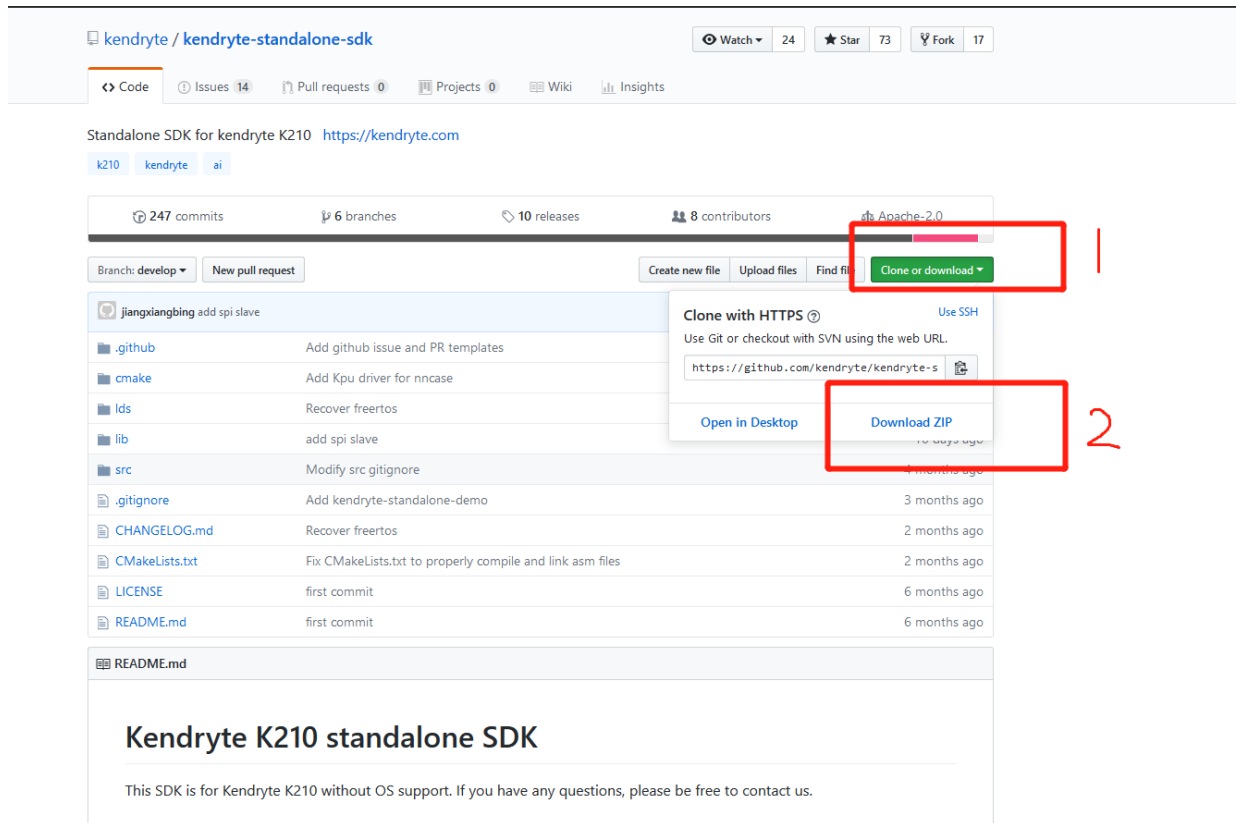
名称	修改日期	类型	大小
make.exe	2019/4/11 17:10	应用程序	214 KB
riscv64-unknown-elf-addr2line.exe	2019/4/11 17:10	应用程序	775 KB
riscv64-unknown-elf-ar.exe	2019/4/11 17:10	应用程序	797 KB
riscv64-unknown-elf-as.exe	2019/4/11 17:10	应用程序	1,048 KB
riscv64-unknown-elf-c++.exe	2019/4/11 17:10	应用程序	1,000 KB
riscv64-unknown-elf-c++filt.exe	2019/4/11 17:10	应用程序	772 KB
riscv64-unknown-elf-cpp.exe	2019/4/11 17:10	应用程序	998 KB
riscv64-unknown-elf-elfedit.exe	2019/4/11 17:10	应用程序	62 KB
riscv64-unknown-elf-g++.exe	2019/4/11 17:10	应用程序	1,000 KB
riscv64-unknown-elf-gcc.exe	2019/4/11 17:10	应用程序	997 KB
riscv64-unknown-elf-gcc-8.2.0.exe	2019/4/11 17:10	应用程序	997 KB
riscv64-unknown-elf-gcc-ar.exe	2019/4/11 17:10	应用程序	58 KB
riscv64-unknown-elf-gcc-nm.exe	2019/4/11 17:10	应用程序	58 KB
riscv64-unknown-elf-gcc-ranlib.exe	2019/4/11 17:10	应用程序	58 KB
riscv64-unknown-elf-gcov.exe	2019/4/11 17:10	应用程序	683 KB
riscv64-unknown-elf-gcov-dump.exe	2019/4/11 17:10	应用程序	561 KB
riscv64-unknown-elf-gcov-tool.exe	2019/4/11 17:10	应用程序	613 KB
riscv64-unknown-elf-gdb.exe	2019/4/11 17:10	应用程序	5,322 KB
riscv64-unknown-elf-gdb-add-index...	2019/4/11 15:34	应用程序	4 KB
riscv64-unknown-elf-gprof.exe	2019/4/11 17:10	应用程序	837 KB
riscv64-unknown-elf-ld.bfd.exe	2019/4/11 17:10	应用程序	1,168 KB
riscv64-unknown-elf-ld.exe	2019/4/11 17:10	应用程序	1,168 KB
riscv64-unknown-elf-nm.exe	2019/4/11 17:10	应用程序	786 KB

