PAN2416AV
产品说明书

2.4GHz 无线收发 SOC 芯片
V1.0

PAN2416AV 产品说明书

2.4GHz 单片高速无线收发 SOC 芯片

概述

PAN2416AV芯片是工作在2.400~2.483GHz世界通用ISM频段的单片无线收发芯片。该芯片集成射频收发机、频率发生器、晶体振荡器、调制解调器和低功耗MCU等功能模块，并且支持一对多组网和带ACK的通信模式。用户通过MCU的I/O口向芯片发出指令，芯片自动完成收发配置进行通信，并根据应答信息自动判断数据发送/接收是否成功，从而进行重发，丢包，继续发送和等待等操作，简化了用户程序。发射输出功率、工作频道以及通信数据率均可配置。PAN2416AV需要少量的外围器件，支持单层/双层印制电路板的方案。

主要特性

1、功耗较低

发射模式（2dBm）工作电流19mA；接收模式工作电流15mA；休眠电流2uA。

2、节省外围器件

支持外围5个元器件，包括1颗晶振和少量电容；

支持双层或单层印制板设计，可以使用印制板微带天线或者导线天线；

芯片自带部分链路层的通信协议；配置少量的参数寄存器，使用方便。

3、性能优异

250K/1M/2M bps模式的接收灵敏度为-91/-87/-83dBm；发射输出功率最大可达8dBm；抗干扰性好，接收滤波器的邻道抑制度高，接收机选择性较好。

4、集成 MCU 功能

OTP：4K×16Bit；

通用RAM：176×8Bit；

MCU集成高精度12位ADC，内置WDT定时器、PWM输出、低压侦测电路等模块。

其它特性

四线 SPI 接口通信	带自动扰码和CRC校验功能
支持最大数据长度为32字节（两级FIFO） 或者 64字节（单级FIFO）	SOP16封装
1M / 2Mbps模式，需要晶振精度 ±40ppm 250kbps模式，需要晶振精度 ±20ppm	工作电压支持2.2~3.3V 工作温度支持-40~+85℃
GFSK通信方式	支持自动应答及自动重传
支持RSSI检测功能	10个GPIO
中断源	三路定时器

应用方案

无线鼠标	电视和机顶盒遥控器
无线游戏手柄	遥控玩具
常用遥控器	智能家居

版本	修订时间	更新内容	相关文档
V0.4	2016. 12	文字勘误	
V1.0	2017.09	文字修改	

目录

1. 命名规则

1.1 PAN2416AV 命名规则

图1.1 PAN2416系列产品命名规则

1.2 PAN2416 系列产品选择

表1-1 PAN2416系列产品选择

产品型号	芯片版本	封装形式
PAN2416AV	A	V：SOP16
PAN2416AF	A	F：SOP14

2. 主要电特性

表2-1 PAN2416AV的RF部分主要电特性

特　性	测试条件(VCC = 3V±5%，TA=25℃)	参数值			单位
		最小	典型	最大	
ICC	休眠模式		2		uA
	待机模式 1		30		uA
	待机模式 3		650		uA
	待机模式 2		780		uA
	发射模式 (-35dBm)		9		mA
	发射模式 (-20dBm)		9.5		mA
	发射模式 (0dBm)		16		mA
	发射模式 (2dBm)		19		mA
	发射模式 (8dBm)		30		mA
	发射模式 (13dBm)		66		mA
	接收模式 (250Kbps)		15		mA
	接收模式 (1Mbps)		15.5		mA
	接收模式 (2Mbps)		16.5		mA
系统指标					
f_{OP}	工作频率	2400		2483	MHz

PLL_{res}	锁相环频率步径		1		MHz
f_{XTAL}	晶振频率		16		MHz
DR	码率	0.25		2	Mbps
Δf_{250K}	调制频偏@250Kbps		125	150	KHz
Δf_{1M}	调制频偏@1Mbps		160	300	KHz
Δf_{2M}	调制频偏@2Mbps		320	550	KHz
FCH_{250K}	频道间隔@250Kbps		1		MHz
FCH_{1M}	频道间隔@1Mbps		1		MHz
FCH_{2M}	频道间隔@2Mbps		2		MHz
发射模式指标					
PRF	典型输出功率	2	8	8	dBm
PRFC	输出功率范围	-35		8	dBm
PBW1	发射带数据调制的 20dB 带宽（250Kbps）		500		KHz
PBW2	发射带数据调制的 20dB 带宽（1Mbps）		1		MHz
PBW3	发射带数据调制的 20dB 带宽（2Mbps）		2		MHz
接收模式指标（注 1）					
RX_{max}	误码率<0.1%时的最大接收幅度		0		dBm
RXSENS1	接收灵敏度（0.1%BER）@250Kbps		-91		dBm
RXSENS2	接收灵敏度（0.1%BER）@1Mbps		-87		dBm
RXSENS3	接收灵敏度（0.1%BER）@2Mbps		-83		dBm
接收模式邻道选择性					
C/I_{CO}	同频的通道选择性@250kbps		2		dBc
C/I_{1ST}	第 1 相邻道选择性@250kbps		-8		dBc
C/I_{2ND}	第 2 相邻道选择性@250kbps		-18		dBc
C/I_{3RD}	第 3 相邻道选择性@250kbps		-24		dBc
C/I_{4TH}	第 4 相邻道选择性@250kbps		-28		dBc
C/I_{5TH}	第 5 相邻道选择性@250kbps		-32		dBc
C/I_{6TH}	第 6 相邻道选择性@250kbps		-35		dBc
C/I_{CO}	同频的通道选择性@1Mbps		10		dBc
C/I_{1ST}	第 1 相邻道选择性@1Mbps		1		dBc
C/I_{2ND}	第 2 相邻道选择性@1Mbps		-18		dBc
C/I_{3RD}	第 3 相邻道选择性@1Mbps		-23		dBc
C/I_{4TH}	第 4 相邻道选择性@1Mbps		-28		dBc
C/I_{5TH}	第 5 相邻道选择性@1Mbps		-32		dBc
C/I_{6TH}	第 6 相邻道选择性@1Mbps		-35		dBc
C/I_{CO}	同频的通道选择性@2Mbps		10		dBc

C/I_{1ST}	第 1 相邻道选择性@2Mbps		-6		dBc
C/I_{2ND}	第 2 相邻道选择性@2Mbps		-10		dBc
C/I_{3RD}	第 3 相邻道选择性@2Mbps		-22		dBc
C/I_{4TH}	第 4 相邻道选择性@2Mbps		-28		dBc
C/I_{5TH}	第 5 相邻道选择性@2Mbps		-34		dBc
操作条件					
VDD	供电电压	2.2	3	3.3	V
VSS	芯片地		0		V
V_{OH}	高电平输出电压	VDD-0.3		VDD	V
V_{OL}	低电平输出电压	VSS		VSS+0.3	V
V_{IH}	高电平输入电压	VDD-0.3		VDD	V
V_{IL}	低电平输入电压	VSS		VSS+0.3	V

*注 1：在晶振 16MHz 的整数倍（如 2416、2432MHz 等）的频道及相邻正负 1MHz 的频道的接收灵敏度退化 2dB；发射信号调制精度（EVM）退化 10%。
*注 2：250kbps 模式下发送数据长度最多 16 字节。

3. 极限最大额定值

表3-1 PAN2416AV芯片极限最大额定值

特　性	条件	参数值			单位
		最小	典型	最大	
最大额定值					
VDD	供电电压	-0.3		3.6	V
V_I	输入电压	-0.3		3.6	V
V_O	输出电压	VSS		VDD	
Pd	总功耗（TA=-40℃~85℃）			300	mW
T_{OP}	工作温度	-40		85	℃
T_{STG}	存储温度	-40		125	℃

*注 1：使用中强行超过一项或多项极限最大额定值会导致器件永久性损坏。
*注 2：静电敏感器件，操作时遵守防护规则。

4. 系统结构方框图

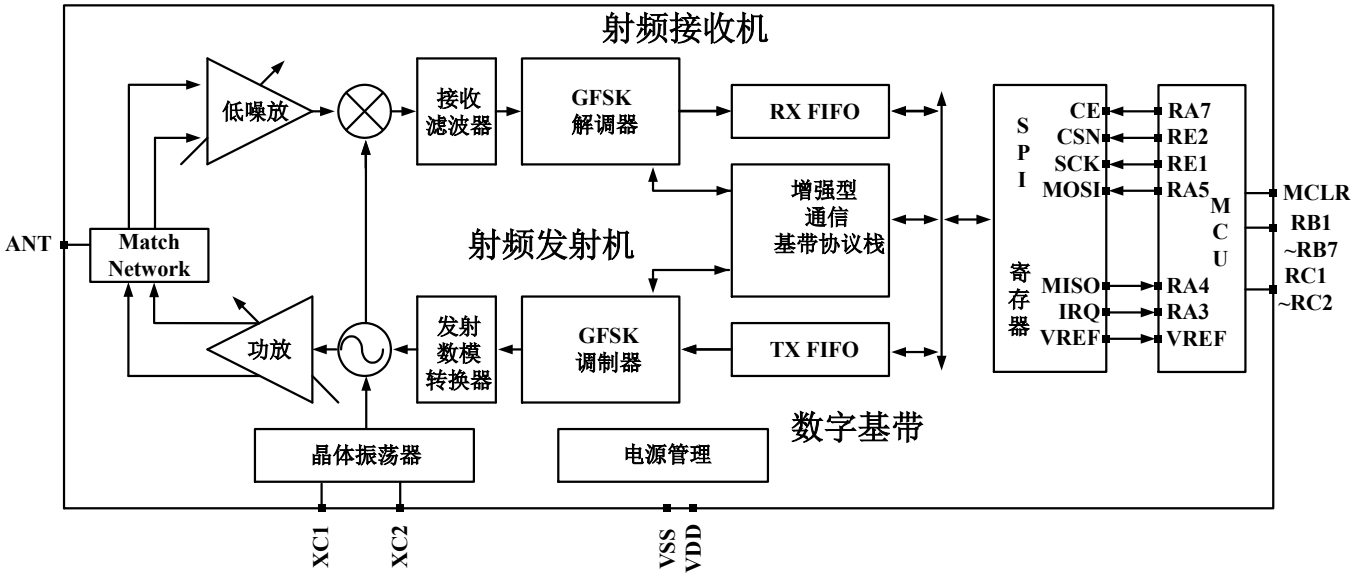


图4.1 PAN2416AV芯片系统结构方框图

5. 引脚定义

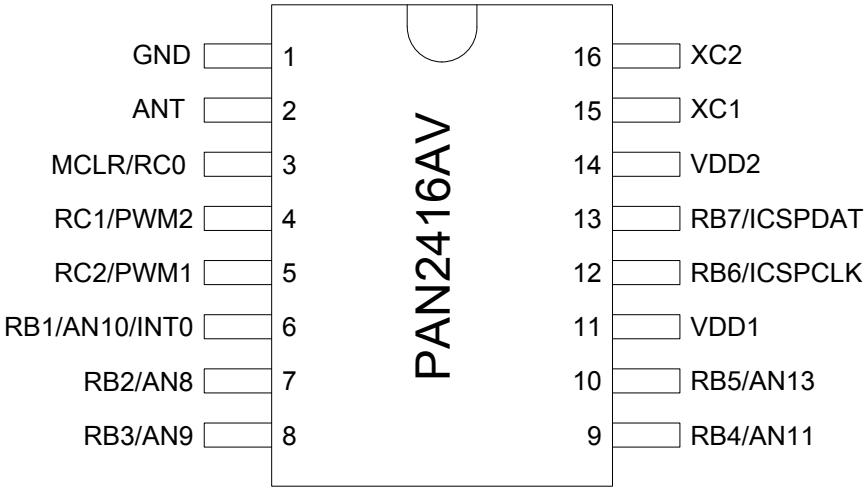


图5.1 PAN2416AV芯片引脚图

表5-1 引脚说明

引出端序号	符号	IO 类型	功能
1	GND	P	地（GND）
2	ANT	I/O	射频信号输入输出
3	RC0	I/O	可编程为输入脚，开漏输出脚
	MCLR	I	烧写高压输入脚
4	RC1	I/O	可编程为输入脚，带上拉电阻输入脚，推挽输出脚
	PWM2	O	PWM 输出
5	RC2	I/O	可编程为输入脚，带上拉电阻输入脚，推挽输出脚
	PWM1	O	PWM 输出
6	RB1	I/O	可编程为输入脚，带上拉电阻输入脚，推挽输出脚
	AN10	I	12 位 ADC 输入
	INT0	O	外部中断输入
7	RB2	I/O	可编程为输入脚，带上拉电阻输入脚，推挽输出脚
	AN8	I	12 位 ADC 输入
8	RB3	I/O	可编程为输入脚，带上拉电阻输入脚，推挽输出脚
	AN9	I	12 位 ADC 输入
9	RB4	I/O	可编程为输入脚，带上拉电阻输入脚，推挽输出脚
	AN11	I	12 位 ADC 输入
10	RB5	I/O	可编程为输入脚，带上拉电阻输入脚，推挽输出脚
	AN13	I	12 位 ADC 输入
11	VDD1	P	电源电压输入脚
12	RB6	I/O	可编程为输入脚，带上拉电阻输入脚，推挽输出脚
	ICSPCLK	I	编程时钟脚
13	RB7	I/O	可编程为输入脚，带上拉电阻输入脚，推挽输出脚
	ICSPDAT	I/O	编程数据脚
14	VDD2	P	电源电压输入脚
15	XC1	I	晶振输入
16	XC2	O	晶振输出

表5-2 RF与MCU连接引脚说明

接口功能	射频部分的接口名称	方向	MCU 部分的接口名称
模式片选信号	CE	→	RA7
SPI 片选信号	CSN	→	RE2
SPI 时钟信号	SCK	→	RE1
SPI 数据主端出从端入	MOSI	→	RA5
SPI 数据主端出从端入	MISO	←	RA4

中断信号	IRQ	←	RA3
基准电压	VREF	→	VREF

6. 芯片工作状态

本章描述PAN2416AV芯片的各种工作模式，以及用于控制芯片进入各工作模式的方法。PAN2416AV芯片自带状态机受控于芯片内部寄存器的配置值和外部引脚信号。

图6.1是PAN2416AV芯片工作状态图，表示5种工作模式之间的跳变。PAN2416AV芯片在VDD大于2.2V才开始正常工作。即使进入休眠模式，MCU还是可以通过SPI发送配置命令及CE管脚使芯片进入其它5种状态。

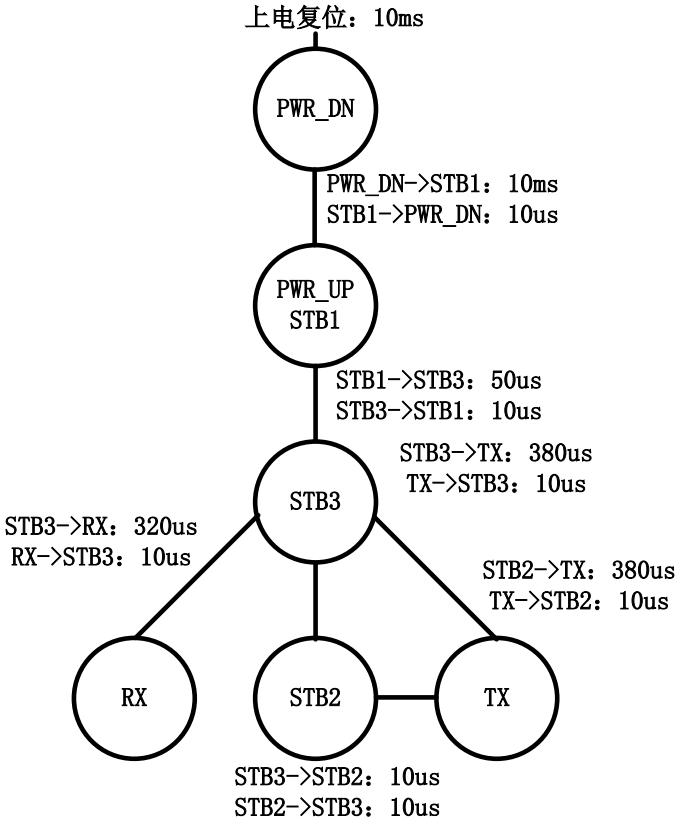


图6.1 工作状态图

控制信号和功能描述见表6-1。

表6-1 控制信号和功能描述

状态名	PWR_DN	STB1	STB3	STB2	RX	TX
控制信号						
PWR_UP	0	1	1	1	1	1
EN_PM	0	0	1	1	1	1
CE	0	0	0	1	1	1
PRIM_RX	X	X	X	0	1	0
功能描述						
SPI操作	√	√	√	√	√	√
保存reg值	√	√	√	√	√	√

晶振起振	X	√	√	√	√	√
晶振输出	X	X	X	√	√	√
电源管理模块使能	X	X	√	√	√	√
发射模块使能	X	X	X	X	X	√
接收模块使能	X	X	X	X	√	X

6.1 休眠模式

在休眠模式下，PAN2416AV芯片所有功能关闭，保持电流消耗最小。进入休眠模式后，PAN2416AV芯片停止工作，但寄存器内容保持不变。休眠模式由寄存器中PWR_UP位控制。

6.2 待机模式-I（STB1）

在待机模式-I下，芯片维持晶振振荡但不输出给其它模块，其余功能模块均关闭，消耗电流较小。在休眠模式下，通过配置寄存器PWR_UP的值为1，芯片即可进入待机模式-I。而处于发射或接收模式时，可以通过配置CE和EN_PM控制信号为0，芯片返回到待机模式-I。

6.3 待机模式-III（STB3）

在待机模式-I时，配置EN_PM控制信号为1，芯片进入到待机模式-III。待机模式-III主要目的是使得芯片的电源管理模块必须先于晶振输出。

6.4 待机模式-II（STB2）

发送端TX FIFO寄存器为空并且CE引脚置1，进入待机模式-II（待机模式-II通常可以理解为预备发射模式）。此时，晶振有较强的输出驱动能力且芯片的电源管理模块开启。待机模式-II下，如果有数据包送入TX FIFO，此时芯片内部锁相环立刻启动工作并且经过一段锁相环的锁定时间后，发射机将数据包发射出去。

6.5 接收模式

当PWR_UP、PRIM-RX、EN_PM、CE置1时，进入接收模式。

在RX模式下，射频部分接收从天线来的信号，将其放大、下变频、滤波和解调，根据地址、校验码、数据长度等，判断是否收包有效，有效收包上传RX FIFO，上报中断。如果RX FIFO是满的，接收的数据包就会被丢弃。

6.6 发射模式

当PWR_UP、EN_PM置1，PRIM-RX置0，CE置1，且TX FIFO中存在有效数据，进入发射模式。

PAN2416AV芯片在数据包发送完之前都会保持在发送模式。发送完成后，返回到待机模式。PAN2416AV芯片采用PLL开环发射方式，数据包是单包发送的。

7. 数据通信模式

PAN2416AV芯片搭配MCU来共同完成通信功能。链路层，如数据组帧、校验、地址判断、数据白化的扰码、数据重传和ACK响应等处理是由芯片内部完成的，无需MCU参与。

PAN2416AV芯片可配置为二个不同的RX FIFO 寄存器（32字节）或者一个RX FIFO寄存器（64字节）（6个接收通道共享）、二个不同的TX FIFO 寄存器（32字节）或者一个TX FIFO寄存器（64字节）。在休眠模式和待机模式下，MCU可以访问FIFO寄存器。

PAN2416AV芯片主要有二种数据通信模式：

不带自动重传不带ACK的通信模式（后简称为普通模式），发射端可以使用命令有W_TX_PAYLOAD, REUSE_TX_PL等；

带自动重传带ACK的通信模式（后简称为增强模式），发射端可以使用命令有W_TX_PAYLOAD, W_TX_PAYLOAD_NOACK, REUSE_TX_PL等；接收端可以使用命令有W_ACK_PAYLOAD等；

表7-1 普通模式

通信名称	普通模式	
通信方	PTX	PRX
特点	单向发送	单向接收
发送数据的组帧方式	I	无
开启REUSE_TX_PL命令	重复发送前一包数据	无

表7-2 增强模式

通信名称	增强模式	
通信方	PTX	PRX
特点	发送数据后，等待接收ACK	接收数据后，回发送ACK
发送数据的组帧方式	发送数据组帧方式II	回发送ACK组帧方式III
PTX使用REUSE_TX_PL命令	重复发送前一包数据	每收到一包，回发送ACK
PTX使用W_TX_PAYLOAD 命令 PRX使用W_ACK_PAYLOAD命令	发送数据后，等待接收ACK PAYLOAD	接收数据后，回发送ACK PAYLOAD，组帧方式II
PTX使用W_TX_PAYLOAD_NOACK命令	发送一次数据，不等ACK，组帧方式II	接收数据，不回ACK

7.1 普通模式

普通模式下，发送端从TX FIFO寄存器中取出数据并且发送，发送完成后上报中断（中断需要清除），同时TX FIFO寄存器清除该数据（TX FIFO需要清空）；接收端接收到有效的地址和数据时上报中断通知MCU，随后MCU可将该数据从RX FIFO寄存器中读出（TX FIFO和RX FIFO需要清空，中断需要清除）。

普通模式，（0X01）EN_AA寄存器置0X00，（0X04）SETUP_RETR寄存器置0X00，（0X1C）DYNPD寄存器置0X00，（0X1D）FEATURE寄存器的低3 bit置000。

7.2 增强模式

增强模式下，把主动发起通信的一方称为PTX（主发端），把接收数据并响应的一方称为PRX（主收端）。PTX发出数据后等待应答信号，PRX接收到有效数据后回应答信号。PTX规定时间内未收到应答信号，自动重新发送数据。自动重传和自动应答功能为PAN2416AV芯片自带，无需MCU参与。

PTX在发送数据后自动转到接收模式等待应答信号。如果没有在规定时间内收到正确的应答信号，PTX将重发相同的数据包，直到收到应答信号，或传输次数超过ARC的值（SETUP_RETR寄存器）产生MAX_RT中断。PTX收到应答信号，即认为数据已经发送成功（PRX收到有效数据），清除TX FIFO中的数据并产生

TX_DS中断（TX FIFO和RX FIFO需要清空，中断需要清除）。

PRX每次收到一包有效数据都会回ACK应答信号，该数据如果为新数据（PID值与上一包数据不同）保存到RX FIFO，否则就丢弃。

增强模式，需要保证PTX的TX地址（TX_ADDR）、通道0的RX地址（如RX_ADDR_P0），以及PRX的RX地址（如RX_ADDR_P5）三者相同。例：在图5中，PTX5对应PRX的数据通道5，地址设置如下：

PTX5：TX_ADDR=0xC2C3C4C5C1

PTX5：RX_ADDR_P0=0xC2C3C4C5C1

RX： RX_ADDR_P5=0xC2C3C4C5C1

增强模式有如下特征：

- 减少MCU的控制，简化软件操作；
- 抗干扰能力强，减少无线传输中因瞬间同频干扰造成的丢包，更易开发跳频算法；
- 重传过程中，减少MCU通过SPI接口的每次写入待发送数据的操作时间。

7.3 增强发送模式

1、CE置0，CONFIG寄存器的PRIM_RX位先置0。

2、当发送数据时，发送地址（TX_ADDR）和有效数据（TX_PLD）通过SPI接口按字节写入地址寄存器和TX FIFO。CSN引脚为低时，数据写入，CSN引脚再次为高，数据完成写入。

3、CE从0置1，启动发射（CE至少持续置1在30us以上，该操作生效）。

4、自动应答模式下（SETUP_RETR寄存器置不为0，ENAA_P0 =1），PTX发送完数据后立即自动将通道0切换到接收模式等待应答信号。如果在有效应答时间范围内收到ACK应答信号，则认为数据发送成功，状态寄存器的TX_DS位置1并自动清除TX FIFO中的数据。如果在设定时间范围内没有接收到应答信号，则自动重传数据。

5、如果自动传输计数器（ARC_CNT）溢出（超过了设定值），则状态寄存器的MAX_RT 位置1，不清除TX FIFO中的数据。当MAX_RT或TX_DS为1时，IRQ引脚产生低电平中断（需要使能相应中断）。中断可以通过写状态寄存器来复位。

6、数据包丢失计数器（PLOS_CNT）在每次产生MAX_RT中断后加一。自动传输计数器ARC_CNT统计重发数据包的次数；数据包丢失计数器PLOS_CNT统计在达到最大允许传输次数时仍没有发送成功的数据包个数。

7、产生MAX_RT或TX_DS中断后，系统进入待机模式。

7.4 增强接收模式

1、CE置0，CONFIG寄存器的PRIM_RX位先置1。准备接收数据的通道必须被使能（EN_RXADDR 寄存器），所有工作在增强型通信模式下的数据通道的自动应答功能是由EN_AA寄存器来使能的，有效数据宽度是由RX_PW_PX寄存器来设置的。

2、接收模式由设置CE为1启动。

3、预设的等待时间后，PRX开始检测无线信号。

4、接收到有效的数据包后，数据存储在RX_FIFO中，同时RX_DR位置1，产生中断。状态寄存器中RX_P_NO位显示数据是由哪个通道接收到的。

5、自动发送ACK应答信号。

6、如果CE保持为1，继续进入接收模式；如果CE置为0，则进入待机模式-III；

7、MCU以合适的速率通过SPI口将数据读出。

7.5 增强模式下的数据包识别

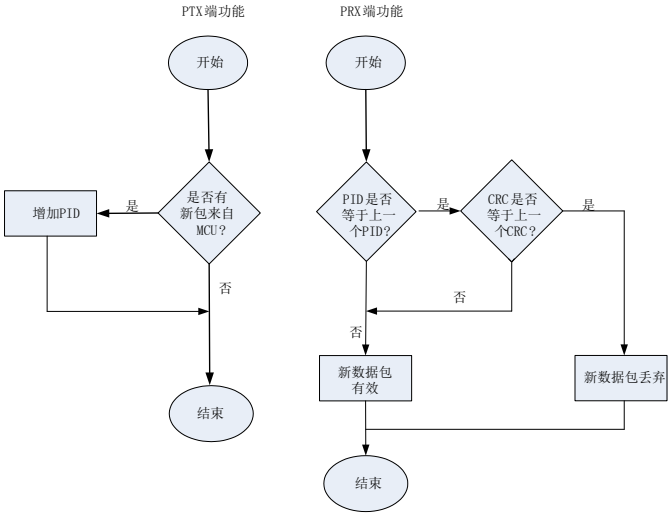


图7.1 PID生成和检测

每一包数据都包括两位的PID（数据包标志位），来帮助接收端识别该数据是新数据包还是重发的数据包，防止多次存入相同的数据包，PID的生成和检测如图7.1所示。发送端从MCU取得一包新数据后PID值加一。

7.6 增强模式下的 PTX 和 PRX 的时序图

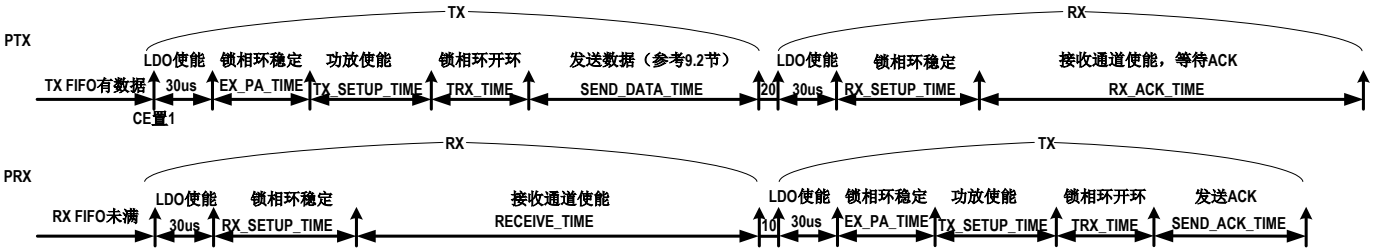


图7.2 增强模式下的PTX和PRX的时序图（发送成功）

如图 7.2 所示的是一次 PTX 和 PRX 通信的芯片内部时序图，使得通信成功必须满足以下两个条件：

- 条件1、PTX（或PRX）发射的锁相环稳定+功放使能+锁相环开环的三段时间之和，大于PRX（或PTX）接收的锁相环稳定时间20us以上，这样可以保证PTX（或PRX）发射数据的时间段落在PRX（或PTX）接收数据的时间段内，即：
  EX_PA_TIME + TX_SETUP_TIME + TRX_TIME > RX_SETUP_TIME +20us；
- 条件2、PRX发送ACK的锁相环稳定+功放使能+锁相环开环+发送ACK的四段时间之和，小于PTX接收的锁相环稳定+等待ACK的两端时间之和80us以上，保证PRX回复ACK的时间端落在PTX等待ACK的时间段内，各时间段的定义参考8章；发送ACK的时间参考9.2节为，发送帧比特数 ÷ 通信数据率，即：
  EX_PA_TIME + TX_SETUP_TIME + TRX_TIME + SEND_ACK_TIME <
  RX_SETUP_TIME + RX_ACK_TIME - 80us。

