高精度 HIRC Flash 单片机

# HT68F2420

版本 : V1.01　日期 : 2021-07-26

www.holtek.com

# 目录

# 特性

## CPU 特性

- 工作电压：
  - ♦ $f_{SYS}$=4MHz：1.8V~5.5V
- $V_{DD}$=5V，系统时钟为 4MHz 时，指令周期为 1μs
- 提供暂停和唤醒功能，以降低功耗
- 振荡器：
  - ♦ 内部高速 4MHz RC 振荡器 – HIRC
  - ♦ 内部低速 32kHz RC 振荡器 – LIRC
- 多种工作模式：快速、低速、空闲和休眠
- 完全集成的内部振荡器，无需外接元器件
- 所有指令都可在 1~2 个指令周期内完成
- 查表指令
- 61 条功能强大的指令系统
- 2 层堆栈
- 位操作指令

## 周边特性

- Flash 程序存储器：1K×13
- 数据存储器：32×8
- 看门狗定时器功能
- 16 个双向 I/O 口
- 1 个可编程的载波输出 – 使用 9-bit 定时器
- 高驱动电流输出引脚
- 单个时基功能，可提供固定时间的中断信号
- 低电压复位功能
- 封装类型：8-pin SOP、16/20-pin NSOP、20-pin SSOP

# 开发工具

为加快产品开发并简化单片机参数设置，Holtekt 提供相关开发工具，用户可通过以下链接下载：

https://www.holtek.com.cn/ESK-IRRC-R00

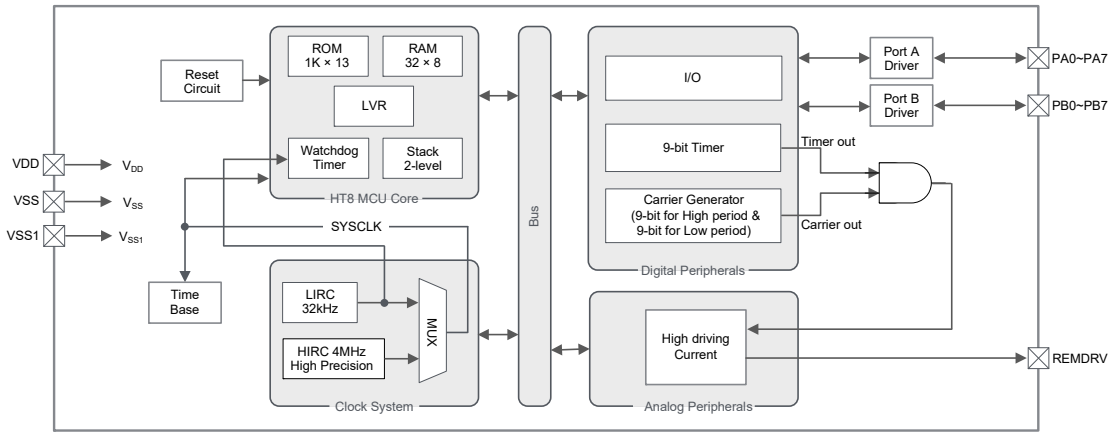
# 概述

HT68F2420 是一款 8 位具有高性能精简指令集的 Flash 单片机，适用于各种红外线遥控器及红外线传输相关产品应用。该单片机具有一系列功能和特性，其 Flash 存储器可多次编程的特性给客户提供了较大的方便。除了 Flash 存储器，该单片机还具有一个 RAM 数据存储器可由用户进行读取和写入。内部看门狗定时器保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电

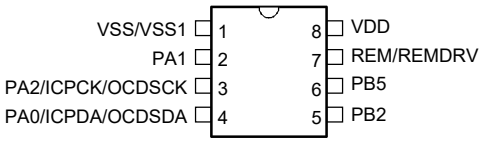
磁干扰环境下可靠地运行。

该单片机具有使用灵活的 9-bit 定时器可用于产生红外载波输出，时基功能可产生固定时间中断信号。同时提供了内部高速和低速振荡器功能，内建完整的系统振荡器，无需外接元器件。包含 I/O 使用灵活、高驱动电流输出能力和其它特性确保了该单片机可以广泛应用于各种需高精度时钟，定时功能以及大驱动电流的红外遥控器和定时器等应用产品中。

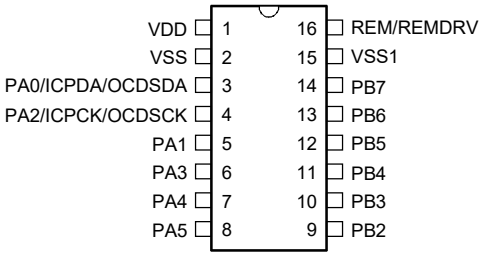
## 方框图



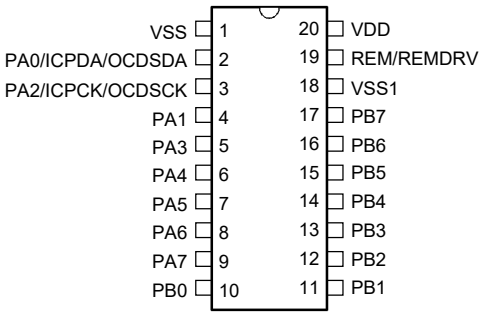

## 引脚图




**HT68F2420/HT68V2420**
**8 SOP-A**




**HT68F2420/HT68V2420**
**16 NSOP-A**




**HT68F2420/HT68V2420**
**20 NSOP/SSOP-A**

注：1. 在较小封装中可能含有未引出的引脚，需合理设置其状态以避免输入浮空造成额外耗电，详见“待机电流注意事项”和“输入 / 输出端口”章节。
2. PA2 和 PA0 引脚共用的 OCDSCK 和 OCDSDA 引脚功能仅存在于 OCDS EV 芯片 HT68V2420。

## 引脚描述

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。下方表格针对最大封装引脚进行说明，有部分引脚在小封装类型中未出现。

| 引脚名称 | 功能 | OPT | I/T | O/T | 描述 |
|---|---|---|---|---|---|
| PA0/ICPDA/OCDSDA | PA0 | PAPU<br>PAWU | ST | CMOS | 通用 I/O 口。通过寄存器使能上拉电阻和唤醒功能。 |
| | ICPDA | — | ST | CMOS | ICP 数据 / 地址引脚 |
| | OCDSDA | — | ST | CMOS | OCDS 数据 / 地址引脚，仅用于 EV 芯片 |
| PA1 | PA1 | PAPU<br>PAWU | ST | CMOS | 通用 I/O 口。通过寄存器使能上拉电阻和唤醒功能。 |
| PA2/ICPCK/OCDSCK | PA2 | PAPU<br>PAWU | ST | CMOS | 通用 I/O 口。通过寄存器使能上拉电阻和唤醒功能。 |
| | ICPCK | — | ST | — | ICP 时钟引脚 |
| | OCDSCK | — | ST | — | OCDS 时钟引脚，仅用于 EV 芯片 |
| PA3~PA7 | PA3~PA7 | PAPU<br>PAWU | ST | CMOS | 通用 I/O 口。通过寄存器使能上拉电阻和唤醒功能。 |
| PB0~PB7 | PB0~PB7 | PBPU<br>PBWU | ST | CMOS | 通用 I/O 口。通过寄存器使能上拉电阻和唤醒功能。 |
| REM/REMDRV | REM | TSR1 | — | CMOS | CMOS 载波输出引脚 |
| | REMDRV | TSR1 | — | NMOS | NMOS 载波输出引脚 |
| VDD | VDD | — | PWR | — | 正电源供电 |
| VSS | VSS | — | PWR | — | 负电源供电，接地 |
| VSS1 | VSS1 | — | PWR | — | REMDRV 负电源供电 |

注：I/T：输入类型；
OPT：引脚选择相关寄存器；
ST：施密特触发输入；
NMOS：NMOS 输出
O/T：输出类型；
PWR：电源；
CMOS：CMOS 输出；

## 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
$I_{OL}$ 总电流..... 80mA
$I_{OH}$ 总电流 ..... -80mA
总功耗 ..... 500mW

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

## 直流电气特性

以下表格中参数测量结果可能受多个因素影响，如振荡器类型、工作电压、工作频率、引脚负载状况、温度和程序指令等。

### 工作电压特性

Ta=25°C

| 符号 | 参数 | 测试条件 | 最小 | 典型 | 最大 | 单位 |
|---|---|---|---|---|---|---|
| $V_{DD}$ | 工作电压 – HIRC | $f_{SYS}$=4MHz | 1.8 | — | 5.5 | V |
| | 工作电压 – LIRC | $f_{SYS}$=32kHz | 1.8 | — | 5.5 | V |

### 工作电流特性

Ta=25°C

| 符号 | 工作模式 | 测试条件 | | 最小 | 典型 | 最大 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $I_{DD}$ | 快速模式 | 3V | $f_{SYS}$=4MHz | — | 1.0 | 2.0 | mA |
| | | 5V | | — | 2.0 | 3.0 | |
| | 低速模式 – LIRC | 3V | $f_{SYS}$=32kHz | — | 10 | 20 | μA |
| | | 5V | | — | 30 | 50 | |

注：当使用该表格电气特性数据时，以下几点需注意：
1. 任何数字输入都设置为非浮空的状态。
2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有的工作电流值都通过一个连续的 NOP 指令循环程序测得。

### 待机电流特性

Ta=25°C

| 符号 | 待机模式 | 测试条件 | | 典型 | 最大 | 最大 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | 85°C | |
| $I_{STB}$ | 休眠模式 | 3V | WDT off | 0.2 | 0.8 | 1 | μA |
| | | 5V | | 0.5 | 1 | 1.2 | |
| | | 3V | WDT on | 3.0 | 5.0 | 6.0 | μA |
| | | 5V | | 5.0 | 10 | 12 | |
| | 空闲模式 0 – LIRC | 3V | $f_{SUB}$ on | 3.0 | 5.0 | 6.0 | μA |
| | | 5V | | 5.0 | 10 | 12 | |
| | 空闲模式 1 – LIRC | 3V | $f_{SUB}$ on, $f_{SYS}$=4MHz | 0.8 | 1.6 | 1.9 | mA |
| | | 5V | | 1.0 | 2.0 | 2.4 | |

注：当使用该表格电气特性数据时，以下几点需注意：

1. 任何数字输入都设置为非浮空的状态。
2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有待机电流数值都是在 HALT 指令执行后测得，因此 HALT 后停止执行所有指令。

## 交流电气特性

以下表格中参数测量结果可能受多个因素影响，如振荡器类型、工作电压、工作频率和温度等等。

### 内部振荡器电气特性

Ta=25°C，除非另有说明

| 符号 | 参数 | 测试条件 | | 最小 | 典型 | 最大 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 温度 | | | | |
| $f_{HIRC}$ | HIRC 频率 | 3V | 25°C | -0.4% | 4 | +0.4% | MHz |
| | | 2.2V~3.6V | -10°C~50°C | -0.8% | 4 | +0.8% | |
| | | | -40°C~85°C | -1.5% | 4 | +1.5% | |
| | | 1.8V~3.6V | -10°C~50°C | -0.8% | 4 | +0.8% | |
| | | | -40°C~85°C | -1.5% | 4 | +1.5% | |
| | | 2.2V~5.5V | -10°C~50°C | -0.8% | 4 | +0.8% | |
| | | | -40°C~85°C | -1.5% | 4 | +1.5% | |
| | | 1.8V~5.5V | -40°C~85°C | -1.5% | 4 | +1.5% | |
| $f_{LIRC}$ | LIRC 频率 | 5V | 25°C | 25.6 | 32 | 38.4 | kHz |
| | | 1.8V~5.5V | 25°C | 12.8 | 32 | 41.6 | |
| | | | -40°C~85°C | 8 | 32 | 60 | |
| $t_{START}$ | LIRC 启动时间 | — | — | — | — | 100 | μs |

### 系统上电时间特性

Ta=25°C

| 符号 | 参数 | 测试条件 | | 最小 | 典型 | 最大 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $t_{SST}$ | 系统启动时间 ( 从系统时钟 off 状态下唤醒 ) | — | $f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$ | — | 16 | — | $t_{HIRC}$ |
| | | — | $f_{SYS}=f_{SUB}=f_{LIRC}$ | — | 2 | — | $t_{LIRC}$ |
| | 系统启动时间 ( 从系统时钟 on 状态下唤醒 ) | — | $f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$ | — | 2 | — | $t_H$ |
| | | — | $f_{SYS}=f_{SUB}=f_{LIRC}$ | — | 2 | — | $t_{SUB}$ |
| | 系统速度切换时间 ( 快速模式 → 低速模式或 低速模式 → 快速模式 ) | — | $f_{HIRC}$ off → on | — | 16 | — | $t_{HIRC}$ |
| $t_{RSTD}$ | 系统复位延迟时间 ( 上电复位或 LVR 硬件复位 ) | — | $RR_{POR}$=5V/ms | 8 | 16 | 60 | ms |
| | 系统复位延迟时间 ( WDTC/LVRC 软件复位 ) | — | — | | | | |
| | 系统复位延迟时间 ( WDT 溢出复位 ) | — | — | 8 | 16 | 60 | ms |
| $t_{SRESET}$ | 软件复位最小脉宽 | — | — | 45 | 90 | 375 | μs |

