

第一篇 基本的 Linux 仿真环境和交叉编译工具链的配置

0、电脑环境：Win10x64

1、下载工具链和仿真环境

https://dl.espressif.com/dl/esp32_win32_msys2_environment_and_toolchain-20160816.zip

2、解压到 c 盘根目录，将会出现一个 msys32 的文件夹。

3、运行 msys32 目录下的 msys2_shell.cmd，输入命令“cd c:/”进入 c 盘根目录

```
Copying skeleton files.
These files are for the users to personalise their msys2 experience.
They will never be overwritten nor automatically updated.

'./.bashrc' -> '/home/shentq/./.bashrc'
'./.bash_logout' -> '/home/shentq/./.bash_logout'
'./.bash_profile' -> '/home/shentq/./.bash_profile'
'./.inputrc' -> '/home/shentq/./.inputrc'
'./.profile' -> '/home/shentq/./.profile'
'./.vimrc' -> '/home/shentq/./.vimrc'

shentq@DESKTOP-2G67VB9 MSYS ~
$ cd c:/
```

4、输入 mkdir esp32_idf,创建 esp32_idf 文件夹，输入 cd esp32_idf/, 进入 esp32_idf 文件夹

```
shentq@DESKTOP-2G67VB9 MSYS /c
$ mkdir esp32_idf

shentq@DESKTOP-2G67VB9 MSYS /c
$ cd esp32_idf/
```

5、克隆 idf 固件库

输入 git clone --recursive <https://github.com/espressif/esp-idf.git>

这个会持续一段时间，除了下载此 git 代码外还要下载其依赖的其他几个，慢慢等待即可

```
shentq@DESKTOP-2G67VB9 MSYS /c/esp32_idf
$ git clone --recursive https://github.com/espressif/esp-idf.git
```

7、输入 export IDF_PATH="C:/esp32_idf/esp-idf"添加编译需要的环境变量，如果你不想每次都输入，可以打开 C:\msys32\etc\profile.d\esp32_toolchain.sh（使用 notepad 打开或者记事本），添加 export IDF_PATH="C:/esp32_idf/esp-idf"一行，保存并退出。

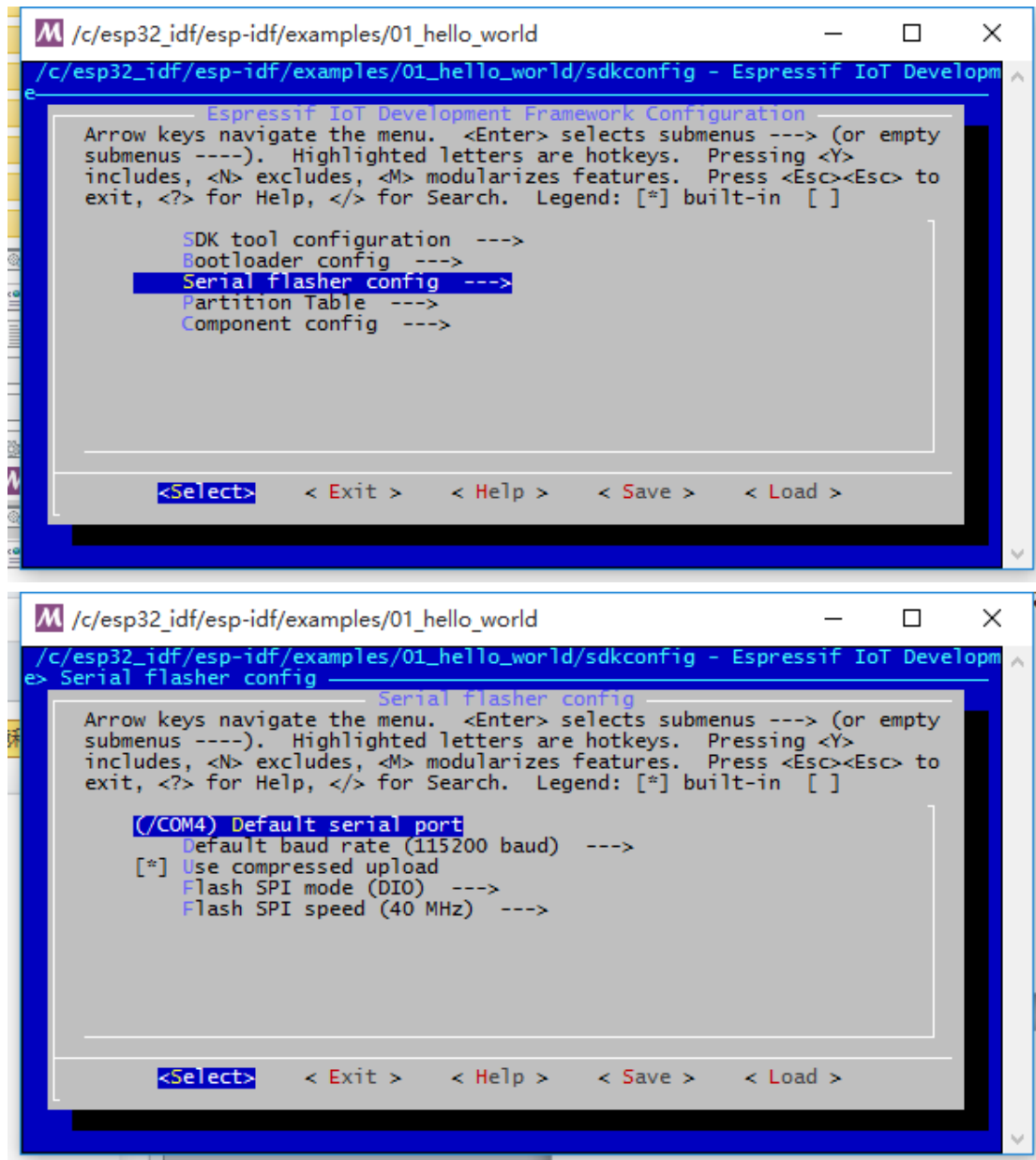
```
export PATH="/mingw32/bin:/usr/local/bin:/usr/bin:/bin:/c/windows/system32:/c/windows/opt/xtensa-esp32-elf/bin"
export IDF_PATH="C:/path/to/esp-idf"
```

