Data Sheet

ZLG9518S

多串口专用 IC

DS01010101 V1.00 Date: 2013/05/27 产品数据手册

概述

ZLG9518S 芯片是广州周立功单片机科技有限公司针对多串口应用而设计的一款专用 IC，也是一款将高速 SPI 转换为 8 路低速 UART 的接口转换芯片；ZLG9518S 芯片在串口数量上支持动态配置，最高可支持 8 个串口；串口与串口之间的资源不共享，每个串口在物理上都是绝对独立的；8 个串口都可动态配置，支持 8 种波特率、5 种校验方式、4 种数据长度、3 种停止位等；每个串口都支持可选的硬件流控功能，用户可以根据需要开启或者关闭硬件流控功能；除此之外，ZLG9518S 芯片还提供了丰富的寄存器，包括可选的中断功能、中断模式、错误状态查询等；SPI 时钟最高可达 33M，且 SPI 协议非常简单。ZLG9518S 芯片满足目前大部分多串口应用场合，大大缩短产品的研发周期，提高产品的可靠性和稳定性。

产品特性

- 支持 1~8 路串口可动态扩展；
- 8 路串口在物理上绝对独立；
- 串口的收、发缓存独立，高达 255bytes；
- 支持 8 种波特率，最高可达 115200bps；
- 支持 5 种校验方式；
- 支持 4 种数据长度；
- 支持 3 种停止位；
- 支持可选的硬件流控功能；
- 支持 RTS 和 CTS 单独使用；
- 支持多种流控触发点配置；
- 支持可选的中断功能、中断模式等；
- 支持多种中断触发点配置；
- 支持错误状态查询和错误中断等；
- SPI 时钟最高可达 33MHz；
- SPI 支持模式 3，协议简单易懂；
- 内设多个寄存器，且寄存器精简易用；
- 温度范围-40~85°C。

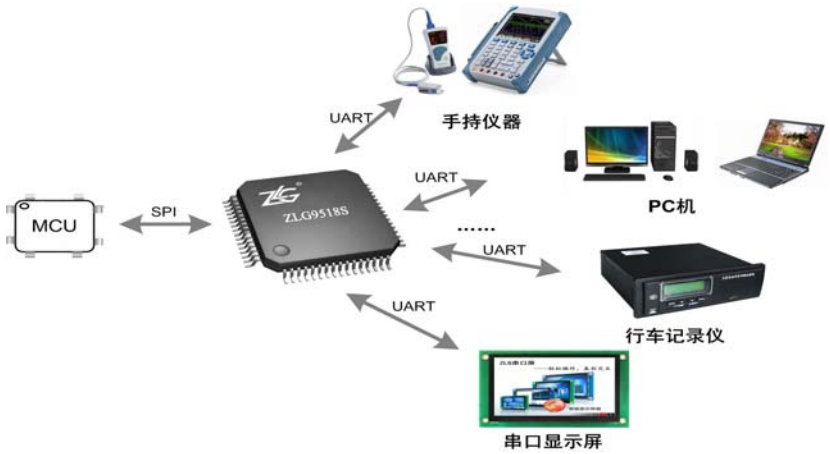
产品应用

- 集散控制系统；
- 数据采集系统；
- 行车记录仪；
- 串口屏。

订购信息

型号	温度范围	封装
ZLG9518S	-40°C ~ +85°C	TQ128

典型应用



修订历史

版本	日期	原因
V1.00	2013/05/27	创建文档

销售与服务网络（一）

广州周立功单片机科技有限公司

地址：广州市天河北路 689 号光大银行大厦 12 楼 F4
邮编：510630
电话：(020)38730916　38730917　38730972　38730976　38730977
传真：(020)38730925
网址：www.zlgmcu.com
新浪微博：ZLG-周立功（http://weibo.com/ligongzhou）

广州专卖店

地址：广州市天河区新赛格电子城 203-204 室
电话：(020)87578634　87569917
传真：(020)87578842

南京周立功

地址：南京市珠江路 280 号珠江大厦 1501 室
电话：(025) 68123901　68123902
传真：(025) 68123900

北京周立功

地址：北京市海淀区知春路 113 号银网中心 A 座 1207-1208 室 （中发电子市场斜对面）
电话：(010)62536178　62536179　82628073
传真：(010)82614433

重庆周立功

地址：重庆市石桥铺科园一路二号大西洋国际大厦（赛格电子市场）1611 室
电话：(023)68796438　68796439
传真：(023)68796439

杭州周立功

地址：杭州市天目山路 217 号江南电子大厦 502 室
电话：(0571)89719480　89719481　89719482
89719483　89719484　89719485
传真：(0571)89719494

成都周立功

地址：成都市一环路南二段 1 号数码科技大厦 403 室
电话：(028)85439836　85437446
传真：(028)85437896

深圳周立功

地址：深圳市福田区深南中路 2072 号电子大厦 12 楼
电话：(0755)83781788（5 线）
传真：(0755)83793285

武汉周立功

地址：武汉市洪山区广埠屯珞瑜路 158 号 12128 室（华中电脑数码市场）
电话：(027)87168497　87168297　87168397
传真：(027)87163755

上海周立功

地址：上海市北京东路 668 号科技京城东座 7E 室
电话：(021)53083452　53083453　53083496
传真：(021)53083491

西安办事处

地址：西安市长安北路 54 号太平洋大厦 1201 室
电话：(029)87881296　83063000　87881295
传真：(029)87880865

厦门办事处

E-mail：sales.xiamen@zlgmcu.com

沈阳办事处

E-mail：sales.shenyang@zlgmcu.com

销售与服务网络（二）

广州致远电子股份有限公司

地址：广州市天河区车陂路黄洲工业区 3 栋 2 楼
邮编：510660
传真：(020)38601859
网址：www.zlg.cn
新浪微博：ZLG-周立功（http://weibo.com/ligongzhou）

技术支持：

CAN-bus：

电话：(020)22644381　22644382　22644253
邮箱：can.support@zlg.cn

iCAN 及数据采集：

电话：(020)28872344　22644373
邮箱：ican@zlg.cn

MiniARM：

电话：(020)28872684　28267813
邮箱：miniarm.support@zlg.cn

以太网：

电话：(020)22644380　22644385
邮箱：ethernet.support@zlg.cn

无线通讯：

电话：(020) 22644386
邮箱：wireless@zlg.cn

串行通讯：

电话：(020)28267800　22644385
邮箱：serial@zlg.cn

编程器：

电话：(020)22644371
邮箱：programmer@zlg.cn

分析仪器：

电话：(020)22644375
邮箱：tools@zlg.cn

ARM 嵌入式系统：

电话：(020) 22644383　22644384
邮箱：NXPARM@zlg.cn

楼宇自动化：

电话：(020)22644376　22644389　28267806
邮箱：mjs.support@zlg.cn
mifare.support@zlg.cn

销售：

电话：(020)22644249　22644399　22644372　22644261　28872524
28872342　28872349　28872569　28872573　38601786

维修：

电话：(020)22644245

目 录

1. 功能简介

ZLG9518S 芯片是一款 SPI 转多路串口的专用 IC，该芯片支持将高达 33MHz 的 SPI 接口转换为 8 路波特率低达 4800bps 的串口，实现了高低速接口之间的无缝连接。

ZLG9518S芯片在串口的数量上支持动态配置，即用户可以任意使用其中的一个或者多个串口，可随时打开或者关闭任意串口；任意两个串口之间的资源是不共享的，无论是收发缓存，还是外部端口，在物理上都是绝对独立的；每个串口都可以支持多种动态配置，如 8 种波特率、5 种校验方式、4 种数据长度、3 种停止位等，且都支持RTS和CTS的流控功能，用户可以根据自己的需要，使能或禁能任意一路或者多路的串口流控功能，甚至可以单独使能RTS功能或者CTS功能等；ZLG9518S芯片还支持中断，且有一个外部中断引脚IRQ，由于 8 路串口共享IRQ引脚，因此只要有任意 1 路串口满足产生中断的条件，IRQ引脚即输出中断信号，中断支持多种模式，比如FIFO中断、流控中断和错误中断等，且这些中断都可以根据自己需要任意启动或关闭；ZLG9518S芯片还具有检错的能力，当在通讯中产生错误时，用户可以通过查询相应的寄存器来得知产生错误的类别，且可以自由设置产生错误之后是否发送中断信号等。ZLG9518S芯片的内部结构如图 1.1所示。

