专题14-串口来做控制台

一、串口工作常识

1.1、UART原理及UART部件使用方法

1.1.1、概念

通用异步收发传输器（Universal Asynchronous Receiver/Transmitter)，通常称作UART，是一种异步收发传输器，是电脑硬件的一部分。它将要传输的资料在串行通信与并行通信之间加以转换。TxD用于发送数据，RxD用于接收数据，GND用于给双方提供参考电平。

通信协议编辑

UART作为异步串口通信协议的一种，工作原理是将传输数据的每个字符一位接一位地传输。
其中各位的意义如下：
起始位：先发出一个逻辑”0”的信号，表示传输字符的开始。
数据位：紧接着起始位之后。资料位的个数可以是4、5、6、7、8等，构成一个字符。通常采用ASCII码。从最低位开始传送，靠时钟定位。
奇偶校验位：资料位加上这一位后，使得“1”的位数应为偶数(偶校验)或奇数(奇校验)，以此来校验资料传送的正确性。

停止位：它是一个字符数据的结束标志。可以是1位、1.5位、2位的高电平。 由于数据是在传输线上定时的，并且每一个设备有其自己的时钟，很可能在通信中两台设备间出现了小小的不同步。因此停止位不仅仅是表示传输的结束，并且提供计算机校正时钟同步的机会。适用于停止位的位数越多，不同时钟同步的容忍程度越大，但是数据传输率同时也越慢。

空闲位：处于逻辑“1”状态，表示当前线路上没有资料传送。

波特率：是衡量资料传送速率的指标。表示每秒钟传送的符号数（symbol）。一个符号代表的信息量（比特数）与符号的阶数有关。例如资料传送速率为120字符/秒，传输使用256阶符号，每个符号代表8bit，则波特率就是120baud，比特率是120*8=960bit/s。这两者的概念很容易搞错。

基本结构
(1) 输出缓冲寄存器，它接收CPU从数据总线上送来的并行数据，并加以保存。
(2) 输出移位寄存器，它接收从输出缓冲器送来的并行数据，以发送时钟的速率把数据逐位移出，即将并行数据转换为串行数据输出。
(3) 输入移位寄存器，它以接收时钟的速率把出现在串行数据输入线上的数据逐位移入，当数据装满后，并行送往输入缓冲寄存器，即将串行数据转换成并行数据。
(4) 输入缓冲寄存器，它从输入移位寄存器中接收并行数据，然后由CPU取走。
(5) 控制寄存器，它接收CPU送来的控制字，由控制字的内容，决定通信时的传输方式以及数据格式等。例如采用异步方式还是同步方式，数据字符的位数，有无奇偶校验，是奇校验还是偶校验，停止位的位数等参数。
(6) 状态寄存器。状态寄存器中存放着接口的各种状态信息，例如输出缓冲区是否空，输入字符是否准备好等。在通信过程中，当符合某种状态时，接口中的状态检测逻辑将状态寄存器的相应位置“1” ，以便让CPU查询。