4.4 设置环境变量



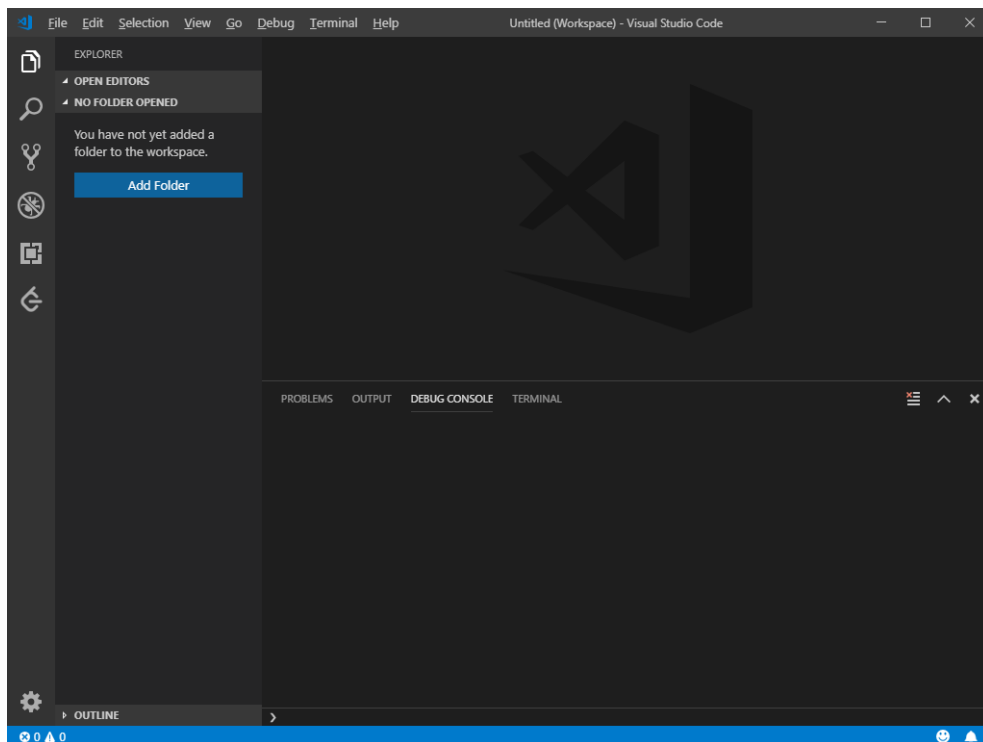
4.5 下载 K210 的最新 SDK

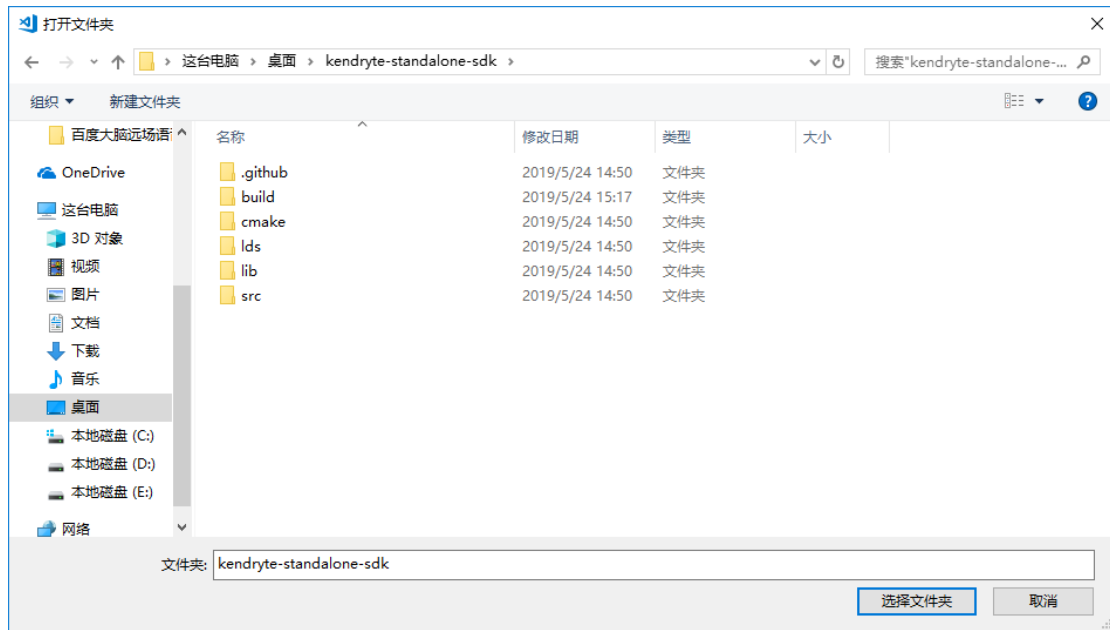
下载地址: <https://github.com/kendryte/kendryte-standalone-sdk>



