

## 技术相关信息

- [应用范例](#)  
– [HA0075S MCU 复位电路和振荡电路的应用范例](#)

## 特性

### CPU 特性

- 工作电压：
  - $f_{SYS}=4\text{MHz}$ : 2.2V~5.5V
  - $f_{SYS}=8\text{MHz}$ : 3.0V~5.5V
  - $f_{SYS}=12\text{MHz}$ : 4.5V~5.5V
- $V_{DD}=5\text{V}$ , 系统时钟为 12MHz 时, 指令周期为 0.33 $\mu\text{s}$
- 提供休眠和唤醒功能, 以降低功耗
- 五种振荡模式:
  - 外部高频晶振 -- HXT
  - 外部低频晶振 -- LXT
  - 外部 RC -- ERC
  - 内部高速 RC -- HIRC
  - 内部低速 RC -- LIRC
- 多种工作模式: 正常、低速和休眠
- 内部集成 4MHz, 8MHz 和 12MHz 振荡器, 无需外接元件
- OTP 程序存储: 1K $\times$ 15 ~ 2K $\times$ 15
- 看门狗定时器功能

- LIRC 振荡用于看门狗时钟
- RAM 数据存储: 96 $\times$ 8
- 所有指令都可在 1 或 2 个指令周期内完成
- 查表指令
- 63 条指令
- 6 层堆栈
- 位操作指令

### 周边特性

- 10 个双向输入/输出口
- 一个与 I/O 口共用引脚的外部中断输入
- 两个 8 位可编程定时/计数器, 具有溢出中断和预分频器功能
- 时基功能
- 4 通道 12 位分辨精度的 A/D 转换器
- 1 通道 8 位 PWM 功能
- 低电压复位功能
- 可编程分频器-- PFD
- 封装类型: 10-pin MSOP, 16-pin NSOP

## 概述

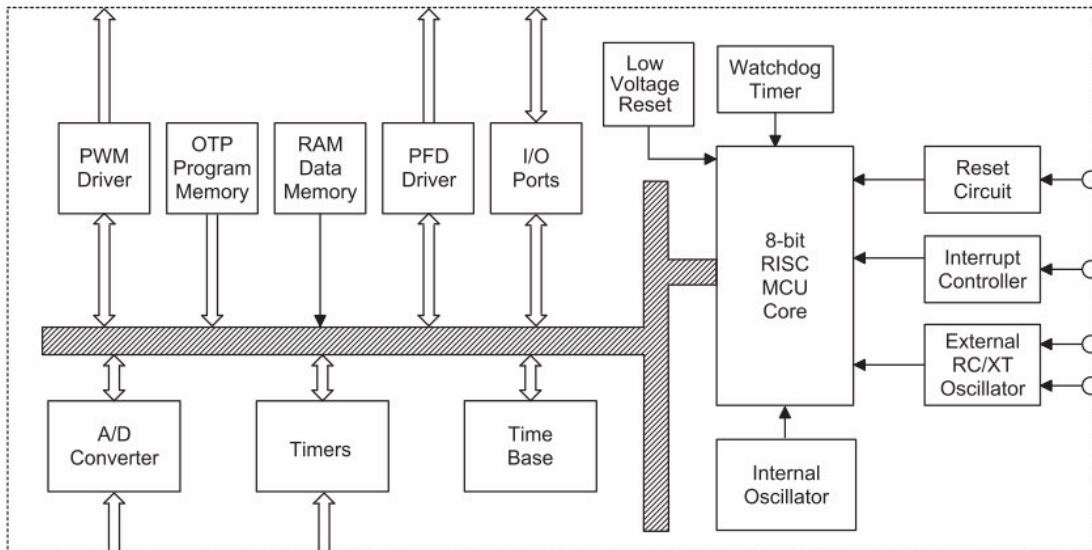
HT4XR0XX 系列单片机是一款小封装型具有 8 位高性能精简指令集单片机。该系列单片机具有低功耗、I/O 灵活、定时器功能、振荡类型可选、休眠和唤醒功能、看门狗和低电压复位等丰富的功能选项，且具有极高的性价比，其内部集成了系统振荡器 HIRC，不需要增加外部元器件，提供三种频率选择，可以广泛适用于各种应用，例如工业控制、消费类产品、家用电器子系统控制等。

## 选型表

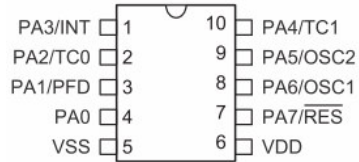
单片机型号	ROM	RAM	I/O	Timer	Time Base	Interrupt		A/D	PWM	Stack	封装类型
						Ext.	Int.				
HT48R01B	1K×15	96×8	8	8-bit×2	1	1	3	—	—	6	10MSOP
HT48R02B	2K×15	96×8	8	8-bit×2	1	1	3	—	—	6	10MSOP
HT46R01B	1K×15	96×8	8	8-bit×2	1	1	4	12-bit×4	8-bit×1	6	10MSOP
HT46R02B	2K×15	96×8	8	8-bit×2	1	1	4	12-bit×4	8-bit×1	6	10MSOP
HT48R01N	1K×15	96×8	10	8-bit×2	1	1	3	—	—	6	16NSOP
HT48R02N	2K×15	96×8	10	8-bit×2	1	1	3	—	—	6	16NSOP
HT46R01N	1K×15	96×8	10	8-bit×2	1	1	4	12-bit×4	8-bit×1	6	16NSOP
HT46R02N	2K×15	96×8	10	8-bit×2	1	1	4	12-bit×4	8-bit×1	6	16NSOP

## 方框图

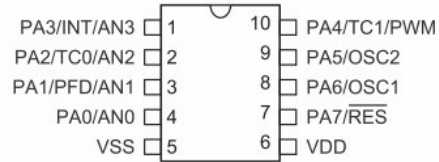
主要功能模块的方框图



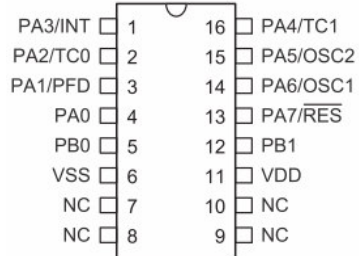
引脚图



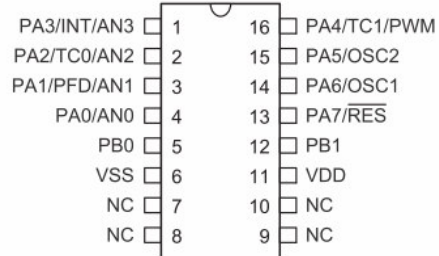
**HT48R01B/HT48R02B**  
**10 MSOP-A**



**HT46R01B/HT46R02B**  
**10 MSOP-A**



**HT48R01N/HT48R02N**  
**16 NSOP-A**



**HT46R01N/HT46R02N**  
**16 NSOP-A**

## 引脚说明

### HT46R01B/HT46R02B

引脚名称	功能	OPT	I/T	O/T	说明
PA0/AN0	PA0	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	AN0	ADCR	AN	—	A/D 通道 0
PA1/PFD/AN1	PA1	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	PFD	CTRL0	—	CMOS	PFD 输出
	AN1	ADCR	AN	—	A/D 通道 1
PA2/TC0/AN2	PA2	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	TC0	—	ST	—	外部定时器 0 时钟源输入脚
	AN2	ADCR	AN	—	A/D 通道 2
PA3/INT/AN3	PA3	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	INT	—	ST	—	外部中断输入脚
	AN3	ADCR	AN	—	A/D 通道 3
PA4/TC1/PWM	PA4	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	TC1	—	ST	—	外部定时器 1 时钟源输入脚
	PWM	CTRL0	—	CMOS	PWM 输出
PA5/OSC2	PA5	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	OSC2	CO	—	OSC	振荡器引脚
PA6/OSC1	PA6	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	OSC1	CO	OSC	—	振荡器输入脚
PA7/ $\overline{\text{RES}}$	PA7	PAWK	ST	NMOS	通用 I/O 口, 可通过寄存器设置唤醒功能
	RES	CO	ST	—	复位输入脚
VDD	VDD	—	PWR	—	正电源
VSS	VSS	—	PWR	—	负电源, 接地

注: I/T: 输入类型; O/T: 输出类型

OPT: 通过配置选项 (CO) 或寄存器选项来设置

PWR: 电源; CO: 配置选项; ST: 斯密特触发输入

CMOS: CMOS 输出; NMOS: NMOS 输出

这里需要重点指出, PB0 和 PB1 并未连接到内部 10-pin MSOP 引脚上。其默认为输入状态, 为了防止输入脚浮空而产生功耗, 设计者可通过 PBPU 寄存器设置其具有上拉电阻功能。

**HT48R01B/HT48R02B**

引脚名称	功能	OPT	I/T	O/T	说明
PA0	PA0	PAPU PAWK	ST	CMOS	通用 I/O 口，可通过寄存器设置带上拉电阻和唤醒功能
PA1/PFD	PA1	PAPU PAWK	ST	CMOS	通用 I/O 口，可通过寄存器设置带上拉电阻和唤醒功能
	PFD	CTRL0	—	CMOS	PFD 输出
PA2/TC0	PA2	PAPU PAWK	ST	CMOS	通用 I/O 口，可通过寄存器设置带上拉电阻和唤醒功能
	TC0	—	ST	—	外部定时器 0 时钟源输入脚
PA3/INT	PA3	PAPU PAWK	ST	CMOS	通用 I/O 口，可通过寄存器设置带上拉电阻和唤醒功能
	INT	—	ST	—	外部中断输入脚
PA4/TC1	PA4	PAPU PAWK	ST	CMOS	通用 I/O 口，可通过寄存器设置带上拉电阻和唤醒功能
	TC1	—	ST	—	外部定时器 1 时钟源输入脚
PA5/OSC2	PA5	PAPU PAWK	ST	CMOS	通用 I/O 口，可通过寄存器设置带上拉电阻和唤醒功能
	OSC2	CO	—	OSC	振荡器引脚
PA6/OSC1	PA6	PAPU PAWK	ST	CMOS	通用 I/O 口，可通过寄存器设置带上拉电阻和唤醒功能
	OSC1	CO	OSC	—	振荡器输入脚
PA7/ $\overline{\text{RES}}$	PA7	PAWK	ST	NMOS	通用 I/O 口，可通过寄存器设置唤醒功能
	$\overline{\text{RES}}$	CO	ST	—	复位输入脚
VDD	VDD	—	PWR	—	正电源
VSS	VSS	—	PWR	—	负电源，接地

注：I/T：输入类型；O/T：输出类型

OPT：通过配置选项（CO）或寄存器选项来设置

PWR：电源；CO：配置选项；ST：斯密特触发输入

CMOS：CMOS 输出；NMOS：NMOS 输出

这里需要重点指出，PB0 和 PB1 并未连接到内部 10-pin MSOP 引脚上。其默认为输入状态，为了防止输入脚浮空而产生功耗，设计者可通过 PBPU 寄存器设置其具有上拉电阻功能。

**HT46R01N/HT46R02N**

引脚名称	功能	OPT	I/T	O/T	说明
PA0/AN0	PA0	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	AN0	ADCR	AN	—	A/D 通道 0
PA1/PFD/AN1	PA1	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	PFD	CTRL0	—	CMOS	PFD 输出
	AN1	ADCR	AN	—	A/D 通道 1
PA2/TC0/AN2	PA2	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	TC0	—	ST	—	外部定时器 0 时钟源输入脚
	AN2	ADCR	AN	—	A/D 通道 2
PA3/INT/AN3	PA3	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	INT	—	ST	—	外部中断输入脚
	AN3	ADCR	AN	—	A/D 通道 3
PA4/TC1/PWM	PA4	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	TC1	—	ST	—	外部定时器 1 时钟源输入脚
	PWM	CTRL0	—	CMOS	PWM 输出脚
PA5/OSC2	PA5	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	OSC2	CO	—	OSC	振荡器引脚
PA6/OSC1	PA6	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	OSC1	CO	OSC	—	振荡器输入脚
PA7/RES	PA7	PAWK	ST	NMOS	通用 I/O 口, 可通过寄存器设置唤醒功能
	RES	CO	ST	—	复位输入脚
PB0	PB0	PBPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置
PB1	PB1	PBPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置
VDD	VDD	—	PWR	—	正电源
VSS	VSS	—	PWR	—	负电源, 接地

注: I/T: 输入类型; O/T: 输出类型

OPT: 通过配置选项 (CO) 或寄存器选项来设置

PWR: 电源; CO: 配置选项; ST: 斯密特触发输入

CMOS: CMOS 输出; NMOS: NMOS 输出

**HT48R01N/HT48R02N**

引脚名称	功能	OPT	I/T	O/T	说明
PA0	PA0	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
PA1/PFD	PA1	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	PFD	CTRL0	—	CMOS	PFD 输出
PA2/TC0	PA2	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	TC0	—	ST	—	外部定时器 0 时钟源输入脚
PA3/INT	PA3	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	INT	—	ST	—	外部中断输入脚
PA4/TC1	PA4	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	TC1	—	ST	—	外部定时器 1 时钟源输入脚
PA5/OSC2	PA5	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	OSC2	CO	—	OSC	振荡器引脚
PA6/OSC1	PA6	PAPU PAWK	ST	CMOS	通用 I/O 口, 可通过寄存器设置带上拉电阻和唤醒功能
	OSC1	CO	OSC	—	振荡器输入脚
PA7/ $\overline{\text{RES}}$	PA7	PAWK	ST	NMOS	通用 I/O 口, 可通过寄存器设置唤醒功能
	$\overline{\text{RES}}$	CO	ST	—	复位输入脚
PB0	PB0	PBPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置
PB1	PB1	PBPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置
VDD	VDD	—	PWR	—	正电源
VSS	VSS	—	PWR	—	负电源, 接地

注: I/T: 输入类型; O/T: 输出类型  
OPT: 通过配置选项 (CO) 或寄存器选项来设置  
PWR: 电源; CO: 配置选项; ST: 斯密特触发输入  
CMOS: CMOS 输出; NMOS: NMOS 输出

## 极限参数

电源供应电压..... $V_{SS}-0.3V \sim V_{SS}+6.0V$	储存温度..... $-50^{\circ}\text{C} \sim 125^{\circ}\text{C}$
端口输入电压..... $V_{SS}-0.3V \sim V_{DD}+0.3V$	工作温度..... $-40^{\circ}\text{C} \sim 85^{\circ}\text{C}$
$I_{OL}$ 总电流.....100mA	$I_{OH}$ 总电流.....-100mA
总功耗.....500mW	

注: 这里只强调额定功率, 超过极限参数所规定的范围将对芯片造成损害, 无法预期芯片在上述标示范围外的工作状态, 而且若长期在标示范围外的条件下工作, 可能影响芯片的可靠性。

直流电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
V <sub>DD</sub>	工作电压	—	f <sub>SYS</sub> =4MHz	2.2	—	5.5	V
			f <sub>SYS</sub> =8MHz	3.0	—	5.5	V
			f <sub>SYS</sub> =12MHz	4.5	—	5.5	V
I <sub>DD1</sub>	工作电流 (HXT,HIRC,ERC)	3V	无负载,	—	0.8	1.2	mA
		5V	f <sub>SYS</sub> =4MHz	—	1.5	2.25	mA
I <sub>DD2</sub>	工作电流 (HXT,HIRC,ERC)	3V	无负载,	—	1.4	2.1	mA
		5V	f <sub>SYS</sub> =8MHz	—	2.8	4.2	mA
I <sub>DD3</sub>	工作电流 (HXT,HIRC,ERC)	5V	无负载, f <sub>SYS</sub> =12MHz	—	4	6	mA
I <sub>DD4</sub>	工作电流 (HIRC+LXT,低速模式)	3V	无负载, f <sub>SYS</sub> =32768Hz	—	5	10	μA
		5V	(LVR 除能, LXTLP=1)	—	12	24	μA
I <sub>STB1</sub>	静态电流 (LIRC on,LXT off)	3V	无负载, HALT	—	—	5	μA
		5V		—	—	10	μA
I <sub>STB2</sub>	静态电流 (LIRC off,LXT off)	3V	无负载, HALT	—	—	1	μA
		5V		—	—	2	μA
I <sub>STB3</sub>	静态电流 (LIRC off,LXT on)	3V	无负载, HALT LXTLP=1	—	—	5	μA
		5V		—	—	10	μA
V <sub>IL1</sub>	输入/输出,TCn 和 INT 脚的低电平输入电压	—	—	0	—	0.3V <sub>DD</sub>	V
V <sub>IH1</sub>	输入/输出,TCn 和 INT 脚的高电平输入电压	—	—	0.7V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	低电平输入电压(RES)	—	—	0	—	0.4V <sub>DD</sub>	V
V <sub>IH2</sub>	高电平输入电压(RES)	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>LVR1</sub>	低电压复位电压 1	—	V <sub>LVR</sub> =4.2V	3.98	4.2	4.42	V
V <sub>LVR2</sub>	低电压复位电压 2	—	V <sub>LVR</sub> =3.15V	2.98	3.15	3.32	V
V <sub>LVR3</sub>	低电压复位电压 3	—	V <sub>LVR</sub> =2.1V	1.98	2.1	2.22	V
I <sub>OL1</sub>	输入/输出口灌电流	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	4	8	—	mA
		5V		10	20	—	mA
I <sub>OH</sub>	输入/输出口源电流	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-2	-4	—	mA
		5V		-5	-10	—	mA
I <sub>OL2</sub>	PA7 灌电流	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	0.8	1.2	—	mA
		5V		2.0	3.0	—	mA
R <sub>PH</sub>	上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

注：静态电流(I<sub>STB1</sub>~I<sub>STB3</sub>)和 I<sub>DD4</sub>是在所有 I/O 作为输入脚且上拉到 VDD 时的测量结果。

交流电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
f <sub>sys</sub>	系统时钟	—	2.2V~5.5V	32	—	4000	kHz
			3.0V~5.5V	32	—	8000	kHz
			4.5V~5.5V	32	—	12000	kHz
f <sub>HIRC</sub>	系统时钟(HIRC)	3V/5V	Ta=25°C	-2%	4	+2%	MHz
		3V/5V	Ta=25°C	-2%	8	+2%	MHz
		5V	Ta=25°C	-2%	12	+2%	MHz
		3V/5V	Ta=0~70°C	-5%	4	+5%	MHz
		3V/5V	Ta=0~70°C	-5%	8	+5%	MHz
		5V	Ta=0~70°C	-5%	12	+5%	MHz
		2.2V~3.6V	Ta=0~70°C	-8%	4	+8%	MHz
		3.0V~5.5V	Ta=0~70°C	-8%	4	+8%	MHz
		3.0V~5.5V	Ta=0~70°C	-8%	8	+8%	MHz
		4.5V~5.5V	Ta=0~70°C	-8%	12	+8%	MHz
		2.2V~3.6V	Ta=-40°C~85°C	-12%	4	+12%	MHz
		3.0V~5.5V	Ta=-40°C~85°C	-12%	4	+12%	MHz
		3.0V~5.5V	Ta=-40°C~85°C	-12%	8	+12%	MHz
4.5V~5.5V	Ta=-40°C~85°C	-12%	12	+12%	MHz		
f <sub>ERC</sub>	系统时钟(ERC)	5V	Ta=25°C R=120kΩ*	-2%	4	+2%	MHz
		5V	Ta=0°C~70°C, R=120kΩ*	-5%	4	+5%	MHz
		5V	Ta=-40°C~85°C R=120kΩ*	-7%	4	+7%	MHz
		2.2V~5.5V	Ta=-40°C~85°C R=120kΩ*	-11%	4	+11%	MHz
f <sub>LXT</sub>	系统时钟(LXT)	—	—	—	32768	—	Hz
f <sub>TIMER</sub>	定时器输入频率(TCn)	—	2.2V~5.5V	0	—	4000	kHz
			3.0V~5.5V	0	—	8000	kHz
			4.5V~5.5V	0	—	12000	kHz
f <sub>LIRC</sub>	LIRC 振荡器	3V	—	5	10	15	kHz
		5V	—	6.5	13	19.5	kHz
t <sub>RES</sub>	外部复位低电平脉宽	—	—	1	—	—	μs
t <sub>SST</sub>	系统启动时间	—	HXT/LXT	—	1024	—	t <sub>sys</sub>
			ERC/IRC(通过 配置选项决定)	—	2	—	t <sub>sys</sub>
				—	1024	—	t <sub>sys</sub>
t <sub>INT</sub>	中断脉宽	—	—	1	—	—	μs
t <sub>LVR</sub>	低压复位时间	—	—	0.25	1	2	ms
t <sub>RSTD</sub>	复位延迟时间	—	—	—	100	—	ms

注: 1、t<sub>sys</sub> = 1/f<sub>sys</sub>

2、\*: 表示电阻的公差会影响外部 RC 的频率, 建议使用精密度较高的电阻。

### A/D 转换器特性

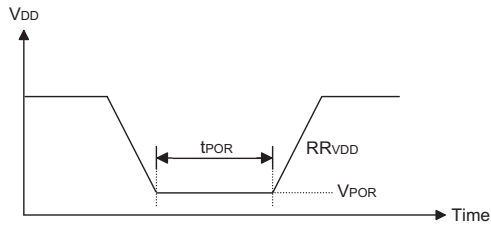
Ta=25°C

符号	参数	测试条件		最小值	典型值	最大值	单位
		V <sub>DD</sub>	条件				
DNL	A/D 非线性微分误差	3V	t <sub>AD</sub> =0.5μs	-2	—	2	LSB
		5V					
INL	A/D 非线性积分误差	3V	t <sub>AD</sub> =0.5μs	-4	—	4	LSB
		5V					
I <sub>ADC</sub>	打开 A/D 增加的功耗	3V	—	—	0.5	0.75	mA
		5V		—	1.0	1.5	mA

### 上电复位特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
V <sub>POR</sub>	上电复位电压	—	—	—	—	100	mV
RR <sub>VDD</sub>	上电复位电压速率	—	—	0.035	—	—	V/ms
t <sub>POR</sub>	V <sub>DD</sub> 保持为 V <sub>POR</sub> 的最小时间	—	—	1	—	—	ms

