UART接口

11

- 串行口结构和工作原理
- **Exynos 4412**串行口特点
- 串行口专用寄存器
- 应用实例

11.1 串行口结构和工作原理

通用异步收发器(Universal Asynchronous Receiver Transmitter，UART)用来传输串行数据。发送数据时，CPU将并行数据写入发送缓冲区，UART按照规定的帧格式，通过发送端口串行发出；接收数据时，UART检测接收端口的信号，串行收集数据并暂时放在接收缓冲区中，之后，CPU即可从接收缓冲区中读取这些数据。

UART使用标准的CMOS逻辑电平(0～5 V、0～3.3 V、0～2.5 V或0～1.8 V四种)来表示数据，高电平为1，低电平为0。为了增强数据的抗干扰能力，提高传输长度，通常将CMOS逻辑电平转换为RS-232逻辑电平(3～15 V以0表示，-3～-15 V以1表示)。

TXD、RXD数据线以“位”为最小单位传输数据，其数据传输流程如下：

(1) 平时数据线处于“空闲”状态(1状态)。

(2) 当要发送数据时，UART改变TXD数据线的状态(变为0状态)并维持1位的时间，这样，接收方检测到开始位后，再等待1.5位的时间就开始一位一位地检测数据线的状态，得到所传输的数据。

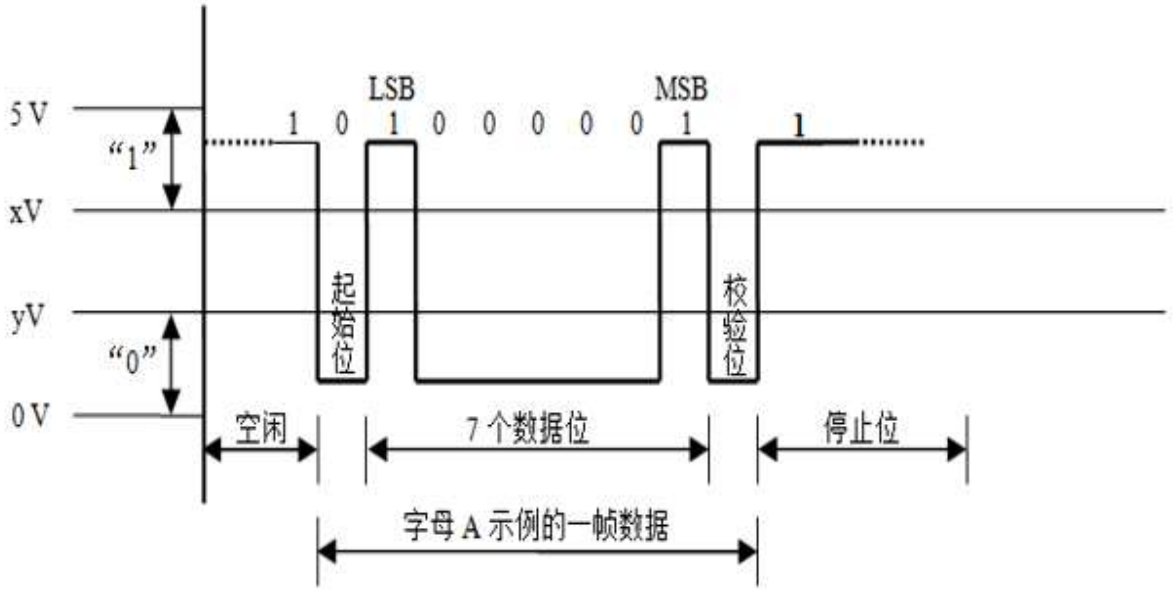
(3) UART一帧中可以有5、6、7或8位数据，发送方一位一位地改变数据线的状态，将它们发送出去，首先发送最低位。