1.1.2、串口硬件引脚

我们通常使用的RS232的9帧串 口，其中最为重要的是2,3,5脚
2 ：RXD:接收数据
3 ：TXD:发送数据
5 ：GND:接地

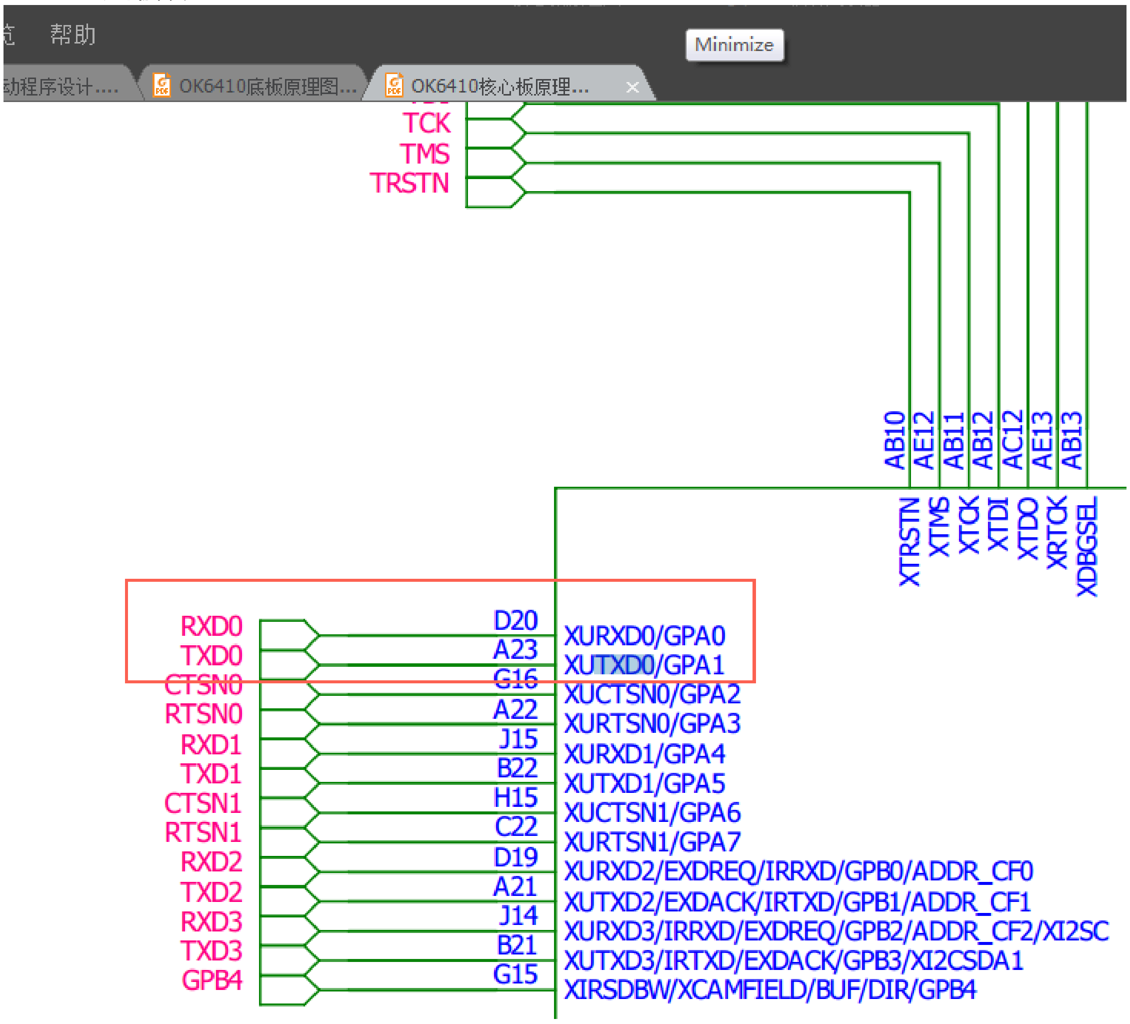
二、OK6410串口设置

2.1、串口初始化

```
#define GPACON         (*((volatile unsigned long *)0x7f008000))
#define ULCON0         (*((volatile unsigned long *)0x7f005000))
#define UCON0          (*((volatile unsigned long *)0x7f005004))

#define UTRSTAT0       (*((volatile unsigned long *)0x7f005010))
#define UTXH0          (*((volatile unsigned char *)0x7f005020))
#define URXH0          (*((volatile unsigned char *)0x7f005024))

#define UBRDIV0        (*((volatile unsigned short *)0x7f005028))
#define UDIVSLOT0      (*((volatile unsigned short *)0x7f00502c))
```

```

#define PCLK    66500000
#define BAUD    115200

void uart_init(void)
{
        //set GPIO to enable UART,GPA0 is RXD,GPA1 is TXD
        GPACON &= ~(0xff);
        GPACON |= ((0b10 << 0) | (0b10 << 4));

        //set frame format:8 bits data,one stop bit,No parity
        ULCON0 = ((0b11 << 0) | (0b0 << 2) | (0b000 << 3) | (0b0 << 6));

        //set UART model:polling mode
        UCON0 |= ((0b01 << 0) | (0b01 << 2));

        //set UART baud rate, bps=115200
        UBRDIV0 = (int)(PCLK / (BAUD * 16) -1);
        UDIVSLOT0 = 0x0;

        //set UART CLK : PCLK
        UCON0 |= (0x00 << 10);
}

```

2.1.1、引脚设置



10.4 REGISTER DESCRIPTION

10.4.1 MEMORY MAP

Register	Address	R/W	Description	Reset Value
GPACON	0x7F008000	R/W	Port A Configuration Register	0x0
GPADAT	0x7F008004	R/W	Port A Data Register	Undefined
GPAPUD	0x7F008008	R/W	Port A Pull-up/down Register	0x00005555
GPACONSLP	0x7F00800C	R/W	Port A Sleep mode Configuration Register	0x0
GPAPUDSLP	0x7F008010	R/W	Port A Sleep mode Pull-up/down Register	0x0

10.5.1 PORT A CONTROL REGISTER

There are five control registers including GPACON, GPADAT, GPAPUD, GPACONSLP and GPAPUDSLP in the Port A Control Registers.

Register	Address	R/W	Description	Reset Value
GPACON	0x7F008000	R/W	Port A Configuration Register	0x0000
GPADAT	0x7F008004	R/W	Port A Data Register	Undefined
GPAPUD	0x7F008008	R/W	Port A Pull-up Register	0x0005555
GPACONSLP	0x7F00800C	R/W	Port A Sleep mode Configuration Register	0x0
GPAPUDSLP	0x7F008010	R/W	Port A Sleep mode Pull-up/down Register	0x0

GPACON	Bit	Description	Initial State
GPA0	[3:0]	0000 = Input 0001 = Output 0010 = UART RXD[0] 0011 = Reserved 0100 = Reserved 0101 = Reserved 0110 = Reserved 0111 = External Interrupt Group 1 [0]	0000
GPA1	[7:4]	0000 = Input 0001 = Output 0010 = UART TXD[0] 0011 = Reserved 0100 = Reserved 0101 = Reserved 0110 = Reserved 0111 = External Interrupt Group 1 [1]	0000

2.1.2、帧格式设置

31.5.1 MEMORY MAP

Register	Address	R/W	Description	Reset Value
ULCON0	0x7F005000	R/W	UART channel 0 line control register	0x00
UCON0	0x7F005004	R/W	UART channel 0 control register	0x00
UFCON0	0x7F005008	R/W	UART channel 0 FIFO control register	0x0
UMCON0	0x7F00500C	R/W	UART channel 0 Modem control register	0x0
UTRSTAT0	0x7F005010	R	UART channel 0 Tx/Rx status register	0x6
UERSTAT0	0x7F005014	R	UART channel 0 Rx error status register	0x0
UFSTAT0	0x7F005018	R	UART channel 0 FIFO status register	0x00
UMSTAT0	0x7F00501C	R	UART channel 0 Modem status register	0x0
UTXH0	0x7F005020	W	UART channel 0 transmit buffer register	-
URXH0	0x7F005024	R	UART channel 0 receive buffer register	0x00
UBRDIV0	0x7F005028	R/W	UART channel 0 Baud rate divisior register	0x0000
UDIVSLOT0	0x7F00502C	R/W	UART channel 0 Dividing slot register	0x0000
UINTP0	0x7F005030	R/W	UART channel 0 Interrupt Pending Register	0x0
UINTSP0	0x7F005034	R/W	UART channel 0 Interrupt Source Pending Register	0x0
UINTM0	0x7F005038	R/W	UART channel 0 Interrupt Mask Register	0x0

31.6.1 UART LINE CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
ULCON0	0x7F005000	R/W	UART channel 0 line control register	0x00
ULCON1	0x7F005400	R/W	UART channel 1 line control register	0x00
ULCON2	0x7F005800	R/W	UART channel 2 line control register	0x00
ULCON3	0x7F005C00	R/W	UART channel 3 line control register	0x00

There are three UART line control registers including ULCON0, ULCON1, ULCON2 and ULCON3 in the UART block.

