



合肥工业大学

嵌入式系统原理

实 验 指 导 书

合肥工业大学计算机与信息学院

《嵌入式系统原理》课程组

2021年5月

目 录

实验一 熟悉 Linux 开发环境	1
一、实验目的	1
二、实验内容	1
三、预备知识	1
四、实验设备及工具（包括软件调试工具）	1
五、实验步骤	1
六、实验修改要求	6
实验二 汇编点亮 LED	7
一、实验目的	7
二、实验内容	7
三、预备知识	7
四、实验设备及工具（包括软件调试工具）	7
五、实验原理	7
六、实验步骤	8
七、实验修改要求	12
实验三 查询方式检测按键	13
一、实验目的	13
二、实验内容	13
三、预备知识	13
四、实验设备及工具（包括软件调试工具）	13
五、实验原理	13
六、实验步骤	15
七、实验修改要求	15
实验四 PWM 定时器实验	16
一、实验目的	16
二、实验内容	16
三、预备知识	16
四、实验设备及工具（包括软件调试工具）	16
五、实验原理	16
六、实验步骤	17
七、实验修改要求	18

实验一 熟悉 Linux 开发环境

一、实验目的

熟悉Linux开发环境，学会基于Tiny6410的Linux开发环境的配置和使用。
使用Linux的arm-Linux-gcc编译，并使用minicom串口方式下载和调试程序。

二、实验内容

本次实验使用Fedora 9.0操作系统环境, 安装ARM-Linux的开发库及编译器。创建一个新目录，并在其中编写hello.c和Makefile文件。学习在Linux下的编程和编译过程，以及ARM开发板的使用和开发环境的设置。下载已经编译好的文件到目标开发板上运行。

三、预备知识

1. 有C语言基础。
2. 会使用Linux下常用的编辑器。
3. 掌握Makefile的编写和使用。
4. 了解Linux下的编译程序与交叉编译的过程。

四、实验设备及工具（包括软件调试工具）

硬件：Tiny6410嵌入式实验平台。

软件：PC机操作系统Fedora9+MINICOM+ARM-Linux 开发环境

五、实验步骤

- 1、登录win7系统，使用管理员权限打开VMware虚拟机软件。确认虚拟机中已安装Fedora系统，否则请通过镜像文件安装。
- 2、确认虚拟机串口已打开，否则通过【编辑此虚拟机】选项，使用“添加”功能添加串行通信端口，并确认使用物理串行端口中为“自动选择串口”选项。
- 3、通过【启动此虚拟机】，启动Fedora系统，选择knight用户，输入登录密码

knight。

4、打开终端（terminal），建立hello工作目录（注：若已存在hello目录，则跳过该步骤。可通过ls命令查看是否存在hello目录。）

```
[root@zxt smile]# mkdir hello
[root@zxt smile]# cd hello
```

此时新建的hello工作目录，会在home目录下出现，说明此次操作成功（**注意，记清楚所创建目录的位置**），如下图所示：

```
[root@localhost home]# mkdir hello
[root@localhost home]# ls
hello rdy
[root@localhost home]# cd hello
```

5、编写程序源代码

hello.c 源代码较简单，如下：

```
#include <stdio.h>

int main()
{
    printf("Hello,World!\n");

    return 0;
}
```

可以用下面的命令来编写hello.c的源代码，进入hello目录使用vim命令来编辑代码（也可以使用gedit命令来编辑hello.c文件：gedit hello.c）：

```
[root@zxt hello]# vi hello.c
```

按“i”或者“a”进入编辑模式，将上面的代码录入进去，**完成后按Esc键进入命令状态**，再用命令**“:wq”保存并退出**。这样便在当前目录下建立了一个名为hello.c的文件。

6、编写Makefile

要使上面的hello.c程序能够运行，必须要编写一个Makefile文件，Makefile文件定义了一系列的规则，它指明了哪些文件需要编译，哪些文件需

要先编译，哪些文件需要重新编译等等更为复杂的命令。使用它带来的好处就是自动编译，只需要敲一个“make”命令整个工程就可以实现自动编译，当然本次实验只有一个文件，它还不能体现出使用Makefile的优越性，但当工程比较大文件比较多时，不使用Makefile几乎是不可能的。下面介绍本次实验用到的Makefile文件。

```
CC=arm-linux-gcc
EXEC=hello
OBS=hello.o
CFLAGS+=
LDFLAGS+=

all:$(EXEC)
$(EXEC):$(OBS)
    $(CC) $(LDFLAGS) -o $@ $(OBS)

clean:
    rm -f $(EXEC) *.elf *.gdb *.o
```

★注意：“\$(CC) \$(LDFLAGS) -o \$@ \$(OBS)”和“-rm -f \$(EXEC) *.elf *.gdb *.o”前空白由一个Tab制表符生成，不能单纯由空格来代替。

