

# XP9952HP

-----2.4GHz 高集成度无线收发 SOC 芯片

# 目录

1. 无线收发 SOC 特点	4
1.1 系统功能	4
1.2 CPU 特点概述	4
1.3 封装信息	5
2. 系统概述和方框图	5
3. 引脚说明及应用电路	6
3.1 引脚图	6
3.2 应用图	7
4. 芯片电气特性	7
4.1 电气特性	7
4.2 极限值	9
5. 功能描述	10
5.1 存储器结构	10
5.1.1 程序存储器	10
5.1.2 数据存储器	10
5.2 寄存器操作	12
5.2.1 INDF (间接寻址寄存器)	12
5.2.2 TMR0 (定时/计数器 Time lock/Counter register)	12
5.2.3 PCL (Low Bytes of Program Counter) & Stack	13
5.2.4 STATUS (状态字寄存器)	14
5.2.5 FSR (间接寻址指针)	15
5.2.6 PORTA, PORTB (Port 寄存器)	15
5.2.7 PCON (电源控制寄存器)	15
5.2.8 WUCON (Port B 输入改变/唤醒控制寄存器)	16
5.2.9 PCHBUF (PC 指针高位缓冲区)	16
5.2.10 PDCON (I/O 下拉控制寄存器)	16
5.2.11 ODCON (I/O 开路控制寄存器)	17
5.2.12 PHCON (I/O 上拉控制寄存器)	17
5.2.13 INTEN (中断屏蔽寄存器)	18
5.2.14 INTFLAG (中断标志寄存器)	18
5.2.15 ACC (Accumulator)累加器	18
5.2.16 OPTION Register (选项寄存器)	18
5.2.17 IOSTA & IOSTB (I/O 口控制寄存器)	19
5.3 I/O Ports	20
5.4 Timer0/WDT & Prescler 2.3.1 Timer0	22

---

5.4.1	使用内部时钟: 定时模式.....	22
5.4.2	使用外部时钟: 计数模式.....	22
5.4.3	看门狗定时器 (WDT).....	22
5.4.4	Prescaler (预置器) .....	22
5.5	中断方式.....	23
5.5.1	外部中断.....	23
5.5.2	Timer0 中断 .....	23
5.5.3	Port B 输入改变中断 .....	23
5.6	省电模式 (SLEEP).....	24
5.7	复位.....	24
5.7.1	上电复位计数器 (Power-up Reset Timer PWRT).....	25
5.7.2	振荡启动计数器(Oscillator Start-up Timer OST) .....	25
5.7.3	复位顺序.....	25
5.8	十六进制转化为十进制(Hexadecimal Convert to Decimal,HCD).....	27
5.9	振荡器配置 (Oscillator Configurations) .....	1
5.10	配置选项.....	3
5.11	RF 通信模块.....	5
5.11.1	工作模式及状态机.....	5
5.11.2	三线制 SPI 接口.....	6
5.11.3	RF 模块寄存器默认值及优化值.....	7
5.11.4	一般通信流程.....	8
5.11.5	输出功率配置.....	9
6.	指令集合.....	10
7.	版本更新历史.....	20

# 1. 无线收发 SOC 特点

## 1.1 系统功能

- ◆ 内部合封 PMS152 单片机
- ◆ 2500B 和 PMS152 按三线 SPI 通讯打线
- ◆ 1.25KW OTP 程序存储器
- ◆ 80 Byte 数据存储器
- ◆ 一个硬件 16-位计数器
- ◆ 一个 8 位定时器（可作为 PWM 生成器，PWM 分辨率可以为 6 位、7 位或 8 位）
- ◆ 一组三连套 11 位 SuLED (Super LED) PWM 生成器及计数器
- ◆ 一个硬件比较器
- ◆ 14 个 IO 引脚并带有上拉电阻选项
- ◆ 提供三组不同的 IO 驱动能力以满足不同的应用需求
  - (1) PB3, PB5, PB7 驱动电流/灌电流= 5mA/ 30mA
  - (2) 其他 IO（除 PA5 外）驱动电流/灌电流= 5mA/ 10mA
  - (3) PA5 灌电流= 10mA
- ◆ 每个 IO 引脚都可设定唤醒功能
- ◆ 时钟源：内部高频 RC 振荡器，内部低频 RC 振荡器和外部晶体震荡
- ◆ 对所有带有唤醒功能的 IO，都支持两种可选择的唤醒速度：正常唤醒和快速唤醒
- ◆ 8 段 LVR 复位电压设定：4.5V, 3.5V, 3.0V, 2.75V, 2.5V, 2.2V, 2.0V 及 1.8V
- ◆ 2 组可选的外部中断引脚：PA0/ PB5, PB0/ PA4
- ◆ Band-gap 电路提供 1.2V 参考电压
  
- ◆ 工作电压范围：2.0V – 3.6V
- ◆ 无线模块工作在 2400—2483MHz 世界通用 ISM 频段
- ◆ 无线模块自动处理数据包
- ◆ 无线模块通信数据率 1Mbps
- ◆ 无线模块输出功率可编程，调节范围广：-24dBm -- +8dBm
- ◆ 无线模块 1M 模式的灵敏度为-89dBm
- ◆ 抗干扰性好，接收滤波器的邻道抑制度高，接收机选择性好
- ◆ 无线模块功耗低，性能优异，外围器件少
- ◆ 工作温度：-20°C~70°C

## 1.2 CPU 特点概述

- ◆ 单一处理单元工作模式
- ◆ 提供 86 个有效指令
- ◆ 大部分都是 1T（单周期）指令
- ◆ 可程序设定的堆栈指针和堆栈深度
- ◆ 数据存取支持直接和间接寻址模式，用数据存储器即可当作间接寻址模式的数据指针(index pointer)
- ◆ IO 地址以及存储地址空间互相独立

## 1.3 封装信息

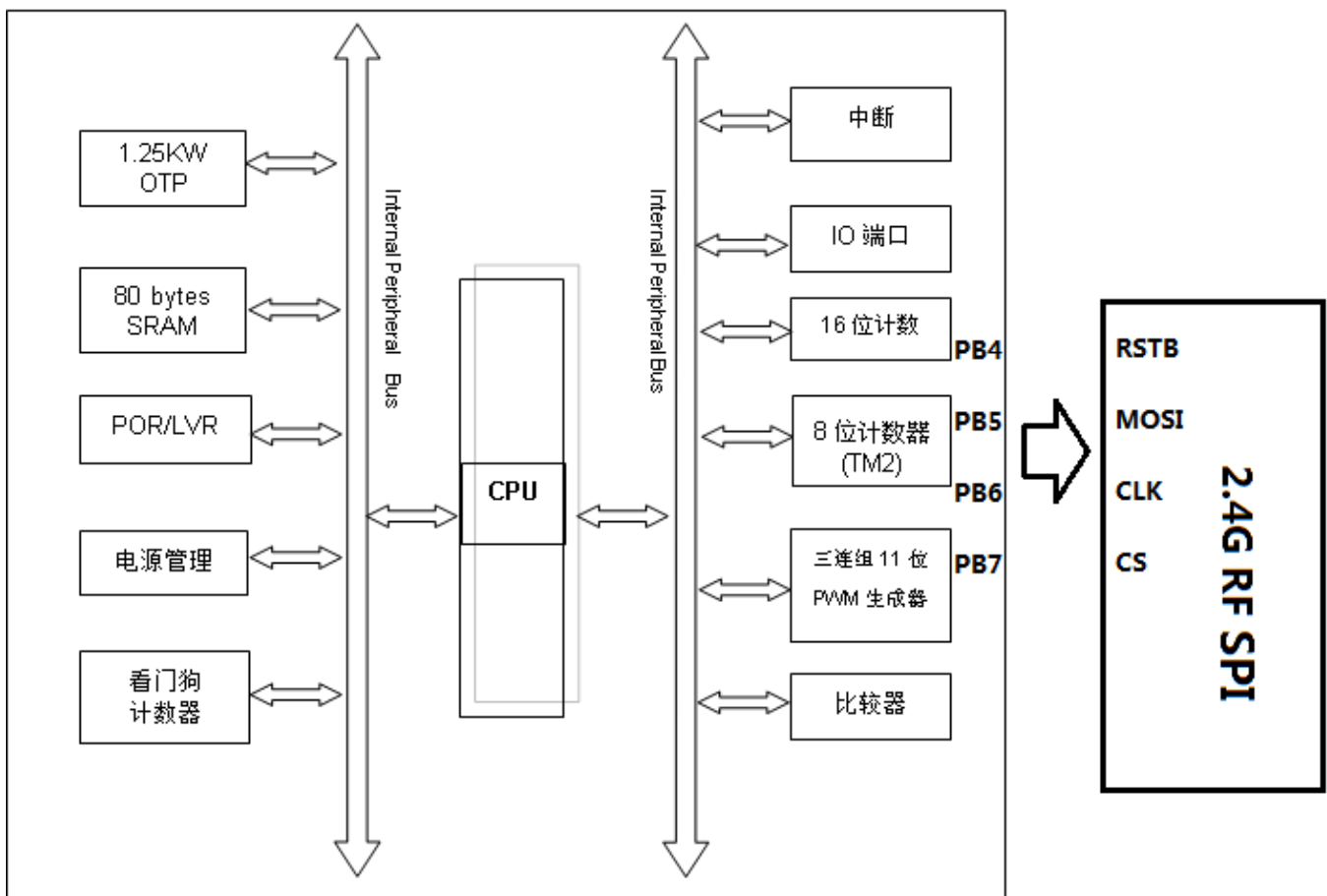
◆ XP9952HP: SOP16

## 2. 系统概述和方框图

XP9952HP是一个IO类型，以OTP为程序存储基础的自带2.4G高速无线收发模块的单片机。无线收发模块工作在 2.402--2.483GHz 世界通用ISM 频段，它集成射频收发机、频率发生器、晶体振荡器、调制解调器等功能模块，并且支持带 ACK 的通信模式。发射输出功率、工作频道以及通信数据率均可配置。它采用 GFSK 通信方式，支持自动应答及自动重传，支持RSSI检测功能，自带扰码和CRC校验功能。可应用于无线鼠标键盘、电视和机顶盒遥控器、无线游戏手柄、有源无线标签、智能家居及安防系统、遥控玩具等领域。

XP9952HP内部集成PMS152单片机。PMS152系列是一款IO类型，完全静态以OTP为程序基础的CMOS 8-bit 微处理器。它运用RISC的架构并且所有的指令架构的执行周期都是一个指令周期，只有少部分指令需要两个指令周期。PMS152内置1.25KW OTP程序存储器以及80字节数据存储器， 内置一个硬件比较器用于比较两个管脚之间或者管脚与内部参考电压VinternalR之间或者管脚与bandgap参考电压之间的信号电压。

PMS152 同时提供三个硬件计数器：一个16位计数器，一个8位PWM计数器，和一组全新设计的11位SuLED PWM生成器及计数器(PWMG0, PWMG1和PWMG2)。



## 3. 引脚说明及应用电路

### 3.1 引脚图

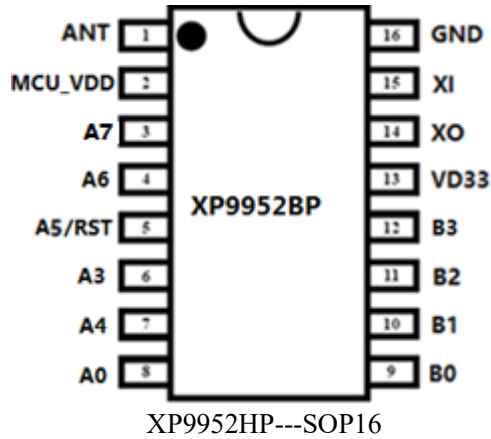


表3.1 MCU/RF芯片内部打线对照表(三线制SPI)

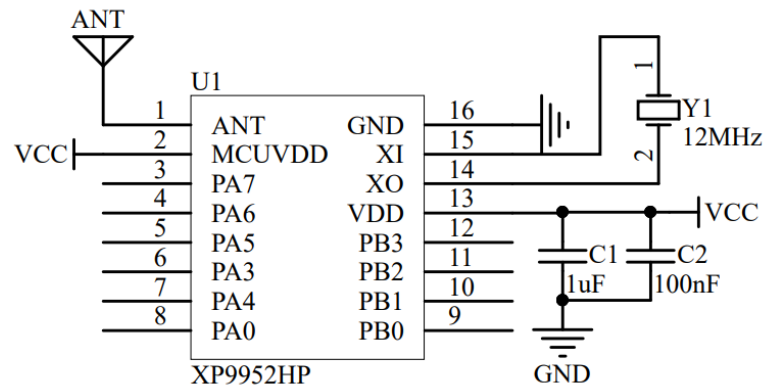
MCU/RF 内部打线引脚对照表	
MCU 引脚名	RF 引脚名
B7	CS
B6	CLK
B5	MOSI
B4	RSTB

