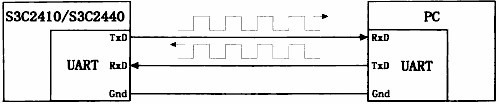
uart串口打印

Universal Asynchronous Receiver and Transmitter

**Uart原理**

发送数据时，CPU将并行数据写入UART，UART按照一定的格式在一根电线上串行发出；接收数据时，UART检测另一根电线上的信号，串行收集然后放在缓冲区中，CPU即可读取UART获得这些数据。UART之间以**全双工方式**传输数据，最精确的连线方法只有3根电线：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，连线如下：



UART使用标准的TTL/CMOS逻辑电平（0~5v、0~3.3v、0~2.5v或0~1.8v）来表示数据，高电平表示1，低电平表示0。为了增强数据的抗干扰能力、提高传输长度，通常将TTL/CMOD逻辑电平转换为RS-232逻辑电平，3~12v表示0，-3~-12v表示1

TxD、RxD数据线以位为最小单位传输数据，而**帧**由具有完整意义的、不可分割的若干位组成，它包含**开始位、数据位、校验位（需要的话）和停止位**。发送数据之前，UART之间要约定好数据的传输速率（即每位所占据的时间，其倒数称为波特率）、数据的传输格式（即有多少个数据位、是否使用校验位、是奇校验还是偶校验、有多少个停止位）。

**数据传输流程如下：**

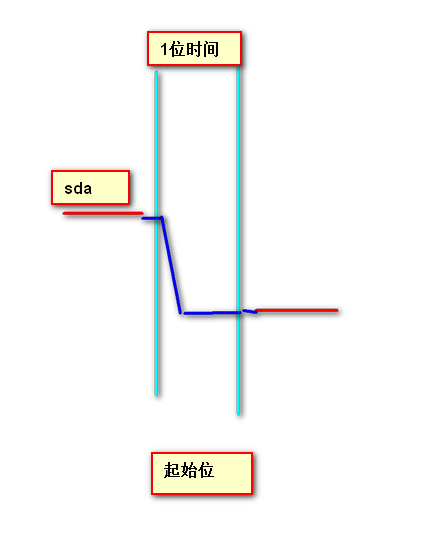
（1）平时**数据线**处于空闲状态（1状态）

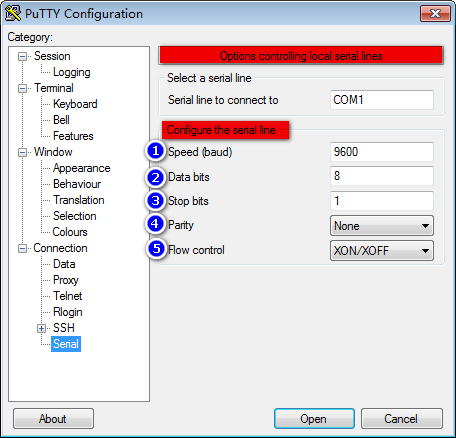
（2）当要发送数据时，UART改变TxD数据线的状态**（变为0状态）**并维持**1位**的时间，这样**接收方**检测到**开始位**后，在等待1.5位的时间就开始一位一位地检测数据线的状态得到所传输的数据。

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

（4）如果使用校验功能，UART在发送完数据后，还要发送1位校验位。有两种校验方法：奇校验、偶校验——数据位连同校验位中，1的数据等于奇数或偶数。

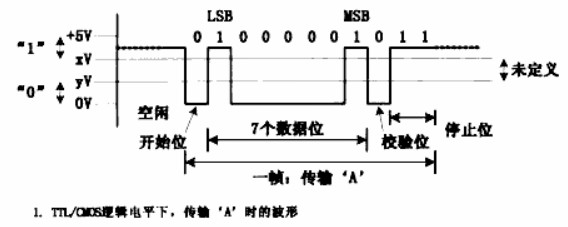
（5）最后，发送停止位，数据线恢复到空闲状态（1状态）。停止位的长度有3种：1位、1.5位、2位。

