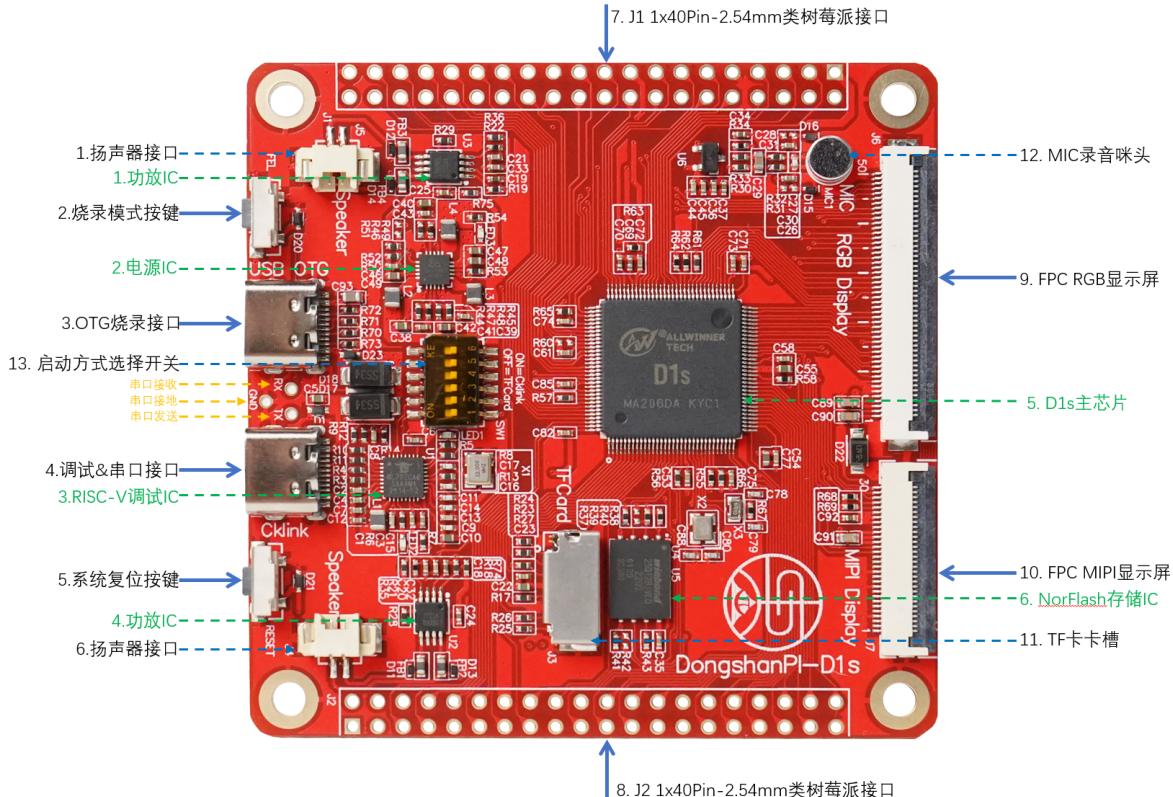


riscv_programming_practice for DongshanPi-D1s

实验环境：

- q 主机硬件平台：DongshanPi-D1s开发板主板
- q 主机操作系统：Windows10/ubuntu
- q GCC版本：9 (riscv64-linux-gnu-gcc)
- q GDB版本：gdb-multiarch。