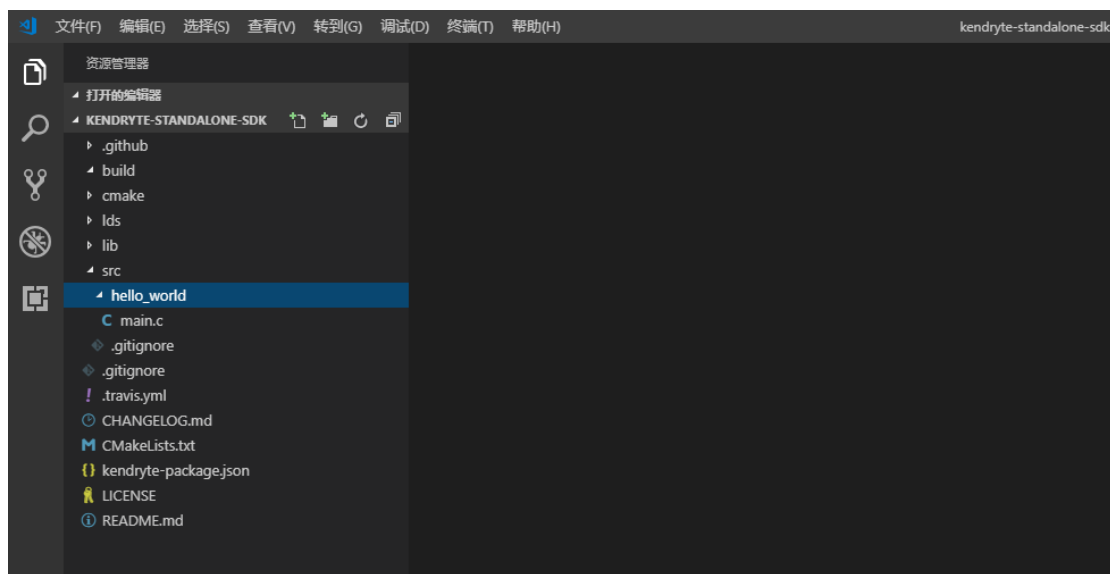
4.6 VSCODE 下编译 SDK

打开 VSCode, 点击左边窗口的 “Add Folder”添加 kendryte-standalone-sdk。如下图:



