



嵌入式系统 课程实验

第一篇实验



- 1 课程实验框架
- 2 开发环境的构建
- 3 第一次实验
- 4 第二次实验
- 5 实验报告及验收标准

1. 课程实验框架

第一篇 windows环境下裸机开发实验

第二篇 Linux环境下裸机开发实验

第三篇 Linux操作系统移植与驱动初体验

无操作系统的程序开发实验

嵌入式Linux系统下驱动实验

目 录

第一篇 windows 环境下裸机开发实验..... 3

实验一 ADS 集成开发环境的熟悉.....3

实验二 ARM 汇编程序的设计..... 14

实验三 ARM 汇编与 C 混合编程..... 15

实验四 ARM 中断实验..... 16

第二篇 Linux 环境下裸机开发实验..... 18

实验五 熟悉 Linux 操作系统..... 18

实验六 Linux 常用命令的熟悉..... 22

实验七 文本编辑器 Vi 的熟悉..... 23

实验八 编译工具 GCC 的使用.....28

实验九 交叉编译工具的使用..... 31

实验十 Linux 环境下裸机程序设计及编译..... 33

第三篇 Linux 操作系统移植与驱动初体验..... 36

实验十一 Linux 系统移植初体验..... 36

实验十二 Linux 操作系统下驱动开发初体验..... 39

- 课程实验16学时 = 4次实验×4个学时

第一次实验

第二次实验

第三次实验

第四次实验

第7周	星期一	星期二	星期三	星期四	星期五
下午(14:30-17:30)					
晚上(18:30-21:30)	1,2,3班	1,2,3班	4,5,6班（未来）		4,5,6班（未来）
第8周	星期一	星期二	星期三	星期四	星期五
下午(14:30-17:30)					
晚上(18:30-21:30)	1,2,3班	1,2,3班	4,5,6班（未来）		4,5,6班（未来）

目 录

第一篇 windows 环境下裸机开发实验.....	3
实验一 ADS 集成开发环境的熟悉.....	3
实验二 ARM 汇编程序的设计.....	14
实验三 ARM 汇编与 C 混合编程.....	15
实验四 ARM 中断实验.....	16
第二篇 Linux 环境下裸机开发实验.....	18
实验五 熟悉 Linux 操作系统.....	18
实验六 Linux 常用命令的熟悉.....	22
实验七 文本编辑器 Vi 的熟悉.....	23
实验八 编译工具 GCC 的使用.....	28
实验九 交叉编译工具的使用.....	31
实验十 Linux 环境下裸机程序设计及编译.....	33
第三篇 Linux 操作系统移植与驱动初体验.....	36
实验十一 Linux 系统移植初体验.....	36
实验十二 Linux 操作系统下驱动开发初体验.....	39

• 4次实验验收

第一次验收

第二次验收

第三次验收

第四次验收

目 录

第一篇 windows 环境下裸机开发实验..... 3

实验一 ADS 集成开发环境的熟悉.....3

实验二 ARM 汇编程序的设计..... 14

实验三 ARM 汇编与 C 混合编程..... 15

实验四 ARM 中断实验..... 16

第二篇 Linux 环境下裸机开发实验..... 18

实验五 熟悉 Linux 操作系统..... 18

实验六 Linux 常用命令的熟悉..... 22

实验七 文本编辑器 Vi 的熟悉..... 23

实验八 编译工具 GCC 的使用.....28

实验九 交叉编译工具的使用..... 31

实验十 Linux 环境下裸机程序设计及编译..... 33

第三篇 Linux 操作系统移植与驱动初体验..... 36

实验十一 Linux 系统移植初体验..... 36

实验十二 Linux 操作系统下驱动开发初体验..... 39

• 2次实验报告

第一次实验报告

第二次实验报告

2. 开发环境构建

2.1 开发模式

2.2 硬件环境的搭建

2.1 开发模式

1) windows + ADS + 开发板（裸机）

实验指导手册“第一篇”

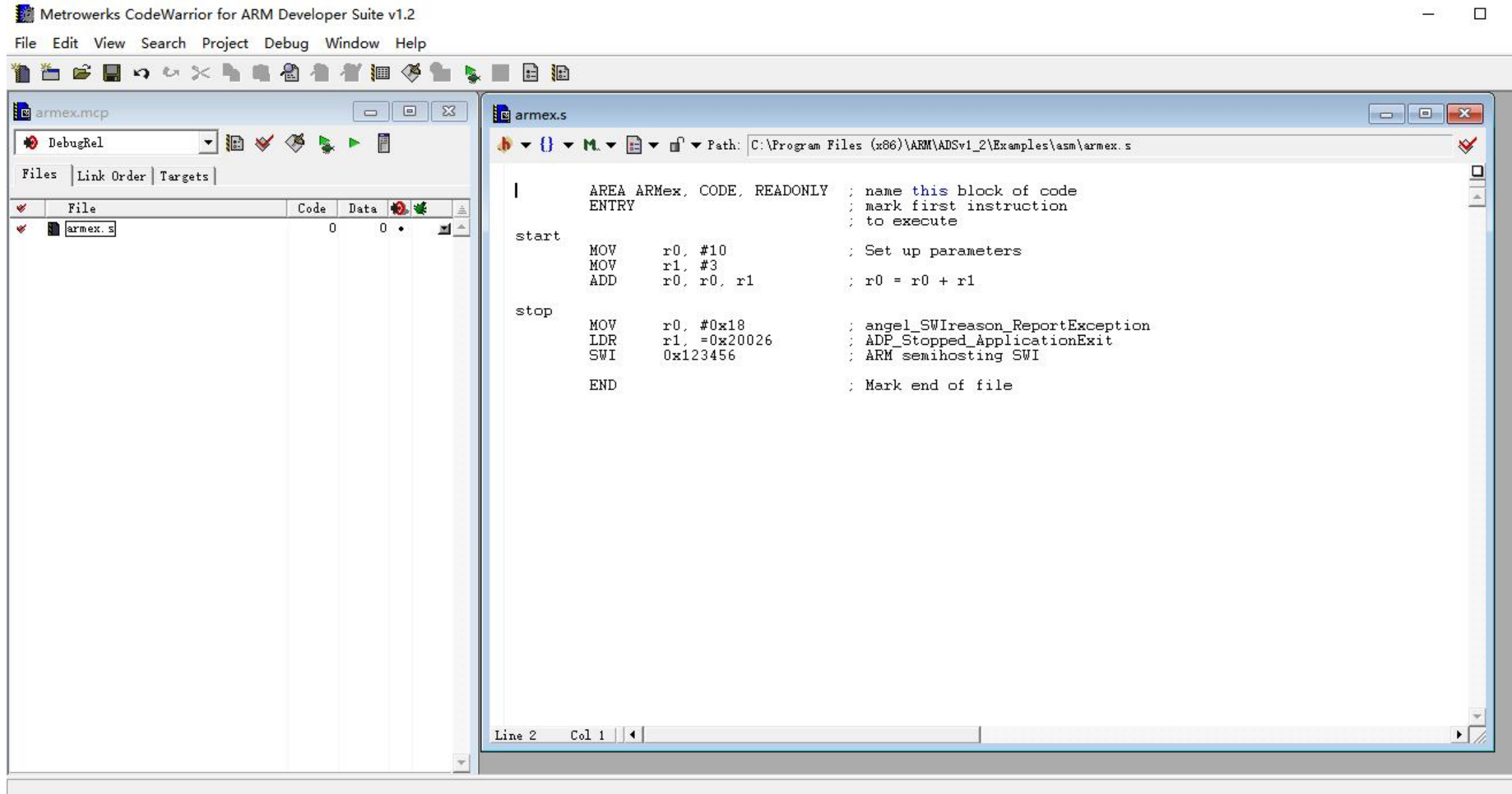
2) Linux + 交叉编译工具 + 开发板（裸机）

实验指导手册“第二篇”

