

MyBlog of 东隅

Embbnux waits for you !

目录视图

摘要视图

RSS 订阅

个人资料



My 东隅

访问：95346次
积分：1331
等级：BLOG > 4
排名：千里之外

原创：34篇
转载：9篇
译文：0篇
评论：41条

文章搜索

Blog of Embbnux

欢迎访问我的另一个博客:

Home 主页

热荐小说

有爱探案解谜之旅
古代言情志异小说:

《玉兰劫》

文章分类

嵌入式学习 (16)
安卓开发学习 (2)
QT5编程学习笔记 (2)
AVR单片机开发 (6)
一起来玩树莓派 (8)
STM32开发 (3)
web开发 (5)

文章存档

2016年10月 (1)
2016年09月 (1)
2016年05月 (1)
2016年03月 (2)

移动信息安全的漏洞和逆向原理 【观点】世界上最好的语言是什么 Get IT技能知识库，50个领域一键直达

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立

标签：stm32 linux gcc makefile 官方库

2013-12-27 19:49 10948人阅读 评论(16) 分享 举报

分类:

STM32开发 (2)

版权声明：本文为博主原创文章，未经博主允许不得转载。

之前在e络盟的意法半导体掏了一个STM32开发板挺好的,却不想在window下开发,也不想用那么占内存的IAR MDK等软件,所以决定在ubuntu下建立该开发环境,像之前avr Linux一样,找了下资料,国内有人做过,但都没有很详尽的教程,所以花了三四天才完成.其实原理很简单,就是安装适用与STM32的GCC,以及建立该工程,主要是Makefile加上STM32的官方库.

个人原创,转载请注明原文出处:

<http://blog.csdn.net/embbnux/article/details/17616809>

参考:

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

博文新地址转为下面链接,有问题到该地址评论哦:

https://www.embbnux.com/2014/02/01/linux_stm32_gcc_makefile/

博主最近在电脑上自建了博客,以后会更多的用那个了,欢迎关注访问,里面也有很多有用资源:

<http://www.embbnux.com/>

环境:

ubuntu 13.10

stm32f103zet6

一 STM 32 GCC 安装

stm32 属于arm cortex-m系列thumb指令集,所以给arm用的arm-none-eabi就可以了,首先是下载

下载地址:

<https://launchpad.NET/gcc-arm-embedded/+download>

下载其中的gcc-arm-none-eabi-version-linux.tar.bz2

解压到你知道的目录会产生 gcc-arm-none-eabi的文件夹

将该编译器添加到你的环境中:

```
[plain]
01. sudo gedit ~/.bashrc
```

在最后一行添加:

```
[plain]
01. export PATH=$PATH:/your_stm_gcc_dir/gcc-arm-none-eabi-
```



超窄边液晶拼接屏

2014年11月 (1)

展开

阅读排行

ubuntu linux下建立stm32:

(10940)

建立树莓派raspberrypi交叉编译环境:

(7369)

基于树莓派raspberrypi: 移植STM32F103:

(6003)

在树莓派上搭建轻量级博

(5221)

ubuntu linux下建立stm32:

(5077)

把ATmega128开发板转

(4848)

构建ubuntu armv7文件系

(4143)

Qt5 学习笔记 二: UI 编程

(3382)

Linux下Android开发连接

(2998)

基于树莓派Raspberry: 移植STM32F103:

(2940)

评论排行

ubuntu linux下建立stm32:

(16)

构建ubuntu armv7文件系

(8)

在树莓派上搭建轻量级博

(4)

基于树莓派raspberrypi: 移植STM32F103:

(4)

开源个安卓程序: 蓝牙遥

(3)

基于树莓派Raspberry: 移植STM32F103:

(2)

Qt5 学习笔记 二: UI 编程

(2)

把ATmega128开发板转

(1)

建立树莓派raspberrypi交叉编译环境:

(1)

AVR 单片机 ATmega16

(1)

最新评论

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立 - MyBlog of 东隅: 很不错

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立 - MyBlog of 东隅: @FishMike: 直接评论就可

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立 - MyBlog of 东隅: z1282429194: 怎么下载不了stsw-stm32121.zip

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立 - MyBlog of 东隅: FishMike: 新博客怎么登陆, 需要注册么, 我在官方库那一部分遇见问题了, 想请教

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立 - MyBlog of 东隅: 师院小Q: src里面也要有makefilehttps://github.com/embbnu

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立 - MyBlog of 东隅: 师院小Q: make库的时候USART_ITCo...