表3.2 芯片引脚说明

序号	引脚名称	引脚类型	功能描述
1	ANT	Analog	天线端口
2	MCU_VDD	电源	外接 3.3V MCU 电源 (2V~3.6V)
3	A7	IO	参考 PMS152 手册
4	A6	IO	参考 PMS152 手册
5	A5	OPEN DRAIN	参考 PMS152 手册
6	A3	IO	参考 PMS152 手册
7	A4	IO	参考 PMS152 手册
8	A0	IO	参考 PMS152 手册
9	B0	IO	参考 PMS152 手册
10	B1	IO	参考 PMS152 手册
11	B2	IO	参考 PMS152 手册
12	B3	IO	参考 PMS152 手册

13	VDD33	Analog	RF 芯片 3.3V 电源输入 (2V~3.6V)
14	XO	Analog	晶体振荡器输出
15	XI	Analog	晶体振荡器输入
16	GND	Analog	地

## 3.2 应用图



## 4 芯片电气特性

### 4.1 电气特性

特性	条件(除另有规定外, VCC = 3.3V, TA=25°C)	参数值			单位	条件
		最小	典型	最大		
VDD	工作电源电压	2.2	3.3	3.6	V	
RF 特性						
ICC	休眠模式		1		uA	
	待机模式		600		uA	
	发射模式 (6dBm)		30		mA	
	接收模式		16		mA	
$f_{OP}$	工作频率	2402		2530	MHz	
$f_{XTAL}$	晶振频率		12		MHz	
$PLL\_stabl$ $e$	PLL 稳定时间		150		us	
	码率		1		Mbps	
FCH	频道间隔		1		MHz	
PRF_MAX	最大输出功率		8		dBm	
PRF	典型输出功率		6		dBm	
PRFC	输出功率范围	-24		8	dBm	
$RXSENS2$	接收灵敏度 (0.1%BER)		-89		dBm	

$C/I_{CO}$	同道干扰			9			dBc		
$C/I_{1ST}$	第 1 相邻道干扰			5			dBc		
$C/I_{2ND}$	第 2 相邻道干扰			-12			dBc		
$C/I_{3RD}$	第 3 相邻道干扰			-24			dBc		
单片机特性									
$f_{SYS}$	IHRC/4 IHRC/8 ILRC	0 0		70K	4M 2M	Hz	$V_{DD} \geq 2.5V$ $V_{DD} \geq 2.2V$ $V_{DD} = 3.3V$		
$I_{OP}$	MCU 工作电流			0.3		mA	$f_{SYS}=1MIPS@3.0V$		
				12		uA	$f_{SYS}=ILRC=70kHz$ $@3.0V$		
IPD	单片机掉电模式消耗电流(用 stopsys 命令)			0.5		uA	$f_{SYS}=0Hz, V_{DD}=3.0V$		
IPS	省电模式消耗电流(用 stopexe 命令, 关闭IHRC)			5		uA	$V_{DD}=3.0V$		
$V_{IL}$	输入低电压	0			$0.3V_{DD}$	V	$V_{DD}=3.0V$		
$V_{IH}$	输入高电压	$0.7 V_{DD}$			$V_{DD}$	V	$V_{DD}=3.0V$		
IOL	IO 输出灌电流 (正常输出)								
	*PA0,PA3,PA4, PB5,PB6		10			mA	$V_{DD}=3.0V, V_{OL}=0.33V$		
	*PA6,PA7, PB7		6						
	*PA5		6						
	IOL	IO 输出灌电流 (低输出)							
		*PA5		5			mA	$V_{DD}=3.0V, V_{OL}=0.33V$	
*其它IO			3						
IOH	IO 输出驱动电流 (正常输出)		4			mA	$V_{DD}=3.0V, V_{OH}=2.97V$		
	IO 输出驱动电流 (低输出)		1.6						
$V_{IN}$	输入电压	-0.3		$V_{DD}+0.3$		V			
$I_{INJ}$ (PIN)	脚位的引入电流			1		mA	$V_{DD}+0.3 \geq V_{IN} \geq -0.3$		
RPH	上拉电阻		200			K $\Omega$			
VLVR	低电压侦测电压 *	2.7 2.5 2.3 2.0 1.8 1.6	3 2.75 2.5 2.2 2.0 1.8	3.3 3 2.7 2.4 2.2 2		V			
$f_{IHRC}$	IHRC 输出频率 (校准后) *	15.84 *	16*	16.16*		MHz	@25°C		
		15.20	16*	16.80*			$V_{DD}=2V\sim 5.5V,$ $-20^{\circ}C < Ta < 70^{\circ}C$ *		



		*				
f <sub>ILRC</sub>	ILRC 输出频率*		70*		KHz	V <sub>DD</sub> =3.0V
t <sub>INT</sub>	中断脉冲宽度	30			ns	V <sub>DD</sub> =5.0V
V <sub>DR</sub>	数据存储器数据保存电压*	1.5			V	掉电模式下
t <sub>WDT</sub>	看门狗超时溢出时间		4096		T <sub>IHRC</sub>	misc[1:0]=00 (默认)
			16384			misc[1:0]=01
			65536			misc[1:0]=10
			262144			misc[1:0]=11
t <sub>SBP</sub>	系统上电开机时间 (正常)		1024		T <sub>ILRC</sub> C	T <sub>IHRC</sub> 是IHRC 时钟周期
	系统上电开机时间 (快开机)		45			
t <sub>RST</sub>	外部复位脉冲宽度	120			us	@ V <sub>DD</sub> =5V
CPos	比较器偏压*	-	±10	±20	mV	
CPcm	比较器共模输入电压*	0		VDD-1.5	V	
CPspt	比较器响应时间**		100	500	ns	上升沿和下降沿一样
CPmc	比较器模式改变稳定时间		2.5	7.5	us	
CPcs	比较器电流消耗		20		uA	@3.3V
Top	工作温度	-20	27	+70	°C	

## 4.2 极限值

参数	符号	最小值	最大值	单位
工作温度	Top	-20	70	°C
存储温度	Tstor	-40	125	°C
工作电压	VDD	-0.3	3.7	V
输入射频信号强度	Pin_max		+10	dBm
ESD(人体模型)	ESD_HBM		2	KV

\* 注意：强行超过一项或多项极限值使用会导致器件永久性损坏。

\* 小心：芯片为静电敏感器件，操作时请遵守防护规则。

## 5. 功能描述

### 5.1 存储器结构

存储器包含程序存储器和数据存储器

#### 5.1.1 程序存储器

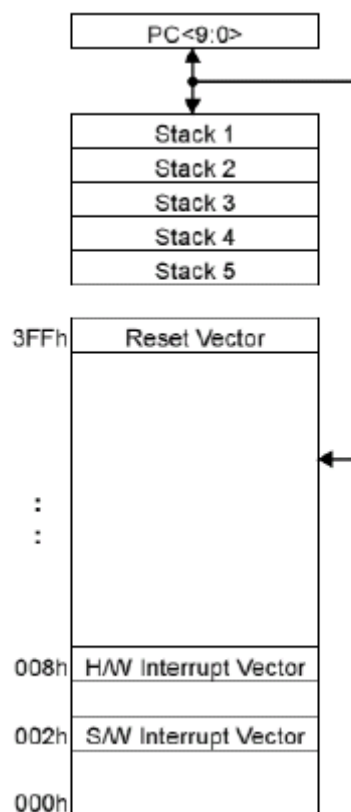
XP9952HP有一个10位PC指针能访问1K×13的存储空间。

XP9952HP的复位地址为3FFh。

H/W中断向量地址008h， S/W中断向量地址002h。

XP9952HP的CALL/GOTO能指向在同一个程序页面（一个程序页面为1K）的所有存储空间。

程序存储器分布图和堆栈结构：



#### 5.1.2 数据存储器

数据存储器包含特殊功能器组和通用寄存器组，所有通用寄存器可以直接寻址或者通过FSR寄存器间接寻址。特殊功能寄存器用来控制CPU或外围功能模块的工作。

表5.1.1寄存器列表

Address	Description
00h	INDF
01h	TMR0
02h	PCL
03h	STATUS
04h	FSR
05h	PORTA
06h	PORTB
07h	General Purpose Register
08h	PCON
09h	WUCON
0Ah	PCHBUF
0Bh	PDCON
0Ch	ODCON
0Dh	PHCON
0Eh	INTEN
0Fh	INTFLAG
10h—3Fh	General Purpose Registers

表5.1.2 通过OPTION或IOST指令控制的寄存器

地址	说明	B7	B6	B5	B4	B3	B2	B1	B0
N/A(W)	OPTION	*	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
05h(W)	IOSTA		Port A I/O 控制器						
06h(W)	IOSTB		Port B I/O 控制器						

表5.1.3 寄存器列表

地址	说明	B7	B6	B5	B4	B3	B2	B1	B0
00h(r/w)	INDF	通过FSR访问数据区（不是一个实际的物理地址）							
01h(r/w)	TMR0	8位定时器/计数器							
02h(r/w)	PCL	低8位PC指针							
03h(r/w)	STATUS	RST	GP1	GP0	/T0	/PD	Z	DC	C
04h(r/w)	FSR	*	*	间接地址访问指针(RAM选择寄存器)					
05h(r/w)	PORTA					IOA3	IOA2	IOA1	IOA0
06h(r/w)	PORTB	IOB7	IOB6	IOB5	IOB4	IOB3	IOB2	IOB1	IOB0
07h(r/w)	SRAM	通用寄存器							
08h(r/w)	PCON	WDTE	EIS	LVDTE	*	*	*	*	*
09h(r/w)	WUCON	WUB7	WUB6	WUB5	WUB4	WUB3	WUB2	WUB1	WUB0
0Ah(r/w)	PCHBUF							2MSB Buffer of PC	

0Bh(r/w)	PDCON		/PDB2	/PDB1	/PDB0	/PDA3	/PDA2	/PDA1	/PDA0
0Ch(r/w)	ODCON	ODB7	ODB6	ODB5	ODB4		ODB2	ODB1	ODB0
0Dh(r/w)	PHCON	/PUB7	/PUB6	/PUB5	/PUB4		/PUB2	/PUB1	/PUB0
0Eh(r/w)	INTEN	GIE	*	*	*	*	INTIE	PBIE	T0IE
0Fh(r/w)	INTFLAG	-	-	-	-	-	INTIF	PBIF	T0IF

Legend: - = unimplemented, read as '0', \* = unimplemented, read as '1'

## 5.2 寄存器操作

### 5.2.1 INDF (间接寻址寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
00h (r/w)	INDF	通过 FSR 访问数据区(不是一个实际的物理地址)							

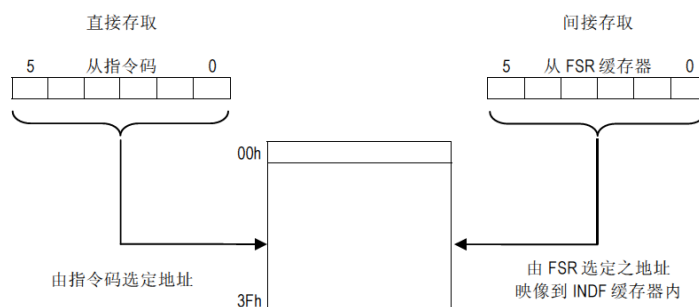
INDF 不是一个实际的物理地址，间接寻址时 INDF 通过 RAM 选择寄存器（FSR）来访问其所指向的地址。间接寻址读操作直接读地址 00h(FSR="0")，间接寻址不能对 INDF 直接进行写操作（尽管有些状态会发生改变）。

FSR 的 5-0 位可以用来选择 64 个寄存器（地址：00h ~ 3Fh）。

#### 例 5.2.1:间接寻址

- 地址38内容为10h
- 地址39内容为0Ah
- 将38写入FSR 中
- 通过A读INDF返回10h
- FSR加1 (@FSR=39h)
- 通过A读INDF返回0A h

图5.2.1：直接/间接存取



### 5.2.2 TMR0 (定时/计数器 Time lock/Counter register)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
00h (r/w)	TMR0	8 位定时/计数器							

TMR0是一个8位定时/计数器寄存器,Timer0的时钟源可以取值于指令周期或外部实时钟（T0CKI pin），使用外部时钟需要设置OPTION的T0CS(T0CS=5)位为1。

使用TMR0的预置器需要设置OPTION的PSA (PSA =3)位为0，这种模式下TMR0值的改变，预置器被清零。

### 5.2.3 PCL (Low Bytes of Program Counter) & Stack

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
02h (r/w)	PCL	8 位定时/计数器							

XP9952HP的PC指针和堆栈的位数为10位，堆栈有5级，低位的PC指针为PCL寄存器,该寄存器时可读写的，高位的PC指针为PCH寄存器，该寄存器包含PC<9:8> 位，该寄存器不能直接读写。PCH寄存器的改变是通过PCHBUF寄存器来实现的。每一条指令执行的时候他的PC指针包含下一条指令的操作地址。指令没有改变PC内容时候、在每一个指令周期PC指针自动加1。