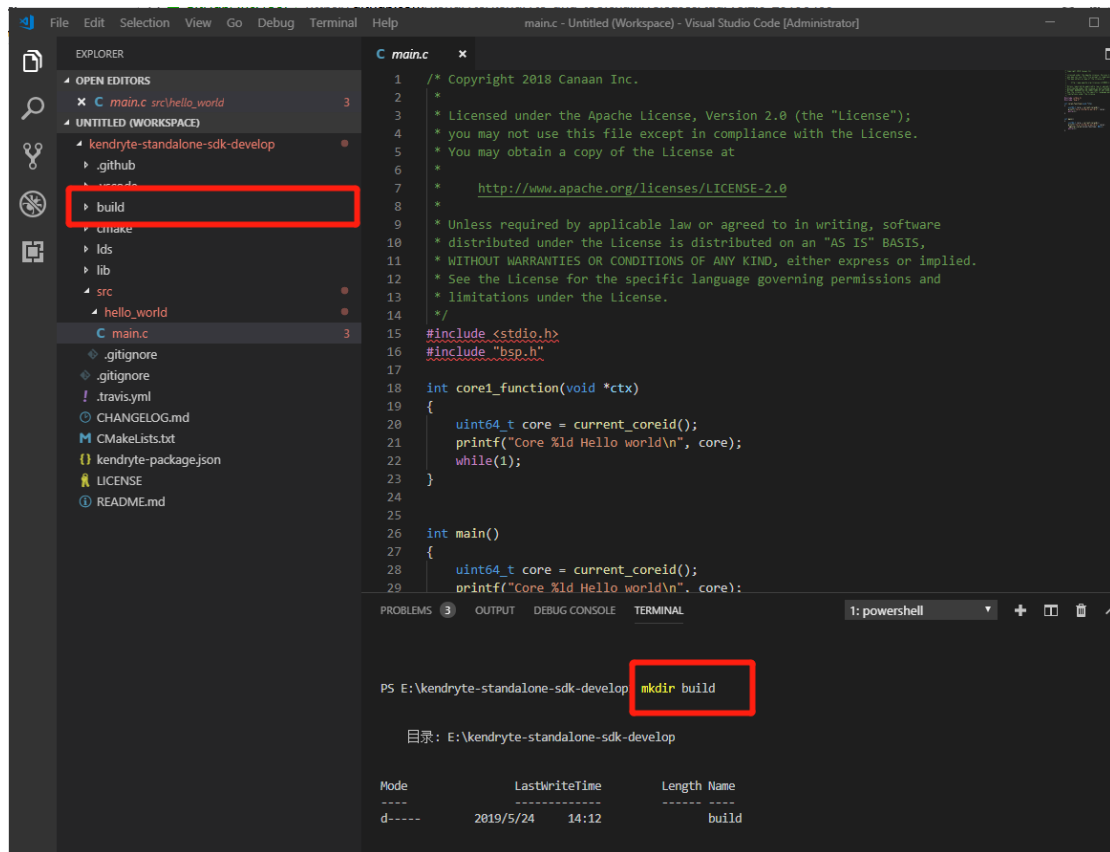


成功添加工程后的界面：

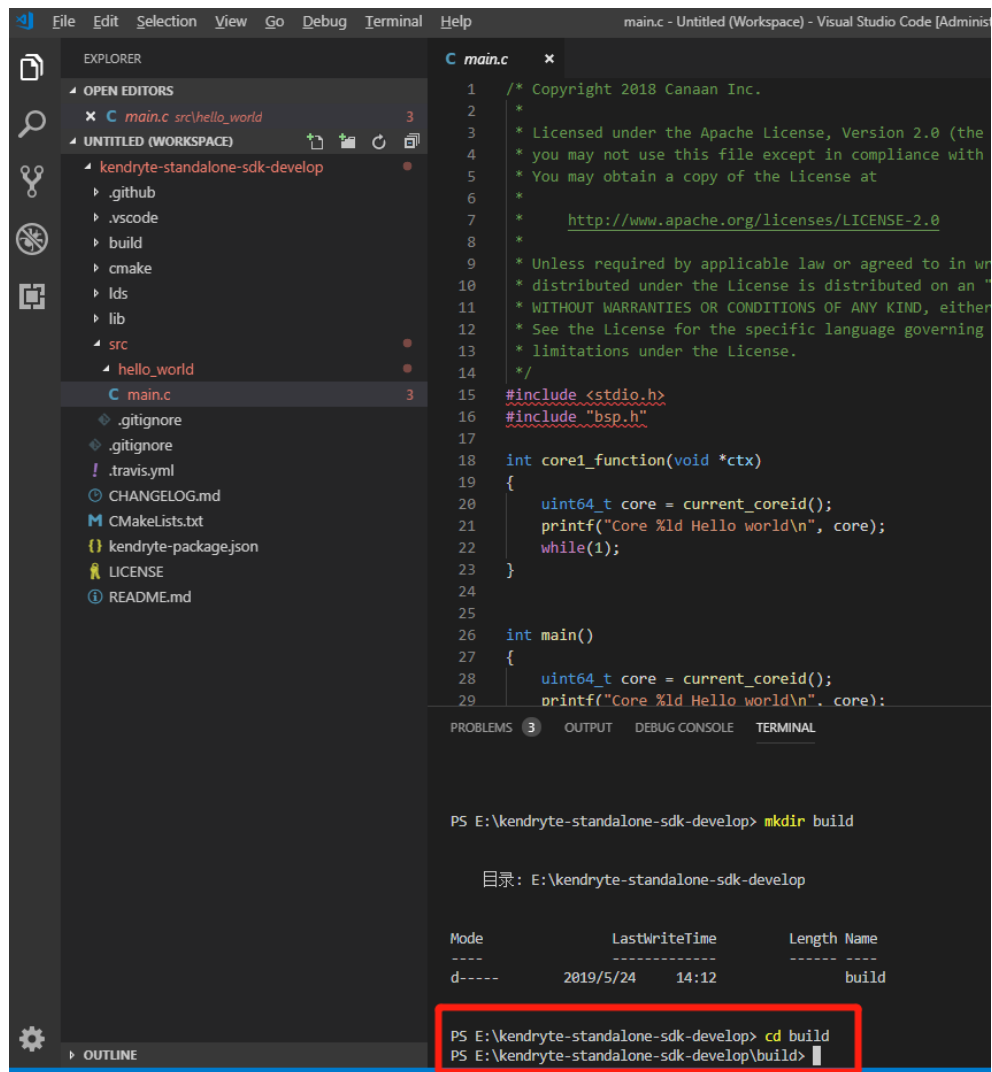


按键盘 `ctrl+shift + ~`，下方出现命令终端。

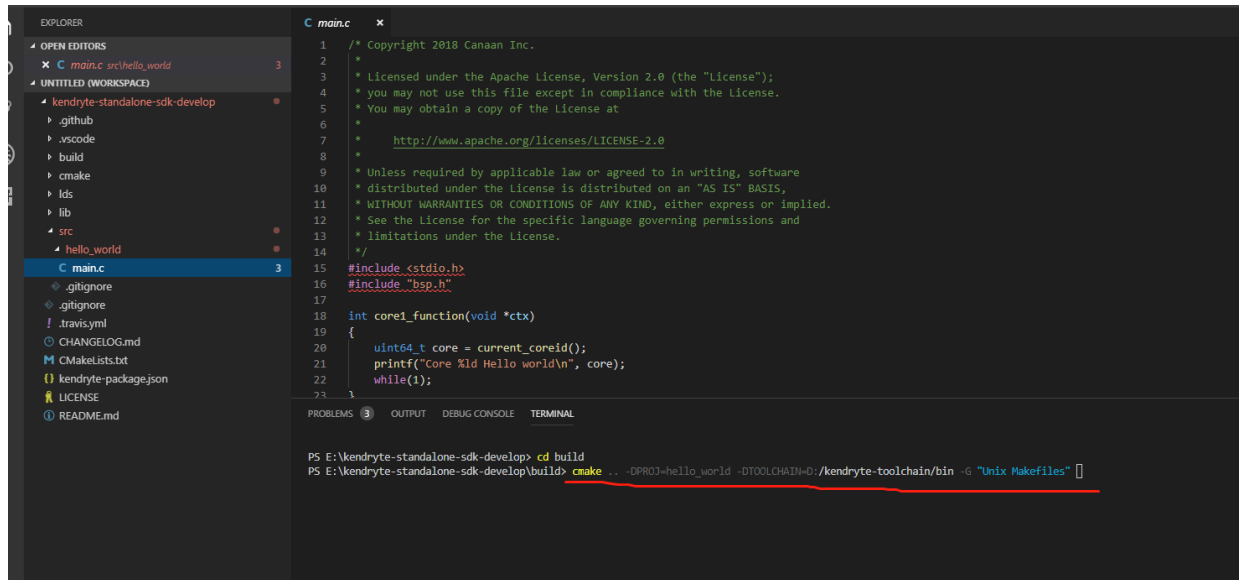
输入命令 `mkdir build`，回车，创建 `build` 文件夹。



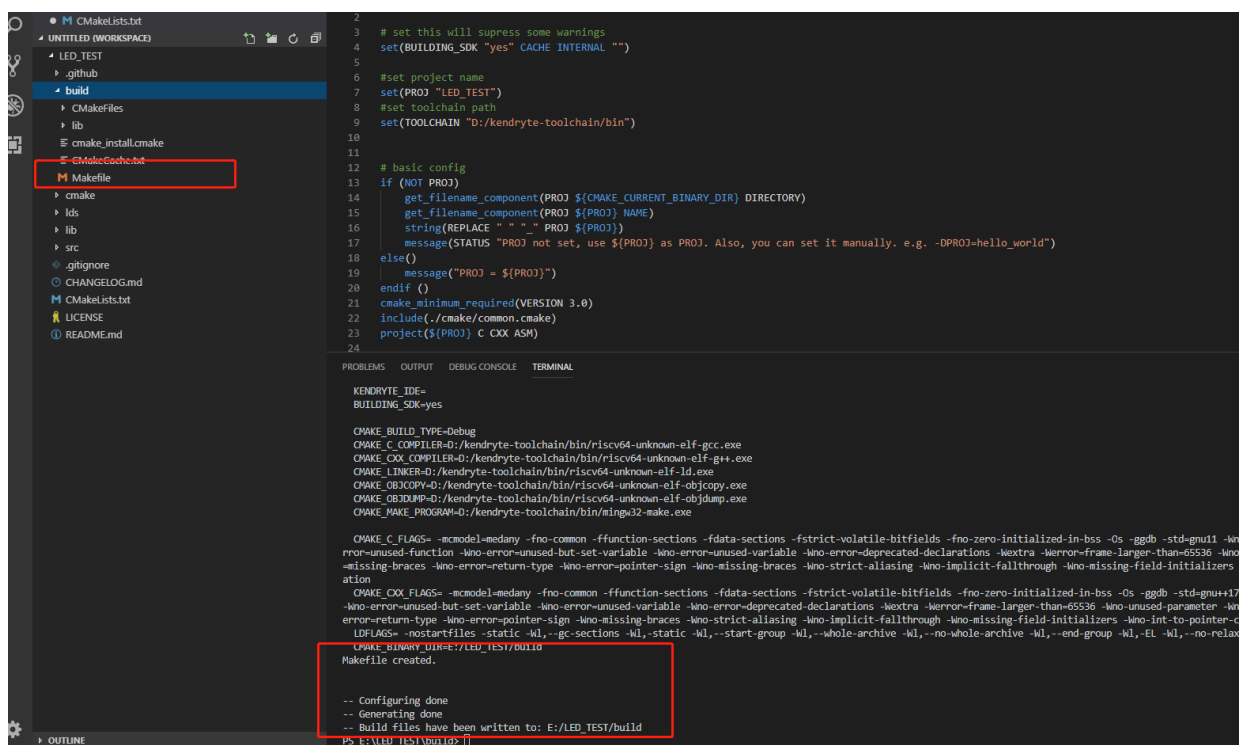
cd build 回车，进入 build 文件夹内。



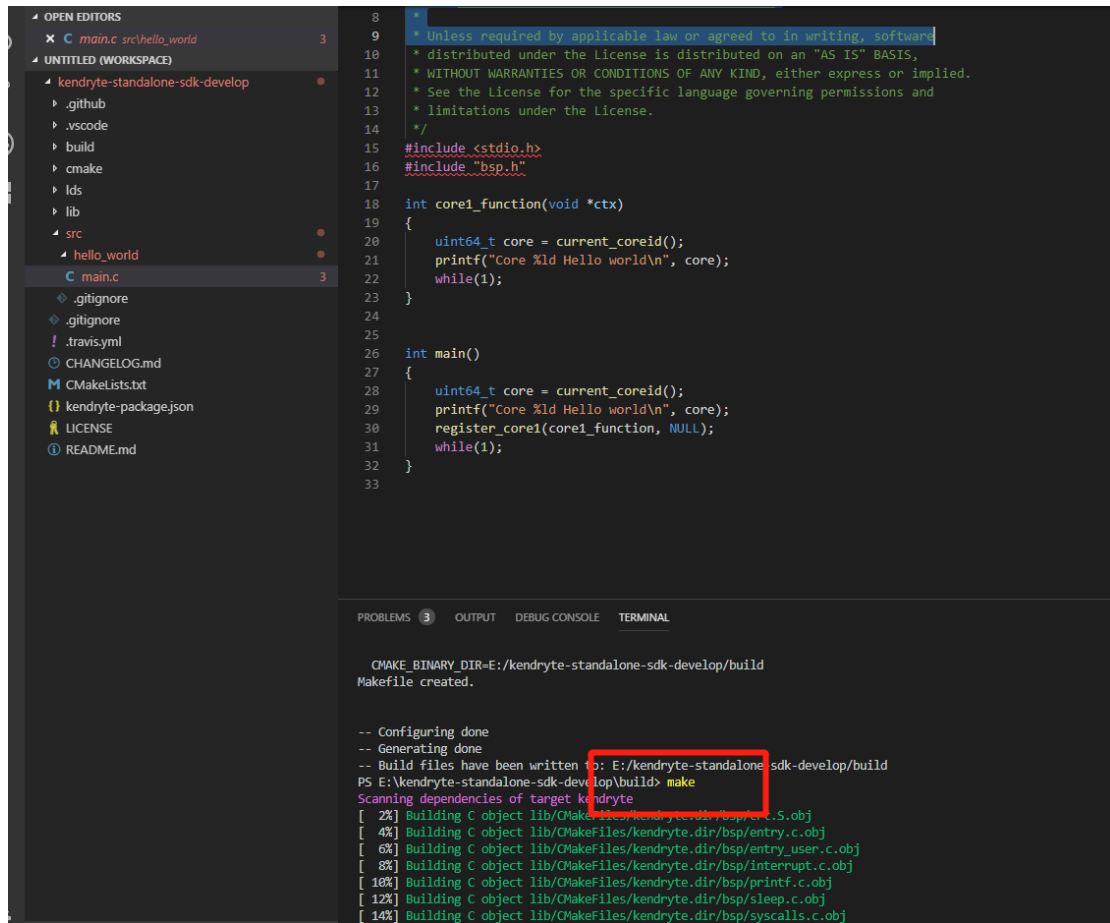
输入 `cmake .. -DPROJ=hello_world -G "Unix Makefiles"` 回车。



此时，在根目录下 `build` 文件中，`makefile` 文件已创建。



继续输入命令 `make`，然后回车。



The screenshot shows a VS Code editor with a project named 'kendryte-standalone-sdk-develop'. The file explorer on the left shows the project structure, including a 'src' directory with 'hello_world' and 'main.c'. The main editor displays the contents of 'main.c', which includes a license header, standard library includes, and two functions: 'core1_function' and 'main'. The 'main' function calls 'current_coreid()', 'printf()', 'register_core1()', and 'while(1)'. The terminal at the bottom shows the output of the 'make' command, indicating that the Makefile has been created and the build process is underway. The terminal output includes the path 'E:/kendryte-standalone-sdk-develop/build' and a list of files being built, such as 'lib/CMakeFiles/kendryte.dir/bsp/entry.c.obj'.

