

上一篇博客寫了在ubuntu下,建立stm32開發環境,程序也已經編譯好生成main.bin,接下來就是要把該文件燒錄到stm32上.在linux下給arm燒錄程序主要使用openocd,這個軟件開源,而且支持眾多芯片,從ARM9到A8都可以,當然STM32也可以.支持的JTAG工具也很多,JLINK ST-LINK OSBDM都可以,我這正好有一個openjtag基於FT2232C的,也是被支持的.

個人原創,轉載請註明原文出處:

<http://blog.csdn.net/embbnux/article/details/17619621>

參考:

[How-to manual Installing a toolchain for Cortex-M3/STM32 on Ubuntu by Peter Seng](#)

## 一 安裝openocd

在ubuntu下安裝openocd

```
[plain] C 8
01. sudo apt-get install openocd
```

也可以到官網下載源碼包自己編譯

## 二安裝openjtag驅動

插上openjtag

```
[plain] C 8
01. user@ubuntu:~/$ lsusb
02. Bus 002 Device 005: ID 093a:2521 Pixart Imaging, Inc.
03. Bus 002 Device 003: ID 1457:5118 First International Computer, Inc. OpenMoko Neo1973 Debug board (V2+)
```

第二個就是了,記下ID 1457:5118

```
[plain] C 8
01. sudo gedit /etc/udev/rules.d/45-ftdi2232-libftdi.rules
```

外

在裡面添加

`SYSFS{idProduct}=="5118", SYSFS{idVendor}=="1457", MODE="666", GROUP="plugdev"`  
權限666,使用openocd就不用sudo了.

```
[plain] C 8
01. sudo /etc/init.d/udev restart
```

拔下在插上就可以了.

## 三使用openocd 連openjtag

外

JTAG接口配置文件openjtag.cfg.根據JTAG設備不同,修改下面

[plain] C 8

```

01. #interface configuration openjtag#####
02. interface ft2232
03. ft2232_device_desc "USB<=>JTAG&RS232"
04. ft2232_layout jtagkey
05. ft2232_vid_pid 0x1457 0x5118

```

可以參考openocd目錄下的文件:/usr/share/openocd/scripts/interface,主要是設備ID以及設備名字,可以通過dmesg | grep usb命令查看.

要燒錄stm32f103就得有這個設備的相關配置,可以查看/usr/share/openocd/scripts/target/stm32f1x.cfg

這裡把兩個文件合在一起openocd.cfg

[plain] C 8

```

01. #daemon configuration#####
02. telnet_port 4444
03. gdb_port 3333
04.
05. #interface configuration openjtag#####
06. interface ft2232
07. ft2232_device_desc "USB<=>JTAG&RS232"
08. ft2232_layout jtagkey
09. ft2232_vid_pid 0x1457 0x5118
10.
11. #board configuration#####
12. # Adjust Work-area size (RAM size) according to MCU in use:
13. #STM32F103RB --> 20KB
14. #set WORKAREASIZE 0x5000
15. #STM32F103ZE --> 64KB
16. set WORKAREASIZE 0x10000
17.
18. #target configuration#####
19. # script for stm32f1x family
20. if { [info exists CHIPNAME] } {
21. set _CHIPNAME $CHIPNAME
22. } else {
23. set _CHIPNAME stm32f1x
24. }
25. if { [info exists ENDIAN] } {
26. set _ENDIAN $ENDIAN
27. } else {
28. set _ENDIAN little
29. }
30. # Work-area is a space in RAM used for flash programming
31. # By default use 16kB
32. if { [info exists WORKAREASIZE] } {
33. set _WORKAREASIZE $WORKAREASIZE
34. } else {
35. set _WORKAREASIZE 0x4000

