8-bit MIC8S

Microcontroller with
4Kx14bit FLASH
Program Memory

LGT8F690A

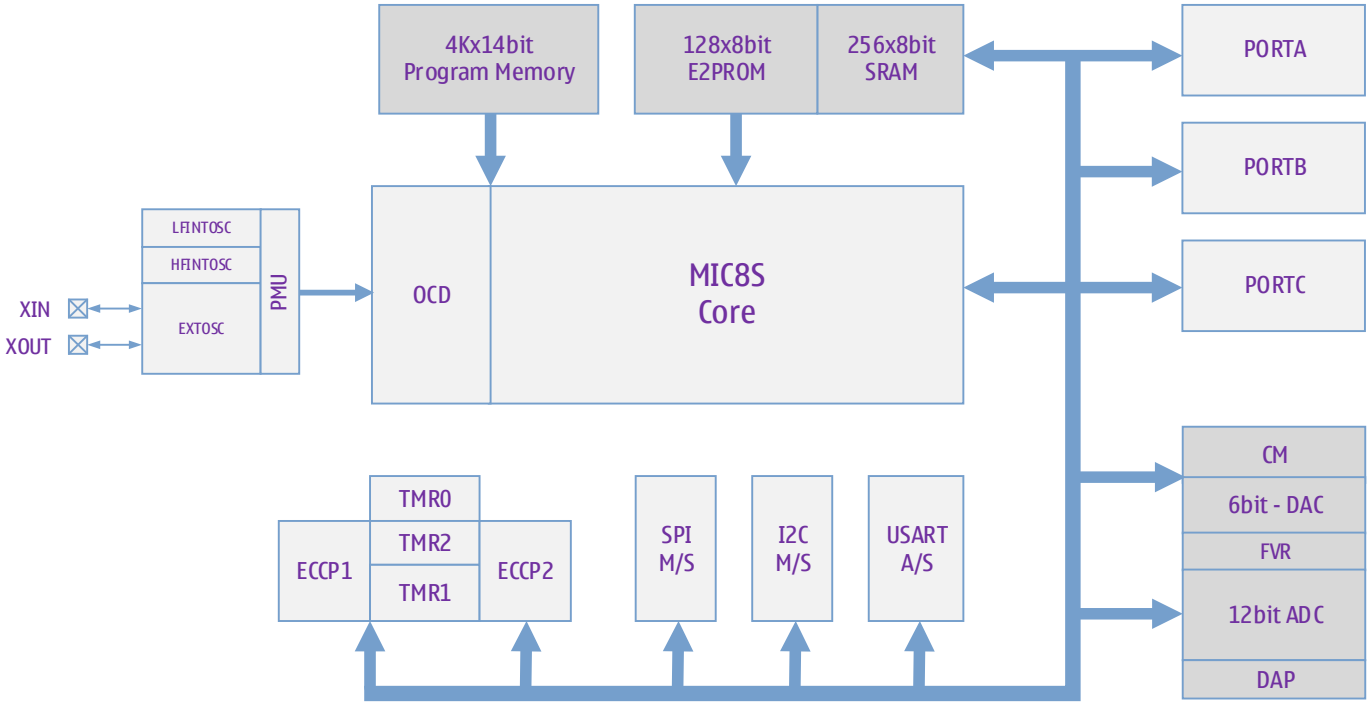
Overview
Version 1.0.9

应用领域
智能家电
手持仪器
自动控制

功能概述

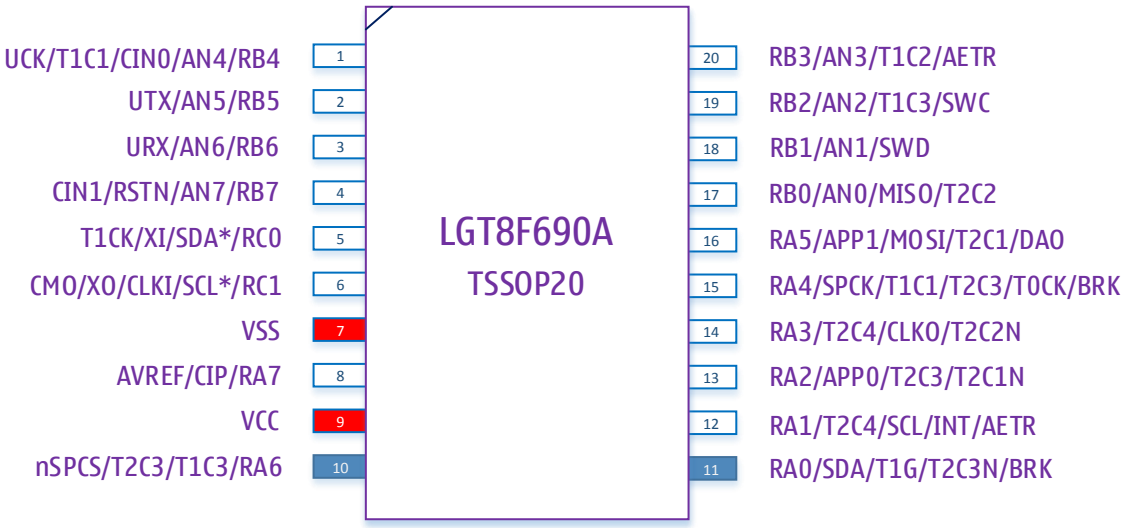
- 高性能低功耗 8 位 MIC8S 内核
  - 35 条指令，可配置 1T/2T/4T 指令周期
  - 8 级深度堆栈寄存器
  - 硬件 16x8, 16/8 乘除法器
- 4Kx14bit 1K-Cycle FLASH 程序存储器
- 128x8bit E2PROM 数据存储器
- 256x8bit SRAM 数据存储器
- 外设控制器
  - 一个 8 位定时器 Timer0
  - 两个 16 位定时器 Timer1/2，独立预分频器
    - 支持外部时钟门控输入
    - 支持同步/异步时钟输入模式
    - 支持同时 3 组互补+3 路独立 PWM 输出
  - 2x 增强俘获/比较/PWM 控制器(ECCP)
    - 上升/下降沿的死区控制
    - 相位控制，消隐控制以及硬件自动关闭
  - I2C 控制器，支持主机/从机模式
  - 增强型 USART 收发器
  - 同步串行接口(SPI)， 支持主机/从机模式
  - 12 位 8 通道模数转换器(ADC)
    - 内置 VCC/5 电源监控通道
    - 集成多路可编程增益(x1/8/16/32)差分放大器
    - 内部可校准 1.5V/2.56V±1%参考电压源
    - 支持自动通道溢出监控模式
  - 多路输入模拟比较器(CM)
  - 6bit 数模转换器(DAC)
  - 可编程看门狗定时器（WDT），独立 32K 低功耗 IRC
- 特殊处理器功能
  - 外部中断源与 I/O 电平变化中断
  - 2x 80mA 大电流推挽驱动，用于驱动高速 PWM
  - 全部 IO 支持 25mA 推挽驱动
  - 内置上电复位电路（POR）与可编程低电压检测电路（LVR）
  - 内置±1%可校准 16MHz RC 振荡器(粗调/细调)
  - 集成在线调试功能(OCD)
  - 支持外部低速/高速晶振(最高到 20MHz)，晶振失效保护
  - 定时器支持时钟倍频模式，产生更快的 PWM 输出
- 封装: TSSOP20/QFNWB20
- 工作环境
  - 工作电压：2.0V ~ 5.5V
  - 工作温度：-40C~+85C
  - 待机功耗：1uA@3.3V

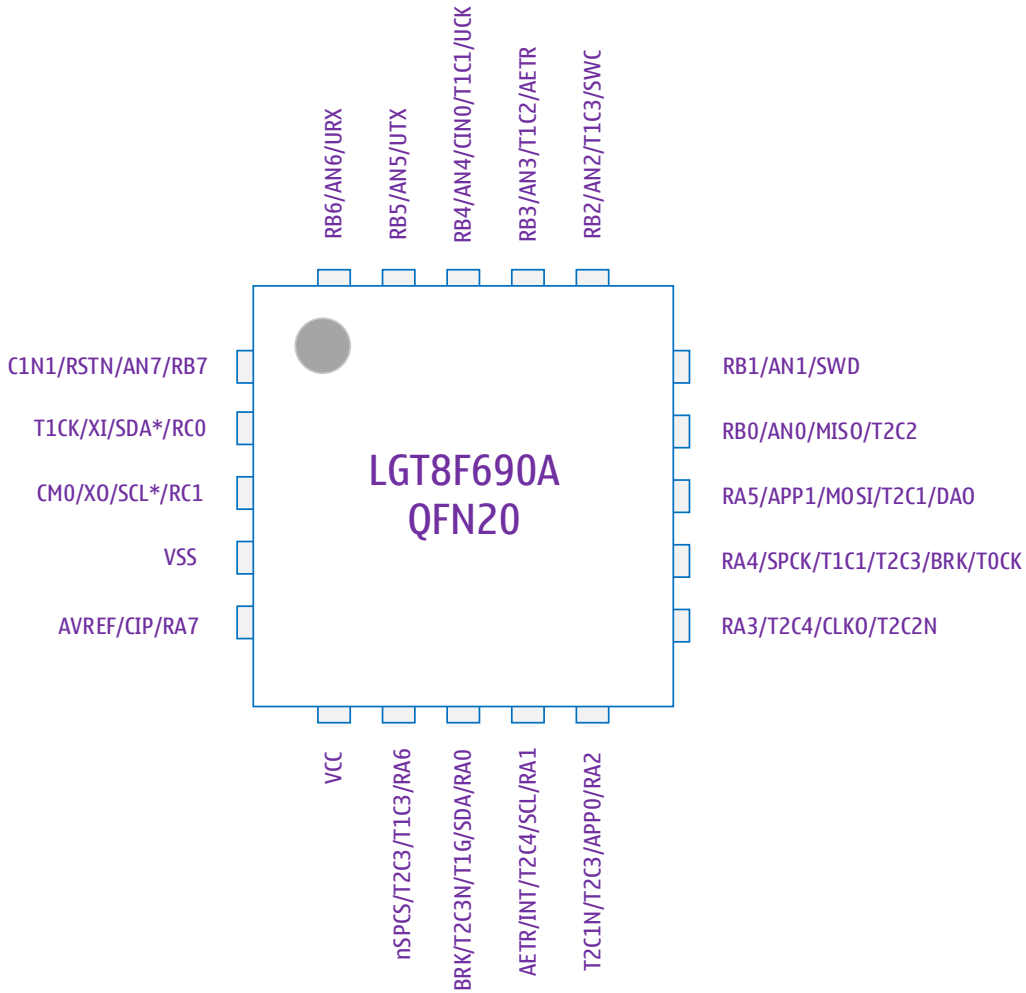
系统框架



模块名称	模块功能
MIC8S	MIC8S 8 位微处理器内核
OCD	量产/调试接口
PMU	功耗管理单元
H/LFINTOSC	内部 RC 振荡器
ECCP1/2	增强俘获/比较/PWM 控制器
I2C M/S	主/从模式 I2C 控制器
USART	增强同步/异步收发器(EUSART)
SPI	同步串行接口
ADC	12 位多通道模数转换器(ADC)
DAP	增益可编程差分放大器
FVR	内部 1.5V/2.56V 内部参考电压
DAC	6bit 数模转换器
CM	多路输入模拟比较器
TMR0/1/2	8/16 位定时器/计数器 0/1/2
COG	互补输出发生器
ECCP	增强比较/俘获/PWM 控制器
PORTA/B/C	通用可编程端口 A/B/C/D

管脚定义


UCK/T1C1/CIN0/AN4/RB4
UTX/AN5/RB5
URX/AN6/RB6
CIN1/RSTN/AN7/RB7
T1CK/XI/SDA*/RC0
CM0/XO/CLKI/SCL*/RC1
VSS
AVREF/CIP/RA7
VCC
nSPCS/T2C3/T1C3/RA6
1
2
3
4
5
6
7
8
9
10
LGT8F690A
TSSOP20
20
19
18
17
16
15
14
13
12
11
RB3/AN3/T1C2/AETR
RB2/AN2/T1C3/SWC
RB1/AN1/SWD
RB0/AN0/MISO/T2C2
RA5/APP1/MOSI/T2C1/DAO
RA4/SPCK/T1C1/T2C3/T0CK/BRK
RA3/T2C4/CLKO/T2C2N
RA2/APP0/T2C3/T2C1N
RA1/T2C4/SCL/INT/AETR
RA0/SDA/T1G/T2C3N/BRK


RB6/AN6/URX
RB5/AN5/UTX
RB4/AN4/CIN0/T1C1/UCK
RB3/AN3/T1C2/AETR
RB2/AN2/T1C3/SWC
C1N1/RSTN/AN7/RB7
T1CK/XI/SDA*/RC0
CM0/XO/SCL*/RC1
VSS
AVREF/CIP/RA7
LGT8F690A
QFN20
RB1/AN1/SWD
RB0/AN0/MISO/T2C2
RA5/APP1/MOSI/T2C1/DAO
RA4/SPCK/T1C1/T2C3/BRK/T0CK
RA3/T2C4/CLKO/T2C2N
VCC
nSPCS/T2C3/T1C3/RA6
BRK/T2C3N/T1G/SDA/RA0
AETR/INT/T2C4/SCL/RA1
T2C1N/T2C3/APP0/RA2

TSSOP20	QFNWB20	引脚名称	功能描述	其他说明
1	18	UCK	同步 USART 时钟	
		T1C1	TMR1 俘获输入/比较输出	
		CIN0	比较器负端输入 0	
		AN4	ADC 模拟输入通道 4	
		RB4	通用可编程端口 RB4	
2	19	UTX	USART 数据发送	
		AN5	ADC 模拟输入通道 5	
		RB5	通用可编程端口 RB5	
3	20	URX	USART 数据接收	
		AN6	ADC 模拟输入通道 6	
		RB6	通用可编程端口 RB6	
4	1	CIN1	比较器负端输入 1	
		RSTN	外部复位输入	
		RB7	通用可编程端口 RB7	
5	2	XI	晶振输入端口	
		SDA*	I2C 数据线(备用)	
		RC0	通用可编程端口 RC0	
6	3	XO	晶振输出端口	
		CMO	比较器输出	
		SCL	I2C 时钟线(备用)	
		CLKI	外部时钟输入	
		RC1	通用可编程端口 RC1	
7	4	VSS	系统地	
8	5	AVREF	ADC 外部参考输入	
		CIP	比较器正端输入	
		AN7	ADC 输入通道 7	
		RA7	通用可编程端口 RA7	
9	6	VDD	系统电源	
10	7	nSPCS	SPI 片选信号	
		T2C3	TMR2 俘获输入/比较输出	HD
		T1C3	TMR1 俘获输入/比较输出	
		RA6	通用可编程端口 RA6	
11	8	T2C3N	T2C3 互补输出	HD
		T1G	TMR1 门控输入	
		SDA	I2C 数据线	
		BRK	TMR2 比较输出刹车控制	
		RA0	通用可编程端口 RA0	

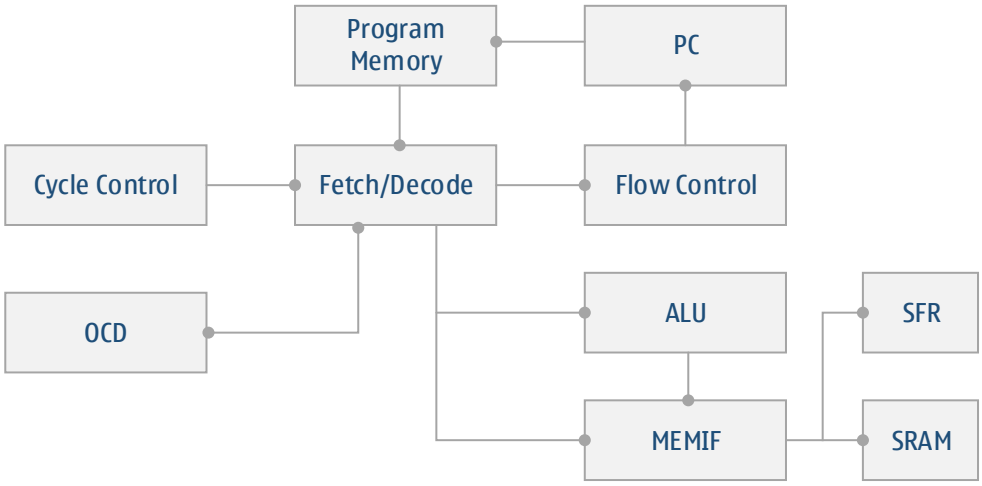
12	9	ATER	ADC 外部触发事件	
		SCL	I2C 时钟线	
		T2C4	TMR2 俘获输入/比较输出	
		RA1	通用可编程端口 RA1	
		INT	外部中断输入	
13	10	T2C1N	T2C1 互补输出	
		T2C3	TMR2 俘获输入/比较输出	
		APP0	差分放大器同向输入 0	
		RA2	通用可编程端口 RA2	
14	11	T2C2N	T2C2 互补输出	
		CLKO	可配置时钟输出	
		T2C4	TMR2 比较输出	
		RA3	通用可编程端口 RA3	
15	12	T2C3	TMR2 俘获输入/比较输出	
		T1C1	TMR1 比较输出	
		SPCK	SPI 接口时钟	
		BRK	TMR2 比较输出刹车控制	
		RA4	通用可编程端口 RA4	
16	13	T2C1	TMR2 俘获输入/比较输出	
		MOSI	SPI 接口主机输出从机输入	
		APP1	差分放大器同向输入 1	
		DAO	DAC 模拟输出	
		RA5	通用可编程端口 RA5	
17	14	T2C2	TMR2 俘获输入/比较输出	
		MISO	SPI 接口主机输入从机输出	
		APN0	差分放大器反向输入 0	
		AN0	ADC 模拟输入通道 0	
		RB0	通用可编程端口 RB0	
18	15	SWD	SWD 调试接口数据线	
		APN1	差分放大器反向输入 1	
		AN1	ADC 输入通道 1	
		RB1	通用可编程端口 RB1	
19	16	SWC	SWD 调试接口时钟线	
		T1C3	TMR2 俘获输入/比较输出	
		APN2	差分放大器反向输入 2	
		AN2	ADC 输入通道 2	
		RB2	通用可编程端口 RB2	
20	17	AETR	ADC 外部触发事件输入	
		T1C2	TMR1 俘获输入/比较输出	
		APN3	差分放大器反向输入 3	
		AN4	ADC 输入通道 3	
		RB3	通用可编程端口 RB3	

注释： HD = 80mA 推挽驱动

MIC8S 内核

- 1T/2T/4T 可配置指令周期
- 共计 35 条指令
- 可配置分页/连续地址映射模式
- 支持程序空间的查表指令
- 直接以及间接寻址模式
- 支持软件/硬件中断源
- 支持片上调试功能(OCD)

综述



MIC8S 指令控制/执行流程图

MIC8S 基于哈佛总线构架，采用分离的指令和数据访问总线控制。与 RISC 构架具有丰富的通用工作寄存器不同，MIC8S 构架仅有一个工作寄存器(W)；但 MIC8S 的大部分指令可以直接访问低端 64 字节的空间，配合分页的寻址模式，指令可以直接访问到全部特殊功能寄存器以及部分 RAM 空间，因此 MIC8S 仍然可以高效的完成所有的运算以及控制指令。

除工作寄存器(W)外，MIC8S 还有几个与指令相关的特殊功能寄存器，这些寄存器协助指令系统完成灵活的数据以及程序寻址，这些寄存器包括：

INDF：间接寻址数据寄存器。对 INDF 读写访问的目标地址由 FSR 寄存器决定；

FSR：间接寻址目标地址或基地址，可以用于访问数据或程序空间；

STATUS： 系统状态寄存器，控制页寻址的地址切换，系统运行状态以及 ALU 的执行结果状态位。

MIC8S 内核支持 1T/2T/4T 三种不同的指令周期模式。指令周期模式控制指令周期与系统时钟周期之间的比例关系。 1T 模式下，指令周期与系统时钟周期相同，指令执行单元以系统时钟相同的频率执行。在这种模式下， 指令最快可以在 1 个系统时钟周期内完成。需要注意的是，读 RAM 空间的指令，仍然需要两个系统时钟周期。2T/4T 模式下， 指令执行单元分别以系统时钟周期的 2/4 分频周期运行，在这两种模式下，所有指令的执行周期都是相等的。

内核的指令周期模式可以通过配置位 TCYC[1:0]设置。LGT8F690A 实现了 MCUCR 控制寄存器，可以通过指令随时更改指令周期模式， 可以灵活的实现内核的功耗控制。

系统上电后，MCUCR 寄存器从配置位中加载指令周期模式 TCYC 配置。此后，软件可以随时通过 MCUCR 寄存器的 TCYC 位更新。

MCUCR 寄存器 TCYC 位定义如下：

MCUCR[2:1] = TCYC 地址：0x19F @ Bank3	功能描述
00	1T 指令周期
01	2T 指令周期
1X	4T 指令周期（默认）

由于 MCUCR 寄存器控制了关键的内核运行模式，因此 MCUCR 寄存器有保护设计，更新 MCUCR 寄存器需要严格按照一个预定的时序：首先向 MCUCR 地址写数据 0x55，然后紧接着写入要更新的数据。

MCUCR 寄存器更新实例：

```
// C 语言代码实例
#include “lgt8f690a.h”

MCUCR = 0x55;              // 解锁寄存器保护
MCUCR = MCUCR & 0xF9;      // 更新 TCYC 为 1T 指令周期
```

MCUCR 寄存器定义细节，请参考本节最后的寄存器定义部分。

内核工作寄存器 W 对于指令而言为一个隐含的寄存器，大部分运算以及存取访问的指令，都隐含的将 W 寄存器作为目的或者源操作数。MIC8S 也将内核寄存器 W 映射到用户寄存器空间，可以通过映射到用户寄存器空间的 WREG 寄存器显式的操作 W 工作寄存器。

W 映射到用户寄存器空间方便实现调试模块对内核信息的监控以及更高效的实现运算加速。MIC8S 运算加速相关信息，请参考本手册运算加速章节。

[说明：对于 C 语言编程，请谨慎使用 WREG 寄存器，以免破坏 C 的运行环境。]

存储系统

程序存储空间

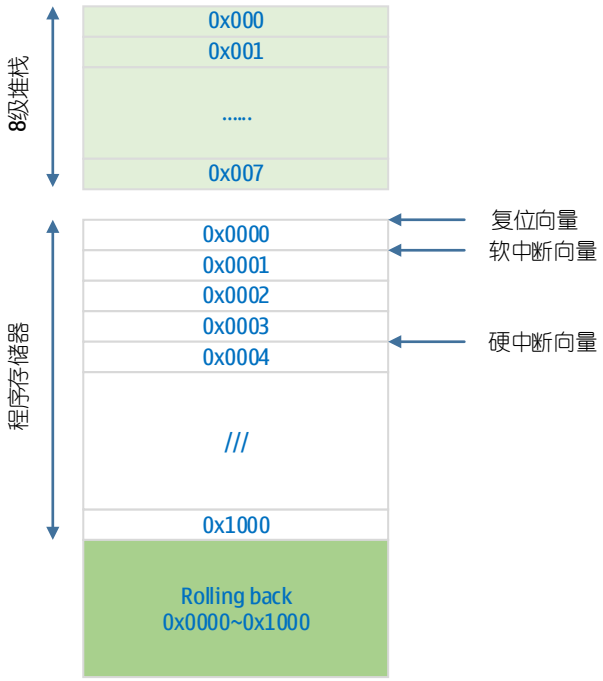
MIC8S 指令可直接寻址最大 8K 指令字空间，GOTO/CALL 指令宽度为一个指令字，可直寻址 2K 指令字空间。超出 GOTO/CALL 空间的跳转，需要配合 PCLATH 寄存器实现。LGT8F690A 实现了 4Kx14 (0x0000 – 0x1000)的程序空间。超出此空间的程序地址访问，将会回滚到 4K 范围内。系统上电/复位后，内核从 0x0000 地址开始执行。LGT8F690A 实现了硬件/软件两种中断请求，硬件中断的向量地址为 0x0004，软件中断(INT 指令)的向量地址为 0x0001。

8 级深度堆栈寄存器

LGT8F690A 实现了一个 8 级深度的硬件堆栈单元。堆栈独立于程序以及数据空间。堆栈只能通过堆栈指针访问；堆栈指针本身并不能通过指令直接访问。当系统执行 CALL 指令或者发生了有效的中断，当前 PC 被压入堆栈。当系统执行了 RETURN/RETLW/RETFIE 指令后，堆栈弹出最后压入的值(到 PC 指针)。在压栈和出栈操作过程中，PCLATH 并不会受到影响。

堆栈的深度为 8 级，当连续执行 8 次压栈操作后，后续的压栈操作将会破坏堆栈中的数据，最早压栈的数据将会从堆栈中弹出；同样如果多次执行出栈的操作，堆栈弹出最早的数据后，将会循序返回之前的数据。这两种情况均会导致系统执行的不确定，因此用户需要在使用时特别注意。

LGT8F690A 的程序/堆栈空间分布如下：



数据存储空间

LGT8F690A 使用分页模式管理内部数据空间。在分页地址映射模式下，LGT8F690A 的数据空间被分 4 个页面空间。4 个页面包括了 LGT8F690A 所有的内核控制寄存器，外设控制寄存器以及 RAM 空间。其中每个页面的大小为 128 字节。每个页面的最低 32 个字节为内核以及外设相关的寄存器；剩下的 96 字节为 RAM 空间以及扩展外设寄存器空间。每个页面的最高 16 个字节，统一映射到第 0 页的 70~7F 地址空间。这 16 个字节的共享数据空间，可以作为通用寄存器使用， 高效的实现在页面直接交换数据。

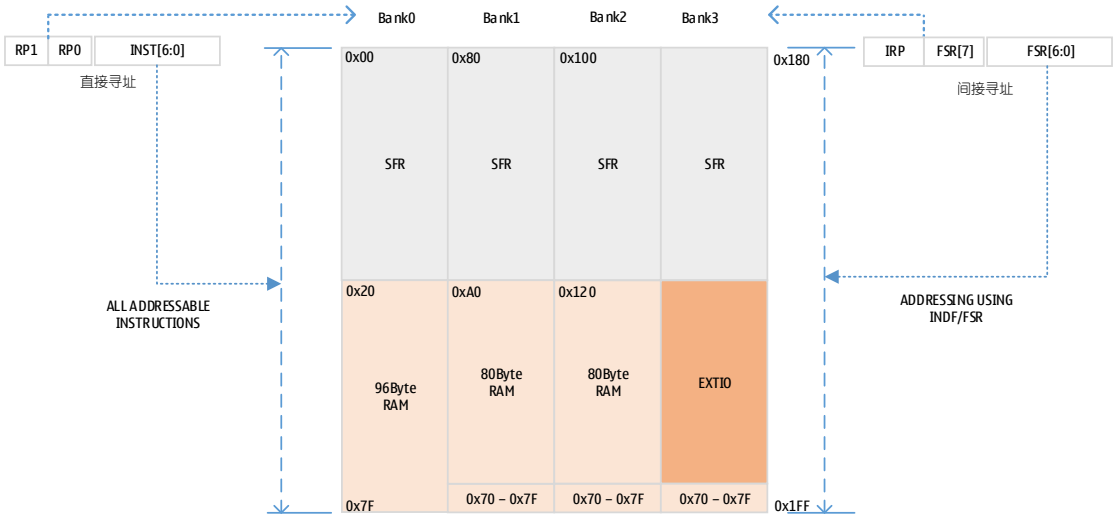
STATUS 寄存器中的 RP[1:0]位用于选择当前访问的页面，当前页面内的所有地址，都可以听过指令直接访问。与使用间距寻址寄存器 INDF/FSR 的间距寻址模式相比， 直接访问更加高效。

RP[1:0]与当前页面的对应关系：

STATUS[6:5] = RP[1:0] 地址：0x003	功能描述
00	选择当前页面 0
01	选择当前页面 1
10	选择当前页面 2
11	选择当前页面 3

[说明：使用 C 语言编程，无需管理页面，编译器会自动根据所访问的寄存器切换页面。这样虽然方便，但也会因此产生多余的页面切换的代码。写代码时可以通过将同一页面的寄存器分组操作减小页面切换]

LGT8F690A 数据存储空间映射以及访问示意图：



MIC8S 实现了直接/间接两种寻址方式。在直接访问模式下，通过 STATUS 寄存器的 RP[1:0]位选择访问的页面，页面内的偏移地址直接由指令给出。间接寻址模式使用 STATUS 寄存器的 IRP 位与 FSR 寄存器作为间接寻址的地址寄存器，通过读写 INDF 寄存器，实现间接数据访问。直接/间接模式都可以访问到所有的寄存器以及 RAM 空间。需要特别指出对 RAM 空间读操作访问，需要两个指令周期才能完成。

LGT8F690A 实现了 256 字节的内部 RAM 空间，其中 96 字节映射到第 0 页的 0x20-0x7F 地址范围；另外 160 字节分为两个 80 字节分别映射到第 1 页的 0xA0-0xEF 以及第 2 页的 0x120-0x16F。第 1/2/3 页位于最高地址的 16 个字节，被统一映射到第 0 页的 0x70-0x7F 区间。因此，这里的 16 个字节在所有页都被映射到相同的物理单元，可用于实现快速的变量访问或者共享。

特殊功能寄存器

特殊功能寄存器包括内核控制寄存器和外设控制寄存器。其中内核控制寄存器被分配在所有页的相同偏移地址。外设寄存器用于控制控制各种外设相关的功能。本节将会详细描述内核寄存器定义，外设寄存器相关的定义， 请参考外设相关的章节.

LGT8F690A 特殊功能寄存器

第 0 页		第 1 页		第 2 页		第 3 页	
寄存器	地址	寄存器	地址	寄存器	地址	寄存器	地址
INDF	000h	INDF	080h	INDF	100h	INDIF	180h
TMR0	001h	OPTION_REG	081h	TMR0	101h	OPTION_REG	181h
PCL	002h	PCL	082h	PCL	102h	PCL	182h
STATUS	003h	STATUS	083h	STATUS	103h	STATUS	183h
FSR	004h	FSR	084h	FSR	104h	FSR	184h
PORTA	005h	TRISA	085h	LATA	105h	TRISA	185h
PORTB	006h	TRISB	086h	LATB	106h	TRISB	186h
PORTC	007h	TRISC	087h	LATC	107h	TRISC	187h
	008h		088h		108h		188h
	009h		089h		109h		189h

LGT8F690A 特殊功能寄存器(续)

第 0 页		第 1 页		第 2 页		第 3 页	
寄存器	地址	寄存器	地址	寄存器	地址	寄存器	地址
PCLATH	00Ah	PCLATH	08Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	00Bh	INTCON	08Bh	INTCON	10Bh	INTCON	18Bh
PIR1	00Ch	PIE1	08Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	00Dh	PIE2	08Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	00Eh	PCON	08Eh	EEDATH	10Eh		18Eh
TMR1H	00Fh	OSCCON	08Fh	EEADRH	10Fh		18Fh
T1CON	010h	OSCTUN	090h	WPDA	110h		190h
TMR2/L	011h	PR2/L	091h	WPDB	111h		191h
T2CON	012h	PR2/H	092h	WPDC	112h		192h
SPDR	013h	SPFR	093h	WPUC	113h		193h
SPCR	014h	SPSR	094h	IOCAF	114h	MDLF	194h
TWCR	015h	WPUA	095h	WPUB	115h	MDHF	195h
TWSR	016h	IOCA	096h	IOCB	116h	MDXF	196h
TWDR	017h	WDTCON	097h	IOCBF	117h	0	197h
RCSTA	018h	TXSTA	098h	DACON	118h	WREG	198h
TXREG	019h	SPBRG	099h	VRTUN	119h	DIVF	199h
RCREG	01Ah	SPBRGH	09Ah	CMCON0	11Ah	MULF	19Ah
TMR2/H	01Bh	BAUDCTL	09Bh	CMCON1	11Bh	DAW	19Bh
PR1L	01Ch	ADCON2	09Ch	ADCON3	11Ch	DSW	19Ch
PR1H	01Dh	ADRESH(*)	09Dh	APCON	11Dh		19Dh
ADRESH	01Eh	ADRESL	09Eh	ANSEL	11Eh	TCCR	19Eh
ADCON0	01Fh	ADCON1	09Fh	ANSELH	11Fh	MCUCR	19Fh
RAM 96 字节	020h	RAM 80 字节		RAM 80 字节		扩展 IO 空间	1A0h
		映射到第 0 页 0x70 – 0x7F		映射到第 0 页 0x70 – 0x7F		映射到第 0 页 0x70 – 0x7F	1F0h
	07Fh						1FFh

第 3 页地址空间从 0x1F0 到 0x1EF 为扩展 IO 空间， 这部分空间也同样被分配为外设控制寄存器空间，主要用于寻址 ECCP1/ECCP2 模块的控制寄存器。寄存器的相信定义，请参考相关章节。

第 3 页扩展 IO 空间寄存器分配:

寄存器	地址	寄存器	地址
ECP1CR0	1A0h	ECP2PR0	1AEh
ECP1CR1	1A1h	ECP2IR0	1AFh
ECP1CR2	1A2h	ECP2CR0	1B0h
	1A3h	ECP2CR1	1B1h
	1A4h	ECP2CR2	1B2h
	1A5h	ECP2CR3	1B3h
ECP1PR0	1A6h	ECP2CR4	1B4h
ECP1IR0	1A7h	ECP2CR5	1B5h
ECP1R1L	1A8h	ECP2DTP	1B6h
ECP1R1H	1A9h	ECP2DTN	1B7h
ECP1R2L	1AAh	ECP2R1L	1B8h
ECP1R2H	1ABh	ECP2R1H	1B9h
ECP1R3L	1ACh	ECP2R2L	1BAh
ECP1R3H	1ADh	ECP2R2H	1BBh
		ECP2R3L	1BCh
		ECP2R3H	1BDh
		ECP2R4L	1BEh
		ECP2R4H	1BFh

LGT8F690A 特殊功能寄存器定义:

特殊功能寄存器: 第 0 页									
名称	地址	位定义							
INDF	0x00	FSR 间接寻址数据							
TMR0	0x01	Timer0 计数寄存器							
PCL	0x02	PC[7:0]							
STATUS	0x03	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
FSR	0x04	FSR[7:0] 间接寻址地址寄存器							
PORTA	0x05	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
PORTB	0x06	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
PORTC	0x07	-	-	-	-	-	-	RC1	RC0
-	0x08	保留							
-	0x09	保留							
PCLATH	0x0A	-	-	-	-	PCLATH[3:0]			
INTCON	0x0B	GIE	PEIE	T0IE	INTIE	RABIE	T0IF	INTF	RABIF
PIR1	0x0C	CCP2IF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
PIR2	0x0D	OSFIF	CMIF	-	EEIF	-	-	-	TWIF
TMR1L	0x0E	16 位 Timer1 计数器低 8 位							
TMR1H	0x0F	16 位 Timer1 计数器高 8 位							
T1CON	0x10	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON
TMR2	0x11	定时器 2 计数器低字节							

T2CON	0x12	-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
SPDR	0x13	SPI 数据收发寄存器							
SPCR	0x14	-	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
TWCR	0x15	TWEN	TWMST	-	-	TACK	RACK	CKPS1	CKPS0
TWSR	0x16	TXP	RXK	TXD	TXS	RXP	TXK	RXD	RXS
TWDR	0x17	I2C 收发数据寄存器							
RCSTA	0x18	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREG	0x19	USART 数据发送寄存器							
RCREG	0x1A	USART 数据接收寄存器							
TMR2H	0x1B	定时器 2 计数器高字节							
PR1L	0x1C	定时器 1 周期寄存器低字节							
PR1H	0x1D	定时器 1 周期寄存器高字节							
ADRESH	0x1E	ADC 转换结果高字节							
ADCON0	0x1F	ADFM	VCFG0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON

特殊功能寄存器: 第 1 页									
名称	地址	位定义							
INDF	0x80	间接寻址数据寄存器							
OPTION_REG	0x81	$\overline{RAPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
PCL	0x82	程序地址指针低 8 位							
STATUS	0x83	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
FSR	0x84	FSR[7:0] 间接寻址地址寄存器							
TRISA	0x85	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
TRISB	0x86	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	0x87	-	-	-	-	-	-	TRISC1	TRISC0
-	0x88								
-	0x89								
PCLATH	0x8A	-	-	-	-	PCLATH[3:0]			
INTCON	0x8B	GIE	PEIE	T0IE	INTE	RAIE	T0IF	INTF	RAIF
PIE1	0x8C	CCP2IE	ADIE	RCIE	TXIE	-	CCP1IE	TMR2IE	TMR1IE
PIE2	0x8D	OSFIE	CMIE	-	EEIE	-	-	-	-
PCON	0x8E	-	SWDD	TMR0D	LVRE	DPSM1	DPSM0	PORF	BORF
OSCCON	0x8F	TSSM	IRCF2	IRCF1	IRCF0	OSTS	HTS	LTS	SCS
OSCTUNE	0x90	RCM 校准位							
PR2L	0x91	TMR2 定时周期寄存器(LSB)							
PR2H	0x92	TMR2 定时周期寄存器(MSB)							
SPFR	0x93	RDFULL	RDEMPT	RDPTR1	RDPTR0	WRFULL	WREMPT	WRPTR1	WRPTR0
SPSR	0x94	SPF	WCOL	-	-	-	DUAL	-	SPI2X
WPUA	0x95	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
IOCA	0x96	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
WDTCON	0x97	-	-	-	WDTPS3	WDTPS2	WDTPS1	WDTPS0	SWDTEN
TXSTA	0x98	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
SPBRG	0x99	USART 波特率设置寄存器低字节							

SPBRGH	0x9A	USART 波特率设置寄存器高字节							
BAUDCTL	0x9B	ABDOVF	RCIDL	RDTP	SCKP	BRG16	IREN	WUE	ABDEN
ADCON2	0x9C	ADOF	OFEN	SPD	SPN	ADTEN	ARSB	ADTS1	ADTS0
ADRESH	0x9D	ADC 转换结果高字节							
ADRSEL	0x9E	ADC 转换结果低字节							
ADCON1	0x9F	VCFG1	ADPS2	ADPS1	ADPS0	DIFS1	DIFS0	VDS1	VDS0

特殊功能寄存器: 第 2 页									
名称	地址	位定义							
INDF	0x100	FSR 间接寻址数据							
TMR0	0x101	Timer0 计数寄存器							
PCL	0x102	PC[7:0]							
STATUS	0x103	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
FSR	0x104	FSR[7:0] 间接寻址地址寄存器							
LATA	0x105	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
LATB	0x106	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
LATC	0x107	-	-	-	-	-	-	LATC1	LATC0
-	0x108								
-	0x109								
PCLATH	0x10A	-	-	-	-	PCLATH[3:0]			
INTCON	0x10B	GIE	PEIE	T0IE	INTIE	RABIE	T0IF	INTF	RABIF
EEDAT	0x10C	EEP 数据寄存器低字节							
EEADRL	0x10D	EEP 地址寄存器低字节							
EEDATH	0x10E	EEP 数据寄存器高字节							
EEADRH	0x10F	EEP 地址寄存器高字节							
WPDA	0x110	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0
WPDB	0x111	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
WPDC	0x112	-	-	-	-	-	-	WPDC1	WPDC0
WPUC	0x113	-	-	-	-	-	-	WPUC1	WPUC0
IOCAF	0x114	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
WPUB	0x115	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
IOCB	0x116	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
IOCBF	0x117	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
DACON	0x118	DAEN	DAOE	DAV5	DAV4	DAV3	DAV2	DAV1	DAV0
VRTUN	0x119	1.5V/2.56V 内部参考校准寄存器							
CMCOM0	0x11A	CMON	COUT	CMOE	CMPOL	-	CPS	CNS1	CNS0
CMCON1	0x11B	-	-	CFS1	CFS0	T1ACS	CMHYS	T1GSS	CMSYNC
ADCON3	0x11C	DAVS1	DAVS0	VRIS	AMEN	AMID	AMFS2	AMFS1	AMFS0
APCON	0x11D	APEN	DPS1	DPS0	DNS2	DNS1	DNS0	GA1	GA0
ANSEL	0x11E	ANSB7	ANSB6	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0
ANSELH	0x11F	ANSA7	-	ANSA5	-	CKEA3	ANSA2	APCM	ANTM

特殊功能寄存器: 第 3 页									
名称	地址	位定义							
INDF	0x180	间接寻址数据寄存器							
OPTION_REG	0x181	$\overline{RAPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
PCL	0x182	程序地址指针低 8 位							
STATUS	0x183	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
FSR	0x184	FSR[7:0] 间接寻址地址寄存器							
TRISA	0x185	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
TRISB	0x186	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	0x187	-	-	-	-	-	-	TRISC1	TRISC0
-	0x188								
-	0x189								
PCLATH	0x18A	-	-	-	-	PCLATH[3:0]			
INTCON	0x18B	GIE	PEIE	T0IE	INTE	RAIE	T0IF	INTF	RAIF
EECON1	0x18C	EEPGD	-	-	-	WRERR	WREN	EEWR	EERD
EECON2	0x18D	EEP 操作保护寄存器							
	0x18E								
	0x18F								
	0x190								
	0x191								
	0x192								
	0x193								
MDLF	0x194	乘除运算低字节(MDR[7:0])							
MDHF	0x195	乘除运算中间字节(MDR[15:8])							
MDXF	0x196	乘法运算高字节(MDR[23:16])							
-	0x197								
WREG	0x198	内核工作寄存器 W							
DIVF	0x199	除法运算除数寄存器，写 DIVF 同时启动除法运算							
MULF	0x19A	乘法运算乘数寄存器，写 MULF 同时启动乘法运算							
DAW	0x19B	加法运算之后的 10 进制 BCD 码转换							
DSW	0x19C	减法运算之后的 10 进制 BCD 码转换							
-	0x19D								
TCCR	0x19E	X2EN	-	T2CF	T2CS	-	-	-	T1CLS
MCUCR	0x19F	PPLP1	PPLP0	IRLD	IFAIL	DIVCY	TCYC1	TCYC0	RSTD

特殊功能寄存器: 第 3 页扩展 IO 空间		
名称	地址	位定义
ECP1CR0	0x1A0	ECCP1 控制寄存器 0
ECP1CR1	0x1A1	ECCP1 控制寄存器 1
ECP1CR2	0x1A2	ECCP1 控制寄存器 2
-	0x1A3	
-	0x1A4	
-	0x1A5	
ECP1PR0	0x1A6	ECCP1 引脚配置寄存器 0
ECP1IR0	0x1A7	ECCP1 中断寄存器 0
ECP1R1L	0x1A8	ECCP1 通道 1 捕获/占空比寄存器低字节
ECP1R1H	0x1A9	ECCP1 通道 1 捕获/占空比寄存器高字节
ECP1R2L	0x1AA	ECCP1 通道 2 捕获/占空比寄存器低字节
ECP1R2H	0x1AB	ECCP1 通道 2 捕获/占空比寄存器高字节
ECP1R3L	0x1AC	ECCP1 通道 3 捕获/占空比寄存器低字节
ECP1R3H	0x1AD	ECCP1 通道 3 捕获/占空比寄存器高字节
ECP2PR0	0x1AE	ECCP2 引脚配置寄存器 0
ECP2IR0	0x1AF	ECCP2 中断寄存器 0
ECP2CR0	0x1B0	ECCP2 控制寄存器 0
ECP2CR1	0x1B1	ECCP2 控制寄存器 1
ECP2CR2	0x1B2	ECCP2 控制寄存器 2
ECP2CR3	0x1B3	ECCP2 控制寄存器 3
ECP2CR4	0x1B4	ECCP2 控制寄存器 4
ECP2CR5	0x1B5	ECCP2 控制寄存器 5
ECP2DTP	0x1B6	ECCP2/T2Cn 通道死区时间寄存器
ECP2DTN	0x1B7	ECCP2/T2CnN 通道死区时间寄存器
ECP2R1L	0x1B8	ECCP2 通道 1 捕获/占空比寄存器低字节
ECP2R1H	0x1B9	ECCP2 通道 1 捕获/占空比寄存器高字节
ECP2R2L	0x1BA	ECCP2 通道 2 捕获/占空比寄存器低字节
ECP2R2H	0x1BB	ECCP2 通道 2 捕获/占空比寄存器高字节
ECP2R3L	0x1BC	ECCP2 通道 3 捕获/占空比寄存器低字节
ECP2R3H	0x1BD	ECCP2 通道 3 捕获/占空比寄存器高字节
ECP2R4L	0x1BE	ECCP2 通道 4 捕获/占空比寄存器低字节
ECP2R4H	0x1BF	ECCP2 通道 4 捕获/占空比寄存器高字节

STATUS – 状态寄存器

STATUS 寄存器包含了 ALU 的算术状态，复位标志以及分页模式下的页选择控制；STATUS 寄存器可以作为任意指令的目标地址，但如果指令本身会影响到 Z,DC 或者 C 标记位，那么指令对这三位的写将会被禁止。这些位的更新仅依赖于硬件逻辑。另外，$\overline{TO}$/$\overline{PD}$这两位为只读位，因此对指令直接对 STATUS 的操作的结果可能会与预期的结果有所不同。

例如，CLRF STATUS，这条指令将清除 STATUS 的高三位，设置 Z 位。指令执行后，STATUS 寄存器的值将为：000u u1uu (其中 u 是保持不变)。鉴于 STATUS 寄存器的特殊性，建议仅仅使用 BCF/BSF/SWAPF/MOVWF 指令更改 STATUS 寄存器，因为这些指令本身不会对 STATUS 产生附加效果。关于指令对 STATUS 状态位的影响，请参考本手册“指令集速查表”部分。

STATUS – 状态寄存器								
地址: 0x03		默认值: 0001_1000						
Bit	7	6	5	4	3	2	1	0
STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
R/W	R/W	R/W	R/W	R	R	R/W	R	R

Bit	Name	描述
7	IRP	间接寻址(INDF/FSR)页选择位 1： 选择当前页 2/3 (0x100 ~ 0x1FF) 0： 选择当前页 1/0 (0x000 ~ 0x0FF)
6:5	RP1/0	直接寻址(直接指令寻址)页选择位 RP[1:0] = 00：选择第 0 页（0x000~0x07F） RP[1:0] = 01：选择第 1 页（0x080~0x0FF） RP[1:0] = 10：选择第 2 页（0x100~0x17F） RP[1:0] = 11：选择第 3 页（0x180~0x1FF）
4	$\overline{TO}$	定时器溢出标志位 0：看门狗溢出 1：上电复位或者执行 CLRWDT/SLEEP 指令
3	$\overline{PD}$	待机标志位 0：SLEEP 指令 1：上电复位或者执行 CLRWDT 指令
2	Z	零标志 0：算术运算结果为 0 1：算术运算结果非 0
1	DC	BCD 半进位/借位标志
0	C	进位/借位标志 0：加法非进位，减法借位 1：加法进位，减法非借位
备注：STATUS[4:3]复位值仅上电复位和 LVR 复位有效		

OPTION_REG – 外设配置寄存器

OPTIO_REG 寄存器包含 Timer0/WDT 预分频控制，外部中断 INT/RA1 触发沿控制，Timer0 相关控制以及全局上拉控制。

当系统进入深睡眠模式后，只能通过外部中断引脚 RA1/INT 或者外部复位唤醒；外部中断唤醒的电平也需要通过 INTEDG 位控制。

Timer0 与 WDT 共享一个预分频模块，同一时间预分频模块只能够分配给其中一个使用。当预分频分配给 WDT 后，Timer0 计数器为 1:1 的预分频模式。

MICS8S 内核实现了一个专用更新 OPTION_REG 寄存器的 OPTION 指令；OPTION 指令将工作寄存器 W 中的数据更新到 OTPION 寄存器中。

OPTION_REG – 外设配置寄存器								
地址: 0x81					1111_1111			
Bit	7	6	5	4	3	2	1	0
OPTION	$\overline{RABPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述
7	$\overline{RABPU}$	PORTA/B 全局上拉控制位 1 = PORTA/B 全局上拉禁止 0 = PORTA/B 全局上拉有效
6	INTEDG	外部中断/唤醒边沿选择 1：INT/RA1 的上升沿产生中断 0：INT/RA1 的下降沿产生中断
5	T0CS	Timer0 时钟源选择 1：选择 T0CKI 外部输入，Timer0 为计数器模式 0：选择内部系统时钟，Timer0 为定时器模式
4	T0SE	Timer0 时钟触发沿选择 1：T0CKI 的下降沿 0：T0CKI 的上升沿
3	PSA	预分频器分配控制位 1：预分频为 WDT 所有 0：预分频为 Timer0 所有
2:0	PS[2:0]	预分频分频选择位

PS[2:0]	Timer0 Rate	WDT Rate
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

INDF – 间接寻址数据寄存器

INDF 寄存器并没有物理上对应的寄存器实现。INDF 寄存器用于间接寻址操作。任何使用 INDF 作为目标寄存器的指令，间接访问由 FSR 寄存器指向的目标地址。间接读 INDF 寄存器本身将返回 0。通过 FSR 寄存器间接写 INDF 本身也是没有意义的。