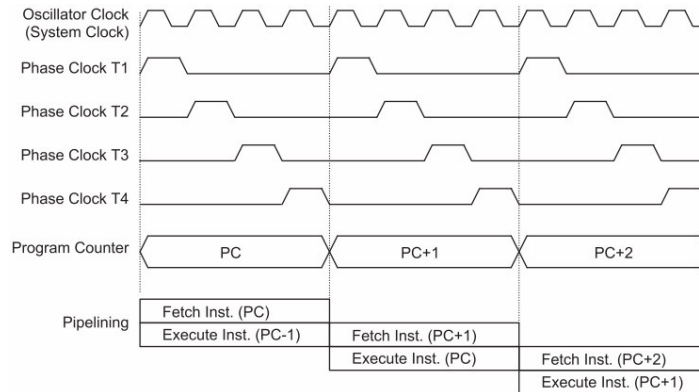


## 系统结构

内部系统结构是盛群单片机具有良好性能的主要因素。由于采用 RISC 结构，此系列单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令外，其它指令都能在一个指令周期内完成。8 位 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、递增、递减和分支等功能，而内部的数据路径则是以通过累加器和 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有最大可靠性和灵活性的 I/O 和 A/D 控制系统时，仅需要少数的外部器件。

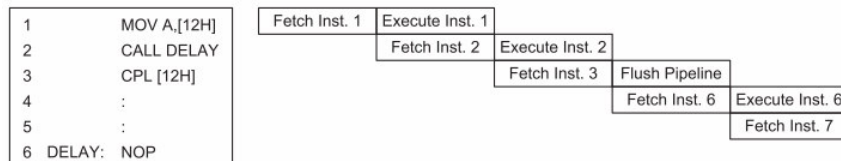
### 时序和流水线结构

主系统时钟由晶体振荡器或 RC 振荡器提供，它被细分为 T1~T4 四个内部产生的非重叠时序。在 T1 时间，程序计数器自动加一并抓取一条新的指令。剩下的时间 T2~T4 完成译码和执行功能，因此，一个 T1~T4 时钟周期构成一个指令周期。虽然指令的抓取和执行发生在连续的指令周期，但单片机流水线结构会保证指令在一个指令周期内被有效执行。除非程序计数器的内容被改变，如子程序的调用或跳转，在这种情况下指令将需要多一个指令周期的时间去执行。



系统时序和流水线

如果指令牵涉到分支，例如跳转或调用等指令，则需要两个指令周期才能完成指令执行。需要一个额外周期的原因是程序先用一个周期取出实际要跳转或调用的地址，再用另一个周期去实际执行分支动作，因此用户需要特别考虑额外周期的问题，尤其是在执行时间要求较严格的时候。



指令捕捉

## 程序计数器

在程序执行期间，程序计数器用来指向下一个要执行的指令地址。除了“JMP”和“CALL”指令需要跳转到一个非连续的程序存储器地址之外，它会在每条指令执行完成以后自动加一。选择不同型号的单片机，程序寄存器的宽度会因程序存储器的容量的不同而不同。只有较低的 8 位，即所谓的程序计数器低字节寄存器 PCL，可以被用户直接读写。

当执行的指令要求跳转到不连续的地址时，如跳转指令、子程序调用、中断或复位等，单片机通过加载所需要的位址到程序寄存器来控制程序，对于条件跳转指令，一旦条件符合，在当前指令执行时取得的下一条指令将会被舍弃，而由一个空指令周期来取代。

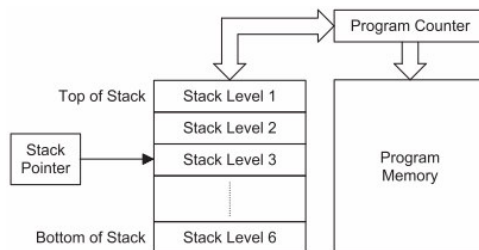
单片机型号	程序计数器	
	程序计数器高字节	PCL 寄存器
HT46R01B/01N	PC9~PC8	PCL7~PCL0
HT48R01B/01N		
HT46R02B/02N	PC10~PC8	PCL7~PCL0
HT48R02B/02N		

### 程序计数器

程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。PCL 的使用可能引起程序跳转，因此需要额外的指令周期。PCL 更多信息请见特殊功能寄存器章节。

### 堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。各单片机有不同的堆栈层数，堆栈既不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针 (SP) 加以指示，同样也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入到堆栈中。当子程序或中断响应结束时，返回指令 (RET 或 RETI) 使程序计数器从堆栈中重新得到它以前的值。当一个芯片复位后，堆栈指针将指向堆栈顶部。



如果堆栈已满，且有非屏蔽的中断发生，中断请求标志会被置位，但中断响应将被禁止。当堆栈指针减少 (执行 RET 或 RETI)，中断将被响应。这个特性提供程序设计者简单的方法来预防堆栈溢出。然而即使堆栈已满，CALL 指令仍然可以被执行，而造成堆栈溢出。使用时应避免堆栈溢出的情况发生，因为这可能导致不可预期的程序分支指令执行错误。

### 算术逻辑单元 — ALU

算术逻辑单元是单片机中很重要的部分，执行指令集中的算术和逻辑运算。ALU 连接到单片机的数据总线，在接收相关的指令码后执行需要的算术与逻辑操作，并将结果存储在指定的寄存器，当 ALU 计算或操作时，可能导致进位、借位或其它状态的变化，而相关的状态寄存器会因此更新内容以显示这些改变，ALU 所提供的功能如下：

- 算术运算：ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- 逻辑运算：AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA

- 移位运算: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- 递增和递减: INCA, INC, DECA, DEC
- 分支判断: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

## 程序存储器

程序存储器用来存放用户代码即存储程序。此系列的单片机提供一次可编程的存储器(OTP)，用户可将应用程序写入到芯片一次。OTP 型单片机提供用户以灵活的方式自由开发他们的应用，这对于需要除错或者需要经常升级和改变程序的产品是很有帮助的。

### 结构

程序存储器的容量为  $2K \times 15$  位，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。

### 特殊向量

程序存储器中某些地址保留用作诸如复位和中断的入口等特殊用途。

#### • 复位向量

该向量是保留用作单片机复位后的程序起始地址。在芯片初始化后，程序将会跳转到这个地址并开始执行。

#### • 外部中断向量

该向量为外部中断服务程序使用。当外部中断引脚发生边沿跳变时，如果中断允许且堆栈未滿，则程序会跳转到该地址开始执行。外部中断有效边沿转换类型由 CTRL1 寄存器指定设定是高到低，还是低到高有效或者两者都可以触发。

#### • 定时/计数器 0/1 中断向量

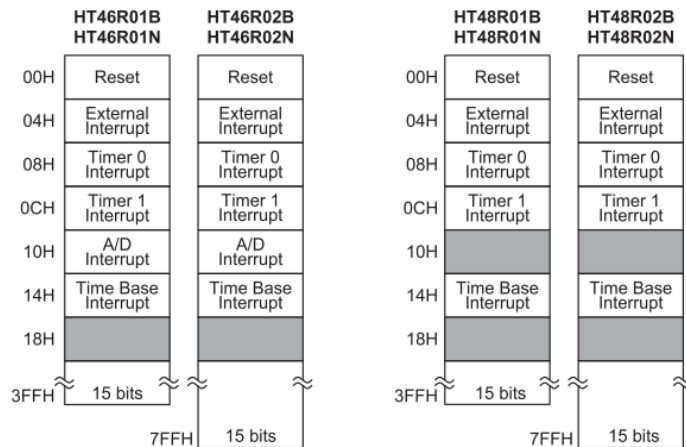
该内部中断向量为定时/计数器使用，当定时/计数器发生溢出，如果中断允许且堆栈未滿，则程序会跳转到相应的地址开始执行。

#### • A/D 转换中断向量

该内部向量为 A/D 转换器使用。当 A/D 转换完成时，如果中断允许且堆栈未滿，则程序将跳转到相应地址开始执行。

#### • 时基中断向量

该内部向量为时基中断使用，当时基溢出发生，如果中断允许且堆栈未滿，则程序将跳转到相应地址开始执行。



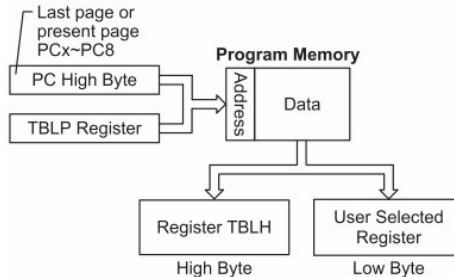
程序存储器结构

## 查表

程序存储器中的任何地址都可以定义成一个表格，以便储存固定的数据。使用表格时，表格指针必须先设定，其方式是将表格的低位地址放在表格指针寄存器 TBLP 中。这个寄存器定义的是表格较低的 8 位地址。

在设定完表格指针后，表格数据可以使用“TABRDC [m]”或“TABRDL [m]”指令分别从程序存储器查表读取。当这些指令执行时，程序存储器中表格数据低字节，将被传送到使用者所指定的数据存储器[m]，程序存储器中表格数据的高字节，则被传送到 TBLH 特殊寄存器，而高字节中未使用的位将被读取为“0”。

下图是查表中寻址/数据流程：



## 查表范例

以下范例说明表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器的最后一页。表格指针的初始值设为“06H”，这可保证从数据表格读取的第一笔数据位于程序存储器地址“306H”，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRDC [m]”指令被使用，则表格指针指向当前页。在这个例子中，表格数据的高字节等于零，而当“TABRDL [m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

由于 TBLH 寄存器为只读寄存器，不能重新储存，若主程序和中断服务程序都使用表格读取指令，应该注意它的保护。使用表格读取指令，中断服务程序可能会改变 TBLH 的值，若随后在主程序中再次使用这个值，则会发生错误，因此建议避免同时使用表格读取指令。然而在某些情况下，如果同时使用表格读取指令是不可避免的，则在执行任何主程序的表格读取指令前，中断应该先除能，另外要注意的是所有与表格相关的指令，都需要两个指令周期去完成操作。

指令	表格地址										
	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC[m]	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

表格地址

注： @7~@0：表格指针TBLP 位

对HT46R01B/HT48R01B/HT46R01N/HT48R01N来说，当前程序计数器位，从PC9~PC8。

对HT46R02B/HT48R02B/HT46R02N/HT48R02N来说，当前程序计数器位，从PC10~PC8。

### • 表格读取程序举例-1K ROM

```

tempreg1 db ? ;temporary register #1
tempreg2 db ? ;temporary register #2
:
:
mov a,06h ;initialise low table pointer - note that this address
mov tblp,a ;is referenced
:
:
tabrdl tempreg1 ;transfers value in table referenced by table pointer data at program
;memory address "306H" transferred to tempreg1 and TBLH

```

```

dec tblp          ; reduce value of table pointer by one
tabrdl tempreg2  ; transfers value in table referenced by table pointer data at program
                  ; memory address "305H" transferred to tempreg2 and TBLH in this
                  ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                  ; register tempreg2
:
:
:
org 300h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
:

```

## 数据存储器的

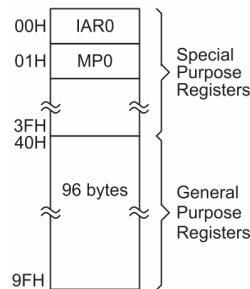
数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

### 结构

数据存储器分为两个区，第一部分是特殊功能数据存储器。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部分数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。

数据存储器的两个部分，即特殊和通用数据存储器，位于连续的地址。全部 RAM 为 8 位宽度，而数据存储器长度因所选择的单片机型号的不同而不同。所有单片机的数据存储器的开始地址都是“00H”。

所有单片机的程序需要一个读/写的存储区，让临时数据可以被储存和再使用。该RAM区域就是通用数据存储器。这个数据存储器区可让用户进行读取和写入操作。使用“SET[m].i”和“CLR[m].i”指令可对个别位进行设置或复位的操作，方便用户在数据存储器中进行位操作。



**数据存储器结构**

注：使用“SET[m].i”和“CLR[m].i”可对大多数的数据存储器区进行位操作，少数专用位除外。数据存储器区也可以通过存储器指针间接寻址。

### 特殊数据存储器

这个区域的数据存储器是存放特殊寄存器，它和单片机的正确操作密切相关。大多数寄存器是可以读取和写入，只有一些是被写保护而只可读取的，相关的介绍请参考特殊功能寄存器的部分。需注意，任何读取指令对于未定义的地址读取将返回“00H”的值。

	HT46R01B HT46R01N	HT46R02B HT46R02N	HT48R01B HT48R01N	HT48R02B HT48R02N
00H	IAR0	IAR0	IAR0	IAR0
01H	MP0	MP0	MP0	MP0
02H	IAR1	IAR1	IAR1	IAR1
03H	MP1	MP1	MP1	MP1
04H				
05H	ACC	ACC	ACC	ACC
06H	PCL	PCL	PCL	PCL
07H	TBLP	TBLP	TBLP	TBLP
08H	TBLH	TBLH	TBLH	TBLH
09H	WDTs	WDTs	WDTs	WDTs
0AH	STATUS	STATUS	STATUS	STATUS
0BH	INTC0	INTC0	INTC0	INTC0
0CH	TMR0	TMR0	TMR0	TMR0
0DH	TMR0C	TMR0C	TMR0C	TMR0C
0EH	TMR1	TMR1	TMR1	TMR1
0FH	TMR1C	TMR1C	TMR1C	TMR1C
10H	PA	PA	PA	PA
11H	PAC	PAC	PAC	PAC
12H	PAPU	PAPU	PAPU	PAPU
13H	PAWK	PAWK	PAWK	PAWK
14H	PB	PB	PB	PB
15H	PBC	PBC	PBC	PBC
16H	PBPU	PBPU	PBPU	PBPU
17H				
18H				
19H				
1AH	CTRL0	CTRL0	CTRL0	CTRL0
1BH	CTRL1	CTRL1	CTRL1	CTRL1
1CH				
1DH				
1EH	INTC1	INTC1	INTC1	INTC1
1FH	PWM0	PWM0		
20H	ADRL	ADRL		
21H	ADRH	ADRH		
22H	ADCR	ADCR		
23H	ACSR	ACSR		
24H				
25H				
...				
3FH				

: Unused, read as "00"

**特殊数据存储**

## 特殊功能寄存器

为了确保单片机能正常工作，数据存储器中设置了一些内部寄存器。这些寄存器确保内部功能（定时器，中断等）和外部功能（输入/输出口数据控制）的正确工作。在数据存储器中，这些寄存器的开始地址为“00H”，且被映射到 Bank0 中。特殊功能寄存器和通用数据存储器起始地址之间，有一些未定义的数据存储器，被保留用来做未来扩充，若从这些地址读取数据将会返回“00H”值。

### 间接寻址寄存器 — IAR0, IAR1

间接寻址寄存器 IAR0 和 IAR1 的地址虽位于数据存储区，但其并没有实际的物理地址。间接寻址的方法准许使用间接寻址指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器 IAR0 和 IAR1 上的任何动作，将对间接寻址指针 MP0 和 MP1 所指定的存储器地址产生对应的读/写操作。它们总是成对出现，可以一起访问数据存储器中的数据。因为这些间接寻址寄存器不是实际存在的，直接读取将返回“00H”的结果，而直接写入此寄存器则不做任何操作。

### 间接寻址指针 — MP0, MP1

该系列单片机提供两个间接寻址指针，即 MP0 和 MP1。由于这些指针在数据存储器中能像普通的寄存器一般被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由间接寻址指针所指定的地址。