对于GOTO指令有PC<9:0>，PCL 映射成PC<7:0>，PCHBUF不变。

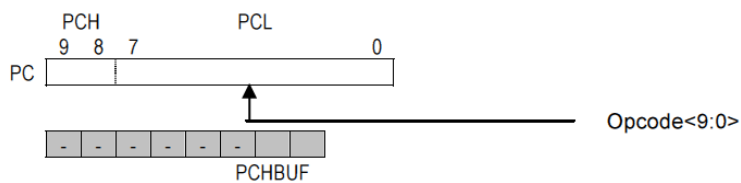
对于CALL指令有PC<9:0>，下一条指令地址被推进堆栈，PCL 映射成PC<7:0>，PCHBUF不变。

对于RETIA, RETFIE, RETURN指令有PC<9:0>，PC的内容更改为出栈信息，PCL 映射成PC<7:0>，PCHBUF不变。

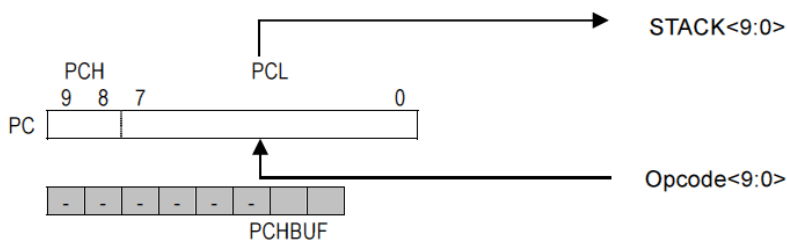
对于其他指令,PCLj就是目标信息, PC<7:0>的内容就是指令地址或。不管怎样, PC<9:8> 来源于 PCHBUF<1:0> 位 (PCHBUF→ PCH)。PCHBUF不会改变，从而PCH不会改变。

图5.2.2: 不同的指令调用PC指针跳转方式

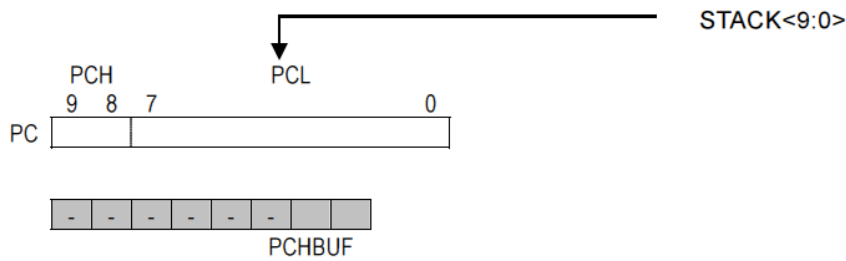
#### 1、GOTO 指令



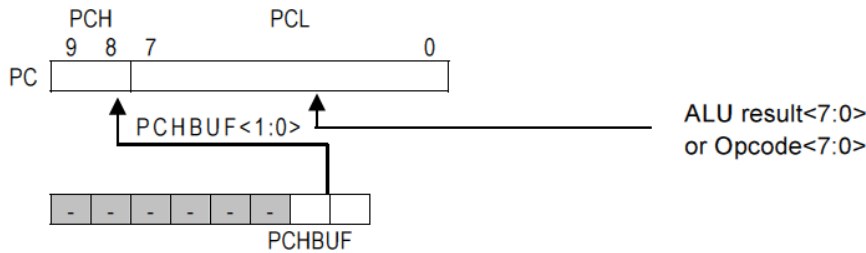
#### 2、CALL 指令



#### 3、RETIA, RETFIE, RETURN指令



4、以 PCL 为目的的指令



注释1. PCHBUF 只有在 PCL 内容是目标地址才有效，当 PCL 是运算结果时候，PCHBUF 不起作用。

5.2.4 STATUS (状态字寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
03h (r/w)	STATUS	RST	GP1	GP0	TO	PD	Z	DC	C

状态字寄存器包含运算标志，结果标志。

指令执行以后可能会影响 STATUS 寄存器的 Z、DC、C 标志位，则不能直接对这三个标志位进行写操作，这些标志位的设置由 MCU 的逻辑自动完成。同时，TO 和 PD 位也是不能通过指令直接改变写操作。因此，与 STATUS 作为目标寄存器的指令后，结果可能会与预期的不同。例如：运行 CLRR STATUS 将把 STATUS 的高三位置零和 Z 标志位置 1 同时该寄存器的内容如下：

0	0	0	u	u	1	u	u
---	---	---	---	---	---	---	---

u 表示为指令执行前后该位没有发生改变

C:进位标志

ADDAR, ADDIA

= 1, 有进位

= 0, 无进位

SUBAR, SUBIA

= 1, 无借位

= 0, 有借位

注释：减法是通过将 2 的补第二个操作数的执行。旋转 (RRR, RLR) 指令，该位装载高或低位源寄存器位。

DC:辅助进位/借位标志.(低四位向高四位进位/借位标志)

ADDAR, ADDIA

= 1, 底 4 位有进位

= 0, 底 4 位无进位

SUBAR, SUBIA

= 1, 底 4 位无借位

= 0, 底 4 位有借位

Z: Zero bit.

= 1, 算术或逻辑运算结果为“0”时

= 0, 算术或逻辑运算结果不为“0”时

PD :Power down flag bit.

- = 1, 当系统上电时或执行“CLRWDT”指令后
- = 0, 当执行“SLEEP”指令后

TO :Time overflow flag bit.

- = 1, 当系统上电时或执行“CLRWDT”或 SLEEP 指令后
- = 0,看门狗定时器溢出

GP1:GP0 :通用寄存器读/写位

RST :定义系统复位类型位.

- = 1, 唤醒 SLEEP 或 Port B 脚位变化唤醒 SLEEP
- = 0, 其他类型唤醒 SLEEP.

### 5.2.5 FSR (间接寻址指针)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
04h (r/w)	FSR	*	*	间接寻址指针					

**Bit5:Bit0** : 用来选择访问间接寻址时目标寄存器地址. 具体描述见 2.1.1。

**Bit7:Bit6** : 没有使用。

### 5.2.6 PORTA, PORTB (Port 寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
05h (r/w)	PORTA					IOA3	IOA2	IOA1	IOA0
06h (r/w)	PORTB	IOB7	IOB6	IOB5	IOB4	IOB3	IOB2	IOB1	IOB0

读端口(PORTA, PORTB 寄存器)的状态依赖于该端口是输入/输出模式，写端口是向锁存器写数据。

PORTA 是一个4位端口数据寄存器，只有低4位被使用 (PORTA<3:0>). Bits 7-4通常作为读/写位。

PORTB是一个8位端口数据寄存器. IOB3只能作为输入。

### 5.2.7 PCON (电源控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
08h (r/w)	PCON	WDTE	EIS	LVDTE	*	*	*	*	*

**Bit4:Bit0** : Not used. 置“1”。

**LVDTE** : LVDT (低电压检测) 使能位。

- = 0, 关闭 LVDT。
- = 1, 使能 LVDT。

**EIS** : 定义管脚 B0/INT功能位

- = 0, IOB0 (双向 I/O 口) is selected. 屏蔽了 INT 功能。
- = 1, INT (外部中断输入脚)，在这种模式下，PORTB 的 IOB0 必须置“1”。IOB0 作为 I/O 口输入功能通过硬件屏蔽了，读取 INT 管脚信息的与读 PORTB.方式相同。

**WDTE** : WDT (watch-dog timer) 使能看门狗定时器

- = 0,关闭 WDT。

= 1,使能WDT。

## 5.2.8 WUCON (Port B 输入改变/唤醒控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
09h (r/w)	WUCON	WUB7	WUB6	WUB5	WUB4	WUB3	WUB2	WUB1	WUB0

**WUB0** := 0,禁止 IIOB0 输入改变/唤醒功能  
=1,使能 IOB0 输入改变/唤醒功能

**WUB1** := 0,禁止 IIOB1 输入改变/唤醒功能  
=1,使能 IOB1 输入改变/唤醒功能

**WUB2** := 0,禁止 IIOB2 输入改变/唤醒功能  
=1,使能 IOB2 输入改变/唤醒功能

**WUB3** := 0,禁止 IIOB3 输入改变/唤醒功能  
=1,使能 IOB3 输入改变/唤醒功能

**WUB4** := 0,禁止 IIOB4 输入改变/唤醒功能  
=1,使能 IOB4 输入改变/唤醒功能

**WUB5** := 0,禁止 IIOB5 输入改变/唤醒功能  
=1,使能 IOB5 输入改变/唤醒功能

**WUB6** := 0,禁止 IIOB6 输入改变/唤醒功能  
=1,使能 IOB6 输入改变/唤醒功能

**WUB7** := 0,禁止 IIOB7 输入改变/唤醒功能  
=1,使能 IOB7 输入改变/唤醒功能

## 5.2.9 PCHBUF (PC 指针高位缓冲区)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0Ah (r/w)	PDHBUF							电脑 2MSBs 缓冲	

**Bit1:Bit0** : 见2.1.3

**Bit7:Bit2** : 没有使用, 置“0”。

## 5.2.10 PDCON (I/O 下拉控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0Bh (r/w)	PDCON		/PDB2	/PDB1	/PDB0	/PDA3	/PDA2	/PDA1	/PDA0

**/PDA0** := 0,使能 IOA0 内部下拉  
= 1,禁止 IOA0 内部下拉

**/PDA1** := 0,使能 IOA1 内部下拉  
= 1,禁止 IOA1 内部下拉

**/PDA2** := 0,使能 IOA2 内部下拉  
= 1,禁止 IOA2 内部下拉

**/PDA3** := 0,使能 IOA3 内部下拉  
= 1,禁止 IOA3 内部下拉

**/PDB0** := 0,使能 IOB0 内部下拉  
= 1,禁止 IOB0 内部下拉

**/PDB1** := 0,使能 IOB1 内部下拉  
= 1,禁止 IOB1 内部下拉



**/PDB2**: = 0,使能 IOB2 内部下拉  
 = 1,禁止 IOB2 内部下拉

**Bit7** : 一般的读/写位

### 5.2.11 ODCON (I/O 开路控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0Ch (r/w)	ODCON	ODB7	ODB6	ODB5	ODB4		ODB2	ODB1	ODB0

**ODB0** : = 0,禁止 IOB0 内部开路  
 = 1,使能 IOB0 内部开路

**ODB1** : = 0,禁止 IOB1 内部开路  
 = 1,使能 IOB1 内部开路

**ODB2** : = 0,禁止 IOB2 内部开路  
 = 1,使能 IOB2 内部开路

**Bit3** : 一般的读/写位

**ODB4** : = 0,禁止 IOB4 内部开路  
 = 1,使能 IOB4 内部开路

**ODB5** : = 0,禁止 IOB5 内部开路  
 = 1,使能 IOB5 内部开路

**ODB6** : = 0,禁止 IOB6 内部开路  
 = 1,使能 IOB6 内部开路

**ODB7** : = 0,禁止 IOB7 内部开路  
 = 1,使能 IOB7 内部开路

### 5.2.12 PHCON (I/O 上拉控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0Dh (r/w)	PHCON	PHB7	PHB6	PHB5	PHB4		PHB2	PHB1	PHB0

**/PHB0** : = 0,使能 IOB0 内部上拉  
 = 1,禁止 IOB0 内部上拉

**/PHB1** : = 0,使能 IOB1 内部上拉  
 = 1,禁止 IOB1 内部上拉

**/PHB2** : = 0,使能 IOB2 内部上拉  
 = 1,禁止 IOB2 内部上拉

**Bit3** : 一般的读/写位

**/PHB4** : = 0,使能 IOB4 内部上拉  
 = 1,禁止 IOB4 内部上拉

**/PHB5** : = 0,使能 IOB5 内部上拉  
 = 1,禁止 IOB5 内部上拉

**/PHB6** : = 0,使能 IOB6 内部上拉  
 = 1,禁止 IOB6 内部上拉

**/PHB7** : = 0,使能 IOB7 内部上拉  
 = 1,禁止 IOB7 内部上拉

### 5.2.13 INTEN (中断屏蔽寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0Eh (r/w)	INTEN	GIE	*	*	*	*	INTIE	PBIE	TOIE

**TOIE** : Timer0溢出中断屏蔽位。  
= 0,禁止Timer0溢出中断  
= 1,使能 Timer0 溢出中断

**PBIE** : Port B输入改变中断屏蔽位  
= 0,禁止Port B输入改变中  
= 1,使能 Port B 输入改变中

**INTIE** : 外部中断屏蔽位  
= 0,禁止外部中断.  
= 1,使能外部中断

**Bit6:BIT3** :没有使用。置“1”

**GIE** : 中断允许总控位  
= 0,禁止所有中断. 对于睡眠唤醒模式的中断事件, MCU 将执行 SLEEP 后的指令。  
= 1,使能所有没有屏蔽的中断. 对于睡眠唤醒模式的中断事件, MCU 将跳转到中断地址 (008h)。

注释 :在中断事件发生时, GIEB被硬件清零并禁止一切中断, 所以GIE以及与该中断相关的中断屏蔽位需要重开启。RETFIE 为退出中断程序并重新设置GIE =1允许中断。

### 5.2.14 INTFLAG (中断标志寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0Fh (r/w)	INTFLAG	-	-	-	-	-	INTIF	PBIF	TOIF

