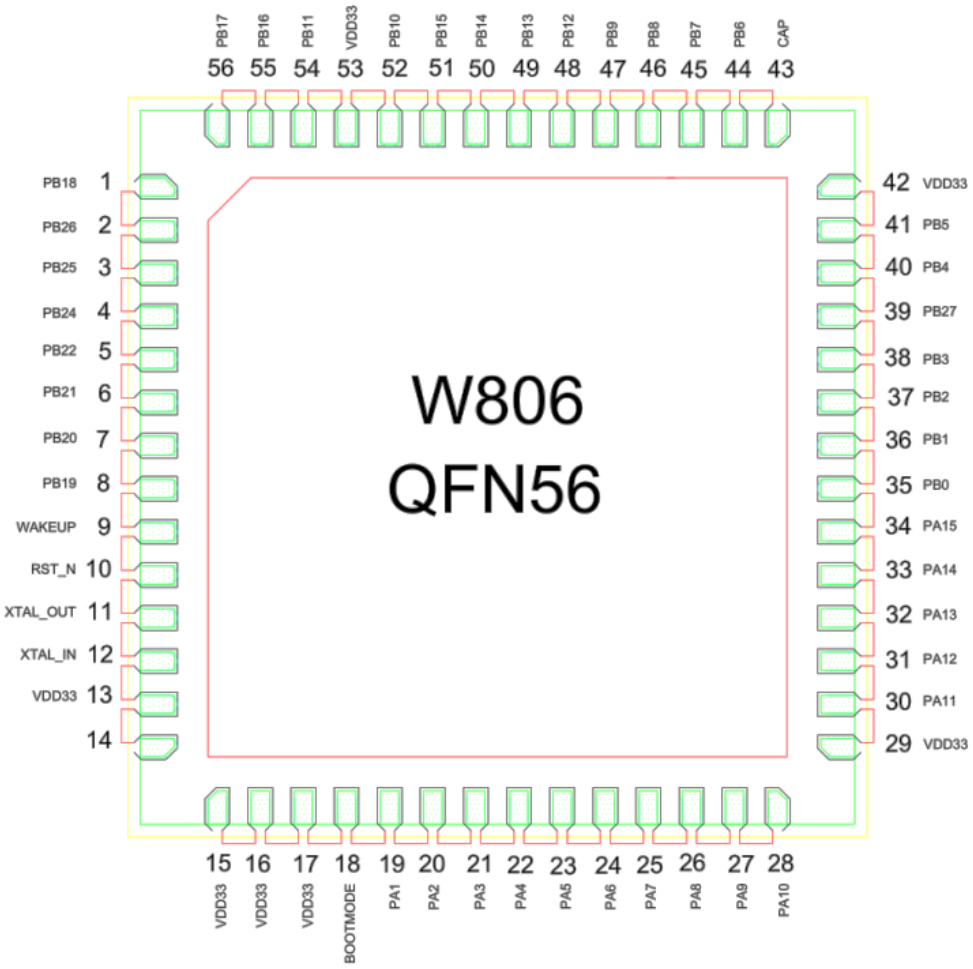
W806 MCU 寄存器目录

W806 地址映射 2
CORET 系统滴答时钟 3
VIC 中断配置 4
总线结构 8
RCC 系统时钟寄存器 9
PMU 电源管理寄存器 12
看门狗 WDG 14
PWM 波形控制器 15
GPIO 寄存器 19
SPI 寄存器 21
I2C 寄存器 24
UARTx 寄存器(x=[0-5]) 25
TIM 定时器 27
ADC 模数控制寄存器 28
I2S 音频控制寄存器 30
DMA 寄存器 34
TOUCH 触摸控制器 37
片内 QFLASH 38
RSA 加密模块 41
PSRAM 接口控制器 42
HSPI 高速 SPI 47
7816 模块(UART2 模块) 43

W806 资料整理



1. W806 低电平复位，当电平值低于 2.0V 时，芯片处于复位状态。复位时低电平需持续 100us 以上。
2. 芯片参考时钟选用 40MHz 频率。
3. ADC 引脚输入电压范围 0-2.4V。当高于 2.4V 时外部需做分压处理后才可进入 ADC 接口。
4. 芯片进入休眠状态后，可用 WAKEUP 功能唤醒芯片,WAKEUP 管脚输入高电平即可唤醒休眠状态中的芯片。芯片在正常工作状态时 WAKEUP 脚为低电平，可以下拉 10K 电阻。

5. CDK 标志符：_I ：只读　　_O：只写　　_IO：可读写
_IM：只读结构体　　_OM：只写结构体　　_IOM：可读写结构体

6.CDK 编程软件配置：

ROM = 20K	始地址：Start = 0x00	大小:Size = 0x5000(20*1024)
RAM = 288K	始地址：Start = 0x20000000	大小:Size = 0x48000(288*1024)

W806 地址映射

名称	细分	起始地址	终止地址	说明
ROM	/	0x0000 0000	0x0004 FFFF	存放固化的固件代码
FLASH	/	0x0800 0000	0x0FFF FFFF	专用指令存储器
SRAM	/	0x2000 0000	0x2002 7FFF	固件内存和指令存储区
Mac RAM	/	0x2002 8000	0x2004 7FFF	SDIO/H-SPI/UART 数据缓存
PSRAM	/	0x3000 0000	0x3080 0000	外设内存
CONFIG	/	0x4000 0000	0x4000 05FF	RSA 配置空间
	/	0x4000 0600	0x4000 07FF	GPSEC 配置空间
	/	0x4000 0800	0x4000 09FF	DMA 配置空间
	/	0x4000 0A00	0x4000 0CFF	SDIO_HOST 配置空间
	/	0x4000 0D00	0x4000 0DFF	PMU 配置空间
	/	0x4000 0E00	0x4000 0EFF	Clock 与 Reset 配置空间
	/	0x4000 0F00	0x4000 0FFF	MacPHY Router 配置空间
	/	0x4000 1000	0x4000 13FF	BBP 配置空间
	/	0x4000 1400	0x4000 17FF	MAC 配置空间
	/	0x4000 1800	0x4000 1FFF	SEC 配置空间
	/	0x4000 2000	0x4000 21FF	FLASH Controller 配置空间
	/	0x4000 2200	0x4000 23FF	PSRAM_CTRL 配置空间
	/	0x4000 2400	0x4000 25FF	SDIO Slave 配置空间
	/	0x4000 2600	0x4000 27FF	H-SPI 配置空间
	/	0x4000 2800	0x4000 29FF	SD Wrapper 配置空间
	/	0x4000 2A00	0x4000 A9FF	BT Core 配置空间
	/	0x4000 B000	0x4000 B0FF	SASC-B1 一级总线内存安全配置模块
	/	0x4000 B100	0x4000 B1FF	SASC-Flash Flash 安全配置模块
	/	0x4000 B200	0x4000 B2FF	SASC-B2 二级总线内存安全配置模块
APB	I2C	0x4001 0000	0x4001 01FF	/
	ADC	0x4001 0200	0x4001 03FF	/
	SPI	0x4001 0400	0x4001 05FF	/
	UART0	0x4001 0600	0x4001 07FF	/
	UART1	0x4001 0800	0x4001 09FF	/
	UART2	0x4001 0A00	0x4001 0BFF	/
	UART3	0x4001 0C00	0x4001 0DFF	/
	UART4	0x4001 0E00	0x4001 0FFF	/
	UART5	0x4001 1000	0x4001 11FF	/
	GPIOA	0x4001 1200	0x4001 13FF	/
	GPIOB	0x4001 1400	0x4001 15FF	/
	WDG	0x4001 1600	0x4001 17FF	/
	TIMER	0x4001 1800	0x4001 19FF	/
	RF	0x4001 1A00	0x4001 1BFF	/

APB	LCD	0x4001 1C00	0x4001 1DFF	/	
	PWM	0x4001 1E00	0x4001 1FFF	/	
	I2S	0x4001 2000	0x4001 22FF	/	
	BT	0x4001 2200	0x4001 23FF	/	
	TOUCH	0x4001 2400	0x4001 25FF	/	
	TIPC	0x4001 2600	0x4001 25FF	/	
	RF_BIST DAC 发射内存		0x4001 4000	0x4000 BFFF	
	RF_BIST ADC 接收内存		0x4001 C000	0x4003 BFFF	

CORET 系统滴答时钟

1.系统嘀嗒计时器频率与 CPU 频率一致。

#define CTRL：计时器状态与控制寄存器

Bit [16] COUNTFLAG：前一次读取此寄存器后计数器是否计数到 0　0:未到 0　1:计数到 0

计数器 VAL 由 1 到 0，置位该位；读 CTRL 寄存器及任何写 VAL 寄存器会清零 COUNTFLAG。

Bit [2] CLKSOURCE：时钟源选择，**写此位无效，**无外部时钟读该位恒为 1　　**默认：1**

CLKSOURCE[2]=0	外部时钟，且外部时钟频率<=内部时钟频率的一半
CLKSOURCE[2]=1	内部时钟

Bit [1] TICKINT:计数器计数至 0 是否使能系统计时器中断　　0：不使能　　1：使能

Bit [0] ENABLE：计时器使能位　　0：不使能　　1：使能

#define LOAD：循环计时值寄存器(用于配置 VAL 寄存器计数至 0 后自动装载的数值)

Bit [23:0] RELOAD：计数值(写 0 即**停止**循环计数器)，**若计时 N 个周期，RELOAD = N-1**

注：使用外部时钟后，须等到计数器正常计数开始后(即 VAL 为非 0 值)，才可以将 LOAD 置为 0，以让计数器在下一次循环时停止工作，否则计数器无法开始第一次计数。

#define VAL：计时器当前值寄存器

Bit [23:0] CURRENT：当前计数值，**写该寄存器即清零该寄存器、且不会改变系统计时器的中断状态位。**

#define CALIB：计时器校准寄存器，只读寄存器

Bit [31] NOREF：设备是否采用外部时钟　　0：外部时钟　　1：无外部时钟

Bit [30] SKEW：10ms 校准是否准确　　0：10ms 校准准确　　1:10ms 校准有误差

Bit [23:0] TENMS：10ms 校准时间对应值(该值为 0 表示校准值未知，可能因为参考时钟是一个未知的输入或者是动态变化的)。

由于系统计时器中 LOAD 和 VAL 两个寄存器没有复位值，在系统计时器工作之前，必须按照下列步骤进行初始化操作：

1.向 LOAD 寄存器里写入需要的回填值，**实际值= (LOAD -1)**；

2.向 VAL 寄存器里写入任意值从而使它清零；

3.操作 CTRL 寄存器，使能系统计时器。

VIC 中断配置

1.中断控制器支持中断嵌套，当处理器正在处理一个中断请求的同时有一个更高优先级的中断请求，处理器将中断当前中断服务程序的处理，响应该更高优先级的中断请求。

2. 中断数量硬件可配，支持 4、8、16、24、32、64、96 和 128。当软件未设置优先级 IPR 寄存器时，所有中断源优先级默认为最高优先级(0 级)，多个中断同时发生时 CPU 根据中断向量编号进行仲裁(编号越低优先级越高)。

3. 软件通过 8bit 寄存器为每个中断配置优先级，**0 优先级为最高优先级**

中断源数量 N	中断优先级配置说明
N <= 32	Bit[7:6]配置优先级，Bit[5:0]保留
32 < N <= 64	Bit[7:5]配置优先级，Bit[4:0]保留
64 < N	Bit[7:4]配置优先级，Bit[3:0]保留

```
typedef struct {
  __IOM uint32_t ISER[4U];  中断使能寄存器                          1:使能
  __IOM uint32_t IWER[4U];  中断低功耗模式下唤醒 MCU 使能寄存器   1:使能
  __IOM uint32_t ICER[4U];  中断禁止寄存器                          1:禁止
  __IOM uint32_t IWDR[4U];  中断低功耗唤醒禁止寄存器              1:禁止低功耗唤醒
  __IOM uint32_t ISPR[4U];  中断挂起寄存器                          1：将中断挂起
  __IOM uint32_t ISSR[4U];  中断使能保护寄存器(配合 ISER 使用)   1：使能保护
  __IOM uint32_t ICPR[4U];  中断除挂寄存器                          1：解挂对应中断
  __IOM uint32_t ICSR[4U];  Security interrupt clear register 清除保护
  __IOM uint32_t IABR[4U]; 中断正在被执行标志寄存器(仅限读)   1：中断正在被执行
  __IOM uint32_t IPR[32U];  中断优先级寄存器(单个 IPR[n]对应 4 个中断源)
  __IM  uint32_t ISR;       中断状态寄存器(显示正在处理中断号和等待优先级最高号)
        Bit [30:12] VECTPENDING：处于等待状态的优先级最高的中断向量号
        Bit [8:0] VECTACTIVE: CPU 当前正在处理的中断向量号
__IOM uint32_t IPTR;  中断优先级阈值寄存器(即：使低抢占优先级中断能得到及时响应，
                      且等待状态的中断优先级必须高于定义的阈值，才能发起中断抢占请求)。
        Bit [31] EN：阈值有效位     0:抢占不需要优先级高于阈值      1:需要高于阈值
        Bit [16:8] VECTTHRESHOLD：优先级阈值对应的中断向量号
        Bit [7:0] PRIOTHRESHOLD：抢占的优先级阈值(根据中断源数量进行配置)
  __IOM uint32_t TSPEND;    Tspend 中断使能(使能后即进入等待状态，且 CPU 最后执行)
        Bit [0] SETENA：使能设置   0：Tspend 未使能      1：Tspend 使能
  __IOM uint32_t TSABR;   Tspend 中断状态查询(仅限读),Bit[0]位写 0 清零，不可写 1
        Bit [0] Active：Tspend 中断是否被响应  读:0：CPU 未响应   1:CPU 正在执行
  __IOM uint32_t TSPR;    Tspend 中断优先级，Tspend 中断的优先级需要设置为最低
        Bit [7:0] PRI：Tspend 中断优先级，设置位数参考中断源数量
} VIC_Type;
```

1.HAL_NVIC_SetPriority(IRQn_Type IRQn, uint32_t Priority); 设置优先级

2.HAL_NVIC_EnableIRQ(IRQn_Type IRQn); 中断使能

VIC->ISER[0]=BIT(IRQn); VIC->ISSR[0]=BIT[IRQn];(仅适用于[0-31]中断)

3.HAL_NVIC_DisableIRQ(IRQn_Type IRQn); 中断禁止

VIC->ICER[0]&=~ BIT(IRQn);

4.中断函数格式：__attribute__((isr)) void 中断函数名():被包含于 Startup.s 文件

5. void SystemClock_Config(uint32_t clk) 配置系统时钟频率

中断向量编号	编号值	中断入口函数	描述
SDIO_IRQn	0	SDIO_IRQHandler	SDIO 中断
MAC_IRQn	1	MAC_IRQHandler	MAC 中断
RF_CFG_IRQn	2	RF_Cfg_IRQHandler	RF_CFG 中断
SEC_IRQn	3	SEC_IRQHandler	SEC 中断
DMA_Channel0_IRQn	4	DMA_Channel0_IRQHandler	DMA0 通道中断
DMA_Channel1_IRQn	5	DMA_Channel1_IRQHandler	DMA1 通道中断
DMA_Channel2_IRQn	6	DMA_Channel2_IRQHandler	DMA2 通道中断
DMA_Channel3_IRQn	7	DMA_Channel3_IRQHandler	DMA3 通道中断
DMA_Channel4_7_IRQn	8	DMA_Channel4_7_IRQHandler	DMA[4-7]通道中断
DMA_BRUST_IRQn	9	DMA_BRUST_IRQHandler	DMA Brust 全局中断
I2C_IRQn	10	I2C_IRQHandler	I2C 中断
ADC_IRQn	11	ADC_IRQHandler	ADC 中断
SPI_LS_IRQn	12	SPI_LS_IRQHandler	低速 SPI 中断
SPI_HS_IRQn	13	SPI_HS_IRQHandler	高速 SPI 中断
GPIOA_IRQn	14	GPIOA_IRQHandler	GPIOA 中断
GPIOB_IRQn	15	GPIOB_IRQHandler	GPIOB 中断
UART0_IRQn	16	UART0_IRQHandler	UART0 中断
UART1_IRQn	17	UART1_IRQHandler	UART1 中断
TOUCH_IRQn	18	TOUCH_IRQHandler	TOUCH 中断
UART2_5_IRQn	19	UART2_5_IRQHandler	UART[2-5]中断
BLE_IRQn	20	BLE_IRQHandler	BLE 中断
BT_IRQn	21	BT_IRQHandler	蓝牙中断
PWM_IRQn	22	PWM_IRQHandler	PWM 中断
I2S_IRQn	23	I2S_IRQHandler	I2S 中断
SIDO_HOST_IRQn	24	SDIO_HOST_IRQHandler	SDIO 主中断
SYS_TICK_IRQn	25	CORET_IRQHandler	CORET 时钟中断
RSA_IRQn	26	RSA_IRQHandler	RSA 中断
CRYPTION_IRQn	27	GPSEC_IRQHandler	GPSEC 中断
FLASH_IRQn	28	FLASH_IRQHandler	FLASH 中断
PMU_IRQn	29	PMU_IRQHandler	PMU 中断
TIM_IRQn	30	TIM0_5_IRQHandler	TIMER[0-5]中断
WDG_IRQn	31	WDG_IRQHandler	WDG 中断
Tspend 中断	/	**tspend_handler**	**SETENA[0]=1 中断就绪**

W806 引脚介绍							
名称	类型	OPT1	OPT2	OPT3	OPT4	OPT5	OPT6
WAKEUP	I	高电平唤醒					
RESET	I	低电平复位					
XTAL_OUT	晶振输出	/	/	/	/	/	/
XTAL_IN	晶振输入	/	/	/	/	/	/
PA_0	BOOTMODE	I2S_MCLK	LSPI_CS	PWM2	I2S_DO	/	/
PA_1	I/O-20MHz	JTAG_CK	I2C_SCL	PWM3	I2S_LRCK	ADC_1	/
PA_2	I/O-20MHz	UART1_RTS	UART2_TX	PWM0	UART3_RTS	ADC_4	/
PA_3	I/O-20MHz	UART1_CTS	UART2_RX	PWM1	UART3_CTS	ADC_3	/
PA_4	I/O-20MHz	JTAG_SWO	I2C_SDA	PWM4	I2S_BCK	ADC_2	/
PA_5	I/O-20MHz	UART3_TX	UART2_RTS	PWM_BREAK	UART4_RTS	/	/
PA_6	I/O-20MHz	UART3_RX	UART2_CTS	/	UART4_CTS	LCD_SEG31	/
PA_7	I/O-20MHz	PWM4	LSPI_MOSI	I2S_MCK	I2S_DI	LCD_SEG3	Touch_0
PA_8	I/O-20MHz	PWM_BREAK	UART4_TX	UART5_TX	I2S_BCLK	LCD_SEG4	/
PA_9	I/O-50MHz	MMC_CLK	UART4_RX	UART5_RX	I2S_LRCLK	LCD_SEG5	TOUCH_1
PA_10	I/O-50MHz	MMC_CMD	UART4_RTS	PWM0	I2S_DO	LCD_SEG6	TOUCH_2
PA_11	I/O-50MHz	MMC_DAT0	UART4_CTS	PWM1	I2S_DI	LCD_SEG7	/
PA_12	I/O-50MHz	MMC_DAT1	UART5_TX	PWM2	LCD_SEG8	TOUCH_13	/
PA_13	I/O-50MHz	MMC_DAT2	UART5_RX	PWM3	LCD_SEG9	/	/
PA_14	I/O-50MHz	MMC_DAT3	UART5_CTS	PWM4	LCD_SEG10	TOUCH_14	/
PA_15	I/O-50MHz	PSRAM_CK	UART5_RTS	PWM_BREAK	LCD_SEG11	/	/
PB_0	I/O-80MHz	PWM0	LSPI_MISO	UART3_TX	PSRAM_CK	LCD_SEG12	Touch_3
PB_1	I/O-80MHz	PWM1	LSPI_CK	UART3_RX	PSRAM_CS	LCD_SEG13	Touch_4
PB_2	I/O-80MHz	PWM2	LSPI_CK	UART2_TX	PSRAM_D0	LCD_SEG14	Touch_5
PB_3	I/O-80MHz	PWM3	LSPI_MISO	UART2_RX	PSRAM_D1	LCD_SEG15	Touch_6
PB_4	I/O-80MHz	LSPI_CS	UART2_RTS	UART4_TX	PSRAM_D2	LCD_SEG16	Touch_7
PB_5	I/O-80MHz	LSPI_MOSI	UART2_CTS	UART4_RX	PSARM_D3	LCD_SEG17	Touch_8
PB_6	I/O-50MHz	UART1_TX	MMC_CLK	HSPI_CK	SDIO_CK	LCD_SEG18	Touch_9
PB_7	I/O-50MHz	UART1_RX	MMC_CMD	HSPI_INT	SDIO_CMD	LCD_SEG19	Touch_10
PB_8	I/O-50MHz	I2S_BCK	MMC_D0	PWM_BREAK	SDIO_D0	LCD_SEG20	Touch_11
PB_9	I/O-50MHz	I2S_LRCK	MMC_D1	HSPI_CS	SDIO_D1	LCD_SEG21	Touch_12
PB_10	I/O-50MHz	I2S_DI	MMC_D2	HSPI_DI	SDIO_D2	LCD_SEG22	/
PB_11	I/O-50MHz	I2S_DO	MMC_D3	HSPI_DO	SDIO_D3	LCD_SEG23	/
PB_12	I/O-50MHz	HSPI_CK	PWM0	UART5_CTS	I2S_BCLK	LCD_SEG24	/
PB_13	I/O-50MHz	HSPI_INT	PWM1	UART5_RTS	I2S_LRCLK	LCD_SEG25	/
PB_14	I/O-50MHz	HSPI_CS	PWM2	LSPI_CS	I2S_DO	LCD_SEG26	/
PB_15	I/O-50MHz	HSPI_DI	PWM3	LSPI_CK	I2S_DI	LCD_SEG27	/
PB_16	I/O-50MHz	HSPI_DO	PWM4	LSPI_MISO	UART1_RX	LCD_SEG28	/
PB_17	I/O-20MHz	UART5_RX	PWM_BREAK	LSPI_MOSI	I2S_MCLK	LCD_SEG29	/
PB_18	I/O-10MHz	UART5_TX	LCD_SEG30	/	/	/	/
PB_19	*I/O-10MHz*	*UART0_TX*	*PWM0*	*UART1_RTS*	*I2C_SDA*	**UART0 用于程序下载，端口引脚勿复用！**	
PB_20	*I/O-10MHz*	*UART0_RX*	*PWM1*	*UART1_CTS*	*I2C_SCL*		

