## 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：
  $f_{SYS}$=4MHz：2.2V~5.5V
  $f_{SYS}$=8MHz：3.0V~5.5V
  $f_{SYS}$=12MHz：4.5V~5.5V
- 在 $V_{DD}$=5V，系统时钟为 12MHz 时，指令周期为 0.33μs
- 暂停模式和唤醒功能可降低功耗
- 振荡模式：
  外部高频晶振 -- HXT
  内部 RC -- HIRC
- 内部集成 4MHz，8MHz 和 12MHz 振荡器，不需要增加外部元器件
- OTP 程序存储器：2K×15
- RAM 数据存储器：128×8
- 看门狗定时器
- LIRC 振荡用于看门狗时钟
- 所有指令都可在 1 个或 2 个指令周期内完成
- 查表指令
- 63 条功能强大的指令系统
- 6 层堆栈
- 位操作指令
- 低电压复位功能
- 20-Pin DIP/SOP 封装

**周边特性**

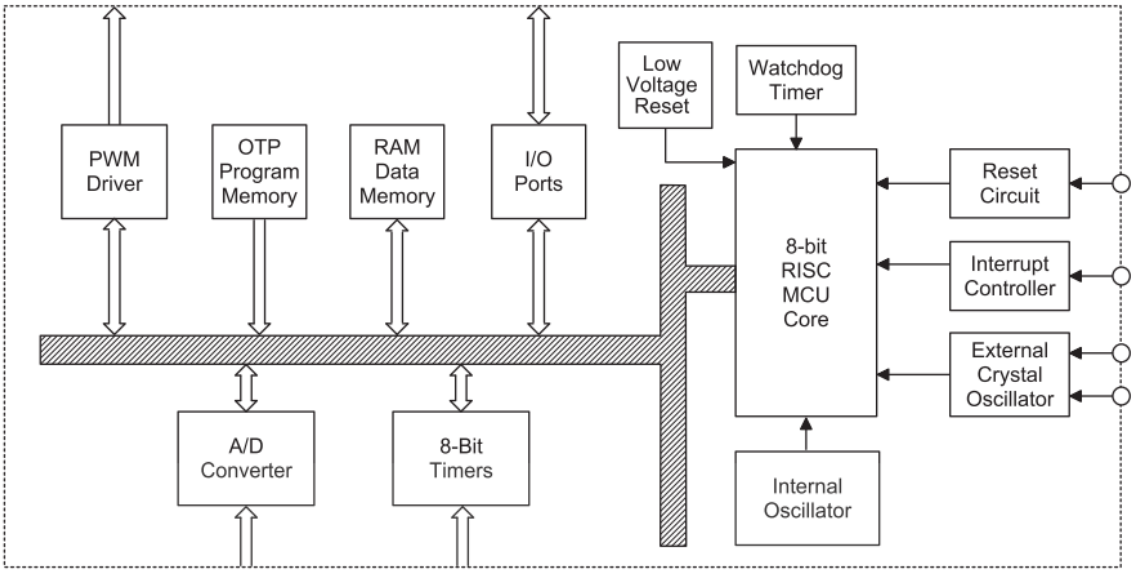
- 18 个双向输入/输出口
- 8 个通道 12 位 ADC
- 2 个通道 8 位 PWM
- 一个与 I/O 口复用的外部中断输入
- 一个 8 位可编程定时/计数器，具有溢出中断和预分频功能

## 概述

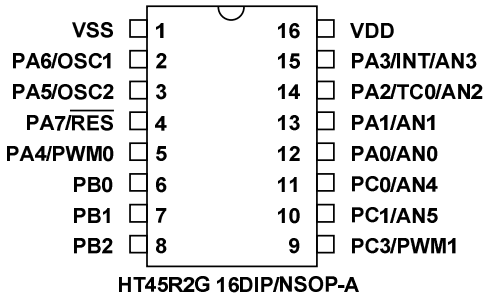
增强 A/D 型系列单片机是 8 位具有高性能精简指令集的单片机，应用相当广泛。秉承 Holtek 单片机具有低功耗、I/O 灵活、定时器功能、振荡类型可选、暂停和唤醒功能、看门狗和低电压复位等丰富的功能选项，具有极高的性价比。其内部集成了系统振荡器 HIRC，提供三种频率选择，不需要增加外部元器件，可以广泛适用于各种应用，例如工业控制、消费类产品、家用电器子系统控制等。

## 方框图

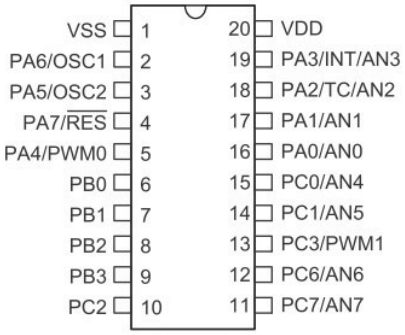
下面方框图描述了主要的功能模块。




## 引脚图




HT45R2G 16DIP/NSOP-A




HT45R2G
20 DIP-A/SOP-A

## 引脚说明

每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。

| 引脚 | 功能 | OPT | I/T | O/T | 说明 |
|---|---|---|---|---|---|
| PA0/AN0 | PA0 | PAPU<br>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 | ANCSR | AN | — | A/D 通道 0 |
| PA1/AN1 | PA1 | PAPU<br>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 |
| | AN1 | ANCSR | AN | — | A/D 通道 1 |
| PA2/TC/AN2 | PA2 | PAPU<br>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 |
| | TC | — | ST | — | 外部定时器时钟源输入脚 |
| | AN2 | ANCSR | AN | — | A/D 通道 2 |
| PA3/INT/AN3 | PA3 | PAPU<br>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 | ANCSR | AN | — | A/D 通道 3 |
| PA4/PWM0 | PA4 | PAPU<br>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 |
| | PWM0 | CTRL0 | — | CMOS | PWM 输出 0 |
| PA5/OSC2 | PA5 | PAPU<br>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<br>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 | — | 振荡器引脚 |
| PA7/$\overline{\text{RES}}$ | PA7 | PAWK | ST | NMOS | 通用 I/O 口，可通过寄存器设置带唤醒功能 |
| | $\overline{\text{RES}}$ | CO | ST | — | 复位输入脚 |
| PB0~PB3 | PBn | PBPU | ST | CMOS | 通用 I/O 口，可通过寄存器设置带上拉电阻 |
| PC0/AN4<br>PC1/AN5<br>PC6/AN6<br>PC7/AN7 | PCn | PCPU | ST | CMOS | 通用 I/O 口，可通过寄存器设置带上拉电阻 |
| | ANn | ANCSR | AN | — | A/D 通道 4,5,6,7 |
| PC2 | PC2 | PCPU | ST | CMOS | 通用 I/O 口，可通过寄存器设置带上拉电阻 |
| PC3/PWM1 | PC3 | PCPU | ST | CMOS | 通用 I/O 口，可通过寄存器设置带上拉电阻 |
| | PWM1 | CTRL0 | — | CMOS | PWM 输出 1 |
| VDD | VDD | — | PWR | — | 正电源 |
| VSS | VSS | — | PWR | — | 负电源，接地 |

注意：I/T：输入类型
O/T：输出类型
OPT：通过配置选项（CO）或者寄存器选项来设置
PWR：电源
CO：配置选项
ST：斯密特触发输入
AN：模拟输入脚
CMOS：CMOS 输出
NMOS：NMOS 输出
HXT：高频晶体振荡器

## 极限参数

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

**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℃**

| 符号 | 参数 | 测试条件 | | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $V_{DD}$ | 工作电压 | — | $f_{SYS}$ =4MHz | 2.2 或 $V_{LVR}$ | — | 5.5 | V |
| | | | $f_{SYS}$ =8MHz | 3.0 或 $V_{LVR}$ | — | 5.5 | V |
| | | | $f_{SYS}$ =12MHz | 4.5 或 $V_{LVR}$ | — | 5.5 | V |
| $I_{DD1}$ | 工作电流 (HXT，HIRC) | 3V | 无负载，$f_{SYS}$ =4MHz，ADC 关闭 | — | 0.80 | 1.20 | mA |
| | | 5V | | — | 1.50 | 2.25 | mA |
| $I_{DD2}$ | 工作电流 (HXT，HIRC) | 3V | 无负载，$f_{SYS}$ =8MHz，ADC 关闭 | — | 1.40 | 2.10 | mA |
| | | 5V | | — | 2.80 | 4.20 | mA |
| $I_{DD3}$ | 工作电流 (HXT，HIRC) | 5V | 无负载，$f_{SYS}$ =12MHz，ADC 关闭 | — | 4 | 6 | mA |
| $I_{STB}$ | 静态电流 (LIRC On) | 3V | 无负载，系统暂停 | — | 1 | 5 | μA |
| | | 5V | | — | 4 | 10 | μA |
| $V_{IL1}$ | I/O，TC 和 INT 的低电平输入电压 | — | — | 0 | — | $0.2V_{DD}$ | V |
| | | 5V | | 0 | — | 1.5 | V |
| $V_{IH1}$ | I/O，TC 和 INT 的高电平输入电压 | — | — | $0.8V_{DD}$ | — | $V_{DD}$ | V |
| | | 5V | | 3.5 | — | 5.0 | 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 | — | $V_{LVR}$ = 4.20V | 3.98 | 4.2 | 4.42 | V |
| | | — | $V_{LVR}$ = 3.15V | 2.98 | 3.15 | 3.32 | V |
| | | — | $V_{LVR}$ = 2.10V | 1.98 | 2.1 | 2.22 | V |
| $I_{OH}$ | I/O 口源电流 (不包含 PA7) | 3V | $V_{OH}$=0.9$V_{DD}$ | -2 | -4 | — | mA |
| | | 5V | | -5 | -10 | — | mA |
| $I_{OL1}$ | I/O 口灌电流 (不包含 PA7) | 3V | $V_{OL}$=0.1$V_{DD}$ | 4 | 8 | — | mA |
| | | 5V | | 10 | 20 | — | mA |
| $I_{OL2}$ | PA7 灌电流 | 3V | $V_{OL}$=0.1$V_{DD}$ | 1.0 | 1.5 | — | mA |
| | | 5V | | 2.0 | 3.0 | — | mA |
| $R_{PH}$ | 上拉电阻 | 3V | — | 20 | 60 | 100 | kΩ |
| | | 5V | — | 10 | 30 | 50 | kΩ |

注：低电压复位选项 $V_{LVR}$ 决定了最小的工作电压。如果最小工作电压低于 $V_{LVR}$，单片机只能工作在 $V_{LVR}$ 下。

## 交流电气特性

Ta=25℃

| 符号 | 参数 | 测试条件 | | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $f_{SYS}$ | 系统时钟 | — | 2.2V~5.5V | 400 | — | 4000 | kHz |
| | | | 3.0V~5.5V | 400 | — | 8000 | kHz |
| | | | 4.5V~5.5V | 400 | — | 12000 | kHz |
| $f_{HIRC}$ | 系统时钟(HIRC) | 3V/5V | Ta=25℃ | -2% | 4 | +2% | MHz |
| | | 3V/5V | Ta=25℃ | -2% | 8 | +2% | MHz |
| | | 5V | Ta=25℃ | -2% | 12 | +2% | MHz |
| | | 3V/5V | Ta=0~70℃ | -5% | 4 | +5% | MHz |
| | | 3V/5V | Ta=0~70℃ | -5% | 8 | +5% | MHz |
| | | 5V | Ta=0~70℃ | -5% | 12 | +5% | MHz |
| | | 2.2V~3.6V | Ta=0~70℃ | -8% | 4 | +8% | MHz |
| | | 3.0V~5.5V | Ta=0~70℃ | -8% | 4 | +8% | MHz |
| | | 3.0V~5.5V | Ta=0~70℃ | -8% | 8 | +8% | MHz |
| | | 4.5V~5.5V | Ta=0~70℃ | -8% | 12 | +8% | MHz |
| | | 2.2V~3.6V | Ta=-40℃~85℃ | -12% | 4 | +12% | MHz |
| | | 3.0V~5.5V | Ta=-40℃~85℃ | -12% | 4 | +12% | MHz |
| | | 3.0V~5.5V | Ta=-40℃~85℃ | -12% | 8 | +12% | MHz |
| | | 4.5V~5.5V | Ta=-40℃~85℃ | -12% | 12 | +12% | MHz |
| $f_{TIMER}$ | 定时器输入频率(TC 脚) | — | 2.2V~5.5V | 0 | — | 4000 | kHz |
| | | | 3.0V~5.5V | 0 | — | 8000 | kHz |
| | | | 4.5V~5.5V | 0 | — | 12000 | kHz |
| $f_{LIRC}$ | LIRC 振荡器频率 | 3V | — | 5.0 | 10.0 | 15.0 | kHz |
| | | 5V | — | 6.5 | 13.0 | 19.5 | kHz |
| $t_{RES}$ | 外部复位低电压脉宽 | — | — | 1 | — | — | μs |
| $t_{INT}$ | 中断脉冲宽度 | — | — | 1 | — | — | μs |
| $t_{LVR}$ | 低电压复位宽度 | — | — | 0.13 | 0.5 | 1.0 | ms |
| $t_{SST}$ | 系统启动时间周期 | — | HXT | 128 | — | — | $t_{SYS}$ |
| | | | HIRC | — | 2 | — | $t_{SYS}$ |
| $t_{RSTD}$ | 复位延时时间 | 3V | — | 26 | 52 | 104 | ms |
| | | 5V | — | 20 | 40 | 80 | ms |