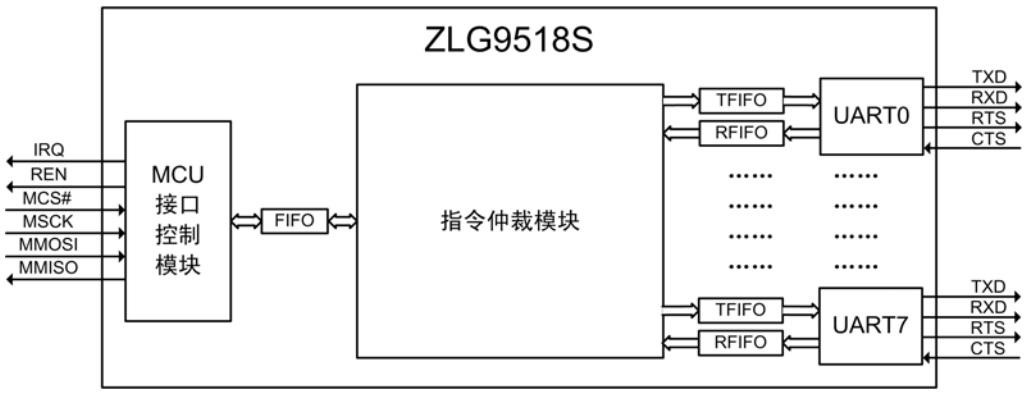


图 1.1　芯片内部框图

2. 引脚描述

ZLG9518S芯片的管脚如图 2.1所示。

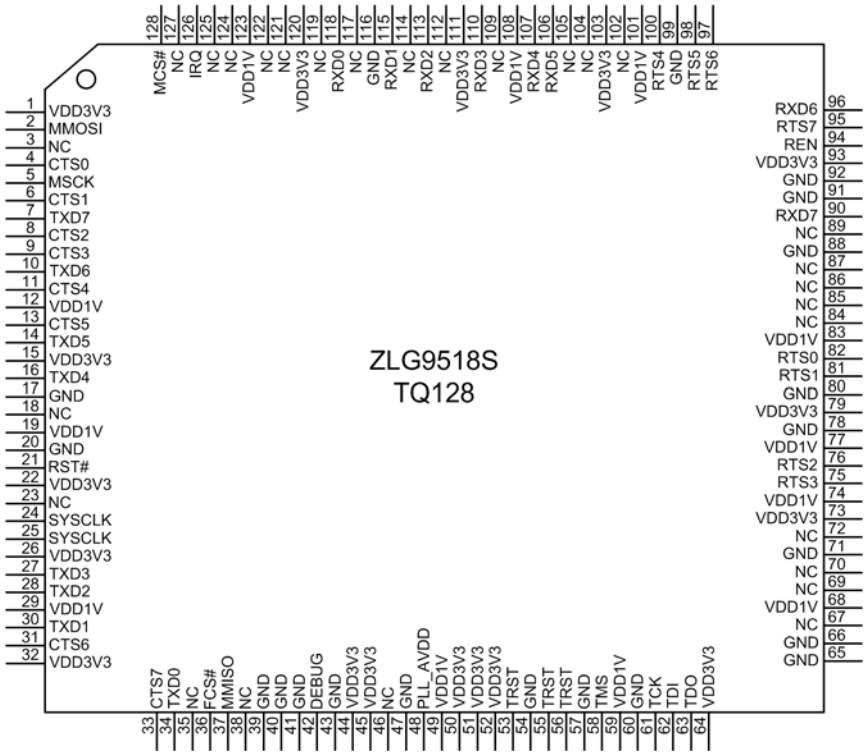


图 2.1　ZLG9518S 芯片引脚图

具体管脚说明如表 2.1所示。

表 2.1　ZLG9518S 芯片管脚说明

接口类型	信号名称	管脚号	属性	说明
调试接口	SYSCLK	24、25	I	20MHz 系统时钟信号
	RST#	21	I	系统复位信号，低电平复位
	DEBUG	42	I/O	调试接口信号
	TRST	53、55、56	I	测试复位输入信号
	TMS	58	I	测试状态机信号
	TCK	61	I	测试时钟信号
	TDI	62	I	测试数据输入信号
	TDO	63	O	测试数据输出信号
MCU 接口	MCS#	128	I	SPI 从机的片选信号，低电平有效
	MSCK	5	I	SPI 从机的时钟信号
	MMOSI	2	I	SPI 从机的串行数据输入信号
	MMISO	37	O	SPI 从机的串行数据输出信号
	REN	94	O	ZLG9518S 芯片的读使能信号，低电平有效
	IRQ	126	O	ZLG9518S 芯片的中断信号，低电平有效
串口	TXD0~TXD7	34、30、28、27、16、14	O	8 路串口的输出信号

续上表

接口类型	信号名称	管脚号	属性	说明
串口	TXD0~TXD7	10、7	O	8 路串口的输出信号
	RXD0~RXD7	118、115、113、110、107、106、96、90	I	8 路串口的输入信号
	RTS0~RTS7	82、81、76、75、100、98、97、95	O	8 路串口的流控 RTS 信号
	CTS0~CTS7	4、6、8、9、11、13、31、33	I	8 路串口的流控 CTS 信号
电源接口	VDD3V3	1、15、22、26、32、44、45、50、51、52、64、73、79、93、103、111、120	P	3.3V 电压输入，电流驱动能力>=330mA
	VDD1V	12、19、29、49、59、68、74、77、83、101、108、123	P	1V 电压输入，电流驱动能力>=180mA
	PLL_AVDD	48	P	模拟 PLL 的 1V 电压输入
	GND	17、20、39、40、41、43、47、54、57、60、65、66、71、78、80、88、91、92、99、116	P	地

3. 典型电路

ZLG9518S芯片的典型电路系统框图如图 3.1所示。值得注意的是，ZLG9518S芯片的固件是存放在外部的SPI Flash中，所以该电路中必须要外置一个SPI Flash，默认型号为旺宏的MX25L6445E。该SPI Flash可按照客户的要求进行修改，最小容量为 512Kbit。

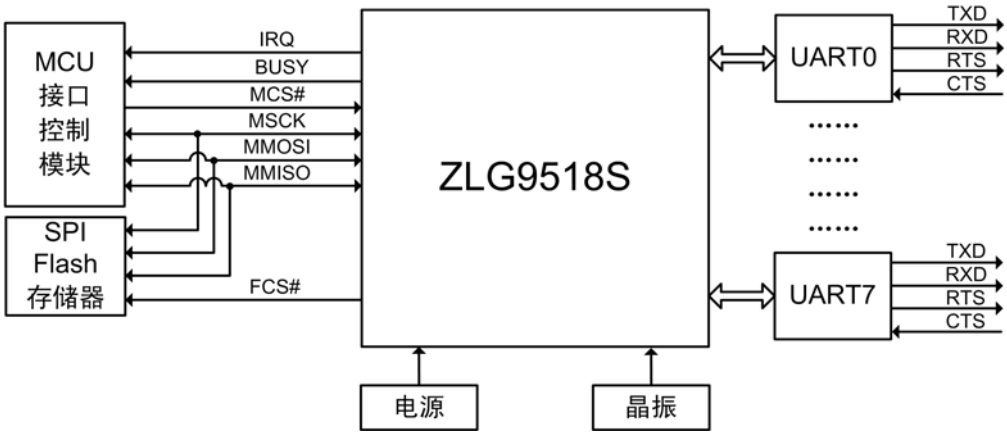


图 3.1　ZLG9518S 芯片系统框图

4. 功能描述

ZLG9518S芯片的所有功能都是通过寄存器来实现的，MCU可以通过SPI接口任意修改和访问ZLG9518S芯片的寄存器。MCU可以在ZLG9518S芯片的工作过程中，复位ZLG9518S芯片的FIFO、开启或关闭任意一个串口、配置任意一个串口的波特率、停止位、校验方式、数据长度等、开启或关闭任意一个串口的流控功能、配置任意一个串口的中断模式和中断使能等。ZLG9518S芯片内部共有 11 个寄存器，且每个寄存器都是 8 位的，如表 4.1所示。