配置windows下开发环境

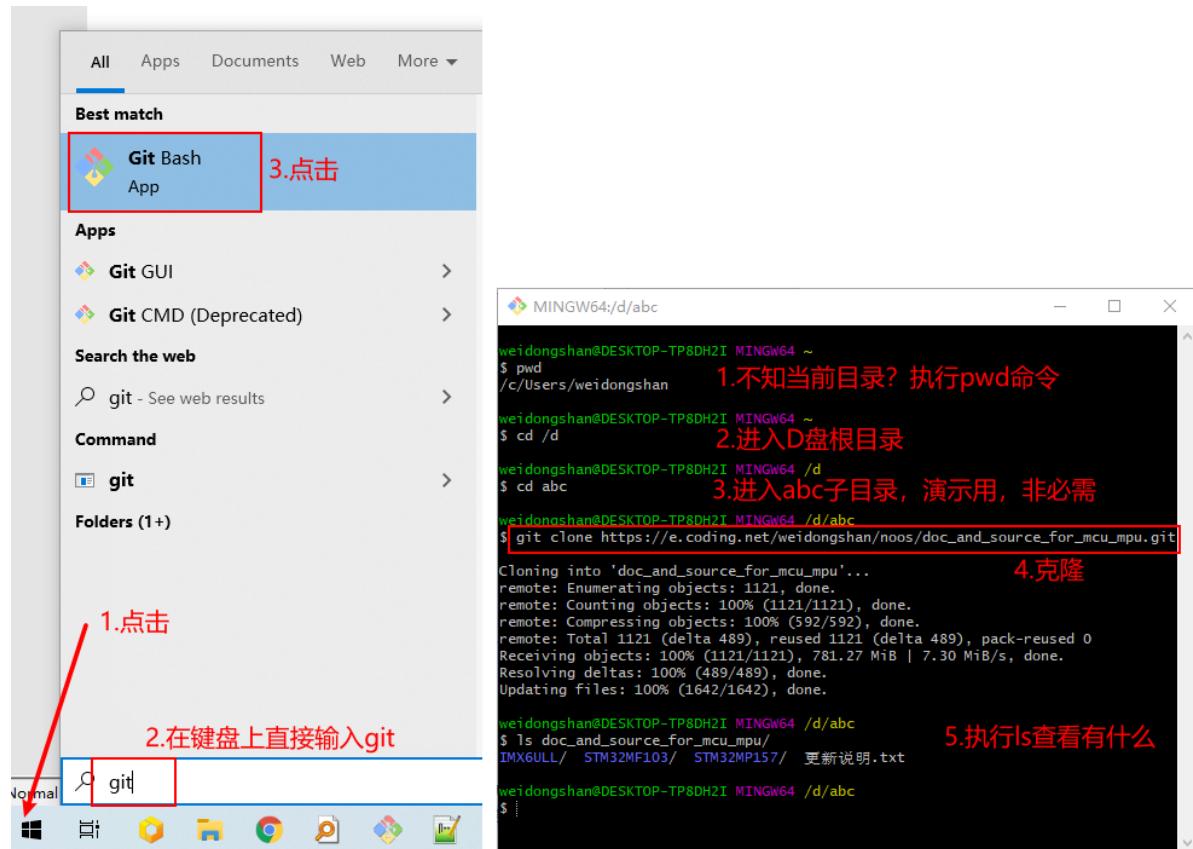
安装Git Bash

进入 05_开发配套工具 先安装 `Git-2.39.1-64-bit.exe`

在Windows下，Git名为msysGit，从 <https://gitforwindows.org> 上下载安装文件，双击安装即可，安装选项很多，使用默认选项即可。如果下载慢，可以在百度上搜：Git-2.28.0-64-bit.exe，自行下载。

对于Windows或Linux，它们的命令行用法相似，对于Windows，进入Git命令行的方法是在 `开始->所有程序->Git` 下启动 Git Bash。Git Bash的命令用法跟Linux完全一样，比如`cd`、`ls`等命令。

安装完成后可以参考下图来打开 git bash工具



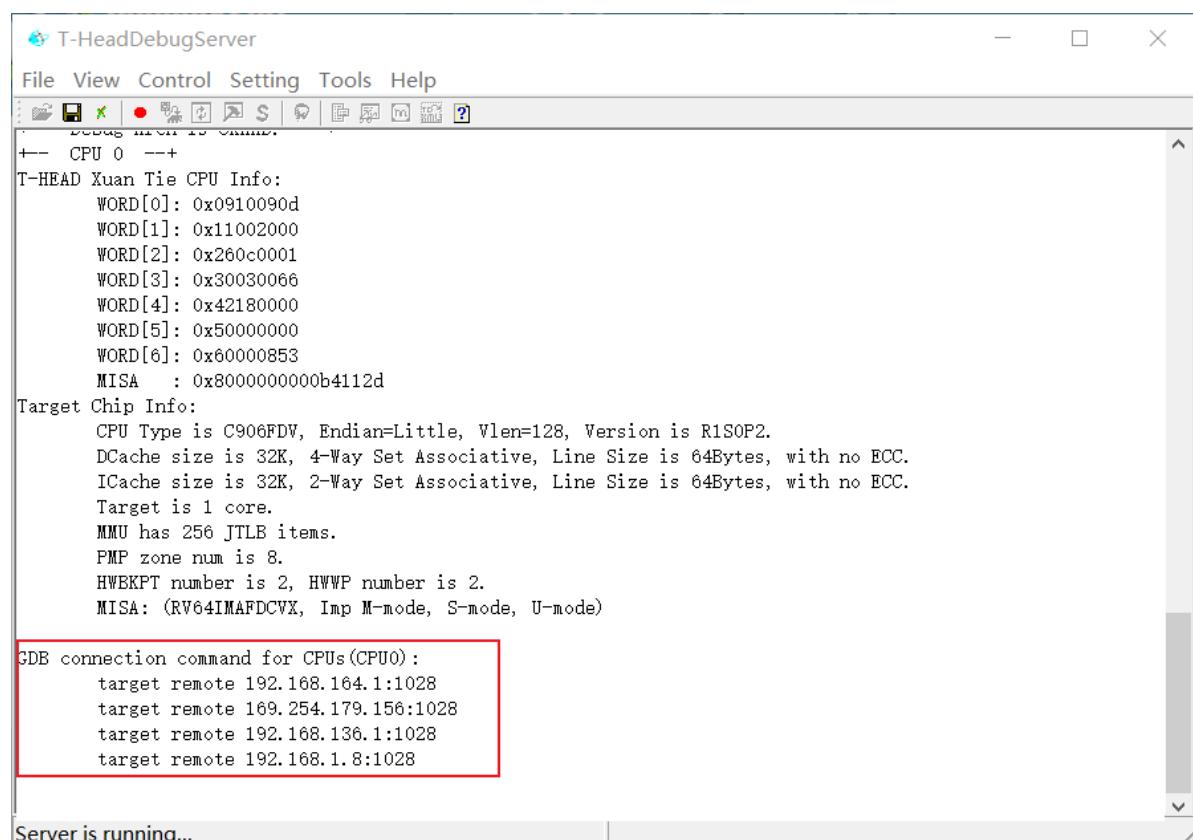
安装Cklink驱动软件

进入 05_开发配套工具\CKLinkServer 安装windows下 T-Head-DebugServer-windows-v5.16.6-20221102-1510.zip 直接进行解压缩安装。