PB_21	I/O-10MHz	UART0_RTS	PCM_SYNC	LCD_COM1	/	/	/
PB_22	I/O-10MHz	UART0_CTS	PCM_CK	LCD_COM2	/	/	/
PB_24	I/O-20MHz	LSPI_CK	PWM2	LCD_SEG2	/	/	/
PB_25	I/O-20MHz	LSPI_MISO	PWM3	LCD_COM0	/	/	/
PB_26	I/O-20MHz	LSPI_MOSI	PWM4	LCD_SEG1	/	/	/
PB_27	I/O-80MHz	PSRAM_CS	UART0_TX	LCD_COM3	/	/	/

合宙 AIR103 引脚(引脚映射待验证)						
名称	类型	OPT1	OPT2	OPT3	OPT4	PSRAM IO
PA_1	/	I2C_SCL	ADC_0	/	/	PSRAM_CS=PB_27
PA_2	IO	ADC_3	PWM30	/	/	PSRAM_D1(SO)=PB_03
PA_3	IO	ADC_2	PWM31	/	/	PSRAM_D2(WP)=PB_04
PA_4	IO	I2C_SDA	ADC_1	/	/	PSRAM_D0(SI)=PB_02
PA_7	IO	PWM04	/	/	/	PSRAM_D3=PB_05
PA_8	IO	/	/	/	/	(HOLD)
PA_9	IO	/	/	/	/	PSRAM_SCLK=PA_015
PA_10	IO	PWM10	/	/	/	
PA_11	IO	PWM11	/	/	/	
PA_12	IO	PWM12	UART5_TX	/	/	
PA_13	IO	UART5_RX	PWM3	/	/	LED IO
PA_14	IO	PWM14	/	/	/	LED1=PB_25
PB_0	IO	PWM00	UART3_TX	/	/	
PB_1	IO	PWM01	UART3_RX	/	/	LED2=PB_26
PB_2	IO	PWM02	SPI0_CK	UART2_TX	PSRAM_D0	
PB_3	IO	PWM03	SPI0_MISO	UART2_RX	PSRAM_D1	LED3=PB_24
PB_4	IO	SPI0_CS	PSRAM_D2	UART4_TX	/	
PB_5	IO	SPI0_MOSI	PSARM_D3	UART4_RX	/	
PB_6	IO	UART1_TX	SDIO_CK	/	/	
PB_7	IO	UART1_RX	SDIO_CMD	/	/	
PB_8	IO	SDIO_D0	/	/	/	
PB_9	IO	SDIO_D1	/	/	/	
PB_10	IO	SDIO_D2	/	/	/	
PB_11	IO	SDIO_D3	/	/	/	
PB_12	IO	PWM12	UART5_TX	/	/	
PB_13	IO	PWM21	/	/	/	
PB_14	IO	PWM22	SPI1_CS	/	/	
PB_15	IO	PWM23	SPI1_CK	/	/	
PB_16	IO	SPI1_MISO	PWM24	/	/	
PB_17	IO	SPI1_MOSI	/	/	/	
PB_18	IO	/	/	/	/	

注：SPI0 与 SPI1 是同一个 SPI 控制器，端口复用仅能选一种使用； PWM 脚的命名数字分 2 位：XY，其中 Y 相同的 PWM 脚，只能选取一个，不能同时使用，例如 PWM01 与 PWM11 不能同时生效。

总线结构

1. **AHB-1 总线包含：**CPU、DMA、GPSEC、ROM(**20K**)、FLASH、SRAM(160KB)、RSA、SDIO-HOST、PSRAM

设备	功能描述
CPU	CPU 最高时钟频率 240MHz
DMA	支持链表结构的独立 8 通道 DMA，支持 8 个请求
GPSEC	通用加密/解密模块，自动完成指定内存空间数据块的加密并回写
ROM(20K)	**存放 CPU 上电后的初始化固件和寄存器空间分配，将 CPU 控制权交给 FLASH 中存储的固件。**
FLASH	1Mb 存储固件代码以及运行参数
SRAM(160KB)	可用于存放指令或数据
RSA	加密运算控制，支持最大 2048Bit 位加密运算
SDIO-HOST	SDIO 接口外设，最高支持 50MHz
PSRAM_CTRL	QSPI 接口的 PSRAM 控制器。可通过此控制器访问外接 PSRAM。QSPI 接口时钟由总线时钟分频得到，最高支持 80MHz 时钟。

2. AHB-2 总线包含:SRAM(128K)、PMU、HSPI、APB_BUS2(**I2C、I2S、LSPI、PWM、UART、WODG、GPIO、TOUCH、7816、LCD**)，总线时钟最快工作在 **40MHz** 频率。

设备	功能描述
SRAM(128K)	SDIO/SPI/UART 等接口使用此 RAM 用作数据缓存
APB_BUS2	I2C、I2S、LSPI、PWM、UART、WODG、GPIO、TOUCH，

3. AHB-1 总线设备可以访问所有内存区域，AHB-2 总线上的设备只能访问二级总线上 128KB SRAM 的内存。

4. W806 系统采用 **40MHz/24MHz** 晶体作为系统时钟源，系统内置 DPLL，固定输出 **480MHz** 时钟作为全系统的时钟源。

5.W806 芯片内部集成 1M Byte QFlash，通过集成在芯片内部的 32Kb Cache 实现执行 QFlash 内部程序;当需要向 QFlash 存入数据时可通过 QFlash 控制寄存器配置地址、执行指令存入；**用户在对 QFlash 进行数据读取、存入时无需进行状态判断和等待操作，此过程 QFlash 寄存器自动完成，QFlash 控制器返回时表示读、写已完成。**

6.W806 最大支持外部 64Mb 的 PSRAM，最高读写速度 80MHz，PSRAM 支持 XIP 方式执行程序，CPU Cache 支持缓存 PSRAM 中数据。

7.芯片上电后 CPU 会启动执行 ROM 中的固件，从而加载 Flash 中用户指定程序，ROM 固件在开始运行时会读取 BootMode(PA0)引脚，根据 PA0 引脚信号进入相应启动状态

BootMode(PA0)状态	启动条件	启动模式
高电平		正常启动流程
低电平	持续时间<30ms,快速测试模式无效(用户慎用)	正常启动流程
	持续时间>=30ms	进入功能模式(如:下载固件)

8.RAM 使用情况

内存块	功能	起始地址	终止地址	大小	说明
160Kbyte	Stack&Heap	0x20000000	0x20003FFF	16Kbyte	ROM 使用
	NC	0x20004000	0x20027FFF	144Kbyte	NC
128Kbyte	NC	0x20028000	0x20047FF	128Kbyte	NC

RCC 系统时钟寄存器

1. **时钟自适应关断：**芯片依据内部的某些状态的迁移，自适应关断某些功能模块的时钟(用户请**不要更改配置**，否则可能会在配置 PMU 功能时导致系统异常。

2. **软件复位：**通过 RST(软件复位控制寄存器)可对子系统软件复位，**复位后该位不会自动清除**，若要子系统恢复正常工作必须**置位**相应位。软复位功能并不会复位 CPU 及 WatchDog。

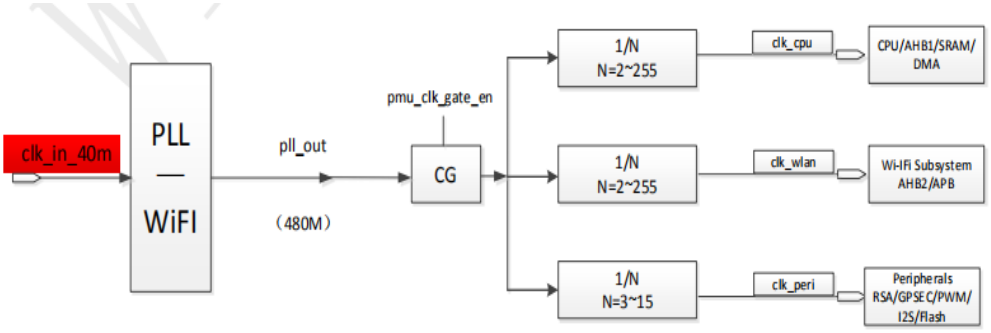


图 4 系统时钟分频关系

3. **系统总线**的时钟与 **CPU 时钟**一致，WLAN 根时钟是整个 **WLAN 系统**的时钟源头，为保证 WLAN 正常工作，其频率必须**不低于 160MHz。**

4. **数据总线(I2C、I2S、LSPI、PWM、UART、WODG、GPIO、TOUCH、7816、LCD)**时钟固定为 **CLK_DIV 寄存器 WLAN[15:8]**分频后的 1/4；**加密模块时钟**则为 **PERIPHERAL[27:24]**分频后的 1/2。

#define CLK_EN：软件时钟门控使能寄存器

Bit [21]	TOUCH：TOUCH 时钟使能，	0：关闭 1：使能 默认值：1
Bit [20]	BT：蓝牙模块时钟使能，	0：关闭 1：使能 默认值：1
Bit [19]	QSRAM：QSPI_SRAM 模块时钟使能，	0：关闭 1：使能 默认值：1
Bit [18]	SDIO：SDIO 主机时钟使能，	0：关闭 1：使能 默认值：1
Bit [17]	GPSEC：GPSEC 加密模块时钟使能，	0：关闭 1：使能 默认值：1
Bit [16]	RSA：RSA 加密模块时钟使能，	0：关闭 1：使能 默认值：1
Bit [15]	I2S：I2S 模块时钟使能，	0：关闭 1：使能 默认值：1
Bit [14]	LCD：LCD 时钟使能，	0：关闭 1：使能 默认值：1
Bit [13]	PWM：PWM 时钟使能，	0：关闭 1：使能 默认值：1
Bit [12]	ADC：ADC 时钟使能，	0：关闭 1：使能 默认值：1
Bit [11]	GPIO：GPIO 时钟使能，	0：关闭 1：使能 默认值：1
Bit [10]	TIMER：定时器时钟使能，	0：关闭 1：使能 默认值：1
Bit [9]	RF：内部使用，勿修改	**默认值：1**
Bit [8]	DMA：DMA 时钟使能，	0：关闭 1：使能 默认值：1
Bit [7]	LSPI：低速 SPI 时钟使能，	0：关闭 1：使能 默认值：1
Bit [6]	UART5：UART5 时钟使能，	0：关闭 1：使能 默认值：1
Bit [5]	UART4：UAR4 时钟使能，	0：关闭 1：使能 默认值：1
Bit [4]	UART3：UART3 时钟使能，	0：关闭 1：使能 默认值：1
Bit [3]	UART2：UART2 时钟使能，	0：关闭 1：使能 默认值：1
Bit [2]	UART1：UART1 时钟使能，	0：关闭 1：使能 默认值：1
Bit [1]	UART0：UART0 时钟使能，	0：关闭 1：使能 默认值：1
Bit [0]	I2C：I2C 时钟使能，	0：关闭 1：使能 默认值：1

#define RST：软件复位控制寄存器(复位后该位不会自动清除，再次使用手动置位该位)

Bit [31] TOUCH：TOUCH 控制器软件复位， 0：复位 1：复位释放
Bit [30] FLASH：FLASH 控制器软件复位， 0：复位 1：复位释放
Bit [29] BT：蓝牙控制器软件复位， 0：复位 1：复位释放
Bit [28] QSPI_RAM：QSPI_RAM 控制器软件复位 0：复位 1：复位释放
Bit [27] SDIO 主机模块复位 0：复位 1：复位释放
Bit [26] GPSEC：GPSEC 控制器软件复位， 0：复位 1：复位释放
Bit [25] RSA：RSA 模块软件复位， 0：复位 1：复位释放
Bit [24] I2S：I2S 模块软件复位， 0：复位 1：复位释放
Bit [23] LCD：LCD 软件复位， 0：复位 1：复位释放
Bit [22] PWM：PWM 软件复位， 0：复位 1：复位释放
Bit [21] ADC：ADC 软件复位， 0：复位 1：复位释放
Bit [20] TIMER：定时器模块软件复位， 0：复位 1：复位释放
Bit [19] GPIO：GPIO 模块软件复位， 0：复位 1：复位释放
Bit [18] RF：内部使用，勿修改 **默认值：1（复位释放）**
Bit [17] SSPI：高速 SPI 模块软件复位， 0：复位 1：复位释放
Bit [16] LSPI ：低速 SPI 模块软件复位， 0：复位 1：复位释放
Bit [15] UART5：UART5 模块软件复位， 0：复位 1：复位释放
Bit [14] UART4：UART4 模块软件复位， 0：复位 1：复位释放
Bit [13] UART3：UART3 模块软件复位， 0：复位 1：复位释放
Bit [12] UART2：UART2 模块软件复位， 0：复位 1：复位释放
Bit [11] UART1：UART1 模块软件复位， 0：复位 1：复位释放
Bit [10] UART0：UART0 模块软件复位， 0：复位 1：复位释放
Bit [9] I2C：I2C 模块软件复位， 0：复位 1：复位释放
Bit [8] BUS2：总线 2 模块软件复位， 0：复位 1：复位释放
Bit [7] BUS1：总线 1 模块软件复位， 0：复位 1：复位释放
Bit [6] ABP ：ABP 桥接模块软件复位， 0：复位 1：复位释放
Bit [5] MEM：内部使用，勿修改 **默认值：1（复位释放）**
Bit [4] DMA：DMA 模块软件复位， 0：复位 1：复位释放
Bit [3] SDIO：SDIO AHB 模块软件复位， 0：复位 1：复位释放
Bit [2] SEC：内部使用，勿修改 **默认值：1（复位释放）**
Bit [1] MAC：内部使用，勿修改 **默认值：1（复位释放）**
Bit [0] BBP：内部使用，勿修改 **默认值：1（复位释放）**

#define CLK_MASK：软件时钟控制寄存器

Bit [6] CPU：CPU 时钟域(CPU、BUS1、ROM、SRAM)**自适应**开启 0:开启 1:禁止 默认:1
Bit [5] TPIK_WEP：内部使用，勿修改
Bit [4] CCMP：内部使用，勿修改
Bit [3] WPI：内部使用，勿修改
Bit [1] SDIOAHB：SDIO AHB 时钟自适应开启/关闭 0:开启 1：禁止 默认:1
Bit [0] PMU：PLL 输出时钟后端 PMU 门控配置位， 0:关闭 PMU，**关断**所有时钟 1:开启

#define CLK_DIV：时钟分频配置寄存器 (数据总线时钟= Fclk_wlan/4)

Bit [31] FREQ_EN：时钟更改使能位，更改 **CPU、WLAN、ADC_DIV、BUS2_SYNCDN_FACTOR** 时钟分频系数后置位该位，数据更新后硬件清零该位。 1：使能

Bit [27:24] PERIPHERAL：PERIPHERAL (RSA、GPSEC、PWM、I2S、FLASH)时钟分频系数 N，Fclk_peri=480MHz/N，该时钟 2 分频、4 分频后提供给加密、接口模块， N 默认为 3

Bit [23:16] BUS2_SYNCDN_FACTOR：**CPU 时钟**与**数据总线**时钟频率比例满足 M：1， **M:整数**

Bit [15:8] WLAN：**WLAN 时钟**分频系数 N(N>=2)，Fclk_wlan=480MHz/N， N 默认值为 3

Bit [7:0] CPU：**CPU 系统时钟**分频系数 N(N>=2)，Fclk_cpu =480MHz/N， N 默认值为 6

注：二级总线(AHB BUS2)时钟及 APB 时钟=WLAN 时钟/4，系统时钟比值(M:1)如下：

$$\text{BUS2\_SYNCDN\_FACTOR}[23:16] = \frac{4 * \text{N\_wlan}}{N\_cpu}$$

#define CLK_SEL：时钟选择寄存器

Bit [16] JTAG：JTAG 使能 0：禁止 JTAG 调试 1：使能 默认:0

Bit [15:8] ADC_DIV：ADC 分频系数 N，Fclk_adc =40MHz/N，设定值后**置位 FREQ_EN[31]**

Bit [6] QFLASH：FLASH 模块对外总线时钟频率选择， 0：40MHz 1：80MHz 默认:0

Bit [5] GPSEC：GPSEC 加、解密模块时钟频率选择， 0：80MHz 1：160MHz 默认:0

Bit [4] RSA：RSA 加密模块时钟频率选择， 0：80MHz 1：160MHz 默认:0

Bit [3] DAC_REV： **内部使用，勿修改**

Bit [2] ADC_REV： **内部使用，勿修改**

Bit [1] TEST： **内部使用，勿修改**

Bit [0] ADC_DAC_LOOP： **内部使用，勿修改**

#define I2S_CLK：I2S 时钟控制寄存器

Bit [17:8] BCLKDIV：BCLK 分频系数 N，F_BCLK = F_I2SCLK /N；若选择**内部 PLL 时钟**(EXTAL_EN[0]=0)， **F_I2SCLK =F_CPU；外部时钟**(EXTAL_EN[0]=1)**F_I2SCLK=外部晶振频率。**

BCLKDIV (N) = F_I2SCLK/(Fs * W * F) Fs:音频采样频率，W:字宽,单声道 F = 1,立体声道 F =2。 **例**：假设 F_CPU=160MHz，使用内部 PLL 且数据宽度为 24 位，格式为立体声格式，采样频率为 128KHz，则 BCLKDIV =（160 * 10^6 / 128 * 10^3 * 24 * 2）= 10'h1a。

Bit [7:2] MCLKDIV：MCLK 分频系数 N，若 EXTAL_EN[0]=1，F_mclk = F_I2SCLK/(2 * N)。

分频系数 N = 0	I2S 时钟为外部时钟。
分频系数 N >= 1	F_mclk = F_I2SCLK，F_mclk 应配置为 256 * fs，其中 fs 是采样频率。

Bit [1] MCLK_EN：MCLK 使能位， 0：禁止 MCLK 1：使能 MCLK 默认值：0

Bit [0] EXTAL_EN：外部时钟选择 0：内部时钟 1：外部时钟 默认值：0

注：使用外部时钟时，**外部时钟必须=2 * N * 256 *fs**，(fs：采样频率、N 必须为整数)

#define RST_STATUS：复位状态寄存器

Bit [3] CPU_CLEAR：CPU 软复位状态清除位， 写 1 清零

Bit [2] WDG_CLEAR：看门狗软复位状态清除位， 写 1 清零

Bit [1] CPU：CPU 软复位状态， 0：未发生复位 1：CPU 发生软复位

Bit [0] WDG：看门狗复位状态， 0：未发生复位 1：发生 WDOG 复位

PMU 电源管理寄存器

1.PMU 模块控制芯片电源开关(控制：40MHz 起振电路、BandGap、数字 PLL、电压检测电路和数字电路 LDO)。

2.在休眠/睡眠模式下提供三种唤醒模式：**Timer 唤醒**、**RTC 唤醒**和 **WAKEUP 引脚拉高唤醒**

3.W806 芯片可选择**两种低功耗模式**：

低功耗模式	描述
Standby 模式	数字电源域的电源被关断，全芯片只有 PMU 模块工作(提供唤醒与复位功能)；全芯片功耗约 15uA 左右。电源关断后**内存中存储数据与内容将全部丢失**，唤醒后将重新载入固件，相当于重新启动。
Sleep 模式	数字电源域的电源将保留，只关断 DPLL 与晶体起振电路，切断时钟，全芯片功耗约在 1mA 左右；内存中存储数据与代码仍保留；唤醒后程序将继续运行。