表 4.1　ZLG9518S 芯片的寄存器说明

寄存器	地址	说明	属性
FCR	0	复位串口的发送和接收 FIFO	RW
LCR0	1	配置串口的状态、数据长度、校验方式和停止位	RW
LCR1	2	配置串口的波特率、流控和中断使能	RW
TCR	3	流控触发点寄存器	RW
TLR	4	FIFO 中断触发点寄存器	RW
TXLVL	5	发送 FIFO 中可用的空间	RO
RXLVL	6	接收 FIFO 中可用的空间	RO
IER	7	中断寄存器	RO
LSR	8	错误状态寄存器	RO
RHR	10	接收 FIFO，大小为 255 字节	RO
THR	11	发送 FIFO，大小为 255 字节	WO

注：其中没有用到的地址均为保留位，主要是方便以后的功能扩展。

ZLG9518S芯片支持 4 线的SPI接口，方便与外部的MCU进行通信，如图 4.1所示。ZLG9518S芯片将内部的复杂运算逻辑简化为如表 4.1所示的 11 个寄存器，使得外部MCU对ZLG9518S芯片的操作变得简单，可将ZLG9518S芯片当成外设来访问。除此之外，ZLG9518S芯片还有一个IRQ中断引脚，这使得MCU可以更加实时和高效的了解ZLG9518S芯片的当前工作状态，以便于及时处理和调整。

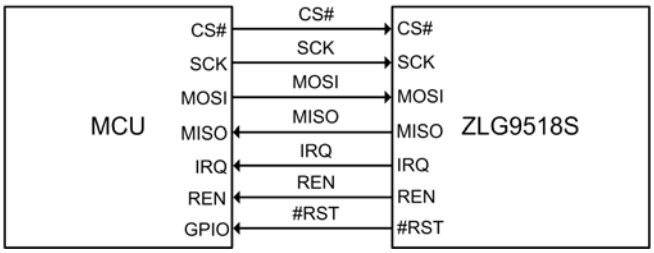


图 4.1　MCU 与 ZLG9518S 芯片的连接

另外，由于 MCU 和 ZLG9518S 芯片的上电不同步，为了确保 MCU 的初始化不至于影响 ZLG9518S 芯片的正常启动以及保证 MCU 的主控权，MCU 需要增加一个 GPIO 来控制 ZLG9518S 芯片的复位引脚，等待 MCU 初始化完毕后再重新复位 ZLG9518S 芯片。

4.1　复位 FIFO

MCU可以在ZLG9518S芯片的工作过程中，复位其串口的发送或者接收FIFO，使得FIFO恢复到初始状态。相比重新复位ZLG9518S芯片而言，MCU可以省略重新配置串口参数的步骤。具体操作是在FCR寄存器中，在对应的位上置 1 将自动复位一次FIFO，复位之后ZLG9518S芯片继续正常工作，此时该位不会自动清零，需由用户手动清零该位。如果该位不清零，ZLG9518S芯片也不会一直处于复位状态。ZLG9518S芯片的FIFO复位动作，只对当前写入的复位FIFO命令执行一趟复位。FCR寄存器详细参数如表 4.2所示。

表 4.2 FCR 寄存器（地址为 0）

位	说明
bit1:bit0	保留，可作为外部存储器被读写和保存数据，复位值为 0
bit2	复位发送 FIFO，置 1 时有效，复位值为 0
bit3	复位接收 FIFO，置 1 时有效，复位值为 0
bit7:bit4	保留，可作为外部存储器被读写和保存数据，复位值为 0

4.2 串口配置

ZLG9518S芯片的串口支持动态配置，MCU可以在ZLG9518S芯片的工作过程中，对任意一个串口进行配置，如波特率、数据长度、校验方式、停止位等。具体配置方法是，将所需的参数写入到LCR0 和LCR1 寄存器中，LCR0 和LCR1 寄存器的详细参数如表 4.3和表 4.4所示。

表 4.3 LCR0 寄存器（地址为 1）

位	说明
bit1:bit0	停止位，复位值为 0
bit4:bit2	校验位，复位值为 0
bit6:bit5	数据长度，复位值为 3
bit7	开关状态，复位值为 0

表 4.4 LCR1 寄存器（地址为 2）

位	说明
bit2:bit0	波特率，复位值为 7
bit3	硬件 CTS 流控使能，置 1 时使能，复位值为 0
bit4	硬件 RTS 流控使能，置 1 时使能，复位值为 0
bit5	FIFO 中断使能，置 1 时使能，复位值为 0
bit6	流控中断使能，置 1 时使能，复位值为 0
bit7	错误中断使能，置 1 时使能，复位值为 0

LCR0 和LCR1 寄存器中具体相关的位所代表的含义如表 4.5至表 4.9所示。

表 4.5 LCR0 寄存器的停止位

bit1:bit0（二进制）	说明
00	1 位停止位
01	1 位半停止位
10	2 位停止位
11	保留，如果写入将会导致错误寄存器的参数错误位置位

表 4.6 LCR0 寄存器的状态位

bit7（二进制）	说明
0	使串口处于关闭状态
1	使串口处于启动状态

表 4.7　LCR0 寄存器的数据长度位

bit6:bit5（二进制）	说明
00	5 位数据长度
01	6 位数据长度
10	7 位数据长度
11	8 位数据长度

表 4.8　LCR0 寄存器的校验位

bit4:bit2（二进制）	说明
000	无校验方式
001	奇校验方式
010	偶校验方式
011	MASK 校验方式
100	SPACE 校验方式
其他	保留，如果写入将会导致错误寄存器的参数错误位置位

表 4.9　LCR1 寄存器的波特率位

bit2:bit0（二进制）	说明
000	串口的波特率为 4800bps
001	串口的波特率为 9600bps
010	串口的波特率为 14400bps
011	串口的波特率为 19200bps
100	串口的波特率为 28800bps
101	串口的波特率为 38400bps
110	串口的波特率为 57600bps
111	串口的波特率为 115200bps

注：当写入的的参数错误时，ZLG9518S 芯片将会在错误寄存器中提示此时的参数输入错误，且此时不会将该参数赋值给寄存器。

4.3　流控功能

ZLG9518S芯片支持硬件流控RTS和CTS功能，其主要作用是为了在串口进行大量数据传输时，防止数据丢失。用户可以根据自己的应用场合，开启或者关闭流控功能。当关闭流控功能时，RTS输出低电平，表示始终允许对方发送了数据；当开启流控功能时，RTS会自动根据串口的接收缓存进行电平控制，而MCU只管通过SPI接口从ZLG9518S芯片发送或读取FIFO的数据即可。开启和关闭RTS和CTS功能主要是通过LCR1 寄存器的bit3 和bit4 来实现的，具体如表 4.4所示。

ZLG9518S芯片的自动流控功能还支持触发点的设置，通过向TCR寄存器写入相应的参数，即可实现对触发位置的调整。其中，TCR寄存器主要分为高 4 位和低 4 位，高 4 位代表恢复流控触发点，低 4 位代表终止流控触发点，具体如表 4.10所示。

表 4.10　TCR 寄存器（地址为 3）

位	说明
bit7:bit4	恢复流控触发点，复位值为 1
bit3:bit0	终止流控触发点，复位值为 2

通过对TCR寄存器的配置，可调整流控RTS引脚的输出电平跳变位置。流控RTS的输出波形就像施密特触发器一样，如图 4.2所示。当串口的接收缓存达到TCR寄存器中的终止流控触发点时，RTS从低电平跳变到高电平；当串口的接收缓存少于TCR寄存器中的恢复流控触发点时，RTS才会从高电平跳变回低电平，恢复串口的接收功能。当拉大恢复触发点和终止触发点的距离时，此时RTS引脚的电平跳变过渡时间变长；相反则过渡时间变短。