与上面编写hello.c的过程类似，用vim来创建一个Makefile文件并将代码录入其中。

```
[root@zxt hello]# vi Makefile
```

7、编译应用程序

在上面的步骤完成后，就可以在hello目录下运行“make”来编译程序。如果进行了修改，重新编译则运行：

```
[root@zxt hello]# make clean
[root@zxt hello]# make
```

★注意：编译、修改程序都是在宿主机（本地PC 机）上进行，不能在minicom下进行。

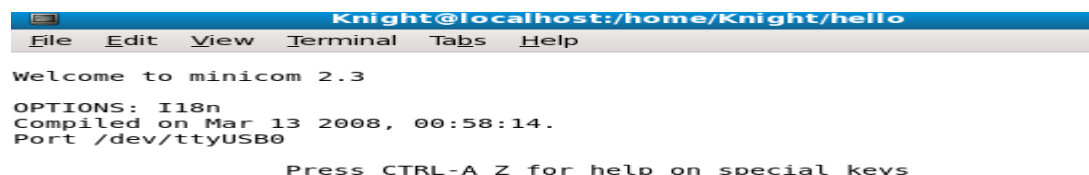
8、下载调试

① 进入root——若当前已经是root用户，则此步跳过。

```
[Knight@localhost hello]$ su root
Password:
[root@localhost hello]#
```

终端输入su root，再输入密码knight即可。

② 使用串口线连接开发板的串口（com0）和PC机的串口。在终端中输入minicom命令，若提示串口不存在，则修改minicom的相关参数后，再执行minicom命令。成功打开串口后，出现下图

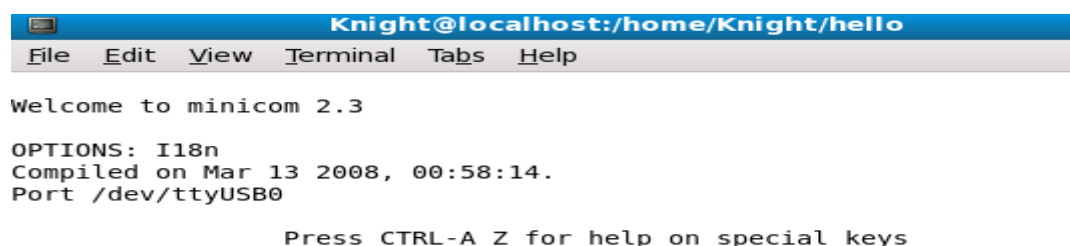


```
Knight@localhost:/home/Knight/hello
File Edit View Terminal Tabs Help

Welcome to minicom 2.3
OPTIONS: I18n
Compiled on Mar 13 2008, 00:58:14.
Port /dev/ttyUSB0

Press CTRL-A Z for help on special keys
```

回车后，出现下图



```
Knight@localhost:/home/Knight/hello
File Edit View Terminal Tabs Help

Welcome to minicom 2.3
OPTIONS: I18n
Compiled on Mar 13 2008, 00:58:14.
Port /dev/ttyUSB0

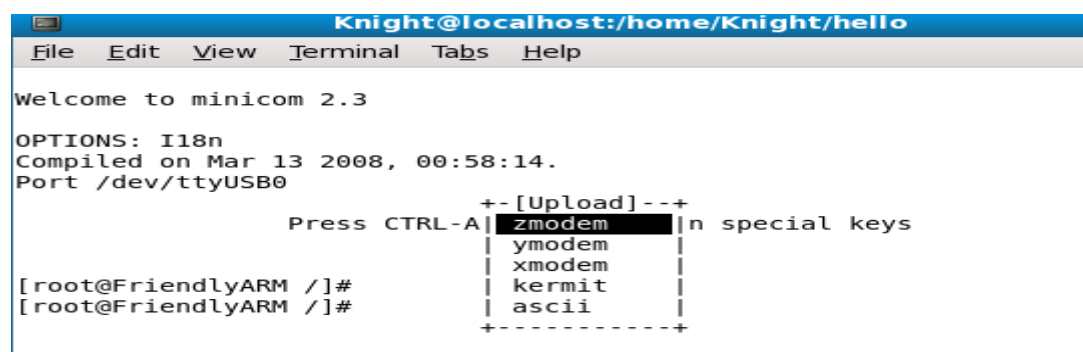
Press CTRL-A Z for help on special keys
```

```
[root@FriendlyARM /]#
[root@FriendlyARM /]#
```