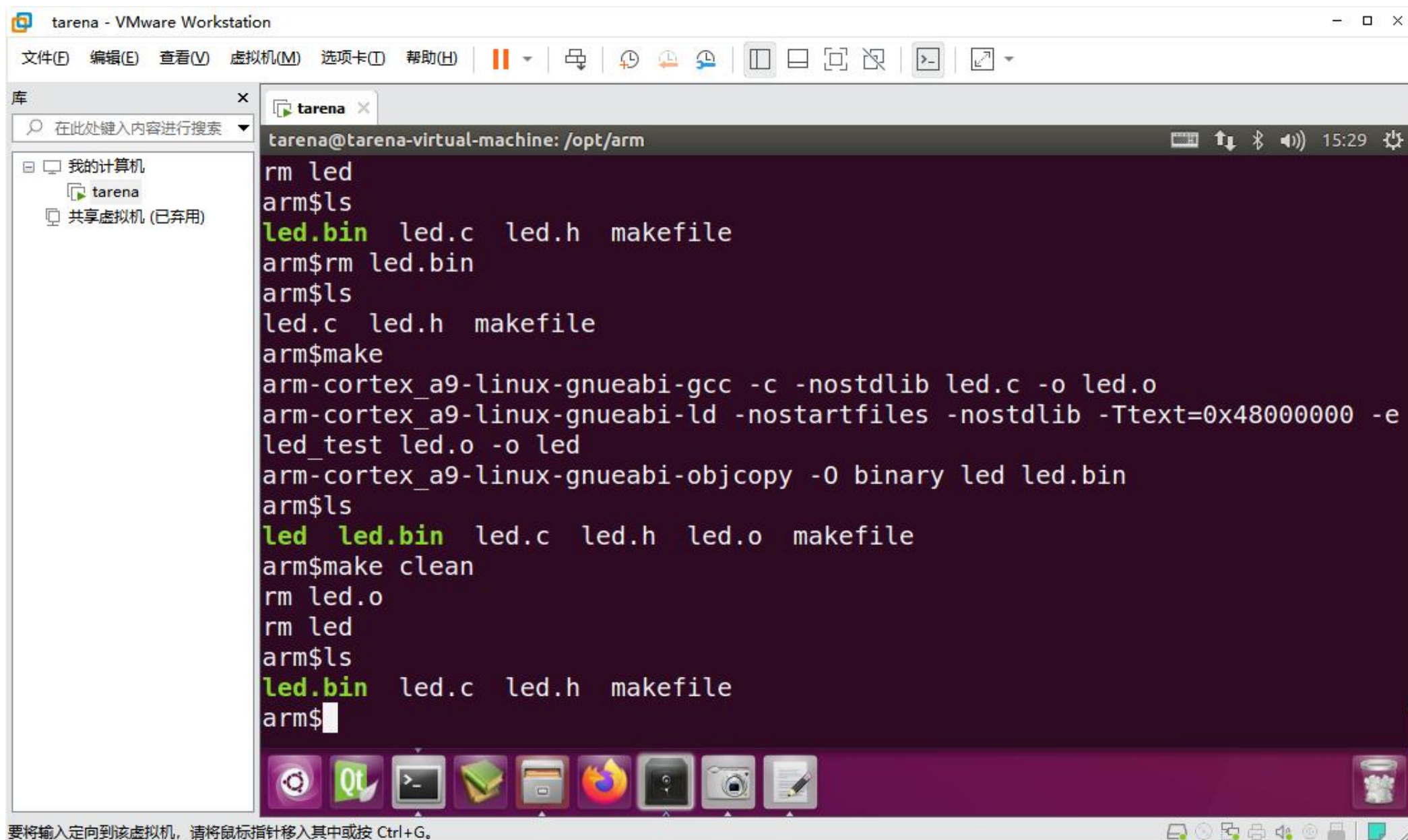
3) Linux + 开发板 + 嵌入式Linux

实验指导手册“第三篇”

1) windows + ADS + 开发板（裸机）



2) Linux + 交叉编译工具 + 开发板 (裸机)



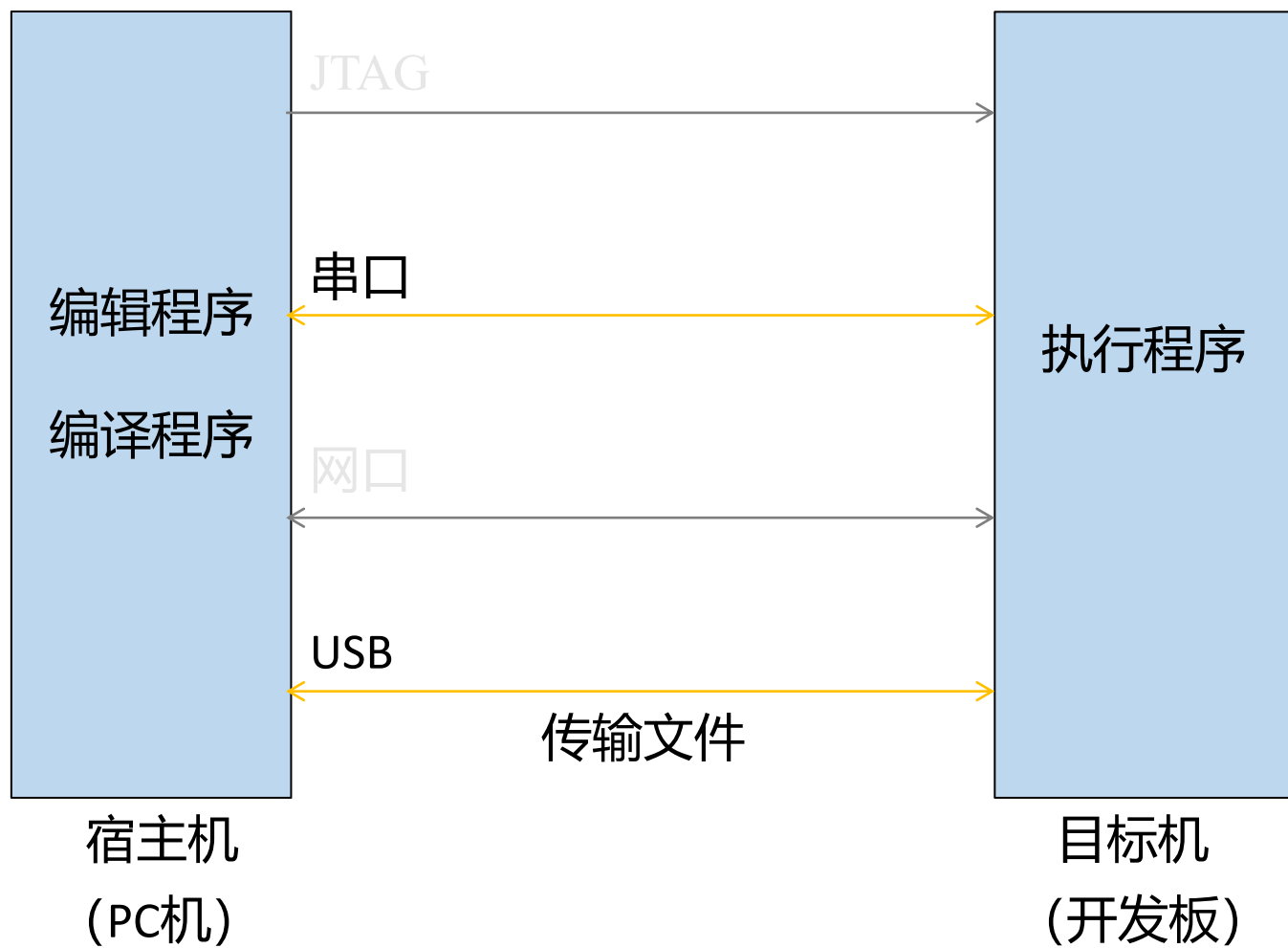
The screenshot shows a VMware Workstation window titled "tarena - VMware Workstation". The main area displays a terminal window for a virtual machine named "tarena". The terminal prompt is "tarena@tarena-virtual-machine: /opt/arm". The terminal output shows the following commands and their results:

```
rm led
arm$ls
led.bin led.c led.h makefile
arm$rm led.bin
arm$ls
led.c led.h makefile
arm$make
arm-cortex_a9-linux-gnueabi-gcc -c -nostdlib led.c -o led.o
arm-cortex_a9-linux-gnueabi-ld -nostartfiles -nostdlib -Ttext=0x48000000 -e
led_test led.o -o led
arm-cortex_a9-linux-gnueabi-objcopy -O binary led led.bin
arm$ls
led led.bin led.c led.h led.o makefile
arm$make clean
rm led.o
rm led
arm$ls
led.bin led.c led.h makefile
arm$
```

The terminal window has a taskbar at the bottom with icons for Qt, a terminal, a folder, Firefox, and other applications. The VMware interface includes a menu bar (文件(F), 编辑(E), 查看(V), 虚拟机(M), 选项卡(T), 帮助(H)), a toolbar, and a sidebar on the left showing the "库" (Library) with "我的计算机" (My Computer) and "共享虚拟机 (已弃用)" (Shared Virtual Machines (Deprecated)).