注：1. 系统启动时间里提到的 $f_{SYS}$ on/off 状态取决于工作模式类型以及所选的系统时钟振荡器。更多相关细节请参考系统工作模式章节。

2. $t_{LIRC}=1/f_{LIRC}$

3. LIRC 被选择作为系统时钟源且在休眠模式下 LIRC 关闭，则上面表格中对应 $t_{SST}$ 数值还需加上 LIRC 频率表格里提供的 LIRC 启动时间 $t_{START}$。

4. 系统速度切换时间实际上是指新使能的振荡器的启动时间。

## 输入 / 输出口电气特性

Ta=25°C

| 符号 | 参数 | 测试条件 | | 最小 | 典型 | 最大 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $V_{IL}$ | I/O 口或输入引脚低电平输入电压 | 5V | — | 0 | — | 1.5 | V |
| | | — | — | 0 | — | $0.2V_{DD}$ | |
| $V_{IH}$ | I/O 口或输入引脚高电平输入电压 | 5V | — | 3.5 | — | 5.0 | V |
| | | — | — | $0.8V_{DD}$ | — | $V_{DD}$ | |
| $I_{OL}$ | I/O 口灌电流 | 3V | $V_{OL}=0.1V_{DD}$ | 5 | 10 | — | mA |
| | | 5V | | 10 | 20 | — | |
| $I_{OH}$ | I/O 口源电流 | 3V | $V_{OH}=0.9V_{DD}$ | -1.25 | -2.5 | — | mA |
| | | 5V | | -2.5 | -5 | — | |
| $R_{PH}$ | I/O 口上拉电阻 (注) | 3V | LVPU=0 | 20 | 60 | 100 | kΩ |
| | | 5V | | 10 | 30 | 50 | |
| | | 3V | LVPU=1 | 6.67 | 15 | 23 | |
| | | 5V | | 3.5 | 7.5 | 12 | |
| $I_{LEAK}$ | 输入漏电流 | 5V | $V_{IN}=V_{DD}$ 或 $V_{IN}=V_{SS}$ | — | — | ±1 | μA |

注：$R_{PH}$ 内部上拉电阻值的计算方法是：将其接地并使能输入引脚的上拉电阻选项，然后在特定电源电压下测量输入灌电流，在此基础上测量上拉电阻上的分压从而得到此上拉电阻值。

## REM/REMDRV 引脚电气特性

Ta=25°C

| 符号 | 参数 | 测试条件 | | 最小 | 典型 | 最大 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $I_{DRVOL}$ | REMDRV 引脚灌电流 | 3V | $V_{OL}$=0.5V | — | 500 | — | mA |
| $I_{OL}$ | REM 引脚灌电流 | 3V | $V_{OL}$=0.1$V_{DD}$ | 5 | 10 | — | mA |
| | | 5V | | 10 | 20 | — | |
| $I_{OH}$ | REM 引脚源电流 | 3V | $V_{OL}$=0.9$V_{DD}$ | -1.25 | -2.5 | — | mA |
| | | 5V | | -2.5 | -5 | — | |
| $t_{READYB}$ | REMDRV 输出功能稳定时间 ( 通过轮询 READYB 位为零 ) | — | 从空闲或休眠模式唤醒或将 REMDRV 位从 1 改为 0 | — | 250 | — | μs |

## LVR 电气特性

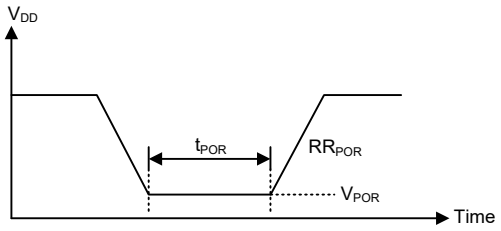
Ta=25°C

| 符号 | 参数 | 测试条件 | | 最小 | 典型 | 最大 | 单位 |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | 条件 | | | | |
| $V_{LVR}$ | 低电压复位电压 | — | LVR 使能，电压选择 1.7V | -5% | 1.7 | +5% | V |
| $I_{LVR}$ | 工作电流 | 3V | LVR 使能，$V_{LVR}$=1.7V | — | — | 15 | μA |
| | | 5V | | — | 15 | 25 | |
| $t_{LVR}$ | 产生 LVR 复位的低电压最短保持时间 | — | — | 250 | 500 | 1000 | μs |

## 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 |
| $RR_{POR}$ | 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 |

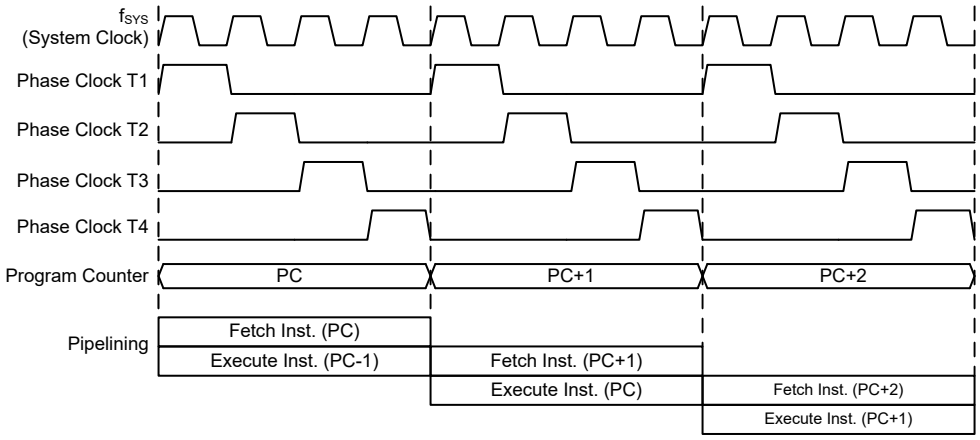


# 系统结构

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，该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需要多一个指令周期外，大部分指令能在一个指令周期内完成。8-bit 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，确保了在提供具有较大可靠度和灵活性的 I/O 控制系统时，仅需要少数的外部器件。这些使得此单片机适用于低成本和批量生产的控制应用。

## 时序和流水线结构

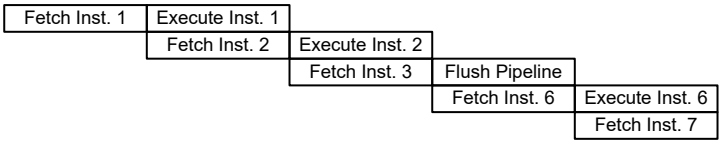
主系统时钟由 HIRC 或 LIRC 振荡器提供，它被细分为 T1~T4 四个内部产生的非重叠时序。在 T1 时间，程序计数器自动加一并抓取一条新的指令。剩下的时间 T2~T4 完成译码和执行功能，因此，一个 T1~T4 时钟周期构成一个指令周期。虽然指令的抓取和执行发生在连续的指令周期，但单片机流水线结构会保证指令在一个指令周期内被有效执行。除非程序计数器的内容被改变，如子程序的调用或跳转，在这种情况下指令将需要多一个指令周期的时间去执行。




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 |
| PC9~PC8 | PCL7~PCL0 |

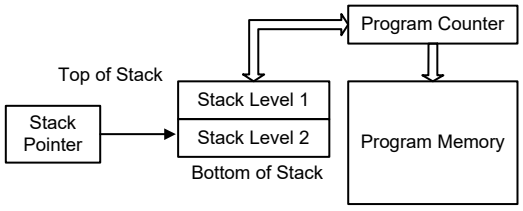
程序计数器

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。注意，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。

## 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 2 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。若堆栈溢出，则首个存入堆栈的程序计数器数据将会丢失。




## 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

与立即数相关的指令仅对 4-bit 低半字节有效。因此，在对一个 8-bit 进行立即数操作时，应先使高 4 位为“0H”并使用 SWAP 指令，处理好高半字节，接着直接赋值给低 4 位。

要实现“MOV A, 03CH”操作，应执行下面三条指令：

```
MOV A,03H       ; High nibble immediate data processed first
SWAP ACC
MOV A,0CH       ; Low nibble immediate data processed later
```

要实现“RET A, 0C3H”操作，应执行下面三条指令：

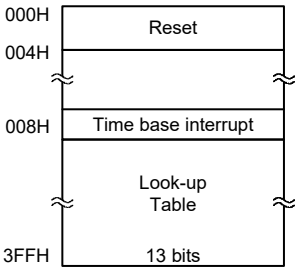
```
MOV A,0CH       ; High nibble immediate data processed first
SWAP ACC
RET A,03H       ; Low nibble immediate data processed later
```

## Flash 程序存储器

程序存储器用来存放用户代码即储存程序。程序存储器为 Flash 类型意味着可以多次重复编程，方便用户使用同一芯片进行程序的修改。使用适当的单片机编程工具，此单片机提供用户灵活便利的调试方法和项目开发规划及更新。

### 结构

程序存储器的容量为 1K×13 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

程序存储器内部某些地址保留用做诸如复位和时基中断入口等特殊用途。地址 000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。008H 为时基中断入口。

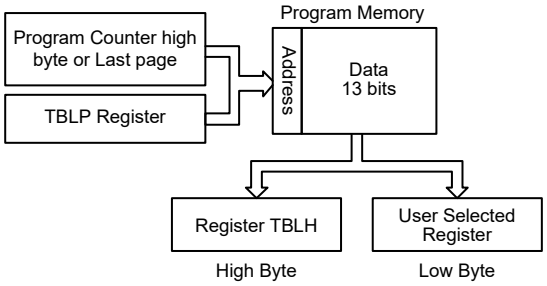
### 查表

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，其方式是将表格的地址放在表格指针寄存器

TBLP 中。此寄存器定义表格低 8 位地址。

在设定完表格指针后，表格数据可以使用如“TABRD [m]”或“TABRDL [m]”等指令分别从程序存储器当前页或最后一页查表读取。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 专用寄存器。

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中。ORG 指令的值“300H”指向的地址是 1K word 程序存储器中最后一页的起始地址。表格指针低字节寄存器的初始值设为 06H，这可保证从数据表格读取的第一笔数据位于程序存储器地址“306H”，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRD [m]”指令被使用，则表格指针指向当前页的第一个地址。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。

TBLH 寄存器为只读寄存器，不能被重复储存，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。

### 表格读取程序范例

```
ds .section 'data'
tempreg1 db?        ; temporary register #1
tempreg2 db?        ; temporary register #2
code0 .section 'code'
mov a,00h
swap acc
mov a,06h           ; initialise low 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
                    ; data at program memory address "306H" transferred to
                    ; tempreg1
dec tblp            ; reduce value of table pointer by one
tabrdl tempreg2     ; transfers value in table referenced by table pointer
                    ; data at program memory address "305H" transferred to
                    ; tempreg2
```

```
                    ; in this example the data "1AH" is transferred to
                    ; tempreg1 and data "0FH" to tempreg2
:
:
org 300h            ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh

:
```

## 在线烧录 – ICP

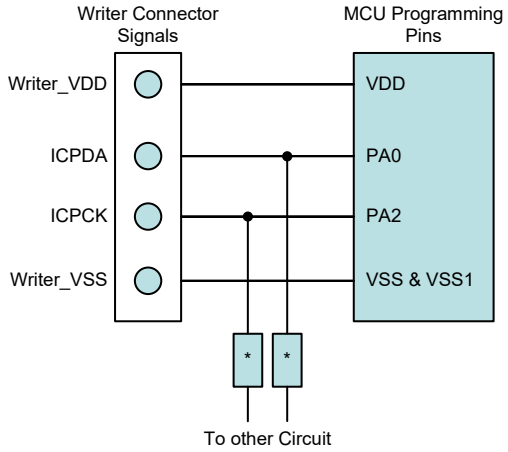
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

另外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧写，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

| 烧录器引脚 | MCU 在线烧录引脚 | 引脚描述 |
|---|---|---|
| ICPDA | PA0 | 串行数据输入 / 输出 |
| ICPCK | PA2 | 串行时钟 |
| VDD | VDD | 电源 |
| VSS | VSS&VSS1 | 地 |

单片机内部程序存储器可以通过 4 线的接口在线进行烧录。其中一条线用于数据串行下载或上传，一条用于串行时钟，剩余两条用于提供电源。芯片在线烧写的详细使用说明超出此文档的描述范围，将由专门的参考文献提供。

在烧录过程中，烧录器会控制 ICPDA 和 ICPCK 脚进行数据和时钟烧录，用户必须确保这两个引脚没有连接至其它输出脚。




注：* 可能为电阻或电容。若为电阻则其值必须大于 1kΩ，若为电容则其必须小于 1nF。

## 片上调试 – OCDS

EV 芯片 HT68V2420 用于单片机 HT68F2420 仿真。此 EV 芯片提供片上调试功能 (OCDS – On-Chip Debug Support) 用于开发过程中的单片机调试。除了片上调试功能方面，EV 芯片和实际单片机在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 HT-IDE 开发工具，从而实现 EV 芯片对实际单片机的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，实际单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚

共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考相应文件“Holtek e-Link for 8-bit MCU OCDS 使用手册”。

| e-Link 引脚 | EV 芯片引脚 | 引脚描述 |
|---|---|---|
| OCDSDA | OCDSDA | 片上调试串行数据 / 地址输入 / 输出 |
| OCDSCK | OCDSCK | 片上调试时钟输入 |
| VDD | VDD | 电源 |
| VSS | VSS & VSS1 | 地 |