唤醒模式	描述
Timer 唤醒	配置 PMU 中的 Timer0 模块，设置好休眠时间。系统进入休眠模式后，当 Timer0 计时到达休眠时间后将会唤醒系统，并产生相应 Timer 中断。系统恢复运行后需要对中断状态寄存器中相应状态位写‘1’清除中断状态,否则,再次进入休眠模式后将立即被中断唤醒。
RTC 实时时钟唤醒	配置 PMU 中 RTC 模块，设置好休眠时间。系统进入休眠模式后，当 RTC 记时到达休眠时间后将会唤醒系统，并给出相应 RTC 中断。系统恢复运行后必须对中断寄存器(IF)中相应状态位写‘1’清除中断状态，否则,下次进入休眠模式后将立即被中断唤醒
外部 IO 唤醒	PMU 检测特定 Wakeup 脚,外部控制器可通过将此 Wakeup_IO(PA0)拉高来唤醒系统，并给出相应 IO 唤醒中断。PMU 在离开休眠模式后不再检测该 IO(PA0)状态。系统恢复运行后必须对中断寄存器(IF)中相应状态位写‘1’清除中断状态，否则,再次进入休眠模式后将立即被中断唤醒。

唤醒配置	配置说明
Timer 唤醒	1.设置定时值，2.设定定时器使能 Bit 位，3.当达到定时时间后产生中断，软件通过写 IF 寄存器的 TIM0[0]来清除中断标志。
RTC 实时时钟唤醒	芯片内部集成 32K 时钟源，因环境因素造成 32K 时钟偏差，可将 32K 时钟源切换到由 40M 时钟分频得到，但当芯片进入**休眠模式时 40M 时钟将被关闭**，32KRC_CAL_EN [3]会自动清 0。待唤醒以后若固件仍要使用精准计时功能，需**重新设置 32KRC_CAL_EN[3]为 1**

4.RTC 模块正常工作时可配置为 40M 时钟分频或内部 32K 时钟，**睡眠状态时只能使用 32K** 时钟，系统被唤醒以后仍然保持使用 32K 时钟。

5.芯片正常工作时 Wakeup_IO 保持高电平,**MUC 进入低功耗模式时，Wakeup_IO 必须低电平**。

#define IF：中断状态寄存器，写 1 清零

Bit [8] SLEEP：Sleep 模式中断标志位

Bit [7] STANDBY：Standby 模式中断标志位

Bit [4] RTC：32K 时钟定时中断标志位

Bit [2] IO_WAKE：Wakeup 唤醒中断标志位

Bit [0] TIM0：Timer0 唤醒中断标志位

#define CR：PMU 电源控制寄存器

Bit [10] WAKEUP_LEVEL：WAKEUP 唤醒中断极性选择　　默认：0

WAKEUP_LEVEL[10]值	WAKEUP_LEVEL[10]=0	WAKEUP_LEVEL[10]=1
IO 电平唤醒中断	唤醒 IO 引脚为低电平时置位中断	唤醒 IO 为高电平时置位中断

Bit [9:6] WAKEUP_COUNT：按键保持时间 n 后唤醒 MCU，单位：128ms　T=n*128ms　默认 n=1

Bit [5] DLDO_CORE：DLDO_CORE 参考电压源选择　　0：ABG　1：DBG　默认值：1

Bit [4] 32KRCBYPASS：32K 由 40MHz 分频获得使能　　1：使能　　默认值：0

Bit [3] 32KRC_CAL_EN：32K RC 振荡器校准使能　0：校准电路复位状态　1:启动校准电路

注：启动 32K 振荡器校准功能**先清 0 该位，再置位该位**。

Bit [2] KSLEEP：按键触发进入**睡眠中断功能**使能　1:正常模式下 WAKEUP 引脚电平达到设定的阈值时间触发 IO_Sleep 中断，并上报 MCU。　默认：0

Bit [1] SLEEP_EN：MCU 进入 Sleep 状态使能　　0：唤醒状态　1：MCU 进入 Sleep 状态

Bit [0] STANDBY_EN：MCU 进入 Standby 状态使能　0：唤醒状态　1：进入 Standby 状态

注：**MUC 进入 Sleep/Standby 状态时，Wakeup_IO 必须低电平，**高电平则唤醒。

#define TIMER0：PMU 定时器 0 寄存器

Bit [16] EN：Timer0 使能

Bit [15:0] VALUE：Timer0 的定时值，单位：秒

5.RTC 实时时钟系统由 PMU 模块提供提供两个 32 位的 BCD 计数/定时寄存器，含秒、分、时、日、月、年进制编码的十进制格式表示。

6. RTC 模块可配置两个时钟源：40M 时钟分频和内部 32K 时钟，正常工作状态可由软件配置具体使用时钟源，睡眠状态时只能使用 32K 时钟，**唤醒后默认使用 32K 时钟**。

7.**计时功能**：配置完时间信息使能 **RTCCR1 寄存器 EN[16]=1**。

8.**定时功能**：配置完时间信息使能 **RTCCR0 寄存器 TIMING_EN[31]=1**，达到定时时间后产生 RTC 中断，对 IF 寄存器 RTC[4]写 1 清除中断标志。

9. 当系统进入睡眠模式之后，RTC 定时器产生的中断会**唤醒系统**。

#define RTCCR0：实时时钟配置寄存器 0

Bit [31] TIMING_EN：RTC **定时**中断使能　　1：使能定时功能

Bit [28:24] DATE：　日初值(日定时值)

Bit [20:16] HOUR：　小时初值(小时定时值)

Bit [13:8] MINUTE：　分钟初值(分钟定时值)

Bit [5:0] SECOND：　秒初值(秒定时值)

#define RTCCR1：实时时钟配置寄存器 1

Bit [16] EN：RTC **计时**功能使能位　　1：使能计时功能

Bit [14:8] YEAR：年初值(年定时值)

Bit [3:0]　MONTH：：月初值(月定时值)

看门狗 WDG

1. WDG 模块定时时间到会产生定时中断，若在中断中未清除 CLR 寄存器 CLR[0]位，则会周期产生定时中断。

2. 设置 LD 寄存器定时值后，启动 CR 寄存器定时(EN[0]=1)和复位(RST_EN[0]=1)功能,看门狗 VAL 寄存器启动倒计时，当 VAL 寄存器计时为 0 产生 WDG 中断。

3. **看门狗喂狗：在定时中断中写 CLR 寄存器中断状态清除位(CLR[0]),若在中断中未清除中断标志位则在下一个中断触发后产生复位信号**。

#define LD:看门狗定时配置寄存器，定时时间以 F_apb=40MHz 为基准

Bit [31:0] LD：看门狗定时值 N(定时时间 T = (N/40000000)秒)　　默认:0xFFFFFFFF

#define VAL:看门狗倒计时寄存器，读该寄存器获得**剩余时间，只读寄存器**

Bit 31:0] VAL：看门狗计数值(该值自动载入 LD 设定值并递减，减为零触发中断)

#define CR：WDG 控制寄存器

Bit [1] RST_EN：复位使能　　0:达到复位条件不产生复位信号　1:产生复位信号

Bit [0] EN：WDG 定时使能位　　0：定时器不工作　　1：定时器工作

#define CLR ：WDG 中断清除寄存器，只写寄存器

Bit [0] CLR：WDG 中断状态清除位，写任意值清除

#define SRC：WDG 中断源寄存器，只读寄存器

Bit [0] SRC：中断源标志位(WDG 定时器中断同时影响该位)

#define STATE：WDG 中断状态寄存器，只读寄存器

Bit [0] STATE：WDG 定时器中断标志位(定时器关闭后不产生该标志)

PWM 波形控制器

1.支持 **PWM0** 和 **PWM4** 两通道 PWM **输入捕捉功能**，捕捉寄存器计数可以通过 DMA 通道快速传输至内存。

2.PWM 控制器的 5 路输出通道均支持**单次输出**和**自动装载模式**：**单次**：输出指定周期数波形后不再输出；**自动装载**：自动输出指定个周期波形后自动装载继续产生 PWM 波。

3.a.支持 **5 通道独立输出模式**；

b.**双通道同步模式**：两个通道输出完全一致；

c.**5 通道同步模式**：通道 1-4 输出跟随 0 通道输出；

d.**双通道互补输出**：两个通道输出波形相反；

e.**互补模式死区设置**：死区长度最多可设置 256 个时钟周期；

f.**制动模式**：制动端口检测到指定电平后，通道输出已设置好的制动电平。

4. PWM 输出频率范围：**3Hz~160kHz**

#define CR：PWM 控制寄存器

Bit [31:27] CNTEN：CH[0-4]计数器使能

Bit 位	Bit[31]	Bit[30]	Bit[29]	Bit[28]	Bit[27]
对应通道	CH4 通道	CH3 通道	CH2 通道	CH1 通道	CH0 通道

Bit [25] CAPINV：捕捉反向使能位　0：输入信号反向无效　1：反向有效(对输入信号取反)

Bit [24] CPEN：输入捕捉使能标志位(0：CH0 输入捕捉无效　1：CH0 捕捉使能，PWM 计数器值分别存储在 CH0CAPDAT_R(上升沿锁存)和 CH0CAPDAT_F(下降沿锁存))

Bit [23:22] CNTTYPE3：CH3 计数器方式

CNTTYPE3 [23:22]值	**描述**
00	边沿对齐模式，仅针对输入捕捉模式，计数器计数方式**递减**
01	边沿对齐模式，仅针对 PWM 模式，　计数器计数方式**递减**
10	中央对齐模式，仅针对 PWM 模式

Bit [21:20] CNTTYPE2：CH2 计数器方式，同上

Bit [19:18] CNTTYPE1：CH1 计数器方式，同上

Bit [17:16] CNTTYPE0：CH0 计数器方式，同上

Bit [15:14] TWOSYNCEN：2 通道同步模式使能,　1：使能

Bit 位值	**描述**
Bit[14]=1	CH1 和 CH0 具有相同相位，相位由 CH0 控制
Bit[15]=1	CH3 和 CH2 具有相同相位，相位由 CH2 控制

Bit [12] POEN:CH0 通道 PWM 管脚输出使能位，　0：管脚输出状态　1：管脚三态

Bit [11:8] CNTMODE：PWM 循环方式选择　0：单次模式　1：自动装载模式

Bit 位	Bit[11]	Bit[10]	Bit[9]	Bit[8]
对应通道	CH3 通道	CH2 通道	CH1 通道	CH0 通道

Bit [6] ALLSYNCEN：全通道同步模式使能　1：CH[0-3]全部同步，相位由 CH0 控制

Bit [5:2] PINV：PWM 输出极性使能　0：输出极性不翻转　1：输出极性翻转

Bit 位	Bit[5]	Bit[4]	Bit[3]	Bit[2]
对应通道	CH3 通道	CH2 通道	CH1 通道	CH0 通道

Bit [1:0] 2COMPLEMENTARY：双通道输出模式　0：非互补模式　1：互补模式

Bit 位	Bit[1]	Bit[0]
对应通道	控制 CH2 和 CH3	控制 CH0 和 CH1

#define CLKDIV01：通道 0、1 时钟分频寄存器，F_chx=40MHz/N(分频系数)

Bit [31:16] CH1：CH1 分频计数器(如不需要分频输入 0 或 1)

Bit [15:0] CH0：CH0 分频计数器(如不需要分频输入 0 或 1)

#define CLKDIV23：通道 2、3 时钟分频寄存器，F_chx=40MHz/N(分频系数)

Bit [31:16] CH3：CH3 分频计数器(如不需要分频输入 0 或 1)

Bit [15:0] CH2：CH2 分频计数器(如不需要分频输入 0 或 1)

#define CMPDAT：PWM 比较寄存器，用于设置占空比

Bit [31:24] CH3：CH3 比较寄存器值

Bit [23:16] CH2：CH3 比较寄存器值

Bit [15:8] CH1：CH3 比较寄存器值

Bit [7:0] CH0：CH3 比较寄存器值

#define PERIOD：周期寄存器，用于设置占空比

Bit [31:24] CH3：CH3 周期寄存器值，不可大于等于 255

Bit [23:16] CH2：CH2 周期寄存器值，不可大于等于 255

Bit [15:8] CH1：CH1 周期寄存器值，不可大于等于 255

Bit [7:0] CH0：CH0 周期寄存器值，不可大于等于 255

CHx 模式	描述
沿对齐模式，计数器计数方式为递减	a.每个通道周期 = PERIOD_CHx +1; b.占空比 = (CMPDAT_CHx+1)/(PERIOD_CHx +1), ①当 CMPDAT_CHx >= PERIOD_CHx：输出恒定为高电平 ②当 CMPDAT_CHx < PERIOD_CHx： 低电平宽度 = PERIOD_CHx-CMPDAT_CHx 高电平宽度 = CMPDAT_CHx+1 c. CMPDAT_CHx = 0 低电平宽度 = PERIOD_CHx 高电平宽度 = 1
中间对齐模式	a.每个通道周期 = 2*(PERIOD_CHx +1); b.占空比 = (2*CMPDAT_CHx+1)/(2*(PERIOD_CHx +1)), ①当 CMPDAT_CHx > PERIOD_CHx：输出恒定为高电平 ②当 CMPDAT_CHx <= PERIOD_CHx： 低电平宽度 = 2*(PERIOD_CHx-CMPDAT_CHx)+1 高电平宽度= 2*CMPDAT_CHx+1 c. CMPDAT_CHx = 0 低电平宽度 = 2*PERIOD_CHx+1 高电平宽度 = 1
注：输出频率 F_output=40MHz/ (CLKDIVn_CHx * PNUM_CHx) CLKDIVn_CHx：CHx 通道分频系数 PNUM_CHx: CHx 通道周期数	

#define PNUM：周期数寄存器，用于配置输出 PWM 个数

Bit [31:24] CH3：PWM3 发生周期数

Bit [23:16] CH2：PWM2 发生周期数

Bit [15:8] CH1：PWM1 发生周期数

Bit [7:0] CH0：PWM0 发生周期数

注：PWM 产生 PNUM_CHx 设置的周期信号后即停止，同时触发中断置位中断标志

#define DTCR：死区控制寄存器(在通道互补模式下 CHx 由高变低(低变高)时 CHx+1 值翻转要在 CHx 值变化后产生，CHx+1 等待翻转时间为死区时间)

Bit [21] DTEN23:CH2 和 CH3 插入死区信号使能，只适用于通道**互补模式**

Bit [20] DTEN01:CH0 和 CH1 插入死区信号使能，只适用于通道**互补模式**

Bit [17:16] DTDIV：死区时钟分频系数 N，死区时钟频率 **F_DT** = 40MHz/N

DTDIV[17:16]值	0x00	0x01	0x02	0x03
分频系数	1	2	4	8

Bit [15:8] DTCNT23:CH2 和 CH3 死区间隔值 N，间隔时间 T=F_DT*N

Bit [7:0] DTCNT01:CH0 和 CH1 死区间隔值 N，间隔时间 T=F_DT*N

#define CH0CAPDAT：CH0 输入捕捉寄存器，只读寄存器

Bit [31:16] CH0CAPDAT_F：捕捉到输入信号下降沿时，计数器值存入当前位(下降沿计数)

Bit [15:0] CH0CAPDAT_R：捕捉到输入信号上升沿时，计数器值存入当前位(上升沿计数)

#define IE：中断控制寄存器

Bit [7] DMAEN：DMA 请求使能 1：使能

Bit [6] FLIEN：下降沿缓存中断使能位（仅适用 CH0） 1：使能

Bit [5] RLIEN：上升沿缓存中断使能位（仅适用 CH0） 1：使能

Bit [4:0] PIEN：PWM 周期中断使能位，分别对应 CH[4-0]

Bit 位	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]
对应通道位	CH4	CH3	CH2	CH1	CH0

#define IF：中断状态寄存器

Bit [9] OVERFL：计数器溢出标志位（捕捉模式计数器溢出） 1:计数器溢出

Bit [8] FLIFOV：下降沿延迟中断标志位(清除 CFLIF[6]该位清除) 1：下降沿延迟中断

Bit [7] RLIFOV：上升沿延迟中断标志位(清除 CRLIF[5]该位清除) 1：上升沿延迟中断

Bit [6] CFLIF：CH0 下降沿捕捉中断标志位 1：捕获到下降沿，写 1 清零

Bit [5] CRLIF：CH0 上升沿捕捉中断标志位 1：捕获到上升沿，写 1 清零

Bit [4:0] PIF:周期中断标志位(产生指定周期 PWM 信号后，该标识位置 1，写 1 清零)

PIF[4:0]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]
对应通道位	CH4	CH3	CH2	CH1	CH0

#define BKCR：制动控制寄存器

Bit [15:11] EN：制动模式使能位，分别对应 CH[4-0] 0：禁止 1：使能

Bit 位	Bit[15]	Bit[14]	Bit[13]	Bit[12]	Bit[11]
对应通道位	CH4	CH3	CH2	CH1	CH0

Bit [7:3] OD:制动输出控制位 0:制动有效 PWM 输出**低电平** 1:制动有效 PWM 输出**高电平**

Bit 位	Bit[7]	Bit[6]	Bit[5]	Bit[4]	Bit[3]
对应通道位	CH4	CH3	CH2	CH1	CH0

#define CH4CR1:CH4 时钟分频寄存器

Bit [31:16] DIV：CH4 分频值(不需要分频输入 0 或 1),取值 0-65535

Bit [15:8] PRD:CH4 周期寄存器值，用于设置占空比

Bit [7:0] PNUM：CH4 生成 PWM 周期数

#define CH4CR3：CH4 通道控制寄存器

Bit [10] DMAEN：CH4 通道 DMA 请求使能位　　1：使能

Bit [9] FLIEN：CH4 下降沿缓存中断使能位　　1：使能

Bit [8] RLIEN：CH4 上升沿缓存中断使能位　　1：使能

Bit [7] OVERFL：CH4 计数器溢出标志(清除 CFLIF[4]、CRLIF[3]位该位清零)

1:输入捕捉模式中计数器溢出

Bit [6] FLIOV：CH4 下降沿延迟中断标志位(清除 CFLIF[4]位该位清零)　1:发生延迟中断

Bit [5] RLIOV：CH4 上升沿延迟中断标志位(清除 CRLIF[3]位该位清零)　1:发生延迟中断

Bit [4] CFLIF：CH4 下降沿输入捕捉中断标志位　　1：捕捉到下降沿，写 1 清零

Bit [3] CRLIF：CH4 上升沿输入捕捉中断标志位　　1：捕捉到上升沿，写 1 清零

Bit [2] POEN：CH4 引脚 PWM 输出使能位　　0：引脚输出　1：引脚为三态状态

Bit [1] CPEN：CH4 输入捕捉使能位　　1：CH4 捕捉使能，PWM 计数器值分别存储在 RCAPDAT(上升沿锁存)和 FCAPDAT(下降沿锁存))

Bit [0] CAPINV：CH4 输入捕捉信号取反使能位　0:不取反　　1:对输入信号取反

#define CH4CR2：CH4 控制寄存器

Bit [15:8] CMP：CH4 周期寄存器值，用于设置占空比

CHx 模式	描述
沿对齐模式，计数器计数方式为递减	a.每个通道周期 = **CH4CR1 寄存器** PRD[15:8]+1; b.占空比 = (CMP[15:8]+1)/(PRD[15:8] +1), ①当 CMP[15:8] >= PRD[15:8]：输出恒定为高电平 ②当 CMP[15:8] < PRD[15:8]： 低电平宽度 = PRD[15:8]- CMP[15:8] 高电平宽度 = CMP[15:8]+1 c. CMP[15:8] = 0 低电平宽度 = PRD[15:8]　高电平宽度 = 1
中间对齐模式	a.每个通道周期 = 2*(PRD[15:8] +1); b.占空比 = (2* CMP[15:8]+1)/(2*(PRD[15:8] +1)), ①当 CMP[15:8] > PRD[15:8]：输出恒定为高电平 ②当 CMP[15:8] <= PRD[15:8]： 低电平宽度 = 2*(PRD[15:8]- CMP[15:8])+1 高电平宽度= 2* CMP[15:8]+1 c. CMP[15:8] = 0　低电平宽度 = 2* PRD[15:8]+1　高电平宽度 = 1
注：输出频率 F_output=40MHz/ (CH4CR1_DIV[31:16] * CH4CR1_PNUM[7:0])	

Bit [4:3] CNTTYPE：CH4 计数器计数方式

CNTTYPE [4:3]值	描述
00	边沿对齐模式，仅针对输入捕捉模式，计数器计数方式**递减**
01	边沿对齐模式，仅针对 PWM 模式，　计数器计数方式**递减**
10	中央对齐模式，仅针对 PWM 模式

Bit [1]　CNTMODE：CH4 PWM 循环方式选择　0：单次模式　　1：自动装载模式

Bit [0]　PINV：CH4 PWM 输出极性使能　　0：输出极性不翻转　1：输出极性翻转

#define CH4CAPDAT：CH4 输入捕捉寄存器，只读寄存器

Bit [31:16] FCAPDAT：CH4 捕捉到输入信号下降沿时，计数器值存入当前位(下降沿计数)

Bit [15:0]　RCAPDAT：CH4 捕捉到输入信号上升沿时，计数器值存入当前位(上升沿计数)

GPIO 寄存器

1.GPIOA 为 16 位，引脚数量 GPIOA-[15:0]；　　GPIOB 为 32 位，引脚数量 GPIOB-[31:0]。

#define　DATA：数据寄存器，用于读、写 GPIOA/B 当前数据

Bit　GPIOA[15:0]/GPIOB[31:0]