**注**：1、$t_{SYS}=1/f_{SYS}$

2、为了保证 HIRC 振荡器的频率精度，$V_{DD}$ 与 Vss 间连接一个 0.1μF 的去耦电容，并尽可能接近芯片。

## ADC 特性

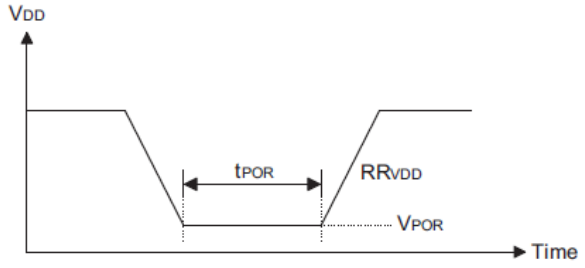
Ta=25℃

| 符号 | 参数 | 测试条件 | | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $AV_{DD}$ | A/D 工作电压 | — | — | 2.7 | — | 5.5 | V |
| DNL | A/D 非线性微分误差 | — | $t_{AD}$=0.5μs | -2 | — | 2 | LSB |
| INL | A/D 非线性积分误差 | — | $t_{AD}$=0.5μs | -4 | — | 4 | LSB |
| $I_{ADC}$ | 打开 A/D 增加的功耗 | 3V | 无负载，$t_{AD}$=0.5μs | — | 0.5 | 0.75 | mA |
| | | 5V | | — | 1.0 | 1.5 | mA |
| $t_{AD}$ | A/D 时钟周期 | 2.7V~5.5V | — | 0.5 | — | 10 | μs |
| $t_{ADC}$ | A/D 转换时间<br>(包括采样和保持时间) | 2.7V~5.5V | — | — | 16 | — | $t_{AD}$ |
| $t_{ADS}$ | A/D 采样时间 | 2.7V~5.5V | — | — | 4 | — | $t_{AD}$ |
| $t_{ON2ST}$ | A/D 启动到执行时间 | 2.7V~5.5V | — | 2 | — | — | μs |

## 上电复位特性

Ta=25℃

| 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 |
| $RR_{VDD}$ | 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 |

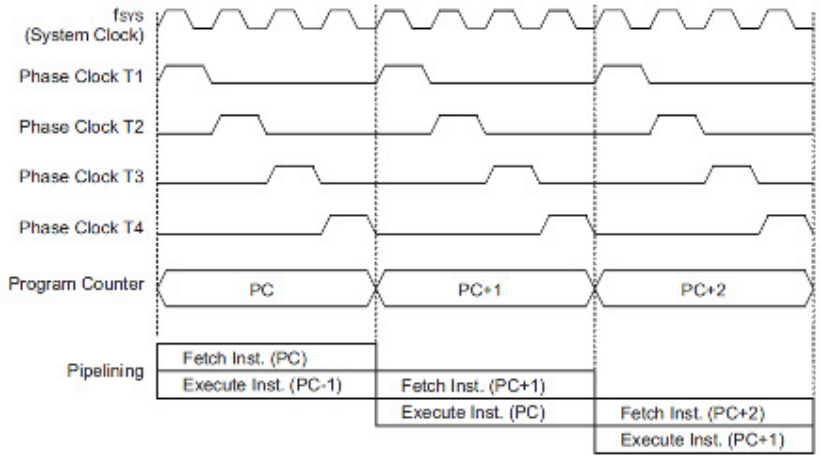


# 系统结构

内部系统结构是盛群半导体公司单片机具有良好性能的主要因素，由于采用 RISC 结构，此系列的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，即指令的取得和执行同时进行，此举使得除了跳转、调用指令之外，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，它可以实现算术运算、逻辑运算、移位、递增、递减和分支等功能，而内部数据路径则是通过累加器或 ALU 的方式来加以简化。有些内部寄存器可在数据寄存器中执行，且可以直接或者间接寻址。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，只需要少数的外部组件。

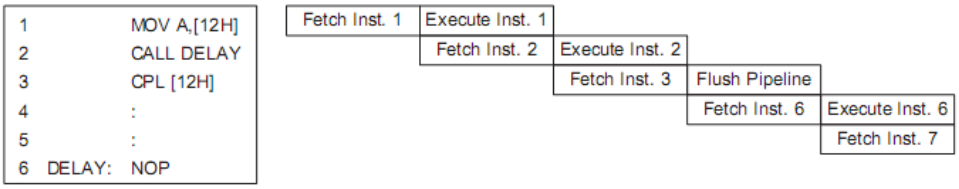
## 时序和流水线结构

主系统时钟由晶体/陶瓷振荡器，或者 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，如子程序的调用或者跳转，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。然而需注意的是，只有较低的 8 位，即所谓的程序低字节寄存器 PCL，可以被用户直接读写。

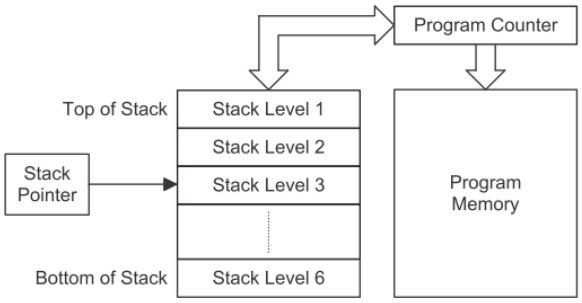
当执行的指令要求跳转到不连续的地址时，如跳转指令、子程序调用、中断或者复位等，单片机通过加载所需要的位址到程序寄存器来控制程序。对于条件跳转指令，一旦条件符合，在当前指令执行时取得的下一条指令将会被舍弃，而由一个空指令周期来取代。

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，需要注意的是会插入一个空指令周期。

程序计数器的低字节可由程序直接进行读取，PCL 的使用可能引起程序跳转，因此需要额外的指令周期。有关 PCL 寄存器的更多信息请参考特殊功能寄存器章节的说明。

## 堆栈

堆栈是一个特殊的存储器空间，用来保存程序计数器中的值。堆栈寄存器既不是数据存储器的一部分，也不是程序存储器的一部分，而且它既不能读出，也不能写入。堆栈的使用是通过堆栈指针 SP 来指示的，堆栈指针也不能读出或写入。当发生子程序调用或中断响应时，程序计数器中的内容会被压入堆栈。在子程序调用结束或中断响应结束时，执行指令 RET 或 RETI，堆栈将原先压入堆栈的内容弹出，重新装入程序计数器中。在系统复位后，堆栈指针会指向堆栈顶部。




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

程序存储器的容量有 2K×15。程序存储器用程序计数器来寻址，其中也包括数据、表格信息和中断入口。表格数据可以设置在程序存储器的任意地址，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**

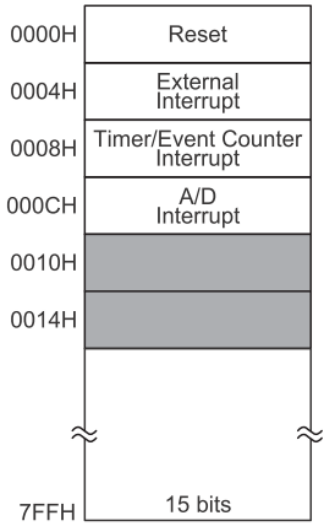
该向量为外部中断服务程序使用。当外部中断引脚发生边沿跳变时，如果中断允许且堆栈未满，则程序会跳转到该地址开始执行。外部中断有效边沿转换类型由 CTRL1 寄存器指定设定是高到低，还是低到高有效或者两者都可以触发。

- **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，则程序会跳转到相应的地址开始执行。

- **A/D 中断向量**

该向量为 A/D 转换使用。当 A/D 转换发生，如果 A/D 转换中断允许且堆栈未满，则程序将跳转到相应地址开始执行。



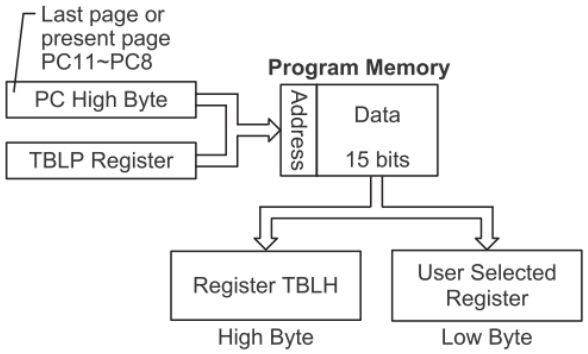

**程序存储器结构**

## 查表

程序存储器中的任何地址都可以定义为一个表格，以便存储固定的数据。使用查表指令时，查表指针需要先行设定，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]”指令从程序所在的存储器的当前页或者存储器的最后一页中查表来读取。当这些指令执行时，程序存储器的表格的低字节，将被传输到用户所指定的数据存储器中。程序存储器表格的高字节，将被传输到特殊寄存器 TBLH 中。传输数据中任何未定义的字节将被读取为“0”。

下图是查表中寻址/数据流程：




## 查表范例

以下范例说明了单片机如何定义表格指针、如何查表。此范例使用的表格数据用 ORG 伪指令存储在存储器的最后一页。在此 ORG 伪指令的值为“700H”，即 2K 程序存储器最后一页的开始地址，查表指针设置的初始值为 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 706H，即是最后一页开始地址的第 6 个地址。值得注意的是假如使用“TABRDC [m]”指令时，表格指针的值必须指向当前页的第一个地址。当“TABRDL [m]”指令被执行时，表格数据的高字节将自动被传送到寄存器 TBLH 中，此范例中表格数据的高字节等于零。

