经济 A/D 型 8-Bit OTP 单片机

HT46R004

版本 : V1.30　日期 : 2016-01-29

www.holtek.com

目录

特性

CPU 特性

- 工作电压：
  f_{SYS}=8MHz：2.3V~5.5V
- V_{DD}=5V，系统时钟为 8MHz 时，指令周期为 0.5μs
- 提供暂停和唤醒功能，以降低功耗
- 两种振荡模式：
  内部高速 RC – HIRC
  内部低速 RC – LIRC
- 内部集成 8MHz 振荡器，无需外接元件
- 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
- 4 层堆栈
- 位操作指令

周边特性

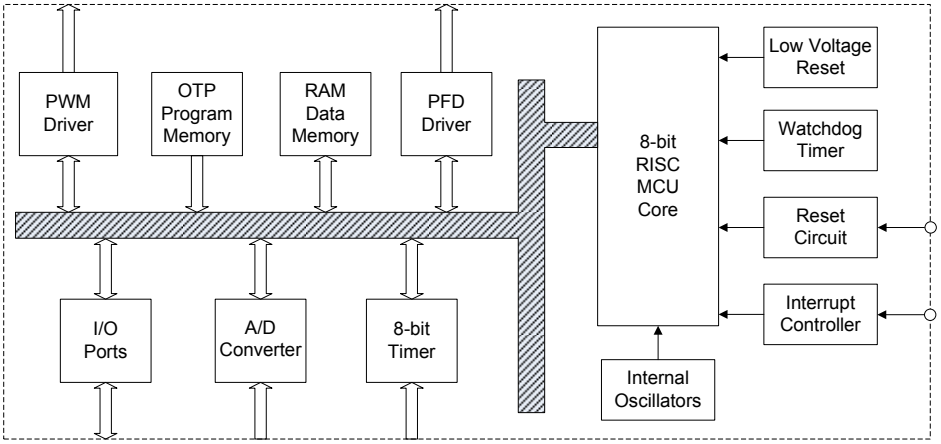
- OTP 程序存储器：2K×14
- RAM 数据存储器：96×8
- 看门狗定时器功能
- 18 个双向输入 / 输出口
- 8 通道 12 位分辨精度的 A/D 转换器
- 1 通道 8 位 PWM 功能
- 一个与 I/O 口共用引脚的外部中断输入
- 1 个 8 位可编程定时 / 计数器，具有溢出中断和预分频器功能
- 低电压复位功能
- 可编程分频器 — PFD
- 封装类型：16-pin DIP/NSOP, 20-pin DIP/SOP/NSOP

概述

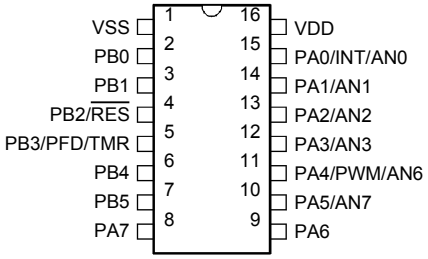
该单片机是一款具有 8 位高性能精简指令集的 OTP 单片机。该单片机具有功耗低、I/O 使用灵活、定时器功能、休眠和唤醒功能、看门狗以及成本低等优点，使该单片机可以广泛应用于如消费类产品、工业控制、子系统控制等方面。

方框图

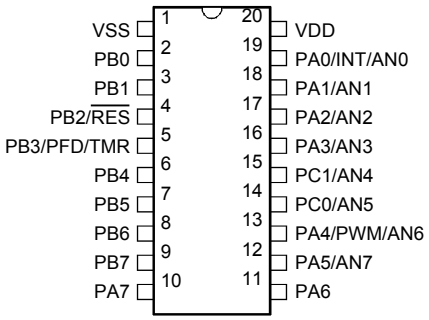
主要功能模块的方框图



引脚图



HT46R004
16 DIP-A/NSOP-A



HT46R004
20 DIP-A/NSOP-A/SOP-A

引脚说明

引脚名称	功能	OPT	I/T	O/T	说明
PA0/INT/AN0	PA0	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT	INTC0 CTRL0	ST	—	外部中断输入脚
	AN0	ADCR1	AI	—	A/D 通道 0
PA1/AN1	PA1	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	AN1	ADCR1	AI	—	A/D 通道 1
PA2/AN2	PA2	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	AN2	ADCR1	AI	—	A/D 通道 2
PA3/AN3	PA3	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	AN3	ADCR1	AI	—	A/D 通道 3
PA4/PWM/AN6	PA4	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	PWM	CTRL0	—	CMOS	PWM 输出
	AN6	ADCR1	AI	—	A/D 通道 6
PA5/AN7	PA5	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	AN7	ADCR1	AI	—	A/D 通道 7
PA6~PA7	PA6~PA7	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
PB0~PB1	PB0~PB1	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
PB2/$\overline{\text{RES}}$	PB2	—	ST	NMOS	通用 I/O 口
	$\overline{\text{RES}}$	—	ST	—	外部复位脚
PB3/PFD/TMR	PB3	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PFD	CTRL0	ST	—	PFD 输出脚
	TMR	TMR0C	ST	—	定时 / 计数器 0 输入脚
PB4~PB7	PB4~PB7	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
PC0/AN5	PC0	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN5	ADCR1	AI	—	A/D 通道 5
PC1/AN4	PC1	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN4	ADCR1	AI	—	A/D 通道 4
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）或者寄存器选项来配置
　　PWR：电源；　　NMOS：NMOS 输出
　　ST：施密特触发输入；　　CMOS：CMOS 输出　　AI：模拟输入

极限参数

电源供应电压 ..... V_{SS}-0.3V~V_{SS}+6.0V
输入电压 ..... V_{SS}-0.3V~V_{DD}+0.3V
储存温度 ..... -50°C~125°C
工作温度 ..... -40°C~85°C

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_{DD}	条件				
V_{DD}	工作电压（HIRC）	—	f_{SYS}=8MHz	2.3	—	5.5	V
I_{DD}	工作电流 (HIRC on)	3V	无负载，f_{SYS}=8MHz, ADC off	—	1.2	1.8	mA
		5V		—	2.4	3.6	mA
I_{STB1}	静态电流 (LIRC on)	3V	无负载，系统进入 HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	静态电流 (LIRC off)	3V	无负载，系统进入 HALT	—	—	1	μA
		5V		—	—	2	μA
V_{IL1}	输入 / 输出口、TMR 及 INT 脚的低电平输入电压	5V	—	0	—	1.5	V
		—		0	—	0.2V_{DD}	V
V_{IH1}	输入 / 输出口、TMR 及 INT 脚的高电平输入电压	5V	—	3.5	—	5	V
		—		0.8V_{DD}	—	V_{DD}	V
V_{IL2}	低电平输入电压 ($\overline{RES}$)	—	—	0	—	0.4V_{DD}	V
V_{IH2}	高电平输入电压 ($\overline{RES}$)	—	—	0.9V_{DD}	—	V_{DD}	V
V_{LVR}	低电压复位电压	—	V_{LVR}=2.10V	2.0	2.1	2.2	V
I_{OH}	输入 / 输出口源电流	3V	V_{OH}=0.9V_{DD}	-4	-8	—	mA
		5V		-8	-16	—	mA
I_{OL1}	输入 / 输出口灌电流	3V	V_{OL}=0.1V_{DD}	8	16	—	mA
		5V		16	32	—	mA
I_{OL2}	PB2 ($\overline{RES}$) 口灌电流	5V	V_{OL}=0.1V_{DD}	2	3	—	mA
R_{PH}	上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

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_{DD}	条件				
f_{SYS}	系统时钟	2.3V~5.5V	—		8		MHz
f_{HIRC}	系统时钟 (HIRC)	3/5V	—	-2%	8	+2%	MHz
		3/5V	Ta=0°C~70°C	-5%	8	+5%	MHz
		3.0~5.5V	Ta=0°C~70°C	-8%	8	+8%	MHz
		3.0~5.5V	Ta=-40°C~85°C	-12%	8	+12%	MHz
f_{TIMER}	定时器输入频率 (TMR)	3.3~5.5V	—	0	—	8	MHz
t_{WDTOSC}	看门狗振荡器时钟周期	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	外部复位低电平脉宽	—	—	1	—	—	μs
t_{RESE}	外部复位低电平脉宽 (with filter)	—	—	—	150	—	ns
t_{SST}	系统启动时间	—	从 halt 中唤醒	—	16	—	t_{SYS}
t_{LVR}	低电压复位时间	—	—	0.25	1	2	ms
t_{RSTD}	系统复位延迟时间 (All Reset)	—	—	25	50	100	ms

Note: 1. $t_{SYS}=1/f_{SYS}$

2. 为确保 HIRC 振荡器频率的精度，需在 VDD 和 VSS 脚间连接一个 0.1μF 去耦电容，并且尽量靠近单片机。

ADC 特性

Ta=25°C

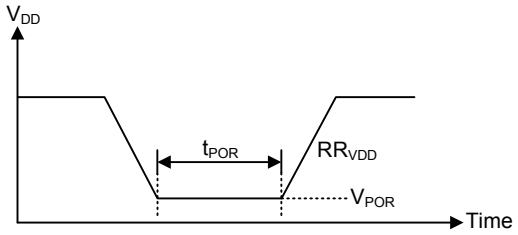
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
AV_{DD}	ADC 工作电压	—	$V_{REF}=AV_{DD}$	2.7	—	5.5	V
V_{AD}	AD 输入电压	—	—	0	—	AV_{DD}/V_{REF}	V
DNL	A/D 非线性微分误差	2.7V	$V_{REF}=AV_{DD}=V_{DD}$ $t_{AD}=0.5\mu s$	-2	—	+2	LSB
		3V					
		5V					
INL	A/D 非线性积分误差	2.7V	VREF=$AV_{DD}=V_{DD}$ $t_{AD}=0.5\mu s$	-4	—	+4	LSB
		3V					
		5V					
I_{ADC}	打开 A/D 增加的功耗	3V	无负载 ($t_{AD}=0.5\mu s$)	—	0.5	—	mA
		5V	无负载 ($t_{AD}=0.5\mu s$)	—	0.6	—	mA
t_{AD}	A/D 时钟周期	2.7~5.5V	—	0.5	—	10	μs
t_{ADC}	A/D 转换时间 (包括 A/D 采样和保持时间)	2.7~5.5V	12 位 ADC	—	16	—	t_{AD}
t_{ON2ST}	A/D 开启到 A/D 开始工作的时间	2.7~5.5V	—	2	—	—	μs

注：ADC 转换时间 (t_{AD})=n(ADC 位数)+4(取样时间)，每位的转换时间为一个 ADC 时钟 (t_{AD})。

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_{DD}	条件				
V_{POR}	上电复位电压	—	—	—	—	100	mV
RRV_{DD}	上电复位电压速率	—	—	0.035	—	—	V/ms
t_{POR}	V_{DD} 保持为 V_{POR} 的最小时间	—	—	1	—	—	ms

