買了一個STM32開發板,卻不想在window下開發,也不想用那麼佔內存的IAR MDK等軟件,所以決定在ubuntu下建立該開發環境,像之前avr linux一樣,找了下資料,國內有人做過,但都沒有很詳盡的教程,所以花了三四天才完成.其實原理很簡單,就是安裝適用與STM32的GCC,以及建立該工程,主要是Makefile加上STM32的官方庫.

個人原創,轉載請註明原文出處:

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

參考:

How-to manual Installing a toolchain for Cortex-M3/STM32 on Ubuntu by Peter Seng

環境:

ubuntu 13.10

stm32f103zet6

一 STM 32 GCC 安裝

stm32 屬於arm cortex-m系列thumb指令集,所以給arm用的arm-none-eabi就可以了,首先是下載

下載地址:

https://launchpad.net/gcc-arm-embedded/+download

下載其中的gcc-arm-none-eabi-version-linux.tar.bz2

解壓到你知道的目錄會產生 gcc-arm-none-eabi的文件夾

把該編譯器添加到你的環境中:

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

在最後一行添加:

```
[plain]
01. export PATH=$PATH:/your_stm_gcc_dir/gcc-arm-none-eabi-4_8-2013q4/bin
```

因為我之前有添加過樹莓派的編譯器了,所以實際上是這樣的:

```
[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
```

兩個編譯器環境中間用冒號隔開;

註銷後測試:

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

可以查看到該編譯器的版本,就表示可以了.

二 工程環境的建立

新建個工程文件夾,及其目錄

```plain
mkdir stm_project
cd stm_project
mkdir libs
mkdir src
mkdir inc
```

下載,安裝官方庫:

stm32的寄存器不像51 avr等單片機,那麼少,自己寫寫庫,背背寄存器就可以了,所以ST公司提供了他們官方的庫,為了避免重複造輪子,就直接採用他們的庫,庫版本為STM32_USB-FS-Device_Lib_V4.0.0,這個庫多了usb支持,下載的話到st官網搜索stm32f10x就有了.

下載鏈接:

stsw-stm32121.zip

解壓,把解壓好的文件夾複製到剛才新建的libs裡面.

在工程根目錄下新建Makefile.common文件,這個為通用makefile

```cpp
# include Makefile

#This file is included in the general Makefile, the libs Makefile and the src Makefile
#Different optimize settings for library and source files can be realized by using arguments
#Compiler optimize settings:
# -O0 no optimize, reduce compilation time and make debugging produce the expected results (default).
# -O1 optimize, reduce code size and execution time, without much increase of compilation time.
# -O2 optimize, reduce code execution time compared to 'O1', increase of compilation time.
# -O3 optimize, turns on all optimizations, further increase of compilation time.
# -Os optimize for size, enables all '-O2' optimizations that do not typically increase code size and other code size optimizations.
#Recommended optimize settings for release version: -O3
#Recommended optimize settings for debug version: -O0
#Valid parameters :
# OptLIB=0 --> optimize library files using the -O0 setting
# OptLIB=1 --> optimize library files using the -O1 setting
# OptLIB=2 --> optimize library files using the -O2 setting
# OptLIB=3 --> optimize library files using the -O3 setting
# OptLIB=s --> optimize library files using the -Os setting
# OptSRC=0 --> optimize source files using the -O0 setting
# OptSRC=1 --> optimize source files using the -O1 setting
# OptSRC=2 --> optimize source files using the -O2 setting
# OptSRC=3 --> optimize source files using the -O3 setting
# OptSRC=s --> optimize source files using the -Os setting
# all --> build all
```

```
# libs --> build libs only
# src --> build src only
# clean --> clean project
# tshow --> show optimize settings
#Example:
# make OptLIB=3 OptSRC=0 all tshow

TOP=$(shell readlink -f  "$(dir $(lastword $(MAKEFILE_LIST)))" )
PROGRAM=main
LIBDIR=$(TOP)/libs

#Adust the following line to the library in use
#=========add by embbnux 根據你的庫不同,調整這個地方的庫目錄地址==================#
 STMLIB=$(LIBDIR)/STM32_USB-FS-Device_Lib_V4.0.0/Libraries
#=========add by embbnux 根據你的stm32芯片型號容量不同,修改這個地方的TypeOfMCU=======#
#Adjust TypeOfMCU in use, see CMSIS file "stm32f10x.h"#STM32F103RBT (128KB FLASH, 20KB RAM) --
> STM32F10X_MD#TypeOfMCU=STM32F10X_MD#STM32F103RET (512KB FLASH, 64KB RAM) -->
STM32F10X_HD#STM32F103ZET (512KB FLASH, 64KB RAM) --> STM32F10X_HD
#============================================== ==========================#
TypeOfMCU=STM32F10X_HD
#============================================== ==========================#
TC=arm-none-eabi
CC=$(TC)-gcc
LD=$(TC)-ld -v
OBJCOPY=$(TC)-objcopy
AR=$(TC)-ar
GDB=$(TC)-gdb
INCLUDE=-I$(TOP)/inc
INCLUDE+=-I$(STMLIB)/CMSIS/Include
INCLUDE+=-I$(STMLIB)/CMSIS/Device/ST/STM32F10x/Include
INCLUDE+=-I$(STMLIB)/CMSIS/Device/ST/STM32F10x/Source/Templates
INCLUDE+=-I$(STMLIB)/STM32F10x_StdPeriph_Driver/inc
INCLUDE+=-I$(STMLIB)/STM32_USB-FS-Device_Driver/inc
COMMONFLAGS=-g -mcpu=cortex-m3 -mthumb
COMMONFLAGSlib=$(COMMONFLAGS)
#Commands for general Makefile and src Makefile
ifeq ($(OptSRC),0)
    COMMONFLAGS+=-O0
    InfoTextSrc=src (no optimize, -O0)
else  ifeq ($(OptSRC),1)
    COMMONFLAGS+=-O1
    InfoTextSrc=src (optimize time+ size+, -O1)
else  ifeq ($(OptSRC),2)
    COMMONFLAGS+=-O2
    InfoTextSrc=src (optimize time++ size+, -O2)
else  ifeq ($(OptSRC),s)
    COMMONFLAGS+=-Os
    InfoTextSrc=src (optimize size++, -Os)
else
    COMMONFLAGS+=-O3
    InfoTextSrc=src (full optimize, -O3)
endif
CFLAGS+=$(COMMONFLAGS) -Wall -Werror $(INCLUDE)
CFLAGS+=-D $(TypeOfMCU)
CFLAGS+=-D VECT_TAB_FLASH