TXD、RXD数据线以“位”为最小单位传输数据，其数据传输流程如下：

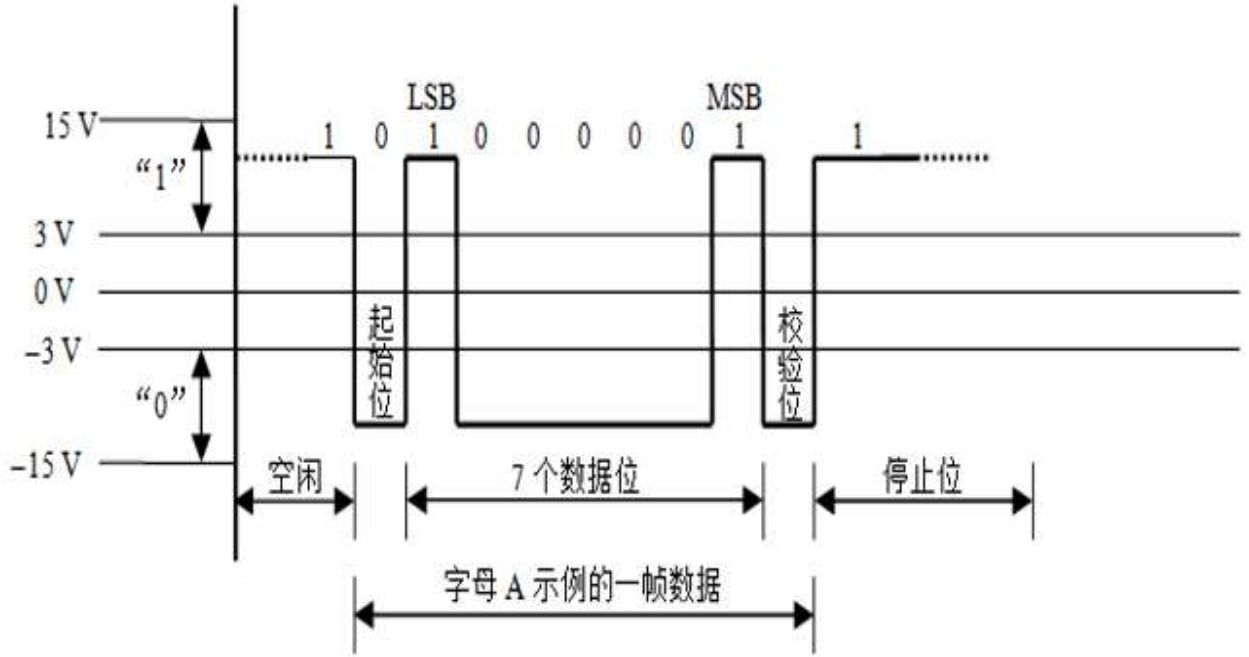
(4) 如果使用校验功能，UART在发送完数据位后，还要发送1个校验位。有两种校验方法，即奇校验和偶校验(数据位连同校验位中“1”的数目分别等于奇数或偶数)。