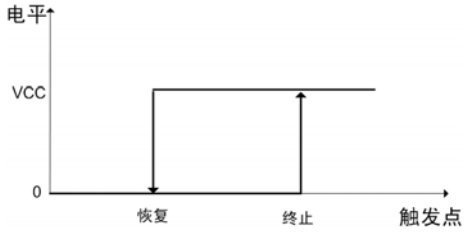


图 4.2　数据流控触发点示例图

调整恢复触发点和终止触发点的方法很简单，只需在TCR寄存器中，将相应的参数写入到该寄存器中即可，具体参数如表 4.11所示。

表 4.11　TCR 寄存器的触发点参数

bit7:bit4/ bit3:bit0(二进制)	说明
0000	保留，如果写入将会导致错误寄存器的参数错误位置位
0001	触发点为 16
0010	触发点为 32
0011	触发点为 48
0100	触发点为 64
0101	触发点为 80
0110	触发点为 96
0111	触发点为 112
1000	触发点为 128
1001	触发点为 144
1010	触发点为 160
1011	触发点为 176
1100	触发点为 192
1101	触发点为 208
1110	触发点为 224
1111	触发点为 240

注：恢复触发点的值必须小于终止触发点的值，否则会导致错误寄存器的参数错误位置位。

4.4　中断功能

ZLG9518S 芯片支持中断，由于 8 路串口共用一个中断引脚 IRQ，因此只要其中有一路串口满足中断条件，且使能了该中断，那么 IRQ 将输出信号通知 MCU，MCU 可以即刻响应中断，以及时了解和处理 ZLG9518S 芯片的事务。

ZLG9518S芯片的每一路串口都支持三种中断，分别为FIFO中断、流控中断和错误中断。这三种中断都可以根据需要使能和禁能，如表 4.4所示的LCR1 寄存器，可以控制该寄存器

的第 5、6 和 7 位来使能或者禁能某一中断。

当ZLG9518S芯片产生了中断信号时，MCU可以通过查询IER寄存器来获知具体是哪一类触发了中断。IER寄存器的具体参数如表 4.12所示。

表 4.12　IER 寄存器（地址为 7）

位	说明
bit7:bit5	保留，读出为 0
bit4	错误中断位，置 1 时有效，复位值为 0
bit3	流控 CTS 中断位，置 1 时有效，复位值为 0
bit2	流控 RTS 中断位，置 1 时有效，复位值为 0
bit1	发送 FIFO 中断位，置 1 时有效，复位值为 0
bit0	接收 FIFO 中断位，置 1 时有效，复位值为 0

注：IER 寄存器中只要有任意一位置 1，则 IRQ 引脚将输出低电平的中断信号，MCU 只能通过读取该寄存器来清零和清除中断信号。

其中，LCR1 寄存器的第 5 位控制着 IER 寄存器的发送和接收 FIFO 中断；LCR1 寄存器的第 6 位控制着 IER 寄存器的流控 RTS 和 CTS 中断；LCR1 寄存器的第 7 位则是控制着 IER 寄存器的错误中断位。当 LCR1 寄存器的相应中断控制位置 1 时，IER 寄存器在满足中断的条件下才会将相应的中断位置 1，进而产生中断；否则，即使满足了中断条件，IER 寄存器的相应中断位也不会置 1，从而导致中断被屏蔽和禁能。

4.4.1 FIFO 中断

ZLG9518S 芯片的任意一个串口收发缓存都各高达 255 字节，MCU 可以使用查询的方式获知某一路串口此时的 FIFO 中接收了多少个数据，进而决定是否要读取该 FIFO 中的数据。然而，采用查询的方式势必会降低 MCU 的效率，且在大多数情况下，用户可能只要接收到十几个数据后通知 MCU 去读数据，因此，采用 FIFO 中断方式即可实现。

FIFO中断分为发送FIFO中断和接收FIFO中断。当发送FIFO中的使用空间数超过发送FIFO的中断触发点时，且使能FIFO中断的条件下，IER寄存器的bit1 将置 1，并在IRQ引脚产生中断信号；同样，当接收FIFO中的使用空间数超过接收FIFO的中断触发点时，且使能FIFO中断的条件下，IER寄存器的bit0 将置 1，并在IRQ引脚产生中断信号。设置发送和接收FIFO的中断触发点主要是通过TLR寄存器，TLR寄存器如表 4.13所示。

表 4.13　TLR 寄存器（地址为 4）

位	说明
bit7:bit4	接收 FIFO 中断触发点，复位值为 1
bit3:bit0	发送 FIFO 中断触发点，复位值为 1

调整发送和接收中断触发点的方法和TCR寄存器一样，只需将相应的参数写入到该寄存器中即可，具体参数如表 4.11所示。一般情况下，接收FIFO中断使用的会比较多，而发送FIFO中断使用的很少，此时，我们可以将发送FIFO的中断触发点设置比较大，比如设置为 240 个字节，这样发送FIFO中断基本就不会出现。

4.4.2 流控中断

与FIFO中断不同，流控中断分为RTS中断和CTS中断。当接收FIFO中的使用空间数超过了终止触发点，且使能流控中断的条件下，如果此时还有数据继续送往接收FIFO，IER寄存器的bit2 将置 1，并在IRQ引脚产生中断信号；而CTS中断则根据对方RTS无效时，此时发送

FIFO中还存在数据，且使能了流控中断的条件下，IER寄存器的bit3 将置 1，并在IRQ引脚产生中断信号。调整恢复触发点和终止触发点，只需在TCR寄存器中，将相应的参数写入到该寄存器中即可，具体参数如表 4.11所示。

4.4.3 错误中断

ZLG9518S芯片还支持错误中断，可以在数据的传输过程中通知MCU通讯错误或者写入命令、参数错误等。用户可以根据自己的应用场合，控制LCR1 寄存器第 7 位来使能或者禁能错误中断。具体如表 4.4所示。

4.5 检错功能

ZLG9518S 芯片支持检错功能，这使得 ZLG9518S 芯片可以实时监控数据通讯总线，当出现错误时，ZLG9518S 芯片可以分析产生错误的种类并及时通知 MCU 去处理，从而具有自我纠错的能力。

检错种类包括发送和接收FIFO溢出错误、命令错误、参数错误、帧错误、串口校验错误和读错误等。当出现错误时，ZLG9518S芯片会将这些错误标记在LSR寄存器，LSR寄存器的详细参数如表 4.14所示。如果此时使能错误中断，那么LSR寄存器会将错误及时通知MCU，以制止错误的继续产生。

表 4.14 LSR 寄存器（地址为 8）

位	说明
bit7	保留，读出为 0
bit6	读错误，置 1 时有效，复位值为 0
bit5	参数错误，置 1 时有效，复位值为 0
bit4	命令错误，置 1 时有效，复位值为 0
bit3	帧错误，置 1 时有效，复位值为 0
bit2	校验错误，置 1 时有效，复位值为 0
bit1	接收 FIFO 溢出错误，置 1 时有效，复位值为 0
bit0	发送 FIFO 溢出错误，置 1 时有效，复位值为 0

注：LSR 寄存器中只要有任意一位置 1，且在只能错误中断的条件下，则 IRQ 引脚将输出低电平的中断信号，MCU 只能通过读取该寄存器来清零和清除中断信号。

4.5.1 FIFO 溢出错误

FIFO 溢出错误包括发送 FIFO 溢出错误和接收 FIFO 溢出错误。

当发送 FIFO 已满时，此时 MCU 如果还继续往 ZLG9518S 芯片的 FIFO 中发送数据，将会导致 ZLG9518S 芯片由于没有再多的空余空间存放数据而造成数据丢失，因此产生发送 FIFO 溢出错误，并在 LSR 寄存器的相应位置 1；同样，当接收 FIFO 已满时，此时如果 ZLG9518S 芯片的串口 RXD 引脚还有数据在接收，这将会导致由于 MCU 没有及时读走数据而造成对方的串口数据丢失，因此产生接收 FIFO 溢出错误，并在 LSR 寄存器相应位置 1。

4.5.2 校验/帧错误

ZLG9518S 芯片可配置为奇校验、偶校验等方式，通过数据校验，可大大提高数据的可靠性。当 ZLG9518S 芯片对接收的串口数据校验出错时，将会在 LSR 的相应位置 1，以提示此时的串口数据存在问题。出现问题的可能性可能是双方的配置不一致，或者是通讯距离过远导致电压不稳定等问题；当 ZLG9518S 芯片提示帧错误时，此时也可能是双方的配置不一致或者停止位错误等问题。

