FT62F08X

Application note

目录

FT62F08X UART 应用

1. USART 接口

1.1. 功能特性

- 同步模式
  - 产生同步时钟输出
- 多芯片通信模式
  - 哑模式唤醒之后，才可以接收数据
  - 可以通过地址匹配和 IDLE 帧唤醒呀模式
- 异步模式
  - 可编程的 7,8,9 比特数据模式
  - 支持 1,2/1.5 bit 停止位
  - 支持红外 1.0 模式
  - 单线半双工
  - 发送接收使能控制
  - 16bit 波特率设置
  - RXNE 中断，TXE 中断，IDLE 帧中断，break 帧中断，奇偶校验错误，overrun 中断，发送完成中断
- 智能卡模式
  - 1.5 bit 停止位
  - 时钟输出
  - guard time
- LIN 主机模式
  - 支持断开帧的发送与检测
- 自动波特率检测

1.2. 功能描述

1.2.1. 一般描述

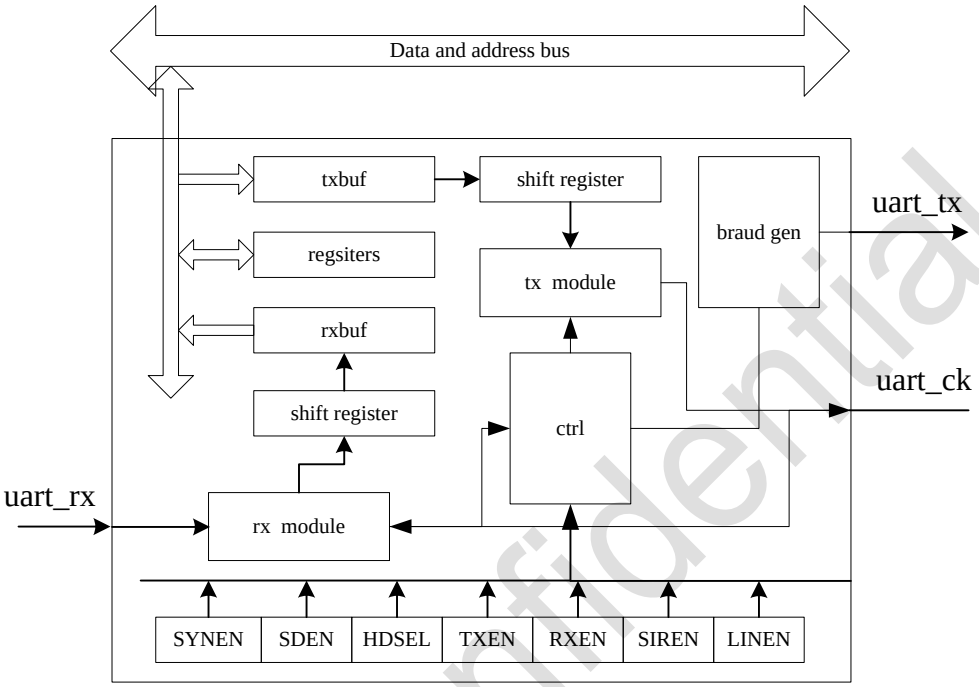

图 15.1 USART 原理框图

串口模块总共有三只引脚：

1. uart_rx：用作串口数据的输入引脚
2. uart_tx：用作串口数据的输出引脚，在用作半双工模式时也用作串行数据的输入引脚
3. uart_ck：在同步模式时用作同步时钟输出，在智能卡模式时用系统作分频时钟输出

该模块支持同步模式，异步模式，半双工模式，LIN Master 模式，红外模式和智能卡模式，默认的状态是工作于异步全双工模式，在确定使用一种模式后，请确保其他模式的相关使能位已关闭。

1.2.2. 异步工作模式

异步工作模式的串口采用异步的方式进行通信，配置的步骤如下：

1. 配置 DLH/DLL 产生相应的波特率进行通信，DLH 和 DLL 共同组成 16 位的波特率分频器，通信的波特率=fmaster/(16*(DL*))，其中 fmaster 为系统时钟，16 位的波特率分频器的值最小位 1，DL*表示的是 DLL 和 DLH 的组合，设置为零时串口不工作；
2. 配置 LCR 寄存器中的 LTH 位和 LCREXT 寄存器中的 EXTEN 来设置通信的数据长度，配置 LCR 寄存器中的 STOP 位来配置停止位的长度，配置 LCR 寄存器中的 PEN 和 EVEN 来配置奇偶校验位配置 IER 寄存器中的中断使能位来允许中断；
3. 配置 MCR 寄存器中的 TXEN 和 RXEN 来使能允许发送和接收；