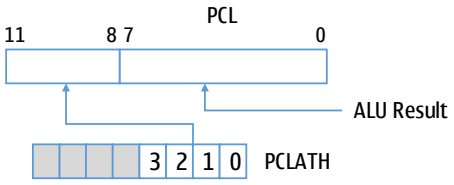
下面我们通过一个实例程序说明如何利用间接寻址进行高效的数据访问；实例程序使用间接寻址方式，清零 0x40 到 0x7F 之间的 RAM 区域：

```
        MOVLW   0x40    ; initialize pointer
        MOVWF   FSR     ; to RAM
NEXT    CLRF    INDF    ; clear INDF register
        INCF    FSR     ; inc pointer
        BTFSS   FSR, 7  ; all done?
        GOTO    NEXT    ; no, go on next
DONE                    ; yes
```

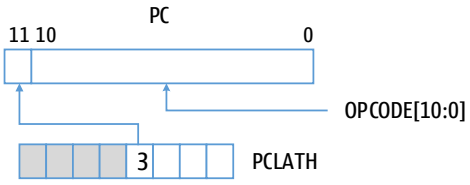
INDF – 间接寻址数据寄存器								
地址: 0x00					默认值: XXXX_XXXX			
Bit	7	6	5	4	3	2	1	0
INDF	INDF[7:0]							
R/W	R/W							
Bit	Name	描述						
7:0	INDF	间接寻址数据寄存器，物理上并没有寄存器实现。读 INDF 将返回 FSR 寻址地址的数据。如果 FSR 指向 INDF 本身，将返回 0x00；						

PCL 与 PCLATH 寄存器

LGT8F690A 实现了 4K 指令字的程序空间，系统复位后，PC 计数器被清零。程序计数器(PC)为 12 位宽。PC 的低字节来自 PCL 寄存器。PCL 寄存器可读/写，PC 的高 4 位来自 PCLATH。下面图例展示内核在不同操作模式下，PC 的更新方式：



将PCL作为目标寄存器的指令



GOTO/CALL

当内核执行以 PCL 为目标寄存器的指令时，PC 的高位(PC[11:8])将会同时从 PCLATH 加载更新，从而可以通过预先设置 PCLATH，然后执行 PCL 更新完成对 PC 指针的更新。

GOTO/CALL 指令执行时，PC[10:0]直接更新为 GOTO/CALL 指令的操作数指向的地址。同时 PC 的最高位加载为 PCLATH[3]。

使用修改 PCL 实现程序空间查表算法时， 需要特别注意当查表的地址跨越一个 256 字节的数据边界时，需要同时调整 PCLATH，以保证得到一个完整准确的 PC 指针。

PCL – PC 低字节寄存器								
地址: 0x02					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
PCL	PCL[7:0]							
R/W	R/W							
Bit	Name	描述						
7:0	PCL	PCL 实时反映当前运行的 PC 低字节的值； 程序可以通过直接修改这个寄存器更改当前指令的执行流程，更新 PCL 后，当前 PC 的值被更新为[PCLATH:PCL]						
备注：更新 PCL 前，请确认 PCLATH 的值为所需值。更新 PCLATH 不会改变 PC 的值。								

PCLATH – PC 高位锁存寄存器								
地址: 0x0A					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
PCLATH					PCLATH[3:0]			
R/W	R/W							
Bit	Name	描述						
7:4	-	Unimplemented						
3:0	PCLATH	当使用指令更新 PCL 或者使用 GOTO/CALL 执行程序流程操作时， PCLATH 的值将会根据实际操作加载到 PC 的高字节。						
备注：更新 PCL 前，请确认 PCLATH 的值为所需值。更新 PCLATH 不会改变 PC 的值。								

MCUCR – MCU 控制寄存器

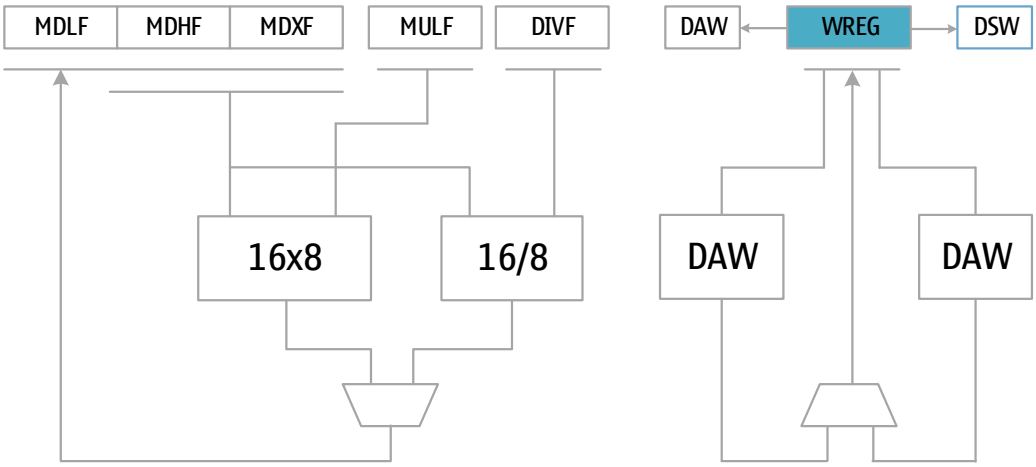
MCUCR 寄存器主要负责对 MCU 内核相关运行的模式进行控制。MCUCR 实例了安全保护机制，更新 MCUCR 寄存器前需要首先向 MCUCR 寄存器写 0x55，之后对 MCUCR 的写操作才会有效。

MCUCR – MCUCR 控制寄存器								
地址: 0x19F					默认值: 0xC6			
Bit	7	6	5	4	3	2	1	0
MCUCR	PPLP1	PPLP0	IRLD	IFAIL	DIVCY	TCYC1	TCYC0	RSTD
R/W	R/W	R/W	W/0	R/0	R/W	R/W	R/W	R/W
Bit	Name	描述						
7:6	PPLP	程序存储器工作模式。详细信息请参考功耗管理部分						
5	IRLD	配置位重新加载控制位。写 1 执行配置信息重载，读返回 0						
4	IFAIL	配置位加载错误指示位。为 1 表示配置加载出错，写无效						
3	DIVCYC	除法运算状态位。为 0 表示除法运算完成						
2:1	TCYC	内核指令周期控制位。 TCYC = 00 ： 1T 指令周期 TCYC = 01 ： 2T 指令周期 TCYC = 1X ： 4T 指令周期						
0	RSTD	RB7 复位禁止控制位。写 1 将禁用 RB7 的外部复位功能。 RB7 默认为外部复位 IO，禁用复位功能后，RB7 可作为通用 IO 使用						

运算加速

- 单周期 16x8
- 8 周期 16/8
- 十进制 BCD 码调整

综述



运算加速框架图

运算加速用于加速四种运算， 包括单周期 16 位数据乘以 8 位数据；8 周期 16 位数据除以 8 位数据以及分别用在加法和减法之后的 10 进制 BCD 码的调整。10 进制的 BCD 码调整用于 10 进制数码显示前对 WREG 寄存器中的数据进行调整。乘除法可以进行相对复杂的运算或者在软件的协助下实现更加复杂的运算。

乘除运算

乘除法运算由乘除运算单元以及相关的操作数寄存器构成。其中包括乘除运算公用的 MDLF/MDHF/MDXF 寄存器以及分别用于提供乘数的 MULF 寄存器以及除数的 DIVF 寄存器。

乘法运算时， 首先通过 MDHF:MDLF 寄存器提供 16 位的被乘数，然后通过向 MULF 寄存器中写入乘数启动乘法运算。一个周期后，乘法运算的结果被更新到 MDXF:MDHF:MDLF 寄存器中，组合为一个 24 位的运算结果。

除法运算时， 通过 MDHF:MDLF 寄存器提供 16 位的被除数，然后通过向 DIVF 寄存器中写入除数启动除法运算。除法运算需要 8 个系统时钟周期（非指令周期）。除法运算开始后，MCUCR 寄存器中的 DIVCY 位被置位，除法运算完成后，DIVCY 位被清零。软件可以通过查询 DIVCY 位等待除法完成。除法运算完成后，除法运算的商更新到 MDHF:MDLF 中，除法运算的余数更新到 8 位的 MDXF 寄存器中。

下面为 C 语言的乘除法运算实现实例。注意代码中使用的 M16F/M24F/M32F。这几个地址以不同的数据类型指针代表 MDXF/MDHF/MDLF 寄存器。详细定义请参考 LGT8F690A 专用头文件。其中 M16F 代表由 MDHF:MDLF 构成的 16 位寄存器；M24F 代表由 MDXF:MDHF:MDLF 构成的 24 位寄存器；M32F 与 M24F 代表同样的寄存器组合，但扩展为一个 32 位数据类型，最高 8 位为补零。

```
#include “lgt8f690a.h”
```

```
// 16x8  函数实现
//  函数返回 32 位乘法运算结果(M32F)
//  也可以返回 24 位结果(M24F)
//  或返回 16 位结果(M16F)
uint32_t wmul16x8 (uint16_t a, uint8_t b)
{
      M16F = a;                                  //  加载 16 位被乘数
      MULF = b;                                  //  加载 8 位乘数并启动乘法运算
      return M32F;                               //  返回 32 位乘法运行结果
}

// 16/8  函数实现
//  返回 16 位除法运行结果
uint16_t fdiv16d8(uint16_t a, uint8_t b)
{
      M16F = a;                                  //  加载 16 位被除数
      DIVF = b;                                  //  加载 8 位除数并启动除法运行
      while(DIVCY == 1);                         //  等待除法运行结束
      return M16F;                               //  返回运算结果
}

//  更加高效实用的 16/8，被除数来自 M16F
//  运算结果同样被更新回 M16F，方便用于递归运算
void __div16d8(uint8_t b)
{
    DIVF = b;
    while(DIVCY == 1);
}
```

段码 LCD 显示应用实例

LGT8F690A 集成除法器在需要显示十进制的段码式 LCD 应用中，能够发挥简洁高效的作用。下面的例子展示如何使用除法器的商和余数功能，高效的完成这一任务：

```
#include “lgt8f690a.h”

uint8_t ones, tens, pers, thus;              //  个位、十位、百位、千位

void displayUpdate(uint16_t data)
{
    M16F = data;                             //  初始化被除数
    _div16d8(10);                            // data = data/10
    ones = MDXF;                             //  个位为余数
    _div16d8(10);                            // data = data/10
    tens = MDXF;                             //  十位为余数
```

```
    _div16d8(10);                    // data = data/10
    pers = MDXF;                     // 百位为余数
    _div16d8(10);                    // data = data/10
    thus = MDXF;                     // 千位为余数
}
```

BCD 码调整

BCD 码调整模块 DSW、DAW 分别用于在减法之后和加法运算之后调整运算结果为 10 进制 BCD 码格式，以便于使用数码管显示。DSW、DAW 的使用也非常简单，DSW/DAW 运算对当前工作寄存器 WREG 中的数据进行调整，需要根据最后更新 WREG 的运算类型使用正确的 DSW/DAW 运行。

使用 DSW/DAW 模块并没有特殊的步骤，在确定使用 DSW 或者 DAW 后， 通过直接读取 DSW/DAW 寄存器即可将当前 WREG 寄存器进行 BCD 码调整，并将调整的结果更新到 DSWF、DAWF 寄存器中。WREG 寄存器的值保持不变。

以下代码示例如何使用 DSW/DAW。代码实现中，我们借用的 C 编译器使用 WREG 传递第一个字节参数的约定。函数参数(a)并没有在函数实现中显式的用到。但我们使用返回 DAWF/DSWF 的方式，触发 DSW/DAW 运行，运算将隐式的使用 WREG 寄存器作为源操作数，对 WREG 中的数据经行对应的运算，并将运算的结果更新到 DAWF/DSWF 寄存器中，然后通过函数返回。

例程中使用的参数传递约定适用于 PICC 以及 XC8 编译器。如果实际使用的编译器不同，请参考编译器使用手册，确认参数传递的方式。如果不能隐式的使用 WREG 寄存器，函数中可以显式的对 WREG 寄存器经行操作。

```
#include “lgt8f690a.h”

// DAW 函数实现，用于加法操作之后
// 函数返回 WREG 寄存器进行 BCD 码调整后的结果
uint8_t fdaw (uint8_t a)
{
    return DAWF;                     // 返回 8 位 DAW 运行结果
}

// DSW 函数实现
// 返回 WREG 寄存器进行 BCD 调整的结果
uint8_t fdsw(uint8_t a)
{
    return DSWF;                     // 返回 8 位 DSW 运行结果
}
```

寄存器定义

MDLF – 乘数运算操作数寄存器

MDLF- 乘数运算操作数寄存器								
地址: 0x194 @bank3					0x00			
Bit	7	6	5	4	3	2	1	0
	MDLF[7:0]							
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	MDLF	乘法运算被乘数最低 8 位以及乘法运算结果的最低 8 位						

MDHF – 乘数运算操作数寄存器

MDHF- 乘数运算操作数寄存器								
地址: 0x195 @bank3					0x00			
Bit	7	6	5	4	3	2	1	0
	MDHF[7:0]							
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	MDHF	乘法运算被乘数高 8 位以及乘法运算结果的中间 8 位						

MDXF – 乘数运算操作数寄存器

MDHF- 乘数运算操作数寄存器								
地址: 0x196 @bank3					0x00			
Bit	7	6	5	4	3	2	1	0
	MDXF[7:0]							
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	MDXF	乘法运算结果最高 8 位以及除法运算的余数						

MULF – 乘数运算乘数寄存器

MULF- 乘数运算乘数寄存器								
地址: 0x19A @bank3					0x00			
Bit	7	6	5	4	3	2	1	0
	MULF[7:0]							
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	MULF	乘法运算乘数寄存器。写 MULF 寄存器同时启动乘法运算						

DIVF – 除法运算除数寄存器

DIVF– 除法运算除数寄存器								
地址: 0x199 @bank3					0x00			
Bit	7	6	5	4	3	2	1	0
	DIVF[7:0]							
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	DIVF	除法运算除数寄存器。写 DIVF 寄存器同时启动除法运算。 除法运算需要 8 个系统时钟周期。运算结果更新到 MDHF:MDLF 寄存器中，余数更新到 MDXF 寄存器中。						

DAW – 10 进制 BCD 码调整寄存器

DAW– 10 进制 BCD 码调整寄存器								
地址: 0x19B @bank3					0x00			
Bit	7	6	5	4	3	2	1	0
	DAW[7:0]							
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	DAW	对 WREG 寄存器的值进行 10 进制 BCD 码调整。用于加法运算之后。						

DSW – 10 进制 BCD 码调整寄存器

DSW– 10 进制 BCD 码调整寄存器								
地址: 0x19C @bank3					0x00			
Bit	7	6	5	4	3	2	1	0
	DSW[7:0]							
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	DSW	对 WREG 寄存器的值进行 10 进制 BCD 码调整。用于减法运算之后。						

MCUCR – MCU 控制寄存器

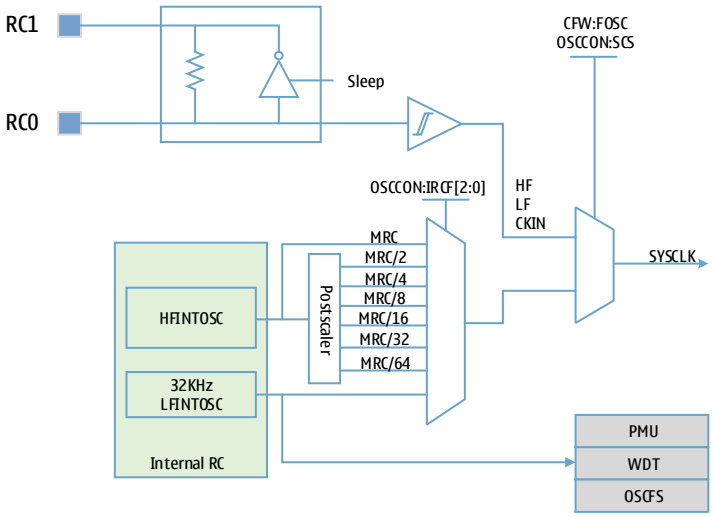
MCUCR– MCU 控制寄存器								
地址: 0x19F @bank3					0x00			
Bit	7	6	5	4	3	2	1	0
	ISAVB	PPLP	IRLD	IFAIL	DIVCY	TCYC1	TCYC0	RSTD
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:4	-	请参考相关章节						
3	DIVCY	除法运算标志，除法运算完成后清零 DIVCY 标志位						
2:0	-	请参考相关章节						

时钟与复位

- 可校准内部主时钟 HFINTOSC 振荡器
  - 倍频模式可输出 32MHz 高速时钟
- 4T/2T/1T 可配置指令周期
- 32KHz 低功耗 LPRC 振荡器
- 外部低频/高频晶振支持
  - 晶振失效检测
- 看门狗定时器复位
- 可编程低压复位电路
- 外部复位输入
- 可配置启动时间

时钟模式

LGT8F690A 的系统时钟可选为内部或者外部时钟源。用户可以根据应用的功耗需求，选择适当的时钟源以达到合理的应用设计。下图说明了系统时钟支持的时钟源配置：



CFW: Configuration Words

系统时钟源可通过配置字设置为来自外部的高频或者低频晶振，外部直接时钟输入以及来自内部多种振荡器时钟源。下面的表格列出了系统支持的全部时钟配置模式。配置字相关的详细定义，请参考本手册“系统配置位”章节。

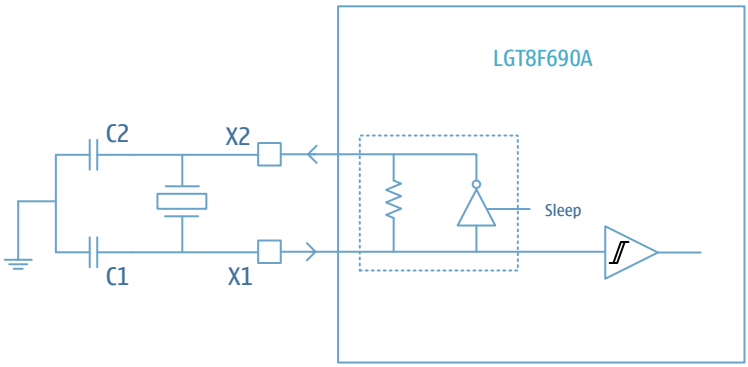
时钟模式	配置描述	其他说明
RCM	系统时钟来自内部高速 HFINTOSC 振荡器	系统运行时钟的频率可以继续通过 OSCCON 寄存器的 IRCF 进行预分频设置
RCK	系统时钟来自内部 32KHz LFINTOSC	只能通过 OSCCON 切换到 RCK
HFOSC	系统时钟来自外部高速晶振	也支持最高到 20MHz 的高频晶振。当系统工作频率高于 16MHz，建议系统提供高于 3.0V 的工作电压
LFOSC	系统时钟来自外部低速晶振	32.768KHz 常用晶振
CLKIN	直接外部时钟源输入(RC0)	

外部时钟源

时钟模式 HFOSC/LFOSC 以及 CLKIN 均为外部时钟输入模式。

HFOSC 外部高频晶振模式，支持外部 2M~20MHz 的外部晶振输入。对于需要精确定时的应用，可以根据需要选择合适的外部晶振。但需要注意，当系统时钟大于 16MHz，建议用户同时将系统保持在高于 3V 的工作电压。同时，对于频率较高的外部晶振，建议在晶振的两端接上 16pF~22pF 左右的电容，这样有利于晶振稳定起振。

LFOSC 外部低速晶振模式，支持外部 32KHz 范围的晶振输入。建议在外部晶振的两端接 16pF 左右的电容，这样便于晶振的稳定起振。



外部晶振与电容的选择：

外部晶振	外接电容
32KHz~4MHz	C1/C2：16pF
4MHz~20MHz	C1/C2：16pF~22pF

当系统时钟配置为外部晶振后，系统在上电/复位以及休眠模式唤醒后，会消耗额外的时间等待晶振稳定。当系统时钟为外部高速晶振，起振时间较快，不会对上电/复位以及休眠唤醒产生明显的影响；如果是外部低速晶振模式，系统需要额外消耗数百毫秒的时间等待晶振稳定。具体等待时间，请参考下面的数据：

晶振模式	附加稳定周期
LPOSC 模式	上电/复位： 休眠唤醒：
HPOSC 模式	上电/复位： 休眠唤醒：

CLKIN 外部有源时钟输入。在应用允许的情况下，可以直接从外部提供一个有源的时钟供系统工作。外部时钟信号直接从 RC0 输入。当系统时钟模式配置为 CLKIN 模式，系统时钟从 RC0 输入，RC1 引脚可以作为通用 GPIO 使用。

对于所有外部晶振输入模式，当系统进入深度休眠模式后，外部晶振将会被硬件自动关闭。用户可以通过寄存器配置选择在休眠模式中开启内部 32KHz 晶振，实现更为灵活的唤醒方式。此部分相关细节请参考本手册“功耗管理”部分。

内部时钟源

对于大部分对时钟精度没有特殊要求的应用， LGT8F690A 提供了丰富的内部时钟资源，用户可以根据需求，在功耗和性能之间做出合理的选择。RCM/RCK 均为内部时钟模式。

RCK 选择内部 32KHz RC 时钟作为系统时钟源。内部 RC32K 非校准时钟，当系统运行电压在 2.0V 到 5.5V 之前变化时，频率将有高于 5%的变化。此 RC32K 仅仅提供一个低频的时钟，可以让系统运行在一个较低功耗下运行。如果应用需要较为精确的 32.768K 时钟源，请选择 LFOSC 模式；

RCM 选择内部高频可校准 RC 振荡器(HFINTOSC)作为主时钟源。HFINTOSC 输出为 16MHz 的中心频率。当 HFINTOSC 使能了倍频模式，将会产生一个对应的 32M 高频输出，这个高频输出主要是提供给定时计数器模块，以产生一个高速高精度的 PWM 输出。HFINTOSC 位可校准时钟源，芯片在出厂测试时，将 16MHz 中心频率的校准写入保留的系统配置空间，系统上电后，将会自动从配置空间加载校准信息，完成对频率的校准工作。晶振校准后，当系统供电电压在 2.0V 到 5.5V 的工作范围内变化时，可以保证±1%的精度。

内部时钟源预分频与多路复用

系统时钟管理模块实现了一个主时钟预分频器，可以进一步对 HFINTOSC 的输出时钟进行预分频。预分频器的输出与内部 32KHz RC 时钟一起，经过多路复位，可以为系统运行提供更加丰富的选择。

主时钟预分频器为一个 5 位预分频单元，时钟源来自内部高频主时钟源(HFINTOSC)，经过预分频器，可以产生 HFINTOSC/2，HFINTOSC/4，....HFINTOSC/64 一共 6 种分频时钟输出。这六种时钟输出加上 HFINTOSC 时钟本身以及内部 32K IRC，一共 8 种时钟选择。这 8 种时钟通过内部多路复用器，在 OSCCON:IRCF[2:0]寄存器的控制下，选择内部系统时钟源(INTOSC)，

内部系统时钟(INTOSC)与来自外部的时钟源，最终通过 OSCCON:SCS 位或者 CFW:FOSC 配置位，选择最终用于驱动系统时钟工作的主时钟源。

OSCCON:SCS 的优先级高于 CFW:FOSC 配置位设置。当 OSCCON:SCS=1 时，系统被限制为只能工作于内部时钟源(INTOSC)，当 OSCCON:SCS=0 时，系统时钟源将由 CFW:FOSC 配置位决定。

OSCCON – 振荡器控制寄存器

OSCCON 寄存器控制着系统时钟以及频率选择。OSCCON 寄存器包括以下功能：

- 内部频率选择位(IRCF)
- 振荡器状态位(HTS/LTS)
- 系统时钟控制位(OSTS/SCS)

OSCCON – 振荡器控制寄存器								
地址: 0x8F					初始状态: 01010000			
Bit	7	6	5	4	3	2	1	0
	-	IRCF2	IRCF1	IRCF0	OSTS	HTS	LTS	SCS
R/W	-	R/W-1	R/W-1	R/W-1	R-1	R-0	R-0	R/W-0
Bit	Name	描述						
7	-	保留						
6:4	IRCF[2:0]	内部时钟选择位						

		111 = RCM 110 = RCM/2 101 = RCM/4 (默认配置) .... 001 = RCM/64 000 = 31KHz IRC
3	OSTS	晶振启动超时状态位 1 = 系统工作于由 FOSC 定义的外部时钟状态 0 = 系统工作于内部时钟状态
2	HTS	内部高速振荡器(HFINTOSC)工作状态 1 = HFINTOSC 输出时钟已稳定 0 = HFINTOSC 输出时钟不稳定
1	LTS	内部低速振荡器(LFINTOSC)工作状态 1 = LFINTOSC 输出时钟已稳定 0 = LFINTOSC 输出时钟不稳定
0	SCS	系统时钟选择 1 = 系统时钟位内部时钟源(INTOSC) 0 = 系统时钟由 CFW:FOSC 配置位控制

OSCTUNE – 内部主振荡器(HFINTOSC)校准寄存器

内部高频主振荡器（HFINTOSC）在出厂时已经进行了校准。用户仍然可以通过 OSCTUNE 寄存器对 HFINTOSC 的输出频率进行调整，以输出合适的系统工作频率。

OSCTUNE 的初始值与系统配置位 RCM 的设置相关，系统上电过程中，初始化模块将根据 RCM 的设置将 OSCTUNE 设置为对应中心频率的校准值。

OSCTUNE 是一个 8 位的可写寄存器，0x00 对应 HFINTOSC 最低输出频率，0xFF 对应 HFINTOSC 输出最高频率。设置 OSCTUNE 仅仅影响当前 HFINTOSC 的中心频率值，对 LFINTOSC 无影响。

OSCTUNE – HFINTOSC 校准寄存器								
地址: 0x90					初始状态: 出厂校准值			
Bit	7	6	5	4	3	2	1	0
	OSCTUNE[7:0]							
R/W	R/W							
Bit	Name	描述						
7:0	OSCTUNE	HFINTOSC 频率校准位 0x00 = 最小输出频率 .... 0xFF = 最大输出频率						

双时钟启动模式

双时钟启动模式针对外部晶振作为主时钟的应用。双时钟启动模式通过最小化外部晶振启动到代码执行之间的时间，获得更低的系统启动功耗。对于一些需要频繁在休眠模式与正常工作模式

至今切换的应用，双时钟启动模式可以避免外部晶振启动到稳定之间较长的等待时间，从而降低了应用周期内的平均功耗。

双时钟启动模式主要是通过允许系统在唤醒之后，首先使用内部时钟(INTOSC)执行一些指令，完成必要检查共工作后，迅速的再次进入休眠模式。

当系统配置为外部晶振工作模式时(LP/HS)，晶振启动定时器(OST)将会被使能。OST 定时器将一般为 1024 个振荡周期，只要当振荡周期计数完成后，系统才开始执行程序。双时钟启动模式通过在唤醒后使用内部振荡器执行程序的方式，最小化 OST 对程序执行产生的影响。当 OST 计数完毕，OSCCON 寄存器的 OSTS 位被置位，系统时钟切换至外部晶振继续工作。

双时钟启动模式配置

使能双时钟启动模式的配置如下：

1. 使能系统配置位 CFW:TSSM = 1
2. 设置 OSCCON:SCS = 0
3. 通过系统配置位 CFW:FOSC 配置系统时钟源为外部晶振

双时钟模式使能后，系统在以下情况时，将进入到双时钟启动模式：

1. 上电复位发生后，在上电定时器超时后；
2. 从休眠模式唤醒后；

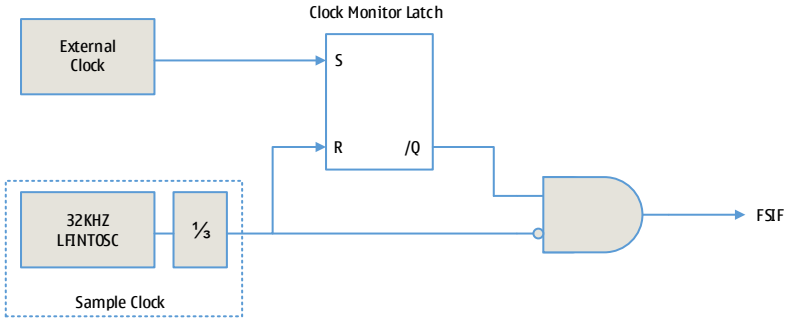
如果系统时钟源被配置为非外部晶振模式，双时钟模式将会自动被禁止。

双时钟启动时序

1. 上电复位或者系统唤醒；
2. 系统使用由 OSCCON:IRCF 设置的内部时钟工作；
3. OST 使能并计数 1024 个周期
4. OST 计数超时；
5. OSCCO:OSTS 位被设置；
6. 系统时钟切换至外部时钟源

外部晶振失效监控

晶振失效监控(FSCM)允许系统在外部晶振故障后，仍然能够继续运行。FSCM 能够检测到 OST 超时后的晶振失效。FSCM 模式通过系统配置位 CFW:FSEN 使能。晶振失效监控仅对于外部晶振模式有效。FSCM 结构如下图所示：



失效保护检测

FSCM 模块通过在采样周期内检测晶振的输出变化来确定晶振是否正常工作。采样频率由内部 32KHz 低频 RC 振荡器经过分频得到。如上 FSCM 结构图所示，FSCM 内部主要检测功能由一个锁存器实现，外部晶振时钟在每一个下降沿置位锁存器输出；采样时钟在其上升沿复位锁存器。当在一个采样周期内，锁存器的输出持续位低电平，FSCM 将会报告一个时钟失效输出。

失效保护操作

当外部晶振发生了时钟失效，时钟管理单元将系统时钟切换到内部振荡器时钟。同时 FSCM 设置晶振失效中断标志位 OSFIF(PIR1 寄存器)。如果此时外设中断使能控制寄存器(PIE1)的 OSFIE 位使能了 FSCM 中断，OSFIF 将会产生一个有效的中断请求。系统在 OSFIF 之后将持续由内部时钟驱动，直到软件成功的重新启动外部晶振并手动切换至外部晶振。

外部晶振失效后，系统将切换至内部时钟源。内部时钟由 OSCCON 寄存器的 IRCF[2:0]决定。这样可以允许在时钟失效发生之前，配置一个合理的备用的内部时钟频率。

时钟失效使用注意事项

FSCM 被设计为检测晶振启动之后的失效状态。晶振启动中包含了一个完整的启动定时器超时计数(OST)。因此 OST 在复位以及唤醒后，都处于有效状态，因此 FSCM 需要在 OST 之后才能够正常工作。当 FSCM 被使能后，双时钟启动模式也同时被使能。因此系统将会在 OST 计数的同时执行程序代码。当用户执行完必须的检测工作后，需要通过检测 OSCCON:OSTS 位来确定外部晶振是否已经完成启动并正确的完成了时钟切换。

复位控制

LGT8F690A 支持以下 4 种复位控制：

1. 上电复位（POR）
2. 看门狗溢出复位（WDT）
3. 外部复位输入（RSTN/RB7）
4. 低电压检测复位（LVR）

系统中大部分寄存器在以上几种复位条件下，都会被复位到一个初始状态。但也有部分寄存器仅仅在 POR 复位或者 LVR 复位有效时才会被复位。此部分相关细节，请参考本章节后面寄存器与复位关系的表格描述。

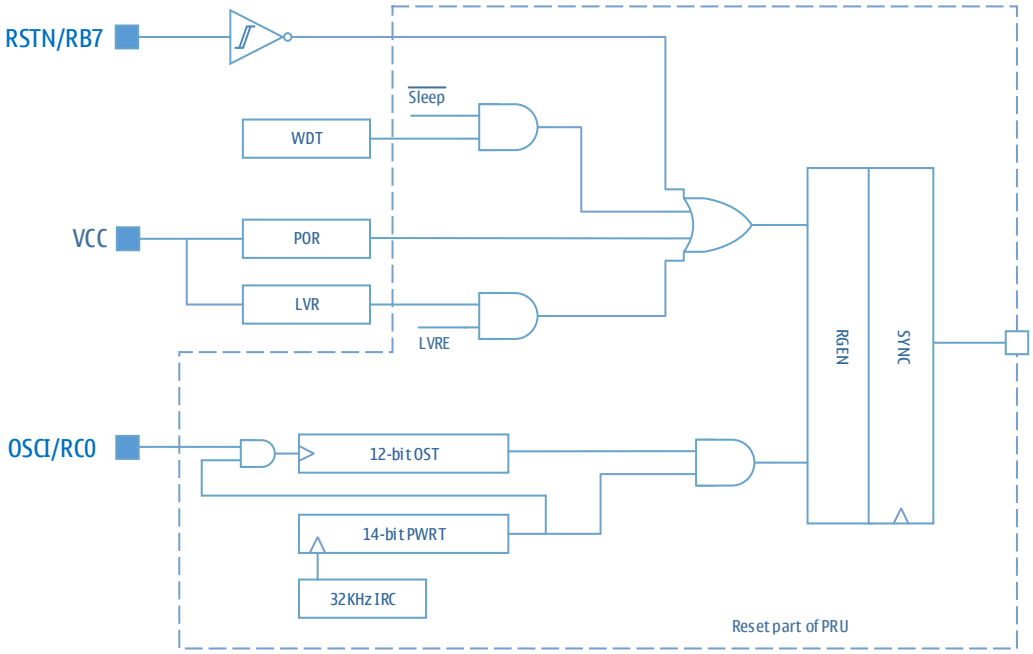
与看门狗正常的溢出复位不同，看门狗唤醒后，不会对寄存器产生影响，因为从休眠模式唤醒，一般是被处理为恢复到正常运行的状态。在系统默认状态下， WDT 是开启状态。用户可以通过 PCON 寄存器或者配置字对 WDT 的运行进行控制。看门狗具体实现，请参考本手册相关章节。

在系统默认状态下，RB7 引脚为外部复位输入。用户可以通过配置字或者 MCUCR 寄存器改变 RB7 的默认状态。关闭 RB7 引脚的外部复位功能后，RB7 可做为通用 I/O 使用。

低电压检测复位（LVR）是 一个阀值可配置的低电压检测模块。用户可以根据应用环境需求，将 LVR 配置为适当的监控阀值，当系统供电（VCC）电压低于预设阀值时，LVR 将产生一个持续的复位信号，将系统强制为复位状态。LVR 模块默认是关闭的，用户可以通过 PCON 寄存器或者配置字控制 LVR 的开启和关闭。LVR 的检测阀值电压只能够通过配置字设置。

STATUS 寄存器中的 TO/PD 位会根据当前系统的复位情况而变化，用户可以通过这两位判定导致系统复位的原因。TO/PD 与复位源的对应关系，请参考本章后续介绍。

下图为系统复位的结构：



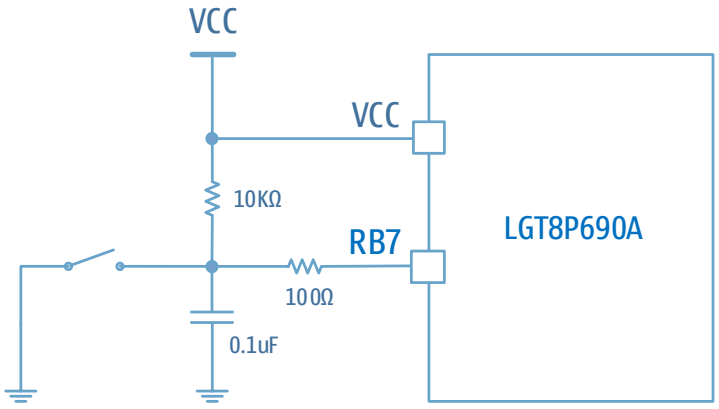
上电复位 (POR)

在系统上电过程中，在 VCC 电压上升到正常的工作电压范围之前，POR 电路将一直保持复位输出有效状态。内部 POR 电路的存在可以省去接在 RSTN/RB7 引脚上的外部复位电路，仅仅使用一个电阻将 RSTN/RB7 与 VCC 相连接即可。在系统工作的过程中， POR 也会持续监控 VCC 的电压变化，当 VCC 电压掉到 POR 阀值以下时，POR 电路产生有效的复位输出，将系统保持在复位状态。

当 LVR 使能后，系统的复位状态除了由 POR 控制，也同时会受到 LVR 电路的控制。LVR 相关特性请参考本手册电气特性部分。

外部复位输入 (RSTN/RB7)

LGT8F690A 的 RB7 引脚默认为外部复位引脚，当此引脚工作在外部复位模式时，内部强制上拉到 VCC。如果需要外接复位电路，建议采用如下参考电路：



低压复位电路 (LVR)

低压复位电路的作用对于供电电压有特殊要求的应用非常重要，比如系统需要运行在较高的频

率，而这个频率需要系统供电保持在一个较高的电压范围内。此时就需要通过开启 LVR 模块，实时检测供电电压的变化。 当供电电压低于设置的阀值时，LVR 输出有效复位，系统保持为复位状态。否则，供电电压下降到无法保证系统在较高的频率运行时，将会因时序问题导致运行错乱。

LVR 默认为关闭状态。用户可以通过配置字开启 LVR 并设置复位阀值。当系统处于休眠模式时，如果系统使能了 LVR，LVR 在休眠模式下仍然会继续保持运行。为获得更低的休眠功耗，可以通过清零配置字的 PMOD 位，休眠控制模块在进入休眠模式前，自动关闭 LVR 模块。LVR 将会在系统被唤醒后自动使能。具体设置，请参考本手册系统配置位相关介绍。

LVR 复位与 POR 复位同属于最高级别的硬件复位，能够复位系统中大部分的寄存器，包括其他复位标志。

启动定时器 (SUT)

LGT8F690A 实现了一个可编程的启动时间定时器，SUT 在 PWRT 中实现，通过控制 PWRT 的定时器实现一个可配置的启动时间。用户可以通过配置字，选择合适的启动时间。

不同的启动时间设置用于特殊的工作环境，比如对于电源上电比较缓慢的应用，POR 在较低的电压点就会释放，如果此时系统进入工作状态，但由于电源上电缓慢，系统就会在一个较低的电压下工作，容易导致系统工作不稳定，造成功能错乱。此时需要将 SUT 配置为较大的时间，这样可以保证在电源上升到合适的电压值之前，系统一直处于复位状态。

另外，对于外部晶振应用，如果晶振本身起振较慢，也建议适当调整 SUT 的配置，保证在系统时钟源切换到外部晶振前，晶振处于稳定状态。

当然，对于要求系统快速启动的应用，也可以通过调整 SUT，达到应用的需要。

启动时间需要通过配置字设置， 系统最终进入正常工作的时间除了与 SUT 相关，还与系统是否启用了外部晶振有关。当系统开启了外部晶振，会在 SUT 之后，附加一个额外的晶振启动时间(OST)。系统的启动时间与 PWRT 以及 OST 的关系，请参考如下列表：

FOSC Mode	SUT Settings	POR or LVR	RSTN or WDT
RCM RCK CLKIN	00	63ms	125us
	01	254ms	
	10	2ms	
	11	16ms	
HFOSC LFOSC	00	63ms + 2048*FOSC	
	01	254ms + 2048*FOSC	
	10	2ms + 2048*FOSC	
	11	16ms + 2048*FOSC	

PCON – 功耗控制寄存器

PCON 主要用于控制系统低功耗模式，PCON 的最低两位为硬件复位标志位，指示当前的复位是上电复位(POR)还是低电压复位(LVR)，PCON 的复位状态位与 STATUS 寄存器中的 TO/PD 标记共同决定着系统中所有复位状态。

PCON 寄存器中也包含 WDT 和 LVR 的使能控制位。WDT 和 LVR 可以通过配置字控制。PCON 提供了一个方便的控制方法，用户可以使用软件对 WDT 或者 LVR 进行控制。

PCON 中包含了两位功耗模式控制位：DPSM1/0。DPSM 用于设置系统的休眠模式，用户需要根据应用的功耗控制需求设置 DPSM，随后执行 SLEEP 指令，内核将进入休眠模式。休眠模式的具体定义和配置使用方法，请参考本手册功耗管理相关章节。

PCON 中的 SWDD 位用于控制编程接口(SWD)。LGT8F690A 的编程接口本身就具有非常安全的数据保护算法。用户仍然可以通过此位禁用所有 SWD 接口的相关操作。

STATUS/PCON 复位状态：

$\overline{POR}$	$\overline{LVR}$	$\overline{TO}$	$\overline{PD}$	Conditions
0	1	1	1	上电复位
U	0	1	1	低压检测复位
U	U	0	U	看门狗复位
U	U	0	0	看门狗唤醒
U	U	U	U	外部复位
U	U	1	0	外部复位唤醒

说明：U = unchanged

PCON– 功耗控制寄存器								
地址: 0x8E					X000_0001			
Bit	7	6	5	4	3	2	1	0
	-	SWDD	TMR0D	LVRE	DPSM1	DPSM0	$\overline{POR}$	$\overline{BOR}$
R/W	-	W/R	W/R	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述
7	-	-
6	SWDD	SWD 接口禁止位，1：禁止 SWD
5	TMR0D	定时器 0 禁止位，1：禁止 Timer0
4	LVRE	LVR 复位使能位，1：使能 LVR 的复位功能
3:2	DPSM1/0	休眠模式控制位 00 ：仅关闭内核运行时钟，用于快速唤醒 01 ：关闭 RCM，系统时钟切换到 32K IRC 1x ：深睡眠模式，关闭所有时钟源
1	$\overline{POR}$	上电复位标志位
0	$\overline{BOR}$	低压检测复位标志位

功耗管理

- 休眠模式控制(DPSM)
- 外设中断以及 WDT 定时唤醒
- TMR1 异步模式定时唤醒
- 最低 1uA@3.3V 待机功耗

综述

LGT8F690A 实现了一个简单高效的功耗管理模块(PRU)，实现自动功耗控制。用户可以根据应用需求，选择合适的功耗管理模式，在满足应用功能以及性能的前提下，实现最为合理的功耗控制。

功耗管理模式主要由 PCON 寄存器以及其他相关配置字控制。LGT8F690A 支持多种时钟源，在应用的性能约束下，建议选择最合适的时钟配置，以获得更低的动态功耗。关于时钟源相关的配置，请参考本手册系统时钟相关章节。

LGT8F690A 的功耗管理模块可以对部分模块进行管理，也有部分模块的管理需要用户自行控制，比如模数转换器，模拟比较器等。如果在休眠模式下这些模块不需要工作，建议将其关闭，以避免不必要的功耗浪费。低压复位模块(LVR)可以通过 PCON 寄存器的 LVRE 位进行控制，也可以通过配置字的 LVRPM 位控制，功耗管理模块可以根据 LVRPM 位的配置，在进入休眠模式前自动关闭 LVR，并在唤醒后自动启动 LVR 模块。LVRPM 相关配置请参考本手册系统配置位章节。

功耗模式

软件使用 PCON 寄存器的 DPSM 位配置休眠模式，然后执行一条 SLEEP 指令进入休眠状态。进入休眠状态后，内核暂停在 SLEEP 指令之后的下一条指令上，其他模块的工作状态以及唤醒方式，与用户的配置有关，下面的表格详细列出休眠模式下模块的工作状况以及唤醒源：

DPSM[1:0]	Core	LVR	WDT	OSC	RCM	RC32K	Analog	Timers	唤醒源
00	X	(1)	(2)	(3)	√	√	(2)	(2)	外部复位 所有中断
01	X	(1)	(2)	(3)	X	√	(2)	(2)	外部复位 所有有效中断
1X	X	(1)	X	X	X	X	X	TMR1 (4)	外部复位(5) RA/B 电平变化

(1)：LVR 在休眠模式下的工作状态由 LVR 的配置位，PCON 的 LVRE 控制位以及配置位 LVRPM 共同决定。如果用户没有通过 LVR 配置字或者 PCON 寄存器使能 LVR，LVR 将一直处于关闭模式。如果系统使能了 LVR，并且 LVRPM=0，LVR 将在进入休眠模式前自动被关闭，并在被唤醒后重新开启 LVR；

(2)：包括 ADC/AC/DAC 等模拟功能模块，在此模式下模块的工作情况取决于用户是否使能了该模块；

(3)：OSC 包括 HFOSC/LFOSC 两种模式，只有在这两种模式下，此时 OSC 处于工作状态；

(4)：TMR1 工作在异步模式，计数时钟来自外部时钟输入或者外部晶振；

(5)：外部复位可以将系统从休眠模式下唤醒，唤醒后系统也同时被复位。

休眠模式由 DPSM 控制，主要分为三个级别：

1. DPSM = 00，这是默认的休眠模式，软件执行 sleep 指令后，内核停止继续取指运行，PC 指针保持指向 SLEEP 之后的指令。PRU 模块关闭内核时钟，SRAM 以及程序存储器进入待机模式。其他外设时钟在此模式下是正常工作的。用户可以根据需要，关闭应用无关的模块以节省功耗。此模式可以通过系统中所有有效的中断唤醒。
2. DPSM = 01，与第一种模式相比，此休眠模式将系统时钟强制切换至内部 32KHz RC 振荡器。并关闭其他所有时钟源。内核停止运行，内核时钟被关闭。其他外设仍然可以继续运行，但此时外设的运行时钟也同时被切换至内部 32KHz RC 振荡器。用户需要特别注意此模式下时钟的变化。此模式可以通过系统中所有可用中断唤醒。用户应该根据需要，关闭其他与应用无关的模块。此模式在获得相对较低的休眠功耗，同时也可以支持外部引脚变化中断唤醒以及 WDT 定时唤醒。
3. DPSM = 1x，当 DPSM[1] = 1 时，LGT8F690A 进入深睡眠模式。此模式下，PRU 关闭所有时钟，包括内部 32KHz RC 振荡器。此模式可以通过以下几种方式唤醒：
   a) 外部复位
   b) 引脚电平变化中断唤醒。
   c) 处于异步模式的 TMR1 溢出中断

动态功耗控制

LGT8F690 增加了动态功耗控制功能。动态功耗控制可以根据系统运行频率配置动态功耗模式，在满足运行性能的同时， 获得更低的运行功耗。

动态功耗控制通过 MCUCR 寄存器的 PPLP 位控制。通过 PPLP 可以实现三种动态功耗模式。 这三种模式的选择由当前系统运行的指令周期决定（注意是指令周期， 不是系统时钟周期）。PPLP 的配置与对应系统周期如下表格所示：

MCUCR : PPLP	指令周期/频率	说明
11	支持 8MHz~20MHz	高速模式，工作电压不低于 4.5V
01	1M~8MHz	低功耗模式，工作电压不低于 2.5V
10/00	500K 以内	省电模式，工作电压不低于 2.0V

低功耗应用注意事项

为在实际应用中获得更低的休眠功耗，需要在应用电路设计以及软件编程时对系统漏电做充分的分析，避免一些不合理的电路配置产生的多余耗电。下面列出一些注意事项，供设计过程参考：

1. 系统功耗(包括动态运行功耗以及休眠功耗)与系统的工作电压最为相关，在应用允许的前提下，能够降低工作电压是获得更低功耗最有效的办法。电压的控制往往被应用本身限制，除了控制电压，用户还可以考虑在满足应用性能需要的前提下，降低系统的运行频率，或者对系统的频率做动态的管理，这样也可以获得更低的动态功耗；
2. 避免浮空的 I/O。I/O 浮空，或者输入状态的 I/O 被外部驱动到一个中间电平，都会因 I/O 内部电路的震荡产生比较大的漏电（一般为几十到上百个 uA）。因此应该避免有浮空的 I/O。LGT8F690A 内部有可控的上拉下拉电阻，用户可以通过寄存器控制将处于输入模式的 I/O 设置到一个合理的电平。
3. 作为输出的 I/O，在进入休眠模式前，应该根据 I/O 外部电路的情况，将 I/O 驱动到合理的电平，避免产生漏电通路。比如对于外部有上拉的 I/O，在休眠模式下应将 I/O 也驱动到高电平，避免上拉电阻对地产生漏电。对于外部没有上下拉的 I/O，应尽量将 I/O 驱动到低电平，因为对于推挽结

构的 I/O，驱动到高电平需要芯片内部消耗多余的电量。

4. 再进入低功耗模式前，关闭所有模拟功能模块以及合理处理相关模拟输入。ADC 模块中的输入分压 VDO 默认情况下并没有关闭，需要在进入休眠模式前通过 ADCON1 寄存器的 VDS 位将其关闭。

