

前言

在很多低功耗应用中需让 MCU 进入 STOP MODE 以节省功耗, 在以前的架构中, 若要实现这种应用, 往往会在进低功耗模式之前将串口的管脚设置成带 EXTI 中断模式的普通 GPIO。虽然这样也可以达到效果, 但是这样难免会丢失开头的数据。

STM32 家族里有些系列芯片已经集成了低功耗 UART, 即 LPUART, 这样就能将低功耗、数据通信、正常唤醒的完美结合。

我们可以使用新的 LPUART 把 MCU 从 STOP 下唤醒, 又可以不丢失通讯数据。当然基于 LPUART 唤醒 STOP 模式下的 MCU 的有一定使用限制。下面我们将详细介绍一下。

1. 使用不同时钟下 LPUART 进行唤醒

1.1 使用 HSI16 的 LPUART

使用 HSI16 作为 LPUART 的时钟, 这样波特率就可以较高, 不过使用 LPUART 来唤醒还是有个问题要注意, 即高速的波特率与 LPUART 的唤醒时间差的问题。我们以 [STM32L431](#) 作为例子, 根据其数据手册, 它的 LPUART 唤醒时间如下 :

Table 42. Wakeup time using USART/LPUART⁽¹⁾

Symbol	Parameter	Conditions	Typ	Max	Unit
$t_{WUUSART}$ $t_{WULPUART}$	Wakeup time needed to calculate the maximum USART/LPUART baudrate allowing to wakeup up from stop mode when USART/LPUART clock source is HSI	Stop mode 0	-	1.7	μs
		Stop mode 1/2	-	8.5	

1. Guaranteed by design.

下面结合某客户的实际案例探讨。他们反映 LPUART 在 576000 时唤醒会丢失字节. 我们来详细分析一下这个情况。

如果应用在 STOP MODE 1/2 下, 唤醒时间最大为 8.5us, 这个时间不能大于串口异步通信所能承受的最大时间偏差。毕竟, 串口异步通信时是不会针对这个唤醒时间做等待的。那么我们需要做的就是在这种情况下, 求得串口的最大安全波特率。

首先我们需要需要以下两个参数 :

$t_{WULPUART}$ (wakeup time from Stop mode), 这个可以从数据手册上查询. (如上表)

LPUART 接收的允许公差(如下表)

Table 165. Tolerance of the LPUART receiver when BRR [3:0] is different from 0000

M bits	$768 \leq \text{BRR} < 1024$	$1024 \leq \text{BRR} < 2048$	$2048 \leq \text{BRR} < 4096$	$4096 \leq \text{BRR}$
8 bits (M=00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M=01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M=10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M=00), 2 stop bit	2.08%	2.86%	4.35%	4.42%
9 bits (M=01), 2 stop bit	1.82%	2.56%	3.90%	4.42%
7 bits (M=10), 2 stop bit	2.34%	3.23%	4.92%	4.42%

下面我们以 8bit , 1STOP, $\text{BRR} \geq 4096$, STOP2 mode 作为例子:

首先我们可以通过上表” Table 165: Tolerance of the LPUART receiver when BRR[3:0] is different from 0000” 得出LPUART在这情况下的接收容差是4.42%.

容错公式为 : $\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{USART}$

DTRA: 预期发送的容错率(这个包含发射器本身振荡器的偏差)

DQUANT: 波特率接收的错误率

DREC: 接收晶体的偏差

DTCL: 发送的偏差率 (一般都是发送器采用不对称的上升沿和下降沿时序)

DWU : 是从stop mode 下唤醒后采样点的偏差而导致的错码率.

为了更容易计算, 我们简化一下公式, 假设DTRA, DQUANT, DREC和DTCL为0%, 所以DWU是4.42%, 为了更准确, 我还要考虑晶体的误差, 我们使用的HSI误差是1%, $t_{\text{WULPUART}} = 8.5\mu\text{s}$ (这里采用的是STOP2):

$\text{DREC} + \text{DWU} < \text{LPUART}$

$\Rightarrow 1\% + \text{DWU} < 4.42\%$

$\Rightarrow \text{DWU} < 3.42\%$

因为我们这里采用的是8bit , 1stop, 所以 :

$M[1:0] = 00$:

$\text{DWU} = t_{\text{WUUSART}} / (10 \times T_{\text{bit}})$

$T_{\text{bit min}} = 8.5\mu\text{s} / (10 \times 3.42\%)$

$T_{\text{bit min}} = 24.8\mu\text{s}$

所以在这个条件下异步串口允许的最大波特率是 $1/24.8\mu\text{s}$, 即要小于40.3K的波特率. 我们客户使用的是576000的波特率显然过高, 丢失首字节就不难理解了. 当修改为19200后, 问题就解决了.

1.2 使用 LSE 的 LPUART

如果使用 LSE 下的 LPUART, 这个相对简单了. 因为 LSE 只有 32.768Khz, LPUART 的波特率最大也只能到 9600, 速度下来了, 自然就没有那么多问题了.

小结: 时不时会有人问起类似问题, 其实关于该问题在 stm32 芯片的相关参考手册中都有描述. 总之, 在开发中遇到问题时首先看看芯片技术手册的相关部分总不会错, 有的可能是专门描述, 有的是可能是额外提醒【Note/Caution】等.

重要通知 – 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对ST 产品和/ 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST 产品的最新信息。ST 产品的销售依照订单确认时的相关ST 销售条款。

买方自行负责对ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST 产品如有不同于此处提供的信息的规定，将导致ST 针对该产品授予的任何保证失效。

ST 和ST 徽标是ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。