8、输入 cd example 进入工程目录 example 目录，下面有很多例程我们一 01hello_world 为例，输入 cd 01_hello_world 进入第一个工程目录。

```
shentq@DESKTOP-2G67VB9 MSYS /c/esp32_idf/esp-idf/examples
$ cd 01_hello_world/

shentq@DESKTOP-2G67VB9 MSYS /c/esp32_idf/esp-idf/examples/01_hello_world
$ |
```

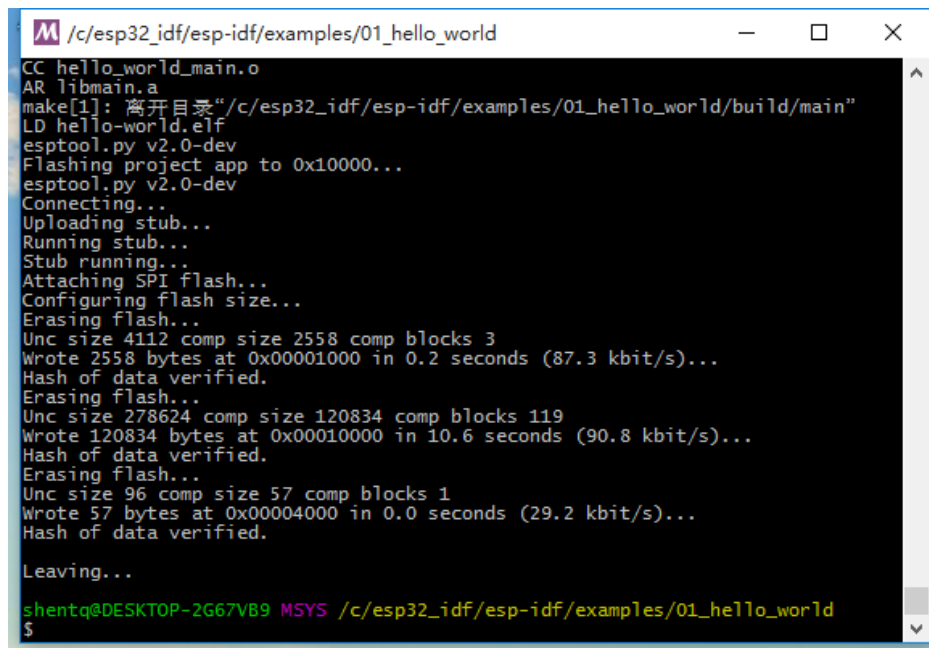
9、输入 `make menuconfig` 选择“Serial flasher config”按回车配置串口号，光标停留在在 Default Serial Prot 的时候继续按回车，输入 `/COM4`(我电脑上的串口是 COM4)。其他的不用修改，使用向右方向键，回车，向右方向键，回车即可退出。最后一次回车保存设置参数。



10、输入 `make clean` 删除了 builds 目录下的文件

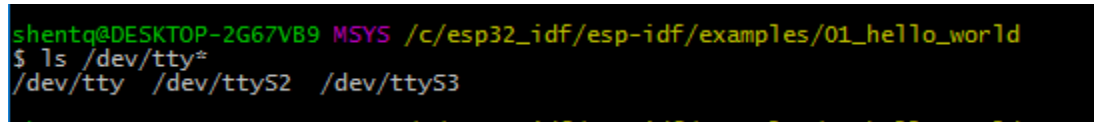
```
rcbuf.d library/nghttp2_hd_huffman.d library/nghttp2_mem.d library/nghttp2_parser.d
make[1]: 离开目录"/c/esp32_idf/esp-idf-template/build/nghttp"
make[1]: 进入目录"/c/esp32_idf/esp-idf-template/build/nvs_flash"
RM libnvs_flash.a src/nvs_api.o src/nvs_storage.o src/nvs_pagemanager.o
c/nvs_page.o src/nvs_types.o src/nvs_api.d src/nvs_storage.d src/nvs_pag
h_list.d src/nvs_page.d src/nvs_types.d
make[1]: 离开目录"/c/esp32_idf/esp-idf-template/build/nvs_flash"
make[1]: 进入目录"/c/esp32_idf/esp-idf-template/build/spi_flash"
RM libspi_flash.a ./esp_spi_flash.o ./esp_spi_flash.d
make[1]: 离开目录"/c/esp32_idf/esp-idf-template/build/spi_flash"
make[1]: 进入目录"/c/esp32_idf/esp-idf-template/build/tcpip_adapter"
RM libtcpip_adapter.a ./tcpip_adapter_lwip.o ./tcpip_adapter_lwip.d
make[1]: 离开目录"/c/esp32_idf/esp-idf-template/build/tcpip_adapter"
make[1]: 进入目录"/c/esp32_idf/esp-idf-template/build/main"
RM libmain.a ./main.o ./main.d
make[1]: 离开目录"/c/esp32_idf/esp-idf-template/build/main"
RM /c/esp32_idf/esp-idf-template/build/app-template.elf
shentq@DESKTOP-2G67VB9 MSYS /c/esp32_idf/esp-idf-template
```