4.5.3 命令/参数错误

命令或参数错误主要是 MCU 在发送命令及命令参数的时候数据错误。比如命令错误，MCU 可能写入了 ZLG9518S 芯片中不存在的寄存器地址，或者是对只读的寄存器进行写操作，或者是对只写的寄存器进行读操作等；而参数错误出现的情况，在前面一些寄存器中也有涉及，主要是写入的参数不合法，在逻辑上有错误等。

4.5.4 读错误

当 ZLG9518S 芯片的 FIFO 为空时，如果此时 MCU 还去读该芯片的 FIFO，将会导致读错误，并在 LSR 寄存器的相应位置 1，且读出的数据均为 0。

4.6 RXLVL 和 TXLVL

ZLG9518S芯片内部还设置了RXLVL和TXLVL两个寄存器，MCU可以通过这两个寄存器来查询此时ZLG9518S芯片内部的接收和发送FIFO中未使用的空间数。当FIFO中使用的空间数不足以触发中断时，可以通过查询这两个寄存器来确定FIFO中是否存在数据，从而决定是否继续读或者写数据。RXLVL和TXLVL寄存器如表 4.15和表 4.16所示。

表 4.15 TXLVL 寄存器（地址为 5）

位	说明
bit7:bit0	发送 FIFO 中剩余的空间数，复位值为 255

表 4.16 RXLVL 寄存器（地址为 6）

位	说明
bit7:bit0	接收 FIFO 中剩余的空间数，复位值为 255

4.7 写 FIFO

写FIFO操作和写其他寄存器操作是一样的，MCU通过将数据写入到THR寄存器即可实现将数据通过ZLG9518S芯片的串口发送出去。THR寄存器如表 4.17所示。写FIFO操作一次只能对FIFO写入一个数据，当串口的数据长度不是 8 的时候，都要按照低位对齐。具体写时序如图 5.2所示。

表 4.17 THR 寄存器（地址为 11）

位	说明
bit7:bit0	发送 FIFO，复位值为 0

4.8 读 FIFO

与写FIFO不同，读FIFO既可以支持一次只读一个数据，也可以支持连读模式，即MCU只需发送一次读操作命令，就可以直接从ZLG9518S芯片的RHR寄存器中连续读多个FIFO的数据，直到读空。RHR寄存器如表所示。具体时序如图 5.4所示。

表 4.18 RHR 寄存器（地址为 10）

位	说明
bit7:bit0	接收 FIFO，复位值为 0

5. SPI 接口协议

ZLG9518S 芯片与 MCU 的通讯接口为标准的 4 线 SPI 接口，采用传输模式 3，SPI 时钟最高可达 33MHz。为了实现 MCU 只通过 SPI 接口就能访问和控制 8 路串口，因此在 SPI 的传输上必须要有一些约定，比如寻址哪个串口、发送什么命令或者参数等，因此在 SPI 接口上需要做一些协议。

5.1 协议

无论MCU是读还是写ZLG9518S芯片，都需要先发送一个字节长度的命令和数据。为了使MCU访问ZLG9518S芯片的方式变得更灵活，ZLG9518S芯片在SPI时序上做了调整，使得MCU可以采用硬件外设或者I/O口模拟SPI时序。主要修改的地方在于片选信号CS上，在发送完寄存器地址或者完整的一帧命令之后，CS线既可以上拉，也可以不用上拉，而无需一定要发送完一整帧命令之后才拉高CS线，这使得MCU可以直接使用 8 位的硬件SPI外设来访问ZLG9518S芯片，使得MCU访问ZLG9518S芯片变得更加灵活和简单。操作ZLG9518S芯片的时序如图 5.1所示。

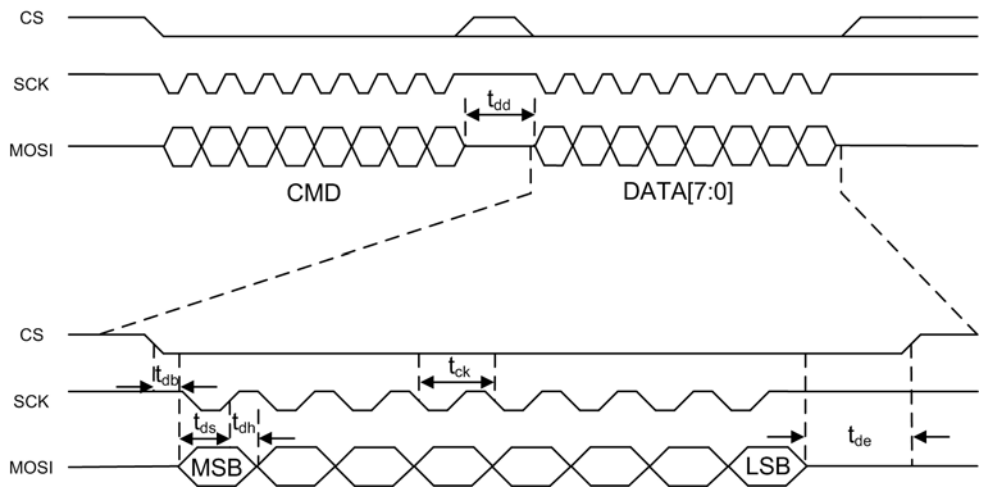


图 5.1 ZLG9518S 芯片的驱动时序

注：以上时序为 ZLG9518S 芯片的写时序，如果前面的功能描述中无提及时序要求，则都是默认按照上图的时序。ZLG9518S 芯片的写时序和读时序稍微不一样，在后面会详细介绍。

图 5.1中对ZLG9518S芯片的驱动时序要求如表 5.1所示。

表 5.1 ZLG9518S 芯片的驱动时序参数

参数	最小值	典型值	最大值	单位	说明
t_{db}	0	-	-	ns	CS 信号与第一个有效数据的时间
t_{ds}	3.72	-	-	ns	数据建立时间
t_{dh}	5.14	-	-	ns	数据保持时间
t_{ck}	30.3	-	-	ns	时钟信号
t_{de}	0	-	-	ns	CS 信号与最后一个有效数据的时间
t_{dd}	0	-	-	ns	发送字节的间隔时间

由于ZLG9518S芯片中包含了 11 个寄存器，且寄存器可读可写，同时又有 8 如串口，因此，在发送的命令中，必须包含这些信息，命令的具体参数如表 5.2所示。

表 5.2　命令协议

位	说明
bit7	读写位，高电平为读，低电平为写
bit6:bit3	寄存器地址
bit2:bit0	串口通道号

5.2　写时序

写寄存器和写FIFO操作是一样的。当写寄存器时，后面的数据就是该寄存器的参数，ZLG9518S芯片会自动判断写入命令和参数的合法性，当命令和参数合法时，才会将该参数写入到相应的寄存器中；而写FIFO的时候，后面的数据就是串口数据，ZLG9518S芯片会将该数据转载到发送FIFO中，然后通过相应的串口将该数据送出。时序如图 5.2所示。

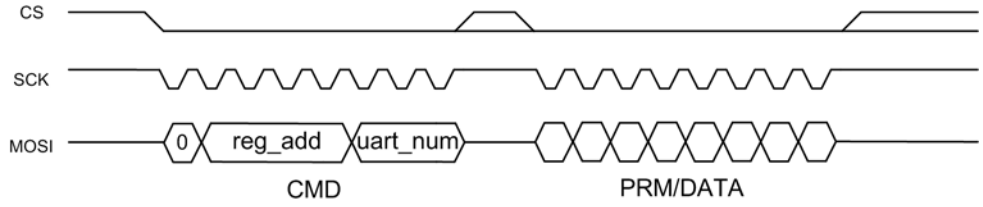


图 5.2　ZLG9518S 芯片的写时序

5.3　读时序

读时序与写时序不同，在发送完命令之后，紧接着的数据并不是FIFO的数据，而是命令的参数，之后才是真正的数据。命令参数实际上只对RHR寄存器有效，而对于其他寄存器是无效的。ZLG9518S芯片读其他寄存器的时序如图 5.3所示。MCU首先发送一个字节长度的读命令，紧接着发送一个字节长度的参数，这个参数中的值可以任意，因为它只对RHR寄存器有效，最后才是寄存器的数据。ZLG9518S芯片的读使能信号REN标识了MCU何时可以去读该寄存器的值，当REN为低时，代表MCU此时可以去读该寄存器的值了。