异步模式通信的数据格式是先发送低位数据位，最后发送高比特位，如下图所示的 8 比特数据格式不带奇偶校验和带奇偶校验的帧格式。

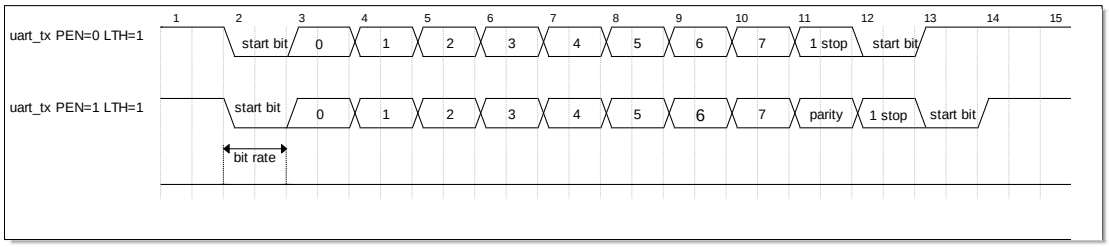

图 15.2 异步模式时序图

异步模式的数据处理流程包括阻塞模式处理和非阻塞模式处理，大致的处理流程如下：

1. 配置完波特率和相关控制位以后，发送端可以向 DATAL/H 发送 buf 寄存器写入数据，在阻塞模式下可以查询 TXEF 标志位，如果查询到 TXEF 为 1，则可以继续向 DATAL/H 写入需要发送的数据；在非阻塞模式下，使能发送为空中断，则在 TXEF 为 1 时，就会自动进入中断，向 DATAL/H 写入数据就可清除 TXEF 标志位，当在向 txbuf 写入最后一个要发送的数据时，禁用发送为空中断；
2. 接收端在阻塞模式下可以查询 RXNEF 标志位，在查询到该标志位为 1 时，表示接收到了数据，通过读取 DATAL/H 来清零 RXNEF 标志位；采用非阻塞模式接收数据时，需要使能 RXNE 中断，在串口接收到数据后，直接进入中断，读取 rxbuf 后清零 RXNEF 标志位；在采用非阻塞模式接收数据时，建议打开 RXSE 中断使能，在接收数据的过程中如遇到接收错误就会直接进入中断进行相关的处理；
3. 在串口发送数据的时候也可以使用 TCF 标志位来处理，在 TCF 标志位为 1 时，表示当前的数据发送已经完成，可以向 txbuf 写入下一个要发送的数据，这时 TCF 标志位会自动清零；

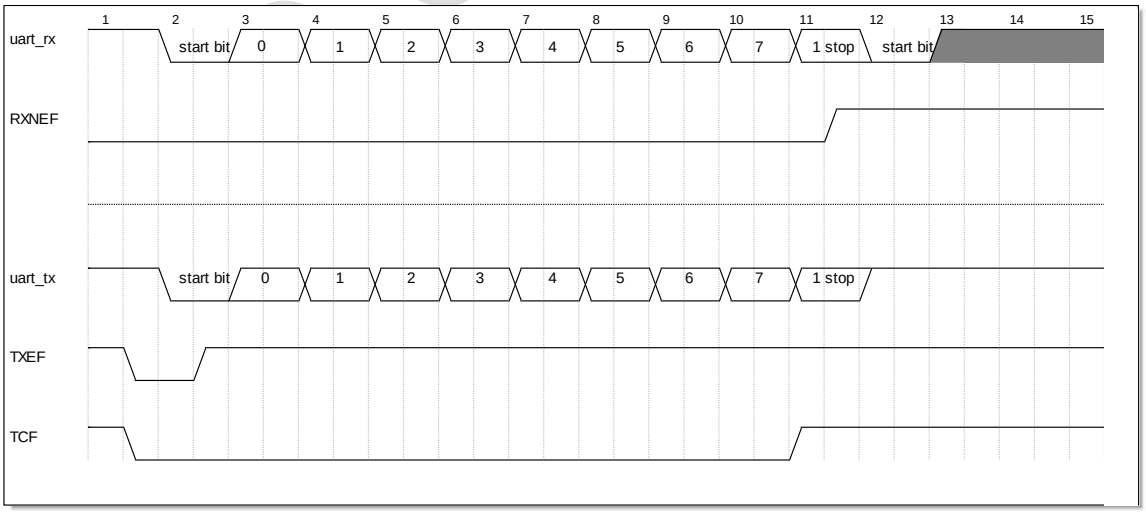

图 15.3 异步模式标志位时序图

1.2.3. 同步工作模式

同步工作模式用于串口模拟 SPI 通信的功能，在串口数据输出的同时，输出一个与数据相关的同步时钟，同步时钟的极性和相位可以通过 URSYNCR 寄存器中的 CPOL 和 CPHA 来配置；URSYNCR 寄存器中的 LBCL 控制的是最后一比特数据的时钟是否输出，该位为零时，只输出数据长度减一个有效同步时钟，最后一个有效时钟不会输出；SYNEN 就是同步时钟输出使能位，在该位为 1 时相应的 IO 会用作同步时钟输出；该模块只能模拟 SPI 主机模式，数据输出是先发送低位数据，然后发送的高位数据，并且时钟引脚只能输出同步时钟，并不能用作时钟输入；

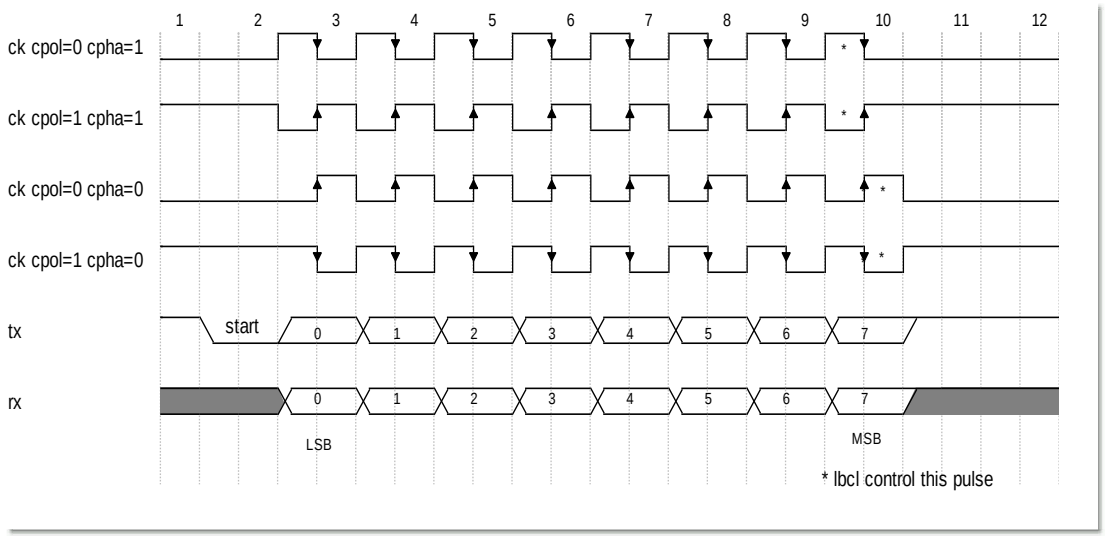

图 15.4 同步模式时序图

如图所示八比特数据格式的同步时钟输出，同步模式中如果没有使能 TXEN 也会产生同步脉冲输出，这时候同步模式只用于接收数据，写入到 DATAL/H 寄存器中的数据会发送到内部的移位寄存器中，用于产生同步脉冲输出，tx 引脚的值一直保持为 1，在同步发送的同时如果使能了 RXEN 接收位，则可以同步接收数据。

