嵌入式系统与开发

实验报告

|  |  |
| --- | --- |
| 实验名称 | 实验 9 串口控制实验 |
| 实验次数 | 实验九 |
| 学生姓名 | 戴高一 |
| 所在专业 | 计算机科学与技术 |
| 所在班级 | 201班 |
| 指导教师 | 李剑 |
| 地点 | 学10-509 |
| 时间 | 2023年5月 |

实验 9 串口控制实验

**一、实验目的**

1．掌握串口通信编程步骤

2．掌握串口通信程序的编写、编译、运行

3．熟悉嵌入式系统启动代码、头文件、Makefile文件的编写

**二、实验内容**

在PC端的串口工具内（串口工具：D:新509\04-常用工具\串口工具\UartAssist.exe），

由数字键盘输入数字1，利用串口控制LED灯从左往右跑马灯闪烁10次；输入2时LED灯从

右往左跑马灯闪烁10次。编写系统的启动代码、串口通信程序、头文件以及Makefile文

件，编译得到可执行文件，下载至开发板，实现在开发板上启动系统。

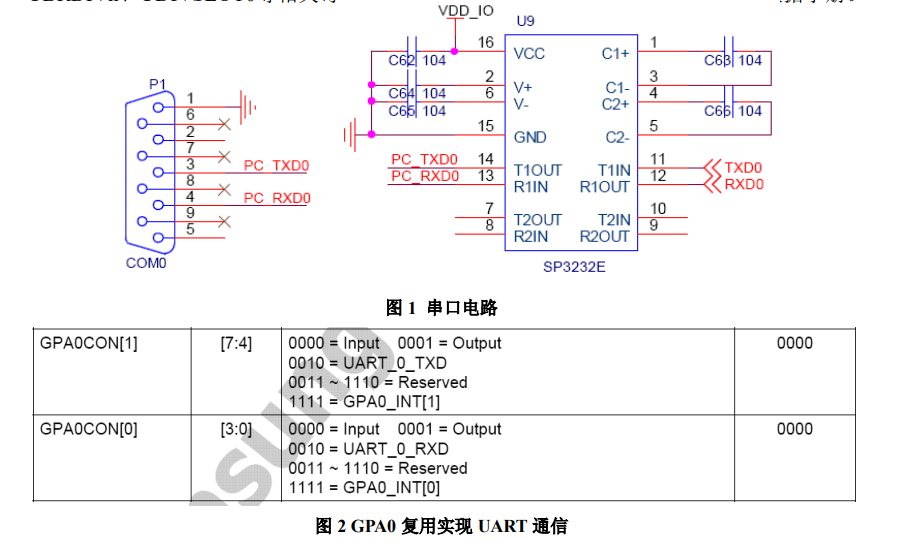
**三、实验原理**

实验采用UART0实现串口通信，如图1所示。UART0的数据收发通过复用GPA0实现，

其中，GPA0\_0复用为UART\_0\_RXD，GPA0\_1复用为UART\_0\_TXD，如图2所示寄存器功能