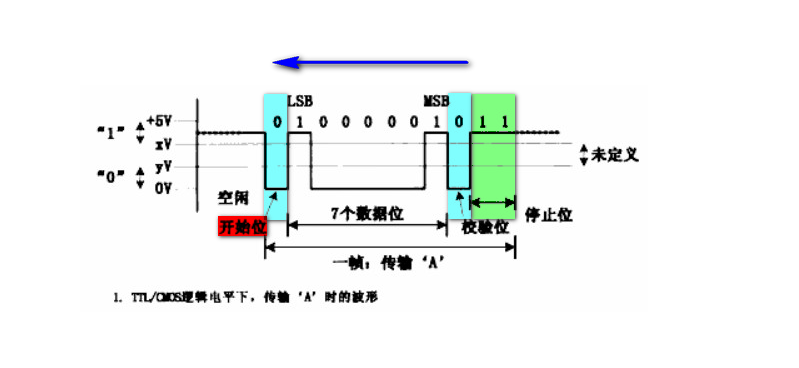




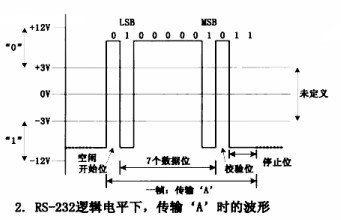
下图演示了UART使用7个数据位、偶校验、2个停止位的格式传输字符‘A’（二进制值为0b1000001）时，TTL/COM逻辑电平和RS-232逻辑电平对应的波形