以下例子说明如何清除一个具有 4 RAM 地址的区块，它们已事先定义成地址 adres1 到 adres4。

#### • 间接寻址程序举例

```
data .section `data`
adres1          db ?
adres2          db ?
adres3          db ?
adres4          db ?
block          db ?
code .section at 0 `code`
org             00h
start:
    mov a, 04h          ; setup size of block
    mov block, a
    mov a, offset adres1 ; Accumulator loaded with first RAM address
    mov mp0, a         ; setup memory pointer with first RAM address

loop:
    clr IAR0          ; clear the data at address defined by mp0
    inc mp0           ; increment memory pointer
    sdz block         ; check if last memory location has been cleared
    jmp loop

continue:
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

### 累加器—ACC

对任何单片机来说，累加器是相当重要的，且与 ALU 所完成的运算有密切关系，所有 ALU 得到的运算结果都会暂时存在 ACC 累加器里。若没有累加器，ALU 必须在每次进行如加法、减法和移位的运算时，将结果写入到数据存储器，这样会造成程序编写和时间的负担。另外数据传送也常常牵涉到累加器的临时储存功能，例如在使用者定义的一个寄存器和另一个寄存器之间传送数据时，由于两寄存器之间不能直接传送数据，因此必须通过累加器来传送数据。

### 程序计数器低字节寄存器—PCL

为了提供额外的程序控制功能，程序计数器低字节设置在数据存储器的特殊功能区域内，程序员可对此寄存器进行操作，很容易的直接跳转到其它程序地址。直接给 PCL 寄存器赋值将导致程序直接跳转到程序存储器的某一地址，然而由于寄存器只有 8 位长度，因此只允许在本页的程序存储器范围内进行跳转，而当使用这种运算时，要注意会插入一个空指令周期。

### 状态寄存器—STATUS

这8位寄存器包括零标志位 (Z)、进位标志位 (C)、辅助进位标志位 (AC)、溢出标志位 (OV)，暂停标志位 (PDF)、和看门狗溢出标志位 (TO)。这些标志位同时记录单片机的状态数据和算术/逻辑运算。

除了TO和PDF标志位以外，状态寄存器的其它位像其它大多数寄存器一样可以被改变。但是任何数据写入状态寄存器将不会改变TO和PDF标志位。另外，执行不同指令操作后，与状态寄存器相关的运算将会得到不同的结果。TO标志位只会受系统上电、看门狗溢出、或执行“CLR WDT”或者“HALT”指令的影响。PDF指令只会受执行“HALT”或“CLR WDT”指令或系统上电的影响。

Z、OV、AC和C标志位通常反映最近的运算操作的状态

另外，当进入一个中断程序或者执行子程序调用时状态寄存器将不会自动压入到堆栈中保存。假如状态寄存器的内容很重要，且中断子程序会改变状态寄存器的内容，则需要保存备份以备恢复。值得注意的是，状态寄存器的0~3位可以读取和写入。

#### • STATUS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	×	×	×	×

“×”为未知

Bit 7~6 未使用，读为“0”

Bit 5 **TO**: 看门狗溢出标志位

0: 系统上电或执行“CLR WDT”或“HALT”指令后

1: 看门狗溢出发生

Bit 4 **PDF**: 暂停标志位

0: 系统上电或执行“CLR WDT”指令后

1: 执行“HALT”指令

Bit 3 **OV**: 溢出标志位

0: 无溢出

1: 运算结果高两位的进位状态异或结果为 1

Bit 2 **Z**: 零标志位

0: 算术或逻辑运算结果不为 0

1: 算术或逻辑运算结果为 0

Bit 1 **AC**: 辅助进位标志位

0: 无辅助进位

1: 在加法运算中低四位产生了向高四位进位，或减法运算中低四位不发生从高四位借位

Bit 0 **C**: 进位标志位

0: 无进位

1: 如果在加法运算中结果产生了进位，或在减法运算中结果不发生借位

**C** 也受循环移位指令的影响。

### 输入/输出和控制寄存器

在特殊功能寄存器中，输入/输出寄存器(PA、PB等)和相关的控制寄存器(PAC、PBC等)是很重要的，这些寄存器在数据存储器有特定的地址。输入/输出寄存器用来传送端口上的输入和输出数据，而这些控制寄存器是用来设置引脚的状态，以决定是输出口还是输入口。若要设定一个引脚为输入，需将控制寄存器相应的位设置为高，若引脚设定为输出，则控制寄存器相应的位设置为低。程序初始化期间，在从输入/输出读取或写入数据之前，必须先设置好控制寄存器以确定引脚的输出状态。使用SET[m].i 和CLR[m].i 可以对寄存器单独的位进行灵活设置。这种在程序中，通过改变输入/输出状态控制寄存器中的某一位而直接改变端口输入/输出状态的能力是此系列单片机十分有用的特性。

### 系统控制寄存器-CTRL0,CTRL1

这些寄存器是用来控制各种内部功能，如PFD控制、PWM控制、系统时钟选择、LXT低功耗控制，外部中断边沿触发类型、看门狗定时器使能、时基分频和LXT振荡器使能控制。

#### • CTRL0 寄存器-HT46R01B/HT46R02B/HT46R01N/HT46R02N

Bit	7	6	5	4	3	2	1	0
Name	—	PFDCS	PWMSEL	—	PWMC0	PFDC	LXTLP	CLKMOD
R/W	—	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	—	0	0	—	0	0	0	0

Bit 7 未使用，读为“0”

Bit 6 **PFDCS**: PFD 时钟源选择位  
 0: timer0  
 1: timer1

Bit 5 **PWMSEL**: PWM 模式选择位  
 0: 6+2 模式  
 1: 7+1 模式

Bit 4 未使用，读为“0”

Bit 3 **PWMC0**: I/O 或 PWM 控制位  
 0: I/O  
 1: PWM 输出

Bit 2 **PFDC**: I/O 或 PFD 控制位  
 0: I/O  
 1: PFD 输出

Bit 1 **LXTLP**: LXT 振荡器低功耗控制功能位  
 0: LXT 振荡器快速启动模式  
 1: LXT 振荡器低功耗模式

Bit 0 **CLKMOD**: 系统时钟模式选择位  
 0: 高速模式—使用 HIRC 作为系统时钟  
 1: 低速模式—使用 LXT 作为系统时钟，HIRC 时钟停止  
 这些设置只有在配置选项设置为 HIRC+LXT 时有效。

注意: 如果PWMC0位选择共用引脚用于PWM输出，则 $f_{TP}$ 来自于系统时钟。  
 ( $f_{TP}$ 可作为定时/计数器0，时基和PWM的时钟源)。

#### • CTRL0 寄存器-HT48R01B/HT48R02B/HT48R01N/HT48R02N

Bit	7	6	5	4	3	2	1	0
Name	—	PFDCS	—	—	—	PFDC	LXTLP	CLKMOD
R/W	—	R/W	—	—	—	R/W	R/W	R/W
POR	—	0	—	—	—	0	0	0

Bit 7 未使用，读为“0”

Bit 6 **PFDCS**: PFD 时钟源选择位  
 0: timer0  
 1: timer1

Bit 5~3 未使用，读为“0”

- Bit 2 **PFDC**: I/O 或 PFD 控制位  
 0: I/O  
 1: PFD 输出
- Bit 1 **LXTLP**: LXT 振荡器低功耗控制功能位  
 0: LXT 振荡器快速启动模式  
 1: LXT 振荡器低功耗模式
- Bit 0 **CLKMOD**: 系统时钟模式选择位  
 0: 高速模式—使用 HIRC 作为系统时钟  
 1: 低速模式—使用 LXT 作为系统时钟, HIRC 时钟停止  
 这些设置只有在配置选项设置为 HIRC+LXT 时有效。

• **CTRL1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	INTEG1	INTEG0	TBSEL1	TBSEL0	WDTEN3	WDTEN2	WDTEN1	WDTEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	1	0	1	0

- Bit 7~6 **INTEG1,INTEG0**: 外部中断边沿触发类型  
 00: 除能  
 01: 上升沿触发  
 10: 下降沿触发  
 11: 双沿触发
- Bit 5~4 **TBSEL1, TBSEL0**: 时基周期选择位  
 00:  $2^{10} \times (1/f_{TP})$   
 01:  $2^{11} \times (1/f_{TP})$   
 10:  $2^{12} \times (1/f_{TP})$   
 11:  $2^{13} \times (1/f_{TP})$
- Bit 3~0 **WDTEN3, WDTEN2, WDTEN1, WDTEN0**: WDT 功能使能  
 1010: WDT 关闭  
 其它值: WDT 使能, 建议值为 0101  
 如果“watchdog timer enable”配置选项被选中, 那么看门狗时钟将总是开启的, 且 WDTEN3~WDTEN0 控制位无效。
- 注意: 只有当 WDT 配置选项是除能的, 并且位 WDTEN3~WDTEN0 = 1010 时, WDT 才能关闭。而只要 WDT 配置选项是使能的, 或者位 WDTEN3~WDTEN0 ≠ 1010 时, WDT 就开启。

**唤醒功能寄存器-PAWK**

当单片机进入休眠模式以后, 多种方式可唤醒单片机, 使其继续正常运行。其中一种方式是通过有唤醒功能的I/O口的下降沿唤醒, 而这个控制寄存器就是用来设置PA口是否具有唤醒功能。

**上拉电阻寄存器-PAPU,PBPU**

当I/O口设置为输入, 则可以设置使用内部连接上拉电阻, 从而省去外接上拉电阻。这些寄存器即用来选择I/O引脚是否连接到内部上拉电阻。

## 振荡器

不同的振荡器选择可以让使用者在不同的应用需求中实现更大范围的功能。振荡器的灵活性使得在速度和功耗方面可以达到最优化。振荡器选择是通过配置选项和寄存器共同完成的。

### 振荡器概述

振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。外部振荡器需要一些外围器件，而集成的两个内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。

类型	名称	频率
外部晶振	HXT	400kHz~12MHz
外部 RC	ERC	400kHz~12MHz
内部高速 RC	HIRC	4, 8 或 12MHz
外部低速晶振	LXT	32768Hz
内部低速 RC	LIRC	13kHz

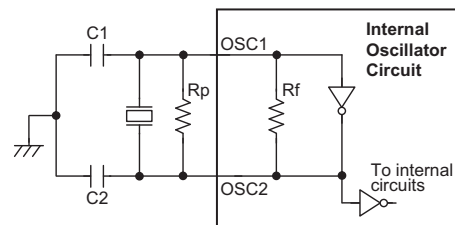
振荡器类型

### 系统时钟配置

此系列的单片机有五个系统振荡器，包括三个高速振荡器和两个低速振荡器。高速振荡器有外部晶体/陶瓷振荡器-HXT，外部 RC 振荡器-ERC 和内部 RC 振荡器-HIRC。两个低速振荡器包括外部 32768Hz 振荡器-LXT 和内部 13kHz ( $V_{DD}=5V$ ) 振荡器-LIRC。

#### 外部晶体/陶瓷振荡器 -- HXT

对于晶体振荡器，只要简单地将晶体连接至 OSC1 和 OSC2，则会产生振荡所需的相移及反馈，而不需其它外部的器件。为保证某些低频率的晶体振荡和陶瓷谐振器的振荡频率更精准，建议连接两个小容量电容 C1 和 C2 到 VSS，具体数值与客户选择的晶体/陶瓷晶振有关。



Note: 1. Rp is normally not required. C1 and C2 are required.  
 2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

#### 晶体/陶瓷振荡器 -- HXT

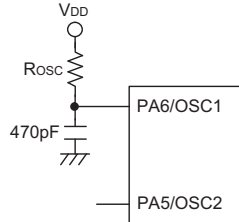
晶体振荡器 C1 和 C2 值		
晶体频率	C1	C2
12MHz	8pF	10pF
8 MHz	8pF	10pF
4 MHz	8pF	10pF
1 MHz	100pF	100pF
注：C1 和 C2 数值仅作参考用		

晶体振荡器电容推荐值

#### 外部 RC 振荡器 -- ERC

使用外部 RC 电路作为系统振荡器，只需要在 OSC1 和 VDD 之间连接一个阻值约在 24kΩ 到 1.5MΩ 之间的电阻，OSC1 与 VSS 之间连接一个电容。系统频率由外部所接电阻的大小决定，外部

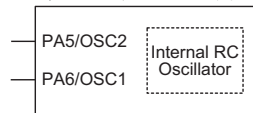
电容并不会影响振荡器的频率值，在这里只起到稳定的目的。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 VDD、温度以及芯片制成工艺不同的影响减至最低程度。这里，提供一个电阻/频率的参考：使用外部 120K 电阻连接到 5V 电源电压，在 25°C 下，振荡器的频率为 4MHz，容差 2%。外部 RC 振荡器仅使用 OSC1 引脚，OSC1 与 PA6 引脚共用，此时 PA5/OSC2 引脚可以作为普通的 I/O 口使用。



**外部 RC 振荡器 -- ERC**

**内部 RC 振荡器 -- HIRC**

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有三种固定的频率：4MHz，8MHz，12MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 VDD、温度以及芯片制成工艺不同的影响减至最低程度。在电源电压为 3V 或者 5V 及温度为 25°C 的条件下，4MHz，8MHz，12MHz 这三个固定频率的容差为 2%。注意的是，如果选择了内部 RC 时钟，无需额外的引脚接其它元件；此时 PA5 和 PA6 可以作为通用 I/O 口使用。

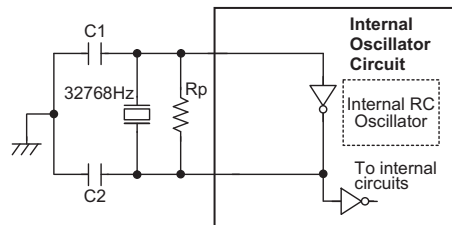


Note: PA5/PA6 used as normal I/Os

**内部 RC 振荡器-HIRC**

**外部 32768Hz 晶体振荡器 -- LXT**

当系统进入休眠模式，系统时钟关闭以降低功耗。然而在某些应用需要在省电模式下保持定时/计数功能能继续运行，系统需要提供额外的系统时钟。配置选项提供了高速时钟和低速时钟搭配使用的选项，LXT 振荡器即用来提供这样的时钟。LXT 振荡器由 32768Hz 晶振接到 OSC1 和 OSC2 引脚。为了保证 LXT 起振和频率的精确，建议使用两个小容量电容 C1 和 C2，具体数值与客户选择的 32768Hz 晶振有关。另外，外部并联的反馈电阻 R<sub>p</sub> 也是必需的。LXT 振荡器需要和 HIRC 振荡器搭配使用。



Note: 1. R<sub>p</sub>, C1 and C2 are required.  
 2. Although not shown pins have a parasitic capacitance of around 7pF.

**外部 LXT 振荡器**

LXT 振荡器 C1 和 C2 值		
晶体频率	C1	C2
32768kHz	8pF	10pF
注： 1、C1 和 C2 数值仅作参考用 2、R <sub>p</sub> 的建议值为 5M~10MΩ		

**32768Hz 振荡器电容推荐值**

### LXT 振荡器低功耗功能

LXT 振荡器有两种模式：快速启动模式和低功耗模式。可以使用寄存器 CTRL0 中的 LXTLP 位来选择模式。

LXTLP 位	LXT 模式
0	快速启动
1	低功耗

系统上电后，LXTLP 会自动清零以确保 LXT 振荡器工作在快速启动模式。在快速启动模式中，LXT 振荡器将起振并且快速的稳定下来。在 LXT 振荡器完全起振后，可设置 LXTLP 位为 1，LXT 振荡器将进入低功耗模式。振荡器将继续运行，但功耗降低，振荡电路只有在 LXT 起振时需要较大的电流。在电池及低功耗应用中，建议在上电两秒后再设置 LXTLP 位为 1，以保证最小的功耗。

注意，无论 LXTLP 位设置为何值，LXT 振荡器都能保持正常工作，不同之处在于，低功耗模式下启动需要更长的时间。

### 内部 13kHz 振荡器 -- LIRC

LIRC 是一个完全独立运行的片内 RC 振荡器，无需外部器件，在常温 5V 条件下，振荡频率值为 13kHz。当单片机进入休眠模式，系统时钟将停止运行。但 WDT 振荡器会继续运行以保持 WDT 的功能。然而，在某些需要节省功耗的应用中，可通过配置选项来关闭 LIRC 以降低功耗。

## 工作模式

使用 LXT 低速振荡器和一个高速振荡器，系统可以工作在下面几个不同的模式：正常模式、低速模式和休眠模式。

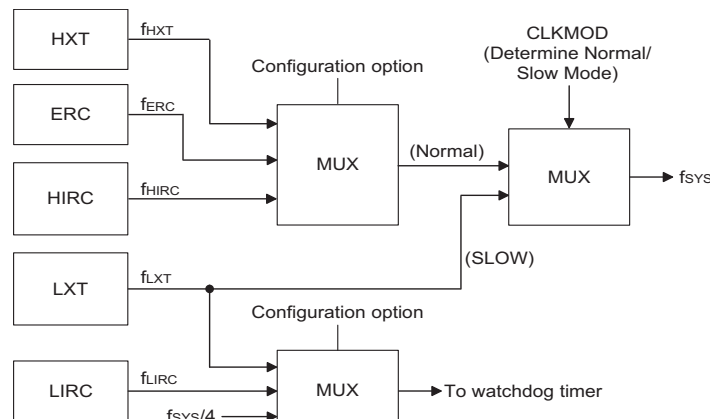
### 模式的类型与选择

使用较高频率的振荡器将获得较高的性能，但是功耗也较大，使用较低频率的振荡器则刚好相反。若单片机能够动态的切换高速和低速振荡器，则单片机就具有足够的灵活性来优化性能/功耗比，该特性对于低功耗的应用特别重要。

单片机如果使用了 LXT 振荡器，则需要使用内部 RC 振荡器 HIRC 作为高速振荡器。如果使用 HXT 或 ERC 作为高速系统时钟，共用该引脚被占用，因此不能外接 LXT 振荡器。寄存器 CTRL0 中的 CLKMOD 位用来切换系统时钟从高速 HIRC 振荡器到低速 LXT 振荡器。当执行 HALT 指令后，单片机进入休眠模式，LXT 振荡器将继续运行。LXT 晶振连接到 OSC1/OSC2 脚，且 LXT 将一直运行。

注意，只有在 HIRC+LXT 振荡器配置下，对 CLKMOD 的操作才是有效的。

当系统进入休眠模式时，高频系统时钟将停止工作。下表显示了 CLKMOD 位、HALT 指令和高/低频振荡器之间的关系。CLKMOD 位用来选择正常模式或低速模式。



系统时钟配置

工作模式	OSC1/OSC2 配置				
	HXT	ERC	HIRC	HIRC+LXT	
				HIRC	LXT
正常模式	运行	运行	运行	运行	运行
低速模式	—	—	—	停止	运行
休眠模式	停止	停止	停止	停止	运行

“—”表示未使用

### 工作模式控制

#### 切换模式

通过设置 CTRL0 寄存器中的 CLKMOD 位和 HALT 指令，可实现单片机工作模式之间的切换。CLKMOD 位用来设置系统时钟是高速或者低速振荡器，从而使系统工作在正常模式或者低速模式。执行 HALT 指令将强制系统进入休眠模式。HALT 指令的执行与 CLKMOD 位的设置无关。

当执行 HALT 指令，系统进入休眠模式，将发生下面的情况：

- 系统振荡器将被关闭，应用程序将停止在“HALT”指令处
- 在 RAM 和寄存器上的内容保持不变
- 如果 WDT 时钟源是来自 LIRC 振荡器或 LXT 振荡器，则 WDT 将被清除然后再重新计数；若来源于系统时钟，则停止计数
- 所有输入/输出端口状态保持不变
- STATUS 寄存器中，PDF 标志位被置位，TO 标志位被清零

#### 静态电流注意事项

要使系统静态电流降到最小，为微安级，除了需要单片机进入休眠模式，还要考虑到电路的设计。特别要注意输入/输出口的状态。所有高阻抗输入引脚需要接高电平或低电平，否则引脚浮空会造成内部振荡进而增大电流的消耗。另外还需要注意单片机输出端口上的负载，尽量减少拉电流或与其它 CMOS 输入相连。

如果配置选项使能看门狗振荡器 LIRC，当进入休眠模式，振荡器继续振荡，并继续消耗电量。在耗电敏感的应用中，使用系统时钟作为 WDT 时钟是更好的选择。如果配置 LXT 使能，当进入休眠模式时也将消耗一定的电量。置位 LXTLP(CTRL0.1)也可使 LXT 振荡功耗最小。

#### 唤醒

当系统进入休眠模式下，可以通过以下几种方式唤醒：

- 外部复位
- PA 口下降沿
- 系统中断
- WDT 溢出

若由外部 RES 引脚唤醒，系统会经过完全复位的过程。若由 WDT 溢出唤醒，则看门狗计数器将被复位清零。这两种唤醒方式都会使系统复位，可以通过状态寄存器中 TO 和 PDF 位来判断它的唤醒源。系统上电或执行清除看门狗的指令，PDF 被清零；执行 HALT 指令，PDF 将被置位。看门狗计数器溢出将会置位 TO 标志并唤醒系统，同时复位程序计数器和堆栈指针，其它标志保持原有状态。

端口 PA0~PA7 中的每个位都可以通过 PAWK 寄存器独立选择唤醒功能。PA 口唤醒后，程序将执行“HALT”指令后的其它指令。

如果系统是通过中断唤醒，则有两种情况，假如中断除能或中断使能但堆栈已满，系统唤醒后继续执行“HALT”指令的其它指令，相应的中断服务程序只有在中断使能后或堆栈空闲后被执行。假如中断使能且堆栈未满，则正常的中断响应将会发生。如果系统进入休眠模式之前外部中断请求标志位被置为“1”，则相关中断的唤醒功能无效。

无论是哪种方式唤醒，单片机从唤醒回到正常运行都需要一定的延迟时间，延时的时长请参照下面的表格。

唤醒源	振荡器类型	
	ERC, IRC	Crystal
外部RES	$t_{RSTD}+t_{SST1}$	$t_{RSTD} +t_{SST2}$
PA 口	$t_{SST1}$	$t_{SST2}$
中断		
WDT 溢出		

- 注：
1.  $t_{RSTD}$  (复位延时时间),  $t_{SYS}$  (系统时钟)
  2.  $t_{RSTD}$  为上电延时, 典型值为 100ms
  3.  $t_{SST1}=2$  或  $1024 t_{SYS}$
  4.  $t_{SST2}= 1024 t_{SYS}$

#### 唤醒延迟时间

## 看门狗定时器

看门狗定时器的功能在于防止如电磁的干扰等外部不可控制事件, 所造成的程序不正常动作或跳转到未知的地址。

### 看门狗定时器操作

当 WDT 溢出时, 会产生系统复位的动作。当 WDT 配置选项设置为除能, 则任何相关的指令操作都无效。通过配置选项和两个内部的寄存器 WDT5 和 CTRL1 可以设置不同的 WDT 选项。通过配置选项和数据存储器中特殊功能寄存器 CTRL1 的 WDTEN 位, 来使能看门狗定时器。

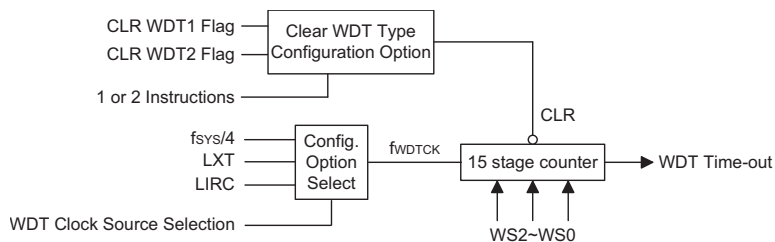
配置选项	CTRL1 寄存器	WDT 功能
除能	除能	OFF
除能	使能	ON
使能	×	ON

#### 看门狗定时器开/关控制

如果 WDT 配置选项除能且 CTRL1 寄存器中 WDTEN3~WDTEN0 位被写入 1010B, 看门狗定时器将被关闭。此时 WDTEN 中的值为系统上电时的默认值。虽然向 CTRL1 的 WDTEN3~WDTEN0 位写入其它的任何数字可开启看门狗定时器, 但为了最大程度保护它, 建议向这些位写入 0101B。

通过配置选项, 看门狗定时器可以选择三种不同的时钟源: LXT,  $f_{SYS}/4$  或 LIRC。注意, 选择  $f_{SYS}/4$  时钟作为 WDT 的时钟源, 当系统进入休眠模式时, 指令时钟会停止且 WDT 将失去其保护功能。对于干扰比较大的应用环境, 推荐使用 LIRC 振荡器或 LXT 作为 WDT 的时钟源。分频比由 WDT5 寄存器的第 0, 1 和 2 位, 即 WS0、WS1 和 WS2 位来决定。如果 WS0、WS1 和 WS2 都置 1, 分频比例为 1:128, 即可提供最大溢出周期。

系统在正常运行状态下, WDT 溢出将导致芯片复位, 并置位状态标志位 TO。但是在系统处于休眠模式时, 如果 WDT 发生溢出, 系统将从休眠中唤醒, 置位状态寄存器中的 TO, 并且它只复位程序计数器 PC 和 SP。有三种方法可以用来清除 WDT 的内容, 第一种是外部硬件复位(RES 引脚低电平), 第二种是通过软件指令, 而第三种是通过“HALT”指令。使用软件指令有两种方法去清除看门狗寄存器, 需要由配置选项选择。第一种选择是使用单一“CLR WDT”指令, 而第二种是使用“CLR WDT1”和“CLR WDT2”两个指令。对于第一种选择, 只要执行“CLR WDT”便清除 WDT。而第二种选择, 需要交替执行“CLR WDT1”和“CLR WDT2”两者才能成功的清除 WDT。关于第二种选择, 如果“CLR WDT1”正被使用来清除 WDT, 接着再执行这条指令将是无效的, 只有执行“CLR WDT2”指令才能清除 WDT。同样的“CLR WDT2”指令已经执行后, 只有接着执行“CLR WDT1”指令才可以清除看门狗定时器。



看门狗定时器

• **WDTs 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	WS2	WS1	WS0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	1	1	1

Bit 7~3: 未定义， 读为“0”

Bit 2~0: **WS2, WS1, WS0**: WDT 溢出周期选择

- 000:  $2^8 t_{WDTCK}$
- 001:  $2^9 t_{WDTCK}$
- 010:  $2^{10} t_{WDTCK}$
- 011:  $2^{11} t_{WDTCK}$
- 100:  $2^{12} t_{WDTCK}$
- 101:  $2^{13} t_{WDTCK}$
- 110:  $2^{14} t_{WDTCK}$
- 111:  $2^{15} t_{WDTCK}$

## 复位和初始化

复位功能是整个单片机中基本的部分，使得单片机可以设定一些与外部参数无关的先置条件。最重要的复位条件是在单片机首次上电以后，经过短暂的延迟，内部硬件电路使得单片机处于预期的稳定状态并开始执行第一条程序指令。上电复位以后，在程序执行之前，部分重要的内部寄存器将会被设定为预先设定的状态。程序计数器就是其中之一，它会被清除为零，使得单片机从最低的程序存储器地址开始执行程序。

除上电复位以外，即使单片机处于正常工作状态，有些情况的发生也会迫使单片机复位。譬如当单片机上电后已经开始执行程序， $\overline{RES}$ 脚被强制拉为低电平。这种复位为正常操作复位，单片机中只有一些寄存器受影响，而大部分寄存器不会改变，在复位引脚恢复至高电平后，单片机可以正常运行。

另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。另一种复位为低电压复位即 **LVR** 复位，在电源供应电压低于 **LVR** 设定值时，系统会产生 **LVR** 复位，这种复位与与 $\overline{RES}$ 脚拉低复位方式相似。

### 复位功能

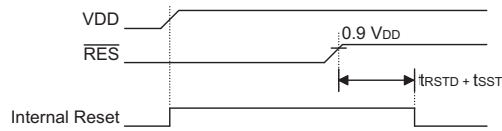
包括内部和外部事件触发复位，单片机共有五种复位方式：

• **上电复位**

这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器从开始地址执行，上电复位也使得其它寄存器被设定在预设条件。所有的输入/输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设定为输入状态。

虽然单片机有一个内部 **RC** 复位功能，如果电源上升缓慢或者上电时电源不稳定，可能导致芯片复位不良，所以推荐使用和 $\overline{RES}$ 引脚连接的外部 **RC** 电路。由 **RC** 电路所造成的时间延迟使得 $\overline{RES}$ 引脚在电源供应稳定前的一段延长周期内保持在低电平。在这段时间内，单片机的正常操作是被禁止的。 $\overline{RES}$ 引脚达到一定电压值后，再经过延迟时间  $t_{RSTD}$  单片机可以开始进行正常操作。下图中 **SST**

是系统延迟周期 System Start-up Timer 的缩写。

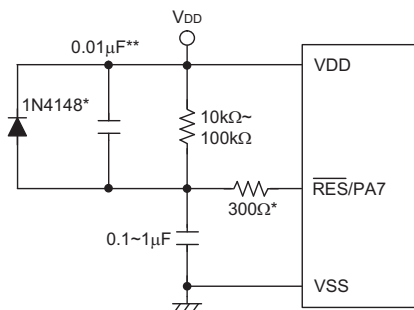


注： $t_{RSTD}$ 为上电延迟时间，典型值为100ms

**上电复位时序图**

在许多应用场合，可以在  $\overline{RES}$  和  $\overline{VDD}$  之间接入一个电阻，在  $\overline{VSS}$  与  $\overline{RES}$  之间接入一个电容作为外部复位电路。与  $\overline{RES}$  脚上所有相连接的线段必须尽量短以减少噪声干扰。

当系统在较强干扰的场合工作时，建议使用增强型的复位电路，如下图所示。



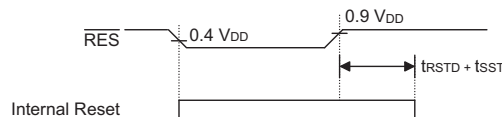
注：“\*”表示建议加上此元件以加强静电保护。  
 “\*\*”表示建议在电源有较强干扰场合加上此元件。

**外部RES电路**

欲知有关外部复位电路的更多信息可参考 HOLTEK 网站上的应用范例 HA0075S。

•  $\overline{RES}$  引脚复位

当单片机正常工作时， $\overline{RES}$  引脚通过外部硬件（如外部开关）强迫拉至低电平时，此种复位形式即会发生。这种复位方式和其它的复位方式一样，程序计数器会被清除为零且程序从头开始执行。

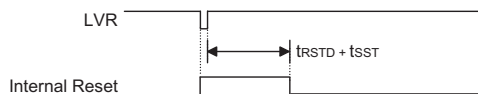


注： $t_{RSTD}$ 为上电延迟时间，典型值为100ms。

**RES复位时序图**

• 低电压复位-- LVR

单片机具有低电压复位电路，用来监测它的电源电压，可通过配置选项进行选择。例如在更换电池的情况下，单片机供应的电压可能会落在  $0.9V \sim V_{LVR}$  的范围内，这时 LVR 将会自动复位单片机。LVR 包含以下的规格：有效的 LVR 信号，即在  $0.9V \sim V_{LVR}$  的低电压状态的时间，必须超过交流电气特性中  $t_{LVR}$  参数的值。如果低电压存在不超过  $t_{LVR}$  参数的值，则 LVR 将会忽略它且不会执行复位功能。 $V_{LVR}$  参数值可通过配置选项进行设定。

