Building STM32 development environment in Linux system

(Using Ubuntu 18.04 + Arm GNU Toolchain 12 + VS Code)

1. Add normal user to root:
   (1) $ sudo vim /etc/sudoers
   (2) Insert the command:

```
# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
gerald  ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

User Name

2. Installing Java
   (1) Installation command:
   $ sudo apt install openjdk-11-jdk
   or $ sudo apt install openjdk-8-jdk (for other versions)

   (2) verify installation:
   $ java –version

```
gerald@embedded: ~
File  Edit  View  Search  Terminal  Help
gerald@embedded:~$ java -version
openjdk version "11.0.19" 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu118.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu118.04.1, mixed mod
e, sharing)
gerald@embedded:~$
```

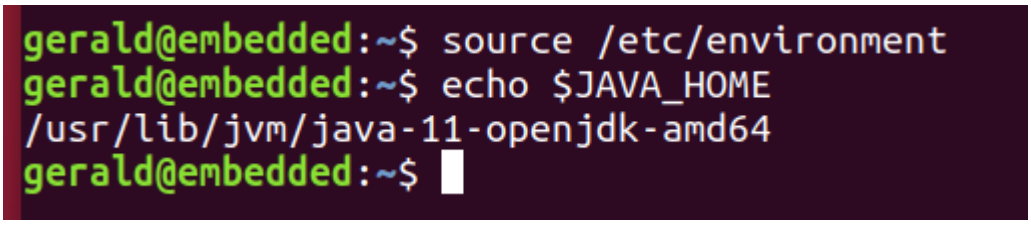
   (3) Add it to environment variable (optional):
