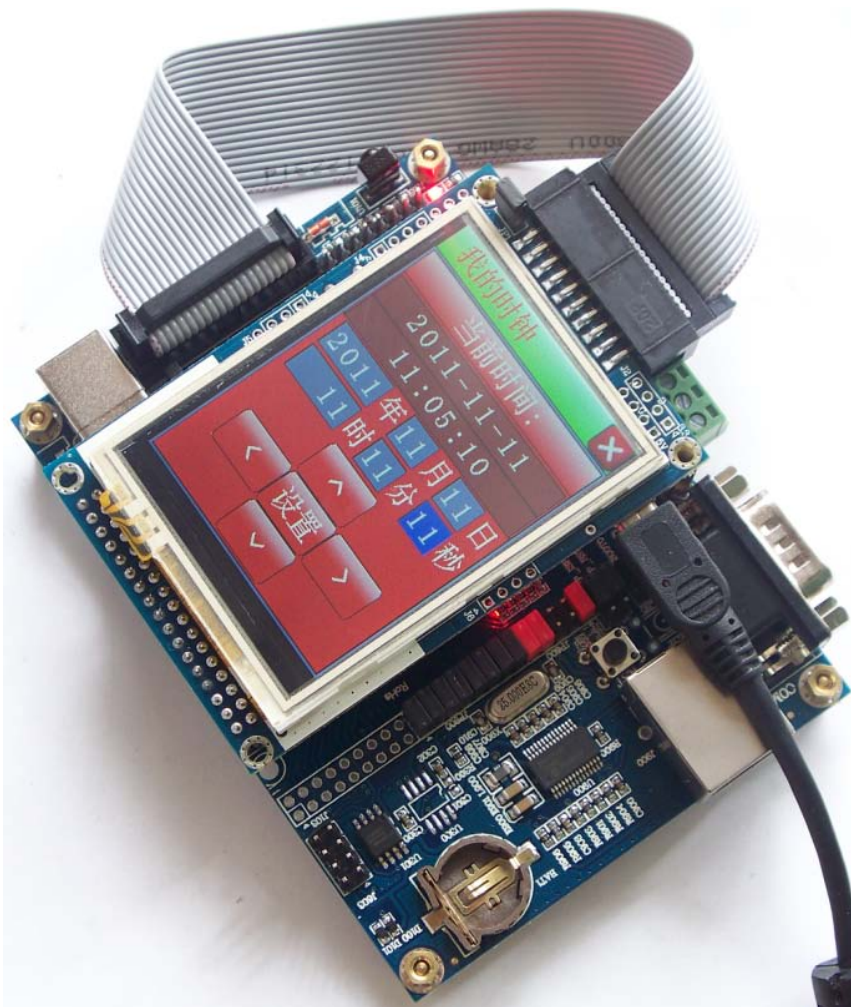




BHS-STM32 实验指导



本文是入门教程，高手请飘过...

提供的例程有使用 MDK 配置文件初始化芯片外设，也有使用 ST 库函数 STM32F10xR.LIB，选择哪种方式完全看你个人习惯。

串口使用工具使用 SSCOM3.2 或者系统自带的超级终端，你自己也可以使用其他串口工具

QQ: 958664258

211C 用户名: banhushui

交流平台: <http://blog.21ic.com/user1/5817/index.html>

Email: banhushui@163.com

淘宝店铺: <http://shop58559908.taobao.com>

半壶水



目录

目录	2
入门例程	3
实验 1: Blinky 闪灯	3
实验 2: Timer 定时器	4
实验 3: Tamper 侵入检测	4
实验 4: RTC 实时时钟	4
实验 5: IWDG 看门狗	5
实验 6: EXTI 外部中断	5
实验 7: PWM_1 固定占空比	5
实验 8: PWM_2 可变占空比	6
实验 9: USART_Pol 轮询方式	7
实验 10: USART_Irq 中断方式	8
实验 11: CAN	8
前后台例程	9
实验 12: SysTick_LED 点灯	9
实验 13: PWM	10
实验 14: 红外	10
实验 15: TFT 测试	10
实验 16: USART 简单	11
实验 17: USART 协议	11
实验 18: USART 协议 RS485	12
实验 19: USART 协议 RS485 RTC	12
实验 20: USART 协议 RS485 RTC Flash TFT	12
实验 21: SD_File_TFT (从 SD 卡读取图片显示)	12
实验 22: 网页控制 LED (BHS-STM2 V1.1)	15
实验 23: IAP 在线升级	16
RTX 例程	17
实验 24: RTX 之 TCP uIP 1.0 (BHS-STM32V1.1)	17
实验 25: RTX_HID	19
实验 26: RTX-CAN	19
实验 27: RTX 之 邮箱+信号量	19

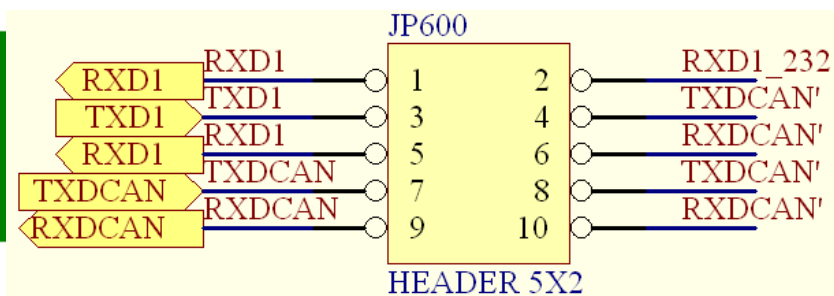
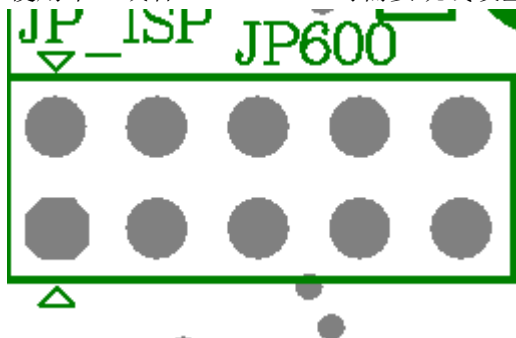


入门例程

入门例程在 \BHS-STM32 程序\MCBSTM32 移植到BHS-STM32 (入门例程 文件夹里面, 入门例程使用MDK配置文件初始化芯片外设。MDK配置文件说明请参见《UV3.chm》和我写的另外一篇文章:《MDK配置向导详解》

实验 1: Blinky 闪灯

使用串口或者 CAN/RS485 时需要跳线设置

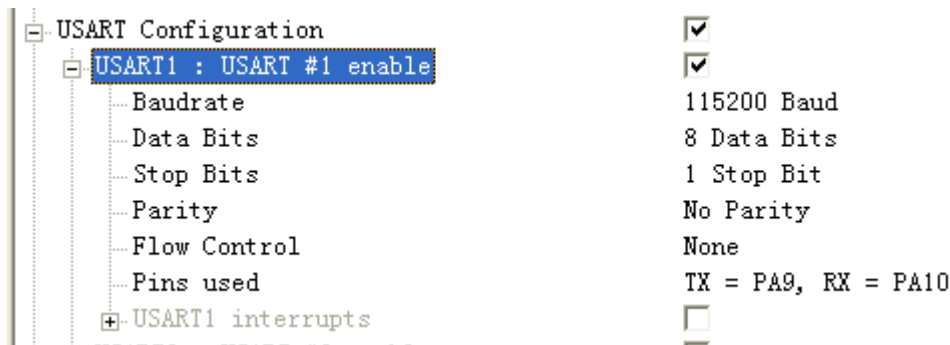


注意:

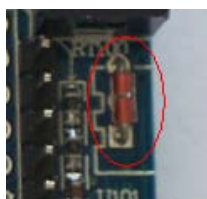
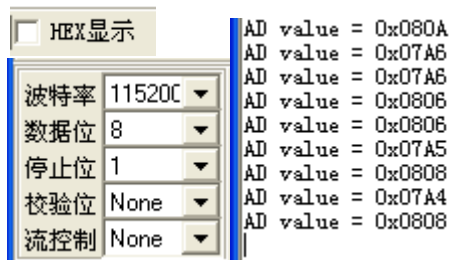
1. 使用 RS485, 请将 JP600 的 1,2 7,8 9,10 断开, 3,4 5,6 短接
2. 使用 RS232, 请将 JP600 的 3,4 5,6 断开, 1,2 短接
3. 使用 CAN, 请将 JP600 的 1,2 3,4 5,6 断开, 7,8 9,10 短接
4. 本板 RS485 是使用 CAN 芯片实现的, 所以串口发送时, 禁止接收 (参考例子)

本例子实现 LED 跑马灯功能, 并且通过串口发送 AD0 的值, AD0 连接一个 NTC 温度电阻
STM32_Init.c 文件中串口 1 配置如下:

STM32_Init.c 是 STM32 外设配置文件, 所有外设初始化都可以使用该配置文件

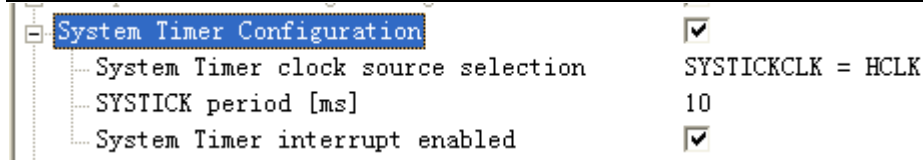


串口工具设置如下:



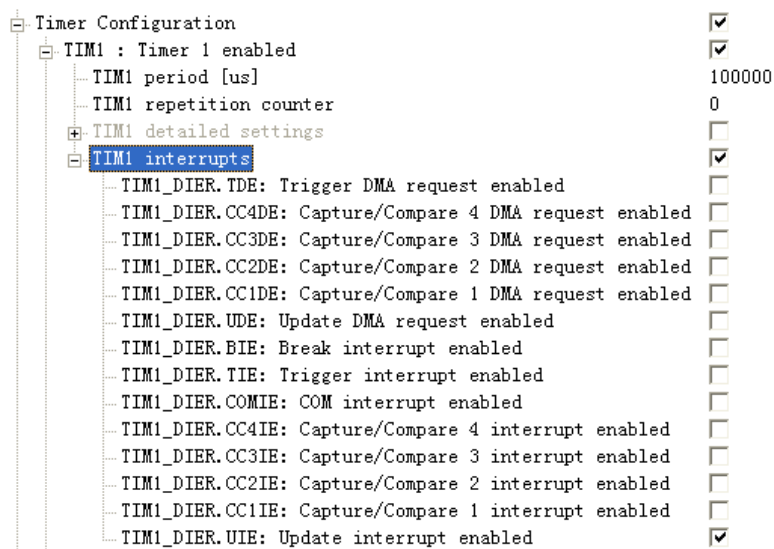
显示数据, 用手触摸 NTC 温度电阻数字将变化

本例中使用系统时钟定时, 定时时间 10ms



实验 2: Timer 定时器

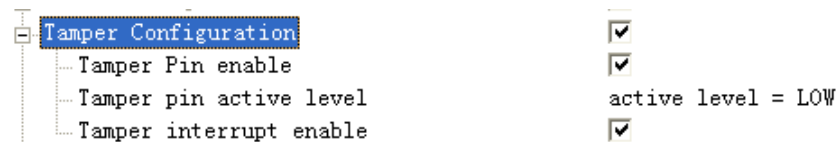
本例子实现 LED 跑马灯功能，与上例不同的是定时器使用 TIM1



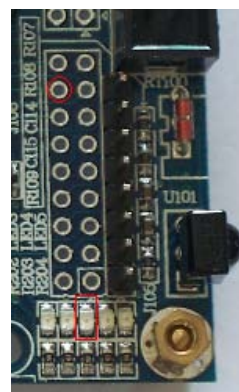
通过修改 TIM1 period[us]可以修改定时时间，从而修改 LED 闪烁频率