11、确保你的模块已经上电，GPIO0/DL 引脚为低电平，按下复位键，使模块进入串口 boot 并等待上位机链接，下载程序。然后输入 `make flash` 就可以编译，编译大概需要三分钟左右的的时间，编译完成后自动进入下载环节。



```
/c/esp32_idf/esp-idf/examples/01_hello_world
CC hello_world_main.o
AR libmain.a
make[1]: 离开目录"/c/esp32_idf/esp-idf/examples/01_hello_world/build/main"
LD hello-world.elf
esptool.py v2.0-dev
Flashing project app to 0x10000...
esptool.py v2.0-dev
Connecting...
Uploading stub...
Running stub...
Stub running...
Attaching SPI flash...
Configuring flash size...
Erasing flash...
Unc size 4112 comp size 2558 comp blocks 3
Wrote 2558 bytes at 0x00001000 in 0.2 seconds (87.3 kbit/s)...
Hash of data verified.
Erasing flash...
Unc size 278624 comp size 120834 comp blocks 119
Wrote 120834 bytes at 0x00010000 in 10.6 seconds (90.8 kbit/s)...
Hash of data verified.
Erasing flash...
Unc size 96 comp size 57 comp blocks 1
Wrote 57 bytes at 0x00004000 in 0.0 seconds (29.2 kbit/s)...
Hash of data verified.
Leaving...
shentq@DESKTOP-2G67VB9 MSYS /c/esp32_idf/esp-idf/examples/01_hello_world
$
```

12、等待下载完成后请将GPIO0/DL连接至高电平，然后按下复位键即可运行。输入 `ls /dev/tty*` 查看shell环境下串口的设备名称，我电脑经测试知道COM4对应了/dev/ttyS3。（测试方法：不断的使用cat命令测试看那个串口设备有输出信息）。



```
shentq@DESKTOP-2G67VB9 MSYS /c/esp32_idf/esp-idf/examples/01_hello_world
$ ls /dev/tty*
/dev/tty /dev/ttyS2 /dev/ttyS3
```

13、输入 `cat /dev/ttyS3` shell 页面将会打印串口输出的信息

```
rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0x00
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3ffc0000,len:0
load:0x3ffc0000,len:920
load:0x40078000,len:2624
load:0x40098000,len:508
entry 0x40098118
I (81) heap_alloc_caps: Initializing heap allocator:
I (82) heap_alloc_caps: Region 19: 3FFB41F0 len 0002BE10 tag 0
I (83) heap_alloc_caps: Region 25: 3FFE8000 len 00018000 tag 1
I (93) cpu_start: Pro cpu up.
I (98) cpu_start: Starting app cpu, entry point is 0x40080954
I (0) cpu_start: App cpu up.
I (114) cpu_start: Pro cpu start user code
rtc v112 Sep 22 2016 16:08:39
XTAL 40M
I (150) cpu_start: Starting scheduler on PRO CPU.
I (43) cpu_start: Starting scheduler on APP CPU.
fr2_timer_task_hdl:3ffb8c58, prio:22, stack:2048
Hello world!
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
Restarting in 6 seconds...
Restarting in 5 seconds...
```

14、大功告成！

第二篇 eclipse 环境配置

0、电脑环境：Win10x64

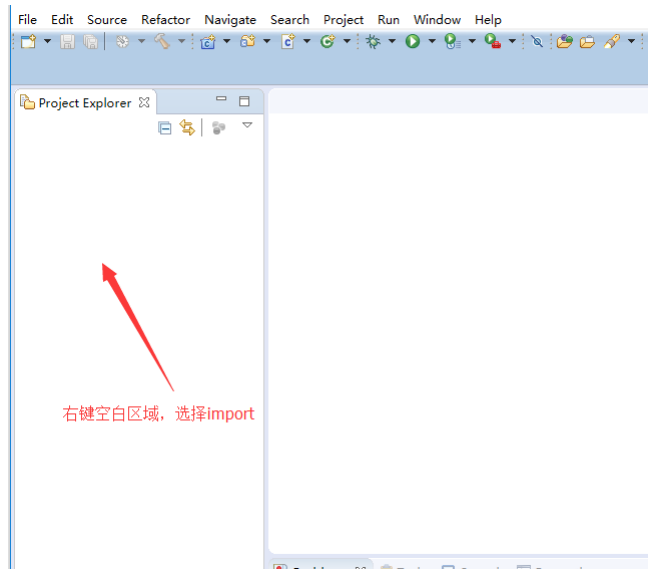
下载 eclipse 的 c++版本 IDE(eclipse-cpp-neon-R-win32.zip)

下载 jre, java 运行时 (jre_8u101_windows_i586_8.0.1010.13.exe)

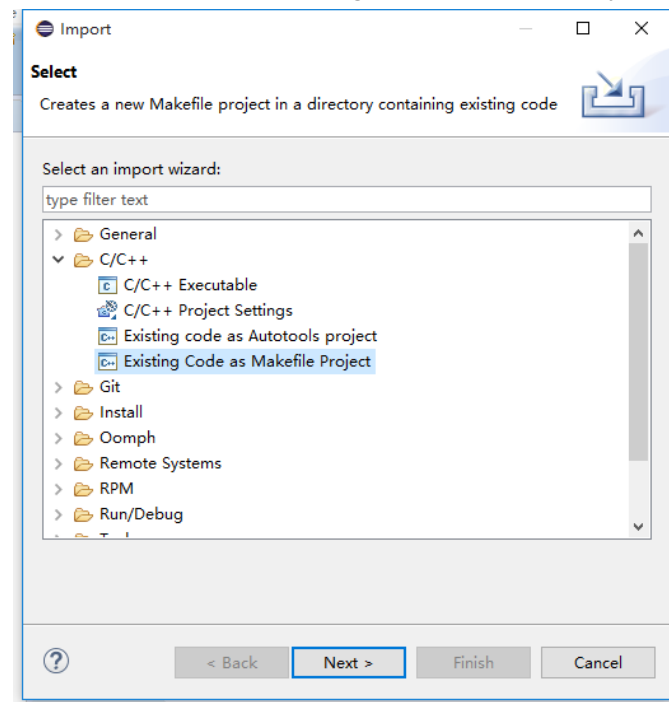
1、安装 jre，完成后解压 eclipse 到 c 盘根目录。运行 eclipse，界面还挺好看，我选择默认的 workspace。