```

```
#Commands for libs Makefile
ifeq ($(OptLIB),0)
    COMMONFLAGSlib+=-O0
    InfoTextLib=libs (no optimize, -O0)
else  ifeq ($(OptLIB),1)
    COMMONFLAGSlib+=-O1
    InfoTextLib=libs (optimize time+ size+, -O1)
else  ifeq ($(OptLIB),2)
    COMMONFLAGSlib+=-O2
    InfoTextLib=libs (optimize time++ size+, -O2)
else  ifeq ($(OptLIB),s)
    COMMONFLAGSlib+=-Os
    InfoTextLib=libs (optimize size++, -Os)
else
    COMMONFLAGSlib+=-O3
    InfoTextLib=libs (full optimize, -O3)
endif
CFLAGSlib+=$(COMMONFLAGSlib) -Wall -Werror $(INCLUDE)
CFLAGSlib+=-D $(TypeOfMCU)
CFLAGSlib+=-D VECT_TAB_FLASH
```

編譯庫文件:

進入libs文件夾,新建Makefile:

[cpp]

```
# libs Makefile
include ../Makefile.common
LIBS+=libstm32.a
CFLAGSlib+=-c

all: libs

libs: $(LIBS)

libstm32.a:
    @echo -n  "Building $@ ..."
    @cd $(STMLIB)/CMSIS/Device/ST/STM32F10x/Source/Templates && \
        $(CC) $(CFLAGSlib) \
            system_stm32f10x.c
    @cd $(STMLIB)/STM32F10x_StdPeriph_Driver/src && \
        $(CC) $(CFLAGSlib) \
            -D "assert_param(expr)=((void)0)"  \
            -I../../CMSIS/Include \
            -I../../CMSIS/Device/ST/STM32F10x/Include \
            -I../inc \
            *.c
# @cd $(STMLIB)/STM32_USB-FS-Device_Driver/src && \
# $(CC) $(CFLAGSlib) \
# -D"assert_param(expr)=((void)0)" \
# -I../../CMSIS/Include \
# -I../../CMSIS/Device/ST/STM32F10x/Include \
# -I../inc \
```

```
# *.c
    @$(AR) cr $(LIBDIR)/$@ \
        $(STMLIB)/CMSIS/Device/ST/STM32F10x/Source/Templates/system_stm32f10x.o \
        $(STMLIB)/STM32F10x_StdPeriph_Driver/src/*.o \
# $(STMLIB)/STM32_USB-FS-Device_Driver/src/*.o
        @echo  "done."
.PHONY: libs clean tshow

clean:
    rm -f $(STMLIB)/CMSIS/Device/ST/STM32F10x/Source/Templates/system_stm32f10x.o
    rm -f $(STMLIB)/STM32F10x_StdPeriph_Driver/src/*.o
    rm -f $(STMLIB)/STM32_USB-FS-Device_Driver/src/*.o
    rm -f $(LIBS)
tshow:
    @echo  "################################################
################################################## #####"
    @echo  "################# optimize settings: $(InfoTextLib), $(InfoTextSrc)"
    @echo  "################################################
################################################## #####"
```

編譯該庫:

```
[plain]
make clean
make
```

就會在lib目錄下生成libstm32.a,這個就是編譯好的靜態庫了.