TTL/COM逻辑电平





RS-232逻辑电平

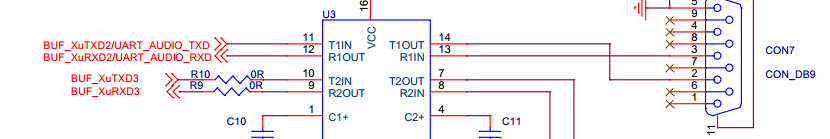


用途；

Uart连接PC和开发板，输出调试信息

实现功能：

用uart2实现与主机PC的通讯，实现数据的发送接收



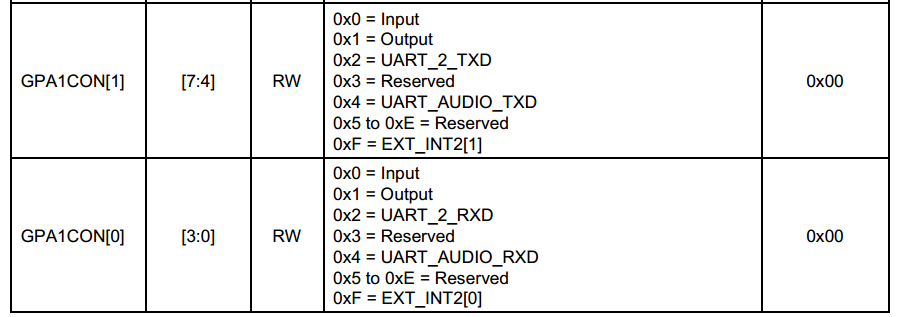


用到的是GPA1\_0和GPA1\_1

参数设置： 波特率: 115200 数据位: 8 校验位: 无 停止位: 1

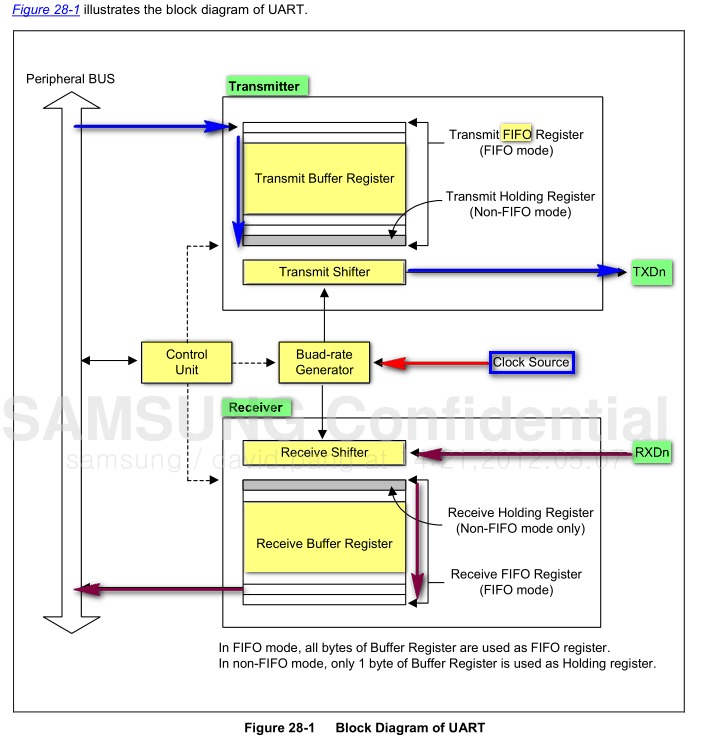
硬件如何驱动：

1. GPIO GPA1\_0 GPA1\_1的相关GPIO设置成UART

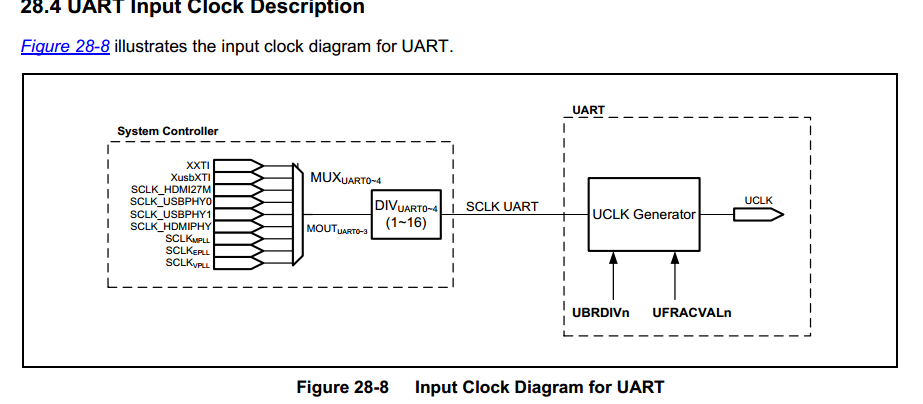


GPA1CON[0] = 0X2 GPA1\_0设置为UART\_2\_RXD

GPA1CON[1] = 0X2 GPA1\_1 设置为UART\_2\_TXD



1. uart时钟设置

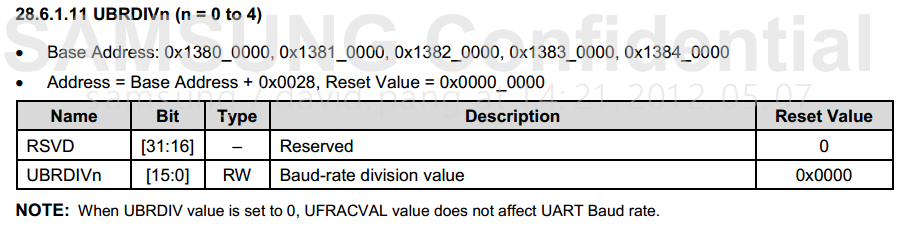


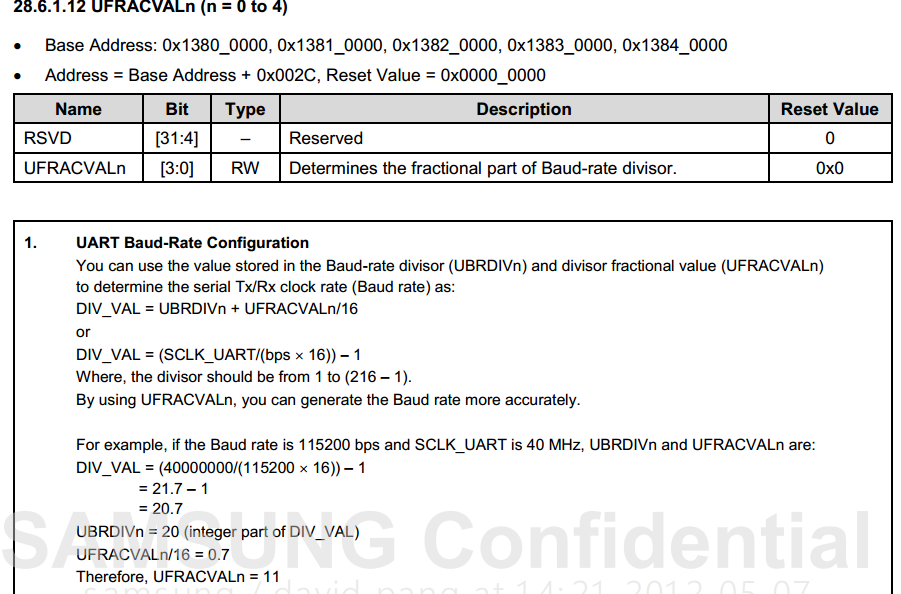
  Exynos4412 UART的时钟源有八种选择： XXTI 、XusbXTI 、SCLK\_HDMI24M 、SCLK\_USBPHY0 、 SCLK\_HDMIPHY 、SCLKMPLL\_USER\_T 、SCLKEPLL 、SCLKVPLL ，由 CLK\_SRC\_PERIL0 寄存器控制。  
选择好时钟源后，还可以通过 DIVUART0 ～4设置分频系数 设置分频系数 ，由 CLK\_DIV\_PERIL0 寄存器控制。 从分频器得到的时钟被称为SCLK UART 。 SCLK UART 经过上图中的“ UCLK Generator”后，得到UCLK ，它的频率就是UART 的波特率。“ Generator UCLK Generator ”通过这 2个寄存器来设置： UBRDIVn 、UFRACVALn (在下面描述)。