ULCONn	Bit	Description	Initial State
Reserved	[7]		0
Infra-Red Mode	[6]	Determine whether or not to use the Infra-Red mode. 0 = Normal mode operation 1 = Infra-Red Tx/Rx mode	0
Parity Mode	[5:3]	Specify the type of parity generation and checking during UART transmit and receive operation. 0xx = No parity 100 = Odd parity 101 = Even parity 110 = Parity forced/checked as 1 111 = Parity forced/checked as 0	000
Number of Stop Bit	[2]	Specify how many stop bits are to be used for end-of-frame signal. 0 = One stop bit per frame 1 = Two stop bit per frame	0
Word Length	[1:0]	Indicate the number of data bits to be transmitted or received per frame. 00 = 5-bit 01 = 6-bit 10 = 7-bit 11 = 8-bit	00

2.1.3、UART模式设置

31.6.2 UART CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
UCON0	0x7F005004	R/W	UART channel 0 control register	0x00
UCON1	0x7F005404	R/W	UART channel 1 control register	0x00
UCON2	0x7F005804	R/W	UART channel 2 control register	0x00
UCON3	0x7F005C04	R/W	UART channel 3 control register	0x00

There are three UART control registers including UCON0, UCON1, UCON2 and UCON3 in the UART block.

UCONn	Bit	Description	Initial State
Clock Selection	[11:10]	Select PCLK or EXT_UCLK0[4] or EXT_UCLK1[4] clock for the UART baud rate. x0 = PCLK : DIV_VAL[1] = (PCLK / (bps x 16)) -1 01 = EXT_UCLK0: DIV_VAL[1] = (EXT_UCLK0 / (bps x 16)) –1 11 = EXT_UCLK1: DIV_VAL[1] = (EXT_UCLK1 / (bps x 16)) –1	0
Tx Interrupt Type	[9]	Interrupt request type. 0 = Pulse (Interrupt is requested as soon as the Tx buffer becomes empty in Non-FIFO mode or reaches Tx FIFO Trigger Level in FIFO mode.) 1 = Level (Interrupt is requested while Tx buffer is empty in Non-FIFO mode or reaches Tx FIFO Trigger Level in FIFO mode.)	0
Rx Interrupt Type	[8]	Interrupt request type. [2] 0 = Pulse (Interrupt is requested the instant Rx buffer receives the data in Non-FIFO mode or reaches Rx FIFO Trigger Level in FIFO mode.) 1 = Level (Interrupt is requested while Rx buffer is receiving data in Non-FIFO mode or reaches Rx FIFO Trigger Level in FIFO mode.)	0
Rx Time Out	[7]	Enable/Disable Rx time-out interrupts when UART FIFO is enabled.	0

Enable		The interrupt is a receive interrupt. [3)] 0 = Disable 1 = Enable	
Rx Error Status Interrupt Enable	[6]	Enable the UART to generate an interrupt upon an exception, such as a break, frame error, parity error, or overrun error during a receive operation. 0 = Do not generate receive error status interrupt. 1 = Generate receive error status interrupt.	0
Loop-back Mode	[5]	Setting loop-back bit to 1 cause the UART to enter the loop-back mode. This mode is provided for test purposes only. 0 = Normal operation 1 = Loop-back mode	0
Send Break Signal	[4]	Setting this bit causes the UART to send a break during 1 frame time. This bit is automatically cleared after sending the break signal. 0 = Normal transmit 1 = Send break signal	0

UCONn	Bit	Description	Initial State
Transmit Mode	[3:2]	Determine which function is currently able to write Tx data to the UART transmit buffer register. 00 = Disable 01 = Interrupt request or polling mode 10 = DMA request (DMA_UART0) 11 = DMA request (DMA_UART1)	00
Receive Mode	[1:0]	Determine which function is currently able to read data from UART receive buffer register. 00 = Disable 01 = Interrupt request or polling mode 10 = DMA request (DMA_UART0) 11 = DMA request (DMA_UART1)	00

NOTES:

1) DIV_VAL = UBRDIVn + (num of 1's in UDIVSLOTn)/16. Refer to UART Baud Rate Configure Registers.
2) RX interrupt type must be set to pulse for every transfer in S3C6410.
3) When the UART does not reach the FIFO trigger level and does not receive data during 3 word time in DMA receive mode with FIFO, the Rx interrupt will be generated (receive time out), and then you must check the FIFO status and read out the rest.
4) EXT_UCLK0 clock is external clock.(XpwmECLK PAD input)
   EXT_UCLK1 clock is generated clock by SYSCON. SYSCON generates EXT_UCLK1 for dividing EPLL or MPLL output.
   When you want to change EXT_UCLK0 to PCLK for UART baudrate , clock selection field must be set to 2'b00.
   But, when you want to change EXT_UCLK1 to PCLK for UART baudrate , clock selection field must be set to 2'b10.

2.1.4、波特率设置

31.6.11 UART BAUD RATE CONFIGURE REGISTER

There are four UART baud rate divisor registers including UBRDIV0, UBRDIV1, UBRDIV2 and UBRDIV3 in the UART block.

The value stored in the baud rate divisor register (UBRDIVn) and dividing slot register(UDIVSLOTn), are used to determine the serial Tx/Rx clock rate (baud rate) as follows:

DIV_VAL = UBRDIVn + (num of 1's in UDIVSLOTn)/16
DIV_VAL = (PCLK / (bps x 16)) –1
DIV_VAL = (EXT_UCLK0 / (bps x 16)) –1
or
DIV_VAL = (EXT_UCLK1 / (bps x 16)) –1

Where, the divisor must be from 1 to (2^{16}-1)
Using UDIVSLOT, you can make more accurate baud rate.
For example, if the baud-rate is 115200 bps and EXT_UCLK0 is UART baud-rate clock and 40 MHz, UBRDIVn and UDIVSLOTn are:

DIV_VAL = (40000000 / (115200 x 16)) -1
= 21.7 -1
= 20.7

UBRDIVn = 20 (integer part of DIV_VAL)
(num of 1's in UDIVSLOTn)/16 = 0.7
then, (num of 1's in UDIVSLOTn) = 11
so, UDIVSLOTn can be 16'b1110_1110_1110_1010 or 16'b0111_0111_0111_0101, etc.

