ARC

串口

支持	www.armrunc.com http://armrunc.taobao.com

Change History

版本号	更改描述	作者	修改时间
1.0.0	第一版	armrunc	2012-3-6

目录

第5章 串口

5.1 串口简介

串口为一种串行通讯总线，用于异步通讯，在嵌入式开发调试中很重要。在任何双向异步通信中，硬件上至少需要两根线，输入数据线(RX),输出数据线(TX). 通过这两个线，串行数据以如下数据帧发送和接收，

STM32 的串口为全双工的异步通信，采用分数波特率发生系统，发送和接收波特率相同，最大速率到 4.5 MBits/s，数据长度可编程为 8 位或者 9 位，停止位可配置为 1 位或者两位。

5.2 串口应用实例 ----- 实现 printf 和 scanf

5.2.1 实例描述

首先初始化串口，波特率为 115200 波特，数据长度为 8，一位停止位，无奇偶校验位。然后在串口终端输出 ”input your string, max: 80” 提示你输入字符串，在你输入的时候，串口没有回显，该字符串以回车结束，当按下回车是，会把你刚才输入的字符串打印出来。该实例应用到了库函数 printf 和 scanf。

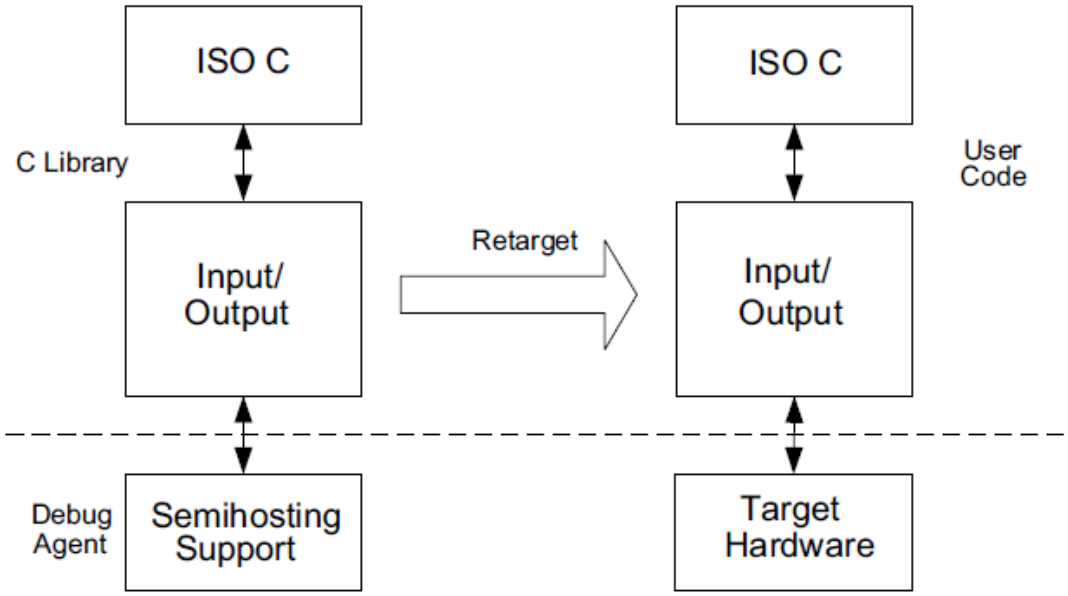
5.2.2 半主机机制 (semihosting)

在了解如何实现 printf 和 scanf 之前，先来了解半主机机制。

半主机是这么一种机制，它使得在 ARM 目标上跑的代码，如果主机电脑运行了调试器，那么该代码可以使用该主机电脑的输入输出设备。

这点非常重要，因为开发初期，可能开发者根本不知道该 ARM 器件上有什么输入输出设备，而半主基机制使得你不用知道 ARM 器件的外设，利用主机电脑的外设就可以实现输入输出调试。