7.7 增强模式下的接收端一对多通信

PAN2416AV 芯片作为发射端，对于一对多通信，可以采用不同的地址与多个接收端进行通信。

PAN2416AV 芯片作为接收端，可以接收 6 路不同地址、相同频率的发送端数据。每个数据通道拥有自己的地址。

使能哪些数据通道是通过寄存器 EN_RXADDR 来设置的。每个数据通道的地址是通过寄存器 RX_ADDR_PX 来配置的。通常情况下不允许不同的数据通道设置完全相同的地址。如下，表 7.3 给出了一例多接收通道地址配置的示例。

表7-3 多通道地址设置

	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Data pipe 0(RX_ADDR_P0)	0xF1	0xD2	0xE6	0xA2	0x33
Data pipe 1(RX_ADDR_P1)	0xD3	0xD3	0xD3	0xD3	0xD3
	↓	↓	↓	↓	
Data pipe 2(RX_ADDR_P2)	0xD3	0xD3	0xD3	0xD3	0xD4
	↓	↓	↓	↓	
Data pipe 3(RX_ADDR_P3)	0xD3	0xD3	0xD3	0xD3	0xD5
	↓	↓	↓	↓	
Data pipe 4(RX_ADDR_P4)	0xD3	0xD3	0xD3	0xD3	0xD6
	↓	↓	↓	↓	
Data pipe 5(RX_ADDR_P5)	0xD3	0xD3	0xD3	0xD3	0xD7

从表 7.3 可以看出数据通道 0 的 5byte 总共 40 位的地址都是可配的；数据通道 1~5 的地址配置为 32 位共用地址（与数据通道 1 共用）+8 位各自的地址（最低字节）。

PAN2416AV 芯片在接收模式下可以与最多 6 路不同通道通信，如图 7.3 所示。每一个数据通道使用不同的地址，共用相同的频道。所有的发射端和接收端设置为增强模式。

PRX 在接收到有效数据后记录 PTX 的 TX 地址，并以此地址为目标地址发送应答信号。PTX 数据通道 0 被用做接收应答信号时，数据通道 0 的 RX 地址要与 TX 地址相等以确保接收到正确的应答信号。图 7.3 给出了 PTX 和 PRX 地址如何配置的例子。

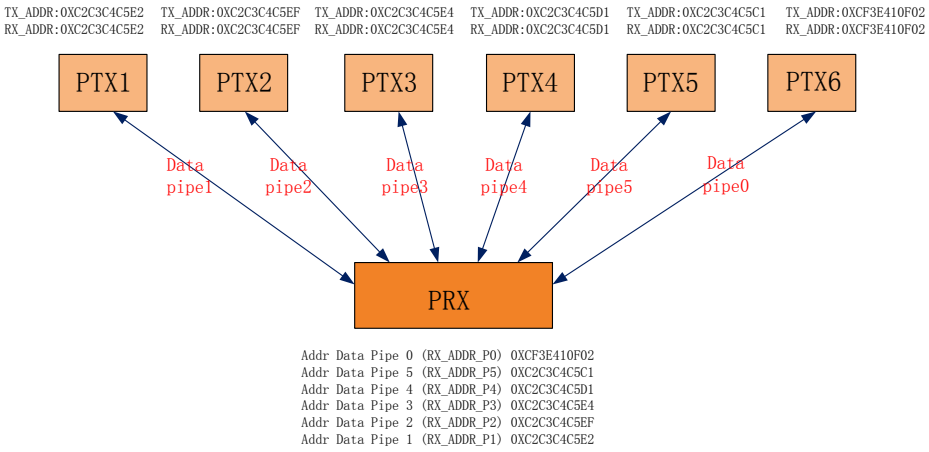


图7.3 多通道数据传输应答地址示例

7.8 DATA FIFO

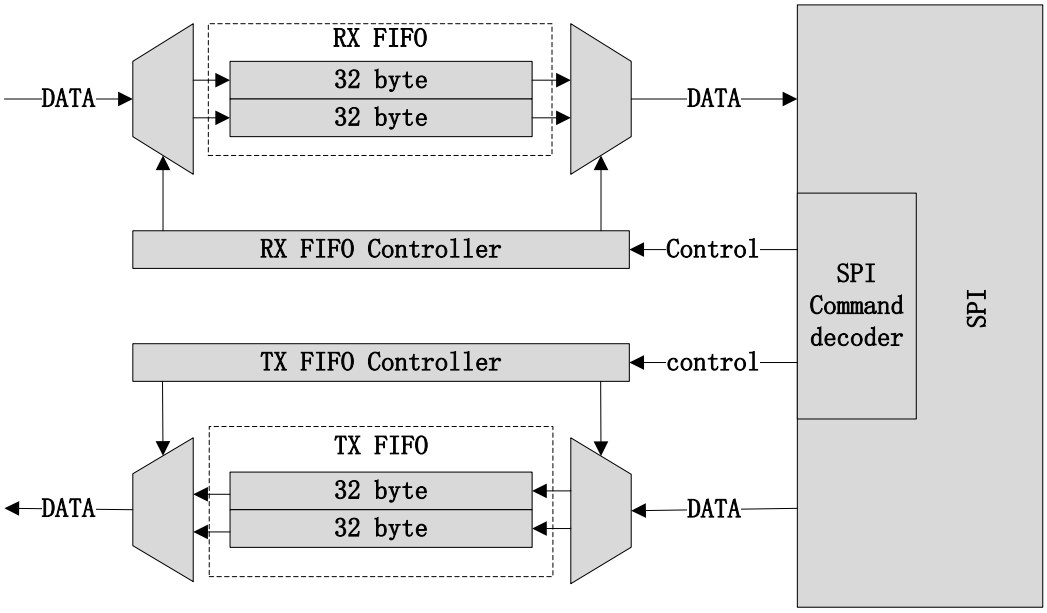


图7.4 FIFO 框图

PAN2416AV 芯片包含发 TX_FIFO,RX_FIFO。通过 SPI 命令可读写 FIFO。在发送模式下通过 W_TX_PAYLOAD 和 W_TX_PAYLOAD_NO_ACK 指令来写 TX_FIFO。如果产生 MAX_RT 中断，在 TX_FIFO 中的数据不会被清除。在接收模式下通过 R_RX_PAYLOAD 指令读取 RX_FIFO 中的 payload，R_RX_PL_WID 指令读取 payload 的长度。FIFO_STATUS 寄存器指示 FIFO 的状态。

7.9 中断引脚

PAN2416AV芯片的中断引脚（IRQ）为低电平触发， IRQ引脚初始状态为高电平，当状态寄存器中TX_DS、RX_DR或MAX_RT为1，以及相应的中断上报使能位为0时，IRQ引脚的中断触发。MCU给相应中断源写‘1’时，清除中断。IRQ引脚的中断触发可以被屏蔽或者使能，通过设置中断上报使能位为1，禁止IRQ引脚的中断触发。

8. SPI 控制接口

PAN2416AV芯片通过SPI控制接口对各寄存器进行读写操作。PAN2416AV芯片作为从机，SPI接口的数据率一般取决于MCU的接口速度，其最大的数据传输率为4Mbps。为了省电，在休眠模式和待机模式-Ⅰ，SPI的最大传输速率为1Mbps。

SPI接口是标准的SPI接口见表8-1，可以使用MCU的通用I/O口模拟SPI接口。CSN引脚为0时，SPI接口等待执行指令。一次CSN引脚由1到0的变化执行一条指令。在CSN引脚由1变0后可以通过MISO来读取状态寄存器的内容。

表8-1 SPI接口

引脚名称	I/O接口方向	SPI引脚的功能描述
CSN	输入	片选使能，低电平使能
SCK	输入	时钟
MOSI	输入	串行输入

MISO	输出	串行输出

表8-2 RF与MCU连接引脚说明

接口功能	射频部分的接口名称	射频部分的接口状态	MCU 部分的接口名称	MCU 部分的接口状态
模式片选信号	CE	输入	RA7	输出
SPI 片选信号	CSN	输入	RE2	输出
SPI 时钟信号	SCK	输入	RE1	输出
SPI 数据主端出从端入	MOSI	输入	RA5	输出
SPI 数据主端出从端入	MISO	输出	RA4	输入
中断信号	IRQ	输出	RA3	输入

注：表8-2中射频部分的接口状态为默认状态，且不可改变；MCU部分的接口状态需要通过软件来配置到上表模式，芯片才能正常工作。

8.1 SPI 指令格式

<命令字：由高位到低位（每字节）><数据字节：低字节到高字节，每一字节高位在前>

表8-3 SPI指令格式

命令名称	命令字（二进制）	后带数据（字节数）	操作
R_REGISTER	000A AAAA	1 to 5 低字节在前	读状态寄存器 AAAAA=5bit 寄存器地址
W_REGISTER	001A AAAA	1 to 5 低字节在前	写状态寄存器 AAAAA=5bit 寄存器地址 仅在休眠和待机模式-I 下执行。
R_RX_PAYLOAD	0110 0001	1 to 32/64 低字节在前	读接收数据，读操作通常由第 0 字节开始，读完过后数据将从 RX FIFO 中删除，接收模式下执行。
W_TX_PAYLOAD	1010 0000	1 to 32/64 低字节在前	写发射数据，写操作通常由 0 字节开始。
FLUSH_TX	1110 0001	0	清 TX FIFO。
FLUSH_RX	1110 0010	0	清 RX FIFO。
REUSE_TX_PL	1110 0011	0	用在 PTX 端，再次使用最后一帧发送的数据并且发送。该命令在刚发送数据并执行 FLUSH_TX 命令后可用。该命令不可以在发送数据的过程中使用。
ACTIVATE DEACTIVATE	0101 0000	1	用该命令后跟数据 0x73，将激活以下功能 • R_RX_PL_WID • W_TX_PAYLOAD_NOACK • W_ACK_PAYLOAD 该命令仅在休眠模式和待机模式下执行。 用该命令后跟数据 0x8C，将关闭上述功能。
R_RX_PL_WID	0110 0000	0	读 RX FIFO 最顶部 RX-payload 数据宽度。
W_ACK_PAYLOAD	1010 1PPP	1 to 64	Rx 模式可用

		低字节在前	写PIPE PPP（PPP 的值从000 到101）响应ACK 时同时回传的数据。最多可设置2个ACK 数据包。同PIPE 的数据将以先进先出的原则发送。 写操作通常从 0 字节开始。
W_TX_PAYLOAD_NOACK	1011 0000	1 to 32/64 低字节在前	写发射数据，写操作通常由 0 字节开始。TX 模式下执行，使用该命令发送数据不判自动应答。
CE_FSPI_ON	1111 1101	1	SPI 命令使 CE 内部逻辑置 1，用该命令后跟数据 0x00。
CE_FSPI_OFF	1111 1100	1	SPI 命令使 CE 内部逻辑置 0，用该命令后跟数据 0x00。
RST_FSPI_HOLD RST_FSPI_RELS	0101 0011	1	用该命令后跟数据 0x5A，使得进入复位状态并保持。 用该命令后跟数据 0xA5，使得释放复位状态并开始正常工作。
NOP	1111 1111	0	无操作。

R_REGISTER和W_REGISTER寄存器可能操作单字节或多字节寄存器。当访问多字节寄存器时首先要读/写的是最低字节的高位。对于多字节寄存器可以只写部分字节，没有写的高字节保持原有内容不变。例如：RX_ADDR_P0寄存器的最低字节可以通过写一个字节给寄存器RX_ADDR_P0来改变。

8.2 SPI 时序

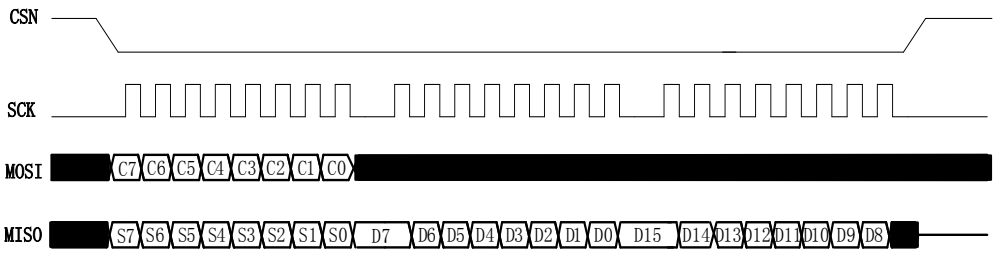


图8.1 SPI读操作

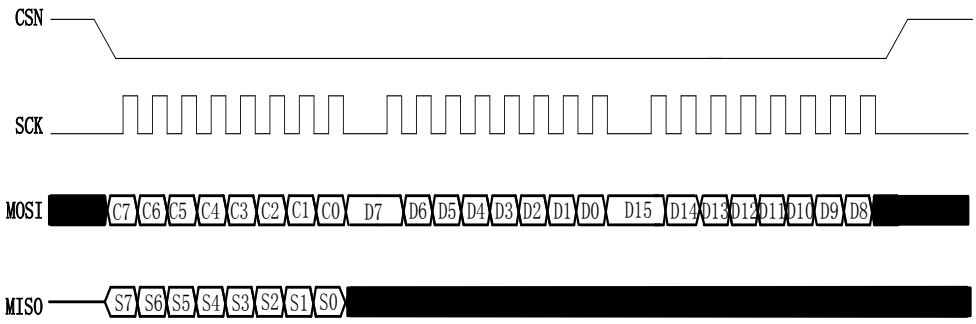


图8.2 SPI写操作

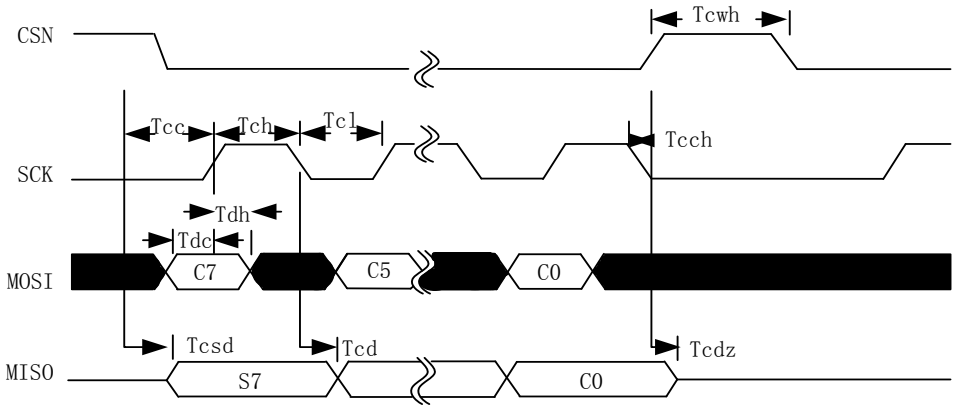


图8.3 SPI, NOP操作时序图

表8-4 SPI操作参考时间

SYMBOL	PARAMETERS	MIN	MAX	UNITS
Tdc	数据建立时间	15		ns
Tdh	数据保持时间	2		ns
Tcsd	CSN信号有效时间		40	ns
Tcd	SCK信号有效时间		51	ns
Tcl	SCK信号低电平时间	38		ns
Tch	SCK信号高电平时间	38		ns
Fsck	SCK信号频率		8	MHz
Tr,Tf	SCK信号上升下降时间		110	ns
Tcc	CSN信号建立时间	2		ns
Tcch	CSN信号保持时间	2		ns
Tcwh	CSN无效时间	49		ns
Tcdz	CSN信号高阻抗		40	ns

*注：表8-4的参数可根据选择的MCU进行调整

图8.1~8.3和表8-3给出了SPI操作及时序。在图中用到了下面的符号：

C_i -SPI指令位

S_i-状态寄存器位

D_i-数据位（备注：由低字节到高字节，每个字节中高位在前）

其中：$i = 1、2、3……n$ 。

9. 控制寄存器

可以通过SPI读写操作表9-1中的寄存器，来配置和控制PAN2416AV芯片。

表9-1 控制寄存器（地址打*的寄存器在使用中需要修改）

地址（HEX）	寄存器	BIT	复位后的默认值	读写	说明
00*	CONFIG				工作寄存器
	EN_PM	7	0	R/W	进入 STB3 模式 （前提 PWR_UP=1） 1: 进入 STB3 0: 进入 STB1 （在 STB3 模式下必须等待 50us 以上，

					才能跳转其它工作状态）
	MASK_RX_DR	6	0	R/W	接收数据成功的中断上报使能位 1: 中断不反映到 IRQ 引脚 0: RX_DR 中断反映到 IRQ 引脚
	MASK_TX_DS	5	0	R/W	发送数据成功的中断上报使能位 1: 中断不反映到 IRQ 引脚 0: TX_DS 中断反映到 IRQ 引脚
	MASK_MAX_RT	4	0	R/W	发送失败并达到最大传输次数的中断上报使能位 1: 中断不反映到 IRQ 引脚 0: MAX_RT 中断反映到 IRQ 引脚
	EN_CRC	3	1	R/W	CRC 使能位 1: CRC 使能，2byte 0: CRC 不使能，并且不判 CRC 校验
	N/A	2	0	R/W	保留，需要置 1
	PWR_UP	1	0	R/W	芯片使能位 1: POWER_UP 0: POWER_DOWN
	PRIM_RX	0	0	R/W	RX/TX 控制位 1: PRX 0: PTX
01	EN_AA Enhanced Burst				接收通道的自动应答使能 （接收端的 EN_AA 不为 0X00 时，为增强模式）
	Reserved	7:6	00	R/W	Only 00 allowed
	ENAA_P5	5	0	R/W	使能 pipe5 自动应答
	ENAA_P4	4	0	R/W	使能 pipe4 自动应答
	ENAA_P3	3	0	R/W	使能 pipe3 自动应答
	ENAA_P2	2	0	R/W	使能 pipe2 自动应答
	ENAA_P1	1	0	R/W	使能 pipe1 自动应答
	ENAA_P0	0	1	R/W	使能 pipe0 自动应答
02	EN_RXADDR				接收通道使能
	Reserved	7:6	00	R/W	Only 00 allowed
	ERX_P5	5	0	R/W	使能 data pipe 5
	ERX_P4	4	0	R/W	使能 data pipe 4
	ERX_P3	3	0	R/W	使能 data pipe 3
	ERX_P2	2	0	R/W	使能 data pipe 2
	ERX_P1	1	0	R/W	使能 data pipe 1
	ERX_P0	0	1	R/W	使能 data pipe 0
03	SETUP_AW				地址宽度设置
	Reserved	7:2	000000	R/W	Only 000000 allowed
	AW	1:0	11	R/W	RX/TX 地址宽度 00: 无效 01: 3 字节 10: 4 字节 11: 5 字节 如果地址宽度设置低于 5 字节，地址使用低字节
04	SETUP_RETR				自动传输设置

	ARD	7:4	0000	R/W	自动传输延时 0000 :250μs 0001 :500μs 0010 :750μs …… 1111: 4000μs
	ARC	3:0	0011	R/W	自动传输次数设置 0000: 普通模式 0001～1111: 增强模式 0001: 增强 1 次传输 0002: 增强 2 次传输 …… 1111: 增强 15 次传输
05	RF_CH				通信频道设置
	Reserved	7	0	R/W	Only 0 allowed
	RF_CH	6:0	1001110	R/W	设置使用频道为 Channel=RF_CH + 2400
06*	RF_SETUP				通信参数配置
	RF_DR	7:6	00	R/W	数据速率 01: 2Mbps 00: 1Mbps 11: 250kbps 10: 保留
	PA_GC	5:3	111	R/W	PA 的 driver 级输出幅度，可以调节发射功率大小 111: 幅度大 …… 000: 幅度小
	PA_PWR	2:0	111	R/W	PA 的输出级功率选择，可以调节发射功率大小 111: 输出功率大 …… 000: 输出功率小
07	STATUS				状态寄存器
	Reserved	7	0	R/W	Only 0 allowed
	RX_DR	6	0	R/W	RX FIFO 接收数据中断位， 在新数据被接收并到达 RX FIFO 时产生中断。 写 1 清中断
	TX_DS	5	0	R/W	TX FIFO 发送数据成功中断位， 在不带自动重传模式下，数据发送完成后产生中断； 在带自动重传模式下，仅在发送端收到 ACK 信号后才会将该位置高。 写 1 清中断
	MAX_RT	4	0	R/W	发送达到最大传输次数未成功中断位。 写 1 清中断 产生该中断后，继续进行通信必须先清该中断
	RX_P_NO	3:1	111	R	可从 RX_FIFO 读取的 pipe 号 000-101: pipe 号

					110: Not Used 111: RX_FIFO 空
	TX_FULL	0	0	R	TX FIFO 满标志 1: TX FIFO 满 0: TX FIFO 未满可用
08	OBSERVE_TX				传输状态寄存器
	PLOS_CNT	7:4	0	R	丢包计数器 该计数器达到最大值 15 时将停止计数， 该计数器在写 RF_CH 时被复位， 未复位该值时可以继续进行通信
	ARC_CNT	3:0	0	R	自动重传的传输次数计数器 传输加一次，ARC_CNT 加一； 在 ARC_CNT 达到 ARC 限定值时，视为丢包，并将 PLOS_CNT 加一； 当新数据写入 TX FIFO 时该计数器复位。
09*	DATAOUT				数据读取寄存器 （前提 DATAOUT_SEL=0）
	ANADATA7	7	0	R	接收机实时 RSSI 值的第 3 位（最高位） （测试用）
	ANADATA6	6	0	R	接收机实时 RSSI 值的第 2 位（测试用）
	ANADATA5	5	0	R	接收机实时 RSSI 值的第 1 位（测试用）
	ANADATA4	4	0	R	接收机实时 RSSI 值的第 0 位（测试用）
	ANADATA3	3	0	R	接收机成功收包的 RSSI 值的第 3 位（最高位）
	ANADATA2	2	0	R	接收机成功接收包的 RSSI 值的第 2 位
	ANADATA1	1	0	R	接收机成功接收包的 RSSI 值的第 1 位
	ANADATA0	0	0	R	接收机成功接收包的 RSSI 值的第 0 位
0A	RX_ADDR_P0	39:0	0xE7E7E7E7E7	R/W	data pipe 0 的接收地址，最长 5 字节。 （由低字开始写。地址长度由 SETUP_AW 定义）
0B	RX_ADDR_P1	39:0	0xC2C2C2C2C2	R/W	data pipe 1 的接收地址，最长 5 字节。 （由低字开始写。地址长度由 SETUP_AW 定义）
0C	RX_ADDR_P2	7:0	0xC3	R/W	data pipe 2 的接收地址，仅最低位， 高位等于 RX_ADDR_P1[39:8]
0D	RX_ADDR_P3	7:0	0xC4	R/W	data pipe 3 的接收地址，仅最低位， 高位等于 RX_ADDR_P1[39:8]
0E	RX_ADDR_P4	7:0	0xC5	R/W	data pipe 4 的接收地址，仅最低位， 高位等于 RX_ADDR_P1[39:8]
0F	RX_ADDR_P5	7:0	0xC6	R/W	data pipe 5 的接收地址，仅最低位， 高位等于 RX_ADDR_P1[39:8]
10	TX_ADDR	39:0	0xE7E7E7E7E7	R/W	发送端地址（由低字节开始写） 只能在配置为 PTX 模式的芯片中使用， 需要设置 RX_ADDR_P0 等于该地址以便接收 ACK 自动应答。
11	RX_PW_P0				data pipe 0 中的 RX payload 的数据长度
	Reserved	7	0	R/W	Only 0 allowed
	RX_PW_P0	6:0	0000000	R/W	data pipe 0 中的 RX payload 的数据长度

					（1 到 32/64 字节） 0: 该 Pipe 未用 1 = 1 byte … 32/64 = 32/64bytes
12	RX_PW_P1				data pipe 1 中的 RX payload 的数据长度
	Reserved	7	0	R/W	Only 0 allowed
	RX_PW_P1	6:0	0000000	R/W	data pipe 1 中的 RX payload 的数据长度 （1 到 32/64 字节） 0: 该 Pipe 未用 1 = 1 byte … 32/64 = 32/64 bytes
13	RX_PW_P2				data pipe 2 中的 RX payload 的数据长度
	Reserved	7	0	R/W	Only 0 allowed
	RX_PW_P2	6:0	0000000	R/W	data pipe 2 中的 RX payload 的数据长度 （1 到 32/64 字节） 0: 该 Pipe 未用 1 = 1 byte … 32/64 = 32/64 bytes
14	RX_PW_P3				data pipe 3 中的 RX payload 的数据长度
	Reserved	7	0	R/W	Only 0 allowed
	RX_PW_P3	6:0	0000000	R/W	data pipe 3 中的 RX payload 的数据长度 （1 到 32/64 字节） 0: 该 Pipe 未用 1 = 1 byte … 32/64 = 32/64 bytes
15	RX_PW_P4				data pipe 4 中的 RX payload 的数据长度
	Reserved	7	0	R/W	Only 0 allowed
	RX_PW_P4	6:0	0000000	R/W	data pipe 4 中的 RX payload 的数据长度 （1 到 32/64 字节） 0: 该 Pipe 未用 1 = 1 byte … 32/64 = 32/64 bytes
16	RX_PW_P5				data pipe 5 中的 RX payload 的数据长度
	Reserved	7	0	R/W	Only 0 allowed
	RX_PW_P5	6:0	0000000	R/W	data pipe 5 中的 RX payload 的数据长度 （1 到 32/64 字节） 0: 该 Pipe 未用 1 = 1 byte … 32/64 = 32/64 bytes
17*	FIFO_STATUS				FIFO 状态寄存器
	N/A	7	0	R	保留
	TX_REUSE	6	0	R	调用上一帧数据发送的指示位 在使用REUSE_TX_PL命令后，该位为1，重传上一次发送中最后一帧数据。该位可以由命令W_TX_PAYLOAD、W_TX_PAYLOAD_NOACK、

					DEACTIVATE、FLUSH TX进行复位操作。
	TX_FULL	5	0	R	TX FIFO 满标志位 1: TX FIFO 满 0: TX FIFO 可用
	TX_EMPTY	4	1	R	TX FIFO 空标志位 1: TX FIFO 空 0: TX FIFO 有数据
	N/A	3	0	R	保留
	N/A	2	0	R	保留
	RX_FULL	1	0	R	RX FIFO 满标志位 1: RX FIFO 满 0: RX FIFO 可用
	RX_EMPTY	0	1	R	RX FIFO 空标志位 1: RX FIFO 空 0: RX FIFO 有数据
N/A	TX_PLD	255:0	X	W	TX 发送数据 通过 SPI 命令写入 TX 数据，数据被存放在 2 级 32 字节或 1 级 64 字节 FIFO 中
N/A	RX_PLD	255:0	X	R	RX 接收数据 通过 SPI 命令读出 RX 数据，数据被存放在 2 级 32 字节或 1 级 64 字节 FIFO 中，所有 RX PIPE 共享同一个 FIFO
19*	DEMOD_CAL	7:0			调制解调参数寄存器 （可由方案需要来配置）
	CHIP	7	0	R/W	设置芯片是否进入测试模式 1: 进入测试模式 0: 退出测试模式
	CARR	6:5	00	R/W	设置芯片是否进入载波测试模式 11: 进入单载波测试模式，且 CHIP 置 1 00: 退出单载波模式
	GAUS_CAL	4:1	0111	R/W	高斯滤波器输出到 DAC 的信号幅度调整，该输出信号大小是发射调制频偏大小的决定因素之一 1111: 幅度较小 …. 1000: 幅度中等 …. 0000: 幅度较大
	Scramble_en	0	1	R/W	扰码功能是否使能，开启扰码功能可以对于待发送的数据进行白化操作，从而减少长 1 长 0 数据， 使能扰码功能需要收发两端进行相同配置 1: 使能扰码 0: 关闭扰码
1A*	RF_CAL2	47:0			补充射频寄存器 （一般使用默认值）
	IVCO_SEL<1:0>	47:46	01	R/W	VCO 的偏置电流选择