连接开发板另一条TypeC线至开发板 Cklink 接口此时设备管理器会多出来一个 Cklink-lite的设备。

此时我们长按开发板上的 **FEL按键**，同时按一下 **RESET按键** 开发板就会自动进入调试模式。

这时打开 T-HeadDebugServer会自动显示设备的CPU 信息以及GDB端口连接地址。

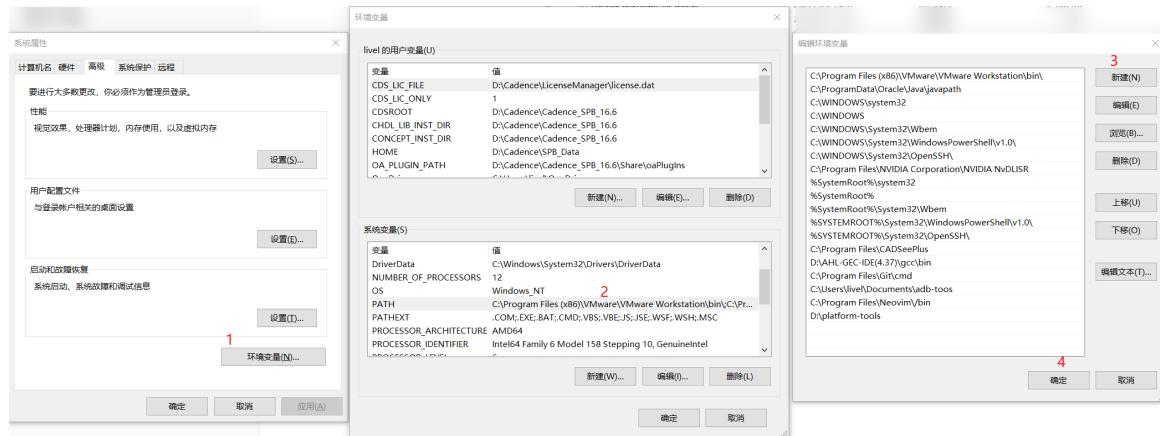


安装交叉编译工具链

进入到 05_开发配套工具\toolchain 目录，解压缩 Xuantie-900-gcc-elf-newlib-mingw-V2.6.1-20220906.tar.gz

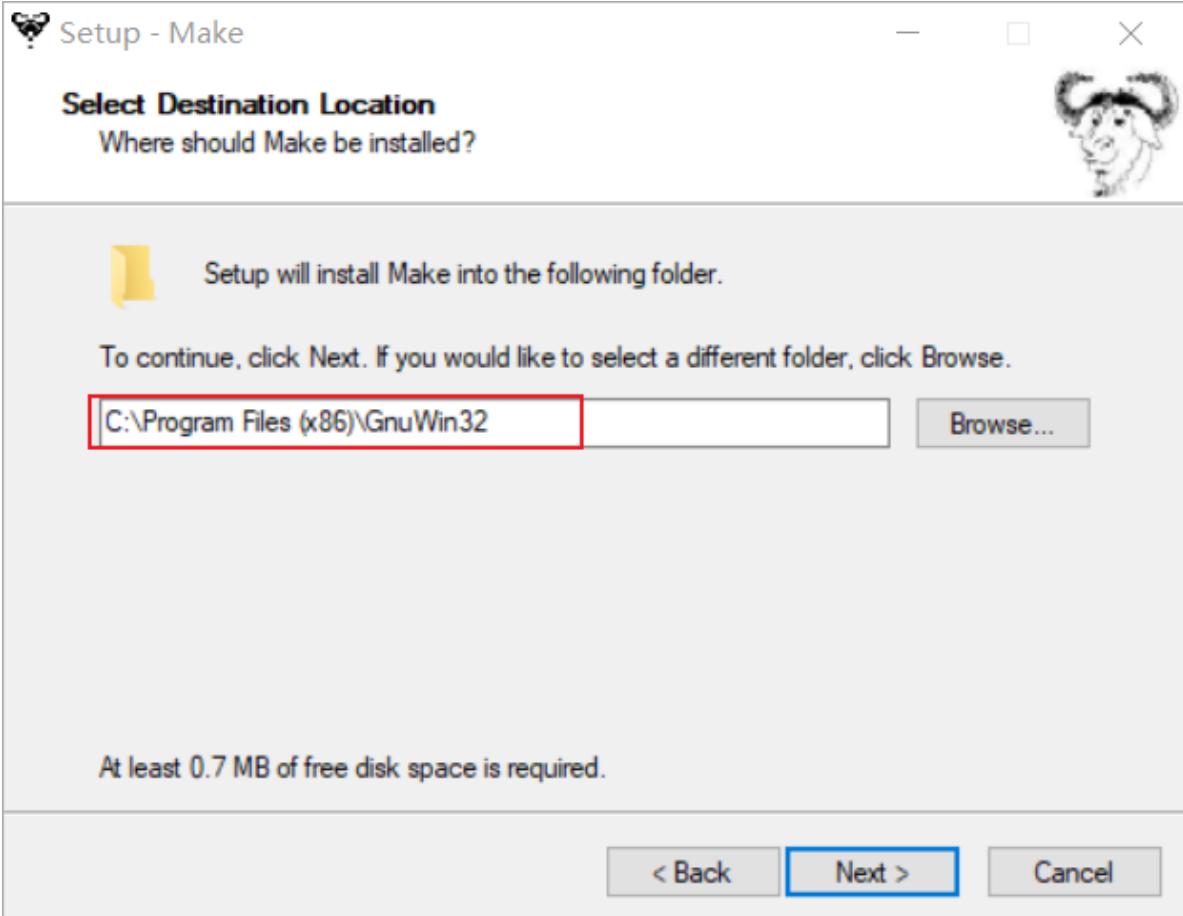
解压缩完成后进入到 如下图红框所示这个目录，记住这个路径的完整目录(尽量不要包含中文) 然后将其添加至系统的 Path环境变量内。