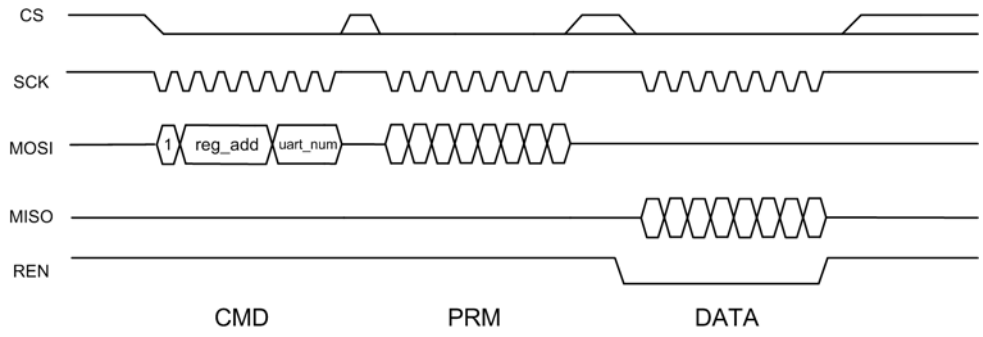


图 5.3　ZLG9518S 芯片的读时序

当读RHR寄存器时，命令参数的作用是代表连读FIFO中数据的个数，详细的命令参数如表 5.3所示。

表 5.3　读命令的参数

位	说明
bit7:bit0	当为 0 或者 1 时，读 FIFO 一个数据; 当为 n（n > 1）时，读取 FIFO 中 n 个数据

读RHR寄存器的时序如图 5.4所示。当MCU发送的命令参数为n时，代表需要连续从FIFO

中读取数据的个数，之后MCU将在REN信号的提示下，逐一从ZLG9518S芯片的FIFO中读取n个数据。

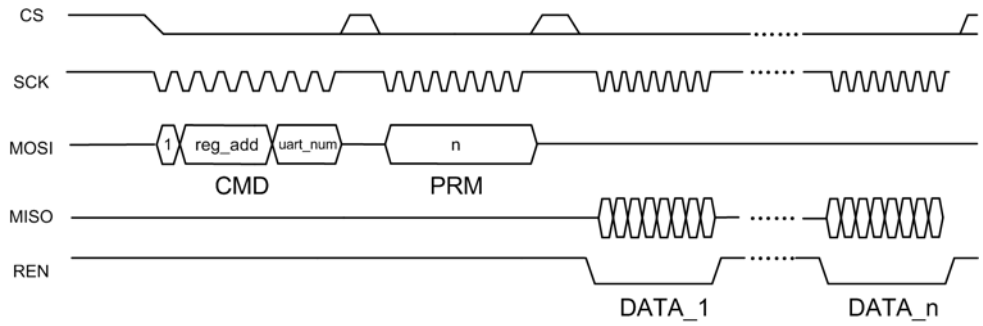


图 5.4　ZLG9518S 芯片的连读时序

6. 操作说明

本次的操作说明都是在 ZLG9518S Demo 板上进行展开。该 Demo 板上主要包含 ZLG9518S 芯片和 LPC1342F 主控芯片。通过 LPC1342F 主控芯片相应的驱动程序来测试 ZLG9518S 芯片。

6.1 初始化配置

ZLG9518S芯片的初始化配置主要是配置串口的波特率、校验方式等，以及流控、中断等，为了提高代码的可读性，我们都以宏定义的形式来编写，如程序清单 6.1所示。

程序清单 6.1 初始化配置函数

```
/*******************************************************************************************
** Function name:           s2u_configure_one_uart
** Descriptions:            配置 ZLG9518S 芯片的一个串口
** input parameters:        uart_chan: 串口通道号
** output parameters:       无
** Returned value:          无
*******************************************************************************************/
void s2u_configure_one_uart(uint8_t uart_chan)
{
    uint8_t cmd_prm;                                    /* 定义一个命令参数变量          */
    /*
    * 配置串口为 8 位数据长度、无校验方式、一位停止位和打开状态
    */
    cmd_prm  =  S2UC_LCR0_PUT_STATE_BIT(S2UC_LCR0_ON_STATE)
             |  S2UC_LCR0_PUT_DLEN_BIT(S2UC_LCR0_8_BIT_DATA)
             |  S2UC_LCR0_PUT_PARITY_BIT(S2UC_LCR0_PARITY_NONE)
             |  S2UC_LCR0_PUT_STOP_BIT(S2UC_LCR0_STOP_1_BIT);
    zlg9518s_send_write_cmd(ZLG9518S_LCRO_REG, uart_chan, cmd_prm, cmd_buf);

    /*
    * 配置串口的波特率为 115200bps、关闭流控功能、流控中断和错误中断，使能 FIFO 中断
    */
    cmd_prm  =  S2UC_LCR1_PUT_BAUD_BIT(S2UC_LCR1_BAUD_115200)
             |  S2UC_LCR1_PUT_CTS_BIT(S2UC_LCR1_CTS_OFF)
             |  S2UC_LCR1_PUT_RTS_BIT(S2UC_LCR1_RTS_OFF)
             |  S2UC_LCR1_PUT_FIFO_ENINT_BIT(S2UC_LCR1_INT_EN)
             |  S2UC_LCR1_PUT_STREAM_ENINT_BIT(S2UC_LCR1_INT_DIS)
             |  S2UC_LCR1_PUT_ERR_ENINT_BIT(S2UC_LCR1_INT_DIS);
    zlg9518s_send_write_cmd(ZLG9518S_LCR1_REG, uart_chan, cmd_prm, cmd_buf);

    /*
    * 配置流控恢复触发点为 32 个字节，终止触发点为 224 字节
    */
```

```
    cmd_prm   =   S2UC_TCR_PUT_STREAM_START_BIT(S2UC_TCR_STREAM_32_BYTE_MAX255)
                | S2UC_TCR_PUT_STREAM_END_BIT(S2UC_TCR_STREAM_224_BYTE_MAX255);
    zlg9518s_send_write_cmd(ZLG9518S_TCR_REG, uart_chan, cmd_prm, cmd_buf);

    /*
    * 配置接收 FIFO 中断触发点为 224 个字节，发送 FIFO 触发点为 240 个字节
    */
    cmd_prm   =   S2UC_TLR_PUT_RFIFO_INT_BIT(S2UC_TCR_STREAM_224_BYTE_MAX255)
                | S2UC_TLR_PUT_TFIFO_INT_BIT(S2UC_TCR_STREAM_240_BYTE_MAX255);
    zlg9518s_send_write_cmd(ZLG9518S_TLR_REG, uart_chan, cmd_prm, cmd_buf);
}
```

这些配置参数可以在ZLG9518S芯片的工作过程中随时随意修改，且配置顺序可以任意调换，无需一定要按照程序清单 6.1中的格式。

如果用户不想修改程序清单 6.1的函数，可以将一些配置参数引出作为形参来传递，但这样的话，形参较多，建议将这些参数封装为数组或者链表后再作为形参使用。本次操作为了简要说明，所以直接初始化。

假设我们现在只初始化串口 0，那么可调用程序清单 6.1的函数，如程序清单 6.2所示。

程序清单 6.2　初始化一个串口

```
s2u_configure_one_uart(UART_CHAN_0);                    /* 初始化串口 0                */
```

如果要初始化多个串口，那么只需多次引用该函数即可，如程序清单 6.3所示。

程序清单 6.3　初始化多个串口

```
s2u_configure_one_uart(UART_CHAN_0);                    /* 初始化串口 0                */
s2u_configure_one_uart(UART_CHAN_1);                    /* 初始化串口 1                */
s2u_configure_one_uart(UART_CHAN_2);                    /* 初始化串口 2                */
s2u_configure_one_uart(UART_CHAN_3);                    /* 初始化串口 3                */
s2u_configure_one_uart(UART_CHAN_4);                    /* 初始化串口 4                */
s2u_configure_one_uart(UART_CHAN_5);                    /* 初始化串口 5                */
s2u_configure_one_uart(UART_CHAN_6);                    /* 初始化串口 6                */
s2u_configure_one_uart(UART_CHAN_7);                    /* 初始化串口 7                */
```