(5) 发送停止位，数据线恢复到“空闲”状态(1状态)。停止位的长度有三种：1位、1.5位、2位。

UART使用不同的电平发送字符“A”，所对应的波形：



(a) COMS逻辑电平下，传输大写字母“A”的帧格式



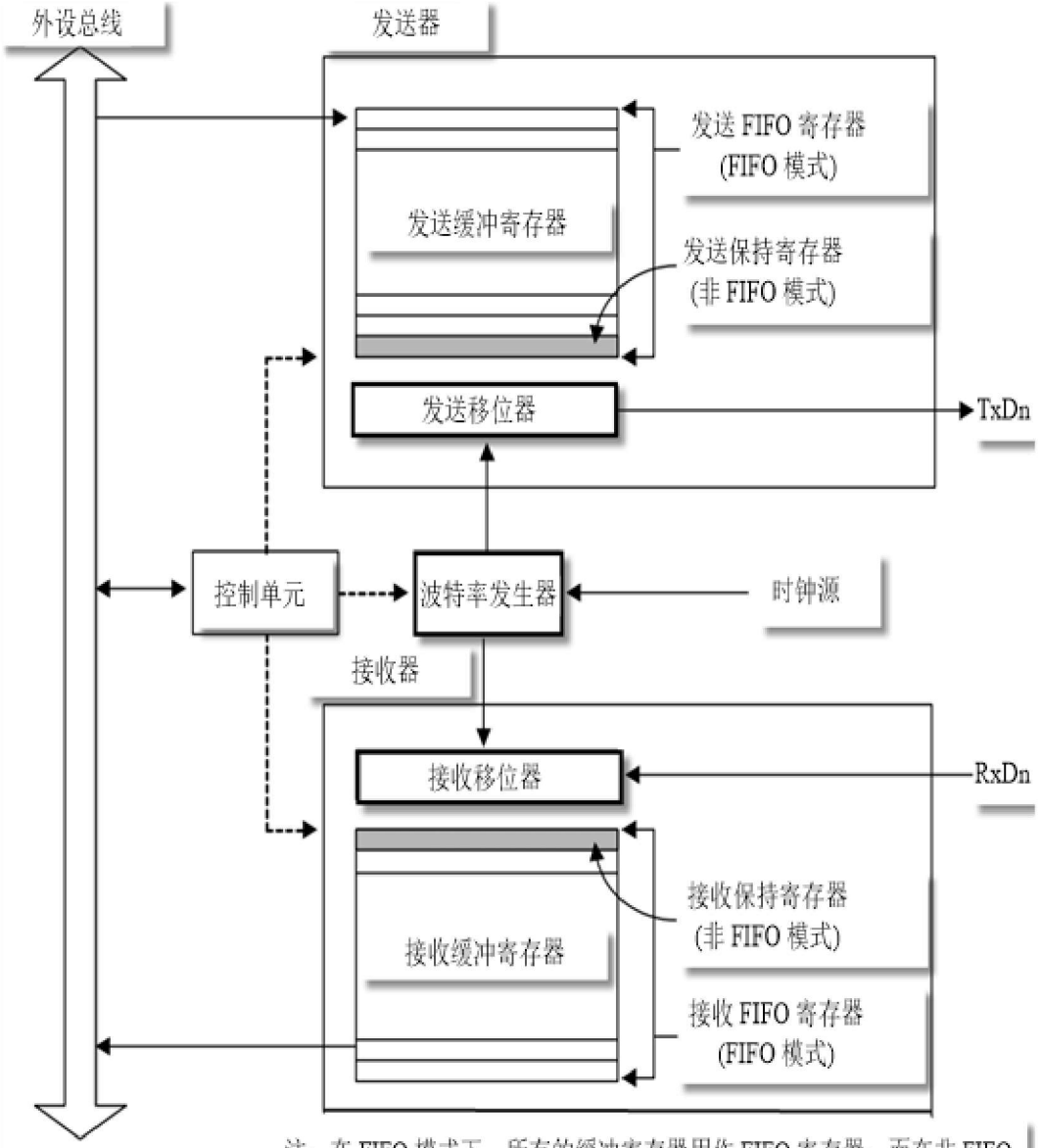
(b) R-S232逻辑电平下，传输大写字母“A”的帧格式

11.2 串行口特点

Exynos 4412的UART有4个独立的通道(通道UART0～UART3)，每个通道都可以工作于中断模式或DMA模式，即UART可以发出中断或DMA请求，以便在UART、CPU之间传输数据。另外，Exynos 4412还提供了一个带有GPS的串行通道(通道UART4)。

Exynos 4412的每个UART的通道包括2个FIFO(First In First Out)缓冲器，用于收/发数据。Exynos 4412 UART的每个通道支持的停止位有1位、2位，数据位有5位、6位、7位或8位，且支持校验功能，另外还有红外发送/接收功能。