因为 TBLH 寄存器是只读寄存器，不能被重新存储，若主程序和中断服务程序都要使用表格读取指令的时候，应注意它的保护。使用表格读取指令，中断服务可能改变 TBLH 的值，若主程序再次使用这个值时，则会出现错误。因此建议避免同时使用表格读取指令。然而当出现同时使用表格读取指令不可避免的情况，则在执行任何主程序的表格读取指令之前，中断应该被禁止。值得注意的是所有表格相关的指令，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                     ; initialise 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“706H”transferred 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 "705H" 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     700h                    ; sets initial address of last page
dc      00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh :
        :
        :
```

| 指令 | 表格地址位 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **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：1. PC10 ~ PC8：当前程序计数器位
2. @7~@0：表格指针 TBLP 位

## 数据存储器

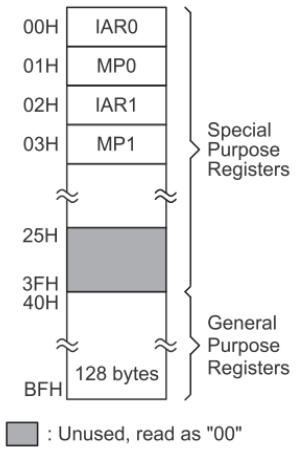
数据存储器是内容可以更改的 8 位 RAM 内部存储器，用来存储临时数据。

### 结构

数据存储器分为两个部分，第一部分是特殊功能寄存器，这些寄存器有特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，而有些是被加以保护而不对用户开放。第二部分是通用数据存储器，所有地址都可在程序的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，而数据储存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**

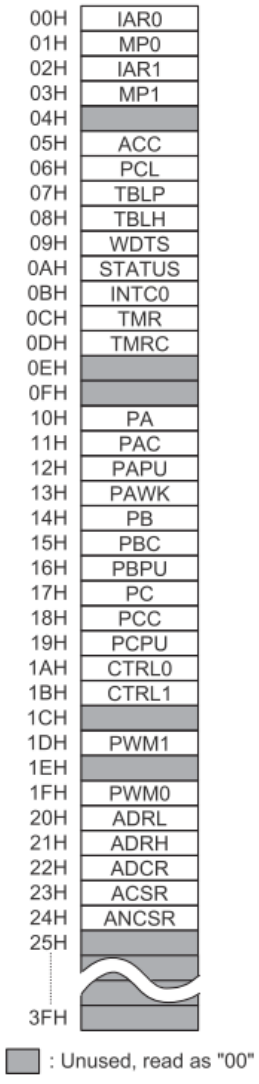
注意：使用“SET [m].i”和“CLR [m].i”可对大多数的数据存储区进行位操作，少数专用位除外。数据存储区也可以通过存储器指针间接寻址。

**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。

## 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。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。




**特殊功能数据存储器**

### 间接寻址寄存器 — IAR0，IAR1

间接寻址寄存器 IAR0 和 IAR1，位于数据储存区，并没有实际的物理地址。间接寻址方式是使用间接寻址寄存器或者存储器指针对数据操作，以取代定义在实际存储器地址的直接存储器寻址方式。在间接寻址寄存器上的任何动作，将对间接寻址指针（MP0 或 MP1）所指定的存储器地址产生对应的读/写操作。IAR0 和 MP0，IAR1 和 MP1 对数据存储器中数据的操作是成对出现的。间接寻址寄存器不是实际存在的，直接读取 IAR 寄存器将会返回 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写入和操作，因此提供了一个有效的间接寻址和数据追踪的方法。当对间接寻址寄存器进行任何操作时，单片机所指向的实际地址是由间接寻址指针所指定的地址。

以下范例说明如何清除一个具有 4 个 RAM 地址的区块，它们已经事先被定义成地址 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:
```

在以上的例子中需要注意的是，没有提及具体的数据存储器地址。

### 累加器－ACC

对于任何单片机来说，累加器是相当重要的，且与ALU所完成的运算有密切关系。所有的ALU得到的运算结果都将暂存在累加器中。如果没有累加器，ALU必须在每次进行如加法、减法和移位等运算时，将结果写入数据存储器中，这样会造成程序编写和时间的负担。另外，数据传输通常涉及到累加器的临时储存功能，如在用户定义的寄存器和另一个寄存器之间，由于两者之间不能直接传送数据，因此需要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，程序计数器的低字节被设置在数据存储器的特殊功能区域，程序员可对此寄存器进行操作，很容易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专用程序存储器某一地址，然而，由于寄存器只有8位的长度，因此只允许在本页的程序存储器中跳转。值得注意的是，使用这种运算时，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。

- **状态寄存器**

| 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 | x | x | x | x |

x：表示未知

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/输出口和控制寄存器

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/输出口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/输出口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。这些功能包括外部中断边沿触发类型和PWM功能控制。

• CTRL0 寄存器

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | PWMSEL | PWMC1 | PWMC0 | — | — | — |
| R/W | — | — | R/W | R/W | R/W | — | — | — |
| POR | — | — | 0 | 0 | 0 | — | — | — |

Bit 7~6 未定义，读为 “0”
Bit 5 **PWMSEL**：PWM 模式选择位
0：6+2 模式
1：7+1 模式
Bit 4 **PWMC1**：I/O 或 PWM1 选择位
0：I/O
1：PWM1 输出
Bit 3 **PWMC0**：I/O 或 PWM0 选择位
0：I/O
1：PWM0 输出
Bit 2~0 未定义，读为 “0”
注意：如果PWMCn位选择PWMn作为输出，则PWM的时钟源$f_{TP}$来自于系统时钟$f_{SYS}$。

• CTRL1 寄存器

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | INTEG1 | INTEG0 | — | — | — | — | — | — |
| R/W | R/W | R/W | — | — | — | — | — | — |
| POR | 1 | 0 | — | — | — | — | — | — |

Bit 7,6 **INTEG1，INTEG0**：外部中断边沿触发类型
00：关闭中断
01：上升沿触发
10：下降沿触发
11：双边沿触发
Bit 5~0 未定义，读为 “0”

**唤醒功能寄存器-PAWK**

当单片机进入暂停模式以后，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, PCPU**

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获得更多范围的功能。振荡器的灵活性使得在速度和功耗之间可以达到最优化。振荡器选项是通过配置选项和寄存器来完成的。

## 振荡器概述

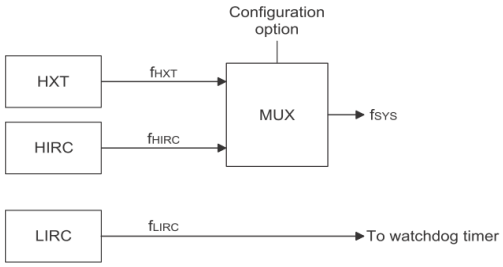
振荡器除了作为系统时钟源，还作为看门狗定时器的时钟源。外部振荡器需要一些外围器件,而集成的两个内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。

| 振荡类型 | 名称 | 频率 | 引脚 |
|---|---|---|---|
| 外部晶振 | HXT | 400kHz~12MHz | OSC1/OSC2 |
| 内部高速RC | HIRC | 4, 8或12MHz | — |

**振荡器类型**

## 系统时钟配置

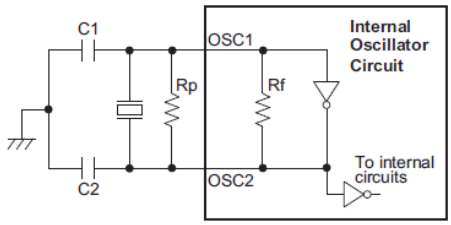
此系列的单片机有两个系统振荡器。这些振荡器有外部晶体/陶瓷振荡器-HXT 和内部 RC 振荡器-HIRC。详细介绍在相关章节描述。




**系统时钟配置**

## 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.

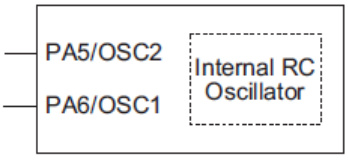
**晶体/陶瓷振荡器-HXT**

| 晶体振荡器 C1 和 C2 值 | | |
|---|---|---|
| 晶体频率 | C1 | C2 |
| 12MHz | — | — |
| 8MHz | — | — |
| 4MHz | — | — |
| 1MHz | — | — |
| 1MHz（陶瓷谐振器） | 100pF | 100pF |
| 注意：C1 和 C2 数值仅作参考用 | | |

**晶体振荡器电容推荐值**

**内部 RC 振荡器 -- HIRC**

内部 RC 振荡器是一个集成的系统振荡器，不需其他外部器件。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含有内部频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响减至最低。注意的是，如果选择了内部系统时钟，无需额外的引脚接其他元件，此时 PA5 和 PA6 可以作为通用 I/O 口使用。




注意：PA5/PA6用作通用的I/O口

**内部 RC 振荡器-HIRC**

**内部低速振荡器 -- LIRC**

LIRC 是一个完全集成 RC 振荡器，它在 5V 电压下运行的典型频率值为 13kHz 且无需外部元件。LIRC 振荡器用作看门狗定时器的时钟源。当系统进入暂停模式时，系统时钟停止运行但 LIRC 振荡器继续运行且看门狗保持工作。

## 暂停模式和唤醒

在实际应用领域中，单片机会被暂停，从而使系统时钟关闭以降低功耗。单片机可用多种方式来唤醒，以恢复正常功能。

**进入暂停模式**

进入暂停模式的方法只有一种，即在应用程序中执行“HALT”指令。当执行该指令后，会发生下面情况：

- 系统振荡器停止运行，应用程序将停止在“HALT”指令处。
- 数据存储器中的内容和寄存器内容保持当前值。
- WDT 将被清除然后再重新计数。
- 输入/输出端口保持当前状态。
- 状态寄存器中暂停标志 PDF 将被置位，看门狗溢出标志 TO 将被清除。

**静态电流的注意事项**

由于单片机进入暂停模式的主要原因是将 MCU 的电流降低到尽可能低，可能到只有几个毫安的级别，所以如果要将电路的电流降到最低，电路设计者还应有其它的考虑。

应该特别注意的是单片机的输入/输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。另外还需注意单片机设为输出的 I/O 引脚上的负载。应将它们设置在有最小拉电流的状态或将它们和其它的 CMOS 输入一样接到没有拉电流的外部电路上。

如果配置选项使能看门狗振荡器 LIRC，当进入暂停模式，振荡器继续振荡，并继续消耗能量。

**唤醒**

当系统进入暂停模式，可以通过以下几种方式唤醒：

- 外部复位
- PA0~PA7口下降沿
- 系统中断
- WDT溢出

若由外部复位唤醒，系统会经过完全复位的过程。若由WDT溢出唤醒，则看门狗定时器将被复位清零。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寄存器置位并由引脚上的负跳转选择唤醒功能。PA口唤醒后，程序将执行“HALT”指令后的其他指令。

如果系统是通过中断唤醒，则有两种可能发生。第一种情况是：相关中断除能或是堆栈已满，则程序会在“HALT”指令之后继续执行。这种情况下，唤醒系统的中断会等到相关中断使能或有堆栈层可以使用之后才执行。第二种情况是：相关中断使能且堆栈未满，则中断可以马上执行。 如果在进入暂停模式之前中断标志位已经被设置为“1”，则相关中断的唤醒功能将无效。

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时时间，延迟的时长请参照下面的表格：

| 唤醒源 | 振荡器类型 | |
|---|---|---|
| | HIRC | Crystal |
| 外部$\overline{RES}$ | $t_{RSTD}+t_{SST1}$ | $t_{RSTD}+t_{SST2}$ |
| PA 口 | $t_{SST1}$ | $t_{SST2}$ |
| 中断 | | |
| WDT 溢出 | | |

注：1. $t_{RSTD}$ 为复位延时时间，$t_{SYS}$ 为系统时钟
2. $t_{RSTD}$ 为上电延时时间，$V_{DD}$=5V 时，典型值为 40ms
3. $t_{SST1}=2t_{SYS}$
4. $t_{SST2}=128t_{SYS}$

**唤醒延时时间**

## 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。通过内部寄存器 WDTS，可以设置不同的 WDT 选项。无论是在正常模式还是在暂停模式，看门狗定时器都为使能。WDT 时钟源来自 LIRC 振荡器。

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 1:128，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。但是在系统处于暂停模式时，如果 WDT 发生溢出，系统将从暂停中唤醒，置位状态标志位 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。使用软件指令只有一种方法去清除看门狗寄存器，由配置选项选择，只要执行“CLR WDT”便清除 WDT。CLR WDT1 和 CLR WDT2 不起作用。

• **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：$2^{10} t_{WDTCK}$

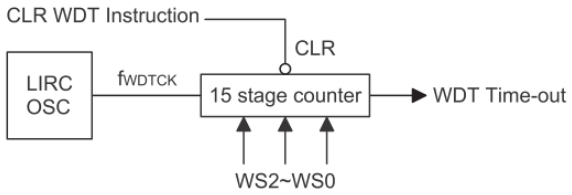
011：$2^{11} t_{WDTCK}$

100：$2^{12} t_{WDTCK}$

101：$2^{13} t_{WDTCK}$

110：$2^{14} t_{WDTCK}$

111：$2^{15} t_{WDTCK}$




**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。

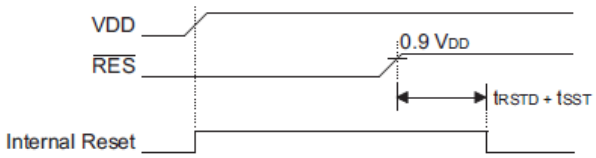
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位，这种复位与$\overline{\text{RES}}$脚拉低复位方式相同。