   $ sudo vim /etc/environment
   add the java installation path: /usr/lib/jvm/java-11-openjdk-amd64

```
gerald@embedded: ~
File  Edit  View  Search  Terminal  Help
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/u
sr/local/games:/snap/bin"
JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"
~
~
```

   (4) Enable the change:
   $ source /etc/environment

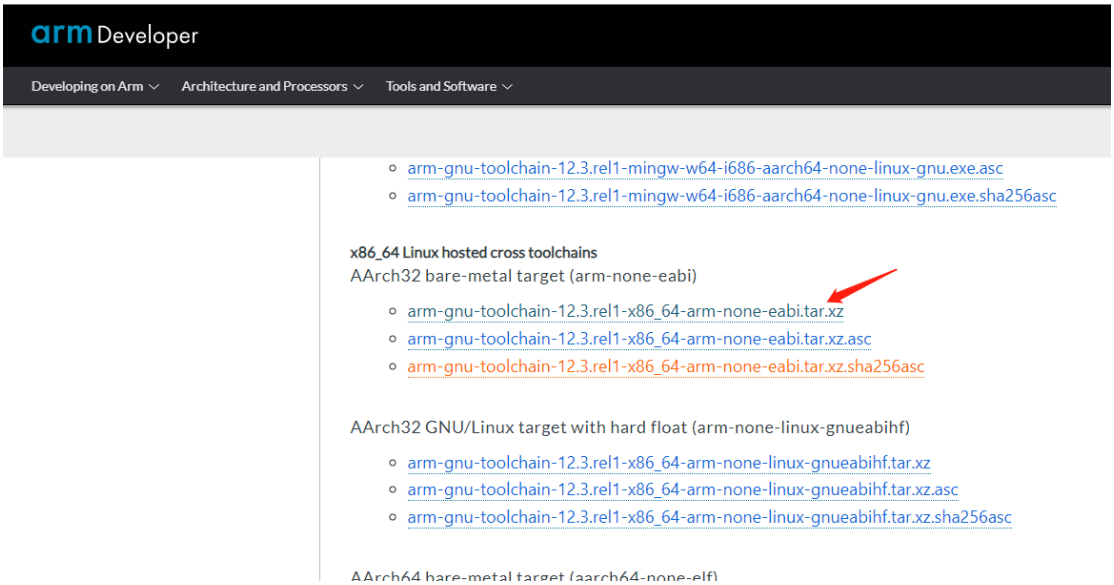
(5) verify the change:

$ echo $JAVA_HOME



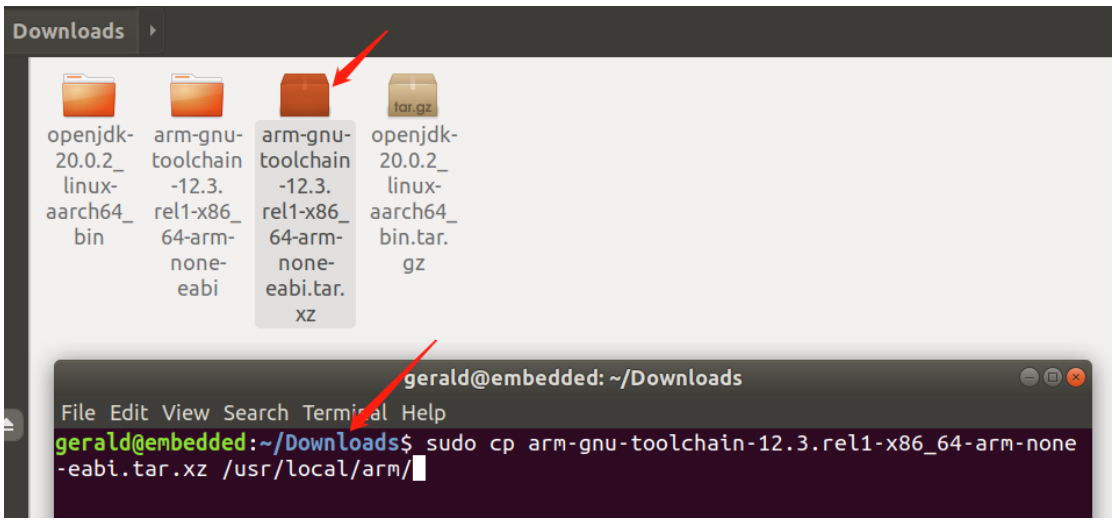
3. Install the GCC-Arm:

(1) Download the gcc-arm toolchain:



(2) Copy the package to /usr/local/arm:

$ sudo cp arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz /usr/local/arm/



(3) Extract the package:

$ sudo tar -xvf arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz

(4) Add path to the environment:

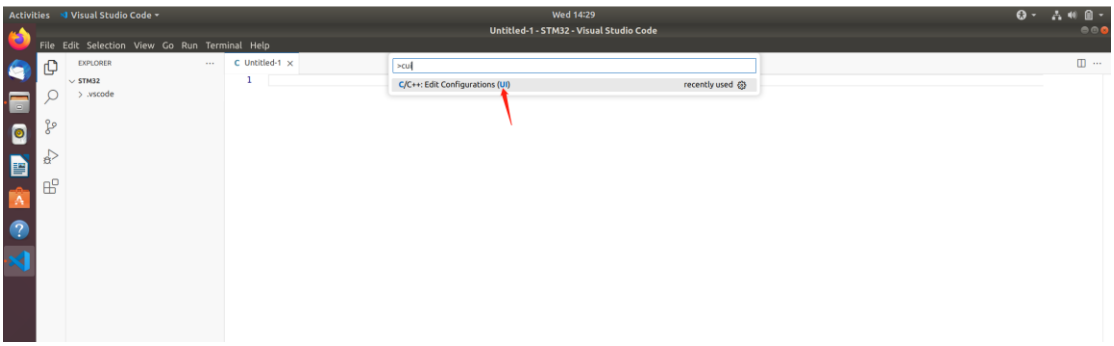
$ sudo vim /etc/profile