名称	修改日期	类型	大小
riscv64-unknown-elf-addr2line.exe	2022/9/6 14:57	应用程序	1,688 KB
riscv64-unknown-elf-ar.exe	2022/9/6 14:57	应用程序	1,710 KB
riscv64-unknown-elf-as.exe	2022/9/6 14:57	应用程序	2,108 KB
riscv64-unknown-elf-c++.exe	2022/9/6 15:54	应用程序	0 KB
riscv64-unknown-elf-c++filt.exe	2022/9/6 14:57	应用程序	1,685 KB
riscv64-unknown-elf-cpp.exe	2022/9/6 15:54	应用程序	27,417 KB
riscv64-unknown-elf-elfedit.exe	2022/9/6 14:57	应用程序	953 KB
riscv64-unknown-elf-g++.exe	2022/9/6 15:54	应用程序	27,419 KB
riscv64-unknown-elf-gcc.exe	2022/9/6 15:54	应用程序	27,415 KB
riscv64-unknown-elf-gcc-10.2.0.exe	2022/9/6 15:54	应用程序	0 KB
riscv64-unknown-elf-gcc-ar.exe	2022/9/6 15:54	应用程序	55 KB
riscv64-unknown-elf-gcc-nm.exe	2022/9/6 15:54	应用程序	55 KB
riscv64-unknown-elf-gcc-ranlib.exe	2022/9/6 15:54	应用程序	55 KB
riscv64-unknown-elf-gcov.exe	2022/9/6 15:55	应用程序	1,524 KB
riscv64-unknown-elf-gcov-dump.exe	2022/9/6 15:55	应用程序	1,356 KB
riscv64-unknown-elf-gcov-tool.exe	2022/9/6 15:55	应用程序	1,406 KB
riscv64-unknown-elf-gdb.exe	2022/9/6 14:58	应用程序	7,376 KB
riscv64-unknown-elf-gdb-add-index	2022/9/6 14:58	文件	4 KB
riscv64-unknown-elf-gprof.exe	2022/9/6 14:57	应用程序	1,750 KB
riscv64-unknown-elf-ld.bfd.exe	2022/9/6 14:57	应用程序	2,082 KB
riscv64-unknown-elf-ld.exe	2022/9/6 14:57	应用程序	0 KB
riscv64-unknown-elf-lto-dump.exe	2022/9/6 15:55	应用程序	42,677 KB
riscv64-unknown-elf-nm.exe	2022/9/6 14:57	应用程序	1,700 KB