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立 - MyBlog of 东隅: habbei: 大神, 请问export PATH=\$PATH:/your_stm_gcc_dir/gcc-arm-n...

基于树莓派Raspberry: 移植STM32F103: 字符设备驱动: 感谢分享, 驱动需要和内核一起编译嘛, 然后驱动的makefile 也是一个关键吧

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立 - MyBlog of 东隅: s1987ea: 我做到最后一步 工程主目录, 下make, 提示 make: *** No rule to make ...

ubuntu linux下建立stm32开发环境: GCC安装以及工程Makefile建立 - MyBlog of 东隅: mapblue: 你生成库的时候用了cd && gcc, 之前配置好的包含文件路径不能下传, 编译时出现了找不到头文件的问...

因为我之前有添加过树莓派的编译器了,所以实际上是这样的:

```
[plain]
01. export PATH=$PATH:/your_pi_gcc_dir/tools-master/arm-bcm2708/arm-bcm2708hardfp-linux-gnueabi/bin:/your_stm_gcc_dir/gcc-arm-none-eabi-4_8-2013q4/bin
```

两个编译器环境中间用冒号隔开;
注销后测试:

```
[plain]
01. arm-none-eabi-gcc -v
```

可以查看到该编译器的版本,就表示可以了.

二 工程环境的建立

新建个工程文件夹,及其目录

```
[plain]
01. mkdir stm_project
02. cd stm_project
03. mkdir libs
04. mkdir src
05. mkdir inc
```

下载,安装官方库:

stm32的寄存器不像51 avr等单片机,那么少,自己写文库,背背寄存器就可以了,所以ST公司提供了他们官方的库,为了避免重复造轮子,就直接采用他们的库,库版本为STM32_USB-FS-Device_Lib_V4.0.0,这个库多了usb支持,下载的话到st官网搜索stm32f10x就有了.

下载链接:

[stsw-stm32121.zip](#)

解压,把解压好的文件夹复制到刚才新建的libs里面.

在工程根目录下新建Makefile.common文件,这个为通用makefile

```
[cpp]
01. # include Makefile
02.
03. #This file is included in the general Makefile, the libs Makefile and the src Makefile
04. #Different optimize settings for library and source files can be realized by using arguments
05. #Compiler optimize settings:
06. # -
07. # 00 no optimize, reduce compilation time and make debugging produce the expected results (default)
08. # 01 optimize, reduce code size and execution time, without much increase of compilation time.
09. # -02 optimize, reduce code execution time compared to '01', increase of compilation time.
10. # -03 optimize, turns on all optimizations, further increase of compilation time.
11. # -Os optimize for size, enables all '-O2' optimizations that do not typically increase code size and other code size optimizations.
12. #Recommended optimize settings for release version: -O3
13. #Recommended optimize settings for debug version: -O0
14. #Valid parameters :
15. # OptLIB=0 --> optimize library files using the -O0 setting
16. # OptLIB=1 --> optimize library files using the -O1 setting
17. # OptLIB=2 --> optimize library files using the -O2 setting
18. # OptLIB=s --> optimize library files using the -Os setting
19. # OptSRC=0 --> optimize source files using the -O0 setting
20. # OptSRC=1 --> optimize source files using the -O1 setting
21. # OptSRC=2 --> optimize source files using the -O2 setting
22. # OptSRC=3 --> optimize source files using the -O3 setting
23. # OptSRC=s --> optimize source files using the -Os setting
24. # all --> build all
25. # libs --> build libs only
26. # src --> build src only
27. # clean --> clean project
28. # tshow --> show optimize settings
29. #Example:
30. # make OptLIB=3 OptSRC=0 all tshow
31.
```