**设置波特率 UBRDEVn 、UFRACVALn**

SCLK\_UART 是100MHZ

我么这里是UART2，所以关于波特率涉及到的寄存器是UBRDEV2和UFRACVAL2





关于波特率和这两个寄存器之间的计算关系，datasheet写的很明白：

  根据给定的波特率、所选择时钟源频率，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)( UART clock / ( buad rate x 16) ) – 1**

     上式计算出来的 UBRDIVn 寄存器值不一定是整数， UBRDIVn 寄存器取其整数部分，小部分由 UFRACVALn 寄存器设置， UFRACVALn 寄存器的引入，使产生波特率更加精确。  
  
我们这里 UART clock为100MHz，要求波特率为115200 bps，则：

100000000/(115200 x 16) – 1 = 54.25 – 1 = 53.25

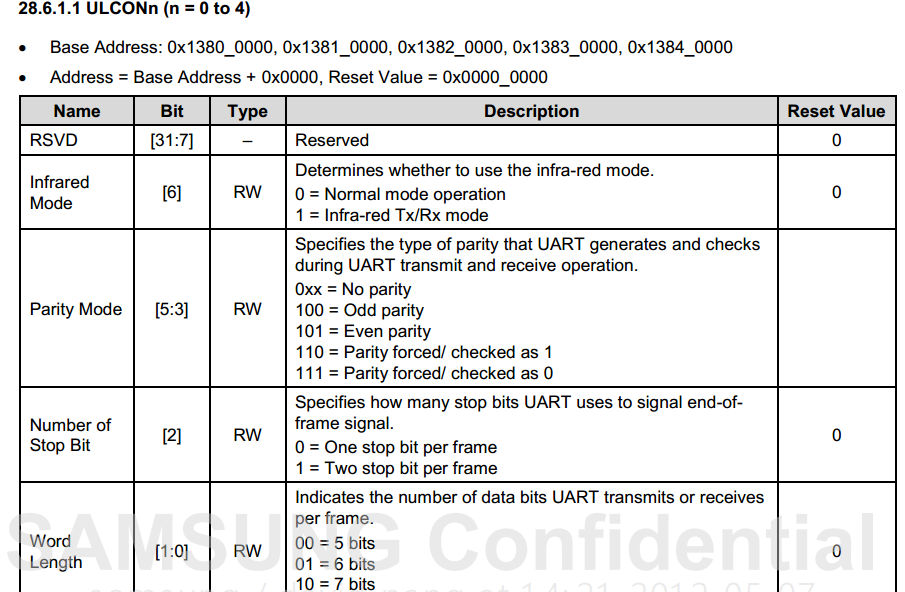
UBRDIVn = 整数部分 = 53

UFRACVALn/16 = 小数部分 = 0.25

UFRACVALn = 4

所以设置寄存器 UBRDIV2 = 53 ,UFRACVAL2 = 4