Exynos 4412 UART的工作原理图


外设总线
发送器
发送 FIFO 寄存器
(FIFO 模式)
发送缓冲寄存器
发送保持寄存器
(非 FIFO 模式)
发送移位器
TxDn
控制单元
波特率发生器
时钟源
接收器
接收移位器
RxDn
接收保持寄存器
(非 FIFO 模式)
接收缓冲寄存器
接收 FIFO 寄存器
(FIFO 模式)
注：在 FIFO 模式下，所有的缓冲寄存器用作 FIFO 寄存器；而在非 FIFO 模式下，缓冲寄存器的一个字节用作保持寄存器。

在使用UART和PC进行通信时，在PC端往往需要设置波特率、数据位、是否使用校验位、有多少个停止位、是否使用流控等。要实现通信，Exynos 4412的UART也要作相同的设置。其具体的设置过程如下：

1. 将所涉及的UART通道引脚设为UART功能。

(2) 选择UART的时钟源。选择好时钟源后，可以通过$DIV_{UART0\sim4}$设置分频系数进行分频，通过CLK_DIV_PERIL0寄存器进行配置。

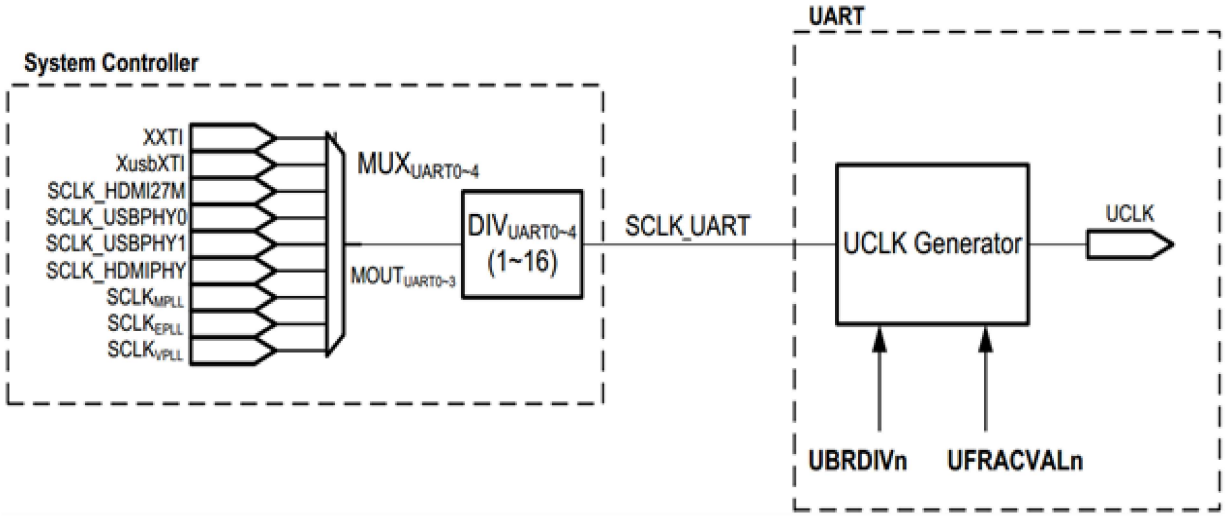


图11.3 Exynos 4412 UART时钟源框图

Exynos 4412 UART的时钟源有八种选择：XXTI、XusbXTI、SCLK_HDMI27M、SCLK_USBPHY0/1、SCLK_HDMIPHY、$SCLK_{MPLL}$、$SCLK_{EPLL}$、$SCLK_{VPLL}$

(3) 设置波特率。可以通过以下公式计算UBRDIVn寄存器(n为0～4，对应5个UART通道)的值：

$$UBRDIVn = (int)\frac{SCLK\_UART}{波特率 \times 16} - 1$$

注意：计算出来的UBRDIVn寄存器值不一定是整数，UBRDIVn寄存器取其整数部分，小数部分由UFRACVALn寄存器设置。

(4) 设置传输格式。传输格式由ULCONn寄存器(n = 0～4)进行配置。

(5) 设置UART工作模式。可参考UCONn寄存器的各位域进行配置。

(6) 配置UFCONn寄存器、UFSTATn寄存器。UFCONn寄存器用于设置是否使用FIFO，设置各FIFO的触发阈值，即发送FIFO中有多少个数据时产生中断，接收FIFO中有多少个数据时产生中断。并可以通过设置UFCONn寄存器来复位各个FIFO。

