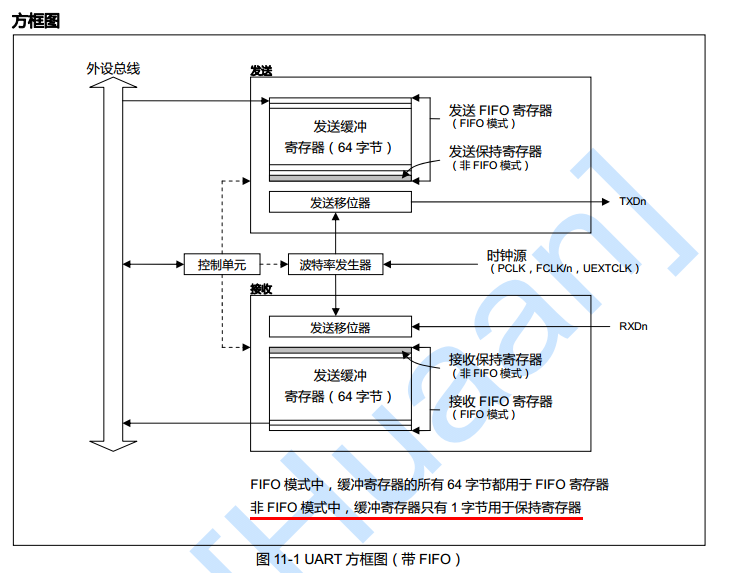
# mini2440裸机试炼之——Uart与pc端实现文件、字符传输

1. 波特率（Baud rate）即调制速率，1波特即指每秒传输1个符号。
2. 非FIFO模式，即传输数据不利用FIFO缓存，一个字节一个字节地传输。
3. 接收到的数据是放到接收缓存器URXHn中，要发送数据时，是把数据放入发送缓存器UTXHn中。由于UART是通过字节方式传输数据的，因此要区分是大端模式还是小端模式，也就是说这两个寄存器在这两种模式下，所在的地址是不同。为了了解当前数据传输的各种状态，还需要一些状态寄存器。传输状态寄存器UTRSTATn非常有用，它的第0位可以用来判断接受缓存器内是否有可接收的数据，第1位和第2位可以用来判断发送缓存器中是否为空，为空时可以发送数据。由于在这里我们不进行传输数据时错误的判断，因此错误状态寄存器UERSTATn不需要，FIFO状态寄存器UFSTATn和MODEM状态寄存器UMSTATn在这里也不需要。

本来是想使用中断方式接收和发送的，但是只实现发送接收一个字符。

之后试下查询方式，结果能行。

## Uart方框图

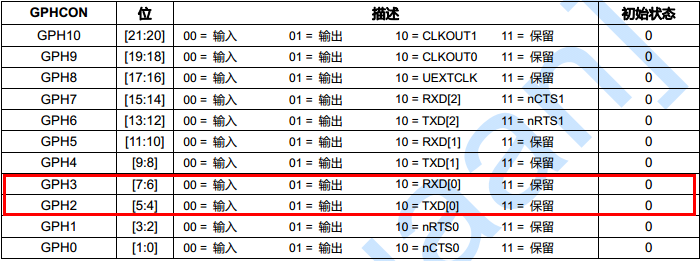


1、使用非FIFO模式时，接收、发送缓存寄存器只存一个字节

2、时钟源使用PCLK（50000000），波特率为115200；

UBRDIVn = (int)( UART时钟 / ( 波特率 ×16) ) – 1

## 配置GPH



rGPHCON|=((2<<4)|(2<<6)); //设置GPH2、GPH3为TXD0、RXD0功能

rGPHUP=0x00; //上拉电阻使能

## 设置Uart寄存器

rUFCON0 = 0x0; //禁止FIFO

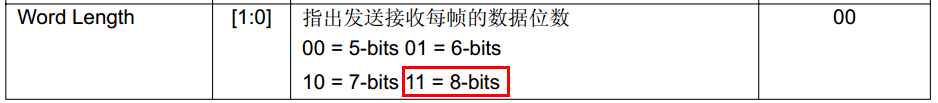
rUMCON0 = 0x0; //禁止AFC

**UCONn控制寄存器：**



rUCON0=0x05; //发送模式和接收模式都使用查询模式

**ULCONn控制寄存器：**

****

rULCON0|=0x03; //设置UART0数据发送8个数据位

**赵老师的一段话（关于中断的，作为笔记）：**

|  |
| --- |
| 最后还要强调几点关于非FIFO模式下UART中断的一些注意事项：  1.对于s3c2440来说，接收数据是被动的，发送数据是主动的，因此一般来说，接收数据用中断方式，发送数据用查询方式较好；  2. 在中断方式下，当接收到数据时，尽管可能该数据无用，但也一定要读取它，否则下次再接收数据时，不会再引起中断，因为接收数据缓存器被上次接收到的数据所霸占，只要没有读取它，它就永远在那里；  3. 由于UART中断涉及到SUBSRCPND寄存器，因此在中断处理程序中不仅要清SRCPND寄存器，还要清SUBSRCPND寄存器，它们的顺序一定是先清SUBSRCPND寄存器，再清SRCPND寄存器，否则就会引起一个中断两次响应的问题。因为是否中断由SRCPND寄存器决定，而SRCPND寄存器的相关状态位由SUBSRCPND寄存器决定，如果先清SRCPND寄存器，而还没有清SUBSRCPND寄存器的话，SRCPND寄存器的相关位还是会被置1，而一旦被置1，则一定还会引起中断。 |