低功耗应用例程：下面的例程示例休眠模式的应用。在进入休眠模式去，首先打开 I/O 的上拉电阻，避免浮空的 I/O 产生漏电；通过 PCON 寄存器设置 DPSM 为深睡眠模式。通过 OPTION 寄存器的 INTEDGE 设置 INT 的唤醒电平为低电平。主循环 LOOP 翻转 RA4 引脚，用于指示程序的运行状态。通过判断 RA5 的电平决定是否进入休眠模式。

```
#include “lgt8f690a.h”

int main(void)
{
    WPUB = 0xFF;                        // 使能 RB 内部上拉
    WPUA = 0xFE;                        // 使能 RA 内部上拉
    WPUC = 0x03;                        // 使能 RC 内部上拉
    nRABPU = 0x00;                      // 使能 RA/B 全局上拉控制

    DPSM1 = 1;                          // 使能深休眠模式

    TRISA0 = 0;                         // 设置 RA0 为输出

    IOCA2 = 1;                          // 使能 RA2 的电平变化唤醒功能

    while(1) {
        LATA0 = ~LATA0;                 // RA0 输出指示频率
        NOP(); NOP();
        if(RB0 == 0) {
            SLEEP();                    // 进入休眠模式
        }
     }
}
```

输入输出

- 内部可编程内部上/下拉电阻
- RA/B 组 I/O 电平变化中断
- RA/B 组 I/O 电平变化唤醒
- 2x80mA PWM 驱动 I/O
- 全部 IO 支持 25mA 推挽驱动

综述

LGT8F690A 最多支持 18 路可编程输入/输出端口(SSOP20L 封装)。I/O 被分为 RA/B/C 三组。当引脚的附加功能使能后，其中一些 I/O 将不能作为可编程输入/输出端口使用。

所有 I/O 都支持可编程的上下拉电阻。 RB7 引脚默认作为外部复位引脚，内部强制上拉。通过配置位或者 MCUCR 控制寄存器的 RSTD 禁用 RB7 的外部复位功能后，RB7 可作为通用 IO 使用。

RB7 在系统上电过程中，强制工作为外部复位模式。如果在上电过程中检测到 RB7 被外部强制拉低，配置位对 RB7 的设置将会无效。建议使用 MCUCR 寄存器的 RSTD 位控制 RB7 的工作模式。当使用 RB7 作为通用 IO 时，需要特别注意 RB7 在系统上电过程中的电平状态，因为在系统上电过程中，RB7 为外部复位输入，如果 RB7 有被外部下拉到低电平，系统上电完成后，MCU 仍将处于复位状态。

I/O 的输入/输出方向通过 TRISA/B/C 寄存器控制。当设置 TRISA/B/C 某位为 1，对应的 I/O 为输入模式（默认），清零 TRISA/B/C 位，相应的 I/O 被设置为输出模式。

设置完成 I/O 的方向后，可以通过写 PORT 寄存器或者 LAT 寄存器设置 I/O 的输出状态，或者读 PORT 寄存器获取当前 I/O 的状态变化。

LGT8F690A 为端口增加了 LAT 寄存器，LAT 与 PORT 的区别在于回读的意义不同。LAT 为端口输出寄存器，不反应当前外部端口的实时状态；读 LAT 寄存器得到上一次写 LAT 寄存器的值，而读 PORT 寄存器得到当前端口的外部电平状态。写 PORT 寄存器实际上是改写 LAT 寄存器的值。建议在设置输出 IO 的输出状态时使用 LAT 寄存器，读 IO 的当前电平状态使用 PORT 寄存器。

TRISA/B/C 寄存器仅控制 I/O 的输入/输出方向。即便端口被设置为模拟功能。用户还是需要确保在 I/O 被设置为模拟功能使用时，TRISA/B/C 设置为输入模式。当 I/O 被配置为模拟端口后，读 PORT 寄存器将固定返回 0。被设置为模拟功能的 I/O 也不能产生引脚电平变化中断。

I/O 初始化设置示例：

```
BANKSEL   PORTA     ; select bank of PORTA
CLRF      PORTA     ; init GPIO
BANKSEL   ANSEL     ; select bank of ANSEL
CLR       ANSEL     ; disable analog function

MOVLW     0x3C      ;
MOVWF     TRISA     ; set RA[1:0] as output
```

端口附加功能

LGT8F690A 的端口除 VCC/GND 之外，其他的 I/O 都支持可编程上下拉控制，引脚电平变化中断以及其他外设专属功能。下面章节将详细介绍 I/O 的这些附加功能。

模拟输入功能

LGT8F690A 内部集成丰富的模拟外设，可接受来自外部端口的模拟信号输入。ANSEL/H 寄存器用于控制比较器相关输入端口的状态。上电复位后，大部分 IO 都处于默认的通用 IO 状态。软件可以通过 ANSEL/H 寄存器关闭对应 IO 的数字输入功能。设置 ANSEL/H 寄存器仅仅影响数字 IO 的输入功能，并不影响 IO 正常的输出高低电平。ANSEL/H 的设计用于当 IO 工作为模拟输入功能时，减小 IO 产品的漏电。即使不设置 ANSEL/H，也不同影响 IO 作为模拟端口使用。

RB0/1 同时复用为编程控制接口(SWD/SWC)，默认状态下，RB0/1 作为 SWD 控制器的端口使用。用户可以通过 ANSEL 寄存器将 RB0/1 配置为模拟功能 I/O。此时 SWD 接口功能将被关闭。

当 I/O 被设置为模拟端口功能后，通过 PORTA/C 寄存器读到该端口的值将被固定为 0。但 ANSEL 位并不会影响到该端口的输出功能。因此如果通过 TRISA/C 将端口设置为输出，同时又通过 ANSEL 将端口设置为模拟功能，这样可能会影响到数字以及模拟两种功能的正常运行，需要在使用时特别注意。

弱上拉/下拉电阻

RA/B/C 都具有可编程的内部上拉/下拉电阻。RA/B/C 每组 IO 的上拉电阻可以通过 WPUA/B/C 寄存器单独控制。系统上电后，默认状态下所有的下拉电阻都处于禁用状态。RA/B 组端口(RB7 除外)的上拉被全局上拉控制位(RABPU)禁止。RB7 默认状态下为外部复位输入，内部强制上拉。关闭 RB7 的外部复位功能后，内部上拉可以通过 WPUA 寄存器控制。

当 I/O 处于附加功能模式时，内部的上/下拉电阻将会被自动关掉。

引脚电平变化中断

RA/B 组每个 I/O 都可以通过 IOCA/B 寄存器分别单独配置为可产生引脚电平变化中断的端口。系统的引脚电平变化中断由 INTCON 寄存器的 RABIE 位控制。软件可以通过 INTCON 寄存器 RABIF 位判断是否为引脚电平变化而产生的中断。当检测发生了 RABIF 电平变化中断后，可以通过 IOCAF/IOCBF 寄存器查询是那一个 IO 触发了电平变化中断。

引脚电平变化中断可用于唤醒控制。软件需要在中断复位中通过写零操作清除中断标记。

RC 组 I/O 没有引脚电平变化中断功能，也不能用于引脚电平变化唤醒。

寄存器定义

PORTA – RA 端口数据寄存器

PORTA– RA 端口数据寄存器								
地址: 0x05					XXXX_XXXX			
Bit	7	6	5	4	3	2	1	0
	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	RA7..0	读/写的端口状态						

PORTB – RB 端口数据寄存器

PORTB– RB 端口数据寄存器								
地址: 0x06					XXXX_XXXX			
Bit	7	6	5	4	3	2	1	0
	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	RB7..0	读/写的端口状态						

PORTC – RC 端口数据寄存器

PORTC– RC 端口数据寄存器								
地址: 0x07					XXXX_XXXX			
Bit	-	-	5	4	3	2	1	0
	-	-	-	-	-	-	RC1	RC0
R/W	-	-	-	-	-	-	W/R	W/R
Bit	Name	描述						
7:2	-	Unimplemented						
1:0	RC1..0	读/写的端口状态						

TRISA – RA 端口方向控制寄存器

TRISA – RA 端口方向控制寄存器								
地址: 0x85					1111_1111			
Bit	7	6	5	4	3	2	1	0
	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	TRISA7..0	设置端口的输入/输出方向 1：端口为输入 0：端口为输出						

TRISB – RB 端口方向控制寄存器

TRISB – RB 端口方向控制寄存器								
地址: 0x86					1111_1111			
Bit	7	6	5	4	3	2	1	0
	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	TRISB7..0	设置端口的输入/输出方向 1：端口为输入 0：端口为输出						

TRISC – RC 端口方向控制寄存器

TRISC – RC 端口方向控制寄存器								
地址: 0x87					XX11_1111			
Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	TRISC1	TRISC0
R/W	-	-	-	-	-	-	W/R	W/R

Bit	Name	描述
7:2	-	Unimplemented
1:0	TRISC1..0	设置端口的输入/输出方向 1：端口为输入 0：端口为输出

LATA – RA 端口输出寄存器

LATA – RA 端口输出寄存器								
地址: 0x105					0x00			
Bit	7	6	5	4	3	2	1	0
	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	LATA7..0	当 RA 配置为输出 IO 时，LATA 设置端口 RA 的输出状态 写 PORTA 寄存器与直接写 LATA 寄存器结果相同。						

LATB – RB 端口输出寄存器

LATB – RB 端口输出寄存器								
地址: 0x106					0x00			
Bit	7	6	5	4	3	2	1	0
	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	LATB7..0	当 RB 配置为输出 IO 时，LATB 设置端口 RB 的输出状态 写 PORTB 寄存器与直接写 LATB 寄存器结果相同。						

LATC – RC 端口输出寄存器

LATC – RC 端口输出寄存器								
地址: 0x107					0x00			
Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	LATA1	LATA0
R/W	-	-	-	-	-	-	W/R	W/R
Bit	Name	描述						
7:2	-	保留未用						
1:0	LATC1..0	当 RC 配置为输出 IO 时，LATC 设置端口 RC 的输出状态 写 PORTC 寄存器与直接写 LATC 寄存器结果相同。						

WPDA – RA 端口下拉控制寄存器

WPDA – RA 端口下拉控制寄存器								
地址: 0x110					0000_0000			
Bit	7	6	5	4	3	2	1	0
	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0

R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	WPDA7..0	端口弱下拉控制。 1：使能端口下拉 0：禁止端口下拉						

WPDB – RB 端口下拉控制寄存器

WPDB – RB 端口下拉控制寄存器								
地址: 0x111					0000_0000			
Bit	7	6	5	4	3	2	1	0
	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	WPDB7..0	端口弱下拉控制。 1：使能端口下拉 0：禁止端口下拉						

WPDC – RC 端口下拉控制寄存器

WPDC – RC 端口下拉控制寄存器								
地址: 0x112					XX00_0000			
Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	WPDC1	WPDC0
R/W	-	-	-	-	-	-	R/W	R/W
Bit	Name	描述						
7:2	-	Unimplemented						
1:0	WPDC1..0	RC1/0 下拉电阻使能控制 1: 使能端口下拉 0: 禁用端口下拉						

WPUA – RA 端口上拉控制寄存器

WPUA – RA 端口上拉控制寄存器								
地址: 0x95					0x00			
Bit	7	6	5	4	3	2	1	0
	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	WPUA7..0	端口弱上拉控制 1：使能端口上拉电阻 0：禁止端口上拉电阻						

WPUB – RB 端口上拉控制寄存器

WPUB - RB 端口上拉控制寄存器								
地址: 0x115					0x00			
Bit	7	6	5	4	3	2	1	0
	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	WPUB7..0	端口弱上拉控制 1：使能端口上拉电阻 0：禁止端口上拉电阻						

WPUC – RC 端口上拉控制寄存器

WPUC- RC 端口上拉控制寄存器								
地址: 0x113					XXXX_XX00			
Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	WPUC1	WPUC0
R/W	-	-	-	-	-	-	W/R	W/R
Bit	Name	描述						
7:2	-	Unimplemented						
1:0	WPUC1..0	端口弱上拉控制 1：使能端口上拉电阻 0：禁止端口上拉电阻						

IOCA – RA 端口电平变化中断控制寄存器

IOCA – 端口 RA 电平变化中断控制寄存器								
地址: 0x96					0000_0000			
Bit	7	6	5	4	3	2	1	0
	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	IOCA7..0	端口电平变化中断控制位 1：使能端口电平变化中断 0：禁止端口电平变化中断						

IOCB – RB 端口电平变化中断控制寄存器

IOCB – 端口 RB 电平变化中断控制寄存器								
地址: 0x116					0000_0000			
Bit	7	6	5	4	3	2	1	0
	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	IOCB7..0	端口电平变化中断控制位						

		1：使能端口电平变化中断 0：禁止端口电平变化中断

IOCAF – RA 端口电平变化标志寄存器

IOCAF – 端口 RA 电平变化标志寄存器								
地址: 0x114					0000_0000			
Bit	7	6	5	4	3	2	1	0
	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	IOCAF7..0	端口电平变化标志位： 1：端口发送电平变化 0：端口未发生电平变化						

IOCBF – RB 端口电平变化标志寄存器

IOCBF – 端口 RB 电平变化标志寄存器								
地址: 0x117					0000_0000			
Bit	7	6	5	4	3	2	1	0
	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	IOCBF7..0	端口电平变化标志位： 1：端口发送电平变化 0：端口未发生电平变化						

中断控制

- 外部中断(RA1/INT)
- RA/B 引脚电平变化中断
- 比较器中断
- ADC 转换完成中断
- Timer0/1/2 溢出中断
- 输入俘获中断
- COG 中断
- EUART 外设中断
- 外部晶振失效中断

综述

LGT8F690A 支持多种内外部中断源。中断为硬件中断，由不同的硬件模块触发。中断控制寄存器(INTCON)与中断使能寄存器 PIE1/2 控制中断的使能与禁止；外设中断请求寄存器(PIR1/2)记录了每种中断的请求标记。

每个模块的中断功能可以通过中断控制寄存器单独禁用。中断控制器寄存器(INTCON)中的 GIE 为全局中断使能控制为。只有通过设置 GIE 使能了全局中断，MIC8S 内核才会在执行过程中响应中断请求。

当系统响应了中断请求，将会执行如下的操作：

1. 硬件自动清零全局中断使能位 GIE，禁止响应后续的中断请求；
2. 当前 PC 作为返回地址被压入堆栈寄存器；
3. PC 载入中断对应的向量地址；

中断服务在执行到 RETFIE 指令后退出，RETFIE 指令同时也置位全局中断使能位 GIE。中断服务程序通过查询中断标志确定中断类型。中断状态位必须在中断得到响应后由软件清零，以避免中断被多次响应。

无论是否使能了模块本身的中断使能位或者 GIE 位，中断标记位都会正常工作。使用指令清零 GIE 位后，后续中断请求将不会被响应。后续发生的中断将保持请求状态直到下次 GIE 被使能，或者使用软件强制清除中断标记位。

外设中断控制的细节介绍，请参考本手册外设相关的章节。

中断向量

LGT8F690A 分别为软/硬中断分配了两个中断向量。当有效的中断请求发生后，当前 PC 被压栈，PC 被更新为中断对应的向量地址处开始执行。

LGT8F690A 中断向量分配如下：

中断类型	中断向量地址
硬件中断(外设请求)	0x004

由于系统仅为硬件中断分配了一个向量地址，因此需要用户在中断服务中通过查询中断标记位区分中断源，并在中断服务中分别进行处理。

下面的代码示例如何使用 LGT8F690A 的中断功能：

```
                ORG     000h            ; reset vector
000h            GOTO    MAIN            ; address 0x000, jump to main code
001h
                ORG     0x004           ; point to vector of hardware interrupt
004h            GOTO    HISR            ; server routine for hardware interrupt
005h    MAIN                            ; main code start here
005h            BCF     OPTION_REG, 6   ; interrupt on falling edge of RA1/INT pin
006h            MOVLW   90h             ; enable GIE and INTIE
007h            MOVWF   INTCON          ; and clear up interrupt flag
008h    LOOP    NOP                     ; main code
009h            GOTO    LOOP            ; main loop
00Ah    SISR    NOP                     ; ISR of software vector
00Bh            RETFIE                  ; return from ISR
00Ch    HISR    NOP                     ; ISR of hardware vector
00Dh            RETFIE                  ; return from ISR
        END
```

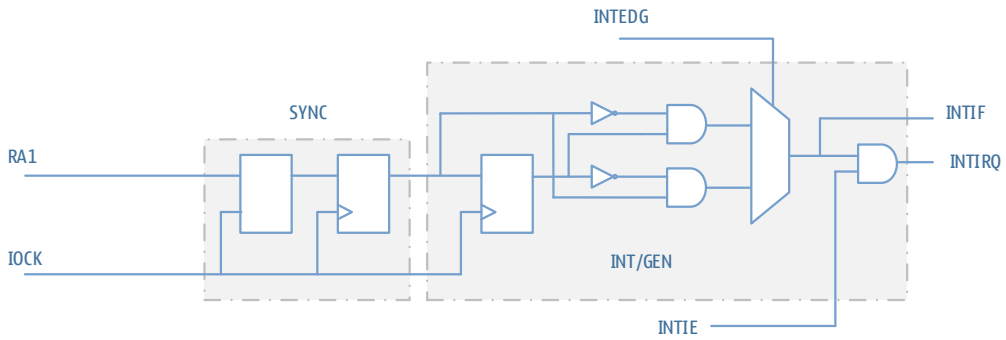
RA1/INT 外部中断

外部中断 INT 为边沿触发，触发边沿通过 OPTION 寄存器的 INTEDG 位设置。当 RA1 上发生有效的边沿的变化，INTCON 寄存器的 INTIF 位被置位。RA1/INT 上的中断功能可以通过 INTCON 寄存器的 INTIE 位禁止。INTIF 位必须在再次使能 GIE 位之前清除，否则会产生重复的中断。

RA1/INT 可用于休眠唤醒。作为唤醒功能，RA1/INT 支持可配置唤醒电平。唤醒电平可以通过 INTEDG 位控制，上升沿对应高电平，下降沿对应低电平。RA1/INT 可作为掉电模式的唤醒输入，需要使能 RA1 的外部中断功能，RA1 才可作为唤醒引脚。

当外部中断输入 INT 上发生电平变化后，由于 I/O 内部同步电路的存在，电平变化会被同步电路后延一个半周期，考虑到进入执行周期的指令不能被中断，因此外部引脚中断的响应时间最大会有 4 个指令周期。在最多 4 个指令周期后，中断向量代码被取指执行。

下图为外部中断输入同步与中断产生示意图：

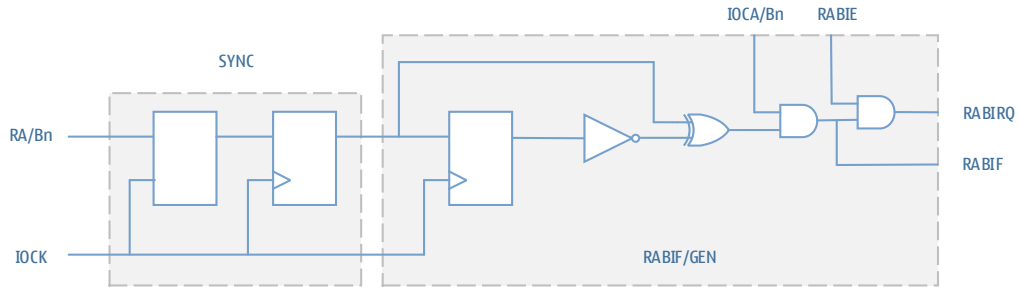


端口电平变化中断

LGT8F690A 的 RA/B 端口都支持电平变化中断。端口上任何类型的电平变化都会产生此类中断。端口电平变化中断可以通过 INTCON 寄存器的 RABIE 位独立控制。单个引脚的电平变化功能也可以通过 IOCA/B 寄存器单独控制。

与外部中断输入类似，GPIO 的输入都通过一个前端的同步电路，因此端口电平变化到内核响应该变化的中断请求，也需要最多 4 个指令周期。

端口电平变化中断电路示意图：



定时器 0 溢出中断

定时器 0 计数器(TMR0)溢出(FFh -> 00h)将会置位 INTCON 寄存器的 T0IF 中断标记位。如果此时使能了 TMR0 溢出中断(INTCON:T0IE)，并同时使能了全局中断 GIE，内核将响应该中断请求，执行硬件中断向量。

定时器 0 的溢出中断可以通过置位/清零 INTCON 寄存器的 T0IE 位实现使能/禁止控制。定时器 0 的相关细节，请查看本手册相关章节。

系统状态保存与恢复

系统响应中断请求后，当前 PC 被压栈保存。一般情况下，中断服务程序会需要使用到某些系统寄存器或者改变一些系统状态。在中断服务中更改这些状态既有可能会破坏主程序的执行环境，因此需要在执行中断服务前保存一些必要的系统状态(比如, W 以及 STATUS 寄存器)，这部分操作只能通过软件实现。

考虑到 LGT8F690A 为分页寻址，为便于数据保存和恢复，并且避免保存操作本身对统状态的影响(比如应该避免更改页寻址位)，应该将数据保存在地址被等同映射到每个页面的存储空间。LGT8F690A 将所用页面的最后 16 个字节全部映射到一个相同的 RAM 空间，非常适合保存全局数据。为避免和主程序应用分配的临时空间冲突，建议使用 RAM 空间 0x70~0x7F 作为保存系统状态的临时空间。

需要在中断服务中保存哪些数据，主要与中断服务程序的实现有关，用户需要详细评估中断服务可能会影响的系统状态。如果这些状态会影响到主程序的运行环境，必须在中断服务中保存，并在中断返回之前恢复。下面的程序片段示例如何在中断服务中保存和恢复系统状态：

```
W_TEMP      EQU     7Fh         ; define W_TEMP register
STATUS_TEMP EQU     7Eh         ; define STATUS_TEMP register

            ORG     04h         ; point to interrupt vector address
HISR        MOVWF   W_TEMP      ; copy W to TEMP regiser
            SWAPF   STATUS, W   ; swap STATUS to W
                                ; using SWAP to avoid status affected
            MOVWF   STATUS_TEMP ; save STATUS to TEMP register
            ......              ;
            (ISR)               ; Insert user code here
            ......              ;
            SWAPF   STATUS_REG, W ; swap STATUS_TEMP into W
            MOVWF   STATUS      ; restore STATUS register
```

```
        SWAPF   W_TEMP, F       ; swap W_TEMP
        SWAPF   W_TEMP, W       ; restore W register
        RETFIE                  ; ISR return
```

寄存器定义

INTCON – 中断控制寄存器

INTCON/INTEN– 中断控制寄存器								
地址: 0x0B/0x8B					0000_0000			
Bit	7	6	5	4	3	2	1	0
	GIE	PEIE	T0IE	INTIE	RABIE	T0IF	INTIF	RABIF
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	GIE	全局中断使能控制位						
6	PEIE	外设中断使能控制 用于全局控制系统中除 T0IE/INTIE/RAIE 之外的其他外设中断，只有在 PEIE 与 GIE 同时有效的前提下，设置 CCP2IE，ADIE，RCIE，TXIF，CCP1IE，CMIE，OSFIE，TMR2IE 以及 TMR1IE 才能够使能对应的中断请求功能						
5	T0IE	定时器 0 溢出中断使能控制位						
4	INTIE	外部中断使能控制位						
3	RABIE	RA/B 引脚电平变化中断使能控制位						
2	T0IF	定时器 0 溢出中断标记位						
1	INTIF	外部中断标记位						
0	RABIF	RA/B 引脚电平变化中断标记位						

PIE1 – 外设中断控制寄存器 1

PIE1 – 外设中断控制寄存器								
地址: 0x8C					0000_0000			
Bit	7	6	5	4	3	2	1	0
	CCP2IE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	CCP2IE	ECCP2 中断使能位						
6	ADIE	ADC 中断使能位						
5	RCIE	EUSART 数据接收完成中断使能						
4	TXIE	EUSART 数据发送完成中断使能						
3	-	-						
2	CCP1IE	ECCP1 中断使能位						
1	TMR2IE	定时器 2 中断使能位						
0	TMR1IE	定时器 1 中断使能位						

PIE2 – 外设中断控制寄存器 2

PIE2 – 外设中断控制寄存器								
地址: 0x8D					0000_0000			
Bit	7	6	5	4	3	2	1	0
	OSFIE	CMIE	-	EEIE	-	-	-	-
R/W	W/R	W/R	-	W/R	-	-	-	-
Bit	Name	描述						
7	OSFIE	晶振失效中断使能						
6	CMIE	模拟比较器中断使能						
5	-	-						
4	EEIE	EEP 编程完成中断使能						
3:0	-	-						

PIR1 – 外设中断标记寄存器 1

PIR1 – 外设中断标记寄存器								
地址: 0x0C					0000_0000			
Bit	7	6	5	4	3	2	1	0
	CCP2IF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	CCP2IF	ECCP2 中断标志位						
6	ADIF	ADC 转换完成中断标志位						
5	RCIF	EUSART 接收完成中断标志位						
4	TXIF	EUSART 发送完成中断标志位						
3	SSPIF	SPI 收发完成标志位						
2	CCP1IF	ECCP1 中断标志位						
1	TMR2IF	定时器 2 溢出中断标志位						
0	TMR1IF	定时器 1 溢出中断标志位						

PIR2 – 外设中断标记寄存器 2

PIR2 – 外设中断标记寄存器 2								
地址: 0x0D					0000_0000			
Bit	7	6	5	4	3	2	1	0
	OSFIF	CMIF	-	EEIF	-	-	-	TWIF
R/W	W/R	W/R	-	W/R	-	-	-	W/R
Bit	Name	描述						
7	OSFIF	晶振实现中断标志位						
6	CMIF	模拟比较器中断标志位						
5	-	-						
4	EEIF	EEP 编程完成标志位						
3:1	-	-						
0	TWIF	IIC2 控制器收发完成标志位						

看门狗定时器

- 时钟来自内部 32KHz LFINTOSC
- 独立 16 位预分频器
- 8 位预分频器(与 TMR0 共享)
- 1ms~268s 超时周期
- 灵活的软件/配置位控制
- 定时唤醒支持

综述

看门狗定时器由一个片内集成的独立 32KHZ RC 振荡器驱动，不要任何外围器件的辅助即可工作。即便在休眠模式下，看门狗定时器仍然可以工作。在正常模式下，看门狗定时器溢出将会产生系统复位。而在休眠模式下，看门狗溢出可将系统从休眠模式中唤醒，系统被唤醒后从休眠之前的状态继续运行，不会产生系统复位。

在默认状态下，看门狗定时器是被使能的。用户可以通过配置字或者 WDTCON 寄存器的 SWDTEN 位进行控制。即使在配置字禁用 WDT 的情况下，仍然可以通过 WDTCON 的 SWDTEN 寄存器将其使能。

看门狗定时器正常模式下超时时间为 18MS。超时时间与温度以及系统供电电压也有关系。WDT 超时时间与电压变化的关系，请参考本手册电气特性部分。

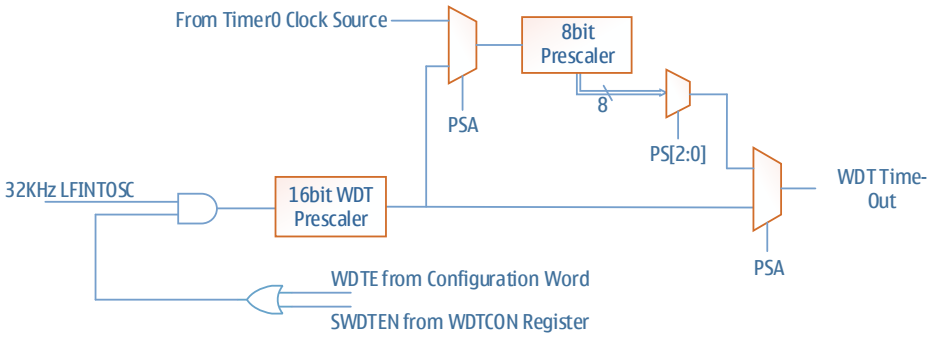
看门狗定时器与 TMR0 共享一个 8 位预分频器。使用预分频配置，最大可以产生正常模式 128 倍的超时时间。预分频的配置通过 OPTION 寄存器的 PS 位设置。OPTION 寄存器的定义请参考本手册 MIC8S 内核部分的介绍。除了与 TMR0 共享的 8 位预分频器， WDT 还有一个独立的 16 位预分频器以及一个 4 位的预分频选择，可以实现最多 12 级最大 65536 倍的超时时间。详细配置请参考 WDTCON 寄存器定义部分。

软件执行 CLRWDT 或者 SLEEP 指令可以清零 WDT 定时器以及预分频器(如果预分频器被分配给 WDT)。清零 WDT 完成喂狗操作。WDT 超时后，STATUS 寄存器中的$\overline{TO}$位被清零。软件可以通过 STATUS 寄存器的$\overline{TO}$/$\overline{PD}$位判定系统复位是否来自 WDT 超时。

Conditions	WDT STATUS
WDTE=0	Cleared
CLRWDT issued	
SLEEP issued	
Oscillator Failed Detected	
Wakeup + System Clock = CLKIN/RCM/RCK	
Wakeup + System Clock = HFSOC/LFOSC	Cleared until the end of OST

Note: 当系统工作在外部晶振模式，OST 定时器被使能。在 OST 计时期间，WDT 模块保持在复位状态，当 OST 超时后，WDT 才开始启动计数。

看门狗定时器结构图：



寄存器定义

OPTION_REG – 外设配置寄存器

OPTION_REG – 外设配置寄存器								
地址: 0x81					初始值: 1111_1111			
Bit	7	6	5	4	3	2	1	0
	$\overline{RABPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述		
7	$\overline{RABPU}$	RA/B 组 I/O 全局上拉禁止控制，低有效		
6	INTEDG	外部中断/唤醒边沿选择		
5	T0CS	Timer0 时钟源选择		
4	T0SE	Timer0 时钟触发沿选择		
3	PSA	预分频器分配控制位 1：预分频为 WDT 所有 0：预分频为 Timer0 所有		
2:0	PS[2:0]	预分频分频选择位		
		PS[2:0]	Timer0 Rate	WDT Rate
		000	1:2	1:1
		001	1:4	1:2
		010	1:8	1:4
		011	1:16	1:8
		100	1:32	1:16
		101	1:64	1:32
		110	1:128	1:64
		111	1:256	1:128

WDTCON – WDT 控制寄存器

WDTCON – WDT 控制寄存器								
地址: 0x18					初始值: XXX0_1000			
Bit	7	6	5	4	3	2	1	0
	-	-	-	WDTPS3	WDTPS2	WDTPS1	WDTPS0	SWDTEN
R/W	-	-	-	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述	
7:5	-	Unimplemented	
4:1	WDTPS	WDT 独立预分频周期选择位	
		WDTPS[3:0]	预分频因子
		0000	1:32
		0001	1:64
		0010	1:128
		0011	1:256
		0100	1:512 (默认设置)
		0101	1:1024
		0110	1:2048
		0111	1:4096
		1000	1:8192
		1001	1:16384
		1010	1:32768
		1011	1:65536
		1100	Reserved
		1101	Reserved
		1110	Reserved
		1111	Reserved
0	SWDTEN	WDT 软件使能/禁止控制位 如果系统配置字的 WDTE=1，WDT 将时钟被使能，不受 SWDTEN 的控制 当系统配置字 WDTE=0 是，软件可以通过 SWDTEN 使能或者关闭 WDT	

定时计数器 0

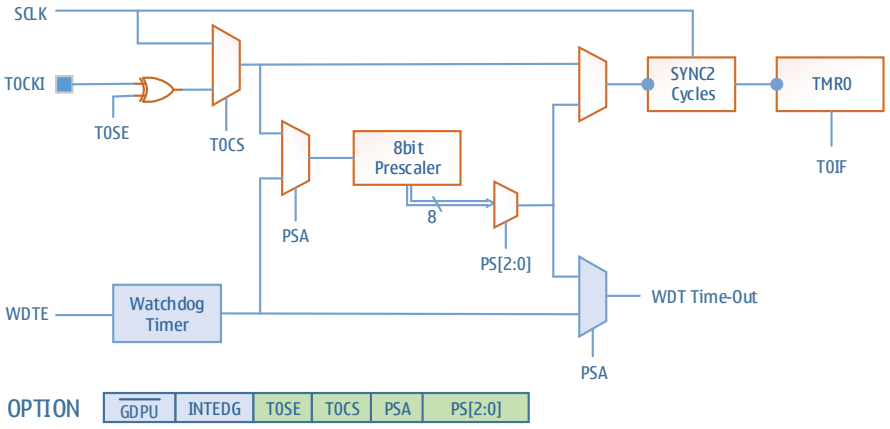
- 8 位定时/计数寄存器(TMR0)
- 8 位预分频器(与 WDT 共享)
- 可编程内部/外部时钟源
- 可编程外部时钟边沿选择
- 溢出中断

综述

TMR0 模块支持两种工作模式：8 位定时器/8 位计数器。

TMR0 计数溢出时产生溢出中断，软件通过 INTCON 寄存器的 T0IF 位查询 TMR0 的中断标志。因为在休眠模式下 TMR0 的接口时钟被关闭，无法产生溢出中断，因此无法支持休眠唤醒。

TMR0 框架图：



定时器模式

当工作于定时器模式，TMR0 的定时器在每个指令周期递增(无预分频)，定时器模式通过清零 OPTION 寄存器的 T0CS 位使能。当对 TMR0 寄存器进行写操作，在写操作发生的接下来两个周期内，TMR0 寄存器被禁止递增。如果需要补充写周期后的定时误差，可以在写 TMR0 之前对将要写的值继续调整。

计数器模式

将 OPTION 寄存器的 T0CS 位置 1，TMR0 工作于计数器模式。在计数器模式下，TMR0 寄存器在 T0CKI 的边沿驱动下递增计数。T0CKI 的边沿通过 OPTION 寄存器的 T0SE 位设置。

可编程预分频器

8 位可编程预分频器为 TMR0 和 WDT 共享。同一时刻预分频器只能被分配给两者之一使用。当 OPTION 寄存器的 PSA 被设置为 1，预分配器被分配给 WDT；否则分配给 TMR0 使用。由于 PSA 位在系统复位后默认为 1，所以预分频器在系统默认的状态下，是被分配给 WDT 使用。

对于 TMR0 模式而言，预分频器支持 8 级分频配置。比例从 1:2 到最大 1:256。分频系数通过 OPTION 寄存器的 PS[2:0]配置。当预分频器被分配给 WDT 时，TMR0 的预分配系数为默认 1:1。

预分频器参数不能够直接的读写访问。当分配给 TMR0 后，任何对 TMR0 寄存器的写操作都将会同时复位预分频器。当预分频器被分配给 WDT 后，CLRWDT 指令将会同时清零 WDT 与预分频器。

因为预分频器可以被分配给 TMR0 或者 WDT，因此在改变预分频配置时，可能会产生非正常的系统复位。下面的程序代码示例如何在 WDT 与 TMR0 之前正确安全的切换预分频器。

预分频器从 TMR0 切换至 WDT:

```
BANKSEL     TMR0            ; select bank 0
CLRWDT                      ; Clear WDT
CLRF        TMR0            ; clear WDT and prescaler

BANKSEL     OPTION_REG      ; select bank 1
BSF         OPTION_REG, PSA ; assign prescaler to PSA
CLRWDT                      ; clear WDT again

MOVLW       b'1111_1000     ; mask prescaler"
ANDWF       OPTION_REG, W   ; bits
IORLW       b'0000_0101     ; set WDT prescaler
MOVWF       OPTION_REG,     ;
```

预分频器从 WDT 切换至 TMR0:

```
CLRWDT                      ; clear WDT and prescaler

BANKSEL     OPTION_REG      ;
MOVLW       B'1111_0000     ; mask TMR0 select and
ANDWF       OPTION_REG, W   ; prescaler bits
IORLW       b'0000_0011     ; set perscaler to 1:16
MOVWF       OPTION_REG      ;
```

TMR0 中断处理

TMR0 在计数溢出(FFh 到 00h)时，T0IF 为在溢出中断发生时。无论系统有没有使能 T0IE，此中断标记位仍然会被置位为 1。在休眠模式下，TMR0 工作时钟被关闭，无法产生用于休眠唤醒中断的信号。

TMR0 的使能与禁止

TMR0 默认状态是被使能的。但默认的 OPTION_REG:T0CS=1 将 TMR0 的时钟源选择为来自外部的 T0CKI 引脚。因此如果应用不需要 TMR0，并且 T0CKI 也没有特定的时钟信号，TMR0 的计数器也等同于停止运行状态，此时不关闭 TMR0 也不会带来明显的多余功耗。 但如果 T0CKI 有比较高频的信号变化，没有关闭的 TMR0 将会带来可观的功耗损失。 此时可以通过置位 PCON 寄存器的 TMR0D 位禁止 TMR0。

定时计数器 1

- 16 位定时计数寄存器 TMR1（TMR1={TMR1H:TMR1L}）
- 16 位周期寄存器 PR1（PR1={PR1H:PR1L}）
- 可选内部或外部时钟源
- 可选外部低频晶振输入（LFOSC）
- 3 位独立预分频器
- 同步/异步模式
- TMR1 计数门控支持：比较器或 T1G 引脚
- 计数溢出中断
- 溢出唤醒（外部时钟模式或异步模式）
- 比较器输出与 TC1 时钟同步

综述

定时计数器 TC1 是一个 16 位的计数器，可工作于定时器模式，也可工作于计数器模式。同时，TC1 既可工作于同步模式，也可工作于异步模式，由 T1CON 寄存器的 T1SYNCB 位来选择。TC1 的计数时钟可来自于内部时钟和外部时钟，由多位寄存器位来选择。TC1 还可工作于门控模式，由外部 T1G 引脚或内部模拟比较器的输出来控制计数。TC1 还可用作编码器模式下的计数。

TC1 的结构框架图如下：

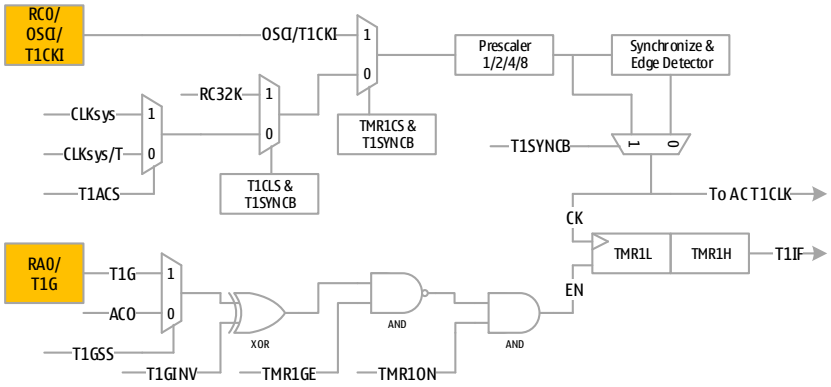


Figure 1 定时器 1 结构框图

时钟源

TC1 的时钟源由多位寄存器位来选择，如下表所示：

Table 1 TC1 时钟源选择表

T1SYNCB	TMR1CS	T1OSCEN	T1CLS	T1ACS	时钟源
0	0	X	X	0	CLKsys/T
0	0	X	X	1	CLKsys
1	1	1	X	X	OSCI
1	1	0	X	X	T1CKI
1	0	X	1	X	RCK

当设置 T1SYNCB 位为 0 时，TC1 工作于同步模式下，时钟源为内部系统时钟或指令时钟，由 T1ACS 位来选择。T1ACS 位为高时，TC1 的时钟源为内部系统时钟；T1ACS 位为低时，TC1 的时钟源为系统指令时钟。指令时钟与系统时钟的关系通过配置字 TCYC 设定，默认为 4T 配置，此时指令时钟为系统时钟的 1/4。详细介绍请参考本手册系统配置位相关信息。
当设置 T1SYNCB 位为 1 时，TC1 工作于异步模式下，时钟源为外部输入或内部低频振荡器的输出时钟 RCK。当 TMR1CS 位为高时，TC1 的时钟源由外部输入，包括外部晶振输入 OSCI 或外部引脚输入 T1CKI，由 T1OSCEN 位来选择。T1OSCEN 位为高，外部晶振被使能，TC1 时钟源为外部晶振输入 OSCI；T1OSCEN 位为低时，外部晶振被禁止，TC1 时钟源为外部引脚输入 T1CKI。当 TMR1CS 位为低而 T1CLS 位为高时，TC1 的时钟源为内部低频振荡器的输出时钟 RCK。

需要注意的是，T1OSCEN 模式只在系统时钟模式为 LFOSC 或者内部 RCM/RCK 时工作。当系统时钟配置为 HFOSC/CLKIN 模式时，应避免使用 T1OSCEN 模式。当使能了外部晶振，在上电复位或者从休眠模式唤醒时，都需要考虑 OST 晶振启动时间延时。

预分频器

TC1 有一个 3 位的预分频器。预分频器支持 1/1，1/2，1/4 以及 1/8 分频，由位于 T1CON 寄存器的 T1CKPS 位来配置。预分频器的溢出用于驱动 TC1 计数。预分频器本身不能够被直接访问。更新 TMR1 计数值寄存器（TMR1H/TRM1L）时会清零预分频器。

同步模式

当设置 T1SYNCB 位为 0 时，TC1 工作在同步模式下，使用内部系统时钟或指令时钟计数。既可以在内部时钟的驱动下进行累加，也可以在外部信号 T1CKI 的上升沿处进行累加。对外部信号 T1CKI 进行上升沿检测前，会先对 T1CKI 进行同步。须注意的是，此同步器在系统休眠模式下无法工作。

异步模式

当设置 T1SYNCB 位为 1 时，TC1 工作在异步模式下，使用非系统时钟进行计数。既可选用内部振荡器产生的 RCK 时钟，也可使用外部晶振产生的 OSCI 时钟，或外部直接输入的 T1CKI 时钟。此时 TC1 可以在系统休眠模式下继续运行。因此，TC1 的溢出中断可用于唤醒休眠模式。
TC1 工作于异步模式时，在读写 TMR1 寄存器时，需要特别注意：
首先，异步模式下访问 TMR1 寄存器有专属的同步逻辑以保证访问的完整有效。但必须考虑到，16 位的 TMR1 计数寄存器由两个 8 位的寄存器组合而成，读取时必须分两次操作，因此有可能在读取高低字节的间隔内，计数器发生溢出，此时读到的数据将不能完整反映当前计数器的状态。
然后，对于写操作，建议用户在写 TMR1 寄存器之前将 TC1 关闭。写操作与计数器的递增操作可能会导致最终写入不可预知的结果。

门控模式

TC1 的门控输入可选用外部引脚 T1G 的输入或者模拟比较器的输出。门控模式可实现用 TC1 测量外部事件。外部事件可以是来自引脚 T1G 或者来自模拟比较器反应的电平变化。可以利用这个功能实现简单的 Delta-Sigma ADC 转换以及其他多种模拟量的测试应用。
TC1 的门控输入的极性可以通过 T1CON 寄存器的 T1GINV 位来设置。用于实现测量高有效或者低有效的信号变化。

编码器接口模式

TC1 可工作在编码器接口模式下，实现对外部输入信号的有效跳变进行向上或向下的计数，具体描述请参考 ECCP1 章节的编码器接口模式相关内容。

溢出中断

当 TMR1 与 PR1 的值发生匹配时，TMR1 的值会回到 0x0000，同时置位 TC1 的溢出中断标志位 TMR1IF 位。软件可通过 PIR1 寄存器访问到 TMR1IF 标志位。当系统同时也使能了 PIE1 寄存器的 TMR1IE 位以及 INTCON 寄存器的 PEIE 和 GIE 位时，内核将响应 TC1 的溢出中断请求，PC 跳转至硬件中断向量

在中断服务程序中， 通过清零 TMR1IF 位来清除 TC1 溢出中断，同时 TMR1 计数寄存器也应在下一次使能中断之前清零。

TC1 支持在休眠模式下工作，此时需要将 TC1 设置为异步计数模式。使能 TC1 溢出中断才能支持唤醒。当计数器发生溢出时，将系统从休眠模式唤醒，继续执行指令。如果使能了全局中断，内核将响应溢出中断请求。

比较器输出同步

TC1 计数器时钟源可用于同步比较器输出，此功能通过 CMCON1 寄存器的 CMSYNC 位来使能。
当使用比较器输出作为 TC1 的门控输入时，比较器需要与 TC1 进行同步，这样可以确保 TC1 能够完整的记录比较去的输出变化。更多信息请参考比较器相关章节。

寄存器定义

T1CON – 定时器 1 控制寄存器

T1CON – 定时器 1 控制寄存器								
地址：0x10					X0000_0000			
Bit	7	6	5	4	3	2	1	0
	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNCB	TMR1ON
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	T1GINV	TMR1 门控信号极性反向控制位； 1：TMR1 门控高有效 0：TMR1 门控低有效						
6	TMR1GE	TMR1 使能控制位，当 TMR1ON 位为 1 时才有效； 1：TMR1 在门控有效时工作 0：TMR1 的工作不受门控信号控制						
5:4	T1CKPS	TMR1 计数时钟预分频器系数选择位； 00：1/1 01：1/2 10：1/4 11：1/8						

3	T1OSCEN	LFOSC 使能控制位； 1：使能外部低频晶振，系统时钟须为 LFOSC 或内部 RCM/RCK 模式 0：禁止外部低频晶振
2	T1SYNCB	TMR1 外部输入时钟同步控制位； 1：异步时钟模式 0：同步时钟模式
1	TMR1CS	TMR1 时钟源选择控制位； 1：时钟输入来自外部 T1CKI 0：时钟来自内部系统时钟或指令时钟，由 CMCON1|T1ACS 位选择
0	TMR1ON	TMR1 使能控制位； 1：TMR1 被使能 0：TMR1 被禁止

TMR1L – 定时器 1 计数值寄存器低字节

TMR1L – 定时器 1 计数值寄存器低字节								
地址：0x0E					X0000_0000			
Bit	7	6	5	4	3	2	1	0
	TMR1L7	TMR1L6	TMR1L5	TMR1L4	TMR1L3	TMR1L2	TMR1L1	TMR1L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	TMR1L	TC1 计数值低字节，TMR1L 与 TMR1H 组成 16 位计数值 TMR1						

TMR1H – 定时器 1 计数值寄存器高字节

TMR1H – 定时器 1 计数值寄存器高字节								
地址：0x0F					X0000_0000			
Bit	7	6	5	4	3	2	1	0
	TMR1H7	TMR1H6	TMR1H5	TMR1H4	TMR1H3	TMR1H2	TMR1H1	TMR1H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	TMR1H	TC1 计数值高字节，TMR1L 与 TMR1H 组成 16 位计数值 TMR1						

PR1L – 定时器 1 周期值寄存器低字节

PR1L – 定时器 1 周期值寄存器低字节								
地址：0x1C					X1111_1111			
Bit	7	6	5	4	3	2	1	0
	PR1L7	PR1L6	PR1L5	PR1L4	PR1L3	PR1L2	PR1L1	PR1L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	PR1L	TC1 周期值低字节，PR1L 与 PR1H 组成 16 位周期值 PR1						

PR1H – 定时器 1 周期值寄存器高字节

PR1H – 定时器 1 周期值寄存器高字节								
地址：0x1D					X1111_1111			
Bit	7	6	5	4	3	2	1	0
	PR1H7	PR1H6	PR1H5	PR1H4	PR1H3	PR1H2	PR1H1	PR1H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	PR1H	TC1 周期值高字节，PR1L 与 PR1H 组成 16 位周期值 PR1						

TCCR – 定时器时钟控制寄存器

TCCR – 定时器时钟控制寄存器								
地址：0x19E					X0000_0000			
Bit	7	6	5	4	3	2	1	0
	X2EN	-	T2CF	T2CS	-	-	-	T1CLS
R/W	W/R	-	R	W/R	-	-	-	W/R
Bit	Name	描述						
7	X2EN	RCM 倍频时钟使能控制位； 1：使能倍频时钟输出 0：禁止倍频时钟输出						
6	-	保留						
5	T2CF	定时器 2 高速时钟标志位； 1：定时器 2 高速时钟设置有效 0：定时器 2 高速时钟设置无效						
4	T2CS	定时器 2 高速时钟使能位； 1：使能定时器 2 高速模式，时钟源为 RCM 倍频时钟 0：禁止定时器 2 高速模式，时钟源为系统指令时钟						
3:1	-	保留						
0	T1CLS	定时器 1 低速时钟使能位； 1：使能定时器 1 低速模式，时钟源为内部 RCK 时钟 0：禁止定时器 1 低速模式，时钟源为系统指令时钟或外部时钟						

定时计数器 2

- 16 位定时计数器 TMR2 {TMR2H:TMR2L}
- 16 位周期寄存器 PR2 {PR2H:PR2L}
- 可编程预分频器(1:1, 1:4, 1:16)
- 可编程后预分频器(1:1 ~ 1:16)
- TMR2 与 PR2 比较匹配中断

综述

定时计数器 TC2 的计数时钟可来自于系统指令时钟和高速时钟，由位于 TCCR 寄存器的 T2CS 位来选择。当设置 T2CS 位为 0 时，TC2 模块的时钟为系统指令时钟；当设置 T2CS 位为 1 时，TC2 模块的时钟为高速时钟，此高速时钟为内部 RC 模块的 2 倍频时钟输出，最大为 32MHz。若选用高速时钟作为 TC2 的时钟源，必须先使能该 2 倍频时钟输出，即设置 TCCR 寄存器的 X2EN 位为 1。