### 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。

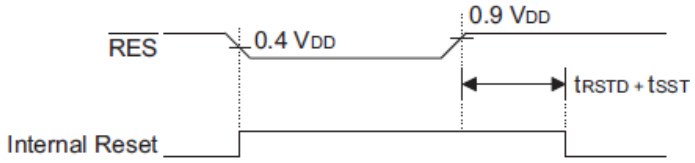



注：$t_{RSTD}$为上电延时时间，典型值为40ms

**上电复位时序图**

- $\overline{RES}$引脚复位

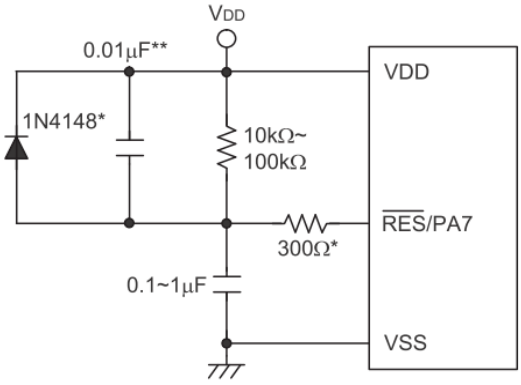
由于复位引脚与 PA.7 引脚共用，外部复位引脚功能必须通过配置选项来选择。虽然单片机有内部 RC 复位的功能，如果 $V_{DD}$ 电源供应时间不够快或上电不能很快稳定，则内部复位功能就不能提供合适的复位选项。因此建议$\overline{RES}$引脚连接一个外部 RC，它可以延时使$\overline{RES}$引脚保持低电压从而让电压供应保持稳定。在这段时间内，单片机的正常操作是被禁止的。当$\overline{RES}$引脚达到某一电压值时，复位延时 $t_{RSTD}$ 要提供一个额外延时，单片机可以开始进行正常操作。算术中的缩写 SST 表示系统的启动时间。在许多应用场合，可以在 $V_{DD}$ 和$\overline{RES}$之间接入一个电阻，在 Vss 与$\overline{RES}$之间接入一个电容作为外部复位电路。任何和$\overline{RES}$脚的连接线必须尽量短以减少噪声干扰。




注意：$t_{RSTD}$为上电延时时间，典型值为40ms。

**$\overline{RES}$复位时序图**

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。



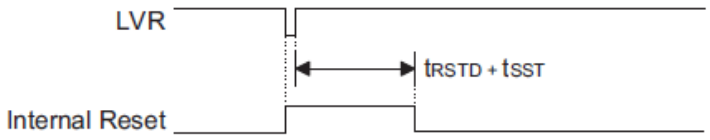

注：“*” 表示建议加上此元件以加强静电保护。
“**”表示建议在电源有较强干扰场合加上此元件。

**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。

- 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。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，需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。需注意的是不论单片机片处于什么模式 LVR 都使能。

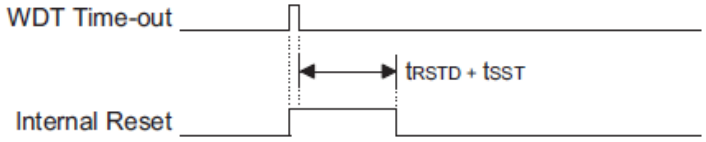



注意：$t_{RSTD}$为上电延时时间，典型值为40ms。

**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$\overline{RES}$复位相同。

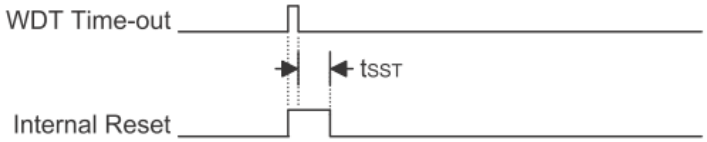



注意：$t_{RSTD}$为上电延时时间，典型值为40ms。

**正常运行时看门狗溢出时序图**

- 暂停模式时看门狗溢出复位

暂停模式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，绝大部份的条件保持不变。图中$t_{SST}$的详细说明请参考交流电气特性。




注意：如果系统时钟为HIRC，$t_{SST}$为2个时钟周期，如果系统时钟为HXT，$t_{SST}$为128个时钟周期。

**暂停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，由暂停模式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$\overline{RES}$复位或 LVR 复位 |
| 1 | u | 正常模式时的 WDT 溢出复位 |
| 1 | 1 | 暂停模式时的 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 |
| 预分频器 | 定时/计数器的预分频器内容清除 |
| 输入/输出口 | 所有 I/O 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。

| 寄存器 | 上电复位 | $\overline{\text{RES}}$或 LVR 复位 | WDT 溢出 (正常模式) | WDT 溢出 (暂停模式) |
|---|---|---|---|---|
| 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 |
| TMR | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| TMRC | 0000 1000 | 0000 1000 | 0000 1000 | uuuu uuuu |
| 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 | - - - - 1111 | - - - - 1111 | - - - - 1111 | - - - - uuuu |
| PBC | - - - - 1111 | - - - - 1111 | - - - - 1111 | - - - - uuuu |
| PBPU | - - - - 0000 | - - - - 0000 | - - - - 0000 | - - - - uuuu |
| PC | 11- - 1111 | 11- - 1111 | 11- - 1111 | uu- - uuuu |
| PCC | 11- - 1111 | 11- - 1111 | 11- - 1111 | uu- - uuuu |
| PCPU | 00- - 0000 | 00- - 0000 | 00- - 0000 | uu- - uuuu |
| CTRL0 | - -00 0- - - | - -00 0- - - | - -00 0- - - | - -uu u- - - |
| CTRL1 | 10- - - - - - | 10- - - - - - | 10- - - - - - | uu- - - - - - |
| PWM0 | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| PWM1 | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADRL | xxxx - - - - | xxxx - - - - | xxxx - - - - | uuuu - - - - |
| ADRH | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADCR | 01- - -000 | 01- - -000 | 01- - -000 | uu- - -uuu |
| ACSR | -1- - -000 | -1- - -000 | -1- - -000 | -u- - - uuu |
| ANCSR | 0000 0000 | 0000 0000 | 0000 0000 | 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/输出口控制具有很大的灵活性。每一个引脚在用户程序控制下都可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，也就是说输入数据需在执行“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，PBPU 和 PCPU 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。要注意的是，PA7 引脚没有上拉电阻功能。

### PA 口唤醒

当使用暂停指令“HALT”迫使单片机进入暂停模式状态，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换为低电平。使用暂停指令“HALT”迫使单片机进入暂停模式状态后，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。值得注意的是，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。

### 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（PAC, PBC, 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，动态地设置带上拉电阻或者不带上拉电阻。若 I/O 引脚要实现输入功能，则对应的控制寄存器的位需设置为“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。值得注意的是，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。

- **PAWK, PAC, PAPU, PBC, PBPU, PCC, PCPU 寄存器**

| 寄存器名称 | POR | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 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 |
| PBC | 0FH | — | — | — | — | PBC3 | PBC2 | PBC1 | PBC0 |
| PBPU | 00H | — | — | — | — | PBPU3 | PBPU2 | PBPU1 | PBPU0 |
| PCC | CFH | PCC7 | PCC6 | — | — | PCC3 | PCC2 | PCC1 | PCC0 |
| PCPU | 00H | PCPU7 | PCPU6 | — | — | PCPU3 | PCPU2 | PCPU1 | PCPU0 |

“—”未定义，读为“0”

**PAWKn**：PA 唤醒功能使能
0：除能
1：使能

**PACn/PBCn/ PCCn**：I/O 类型选择
0：输出
1：输入

**PAPUn/PBPUn/ PCPUn**：上拉功能使能
0：除能
1：使能

### 引脚重置功能

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。多功能输入/输出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 INTC 寄存器中的有关位。此外，还需要通过端口控制寄存器的 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TC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者脉冲宽度测量模式, 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。注意的是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。

- **PWM输出**

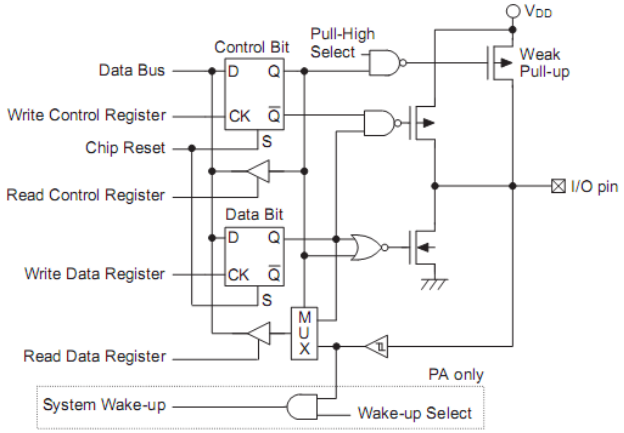
PWM功能与输入/输出引脚共用。PWM输出功能通过CTRL0寄存器来设置。注意端口控制寄存器相应的位需要设置为输出高才能使能PWM的输出。如果端口控制寄存器被设置为输入，即使PWM寄存器已经使能PWM功能，该引脚都只作为普通逻辑输入脚，并且允许选择上拉电阻。

- **A/D输入**

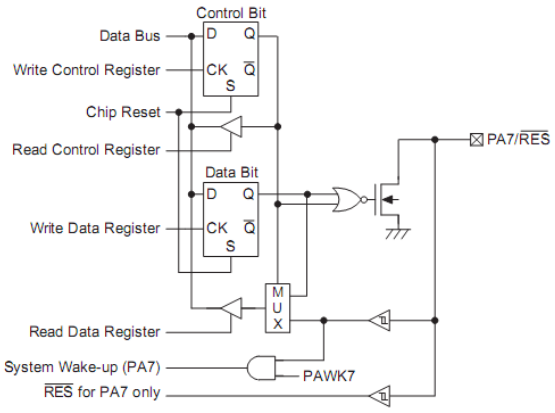
单片机为 A/D 转换器提供了 8 个输入口，并与输入/输出引脚共用。如果将引脚设为 A/D 输入，则 A/D 转换寄存器 ANCSR 中相应的位 PCRn 应置位。A/D 转换器没有相关的配置选项。如果引脚作为普通 I/O 脚用，可使用上拉电阻，而设置为 A/D 输入，则所有上拉电阻会自动断开。

### 输入/输出引脚结构

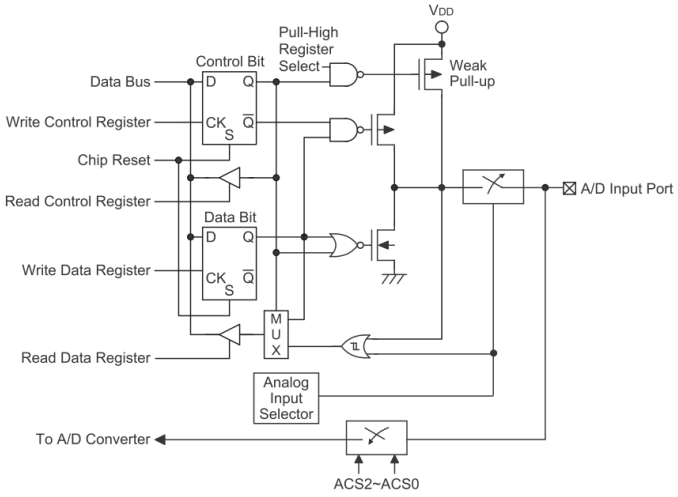
下图为输入/输出引脚的内部结构图。输入/输出引脚的准确逻辑结构图可能与此图不同，这里只是为了方便对功能的理解提供的一个参考。




**通用输入/输出端口**



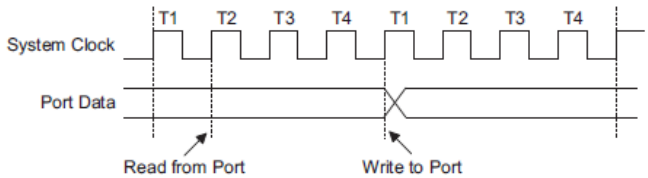

**PA7 NMOS 输入/输出端口**




**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。注意的是当使用SET [m].i、CLR [m].i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。单片机处于暂停模式时，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的转换的方式，也可设置PA口一个或者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。这些芯片具有一个 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 TPSC0~ TPSC2 来确定。内部时钟源可以通过 TMRC 寄存器的 TS 位来选择 $f_{SYS}$ 或 $f_{SYS}/4$。