```
gerald@embedded: /usr/local/arm/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi
File  Edit  View  Search  Terminal  Help
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi

export PATH=$PATH:/usr/local/arm/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eab
i/bin
                                                                  29,1          Bot
```

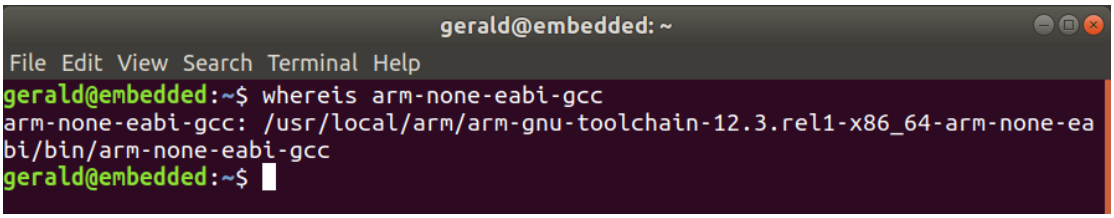
(5) After restart, verify the change:

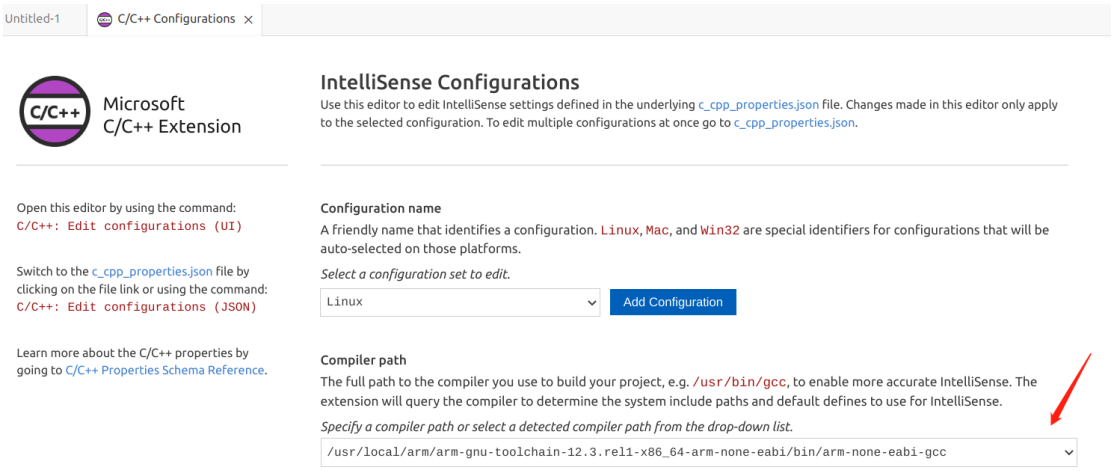
```
gerald@embedded: ~
File  Edit  View  Search  Terminal  Help
gerald@embedded:~$ arm-none-eabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-none-eabi-gcc
COLLECT_LTO_WRAPPER=/usr/local/arm/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-e
abi/bin/../libexec/gcc/arm-none-eabi/12.3.1/lto-wrapper
Target: arm-none-eabi
Configured with: /data/jenkins/workspace/GNU-toolchain/arm-12/src/gcc/configure
--target=arm-none-eabi --prefix=/data/jenkins/workspace/GNU-toolchain/arm-12/bui
ld-arm-none-eabi/install --with-gmp=/data/jenkins/workspace/GNU-toolchain/arm-12
/build-arm-none-eabi/host-tools --with-mpfr=/data/jenkins/workspace/GNU-toolchai
n/arm-12/build-arm-none-eabi/host-tools --with-mpc=/data/jenkins/workspace/GNU-t
oolchain/arm-12/build-arm-none-eabi/host-tools --with-isl=/data/jenkins/workspac
e/GNU-toolchain/arm-12/build-arm-none-eabi/host-tools --disable-shared --disable
-nls --disable-threads --disable-tls --enable-checking=release --enable-language
s=c,c++,fortran --with-newlib --with-gnu-as --with-gnu-ld --with-sysroot=/data/j
enkins/workspace/GNU-toolchain/arm-12/build-arm-none-eabi/install/arm-none-eabi
--with-multilib-list=aprofile,rmprofile --with-pkgversion='Arm GNU Toolchain 12.
3.Rel1 (Build arm-12.35)' --with-bugurl=https://bugs.linaro.org/
Thread model: single
Supported LTO compression algorithms: zlib
gcc version 12.3.1 20230626 (Arm GNU Toolchain 12.3.Rel1 (Build arm-12.35))
```

(6) Set VSCode:

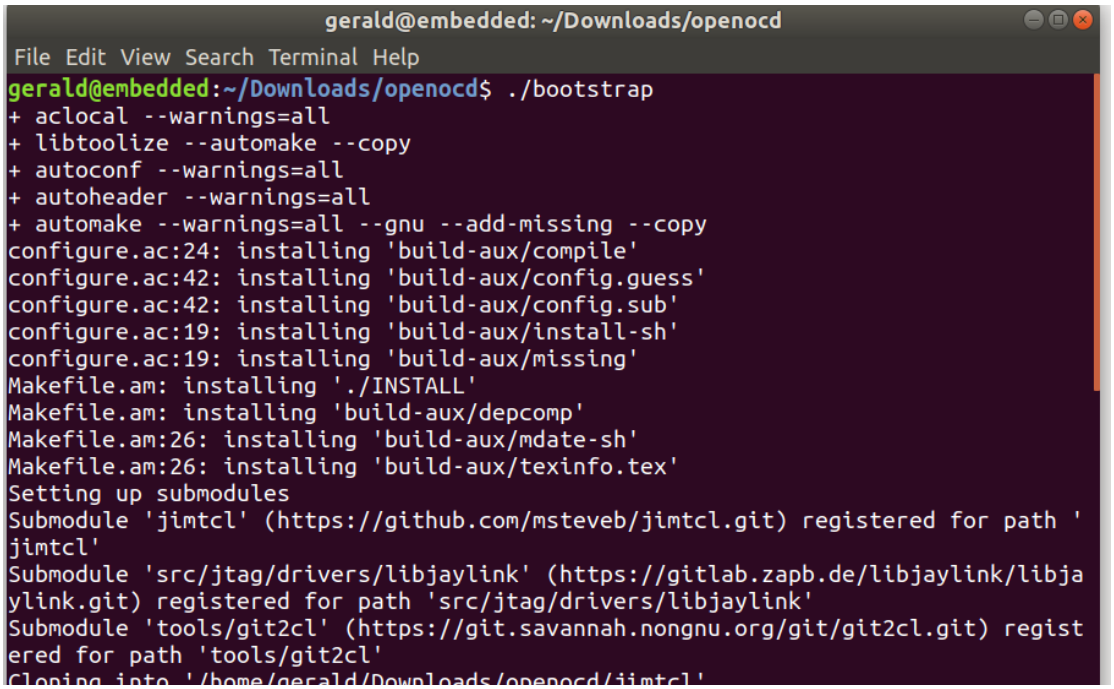


find GCC installation path:





4. Install the OpenOCD:
   (1) Use git to download the openocd:
   $ git clone https://github.com/openocd-org/openocd.git
   (2) Download the essential library:
   $ sudo apt install build-essential pkg-config autoconf automake libtool libusb-dev libusb-1.0-0-dev libhidapi-dev libtool libsysfs-dev
   (3) Open the OpenOCD directory and enter $ sudo ./bootstrap



(4) Enter the $ sudo ./configure