(7) 配置UTRSTATn寄存器(UART收/发状态寄存器)。UTRSTATn寄存器用来表明数据是否已经发送完毕，是否已经接收到数据。

(8) 配置UERSTATn寄存器(UART错误状态寄存器)。UERSTATn寄存器用来表示各种错误是否发生，位[0]至位[3]为1时分别表示溢出错误、校验错误、帧错误、检测到“break”信号。

(9) 配置UTXHn寄存器(UART发送缓冲寄存器)。CPU将数据写入UTXHn寄存器，UART即会将它保存到缓冲区中，并自动发送出去。

(10) 配置URXHn寄存器(UART接收缓冲寄存器)。当UART接收到数据时，读取URXHn寄存器，即可获得数据。

11.3 串行口专用寄存器

1. 串口时钟源选择寄存器

该寄存器用于为各通道的UART选择时钟源，如表11.1所示。

名称	位域	类型	功 能 描 述	复位值
通道 4 时钟选择	[19:16]	RW	0000 = XXTI；0001 = XusbXTI； 0010 = SCLK_HDMI24M 0011 = SCLK_USBPHY0 0101 = SCLK_HDMIPHY 0110 = SCLKMPLL_USER_T 0111 = SCLKEPLL 1000 = SCLKVPLL Others = Reserved	0x0
通道 3 时钟选择	[15:12]	RW	同上	0x1
通道 2 时钟选择	[11:8]	RW	同上	0x1
通道 1 时钟选择	[7:4]	RW	同上	0x1
通道 0 时钟选择	[3:0]	RW	同上	0x1

表11.1 串口时钟源选择寄存器(CLK_SRC_PERIL0)

2. 时钟分频系数寄存器

该寄存器用于为5路UART通道设置分频系数，如表11.2所示。

名 称	位域	类型	功 能 描 述	复位值
通道4分频因子	[19:16]	RW	通道 4 分频因子, SCLK_UART4 = MOUTUART4/(UART4_RATIO + 1)	0x0
通道3分频因子	[15:12]	RW	通道 3 分频因子, SCLK_UART3 = MOUTUART3/(UART3_RATIO + 1)	0x0
通道2分频因子	[11:8]	RW	通道 2 分频因子, SCLK_UART2 = MOUTUART2/(UART2_RATIO + 1)	0x0
通道1分频因子	[7:4]	RW	通道 1 分频因子, SCLK_UART1 = MOUTUART1/(UART1_RATIO + 1)	0x0
通道0分频因子	[3:0]	RW	通道 0 分频因子, SCLK_UART0 = MOUTUART0/(UART0_RATIO + 1)	0x0

表11.2 时钟分频系数寄存器(CLK_DIV_PERIL0)

3. 波特率分频寄存器**UBRDIVn(n = 0～4)**

该类寄存器用于设置波特率分频值。

4. 寄存器**UFRACVALn(n = 0～4)**

该类寄存器用来处理波特率分频值的小数部分。

5. UART控制寄存器ULCONn (n = 0～4)

该类寄存器主要用于配置串行数据帧的格式等，如表11.5

所示。

名称	位域	类型	功 能 描 述	复位值
红外模式	[6]	RW	0＝正常模式；1＝红外模式	0
奇偶校验	[5:3]	RW	0xx＝无校验；100＝奇校验；101＝偶；110＝校验位强制为 1；111＝校验位强制为 0	—
停止位个数	[2]	RW	0＝1 个；1＝2 个	0
数据位个数	[1:0]	RW	00＝5；01＝6；10＝7；11＝8	00

