Because image has too much dependency on it, there will be something to modify for preparing the image. The following is the instruction to split them.

Output/input	Folder/tool files	Contains/dependey files in nvm	original path	size
Input	**FERMION_NVM_PROGRAMMER.elf** nvm_programmer.py fw_upgrade\: gen*.py gdb_framework\: gdb*.py,hexfile.py	.\FERMION_NVM_PROGRAMMER.elf .\nvm_programmer.py .\fw_upgrade\ .\gdb_framework\	qccsdk\tools\nvm_programmer\bin\ qccsdk\tools\nvm_programmer\scripts\ qccsdk\tools\fw_upgrade\ qccsdk\tools\nvm_programmer\scripts\	236KB 92.2KB 56.5KB
	openocd.exe libhidapi-0.dll libusb-1.0.dll qcc730_openocd_ch347.cfg	.\openocd.exe .\libhidapi-0.dll .\libusb-1.0.dll .\qcc730_openocd_ch347.cfg	{system install path}\OpenOCD_CH347\bin: openocd.exe, libhidapi-0.dll, libusb-1.0.dll Example: C:\work\tools\OpenOCD_CH347\OpenOCD_CH347\bin qccsdk/tools/nvm_programmer/scripts/qcc730_openocd_ch347.cfg	5330KB
	arm-none-eabi-gdb-py3.exe DLLs*、libgcc_s_seh-1.dll、libgmp-10.dll、libiconv-2.dll、libmpc-3.dll、libmpfr-4.dll、libstdc++-6.dll、libwinpthread-1.dll、libzstd.dll、python311.dll、python311.zipped arm-none-eabi\share\gdb\python\gdb*	.\bin\arm-none-eabi-gdb-py3.exe .\bin\: DLLs*、libgcc_s_seh-1.dll、libgmp-10.dll、libiconv-2.dll、libmpc-3.dll、libmpfr-4.dll、libstdc++-6.dll、libwinpthread-1.dll、libzstd.dll、python311.dll、python311.zip .\arm-none-eabi\: \share\gdb\python\gdb*	{system install path}\xpack-arm-none-eabi-gcc-12.2.1-1.2\bin Example: C:\work\tools\xpack-arm-none-eabi-gcc-12.2.1-1.2\bin {system install path}\xpack-arm-none-eabi-gcc-12.2.1-1.2\arm-none-eabi Example: C:\work\tools\xpack-arm-none-eabi-gcc-12.2.1-1.2\arm-none-eabi\share\gdb\python\gdb	35.4MB 321KB
Output	nvm_programmer.exe	.\dist		44.7MB

1.split the file dependency of the image in new folder nvm

1.1 Tools Installation package reference link:

- **python support version**: 3.11

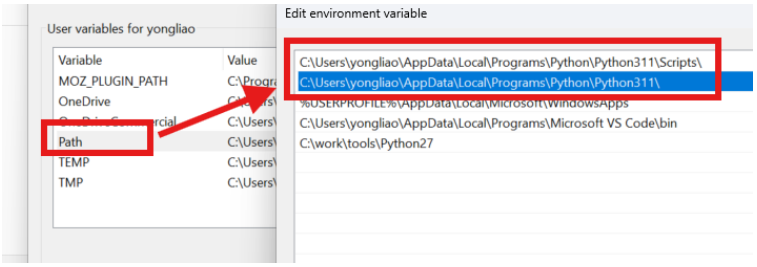
Some support scripts are Python-based. It is recommended to use Python v3.11 downloaded from the following address:
https://www.python.org/downloads/release/python-3115

Add the paths of python binaries to the PATH environment variable.

Example:

set %PATH%=C:\Python311;%PATH%

set %PATH%=C:\Python311\Scripts;%PATH%



pyinstaller :pip install pyinstaller

- **openocd**:

Download the driver (OpenOCD_CH347.zip) for high-speed USB adapter chip CH347 from the following address:
https://github.com/WCHSoftGroup/ch347/releases/tag/CH347_OpenOCD_Release

for windows driver: https://www.wch.cn/downloads/CH341PAR_EXE.htm

Add the OPENOCD_PATH environment variable to the system environment and set the value to the directory that contains openocd.exe.（optional）

Example:

set OPENOCD_PATH =C:\work\tools\OpenOCD_CH347\OpenOCD_CH347\bin

- **xpack-arm-none-eabi-gdb-py3:**

Version 12.2.1-1.2 is recommended and can be found here:

https://github.com/xpack-dev-tools/arm-none-eabi-gcc-xpack/releases/download/v12.2.1-1.2/xpack-arm-none-eabi-gcc-12.2.1-1.2-win32-x64.zip