**TOIF** : 溢出中断标志, 发生 Timer0 溢出中断置 1, 软件设置清零

**PBIF** : Port B 输入改变中断标志 interrupt flag. Port B 输入改变时置 1, 软件设置清零

**INTIF** : 外部中断标志. 当管脚 INT 上升沿/下降沿 (是上升沿/下降沿由 INTEDG 位 (OPTION<6>)决定) 时置 1, 软件设置清零

**Bit7:BIT3** : 没有使用, 置 0

### 5.2.15 ACC (Accumulator)累加器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
N/A (r/w)	ACC	累加器							

累加器是一个内部数据转化、指令操作和存放操作结果的存储单元, 不能被访问。

### 5.2.16 OPTION Register (选项寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
N/A (w)	OPTION	*	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

通过OPTION 指令访问

在执行OPTION 指令时候,该数据单元由ACC (累加器) 转化为选项寄存器 (OPTION Register)。

选项寄存器是一个7位只写寄存器, 它的一些控制位主要用来配置与Timer0/WDT 分频器, Timer0, 外部中断选项相关

信息。

除INTEDG位以外其他位是只写并可以置1。

**PS2:PS0** : 分频率选择控制位

PS2:PS0	Timer0 Rate	WDT Rate
0 0 0	1:2	1:1
0 0 1	1:4	1:2
0 1 0	1:8	1:4
0 1 1	1:16	1:8
1 0 0	1:32	1:16
1 0 1	1:64	1:32
1 1 0	1:128	1:64
1 1 1	1:256	1:128

**PSA** : 分频器选择位.

= 1, WDT (看门狗定时器)

= 0, TMR0 (Timer0)

**T0SE** : TMR0触发方式控制位

= 1, T0CKI 脚下降沿触发计数

= 0, T0CKI 脚上升沿触发计数

**T0CS** : TMR0 时钟源选择控制位

= 1, 外部 T0CKI 脚. 当 IOST IOB2 = "0".时, IOB2/T0CKI 脚设置为输入

= 0, internal instruction clock cycle

**INTEDG** : 中断触发方式控制位.

= 1, 中断触发方式为 INT 脚上升沿出发

= 0, 中断触发方式为 INT 脚下降沿出发

**Bit7** : 没有使用

### 5.2.17 IOSTA & IOSTB (I/O 口控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
N/A (w)	IOSTA					IOSTA3	IOSTA2	IOSTA1	IOSTA0
N/A (w)	IOSTB	IOSTB7	IOSTB6	IOSTB5	IOSTB4	IOSTB3	IOSTB2	IOSTB1	IOSTB0

通过IOST指令访问

通过指令 OST R (05h~06h)把累加器 A 的内容加载到 I/O 控制寄存器, 按位将 IOSTA, IOSTB 设为 1 表示该脚为输入 (高阻抗)、设为 "0" 时表示该脚为输出。

IOST 寄存器只写, 系统复位以后设置为输入 (高阻抗)。

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0Bh (r/w)	PDCON		/PDB2	/PDB1	/PDB0	/PDA3	/PDA2	/PDA1	/PDA0

**/PDA0** : = 0,使能 IOA0 内部下拉

= 1,禁止 IOA0 内部下拉.

**/PDA1** : = 0,使能 IOA1 内部下拉

= 1,禁止 IOA1 内部下拉

**/PDA2** : = 0,使能 IOA2 内部下拉

= 1,禁止 IOA2 内部下拉

**/PDA3** : = 0,使能 IOA3 内部下拉

= 1,禁止 IOA3 内部下拉

**/PDB0:** = 0,使能 IOB 0 内部下拉  
= 1,禁止 IOB0 内部下拉

**/PDB1:** = 0,使能 IOB1 内部下拉  
= 1,禁止 IOB1 内部下拉

**/PDB2:** = 0,使能 IOB2 内部下拉  
= 1,禁止 IOB2 内部下拉

**Bit7 :** 一般的读/写位

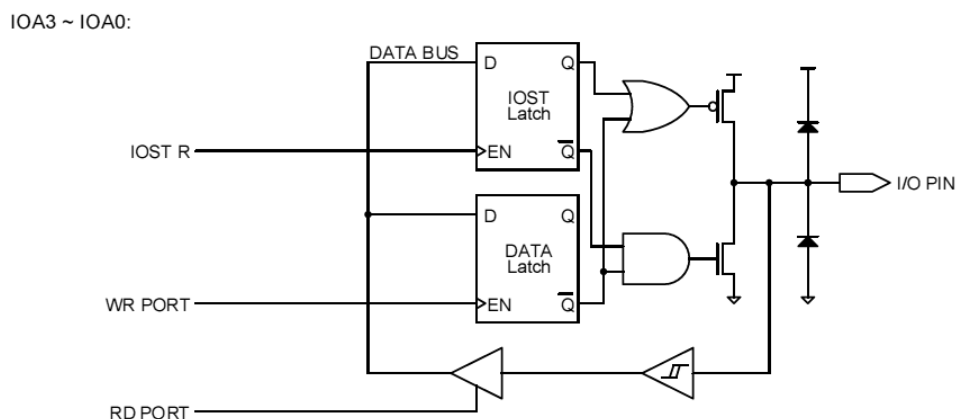
## 5.3 I/O Ports

Port A 和 port B为双向三态I/O 口. Port A为4脚I/O口. Port B 为8脚I/O口. 注意IOB3只能作为输入口。除了 IOB3 只作为输入和IOB2需要通过选项寄存器（Option）的T0CS ((OPTION<5>))位控制，所有的I/O的输入/输出方式.有I/O控制寄存器(IOSTA, IOSTB)设置。 IOB<7:4> 和 IOB<2:0>有相应的上拉控制位(PHCON 寄存器)来设置使能内部上拉，如果设置为输出模式，内部上拉功能会自动关闭。IOA<3:0>和IOB<2:0>有相应的下拉控制位(PDCON寄存器)来设置使能内部下拉，如果设置为输出模式，内部下拉功能会自动关闭。IOB<7:4>和IOB<2:0>有相应的开路控制位(ODCON寄存器)来设置使能开路来设置输出为开路输出。

IOB<7:0> 有输入改变中断/唤醒功能.它的每个管脚是否具有该功能通过取决于WUCON寄存器的相应位。当EIS(PCON<6>)=1 时， IOB0作为外部中断输入脚，在该模式下IOB0 输入改变中断/唤醒功能i被硬件屏蔽，即使软件已经设置为中断/唤醒功能可用也不可启用该功能。

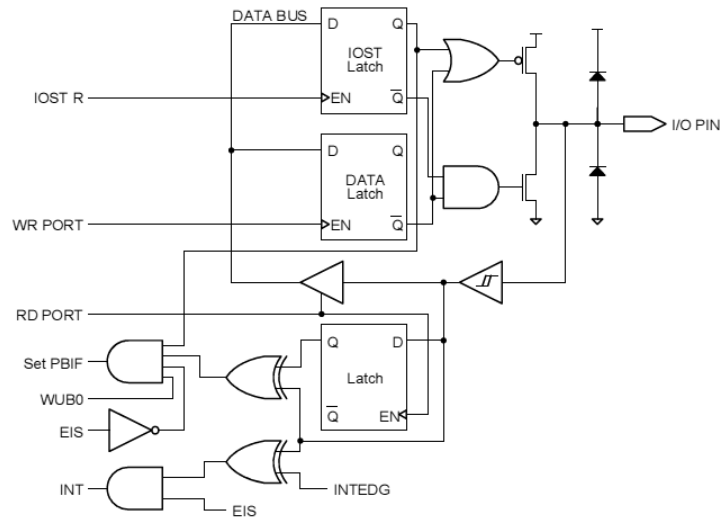
配置能交替设置I/O口的不同功能，功能交替设置完以后，读的I/O的值为0。

图 5.3: I/O 脚的方块图



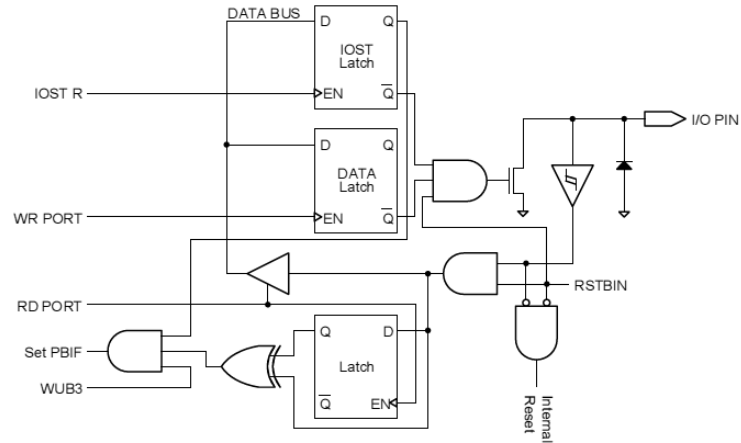
下拉在图中未显示

IOB0/INT:



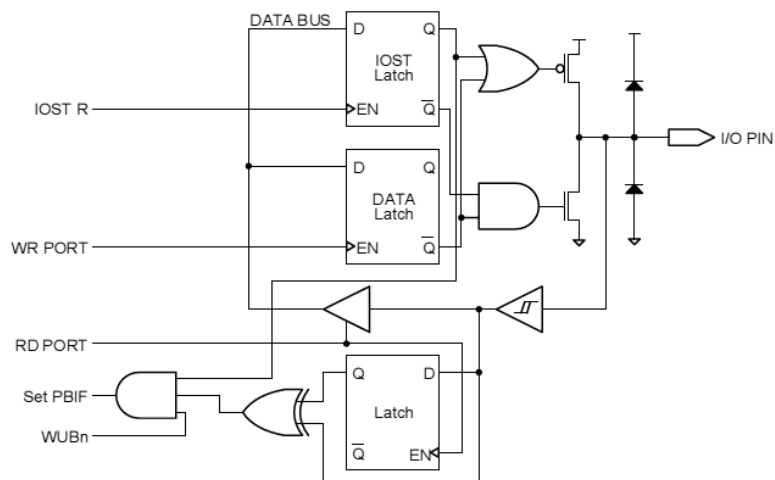
上拉/下拉和漏极开路在图中未显示

IOB3:



电压在这个引脚禁止超过 VDD

IOB7 ~ IOB4, IOB2 ~ IOB1:



上拉/下拉和漏极开路在图中未显示

## 5.4 Timer0/WDT & Prescaler 2.3.1 Timer0

Timer0 为 8 位定时/计数器，Timer0 的时钟源可以是内部或外部时钟源(TOCLKI pin)

### 5.4.1 使用内部时钟：定时模式

T0CS(OPTION<5>)=0 为定时模式，定时模式在没有预置器的情况下，定时寄存器每个指令周期自动加 1,设置 TMR0 以后，定时器将在两个时钟周期以后开始自增。

### 5.4.2 使用外部时钟：计数模式

T0CS(OPTION<5>)=1 为计数模式，是选择通过 TOCK 管脚的上升或下降沿触发 Timer0 寄存器的增加由 T0SE 位 (OPTION<4>)决定，

外在时钟要求与内部时钟(Tosc)同步。同步以后，Timer0 实际增加有一个延迟。

在没有预置器的情况下，外部时钟输入同样也可以作为预置器输出；TOCKI 与内部时钟同步时能方便处理在 T2 和 T4 周期上的预分频。因此 TOCKI 为高或低电平必须要保持两个以上时钟周期才有效。

有预置分频时器，外部时钟输入被异步分频器平分，这种常用来计算波形。因此：因此TOCKI的一个波形周期至少 4Tosc才能被 预置器平分。

### 5.4.3 看门狗定时器 (WDT)

看门狗定时器 (WDT) 的运行依赖于芯片里的 RC 振荡器，无需任何额外电路即能工作。不管时钟 OSCI 和 OSCO 管脚是否关闭，它都能运行，如在睡眠模式。在一般操作或睡眠模式情况下，看门狗定时器的溢出都会导致 MCU 复位同时 TO (STATUS<4>)位被清零。

如WDTE 位(PCON<7>)清零。看门狗定时器不能工作。

在没有预置器时看门狗的溢出为18 ms, 4.5ms, 288ms ， 72ms这个时间可以通过SUT<1:0> 设置。

需要看门狗的t溢出周期变长可以通过设置OPTION寄存器的看门狗定时器分频大于1:128.,因此最长的看门狗的t溢出周期为 36.8 秒。

CLRWDWT指令能使WDT和预置器清零,启用看门狗可以防止超时，如果超时MCU能复位。

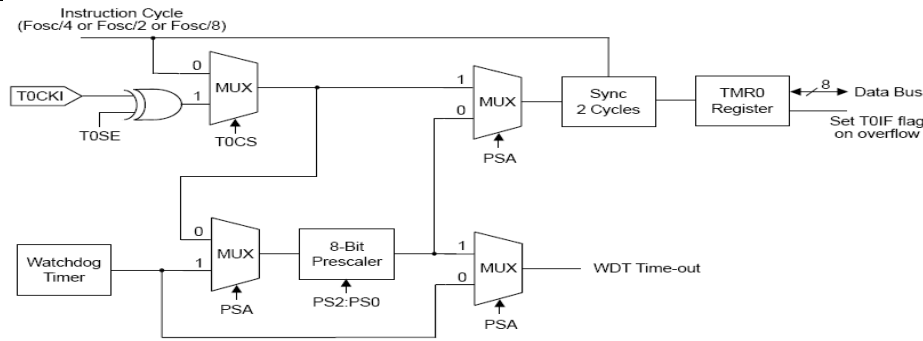
SLEEP 指令重置 WDT 和预置器，启用看门狗就给机器分派了一个最大睡眠时间。