表11.5 ULCONn(n = 0～4)寄存器

6. UART控制寄存器UCONn(n = 0～4)

该类寄存器主要用于配置UART的工作方式，如表11.6所示。

名　称	位域	类型	功 能 描 述	复位值
Tx DMA Burst 大小	[22:20]	RW	000 = 1Byte；001 = 4 Byte；010 = 8 Byte；011 = 16 Byte；其余保留	0
Rx DMA Burst 大小	[18:16]	RW	000 = 1Byte；001 = 4 Byte；010 = 8 Byte；011 = 16 Byte；其余保留	0
接收超时中断间隔	[15:12]	RW	在 8 × (N + 1)时间内接收不到数据，则发生中断	0x3
空 FIFO 接收超时	[11]	RW	0 = 禁止；1 = 使能	0
接收超时 DMA 挂起	[10]	RW	0 = 禁止；1 = 使能	0
发送中断类型	[9]	RW	0 = 脉冲；1 = 电平	0
接收中断类型	[8]	RW	0 = 脉冲；1 = 电平	0
接收超时使能	[7]	RW	0 = 禁止；1 = 使能	0
接收错误状态中断使能	[6]	RW	0 = 禁止；1 = 使能	0
回送模式	[5]	RW	0 = 正常模式；1 = 回送模式	0
发送突变信号	[4]	RW	0 = 正常模式；1 = 发送突变信号	0
发送模式	[3:2]	RW	00 = 禁止；01 = 中断请求；10 = DMA 模式；11 = 保留	00
接收模式	[1:0]	RW	00 = 禁止；01 = 中断请求；10 = DMA 模式；11 = 保留	00

表11.6 UART控制寄存器UCONn(n = 0～4)

7. UART FIFO控制寄存器UFCONn(n = 0～4)

该类寄存器用于配置UART FIFO缓冲器的大小和触发水平等，

如表11.7所示。

名　称	位域	类型	功 能 描 述	复位值
Tx FIFO 触发水平	[10:8]	RW	[Channel 0] 000 = 0 Byte；001 = 32 Byte；010 = 64 Byte；011 = 96 Byte； 100 = 128 Byte；101 = 160 Byte；110 = 192 Byte；111 = 224 Byte [Channel 1, 4] 000 = 0 Byte；001 = 8 Byte；010 = 16 Byte；011 = 24 Byte； 100 = 32 Byte；101 = 40 Byte；110 = 48 Byte；111 = 56 Byte [Channel 2, 3] 000 = 0 Byte；001 = 2 Byte；010 = 4 Byte；011 = 6 Byte； 100 = 8 Byte；101 = 10 Byte；110 = 12 Byte；111 = 14 Byte	000
Rx FIFO 触发水平	[6:4]	RW	[Channel 0] 000 = 0 Byte；001 = 32 Byte；010 = 64 Byte；011 = 96 Byte； 100 = 128 Byte；101 = 160 Byte；110 = 192 Byte；111 = 224 Byte [Channel 1, 4] 000 = 0 Byte；001 = 8 Byte；010 = 16 Byte；011 = 24 Byte； 100 = 32 Byte；101 = 40 Byte；110 = 48 Byte；111 = 56 Byte[Channel 2, 3] 000 = 0 Byte；001 = 2 Byte；010 = 4 Byte；011 = 6 Byte； 100 = 8 Byte；101 = 10 Byte；110 = 12 Byte；111 = 14 Byte	000

表11.7 UART FIFO控制寄存器UFCONn(n = 0～4)