We recommend selecting UDIVSLOTn as described in the following table:

Num of 1's	UDIVSLOTn	Num of 1's	UDIVSLOTn
0	0x0000(0000_0000_0000_0000b)	8	0x5555(0101_0101_0101_0101b)
1	0x0080(0000_0000_0000_1000b)	9	0xD555(1101_0101_0101_0101b)
2	0x0808(0000_1000_0000_1000b)	10	0xD5D5(1101_0101_1101_0101b)
3	0x0888(0000_1000_1000_1000b)	11	0xDDD5(1101_1101_1101_0101b)
4	0x2222(0010_0010_0010_0010b)	12	0xDDDD(1101_1101_1101_1101b)
5	0x4924(0100_1001_0010_0100b)	13	0xDFDD(1101_1111_1101_1101b)
6	0x4A52(0100_1010_0101_0010b)	14	0xDFDF(1101_1111_1101_1111b)
7	0x54AA(0101_0100_1010_1010b)	15	0xFFDF(1111_1111_1101_1111b)

31.6.12 BAUD-RATE ERROR TOLERANCE

UART Frame error should be less than 1.87%(3/160)
tUPCLK = (UBRDIVn + 1) x 16 x 1Frame / (PCLK ,EXT_UCLK0 or EXT_UCLK1) tUPCLK: Real UART Clock
tEXTUARTCLK = 1Frame / baud-rate tEXTUARTCLK: Ideal UART Clock
UART error = (tUPCLK – tEXTUARTCLK) / tEXTUARTCLK x 100%

NOTE: 1Frame = start bit + data bit + parity bit + stop bit.

31.6.13 UART CLOCK AND PCLK RELATION

There is a constraint on the ratio of clock frequencies for PCLK to UARTCLK.

The frequency of UARTCLK must be no more than 5.5/3 times faster than the frequency of PCLK :

$F_{UARTCLK} <= 5.5/3 \times F_{PCLK}$ **$F_{UARTCLK}$ = baudrate x 16**

This allows sufficient time to write the received data to the receive FIFO

Register	Address	R/W	Description	Reset Value
UBRDIV0	0x7F005028	R/W	Baud rate divisior register 0	0x0000
UBRDIV1	0x7F005428	R/W	Baud rate divisior register 1	0x0000
UBRDIV2	0x7F005828	R/W	Baud rate divisior register 2	0x0000
UBRDIV3	0x7F005C28	R/W	Baud rate divisior register 3	0x0000

UBRDIV n	Bit	Description	Initial State
UBRDIV	[15:0]	Baud rate division value (When UART clock source is PCLK, UBRDIVn must be more than 0 (UBRDIVn >0))	-

NOTE: If UBRDIV value is 0, UART baudrate is not affected by UDIVSLOT value.

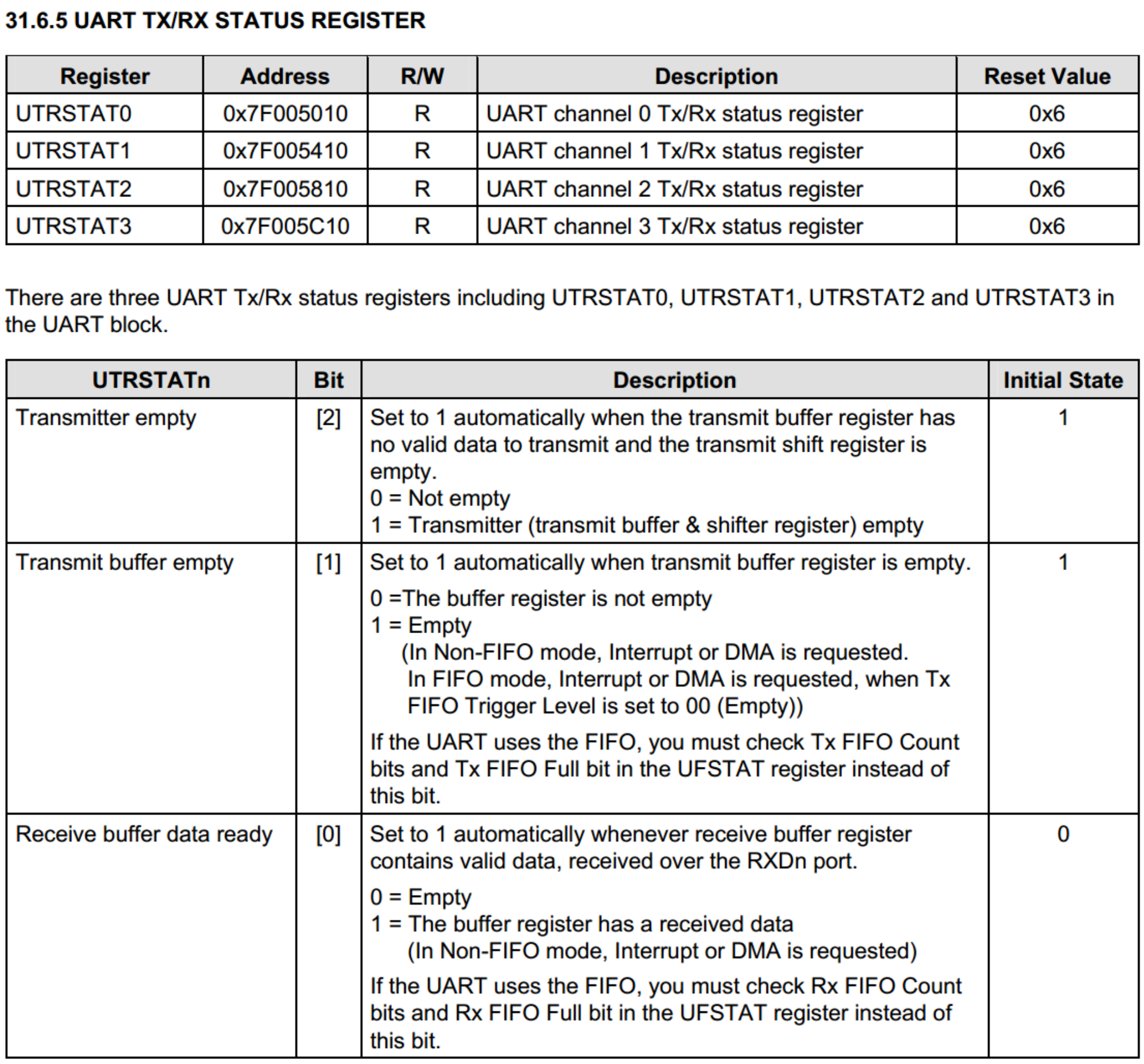
Register	Address	R/W	Description	Reset Value
UDIVSLOT0	0x7F00502C	R/W	Baud rate divisior register 0	0x0000
UDIVSLOT1	0x7F00542C	R/W	Baud rate divisior register 1	0x0000
UDIVSLOT2	0x7F00582C	R/W	Baud rate divisior register 2	0x0000
UDIVSLOT3	0x7F005C2C	R/W	Baud rate divisior register 3	0x0000