计数时钟输入到 TC2 内部的预分频器。预分频器支持 1/1，1/4 以及 1/16 分频，由位于 T2CON 寄存器的 T2CKPS 位来配置。预分频器的溢出用于驱动 TC2 计数。

TC2 有两种工作模式，单向计数模式和双向计数模式，由位于 ECP2CR3 寄存器的 ECP2_PM 位来选择。

ECP2 模块通过 TC2 来产生 PWM，当设置 ECP2_PM 位为低时，ECP2 模块所产生的 PWM 为边沿对齐模式，相应的 TC2 的计数方式为单向计数，计数的方向为递增或递减，由位于 TCR2 寄存器的 TC2DIR 位来选择。TC2DIR 位为高时，TC2 的计数模式为递增，即计数器从最小值 0 开始递增，累加到最大值 PR2 后又会回到最小值 0 重新递增。TC2DIR 位为低时，TC2 的计数模式为递减，即计数器从最大值 PR2 开始递增，累减到最小值 0 后又会回到最大值 PR2 重新递减。

当设置 ECP2_PM 位为高时，ECP2 模块所产生的 PWM 为中心对齐模式，相应的 TC2 的计数方式为双向计数，计数的方向为递增和递减交替，不受 TC2DIR 为的控制。计数器从最小值 0 开始递增，累加到最大值 PR2 后开始递减，累减到最小值 0 后又开始递增，如此反复。

当 TMR2 与 PR2 的值发生匹配时，TC2 的后分频器递增。后分频器的分频系数从 1:1 到 1:16 可选，由位于 T2CON 寄存器的 TOUTPS 位来配置。当 TC2 的后分频器发生溢出时，会置位 TC2 的中断标志位 T2IF 位。软件可通过 PIR1 寄存器访问到 T2IF 标志位。

TMR2 寄存器与 PR2 寄存器支持软件读写访问。在发生任何种类的复位后，TMR2 寄存器被复位到 0，PR2 寄存器被复位到 0xFFFF。

预分频器和后分频器不能被软件直接访问，在以下情况发生时，将清零分频计数器：

- 写 TMR2 寄存器，包括 TMR2L 和 TRM2H；
- 写 T2CON 寄存器；
- 任何种类的复位

通过设置 T2CON 寄存器的 TMR2ON=1 使能 TC2 模块；清零 TMR2ON 将禁止 TC2 模块的任何功能。

TC2 的结构框架图如下：

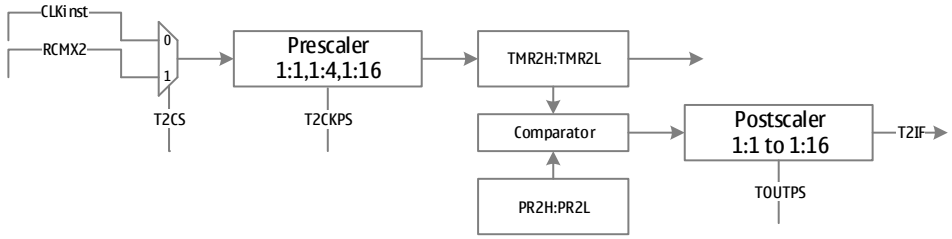


Figure 2 定时器 2 结构框图

寄存器定义

T2CON – 定时器 2 控制寄存器

T2CON – 定时器 2 控制寄存器								
地址：0x12					X0000_0000			
Bit	7	6	5	4	3	2	1	0
	TC2DIR	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	TC2DIR	TMR2 计数方向选择控制位； 1：递增计数 0：递减计数						
6:3	TOUTPS	TMR2 输出后分频系数选择位； 0000：1/1 0001：1/2 0010：1/3 …… 1101：1/14 1110：1/15 1111：1/16						
2	TMR2ON	TMR2 使能控制位； 1：使能 0：禁止						
1:0	T2CKPS	TMR2 计数时钟预分频器系数选择位； 00：1/1 01：1/4 1X：1/16						

TMR2L – 定时器 2 计数值寄存器低字节

TMR2L – 定时器 2 计数值寄存器低字节								
地址：0x11					X0000_0000			
Bit	7	6	5	4	3	2	1	0
	TMR2L7	TMR2L6	TMR2L5	TMR2L4	TMR2L3	TMR2L2	TMR2L1	TMR2L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						

7:0	TMR2L	TC2 计数值低字节，TMR2L 与 TMR2H 组成 16 位计数值 TMR2

TMR2H – 定时器 2 计数值寄存器高字节

TMR2H – 定时器 2 计数值寄存器高字节								
地址：0x1B					X0000_0000			
Bit	7	6	5	4	3	2	1	0
	TMR2H7	TMR2H6	TMR2H5	TMR2H4	TMR2H3	TMR2H2	TMR2H1	TMR2H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	TMR2H	TC2 计数值高字节，TMR2L 与 TMR2H 组成 16 位计数值 TMR2						

PR2L – 定时器 2 周期值寄存器低字节

PR2L – 定时器 2 周期值寄存器低字节								
地址：0x91					X1111_1111			
Bit	7	6	5	4	3	2	1	0
	PR2L7	PR2L6	PR2L5	PR2L4	PR2L3	PR2L2	PR2L1	PR2L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	PR2L	TC2 周期值低字节，PR2L 与 PR2H 组成 16 位周期值 PR2						

PR2H – 定时器 2 周期值寄存器高字节

PR2H – 定时器 2 周期值寄存器高字节								
地址：0x92					X1111_1111			
Bit	7	6	5	4	3	2	1	0
	PR2H7	PR2H6	PR2H5	PR2H4	PR2H3	PR2H2	PR2H1	PR2H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	PR2H	TC2 周期值高字节，PR2L 与 PR2H 组成 16 位周期值 PR2						

TCCR – 定时器时钟控制寄存器

TCCR – 定时器时钟控制寄存器								
地址：0x19E					X0000_0000			
Bit	7	6	5	4	3	2	1	0
	X2EN	-	T2CF	T2CS	-	-	-	T1CLS
R/W	W/R	-	R	W/R	-	-	-	W/R
Bit	Name	描述						
7	X2EN	RCM 倍频时钟使能控制位； 1：使能倍频时钟输出 0：禁止倍频时钟输出						
6	-	保留						
5	T2CF	定时器 2 高速时钟标志位； 1：定时器 2 高速时钟设置有效						

		0：定时器 2 高速时钟设置无效
4	T2CS	定时器 2 高速时钟使能位； 1：使能定时器 2 高速模式，时钟源为 RCM 倍频时钟 0：禁止定时器 2 高速模式，时钟源为系统指令时钟
3:1	-	保留
0	T1CLS	定时器 1 低速时钟使能位； 1：使能定时器 1 低速模式，时钟源为内部 RCK 时钟 0：禁止定时器 1 低速模式，时钟源为系统指令时钟或外部时钟

增强型比较/俘获/PWM 模块(ECCP1)

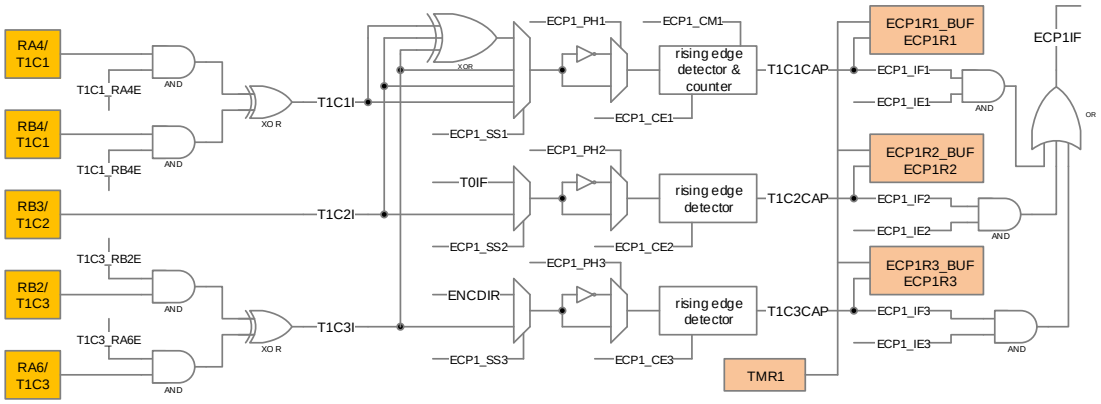
- 支持 Capture 模式，捕捉外部事件，支持 3 路输入通道
- 支持 PWM 模式，产生 3 路频率和占空比可控的 PWM 信号
- 支持编码器接口模式，对外部信号进行带方向的计数

综述

增强型捕获及脉冲宽度调制模块（简称为 ECP 模块）是一个用来计时和控制各种事件的外设。Capture 模式下，可以对外部事件进行计时。PWM 模式下，可以产生频率和占空比可调的 PWM 信号。基于定时计数器 1 的 ECP 模块，被称为 ECP1 模块。本章节系统地描述 ECP1 的捕获，PWM 及编码器模式。本章使用小写字母 n 来代表 1、2、3，分别对应于 ECP1 的三个不同的通道。三个通道所对应的外部引脚为 T1Cn，捕获模式和编码器模式下为输入，PWM 模式下为输出。

Capture 模式

捕获模式的结构图如下所示：



Capture 使能配置

设置位于 ECP1CR0 寄存器的 ECP1_CEn 位为 1 时，会使能该通道的捕获功能，三个通道可同时使能。需注意的是，当该通道的捕获功能开启后，应避免开启其 PWM 功能。

Capture 输入配置

三个通道的捕获源可来自外部引脚 T1Cn 或内部信号，由位于 ECP1CR1 寄存器的 ECP1_SSn 位来选择，选择详情见寄存器描述。

当选用的捕获源来自于外部引脚 T1Cn 时，设置其相应的 TRIS 控制位来配置 T1Cn 引脚作为输入。对通道 1 和通道 3 来说，都有两个外部引脚，除了设置其相应的 TRIS 控制位外，还必须设置其相应的引脚选择配置位。以通道 1 为例，RA4 和 RB4 均为通道 1 的外部引脚 T1C1，若要选用 RA4 作为通道 1 的外部捕获源 T1C1I，须设置位于 ECP1PR0 寄存器的 T1C1_RA4E 位为 1，使能 RA4 作为通道 1 的外部输入；若要选用 RB4 作为通道 1 的外部捕获源 T1C1I，须设置位于 ECP1PR0 寄存器的 T1C1_RB4E 位为 1，使能 RB4

作为通道 1 的外部输入；若设置 T1C1_RA4E 和 T1C1_RB4E 位都为 1，RA4 和 RB4 都被选用为通道 1 的外部输入，此时，通道 1 的外部捕获源 T1C1I 为 RA4 和 RB4 的异或结果；若设置 T1C1_RA4E 和 T1C1_RB4E 位都为 0，RA4 和 RB4 都没有被选用为通道 1 的外部捕获源 T1C1I，此时，通道 1 的外部捕获源为 0，捕获不会发生。

当选用的捕获源来自于内部信号时，可实行特定的捕获。如选用 T0IF 作为捕获源时，可定时进行捕获，捕获的时间长度由产生定时计数器 0 的中断标志位 T0IF 来设定。如选用编码器的方向信号 ENCDIR 作为捕获源时，可在编码器的方向发生翻转时进行捕获。

被选用的捕获源在进行捕获之前，还可进行极性的配置，由位于 ECP1CR2 寄存器的 ECP1_PHn 来选择。

Capture 模式配置

Capture 模式下，对通道 2 和 3 来说，捕获事件为捕获源的任一上升沿，对通道 1 来说，该事件为以下四种当中的一种，由 ECP1CR1 寄存器中的 ECP1_CM1[1:0]位来选择，包括：

- 任一上升沿
- 第 2 个升沿
- 第 3 个上升沿
- 第 4 个上升沿

Capture 模式下，当所选捕获源的事件发生时，定时器 1 的 16 位计数值 TMR1 会被捕获到 16 位缓存器 ECP1Rn_BUF 中去，然后再更新到 16 位捕获寄存器 ECP1Rn 中去，软件通过读取 ECP1RnH:ECP1RnL 寄存器中的值获得捕获值。软件读取 ECP1RnH:ECP1RnL 寄存器时，须先读取高位寄存器 ECP1RnH，再读取低位寄存器 ECP1RnL。因为在软件读取完高位寄存器 ECP1RnH 后，硬件会关闭捕获更新，直到读取完低位寄存器 ECP1RnL 后，硬件才会允许捕获更新，缓存器中的值才会被更新到捕获寄存器中去。

当捕获产生后，位于 ECP1IR0 寄存器的 ECP1_IFn 位会被置位，同时，位于 PIR1 寄存器中的相应中断请求标志位 ECP1IF 也会被置位。该标志位都必须由软件来清零，其中，对 ECP1_IFn 位分别写 0，可分别清除 ECP1_IFn 位；对 ECP1IF 位写 0，会同时清除 ECP1IF 位和三个 ECP1_Ifn 位。在 ECP1RnH 和 ECP1RnL 寄存器的值被读走之前，如果发生新的捕获，原来的捕获值会被新的捕获值覆盖掉。

定时器 1 模式选择

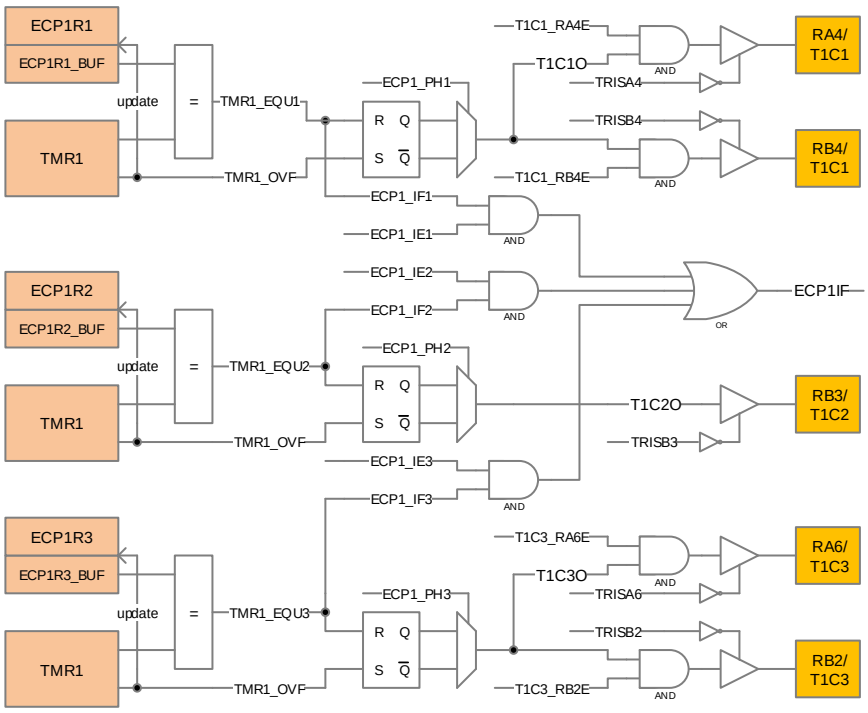
Capture 模式下，定时器 1 须工作在定时模式或同步计数模式。异步计数模式下，捕获可能工作不正常。

软件中断

当 Capture 模式发生改变时，可能会产生一次假的捕获中断。用户需要保持位于 PIE1 寄存器的 ECP1IE 位为零来避免此假中断。此外，还需要在改变 Capture 模式后清零位于 PIR1 寄存器的 ECP1IF 中断标志位。

PWM 模式

PWM 模式的结构图如下所示：



PWM 使能配置

设置位于 ECP1CR0 寄存器的 ECP1_PEn 位为 1 时，会使能该通道的 PWM 功能，三个通道可同时使能。需注意的是，当该通道的 PWM 功能开启后，应避免开启其捕获功能。

PWM 输出配置

三个通道所产生的 PWM 信号用 T1PWMn 来表示，均可输出到外部引脚 T1Cn，由位于 ECP1CR2 寄存器的 ECP1_OEn 位来控制，还须设置其相应的 TRIS 控制位来配置 T1Cn 引脚作为输出。

三个通道所产生的 PWM 信号 T1PWMn 输出到外部引脚之前，还可进行极性的配置，由位于 ECP1CR2 寄存器的 ECP1_PHn 来选择。

PWM 特性配置

PWM 信号 T1PWMn 的周期、占空比和分辨率由以下寄存器来决定：

- PR1H:PR1L
- ECP1RnH:ECP1RnL
- T1CON

其中，PR1H:PR1L 为 16 位定时器 1 的计数最大值，用 PR1 来表示。16 位的占空比寄存器 ECP1RnH:ECP1RnL 还有对应的 16 位占空比缓存器 ECP1RnH_BUF:ECP1RnL_BUF，PWM 信号的占空比最终是由缓存器来决定。更新占空比有两种方式，由位于 ECP1CR0 的 ECP1_DCLM 位来选择。当 ECP1_DCLM 位为 0 时，写入 ECP1RnH：ECP1RnL 寄存器中的内容会直接更新到缓存器中，占空比更新立即完成；当 ECP1_DCLM 位为 1 时，只有在定时器 1 发生溢出时，ECP1RnH：ECP1RnL 寄存器中的内容才会更新好缓冲器中，实现占空比配置的更新。

PWM 周期

PWM 周期由定时器 1 的 PR1 寄存器决定。PWM 周期由下面的公式来计算：

PWM Period = [(PR1) + 1] * Tosc * TMR1PR

其中，PR1 为计数最大值，Tosc 为定时器 1 的计数时钟，TRM1PR 为预分频系数。

当 TMR1 等于 PR1 时，下一个计数周期会发生以下三件事：

- TMR1 被清零
- T1PWMn 被置位（例外：若 PWM 占空比为 0%，PWM 信号将不会被置位）
- PWM 占空比值由 ECP1RnH:ECP1RnL 锁存到其相应的缓存器中（ECP1_DCLM=1 时）

PWM 占空比

PWM 占空比由 16 位寄存器 ECP1RnH:ECP1RnL 的值来决定。

PWM 脉冲宽度由下面的公式来计算：

Pulse Width = (ECP1RnH:ECP1RnL) * Tosc * (TMR1PR)

PWM 占空比由下面的公式来计算：

Duty Cycle Ratio = (ECP1RnH:ECP1RnL) / ((PR1+1))

当 TRM1 和 ECP1RnH:ECP1RnL 发生匹配时，T1PWMn 被清零。

PWM 分辨率

对于给定的周期，分辨率决定了可能的占空比周期数。16 位分辨率有 65536 个连续的占空比周期，10 位分辨率有 1024 个连续的占空比周期，8 位分辨率有 256 个占空比周期。

当 PR1 被设置为 65535 时，拥有最大的 PWM 分辨率，即 16 位。分辨率是 PR1 寄存器的函数，如下面公式所示：

Resolution = log[(PR1 + 1)] / log2 bits

下面两个表格为 PWM 频率和占空比的示例。

PWM 频率和分辨率（Fosc = 16 MHz）

Frequency (KHz)	0.031	0.061	0.244	15.625	62.5	250
TMR1 Prescale	8	4	1	1	1	1
PR1 Value	0xFFFF	0xFFFF	0xFFFF	0x3FF	0xFF	0x3F
Max Resulution	16	16	16	10	8	6

PWM 操作的配置

当配置 ECP1 模块工作在 PWM 模式时，可按照以下流程来执行：

1. 置位相应的 TRIS 位来禁止 PWM 引脚（T1Cn）的输出驱动；
2. 加载 PR1 寄存器来设置 PWM 周期；
3. 加载 ECP1RnH 寄存器和 ECP1RnL 寄存器来设置 PWM 占空比；
4. 加载 ECP1CR2 寄存器来设置 PWM 信号的极性和使能输出；
5. 加载 ECP1CR0 寄存器来设置 ECP1 模块的某个或多个通道为 PWM 模式以及占空比的更新方式；
6. 配置和启动定时器 1，包括：
   - 清零位于 PIR1 寄存器的 TMR1IF 中断标志位
   - 加载位于 T1CON 寄存器的 T1CKPS 位来设置定时器 1 的预分频系数
   - 置位位于 T1CON 寄存器的 TMR1ON 位来使能定时器 1
7. 一个新的 PWM 周期开始后，使能 PWM 输出，即：
   - 等待定时器 1 溢出（即位于 PIR1 寄存器的 TMR1IF 位置位）
   - 清零相应的 TRIS 位来使能 T1Cn 引脚的输出驱动

编码器接口模式

编码器接口模式一般用于伺服马达控制，对外部输入信号的有效跳变进行向上或向下计数，相当于使用了一个带有方向选择的外部时钟。

两个输入 EI1 和 EI2 被用来作为增量编码器的接口。其中，输入信号 EI1 与捕获通道 2 的输入一致，输入信号 EI2 与捕获通道 3 的输入一致，由位于 ECP1CR1 寄存器的 ECP1_SSn 位来选择，选择详情见寄存器描述。

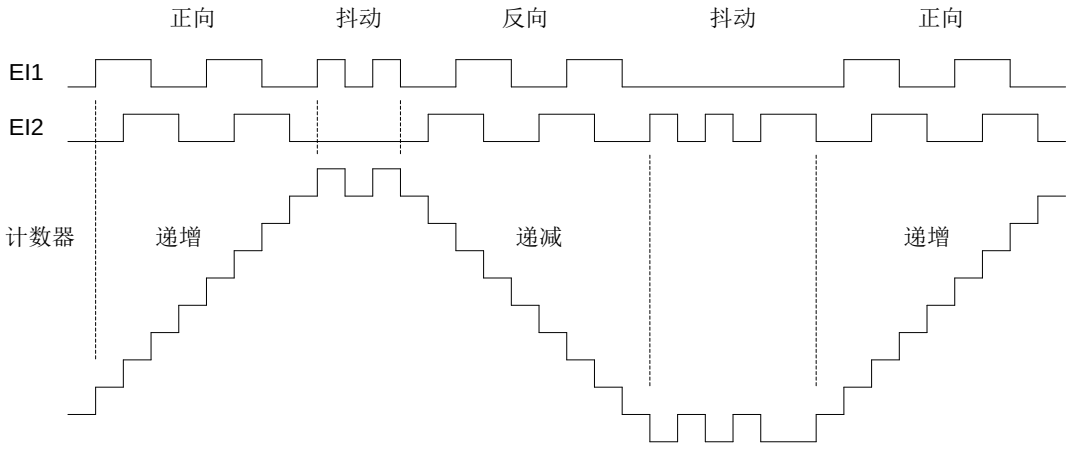
使能编码器接口模式的方法是：如果计数器只在 EI1 的边沿计数，则设置 ECP1_EM[1:0]=01；如果计数器只在 EI2 的边沿计数，则设置 ECP1_EM[1:0]=10；如果计数器在 EI1 和 EI2 的边沿计数，则设置 ECP1_EM[1:0]=11。禁止编码器接口模式的方法是设置 ECP1_EM[1:0]=00。

当计数器已经使能且编码器接口模式也被使能时，计数器在每次 EI1 或 EI2 上产生有效跳变时计数。根据两个输入信号的跳变顺序，产生计数脉冲和方向信号，计数器向上或向下计数。计数方向与编码器信号的关系如下表所示，假设 EI1 和 EI2 不会同时变换：

计数方向与编码器信号的关系

ECP1_EM	相对信号的电平 （EI1 的相对信号为 EI2， EI2 的相对信号为 EI1）	EI1 信号		EI2 信号	
		上升沿	下降沿	上升沿	下降沿
00	-	不计数	不计数	不计数	不计数
01	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
10	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
11	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择双边沿模式时，输入抖动是如何被抑制的，抖动可能会在传感器的位置靠近一个转换点时产生。



寄存器定义

ECP1CR0 – ECP1 控制寄存器 0

ECP1CR0 – ECP1 控制寄存器 0								
地址：0x1A0					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP1_DCLM	-	ECP1_PE3	ECP1_CE3	ECP1_PE2	ECP1_CE2	ECP1_PE1	ECP1_CE1
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	ECP1_DCLM	占空比更新模式选择控制位 1：定时器 1 溢出时 ECP1RnH:ECP1RnL 寄存器值更新到占空比缓存器 0：写入 ECP1RnH:ECP1RnL 寄存器中的值直接更新到占空比缓存器 三个通道的占空比更新模式均由此位来选择。						
6	-	保留						
5	ECP1_PE3	通道 3 的 PWM 模式使能控制位 1：通道 3 的 PWM 模式被使能 0：通道 3 的 PWM 模式被禁止						
4	ECP1_CE3	通道 3 的捕获模式使能控制位 1：通道 3 的捕获模式被使能 0：通道 3 的捕获模式被禁止						
3	ECP1_PE2	通道 2 的 PWM 模式使能控制位 1：通道 2 的 PWM 模式被使能 0：通道 2 的 PWM 模式被禁止						
2	ECP1_CE2	通道 2 的捕获模式使能控制位 1：通道 2 的捕获模式被使能 0：通道 2 的捕获模式被禁止						
1	ECP1_PE1	通道 1 的 PWM 模式使能控制位 1：通道 1 的 PWM 模式被使能 0：通道 1 的 PWM 模式被禁止						
0	ECP1_CE1	通道 1 的捕获模式使能控制位 1：通道 1 的捕获模式被使能 0：通道 1 的捕获模式被禁止						

ECP1CR1 – ECP1 控制寄存器 1

ECP1CR1 – ECP1 控制寄存器 1								
地址：0x1A1					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP1_EM1	ECP1_EM0	ECP1_CM11	ECP1_CM10	ECP1_SS3	ECP1_SS2	ECP1_SS11	ECP1_SS10
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:6	ECP1_EM	编码器模式选择控制位 11：在 EI1 和 EI2 的边沿计数 10：只在 EI2 的边沿计数 01：只在 EI1 的边沿计数						

		00：不计数，编码器接口模式关闭
5:4	ECP1_CM1	通道 1 的捕获模式选择控制位 11：通道 1 捕获源的第 4 个上升沿 10：通道 1 捕获源的第 3 个上升沿 01：通道 1 捕获源的第 2 个上升沿 00：通道 1 捕获源的任一上升沿
3	ECP1_SS3	通道 3 的捕获源选择控制位 1：编码器的方向标志位 0：外部引脚 T1C3 的输入
2	ECP1_SS2	通道 2 的捕获源选择控制位 1：定时器 0 的溢出标志位 T0IF 0：外部引脚 T1C2 的输入
1:0	ECP1_SS1	通道 1 的捕获源选择控制位 11：外部引脚 T1C3，T1C2，T1C1 输入的异或 10：外部引脚 T1C3 的输入 01：外部引脚 T1C2 的输入 00：外部引脚 T1C1 的输入

ECP1CR2 – ECP1 控制寄存器 2

ECP1CR2 – ECP1 控制寄存器 2								
地址：0x1A2					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP1_ENCDIR	-	ECP1_PH3	ECP1_OE3	ECP1_PH2	ECP1_OE2	ECP1_PH1	ECP1_OE1
R/W	W/R	-	W/R	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述
7	ECP1_ENCDIR	编码器方向标志位 1：正向，计数器进行递增操作 0：反向，计数器进行递减操作
6	-	保留
5	ECP1_PH3	通道 3 的极性选择控制位 1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变
4	ECP1_OE3	通道 3 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止
3	ECP1_PH2	通道 2 的极性选择控制位 1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变
2	ECP1_OE2	通道 2 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止
1	ECP1_PH1	通道 1 的极性选择控制位 1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变

0	ECP1_OE1	通道 1 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止

ECP1PR0 – ECP1 引脚配置寄存器 0

ECP1PR0 – ECP1 引脚配置寄存器 0								
地址：0x1A6					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	T1C3_RB2E	T1C2_RA6E	T1C1_RB4E	T1C1_RA4E
R/W	-	-	-	-	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:4	-	保留						
3	T1C3_RB2E	T1C3@引脚 RB2 使能控制位 1：T1C3 分配到引脚 RB2 上，捕获模式下，T1C3 从引脚 RB2 上输入；PWM 模式下，T1C3 从 RB2 上输出 0：T1C3 不分配到引脚 RB2 上						
2	T1C3_RA6E	T1C3@引脚 RA6 使能控制位 1：T1C3 分配到引脚 RA6 上，捕获模式下，T1C3 从引脚 RA6 上输入；PWM 模式下，T1C3 从 RA6 上输出 0：T1C3 不分配到引脚 RA6 上						
1	T1C1_RB4E	T1C1@引脚 RB4 使能控制位 1：T1C1 分配到引脚 RB4 上，捕获模式下，T1C1 从引脚 RB4 上输入；PWM 模式下，T1C1 从 RB4 上输出 0：T1C1 不分配到引脚 RB4 上						
0	T1C1_RA4E	T1C1@引脚 RA4 使能控制位 1：T1C1 分配到引脚 RA4 上，捕获模式下，T1C1 从引脚 RA4 上输入；PWM 模式下，T1C1 从 RA4 上输出 0：T1C1 不分配到引脚 RA4 上						

ECP1IR0 – ECP1 中断寄存器 0

ECP1IR0 – ECP1 中断寄存器 0								
地址：0x1A7					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	-	ECP1_IE3	ECP1_IE2	ECP1_IE1	-	ECP1_IF3	ECP1_IF2	ECP1_IF1
R/W	-	W/R	W/R	W/R	-	W/R	W/R	W/R
Bit	Name	描述						
7	-	保留						
6	ECP1_IE3	通道 3 中断使能控制位 1：通道 3 中断被使能，设置 ECP1_IE3 位为 1 后，当 ECP1_IF3 位被置位时，将会置位 ECP1 模块中断标志位 ECP1IF 0：通道 3 中断被禁止						
5	ECP1_IE2	通道 2 中断使能控制位						

		1：通道 2 中断被使能，设置 ECP1_IE2 位为 1 后，当 ECP1_IF2 位被置位时，将会置位 ECP1 模块中断标志位 ECP1IF 0：通道 2 中断被禁止
4	ECP1_IE1	通道 1 中断使能控制位 1：通道 1 中断被使能，设置 ECP1_IE1 位为 1 后，当 ECP1_IF1 位被置位时，将会置位 ECP1 模块中断标志位 ECP1IF 0：通道 1 中断被禁止
3	-	保留
2	ECP1_IF3	通道 3 中断标志位 1：捕获模式下，通道 3 发生捕获事件会置位 ECP1_IF3 位；PWM 模式下，通道 3 占空比与计数值发生匹配时会置位 ECP1_IF3 位 0：通道 3 中断标志位未被置位，对 ECP1_IF3 位写 0 将会清零该位，对 ECP1 模块中断标志位 ECP1IF 清零时也会清零 ECP1_IF3 位
1	ECP1_IF2	通道 2 中断标志位 1：捕获模式下，通道 2 发生捕获事件会置位 ECP1_IF2 位；PWM 模式下，通道 2 占空比与计数值发生匹配时会置位 ECP1_IF2 位 0：通道 2 中断标志位未被置位，对 ECP1_IF2 位写 0 将会清零该位，对 ECP1 模块中断标志位 ECP1IF 清零时也会清零 ECP1_IF2 位
0	ECP1_IF1	通道 1 中断标志位 1：捕获模式下，通道 1 发生捕获事件会置位 ECP1_IF1 位；PWM 模式下，通道 1 占空比与计数值发生匹配时会置位 ECP1_IF1 位 0：通道 1 中断标志位未被置位，对 ECP1_IF1 位写 0 将会清零该位，对 ECP1 模块中断标志位 ECP1IF 清零时也会清零 ECP1_IF1 位

ECP1R1L – ECP1 通道 1 捕获/占空比寄存器低字节

ECP1R1L – ECP1 通道 1 捕获/占空比寄存器低字节								
地址：0x1A8					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP1R1L7	ECP1R1L6	ECP1R1L5	ECP1R1L4	ECP1R1L3	ECP1R1L2	ECP1R1L1	ECP1R1L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP1R1L	ECP1 通道 1 捕获/占空比寄存器低字节 捕获模式下，ECP1R1L 是通道 1 的 16 位捕获寄存器的低 8 位。发生捕获事件后，硬件把 16 位定时器 1 的计数值保存到 ECP1R1H:ECP1R1L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP1R1L 是通道 1 的 16 位占空比寄存器的低 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP1R1H:ECP1R1L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP1R1H – ECP1 通道 1 捕获/占空比寄存器高字节

ECP1R1H – ECP1 通道 1 捕获/占空比寄存器高字节								
地址：0x1A9					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0

Name	ECP1R1H7	ECP1R1H6	ECP1R1H5	ECP1R1H4	ECP1R1H3	ECP1R1H2	ECP1R1H1	ECP1R1H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP1R1H	ECP1 通道 1 捕获/占空比寄存器高字节 捕获模式下，ECP1R1H 是通道 1 的 16 位捕获寄存器的高 8 位。发生捕获事件后，硬件把 16 位定时器 1 的计数值保存到 ECP1R1H:ECP1R1L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP1R1H 是通道 1 的 16 位占空比寄存器的高 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP1R1H:ECP1R1L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP1R2L – ECP1 通道 2 捕获/占空比寄存器低字节

ECP1R1L – ECP1 通道 2 捕获/占空比寄存器低字节								
地址：0x1AA					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP1R2L7	ECP1R2L6	ECP1R2L5	ECP1R2L4	ECP1R2L3	ECP1R2L2	ECP1R2L1	ECP1R2L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP1R2L	ECP1 通道 2 捕获/占空比寄存器低字节 捕获模式下，ECP1R2L 是通道 2 的 16 位捕获寄存器的低 8 位。发生捕获事件后，硬件把 16 位定时器 1 的计数值保存到 ECP1R2H:ECP1R2L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP1R2L 是通道 2 的 16 位占空比寄存器的低 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP1R2H:ECP1R2L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP1R2H – ECP1 通道 2 捕获/占空比寄存器高字节

ECP1R2H – ECP1 通道 2 捕获/占空比寄存器高字节								
地址：0x1AB					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP1R2H7	ECP1R2H6	ECP1R2H5	ECP1R2H4	ECP1R2H3	ECP1R2H2	ECP1R2H1	ECP1R2H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP1R2H	ECP1 通道 2 捕获/占空比寄存器高字节 捕获模式下，ECP1R2H 是通道 2 的 16 位捕获寄存器的高 8 位。发生捕获事件后，硬件把 16 位定时器 1 的计数值保存到 ECP1R2H:ECP1R2L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP1R2H 是通道 2 的 16 位占空比寄存器的高 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP1R2H:ECP1R2L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP1R3L – ECP1 通道 3 捕获/占空比寄存器低字节

ECP1R3L – ECP1 通道 3 捕获/占空比寄存器低字节								
地址：0x1AC					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP1R3L7	ECP1R3L6	ECP1R3L5	ECP1R3L4	ECP1R3L3	ECP1R3L2	ECP1R3L1	ECP1R3L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP1R3L	ECP1 通道 3 捕获/占空比寄存器低字节 捕获模式下，ECP1R3L 是通道 3 的 16 位捕获寄存器的低 8 位。发生捕获事件后，硬件把 16 位定时器 1 的计数值保存到 ECP1R3H:ECP1R3L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP1R3L 是通道 3 的 16 位占空比寄存器的低 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP1R3H:ECP1R3L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP1R3H – ECP1 通道 3 捕获/占空比寄存器高字节

ECP1R3H – ECP1 通道 3 捕获/占空比寄存器高字节								
地址：0x1AD					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP1R3H7	ECP1R3H6	ECP1R3H5	ECP1R3H4	ECP1R3H3	ECP1R3H2	ECP1R3H1	ECP1R3H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP1R3H	ECP1 通道 3 捕获/占空比寄存器高字节 捕获模式下，ECP1R3H 是通道 3 的 16 位捕获寄存器的高 8 位。发生捕获事件后，硬件把 16 位定时器 1 的计数值保存到 ECP1R3H:ECP1R3L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP1R3H 是通道 3 的 16 位占空比寄存器的高 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP1R3H:ECP1R3L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

PIR1 – 外设中断标记寄存器

PIR1 – 外设中断标记寄存器								
地址：0x0C					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2IF	ADIF	RXIF	TXIF	SPIF	ECP1IF	T2IF	T1IF
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
2	ECP1IF	ECP1 中断标记位 发生捕获、比较匹配或自动关闭时，会置位 ECP1IF。如果 ECP1 中断被使能时，会产生 ECP1 中断。软件须写“0”来清零 ECP1IF 位。						

PIE1 – 外设中断使能寄存器

PIR1 – 外设中断使能寄存器								
地址：0x8C					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2IE	ADIE	RXIE	TXIE	SPIE	ECP1IE	T2IE	T1IE
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
2	ECP1IE	ECP1 中断使能位 1：ECP1 中断被使能 0：ECP1 中断被禁止						

TRISA – PORTA 三态控制寄存器

TRISA – PORTA 三态控制寄存器								
地址：0x85					默认值：1111_1111			
Bit	7	6	5	4	3	2	1	0
Name	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
6	TRISA6	RA6 三态控制 1：RA6 引脚被配置为输入，捕获源 T1C3 可由此输入 0：RA6 引脚被配置为输出，PWM 输出信号 T1C3 可由此输出						
4	TRISA4	RA4 三态控制 1：RA4 引脚被配置为输入，捕获源 T1C1 可由此输入 0：RA4 引脚被配置为输出，PWM 输出信号 T1C1 可由此输出						

TRISB – PORTB 三态控制寄存器

TRISB – PORTB 三态控制寄存器								
地址：0x86					默认值：1111_1111			
Bit	7	6	5	4	3	2	1	0
Name	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
4	TRISB4	RB4 三态控制 1：RB4 引脚被配置为输入，捕获源 T1C1 可由此输入 0：RB4 引脚被配置为输出，PWM 输出信号 T1C1 可由此输出						
3	TRISB3	RB3 三态控制 1：RB3 引脚被配置为输入，捕获源 T1C2 可由此输入 0：RB3 引脚被配置为输出，PWM 输出信号 T1C2 可由此输出						
2	TRISB2	RB2 三态控制 1：RB2 引脚被配置为输入，捕获源 T1C3 可由此输入 0：RB2 引脚被配置为输出，PWM 输出信号 T1C3 可由此输出						

增强型比较/俘获/PWM 模块(ECCP2)

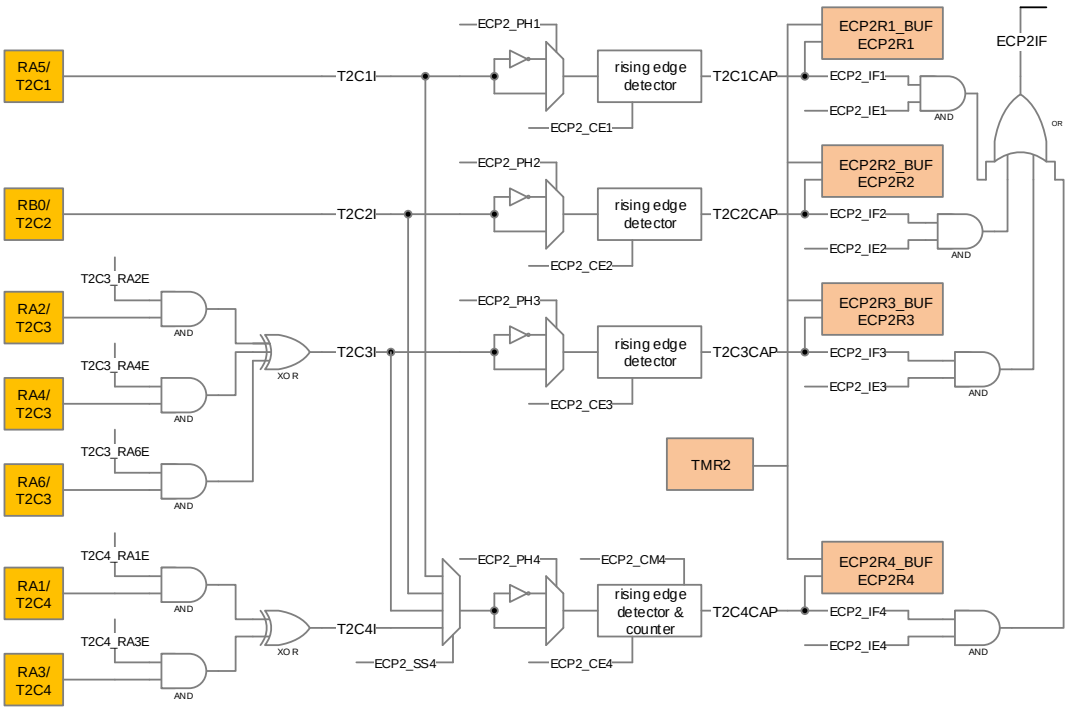
- 支持 Capture 模式，捕捉外部事件，支持 4 路输入通道
- 支持 PWM 模式，产生 4 路频率和占空比可控的 PWM 信号，3 组带死区控制的互补 PWM 信号
- 支持高速时钟模式，最高计数时钟达 32MHz

综述

增强型捕获及脉冲宽度调制模块（简称为 ECP 模块）是一个用来计时和控制各种事件的外设。Capture 模式下，可以对外部事件进行计时。PWM 模式下，可以产生频率和占空比可调的 PWM 信号。基于定时计数器 2 的 ECP 模块，被称为 ECP2 模块。本章节系统地描述 ECP2 的捕获，PWM 及编码器模式。本章使用小写字母 n 来代表 1、2、3、4，分别对应于 ECP2 的四个不同的通道。四个通道所对应的外部引脚为 T2Cn，捕获模式下为输入，PWM 模式下为输出。

Capture 模式

捕获模式的结构图如下所示：



Capture 使能配置

设置位于 ECP2CR0 寄存器的 ECP2_CEn 位为 1 时，会使能该通道的捕获功能，三个通道可同时使能。需注意的是，当该通道的捕获功能开启后，应避免开启其 PWM 功能。

Capture 时钟配置

ECP2 模块是基于定时器 2 来产生 PWM 的，因此，ECP2 模块的工作时钟与定时器 2 的工作时钟是一致

的。ECP2 模块时钟源有系统时钟和内部 RC 模块的 2 倍频时钟两种，由位于 TCCR 寄存器的 T2CS 位来选择。当设置 T2CS 位为 0 时，ECP2 模块时钟为系统时钟，最大为 16MHz；当设置 T2CS 位为 1 时，ECP2 模块时钟为内部 RC 模块的 2 倍频时钟输出，最大为 32MHz。须注意的是，若要选用内部模块的 2 倍频时钟输出作为 ECP2 模块的时钟源，必须先使能该 2 倍频时钟，即设置位于 TCCR 寄存器的 X2EN 位为 1。

Capture 输入配置

四个通道的捕获源均来自外部引脚 T2Cn，其中通道 4 可来自全部四个外部引脚，由位于 ECP2CR1 寄存器的 ECP2_SSn 位来选择，选择详情见寄存器描述。

当选用的捕获源来自于外部引脚 T2Cn 时，设置其相应的 TRIS 控制位来配置 T2Cn 引脚作为输入。对通道 3 和通道 4 来说，都有多个外部引脚，除了设置其相应的 TRIS 控制位外，还必须设置其相应的引脚选择配置位。以通道 4 为例，RA1 和 RA3 均为通道 4 的外部引脚 T2C4，若要选用 RA1 作为通道 4 的外部捕获源 T2C4I，须设置位于 ECP2PR0 寄存器的 T2C4_RA1E 位为 1，使能 RA1 作为通道 4 的外部输入；若要选用 RA3 作为通道 4 的外部捕获源 T2C4I，须设置位于 ECP2PR0 寄存器的 T2C4_RA3E 位为 1，使能 RA3 作为通道 1 的外部输入；若设置 T2C4_RA1E 和 T2C4_RA3E 位都为 1，RA1 和 RA3 都被选用为通道 4 的外部输入，此时，通道 4 的外部捕获源 T2C4I 为 RA1 和 RA3 的异或结果；若设置 T2C4_RA1E 和 T2C4_RA3E 位都为 0，RA1 和 RA3 都没有被选用为通道 4 的外部捕获源 T2C4I，此时，通道 4 的外部捕获源为 0，捕获不会发生。

被选用的捕获源在进行捕获之前，还可进行极性的配置，由位于 ECP2CR2 寄存器的 ECP2_PHn 来选择。

Capture 模式配置

Capture 模式下，对通道 1，2 和 3 来说，捕获事件为捕获源的任一上升沿，对通道,4 来说，该事件为以下四种当中的一种，由 ECP2CR1 寄存器中的 ECP2_CM4[1:0]位来选择，包括：

- 任一上升沿
- 第 2 个升沿
- 第 4 个上升沿
- 第 16 个上升沿

Capture 模式下，当所选捕获源的事件发生时，定时器 2 的 16 位计数值 TMR2 会被捕获到 16 位缓存器 ECP2Rn_BUF 中去，然后再更新到 16 位捕获寄存器 ECP2Rn 中去，软件通过读取 ECP2RnH:ECP2RnL 寄存器中的值获得捕获值。软件读取 ECP2RnH:ECP2RnL 寄存器时，须先读取高位寄存器 ECP2RnH，再读取低位寄存器 ECP2RnL。因为在软件读取完高位寄存器 ECP2RnH 后，硬件会关闭捕获更新，直到读取完低位寄存器 ECP2RnL 后，硬件才会允许捕获更新，缓存器中的值才会被更新到捕获寄存器中去。

当捕获产生后，位于 ECP2IR0 寄存器的 ECP2_IFn 位会被置位，同时，位于 PIR1 寄存器中的相应中断请求标志位 ECP2IF 也会被置位。该标志位都必须由软件来清零，其中，对 ECP2_IFn 位分别写 0，可分别清除 ECP2_IFn 位；对 ECP2IF 位写 0，会同时清除 ECP2IF 位和三个 ECP2_Ifn 位。在 ECP2RnH 和 ECP2RnL 寄存器的值被读走之前，如果发生新的捕获，原来的捕获值会被新的捕获值覆盖掉。

定时器 2 模式选择

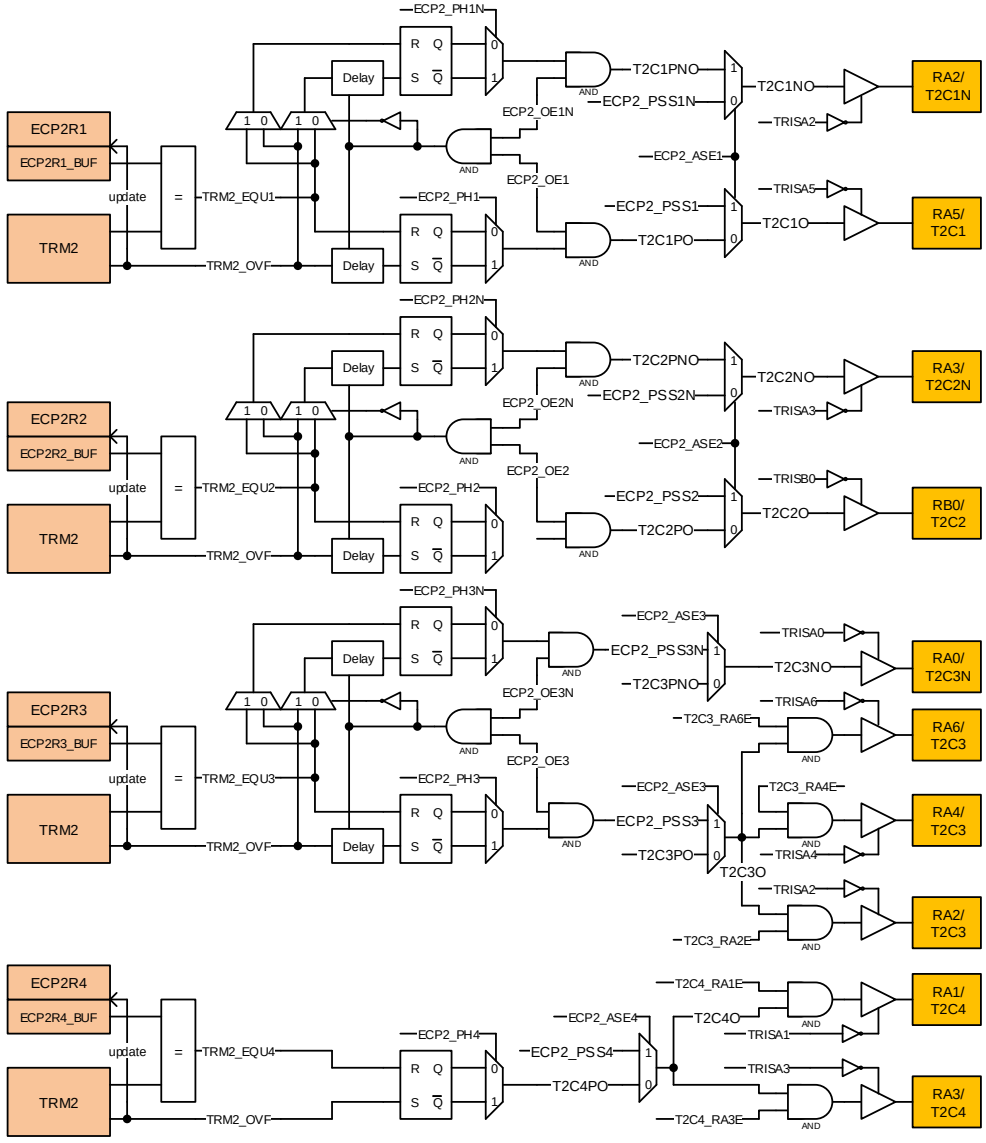
Capture 模式下，定时器 2 可工作在定时模式或计数模式。同步计数模式和异步计数模式下，捕获都可以正常工作。