					00：IBG 0uA + IPTAT 55uA = 55uA 01：IBG 9uA + IPTAT 44uA = 53uA 10：IBG 18uA + IPTAT 33uA = 51uA 11：IBG 27uA + IPTAT 22uA = 49uA
	BW_500K	45	0	R/W	滤波器带宽选择 0：窄带宽 1：宽带宽
	GC_500K	44	1	R/W	滤波器增益选择 0：低增益 1：高增益
	NC	43:39	00000	R/W	Reserved
	PA_ramp_sel	38:37	01	R/W	选择 PA ramp up 的方式 00：No ramp up 01：4us ramp each step 10：从半电流开始 ramp 11：2us ramp each step
	TX_VCO_BIAS<2:0>	36:34	110	R/W	VCO 电流设置 000: 900uA 001: 1050uA 010: 1200uA 011: 1350uA 100: 1500uA 101: 1650uA 110: 1800uA 111: 1950uA
	NC	33	1	R/W	Reserved
	BPF_CTRL_BW	32	0	R/W	接收中频滤波器的 1dB 带宽选择 1: ×1 0: ×0.85
	BPF_CTRL_GAIN	31	1	R/W	接收中频滤波器增益控制 1: 5dB 0: 19dB
	VCOBUF_IC	30:29	01	R/W	VCO 驱动 MIXH 的驱动器电流选择 00: 600uA 01: 800uA 10: 1mA 11: 1.2mA
	VCO_CT	28:27	01	R/W	VCO 负载添加电容选择 00: 电容少，VCO 频率高 11: 电容多，VCO 频率低
	CAL_VREF_SEL	26	1	R/W	VCO 自动校正参考电压选择 1: 1.15V 0: 1.25V
	SPI_CAL_EN	25	0	R/W	VCO 单次触发自动校正过程 每次该位从 0 置 1 的过程都会触发一次 VCO 自动校正过程 此外，在改变工作频道和从待机进入收发状态下，也会触发 VCO 自动校正过程
	PREAMP_CTM	24:22	011	R/W	PA 的 driver 级的负载电容选择 000: 399fF 100: 171fF 111: 0fF

	DA_LPF_BW	21	1	R/W	DAC 的滤波带宽选择 1: 宽带 0: 窄带
	RX_CTM	20:19	01	R/W	LNA 的谐振频率（负载电容）选择， 00: 2.45GHz 01: 2.52GHz 10: 2.59GHz 11: 2.66GHz
	RCCAL_EN	18	1	R/W	接收带通滤波器的自动校正使能 1: 使能 0: 不使能
	EN_VCO_CAL	17	1	R/W	VCO 自动校正使能位 1: 使能 0: 不使能
	PRE_BC	16:14	100	R/W	预分频器直流电流选择 000: ×1 001&010: ×1.5 100&011: ×2 101&110: ×2.5 111: ×3
	VCO_CODE_IN	13:10	1000	R/W	VCO 频段选择位， 仅在 EN_VCO_CAL 为 0 时有效 1111: 高频段 0000: 低频段
	RCCAL_IN	9:4	010100	R/W	接收带通滤波器中频校正位设置， 仅在 RCCAL_EN 为 0 时有效 111111: 中频中心频率低 000000: 中频中心频率高
	CPSEL	3:2	01	R/W	锁相环电荷泵电流设置 RX TX 00: 26uA 26uA 01: 26uA 52uA 10: 52uA 78uA 11: 78uA 104uA
	DATAOUT_SEL	1	0	R/W	数据读取选择位，置 0
	RSSI_SEL	0	1	R/W	RSSI 信号采样点选择 1: 采样信号经过滤波器 0: 采样信号不经滤波器（测试用）
1B	DEM_CAL2	23:0			补充解调参数寄存器 （一般使用默认值）
	PIN	23:21	000	R/W	设置芯片进入测试模式后的输出 PIN（MISO 引脚/IRQ 引脚） 000（且 CHIP 为 0）为工作模式，作数据输出和中断输出 000（且 CHIP 为 1）为测试灵敏度模式，作解调数据和时钟输出 110（且 CHIP 为 1）为测试接收模式，作 limit I 和 Q 两路输出
	EN_RX	20	0	R/W	接收通道是否与锁相环同时开启 1: 同时打开

					0: 分时打开
	DELAY1	19	0	R/W	锁相环开环是否使能，锁相环使能开环状态可以作为发射的载波漂移测试 1: 锁相环使能开环 0: 锁相环开环受状态机控制
	DELAY0	18	0	R/W	解调器是否叠加收报的初始偏移量，解调器不叠加初始偏移量可作为接收灵敏度测试 1: 不叠加初始偏移量 0: 叠加初始频偏，接收状态下可以抵消由于中心频偏引起的误码
	TH1	17	1	R/W	在待机模式-II 下，LDO（除 DVDD 的 LDO 外）是否使能， 在测试模式下，测试发射单载波和接收灵敏度时该位置 1 1: 使能 0: 不使能
	PTH	16:13	0110	R/W	接收机数字解调器前导码相关阈值设置，24 位前导码的相关阈值=PTH+16 1000: 24 位 0110: 22 位 0000: 16 位
	SYNC_SEL	12	1	R/W	接收机数字解调器的 4 倍采样，取几点相关上计算该位数据正确 1: 3bit 0: 2bit
	DECOD_INV	11	1	R/W	前导码是否按位取反，一般置 1 使能该功能需要收发两端进行 1: 不按位取反 0: 按位取反
	GAIN1	10:7	1110	R/W	解调器的数据中心值调整环路的基准波形的幅度，置 1110
	GAIN2	6:1	000101	R/W	解调器的数据中心值调整环路的根据基准波形的调整速度，置 000101
	AGGRESSIVE	0	1	R/W	解调器的码率同步单元的速度选择 1: 大步长调整，速度快 0: 小步长调整，速度慢
1C	DYNPD				动态 PAYLOAD 长度使能
	Reserved	7:6	00	R/W	Only 00 allowed
	DPL_P5	5	0	R/W	使能 PIPE 5 动态 PAYLOAD 长度 (需要 EN_DPL 和 ENAA_P5)
	DPL_P4	4	0	R/W	使能 PIPE 4 动态 PAYLOAD 长度 (需要 EN_DPL 和 ENAA_P4)
	DPL_P3	3	0	R/W	使能 PIPE 3 动态 PAYLOAD 长度 (需要 EN_DPL 和 ENAA_P3)
	DPL_P2	2	0	R/W	使能 PIPE 2 动态 PAYLOAD 长度 (需要 EN_DPL 和 ENAA_P2)
	DPL_P1	1	0	R/W	使能 PIPE 1 动态 PAYLOAD 长度 (需要 EN_DPL 和 ENAA_P1)

	DPL_P0	0	0	R/W	使能 PIPE 0 动态 PAYLOAD 长度 (需要 EN_DPL 和 ENAA_P0)
1D*	FEATURE	7:0		R/W	特征寄存器
	Reserved	7	0	R/W	Only 00 allowed
	MUX_PA_IRQ	6	0	R/W	选择 IRQ 信号输出还是 EN_PA 信号输出到 PIN 0：IRQ 信号输出到 PIN 1：EN_PA 信号输出到 PIN
	CE_SEL	5	0	R/W	使能 CE 用命令方式开启 0：CE 由 CE 的引脚控制 1：CE 由命令方式控制
	DATA_LEN_SEL	4:3	00	R/W	数据长度选择 11: 64byte（512bit）模式 00: 32byte（256bit）模式
	EN_DPL	2	0	R/W	使能动态 PAYLOAD 长度
	EN_ACK_PAY	1	0	R/W	使能 ACK 带 PAYLOAD
	EN_NOACK	0	0	R/W	使能 W_TX_PAYLOAD_NOACK 命令
1E*	RF_CAL	23:0		R/W	射频参数寄存器 （可由方案需要来配置）
	OSC_IC	23	0	R/W	OSC 的激励电流选择 1：×1 0：×0.75
	DA_VREF_MB	22:20	101	R/W	DAC 的比较电路的正端参考电压 正端参考电压较大，DAC 输出幅度较大 111: 正端参考电压较大 000: 正端参考电压较小
	DA_VREF_LB	19:17	110	R/W	DAC 的比较电路的负端参考电压 负端参考电压较大，DAC 输出幅度较小 111: 负端参考电压较小 000: 负端参考电压较大
	DA_LPF_CTRL	16	1	R/W	DAC 的输出幅度控制位 1: 输出幅度×0.8 倍 0: 输出幅度×0.5 倍
	RSSI_EN	15	0	R/W	RSSI 使能位 1: RSSI 使能 0: RSSI 不使能
	RSSI_Gain_CTR	14:13	01	R/W	RSSI 的信号增益衰减的选择位 00: 不衰减 01: -6dB 10: -12dB 11: -18dB
	MIXL_GC	12	1	R/W	接收 MIXL 的增益选择 1: 14dB 0: 8dB
	POWPA_CTM<1:0>	11:10	11	R/W	PA 输出级的谐振电容选择 00: 大电容，谐振频点低 01: … 10: … 11: 小电容，谐振频点高

	LNA_GC	9:8	11	R/W	LNA 增益选择 11: 17dB 10: 11dB 01: 5.4dB 00: -0.4dB
	RX_VCO_BIAS<2:0>	7:5	111	R/W	RX 的 VCO 电流设置 000: 900uA 001: 1050uA 010: 1200uA 011: 1350uA 100: 1500uA 101: 1650uA 110: 1800uA 111: 1950uA
	RES_SEL	4:3	10		芯片偏置电流的负载选择 00: 26kR 01: 24kR 10: 22kR 11: 20kR
	LNA_HCURR	2	1	R/W	设置 LNA 高电流使能 1: 高电流 0: 低电流
	MIXL_BC	1	1	R/W	接收 MIXL 电流选择 1: ×1 0: ×0.5
	IB_BPF_TRIM	0	0	R/W	接收带通滤波器的电流选择 1: ×1 0: ×0.5
1F*	BB_CAL	7:0 15:8 23:16 31:24 39:32		R/W	数字基带参数寄存器 （一般使用默认值）
	Reserved	39:32	01000110	R/W	Only 0X01000110 allowed
	INVERTER	31	1	R/W	进入 RX_block 前是否取反 RX 通路数据 1: 取反 0: 保持不变
	DAC_MODE	30	0	R/W	dac_out[5:0]是否需要取反输出， dac_out[5:0]为 DAC 数据输入端 1:dac_out[5:0]<= [0:5] 0:dac_out[5:0]<= [5:0]
	DAC_BASAL	29:24	011100	R/W	预发送阶段的 DAC 数据输入的初始值
	TRX_TIME	23:21	011	R/W	锁相环开环到开始发射数据的时间间隔，时间长度计算： TRX_TIME×8+7.5，单位为 us
	EX_PA_TIME	20:16	00111	R/W	发射锁相环使能到 PA 使能的时间间隔，时间长度计算： EX_PA_TIME×16，单位为 us
	TX_SETUP_TIME	15:11	01101	R/W	发射 PA 使能到锁相环开环的时间间隔，时间长度计算： TX_SETUP_TIME×16，单位为 us
	RX_SETUP_TIME	10:6	10100	R/W	RX 射频通路锁相环稳定时间，

					时间长度计算： RX_SETUP_TIME×16，单位为 us
	RX_ACK_TIME	5:0	001010	R/W	PTX 转为接收模式后等待 ACK 的最长时间，超出该时间则认为本次传输失败， 2Mbps 模式下的时间长度计算： RX_ACK_TIME×16，单位为 us 1Mbps 模式下的时间长度计算： RX_ACK_TIME×32，单位为 us 250kbps 模式下的时间计算： RX_ACK_TIME×128，单位为 us

*注1：表9.1的配置为0X1B、0X1F寄存器的默认值可以工作；0X19、0X1A 、0X1E寄存器为需要配置的寄存器。

*注2：当访问多字节寄存器/地址/数据时，读/写顺序为低字节在前高字节在后。单个字节内部高bit在前低bit在后。

10. 数据包格式描述

10.1 普通模式的数据包形式

普通模式的数据包格式如表10-1所示，组帧方式 I。

表10-1 普通模式的数据包形式

前导码 （3字节）	地址 （3~5字节）	数据 （1~32/64字节）	CRC校验 （0/2字节）

表10-1中地址和数据部分可以选择扰码方式，根据使能/关闭扰码配置位。

10.2 增强模式的数据包形式

增强模式的数据包格式如表 10-2 所示，组帧方式 II。

表10-2 增强模式的数据包形式

前导码 （3字节）	地址 （3~5字节）	标识（10bit）			数据 （0~32/64字节）	CRC校验 （0/2字节）
		数据长度标识 （7bit）	PID标识 （2bit）	NO_ACK标识 （1bit）		

表 10-2 中地址、标识和数据部分可以选择扰码方式，根据使能/关闭扰码配置位。

10.3 增强模式的 ACK 包形式

增强模式的 ACK 包格式如表 10-3 所示，组帧方式 III。

表10-3 增强模式的数据包形式

前导码 （3字节）	地址 （3~5字节）	标识（10bit）			CRC校验 （0/2字节）
		数据长度标识 （7bit）	PID标识 （2bit）	NO_ACK标识 （1bit）	

表 10-3 中地址和标识部分需要选择与 PTX 相同的使能/关闭扰码方式。

11. MCU 寄存器

11.1 性能特性

- 内存
  OTP：4K×16Bit
  通用 RAM：176×8Bit
- 8 级堆栈缓存器
- 简洁实用的指令系统(68 条指令)
- 高精度 12 位 ADC
- 内置低压侦测电路
- 中断源
  3 个定时中断
  RB 口电平变化中断
  其它外设中断
- PWM 模块
- 工作电压范围：2.5V—3.3V@8MHz
  2.2V—3.3V@4MHz
  工作温度范围：-40℃—85℃
- 一种振荡方式
  内部 RC 振荡：设计频率 8MHz
- 指令周期（单指令或双指令）
- 内置 WDT 定时器
- 查表功能
- 定时器：
  8 位定时器 TIMER0，TIMER2
  16 位定时器 TIMER1

11.2 系统结构框图

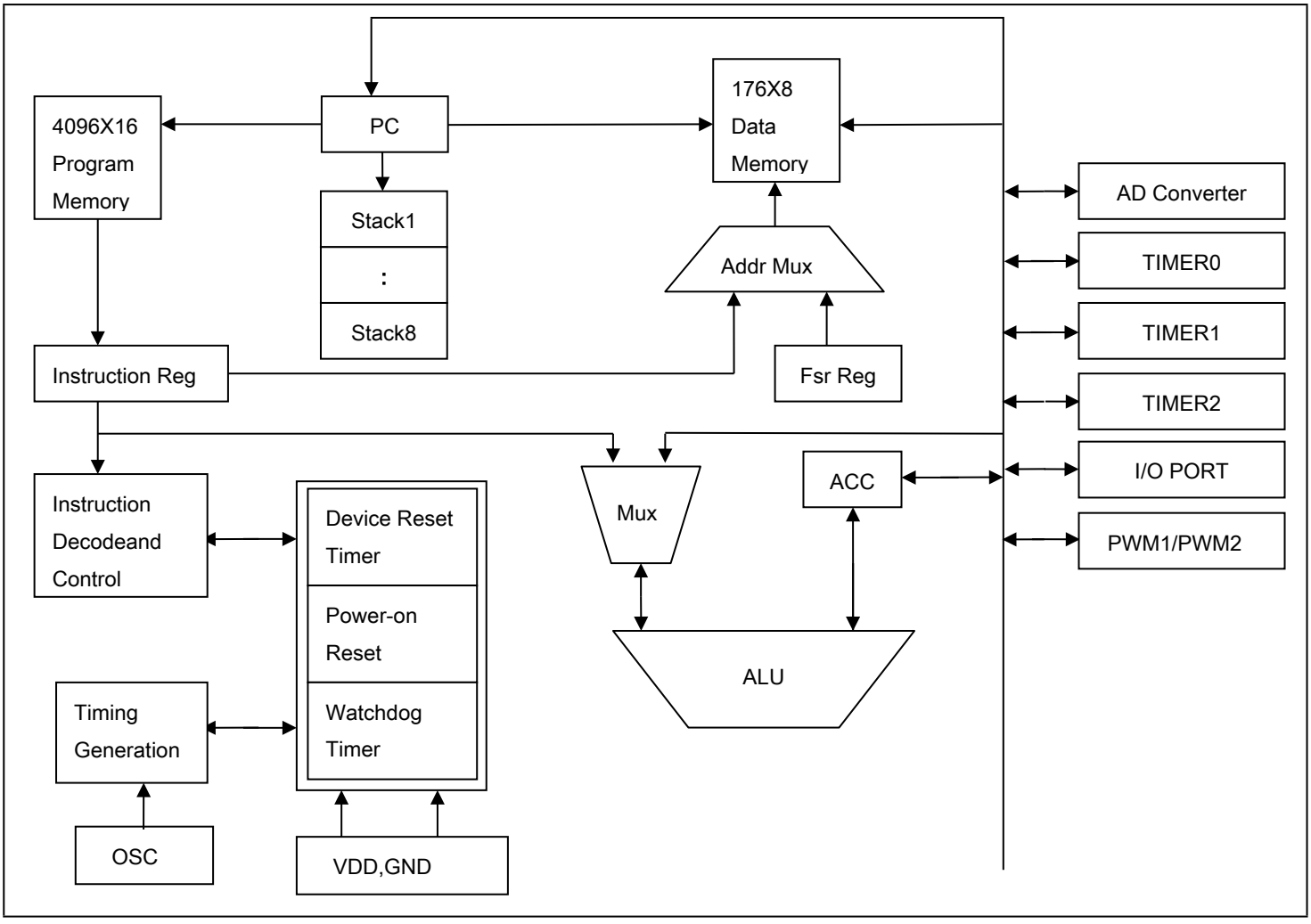


图11.2 MCU系统结构框图

11.3 系统配置寄存器

系统配置寄存器（CONFIG）是 MCU 初始条件的 OTP 选项。它只能被烧写器烧写，用户不能访问及操

作。它包含了以下内容：

1、OSC(振荡方式选择)

◆INTRC 内部 RC 振荡

2、WDT(看门狗选择)

◆ENABLE 打开看门狗定时器

◆DISABLE 关闭看门狗定时器

3、PROTECT(加密)

◆DISABLE OTP 代码不加密

◆ENABLE OTP 代码加密，加密后读出来的值将不确定

4、LVR_SEL(低压侦测电压选择)

◆1.8V

◆2.5V

11.4 在线串行编程

可在最终应用电路中对单片机进行串行编程。编程可以简单地通过以下 5 根线完成：

•电源线 VDD

•接地线 GND

•数据线 DAT

•时钟线 CLK

•VPP

这使用户可使用未编程的器件制造电路板，而仅在产品交付前才对单片机进行编程。从而可以将最新版本的固件或者定制固件烧写到单片机中。

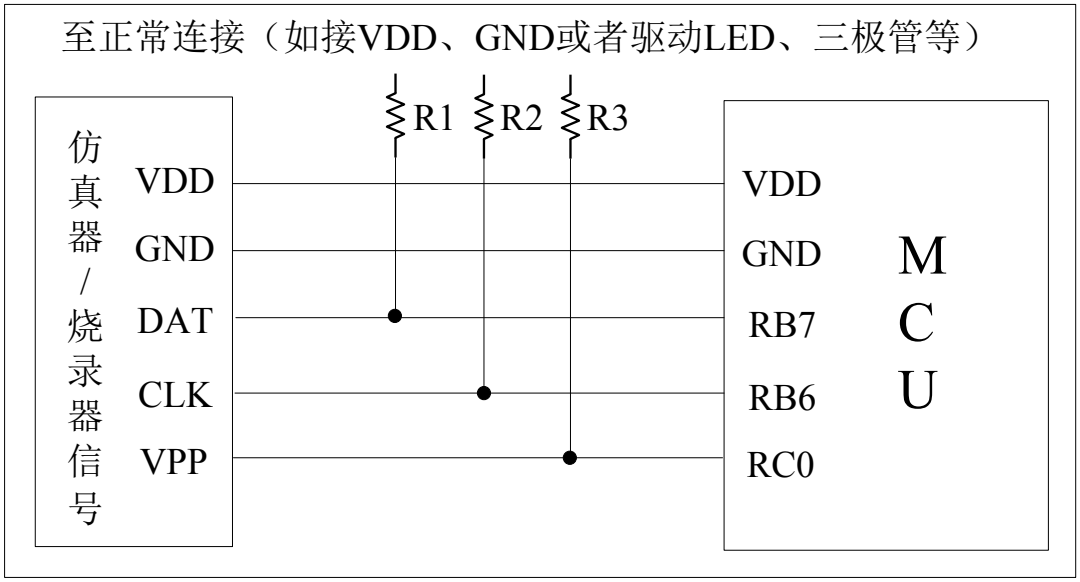


图11.2 典型的在线串行编程连接方

上图中，R1、R2 为电气隔离器件，常以电阻代替，其阻值如下：R1≥4.7K，R2≥4.7K，R3≥30K。VPP 口最好加滤波电容，建议为 0.1uF 的瓷片电容。

12. 中央处理器(CPU)

12.1 内存

12.1.1 程序内存

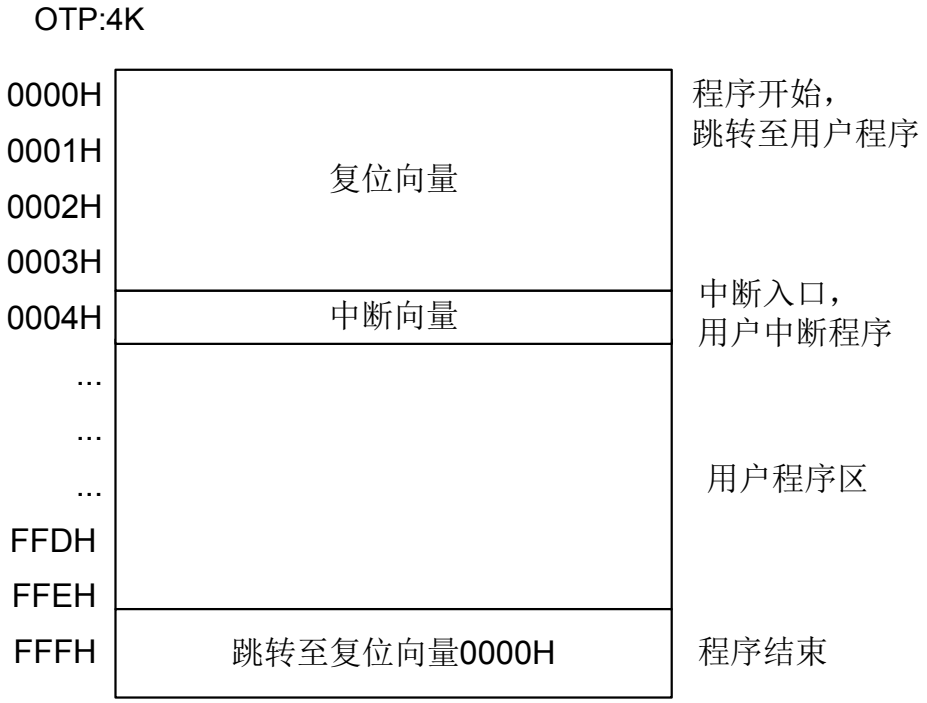


图12.1 PAN2416AV芯片程序存储器空间

12.1.1.1 复位向量(0000H)

单片机具有一个字长的系统复位向量（0000H）。具有以下三种复位方式：

◆上电复位

◆看门狗复位

◆低压复位（LVR）

发生上述任一种复位后，程序将从 0000H 处重新开始执行，系统寄存器也都将恢复为默认值。根据 STATUS 寄存器中的 PD 和 TO 标志位的内容可以判断系统复位方式。下面一段程序演示了如何定义 ROM 中的复位向量。

例：定义复位向量

```
        ORG     0000H       ;系统复位向量
        JP      START
        ORG     0010H       ;用户程序起始
START:
        …                   ;用户程序
        …
        END                 ;程序结束
```

12.1.1.2 中断向量

中断向量地址为 0004H。一旦有中断响应，程序计数器 PC 的当前值就会存入堆栈缓存器并跳转到 0004H 开始执行中断服务程序。所有中断都会进入 0004H 这个中断向量，具体执行哪个中断将由用户根据中断请求标志位寄存器的位决定。下面的示例程序说明了如何编写中断服务程序。

例：定义中断向量，中断程序放在用户程序之后

```
        ORG     0000H       ;系统复位向量
        JP      START
```

```
        ORG     0004H       ;用户程序起始
INT_START:
        CALL    PUSH        ;保存 ACC 跟 STATUS
        …                   ;用户中断程序
        …
INT_BACK:
        CALL    POP         ;返回 ACC 跟 STATUS
        RETI                ;中断返回
START:
        …                   ;用户程序
        …
        END                 ;程序结束
```

注：由于单片机并未提供专门的出栈、压栈指令，故用户需自己保护中断现场。

例：中断入口保护现场

```
PUSH:
        LD      ACC_BAK,A       ;保存 ACC 至自定义寄存器 ACC_BAK
        SWAPA   STATUS          ;状态寄存器 STATUS 高低半字节互换
        LD      STATUS_BAK,A    ;保存至自定义寄存器 STATUS_BAK
        RET                     ;返回
```

例：中断出口恢复现场

```
POP:
        SWAPA   STATUS_BAK      ;将保存至 STATUS_BAK 的数据高低半字节互换给 ACC
        LD      STATUS,A        ;将 ACC 的值给状态寄存器 STATUS
        SWAPR   ACC_BAK         ;将保存至 ACC_BAK 的数据高低半字节互换
        SWAPA   ACC_BAK         ;将保存至 ACC_BAK 的数据高低半字节互换给 ACC
        RET                     ;返回
```

12.1.1.3 查表

芯片具有查表功能，ROM 空间的任何地址都可做为查表使用。

相关指令：

·TABLE [R]把表格内容的低字节送给寄存器 R，高字节送到寄存器 TABLE_DATAH。

·TABLEA 把表格内容的低字节送给累加器 ACC，高字节送到寄存器 TABLE_DATAH。

相关寄存器：

·TABLE_SPH 可读写寄存器，用来指明表格高 5 位地址。

·TABLE_SPL 可读写寄存器，用来指明表格低 8 位地址。

·TABLE_DATAH 只读寄存器，存放表格高字节内容。

注：在查表之前要先把表格地址写入 TABLE_SPH 和 TABLE_SPL 中。如果主程序和中断服务程序都用到查表指令，主程序中的 TABLE_SPH 的值可能会因为中断中执行的查表指令而发生变化，产生错误。也就是说要避免在主程序和中断服务程序中都使用查表指令。但如果必须这样做的话，我们可以在查表指令前先将中断禁止，在查表结束后再开放中断，以避免发生错误。

表格调用的流程如下图：

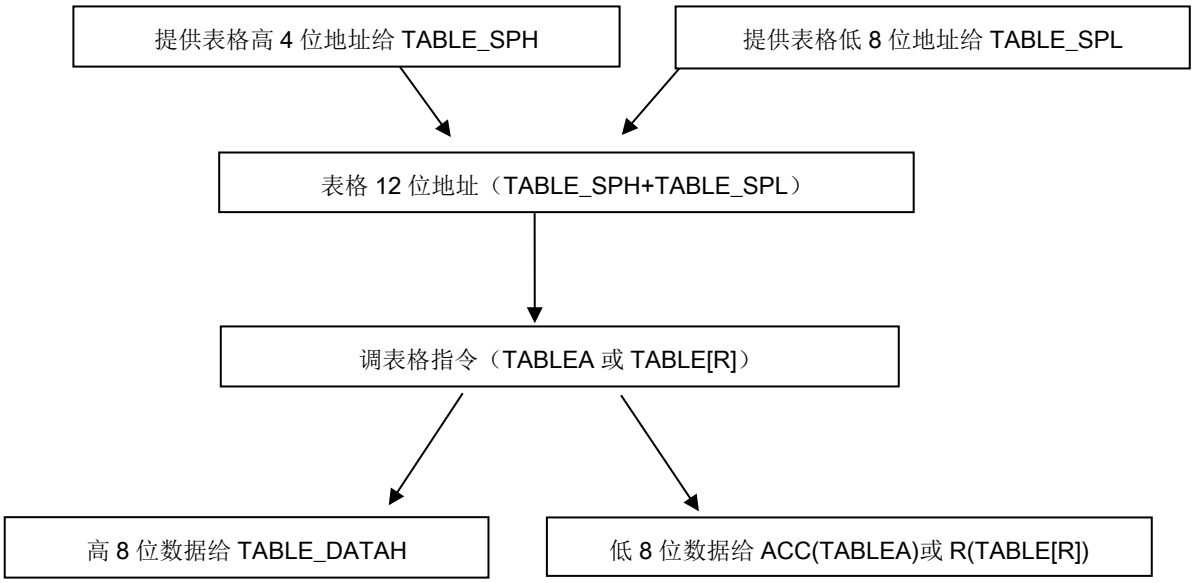


图 12.4 调用流程图

下面例子给出了如何在程序中调用表格。

```
…                                   ;上接用户程序
LDIA        02H                     ;表格低位地址
LD          TABLE_SPL,A
LDIA        06H                     ;表格高位地址
LD          TABLE_SPH,A
TABLE       R01                     ;表格指令，将表格低 8 位(56H)给自定义寄存器 R01
LD          A,TABLE_DATAH           ;将查表结果的高 8 位(34H)给累加器 ACC
LD          R02,A                   ;将 ACC 值(34H)给自定义寄存器 R02
…                                   ;用户程序

ORG         0600H                   ;表格起始地址
DW          1234H                   ;0600H 地址表格内容
DW          2345H                   ;0601H 地址表格内容
DW          3456H                   ;0602H 地址表格内容
DW          0000H                   ;0603H 地址表格内容
```