1.2.4. 半双工模式

半双工模式属于异步工作方式的一种，只是在通信时只用到了 tx 引脚，tx 引脚 IO 的应该配置成开漏模式，发送与接收的处理由软件控制 RXEN 和 TXEN 来实现；需要注意的是如果在发送过程中同时使能了接收，则发送的数据也会被本机接收到；置位 HDSEL 即可启用半双工模式。

1.2.5. 红外工作模式

红外模式用于红外通信，置位 SIREN 位可以使能红外模式，同时 LTH 位置位为 1，启用八比特数据格式；通信的波特率设置跟异步串口的配置方法相同。

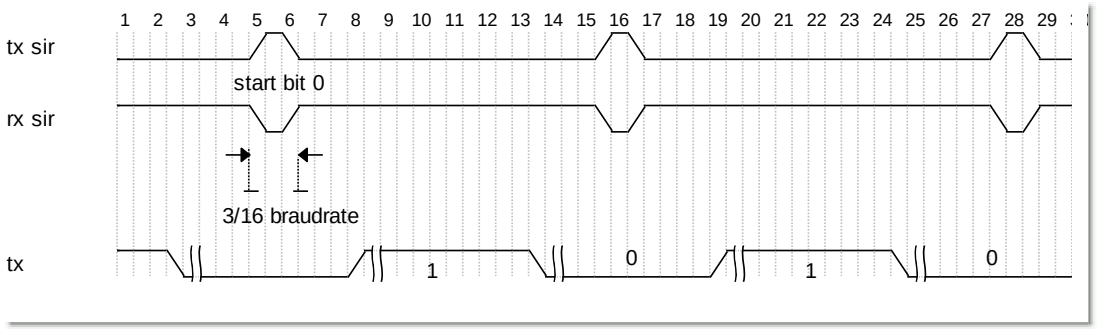

图 15.5 红外模式时序图

如图所示红外模块发送的脉冲宽度是比特周期的 3/16，当发送的数据为零时会产生一个高脉冲；接收时的低脉冲会被解释成零；接收与发送的总线极性是相反的，发送空闲时总线保持低电平，接收空闲时总线保持为高电平。

红外模式可以工作在低功耗模式，红外模式通常工作在系统的时钟频率下，红外的通信波特率=fmaster/(16*DL*)；当使能了 SIRLP 以后，红外的通信波特率= fmaster/(PSC*16*DL*)；这里的 DL*表示 DLL 和 DLH 的组合，在 psc 设置为 0 或 1 时，psc 分频模块无效，波特产生模块直接使用 Fmaster，如下图所示。

图 15.6 红外低功耗模式原理框图

1.2.6. 智能卡模式

智能卡模式属于半双工模式，支持 ISO7816-3 标准，置位 SDEN 来启用智能卡模式，除此之外根据协议要求需要使能 1.5 比特停止位 STOP 和奇偶校验位 PEN，同时需要配置相应的 IO 为开漏模式。

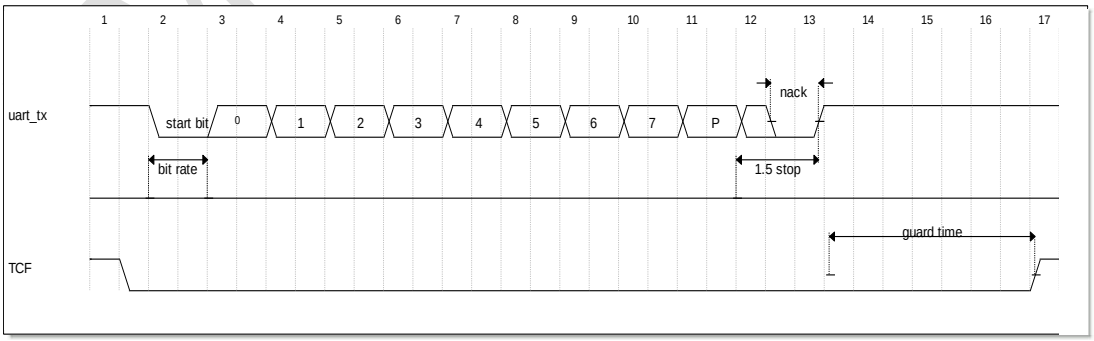

图 15.7 智能卡模式时序图

在使能了 NACK 位以后，接收方在检测到奇偶校验出错以后，会在 0.5 个停止位之后拉低总线一个比特周期，同时发送方会在停止位处检测总线是否被拉低，若检测到总线被拉低，则会产生帧错误标志 FEF，发送方根据要求可以选择重发当前的数据，发送次数由用户决定。在没有使能 NACK 位时，接收方在检测到奇偶校验错误时，不会拉低总线而是会产生一个奇偶检验

错误标志位 PEF。

智能卡模式的发送方发送完成数据后，TCF 标志位会在经过 GT 个波特周期后置位；发送与接收的处理由软件控制 TXEN 和 RXEN 来处理。

智能卡模式中，可以通过使能 CKOE 来输出一个时钟供给使能卡使用，输出的时钟频率详见 URSDCR2 寄存器说明；**需要注意的是在置位 CKOE 后，请配置 PSC 的值为有效值，否则不应该置位 CKOE。**

1.2.7. LIN Master 模式

串口模块支持 LIN Master 模式，使能 LINEN 后进入 LIN Master 模式，在发送断开帧之前请先配置一下断开帧的长度 BLTH；如图所示，在置位 BKREQ 后，tx 引脚会发送 BLTH 个连续的低电平，发送完成后自动清零，在使能该位后，可以查询 BKREQ 的状态，等到 BKREQ 为 0 时则表示断开帧发送完成；在发送断开帧的过程中请勿手动清零 BKREQ。