软件中断

当 Capture 模式发生改变时，可能会产生一次假的捕获中断。用户需要保持位于 PIE1 寄存器的 ECP2IE 位为零来避免此假中断。此外，还需要在改变 Capture 模式后清零位于 PIR1 寄存器的 ECP2IF 中断标志位。

PWM 模式

PWM 模式的结构图如下所示：



PWM 使能配置

设置位于 ECP2CR0 寄存器的 ECP2_PEn 位为 1 时，会使能该通道的 PWM 功能，四个通道可同时使能。需注意的是，当该通道的 PWM 功能开启后，应避免开启其捕获功能。

PWM 输出配置

四个通道所产生的 PWM 信号用 T2PWMn 来表示，均可输出到外部引脚 T2Cn，由位于 ECP2CR2 和 ECP2CR3 寄存器的 ECP2_OEn 位和 ECP2_OEnN 位来控制，还须设置其相应的 TRIS 控制位来配置 T2Cn 引脚作为输

出。

对同一个通道而言，以通道 1 为例，当设置 ECP2_OE1 位和 ECP2_OE1N 位都为 1 时，在引脚 T2C1 和 T2C1N 上会输出二路互补的 PWM 信号。当设置 ECP2_OE1 位为 1 而 ECP2_OE1N 位为 0 时，只在引脚 T2C1 上输出单路 PWM 信号。当设置 ECP2_OE1 位为 0 而设置 ECP2_OE1N 位为 1 时，只在引脚 T2C1N 上输出单路 PWM 信号。当设置 ECP2_OE1 位和 ECP2_OE1N 位都为 0 时，不会输出 PWM 信号。

通道 1,2 和 3 上均可产生互补输出的 PWM 信号，通道 4 只能产生单路输出的 PWM 信号。

产生互补输出 PWM 信号时，会在 PWM 信号由无效驱动状态变为有效驱动状态前插入持续的死区时间，T2Cn 上所插入的死区时间是由 ECP2DTP 寄存器来控制，T2CnN 上所插入的死区时间由 ECP2DTN 寄存器来控制，死区时间的具体计算方法见寄存器描述。

四个通道所产生的 PWM 信号 T2PWMn 和 T2PWMnN 输出到外部引脚之前，还可进行极性的配置，由位于 ECP2CR2 寄存器的 ECP2_PHn 来选择。

PWM 时钟配置

ECP2 模块是基于定时器 2 来产生 PWM 的，因此，ECP2 模块的工作时钟与定时器 2 的工作时钟是一致的。ECP2 模块时钟源有系统时钟和内部 RC 模块的 2 倍频时钟两种，由位于 TCCR 寄存器的 T2CS 位来选择。当设置 T2CS 位为 0 时，ECP2 模块时钟为系统时钟，最大为 16MHz；当设置 T2CS 位为 1 时，ECP2 模块时钟为内部 RC 模块的 2 倍频时钟输出，最大为 32MHz。须注意的是，若要选用内部模块的 2 倍频时钟输出作为 ECP2 模块的时钟源，必须先使能该 2 倍频时钟，即设置位于 TCCR 寄存器的 X2EN 位为 1。

PWM 特性配置

PWM 信号 T2PWMn 的周期、占空比和分辨率由以下寄存器来决定：

- PR2H:PR2L
- ECP2RnH:ECP2RnL
- T2CON

其中，PR2H:PR2L 为 16 位定时器 2 的计数最大值，用 PR1 来表示。16 位的占空比寄存器 ECP2RnH:ECP2RnL 还有对应的 16 位占空比缓存器 ECP2RnH_BUF:ECP2RnL_BUF，PWM 信号的占空比最终是由缓存器来决定。更新占空比有两种方式，由位于 ECP2CR1 的 ECP2_DCLM 位来选择。当 ECP2_DCLM 位为 0 时，写入 ECP2RnH：ECP2RnL 寄存器中的内容会直接更新到缓存器中，占空比更新立即完成；当 ECP2_DCLM 位为 1 时，只有在定时器 2 发生溢出时，ECP2RnH：ECP2RnL 寄存器中的内容才会更新好缓冲器中，实现占空比配置的更新。

PWM 周期

PWM 周期由定时器 2 的 PR2 寄存器决定。PWM 周期由下面的公式来计算：

PWM Period = [(PR2) + 1] * Tosc * TMR2PR

其中，PR2 为计数最大值，Tosc 为定时器 2 的计数时钟，TRM2PR 为预分频系数。

当 TMR2 等于 PR2 时，下一个计数周期会发生以下三件事：

- TMR2 被清零
- T2PWMn 被置位（例外：若 PWM 占空比为 0%，PWM 信号将不会被置位）
- PWM 占空比值由 ECP2RnH:ECP2RnL 锁存到其相应的缓存器中（ECP2_DCLM=1 时）

PWM 占空比

PWM 占空比由 16 位寄存器 ECP2RnH:ECP2RnL 的值来决定。

PWM 脉冲宽度由下面的公式来计算：

Pulse Width = (ECP2RnH:ECP2RnL) * Tosc * (TMR2PR)

PWM 占空比由下面的公式来计算：

Duty Cycle Ratio = (ECP2RnH:ECP2RnL) / ((PR2+1))

当 TRM2 和 ECP2RnH:ECP2RnL 发生匹配时，T2PWMn 被清零。

PWM 分辨率

对于给定的周期，分辨率决定了可能的占空比周期数。16 位分辨率有 65536 个连续的占空比周期，10 位分辨率有 1024 个连续的占空比周期，8 位分辨率有 256 个占空比周期。

当 PR2 被设置为 65535 时，拥有最大的 PWM 分辨率，即 16 位。分辨率是 PR2 寄存器的函数，如下面公式所示：

Resolution = log[(PR2 + 1)] / log2 bits

下面两个表格为 PWM 频率和占空比的示例。

PWM 频率和分辨率（Fosc = 16 MHz）

Frequency (KHz)	0.031	0.061	0.244	15.625	62.5	250
TMR2 Prescale	8	4	1	1	1	1
PR2 Value	0xFFFF	0xFFFF	0xFFFF	0x3FF	0xFF	0x3F
Max Resulution	16	16	16	10	8	6

PWM 对齐方式

ECP2 模块产生的 PWM 有两种对齐方式，边沿对齐和中心对齐，由位于 ECP2CR3 寄存器的 ECP2_PM 位来选择。

当设置 ECP2_PM 位为低时，是边沿对齐模式，定时器 2 的计数方式为单向计数，计数的方向为递增或递减，由位于 TCR2 寄存器的 TC2DIR 位来选择。TC2DIR 位为高时，定时器 2 的计数模式为递增，即计数器从最小值 0 开始递增，累加到最大值 PR2 后又回到最小值 0 重新递增。TC2DIR 位为低时，定时器 2 的计数模式为递减，即计数器从最大值 PR2 开始递增，累减到最小值 0 后又回到最大值 PR2 重新递减。在最大值和最小值跳变的时候，产生一个 PWM 周期的一个边沿，故称为边沿对齐模式。在占空比值与计数值发生匹配时，产生一个 PWM 周期的另一个边沿。

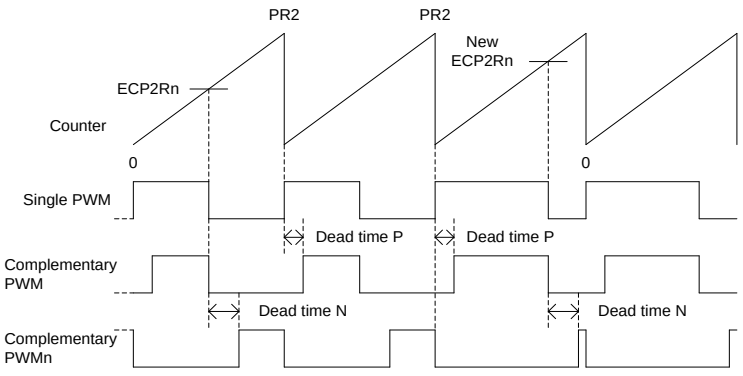


Figure 3 边沿对齐模式下的计数和 PWM 产生方式

当设置 ECP2_PM 位为高时，是中心对齐模式，定时器 2 的计数方式为双向计数，计数的方向为递增和递减交替，不受 TC2DIR 位的控制。计数器从最小值 0 开始递增，累加到最大值 PR2 后开始递减，累减到最小值 0 后又开始递增，如此反复。在计数上升的过程中，占空比值与计数值发生匹配时，产生一个 PWM 周期的一个边沿；在计数下降的过程中，占空比值与计数值发生匹配时，产生一个 PWM 周期的另一个边沿。此时，PWM 信号会已计数最大值为中心对齐，故称为中心对齐模式。

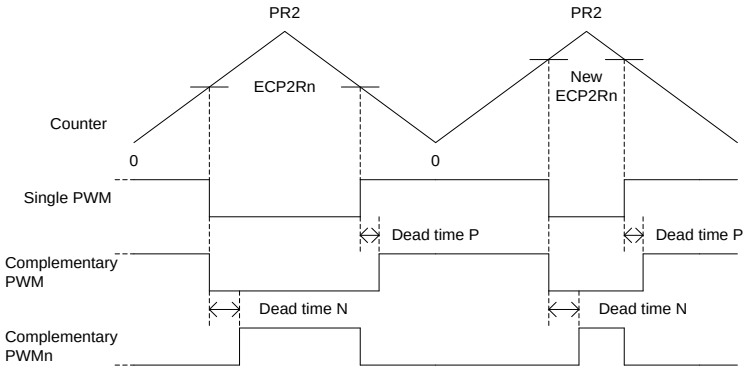


Figure 4 中心对齐模式下的计数和 PWM 产生方式

PWM 自动关闭方式

ECP2 模块产生的 PWM 可自动关闭，由位于 ECP2CR4 寄存器来控制。其中，ECP2_ASE 位为 4 个通道共有的控制信号，ECP2_ASDEn 位为 4 个通道自动关闭的使能信号。当 ECP2_ASE 和 ECP2_ASDEn 均为高时，该通道的 PWM 输出被关闭，相应引脚的输出状态为自动关闭时所设定的状态（由 ECP2CR5[6:0]寄存器来选择，见寄存器描述）。

其中，ECP2_ASE 位可由软件直接置位来控制 PWM 是否关闭，也可以由硬件来控制其置位，即满足一定触发条件时才关闭 PWM 输出，而切换为固定状态输出。置位 ECP2_ASE 位的硬件触发条件有三个，并有相应的寄存器位来使能，如下所示：

1. 定时器 0 的中断标志位 T0IF 为高，由 ECP2_ASSE0 位来使能；
2. 比较器的输出 COUT 为高，由 ECP2_ASSE1 位来使能；
3. 外部输入的刹车信号 T2BK 为高，由 ECP2_ASSE2 位来使能，且 T2BK 信号的极性可由 ECP2_BKP 位来选择；

当此三个条件中的任一条件满足时，ECP2_ASE 位会被置位，并保持住，此时，各个 PWM 输出引脚上切换为所预设的电平状态，并保持。

当此三个条件均不满足时，软件才可清零 ECP2_ASE 位。软件可通过对 ECP2_ASE 位直接写 0，或通过清零中断标志位 ECP2IF 位来清零 ECP2_ASE 位。

PWM 操作的配置

当配置 ECP2 模块工作在 PWM 模式时，可按照以下流程来执行：

8. 置位相应的 TRIS 位来禁止 PWM 引脚（T2Cn）的输出驱动；
9. 加载 PR2 寄存器来设置 PWM 周期；
10. 加载 ECP2RnH 寄存器和 ECP2RnL 寄存器来设置 PWM 占空比；
11. 加载 ECP2CR2 寄存器来设置 PWM 信号的极性和使能输出；

12. 加载 ECP2CR0 寄存器来设置 ECP2 模块的某个或多个通道为 PWM 模式以及占空比的更新方式；
13. 配置和启动定时器 2，包括：
    - 清零位于 PIR1 寄存器的 TMR2IF 中断标志位
    - 加载位于 T2CON 寄存器的 T2CKPS 位来设置定时器 2 的预分频系数
    - 置位位于 T2CON 寄存器的 TMR2ON 位来使能定时器 2
14. 一个新的 PWM 周期开始后，使能 PWM 输出，即：
    - 等待定时器 2 溢出（即位于 PIR1 寄存器的 TMR2IF 位置位）
    - 清零相应的 TRIS 位来使能 T2Cn 引脚的输出驱动

寄存器定义

ECP2CR0 – ECP2 控制寄存器 0

ECP2CR0 – ECP2 控制寄存器 0								
地址：0x1B0					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2_PE4	ECP2_CE4	ECP2_PE3	ECP2_CE3	ECP2_PE2	ECP2_CE2	ECP2_PE1	ECP2_CE1
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述
7	ECP2_PE4	通道 4 的 PWM 模式使能控制位 1：通道 4 的 PWM 模式被使能 0：通道 4 的 PWM 模式被禁止
6	ECP2_CE4	通道 4 的捕获模式使能控制位 1：通道 4 的捕获模式被使能 0：通道 4 的捕获模式被禁止
5	ECP2_PE3	通道 3 的 PWM 模式使能控制位 1：通道 3 的 PWM 模式被使能 0：通道 3 的 PWM 模式被禁止
4	ECP2_CE3	通道 3 的捕获模式使能控制位 1：通道 3 的捕获模式被使能 0：通道 3 的捕获模式被禁止
3	ECP2_PE2	通道 2 的 PWM 模式使能控制位 1：通道 2 的 PWM 模式被使能 0：通道 2 的 PWM 模式被禁止
2	ECP2_CE2	通道 2 的捕获模式使能控制位 1：通道 2 的捕获模式被使能 0：通道 2 的捕获模式被禁止
1	ECP2_PE1	通道 1 的 PWM 模式使能控制位 1：通道 1 的 PWM 模式被使能 0：通道 1 的 PWM 模式被禁止
0	ECP2_CE1	通道 1 的捕获模式使能控制位 1：通道 1 的捕获模式被使能 0：通道 1 的捕获模式被禁止

ECP2CR1 – ECP2 控制寄存器 1

ECP2CR1 – ECP2 控制寄存器 1								
地址：0x1B1					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2_CM41	ECP2_CM40	ECP2_SS41	ECP2_SS40	ECP2_PSUE	ECP2_PHUE	ECP2_PLEN	ECP2_DCLM
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:6	ECP2_CM4	通道 4 的捕获模式选择控制位 11：通道 4 捕获源的第 16 个上升沿 10：通道 4 捕获源的第 4 个上升沿 01：通道 4 捕获源的第 2 个上升沿 00：通道 4 捕获源的任一上升沿						
5:4	ECP2_SS4	通道 4 的捕获源选择控制位 11：外部引脚 T2C4 的输入 10：外部引脚 T2C3 的输入 01：外部引脚 T2C2 的输入 00：外部引脚 T2C1 的输入						
3	ECP2_PSUE	控制寄存器软件更新使能控制位 1：当 ECP2_PLEN 位为 1 时，软件写 ECP2_PSUE 位为 1，才更新 ECP2CR2 和 ECP2CR3[5:0]寄存器值更新到控制寄存器的缓存器中，硬件会自动清零 ECP2_PSUE 位 0：软件更新被禁止						
2	ECP2_PHUE	控制寄存器硬件更新使能控制位 1：当 ECP2_PLEN 位为 1 时，通道 4 中断标志位 ECP2_IF4 被置位时，才会更新 ECP2CR2 和 ECP2CR3[5:0]寄存器值更新到控制寄存器的缓存器中 0：硬件更新被禁止						
1	ECP2_PLEN	控制寄存器更新模式选择控制位 1：更新条件满足时，ECP2CR2 和 ECP2CR3[5:0]寄存器值更新到控制寄存器的缓存器后才会生效，更新条件由 ECP2_PHUE 和 ECP2_PSUE 来选择 0：写入 ECP2CR2 和 ECP2CR3[5:0]寄存器中的值直接更新到控制寄存器的缓存器中，并立即生效						
0	ECP2_DCLM	占空比更新模式选择控制位 1：定时器 2 溢出时 ECP2RnH:ECP2RnL 寄存器值更新到占空比缓存器 0：写入 ECP2RnH:ECP2RnL 寄存器中的值直接更新到占空比缓存器 该控制位对四个通道均有效						

ECP2CR2 – ECP2 控制寄存器 2

ECP2CR2 – ECP2 控制寄存器 2								
地址：0x1B2					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2_PH2N	ECP2_OE2N	ECP2_PH2	ECP2_OE2	ECP2_PH1N	ECP2_OE1N	ECP2_PH1	ECP2_OE1
R/W	W/R	-	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	ECP2_PH2N	通道 2N 的极性选择控制位						

		1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变
6	ECP2_OE2N	通道 2N 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止 当 ECP2_OE2 和 ECP2_OE2N 位同时设为 1 时，通道 2 和通道 2N 输出的 PWM 位互补输出。
5	ECP2_PH2	通道 2 的极性选择控制位 1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变
4	ECP2_OE2	通道 2 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止 当 ECP2_OE2 和 ECP2_OE2N 位同时设为 1 时，通道 2 和通道 2N 输出的 PWM 位互补输出。
3	ECP2_PH1N	通道 1N 的极性选择控制位 1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变
2	ECP2_OE1N	通道 1N 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止 当 ECP2_OE1 和 ECP2_OE1N 位同时设为 1 时，通道 1 和通道 1N 输出的 PWM 位互补输出。
1	ECP2_PH1	通道 1 的极性选择控制位 1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变
0	ECP2_OE1	通道 1 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止 当 ECP2_OE1 和 ECP2_OE1N 位同时设为 1 时，通道 1 和通道 1N 输出的 PWM 位互补输出。

ECP2CR3 – ECP2 控制寄存器 3

ECP2CR3 – ECP2 控制寄存器 3								
地址：0x1B3					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2_PM	-	ECP2_PH4	ECP2_OE4	ECP2_PH3N	ECP2_OE3N	ECP2_PH3	ECP2_OE3
R/W	W/R	-	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	ECP2_PM	PWM 对齐方式选择控制位 1：PWM 对齐方式为中心对齐 0：PWM 对齐方式为边沿对齐 该控制位对四个通道均有效						
6	-	保留						

5	ECP2_PH4	通道 4 的极性选择控制位 1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变
4	ECP2_OE4	通道 4 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止
3	ECP2_PH3N	通道 3N 的极性选择控制位 1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变
2	ECP2_OE3N	通道 3N 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止 当 ECP2_OE3 和 ECP2_OE3N 位同时设为 1 时，通道 3 和通道 3N 输出的 PWM 位互补输出。
1	ECP2_PH3	通道 3 的极性选择控制位 1：捕获源极性进行反向/PWM 输出极性进行反向 0：捕获源极性保持不变/PWM 输出极性保持不变
0	ECP2_OE3	通道 3 的 PWM 信号输出使能控制位 1：输出使能 0：输出禁止 当 ECP2_OE3 和 ECP2_OE3N 位同时设为 1 时，通道 3 和通道 3N 输出的 PWM 位互补输出。

ECP2CR4 – ECP2 控制寄存器 4

ECP2CR3 – ECP2 控制寄存器 4								
地址：0x1B4					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2_ASE	ECP2_ASSE2	ECP2_ASSE1	ECP2_ASSE0	ECP2_ASDE4	ECP2_ASDE3	ECP2_ASDE2	ECP2_ASDE1
R/W	W/R	-	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	ECP2_ASE	PWM 自动关闭控制位 1：PWM 自动关闭被使能 0：PWM 自动关闭被禁止 该控制位对四个通道均有效，软件可直接对该位进行写 1 置位/写 0 清零操作，软件清零 ECP2IF 时也可清零该位；硬件只可置位该位，满足触发事件即可置位，且置位优先级比软件清零优先级高，触发事件有三个触发源，由 ECP2_ASSE[2:0] 来选择控制						
6	ECP2_ASSE2	PWM 自动关闭触发源 2 使能控制位 1：刹车控制信号 T2BK 被选为 PWM 自动关闭触发源之一，当 T2BK 经过极性控制后的信号 T2BKI 为高时，会置位 ECP2_ASE 位来关闭 PWM 输出 0：T2BK 未被选作 PWM 自动关闭触发源						
5	ECP2_ASSE1	PWM 自动关闭触发源 1 使能控制位 1：比较器输出状态 COUT 被选为 PWM 自动关闭触发源之一，当 COUT 为高时，会置位 ECP2_ASE 位来关闭 PWM 输出						

		0：COUT 未被选作 PWM 自动关闭触发源
4	ECP2_ASSE0	PWM 自动关闭触发源 0 使能控制位 1：定时器 0 中断标志位 T0IF 被选为 PWM 自动关闭触发源之一，当 T0IF 为高时，会置位 ECP2_ASE 位来关闭 PWM 输出 0：T0IF 未被选作 PWM 自动关闭触发源
3	ECP2_ASDE4	通道 4 的自动关闭使能控制位 1：通道 4 的 PWM 自动关闭被使能，当 ECP2_ASE 被置位时，通道 4 的 PWM 输出被关闭 0：通道 4 的 PWM 自动关闭被禁止
2	ECP2_ASDE3	通道 3/3N 的自动关闭使能控制位 1：通道 3/3N 的 PWM 自动关闭被使能，当 ECP2_ASE 被置位时，通道 3/3N 的 PWM 输出被关闭 0：通道 3/3N 的 PWM 自动关闭被禁止
1	ECP2_ASDE2	通道 2/2N 的自动关闭使能控制位 1：通道 2/2N 的 PWM 自动关闭被使能，当 ECP2_ASE 被置位时，通道 2/2N 的 PWM 输出被关闭 0：通道 2/2N 的 PWM 自动关闭被禁止
0	ECP2_ASDE1	通道 1/1N 的自动关闭使能控制位 1：通道 1/1N 的 PWM 自动关闭被使能，当 ECP2_ASE 被置位时，通道 1/1N 的 PWM 输出被关闭 0：通道 1/1N 的 PWM 自动关闭被禁止

ECP2CR5 – ECP2 控制寄存器 5

ECP2CR3 – ECP2 控制寄存器 5								
地址：0x1B5					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2_BKP	ECP2_PSS3N	ECP2_PSS2N	ECP2_PSS1N	ECP2_PSS4	ECP2_PSS3	ECP2_PSS2	ECP2_PSS1
R/W	W/R	-	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	ECP2_BKP	刹车信号 T2BK 极性选择控制位 1：刹车信号 T2BK 极性进行反向 0：刹车信号 T2BK 极性保持不变						
6	ECP2_PSS3N	通道 3N 在自动关闭下输出状态选择控制位 1：自动关闭时，通道 3N 输出保持高电平状态 0：自动关闭时，通道 3N 输出保持低电平状态						
5	ECP2_PSS2N	通道 2N 在自动关闭下输出状态选择控制位 1：自动关闭时，通道 2N 输出保持高电平状态 0：自动关闭时，通道 2N 输出保持低电平状态						
4	ECP2_PSS1N	通道 1N 在自动关闭下输出状态选择控制位 1：自动关闭时，通道 1N 输出保持高电平状态 0：自动关闭时，通道 1N 输出保持低电平状态						
3	ECP2_PSS4	通道 4 在自动关闭下输出状态选择控制位 1：自动关闭时，通道 4 输出保持高电平状态 0：自动关闭时，通道 4 输出保持低电平状态						

2	ECP2_PSS3	通道 3 在自动关闭下输出状态选择控制位 1：自动关闭时，通道 3 输出保持高电平状态 0：自动关闭时，通道 3 输出保持低电平状态
1	ECP2_PSS2	通道 2 在自动关闭下输出状态选择控制位 1：自动关闭时，通道 2 输出保持高电平状态 0：自动关闭时，通道 2 输出保持低电平状态
0	ECP2_PSS1	通道 1 在自动关闭下输出状态选择控制位 1：自动关闭时，通道 1 输出保持高电平状态 0：自动关闭时，通道 1 输出保持低电平状态

ECP2DTP – ECP2 互补 P 通道死区时间寄存器

ECP2DTP – ECP2 互补 P 通道死区时间寄存器								
地址：0x1B6					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2DTP7	ECP2 DTP 6	ECP2DTP 5	ECP2DTP 4	ECP2DTP 3	ECP2DTP 2	ECP2DTP 1	ECP2DTP 0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述	
7:0	ECP2DTP	ECP2 互补 P 通道死区时间寄存器 PWM 模式下，当 ECP2OEn 和 ECP2OEnN 都为高时，T2Cn 和 T2CnN 所输出的 PWM 为互补输出，ECP2DTP 定义了 T2Cn 通道所插入互补输出之间的死区持续时间，如下表所示，Tt2clk 表示定时器 2 的计数时钟周期	
		ECP2DTP[7:5]	T2Cn 通道插入的死区时间
		0xx	ECP2DTP[7:0] * Tt2clk
		10x	(ECP2DTP[5:0] + 64) * Tt2clk
		110	(ECP2DTP[4:0] + 32) * 8 * Tt2clk
		111	(ECP2DTP[4:0] + 32) * 16 * Tt2clk

ECP2DTN – ECP2 互补 N 通道死区时间寄存器

ECP2DTN – ECP2 互补 N 通道死区时间寄存器								
地址：0x1B7					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2DTN7	ECP2 DTN 6	ECP2 DTN 5	ECP2DTN 4	ECP2DTN 3	ECP2DTN2	ECP2DTN1	ECP2DTN0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述	
7:0	ECP2DTN	ECP2 互补 N 通道死区时间寄存器 PWM 模式下，当 ECP2OEn 和 ECP2OEnN 都为高时，T2Cn 和 T2CnN 所输出的 PWM 为互补输出，ECP2DTN 定义了 T2CnN 通道所插入互补输出之间的死区持续时间，如下表所示，Tt2clk 表示定时器 2 的计数时钟周期	
		ECP2DTN[7:5]	T2CnN 通道插入的死区时间
		0xx	ECP2DTN[7:0] * Tt2clk
		10x	(ECP2DTN[5:0] + 64) * Tt2clk

		110	(ECP2DTN[4:0] + 32) * 8 * Tt2clk
		111	(ECP2DTN[4:0] + 32) * 16 * Tt2clk

ECP2PR0 – ECP2 引脚配置寄存器 0

ECP2PR0 – ECP2 引脚配置寄存器 0								
地址：0x1AE					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	T2BK_RA4E	T2BK_RA0E	T2C4_RA3E	T2C4_RA1E	T2C3_RA6E	T2C3_RA4E	T2C3_RA2E	-
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	-

Bit	Name	描述
7	T2BK_RA4E	T2BK@引脚 RA4 使能控制位 1：T2BK 分配到引脚 RA4 上，PWM 模式下，刹车控制信号 T2BK 从引脚 RA4 上输入； 0：T2BK 不分配到引脚 RA4 上
6	T2BK_RA0E	T2BK@引脚 RA0 使能控制位 1：T2BK 分配到引脚 RA0 上，PWM 模式下，刹车控制信号 T2BK 从引脚 RA0 上输入； 0：T2BK 不分配到引脚 RA0 上
5	T2C4_RA3E	T2C4@引脚 RA3 使能控制位 1：T2C4 分配到引脚 RA3 上，捕获模式下，T2C4 从引脚 RA3 上输入；PWM 模式下，T2C4 从 RA3 上输出 0：T2C4 不分配到引脚 RA3 上
4	T2C4_RA1E	T2C4@引脚 RA1 使能控制位 1：T2C4 分配到引脚 RA1 上，捕获模式下，T2C4 从引脚 RA1 上输入；PWM 模式下，T2C4 从 RA1 上输出 0：T2C4 不分配到引脚 RA1 上
3	T2C3_RA6E	T2C3@引脚 RA6 使能控制位 1：T2C3 分配到引脚 RA6 上，捕获模式下，T2C3 从引脚 RA6 上输入；PWM 模式下，T2C3 从 RA6 上输出 0：T2C3 不分配到引脚 RA6 上
2	T2C3_RA4E	T2C3@引脚 RA4 使能控制位 1：T2C3 分配到引脚 RA4 上，捕获模式下，T2C3 从引脚 RA4 上输入；PWM 模式下，T2C3 从 RA4 上输出 0：T2C3 不分配到引脚 RA4 上
1	T2C3_RA2E	T2C3@引脚 RA2 使能控制位 1：T2C3 分配到引脚 RA2 上，捕获模式下，T2C3 从引脚 RA2 上输入；PWM 模式下，T2C3 从 RA2 上输出 0：T2C3 不分配到引脚 RA2 上
0	-	保留

ECP2IR0 – ECP2 中断寄存器 0

ECP2IR0 – ECP2 中断寄存器 0								
地址：0x1AF					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2_IE4	ECP2_IE3	ECP2_IE2	ECP2_IE1	ECP2_IF4	ECP2_IF3	ECP2_IF2	ECP2_IF1
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	ECP2_IE4	通道 4 中断使能控制位 1：通道 4 中断被使能，设置 ECP2_IE4 位为 1 后，当 ECP2_IF4 位被置位时，将会置位 ECP2 模块中断标志位 ECP2IF 0：通道 4 中断被禁止						
6	ECP2_IE3	通道 3 中断使能控制位 1：通道 3 中断被使能，设置 ECP2_IE3 位为 1 后，当 ECP2_IF3 位被置位时，将会置位 ECP2 模块中断标志位 ECP2IF 0：通道 3 中断被禁止						
5	ECP2_IE2	通道 2 中断使能控制位 1：通道 2 中断被使能，设置 ECP2_IE2 位为 1 后，当 ECP2_IF2 位被置位时，将会置位 ECP2 模块中断标志位 ECP2IF 0：通道 2 中断被禁止						
4	ECP2_IE1	通道 1 中断使能控制位 1：通道 1 中断被使能，设置 ECP2_IE1 位为 1 后，当 ECP2_IF1 位被置位时，将会置位 ECP2 模块中断标志位 ECP2IF 0：通道 1 中断被禁止						
3	ECP2_IF4	通道 4 中断标志位 1：捕获模式下，通道 4 发生捕获事件会置位 ECP2_IF4 位；PWM 模式下，通道 4 占空比与计数值发生匹配时会置位 ECP2_IF4 位 0：通道 4 中断标志位未被置位，对 ECP2_IF4 位写 0 将会清零该位，对 ECP2 模块中断标志位 ECP2IF 清零时也会清零 ECP2_IF4 位						
2	ECP2_IF3	通道 3 中断标志位 1：捕获模式下，通道 3 发生捕获事件会置位 ECP2_IF3 位；PWM 模式下，通道 3 占空比与计数值发生匹配时会置位 ECP2_IF3 位 0：通道 3 中断标志位未被置位，对 ECP2_IF3 位写 0 将会清零该位，对 ECP2 模块中断标志位 ECP2IF 清零时也会清零 ECP2_IF3 位						
1	ECP2_IF2	通道 2 中断标志位 1：捕获模式下，通道 2 发生捕获事件会置位 ECP2_IF2 位；PWM 模式下，通道 2 占空比与计数值发生匹配时会置位 ECP2_IF2 位 0：通道 2 中断标志位未被置位，对 ECP2_IF2 位写 0 将会清零该位，对 ECP2 模块中断标志位 ECP2IF 清零时也会清零 ECP2_IF2 位						
0	ECP2_IF1	通道 1 中断标志位 1：捕获模式下，通道 1 发生捕获事件会置位 ECP2_IF1 位；PWM 模式下，通道 1 占空比与计数值发生匹配时会置位 ECP2_IF1 位 0：通道 1 中断标志位未被置位，对 ECP2_IF1 位写 0 将会清零该位，对 ECP2 模块中断标志位 ECP2IF 清零时也会清零 ECP2_IF1 位						

ECP2R1L – ECP2 通道 1 捕获/占空比寄存器低字节

ECP2R1L – ECP2 通道 1 捕获/占空比寄存器低字节								
地址：0x1B8					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2R1L7	ECP2R1L6	ECP2R1L5	ECP2R1L4	ECP2R1L3	ECP2R1L2	ECP2R1L1	ECP2R1L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP2R1L	ECP2 通道 1 捕获/占空比寄存器低字节 捕获模式下，ECP2R1L 是通道 1 的 16 位捕获寄存器的低 8 位。发生捕获事件后，硬件把 16 位定时器 2 的计数值保存到 ECP2R1H:ECP2R1L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP2R1L 是通道 1 的 16 位占空比寄存器的低 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP2R1H:ECP2R1L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP2R1H – ECP2 通道 1 捕获/占空比寄存器高字节

ECP2R1H – ECP2 通道 1 捕获/占空比寄存器高字节								
地址：0x1B9					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2R1H7	ECP2R1H6	ECP2R1H5	ECP2R1H4	ECP2R1H3	ECP2R1H2	ECP2R1H1	ECP2R1H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP2R1H	ECP2 通道 1 捕获/占空比寄存器高字节 捕获模式下，ECP2R1H 是通道 1 的 16 位捕获寄存器的高 8 位。发生捕获事件后，硬件把 16 位定时器 2 的计数值保存到 ECP2R1H:ECP2R1L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP2R1H 是通道 1 的 16 位占空比寄存器的高 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP2R1H:ECP2R1L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP2R2L – ECP2 通道 2 捕获/占空比寄存器低字节

ECP2R1L – ECP2 通道 2 捕获/占空比寄存器低字节								
地址：0x1BA					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2R2L7	ECP2R2L6	ECP2R2L5	ECP2R2L4	ECP2R2L3	ECP2R2L2	ECP2R2L1	ECP2R2L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP2R2L	ECP2 通道 2 捕获/占空比寄存器低字节 捕获模式下，ECP2R2L 是通道 2 的 16 位捕获寄存器的低 8 位。发生捕获事件后，硬件把 16 位定时器 2 的计数值保存到 ECP2R2H:ECP2R2L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP2R2L 是通道 2 的 16 位占空比寄存器的低 8 位，配置占空比时，							

		软件把 16 位占空比值写入到 ECP2R2H:ECP2R2L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。

ECP2R2H – ECP2 通道 2 捕获/占空比寄存器高字节

ECP2R2H – ECP2 通道 2 捕获/占空比寄存器高字节								
地址：0x1BB					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2R2H7	ECP2R2H6	ECP2R2H5	ECP2R2H4	ECP2R2H3	ECP2R2H2	ECP2R2H1	ECP2R2H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP2R2H	ECP2 通道 2 捕获/占空比寄存器高字节 捕获模式下，ECP2R2H 是通道 2 的 16 位捕获寄存器的高 8 位。发生捕获事件后，硬件把 16 位定时器 2 的计数值保存到 ECP2R2H:ECP2R2L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP2R2H 是通道 2 的 16 位占空比寄存器的高 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP2R2H:ECP2R2L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP2R3L – ECP2 通道 3 捕获/占空比寄存器低字节

ECP2R3L – ECP2 通道 3 捕获/占空比寄存器低字节								
地址：0x1BC					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2R3L7	ECP2R3L6	ECP2R3L5	ECP2R3L4	ECP2R3L3	ECP2R3L2	ECP2R3L1	ECP2R3L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP2R3L	ECP2 通道 3 捕获/占空比寄存器低字节 捕获模式下，ECP2R3L 是通道 3 的 16 位捕获寄存器的低 8 位。发生捕获事件后，硬件把 16 位定时器 2 的计数值保存到 ECP2R3H:ECP2R3L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP2R3L 是通道 3 的 16 位占空比寄存器的低 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP2R3H:ECP2R3L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP2R3H – ECP2 通道 3 捕获/占空比寄存器高字节

ECP2R3H – ECP2 通道 3 捕获/占空比寄存器高字节								
地址：0x1BD					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2R3H7	ECP2R3H6	ECP2R3H5	ECP2R3H4	ECP2R3H3	ECP2R3H2	ECP2R3H1	ECP2R3H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP2R3H	ECP2 通道 3 捕获/占空比寄存器高字节 捕获模式下，ECP2R3H 是通道 3 的 16 位捕获寄存器的高 8 位。发生捕获事件后，硬件把 16 位定时器 2 的计数值保存到 ECP2R3H:ECP2R3L 中作为捕获值。读取 16							

		位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP2R3H 是通道 3 的 16 位占空比寄存器的高 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP2R3H:ECP2R3L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。

ECP2R4L – ECP2 通道 4 捕获/占空比寄存器低字节

ECP2R4L – ECP2 通道 4 捕获/占空比寄存器低字节								
地址：0x1BE					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2R4L7	ECP2R4L6	ECP2R4L5	ECP2R4L4	ECP2R4L3	ECP2R4L2	ECP2R4L1	ECP2R4L0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP2R4L	ECP2 通道 4 捕获/占空比寄存器低字节 捕获模式下，ECP2R4L 是通道 4 的 16 位捕获寄存器的低 8 位。发生捕获事件后，硬件把 16 位定时器 2 的计数值保存到 ECP2R4H:ECP2R4L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP2R4L 是通道 4 的 16 位占空比寄存器的低 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP2R4H:ECP2R4L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

ECP2R4H – ECP2 通道 4 捕获/占空比寄存器高字节

ECP2R4H – ECP2 通道 4 捕获/占空比寄存器高字节								
地址：0x1BF					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2R4H7	ECP2R4H6	ECP2R4H5	ECP2R4H4	ECP2R4H3	ECP2R4H2	ECP2R4H1	ECP2R4H0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7:0	ECP2R4H	ECP2 通道 4 捕获/占空比寄存器高字节 捕获模式下，ECP2R4H 是通道 4 的 16 位捕获寄存器的高 8 位。发生捕获事件后，硬件把 16 位定时器 2 的计数值保存到 ECP2R4H:ECP2R4L 中作为捕获值。读取 16 位捕获值时须先读取高 8 位，再读取低 8 位。 PWM 模式下，ECP2R4H 是通道 4 的 16 位占空比寄存器的高 8 位，配置占空比时，软件把 16 位占空比值写入到 ECP2R4H:ECP2R4L 中。写入 16 位占空比值时须先写入高 8 位，再写入低 8 位。							

PIR1 – 外设中断标记寄存器

PIR1 – 外设中断标记寄存器								
地址：0x0C					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2IF	ADIF	RCIF	TXIF	SPIF	ECP1IF	T2IF	T1IF
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	ECP2IF	ECP2 中断标记位						

		发生捕获、比较匹配或自动关闭时，会置位 ECP2IF。如果 ECP2 中断被使能时，会产生 ECP2 中断。软件须写“0”来清零 ECP2IF 位。

PIE1 – 外设中断使能寄存器

PIR1 – 外设中断使能寄存器								
地址：0x8C					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2IE	ADIE	RCIE	TXIE	SPIE	ECP1IE	T2IE	T1IE
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	ECP2IE	ECP2 中断使能位 1：ECP2 中断被使能 0：ECP2 中断被禁止						

TRISA – PORTA 三态控制寄存器

TRISA – PORTA 三态控制寄存器								
地址：0x85					默认值：1111_1111			
Bit	7	6	5	4	3	2	1	0
Name	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
7	TRISA7	-						
6	TRISA6	RA6 三态控制 1：RA6 引脚被配置为输入，捕获源 T2C3 可由此输入 0：RA6 引脚被配置为输出，PWM 输出信号 T2C3 可由此输出						
5	TRISA5	RA5 三态控制 1：RA5 引脚被配置为输入，捕获源 T2C1 可由此输入 0：RA5 引脚被配置为输出，PWM 输出信号 T2C1 可由此输出						
4	TRISA4	RA4 三态控制 1：RA4 引脚被配置为输入，捕获源 T2C3 可由此输入 0：RA4 引脚被配置为输出，PWM 输出信号 T2C3 可由此输出						
3	TRISA3	RA3 三态控制 1：RA3 引脚被配置为输入，捕获源 T2C4 可由此输入 0：RA3 引脚被配置为输出，PWM 输出信号 T2C4 和 T2C2N 可由此输出						
2	TRISA2	RA2 三态控制 1：RA2 引脚被配置为输入，捕获源 T2C3 可由此输入 0：RA2 引脚被配置为输出，PWM 输出信号 T2C3 和 T2C1N 可由此输出						
1	TRISA1	RA1 三态控制 1：RA1 引脚被配置为输入，捕获源 T2C4 可由此输入 0：RA1 引脚被配置为输出，PWM 输出信号 T2C4 可由此输出						
0	TRISA0	RA0 三态控制 0：RA0 引脚被配置为输出，PWM 输出信号 T2C3N 可由此输出						

TRISB – PORTB 三态控制寄存器

TRISB – PORTB 三态控制寄存器								
地址：0x86					默认值：1111_1111			
Bit	7	6	5	4	3	2	1	0
Name	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
0	TRISB0	RB0 三态控制 1：RB0 引脚被配置为输入，捕获源 T2C2 可由此输入 0：RB0 引脚被配置为输出，PWM 输出信号 T2C2 可由此输出						

串行收发器(USART)

- 全双工异步收发（独立的串行接收和发送寄存器）
- 异步或同步操作
- 主机或从机操作
- 高精度的波特率发生器
- 支持 8，或 9 个数据位和 1 个停止位
- 数据过速检测
- 帧错误检测
- 噪声滤波，包括错误的起始位检测以及数字低通滤波器
- 发送结束中断，接收结束中断
- 地址检测模式
- 波特率自动检测和校准
- 接收到间隔字符时唤醒
- 间隔字符发送

综述

USART 主要包括三个部分：时钟发生器，发送器和接收器。控制和状态寄存器由这三个部分共享。时钟发生器由波特率发生器和同步从机操作模式下外部输入时钟的同步逻辑组成。UCK 引脚只用于同步传输模式。发送器包括一个写数据缓冲器，串行移位寄存器，奇偶发生器以及处理不同帧格式所需的控制逻辑。写数据缓冲器允许连续发送数据而不会在数据帧之间引入延迟。接收器具有时钟和数据恢复单元，用于异步数据的接收。除了恢复单元，接收器还包括奇偶校验，控制逻辑，串行移位寄存器和一个两级接收缓冲器。接收器支持与发送器相同的帧格式，而且可以检测帧错误，数据过速错误。

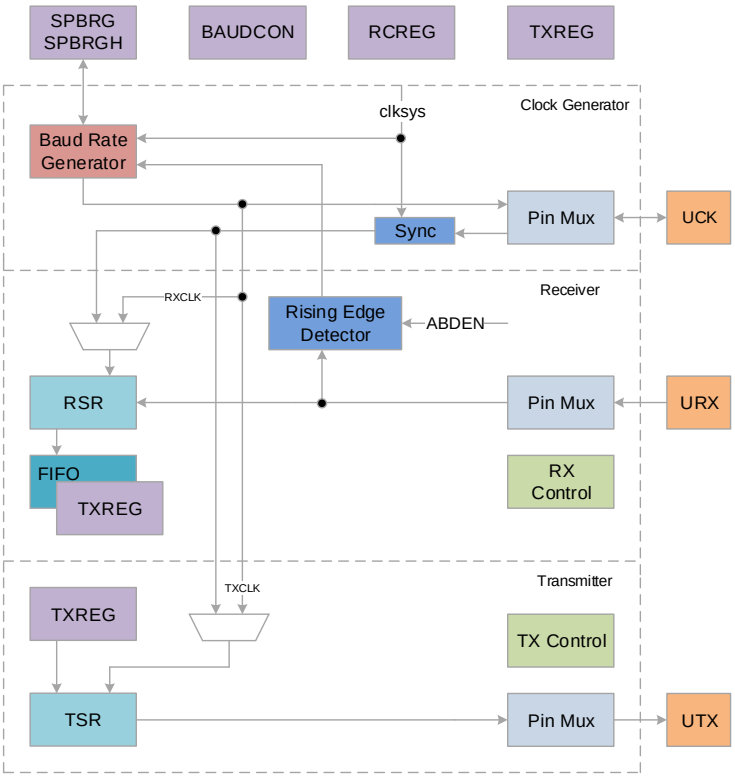


Figure 5 USART 结构图

时钟产生

时钟产生逻辑为发送器和接收器产生基础时钟。USART 支持 4 种模式的时钟：异步模式，主机同步模式，以及从机同步模式。位于 TXSTA 寄存器的 SYNC 位选择同步或异步模式。同步模式下，位于 TXSTA 的 CSRC 位决定了时钟源是由内部产生(主机模式)还是外部产生(从机模式)。

波特率发生器

波特率发生器（Baud Rate Generator， BRG）是一个 8 位或 16 位定时器，专用于支持异步和同步 EUSART 操作。默认情况下，位于 BAUDCON 寄存器的 BRG16 位为 0，BRG 工作在 8 位模式。当设置 BRG16 位为 1 时，BRG 工作在 16 位模式。

SPBRGH 和 SPBRG 寄存器对决定自由运行的波特率定时器的周期。在异步模式下，波特率周期的倍频值由 TXSTA 寄存器的 BRGH 位和 BAUDCON 寄存器的 BRG16 位决定。在同步模式下， BRGH 位被忽略。

下表给出了各种工作模式下计算波特率（位/秒）以及 BRG 值的公式。

SYNC	BRG16	BRGH	波特率计算公式[1]	BRG 值计算公式
0	0	0	BAUD = f_{sys}/(64*(BRG+1))	BRG = f_{sys}/64*BAUD) - 1
0	0	1	BAUD = f_{sys}/(16*(BRG+1))	BRG= f_{sys}/(16*BAUD) - 1
0	1	0		
0	1	1	BAUD = f_{sys}/(4*((BRG|0x3)+1))	BRG= f_{sys}/(4*BAUD) - 1
1	0	X		
1	1	X		

说明：

1. 波特率定义为每秒的位传输速度（bps）；
2. BUAD 为波特率，f_{sys} 为系统时钟，BRG 为波特率寄存器 SPBRGH 和 SPBRG 的组合值。

使用高波特率（BRGH = 1）或 16 位 BRG（BRG16 = 1）有助于降低波特率误差。16 位 BRG 模式用于在快速振荡器频率条件下实现低波特率。

将新值写入 SPBRGH 和 SPBRGL 寄存器对将导致 BRG 定时器复位（或清零）。这可以确保 BRG 无需等待定时器溢出就可以输出新的波特率。如果在有效接收操作期间更改了系统时钟，则可能导致接收错误或数据丢失。要避免这个问题，请检查 RCIDL 位的状态以确保在更改系统时钟之前接收操作处于空闲状态。

自动波特率检测

USART 模块支持波特率自动检测和校准。

在自动波特率检测（Auto-Baud Detect，ABD）模式下，提供给 BRG 的时钟是反向的。不是由 BRG 向传入的 URX 信号提供时钟，而是由 URX 信号为 BRG 定时。波特率发生器用来测量接收 55h 字符的周期(ASCII“U”，也是 LIN 总线的同步字符)。该字符的惟一特性是它具有 5 个上升沿（包括停止位边沿在内）。

将 BAUDCON 寄存器的 ABDEN 位置 1 可启动自动波特率校准序列。执行 ABD 序列时， USART 状态机保持空闲状态。在起始位之后，SPBRG 使用 BRG 计数器时钟在接收信号的第一个上升沿开始计数。在第 8 个位周期结束时，在 URX 引脚上出现第 5 个上升沿。此时，会将相应 BRG 周期内累计所得的值保存到相应的 SPBRGH 和 SPBRG 寄存器对中，ABDEN 位自动清零，并将 RCIF 中断标志位置 1。要清零 RCIF

中断标志位，需要读 RCREG 中的值。应丢弃 RCREG 的内容。当在不使用 SPBRGH 寄存器的模式下进行校准时，用户可以通过检查 SPBRGH 寄存器中的值是否为 00h 来验证 SPBRG 寄存器是否未溢出。

BRG 自动波特率时钟由 BRG16 和 BRGH 位决定，如下表所示。在自动波特率检测期间，SPBRGH 和 SPBRG 寄存器都可用作 16 位计数器，与 BRG16 位的设置无关。校准波特率周期时，SPBRGH 和 SPBRG 寄存器以 BRG 基本时钟速率的 1/8 作为时钟。此时测量一个字节的时间就相当于全速时钟模式下传输一位所需的平均时间。

Table 2 BRG 计数器时钟速率

BRG16	BRGH	BRG 基本时钟	BRG ABD 时钟
0	0	$f_{sys}/64$	$f_{sys}/512$
0	1	$f_{sys}/16$	$f_{sys}/128$
1	0	$f_{sys}/16$	$f_{sys}/128$
1	1	$f_{sys}/4$	$f_{sys}/32$

自动波特率溢出

在自动波特率检测过程中，如果在 URX 引脚上检测到第 5 个上升沿之前，波特率计数器溢出，则 BAUDCON 寄存器的 ABDOVF 位置 1。ABDOVF 位指示计数器超出了 SPBRGH:SPBRG 寄存器对的 16 位所能允许的最大计数值。在 ABDOVF 置 1 后，计数器将一直计数，直到在 URX 引脚上检测到第 5 个上升沿为止。检测到第 5 个 URX 边沿时，硬件将置位 RCIF 中断标志位并清零 BAUDCON 寄存器的 ABDEN 位。随后通过读 RCREG 寄存器，RCIF 中断标志位也将清零。BAUDCON 寄存器的 ABDOVF 标志位可用软件直接清零。