★此时才能进行正常的下载，若没有出现，表明串口没有正常连接，需要检查硬件连线及相关串口参数。

注意：关闭minicom请先按Ctrl+A，再按X。不要使用界面右上角的×退出，否则会在下一次执行minicom命令时提示被锁定，拒绝访问。

③ 按Ctrl+A，再按S，出现下图



```
Knight@localhost:/home/Knight/hello
File Edit View Terminal Tabs Help

Welcome to minicom 2.3
OPTIONS: I18n
Compiled on Mar 13 2008, 00:58:14.
Port /dev/ttyUSB0

Press CTRL-A +- [Upload] ---+
                | zmodem      | n special keys
                | ymodem      |
                | xmodem      |
                | kermit       |
                | ascii        |
                +-----+

[root@FriendlyARM /]#
[root@FriendlyARM /]#
```

选择第一个zmodem，回车后，出现下图

```
Knicht@localhost:/home/Knight/hello
File Edit View Terminal Tabs Help

We+-----[Select one or more files for upload]-----+
|Directory: /root
OP| [..]
Co| [.designer]
Po| [.gconf]
| [.gconfd]
| [.gnome2]
| [.gnome2_private]
| [.qt]
[r] .bash_history
[r] .bash_logout
| .bash_profile
| .bashrc
| .bashrc~
| .cshrc
| .designerrc
| .designerrc~
+-----+
( Escape to exit, Space to tag )

[Goto] [Prev] [Show] [Tag] [Untag] [Okay]
```

CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.3 | VT102 | Offline

回车后，出现下图

```
Knicht@localhost:/home/Knight/hello
File Edit View Terminal Tabs Help

We+-----[Select one or more files for upload]-----+
|Directory: /root
OP| [..]
Co| [.designer]
Po| [.gconf]
| [.gconfd]
| [.gnome2]
| [.gnome2_private]
| [.qt]
[r] .bash_history
[r] .bash_logout
| .bash_profile
| .bashrc
| .bashrc~
| .cshrc
| .designerrc
| .designerrc~
+-----+
( Escape to exit, Space to tag )

[Goto] [Prev] [Show] [Tag] [Untag] [Okay]
```

CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.3 | VT102 | Offline

输入绝对路径或者通过键盘快捷键选择文件后，出现下图

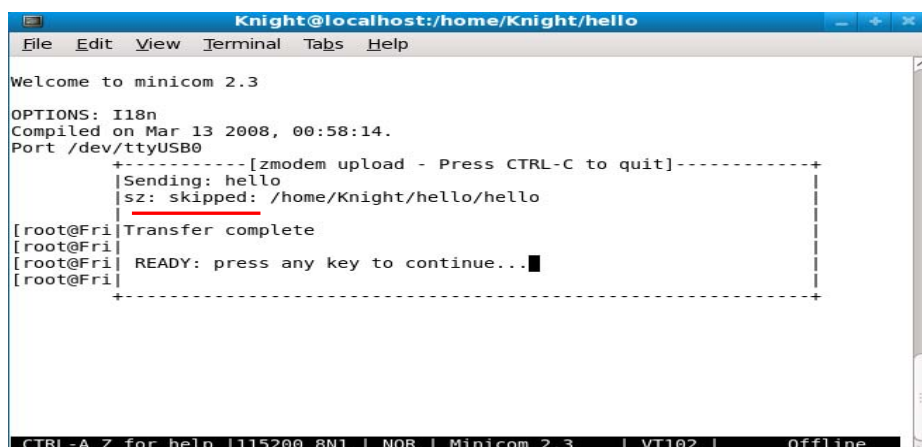
```
Knicht@localhost:/home/Knight/hello
File Edit View Terminal Tabs Help

We+-----[Select one or more files for upload]-----+
|Directory: /root
OP| [..]
Co| [.designer]
Po| [.gconf]
| [.gconfd]
| [.gnome2]
| [.gnome2_private]
| [.qt]
[r] .bash_history
[r] .bash_logout
| .bash_profile
| .bashrc
| .bashrc~
| .cshrc
| .designerrc
| .designerrc~
+-----+
( Escape to exit, Space to tag )

[Goto] [Prev] [Show] [Tag] [Untag] [Okay]
```

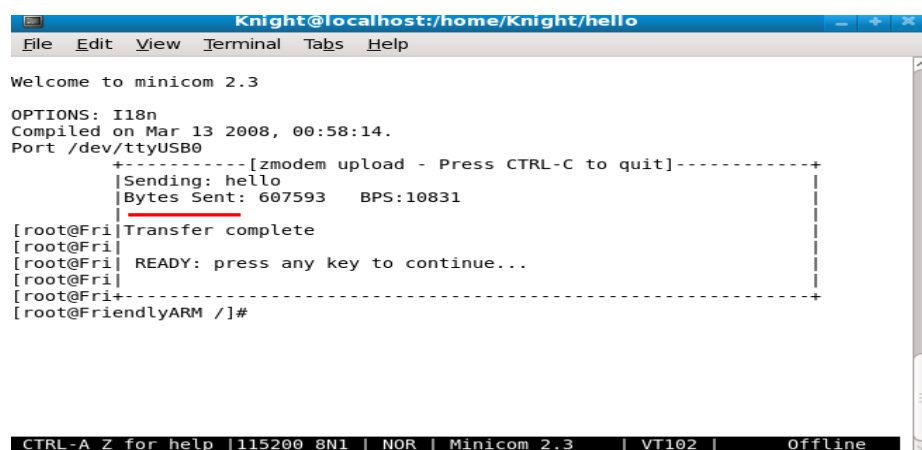
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.3 | VT102 | Offline