接收端在接收到大于起始位+数据长度+停止位个数的连续低电平以后，会被认为接收到了断开帧，BKF 会被置 1。

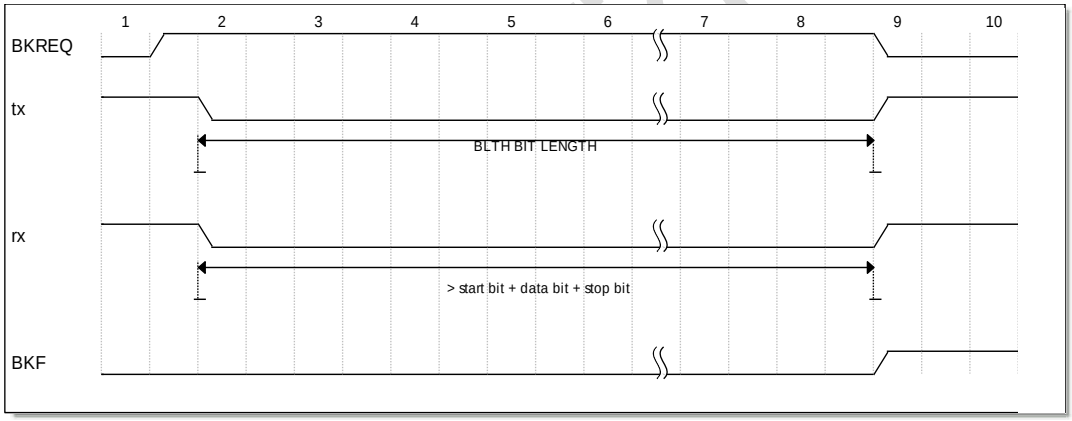

图 15.8 LIN Master 模式时序图

需要注意的是断开帧的接收与发送并不局限于 LIN mode，其他异步模式，红外模式等也是可以应用。

1.2.8. 多芯片通信模式

多处理器通信用于一个芯片用作主机模式，其他芯片用作从机模式，从机的发送引脚通过逻辑与的方式连接到主机的 RX 引脚，这种模式中主机希望接收特定的消息，只有在特定的条件触发后才会接收数据。

置位 RWU 后即可进入哑模式，屏蔽一切接收，根据 WAKE 的配置，可以唤醒接收主机接收数据：

1. WAKE 置零，在接收到起始位+数据位+停止位总个数波特周期后唤醒；
2. WAKE 置一，在接收到匹配的地址后唤醒；

- 地址空闲唤醒，在置位 RWU 后，如果总线数据一致繁忙，则不唤醒接收，在检测到连续的一帧空闲时间（起始位+数据位+停止位）后唤醒开始接收数据。

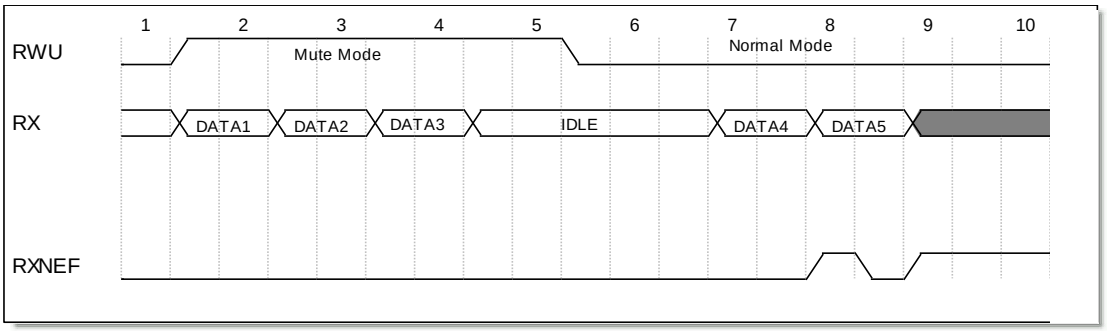

图 15.9 哑模式空闲唤醒时序图

- 地址匹配唤醒，在置位 RWU 后，每次接收到数据后都会判断数据的高位是否为 1，若为 1 则把数据的低四位与 URRAR 的值进行比较，若相等则退出哑模式开始接收之后的数据，后续如果再次接收到地址数据（该模式下数据的高位为 1 则表示接收到的数据为地址数据），则还是会与本机的地址 URRAR 进行比较，若不同立即进入哑模式；该模式下每次匹配到地址后 ADDRF 会被置 1，反之没有匹配到地址时一直为零。

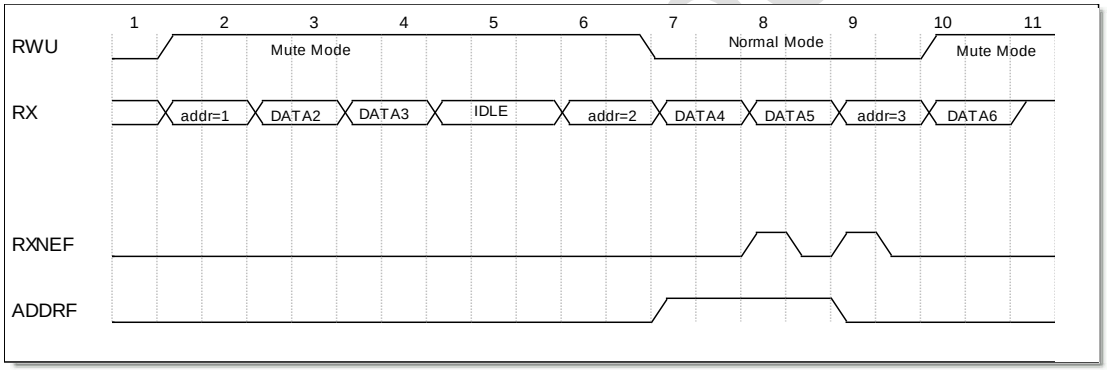

图 15.10 哑模式地址匹配唤醒时序图

1.2.9. 自动波特率检测

自动波特率检测功能用于接收端校准通信波特率，保持与发送端波特率相同，串口模块实现的波特率检测模块有两种模式：

1. 检测起始位的长度（model0）；这种模式要求数据的第一比特为一，例如数据 0x03、0x55 等；
2. 检测起始比特和第一比特的长度（model1）；这种模式要求第一比特的数据为 1，第二比特的数据为 0，例如数据 0x55，0x01 等；