## 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。

数据存储器分为两个部分，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。

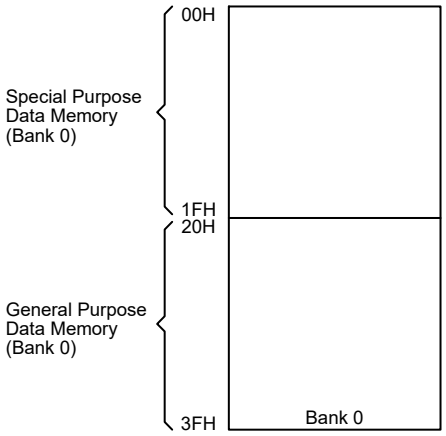
### 结构

数据存储器具有一个 Bank，可在 8-bit 的存储器中实现。数据存储器 Bank 分为两类，即特殊功能数据存储器和通用数据存储器。

该单片机特殊功能数据存储器的地址范围为 00H 到 1FH，而通用数据存储器地址范围为 20H 到 3FH。

| 特殊功能数据存储器 | | 通用数据存储器 | |
|---|---|---|---|
| 所在 Bank | Bank：地址 | 容量 | Bank：地址 |
| 0 | 0: 00H~1FH | 32×8 | 0: 20H~3FH |

数据存储器概要




数据存储器结构

### 通用数据存储器

所有的单片机程序需要一个读 / 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。使用者可对这个数据存储区进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，较大地方便了用户在数据存储器内进行位操作。

### 特殊功能数据存储器

这个区域的数据存储器是存放特殊寄存器的，这些寄存器与单片机的正确操作密切相关，大多数的寄存器可进行读取和写入，只有一些是被写保护而只能读取的，相关细节的介绍请参看有关特殊功能寄存器的部分。要注意的是，任何读取指令对存储器中未定义的地址进行读取将返回“00H”。

| Bank 0 | | Bank 0 | |
|---|---|---|---|
| 00H | IAR0 | 10H | PA |
| 01H | MP0 | 11H | PAC |
| 02H | WDTC | 12H | PAPU |
| 03H | SCC | 13H | PAWU |
| 04H | HIRCC | 14H | PB |
| 05H | ACC | 15H | PBC |
| 06H | PCL | 16H | PBPU |
| 07H | TBLP | 17H | PBWU |
| 08H | TBLH | 18H | |
| 09H | | 19H | LVRC |
| 0AH | STATUS | 1AH | TSR0 |
| 0BH | INTC | 1BH | TSR1 |
| 0CH | LVPUC | 1CH | CARL0 |
| 0DH | PSCR | 1DH | CARL1 |
| 0EH | TBC | 1EH | CARH0 |
| 0FH | RSTFC | 1FH | CARH1 |

: Unused, read as 00H

特殊功能数据存储器

## 特殊功能寄存器

大部分特殊功能寄存器的细节将在相关功能章节描述，但有几个寄存器需在此章节单独描述。

### 间接寻址寄存器 – IAR0

间接寻址寄存器 IAR0 的地址虽位于数据存储区，但其并没有实际的物理地址。间接寻址的方法使用这个间接寻址寄存器和存储器指针做数据操作，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 上的任何动作，将对存储器指针 MP0 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，IAR0 和 MP0 可以访问 Bank0。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。

### 存储器指针 – MP0

该单片机提供了存储器指针 MP0。由于该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。当对相关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存储器指针 MP0 所指定的地址，此时间接寻址寄存器 IAR0 用于访问 Bank0 中的数据。