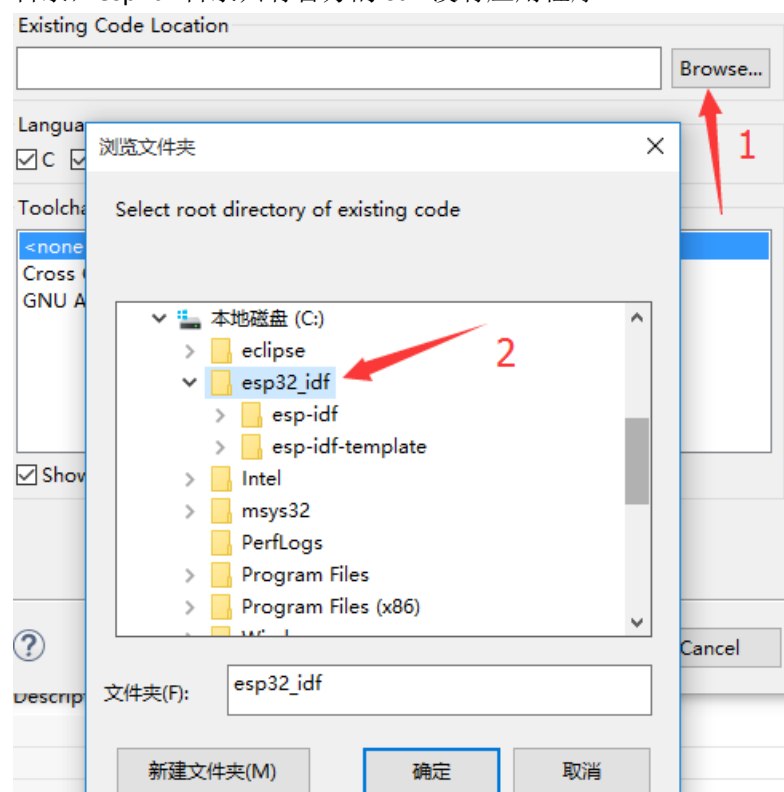
2、进入 eclipse 主页面，第一次打开会出现 welcome 界面，关闭即可，然后会进入主工作区域，右键左侧的空白区域，选择 import 快捷菜单，如图示位置



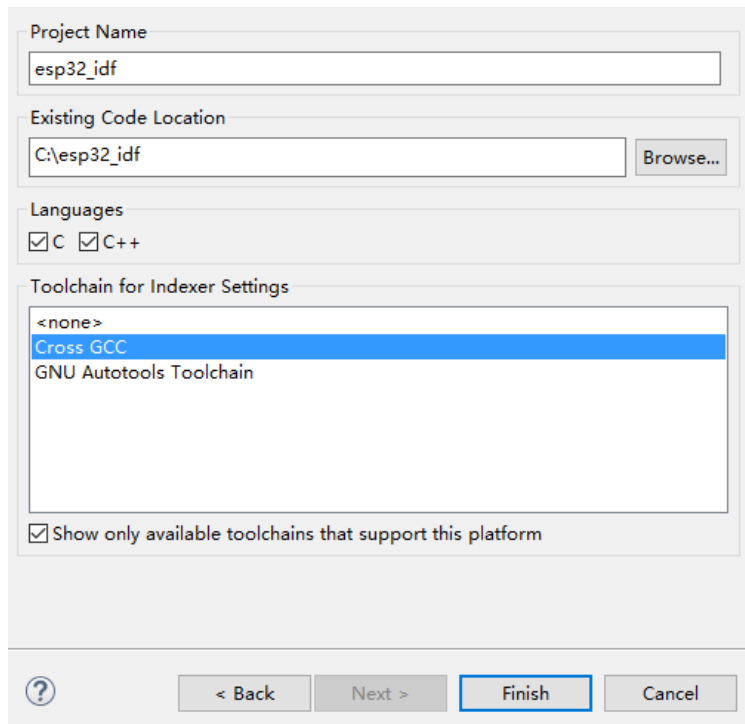
3、选择 C/C++ 下面的 Existing code ad Makefile Project,单击 Next。



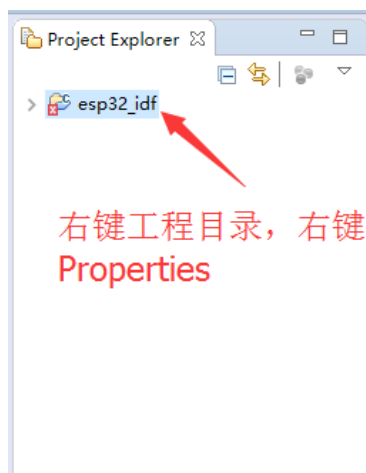
4、选择 Browse，然后选择 c 盘根目录自己建立的 esp32_idf 文件夹，千万不要选择 esp-idf 目录，esp-idf 目录只有官方的 sdk 没有应用程序。