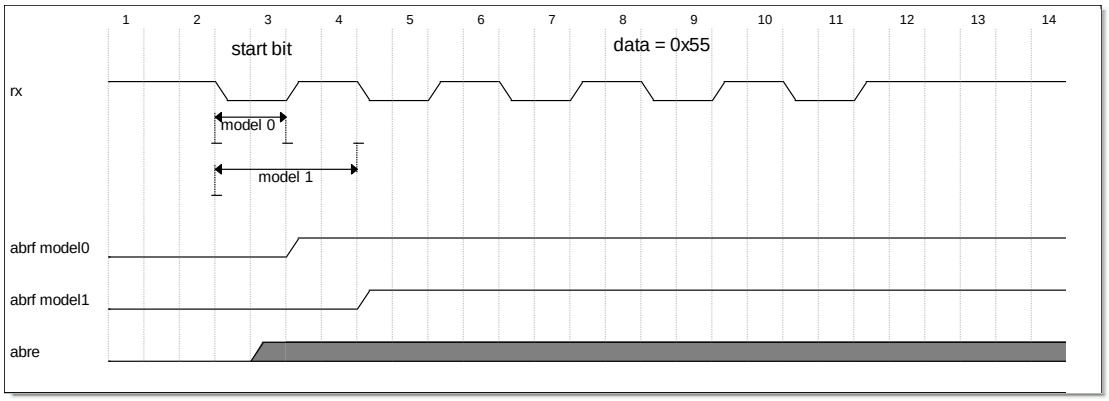

图 15.11 自动波特率检测时序图

使用波特率检测功能，首先使能 ABREN，然后根据自己使用的检测模式配置 ABRM，读取 ABRF 是否为 1（上次使用过后未清零），如果为 1，则写零清零；然后开始接收数据，波特率检测完成后 ABRF 就会置 1，在 ABRF 置 1 后，不要立即清零 ABRF，因为清理 ABRF 会立即在当前传输的位置（可能已经不是起始比特的位置）进行波特率检测，这样会导致错误的结果；当前数据接收完成后会产生 RXNEF 标志位，然后可以清零 ABRF，开始下一次波特率的检测，假如不清零 ABRF 标志位，则下次接收数据时不会启动波特率检测；如果波特率检测超出了范围则会产生 ABRE 标志位 ，表示波特率检测出错。

波特率检测完成后，如果后续还需要检测波特率则不需要立即清零 ABRF，只有在需要再次检测的时候清零 ABRF 即可。

需要注意的是波特率检测的数据是用来配置 DLL/DLH 寄存器用的，如果发送方波特率数据不靠近 Fbraudrate=Fmaster/(16*{DLH,DLL})，则波特率检测模块会自动配置本机为靠近支持的波特率，串口模块并不支持小数波特率，因此该模块的波特率检测存在误差。

1.3. 寄存器汇总

名称	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
URDATAL	0x48C	DATAL[7:0]								0000 0000
URDATAH	0x48D	—							DATAH	---- ---0
URIER	0x48E	—		TCEN	—	IDELE	RXSE	URTE	URRXNE	--0- 0010
URLCR	0x48F	—	BKREQ	—	EVEN	PEN	STOP	—	LTH	0000 0000
URLCREXT	0x490	—						RWU	EXTEN	---- --00
URMCR	0x491	—		SIRLP	TXEN	RXEN	WAKE	HDSEL	SIREN	---0 0000
URLSR	0x492	ADDRF	IDLEF	TXEF	BKF	FEF	PEF	OVERF	RXNEF	0000 0000
URRAR	0x493	RAR[3:0]								0000 0000
URDLL	0x494	DLL[7:0]								0000 0000
URDLH	0x495	DLH[7:0]								0000 0000
URABCR	0x496	—				ABRE	ABRM	ABRF	ABREN	---- 0000
URSYNCR	0x497	—				LBCL	CPHA	CPOL	SYNEN	---- -000
URLINCR	0x498	—			LINEN	BLTH[3:0]				---0 0000
URSDCR0	0x499	—	NACK	CKOE	SDEN	—				-000 ----

URSDCR1	0x49A	GT[7:0]		0000 0000
URSDCR2	0x49B	PSC[7:0]		0000 ----
URTC	0x49C	—	TCF	---- ---1

1.3.1. URDATAL 寄存器，地址 0x48C

Bit	7:0
Name	DATAL
Reset	0x00
Type	RW

Bit	Name	Function
7:0	DATAL	数据发送/接收寄存器低八位，请使用字节传输指令对该寄存器进行操作

1.3.2. URDATAH 寄存器，地址 0x48D

Bit	7:1	0
Name	—	DATAH
Reset	—	0x0
Type	RO-0	RW

Bit	Name	Function
7:1	N/A	保留位，读 0
0	DATAH	数据发送/接受寄存器高八位，这一位只有启用 9 比特数据格式才有用，这种模式下，一定要先写低八位再写高 1 位。 1：表示 DATAL 是地址 0：表示 DATAL 是数据

1.3.3. URIER 寄存器，地址 0x48E

Bit	7:6	5	4	3	2	1	0
Name	—	TCIE	—	IDELE	RXSE	URTE	URRXNE
Reset	—	0x0	—	0x0	0x0	0x0	0x0
Type	RO-0	RW	RO-0	RW	RW	RW	RW

Bit	Name	Function
7:6	N/A	保留位，读 0
5	TCEN	发送完成中断使能 1：使能发送完成中断

		0：禁用发送完成中断
4	N/A	-保留位，读 0
3	IDELE	空闲帧中断使能 1：使能空闲帧中断 0：禁用空闲帧中断
2	RXSE	接收状态使能，包括帧断开，帧错误，奇偶校验错误，接收溢出错误 1：使能接收到断开帧/帧错误/奇偶校验错误/接收溢出错误状态中断 0：禁用状态信息中断
1	URTE	发送 buf 为空中断使能 1：使能发送为空中断 0：禁用发送为空中断
0	URRXNE	接受到数据中断使能 1：使能接收到数据中断 2：禁用接收到数据中断