UDIVSLOT n	Bit	Description	Initial State
UDIVSLOT	[15:0]	Select the slot where clock generator divide clock source	-

2.1.5、UART时钟源选择

2.2、实现串口收发数据

2.2.1、发送一个字符

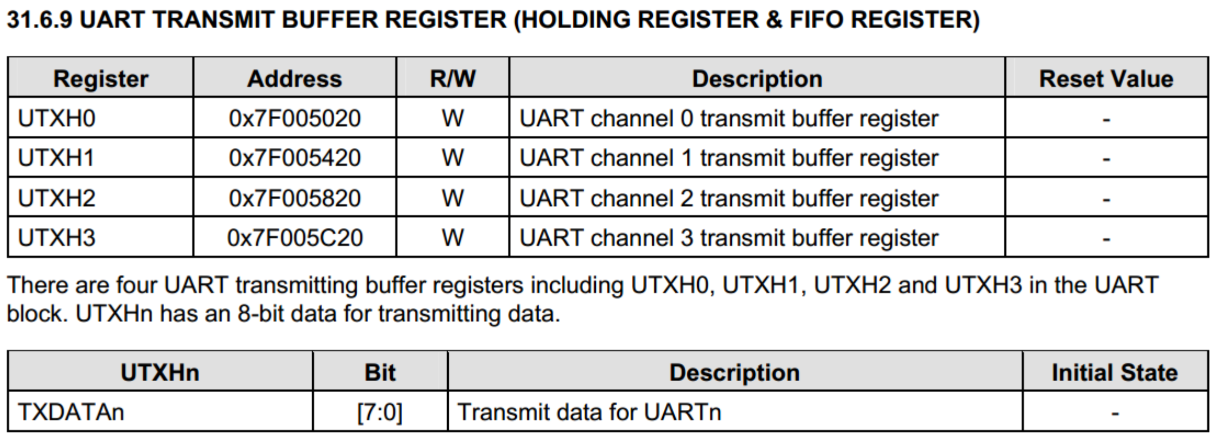


31.6.5 UART TX/RX STATUS REGISTER

Register	Address	R/W	Description	Reset Value
UTRSTAT0	0x7F005010	R	UART channel 0 Tx/Rx status register	0x6
UTRSTAT1	0x7F005410	R	UART channel 1 Tx/Rx status register	0x6
UTRSTAT2	0x7F005810	R	UART channel 2 Tx/Rx status register	0x6
UTRSTAT3	0x7F005C10	R	UART channel 3 Tx/Rx status register	0x6

There are three UART Tx/Rx status registers including UTRSTAT0, UTRSTAT1, UTRSTAT2 and UTRSTAT3 in the UART block.

UTRSTATn	Bit	Description	Initial State
Transmitter empty	[2]	Set to 1 automatically when the transmit buffer register has no valid data to transmit and the transmit shift register is empty. 0 = Not empty 1 = Transmitter (transmit buffer & shifter register) empty	1
Transmit buffer empty	[1]	Set to 1 automatically when transmit buffer register is empty. 0 =The buffer register is not empty 1 = Empty (In Non-FIFO mode, Interrupt or DMA is requested. In FIFO mode, Interrupt or DMA is requested, when Tx FIFO Trigger Level is set to 00 (Empty)) If the UART uses the FIFO, you must check Tx FIFO Count bits and Tx FIFO Full bit in the UFSTAT register instead of this bit.	1
Receive buffer data ready	[0]	Set to 1 automatically whenever receive buffer register contains valid data, received over the RXDn port. 0 = Empty 1 = The buffer register has a received data (In Non-FIFO mode, Interrupt or DMA is requested) If the UART uses the FIFO, you must check Rx FIFO Count bits and Rx FIFO Full bit in the UFSTAT register instead of this bit.	0



31.6.9 UART TRANSMIT BUFFER REGISTER (HOLDING REGISTER & FIFO REGISTER)

Register	Address	R/W	Description	Reset Value
UTXH0	0x7F005020	W	UART channel 0 transmit buffer register	-
UTXH1	0x7F005420	W	UART channel 1 transmit buffer register	-
UTXH2	0x7F005820	W	UART channel 2 transmit buffer register	-
UTXH3	0x7F005C20	W	UART channel 3 transmit buffer register	-

There are four UART transmitting buffer registers including UTXH0, UTXH1, UTXH2 and UTXH3 in the UART block. UTXHn has an 8-bit data for transmitting data.

UTXHn	Bit	Description	Initial State
TXDATAn	[7:0]	Transmit data for UARTn	-

```
void uart_put_char(unsigned char c)
{
	while (!(UTRSTAT0 & (1 << 2))) ;
	UTXH0 = c;
}

```

2.2.2、接收一个字符

31.6.10 UART RECIVE BUFFER REGISTER (HOLDING REGISTER & FIFO REGISTER)

There are four UART receiving buffer registers including URXH0, URXH1, URXH2 and URXH3 in the UART block. URXHn has an 8-bit data for received data.