Add the GDB_CLIENT_PATH environment variable to the system environment and set the value to the directory that contains **arm-none-eabi-gdb-py3.exe**.（optional）

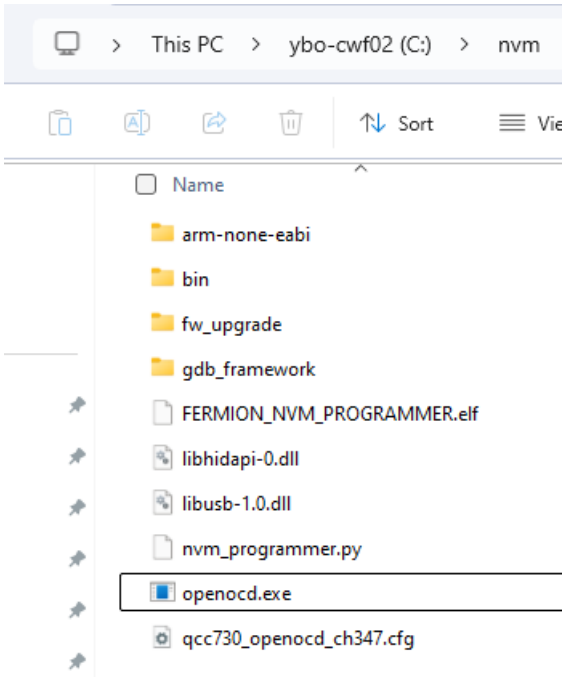
Example:

set GDB_CLIENT_PATH=C:\work\tools\xpack-arm-none-eabi-gcc-12.2.1-1.2-win32-x64\bin

1.2 nvm folder

Both of the above 2 tools(openocd, xpack-arm-none-eabi-gdb) needn't to be installed independently, you just need to put the corresponding .exe files and dependency files under the nvm folder.

The list is as follows:



2. Running nvm_programmer.exe by using PACK_ENABLE

Control switches（PACK_ENABLE） directly by adding environment variable settings

2.1. Packing Command in nvm

After going through the above steps execute the following in the cmd window of nvm to generate the nvm_programmer.exe:

nvm>pyinstaller -F .\nvm_programmer.py --clean --add-data .\gdb_framework:gdb_framework --hidden-import=gdb_framework.gdb_server_ch347 --hidden-import=gdb_framework.gdb_server_jlink --add-data .\bin:bin --add-data .\arm-none-eabi:arm-none-eabi --add-data .*:.

PS:the nvm_programmer.exe is generated in \nvm\dist\

2.2 running the nvm_programmer.exe in \nvm\dist\:

set PACK_ENABLE=True

nvm_programmer.exe -s ch347 --nvm-name rram -b 0x208000 -f {frn_curr_age_with_app_bin.bin} --reset

nvm_programmer.exe -s ch347 --nvm-name rram -b 0x21a400 -f {FERMION_IOE_QCLI_DEMO.bin} --reset

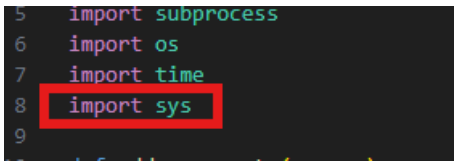
nvm_programmer.exe -s ch347 --nvm-name rram -b 0x20a400 -f {FERMION_SBL_HASHED.elf} --reset

PS: Always burn frn_curr_age_with_app_bin.bin in the first time.

3. Running nvm_programmer.exe by (not suggest)

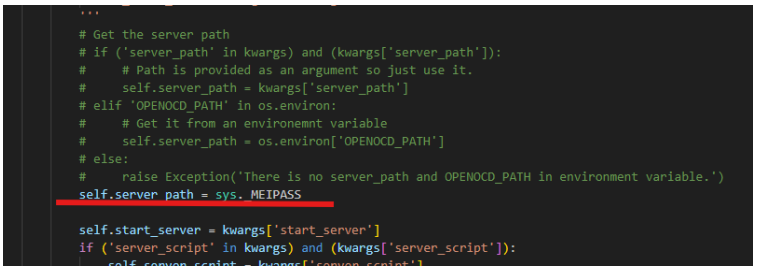
3.1 modify the gdb server path in nvm\gdb_framework\gdb_server_ch347.py