下面的例子显示了如何清除一个具有 4 个 RAM 地址的模块。它们已事先定义成 adres1 到 adres4。

间接寻址程序范例

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 code
org 00h
start:
mov a,00h
swap acc
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:
```

需注意，范例中并没有确定 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-bit 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。

## 表格寄存器 – TBLP, TBLH

这两个特殊功能寄存器对存储在程序存储器中的表格进行操作。TBLP 为表格指针，指向当前页或最后一页表格数据存储的地址。它的值必须在任何表格读取指令执行前加以设定，由于它们的值可以被如“INC”或“DEC”的指令所改变，这就提供了一种简单的方法对表格数据进行读取。表格读取数据指令执行之后，表格数据高字节存储在 TBLH 中。其中要注意的是，表格数据低字节会被传送到使用者指定的地址。

## 状态寄存器 – STATUS

该 8-bit 的状态寄存器由零标志位 (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) 和看门狗定时器溢出标志位 (TO) 组成。这些算术 / 逻辑操作和系统运行标志位是用来记录单片机的运行状态。

除了 TO 和 PDF 标志外，状态寄存器中的位像其它大部分寄存器一样可以被改变。任何数据写入到状态寄存器将不会改变 TO 或 PDF 标志位。另外，执行不同的指令后，与状态寄存器有关的运算可能会得到不同的结果。TO 标志位只会受系统上电、看门狗溢出或执行“CLR WDT”或“HALT”指令影响。PDF 标志位只会受执行“HALT”或“CLR WDT”指令或系统上电影响。

Z、OV、AC 和 C 标志位通常反映最近运算的状态。

- C：当加法运算的结果产生进位，或减法运算的结果没有产生借位时，则 C 被置位，否则 C 被清零。
- AC：当低半字节加法运算的结果产生进位，或低半字节减法运算的结果没有产生借位时，AC 被置位，否则 AC 被清零。
- Z：当算术或逻辑运算结果是零时，Z 被置位，否则 Z 被清零。
- OV：当运算结果高两位的进位状态异或结果为 1 时，OV 被置位，否则 OV 被清零。
- PDF：系统上电或执行“CLR WDT”指令会清零 PDF，而执行“HALT”指令则会置位 PDF。
- TO：系统上电或执行“CLR WDT”或“HALT”指令会清零 TO，而当 WDT 溢出则会置位 TO。

另外，当进入一个中断程序或执行子程序调用时，状态寄存器不会自动压入到堆栈保存。假如状态寄存器的内容是重要的且子程序可能改变状态寄存器的话，则需谨慎的去做正确的储存。

- **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 | x | x | x | x |

“x”：未知

Bit 7~6　未定义，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
进位标志位 C 也受循环移位指令的影响。

# 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较佳的优化。振荡器选择是通过相关的控制寄存器来完成的。

## 振荡器概述

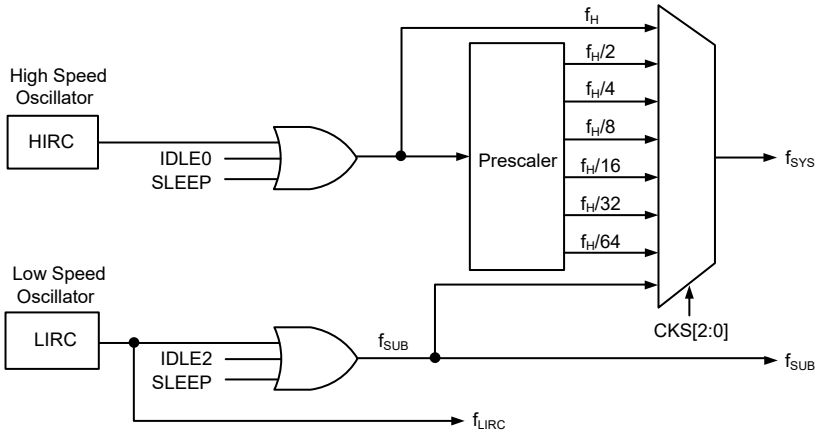
振荡器除了作为系统时钟源，还作为看门狗定时器和时基中断的时钟源。该单片机集成的两个内部振荡器不需要任何外围器件，它们提供的高速和低速系统振荡器具有较宽的频率范围。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换不同系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

| 类型 | 名称 | 频率 |
|---|---|---|
| 内部高速 RC | HIRC | 4MHz |
| 内部低速 RC | LIRC | 32kHz |

振荡器类型

## 系统时钟配置

该单片机有两个振荡器，包括一个高速振荡器和一个低速振荡器。高速振荡器 HIRC 提供 4MHz 频率。低速振荡器为内部 32kHz 低速振荡器 LIRC。使用高速或低速时钟作为系统时钟的选择是通过设置 SCC 寄存器中的 CKS2~CKS0 位决定的，系统时钟可动态选择。




系统时钟配置

## 内部高速 RC 振荡器 – HIRC

内部 RC 振荡器是一个完全内部集成的系统振荡器，不需其它外部器件。内部 RC 振荡器可提供 4MHz 固定时钟。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较大程度地降低。

### 内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器是一个完全集成的 RC 振荡器，它在 5V 电压下运行的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响较大程度地降低。

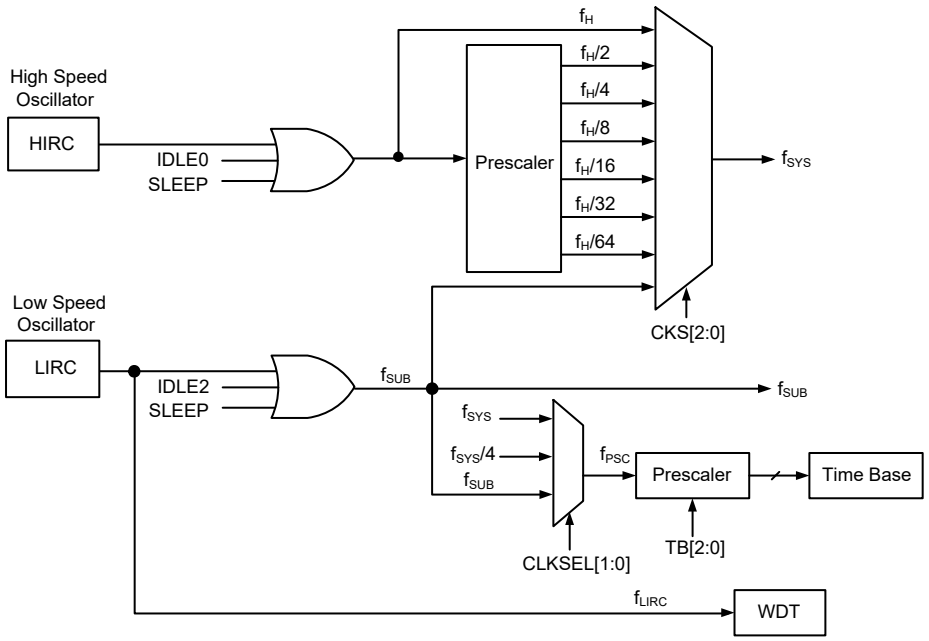
## 工作模式和系统时钟

现今的应用要求单片机具有较高的性能及尽可能低的功耗，这种矛盾的要求在便携式电池供电的应用领域尤为明显。高性能所需要的高速时钟将增加功耗，反之亦然。此单片机提供高、低速两种时钟源，它们之间可以动态切换，用户可通过优化单片机操作来获得较佳性能 / 功耗比。

### 系统时钟

单片机为 CPU 和外围功能操作提供了多种不同的时钟源。用户使用寄存器编程可获取多种时钟，进而使系统时钟获取较大的应用性能。

主系统时钟可来自高频时钟源 $f_H$ 或低频时钟源 $f_{SUB}$，通过 SCC 寄存器中的 CKS2~CKS0 位进行选择。高频时钟来自 HIRC 振荡器，低频时钟来自 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。




单片机时钟配置

注：当系统时钟源 $f_{SYS}$ 由 $f_H$ 切换到 $f_{SUB}$ 时，可以通过设置相应的高速振荡器使能控制位，选择停止以节省耗电，或者选择继续振荡，为外围电路提供 $f_H$~$f_H/64$ 频率的时钟源。

### 系统工作模式

单片机有 6 种不同的工作模式，每种有它自身的特性，根据应用中不同的性能和功耗要求可选择不同的工作模式。单片机正常工作有两种模式：快速模式和低速模式。剩余的 4 种工作模式：休眠模式、空闲模式 0、空闲模式 1 和空闲模式 2 用于单片机 CPU 关闭时以节省耗电。

| 工作模式 | CPU | 寄存器设置 | | | $f_{SYS}$ | $f_H$ | $f_{SUB}$ | $f_{LIRC}$ |
|---|---|---|---|---|---|---|---|---|
| | | FHIDEN | FSIDEN | CKS2~CKS0 | | | | |
| 快速模式 | On | x | x | 000~110 | On | On | On | On |
| 低速模式 | On | x | x | 111 | On | On/Off (1) | On | On |
| 空闲模式 0 | Off | 0 | 1 | 000~110 | Off | Off | On | On |
| | | | | 111 | On | | | |
| 空闲模式 1 | Off | 1 | 1 | xxx | On | On | On | On |
| 空闲模式 2 | Off | 1 | 0 | 000~110 | On | On | Off | On |
| | | | | 111 | Off | | | |
| 休眠模式 | Off | 0 | 0 | xxx | Off | Off | Off | On/Off (2) |

“x”：无关

注：1. 在低速模式中，$f_H$ 开启或关闭由相应的振荡器使能位控制。
2. 在休眠模式中，若 WDT 功能使能，$f_{LIRC}$ 时钟开启，若 WDT 功能除能，$f_{LIRC}$ 时钟关闭。

### 快速模式

这是主要的工作模式之一，单片机的所有功能均可在此模式中实现且系统时钟由一个高速振荡器提供。该模式下单片机正常工作的时钟源来自 HIRC 振荡器。高速振荡器频率可被分为 1~64 的不等比率，实际的比率由 SCC 寄存器中的 CKS2~CKS0 位选择。单片机使用高速振荡器分频作为系统时钟可减少工作电流。

### 低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。该低速时钟源可来自 $f_{SUB}$，而 $f_{SUB}$ 来自于 LIRC 振荡器。

### 休眠模式

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为低时，系统进入休眠模式。在休眠模式中，CPU 停止运行，$f_{SUB}$ 停止为外围功能提供时钟。若看门狗定时器功能使能，$f_{LIRC}$ 时钟将继续运行，若此功能除能，$f_{LIRC}$ 时钟将停止运行。

### 空闲模式 0

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 位为低、FSIDEN 位为高时，系统进入空闲模式 0。在空闲模式 0 中，CPU 停止，但低速振荡器会开启以驱动一些外围功能。

### 空闲模式 1

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但高速和低速振荡器都会开启以确保一些外围功能继续工作。

### 空闲模式 2

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 位为高、FSIDEN 位为低时，系统进入空闲模式 2。在空闲模式 2 中，CPU 停止，但高速振荡器会开启以确保一些外围功能继续工作。

## 控制寄存器

寄存器 SCC 和 HIRCC 用于控制系统时钟和相应的振荡器配置。

| 寄存器名称 | 位 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SCC | CKS2 | CKS1 | CKS0 | — | — | — | FHIDEN | FSIDEN |
| HIRCC | — | — | — | — | — | — | HIRCF | HIRCEN |

系统工作模式控制寄存器列表

- **SCC 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CKS2 | CKS1 | CKS0 | — | — | — | FHIDEN | FSIDEN |
| R/W | R/W | R/W | R/W | — | — | — | R/W | R/W |
| POR | 0 | 0 | 0 | — | — | — | 0 | 0 |

Bit 7~5 **CKS2~CKS0**：系统时钟选择位
000：$f_H$
001：$f_H/2$
010：$f_H/4$
011：$f_H/8$
100：$f_H/16$
101：$f_H/32$
110：$f_H/64$
111：$f_{SUB}$
这三位用于选择系统时钟源。除了 $f_H$ 或 $f_{SUB}$ 提供的系统时钟源外，也可使用高频振荡器的分频作为系统时钟。

Bit 4~2 未定义，读为“0”

Bit 1 **FHIDEN**：CPU 关闭时高频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后高速振荡器是运行还是停止。

Bit 0 **FSIDEN**：CPU 关闭时低频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后低速振荡器是运行还是停止。由于 LIRC 振荡器需提供时钟 $f_{LIRC}$ 给 WDT 功能，LIRC 振荡器的 ON/OFF 同时也受 WDT 功能控制。若 WDT 功能使能，即使该位为 0，LIRC 还会继续运行。

- **HIRCC 寄存器**

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

Bit 7~2 未定义，读为“0”

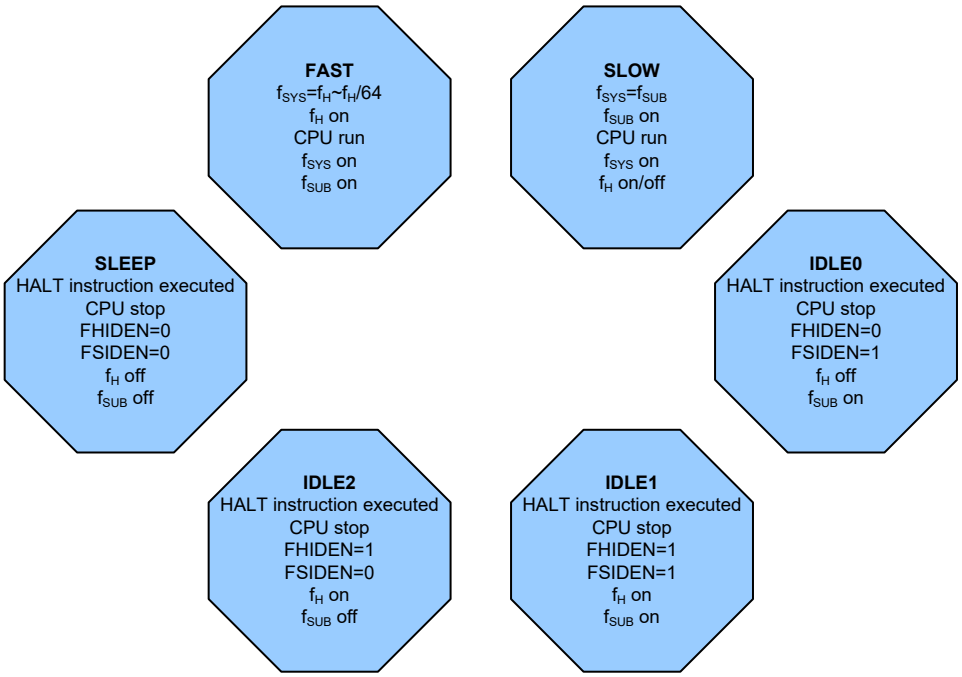
Bit 1 **HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 未稳定
1：HIRC 稳定
此位用于指示 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器后，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

Bit 0 **HIRCEN**：HIRC 振荡器使能控制位
0：除能
1：使能

## 工作模式切换

该单片机可在各个工作模式间自由切换，使得用户可根据所需选择较佳的性能 / 功耗比。用此方式，对单片机工作的性能要求不高的情况下，可使用较低频时钟以减少工作电流，在便携式应用上延长电池的使用寿命。

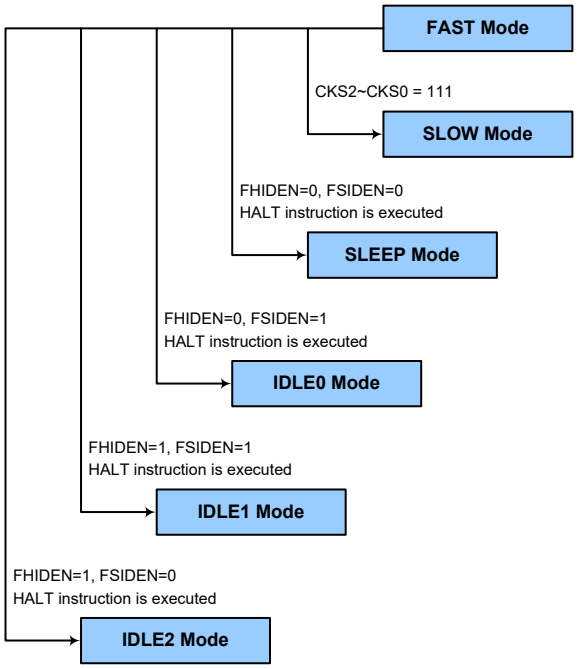
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。




### 快速模式切换到低速模式

系统运行在快速模式时使用高速系统振荡器，因此较为耗电。可通过设置 SCC 寄存器中的 CKS2~CKS0 位为“111”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。

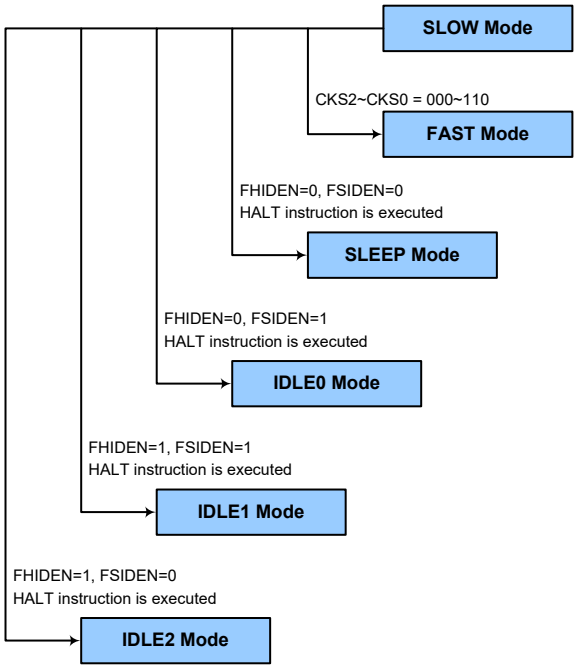
低速模式的时钟源来自 LIRC 振荡器，此振荡器需在所有模式切换动作发生前稳定下来。




### 低速模式切换到快速模式

在低速模式时系统时钟来自 $f_{SUB}$。切换回快速模式时，需设置 CKS2~CKS0 位为“000”~“110”使系统时钟从 $f_{SUB}$ 切换到 $f_H$~$f_H$/64。

然而，如果在低速模式下 $f_H$ 因未使用而关闭，那么从低速模式切换到快速模式时，它需要一定的时间来重新起振和稳定，可通过检测 HIRCC 寄存器中的 HIRCF 位进行判断，所需的高速系统振荡器稳定时间在系统上电时间电气特性中有说明。



### 进入休眠模式

进入休眠模式的方法仅有一种即应用程序中执行“HALT”指令前需设置 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为“0”。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟停止运行，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 若 WDT 功能使能，WDT 将被清零并重新开始计数。若 WDT 功能除能，WDT 将被清零并停止。

### 进入空闲模式 0

进入空闲模式 0 的方法仅有一种即应用程序中执行“HALT”指令前需设置 SCC 寄存器中的 FHIDEN 位为“0”且 FSIDEN 位为“1”。在上述条件下执行该指令后，将发生的情况如下：

- $f_H$ 时钟停止运行，应用程序停止在“HALT”指令处，但 $f_{SUB}$ 时钟将继续运行。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 若 WDT 功能使能，WDT 将被清零并重新开始计数。若 WDT 功能除能，WDT 将被清零并停止。

### 进入空闲模式 1

进入空闲模式 1 的方法仅有一种即应用程序中执行“HALT”指令前需设置 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为“1”。在上述条件下执行该指令后，将发生的情况如下：

- $f_H$ 和 $f_{SUB}$ 时钟开启，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 若 WDT 功能使能，WDT 将被清零并重新开始计数。若 WDT 功能除能，WDT 将被清零并停止。

### 进入空闲模式 2

进入空闲模式 2 的方法仅有一种即应用程序中执行“HALT”指令前需设置 SCC 寄存器中的 FHIDEN 位为“1”且 FSIDEN 位为“0”。在上述条件下执行该指令后，将发生的情况如下：

- $f_H$ 时钟开启，$f_{SUB}$ 时钟关闭，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 若 WDT 功能使能，WDT 将被清零并重新开始计数。若 WDT 功能除能，WDT 将被清零并停止。

## 待机电流注意事项

由于单片机进入休眠或空闲模式的主要原因是将单片机的电流降低到尽可能低，在空闲模式 0 和休眠模式下可能到只有几个微安的级别，所以如果要将电路的电流进一步降低，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。

在空闲模式 1 和空闲模式 2 中，高速振荡器开启。若系统时钟来自高速系统振荡器，额外的待机电流也可能会有几百微安。

## 唤醒

单片机进入休眠模式或空闲模式后，系统时钟将停止以降低功耗。然而单片机再次唤醒，原来的系统时钟重新起振、稳定且恢复正常工作需要一定的时间。

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- I/O 口下降沿
- 系统中断
- WDT 溢出

单片机执行 HALT 指令，系统进入空闲或休眠模式，PDF 将被置位；系统上电或执行清除看门狗的指令，PDF 将被清零。看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。

PA 和 PB 口中的每个引脚都可以通过 PAWU 和 PBWU 寄存器中的相应位使能下降沿唤醒功能。I/O 端口唤醒后，程序将在“HALT”指令后继续执行。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：相关中断除能或是中断使能且堆栈已满，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。 如果在进入休眠或空闲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。

# 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

WDT 定时器时钟源来自于内部时钟 $f_{LIRC}$，由内部振荡器 LIRC 提供。内部振荡器 LIRC 的频率大约为 32kHz。需要注意的是，具体的内部时钟周期随 $V_{DD}$、温度和制程的不同而变化。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 WDTC 寄存器中的 WS2~WS0 位来决定。

## 看门狗定时器控制寄存器

WDTC 寄存器用于控制 WDT 功能的使能 / 除能控制，溢出周期选择及复位单片机操作。

- **WDTC 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | WE4 | WE3 | WE2 | WE1 | WE0 | WS2 | WS1 | WS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |

Bit 7~3 **WE4~WE0**：WDT 软件控制位
10101：除能
01010：使能
其它值：MCU 复位
如果由于不利的环境因素使这些位变为其它值，单片机将复位。复位动作发生在一段延迟时间 $t_{SRESET}$ 后，且 RSTFC 寄存器的 WRF 位将置为“1”。

Bit 2~0 **WS2~WS0**：WDT 溢出周期选择位
当通过“CLR WDT”指令或“HALT”指令清除看门狗定时器的内容，WDT 溢出时间周期为：
000：$(2^8\text{-}2^0\sim2^8)/f_{LIRC}$
001：$(2^9\text{-}2^1\sim2^9)/f_{LIRC}$
010：$(2^{10}\text{-}2^2\sim2^{10})/f_{LIRC}$
011：$(2^{11}\text{-}2^3\sim2^{11})/f_{LIRC}$
100：$(2^{12}\text{-}2^4\sim2^{12})/f_{LIRC}$
101：$(2^{13}\text{-}2^5\sim2^{13})/f_{LIRC}$
110：$(2^{14}\text{-}2^6\sim2^{14})/f_{LIRC}$
111：$(2^{15}\text{-}2^7\sim2^{15})/f_{LIRC}$
当通过上电复位，WDT 溢出和 WDTC 软件复位清除看门狗定时器的内容，WDT 溢出周期为：
000：$2^8/f_{LIRC}$
001：$2^9/f_{LIRC}$
010：$2^{10}/f_{LIRC}$
011：$2^{11}/f_{LIRC}$
100：$2^{12}/f_{LIRC}$
101：$2^{13}/f_{LIRC}$
110：$2^{14}/f_{LIRC}$
111：$2^{15}/f_{LIRC}$
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。

- **RSTFC 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | LVRF | LRF | WRF |
| R/W | — | — | — | — | — | R/W | R/W | R/W |
| POR | — | — | — | — | — | x | 0 | 0 |

“x”：未知

Bit 7~3　未定义，读为“0”

Bit 2　**LVRF**：LVR 复位标志位
详见低电压复位章节

Bit 1　**LRF**：LVR 控制寄存器软件复位标志位
详见低电压复位章节

Bit 0　**WRF**：WDT 控制寄存器软件复位标志位
0：未发生
1：发生
当 WDT 控制寄存器软件复位发生时，此位被置为“1”。需注意此位只能通过应用程序清零。

## 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。无论什么原因，程序失常跳转到一个未知的地址或进入一个死循环，这些清除指令都不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。看门狗定时器控制寄存器 WDTC 中的 WE4~WE0 位可提供使能 / 除能控制以及控制看门狗定时器复位操作。当 WE4~WE0 设置为“10101B”时除能 WDT 功能，而当设置为“01010B”时使能 WDT 功能。如果 WE4~WE0 设置为除“01010B”和“10101B”以外的值时，单片机将在一段延迟时间 $t_{SRESET}$ 后复位。上电后这些位的值为“01010B”。

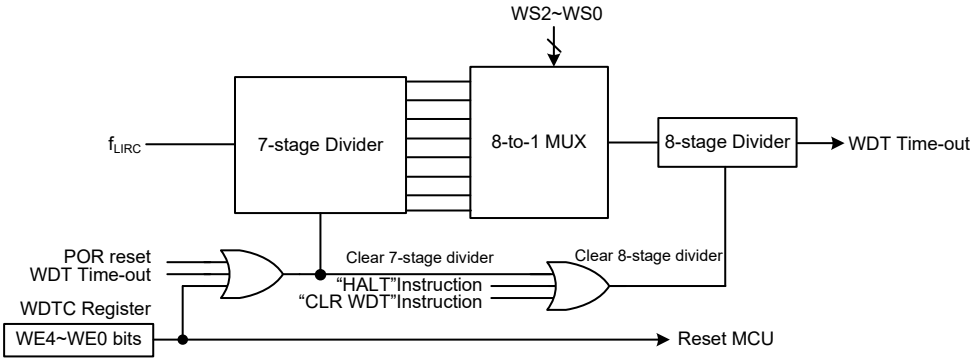