要将输入定向到该虚拟机, 请将鼠标指针移入其中或按 Ctrl+G.

2.2 硬件环境搭建





TQ2440开发板

USB线



串口线



宿主机和目标机间的通信

- **串口传输：**传输协议通常是 xmodem / ymodem / zmodem 。程序简单，传输的速度比较慢，115200bps
- **以太网传输：**TFTP、NFS 协议是最常见的方式。这时主机上要开启NFS和TFTP服务。（10-100M）（内核和应用程序的下载）
- **USB传输：**双方要都支持USB。（usb1.0-1.5Mbps，usb2.0-480Mbps）
- **网络文件系统NFS：**开发过程中，将主机文件系统挂载到目标板嵌入式系统中，就可在嵌入式系统终端上直接执行主机上的可执行程序。

构建环境所用到的软件

windows

- **ADS1.2** ——是windows下的ADS开发工具
- **SecureCRT_5.1.0**汉化版 ——下串口超级终端
- **dnw**——USB传输软件
- **USB驱动**——USB连接开发板，识别开发板的驱动程序
- **USB转串口驱动**——对于没有串口的电脑，需要使用USB转换成串口使用

Linux

- **VMware**——虚拟机软件（推荐16版）
- **redhat**——红帽子Linux操作系统
- **arm-linux-gcc**——Linux下交叉编译工具

3. 第一次实验

实验指导书中实验一、实验二

实验一 ADS 集成开发环境的熟悉

实验目的:	熟悉 windows 下 ADS 开发环境
	熟悉 TQ2440 电路板。
硬件连接:	开发板三线连接 (电源线、串口线、USB 线), Nor Flash 启动
实验要求:	掌握 ADS 开发环境的使用, 完成硬件连接及程序下载。
实验内容:	1. 在 ADS 中的实现新工程创建、添加文件、编译等步骤
	2. 采用 AXD 软件仿真工具调试程序、观察参数等
	3. 可执行程序的下载
	4. 点亮开发板上 LED 灯

实验二 ARM 汇编程序的设计

实验目的:	掌握 ARM 汇编编程
	掌握 ARM 的 GPIO 口的使用
	理解 TQ2440 电路板中 4 个 LED 灯的 GPIO 连接。
硬件连接:	开发板三线连接 (电源线、串口线、USB 线), Nor Flash 启动
实验要求:	结合 TQ2440 开发板底板原理图及所给部分代码, 实现 4 个 LED 灯二进制流水灯点亮。
实验内容:	1.使用 ADS 编写代码
	2.使用 ADS 编译生成可执行程序
	3.下载可执行程序到开发板, 观察实验结果

3.1 ADS 环境中汇编程序输入规则：

- 1、汇编指令和伪指令、伪操作都**不能顶格写**；
- 2、**标号必须顶格写**，且标号后**不能带冒号**；
- 3、指令中**不能大小写混写**；
- 4、汇编源文件中，语句后面加“**;**”在程序后做标注；
- 5、在 C 源文件中，可以用**//或/*....*/**方式加程序的标注。