```
gerald@embedded: ~/Downloads/openocd
File  Edit  View  Search  Terminal  Help
gerald@embedded:~/Downloads/openocd$ ./configure
checking for makeinfo... no
configure: WARNING: Info documentation will not be built.
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking for gcc option to accept ISO C99... none needed
checking for ranlib... ranlib
```

(5) Enter $ sudo make and then enter $ sudo make install to finish the installation.

(6) Verify the installation:

```
gerald@embedded:~$ openocd --version
Open On-Chip Debugger 0.12.0+dev-01311-gea0ac6dc7 (2023-08-30-01:40)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
gerald@embedded:~$
```

5. Install STM32CubeMX

(1) After unzipping the package, set the installation file as an executable file:

$ chmod 777 SetupSTM32CubeMX-6.9.1

(2) Run the installation application:

$ ./SetupSTM32CubeMX-6.9.1

(3) After finishing the installation, add STM32CubeMX path to environment variable:

$ sudo vim ~/.bashrc

```
File  Edit  View  Search  Terminal  Help
gerald@embedded:~$ sudo vim ~/.bashrc
```

And then adding this command to the end of file:

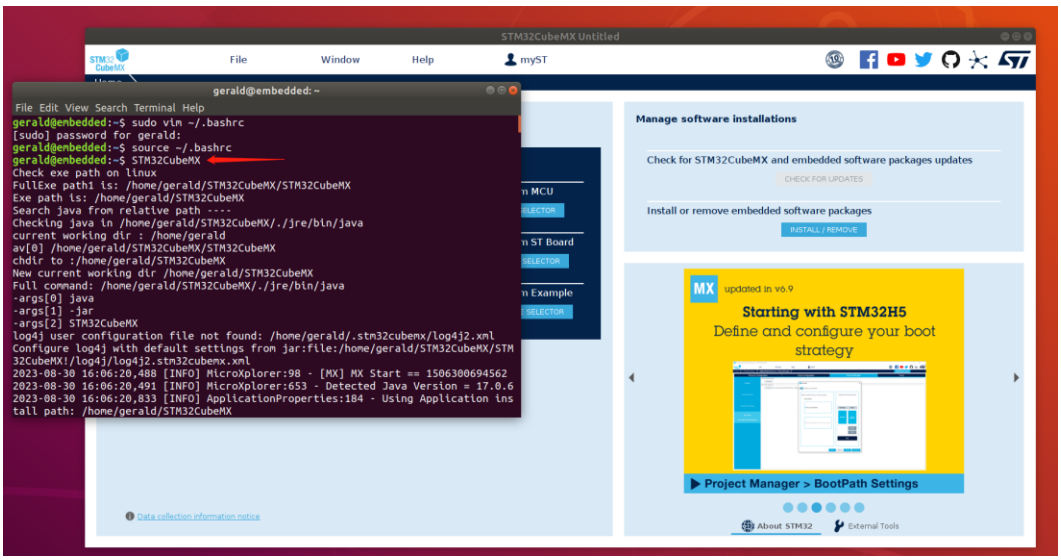
```
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

