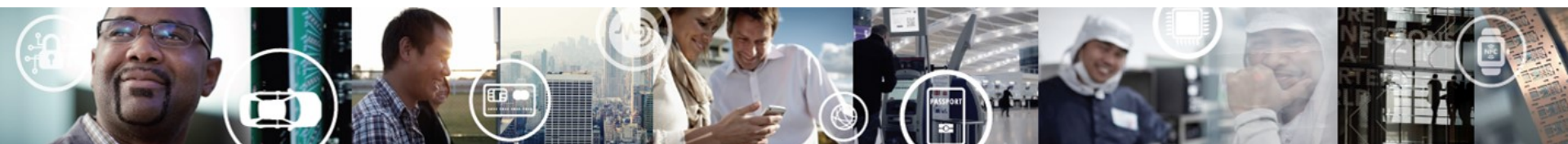


# LPC82X 培训资料

## 异步串行接口USART

MAY, 2016



EXTERNAL USE



SECURE CONNECTIONS  
FOR A SMARTER WORLD

# 内容

- USART性能概述
- USART具体应用操作
- USART低功耗模式唤醒

# USART性能概述

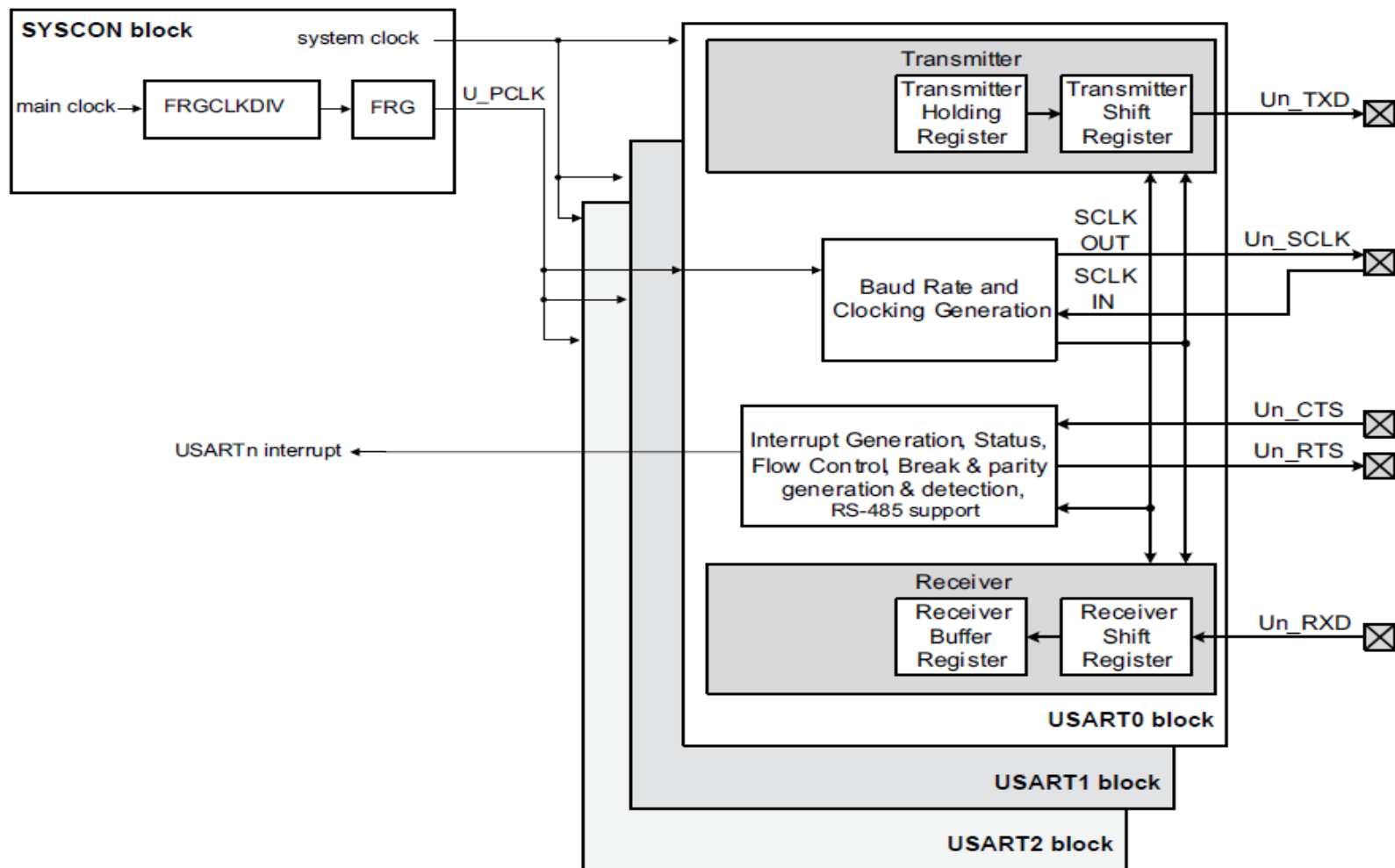
# 性能概述 - 基本功能

- 异步下最高速率可达1.875Mbits/s
- 同步下最高速率可达10Mbits/s
- 数据格式：7、8、9个数据位, 1、2个停止位
- 数据校验：无校验/奇校验/偶校验
- 中断源：接收就绪，发送就绪，发送器闲置，校验错，帧错
- DMA传输：数据收发可使用DMA
- 每数据位的过采样倍率从5-16，从最接近位时间中央的3个点中表决
- 分数波特率发生器：可使用常用时钟产生包括115200在内的几乎所有波特率

# 性能概述 – 扩展功能

- 使用RTS和CTS信号支持硬件流控
- 可以产生与识别break信号
- RS-485通信模式
- 多机通信（9个位）模式，软件匹配地址
- 同步通信模式，支持主机和从机，可以选择数据相位和连续时钟输出
- 自动检测波特率
- 异步通信模式下只能从睡眠模式下唤醒
- 同步通信模式下可以从全部低功耗模式下唤醒