### 5.4.4 Prescaler (预置器)

有一个8位的向下计数器作为Timer0和看门狗定时器(WDT)的预置器。注意该预置器只能分配给Timer0 或 WDT 使用，不能两者同时使用。PSA 位(OPTION<3>) 决定预置器是指派给Timer0还是WDT. PS<2:0> 位 (OPTION<2:0>) 配置分频。当作为Timer0的预置器的时候， TMR0会被预置器清零。当作为WDT的预置器的时候， CLRWD 指令会清除预置器内容。预置器不能读写。机器复位，预置器各位全为1。

为了避免机器非正常复位，当Timer0 或 WDT的预置器发生改变的时候，需要执行CLRWDWT 或 CLRR TMR0 指令，反之亦然。

图5.4: Block Diagram of The Timer0/WDT Prescaler



## 5.5 中断方式

XP9952HP系统具备有三种中断方式:

1. INT 管脚的外部中断
2. TMR0 溢出中断
3. Port B 输入改变中断 (IOB7:IOB0脚)

INTFLAG为中断标志寄存器，决定该寄存器机器所发生的中断状态。

中断允许总控位 GIE (INTEN<7>), 能使所有中断被开放 (GIE=1) 或屏蔽所有中断(GIE=0), 每中断能否启用决定 INTEN 寄存器同时保证 GIE=1。

中断发生时 GIE 位 (在中断发生前 GIE 位和该中断相关的中断屏蔽位置 1) 被硬件清零从而禁止进一步中断 (XP9952HP 不区分中断优先级),同时下条指令跳到 008h 后开始执行。中断标志位在中断允许总控位 GIE 重新置 1 的时候需要被软件清零以防止重复中断。一个中断标志位 (PBIF 除外的) 会被它的中断事件置 1, 而不管与它相关的中断屏蔽位是否启用。通过 INTFLAG 和 INTEN 的相应中位来判断是否发生中断以及中断类型。当通过 INT 指令发生软中断时, 下条指令跳到 002 后开始执行。

### 5.5.1 外部中断

外部中断INT管脚上升沿还是下降沿触发由 INTEDG 位 (OPTION<6>)决定, 当一个有效的跳变发生时标志位 INTIF置1,如INTIE位(INTEN<2>)清零, 该中断被屏蔽。

在睡眠之前INTIE 位已被置1, INT管脚可以作为系统睡眠条件。在睡眠之前GIE位已被置1机器唤醒以后会执行中断服务程序, 否则会运行睡眠以后的下一条指令。

### 5.5.2 Timer0 中断

TMR0 发生溢出 (FFh → 00h)时 T0IF 标志位置 1 (INTFLAG<0>). T0IE 位(INTEN<0>)清零, 该中断被屏蔽。

### 5.5.3 Port B 输入改变中断

输入改变中断触发时IOB<7:0> PBIF标志位置1 (INTFLAG<1>). PBIE位(INTEN<1>)清零, 该中断被屏蔽。

在输入改变中断发生之前, 必须读取port B信息

与PortB的管脚相对应的WUBn位 (WUCON<7:0>) 清零或设置为输出或IOB0 脚设置为外部中断输入脚INT 都拥有该功能。PBIE在睡眠之前置1, port B 输入脚改变中断也可以作为睡眠唤醒条件。在睡眠之前GIE位已被置1机器

唤醒以后会执行中断服务程序，否则会运行睡眠以后的下一条指令。

## 5.6 省电模式 (SLEEP)

执行SLEEP 指令以后机器进入省电模式。

执行SLEEP 指令, PD 位清零 (STATUS<3>) , TO位置1,看门狗清零同时保持运行状态, 晶体停振。

I/O 维持原状

在睡眠状态下, 单片机能通过以下方式唤醒:

1. RSTB 管脚复位
2. 看门狗复位 (机器设置了看门狗).
3. RB0/INT管脚中断,或PORTB 输入改变中断.

外部的RSTB管脚和看门狗通过设置PD 和TO 位都能使机器复位. PD 和TO 位 , PD位置1用于上电复位, 清零用于SLEEP复位, TO 位清零用于看门狗溢出复位。 .

机器通过中断唤醒, 该中断屏蔽位置1, 中断唤醒不管GIE是否置1.。 当GIE位被清零, 机器唤醒以后执行SLEEP指令以后的指令; 当GIE位被置1, 机器唤醒以后跳转到中断复位地址 (008h)。在高频或低频模式机器复位延迟时间为18/4.5/288/72ms (该延迟时间由SUT<1:0>设置) 加上16个振荡周期。

在 IRC/ERIC or ERC 模式, 机器复位延迟时间为 140us。

## 5.7 复位

XP9952HP 单片机能通过以下方式复位:

1. 上电复位(POR)
2. 掉电复位(Brown-out Reset BOR)
3. RSTB 管脚复位
4. 看门狗WDT溢出复位

一些寄存器在一些复位条件下没有影响, 在上电和其他一些复位情况下它们的状态是未知的。 . 大多数寄存器会回到复位状态在上电复位, RSTB 管脚复位, 看门狗WDT溢出复位。

对Vdd上升信号检测告之芯片是否加上上电复位脉冲信号。 要使用这个特点, 用户需要把RSTB管脚连接到Vdd。

掉电复位作为一种典型应用主要用在 AC 或重载交换的应用上。

芯片上的低电压检测模块到电压低于一个固定的电压也会对使芯片复位, 这样能保证芯片只能在正常电压范围内工作。 .

RSTB或WDT睡眠唤醒也导致芯片复位, 其复位操作的不会在睡眠之前。

根据不同的复原状态设置对TO和PD位(STATUS<4 : 3>)置1或清零。



### 5.7.1 上电复位计数器 (Power-up Reset Timer PWRT)

上电复位计数器提供一个 18/4.5/288/72ms 延迟时间 (该延迟时间由 SUT<1:0>设置) (或 140us,基于不同的振荡源和复位条件) 在 Power-on Reset (POR), Brown-out Reset (BOR), RSTB Reset 或 看门狗溢出复位。只要 PWRT 在运行, 设备就一直保持的复位状态

Vdd、温度和其他变化而会影响 PWDT 控制的设备延迟时间。

表5.7: PWRT Period

Oscillator Mode	Power-on Reset Brown-out Reset	RSTB Reset WDT time-out Reset
ERC & IRC/ERIC	18/4.5/288/72 ms	140 us
HF & LF	18/4.5/288/72 ms	18/4.5/288/72ms

### 5.7.2 振荡启动计数器(Oscillator Start-up Timer OST)

在HF或 LF振荡模式下在PWRT 延迟 (18/4.5/288/72ms) 之后振荡启动计数器会再提供一个16个clock的延迟。这种延迟晶体谐振器能提供稳定的振荡源, 这段时间内只要OST在工作, 设备就一直保持的复位状态。

在 OSCI 信号的振幅到达振荡器输入最大振幅之后, 该计数器只开始增加。

### 5.7.3 复位顺序

XP9952HP复位时序如下:

1. 复位锁存器置1, PWRT & OST 清零。
2. 当内部的 POR, BOR, RSTB 复位或 WDT 溢出复位脉冲加载完成后, PWRT 开始计数。
3. PWRT溢出以后, OST开始计数延迟。
4. OST延迟完成以后, 复位锁存器清零最后芯片得到一个复位信号。

在高频或低频振荡模式机器复位延迟时间为18/4.5/288/72ms加上16个振荡周期, 在IRC/ERIC, ERC振荡模式单片机会在Power-on Reset (POR), Brown-out Reset (BOR), 或RSTB复位以后在延迟140us, 看门狗溢出复位后再延迟18/4.5/288/72ms的时间。

图5.7.3: 复位电路结构图

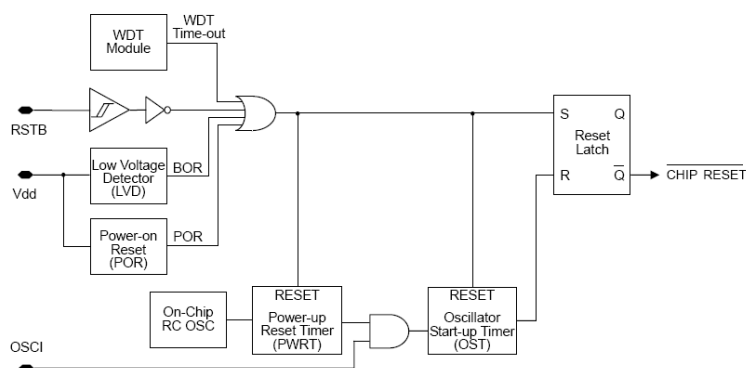


表 5.7.3: 复位以后各个寄存器状态列表

寄存器	地址	上电复位 掉电复位	RSTB 复位 WDT 复位
ACC	N/A	xxxx xxxx	uuuu uuuu
OPTION	N/A	-011 1111	-011 1111
IOSTA	N/A	---- 1111	---- 1111
IOSTB	N/A	1111 1111	1111 1111
INDF	00h	xxxx xxxx	uuuu uuuu
TMR0	01h	xxxx xxxx	uuuu uuuu
PCL	02h	1111 1111	1111 1111
STATUS	03h	0001 1xxx	000# #uuu
FSR	04h	11xx xxxx	11uu uuuu
PORTA	05h	xxxx xxxx	uuuu uuuu
PORTB	06h	xxxx xxxx	uuuu uuuu
General Purpose Register	07h	xxxx xxxx	uuuu uuuu
PCON	08h	101- ----	101- ----
WJCON	09h	0000 0000	0000 0000
PCHBUF	0Ah	-----00	---- --00
PDCON	0Bh	1111 1111	1111 1111
ODCON	0Ch	0000 0000	0000 0000
PHCON	0Dh	1111 1111	1111 1111
INTEN	0Eh	0--- -000	0--- -000
INTFLAG	0Fh	---- -000	---- -000
General Purpose Registers	10 ~ 3Fh	xxxx xxxx	uuuu uuuu

Legend: u = 不变, x = 未知, - = 不起作用, # = 参见下表的值

表5.7.4: RST/TO/PD 复位和唤醒后的状态

RST	/TO	/PD	复位方式
0	1	1	Power-on Reset
0	1	1	Brown-out reset
0	u	u	RSTB Reset during normal operation
0	1	0	RSTB Reset during SLEEP
0	0	1	WDT Reset during normal operation
0	0	0	WDT Wake-up during SLEEP
1	1	0	Wake-up on pin change during SLEEP

Legend: u = 不变

表: 5.7.5: TO/PD状态位影响事件

事件	TO	PD
Power-on	1	1
WDT Time-Out	0	u
SLEEP instruction	1	0
CLRWDT instruction	1	1

Legend: u = 不变

## 5.8 十六进制转化为十进制(Hexadecimal Convert to Decimal,HCD)

XP9952HP具有十进制格式化功能。当一个寄存器里面的内容需要十进制转化的时候,在执行操作ALU以后必须把结果进行相应的进制转化。一个数据在处理过程中进行了转化成了十进制,那么所有对这个数进行的操作(包含存放该数据的RAM单元, accumulator (ACC),立即数,以及所要查表信息)都的进行十进制转化,这样的运算结果才正确。

DAA指令能在加法运算完成以后将ACC 里的数据从十六进制转化为十进制重存给ACC

转换操作在例子 2.2 中被说明。

### 例 5.8.1: DAA 转化

Address	Code
NA	#include <8PB53B.ASH>
n	...
n+1	MOVIA 0x90 ;Set immediate data = decimal format number "90" (ACC ← 90h)
n+2	MOVAR 0x30 ;Load immediate data "90" to data memory address 30H
n+3	MOVIA 0x10 ;Set immediate data = decimal format number "10" (ACC ← 10h)
n+4	ADDAR 0x30,A ;Contents of the data memory address 30H and ACC are binary-added ;the result loads to the ACC (ACC ← A0h, C ← 0)
n+5	DAA ;Convert the content of ACC to decimal format, and restored to ACC ;The result in the ACC is "00" and the carry bit C is "1". This represents the ;decimal number "100"
n+6	...