实验 3: Tamper 侵入检测

本例子实现侵入检测功能，该功能脚位于 PC13 上，可设置为低电平/高电平触发侵入检测配置

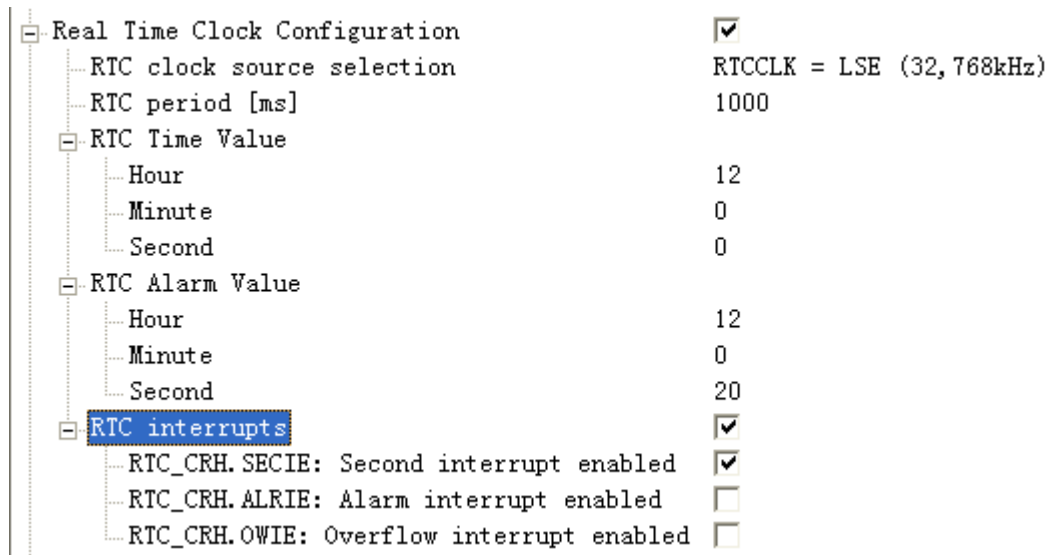


本例设置为低电平触发，当检测到侵入时(图中红圈对地短接)LED3(红框内)灭



实验 4: RTC 实时时钟

RTC 实时时钟配置



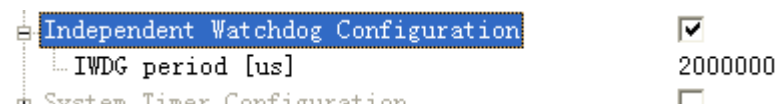
该例子用 RTC 产生 1 秒中断，驱动 LED 闪烁

STM32 的 RTC 实际是一个 32 位的计数器，要得到时分秒信号需要自己转换，可以参考我 BHS-GUI 里面的例子

实验 5: IWDG 看门狗

本例子实现看门狗功能

看门狗配置如下：

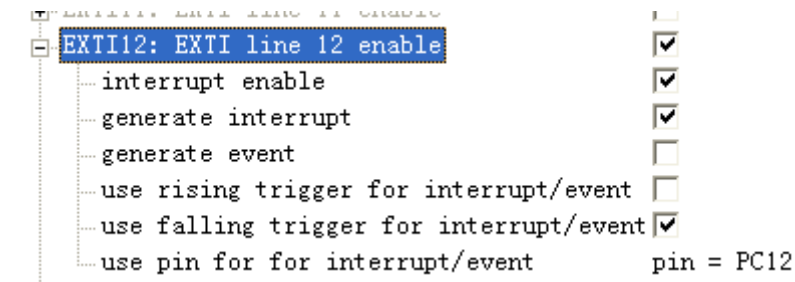


使用该例程时，板上启动模式要与调试模式保持一致，可以看到板上 LED 闪烁一段时间后系统将复位

实验 6: EXTI 外部中断

本例子实现外部中断功能

中断配置如下



例子使用 PC12, 由于 PC12 也连接红外接收，所以用红外遥控器对着接收头按下按键将看到 LED2 闪烁

实验 7: PWM_1 固定占空比

配置如下



[-] Timer Configuration	<input checked="" type="checkbox"/>
[+] TIM1 : Timer 1 enabled	<input type="checkbox"/>
[+] TIM2 : Timer 2 enabled	<input type="checkbox"/>
[+] TIM3 : Timer 3 enabled	<input type="checkbox"/>
[-] TIM4 : Timer 4 enabled	<input checked="" type="checkbox"/>
TIM4 period [us]	1000
[-] TIM4 detailed settings	<input checked="" type="checkbox"/>
TIM4.PSC: Timer 4 Prescaler	7199
TIM4.ARR: Timer 4 Auto-reload	9999
[+] Timer 4 Control Register 1 Configuration (TIM4_CR1)	
[+] Timer 4 Control Register 2 Configuration (TIM4_CR2)	
[+] Timer 4 Slave mode control register Configuration (TIM4_SMC)	
[+] Channel 1 Configuration	
[+] Channel 2 Configuration	
[-] Channel 3 Configuration	
[+] Channel configured as output	
[+] Channel configured as input	
TIM4_CCR3: Capture/compare register 3	5000
[-] Channel 4 Configuration	
[+] Channel configured as output	
[+] Channel configured as input	
TIM4_CCR4: Capture/compare register 4	2500
[-] TIM4 interrupts	<input type="checkbox"/>

例子使用 TIM4,的通道 3, 通道 4 产生 2 个固定占空比的 PWM 脉冲, 其中

通道 3 (PB8) 产生占空比 50% 的 PWM 脉冲

通道 4 (PB9) 产生占空比 25% 的 PWM 脉冲

本例最好使用示波器观察不同的占空比波形

实验 8: PWM_2 可变占空比

配置如下



<input checked="" type="checkbox"/> TIM4 : Timer 4 enabled	<input checked="" type="checkbox"/>
TIM4 period [us]	1000
<input checked="" type="checkbox"/> TIM4 detailed settings	<input checked="" type="checkbox"/>
TIM4.PSC: Timer 4 Prescaler	7199
TIM4.ARR: Timer 4 Auto-reload	99
+ Timer 4 Control Register 1 Configuration (TIM4_CR1)	
+ Timer 4 Control Register 2 Configuration (TIM4_CR2)	
+ Timer 4 Slave mode control register Configuration (TIM4_SMC)	
+ Channel 1 Configuration	
+ Channel 2 Configuration	
- Channel 3 Configuration	
+ Channel configured as output	
+ Channel configured as input	
TIM4_CCR3: Capture/compare register 3	5000
- Channel 4 Configuration	
+ Channel configured as output	
+ Channel configured as input	
TIM4_CCR4: Capture/compare register 4	2500
<input checked="" type="checkbox"/> TIM4 interrupts	<input checked="" type="checkbox"/>
TIM4_DIER.TDE: Trigger DMA request enabled	<input type="checkbox"/>
TIM4_DIER.CC4DE: Capture/Compare 4 DMA request enabled	<input type="checkbox"/>
TIM4_DIER.CC3DE: Capture/Compare 3 DMA request enabled	<input type="checkbox"/>
TIM4_DIER.CC2DE: Capture/Compare 2 DMA request enabled	<input type="checkbox"/>
TIM4_DIER.CC1DE: Capture/Compare 1 DMA request enabled	<input type="checkbox"/>
TIM4_DIER.UDE: Update DMA request enabled	<input type="checkbox"/>
TIM4_DIER.TIE: Trigger interrupt enabled	<input type="checkbox"/>
TIM4_DIER.CC4IE: Capture/Compare 4 interrupt enabled	<input type="checkbox"/>
TIM4_DIER.CC3IE: Capture/Compare 3 interrupt enabled	<input type="checkbox"/>
TIM4_DIER.CC2IE: Capture/Compare 2 interrupt enabled	<input type="checkbox"/>
TIM4_DIER.CC1IE: Capture/Compare 1 interrupt enabled	<input type="checkbox"/>
TIM4_DIER.UIE: Update interrupt enabled	<input checked="" type="checkbox"/>

例子使用 TIM4,的通道 3, 通道 4 产生 2 个可变占空比的 PWM 脉冲, 其中通道 3 (PB8) 通道 4 (PB9)

本例最好使用示波器观察不同的占空比波形

本例可以看到与上例不同是占空比在变化。

实验 9: USART_Pol 轮询方式

轮询方式就是不断查询串口状态, 看串口是否收到数据

串口 1 配置如下

<input checked="" type="checkbox"/> USART Configuration	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> USART1 : USART #1 enable	<input checked="" type="checkbox"/>
Baudrate	115200 Baud
Data Bits	8 Data Bits
Stop Bits	1 Stop Bit
Parity	No Parity
Flow Control	None
Pins used	TX = PA9, RX = PA10
+ USART1 interrupts	<input type="checkbox"/>

串口工具设置如下:

☐ HEX 显示



波特率	115200
数据位	8
停止位	1
校验位	None
流控制	None

本串口程序是接收到什么字符就返回什么字符

Polling mode Serial I/O Example

```
Press a key. 1
You pressed '1'.

Press a key. 2
You pressed '2'.

Press a key. 3
You pressed '3'.

Press a key. a
You pressed 'a'.

Press a key. c
You pressed 'c'.

Press a key.
```

实验 10: USART_Irq 中断方式

中断方式不再查询串口状态, 这样 CPU 效率将显著提高

串口 1 配置如下

USART Configuration		<input checked="" type="checkbox"/>
USART1 : USART #1 enable		<input checked="" type="checkbox"/>
Baudrate	115200 Baud	
Data Bits	8 Data Bits	
Stop Bits	1 Stop Bit	
Parity	No Parity	
Flow Control	None	
Pins used	TX = PA9, RX = PA1	
USART1 interrupts		<input checked="" type="checkbox"/>
USART1_CR1.IDLEIE: IDLE Interrupt enable		<input type="checkbox"/>
USART1_CR1.RXNEIE: RXNE Interrupt enable		<input checked="" type="checkbox"/>
USART1_CR1.TCIE: Transmission Complete Interrupt enable		<input type="checkbox"/>
USART1_CR1.TXEIE: TXE Interrupt enable		<input checked="" type="checkbox"/>
USART1_CR1.PEIE: PE Interrupt enable		<input type="checkbox"/>
USART1_CR2.LBDIE: LIN Break Detection Interrupt enable		<input type="checkbox"/>
USART1_CR3.EIE: Error Interrupt enable		<input type="checkbox"/>
USART1_CR3.CTSIE: CTS Interrupt enable		<input type="checkbox"/>