1.3.4. URLCR 寄存器，地址 0x48F

Bit	7	6	5	4	3	2	1	0
Name	—	BKREQ	—	EVEN	PEN	STOP	—	LTH
Reset	—	0x0	—	0x0	0x0	0x0	—	0x0
Type	RO-0	RW	RO-0	RW	RW	RW	RO-0	RW

Bit	Name	Function
7	N/A	保留位，读 0
6	BKREQ	发送断开帧使能，发送完成后改位自动清零，断开帧发送过程中不应该写零该位；发送断开帧之前请先设置断开帧的长度，在 lincr 寄存器的 lth 字段 1：请求发送断开帧/断开帧发送中 0：未请求发送断开帧/断开帧发送完成
5	N/A	保留位，读 0
4	EVEN	置 1 表示偶检验使能，置零表示奇校验使能 1：表示使用偶检验 0：表示使用奇校验
3	PEN	校验位使能 1：使能校验位 0：禁用校验位
2	STOP	停止位长度设定 0：表示 1 个停止位 1：智能卡模式启用时表示 1.5 个停止位，否则表示两个停止位
1	N/A	保留位，读 0
0	LTH	通信数据长度设定 0：表示数据长度是 7 位，此位不包含检验位长度 1：表示数据长度是 8 位，此位不包含检验位长度

1.3.5. URLCREXT 寄存器，地址 0x490

Bit	7:2	1	0
Name	—	RWU	EXTEN
Reset	—	0x0	0x0
Type	RO-0	RW	RW

Bit	Name	Function
7:2	N/A	保留位，读 0
1	RWU	多处理器模式接收唤醒 1：设置进入哑模式 0：未设置进入哑模式或者已经退出哑模式
0	EXTEN	发送的数据是否时 9 比特长度 1：发送的数据长度是 9 比特长度 0：发送的数据不是 9 比特长度

1.3.6. URMCR 寄存器，地址 0x491

Bit	7:6	5	4	3	2	1	0
Name	—	SIRLP	TXEN	RXEN	WAKE	HDSEL	SIREN
Reset	—	0x0	0x0	0x0	0x0	0x0	0x0
Type	RO-0	RW	RW	RW	RW	RW	RW

Bit	Name	Function
7:6	N/A	保留位，读 0
5	SIRLP	红外低功耗模式使能 1：使能红外低功耗模式 0：禁用红外低功耗模式
4	TXEN	发送使能 1：使能接口的发送，相应的 IO 会被用作 TX 引脚 0：禁用接口的发送
3	RXEN	接收使能 1：允许接口接收，相应的 IO 会被用作 RX 引脚 0：禁用接口接收
2	WAKE	哑模式唤醒方式选择 1：选择地址匹配 0：选择 IDLE 帧
1	HDSEL	半双工使能 1：使能半双工模式

		0：禁用半双工模式
0	SIREN	红外模式使能 1：使能红外模式 0：禁用红外模式

1.3.7. URLSR 寄存器，地址 0x492

Bit	7	6	5	4	3	2	1	0
Name	ADDRF	IDLEF	TXEF	BKF	FEF	PEF	OVERF	RXNEF
Reset	0x0	0x0	0x1	0x0	0x0	0x0	0x0	0x0
Type	RO	W0	RO	W0	W0	W0	W0	RO

Bit	Name	Function
7	ADDRF	哑模式地址匹配标志位，指示位，软件不可清 1：哑模式地址匹配唤醒模式中，匹配到了地址 0：哑模式地址匹配唤醒模式中，未匹配到地址
6	IDLEF	空闲帧标志，写 0 清 0，写 1 无效 1：检测到空闲帧 0：未检测到空闲帧
5	TXEF	发送寄存器的状态，在启用了 9 比特数据格式时，写 DATAH 寄存器清零，否则写取 DATAL 寄存器清零 1：发送寄存器为空 0：发送寄存器不为空
4	BKF	断开帧标志，写 0 清 0，写 1 无效 1：接收到了断开帧 0：未接收到断开帧或已清零
3	FEF	帧错误标志，写 0 清 0，写 1 无效 1：接收出现了帧错误 0：未接收到帧错误会已清零
2	PEF	奇偶校验错误标识，写 0 清 0，写 1 无效零 1：接收到了奇偶校验错误 0：未接收到奇偶校验错误或已清零
1	OVERF	溢出错误标志，写 0 清 0，写 1 无效 1：接收寄存器出现溢出 0：接收未出现溢出或已清零
0	RXNEF	在启用 9 比特数据格式时，读取 DATAH 寄存器清零，否则读取 DATAL 寄存器清零 1：接收寄存器非空 0：接收寄存器为空或已清零

1.3.8. URRAR 寄存器，地址 0x493

Bit	7:4	3:0
Name	—	Receive address
Reset	—	0x0
Type	RO-0	RW

Bit	Name	Function
7:4	N/A	保留位，读 0
3:0	Receive address	多处理器模式中的本机地址

1.3.9. URDLL 寄存器，地址 0x494

Bit	7:0
Name	DLL
Reset	0x0
Type	RW

Bit	Name	Function
7:0	DLL	波特率分频计数器低八位

1.3.10. URDLH 寄存器，地址 0x495

Bit	7:0
Name	DLH
Reset	0x0
Type	RW

Bit	Name	Function
7:0	DLH	波特率分频计数器高八位 波特率=Fmaster/(16*{DLH,DLL})，默认值{DLH，DLL}为 0x0000 时，串口不工作； {DLH，DLL}最小值为 0x0001