#define　DATA_B_EN：数据使能寄存器，用于配置 GPIOx_DATA 的使能位

Bit　DATA_EN[n]=0 ： 对应 GPIOx_DATA[n]位不可进行读、写操作

Bit　DATA_EN[n]=1 ： 对应 GPIOx_DATA[n]位可进行读、写操作

#define　DIR：GPIO 方向控制寄存器，用于配置 GPIO 输入或输出

Bit　DIR[n]=0 ： 对应 GPIOx[n]引脚为输入

Bit　DIR[n]=1 ： 对应 GPIOx[n]引脚为输出

#define　PULLUP_EN：GPIO 上拉控制寄存器，用于配置上拉，低位有效

Bit　PULLUP_EN[n] = 0 : GPIOx[n]位上拉有效

Bit　PULLUP_EN[n] = 1 : GPIOx[n]位上拉无效

#define　PULLDOWN_EN：GPIO 下拉控制寄存器，用于配置下拉，高位有效

Bit　PULLDOWN_ENA[n] = 1 ： 下拉有效

Bit　PULLDOWN_ENA[n] = 0 ： 下拉无效

#define　AF_SEL:GPIO 复用功能选择寄存器，用于配置 GPIO 复用功能，AF_S1 和 AF_S0 两个寄存器相应 BIT 位决定复用功能。

Bit　AF_SEL[n] = 0 ： 复用功能关闭

Bit　AF_SEL[n] = 1 ： 复用功能打开，

注:AF_SEL[n]=0 时，若 DIR[n] =0 且 PULLUP_EN[n] =1，则 GPIO 复用为 opt6 模拟 IO 功能

#define　AF_S1 ： GPIO 复用功能选择寄存器 1，配置高地址位复用功能

#define　AF_S0 ： GPIO 复用功能选择寄存器 0，配置低地址位复用功能

序号	AF_SEL[n]	AF_S1[n]值	AF_S0[n]值	复用功能描述
0	1	0	0	复用功能 1(opt1)
1	1	0	1	复用功能 2(opt2)
2	1	1	0	复用功能 3(opt3)
3	1	1	1	复用功能 4(opt4)

#define　IS ： GPIO 中断触发方式配置寄存器，用于配置中断触发方式

Bit　IS[n] = 0 :GPIOx[n]中断为边沿触发

Bit　IS[n] = 1 :GPIOx[n]中断为电平触发

#define　IBE ： GPIO 中断边沿触发模式配置寄存器

Bit　IBE[n] =0 ： GPIOx[n]边沿触发中断模式由 GPIOx_IEV 寄存器决定

Bit　IBE[n] =1 ： 上升、下降沿均可触发中断

#define　IEV ： GPIO 中断上、下边沿触发配置寄存器

Bit　IEV[n] =0 ： GPIOx[n]引脚中断为低电平或下降沿触发

Bit　IEV[n] =1 ： GPIOx[n]引脚中断为高电平或上升沿触发

#define IE ：GPIO 中断使能配置寄存器

Bit IE[n] =0 ：中断关闭

Bit IE[n] =1 ：中断使能

#define RIS ：GPIO 中断状态寄存器，查询 GPIO 中断状态

Bit RIS[n] =0 ：没有中断产生

Bit RIS[n] =1 ：有中断产生

#define MIS ：GPIO 屏蔽后中断状态寄存器，查询屏蔽中断状态

Bit MIS[n] =0 ：没有中断产生

Bit MIS[n] =1 ：有中断产生

#define IC ：GPIO 中断清除控制寄存器，清除 GPIO 中断标志位

Bit IC[n] =0 ：无操作

Bit IC[n] =1 ：清除 GPIOx[n]对应中断状态位

SPI 寄存器

1.SPI 最高速度 **20MHz,从设备输入信号**时钟应该小于 **F_apb/6=(40MHz/6)**;FIFO 寄存器集成两个(接收、发送)各 8 个字(32 字节)的存储器。

3. FIFO 存储器是一个**先入先出**的双口缓冲器，即第一个进入其内的数据第一个被移出。

#define CH_CFG：SPI 配置寄存器

Bit [30:23] RXINVALIDBIT:接收 N 个无效数据 Bit，该数据不进入 Rx FIFO

实际存入 Rx FIFO 的数据量为 LEN[18:3]－N

Bit [22] CLEARFIFOS:清除发送/接收 FIFO 数据使能　　1：清除,清除后硬件清零该位

Bit [21] CONTINUEMODE:连续发送模式使能位

CONTINUEMODE[21]=0	Tx FIFO 空后，SCK 停止，需要等待 FIFO 中装入数据； Rx FIFO 满后，SCK 停止，等待 Rx FIFO 有空间再接收数据
CONTINUEMODE[21]=1	Tx FIFO 空后，仍可以传输，直至传输完成(**为防止传输无效数据，一般不使用此模式**)；Rx FIFO 满后，SCK 停止，等待 Rx FIFO 有空间再接收数据

Bit [20] RXON：接收使能位

Bit [19] TXON：发送使能位

Bit [18:3] LEN：发送或接收数据的长度 N，单位:Bit

Bit [2] CSLEVEL:配置 CS 引脚状态　　0：CS 引脚低电平　1：CS 引脚高电平

Bit [1] CSSEL：CS 片选信号使能配置　0：硬件配置　　1：软件配置(CSLEVEL[2]配置)

Bit [0] START：SPI 操作使能位　　1：**启动一次**接收或发送，置位后硬件自动清 0

#define CLK_CFG:SPI 时钟频率设置， F_clk = 40MHz / (2 * (div + 1))

Bit [15:0] DIV：分频系数，软件取值(0、1、3、4、9、19)

#define MODE_CFG:模式配置寄存器，Rx/Tx Buffer 中数据数量大于设置 N 值触发中断/DMA

Bit [8:6] RXLEVEL:Rx FIFO 数据数量触发中断或 DMA 请求阈值 N，取值：0-7 字,**RxFIFO>N**

Bit [4:2]TXLEVEL:Tx FIFO 数据数量触发中断或 DMA 请求阈值 N，取值：0-7 字,**TxFIFO>=N**

Bit [1] RXDMA：采用 DMA 搬移数据使能

Bit [0] TXDMA：采用 DMA 搬移数据使能

#define INT_MASK：中断控制寄存器

Bit [7] TIMEOUT：SPI 超时使能位　　0:使能　1：禁止　默认：1

Bit [6] DONE：发送/接收完成中断使能　　0:使能　1：禁止　默认：1

Bit [5] RXOV：Rx FIFO 向上溢出中断使能(overflow)　0:使能　1：禁止　默认：1

Bit [4] RXUN：Rx FIFO 向下溢出中断使能(underflow)　0:使能　1：禁止　默认：1

Bit [3] TXOV：Tx FIFO 向上溢出中断使能(overflow)　0:使能　1：禁止　默认：1

Bit [2] TXUN：Tx FIFO 向下溢出中断使能(underflow)　0:使能　1：禁止　默认：1

Bit [1] RXRDY：Rx FIFO 有数据上传中断使能　　0:使能　1：禁止　默认：1

Bit [0] TXRDY：向 Tx FIFO 写数据中断使能　　0:使能　1：禁止　默认：1

注：overflow：由最大值发生上溢变最小　　**underflow:**由最小值发生下溢变最大

#define SPI_CFG: SPI 配置寄存器，配置 SPI 传输模式和大小端模式

Bit [18:17]：FORMAT：选择主机模式下 SPI 支持哪个厂家的协议

FORMAT[18:17]值	00	01	10	11
支持厂家	MOTO(摩托罗拉)	TI(德州仪器)	microwire	RSV(保留)

Bit [16] PINOUT:输出配置　0：只有 CS 片选有效时 SPI 输出　1：SPI 输出一直有效

Bit [14:12] CSHOLD:CS 片选信号在传输完成后持续 N 时间，N>=x 个 APB 总线时钟(40MHz)

CSHOLD[14:12]值	000	001	010	011	100	101	110	111
时钟 CLK	1	2	4	8	16	32	64	127

Bit [11:9] CSSETUP：CS 片选信号在数据传输前提前 N 个时间有效，N>=x 个 APB 总线时钟

CSSETUP[11:9]值	000	001	010	011	100	101	110	111
时钟 CLK	1	2	4	8	16	32	64	127

Bit [8:7] OUTDELAY：SPI 数据输出相对于 SCK 信号的延迟 N 时间，N = APB 总线时钟数

OUTDELAY[8:7]值	00	01	10	11
时钟 CLK	0	2	4	8

Bit [6:4] FRAMEDELAY：一帧传输完成(CS=1)到下一帧开始(CS=0)间 CS=1 的时间为 N 个 SCK

FRAMEDELAY[6:4]值	000	001	010	011	100	101	110	111
描述	0(默认 0.5 个 SCK)	2	4	8	16	32	64	127

Bit [3] BIGENDIAN:大、小端模式使能位　0:小端(先发送低字节)　1:大端(先发送高字节)

Bit [2] MASTER：主机模式使能位　默认：1

Bit [1] CPHA：时钟相位　0：前沿采样，后沿移出　1：后沿采样，前沿移出

Bit [0] CPOL：时钟极性　0：SCLK 空闲状态低电平　1：SCLK 空闲状态高电平

#define INT_SRC：中断状态寄存器，写 1 清零该位

Bit [7] TIMEOUT：超时标志位　0：RxFIFO 中无新接收数据　1：RxFIFO 中有新接收数据

Bit [6] DONE：发送/接收完成标志位　0：未完成　1：发送、接收未完成，写 1 清零

Bit [5] RXOV：Rx FIFO 向上溢出中断标志位(overflow)　1：溢出，写 1 清零

Bit [4] RXUN：Rx FIFO 向下溢出中断标志位(underflow)　1：溢出，写 1 清零

Bit [3] TXOV：Tx FIFO 向上溢出中断标志位(overflow)　1：溢出，写 1 清零

Bit [2] TXUN：Tx FIFO 向下溢出中断标志位(underflow)　1：溢出，写 1 清零

Bit [1] RXRDY:RxFIFO 数据标志位　0:未满(RxFIFO 数据<RXLEVEL[8:6])　1:满(需要读取)

Bit [0] TXRDY:TxFIFO 数据标志位 0:禁止写(TxFIFO 数据>TXLEVEL[4:2]) 1:可以写 TxFIFO

#define STATUS：SPI 状态寄存器，只读寄存器

Bit [12] BUSY：SPI 忙标志位　0：没接收/发送任务　1：处于发送或接收过程

Bit [11:6] RXFIFO：RxFIFO 中数据数量(单位字节)

Bit [5:0] TXFIFO：TxFIFO 中数据数量(单位字节)

#define TIMEOUT:SPI 超时寄存器

Bit [31] TIMEREN：SPI Timer 计时使能　1：使能计时　默认：0

Bit [30:0] TIMERVAL：TIMER 值 N，所表示时间 T = N/40MHz

注：当 RxFIFO 处于空闲状态(**没有读、写操作，没有 DMA 请求，CS 无效**，RxFIFO 接收数据量<RXLEVEL[8:6]设定值)开始计数，达到 TIMERVAL[30:0]设定值触发 Timerout 中断，请求 CPU 读取数据，**任何对 RxFIFO 的读、写操作都会清零 TIMERVAL[30:0]计时器值。**

#define TRANS_MODE：SPI 传输模式配置寄存器

Bit [29:24] BLKLEN：在 TI(德州仪器)模式下每个块的数据长度，即 CS=0 时传输数据长度

BLKLEN[29:24]值	0x04	0x05	……	0x1F	0x20
数据长度(取值：4-32Bit)	4bit	5bit	……	31bit	32bit

Bit [16] MICROBURST：Microwire 模式配置 0：不采用 Burst 传输 1：采用 Burst 传输

Burst 传输模式		Tx 发送字，Rx 接收数据交替进行，MICROCONTROLLEN[5:0] ①：控制字长度； MICRODATLEN[13:8] ②：发送或接收字的长度；寄存器 CH_CFG 中 LEN[18:3] ③：传输过程中的有效 SCK；发送、接收交替进行的次数 N = ③/(①+②+1)
非 Burst 模式	只发送	发送数据长度②*m=(③ - ①) m:发送多少个②设定长度的字
	发送+接收	接收数据长度②*m=(③-① -1) m: 接收多少个②设定长度的字

Bit [13:8] MICRODATLEN：在 Microwire Burst 模式下每个 Burst 传输数据长度

MICRODATLEN[13:8]值	0x01	0x02	……	0x1F	0x20
数据长度(取值：1-32Bit)	1bit	2bit	……	31bit	32bit

Bit [5:0] MICROCONTROLLEN：在 Microwire Burst 模式下命令字的长度

MICROCONTROLLEN[5:0]值	0x01	0x02	……	0x1F	0x20
字长度(取值：1-32Bit)	1bit	2bit	……	31bit	32bit

#define SLV_LEN:从机数据长度寄存器

Bit [31:16] TXLEN：作为从机在 CS 有效(CS=0)期间发送出去数据长度，单位 Bit

Bit [15:0] RXLEN：作为从机在 CS 有效(CS=0)期间接收到的数据长度，单位 Bit

#define TXDATA：发送数据寄存器

Bit [31:0] TXDATA

#define RXDATA：接收数据寄存器

Bit [31:0] RXDATA

I2C 寄存器

1.I2C 工作频率可配置 100KHz-400KHz

#define PRESCALE_L：I2C 分频系数低位寄存器

Bit [7:0] PRESCALE_L：分频值

#define PRESCALE_H：I2C 分频系数高位寄存器，对 F_apb 时钟分频

Bit [7:0] PRESCALE_H：分频值

注：I2C 时钟由 APB_clk=40MHz 分频得到，

$$PRESCALE = \frac{40*1000}{5*F_{I2C}} - 1 \quad (KHz)$$

F_{I2C}：I2C 通信频率

#define EN:I2C 控制寄存器

Bit [7] ENABLE：I2C 使能位　　0：不使能　　1：使能　　默认：0

Bit [6] IEMASK：I2C 中断使能位　0：使能　　1：不使能　默认：1

#define DATA:I2C 数据寄存器

Bit [7:0] DATA：I2C 数据位

寄存器操作	**描述**	写地址描述
写寄存器	写发送寄存器 TXR	写设备地址时，DATA[0]=0,表示**写地址**
读寄存器	读接收寄存器 RXR	DATA[0]=1,表示**读地址**

#define CR_SR：I2C 收、发控制寄存器，该寄存器具有双重功能，具体如下：

1.**写该寄存器时，为 CR 寄存器，**功能如下：

Bit [7] START：I2C 起始使能位　　1：产生 Start 信号

Bit [6] STOP：I2C 停止使能位　　1：产生 Stop 信号

Bit [5] RD：读从机数据使能位　　1：读从机数据

Bit [4] WR：写从机数据使能位　　1：向从机写数据

Bit [3] ACK：向从机发送 ACK 回应值　0：回 ACK　　1：回 NACK

Bit [0] IF：清除中断标志位，写 1 清除

2.**读该寄存器时，为 SR 寄存器，**功能如下：

Bit [7] RXACK：从机应答信号位　　0：从机应答 ACK　　1：从机 NACK

Bit [6] BUSY：忙标志位　　执行 STOP 后硬件清零该位，执行 START 后硬件该位置位

Bit [1] TIP：传输状态　　0：无传输　　1：正在传输

Bit [0] IF：中断状态位　　0：无中断　　1：传输完成产生中断

UARTx 寄存器(x=[0-5])

1.奇校验：数据位中 1 的个数是奇数，校验位为 0，是偶数个，验位为 1。

2.偶校验：数据位中 1 的个数是奇数，校验位为 1，是偶数个，验位为 0。

3. **硬件流控制**：W806 UART 支持 RTS/CTS 硬件流控制，目的是防止 UART FIFO 里的数据因软件来不及处理而造成丢失。

1. **连接方式**：主设备 RTS(CTS)引脚对应连接 CTS(RTS)引脚

2. **工作方式**：a.当 FC 寄存器中 AFCE[0]置位时，若 RxFIFO 里接收数据个数小于 RTSL[4:2]设定数量时 RTS 引脚拉低，从设备向主设备传送数据；

b.当 RxFIFO 里接收数据个数大于 RTSL[4:2]设定数量时 RTS 引脚拉高，从设备停止向主设备传送数据。

c.当 AFCE[0]为 0 时，软件通过置位 RTSS[1]位来拉高 RTS 引脚。

#define LC:数据流控制寄存器

Bit [7] RE：接收使能　　1：使能

Bit [6] TE：发送使能　　1：使能

Bit [5] SBE：发送 Break 数据包使能，置 1 后 UART 将发送一个 Break 数据包，发送完清零

Bit [4] PS：奇偶校验配置　　0：偶校验　　1：奇校验

Bit [3] PCE：奇偶校验使能位　　1：使能

Bit [2] STOP：停止位长度　　0:1 个停止位　　1:2 个停止位

Bit [1:0] DATAL：传输数据位长度选择，(不包含起始位和结束位)

DATAL[1:0]值	0x00	0x01	0x02	0x03
数据长度	5 位	6 位	7 位	8 位

#define FC:硬件流控制寄存器(防止 FIFO 里的数据因软件处理不及时造成丢失)

Bit [4:2] RTSL：硬件流(AFCE[0]=1 后，Rx FIFO 存入数据>N **字节**将 RTS 置无效位(即拉高))

RTSL[4:2]	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
字节数	4 Byte	8 Byte	12 Byte	16 Byte	20 Byte	24 Byte	28 Byte	31 Byte

Bit [1] RTSS：RTS 引脚软控制位　仅限 **AFCE[0]=0** 时，置 1 该位拉高 RTS 引脚，0:RTS 低

Bit [0] AFCE：硬件流控制使能位　　1：使能，使用 RTS[4:2]配置值控制 RTS 引脚

#define DMAC:DMA 配置寄存器

Bit [7:3] RTO：RxFIFO 接收超时时间 n(RxFIFO 中数据个数小于 RFL[5:4]设定数，经过 n 个数据包时间仍未接收新数据,则产生接收超时中断;在接收到**至少 1 个数据**后才开始计时)

Bit [2] RTOE：接收数据超时使能位　　0：使能　　默认：1

Bit [1] RDE：接收 DMA 使能位　　0：接收过程使用中断　　1：接收 DMA 使能

Bit [0] TDE：发送 DMA 使能位　　0：接收过程使用中断　　1：接收 DMA 使能

#define BAUDR:波特率寄存器，　　BAUDR=((DIVF<<16)|DIV)

Bit [19:16] DIVF：波特率小数值　$\mathrm{DIVF}=\dfrac{(40*10^6\%(baud*16))}{baud}$　取整

Bit [15:0] DIV：波特率整数值　$\mathrm{DIV}=\dfrac{40*10^6}{16*\mathrm{baud}}-1$　　**baud**：传输速率(9600...)