6.2　写/读命令

发送读命令或者写命令的时候，主要是将数据按照表 5.2的命令协议格式进行处理。如程序清单 6.4所示。

程序清单 6.4　命令函数

```
/*********************************************************************************************
** Function name:           zlg9518s_set_cmd
** Descriptions:            ZLG9518S 芯片的命令协议格式
** input parameters:        rw：读写位；reg：寄存器地址；uart_chan：串口通道号；prm：命令参数
** output parameters:       cmd_buf：命令缓存
** Returned value:          无
*********************************************************************************************/
```

```
void zlg9518s_set_cmd(uint8_t rw, uint8_t reg, uint8_t uart_chan, uint8_t prm, uint16_t *cmd_buf)
{
    uint16_t cmd;
    cmd   =   S2UC_PUT_RW_BIT(rw)
        |   S2UC_PUT_REG_BIT(reg)
        |   S2UC_PUT_CHAN_BIT(uart_chan)
        |   S2UC_PUT_DATA_BIT(prm);

    cmd_buf[0] = (cmd >> 8) & 0xFF;
    cmd_buf[1] = (cmd >> 0) & 0xFF;
}
```

6.2.1 写命令

写命令函数可以直接调用程序清单 6.4中的命令函数，如程序清单 6.5所示。

程序清单 6.5　写命令函数

```
/*********************************************************************************************
** Function name:          zlg9518s_send_write_cmd
** Descriptions:           ZLG9518S 芯片的写命令
** input parameters:       reg：寄存器地址；uart_chan：串口通道号；prm：命令参数；cmd_buf：缓存
** output parameters:      无
** Returned value:         无
*********************************************************************************************/
void zlg9518s_send_write_cmd(uint8_t reg, uint8_t uart_chan, uint8_t prm, uint16_t *cmd_buf)
{
    zlg9518s_set_cmd(S2UC_WRITE_FORM, reg, uart_chan, prm, cmd_buf);
    SPI_OUT(cmd_buf, 2);                                    /* 发送写命令                   */
}
```

假设现在向串口 0 的THR寄存器写入 0x95 和 0x18 这两个数据，代码如程序清单 6.6所示。

程序清单 6.6　写命令函数使用示例

```
/*
* 向串口 0 的 THR 寄存器分别发送 0x95 和 0x18 数据
*/
zlg9518s_send_write_cmd(ZLG9518S_THR_REG, UART_CHAN_0, 0x95, cmd_buf);
zlg9518s_send_write_cmd(ZLG9518S_THR_REG, UART_CHAN_0, 0x18, cmd_buf);
```

执行程序清单 6.6的代码之后，通过逻辑分析仪捕获的时序如图 6.1所示。

图 6.1　写命令函数使用示例时序图

6.2.2 读命令

读命令函数也可以直接调用程序清单 6.4中的命令函数，如程序清单 6.7所示。

程序清单 6.7 读命令函数

```
/*********************************************************************************************
** Function name:           zlg9518s_send_read_cmd
** Descriptions:            ZLG9518S 芯片的读命令
** input parameters:        reg：寄存器地址；uart_chan：串口通道号；prm：命令参数；cmd_buf：缓存
** output parameters:       无
** Returned value:          无
*********************************************************************************************/
void zlg9518s_send_read_cmd(uint8_t reg, uint8_t uart_chan, uint8_t prm, uint16_t *cmd_buf)
{
    zlg9518s_set_cmd(S2UC_READ_FORM, reg, uart_chan, prm, cmd_buf);
    SPI_OUT(cmd_buf, 2);                                    /* 发送读命令                   */
}
```

由于读寄存器或FIFO之前，需要判断REN信号，且还有连读模式等，因此我们需要再编写一个读寄存器或数据的函数，如程序清单 6.8所示。

程序清单 6.8 读数据函数

```
/*********************************************************************************************
** Function name:           zlg9518s_read_reg_pack
** Descriptions:            ZLG9518S 芯片的读数据函数
** input parameters:        reg：寄存器地址；uart_chan：串口通道号；prm：命令参数
** output parameters:       无
** Returned value:          无
*********************************************************************************************/
uint8_t zlg9518s_read_reg_pack(uint8_t reg, uint8_t uart_chan, uint8_t prm)
{
    uint8_t uart_data;
    zlg9518s_send_read_cmd(reg, uart_chan, prm, cmd_buf);   /* 发送读命令                   */

    if (reg == ZLG9518S_RHR_REG) {                          /* RHR 寄存器连读               */
        if (prm > 1) {
            zlg9518s_read_mul_data(prm, data_buf);
            return 0;
        }
    }

    zlg9518s_ren();                                         /* 判断读使能信号               */
    uart_data = zlg9518s_read_data(reg_buf);                /* 读所有寄存器                 */
    return uart_data;
}
```

假设现在ZLG9518S芯片的RHR寄存器有两个数据，分别为 0x95 和 0x18，那么我们可

以发送相应的读命令，并将这两个数据从RHR中读出来，代码如程序清单 6.9所示。

程序清单 6.9　读命令函数使用示例

```
uint8_t uart_left;                                          /* RHR 中的数据个数          */
                                                            /*  读取 RXLVL 寄存器        */
uart_left = UART_TX_BUF_SIZE - read_reg_pack(ZLG9518S_RXLVL_REG, UART_CHAN_0, 0);
read_reg_pack(ZLG9518S_RHR_REG, UART_CHAN_0, uart_left); /*  读取 RHR 寄存器          */
```

执行程序清单 6.9的代码之后，通过逻辑分析仪捕获的时序如图 6.2所示。

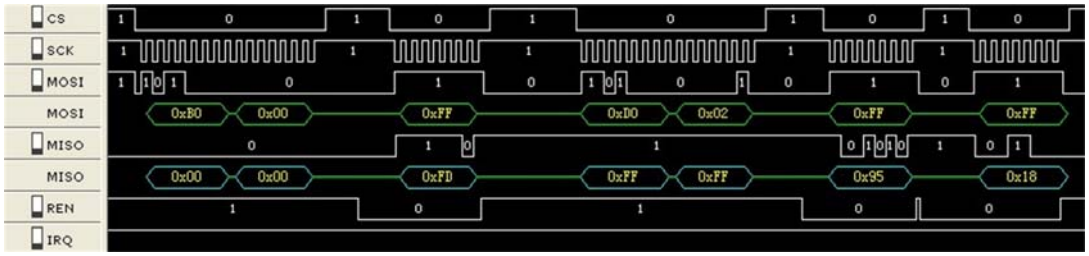


图 6.2　读命令函数使用示例时序图

6.3　硬件流控

硬件流控功能的使用可以在初始化的时候就就开启，也可以在ZLG9518S芯片工作的过程中随时启动。开启硬件流控功能的代码如程序清单 6.10所示。

程序清单 6.10　开启硬件流控的配置函数

```
/*******************************************************************************************
** Function name:           s2u_configure_one_uart
** Descriptions:            配置 ZLG9518S 芯片的一个串口
** input parameters:        uart_chan: 串口通道号
** output parameters:       无
** Returned value:          无
*******************************************************************************************/
void s2u_configure_one_uart(uint8_t uart_chan)
{
    uint8_t cmd_prm;                                        /* 定义一个命令参数变量      */

    /*
    * 配置串口的波特率为 115200bps、关闭 FIFO 中断和错误中断，开启流控功能和使能流控中断
    */
    cmd_prm  =   S2UC_LCR1_PUT_BAUD_BIT(S2UC_LCR1_BAUD_115200)
               | S2UC_LCR1_PUT_CTS_BIT(S2UC_LCR1_CTS_ON)
               | S2UC_LCR1_PUT_RTS_BIT(S2UC_LCR1_RTS_ON)
               | S2UC_LCR1_PUT_FIFO_ENINT_BIT(S2UC_LCR1_INT_EN)
               | S2UC_LCR1_PUT_STREAM_ENINT_BIT(S2UC_LCR1_INT_DIS)
               | S2UC_LCR1_PUT_ERR_ENINT_BIT(S2UC_LCR1_INT_DIS);
    zlg9518s_send_write_cmd(ZLG9518S_LCR1_REG, uart_chan, cmd_prm, cmd_buf);

    /*
```

```
    * 配置流控恢复触发点为 16 个字节，终止触发点为 32 字节
    */
    cmd_prm   =   S2UC_TCR_PUT_STREAM_START_BIT(S2UC_TCR_STREAM_16_BYTE_MAX255)
                | S2UC_TCR_PUT_STREAM_END_BIT(S2UC_TCR_STREAM_32_BYTE_MAX255);
    zlg9518s_send_write_cmd(ZLG9518S_TCR_REG, uart_chan, cmd_prm, cmd_buf);
}
```