| WE4~WE0 | WDT 功能 |
|---|---|
| 10101B | 除能 |
| 01010B | 使能 |
| 其它值 | 复位 MCU |

看门狗定时器使能 / 除能控制

程序正常运行时，WDT 溢出将导致芯片复位，并置位状态标志位 TO。若系统处于休眠或空闲模式，当 WDT 发生溢出时，状态寄存器中的 TO 标志位会被置位，程序计数器 PC 和堆栈指针 SP 会被复位。有三种方法可以用来清除 WDT 的内容。第一种是 WDT 复位，即将 WE4~WE0 位设置成除了“01010B”和“10101B”外的任意值；第二种是通过看门狗定时器软件清除指令，而第三种是通过“HALT”指令。

该单片机只有一种软件指令清除 WDT 的方式，即清看门狗指令“CLR WDT”。因此只要执行“CLR WDT”便能清除 WDT。

当设置分频比为 $2^{15}$ 时，溢出周期最大。例如，时钟源为 32kHz LIRC 振荡器，分频比为 $2^{15}$ 时最大溢出周期约 1s，分频比为 $2^{8}$ 时最小溢出周期约 8ms。




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，使得单片机可以设置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置为预先设置的状态。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。

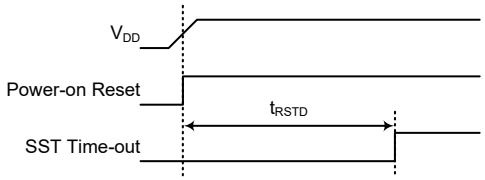
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于一定阈值的时候，系统会产生完全复位。此外还有看门狗定时器溢出也是单片机复位方式之一。不同方式的复位操作会对寄存器产生不同的影响。

## 复位功能

通过内部事件触发复位，单片机共有以下几种复位方式：

### 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置在预设条件。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置为输入状态。

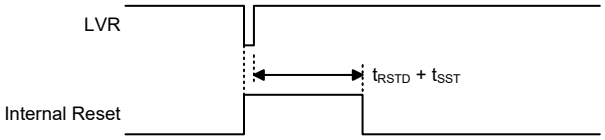



上电复位时序图

### 低电压复位 – LVR

单片机具有低电压复位电路，用来监测它的电源电压。电源电压低于某一预定值时，它将复位单片机。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~$V_{LVR}$ 之间，这时 LVR 将会自动复位单片机且 RSTFC 寄存器中的 LVRF 标志位置位。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 LVR 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。寄存器 LVRC

可提供 LVR 功能使能 / 除能控制以及控制单片机复位操作。当 LVRC 寄存器设置为 10100101B 时，LVR 功能将除能，当 LVRC 寄存器设置为 01011010B 时，在快速模式和低速模式下，低电压复位功能总是使能且 $V_{LVR}$ 固定电压值 1.7V。若 LVRC 的值由于不利的环境因素如噪声而改变为除“10100101B”和“01011010B”以外的值时，单片机将在一段延迟时间 $t_{SRESET}$ 后复位，此时 RSTFC 寄存器中的 LRF 位将被置为 1。上电后该寄存器的默认值为 01011010B。注意当单片机进入空闲或休眠模式，LVR 功能将自动关闭。




低电压复位时序图

- **LVRC 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | LVS7 | LVS6 | LVS5 | LVS4 | LVS3 | LVS2 | LVS1 | LVS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

Bit 7~0 **LVS7~LVS0**：LVR 功能使能控制
01011010：使能，选择 1.7V
10100101：除能
其它值：单片机复位 – 寄存器复位为 POR 值
若低电压情况发生即 $V_{DD}$ 电压降至 1.7V 且低电压状态保持时间大于 $t_{LVR}$，则单片机复位发生。此时复位后的寄存器内容保持不变。
除了 01011010 和 10100101 外其它寄存器值也能导致单片机复位。复位操作会在 $t_{SRESET}$ 时间后执行。注意的是此处单片机复位后，寄存器的值将恢复到上电复位值。

- **RSTFC 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | LVRF | LRF | WRF |
| R/W | — | — | — | — | — | R/W | R/W | R/W |
| POR | — | — | — | — | — | x | 0 | 0 |

“x”：未知

Bit 7~3 未定义，读为“0”

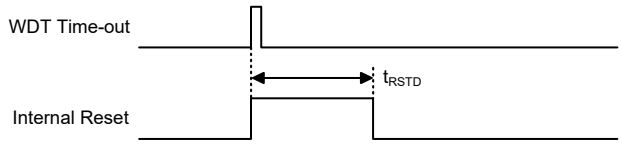
Bit 2 **LVRF**：LVR 复位标志位
0：未发生
1：发生
当特定的低电压复位条件发生时，此位被置为“1”，且只能通过应用程序清零。

Bit 1 **LRF**：LVR 控制寄存器软件复位标志位
0：未发生
1：发生
如果 LVRC 寄存器包含任何非定义的 LVR 电压值，此位被置为“1”，这类似于软件复位功能，且只能通过应用程序清零。

Bit 0 **WRF**：WDT 控制寄存器软件复位标志位
具体描述见看门狗定时器控制寄存器章节。

### 正常工作时的看门狗溢出复位

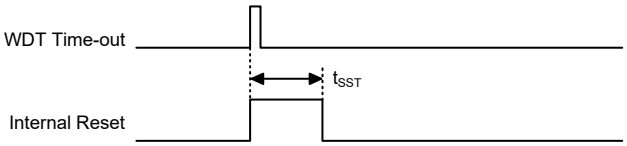
正常工作时看门狗溢出复位会将看门狗溢出标志位 TO 设为“1”。




正常工作时看门狗溢出时序图

### 休眠或空闲时看门狗溢出复位

休眠或空闲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。




休眠或空闲时的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，由休眠或空闲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 | 快速模式或低速模式时 LVR 复位 |
| 1 | u | 正常工作时的 WDT 溢出复位 |
| 1 | 1 | 空闲或休眠模式时的 WDT 溢出复位 |

注：“u”表示不变

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 |
| WDT，时基 | 复位后清零，且 WDT 重新开始计数 |
| 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状态是非常重要的。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 | WDT 溢出<br>( 正常工作 ) | WDT 溢出<br>( 空闲 / 休眠 ) |
|---|---|---|---|
| IAR0 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| MP0 | 11xx xxxx | 11xx xxxx | 11uu uuuu |
| WDTC | 0101 0111 | 0101 0111 | uuuu uuuu |
| SCC | 000- --00 | 000- --00 | uuu- --uu |
| HIRCC | ---- --01 | ---- --01 | ---- --uu |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TBLH | ---x xxxx | ---u uuuu | ---u uuuu |
| STATUS | --00 xxxx | --1u uuuu | --11 uuuu |
| INTC | --0- -0-0 | --0- -0-0 | --u- -u-u |
| LVPUC | ---- ---0 | ---- ---0 | ---- ---u |
| PSCR | ---- -000 | ---- -000 | ---- -uuu |
| TBC | 0--- -000 | 0--- -000 | u--- -uuu |
| RSTFC | ---- ---0 | ---- ---u | ---- ---u |
| PA | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAPU | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAWU | 0000 0000 | 0000 0000 | uuuu uuuu |
| PB | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBPU | 0000 0000 | 0000 0000 | uuuu uuuu |
| PBWU | 0000 0000 | 0000 0000 | uuuu uuuu |
| LVRC | 0101 1010 | 0101 1010 | uuuu uuuu |
| TSR0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TSR1 | 100- --00 | 100- --00 | uuu- --uu |
| CARL0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CARL1 | ---- --00 | ---- --00 | ---- --uu |
| CARH0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CARH1 | ---- --10 | ---- --10 | ---- --uu |

注：“u”表示未改变
“x”表示未知
“-”表示未定义

# 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。

此单片机提供了双向输入 / 输出口 PA~PB。这些端口在数据存储器有特定的地址，如特殊功能数据存储器表所示。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。

| 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 | PBPU2 | PBPU1 | PBPU0 |
| PBWU | PBWU7 | PBWU6 | PBWU5 | PBWU4 | PBWU3 | PBWU2 | PBWU1 | PBWU0 |

输入 / 输出逻辑功能寄存器列表

## 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 寄存器搭配 LVPUC 寄存器来设置。PAPU~PBPU 寄存器用于使能 / 除能上拉电阻，LVPUC 用于选择上拉电阻值。使用一个 PMOS 晶体管来实现上拉电阻功能。

需要注意的是，仅当 I/O 引脚设为逻辑输入或 NMOS 输出时，上拉功能才会受 PxPU 控制开启，其它输出状态下上拉功能不可用。

- **PxPU 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PxPU7 | PxPU6 | PxPU5 | PxPU4 | PxPU3 | PxPU2 | PxPU1 | PxPU0 |
| 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 |

**PxPUn**：I/O Px.n 端口上拉电阻控制位
0：除能
1：使能

PxPUn 位用于控制 Px.n 端口上拉电阻功能。这里的 x 可以是端口 A 和 B。但是，每个 I/O 端口实际有效位可能不同。

- **LVPUC 寄存器**

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

Bit 7~1　未定义，读为“0”

Bit 0　**LVPU**：低供电电压时的上拉电阻选择
0：所有引脚连接的上拉电阻为 60kΩ (typ.) @ 3V
1：所有引脚连接的上拉电阻为 15kΩ (typ.) @ 3V
此位主要用于低电压应用时，上拉电阻值的选择。需先通过 PxPU 寄存器使能相应引脚上拉电阻功能，然后设置 LVPU 位，则该引脚的上拉电阻值即可被设置。若上拉电阻功能除能，LVPU 位的设置无效。

## I/O 端口唤醒

当使用“HALT”指令迫使单片机进入休眠或空闲模式，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 PA 或 PB 口的其中一个引脚从高电平转为低电平。这个功能特别适合于通过外部开关来唤醒的应用。PA~PB 口的每个引脚可以通过设置 PAWU~PBWU 寄存器来单独选择是否具有唤醒功能。

需要注意的是，只有当引脚功能为通用 I/O 功能且单片机处于休眠或空闲模式时，唤醒功能才会受 PxWU 控制开启，其它状态下此唤醒功能不可用。

- **PxWU 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PxWU7 | PxWU6 | PxWU5 | PxWU4 | PxWU3 | PxWU2 | PxWU1 | PxWU0 |
| 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 |

**PxWUn**：I/O Px.n 端口唤醒功能控制位
0：除能
1：使能
PxWUn 位用于控制 Px.n 端口唤醒功能。这里的 x 可以是端口 A 和 B。但是，每个 I/O 端口实际有效位可能不同。

### 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，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。

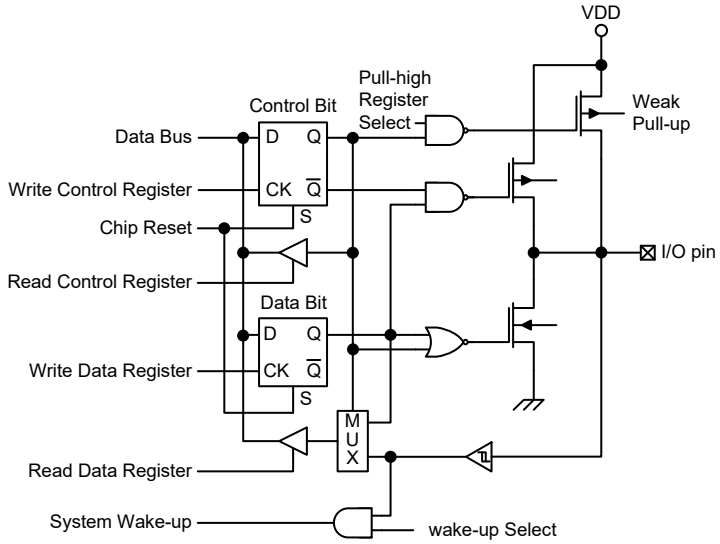
- **PxC 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PxC7 | PxC6 | PxC5 | PxC4 | PxC3 | PxC2 | PxC1 | PxC0 |
| 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 |

**PxCn**：I/O Px.n 端口输入 / 输出类型选择
0：输出
1：输入
PxCn 位用于控制 Px.n 端口输入 / 输出类型。这里的 x 可以是端口 A 和 B。但是，每个 I/O 端口实际有效位可能不同。

## 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。图中的引脚共用结构并非针对所有单片机。




逻辑功能输入 / 输出端口结构

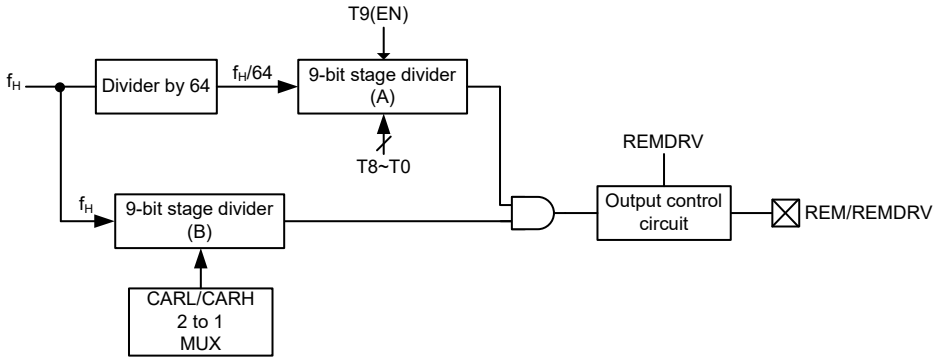
## 编程注意事项

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。

输入 / 输出口的每个引脚都带唤醒功能。单片机处于休眠或空闲模式时，有很多方法可以唤醒单片机，其中之一就是通过 PA 或 PB 任一引脚电平从高到低转换的方式，可以设置输入 / 输出口一个或多个引脚具有唤醒功能。

## 9-bit 定时器及载波输出

定时器是一个产生遥控传输模式的内部单元。其中包括一个 9-bit 的向下计数器。两组寄存器对，CARL1&CARL0 及 CARH1&CARH0，用于分别设置载波信号的低电平和高电平周期。REM/REMDRV 引脚可输出载波大电流驱动信号。




定时器及载波输出方框图

### 定时器及载波输出控制寄存器

定时器及载波输出功能由一系列寄存器控制完成。T8~T0 位用于设置 9-bit 向下计数器初始值。T9 位为定时器使能控制位。TOEF 位为定时器操作结束标志位。REMDRV 位用于选择双功能引脚 REM/REMDRV 的引脚功能。READYB 位用于指示 REMDRV 引脚输出功能是否就绪。CARL1&CARL0 寄存器用于设置载波低电平周期，CARH1&CARH0 用于设置高电平周期。CARH1 寄存器的 CH9 位用于启动载波输出。

| 寄存器名称 | 位 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TSR0 | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| TSR1 | TOEF | REMDRV | READYB | — | — | — | T9 | T8 |
| CARL0 | CL7 | CL6 | CL5 | CL4 | CL3 | CL2 | CL1 | CL0 |
| CARL1 | — | — | — | — | — | — | CL9 | CL8 |
| CARH0 | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| CARH1 | — | — | — | — | — | — | CH9 | CH8 |

定时器及载波输出控制寄存器列表

- **TSR0 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| 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　9-bit 定时器向下计数器 bit 7~0

对 TSR0 执行写入时仅将数据写入 TSR0 寄存器 (T7~T0)，对 TSR1 (T8) 执行写入时，则将指定数据和 TSR0 寄存器的内容传输到向下计数器。当数据从 TSR1 和 TSR0 传输到向下计数器完成之后，TOEF 位将被清零。

- **TSR1 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TOEF | REMDRV | READYB | — | — | — | T9 | T8 |
| R/W | R | R/W | R | — | — | — | R/W | R/W |
| POR | 1 | 0 | 0 | — | — | — | 0 | 0 |

Bit 7 **TOEF**：定时器操作结束标志位
0：定时器操作进行中
1：定时器操作完成

Bit 6 **REMDRV**：REM/REMDRV 引脚功能选择
0：REMDRV 功能
1：REM 功能

Bit 5 **READYB**：REMDRV 输出功能就绪标志位，载波输出前需确认此位值
0：REMDRV 功能已就绪，可输出载波
1：REMDRV 未就绪
READYB 位用于指示 REMDRV 引脚输出驱动器是否已就绪可以进行载波输出。当 MCU 从休眠或空闲模式中唤醒，或者当 REM/REMDRV 引脚功能从 REM 功能切换到 REMDRV 功能时，REMDRV 功能将被重新使能，输出驱动器需一段时间来稳定，之后才能输出载波。用户需在使能定时器前先通过轮询 READYB 位是否为零以确保 REMDRV 输出驱动器稳定。若 REM/REMDRV 引脚作为 REM 功能引脚，则该位无效并固定为零。

Bit 4~2 未定义，读为“0”

Bit 1 **T9**：定时器使能控制
0：除能
1：使能
将 T9 位置高，则定时器开始计数。当计数值减为零时，定时器停止计数，此时 TOEF 位被置高。若在定时器计数期间 T9 位被清零，定时器将停止计数。一旦设置 T9 位从 1→0→1，向下计数器将从 T8~T0 取得新数据，重新开始向下计数。

Bit 0 **T8**：9-bit 定时器向下计数器 bit 8
对 TSR0 执行写入时仅将数据写入 TSR0 寄存器 (T7~T0)，对 TSR1 (T8) 执行写入时，则将指定数据和 TSR0 寄存器的内容传输到向下计数器。当数据从 TSR1 和 TSR0 传输到向下计数器完成之后，TOEF 位将被清零。

- **CARL0 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CL7 | CL6 | CL5 | CL4 | CL3 | CL2 | CL1 | CL0 |
| 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 **CL7~CL0**：载波低电平周期控制位 bit 7~0

- **CARL1 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | CL9 | CL8 |
| R/W | — | — | — | — | — | — | R | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2 未定义，读为“0”

Bit 1 **CL9**：固定为“0”

Bit 0 **CL8**：载波低电平周期控制位 bit 8

- **CARH0 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| 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 **CH7~CH0**：载波高电平周期控制位 bit 7~0

- **CARH1 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | CH9 (CARY) | CH8 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 1 | 0 |

Bit 7~2 未定义，读为“0”

Bit 1 **CH9**：载波输出控制
0：有载波
1：无载波

Bit 0 **CH8**：载波高电平周期控制位 bit 8

## 定时器操作

在向下计数器中写入非零值，然后设置 T9 位为高，将启动计数器向下计数。注意若向下计数器的值为 000H，设置 T9 位为高也可启动定时器计数，但仅计数一次。此时定时器输出时间为 $[(0+1)\times 64/f_H=64/f_H]$。

在一个 $64/f_H$ 的周期，计数器减 1。如果计数器值为“0”，零检测器将发出一个定时器运行结束的信号使定时器停止计数。同时，TOEF 置 1。若计数器值为 0，则定时器运行结束的信号将一直发出，定时器处于停止状态。

定时器输出时间和计数器初始值 T[8:0] 关系如下所示。

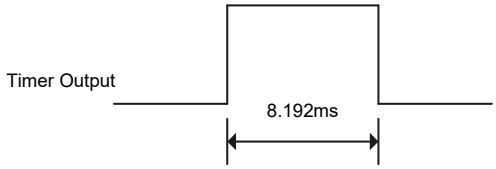
定时器输出时间 = $(T[8:0]+1)\times 64/f_H$

举例如下，其中 $f_H$=4MHz