5、选择编译工具链：Cross GCC，然后选择 Finish



6、接下来需要设置工程的一些参数了。



7、选择 C/C++ Build 页面下的 Environment 选项卡，

A) 单击 Add 按钮，在 name 栏输入 V，value 栏输入 1；

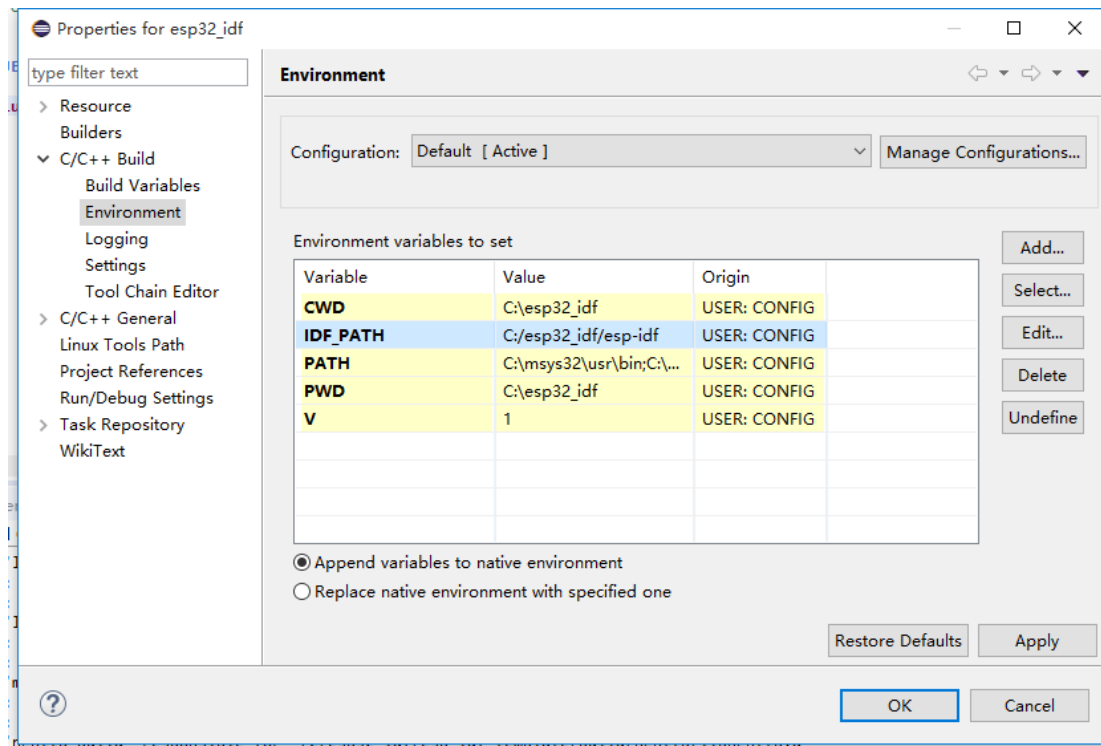
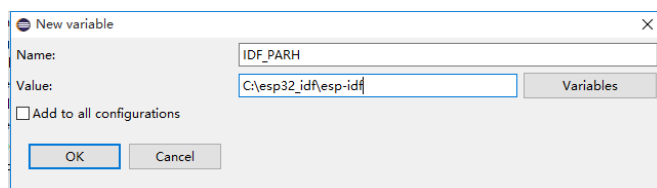
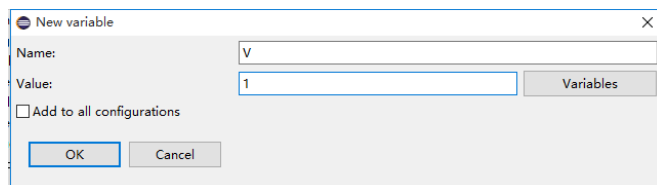
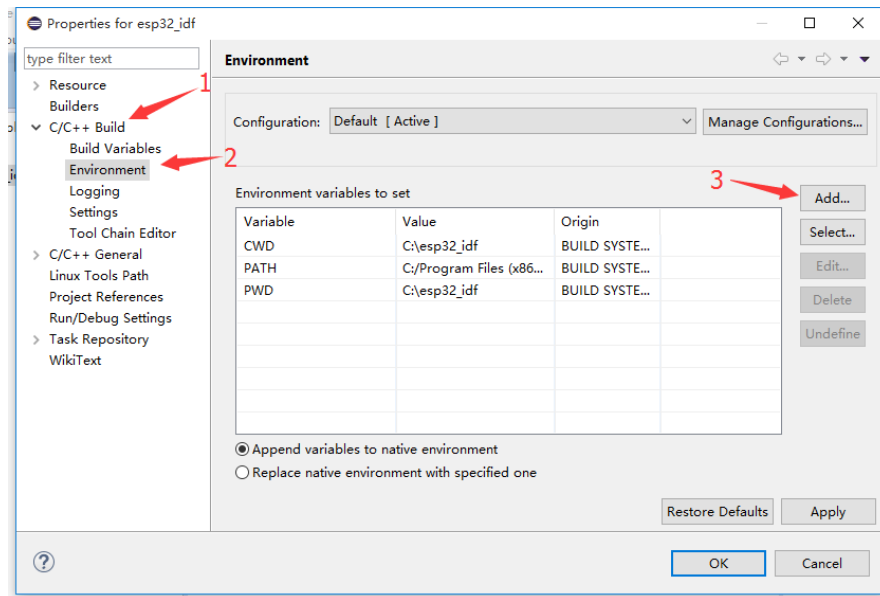
B) 再次单击 Add 按钮，在 name 栏输入 IDF_PATH，value 输入 esp-idf 固件库的目录。C:/esp32_idf/esp-idf，分清楚，这里选择的是 idf 固件库的目录；需要注意的是，此处是/不是\，否则编译会找不到正确的路径。