3.2 常用的伪指令LDR

功能：

用于加载 32 位立即数或一个绝对地址值到指定的寄存器（写寄存器）；

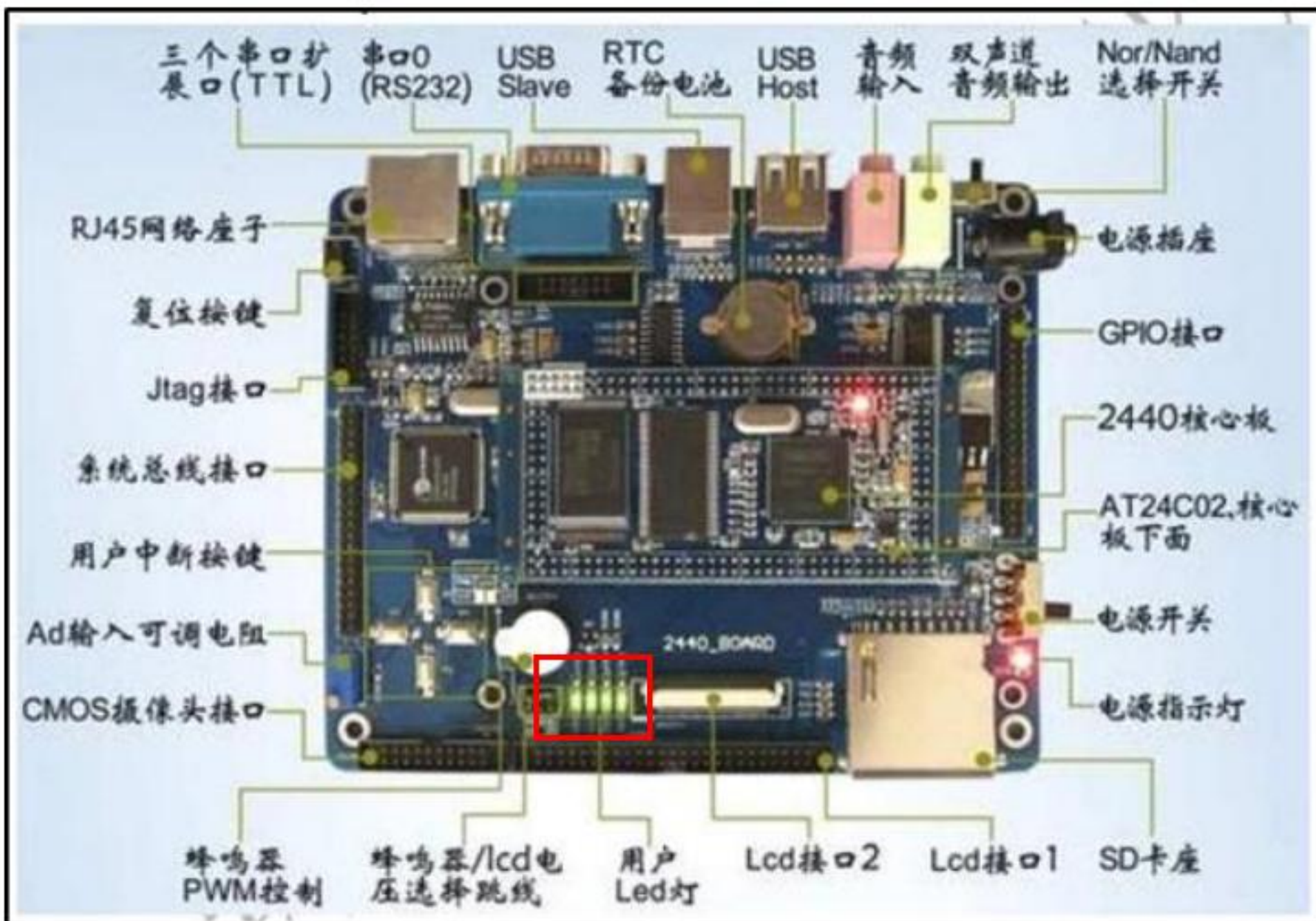
语法格式：

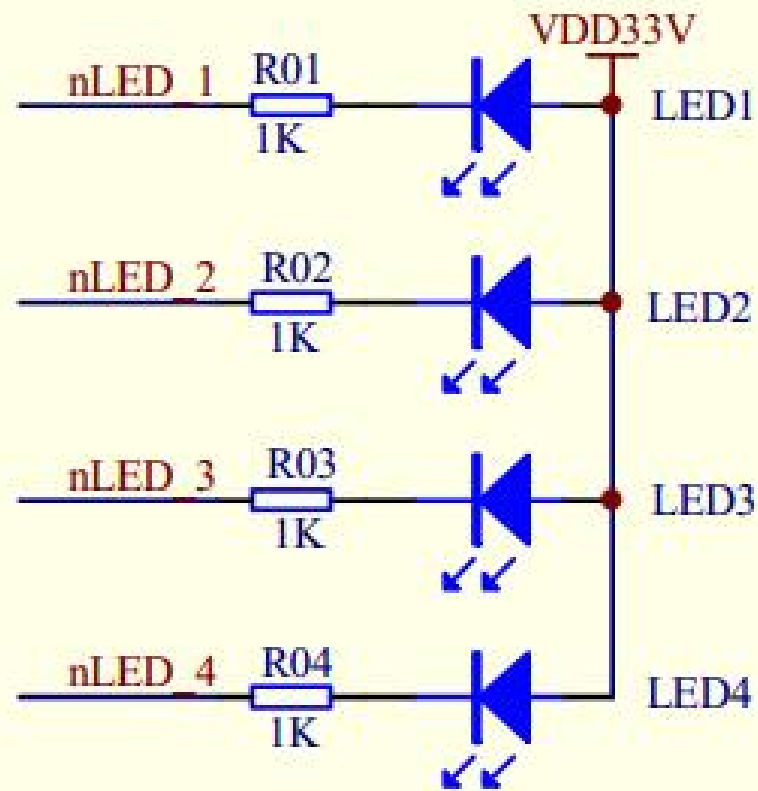
LDR {cond} register, =expr

其中：-Register：加载的目标寄存器。

- expr：32 位常量或地址表达式。

例如：ldr r0,=0x54000010





LDATA29	H10	DATA29
LDATA30	H13	DATA30
LDATA31	F13	DATA31
nXDACK0	L3	nXDACK0/GPB9
nLED_3	K7	nXDACK1/GPB7
nXDREQ0	K6	nXDREQ0/GPB10
nLED_4	K5	nXDREQ1/GPB8
nLED_1	K2	nXBACK/GPB5
nLED_2	L5	nXBREQ/GPB6
nGCS0	F6	nGCS0
nGCS1	B2	nGCS1/GPA12
nGCS2	C3	nGCS2/GPA13
nGCS3	C4	nGCS3/GPA14
nGCS4	D3	nGCS4/GPA15
nGCS5	C2	nGCS5/GPA16
LnOE	C5	nOE
nWAIT	E4	nWAIT

3.3 GPIO口

- S3C2410共有117个多功能I/O端口，分为A~H共8组：

端口A(GPA): 23个输出引脚;

端口B(GPB): 11个输入/输出引脚;

端口C(GPC): 16个输入/输出引脚;

端口D(GPD): 16个输入/输出引脚;

端口E(GPE): 16个输入/输出引脚;

端口F(GPF): 8个输入/输出引脚;

端口G(GPG): 16个输入/输出引脚;

端口H(GPH): 11个输入/输出引脚;

可做通用输入输出口

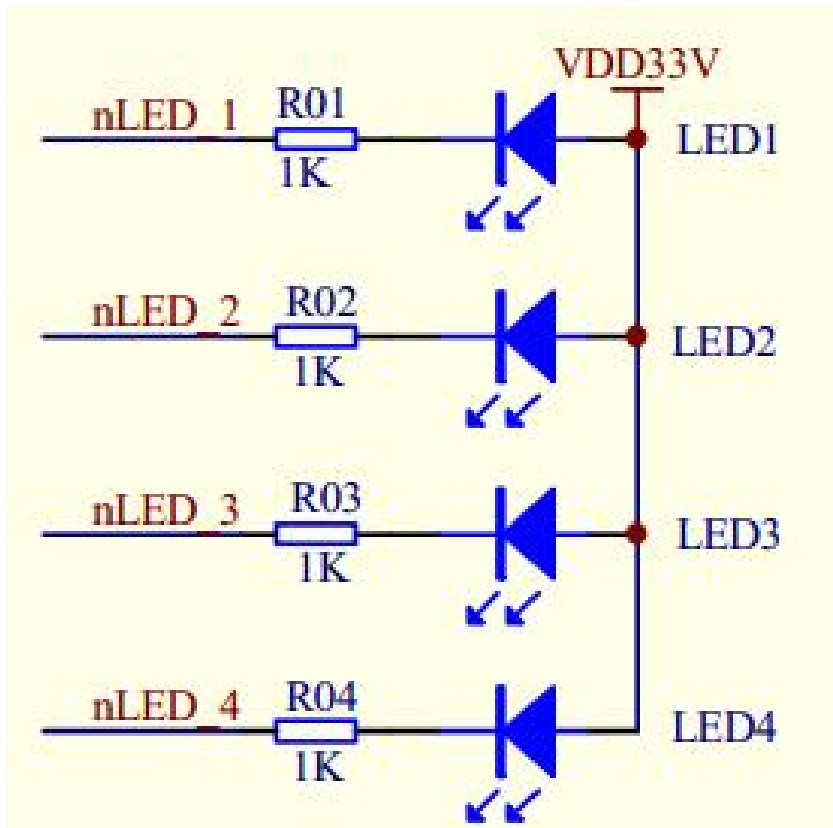
- 可以通过设置寄存器来确定某个引脚用于输入、输出还是其他特殊功能。

GPIO的寄存器

通过寄存器来操作GPIO引脚：

- **GPxCON**：用于选择引脚功能；
- **GPxDAT**：用于读/写引脚数据；
- **GPxUP**：用于确定是否使用内部上拉电阻。

x为B、...、H。



LDATA29	L3	DATA29
LDATA30	H13	DATA30
LDATA31	F13	DATA31
nXDACK0	L3	nXDACK0/GPB9
nLED 3	K7	nXDACK1/GPB7
nXDREQ0	K6	nXDREQ0/GPB10
nLED 4	K5	nXDREQ1/GPB8
nLED 1	K2	nXBACK/GPB5
nLED 2	L5	nXBREQ/GPB6
nGCS0	F6	nGCS0
nGCS1	B2	nGCS1/GPA12
nGCS2	C3	nGCS2/GPA13
nGCS3	C4	nGCS3/GPA14
nGCS4	D3	nGCS4/GPA15
nGCS5	C2	nGCS5/GPA16
LnOE	C5	nOE
nWAIT	E4	nWAIT

Register	Address	R/W	Description	Reset Value
GPBCON	0x56000010	R/W	Configures the pins of port B	0x0
GPBDAT	0x56000014	R/W	The data register for port B	Undef.
GPBUP	0x56000018	R/W	Pull-up disable register for port B	0x0
Reserved	0x5600001c			

- GPBCON

GPBCON寄存器：用于配置引脚功能。每两位对应一根引脚。

PBCON	Bit	Description	
GPB10	[21:20]	00 = Input 10 = nXDREQ0	01 = Output 11 = reserved
GPB9	[19:18]	00 = Input 10 = nXDACK0	01 = Output 11 = reserved
GPB8	[17:16]	00 = Input 10 = nXDREQ1	01 = Output 11 = Reserved
GPB7	[15:14]	00 = Input 10 = nXDACK1	01 = Output 11 = Reserved
GPB6	[13:12]	00 = Input 10 = nXBREQ	01 = Output 11 = reserved
GPB5	[11:10]	00 = Input 10 = nXBACK	01 = Output 11 = reserved
GPB4	[9:8]	00 = Input 10 = TCLK [0]	01 = Output 11 = reserved
GPB3	[7:6]	00 = Input 10 = TOUT3	01 = Output 11 = reserved
GPB2	[5:4]	00 = Input 10 = TOUT2	01 = Output 11 = reserved]
GPB1	[3:2]	00 = Input 10 = TOUT1	01 = Output 11 = reserved
GPB0	[1:0]	00 = Input 10 = TOUT0	01 = Output 11 = reserved

- **GPBDAT**

GPBDAT寄存器：用于读/写引脚。

- 引脚设为输入时，读此寄存器可知相应引脚的电平状态是高还是低；
- 引脚设为输出时，写此寄存器相应位可令此引脚输出高或低电平。

GPBDAT	Bit	Description
GPB[10:0]	[10:0]	When the port is configured as input port, the corresponding bit is the pin state. When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

- **GPBUP**

GPBUP寄存器：设置某位是否禁止接内部上拉电阻

=1，禁止，即不接内部上拉电阻，则相应管脚电平=0；

=0，不禁止，即接，则相应管脚电平=1。

GPBUP	Bit	Description
GPB[10:0]	[10:0]	0: The pull up function attached to the corresponding port pin is enabled. 1: The pull up function is disabled.