```
MOV  A, 0FH
SWAP ACC
MOV  A, 0FH
MOV  TSR0, A
MOV  A, 00H
SWAP ACC
MOV  A, 01H
MOV  TSR1, A
SET  TSR1.1
```

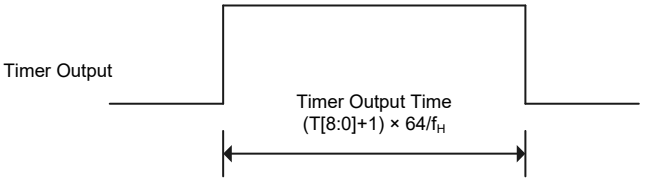
如上设置，则定时器输出时间如下所示：

$$(T[8:0]+1)\times 64/f_H=(511+1)\times 16\mu s=8.192ms$$

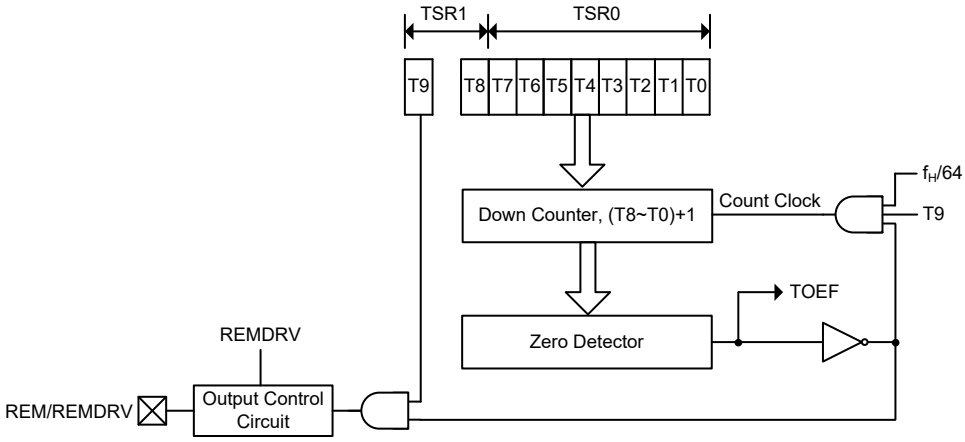



**定时器输出示意 – $f_H$=4MHz，T[8:0]=511**

设置 T9 位为高可使能 REM/REMDRV 引脚定时器输出。The REM/REMDRV 引脚输出的波形为定时器输出和载波信号的组合。



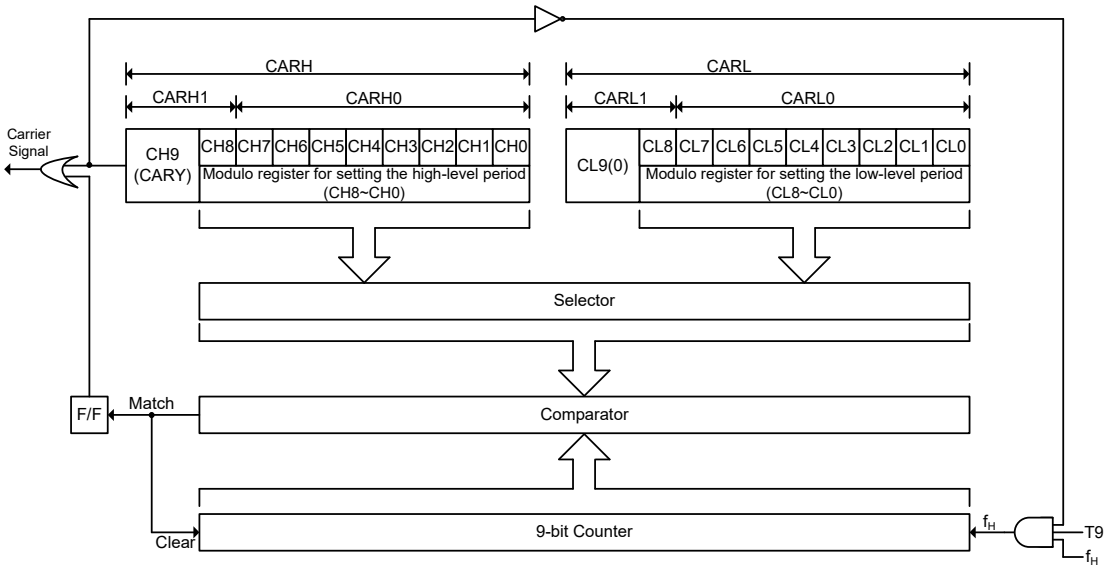

载波无输出时的定时器输出




定时器结构

## 载波输出

载波发生器由一个 9 位计数器和两个独立设置高、低电平周期的寄存器对 CARH、CARL 组成。




遥控载波发生器结构图

注：1. CARL1 寄存器中的 CL9 位固定为零。
2. TSR1 寄存器中的 T9 位用于使能定时器输出。

### 载波周期设置

载波占空比和载波频率是通过使用模寄存器分别设置高、低电平的宽度来决定的。在 $f_H$=4MHz 时，高低电平的宽度范围是 500ns~64μs。

下面的范例将介绍如何通过这两个寄存器对来设置载波的高、低电平周期。

另外需确保写入 CARL 和 CARH 的输入值在 001H~1FFH 范围之内。

**范例：**

注意，因与立即数相关的指令仅对 4-bit 低半字节有效。因此，在对一个 8-bit 进行立即数操作时，应先使用 SWAP 指令处理高半字节，接着再直接赋值给低 4 位。要实现下方的“MOV A, XXH”操作，应执行三条指令，具体操作见“算术逻辑单元 – ALU”章节范例。