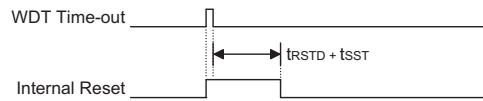


注： $t_{RSTD}$ 为上电延迟时间，典型值为100ms。

**低电压复位时序图**

• 正常运行时看门狗溢出复位

除了看门狗溢出标志位 TO 将被设为“1”之外，正常运行时看门狗溢出复位和 $\overline{\text{RES}}$ 复位相同。

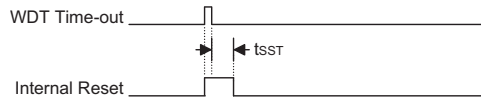


注： $t_{\text{rSTD}}$ 为上电延迟时间，典型值为100ms。

**正常运行时看门狗溢出时序图**

• 休眠时看门狗溢出复位

休眠时看门狗溢出复位和其它种类的复位有些不同。除了程序计数器与堆栈指针将被清“0”及 TO 位被设为“1”外，绝大部分的条件保持不变。图中  $t_{\text{sST}}$  的详细说明请参考交流电气特性。



注：如果系统时钟源为ERC或HIRC时， $t_{\text{sST}}$ 为1024或2个时钟周期。  
如果系统时钟源为HXT或LXT，则 $t_{\text{sST}}$ 为1024个时钟周期。

**休眠时看门狗溢出复位时序图**

**复位初始状态**

不同的复位形式以不同的途径影响复位标志位。这些标志位，即 PDF 和 TO 位存放在状态寄存器中，由休眠功能或看门狗计数器等几种控制器操作控制。复位标志位如下所示：

TO	PDF	复位条件
0	0	上电时的 $\overline{\text{RES}}$ 复位
u	u	正常模式或低速模式时的RES复位或LVR复位
1	u	正常模式或低速模式时的WDT溢出复位
1	1	休眠模式时的WDT溢出复位

注：“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

项目	复位后情况
程序计数器	清除为零
中断	所有中断被除能
看门狗定时器	WDT清除并重新计数
定时/计数器	所有定时/计数器停止
输入/输出口	I/O口设为输入模式
堆栈指针	堆栈指针指向堆栈顶端

不同的复位形式对单片机内部寄存器的影响是不同的。为保证复位后程序能正常执行，了解寄存器在特定条件复位后的设置是非常重要的。下表即为不同方式复位后内部寄存器的状况。若芯片有多种封装类型，表格反应较大的封装的情况。

• HT46R01B/HT46R02B/HT46R01N/HT46R02N

寄存器	上电复位	RES或LVR复位	WDT 溢出 (正常模式)	WDT 溢出 (休眠模式)
PCL	0000 0000	0000 0000	0000 0000	0000 0000
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	- xxx xxxx	- uuu uuuu	- uuu uuuu	- uuu uuuu
WDTS	-----111	-----111	-----111	-----uuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	-- 11 uuuu
INTC0	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
INTC1	-- 00 -- 00	-- 00 -- 00	-- 00 -- 00	-- uu -- uu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1000	0000 1000	0000 1000	uuuu uuuu
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1- - -	0000 1- - -	0000 1- - -	uuuu u- - -
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAWK	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
PB	-----11	-----11	-----11	-----uu
PBC	-----11	-----11	-----11	-----uu
PBPU	-----00	-----00	-----00	-----uu
CTRL0	- 00- 0000	- 00- 0000	- 00- 0000	- uu- uuuu
CTRL1	1000 1010	1000 1010	1000 1010	uuuu uuuu
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xxxx - - - -	xxxx - - - -	xxxx - - - -	uuuu - - - -
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	10 - - -000	10 - - -000	10 - - -000	uu- - -uuu

注：“-”表示未定义  
“u”表示不改变  
“x”表示未知

• HT48R01B/HT48R02B/HT48R01N/HT48R02N

寄存器	上电复位	RES或 LVR 复位	WDT 溢出 (正常模式)	WDT 溢出 (休眠模式)
PCL	0000 0000	0000 0000	0000 0000	0000 0000
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	- xxx xxxx	- uuu uuuu	- uuu uuuu	- uuu uuuu
WDTS	- - - - -111	- - - - -111	- - - - -111	- - - - -uuu
STATUS	- -00 xxxx	- -uu uuuu	- -1u uuuu	- - 11 uuuu
INTC0	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
INTC1	- - 0- - - 0-	- - 0- - - 0-	- - 0- - - 0-	- - u- - - u-
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1000	0000 1000	0000 1000	uuuu uuuu
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1- - -	0000 1- - -	0000 1- - -	uuuu u- - -
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAWK	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
PB	- - - - -11	- - - - -11	- - - - -11	- - - - -uu
PBC	- - - - -11	- - - - -11	- - - - -11	- - - - -uu
PBPU	- - - - -00	- - - - -00	- - - - -00	- - - - -uu
CTRL0	- 0 -- -000	- 0 -- -000	- 0 -- -000	- u -- -uuu
CTRL1	1000 1010	1000 1010	1000 1010	uuuu uuuu

注: “-”表示未定义  
“u”表示不改变  
“x”表示未知

## 输入/输出端口

盛群单片机的输入/输出控制具有很大的灵活性。大部分引脚可在用户程序控制下被设定为输入或输出。所有引脚的上拉电阻设置以及指定引脚的唤醒设置也都由软件控制，这些特性也使得此类单片机在广泛应用上都能符合开发的需求。

作为输入操作，输入引脚无锁存功能，也就是说输入数据必须在执行“MOV A, [m]”，T2 的上升沿准备好，m 为端口地址。对于输出操作，所有数据都是被锁存的，且保持不变直到输出锁存被重写。

### 上拉电阻

许多产品应用在端口处于输入状态时需要外加一个上拉电阻来实现上拉的功能。为了免去外部上拉电阻，当引脚规划为输入时，可由内部连接到一个上拉电阻。这些上拉电阻可通过寄存器 PAPU 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。注意，PA7 引脚没有上拉电阻功能。

### PA 口唤醒

当使用暂停指令“HALT”迫使单片机进入休眠模式，单片机的系统时钟将会停止以降低功耗，此功能对于电池及低功耗应用很重要。唤醒单片机有很多种方法，其中之一就是使 PA0~PA7 的其中一个引脚从高电平转为低电平。使用暂停指令“HALT”迫使单片机进入休眠模式状态后，处理器将会一直保持低功耗状态，直到 PA 口上被选为唤醒输入的引脚电平发生下降沿跳变。这个功能特别适合于通过外部开关来唤醒的应用。注意，PA0~PA7 是可以通过设置 PAWK 寄存器来单独选择是否具有唤醒功能。

#### • PAWK, PAC, PAPU 寄存器

寄存器名称	POR	位							
		7	6	5	4	3	2	1	0
PAWK	00H	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PAC	FFH	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	00H	—	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0

“—” 未定义，读为“0”

**PAWK<sub>n</sub>**: PA 口唤醒功能控制位

0: 除能

1: 使能

**PAC<sub>n</sub>**: I/O 类型选择位

0: 输出

1: 输入

**PAPU<sub>n</sub>**: 上拉功能控制位

0: 除能

1: 使能

#### • PBPU 寄存器

寄存器名称	POR	位							
		7	6	5	4	3	2	1	0
PBC	FFH	—	—	—	—	—	—	PBC1	PBC0
PBPU	00H	—	—	—	—	—	—	PBPU1	PBPU0

“—” 未定义，读为“0”

**PBC<sub>n</sub>**: I/O 类型选择位

0: 输出

1: 输入

**PBPUn:** 上拉功能控制位

- 0: 除能
- 1: 使能

### 输入/输出端口控制寄存器

每一个输入/输出口都具有各自的控制寄存器，即 PAC，PBC，用来控制输入/输出状态。从而每个 I/O 引脚都可以通过软件控制，动态的设置为 COMS 输出或输入。所有的 I/O 端口的引脚都各自对应于 I/O 端口控制的某一位。若 I/O 引脚要实现输入功能，则对应的控制寄存器的位需要设置为“1”。这时程序指令可以直接读取输入脚的逻辑状态。若控制寄存器相应的位被设定为“0”，则此引脚被设置为 COMS 输出。当引脚设置为输出状态时，程序指令读取的是输出端口寄存器的内容。注意，如果对输出口做读取动作时，程序读取到的是内部输出数据锁存器中的状态，而不是输出引脚上实际的逻辑状态。

### 引脚共用功能

引脚的共用功能可以增加单片机应用的灵活性。有限的引脚个数将会限制设计者，而引脚的多功能将会解决很多此类问题。共用引脚的功能选择，有些是由配置选项进行设定，有些则是在应用程序中进行控制。

#### • 外部中断输入

外部中断引脚 INT 与一个 I/O 引脚共用。为了使用该引脚作为外部中断输入引脚，需要正确设置 INTC0 寄存器中的有关位。此外，还需要通过端口控制寄存器中的 PAC3 位来设置该引脚为输入脚。如果需要，可以通过上拉电阻寄存器来选择带上拉电阻。注意即使该引脚被配置为外部中断输入，引脚的输入/输出功能将依然存在。

#### • 外部定时/计数器输入

定时/计数器引脚 TC0 和 TC1 与输入/输出引脚共用。如果设定为定时/计数器的输入，则需要通过设置外部定时/计数器控制寄存器相应的位将外部定时/计数器配置为外部事件计数模式或脉冲宽度测量模式，同时该引脚需要通过端口控制寄存器设置为输入，上拉电阻也可以通过上拉电阻寄存器进行设置。注意，即使该引脚被配置为外部定时/计数器输入，输入/输出功能依然存在。

#### • PFD 输出

此系列单片机均提供有 PFD 信号输出，与输入/输出引脚共用。PFD 的输出可通过 CTRL0 寄存器进行设置。注意端口控制寄存器相应的位需要设置为输出高才能使能 PFD 的输出。如果端口控制寄存器被设置为输入，即使正确设置了 PFD 的输出，该引脚都只作为普通逻辑输入脚，并且允许选择上拉电阻。

#### • PWM 输出

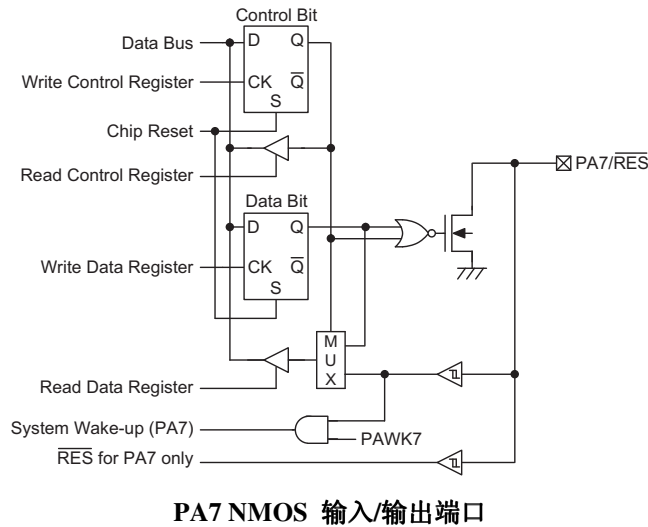
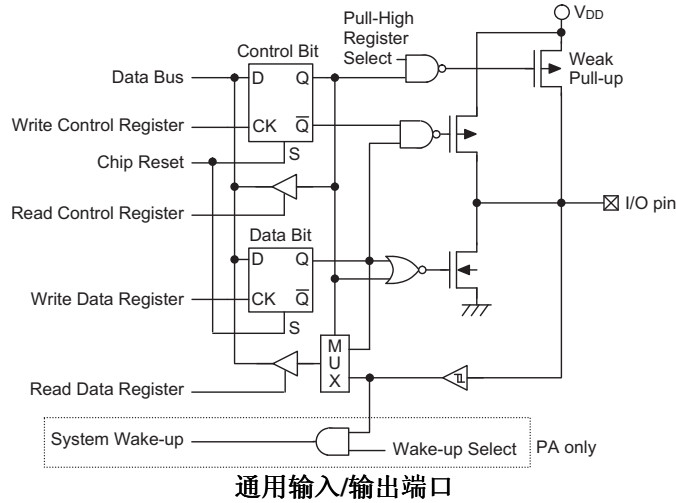
此系列单片机均提供有 PWM 功能，与输入/输出引脚共用。PWM 输出功能通过 CTRL0 寄存器来设置。注意端口控制寄存器相应的位需要设置为输出，才能使能 PWM 的输出。如果端口控制寄存器被设置为输入，即使 PWM 寄存器已经使能 PWM 功能，该引脚都只作为普通逻辑输入脚，并且允许选择上拉电阻。

#### • A/D 输入引脚

此系列单片机具有 4 个 A/D 转换器输入。所有的模拟输入与 PA 引脚共用。如果需要将这些引脚用作为 A/D 输入而非 I/O 脚，则需要正确设定 A/D 转换控制寄存器 ADCR 中相应的 PCRn 位。在配置选项中，没有与 A/D 转换器相关的选项。如果这些引脚作为输入/输出脚使用，仍可以通过配置选项选择是否要接上拉电阻，然而如果作为 A/D 输入使用，则这些引脚上的上拉电阻会自动断开。

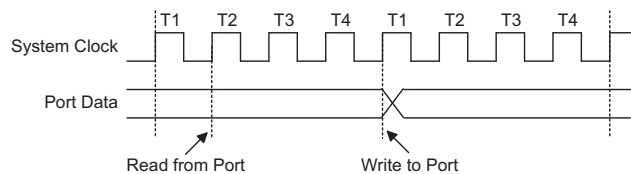
### 输入/输出引脚结构

下图为输入/输出引脚的内部结构图。输入/输出引脚的准确逻辑结构图可能与此图不同，这里只是为了方便对 I/O 引脚功能的理解提供的一个参考。



### 编程注意事项

在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入/输出数据及端口控制寄存器都将被设为逻辑高。所有输入/输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器某些引脚位被设定输出状态，这些输出引脚会有初始高电平输出，除非数据寄存器端口在程序中被预先设定。设置哪些引脚是输入及哪些引脚是输出，可通过设置正确的值到适当的端口控制寄存器，或者使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读-修改-写的操作。单片机需要先读入整个端口上的数据，修改个别的位，然后重新把这些数据写入到输出端口。



读写时序图

PA0~PA7 的每个引脚可通过 PAWK 寄存器设置带唤醒功能。单片机处于休眠模式时，有很多方法可以唤醒单片机，其中之一就是通过 PA 任一引脚电平从高到低转换的方式，可以设置 PA 口一个或多个引脚具有唤醒功能。

## 定时/计数器

定时/计数器在任何单片机中都是一个很重要的部分，提供程序设计者一种实现和时间有关功能的方法。该系列单片机具有两个 8 位的向上计数器。每个定时/计数器有三种不同的工作模式，可以当作一个普通定时器、外部事件计数器或脉冲宽度测量使用。并且提供了一个内部时钟分频器，以扩大定时器的范围。

有两种和定时/计数器相关的寄存器。第一种类型的寄存器是用来存储实际的计数值，赋值给此寄存器可以设定初始值。读取此寄存器可获得定时/计数器的内容。第二种类型的寄存器为定时器控制寄存器，用来定义定时/计数器工作模式和定时设置。定时/计数器的时钟源可来自内部时钟源或外部定时器引脚。

### 配置定时/计数器输入时钟源

定时/计数器的时钟源可有多种选择，可以是内部时钟，也可以是外部引脚。当定时/计数器工作在定时器模式或脉冲宽度测量模式时，使用内部时钟作为时钟源。对于某些定时/计数器，内部时钟首先由分频器分频，分频比由定时器控制寄存器的位 T0PSC0~ T0PSC2 来确定。对于定时/计数器 0，内部时钟源可以通过 TMR0C 寄存器的 T0S 位来选择  $f_{SYS}$  或 LXT 振荡器。

当定时/计数器在事件计数模式时，使用外部时钟源，时钟源由外部时钟输入引脚 TCn 提供。每次外部引脚由高电平到低电平或由低电平到高电平(由 TnEG 位决定)进行转换时，计数器增加一。

### 定时/计数寄存器 - TMR0, TMR1

定时/计数寄存器 TMR0 和 TMR1，是位于特殊数据存储单元内的特殊功能寄存器，用于储存定时器的当前值。在用作内部定时且收到一个内部计数脉冲或用作外部计数且外部定时/计数器引脚发生状态跳变时，此寄存器的值将会加一。定时器将从预置寄存器所载入的值开始计数，到 FFH 时定时器溢出且会产生一个内部中断信号。定时器的值随后被预置寄存器的值重新载入并继续计数。

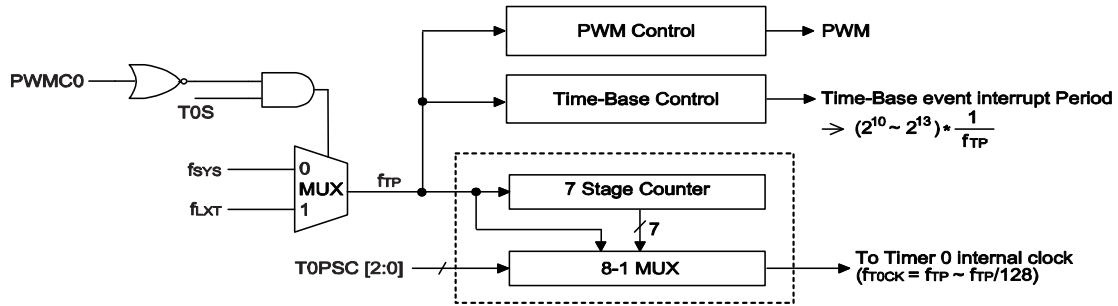
注意，上电后预置寄存器处于未知状态。为了得到定时器的最大计算范围 FFH，预置寄存器需要先清为零。定时/计数器在关闭条件下，写数据到预置寄存器，会立即写入实际的定时器。而如果定时/计数器已经打开且正在计数，在这个周期内写入到预置寄存器的任何新数据将保留在预置寄存器，直到溢出发生时才被写入实际定时器。

### 定时/计数控制寄存器 - TMR0C, TMR1C

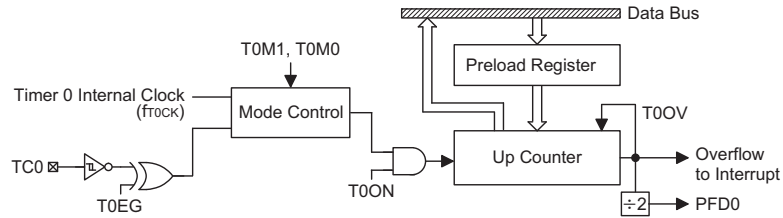
Holtek 单片机灵活的特性也表现在定时器的多功能上，定时/计数器能提供三种不同的工作模式，由相应的控制寄存器来选择定时/计数器的工作方式。

定时/计数控制寄存器为 TMRnC，配合相应的定时寄存器控制定时/计数器的全部操作。在使用定时器之前，需要先正确地设定定时/计数控制寄存器，以便保证定时器能正确操作，而这个过程通常在程序初始化期间完成。

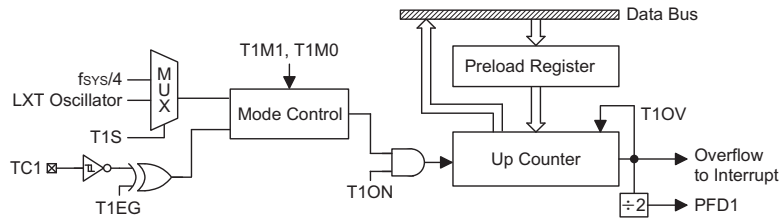
定时/计数控制寄存器的第 7 位和第 6 位，即 TnM1/TnM0，用来设定定时器的工作模式。定时/计数控制寄存器的第 4 位即 TnON，用于定时器开关控制，设定为逻辑高时，计数器开始计数，而清零时则停止计数。定时/计数控制寄存器的第 0~2 位用来控制输入时钟预分频器。如果使用外部时钟源，预分频器位将不起作用。如果定时/计数器工作在外部事件计数模式或脉冲宽度测量模式，TnEG 位即定时控制寄存器的第 3 位将可用来选择上升沿或下降沿触发。TnS 位用来选择内部时钟源。



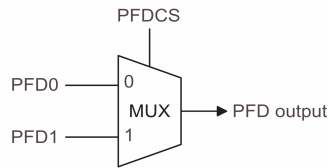
Timer/PWM/Time Base 的时钟源结构图



8 位定时/计数器 0 结构图



8 位定时/计数器 1 结构图



注：当 PWM0 使能时， $f_{TP}$  来自  $f_{SYS}$ （忽略 TOS）

• **TMR0C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	T0S	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

- Bit 7~6 **T0M1, T0M0**: 选择 Timer0 工作模式  
 00: 未使用  
 01: 计数器模式  
 10: 定时器模式  
 11: 脉冲宽度测量模式
- Bit 5 **T0S**: 定时器时钟源  
 0:  $f_{SYS}$   
 1: LXT 振荡器  
 T0S 用来选择 Timer0、时基和 PWM 的时钟源  $f_{TP}$ 。如果 PWM 使能,  $f_{TP}$  的时钟源为  $f_{SYS}$ , 忽略 T0S 的设置。
- Bit 4 **T0ON**: 定时/计数器使能  
 0: 除能  
 1: 使能
- Bit 3 **T0EG**:  
 计数器有效边沿选择  
 0: 在上升沿计数  
 1: 在下降沿计数  
 脉冲宽度测量有效边沿选择  
 0: 在下降沿启动计数, 在上升沿停止计数  
 1: 在上升沿启动计数, 在下降沿停止计数
- Bit 2~0 **T0PSC2, T0PSC1, T0PSC0**: 选择定时器预分频比  
 定时器内部时钟 =  
 000:  $f_{TP}$   
 001:  $f_{TP}/2$   
 010:  $f_{TP}/4$   
 011:  $f_{TP}/8$   
 100:  $f_{TP}/16$   
 101:  $f_{TP}/32$   
 110:  $f_{TP}/64$   
 111:  $f_{TP}/128$

• **TMR1C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	T1S	T1ON	T1EG	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	1	—	—	—

- Bit 7~6 **T1M1, T1M0**: 选择 Timer1 工作模式  
 00: 未使用  
 01: 计数器模式  
 10: 定时器模式  
 11: 脉冲宽度测量模式
- Bit 5 **T1S**: 定时器时钟源  
 0:  $f_{SYS}/4$   
 1: LXT 振荡器
- Bit 4 **T1ON**: 定时/计数器使能  
 0: 除能  
 1: 使能
- Bit 3 **T1EG**:  
 计数器有效边沿选择  
 0: 在上升沿计数

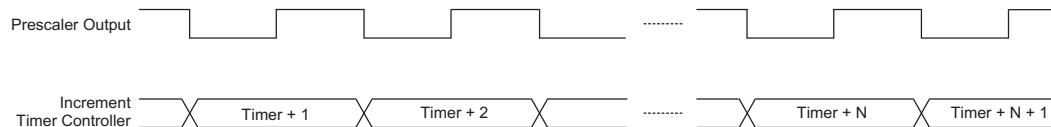
- 1: 在下降沿计数
- 脉冲宽度测量有效边沿选择
- 0: 在下降沿启动计数, 在上升沿停止计数
- 1: 在上升沿启动计数, 在下降沿停止计数

Bit 2~0 未定义, 读为“0”

### 定时器模式

在这个模式下, 定时器可以用来测量固定时间间隔, 当定时器发生溢出时, 就会产生一个内部中断信号。为使定时/计数器工作在定时器模式, **TnM1/TnM0** 需要设置成 1 和 0。

在定时器模式中,  $f_{SYS}$ 、 $f_{SYS}/4$  或 LXT 振荡器被用来当定时器的输入时钟源。然而, 该定时器时钟源被预分频器进一步分频, 分频比是由定时器控制寄存器的 **TnPSC2~TnPSC0** 位来确定。定时器控制寄存器第 4 位, 即 **TnON** 位需要设为逻辑高, 才能令定时器工作。每次内部时钟由高到低的电平转换都会使定时器值增加一; 当定时器计数已满即溢出时, 会产生中断信号且定时器会重新载入预置寄存器的值, 然后继续计数。定时器溢出以及相应的内部中断产生也是唤醒暂停模式的一种方法。通过设置中断寄存器 **INTC0** 中的位 **TnE** 为 0, 可以禁止计数器中断。



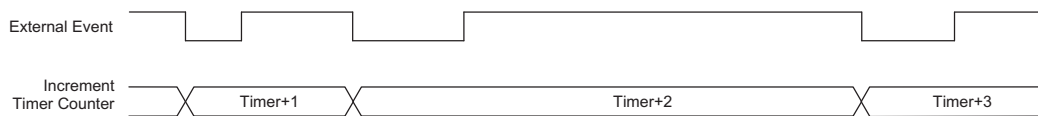
定时器模式时序图

### 外部事件计数模式

定时/计数器工作在外部事件计数模式, 可以通过定时/计数器来记录发生在 **TCn** 引脚的外部逻辑事件变化的次数。为使定时/计数器工作在外部事件计数模式, **TnM1/TnM0** 需要设置成 0 和 1。

在外部事件计数模式, 外部定时脚 **TCn** 被用来当定时/计数器的计时源且不被内部预分频器进一步分频。在设置完定时/计数控制寄存器其它位, 定时/计数器控制寄存器第 4 位, 即 **TnON** 位需要设为逻辑高, 才能使计数器工作。当定时控制寄存器第 3 位, 即 **TnEG** 设置为逻辑低时, 每次外部计数引脚接收到由低到高电平的转换将使计数器加一。而当 **TnEG** 为逻辑高时, 每次外部定时/计数器引脚接收到由高到低电平的转换将使计数器加一。当计数器计数满, 即溢出时会产生中断信号且计数器会重新加载预置寄存器的值, 然后继续计数。计数器溢出中断可通过设置相应的中断寄存器中的定时/计数器中断使能位为 0 而禁止。

由于外部时钟引脚和普通输入/输出引脚共用, 为了确保工作在外部事件计数模式, 要注意两点。首先是要将定时/计数器的工作模式设定在事件计数模式, 其次是确定端口控制寄存器将这个引脚设定为输入状态。注意, 在外部事件计数模式下, 当单片机工作在休眠模式时也保持对外部 **TCn** 引脚的事件计数功能。当计数器溢出时, 将产生一个定时器中断, 并且可以作为唤醒暂停模式的一种方法。



事件计数器模式时序图 (**TnEG=1**)

### 脉冲宽度测量模式

定时/计数器工作在脉冲宽度测量这个模式, 可以测量外部定时器引脚上的外部脉冲宽度。为使定时/计数器工作在脉冲宽度测量模式, **TnM1/TnM0** 需要设置 1 和 1。

在脉冲宽度测量模式中,  $f_{SYS}$ 、 $f_{SYS}/4$  或 LXT 作为 8 位定时/计数器的内部时钟源, 并可被预分频器进一步分频。分频比由预分频选择位 **TnPSC2~TnPSC0**, 即定时控制寄存器的第 2~0 位来确定。在设置完定时/计数控制寄存器其它位, 定时器控制寄存器第 4 位, 即 **TnON** 位需要设为逻辑高,

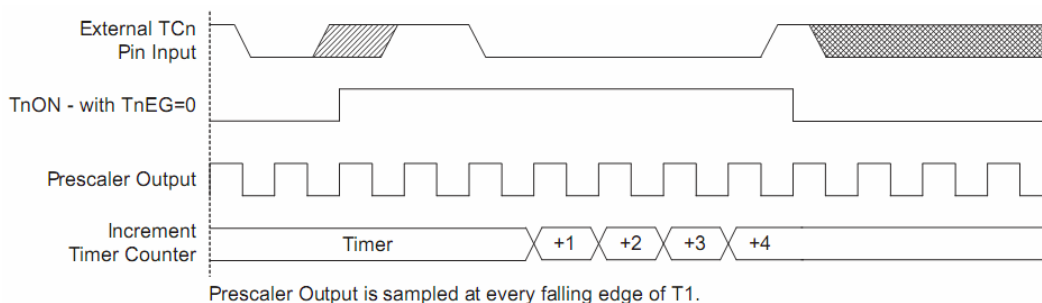
才能使定时/计数器工作。然而，只有当在外部定时器引脚上接收到有效的逻辑转换时，定时/计数器才真正开始启动计数。

当定时控制寄存器第 3 位，即 TnEG 设置为逻辑低时，每次外部定时器引脚接收到由高到低电平的转换时将开始计数直到外部定时/计数器引脚回到它原来的高电平。此时使能位将自动清除为 0 以停止计数。而当 TnEG 为逻辑高时，每次外部定时器接收到由低到高电平的转换时将开始计数直到外部定时/计数器引脚回到它原来的低电平。同样使能位将自动清除为 0 以停止计数。注意，在脉冲宽度测量模式中，当外部定时器上的外部控制信号回到它原来的电平时，使能位将自动地清除为 0。而在其它两种模式，使能位只能在程序控制下清除为 0。

可以通过程序读取定时/计数器当前值，获得 TCn 外部引脚的信号脉冲宽度。当使能位重新复位，任何出现在外部定时器引脚上信号脉冲将被忽略。直到使能位被程序重新置高，开始重新测量外部脉冲。这种方式使得测量窄脉冲将会很容易实现。

注意，在这种模式下，定时/计数器是通过外部定时器引脚上的逻辑转换来控制，而不是通过逻辑电平。当定时/计数器计满，即溢出时会产生中断信号且定时/计数器会重新加载预置寄存器的值，然后继续向上计数。定时/计数器溢出中断可通过设置相应的中断寄存器中的定时/计数器使能位为 0 而禁止。

由于 TCn 引脚和普通输入/输出引脚共用，为了确保工作在脉冲宽度测量模式，要注意两点。首先是要将定时/计数器的工作模式设定在脉冲宽度测量模式，其次是确定端口控制寄存器将这个引脚设定为输入状态。



脉冲宽度测量模式时序图 (TnEG=0)

### 预分频器

TMRnC 寄存器的 TnPSC0~TnPSC2 位用来确定定时/计数器的内部时钟的分频比，从而能够设置更长的定时器溢出周期。

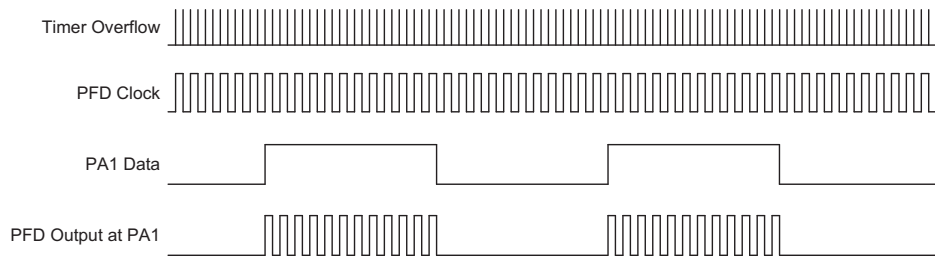
### PF0 功能

PF0 (Programmable Frequency Divider) 提供了一个可编程分频器，适用于像蜂鸣器驱动或其它需要精确频率的场合。

PF0 功能的时钟源是定时/计数器溢出信号，由 CTRL0 中的 PFDCS 来控制。该时钟源可以来自定时/计数器 0 或定时/计数器 1。输出频率的控制可以通过设置适当值到预分频器和定时寄存器来达到要求的分频比。计数器将开始从预置值开始向上计数，直到满，此时，将产生一个溢出信号，导致 PF0 输出改变状态。然后，计数器将自动从预置寄存器中加载预置值，并且继续向上计数。

如果 CTRL0 寄存器已经选择 PF0 功能，为了能对 PF0 输出进行操作，PA 控制寄存器 PAC 的设置是至关重要的，需要设置 PF0 引脚为输出。PA1 需要设置高来激活 PF0。输出数据位可以用来作为 PF0 输出的开关控制位。注意，如果输出数据位清零，PF0 输出将一直输出低。

如果使用晶体振荡器作为系统时钟，通过该方法能产生非常准确的频率。



**PFD 功能**

**输入/输出接口**

当定时/计数器运行在计数或者脉冲宽度测量模式下，定时/计数器需要使用外部定时器引脚以确保正确的动作。由于该引脚是共用引脚，因此需要正确的将其配置为定时/计数器输入引脚。这可以通过定时/计数器控制寄存器的模式选择位来选择是计数模式或脉冲宽度测量模式。此外，相应的端口控制寄存器位需要被设置为高，来确保该引脚是作为输入脚。即使该引脚用作定时/计数器输入，任何连接到这个引脚的上拉电阻将仍然是有效的。

**编程注意事项**

当定时/计数器工作在定时器模式时，内部的系统时钟作为定时器的时钟源，因此与单片机所有运算都能同步。在这个模式下，当定时器寄存器溢出时，微控制器将产生一个内部中断信号，使程序进入相应的内部中断向量。对于脉冲宽度测量模式，定时器时钟源同样使用内部的系统时钟，然而，只有正确的逻辑条件出现在定时器输入引脚时，定时器才开始运行。当这个外部事件没有和内部定时器时钟同步时，只有当下一个定时器时钟到达时，单片机才会看到这个外部事件，因此在测量值上可能有小的差异，需要程序设计者在程序应用时加以注意。同样的情况发生在定时器设置为外部事件计数模式时，它的时钟来源是外部事件，与内部系统时钟或定时器时钟不同步。

当读取定时/计数器值或写数据到预置寄存器时，计数时钟会被禁止以避免发生错误，但这样做可能会导致计数错误，所以程序设计者应该考虑到这点。在第一次使用定时/计数器之前，要仔细确认有没有正确地设定初始值。中断控制寄存器中的定时器使能位需要正确的设置，否则相应定时/计数器内部中断仍然无效。定时/计数器控制寄存器中的触发边沿选择、定时/计数器工作模式和时钟源控制位也需要正确的设定，以确保定时/计数器按照应用需求而正确的配置。在定时/计数器打开之前，需要确保先载入定时/计数器寄存器的初始值；这是因为在上电后，定时/计数器寄存器中的初始值是未知的。定时/计数器初始化后，可以使用定时/计数器控制寄存器中的使能位来打开或关闭定时器。

当定时/计数器产生溢出，中断控制寄存器中相应的中断请求标志将置位。若中断允许，将会依次产生一个中断信号。不管中断是否允许，在省电状态下，定时/计数器的溢出也会产生唤醒。这种情况可能发生在外部信号变化的计数模式中。定时/计数器向上计数直至溢出并唤醒系统。若在省电模式下，不需要定时器中断唤醒系统，可以在执行“HALT”指令之前将相应中断请求标志位置位。

## 定时/计数器应用范例

这个例子说明了如何设置定时/计数器的寄存器，如何设置和控制中断。另外还需注意怎样通过寄存器的第 4 位来启停定时/计数器。此应用范例设置定时/计数器为定时模式，时钟来源于内部的系统时钟。

### PFD 编程应用范例