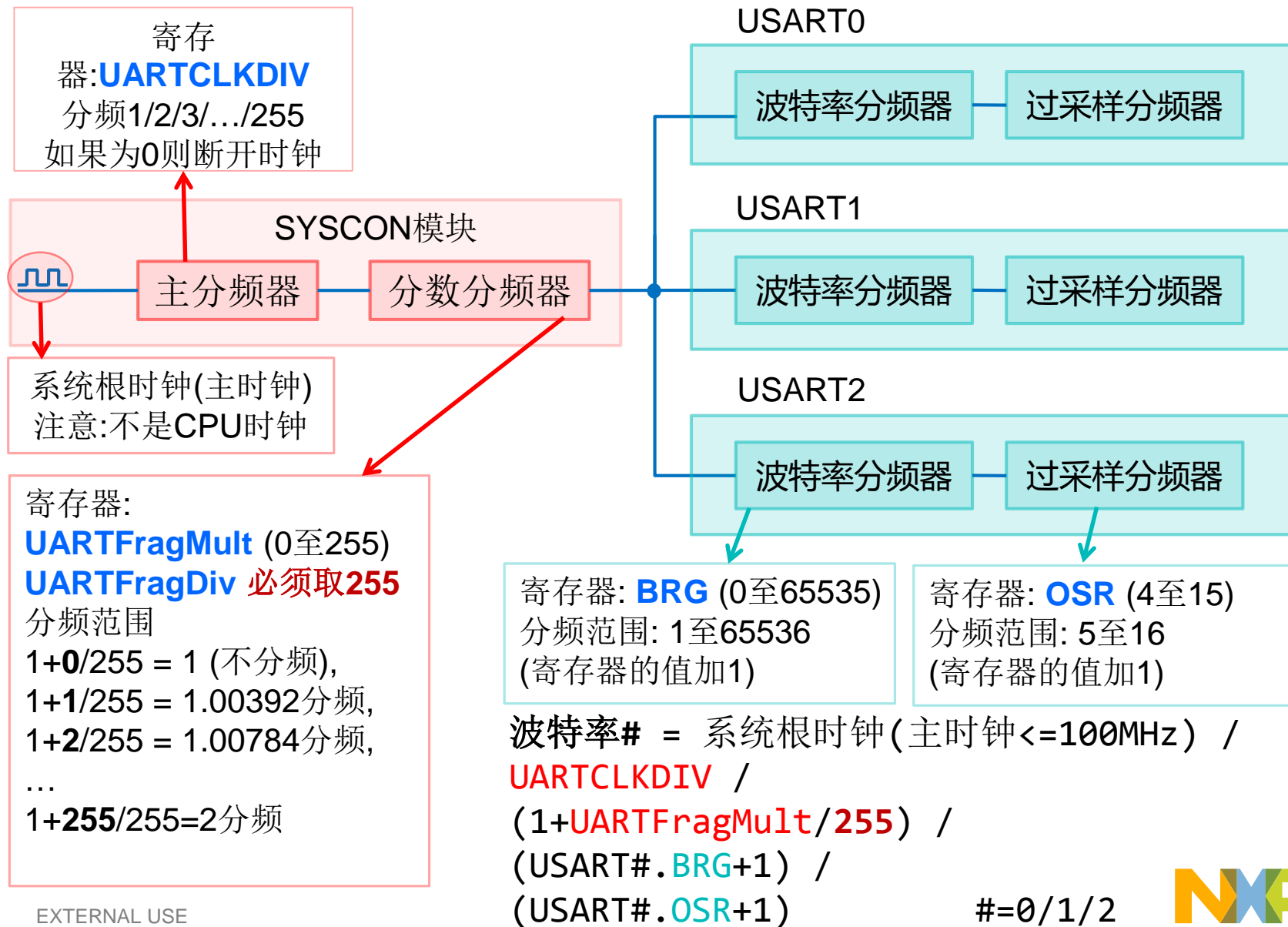
# USART功能框图



$$U\_PCLK = FRGCLKDIV / (1 + MULT / DIV)$$

# USART具体应用操作

# 异步模式下波特率的产生





# 使用USART模块的主要步骤

SYSCON

- 通过SYSCON模块 ( SYSAHBCLKCTRL:bit14/15/16 ) 开启 USART0/1/2模块的时钟，解除复位PININT模块
- 通过SYSCON模块配置全部UART的主分频器和分数分频器

SWM,  
IOCON

- 通过SWM分配引脚，通过IOCON正确配置引脚属性

NVIC  
/DMA

- 通过NVIC打开对应的USART中断源 ( 每个USART使用1路 )，按需配置优先级。USART亦受DMA的支持

USART#

- 配置USART模块自身 ( 见下文详述 )

# 以“115200,n,8,1”收发串口数据 基本配置

- 目标: 配置USART在12MHz的主时钟下异步通信, 串口设置为115200, 无流控, 8, 1

## -1. 通过USART的CFG寄存器配置关键参数:

- 使能USART:  $\text{CFG}[0:0] = 1$
- 配置数据位为8位:  $\text{CFG}[3:2] = 1$
- 配置校验为无校验:  $\text{CFG}[5:4] = 0$
- 配置停止位为1个:  $\text{CFG}[6:6] = 0$
- 配置为无流控:  $\text{CFG}[9:9] = 0$

- 不使用同步收发模式:  $\text{CFG}[12:11,14] = 0$