要在 RCIF 中断标志位置 1 之前终止自动波特率检测过程，可依次将 BAUDCON 寄存器的 ABDEN 位和 ABDOVF 位清零。如果没有先将 ABDEN 位清零，则 ABDOVF 位保持置 1 状态。

外部时钟

同步从机操作模式由外部时钟驱动。外部时钟经过同步寄存器和边沿检测器之后才被发送器和接收器使用，这一过程会引入两个系统时钟的延时，因此外部 UCK 的最大时钟频率由以下公式限制：

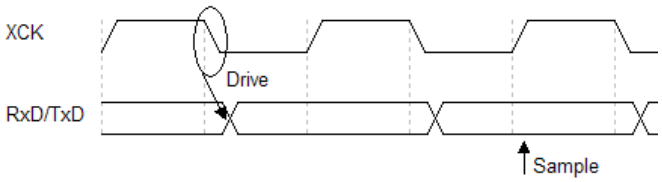
$$f_{UCK} < f_{sys}/4$$

要注意 fsys 有系统时钟的稳定性决定，为了防止因频率漂移而丢失数据，建议保留足够的裕量。

同步时钟操作

同步模式下，UCK 引脚被用于时钟输入（从机模式）或时钟输出（主机模式）。时钟的边沿与数据采样和数据变化关系的基本规律是：对数据输入端（URX）采样所使用的时钟沿与数据输出端变化所使用的时钟沿是相反的。

SCKP = 0



SCKP = 1

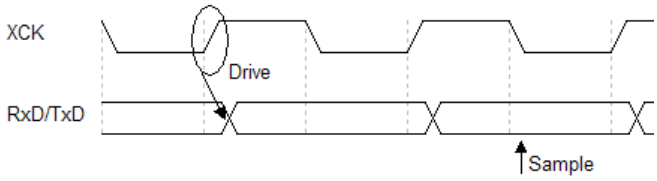


Figure 6 同步模式下的 UCK 时序

如上图所示，当 SCKP 值为“0”时，在 UCK 的下降沿改变数据输出，在 UCK 的上升沿进行数据采样；当 SCKP 值为“1”时，在 UCK 的上升沿改变数据输出，在 UCK 的下降沿进行数据采样。

USART 异步模式

USART 采用标准非归零（non-return-to-zero，NRZ）格式发送和接收数据。NRZ 实现为两种电平：VOH 标记状态（mark state）代表 1 数据位，而 VOL 空格状态（space state）代表 0 数据位。NRZ 指的是连续发送相同值的数据位时，将保持该位的输出电平不变，而不会在发送完每个位后返回到中间电平。NRZ 发送端口在标记状态空闲。每个字符发送包含 1 个起始位及随后的 8 个或 9 个数据位，并始终由 1 个或多个停止位终止。起始位始终处于空格状态，停止位始终处于标记状态。最常用的数据格式为 8 位。每个发送位持续时间为 1/(波特率)。使用专用 8 位/16 位波特率发生器产生标准波特率频率。

USART 首先发送和接收 LSB。USART 发送器和接收器在功能上是相互独立的，但采用相同的数据格式和波特率。硬件不支持奇偶校验，但可以用软件实现并作为第 9 个数据位存储。

USART 异步发送器

下图给出了 USART 发送器框图。发送器的核心是串行发送移位寄存器(Transmit Shift Register，TSR)，该寄存器不能用软件直接访问。TSR 从发送缓冲区（即 TXREG 寄存器）获取数据。

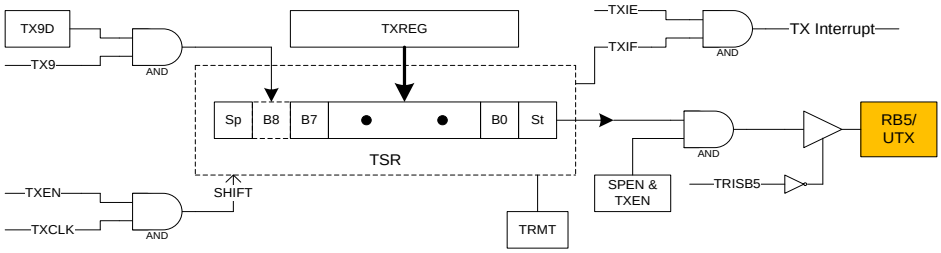


Figure 7 USART 异步发送框图

使能发送器

USART 发送器可通过配置以下 3 个控制位使能为异步操作：

- TXEN = 1
- SYNC = 0
- SPEN = 1

假定所有其他 USART 控制位均处于其默认状态。将 TXSTA 寄存器的 TXEN 位置 1 可使能 USART 的发送器电路。清零 TXSTA 寄存器的 SYNC 位可将 EUSART 配置为异步操作。将 RCSTA 寄存器的 SPEN 位置 1 可使能 USART。软件需要配置相应的 TRIS 位将 UTX 引脚配置为输出。如果与模拟外设共用 UTX 引脚，那么模拟 I/O 功能必须通过将相应的 ANSEL 位清零来禁止。

发送数据

通过向 TXREG 寄存器写入一个字符来启动发送。如果这是第一个字符，或前一个字符已经从 TSR 中送

出，TXREG 中的数据就立即被传送到 TSR 寄存器。如果 TSR 仍包含前一个字符的全部或部分，则新字符数据保存在 TXREG 中，直到前一个字符的停止位发送完毕。停止位发送后立即将 TXREG 中待处理的字符传送到 TSR。TXREG 中的数据被传送到 TSR 后，立即开始起始位、数据位和停止位序列的发送。

发送 9 位字符

USART 支持 9 位字符发送。当 TXSTA 寄存器的 TX9 位置 1 时，USART 将在发送每个字符时移出 9 位。TXSTA 寄存器的 TX9D 位是第 9 个数据位，也是最高有效位。发送 9 位数据时，必须在将低 8 位写入 TXREG 之前先写入 TX9D 数据位。写入 TXREG 后，所有 9 个数据位将被立即传送到 TSR 移位寄存器。有多个接收器时，可使用一种特殊的 9 位地址模式。关于地址模式的更多信息，请参见地址检测章节。

发送数据极性

发送数据的极性用 BAUDCON 寄存器的 SCKP 位来控制。该位的默认状态为 0，这时选择高电平发送空闲和数据位。将 SCKP 位设置为 1 时将翻转发送数据，产生低电平空闲和数据位。SCKP 位仅控制异步模式下发送数据的极性。在同步模式下，SCKP 具有不同的功能，参见同步时钟操作章节。

发送标志位与中断

只有 USART 发送器发送完 TSR 中的字符且 TXREG 中没有等待发送的字符，PIR1 寄存器的 TXIF 中断标志位才会被置 1。软件可通过写 0 对其进行清零。将 PIE1 寄存器的 TXIE 中断允许位置 1 可允许 TXIF 中断。

TSR 的状态

TXSTA 寄存器的 TRMT 位表示 TSR 寄存器的状态，该位是只读位。TSR 寄存器为空时，TRMT 位置 1，当有字符从 TXREG 传送到 TSR 寄存器时，该位清零。TRMT 位保持清零状态，直到所有位移出 TSR 寄存器为止。该位不与任何中断逻辑有关，因此用户必须查询该位以确定 TSR 的状态。

异步发送设置

1. 初始化 SPBRGH 和 SPBRGL 寄存器对以及 BRGH 和 BRG16 位，以获得所需的波特率
2. 通过清零 SYNC 位并将 SPEN 位置 1，使能异步串口。
3. 如果需要发送 9 位数据，将 TX9 控制位置 1。接收器置于地址检测模式时，第 9 个数据位置 1 表示低 8 个数据位为地址。
4. 如果需要翻转发送结果，将 SCKP 位置 1。
5. 将 TXEN 控制位置 1 来使能发送。
6. 如果需要中断，将 PIE1 寄存器的 TXIE 中断允许位置 1。如果 INTCON 寄存器的 GIE 和 PEIE 位也置 1，则 TXIF 位被置位时会立即产生中断。
7. 如果选择了发送 9 位数据，应将第 9 位装入 TX9D 数据位。
8. 将 8 位数据装入 TXREG 寄存器。这将立即启动发送。
9. 数据发送完毕后且没有再次写入数据到 TXREG 寄存器，TXIF 位将会被置位，发送完成。

USART 异步接收器

异步模式通常用于 RS-232 系统中。下图给出了 USART 接收器框图。数据通过 URX 引脚接收并驱动数据恢复模块。数据恢复模块实际上是以 16 倍波特率工作的高速移位器，而串行接收移位寄存器（Receive Shift Register，RSR）工作在比特率下。字符的所有 8 位或 9 位移入后被立即传送到双字符的先进先出（First-In-First-Out，FIFO）存储区中。FIFO 缓冲区允许先接收 2 个完整字符和第 3 个字符的起始位后，再由软件将数据提供给 USART 接收器。FIFO 和 RSR 寄存器不能直接用软件访问，可通过 RCREG 寄存器访问所接收的数据。

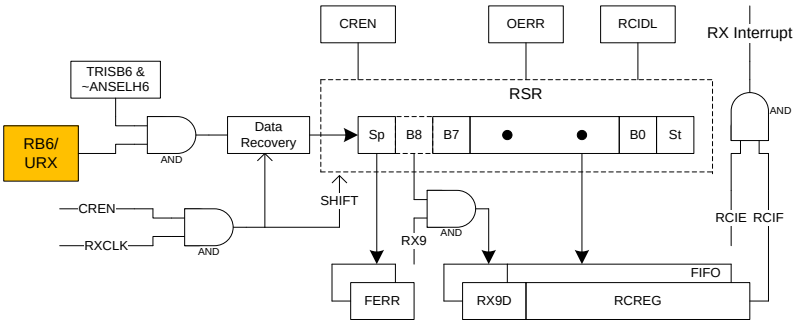


Figure 8 USART 异步接收器框图

使能接收器

USART 接收器可通过配置以下 3 个控制位使能为异步操作：

- CREN = 1
- SYNC = 0
- SPEN = 1

假定所有其他 USART 控制位均处于其默认状态。将 RCSTA 寄存器的 CREN 位置 1 来使能 USART 的接收器电路。清零 TXSTA 寄存器的 SYNC 位可将 USART 配置为异步操作。将 RCSTA 寄存器的 SPEN 位置 1 可使能 USART。软件必须将相应的 TRIS 置 1 以将 URX 引脚配置为输入。如果 URX 功能在模拟引脚上，那么还必须清零相应的 ANSEL 位以使接收器正常工作。

接收数据

接收器的数据恢复电路在第 1 位的下降沿启动字符接收。第 1 位也称起始位，始终为零。数据恢复电路计数半个位时间至起始位的中点，并检验该位是否仍为零。如果该位不为零，则数据恢复电路中止字符接收，而不会产生错误，并继续查找起始位的下降沿。如果起始位零校验通过，则数据恢复电路计数 1 个完整的位时间至下一个位的中点。然后，由择多检测电路对该位采样并将结果（0 或 1）移入 RSR。重复此过程直到所有数据位均被采样并移入 RSR 为止。测量最后一个位时间并采样其电平。此位为停止位，始终为 1。如果数据恢复电路在停止位处采样到 0，则此字符的帧错误标志置 1，否则清零。更多关于帧错误的信息，请参见接收帧错误章节。

接收到所有数据位和停止位之后，立即将 RSR 中的字符传送到 USART 接收 FIFO，且 PIR1 寄存器的 RCIF 中断标志位置 1。通过读 RCREG 寄存器，将 FIFO 顶部的字符送出 FIFO，获取接收到的数据。如果接收 FIFO 溢出，在溢出条件清除前不会接收更多字符。更多关于溢出错误的信息，请参见接收溢出错误章节。

接收 9 位字符

USART 支持 9 位字符接收。当 RCSTA 寄存器的 RX9 位置 1 时， USART 将接收到的每个字符的 9 个位移入 RSR。RCSTA 寄存器的 RX9D 位是第 9 位，也是接收 FIFO 顶部未读字符的最高有效数据位。从接收 FIFO 缓冲区读 9 位数据时，在读 RCREG 的低 8 位之前必须先读 RX9D 数据位。

地址检测

当多个接收器共用同一条传输线时（如在 RS-485 系统中），可使用特殊地址检测模式。将 RCSTA 寄存器的 ADDEN 位置 1 可使能地址检测。地址检测要求接收 9 位字符。使能地址检测时，只有第 9 个数据位置 1 的字符会被传送到接收 FIFO 缓冲区，从而将 RCIF 中断标志位置 1。忽略所有其他字符。接收到地址字符后，用户软件判断此地址是否与其匹配。地址匹配时，用户软件必须在出现下一个停止位之前通过清

零 ADDEN 位来禁止地址检测。当用户软件检测到报文的末尾（由所使用的报文协议确定）时，软件通过将 ADDEN 位置 1 把接收器重新置于地址检测模式。

接收结束标志及中断

只要 USART 接收器使能且接收 FIFO 中存在未读字符， PIR1 寄存器的 RCIF 中断标志位就会置 1。RCIF 中断标志位是只读位，不能用软件置位 或清零。

将以下位置 1 可允许 RCIF 中断：

• PIE1 寄存器的 RCIE 中断允许位

• INTCON 寄存器的 PEIE 外设中断允许位

• INTCON 寄存器的 GIE 全局中断允许位

当 FIFO 中存在未读字符时，不管中断允许位的状态如何，RCIF 中断标志位都会置 1。

接收帧错误

接收 FIFO 缓冲区中的每个字符都有相应的帧错误状态位。帧错误表示在预期时间内未接收到停止位。通过 RCSTA 寄存器的 FERR 位可访问帧错误状态。FERR 位表示接收 FIFO 顶部未读字符的状态。因此，在读 RCREG 之前必须先读 FERR 位。

FERR 位是只读位，只用于接收 FIFO 顶部的未读字符。帧错误（FERR = 1）不会阻止接收其他字符。此时不必将 FERR 位清零。从 FIFO 缓冲区读下一个字符将前进至 FIFO 的下一个字符和下一个对应的帧错误状态。

将 RCSTA 寄存器的 CREN 位清零可复位 USART 接收器，从而将 FERR 位强制清零。帧错误本身不会产生中断。

接收溢出错误

接收 FIFO 缓冲区可保存两个字符。如果在访问 FIFO 之前接收到完整的第三个字符，则会产生溢出错误。此时，RCSTA 寄存器的 OERR 位置 1。FIFO 缓冲区中已有的字符可被读出，但溢出错误清除之前不会接收其他字符。必须通过清零 RCSTA 寄存器的 CREN 位使 USART 接收器复位来清除该错误。

异步接收设置

1. 初始化 SPBRGH 和 SPBRG 寄存器对以及 BRGH 和 BRG16 位，以获得所需的波特率。
2. 将 URX 引脚的 ANSEL 位清零（如果适用）。
3. 通过将 SPEN 位置 1，使能串口。SYNC 位必须清零才能进行异步操作。
4. 如果需要中断，将 PIE1 寄存器的 RCIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
5. 如果需要接收 9 位数据，将 RX9 位置 1。
6. 通过将 CREN 位置 1，使能接收。
7. 当字符从 RSR 传送到接收缓冲区时，RCIF 中断标志位置 1。如果 RCIE 中断允许位也置 1，则产生中断。
8. 读 RCSTA 寄存器以获取错误标志和第 9 个数据位（如果使能了 9 位数据接收）。
9. 通过读 RCREG 寄存器从接收缓冲区获取所接收数据的低 8 位。
10. 如果发生溢出，则通过清零 CREN 接收器使能位来清零 OERR 标志。

9 位地址检测模式设置

此模式通常用于 RS-485 系统中。要设置使能了地址检测的异步接收：

1. 初始化 SPBRGH 和 SPBRG 寄存器对以及 BRGH 和 BRG16 位，以获得所需的波特率。
2. 将 URX 引脚的 ANSEL 位清零（如果适用）。

3. 通过将 SPEN 位置 1，使能串口。SYNC 位必须清零才能进行异步操作。
4. 如果需要中断，将 PIE1 寄存器的 RCIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
5. 通过将 RX9 位置 1，使能 9 位接收。
6. 通过将 ADDEN 位置 1，使能地址检测。
7. 通过将 CREN 位置 1，使能接收。
8. 当第 9 位置 1 的字符从 RSR 传送到接收缓冲区时，RCIF 中断标志位置 1。如果 RCIE 中断允许位也置 1，则产生中断。
9. 读 RCSTA 寄存器以获取错误标志。第 9 个数据位始终置 1。
10. 通过读 RCREG 寄存器从接收缓冲区获取所接收数据的低 8 位。软件判断此地址是否为器件地址。
11. 如果发生溢出，则通过清零 CREN 接收器使能位来清零 OERR 标志。
12. 如果器件被寻址，将 ADDEN 位清零以允许所有接收到的数据送入接收缓冲区并产生中断。

USART 同步模式

同步模式下有三条信号线：两根数据线和一根时钟线。USART 可作为主器件，也可作为从器件。主器件包含生成波特率所需的电路，并为系统中的所有器件提供时钟。从器件使用主器件提供的外部时钟将串行数据移入移出相应的接收和发送移位寄存器。

同步发送时不使用起始位和停止位。

USART 同步主机模式

使用以下位将 USART 配置为同步主操作：

- SYNC = 1
- CSRC = 1
- SREN = 1（用于单个字符接收）；CREN = 1（用于连续字符接收）
- TXEN = 1（用于连续字符发送）
- SPEN = 1

将 TXSTA 寄存器的 SYNC 位置 1 可将器件配置为同步操作。将 TXSTA 寄存器的 CSRC 位置 1 可将器件配置为主器件。将 TXSTA 寄存器的 TXEN 位置 1 可使能发送。将 RCSTA 寄存器的 SREN 或 CREN 位置 1 可使能接收，其中，CREN 位为 1 时为连续接收模式，CREN 位为 0 且 SREN 位为 1 时为单字符接收，接收到 1 个字符后 SREN 位将被自动清零。将 RCSTA 寄存器的 SPEN 位置 1 可使能 USART。

主时钟

同步数据传输使用独立于数据线但与数据线同步的时钟线。配置为主器件的器件将时钟信号发送到 UCK 线上。USART 配置为同步发送或接收操作时，还需要使能 UCK 引脚的输出驱动器。串行数据位在每个时钟的上升沿改变，以确保其在时钟的下降沿有效。每个时钟周期传送一个数据位。数据位有多少，就产生多少个时钟周期。

时钟极性

时钟极性通过 BAUDCON 寄存器的 SCKP 位选择。将 SCKP 位置 1 可将时钟空闲状态设置为高电平。当 SCKP 位置 1 时，数据在每个时钟的下降沿改变。将 SCKP 位清零可将空闲状态设置为低电平。当 SCKP 位清零时，数据在每个时钟的上升沿改变。

同步主发送

USART 配置为同步主发送操作时，软件须使能 UTX 和 UCK 引脚的输出驱动器。

通过向 TXREG 寄存器写入一个字符来启动发送。如果 TSR 仍包含前一个字符的全部或部分，则新字符数据保存在 TXREG 中，直到前一个字符的最后一位发送完毕。如果这是第一个字符，或前一个字符被完全从 TSR 中送出，TXREG 中的数据就立即被传送到 TSR。字符发送在数据从 TXREG 送入 TSR 后立即开始。每个数据位在主时钟的时钟上升沿改变，并在下一个时钟上升沿到来前保持有效。

同步主发送设置：

1. 初始化 SPBRGH 和 SPBRG 寄存器以及 BRGH 和 BRG16 位，以获得所需的波特率。
2. 通过将 SYNC、SPEN 和 CSRC 位置 1，使能同步主串行端口。
3. 通过将 SREN 和 CREN 位清零，禁止接收模式。
4. 通过将 TXEN 位置 1，使能发送模式。
5. 如果需要发送 9 位数据，将 TX9 位置 1。
6. 如果需要中断，将 PIE1 寄存器的 TXIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
7. 如果选择发送 9 位数据，将第 9 位数据装入 TX9D 位。
8. 将数据装入 TXREG 寄存器，启动发送。

同步主接收

USART 配置为同步主接收操作时，从 URX 引脚上接收数据，须禁止 URX 引脚的输出驱动器。

在同步模式下，可通过将单字节接收使能位（RCSTA 寄存器的 SREN）或连续接收使能位（RCSTA 寄存器的 CREN）置 1 来使能接收。

SREN 置 1 且 CREN 清零时，一个字符中有多少数据位就产生多少个时钟周期。一个字符接收完成后 SREN 位自动清零。CREN 置 1 时，将连续产生时钟直到 CREN 清零。如果 CREN 在字符接收过程中清零，则 UCK 时钟立即停止，字符中接收到的那部分将被丢弃。如果 SREN 和 CREN 同时置 1，则第一个字符接收完成时 SREN 清零， CREN 的优先级更高。

要启动接收，将 SREN 或 CREN 置 1。在 UCK 时钟引脚的下降沿对 URX 引脚上的数据进行采样，并将采样结果移入接收移位寄存器（RSR）。当 RSR 接收到一个完整的字符后，RCIF 位置 1 且该字符自动传送到两个字符的接收 FIFO。接收 FIFO 顶部字符的低 8 位在 RCREG 中。只要接收 FIFO 中有未读字符，RCIF 位就保持置 1 状态。

接收溢出错误

接收 FIFO 缓冲区可容纳两个字符。如果在读 RCREG 之前接收到完整的第三个字符，则会产生溢出错误。此时，RCSTA 寄存器的 OERR 位置 1。FIFO 中的前一个数据不会被覆盖。FIFO 缓冲区中的两个字符可被读出，但溢出错误清除之前不会接收其他字符。只能通过清除溢出条件来清零 OERR 位。如果 SREN 位置 1 且 CREN 位清零时发生溢出错误，则可通过读 RCREG 来清除该错误。如果 CREN 位置 1 时发生溢出，则可通过清零 RCSTA 寄存器的 CREN 位来清除错误条件。

接收 9 位字符

USART 支持 9 位字符接收。当 RCSTA 寄存器的 RX9 位置 1 时， USART 将在接收每个字符时将 9 个位移入 RSR。RCSTA 寄存器的 RX9D 位是第 9 位，也是接收 FIFO 顶部未读字符的最高有效数据位。从接收 FIFO 缓冲区读 9 位数据时，在读 RCREG 的低 8 位之前必须先读 RX9D 数据位。

同步主接收设置：

1. 初始化 SPBRGH 和 SPBRG 寄存器以及 BRGH 和 BRG16 位，以获得所需的波特率。
2. 将 URX 引脚设置为输入并将相应的 ANSEL 位清零（如果适用）。
3. 通过将 SYNC、SPEN 和 CSRC 位置 1，使能同步主串行端口。
4. 确保 CREN 和 SREN 位清零。
5. 如果需要中断，将 PIE1 寄存器的 RCIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
6. 若需要接收 9 位数据，将 RX9 位置 1。
7. 将 SREN 位置 1 启动接收，或将 CREN 位置 1 启动连续接收。
8. 字符接收完成时，中断标志位 RCIF 置 1。如果中断允许位 RCIE 也置 1，则产生中断。
9. 读 RCSTA 寄存器以获取第 9 个数据位（如果已使能），并判断在接收过程中是否发生了任何错误。
10. 通过读 RCREG 寄存器来读取接收到的 8 位数据。
11. 如果发生了溢出错误，则可通过清零 RCSTA 寄存器的 CREN 位来清除该错误。

USART 同步从机模式

使用以下位将 USART 配置为同步从操作：

- SYNC = 1
- CSRC = 0
- CREN = 1（用于连续字符接收）
- TXEN = 1（用于连续字符发送）
- SPEN = 1

将 TXSTA 寄存器的 SYNC 位置 1 可将器件配置为同步操作。将 TXSTA 寄存器的 CSRC 位清零可将器件配置为从器件。将 TXSTA 寄存器的 TXEN 位置 1 可使能发送。将 RCSTA 寄存器的 CREN 位置 1 可使能接收。将 RCSTA 寄存器的 SPEN 位置 1 可使能 USART。

从时钟

同步数据传输使用独立于数据线但与数据线同步的时钟线。配置为从器件的器件在 UCK 线上接收时钟信号。器件配置为同步从发送或接收操作时，须禁止 UCK 引脚的输出驱动器。如果 UCK 功能在模拟引脚上，那么还必须清零相应的 ANSEL 位来正常工作。串行数据位在每个时钟的上升沿改变，以确保其在时钟的下降沿有效。每个时钟周期传送一个数据位。数据位有多少，就产生多少个接收时钟周期。

同步从发送

同步从发送模式和同步主发送模式的工作原理是相同的，参见同步主发送章节。

同步从发送设置：

1. 将 SYNC 和 SPEN 位置 1 并清零 CSRC 位。
2. 将 UCK 引脚的 ANSEL 位清零（如果适用）。
3. 将 CREN 和 SREN 位清零。
4. 如果需要中断，将 PIE1 寄存器的 TXIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
5. 如果需要发送 9 位数据，将 TX9 位置 1。
6. 通过将 TXEN 位置 1，使能发送。
7. 如果选择发送 9 位数据，则将最高有效位插入 TX9D 位。
8. 通过将低 8 位写入 TXREG 寄存器来启动发送。

同步从接收

同步从接收模式和同步主接收模式的工作原理是基本相同的，参见同步主接收章节，不同之处在于 SREN 位在从模式下为无关位，因此 CREN 位始终要置 1，接收器从不空闲。

同步从接收设置：

1. 将 SYNC 和 SPEN 位置 1 并清零 CSRC 位。
2. 将 UCK 和 URX 引脚设置为输入并将相应的 ANSEL 位清零（如果适用）。
3. 如果需要中断，将 PIE1 寄存器的 RCIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
4. 如果需要接收 9 位数据，将 RX9 位置 1。
5. 将 CREN 位置 1，以使能接收。
6. 接收完成后， RCIF 位置 1。如果 RCIE 位也置 1，则产生中断。
7. 如果使能了 9 位模式，从 RCSTA 寄存器的 RX9D 位取出最高有效位。
8. 通过读 RCREG 寄存器从接收 FIFO 取出低 8 位。
9. 如果发生了溢出错误，则可通过清零 RCSTA 寄存器的 CREN 位来清除该错误。

USART 寄存器

Table 3 USART 寄存器列表

寄存器	地址	默认值	描述
RCSTA	0x18	0x00	USART 接收控制和状态寄存器
TXREG	0x19	0x00	USART 发送数据寄存器
RCREG	0x1A	0x00	USART 接收数据寄存器
TXSTA	0x98	0x02	USART 发送控制和状态寄存器
SPBRG	0x99	0x00	USART 波特率寄存器低字节
SPBRGH	0x9A	0x00	USART 波特率寄存器高字节
BAUDCTL	0x9B	0x40	USART 波特率控制寄存器

RCSTA – *USART 接收控制和状态寄存器*

RCSTA – USART 接收控制和状态寄存器								
地址: 0x18					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
R/W	R/W	R/W	R/W	R/W	R/W	R	R	R

Bit	Name	R/W	描述
7	SPEN	R/W	串行端口使能控制位 1：串行端口被使能 0：串行端口被禁止
6	RX9	R/W	第 9 位接收使能控制位 1：选择 9 位数据接收模式 0：选择 8 位数据接收模式
5	SREN	R/W	单字节接收使能位。 异步模式：无关；

			同步主机模式：1：使能单字节接收，0：禁止单字节接收； 同步从机模式：无关
4	CREN	R/W	连续接收使能位 异步模式：1：使能接收器，0：禁止接收器； 同步模式：1：使能连续接收（CREN 优于 SREN），0：禁止连续接收
3	ADDEN	R/W	地址检测使能位 异步 9 位模式（RX9=1）：1：使能地址检测，0：禁止地址检测； 异步 8 位模式（RX9=0）：无关
2	FERR	R	帧错误标志位。 当 FE 为“1”时，表明接收数据缓冲器接收到的数据有帧错误，即第一个停止位为“0”。当 FE 为“0”时，表明接收数据缓冲器接收到的数据没有帧错误，即第一个停止位为“1”。FE 被置位后会一直有效到 UDR 被读取。对 UCSRA 进行写入时，FE 这一位要写“0”。
1	OERR	R	数据溢出标志位。 当接收缓冲器为满（包含了两个数据），接收移位寄存器中有数据，若此时检测到一个新的起始位，数据溢出产生，DOR 被置位，一直有效到 UDR 被读取。对 UCSRA 进行写入时，DOR 这一位要写“0”。
0	RX9D	R/W	接收数据第 9 位。 当数据帧长度为 9 位时，RX9D 是接收数据的最高位。读取 RCREG 所包含的低 8 位数据之前要先读取 RX9D。可作为地址/数据位或校验位。

TXSTA – *USART 发送控制和状态寄存器*

TXSTA – USART 发送控制和状态寄存器								
地址: 0x98					默认值: 0x02			
Bit	7	6	5	4	3	2	1	0
Name	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W

Bit	Name	R/W	描述
7	CSRC	R/W	
6	TX9	R/W	第 9 位发送使能控制位 1：选择 9 位数据发送模式 0：选择 8 位数据发送模式
5	TXEN	R/W	发送使能位。 1：发送被使能 0：发送被禁止
4	SYNC	R/W	串口模式选择控制位 1：同步模式 0：异步模式
3	SENDB	R/W	发送 BREAK 字符使能位 异步模式：1：下一次传输发送间隔字符（完成后被硬件清零），0：间隔字符发送完成； 同步模式：无关
2	BRGH	R/W	高速波特率选择控制位 异步模式：1：高速，0：低速； 同步模式：无关
1	TRMT	R	发送移位寄存器状态位。

			1：发送移位寄存器为空 0：发送移位寄存器为满
0	TX9D	R/W	发送数据第 9 位。 当数据帧长度为 9 位时，TX9D 是发送数据的最高位。写入 TXREG 所包含的低 8 位数据之前要先写入 TX9D。可作为地址/数据位或校验位。

BAUDCTL– *USART 波特率控制寄存器*

***BAUDCTL*– USART 波特率控制寄存器**									
地址: 0x9B					默认值: 0x86				
Bit	7		6	5	4	3	2	1	0
Name	ABDOVF		RCIDL	RDTP	SCKP	BRG16	IREN	WUE	ABDEN
R/W	R		R	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	R/W	描述
7	ABDOVF	R	自动波特率检测溢出标志位 异步模式：1：自动波特率检测计时器已溢出，0：自动波特率检测计时器未溢出； 同步模式：无关
6	RCIDL	R	接收器状态标志位 异步模式：1：接收器空闲状态；0：接收器正在接收 同步模式：无关
5	RDTP	R/W	接收数据极性选择位 1：接收数据反向（低电平有效）， 0：接收数据不反向（高电平有效）
4	SCKP	R/W	发送时钟极性选择位 异步模式：1：发送数据反向（空闲状态为低电平），0：发送数据不反向（空闲状态为高电平）； 同步模式：1：时钟反向（空闲状态为高电平），0：时钟不反向（空闲状态为低电平）
3	BRG16	R/W	16 位波特率产生器使能位 1：16 位波特率发生器（SPBRGH 和 SPBRG 均有效） 0：8 位波特率发生器（仅 SPBRG 有效）
2	IREN	R/W	IrDA 模式使能位 1：IrDA 模式被使能 0：IrDA 模式被禁止
1	WUE	R/W	唤醒使能位 异步模式：1：接收器检测到下降沿来临时，置位 RCIF 位，但不接收字符，WUE 位自动清零；0：接收器正常工作 同步模式：无关
0	ABDEN	R/W	自动波特率检测使能位 异步模式：1：自动波特率检测模式被使能（检测完成时清零该位）；0：自动波特率检测被禁止 同步模式：无关

TXREG – *USART 发送数据寄存器*

TXREG **– USART 发送数据寄存器**								
地址: 0x19					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	TXREG7	TXREG6	TXREG5	TXREG4	TXREG3	TXREG2	TXREG1	TXREG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	R/W	描述
7:0	TXREG	R/W	USART 发送的数据。 USAR 发送数据缓冲器和接收数据缓冲器共享 USART 数据寄存器 UDR。将数据写入 UDR 即写入发送数据缓冲器，从 UDR 读取数据即读取接收数据缓冲器。 在 5 到 8 位数据帧模式下，未使用的第 9 位被发送器忽略，而接收器则将它们设置为 0。 只有当 UCSRA 寄存器的 UDRE 标志为“1”时才能对发送缓冲器进行写操作，否则发送器的操作会出错。当发送移位寄存器为空时，发送器会把发送缓冲器中的数据加载到发送移位寄存器中，然后数据串行地从 UTX 引脚输出。 接收缓冲器包含一个两级 FIFO，一旦接收缓冲器被读取，FIFO 就会改变它的状态。

RCREG – *USART 接收数据寄存器*

RCREG **– USART 接收数据寄存器**								
地址: 0x1A					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	RCREG7	RCREG6	RCREG5	RCREG4	RCREG3	RCREG2	RCREG1	RCREG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	R/W	描述
7:0	RCREG	R/W	USART 接收的数据。 USART 发送数据缓冲器和接收数据缓冲器共享 USART 数据寄存器 UDR。将数据写入 UDR 即写入发送数据缓冲器，从 UDR 读取数据即读取接收数据缓冲器。 在 5 到 8 位数据帧模式下，未使用的第 9 位被发送器忽略，而接收器则将它们设置为 0。 只有当 UCSRA 寄存器的 UDRE 标志为“1”时才能对发送缓冲器进行写操作，否则发送器的操作会出错。当发送移位寄存器为空时，发送器会把发送缓冲器中的数据加载到发送移位寄存器中，然后数据串行地从 UTX 引脚输出。 接收缓冲器包含一个两级 FIFO，一旦接收缓冲器被读取，FIFO 就会改变它的状态。

SPBRG – *USART 波特率寄存器低字节*

SPBRG **– USART 波特率寄存器低字节**								
地址: 0x99					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	R/W	描述
7:0	BRG[7:0]	R/W	USART 波特率寄存器的低字节部分。 USART 波特率寄存器包含 SPBRG 和 SPBRGH 两部分，用来设置通信的波特率。

UBRRH – *USART 波特率寄存器高字节*

***UBRRH*– USART 波特率寄存器高字节**								
地址: 0x9A					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG9	BRG8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	R/W	描述			
7:0	BRG[15:8]	R/W	USART 波特率寄存器的高字节部分。 USART 波特率寄存器包含 SPBRG 和 SPBRGH 两部分，用来设置通信的波特率。BRG = {SPBRGH, SPBRG}			
			SYNC	BRG16	BRGH	波特率计算公式
			0	0	0	BAUD = f_{sys}/(64*(BRG+1))
			0	0	1	BAUD = f_{sys}/(16*(BRG+1))
			0	1	0	
			0	1	1	BAUD = f_{sys}/(4*(BRG+1))
			1	0	X	
			1	0	X	

PIR1 – 外设中断标记寄存器

PIR1 – 外设中断标记寄存器								
地址：0x0C					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2IF	ADIF	RCIF	TXIF	SPIF	ECP1IF	T2IF	T1IF
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述
5	RCIF	接收结束标志位 当 RCIF 的值为“1”时，表明接收缓冲器中有未读出的数据。当 RCIF 的值为“0”时，表明接收缓冲器中没有未读出的数据。接收器禁止时，接收缓冲器被刷新，导致 RCIF 被清零。
4	TXIF	发送结束标志位 发送移位寄存器中的数据被送出，且发送缓冲器为空时 TXIF 被置位。软件通过对 TXIF 位写“0”来进行清零。

TRISB – PORTB 三态控制寄存器

TRISB – PORTB 三态控制寄存器								
地址：0x86					默认值：1111_1111			
Bit	7	6	5	4	3	2	1	0
Name	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

Bit	Name	描述
4	TRISB4	RB4 三态控制 0：RB4 引脚被配置为输出，USART 同步主时钟 UCK 可由此输出

		1：RB4 引脚被配置为输入，USART 同步从时钟 UCK 可由此输入
5	TRISB5	RB5 三态控制 0：RB5 引脚被配置为输出，USART 发送信号 UTX 可由此输出
6	TRISB6	RB5 三态控制 1：RB6 引脚被配置为输入，USART 接收信号 URX 可由此输入

同步外设接口 (SPI)

- 全双工，三线同步数据传输
- 主机或从机操作
- 最低位或最高位优先传输
- 7 种可编程的比特率
- 发送结束标志位
- 写入冲突标志保护机制
- 可从闲置模式唤醒
- 主机操作时具有倍速模式
- 支持主机双线输入模式
- 输入输出均有 4 个缓冲寄存器

综述

SPI 主要包括三个部分：时钟预分频器，时钟检测器，从机选择检测器，发送器和接收器。控制和状态寄存器由这三个部分共享。时钟预分频器只工作在主机操作模式下，由比特率控制位来选择分频系数，从而产生相应的分频时钟，输出到 SPCK 引脚上。时钟检测器只工作在从机操作模式下，检测从 SPCK 引脚上输入的时钟沿，根据 SPI 的数据传输模式对发送和接收移位寄存器进行移位操作。从机选择检测器对从机选择信号 SPSS 进行检测，得到传输的状态来控制发送器和接收器的操作。发送器由一个移位寄存器，四个发送缓冲器和发送控制逻辑组成。接收器由一个移位寄存器，四个接收缓冲器和接收控制逻辑组成。

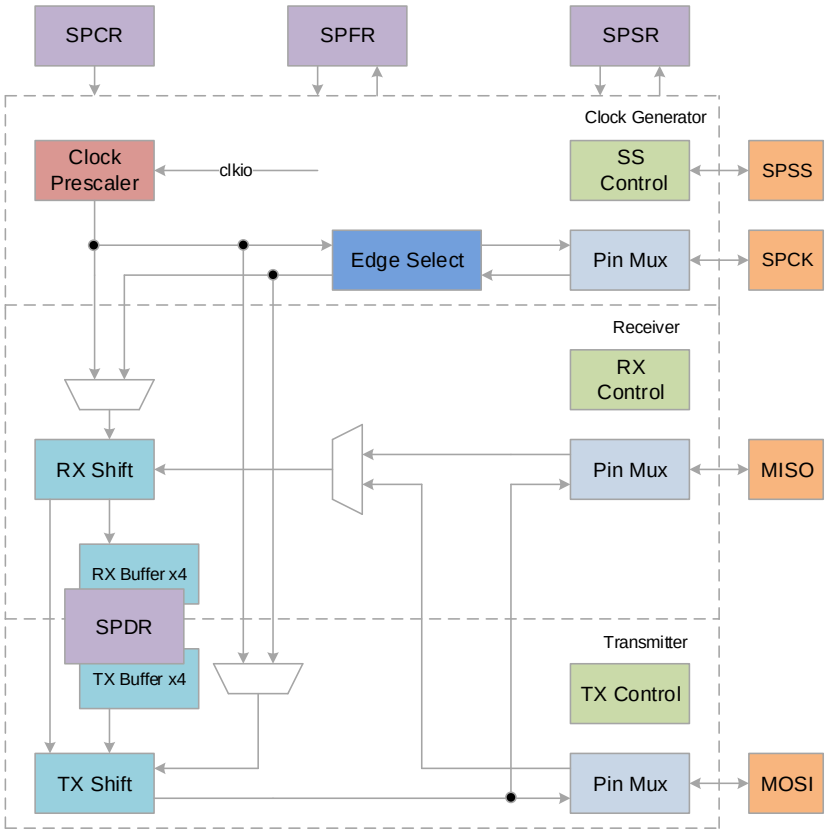


Figure 9 SPI 结构图

时钟产生

时钟产生逻辑分为主机时钟预分频器和从机时钟检测器，分别工作在主机操作和从机操作模式下。时钟预分频器由比特率控制位和倍速控制位来选择分频系数，产生相应的分频时钟（共有 7 种可选的分频系数，详细信息见寄存器描述），输出到 SPCK 引脚为通信提供时钟，同时为内部发送和接收移位寄存器提供移位时钟。时钟检测器对输入时钟 SPCK 进行边沿检测，根据 SPI 的数据传输模式对发送器和接收器进行移位操作。为保证对时钟信号的正确采样，SPCK 时钟的高电平和低电平的宽度均须大于 2 个系统时钟周期。

发送和接收

SPI 模块在单线模式下支持同时发送和接收，在双线模式下只支持主机双线接收。

单线发送和接收

SPI 的主机将需要通信的从机选择信号 SPSS 拉低，即可启动一次传输过程。主机和从机将需要传输的数据准备好，主机在时钟信号 SPCK 上产生时钟脉冲以交换数据，主机的数据从 MOSI 移出，从 MISO 移入，从机的数据从 MISO 移出，从 MOSI 移入，交换完数据后主机拉高 SPSS 信号即可完成通信。

当配置为主机时，SPI 模块并不控制 SPSS 引脚，必须由用户软件来处理。软件拉低 SPSS 引脚，选择要通信的从机，启动传输。软件将需要传输的数据写入 SPDR 寄存器即会启动时钟发生器，硬件产生通信的时钟，并把 8 位数据移出给从机，同时把从机的数据移入。SPI 模块在发送方向有四个缓冲器，当写入缓冲器中的数据均发送完毕后，硬件会停止时钟发生器，并置位传输完成标志 SPIF。软件可以拉高 SPSS 信号来结束当前传输。

当配置为从机时，只要 SPSS 信号一直为高，SPI 模块将保持睡眠状态，并保持 MISO 引脚为三态。这时软件可更新 SPDR 寄存器的内容。即使此时 SPCK 引脚上有输入时钟脉冲，SPDR 的数据也不会被移出，直至 SPSS 信号被拉低。SPI 在接收方向有四个缓冲器，当接收到四个字节的数据而未被软件及时读走，硬件置位传输完成标志 SPIF。此时软件在读取移入的数据之前可继续往 SPDR 寄存器写入数据，最后进来的数据将保存在接收缓冲器中。

SPI 模块在发送方向只有四个缓冲器，在接收方向也有四个缓冲器。在发送数据时，当发送缓冲器处于非满状态（即发送缓冲器满标志位 WRFULL 位为低）时，可对 SPDR 寄存器进行写操作。而在接收数据时，当接收缓冲器属于非空状态（即接收缓冲器空标志位 RDEMPT 位为低）时，可通过访问 SPDR 寄存器读取已经接收到的字符。

主机双线接收

SPI 模块的双线模式只在主机操作模式下有效，与单线模式的不同在于 MOSI 和 MISO 都用于主机接收数据，每一个 SPCK 时钟脉冲同时接收 2 个比特的数据（MISO 线上的数据在前，MOSI 线上的数据在后），接收完两个字节的数据之后硬件置位传输完成标志 SPIF，数据保存到接收缓冲器和移位寄存器中。此时软件须读取 SPDR 寄存器两次来得到所接收的两个字节的数据。需要注意的是，虽然双线模式下主机不向从机发送数据，软件仍需要往 SPDR 寄存器写入数据来启动时钟发生器产生通信时钟，写入一次 SPDR 寄存器即可接收两个字节的数据。

数据模式

单线模式下，相对于串行数据，SPI 有 4 种 SPCK 相位和极性的组合方式，由 CPHA 和 CPOL 来控制，如下表所示。

Table 4 CPHA 和 CPOL 选择数据传输模式

CPOL	CPHA	起始沿	结束沿	SPI 模式
0	0	采样（上升沿）	设置（下降沿）	0
0	1	设置（上升沿）	采样（下降沿）	1
1	0	采样（下降沿）	设置（上升沿）	2
1	1	设置（下降沿）	采样（上升沿）	3

当 CPHA = 0 时，数据采样和设置的时钟沿如下图所示：

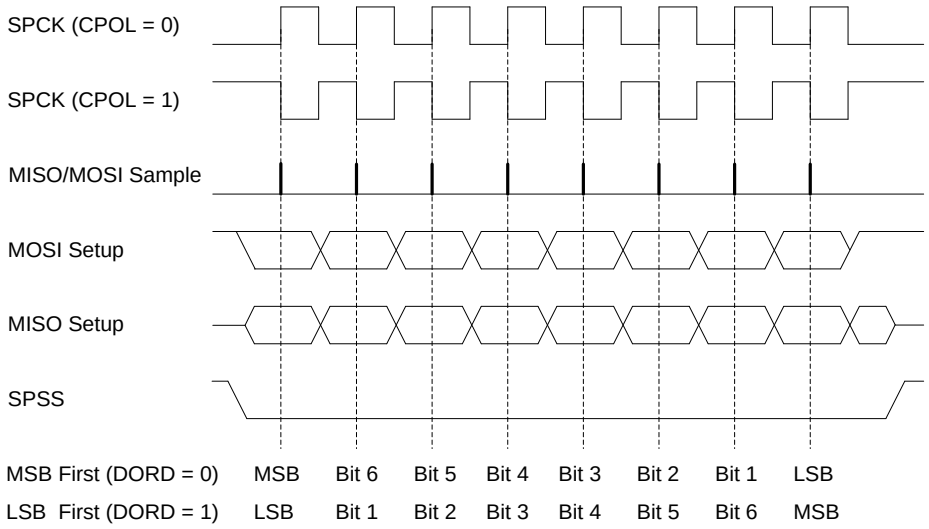


Figure 10 CPHA 为“0”时 SPI 数据传输模式

当 CPHA = 1 时，数据采样和设置的时钟沿如下图所示：

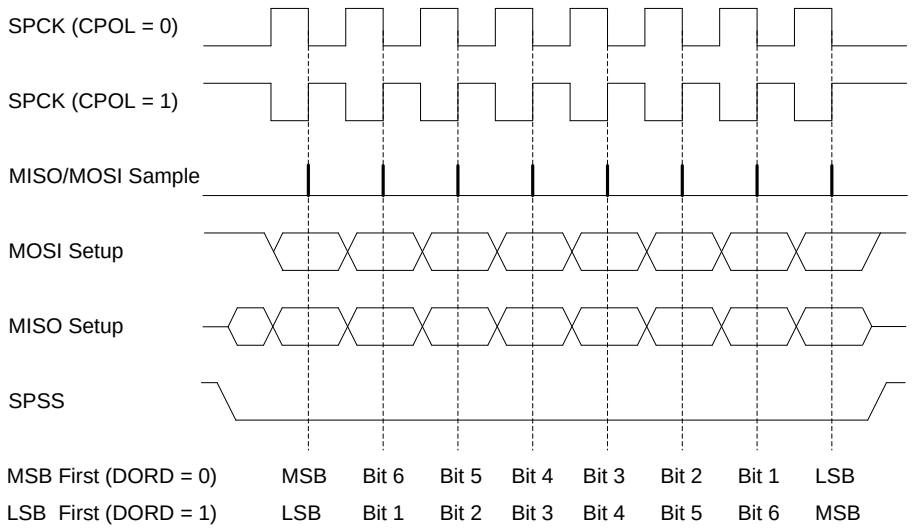


Figure 11 CPHA 为“1”时 SPI 数据传输模式

双线模式下，MISO 和 MISO 均用做主机的输入，数据采样的时刻仍由数据传输模式决定，采样的方式如下图所示：

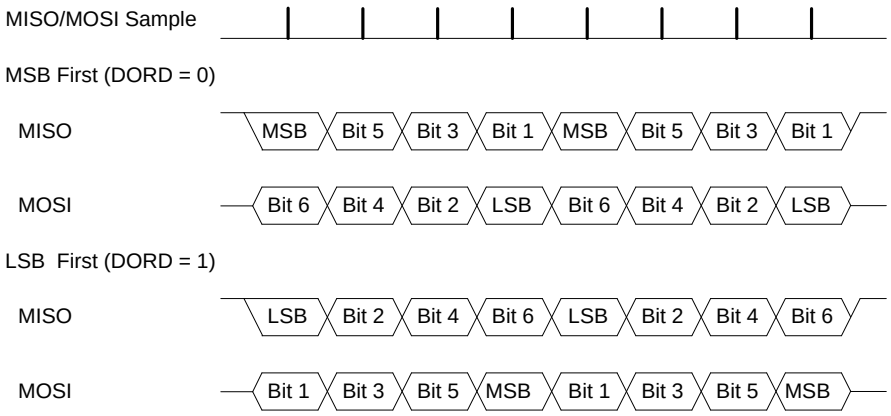


Figure 12 主机模式下 DUAL 为“1”时 SPI 数据采样模式

SPSS 引脚功能

当配置为从机时，从机选择信号 SPSS 引脚总是作为输入。当 SPSS 引脚保持为低时，SPI 接口被激活，MISO 引脚成为输出引脚（软件进行相应的端口配置），其它引脚均为输入。当 SPSS 引脚保持为高时，SPI 模块被复位，且不再接收数据。SPSS 引脚对于数据包/字节的同步非常有用，可以使从机的位计数器和主机的时钟发生器同步。当 SPSS 拉高时，SPI 从机立即复位接收和发送逻辑，并丢弃移位寄存器里不完整的数据。