Register	Address	R/W	Description	Reset Value
URXH0	0x7F005024	R	UART channel 0 receive buffer register	0x00
URXH1	0x7F005424	R	UART channel 1 receive buffer register	0x00
URXH2	0x7F005824	R	UART channel 2 receive buffer register	0x00
URXH3	0x7F005C24	R	UART channel 3 receive buffer register	0x00

URXHn	Bit	Description	Initial State
RXDATAn	[7:0]	Receive data for UARTn	0x00

NOTE: When an overrun error occurs, the URXHn must be read. If not, the next received data will also make an overrun error, even though the overrun bit of UERSTATn had been cleared.

```
unsigned char uart_get_char(void)
{
        unsigned char c;

        while (!(UTRSTAT0 & (1 << 0))) ;
        c = URXH0;

        if ((c == 0x0d) || (c == 0x0a)) {
                uart_put_char(0x0d);
                uart_put_char(0x0a);
        }
        else
                uart_put_char(c);

        return c;
}

```

2.3、其他寄存器

31.6.3 UART FIFO CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
UFCON0	0x7F005008	R/W	UART channel 0 FIFO control register	0x0
UFCON1	0x7F005408	R/W	UART channel 1 FIFO control register	0x0
UFCON2	0x7F005808	R/W	UART channel 2 FIFO control register	0x0
UFCON3	0x7F005C08	R/W	UART channel 3 FIFO control register	0x0

There are three UART FIFO control registers including UFCON0, UFCON1, UFCON2 and UFCON3 in the UART block.

UFCONn	Bit	Description	Initial State
Tx FIFO Trigger Level	[7:6]	Determine the trigger level of transmit FIFO. 00 = Empty 01 = 16-byte 10 = 32-byte 11 = 48-byte	00
Rx FIFO Trigger Level	[5:4]	Determine the trigger level of receive FIFO.1) 00 = 1-byte 01 = 8-byte 10 = 16-byte 11 = 32-byte	00
Reserved	[3]		0
Tx FIFO Reset	[2]	Auto-cleared after resetting FIFO 0 = Normal 1= Tx FIFO reset	0
Rx FIFO Reset	[1]	Auto-cleared after resetting FIFO 0 = Normal 1= Rx FIFO reset	0
FIFO Enable	[0]	0 = Disable 1 = Enable	0

NOTES:

1) For using RX DMA in FIFO mode, Rx FIFO trigger level must be same value as DMA burst size. When using DMA single operation, RX FIFO trigger level must be set to 1-byte.

31.6.4 UART MODEM CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
UMCON0	0x7F00500C	R/W	UART channel 0 Modem control register	0x0
UMCON1	0x7F00540C	R/W	UART channel 1 Modem control register	0x0
Reserved	0x7F00580C	-	Reserved	Undef
Reserved	0x7F005C0C	-	Reserved	Undef

There are two UART MODEM control registers including UMCON0 and UMCON1 in the UART block.

UMCONn	Bit	Description	Initial State
RTS trigger Level	[7:5]	When AFC bit is enabled, these bits determine when to inactivate nRTS signal. 000 = When RX FIFO contains 63 bytes. 001 = When RX FIFO contains 56 bytes. 010 = When RX FIFO contains 48 bytes. 011 = When RX FIFO contains 40 bytes. 100 = When RX FIFO contains 32 bytes. 101 = When RX FIFO contains 24 bytes. 110 = When RX FIFO contains 16 bytes. 111 = When RX FIFO contains 8 bytes.	000
Auto Flow Control (AFC)	[4]	0 = Disable 1 = Enable	0
Modem Interrupt enable	[3]	Modem interrupt enable 0 = Disable 1 = Enable	000
Reserved	[2:1]	These bits must be 0's	000
Request to Send	[0]	If AFC bit is enabled, this value will be ignored. In this case the S3C6410 will control nRTS automatically. If AFC bit is disabled, nRTS must be controlled by software. 0 = 'H' level (Inactivate nRTS) 1 = 'L' level (Activate nRTS)	0

NOTE: UART 2 does not support AFC function, because the S3C6410 has no nRTS2 and nCTS2.
UART 3 does not support AFC function, because the S3C6410 has no nRTS3 and nCTS3.

31.6.6 UART ERROR STATUS REGISTER

Register	Address	R/W	Description	Reset Value
UERSTAT0	0x7F005014	R	UART channel 0 Rx error status register	0x0
UERSTAT1	0x7F005414	R	UART channel 1 Rx error status register	0x0
UERSTAT2	0x7F005814	R	UART channel 2 Rx error status register	0x0
UERSTAT3	0x7F005C14	R	UART channel 3 Rx error status register	0x0

There are three UART Rx error status registers including UERSTAT0, UERSTAT1, UERSTAT2 and UERSTAT3 in the UART block.

UERSTATn	Bit	Description	Initial State
Break Detect	[3]	Set to 1 automatically to indicate that a break signal has been received. 0 = Send Break signal is not received 1 = Send Break (signal is received Interrupt is requested.)	0
Frame Error	[2]	Set to 1 automatically whenever a frame error occurs during reception operation. 0 = No frame error during reception 1 = Frame error (Interrupt is requested.)	0
Parity Error	[1]	Set to 1 automatically whenever a parity error occurs during receive operation. 0 = No parity error during reception 1 = Parity error (Interrupt is requested.)	0
Overrun Error	[0]	Set to 1 automatically whenever an overrun error occurs during receive operation. 0 = No overrun error during reception 1 = Overrun error (Interrupt is requested.)	0

NOTE: These bits (UERSATn[3:0]) are automatically cleared to 0 when the UART error status register is read.

31.6.7 UART FIFO STATUS REGISTER

Register	Address	R/W	Description	Reset Value
UFSTAT0	0x7F005018	R	UART channel 0 FIFO status register	0x00
UFSTAT1	0x7F005418	R	UART channel 1 FIFO status register	0x00
UFSTAT2	0x7F005818	R	UART channel 2 FIFO status register	0x00
UFSTAT3	0x7F005C18	R	UART channel 3 FIFO status register	0x00

There are three UART FIFO status registers including UFSTAT0, UFSTAT1, UFSTAT2 and UFSTAT3 in the UART block.