- 不使用环回(自发自收):  $\text{CFG}[15:15] = 0$
- 不使用RS-485:  $\text{CFG}[21:18] = 0$
- 不反向识别波形极性:  $\text{CFG}[23:22] = 0$
- 寄存器配置 (未定义的空位必须写0):
  - $\text{UART}\#\text{.CFG} = 1 \ll 0 \mid 1 \ll 2 \mid 0 \ll 4 \mid 0 \ll 6 \mid 0 \ll 9 \mid 0 \ll 11 \mid 0 \ll 14 \mid 0 \ll 15 \mid 0 \times \text{F} \ll 18 \mid 3 \ll 22;$
- 相关API函数: `Chip_UART_ConfigData()`

用不着的功能, 但是又必须妥善处理

USART		<>
CFG		
0	Enable	D/e
1		
2	DataLen	
3	7b/8b/9b/-	
4	ParitySel	
5	no/-/Even/Odd	
6	StopEn1b/2b	
7		
8		
9	CTSEn	!FlowCtl/CTS
10		
11	SyncEn	UART/SART
12	ClkPol	(SyncMode) SCLK EdgFal/Ris
13		
14	SyncMst	master n/y
15	Loop	loopback n/y
16		
17		
18	OETA	OExtra1B n/y
19	AutoAddr	D/E
20	OEsel	RTS usage Std/as OE 4 RS485
21	OEPol	IOEActvL/H
22	RXPOL	-Std/Inv
23	TXPOL	-Std/Inv
24		
..		
31		