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 TC 提供。每次外部引脚由高电平到低电平或者由低电平到高电平(由 TEG 位决定)进行转换时，计数器增加一。

## 定时/计数寄存器—TMR

定时/计数寄存器 TMR，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。

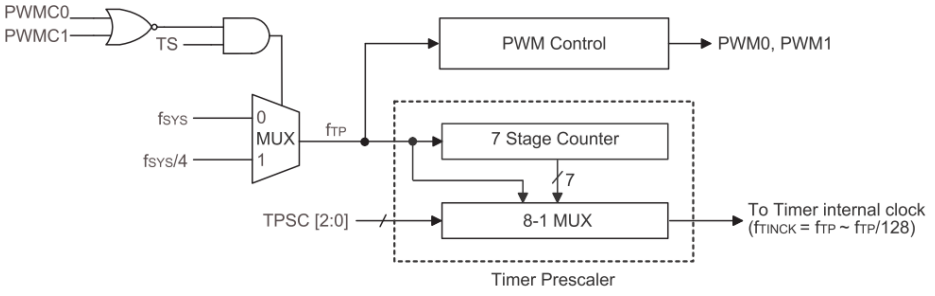
为了得到定时器的最大计算范围，预置寄存器需要先清为零。注意的是上电后预置寄存器处于未知状态。定时/计数器在关闭条件下，写数据到预置寄存器，会立即写入实际的定时器。而如果定时/计数器已经打开且正在计数，在这个周期内写入到预置寄存器的任何新数据将保留在预置寄存器，直到溢出发生时才被写入实际定时器。

## 定时/计数器控制寄存器—TMRC

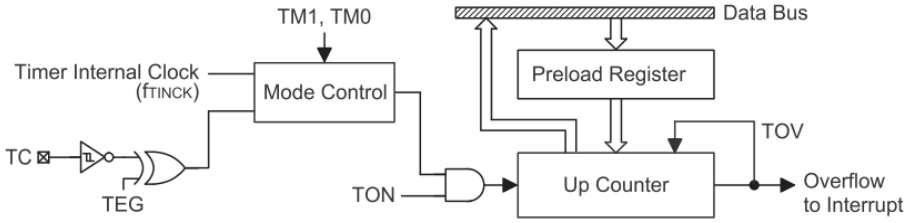
Holtek 单片机灵活的特性也表现在定时器的多功能上，定时/计数器能提供三种不同的工作模式，由相应的控制寄存器来选择定时/计数器的工作方式。

定时/计数控制寄存器为 TMRC，配合相应的 TMR 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 TM1/TM0，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 TON，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，TEG 位即 TMRC 寄存器的第 3 位将可用来选择上升沿或下降沿触发。




**Timer/PWM时钟源结构图**




**8位定时/计数器结构图**

注意：当PWM0/PWM1使能时，$f_{TP}$来自于$f_{SYS}$，TS无效。

- **TMRC 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TM1 | TM0 | TS | TON | TEG | TPSC2 | TPSC1 | TPSC0 |
| 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 **TM1**，**TM0**：Timer 工作模式选项
00：无可用模式
01：计数器模式
10：定时器模式
11：脉冲宽度测量模式

Bit 5 **TS**：定时器时钟源
0：$f_{SYS}$
1：$f_{SYS}$ /4
TS 用来选择 Timer、时基和 PWM 的时钟源 $f_{TP}$，如果 PWM 使能，$f_{TP}$ 选择 $f_{SYS}$，忽略 TS 的设置。

Bit 4 **TON**：定时/计数器使能
0：除能
1：使能

Bit 3 **TEG**：
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

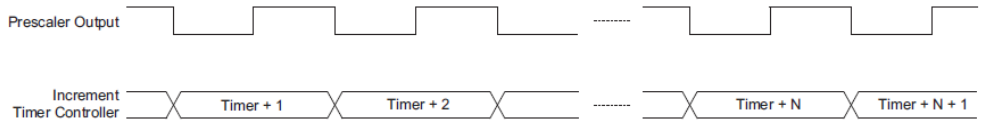
Bit 2~0 **TPSC2**，**TPSC1**，**TPSC0**：选择定时器预分频比
定时器内部时钟=
000：$f_{TP}$
001：$f_{TP}$/2
010：$f_{TP}$/4
011：$f_{TP}$/8
100：$f_{TP}$/16
101：$f_{TP}$/32
110：$f_{TP}$/64
111：$f_{TP}$/128

**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，位 TM1/TM0 必须分别设置为 1 和 0。

在定时器模式中，$f_{SYS}$ 或 $f_{SYS}$/4 被用来当定时器的输入时钟源。然而，该定时器时钟源可以被预分频器进一步分频，分频比是由定时器控制寄存器的 TPSC2~TPSC0 位来确定。定时器控制寄存器第 4 位，即 TON 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 INTC 中的位 TE 为 0，可以禁止计数器中断。



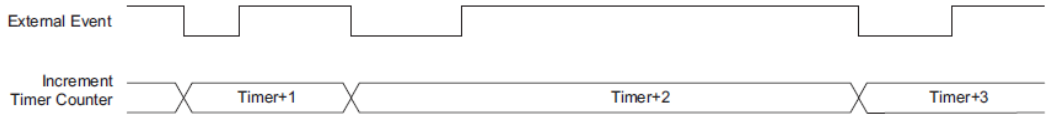

**定时器模式时序图**

**外部事件计数器模式**

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TC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，工作模式选择位TM1/TM0必须分别设置为0和1。

在外部事件计数模式，外部定时脚 TC 被用来当定时/计数器的时钟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 TON 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 TEG 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 TEG 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。值得注意的是，在外部事件计数模式下，当单片机工作在暂停模式时也保持对外部 TC 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（TEG=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，工作模式选择位TM1/TM0必须分别设置为1。

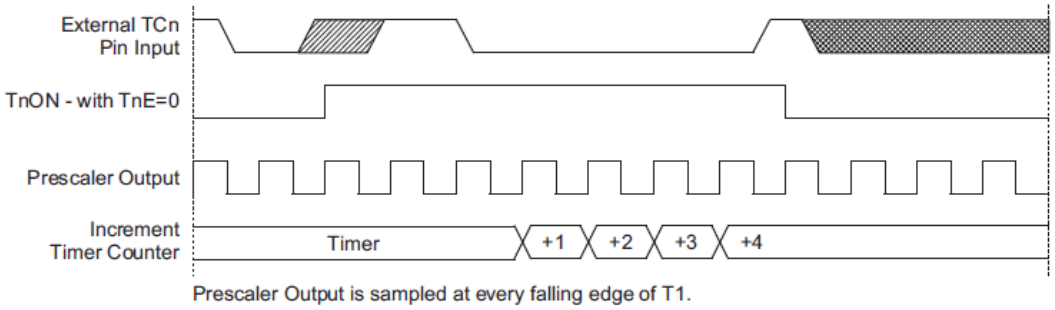
在脉冲宽度测量模式中，$f_{SYS}$ 或 $f_{SYS}/4$ 作为 8 位定时/计数器的内部时钟源，并可被预分频器进一步分频。分频比由预分频选择位 TPSC2~TPSC0，即定时控制寄存器的第 2~0 位来确定。在设置完定时/计数控制寄存器其它位，定时器控制寄存器第 4 位，即 TON 位需要设为逻辑高，才能使定时/计数器工作。然而，只有当在外部定时器引脚上接收到有效的逻辑转换时，定时/计数器才真正开始启动计数。

当定时控制寄存器第 3 位，即 TEG 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 TEG 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。注意的是，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 TC 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的信号脉冲将被忽略。直到使能位被程序重新置高，定时器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。

要注意的是在这种模式下，定时/计数器是通过外部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 TC 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。




**脉冲宽度测量时序图（TEG=0）**

### 预分频器

TMRC 寄存器的 TPSC0~TPSC2 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。

### 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者脉冲宽度测量模式。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，与内部系统时钟或者定时器时钟不同步。

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

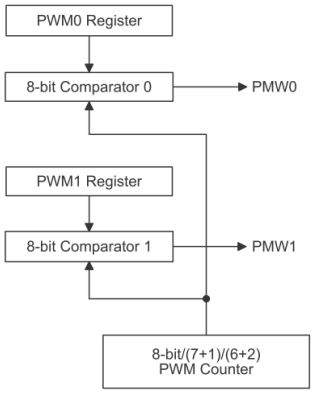
这个例子说明了如何设置定时/计数器的寄存器，如何设置和控制中断。另外还需注意的是，怎样通过寄存器的第 4 位来启停定时/计数器。此应用范例设置定时/计数器为定时模式，时钟来源于内部的系统时钟。

- **定时器编程应用范例**

```
org 04h                          ; external interrupt vector
org 08h                          ; Timer Counter interrupt vector
jmp tmrint                       ; jump here when Timer overflows
:          :
org 20h                          ; main program
:          :
;internal Timer interrupt routine
tmrint：
:
; Timer main program placed here
:
:
begin:
;setup Timer register
mov a,09bh                 ; setup Timer preload value
mov tmr,a
mov a,081h                 ; setup Timer control register
mov tmrc,a                 ; timer mode and prescaler set to /2
;setup interrupt register
mov a,00dh                 ; enable master interrupt and timer interrupt
mov intc,a
:       :
set tmrc.4                  ; start Timer
:       :
```

## 脉冲宽度调制

单片机具有两个 8 位的脉冲宽度调制 PWM 功能。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方框图**

### PWM 操作

在数据存储器中，单片机为每一个 PWM 都指定了对应的寄存器，称为 PWMn 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，每个调制周期被细分为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 寄存器来选择每个 PWM 通道所需的模式和开关控制。注意的是，当使用 PWM 时，只要将所需的值写入相应的 PWMn 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 $f_{SYS}$。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 周期频率与 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<br>$f_{SYS}$ /128用于7+1模式 | $f_{SYS}$ /256 | [PWM]/256 |

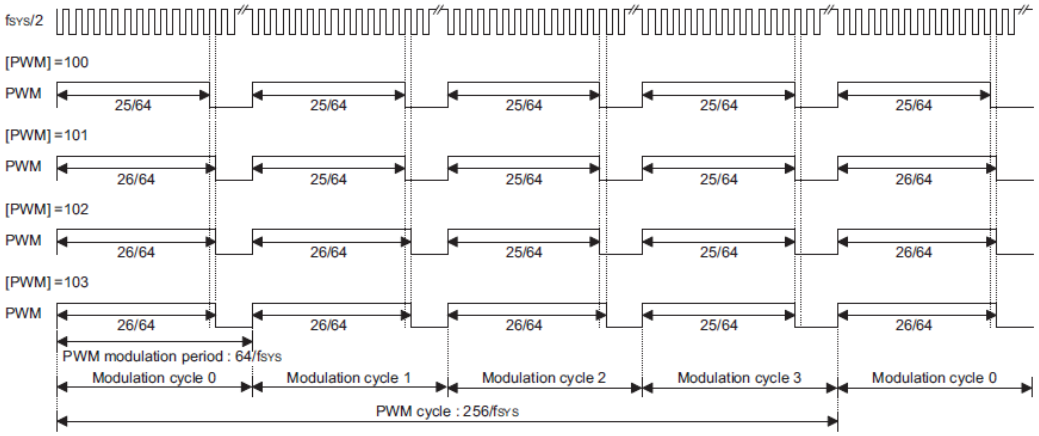
**6+2 PWM 模式**

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。

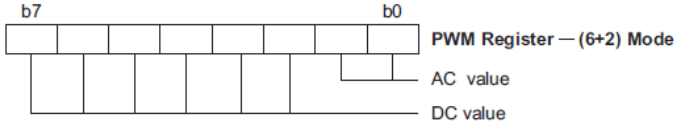
| 参数 | AC（0~3） | DC（占空比） |
|---|---|---|
| 调制周期i<br>（i=0~3） | i<AC | $\frac{DC+1}{64}$ |
| | i≥AC | $\frac{DC}{64}$ |