当配置为主机时，用户软件可以决定 SPSS 引脚的方向。

若 SPSS 配置为输出，则它可以用来驱动从机的 SPSS 引脚。若 SPSS 配置为输入，必须保持为高以保证主机的正常工作。当配置为主机且 SPSS 引脚为输入，外部电路拉低 SPSS 引脚时，SPI 模块会认为是另外一个主机选择自己作为从机并开始传输数据。为了防止总线冲突，SPI 模块将进行如下动作：

1. 清零位于 SPCR 寄存器的 MSTR 位，转换为从机，从而 MOSI 和 SPCK 变为输入；
2. 置位位于 SPSR 寄存器的 SPIF 位；

因此，使用中断方式处理 SPI 主机的数据传输，并且存在 SPSS 被拉低的可能性时，中断服务程序应该检查 MSTR 位是否为“1”。若被清零，软件须将其置位，以重新使能 SPI 主机模式。

SPI 初始化

进行通信之前首先要对 SPI 进行初始化。初始化过程通常包括主机从机操作的选择，数据传输模式的设定，比特率的选择，以及各个引脚的方向控制等。其中主机和从机操作下引脚方向的控制各不相同，如下表所示：

Table 5 引脚方向控制

引脚	主机模式下的方向	从机模式下的方向
MOSI	用户软件定义	输入
MISO	输入	用户软件定义
SPCK	用户软件定义	输入
SPSS	用户软件定义	输入

SPI 主机初始化

SPI 主机模式的初始化过程如下：

1. 置位 MSTR 位，设置比特率选择控制位，数据传输模式，数据传输次序，中断使能与否，以及双线使能与否；
2. 设置 MOSI 和 SPCK 引脚为输出；
3. 置位 SPE 位。

主机模式下，当不希望 SPI 模块被别的主机选择作为从机使用时，可设置 SPSS 引脚为输出。

SPI 从机初始化

SPI 从机模式初始化过程如下：

1. 清零 MSTR 位，设置数据传输模式，数据传输次序，中断使能与否；
2. 设置 MISO 引脚为输出；
3. 置位 SPE 位。

寄存器定义

Table 6 SPI 寄存器列表

寄存器	地址	默认值	描述
SPCR	0x014	0x00	SPI 控制寄存器
SPSR	0x094	0x00	SPI 状态寄存器
SPDR	0x013	0x00	SPI 数据寄存器
SPFR	0x093	0x00	SPI 缓冲寄存器

SPCR – SPI 控制寄存器

SPCR – SPI 控制寄存器									
地址: 0x14					默认值: 0x00				
Bit	7	6	5	4	3	2	1	0	
Name	-	SPIEN	SPIDORD	SPIMST	SPICPOL	SPICPHA	SPISPR1	SPISPR0	
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Bit	Name	R/W	描述
7	-	-	保留。
6	SPIEN	R/W	SPI 使能位。 当设置 SPE 位为“1”时，SPI 模块被使能。进行任何 SPI 操作之前必须置位 SPE。 当设置 SPE 位为“0”时，SPI 模块被禁止。
5	SPIDORD	R/W	数据次序控制位。 当设置 DORD 位为“1”时，数据的 LSB 首先发送。 当设置 DORD 位为“0”时，数据的 MSB 首先发送。
4	SPIMST	R/W	主机从机选择控制位。 当设置 MSTR 位为“1”时，选择为主机模式。 当设置 MSTR 位为“0”时，选择为从机模式。 主机模式下，SPSS 引脚配置为输入且被拉低时，MSTR 位将被清零，位于 SPSR 寄存器的 SPIF 被置位，用户必须重新设置 MSTR 进入主机模式。
3	SPICPOL	R/W	时钟极性控制位。 当设置 CPOL 位为“1”时，空闲状态下 SPCK 为高电平。

			当设置 CPOL 位为“0”时，空闲状态下 SPCK 为低电平。
			CPOL 起始沿 结束沿
			0 上升沿 下降沿
			1 下降沿 上升沿
2	SPICPHA	R/W	时钟相位控制位。 当设置 CPHA 位为“1”时，起始沿设置数据，结束沿采样数据。 当设置 CPHA 位为“0”时，起始沿采样数据，结束沿设置数据。
			CPHA 起始沿 结束沿
			0 采样 设置
			1 设置 采样
1	SPISPR1	R/W	时钟速率选择位 1。 SPR1 和 SPR0 用来选择 SPI 传输的时钟速率。具体控制方式见 SPCK 和系统时钟的关系表格。
0	SPISPR0	R/W	时钟速率选择位 0。 SPR1 和 SPR0 用来选择 SPI 传输的时钟速率。具体控制方式见 SPCK 和系统时钟的关系表格。

SPSR – *SPI 状态寄存器*

SPSR – SPI 状态寄存器								
地址: 0x94					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	SPIIF	SPIWC	SPISSM	SPISSI	-	SPIDM	-	SPI2X
R/W	R	R	W/R	W/R	R	R/W	R	R/W

Bit	Name	R/W	描述
7	SPIIF	R	等同于 SPI 中断标志位 SPIF（位于 PIR1），只能读取。
6	SPIWC	R	写冲突标志位。 在数据传输的过程中写 SPDR 寄存器将置位 WCOL 位。WCOL 位可以通过先读取 SPSR 寄存器再访问 SPDR 寄存器来清零。
5	SPISSM	R/W	SPSS 模式选择位。 当设置 SPISSM 位为“1”时，输入的从设备选择控制来自内部 SPISSI 位。 当设置 SPISSM 位为“0”时，输入的从设备选择控制来自外部 SPSS 引脚。
4	SPISSI	R/W	内部 SPSS 控制。 只有在 SPISSM 位被置位的情况下才有效。 当设置 SPISSI 位为“1”时，此从设备未被选择。 当设置 SPISSI 位为“0”时，此从设备被选择。
3	-	R	保留。
2	SPIDM	R/W	双线模式控制位。 当设置 DUAL 位为“1”时，使能 SPI 双线传输模式。 当设置 DUAL 位为“0”时，禁止 SPI 双线传输模式。 双线传输模式只在 SPI 主机模式下有效，MISO 和 MOSI 均用作主机数据输入，数据的传输方式见主机双线接收和数据模式章节描述。
1	-	R	保留。
0	SPI2X	R/W	SPI 倍速控制位。

			当设置 SPI2X 位为“1”时，SPI 的传输速度加倍。 当设置 SPI2X 位为“0”时，SPI 的传输速度不加倍。 具体控制方式见 SPCK 和系统时钟的关系表格。

下表为 SPCK 和系统时钟的关系。

Table 7 SPCK 和系统时钟的关系

SPI2X	SPR1	SPR0	SPCK 的频率
0	0	0	$f_{sys}/4$
0	0	1	$f_{sys}/16$
0	1	0	$f_{sys}/64$
0	1	1	$f_{sys}/128$
1	0	0	$f_{sys}/2$
1	0	1	$f_{sys}/8$
1	1	0	$f_{sys}/32$
1	1	1	$f_{sys}/64$

SPDR – *SPI 数据寄存器*

***SPDR*– SPI 数据寄存器**								
地址: 0x13					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	SPDR7	SPDR6	SPDR5	SPDR4	SPDR3	SPDR2	SPDR1	SPDR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	R/W	描述
7:0	SPDR	R/W	SPI 发送和接收的数据。 SPI 发送数据和接收数据共享 SPI 数据寄存器 SPDR。将数据写入 SPDR 即写入发送数据移位寄存器器，从 SPDR 读取数据即读取接收数据缓冲器。

SPFR – *SPI 缓冲寄存器*

***SPFR*– SPI 缓冲寄存器**								
地址: 0x93					默认值: 0x00			
Bit	7	6	5	4	3	2	1	0
Name	RDFULI	RDEMPT	RDPTR1	RDPTR0	WRFULL	WREMPT	WRPTR1	WRPTR0
R/W	R	R/W	R	R	R	R/W	R	R

Bit	Name	R/W	描述
7	RDFULL	R	接收缓冲器满标志位。 当接收缓冲器中的数据达到四个字节时，RDFULL 位为高，表明接收缓冲器为满，同时会置位中断标志位。 当接收缓冲器中的数据少于四个字节时，RDFULL 位为低，表明接收缓冲器为非满。
6	RDEMPT	W/R	接收缓冲器空标志位。 当未接收到数据时，RDEMPT 位为高，表明接收缓冲器为空。 当有接收到数据时，会存入接收缓冲器，RDEMPT 位为低，表明接收缓冲器为非空，此时 MCU 可通过访问 SPDR 寄存器来读取接收缓冲器中的数据。 当对 RDEMPT 位进行置位操作（写 1）时，接收缓冲器地址将归零。

			当同时对 RDEMPT 位和 WREMPT 位进行置位操作时，接收缓冲器地址和接收移位寄存器指针都将归零。
5	RDPTR1	R	接收缓冲器地址高位。
4	RDPTR0	R	接收缓冲器地址低位。 当对 SPDR 寄存器进行读操作时，MCU 将会从接收缓冲器中读到所接收的数据，同时接收缓冲器地址会进行累加。 当对 RDEMPT 位进行置位操作（写 1）时，接收缓冲器地址将归零。
3	WRFULL	R	发送缓冲器满标志位。 当发送缓冲器中的数据达到四个字节时，WRFULL 位为高，表明发送缓冲器为满，同时会置位中断标志位 SPIF。 当发送缓冲器中的数据少于四个字节时，WRFULL 位为低，表明发送缓冲器为非满。
2	WREMPT	R/W	发送缓冲器空标志位。 当未对 SPDR 寄存器进行写操作，或者写入发送缓冲器的数据均已发送完毕时，WREMPT 位为高，表明发送缓冲器为空。 当对 SPDR 寄存器进行写操作后，发送缓冲器地址会累加，写入发送缓冲器的数据未被全部发送时，接收缓冲器中至少有一个字节的数据，WREMPT 位为低，表明发送缓冲器非空。 当对 WREMPT 位进行置位操作（写 1）时，发送缓冲器地址将归零。
1	WRPTR1	R	发送缓冲器地址高位。
0	WRPTR0	R	发送缓冲器地址低位。 当对 SPDR 寄存器进行写操作时，SPDR 中的数据将会被写入发送缓冲器，同时发送缓冲器地址会进行累加。 当对 WREMPT 位进行置位操作（写 1）时，发送缓冲器地址将归零。

PIR1 – 外设中断标记寄存器

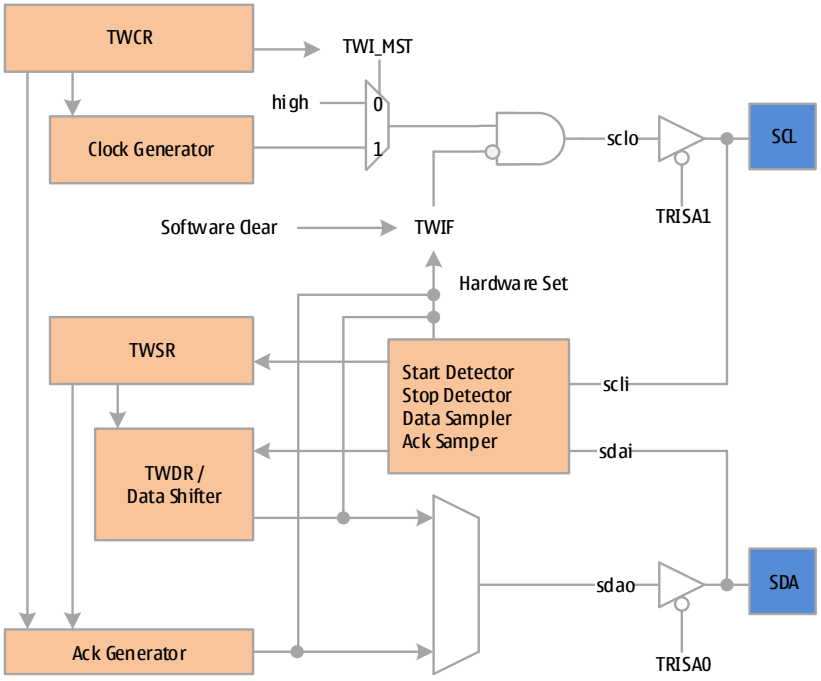
PIR1 – 外设中断标记寄存器								
地址：0x0C					默认值：0000_0000			
Bit	7	6	5	4	3	2	1	0
Name	ECP2IF	ADIF	RXIF	TXIF	SPIF	ECP1IF	T2IF	T1IF
R/W	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R
Bit	Name	描述						
3	SPIF	SPI 传输完成标志位。 串行传输结束后置位 SPIF 标志，主机模式下，配置 SPSS 引脚为输入且被拉低时，SPIF 也将被置位。软件可对该位写 0 来清零，或者通过先读取 SPSR 寄存器再访问 SPDR 寄存器来清零 SPIF 位。						

I2C 主从控制器（I2C）

- 支持主机/从机模式
- 支持传输速率设置

综述

TWI 控制器是一种双线通讯接口控制器。协议实现兼容 I2C 总线。可用于与其他使用 I2C 接口的外设通讯。



TWI 结构框图

工作模式

如需要 TWI 控制器工作为从机模式，需要置位 TWCR 寄存器的 TWEN。如需要工作为 I2C 主机模式，需要同时设置 TWEN 位与 TWMST 位，然后通过 TWCR 寄存器的 CKPS 位设置产生通讯所需的 SCL 时钟信号。

当接收到一个开始条件，重启位，一个停止位，8 位数据或一个响应位，控制器将会置位 TWSR 寄存器的对应状态信息位。软件可以通过读取 TWSR 寄存器获得接收信息，更新状态以及相关操作，然后通过写 0 清除相关的标记位。

当需要发生一个 8 位数据，响应位或者主机发生一个开始或者停止条件，软件需要设置 TWSR 寄存器中相应的标志位，之后控制器会做出相应的动作并清除标志位。当主机发送一个开始位或者停止位时，控制位也会同时接收到这个开始或者停止位。

当接收到一个开始位或一个停止位或 8 位数据或 1 位响应或者是发送一个响应位，控制器会同时置位 PIR1 寄存器中的 TWIF 中断标志位。TWIF 被置位后，控制器将 SCL 保持为低电平直到 TWIF 会被清零。

操作模式

TWI 控制可工作于 4 种主要模式。这些模式包括：主机发送(MT)，主机接收(MR)，从机发送送(ST)，从机接收(SR)。下面的章节将会分别描述这些模式。在下面的图示中，"Software"表示软件相关操作；"Hardware"表示硬件相关操作；"Master"表示主机操作；"Slave"表示从机操作；"SLA"表示从机地址；"W"表示写操作(SDA 低电平)；"R"表示读操作(SDA 高电平)。"ACK"表示响应位；"NACK"表示无响应。

主机发送模式

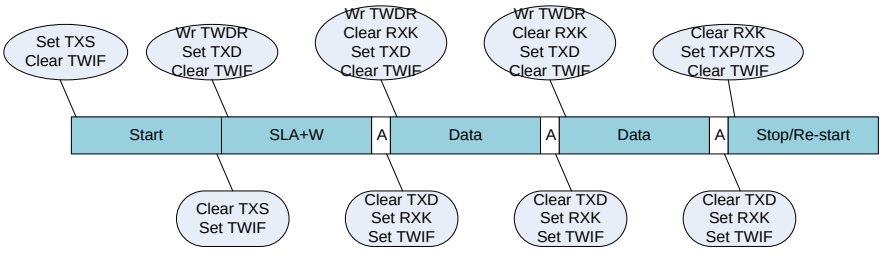
如需工作在主机发送模式，首先，TWI 需要配置为主机工作模式。

通过置位 TWSR 寄存器的 TXS 位并清零 TWIF 中断状态，控制器将会产生在总线空闲时产生一个传输开始条件(START)。开始条件传输完成后，TWIF 状态位被置位，同时 TXS 位被清零。

接下来， 需要发送一个 SLA+W。这个操作可以通过向 TWDR 寄存器中写入 SLA+W 完成，设置 TWSR 寄存器的 TXD 位，清零 TWIF。当 SLA+W 发送完成并接收到一个响应位后，TWIF 位再次被置位，TXD 位清零，RXK 位设置以表明当前已收到正确的响应信息。

SLA+W 成功传输后，接下来可以发送数据。将需要发送的数据写入 TWDR 寄存器，清零 RXK 位，置位 TXD 并清零 TWIF。当数据发送完成并接收到正确的响应信息后，TWIF 位被重新置位，TXD 被清零，RXK 被置位。

接下来重复数据发送流程，直到最后一个字节的数据发送完成。然后发送一个停止位停止数据传输或者重复一个开始位，继续传输其他的数据。停止位通过写 TWSR 寄存器的 TXP 位，清零 RXK 位与 TWIF 位实现。重复开始条件可以通过置位 TXS 位，清零 RXK 以及 TWIF 位实现。停止位发送结束后，TXP 位被清零，但 TWIF 此时不会被置位。当重复开始发送结束后，TXS 被清零，同时 TWIF 标志位被置位。



主机发送模式数据流图

主机接收模式

首先，需要将 TWI 配置为 I2C 主机模式。

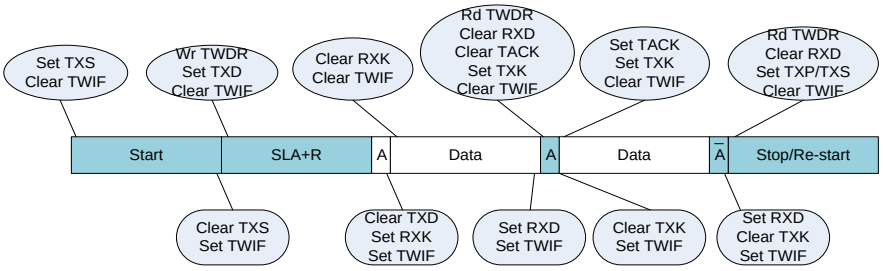
通过置位 TWSR 寄存器的 TXS 位，清零 TWIF 标志位，启动传输流程。TWI 控制器将会在总线空闲后发起一个开始位。开始位发送完成后，TWIF 标志位置位，同时 TXS 位清零。

进入 MR 模式前，需要先发送 SLA+R。将 SLA+R 写入 TWDR 寄存器，置位 TXD，清零 TWIF 将会启动 SLA+R 传输。在 SLA+R 传输完成并成功接收到响应信号后，TWIF 被再次置位，TXD 位被清零，同时 RXK 位被置位。

SLA+R 成功传输后，控制器可以进入数据接收状态。清零 RXK 以及 TWIF 标志位，控制器将会在 SCL 产生 8 个时钟周期脉冲，并接收完成一个自己的数据。数据接收完成后，TWSR 寄存器的 RXD 被置位。接收的数据可以通过读取 TWDR 寄存器获得。读完数据后需清零 RXD 标志位。数据接收后，需发送一个响应给从机。置位 TXK 标志位并清零 TWIF 标志位，控制器发送的是响应(ACK)还是非响应(NACK)取决于 TWCR 寄存器的 TACK 位。如果 TACK 为 0，主机将返回一个 ACK，否则返回一个 NACK。响应信号发送完成后，TXK 位被清零，同时 TWIF 位被再次置位。

数据接收过程将会重复直到接收到最后一个字节。数据接收的过程也可以被进一步简化。简化接收流程的操作通过预先设定响应位并置位 TXK 位实现。TXK 置位后，控制器在接收到一个字节的数据后，并不会触发 TWIF 标志位。待到响应位成功发送后，TWIF 置位。接收的数据可以从 TWDR 中读取。

传输由一个停止位或者一个重复开始位结束。通过置位 TXP 位并清零 RXK 位以及 TWIF 标志位，可以向总线发送一个停止位。重复开始条件可以通过置位 TXS 位，清零 RXK 位以及 TWIF 标志位实现。停止位发送后，TXP 位清零，TWIF 标志位将不会被置位。当重复开启位发送后，TXS 位被清零，同时 TWIF 位被置位。



Master Received Mode Data Flow Diagram

从机发送模式

首先，需要设置 TWI 控制为 I2C 从机工作模式.

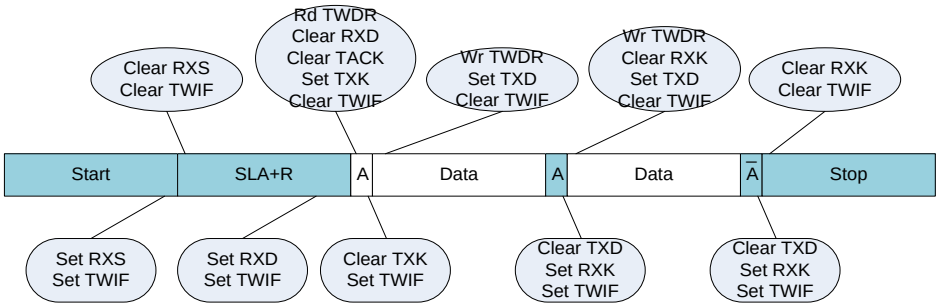
将控制器在总线上检测到一个传输开始位，TWSR 寄存器的 RXS 位被置位，同时 TWIF 位也被置位。这种状态说明总线此时已被其他的主机占用。软件需要清零 RXS 位以及 TWIF 位准备接收接下来主机发送的地址以及读写控制位。

当接收到 SLA+W/R 字节后，RXD 标志位被置位，同时 TWIF 位也被置位。软件响应 TWIF 中断，读取 SLA+W/R 数据并根据其中的地址信息判断是否被寻址。如果本机是访问目标，清零 TACK 标志位，否则置位 TACK 标志位。如果本机被寻址，软件需要更加 W/R 控制位确定接下来的操作。如果是主机发起的读操作，控制器将会进入到从机发送模式，软件可以通过清零 RXD，设置 TXK 标志位以及清零 TWIF 标志位发送响应信息。

响应位发送后，TXK 位被清零，TWIF 位被置位。此后，可以发送数据部分。将待发送的数据写入 TWDR 寄存器，置位 TXD 标志位并清零 TWIF 标志位，启动数据发送。当数据被成功发送并接收到来自主机的响应位后，TXD 位被清零，RXK 以及 TWIF 位再次被置位。

数据传输流程将会反复循环直到发送完成最后一个数据字节或接收到 NACK，接收到一个停止位，或者接收

到一个重复开始条件。



从机发送模式数据流程图

从机接收模式

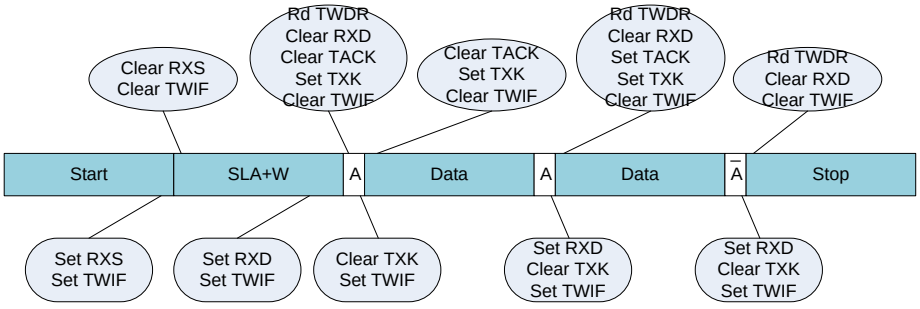
进入到从机接收模式前，首先需要将 TWI 控制配置为 I2C 从机模式.

当控制器在总线上检测到开始传输条件，RXS 位置位，同时 TWIF 也被置位。此时的状态说明总线已被其他主机占用。 软件需清零 RXS 以及 TWIF 位以接收从机地址控制数据。

SLA+W/R 字节接收完成后，RXD 位以及 TWIF 位同时被置位。软件通过 TWDR 寄存器获得接收的数据并根据接收的地址信息判断是否被寻址。如果被主机寻址，软件需要清零 TACK 位，否则置位 TACK。被寻址的控制器还需要根据读写控制位确定接下来需要进入的工作模式。如果是接收到一个写操作，控制器进入从机接收模式。控制器通过清零 RXD 位，置位 TXK 位以及清零 TWIF 位向主机发送响应信息。

响应信息发送完成后，TXK 被清零，TWIF 再次置位。此时可以接收数据。与主机接收模式相同，从机的数据接收流程也可以被简化。通过预定义响应位(TACK 位)，设置 TXK 位以及清零 TWIF 位，可使控制器工作于简化数据接收模式。控制器接收完一个字节的数据并发送完响应信息后，TXK 位清零，TWIF 位被置位。接收的数据被保存到 TWDR 寄存器。

同样，数据接收流程将会被循环直到接收完最后一个字节并完成 NACK 的发送。



从机接收模式流程图

寄存器定义

TWCR – TWI 控制寄存器

TWCR		地址:0x9E	初始值:0x08
Bit	Name	描述	
7	TWEN	TWI 使能控制位，高有效。	
6	TWMST	TWI 主机模式控制位： 1 = 主机模式 0 = 从机模式	
5:4	-	-	
3	TACK	发送总线响应控制位 0 = ACK 1 = NACK	
2	RACK	接收总线响应标志位 0 = ACK 1 = NACK	
1:0	CKPS	I2C 总线时钟分频控制： 00: F_{SCL} = Fsys/10 01: F_{SCL} = Fsys/20 10: F_{SCL} = Fsys/40 11: F_{SCL} = Fsys/64	

TWSR – TWI 状态存器

TWSR		地址:0x98	初始值:0x00
Bit	Name	描述	
7	TXP	软件写 1 发送停止位 硬件清零，表示已接收到停止位	
6	RXK	接收到总线响应。硬件位置，软件清零	
5	TXD	软件写 1 发送数据 硬件清零，表示数据发送完成	
4	TXS	软件写 1 发送开始位 硬件清零，表示开始位已发送完成	
3	RXP	接收到停止位。硬件置位，软件清零	
2	TXK	软件写 1 发送响应位 硬件清零，表示停止位已发送完成	
1	RXD	数据接收完成。硬件置位，软件清零	
0	RXS	接收到开始位。硬件置位，软件清零	

TWDR – TWI 数据接收/发送寄存器

TWDR		地址:0x8D	初始值:0x00
Bit	Name	描述	
7:0	TWDR	TWI 接收/发送数据寄存器	

EEP 控制器(EEP)

- 128 字节 EEPROM 读写访问
- 程序储存器读写访问

概述

LGT8F690 集成 128 字节，至少 10K 次擦写周期的 EEPROM 数据存储器。可实现基于字节的数据读写访问。芯片内部 4Kx14bit 程序存储空间也支持在正常工作电压下的编程操作，可用于实现在线固件升级(IAP)。

通过 EEP 控制器访问 EEPROM 数据存储空间时，EEDAT 寄存器存储读写操作的目标数据，目标数据所在的地址由 EEAR 寄存器给出。高位地址寄存器 EEARH 必须清零。

通过 EEP 控制器访问程序存储器是， EEDAT/EEDATH 组合为一个 16 位寄存器，提供读写操作的数据寄存器。访问的目标地址由 EEARH/EEAR 组合的 16 位地址寄存器提供。

需要特别注意， 通过 EEP 控制写程序空间， 可通过 EEDATH/EEDAT 写入完整的 14 位数据。但通过 EEP 控制器读程序空间时，只能读出低字节数据(EEDAT)。

编程操作对系统工作电压有一定的要求。通过 EEP 控制器写 EEPROM 数据空间，VCC 只需要大于 2.5V 即可。通过 EEP 控制器写程序存储空间，系统工作电压 VCC 需要大于 3.3V，并且建议 VCC 在 4V 以上。

EEP 控制器可直接执行对 EEPROM 或者程序空间的写操作，无需单独执行擦除操作。

EEP 时钟控制

通过 EEP 控制器写 EEPROM 或者程序空间时， 需要系统额外提供一个写时钟。LGT8F690A 使用内部 16MHz HFINTOSC 的分频时钟实现。当系统工作时钟同时也为内部 16MHZ HFINTOSC 时，无需特别的操作。当 EEP 启动写操作时，会同时使能 EEP 写时钟。

但如果系统工作时钟非 HFINTOSC 时， HFINTOSC 处于关闭状态。在这种情况下使用 EEP 写 EEPROM 或者程序空间时， 需要首先通过 EECON1 寄存器的 EECKE 位使能 HFINTOSC 时钟。

EEPROM 读写访问

EEP 通过 EECON1 寄存器的 EEPGD 位选择当前操作的目标。当 EEPGD 为 1，EEP 可访问程序空间；清零 EEPGD 位， 访问 EEPROM 数据空间。

EEPGD 设置完成后，通过置位 EECON1 寄存器的 RD 位，启动一次读操作，MCU 内核被挂起，当数据从 EEPROM 中读出并更新到 EEDAT 寄存器中后，MCU 恢复运行，执行 RD 之后的下一条指令。

EEPROM 的写操作控制比读稍复杂，主要是增加了对写操作的时序保护，保护 EEPROM 数据安全。写操作也是首先需要通过 EEPGD 指定到 EEPROM 区域，然后通过 EEADR 以及 EEDAT 准备好写 EEPROM 的地址和数据。接下来就是写操作必须执行的时序部分：

1. 首先置位 EECON1 寄存器的 WREN 位，使能写操作；
2. 临时关闭全局中断，防止接下来对 EECON2 寄存器的操作被中断打断‘
3. 向 EECON2 寄存器中依次写 0x55 和 0xAA，解锁写保护；
4. 置位 EECON1 寄存器的 WREN 位，启动写操作；
5. 清零 WREN，关闭写使能；
6. 恢复全局中断设置

EEPROM 读写操作实例代码：

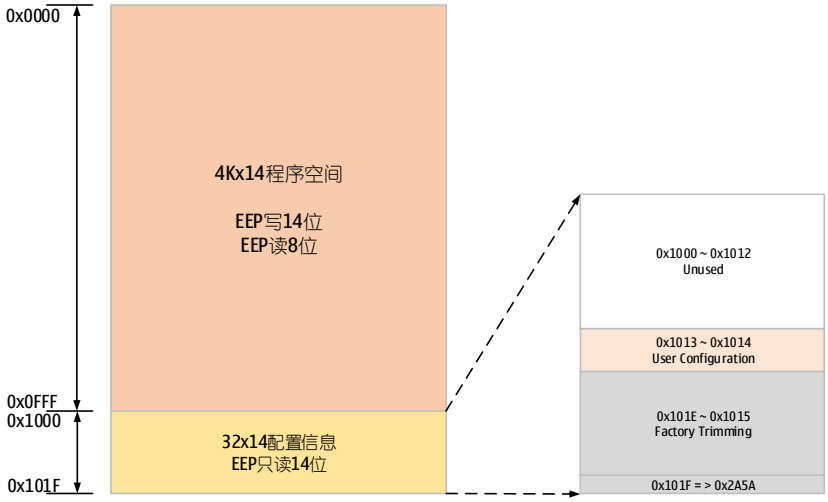
```
#include “lgt8f690a.h”

// 可选操作，仅当系统时钟非 HFINTOSC 时使用
// 请参考 EEP 时钟控制部分
void eepInit()
{
    EECKE = 1;
}

// EEPROM 读操作
uint8_t e2pRead (uint8_t address)
{
    EEADRH = 0;
    EEADR = address;
    EEPGD = 0;                          // select EEPROM
    EERD = 1;                           // start read
    return EEDAT;
}

// EEPROM 写操作
void e2pWrite(uint8_t address, uint8_t data)
{
    uint8_t btmp;

    EEPGD = 0;                          // select EEPROM
    EEADRH = 0;
    EEADR = address;
    EEDAT = data;

    WREN = 1;                           // enable write
    btmp = INTCON;
    GIE = 0;                            // disable global interrupt

    EECON2 = 0x55;
    EECON2 = 0xAA;
    EEWR = 1;                           // start programming

    WREN = 0;                           // disable write
    INTCON = btmp;                      // restore INTCON
}
```

程序空间读写访问

LGT8F690A 内部集成 4K*14 位程序存储空间。在程序空间之后， 是 32*14 位的配置信息空间。使用 EEP 控制可以实现对程序空间读写访问。配置信息空间为只读空间。

需要注意， EEP 对程序空间可以写完整的 14 位数据，但只能读低 8 位数据。对配置的访问没有限制，可以正常的读写 14 位数据。

下图为 LGT8F690A 程序空间与配置位的地址映射：



使用 EEP 读写程序空间的方法与 EEP 读写 EEPROM 空间的方法相同。区别是需要置位 EECON1 寄存器的 EEPGD 位，选择程序和配置空间。读写操作的目标地址，需要由 EEADRH/EEADR 寄存器共同给出。写 14 位数据时，数据由 EEDATH/EEDAT 给出。读配置信息数据来自 EEDATH/EEDAT；读程序空间，数据仅 EEDAT 有效。

寄存器定义

EEDAT/L – EEP 数据寄存器低字节

EEDAT/L– EEP 数据寄存器低字节								
地址: 0x10C					0x00			
Bit	7	6	5	4	3	2	1	0
	EEDAT[7:0]							
R/W	R/W	R/W	R/W	R/W	W/R	R/W	R/W	W/R
Bit	Name	描述						
7:0	EEDAT/L	EEP 数据寄存器低字节 访问程序空间或配置信息时与 EEDATH 组成 16 位的数据寄存器 访问 EEPROM 区域时的数据寄存器						

EEDATH – EEP 数据寄存器高字节

EEDATH– EEP 数据寄存器高字节

地址: 0x10E					0x00			
Bit	7	6	5	4	3	2	1	0
	EEDATH[7:0]							
R/W	R/W	R/W	R/W	R/W	W/R	R/W	R/W	W/R
Bit	Name	描述						
7:0	EEDATH	EEP 数据寄存器低字节 访问程序空间或配置信息时与 EEDAT/L 组成 16 位的数据寄存器						

EEADR/L – EEP 地址寄存器低字节

EEADR/L– EEP 地址寄存器低字节								
地址: 0x10D					0x00			
Bit	7	6	5	4	3	2	1	0
	EEADR[7:0]							
R/W	R/W	R/W	R/W	R/W	W/R	R/W	R/W	W/R
Bit	Name	描述						
7:0	EEADR/L	EEP 地址寄存器低字节 访问程序空间或配置信息时与 EEADRH 组成 16 位的数据寄存器 访问 EEPROM 空间时指定目标地址						

EEADRH – EEP 地址寄存器高字节

EEADRH– EEP 地址寄存器高字节								
地址: 0x10F					0x00			
Bit	7	6	5	4	3	2	1	0
	EEADRH[7:0]							
R/W	R/W	R/W	R/W	R/W	W/R	R/W	R/W	W/R
Bit	Name	描述						
7:0	EEADRH	EEP 地址寄存器高字节						

EECON1 – EEP 控制寄存器 1

EECON1– EEP 控制寄存器 1								
地址: 0x18C					0x00			
Bit	7	6	5	4	3	2	1	0
	EEPGD				WRERR	WREN	EEWR	EERD
R/W	R/W	-	-	-	W/R	R/W	R/W	W/R
Bit	Name	描述						
7	EEPGD	程序空间/EEPROM 空间选择位 1 = 选择 EEP 控制的目标地址为程序空间 0 = 选择 EEP 控制的目标地址为 EEPROM 空间						
6:4	-	保留不用						
3	WRERR	EEP 控制器出错标记位 此位可以配合 WDT 实现写操作的超时功能。 在开启 WDT 的后， 如果在 EEP 操作时发生超时导致 WDT 复位， WRERR 将会置位。EEADR/EEDAT 的数						

		据将会保持不变。
2	WREN	EEP 写操作使能控制位 1: 使能写操作， 使能写操作后， EEWR 操作有效 0: 禁止写操作， 禁止后， EEWR 将不能启动写操作
1	EEWR	写操作启动位， 在 WREN 为 1 时， 写 EEWR = 1 启动一次 EEP 写操作
0	EERD	读操作启动位， 写 EERD =1 ， 启动一次 EEP 读操作

EECON2 – EEP 控制寄存器 2

EECON2– EEP 控制器存器 2								
地址: 0x18D					XX			
Bit	7	6	5	4	3	2	1	0
	EECON2[7:0]							
R/W	WO	WO	WO	WO	WO	WO	WO	WO
Bit	Name	描述						
7:0	EECON2	EECON2 为只写寄存器， 用于实现 EEP 写操作的保护。 使能 WREN 后， 必须先对 EECON2 寄存器顺序写入 0x55 和 0xAA, 才能使用 EEWR 开启写操作。 否则写操作无效。						

模拟比较器（CM）

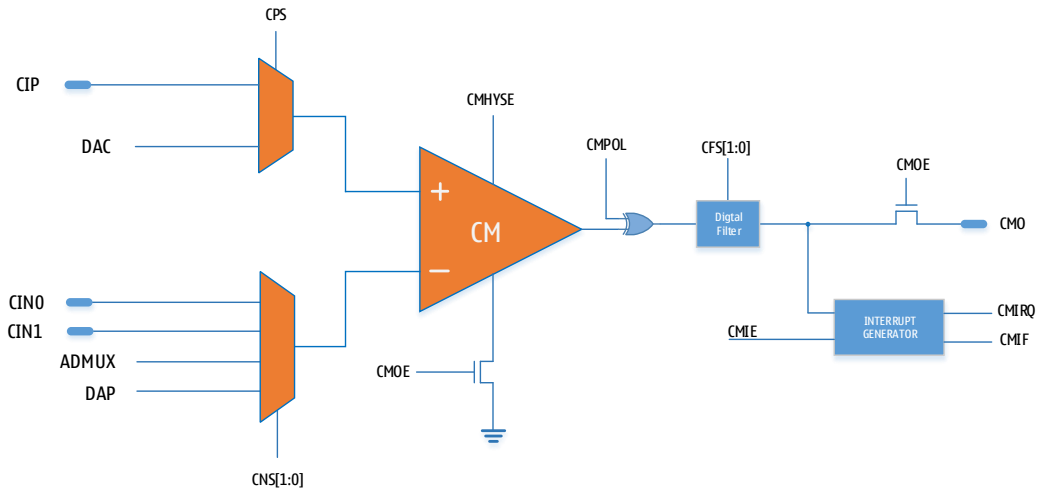
- 低于 10mV 的比较精度
- 多路内部/外部比较器输入
- 比较器输出直接到端口
- 可编程输出极性控制
- 支持休眠唤醒
- CM 输出可作为 TMR1 门控
- CM 输出与 TMR1 同步
- 输出迟滞控制
- 可选内部数字滤波

概述

模拟比较器对输入比较器正端与负极的电平进行比较，当正端电压比负端电压高时，模拟比较器的输出 COUT 被置位。当 COUT 的电平发生变化时，可触发中断。输出信号 COUT 还可用来触发定时计数器 1 的输入捕捉以及对定时器产生的 PWM 输出进行控制。

LGT8F690A 集成一个多路输入的模拟比较器，比较器正、负端输入源可以选择来自外部端口或者来自多种内部产生的参考源。模拟比较器本身支持失调校准，可以保证比较器工作的一致性。比较器支持一个可选的硬件迟滞功能，用于改善比较器输出的稳定性。同时比较器输出端集成一个硬件可以编程数字滤波器，可以根据应用需求，选择合适的滤波设置，以获得更加稳定的比较输出。

比较器输出状态可以直接通过寄存器读取，也可以产生中断请求，实现更高效的实时事件俘获功能。比较器的输出也可以直接输出到外部 IO 端口(CMO)。



比较器 CM 结构图

模拟比较器的输入

模拟比较器的两个输入端都支持多种可选输入源。正端的输入两路可选：

1. 外部独立模拟输入 CIP
2. 内部 6 位 DAC 的输出 DAC

比较器正端输入由 CMCON1 寄存器 CPS 位控制，具体请参考本章节寄存器描述部分。

DAC 来自内部 6 位 DAC 的输出。DAC 的参考源可以选择来自系统电源，内部参考或者来自外部参考的输入。DAC 的配置请参考 DAC 相关章节。

CPS	CM 正端输入
0	CIP
1	DAC

负端输入有选择 4 种不同的模拟输入:

1. 外部模拟输入 CIN0
2. 外部模拟输入 CIN1
3. 内部差分放大器 DAP 输出
4. ADC 的多路复用输出(ADMUX)

比较器负端输入通道选择由来自 CMCON1 寄存器的 CNS 位控制。

当比较器负端输入选择为 ADC 的多路复用输出时，需要通过 ADCON0 寄存器 CHS 位选择输入源。

DAP 来自内部差分放大器输出。差分放大器可选 x1/x8/x16/x32 增益，可实现小信号的测量。

CNS1	CNS0	CM 负端输入
0	0	AC0N
0	1	AC1N
1	0	DAP
1	1	ADMUX

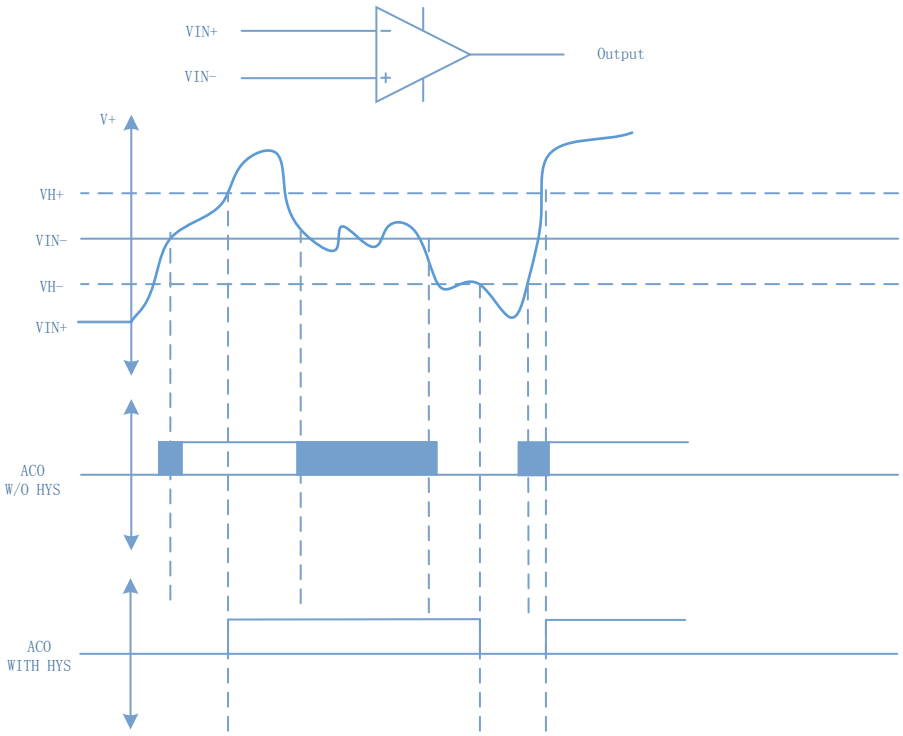
比较器输出滤波

比较器输出端内部支持一个可控的迟滞电。用户可以通过 CMCON1 寄存器的 CMHYS 位使能迟滞电路。迟滞电路可以消除比较器状态变化过程的不稳定状态，达到输出滤波功能。

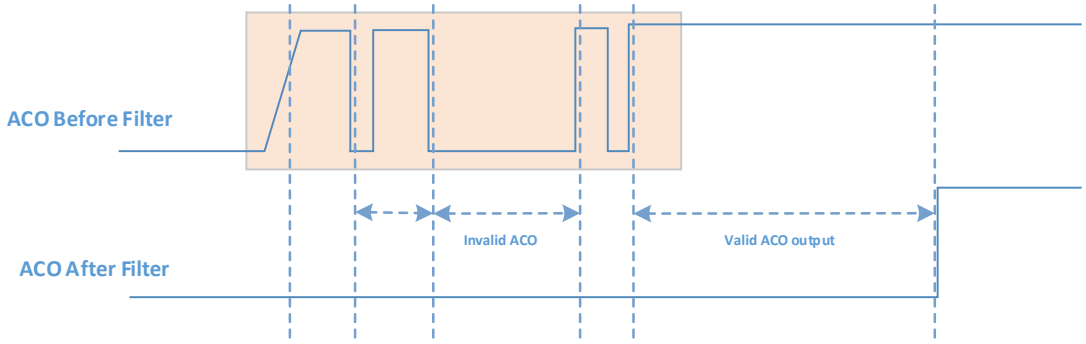
建议用户在使用比较器时，打开迟滞电路，获得一个稳定的比较器输出。

如下图所示，迟滞电路位于比较器模拟输出与数字输出之间。当比较器正端的输入电压 V_{IN+}大于(V_{IN-} + V_{H+})时，比较器 COUT 输出为高；当 V_{IN+}电压小于(V_{IN-} - V_{H-})时，比较器输出低。迟滞电路避免了当比较器正端电压接近负端电压时，电路本身带来的抖动。

比较器迟滞电压与比较器输出关系图:



尽管迟滞电路对于抑制接近比较器阀值的电压纹波非常有效，但实际应用环境中，输入信号会受到不同强度的干扰。较强的干扰可能会导致输入电平瞬间抬高，超出迟滞电路的阀值范围，无法被有效抑制。LGT8F690A 在比较器输出端集成了一个可编程的数字滤波器，可以滤除瞬时干扰对比较器输出产生的影响。数字滤波器可以根据应用需求，选择合适的滤波时间宽度，只有当比较器的输出稳定持续满足滤波时间限制，滤波电路才更新比较器的输出。从而达到一个更加稳定的输出结果。



比较器输出滤波时序

CM 的数字滤波通过 CMCON1 寄存器的 CFS 位选择控制，具体设置方式请参考本章寄存器定义部分。

比较器输出与 PWM 控制

LGT8F690A 支持多通道 PWM 输出，PWM 信号可以与比较器模块配合使用。比较器的输出，可用于直接关断 PWM 信号，从而实现比较灵活的 PWM 保护方案。

与 PWM 输出相关的控制，请参考定时器章节的相关部分。

寄存器定义

CMCON0 – 比较器控制寄存器 0

CMCON0– 比较器控制寄存器 0								
地址: 0x11A					XX000_0000			
Bit	7	6	5	4	3	2	1	0
	CMON	COUT	CMOE	CMPOL	-	CPS	CNS1	CNS0
R/W	R/W	R/W	R/W	R/W	-	W/R	R/W	W/R

Bit	Name	描述
7	CMON	比较器使能控制 1 = 使能模拟比较器 0 = 禁用模拟比较器
6	COUT	比较器输出状态
5	CMOE	使能 COUT 到外部 IO(CMO) 1 = 使能 COUT 输出到 CMO 引脚 0 = 禁止 COUT 输出到 CMO 引脚
4	CMPOL	比较器输出极性控制位
3	-	-
2	CPS	比较器正端输入通道选择: 0 = 比较器正端输入选择 CIP 引脚 1 = 比较器正端输入选择内部 DAC 输出
1	CNS1	比较器负端输入通道选择高位
0	CNS0	比较器负端输入通道选择低位。{CNS1, CNS0} = 00 = 负端输入选择 CIN0 引脚 01 = 负端输入选择 CIN1 引脚 10 = 负端输入选择 DAP 输出 11 = 负端输入选择 ADC 多路复用输出

CMCON1 – 比较器控制寄存器 1

CMCON1– 比较器控制寄存器 1								
地址: 0x11B					0000_0010			
Bit	7	6	5	4	3	2	1	0
	-	-	CFS1	CFS0	T1ACS	CMHYS	T1GSS	CMSYNC
R/W	-	-	R/W	R/W	R/W	R/W	W/R	W/R

Bit	Name	描述
7:6	-	-
5	CFS1	比较器输出数字滤波选择位
4	CFS0	比较器输出数字滤波选择位。{CFS1, CFS0} = 00 = 禁用数字滤波 01 = 32us 数字滤波 10 = 64us 数字滤波 11 = 96us 数字滤波
3	T1ACS	定时器 TMR1 计数时钟选择位 1 = 计数时钟为系统时钟

		0 = 计数时钟为指令周期
2	CMHYS	比较器迟滞功能使能控制，1=使能；0=禁止
1	T1GSS	定时器 TMR1 门控输入源选择 1 = 外部 T1G 输入作为 TMR1 门控 0 = 比较器输出作为 TMR1 门控
0	CMSYNC	比较器输出同步控制 1 = 比较器输出与 TMR1 时钟同步 0 = 比较器输出为异步模式

ANSEL – 端口模拟功能控制寄存器

ANSEL– 端口模拟功能控制寄存器								
地址: 0x11E					0x00			
Bit	7	6	5	4	3	2	1	0
	ANSB7	ANSB6	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0
R/W	R/W	R/W	R/W	R/W	W/R	R/W	R/W	W/R
Bit	Name	描述						
7	ANSB7	RB7/CIN1 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
6	ANSB6	RB6/AN6 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
5	ANSB5	RB5/AN5 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
4	ANSB4	RB4/AN4 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
3	ANSB3	RB3/AN3 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
2	ANSB2	RB2/AN2 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
1	ANSB1	RB1/AN1 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
0	ANSB0	RB0/AN0 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						