关闭





```

32. TOP=$(shell readlink -f "$(dir $(lastword $(MAKEFILE_LIST)))")
33. PROGRAM=main
34. LIBDIR=$(TOP)/libs
35.
36. #Adjust the following line to the library in use
37. #=====add by embnux 根据你的库不同,调整这个地方的库目录地址=====#
38. STMLIB=$(LIBDIR)/STM32_USB-FS-Device_Lib_V4.0.0/Libraries
39. #=====add by embnux 根据你的stm32芯片型号容量不同,修改这个地方的TypeOfMCU=====#
40. #Adjust TypeOfMCU in use, see CMSIS file "stm32f10x.h"#STM32F103RBT (128KB FLASH, 20KB RAM) -
-> STM32F10X_MD#TypeOfMCU=STM32F10X_MD#STM32F103RET (512KB FLASH, 64KB RAM) --
> STM32F10X_HD#STM32F103ZET (512KB FLASH, 64KB RAM) --> STM32F10X_HD
41. #=====#
42. TypeOfMCU=STM32F10X_HD
43. #=====#
44. TC=arm-none-eabi
45. CC=$(TC)-gcc
46. LD=$(TC)-ld -v
47. OBJCOPY=$(TC)-objcopy
48. AR=$(TC)-ar
49. GDB=$(TC)-gdb
50. INCLUDE=-I$(TOP)/inc
51. INCLUDE+=-I$(STMLIB)/CMSIS/Include
52. INCLUDE+=-I$(STMLIB)/CMSIS/Device/ST/STM32F10x/Include
53. INCLUDE+=-I$(STMLIB)/CMSIS/Device/ST/STM32F10x/Source/Templates
54. INCLUDE+=-I$(STMLIB)/STM32F10x_StdPeriph_Driver/inc
55. INCLUDE+=-I$(STMLIB)/STM32_USB-FS-Device_Driver/inc
56. COMMONFLAGS=-g -mcpu=cortex-m3 -mthumb
57. COMMONFLAGSLIB=$(COMMONFLAGS)
58. #Commands for general Makefile and src Makefile
59. ifeq ($(OptSRC),0)
60. COMMONFLAGS+=-O0
61. InfoTextSrc=src (no optimize, -O0)
62. else ifeq ($(OptSRC),1)
63. COMMONFLAGS+=-O1
64. InfoTextSrc=src (optimize time+ size+, -O1)
65. else ifeq ($(OptSRC),2)
66. COMMONFLAGS+=-O2
67. InfoTextSrc=src (optimize time++ size+, -O2)
68. else ifeq ($(OptSRC),s)
69. COMMONFLAGS+=-Os
70. InfoTextSrc=src (optimize size++, -Os)
71. else
72. COMMONFLAGS+=-O3
73. InfoTextSrc=src (full optimize, -O3)
74. endif
75. CFLAGS+=$(COMMONFLAGS) -Wall -Werror $(INCLUDE)
76. CFLAGS+=-D $(TypeOfMCU)
77. CFLAGS+=-D VECT_TAB_FLASH
78.
79. #Commands for libs Makefile
80. ifeq ($(OptLIB),0)
81. COMMONFLAGSLIB+=-O0
82. InfoTextLib=libs (no optimize, -O0)
83. else ifeq ($(OptLIB),1)
84. COMMONFLAGSLIB+=-O1
85. InfoTextLib=libs (optimize time+ size+, -O1)
86. else ifeq ($(OptLIB),2)
87. COMMONFLAGSLIB+=-O2
88. InfoTextLib=libs (optimize time++ size+, -O2)
89. else ifeq ($(OptLIB),s)
90. COMMONFLAGSLIB+=-Os
91. InfoTextLib=libs (optimize size++, -Os)
92. else
93. COMMONFLAGSLIB+=-O3
94. InfoTextLib=libs (full optimize, -O3)
95. endif
96. CFLAGSLIB+=$(COMMONFLAGSLIB) -Wall -Werror $(INCLUDE)
97. CFLAGSLIB+=-D $(TypeOfMCU)
98. CFLAGSLIB+=-D VECT_TAB_FLASH