```

org 04h          ; external interrupt vector

org 08h          ; Timer Counter 0 interrupt vector
jmp tmr0int      ; jump here when Timer 0 overflows
:               :
org 20h          ; main program
:               :
;internal Timer 0 interrupt routine
tmr0int:
:
;Timer 0 main program placed here
:
:
begin:
;setup Timer 0 registers
mov a,09bh      ; setup Timer 0 preload value
mov tmr0,a
mov a,081h      ; setup Timer 0 control register
mov tmr0c,a     ; timer mode and prescaler set to /2
;setup interrupt register
mov a,00dh      ; enable master interrupt and both timer interrupts
mov intc0,a
:               :
set tmr0c.4     ; start Timer 0
:               :

```

## 时基功能

此单片机具有时基功能，用来产生一个有规律的时间间隔信号。

时基功能中时间长度可以通过内部 13 级计数器设置时钟源的分频比来实现，而分频比则是由 CTRL1 寄存器中的 TBSEL0 和 TBSEL1 来设置。另外，TMR0C 寄存器中的 T0S 位可以用来选择时基的时钟源。

当时基溢出时，将产生一个时基中断信号。需要注意的是，时基中断时钟源和定时/计数器的时钟源相同，在编程的时候要多加小心。

## 脉冲宽度调制 - PWM

该系列单片机提供一个 8 位的脉冲宽度调制(PWM)输出。这在马达速率控制应用方面十分有用，通过给相应的 PWM 寄存器设定一定数值，PWM 功能可提供占空比可调但频率固定的 PWM 信号输出。

### PWM 工作模式

在数据存储寄存器中，单片机为 PWM 指定了对应的寄存器，称为 PWM 寄存器。此寄存器为 8 位，表示输出波形中每个调制周期的占空比。为了提高 PWM 调制频率，每个调制周期被划分成两个或四个独立的调制子区段，即分别是 7+1 模式或 6+2 模式。可以通过设置 CTRL0 寄存器来选择 PWM

通道所需的模式和开关控制。注意,当使用 PWM 时,只要将所需的值写入 PWM 寄存器并通过 CTRL0 寄存器设置所需模式和开关控制,单片机内部电路即自动完成 PWM 各子调制周期的划分输出 PWM 信号。PWM 的时钟源只能为系统时钟。将原始调制周期分成 2 个或 4 个子周期的方法,使产生更高的 PWM 频率成为可能,这样可以提供更广泛的应用。使用者需要理解 PWM 频率与 PWM 调制频率的不同之处。PWM 时钟为系统时钟  $f_{SYS}$ , 当 PWM 值为 8 位时,整个 PWM 周期的频率为  $f_{SYS}/256$ 。在 7+1 模式, PWM 调制频率将会是  $f_{SYS}/128$ , 在 6+2 模式, PWM 调制频率将会是  $f_{SYS}/64$ 。

PWM 调制频率	PWM 频率	PWM 占空比
$f_{SYS}/64$ 用于 6+2 模式 $f_{SYS}/128$ 用于 7+1 模式	$f_{SYS}/256$	[PWM]/256

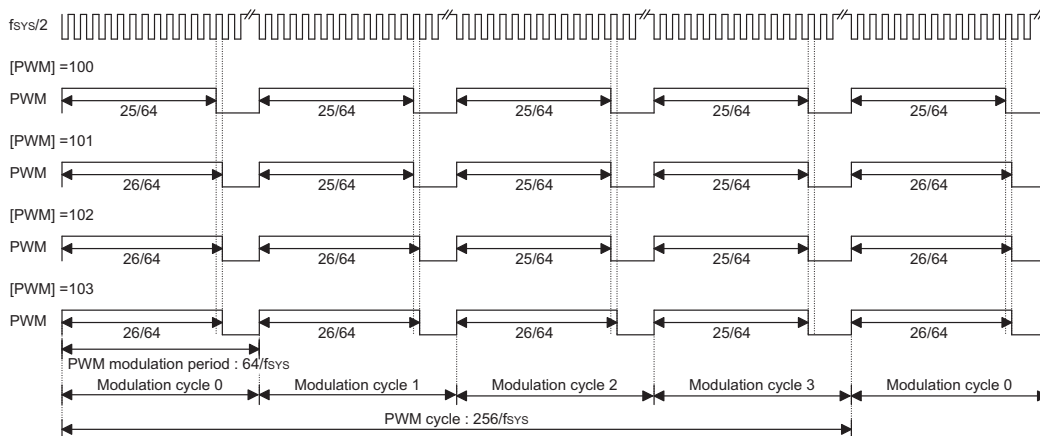
### 6+2 PWM 模式

通过一个 8 位的 PWM 寄存器控制, 每个完整的 PWM 周期由 256 个时钟周期组成。在 6+2 PWM 模式中, 每个 PWM 周期又被分成四个独立的子周期, 称为调制周期 0~调制周期 3, 在表格中以  $i$  表示。四个子周期各包含 64 个时钟周期。在这个模式下, 得到以 4 为因数增加的调制频率。8 位的 PWM 寄存器被分成两个部分, 这个寄存器的值表明整个 PWM 波形的占空比。第一部分包括第 2 位~第 7 位, 表示 DC 值。第二部分为第 0 位~第 1 位, 表示 AC 值。在 6+2 PWM 模式中, 四个调制子周期的占空比, 分别如下表所示。

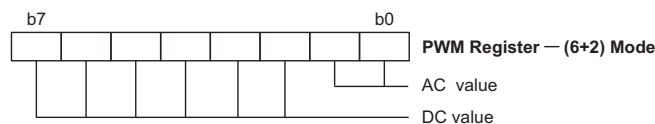
参数	AC (0~3)	DC (占空比)
调制周期 $i$ ( $i=0\sim3$ )	$i < AC$	$\frac{DC + 1}{64}$
	$i \geq AC$	$\frac{DC}{64}$

### 6+2 模式调制周期值

下图表示在 6+2 模式下 PWM 输出的波形。请特别注意单个的 PWM 周期是如何被划分为四个单独的调制周期 0~3 以及 AC 值与 PWM 值之间的关系。



### 6+2 PWM 模式



### 6+2 模式时的 PWM 寄存器

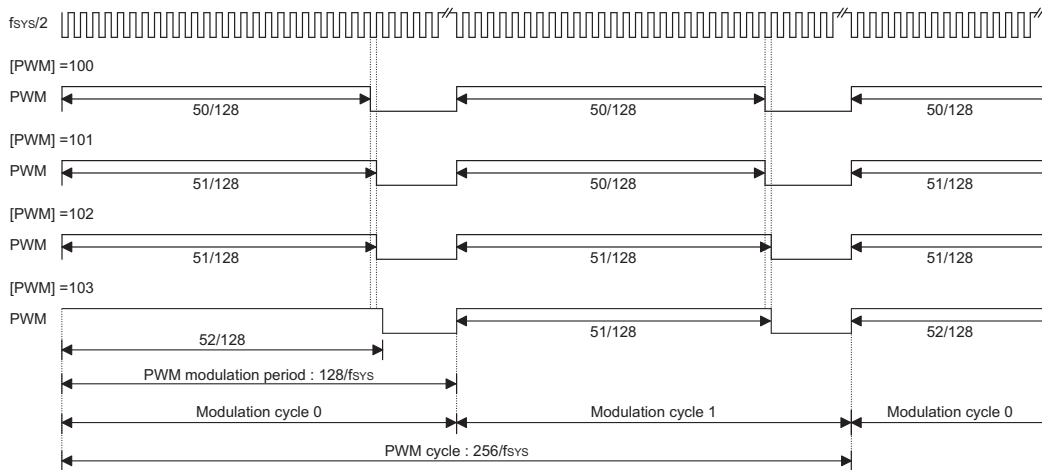
### 7+1 PWM 模式

通过一个 8 位的 PWM 寄存器控制,每个完整的 PWM 周期由 256 个时钟周期组成。在 7+1 PWM 模式中,每个 PWM 周期又被分成两个独立的子周期,称为调制周期 0~调制周期 1,在表格中以“i”表示。两个子周期各包含 128 个时钟周期。在这个模式下,得到以 2 为因数增加的调制频率。8 位的 PWM 寄存器被分成两个部分,这个寄存器的值表明整个 PWM 波形的占空比。第一部分包括第 1 位~第 7 位,表示 DC 值。第二部分为第 0 位,表示 AC 值。在 7+1 PWM 模式中,两个调制子周期的占空比,分别如下表所示。

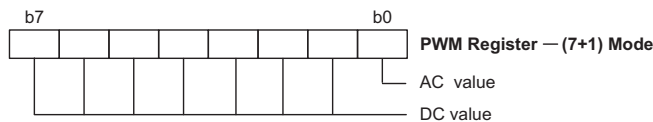
参数	AC (0~1)	DC (占空比)
调制周期 i (i=0~1)	i < AC	$\frac{DC + 1}{128}$
	i ≥ AC	$\frac{DC}{128}$

7+1 模式调制周期值

下图表示在 7+1 模式下 PWM 输出的波形。请特别注意单个的 PWM 周期是如何被划分为两个单独的调制周期 0~1 以及 AC 值与 PWM 值之间的关系。



7+1 PWM 模式



7+1 模式的 PWM 寄存器

### PWM 输出控制

此单片机的 PWM 输出引脚与 I/O 脚 PA4 共用。要使某个引脚作为 PWM 输出而非普通的 I/O 引脚,需要在 CTRL0 寄存器中设置正确的位,在 I/O 端口控制寄存器相应的位 PAC.4 也需要写 0,以确保所需要的 PWM 输出引脚设置为输出状态。在完成这两个初始化步骤,以及将所要求的 PWM 值写入 PWM 寄存器之后,将“1”写入到 PA.4 输出数据寄存器的相应位,使 PWM 数据能够出现在引脚上。将“0”写入到 PA.4 输出数据寄存器的相应位,则会使 PWM 输出功能失效并强制输出低电平。通过这种方式,端口数据寄存器即作为 PWM 功能的开关控制位。注意,如果 CTRL0 寄存器选择 PWM 功能,但是对 PAC 控制寄存器的相应位写入 1 设置此引脚为输入,则该引脚仍可作为带上拉电阻的普通输入端口使用。

### PWM 编程应用范例