8. 发送寄存器**UTXHn(n = 0～4)**

该类寄存器用于存放待发送的数据。

9. 接收寄存器**URXHn(n = 0～4)**

该类寄存器用于存放接收到的数据。

11.4 应用实例

通过本实例讲解UART的配置和使用方法。该实例中使用了UART0，程序中仅对该串口进行了设置。目的是自串口输入一个字符，再从串口终端原样输出。

代码如下：

1. 定义指向寄存器物理地址的变量

```
//GPA0口将被设置为功能接口，用于UART的收/发
#define GPA0CON        (*(volatile unsigned int *)0x11400000)
//UART 时钟相关寄存器
#define CLK_SRC_PERIL0   (*(volatile unsigned int *)0x1003C250)
#define CLK_DIV_PERIL0        (*(volatile unsigned int *)0x1003C550)
//UART相关寄存器
#define UART_BASE              0x13800000
#define ULCON0             (*(volatile unsigned int *) UART_BASE+ 0x0000)
#define UCON0                  (*(volatile unsigned int *) UART_BASE+
0x0004)
#define UFCON0             (*(volatile unsigned int *) UART_BASE+ 0x0008)
#define UTRSTAT0               (*(volatile unsigned int *) UART_BASE+
0x0010)
#define UTXH0                  (*(volatile unsigned int *) UART_BASE+
0x0020)
#define URXH0                  (*(volatile unsigned int *) UART_BASE+
0x0024)
#define UBRDIV0            (*(volatile unsigned int *) UART_BASE+ 0x0028)
#define UFRACVAL0              (*(volatile unsigned int *) UART_BASE+
0x002c)
```

2. UART初始化

```
void uartInit( )
{
  /* 1. 配置GPA0位[0]、[1]为串口UART_0_RXD、UART_0_TXD功能 */
   GPA0CON & = ~(0xff);       //设置UART0对应的GPIO为UART功
能
   GPA0CON |= ((0x2<<0)|(0x2<<4));
  /*2. 设置UART时钟源SCLK_UART*/
   CLK_SRC_DMC |= (0x1<12);
   CLK_SRC_TOP1 |= (0x1<12);
   CLK_SRC_PERIL0 & = ~(0xf);      // UART0_SEL=6
   CLK_SRC_PERIL0 | = (0x6<<0);    //所以, MOUTUART0即等于MPLL的
输出, 800 MHz
    CLK_DIV_PERIL0 & = ~(0xf);
   CLK_DIV_PERIL0 | = (7<<0);
```

```
/* 3. 设置串口0相关 */
UFCON0 &= ~((0x7<<0)| (0x7<<4)| (0x7<<8));
UFCON0|= ((0x1<<0)| (0x1<<4)| (0x2<<8));
ULCON0 &= ~((0x3<<0)| (0x1<<2)| (0x3<<3));
ULCON0 |= ((0x3<<0)| (0x0<<2)| (0x0<<3));
UCON0 &= ~((0x3<<0)| (0x3<<2));
UCON0 |= ((0x1<<0)| (0x1<<2));
/* SCLK_UART0=100MHz, 波特率设置为115200
* 寄存器的值如下计算:
* DIV_VAL = 100 000 000 / (115200 × 16) - 1 = 53.25
* UBRDIVn0 = 整数部分 = 53
* UFRACVAL0 = 小数部分 × 16 = 0.25 * 16 = 4
*/
UBRDIV0 = 53;
UFRACVAL0 = 4;
}
```

3. 收发字符/字符串函数

```
char getc(void)
{
    char c;
    while (!(UTRSTAT0 & (1<<0))); //查询状态寄存器，直到接收完有效数据
    c = URXH0;       //读取接收寄存器的值
    return c;
}
void putc(char c)
{
   char c;
  while (!(UTRSTAT0 & (1<<2)));      //查询状态寄存器，直到发送缓存为空
   UTXH0 = c;                        //写入发送寄存器
   return 0;
}
void puts(char *s)
{
   while(*s)
   {
      putc(*s);
      s++;}}
```

4. 主函数

```
include "serial.h"
int main( )
{
  unsigned char c;
  uartInit( );
  while(1)
  {
    c = getc( );
    if (isDigit(c) || isLettle(c))
      putc(c);
  }
  return 0;
}
```

问题与思考：

1. 串行通信与并行通信的概念分别是什么？

2. RS-232C串行通信接口规范是什么？

3. 在Exynos 4412的串口控制器中，主要寄存器的作用是什么？

4. 编写一个串口程序，采用中断方式，实现向PC的串口终端打印一个字符串“hello”的功能。