ANSELH – 端口模拟功能控制寄存器

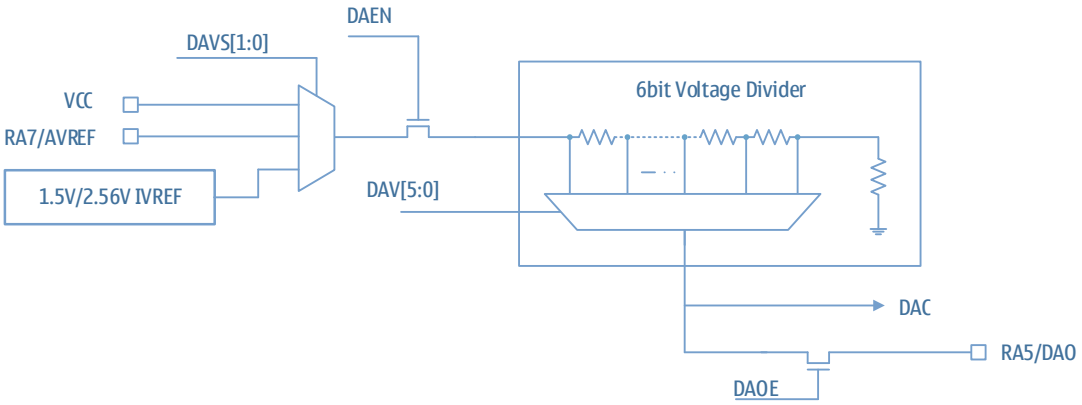
ANSELH– 端口模拟功能控制寄存器								
地址: 0x11F					0x00			
Bit	7	6	5	4	3	2	1	0
	ANSA7	-	ANSA5	-	CKEA3	ANSA2	DAPM	ANTM
R/W	R/W	-	R/W	-	W/R	R/W	R/W	W/R
Bit	Name	描述						
7	ANSA7	RA7/AN7 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
6	-	-						
5	ANSA5	RA5 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
4	-	-						
3	CKEA3	RA3 时钟输出控制, 1 = RA3 输出系统时钟， 0 = 禁止 RA3 输出系统时钟						
2	ANSA2	RA2 模拟输入模式控制, 1 = 模拟模式， 0 = 数字模式						
1	DAPM	DAP 校准寄存器选择 0 = 校准 DAP 校准配置 0；1 = 校准 DAP 校准配置 1						
0	ANTM	ADC 测试模拟。1 = 使能 ADC 测试输出； 0 = ADC 正常工作模式						

数模转换器(DAC)

- 6 位数模转换输出
- DAC 输出可作为模拟比较器参考输入
- 支持 DAC 输出到 I/O 端口
- 可选 VCC/AVREF/IVREF 分压电源

综述

LGT8F690A 内部集成一个 6 位可编程数模转换器(DAC) 。DAC 的输入参考可以选择为来自系统工作电源，内部基准电压源或者来自芯片外部的 RA7/AVREF 输入。DAC 的输出可选择作为内部比较器 CM 的输入参考，也可以之间输出至芯片的外部引脚上作为外部参考使用。当 DAC 输出至外部引脚(RA5)时，不能直接用于驱动负载，需要通过电压跟随器或其他类似的驱动电路。DAC 内部结构如下图所示：



寄存器定义

DACON – DAC 控制寄存器

DACON– DAC 控制寄存器								
地址: 0x118					0000_0000			
Bit	7	6	5	4	3	2	1	0
	DAEN	DAOE	DAV5	DAV4	DAV3	DAV2	DAV1	DAV0
R/W	R/W	R/W	R/W	R/W	R/W	W/R	R/W	W/R

Bit	Name	描述
7	DAEN	DAC 使能控制。1 = 使能 DAC 模块
6	DAOE	DAC 输出到外部引脚 RA5 使能控制。1 = 使能 DAC 输出到外部引脚
5:0	DAV	DAC 输出电压控制位。DAV[5:0] = 000000 = VIN/64 000001 = 2*VIN/64 ..... 111110 = 63*VIN/64 111111 = VIN 注：VIN 为 DAC 输入参考电压。

ADCON3 –ADC 控制寄存器 3

ADCON3– ADC 控制寄存器 3								
地址: 0x11C					0000_0000			
Bit	7	6	5	4	3	2	1	0
	DAVS1	DAVS0	VRIS	AMEN	AMID	AMFS2	AMFS1	AMFS0
R/W	R/W	W/R	W/R	W/R	R/W	W/R	R/W	W/R

Bit	Name	描述
7:6	DAVS	DAC 输入参考电压选择: 00 = 系统工作电源(VCC) 01 = AVREF 参考输入引脚 1X = 选择内部 1.5V/2.56V 参考电压
5	VRIS	当 ADC 的的参考电压选择非内部参考时，VRIS 用于选择内部参考的输出: VRIS = 0: 内部参考输出 1.5V VRIS = 1: 内部参考输出 2.56V
4	AMEN	ADC 自动阀值监控模式使能。1 = 使能 ADC 自动阀值监控
3	AMID	ADC 溢出类型指示位。当 ADC 发送阀值溢出时，AMID 表示溢出类型: AMID = 0: 下溢出 AMID = 1: 上溢出
2:0	AMFC	ADC 自动阀值监控数字滤波配置位， AMFC = 000: 关闭数字滤波 001: 连续 3 次相同有效 010: 连续 5 次相同有效 011: 连续 7 次相同有效 ..... 111: 连续 15 次相同有效

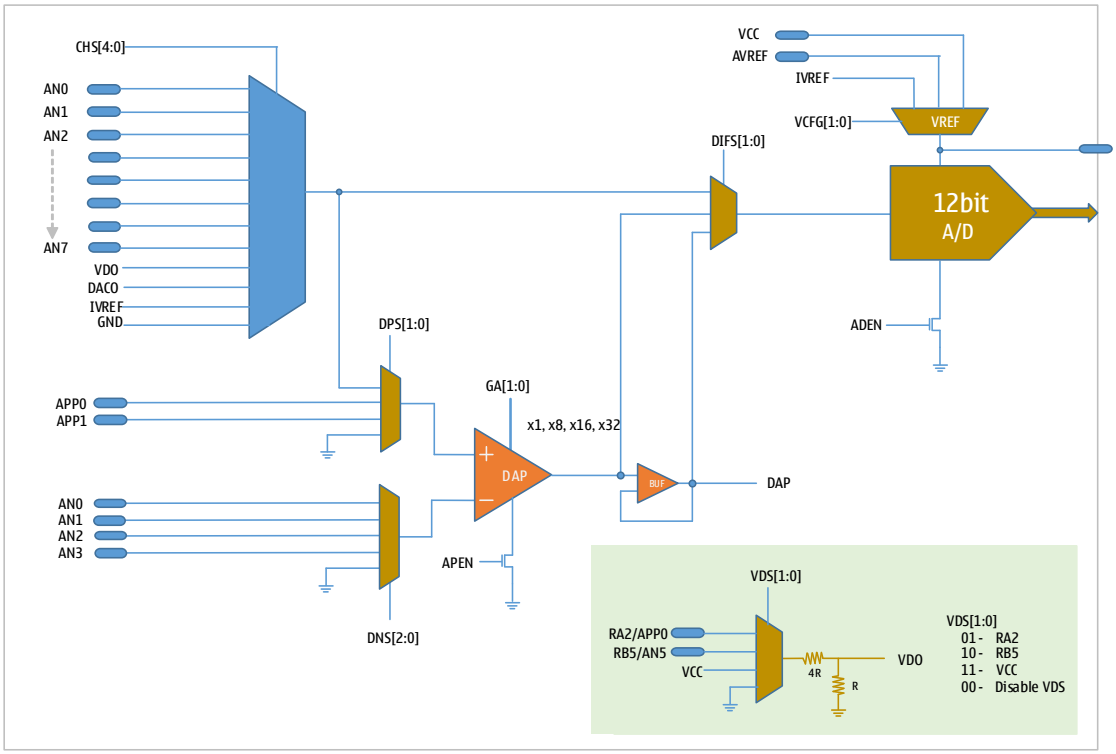
12 位模数转换器(ADC)

- INL: ±1.5LSB, DNL: ±1.5LSB
- 8 通道外部 ADC 输入
- 内置多路输入 1/5 分压电路(VDO)
- 内置多路输入差分放大器(DAP)
- 可编程差分放大增益(x1/8/16/32)
- 内部 1.5/2.56V±1%可校准参考电源(IVREF)
- 差分放大器出厂失调校准
- VCC/IVREF/AVREF 可选参考电源
- 支持自动失调校准
- 支持自动通道溢出监控模式

概述

LGT8F690A 内部集成一个高精度的 12 位模数转换器，可以将模拟输入高速高精度的转换为 12 位的数字信号。ADC 支持多路模拟输入，内部采用一个模拟多路复用开关将选中的模拟通道与 ADC 内部的采样保持电路连接。ADC 量化器采用主次逼近算法将采样保持电路的模拟信号转换为一个 12 位的二进制结果。在转换周期结束后，转换的数字信号被存储在 ADC 结果寄存器中(ADRESL/ADRESH).

ADC 内部结构示意图：



ADC 的参考基准可以根据应用需要在三种内、外部输入源之间选择，包括系统工作电源 VCC，来自部引脚输入的参考电压，以及来自内部 1.5V/2.56V 基准电压源。

由于 ADC 转换的精度对参考基准以及系统工作电源有决定性的关系，因此在一些高精度的 ADC 应用中，应该对芯片的工作电源采取有效的滤波处理。同时对来自外部的模拟输入也需要在应用允许的范围内，

在靠近芯片输入引脚的位置设置较小的滤波电容(pF 级)。

差分放大器(DAP)

单端模拟输入信号无法抑制来自系统电源以及周围环境的共模干扰，因此对于大部分小信号模数转换应用，都无法简单的通过 ADC 直接转换获得高精度的小信号测量。LGT8F690A 内部集成一个出厂校准的可编程增益差分放大器，用以克服单端输入小信号共模干扰的问题。差分放大器采用轨对轨输入结构，可接受系统工作电源范围内的输入信号，内部增益支持 x1/x8/x16/x32。用户可以根据输入信号的级别自由的选择。

APCON 的 APEN 位用于使能或者关闭差分放大器。差分放大器不仅仅可以与 ADC 协同工作，也可以完全独立运行。差分放大器的输入可以来自 ADC 内部的多路复用输出，也可以选择直接来自外部引脚。差分放大器的输出可以供 ADC 转换，也可以输出给比较器 CM，实现更加高效的电压、电流监测和保护。

软件通过 APCON 寄存器完成对差分放大器的设置。可以通过 APCON 完成差分放大器的输入源选择，放大增益选择。

DAP 设置完成后，可以通过 ADCON1 寄存器的 DIFS 位，将放大器的输出与 ADC 的采样保持电路连接，使用 ADC 实现对放大信号的测量。DAP 的输出经过一个轨到轨的跟随器，用于增强 DAP 的输出驱动能力，但跟随器会带来额外的偏移误差。对于小信号应用，建议使用 DIFS 直接选择 DAP 的输出进入 ADC 的采样通道。APCON 以及相关寄存器位的定义，请参考本章节寄存器定义部分。

LGT8F690A 的 DAP 支持调零和固定偏置电压两种校准模式。调零将 DAP 的系统失调调到最低，最大限度的提高放大输出的绝对精度。但失调调到最低同时也会让 DAP 的工作接近饱和状态，导致小信号无法测量。固定偏置的方式是将 DAP 的输入端注入一个固定的偏置电压，相当于给 DAP 一个静态工作点。这种方式比较适合测量小信号输入，偏置电压将会叠加到输入上一同放大。可以通过减去叠加，得到实际输入。

LGT8F690A 出厂测试时， 将会把调零和固定偏置的校准值都写入到配置信息。软件可以通过 ADCON3 寄存器的 APCM 位，选择合适的校准方式。

1/5 多路输入分压器(VDO)

对于一些常用的电压测量，由于采用比较低的内部或者外部参考，不能用于测试高于参考电压的电位。同样，为实现 ADC 的内部电源电压测量通道，我们必须将电源电压分压至一个合理的电位，然后才能使用 ADC 进行准确的转换。为此，LGT8F690A 内部集成了一个具有固定 1/5 分压的多路输入分压器，可以满足对系统电源电压的直接测量。

多路输入分压器的输入可以通过 ADCON1 寄存器的 VDS1/0 位进行选择控制。分压输入可以在外部引脚输入 RA2/APP0，RB5/AN5 或者系统工作电源 VCC 三者之间选择。当选择系统工作电源时，可以通过设置 ADC 的模拟输入多路复用，选中 VDO 作为 ADC 当前采用输入通道，完成对电源电压的测量。

在系统进入休眠模式前， 请确认 VDS 选择为 0，禁止 VDS 分压电路，关闭分压电路带来的耗电。

ADC 自动触发模式

ADC 可以在外部事件的触发下自动启动转换。LGT8F690A 支持 4 种外部触发事件：

1. 来自定时器 1 通道 3 的比较匹配事件
2. 来自定时器 2 通道 4 的 PWM 输出
3. 来自外部 RA1 引脚的触发事件
4. 来自外部 RB3 引脚的触发事件

ADC 在自动触发模式下，将在检测到上述任一事件的上升沿后，启动 ADC 转换。来自定时器 1 通道 3 的比较匹配事件，可以用于实现定时 ADC 转换的控制。来自外部引脚的触发事件，可以实现更加灵活的自

动转换模式。

自动触发模式也可以与接下来介绍的通道阀值监控模式配合使用，实现定时事件触发的通道监控。

使用自动触发模式前， 首先根据需求，配置 ADC 参考电压，选择需要监控的模拟输入通道。自动触发模式由 ADCON2 寄存器的 ATEN 位控制。触发源通过 ADCON2 寄存器的 ADTS 位选择。

上述参数配置完成后， 通过置位 ADCON0 寄存器的 ADON 使能 ADC 模块。ADC 即就绪进入自动触发模式。

ADC 通道阀值监控

通道监控模式用于实时监控指定 ADC 输入通道的电压变化情况。通过给定最低阀值和最高阀值，ADC 实时采样通道电压， 并将采样结果与最低和最高阀值进行比较。当 ADC 输入电压低于最低阀值或者高于最高阀值时， ADC 转换即刻停止，并置位 ADIF 中断标志位，并根据溢出情况设置 ADCON3 寄存器的 AMID 标志位。软件可以通过读取 AMID 的值，判断溢出类型。

自动阀值监控功能通过置位 ADCON3 寄存器的 AMEN 位使能和关闭。ADC 通道阀值监控模式可以与自动触发模式配合工作。在没有使能自动触发模式的情况下，ADC 阀值监控工作在自动转换模式，自动转换模式在一次转换完成后立刻开始下一次转换，直到出现 ADC 的转换结果在监控阀值之外。自动转换模式具有很好的实时性，但也会带来功耗的增加。

在使能自动触发模式后，阀值监控也同时工作在自动触发模式。在这种模式下，可以使用定时器比较匹配事件作为触发源，通过对定时器的配置，可以实现确定时间的同步阀值监控，即可以降低 ADC 频繁工作带来的功耗， 同时也可以与自动触发事件配合实现更加灵活的闭环控制系统。

ADC 阀值监控的高低阀值设置， 是通过 ADTL/ADTH 寄存器实现。ADTL/ADTH 为 8 位寄存器。进行阀值比较时，ADTL 作为最低阀值，与 ADC 采样结果的高 8 位(ADRES[11:4])进行比较，用于监控输入通道的下溢出门限。ADTH 作为最高阀值，与 ADC 采样结果的高 8 位(ADRES[11:4])进行比较，用于监控输入通道的上溢出门限。

ADTL/ADTH 寄存器为只写寄存器，寄存器地址与 ADRESL/ADRESH 复用。ADRESL/ADRESH 为只读寄存器。ADCON2 寄存器的 ARSB 位用于使能对 ADTL/ADTH 寄存器的写操作。置位 ARSB 位，然后写 ADRESL、ADRESH 寄存器地址，完成对 ADTL、ADTH 寄存器的写操作。在任何情况下，读 ADRESL/ADRESH 寄存器都将返回 ADC 上一次采样转换的结果。

设置 ADTL/ADTH 实例代码：

```
#include “lgt8f690a.h”

// ADC 初始化
void adcInit()
{
    ARSB = 1;                                   // 使能 ADTL/ADTH 写访问
    ADTL = 0x80;                                // 设置下溢阀值
    ADTH = 0x83;                                // 设置上溢阀值
    ARSB = 0;                                   // 关闭，可选操作
    .....
}
```

为避免输入干扰对阀值监控的影响，ADC 阀值监控实现了一个可配置的滤波功能。通过 ADCON3 寄存器的 AMFC 位，可以设置阀值监控的有效采样计数(AMFC *2 + 1)。比如设置 AMFC=2 时，阀值滤波计数设置为 2*2+1 = 5，此时需要连续采样到 5 次相同的结果，才会对是否溢出做出最终判断。默认情况下 AMFC=0，也就是关闭了采样滤波功能。

当发生阀值溢出时，自动阀值监控功能将会被关闭(AMEN=0)，同时置位 ADC 中断标志位 ADIF。软件可以根据 ADCON3 寄存器的 AMID 位查询本次溢出的类型（上、下溢出）。需要通过软件重新置位 AMEN 位才能够重新启动阀值监控功能。

ADC 自动失调校准

LGT8F690A 的 ADC 支持失调校准。失调校准在每次测量时进行，因此可以很好的避免电压以及温度变化对失调造成的影响。

失调校准的原理是利用比较器失调极性与输入极性的关系实现。ADC 内部量化比较器是 ADC 失调的主要原因。当比较的输入反向后，失调也同样呈现极性相反的特性。比如在正常的转换模式下，ADC 的转换电压测量值为：

Vadc = Vin + Vos (其中 Vadc 为 ADC 转换电压，Vin 为实际输入电压，Vos 为失调)

当把比较器输入反向后，转换电压为：

Vadc = Vin – Vos

根据比较器的这个特性， 我们可以将两次转换的结果相加，把失调的影响相互抵消。因此失调校准需要进行连续两次转换，然后相加求均值。这实际上也经行了一次 ADC 的滤波处理。

ADC 的自动失调校准由 ADCON2 寄存器的 OFEN 位控制，置位 OFEN 位使能自动失调校准。此后的 ADC 转换都将自动失调校准。由于需要进行 2 次 ADC 转换，因此自动校准模式下，ADC 的转换周期是普通模式的 2 倍。普通模式下一次 ADC 转换需要 14 个周期，自动校准模式下需要 28 个采样周期。

ADC 工作相关配置

在使用 ADC 之前，需要首先完成一下系统准备工作：

- 端口模式配置
- ADC 输入通道选择
- ADC 参考电压选择
- ADC 转换时钟选择
- 中断相关控制设置
- ADC 转换结果格式设置

端口相关配置

ADC 常用与转换模拟输入信号。因此在转换之前需要将输入 I/O 设置为模拟工作模式。ANSEL/H 寄存器用于关闭 IO 的数字输入，并不会控制端口的数字输入输出方向；需要通过 TRISA/C 寄存器将对应的端口设置为输入状态后，端口才能正确的表现外部模拟输入的电平变化。

ADC 输入通道选择

ADC 支持多路模拟输入，内部通过一个模拟多路复用器将多路输入复用到 ADC 的采样保持电路。因此在启动 ADC 转换器，首先需要设置 ADC 转换的当前输入通道。LGT8F690A 的 ADC 多路复用内部一共支持 12 路输入信号，其中 0~7 通道对应芯片的外部输入端口 AN0~7；通道 8~11 为内部模拟输入专用。通道 8 连接内部 1/5 分压器(VDO)；通道 9 接内部 DAC 输出，通道 10 连接内部的 1.5V/2.56V 参考基准；通道 11 接芯片内部模拟地(AGND)。ADCON0 寄存器的 CHS3-0 用于选择当前模拟输入通道。

除了 ADC 专用的多路复用输入通道，ADC 还可以转换来自内部差分放大器(DAP)输出；通过将 ADCON1 寄存器的 DIFS[1:0]位，将差分放大器的输出接入 ADC 的转换输入，可以实现对差分放大器输出电压的测量。

ADC 转换参考基准

ADCON0 寄存器的 VCFG0 与 ADCON1 寄存器的 VCFG1 共同组合为 VCFG[1:0]，用于选择 ADC 转换的参考电压基准。参考基准可以在系统工作电源电压，外部 RA7/AVREF 输入电压以及内部基准之间选择。

ADC 转换时钟选择

LGT8F690A 内部 ADC 最高支持 2MHz 的转换时钟频率。高速的转换速率将会对 ADC 的转换精度产生影响。当用于主要较高的转换精度时，可通过降低 ADC 的转换时钟频率实现。ADC 推荐的转换时钟为 256KHz~1MHz 左右。可以根据应用的系统时钟频率， 通过 ADC 时钟与分频器，选择合适的转换时钟。

ADC 采样逐次逼近的量化算法，转换 1 个位需要 1 个 ADC 转换周期，12 位的结果一共需要 13 个转换周期才能完成，加上系统采用转换完成信号的同步时钟，ADC 一次转换需要 14 个 ADC 转换时钟周期。

ADCON1 寄存器的 ADPS 位用于选择 ADC 的转换时钟，转换时钟可以在系统工作时钟频率的 2 分频到 64 分频之间选择。详细定义，请参考本章节寄存器定义部分。

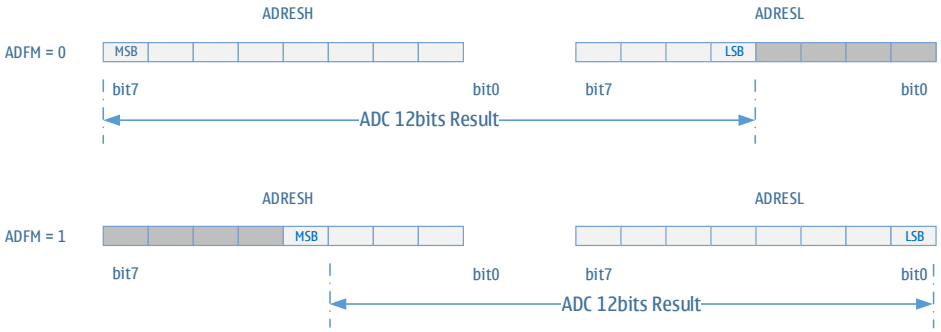
相关中断设置

ADC 在转换完成后，可以产生一个转换完成中断请求。ADC 转换完成中断的标志位位于 PIE1 寄存器的 ADIF 位。ADC 中断使能控制位为 PIE1 寄存器的 ADIE 位。ADIF 标志位必须在中断复位程序中由软件清零。

ADIF 位在每次转换完成的同一周期产生。即使没有使能 ADC 的中断功能， ADIF 位仍然会置位。当系统使能了 ADC 的中断请求功能，并且系统也通过 GIE 位使能了全局中断请求，ADC 转换完成后，将会中断系统指令的正常运行流程，转向执行硬件中断服务程序。用户需要在硬件中断服务程序中通过 PIE1 寄存器判断中断发生的设备，根据中断源的不同进行对应的处理。

转换结果格式

ADC 的转换结果位一个 12 位的二进制数据。被分为两个部分保存到两个 8 位的寄存器(ADRESH/L)。ADCON0 寄存器的 ADFM 位用于设置 12 位结果在两个 8 位寄存器中的存放格式。ADFM 的配置与对应的 ADC 结果格式如下图所示：



ADC 操作流程

启动转换

ADCON0 寄存器的 ADON 位用于使能 ADC 模块；ADON=1 后，系统开启给 ADC 供电，ADC 进入等待转换开始模式。此后，需要通过设置 ADCON0 寄存器的 GO/DONE 位开启一次转换流程。需要注意的时，开启 ADC 转换流程与 ADC 使能不可同时设置。同时设置的开启转换动作无效。

转换完成识别

ADC 完成一次转换后，将会进行如下状态变化：

- 将 ADCON0 的 GO/DONE 位清零
- 设置 ADIF 标志位
- 将 ADC 的转换结果更新到 ADRESH/L 寄存器

ADC 转换流程实例

```
        ; ADC  转换实例配置

DATAH   equ     0x70
DATAL   equ     0x71            ; define temple variable

        BSF     STATUS, RP0     ; select bank 1
        MOVLW   0x10            ; ADCS = 001
        MOVWF   ADCON1
        BSF     TRISA, 0x2      ; set RA2 to input
        BSF     ANSEL, 0x2      ; set RA2 to analog

        BCF     STATUS, RP0     ; select bank 0
        MOVLW   0x89            ; set ADFM=1, select AN2 as current channel
        MOVWF   ADCON0          ; and enable ADC module
START   NOP
        BSF     ADCON0, 0x1     ; start ADC convert
CLOOP   NOP
        BTFSZ   ADCON0, 0x1     ; wait for coverting done
        GOTO    CLOOP           ; continue waiting
        MOVF    ADRESH, W       ; store ADC high byte
        MOVWF   DATAH           ; to template memory (we only need high 8bits)
        CALL    adc_function    ; call data process
        GOTO    START           ; start another convert cycle
        END                     ;
```

寄存器定义

ADCON0 –ADC 控制寄存器 0

ADCON0– ADC 控制寄存器 0								
地址: 0x1F					0000_0000			
Bit	7	6	5	4	3	2	1	0
	ADFM	VCFG1	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
R/W	R/W	R/W	R/W	R/W	R/W	W/R	R/W	W/R

Bit	Name	描述
7	ADFM	ADC 转换结果格式设置。1 = 右对齐；0 = 左对齐
6	VCFG1	ADC 参考电压配置位 1。VCFG1 与 ADCON1 寄存器的 VCFG0 组合选择 ADC 的参考电压。VCFG[1:0] = 00：选择系统工作电源作为参考电压 01：选择内部 1.5V 参考基准 10：选择 RA7/AVREF 外部参考 11：选择内部 2.56V 参考基准
5:2	CHS[3:0]	ADC 输入通道选择位，CHS[3:0] =

		0000：选择外部输入 AN0 0001：选择外部输入 AN1 ..... 0111：选择外部输入 AN7 1000：选择内部 1/5 分压器 1001：选择内部 DAC 的输出 1010：选择内部 1.5V/2.56V 基准电压 1011：选择内部 VGND
1	GO/DONE	ADC 转换开始以及转换完成标志位，写 1 启动一次 ADC 转换。此位为 0 表示 ADC 转换完成，ADC 处于空闲状态。
0	ADON	ADC 使能控制位，1=使能 ADC 模块；0=关闭 ADC 模块

ADCON1 –ADC 控制寄存器 1

ADCON1– ADC 控制寄存器 1								
地址: 0x9F					0000_0000			
Bit	7	6	5	4	3	2	1	0
	VCFG0	ADCS2	ADCS1	ADCS0	DIFS1	DIFS0	VDS1	VDS0
R/W	R/W	R/W	R/W	R/W	R/W	W/R	R/W	W/R
Bit	Name	描述						
7	VCFG0	与 ADCON0 的 VCFG1 组合用于选择 ADC 参考电压。 请参考 ADCON0 寄存器关于 VCFG1 的定义。						
6:4	ADCS[2:0]	ADC 转换时钟配置位。ADCS[2:0] = 000/011: 转换时钟为 FSYS/2 001: 转换时钟为 FSYS/8 010: 转换时钟为 FSYS/32 100: 转换时钟为 FSYS/4 101: 转换时钟为 FSYS/16 110: 转换时钟为 FSYS/64 111: 转换时钟为 FSYS/32 其中 FSYS 为系统工作频率						
3:2	DIFS	差分放大器输出到 ADC 采用通道使能位。DIFS = 00 = ADC 采样模拟多路复用 01 = ADC 采样经过跟随器的 DAP 输出 1X = ADC 直接采样 DAP 输出						
1:0	VDS[1:0]	内部 1/5 分压电路输入源选择。VDS[1:0] = 00：关闭分压电路模块(VDO) 01：选择 RA2/APP0 作为分压电路输入电压 10：选择 RB5/AN5 作为分压电路输入电压 11：选择系统工作电源 VCC 作为输入电压						

ADCON2 –ADC 控制寄存器 2

ADCON2– ADC 控制寄存器 2								
地址: 0x9C					0000_0000			
Bit	7	6	5	4	3	2	1	0
	OFID	OFEN	SPD	SPN	ATEN	ARSB	ADTS1	ADTS0
R/W	R/W	R/W	R/W	R/W	R/W	W/R	R/W	W/R

Bit	Name	描述
7	OFID	ADC 内部比较器指示位，用于失调校准
6	OFEN	ADC 自动失调校准使能位。1 = 使能自动失调校准
5	SPD	ADC 低速采样模式使能。用于弱信号低速采样
4	SPN	ADC 内部比较器输入方向控制位
3	ATEN	ADC 自动触发模式使能。1 = 使能自动触发模式
2	ARSB	ADC 溢出阀值写选择位。 ARSB = 1，写 ADRESL/H 寄存器完成对 ADTL/H 的设置
1:0	ADTS	ADC 自动转换触发源选择。ADTS = 00: 触发源来自定时器 1 通道 3 匹配溢出事件 01: 触发源来自定时器 2 通道 4 的 PWM 输出 10: 触发源来自外部输入 RA1 的上升沿 11: 触发源来自外部输入 RB3 的上升沿

ADCON3 –ADC 控制寄存器 3

ADCON3– ADC 控制寄存器 3								
地址: 0x11C					0000_0000			
Bit	7	6	5	4	3	2	1	0
	DAVS1	DAVS0	VRIS	AMEN	AMID	AMFC2	AMFC1	AMFC0
R/W	R/W	R/W	R/W	R/W	R/W	W/R	R/W	W/R

Bit	Name	描述
7:6	DAVS	DAC 输入参考电压选择： 00 = 系统工作电源(VCC) 01 = AVREF 参考输入引脚 1X = 选择内部 1.5V/2.56V 参考电压
5	VRIS	当 ADC 的的参考电压选择非内部参考时，VRIS 用于选择内部参考的输出： VRIS = 0: 内部参考输出 1.5V VRIS = 1: 内部参考输出 2.56V
4	AMEN	ADC 自动阀值监控模式使能。1 = 使能 ADC 自动阀值监控
3	AMID	ADC 溢出类型指示位。当 ADC 发送阀值溢出时，AMID 表示溢出类型： AMID = 0: 下溢出 AMID = 1: 上溢出
2:0	AMFC	ADC 自动阀值监控数字滤波配置位， AMFC = 000: 关闭数字滤波 001: 连续 3 次相同有效 010: 连续 5 次相同有效 011: 连续 7 次相同有效

		…… 111: 连续 15 次相同有效

ADRESH –ADC 转换结果高位

ADRESH– ADC 转换结果高位								
地址: 0x1E/0x9D					XXXX_XXXX			
Bit	7	6	5	4	3	2	1	0
	ADRESH[7:0]							
R/W	R/0	R/0	R/0	R/0	R/0	R/0	R/0	R/0
Bit	Name	描述						
7:0	ADRESH	ADC 转换结果高位，具体含义请参考本章节关于转换结果格式的介绍 ADRESH 被同时映射到第 0 页的 0x1E 和第 1 页的 0x9D 使用 0x9D 地址与 ADRESL 在同一个页，可以减少切切换代价。						

ADRESL –ADC 转换结果低位

ADRESL– ADC 转换结果低位								
地址: 0x9E					XXXX_XXXX			
Bit	7	6	5	4	3	2	1	0
	ADRESL[7:0]							
R/W	R/0	R/0	R/0	R/0	R/0	R/0	R/0	R/0
Bit	Name	描述						
7:0	ADRESL	ADC 转换结果低位，具体含义请参考本章节关于转换结果格式的介绍						

ADTL –ADC 下溢出阀值

ADTL– ADC 下溢出阀值高 8 位								
地址: 0x9E					0x00			
Bit	7	6	5	4	3	2	1	0
	ADTL[7:0]							
R/W	W/0	W/0	W/0	W/0	W/0	W/0	W/0	W/0
Bit	Name	描述						
7:0	ADTL	ADC 下溢出阀值高 8 位。为只写寄存器。 写 ADTL 之前，需要先置位 ADCON2 寄存器的 ARSB 位						

ADTH –ADC 上溢出阀值

ADTH– ADC 上溢出阀值高 8 位								
地址: 0x1E/0x9D					0x00			
Bit	7	6	5	4	3	2	1	0
	ADTH[7:0]							
R/W	W/0	W/0	W/0	W/0	W/0	W/0	W/0	W/0
Bit	Name	描述						
7:0	ADTH	ADC 上溢出阀值高 8 位。为只写寄存器。 写 ADTH 之前，需要先置位 ADCON2 寄存器的 ARSB 位						

APCON – 差分放大器控制寄存器

APCON– 差分放大器控制寄存器								
地址: 0x11d					0000_0000			
Bit	7	6	5	4	3	2	1	0
	APEN	DPS1	DPS0	DNS2	DNS1	DNS0	GA1	GA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	描述
7	APEN	差分放大器使能控制。1 = 使能差分放大器
6:5	DPS	差分放大器正向输入源选择; DPS[1:0] = 00 : 选择 ADC 多路复用输出 01 : 选择 RA2/APP0 10 : 选择 RA5/APP1 11 : 选择内部 GND
4:2	DNS	差分放大器反向输入源选择; DNS[2:0] = 000 : 选择 RB0/AN0 001 : 选择 RB1/AN1 010 : 选择 RB2/AN2 011 : 选择 RB3/AN3 100 : 选择内部 GND 其他配置: 关闭 DNS 通道
1:0	GA[1:0]	差分放大器增益选择; GA[1:0] = 00 : x1 01 : x8 10 : x16 11 : x32

ANSEL – 端口模拟功能控制寄存器

ANSEL– 端口模拟功能控制寄存器								
地址: 0x11E					0x00			
Bit	7	6	5	4	3	2	1	0
	ANSB7	ANSB6	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0
R/W	R/W	R/W	R/W	R/W	W/R	R/W	R/W	W/R

Bit	Name	描述
7	ANSB7	RB7/CIN1 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
6	ANSB6	RB6/AN6 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
5	ANSB5	RB5/AN5 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
4	ANSB4	RB4/AN4 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
3	ANSB3	RB3/AN3 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
2	ANSB2	RB2/AN2 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
1	ANSB1	RB1/AN1 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
0	ANSB0	RB0/AN0 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式

ANSELH – 端口模拟功能控制寄存器

ANSELH– 端口模拟功能控制寄存器								
地址: 0x11F					0x00			
Bit	7	6	5	4	3	2	1	0
	ANSA7	-	ANSA5	-	CKEA3	ANSA2	DAPM	ANTM
R/W	R/W	-	R/W	-	W/R	R/W	R/W	W/R

Bit	Name	描述
7	ANSA7	RA7/AN7 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
6	-	-
5	ANSA5	RA5 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
4	-	-
3	CKEA3	RA3 时钟输出控制, 1 = RA3 输出系统时钟, 0 = 禁止 RA3 输出系统时钟
2	ANSA2	RA2 模拟输入模式控制, 1 = 模拟模式, 0 = 数字模式
1	DAPM	DAP 校准寄存器选择 0 = 校准 DAP 校准配置 0; 1 = 校准 DAP 校准配置 1
0	ANTM	ADC 测试模拟。1 = 使能 ADC 测试输出; 0 = ADC 正常工作模式

编程烧录接口

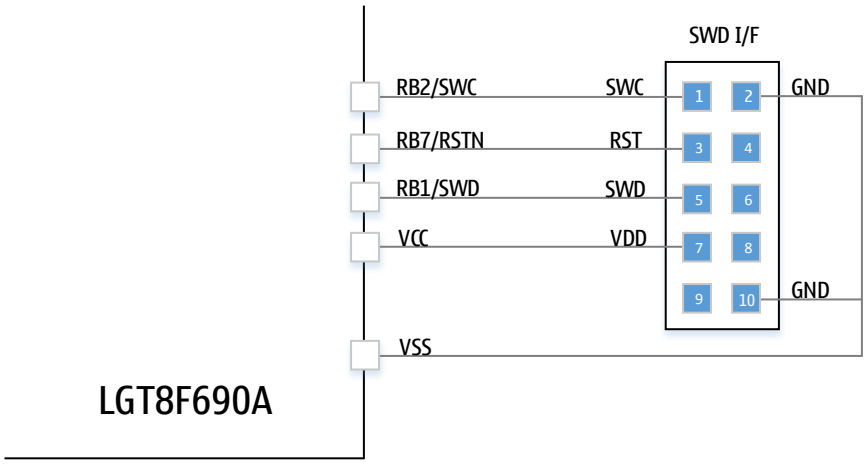
LGT8F690A 支持通过 SWD 双线串行接口编程烧录。编程接口定义如下：

SSOP20/QFN20	引脚名称	SWD 编程接口
18	RB1/SWD	SWD
19	RB2/SWC	SWC
4	RB7/RSTN	RST
9	VCC	VDD
7	VSS	GND

LGT8F690A 编程烧录所需软件/硬件：

硬件	硬件驱动	PC 配套软件
SWDICE mkII Pro 在线编程器 购买连接	SISP 模式 SWDISP_mkII Driver	LGTMix_ISP 版本 v3.7.2 或更新
LGTUSB_Offline HV 脱机下载器 购买连接	LGTUSB_Offline Driver	LGTMCU_Offline 版本 v2.4 或更新

编程器接口连接示意图:



系统配置位

LGT8F690A 包含一个独立的配置字，用于设置系统以及外设的运行模式。本章主要介绍配置字的定义，配置字以及编程信息，请参考相关资料。

配置字 1：系统时钟相关配置

Bit13 Bit0

TCYC[1:0]	RSTNE	WDTE	SUT[1:0]	LVDPM	LVDT[2:0]	OSCFSEN	FOSC[2:0]

位定义	
FOSC[2:0]	系统时钟模式配置字 000/111: HFINTOSC 模式，内部高速 RC ~~110: RCK，32KHz 内部 RC 时钟~~ 101: HFOSC，外部高速晶振模式 100: CLKIN，外部时钟输入模式(RA5) 011: LFOSC，外部低速(32.768KHz)晶振模式 Others: Unimplemented
OSCFSEN	晶振生效保护控制。1 = 使能晶振失效保护； 0 = 关闭晶振失效保护
LVDT[2:0]	低压复位模块复位电压阀值设置 000: 2.0V 001: 2.2V 010: 2.4V 011: 2.6V 100: 3.6V 101: 4.0V 110/111: 关闭低压复位模块
LVDPM	低压复位模块功耗模式控制 1: 休眠时 LVD 保持工作 0: 休眠时关闭 LVD
SUT[1:0]	启动时间配置字 11: 2ms 10: 150us 01: 254ms 00: 63ms
WDTE	使能看门狗定时器
RSTE	使能 RA3 作为外部复位输入
TCYC[1:0]	内核指令周期模式设置 00: 1T 指令周期 01: 2T 指令周期 1X: 4T 指令周期

指令集速查表

指令名称	操作	功能描述	指令字	状态位	周期
基本算术运算指令					
SUBWF	SUBWF F[1], d[2]	[W/F] = (F) – W	1	C/DC/Z	1
DECF	DECF F, d	[W/F] = (F) – 1	1	Z	1
ADDWF	ADDWF F, d	[W/F] = (F) + W	1	C/DC/Z	1
COMF	COMF F, d	[W/F] = complement F	1	Z	1
INCF	INCF F, d	[W/F] = (F) + 1	1	Z	1
SUBLW	SUBLW k[4]	W = k – W	1	C/DC/Z	1
ADDLW	ADDLW k	W = k + W	1	C/DC/Z	1
ADCWF	ADCWF F, d	[W/F] = (F) + W + C	1	C/DC/Z	1
SBCWF	SBCWF F, d	[W/F] = (F) – W + C	1	C/DC/Z	1
CLRF	CLRF F	(F) = 0	1	Z	1
CLRW	CLRW	W = 0	1	Z	1
基本逻辑运行指令					
IORWF	IORWF F, d	[W/F] = (F) | W	1	Z	1
ANDWF	ANDWF F, d	[W/F] = (F) & W	1	Z	1
XORWF	XORWF F, d	[W/F] = (F) ^ W	1	Z	1
RRF	RRF F, d	[W/F] = {C, F[6:0]}	1	C	1
RLF	RLF F, d	[W/F] = {F[6:0], C}	1	C	1
SWAPF	SWAPF F, d	[W/F] = {F[3:0], F[7:4]}	1	-	1
BCF	BCF F, b[3]	F[b] = 0	1	-	1
BSF	BSF F, b	F[b] = 1	1	-	1
ANDLW	ANDLW k	W = k & W	1	Z	1
XORLW	XORLW k	W = k ^ W	1	Z	1
IORLW	IORLW k	W = k | W	1	Z	1
基本流程控制指令					
DECFSZ	DECFSZ F, d	[W/F] = (F) – 1 IF(Z) PC = PC + 2	1	-	1/2
INCFSZ	INCFSZ F, d	[W/F] = (F) + 1 IF(Z) PC = PC + 2	1	-	1/2
BTFSC	BTFSC F, b	IF(!F[b]) PC = PC + 2	1	-	1/2
BTFSS	BTFSS F, b	IF(F[b]) PC = PC + 2	1	-	1/2
GOTO	GOTO k	PC = k	1	-	2
CALL	CALL k	PC+1 -> STACK, PC = k	1	-	2
RETURN	RETURN	PC <- STACK	1	-	2
RETFIE	RETFIE	GIE = 1 PC <- STACK	1	-	2
RETLW	RETLW k	W = k, PC <- STACK	1	-	2
基本数据传输指令					

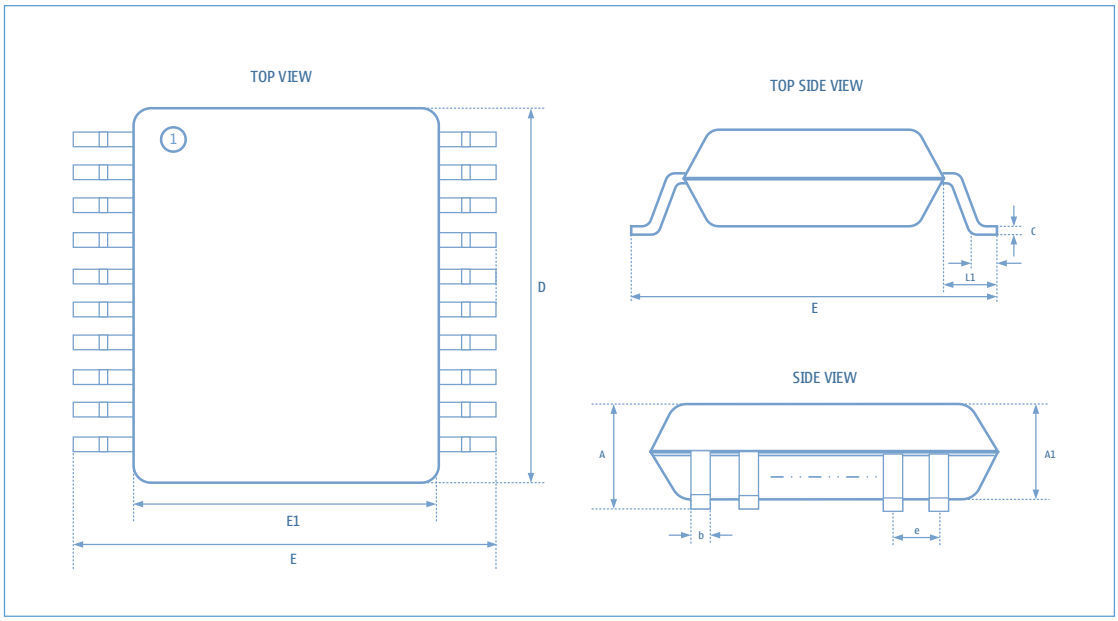
MOVWF	MOVWF F	(F) = W	1	-	1
MOVF	MOVF F, d	[W/F] = (F)	1	Z	1
MOVLW	MOVLW k	W = k	1	-	1
其他辅助指令					
NOP	NOP	NO operation	1	-	1
OPTION	OPTION	OPTION = W	1	-	1
TRIS	TRIS F	IOSTA/B = W, (F=5/6)	1	-	1
SLEEP	SLEEP	Sleep mode	1	TO/PD	1
CLRWDT	CLRWDT	Clear WDT	1	TO/PD	1

说明：

1. 指令集中的 F 为一个 6 位宽度的立即数，用于指定 IO/RAM 的地址；
2. 指令集中的 d 为一个 1 位宽度的立即数，当 d=0（默认）时，指令执行的结果回写到 W 工作寄存器，否则写入到 F 指定地址；
3. 指令集中的 b 为一个 3 位的立即数， 用于指定访问数据的位地址；
4. 指令集中的 k 为一个立即数，根据指令不同，k 的宽度分为 8/10 位不等，8 位宽度一般是数据，10 位为程序目标地址；
5. 指令集中的 q 为一个 4 位的立即数，指定间接寻址模式的地址偏移量
6. PMEM 表示程序空间，DMEM 代表数据寄存器空间

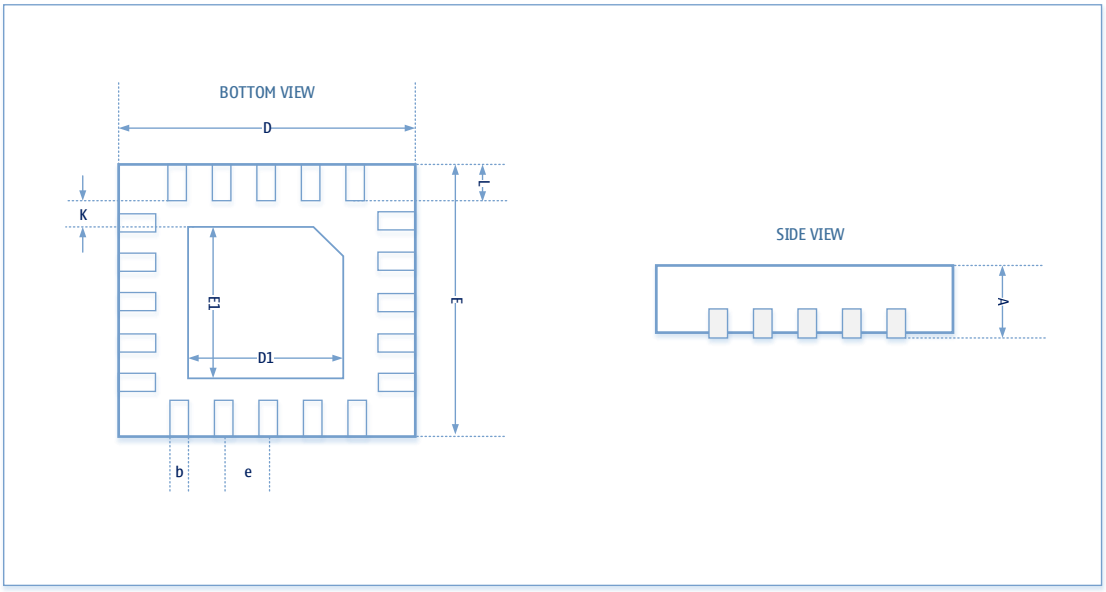
封装参数

TSSOP20L 封装



Item	Min.	Typ.	Max.	Unit
A	1.05	1.15	1.2	mm
A1	0.95	1.00	1.05	mm
b	0.16	0.22	0.31	mm
e	-	0.65BSC	-	mm
E	6.2	6.4	6.6	mm
E1	4.3	4.4	4.5	mm
D	-	-	6.9	mm
C	0.14	0.15	0.18	mm
L	0.45	0.5	0.75	mm
L1	-	1.0BSC	-	mm

QFNWB3X3 - 20L 封装



Item	Min.	Typ.	Max.	Unit
A	0.700/0.800		0.800/0.900	mm
D	2.924		3.076	mm
E	2.924		3.076	mm
D1	1.400		1.600	mm
E1	1.400		1.600	mm
K	-	0.200MIN		mm
b	0.150		0.250	mm
e		0.400TYP		mm
L	0.324		0.476	mm

版本历史

V1.0.9 2018/9/1	增加编程烧录接口定义
V1.0.8 2018/3/22	更正了部分寄存器描述中的笔误 增加 QFNWB3x3-20 封装定义
V1.0.7 2017/10/17	修改 EEP 章节中关于 EECON1 寄存器的定义，增加 EECON2 寄存器说明 改正寄存器列表中 EEP 相关寄存器的地址错误！
V1.0.6 2017/9/7	更正 AN7 的引脚位置到 RB7 更新 TIMER1 时钟源选择表格中 TMR1CS 位的定义
V1.0.5	更正 CLKI 时钟输入到 RC1 引脚 增加动态功耗控制 PPLP 的操作说明
V1.0.4 2017/6/30	更正外部中断输入为 RA1/INT 端口 ECCP2 的 PWM 自动关闭，将 CMIF 源更改为比较器输出(COUT)
V1.0.3 2017/6/20	更正高速模式了 USART 波特率计算公式 更正了 SPI 部分寄存器命名（与头文件定义一致） 内部参考 1.5V/3.0V 更改为 1.5V/2.56V
V1.0.2 2017/6/8	更改了模拟比较器 CM 的负端输入相关说明
V1.0.1 2017/05/13	更改 ADCON1/VDS 定义的描述 更改低功耗例程代码
V1.0.0 2017/05/05	初始版本