1. 设置uart相关参数



ULCON2[8] 红外线模式还是普通模式 ，我们选择普通模式 0

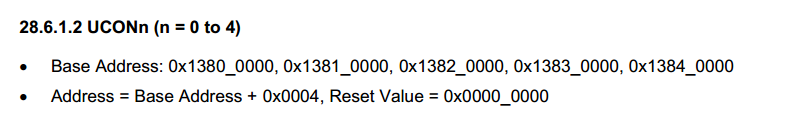
ULCON2[1:0] 数据位数 这里我们设置为8bit ，0x3

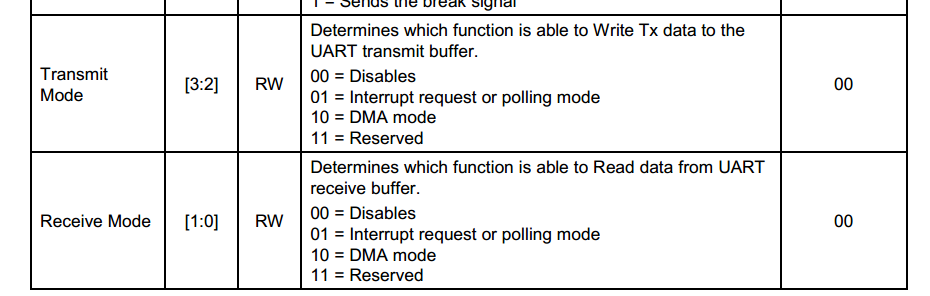
ULCON2[2] 停止位位数， 我们设置为1bit，0x0

ULCON2[5::3] 校验位 我们设置为无 0x0

综上，ULCON2 = 0b11 = 0x03

1. 设置uart 收发模式 UCONn

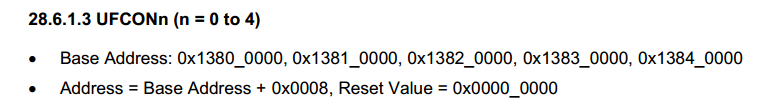




UCON2 [1:0] = 01, 接收 poll模式

UCON2[3:2] = 01, 发送 poll模式

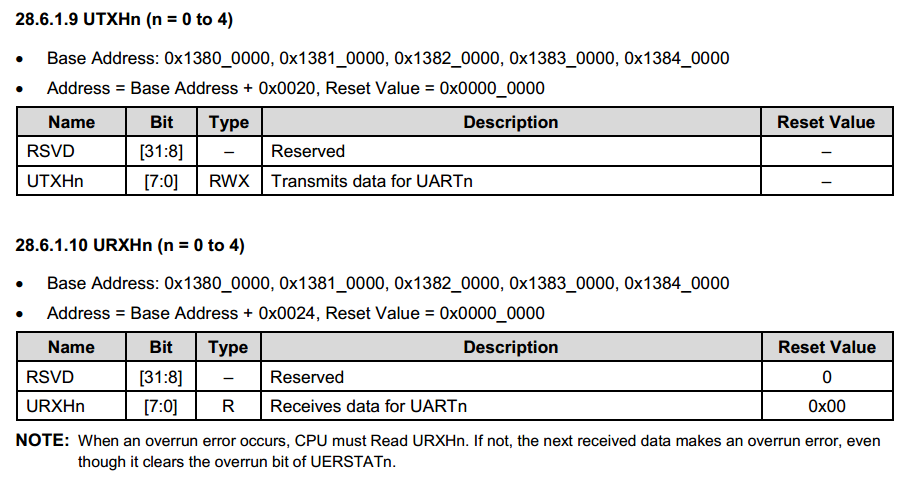
1. 关闭发送接收FIFO





UFCON2 [0] = 0 关闭发送接收FIFO

1. 实现发送数据

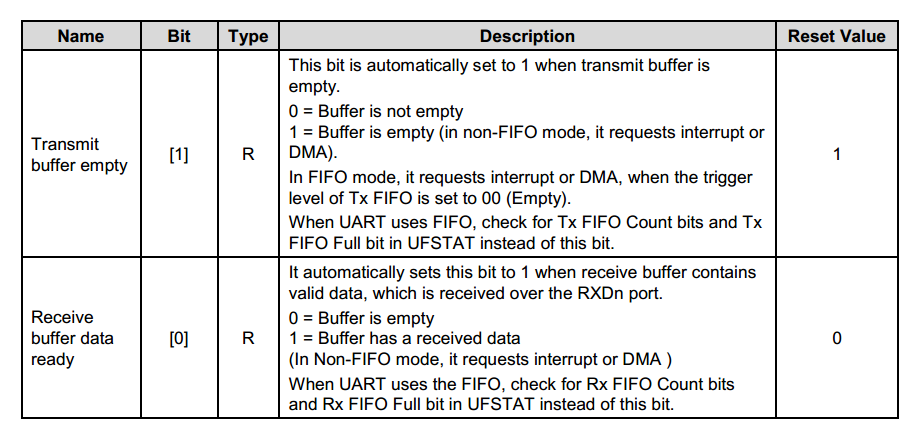
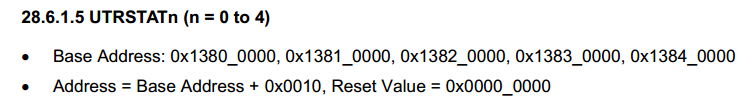
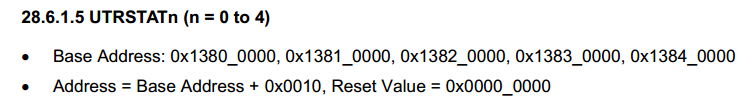