12.1.1.4 跳转表

跳转表能够实现多地址跳转功能。由于 PCL 和 ACC 的值相加即可得到新的 PCL，因此，可以通过对 PCL 加上不同的 ACC 值来实现多地址跳转。ACC 值若为 n，PCL+ACC 即表示当前地址加 n，执行完当前指令后 PCL 值还会自加 1，可参考以下范例。如果 PCL＋ACC 后发生溢出，PC 不会自动进位，故编写程序时应注意。这样，用户就可以通过修改 ACC 的值轻松实现多地址的跳转。

PCLATH 为 PC 高位缓冲寄存器，对 PCL 操作时，必须先对 PCLATH 进行赋值。

例：正确的多地址跳转程序示例

```
    OTP 地址
                LDIA        01H
                LD          PCLATH,A        ;必须对 PCLATH 进行赋值
                …
0110H:          ADDR        PCL             ;ACC+PCL
0111H:          JP          LOOP1           ;ACC=0，跳转至 LOOP1
0112H:          JP          LOOP2           ;ACC=1，跳转至 LOOP2
0113H:          JP          LOOP3           ;ACC=2，跳转至 LOOP3
0114H:          JP          LOOP4           ;ACC=3，跳转至 LOOP4
0115H:          JP          LOOP5           ;ACC=4，跳转至 LOOP5
0116H:          JP          LOOP6           ;ACC=5，跳转至 LOOP6
```

例：错误的多地址跳转程序示例

```
    OTP 地址
                CLR         PCLATH
                …
00FCH:          ADDR        PCL             ;ACC+PCL
00FDH:          JP          LOOP1           ;ACC=0，跳转至 LOOP1
00FEH:          JP          LOOP2           ;ACC=1，跳转至 LOOP2
00FFH:          JP          LOOP3           ;ACC=2，跳转至 LOOP3
0100H:          JP          LOOP4           ;ACC=3，跳转至 0000H 地址
0101H:          JP          LOOP5           ;ACC=4，跳转至 0001H 地址
0102H:          JP          LOOP6           ;ACC=5，跳转至 0002H 地址
```

注：由于 PCL 溢出不会自动向高位进位，故在利用 PCL 作多地址跳转时，一定要注意该段程序一定不能放在 OTP 空间的分页处。

12.1.2 数据存储器

	地址		地址		地址		地址
INDF	00H	INDF	80H	INDF	100H	INDF	180H
TMR0	01H	OPTION_REG	81H	TMR0	101H	OPTION_REG	181H
PCL	02H	PCL	82H	PCL	102H	PCL	182H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
FSR	04H	FSR	84H	FSR	104H	FSR	184H
PORTA	05H	TRISA	85H	WDTCON	105H	----	185H
PORTB	06H	TRISB	86H	PORTB	106H	TRISB	186H
PORTC	07H	TRISC	87H	----	107H	----	187H
----	08H	----	88H	----	108H	----	188H
PORTE	09H	TRISE	89H	----	109H	----	189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
PIR1	0CH	PIE1	8CH	----	10CH	----	18CH
----	0DH	----	8DH	----	10DH	----	18DH
TMR1L	0EH	----	8EH	----	10EH	WPUA	18EH
TMR1H	0FH	OSCCON	8FH	----	10FH	WPUC	18FH
T1CON	10H	OSCTUNE	90H	TABLE_SPH	110H	----	190H
TMR2	11H	----	91H	TABLE_SPL	111H	----	191H
T2CON	12H	PR2	92H	TABLE_DATAH	112H	----	192H
----	13H	----	93H	----	113H	----	193H
----	14H	----	94H	----	114H	----	194H
CCPR1L	15H	WPUB	95H	----	115H	----	195H
CCPR1H	16H	IOCB	96H	----	116H	----	196H
CCP1CON	17H	----	97H	----	117H	----	197H
----	18H	----	98H	----	118H		198H
----	19H	----	99H	----	119H		199H
----	1AH	----	9AH	WPUE	11AH		19AH
CCPR2L	1BH	----	9BH	----	11BH	----	19BH
CCPR2H	1CH	----	9CH	----	11CH		19CH
CCP2CON	1DH	----	9DH	----	11DH		19DH
ADRESH	1EH	ADRESL	9EH	----	11EH		19EH
ADCON0	1FH	ADCON1	9FH	----	11FH		19FH
通用寄存器 96 字节	20H	通用寄存器 80 字节	A0H	----	120H	----	1A0H
	6FH		EFH		16FH		1EFH
	70H -- 7FH	快速存储区 70H-7FH	F0H -- FFH	快速存储区 70H-7FH	170H -- 17FH	快速存储区 70H-7FH	1F0H -- 1FFH
BANK0		BANK1		BANK2		BANK3	

图12.5 PAN2416AV芯片数据存储器列表

数据存储器由 512×8 位组成，分为两个功能区间：特殊功能寄存器和通用数据存储器。数据存储器单元大多数是可读/写的，但有些只读的。特殊功能寄存器地址为从 00H-1FH，80-9FH，100-11FH，180-197H。

表 12-1 MCU 特殊功能寄存器汇总 Bank0

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
00H	INDF	寻址该单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
01H	TMR0	TIMER0数据寄存器								xxxxxxxx
02H	PCL	程序计数器低字节								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	间接数据存储器地址指针								xxxxxxxx
05H	PORTA	RA7	----	RA5	RA4	RA3	----	----	----	x-xxx---
06H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	----	xxxxxxx-
07H	PORTC	----	----	----	----	----	RC2	RC1	RC0	-----xxx
09H	PORTE	----	----	----	----	----	RE2	RE1	----	-----xx-
0AH	PCLATH	----	---	----	----	程序计数器高4位的写缓冲器				----0000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	PIR1	----	ADIF	----	----	----	----	TMR2IF	TMR1IF	-0---000
0EH	TMR1L	16位TIMER1寄存器低字节的数据寄存器								xxxxxxxx
0FH	TMR1H	16位TIMER1寄存器高字节的数据寄存器								xxxxxxxx
10H	T1CON	----	----	T1CKPS1	T1CKPS0	----	----	----	TMR1ON	--00---0
11H	TMR2	TIMER2模块寄存器								00000000
12H	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-0000000
15H	CCPR1L	PWM寄存器1的低字节								xxxxxxxx
16H	CCPR1H	PWM寄存器1的高字节								xxxxxxxx
17H	CCP1CON	----	----	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--000000
1BH	CCPR2L	PWM寄存器2的低字节								xxxxxxxx
1CH	CCPR2H	PWM寄存器2的高字节								xxxxxxxx
1DH	CCP2CON	----	----	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--000000
1EH	ADRESH	A/D结果寄存器的高字节								xxxxxxxx
1FH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	00000000

表 12-2 MCU 特殊功能寄存器汇总 Bank1

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
80H	INDF	寻址该地址单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
81H	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	11111111
82H	PCL	程序计数器（PC）的低字节								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	间接数据存储器地址指针								xxxxxxxx
85H	TRISA	----	----	----	TRISA4	TRISA3	TRISA2	----	----	---111--
86H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	----	1111111-
87H	TRISC	----	----	----	----	----	TRISC2	TRISC1	----	-----11-
89H	TRISE	----	----	----	----	----	TRISE2	TRISE1	----	-----11-
8AH	PCLATH	----	----	----	----	程序计数器高4位的写缓冲器				----0000
8BH	INTCON	GIE	PEIE	T01E	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	PIE1	----	ADIE	----	----	----	----	TMR2IE	TMR1IE	-0----00
8FH	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	SCS	-110---0
90H	OSCTUNE	----	----	----	TUN4	TUN3	TUN2	TUN1	TUN0	---00000
92H	PR2	TIMER2周期寄存器								---11111
95H	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	----	0000000-
96H	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	----	0000000-
9EH	ADRESL	A/D结果寄存器的低字节								xxxxxxxx
9FH	ADCON1	ADFM	----	----	----	----	----	----	----	0-------

表 12-3 MCU 特殊功能寄存器汇总 Bank2

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
100H	INDF	寻址该地址单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
101H	TMR0	TIMER0模块寄存器								xxxxxxxx
102H	PCL	程序计数器（PC）的低字节								00000000
103H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
104H	FSR	间接数据存储器地址指针								xxxxxxxx
105H	WDTCON	----	----	----	----	----	----	----	SWDTEN	-------0
106H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	----	xxxxxxx-
10AH	PCLATH	----	----	---	程序计数器高5位的写缓冲器					---00000
10BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
110H	TABLE_SPH	表格高位指针								---xxxxx
111H	TABLE_SPL	表格低位指针								xxxxxxxx
112H	TABLE_DATAH	表格高位数据								xxxxxxxx

表12-4 MCU特殊功能寄存器汇总Bank3

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
180H	INDF	寻址该地址单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
181H	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	11111111
182H	PCL	程序计（PC）的低字节								00000000
183H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
184H	FSR	间接数据存储器地址指针								xxxxxxxx
186H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB73	TRISB2	TRISB1	----	1111111-
18AH	PCLATH	----	----	----	----	程序计数器高4位的写缓冲器				----0000
18BH	INTCON	GIE	PEIE	T01E	INTE	RBIE	T01F	INTF	RBIF	00000000

12.2 寻址方式

12.2.1 直接寻址

通过工作寄存器（ACC）来对 RAM 进行操作。

例：ACC 的值送给 30H 寄存器

```
LD          30H,A
```

例：30H 寄存器的值送给 ACC

```
LD          A,30H
```

12.2.2 立即寻址

把立即数传给工作寄存器（ACC）

例：立即数 12H 送给 ACC

```
LDIA        12H
```

12.2.3 间接寻址

数据存储器能被直接或间接寻址。通过 INDF 寄存器可间接寻址，INDF 不是物理寄存器。当对 INDF 进行存取时，它会根据 FSR 寄存器内的值（低 8 位）和 STATUS 寄存器的 IRP 位（第 9 位）作为地址，并指向该地址的寄存器，因此在设置了 FSR 寄存器和 STATUS 寄存器的 IRP 位后，就可把 INDF 寄存器当作目的寄存器来存取。间接读取 INDF（FSR=0）将产生 00H。间接写入 INDF 寄存器，将导致一个空操作。以下例子说明了程序中间接寻址的用法。

例：FSR 及 INDF 的应用

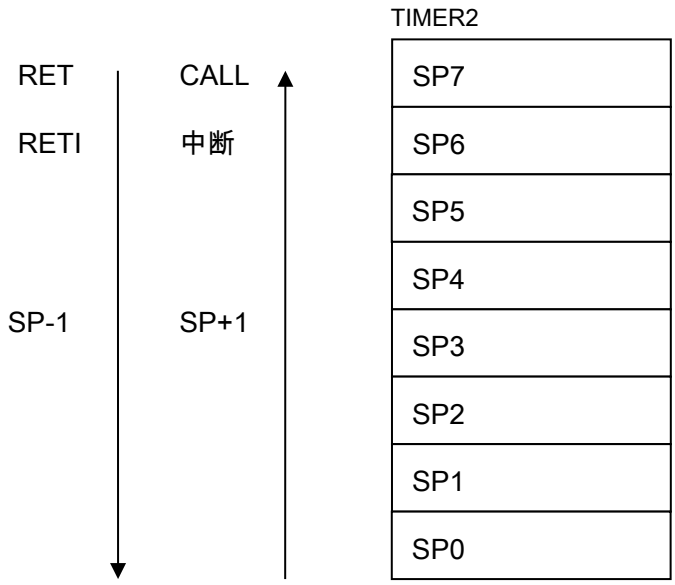
```
        LDIA    30H
        LD      FSR,A       ;间接寻址指针指向 30H
        CLRB    STATUS,IRP  ;指针第 9 位清零
        CLR     INDF        ;清零 INDF 实际是清零 FSR 指向的 30H 地址 RAM
```

例：间接寻址清 RAM(20H-7FH)举例：

```
        LDIA    1FH
        LD      FSR,A       ;间接寻址指针指向 1FH
        CLRB    STATUS,IRP
LOOP:
        INCR    FSR         ;地址加 1，初始地址为 30H
        CLR     INDF        ;清零 FSR 所指向的地址
        LDIA    7FH
        SUBA    FSR
        SNZB    STATUS,C    ;一直清零至 FSR 地址为 7FH
        JP      LOOP
```

12.3 堆栈

芯片的堆栈缓存器共 8 层，堆栈缓存器既不是数据存储器的一部分，也不是程序内存的一部分，且既不能被读出，也不能被写入。对它的操作通过堆栈指针（SP）来实现，堆栈指针（SP）也不能读出或写入，当系统复位后堆栈指针会指向堆栈顶部。当发生子程序调用及中断时的程序计数器（PC）值被压入堆栈缓存器，当从中断或子程序返回时将数值返回给程序计数器（PC）,下图说明其工作原理。



堆栈缓存器的使用将遵循一个原则“先进后出”。

注：堆栈缓存器只有 8 层，如果堆栈已满，并且发生不可屏蔽的中断，那么只有中断标志位会被记录下来，而中断响应则会被抑制，直到堆栈指针发生递减，中断才会被响应，这个功能可以防止中断使堆栈溢出，同样如果堆栈已满，并且发生子程序调用，那么堆栈将会发生溢出，首先进入堆栈的内容将会丢失，只有最后 8 个返回地址被保留，故用户在写程序时应注意此点，以免发生程序走飞。

12.4 工作寄存器（ACC）

12.4.1 概述

ALU 是 8Bit 宽的算术逻辑单元，MCU 所有的数学、逻辑运算均通过它来完成。它可以对数据进行加、减、移位及逻辑运算；ALU 也控制状态位（STATUS 状态寄存器中），用来表示运算结果的状态。

ACC 寄存器是一个 8Bit 的寄存器，ALU 的运算结果可以存放在此，它并不属于数据存储器的一部分而是位于 CPU 中供 ALU 在运算中使用，因此不能被寻址，只能通过所提供的指令来使用。

12.4.2 ACC 应用

例：用 ACC 做数据传送

```
        LD      A,R01       ;将寄存器 R01 的值赋给 ACC
        LD      R02,A       ;将 ACC 的值赋给寄存器 R02
```

例：用 ACC 做立即寻址目标操作数

```
        LDIA    30H         ;给 ACC 赋值 30H
        ANDIA   30H         ;将当前 ACC 的值跟立即数 30H 进行“与”操作，
                            ;结果放入 ACC
        XORIA   30H         ;将当前 ACC 的值跟立即数 30H 进行“异或”操作，
                            ;结果放入 ACC
```

例：用 ACC 做双操作数指令的第一操作数

```
        HSUBA   R01         ;ACC-R01，结果放入 ACC
        HSUBR   R01         ;ACC-R01，结果放入 R01
```

例：用 ACC 做双操作数指令的第二操作数

```
        SUBA    R01         ;R01-ACC，结果放入 ACC
        SUBR    R01         ;R01-ACC，结果放入 R01
```

12.5 程序状态寄存器(STATUS)

STATUS 寄存器如下表所示，包含：

•ALU 的算术状态

•复位状态

•数据存储器（GPR 和 SFR）的存储区选择位

与其他寄存器一样，STATUS 寄存器可以是任何指令的目标寄存器。如果一条影响 Z、DC 或 C 位的指令以 STATUS 寄存器作为目标寄存器，则不能写这 3 个状态位。这些位根据器件逻辑被置 1 或清零。而且也不能写 TO 和 PD 位。因此将 STATUS 作为目标寄存器的指令可能无法得到预期的结果。

例如，CLR STATUS 会清零高 3 位，并将 Z 位置 1。这样 STATUS 的值将为 000uu1uu（其中 u=不变）。因此，建议仅使用 CLRB、SETB、SWAPA、SWAPR 指令来改变 STATUS 寄存器，因为这些指令不会影响任何状态位。

程序状态寄存器 STATUS（03H）

03H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	1	1	X	X	X

Bit7 **IRP**：寄存器存储区选择位（用于间接寻址）
1=Bank2和Bank3（100h-1FFh）
0=Bank0和Bank1（00h-FFh）

Bit6-5 **RP[1:0]**：存储区选择位
0 0 选择Bank 0
0 1 选择Bank 1
1 0 选择Bank 2
1 1 选择Bank 3

Bit4 **TO**：超时位
1=上电或是执行了CLRWDT指令或STOP指令
0=发生了WDT超时

Bit3 **PD**：掉电位
1=上电或执行了CLRWDT指令
0=执行了STOP指令

Bit2 **Z**：结果为零位
1=算术或逻辑运算的结果为零
0=算术或逻辑运算的结果不为零

Bit1 **DC**：半进位/借位位
1=发生了结果的第4低位向高位进位
0=结果的第4低位没有向高位进位

Bit0 **C**：进位/借位位
1=结果的最高位发生了进位
0=结果的最高位没有发生进位

TO 和 PD 标志位可反映出芯片复位的原因，下面列出影响 TO、PD 的事件及各种复位后 TO、PD 的状态。

事件	TO	PD
电源上电	1	1
WDT 溢出	0	X
STOP 指令	1	0
CLRWDT 指令	1	1
休眠	1	0

影响 PD、TO 的事件表

TO	PD	复位原因
0	0	WDT 溢出唤醒休眠 MCU
0	1	WDT 溢出非休眠态
----	----	----
1	1	电源上电

复位后 TO/PD 的状态

12.6 预分频器(OPTION_REG)

OPTION_REG 寄存器是可读写的寄存器，包括各种控制位用于配置：

•TIMER0/WDT 预分频器

•TIMER0

•PORTB 上拉电阻控制

预分频器控制寄存器 OPTION_REG（81H）

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

Bit7 **RBPU**：PORTB上拉使能位
1=禁止PORTB上拉
0=由端口的各个锁存值使能 PORTB 上拉

Bit6 **INTEDG**：触发中断的边沿选择位
1=INT引脚上升沿触发中断
0=INT引脚下降沿触发中断

Bit5 **T0CS**：TIMER0时钟源选择位
1=T0CKI引脚上的跳变沿
0=内部指令周期时钟（$F_{OSC}/4$）

Bit4 **T0SE**：TIMER0时钟源边沿选择位
1=在T0CKI引脚信号从高电平跳变到低电平时递增
0=在 T0CKI 引脚信号从低电平跳变到高电平时递增

Bit3 **PSA**：预分频器分配位
1=预分频器分配给 WDT
0=预分频器分配给 TIMER0 模块

Bit2-0 **PS2～PS0**：预分配参数配置位

PS2-PS1-PS0	TIMER0 分频比	WDT 分频比
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

预分频寄存器实际上是一个 8 位的计数器，用于监视寄存器 WDT 时，是作为一个后分频器；用于定时器/计数器时，作为一个预分频器，通常统称作预分频器。在片内只有一个物理的分频器，只能用于 WDT/TIMER0 两者之一，不能同时使用。也就是说，若用于 TIMER0，WDT 就不能使用预分频器，反之亦然。

当用于 WDT 时，CLRWDT 指令将同时对预分频器和 WDT 定时器清零。

当用于 TIMER0 时，有关写入 TIMER0 的所有指令（如：CLR TMR0,SETB　TMR0,1 等）都会对预分频器清零。

由 TIMER0 还是 WDT 使用预分频器，完全由软件控制，可以动态改变。为了避免出现不该有的芯片复位，当从 TIMER0 换为 WDT 使用时，应该执行以下指令。

```
CLR     TMR0            ;TMR0 清零
CLRWDT                  ;WDT 清零
LDIA    B'00xx1111'
LD      OPTION_REG,A
LDIA    B'00xx1xxx'     ;设置新的预分频器
LD      OPTION_REG,A
```

将预分频器从分配给 WDT 切换为分配给 TIMER0 模块，应该执行以下指令

```
CLRWDT                  ;WDT 清零
LDIA    B'00xx0xxx'     ;设置新的预分频器
LD      OPTION_REG,A
```

注：要使 TIMER0 获取 1:1 的预分频比配置，可通过将选项寄存器的 PSA 位置 1 将预分频器分配给 WDT。

12.7 程序计数器（PC）

程序计数器(PC)控制程序内存 OTP 中的指令执行顺序，它可以寻址整个 OTP 的范围，取得指令码后，程序计数器(PC)会自动加一，指向下一个指令码的地址。但如果执行跳转、条件跳转、向 PCL 赋值、子程序调用、初始化复位、中断、中断返回、子程序返回等操作时，PC 会加载与指令相关的地址而不是下一条指令的地址。

当遇到条件跳转指令且符合跳转条件时，当前指令执行过程中读取的下一条指令将会被丢弃，且会插入一个空指令操作周期，随后才能取得正确的指令。反之，就会顺序执行下一条指令。

程序计数器(PC)是 12Bit 宽度，低 8 位通过 PCL（02H）寄存器用户可以访问，高 4 位用户不能访问。可容纳 4Kx16 位程序地址。对 PCL 赋值将会产生一个短跳转动作，跳转范围为当前页的 256 个地址。

注：当程序员在利用 PCL 作短跳转时，要先对 PC 高位缓冲寄存器 PCLATH 进行赋值。

下面给出几种特殊情况的 PC 值

复位时	PC=0000；
中断时	PC=0004(原来的 PC+1 会被自动压入堆栈)；
CALL 时	PC=程序指定地址(原来的 PC+1 会被自动压入堆栈)；
RET、RETI、RETi 时	PC=堆栈出来的值；
操作 PCL 时	PC[11:8]不变，PC[7:0]=用户指定的值；
JP 时	PC=程序指定的值；
其它指令	PC=PC+1；

12.8 看门狗计数器(WDT)

看门狗定时器（Watch Dog Timer）是一个片内自振式的 RC 振荡定时器，无需任何外围组件，即使芯片的主时钟停止工作，WDT 也能保持计时。WDT 计时溢出将产生复位。

12.8.1 WDT 周期

WDT 与 TIMER0 共用 8 位预分频器。在所有复位后，WDT 溢出周期 18ms，假如你需要改变的 WDT 周期，可以设置 OPTION_REG 寄存器。WDT 的溢出周期将受到环境温度，电源电压等参数影响。

“CLRWDT”和“STOP”指令将清除 WDT 定时器以及预分频器里的计数值（当预分频器分配给 WDT 时）。WDT

一般用来防止系统失控，或者可以说是用来防止单片机程序失控。在正常情况下，WDT 应该在其溢出前被“CLRWDT”指令清零，以防止产生复位。如果程序由于某种干扰而失控，那么不能在 WDT 溢出前执行“CLRWDT”指令，就会使 WDT 溢出而产生复位。使系统重启而不至于失去控制。若是 WDT 溢出产生的复位，则状态寄存器（STATUS）的“TO”位会被清零，用户可根据此位来判断复位是否是 WDT 溢出所造成的。

注：1.若使用 WDT 功能，一定要在程序的某些地方放置“CLRWDT”指令，以保证在 WDT 溢出前能被清零。否者会使芯片不停的复位，造成系统无法正常工作。
2.不能在中断程序中对 WDT 进行清零，否则无法侦测到主程序“跑飞”的情况。
3.程序中应在主程序中有一次清 WDT 的操作，尽量不要在多个分支中清零 WDT，这种架构能最大限度发挥看门狗计数器的保护功能。
4.看门狗计数器不同芯片的溢出时间有一定差异，所以设置清 WDT 时间时，应与 WDT 的溢出时间有较大的冗余，以避免出现不必要的 WDT 复位。

12.8.2 看门狗定时器控制寄存器 WDTCON

看门狗定时器控制寄存器 WDTCON(105H)

105H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	----	----	----	----	----	----	----	SWDTEN
R/W	----	----	----	----	----	----	----	R/W
复位值	----	----	----	----	----	----	----	0

Bit7-1 未用，读为 0
Bit0 SWDTEN：软件使能或禁止看门狗定时器位
1=使能 WDT
0=禁止 WDT（复位值）

注 1：如果 CONFIG 中 WDT 配置位=1，则 WDT 始终被使能，而与 SWDTEN 控制位的状态无关。如果 CONFIG 中 WDT 配置位=0，则可以使用 SWDTEN 控制位使能或禁止 WDT。

13. 系统时钟

13.1 概述

时钟信号由内部振荡产生，在片内产生 4 个非重叠正交时钟信号，分别称作 Q1、Q2、Q3、Q4。在 IC 内部每个 Q1 使程序计数器（PC）增量加一，Q4 从程序存储单元中取出该指令，并将其锁存在指令寄存器中。在下一个 Q1 到 Q4 之间对取出的指令进行译码和执行，也就是说 4 个时钟周期才会执行一条指令。下图表示时钟与指令周期执行时序图。

一个指令周期含有 4 个 Q 周期，指令的执行和获取是采用流水线结构，取指占用一个指令周期，而译码和执行占用另一个指令周期，但是由于流水线结构，从宏观上看，每条指令的有效执行时间是一个指令周期。如果一条指令引起程序计数器地址发生改变（例如 JP）那么预取的指令操作码就无效，就需要两个指令周期来完成该条指令，这就是对 PC 操作指令都占用两个时钟周期的原因。

时钟与指令周期时序图如 13.1 所示。

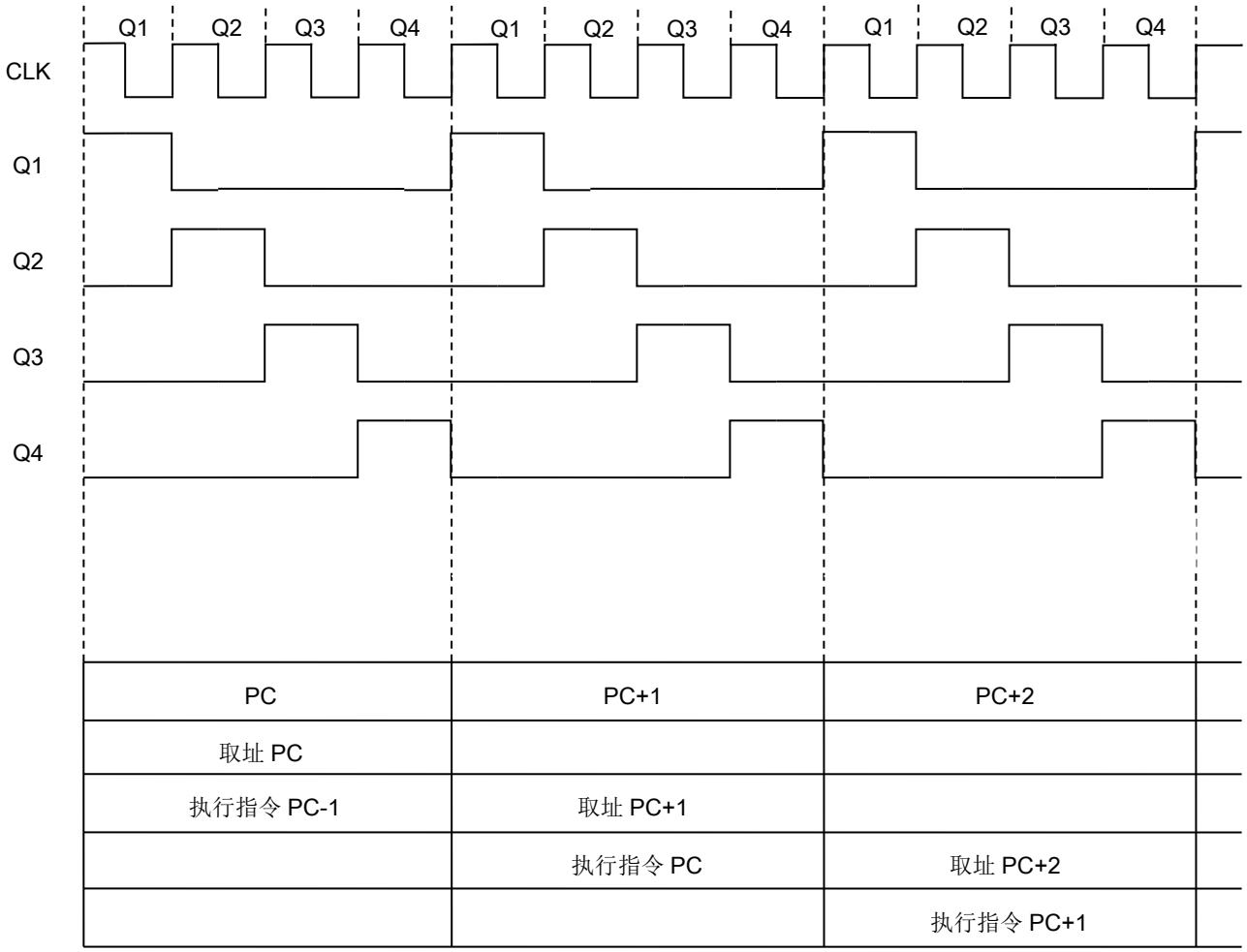


图 13.1 时钟与指令周期时序

下面列出振荡频率与指令速度的关系

频率	双指令周期	单指令周期
1MHz	8μs	4μs
2MHz	4μs	2μs

4MHz	2μs	1μs
8MHz	1μs	500ns

13.2 系统振荡器

芯片有 1 种振荡方式，内部 RC 振荡。

13.2.1 内部 RC 振荡

芯片默认的振荡方式为内部 RC 振荡，其振荡频率为 8MHz 可通过 OSCCON 寄存器设置芯片工作频率。振荡频率在出厂时校正，其误差在±3%以内。

13.3 起振时间

起振时间（Reset Time）是指从芯片复位到芯片振荡稳定这段时间，其设计值为 18ms。

注：无论芯片是电源上电复位，还是其它原因引起的复位，都会存在这个起振时间。

13.4 振荡器控制寄存器

振荡器控制（OSCCON）寄存器控制系统时钟和频率选择，振荡器调节寄存器 OSCTUNE 可以用软件调节内部振荡频率。

振荡器控制寄存器 OSCCON(8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	SCS
R/W	----	R/W	R/W	R/W	----	----	----	R/W
复位值	----	1	1	0	----	----	----	0