为了方便测试和观察，我们可以通过MCU的串口TXD连续发送 32 个字节数据给ZLG9518S芯片，ZLG9518S芯片会将这 32 个串口数据存放在FIFO中，且此时由于FIFO中有效数据个数已经达到流控终止触发点，ZLG9518S芯片将使RTS引脚变为无效，并使IRQ引脚输出中断信号，如图 6.3所示。

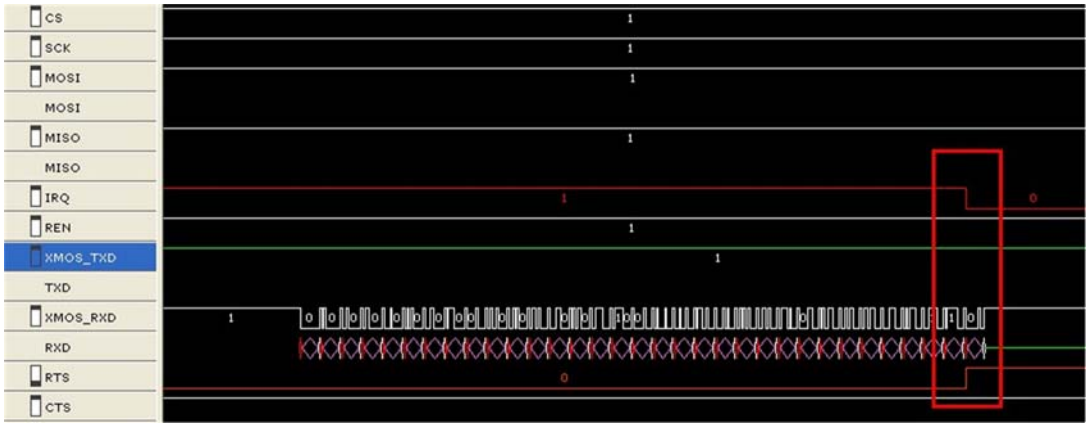


图 6.3　硬件流控触发及中断

此时如果MCU通过SPI接口将缓存在ZLG9518S芯片的FIFO中的串口数据读出来，那么将会在读走 16 个数据之后，RTS引脚恢复为低电平。当MCU发送读IER寄存器时，则IRQ中断信号自动清除。如图 6.4所示。

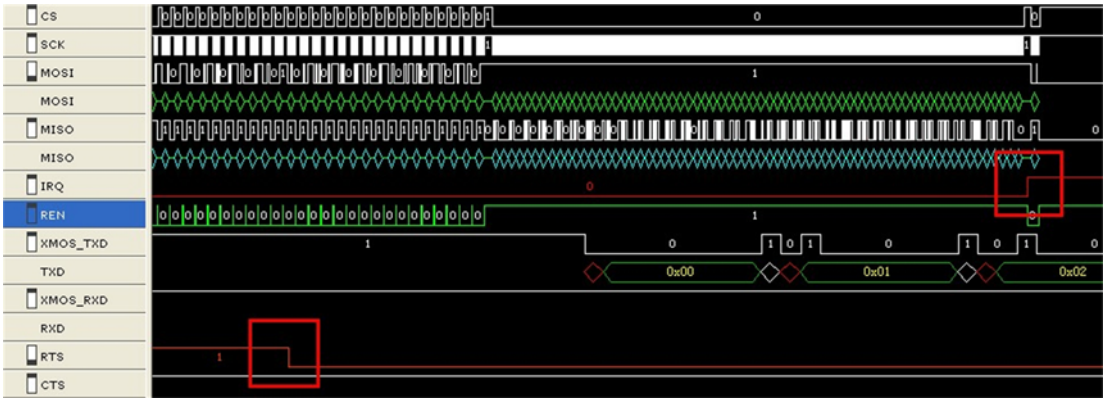


图 6.4　硬件流控和中断信号的清除

6.4　中断处理

由于ZLG9518S芯片的中断是低电平有效，因此，MCU需要将与ZLG9518S芯片IRQ引脚连接的I/O口配置为下降沿触发中断模式，并编写相应的中断服务程序。中断服务程序主要执行的任务是读取IER中断寄存器并判断中断的类型，处理相应的中断事件，如程序清单 6.11所示。用户可以根据自己的需要添加相应的应用程序。

程序清单 6.11　ZLG9518S 芯片中断处理函数

```
/*********************************************************************************************
** Function name:           s2u_irq
** Descriptions:            ZLG9518S 芯片的中断处理
** input parameters:        uart_chan：串口通道号
** output parameters:       无
** Returned value:          无
*********************************************************************************************/
void s2u_irq(uint8_t uart_chan)
{
    uint8_t ier_reg;
    uint8_t lsr_reg;
    uint8_t uart_left;
                                                    /* 读取 IER 寄存器                  */
    ier_reg = read_reg_pack(ZLG9518S_IER_REG,    uart_chan, 0);
                                                    /* 接收 FIFO 中断或 RTS 流控中断    */
    if (S2UC_IER_IS_RHR_INT(ier_reg) || S2UC_IER_IS_RTS_INT(ier_reg)) {
        uart_left = UART_TX_BUF_SIZE - read_reg_pack(ZLG9518S_RXLVL_REG, uart_chan, 0);
        read_reg_pack(ZLG9518S_RHR_REG, uart_chan, uart_left);
    }
                                                    /* 发送 FIFO 中断或 CTS 流控中断    */
    if (S2UC_IER_IS_THR_INT(ier_reg) || S2UC_IER_IS_CTS_INT(ier_reg)) {
                                                    /* 添加用户程序                     */
    }
                                                    /* 错误中断                         */
    if (S2UC_IER_IS_ERR_INT(ier_reg)) {
                                                    /* 读取 LSR 寄存器                  */
        lsr_reg = read_reg_pack(ZLG9518S_LSR_REG,    uart_chan, 0);

        if (S2UC_LSR_IS_TFIFO_SPILL_ERR(lsr_reg)) {     /* 发送 FIFO 溢出错误           */
                                                        /* 添加用户程序                 */
        }

        if (S2UC_LSR_IS_RFIFO_SPILL_ERR(lsr_reg)) {     /* 接收 FIFO 溢出错误           */
                                                        /* 添加用户程序                 */
        }
                                                        /* 校验错误或帧错误             */
        if (S2UC_LSR_IS_PARITY_ERR(lsr_reg) || S2UC_LSR_IS_FRAME_ERR(lsr_reg)) {
                                                        /* 添加用户程序                 */
        }
                                                        /* 参数错误或命令错误           */
        if (S2UC_LSR_IS_PRM_ERR(lsr_reg) || S2UC_LSR_IS_CMD_ERR(lsr_reg)) {
                                                        /* 添加用户程序                 */
        }
```

```
                                                    /* 读错误                    */
            if (S2UC_LSR_IS_READ_ERR(lsr_reg)) {
                                                    /* 添加用户程序              */
            }
        }
}
```

7. 免责声明

本数据手册所陈述的产品文本及相关软件版权均属广州周立功单片机有限公司所有，其产权受国家法律绝对保护，未经本公司授权，其它公司、单位、代理商及个人不得非法使用和拷贝，否则将受国家法律的严厉制裁。广州周立功单片机有限公司保留任何时候在不事先声明的情况下对本文档的修改权力。您如果需要我们公司的产品及相关信息，请及时与我们联系， 我们将热情接待。