发送数据

将要发送的8bit数据放入UTXHn寄存器低8位

UTXH2 = txdata

检测数据是否发出完成



检测是否发送完毕，检测寄存器UTRSTAT2的bit 1是否置1，置1表示发送完成。

发送代码：

// write fifo

UTXH[0:7] = data;

// wait finish

while (!(UTRSTAT[1] == 0b1));

8.接收数据

将寄存器URXHn的低8位数据取出来。

但是要等到receive buffer data ready，也即是有数据可读。查看是否有数据可读可以查看寄存器UTRSTAT2的bit 0，置1表示有数据可读。

接收代码：

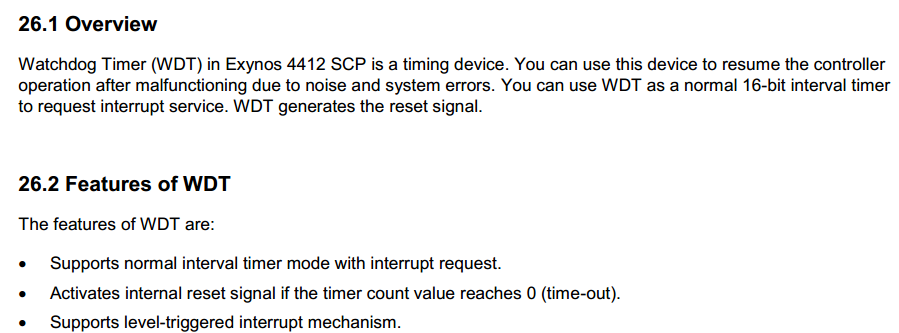
// wait for data

while (!(UTRSTAT[0] == 0b1));

// read fifo

data = URXH[0:7];

**看门狗(Watchdog Timer)**



看门狗定时器是一个定时器设备，所谓定时器(可以想象成定时炸弹)，我们可以用看门狗定时器来复位控制器在发生噪音和系统错误之后。我们可以使用WDT作为一个普通的16位定时器，产生中断或者产生复位信号。

看门狗的特点：

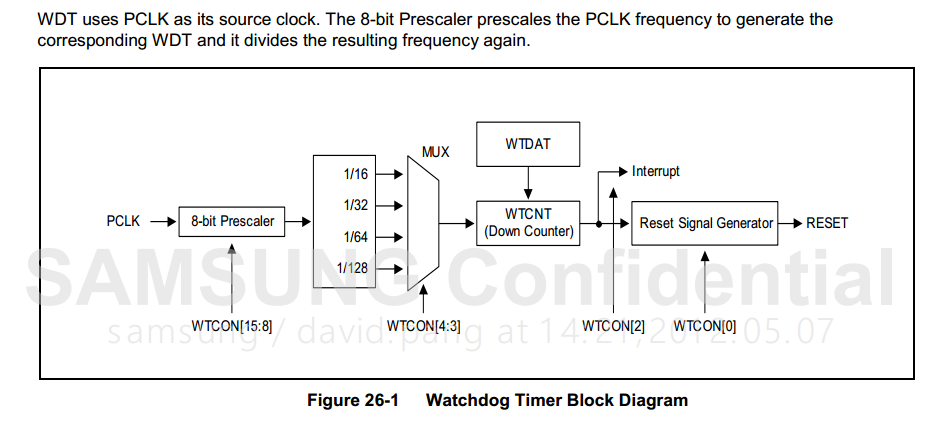
支持普通定时器产生中断

当定时器计数寄存器变为0的时候，系统复位。

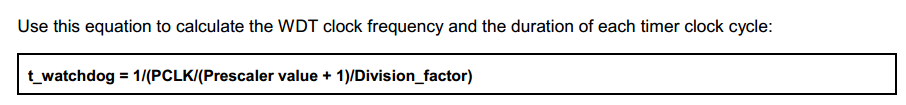
实现功能：

当产生数据终止异常时，处理该异常，启动看门狗，10s之后复位。

关于看门狗定时器的操作：



看门狗模块包括一个预比例因子放大器，一个四分频的分频器，一个16位计数器。看门狗的时钟信号源来自PCLK，为了得到宽范围的看么狗信号，PCLK先被预分频，然后再进过分频器分频。预分频比例因子和分频器的分频值，都可以由看门狗控制寄存器(WTCON)决定，预分频比例因子的范围是0~255，分频器的分频比可以是16、32、64或128。看门狗定时器时钟周期的计算如下：