表。UART数据通信要求对ULCONn、UCONn、UFCONn、UTRSTATn、UTXHn、URXHn、

UBRDIVn、UDIVSLOT0等相关寄存器进行正确配置，寄存器功能详情参考芯片数据手册。



**四、实验步骤**

1．编写键控制代码，将代码编译为二进制文件。

1）在 Linux 系统的共享文件夹 forlinux 中新建 uart 目录。

2）切换到 uart 目录，在 uart 文件夹中新建启动文件 start.S 并添加代码（参考附录 1）。

3）新建时钟配置函数文件 clock.c 并添加代码（参考附录 2）。

4）新建 UART 初始化文件 uart.c，自行添加代码。

5）新建主函数文件 main.c，自行添加代码。

6）新建 Makefile 文件并添加代码（参考附录 3）。

7）执行 make 命令编译生成二进制文件 uart.bin。

2．安装 USB 驱动。

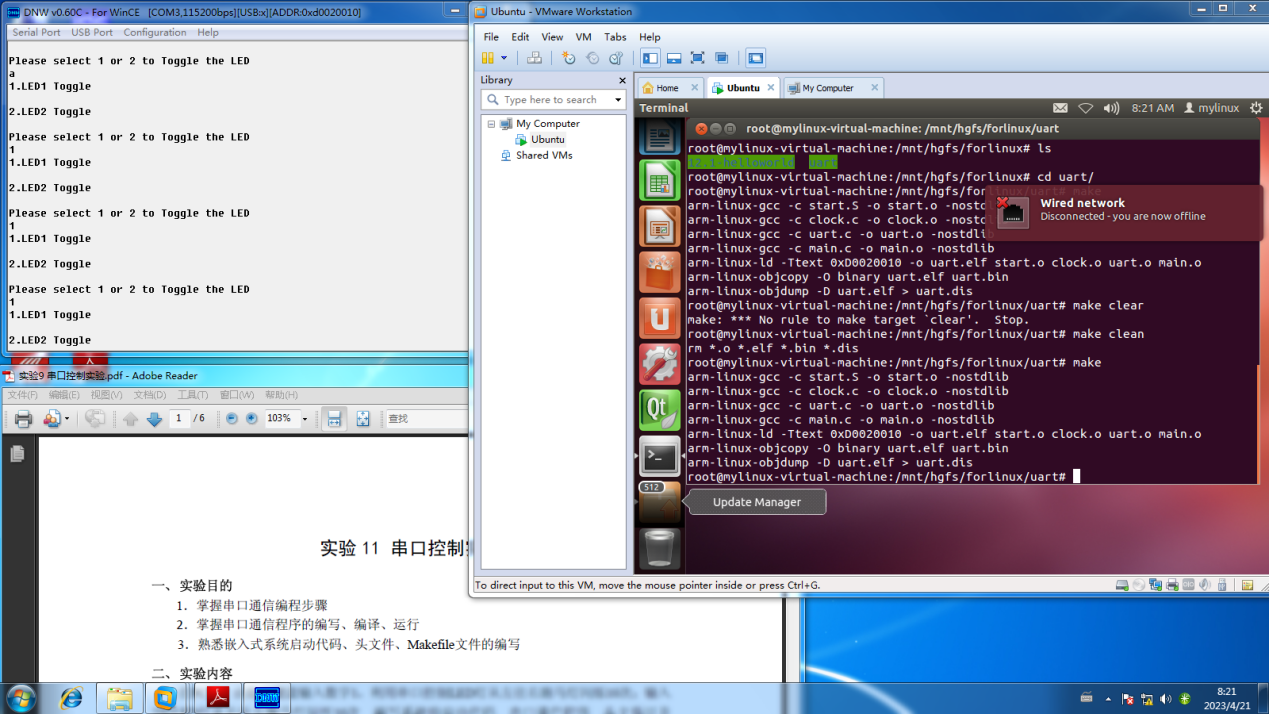
3. 使用 DNW 软件下载裸机程序 uart.bin 至 SRAM 中运行。

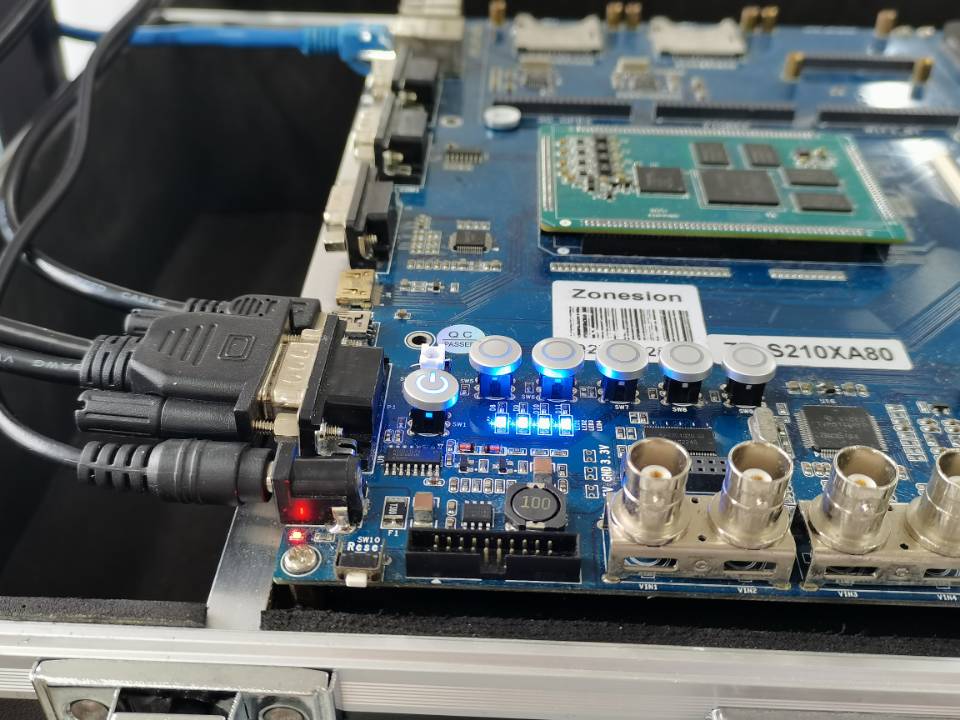
4. 修改 main.c 文件增加蜂鸣器控制功能：键盘输入“a”，蜂鸣器间隔鸣响 10