回车，出现下图



从上图可以看出，文件未被下载，原因是Tiny6410板子已经有了hello可执行文件（之前下载的），若需要下载，则需要删除之前的hello文件，rm hello即可。可通过ls命令查看是否删除成功。

删除之后，继续下载，出现下图



下载成功。

④ 运行程序

在电脑终端输入./hello，或者在Tiny6410终端输入hello都可。

六、实验修改要求

将原程序中显示的“Hello, World!”修改为“Hello, HFUT!”，并增加一行，显示本人学号和实验日期，显示格式为：“201*****|2021-07-05”

实验二 汇编点亮 LED

一、实验目的

学会Linux系统中开发汇编程序的步骤和方法。在此基础上，掌握通过汇编程序访问GPIO端口，以实现控制Tiny6410开发板上LED的方法。

二、实验内容

本次实验使用Fedora 9.0操作系统环境, 安装ARM-Linux的开发库及编译器。学习在Linux下的编程和编译过程，即创建一个新目录leds_s，使用编辑器建立start.S和Makefile文件，并使用汇编语言编写LED控制程序。编译程序，并下载文件到目标开发板上运行。

三、预备知识

- 1、清楚ARM微处理器芯片S3C6410的GPIO口硬件资源及相应寄存器结构。
- 2、有ARM汇编语言基础。
- 3、会使用Linux下常用的编辑器。
- 4、掌握Makefile的编写和使用。
- 5、了解Linux下的编译程序与交叉编译的过程。

四、实验设备及工具（包括软件调试工具）

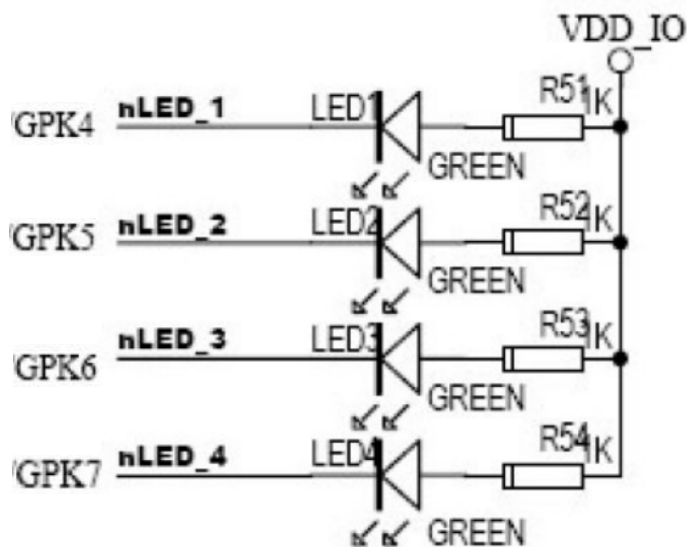
硬件：Tiny6410嵌入式实验平台。

软件：PC机操作系统Fedora9+MiniTools+ARM-Linux开发环境

五、实验原理

1、LED硬件原理图

Tiny6410开发板上提供了4个可编程用户LED，原理图如下：



LED硬件原理图

可见，LED1、2、3、4分别使用的CPU GPIO端口资源为GPK_4、5、6、7。

2、点亮LED控制原理

点亮4个LED需如下2个步骤：

- 把外设的基地址告诉CPU。对于6410来说，内存的地址范围为0x00000000-0x60000000，外设的地址范围为0x70000000-0x7fffffff。
- 关闭看门狗，防止程序不断重启；设置寄存器GPKCON0，使GPK_4/5/6/7四个引脚为输出功能；往寄存器GPKDAT写0，使GPK_4/5/6/7四个引脚输出低电平，4个LED会亮；相反，往寄存器GPKDAT写1，使GPK_4/5/6/7四个引脚输出高电平，4个LED会灭。

六、实验步骤

1、建立工作目录leds（若系统中已建立该目录，可跳过本步骤）。

点击【虚拟机】菜单中的【设置】，选择【选项】中的“共享文件夹”，添加Windows系统中的桌面路径为共享文件夹。在Windows系统的桌面上，右键复制leds文件夹，然后进入虚拟机当前用户的Home目录，使用右键粘贴，将文件夹从windows系统复制到虚拟机的系统中。