DAS指令能在减法运算完成以后将ACC 里的数据从十六进制转化为十进制重存给ACC  
转换操作在例子2.3中被说明

### 例 5.8.2: DAS 转化

Address	Code
NA	#include <8PB53B.ASH>
n	...
n+1	MOVIA 0x10 ;Set immediate data = decimal format number "10" (ACC ← 10h)
n+2	MOVAR 0x30 ;Load immediate data "90" to data memory address 30H
n+3	MOVIA 0x20 ;Set immediate data = decimal format number "20" (ACC ← 20h)
n+4	SUBAR 0x30,A ;Contents of the data memory address 30H and ACC are binary-subtracted ;the result loads to the ACC (ACC ← F0h, C ← 0)
n+5	DAS ;Convert the content of ACC to decimal format, and restored to ACC ;The result in the ACC is "90" and the carry bit C is "0". This represents the ;decimal number "-10"
n+6	...

## 5.9 振荡器配置 (Oscillator Configurations)

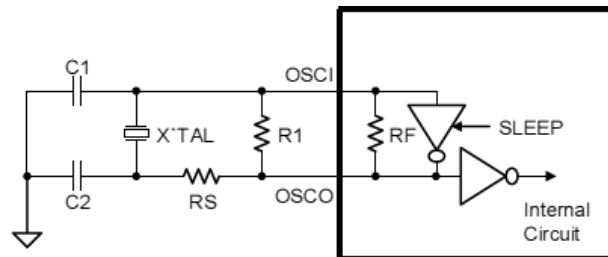
XP9952HP有六种不同的振荡模式，用户可通过编程Fosc配置位来选择相应的振荡方式:

- LF: 低频晶体器
- HF: 高频晶体/谐振器
- IRC:内部电阻内部电容振荡器
- ERIC:外部电阻内部电容振荡器
- ERC: 外部RC振荡器
- XT: 晶体/陶瓷振荡器
- 

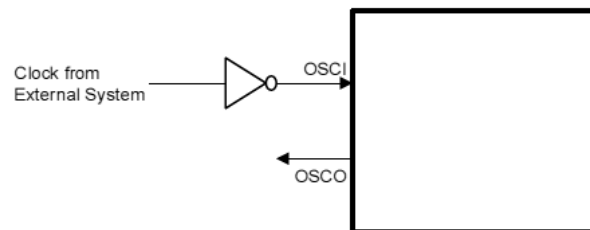
In LF,XT 或 HF 模式下，一台水晶或陶瓷谐振器连接到 OSCI 和 OSCO 管脚建立振荡源。当在 In LF, XT 或 HF 模式下，单片机通过 OSCI 脚接入外部时钟源。使用 ERC 振荡模式为成本节省主要使用在定时无须精确场合下的应用，RC 振荡器频率取决于电阻和电容 (Cext)，操作温度以及其他过程参数。

使用 IRC/ERIC 振荡模式为成本节省主要使用在定时无须精确场合下的应用，单片机具有 4 种不同的振荡频率, 8MHz, 4MHz, 1MHz, and 455KHz, 通过 (RCM<1:0>)来选择一种. 或则用户改变外部电阻来实现。 ERIC 振荡器频率取决于电阻和电容 (Cext), 操作温度以及其他过程参数。

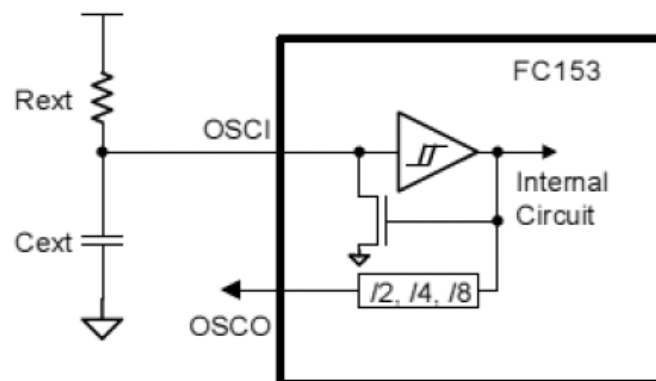
图解 5.9.1: HF, XTor LF 振荡器模式(晶振操作或陶瓷共鸣器)



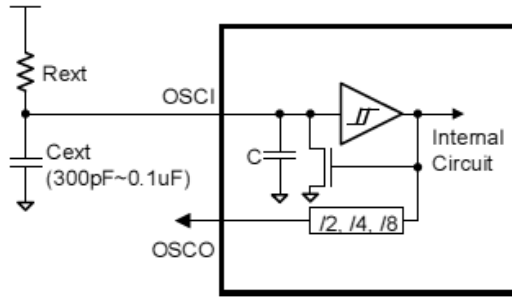
图解 5.9.2: HF,XT or LF 振荡器模式 (外部时钟输入操作)



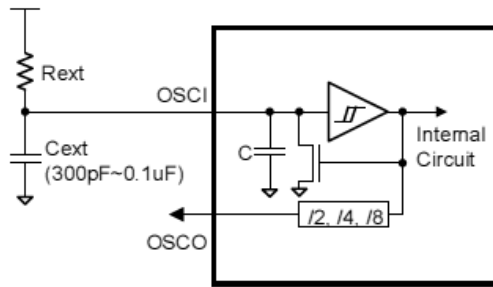
图解 5.9.3: ERC 振荡器模式 (外部的 RC 振荡器)



图解 5.9.4: ERIC 振荡器模式 (外部 R, 内部 C 振荡器)



图解 5.9.5: IRC 振荡器模式 (内部 R, 内部 C 振荡器)



## 5.10 配置选项

表 5.10.1: 配置选项 0

位	名称	说明
2, 1, 0	Fosc<2:0>	振荡源选择位 = 1, 1, 1 → mode (外部的 RC 振荡器) (默认) IOB4/OSCO 管脚为取 OSCOUT 功能 = 1, 1, 0 → HF mode = 1, 0, 1 → XT mode = 1, 0, 0 → LF mode = 0, 1, 1 → IRC mode (internal R & C) IOB4/OSCO 管脚为取 OSCOUT 功能 = 0, 1, 0 → ERIC mode (external R & internal C) IOB4/OSCO 管脚为取 OSCOUT 功能
5, 4, 3	LVDT<2:0>	低电压检测选择位 = 1, 1, 1 → 禁止低电压检测(默认) = 1, 1, 0 → enable, LVDT voltage = 2.0V, 睡眠模式 = 1, 0, 1 → enable, LVDT voltage = 2.0V = 1, 0, 0 → enable, LVDT voltage = 3.6V = 0, 1, 1 → enable, LVDT voltage = 1.8V = 0, 1, 0 → enable, LVDT voltage = 2.2V = 0, 0, 1 → enable, LVDT voltage = 2.4V = 0, 0, 0 → enable, LVDT voltage = 2.6V
7, 6	RCM<1:0>	IRC 选择位 = 1, 1 → 4MHz (默认) = 1, 0 → 8MHz = 0, 1 → 1MHz = 0, 0 → 455KHz

10, 9, 8	SUT<2:0>	PWRT & WDT 计数周期选择位 (其值必须是分频率的倍数) = 1, 1, 1 → PWRT = WDT prescaler rate = 18ms (default) = 1, 1, 0 → PWRT = WDT prescaler rate = 4.5ms = 1, 0, 1 → PWRT = WDT prescaler rate = 288ms = 1, 0, 0 → PWRT = WDT prescaler rate = 72ms = 0, 1, 1 → PWRT = 140us, WDT prescaler rate = 18ms = 0, 1, 0 → PWRT = 140us, WDT prescaler rate = 4.5ms = 0, 0, 1 → PWRT = 140us, WDT prescaler rate = 288ms = 0, 0, 0 → PWRT = 140us, WDT prescaler rate = 72ms
11	OSCOU	IRC/ERIC/ERC 模式下 IOB4/OSCO 功能选择位置 = 1, OSCO (默认) = 0, IOB4
12	RSTBIN	IOB3/RSTB 选择位置 = 1, IOB3 (默认) = 0, RSTB

表 5.10.2: 配置选项 1

位	名称	说明
0	WDTEN	看门狗使能位 = 1, 使能 WDT (默认) = 0, 禁止 WDT
1	PROTECT	代码保护选择位 = 1, 1 → 代码不加密 EPROM code protection off (默认) = 0, 0 → 代码加密 EPROM code protection on
3, 2	OSCD<1:0>	指令运行周期选择位 = 1, 1 → 4 个振荡周期 (默认) = 1, 0 → 2 个振荡周期 = 0, 1 → 1 个振荡周期 = 0, 0 → 8 个振荡周期
4	PMOD	省电模式控制位 = 1, 非省电模式 (默认) = 0, 省电模式
5	RDPORT	IO 作为输出时, 读端口方式控制位 = 1, 从寄存器读 (默认) = 0, 从管脚读
6	SCHMITT	I/O 输入缓冲控制位 = 1, 通过 Schmitt 触发器 (默认) = 0, 不通过 Schmitt 触发器
12 ~ 7	-	没有使用

表 5.10.3: 配置选项 2

位	名称	说明
4 ~ 0	CAL<3:0>	IRC 方式选择位
12 ~ 5	-	没有使用

表 5.10.4: Selection of IOB5/OSCI and IOB4/OSCO Pins

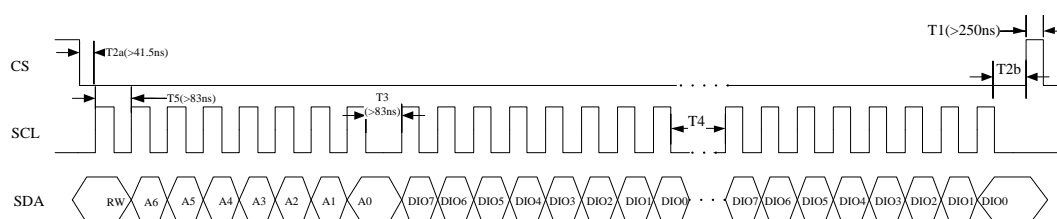
振荡方式	IOB5/OSCI	IOB4/OSCO
IRC/ERIC	IOB5 (OSCIN=0)	IOB4/OSCO selected by OSCOUT bit
	OSCI (OSCIN=1)	IOB4/OSCO selected by OSCOUT bit
ERC	OSCI	IOB4/OSCO selected by OSCOUT bit
HF	OSCI	OSCO
LF	OSCI	OSCO





## 5.11.2 三线制 SPI 接口

MCU通过内部自定义的三线制SPI接口和RF模块进行通信。



自定义的三线制 SPI 接口：标准 SPI 接口中的 MOSI 和 MISO 共用一根数据线 SDA，其工作时序和标准 SPI 接口一样。在使用三线 SPI 接口时，需要先写寄存器使能，即上电后将 **reg0x2A 从 0xA001 改写为 0x2001**。

Notes:

- 1、SPI 为下降沿采样数据，上升沿改变数据；
- 2、SPI 读写位： 写=0，读=1；
- 3、访问接收 FIFO 寄存器 0X28 时，可以按字节读。访问多个 FIFO 数据时可以用一个 CS 周期；
- 4、访问除 FIFO 外的其他寄存器时，一次要读写 16bits；
- 5、访问除 FIFO 外的其他多个寄存器时，可以用一个 CS 周期。此时，地址只要写一次，然后是 16bit 数据，当写完一个寄存器值后，芯片会自动增加寄存器地址。

Name	Min	Typ.	Max	Description
T1	250ns			两次 SPI 访问的间隔时间
T2a, T2b	41.5ns			CS 和 SPI_CLK 的间隔
T3	Note 1			地址和数据间隔时间
T4	Note 1			高位字节和低位字节的时间间隔
T5	83ns			SPI_CLK 时钟周期
T6	Note 2			两个寄存器数据的时间间隔

Notes:

1. 在访问寄存器 0x28 中的 FIFO 数据时，芯片需要 450ns 去找到正确的读 FIFO 读取的指针地址。
2. 当读寄存器 0x28 中的 FIFO 数据时，至少需要等 450ns。读其他寄存器时， $T6_{min}=41.5ns$ 。

### 5.11.3 RF 模块寄存器默认值及优化值

RF模块复位之后，所有寄存器均为默认值，如下表所示：

Address	reset value	Address	reset value
0x00	0x0030	0x15	Read only
0x01	0x2077	0x16	Read only
0x02	0x4060	0x17	0x0000
0x03	0x5800	0x18	0x6FE1
0x04	0x4A00	0x19	0x1300
0x05	0x7FA6	0x1a	0x00F7
0x06	0x1988	0x1b	0x1800
0x07	0x7311	0x1c	0x4008
0x08	0x1659	0x1d	空
0x09	0x007B	0x1e	0x7FF4
0x0a	0x20A3	0x1f	0x0101
0x0b	0x837F	0x20	0x0202
0x0c	0x3E11	0x21	0x0303
0x0d	0x6003	0x22	0x0404
0x0e	空	0x23	0x8001
0x0f	0x661D	0x24	0x4401
0x10	0x5F8F	0x25	0x0000
0x11	Read only	0x26	0x0000
0x12	Read only	0x27	0x0000
0x13	Read only	0x28	0x0000
0x14	空	0x29	0x0000
		0x2A	0xA001