Bit7 **未用，读为 0**

Bit6-4 **IRCF<2:0>**：内部振荡器频率选择位
111=8MHz
110=4MHz（默认）
101=2MHz
100=1MHz
011=500kHz
010=250kHz
001=125kHz
000=32kHz（LFINTOSC）

Bit3-Bit1 **未用**

Bit0 **SCS**：系统时钟选择位
1=内部振荡器用作系统时钟
0=时钟源由CONFIG定义

振荡器调节寄存器 OSCTUNE(90H)

90H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCTUNE	----	----	----	TUN4	TUN3	TUN2	TUN1	TUN0
R/W	----	----	----	R/W	R/W	R/W	R/W	R/W
复位值	----	----	----	0	0	0	0	0

Bit7-5	**未用**
Bit4-0	**TUN<4:0>**：频率调节位 01111=最高频率 01110= . . . 00001= 00000=振荡器模块以厂家校准后的频率运行 11111= . . . 10000=最低频率

14. 复位

芯片可用如下 3 种复位方式：

◆上电复位

◆LVR 复位

◆正常工作下的看门狗溢出复位

上述任意一种复位发生时，所有的系统寄存器将恢复默认状态，程序停止运行，同时程序计数器 PC 清零，复位结束后程序从复位向量 0000H 开始运行。STATUS 的 TO 和 PD 标志位能够给出系统复位状态的信息，（详见 STATUS 的说明），用户可根据 PD 和 TO 的状态，控制程序运行路径。

任何一种复位情况都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。

14.1 上电复位

上电复位与 LVR 操作密切相关。系统上电的过程呈逐渐上升的曲线形式，需要一定时间才能达到正常电平值。下面给出上电复位的正常时序：

- 上电：系统检测到电源电压上升并等待其稳定；
- 系统初始化：所有的系统寄存器被置为初始值；
- 振荡器开始工作：振荡器开始提供系统时钟；
- 执行程序：上电结束，程序开始运行。

14.2 掉电复位

14.2.1 掉电复位概述

掉电复位针对外部因素引起的系统电压跌落情形（例如，干扰或外部负载的变化）。电压跌落可能会进入系统死区，系统死区意味着电源不能满足系统的最小工作电压要求。

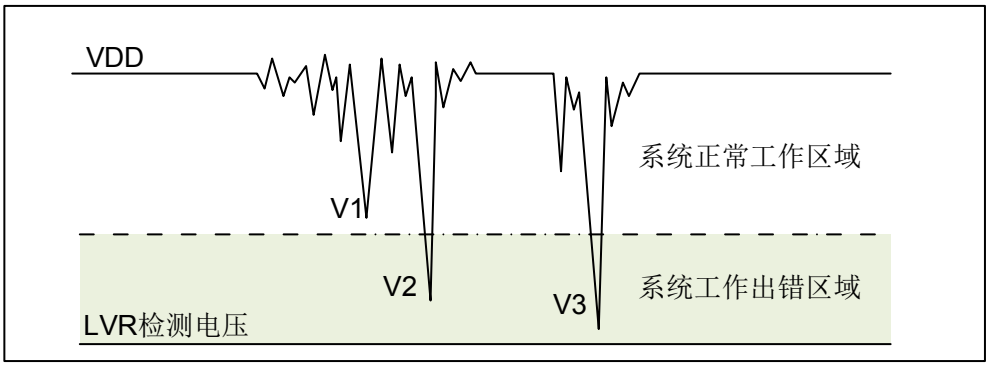


图14.1 掉电复位示意图

上图是一个典型的掉电复位示意图。图中，VDD 受到严重的干扰，电压值降的非常低。虚线以上区域系统正常工作，在虚线以下的区域内，系统进入未知的工作状态，这个区域称作死区。当 VDD 跌至 V1 时，系统仍处于正常状态；当 VDD 跌至 V2 和 V3 时，系统进入死区，则容易导致出错。

以下情况系统可能进入死区：

DC 应用中：

DC 应用中一般都采用电池供电，当电池电压过低或单片机驱动负载时，系统电压可能跌落并进入死区。这时，电源不会进一步下降到 LVR 检测电压，因此系统维持在死区。

AC 应用中：

系统采用 AC 供电时，DC 电压值受 AC 电源中的噪声影响。当外部负载过高，如驱动马达时，负载动作产生的干扰也影响到 DC 电源。VDD 若由于受到干扰而跌落至最低工作电压以下时，则系统将有可能进入不稳定工作状态。

在 AC 应用中，系统上、掉电时间都较长。其中，上电时序保护使得系统正常上电，但掉电过程却和 DC 应用中情形类似，AC 电源关断后，VDD 电压在缓慢下降的过程中易进入死区。

如上图所示，系统正常工作电压区域一般高于系统复位电压，同时复位电压由低电压检测（LVR）电平决定。当系统执行速度提高时，系统最低工作电压也相应提高，但由于系统复位电压是固定的，因此在系统最低工作电压与系统复位电压之间就会出现一个电压区域，系统不能正常工作，也不会复位，这个区域即为死区。

14.2.2 掉电复位的改进办法

如何改进系统掉电复位性能，以下给出几点建议：

- 选择较高的 LVR 电压，有助于复位更可靠
- 开启看门狗定时器
- 降低系统的工作频率
- 增大电压下降斜率

看门狗定时器

看门狗定时器用于保证程序正常运行，当系统进入工作死区或者程序运行出错时，看门狗定时器会溢出，系统复位。

降低系统的工作速度

系统工作频率越快，系统最低工作电压越高。从而增大了工作死区的范围，降低系统工作速度就可以降低最低工作电压，从而有效的减小系统工作在死区电压的机率。

增大电压下降斜率

此方法可用于系统工作在 AC 供电的环境，一般 AC 供电系统，系统电压在掉电过程中下降很缓慢，这就会造成芯片较长时间工作在死区电压，此时若系统重新上电，芯片工作状态可能出错，建议在芯片电源与地线间加一个放电电阻，以便让 MCU 快速通过死区，进入复位区，避免芯片上电出错可能性。

14.3 看门狗复位

看门狗复位是系统的一种保护设置。在正常状态下，由程序将看门狗定时器清零。若出错，系统处于未知状态，看门狗定时器溢出，此时系统复位。看门狗复位后，系统重启进入正常状态。

看门狗复位的时序如下：

1、看门狗定时器状态：系统检测看门狗定时器是否溢出，若溢出，则系统复位；

2、初始化：所有的系统寄存器被置为默认状态；

3、振荡器开始工作：振荡器开始提供系统时钟；

4、程序：复位结束，程序开始运行。

15. 休眠模式

15.1 进入休眠模式

执行 STOP 指令可进入掉电模式。如果 WDT 使能，那么：

•WDT 将被清零并继续运行。

•STATUS 寄存器中的 PD 位被清零。

•TO 位被置 1。

•关闭振荡器驱动器。

•I/O 端口保持执行 STOP 指令之前的状态（驱动为高电平、低电平或高阻态）。

在休眠模式下，为了尽量降低电流消耗，所有 I/O 引脚都应该保持为 VDD 或 GND，没有外部电路从 I/O 引脚消耗电流。为了避免输入引脚悬空而引入开关电流，应在外部将高阻输入的 I/O 引脚拉为高电平或低电平。为了将电流消耗降至最低，还应考虑芯片内部上拉电阻的影响。

15.2 从休眠状态唤醒

可以通过下列任一事件将器件从休眠状态唤醒：

1、看门狗定时器唤醒（WDT 强制使能）

2、PORTB 电平变化中断或外设中断。

上述两种事件被认为是程序执行的延续,STATUS 寄存器中的 TO 和 PD 位用于确定器件复位的原因。PD 位在上电时被置 1，而在执行 STOP 指令时被清零。TO 位在发生 WDT 唤醒时被清零。

当执行 STOP 指令时，下一条指令(PC+1)被预先取出。如果希望通过中断事件唤醒器件，则必须将相应的中断允许位置 1（允许）。唤醒与 GIE 位的状态无关。如果 GIE 位被清零（禁止），器件将继续执行 STOP 指令之后的指令。如果 GIE 位被置 1（允许），器件执行 STOP 指令之后的指令，然后跳转到中断地址（0004h）处执行代码。如果不想执行 STOP 指令之后的指令，用户应该在 STOP 指令后面放置一条 NOP 指令。器件从休眠状态唤醒时，WDT 都将被清零，而与唤醒的原因无关。

15.3 使用中断唤醒

当禁止全局中断（GIE 被清零）时，并且有任一中断源将其中断允许位和中断标志位置 1，将会发生下列事件之一：

•如果在执行 STOP 指令之前产生了中断，那么 STOP 指令将被作为一条 NOP 指令执行。因此，WDT 及其预分频器和后分频器（如果使能）将不会被清零，并且 TO 位将不会被置 1，同时 PD 也不会被清零。

•如果在执行 STOP 指令期间或之后产生了中断，那么器件将被立即从休眠模式唤醒。STOP 指令将在唤醒之前执行完毕。因此，WDT 及其预分频器和后分频器（如果使能）将被清零，并且 TO 位将被置 1，同时 PD 也将被清零。即使在执行 STOP 指令之前检查到标志位为 0，它也可能在 STOP 指令执行完毕之前被置 1。要确定是否执行了 STOP 指令，可以测试 PD 位。如果 PD 位置 1，则说明 STOP 指令被作为一条 NOP 指令执行了。在执行 STOP 指令之前，必须先执行一条 CLRWDT 指令，来确保将 WDT 清零。

15.4 休眠模式应用举例

系统在进入休眠模式之前，若用户需要获得较小的休眠电流，请先确认所有 I/O 的状态，若用户方案中存在悬空的 I/O 口，把所有悬空口都设置为输出口，确保每一个输入口都有一个固定的状态，以避免 I/O 为输入状态时，口线电平处于不定态而增大休眠电流；关断 AD 等其它外设模块；根据实际方案的功能需求可禁止 WDT 功能来减小休眠电流。

例：进入休眠的处理程序

```
SLEEP_MODE：
            CLR     INTCON      ;关断中断使能；
            LDIA    B’00000000’
            LD      TRISA,A
            LD      TRISB,A     ;所有 I/O 设置为输出口；
            LD      TRISC,A
            LD      TRISE,A
            …                   ;关闭其它功能；
            LDIA    0A5H
            LD      SP_FLAG,A   ;置休眠状态记忆寄存器(用户自定义)；
            CLRWDT              ;清零 WDT；
            STOP                ;执行 STOP 指令。
```

15.5 休眠模式唤醒时间

当 MCU 从休眠态被唤醒时，需要等待一个振荡稳定时间（Reset Time）,这个时间标称值为 18ms。

16. I/O 端口

芯片有 4 组 I/O 端口：PORTB、PORTC 为外部 I/O，PORTA、PORTE 为内部通讯口。可读写端口数据寄存器可直接存取这些端口。

表16-1 端口配置总概

端口	位	管脚描述	输入/输出
PORTB	1	施密特触发输入，推挽式输出，AN10 输入	I/O
	2	施密特触发输入，推挽式输出，AN8 输入	I/O
	3	施密特触发输入，推挽式输出，AN9 输入	I/O
	4	施密特触发输入，推挽式输出，AN11 输入	I/O
	5	施密特触发输入，推挽式输出，AN13 输入	I/O
	6	施密特触发输入，推挽式输出，编程时钟输入	I/O
	7	施密特触发输入，推挽式输出，编程数据输入/输出	I/O
PORTC	0	施密特触发输入，NMOS 开漏输出	I/O
	1	施密特触发输入，推挽式输出， CCP2	I/O
	2	施密特触发输入，推挽式输出， CCP1	I/O

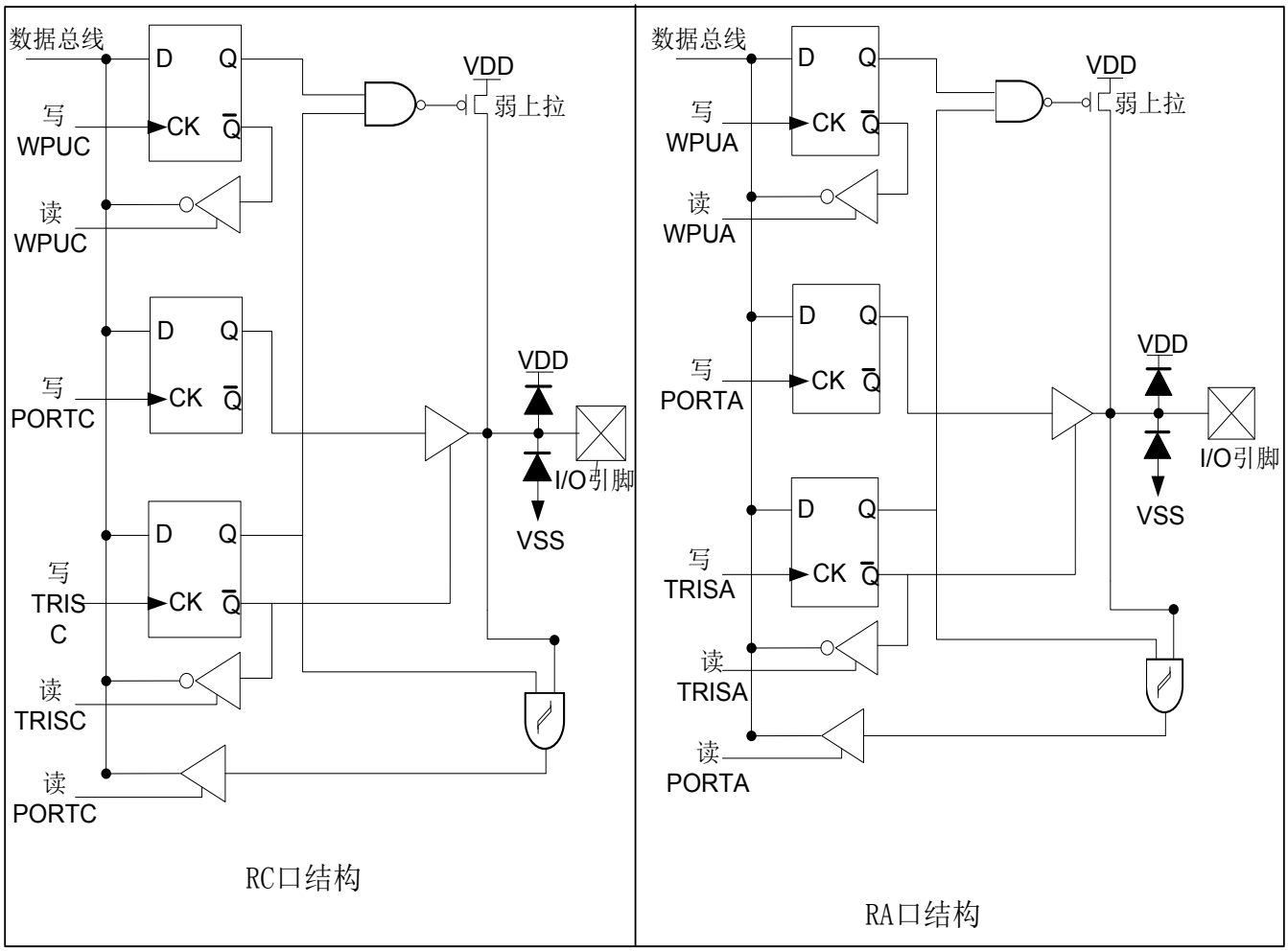


图16.1 RA/RC口结构图

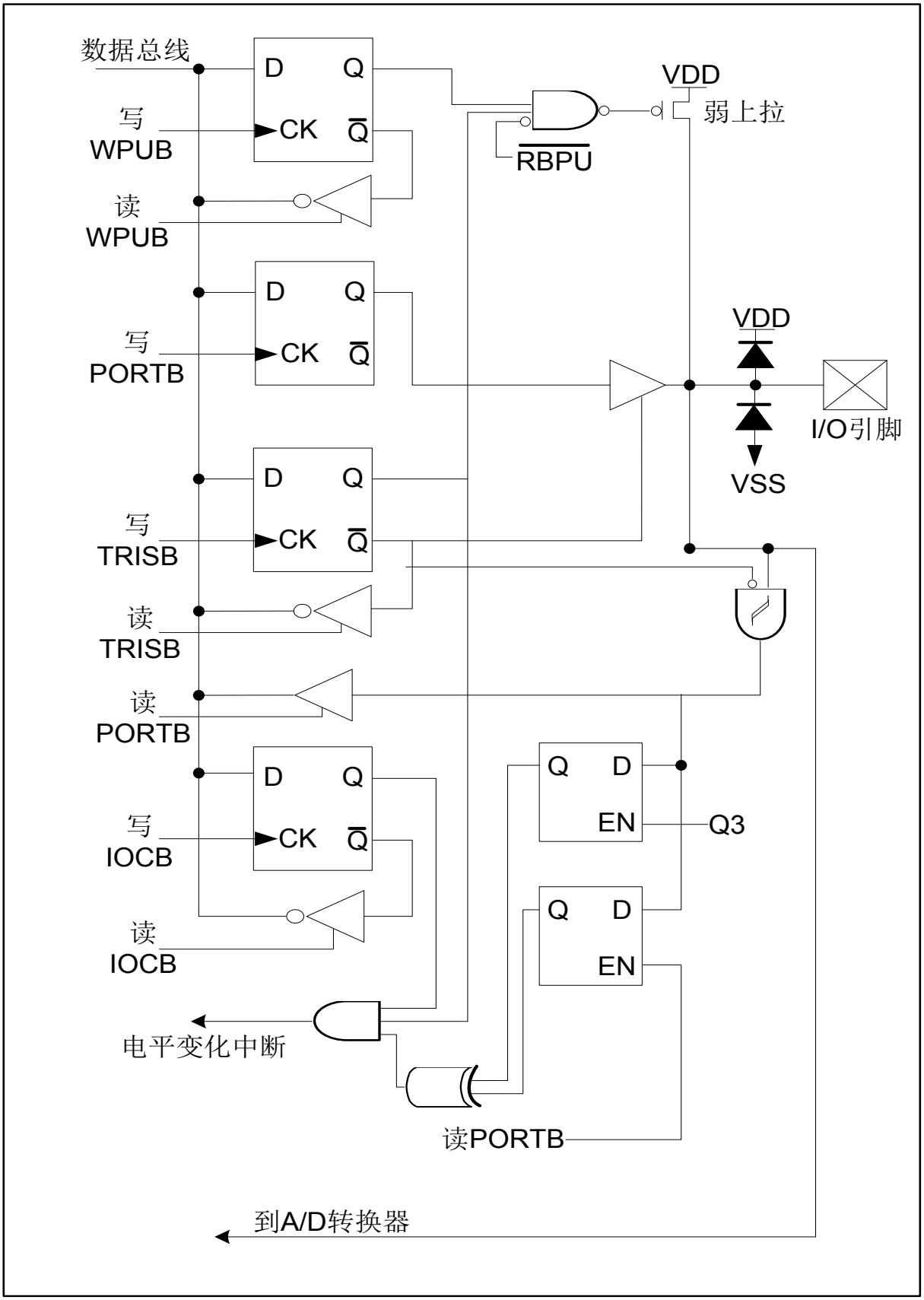


图 16.2 RB 口结构图

16.1 PORTA

16.1.1 PORTA 数据及方向控制

PORTA 是 4 位宽的双向端口。它所对应的数据方向寄存器是 TRISA。将 TRISA 的一个位置 1（=1）可以将相应的引脚配置为输入。清零 TRISA 的一个位（=0）可将相应的 PORTA 引脚配置为输出。

读 PORTA 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读－修改－写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。**RA7 和 RA5 口使用过程中必须设置为输入口。**

与 PORTA 口相关寄存器有 PORTA、TRISA。

PORTA 数据寄存器 PORTA(05H)

05H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	RA7	----	RA5	RA4	RA3	----	----	----
R/W	R/W	----	R/W	R/W	R/W	----	----	----
复位值	x	----	x	x	x	----	----	----

Bit7 PORTA<7>：PORTA I/O 引脚位
1=端口引脚电平>V_{IH}
0=端口引脚电平<V_{IL}

Bit5-3 PORTA<5:3>：PORTA I/O引脚位
1=端口引脚电平>V_{IH}
0=端口引脚电平<V_{IL}

PORTA 方向寄存器 TRISA(85H)

85H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	TRISA7	----	TRISA5	TRISA4	TRISA3	----	----	----
R/W	R/W	----	R/W	R/W	R/W	----	----	----
复位值	1	----	1	1	1	----	----	----

Bit7 TRISA<7>：PORTA 三态控制位
1=PORTA 引脚被配置为输入
0=PORTA 引脚被配置为输出

Bit5-3 TRISA<5:3>：PORTA 三态控制位
1=PORTA 引脚被配置为输入
0=PORTA 引脚被配置为输出

例：PORTA 口处理程序

```
LDIA    B'11110000'     ;设置PORTA<3:0>为输出口，PORTA<7:4>为输入口
LD      TRISA,A
LDIA    03H             ;PORTA<1:0>输出高电平，PORTA<3:2>输出低电平
LD      PORTA,A         ;由于PORTA<7:4>为输入口，所以赋0或1都没影响
```

16.2 PORTB

16.2.1 PORTB 数据及方向

PORTB 是一个 7 位宽的双向端口。对应的数据方向寄存器为 PORTB。将 TRISB 中的某个位置 1（=1）可以使对应的 PORTB 引脚作为输入引脚。将 TRISB 中的某个位清零（=0）将使对应的 PORTB 引脚作为输出引脚。

读 PORTB 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读－修改－写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。

与 PORTB 口相关寄存器有 PORTB、TRISB、WPUB、IOCB 等。

PORTB 数据寄存器 PORTB(06H)

06H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	----
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	----
复位值	x	x	x	x	x	x	x	----

Bit7-1 **PORTB<7:1>**：PORTBI/O 引脚位
1=端口引脚电平>V_{IH}
0=端口引脚电平<V_{IL}

PORTB 方向寄存器 TRISB(86H)

86H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	----
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	----
复位值	1	1	1	1	1	1	1	----

Bit7-1 **TRISB<7:1>**：PORTB 三态控制位
1=PORTB 引脚被配置为输入（三态）
0=PORTB 引脚被配置为输出

例：PORTB 口处理程序

```
CLR   PORTB              ;清数据寄存器
LDIA  B'00110000'        ;设置 PORTB<5:4>为输入口，其余为输出口
LD    TRISB,A
```

16.2.2 PORTB 上拉电阻

每个 PORTB 引脚都有可单独配置的内部弱上拉。控制位 WPUB<7:1>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。在上电复位时，弱上拉由 OPTION_REG 寄存器的 RBPU 位禁止。

PORTB 上拉电阻寄存器 WPUB(95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	----
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	----
复位值	0	0	0	0	0	0	0	----

Bit7-1 **WPUB<7:1>：**弱上拉寄存器位
1=使能上拉
0=禁止上拉

注 1：要单独使能任一个上拉，OPTION_REG 寄存器的全局 RBPU 位必须清零。

16.2.3 PORTB 电平变化中断

所有的 PORTB 引脚都可以被单独配置为电平变化中断引脚。控制位 IOCB<7:0>允许或禁止每个引脚的该中断功能。上电复位时禁止引脚的电平变化中断功能。

对于已允许电平变化中断的引脚，则将该引脚上的值与上次读 PORTB 时锁存的旧值进行比较。与上次读操作“不匹配”将会使 INTCON 寄存器中的 PORTB 电平变化中断标志位（RBIF）置 1。

该中断可将器件从休眠中唤醒。用户可在中断服务程序中通过以下方式清除中断：

a)对 PORTB 进行读或写操作。这将结束引脚电平的不匹配状态。

b)将标志位 RBIF 清零。

不匹配状态会不断将 RBIF 标志位置 1。而读或写 PORTB 将结束不匹配状态，并且允许将 RBIF 标志位清零。锁存器将保持最后一次读取的值不受欠压复位的影响。在复位之后，如果不匹配仍然存在，RBIF 标志位将继续置 1。

注：如果在执行读取操作时（Q2 周期的开始）I/O 引脚的电平发生变化，则 RBIF 中断标志位不会被置 1。此外，由于对端口的读或写影响到该端口的所有位，所以在电平变化中断模式下使用多个引脚的时候必须特别小心。在处理一个引脚电平变化的时候可能不会注意到另一个引脚上的电平变化。

PORTB 电平变化中断寄存器 IOCB(96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	----
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	----
复位值	0	0	0	0	0	0	0	----

Bit7-1 **IOCB<7:1>：**PORTB 的电平变化中断控制位
1=允许电平变化中断
0=禁止电平变化中断

16.3 PORTC

16.3.1 PORTC 数据及方向

PORTC 是一个 3 位宽的双向端口。对应的数据方向寄存器为 TRISC。将 TRISC 中的某个位置 1（=1）可以使对应的 PORTC 引脚作为输入引脚。将 TRISC 中的某个位清零（=0）将使对应的 PORTC 引脚作为输出引脚。

读 PORTC 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读－修改－写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。

PORTC 数据寄存器 PORTC(07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTC	----	----	----	----	----	RC2	RC1	RC0
R/W	----	----	----	----	----	R/W	R/W	R/W
复位值	----	----	----	----	----	x	x	x

Bit2-0 **PORTC<2:0>**：PORTCI/O 引脚位
1=端口引脚电平>V_{IH}
0=端口引脚电平<V_{IL}

PORTC 方向寄存器 TRISC(87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	----	----	----	----	----	TRISC2	TRISC1	TRISC0
R/W	----	----	----	----	----	R/W	R/W	R/W
复位值	----	----	----	----	----	1	1	1

Bit2-0 **TRISC<2:0>**：PORTC 三态控制位
1=PORTC 引脚被配置为输入（三态）
0=PORTC 引脚被配置为输出

注:RC0 置输出态时,只有开漏输出功能

例：PORTC 口处理程序

```
CLR    PORTC                ;清数据寄存器
LDIA   B'01110000'          ;设置 PORTC<3:0>为输出口
LD     TRISC,A
```

16.3.2 PORTC 上拉电阻

每个 PORTC 引脚都有可单独配置的内部弱上拉。控制位 WPUC<7:0>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。

PORTC 上拉电阻寄存器 WPUC(18FH)

18FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUC	----	----	----	----	----	WPUC2	WPUC1	----
R/W	----	----	----	----	----	R/W	R/W	----
复位值	----	----	----	----	----	0	0	----

Bit2-1 **WPUC<2:1>**：弱上拉寄存器位
1=使能上拉
0=禁止上拉

注 1：如果引脚被配置为输出，将自动禁止弱上拉。

16.4 PORTE

16.4.1 PORTE 数据及方向

PORTE 是一个 2 位宽的双向端口。对应的数据方向寄存器为 TRISE。将 TRISE 中的某个位置 1（=1）可以使对应的 PORTE 引脚作为输入引脚。将 TRISE 中的某个位清零（=0）将使对应的 PORTE 引脚作为输出引脚。

读 PORTE 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读－修改－写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。**RE1 和 RE2 口在使用过程中必须设置为输入口。**

PORTE 数据寄存器 PORTE(09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTE	----	----	----	----	----	RE2	RE1	----
R/W	----	----	----	----	----	R/W	R/W	----
复位值	----	----	----	----	----	x	x	----

Bit2-1 **PORTE<2:1>**：PORTEI/O 引脚位

1=端口引脚电平>V_{IH}

0=端口引脚电平<V_{IL}

PORTE 方向寄存器 TRISE(89H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISE	----	----	----	----	----	TRISE2	TRISE1	----
R/W	----	----	----	----	----	R/W	R/W	----
复位值	----	----	----	----	----	1	1	----