**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**

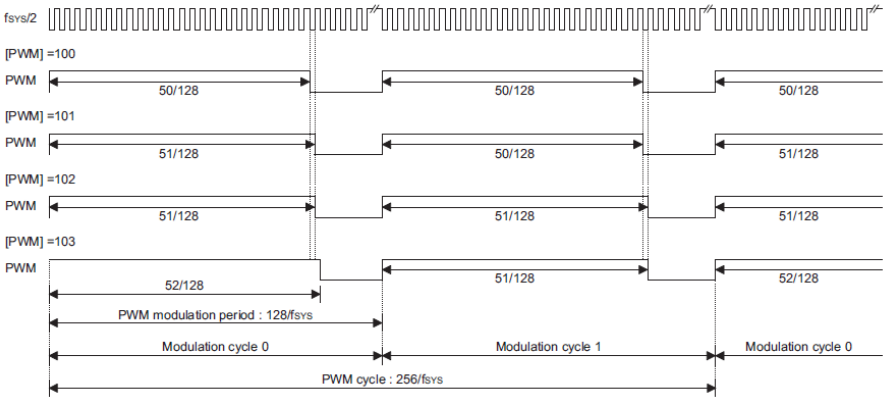
通过一个 8 位的 PWM 寄存器控制，每个完整的 PWM 周期由 256 个时钟周期组成。在 7+1PWM 模式中，每个 PWM 周期又被分成两个独立的子周期，称为调制周期 0~调制周期 1，在表格中以“i”表示。两个子周期各包含 128 个时钟周期。在这个模式下，得到以 2 为因数增加的调制频率。8 位的 PWM 寄存器被分成两个部分，这个寄存器的值表明整个 PWM 波形的占空比。第一部分包括第 1 位~第 7 位，表示 DC 值，第二部分为第 0 位，表示 AC 值。在 7+1 PWM 模式中，两个调制子周期

的占空比，分别如下表所示。

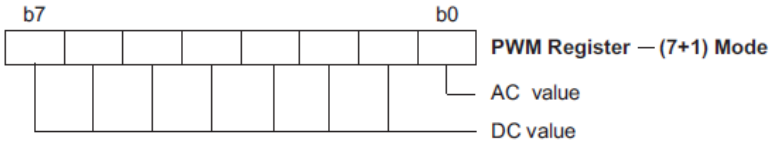
| 参数 | AC（0~1） | DC（占空比） |
|---|---|---|
| 调制周期i<br>（i=0~1） | i<AC | DC+1<br>128 |
| | i≥AC | DC<br>128 |

**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2个单独的调制周期0～1以及AC值与PWM值的关系。




**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 和 PC3 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 和 PCC.3 也需写 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 PWMn 寄存器之后，将“1”写入到 PA.4 和 PC.3 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 和 PC.3 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。需注意的是，如果 CTRL0 寄存器选择 PWM 功能，但是对 PAC 或 PCC 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
```

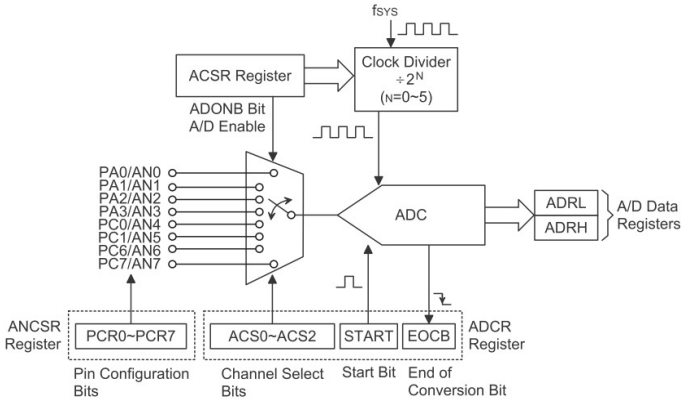
# A/D 转换器

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 转换器的芯片，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。在转换过程发生后，单片机可以直接读取这两个寄存器，以获得数字化的转换值。只有高字节寄存器 ADRH 完全利用了 8 位。而低字节寄存器 ADRL 只利用了 8 位中的 4 位，它包含的 12 位转换值中低 4 位。下表中，D0~D11 是 A/D 换转数据结果位。

| 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 |
| 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 | x | x | x | x | x | x | x | x | x | x | x | x | — | — | — | — |

D11~ D0 是 A/D 换转数据结果位。“—”读为“0”。 “x”未知

## A/D 转换控制寄存器 –ADCR，ACSR，ANCSR

寄存器 ADCR，ACSR，ANCSR 用来控制 A/D 转换器的功能和操作。这些 8 位的寄存器定义包括选择连接至内部 A/D 转换器的模拟通道（该引脚即可作为模拟输入脚，又可作为普通的 I/O 输入），A/D 时钟源，并控制和监视 A/D 转换器的开始和转换结束状态。

寄存器 ADCR 的 ACS2~ACS0 位定义了通道编号。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。ACS2~ACS0 位的功能决定选择哪个模拟输入通道被连接到内部 A/D 转换器。

• **ADCR 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 转换位
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 5~3 未定义，读为“0”

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

• **ACSR 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | ADONB | — | — | — | ADCS2 | ADCS1 | ADCS0 |
| R/W | — | R/W | — | — | — | R/W | R/W | R/W |
| POR | — | 0 | — | — | — | 0 | 0 | 0 |

Bit 7 未定义，读为“0”

Bit 6 **ADONB**：ADC 模块电源开/关控制位
0：ADC 模块电源开
1：ADC 模块电源关
注：建议在进入暂停模式前，设置 ADONB=1 以减少功耗。

Bit 5~3 未定义，读为“0”。

Bit 2~0 **ADCS2~ADCS0**：选择 A/D 时钟源
000：$f_{SYS}/2$
001：$f_{SYS}/8$
010：$f_{SYS}/32$
011：未定义，不使用
100：$f_{SYS}$
101：$f_{SYS}/4$
110：$f_{SYS}/16$
111：未定义，不使用

• **ANCSR 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 定义 PC7 是否为 A/D 输入
0：不是 A/D 输入
1：A/D 输入，AN7

Bit 6 定义 PC6 是否为 A/D 输入
0：不是 A/D 输入
1：A/D 输入，AN6

Bit 5 定义 PC1 是否为 A/D 输入
0：不是 A/D 输入
1：A/D 输入，AN5

Bit 4 定义 PC0 是否为 A/D 输入
0：不是 A/D 输入
1：A/D 输入，AN4
Bit 3 定义 PA3 是否为 A/D 输入
0：不是 A/D 输入
1：A/D 输入，AN3
Bit 2 定义 PA2 是否为 A/D 输入
0：不是 A/D 输入
1：A/D 输入，AN2
Bit 1 定义 PA1 是否为 A/D 输入
0：不是 A/D 输入
1：A/D 输入，AN1
Bit 0 定义 PA0 是否为 A/D 输入
0：不是 A/D 输入
1：A/D 输入，AN0

注：如果 PCR7~PCR0 位都置 0，则所有 PA0~PA3, PC0, PC1, PC6 和 PC7 引脚都作为普通 I/O 使用

ANCSR 控制寄存器中的 PCR7~PCR0 位，用来定义 PA0~PA3，PC0~PC1，PC6~PC7 中的哪些引脚为 A/D 转换器的模拟输入，哪些引脚作为普通 I/O 口使用。

**A/D 操作**

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换转器的开/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 转换器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~ADCS0 位决定。

控制 A/D 转换模块电源开/关，通过 ADONB 位来完成。

虽然 A/D 时钟源是由系统时钟 $f_{SYS}$，ADCS2~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 0.5μs，当系统时钟速度等于或超过 4MHz 时就必须小心。如果系统时钟速度为 4MHz 时，ADCS2~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，因为它们的 A/D 转换时钟周期小于规定的最小值。

| $f_{SYS}$ | 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 或 PC 端口的 I/O 引脚共用。使用 ANCSR 寄存器中的 PCR7~PCR0 位，可以将它们设置为普通输入/输出脚或 A/D 转换器模拟输入脚。通过这种方式，引脚的功能可由程序来控制，从普通的 I/O 操作功能到模拟输入，反过来也一样。当输入引脚作为普通的 I/O 引脚使用时，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，PAC 或 PCC 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，端口控制寄存器的状态将被重置。

**A/D 转换步骤**

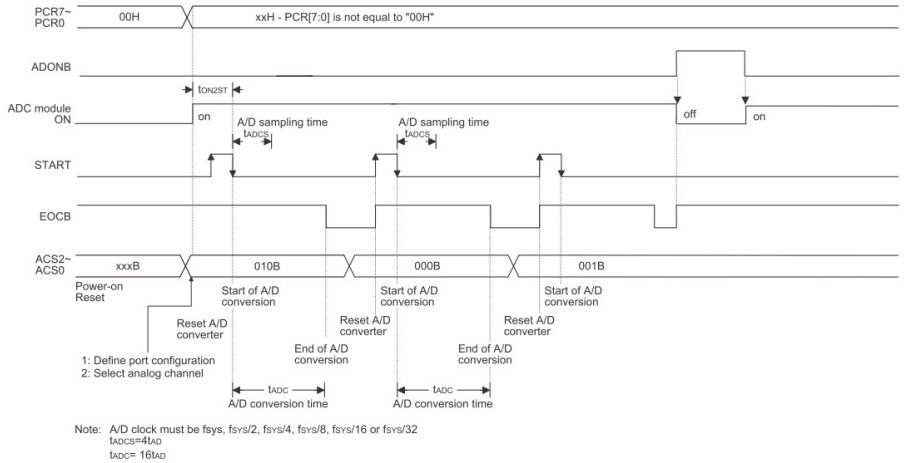
下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1
  通过 ACSR 寄存器中的 ADCS2~ADCS0 位，选择所需的 A/D 转换时钟。
- 步骤 2
  清零 ACSR 寄存器中的 ADONB 位使能 A/D。
- 步骤 3
  通过 ADCR 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 4
  通过 ANCSR 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 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。总中断控制位 EMI 需要置位为“1”，以及 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满，则程序等待 A/D 中断发生。

注：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 转换的设置和选项功能由应用程序控制，因为 A/D 转换器中没有配置选项。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。




**A/D 转换时序图**

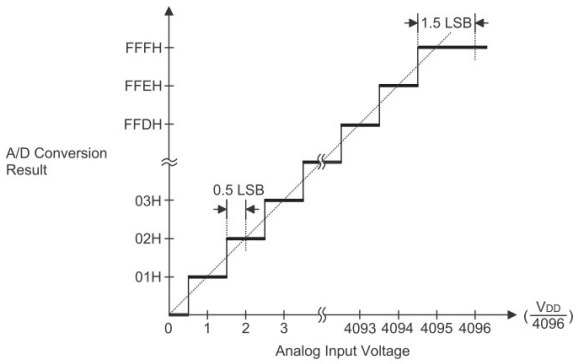
**编程注意事项**

在编程序时，必须特别注意寄存器 ANCSR 中的 A/D 转换通道选择位 PCR[7:0]。如果这些位全部为零，那么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 口使用。这样可以减少 A/D 转换部分的功耗。通过设置 ACSR 寄存器中的 ADONB 为高，关闭 A/D 内部电路以减少电源功耗，在电路灵敏的应用中，这种方法很重要。

**A/D 转换功能**

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。

需要注意的是，为了减小误差，在 A/D 转换器的输入脚加上 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,00101011B         ;setup ANCSR to configure the Ports as A/D inputs
    mov     ANCSR,a
    mov     a,00000000B         ;setup ADCR to select AN0 to be connected to the A/D converter
    mov     ADCR,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
```