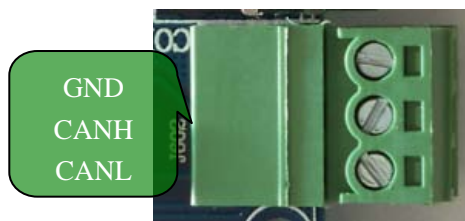
该程序功能与轮询方式例子一样只是实现方式不同而已

实验 11: CAN

CAN 实验需要两个带 CAN 接口的板子

准备工作, 准备两个 CAN 板子, 连接两个 CAN 接口, 两个板子都下载本程序。

本实验通过 CAN 总线控制另外一个板子的 LED2~LED5 闪烁的例子, 如果通信正常可以看到两个板子的 LED 都再闪烁, 如果通信失败或者断开通信线, LED 不再闪烁





前后台例程

前后台例程在 [\BHS-STM32 程序\前后台](#) 文件夹里面，这个文件夹的例程不再使用MDK配置文件初始化芯片外设，而是使用ST库函数STM32F10xR.LIB，因为你的安装路径和我的可能不同，所以需要修改库文件路径，方法请参考《STM32 新手入门》，STM32F10xR.LIB路径在MDK安装目录下：[\ARM\RV31\LIB\ST](#)，另外该库的源文件路径在[\ARM\RV31\LIB\ST\STM32F10x](#)；库文件说明参考《STM32F101xx与STM32F103xx固件函数库用户手册》

实验 12：SysTick_LED 点灯

同样是点灯的程序，本例使用系统时钟定时，系统初始化函数采用 ST 库函数：

```
void SysTick_Init(void)
```

```
{

    /* SysTick end of count event each 1ms with input clock equal to 9MHz (HCLK/8, default) */
    SysTick_SetReload(9000);
    /* Enable SysTick interrupt */
    SysTick_ITConfig(ENABLE);
    /* Enable the SysTick Counter */
    SysTick_CounterCmd(SysTick_Counter_Enable);
}

//下面是【实验 1】的系统时钟初始函数，要想弄明白请先了解 STM32 的寄存器
#define __SYSTICK_SETUP          1
#define __SYSTICK_CTRL_VAL      0x00000006
#define __SYSTICK_PERIOD        0x0000000A

#if __SYSTICK_SETUP
/*-----
STM32 System Timer setup.
initializes the SysTick register
*-----*/
__inline static void stm32_SysTickSetup (void) {

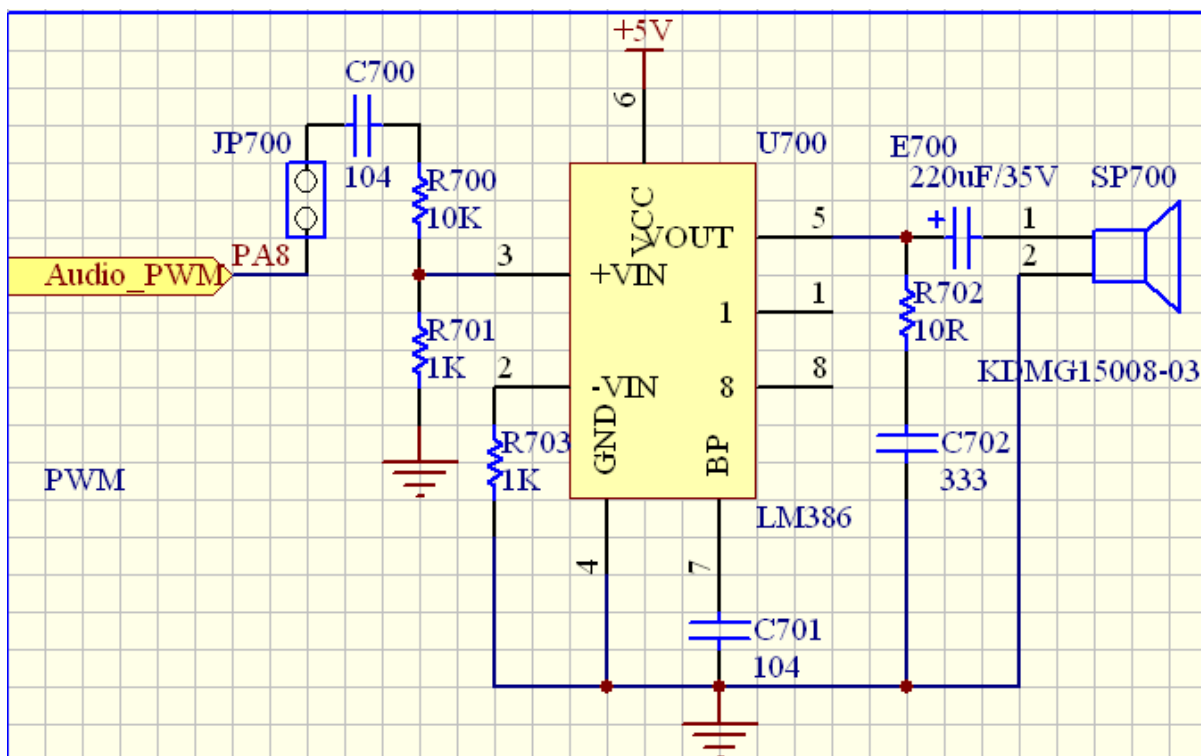
    if ((__SYSTICK_PERIOD*(__SYSTICKCLK/1000)-1) > 0xFFFFFFFF) // reload value too large
        #error "Reload Value too large! Please use 'HCLK/8' as System Timer clock source or smaller period"
    else
        SysTick->LOAD = __SYSTICK_PERIOD*(__SYSTICKCLK/1000)-1; // set reload register
        SysTick->CTRL = __SYSTICK_CTRL_VAL; // set clock source and Interrupt enable

        SysTick->VAL = 0; // clear the counter
        SysTick->CTRL |= SYSTICK_CSR_ENABLE; // enable the counter
    #endif
} // end of stm32_SysTickSetup
#endif
```

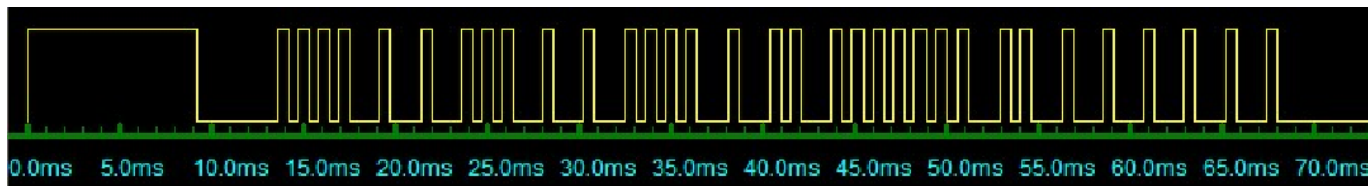


实验 13: PWM

本例程的 PWM 使用 PA8 直接驱动扬声器，运行程序可以听到扬声器声音在不断变化



实验 14: 红外



说明：图中波形是反向的

//红外数据格式

同步头 + 8bit 用户码 + 8bit 用户码反码 + 8bit 数据 + 8bit 数据反码

//同步头： 9ms 低电平，4.5ms 高电平

数据：

//0.5ms 低电平,0.5ms 高电平 ==BIT 0

//0.5ms 低电平,1ms 高电平 ==BIT 1

连续码：

9ms 低电平+2.2ms 高电平+0.6ms 低电平

上面的时间是个大概值，不一定精确

*/

接收到红外信号 LED 将交替亮灭

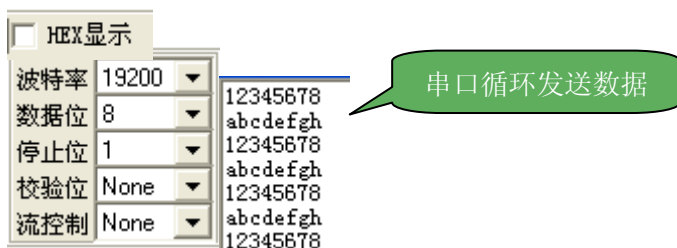
实验 15: TFT 测试

本例是简单的 TFT 测试，只是输出单色测试 TFT 模块是否正常工作，如果工作支持可以看到 TFT 模块分别显示红绿蓝颜色



实验 16: USART 简单

本例是简单串口程序
串口工具设置如下:



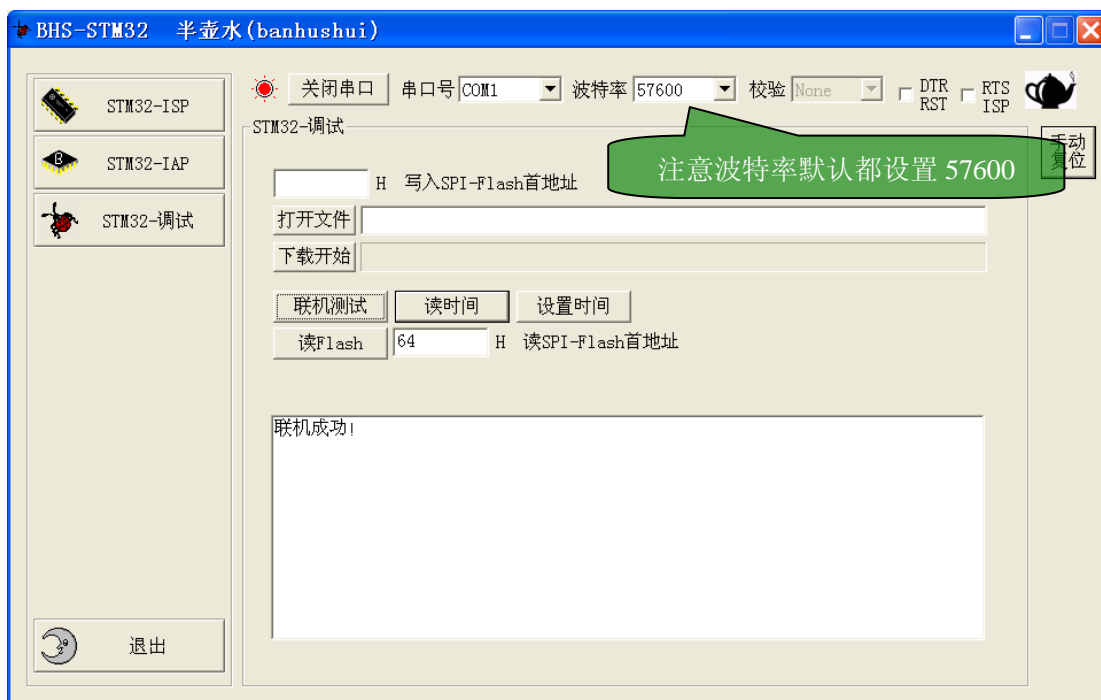
通过调用 ST 库函数配置串口

```
void USART_InitConfig(uint32 BaudRate)
{
    USART_InitTypeDef USART_InitStructure;
    USART_InitStructure.USART_BaudRate = BaudRate;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
    USART_InitStructure.USART_Parity = USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    USART_Init(USART1, &USART_InitStructure); /* Configure USART1 */
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE); /* Enable USART1 Receive and Transmit interrupts */
    //USART_ITConfig(USART1, USART_IT_TXE, ENABLE);
    USART_Cmd(USART1, ENABLE); /* Enable the USART1 */
}
```

USART_InitConfig(19200);

实验 17: USART 协议

本例是一个带协议的串口 RS232 程序, 通信协议详见我提供的文档《BHS-STM32 IAP 通讯协议 V0.2》
该程序需要配合 PC 软件实现, BHS-STM32-ISP-IAP V1.2

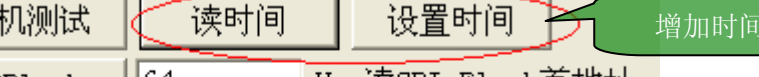


本例只支持【联机测试】功能, 其他命令都返回不支持, 下面的例程将介绍复杂的功能



本例是一个带协议的串口 RS485 程序，RS485 收发器是和 CAN 复用的，CAN，RS485 都是差分信号，CAN 发送时同时接收，所以做 485 使用时发送数据必须禁止数据接收。

实验 19: USART 协议 RS485 RTC



读时间

读Flash 64 H 读SPI-Flash首地址

2010-02-12 15:02:25





本例是一个带协议的串口 RS485 程序，在实验 19 基础上增加 SPI-FLASH 读操作，增加 TFT 驱动

[illegible]

本例是一个从 SD 卡读取图片文件显示到 TFT 上的程序，使用时将【将此文件夹里的文件复制到 SD 卡】文件夹里的文件先复制到 SD 卡上，将 SD 卡插入板子卡座内。开发板串口 1 连接电脑串口，程序将通过串口调试工具输入命令显示图片，如果提示找不到 File_Config.h 文件，请将文件夹下的 File_Config.h 复制到 MDK 安装目录 \ARM\RV31\INC

dir 显示所有文件

文件系统使用了 MDK 自带的文件系统库 FS_CM3.lib，详细说明请参见《rlarm.chm》

 USART Configuration	
 USART1 : USART #1 enable	
Baudrate	115200 Baud
Data Bits	8 Data Bits
Stop Bits	1 Stop Bit
Parity	No Parity
Flow Control	None
Pins used	TX = PA9, RX = PA10

☐ HEX显示



波特率	115200
数据位	8
停止位	1
校验位	None
流控制	None

```
+-----+
|                SD/MMC Card File Manipulation example                |
+-----+-----+
| command | function |
+-----+-----+
| disp "fname" [/A] | display pic |
| CAP "fname" [/A] | captures serial data to a file |
|                   | [/A option appends data to a file] |
| FILL "fname" [nnnn] | create a file filled with text |
|                   | [nnnn - number of lines, default=1000] |
| TYPE "fname" | displays the content of a text file |
| REN "fname1" "fname2" | renames a file 'fname1' to 'fname2' |
| COPY "fin" ["fin2"] "fout" | copies a file 'fin' to 'fout' file |
|                   | ['fin2' option merges 'fin' and 'fin2'] |
| DEL "fname" | deletes a file |
| DIR "[mask]" | displays a list of files in the directory |
| FORMAT [label [/FAT32]] | formats Flash Memory Card |
|                   | [/FAT32 option selects FAT32 file system] |
| HELP or ? | displays this help |
+-----+-----+

press any key
```