```

下面的范例程序说明了如何设置及控制 PWM0 输出
mov  a,64h          ; setup PWM value of decimal 100
mov  pwm0,a
set  ctrl0.5        ; select the 7+1 PWM mode
set  ctrl0.3        ; select pin PA4 to have a PWM function
clr  pac.4          ; setup pin PA4 as an output
set  pa.4           ; enable the PWM output
:
:
clr  pa.4           ; disable the PWM output_pin
                        ; PA4 forced low
    
```

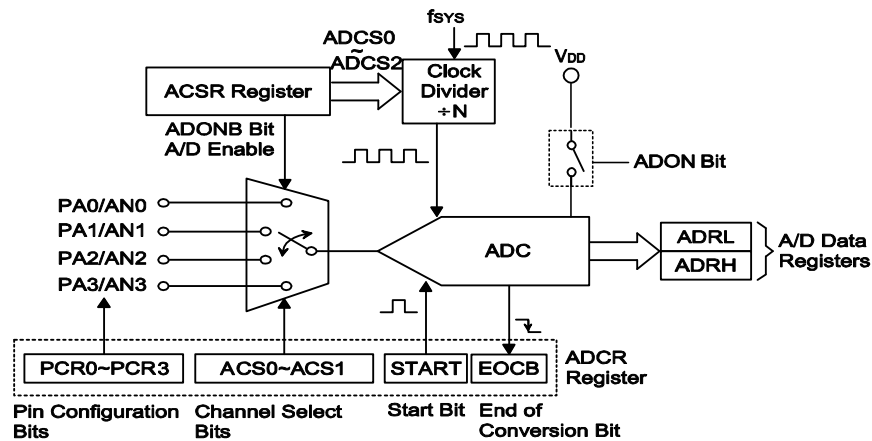
### A/D 转换器

对于大多数的电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

#### A/D 简介

该系列单片机都包含一个 4 通道的 A/D 转换器，它们可以直接接入外部模拟信号(来自传感器或其它控制信号)并直接将这此信号转换成 12 位的数字量。

下图显示了 A/D 转换器内部结构和相关的寄存器。



A/D 转换器结构

#### A/D 转换器数据寄存器 - ADRL, ADRH

对于具有 12 位 A/D 转换器的单片机，需要两个数据寄存器，一个高字节寄存器 ADRH 和一个低字节寄存器 ADRL。在 A/D 转换完毕后，单片机可以直接读取这些寄存器以获得转换结果。只有高位寄存器 ADRH 完全利用了 8 位。而低位寄存器 ADRL 只使用了 8 位中的 4 位，它存放的是 12 位转换值中的低 4 位。在下表中，D0~D11 是 A/D 转换数据结果位。

寄存器	7	6	5	4	3	2	1	0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

A/D 数据寄存器

• **ADRH、ADRL 寄存器**

位	ADRH								ADRL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
名称	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	—	—	—	—
R/W	R	R	R	R	R	R	R	R	R	R	R	R	—	—	—	—
POR	×	×	×	×	×	×	×	×	×	×	×	×	—	—	—	—

“×”表示未知

“—”未定义，读为“0”

**D11~D0:** 是 A/D 转换数据

**A/D 转换控制寄存器 - ADCR, ACSR**

寄存器 ADCR 和 ACSR 用来控制 A/D 转换器的功能和操作。这两个 8 位的寄存器定义包括选择哪一个模拟通道连接至内部 A/D 转换器，哪个引脚是模拟输入，哪个引脚是基本输入/输出端口，A/D 时钟源，并控制和监视 A/D 转换器的开始和复位功能。

寄存器 ADCR 包含 ACS1~ACS0 位，它们定义通道的编号。由于每个单片机只包含一个实际的模数转换电路，因此这 4 个模拟输入中的每一个都需要分别被发送到转换器。ADCR 寄存器中 ACS1~ACS0 位的功能正是决定哪个模拟通道真正连接到内部 A/D 转换器。

ADCR 寄存器中的 PCR3~PCR0 位，用来定义 PA0~PA3 中的哪些引脚为 A/D 转换器的模拟输入，哪些引脚为正常的 I/O。注意，如果 PCR3~PCR0 全都设为“0”，则 PA0~PA3 引脚都被设定为正常的 I/O。

• **ADCR 寄存器**

Bit	7	6	5	4	3	2	1	0
名称	START	EOCB	PCR3	PCR2	PCR1	PCR0	ACS1	ACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	0	0	0	0	0

- Bit 7      **START:** 启动 A/D 转换  
0→1→0    : 启动  
0→1        : 重置 A/D 转换，并且设置 EOCB 为“1”
- Bit 6      **EOCB:** A/D 转换结束标志  
0: A/D 转换结束  
1: A/D 转换中
- Bit 5~2    **PCRn(n=3~0):** A/D 通道配置  
0: 输入/输出口  
1: 模拟输入 ANn(n=0~3)  
如果 PCR0~PCR3 都为 0，则 ADC 模块关闭以降低功耗。
- Bit 1~0    **ACS1,ACS0:** 选择 A/D 通道  
00: AN0  
01: AN1  
10: AN2  
11: 保留位

• **ACSR 寄存器**

Bit	7	6	5	4	3	2	1	0
名称	TEST	ADONB	—	—	—	ADCS2	ADCS1	ADCS0
R/W	R/W	R/W	—	—	—	R/W	R/W	R/W
POR	1	0	—	—	—	0	0	0

- Bit 7           **TEST**: 只用来测试
- Bit 6           **ADONB**: 控制 ADC 模块电源开启/关闭  
 0: ADC 模块电源开启  
 1: ADC 模块电源关闭  
 注意: 1. 建议在进入休眠模式之前, 设置 ADONB=1 以减小功耗  
 2. ADONB=1 将关闭 ADC 模块的电源
- Bit 5~3       未定义, 读为 “0”
- Bit 2~0       **ADCS2~ADCS0**: 选择 A/D 转换器时钟源  
 000:  $f_{SYS}/2$   
 001:  $f_{SYS}/8$   
 010:  $f_{SYS}/32$   
 011: 未定义, 不能使用  
 100:  $f_{SYS}$   
 101:  $f_{SYS}/4$   
 110:  $f_{SYS}/16$   
 111: 未定义, 不能使用

ADCR 寄存器中的 **START** 位, 用于打开和复位 A/D 转换器。当单片机设定此位从逻辑低到逻辑高, 然后再到逻辑低, 就会开始一个模数转换周期。当 **START** 位从逻辑低到逻辑高, 但不再回到逻辑低时, ADCR 寄存器中的 **EOCB** 位置 “1”, 复位模数转换器。**START** 位用于控制内部模数转换器的开/关动作。

ADCR 寄存器中的 **EOCB** 位用于表明模数转换过程的完成。在转换周期结束后, **EOCB** 位会被单片机自动地置为 “0”。此外, 也会置位中断控制寄存器内相应的 A/D 中断请求标志位, 如果中断使能, 就会产生对应的内部中断信号。A/D 内部中断信号将引导程序到相应的 A/D 内部中断入口。如果 A/D 内部中断被禁止, 可以让单片机轮询 ADCR 寄存器中的 **EOCB** 位, 检查此位是否被清除, 以作为另一种侦测 A/D 转换周期结束的方法。

A/D 转换的时钟源为系统时钟  $f_{SYS}$  分频, 而分频系数由 ACSR 寄存器中的 **ADCS2**, **ADCS1** 和 **ADCS0** 位决定。

控制 A/D 转换电路的电源开/关, 是通过使用寄存器 ACSR 中的 **ADONB** 位和 ADCR 寄存器中的 **PCRn** 来实现的(见下表)。不论 **ADONB** 清零还是 **PCRn** 设为零都可关闭 A/D 转换器。所以这在电源敏感的应用中需要多加注意。如下表所示, 执行 **HALT** 时不影响 A/D 转换器开/关控制及所产生的功耗。

PCRn	HALT	ADONB	ADC 开/关
=0	×	×	Off
>0	×	0	On
>0	×	1	Off

×: 无关

**A/D 转换器开/关控制**

虽然 A/D 时钟源是由系统时钟  $f_{SYS}$ , **ADCS2**, **ADCS1** 和 **ADCS0** 位决定, 但可选择的最大 A/D 时钟源则有一些限制。允许的 A/D 时钟周期  $t_{AD}$  的最小值为  $0.5\mu s$ , 当系统时钟速度超过 4MHz 时必须小心。当系统时钟速度等于 4MHz 时, **ADCS2**, **ADCS1** 和 **ADCS0** 位不能设为 “100”。必须保证设定的 A/D 转换时钟周期不小于时钟周期的最小值, 否则将会产生不准确的 A/D 转换值。使用者可以参考下面的表格, 被标上星号\*的数值是不允许的, 因为它们 A/D 转换时钟周期小于规定的最小值。

f <sub>SYS</sub>	A/D 时钟周期 (t <sub>AD</sub> )						
	ADCS2, ADCS1, ADCS0=000 (f <sub>SYS</sub> /2)	ADCS2, ADCS1, ADCS0=001 (f <sub>SYS</sub> /8)	ADCS2, ADCS1, ADCS0=010 (f <sub>SYS</sub> /32)	ADCS2, ADCS1, ADCS0=100 (f <sub>SYS</sub> )	ADCS2, ADCS1, ADCS0=101 (f <sub>SYS</sub> /4)	ADCS2, ADCS1, ADCS0=110 (f <sub>SYS</sub> /16)	ADCS2, ADCS1, ADCS0=011, 111
1MHz	2μs	8μs	32μs	1μs	4μs	16μs	未定义
2MHz	1μs	4μs	16μs	500ns	2μs	8μs	未定义
4MHz	500ns	2μs	8μs	250ns*	1μs	4μs	未定义
8MHz	250ns*	1μs	4μs	125ns*	500ns	2μs	未定义
12MHz	167ns*	667ns	2.67μs	83ns*	333ns*	1μs	未定义

A/D 时钟周期范例

### A/D 输入引脚

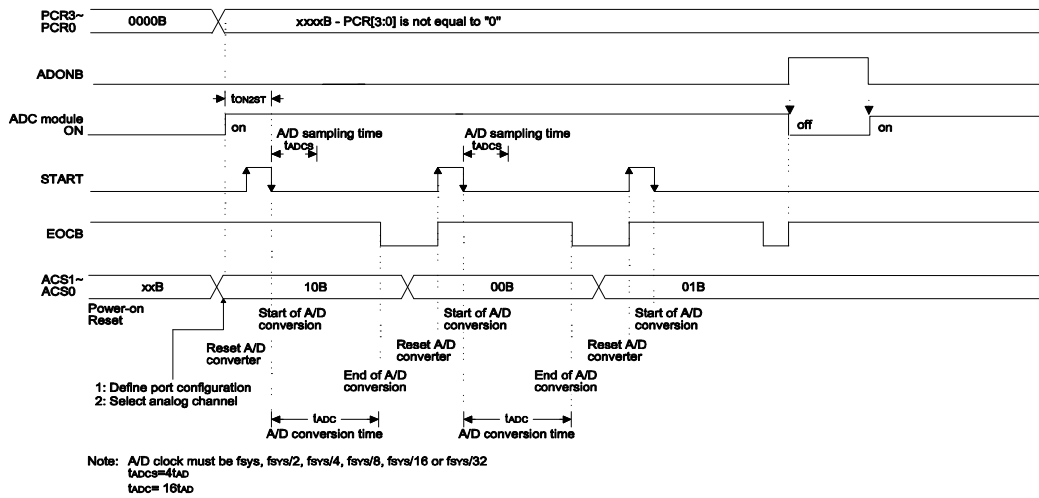
所有的 A/D 模拟输入引脚都与 PA 端口的 I/O 引脚共用。使用 ADCR 寄存器中的 PCR3~PCR0 位，可以将他们设置为普通输入/输出脚或模拟输入脚。通过这种方式，引脚的功能可由程序来控制，灵活地切换引脚功能。当输入引脚作为普通 I/O 脚使用时，可使用上拉电阻，若设置为 A/D 输入，则上拉电阻会自动断开。请注意，PA 端口控制寄存器不需要为使能 A/D 输入而先设定为输入模式，当 PCR3~PCR0 位使能 A/D 输入时，不需要考虑端口控制寄存器的状态。

### A/D 转换步骤

下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1  
通过 ACSR 寄存器中的 ADCS2、ADCS1 和 ADCS0 位，选择所需的 A/D 转换时钟。
- 步骤 2  
通过 ADCR 寄存器中的 PCR3~PCR0 位，选择哪些引脚规划为 A/D 输入引脚。
- 步骤 3  
清零 ACSR 寄存器中的 ADONB 位来使能 A/D。
- 步骤 4  
通过 ADCR 寄存器中的 ACS1~ACS0 位，选择连接至内部 A/D 转换器的通道。
- 步骤 5  
如果要使用中断，则中断控制寄存器需要正确地设置，以确保 A/D 转换功能是激活的。中断控制寄存器 INTC0 里总中断控制位 EMI 需要置位为“1”，以及 INTC1 寄存器中的 A/D 转换器中断位 ADE 也需要置位为“1”。
- 步骤 6  
现在可以通过设定 ADCR 寄存器中的 START 位从“0”到“1”再回到“0”，开始模数转换的过程。注意，该位需初始化为“0”。
- 步骤 7  
可以轮询 ADCR 寄存器中的 EOCB 位，检查模数转换过程是否完成。当此位成为逻辑低时，表示转换过程已经完成。转换完成后，可读取 A/D 数据寄存器 ADRL 和 ADRH 获得转换后的值。另一种方法是，若中断使能且堆栈未滿，则转换完成后，程序会进入 A/D 中断服务子程序。  
注意：若使用轮询 ADCR 寄存器中 EOCB 位的状态的方法来检查转换过程是否结束时，则中断使能的步骤可以省略。

下列时序图表示模数转换过程中不同阶段的图形与时序。



### A/D 转换时序图

A/D 转换器没有对应的配置选项，它的功能设定与操作完全由应用程序控制。由应用程序控制开始 A/D 转换过程后，单片机的内部硬件就会开始进行转换，在这个过程中，程序可以继续其它功能。A/D 转换时间为  $16t_{AD}$ ， $t_{AD}$  为 A/D 时钟周期。

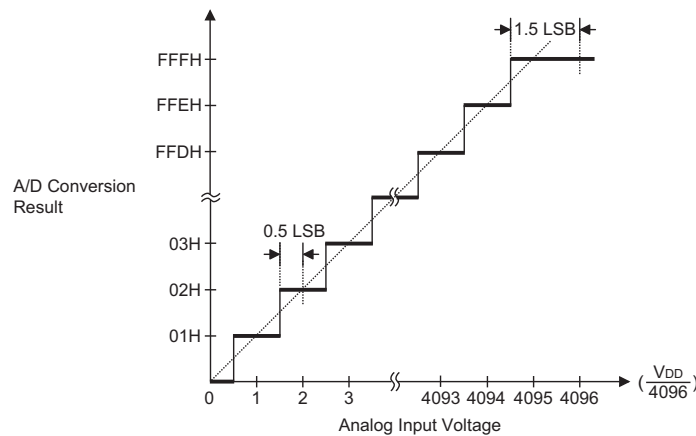
### 编程注意事项

在编程时，需要特别注意寄存器中的 PCR[3:0]。如果这些全部为零，则没有外部引脚连接到 A/D 转换器上，此时的外部引脚可作为普通的 I/O 脚使用。当需要关闭内部 A/D 转换电路以达到降低电源功耗，可以通过设置 ADONB 位为 1 来实现，这一点对电池供电的系统非常重要。

### A/D 转换功能

该系列单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于  $V_{DD}$  的电压值，因此每一位可表示  $V_{DD}/4096$  的模拟输入值。下图显示 A/D 转换器模拟输入值和数字输出值之间理想的转换功能。

为了减少量化错误，A/D 转换器输入端会加入 0.5 LSB 的偏移量。除了数字化数值 0，其后的数字化数值会在精确点之前的 0.5 LSB 处改变，而数字化数值的最大值将在  $V_{DD}$  之前的 1.5 LSB 处改变。



理想的 A/D 转换功能

## A/D 转换应用范例

下面两个范例程序用来说明怎样使用 A/D 转换。第一个范例是轮询 ADCR 寄存器中的 EOCB 位来判断 A/D 转换是否完成；第二个范例则使用中断的方式判断。

**范例：使用查询 EOCB 的方式来检测转换结束**

```

clr  ADE                ; disable ADC interrupt
mov  a,00000001B
mov  ACSR,a             ; select fSYS/8 as A/D clock and ADONB=0
mov  a,00000100B       ; setup ADCR register to configure Port as A/D inputs
mov  ADCR,a             ; and select AN0 to be connected to the A/D converter
:
:
Start_conversion:
clr  START
set  START              ; reset A/D
clr  START              ; start A/D
Polling_EOC:
sz   EOCB               ; poll the ADCR register EOCB bit to detect end
                        ; of A/D conversion

jmp  polling_EOC        ; continue polling
mov  a,ADRL             ; read low byte conversion result value
mov  adr_l_buffer,a     ; save result to user defined register
mov  a,ADRH             ; read high byte conversion result value
mov  adr_h_buffer,a     ; save result to user defined register
:
jmp  start_conversion   ; start next A/D conversion

```

**注：如果需要关闭 ADC 模块的电源，则需要设置 ADONB 为 1。**

范例：使用中斷的方式来检测转换结束