式中Prescaler value 为预分频比例放大器的值；Divison\_factor是四分频的分频比，可以是16、32、64或128

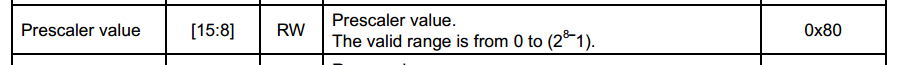
  一旦看门狗定时器被允许，看门狗定时器数据寄存器（WTDAT）的值就不能被自动地装在到看门狗定时器（WTCNT）中。因此，看门狗启动前要将一个初始值写入看门狗计数器（WTCNT）中。当4412用嵌入式ICE调试时，看门狗定时器的复位功能就不被启动，看门狗定时器能从CPU内核信号判断出当前CPU是否处于调试状态。如果看门狗定时器确定当前模式是调试模式，尽管看门狗产生溢出信号，但是仍然不会产生复位信号。

启动看门狗，设置WTCON[0]和WTCON[5]为1。

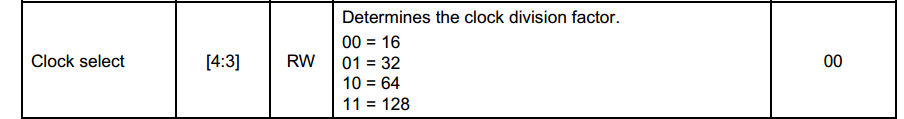
**看门狗定时器初始化：**

1. 设置开门狗定时器的定时时间

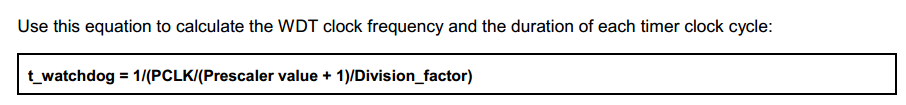
WTCON[15:8] = 0xff //预分频器设置为255



WTCON[4:3] = 0X3 //分频器设置为128



那么一个tick时钟周期时间用如下公式计算：



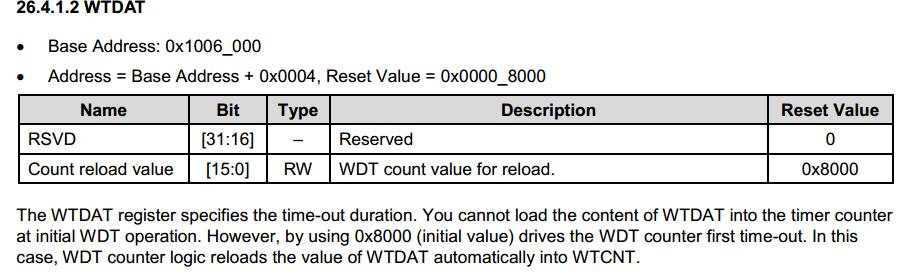
Tick = 1/(100Mhz/(255+1)/128) = 1/3051 s

看门狗定时器是寄存器WTCNT的值从WTDAT减到0的时候产生中断或者复位信号。

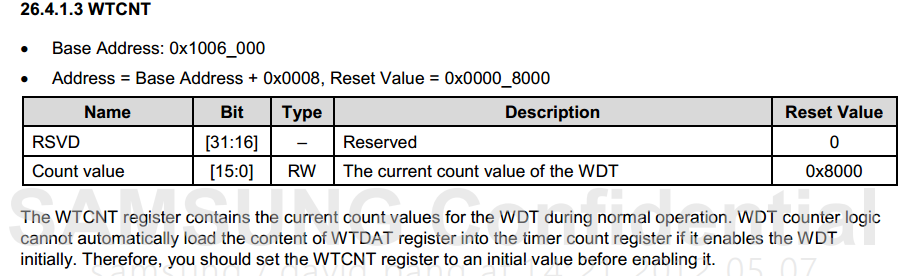
所以我们要设置WTDAT和WTCNT寄存器的值。

这里面关键是我们要设置WTCNT和WTDAT的初始值为多少？

WTDAT用于指定超时时间，在看门狗把复位功能禁止并打开中断使能后，此时看门狗定时器就是一个普通的定时器，使用方法和普通定时器一样。当使用复位功能后，由于WTCNT的值减到0时，系统就会复位，所以WTDAT的值装不进看门狗计数寄存器（WTCNT）中。复位后初始值为0x8000。WTDAT描述如下：