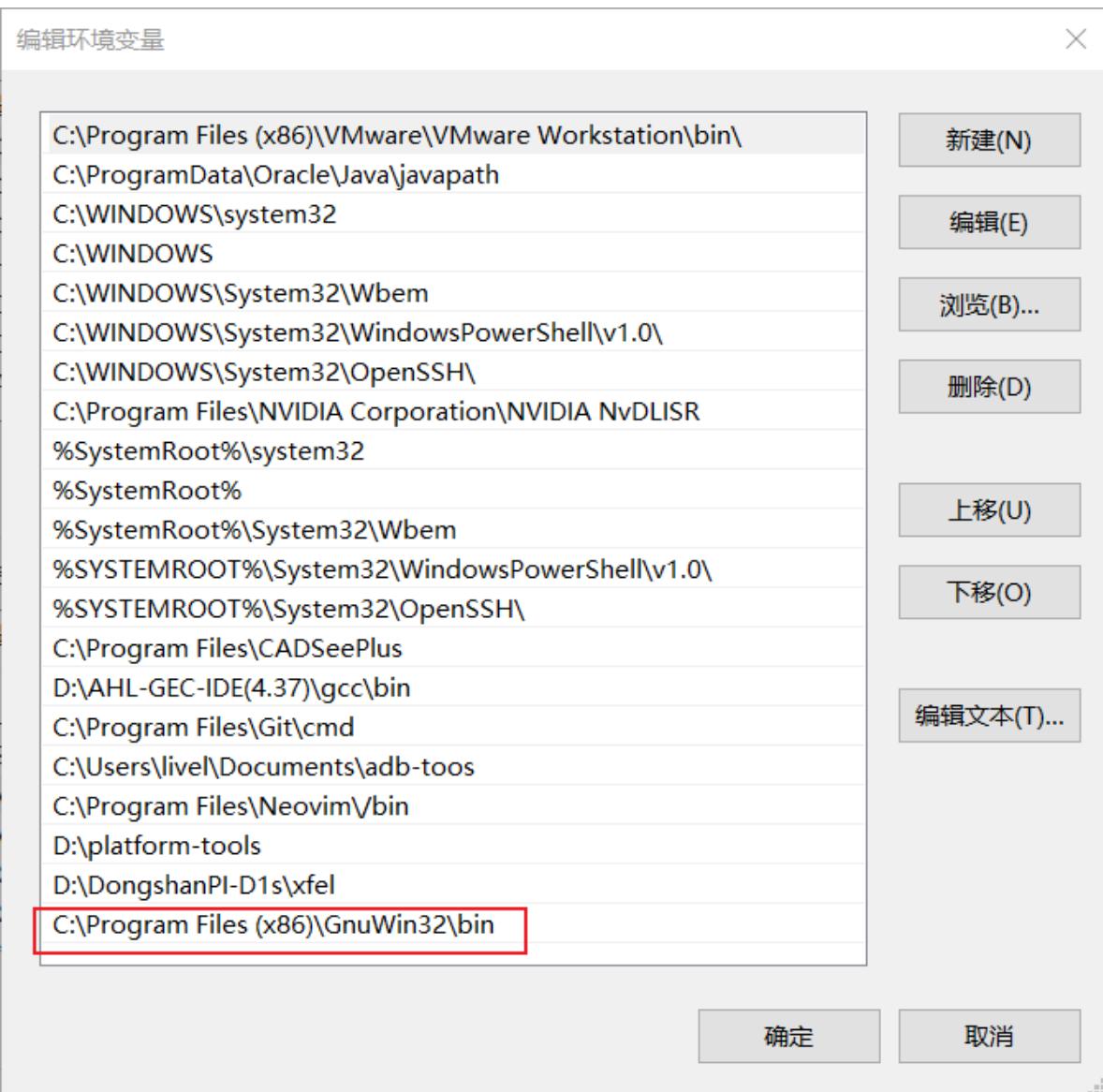


配置make工具

进入 05_开发配套工具\Make 目录内，双击 make-3.81.exe 开始安装，安装时请注意一下安装所在路径，等待安装完成后我们需要将此路径添加至系统环境变量里面。