```

clr  ADE                ; disable ADC interrupt
mov  a,00000001B
mov  ACSR,a            ; select fSYS/8 as A/D clock and ADONB=0
mov  a,00000100B      ; setup ADCR register to configure Port as A/D inputs
mov  ADCR,a           ; and select AN0 to be connected to the A/D
:
:
Start_conversion:
clr  START
set  START            ; reset A/D
clr  START            ; start A/D
clr  ADF              ; clear ADC interrupt request flag
set  ADE              ; enable ADC interrupt
set  EMI              ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_:
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
:
mov  a,ADRL           ; read low byte conversion result value
mov  adr1_buffer,a    ; save result to user defined register
mov  a,ADRH           ; read high byte conversion result value
mov  adr2_buffer,a    ; save result to user defined register
:
:
EXIT_ISR:
mov  a,status_stack
mov  STATUS,a        ; restore STATUS from user defined memory
mov  a,acc_stack     ; restore ACC from user defined memory
clr  ADF             ; clear ADC interrupt flag
reti

```

注：如果需要关闭 ADC 模块的电源，则需要设置 ADONB 为 1。

## 中断

中断是单片机一个重要功能。当发生外部中断或内部中断(如定时/计数器或时基), 系统会中止当前的程序, 而转到相对应的中断服务程序中。

此系列单片机提供一个外部中断和多个内部中断, 外部中断由 INT 引脚信号触发, 而内部中断由定时/计数器和时基溢出控制。

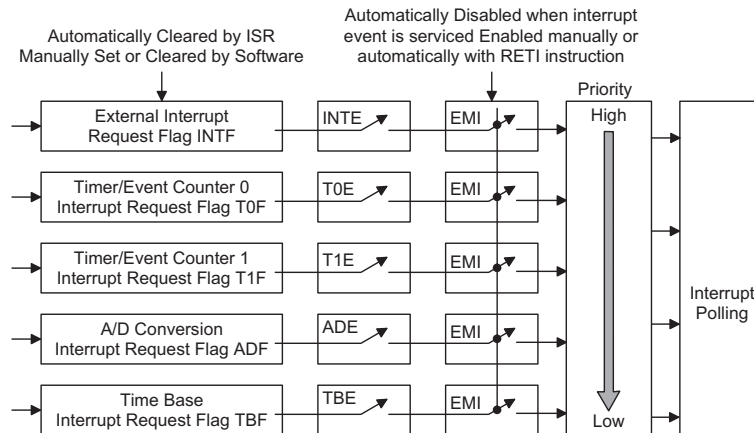
### 中断寄存器

所有中断允许和请求标志均由 INTC0 和 INTC1 寄存器控制。通过控制相应的中断使能位实现对应中断的打开或关闭。当发生中断, 相应中断请求标志将被置位。总中断请求标志清零将关闭所有中断。

### 中断操作

定时/计数器溢出、时基事件或外部中断引脚上有一个有效的边沿信号都会产生一个中断请求, 如果相应的中断允许, 系统将要执行指令的下条地址压入堆栈, 并将相应的中断向量地址加载至 PC 中, 然后从此向量取下条指令。中断向量处通常为跳转指令, 以跳转到相应的中断服务程序。中断服务程序必须以 RETI 指令返回, 系统将先前压入堆栈的地址返回 PC, 以继续执行中断发生时的程序。

各个中断使能位以及相应的请求标志位, 以优先级的顺序如下图所示。

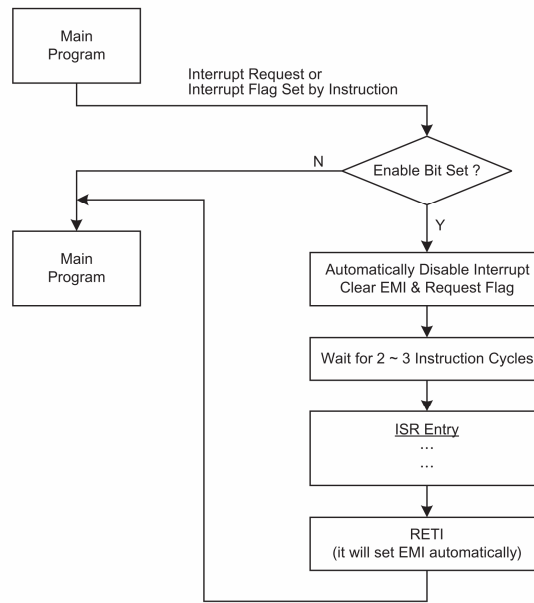


注: HT48R01B/HT48R02B/HT48R01N/HT48R02N 无 ADC 中断

### 中断示意图

一旦中断子程序被响应, 所有其它的中断将被屏蔽(系统自动清除 EMI 位), 这个方式可以防止中断嵌套。如果其它的中断请求发生在此期间, 只有中断请求标志位会被置位。也可以开启中断嵌套, 即在某个中断服务子程序中置位 EMI, 此时如果有另一个中断发生则允许响应此中断。如果堆栈已满, 即使此中断使能, 中断请求也不会被响应, 直到 SP 减少为止。如果要求中断服务程序立即响应, 则堆栈必须避免成为储满状态。

当中断请求产生后, 需要插入 2 个或 3 个指令周期, 程序才能跳转到相应的中断向量地址。而单片机在休眠模式被唤醒时, 需要插入 3 个指令周期, 程序才能跳转到相应的中断向量地址。



中断流程图

### 中断优先级

当中断发生在两个连续的 T2 脉冲上升沿之间时，如果相应的中断请求被允许，中断将在后一个 T2 脉冲响应。下表指出在同时提出请求的情况下的优先权。中断请求可以通过重新设定 EMI 位来加以屏蔽。

• HT46R01B/HT46R02B/HT46R01N/HT46R02N

中断源	优先级	向量
外部中断	1	04H
定时/计数器0溢出中断	2	08H
定时/计数器1溢出中断	3	0CH
A/D转换完成中断	4	10H
时基溢出中断	5	14H

• HT48R01B/HT48R02B/HT48R01N/HT48R02N

中断源	优先级	向量
外部中断	1	04H
定时/计数器0溢出中断	2	08H
定时/计数器1溢出中断	3	0CH
时基溢出中断	4	14H

当外部中断和内部中断均被使能，如果同时发生中断，则外部中断永远优先处理，首先被响应。使用中断寄存器适当地屏蔽个别中断，可以防止同时发生的情况。

### 外部中断

要使外部中断发生，总中断控制位 EMI、外部中断使能位 INTE 需要先被置位。外部中断通过外部 INT 引脚上的电平转换来触发，并置位外部中断请求标志位 INTF。通过 INTEG0 和 INTEG1 位(CTRL1 寄存器的第 6 位和第 7 位)可以设置外部中断触发方式为下降沿触发、上升沿触发或者双边沿触发，也可以设置关闭外部中断功能。

INTEG1	INTEG0	边沿触发类型
0	0	外部中断关闭
0	1	上升沿触发
1	0	下降沿触发
1	1	双沿触发

外部中断与 PA3 共用引脚，如果 INTC0 中相应的外部中断使能位被置位并且在 CTRL1 寄存器中也设置了中断边沿触发类型，PA3 将只能被作为外部中断输入口使用，同时 PAC.3 需将 PA3 设为输入口。当中断使能、堆栈未满足且外部中断产生时，将调用位于地址 04H 处的子程序。当进入外部中断服务程序时，外部中断请求标志位 INTF，EMI 位都会被自动清零以屏蔽其它中断。注意，即使作为外部中断引脚，PA3 依然可以设置带有上拉电阻功能。

• INTC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	T1F	T0F	INTF	T1E	T0E	INTE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

- Bit 7 未定义，读为“0”
- Bit 6 **T1F**: 定时/计数器 1 中断请求标志  
0: 无效  
1: 有效
- Bit 5 **T0F**: 定时/计数器 0 中断请求标志位  
0: 无效  
1: 有效
- Bit 4 **INTF**: 外部中断请求标志位  
0: 无效  
1: 有效
- Bit 3 **T1E**: 定时/计数器 1 中断使能  
0: 除能  
1: 使能
- Bit 2 **T0E**: 定时/计数器 0 中断使能  
0: 除能  
1: 使能
- Bit 1 **INTE**: 外部中断使能  
0: 除能  
1: 使能
- Bit 0 **EMI**: 总中断使能  
0: 除能  
1: 使能

• INTC1 寄存器

◆ HT46R01B/HT46R02B/HT46R01N/HT46R02N

Bit	7	6	5	4	3	2	1	0
Name	—	—	TBF	ADF	—	—	TBE	ADE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 未定义，读为“0”
- Bit 5 **TBF**: 时基中断请求标志  
0: 无效  
1: 有效

- Bit 4           **ADF:** A/D 转换中断请求标志  
0: 无效  
1: 有效
- Bit 3~2       未定义, 读为 “0”
- Bit 1           **TBE:** 时基中断使能  
0: 除能  
1: 使能
- Bit 0           **ADE:** A/D 转换中断使能  
0: 除能  
1: 使能

◆ **HT48R01B/HT48R02B/HT48R01N/HT48R02N**

Bit	7	6	5	4	3	2	1	0
Name	—	—	TBF	—	—	—	TBE	—
R/W	—	—	R/W	—	—	—	R/W	—
POR	—	—	0	—	—	—	0	—

- Bit 7~6       未定义, 读为 “0”
- Bit 5           **TBF:** 时基中断请求标志  
0: 无效  
1: 有效
- Bit 4~2       未定义, 读为 “0”
- Bit 1           **TBE:** 时基中断使能  
0: 除能  
1: 使能
- Bit 0           未定义, 读为 “0”

### 定时/计数器中断

要产生定时/计数器中断, 总中断控制位 **EMI** 和相应的定时/计数器中断使能位 **TnE** 需要先被置位。当定时/计数器发生溢出, 相应的中断请求标志位 **TnF** 将置位并触发定时/计数器中断。若中断使能, 堆栈未满, 当发生定时/计数器中断时, 将调用相应定时器中断子程序。当定时/计数器中断被响应时, 中断请求标志位 **TnF** 被复位且 **EMI** 被清零以除能其它中断。

### 时基中断

要产生时基中断, 总中断使能位 **EMI** 和时基中断使能位 **TBE** 需要先被置位。当时基发生溢出, 将置位时基请求标志位 **TBF**, 并触发时基中断。当中断被允许且堆栈未满, 一旦时基发生溢出, 将调用相应时基中断子程序。当时基中断响应时, 时基中断标志位 **TBF** 被复位且 **EMI** 位被清零以除能其它中断。

### 编程注意事项

通过除能中断使能位, 可以屏蔽中断请求。然而, 一旦请求标志位被置位, 它将保存在中断寄存器中, 直到相应的中断被响应或被软件指令清除。

建议用户不要在中断子程序中使用 “Call 子程序” 指令。中断通常发生在不可预料的情况或需要立即执行的某些应用。假如只剩下一层堆栈且没有控制好中断, 一旦 “Call 子程序” 在中断子程序中执行时, 将破坏原来的控制序列。

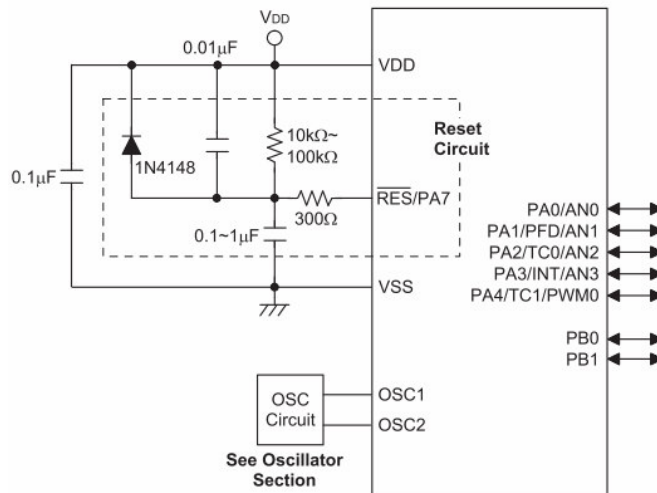
所有的中断都具有将处于休眠模式的单片机唤醒的功能。但只有程序计数器被压入堆栈中, 一旦中断服务程序使寄存器和状态寄存器中的内容发生改变, 则会破坏想要的控制序列, 因此需要事先将这些数据保存起来。

## 配置选项

配置选项在烧写程序时写入芯片。通过 HT-IDE 的软件开发环境，使用者在开发过程中可以选择配置选项。当配置选项烧入单片机后，无法再通过应用程序修改。所有位必须按系统的需要定义，具体内容可参考下表：

编号	选 项
1	看门狗定时器：打开或关闭
2	看门狗定时器时钟源：LXT, LIRC 或 $f_{SYS}/4$ 注：如果 WDT 时钟源来自 LXT，在 OSC 配置选项需要选择 LXT 振荡器
3	清除看门狗指令：1 条或 2 条
4	系统振荡器配置选项：HXT, HIRC, ERC, HIRC+LXT
5	LVR 功能：使能或禁止
6	LVR 电压：2.1V, 3.15V 或 4.2V
7	RES 或 PA7 选择
8	系统启动延时时间：1024 个时钟或 2 个时钟(为 HIRC/ERC 选择 $t_{SST}$ )
9	内部 RC：4MHz, 8MHz 或 12MHz

## 应用电路



## 指令集

### 简介

任何单片机成功运作的核心在于它的指令集，此指令集为一组程序指令码，用来指导单片机如何去执行指定的工作。在 HOLTEK 单片机中，提供了丰富且灵活的指令，共超过六十条，程序设计者可以事半功倍地实现他们的应用。

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

### 指令周期

大部分的操作均只需要一个指令周期来执行。分支、调用或查表则需要两个指令周期。一个指令周期相当于四个系统时钟周期，因此如果在 8MHz 的系统时钟振荡器下，大部分的操作将在 0.5 $\mu$ s 中执行完成，而分支或调用操作则将在 1 $\mu$ s 中执行完成。虽然需要两个指令周期的指令通常指的是 JMP、CALL、RET、RETI 和查表指令，但如果牵涉到程序计数器低字节寄存器 PCL 也将多花费一个周期去加以执行。即指令改变 PCL 的内容进而导致直接跳转至新地址时，需要多一个周期去执行，例如“CLR PCL”或“MOV PCL, A”指令。对于跳转指令必须注意的是，如果比较的结果牵涉到跳转动作将多花费一个周期，如果没有则需一个周期即可。

### 数据的传送

单片机程序中数据传送是使用最为频繁的操作之一，使用三种 MOV 的指令，数据不但可以从寄存器转移至累加器(反之亦然)，而且能够直接移动立即数到累加器。数据传送最重要的应用之一是从输入端口接收数据或传送数据到输出端口。

### 算术运算

算术运算和数据处理是大部分单片机应用所必需具备的能力，在盛群单片机内部的指令集中，可直接实现加与减的运算。当加法的结果超出 255 或减法的结果少于 0 时，要注意正确的处理进位和借位的问题。INC、INCA、DEC 和 DECA 指令提供了对一个指定地址的值加一或减一的功能。

### 逻辑和移位运算

标准逻辑运算例如 AND、OR、XOR 和 CPL 全都包含在盛群单片机内部的指令集中。大多数牵涉到数据运算的指令，数据的传送必须通过累加器。在所有逻辑数据运算中，如果运算结果为零，则零标志位将被置位，另外逻辑数据运用形式还有移位指令，例如 RR、RL、RRC 和 RLC 提供了向左或向右移动一位的方法。不同的移位指令可满足不同的应用需要。移位指令常用于串行端口的程序应用，数据可从内部寄存器转移至进位标志位，而此位则可被检验，移位运算还可应用在乘法与除法的运算组成中。

### 分支和控制转换

程序分支是采取使用 JMP 指令跳转至指定地址或使用 CALL 指令调用子程序的形式，两者之不同在于当子程序被执行完毕后，程序必须马上返回原来的地址。这个动作是由放置在子程序里的返回指令 RET 来实现，它可使程序跳回 CALL 指令之后的地址。在 JMP 指令中，程序则只是跳到一个指定的地址而已，并不需如 CALL 指令般跳回。一个非常有用的分支指令是条件跳转，跳转条件是由数据存储器或指定位来加以决定。遵循跳转条件，程序将继续执行下一条指令或略过且跳转至接下来的指令。这些分支指令是程序走向的关键，跳转条件可能是外部开关输入，或是内部数据位的值。

### **位运算**

提供数据存储单元中单个位的运算指令是盛群单片机的特性之一。这特性对于输出端口位的设置尤其有用，其中个别的位或端口的引脚可以使用“SET [m].i”或“CLR [m].i”指令来设定其为高位或低位。如果没有这特性，程序设计师必须先读入输入输出的 8 位数据，处理这些数据，然后再输出正确的新数据。这种读入-修改-写出的过程现在则被位运算指令所取代。

### **查表运算**

数据的储存通常由寄存器完成，然而当处理大量固定的数据时，它的存储量常常造成对个别存储器的不便。为了改善此问题，盛群单片机允许在程序存储器中建立一个表格作为数据可直接存储的区域，只需要一组简易的指令即可对数据进行查表。

### **其它运算**

除了上述功能指令外，其它指令还包括用于省电的“HALT”指令和使程序在极端电压或电磁环境下仍能正常工作的看门狗定时器控制指令。这些指令的使用则请查阅相关的章节。

## 指令集概要

下表中说明了按功能分类的指令集，用户可以将该表作为基本的指令参考。

### 惯例

x: 立即数                      m: 数据存储器地址  
A: 累加器                    i: 第 0~7 位                      addr: 程序存储器地址

助记符		说明	指令周期	影响标志位
<b>算术运算</b>				
ADD	A,[m]	ACC 与数据存储器相加，结果放入 ACC	1	Z,C,AC,OV
ADDM	A,[m]	ACC 与数据存储器相加，结果放入数据存储器	1 <sup>注</sup>	Z,C,AC,OV
ADD	A, x	ACC 与立即数相加，结果放入 ACC	1	Z,C,AC,OV
ADC	A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	1	Z,C,AC,OV
ADCM	A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	1 <sup>注</sup>	Z,C,AC,OV
SUB	A, x	ACC 与立即数相减，结果放入 ACC	1	Z,C,AC,OV
SUB	A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z,C,AC,OV
SUBM	A,[m]	ACC 与数据存储器相减，结果放入数据存储器	1 <sup>注</sup>	Z,C,AC,OV
SBC	A,[m]	ACC 与数据存储器、进位标志的反相减，结果放入 ACC	1	Z,C,AC,OV
SBCM	A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1 <sup>注</sup>	Z,C,AC,OV
DAA	[m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1 <sup>注</sup>	C
<b>逻辑运算</b>				
AND	A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR	A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR	A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM	A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1 <sup>注</sup>	Z
ORM	A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1 <sup>注</sup>	Z
XORM	A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1 <sup>注</sup>	Z
AND	A, x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR	A, x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR	A, x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL	[m]	对数据存储器取反，结果放入数据存储器	1 <sup>注</sup>	Z
CPLA	[m]	对数据存储器取反，结果放入 ACC	1	Z
<b>递增和递减</b>				
INCA	[m]	递增数据存储器，结果放入 ACC	1	Z
INC	[m]	递增数据存储器，结果放入数据存储器	1 <sup>注</sup>	Z
DECA	[m]	递减数据存储器，结果放入 ACC	1	Z
DEC	[m]	递减数据存储器，结果放入数据存储器	1 <sup>注</sup>	Z
<b>移位</b>				
RRA	[m]	数据存储器右移一位，结果放入 ACC	1	无
RR	[m]	数据存储器右移一位，结果放入数据存储器	1 <sup>注</sup>	无
RRCA	[m]	带进位将数据存储器右移一位，结果放入 ACC	1	C
RRC	[m]	带进位将数据存储器右移一位，结果放入数据存储器	1 <sup>注</sup>	C
RLA	[m]	数据存储器左移一位，结果放入 ACC	1	无
RL	[m]	数据存储器左移一位，结果放入数据存储器	1 <sup>注</sup>	无
RLCA	[m]	带进位将数据存储器左移一位，结果放入 ACC	1	C
RLC	[m]	带进位将数据存储器左移一位，结果放入数据存储器	1 <sup>注</sup>	C
<b>数据传送</b>				
MOV	A,[m]	将数据存储器送至 ACC	1	无
MOV	[m],A	将 ACC 送至数据存储器	1 <sup>注</sup>	无
MOV	A, x	将立即数送至 ACC	1	无
<b>位运算</b>				

助记符	说明	指令周期	影响标志位
CLR [m].i	清除数据存储器的位	1 <sup>注</sup>	无
SET [m].i	置位数据存储器的位	1 <sup>注</sup>	无
<b>转移</b>			
JMP addr	无条件跳转	2	无
SZ [m]	如果数据存储器为零, 则跳过下一条指令	1 <sup>注</sup>	无
SZA [m]	数据存储器送至 ACC, 如果内容为零, 则跳过下一条指令	1 <sup>注</sup>	无
SZ [m].i	如果数据存储器的第 i 位为零, 则跳过下一条指令	1 <sup>注</sup>	无
SNZ [m].i	如果数据存储器的第 i 位不为零, 则跳过下一条指令	1 <sup>注</sup>	无
SIZ [m]	递增数据存储器, 如果结果为零, 则跳过下一条指令	1 <sup>注</sup>	无
SDZ [m]	递减数据存储器, 如果结果为零, 则跳过下一条指令	1 <sup>注</sup>	无
SIZA [m]	递增数据存储器, 将结果放入 ACC, 如果结果为零, 则跳过下一条指令	1 <sup>注</sup>	无
SDZA [m]	递减数据存储器, 将结果放入 ACC, 如果结果为零, 则跳过下一条指令	1 <sup>注</sup>	无
CALL addr	子程序调用	2	无
RET	从子程序返回	2	无
RET A, x	从子程序返回, 并将立即数放入 ACC	2	无
RETI	从中断返回	2	无
<b>查表</b>			
TABRDC [m]	读取当前页的 ROM 内容, 并送至数据存储器 and TBLH	2 <sup>注</sup>	无
TABRDL [m]	读取最后页的 ROM 内容, 并送至数据存储器 and TBLH	2 <sup>注</sup>	无
<b>其它指令</b>			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1 <sup>注</sup>	无
SET [m]	置位数据存储器	1 <sup>注</sup>	无
CLR WDT	清除看门狗定时器	1	TO,PDF
CLR WDT1	预清除看门狗定时器	1	TO,PDF
CLR WDT2	预清除看门狗定时器	1	TO,PDF
SWAP [m]	交换数据存储器的高低字节, 结果放入数据存储器	1 <sup>注</sup>	无
SWAPA [m]	交换数据存储器的高低字节, 结果放入 ACC	1	无
HALT	进入暂停模式	1	TO,PDF

- 注: 1、对跳转指令而言, 如果比较的结果牵涉到跳转即需 2 个周期, 如果没有发生跳转, 则只需一个周期。  
2、任何指令若要改变 PCL 的内容将需要 2 个周期来执行。  
3、对于“CLR WDT1”或“CLR WDT2”指令而言, TO 和 PDF 标志位也许会受执行结果影响, “CLR WDT1”和“CLR WDT2”被连续地执行后, TO 和 PDF 标志位会被清除, 否则 TO 和 PDF 标志位保持不变。

### 指令定义

- ADC A, [m]** Add data memory and carry to the accumulator  
说明: 将指定的数据存储器、累加器内容以及进位标志相加, 结果存放到累加器。  
运算过程:  $ACC \leftarrow ACC + [m] + C$   
影响标志位: OV、Z、AC、C
- ADCM A, [m]** Add the accumulator and carry to the accumulator  
说明: 将指定的数据存储器、累加器内容和进位标志位相加, 结果存放到指定的数据存储器。  
运算过程:  $[m] \leftarrow ACC + [m] + C$   
影响标志位: OV、Z、AC、C
- ADD A, [m]** Add data memory to the accumulator  
说明: 将指定的数据存储器 and 累加器内容相加, 结果存放到累加器。

运算过程:	$ACC \leftarrow ACC + [m]$
影响标志位:	OV、Z、AC、C
<b>ADD</b> <b>A, x</b>	Add immediate data to the accumulator
说明:	将累加器和立即数相加，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC + x$
影响标志位:	OV、Z、AC、C
<b>ADDM</b> <b>A, [m]</b>	Add the accumulator to the data memory
说明:	将指定的数据存储器 and 累加器内容相加，结果存放到指定的数据存储器。
运算过程:	$[m] \leftarrow ACC + [m]$
影响标志位:	OV、Z、AC、C
<b>AND</b> <b>A, [m]</b>	Logical AND accumulator with data memory
说明:	将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "AND" } [m]$
影响标志位:	Z
<b>AND</b> <b>A, x</b>	Logical AND immediate data to the accumulator
说明:	将累加器中的数据和立即数做逻辑与，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "AND" } x$
影响标志位:	Z
<b>ANDM</b> <b>A, [m]</b>	Logical AND data memory with the accumulator
说明:	将指定数据存储器内容和累加器中的数据做逻辑与，结果存放到数据存储器。
运算过程:	$[m] \leftarrow ACC \text{ "AND" } [m]$
影响标志位:	Z
<b>CALL</b> <b>addr</b>	Subroutine call
说明:	无条件的调用指定地址的子程序，此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈，接着载入指定地址并从新地址执行程序。由于指令需要额外的运算，所以此指令为 2 个周期。
运算过程:	$Stack \leftarrow Program\ Counter + 1$ $Program\ Counter \leftarrow addr$
影响标志位:	无
<b>CLR</b> <b>[m]</b>	Clear data memory
说明:	将指定数据存储器的内容清零。
运算过程:	$[m] \leftarrow 00H$
影响标志位:	无
<b>CLR</b> <b>[m].i</b>	Clear bit of data memory
说明:	将指定数据存储器的 i 位内容清零。
运算过程:	$[m].i \leftarrow 0$
影响标志位:	无
<b>CLR</b> <b>WDT</b>	Clear Watchdog Timer
说明:	WDT 计数器、暂停标志位 PDF 和看门狗溢出标志位 TO 清零。
运算过程:	$WDT \leftarrow 00H$ $PDF \ \& \ TO \leftarrow 0$
影响标志位:	TO、PDF

<b>CLR</b>	<b>WDT1</b>	<p>Preclear Watchdog Timer</p> <p>说明: PDF 和 TO 标志位都被清 0。必须配合 CLR WDT2 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT1, 而没有执行 CLR WDT2 时, PDF 与 TO 保留原状态不变。</p> <p>运算过程: WDT ← 00H PDF &amp; TO ← 0</p> <p>影响标志位: TO、PDF</p>