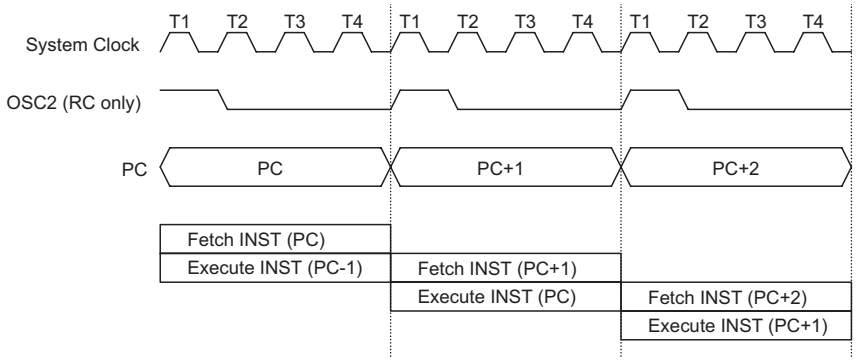


系统结构

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度和灵活性的 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

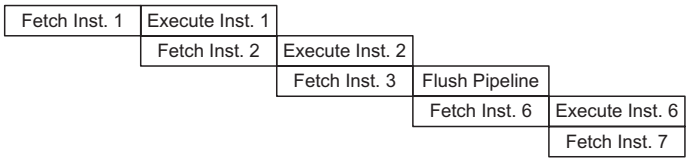
主系统时钟由 HIRC 振荡器提供，它被细分为 T1~T4 四个内部产生的非重叠时序。在 T1 时间，程序计数器自动加一并抓取一条新的指令。剩下的时间 T2~T4 完成译码和执行功能，因此，一个 T1~T4 时钟周期构成一个指令周期。虽然指令的抓取和执行发生在连续的指令周期，但单片机流水线结构会保证指令在一个指令周期内被有效执行。除非程序计数器的内容被改变，如子程序的调用或跳转，在这种情况下指令将需要多一个指令周期的时间去执行。



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。

```
1            MOV A,[12H]
2            CALL DELAY
3            CPL [12H]
4            :
5            :
6  DELAY:    NOP
```



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。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。

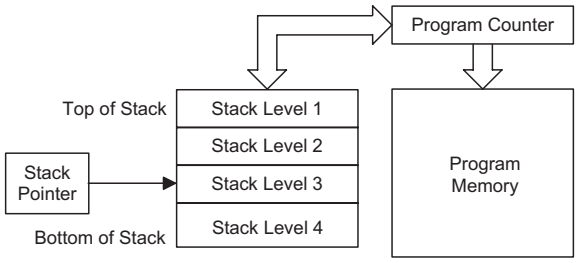
程序计数器	
程序计数器高字节	**PCL 寄存器**
PC10~PC8	PCL7~PCL0

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。该单片机有 4 层堆栈，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、借位或其它状态的改变，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：

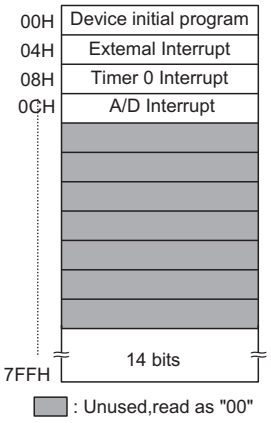
- 算术运算：ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA
- 逻辑运算：AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA
- 移位运算：RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC
- 递增和递减：INCA，INC，DECA，DEC
- 分支判断：JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI

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×14 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。



程序存储器结构

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。

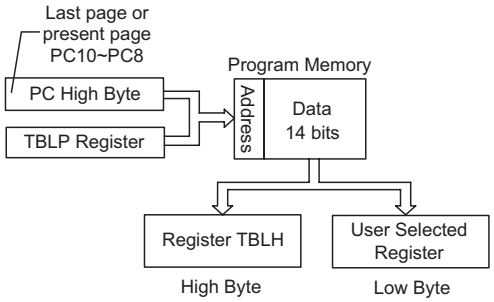
- 复位向量
  该向量是保留用做单片机复位后的程序起始地址。在芯片初始化后，程序将会跳转到这个地址并开始执行。
- 外部中断向量
  该向量为外部中断服务程序使用。当外部中断引脚发生边沿跳变时，如果中断允许且堆栈未满，则程序会跳转到该地址开始执行。外部中断有效边沿转换类型由 CTRL0 寄存器指定设定是高到低，还是低到高有效或者两者都可以触发。
- 定时 / 计数器 0 中断向量
  该内部中断向量为定时 / 计数器使用，当定时 / 计数器发生溢出，如果中断允许且堆栈未满，则程序会跳转到该地址开始执行。
- A/D 转换中断向量
  该内部向量为 A/D 转换器使用。当 A/D 转换完成时，如果中断允许且堆栈未满，则程序将跳转到该地址开始执行。

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，表格指针必须先行设定，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“706H”，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。

表格读取程序举例

```
tempreg1 db ?       ; temporary register #1
tempreg2 db ?       ; temporary register #2
     :
     :
mov a,06h           ; initialize table pointer - note that this address
                    ; is referenced
mov tblp, a         ; to the last page or present page
     :
     :
tabrdl tempreg1     ; transfers value in table referenced by table pointer
                    ; to tempreg1
                    ; data at prog. memory address "0706H" transferred to
                    ; to tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrdl tempreg2     ; transfers value in table referenced by table pointer
                    ; to tempreg2
                    ; data at prog. memory address "0705H" transferred to
                    ; tempreg2 and TBLH
                    ; in this example the data "1AH" is transferred to
                    ; tempreg1 and data "0FH" to register tempreg2
                    ; the value "00H" will be transferred to the high byte
                    ; register TBLH
     :
     :
org 0700h           ; sets initial address of last page
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
     :
     :
```

数据存储器

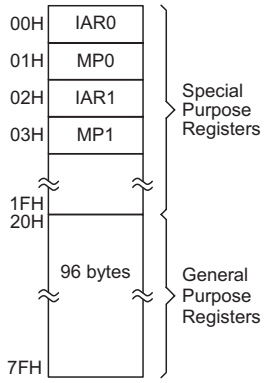
数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

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，数据存储器的开始地址是“00H”。

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区进行位操作，少数专用位除外。数据存储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。

00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	EXTRESB
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	WDTS
0AH	STATUS
0BH	INTC0
0CH	TMR0
0DH	TMR0C
0EH	ADRL
0FH	ADRH
10H	PA
11H	PAC
12H	PAPU
13H	PAWU
14H	PB
15H	PBC
16H	PBPU
17H	PC
18H	PCC
19H	PCPU
1AH	CTRL0
1BH	ACSR
1CH	WDTLVRC
1DH	ADCR0
1EH	ADCR1
1FH	PWM

特殊数据存储器

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。

系统控制寄存器

该寄存器是用来控制各种内部功能，如 PFD 控制、PWM 控制、外部中断边沿触发类型。

CTRL0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	INTES1	INTES0	PWMSEL	—	PWMC	PFDC	—	—
R/W	R/W	R/W	R/W	—	R/W	R/W	—	—
POR	1	0	0	—	0	0	—	—

Bit 7~6 **INTES1，INTES0**：外部中断边沿类型选择位
00：除能
01：上升沿触发
10：下降沿触发
11：双沿触发

Bit 5 **PWMSEL**：PWM 类型选择位
0：6+2
1：7+1

Bit 4 未使用，读为 "0"

Bit 3 **PWMC**：I/O or PWM 选择位
0：PA4
1：PWM

Bit 2 **PFDC**：I/O or PFD 选择位
0：PB3
1：PFD

Bit 1~0 未使用，读为 "0"

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。

振荡器概述

振荡器除了作为系统时钟源，还有作为看门狗定时器的时钟源。

类型	名称	频率
内部高速 RC	HIRC	8MHz
内部低速 RC	LIRC	12kHz

振荡器类型

系统时钟配置

此单片机提供一个内部 8MHz 振荡器 HIRC 作为系统时钟源及一个内部 12kHz 振荡器 LIRC 作为 WDT 的时钟源。

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 振荡器具有一种固定的频率：8MHz。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 V_{DD}、温度以及芯片制成工艺不同的影响减至最低程度。注意的是，此内部 RC 时钟无需额外的引脚接其它元件。

内部 12kHz 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 12kHz。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，可通过关闭 WDT 功能关闭 LIRC 以降低功耗。

暂停模式和唤醒

暂停模式

Holtek 所有的单片机都有进入暂停模式（即 HALT Mode 或 Sleep Mode）的能力。当单片机进入该模式后，正常工作电流将减小至一个极低的静态电流水平，这是由于当单片机进入暂停模式后，系统振荡器停止运行，由此降低功耗。但是，当单片机保持当前内部条件时，系统可在稍后的一段时间内唤醒并且继续运行，无需完全复位的动作。该特性在 MCU 须具备电源连续供电以保持单片机处于可知状态的下的应用方面是极为重要的。

进入暂停模式

暂停模式是由 HALT 指令来实现的，暂停模式时系统状态如下：

- 系统振荡器停止运行，应用程序将停止在“HALT”指令。
- RAM 和寄存器内容保持不变。
- 如果 WDT 时钟源来自 LIRC 振荡器，WDT 被清除并重新开始计数。
- 所有输入 / 输出口都保持其当前状态。
- 置位 PDF 标志，清除 TO 标志。

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。

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 $\overline{\text{RES}}$ 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 PAWU 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
	HIRC，LIRC
外部 $\overline{RES}$	$t_{RSTD}+t_{SST}$
PA 口	t_{SST}
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 50ms
3. t_{SST}=16 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。

看门狗定时器时钟源

看门狗定时器的时钟源可来自内部低频时钟 LIRC 振荡器、系统时钟 f_{SYS} 或 $f_{SYS}/4$。

看门狗定时器的时钟源可分频为 2^8~2^{15} 以提供更大的溢出周期，分频比由 WDTS 寄存器中的 WS2~WS0 位来决定。电压为 5V 时内部振荡器 LIRC 的周期大约为 12kHz。需要注意的是，这个特殊的内部时钟周期随 V_{DD}、温度和制成的不同而变化。WDT 可通过 WDTLVRC 寄存器中的 WDTEN2~WDTEN0 位控制使能或除能。

看门狗定时器控制寄存器

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　未使用，读为“0”

Bit 2~0　**WS2~WS0**：WDT 溢出周期选择位
000：$2^8/f_S$
001：$2^9/f_S$
010：$2^{10}/f_S$
011：$2^{11}/f_S$
100：$2^{12}/f_S$
101：$2^{13}/f_S$
110：$2^{14}/f_S$
111：$2^{15}/f_S$
这三位控制 WDT 时钟源的分频比，从而实现对 WDT 溢出周期的控制。

WDTLVRC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	WDTCLS1	WDTCLS0	LVREN2	LVREN1	LVREN0	WDTEN2	WDTEN1	WDTEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6　**WDTCLS1~WDTCLS0**：WDT/Timer 时钟源选择位
00：f_{LIRC}
01：$f_{SYS}/4$
10：f_{SYS}
11：f_{SYS}