4. 第二次实验

实验指导书中实验三、实验四

实验三 ARM 汇编与 C 混合编程

实验目的:	掌握 ARM 汇编与 C 语言程序的混合编程
	理解 TQ2440 电路板中 4 个 LED 灯的 GPIO 连接。
	理解 TQ2440 电路板中 4 个开关的 GPIO 连接。
硬件连接:	开发板三线连接 (电源线、串口线、USB 线), Nor Flash 启动
实验要求:	结合 TQ2440 开发板底板原理图及所给部分代码, 实现 4 个按键控制 4 个 LED 灯的亮灭, 本实验采用查询的方式实现开关状态的检测。
实验内容:	1.使用 ADS 编写代码
	2.使用 ADS 编译生成可执行程序
	3.下载可执行程序到开发板, 观察实验结果

实验四 ARM 中断实验

实验目的:	掌握 ARM 的中断控制器的原理及使用
	理解 TQ2440 电路板中 4 个 LED 灯的 GPIO 连接。
	理解 TQ2440 电路板中 4 个开关的 GPIO 连接。
硬件连接:	开发板三线连接 (电源线、串口线、USB 线), Nor Flash 下载程序, Nand Flash 启动运行程序。
实验要求:	结合 TQ2440 开发板底板原理图及所给部分代码, 实现 4 个按键分别控制 4 个 LED 灯的亮灭, 本实验采用中断的方式实现开关状态的检测。
实验内容:	1.使用 ADS 编写代码
	2.使用 ADS 编译生成可执行程序
	3.下载可执行程序到开发板, 观察实验结果

4.1 C和汇编混合编程

满足基本ARM ATPCS标准：

- 寄存器的名称及其使用规则

- 数据栈的使用规则

- 参数传递的规则

FD (Full Descending) 满递减

输入参数：R0~R3 (4个参数及以内)

数据栈 (4个参数以上)

输出参数：R0~ R3

C调用汇编编程

- 程序编写遵循ATPCS标准
- 汇编程序中对函数做EXPORT伪指令声明;
- C程序中对函数做extern关键字声明;

```
extern void mystrcopy(char *d, const char *s);  
int main(void)  
{  
    const char *src = "Source";  
    char dest[10];  
  
    ...  
    mystrcopy(dest, src);  
    ...  
}
```

R0

R1

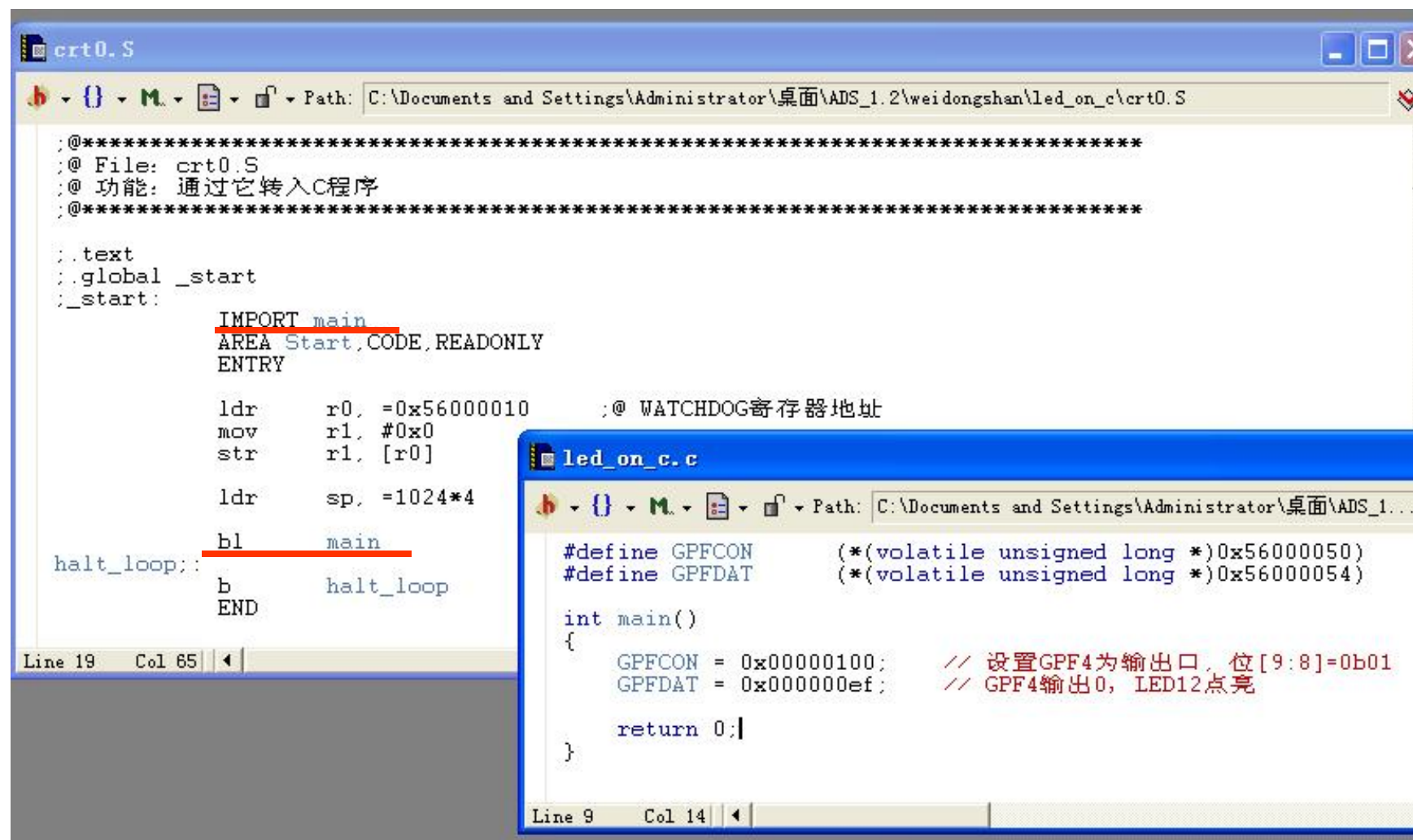
CALL

```
AREA StringCopy, CODE, READONLY  
EXPORT mystrcopy
```

```
mystrcopy  
    LDRB r2, [r1], #1  
    STRB r2, [r0], #1  
    CMP r2, #0  
    BNE mystrcopy  
    MOV pc, lr  
  
END
```

汇编调用C编程

- 程序编写遵循**ATPCS**标准
- 汇编程序中对C函数做**IMPORT**伪指令声明;
- 汇编程序中用**B**或**BL**指令调用;



The image shows two overlapping code editors. The background editor is titled 'crt0.S' and contains assembly code. The foreground editor is titled 'led_on_c.c' and contains C code. Both editors show the file path: 'C:\Documents and Settings\Administrator\桌面\ADS_1.2\weidongshan\led_on_c\crt0.S'.

```
@*****
@ File: crt0.S
@ 功能: 通过它转入C程序
@*****

.text
.global _start
_start:
    IMPORT main
    AREA Start, CODE, READONLY
    ENTRY

    ldr    r0, =0x56000010    ;@ WATCHDOG寄存器地址
    mov    r1, #0x0
    str    r1, [r0]

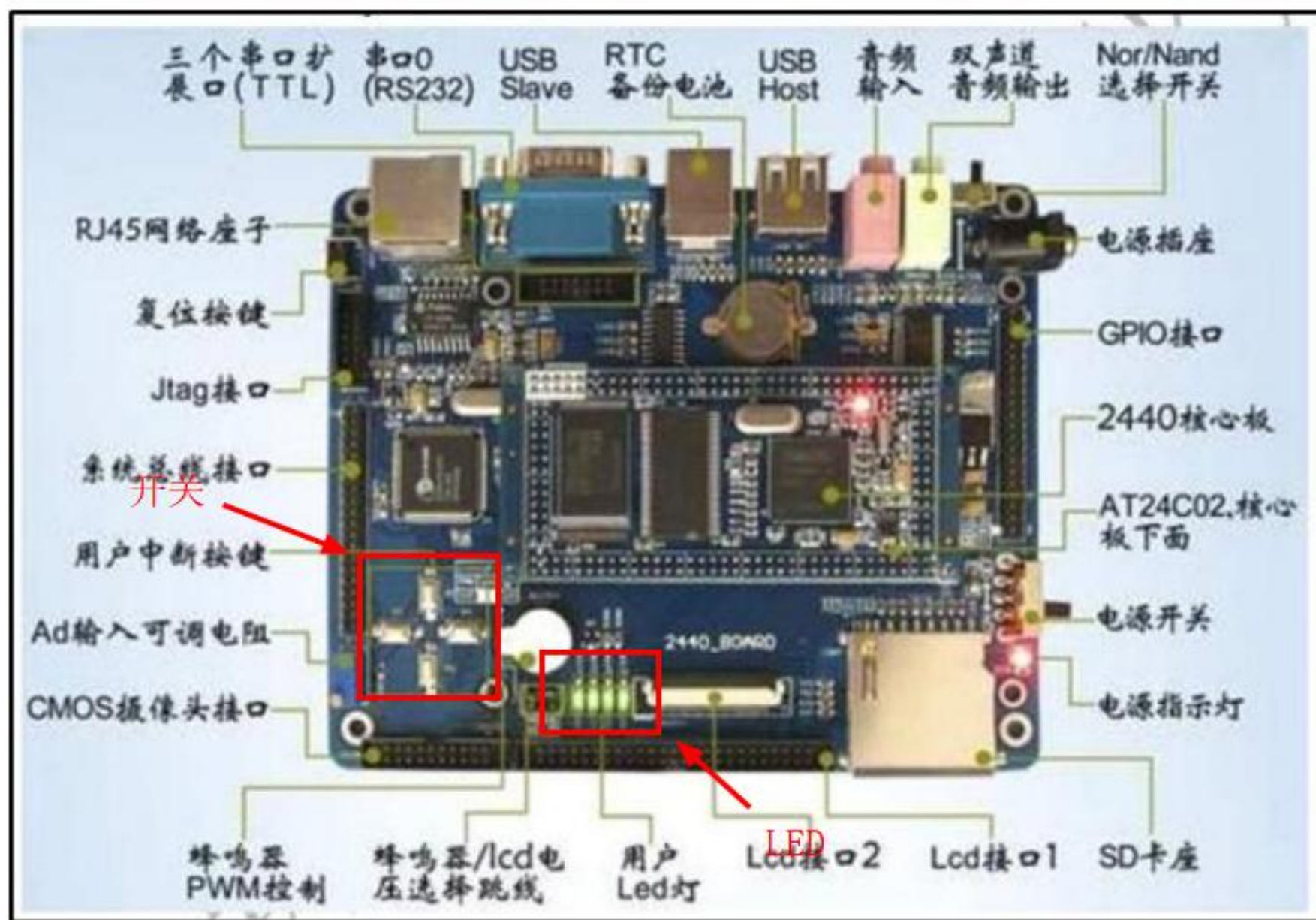
    ldr    sp, =1024*4

    bl     main
halt_loop:
    b      halt_loop
END
```

```
#define GPFCON (*(volatile unsigned long *)0x56000050)
#define GPFDAT (*(volatile unsigned long *)0x56000054)

int main()
{
    GPFCON = 0x00000100;    // 设置GPF4为输出口, 位[9:8]=0b01
    GPFDAT = 0x000000ef;    // GPF4输出0, LED12点亮

    return 0;
}
```