1.3.11. URABCR 寄存器，地址 0x496

Bit	ABCR	3	2	1	0
Name	—	ABRE	ABRM	ABRF	ABREN

Reset	—	0x0	0x0	0x0	0x0
Type	RO-0	RW	RW	RW	RW

Bit	Name	Function
7:4	N/A	保留位，读 0
3	ABRE	波特率检测溢出 1：波特率检测超出范围 0：波特率检测未超出范围
2	ABRM	波特率检测模式 0：只检测起始位长度；这种模式要求第一比特为 1 1：检测起始位和数据第一位的长度，然后除以 2；这种模式要求第一比特为 1，第二比特为零
1	ABRF	检测到波特率标识，写零清零；该位在清零后，会立即再次进入波特率检测，为了保证每次检测的都是起始位，可以在 RXNEF 置位后，再清零 1：检测到波特率 0：未检测到波特率
0	ABREN	自动波特率检测使能 1：使能波特率检测功能 0：禁用波特率检测功能

1.3.12. URSYNCR 寄存器，地址 0x497

Bit	7:4	3	2	1	0
Name	—	LBCL	CPHA	CPOL	SYNEN
Reset	—	0x0	0x0	0x0	0x0
Type	RO-0	RW	RW	RW	RW

Bit	Name	Function
7:4	N/A	保留位，读 0
3	LBCL	同步模式最后一个比特时钟使能 1：同步模式中最后一比特的时钟输出 0：同步模式中最后一比特的时钟不输出
2	CPHA	同步模式时钟相位设置 1：主机在时钟变化的第二个沿开始采样第一个数据 0：主机在时钟变化的第一个沿开始采样第一个数据
1	CPOL	同步模式时钟极性设置 1：同步模式中时钟空闲时为高电平 0：同步模式中时钟空闲时为低电平
0	SYNEN	同步模式使能 1：使能同步传输模式，相应的 IO 会用作同步时钟输出 0：禁用时钟同步模式

1.3.13. URLINCR 寄存器，地址 0x498

Bit	7:5	4	3:0
Name	—	LINEN	BLTH
Reset	—	0x0	0x0
Type	RO-0	RW	RW

Bit	Name	Function
7:5	N/A	保留位，读 0
4	LINEN	Lin 模式使能 1：使能 LIN Master 模式 0：禁用 LIN Master 模式
3:0	BLTH	断开帧比特长度，大于零有效，一般设置为 12/13 比特长度，设置太短会被认为接收到的是正常的帧

1.3.14. URSDCR0 寄存器，地址 0x499

Bit	7	6	5	4	3:0
Name	—	NACK	CKOE	SDEN	—
Reset	—	0x0	0x0	0x0	—
Type	RO-0	RW	RW	RW	RO-0

Bit	Name	Function
6	NACK	智能卡回复 Nack 使能 1：使能检测到奇偶校验出错时，发送 NACK 0：检测到奇偶校验位错误时不发送 NACK
5	CKOE	给智能卡提供时钟时的使能 1：使能时钟输出，请配置 PSC 寄存器的值为有效值 0：禁止时钟输出
4	SDEN	智能卡模式使能，启用智能卡模式必须设定停止位为 1.5 位 1：使能智能卡模式 0：禁用智能卡模式
3:0	N/A	保留位，读 0

1.3.15. URSDCR1 寄存器，地址 0x49A

Bit	7:0

Name	GT
Reset	0x0
Type	RW

Bit	Name	Function
7:0	GT	智能卡模式:两字符之间波特间隔，最小为 1，即使设置为 0，也有一个间隔

1.3.16. URSDCR2 寄存器，地址 0x49B

Bit	7:0
Name	PSC
Reset	0x0
Type	RW

Bit	Name	Function
7:0	PSC	-给智能卡提供时钟时的时钟分频系数 0：无效 1：2 分频 2：3 分频 3：4 分频 …… -红外低功耗模式系统时钟分频 0:无效 1：1 分频 2：2 分频 3：3 分频 ……

1.3.17. URTC 寄存器，地址 0x49C

Bit	7:1	0
Name	—	TCF
Reset	—	0x1
Type	RO-0	RW

Bit	Name	Function
7:1	N/A	保留位，读 0
0	TCF	发送完成状态标志 1：数据发送完成 0：数据发送未完成，写 1 清零或写 DATAL/DATAH 寄存器后清零（在使能了 9 比特