Bit 5~3　详见其他章节

Bit 2~0　**WDTEN2~WDTEN0**：WDT 使能控制位
000：使能
101：除能
其它值：MCU 复位

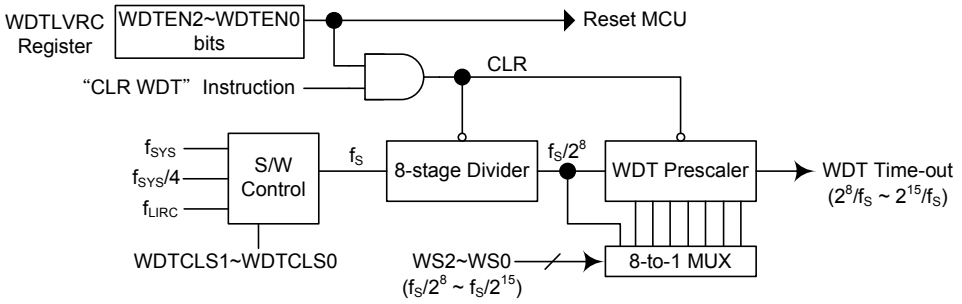
看门狗定时器操作

当 WDT 溢出时，它产生一个芯片复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗溢出前有策略地清看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。注意，如果看门狗定时器功能除能，那么执行任何看门狗定时器相关指令都是无效的。

通过设置 WDTLVRC 寄存器中的 WDTEN2~WDTEN0 位为 101B 可关闭 WDT 功能，设置这三位为 000B 将使能 WDT 功能。如果写入除此两组数据之外的值，MCU 将复位。

看门狗定时器的时钟源可通过 WDTLVRC 寄存器中的 WDTCLS1~WDTCLS0 选择来自内部低频时钟 LIRC 振荡器、系统时钟 f_{SYS} 或 $f_{SYS}/4$。值得注意的是，当系统进入暂停模式时，指令时钟停止运行，若此时 WDT 时钟源来自 f_{SYS} 或 $f_{SYS}/4$，WDT 也将停止工作。在比较恶劣的环境中，建议使用 LIRC 作为 WDT 的时钟源。分频比由 WDTS 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 1:32768，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 ($\overline{RES}$ 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。使用软件指令是执行“CLR WDT”指令清除 WDT 的内容。



看门狗定时器

复位和初始化

复位功能是任何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，$\overline{\text{RES}}$ 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 $\overline{\text{RES}}$ 脚拉低复位方式相似。

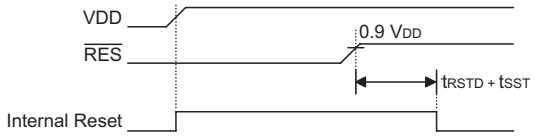
复位功能

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，内部 RC 振荡可能导致芯片复位不良，所以推荐使用和 $\overline{\text{RES}}$ 引脚连接的外部 RC 电路。由 RC 电路所造成的时间延迟使得 $\overline{\text{RES}}$ 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。$\overline{\text{RES}}$ 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。

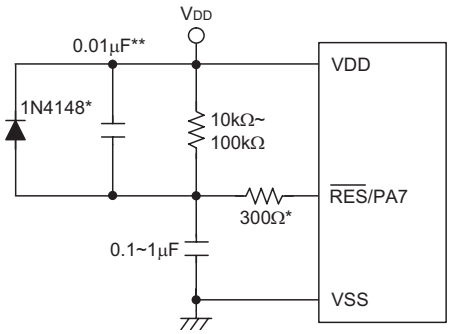


注：t_{RSTD} 为上电延迟时间，典型值为 50ms

上电复位时序图

在许多应用场合，可以在 VDD 和 $\overline{\text{RES}}$ 之间接入一个电阻，在 VSS 与 $\overline{\text{RES}}$ 之间接入一个电容作为外部复位电路。与 $\overline{\text{RES}}$ 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。

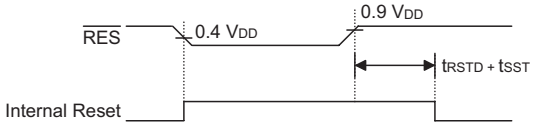
外部 RES 电路

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

RES 引脚复位

由于复位引脚与 PB.2 共用，复位功能必须使用 EXTRESB 寄存器的 RESBEN2~RESBEN0 位选择。

当单片机正常工作时，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 50ms。

RES 复位时序图

- **EXTRESB 寄存器**

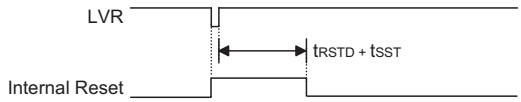
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	RESBEN2	RESBEN1	RESBEN0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3　未使用，读为“0”

Bit 2~0　**RESBEN2~RESBEN0**：PB2/RES 选择位
000：PB2
101：RES
其它值：MCU 复位

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，LVR 功能的使能与除能通过 WDTLVRC 寄存器中的 LVREN2~LVREN0 位控制，复位电压固定在 2.1V。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~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~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。



注：t_{RSTD} 为上电延迟时间，典型值为 50ms。

低电压复位时序图

- **WDTLVRC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	WDTCLS1	WDTCLS0	LVREN2	LVREN1	LVREN0	WDTEN2	WDTEN1	WDTEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

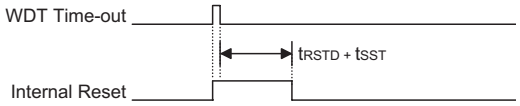
Bit 7~6　详见其他章节

Bit 5~3　**LVREN2~LVREN0**：LVR 使能控制位
000：使能
101：除能
其它值：MCU 位

Bit 2~0　详见其他章节

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 $\overline{RES}$ 复位相同。

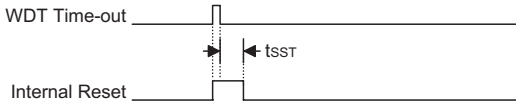


注：t_{RSTD} 为上电延迟时间，典型值为 50ms。

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_{SST} 的详细说明请参考交流电气特性。



注：t_{SST} 为 16 个 t_{SYS} 的时间。

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	上电复位
u	u	正常模式时的 $\overline{\text{RES}}$ 复位或 LVR 复位
1	u	正常模式时的 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。

寄存器	上电复位	$\overline{\text{RES}}$ 复位 (正常工作)	$\overline{\text{RES}}$ 复位 (HALT)	WDT 溢出复位 (正常工作)	WDT 溢出复位 (HALT)*
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
MP0	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
WDTS	---- -111	---- -111	---- -111	---- -111	---- -uuu
STATUS	--00 xxxx	--uu uuuu	--01 uuuu	--1u uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 -000	-000 0000	--uu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0-00	0000 0-00	0000 0-00	0000 0-00	uuuu u-uu
PC	---- --11	---- --11	---- --11	---- --11	---- --uu
PCC	---- --11	---- --11	---- --11	---- --11	---- --uu
PCPU	---- --00	---- --00	---- --00	---- --00	---- --uu
CTRL0	100- 00--	100- 00--	100- 00--	100- 00--	uuu- uu--
WDTLVRC	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR0	01-- -000	01-- -000	01-- -000	01-- -000	uu-- -uuu
ADCR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACSR	10-- -000	10-- -000	10-- -000	10-- -000	uu-- -uuu
EXTRESB	---- -000	---- -000	---- -000	---- -000	---- -uuu

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 / 输出口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。

该单片机提供 PA~PC 双向输入 / 输出口。这些寄存器在数据存储器有特定的地址。所有 I/O 口用于输入输出操作。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。

I/O 口寄存器列表

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	—	PBPU1	PBPU0
PC	—	—	—	—	—	—	PC1	PC0
PCC	—	—	—	—	—	—	PCC1	PCC0
PCPU	—	—	—	—	—	—	PCPU1	PCPU0

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~PCPU 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。注意，PB2 引脚没有上拉电阻功能。

PAPU 寄存器

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PAPUn**：PA 上拉功能控制位
0：除能
1：使能

PBPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	—	PBPU1	PBPU0
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 2　　未定义，读为“0”

Bit7~3, 1~0　**PBPUn**：PB 上拉功能控制位
0：除能
1：使能

PCPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PCPU1	PCPU0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　　未定义，读为“0”

Bit 1~0　　**PCPUn**：PC 上拉功能控制位
0：除能
1：使能

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 PAWU 寄存器来单独选择是否具有唤醒功能。

PAWU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0　　**PAWUn**：PA 唤醒功能控制位
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~PCC，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 CMOS 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 CMOS 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。

PAC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PACn**：I/O 类型选择位
0：输出
1：输入

PBC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PBCn**：I/O 类型选择位
0：输出
1：输入

PCC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PCC1	PCC0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	1	1

Bit 7~2 未定义，读为“0”

Bit 1~0 **PCCn**：I/O 类型选择位
0：输出
1：输入

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，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 PAC0 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 TMR 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

  此单片机提供 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

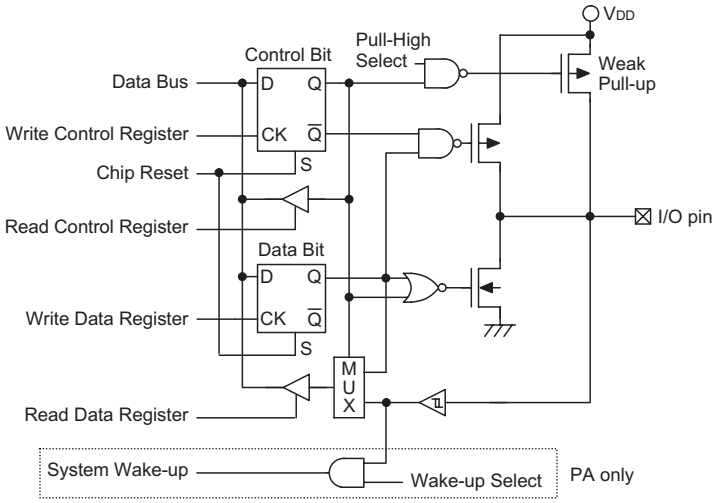
  此单片机提供 PWM 功能，与输入 / 输出引脚共用。PWM 输出功能通过 CTRL0 寄存器来设置。注意端口控制寄存器相应的位需要设置为输出，才能使能 PWM 的输出。如果端口控制寄存器被设置为输入，即使 PWM 寄存器已经使能 PWM 功能，该引脚都只作为普通逻辑输入脚，并且允许选择上拉电阻。

- A/D 输入引脚

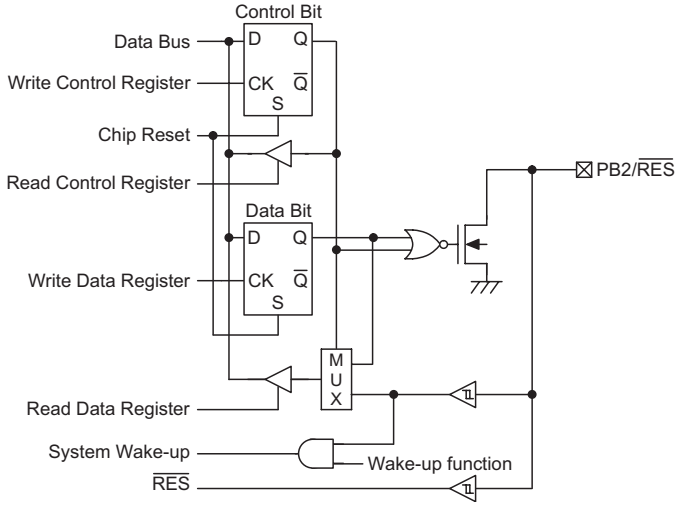
  此单片机具有 8 个 A/D 转换器输入。所有的模拟输入与 PA 引脚共用。如果需要将这些引脚用作为 A/D 输入而非 I/O 脚，则需要正确设定 A/D 转换控制寄存器 ADCR1 中相应的 PCRn 位。如果这些引脚作为输入 / 输出脚使用，仍可以通过寄存器选择是否要接上拉电阻，然而如果作为 A/D 输入使用，则这些引脚上的上拉电阻会自动断开。