#define FIFOC:FIFO 控制寄存器

Bit [5:4] RFL：接收中断触发选择，当 Rx FIFO 中数据字节>=RFL[5:4]设置数量触发中断或触发 RXDMA

RFL[5:4]值	0x00	0x01	0x02	0x03
字节	1Byte	4 Byte	8 Byte	16 Byte

Bit [3:2] TFL：发送中断触发选择，当 TXFIFO 中数据字节<=TFL[3:2]设置数量触发中断或触发 TXDMA

TFL[3:2]值	0x00	0x01	0x02	0x03
字节	0 Byte	4 Byte	8 Byte	16 Byte

Bit [1] RFRST：清除 Rx FIFO　1：清除

Bit [0] TFRST：清除 Tx FIFO　1：清除

#define INTM:中断屏蔽寄存器

Bit [8] OE：RxFIFO 数据溢出中断屏蔽位　1：屏蔽　默认值：1

Bit [7] PE：奇偶校验中断屏蔽位　1：屏蔽　默认值：1

Bit [6] FE：数据帧错误中断屏蔽位　1：屏蔽　默认值：1

Bit [5] BD：Break 信号检测中断屏蔽位　1：屏蔽　默认值：1

Bit [4] CTS：CTS 信号变化中断屏蔽位　1：屏蔽　默认值：1

Bit [3] RTO：RxFIFO 数据超时中断屏蔽位　1：屏蔽　默认值：1

Bit [2] RL：RxFIFO 数据达到触发中断值屏蔽位　1：屏蔽　默认值：1

Bit [1] TL：TxFIFO 数据达到触发中断值屏蔽位　1：屏蔽　默认值：1

Bit [0] TEMPT：TxFIFO 数据空中断屏蔽位　1：屏蔽　默认值：1

#define INTS:中断状态寄存器,　　各标志位置 1 有效，写 1 清零

Bit [8] OE：接收数据溢出标志位

Bit [7] PE：校验错误标志位　　(DMA 情况下此中断仍会发生，DMA 操作不关心此中断)

Bit [6] FE：接收数据停止位错误标志位　(DMA 情况下此中断仍会发生，DMA 操作不关心)

Bit [5] BD：接收到 Break 数据包　(DMA 情况下此中断仍会发生，DMA 操作不关心此中断)

Bit [4] CTS：CTS 信号变化标志位

Bit [3] RTO：接收数据超时中断标志位

Bit [2] RL：实际接收数据>=RFL[5:4]配置个数中断标志位

Bit [1] TL：发送数据<=TFL[3:2]配置个数中断标志位

Bit [0] TEMPT：当前数据发送完成且 TXFIFO 空中断标志位

#define FIFOS:FIFO 状态寄存器

Bit [12] CTSS：当前 CTS 信号状态

Bit [11:6] RFC：RXFIFO 中数据个数

Bit [5:0]　TFC：TXFIFO 中数据个数

#define TDW：UART 发送数据寄存器，**只写**

Bit [31:0] TDW：数据值，**UART 发送数据只支持字节操作**

#define RDW：UART 接收数据寄存器，**只读**

Bit [31:0] RDW：数据值，**UART 发送数据只支持字节操作**

注：采用 burst 传输时，可使用字节地址递增的方式，设计中最多支持 16-burst，即 16Byte。

TIM 定时器

1. W806 拥有 6 路定时、计数器，包含一个 32bit 自动加载的计数器。
2. 定时器以 APB 总线(数据总线)时钟为基准，定时单位可配置为 ms、us。

#define TMR_CONFIG：标准 us 配置寄存器，APBclk 时钟频率为 40MHz(1s=10^3ms=10^6us)
Bit [6:0] DIV:标准 1us 配置位，(固定配置为：DIV=40-1=39)

#define CR：定时器控制寄存器

Bit [29] TIM5_TIF：Timer5 中断标志位	0：无中断发生	1：发生中断，写 1 清零
Bit [28] TIM5_TIE：Timer5 定时中断使能位	0：不使能	1：定时中断使能
Bit [27] TIM5_EN：Timer5 定时器使能位	0：不使能	1：定时器使能
Bit [26] TIM5_MODE：Timer5 工作模式	0：重复定时	1：只定时一次
Bit [25] TIM5_UNIT：Timer5 定时单位	0：us 模式	1：ms 模式
Bit [24] TIM4_TIF：Timer4 中断标志位	0：无中断发生	1：发生中断，写 1 清零
Bit [23] TIM4_TIE：Timer4 中断使能位	0：不使能	1：定时中断使能
Bit [22] TIM4_EN：Timer4 定时器使能位	0：不使能	1：定时器使能
Bit [21] TIM4_MODE：Timer4 工作模式	0：重复定时	1：只定时一次
Bit [20] TIM4_UNIT：Timer4 定时单位	0：us 模式	1：ms 模式
Bit [19] TIM3_TIF：Timer3 中断标志位	0：无中断发生	1：发生中断，写 1 清零
Bit [18] TIM3_TIE：Timer3 中断使能位	0：不使能	1：定时中断使能
Bit [17] TIM3_EN：Timer3 定时器使能位	0：不使能	1：定时器使能
Bit [16] TIM3_MODE：Timer3 工作模式	0：重复定时	1：只定时一次
Bit [15] TIM3_UNIT：Timer3 定时单位	0：us 模式	1：ms 模式
Bit [14] TIM2_TIF：Timer2 中断标志位	0：无中断发生	1：发生中断，写 1 清零
Bit [13] TIM2_TIE：Timer2 中断使能位	0：不使能	1：定时中断使能
Bit [12] TIM2_EN：Timer2 定时器使能位	0：不使能	1：定时器使能
Bit [11] TIM2_MODE：Timer2 工作模式	0：重复定时	1：只定时一次
Bit [10] TIM2_UNIT：Timer2 定时单位	0：us 模式	1：ms 模式
Bit [9] TIM1_TIF：Timer1 中断标志位	0：无中断发生	1：发生中断，写 1 清零
Bit [8] TIM1_TIE：Timer1 中断使能位	0：不使能	1：定时中断使能
Bit [7] TIM1_EN：Timer1 定时器使能位	0：不使能	1：定时器使能
Bit [6] TIM1_MODE：Timer1 工作模式	0：重复定时	1：只定时一次
Bit [5] TIM1_UNIT：Timer1 定时单位	0：us 模式	1：ms 模式
Bit [4] TIM0_TIF：Timer0 中断标志位	0：无中断发生	1：发生中断，写 1 清零
Bit [3] TIM0_TIE：Timer0 中断使能位	0：不使能	1：定时中断使能
Bit [2] TIM0_EN：Timer0 定时器使能位	0：不使能	1：定时器使能
Bit [1] TIM0_MODE：Timer0 工作模式	0：重复定时	1：只定时一次
Bit [0] TIM0_UNIT：Timer0 定时单位	0：us 模式	1：ms 模式

注意：定时器中断中需手动清零中断标志位，硬件不清零。

#define TIMx_PRD：定时器 x 定时配置寄存器，x =[0-5]
Bit [31:0] TIMx_PRD：当 CNTx 计数值等于 PRDx 值产生中断

#define TIMx_CNT:定时器 x 当前计数值(向上计数)，x =[0-5]，只读寄存器
Bit [31:0] TIMx_CNT:定时器 x 计数值,**关闭 TIMERx 后读取 TIMx_CNT 则清零 TIMx_CNT**

ADC 模数控制寄存器

1. W806 芯片 ADC 为采样精度为 16bit，最多支持 4 路模拟信号采集，最高采样速度 **1KHz**
2. ADC 芯片内部供电电压为 2.5V,外部输入电压 Vmax **小于 2.5V**,ADC 工作频率配置为 **1MHZ**。
3. 单端输入模式下运放**负极输入端 VCmin** 说明：默认状态 VCmin =1.311V(Step =39.1mV)。

#define DR:ADC 转换结果寄存器，有效位数 Bit[17:0]，只读寄存

Bit [17:0]：ADC 转换结果**(注：**转换结果为 18bit 数据位，ADC 对输出码进行了处理(最高位 DR Bit [17]取反),读该寄存器值后需将 DR Bit[17]取反,因 ADC 精度为 16bit，最低两位 Bit[1:0]为滤波位，转换结果为 Bit[17:2])

#define ANA_CR:ADC 模拟寄存器

Bit [11:8] CH：ADC 通道选择

CH[11:8]值	0x00	0x01	0x02	0x03	0x08	0x09	0x0C	0x0D	0x0E
功能描述	通道 0	通道 1	通道 2	通道 3	AIN0+/AIN1-差分输入	AIN2+/AIN3-差分输入	温度监测	电压检测	偏移校准

Bit [6:5] CHOPENR：LDO 斩波信号 PD 信号选择

Bit [4] CHOPENS：斩波信号 PD 信号使能位　　0：使能　　1：不使能

Bit [2] PD：adc 模拟模块掉电使能　　0：工作状态　　1：掉电使能

Bit [1] RST：模拟模块数字逻辑电路复位信号　　0：复位　　1：正常工作

Bit [0] LDOEN：ADC LDO 使能位　　0：掉电状态　　1：工作

#define PGA_CR:PGA 配置寄存器

Bit [8:4] CAIN:增益控制(bit[8:7]=**GAIN2**　bit[6:4]=**GAIN1**　总增益=**GAIN2 * GAIN2**)

Bit [3] BP：Bypass_PGA 信号旁路使能位　0：不旁路(正常使用)　1：旁路 PGA(测试)

Bit [2] BPREF：内部参考电压旁路使能位　0：不旁路　1：旁路(参考电压由测试引脚添加)

Bit [1] CHOPEN：PGA 斩波信号使能位　　0：不使能　　1：使能

Bit [0] PGAEN：PGA 信号使能位　　0：不使能　　1：使能

Bit[8:7] / Bit[6:4]	0x00(00)	0x01(01)	0x02(10)	0x03(11)
0x00(000)	1	2	3	4
0x01(001)	16	32	48	64
0x02(010)	32	64	96	128
0x03(011)	64	128	192	256
0x04(100)	128	256	384	512
0x05(101)	256	512	768	1024

注：同样放大倍数优先采用 GAIN2 放大倍数较大配置

#define TEMP_CR:TEMP 配置寄存器

Bit [5:4] GAIN：温度增益控制位

GAIN[5:4]	0x00	0x01	0x02	0x03
温度增益	2	4	6	8

Bit [1] OFFSET：偏移校准使能　　1：使能

Bit [0] EN：TEMP 使能位　　1：使能

#define ADC_CR:ADC 功能配置寄存器

Bit [29:20] SWITCHTIME：软件切换数据通道，模拟电路等待稳定时间 T= n/40MHz，默认:80

Bit [17:8] INITTIME:软件启动 ADC 后，模拟电路等待稳定时间，T= n/40MHz， 默认:80

Bit [6] CMPPOL：ADC 数据比较设置位(0：ADC 值>=**CMP_VAL** 值 **IF** CMP[1]置位并产生中断

1：ADC 值<**CMP_VAL** 值 **IF** CMP[1]置位并产生中断)

Bit [5] CMPIE：ADC 数据比较中断使能位 0：不使能 1：使能

Bit [4] CMPEN：ADC 数据比较使能位(引脚电压与 **CMP_VAL** 寄存器设置值进行比较) 1:使能

Bit [1] ADCIE：ADC 转换中断使能位 1：使能

Bit [0] DMAEN：ADC DMA 使能位 1：使能

#define CMP_VAL:ADC 比较阈值寄存器 (BIT[17:0]比较数值。此数值需要转换为 18bit 的有符号数据，最高位(Bit[18])为符号位)，转换格式参考 DR 寄存器。

#define IF:ADC 中断状态寄存器

Bit [1] CMP：比较中断标志位，满足比较条件置位，写 1 清零

Bit [0] ADC：ADC 数据转换完成中断标志位，转换完成置位，写 1 清零

a. **偏移量测量**：1.配置 ANA_CR 寄存器(LDOEN[0]=1、RST[1]=1、CHOPENR[6:5]=0x03)；

2.配置 PGA_CR 寄存器(CHOPEN[1]=1、 PGAEN[0]=1、CAIN[8:4]增益配置为 1)；

3.配置 ADC 时钟频率为 **1MHz**：RCC 模块 CLK_SEL 寄存器 ADC_DIV[15:8]；

4.等待 2ms 后读取 ADC 转换结果 DR，**每次间隔 2ms 重复读取 10 次，求其平均值**。

b. **差分输入模式**：1.按步骤 a 获取偏移量 Val_Offset；

2. 配置 ANA_CR 寄存器(LDOEN[0]=1、RST[1]=1、选择差分通道 CH[11:8])；

3. 配置 PGA_CR 寄存器(CHOPEN[1]=1、 PGAEN[0]=1、合理配置 CAIN[8:4]增益)；

4.配置 ADC 时钟频率为 **1MHz**：RCC 模块 CLK_SEL 寄存器 ADC_DIV[15:8]（已配可省）；

5. 等待 2ms 后读取 ADC 转换结果 DR，**每次间隔 2ms 重复读取 10 次，求其平均值；**

6.得出步骤 5 平均值 Val，差分输入信号 = (Val- Val_Offset)*68.5uV/总增益数

c. **单端输入模式**：1.按步骤 a 获取偏移量 Val_Offset；

2.配置 ANA_CR 寄存器(LDOEN[0]=1、RST[1]=1、选择正端输入通道 CH[11:8]，负端为内部 VCmin=1.311V)；

3. 配置 PGA_CR 寄存器(CHOPEN[1]=1、 PGAEN[0]=1、配置 CAIN[8:4]增益为 1)；

4. 配置 ADC 时钟频率为 **1MHz**：RCC 模块 CLK_SEL 寄存器 ADC_DIV[15:8]（已配可省）；

5. 等待 2ms 后读取 ADC 转换结果 DR，**每次间隔 2ms 重复读取 10 次，求其平均值**；

6.得出步骤 5 平均值 Val，单端输入信号值：

Vin_Single= (Val- Val_Offset)*68.5uV/总增益数+VCmin，VCmin 默认=1.311V。

d. **温度检测模式**：1.配置 ANA_CR 寄存器(LDOEN[0]=1、RST[1]=1、CH[11:8]=0x0C)；

2. 配置 PGA_CR 寄存器(CHOPEN[1]=1、 PGAEN[0]=1、配置 CAIN[8:4]增益为 4)

3. 配置 ADC 时钟频率 **1MHZ**：RCC 模块 CLK_SEL 寄存器 ADC_DIV[15:8]（已配可省）；

4. 配置 TEMP_CR 寄存器(EN[0]=1、 OFFSET[1]=0、增益 GAIN[5:4]为 2)；

5. 等待 2ms 后读取 ADC 转换结果 DR，进行数据处理后记为 Val_1；

6. 配置 TEMP_CR 寄存器(EN[0]=1、 OFFSET[1]=1、增益 GAIN[5:4]为 2)；

7. 等待 2ms 后读取 ADC 转换结果 DR，进行数据处理后记为 Val_2；

8.重复步骤 4-7，共计 10 次，分别计算出平均值 Val_11 和 Val_22；

9.芯片温度电压 Vtemp =(Val_11- Val_22)/16 = (15.548*芯片温度)+4444.1。

I2S 音频控制寄存器

1.I2S 是针对数字音频数据传输而制定的一种总线标准，它采用独立的导线传输时钟、数据信号的设计，通过将数据和时钟信号分离避免了因时差诱发的失真。

2.标准 I2S 总线由：**TDM(时分多路复用)数据线**、**字选择线**和**时钟线**三根线组成，最高采样频率 **192KHz**，支持单声道和立体声道模式。

3. I2S 模块提供 2 个 **8 个字**大小的 FIFO，即 TX FIFO 和 RX FIFO(**先入先出**的双口缓冲器)。

a.8 Bit 模式下一个字数据格式：

N+3		N+2		N+1		N	
7	0	7	0	7	0	7	0
数据存储方式							
左声道数据+1		右声道数据+1		左声道数据		右声道数据	

b.16Bit 模式下一个字数据格式：

N+1		N	
15	0	15	0
数据存储方式			
左声道数据		右声道数据	

c. 24Bit 模式下一个字数据格式

	N	
	23	0
N	左声道数据	
N+1	右声道数据	

d. 32Bit 模式下一个字数据格式

	N	
	31	0
N	左声道数据	
N+1	右声道数据	

4.支持**四种格式**：I2S 格式、MSB justified 格式、PCM A 格式和 PCM B 格式。

5.**零交叉检测**：为了避免数据损坏导致频率突然变化而产生噪声，I2S 模块对**每个声道**数据进行检测，当发送的相邻两个数据符号位出现变化时，模块会产生中断，提醒 MCU 进行检测和处理，同时将后一个数据强制静音(即数据清零)。

6.**静音功能**：当静音功能打开时，数据仍将发送，但输出的数据将被强制置为 0。

7.I2S 有三个时钟线，即：**I2S_MCLK、I2S_BCK 和 I2S_LRCK**，且三个时钟一条线。

a.**I2S_MCLK**：主时钟，是 I2S_BCK 和 I2S_LRCK 时钟的参考时钟；

b.**I2S_BCK**：串行时钟(位时钟)，对应音频数据的每一位，**频率=声道*采样频率*采样宽度**；

c.**I2S_LRCK**：帧时钟，用于切换左右声道数据，一个时钟周期代表一个音频采样点数据，为 1(或 0)表示正在传输左声道，为 0(或 1)表示正在传输右声道，其**频率=采样频率**。

8.I2S_D0、I2S_D1 数据传输线　　**D0：用于数据发送，**　　**D1：用于数据接收**

#define RXDR：数据接收寄存器(内置 8 个字长的 FIFO，每次向 Rx FIFO 写入一个字，只读)

Bit [31:0] DATA

#define TXDR：数据发送寄存器(内置 8 个字长的 FIFO，每次向 Tx FIFO 写入一个字)

Bit [31:0] DATA

#define CR：I2S 控制寄存器

Bit[28] MODE：主从模式选择　　0：主机模式　　1：从机模式

Bit[27] DUPLEX：双工模式使能位　　0：双工模式关闭　　1：双工模式使能

Bit[26] RXTIMEOUT：计数超时控制位　1：传输进程被主设备强制停止时，将不发生接收完成(RXDONE)中断。

Bit[25:24] FORMAT：数据格式选择(尾部附四种格式时序)

FORMAT[25:24]值	0x00	0x01	0x10	0x11
描述	I2S 数据格式	MSB justified 数据格式	PCM A 声音数据格式	PCM B 声音数据格式

Bit[23] RXLRCH：声道接收控制位(**单声道模式有效**)　0:接收右声道　1:接收左声道

Bit[22] MONO_STEREO：单声道立体声选择　0:立体声道格式传输　1:单声道格式传输

Bit[21] RXDMA_EN：接收 DMA 请求使能　0:不使能 DMA 请求　1:使能 DMA 请求

注：使能 DMA 请求后，RXFIFO 中字的个数**等于**或者**大于** RXFIFO_TH[14:12]时，I2S 控制器会向 DMA 发出传输请求，直至 RXFIFO 为空才停止 DMA 传输。

Bit[20] TXDMA_EN：发送 DMA 请求使能　0：不使能 DMA 请求　1：使能 DMA 请求

注：使能 DMA 请求后，TXFIFO 中字的个数**小于** TXFIFO_TH[11:9]时，I2S 控制器会向 DMA 发出传输请求，直至 TXFIFO 满才停止 DMA 传输。

Bit[19] RXFIFO_CLR：清空 RX FIFO 使能位(**读该位恒为 0**)　0：无效　1:清空 RX FIFO

Bit[18] TXFIFO_CLR 清空 TX FIFO 使能位(**读该位恒为 0**)　0：无效　1:清空 TX FIFO

Bit[17] LZCEN：左声道零交叉检测使能位　0：禁止　1：使能左声道零交叉检测

Bit[16] RZCEN：右声道零交叉检测使能位　0：禁止　1：使能右声道零交叉检测

Bit[15] RXCLK_INVERSE：接收时钟相位选择　0：默认模式　1:反转模式

Bit[14:12] RXFIFO_TH：RX FIFO 阈值

RXFIFO_TH[14:12]值	0x00	0x01	……	0x06	0x07
Rx FIFO 阈值	0 字	1 字	……	6 字	7 字

说明：当 RXFIFO 中现有字数量**等于**或**大于** RXFIFO_TH[14:12]设置的阈值时，RXTHIF 位会被置位，此时可以根据设置来选择触发 RXDMA 或者 I2S 中断。

Bit[11:9] TXFIFO_TH：TX FIFO 阈值

TXFIFO_TH[11:9]值	0x00	0x01	……	0x06	0x07
Tx FIFO 阈值	0 字	1 字	……	6 字	7 字

说明：TXFIFO 中现有字数量**等于**或**少于** TXFIFO_TH[11:9]设置的阈值时，TXTHIF 位会被置位。此时可以根据设置来选择触发 TXDMA 或者 I2S 中断。

Bit[8] TXCLK_INVERSE：发射时钟相位模式选择　0：默认模式　1：反转模式

Bit[5:4] DATALEN：传输字长配置为位

DATALEN [5:4]值	0x00	0x01	0x10	0x11
传输字长	8Bit	16Bit	24Bit	32Bit

Bit[3] MUTE：传输静音使能位　0：正常模式　1：传输数据置 0，**声音静**音

Bit[2] RXEN：I2S 数据接收使能位　1：使能接收

Bit[1] TXEN：I2S 数据传输使能位　1：使能传输

Bit[0] EN：I2S 使能位　1：使能

#define IM：I2S 中断配置寄存器

Bit[9] LZCI：左声道零交叉中断使能位　1:使能(检测到左声道上有零交叉时,产生中断)
Bit[8] RZCI：右声道零交叉中断使能位　1:使能(检测到右声道上有零交叉时,产生中断)
Bit[7] TXDONE：发送完成中断使能位　1：使能(TX FIFO 为空时产生中断)
Bit[6] TXTH：TXFIFO 阙值中断使能位　1：使能(TXFIFO 中数据数量<=阈值时产生中断)
Bit[5] TX_OVERFLOW：TX FIFO 溢出中断使能位　1：使能
Bit[4] TX_UNDERFLOW：TX FIFO 下溢中断使能位　1：使能
Bit[3] RXDONE：接收完成中断使能位　1：使能
Bit[2] RXTH：RXFIFO 阙值中断使能位　1：使能(RXFIFO 中数据数量>=阈值时产生中断)
Bit[1] RX_OVERFLOW: RX FIFO 溢出中断使能位　1：使能
Bit[0] RX_UNDERFLOW：RX FIFO 下溢中断使能位　1：使能

#define IF：I2S 中断状态寄存器

Bit[12] TX：I2S 发送中断标志　0：未发生中断　1：发送中断产生
Bit[11] RX：I2S 接收中断标志　0：未发生中断　1：接收中断产生
Bit[10] I2S：I2S 中断标志位 0:未发生中断　1:有中断产生(RX、TX 任一中断置位该位)
Bit[9] LZCI：左声道零交叉检测标志　1：检测到零交叉，写 1 清除该位
Bit[8] RZCI：右声道零交叉检测标志　1：检测到零交叉，写 1 清除该位
Bit[7] TXDONE：发送完成中断标志位　1：本次发送完成，写 1 清除该位
Bit[6] TXTH：TX FIFO 阈值中断标志位　1：Tx FIFO 字<=TXFIFO_TH[11:9]设置的阈值
Bit[5] TX_OVERFLOW：Tx FIFO 溢出中断标志位　1：溢出中断，写 1 清零
Bit[4] TX_UNDERFLOW：Tx FIFO 下溢中断标志位　1：向下溢出中断，写 1 清零
Bit[3] RXDONE：接收完成中断标志位　1：本次接收完成，写 1 清除该位
Bit[2] RXTH：RX FIFO 阈值中断标志位　1：Rx FIFO 字>=RXFIFO_TH[14:12]设置的阈值
Bit[1] RX_OVERFLOW：Rx FIFO 溢出中断标志位　1：溢出中断，写 1 清零
Bit[0] RX_UNDERFLOW：Rx FIFO 下溢中断标志位　1：向下溢出中断，写 1 清零