#STM32CubeMX
if [ -x /home/gerald/STM32CubeMX/STM32CubeMX ]; then
     alias STM32CubeMX='/home/gerald/STM32CubeMX/STM32CubeMX'
fi
"~/.bashrc" 122L, 3903C                                   121,60        Bot
```

Save and quit Vim. Next enable the change:

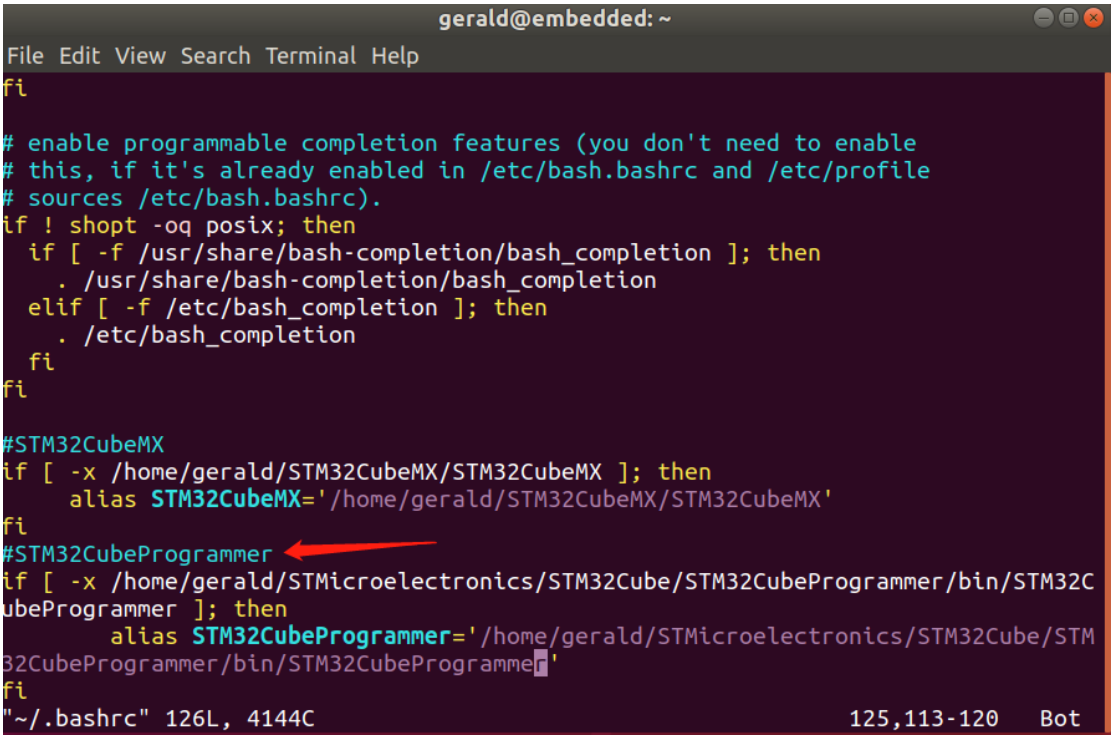
$ source ~/.bashrc

```
gerald@embedded:~$ sudo vim ~/.bashrc
[sudo] password for gerald:
gerald@embedded:~$ source ~/.bashrc
```

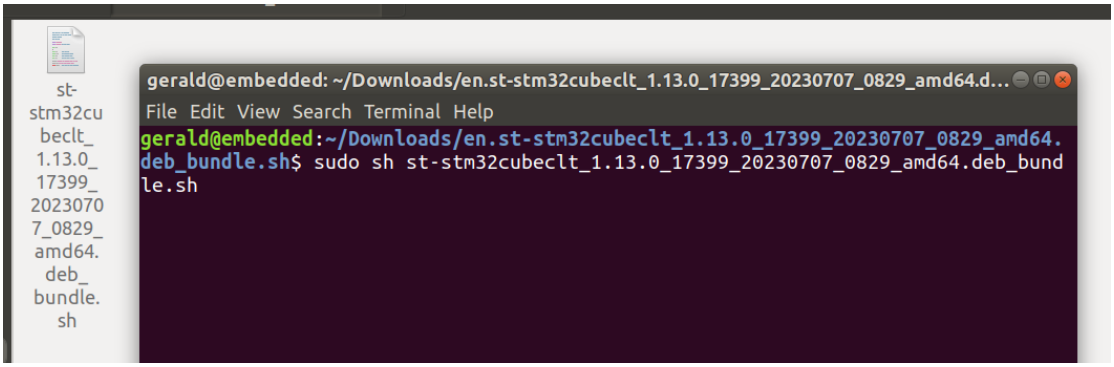
(4) Verify the Change:



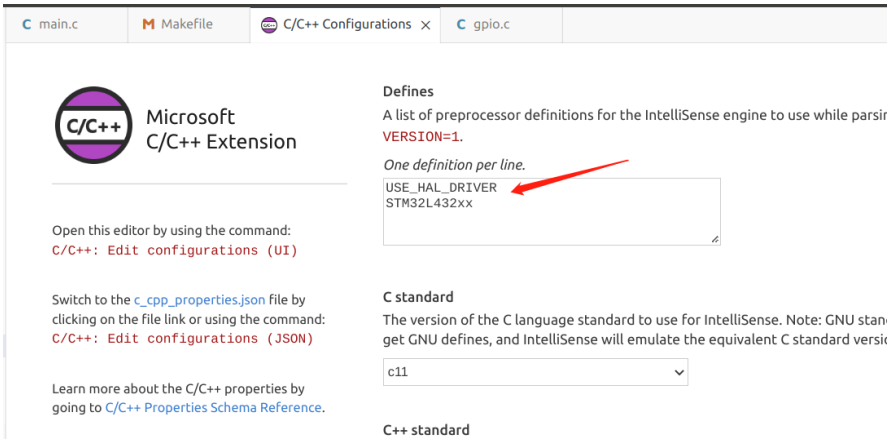
6. Add STM32CubeProgrammer to the environment path:
   $ sudo vim ~/.bashrc and then $ source ~/.bashrc



7. Install the STM32CubeCLT:



8. VSCode and Makefile setting:
   (1) VScode



get Defines from Makefiles:

main.c | Makefile | C/C++ Configurations | gpio.c

Makefile