 WTCNT包含看门狗定时器工作的时候，计数器的当前计数值。WTCNT描述如下：



我们要10s之后产生复位信号，而一个tick是1/3051s

那么WTDAT = 10\*3051 = 30510

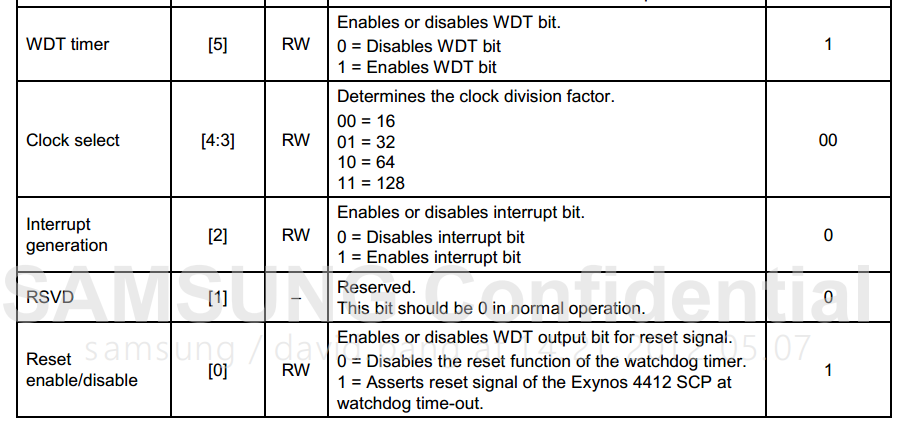
WTCNT也需要设置初始值，我们可以看到最后一句英文注释是这么讲的。

WTCNT = 10\*3051

这样WTCNT计数器就会从3051\*10 减到0，总共耗时10s。

1. 启动看门狗定时器 WTCON

关于看门狗定时器的启动，主要是看WTCON寄存器



我们可以看到WTCON[5]是使能或者关闭看门狗定时器的。但是WTCON[2]和WTCON[0]是分别用来控制是产生中断还是复位信号，这里我们是需要产生复位信号，所以如下设置：

WTCON[5] = 1

WTCON[2] = 0

WTCON[0] = 1

这样关于看门狗定时器的启动就完成，我们只需要在取数据异常处理函数里面启动看门狗定时器，10s之后就会产生复位。

1. 补充data\_abort异常处理代码

**\_data\_abort:**

**.word** data\_abort

**data\_abort:**

ldr sp,=0x02058000 @数据异常栈

stmfd sp!,{r0-r12,lr}

ldr pc,=data\_abort\_error\_handler @跳到数据异常处理函数执行

1. 实现数据异常处理函数

**void** **data\_abort\_error\_handler**()

{

//启动看门狗，10s后复位

wdt\_start();

//做些事情，等待复位，比如让灯闪烁

**while**(1)

{

//亮

REG\_WRITE(GPX2DAT,REG\_READ(GPX2DAT) & (~(0x1<<7)) | (0x1<<7));

mydelay\_ms(500);

//灭

REG\_WRITE(GPX2DAT,REG\_READ(GPX2DAT) & (~(0x1<<7)));

mydelay\_ms(500);

}

}

1. 主函数初始化wdt，并制造一个数据终止异常事件出来，验证是否10s后看门狗复位

wdt\_init();//看门狗初始化，设置看门狗定时器时间为10s

**while**(1)

{

mydelay\_ms(2000);

//自造数据异常事件，0x0地址不可读写

p = (**int** \*)0x0;

val = \*p;

}

问题:

[1]：时序图

站在时序图的左面看，数据是从右边来的，是一个随时间变化的动态过程

[2]: 4412的带控制器的外围设备的控制

大致上分两部分  
 [1].GPIO的设置

[2].控制的设置(中断GIC,UART)

[3]: 串口的操作

A.串口的初始化

B.数据发送

C.数据接收