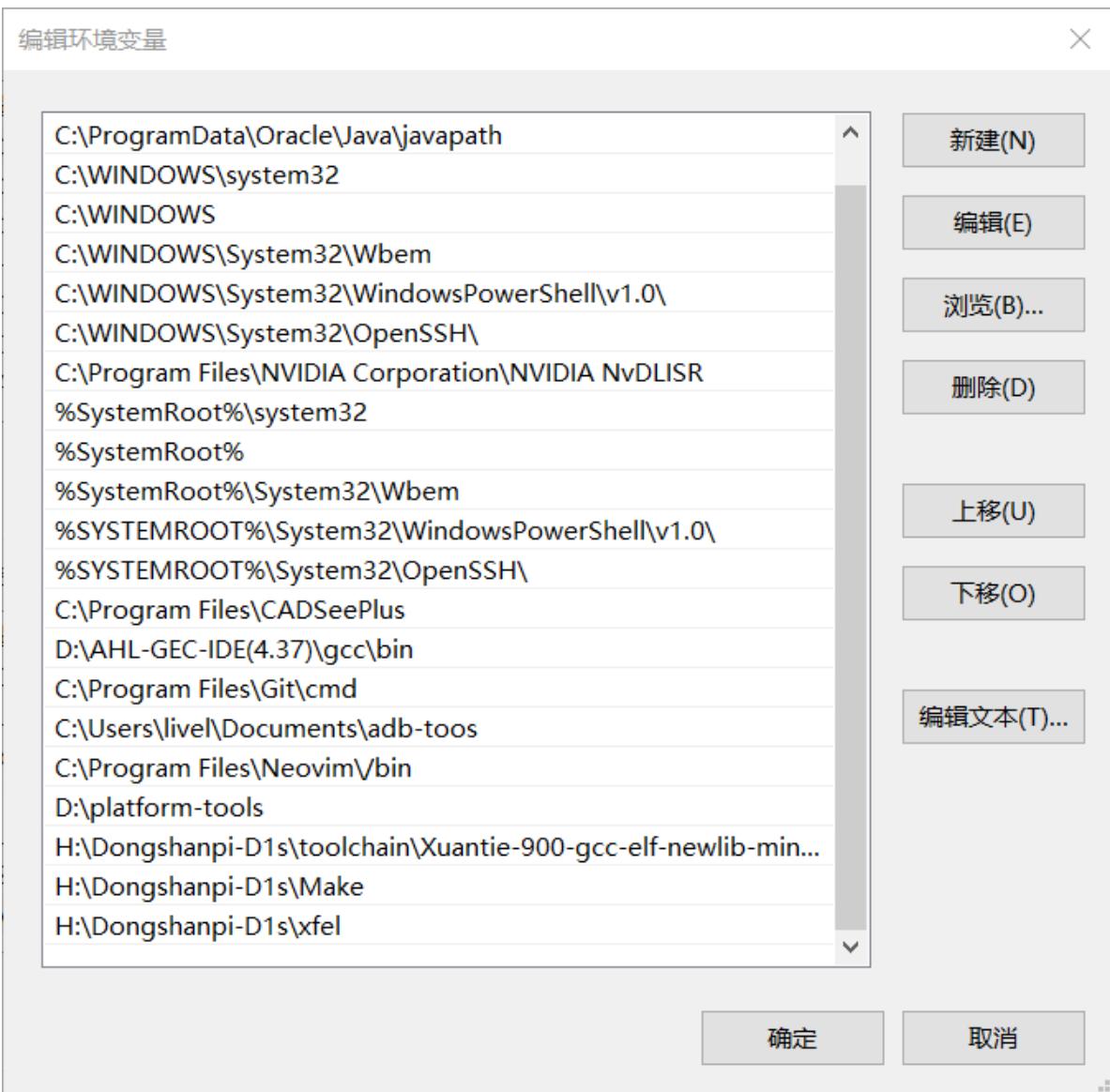


添加完成后如下图所示，之后我们可以通过git bash命令进行验证。



配置Xfel工具

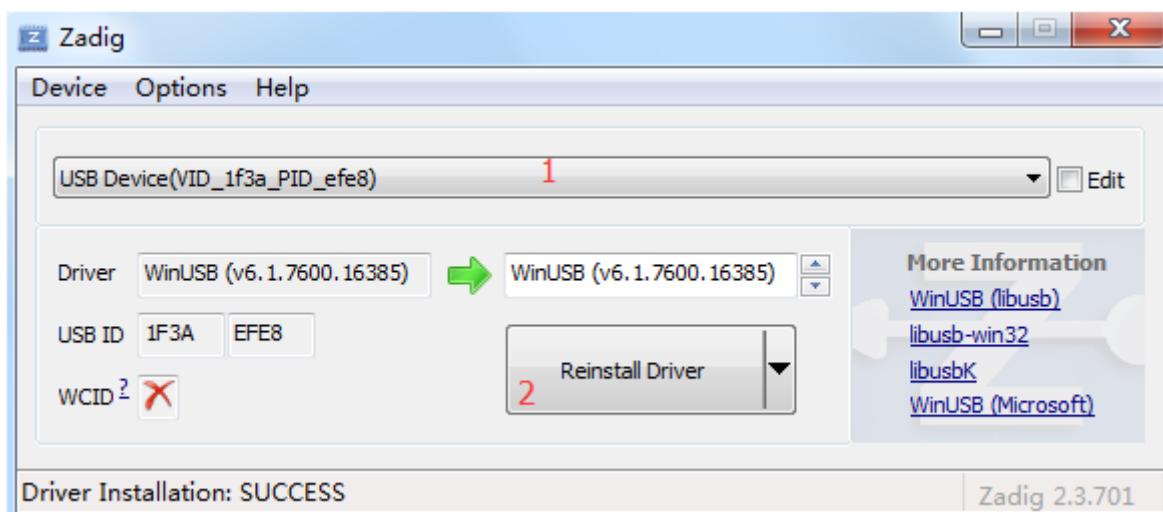
进入 05_开发配套工具\xfel目录内，复制完整的路径，将其单独添加到 windows 环境变量内。



在windows下使用xfel烧录工具之前，我们需要先连接一条 Typec线至开发板OTG接口 连接成功后后，长按 **FEL按键** 此时按一下 **RESET按键** 开发板就会进入烧录模式，如果之前安装过 全志的usb烧写驱动，电脑的设备管理器会出来一个 usb device设备，表明已经进入烧写模式成功。如果你的电脑没有安装驱动，会提示一个 位置设备 <https://dongshanpi.com/DongshanNezhaSTU/03-QuickStart/#usb> 那么请参考这篇文章安装一下驱动。



连接成功后，我们进入到 **05_开发配套工具\xfel\Drivers** 目录下，参考下图安装一下 **WinUSB** 驱动，就可以使用xfel识别到设备了。



验证工具配置

打开git bash界面 来依次验证工具是否正常。

- 验证xfel烧录工具

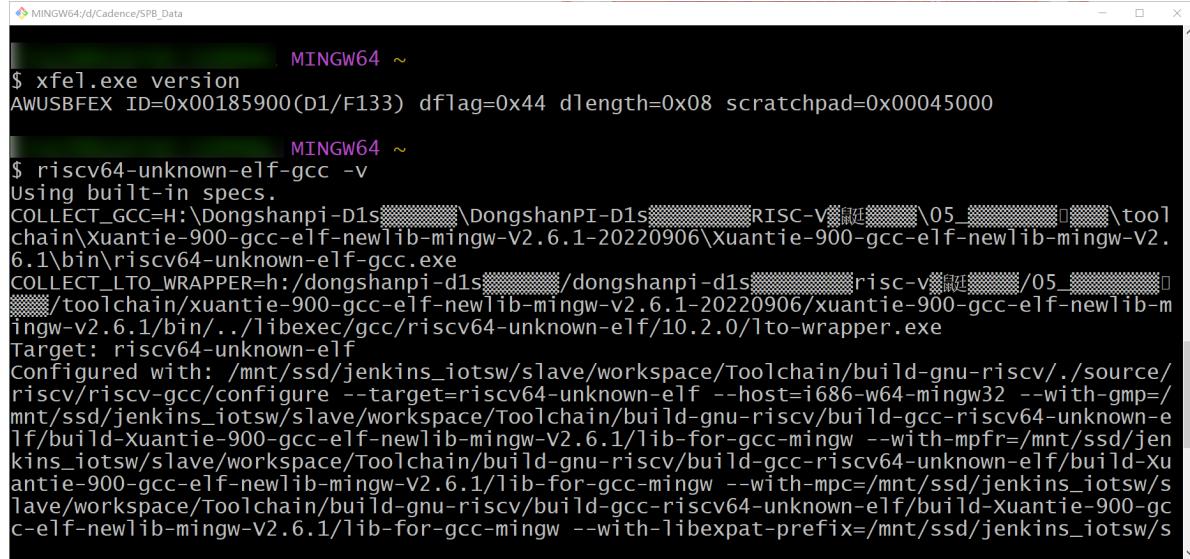
先连接一条 Typec线至开发板OTG接口 连接成功后后，长按 **FEL按键** 此时按一下 **RESET按键** 开发板就会进入烧录模式,此时在 git bash终端里面 输入 `xfel.exe version` 命令就可以识别到芯片的详细信息。

```
MINGW64/d/Cadence/SPB_Data
MINGW64 ~
$ xfel.exe version
AWUSBFEX ID=0x00185900(D1/F133) dfFlag=0x44 dLength=0x08 scratchPad=0x00045000
MINGW64 ~
```