次。（蜂鸣器引脚为GPD0DAT\_2）

**五、实验报告**

1. 附主要步骤中的命令及其结果截图。







2. 归纳总结。

按实验步骤操作并无太多问题，主要注意操作权限以及驱动等方面。整体实验较为简单，过程顺利。

3.实验代码

**uart.c**

#define GPA0CON \*((volatile unsigned int \*)0xE0200000)

#define ULCON0 \*((volatile unsigned int \*)0xE2900000)

#define UCON0 \*((volatile unsigned int \*)0xE2900004)

#define UFCON0 \*((volatile unsigned int \*)0xE2900008)

#define UTRSTAT0 \*((volatile unsigned int \*)0xE2900010)

#define UTXH0 \*((volatile unsigned int \*)0xE2900020)

#define URXH0 \*((volatile unsigned int \*)0xE2900024)

#define UBRDIV0 \*((volatile unsigned int \*)0xE2900028)

#define UDIVSLOT0 \*((volatile unsigned int \*)0xE290002C)

/\*

\*\* UART0初始化

\*/

void uart\_init()

{

/\*

\*\* 配置GPA0\_0为UART\_0\_RXD

\*\* 配置GPA0\_1为UART\_0\_TXD

\*/

GPA0CON &= ~0xFF;

GPA0CON |= 0x22; //0b 00100010

/\* 8-bits/One stop bit/No parity/Normal mode operation \*/

/\*每次8位，1个停止位，无奇偶验证，正常发送模式（非红外）\*/

ULCON0 = 0x3 | (0 << 2) | (0 << 3) | (0 << 6); // 0b 0 0xx 0 11

/\* Interrupt request or polling mode/Normal transmit/Normal operation/PCLK/\*/

/\*发送和接受引脚采用中断和轮询查询模式，正常发送，常规操作，时钟选择为PCLK\*/

UCON0 = 1 | (1 << 2) | (0 << 10);

/\* 禁止FIFO \*/

UFCON0 = 0;

/\*

\*\* 波特率计算：115200bps

\*\* PCLK = 66MHz

\*\* DIV\_VAL = (66000000/(115200 x 16))-1 = 35.8 - 1 = 34.8

\*\* UBRDIV0 = 34 (DIV\_VAL的整数部分)

\*\* (num of 1's in UDIVSLOTn)/16 = 0.8 (DIV\_VAL的小数部分)

\*\* (num of 1's in UDIVSLOTn) = 12

\*\* UDIVSLOT0 = 0xDDDD (在数据手册上880页查表)

\*/

UBRDIV0 = 34;//波特率分度值

UDIVSLOT0 = 0xDDDD;

}

void uart\_send\_byte(unsigned char byte)

{

while (!(UTRSTAT0 & (1 << 2))); /\* 等待发送缓冲区为空 \*/

UTXH0 = byte; /\* 发送一字节数据 \*/

}

unsigned char uart\_recv\_byte()

{

while (!(UTRSTAT0 & 1)); /\* 等待接收缓冲区有数据可读 \*/

return URXH0; /\* 接收一字节数据 \*/

}

void uart\_send\_string(char \*str)

{

char \*p = str;

while (\*p)

uart\_send\_byte(\*p++);

}

**main.c**

#define GPJ2CON (\*(volatile unsigned long \*) 0xE0200280)

#define GPJ2DAT (\*(volatile unsigned long \*) 0xE0200284)

#define GPD0CON (\*(volatile unsigned long \*)0xE02000A0)

#define GPD0DAT (\*(volatile unsigned long \*)0xE02000A4)

extern void uart\_send\_byte(unsigned char byte);

extern unsigned char uart\_recv\_byte();

extern void uart\_send\_string(char \*str);

void delay(volatile unsigned int t)

{

volatile unsigned int t2 = 0xFFFF;

while (t--)

for (; t2; t2--);

}

void led\_init()

{

GPJ2CON &= ~(0xFF << 0);

GPJ2CON |= ((0x01 << 0) | (0x01 << 4) | (0x01 << 8) | (0x01 << 12));//((0x01 << 0) | (0x01 << 4));

GPJ2DAT |= (0xFF << 0);

}

int main()

{

char byte;

led\_init();

uart\_send\_string("\r\nUART Test in S5PV210\r\n");

while (1)

{

uart\_send\_string("\r\n1.LED1 Toggle\r\n");

uart\_send\_string("\r\n2.LED2 Toggle\r\n");

uart\_send\_string("\r\nPlease select 1 or 2 to Toggle the LED\r\n");

byte = uart\_recv\_byte();

uart\_send\_byte(byte);

if (byte == '1')