按任意键直到出现 Cmd>为止

```
+-----+
|                SD/MMC Card File Manipulation example                |
+-----+-----+
| command | function |
+-----+-----+
| disp "fname" [/A] | display pic |
| CAP "fname" [/A] | captures serial data to a file |
|                   | [/A option appends data to a file] |
| FILL "fname" [nnnn] | create a file filled with text |
|                   | [nnnn - number of lines, default=1000] |
| TYPE "fname" | displays the content of a text file |
| REN "fname1" "fname2" | renames a file 'fname1' to 'fname2' |
| COPY "fin" ["fin2"] "fout" | copies a file 'fin' to 'fout' file |
|                   | ['fin2' option merges 'fin' and 'fin2'] |
| DEL "fname" | deletes a file |
| DIR "[mask]" | displays a list of files in the directory |
| FORMAT [label [/FAT32]] | formats Flash Memory Card |
|                   | [/FAT32 option selects FAT32 file system] |
| HELP or ? | displays this help |
+-----+-----+

press any key
press any key
press any key
press any key
press any key
press any key
press any key
press any key
Cmd>
```

输入 dir 命令查看 SD 卡上文件



```
press any key
press any key
press any key
press any key
press any key
press any key
press any key
Cmd> dir
File System Directory...
DCIM          <DIR>          01.01.2008  12:00
MISC          <DIR>          01.01.2008  12:04
.system      <DIR>          27.11.2009  12:26
GG1.BIN      153.600    09.07.2009  01:34
HH1.BIN      153.120    09.07.2009  01:37
MM1.BIN      142.560    09.07.2009  01:06
MM2.BIN      115.200    09.07.2009  01:33
MM3.BIN      146.880    09.07.2009  01:48
MM4.BIN      146.880    09.07.2009  01:49
MM5.BIN      139.200    09.07.2009  01:49
7 File(s)    997.440 bytes
3 Dir(s)     1.953.759.232 bytes free.
Cmd>

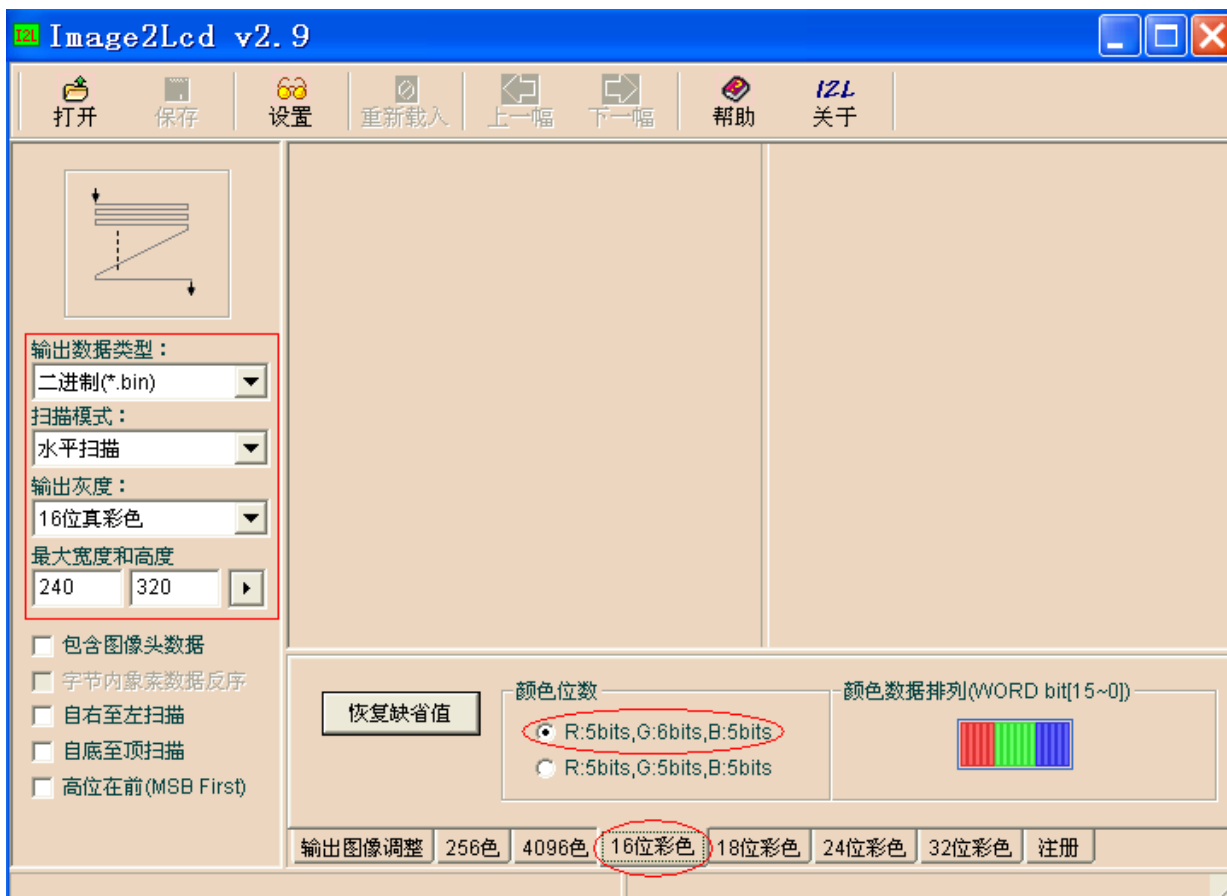
File System Directory...
DCIM
MISC
.system
GG1.BIN
HH1.BIN
MM1.BIN
MM2.BIN
MM3.BIN
MM4.BIN
MM5.BIN
7 File(s)
3 Dir(s)
Cmd> disp gg1.bin
Cmd> |
```

输入 disp gg1.bin 后看 TFT 是不是有图片显示出来了

继续输入命令

```
Cmd> disp gg1.bin
Cmd> disp mm1.bin
Cmd> disp mm2.bin
Cmd> disp mm3.bin
Cmd> disp mm4.bin
```

图片文件是经过处理的，处理工具软件：Image2Lcd V2.9，参数设置如下





有朋友说没提供文件写的例子，其实文件写也挺简单，例子如下：

```
void testWriteFile(void)
{
    FILE *file;
    char *fname="test.txt";
    U8 buf8[100];
    U32 buf32[100]={0, 0xaa55, 0x55aa};

    file = fopen (fname, "a");           /* open a file for writing          */
    if (file == NULL)
    {

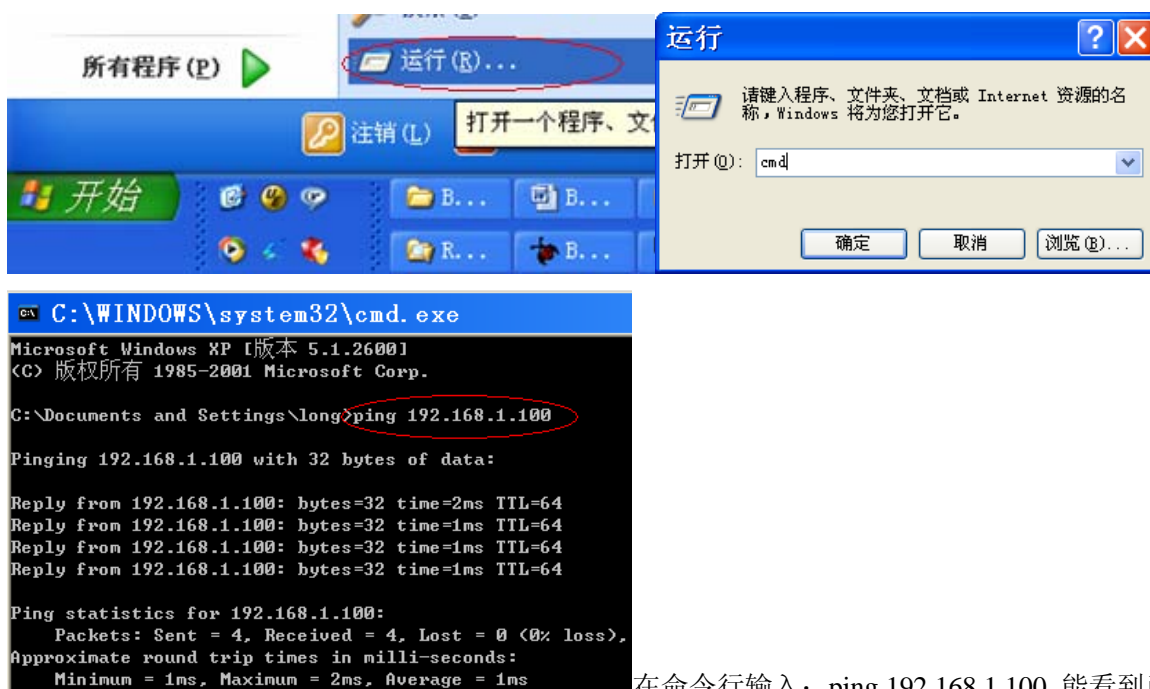
    }
    else
    {
        memset(buf8, 0, 10);
        memset(buf8+10, 0xaa, 10);
        memset(buf8+20, 0x55, 10);
        fwrite (buf8, 1, 30, file);
        fwrite (buf8, 1, 2, file);
        fwrite (buf32, 4, 3, file);
        fclose (file);                  /* close the output file          */
    }
}
```