<b>CLR</b>	<b>WDT2</b>	<p>Preclear Watchdog Timer</p> <p>说明: PDF 和 TO 标志位都被清 0。必须配合 CLR WDT1 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT2, 而没有执行 CLR WDT1 时, PDF 与 TO 保留原状态不变。</p> <p>运算过程: WDT ← 00H PDF &amp; TO ← 0</p> <p>影响标志位: TO、PDF</p>
<b>CPL</b>	<b>[m]</b>	<p>Complement data memory</p> <p>说明: 将指定数据存储器中的每一位取逻辑反, 相当于从 1 变 0 或从 0 变 1。</p> <p>运算过程: [m] ← [m̄]</p> <p>影响标志位: Z</p>
<b>CPLA</b>	<b>[m]</b>	<p>Complement data memory</p> <p>说明: 将指定数据存储器中的每一位取逻辑反, 相当于从 1 变 0 或从 0 变 1, 结果被存放回累加器且数据寄存器的内容保持不变。</p> <p>运算过程: ACC ← [m̄]</p> <p>影响标志位: Z</p>
<b>DAA</b>	<b>[m]</b>	<p>Decimal-Adjust accumulator for addition</p> <p>说明: 将累加器中的内容转换为 BCD (二进制转成十进制) 码。如果低四位的值大于“9”或 AC=1, 那么 BCD 调整就执行对原值加“6”, 否则原值保持不变; 如果高四位的值大于“9”或 C=1, 那么 BCD 调整就执行对原值加“6”。BCD 转换实质上是根据累加器和标志位执行 00H, 06H, 60H 或 66H 的加法运算, 结果存放回累加器。只有进位标志位 C 受影响, 用来指示原始 BCD 的和是否大于 100, 并可以进行双精度十进制数的加法运算。</p> <p>操作: [m] ← ACC+00H 或 [m] ← ACC+06H [m] ← ACC+60H 或 [m] ← ACC+66H</p> <p>影响标志位: C</p>
<b>DEC</b>	<b>[m]</b>	<p>Decrement data memory</p> <p>说明: 将指定数据存储器的内容减 1。</p> <p>运算过程: [m] ← [m]-1</p> <p>影响标志位: Z</p>
<b>DECA</b>	<b>[m]</b>	<p>Decrement data memory and place result in the accumulator</p> <p>说明: 将指定数据存储器的内容减 1, 把结果存放回累加器并保持指定数据存储器的内容不变。</p> <p>运算过程: ACC ← [m]-1</p> <p>影响标志位: Z</p>

<b>HALT</b>	Enter power down mode
说明:	此指令终止程序执行并关掉系统时钟, RAM 和寄存器的内容保持原状态, WDT 计数器和分频器被清“0”, 暂停标志位 PDF 被置位 1, WDT 溢出标志位 TO 被清 0。
运算过程:	TO $\leftarrow$ 0 PDF $\leftarrow$ 1
影响标志位:	TO、PDF
<b>INC [m]</b>	Increment data memory
说明:	将指定数据存储器的内容加 1。
运算过程:	[m] $\leftarrow$ [m]+1
影响标志位:	Z
<b>INCA [m]</b>	Increment data memory and place result in the accumulator
说明:	将指定数据存储器的内容加 1, 结果存放回累加器并保持指定的数据存储器内容不变。
运算过程:	ACC $\leftarrow$ [m]+1
影响标志位:	Z
<b>JMP addr</b>	Directly jump
说明:	程序计数器的内容无条件地由被指定的地址取代, 程序由新的地址继续执行。当新的地址被加载时, 必须插入一个空指令周期, 所以此指令为 2 个周期的指令。
运算过程:	PC $\leftarrow$ addr
影响标志位:	无
<b>MOV A, [m]</b>	Move data memory to the accumulator
说明:	将指定数据存储器的内容复制到累加器。
运算过程:	ACC $\leftarrow$ [m]
影响标志位:	无
<b>MOV A, x</b>	Move immediate data to the accumulator
说明:	将 8 位立即数载入累加器。
运算过程:	ACC $\leftarrow$ x
影响标志位:	无
<b>MOV [m], A</b>	Move the accumulator data to memory
说明:	将累加器的内容复制到指定的数据存储器。
运算过程:	[m] $\leftarrow$ ACC
影响标志位:	无
<b>NOP</b>	No operation
说明:	空操作, 顺序执行下一条指令。
运算过程:	PC $\leftarrow$ PC+1
影响标志位:	无
<b>OR A, [m]</b>	Logical OR accumulator with data memory
说明:	将累加器中的数据和指定的数据存储器内容逻辑或, 结果存放到累加器。
运算过程:	ACC $\leftarrow$ ACC “OR” [m]
影响标志位:	Z
<b>OR A, x</b>	Logical OR immediate data to the accumulator
说明:	将累加器中的数据和立即数逻辑或, 结果存放到累加器。
运算过程:	ACC $\leftarrow$ ACC “OR” x
影响标志位:	Z

<p><b>ORM</b>    <b>A, [m]</b></p> <p>说明:</p> <p>运算过程:</p> <p>影响标志位:</p>	<p>Logical OR data memory with accumulator</p> <p>将存在指定数据存储器中的数据 and 累加器逻辑或，结果放到数据存储器。</p> <p><math>[m] \leftarrow \text{ACC} \text{ "OR" } [m]</math></p> <p>Z</p>
<p><b>RET</b></p> <p>说明:</p> <p>运算过程:</p> <p>影响标志位:</p>	<p>Return from subroutine</p> <p>将堆栈寄存器中的程序计数器值恢复，程序由取回的地址继续执行。</p> <p><math>\text{PC} \leftarrow \text{Stack}</math></p> <p>无</p>
<p><b>RET A, x</b></p> <p>说明:</p> <p>运算过程:</p> <p>影响标志位:</p>	<p>Return and place immediate data in the accumulator</p> <p>将堆栈寄存器中的程序计数器值恢复且累加器载入指定的立即数，程序由取回的地址继续执行。</p> <p><math>\text{PC} \leftarrow \text{Stack}</math></p> <p><math>\text{ACC} \leftarrow x</math></p> <p>无</p>
<p><b>RETI</b></p> <p>说明:</p> <p>运算过程:</p> <p>影响标志位:</p>	<p>Return from interrupt</p> <p>将堆栈寄存器中的程序计数器值恢复且中断功能通过设置 EMI 位重新使能。EMI 是控制中断使能的主控制位。如果在执行 RETI 指令之前还有中断未被相应，则这个中断将在返回主程序之前被相应。</p> <p><math>\text{PC} \leftarrow \text{Stack}</math></p> <p><math>\text{EMI} \leftarrow 1</math></p> <p>无</p>
<p><b>RL</b>        <b>[m]</b></p> <p>说明:</p> <p>运算过程:</p> <p>影响标志位:</p>	<p>Rotate data memory left</p> <p>将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。</p> <p><math>[m].(i+1) \leftarrow [m].i \quad (i=0\sim6)</math></p> <p><math>[m].0 \leftarrow [m].7</math></p> <p>无</p>
<p><b>RLA</b>      <b>[m]</b></p> <p>说明:</p> <p>运算过程:</p> <p>影响标志位:</p>	<p>Rotate data memory left and place result in the accumulator</p> <p>将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。</p> <p><math>\text{ACC}.(i+1) \leftarrow [m].i \quad (i=0\sim6)</math></p> <p><math>\text{ACC}.0 \leftarrow [m].7</math></p> <p>无</p>
<p><b>RLC</b>      <b>[m]</b></p> <p>说明:</p> <p>运算过程:</p> <p>影响标志位:</p>	<p>Rotate data memory left through carry</p> <p>将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位。</p> <p><math>[m].(i+1) \leftarrow [m].i \quad (i=0\sim6)</math></p> <p><math>[m].0 \leftarrow C</math></p> <p><math>C \leftarrow [m].7</math></p> <p>C</p>
<p><b>RLCA</b>    <b>[m]</b></p> <p>说明:</p> <p>运算过程:</p> <p>影响标志位:</p>	<p>Rotate left through carry and place result in the accumulator</p> <p>将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。</p> <p><math>\text{ACC}.(i+1) \leftarrow [m].i \quad (i=0\sim6)</math></p> <p><math>\text{ACC}.0 \leftarrow C</math></p> <p><math>C \leftarrow [m].7</math></p> <p>C</p>

<b>RR</b>	[m]	Rotate data memory right
说明:		将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
运算过程:		$[m].i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $[m].7 \leftarrow [m].0,$
影响标志位:		无
<b>RRA</b>	[m]	Rotate right and place result in the accumulator
说明:		将指定数据存储器的内容循环右移 1 位, 第 0 位移到第 7 位, 移位结果存放到累加器, 而指定数据存储器的内容保持不变。
运算过程:		$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
影响标志位:		无
<b>RRC</b>	[m]	Rotate data memory right through carry
说明:		将指定数据存储器的内容连同进位标志右移 1 位, 第 0 位取代进位标志且原本的进位标志移到第 7 位。
运算过程:		$[m].i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位:		C
<b>RRCA</b>	[m]	Rotate right through carry and place result in the accumulator
说明:		将指定数据存储器的内容连同进位标志右移 1 位, 第 0 位取代进位标志且原本的进位标志移到第 7 位, 移位结果送回累加器, 但是指定数据寄存器的内容保持不变。
运算过程:		$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位:		C
<b>SBC</b>	A,[m]	Subtract data memory and carry from the accumulator
说明:		将累加器减去指定数据存储器的内容以及进位标志的反, 结果存放到累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:		$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位:		OV、Z、AC、C
<b>SBCM</b>	A,[m]	Subtract data memory and carry from the accumulator
说明:		将累加器减去指定数据存储器的内容以及进位标志的反, 结果存放到数据存储器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:		$[m] \leftarrow ACC - [m] - \bar{C}$
影响标志位:		OV、Z、AC、C
<b>SDZ</b>	[m]	Skip if decrement data memory is 0
说明:		将指定的数据存储器的内容减 1, 判断是否为 0, 若为 0 则跳过下一条指令, 由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:		$[m] \leftarrow [m] - 1,$ 如果[m]=0 跳过下一条指令执行
影响标志位:		无

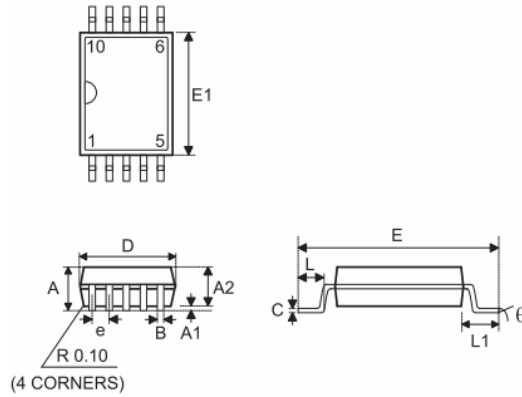
<b>SDZA</b> [m]	<b>Decrement data memory and place result in ACC,skip if 0</b>
说明:	将指定数据存储器内容减 1, 判断是否为 0, 如果为 0 则跳过下一条指令, 此结果将存放到累加器, 但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:	$ACC \leftarrow [m]-1$ , 如果 $ACC=0$ 跳过下一条指令执行
影响标志位:	无
<b>SET</b> [m]	<b>Set data memory</b>
说明:	将指定数据存储器的每一位设置为 1。
运算过程:	$[m] \leftarrow FFH$
影响标志位:	无
<b>SET</b> [m].i	<b>Set bit of data memory</b>
说明:	将指定数据存储器的第 i 位设置为 1。
运算过程:	$[m].i \leftarrow 1$
影响标志位:	无
<b>SIZ</b> [m]	<b>Skip if increment data memory is 0</b>
说明:	将指定的数据存储器的内容加 1, 判断是否为 0, 若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:	$[m] \leftarrow [m]+1$ , 如果 $[m]=0$ 跳过下一条指令执行
影响标志位:	无
<b>SIZA</b> [m]	<b>Increment data memory and place result in ACC,skip if 0</b>
说明:	将指定数据存储器的内容加 1, 判断是否为 0, 如果为 0 则跳过下一条指令, 此结果会被存放到累加器, 但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:	$ACC \leftarrow [m]+1$ , 如果 $ACC=0$ 跳过下一条指令执行
影响标志位:	无
<b>SNZ</b> [m].i	<b>Skip if bit I of the data memory is not 0</b>
说明:	判断指定数据存储器的第 i 位, 若不为 0, 则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果为 0, 则程序继续执行下一条指令。
运算过程:	如果 $[m].i \neq 0$ , 跳过下一条指令执行
影响标志位:	无
<b>SUB</b> A, [m]	<b>Subtract data memory from the accumulator</b>
说明:	将累加器的内容减去指定的数据存储器的数据, 把结果存放到累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:	$ACC \leftarrow ACC - [m]$
影响标志位:	OV、Z、AC、C
<b>SUBM</b> A, [m]	<b>Subtract data memory from the accumulator</b>
说明:	将累加器的内容减去指定数据存储器的数据, 结果存放到指定的数据存储器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:	$[m] \leftarrow ACC - [m]$
影响标志位:	OV、Z、AC、C

<b>SUB</b>	<b>A, x</b>	<p>Subtract immediate data from the accumulator</p> <p>说明：将累加器的内容减去立即数，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。</p> <p>运算过程：<math>ACC \leftarrow ACC - x</math></p> <p>影响标志位：OV、Z、AC、C</p>
<b>SWAP</b>	<b>[m]</b>	<p>Swap nibbles within the data memory</p> <p>说明：将指定数据存储器的低 4 位和高 4 位互相交换。</p> <p>运算过程：<math>[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4</math></p> <p>影响标志位：无</p>
<b>SWAPA</b>	<b>[m]</b>	<p>Swap data memory and place result in the accumulator</p> <p>说明：将指定数据存储器的低 4 位和高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。</p> <p>运算过程：<math>ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4</math>  <math>ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0</math></p> <p>影响标志位：无</p>
<b>SZ</b>	<b>[m]</b>	<p>Skip if data memory is 0</p> <p>说明：判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。</p> <p>运算过程：如果 <math>[m] = 0</math>，跳过下一条指令执行</p> <p>影响标志位：无</p>
<b>SZA</b>	<b>[m]</b>	<p>Move data memory to ACC, skip if 0</p> <p>说明：将指定数据存储器内容复制到累加器，并判断指定数据存储器的内容是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。</p> <p>运算过程：<math>ACC \leftarrow [m]</math>，如果 <math>[m] = 0</math>，跳过下一条指令执行</p> <p>影响标志位：无</p>
<b>SZ</b>	<b>[m]. i</b>	<p>Skip if bit I of the data memory is 0</p> <p>说明：判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。</p> <p>运算过程：如果 <math>[m].i = 0</math>，跳过下一条指令执行</p> <p>影响标志位：无</p>
<b>TABRDC</b>	<b>[m]</b>	<p>Move the ROM code(current page) to TBLH and data memory</p> <p>说明：将表格指针 TBLP 所指的程序代码低字节（当前页）移至指定的数据存储器且将高字节移至 TBLH。</p> <p>运算过程：<math>[m] \leftarrow</math> 程序代码（低字节）  <math>TBLH \leftarrow</math> 程序代码（高字节）</p> <p>影响标志位：无</p>
<b>TABRDL</b>	<b>[m]</b>	<p>Move the ROM code(last page) to TBLH and data memory</p> <p>说明：将表格指针 TBLP 所指的程序代码低字节（最后一页）移至指定的数据存储器且将高字节移至 TBLH。</p> <p>运算过程：<math>[m] \leftarrow</math> 程序代码（低字节）  <math>TBLH \leftarrow</math> 程序代码（高字节）</p> <p>影响标志位：无</p>

<b>XORA,</b> [m]	Logical XOR accumulator with data memory
说明:	将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "XOR" } [m]$
影响标志位:	Z
<b>XORM</b> A, [m]	Logical XOR data memory with accumulator
说明:	将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。
运算过程:	$[m] \leftarrow ACC \text{ "XOR" } [m]$
影响标志位:	Z
<b>XOR</b> A, x	Logical XOR immediate data to the accumulator
说明:	将累加器的数据与立即数逻辑异或，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "XOR" } x$
影响标志位:	Z

封装信息

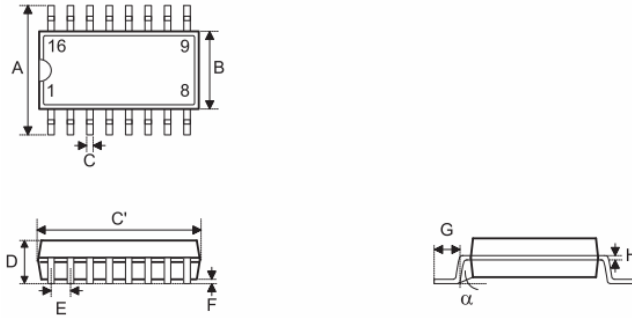
10-pin MSOP 外形尺寸



符号	尺寸(单位: mil)		
	最小	正常	最大
A	—	—	43
A1	0	—	6
A2	30	33	37
B	7	—	11
C	—	—	10
D	—	118	—
E	—	193	—
E1	—	118	—
e	—	20	—
L	16	24	31
L1	—	37	—
$\theta$	0°	—	8°

符号	尺寸(单位: mm)		
	最小	正常	最大
A	—	—	1.10
A1	0.0	—	0.15
A2	0.75	0.85	0.95
B	0.17	—	0.27
C	—	—	0.25
D	—	3.0	—
E	—	4.9	—
E1	—	3.0	—
e	—	0.5	—
L	0.4	0.6	0.8
L1	—	0.95	—
$\Theta$	0°	—	8°

16-pin NSOP(150mil)外形尺寸

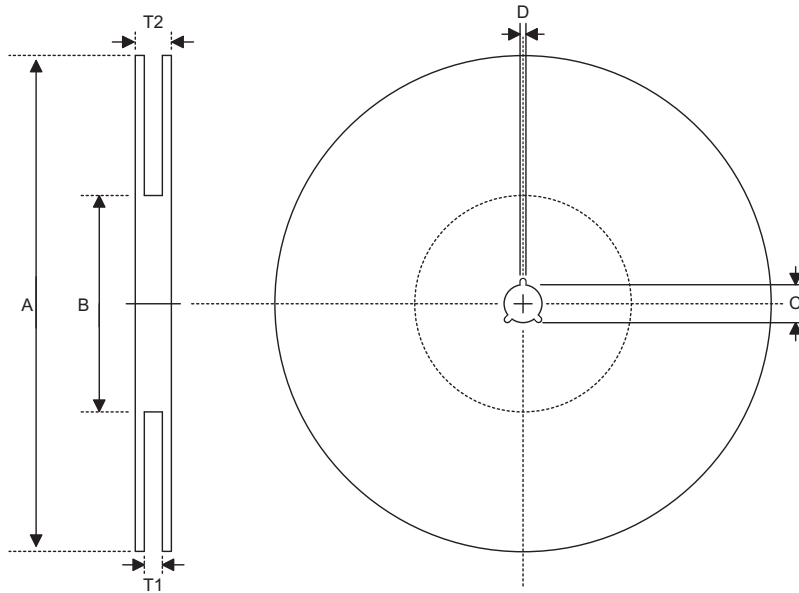


• MS-012

符号	尺寸(单位: mil)		
	最小	正常	最大
A	228	—	244
B	150	—	157
C	12	—	20
C'	386	—	394
D	—	—	69
E	—	50	—
F	4	—	10
G	16	—	50
H	7	—	10
$\alpha$	0°	—	8°

符号	尺寸(单位: mm)		
	最小	正常	最大
A	5.79	—	6.20
B	3.81	—	3.99
C	0.30	—	0.51
C'	9.80	—	10.01
D	—	—	1.75
E	—	1.27	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.18	—	0.25
$\alpha$	0°	—	8°

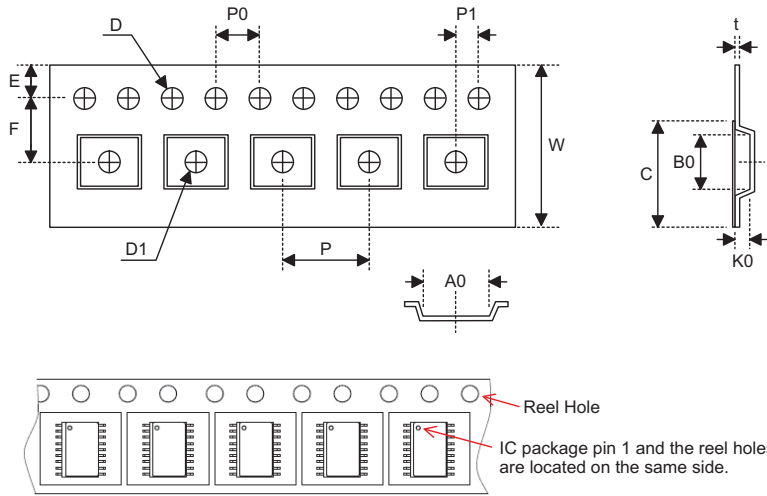
包装带和卷轴规格  
卷轴尺寸



SOP 16N (150mil)

符号	说明	尺寸 (单位: mm)
A	卷轴外圈直径	$330 \pm 1$
B	卷轴内圈直径	$100 \pm 1.5$
C	轴心直径	$13^{+0.5/-0.2}$
D	缝宽	$2 \pm 0.5$
T1	轮沿宽	$16.8^{+0.37/-0.2}$
T2	卷轴宽	$22.2 \pm 0.2$

运输带尺寸



SOP 16N(150mil)

符号	说明	尺寸 (单位: mm)
W	运输带宽	16.0±0.3
P	空穴间距	8.0±0.1
E	穿孔位置	1.75±0.1
F	空穴至穿孔距离(宽度)	7.5±0.1
D	穿孔直径	1.55 <sup>+0.10/-0.00</sup>
D1	空穴中之小孔直径	1.5 <sup>+0.25/-0.00</sup>
P0	穿孔间距	4.0±0.1
P1	空穴至穿孔距离(长度)	2.0±0.1
A0	空穴长	6.5±0.1
B0	空穴宽	10.3±0.1
K0	空穴深	2.1±0.1
T	传输带厚度	0.30±0.05
C	覆盖带宽度	13.3±0.1

**盛群半导体股份有限公司（总公司）**

新竹市科学工业园区研新二路 3 号

电话: 886-3-563-1999

传真: 886-3-563-1189

网站: [www.holtek.com.tw](http://www.holtek.com.tw)

**盛群半导体股份有限公司（台北业务处）**

台北市南港区园区街 3 之 2 号 4 楼之 2

电话: 886-2-2655-7070

传真: 886-2-2655-7373

传真: 886-2-2655-7383 (International sales hotline)

**盛扬半导体有限公司（深圳业务处）**

深圳市南山区科技园科技中三路与高新中二道交汇处生产力大楼 A 单元五楼 518057

电话: 86-755-8616-9908, 86-755-8616-9308

传真: 86-755-8616-9722

**Holtek Semiconductor(USA), Inc.（北美业务处）**

46729 Fremont Blvd., Fremont, CA 94538, USA

电话: 1-510-252-9880

传真: 1-510-252-9885

网站: [www.holtek.com](http://www.holtek.com)

Copyright © 2010 by HOLTEK SEMICONDUCTOR INC.

使用指南中所出现的信息在出版当时相信是正确的，然而盛群对于说明书的使用不负任何责任。文中提到的应用目的仅仅是用来做说明，盛群不保证或表示这些没有进一步修改的应用将是适当的，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。盛群产品不授权使用于救生、维生从机或系统中做为关键从机。盛群拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 <http://www.holtek.com.tw>