2、编写程序源代码

在Linux下的文本编辑器有许多，常用的是vim和Xwindow界面下的gedit

等，建议在实验中使用vim（需要学习vim的操作方法，请参考相关书籍中的关于vim的操作指南）。

① start.S的汇编源程序如下：

```
.global _start
_start:
// 把外设的基地址告诉CPU
ldr r0, =0x70000000 //对于6410来说,内存(0x00000000~0x60000000),外设(0x70000000~0x7fffffff)
orr r0, r0, #0x13 //外设大小:256M
mcr p15,0,r0,c15,c2,4 //把r0的值(包括了外设基地址+外设大小)告诉cpu
// 关看门狗
ldr r0, =0x7E004000
mov r1, #0
str r1, [r0]
// 设置GPKCON0
ldr r1, =0x7F008800
ldr r0, =0x11110000
str r0, [r1]
mov r2, #0x1000
led_blink:
// 设置GPKDAT, 使GPK_4/5/6/7引脚输出低电平, LED亮
ldr r1, =0x7F008808
mov r0, #0
str r0, [r1]
// 延时
bl delay
// 设置GPKDAT, 使GPK_4/5/6/7引脚输出高电平, LED灭
ldr r1, =0x7F008808
mov r0, #0xf0
str r0, [r1]
// 延时
bl delay
sub r2, r2, #1
cmp r2, #0
bne led_blink
halt:
b halt
delay:
mov r0, #0x1000000
delay_loop:
cmp r0, #0
sub r0, r0, #1
bne delay_loop
mov pc, lr
```

② Makefile文件如下：

```
led.bin: start.o
arm-linux-ld -Ttext 0x50000000 -o led.elf $^
arm-linux-objcopy -O binary led.elf led.bin
arm-linux-objdump -D led.elf > led_elf.dis
%.o : %.S
arm-linux-gcc -o $@ $< -c

%.o : %.c
arm-linux-gcc -o $@ $< -c

clean:
rm *.o *.elf *.bin *.dis -rf
```

在Makefile所在目录下执行make命令时，系统会执行如下操作：

- 执行arm-Linux-gcc-o \$@ \$< -c命令，将当前目录下存在的汇编文件和C文件编译成.o文件；

- 执行`arm-Linux-ld -Ttext 0x50000000 -o led.elf $^`，将所有.o文件链接成elf文件，`-Ttext 0x50000000`表示程序的运行地址是0x50000000，即程序只有位于该地址上才能正常运行；
- 执行`arm-Linux-objcopy -O binary led.elf led.bin`，将elf文件抽取为可在开发板上运行的bin文件；
- 执行`arm-Linux-objdump -D led.elf > led_elf.dis`，将elf文件反汇编后保存在dis文件中，调试程序时可能会用到。