UFSTATn	Bit	Description	Initial State
Reserved	[15]		0
Tx FIFO Full	[14]	Set to 1 automatically whenever transmit FIFO is full during transmit operation 0 = 0-byte ≤ Tx FIFO data ≤ 63-byte 1 = Full	0
Tx FIFO Count	[13:8]	Number of data in Tx FIFO	0
Reserved	[7]		0
Rx FIFO Full	[6]	Set to 1 automatically whenever receive FIFO is full during receive operation 0 = 0-byte ≤ Rx FIFO data ≤ 63-byte 1 = Full	0
Rx FIFO Count	[5:0]	Number of data in Rx FIFO	0

31.6.8 UART MODEM STATUS REGISTER

Register	Address	R/W	Description	Reset Value
UMSTAT0	0x7F00501C	R	UART channel 0 Modem status register	0x0
UMSTAT1	0x7F00541C	R	UART channel 1 Modem status register	0x0
Reserved	0x7F00581C	-	Reserved	Undef
Reserved	0x7F005C1C	-	Reserved	Undef

There are two UART modem status registers including UMSTAT0 and UMSTAT1 in the UART block.

UMSTAT0	Bit	Description	Initial State
Reserved	[7:5]	reserved	000
Delta CTS	[4]	Indicate that the nCTS input to the S3C6410 has changed state since the last time it was read by CPU. (Figure 31-8) 0 = Has not changed 1 = Has changed	0
Reserved	[3:1]	reserved	00
Clear to Send	[0]	0 = CTS signal is not activated (nCTS pin is high) 1 = CTS signal is activated (nCTS pin is low)	0

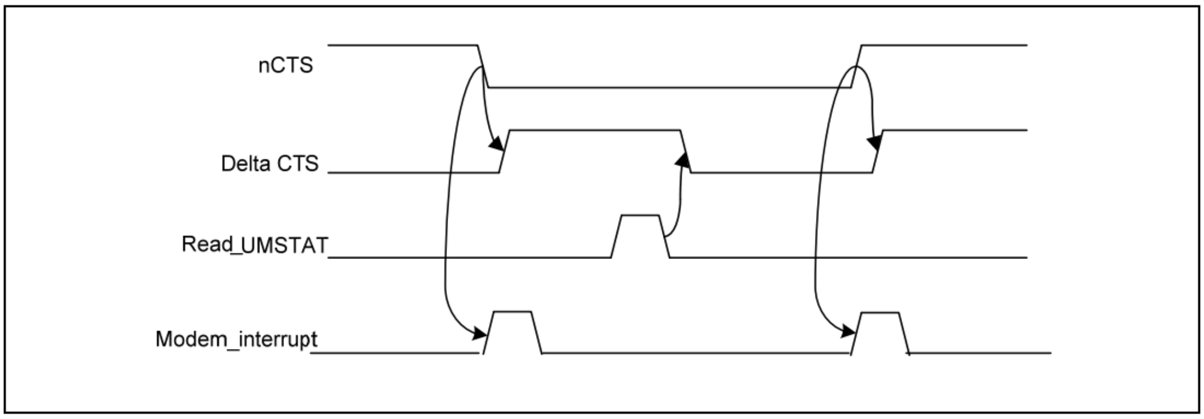


Figure 31-8. nCTS and Delta CTS Timing Diagram

31.6.14 UART INTERRUPT PENDING REGISTER

Register	Address	R/W	Description	Reset Value
UINTP0	0x7F005030	R/W	Interrupt Pending Register for UART channel 0	0x0
UINTP1	0x7F005430	R/W	Interrupt Pending Register for UART channel 1	0x0
UINTP2	0x7F005830	R/W	Interrupt Pending Register for UART channel 2	0x0
UINTP3	0x7F005C30	R/W	Interrupt Pending Register for UART channel 3	0x0

Interrupt pending register contains the information of the interrupts, which are generated.

UINTPn	Bit	Description	Initial State
MODEM	[3]	Modem interrupt generated.	0
TXD	[2]	Transmit interrupt generated.	0
ERROR	[1]	Error interrupt generated.	0
RXD	[0]	Receive interrupt generated.	0

Whenever one of above 4 bits is logical high (‘1’), each UART channel generates interrupt.
This register has to be cleared in the interrupt service routine.
You can clear specific bits of UINTP register by writing 1’s to the bits that you want to clear.

31.6.15 UART INTERRUPT SOURCE PENDING REGISTER

Interrupt Source Pending Register contains the information which interrupt are generated regardless of the value of Interrupt Mask Register

You can clear specific bits of UINTSP register by writing 1's to the bits that you want to clear.

Register	Address	R/W	Description	Reset Value
UINTSP0	0x7F005034	R/W	Interrupt Source Pending Register 0	0x0
UINTSP1	0x7F005434	R/W	Interrupt Source Pending Register 1	0x0
UINTSP2	0x7F005834	R/W	Interrupt Source Pending Register 2	0x0
UINTSP3	0x7F005C34	R/W	Interrupt Source Pending Register 3	0x0

UINTSPn	Bit	Description	Initial State
MODEM	[3]	Modem interrupt generated.	0
TXD	[2]	Transmit interrupt generated.	0
ERROR	[1]	Error interrupt generated.	0
RXD	[0]	Receive interrupt generated.	0

31.6.16 UART INTERRUPT MASK REGISTER

Interrupt mask register contains the information of the masked interrupt. If a specific bit is set to 1, interrupt request signal to Interrupt Controller is not generated even though corresponding interrupt is generated. (Note that even in such a case, the corresponding bit of UINTSPn register is set to 1). If the mask bit is 0, the interrupt request can be serviced from the corresponding interrupt source

(Note that even in such a case, the corresponding bit of UINTSPn register is set to 1).