在通信之前，需要对部分寄存器重新写入优化值，如下表所示：

Address	Opt value
0x0a	0x2053
0x03	0x5810

## 5.11.4 一般通信流程

### 1、上电和寄存器初始化



- 1) 模块内部集成上电复位功能（POR），复位时间（T1）约 0.5ms
- 2) T2 是晶振稳定时间，约为 1.5ms，然后由 MCU 初始化寄存器
- 3) 寄存器初始化完成后（即对部分寄存器写入优化值），模块可以开始发射或接收数据
- 4) 除了内部自动 POR 复位，模块还集成了软件复位功能：  
先写 0x1e[0]寄存器为‘1’，使能软件复位功能，再对 0x01[7]寄存器写‘1’就可以完成复位操作，复位之后，所有 RF 模块的寄存器值都变成默认值。

### 2、TX流程

- a、初始化寄存器，即写入需要优化的寄存器值

```
write reg[0x0a] = 0x2053;
```

```
write reg[0x03] = 0x5810;
```

- b、清空发送 FIFO

```
write reg[0x26] = 0x8080;
```

- c、写数据到FIFO（假如发送数据：0x05 0x01 0x02 0x03 0x04 0x05，第一个字节表示长度）

```
write reg[0x27] = 0x0501;
```

```
write reg[0x27] = 0x0203;
```

```
write reg[0x27] = 0x0405;
```

- d、启动发送使能，同时设置频道

```
write reg[0x00] = 0x80XX;//低 7 位为频道号
```

- e、等待 pkt\_flag 为高，表示发送完成

## 2、RX流程

a、初始化寄存器，即写入需要优化的寄存器值

```
write reg[0x0a] = 0x2053;
```

```
write reg[0x03] = 0x5810;
```

b、清空接收 FIFO

```
write reg[0x26] = 0x8080;
```

c、启动接收使能，同时设置频道

```
write reg[0x00] = 0x80XX;//低7位为频道号
```

d、等待 pkt\_flag 为高，表示接收到一帧数据，然后 MCU 可以从接收 FIFO 读数据,读出

的第一个字节为数据长度，后面的字节为数据

```
read reg[0x28]
```

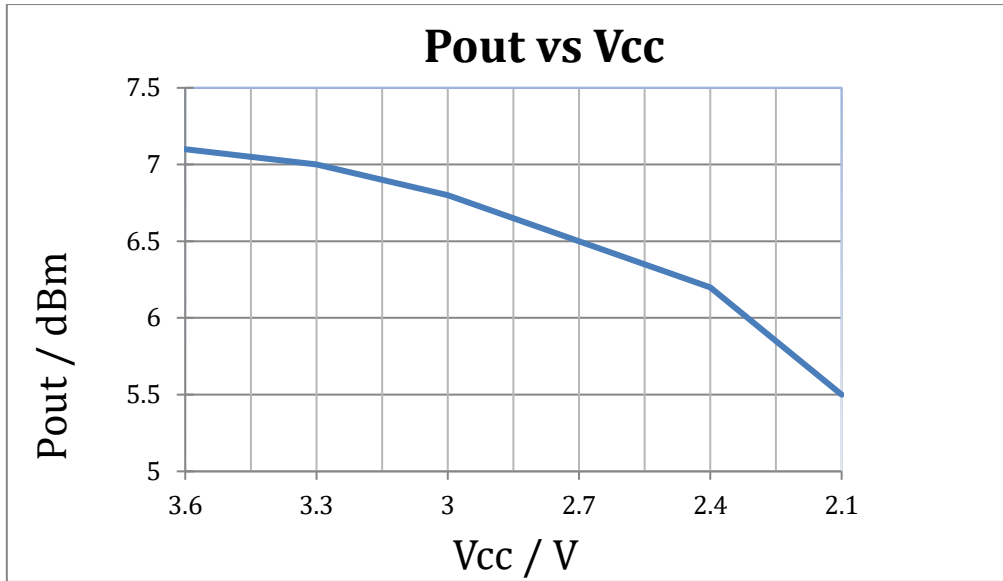
```
read reg[0x28], .....
```

### 5.11.5 输出功率配置

PA 的输出功率可以通过 reg0x02 寄存器来设置。最大输出功率可到+8dBm, 最小可到-24dBm。下表列出部分功率配置（如果需要其它功率配置，请联系公司技术人员）：

Reg0x02	Pout (dBm)	Ivcc (mA)
0x4060	7	34
0x4061	6	30
0x4065	3	23
0x4067	1	20
0x2020	0	16
0x2061	-2	15
0x2064	-5	13
0x2066	-7	12
0x2068	-10	11.6
0x306A	-13	11
0x307A	-16	9
0x306F	-24	8

电源电压的变化会导致输出功率也随着变化，在实际应用的时候需要注意。以下是输出功率和电源电压的对应关系：



## 6. 指令集合

操作语法	说明	操作内容	指令周期	影响标志位
<b>BCR</b> R, bit	Clear bit in R	0→R<b>	1	-
<b>BSR</b> R, bit	Set bit in R	1→R<b>	1	-
<b>BTRSC</b> R, bit	Test bit in R, Skip if Clear	Skip if R<b> = 0	1/2 <sup>(1)</sup>	-
<b>BTRSS</b> R, bit	Test bit in R, Skip if Set	Skip if R<b> = 1	1/2 <sup>(1)</sup>	-
<b>NOP</b>	No Operation	No operation	1	-
<b>CLRWDT</b>	Clear Watchdog Timer	00h→WDT, 00h →WDT prescaler	1	TO,PD
<b>OPTION</b>	Load OPTION register	ACC→ OPTION	1	-
<b>SLEEP</b>	Go into power-down mode	00h→ WDT, 00h→ WDT prescaler	1	TO,PD
<b>DAA</b>	Adjust ACC's data format from HEX to DEC after any addition operation	ACC(hex) → ACC(dec)	1	C
<b>DAS</b>	Adjust ACC's data format from HEX to DEC after any subtraction operation	ACC(hex) → ACC(dec)	1	-
<b>INT</b>	S/W interrupt	PC + 1→Top of Stack, 002h→ PC	2	-
<b>RETURN</b>	Return from subroutine	Top of Stack→ PC	2	-
<b>RETFIE</b>	Return from interrupt, set GIE bit	Top of Stack→ PC, 1→ GIE	2	-
<b>CLRA</b>	Clear ACC	00h ACC	1	Z
<b>IOST</b> R	Load IOST register	ACC→ IOST register	1	-
<b>CLRR</b> R	Clear R	00h→ R	1	Z

<b>MOVAR R</b>	Move ACC to R	ACC→ R	1	-
<b>MOVR R, d</b>	Move R	R→ dest	1	Z
<b>DECR R, d</b>	Decrement R	R - 1 →dest	1	Z
<b>DECRSZ R, d</b>	Decrement R, Skip if 0	R - 1→ dest, Skip if result = 0	1/2 <sup>(1)</sup>	-
<b>INCR R, d</b>	Increment R	R + 1→ dest	1	Z
<b>INCRSZ R, d</b>	Increment R, Skip if 0	R + 1→ dest, Skip if result = 0	1/2 <sup>(1)</sup>	-
<b>ADDAR R, d</b>	Add ACC and R	R + ACC→ dest	1	C, DC, Z
<b>SUBAR R, d</b>	Subtract ACC from R	R - ACC→ dest	1	C, DC, Z
<b>ADCAR R, d</b>	Add ACC and R with Carry	R + ACC + C → dest	1	C, DC, Z
<b>SBCAR R, d</b>	Subtract ACC from R with Carry	R + ACC + C→dest	1	C, DC, Z
<b>ANDAR R, d</b>	AND ACC with R	ACC and R→dest	1	Z
<b>IORAR R, d</b>	Inclusive OR ACC with R	ACC or R→ dest	1	Z
<b>XORAR R, d</b>	Exclusive OR ACC with R	R xor ACC→ dest	1	Z
<b>COMR R, d</b>	Complement R	R→dest	1	Z
<b>RLR R, d</b>	Rotate left f through Carry	R<7>→C, R<6:0>→ dest<7:1>, C→ dest<0>	1	C
<b>RRR R, d</b>	Rotate right f through Carry	C ←dest<7>, R<7:1> ← dest<6:0>, R<0>← C	1	C
<b>SWAPR R, d</b>	Swap R	R<3:0> →dest<7:4>, R<7:4> →dest<3:0>	1	-
<b>MOVIA I</b>	Move Immediate to ACC	I →ACC	1	-
<b>ADDIA I</b>	Add ACC and Immediate	I + ACC →ACC	1	C, DC, Z
<b>SUBIA I</b>	Subtract ACC from Immediate	I - ACC →ACC	1	C, DC, Z
<b>ANDIA I</b>	AND Immediate with ACC	ACC and I →ACC	1	Z
<b>IORIA I</b>	OR Immediate with ACC	ACC or I →ACC	1	Z
<b>XORIA I</b>	Exclusive OR Immediate to ACC	ACC xor I →ACC	1	Z
<b>RETIA I</b>	Return, place Immediate in ACC	I ACC, Top of Stack → PC	2	-
<b>CALL I</b>	Call subroutine	PC + 1 →Top of Stack, I → PC	2	-
<b>GOTO I</b>	Unconditional branch	I →PC	2	-

注释: 1. 两周期指令为分支跳转指令

2. bit : Bit 地址为 8 位寄存器 R 中的某一位

R : 寄存器地址 (00h to 3Fh)

I :立即数

ACC : 累加器

d : 目的选择;

=0 (结果存放在 ACC)

=1 (结果存放在 R)

dest : 目的地

PC : 程序指针

PCHBUF : 高位缓冲程序指针

WDT : 看门狗计数器

**GIE** :中断允许总控制位  
**TO** : 计数溢出位  
**PD** : 省电模式选择位  
**C** : 进位/借位标志  
**DC** : 辅助进位/借位标志.(低四位向高四位进位/借位标志)  
**Z** : 零标志

<b>ADCAR(带进位加法)</b>	<b>Add ACC and R with Carry</b>
语 法	ADCAR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$R + ACC + C \rightarrow dest$
受影响的标志	C, DC, Z
说 明	將 A 寄存器的內含值加上 R 寄存器的內含值（带进位），如果 ‘d’ 是 0 结果在 ACC 中存放。 如果 ‘d’ 是 ‘1’ 结果在 ‘R’ 中存放’。
指令执行周期	1
<b>ADDAR (加法指令)</b>	<b>ACC and R with Carry</b>
语 法	ADDAR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$R + ACC \rightarrow dest$
受影响的标志	C, DC, Z
说 明	將 A 寄存器的內含值加上 R 寄存器的內含值（不带进位），如果 ‘d’ 是 0 结果在 ACC 中存放。 如果 ‘d’ 是 ‘1’ 结果在 ‘R’ 中存放’。
指令执行周期	1
<b>ADDIA</b>	<b>Add ACC and Immediate</b>
语 法	ADDIA I
操作数	$0 \leq I \leq 255$
操作内容	$ACC + I \rightarrow ACC$
受影响的标志	C, DC, Z
说 明	將 A 寄存器的內含值加上立即数 ‘I’ ，结果在 ACC 中存放。
指令执行周期	1
<b>ANDAR</b>	<b>AND ACC and R</b>
语 法	ANDAR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$ACC \text{ and } R \rightarrow dest$
受影响的标志	Z
说 明	將 A 寄存器內含值和 R 寄存器做相与操作，如果 ‘d’ 是 0 结果在 ACC 中存放。 如果 ‘d’ 是 ‘1’ 结果在 ‘R’ 中存放’。
指令执行周期	1
<b>ANDIA</b>	<b>AND Immediate with ACC</b>

语 法	ANDIA I
操作数	$0 \leq I \leq 255$
操作内容	ACC and I $\rightarrow$ dest
受影响的标志	Z
说 明	将 A 寄存器的内含值与立即数 ‘I’ 做相与操作，结果在 ACC 中存放
指令执行周期	1
<b>BSR</b>	<b>Set Bit in R</b>
语 法	BCF R, b
操作数	$0 \leq R \leq 63$ $0 \leq b \leq 7$
操作内容	$1 \rightarrow R<b>$
受影响的标志	无
说 明	R 寄存器的位 “b” 被设成 1
指令执行周期	1
<b>BTRSC</b>	<b>Test Bit in R, Skip if Clear</b>
语 法	BTRSC R, b
操作数	$0 \leq R \leq 63$ $0 \leq b \leq 7$
操作内容	当 $R<b> = 0$ 跳过下条指令
受影响的标志	无
说 明	$R<b> = 0$ 跳过下条指令 $R<b> = 0$ 时，该指令周期中提取的下条指令被丢弃，并以执行 NOP 操作来替换这条 2 周期指令。
指令执行周期	1(2)
<b>BTRSS</b>	<b>Test Bit in R, Skip if Set</b>
语 法	BTRSS R, b
操作数	$0 \leq R \leq 63$ $0 \leq b \leq 7$
操作内容	当 $R<b> = 1$ 跳过下条指令
受影响的标志	无
说 明	$R<b> = 1$ 跳过下条指令 $R<b> = 1$ 时，该指令周期中提取的下条指令被丢弃，并以执行 NOP 操作来替换这条 2 周期指令。
指令执行周期	1(2)
<b>CALL</b>	<b>Subroutine Call</b>
语 法	CALL I
操作数	$0 \leq I \leq 1023$
操作内容	PC +1 $\rightarrow$ Top of Stack; I $\rightarrow$ PC
受影响的标志	无
说 明	子程序调用。首先下一条指令地址(PC+1)进栈。10 位立即地址被装载入 PC 指针的位<9 : 0>. CALL 是二周期指令。
指令执行周期	2
<b>CLRA</b>	<b>Clear ACC</b>