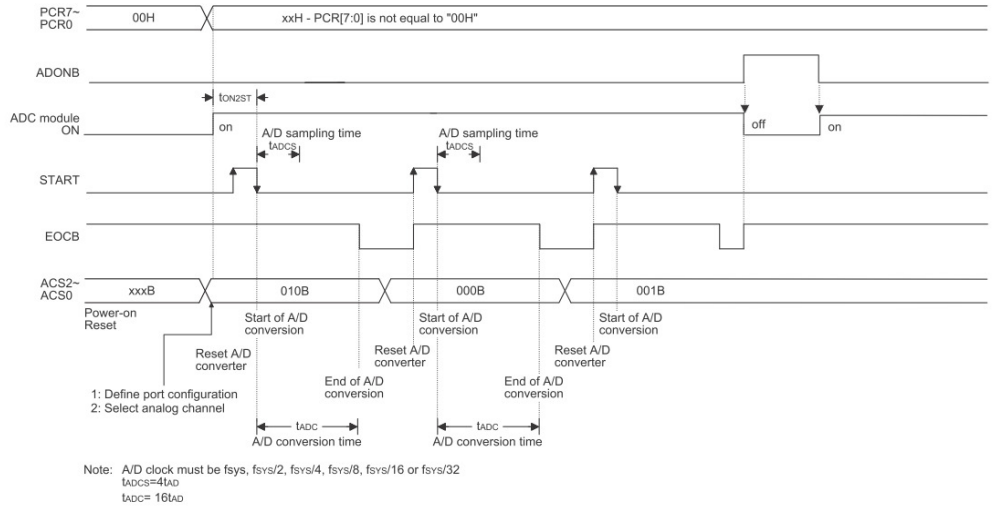
```
    mov   adrh_buffer,a          ; save result to user defined register
    :
    jmp   start_conversion       ; start next A/D conversion
```

注：要关闭 ADC 模式，需要置 ADONB 为高位

**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,00101011B           ;setup ANCSR to configure the Ports as A/D inputs
    mov    ANCSR,a
    mov    a,00000000B           ;setup ADCR to select AN0 to be connected to the A/D converter
    mov    ADCR,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：要关闭 ADC 模式，需要置 ADONB 为高位




**A/D 转换时序图**

# 中断

中断是单片机一个重要功能。当外部事件或内部功能如定时器模块或外部中断有效，系统会暂时中止当前的程序而转到执行相对应的中断服务程序。

此系列每一款单片机均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 INTC 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。

**•INTC寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | ADF | TF | INTF | ADE | TE | 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 转换中断请求标志

0：无效

1：有效

Bit 5　**TF**：定时/计数器中断请求标志

0：无效

1：有效

Bit 4　**INTF**：外部中断请求标志

0：无效

1：有效

Bit 3　**ADE**：A/D 转换中断使能位

0：除能

1：使能

Bit 2　**TE**：定时/计数器中断使能位

0：除能

1：使能

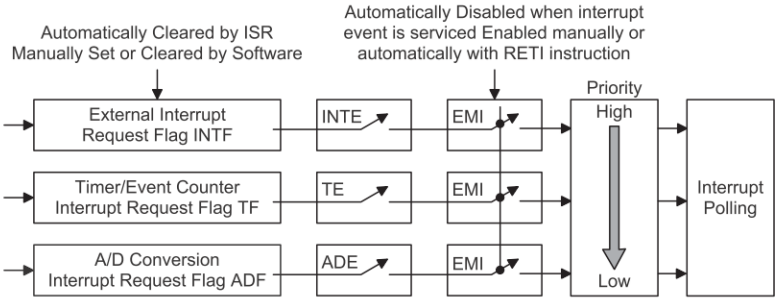
Bit 1 **INTE**：外部中断使能位
0：除能
1：使能

Bit 0 **EMI**：总中断使能位
0：除能
1：使能

**中断操作**

定时/计数器溢出、完整的 A/D 转换或者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，并置位相应的请求标志位。如果相应的中断允许，系统将要执行指令的下一条地址压入堆栈，并将相应的中断向量地址加载至 PC 中，然后从此向量取下一条指令。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。中断服务程序需以 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。

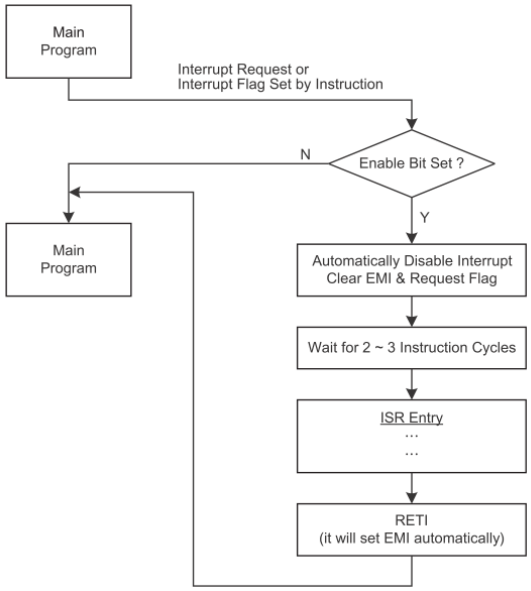
各个中断使能位以及相应的请求标志位，以及优先级的顺序如下图所示。




**中断示意图**

一旦中断子程序被响应，系统将自动清除 EMI 位，所有其它的中断将被屏蔽，这个方式可以防止任何进一步的中断嵌套。其它中断请求可能发生在此期间，虽然中断不会立即响应，但是中断请求标志位会被记录。如果某个中断服务子程序正在执行时，有另一个中断要求立即响应，那么 EMI 位应在程序进入中断子程序后置位，以允许此中断嵌套。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。如果要求立刻动作，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。而单片机在暂停模式被唤醒时，需要插入3个指令周期，程序才能跳转到响应的中断向量地址。




**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、上升沿触发或者两者都可以触发，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 INTC 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，堆栈未满且外部中断产生时，将调用位于地址 04H 处的子程序。当响应外部中断服务子程序时，中断请求标志位 INTF 会自动复位且 EMI 位会被清零以除能其它中断。注意，即使此引脚被用作外部中断输入，其配置选项中的上拉电阻仍保持有效。

**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 TE 需先被置位。当定时/计数器发生溢出，相应的中断请求标志位 TF 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 TF 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，总中断使能位 EMI 和相关的中断使能位 ADE 需先被置位。当 A/D 转换器中断请求标志位 ADF 被置位，即 A/D 转换过程完成时，中断请求发生。当中断使能，堆栈未满且 A/D 转换动作结束时，将调用它们各自的中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 ADF 会自动复位且 EMI 位也会被清零以除能其它中断。当中断与其它中断共用向量地址时，要通过配置选项来选择。

**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。

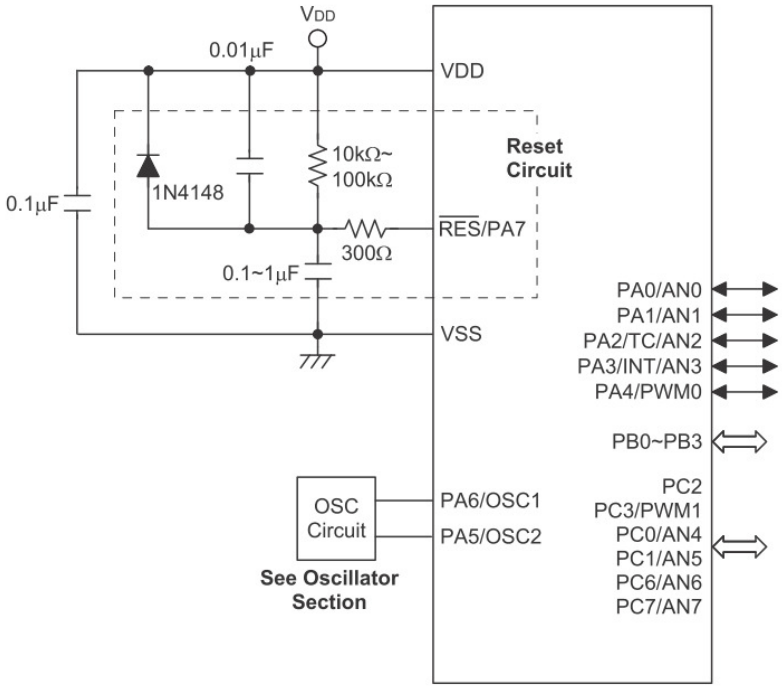
所有的中断都具有将处于暂停模式的单片机唤醒的功能。但只有程序计数器被压入堆栈中，一旦中断服务程序使寄存器和状态寄存器中的内容发生改变，则会破坏想要的控制序列，因此需要事先将这些数据保存起来。

## 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：

| 序号 | 选项 |
|---|---|
| **振荡器选项** | |
| 1 | 系统振荡器配置：HXT 或 HIRC |
| 2 | 高速内部 RC：4MHz，8MHz 或 12MHz |
| **复位脚选项** | |
| 3 | 引脚功能：$\overline{\text{RES}}$或 PA7 |
| **LVR 选项** | |
| 4 | LVR 电压选择：2.1V，3.15V 或 4.2V |

## 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。在盛群单片机中，提供了丰富且灵活的指令集，共超过 60 条，程序设计师可以事半功倍地实现他们的应用。

为了更容易地了解各式各样的指令码，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”。对于跳转命令必须注意的是，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，两者的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 |
| **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 |

| 助记符 | | 说明 | 指令周期 | 影响标志位 |
|---|---|---|---|---|
| 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** | | | | |
| 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 |
| 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 | 1 | TO, PDF |
| CLR | WDT2 | 预清除看门狗定时器 2 | 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 1 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 将指定数据存储器、累加器和进位标志位的内容相加后，把结果储存回累加器。
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 将指定数据存储器、累加器和进位标志位的内容相加后，把结果储存回指定数据存储器。
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 将指定数据存储器和累加器的内容相加后，把结果储存回累加器。
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 将累加器和立即数的内容相加后，把结果储存回累加器。
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 将指定数据存储器和累加器的内容相加后，把结果储存回指定数据存储器。
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 将存在累加器和指定数据存储器中的数据作AND的运算，然后把结果储存回累加器。
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 将存在累加器中的数据和立即数作AND的运算，然后把结果储存回累加器。
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 将存在指定数据存储器和累加器中的数据作AND的运算，然后把结果储存回数据存储器。
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 指定数据存储器中的每一位均清除为0。
功能表示 [m] ← 00H
影响标志位 无

**CLR [m].i** Clear bit of Data Memory
指令说明 指定数据存储器中的i位清除为0。
功能表示 [m].i ← 0
影响标志位 无

**CLR WDT** Clear Watchdog Timer
指令说明 将TO , PDF标志位和WDT全都清零。
功能表示 WDT cleared
TO ← 0
PDF ← 0
影响标志位 TO , PDF

**CLR WDT1** Pre-clear Watchdog Timer
指令说明 将TO、PDF标志位和WDT全都清零，请注意此指令要结合CLR WDT2一起动作且必须交替执行才有作用，重复执行此项指令而没有与CLR WDT2交替执行将无任何作用。
功能表示 WDT cleared
TO ← 0
PDF← 0
影响标志位 TO , PDF

**CLR WDT2** Pre-clear Watchdog Timer
指令说明 将TO、PDF标志位和WDT全都清零，请注意此指令要结合CLR WDT1一起动作且必须交替执行才有作用，重复执行此项指令而没有与CLR WDT1交替执行将无任何作用。
功能表示 WDT cleared
TO ← 0
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 将存在累加器中的内容数值转换为BCD（二进制转成十进制）数值，如果低4位大于9或AC标志位被置位，则在低4位加上一个6，不然低4位的内容不变，如果高4位大于9或C标志位被置位，则在高4位加上一个6，十进制的转换主要是依照累加器和标志位状况，分别加上00H、06H、60H或66H，只有C标志位也许会被此指令影响，它会指出原始BCD数是否大于100，并可以进行双精度十进制数相加。

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 将在指定数据存储器内的数据减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 将在指定数据存储器内的数据减1，把结果储存回累加器且数据存储器中的内容不变。
功能表示 ACC ← [m] – 1
影响标志位 Z

**HALT** Enter power down mode
指令说明 此指令停止程序的执行并且关闭系统时钟，但数据存储器和寄存器的内容仍被保留，WDT和预分频器被清零，暂停标志位PDF被置位且WDT溢出标志位TO被清零。
功能表示 TO ← 0
PDF ← 1
影响标志位 TO , PDF