#define SR：I2S 状态寄存器，只读寄存器

Bit[9:8] VALIDBYTE：Rx FIFO 中最后一个字(4Byte)中可用字节数

Bit[9:8]值	0x00	0x01	0x02	0x03
可用字节数	所有字节可用	1 个字节可用	2 个字节可用	3 个字节可用

Bit[7:4] TXCNT：记录当前时刻 Tx FIFO 中字的个数

TXCNT[7:4]值	0x00	0x01	……	0x07	0x08
字个数	无数据	1 个字	……	7 个字	8 个字

Bit[3:0] RXCNT：记录当前时刻 Rx FIFO 中字的个数

RXCNT[3:0]值	0x00	0x01	……	0x07	0x08
字个数	无数据	1 个字	……	7 个字	8 个字

a. **F_mclk=F_i2sclk / MCLKDIV(分频系数)**

F_mclk	**主时钟 MCLK 频率**
F_i2sclk	选用内部时钟，F_i2sclk 为 **RCC 中 CLK_DIV 寄存器 PERIPHERAL[27:24]**对系统分频值（默认 160MHz）；选用外部时钟时 F_i2sclk=外部输入的时钟频率。
MCLKDIV	分频系数，**RCC 中 I2S_CLK 寄存器 MCLKDIV[7:2]值，**该值>=2

b.F_bclk=F_i2sclk / BCLKDIV(分频系数)

F_bclk	**串行时钟 BCLK 频率**
F_i2sclk	选用内部时钟，F_i2sclk 为 **RCC 中 CLK_DIV 寄存器 PERIPHERAL[27:24]**对系统分频值（默认 160MHz）；选用外部时钟时 F_i2sclk=外部输入的时钟频率。
BCLKDIV	分频系数，**RCC 中 I2S_CLK 寄存器 BCLKDIV[17:8]值配置**，根据以下配置：**BCLKDIV=F_ i2sclk /(Fs*W*F)，其中：Fs**:音频数据的采样频率(最高支持 192KHz),**W**:采样位宽(8/1624/32bit)，单声道 **F**=1，双声道时 **F**=2。

数据传输流程：

A.主端发送音频数据 1.配置使用的引脚(SDO，BCLK，LRCLK 复用引脚)；

2.设置时钟(RCC 模块 I2S_CLK 寄存器)、使能时钟(RCC 模块 I2S_CLK 寄存器 I2S[15])；

3.配置 CR 寄存器(主机模式、传输格式、声道选择、数据位宽、左右声道、过零检测、数据发送使能)；　　4.设置 IM 寄存器(使能使用的中断)；

*5.使用 DMA 传输数据(DMA 模块 REQCH 寄存器配置 I2S 通道并配置发射数据的地址和长度；I2S 模块 CR 寄存器中使能 DMA、设置 FIFO 阈值)；

6.向 I2S 模块 TXDR 寄存器写入发送数据，使能 CR 寄存器 EN[0]位(发送数据)；

7.当 FIFO 中数据小于设置的阈值时,模块将向 DMA 模块请求数据,或发送 TXTHIF 中断。

8.当 FIFO 中数据全部取出后 TXDONE 中断将置位，最后一帧发送完成，TXUDIF 中断将置位，通知 CPU 发送完成，模块停止发送。

B. 主端接收音频数据:1.配置使用的引脚(SD1，BCLK，LRCLK 复用引脚)；

2.设置时钟(RCC 模块 I2S_CLK 寄存器)、使能时钟(RCC 模块 I2S_CLK 寄存器 I2S[15])；

3.配置 CR 寄存器(主机模式、传输格式、声道选择、数据位宽、左右声道、过零检测、数据接收使能)；　　4.设置 IM 寄存器(使能使用的中断)；

*5.使用 DMA 传输数据(DMA 模块 REQCH 寄存器配置 I2S 通道并配置发射数据的地址和长度；I2S 模块 CR 寄存器中使能 DMA、设置 FIFO 阈值)；

6.使能 CR 寄存器 EN[0]位,模块即自动发送 BCLK 和 LRCLK，同时从 SDI 上采集数据存储在 FIFO 中。当 FIFO 中数据高于设置的阈值时，模块将向 DMA 请求搬运数据到内存中，或者发送 RXTHIF 中断。当 CPU 关闭 RXEN 时 RXDONE 中断产生。CPU 可通过 SR 寄存器查询接收 FIFO 中有多少个数据和最后一个 word 中有多少个 byte 的有效数据。

7.数据接收完成后关闭 i2s 使能(EN[0]=0)。

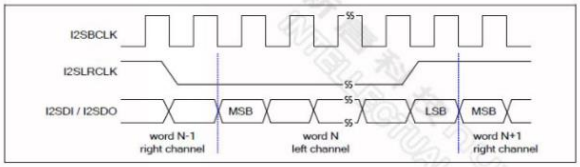


Fig1. I2S Bus Timing Diagram (PCM=0, Format=0)

I2S 数据格式

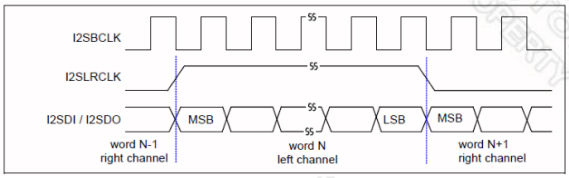


Fig2. MSB Justified Timing Diagram (PCM=0, Format=1)

MSB justified 数据格式

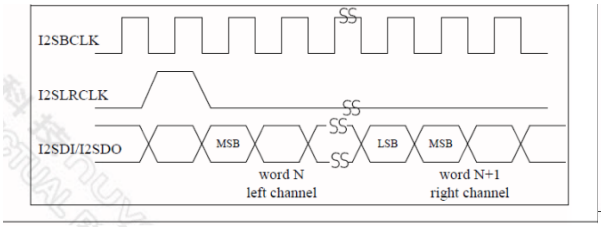


Fig3. PCM A Audio Diagram (PCM=1, Format=0)

PCM A 数据格式

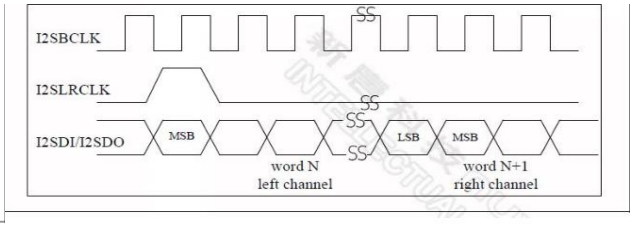


Fig4. PCM B Audio Diagram (PCM=1, Format=1)

PCM B 数据格式

DMA 寄存器

1.DMA 用于在外设与存储器之间以及存储器与存储器之间提供高速数据传输。可以在无需任何 CPU 操作的情况下通过 DMA 快速移动数据，支持 byte、half-word，word 单位传输操作。
2.DMA 挂载在 AHB 总线上，最大支持 8 通道,各通道互不干涉，可以同时运行。
3.每个 DMA 通道分配在不同的寄存器地址偏移段,可以直接选择相应通道的地址段进行配置使用，不同通道的寄存器配置方式完全一致。

DMA 基地址	0x40000800			
DMA_CH0	偏移量(0x10-0x38)		DMA_CH4	偏移量(0xD0-0xF8)
DMA_CH1	偏移量(0x40-0x68)		DMA_CH5	偏移量(0x100-0x128)
DMA_CH2	偏移量(0x70-0x98)		DMA_CH6	偏移量(0x130-0x158)
DMA_CH3	偏移量(0xA0-0xC8)		DMA_CH7	偏移量(0x160-0x188)

4.DMA 的**源地址**、**目的地址**可以设置为每次 DMA 操作完成之后**不变**、**递增**或**循环**三种模式；DMA 循环地址模式是指设置 DMA 的源地址和目的地址之后，数据搬运达到设定的循环边界之后，会跳转到循环起始地址，如此循环执行，直到到达设定的传输字节。
5.DMA 支持 **3 种传输模式**：内存到内存、内存到外设、外设到内存。

a.**内存到内存**：将源、目的地址配置为传输内存地址，MODE 寄存器 SHM[0]=0(软件方式)。

b.**内存到外设**:**源地址配置为内存地址，目的地址为外设地址**，MODE 寄存器 SHM[0]=1(硬件方式)，CH[5:2]选择所使用的外设。

c.**外设到内存**:**源地址配置为外设地址，目的地址为内存地址**，MODE 寄存器 SHM[0]=1(硬件方式)，CH[5:2]选择所使用的外设。

6.DMA 外设选择(内存到外设或外设到内存)除相应外设设置为 DMA Tx 或 Rx 外，还应 CH[5:2]选择所使用的外设；**注意：在 UART 使用 DMA 时，需通过 REQCH 寄存器 UART[2:0]选择 UART**。
7.**DMA 链表模式**：在 DMA 搬运当前链表内存数据时，可以提前向下一个链表中填充数据，DMA 搬完当前链表之后，判断到下一个链表有效（不为 0）后直接搬运下一个链表的数据。

操作流程：MODE 寄存器 LNM[1]=1，**DA 目的地址**寄存器设为链表起始地址，**CR1** 寄存器 START[0]=1 使能 DMA，当 DMA 处理完当前地址数据搬移后，通过软件设置有效标志，通知 DMA 链表中依然存在有效的数据，DMA 依据链表的有效标志位处理下一个待搬移的数据。

#define IM ：中断屏蔽寄存器，默认值：1(屏蔽状态)

Bit [15] CH7_TRANSFER_DONW：CH7 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [14] CH7_BURST_DONE：CH7 Burst 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [13] CH6_TRANSFER_DONW：CH6 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [12] CH6_BURST_DONE：CH6 Burst 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [11] CH5_TRANSFER_DONW：CH5 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [10] CH5_BURST_DONE：CH5 Burst 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [9] CH4_TRANSFER_DONW：CH4 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [8] CH4_BURST_DONE：CH4 Burst 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [7] CH3_TRANSFER_DONW：CH3 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [6] CH3_BURST_DONE：CH3 Burst 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [5] CH2_TRANSFER_DONW：CH2 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [4] CH2_BURST_DONE：CH2 Burst 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [3] CH1_TRANSFER_DONW：CH1 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [2] CH1_BURST_DONE：CH1 Burst 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [1] CH0_TRANSFER_DONW：CH0 传输完成中断屏蔽 0:使能中断 1:屏蔽中断
Bit [0] CH0_BURST_DONE：CH0 Burst 传输完成中断屏蔽 0:使能中断 1:屏蔽中断

#define IF：中断状态寄存器(查询中断状态)

Bit [15] CH7_TRANSFER_DONW：CH7 传输完成中断标志位， 写 1 清零

Bit [14] CH7_BURST_DONE：CH7 Burst 传输完成中断标志位， 写 1 清零

Bit [13] CH6_TRANSFER_DONW：CH6 传输完成中断标志位， 写 1 清零

Bit [12] CH6_BURST_DONE：CH6 Burst 传输完成中断标志位， 写 1 清零

Bit [11] CH5_TRANSFER_DONW：CH5 传输完成中断标志位， 写 1 清零

Bit [10] CH5_BURST_DONE：CH5 Burst 传输完成中断标志位， 写 1 清零

Bit [9] CH4_TRANSFER_DONW：CH4 传输完成中断标志位， 写 1 清零

Bit [8] CH4_BURST_DONE：CH4 Burst 传输完成中断标志位， 写 1 清零

Bit [7] CH3_TRANSFER_DONW：CH3 传输完成中断标志位， 写 1 清零

Bit [6] CH3_BURST_DONE：CH3 Burst 传输完成中断标志位， 写 1 清零

Bit [5] CH2_TRANSFER_DONW：CH2 传输完成中断标志位， 写 1 清零

Bit [4] CH2_BURST_DONE：CH2 Burst 传输完成中断标志位， 写 1 清零

Bit [3] CH1_TRANSFER_DONW：CH1 传输完成中断标志位， 写 1 清零

Bit [2] CH1_BURST_DONE：CH1 Burst 传输完成中断标志位， 写 1 清零

Bit [1] CH0_TRANSFER_DONW：CH0 传输完成中断标志位， 写 1 清零

Bit [0] CH0_BURST_DONE：CH0 Burst 传输完成中断标志位， 写 1 清零

#define REQCH：DMA 请求配置寄存器

Bit [20] SDIO: SDIO DMA 请求清除使能位， 写 1 清零

Bit [19] I2STX: I2STX DMA 请求清除使能位， 写 1 清零

Bit [18] I2SRX: I2SRX DMA 请求清除使能位， 写 1 清零

Bit [17] ADC3: ADC3 DMA 请求清除使能位， 写 1 清零

Bit [16] ADC2: ADC2 DMA 请求清除使能位， 写 1 清零

Bit [15] ADC1: ADC1 DMA 请求清除使能位， 写 1 清零

Bit [14] ADC0: ADC0 DMA 请求清除使能位， 写 1 清零

Bit [13] LSPI_TX: LSPI_TX DMA 请求清除使能位， 写 1 清零

Bit [12] LSPI_RX: LSPI_RX DMA 请求清除使能位， 写 1 清零

Bit [11] PWM_CAP1: PWM_CAP1 DMA 请求清除使能位， 写 1 清零

Bit [10] PWM_CAP0: PWM_CAP0 DMA 请求清除使能位， 写 1 清零

Bit [9] UART_TX: UART_TX DMA 请求清除使能位， 写 1 清零

Bit [8] UART_RX: UART_RX DMA 请求清除使能位， 写 1 清零

Bit [2:0] UART：通道 UART 选择

UART[2:0]	0x00	0x01	0x02	0x03	0x04	0x05
选择通道	UART0	UART1	UART2	UART3	UART4	UART5

#define SA：DMA 源地址寄存器(配置 DMA 传输的源地址)

Bit [31:0] ADDR

#define DA：DMA 目的地址寄存器(配置 DMA 传输的目的地址)

Bit [31:0] ADDR

#define SWA:循环模式 DMA 源地址寄存器(配置 DMA 传输的源地址)

Bit [31:0] ADDR

#define DWA:循环模式 DMA 目的地址寄存器(配置 DMA 传输的目的地址)

Bit [31:0] ADDR

#define WLEN:DMA 循环长度配置寄存器，配置目的地址、源地址的循环长度

Bit [31:16] WLEN_D：目的地址循环长度，**循环模式下有效**

Bit [15:0] WLEN_S：源地址循环长度，**循环模式下有效**

注：DMA 从目的(源)起始地址**递增搬运数据**，当搬运数据字节数达到此设定值之后，会跳转到循环起始地址，从起始地址开始继续搬运数据。

#define CR1:DMA 通道控制寄存器(转换 DMA 通道)

Bit [1] STOP：DMA 停止使能，该位置位后 DMA 在完成当前 Burst 操作后停止

Bit [0] START：启动 DMA 当前通道转换使能，完成通道转换或设置停止后硬件清零

#define MODE:DMA 通道选择寄存器

Bit [6] LINK：链表处理方式使能　　1:DMA 处理完第一个链表后继续读取下一个链表

Bit [5:2] CH：16 种 DMA 请求选择

CH[5:2]值	0x00	0x01	0x02	0x03	0x04	0x05	0x06
DMA 请求	UART_RX	UART_TX	PWM_CAP0	PWM_CAP1	LSPI_RX	LSPI_TX	ADC0
CH[5:2]值	0x07	0x08	0x09	0x0A	0x0B	0x0C	
DMA 请求	ADC1	ADC2	ADC3	I2SRX	I2STX	SDIO	

Bit [1] LNM：链表模式使能　　0：普通模式　　1：链表模式

Bit [0] SHM：DMA 模式配置　　0：软件方式　　1：硬件方式

#define CR2:UART 选择寄存器

Bit [23:8] LEN：每次启动 DMA 传输的字节(Byte)总数(0-65535Byte)，值为搬运单位**倍数**

Bit [7] BURST_SIZE:每次搬运数据单位　0:每次搬运 1 个单位　　1:每次搬运 4 个单位

Bit [6:5] TRANS_SIZE：每次搬运数据的单位，可选字节、半字(2Byte)、字(word=4Byte)

TRANS_SIZE[6:5]	0x00	0x01	0x02
描述	字节	半字=2Byte	字=4Byte

Bit [4:3] DINC：每次 DMA 操作后**目的地址**配置

DINC[4:3]	0x00	0x01	0x03
描述	操作后目的地址不变	目的地址自动累加	循环模式，目的地址自动循环

Bit [2:1] SINC：每次 DMA 操作后**源地址**配置

SINC[2:1]	0x00	0x01	0x03
描述	操作后源地址不变	源地址自动累加	循环模式，源地址自动循环

Bit [0] AUTORELOAD：自动加载使能　1:一次 DMA 传输后自动按当前配置再次启动 DMA 传输

#define SR：DMA 传输字节数寄存器(获取当前已传输的字节数)，只读寄存器

Bit [15:0] CNT：当前传输的字节个数　　(CR1 寄存器 STAR[0]置 1 或清零开始重新计数)

#define LA：DMA 链表入口地址寄存器

Bit [31:0] LA_ADDR

#define CDA：DMA 当前目的地址寄存器(当前 DMA 操作的地址)，只读寄存器

Bit [31:0] CDA_ADDR

TOUCH 触摸控制器

1.触摸原理：系统采用 40Mhz 时钟对计数窗口进行计数，当按键被触摸时，按键电路的电容值发生变化，从而影响模块输出时钟频率。通过检测模块的输出时钟的频率判断按键的状态,**W806 支持 15 路(TOUCH[0-14])**。

2.硬件实现：每隔一定时间逐次扫描每个触摸按键的状态，在设定的时间内计数并记录每个按键的状态;如果扫描值超过设定值，则判断该按键被触摸，**通过中断上报 MCU 系统**。

3.**扫描周期**：MCU 按照设定时间 T 轮询各个 TOUCHx 窗口。

#define CR 触摸控制寄存器

Bit [31:26] SCAN_PERIOD：扫描周期 N，单位：16ms，即隔 **N*16ms 扫描一次**

Bit [25:20] CAPDET_CNT：TOUCHx 窗口脉冲计数值 M，即每个触摸按键的检测脉冲个数

注：为避免切换通道导致的抖动，硬件在第三个脉冲才开始计数，因此如果**设置为 M，则实际计数窗口为 M-2**。例：设置为 20 ，则实际以 18 个 CAPDET 脉冲的周期为计数窗口。

Bit [19:4] CH_SEL：扫描对应触摸按键，每 Bit 位对应一位按键　　**0**：不扫描

BIT[x]位	BIT[19]	BIT[18]	……	BIT[5]	BIT[4]
对应通道	TOUCH15**(无效)**	TOUCH14	……	TOUCH1	TOUCH0

Bit [0] EN：触摸按键使能位　　0：禁止　　1：使能

#define CHxCR：x 通道单路控制寄存器，x = [0-14]

Bit [22:8] COUNT：TOUCHx 触摸计数值(用户配置)

Bit [6:0] THRESHOLD：TOUCHx 触摸阈值(用户配置)

注：**硬件扫描 TOUCHx 值> 计数值(COUNT[22:8])+阈值(THRESHOLD[6:0])表示按键被触摸**。

#define IE_IF：中断控制器

Bit [31:16] EN：触摸 x 触发中断使能　　1：使能

BIT[x]位	BIT[31]	BIT[30]	……	BIT[17]	BIT[16]
对应通道	TOUCH15**(无效)**	TOUCH14	……	TOUCH1	TOUCH0

Bit [15:0] FLAG：触摸 xPAD 标志位　　1：PAD 被触发

BIT[x]位	BIT[15]	BIT[14]	……	BIT[1]	BIT[0]
对应通道	TOUCH15**(无效)**	TOUCH14	……	TOUCH1	TOUCH0

注意：当使用 TOUCH 功能时，需使用 TOUCH0。**如果不使用 TOUCH0，对应 TOUCH0 IO（GPIO A7 也不能复用其他功能(必须置于高阻状态） ，否则可能烧毁 touch sensor 内部电路。**

a.配置流程：1.配置 CR 寄存器的**扫描周期、扫描窗口值、选择扫描 IO**。

2.配置 CHxCR 寄存器的计数值和触摸阈值。

3.使能 CR 寄存器 EN[0]=1;

b.PAD 触发流程：使能触摸按键模块后，硬件会将**计数值(COUNT[22:8])**存储下来，**作为基数**；间隔扫描周期后扫描配置的 IO，并将当前扫描值存储下来；完成所有 IO 扫描后硬件将每个 IO **当前扫描值**与基数作比较，如果**当前扫描值> 基数 + 阈值**，则判定按键被触摸，**IE_IF 寄存器**相应 FLAG[15:0]位被置位，并通过中断上报给 CPU。