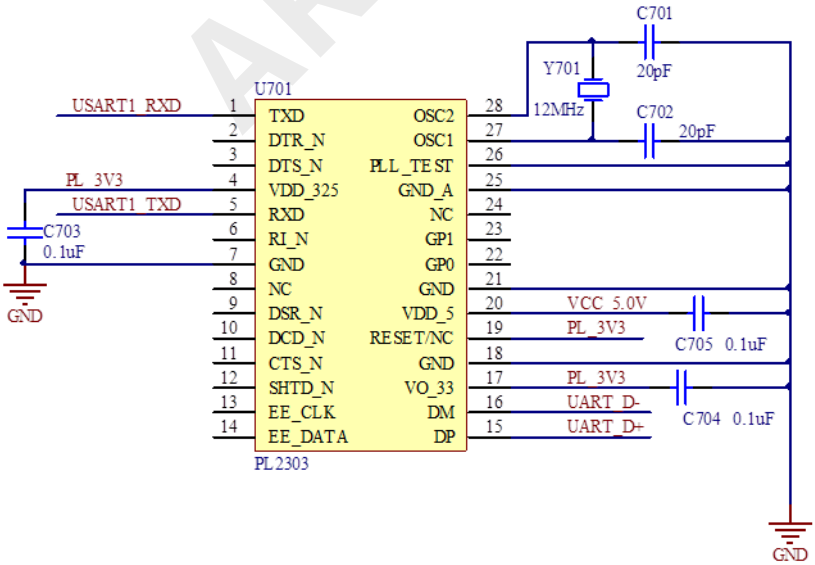
所以要利用目标 ARM 器件的输入输出设备，首先要关掉半主机机制。然后再将输入输出重定向到 ARM 器件上，如 printf 和 scanf，你需要重写 fputc 和 fgetc 函数。



5.2.3 硬件设计

硬件集成了 USB 转 串口，数据传输和接收用到了两根数据线 TX 和 RX，分别对应 PA9 和 PA10。

USART1_TXD	PA9
USART1_RXD	PA10



5.2.4 软件设计

首先调用 ARC_COM_Init(), 来初始化串口，串口使用到 PA9 和 PA10 的复用功能，首先使能端口 A 和复用功能的时钟，然后初始化该口为复用推挽输出，最大速率为 50MHz。再配置串口，波特率为 115200 波特，8 比特数据长度，1 比特停止位，无奇偶检验位，无流量控制。在主循环内，提示并且等待用户在终端内输入最大长度为 80 的字符串，以回车结束。一旦字符串输入结束，在终端输出用户刚刚输入的字符串。字符串输入输出使用了库函数 scanf 和 printf。在文件 ARC_Retarget.c 内将标准的 C 函数输入输出重定位为串口输入输出，具体实现方法参考 fputc() 和 fgetc()函数。

该实例的主要代码如下：

文件 USART_main.c

```
/**
  * @brief   Main program, retargeting the standard C printf and scanf
                to   UART output.
  * @param   None
  * @retval None
  */
int main(void)
{
    char str[80];
    ARC_COM_Init();
    /* Enable USART */
    USART_Cmd(USART1, ENABLE);

    while (1)
    {
        printf("\ninput your string, max: 80\n");
        scanf("%s",str);
        printf("your input string is\n%s\n", str);
    }
}
```

文件 ARC_USART.c

```
/**
  * @brief   Configures COM port parameters.
  * @param   None
  * @retval None
  */
void ARC_COM_PARAM_Init()
{
    /* USART1 configured as follow:
```

```
            - BaudRate = 115200 baud
            - Word Length = 8 Bits
            - One Stop Bit
            - No parity
            - Hardware flow control disabled (RTS and CTS signals)
            - Receive and transmit enabled
        */
        USART_InitTypeDef USART_InitStructure;

        USART_InitStructure.USART_BaudRate = 115200;
        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 configuration */
        USART_Init(USART1, &USART_InitStructure);
}

/**
   * @brief    Initialize COM port.
   * @param    None
   * @retval None
   */
void ARC_COM_Init()
{
        ARC_COM_RCC_Init();
        ARC_COM_GPIO_Init();
        ARC_COM_PARAM_Init();
}
```

文件 ARC_RCC.c

```
/**
   * @brief    Configures COM clocks.
   * @param    None
   * @retval None
   */
void ARC_COM_RCC_Init(void)
{
        /* Enable GPIO clock */
```

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA |
RCC_APB2Periph_AFIO, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
}
```

文件 ARC_GPIO.c

```
/**
  * @brief  Configures UART GPIO ports.
  * @param  None
  * @retval None
  */

/*
        -------------
        | TX | PA9   |
        -------------
        | RX | PA10  |
        -------------
*/
void ARC_COM_GPIO_Init()
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Configure USART Tx as alternate function push-pull */
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure USART Rx as input floating */
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

文件 ARC_Retarget.c

```
/**
  * @brief  Retargets the C library printf function to the USART.
  * @param  ch: the char to be send.
  * @param  *f:
  * @retval the char that send out.
  */
int fputc(int ch, FILE *f)
{
```

```
    /* Place your implementation of fputc here */
    /* e.g. write a character to the USART */

    /* Loop until the end of transmission */
    while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET)
    {
    }
    USART_SendData(USART1, (uint8_t) ch);

    return ch;
}

/**
  * @brief  Retargets the C library printf function to the USART.
  * @param  *f
  * @retval the char that received.
  */
int fgetc(FILE *f)
{
    while (USART_GetFlagStatus(USART1, USART_FLAG_RXNE) ==
RESET)
    {
    }
    return(USART_ReceiveData(USART1));
}
```