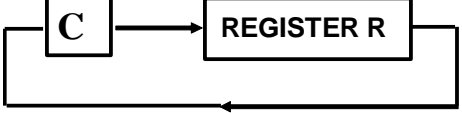



语法	CLRA
操作数	无
操作内容	00h → ACC I → Z
受影响的标志	Z
说明	ACC 被清零, Z 标志为置 1
指令执行周期	1
<hr/>	
<b>CLRR</b>	<b>Clear R</b>
语法	CLRR R
操作数	0 ≤ R ≤ 63
操作内容	00h → R I → Z
受影响的标志	Z
说明	R 被清零, Z 标志为置 1
指令执行周期	1
<hr/>	
<b>CLRWDT</b>	<b>Clear Watchdog Timer</b>
语法	CLRWDT
操作数	无
操作内容	00h → WDT; 00h → WDT prescaler (已经设置了 WDT 预置器); 1 → TO; 1 → PD
受影响的标志	TO,PD
说明	CLRWDT 指令重置 WDT, 如已经设置了 WDT 预置器, 也重置 WDT 预置器; 并把 TO,PD 位置 1
指令执行周期	1
<hr/>	
<b>COMR</b>	<b>Complement R</b>
语法	COMR R, d
操作数	0 ≤ R ≤ 63 d ∈ [0,1]
操作内容	R → dest
受影响的标志	Z
说明	将 R 内含内容取补数, 如果 ‘d’ 是 0 结果在 ACC 中存放。 如果 ‘d’ 是 1 结果在 ‘R’ 中存放。
指令执行周期	1
<hr/>	
<b>DAA</b>	<b>Adjust ACC's data format from HEX to DEC</b>
语法	DAA
操作数	无
操作内容	ACC(hex) → ACC(dec)
受影响的标志	C
说明	在有些加法操作以后把 ACC 内值的十六进制转化十进制,
指令执行周期	1
<hr/>	
<b>DAS</b>	<b>Adjust ACC's data format from HEX to DEC</b>
语法	DAS
操作数	无
操作内容	ACC(hex) → ACC(dec)
受影响的标志	C
说明	在有些减法操作以后把 ACC 内值的十六进制转化十进制,
指令执行周期	1

<b>DECR</b>	<b>Decrement R</b>
语 法	DECR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$R - 1 \rightarrow \text{dest}$
受影响的标志	Z
说 明	递减 R 寄存器的值，如果 ‘d’ 是 0 结果在 ACC 中存放。如果 ‘d’ 是 1 结果在 ‘R’ 中存放。
指令执行周期	1
<b>DECRSZ</b>	<b>Decrement R, Skip if 0</b>
语 法	DECRSZ R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$R - 1 \rightarrow \text{dest}$ 如果结果等于 0，跳过下条指令
受影响的标志	无
说 明	递减 R 寄存器的值，如果 ‘d’ 是 0 结果在 ACC 中存放。如果 ‘d’ 是 1 结果在 ‘R’ 中存放。 如果结果等于 0，该指令周期中提取的下条指令被丢弃，并以执行 NOP 操作来替换这条 2 周期指令。
指令执行周期	1(2)
<b>GOTO</b>	<b>Unconditional Branch</b>
语 法	GOTO I
操作数	$0 \leq I \leq 1023$
操作内容	$I \rightarrow \text{PC}$
受影响的标志	无
说 明	无条件跳转。10 位立即地址被装载入 PC 指针的位<9 : 0>。GOTO 是二周期指令。
指令执行周期	2
<b>INCR</b>	<b>Increment R</b>
语 法	INCR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$R + 1 \rightarrow \text{dest}$
受影响的标志	Z
说 明	将被指定 R 寄存器的内含值加 1，如果 ‘d’ 是 0 结果在 ACC 中存放。如果 ‘d’ 是 1 结果在 ‘R’ 中存放。
指令执行周期	2
<b>INCRSZ</b>	<b>Increment R, Skip if 0</b>
语 法	INCRSZ R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$R + 1 \rightarrow \text{dest}$ 如果结果等于 0，跳过下条指令
受影响的标志	无
说 明	将被指定 R 寄存器的内含值加 1，如果 ‘d’ 是 0 结果在 ACC 中存放。如果 ‘d’ 是 1 结果在 ‘R’ 中存放。 如果结果等于 0，该指令周期中提取的下条指令被丢弃，并以执行 NOP 操作来替换这条 2 周期指令。
指令执行周期	1(2)
<b>INT</b>	<b>S/W Interrupt</b>

语法	INT
操作数	无
操作内容	PC +1 → Top of Stack; 002h → PC
受影响的标志	无
说明	子程序调用。首先下一条指令地址(PC+1)进栈。10 位地址 002h 被装载入 PC 指针的位<9 : 0>. CALL 是二周期指令。
指令执行周期	2
<b>IORAR</b>	<b>OR ACC with R</b>
语法	<b>IORAR</b>
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	ACC or R → dest
受影响的标志	Z
说明	将 A 寄存器内含值和 R 寄存器做或操作, 如果 'd' 是 0 结果在 ACC 中存放。如果 'd' 是 1 结果在 'R' 中存放'。
指令执行周期	1
<b>IORIA</b>	<b>OR Immediate with ACC</b>
语法	<b>IORIA I</b>
操作数	$0 \leq I \leq 255$
操作内容	ACC or I → dest
受影响的标志	Z
说明	将 A 寄存器的内含值与立即数 'I' 做相与操作, 结果在 ACC 中存放
指令执行周期	1
<b>IOST</b>	<b>Load IOST Register</b>
语法	<b>IOST R</b>
操作数	R = 5 or 6
操作内容	ACC → IOST register R
受影响的标志	无
说明	将 A 寄存器的内含值加载到 IOST register R 中
指令执行周期	1
<b>MOVAR</b>	<b>Move ACC to R</b>
语法	<b>MOVAR R</b>
操作数	$0 \leq R \leq 63$
操作内容	ACC → R
受影响的标志	无
说明	将数据从 ACC 传送到 R
指令执行周期	1
<b>MOVIA</b>	<b>Move Immediate to ACC</b>
语法	<b>MOVIA I</b>
操作数	$0 \leq I \leq 255$
操作内容	I → ACC
受影响的标志	无
说明	将立即值载入 A 寄存器中
指令执行周期	1
<b>MOVR</b>	<b>Move Immediate to ACC</b>

语法	MOVR R, d
操作数	$0 \leq R \leq 63$ $d \in [0, 1]$
操作内容	$R \rightarrow dest$
受影响的标志	无
说明	将 A 寄存器内容载入 R 中,, 如果 ‘d’ 是 0 结果在 ACC 中存放。 如果 ‘d’ 是 1 结果在 ‘R’ 中存放’。 ‘d’ 为 1 用来测试该寄存器对标志 Z 是否有影响
指令执行周期	1
<b>NOP</b>	<b>No Operation</b>
语法	NOP
操作数	无
操作内容	无操作
受影响的标志	无
说明	不做任何操作
指令执行周期	1
<b>OPTION</b>	<b>Load OPTION Register</b>
语法	OPTION
操作数	无
操作内容	$ACC \rightarrow OPTION$
受影响的标志	无
说明	将 A 寄存器内容载入 OPTION 中
指令执行周期	1
<b>RETFIE</b>	<b>Return from Interrupt, Set ‘GIE’ Bit</b>
语法	RETFIE
操作数	无
操作内容	$Top\ of\ Stack \rightarrow PC$
受影响的标志	无
说明	程序计数器载入堆栈返回地址。 ‘GIE’ 位被设置到 1。 这是二周期指令。
指令执行周期	2
<b>RETIA</b>	<b>Return with Immediate in ACC</b>
语法	RETIA I
操作数	$0 \leq I \leq 255$
操作内容	$I \rightarrow ACC;$ $Top\ of\ Stack \rightarrow PC$
受影响的标志	无
说明	程序计数器载入堆栈返回地址, 并把立即数送入 A 中。这是二周期指令。
指令执行周期	2
<b>RETURN</b>	<b>Return from Subroutine</b>
语法	RETURN
操作数	无
操作内容	$Top\ of\ Stack \rightarrow PC$
受影响的标志	无
说明	程序计数器载入堆栈返回地址。这是二周期指令。
指令执行周期	2
<b>RLR</b>	<b>Rotate Left f through Carry</b>

语法	R LR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$R \langle 7 \rangle \rightarrow C$ ; $R \langle 6:0 \rangle \rightarrow \text{dest} \langle 7:1 \rangle$ ; $C \rightarrow \text{dest} \langle 0 \rangle$
受影响的标志	C
说明	R 寄存器的内含值又移 1-bit，右移时包含 C(进位标志)，如下图，结果存放由 'd' 决定，如果 'd' 是 0 结果在 ACC 中存放。如果 'd' 是 '1' 结果在 'R' 中存放。
	
指令执行周期	1
<hr/>	
<b>RRR</b>	<b>Rotate Right f through Carry</b>
语法	RRR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$C \rightarrow \text{dest} \langle 7 \rangle$ ; $R \langle 7:1 \rangle \rightarrow \text{dest} \langle 6:0 \rangle$ ; $R \langle 0 \rangle \rightarrow C$
受影响的标志	C
说明	R 寄存器的内含值又移 1-bit，右移时包含 C(进位标志)，如下图，结果存放由 'd' 决定，如果 'd' 是 0 结果在 ACC 中存放。如果 'd' 是 '1' 结果在 'R' 中存放。
	
指令执行周期	1
<hr/>	
<b>SLEEP</b>	<b>SLEEP</b>
语法	SLEEP
操作数	无
操作内容	00h → WDT; 00h → WDT prescaler; 1 → TO; 0 → PD
受影响的标志	TO, PD
说明	TO 位置 1。PD 位清零，WDT 和 WDT 预置器清零 单片机进入睡眠模式
指令执行周期	1
<hr/>	
<b>SBCAR (带借位加法)</b>	<b>Subtract ACC from R with Carry</b>
语法	SBCAR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$(R - \text{ACC} - C) \rightarrow \text{dest}$
受影响的标志	C, DC, Z
说明	将 R 寄存器的内含值减去 A 寄存器的内含值 (带借位)，如果 'd' 是 0 结果在 ACC 中存放。如果 'd' 是 '1' 结果在 'R' 中存放。
指令执行周期	1

<b>SUBAR</b>	<b>Subtract ACC from R</b>
语法	SUBAR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$R - ACC \rightarrow dest$
受影响的标志	C, DC, Z
说明	将 R 寄存器的内含值减去 A 寄存器的内含值（不带借位），如果 ‘d’ 是 0 结果在 ACC 中存放。如果 ‘d’ 是 ‘1’ 结果在 ‘R’ 中存放。
指令执行周期	1
<b>SUBIA</b>	<b>Subtract ACC from Immediate</b>
语法	SUBIA I
操作数	$0 \leq I \leq 255$
操作内容	$ACC - I \rightarrow ACC$
受影响的标志	C, DC, Z
说明	将 A 寄存器的内含值减去立即数 ‘I’，结果在 ACC 中存放。
指令执行周期	1
<b>SWAPR</b>	<b>Swap nibbles in R</b>
语法	SWAPR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$R<3:0> \rightarrow dest<7:4>;$ $R<7:4> \rightarrow dest<3:0>$
受影响的标志	无
说明	将所选定的寄存器，高 4 位以及低 4 位，如果 ‘d’ 是 0 结果在 ACC 中存放。如果 ‘d’ 是 1 结果在 ‘R’ 中存放。
指令执行周期	1
<b>XORAR</b>	<b>Exclusive OR ACC with R</b>
语法	SWAPR R, d
操作数	$0 \leq R \leq 63$ $d \in [0,1]$
操作内容	$ACC \text{ xor } R \rightarrow dest R$
受影响的标志	Z
说明	将 A 寄存器的值和 R 寄存器的值 XOR 在一起，如果 ‘d’ 是 0 结果在 ACC 中存放。如果 ‘d’ 是 ‘1’ 结果在 ‘R’ 中存放。
指令执行周期	1
<b>XORIA</b>	<b>Exclusive OR Immediate with ACC</b>
语法	XORIA I
操作数	$0 \leq I \leq 255$
操作内容	$ACC \text{ xor } I \rightarrow ACC$
受影响的标志	Z
说明	将 A 寄存器的值和立即数 ‘I’ XOR 在一起，结果在 ACC 中存放。
指令执行周期	1

---

## 7. 版本更新历史

版本号	创建时间	创建内容	创建者	备注
XP9952HP_V1.0	2020.8.5	该文档是从 XP9952BP_V1.1 更新而来： 相比 XP9952BP, 3 脚 LDO_OUT 变成 PA7	ZH	