```

编译库文件:

进入libs文件夹,新建Makefile:

关闭



```

01. # libs Makefile
02. include ../Makefile.common
03. LIBS+=libstm32.a
04. CFLAGSlib+=-c
05.
06. all: libs
07.
08. libs: $(LIBS)
09.
10. libstm32.a:
11.     @echo -n "Building $@ ..."
12.     @cd $(STMLIB)/CMSIS/Device/ST/STM32F10x/Source/Templates && \
13.         $(CC) $(CFLAGSlib) \
14.             system_stm32f10x.c
15.     @cd $(STMLIB)/STM32F10x_StdPeriph_Driver/src && \
16.         $(CC) $(CFLAGSlib) \
17.             -D"assert_param(expr)=((void)0)" \
18.             -I../CMSIS/Include \
19.             -I../CMSIS/Device/ST/STM32F10x/Include \
20.             -I../inc \
21.             *.c
22. # @cd $(STMLIB)/STM32_USB-FS-Device_Driver/src && \
23. # $(CC) $(CFLAGSlib) \
24. # -D"assert_param(expr)=((void)0)" \
25. # -I../CMSIS/Include \
26. # -I../CMSIS/Device/ST/STM32F10x/Include \
27. # -I../inc \
28. # *.c
29. @$ (AR) cr $(LIBDIR)/$@ \
30.     $(STMLIB)/CMSIS/Device/ST/STM32F10x/Source/Templates/system_stm32f10x.o \
31.     $(STMLIB)/STM32F10x_StdPeriph_Driver/src/*.o \
32.     $(STMLIB)/STM32_USB-FS-Device_Driver/src/*.o
33.     @echo "done."
34. .PHONY: libs clean tshow
35.
36. clean:
37.     rm -f $(STMLIB)/CMSIS/Device/ST/STM32F10x/Source/Templates/system_stm32f10x.o
38.     rm -f $(STMLIB)/STM32F10x_StdPeriph_Driver/src/*.o
39.     rm -f $(STMLIB)/STM32_USB-FS-Device_Driver/src/*.o
40.     rm -f $(LIBS)
41. tshow:
42.     @echo "#####"
43.     @echo "##### optimize settings: $(InfoTextLib), $(InfoTextSrc)"
44.     @echo "#####"

```

编译该库:

```

[plain]
01. make clean
02. make

```

就会在lib目录下生成libstm32.a,这个就是编译好的静态库了.

建立工程编译ld文件

这个ld文件,为在编译时告诉编译器把代码放到什么地址,根据芯片的内存以及flash容量不同有所调整

在工程根目录下新建linker.ld文件

代码较长,请到我的资源里面下载,或者查看参考pdf里面的:

<http://download.csdn.net/detail/canyue102/6778837>

这里说明需要修改的地方,根据芯片型号不同,选择相应的RAM FLASH大小

关闭

```

[css]
01. MEMORY {
02.     /*Adust LENGTH to RAMsize of target MCU:*/
03.     /*STM32F103RBT --> 20K*/
04.     /*RAM (RWX) : ORIGIN = 0x20000000 , LENGTH = 20K*/
05.     /*STM32F103RET --> 64K*/
06.     /*STM32F103ZET --> 64K*/
07.     RAM (RWX) : ORIGIN = 0x20000000 , LENGTH = 64K
08.     EXTSRAM (RWX) : ORIGIN = 0x68000000 , LENGTH = 0
09.     /*Adust LENGTH to (FLASHsize - FeePROMsize) of target MCU*/
10.     /*STM32F103RBT --> 126K*/
11.     FLASH (RX) : ORIGIN = 0x08000000 , LENGTH = 126K