输入 / 输出引脚结构

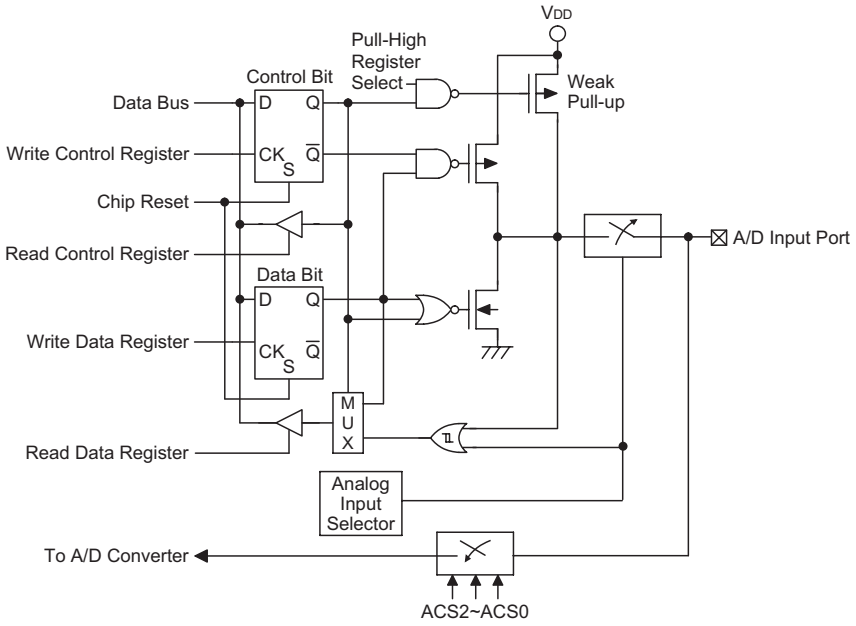
下图为输入 / 输出引脚的内部结构图。输入 / 输出引脚的准确逻辑结构图可能与此图不同，这里只是为了方便对 I/O 引脚功能的理解提供的一个参考。



通用输入 / 输出端口



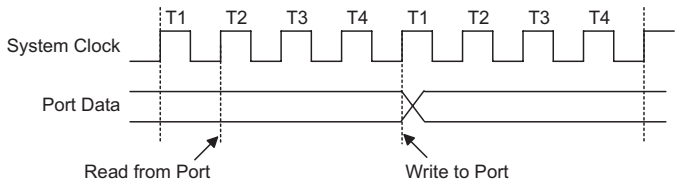
PB2 NMOS 输入 / 输出端口



A/D 输入 / 输出端口

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

PA 的每个引脚可通过 PAWU 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

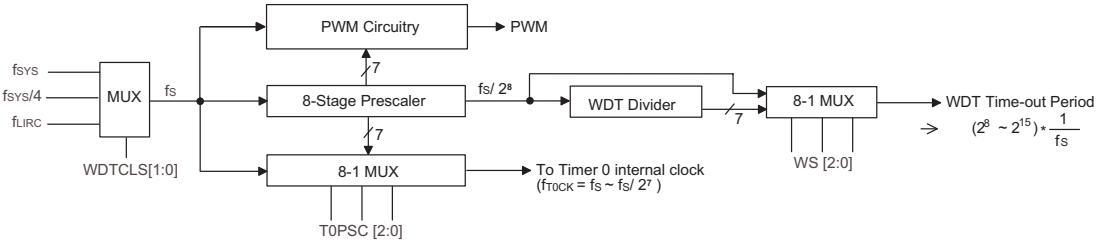
定时 / 计数器在任何单片机中都是一个很重要的部分，提供程序设计者一种实现和时间有关功能的方法。该单片机具有 1 个 8 位的向上计数器。每个定时 / 计数器有三种不同的工作模式，可以当作一个普通定时器、外部事件计数器或脉冲宽度测量使用。并且提供了一个内部时钟分频器，以扩大定时器的范围。

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。

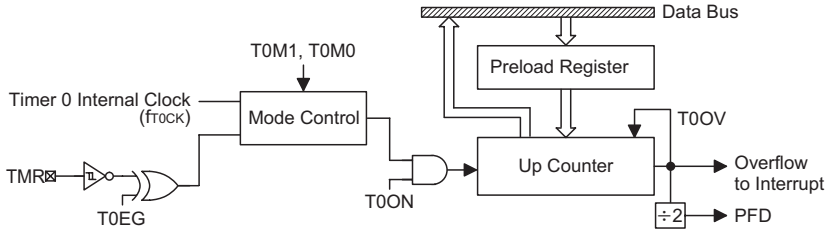
配置定时 / 计数器输入时钟源

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 WDTLVRC 寄存器的 WDTCLS1~WDTCLS0 位来选择 f_{SYS}，$f_{SYS}/4$ 或 LIRC 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 TMR 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 T0EG 位决定) 进行转换时，计数器增加一。



Timer/PWM/WDT 的时钟源结构图



8 位定时 / 计数器 0 结构图

定时 / 计数寄存器 – TMR0

定时 / 计数寄存器 TMR0，是位于特殊数据存储器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

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 TMR0C，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 T0M1/T0M0，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 T0ON，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，T0EG 位即 TMR0C 寄存器的第 3 位将可用来选择上升沿或下降沿触发。

TMR0C 寄存器

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	—	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	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 未使用，读为“0”

Bit 4 **T0ON**：定时 / 计数器使能
0：除能
1：使能

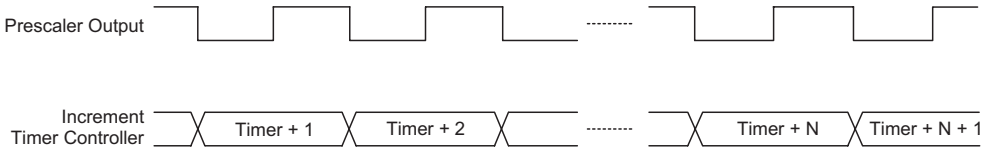
Bit 3 **T0EG：**
计数器有效边沿选择
0：在上升沿计数
1：在下降沿计数
脉冲宽度测量有效边沿选择
0：在下降沿启动计数，在上升沿停止计数
1：在上升沿启动计数，在下降沿停止计数

Bit 2~0 **T0PSC2, T0PSC1, T0PSC0：**选择定时器预分频比
定时器内部时钟 =
000：f_S
001：$f_S/2$
010：$f_S/4$
011：$f_S/8$
100：$f_S/16$
101：$f_S/32$
110：$f_S/64$
111：$f_S/128$

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，T0M1/T0M0 需要设置成 1 和 0。

在定时器模式中，f_{SYS}、$f_{SYS}/4$ 或 LIRC 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 T0PSC2~T0PSC0 位来确定。定时器控制寄存器第 4 位，即 T0ON 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 T0E 为 0，可以禁止计数器中断。



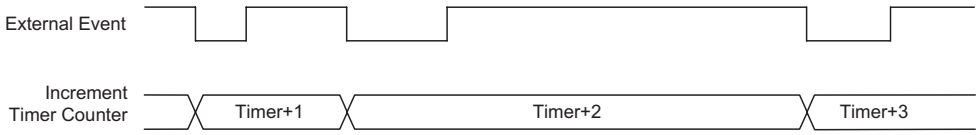
定时器模式时序图

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 TMR 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，T0M1/T0M0 需要设置成 0 和 1。

在外部事件计数模式，外部定时脚 TMR 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 T0ON 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 T0EG 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 T0EG 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 TMR 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（T0EG=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，T0M1/T0M0 需要设置 1 和 1。

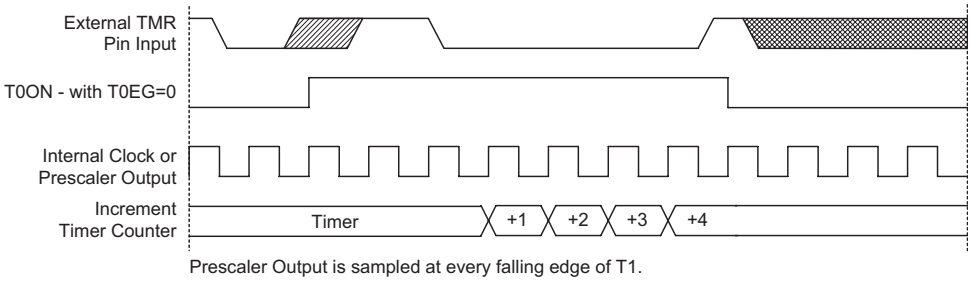
在脉冲宽度测量模式中，f_{SYS}，$f_{SYS}/4$ 或 LIRC 作为 8 位定时 / 计数器的内部时钟源，并可被预分频器进一步分频。分频比由预分频选择位 T0PSC2~T0PSC0，即定时控制寄存器的第 2~0 位来确定。在设置完定时 / 计数控制寄存器其它位，定时器控制寄存器第 4 位，即 T0ON 位需要设为逻辑高，才能使定时 / 计数器工作。然而，只有当在外部定时器引脚上接收到有效的逻辑转换时，定时 / 计数器才真正开始启动计数。

当定时控制寄存器第 3 位，即 T0EG 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 T0EG 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 TMR 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 TMR 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（T0EG=0）

预分频器

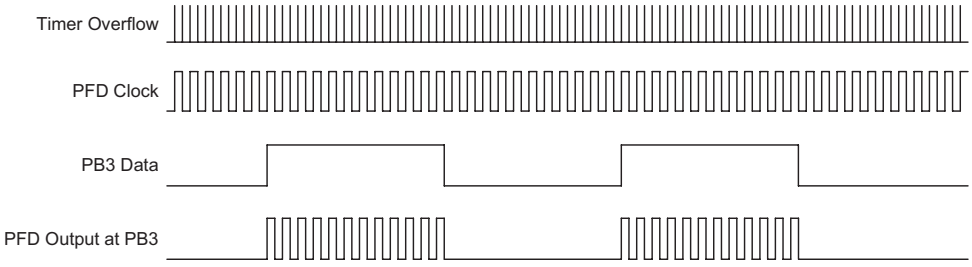
TMR0C 寄存器的 T0PSC0~T0PSC2 位用来确定定时 / 计数器的内部时钟的分频比，从而能够设置更长的定时器溢出周期。

PFD 功能

PFD（Programmable Frequency Divider）提供了一个可编程分频器，适用于像需要精确频率的场合。

PFD 功能的时钟源是定时 / 计数器溢出信号，由 CTRL0 中的 PFDC 来控制。该时钟源来自定时 / 计数器 0。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 PFD 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 PFD 功能，为了能对 PFD 输出进行操作，PB 控制寄存器 PBC 的设置是至关重要的，需要设置 PFD 引脚为输出。PB3 需要设置高来激活 PFD。输出数据位可以用来作为 PFD 输出的开关控制位。注意，如果输出数据位清零，PFD 输出将一直输出低。