A screenshot of a terminal window titled 'MINGW64/d/Cadence/SPB_Data'. The command `xfel.exe version` was run, and the output shows the AWUSBFEX ID, dfFlag, dLength, and scratchPad values.

- 验证交叉编译工具链

在 git bash 输入 `riscv64-unknown-elf-gcc -v` 既可看到详细的工具链参数输出，表示已经配置成功

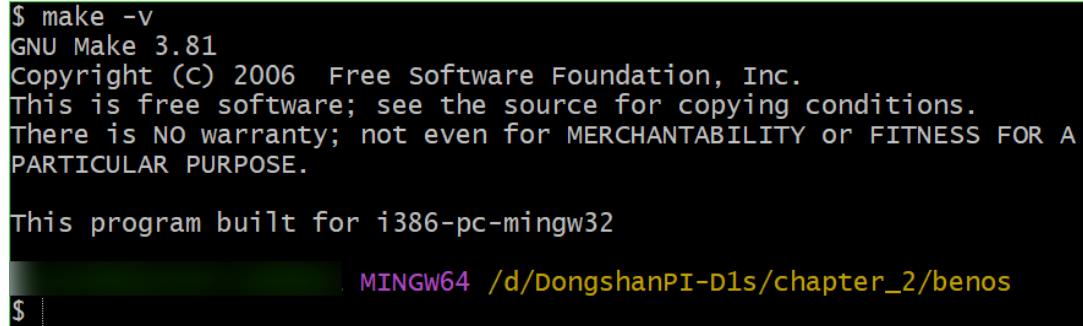


```
MINGW64 ~
$ xfel.exe version
AWUSBFEX ID=0x00185900(D1/F133) dFlag=0x44 dLength=0x08 scratchpad=0x00045000

MINGW64 ~
$ riscv64-unknown-elf-gcc -v
Using built-in specs.
COLLECT_GCC=H:\Dongshanpi-D1s\...\DongshanPI-D1s\...\RISC-V\...\05...\toolchain\Xuantie-900-gcc-elf-newlib-mingw-V2.6.1-20220906\Xuantie-900-gcc-elf-newlib-mingw-V2.6.1\bin\riscv64-unknown-elf-gcc.exe
COLLECT_LTO_WRAPPER=h:/dongshanpi-d1s\...\dongshanpi-d1s\...\risc-v\...\05...\...\toolchain\xuantie-900-gcc-elf-newlib-mingw-v2.6.1-20220906/xuantie-900-gcc-elf-newlib-mingw-v2.6.1/bin/..../libexec/gcc/riscv64-unknown-elf/10.2.0/lto-wrapper.exe
Target: riscv64-unknown-elf
Configured with: /mnt/ssd/jenkins_iotsw/slave/workspace/Toolchain/build-gnu-riscv/.source/riscv/riscv-gcc/configure --target=riscv64-unknown-elf --host=i686-w64-mingw32 --with-gmp=/mnt/ssd/jenkins_iotsw/slave/workspace/Toolchain/build-gnu-riscv/build-gcc-riscv64-unknown-elf/build-xuantie-900-gcc-elf-newlib-mingw-V2.6.1/lib-for-gcc-mingw --with-mpfr=/mnt/ssd/jenkins_iotsw/slave/workspace/Toolchain/build-gnu-riscv/build-gcc-riscv64-unknown-elf/build-Xuantie-900-gcc-elf-newlib-mingw-V2.6.1/lib-for-gcc-mingw --with-mpc=/mnt/ssd/jenkins_iotsw/slave/workspace/Toolchain/build-gnu-riscv/build-gcc-riscv64-unknown-elf/build-Xuantie-900-gcc-elf-newlib-mingw-V2.6.1/lib-for-gcc-mingw --with-libexpat-prefix=/mnt/ssd/jenkins_iotsw/s
```