实验 22：网页控制 LED(BHS-STM2 V1.1)

本例是一个简单的 WEB 程序，通过网页控制 LED 的亮灭，程序默认 IP 是 192.168.1.100

要使用该例程需要你将电脑 IP 设置为同一个网段：192.168.1.xxx

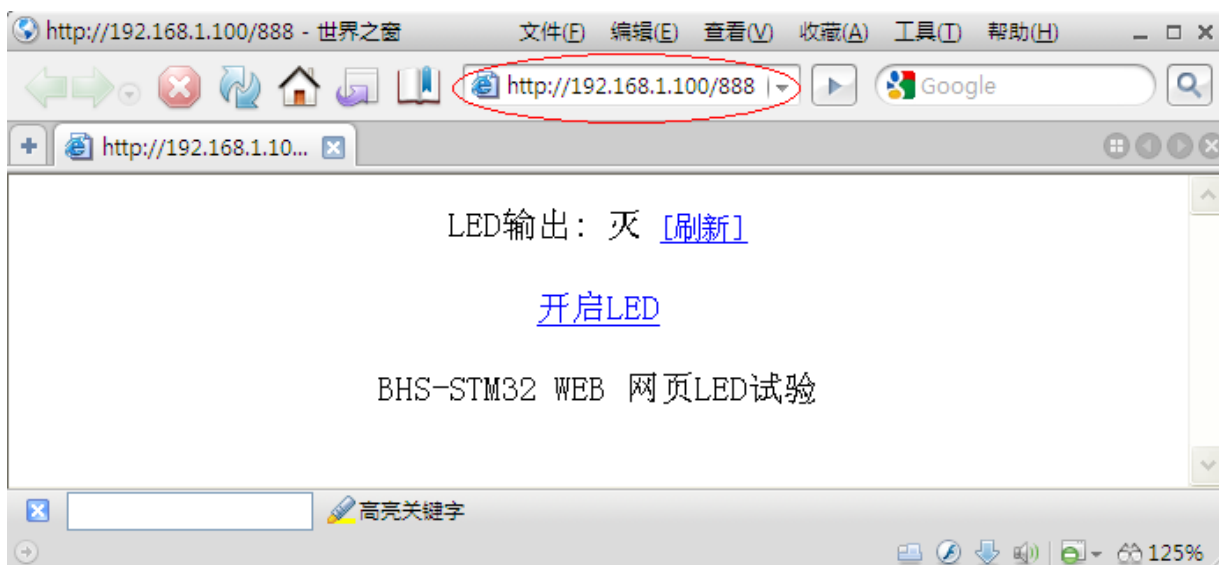
使用提供的交叉网线连接好开发板和电脑的网口



在命令行输入：ping 192.168.1.100 能看到已经连接上网络了



在IE地址蓝输入: <http://192.168.1.100/888> 将打开如下网页, 现在可以同网页控制LED了



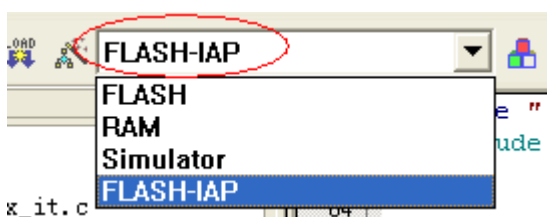
实验 23: IAP 在线升级

本例演示如何在线升级, 通信协议详见我提供的文档《BHS-STM32 IAP 通讯协议 V0.2》

[USART 协议 iapBootloader]仅支持 RS232, [USART 协议 RS485iapBootloader]支持 RS232 和 RS485

将[USART 协议 RS485iapBootloader\ObjFlash]下的 HEX 文件编程到 STM32,

用户程序选择【USART 协议 RS485】做测试, 选择 FLASH-IAP 编译, 在 ObjFlashIAP 文件下将输出 stm32-IAP-TEST(已加密用户程序).bin 文件



仍然使用我提供的工具软件



IAP 原理如下:

在 STM32 0x8000000 地址写入一个引导程序, 该程序通过串口将用户程序下载到 FLASH 中成功后, 立即跳转到用户程序运行。本实验属于 STM 深入应用, 需要先了解 STM32 结构才能更容易成功。刚上手的用户建议先跳过



RTX 例程

RTX例程在 [\BHS-STM32 程序\RTX](#) 文件夹里面, 这个文件夹的例程主要是基于MDK自带的操作系统的应用, RTX官方文档在安装路径的HLP文件下《rlarm.chm》做了详细介绍, 光盘里也有个中文版的, 要使用RTX的朋友请先阅读该文档。

实验 24: RTX 之 TCP uIP 1.0 (BHS-STM32V1.1)