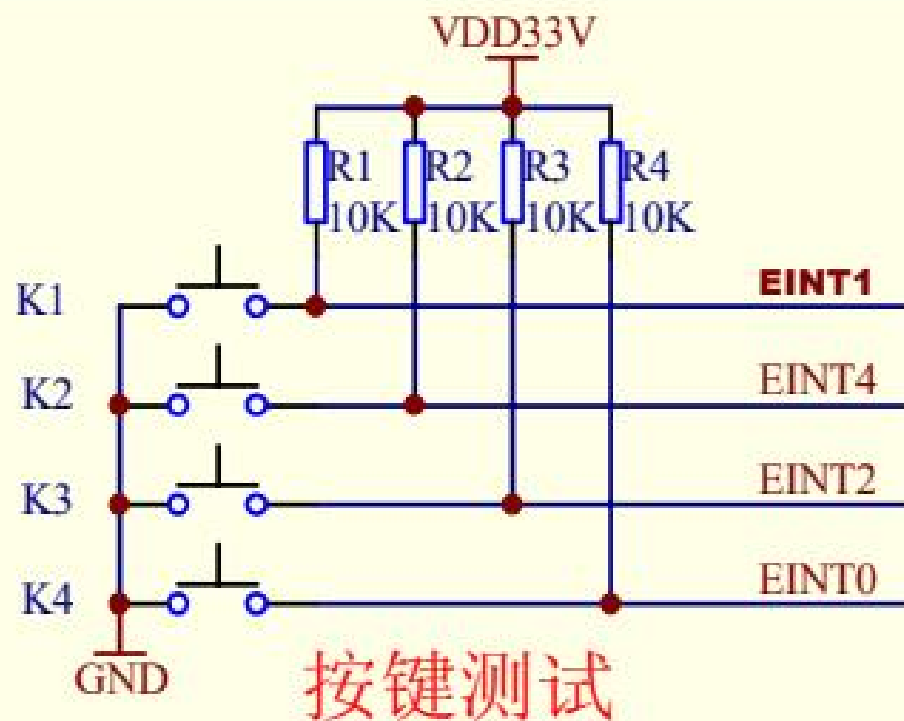


图 2-3 开关连接原理图

External Interrupt

EINT17/GPG9/nRTS1	M11	EINT11/
EINT16/GPG8	T10	nCD SD
EINT9/GPG1	T9	EINT9
EINT8/GPG0	N9	EINT8
EINT7/GPF7	L16	EINT7
EINT6/GPF6	L15	EINT6
EINT5/GPF5	L14	EINT5
EINT4/GPF4	M17	EINT4
EINT3/GPF3	M15	EINT3
EINT2/GPF2	L13	EINT2
EINT1/GPF1	M16	EINT1
EINT0/GPF0	N17	EINT0
	N13	

图 2-4 开关在 2440 端连接图

Register	Address	R/W	Description	Reset Value
GPFCON	0x56000050	R/W	Configures the pins of port F	0x0
GPFDAT	0x56000054	R/W	The data register for port F	Undef.
GPFUP	0x56000058	R/W	Pull-up disable register for port F	0x000
Reserved	0x5600005c	—	—	—

4.2 CPU与开关的交互方式

① 查询方式（轮询方式）：

CPU 不断地查询设备的状态。该方式实现比较简单，但 CPU 利用率很低，不适合多任务的系统。

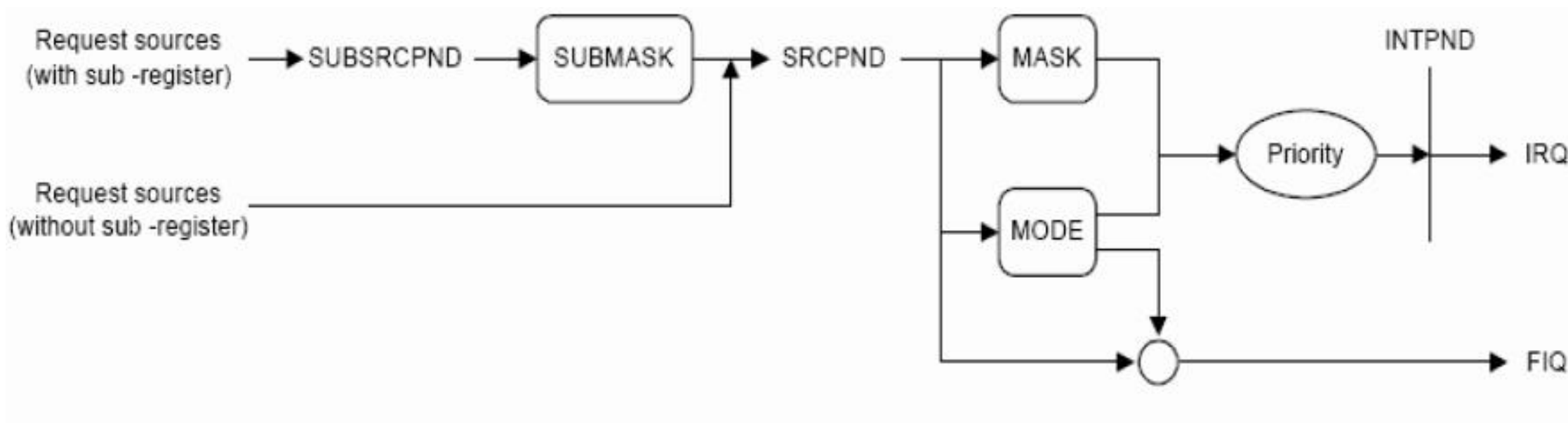
② 中断方式：

CPU 在告知硬件开始一项工作后，就去做别的事去了，当硬件完成了该项工作后，向CPU 发送一个信号，告知 CPU 它已经完成了这项工作。

4.3 中断

① **中断源**：在中断的生命周期中，中断源的作用是负责产生中断信号。S3C2410 支持56 个中断源；S3C2440 支持 60个中断源；S5PV210 支持 93 个中断源。