**INC [m]** Increment Data Memory
指令说明 将指定数据存储器内的数据加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 将指定数据存储器内的数据加1，把结果储存回累加器且数据存储器中的内容不变。
功能表示 ACC ← [m] + 1
影响标志位 Z

**JMP addr** Jump unconditionally
指令说明 程序计数器的内容被指定地址所取代，程序由新地址继续执行，当新地址被加载时，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中。
功能表示 ACC← [m]
影响标志位 无

**MOV A, x** Move immediate data to ACC
指令说明 将立即数载入至累加器中。
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 将存在累加器和指定数据存储器中的数据作OR的运算，然后把结果储存回累加器。
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 将存在累加器中的数据和立即数作OR的运算，然后把结果储存回累加器。
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和累加器中的数据作OR的运算，然后把结果储存回数据存储器。
功能表示 [m] ← ACC“OR”[m]
影响标志位 Z

**RET** Return from subroutine
指令说明 将堆栈区的数据取回至程序计数器，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 将堆栈区的数据取回至程序计数器且累加器载入立即数，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 将堆栈区的数据取回至程序计数器且中断功能通过EMI位重新被使能，EMI是控制中断使能的主中断位(寄存器INTC的第0位)，如果在执行RETI指令之前还有中断未被响应，则这个中断将在返回主程序之前被响应。
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向左移1个位，且第7位移回第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向左移1个位，且第7位移回第0位，而移位的结果储存回累加器且数据存储器中的内容不变。
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位向左移1个位，第7位取代进位位且原本的进位标志位移至第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

**RLCA [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位向左移1个位，第7位取代进位位且原本的进位标志位移至第0位，而移位的结果储存回累加器且数据存储器中的内容不变。
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向右移1个位，且第0位移回第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向右移1个位，且第0位移回第7位，而移位的结果储存回累加器且数据存储器中的内容不变。
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位向右移1个位，第0位取代进位位且原本的进位标志位移至第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位向右移1个位，第0位取代进位位且原本的进位标志位移至第7位，而移位的结果储存回累加器且数据存储器中的内容不变。
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 将累加器中的数据与指定数据存储器内容和进位标志位的反相减，把结果储存回累加器。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

**SBCM A, [m]** Subtract Data Memory from ACC with Carry and result in Data Memory
指令说明 将累加器中的数据与指定数据存储器内容和进位标志位的反相减，把结果储存回数据存储器。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

**SDZ [m]** Skip if Decrement Data Memory is 0
指令说明 将指定数据存储器的内容先减去1后，如果结果为0，则程序计数器再加1跳过下一条指令，由于取得下一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，则程序继续执行下面的指令。
功能表示 [m] ← [m] – 1
Skip if [m] = 0
影响标志位 无

**SDZA [m]** Skip if decrement Data Memory is zero with result in ACC
指令说明 将指定数据存储器的内容先减去1后，如果结果为0，则程序计数器再加1 跳过下一条指令，此结果会被储存回累加器且指定数据存储器中的内容不变，由于取得下一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，则程序继续执行下面的指令。
功能表示 ACC ← [m] – 1
Skip if ACC = 0
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个位置位为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 将指定数据存储器的内容先加上1后，如果结果为0，则程序计数器再加1 跳过下一条指令，由于取得下一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，则程序继续执行下面的指令。
功能表示 [m] ← [m] + 1
Skip if [m] = 0
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先加上1后，如果结果为0，则程序计数器再加1跳过下一条指令，此结果会被储存回累加器且指定数据存储器中的内容不变，由于取得下一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，则程序继续执行下面的指令。
功能表示 ACC← [m] + 1
Skip if ACC = 0
影响标志位 无

**SNZ [m].i** Skip if bit i of Data Memory is not 0
指令说明 如果指定数据存储器的第i位不为0，则程序计数器再加1跳过下一条指令，由于取得下一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，程序继续执行下面的指令。
功能表示 Skip if [m].i ≠ 0
影响标志位 无

**SUB A, [m]** Subtract Data Memory from ACC
指令说明 将累加器中内容减去指定数据存储器的数据，把结果储存回累加器。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

**SUBM A, [m]** Subtract Data Memory from ACC with result in Data Memory
指令说明 将累加器中内容减去指定数据存储器的数据，把结果储存回数据存储器。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

**SUB A, x** Subtract immediate Data from ACC
指令说明 将累加器中内容减去立即数，把结果储存回累加器。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

**SWAP [m]** Swap nibbles of Data Memory
指令说明 将指定数据存储器的低4位与高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，然后把结果储存回累加器且数据存储器的内容不变。
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 如果指定数据存储器的内容为0，则程序计数器再加1跳过下一条指令，由于取得下一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，程序继续执行下面的指令。
功能表示 Skip if [m] = 0
影响标志位 无

**SZA [m]** Skip if Data Memory is 0 with data movement to ACC
指令说明 将指定数据存储器的内容复制到累加器，如果值为0，则程序计数器再加1跳过下一条指令，由于取得下一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，程序继续执行下面的指令。
功能表示 ACC ← [m]
Skip if [m] = 0
影响标志位 无

**SZ [m].i** Skip if bit i of Data Memory is 0
指令说明 如果指定数据存储器第i 位为0，则程序计数器再加1 跳过下一条指令，由于取得下一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，程序继续执行下面的指令。
功能表示 Skip if [m].i = 0
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 将存在累加器和指定数据存储器中的数据作XOR的运算，然后把结果储存回累加器。
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 将存在指定数据存储器和累加器中的数据作XOR的运算，然后把结果储存回数据存储器。
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 将存在累加器中的数据和立即数作XOR的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“XOR”x
影响标志位 Z

# 封装信息

**20-pin DIP(300mil)外形尺寸**

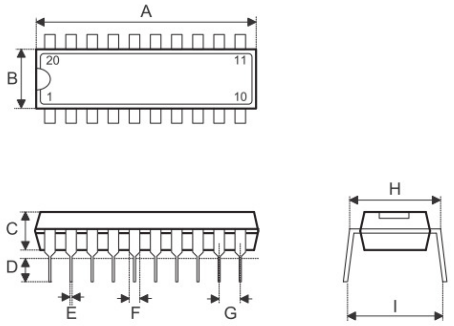



**Fig1. Full Lead Packages**

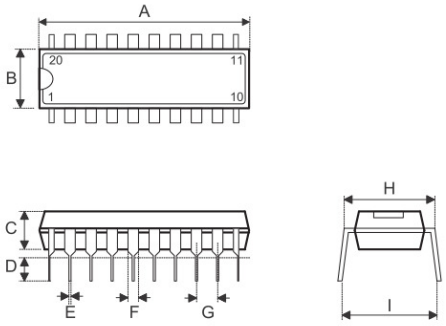



**Fig2. 1/2 Lead Packages**

• **MS-001d(见 fig1)**

| 符号 | 尺寸(单位：inch) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| 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 | — |

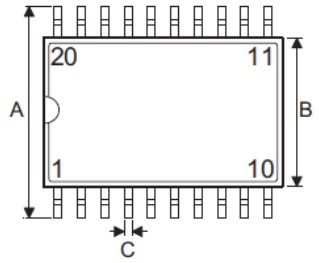
| 符号 | 尺寸(单位：mm) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| A | 24.89 | — | 26.92 |
| B | 6.10 | — | 7.11 |
| C | 2.92 | — | 4.95 |
| D | 2.92 | — | 3.81 |
| E | 0.36 | — | 0.56 |
| F | 1.14 | — | 1.78 |
| G | — | 2.54 | — |
| H | 7.62 | — | 8.26 |
| I | — | 10.92 | — |

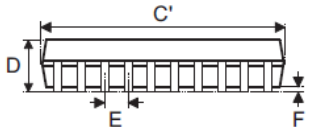
- **MO-095a(见 fig2)**

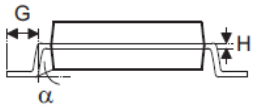
| 符号 | 尺寸(单位：inch) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| A | 0.945 | — | 0.985 |
| B | 0.275 | — | 0.295 |
| C | 0.120 | — | 0.150 |
| D | 0.110 | — | 0.150 |
| E | 0.014 | — | 0.022 |
| F | 0.045 | — | 0.060 |
| G | — | 0.100 | — |
| H | 0.300 | — | 0.325 |
| I | — | 0.430 | — |

| 符号 | 尺寸(单位：mm) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| 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**



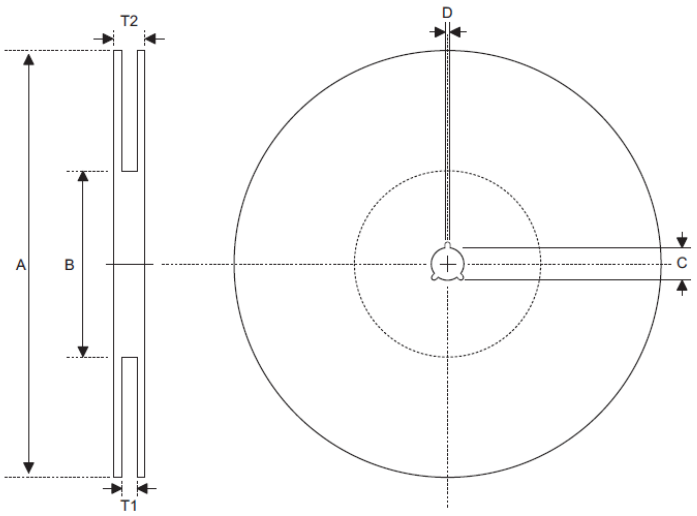

• **MS-013**

| 符号 | 尺寸（ 单位：inch） | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| 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） | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| 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° |

# 包装带和卷轴规格

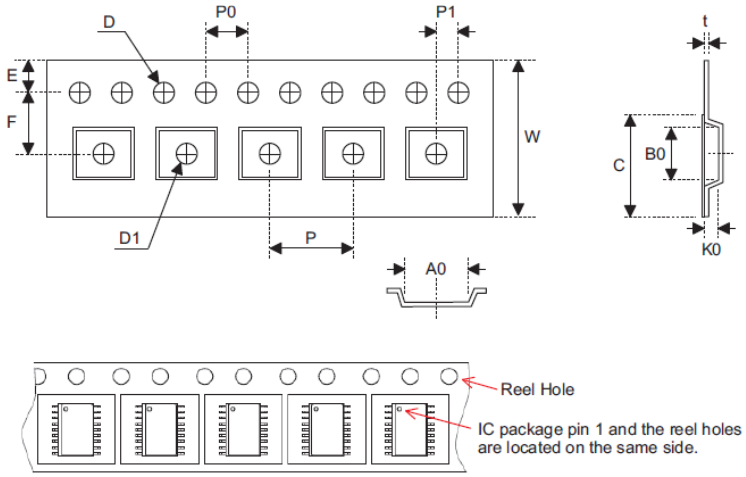
## 卷轴尺寸




**SOP 20W (300mil)**

| 符号 | 说明 | 尺寸(mm) |
|---|---|---|
| A | 卷轴外圈直径 | 330.0±1.0 |
| B | 卷轴内圈直径 | 100.0±1.5 |
| C | 轴心直径 | $13.0^{+0.5/-0.2}$ |
| D | 缝宽 | 2.0±0.5 |
| T1 | 轮缘宽 | $24.8^{+0.3/-0.2}$ |
| T2 | 卷轴宽 | 30.2±0.2 |

运输带尺寸




**SOP 20W(300mil)**

| 符号 | 说明 | 尺寸(mm) |
|---|---|---|
| W | 运输带宽 | $24.0^{+0.3/-0.1}$ |
| P | 空穴间距 | 12.0±0.1 |
| E | 穿孔位置 | 1.75±0.10 |
| F | 空穴至穿孔距离(宽度) | 11.5±0.1 |
| D | 穿孔直径 | $1.5^{+0.1/-0.0}$ |
| D1 | 空穴中的小孔直径 | $1.50^{+0.25/-0.00}$ |
| 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 | 10.8±0.1 |
| B0 | 空穴宽 | 13.3±0.1 |
| K0 | 空穴深 | 3.2±0.1 |
| t | 传送带厚度 | 0.30±0.05 |
| C | 覆盖带宽度 | 21.3±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

**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