```
MOV A,XXH       ; XXH=00H~FFH
MOV CARL0,A
MOV A,XXH       ; XXH 01H, CL8 (CARL1.0)
MOV CARL1,A
MOV A,XXH       ; XXH=00H~FFH
MOV CARH0,A
MOV A,XXH       ; XXH 02H, CH8 (CARH1.0)
MOV CARH1,A
CLR CARH1.1     ; The carrier is started by clearing CARY(CARH1.1)="0"
```

CARH 和 CARL 的值计算如下：

CARL (CARL1.0, CARL0.7~CARL0.0)=($f_H$×(1–D)×T)–1 …… (1)

CARH (CARH1.0, CARH0.7~CARH0.0)=($f_H$×D×T)–1 …… (2)

(1)+(2)⇒CARL+CARH=($f_H$×T)–2⇒Actual Carrier Frequency=$f_H$/(CARL+CARH+2)

D：载波占空比 (0<D<1)

$f_H$：输入时钟 (4MHz)

T：载波周期 (μs)

范例：

若 $f_H$=4MHz, Target fc=38kHz, T=1/fc=26.3157μs=$t_L$+$t_H$, duty=1/3

CARL=(4.00M×(1–1/3)×26.3157μs) – 1=69.1752

select 69=45H, actual $t_L$=(69+1)/4.00M=17.5μs

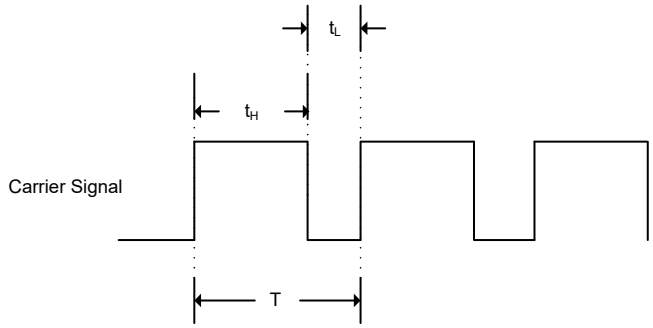
CARH=(4.00M×1/3×26.3157μs) – 1=34.087

select 34=22H, actual $t_H$=(34+1)/4.00M=8.75μs

实际载波频率 = $f_H$/(CARL+CARH+2)

实际 fc=$f_H$/(CARL+CARH+2)=4000kHz/(69+34+2)=38.09kHz