**PMU 模块中 WLAN_STTS(偏移地址 0x1C)寄存器用于配置 TOUCH 模块充电电流，其定义如下：

#define WLAN_STTS：TOUCH 模块充电电流配置

Bit [6:4] DA_CAPDET_IBIAS:TOUCH 充电电流配置　0:无电流,不工作　推荐:8(5uA 电流)

DA_CAPDET_IBIAS[6:4]位置“1”对应充电电流	Bit[6]	Bit[5]	Bit[4]
	5uA	2.5uA	1.25uA

****由于此寄存器未配置在 PMU 模块中，使用时需在 PMU_TypeDef 结构体尾部添加：

__IOM uint32_t RSV;

__IOM uint32_t WLAN_STTS;

在 DA_CAPDET_IBIAS[6:4]=0x04(5uA)，扫描窗口值 CAPDET_CNT[25:20]=3 情况下各 TOUCH 计数值 COUNT[22:8]推荐：

TOUCH 通道	TOUCH0	TOUCH3	TOUCH4	TOUCH5	TOUCH6
计数值	254	247	239	212	203
TOUCH7	TOUCH8	TOUCH9	TOUCH10	TOUCH11	TOUCH123
199	186	181	172	161	160

片内 QFLASH

1.W806 内置 QFLASH 控制器，提供总线方式的 QFLASH 读、写、擦操作，FLASH 在系统中地址从 0x08000000 开始，可以直接对该地址空间进行读操作，支持 24 位或 32 位地址访问 FLASH。

2.**命令的配置和启动**：FLASH 控制命令写入 **CMD_INFO** 寄存器的 INFO_CMD[7:0]位，再根据命令格式置位其他相应位。

3.**通信模式配置**：先向 FLASH 发送命令将 FLASH 配置为 QIO 或 QPI 模式，再将 FLASH 控制器(FLASH_CR)的 CR_QPIM[1]或 CR_QIOM[0]位置位

4.**数据缓存**：通过寄存器对 FLASH 进行读写时，使用 0x40000000 ~ 0x400003ff 地址的存储空间作为数据缓存，该段地址空间同时也作为 RSA 加解密模块使用的数据缓存。即读数据时，控制器将从 FLASH 读出的数据顺序存储在从 0x40000000 开始的地址空间。向 FLASH 写数据时也是从 0x40000000 开始读取数据后发送给 flash。

5. 每次读操作最多可读取 256 字节数据，可读取 QFlash 任意位置的数据。

读流程：a.填写命令寄存器，使内容为读数据命令，b.设置命令启动寄存器，c.从 RSA_BUFF 中读取目标数据。

6.擦除命令不需要数据，直接将命令寄存器配置为相应命令，然后再启动

#define CMD_INFO：QFLASH 命令配置寄存器

Bit [31] INFO_A：该位置 1 表示 **CMD_START** 寄存器中的 START_ADDR [27:8]携带 20 位地址

Bit [30] CRM：该位置 1 表示 **CMD_START** 寄存器中的 START_CRM[7:0]携带有内容

Bit [29] DM：该位置 1 表示 **CMD_INFO** 寄存器中的 DM_LEN[28:26]携带 Dummy 周期

Bit [28:26] DM_LEN:Dummy 所携带的周期数

Bit [25:16] DATA_LEN：指示 **CMD_INFO** 携带数据字节数 n

Bit [15] DATA：该位置 1 表示 **CMD_INFO** 携带数据

Bit [14] INFO_RD：读操作命令使能位，命令包含 RDSR、FR、QIOFR、RDPDRID、RSR 等

Bit [13] INFO_WRSR：该位置 1 表示命令域(INFO_CMD [7:0])填写的是写状态寄存器命令

Bit [12] INFO_PP：该位置 1 表示命令域(INFO_CMD [7:0])填写的是页编程命令

Bit [11] INFO_SE：该位置 1 表示命令域(INFO_CMD [7:0])填写的是扇区擦除命令

Bit [10] INFO_BE：该位置 1 表示命令域(INFO_CMD [7:0])填写的是 32K block 擦除命令

Bit [9] INFO_HBE：该位置 1 表示命令域(INFO_CMD [7:0])填写的是 64K block 擦除命令

Bit [8] INFO_CE：该位置 1 表示命令域(INFO_CMD [7:0])填写的是全芯片擦除命令

Bit [7:0] INFO_CMD：命令域，表示 QFLASH 操作命令，如下：

命令名称	命令字	命令缩写	命令名称	命令字	命令缩写
Write Enable	0x06	WREN	Quad IO Fast Read	0xEB	QIOFR
Write Disable	0x04	WRDI	Page Program	0x02	PP
Read Status	0x05	RDSR	Sector Erase	0x20	SE
Write Status	0x01	WRSR	Fast Read	0x0B	FR

#define CMD_START:QFLASH 命令启动寄存器

Bit [28] START_CMD：启动命令使能位，操作完成后硬件清零

Bit [27:8] START_ADDR：低 20 位地址 (在 **CMD_INFO** 寄存器 INFO_A[31]置 1 时有效)

Bit [7:0] START_CRM:CRM 内容值 (在 **CMD_INFO** 寄存器 CRM [30]置 1 时有效)

#define FLASH_CR:FLASH 控制寄存器，配置 FLASH 操作模式

Bit [6:4] CR_ADMOD

Bit [4:2] CR_TSHSL

Bit [1] CR_QPIM

Bit [0] CR_QIOM

#define REMAP:重映射寄存器

Bit [0] REMAP

#define ADDR:地址寄存器，配置操作地址

Bit [31:0] ADDR

#define DECRYPT_CR：解密控制寄存器，用于配置固件加密

Bit [7] CR_START

Bit [6:4] CR_EXPTSEL

Bit [3] CR_ DATADECRYPT

Bit [2] CR_DBUSDECRYPT

Bit [1] CR_CODEDECRYPT

#define DECRYPT_STA:解密状态寄存器，用于配置固件解密，只读寄存器

Bit [2] STA_KEYERR

Bit [1] STA_KEYRDY

Bit [0] STA_RSA

注意：用于缓存 QFLASH 读、写的数据或者 RSA 的加解密数据内存为 QFLASH 控制器和 RSA 共用模块，因此 QFLASH 读写与 RSA 操作不能同时，需要在驱动实现时增加互斥操作保护。

RSA 加密模块

1. RSA 运算硬件协助处理器， 提供 Montgomery（FIOS 算法）模乘运算功能。 配合 RSA 软件库一同实现 RSA 算法。支持 128 位到 2048 位模乘(模乘长度是 32 位的整数倍)。
2. 支持：D*D、X*Y、D*Y、X*X 四个模乘模式。

#define XBUF[256]：数据 X 寄存器(2048Bit)，对应地址 0x0000-0x00FC

0x0000	0x0004	0x0008	……	0x00F8	0x00FC
X[31:0]	X[63:32]	X[95:64]	……	X[2015:1984]	X[2047:2016]

#define YBUF[256]：数据 Y 寄存器(2048Bit)，对应地址 0x0100-0x01FC

0x0100	0x0104	0x0108	……	0x01F8	0x01FC
Y[31:0]	Y[63:32]	Y[95:64]	……	Y[2015:1984]	Y[2047:2016]

#define MBUF[256]：数据 M 寄存器(2048Bit)，对应地址 0x0200-0x02FC

0x0200	0x0204	0x0208	……	0x02F8	0x02FC
M[31:0]	M[63:32]	M[95:64]	……	M[2015:1984]	M[2047:2016]

#define DBUF[256]：数据 D 寄存器(2048Bit)，对应地址 0x0300-0x03FC

0x0300	0x0304	0x0308	……	0x03F8	0x03FC
D[31:0]	D[63:32]	D[95:64]	……	D[2015:1984]	D[2047:2016]

#define RSACON:RSA 控制寄存器

Bit[5]：模乘启动控制使能位， 1：启动模乘运算，运算结束硬件清零

Bit[4]：软复位使能位 1：软件复位，复位后硬件清零

Bit[3:2]：模乘模式选择

Bit[3:2]值	00	01	10	11
模乘模式	X=D*D	X=D*Y	D=X*Y	D=X*X

Bit[0]：模乘运算完成标志位，运算完成硬件置 1

#define RSAMC：MC 参数寄存器

Bit[31:0]：RSAMC 对应参数，读该寄存器为 0

#define RSAN：参数 N 寄存器(读该寄存器高 25 位为 0)

Bit[6:0]：RSAN 对应参数 N，N 值为模乘长度(x bit)除以 32 的值(若为 1024bit,N=32)

PSRAM 接口控制器

1.**QSPI 接口包含**：时钟信号 SCK，片选信号 CS 及 SD[0] - SD[3]共 4 根双向数据 IO ,其中 SD[0]对应 MISO，SD[1]对应 MOSI。

2.W806 芯片**内置** SPI/QSPI 接口的 PSRAM，提供总线方式的 PSRAM 读、写和擦除操作，**最高速率 80MHz，最大容量支持 64Mb**。

3.**在 SPI 模式下**：完成一次写操作需要 64 个 SCLK 周期，读操作需要 72 个 SCLK 周期。

4.**在 QSPI 模式下**：完成一次写操作需要 16 个 SCLK 周期，读操作需要 22 个 SCLK 周期。

5.对 PSRAM 进行读写的方式与普通 SRAM 相同，即向相应的总线地址写入/读出数据

6. **PSRAM 控制器不支持 WRAP 形式的 BURST，若访问 PSRAM 用到 WRAP BURST，务必将 PSRAM_CTRL 寄存器 INC_EN BURST[2]设为 0**。

#define PSRAM_CTRL：控制寄存器

Bit [11] HSM：PSRAM 半休眠模式使能　　1：使能

Bit [10:8] TCPH:片选信号 CS 高电平持续时间 N，T=N/40MHz，N 必须大于 1(详见芯片手册)

Bit [7:4] SPI_DIV：分频系数 M(M 必须>2),对 AHB(**CPU 时钟**)时钟进行分频。　默认值：4

Bit [2] INC_EN BURST：BURST 功能使能　　0:不支持　1:支持 AHB 总线上的 BURST 功能

注：当 AHB 总线上 HBURST 为 1/3/5/7 时，表示 AHB 总线启动一次地址递增的连续读/写，为提高 PSRAM 的访问速度，控制器在完成一个字的读/写后并不会拉高 CS，而是直接读/写下一个字的数据。

Bit [1] QUAD：QSPI 模式使能　　0：SPI 模式　　1：QSPI 模式

Bit [0] PSRAM：PSRAM 复位使能　　1：PSRAM 复位，复位后硬件清零该位

注：设置 PSRAM 工作模式**必须在初始化操作完成后**，不能在初始化期间同时设置。

#define PSRAM_OVERTIMER：超时控制寄存器

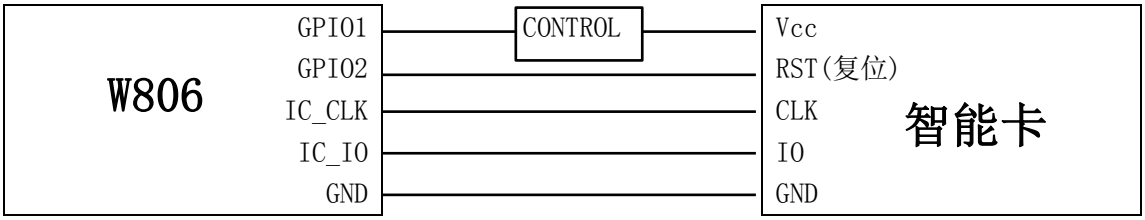
Bit [11:0] NUM：设置 CS 为低电平的最大时间，用于 BURST 模式

注：每开始一次 BURST 操作内部计数器从 0 开始计数，当计数器值> NUM[11:0]设定数值时，完成当前字的读/写后，控制器自动停止 BURST 操作(将 CS 变为高电平)，若此时 AHB 总线上还有数据需要读/写则会当成单独的字进行。

7816 模块(UART2 模块)

1. 7816 是国际标准的智能卡协议，协议规定了智能卡物理特性、尺寸和接口、电信号和传输协议、命令、安全等多方面信息。

2.W806 支持一组 UART(**UART2**)接口复用 7816，主要实现 ISO 7816 -3 电信号和传输协议，支持 T0 和 T1 卡，集成智能卡的**时钟**和**数据**两个接口，实现数据和命令的通信电信号逻辑；(实际智能卡应用中，还有 RST(复位引脚)、VCC、GND 这三个信号)，7816 模块实现异步半双工通信模式。



7816 连接示意图

2.********7816 模式由 UART2->寄存器进行配置**********

#define LC:数据流控制寄存器

Bit [24] SC_MODE：模式配置　　0：UART 模式　　1:7816 模式

Bit [23] T0_RxRe_EN：T0 模式自动重传配置 0：Rx 自动重传无效 1：Rx 自动重传使能

Bit [22:20] Rx_RETRANS_CNT：卡 T0 模式 Rx 自动重传次数

Bit [19] T0_TxRe_EN：T0 模式自动重传配置 0：Tx 自动重传无效 1：Tx 自动重传使能

Bit [18:16] Tx_RETRANS_CNT：卡 T0 模式 Tx 自动重传次数

Bit [15:11] MIN_BGT：最小块保护时间 N　　总时间 T = 10+Stop 位(默认 2 位)+N

卡 T0 模式：在发送和接收的两个相反方向连续字符的起始位下降沿之间的最小时间间隔不能小于 16 个 ETU(接收到的其起始位下降沿和最后发送的字节起始位下降沿间隔为 15 个 ETU 的字符。)。
卡 T1 模式：在发送和接收的两个相反方向连续字符的起始位下降沿之间的最小时间间隔（块保护时间，BGT）必须为 22 个 ETU(接收到的其起始位下降沿和最后发送的字节起始位下降沿间隔为 21 个 ETU 以内接收到的字符)。

Bit [10] Ti_CON：卡时钟控制配置　　0：配置为卡模式时产生卡时钟输出　1:时钟停止

Bit [9] PARIT：卡奇偶校验错误时是否接收数据　　0：不接受　　1：接收

Bit [8] T_CHO：卡 T0、T1 模式选择　　0：T0 模式　　1：T1 模式

Bit [7] rx_enable：uart/7816 模式下接收使能　　0：不使能　　1：使能

Bit [6] tx_enable：uart/7816 模式发送收使能　　0：不使能　　1：使能

Bit [5] SEBE:发送 Break 数据包使能　置位后 UART 发送一个 break 数据包，发完硬件清 0

Bit [4] PA：模式配置，**UART 模式**(0：偶校验　1：奇校验)　**7816 模式**(0：LSB　1：MSB)

Bit [3] PA_EN：UART 模式下奇偶校验使能　　1：奇偶校验使能

Bit [2] STOP：停止位配置，**UART 模式** 0:1 个　1:2 个　**7816 模式**：0:0.5 个　1:1.5 个

Bit [1:0] BIT_LEN：UART 模式下传输数据长度

BIT_LEN[1:0]值	0x00	0x01	0x02	0x03
数据长度	5bit	6bit	7bit	8bit

#define FC:硬件流控制寄存器(防止 FIFO 里的数据因软件处理不及时造成丢失)

Bit [4:2] RTS_TL：UART 模式下 RxFIFO 字节数>以下设置字节 N 时将 RTS 引脚置无效(即=1)

RTS[4:2]	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
字节	4 字节	8 字节	12 字节	16 字节	20 字节	24 字节	28 字节	31 字节

Bit [1] RTS set：UART 模式下 AFC_EN[0]=0 时，软件配置 RTS 引脚置位、清零

Bit [0] AFC_EN：UART 模式下接收条件 RTS 引脚由 RTS_TL[4:2]设定值控制 1：使能

#define DMAC:DMA 配置寄存器

Bit [7:3] RTO：RxFIFO 接收周期数 n(RxFIFO 中数据个数小于 RFL[5:4]设定个数，经过 n 个周期后仍未接收到新数据，则产生接收超时中断)

Bit [2] RTOE：接收数据超时使能位

Bit [1] RDE：接收 DMA 使能位 1：使能

Bit [0] TDE：发送 DMA 使能位 1：使能

注意：UART&7816 DMA 传输只能设置为 Byte 模式，不支持 half_word 和 word 传输模式。

#define BAUDR:波特率寄存器，(7816 模式波特率计算) BAUDR=((DIVF<<16)|DIV)

Bit [19:16] DIVF：波特率小数值 $DIVF = \frac{(40 * 10^6 + Fsc\_clk)}{2 * Fsc\_clk} - 1$ 取整

Bit [15:0] DIV：波特率整数值 $DIV = \frac{Fi}{Di}$ Fi、Di 为智能卡反馈参数

上述：Fsc_clk 为需要给智能卡提供的 CLK EDU 是智能卡的一个时间单位，智能卡按此时间单位来传输数据和命令，其中 EDU 频率 F_edu= Fsc_clk/(DIV+1)

Fi 取值参考

8-5bits	0000	0001	0010	0011	0100	0101	0110	0111
Fi	372	372	558	744	1116	1488	1860	RFU
F(max)MHz	4	5	6	8	12	16	20	-
8-5bits	1000	1001	1010	1011	1100	1101	1110	1111
Fi	RFU	512	768	1024	1536	2048	RFU	RFU
F(max)MHz	-	5	7.5	10	15	20	-	-

Di 取值参考

4-1bits	0000	0001	0010	0011	0100	0101	0110	0111
Di	RFU	1	2	4	8	16	32	64
4-1bits	1000	1001	1010	1011	1100	1101	1110	1111
Di	12	20	RFU	RFU	RFU	RFU	RFU	RFU

#define FIFOS:FIFO 状态寄存器

Bit [12] CTSS：当前 CTS 信号状态

Bit [11:6] RFC：RXFIFO 中数据个数

Bit [5:0] TFC：TXFIFO 中数据个数

#define FIFOC:FIFO 控制寄存器(设置 uart&7816 fifo 触发等级)

Bit [5:4] RFL：接收中断触发选择，当 RXFIFO 中数据字节大于、等于 RFL 设置数量触发中断或触发 RXDMA

RFL[5:4]值	0x00	0x01	0x02	0x03
描述	1 字节	4 字节	8 字节	16 字节

Bit [3:2] TFL：发送中断触发选择，当 TXFIFO 中数据字节小于、等于 TFL 设置数量触发中断或触发 TXDMA

TFL[3:2]值	0x00	0x01	0x02	0x03
描述	0	4	8	16

Bit [1] RFRST：清除接收 FIFO　1：清除

Bit [0] TFRST：清除发送 FIFO　1：清除

#define INTM:中断屏蔽寄存器

Bit [9] ER：7816 卡发送数据时收到错误信号　1：错误信号

Bit [8] OE：接收数据溢出中断屏蔽位　1：屏蔽　默认值：1

Bit [7] PE：奇偶校验中断屏蔽位　1：屏蔽　默认值：1

Bit [6] FE：数据帧错误中断屏蔽位　1：屏蔽　默认值：1

Bit [5] BD：Break 信号检测中断屏蔽位　1：屏蔽　默认值：1

Bit [4] CTS：CTS 信号变化中断屏蔽位　1：屏蔽　默认值：1

Bit [3] RTO：接收数据超时中断屏蔽位　1：屏蔽　默认值：1

Bit [2] RL：接收数据达到触发中断值屏蔽位　1：屏蔽　默认值：1

Bit [1] TL：发送数据达到触发中断值屏蔽位　1：屏蔽　默认值：1

Bit [0] TEMPT：发送数据空中断屏蔽位　1：屏蔽　默认值：1

#define INTS:中断状态寄存器

Bit [8] OE：接收数据溢出标志位

Bit [7] PE：校验错误标志位

Bit [6] FE：接收数据停止位错误标志位

Bit [5] BD：接收到 Break 数据包

Bit [4] CTS：CTS 信号变化标志位

Bit [3] RTO：接收数据超时中断标志位

Bit [2] RL：实际接收数据大于、等于 RFL 配置个数中断标志位

Bit [1] TL：发送数据小于、等于 TFL 配置个数中断标志位

Bit [0] TEMPT：当前数据发送完成且 TXFIFO 空中断标志位

#define TDW：UART 发送数据寄存器，只写

注:UART 发送、接收数据只支持字节(byte)操作,采用 Burst 传输时最多支持 4 个字(16byte)

#define RDW：UART 接收数据寄存器，只读

注:UART 发送、接收数据只支持字节(byte)操作,采用 Burst 传输时最多支持 4 个字(16byte)

#define GUARD_TIME：7816 保护时间寄存器 (7816 数据间保护时间)

Bit [7:0] EX_NUM：7816 模式下块保护时间计数 N，保护时间 T=10+stop 位+N

#define WAIT_TIME：7816 超时时间寄存器 （7816 接收数据超时时间）

Bit [23:0] WAIT_TIME：7816 模式下等待时间计数(以 ETU 为单位)