② S3c2440 中断控制器：



4.3 中断

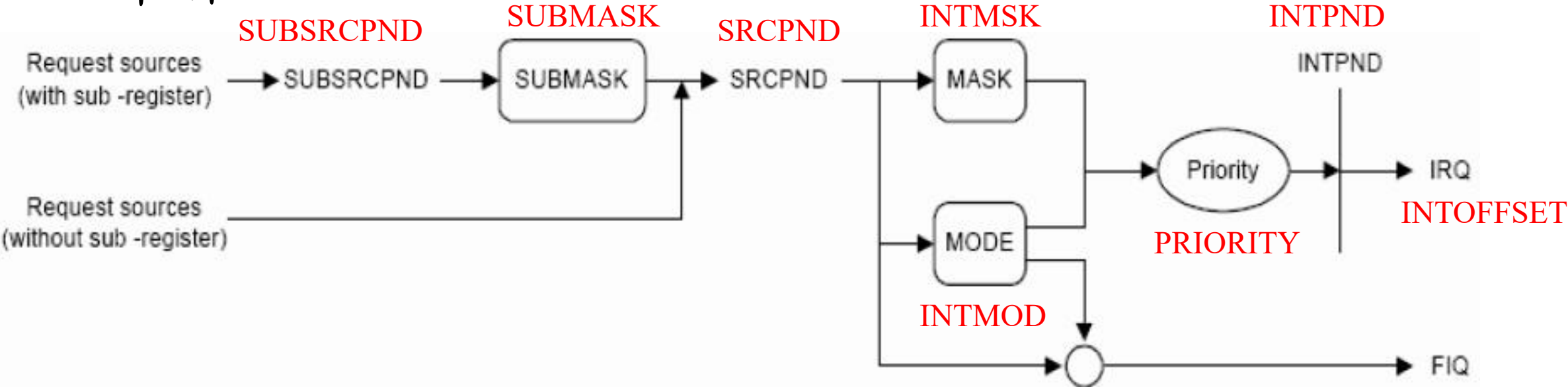


表5.5.1 中断控制器专用寄存器

寄存器	地址	读/写	说明	复位后的值
SRCPND	0x4A000000	读/写	中断标志寄存器	0x00000000
INTMOD	0x4A000004	读/写	中断模式寄存器	0x00000000
INTMSK	0x4A000008	读/写	中断屏蔽寄存器	0xFFFFFFFF
PRIORITY	0x4A00000C	读/写	中断优先级寄存器	0x7F
INTPND	0x4A000010	读/写	中断服务寄存器	0x00000000
INTOFFSET	0x4A000014	读/写	中断偏移寄存器	0x00000000
SUBSRCPND	0x4A000018	读/写	子源挂起寄存器	0x00000000
INTSUBMSK	0x4A00001C	读/写	中断子源屏蔽寄存器	0x7FF

• SRCPND 中断标志寄存器

寄存器名	地址	是否读写	描述	复位默认值
SRCPND	0x4A000000	R/W	中断源暂存寄存器，保存中断请求状态： 0：没有中断请求信号 1：中断请求信号产生	0x00000000

EINT1	[1]	0 = Not requested, 1 = Requested	0
EINT0	[0]	0 = Not requested, 1 = Requested	0

当某个中断信号产生之后，SRCPND 对应位被**自动置 1**，该位会一直保持被置位，只到中断处理程序将其清除为止。清除中断是通过向对应位**写1清0**。

• INTMSK中断屏蔽寄存器

寄存器名	地址	是否读写	描述	复位默认值
INTMSK	0x4A000008	R/W	中断源信号屏蔽寄存器，设置相应位来屏蔽中断信号： 0：未屏蔽，中断可用 1：屏蔽中断信号	0xFFFFFFFF

EINT1	[1]	0 = Service available, 1 = Masked	1
EINT0	[0]	0 = Service available, 1 = Masked	1

该寄存器用来屏蔽中断源信号，默认值为全部中断都被屏蔽掉，因此要想处理某个硬件中断，必须要打开中断屏蔽位，通过**写入 0**来使能中断。

• INTMOD 中断模式寄存器

寄存器名	地址	是否读写	描述	复位默认值
INTMOD	0x4A000004	R/W	中断模式寄存器，指定对应中断模式： 0 = IRQ一般中断模式 1 = FIQ快速中断模式	0x0000000

EINT1	[1]	0 = IRQ, 1 = FIQ	0
EINT0	[0]	0 = IRQ, 1 = FIQ	0

通过 INTMOD 寄存器设置中断模式，如果指定为IRQ中断，那么中断信号会进行优先级仲裁，如果指定为FIQ中断，那么中断信号直接送给 ARM 内核产生中断。

快速中断不存在优先级仲裁，只能有一位被设置为 FIQ 模式。

• INTPND 中断服务寄存器

寄存器名	地址	是否读写	描述	复位默认值
INTPND	0x4A000010	R/W	最高优先级中断暂存寄存器里面保存有经过优先级仲裁的结果： 0：没有中断请求信号 1：中断请求信号产生	0x00000000

EINT1	[1]	0 = Not requested, 1 = Requested	0
EINT0	[0]	0 = Not requested, 1 = Requested	0

该寄存器保存了经过优先级仲裁出的中断信号位，它是所有当前中断请求里优先级别最高的中断，因此该寄存器里的值最多有一位被置 1，通常中断处理程序中会通过读取该寄存器的值来获得当前正在处理的中断请求。中断处理完成之后，通过写入 1 来清除中断。

• INTOFFSET中断偏移寄存器

寄存器名	地址	是否读写	描述	复位默认值
INTOFFSET	0x4A000014	R	中断号偏移量寄存器，用来保存当前处理的中断号	0x0000000

INT Source	The OFFSET value	INT Source	The OFFSET value
INT_ADC	31	INT_UART2	15
INT_RTC	30	INT_TIMER4	14
INT_SPI1	29	INT_TIMER3	13
INT_UART0	28	INT_TIMER2	12
INT_IIC	27	INT_TIMER1	11
INT_USBH	26	INT_TIMER0	10
INT_USBD	25	INT_WDT_AC97	9
INT_NFCON	24	INT_TICK	8
INT_UART1	23	nBATT_FLT	7
INT_SPI0	22	INT_CAM	6
INT_SDI	21	EINT8_23	5
INT_DMA3	20	EINT4_7	4
INT_DMA2	19	EINT3	3
INT_DMA1	18	EINT2	2
INT_DMA0	17	EINT1	1
INT_LCD	16	EINT0	0

- 外部中断触发方式寄存器

Register	Address	R/W	Description	Reset Value
EXTINT0	0x56000088	R/W	External interrupt control register 0	0x000000
EXTINT1	0x5600008c	R/W	External interrupt control register 1	0x000000
EXTINT2	0x56000090	R/W	External interrupt control register 2	0x000000

由于按键按下时让它产生中断，也就是从高电平变为低电平时产生，因此我们设置 EXTINT0中断信号的触发方式为下降沿触发。

EINT0	[2:0]	Setting the signaling method of the EINT0. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
-------	-------	---

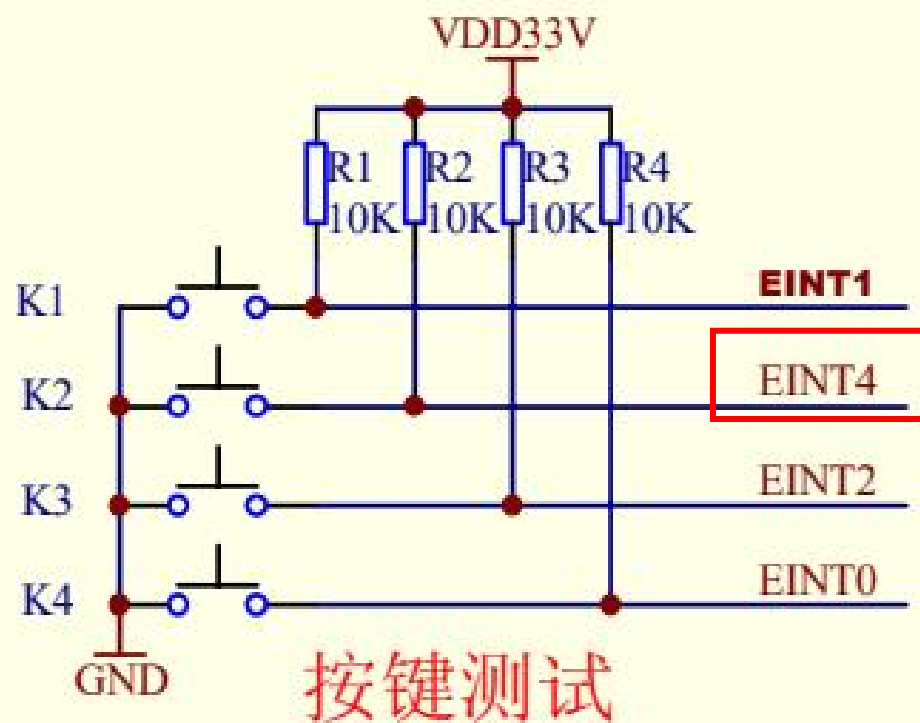


图 2-3 开关连接原理图

EINT8_23	External interrupt 8 – 23	ARB1
EINT4_7	External interrupt 4 – 7	ARB1
EINT3	External interrupt 3	ARB0
EINT2	External interrupt 2	ARB0
EINT1	External interrupt 1	ARB0
EINT0	External interrupt 0	ARB0