## 总结：

使用查询方法确实可以通过PC端的串口工具发送文件或者字符串，再通过uart0接收文件或字符串发送给PC端的串口工具和存储于文件uart.txt中，但是过程中会出现字符丢失（发送回PC端串口工具和文件中都是一样的情况）的情况，现在感觉可能是串口工具同步的问题或者机器之间的延迟问题，时间的延迟并不会影响串口之间数据传输

## 代码区：

Main.c

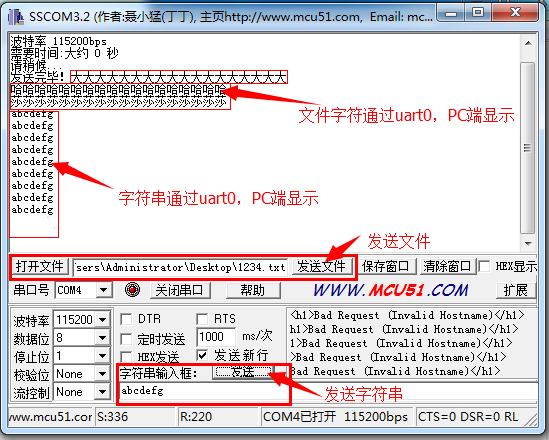
|  |
| --- |
| #define GLOBAL\_CLK 1  #include "def.h"  #include "option.h"  #include "2440addr.h"  #include "2440lib.h" //函数声明  #include "2440slib.h"  #include "mmu.h"  #include "profile.h"  extern void Uart**(**void**);**  void Main**(**void**)**  **{** U32 mpll\_val **=** 0**,**consoleNum**;**  Port\_Init**();**    mpll\_val **=** **(**92**<<**12**)|(**1**<<**4**)|(**1**);**    //init FCLK=400M,  ChangeMPllValue**((**mpll\_val**>>**12**)&**0xff**,** **(**mpll\_val**>>**4**)&**0x3f**,** mpll\_val**&**3**);**  ChangeClockDivider**(**14**,** 12**);** //the result of rCLKDIVN [0:1:0:1] 3-0 bit  cal\_cpu\_bus\_clk**();** //HCLK=100M PCLK=50M      MMU\_Init**();** //地址映射初始化    Beep**(**2000**,** 100**);**    Uart**();** //实现串口传送文件 字符串到开发板，并显示在串口终端(调试成功！)  **}** |

**Uart.c**

|  |
| --- |
| #include "2440addr.h"  #include "def.h"  #include "2440lib.h"  #include <stdio.h> //文件库函数  void Uart\_init**(){**  //以下使用uart0  rGPHCON**&=~((**3**<<**4**)|(**3**<<**6**));** //GPH2--TXD0;GPH3--RXD0  rGPHCON**|=((**2**<<**4**)|(**2**<<**6**));** //设置GPH2、GPH3为TXD0、RXD0功能  //手册中GPH2、GPH3描述如下:  //GPHCON Bit Description  //GPH3 [7:6] 00 = Input 01 = Output  // 10 = RXD0 11 = reserved  //GPH2 [5:4] 00 = Input 01 = Output  // 10 = TXD0 11 = Reserved    rGPHUP**=**0x00**;** //上拉电阻使能  rUFCON0 **=** 0x0**;** //禁止FIFO  rUMCON0 **=** 0x0**;** //撤销nRTS  rULCON0**|=**0x03**;** //设置UART0数据发送8个数据位  rUCON0**=**0x05**;** //发送模式和接收模式都使用查询模式    //设置波特率，其中波特率作为一个参数传递到该初始化函数  rUBRDIV0**=(**int**)((**50000000**/(**115200**\***16**))-**1**);**  Delay**(** 10 **)** **;**    **}**    void Uart**(){**  int i**=**0**;**  char ch**;**  FILE **\***FP**;** //文件指针  FP**=**fopen**(**"uart.txt"**,**"w"**);** //以写方式打开uart.txt文件  Uart\_init**();** //uart初始化函数  **while(**1**){**  **if(**rUTRSTAT0**&**1**)** **{** //如果接收数据缓存器接收到有效数据    ch **=** rURXH0**;** //ch存储接收字节数据  rUTXH0**=**ch**;** //PC机将接收的字符通过串口调试工具显示在屏幕上    fputc**(**ch**,**FP**);** //将ch字符内容存进文件中  i**++;** **if(**i**>**200**)** fclose**(**FP**);** //如果有200个字节关闭文件  **}**  **}**    **}** |

## 结果截图：

1、PC端串口工具



2、Uart.txt是存储于项目2440test\_Data\DebugRel\_bin目录下，与bin文件同一目录