1.**智能卡上电复位步骤**：a.将 MCU 对应 IO、CLK、RST 配置为普通 GPIO 模式并保持低电平；

b. 设置 7816 为 T0 模式，即 UART2->LC(T_CHO[8]=0)；

c.MCU GPIO1 控制 Vcc 向 Smart Card 上电；

d. 配置 MCU 的 IO、CLK 为 7816 模式，由 7816 驱动时钟和数据；

e. 配置 7816 时钟频率并允许时钟输出；

f.置位 RST 引脚，等待接收 ATR 数据，若 40000 个时钟内没有收到 ATR 数据，则执行失活流程，卡片失活。

2.**7816 热复位流程**：a. 保持 VCC 上电状态；

b. 拉低 RST 引脚至少 400 个时钟周期；

c. 拉高 RST 引脚， 等待接收 ATR 数据， 若 40000 个时钟内没有收到 ATR 数据， 则执行失活流程，卡片失活。

3.**7816 失活流程**：a. 保持 VCC 上电状态；

b. 拉低 RST 引脚；

c. 配置 CLK 和 IO 为 GPIO 模式，并拉低；

d. 通过 GPIO 控制 VCC 掉电；

4.7816 接口配置流程：a.配置 7816 工作模式，即：UART2->LC(SC_MODE[24]=1)；

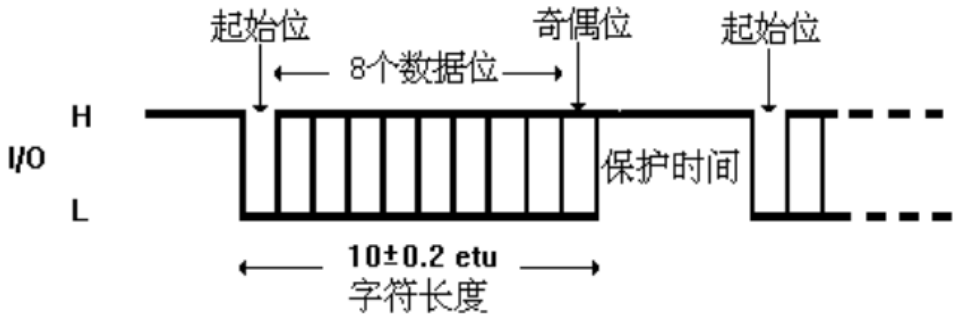
b.设置传输模式，即：UART2->LC(PA[4]=0 或 1，LSB 或 MSB)；

c.设置停止位，即：UART2->LC (STOP[2]=0 或 1,0.5 个或 1.5 个)；

d.选择卡类型，即：UART2->LC(T_CHO[8]=0 或 1，T0 卡或 T1 卡)；

e. 配置智能卡通信超时时间，WAIT_TIME 设置超时时间，接收数据超时未收到数据则产生超时中断。

5.**数据传输由硬件完成**：



使用 7816 功能在 USART_TypeDef 结构体后加入以下内容：

```
typedef struct
{        ……
__IOM uint32_t RSV4;//0x34
__IOM uint32_t RSV5;//0x38
__IOM uint32_t RSV6;//0x3C
__IOM uint32_t GUARD_TIME;//0x40
__IOM uint32_t WAIT_TIME;//0x44
} USART_TypeDef;
```

HSPI 高速 SPI

1.高速 SPI 共 5 根数据线： CS：从设备使能信号，由主设备控制　　SDI：串行数据输入
SCLK：时钟信号，由主设备产生　　SDO：串行数据输出　　INT：从设备有数据需要上传时，产生一个下降沿的中断，实现数据的主动上报

2. 芯片内部的HSPI和 wrapper控制器一起工作,wrapper控制器内部集成DMA,通过DMA 实现 HSPI 内部 FIFO 和芯片内部缓存之间的数据交换(数据交换由硬件控制实现)

3.数据格式分为**命令域**和**数据域**，命令域固定长度 8bit，数据域无固定

a.命令域最高位(Bit[7])为读写标志位，后 7 位 Bit[6:0]为地址，Bit[7]含义如下：

Bit[7]=0	向后 Bit[6:0]7bit 地址处读数据
Bit[7]=1	向后 Bit[6:0]7bit 地址处写数据

8bit		16 bit 或 32*n bit
Bit[7]R/W	Bit[6:0]address	DATA

4. 命令和数据之间可以没有等待时间，即传输命令字段后可以紧接着数据传输，不需要多余空闲时钟或者空闲时间。

5. 中断信号由**从设备**发送给主设备，通过 SPI_INT 管脚触发，低电平有效。每完成一帧数据上传对应一个中断，在每次中断中主机读取寄存器顺序为：**读 SPI_INT_HOST_MASK→读 SPI_INT_HOST_STTS→读数据。**

#defineHSPI_BASE_ADDR　　(DEVICE_BASE_ADDR + 0x2600)

#define CLEAR_FIFO :FIFO 清空寄存器

Bit [0] Clear FIFOs：清除 FIFO 使能位　　1：清除 Rx、Tx 中数据，完成后硬件清零

#define SPI_CFG ：HSPI 配置寄存器

Bit [3] Bigendian：大端模式使能　　0：小端模式　　1：大端模式
Bit [2] tx_drive：输出配置　0:片选有效时输出有效，其余时刻高阻　1:输出一直有效
Bit [1] CPHA：时钟相位　　0：前沿采样，后沿移出　　1：后沿采样，前沿移出
Bit [0] CPOL：时钟极性　　0：时钟在空闲时间低电平　　1：时钟在空闲时间高电平

#define MODE_CFG：HSPI 模式配置寄存器

Bit [0] Burst len:burst 数据长度　　0:1 个字　　1:4 个字(推荐此配置)

#define INT_MASK：HSPI 中断配置寄存器

Bit [1] Rx overflow:Rx FIFO 溢出中断使能　　**0：使能**　　1：不使能
Bit [0] Tx underflow：Tx FIFO 下溢中断使能　　**0：使能**　　1：不使能

#define INT_STTS：HSPI 中断状态寄存器

Bit [1] Rx overflow:Rx FIFO 溢出中断标志位，写 1 清零
Bit [0] Tx underflow：Tx FIFO 下溢中断标志位，写 1 清零

#define RXDAT_LEN：HSPI 数据上传长度寄存器 ，单位：字节(Byte)

Bit [15:0] Rx_dat_len：上传数据长度(上传长度都是**字的整数倍**)

3. 主机端通过固定的 SPI 命令格式访问 SPI 接口寄存器。**命令**长度固定为一个字节，**数据**长度固定为两个字节。

4.HSPI **主机端访问**的控制寄存器如下：

#define RX_DAT_LEN：数据长度寄存器，主机只读寄存器

Bit [15:0] DAT：数据长度 (上传数据时，主机用来获取从机端读取的数据长度),单位:字

#define TX_BUFF_AVAIL：下发数据标志寄存器，主机只读寄存器

Bit [1] TCA：发送命令允许位　　0：不允许　　1：允许(主机可以下发命令)

Bit [0] TBA：发送数据允许位　　0：不允许　　1：允许(主机可以下发数据)

#define SPI_INT_HOST_MASK：中断配置寄存器

Bit [0] IDC：中断屏蔽使能　　0：使能中断　　1：屏蔽中断

#define SPI_INT_HOST_STTS：中断状态寄存器，主机只读寄存器，读清零

Bit [0] UDCR：从机中断状态寄存器　　0：数据、命令没准备好　1：数据、命令已准备好

#define　DAT_PORT0：数据寄存器 0(主机通过本端口向从机下发数据，下发前 N-1 个数据)

Bit [8*n] DAT0：主机写该寄存器，即可下发数据，读该寄存器，即可上传数据。该寄存器用于传输非最后(N-1)个数据。

#define DAT_PORT1：数据寄存器 1(主机通过本端口向从机下发数据，下发最后第 N 个数据)

Bit [8*n] DAT1：主机写该寄存器，即可下发数据，读该寄存器，即可上传数据。该寄存器用于传输最后(N)个数据。

#define DN_CMD_PORT0：命令寄存器 0，只写寄存器

Bit [8*n] CMD0：主机写该寄存器，即可下发命令，下发前 N-1 个命令

#define DN_CMD_PORT1：命令寄存器 1，只写寄存器

Bit [8*n] CMD1：主机写该寄存器，即可下发命令，下发最后第 N 个命令

1.主机下发数据流程：a.主机准备好下发数据；

b.查询 **TX_BUFF_AVAIL** 寄存器 TBA[0]==1；

c.前部 SPI 命令=**0x80**(即 7bit 地址为 0x00),最后一次发送命令=**0x90**(7bit 地址=0x10)

注：下发的数据的长度必须是以字为单位，如果非整字，填 0 补齐

2.主机下发命令流程：a.主机准备好下发的命令数据；

b.查询 **TX_BUFF_AVAIL** 寄存器 TCA[1]==1；

c.前部 SPI 命令=**0x81**(即 7bit 地址为 0x01),最后一次发送命令=**0x91**(7bit 地址=0x11)

注：下发的命令长度必须是以字为单位，如果非整字，填 0 补齐

3.主机读取从机数据、命令流程：a.等待从机 SPI 设备产生中断；

b.中断中读取 **SPI_INT_HOST_STTS 寄存器，**获取 UDCR[0]值；

c. 读取 **RX_DAT_LEN** 寄存器，获得上传数据长度；

d.主机读取数据,前部读取采用地址 0x00(命令=0x00),最后一次读取采用地址 0x10(命令=0x10)；数据读取完成,根据描述符判断接收到的是数据还是命令。

SDIO_HOST 模块

1.SDIO 设备挂载于 **AHB1 总线，主机最高速度 50MHz，最大支持 1024 字节的 Block(块)**操作。
2. SDIO HOST 兼容 SDIO 2.0 协议的 SDIO 设备和 SD 卡设备，共 6 个接口:**CK、CMD 和 D0-D3**。

#define MMC_CTL：SDIO 控制寄存器//0x00

Bit [10] WA_EN：SDIO 读等待使能　　0：关闭　　1：使能
Bit [9] INT_EN：SDIO 中断使能　　0：关闭　　1：使能
Bit [8] MODE：SDIO 模式选择　　0：SD/MMC 模式　　1：SDIO 模式
Bit [7] DAT_LEN：SD/MMC/SDIO 接口数据宽度　0:1bit　　1:4bit
Bit [6] SPEED：SD/MMC/SDIO 传输模式　　0：低速模式　　1：高速模式

低速模式	Flow=**1MHz**
高速模式	合理选择 TI_RATE[5:3]分频系数 N，使 **Fcpu/N <=50MHz**

Bit [5:3] TI_RATE：SD/MMC/SDIO 时钟速率，对**基准频率 Base clk=Fcpu/N**

TI_RATE[5:3]	Fcpu 分频系数 N		TI_RATE[5:3]	Fcpu 分频系数 N
000	2		100	10
001	4		101	12
010	6		110	14
011	8		111	16

Bit [2] DRI_MODE:SD/MMC/SDIO 接口驱动模式选择　0：推挽模式　1：开漏模式
Bit [1] SI_MODE：信号模式选择　　0：由 **MMC_PORT 寄存器**决定　　1:自动选择传输模式
Bit [0] IO_MODE：端口模式选择　　0：SPI 模式　　1：MMC 模式

#define MMC_IO：SDIO IO 配置寄存器//0x04

Bit [9] CMD12C:CMD12(IO 停止)标志　　0：当前命令不是 CMD12　　1：前命令是 CMD12
Bit [8] CMDATT：SDIO 当前命令属性　0:命令后无数据块和命令响应　　1:命令后有数据块
Bit [7] NOPEN:命令(响应)和单个数据块后自动生成 8 个 NULL 时钟使能　0:关闭　1:使能
Bit [6] COMEN：发送命令后自动接收从机 response 使能　　0：关闭　　1：使能
Bit [5] SNOPEN:时钟线上生成 8 个 NULL 时钟使能　0:由 **NULLCH[3]**配置　1:使能 8 个 NULL
Bit [4] RCIS：读 CID、CSD 寄存器配置(发送 CID 或 CSD 命令时，SD 卡设备会在 CMD 线上回复 136bit CID 或 11CSD 数据)　置 1:CID/CSD 数据将存储在命令 buffef 区的[135:8]中。
Bit **[3] NULLCH**:SNOPEN[5]=0 时 8 个 NULL 时钟选择　　0:发送命令生成　　1:接收回应生成
Bit [2] NTEN:自动生成 8 个 NULL 时钟使能(传输完成自动清 0)　　0:关闭　　1:使能
Bit [1] DATADI：传输数据方向设置　　0:读数据　　1:写数据
Bit [0] AUTR：数据自动传输使能(传输完毕硬件清零　　0：关闭　　1：使能

#define MMC_BYTECNTL：传输数据计数寄存器,单位：字节//0x08

Bit [15:0] BYTE_CNT

#define MMC_TR_BLOCKCNT：多数据块传输时，已完成数据块计数器,只读寄存器//0x0C

Bit [15:0] BLOCK_CNT

#defefine MMC_CRCCTL：校验控制寄存器//0x10

Bit [7] CMREN：IO 端口 CMD 线上 CRC 校验使能　　0：关闭　　1：使能
Bit [6] DACRC：SDIO_D0-3 数据端口 CRC 校验使能　　0：关闭　　1：使能

Bit [5] AUTCRC:自动 CRC 校验使能 0：关闭 1：使能，当 crc_status !=3’b010 时，会产生 crc 状态中断，写数据传输将被 stop 命令中断，并且 mmc_io[0] 或 mmc_io_mbctl[2:0] 将被清除

Bit [4] REBRE：在 response 之前读多个数据块使能 0：关闭 1：使能

Bit [3:2] DACRS：数据校验选择，参见 table4

Bit [1] CMDFLAG：CMD 校验标志位(只读) 1：校验错误

Bit [0] DAFLAG：数据校验标志位(只读) 1：校验错误

#define CMD_CRC：校验寄存器，只读寄存器//0x14

Bit [6:0] CMD CRC：CMD 命令校验值

#define DAT_CRCL：数据校验低位寄存器，只读寄存器//0x18

Bit [7:0] DAT CRC：数据低位校验值

#define DAT_CRCH：数据校验高位寄存器，只读寄存器//0x1C

Bit [7:0] DAT CRC：数据高位校验值

#define MMC_PORT：端口配置寄存器//0x20

Bit [7] SCKS：SDIO 时钟线信号使能 ？？？ 1：使能 默认：0

Bit [6] CMDS：SDIO 命令线信号使能 1：使能 默认：1

Bit [5] DATS：SDIO 数据线信号使能 1：使能 默认：1

Bit [4] AUTNCR：NCR 超时自动检查使能 1：使能 默认

Bit [3:0] NCRNUM：NCR 超时计数值 N(表示 N 个 SDIO 时钟)

#define MMC_INT_MASK：中断状态寄存器，只读寄存器//0x24

Bit [8] DA1DI：SDIO 数据线 D1 中断屏蔽 0：屏蔽 1：中断使能 默认：0

Bit [7] TOCRC:CRC 校验 TOKEN 错误中断屏蔽 0：屏蔽 1：中断使能 默认：0

Bit [6] CRNCR:命令和响应 NCR 超时中断屏蔽 0：屏蔽 1：中断使能 默认：0

Bit [5] MDBLOC：多数据块超时中断屏蔽 0：屏蔽 1：中断使能 默认：0

Bit [4] MDBLTR：多数据块传输完成中断屏蔽 0：屏蔽 1：中断使能 默认：0

Bit [3] CMCRCE：命令校验错误中断屏蔽 0：屏蔽 1：中断使能 默认：0

Bit [2] DACRCE：数据校验错误中断屏蔽 0：屏蔽 1：中断使能 默认：0

Bit [1] DATRO：数据传输完成中断屏蔽 0：屏蔽 1：中断使能 默认：0

Bit [0] CMTRO：命令传输完成中断屏蔽 0：屏蔽 1：中断使能 默认：0

#define MMC_INT_SRC：中断标志清除寄存器//0x28

Bit [8] DA1DI：SDIO 数据线 D1 中断状态 **读：获取中断状态，写：清零相应位**

Bit [7] TOCRC:CRC 校验 TOKEN 错误中断状态

Bit [6] CRNCR:命令和响应 NCR 超时中断状态

Bit [5] MDBLOC：多数据块超时中断状态

Bit [4] MDBLTR：多数据块传输完成中断状态

Bit [3] CMCRCE：命令校验错误中断状态

Bit [2] DACRCE：数据校验错误中断状态

Bit [1] DATRO：数据传输完成中断状态

Bit [0] CMTRO：命令传输完成中断状态

#define MMC_CARDSEL：SD 卡配置寄存器//0x2C

Bit [7] SDCON:SDIO 控制器使能　　0：关闭　　1：使能

Bit [6] SDCLE:SDIO 卡设备时钟使能　　0：关闭　　1：使能

Bit [5:0] 1MDA:SDIO **1MHZ** 时钟基准配置位(1MHz=Fcpu/N/((1MDA[5:0]+1)*2)

#define MMC_SIG:SDIO 信号寄存器//0x30

Bit [7] CMDPOR:CMD 线信号　　读该位获取 CMD 信号线状态　　复位值：1

Bit [6:4] TOKCRC：数据 CRC TOKEN 校验值　　复位值:111

Bit [3] DAT3：数据 D3 信号　　复位值：1

Bit [2] DAT2：数据 D2 信号　　复位值：1

Bit [1] DAT1：数据 D1 信号　　复位值：1

Bit [0] DAT0：数据 D0 信号　　复位值：1

#define MMC_IO_MBCTL:数据块 IO 控制寄存器//0x34

Bit [7:6] TIOTS：NAC 超时配置，table5

Bit [5:4] TIOB：设备忙超配置，table6

Bit [3] POPH:端口时钟极性选择 0:上升沿移出，下降沿采样　1:下降沿移出，上升沿采集

Bit [2] CBATR:端口自动传输命令和数据块传输使能　　0：关闭　　1：使能

设置此 bit 为 1 （mmc_io[7:6]==11） 会触发一个 SD/MMC/SDIO 的命令，响应，8 null 时钟，多数据块传输。当数据传输完毕，此 bit 会自动清零

Bit [1] SMBT：多数据块传输配置　　0：写数据　　1：读数据

Bit [0] SMBTE：多数据块自动传输使能　　0：关闭　　1：使能

设置此 bit 为 1 （mmc_io[7:6]==11） 会触发一个 SD/MMC/SDIO 的多数据块传输。数据块个数在 mmc_blocknt 寄存器中设置。当数据传输完毕，此 bit 会自动清零。

#define MMC_BLOCKCNT:数据块数量配置寄存器//0x38

Bit [15:0] BCNT：配置传输 N 个数据块

#define MMC_TIMEOUTCNT:数据传输超时寄存器//0x3C

Bit [7:0] TIMNUN:超时值 N，T=Scale/N　Scale=**MMC_IO_MBCTL[7:6]/ MMC_IO_MBCTL[5:4]**

命令缓存	状态	偏移地址	Bit 位	命令线映射
CMD_Buff[0]	RW	0x40	Bit[7:0]	Bit[15:8]
CMD_Buff[1]	RW	0x44	Bit[7:0]	Bit[23:16]
CMD_Buff[2]	RW	0x48	Bit[7:0]	Bit[31:24]
CMD_Buff[3]	RW	0x4C	Bit[7:0]	Bit[39:32]
CMD_Buff[4]	RW	0x50	Bit[7:0]	Bit[47:40]
CMD_Buff[5]	RW	0x54	Bit[7:0]	Bit[55:48]
CMD_Buff[6]	RW	0x58	Bit[7:0]	Bit[63:56]
CMD_Buff[7]	RW	0x5C	Bit[7:0]	Bit[71:64]
CMD_Buff[8]	RW	0x60	Bit[7:0]	Bit[79:72]
CMD_Buff[9]	RW	0x64	Bit[7:0]	Bit[87:80]

CMD_Buff[10]	RW	0x68	Bit[7:0]	Bit[95:88]
CMD_Buff[11]	RW	0x6C	Bit[7:0]	Bit[103:96]
CMD_Buff[12]	RW	0x70	Bit[7:0]	Bit[111:104]
CMD_Buff[13]	RW	0x74	Bit[7:0]	Bit[119:112]
CMD_Buff[14]	RW	0x78	Bit[7:0]	Bit[127:120]
CMD_Buff[15]	RW	0x7C	Bit[7:0]	Bit[135:128]

#define BUF_CTL：数据缓存寄存器，用于清 0 相应配置//0x80

Bit [15] CLEN:数据缓存清除使能　　0：保持　　1:使能清除(一个时钟后清零该寄存器)

Bit [14] DMADE：DMA 屏蔽(**必须在使能 DMA 之前配置该位**) 0:不屏蔽　1:屏蔽　默认:0

Bit [12] DFIFO：数据 FIFO 信号屏蔽　　1：屏蔽　　默认:0

读卡时屏蔽 FIFO FULL(满)信号，写卡时屏蔽 FIFO 空信号

Bit [11] BUFL：缓冲访问方向设置　　0：读　　1：写

Bit [10] DMAE：DMA 硬件接口使能　　0：AHB 访问数据缓存　　1：DMA 接口

使用 DMA 接口时，在单数据块或者多数据块传输时当数据传输完成后将自动复位此位。

Bit [9:2] BUCNT:缓存数据量 N，仅在 DMAE[10]=1 时有效，**N 必须<=127(缓存深度 128 字)**

Bit [1] FIFOEP:数据缓存空信号(**只读**)　　默认：1

Bit [0] FIFOFU：数据缓存满信号(**只读**)　　默认：0

#define DATA_BUFF[128]：数据缓存空间　　地址 0x100-0x2FF(512Byte = 128 字)

Bit [31:0] DATA