- 验证 make 命令

在 git bash 输入 `make -v` 命令，如果可以显示出来版本，就表示已经可以使用。



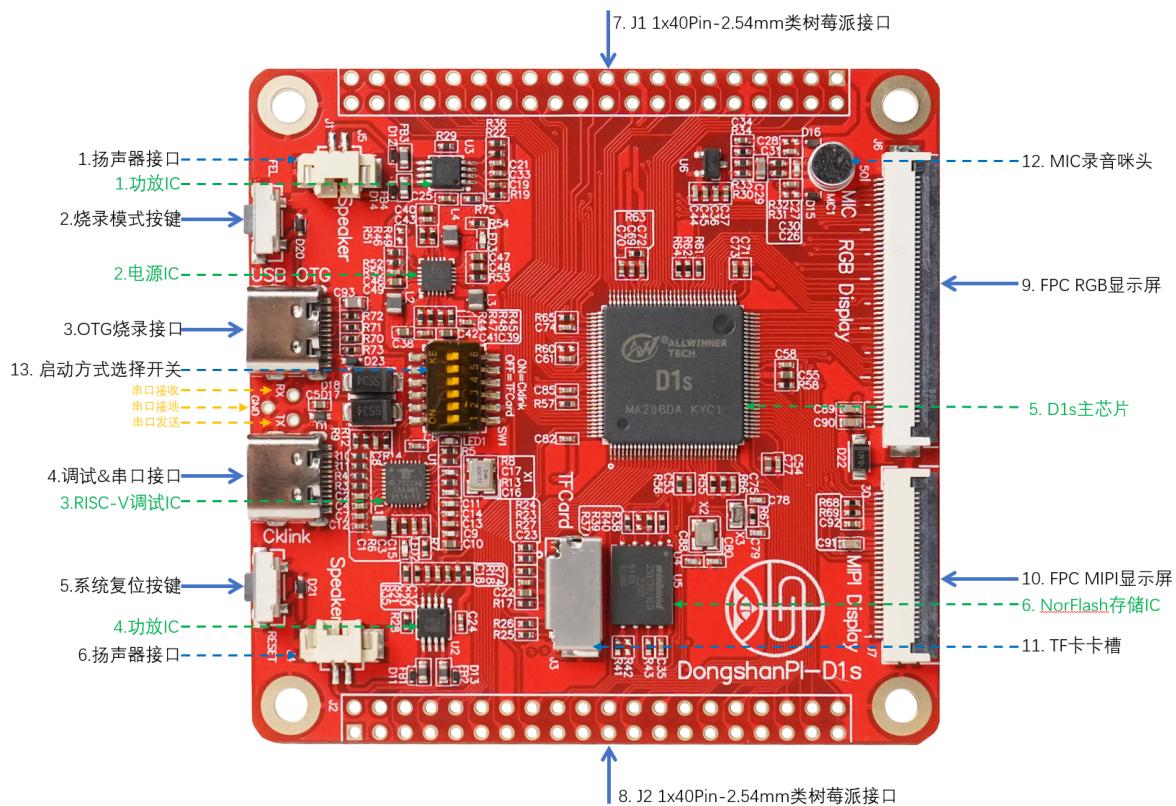
```
$ make -v
GNU Make 3.81
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

This program built for i386-pc-mingw32

MINGW64 /d/DongshanPI-D1s/chapter_2/benos
$
```

编译烧写程序

如下图为开发板的功能示意图，我们需要先将 配套的typec线一根插至 黑色序号3. OTG烧录接口，用于进行供电和烧录系统操作。



首先进入到源码目录 `riscv_programming_practice-for-dongshan\chapter_2\benos` 在当前目录下，鼠标右键，点击 `git bash Here`



在源码目录下执行 `make clean` 清理缓存，之后执行 `make` 命令开始编译，编译完成后，可以执行 `make burn` 来自动烧录镜像到开发板内，可以看到如下图所示。

注意：烧录开发板之前需要先将两根 typec 线全部连接至开发板上，然后长按 **FEL按键** 短按一下 **RESET 键** 进入烧录模式，之后就可以执行上述烧录步骤了，烧录完成后，可以使用串口工具 打开 Cklinke自带的 串口设备，波特率 115200 就可以看到程序的输出信息。

```

$ make
rm -rf build_src build_sbt *.bin *.map *.elf *.dis
CC build_src/kernel_c.o
CC build_src/sys-clock_c.o
CC build_src/uart_c.o
AS build_src/boot_s.o
LD build_src/benos.elf
OBJCOPY benos.bin
OBJDUMP benos.dis
CC build_sbt/sbt_main_c.o
CC build_sbt/sbt_main_c.o
AS build_sbt/sbt_boot_s.o
AS build_sbt/sbt_payload_s.o
LD build_sbt/mybenos.elf
OBJCOPY mybenos.bin
OBJDUMP mybenos.dis
LD build_sbt/benos_payload.elf
OBJCOPY benos_payload.bin
MKUNIXI benos_payload.bin
OBJDUMP benos_payload.dis

$ ls /d/DongshanPI-D1s/chapter_2/benos
' CC build_src' -fomit-frame-pointer -o/      benos.map      echo/      mybenos.map
-DCONFIG_BOARD_QEMU -g/           include/      benos.payload.bin    include/      mybenos.map
-Iinclude/          -mabi/        lib64/       benos.payload.dis  lib64/       riscv64-unknown-elf-gcc/
-0/                -march/      Makefile     benos_payload.elf   medany/      rv64imafd/
-wall/             -mcmode1/    benos.bin    benos_payload.map  mybenos.bin  sbt/
-c/                -nostdinc/  benos.dis    build_sbt/      mybenos.dis  src/
-fno-PIE/          -nostdlib/  benos.elf#  build_src/      mybenos.elf#  tools/
$ make burn
xfel spinor write 0 benos_payload.bin
100% [=====] 24.000 KB, 181.203 KB/s

```

使用GDB 调试