本例是移植的开源的 uIP1.0

```
C:\Documents and Settings\long>ping 192.168.1.100

Pinging 192.168.1.100 with 32 bytes of data:

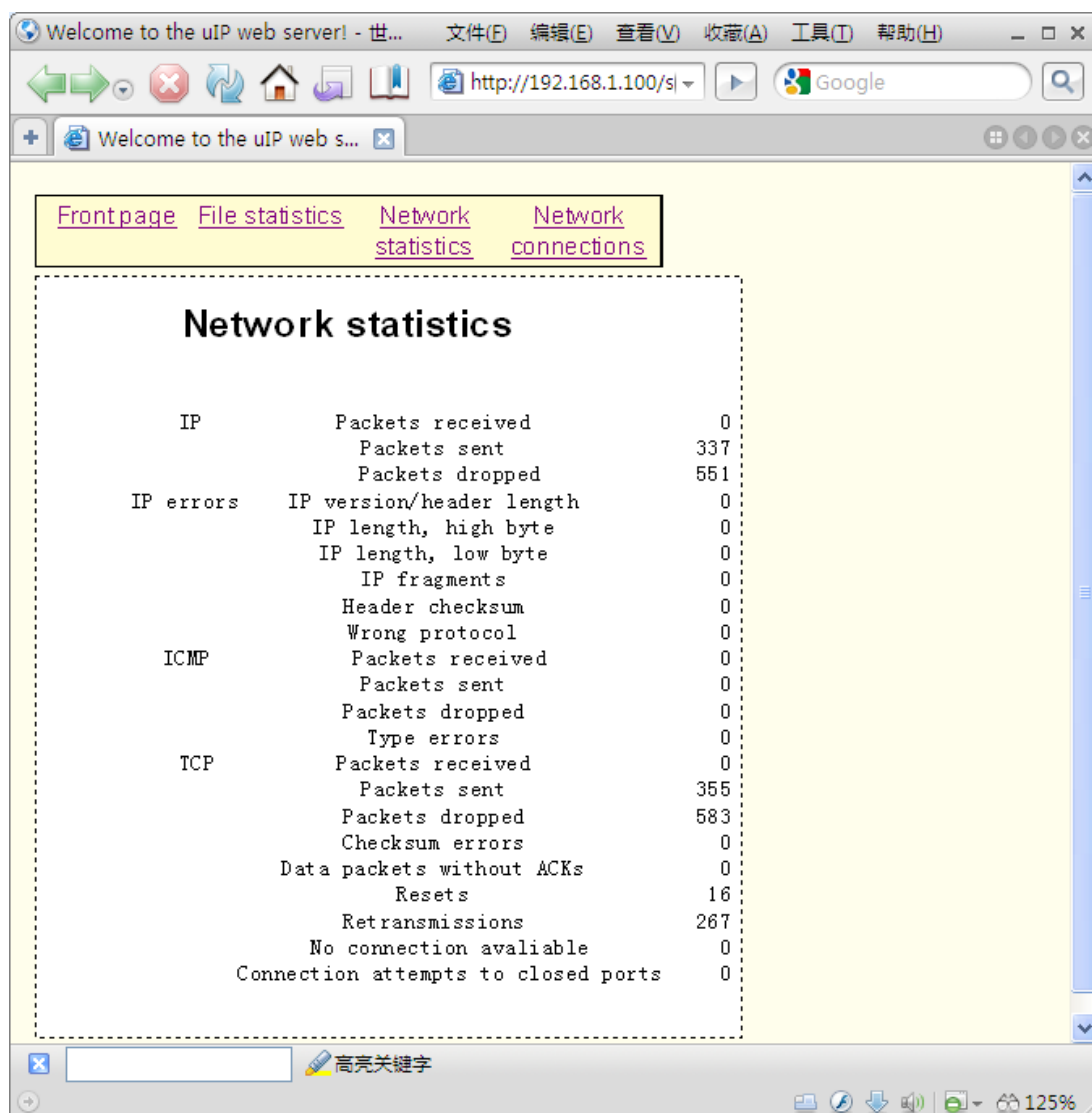
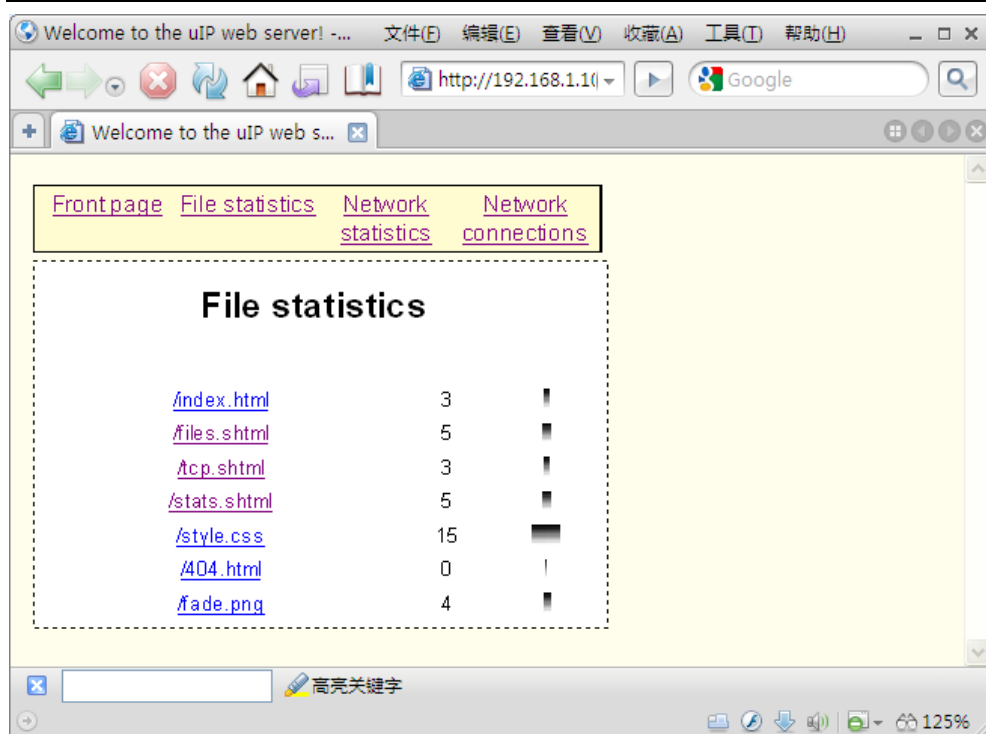
Reply from 192.168.1.100: bytes=32 time=2ms TTL=128
Reply from 192.168.1.100: bytes=32 time=1ms TTL=128
Reply from 192.168.1.100: bytes=32 time=1ms TTL=128
Reply from 192.168.1.100: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms
```

在命令行输入: ping 192.168.1.100 能看到已经连接上网络了

在IE地址蓝输入: <http://192.168.1.100>

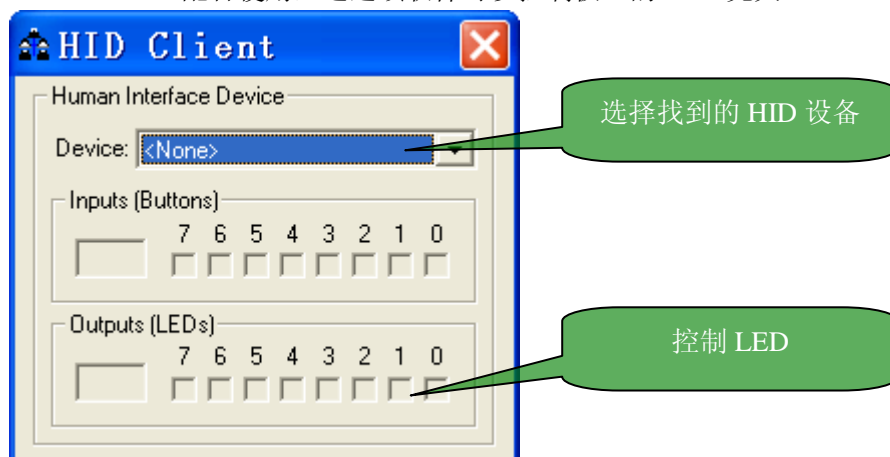






实验 25: RTX_HID

本例是 USB-HID 的应用, HID 无需安装驱动, 因为操作系统都带了 HID 的驱动了, 该例子需要 PC 软件 HIDClient.exe 配合使用, 通过该软件可以控制板上的 LED 亮灭



实验 26: RTX-CAN

本例是与实验 11 可以通信, 实验方法完全同实验 11, 请参考实验 11

实验 27: RTX 之邮箱+信号量

本程序完全用模拟仿真查看结果, 选择此项编译软件仿真
例程演示多任务之间怎样传递数据



说明, RTX 例程功能基本同前后台例程功能相同, 只是利用 MDK 自带的操作系统实现
关于 UCOS, UCGUI, freeRTOS 等这里不做介绍, 这些都是比较复杂的应用, 光盘有相关资料文档。