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，单片机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
mov  a, 0c0H          ; select fSYS for the TMR0 clock source
mov  wdtlvrc, a
                      ; setup interrupt register
mov  a,05h            ; enable master interrupt and both timer interrupts
mov  intc0,a
:
:
set  tmr0c.4          ; start Timer 0
:
:
```

脉冲宽度调制 – PWM

此单片机提供一个 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 的时钟源来自 f_S，可选择为 f_{SYS}, $f_{SYS}/4$ 或 f_{LIRC}，通过 WDTLVRC 寄存器中的 WDTCLS1~WDTCLS0 位进行选择。

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。当 PWM 值为 8 位时，整个 PWM 周期的频率为 $f_S/256$。在 7+1 模式，PWM 调制频率将会是 $f_S/128$，在 6+2 模式，PWM 调制频率将会是 $f_S/64$。

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

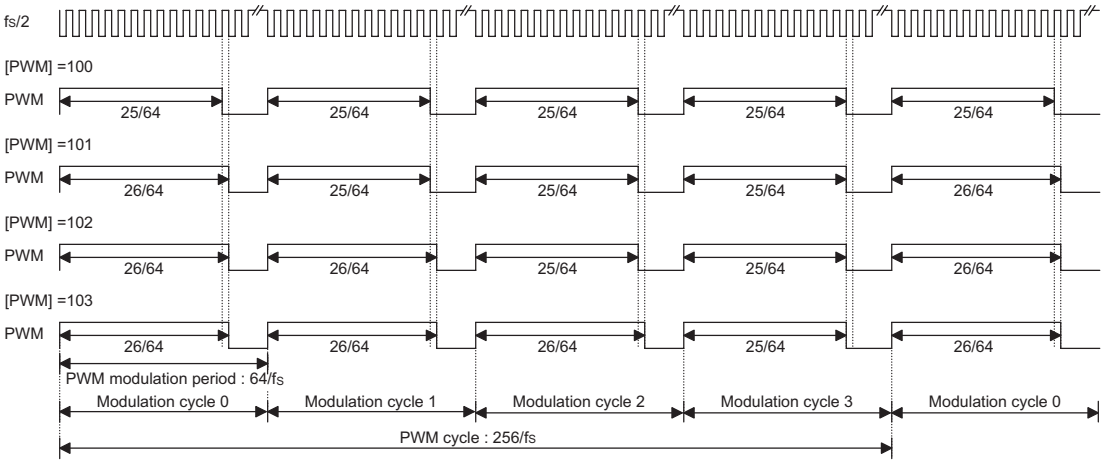
6+2 PWM 模式

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。

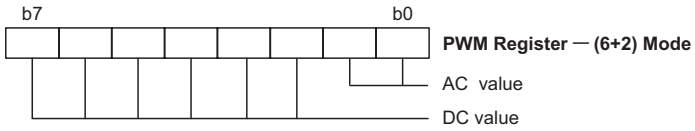
参数	AC (0~3)	DC (占空比)
调制周期 i （i=0~3）	i<AC	$\frac{DC+1}{64}$
	i ≥ AC	$\frac{DC}{64}$

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

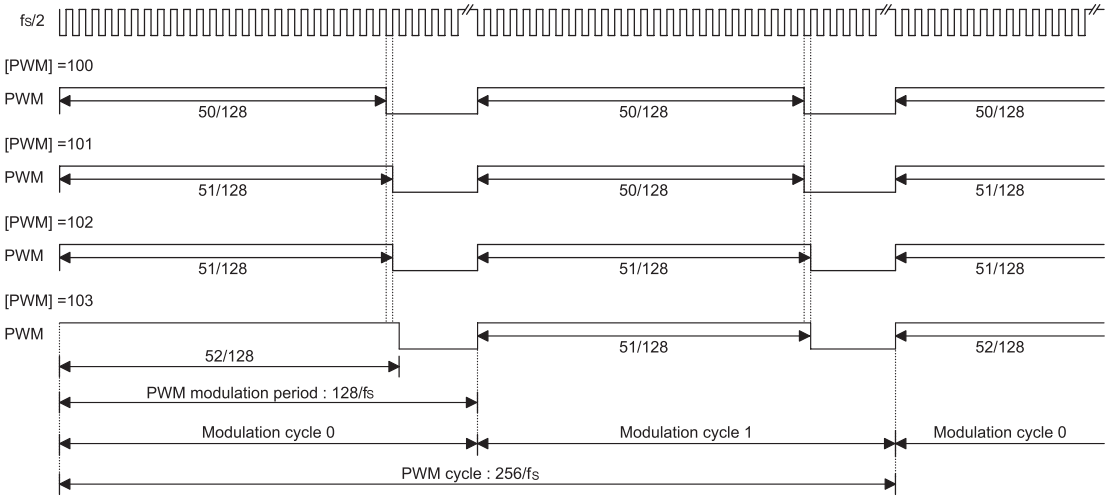
7+1 PWM 模式

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。

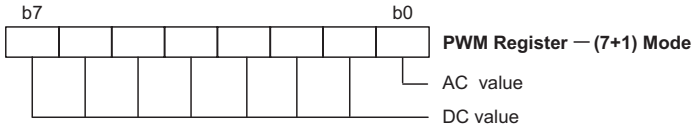
参数	AC (0~1)	DC (占空比)
调制周期 i（i=0~1）	i<AC	$\frac{DC+1}{128}$
	i ≥ AC	$\frac{DC}{128}$

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 PWM 输出