建立工程編譯ld文件

這個ld文件,為在編譯時告訴編譯器把代碼放到什麼地址,根據芯片的內存以及flash容量不同有所調整

在工程根目錄下新建linker.ld文件

代碼較長,請到我的資源里面下載,或者查看參考pdf裡面的:

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

這裡說明需要修改的地方,根據芯片型號不同,選擇相應的RAM FLASH大小

```
[css]
MEMORY {
    /*Adust LENGTH to RAMsize of target MCU:*/
    /*STM32F103RBT --> 20K*/
    /*RAM (RWX) : ORIGIN = 0x20000000 , LENGTH = 20K*/
    /*STM32F103RET --> 64K*/
    /*STM32F103ZET --> 64K*/
    RAM (RWX) : ORIGIN =  0x20000000  , LENGTH =  64 K
    EXTSRAM (RWX) : ORIGIN =  0x68000000  , LENGTH =  0
    /*Adust LENGTH to (FLASHsize - FeePROMsize) of target MCU:*/
    /*STM32F103RBT --> 126K*/
    FLASH (RX) : ORIGIN =  0x08000000  , LENGTH =  126 K
    /*STM32F103RET --> 508K*/
    /*FLASH (RX) : ORIGIN = 0x08000000 , LENGTH = 508K*/
    /*STM32F103ZET --> 508K*/
    FLASH (RX) : ORIGIN =  0x08000000  , LENGTH =  508 K
```

```
    /*Adust ORIGIN to (0x08000000 + (FLASHsize-FeePROMsize)) of target MCU*/
    /*and adust LENGTH to FeePROMsize allocated:*/
    /*STM32F103RBT --> 0x08000000+126K, 2K*/
    EEMUL (RWX) : ORIGIN =  0x08000000 + 126 K, LENGTH =  2 K
    /*STM32F103RET --> 0x08000000+508K, 4K*/
    /*EEMUL (RWX) : ORIGIN = 0x08000000+508K, LENGTH = 4K*/
}
```

在工程根目錄下新建Makefile文件:

[plain]

```
# general Makefile

include Makefile.common
LDFLAGS=$(COMMONFLAGS) -fno-exceptions -ffunction-sections -fdata-sections -L$(LIBDIR) -nostartfiles -Wl,--gc-sections,-Tlinker.ld

LDLIBS+=-lm
LDLIBS+=-lstm32

STARTUP=startup.c

all: libs src
    $(CC) -o $(PROGRAM).elf $(LDFLAGS) \
        -Wl,--whole-archive \
            src/app.a \
        -Wl,--no-whole-archive \
            $(LDLIBS)
    $(OBJCOPY) -O ihex $(PROGRAM).elf $(PROGRAM).hex
    $(OBJCOPY) -O binary $(PROGRAM).elf $(PROGRAM).bin
#Extract info contained in ELF to readable text-files:
    arm-none-eabi-readelf -a $(PROGRAM).elf > $(PROGRAM).info_elf
    arm-none-eabi-size -d -B -t $(PROGRAM).elf > $(PROGRAM).info_size
    arm-none-eabi-objdump -S $(PROGRAM).elf > $(PROGRAM).info_code
    arm-none-eabi-nm -td -S --size-sort -s $(PROGRAM).elf > $(PROGRAM).info_symbol

.PHONY: libs src clean tshow

libs:
    $(MAKE) -C libs $@
src:
    $(MAKE) -C src $@
clean:
    $(MAKE) -C src $@
    $(MAKE) -C libs $@
    rm -f $(PROGRAM).elf $(PROGRAM).hex $(PROGRAM).bin $(PROGRAM).info_elf $(PROGRAM).info_size
    rm -f $(PROGRAM).info_code
    rm -f $(PROGRAM).info_symbol
tshow:
        @echo "################################################ ##################################################### #####"
        @echo "################ optimize settings: $(InfoTextLib), $(InfoTextSrc)"
        @echo "################################################"
```

```
################################################ #####"
```

差不多就好了,在src裡面添加測試源碼

主要是startup.c 以及main.c,這裡就不在說明了,可以查看該pdf或者到我的資源下載

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

然後進入工程主目錄,下make就好了.

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

然後,就完成了,關於ubuntu下燒錄程序到stm32下,請見下一篇博客