- **import sys**

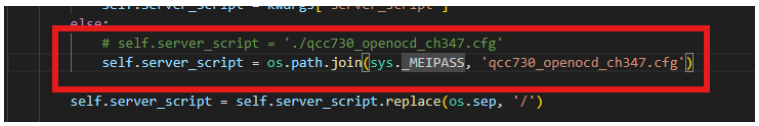


- # Get the server path
  # if ('server_path' in kwargs) and (kwargs['server_path']):
  # # Path is provided as an argument so just use it.
  # self.server_path = kwargs['server_path']
  # elif 'OPENOCD_PATH' in os.environ:
  # # Get it from an environemnt variable
  # self.server_path = os.environ['OPENOCD_PATH']
  # else:

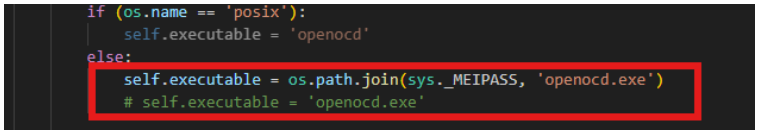
raise Exception('There is no server_path and OPENOCD_PATH in environment variable.')
self.server_path = sys._MEIPASS



- **self.server_script = os.path.join(sys._MEIPASS, 'qcc730_openocd_ch347.cfg')**

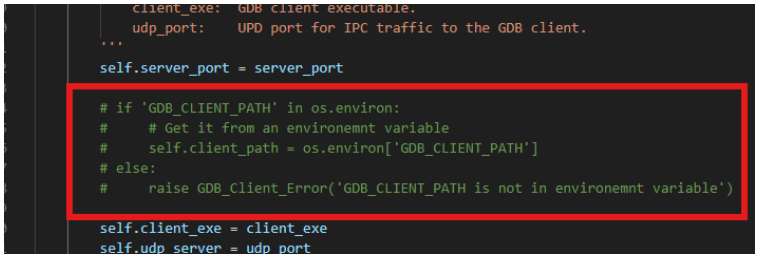


- **self.executable = os.path.join(sys._MEIPASS, 'openocd.exe')**



3.2. modify the gdb client path in nvm\gdb_framework\gdb_client.py

- # if 'GDB_CLIENT_PATH' in os.environ:
  # # Get it from an environemnt variable
  # self.client_path = os.environ['GDB_CLIENT_PATH']
  # else:
  # raise GDB_Client_Error('GDB_CLIENT_PATH is not in environemnt variable')

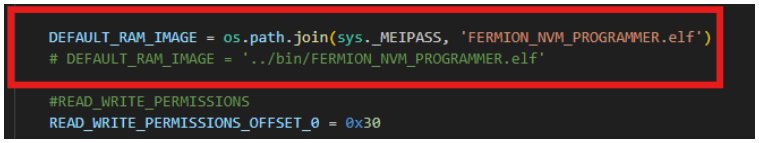


- **self.client_proc = subprocess.Popen([os.path.join(sys._MEIPASS, 'bin', self.client_exe), '-command', __file__.rstrip('c'), '-batch', '-return-child-result'], stdout=self.client_log)**

3.3. modify the fw_upgrade path in nvm\nvm_programmer.py

- **# FW_UPGRADE_SCRIPTS_PATH = os.path.abspath(os.path.join(os.path.dirname(os.path.abspath(__file__)), "../../fw_upgrade"))**
  # sys.path.append(FW_UPGRADE_SCRIPTS_PATH)
  from fw_upgrade.gen_download_table import Download_Table

- **DEFAULT_RAM_IMAGE = os.path.join(sys._MEIPASS, 'FERMION_NVM_PROGRAMMER.elf')**
  # DEFAULT_RAM_IMAGE = '../bin/FERMION_NVM_PROGRAMMER.elf'



3.4 Packing Command in nvm

After going through the above steps execute the following in the cmd window of nvm to generate the nvm_programmer.exe:

nvm>pyinstaller -F .\nvm_programmer.py --clean --add-data .\gdb_framework:gdb_framework --hidden-import=gdb_framework.gdb_server_ch347 --hidden-import=gdb_framework.gdb_server_jlink --add-data .\bin:bin --add-data .\arm-none-eabi:arm-none-eabi --add-data .*:.

3.5 running the nvm_programmer.exe in \nvm\dist\

nvm_programmer.exe -s ch347 --nvm-name rram -b 0x208000 -f {frn_curr_age_with_app_bin.bin} --reset

nvm_programmer.exe -s ch347 --nvm-name rram -b 0x21a400 -f {FERMION_IOE_QCLI_DEMO.bin} --reset

nvm_programmer.exe -s ch347 --nvm-name rram -b 0x20a400 -f {FERMION_SBL_HASHED.elf} --reset

PS: Always burn frn_curr_age_with_app_bin.bin in the first time.