```
mov  a,64h                ; setup PWM value of decimal 100
mov  pwm,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
```

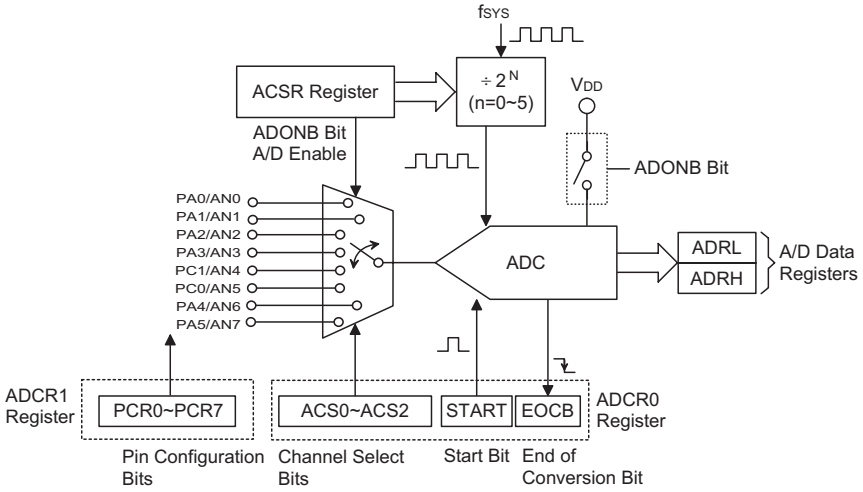
A/D 转换器

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

此单片机都包含一个 8 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) 并直接将这些信号转换成 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 换转数据结果位。

ADRH、ADRL 寄存器

Bit	ADRH								ADRL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	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 换转数据

A/D 转换控制寄存器 – ADCR0，ADCR1，ACSR

寄存器 ADCR0，ADCR1 和 ACSR 用来控制 A/D 转换器的功能和操作。这三个 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 ADCR0 包含 ACS2~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 8 个模拟输入中的每一个都需要分别被发送到转换器。ADCR0 寄存器中 ACS2~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。

ADCR1 寄存器中的 PCR7~PCR0 位，用来定义 PA0~PA5, PC0~PC1 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。如果 PCRn 位设置为“1”，则意味着对应的 ANn 脚设置为模拟输入脚。注意，如果 PCR7~PCR0 全都设为“0”，则 PA0~PA5, PC0~PC1 引脚都被设定为正常的 I/O，A/D 输入全部除能，同时 A/D 转换器电路也将随之关闭。

ADCR0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	START	EOCB	—	—	—	ACS2	ACS1	ACS0
R/W	R/W	R	—	—	—	R/W	R/W	R/W
POR	0	1	—	—	—	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”
此位用于初始化 A/D 转换过程。通常此位为低，但如果设为高再被清零，将初始化 A/D 转换过程。当此位为高，将重置 A/D 转换器。

Bit 6 **EOCB**：A/D 转换结束标志
0：A/D 转换结束
1：A/D 转换中
此位用于表明 A/D 转换过程的完成。当转换正在进行时，此位为高。

Bit 5~3 未定义，读为“0”

Bit 2~0 **ACS2，ACS1，ACS0**：选择 A/D 通道
000：AN0
001：AN1
010：AN2
011：AN3
100：AN4
101：AN5
110：AN6
111：AN7

ADCR1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PCR7	PCR6	PCR5	PCR4	PCR3	PCR2	PCR1	PCR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PCR7~PCR0**：A/D 通道配置
0：输入 / 输出口
1：模拟输入 ANn
如果 PCR0~PCR7 都为 0，则 ADC 模块关闭以降低功耗。

ACSR 寄存器

Bit	7	6	5	4	3	2	1	0
Name	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

A/D 操作

ADCR0 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，ADCR0 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换转器的开 / 关动作。

ADCR0 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 ADCR0 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 ADCR1 寄存器中的 PCR7~PCR0 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。

PCR7~PCR0	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} 的范围值为 0.5μs~10μs，当系统时钟速度超过 4MHz 时就必须小心。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 转换时钟周期不超出时钟周期的规定范围，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，因为它们的 A/D 转换时钟周期超过了规定值的范围。

f_{SYS}	A/D 时钟周期 (t_{AD})						
	ADCS2, ADCS1, ADCS0=000 ($f_{SYS}/2$)	ADCS2, ADCS1, ADCS0=001 ($f_{SYS}/8$)	ADCS2, ADCS1, ADCS0=010 ($f_{SYS}/32$)	ADCS2, ADCS1, ADCS0=100 (f_{SYS})	ADCS2, ADCS1, ADCS0=101 ($f_{SYS}/4$)	ADCS2, ADCS1, ADCS0=110 ($f_{SYS}/16$)	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 端口或 PC 端口的 I/O 引脚共用。使用 ADCR1 寄存器中的 PCR7~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 或 PC 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 PCR7~PCR0 位使能 A/D 输入时，不需要考虑端口控制寄存器的状态。

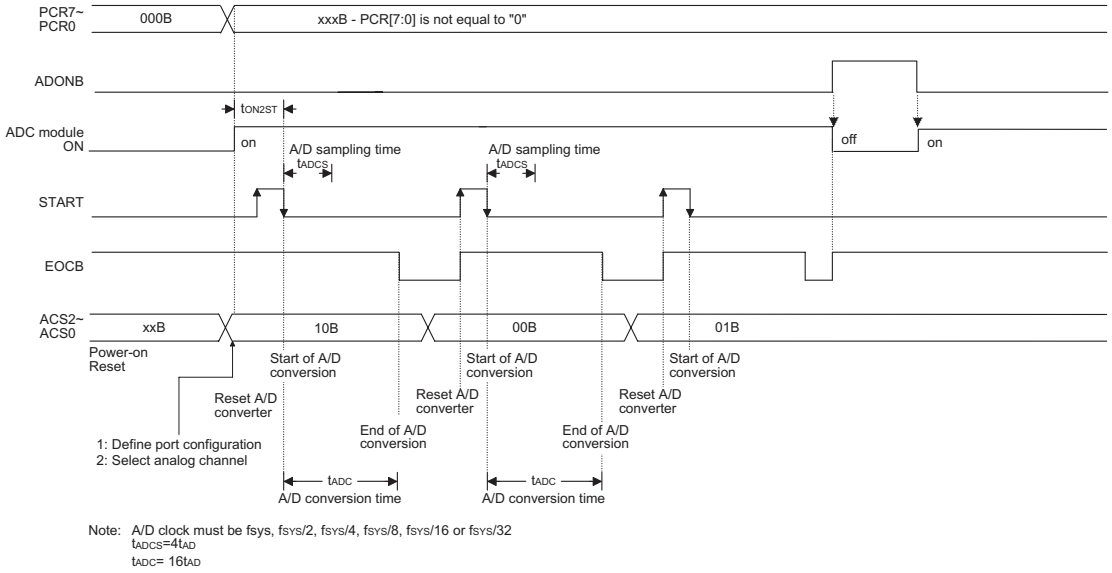
A/D 转换步骤

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 ADCR1 寄存器中的 PCR7~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 ADCR0 寄存器中的 ACS2~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 ADCR0 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 ADCR0 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满，则转换完成后，程序会进入 A/D 中断服务子程序。

注：若使用轮询 ADCR0 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。



Note: A/D clock must be fsys, fSYS/2, fSYS/4, fSYS/8, fSYS/16 or fSYS/32
$t_{ADCS}=4t_{AD}$
$t_{ADC}=16t_{AD}$

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。

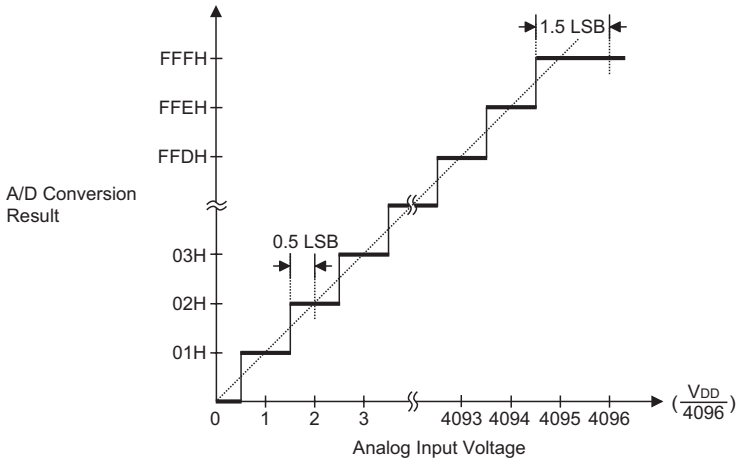
编程注意事项

在编程时，需要特别注意寄存器中的 PCR[7:0]。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 VDD 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 ADCR0 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,00000001B            ; setup ADCR1 register to configure Port as A/D
                            ; inputs
mov  ADCR1,a
mov  a, 00000000B           ; select AN0 to be connected to the A/D converter
mov  ADCR0, a

:
:
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  adrl_buffer,a          ; save result to user defined register
mov  a,ADRH                 ; read high byte conversion result value
mov  adrh_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 : 使用中断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,00000001B    ; setup ADCR register to configure Port as A/D inputs
mov  ADCR1,a
mov  a, 00000000B   ; select AN0 to be connected to the A/D converter
mov  ADCR0, a
:
:
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  adrl_buffer,a  ; save result to user defined register
mov  a,ADRH         ; read high byte conversion result value
mov  adrh_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)，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 A/D 转换器控制。

中断寄存器

所有中断允许和请求标志均由 INTC0 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，相应有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。

功能	使能位	请求标志位
总中断	EMI	—
INT 引脚	INTE	INTF
定时 / 计数器	T0E	T0F
A/D 转换器	ADE	ADF

INTC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	T0F	INTF	ADE	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　**ADF**：A/D 转换器中断请求标志
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　**ADE**：A/D 转换器中断使能
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

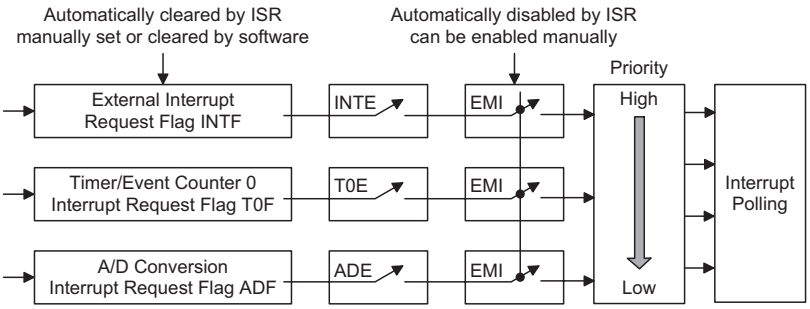
Bit 1　**INTE**：外部中断使能
0：除能
1：使能

Bit 0　**EMI**：总中断使能
0：除能
1：使能

中断操作

定时 / 计数器溢出、A/D 转换完成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。

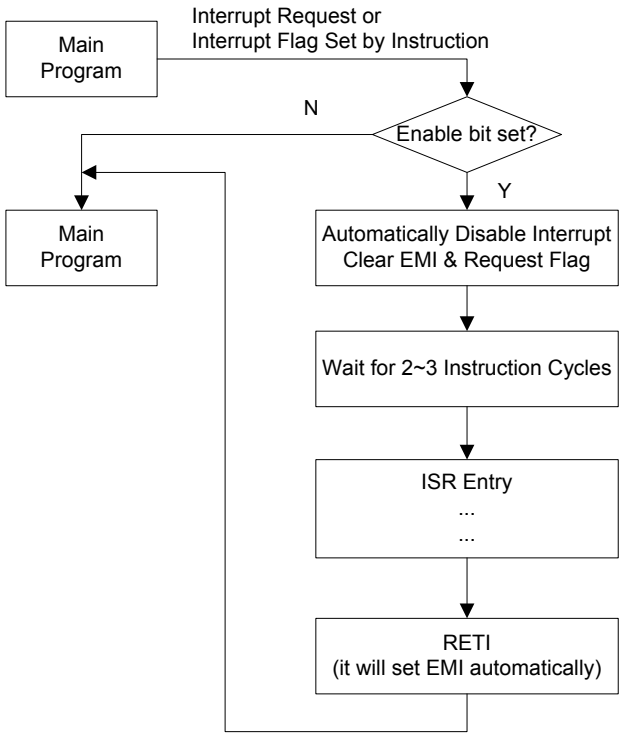
各个中断使能位以及相应的请求标志位，以优先级的顺序如下图所示。



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。

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
A/D 转换完成中断	3	0CH

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 INTES0 和 INTES1 位 (CTRL0 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。

INTES1	INTES0	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 PA0 共用引脚，如果 INTC0 中相应的外部中断使能位被置位并且在 CTRL0 寄存器中也设置了中断边沿触发类型，PA0 将只能被作为外部中断输入口使用，同时 PAC.0 需将 PA0 设为输入口。当中断使能、堆栈未满且外部中断产生时，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，PA0 依然可以设置带有上拉电阻功能。

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 T0E 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 T0F 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 T0F 被复位且 EMI 被清零以除能其它中断。

A/D 转换器中断

要使 A/D 转换中断发生，总中断控制位 EMI 和 A/D 中断使能位 ADE 必须先被置位。当 A/D 转换结束，A/D 中断请求标志 ADF 被置位，将会发生 A/D 中断。当中断使能，堆栈未满和 A/D 转换结束，将调用相应 A/D 中断子程序。当响应 A/D 转换中断服务子程序时，中断请求标志位 ADF 会被复位且 EMI 位会被清零以除能其它中断。

中断唤醒功能

单片机处于休眠模式时，每种中断都具有唤醒单片机的功能。当中断请求标志由“0”变为“1”时，单片机即可被唤醒，与中断使能位的状态无关。因此，即使单片机处于休眠模式，系统振荡器停止运行，若有如外部中断脚上的中断边沿信号产生、定时 / 计数器溢出发生时，其相应的中断请求标志位置位，单片机也可以被唤醒。值得注意的是，应避免伪唤醒条件的产生。若要除能相应中断的唤醒功能，需在单片机进入休眠模式前使其中断请求标志位置位。中断唤醒功能与中断使能位的状态无关。

编程注意事项

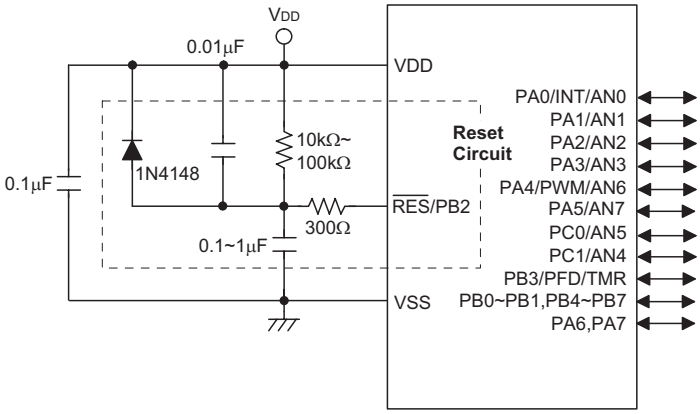
通过除能中断使能位，可以屏蔽中断请求。然而，一旦请求标志位被置位，它将保存在中断寄存器中，直到相应的中断被响应或被软件指令清除。

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。

若从中断子程序中返回可执行 RET 或 RETI 指令。除了能返回至主程序外，RETI 指令还能自动设置 EMI 位为高，允许进一步中断。RET 指令只能返回至主程序，清除 EMI 位，除能进一步中断。

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它们的应用。

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 0.5μs 中执行完成，而分支或调用操作则将在 1μs 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

提供数据存储器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出口的 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
算术运算			
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 注	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 注	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 注	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 注	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 注	C