3、编译及下载运行程序

① 编译代码

确保当前用户为root用户（可使用`su root`命令切换到root用户）的条件下，在 Fedora的终端中执行如下命令：

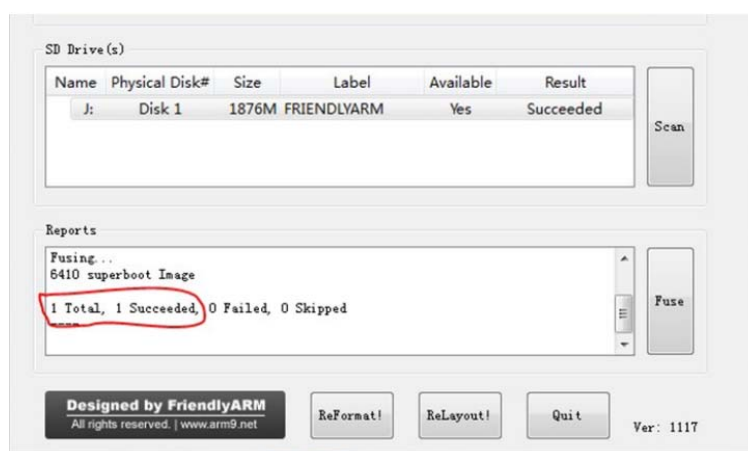
```
# cd leds
```

```
# make
```

执行 `make` 后会生成 `led.bin` 文件。

② 下载（烧写）和运行程序

在windows系统中，以管理员权限使用SD-Flasher程序，将引导程序 Superboot-6410.bin烧入SD卡。成功烧写后的结果如下图所示：



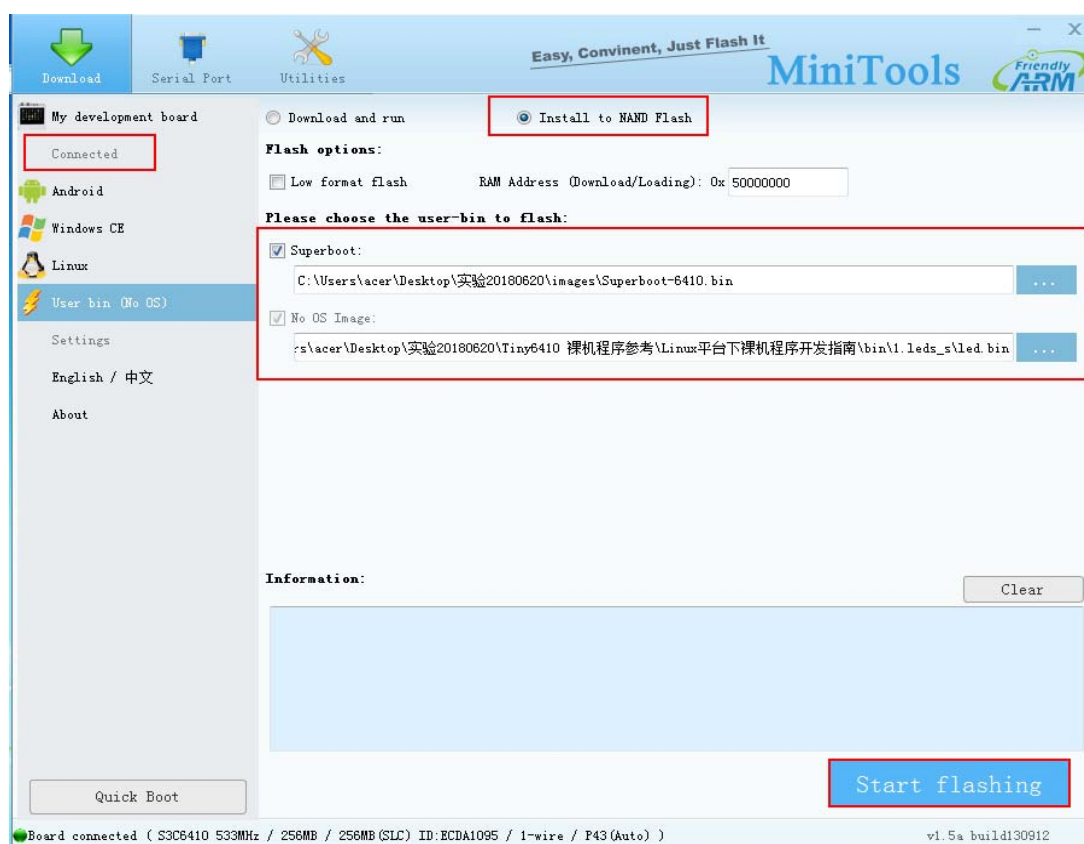
在SD卡中建立一个images文件夹，并把配置文件FriendlyARM.ini复制到该

文件夹中。双击打开SD卡中的该配置文件，在任意位置加入以下内容：USB-Mode = yes （注意字符的大小写）。如下图所示：



将烧写完成的SD卡，插入Tiny6410开发板的SD卡槽，并将开发板S2拨动开关置于SDBOOT位置，然后打开开发板电源，开发板将进入USB下载模式，LCD显示屏上显示“USB Mode: Waiting...”。此时用MiniUSB线连接开发板与PC机，LCD上会显示“USB Mode: Connected”。

在windows系统中，以管理员身份运行MiniTools软件，打开如下软件界面：



确认界面下方显示绿色图标，表明MiniTools已通过MiniUSB线与开发板成

功连接。接着，选择Install to NAND Flash，表示将裸机程序烧入NAND Flash，但不需要运行。设置好下载地址“RAM Address(Download/Loading)=0x50000000”，同时勾选Superboot，并选择烧写时需要的引导程序Superboot-6410.bin和要烧写的裸机程序。最后，点击【Start flashing】按钮，执行选择的某一裸机程序（bin文件）烧写。成功后显示如下信息：

```
Information:
Set User-Bin's Address Succeeded
Send File completed, Waitting...
Installing bootloader...
Send File completed, Waitting...
Downloading User-Bin succeed
Installing User-Bin succeed
All operations was completed successfully.
```

关闭开发板电源，将S2拨动开关置于NAND位置，然后重新开启开发板电源，则开发板运行刚刚烧入的裸机程序。

七、实验修改要求

在延时不变的条件下，将原程序中“四个LED全亮，然后全灭”的显示状态，修改为“四个全亮，然后1和2亮（3和4灭），然后3和4亮（1和2灭），最后全灭”。

实验三 查询方式检测按键

一、实验目的

学会Linux系统中开发C程序的步骤和方法。在此基础上，掌握通过汇编程序实现Tiny6410中断控制器初始化及通过C程序实现中断处理的方法。

二、实验内容

本次实验使用Fedora 9.0操作系统环境, 安装ARM-Linux的开发库及编译器。学习在Linux下的编程和编译过程，即创建一个新目录irq，使用编辑器建立start.S、main.c、irq.c和Makefile等文件，并使用C语言编写中断处理程序。编译程序，并下载文件到目标开发板上运行。