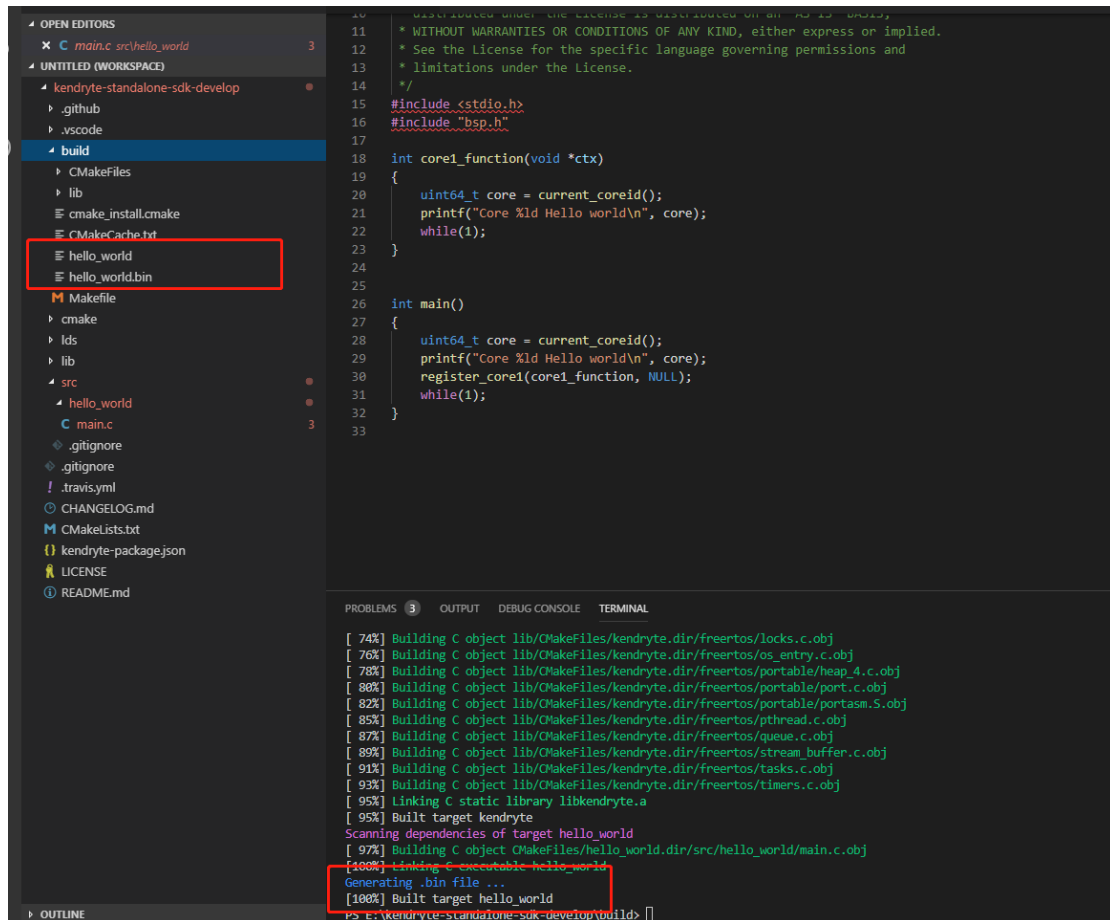
```
8  *
9  * Unless required by applicable law or agreed to in writing, software
10 * distributed under the license is distributed on an "AS IS" BASIS,
11 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 * See the License for the specific language governing permissions and
13 * limitations under the License.
14 */
15 #include <stdio.h>
16 #include "bsp.h"
17
18 int core1_function(void *ctx)
19 {
20     uint64_t core = current_coreid();
21     printf("Core %ld Hello world\n", core);
22     while(1);
23 }
24
25
26 int main()
27 {
28     uint64_t core = current_coreid();
29     printf("Core %ld Hello world\n", core);
30     register_core1(core1_function, NULL);
31     while(1);
32 }
33
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL

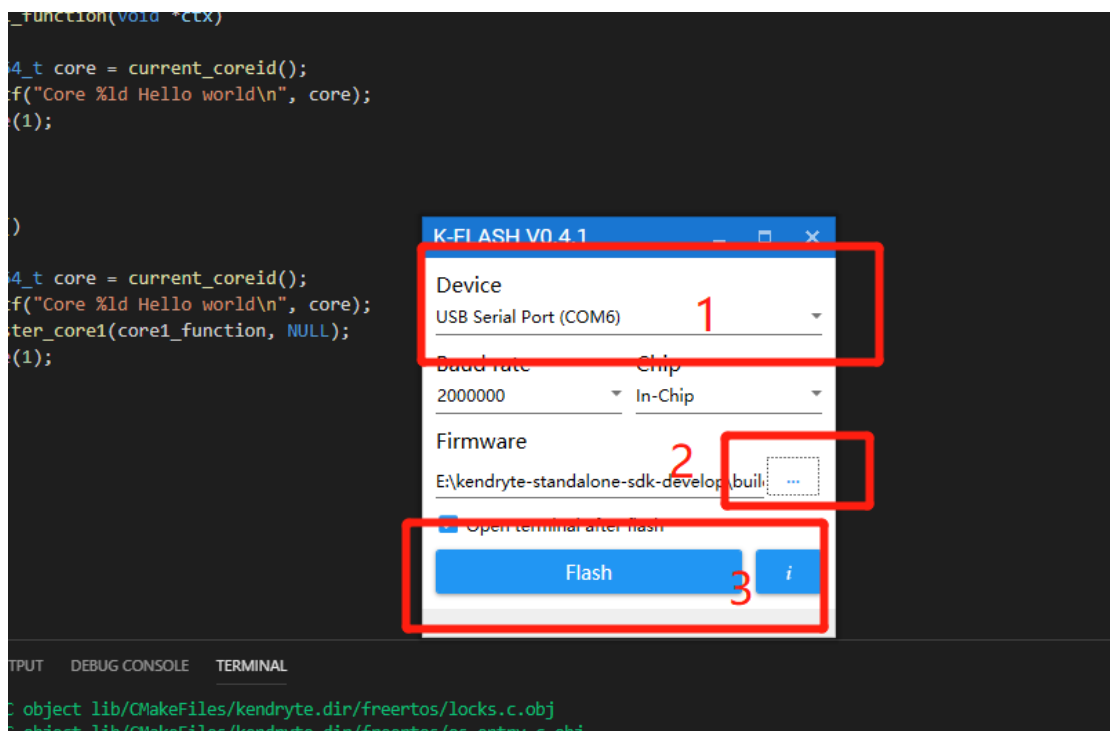
CMAKE_BINARY_DIR=E:/kendryte-standalone-sdk-develop/build
Makefile created.

-- Configuring done
-- Generating done
-- Build files have been written to: E:/kendryte-standalone-sdk-develop/build
PS E:/kendryte-standalone-sdk-develop/build> make
Scanning dependencies of target kendryte
[2%] Building C object lib/CMakeFiles/kendryte.dir/bsp/entry.c.obj
[4%] Building C object lib/CMakeFiles/kendryte.dir/bsp/entry_user.c.obj
[6%] Building C object lib/CMakeFiles/kendryte.dir/bsp/interrupt.c.obj
[8%] Building C object lib/CMakeFiles/kendryte.dir/bsp/printf.c.obj
[10%] Building C object lib/CMakeFiles/kendryte.dir/bsp/sleep.c.obj
[12%] Building C object lib/CMakeFiles/kendryte.dir/bsp/syscalls.c.obj
[14%] Building C object lib/CMakeFiles/kendryte.dir/bsp/syscalls.c.obj

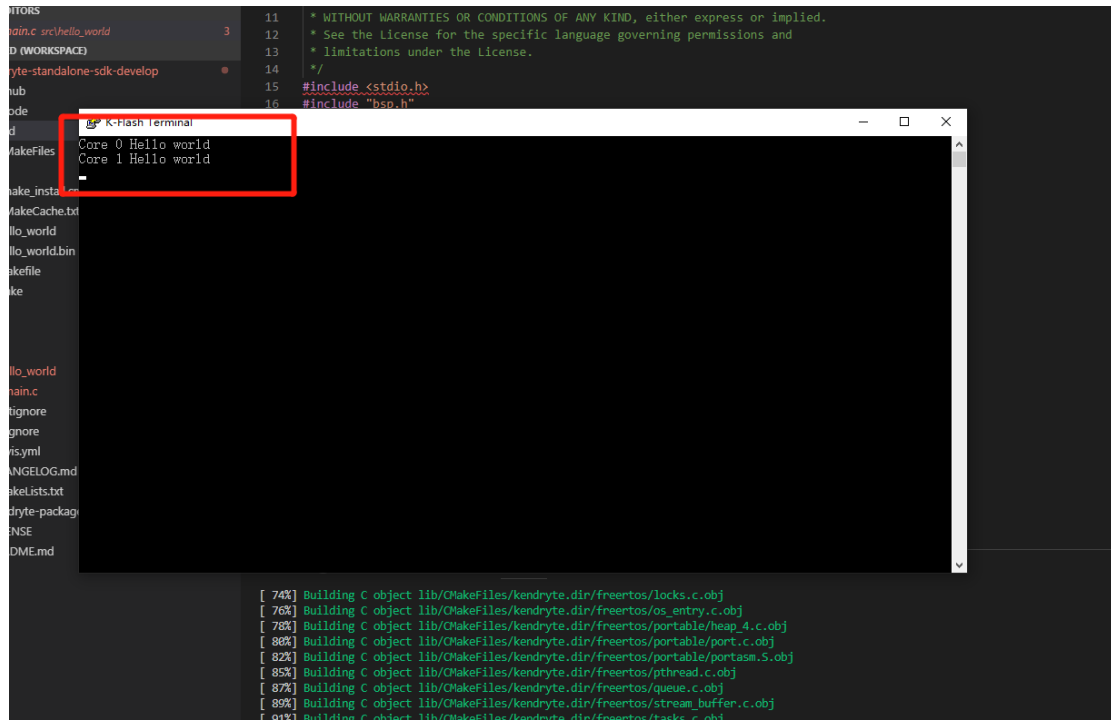
完成编译后，将生成 `hello_world.bin`，`hello_world.bin` 为可直接烧写的文件，如下图：



打开 K-Flash 工具，加载刚才生成的 bin 文件,点击“Flash”等待完成即可。



下载完成后，自动弹出的命令行窗口会直接打印“Core 0 hello_wolrd”。



The screenshot shows a development environment with a source code editor on the left and a terminal window on the right. The source code editor displays a C file named `hello_world.c` with the following content:

```
11 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
12 * See the License for the specific language governing permissions and  
13 * limitations under the License.  
14 */  
15 #include <stdio.h>  
16 #include "bsp.h"
```

The terminal window, titled "K-HASH terminal", shows the output of the program:

```
Core 0 Hello world  
Core 1 Hello world
```

At the bottom of the screen, a progress bar indicates the build status of various objects:

```
[ 74%] Building C object lib/MakeFiles/kendryte.dir/freertos/locks.c.obj  
[ 76%] Building C object lib/MakeFiles/kendryte.dir/freertos/os_entry.c.obj  
[ 78%] Building C object lib/MakeFiles/kendryte.dir/freertos/portable/heap_4.c.obj  
[ 80%] Building C object lib/MakeFiles/kendryte.dir/freertos/portable/port.c.obj  
[ 82%] Building C object lib/MakeFiles/kendryte.dir/freertos/portable/portasm.S.obj  
[ 85%] Building C object lib/MakeFiles/kendryte.dir/freertos/pthread.c.obj  
[ 87%] Building C object lib/MakeFiles/kendryte.dir/freertos/queue.c.obj  
[ 89%] Building C object lib/MakeFiles/kendryte.dir/freertos/stream_buffer.c.obj  
[ 91%] Building C object lib/MakeFiles/kendryte.dir/freertos/tasks.c.obj
```