逻辑运算			
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 注	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 注	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 注	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 注	Z
CPLA [m]	对数据存储器取反，结果放入 ACC	1	Z
递增和递减			
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 注	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 注	Z

助记符	说明	指令周期	影响标志位
移位			
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 注	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 注	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 注	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 注	C
数据传送			
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 注	无
MOV A, x	将立即数送至 ACC	1	无
位运算			
CLR [m].i	清除数据存储器的位	1 注	无
SET [m].i	置位数据存储器的位	1 注	无
转移			
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 注	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 注	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 注	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 注	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 注	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 注	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 注	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 注	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
查表			
TABRD [m]	读取特定页的 ROM 内容，并送至数据存储器和 TBLH	2 注	无
TABRDC [m]	读取当前页的 ROM 内容，并送至数据存储器和 TBLH	2 注	无
TABRDL [m]	读取最后页的 ROM 内容，并送至数据存储器和 TBLH	2 注	无
其它指令			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1 注	无
SET [m]	置位数据存储器	1 注	无

助记符	说明	指令周期	影响标志位
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 注	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 to ACC with Carry
指令说明 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。
功能表示 ACC ← ACC + [m] + C
影响标志位 OV、Z、AC、C

ADCM A, [m] Add ACC to Data Memory with Carry
指令说明 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。
功能表示 [m] ← ACC + [m] + C
影响标志位 OV、Z、AC、C

ADD A, [m] Add Data Memory to ACC
指令说明 将指定的数据存储器和累加器内容相加，结果存放到累加器。
功能表示 ACC ← ACC + [m]
影响标志位 OV、Z、AC、C

ADD A, x Add immediate data to ACC
指令说明 将累加器和立即数相加，结果存放到累加器。
功能表示 ACC ← ACC + x
影响标志位 OV、Z、AC、C

ADDM A, [m] Add ACC to Data Memory
指令说明 将指定的数据存储器和累加器内容相加，结果存放到指定的数据存储器。
功能表示 [m] ← ACC + [m]
影响标志位 OV、Z、AC、C

AND A, [m] Logical AND Data Memory to ACC
指令说明 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。
功能表示 ACC ← ACC “AND” [m]
影响标志位 Z

AND A, x	Logical AND immediate data to ACC
指令说明	将累加器中的数据和立即数做逻辑与，结果存放到累加器。
功能表示	ACC ← ACC“AND”x
影响标志位	Z

ANDM A, [m]	Logical AND ACC to Data Memory
指令说明	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。
功能表示	[m] ← ACC“AND”[m]
影响标志位	Z

CALL addr	Subroutine call
指令说明	无条件地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 地址并从新地址继续执行程序，由于此指令需要额外的运 算，所以为一个 2 周期的指令。
功能表示	Stack ← Program Counter + 1 Program Counter ← addr
影响标志位	无

CLR [m]	Clear Data Memory
指令说明	将指定数据存储器的内容清零。
功能表示	[m] ← 00H
影响标志位	无

CLR [m].i	Clear bit of Data Memory
指令说明	将指定数据存储器的 i 位内容清零。
功能表示	[m].i ← 0
影响标志位	无

CLR WDT	Clear Watchdog Timer
指令说明	WDT 计数器、暂停标志位 PDF 和看门狗溢出标志位 TO 清零。
功能表示	WDT cleared TO & PDF ← 0
影响标志位	TO、PDF

CLR WDT1 Preclear Watchdog Timer
指令说明 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。
功能表示 WDT ← 00H
TO & PDF ← 0
影响标志位 TO、PDF

CLR WDT2 Preclear Watchdog Timer
指令说明 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。
功能表示 WDT ← 00H
TO & PDF ← 0
影响标志位 TO、PDF

CPL [m] Complement Data Memory
指令说明 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 0 变 1。
功能表示 [m] ← $[\overline{m}]$
影响标志位 Z

CPLA [m] Complement Data Memory with result in ACC
指令说明 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 0 变 1，而结果被储存回累加器且数据存储器中的内容不变。
功能表示 ACC ← $[\overline{m}]$
影响标志位 Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

指令说明 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，结果存放到数据存储器。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。

功能表示 [m] ← ACC + 00H 或
[m] ← ACC + 06H 或
[m] ← ACC + 60H 或
[m] ← ACC + 66H

影响标志位 C

DEC [m] Decrement Data Memory

指令说明 将指定数据存储器内容减 1。

功能表示 [m] ← [m] – 1

影响标志位 Z

DECA [m] Decrement Data Memory with result in ACC

指令说明 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。

功能表示 ACC ← [m] – 1

影响标志位 Z

HALT Enter power down mode

指令说明 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。

功能表示 TO ← 0
PDF ← 1

影响标志位 TO、PDF

INC [m] Increment Data Memory

指令说明 将指定数据存储器的内容加 1。

功能表示 [m] ← [m] + 1

影响标志位 Z

INCA [m] Increment Data Memory with result in ACC
指令说明 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。
功能表示 ACC ← [m] + 1
影响标志位 Z

JMP addr Jump unconditionally
指令说明 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。
功能表示 Program Counter ← addr
影响标志位 无

MOV A, [m] Move Data Memory to ACC
指令说明 将指定数据存储器的内容复制到累加器。
功能表示 ACC ← [m]
影响标志位 无

MOV A, x Move immediate data to ACC
指令说明 将 8 位立即数载入累加器。
功能表示 ACC ← x
影响标志位 无

MOV [m], A Move ACC to Data Memory
指令说明 将累加器的内容复制到指定的数据存储器。
功能表示 [m] ← ACC
影响标志位 无

NOP No operation
指令说明 空操作，接下来顺序执行下一条指令。
功能表示 PC ← PC+1
影响标志位 无

OR A, [m] Logical OR Data Memory to ACC
指令说明 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。
功能表示 ACC ← ACC “OR” [m]
影响标志位 Z

OR A, x Logical OR immediate data to ACC
指令说明 将累加器中的数据和立即数逻辑或，结果存放到累加器。
功能表示 ACC ← ACC“OR”x
影响标志位 Z

ORM A, [m] Logical OR ACC to Data Memory
指令说明 将存在指定数据存储器中的数据和累加器逻辑或，结果放到数据存储器。
功能表示 [m] ← ACC“OR”[m]
影响标志位 Z

RET Return from subroutine
指令说明 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。
功能表示 Program Counter ← Stack
影响标志位 无

RET A, x Return from subroutine and load immediate data to ACC
指令说明 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。
功能表示 Program Counter ← Stack
ACC ← x
影响标志位 无

RETI Return from interrupt
指令说明 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。
功能表示 Program Counter ← Stack
EMI ← 1
影响标志位 无

RL [m] Rotate Data Memory left
指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。
功能表示 [m].(i+1) ← [m].i (i=0~6)
[m].0 ← [m].7
影响标志位 无

RLA [m] Rotate Data Memory left with result in ACC
指令说明 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。
功能表示 ACC.(i+1) ← [m].i (i=0~6)
ACC.0 ← [m].7
影响标志位 无

RLC [m] Rotate Data Memory Left through Carry
指令说明 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。
功能表示 [m].(i+1) ← [m].i (i=0~6)
[m].0 ← C
C ← [m].7
影响标志位 C

RLC A [m] Rotate Data Memory left through Carry with result in ACC
指令说明 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。
功能表示 ACC.(i+1) ← [m].i (i=0~6)
ACC.0 ← C
C ← [m].7
影响标志位 C

RR [m] Rotate Data Memory right
指令说明 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。
功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7 ← [m].0
影响标志位 无

RRA [m] Rotate Data Memory right with result in ACC
指令说明 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。
功能表示 ACC.i ← [m].(i+1) (i=0~6)
ACC.7 ← [m].0
影响标志位 无

RRC [m] Rotate Data Memory right through Carry

指令说明 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。

功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7 ← C
C ← [m].0

影响标志位 C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

指令说明 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。

功能表示 ACC.i ← [m].(i+1) (i=0~6)
ACC.7 ← C
C ← [m].0

影响标志位 C

SBC A, [m] Subtract Data Memory from ACC with Carry

指令说明 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。

功能表示 ACC ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

SBCM A, [m] Subtract Data Memory from ACC with Carry and result in Data Memory