三、预备知识

- 1、清楚ARM微处理器芯片S3C6410的外部中断口硬件资源及相应寄存器结构。
- 2、有ARM汇编和C语言基础。
- 3、会使用Linux下常用的编辑器。
- 4、掌握Makefile的编写和使用。
- 5、了解Linux下的编译程序与交叉编译的过程。

四、实验设备及工具（包括软件调试工具）

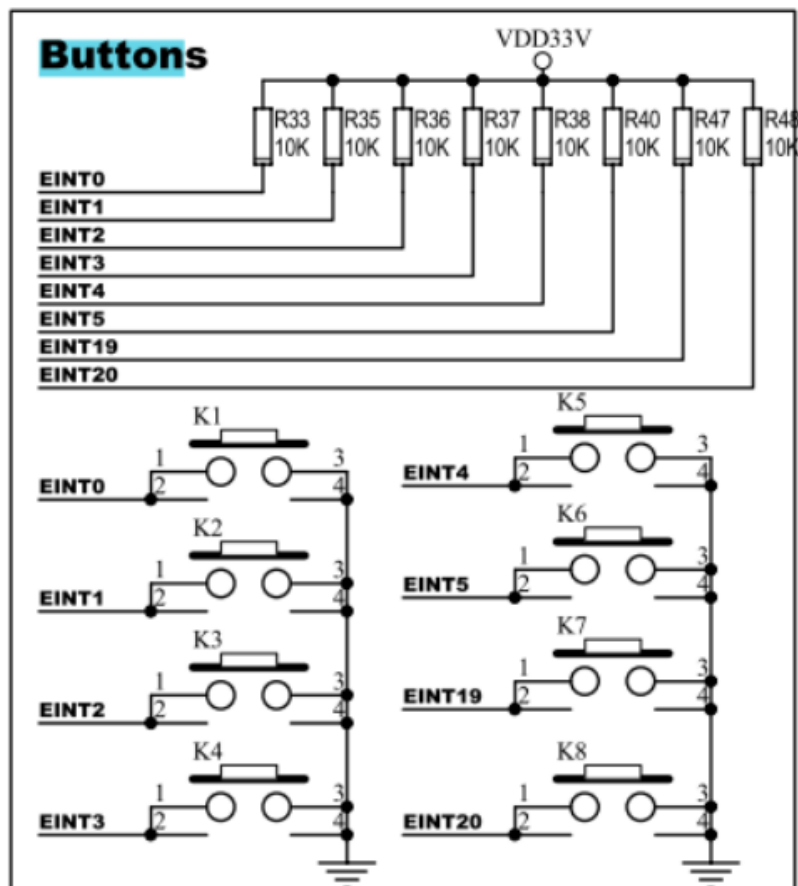
硬件：Tiny6410嵌入式实验平台。

软件：PC机操作系统Fedora9+MiniTools+ARM-Linux开发环境

五、实验原理

1、按键硬件原理图

Tiny6410中共有8个用户按键（☆说明：部分型号的底板上，只有K1~K4四个按键），原理图如下：



按键原理图

相关的引脚信息如下图：

按键	K1	K2	K4	K4	K5	K6	K7	K8
对应的中断	EINT0	EINT1	EINT2	EINT3	EINT4	EINT5	EINT19	EINT20
可复用为GPIO	GPN0	GPN1	GPN2	GPN3	GPN4	GPN5	GPL11	GPL12

按键引脚图

2、查询按键原理

- 把外设的基地址告诉CPU。对于6410来说，内存的地址范围为0x00000000-0x60000000，外设的地址范围为0x70000000-0x7fffffff。
- 关闭看门狗，防止程序不断重启；设置寄存器GPNCON，使接按键K1~K4的GPN0~GPN3为输入功能。设置寄存器GPKCON0，使接LED1~LED4的GPK4~GPK7为输出功能。
- 由原理图可知，按键按下时为低电平。读取寄存器GPNDATA，依次判断哪个按键按下，并点亮对应的LED灯。

六、实验步骤

1、建立工作目录key_led

建立方法同实验二。

2、编写程序源代码

在Linux下的文本编辑器有许多，常用的是vim和Xwindow界面下的gedit等，建议在实验中使用vim（需要学习vim的操作方法，请参考相关书籍中的关于vim的操作指南）。

3、编译及下载运行程序

① 编译代码

确保当前用户为root用户（可使用su root命令切换到root用户）的条件下，在Fedora的终端中执行如下命令：

```
# cd key_led
```

```
# make
```

执行make后会生成key.bin文件。

② 下载（烧写）和运行程序

按实验二给出的方法下载程序后，关闭开发板电源，将S2拨动开关置于NAND位置，然后重新开启开发板电源。当未按下任何按键时，四个LED灯全灭，当按下KEY1~KEY4任意一个按键时，则对应的LED被点亮。