# 以“115200,n,8,1”收发串口数据 设置波特率

## -2. 配置波特率，有多种解法，当主时钟为12MHz时：

- 保守解法: 先配置整数分频器，有误差时用分数分频器弥补
  - `SYSCON.UARTCLKDIV=1`
  - `USART#.OSR=16-1=15` (标准过采样率就是16)
  - `USART#.BRG=6-1=5` ( $12000000/115200/16 = 6.51$ )
  - `SYSCON.UARTFRAGMULT=22` ( $(6.51/6-1)*255=22$ )
  - 波特率= $12000000/1/(1+22/255)/6/16=115072$ , 误差约0.12%
- 一种“投机”解法 :在线路较好时不使用分数分频器, 而是减少过采样倍率
  - `SYSCON.UARTCLKDIV=1`
  - `SYSCON.UARTFRAGMULT=0` (不使用分数分频器)
  - `USART.OSR=13-1=12` (减少过采样倍率,  $12000000/115200/13=8.013$ , 除不尽的部分最小)
  - `USART.BRG=8-1=7` (最后再确定BRG的值)
  - 波特率= $12000000/1/(1+0/255)/8/13=115384$ , 误差约为0.3%
  - 注意：减少过采样倍率会提高对信号波形的质量要求
- 相关API函数 (仅支持16倍过采样)：
  - `Chip_Clock_SetUSARTNBaseClockRate()` , `Chip_UART_SetBaud()`

# 以“115200,n,8,1”收发串口数据 杂项配置

## -3. 通过USART的CTL寄存器使能发送器

- 解除对发送器的禁用:  $CTL[6:6] = 0$
- 不发送断路信号:  $CTL[1:1] = 0$
- 不使用RS-485的地址检测功能:  $CTL[2:2] = 0$
- 不使用同步模式(连续时钟发送):  $CTL[9:8] = 0$
- 不做自动波特率检测:  $CTL[16:16] = 0$
- 寄存器配置:
  - $UART\#.CTL = 0 \ll 1 \mid 0 \ll 2 \mid 0 \ll 6 \mid 0 \ll 8 \mid 0 \ll 16;$
- 相关API函数: `Chip_UART_TXEnable();`

用不着的功能，但是又必须妥善处理

USART		<>
CTL		
0		
1	TXBrkEn	-D/E
2	AddrDET	
	AddrDetect	N/Y
3		
4		
5		
6	TxDis	-N/Y
7		
8	CC -cntnusClk	N/Y
9	ClrCCOnRx	N/Y
10		
..		
15		
16	AutoBaud	D/E
	SetOnlyIfRxIdle	
	Measur1stStartBit	
	UpdtBRGAdpt	
17		
..		
31		

# 以“115200,n,8,1”收发串口数据 中断配置

-4. 通过两个寄存器INTENSET和INTENCLR来使能需要的中断，禁止不需要的中断：

- 它们均写1有效。INTENSET还用于读取当前使能的中断
- 使能接收就绪中断：INTENSET[0:0]=1
- 发送器闲置中断：INTENSET[3:3] = 1
  - 使能：发送器一空闲就发中断，所以要在发送一个数据字后再使能
  - 禁用：在发送全部数据后，要禁用这个中断。
- 使能接收数据溢出中断：INTENSET[8:8] = 1
- 禁用发送器就绪中断: INTENCLR[2:2] = 1
  - 发送器就绪中断有助于提高发送效率，但是中断产生时最后一个数据很可能尚在发送中（只是可以接收下个数据而已），容易出错。
- 禁用其它中断
- 寄存器操作：
  - 实现技巧：可以先除能全部中断，再使能需要的中断：
    - UART#.INTENCLR = 0x0001F96D; (空白位不能写1)
    - UART#.INTENSET = 1<<0 | 1<<3 | 1<<8;
- 相关API函数：
  - Chip\_UART\_IntEnable(), Chip\_UART\_IntDisable()

USART	
IntEnSet	<>1
IntEnClr	>1
<IntStat	EnabledInt
0	RxRdyEn
1	
2	TxRdyEn
3	TxIdleEn
4	-
5	DeltaCTSEn
6	TxDisEn
7	
8	OverrunEn
9	
10	-
11	DeltaRxBrkEn
12	StartEn
13	FrameErrEn
14	ParityErrEn
15	RxNoiseEn
16	ABErrEn
17	
..	
31	