指令说明 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。

功能表示 [m] ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

SDZ [m] Skip if Decrement Data Memory is 0

指令说明 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。

功能表示 [m] ← [m] -1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

SDZA [m] Decrement data memory and place result in ACC,skip if 0
指令说明 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。
功能表示 ACC ← [m]-1，如果 ACC=0 跳过下一条指令执行
影响标志位 无

SET [m] Set Data Memory
指令说明 将指定数据存储器的每一位设置为 1。
功能表示 [m] ← FFH
影响标志位 无

SET [m].i Set bit of Data Memory
指令说明 将指定数据存储器的第 i 位置位为 1。
功能表示 [m].i ← 1
影响标志位 无

SIZ [m] Skip if increment Data Memory is 0
指令说明 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。
功能表示 [m] ← [m]+1，如果 [m]=0 跳过下一条指令执行
影响标志位 无

SIZA [m] Skip if increment Data Memory is zero with result in ACC
指令说明 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。
功能表示 ACC ← [m]+1，如果 ACC=0 跳过下一条指令执行
影响标志位 无

SNZ [m].i Skip if bit i of Data Memory is not 0
指令说明 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。
功能表示 如果 [m].i≠0，跳过下一条指令执行
影响标志位 无

SUB A, [m] Subtract Data Memory from ACC
指令说明 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。
功能表示 ACC ← ACC – [m]
影响标志位 OV、Z、AC、C、SC、CZ

SUBM A, [m] Subtract Data Memory from ACC with result in Data Memory
指令说明 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。
功能表示 [m] ← ACC – [m]
影响标志位 OV、Z、AC、C、SC、CZ

SUB A, x Subtract immediate Data from ACC
指令说明 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。
功能表示 ACC ← ACC – x
影响标志位 OV、Z、AC、C、SC、CZ

SWAP [m] Swap nibbles of Data Memory
指令说明 将指定数据存储器的低 4 位和高 4 位互相交换。
功能表示 [m].3~[m].0 ↔ [m].7~[m].4
影响标志位 无

SWAPA [m] Swap nibbles of Data Memory with result in ACC
指令说明 将指定数据存储器的低 4 位与高 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。
功能表示 ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0
影响标志位 无

SZ [m] Skip if Data Memory is 0

指令说明 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。

功能表示 如果 [m]=0, 跳过下一条指令执行

影响标志位 无

SZA [m] Skip if Data Memory is 0 with data movement to ACC

指令说明 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。

功能表示 ACC ← [m]，如果 [m]=0，跳过下一条指令执行

影响标志位 无

SZ [m].i Skip if bit i of Data Memory is 0

指令说明 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。

功能表示 如果 [m].i=0，跳过下一条指令执行

影响标志位 无

TABRD [m] Read table (specific page) to TBLH and Data Memory

指令说明 将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定数据存储器且将高字节移至 TBLH。

功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)

影响标志位 无

TABRDC [m] Read table (current page) to TBLH and Data Memory

指令说明 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。

功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)

影响标志位 无

TABRDL [m] Read table (last page) to TBLH and Data Memory
指令说明 将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)
影响标志位 无

XOR A, [m] Logical XOR Data Memory to ACC
指令说明 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。
功能表示 ACC ← ACC“XOR”[m]
影响标志位 Z

XORM A, [m] Logical XOR ACC to Data Memory
指令说明 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。
功能表示 [m] ← ACC“XOR”[m]
影响标志位 Z

XOR A, x Logical XOR immediate data to ACC
指令说明 将累加器的数据与立即数逻辑异或，结果存放到累加器。
功能表示 ACC ← ACC“XOR”x
影响标志位 Z

封装信息

请注意，这里提供的封装信息仅作为参考。由于这个信息经常更新，提醒用户咨询 Holtek 网站以获取最新版本的封装信息。

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

- 封装信息（包括外形尺寸、包装带和卷轴规格）
- 封装材料信息
- 纸箱信息

16-pin DIP (300mil) 外形尺寸

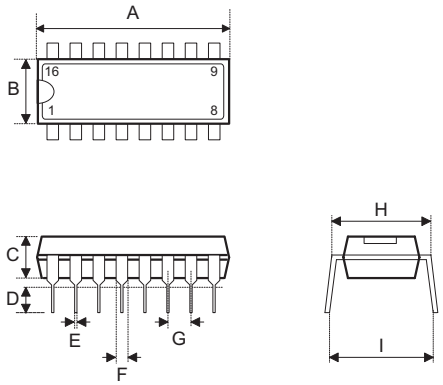


Fig1. Full Lead Packages

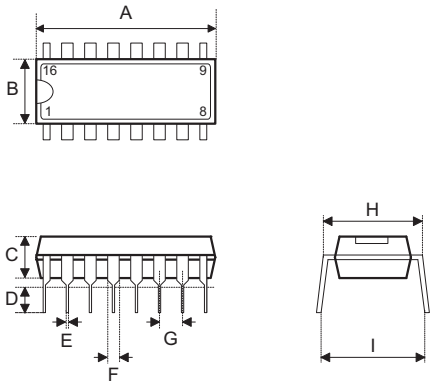


Fig2. 1/2 Lead Packages

见 **fig1**

符号	尺寸 (单位：inch)		
	最小	正常	最大
A	0.780	—	0.880
B	0.240	—	0.280
C	0.115	—	0.195
D	0.115	—	0.150
E	0.014	—	0.022
F	0.045	—	0.070
G	—	0.100	—
H	0.300	—	0.325
I	—	0.430	—

符号	尺寸 (单位：mm)		
	最小	正常	最大
A	19.81	—	22.35
B	6.10	—	7.11
C	2.92	—	4.95
D	2.92	—	3.81
E	0.36	—	0.56
F	1.14	—	1.78
G	—	2.54	—
H	7.62	—	8.26
I	—	10.92	—

见 **fig2**

符号	尺寸 (单位：inch)		
	最小	正常	最大
A	0.735	—	0.775
B	0.240	—	0.280
C	0.115	—	0.195
D	0.115	—	0.150
E	0.014	—	0.022
F	0.045	—	0.070
G	—	0.100	—
H	0.300	—	0.325
I	—	0.430	—

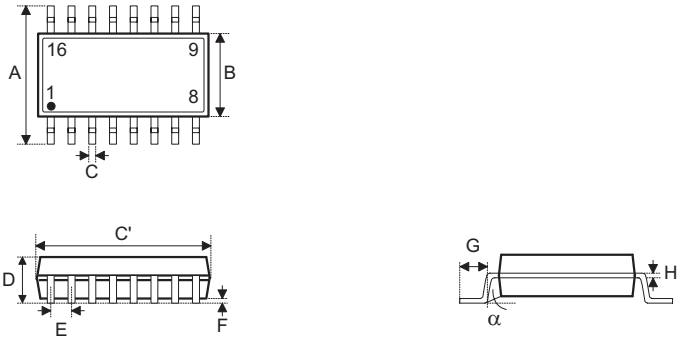
符号	尺寸 (单位：mm)		
	最小	正常	最大
A	18.67	—	19.69
B	6.10	—	7.11
C	2.92	—	4.95
D	2.92	—	3.81
E	0.36	—	0.56
F	1.14	—	1.78
G	—	2.54	—
H	7.62	—	8.26
I	—	10.92	—

见 **fig2**

符号	尺寸 (单位：inch)		
	最小	正常	最大
A	0.745	—	0.785
B	0.275	—	0.295
C	0.120	—	0.150
D	0.110	—	0.150
E	0.014	—	0.022
F	0.045	—	0.060
G	—	0.100	—
H	0.300	—	0.325
I	—	0.430	—

符号	尺寸 (单位：mm)		
	最小	正常	最大
A	18.92	—	19.94
B	6.99	—	7.49
C	3.05	—	3.81
D	2.79	—	3.81
E	0.36	—	0.56
F	1.14	—	1.52
G	—	2.54	—
H	7.62	—	8.26
I	—	10.92	—

16-pin NSOP (150mil) 外形尺寸



符号	尺寸 (单位：inch)		
	最小	正常	最大
A	0.228	—	0.244
B	0.150	—	0.157
C	0.012	—	0.020
C'	0.386	—	0.402
D	—	—	0.069
E	—	0.050	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.007	—	0.010
α	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	—	10.21
D	—	—	1.75
E	—	1.27	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.18	—	0.25
α	0°	—	8°

20-pin DIP (300mil) 外形尺寸

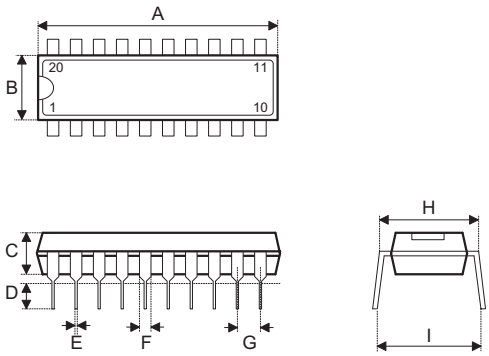


Fig1. Full Lead Packages

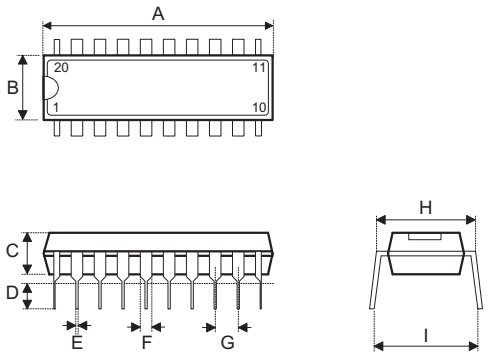


Fig2. 1/2 Lead Packages

见 **fig1**

符号	尺寸 (单位：inch)		
	最小	正常	最大
A	0.980	—	1.060
B	0.240	—	0.280
C	0.115	—	0.195
D	0.115	—	0.150
E	0.014	—	0.022
F	0.045	—	0.070
G	—	0.100	—
H	0.300	—	0.325
I	—	0.430	—

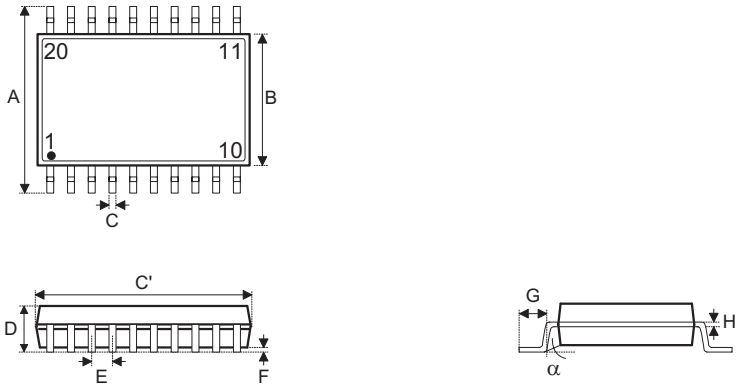
符号	尺寸 (单位：mm)		
	最小	正常	最大
A	24.89	—	26.92
B	6.10	—	7.11
C	2.92	—	4.95
D	2.92	—	3.81
E	0.36	—	0.56
F	1.14	—	1.78
G	—	2.54	—
H	7.62	—	8.26
I	—	10.92	—

见 **fig2**

符号	尺寸 (单位：inch)		
	最小	正常	最大
A	0.945	—	0.985
B	0.275	—	0.295
C	0.120	—	0.150
D	0.110	—	0.150
E	0.014	—	0.022
F	0.045	—	0.060
G	—	0.100	—
H	0.300	—	0.325
I	—	0.430	—

符号	尺寸 (单位：mm)		
	最小	正常	最大
A	24.00	—	25.02
B	6.99	—	7.49
C	3.05	—	3.81
D	2.79	—	3.81
E	0.36	—	0.56
F	1.14	—	1.52
G	—	2.54	—
H	7.62	—	8.26
I	—	10.92	—

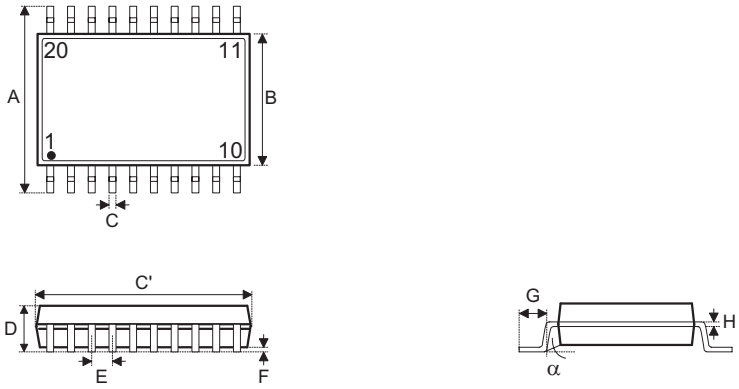
20-pin SOP (300mil) 外形尺寸



符号	尺寸 (单位：inch)		
	最小	正常	最大
A	0.393	—	0.419
B	0.256	—	0.300
C	0.012		0.020
C'	0.496	—	0.512
D	—	—	0.104
E	—	0.050	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小	正常	最大
A	9.98	—	10.64
B	6.50	—	7.62
C	0.30	—	0.51
C'	12.60	—	13.00
D	—	—	2.64
E	—	1.27	—
F	0.10	—	0.30
G	0.41	—	1.27
H	0.20	—	0.33
α	0°	—	8°

20-pin NSOP (150mil) 外形尺寸



符号	尺寸 (单位：inch)		
	最小	正常	最大
A	0.228	0.236	0.244
B	0.146	0.154	0.161
C	0.009	—	0.012
C’	0.382	0.390	0.398
D	—	—	0.069
E	—	0.032 BSC	—
F	0.002	—	0.009
G	0.020	—	0.031
H	0.008	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小	正常	最大
A	5.80	6.00	6.20
B	3.70	3.90	4.10
C	0.23	—	0.30
C’	9.70	9.90	10.10
D	—	—	1.75
E	—	0.80 BSC	—
F	0.05	—	0.23
G	0.50	—	0.80
H	0.21	—	0.25
α	0°	—	8°

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Microprocessors - MPU ***category:***

Click to view products by Holtek ***manufacturer:***

Other Similar products are found below :

MCIMX6D5EYM12AD A2C00010998 A ALXD800EEXJCVD C3 LS1020ASE7KQB LS1020AXE7KQB A2C00010729 A T1022NSE7MQB T1024NXE7PQA T1042NSN7WQB MPC8313EVRADDC BOXSTCK1A8LFCL LS1021ASE7KQB LS1021ASN7KQB MPC855TZQ80D4 MPC8569VJAUNLB P5020NSN7QMB P5020NXE7TNB T1024NXN7MQA T2080NXE8MQB T2080NXN8PTB MCIMX6L3EVN10AB T2080NXE8PTB T1024NXE7MQA CM8063501521600S R19L LS1043AXE7MQB T1024NXN7PQA LS1043ASE7QQB LS1012AXE7HKA T4240NSN7PQB MVF30NN152CKU26 FH8067303534005S R3ZM R9A07G044L24GBG#AC0 SVF311R3K2CKU2 HW8076502640002S R38F R7S721030VLFP#AA0 M0516LBN MCF5208CVM166 MCIMX6S6AVM08AC MCIMX6U5DVM10AC TEN54LSDV23GME MC68302AG33C MC68302EH16C MCF5233CVM150 MCIMX6D6AVT10AD MCIMX6G1CVM05AB MPC8245LZU350D MPC8314ECVRAGDA MPC8314VRAGDA MPC8315VRAGDA MPC8541VTAPF