C) 修改 PATH：删除原来的内容，然后输入

`C:\msys32\usr\bin;C:\msys32\mingw32\bin;C:\msys32\opt\xtensa-esp32-elf\bin`

（前提是你按照教程解压的官方工具链文件）。

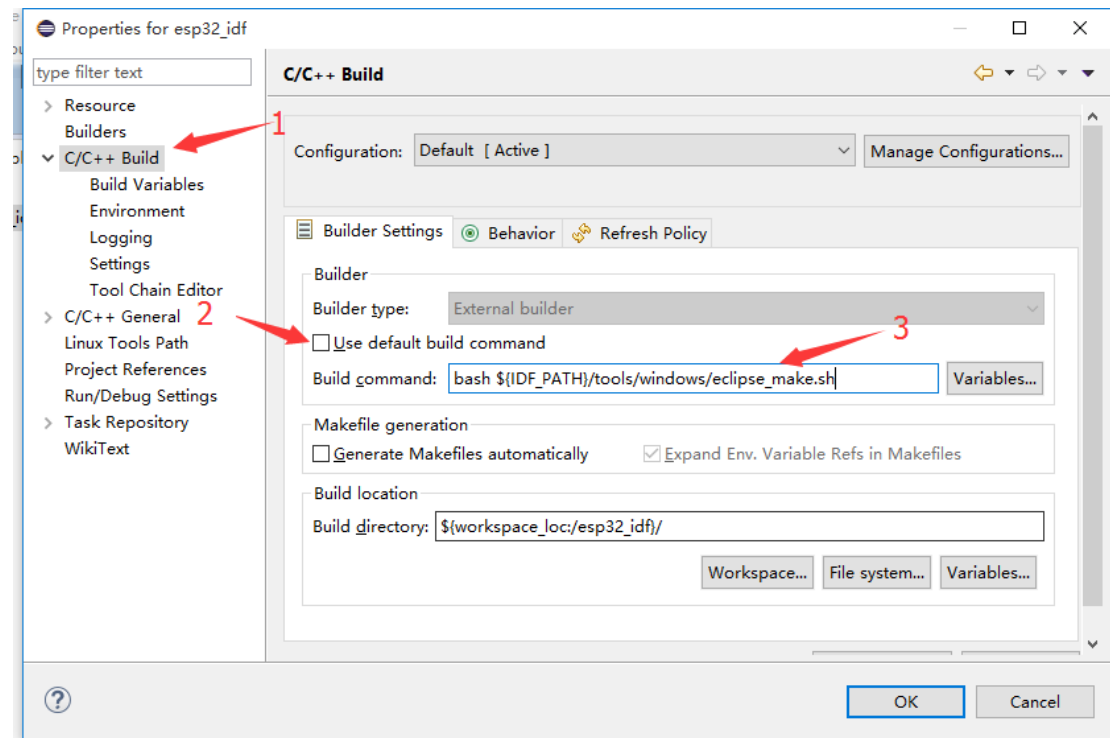
输入完成后单击 OK。



8、选择 C/C++ Build 选项，然后将 "Use default build command" 选项前面的标记取消，Build command 选项就处于可编辑的状态了，输入

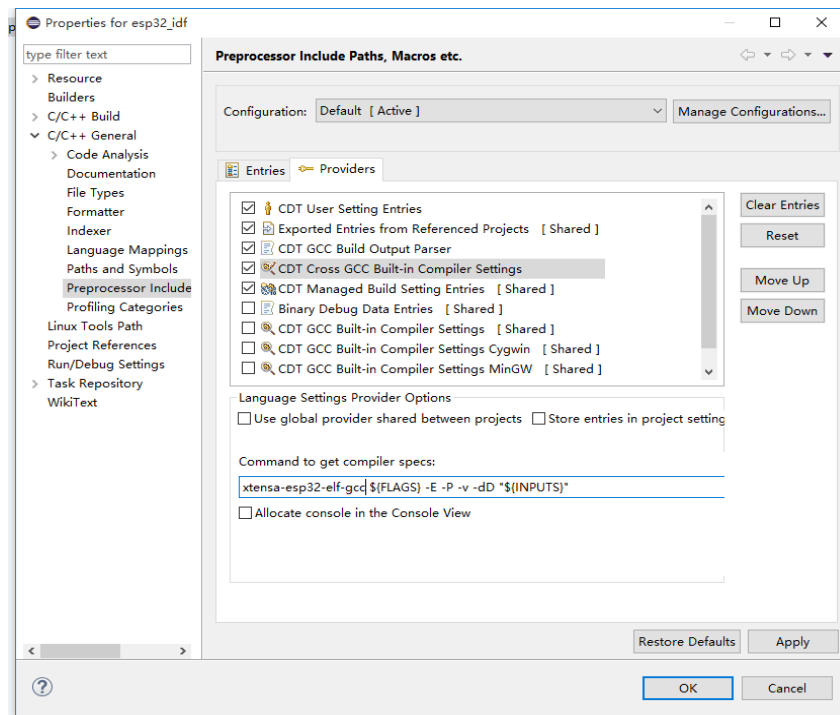
```
bash ${IDF_PATH}/tools/windows/eclipse_make.sh
```