```



```

12.      /*STM32F103RET --> 508K*/
13.      /*FLASH (RX) : ORIGIN = 0x08000000 , LENGTH = 508K*/
14.      /*STM32F103ZET --> 508K*/
15.      FLASH (RX) : ORIGIN = 0x08000000 , LENGTH = 508K
16.      /*Adust ORIGIN to (0x08000000 + (FLASHsize-FeePROMsize)) of target MCU*/
17.      /*and adust LENGTH to FeePROMsize allocated:*/
18.      /*STM32F103RBT --> 0x08000000+126K, 2K*/
19.      EEMUL (RWX) : ORIGIN = 0x08000000+126K, LENGTH = 2K
20.      /*STM32F103RET --> 0x08000000+508K, 4K*/
21.      /*EEMUL (RWX) : ORIGIN = 0x08000000+508K, LENGTH = 4K*/
22.  }

```

在工程根目录下新建 **Makefile** 文件:

```

[plain]
01. # general Makefile
02.
03. include Makefile.common
04. LDFLAGS=$(COMMONFLAGS) -fno-exceptions -ffunction-sections -fdata-sections -L$(LIBDIR) -
nostartfiles -Wl,--gc-sections,-Tlinker.ld
05.
06. LDLIBS+=-lm
07. LDLIBS+=-lstm32
08.
09. STARTUP=startup.c
10.
11. all: libs src
12.     $(CC) -o $(PROGRAM).elf $(LDFLAGS) \
13.         -Wl,--whole-archive \
14.             src/app.a \
15.         -Wl,--no-whole-archive \
16.             $(LDLIBS)
17.     $(OBJCOPY) -O ihex $(PROGRAM).elf $(PROGRAM).hex
18.     $(OBJCOPY) -O binary $(PROGRAM).elf $(PROGRAM).bin
19. #Extract info contained in ELF to readable text-files:
20.     arm-none-eabi-readelf -a $(PROGRAM).elf > $(PROGRAM).info_elf
21.     arm-none-eabi-size -d -B -t $(PROGRAM).elf > $(PROGRAM).info_size
22.     arm-none-eabi-objdump -S $(PROGRAM).elf > $(PROGRAM).info_code
23.     arm-none-eabi-nm -t d -S --size-sort -s $(PROGRAM).elf > $(PROGRAM).info_symbol
24.
25. .PHONY: libs src clean tshow
26.
27. libs:
28.     $(MAKE) -C libs $@
29. src:
30.     $(MAKE) -C src $@
31. clean:
32.     $(MAKE) -C src $@
33.     $(MAKE) -C libs $@
34.     rm -
35. f $(PROGRAM).elf $(PROGRAM).hex $(PROGRAM).bin $(PROGRAM).info_elf $(PROGRAM).info_size
36.     rm -f $(PROGRAM).info_code
37.     rm -f $(PROGRAM).info_symbol
38. tshow:
39.     @echo "#####
40.     @echo "##### optimize settings: $(InfoTextLib), $(InfoTextSrc)"
41.     @echo "#####

```

差不多就好了,在src里面添加测试源码

主要是startup.c 以及main.c,这里就不在说明了,可以查看该pd

<http://download.csdn.net/detail/canyue102/6778885>

然后进入工程主目录,下make就好了.

```

[plain]
01. make clean
02. make OptLIB=0 OptSRC=0 all tshow