```
# fpu
FPU = -mfpu=fpv4-sp-d16

# float-abi
FLOAT-ABI = -mfloat-abi=hard

# mcu
MCU = $(CPU) -mthumb $(FPU) $(FLOAT-ABI)

# macros for gcc
# AS defines
AS_DEFS =

# C defines
C_DEFS =  \
-DUSE_HAL_DRIVER \
-DSTM32L432xx


# AS includes
AS_INCLUDES =
```

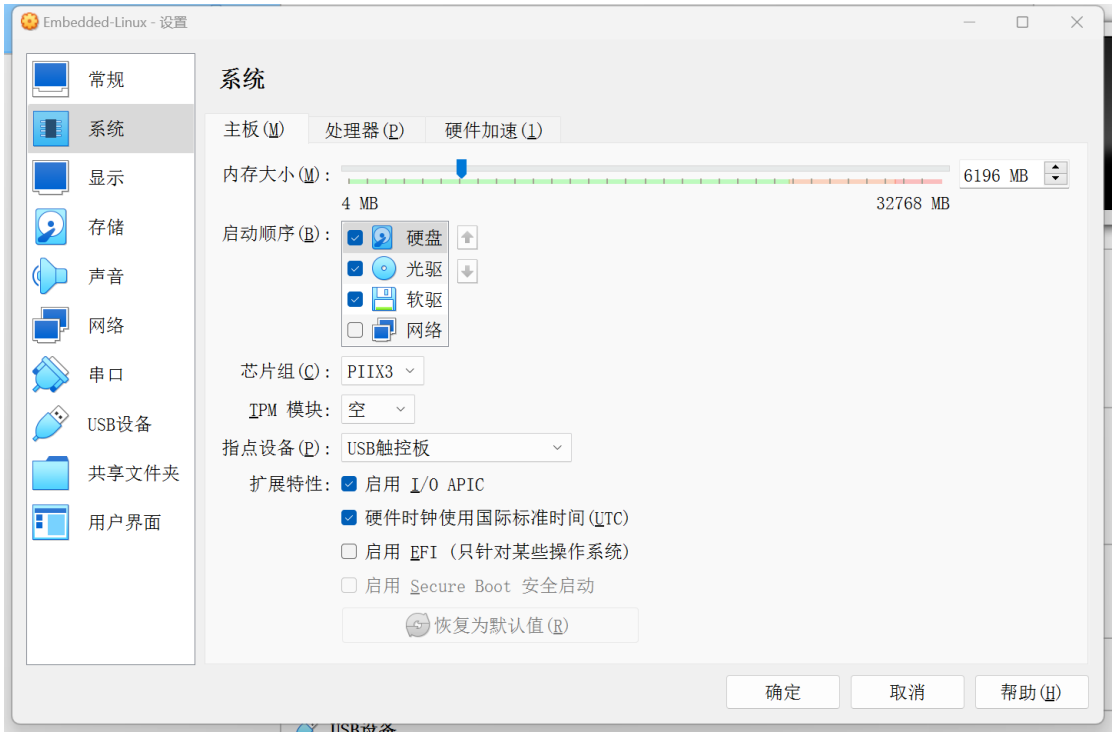
(2) Add make flash command to Makefile file:

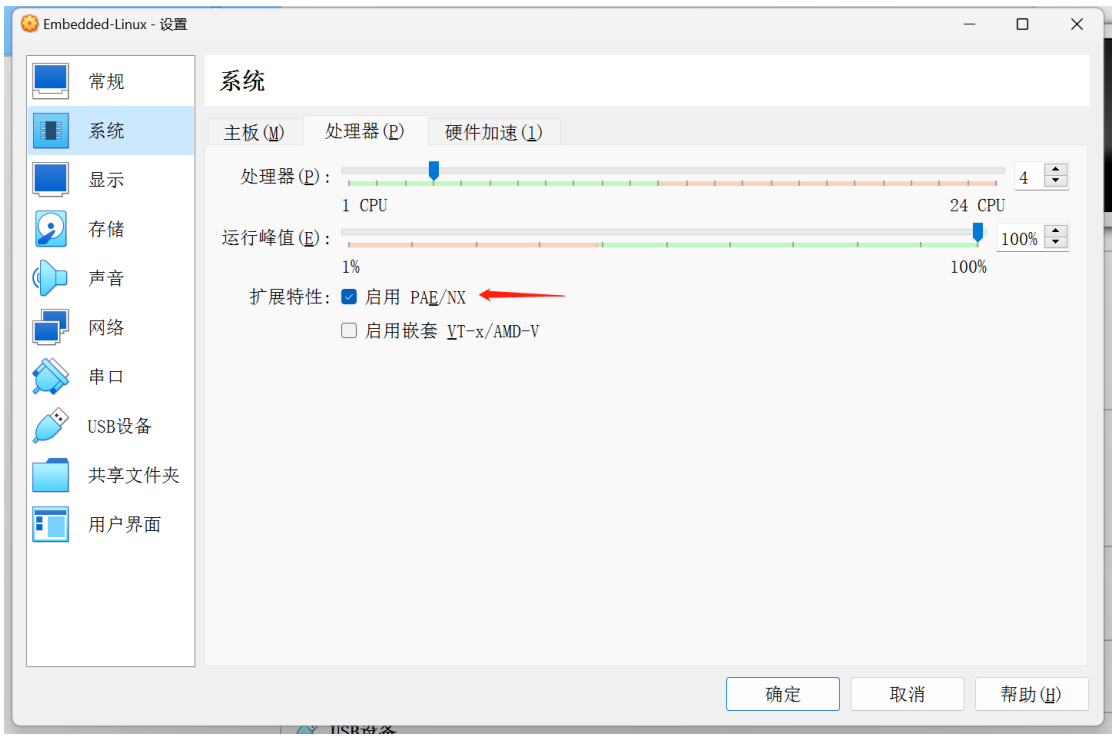
Open | Makefile ~/Documents/STM32/STM32L432KCU6 | Save

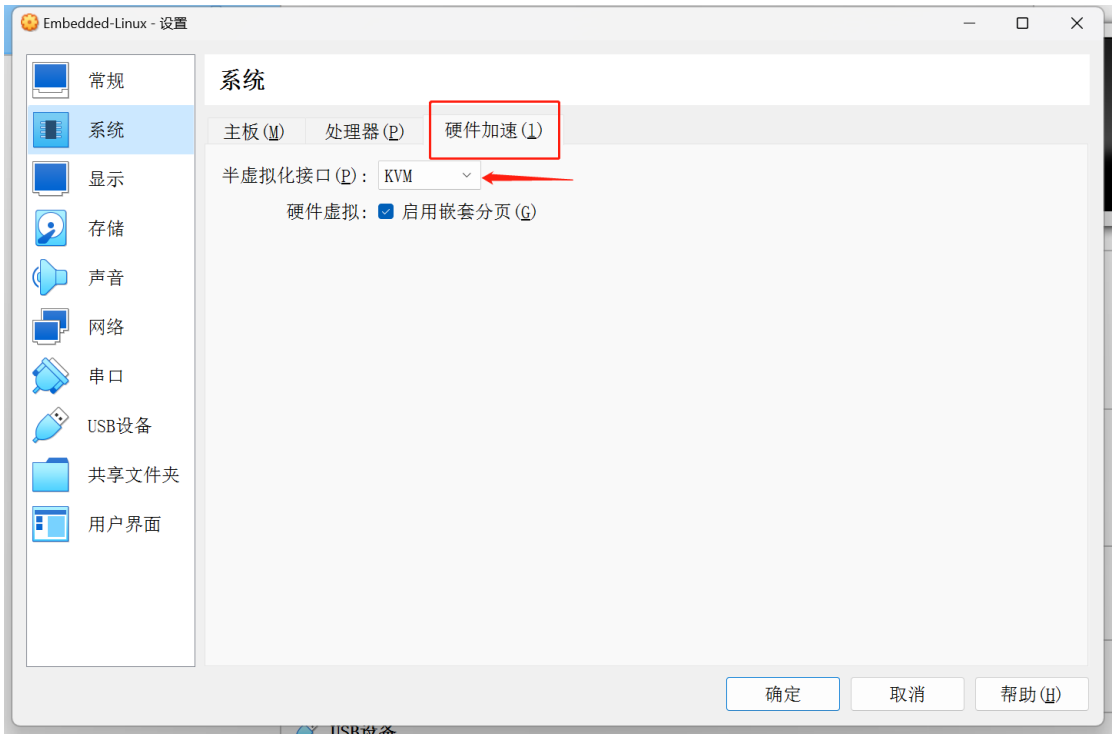
```
        $(BIN) $< $@