# 以“115200,n,8,1”收发串口数据 标志位

-4. USART的STAT寄存器显示和清除当前的事件标志，有些标志写1清除，有些由硬件自动设置和清除。

- 接收到数据标志: STAT[0:0], 只读，读取数据时自动清除
- 发送器闲置标志：STAT[3:3], 只读，写入新数据时自动清除
- 接收数据爆棚标志：STAT[8:8], 手动写1清除。
- 其它略
- 如果是使用查询方式，需要在循环中轮询上述标志：
  - 例如：while (!(LPC\_UART#->STAT & 1<<0) 等待数据接收
- 如果是使用中断方式，只需在INTENSET寄存器中打开相应标志对应的中断开关

USART	
Stat	
0	<RxRdy -ClrOnRdDat
1	<RxIdle -0=isRxing
2	<TxRdy -OkToWrDat
3	<TxIdle ->=TxRdy
4	<CTS -RT CTS state Unless loopback
5	DeltaCTS CTS changed >1c
6	<TxDisStat XmtrIdleAftRDis
7	
8	OverrunInt >1c
9	
10	RxBrk >1c RxLowFor16bitItvs
11	DeltaRxBrk>1c
12	Start >1c StartBitDtct For wakeup in sSp
13	FrameErrInt>1c
14	ParityErrInt>1c
15	RxNoiseInt >1c 3vote2,SetIf1Diff
16	ABErr >1c AutoBaudErr SetIfBRGCntrOvf1
17	
..	
31	

# 以“115200,n,8,1”收发串口数据 数据处理

- 发送数据：

- 在“发送器闲置”(STAT.bit2)或“发送器就绪”(STAT.bit3)标志设置时，往USART的TXDAT寄存器写数据

- 接收数据

- 轮询“接收到数据”(STAT.bit0)标志或者打开接收中断
- 读取USART的RXDAT寄存器可以读取纯数据，或者
- 读取USART的RXDATSTAT寄存器，这会在读取的每笔数据上同时附着对应的错误标志

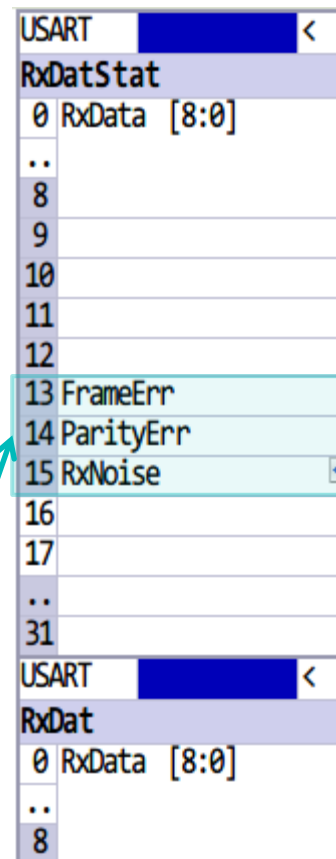
- [8:0]：数据字

- [13:13]: 这个数据字存在帧错误（停止位错）

- [14:14]:这个数据字存在奇/偶校验错

- [15:15]: 接收这个数据字时线路噪音大(过采样表决“意见”不统一)

- 读取数据时会自动清除“接收到数据”(STAT.bit0)标志，因此ISR无需手工清除标志。



# USART上的DMA数据收发

- 每个USART的每个方向(接收/发送)各对应一个DMA通道
- USART对DMA的请求是自动完成的，在USART上无需作任何配置
  - 每当发送器有数据要发送时，就在发送通道上产生DMA请求
  - 每当接收器收到新数据时，就在接收通道上产生DMA请求
- 在DMA控制器上正确初始化
  - 填写对应通道的描述符(16字节)
  - 配置对应的DMA通道(DMA.CFG#寄存器)
  - 开启传输完成中断
  - 使能外设请求

DMA通道号	USART数据路径
0	USART0. RX (数据接收)
1	USART0. TX (数据发送)
2	USART1. RX (数据接收)
3	USART1. TX (数据发送)
4	USART2. RX (数据接收)
5	USART2. TX (数据发送)



# USART对RS-485的支持 接收时地址与数据的管理

- 从设备的地址识别与数据接收
  - 1. 在USART#.CFG[3:2]位段中，设置数据位为9个
  - 2. 设置USART#.ADDR寄存器为本机从设备地址
  - 3. 设置USART#.CFG.bit19 (AUTOADDR) 为1
  - 4-1. 接收地址时，设置USART#.CTL.bit2 (ADDRDET)为1，此时仅当收到的数据第9位为1，并且8位地址与ADDR的设置相匹配时才接受（产生中断/DMA）
  - 4-2. 接收数据时，设置USART#.CTL.bit2(ADDRDET)为0，此时接收全部数据
- 主设备上的操作
  - 在USART#.CFG[3:2]位段中，设置数据位为9个
  - 主设备无需辨别地址还是数据，可以不作其它配置