Bit2-1 **TRISE<2:1>**：PORTE 三态控制位

1=PORTE 引脚被配置为输入

0=PORTE 引脚被配置为输出

16.5 I/O 使用

16.5.1 写 I/O 口

芯片的 I/O 口寄存器，和一般通用寄存器一样，可以通过数据传输指令，位操作指令等进行写操作。

例：写 I/O 口程序

```
        LD      PORTA,A     ;ACC 值赋给 PORTA 口
        CLRB    PORTB,1     ;PORTB.1 口置零
        CLR     PORTC       ;PORTC 口清零
        SET     PORTA       ;PORTA 所有输出口置 1
        SETB    PORTB,1     ;PORTB.1 口置 1
```

16.5.2 读 I/O 口

例：读 I/O 口程序

```
        LD      A,PORTA     ;PORTA 的值赋给 ACC
        SNZB    PORTA,1     ;判断 PORTA,1 口是否为 1，为 1 跳过下一条语句
        SZB     PORTA,1     ;判断 PORTA,1 口是否为 0，为 0 跳过下一条语句
```

注：当用户读一个 I/O 口状态时，若此 I/O 口为输入口，则用户读回的数据将是此口线外部电平的状态，若此 I/O 口为输出口那么读出的值将会是此口线内部输出寄存器的数据。

16.6 I/O 口使用注意事项

在操作 I/O 口时，应注意以下几个方面：

◆ 当 I/O 从输出转换为输入时，要等待几个指令周期的时间，以便 I/O 口状态稳定。

◆ 若使用内部上拉电阻，那么当 I/O 从输出转换为输入时，内部电平的稳定时间，与接在 I/O 口上的电容有关，用户应根据实际情况，设置等待时间，以防止 I/O 口误扫描电平。

◆ 当 I/O 口为输入口时，其输入电平应在“VDD+0.7V”与”GND-0.7V”之间。若输入口电压不在此范围内可采用如下图所示方法。

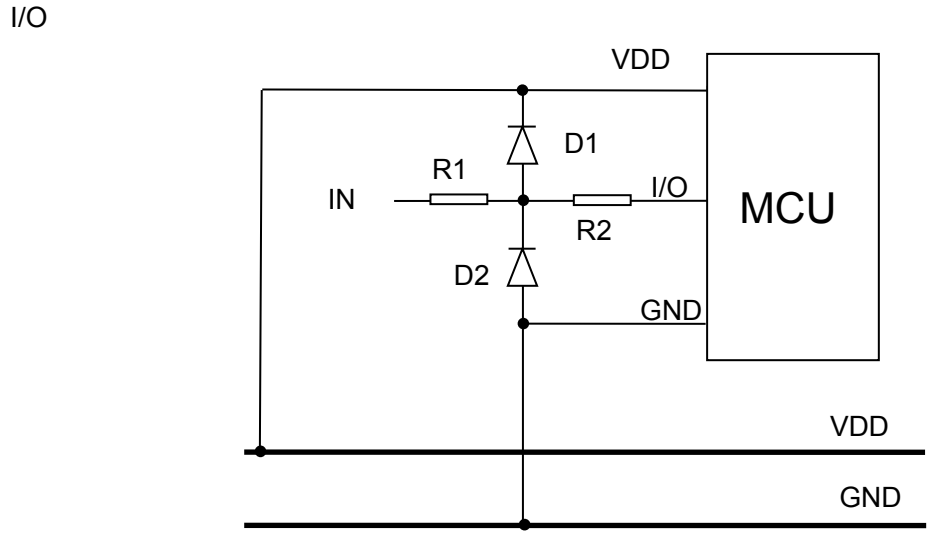


图 16.3 I/O 的 ESD 防护图

◆ 若在 I/O 口在线串入较长的连接线，请在靠近芯片 I/O 的地方加上限流电阻以增强 MCU 抗 EMC 能力。

17. 中断

17.1 中断概述

芯片具有以下多种中断源：

•TIMER0 溢出中断

•TIMER1 溢出中断

•TIMER2 匹配中断

•INT 中断

•PORTB 电平变化中断

•A/D 中断

中断控制寄存器（INTCON）和外设中断请求寄存器（PIR1）在各自的标志位中记录各种中断请求。INTCON 寄存器还包括各个中断允许位和全局中断允许位。

全局中断允许位 GIE（INTCON<7>）在置 1 时允许所有未屏蔽的中断，而在清零时，禁止所有中断。可以通过 INTCON、PIE1 寄存器中相应的允许位来禁止各个中断。复位时 GIE 被清零。

执行“从中断返回”指令 RETI 将退出中断服务程序并将 GIE 位置 1，从而重新允许未屏蔽的中断。

中断执行示意图如图 17.1 所示。

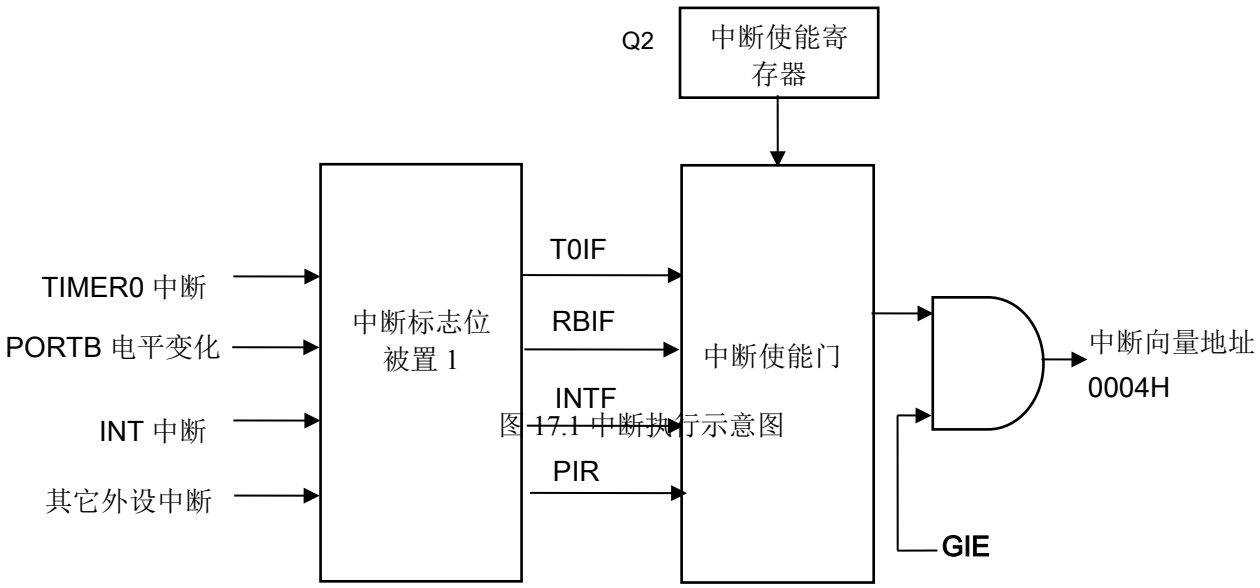


图 17.1 中断执行示意图

17.2 中断控制寄存器

17.2.1 中断控制寄存器

中断控制寄存器 INTCON 是可读写的寄存器，包含 TMR0 寄存器溢出、PORTB 端口电平变化中断等的允许和标志位。

当有中断条件产生时，无论对应的中断允许位或（INTCON 寄存器中的）全局允许位 GIE 的状态如何，中断标志位都将置 1。用户软件应在允许一个中断之前，确保先将相应的中断标志位清零。

中断控制寄存器 INTCON（0BH）

0BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 **GIE**：全局中断允许位
1=允许所有未被屏蔽的中断
0=禁止所有中断

Bit6 **PEIE**：外设中断允许位
1=允许所有未被屏蔽的外设中断
0=禁止所有外设中断

Bit5 **T0IE**：TIMER0溢出中断允许位
1=允许TIMER0中断
0=禁止TIMER0中断

Bit4 **INTE**：INT外部中断允许位
1=允许INT外部中断
0=禁止INT外部中断

Bit3 **RBIE**：PORTB电平变化中断允许位(1)
1=允许PORTB电平变化中断
0=禁止PORTB电平变化中断

Bit2 **T0IF**：TIMER0溢出中断标志位(2)
1=TMR0寄存器已经溢出（必须由软件清零）
0=TMR0寄存器未发生溢出

Bit1 **INTF**：INT外部中断标志位
1=发生INT外部中断（必须由软件清零）
0=未发生INT外部中断

Bit0 **RBIF**：PORTB电平变化中断标志位
1=PORTB端口中至少有一个引脚的电平状态发生了改变（必须由软件清零）
0=没有一个PORTB通用I/O引脚的状态发生了改变

注：
1、IOCB 寄存器也必须使能，相应的口线需设置为输入态。
2、T0IF 位在 TMR0 计满归 0 时置 1。复位不会使 TMR0 发生改变，应在将 T0IF 位清零前对其进行初始化。

17.2.2 外设中断允许寄存器

外设中断允许寄存器有 PIE1，在允许任何外设中断前，必须先将 INTCON 寄存器的 PEIE 位置 1。

外设中断允许寄存器 PIE1(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	----	ADIE	----	----	----	----	TMR2IE	TMR1IE
R/W	----	R/W	----	----	----	----	R/W	R/W
复位值	----	0	----	----	----	----	0	0

Bit7 **未用，读为0**

Bit6 **ADIE**：A/D转换器（ADC）中断允许位
1=允许ADC中断
0=禁止ADC中断

Bit5-2 **未用**

Bit1 **TMR2IE**：TIMER2与PR2匹配中断允许位
1=允许TMR2与PR2匹配中断
0=禁止TMR2与PR2匹配中断

Bit0 **TMR1IE**：TIMER1溢出中断允许位
1=允许TIMER1溢出中断
0=禁止TIMER1溢出中断

17.2.3 外设中断请求寄存器

外设中断请求寄存器为 PIR1 和 PIR2。当有中断条件产生时，无论对应的中断允许位或全局允许位 GIE 的状态如何，中断标志位都将置 1。用户软件应在允许一个中断之前，确保先将相应的中断标志位清零。

外设中断请求寄存器 PIR1(0CH)

0CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	----	ADIF	----	----	----	----	TMR2IF	TMR1IF
R/W	----	R/W	----	----	----	----	R/W	R/W
复位值	----	0	----	----	----	----	0	0

Bit7 **未用**

Bit6 **ADIF**：A/D转换器中断标志位
1=A/D转换完成（必须由软件清零）
0=A/D转换未完成或尚未启动

Bit5-2 **未用**

Bit1 **TMR2IF**：TIMER2与PR2匹配中断标志位
1=发生了TIMER2与PR2匹配（必须由软件清零）
0=TIMER2与PR2不匹配

Bit0 **TMR1IF**：TIMER1溢出中断标志位
1=TMR1寄存器溢出（必须由软件清零）
0=TMR1寄存器未溢出

17.3 中断现场的保护方法

有中断请求发生并被响应后，程序转至 0004H 执行中断子程序。响应中断之前，必须保存 ACC、STATUS 的内容。芯片没有提供专用的入栈保存和出栈恢复指令，用户需自己保护 ACC 和 STATUS 的内容，以避免中断结束后可能的程序运行错误。

例：对 ACC 与 STATUS 进行入栈保护

```
                ORG     0000H
                JP      START           ;用户程序起始地址
                ORG     0004H
                JP      INT_SERVICE     ;中断服务程序
                ORG     0008H
START：
                …
                …
INT_SERVICE：
PUSH：                                  ;中断服务程序入口，保存 ACC 及 STATUS
                LD      ACC_BAK,A       ;保存 ACC 的值，(ACC_BAK 需自定义)
                SWAPA   STATUS
                LD      STATUS_BAK,A    ;保存 STATUS 的值，（STATUS_BAK 需自定义）
                …
                …
POP：                                   ;中断服务程序出口，还原 ACC 及 STATUS
                SWAPA   STATUS_BAK
                LD      STATUS,A        ;还原 STATUS 的值
                SWAPR   ACC_BAK         ;还原 ACC 的值
                SWAPA   ACC_BAK
                RETI
```

17.4 中断的优先级，及多中断嵌套

芯片的各个中断的优先级是平等的，当一个中断正在进行的时候，不会响应另外一个中断，只有执行“RETI”指令后，才能响应下一个中断。

多个中断同时发生时，MCU 没有预置的中断优先级。首先，必须预先设定好各中断的优先权；其次，利用中断使能位和中断控制位，控制系统是否响应该中断。在程序中，必须对中断控制位和中断请求标志进行检测。

18. 定时计数器 TIMER0

18.1 定时计数器 TIMER0 概述

TIMER0 由如下功能组成：

•8 位定时器/计数器寄存器（TMR0）

•8 位预分频器（与看门狗定时器共用）

•可编程内部或外部时钟源

•可编程外部时钟边沿选择

•溢出中断

图 18.1 为 TIMER0 模块的框图。

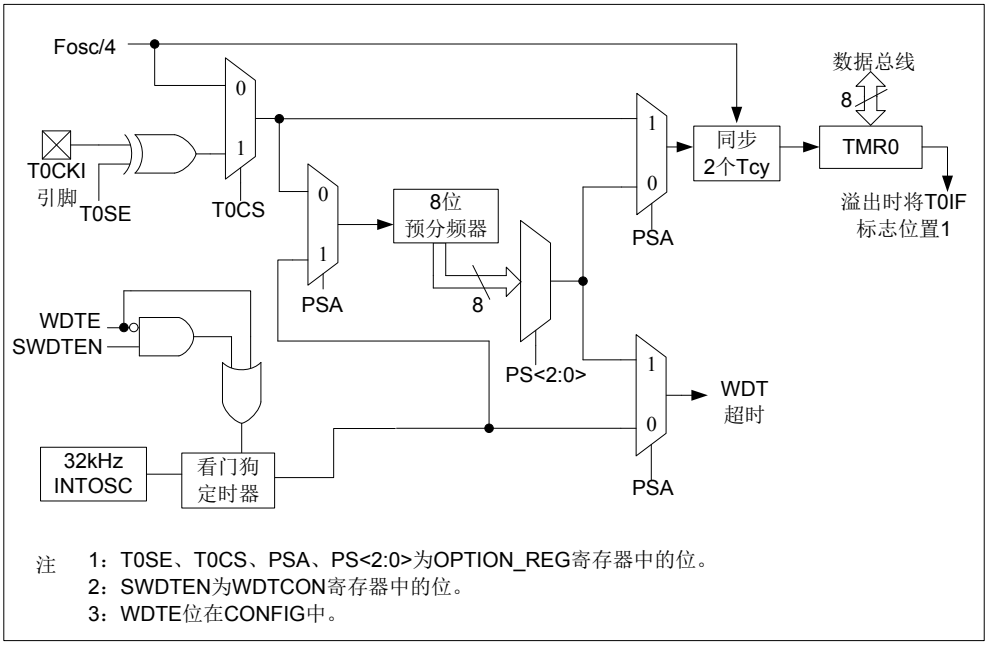


图18.1 TIMER0/WDT结构图

18.2 TIMER0 的工作原理

TIMER0 模块既可用作 8 位定时器也可用作 8 位计数器。

18.2.1 8 位定时器模式

用作定时器时，TIMER0 模块将在每个指令周期递增（不带预分频器）。通过将 OPTION_REG 寄存器的 T0CS 位清 0 可选择定时器模式。如果对 TMR0 寄存器执行写操作，则在接下来的两个指令周期将禁止递增。可调整写入 TMR0 寄存器的值，使得在写入 TMR0 时计入两个指令周期的延时。

18.2.2 8 位计数器模式

用作计数器时，TIMER0 模块将在 T0CKI 引脚的每个上升沿或下降沿递增。递增的边沿取决于 OPTION_REG 寄存器的 T0SE 位。通过将 OPTION_REG 寄存器的 T0CS 位置 1 可选择计数器模式。

18.2.3 软件可编程预分频器

TIMER0 和看门狗定时器（WDT）共用一个软件可编程预分频器，但不能同时使用。预分频器的分配由 OPTION_REG 寄存器的 PSA 位控制。要将预分频器分配给 TIMER0，PSA 位必须清 0。

TIMER0 模块具有 8 种预分频比选择，范围为 1:2 至 1:256。可通过 OPTION_REG 寄存器的 PS<2:0>位选择预分频比。要使 TIMER0 模块具有 1:1 的预分频比，必须将预分频器分配给 WDT 模块。

预分频器不可读写。当预分频器分配给 TIMER0 模块时，所有写入 TMR0 寄存器的指令都将使预分频器清零。当预分频器分配给 WDT 时，CLRWDT 指令将同时清零预分频器和 WDT。

18.2.4 在 TIMER0 和 WDT 模块间切换预分频器

将预分频器分配给 TIMER0 或 WDT 后，在切换预分频比时可能会产生无意的器件复位。要将预分频器从分配给 TIMER0 改为分配给 WDT 模块时，必须执行如例 18-1 所示的指令序列。

例 18-1 更改预分频器（TMR0-WDT）

```
        CLR     TMR0            ;TMR0 清零
        CLRWDT                  ;WDT 清零
        LDIA    B'00xx1111'
        LD      OPTION_REG,A
        LDIA    B'00xx1xxx'     ;设置新的预分频器
        LD      OPTION_REG,A
```

将预分频器从分配给 WDT 切换为分配给 TIMER0 模块，应该执行以下指令

```
        CLRWDT                  ;WDT 清零
        LDIA    B'00xx0xxx'     ;设置新的预分频器
        LD      OPTION_REG,A
```

18.2.5 TIMER0 中断

当 TMR0 寄存器从 FFh 溢出至 00h 时，产生 TIMER0 中断。每次 TMR0 寄存器溢出时，不论是否允许 TIMER0 中断，INTCON 寄存器的 T0IF 中断标志位都会置 1。T0IF 位必须在软件中清零。TIMER0 中断允许位是 INTCON 寄存器的 T0IE 位。

注：由于在休眠状态下定时器是关闭的，所以 TIMER0 中断无法唤醒处理器。

18.3 与 TIMER0 相关寄存器

有两个寄存器与 TMR0 相关，8 位定时器 / 计数器（TMR0）, 8 位可编程控制寄存器（OPTION_REG）。

TMR0 为一个 8 位可读写的定时/计数器，OPTION_REG 为一个 8 位只写寄存器，用户可改变 OPTION_REG 的值，来改变 TMR0 的工作模式等。请参看 2.6 关于预分频寄存器（OPTION_REG）的应用。

8 位定时器/计数器 TMR0（01H）

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR0								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

OPTION_REG 寄存器（81H）

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

Bit7 **RBPU**：PORTB 上拉使能位
1=禁止 PORTB 上拉
0=由端口的各个锁存值使能 PORTB 上拉

Bit6 **INTEDG**：中断边沿选择位
1=INT引脚的上升沿触发中断
0=INT 引脚的下降沿触发中断

Bit5 **T0CS**：TIMER0 时钟源选择位
1=T0CKI 引脚上的跳变沿
0=内部指令周期时钟（$F_{OSC}/4$）

Bit4 **T0SE**：TIMER0 时钟源边沿选择位
1=在 T0CKI 引脚信号从高电平跳变到低电平时递增
0=在 T0CKI 引脚信号从低电平跳变到高电平时递增

Bit3 **PSA**：预分频器分配位
1=预分频器分配给 WDT
0=预分频器分配给 TIMER0 模块

Bit2-0 **PS2～PS0**：预分配参数配置位

PS2-PS1-PS0	TMR0 分频比	WDT 分频比
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

19. 定时计数器 TIMER1

19.1 TIMER1 概述

TIMER1 模块是一个 16 位定时器/计数器，具有以下特性：

- 16 位定时器/ 计数器寄存器（TMR1H:TMR1L）
- 3 位预分频器
- 溢出中断
- 溢出时唤醒（仅外部时钟异步模式）
- 捕捉/ 比较功能的时基
- 特殊事件触发功能（带有 ECCP）

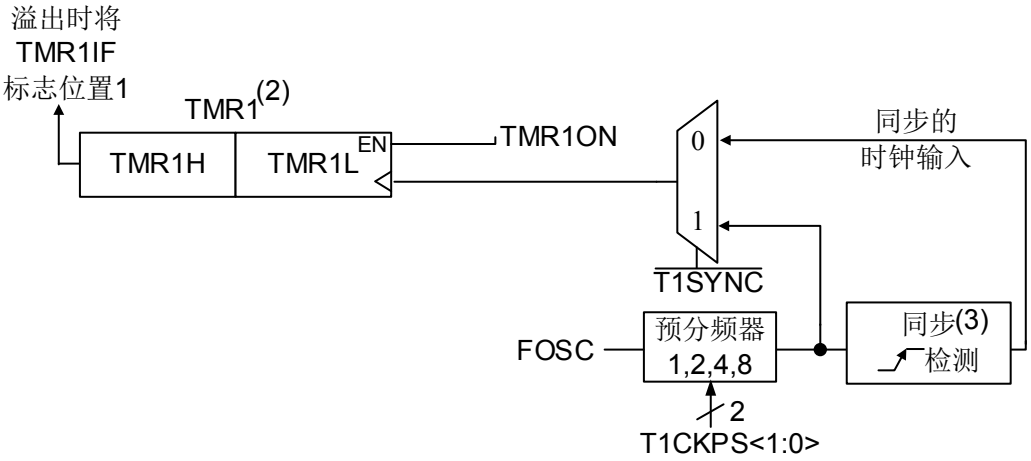


图19.1 TIMER1结构图

19.2 TIMER1 的工作原理

TIMER1 模块是一个通过一对寄存器 TMR1H:TMR1L 访问的 16 位递增计数器。写入 TMR1H 或 TMR1L 可直接更新该计数器。

当 TIMER1 工作时，TMR1H:TMR1L 寄存器将以 FOSC 的倍数为频率递增，具体倍数由 TIMER1 预分频器决定。

19.3 TIMER1 预分频器

TIMER1 具有四种预分频比选择，允许对时钟输入进行 1、2、4 或 8 分频。T1CON 寄存器的 T1CKPS 位控制预分频计数器。不能直接对预分频计数器进行读或写操作。

19.4 TIMER1 中断

一对 TIMER1 寄存器（TMR1H:TMR1L）递增计数到 FFFFh 后，将溢出返回 0000h。当 TIMER1 溢出时，PIR1 寄存器的 TIMER1 中断标志位被置 1。要允许该溢出中断，用户应将以下位置 1：

- PIE1 寄存器中的 TIMER1 中断允许位
- INTCON 寄存器中的 PEIE 位

• INTCON 寄存器中的 GIE 位

在中断服务程序中将 TMR1IF 位清零可以清除该中断。

注： 再次允许该中断前，应将 TMR1H:TMR1L 这对寄存器以及 TMR1IF 位清零。
由于在休眠状态下定时器是关闭的，所以 TIMER1 中断无法唤醒处理器。

19.5 TIMER1 相关寄存器

TIMER1 主要由 3 个 RAM 控制：TMR1 控制寄存器 T1CON；数据寄存器 TMR1L、TMR1H。数据寄存器在赋值的时候必须先赋值低位 TMR1L，再赋值 TMR1H。

TIMER1 数据低位寄存器 TMR1L（0EH）

0EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1L								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	x	x	x	x	x	x	x	x

TIMER1 数据高位寄存器 TMR1H（0FH）

0FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1H								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	x	x	x	x	x	x	x	x

TIMER1 控制寄存器 T1CON（10H）

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	----	----	T1CKPS1	T1CKPS0	----	----	----	TMR1ON
读写	----	----	R/W	R/W	----	----	----	R/W
复位值	----	----	0	0	----	----	----	0

Bit 7 **未用**

Bit 6 **未用 需置零**

Bit 5-4 **T1CKPS<1:0>**：TIMER1 输入时钟预分频比选择位
11 = 1:8 预分频比
10 = 1:4 预分频比
01 = 1:2 预分频比
00 = 1:1 预分频比

Bit3 **未用 需置零**

Bit2 **未用**

Bit1 **未用 需置零**

Bit0 **TMR1ON**：TIMER1 使能位
1 = 使能TIMER1
0 = 禁止TIMER1

20. 定时计数器 TIMER2

20.1 TIMER2 概述

TIMER2 模块是一个 8 位定时器/计数器，具有以下特性：

•8 位定时器寄存器（TMR2）

•8 位周期寄存器（PR2）

•TMR2 与 PR2 匹配时中断

•软件可编程预分频比（1:1、1:4 和 1:16）

•软件可编程后分频比（1:1 至 1:16）

TIMER2 的框图请参见图 20.1。

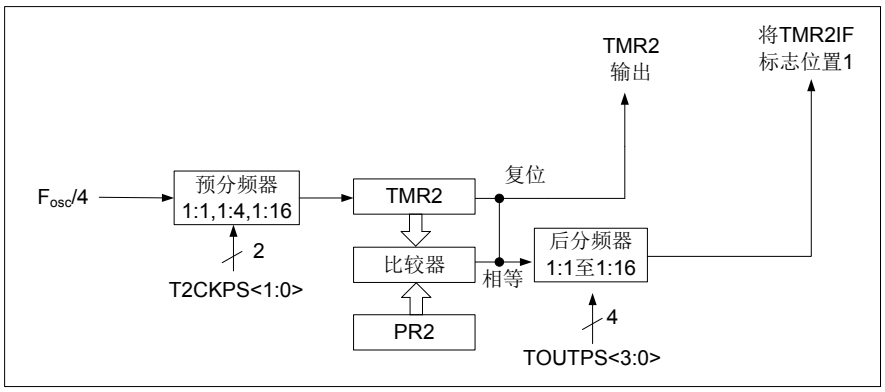


图20.1 TIMER2框图

20.2 TIMER2 的工作原理

TIMER2 模块的时钟输入是系统指令时钟（$F_{OSC}/4$）。时钟被输入到 TIMER2 预分频器，有如下几种分频比可供选择：1:1、1:4 或 1:16。预分频器的输出随后用于使 TMR2 寄存器递增。

持续将 TMR2 和 PR2 的值做比较以确定它们何时匹配。TMR2 将从 00h 开始递增直至与 PR2 中的值匹配。匹配发生时，会发生以下两个事件：

•TMR2 在下一递增周期被复位为 00h

•TIMER2 后分频器递增

TIMER2 与 PR2 比较器的匹配输出随后输入给 TIMER2 的后分频器。后分频器具有 1:1 至 1:16 的预分频比可供选择。TIMER2 后分频器的输出用于使 PIR1 寄存器的 TMR2IF 中断标志位置 1。

TMR2 和 PR2 寄存器均可读写。任何复位时，TMR2 寄存器均被设置为 00h 且 PR2 寄存器被设置为 FFh。

通过将 T2CON 寄存器的 TMR2ON 位置 1 使能 TIMER2。

通过将 TMR2ON 位清零禁止 TIMER2。

TIMER2 预分频器由 T2CON 寄存器的 T2CKPS 位控制。

TIMER2 后分频器由 T2CON 寄存器的 TOUTPS 位控制。

预分步器和后分步器计数器在以下情况下被清零：

•对 TMR2 寄存器执行写操作

•对 T2CON 寄存器执行写操作

•发生任何器件复位（上电复位、看门狗定时器复位或欠压复位）。

注：写 T2CON 不会将 TMR2 清零。

20.3 TIMER2 相关的寄存器

有 2 个寄存器与 TIMER2 相关，分别是数据存储器 TMR2，控制寄存器 T2CON。

TIMER2 数据寄存器 TMR2（11H）

11H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

TIMER2 控制寄存器 T2CON(12H)

12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
读写	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	0	0	0	0	0	0	0

Bit7 **未用，读为 0**

Bit6-3 **TOUTPS<3:0>**：TIMER2输出后分频比选择位
0000=1:1后分频比
0001=1:2后分频比
0010=1:3后分频比
0011=1:4后分频比
0100=1:5后分频比

0101=1:6后分频比
0110=1:7后分频比
0111=1:8后分频比
1000=1:9后分频比
1001=1:10后分频比
1010=1:11后分频比
1011=1:12后分频比
1100=1:13后分频比
1101=1:14后分频比
1110=1:15后分频比
1111=1:16 后分频比

Bit2 **TMR2ON**：TIMER2使能位
1=使能TIMER2
0=禁止 TIMER2

Bit1-0 **T2CKPS<1:0>**：TIMER2时钟预分频比选择位
00=预分频值为1
01=预分频值为4
1x=预分频值为16

21. 模数转换（ADC）

21.1 ADC 概述

模数转换器（ADC）可以将模拟输入信号转换为表示该信号的一个 12 位二进制数。器件使用的模拟输入通道共用一个采样保持电路。采样保持电路的输出与模数转换器的输入相连。模数转换器采用逐次逼近法产生一个 12 位二进制结果，并将该结果保存在 ADC 结果寄存器（ADRESL 和 ADRESH）中。