		数据格式时，写 DATAH 寄存器后清零，否则写 DATAL 寄存器后清零）

2 应用范例

```
//************************************************************
/*   文件名: TEST_62F08x_UART.c
*    功能：    FT62F08x-UART 功能演示
*    IC:      FT62F088 LQFP32
*    内部：    16M
*    empno: 500
*    说明：   串口上电发送 10 个字符，然后等待接收 10 个字节数据（通过串口助手发送接收）
*
*
*
*
*    参考原理图  TEST_62F08x_sch.pdf
*/
//************************************************************
#include "SYSCFG.h"
//************************************************************
//************************宏定义*********************************
#define   unchar        unsigned char
#define   unint         unsigned int
#define   unlong        unsigned long

#define   DemoPortOut RB3
#define   DemoPortIn          RC3

//volatile    unchar mydata;                //全局查看变量定义

volatile   unchar receivedata[10] =0;
volatile   unchar senddata     =0;

volatile   unchar      toSend[11]={0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xaa};
unchar      i=0;
unchar      mmm=0;
/*-------------------------------------------------
 *   函数名：中断
 *   功能：
 *   输入：    无
 *   输出：    无
 -------------------------------------------------*/
```

```
void interrupt ISR(void)
{
    //中断处理************************
    if(URRXNE && RXNEF)          //接收中断
    {
        receivedata[mmm++] =URDATAL;

        if(mmm>=10)
        {
            mmm=0;
        }
        NOP();
    }
    //----------------------------
    if(TCEN && TCF)              //发送中断
    {
        TCF=1;          //写 1 清 0

        if(i<10)
        {
            URDATAL =toSend[i++];
        }
        else
        {
           i=0;
        }
        NOP();
     }
}
/*---------------------------------------------
 *   函数名：POWER_INITIAL
 *   功能：    上电系统初始化
 *   输入：    无
 *   输出：    无
 ---------------------------------------------*/
void POWER_INITIAL (void)
{

    OSCCON = 0B01110001;    //WDT 32KHZ IRCF=111=16MHZ
                            //Bit0=1,系统时钟为内部振荡器
                            //Bit0=0,时钟源由 FOSC<2：0>决定即编译选项时选择
    INTCON = 0;             //暂禁止所有中断

    PORTA = 0B00000000;
```

```
    TRISA = 0B10111111;         //PA 输入输出  0-输出  1-输入
    PORTB = 0B00000000;
    TRISB = 0B11110111;         //PB 输入输出  0-输出  1-输入
    PORTC = 0B00000000;
    TRISC = 0B11111111;         //PC 输入输出  0-输出  1-输入
    PORTD = 0B00000000;
    TRISD = 0B11111111;         //PD 输入输出  0-输出  1-输入

    WPUA = 0B00000000;          //PA 端口上拉控制  1-开上拉  0-关上拉
    WPUB = 0B00000000;          //PB 端口上拉控制  1-开上拉  0-关上拉
    WPUC = 0B00001000;          //PC 端口上拉控制  1-开上拉  0-关上拉
    WPUD = 0B00000000;          //PD 端口上拉控制  1-开上拉  0-关上拉

    WPDA = 0B00000000;          //PA 端口上拉控制  1-开下拉  0-关下拉
    WPDB = 0B00000000;          //PB 端口上拉控制  1-开下拉  0-关下拉
    WPDC = 0B00000000;          //PC 端口上拉控制  1-开下拉  0-关下拉
    WPDD = 0B00000000;          //PD 端口上拉控制  1-开下拉  0-关下拉

    PSRC0   = 0B11111111;       //PORTA,PORTB 源电流设置最大
    //BIT7~BIT6:PORTB[7:4]源电流能力控制,BIT5~BIT4:PORTB[3:0]源电流能力控制
    //BIT3~BIT2:PORTA[7:4]源电流能力控制,BIT1~BIT0:PORTA[3:0]源电流能力控制

    PSRC1   = 0B11111111;       //PORTC,PORTD 源电流设置最大
    //BIT7~BIT6:PORTD[7:4]源电流能力控制,BIT5~BIT4:PORTD[3:0]源电流能力控制
    //BIT3~BIT2:PORTC[7:4]源电流能力控制,BIT1~BIT0:PORTC[3:0]源电流能力控制

    PSINK0 = 0B11111111;        //PORTA 灌电流设置最大  0:最小，1:最大
    PSINK1 = 0B11111111;        //PORTB 灌电流设置最大  0:最小，1:最大
    PSINK2 = 0B11111111; //PORTC 灌电流设置最大  0:最小，1:最大
    PSINK3 = 0B11111111; //PORTD 灌电流设置最大  0:最小，1:最大

    ANSELA = 0B00000000;      //全为数字管脚

}
/*---------------------------------------------
 *   函数名称：DelayUs
 *   功能：      短延时函数  --16M-2T--大概快 1%左右.
 *   输入参数：Time 延时时间长度  延时时长 Time Us
 *   返回参数：无
 ---------------------------------------------*/
void DelayUs(unsigned char Time)
{
    unsigned char a;
    for(a=0;a<Time;a++)
```

```
    {
        NOP();
    }
}
/*---------------------------------------------
 *  函数名称：DelayMs
 *  功能：    短延时函数
 *  输入参数：Time 延时时间长度  延时时长 Time ms
 *  返回参数：无
 ---------------------------------------------*/
void DelayMs(unsigned char Time)
{
    unsigned char a,b;
    for(a=0;a<Time;a++)
    {
        for(b=0;b<5;b++)
        {
            DelayUs(197); //快 1%
        }
    }
}
/*---------------------------------------------
 *  函数名称：DelayS
 *  功能：    短延时函数
 *  输入参数：Time 延时时间长度  延时时长 Time S
 *  返回参数：无
 ---------------------------------------------*/
void DelayS(unsigned char Time)
{
    unsigned char a,b;
    for(a=0;a<Time;a++)
    {
        for(b=0;b<10;b++)
        {
            DelayMs(100);
        }
    }
}
/*---------------------------------------------
 *  函数名: UART_INITIAL
 *  功能：  主函数
 *  输入：  无
 *  输出：  无
 ---------------------------------------------*/
```

```
void UART_INITIAL(void)
{
    PCKEN |=0B00100000; //打开 UART 时钟

    URIER =0B00100001;   //使能发送接收完成中断
    URLCR =0B00000001;   //8 位数据，停止位 1，无奇偶校验
    URMCR =0B00011000;

    URDLL =104;              //9600 波特率  = Fosc/16*[URDLH:URDLL]
    URDLH =0;
    TCF=1;
    INTCON=0B11000000;

    //TCF:  发送完成标志
    //TXEF:1 发送寄存器为空
    //RXNEF:1 按收寄存器非空
}
/*-----------------------------------------------
 *   函数名: main
 *   功能：    主函数
 *   输入：    无
 *   输出：    无
 -----------------------------------------------*/
void main(void)
{
    POWER_INITIAL();          //系统初始化
    UART_INITIAL();
    DelayMs(100);

    if(TXEF)                  //上电发送 10+1 个数据
    {
       URDATAL =0xaa;
    }

    while(1)
    {
        NOP();
    }
}
```

Fremont Micro Devices (SZ) Limited

#5-8, 10/F, Changhong Building, Ke-Ji Nan 12 Road, Nanshan District, Shenzhen, Guangdong 518057
Tel: (86 755) 86117811
Fax: (86 755) 86117810

Fremont Micro Devices (Hong Kong) Limited

#16, 16/F, Blk B, Veristrong Industrial Centre, 34-36 Au Pui Wan Street, Fotan, Shatin, Hong Kong
Tel: (852) 27811186
Fax: (852) 27811144

Fremont Micro Devices (USA), Inc.

42982 Osgood Road Fremont, CA 94539
Tel: (1-510) 668-1321
Fax: (1-510) 226-9918

Web Site: http://www.fremontmicro.com/