七、实验修改要求

当K1按下时，LED1和LED2被点亮；当K2按下时，LED3和LED4被点亮；当K3按下时，LED1和LED3被点亮；当K4按下时，LED2和LED4被点亮。

实验四 PWM 定时器实验

一、实验目的

学会Linux系统中开发C程序的步骤和方法。在此基础上，掌握通过C程序实现Tiny6410定时器初始化及中断处理的方法。

二、实验内容

本次实验使用Fedora 9.0操作系统环境, 安装ARM-Linux的开发库及编译器。学习在Linux下的编程和编译过程，即创建一个新目录timer，使用编辑器建立start.S、main.c、timer.c和Makefile等文件。编译程序，并下载文件到目标开发板上运行。

三、预备知识

- 1、清楚ARM微处理器芯片S3C6410的定时器硬件资源及相应寄存器结构。
- 2、有ARM汇编和C语言基础。
- 3、会使用Linux下常用的编辑器。
- 4、掌握Makefile的编写和使用。
- 5、了解Linux下的编译程序与交叉编译的过程。

四、实验设备及工具（包括软件调试工具）

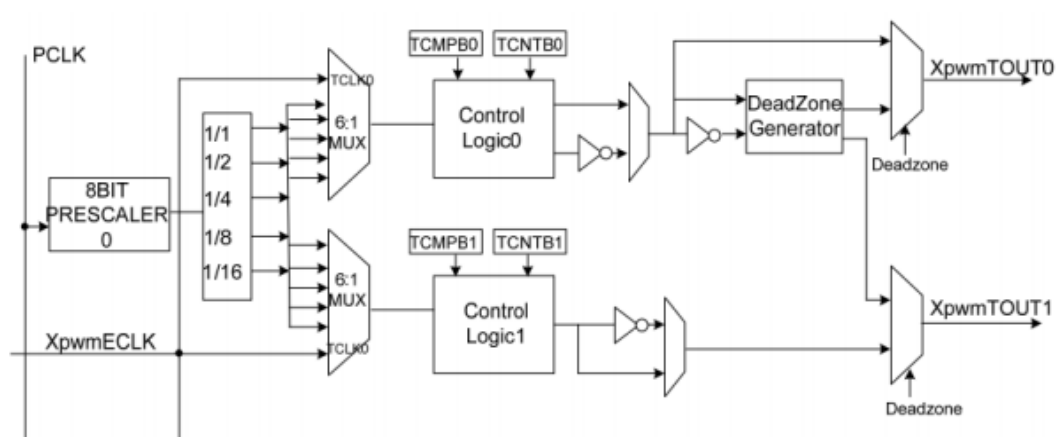
硬件：Tiny6410嵌入式实验平台。

软件：PC机操作系统Fedora9+MiniTools+ARM-Linux开发环境

五、实验原理

1、PWM定时器

S3C6410内部包含五个32位定时器0~4。这些计时器通过产生内部定时中断来与ARM微处理器交互。



定时器0和1的内部结构图

2、中断控制原理

- 把外设的基地址告诉CPU。对于6410来说，内存的地址范围为0x00000000-0x60000000, 外设的地址范围为0x70000000-0x7ffffff。
- 关闭看门狗，防止程序不断重启；设置CPSR，允许IRQ中断；**设置中断控制器中的寄存器VIC0INTENABLE，允许定时器0中断**；设置寄存器TCFG0和TCFG1，写入相应的两次分频系数；设置寄存器TCNTB0和TCMPB0，写入计数初值和PWM比较值；设置寄存器TCON，手动更新计数值，自动重载计数初值，并启动定时器；**设置寄存器TINT_CSTAT，允许定时器0中断**。
- 在中断处理函数中，执行相应的处理。最后，向寄存器TINT_CSTAT对应位写“1”，以实现清除中断标记。

六、实验步骤

1、建立工作目录timer（若系统中已建立该目录，可跳过本步骤）。

2、编写程序源代码

在Linux下的文本编辑器有许多，常用的是vim和Xwindow界面下的gedit等，建议在实验中使用vim（需要学习vim的操作方法，请参考相关书籍中的关于vim的操作指南）。

3、编译及下载运行程序

① 编译代码

确保当前用户为root用户（可使用su root命令切换到root用户）的条件下，在Fedora的终端中执行如下命令：

```
# cd timer
```

```
# make
```

执行make后会生成timer.bin文件。

② 下载（烧写）和运行程序

按实验二给出的方法下载程序后，关闭开发板电源，将S2拨动开关置于NAND位置，然后重新开启开发板电源。初始状态，四个LED灯都为灭。每当定时器0的1秒定时达到后，四个LED切换状态（灭→亮/亮→灭）。

七、实验修改要求

将1秒定时到达后四个LED灯同时亮/灭，修改为四个LED灯按流水灯方式循环显示（LED1→LED2→LED3→LED4→LED1→.....）。