ADC 参考电压始终为内部产生。

ADC 在转换完成之后可以产生一个中断。

图 21.1 所示为 ADC 的框图。

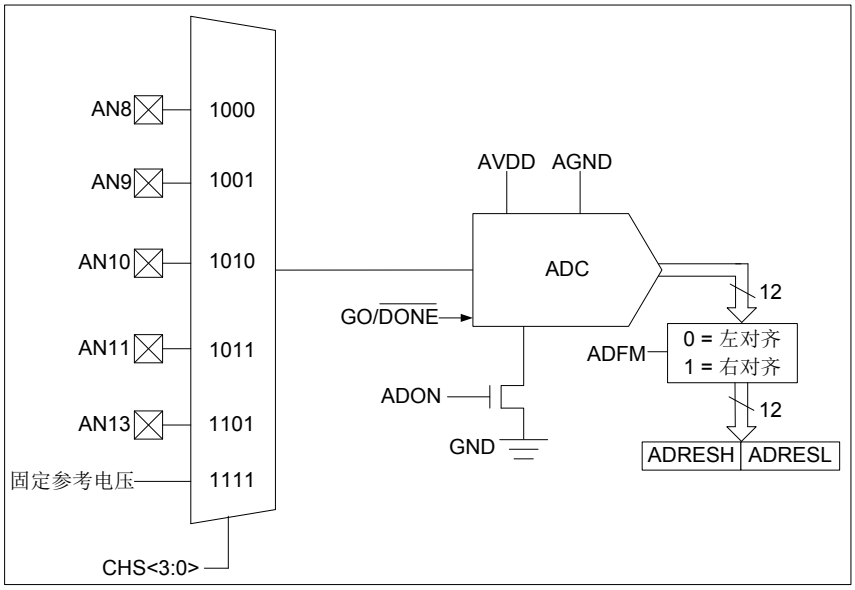


图21.1 ADC框图

21.2 ADC 配置

配置和使用 ADC 时，必须考虑如下因素：

•端口配置

•通道选择

•ADC 转换时钟源

•中断控制

•结果的存储格式

21.2.1 端口配置

ADC 既可以转换模拟信号，又可以转换数字信号。当转换模拟信号时，应该通过将相应的 TRIS 位置 1，将 I/O 引脚配置为输入引脚。更多信息请参见相应的端口章节。

注：对定义为数字输入的引脚施加模拟电压可能导致输入缓冲器出现过电流。

21.2.2 通道选择

由 ADCON0 寄存器的 CHS 位决定将哪个通道连接到采样电路。如果更改了通道，在下一次转换开始前需要一定的延迟。

21.2.3 ADC 参考电压

ADC 的参考电压始终是由芯片的 VDD 和 GND 提供。

21.2.4 转换时钟

可以通过软件设置 ADCON0 寄存器的 ADCS 位来选择转换的时钟源。有以下 4 种可能的时钟频率可供选择：

•$F_{OSC}/8$

•$F_{OSC}/16$

•$F_{OSC}/32$

•F_{RC}（专用内部振荡器）

完成一位转换的时间定义为 TAD。一个完整的 12 位转换需要 49 个 TAD 周期。

必须符合相应的 TAD 规范，才能获得正确的转换结果。

表 21.1 给出了正确选择 ADC 时钟的示例。

注：除非使用 FRC，否则系统时钟频率的任何改变都会改变 ADC 时钟的频率，从而对 ADC 转换结果产生负面影响。

表 21.1 ADC 时钟周期（TAD）与器件工作频率的关系（VDD□5.0V）

ADC 时钟周期		器件频率		
ADC 时钟源	ADCS<1:0>	8MHz	4MHz	1MHz
Fosc/8	00	49.0μs	98.0μs	392.0μs
Fosc/16	01	98.0μs	196.0μs	784.0μs
Fosc/32	10	196.0μs	392.0μs	1.5ms
FRC	11	1-3ms	1-3ms	1-3ms

图注：建议不要使用阴影单元内的值。

21.2.5 ADC 中断

ADC 模块允许在完成模数转换后产生一个中断。ADC 中断标志位是 PIR1 寄存器中的 ADIF 位。ADC 中断允许位是 PIE1 寄存器中的 ADIE 位。ADIF 位必须用软件清零。每次转换结束后 ADIF 位都会被置 1，与是否允许 ADC 中断无关。

不管器件处于工作模式还是休眠模式都可以产生中断。如果器件处于休眠模式，该中断可将器件唤醒。当将器件从休眠状态唤醒后，总是执行 STOP 指令后的下一条指令。如果用户尝试使器件从休眠模式唤醒并按顺序恢复代码执行，则必须禁止全局中断。如果允许全局中断，程序将跳转到中断服务程序处执行。

21.2.6 结果格式化

12 位 A/D 转换的结果可采用两种格式：左对齐或右对齐。由 ADCON0 寄存器的 ADFM 位控制输出格式。

当 ADFM=0 时，AD 转换结果左对齐，AD 转换结果为 12Bit；当 ADFM=1 时，AD 转换结果右对齐，AD 转换结果为 10Bit。

21.3 ADC 工作原理

21.3.1 启动转换

要使能 ADC 模块，必须将 ADCON0 寄存器的 ADON 位置 1，将 ADCON0 寄存器的 GO/DONE 位置 1 开始模数转换。

注：不能用开启 A/D 模块的同一指令将 GO/DONE 位置 1

21.3.2 完成转换

当转换完成时，ADC 模块将：

•清零 GO/DONE 位

•将 ADIF 标志位置 1

•用转换的新结果更新 ADRESH:ADRESL 寄存器

21.3.3 终止转换

如果必须要在转换完成前终止转换，则可用软件清零 GO/DONE 位。不会用尚未完成的模数转换结果更新 ADRESH:ADRESL 寄存器。因此，ADRESH:ADRESL 寄存器将保持上次转换所得到的值。此外，在 A/D 转换终止以后，必须经过 2 个 TAD 的延时才能开始下一次采集。延时过后，将自动开始对选定通道的输入信号进行采集。

注：器件复位将强制所有寄存器进入复位状态。因此，复位会关闭ADC模块并且终止任何待处理的转换。

21.3.4 ADC 在休眠模式下的工作原理

ADC 模块可以工作在休眠模式下。此操作需要将 ADC 时钟源设置为 FRC 选项。如果选择了 FRC 时钟源，ADC 在开始转换之前要多等待一个指令周期。从而允许执行 STOP 指令，以降低转换中的系统噪声。如果允许 ADC 中断，当转换结束时，将使器件从休眠模式唤醒。如果禁止 ADC 中断，即使 ADON 位保持置 1，则转换结束后也还是会关闭 ADC 模块。如果 ADC 时钟源不是 FRC，即使 ADON 位仍保持置 1，执行 STOP 指令还是会中止当前的转换并关闭 A/D 模块。

21.3.5 A/D 转换步骤

如下步骤给出了使用 ADC 进行模数转换的示例：

1、口配置：

•禁止引脚输出驱动器（见 TRIS 寄存器）

•将引脚配置为模拟输入引脚

2、置 ADC 模块：

•选择 ADC 转换时钟

•选择 ADC 输入通道

•选择结果的格式

•启动 ADC 模块

3、置 ADC 中断（可选）：

•清零 ADC 中断标志位
•允许 ADC 中断
•允许外设中断
•允许全局中断

4、待所需的采集时间。
5、GO/DONE 置 1 启动转换。
6、如下方法之一等待 ADC 转换结束：
•查询 GO/DONE 位
•等待 ADC 中断（允许中断）
7、ADC 结果
8、ADC 中断标志位清零（如果允许中断的话，需要进行此操作）。

注：如果用户尝试在使器件从休眠模式唤醒后恢复顺序代码执行，则必须禁止全局中断。

例 21-1AD 转换

```
LDIA    B'10000000'
LD      ADCON1,A
SETB    TRISA,0          ；设置 PORTA.0 为输入口
LDIA    B'11000001'
LD      ADCON0,A
CALL    DELAY            ；延时一段时间
SETB    ADCON0,GO
SZB     ADCON0,GO        ；等待 AD 转换结束
JP      $-1
LD      A,ADRESH         ；保存 AD 转换结果高位
LD      RESULTH,A
LD      A,ADRESL         ；保存 AD 转换结果低位
LD      RESULTL,A
```

21.4 ADC 相关 RAM

主要有 4 个 RAM 与 AD 转换相关，分别是控制寄存器 ADCON0 和 ADCON1，数据寄存器 ADRESH 和 ADRESL。

AD 控制寄存器 ADCON0(1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7-6 **ADCS<1:0>**：A/D转换时钟选择位
00=FOSC/8
01=FOSC/16
10=FOSC/32
11=FRC（由专用的内部振荡器产生频率最高为 32kHz 的时钟）

Bit5-2 **CHS<3:0>**：模拟通道选择位
1000=AN8
1001=AN9
1010=AN10
1011=AN11
1101=AN13
1111=固定参考电压

Bit1 **GO/DONE**：A/D转换状态位
1=A/D转换正在进行。将该位置1启动A/D转换。当A/D转换完成以后，该位由硬件自动清零。
0=A/D 转换完成/或不在进行中

Bit0 **ADON**：ADC使能位
1=使能ADC
0=禁止ADC，不消耗工作电流

AD 控制寄存器 ADCON1(9FH)

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	----	----	----	----	----	----	----
读写	R/W	----	----	----	----	----	----	----
复位值	0	----	----	----	----	----	----	----

Bit7 **ADFM**：A/D转换结果格式选择位
1=右对齐
0=左对齐

Bit6-0 **未用，读为0**

AD 数据寄存器高位 ADRESH(1EH)，ADFM=0

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
读写	R	R	R	R	R	R	R	R
复位值	x	x	x	x	x	x	x	x

Bit7-0 **ADRES<11:4>**：ADC结果寄存器位
12 位转换结果的高 8 位

AD 数据寄存器低位 ADRESL(9EH)，ADFM=0

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES3	ADRES2	ADRES1	ADRES0	----	----	----	----
读写	R	R	R	R	----	----	----	----
复位值	x	x	x	x	----	----	----	----

Bit7-4 **ADRES<3:0>**：ADC结果寄存器位
12 位转换结果的低 4 位
Bit3-0 **未用**

AD 数据寄存器高位 ADRESH(1EH)，ADFM=1

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES11	ADRES10
读写	----	----	----	----	----	----	R	R
复位值	----	----	----	----	----	----	x	x

Bit7-2 **未用**
Bit1-0 **ADRES<11:10>**：ADC结果寄存器位
12 位转换结果的高 2 位

AD 数据寄存器低位 ADRESL(9EH)，ADFM=1

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
读写	R	R	R	R	R	R	R	R
复位值	x	x	x	x	x	x	x	x

Bit7-0 **ADRES<9:2>**：ADC结果寄存器位
12 位转换结果的第 9-2 位

注：在ADFM=1的情况下，AD转换结果只保存12位结果的高10位，其中ADRESH保存高2位，ADRESL保存第2位至第9位。

22. PWM 模块

芯片包含两个 PWM1 和 PWM2。PWM1 和 PWM2 模块的操作基本相同。

22.1 PWM1

PWM 模式可产生频率和占空比都可变化的脉宽调制信号。在 PWM 模式下则需要用定时器 TIMER2。

PWM1 控制寄存器 CCP1CON（17H）

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CCP1CON	----	----	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
读写	----	----	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	----	0	0	0	0	0	0

Bit7-6 **未用**

Bit5-4 **DC1B<1:0>**：PWM占空比的低两位
这两位是 10 位 PWM 占空比的低 2 位。占空比的高 8 位在 CCPR1L 中。

Bit3-0 **CCP1M<3:0>**：CCP1模式选择位
0000= PWM关闭（复位ECCP模块）
0001=未使用（保留）
0010=未使用（保留）
0011=未使用（保留）
0100=未使用（保留）
0101=未使用（保留）
0110=未使用（保留）
0111=未使用（保留）
1000=未使用（保留）
1001=未使用（保留）
1010=未使用（保留）
1011=未使用（保留）
11xx=PWM模式有效；

22.2 PWM2

PWM 模式可产生频率和占空比都可变化的脉宽调制信号。在 PWM 模式下则需要用定时器 TIMER2。

CCP2 控制寄存器 CCP2CON(1DH)

1DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CCP2CON	----	----	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
读写	----	----	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	----	0	0	0	0	0	0

Bit7-6 **未用**

Bit5-4 **DC2B<1:0>**：PWM占空比的低两位
捕捉模式：
未使用。
比较模式：
未使用。
PWM模式：
这两位是 10 位 PWM 占空比的低 2 位。占空比的高 8 位在 CCPR2L 中。

Bit3-0 **CCP2M<3:0>**：CCP2模式选择位
0000= PWM关闭（复位ECCP模块）
0001=未使用（保留）
0010=未使用（保留）
0011=未使用（保留）
0100=未使用（保留）
0101=未使用（保留）
0110=未使用（保留）
0111=未使用（保留）
1000=未使用（保留）
1001=未使用（保留）
1010=未使用（保留）
1011=未使用（保留）
11xx=PWM模式有效；

22.3 PWM 模式

PWM 模式在 CCPx 引脚上产生脉宽调制信号。由以下寄存器确定占空比、周期和分辨率：

•PR2

•T2CON

•CCPRxL

•CCPxCON

在脉宽调制（PWM）模式下，PWM 模块可在 CCPx 引脚上输出分辨率高达 10 位的 PWM 信号。由于 CCPx 引脚与端口数据锁存器复用，必须清零相应的 TRIS 位才能使能 CCPx 引脚的输出驱动器。

注：清零 CCPxCON 寄存器将放弃 CCPx 对 CCPx 引脚的控制权。

图 22.3 给出了 PWM 操作的简化框图。

图 22.4 给出了 PWM 信号的典型波形。

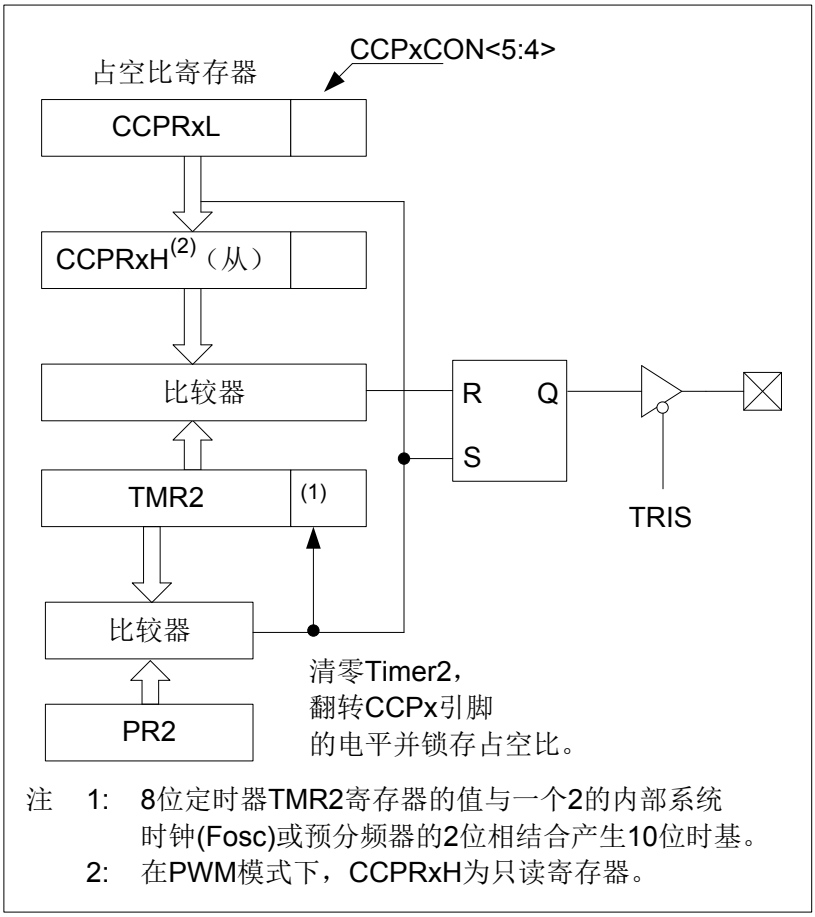


图22.3 PWM简化框图

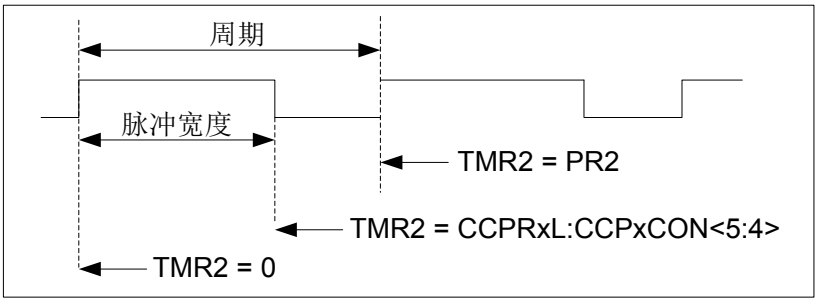


图22.4 CCPPWM输出

22.3.1 PWM 周期

PWM 周期是通过写 TIMER2 的 PR2 寄存器来指定的。

可以使用公式 22-1 计算 PWM 周期：

公式 22-1PWM 周期

PWM周期=[(*PR2)+1]*4*T_{OSC}* (*TMR2*预分频值)
注：$T_{OSC}=1/F_{OSC}$

当 TMR2 等于 PR2 时，在下一个递增计数周期中会发生以下 3 个事件：

•TMR2 被清零

•CCPx 引脚被置 1（例外情况：如果 PWM 占空比=0%，CCPx 引脚将不被置 1）

•PWM 占空比从 CCPRxL 被锁存到 CCPRxH

注：在确定 PWM 频率时不使用 TIMER2 后分频比（见第 10.2 节“TIMER2 的工作原理”）。

22.3.2 PWM 占空比

可通过将一个 10 位值写入以下多个寄存器来指定 PWM 占空比：CCPRxL 寄存器和 CCPxCON 寄存器的 DCxB<1:0>位。CCPRxL 保存占空比的高 8 位，而 CCPxCON 寄存器的 DCxB<1:0>位保存占空比的低 2 位。可以在任何时候写入 CCPRxL 和 CCPxCON 寄存器的 DCxB<1:0>位，但直到 PR2 和 TMR2 中的值匹配（即周期结束）时，占空比的值才被锁存到 CCPRxH 中。在 PWM 模式下，CCPRxH 是只读寄存器。

公式 22-2 用于计算 PWM 脉冲的宽度；公式 12-3 用于计算 PWM 占空比。

公式 22-2 脉冲宽度

脉冲宽度=*(CCPRxL:CCPxCON<5:4>)*TOSC*(TMR2*预分频值)

公式22-3占空比

$$占空比=\frac{(CCPRxL{:}CCPxCON<5{:}4>)}{4(PR2+1)}$$

CCPRxH 寄存器和一个 2 位的内部锁存器用于为 PWM 占空比提供双重缓冲。这种双重缓冲结构极其重要，可以避免在 PWM 操作过程中产生毛刺。

8 位定时器 TMR2 寄存器的值与一个 2 位的内部系统时钟（F_{OSC}）或预分频器的 2 位相结合，产生 10 位时基。当 TIMER2 预分频比为 1:1 时使用系统时钟。

当 10 位时基与 CCPRxH 和 2 位锁存器相结合的值匹配时，CCPx 引脚被清零（见图 12-3）。

22.3.3 PWM 分辨率

分辨率决定在给定周期内的占空比数。例如，10 位分辨率将产生 1024 个离散的占空比，而 8 位分辨率将产生 256 个离散的占空比。

当 PR2 为 255 时，PWM 的最大分辨率为 10 位。如公式 12-4 所示，分辨率是 PR2 寄存器值的函数。

公式 22-4PWM 分辨率

$$\text{分辨率} = \frac{log\ [4(PR2+1)]}{log\ (2)}$$

注：如果脉冲宽度大于周期值，指定的PWM引脚将保持不变。

下列表格给出了在 Fosc=8M 的情况下，PWM 的频率和分辨率的值。

表 12-1PWM 频率和分辨率示例（F_{OSC}=8MHz）

PWM 频率	1.22kHz	4.90kHz	19.61kHz	76.92kHz	153.85kHz	200.0kHz
定时器预分频值（1、4 或 16）	16	4	1	1	1	1
PR2 值	0x65	0x65	0x65	0x19	0x0C	0x09
最高分辨率（位）	8	8	8	6	5	5

22.3.4 休眠模式下的操作

在休眠模式下，TMR2 寄存器将不会递增并且模块的状态将保持不变。如果 CCPx 引脚有输出，将继续保持该输出值不变。当器件被唤醒时，TMR2 将从原先的状态继续工作。

22.3.5 系统时钟频率的改变

PWM 频率是由系统时钟频率产生的。系统时钟频率发生任何改变都会使 PWM 频率发生变化。

22.3.6 复位的影响

任何复位都会将所有端口强制为输入模式，并强制 CCP 寄存器进入其复位状态。

22.3.7 设置 PWM 操作

在将 CCP 模块配置为 PWM 操作模式时应该执行以下步骤：

1、通过将相应的 TRIS 位置 1，禁止 PWM 引脚（CCPx）的输出驱动器，使之成为输入引脚。

2、通过装载 PR2 寄存器设置 PWM 周期。

3、通过用适当的值装载 CCPxCON 寄存器配置 CCP 模块的 PWM 模式。

4、通过装载 CCPRxL 寄存器和 CCPxCON 寄存器中的 DCxB<1:0>位设置 PWM 占空比。

5、配置并启动 TIMER2：

•清零 PIR1 寄存器中的 TMR2IF 中断标志位。

•通过装载 T2CON 寄存器的 T2CKPS 位来设置 TIMER2 预分频比。

•通过将 T2CON 寄存器中的 TMR2ON 位置 1 来使能 TIMER2。

6、在新的 PWM 周期开始后，使能 PWM 输出：

•等待 TIMER2 溢出（PIR1 寄存器中的 TMR2IF 位置 1）。

•通过将相应的 TRIS 位清零，使能 CCPx 引脚输出驱动器。

23. MCU 电气参数

23.1 MCU DC 特性

符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
VDD	工作电压	-	8MHz	2.5	-	3.3	V
		-	4MHz	2.2	-	3.3	V
Idd	工作电流	3V	ADC 使能	-	2	-	mA
		2V	ADC 使能	-	1	-	mA
Istb	静态电流	3V	----	-	0.1	10	μA
		2V	----	-	0.1	5	μA
Vil	低电平输入电压	-	----	-	-	0.3VDD	V
Vih	高电平输入电压	-	----	0.7VDD	-	-	V
Voh	高电平输出电压	-	不带负载	0.9VDD	-	-	V
VoL	低电平输出电压	-	不带负载	-	-	0.1VDD	V
VADI	AD 口输入电压	-	----	0	-	VDD	V
VAD	AD 模块工作电压	-	----	2	-	3.6	V
EAD	AD 转换误差	-	----	-	±2	-	-
Rph	上拉电阻阻值	3V	----	-	35	-	K
		2V	----	-	65	-	K
IoL	输出口灌电流	3V	Vol=0.3VDD	-	50	-	mA
		2V	Vol=0.3VDD	-	25	-	mA
IoH	输出口拉电流	3V	Voh=0.7VDD	-	15	-	mA
		2V	Voh=0.7VDD	-	10	-	mA

23.2 MCU AC 特性

符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
T_{WDT}	WDT 复位时间	3V	---	-	18	-	ms
		2V	---	-	36	-	ms
T_{AD}	AD 转换时间	3V	---	-	49	-	CLK
		2V	---	-	49	-	CLK
F_{INTRC}	内振频率 8MHz	VDD=2 to 3.6V，25℃		-1.5%		1.5%	
		VDD=2 to 3.6V，-40 to 85℃		-2.5%		2.5%	

23.3 指令一览表

助记符	操作	指令周期	标志
控制类-4			
NOP	空操作	1	None
STOP	进入休眠模式	1	TO,PD
CLRWDT	清零看门狗计数器	1	TO,PD
数据传送-4			
LD [R],A	将 ACC 内容传送到 R	1	NONE
LD A,[R]	将 R 内容传送到 ACC	1	Z
TESTZ R	将数据存储器内容传给数据存储器	1	Z
LDIA i	立即数 i 送给 ACC	1	NONE
逻辑运算-16			
CLR A	清零 ACC	1	Z
SET [R]	置位数据存储器 R	1	NONE
CLR [R]	清零数据存储器 R	1	Z
ORA [R]	R 与 ACC 内容做“或”运算，结果存入 ACC	1	Z
ORR [R]	R 与 ACC 内容做“或”运算，结果存入 R	1	Z
ANDA [R]	R 与 ACC 内容做“与”运算，结果存入 ACC	1	Z
ANDR [R]	R 与 ACC 内容做“与”运算，结果存入 R	1	Z
XORA [R]	R 与 ACC 内容做“异或”运算，结果存入 ACC	1	Z
XORR [R]	R 与 ACC 内容做“异或”运算，结果存入 R	1	Z
SWAPA [R]	R 寄存器内容的高低半字节转换，结果存入 ACC	1	NONE
SWAPR [R]	R 寄存器内容的高低半字节转换，结果存入 R	1	NONE
COMA [R]	R 寄存器内容取反，结果存入 ACC	1	Z
COMR [R]	R 寄存器内容取反，结果存入 R	1	Z
XORIA i	ACC 与立即数 i 做“异或”运算，结果存入 ACC	1	Z
ANDIA i	ACC 与立即数 i 做“与”运算，结果存入 ACC	1	Z
ORIA i	ACC 与立即数 i 做“或”运算，结果存入 ACC	1	Z
移位操作，8			
RRCA [R]	数据存储器带进位循环右移一位，结果存入 ACC	1	C
RRCR [R]	数据存储器带进位循环右移一位，结果存入 R	1	C
RLCA [R]	数据存储器带进位循环左移一位，结果存入 ACC	1	C
RLCR [R]	数据存储器带进位循环左移一位，结果存入 R	1	C
RLA [R]	数据存储器不带进位循环左移一位，结果存入 ACC	1	NONE
RLR [R]	数据存储器不带进位循环左移一位，结果存入 R	1	NONE
RRA [R]	数据存储器不带进位循环右移一位，结果存入 ACC	1	NONE
RRR [R]	数据存储器不带进位循环右移一位，结果存入 R	1	NONE
递增递减，4			
INCA [R]	递增数据存储器 R，结果放入 ACC	1	Z
INCR [R]	递增数据存储器 R，结果放入 R	1	Z
DECA [R]	递减数据存储器 R，结果放入 ACC	1	Z
DECR [R]	递减数据存储器 R，结果放入 R	1	Z
位操作，2			
CLRB [R],b	将数据存储器 R 中某位清零	1	NONE
SETB [R],b	将数据存储器 R 中某位置一	1	NONE
查表，2			
TABLE [R]	读取 OTP 内容结果放入 TABLE_DATAH 与 R	2	NONE
TABLEA	读取 OTP 内容结果放入 TABLE_DATAH 与 ACC	2	NONE
数学运算，16			
ADDA [R]	ACC+[R]→ACC	1	Z,C,DC,OV
ADDR [R]	ACC+[R]→R	1	Z,C,DC,OV
ADDCA [R]	ACC+[R]+C→ACC	1	Z,C,DC,OV
ADDCR [R]	ACC+[R]+C→R	1	Z,C,DC,OV
ADDIA i	ACC+i→ACC	1	Z,C,DC,OV
SUBA [R]	[R]-ACC→ACC	1	Z,C,DC,OV

SUBR [R]	[R]-ACC→R	1	Z,C,DC,OV
SUBCA [R]	[R]-ACC-C→ACC	1	Z,C,DC,OV
SUBCR [R]	[R]-ACC-C→R	1	Z,C,DC,OV
SUBIA i	i-ACC→ACC	1	Z,C,DC,OV
HSUBA [R]	ACC-[R]→ACC	1	Z,C,DC,OV
HSUBR [R]	ACC-[R]→R	1	Z,C,DC,OV
HSUBCA [R]	ACC-[R]- $\overline{C}$ →ACC	1	Z,C,DC,OV
HSUBCR [R]	ACC-[R]- $\overline{C}$ →R	1	Z,C,DC,OV
HSUBIA i	ACC-i→ACC	1	Z,C,DC,OV
无条件转移，5			
RET	从子程序返回	2	NONE
RET i	从子程序返回，并将立即数 I 存入 ACC	2	NONE
RETI	从中断返回	2	NONE
CALL ADD	子程序调用	2	NONE
JP ADD	无条件跳转	2	NONE
条件转移，8			
SZB [R],b	如果数据存储器 R 的 b 位为“0”，则跳过下一条指令	1or2	NONE
SNZB [R],b	如果数据存储器 R 的 b 位为“1”，则跳过下一条指令	1or2	NONE
SZA [R]	数据存储器 R 送至 ACC，若内容为”0”,则跳过下一条指令	1or2	NONE
SZR [R]	数据存储器 R 内容为”0”，则跳过下一条指令	1or2	NONE
SZINCA [R]	数据存储器 R 加“1”，结果放入 ACC，若结果为”0”,则跳过下一条指令	1or2	NONE
SZINCR [R]	数据存储器 R 加“1”，结果放入 R，若结果为”0”,则跳过下一条指令	1or2	NONE
SZDECA [R]	数据存储器 R 减“1”，结果放入 ACC，若结果为”0”,则跳过下一条指令	1or2	NONE
SZDECR [R]	数据存储器 R 减“1”，结果放入 R，若结果为”0”,则跳过下一条指令	1or2	NONE