# USART对RS-485的支持 半双工数据发送的实现

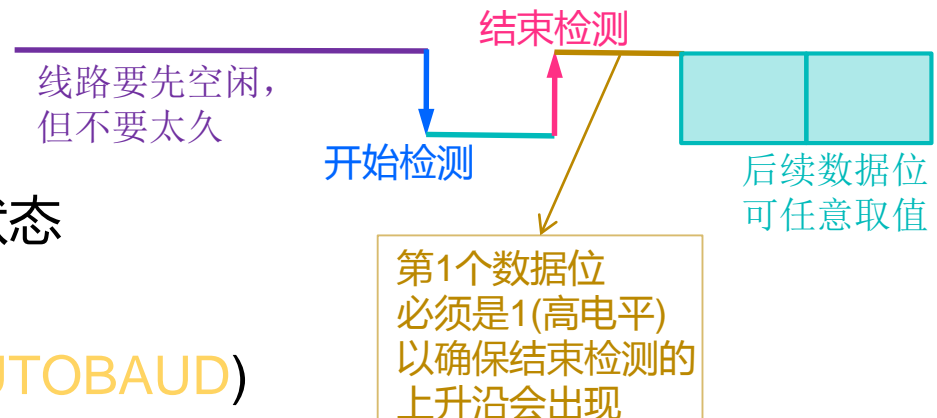
- USART的"RTS"信号在RS-485模式下当作"OE"信号使用
- 控制"OE"信号以实现半双工
  - 设置USART#.CFG.bit20(OESEL)为1, 把"RTS"信号挪用为RS-485的"OE"信号
  - 根据RS-485收发器对OE极性的要求设置USART#.CFG.bit21(OEPOL)
    - 0=OE低电平有效, 1=OE高电平有效
  - 根据软件反应的延迟设置USART#.CFG.bit18(OETA)
    - 0=发完数据后立即解除OE有效, 1=发完数据后再等一个帧的时间才解除OE有效
  - 使用DMA发送时可立即释放OE, 使用中断发送时最好等待一个帧时间给ISR作处理。

# 自动检测波特率

- 工作原理：检测起始位的持续时间，要求起始位后面的第1个数据位必须是1——也就是说数据必须是奇数(1,3,5,...255)。
- 检测到起始位后，更新BRG寄存器的值到最接近的，不处理其它的3个分频因子

- 使用步骤

- 确保首先RXD线路是空闲状态
- 设置USART#.BRG为0
- 置1 USART#.CTL.bit16(AUTOBAUD)
- 发送方尽快发送一个奇数过来
  - 如果不及时会超时退出，并设置错误标志USART#.STAT.bit16(ABERR)
- 成功接收后，USART自动把最接近的设置写到BRG寄存器中，自动清除AUTOBAUD位；收到的数据也写到RXDAT寄存器，供用户核实。



# USART低功耗模式唤醒

# 低功耗模式唤醒-1

- 睡眠模式下，任何触发USART中断的信号都可以唤醒芯片，相关配置如下：

第一步

- 配置USART为异步模式或同步模式

第二步

- 使能NVIC寄存器中的USART中断

第三步

- 使能USART中断使能寄存器INTENSET中的中断

## 低功耗模式唤醒-2

- 深度睡眠/掉电模式下，只能支持USART同步从机模式的唤醒（因为USART 时钟被关闭了），相关配置如下：

### 第一步

- 配置USART为同步从机模式（注：USART的SCLK时钟信号必须连到一个引脚并外接到一个主设备）

### 第二步

- 使能STARTERP1寄存器中的USART唤醒中断

### 第三步

- 使能NVIC寄存器中的USART中断

### 第四步

- 在PDAWAKE寄存器中，配置所有唤醒后需要正常工作的外设模块

### 第五步

- 使能USART中断使能寄存器INTENSET中的中断作为唤醒事件（比如：接收到一个起始位，接收buffer已经接收到一个字节等）



SECURE CONNECTIONS  
FOR A SMARTER WORLD