```

```
36. }
37. # JTAG speed should be <= F_CPU/6. F_CPU after reset is 8MHz, so use F_JTAG = 1MHz
38. adapter_khz 500
39. adapter_nsrst_delay 100
40. jtag_nrst_delay 100
41. #jtag scan chain
42. if { [info exists CPUTAPID] } {
43.     set _CPUTAPID $CPUTAPID
44. } else {
45.     # See STM Document RM0008
46.     # Section 31.6.3
47.     set _CPUTAPID 0x3ba00477
48. }
49. jtag newtap $_CHIPNAME cpu -irlen 4 -ircapture 0x1 -irmask 0xf -expected-id $_CPUTAPID
50. if { [info exists BSTAPID] } {
51.     # FIXME this never gets used to override defaults...
52.     set _BSTAPID $BSTAPID
53. } else {
54.     # See STM Document RM0008
55.     # Section 31.6.2
56.     # Low density devices, Rev A
57.     set _BSTAPID1 0x06412041
58.     # Medium density devices, Rev A
59.     set _BSTAPID2 0x06410041
60.     # Medium density devices, Rev B and Rev Z
61.
62.     set _BSTAPID3 0x16410041
63.     set _BSTAPID4 0x06420041
64.     # High density devices, Rev A
65.     set _BSTAPID5 0x06414041
66.     # Connectivity line devices, Rev A and Rev Z
67.     set _BSTAPID6 0x06418041
68.     # XL line devices, Rev A
69.     set _BSTAPID7 0x06430041
70.     # VL line devices, Rev A and Z In medium-density and high-density value line devices
71.     set _BSTAPID8 0x06420041
72.     # VL line devices, Rev A
73.     set _BSTAPID9 0x06428041
74. }
75. jtag newtap $_CHIPNAME bs -irlen 5 -expected-id $_BSTAPID1 \
76.     -expected-id $_BSTAPID2 -expected-id $_BSTAPID3 \
77.     -expected-id $_BSTAPID4 -expected-id $_BSTAPID5 \
78.     -expected-id $_BSTAPID6 -expected-id $_BSTAPID7 \
79.     -expected-id $_BSTAPID8 -expected-id $_BSTAPID9
80.
81. set _TARGETNAME $_CHIPNAME.cpu
82. target create $_TARGETNAME cortex_m -endian $_ENDIAN -chain-position $_TARGETNAME
83.
84. $_TARGETNAME configure -work-area-phys 0x20000000 -work-area-size $_WORKAREASIZE -work-area-
    backup 0
85.
86. # flash size will be probed
87. set _FLASHNAME $_CHIPNAME.flash
88. flash bank $_FLASHNAME stm32f1x 0x08000000 0 0 0 $_TARGETNAME
89.
90. # if srst is not fitted use SYSRESETREQ to
```

```
91. # perform a soft reset
92. cortex_m reset_config sysresetreq
```

開始燒錄:

< 1 > 在一個終端下執行:

```
[plain] C 8
01. openocd -f openocd.cfg
```

出現:

```
[plain] C 8
01. Open On-Chip Debugger 0.7.0 (2013-05-15-17:28)
02. Licensed under GNU GPL v2
03. For bug reports, read
04.     http://openocd.sourceforge.net/doc/doxygen/bugs.html
05. Info : only one transport option; autoselect 'jtag'
06. adapter speed: 500 kHz
07. adapter_nsrst_delay: 100
08. jtag_ntrst_delay: 100
09. cortex_m3 reset_config sysresetreq
10. Info : clock speed 500 kHz
11. Info : JTAG tap: stm32f1x.cpu tap/device found: 0x3ba00477 (mfg: 0x23b, part: 0xba00, ver:
    0x3)
12. Info : JTAG tap: stm32f1x.bs tap/device found: 0x06414041 (mfg: 0x020, part: 0x6414, ver:
    0x0)
13. Info : stm32f1x.cpu: hardware has 6 breakpoints, 4 watchpoints
```

沒有提示出錯,就表示連接上STM32了.如果出現出錯,就在開發板上按下RESET 鍵復位,查看BOOT0和BOOT1有沒有設置出錯.

< 2 >在另一個終端下,輸入:

```
[plain] C 8
01. telnet localhost 4444
```

依次輸入:

```
[plain] C 8
01. reset halt
02. flash probe 0
03. stm32f1x mass_erase 0
```

```
04. flash write_bank 0 /you_stm32_project_dir/main.bin 0
05. reset run
```

程序就燒好了,按下reset鍵,就開始運行了.

要輸入這麼多命令太麻煩了,寫個perl腳本使它一步運行.

首先安裝perl-telnet

```
[plain] C 8
01. sudo apt-get install libnet-telnet-perl
```

在工程目錄下新建do\_flash.pl文件

```
[plain] C 8
01. #!/usr/bin/perl
02. use Net::Telnet;
03.
04. $numArgs = $#ARGV + 1;
05. if($numArgs != 1){
06.     die( "Usage ./do_flash.pl [main.bin] \n");
07. }
08.
09. $file = $ARGV[0];
10. $ip = "127.0.0.1";
11. $port = 4444;
12. $telnet = new Net::Telnet (
13.     Port => $port,
14.     Timeout=>10,
15.     Errmode=>'die',
16.     Prompt =>'>');
17.
18. $telnet->open($ip);
19.
20. print $telnet->cmd('reset halt');
21. print $telnet->cmd('flash probe 0');
22. print $telnet->cmd('stm32flx mass_erase 0');
23. print $telnet->cmd('flash write_bank 0 '.$file.' 0');
24. print $telnet->cmd('reset halt');
25. print $telnet->cmd('exit');
26.
27. print "\n";
```

在根目錄下的Makefile文件裡面加入這段語句:

```
[plain] C 8
01. flash:all
02.     ./do_flash.pl $(TOP)/main.bin
```

這樣只要,執行make flash就可以直接運行第二步了,方便簡介.

外