23.4 指令说明

ADDA [R]
操作: 将 R 加 ACC，结果放入 ACC
周期: 1
影响标志位: C，DC，Z，OV
举例:

```
LDIA    09H     ;给 ACC 赋值 09H
LD      R01,A   ;将 ACC 的值（09H）赋给自定义寄存器 R01
LDIA    077H    ;给 ACC 赋值 77H
ADDA    R01     ;执行结果：ACC=09H+77H=80H
```

ADDR [R]
操作: 将 R 加 ACC，结果放入 R
周期: 1
影响标志位: C，DC，Z，OV
举例:

```
LDIA    09H     ;给 ACC 赋值 09H
LD      R01,A   ;将 ACC 的值（09H）赋给自定义寄存器 R01
LDIA    077H    ;给 ACC 赋值 77H
ADDR    R01     ;执行结果：R01=09H+77H=80H
```

ADDCA [R]
操作: 将 R 加 ACC 加 C 位，结果放入 ACC
周期: 1
影响标志位: C，DC，Z，OV
举例:

```
LDIA    09H     ;给 ACC 赋值 09H
LD      R01,A   ;将 ACC 的值（09H）赋给自定义寄存器 R01
LDIA    077H    ;给 ACC 赋值 77H
ADDCA   R01     ;执行结果：ACC=09H+77H+C=80H(C=0)
                ACC=09H+77H+C=81H(C=1)
```

ADDCR [R]
操作: 将 R 加 ACC 加 C 位，结果放入 R
周期: 1
影响标志位: C，DC，Z，OV
举例:

```
LDIA    09H     ;给 ACC 赋值 09H
LD      R01,A   ;将 ACC 的值（09H）赋给自定义寄存器 R01
LDIA    077H    ;给 ACC 赋值 77H
ADDCR   R01     ;执行结果：R01=09H+77H+C=80H(C=0)
                R01=09H+77H+C=81H(C=1)
```

ADDIA i
操作: 将立即数 i 加 ACC，结果放入 ACC
周期: 1
影响标志位: C，DC，Z，OV
举例:

```
LDIA    09H     ;给 ACC 赋值 09H
ADDIA   077H    ;执行结果：ACC=ACC(09H)+i(77H)=80H
```

ANDA [R]
操作: 寄存器 R 跟 ACC 进行逻辑与运算，结果放入 ACC
周期: 1
影响标志位: Z
举例:

```
LDIA    0FH     ;给 ACC 赋值 0FH
LD      R01,A   ;将 ACC 的值(0FH)赋给寄存器 R01
LDIA    77H     ;给 ACC 赋值 77H
```

```
        ANDA        R01         ;执行结果：ACC=(0FHand77H)=07H
```

ANDR [R]
操作: 寄存器 R 跟 ACC 进行逻辑与运算，结果放入 R
周期: 1
影响标志位: Z
举例:

```
        LDIA        0FH         ;给 ACC 赋值 0FH
        LD          R01,A       ;将 ACC 的值(0FH)赋给寄存器 R01
        LDIA        77H         ;给 ACC 赋值 77H
        ANDR        R01         ;执行结果：R01=(0FHand77H)=07H
```

ANDIA i
操作: 将立即数 i 与 ACC 进行逻辑与运算，结果放入 ACC
周期: 1
影响标志位: Z
举例:

```
        LDIA        0FH         ;给 ACC 赋值 0FH
        ANDIA       77H         ;执行结果：ACC=(0FHand77H)=07H
```

CALL ADD
操作: 调用子程序
周期: 2
影响标志位: 无
举例:

```
        CALL        LOOP        ;调用名称定义为”LOOP”的子程序地址
```

CLRA
操作: ACC 清零
周期: 1
影响标志位: Z
举例:

```
        CLR         A           ;执行结果：ACC=0
```

CLR [R]
操作: 寄存器 R 清零
周期: 1
影响标志位: Z
举例:

```
        CLR         R01         ;执行结果：R01=0
```

CLRB [R],b
操作: 寄存器 R 的第 b 位清零
周期: 1
影响标志位: 无
举例:

```
        CLRB        R01,3       ;执行结果：R01 的第 3 位为零
```

CLRWDT
操作: 清零看门狗计数器
周期: 1
影响标志位: TO，PD
举例:

```
        CLRWDT                  ;看门狗计数器清零
```

COMA **[R]**
操作: 寄存器 R 取反，结果放入 ACC
周期: 1
影响标志位: Z
举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
COMA    R01     ;执行结果：ACC=0F5H
```

COMR **[R]**
操作: 寄存器 R 取反，结果放入 R
周期: 1
影响标志位: Z
举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
COMR    R01     ;执行结果：R01=0F5H
```

DECA **[R]**
操作: 寄存器 R 自减 1，结果放入 ACC
周期: 1
影响标志位: Z
举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
DECA    R01     ;执行结果：ACC=(0AH-1)=09H
```

DECR **[R]**
操作: 寄存器 R 自减 1，结果放入 R
周期: 1
影响标志位: Z
举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
DECR    R01     ;执行结果：R01=(0AH-1)=09H
```

HSUBA **[R]**
操作: ACC 减 R，结果放入 ACC
周期: 1
影响标志位: C,DC,Z,OV
举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBA   R01     ;执行结果：ACC=(80H-77H)=09H
```

HSUBR **[R]**
操作: ACC 减 R，结果放入 R
周期: 1
影响标志位: C,DC,Z,OV
举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBR   R01     ;执行结果：R01=(80H-77H)=09H
```

HSUBCA **[R]**
操作: ACC 减 R 减 $\overline{C}$，结果放入 ACC

周期: 1
影响标志位: C,DC,Z,OV
举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBCA  R01     ;执行结果：ACC=(80H-77H- C )=09H(C=0)
                ACC=(80H-77H- C )=08H(C=1)
```

HSUBCR [R]
操作: ACC 减 R 减 $\overline{C}$，结果放入 R
周期: 1
影响标志位: C,DC,Z,OV
举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBCR  R01     ;执行结果：R01=(80H-77H- C )=09H(C=0)
                R01=(80H-77H- C )=08H(C=1)
```

INCA [R]
操作: 寄存器 R 自加 1，结果放入 ACC
周期: 1
影响标志位: Z
举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
INCA    R01     ;执行结果：ACC=(0AH+1)=0BH
```

INCR [R]
操作: 寄存器 R 自加 1，结果放入 R
周期: 1
影响标志位: Z
举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
INCR    R01     ;执行结果：R01=(0AH+1)=0BH
```

JP ADD
操作: 跳转到 add 地址
周期: 2
影响标志位: 无
举例:

```
JP      LOOP    ;跳转至名称定义为"LOOP"的子程序地址
```

LD A，[R]
操作: 将 R 的值赋给 ACC
周期: 1
影响标志位: Z
举例:

```
LD      A,R01   ;将寄存器 R0 的值赋给 ACC
LD      R02,A   ;将 ACC 的值赋给寄存器 R02，实现了数据从 R01→R02 的移动
```

LD [R]，A
操作: 将 ACC 的值赋给 R
周期: 1
影响标志位: 无

举例:

```
        LDIA    09H         ;给 ACC 赋值 09H
        LD      R01,A       ;执行结果：R01=09H
```

LDIA i

操作: 立即数 i 赋给 ACC
周期: 1
影响标志位: 无
举例:

```
        LDIA    0AH         ;ACC 赋值 0AH
```

NOP

操作: 空指令
周期: 1
影响标志位: 无
举例:

```
        NOP
```

ORIA i

操作: 立即数与 ACC 进行逻辑或操作，结果赋给 ACC
周期: 1
影响标志位: Z
举例:

```
        LDIA    0AH         ;ACC 赋值 0AH
        ORIA    030H        ;执行结果：ACC=(0AHor30H)=3AH
```

ORA [R]

操作: 寄存器 R 跟 ACC 进行逻辑或运算，结果放入 ACC
周期: 1
影响标志位: Z
举例:

```
        LDIA    0AH         ;给 ACC 赋值 0AH
        LD      R01,A       ;将 ACC(0AH)赋给寄存器 R01
        LDIA    30H         ;给 ACC 赋值 30H
        ORA     R01         ;执行结果：ACC=(0AHor30H)=3AH
```

ORR [R]

操作: 寄存器 R 跟 ACC 进行逻辑或运算，结果放入 R
周期: 1
影响标志位: Z
举例:

```
        LDIA    0AH         ;给 ACC 赋值 0AH
        LD      R01,A       ;将 ACC(0AH)赋给寄存器 R01
        LDIA    30H         ;给 ACC 赋值 30H
        ORR     R01         ;执行结果：R01=(0AHor30H)=3AH
```

RET

操作: 从子程序返回
周期: 2
影响标志位: 无
举例:

```
        CALL    LOOP        ;调用子程序 LOOP
        NOP                 ;RET 指令返回后将执行这条语句
        …                   ;其它程序
LOOP:
        …                   ;子程序
        RET                 ;子程序返回
```

RET i

操作: 从子程序带参数返回，参数放入 ACC

周期: 2
影响标志位: 无
举例:

```
        CALL    LOOP        ;调用子程序 LOOP
        NOP                 ;RET 指令返回后将执行这条语句
        …                   ;其它程序
LOOP:
        …                   ;子程序
        RET     35H         ;子程序返回,ACC=35H
```

RETI
操作: 中断返回
周期: 2
影响标志位: 无
举例:

```
INT_START                   ;中断程序入口
        …                   ;中断处理程序
        RETI                ;中断返回
```

RLCA [R]
操作: 寄存器 R 带 C 循环左移一位，结果放入 ACC
周期: 1
影响标志位: C
举例:

```
        LDIA    03H         ;ACC 赋值 03H
        LD      R01,A       ;ACC 值赋给 R01,R01=03H
        RLCA    R01         ;操作结果：ACC=06H(C=0);
                             ACC=07H(C=1)
                             C=0
```

RLCR [R]
操作: 寄存器 R 带 C 循环左移一位，结果放入 R
周期: 1
影响标志位: C
举例:

```
        LDIA    03H         ;ACC 赋值 03H
        LD      R01,A       ;ACC 值赋给 R01,R01=03H
        RLCR    R01         ;操作结果：R01=06H(C=0);
                             R01=07H(C=1);
                             C=0
```

RLA [R]
操作: 寄存器 R 不带 C 循环左移一位，结果放入 ACC
周期: 1
影响标志位: 无
举例:

```
        LDIA    03H         ;ACC 赋值 03H
        LD      R01,A       ;ACC 值赋给 R01,R01=03H
        RLA     R01         ;操作结果：ACC=06H
```

RLR [R]
操作: 寄存器 R 不带 C 循环左移一位，结果放入 R
周期: 1
影响标志位: 无
举例:

```
        LDIA    03H         ;ACC 赋值 03H
        LD      R01,A       ;ACC 值赋给 R01,R01=03H
        RLR     R01         ;操作结果：R01=06H
```

RRCA [R]

操作: 寄存器 R 带 C 循环右移一位，结果放入 ACC
周期: 1
影响标志位: C
举例:

```
LDIA    03H     ;ACC 赋值 03H
LD      R01,A   ;ACC 值赋给 R01,R01=03H
RRCA    R01     ;操作结果：ACC=01H(C=0);
                        ACC=081H(C=1);
                        C=1
```

RRCR [R]
操作: 寄存器 R 带 C 循环右移一位，结果放入 R
周期: 1
影响标志位: C
举例:

```
LDIA    03H     ;ACC 赋值 03H
LD      R01,A   ;ACC 值赋给 R01,R01=03H
RRCR    R01     ;操作结果：R01=01H(C=0);
                        R01=81H(C=1);
                        C=1
```

RRA [R]
操作: 寄存器 R 不带 C 循环右移一位，结果放入 ACC
周期: 1
影响标志位: 无
举例:

```
LDIA    03H     ;ACC 赋值 03H
LD      R01,A   ;ACC 值赋给 R01,R01=03H
RRA     R01     ;操作结果：ACC=81H
```

RRR [R]
操作: 寄存器 R 不带 C 循环右移一位，结果放入 R
周期: 1
影响标志位: 无
举例:

```
LDIA    03H     ;ACC 赋值 03H
LD      R01,A   ;ACC 值赋给 R01,R01=03H
RRR     R01     ;操作结果：R01=81H
```

SET [R]
操作: 寄存器 R 所有位置 1
周期: 1
影响标志位: 无
举例:

```
SET     R01     ;操作结果：R01=0FFH
```

SETB [R],b
操作: 寄存器 R 的第 b 位置 1
周期: 1
影响标志位: 无
举例:

```
CLR     R01     ;R01=0
SETB    R01,3   ;操作结果：R01=08H
```

STOP
操作: 进入休眠状态
周期: 1
影响标志位: TO，PD
举例:

```
STOP            ;芯片进入省电模式，CPU、振荡器停止工作，IO 口保持原来状态。
```

SUBIA i
操作: 立即数 i 减 ACC，结果放入 ACC
周期: 1
影响标志位: C,DC,Z,OV
举例:

```
LDIA    077H    ;ACC 赋值 77H
SUBIA   80H     ;操作结果：ACC=80H-77H=09H
```

SUBA [R]
操作: 寄存器 R 减 ACC，结果放入 ACC
周期: 1
影响标志位: C,DC,Z,OV
举例:

```
LDIA    080H    ;ACC 赋值 80H
LD      R01,A   ;ACC 的值赋给 R01，R01=80H
LDIA    77H     ;ACC 赋值 77H
SUBA    R01     ;操作结果：ACC=80H-77H=09H
```

SUBR [R]
操作: 寄存器 R 减 ACC，结果放入 R
周期: 1
影响标志位: C,DC,Z,OV
举例:

```
LDIA    080H    ;ACC 赋值 80H
LD      R01,A   ;ACC 的值赋给 R01，R01=80H
LDIA    77H     ;ACC 赋值 77H
SUBR    R01     ;操作结果：R01=80H-77H=09H
```

SUBCA [R]
操作: 寄存器 R 减 ACC 减 C，结果放入 ACC
周期: 1
影响标志位: C,DC,Z,OV
举例:

```
LDIA    080H    ;ACC 赋值 80H
LD      R01,A   ;ACC 的值赋给 R01，R01=80H
LDIA    77H     ;ACC 赋值 77H
SUBCA   R01     ;操作结果：ACC=80H-77H-C=09H(C=0);
                ACC=80H-77H-C=08H(C=1);
```

SUBCR [R]
操作: 寄存器 R 减 ACC 减 C，结果放入 R
周期: 1
影响标志位: C,DC,Z,OV
举例:

```
LDIA    080H    ;ACC 赋值 80H
LD      R01,A   ;ACC 的值赋给 R01，R01=80H
LDIA    77H     ;ACC 赋值 77H
SUBCR   R01     ;操作结果：R01=80H-77H-C=09H(C=0)
                R01=80H-77H-C=08H(C=1)
```

SWAPA [R]
操作: 寄存器 R 高低半字节交换，结果放入 ACC
周期: 1
影响标志位: 无
举例:

```
LDIA    035H    ;ACC 赋值 35H
LD      R01,A   ;ACC 的值赋给 R01，R01=35H
SWAPA   R01     ;操作结果：ACC=53H
```

SWAPR [R]
操作: 寄存器 R 高低半字节交换，结果放入 R
周期: 1
影响标志位: 无
举例:

```
LDIA    035H    ;ACC 赋值 35H
LD      R01,A   ;ACC 的值赋给 R01，R01=35H
SWAPR   R01     ;操作结果：R01=53H
```

SZB [R],b
操作: 判断寄存器 R 的第 b 位，为 0 间跳，否则顺序执行
周期: 1or2
影响标志位: 无
举例:

```
SZB     R01,3   ;判断寄存器 R01 的第 3 位
JP      LOOP    ;R01 的第 3 位为 1 才执行这条语句，跳转至 LOOP
JP      LOOP1   ;R01 的第 3 位为 0 时间跳，执行这条语句，跳转至 LOOP1
```

SNZB [R],b
操作: 判断寄存器 R 的第 b 位，为 1 间跳，否则顺序执行
周期: 1or2
影响标志位: 无
举例:

```
SNZB    R01,3   ;判断寄存器 R01 的第 3 位
JP      LOOP    ;R01 的第 3 位为 0 才执行这条语句，跳转至 LOOP
JP      LOOP1   ;R01 的第 3 位为 1 时间跳，执行这条语句，跳转至 LOOP1
```

SZA [R]
操作: 将寄存器 R 的值赋给 ACC，若 R 为 0 则间跳，否则顺序执行
周期: 1or2
影响标志位: 无
举例:

```
SZA     R01     ;R01→ACC
JP      LOOP    ;R01 不为 0 时执行这条语句，跳转至 LOOP
JP      LOOP1   ;R01 为 0 时间跳，执行这条语句，跳转至 LOOP1
```

SZR [R]
操作: 将寄存器 R 的值赋给 R，若 R 为 0 则间跳，否则顺序执行
周期: 1or2
影响标志位: 无
举例:

```
SZR     R01     ;R01→R01
JP      LOOP    ;R01 不为 0 时执行这条语句，跳转至 LOOP
JP      LOOP1   ;R01 为 0 时间跳执行这条语句，跳转至 LOOP1
```

SZINCA [R]
操作: 将寄存器 R 自加 1，结果放入 ACC，若结果为 0，则跳过下一条语句，否则顺序执行
周期: 1or2
影响标志位: 无
举例:

```
SZINCA  R01     ;R01+1→ACC
JP      LOOP    ;ACC 不为 0 时执行这条语句，跳转至 LOOP
JP      LOOP1   ;ACC 为 0 时执行这条语句，跳转至 LOOP1
```

SZINCR [R]
操作: 将寄存器 R 自加 1，结果放入 R，若结果为 0，则跳过下一条语句，否则顺序执行
周期: 1or2
影响标志位: 无
举例:

```
SZINCR      R01         ;R01+1→R01
JP          LOOP        ;R01 不为 0 时执行这条语句，跳转至 LOOP
JP          LOOP1       ;R01 为 0 时执行这条语句，跳转至 LOOP1
```

SZDECA [R]

操作: 将寄存器 R 自减 1，结果放入 ACC，若结果为 0，则跳过下一条语句，否则顺序执行
周期: 1or2
影响标志位: 无
举例:

```
SZDECA      R01         ;R01-1→ACC
JP          LOOP        ;ACC 不为 0 时执行这条语句，跳转至 LOOP
JP          LOOP1       ;ACC 为 0 时执行这条语句，跳转至 LOOP1
```

SZDECR [R]

操作: 将寄存器 R 自减 1，结果放入 R，若结果为 0，则跳过下一条语句，否则顺序执行
周期: 1or2
影响标志位: 无
举例:

```
SZDECR      R01         ;R01-1→R01
JP          LOOP        ;R01 不为 0 时执行这条语句，跳转至 LOOP
JP          LOOP1       ;R01 为 0 时执行这条语句，跳转至 LOOP1
```

TABLE [R]

操作: 查表，查表结果低 8 位放入 R，高位放入专用寄存器 TABLE_SPH
周期: 2
影响标志位: 无
举例:

```
LDIA        01H         ;ACC 赋值 01H
LD          TABLE_SPH,A ;ACC 值赋给表格高位地址，TABLE_SPH=1
LDIA        015H        ;ACC 赋值 15H
LD          TABLE_SPL,A ;ACC 值赋给表格地位地址，TABLE_SPL=15H
TABLE       R01         ;查表 0115H 地址，操作结果：TABLE_DATAH=12H，R01=34H
…
ORG         0115H
DW          1234H
```

TABLEA

操作: 查表，查表结果低 8 位放入 ACC，高位放入专用寄存器 TABLE_SPH
周期: 2
影响标志位: 无
举例:

```
LDIA        01H         ;ACC 赋值 01H
LD          TABLE_SPH,A ;ACC 值赋给表格高位地址，TABLE_SPH=1
LDIA        015H        ;ACC 赋值 15H
LD          TABLE_SPL,A ;ACC 值赋给表格地位地址，TABLE_SPL=15H
TABLEA                  ;查表 0115H 地址，操作结果：TABLE_DATAH=12H，ACC=34H
…
ORG         0115H
DW          1234H
```

TESTZ [R]

操作: 将 R 的值赋给 R,用以影响 Z 标志位
周期: 1
影响标志位: Z
举例:

```
TESTZ       R0          ;将寄存器 R0 的值赋给 R0，用于影响 Z 标志位
SZB         STATUS,Z    ;判断 Z 标志位，为 0 间跳
```

```
        JP      Add1        ;当寄存器 R0 为 0 的时候跳转至地址 Add1
        JP      Add2        ;当寄存器 R0 不为 0 的时候跳转至地址 Add1
```

XORIA i

操作: 立即数与 ACC 进行逻辑异或运算，结果放入 ACC
周期: 1
影响标志位: Z
举例:

```
        LDIA    0AH         ;ACC 赋值 0AH
        XORIA   0FH         ;执行结果：ACC=05H
```

XORA [R]

操作: 寄存器 R 与 ACC 进行逻辑异或运算，结果放入 ACC
周期: 1
影响标志位: Z
举例:

```
        LDIA    0AH         ;ACC 赋值 0AH
        LD      R01,A       ;ACC 值赋给 R01,R01=0AH
        LDIA    0FH         ;ACC 赋值 0FH
        XORA    R01         ;执行结果：ACC=05H
```

XORR [R]

操作: 寄存器 R 与 ACC 进行逻辑异或运算，结果放入 R
周期: 1
影响标志位: Z
举例:

```
        LDIA    0AH         ;ACC 赋值 0AH
        LD      R01,A       ;ACC 值赋给 R01,R01=0AH
        LDIA    0FH         ;ACC 赋值 0FH
        XORR    R01         ;执行结果：R01=05H
```

24. 典型应用电路（参考）

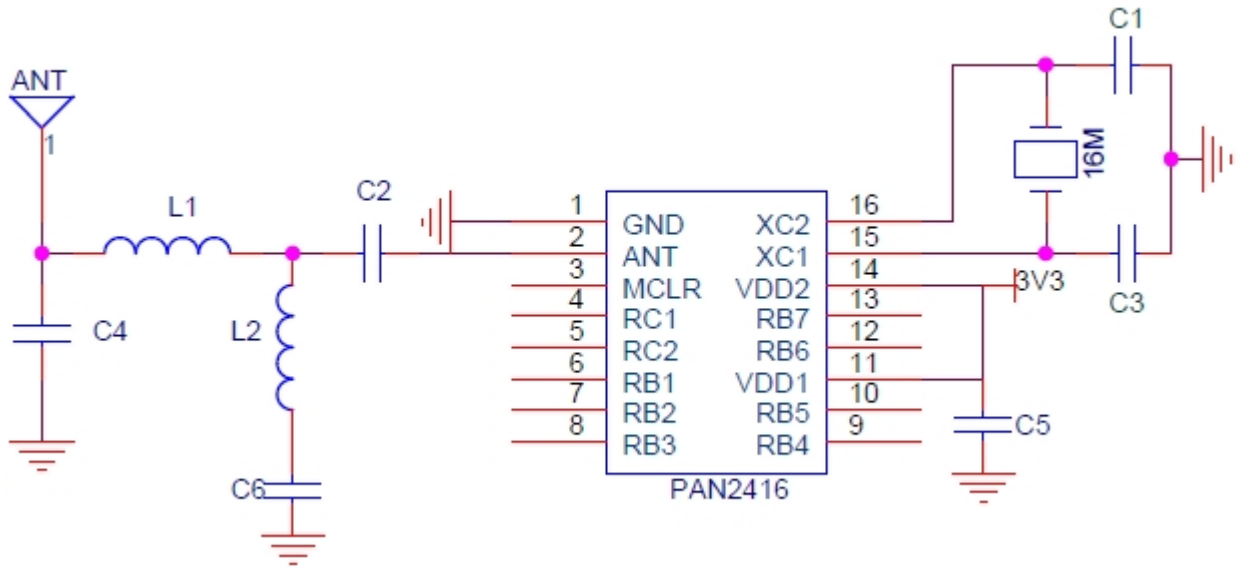


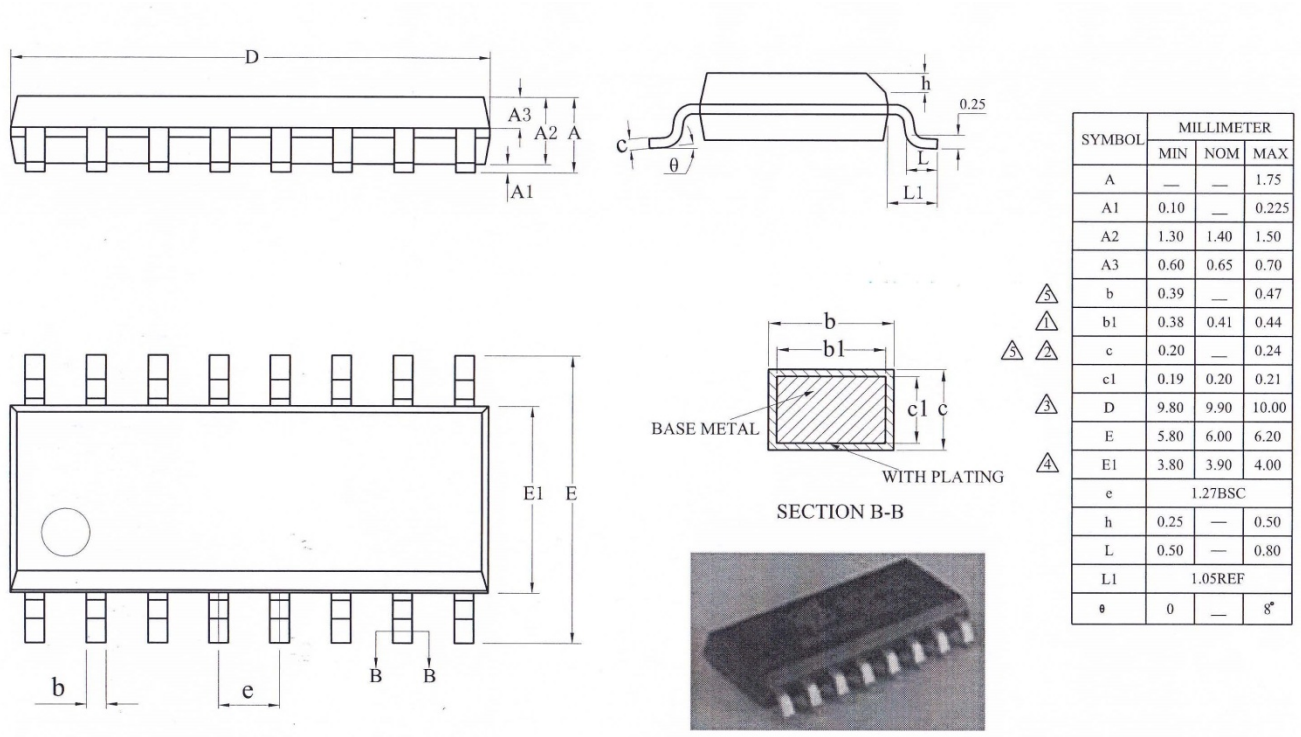
图11 PAN2416AV的应用电路

注：射频匹配部分为经验证的单/双面板上测试发射接收均可通过安规认证的方式。

物料编号	备注
C1 / C3	谐振电容，根据不同型号的晶振进行微调，推荐范围 15~36pF
C5	0.1uF，对电源进行滤波
C2	推荐 3.3pF，可选择范围在 2~4pF
L1	5.6nH
L2	2.2nH
C4	0.5pF
C6	0.5pF

25. 封装尺寸

PAN2416AV采用SOP16封装方式，封装尺寸如图25.1所示。



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.75
A1	0.10	—	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	—	0.47
b1	0.38	0.41	0.44
c	0.20	—	0.24
c1	0.19	0.20	0.21
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	—	0.50
L	0.50	—	0.80
L1	1.05REF		
θ	0	—	8°

图25.1 PAN2416AV封装尺寸

26. 联系方式

上海磐启微电子
地址：上海市张江高科技园区盛夏路666号E栋802室
电话：+86-021-50802372

苏州磐启微电子
地址：苏州工业园区东平街282号汉嘉大厦3002室
电话：+86-0512-80968880

磐启微电子（深圳）
地址：深圳南山区科技路11号桑达科技园伟杰大厦106室
电话：+86-0755-26403799

www.panchip.com