Register	Address	R/W	Description	Reset Value
UINTM0	0x7F005038	R/W	Interrupt Mask Register for UART channel 0	0x0
UINTM1	0x7F005438	R/W	Interrupt Mask Register for UART channel 1	0x0
UINTM2	0x7F005838	R/W	Interrupt Mask Register for UART channel 2	0x0
UINTM3	0x7F005C38	R/W	Interrupt Mask Register for UART channel 3	0x0

UINTMn	Bit	Description	Initial State
MODEM	[3]	Mask Modem interrupt.	0
TXD	[2]	Mask Transmit interrupt.	0
ERROR	[1]	Mask Error interrupt.	0
RXD	[0]	Mask Receive interrupt.	0

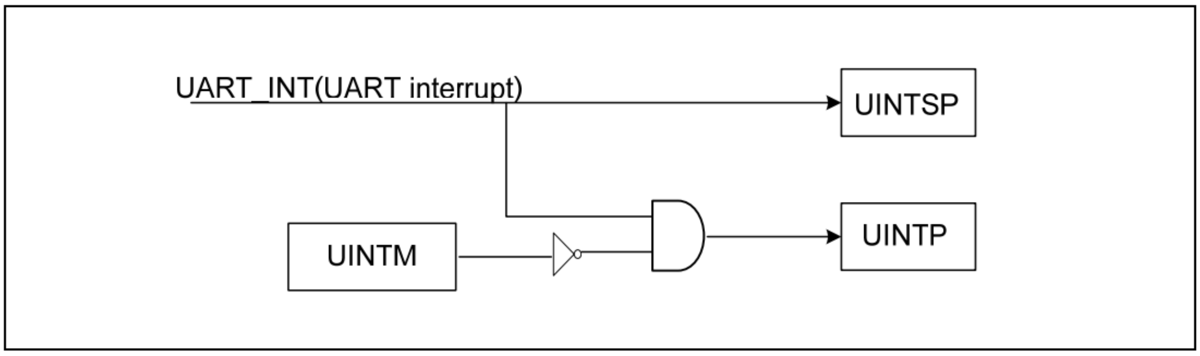


Figure 31-9. UINTSP,UINTP and UINTM block diagram

三、实现串口控制台

3.1、搭建控制台框架

3.1.1、控制台分类介绍

3.1.1.1、菜单型

3.1.1.2、解析型

3.1.2、菜单控制台搭建

```
    while(1) {
        printf("\n*****************************\n\r");
        printf("\n************U-Boot***********\n\r");
        printf("[1]:Download Linux Kerel from TFTP server!\n\r");
        printf("[2]:Boot Linux from RAM!\n\r");
        printf("[3]:Boot Linux from Nand Flash!\n\r");
        printf("\n Plese Select:");

        scanf("%d", &num);

        switch (num) {
            case 1:
                //tftp_load();
                break;

            case 2:
                //boot_linux_arm();
                break;

            case 3:
                //boot_linux_nand();
                break;

            default:
                printf("Error: Wrong selection!\n\r");
                break;
        }

    }
```

3.2、printf/scanf实现

利用现有的字符串处理函数进行转换（善假于物）。

```
#include "vsprintf.h"

unsigned char outbuff[1024];
unsigned char inbuff[1024];

int printf(const char* format, ...)
{
    va_list args;
    int i;

    //1.Converts a variable parameter to a string
    va_start(args,format);

    vsprintf((char *)outbuff,format,args);

    va_end();

    //2.print string
    for (i=0; i < strlen((char *)outbuff); i++)
    {
        uart_put_char(outbuff[i]);
    }

```

```
        return i;
}

int scanf(const char* format, ...)
{
        unsigned char c;
        int i = 0;
        va_list args;

        //1.get input char
        while (1) {
                c = uart_get_char();
                if ((c == 0x0d) || (c == 0x0a)) {
                        inbuff[i] = '\n';
                        break;
                }
                else {
                        inbuff[i++] = c;
                }
        }
        //2.format convert
        va_start(args,format);
        vsscanf((char *)inbuff,format,args);
        va_end(args);

        return i;
}

```

3.3、程序结构优化

便于管理，模块化。

优化遇到的问题：程序优化时，如果不移动nand.c到dev，就是正常的；一旦移动nand.c到dev，且在两个Makefile中做了对应的修改，虽然能正常编译生成uboot.bin,但运行却无法得到预期的结果，我对比了实例代码，并没有找到区别。

解决思路：

1、排除法：

【1】测试示代码会不会有问题？（没有问题）：说明移动文件是可行的。

【2】使用示例代码的nand.c文件替换自己编写的nand.c文件，是否依然有问题？（依然存在）：可以先排除nand.c文件的问题。

【3】为什么是移动nand.c文件就会有问题？分析nand.c文件的作用：从nand flash拷贝文件到DDR中运行，如果该部分代码不能正确执行，则大于8K地址的程序都不能正确执行。

【4】检查nand.c文件里面的程序所处的内存地址。通过反汇编发现nand.c文件的程序的内存地址在17k之后，至此确定问题点。

2、大胆猜测，小心验证：

【5】为什么会出现这样的情况？编译时只在连接器脚本指定了入口文件（start.o）,其他的.o文件在内存中的地址是如何确定的呢？由编译器指定。

【6】编译器根据什么顺序排列.o文件内存地址？暂时未知，可以通过对比猜测下：查看示例程序反汇编文件里放nand.o里程序的内存在自己编写的程序里放的是什么程序。（字符转换函数，隶属于lib/lib.o）：因此查看Makefile

```
OBJS := start.o mem.o main.o lib/lib.o dev/dev.o

CFLAGS := -fno-builtin -I$(shell pwd)/include
export CFLAGS

```

```
gboot.bin : gboot.elf
	arm-linux-objcopy -O binary gboot.elf gboot.bin

gboot.elf : $(OBJS)
	arm-linux-ld -Tgboot.lds -o gboot.elf $^
```

发现lib.lib.o在dev/dev.o前面，那调整顺序为

```
OBJS := start.o mem.o main.o dev/dev.o lib/lib.o
```

测试发现问题解决，于是可以猜测是按Makefile内的排列顺序进行地址分配的。