然后点击 OK。



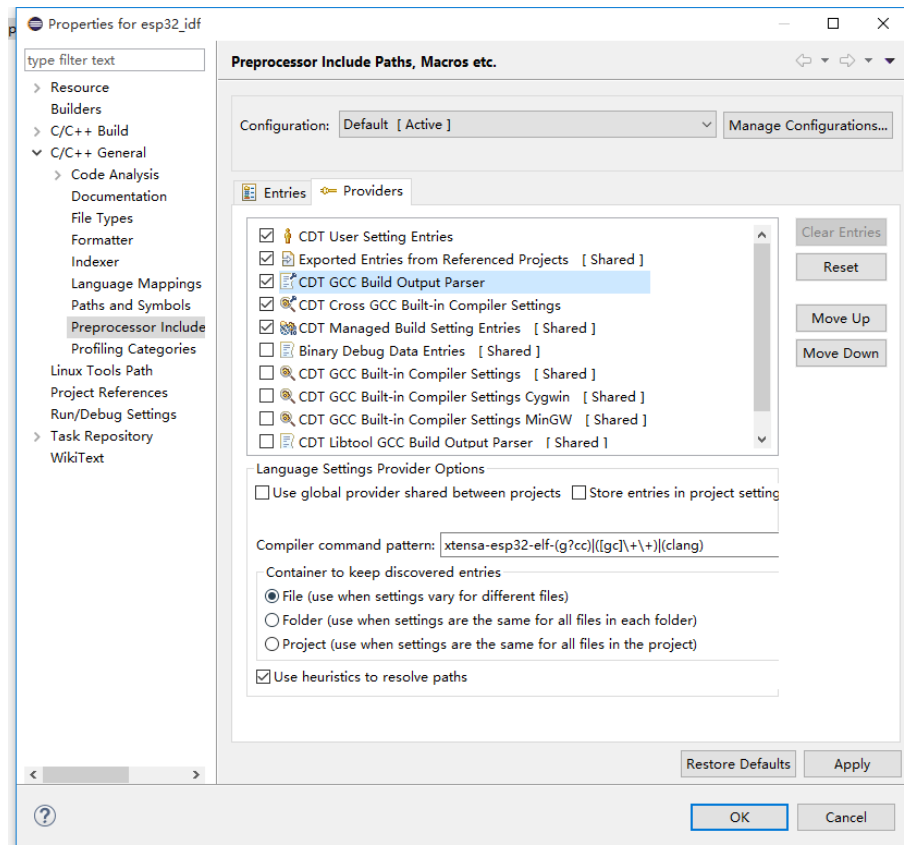
至此编译器选项已经设置完成。

9、进入 "C/C++ General" -> "Preprocessor Include Paths" 选项，然后选择 "Providers" 页面。选择下拉列表中的 "CDT Cross GCC Built-in Compiler Settings"，在下方会出现一个 "Command to get compiler specs" 的可输入窗口，使用 `xtensa-esp32-elf-gcc` 替换 `${COMMAND}`，或者直接输入 `xtensa-esp32-elf-gcc ${FLAGS} -E -P -v -dD "${INPUTS}"`。



10、进入 "C/C++ General" -> "Preprocessor Include Paths" 选项，然后选择"Providers" 页面。选择下拉列表中的 " CDT GCC Build Output Parser"，在 Compiler command pattern 输入框中的开始位置插入 `xtensa-esp32-elf-`。

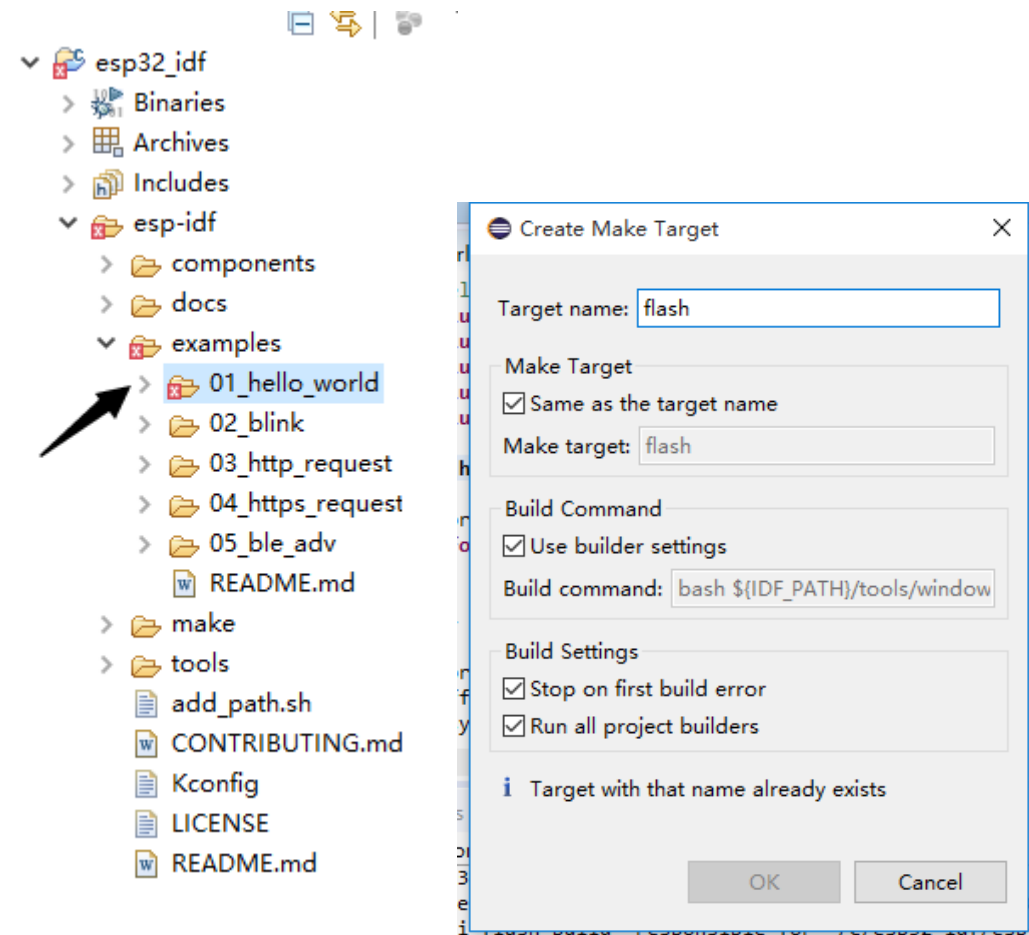
最后改输入框内容为 `xtensa-esp32-elf-(g?cc)|([gc]\+\\+)|(clang)`
单击 OK 结束配置。



环境配置到此终于完成了!