4.3 中断

EINTPEND

EINTMASK

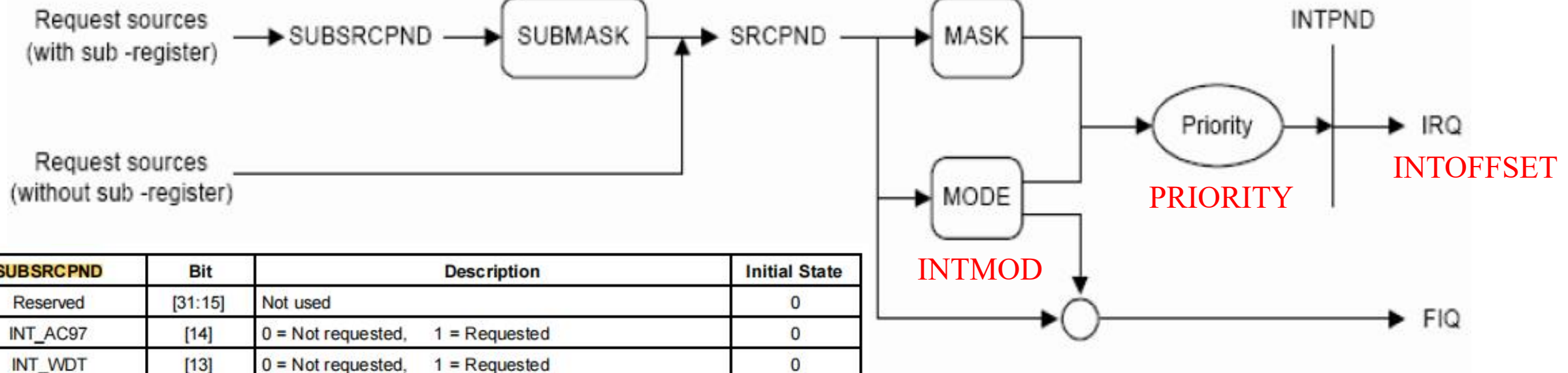
SUBSRCPND

SUBMASK

SRCPND

INTMSK

INTPND



SUBSRCPND	Bit	Description	Initial State
Reserved	[31:15]	Not used	0
INT_AC97	[14]	0 = Not requested, 1 = Requested	0
INT_WDT	[13]	0 = Not requested, 1 = Requested	0
INT_CAM_P	[12]	0 = Not requested, 1 = Requested	0
INT_CAM_C	[11]	0 = Not requested, 1 = Requested	0
INT_ADC_S	[10]	0 = Not requested, 1 = Requested	0
INT_TC	[9]	0 = Not requested, 1 = Requested	0
INT_ERR2	[8]	0 = Not requested, 1 = Requested	0
INT_TXD2	[7]	0 = Not requested, 1 = Requested	0
INT_RXD2	[6]	0 = Not requested, 1 = Requested	0
INT_ERR1	[5]	0 = Not requested, 1 = Requested	0
INT_TXD1	[4]	0 = Not requested, 1 = Requested	0
INT_RXD1	[3]	0 = Not requested, 1 = Requested	0
INT_ERR0	[2]	0 = Not requested, 1 = Requested	0
INT_TXD0	[1]	0 = Not requested, 1 = Requested	0
INT_RXD0	[0]	0 = Not requested, 1 = Requested	0

• 外部中断标志寄存器

EINTPEND (External Interrupt Pending Register)

Register	Address	R/W	Description	Reset Value
EINTPEND	0x560000a8	R/W	External interrupt pending register	0x00

外部中断标志寄存器给出EINT4~23的外部中断标志，当对应位为0表示该引脚无外部中断发生，当对应位为1表示该引脚有外部中断发生。

		0 = Not occur 1 = Occur interrupt	
EINT7	[7]	It is cleared by writing "1" 0 = Not occur 1 = Occur interrupt	0
EINT6	[6]	It is cleared by writing "1" 0 = Not occur 1 = Occur interrupt	0
EINT5	[5]	It is cleared by writing "1" 0 = Not occur 1 = Occur interrupt	0
EINT4	[4]	It is cleared by writing "1" 0 = Not occur 1 = Occur interrupt	0
Reserved	[3:0]	Reserved	0000

- 外部中断使能寄存器

EINTMASK (External Interrupt Mask Register)

Register	Address	R/W	Description	Reset Value
EINTMASK	0x560000a4	R/W	External interrupt mask register	0x000fffff

外部中断使能寄存器设置EINT4~23的外部引脚是否使能中断，当对应位为0表示使能该中断，当对应位为1表示屏蔽该中断。

EINT9	[9]	0 = enable interrupt	1= masked
EINT8	[8]	0 = enable interrupt	1= masked
EINT7	[7]	0 = enable interrupt	1= masked
EINT6	[6]	0 = enable interrupt	1= masked
EINT5	[5]	0 = enable interrupt	1= masked
EINT4	[4]	0 = enable interrupt	1= masked
Reserved	[3:0]	Reserved	

• 中断的步骤

① 在使用中断前，先**设置**好相应模式下的**堆栈**。

② 对于**子中断**，将**INTSUBMSK**中的相应位设为0。

③ 将**INTMSK**中的相应位设为0。

④ 是FIQ还是IRQ：

- 如果是FIQ，则在**INTMOD**寄存器设置相应位为1。
- 如果是IRQ，则在**PRIORITY**寄存器中设置优先级。

⑤ 准备好中断处理函数

- 在**中断向量表**设置好当中断触发时的**跳转函数**。
- 对于IRQ，在跳转函数中读取INTPND寄存器或INTOFFSET寄存器的值来确定中断源，然后调用具体的处理函数。
- 对于FIQ，因为只有一个中断可以设为FIQ，无需判断。

- 1、设置堆栈
- 2、允许中断
- 3、设置优先级
- 4、写中断处理函数
- 5、中断返回的处理
- 6、写1清中断
- 7、开中断

• 中断的步骤

- 中断处理函数进入和返回时**现场保护和恢复**。
 - 中断返回前需要**清中断**，往SUBSRCPND、SRCPND、INTPND中相应位写1即可（写1清0）。
- ⑥ 设置CPSR寄存器中的F-bit或I-bit为0，**开中断**。

- 1、设置堆栈
- 2、允许中断
- 3、设置优先级
- 4、写中断处理函数
- 5、中断返回的处理
- 6、写1清中断
- 7、开中断

5. 实验报告及验收标准

5.1 实验报告

- ① 给出实验二实现开发板上4个LED按二进制流水灯方式闪烁的程序，其中包含必要注释，并附AXD中调试成功的截图；
- ② 给出实验三中软件实现流程图；
- ③ 给出实验四中软件实现流程图；
- ④ 思考题
- ⑤ 体会和建议

5.2 思考题

- ① 在嵌入式系统编程当中，汇编语言和C语言分别有什么优势？是否可以完全摒弃其中一种语言？为什么？
- ② ARM汇编调用C语言以及C语言调用ARM汇编时，如何传递参数？实验二，三程序中参数是如何传递的？
- ③ c语言中和汇编语言中是如何操作寄存器的？
- ④ 比较实验二、三和四中ADS下的工程设置的有何异同，并分析其理由。
- ⑤ 在中断实验中为什么要把可执行程序下载到NANDFlash中运行，而不是直接下载到SDRAM中运行？如果直接下载到SDRAM中运行会发生什么情况？
- ⑥ 结合实验，叙述NAND Flash启动的流程。

5.3 验收标准

	ADS工程建立	编译、链接	AXD调试	下载	实验结果	代码理解优化
实验一二	10	10	20	10	20	25+5
	c和汇编 混合编程	中断理解	-	-	实验结果	代码优化
实验三四	20	20	-	-	45	15

实验规范：

- 1、实验板带电运行时，尽量勿用手触碰实验板背面，以免导致短路；
- 3、实验过程中，爱护实验板，轻拿轻放；
- 4、实验结束后，将开发板恢复原样收拾好包装，放回原处；
- 5、离开前，请及时关闭电脑，清理好桌面，放好板凳；
- 6、离开实验室时，请将产生的垃圾带出实验室。

实验教师：

张莉君 刘玮

实验助理教师：

李亚菲 吴秀成 王子涵 邱晨盼 高翊翔