$(BUILD_DIR):
        mkdir $@

#######################################
# clean up
#######################################
clean:
        -rm -fR $(BUILD_DIR)

#######################################
# dependencies
#######################################
-include $(wildcard $(BUILD_DIR)/*.d)

#######################################
# Auto Run
#######################################
#Use The Tool By Openocd Download the Project
OPENOCD := openocd -f interface/stlink.cfg \
        -f target/stm32l4x.cfg
# download your program
flash: all
        $(OPENOCD) -c init \
                -c 'reset halt' \
                -c 'flash write_image erase $(BUILD_DIR)/$(TARGET).elf' \
                -c 'reset run' \
                -c exit
# *** EOF ***
```

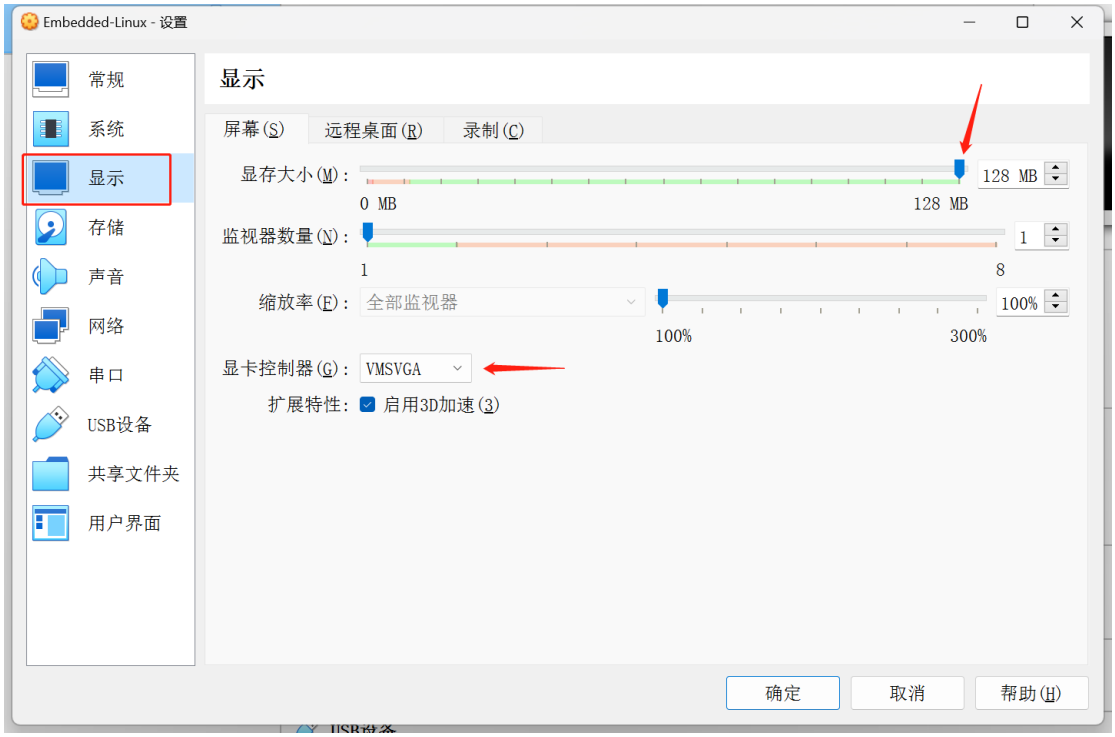
Makefile | Tab Width: 8 | Ln 180, Col 40 | INS

9. Set full screen display in Virtualbox:






Embedded-Linux - 设置
常规
系统
显示
存储
声音
网络
串口
USB设备
共享文件夹
用户界面
系统
主板(M)
处理器(P)
硬件加速(1)
半虚拟化接口(P)：
KVM
硬件虚拟：
启用嵌套分页(G)
确定
取消
帮助(H)


Embedded-Linux - 设置
常规
系统
显示
存储
声音
网络
串口
USB设备
共享文件夹
用户界面
显示
屏幕(S)
远程桌面(R)
录制(C)
显存大小(M)：
128 MB
0 MB
128 MB
监视器数量(N)：
1
1
8
缩放率(F)：
全部监视器
100%
100%
300%
显卡控制器(G)：
VMSVGA
扩展特性：
启用3D加速(3)
确定
取消
帮助(H)