-----华丽的分割线-----

11、右键 `example` 目录下的工程目录 (`01_hello_world`)，选择 `Make Targets->creat`，在弹出的对话框中输入 `flash`，如图所示。输入完成单击 `OK`。

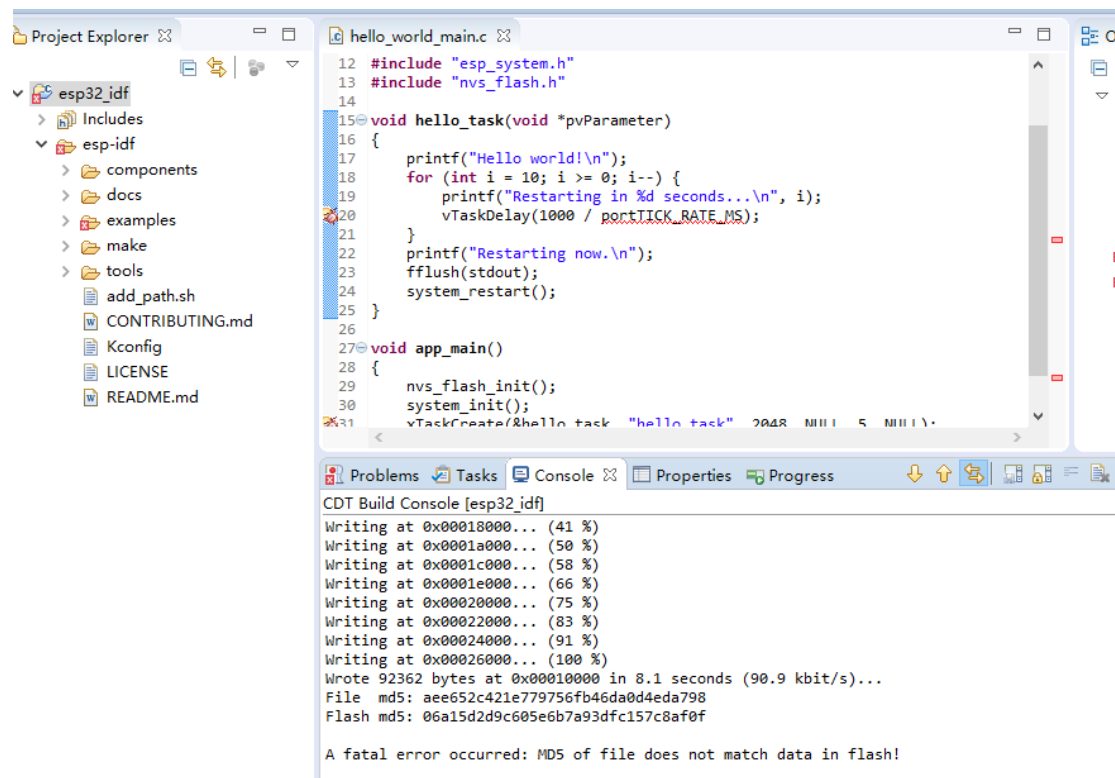


12、工作做到现在已经完成了 90%。先要暂停一下，不要着急着编译，因为你编译也会卡主的。工程编译必须先进行 `config` 才行，否则无法编译通过。这时候要打开 `msys32_shell.cmd`，进入 `01_hello_world` 目录，输入 `make menuconfig`，进行串口设置（详细步骤请参考上一章节内容）

13、确保你的模块已经上电，`GPIO0/DL` 引脚为低电平，按下复位键，使模块进入串口 `boot` 并等待上位机链接，下载程序。

14、就差最后一步了。。。鼠标单击 `esp32_idf` 文件夹，使其处于选中的状态。一定要保证 `esp-idf` 是被选中的！点击菜单栏的 `Project->Make Targets->Build`。在弹出的对话框中选择 `Target` 栏下的 `flash`，然后单击下方的 `Build`，此时就是等待(开始可能会卡一分钟左右)编译器编译完成，并自动下载。

编译完成后会在 `esp-idf-template->build` 目录下生成很多编译输出的文件。我们需要用的是 `bootloader->bootloader.bin`，`app-template.bin` 和 `partitions_singleapp.bin` 三个文件。将这三个文件通过官方提供的下载工具下载至芯片即可



15、等待下载完成后请将 GPIO0/DL 连接至高电平，然后按下复位键即可运行。打开串口调试助手。选择正确的串口，波特率 115200,即可在串口调试助手上看到输出信息。

```

ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
ets Jun  8 2016 00:22:57

rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0x00
clk_drv:0x00, q_drv:0x00, d_drv:0x00, cs0_drv:0x00, hd_drv:0x00, wp_drv:0x00
mode:DIO, clock div:2
load:0x3ffc0000, len:0
load:0x3ffc0000, len:920
load:0x40078000, len:2624
load:0x40098000, len:508
entry 0x40098118
[0:32mI (81) heap_alloc_caps: Initializing heap allocator: [0m
[0:32mI (82) heap_alloc_caps: Region 19: 3FFB41F0 len 0002BE10 tag 0 [0m
[0:32mI (83) heap_alloc_caps: Region 25: 3FFE8000 len 00018000 tag 1 [0m
[0:32mI (93) cpu_start: Pro cpu up. [0m
[0:32mI (98) cpu_start: Starting app cpu, entry point is 0x40080954 [0m
[0:32mI (0) cpu_start: App cpu up. [0m
[0:32mI (114) cpu_start: Pro cpu start user code [0m
rtc v112 Sep 22 2016 16:08:39
XTAL 40M
[0:32mI (150) cpu_start: Starting scheduler on PRO CPU. [0m
[0:32mI (43) cpu_start: Starting scheduler on APP CPU. [0m
frc2_timer_task_hdl:3ffb8c58, prio:22, stack:2048
Hello world!
Restarting in 10 seconds...
Restarting in 9 seconds...

```

16、大功告成！