{

GPJ2DAT ^= 1 << 3;

delay(1000);

GPJ2DAT ^= 1 << 2;

delay(1000);

GPJ2DAT ^= 1 << 1;

delay(1000);

GPJ2DAT ^= 1 << 0;

delay(2000);

GPJ2DAT = 0xFF;

}

else if (byte == '2')

{

GPJ2DAT ^= 1 << 0;

delay(1000);

GPJ2DAT ^= 1 << 1;

delay(1000);

GPJ2DAT ^= 1 << 2;

delay(1000);

GPJ2DAT ^= 1 << 3;

delay(2000);

GPJ2DAT = 0xFF;

}

delay(1000);

}

return 0;

}

**start.S**

.global \_start

\_start:

@ bl clock\_init

bl uart\_init

bl main

halt:

b halt

**clock.c**

#define APLLCON0 \*((volatile unsigned int \*)0xE0100100)

#define MPLLCON \*((volatile unsigned int \*)0xE0100108)

#define EPLLCON0 \*((volatile unsigned int \*)0xE0100110)

#define VPLLCON \*((volatile unsigned int \*)0xE0100120)

#define CLK\_SRC0 \*((volatile unsigned int \*)0xE0100200)

#define CLK\_DIV0 \*((volatile unsigned int \*)0xE0100300)

#define CLK\_DIV1 \*((volatile unsigned int \*)0xE0100304)

#define CLK\_DIV2 \*((volatile unsigned int \*)0xE0100308)

#define CLK\_DIV3 \*((volatile unsigned int \*)0xE010030C)

void clock\_init()

{

/\* 1、设置PLL\_LOCK寄存器（这里使用默认值） \*/

/\* 2、设置PLL\_CON寄存器（使用芯片手册推荐的值） \*/

APLLCON0 = (1 << 0) | (3 << 8) | (125 << 16) | (1 << 31); /\* FOUTAPLL = 1000MHz \*/

MPLLCON = (1 << 0) | (12 << 8) | (667 << 16) | (1 << 31); /\* FOUTMPLL = 667MHz \*/

EPLLCON0 = (2 << 0) | (3 << 8) | (48 << 16) | (1 << 31); /\* FOUTEPLL = 96MHz \*/

VPLLCON = (3 << 0) | (6 << 8) | (108 << 16) | (1 << 31); /\* FOUTVPLL = 54MHz \*/

/\* 3、选择PLL为时钟输出 \*/

/\* MOUT\_MSYS = SCLKAPLL = 1000MHz

\*\* MOUT\_DSYS = SCLKMPLL = 667MHz

\*\* MOUT\_PSYS = SCLKMPLL = 667MHz

\*/

CLK\_SRC0 = (1 << 0) | (1 << 4) | (1 << 8) | (1 << 12);

/\* 4、设置系统时钟分频值 \*/

/\* freq(ARMCLK) = MOUT\_MSYS / (APLL\_RATIO + 1) = 1000MHz / (0 + 1) = 1000MHz

\*\* freq(HCLK\_MSYS) = ARMCLK / (HCLK\_MSYS\_RATIO + 1) = 1000MHz / (4 + 1) = 200MHz

\*\* freq(PCLK\_MSYS) = HCLK\_MSYS / (PCLK\_MSYS\_RATIO + 1) = 200MHz / (1 + 1) = 100MHz

\*\* freq(HCLK\_DSYS) = MOUT\_DSYS / (HCLK\_DSYS\_RATIO + 1) = 667 / (3 + 1) = 166MHz

\*\* freq(PCLK\_DSYS) = HCLK\_DSYS / (PCLK\_DSYS\_RATIO + 1) = 166 / (1 + 1) = 83MHz

\*\* freq(HCLK\_PSYS) = MOUT\_PSYS / (HCLK\_PSYS\_RATIO + 1) = 667 / (4 + 1) = 133MHz

\*\* freq(PCLK\_PSYS) = HCLK\_PSYS / (PCLK\_PSYS\_RATIO + 1) = 133 / (1 + 1) = 66MHz

\*/

CLK\_DIV0 = (0 << 0) | (4 << 8) | (1 << 12) | (3 << 16) | (1 << 20) | (4 << 24) | (1 << 28);

}

**Makefile**

uart.bin: start.o clock.o uart.o main.o

arm-linux-ld -Ttext 0xD0020010 -o uart.elf $^

arm-linux-objcopy -O binary uart.elf $@

arm-linux-objdump -D uart.elf > uart.dis

%.o : %.c

arm-linux-gcc -c $< -o $@ -nostdlib

%.o:%.S

arm-linux-gcc -c $< -o $@ -nostdlib

clean:

rm \*.o \*.elf \*.bin \*.dis