```

然后,就完成了,关于ubuntu下烧录程序到stm32下,请见下一篇

关闭



顶

9

踩

0

上一篇

基于树莓派raspberry: 移植 2.4寸TFT显示屏以及源码分析

下一篇

ubuntu linux下建立stm32开发环境: 程序烧录 openocd+openjtag


我的同类文章

STM32开发（2）

• 对STM32官方库封装一:GP... 2014-01-30 阅读 1568


• ubuntu linux下建立stm32... 2013-12-27 阅读 5084

参考知识库



.NET知识库

1396 关注 | 800 收录



Linux知识库

5726 关注 | 3268 收录

猜你在找

- 零基础玩转Linux+Ubuntu(嵌入式开发基础课程)
- ubuntu1204 64建立交叉编译环境binarm-none-linux-
- Linux快速学习以及监控分析实战【小强测试出品】
- linux 安装stm32开发环境
- 嵌入式Linux开发基础
- stm32库函数在gcc下的开发环境配置完成
- 《C语言/C++学习指南》Linux开发篇
- STM32的Eclipse开发环境配置采用GCC编译器
- Linux环境C语言编程基础
- GTK编程linux下GTK开发环境建立ubuntu

广告


查看评论

12楼 popebl 4天前 14:41发表




很不错

11楼 z1282429194 2016-06-11 01:03发表



怎么下载不了 stsw-stm32121.zip

10楼 FishMike 2016-05-30 21:05发表



新博客怎么登陆，需要注册么，我在官方库那一部分遇见问题了，想请教

Re: My东隅 2016-08-02 11:09发表



回复FishMike：直接评论就可

9楼 师院小Q 2016-03-30 11:40发表




src里面也要有makefile

https://github.com/embbnux/stm32_development_on_linux

楼主的工程，可供下载

8楼 师院小Q 2016-02-15 15:06发表



make库的时候

stm32f10x_usart.c: In function 'USART_ITConfig':

stm32f10x_usart.c:373:3: error: dereferencing type-punned pointer with 'aliasing']



关闭





usartxbase = (*(uint32_t*)&(USARTx));
^


是什么问题

7楼 habbei 2015-12-13 19:21发表



大神，请问export PATH=\$PATH:/your_stm_gcc_dir/gcc-arm-none-eabi-4_8-2013q4/bin这句话中your_stm_gcc_dir是指什么啊？我试了用文件夹的路径，但是用不了

6楼 s1987ea 2015-06-24 17:22发表




我做到 最后一步 工程主目录,下make，提示 make[1]: *** No rule to make target `src'. Stop.
是不是在src文件夹下少建了一个 makefile？

5楼 maplblue 2015-05-29 16:51发表



你生成库的时候用了cd && gcc,之前配置好的包含文件路径不能下传，编译时出现了找不到头文件的问题。

4楼 Hlgeo 2014-09-25 13:10发表



我从官网下载的库怎么感觉跟你的不一样啊，你库文件的链接跳了，能不能给发一份？

3楼 dar_k_night 2014-08-05 17:13发表



[yongqi@localhost libs]\$ make clean
./Makefile.common:72: *** missing separator. Stop.

好像有错误

Re: My东隅 2014-08-13 19:41发表




回复dar_k_night: 可能我代码发到网上格式有错，你仔细看下

2楼 dingwBLOG 2014-05-05 12:08发表



很轻大啊！Ubuntu用64位还是用32位

Re: My东隅 2014-05-06 16:04发表



回复dingwBLOG: 32和64都可以，64位的要换64位的gcc，具体见我新博客

1楼 A_A6ss 2014-03-16 11:15发表



看完继续学习 希望楼主有时间可以指点下

Re: My东隅 2014-03-18 19:06发表



回复A_A6ss: 好的嘛,欢迎到我的新博客http://www.embbnux.com
提问分享

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker

OpenStack

VPN

Spark

ERP

IE10

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC

WAP

jQuery

BI

HTML5

Spring

Apache

.NET

API

HTML

SDK

IIS

Fedora

XML

LBS

Unity

Splashtop

UML

components

Windows Mobile

Rails

QEMU

KDE

Cassandra

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

SpringSide

Maemo

Compuware

大数据

aptech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

HBase

Pure

Solr

Angular

Cloud Foundry

Redis

Scala

Django


Bootstrap

关闭

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved



超窄边液晶拼接屏



http://blog.csdn.net/embbnux/article/details/17616809

7/7