```
MOV A,04H
SWAP ACC
MOV A, 05H
MOV CARL0,A
MOV A,02H
SWAP ACC
MOV A, 02H
MOV CARH0,A
CLR CARH1.1     ; The carrier is started by clearing CARY(CARH1.1)="0"
```




载波高电平和低电平周期

| 目标值 | | 设置 | | 实际值 | | | |
|---|---|---|---|---|---|---|---|
| $f_C$ (kHz) | Duty | CARH (CH[8:0] bits) | CARL (CL[8:0] bits) | $t_H$ (µs) | $t_L$ (µs) | T (µs) | $f_C$ (kHz) |
| 36 | 1/3 | 24H | 49H | 9.25 | 18.50 | 27.75 | 36.04 |
| 38 | 1/3 | 22H | 45H | 8.75 | 17.50 | 26.25 | 38.10 |
| 56 | 1/3 | 17H | 2EH | 6.00 | 11.75 | 17.75 | 56.34 |
| 56 | 1/2 | 23H | 22H | 9.00 | 8.75 | 17.75 | 56.34 |

载波频率设置 ($f_H$=4MHz)

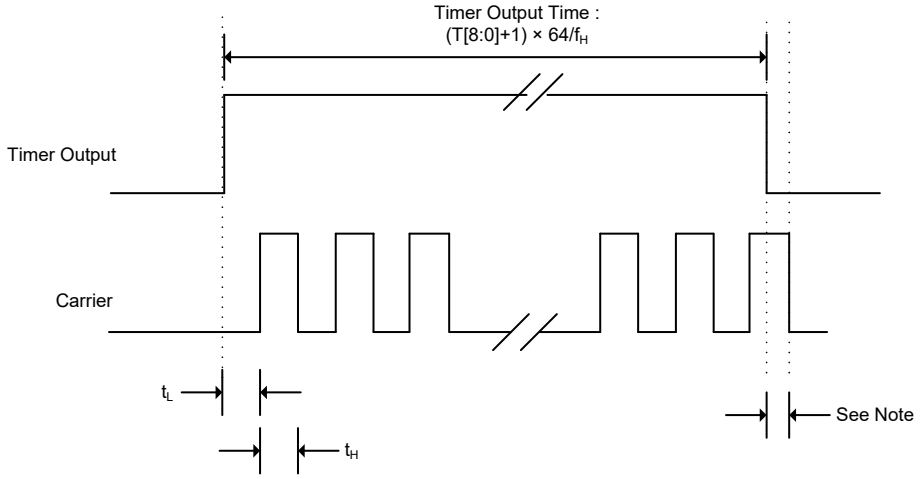
## 载波控制输出

遥控器载波从 REM 引脚输出，是通过清零 CARH1 寄存器中的 CH9 (CARY) 位来实现的。

若要执行载波输出，需确保在设置完 CARH ( CH[8:0] 位 ) 和 CARL ( CL[8:0] 位 ) 后，使能定时器操作。

注意，在 REM 引脚输出载波过程中，如果改变 CARH 和 CARL 的值，可能会产生错误。

使能定时器操作后，载波先输出低电平。



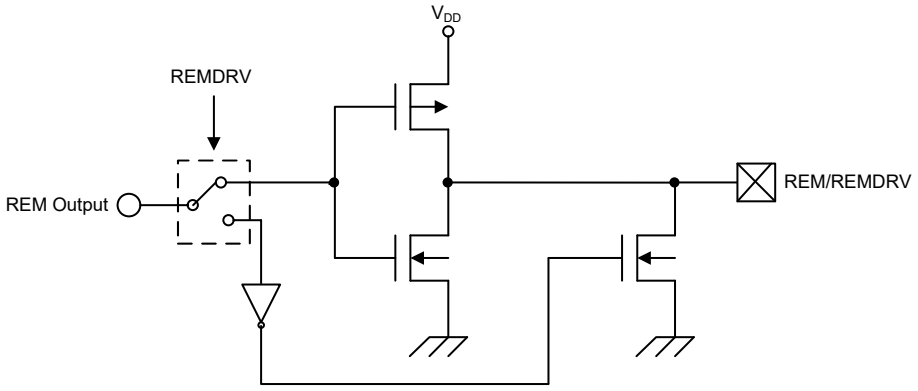

带载波输出的定时器输出

注：当载波信号使能和定时器的信号为高电平时，如果定时器输出要转换为低，则载波信号要先输出完高电平后再转换到低电平。

## 载波输出引脚

此产品提供一个双功能遥控载波输出引脚 REM/REMDRV。TSR1 寄存器中的 REMDRV 位用于控制此引脚作为 REM 功能还是 REMDRV 功能。REM 功能为 CMOS 输出结构，REMDRV 引脚是 NMOS 开漏输出结构。复位后，REM 功能输出低电平，REMDRV 载波输出引脚输出浮空。

如下方框图给出了 REM 和 REMDRV 引脚的大体结构图。所画的结构简化图仅仅是为了帮助更好的理解遥控载波输出引脚的功能，实际的载波输出引脚结构与所画的会有不同。




**REM/REMDRV 引脚结构示意图**

REM/REMDRV 引脚的输出由 CARH1 寄存器中的 CH9 (CARY) 位、定时器输出使能控制位 T9 和 9-bit 的向下计数器值 (T[8:0]) 共同决定。

| CH9 bit (CARY) | T9 bit | T[8:0] Bit | REM 引脚 (CMOS 输出 ) | REMDRV 引脚 (NMOS 输出 ) |
|---|---|---|---|---|
| 0 | 0 | 0 | 输出低电平 | 输出浮空 |
| 0 | 0 | 非零值 | | |
| 0 | 1 | 0 | 64/$f_H$ ( 载波输出 ) | 64/$f_H$ ( 载波输出 ) |
| 0 | 1 | 非零值 | 载波输出 (注) | 载波输出 |
| 1 | 0 | — | 输出低电平 | 输出浮空 |
| 1 | 1 | — | 输出高电平 | 输出低电平 |

**REM 引脚输出控制**

注：当 REM 引脚处于低电平时 (T9=0 或 T[8:0]=0)，需设置 CARH (CH[8:0]) 和 CARL (CL[8:0]) 的值。

# 中断

中断是单片机一个重要功能。当外部事件或内部功能产生中断时，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。此单片机仅提供一个内部中断功能即时基中断。

## 中断寄存器

中断控制基本上是在一定单片机条件发生时设置请求标志位，应用程序中中断使能位的设置是通过位于特殊功能数据存储器中的一个寄存器控制的，即用于设置基本中断的 INTC 寄存器。

寄存器中含有中断控制位和中断请求标志位。中断控制位用于使能或除能总中断和时基中断，中断请求标志位用于存放时基中断请求的状态。

- **INTC 寄存器**

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

Bit 7~6　未定义，读为“0”

Bit 5　**TBF**：时基中断请求标志位
0：无请求
1：中断请求

Bit 4~3　未定义，读为“0”

Bit 2　**TBE**：时基中断控制位
0：除能
1：使能

Bit 1　未定义，读为“0”

Bit 0　**EMI**：总中断控制位
0：除能
1：使能

## 中断操作

若中断事件条件产生，相关中断请求标志将置起。中断标志产生后程序是否会跳转至相关中断向量执行是由中断使能位的条件决定的。若使能位为“1”，程序将跳至相关中断向量中执行；若使能位为“0”，即使中断请求标志置起中断也不会发生，程序也不会跳转至相关中断向量执行。若总中断使能位为“0”，所有中断都将除能。

当中断发生时，下条指令的地址将被压入堆栈。相应的中断向量地址加载至 PC 中。系统将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”指令返回至主程序，以继续执行原来的程序。

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，直到堆栈减少为止。如果要求立刻动作，则堆栈必须避免成为储满状态。请求同时发生时，执行优先级如下流程图所示。

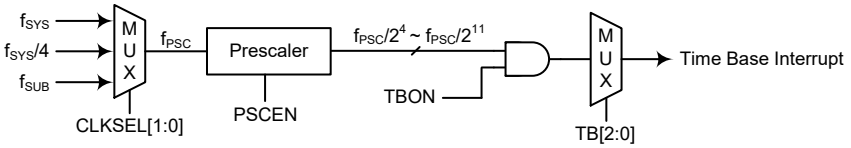
所有被置起的中断请求标志都可把单片机从休眠或空闲模式中唤醒，若要防止唤醒动作发生，在单片机进入休眠或空闲模式前应将相应的标志置起。

## 时基中断

时基中断向量位于程序存储器 008H 地址。

时基中断提供一个固定周期的中断信号，由其内部定时器功能产生溢出信号控制。当出现这种情况时其中断请求标志 TBF 被置位，中断请求发生。若要跳转到其相应的中断向量地址，总中断使能位 EMI 和时基使能位 TBE 需先被置位。当中断使能，堆栈未满且时基溢出时，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 TBF 会自动复位且 EMI 位会被清零以除能其它中断。

时基中断的目的是提供一个固定周期的中断信号。其时钟源 $f_{PSC}$ 来源于内部时钟源 $f_{SYS}$、$f_{SYS}/4$ 或者 $f_{SUB}$，经过一个分频器，其分频比可通过配置 TBC 寄存器中的位选择以获得更长的中断周期。




时基中断

- **PSCR 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | PSCEN | CLKSEL1 | CLKSEL0 |
| R/W | — | — | — | — | — | R/W | R/W | R/W |
| POR | — | — | — | — | — | 0 | 0 | 0 |

Bit 7~3　未定义，读为“0”

Bit 2　**PSCEN**：分频器时钟输出控制位
0：除能
1：使能
PSCEN 位用于使能 / 除能分频器时钟输出。在未使用时，可选择关闭此时钟，以减少功耗。

Bit 1~0　**CLKSEL1~CLKSEL0**：Prescaler 时钟源 $f_{PSC}$ 选择
00：$f_{SYS}$
01：$f_{SYS}/4$
1x：$f_{SUB}$

- **TBC 寄存器**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TBON | — | — | — | — | TB2 | TB1 | TB0 |
| R/W | R/W | — | — | — | — | R/W | R/W | R/W |
| POR | 0 | — | — | — | — | 0 | 0 | 0 |

Bit 7　**TBON**：时基使能控制位
0：除能
1：使能

Bit 6~3　未定义，读为“0”

Bit 2~0 **TB2~TB0**：时基溢出周期选择位
000：$2^4/f_{PSC}$
001：$2^5/f_{PSC}$
010：$2^6/f_{PSC}$
011：$2^7/f_{PSC}$
100：$2^8/f_{PSC}$
101：$2^9/f_{PSC}$
110：$2^{10}/f_{PSC}$
111：$2^{11}/f_{PSC}$

## 中断唤醒功能

每个中断都具有将处于休眠或空闲模式的单片机唤醒的能力。当中断请求标志由低到高转换时唤醒动作产生，其与中断是否使能无关。因此，尽管单片机处于休眠或空闲模式且系统振荡器停止工作，如有中断标志被置位，由此产生中断，因此必须注意避免伪唤醒情况的发生。若中断唤醒功能被除能，单片机进入休眠或空闲模式前相应中断请求标志应被置起。中断唤醒功能不受中断使能位的影响。

## 编程注意事项

通过禁止相关中断使能位，可以屏蔽中断请求，然而，一旦中断请求标志位被设定，它们会被保留在中断控制寄存器内，直到相应的中断服务子程序执行或请求标志位被软件指令清除。

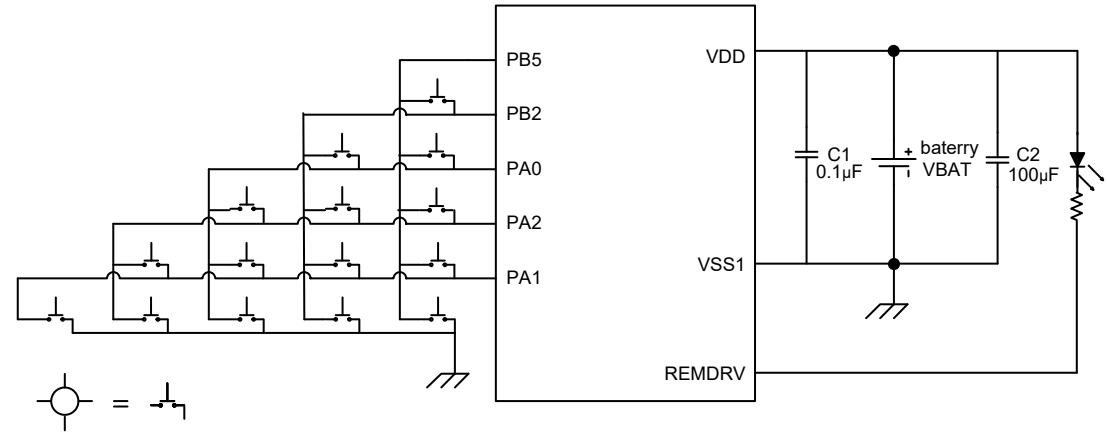
建议在中断服务子程序中不要使用“CALL 子程序”指令。中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

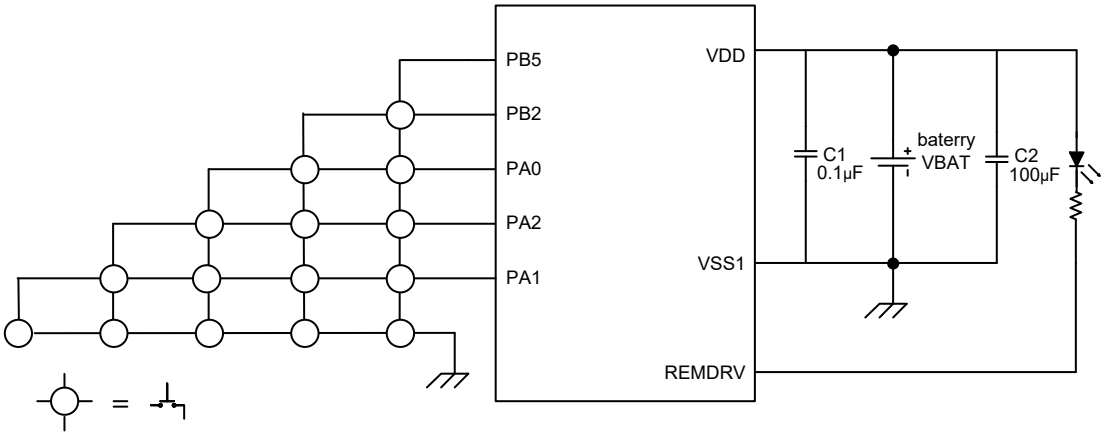
所有中断在休眠或空闲模式下都具有唤醒功能，当中断请求标志发生由低到高的转变时都可产生唤醒功能。若要避免相应中断产生唤醒动作，在单片机进入休眠或空闲模式前需先将相应请求标志置为高。

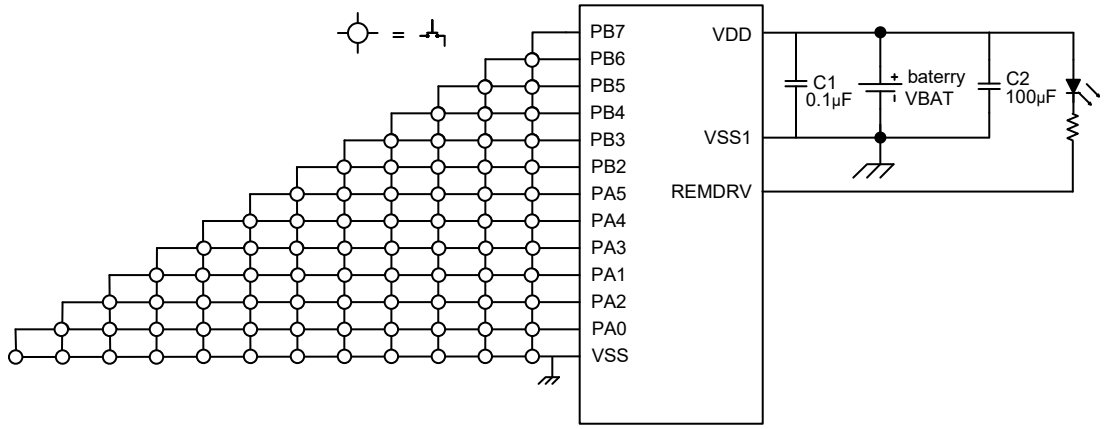
当进入中断服务程序，系统仅将程序计数器的内容压入堆栈，如果中断服务程序会改变状态寄存器或其它的寄存器的内容而破坏控制流程，应事先将这些数据保存起来。

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


PB5
PB2
PA0
PA2
PA1
VDD
VSS1
REMDRV
C1
0.1μF
baterry
VBAT
C2
100μF
=


PB5
PB2
PA0
PA2
PA1
VDD
VSS1
REMDRV
C1
0.1μF
baterry
VBAT
C2
100μF
=


=
PB7
PB6
PB5
PB4
PB3
PB2
PA5
PA4
PA3
PA1
PA2
PA0
VSS
VDD
VSS1
REMDRV
C1
0.1μF
baterry
VBAT
C2
100μF

# 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 Holtek 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 Holtek 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 Holtek 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，Holtek 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 |
| RLA [m] | 数据存储器左移一位，结果放入 ACC | 1 | 无 |

| 助记符 | 说明 | 指令周期 | 影响标志位 |
|---|---|---|---|
| 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] | 读取特定页或当前页的 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 |
| 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。

## 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，<br>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<br>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第 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<br>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<br>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)<br>[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)<br>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，<br>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)<br>[m].0 ← C<br>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)<br>ACC.0 ← C<br>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 $[m] \leftarrow ACC - [m] - \overline{C}$

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 or current page) to TBLH and Data Memory |
| 指令说明 | 将表格指针 (TBHP 和 TBLP，若无 TBHP 则仅 TBLP) 所指的程序代码低字节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 )<br>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 )<br>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 |

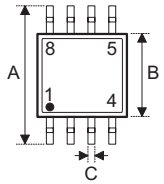
# 封装信息

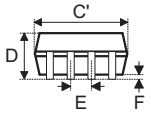
请注意，这里提供的封装信息仅作为参考。由于这个信息经常更新，提醒用户咨询 Holtek 网站 以获取最新版本的封装信息。

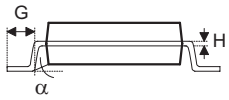
封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

- 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

## 8-pin SOP (150mil) 外形尺寸



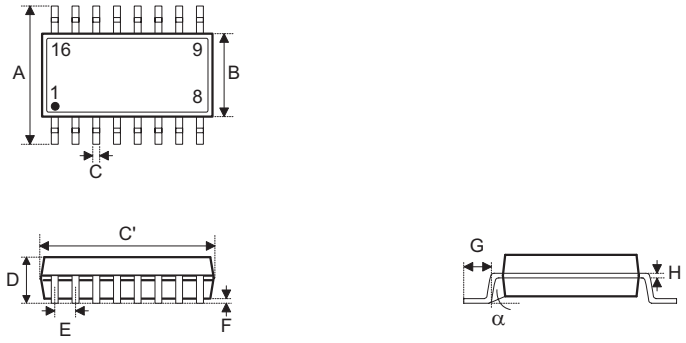




| 符号 | 尺寸 ( 单位：inch ) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| A | — | 0.236 BSC | — |
| B | — | 0.154 BSC | — |
| C | 0.012 | — | 0.020 |
| C′ | — | 0.193 BSC | — |
| D | — | — | 0.069 |
| E | — | 0.050 BSC | — |
| F | 0.004 | — | 0.010 |
| G | 0.016 | — | 0.050 |
| H | 0.004 | — | 0.010 |
| α | 0° | — | 8° |

| 符号 | 尺寸 ( 单位：mm ) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| A | — | 6.00 BSC | — |
| B | — | 3.90 BSC | — |
| C | 0.31 | — | 0.51 |
| C′ | — | 4.90 BSC | — |
| D | — | — | 1.75 |
| E | — | 1.27 BSC | — |
| F | 0.10 | — | 0.25 |
| G | 0.40 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |

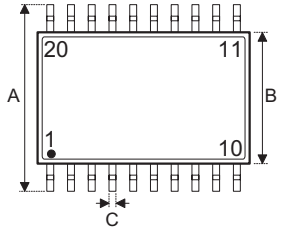
## 16-pin NSOP (150mil) 外形尺寸

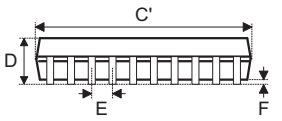


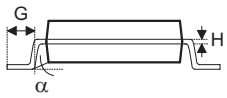

| 符号 | 尺寸 ( 单位：inch ) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| A | — | 0.236 BSC | — |
| B | — | 0.154 BSC | — |
| C | 0.012 | — | 0.020 |
| C’ | — | 0.390 BSC | — |
| D | — | — | 0.069 |
| E | — | 0.050 BSC | — |
| F | 0.004 | — | 0.010 |
| G | 0.016 | — | 0.050 |
| H | 0.004 | — | 0.010 |
| α | 0° | — | 8° |

| 符号 | 尺寸 ( 单位：mm ) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| A | — | 6.00 BSC | — |
| B | — | 3.90 BSC | — |
| C | 0.31 | — | 0.51 |
| C’ | — | 9.90 BSC | — |
| D | — | — | 1.75 |
| E | — | 1.27 BSC | — |
| F | 0.10 | — | 0.25 |
| G | 0.40 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |

## 20-pin NSOP (150mil) 外形尺寸



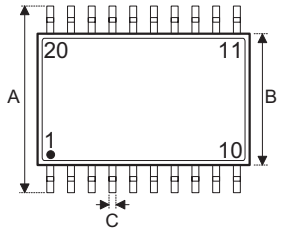



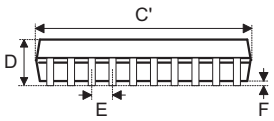



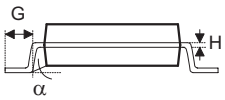

| 符号 | 尺寸 ( 单位：inch ) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| A | 0.228 | 0.236 | 0.244 |
| B | 0.146 | 0.154 | 0.161 |
| C | 0.009 | — | 0.012 |
| C’ | 0.382 | 0.390 | 0.398 |
| D | — | — | 0.069 |
| E | — | 0.032 BSC | — |
| F | 0.002 | — | 0.009 |
| G | 0.020 | — | 0.031 |
| H | 0.008 | —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

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

## 20-pin SSOP (150mil) 外形尺寸




| 符号 | 尺寸 ( 单位：inch ) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| A | — | 0.236 BSC | — |
| B | — | 0.154 BSC | — |
| C | 0.008 | — | 0.012 |
| C’ | — | 0.341 BSC | — |
| D | — | — | 0.069 |
| E | — | 0.025 BSC | — |
| F | 0.004 | — | 0.010 |
| G | 0.016 | — | 0.050 |
| H | 0.004 | — | 0.010 |
| α | 0° | — | 8° |

| 符号 | 尺寸 ( 单位：mm ) | | |
|---|---|---|---|
| | 最小值 | 典型值 | 最大值 |
| A | — | 6.00 BSC | — |
| B | — | 3.90 BSC | — |
| C | 0.20 | — | 0.30 |
| C’ | — | 8.66 BSC | — |
| D | — | — | 1.75 |
| E | — | 0.635 BSC | — |
| F | 0.10 | — | 0.25 |
| G | 0.41 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |