HT46F46E/HT46F47E/HT46F48E/HT46F49E
内置 EEPROM 经济 A/D 型 Flash 单片机

注意，规格中提到的 HT46F46E/ HT46F48E 产品已终止，目前不再可用。

特性

- 工作电压：
  f_{SYS}=4MHz：2.2V ~5.5V
  f_{SYS}=8MHz：3.3V~5.5V
  f_{SYS}=12MHz：4.5V~5.5V
- 13 到 23 个双向输入/输出口
- 与输入/输出口共用引脚的外部中断输入
- 8 位可编程定时/计数器，具有溢出中断和 7 级预分频器
- 外部晶体和 RC 振荡驱动电路
- 看门狗定时器
- 具有 PFD 功能，可以用来发声
- 暂停和唤醒功能可降低功耗
- 在 V_{DD} = 5V，系统频率为 8MHz 时，指令周期为 0.5μs
- 4 或 6 层硬件堆栈
- 4 通道 8 位或 9 位分辨率的 A/D 转换器
- 1 或 2 通道 8 位的 PWM 输出口，与输入/输出口共用引脚
- 位操作指令
- 查表指令
- 63 条指令
- 指令执行时间为 1 或 2 个指令周期
- 低电压复位功能
- 10,000 次可擦/写闪存程序存储器
- 100,000 次可擦/写 True EEPROM 数据存储器
- 闪存程序存储器数据有效期>10 年
- True EEPROM 数据有效期>10 年
- 在线烧写界面（ISP）
- 多种封装

概述

经济型 A/D 单片机是内置 True EEPROM 的 8 位 FLASH 型高性能精简指令集 MCU，专门为需要 A/D 转换的产品而设计，例如传感器信号输入。所有单片机都集成了多通道模数转换器和 1 或者 2 通道 PWM 输出。暂停和唤醒功能、振荡类型选择、可编程分频器等功能，使得实际应用时只需要很少的外部器件。

具有 A/D 和 PWM、低功耗、高性能、灵活的输入/输出口和低成本，使得这款单片机可以广泛应用于带传感器信号处理、马达驱动、工业控制、消费类产品和子系统控制器等。此系列芯片大部分特性都是通用的，然而，主要不同在于输入/输出引脚数目、RAM 和 ROM 的容量、A/D 分辨率、堆栈层数和封装类型。

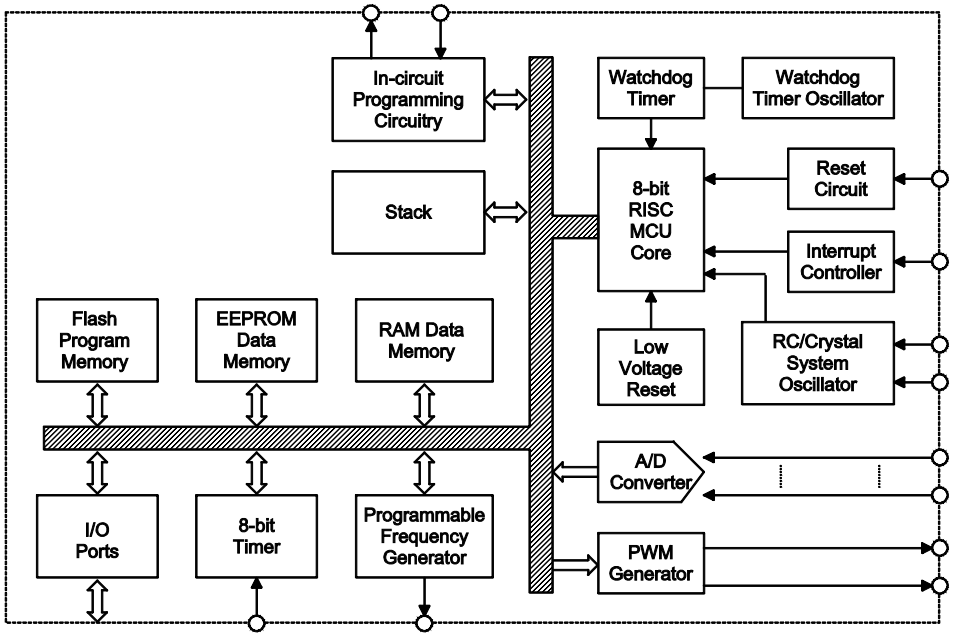
选型表

此系列芯片大部分特性都是通用的，主要区别在于程序存储器容量、输入/输出口数目、A/D 分辨率、堆栈的层数和封装的类型。以下表格为各单片机的主要特性。

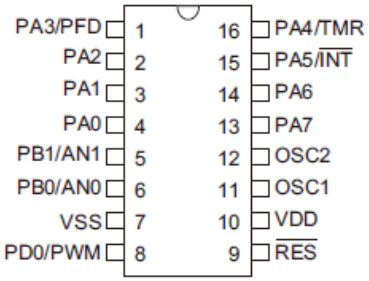
型号	电压	程序存储器	数据存储器	数据 EEPROM	输入/输出	定时器	中断	A/D	PWM	堆栈	封装种类
HT46F46E	2.2V~5.5V	1K×14	64×8	128×8	13	8-bit×1	3	8-bit×4	8-bit×1	4	16NSOP, 18SOP
HT46F47E	2.2V~5.5V	2K×14	64×8	128×8	13	8-bit×1	3	9-bit×4	8-bit×1	6	18SOP, 20SSOP
HT46F48E	2.2V~5.5V	2K×14	88×8	128×8	19	8-bit×1	3	9-bit×4	8-bit×1	6	24SOP/SSOP
HT46F49E	2.2V~5.5V	4K×15	128×8	256×8	23	8-bit×1	3	9-bit×4	8-bit×2	6	24/28SOP, 24SSOP

注意： 对于具有两种封装形式的单片机而言，选型表针对较大封装的情况。

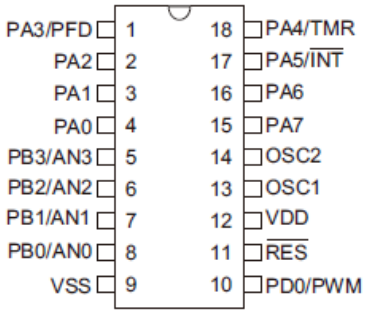
方框图



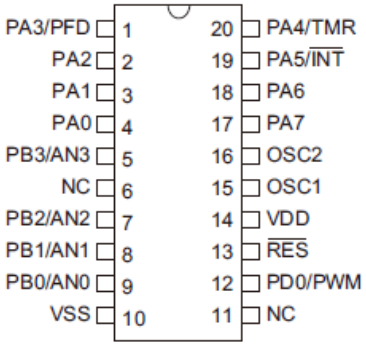
引脚图



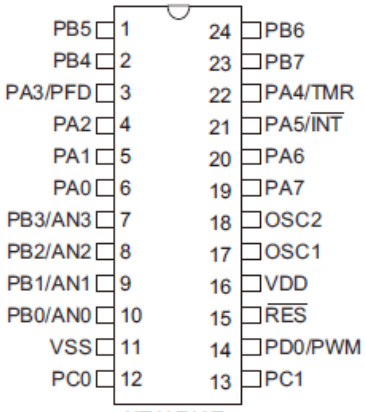
HT46F46E
16 NSOP-A



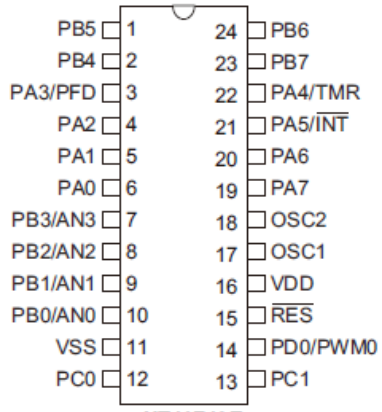
HT46F46E/HT46F47E
18 SOP-A



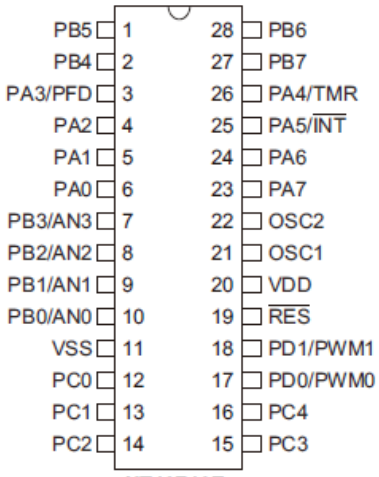
HT46F47E
20SSOP-A



HT46F48E
24 SOP-A/SSOP-A



HT46F49E
24 SOP-A/SSOP-A



HT46F49E
28 SOP-A

引脚说明

HT46F46E,HT46F47E

引脚名称	输入/输出	掩膜选项	说明
PA0~PA2 PA3/PFD PA4/TMR PA5/$\overline{\text{INT}}$ PA6~PA7	输入/输出	上拉电阻 唤醒功能 PA3 或 PFD	8 位双向输入/输出口。每一位可由掩膜选项设置为唤醒输入。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。PFD、TMR、$\overline{\text{INT}}$ 分别与 PA3，PA4、PA5 共用引脚。
PB0/AN0 PB1/AN1 PB2/AN2 PB3/AN3	输入/输出	上拉电阻	4 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。A/D 输入与 PB 口共用引脚。一旦 PB 作为 A/D 输入（由软件设置），则相应输入/输出功能和上拉电阻会自动失效。
PD0/PWM	输入/输出	上拉电阻 PD0 或 PWM	1 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。PWM 输出与 PD0 共用引脚（由 PWM 选项决定）。
OSC1 OSC2	输入 输出	晶体或 RC	OSC1、OSC2 连接 RC 或晶体（由掩膜选项确定）以产生内部系统时钟。在 RC 振荡方式下，OSC2 是系统时钟四分频的输出口。
$\overline{\text{RES}}$	输入	—	斯密特触发复位输入，低电平有效。
VDD	—	—	正电源。
VSS	—	—	负电源，接地。

注：1、PA 的每个端口都可通过掩膜选项设定成具有唤醒功能。

2、引脚可以单独选择带上拉电阻。

3、引脚 PB2/AN2～PB3/AN3 在 16－pin 封装中没有引出。

4、未引出的端口可设置成输出或带上拉的输入以降低功耗。

HT46F48E

引脚名称	输入/输出	掩膜选项	说明
PA0~PA2 PA3/PFD PA4/TMR PA5/$\overline{\text{INT}}$ PA6~PA7	输入/输出	上拉电阻 唤醒功能 PA3 或 PFD	8 位双向输入/输出口。每一位可由掩膜选项设置为唤醒输入。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。PFD、TMR、$\overline{\text{INT}}$ 分别与 PA3，PA4、PA5 共用引脚。
PB0/AN0 PB1/AN1 PB2/AN2 PB3/AN3 PB4~PB7	输入/输出	上拉电阻	8 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。A/D 输入与 PB 口共用引脚。一旦 PB 作为 A/D 输入（由软件设置），则相应输入/输出功能和上拉电阻会自动失效。
PC0~PC1	输入/输出	上拉电阻	2 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。
PD0/PWM	输入/输出	上拉电阻 输入/输出或 PWM	1 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。PWM 输出与 PD0 共用引脚（由 PWM 选项决定）。
OSC1 OSC2	输入 输出	晶体或 RC	OSC1、OSC2 连接 RC 或晶体（由掩膜选项确定）以产生内部系统时钟。在 RC 振荡方式下，OSC2 是系统时钟四分频的输出口。

引脚名称	输入/输出	掩膜选项	说明
$\overline{RES}$	输入	—	斯密特触发复位输入，低电平有效。
VDD	—	—	正电源。
VSS	—	—	负电源，接地。

注：1、PA 的每个端口都可通过掩膜选项设定成具有唤醒功能。

2、引脚可以单独选择带上拉电阻。

HT46F49E

引脚名称	输入/输出	掩膜选项	说明
PA0~PA2 PA3/PFD PA4/TMR PA5/$\overline{INT}$ PA6~PA7	输入/输出	上拉电阻 唤醒功能 PA3 或 PFD	8 位双向输入/输出口。每一位可由掩膜选项设置为唤醒输入。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。PFD、TMR、$\overline{INT}$ 分别与 PA3，PA4、PA5 共用引脚。
PB0/AN0 PB1/AN1 PB2/AN2 PB3/AN3 PB4~PB7	输入/输出	上拉电阻	8 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。A/D 输入与 PB 口共用引脚。一旦 PB 作为 A/D 输入（由软件设置），则相应输入/输出功能和上拉电阻会自动失效。
PC0~PC4	输入/输出	上拉电阻	5 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。
PD0/PWM0 PD1/PWM1	输入/输出	上拉电阻 输入/输出 或 PWM	2 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定）的斯密特触发输入。PWM 输出与 PD0 或 PD1 共用引脚（由 PWM 选项决定）。
OSC1 OSC2	输入 输出	晶体或 RC	OSC1、OSC2 连接 RC 或晶体（由掩膜选项确定）以产生内部系统时钟。在 RC 振荡方式下，OSC2 是系统时钟四分频的输出口。
$\overline{RES}$	输入	—	斯密特触发复位输入，低电平有效。
VDD	—	—	正电源。
VSS	—	—	负电源，接地。

注：1、PA 的每个端口都可通过掩膜选项设定成具有唤醒功能。

2、引脚可以单独选择带上拉电阻。

3、PC2~PC4 和 PD1/PWM1 在 24-pin 的封装中没有引出。

4、未引出的端口可设置成输出或带上拉的输入以降低功耗。

极限参数

电源供电……..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……150mA
总功耗…… 500mW

储存温度…….-60℃~150℃
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	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	f_{SYS}=4MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	3.3	—	5.5	V
			f_{SYS}=12MHz	4.5	—	5.5	V
I_{DD1}	工作电流(晶体振荡)	3V	无负载，f_{SYS}=4MHz ADC 关闭	—	0.6	1.5	mA
		5V		—	2	4	mA
I_{DD2}	工作电流(RC 振荡)	3V	无负载，f_{SYS}=4MHz ADC 关闭	—	0.8	1.5	mA
		5V		—	2.5	4	mA
I_{DD3}	工作电流 (晶体振荡，RC 振荡)	5V	无负载，f_{SYS}=8MHz ADC 关闭	—	4	8	mA
I_{DD4}	工作电流 (晶体振荡，RC 振荡)	5V	无负载，f_{SYS}=12MHz ADC 关闭	—	5	10	mA
I_{STB1}	静态电流(看门狗打开)	3V	无负载，系统 HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	静态电流(看门狗关闭)	3V	无负载，系统 HALT	—	—	1	μA
		5V		—	—	2	μA
V_{IL1}	输入/输出口、TMR、$\overline{INT}$ 的低电平输入电压	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	输入/输出口、TMR、$\overline{INT}$ 的高电平输入电压	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	低电平输入电压($\overline{RES}$)	—	—	0	—	$0.4\ V_{DD}$	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	—	LVR 打开，2.1V 选择	1.98	2.10	2.22	V
			LVR 打开，3.15V 选择	2.98	3.15	3.32	V
			LVR 打开，4.2V 选择	3.98	4.20	4.42	V
I_{OL}	输入/输出口灌电流	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH}	输入/输出口源电流	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V	V_{OH}=0.9V_{DD}	-5	-10	—	mA
R_{PH}	上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V_{AD}	A/D 输入电压	—	—	0	—	V_{DD}	V
E_{AD}	A/D 转换误差	—	—	—	±0.5	±1	LSB
I_{ADC}	打开 ADC 增加的功耗	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA

交流电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{SYS}	系统时钟	—	2.2V~5.5V	400	—	4000	kHz
			3.3V~5.5V	400	—	8000	kHz
			4.5V~5.5V	400	—	12000	kHz
f_{TIMER}	定时器输入频率（TMR）	—	2.2V~5.5V	0	—	4000	kHz
			3.3V~5.5V	0	—	8000	kHz
			4.5V~5.5V	0	—	12000	kHz
t_{WDTOSC}	看门狗振荡器周期	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	外部复位低电平脉冲宽度	—	—	1	—	—	μs
t_{SST}	系统启动延迟时间	—	从 HALT 状态唤醒	—	1024	—	$*t_{SYS}$
t_{LVR}	低电压复位时间	—	—	0.25	1	2	ms
t_{INT}	中断脉冲宽度	—	—	1	—	—	μs
t_{AD1}	A/D 时钟周期-HT46F46E	—	—	0.5	—	—	μs
t_{AD2}	A/D 时钟周期-HT46F47E/HT46F48E/HT46F49E	—	—	1	—	—	μs
t_{ADC1}	A/D 转换时间-HT46F46E	—	—	—	64	—	t_{AD1}
t_{ADC2}	A/D 转换时间-HT46F47E/HT46F48E/HT46F49E	—	—	—	76	—	t_{AD2}
t_{ADCS1}	A/D 采样时间-HT46F46E	—	—	—	32	—	t_{AD1}
t_{ADCS2}	A/D 采样时间-HT46F47E/HT46F48E/HT46F49E	—	—	—	32	—	t_{AD2}

注：$*t_{SYS}=1/f_{SYS}$

EEPROM 交流电气特性

Ta=25℃

符号	参数	V_{CC}=5V±10%		V_{CC}=2.2V±10%		单位
		最小	最大	最小	最大	
f_{SK}	时钟频率	0	2	0	1	MHz
t_{SKH}	SK 高电平时间	250	—	500	—	ns
t_{SKL}	SK 低电平时间	250	—	500	—	ns
t_{CSS}	CS 建立时间	50	—	100	—	ns
t_{CSH}	CS 保持时间	0	—	0	—	ns
t_{CDS}	CS 屏蔽时间	250	—	250	—	ns
t_{DIS}	DI 建立时间	100	—	200	—	ns
t_{DIH}	DI 保持时间	100	—	200	—	ns
t_{PDI}	D0 等待“1”时间	—	250	—	500	ns
t_{PD0}	D0 等待“0”时间	—	250	—	500	ns
t_{SV}	状态有效时间	—	250	—	250	ns
t_{PR}	写周期时间	—	5	—	5	ms

系统结构

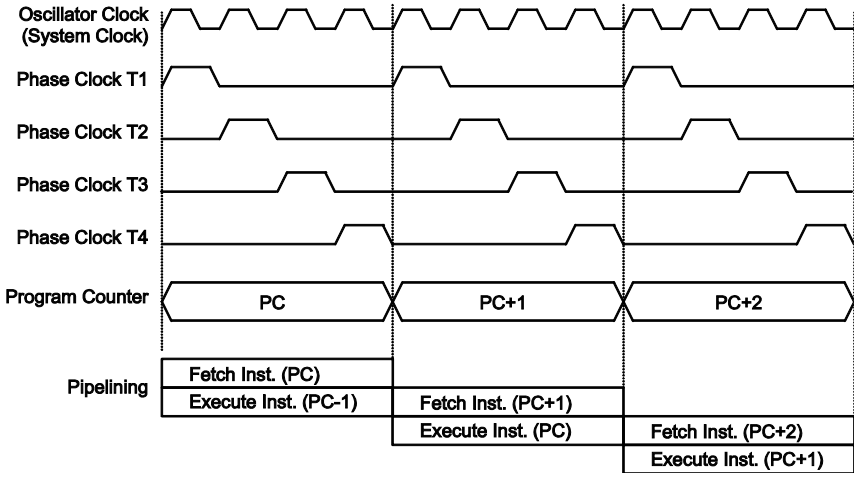
内部系统结构是盛群半导体公司内置 EEPROM 经济 A/D 型 FLASH 单片机具有良好运行性能的主要因素。由于采用 RISC 结构，此系列单片机具有高运算速度和高性能的特性。通过流水线的方式，指令的取得和执行同时进行，此举使得除了分支和调用指令外，其它指令都能在一个指令周期内完成。8 位的 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、加、减和分支等功能，而内部的数据路径则以通过累加器或 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供较大可靠度和灵活性的 I/O 和 A/D 控制系统时，仅需要少数的外部器件。使得这些单片机适合用在低成本和批量生产的控制应用上，且可以提供 1K 至 4K 字的程序存储器和 64 至 128 字节数据储存。

时序和流水线结构

系统时钟由晶体/陶瓷振荡器或 RC 振荡器提供，细分为 T1~T4 四个内部产生的非重叠时钟。程序计数器在 T1 时自动加一并抓取一条新的指令，剩下的 T2~T4 时钟完成指令的解码和执行，因此一个 T1~T4 时钟形成一个指令周期。虽然指令的取得和执行发生在连续的指令周期，但单片机流水线的结构会保证指令在一个指令周期内被有效的执行，特殊的情况发生在程序计数器的内容被改变的时候，如子程序的调用或跳转，在这情况下指令将需要多一个指令周期的时间去执行。

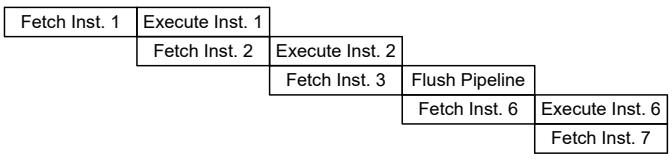
当使用 RC 振荡器时，OSC2 可以获得一个 T1 相时钟同步信号，这个 T1 相时钟有 $f_{SYS}/4$ 的频率，拥有 1:3 高/低的占空比。

如果指令牵涉到分支，例如跳转或调用等指令，则需要两个机器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，因此用户必须特别考虑额外周期的问题，尤其是在执行时间要求较严格时。



系统时序和流水线

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



指令捕捉

程序计数器

程序执行期间，程序计数器用来指向下一条要执行的指令地址。除了 JMP 或 CALL 这些要求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后自动增加一。对于内置 EEPROM 经济 A/D 型 FLASH 系列的单片机，根据所选择的单片机型号不同，程序计数器宽度会因程序存储器容量的不同而不同。然而必须要注意只有低 8 位，即程序计数器低字节寄存器 PCL，是可以让使用者直接读写的。

当执行的指令要求跳转到非连续的地址时，如跳转指令、子程序调用、中断或复位等，单片机通过载入所需的地址到程序计数器来控制程序。对于条件跳转指令，一旦条件符合，下一条在现在指令执行时所取得的指令即会被摒弃，而由一个空指令周期来加以取代。

程序计数器低字节，即程序计数器低字节寄存器 PCL，可以通过程序控制取得，且它是可以读取和写入的寄存器。通过直接传送数据到这寄存器，一个程序短跳转可以直接被执行，然而因为只有低字节的运用是有效的，因此跳转被限制在同页存储器，即 256 个存储器地址的范围内，当这样一个程序跳转要执行时，需注意会插入一个空指令周期。

程序计数器低字节在程序控制下是完全可用的。PCL 的使用可能导致程序分支，所以额外的周期需要预先取得。有关 PCL 寄存器更多的信息可在特殊功能寄存器部份中找到。

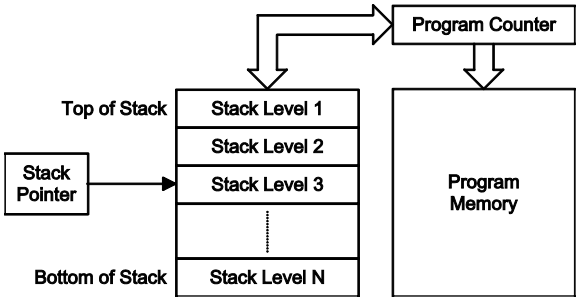
模式	程序计数器											
	b11	**b10**	**b9**	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
初始化复位	0	0	0	0	0	0	0	0	0	0	0	0
外部中断	0	0	0	0	0	0	0	0	0	1	0	0
定时/计数器溢出	0	0	0	0	0	0	0	0	1	0	0	0
A/D 转换中断	0	0	0	0	0	0	0	0	1	1	0	0
条件跳跃	PC+2											
装载 PCL	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
跳转、子程序调用	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

注： PC11～PC8：当前程序计数器位
@7～@0：PCL 位
#11～#0：指令码位
S11～S0：堆栈寄存器位
对于 HT46F49E，程序计数器为 12 位，即 b11～b0
对于 HT46F47E 和 HT46F48E，程序计数器为 11 位，即 b10～b0，表格中 b11 列无效
对于 HT46F46E，程序计数器为 10 位，即 b9～b0，表格中 b11 和 b10 列无效

堆栈

堆栈是存储器中一个特殊的部分，它只用来储存程序计数器中的内容。根据选择的单片机，堆栈可介于 4 或 6 层之间，它们既不是数据部份也不是程序空间部份，且既不可读取也不可写入。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堆栈。当子程序或中断服务程序结束时，返回指令（RET 或 RETI）使程序计数器从堆栈中返回它之前的值。当芯片复位之后，SP 将指向堆栈的顶部。



注：对于 HT46F46E 而言，有 4 层堆栈；而对于 HT46F47E、HT46F48E 和 HT46F49E，具有 6 层堆栈。

如果堆栈已满，且有非屏蔽的中断发生，中断请求标志位会被置位，但是中断响应将被禁止。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，但会造成堆栈溢出。使用时应避免堆栈溢出的情况发生，因为这可能会造成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份，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运算，并将结果储存在指定的寄存器，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

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，此系列所有单片机提供用户灵活便利的调试方法和项目开发规划。

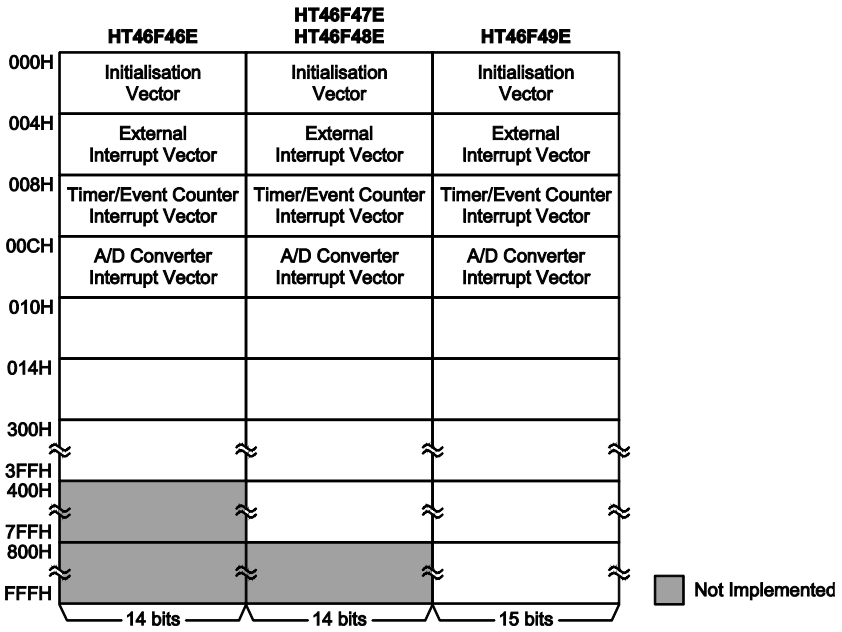
结构

14 位的程序存储器的容量是 1K 或 2K，15 位的程序存储器的容量则是 4K，这取决于选用哪种单片机。程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口，数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。

特殊向量

程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。

- 地址 000H
  该地址为程序初始化保留。系统复位后，程序总是从 000H 开始执行。
- 地址 004H
  该地址为外部中断服务程序保留。当 $\overline{INT}$ 引脚转成低电平，如果中断允许且堆栈未满，则程序会跳转到 004H 地址开始执行。
- 地址 008H
  该地址为定时/计数器中断服务程序保留。当定时/计数器溢出，如果中断允许且堆栈未满，则程序会跳转到 008H 地址开始执行。
- 地址 00CH
  该地址为 A/D 转换中断服务程序保留。当 A/D 转换完成，如果中断允许且堆栈未满，则程序会跳转到 00CH 地址开始执行。



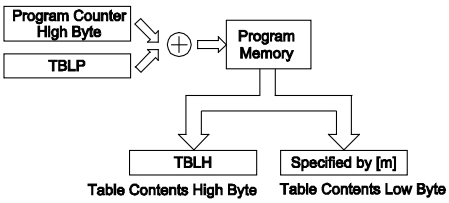
程序存储器结构

查表

程序存储器中的任何地址都可以定义成一个表格，以便储存固定的数据。使用表格时，表格指针必须先行设定，其方式是将表格的低字节地址放在表格指针寄存器 TBLP 中。这个寄存器定义表格低字节的 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]”指令从当前的程序所在的存储器页或存储器最后一页中来查表读取。当这些指令执行时，程序存储器中表格数据低字节，将被传送到使用者所指定的数据存储器[m]，程序存储器中表格数据的高字节，则被传送到 TBLH 特殊寄存器，而高字节中未使用的位将被读取为“0”。

下图是查表中寻址/数据流程：



查表范例

以下范例说明 HT46F47E 中，表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器的最后一页，在此 ORG 伪指令中的值为 700H，即 2K 程序存储器 HT46F47E 单片机中最后一页存储器的起始地址，而表格指针的初始值则为 06H，这可保证从数据表格读取的第一笔数据位于程序存储器地址 706H，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRDC [m]”指令被使用，则表格指针指向当前页。在这个例子中，表格数据的高字节等于零，而当“TABRDL [m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

```
tempreg1   db ? ;temporary register #1
tempreg2   db ? ;temporary register #2
     :
     :

mov    a,06h    ;initialize table pointer - note that this address
                ;is referenced

mov    tblp,a   ;to the last page or present page
     :
     :

tabrdl tempreg1 ;transfers value in table referenced by table pointer
                ;to tempreg1
                ;data at prog. memory address 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 (for HT46F47E)

dc   00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
     :
```

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

指令	表格地址											
	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC[m]	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL[m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

表格区

注：PC11～PC8：当前程序计数器位

@7～@0：表格指针 TBLP 位

对于 HT46F49E 而言，表格地址为 12 位，即 b11～b0

对于 HT46F47E 和 HT46F48E 而言，表格地址为 11 位，即 b10～b0

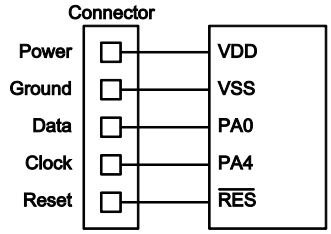
对于 HT46F46E 而言，表格地址为 10 位，即 b9～b0

在线编程

FLASH 型程序存储器提供用户和设计者便利地对同一芯片进行程序的更新和修改。HOLTEK 单片机提供在线编程的方式。用户可将进行过编程或未经过编程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，在无需去除或者重新插入芯片的情况下方便地保持程序为最新版本。

引脚名称	说明
PA0	串行数据输入/输出
PA4	串行时钟输入
$\overline{RES}$	复位
VDD	电源
VSS	地

芯片内部程序存储器和 EEPROM 存储器都可以通过 5 线的接口在线进行编程。其中 PA0 用于数据串行下载或上传、PA4 用于串行时钟、两条用于提供电源，另外一条用于复位。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。



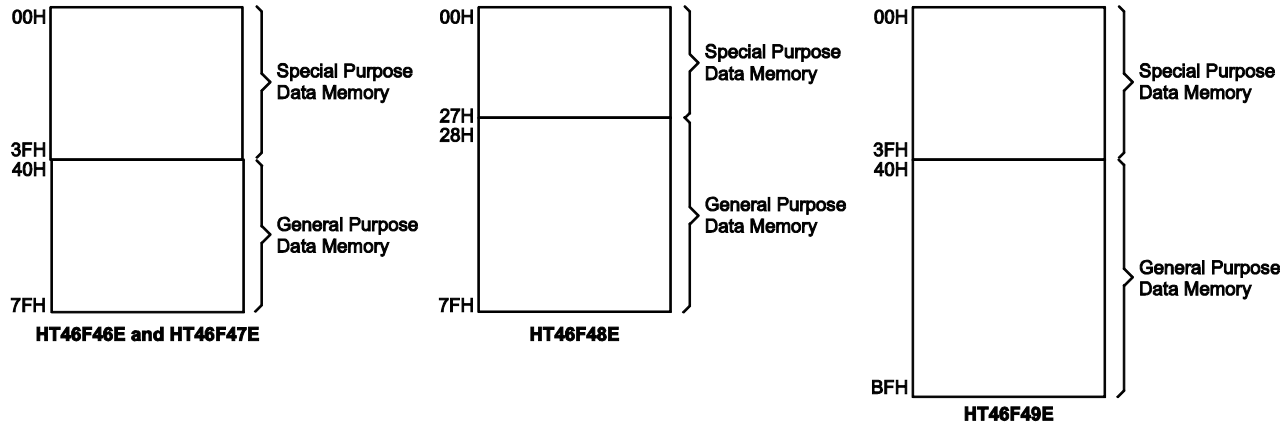
在线编程接口

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。分为两部份，第一部份是特殊功能寄存器，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。第二部份数据存储器是作一般用途使用，都可在程序控制下进行读取和写入。

结构

数据存储器分为两个区块，即 BANK0 和 BANK1，全部 8 位宽度。数据存储器大部分位于 BANK0 区块，分为两个部份，即专用和通用数据存储器。所有单片机的数据存储器的起始地址都是 00H。常见的寄存器，如 ACC 和 PCL 等，全都具有相同的数据存储器地址。



数据存储器结构

注意：除部分特殊位，大部分数据存储区可以通过“SET [m].i”和“CLR [m].i”直接寻址。数据存储器也可以通过指针 MP0 和 MP1 间接寻址。

数据存储器 BANK1 区块仅包含一个特殊功能寄存器 EECR，用来控制 EEPROM 通讯，全部具有相同地址 40H。

BANK1 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。这个数据存储区可让使用者进行读取和写入的操作。使用“SET [m].i”和“CLR [m].i”指令可对个别的位做置位或复位的操作，方便用户在数据存储器内进行位操作。

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份。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得到“00H”的值。

特殊功能寄存器

为了确保单片机能成功的工作，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（如定时器和中断等）和外部功能（如 I/O 数据控制和 A/D 转换操作）的正确工作。在数据存储器中，这些寄存器以 00H 作为起始地址。在特殊功能寄存器和通用数据存储器的起始地址之间，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返回 00H 值。

间接寻址寄存器 — IAR0，IAR1

间接寻址寄存器 IAR0 和 IAR1 位于数据存储区，并没有实际的物理地址。间接寻址的方法准许使用间接寻址指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器 IAR0 和 IAR1 上的任何动作，将对间接寻址指针 MP0 或 MP1 所指定的存储器地址产生对应的读/写操作。IAR0 和 MP0 配合使用将对 BANK0 区块数据进行存取，而 IAR1 和 MP1 配合使用可以对 BANK0 和 BANK1 区块数据进行存取。直接读取间接寻址寄存器将返回 00H 的结果，而直接写入此寄存器则不做任何操作。

间接寻址指针 — MP0, MP1

对于该系列所有单片机，系统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象普通的寄存器一般被写入和操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由间接寻址指针所指定的地址。当 MP0 和 IAR0 配合使用将对 BANK0 区块数据进行存取，而 MP1 和 IAR1 配合使用可以对 BANK0 和 BANK1 区块数据进行存取。

对于内含 64 或 88 个字节 RAM 空间的单片机，间接寻址指针的第 7 位没有作用。然而必须注意当读取间接寻址指针时，第 7 位将返回“1”。

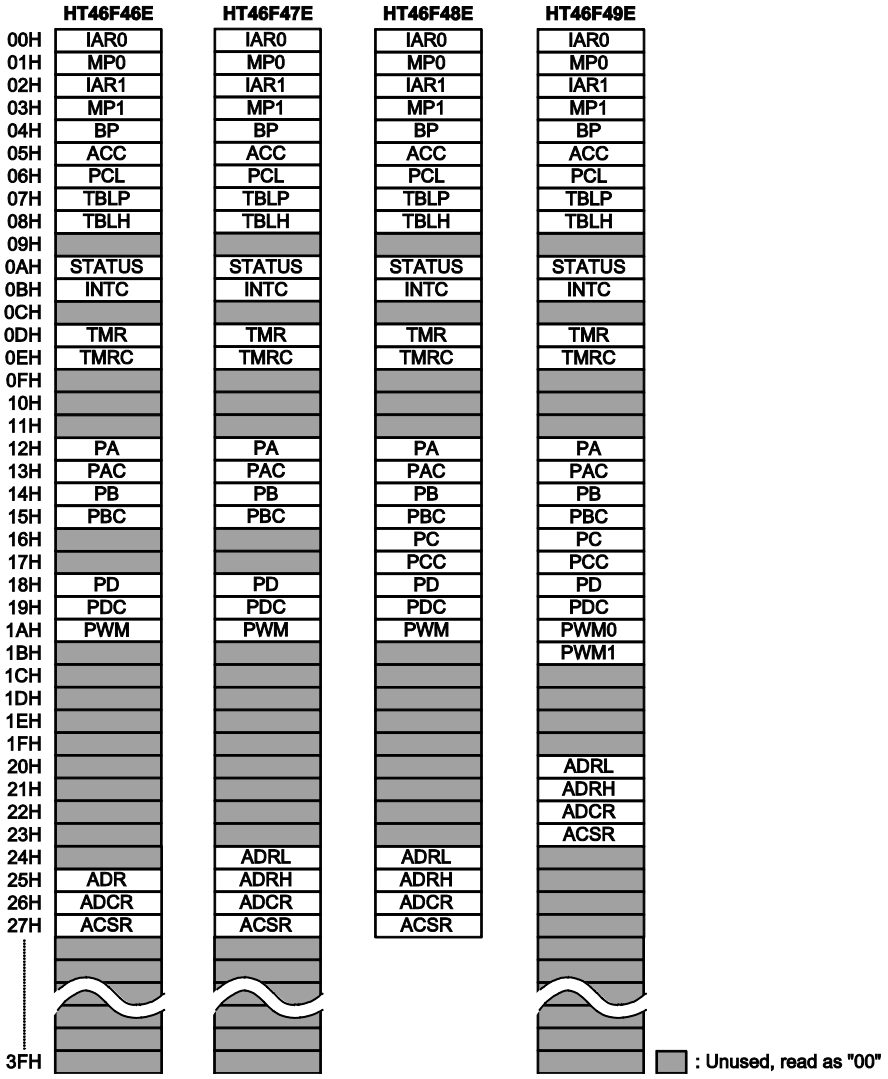
以下例子说明如何清除一个具有 4 RAM 地址的区块，它们已事先定义成地址 adres1 到 adres4。

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

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

continue:
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。



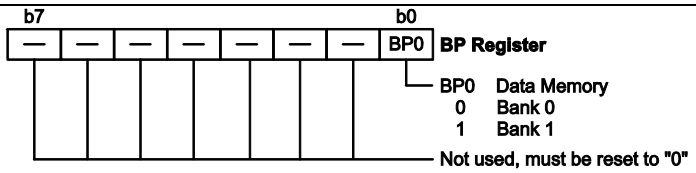
特殊数据存储器结构

存储区指针 – BP

数据存储器分为两个区块，即 BANK0 和 BANK1。除了 EECR 寄存器外，其他特殊寄存器和通用寄存器都位于 BANK0。区块 BANK1 仅有一个 EEPROM 控制寄存器，即 EECR。可以使用存储区指针 BP 来选择对应的数据存储器区块。如果对 BANK0 进行数据存取，必须先设置存储区指针 BP 的值为 00H；如果对 BANK1 进行数据存取，必须先设置存储区指针 BP 的值为 01H。

将 MP0 和 IAR0 配合使用，无需设置存储区指针 BP 的值，即可对 BANK0 进行数据存取。寄存器 EECR 位于 BANK1 中 40H 地址，内部 EEPROM 数据的写入与读出可通过 EECR 进行操作，将 MP1 和 IAR1 配合使用，存储区指针 BP 的值设置为“01H”即可对 EECR 进行间接存取。

复位后，通用数据存储器会初始化到 BANK0，但是在 HALT 模式下的 WDT 溢出复位，不会改变通用数据存储器的存储区块。请注意，特殊功能数据存储器不受存储区的影响，也就是说，不论是在 BANK0、BANK1 都能对特殊功能寄存器进行读写操作。直接寻址只会对 BANK0 内数据寄存器进行存取，而无需设置存储区指针 BP 的值。



存储区指针

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储存在 ACC 累加器。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寄存器和另一个寄存器之间传送数据时，由于两寄存器之间不能直接传送数据，因此必须通过累加器来传送数据。

程序计数器低字节寄存器 — PCL

为了提供额外的程序控制功能，程序计数器低字节设置在数据存储器的特殊功能区域内，程序员可对此寄存器进行操作，很容易的直接跳转到其它程序地址。直接给 PCL 寄存器赋值将导致程序直接跳转到程序存储器的某一地址，然而由于寄存器只有 8 位的长度，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储存在程序存储器中的表格进行操作。TBLP 为表格指针，指向表格数据的地址。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

8 位的 STATUS 寄存器包含零标志位（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），它同时记录算术/逻辑运算的和系统工作状态数据。

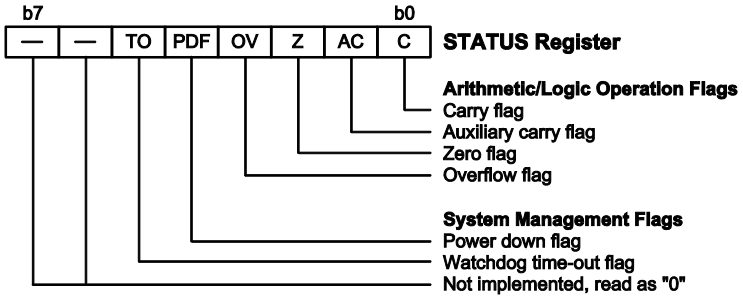
除了 TO 和 PDF 标志位外，状态寄存器中的位像其它大部份寄存器一样可以被改变，但任何数据写入到状态寄存器将不会改变 TO 或 PDF 标志位。另外，执行不同的指令后，与状态寄存器有关的运算可能会得到不同的结果。TO 标志位只会受系统上电、看门狗溢出、或执行“CLR WDT”或“HALT”指令影响。PDF 标志位只会受执行“HALT”或“CLR WDT”指令或系统上电影响。

Z、OV、AC 和 C 标志位通常反映最近运算的状态：

- 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，同时 C 也会被带进位/借位的移位指令所影响。
- 当低半字节加法运算的结果产生进位，或高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。
- 当算术或逻辑运算结果是零时，Z 被置位，否则 Z 被清零。

- 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。
- 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。
- 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。



状态寄存器

中断控制寄存器 — INTC

8 位的 INTC 寄存器用来控制外部和内部中断的动作。通过标准的位操作指令来设定这些寄存器的位，每个中断的使能/除能功能可分别被控制。INTC 寄存器内的总中断位 EMI 控制所有中断的使能/除能，用来设定所有中断使能位的开或关。当一个中断程序被响应时，就会自动屏蔽其它中断，EMI 位将被清除，而执行“RETI”指令则会置位 EMI 位。

定时/计数器 — TMR，TMRC

该系列的单片机提供了一个 8 位定时/计数器。TMR 寄存器用于存放 8 位设定值，可以预先写入固定的数据，以允许设定不同的时间中断。而 TMRC 寄存器设置定时/计数器的工作模式、打开或关闭定时/计数器。

输入/输出控制寄存器

在特殊功能寄存器中，输入/输出寄存器和它们相对应的控制寄存器很重要。所有的输入/输出端口都有相对应的寄存器，标示为 PA、PB、PC 和 PD。如数据存储器结构图中所示，这些输入/输出寄存器映射到数据存储器的特定地址，用以传送端口上的输入/输出数据。每个输入/输出端口有一个相对应的控制寄存器，分别为 PAC、PBC、PCC 和 PDC，也同样映射到数据存储器的特定地址。这些控制寄存器设定引脚的状态，以决定哪些是输入口，哪些是输出口。要设定一个引脚为输入，控制寄存器对应的位必须设定成逻辑高，若引脚设定为输出，则控制寄存器对应的位必须设为逻辑低。程序初始化期间，在从输入/输出端口中读取或写入数据之前，必须先设定控制寄存器的位以确定引脚为输入或输出。使用“SET [m].i”和“CLR [m].i”指令可以直接设定这些寄存器的某一位。这种在程序中可以通过改变输入/输出端口控制寄存器中某一位而直接改变该端口输入/输出状态的能力是此系列单片机非常有用的特性。

脉宽调制寄存器 — PWM，PWM0，PWM1

此系列单片机都提供了 1 个或 2 个脉冲宽度调制器(即 PWM)。每个 PWM 都具有自己独立的控制寄存器。对于只有一个脉冲宽度调制器的单片机，它的控制寄存器为 PWM。具有两个脉冲宽度调制器的单片

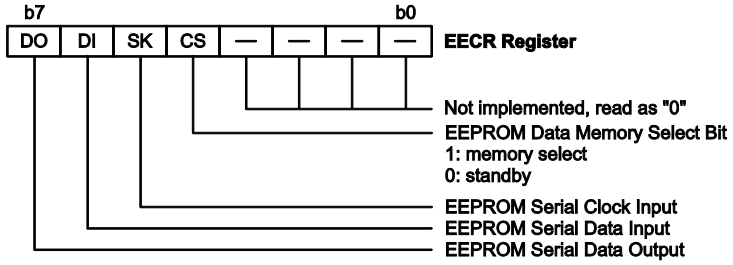
机，控制寄存器为 PWM0 和 PWM1。这些 8 位的寄存器定义相应的脉冲宽度调制器调制周期的占空比。

A/D 转换寄存器 — ADR，ADRL，ADRH，ADCR，ACSR

此系列单片机都提供了 4 通道 8 位或 9 位 A/D 转换器。A/D 转换需要涉及到 1 个或 2 个数据存储器、一个控制寄存器和一个时钟控制寄存器。对于 HT46F46E，A/D 转换结果寄存器为 8 位的 ADR；而对于此系列其它 9 位 A/D 分辨率单片机，其 A/D 转换结果寄存器为高位 ADRH 寄存器和低位 ADRL 寄存器。当一个模数转换周期结束后，转换出的数字量将保存在这些寄存器中。通道的选择和 A/D 转换器的设置通过寄存器 ADCR 控制，A/D 时钟频率由时钟源寄存器 ACSR 定义。

EEPROM 控制寄存器 – EECR

此系列所有单片机的一个特性是内建 EEPROM 数据存储器。“Electrically Erasable Programmable Read Only Memory”为电可擦可编程只读存储器，由于其非挥发的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。通过将这些存储器合并，对设计者来说增加了许多新的应用机会，EEPROM 的存储功能使得此系列单片机可以用来存储产品编号、校准值、用户特定数据、系统配置参数或其他产品信息等。



EEPROM 控制寄存器

EEPROM 数据存储器

内部 EEPROM 数据寄存器容量为 128×8 位或者 256×8 位，取决于选用哪种单片机。由于映射方式与程序存储器和数据存储器不同，只能使用间接寻址且必须通过 EECR 寄存器进行操作。

单片机	EEPROM 存储器容量
HT46F49E 除外	128×8
HT46F49E	256×8

EEPROM 数据存储器容量

EEPROM 数据存储器的存取

内部 EEPROM 数据寄存器的操作由 7 条指令实现，控制 EEPROM 数据寄存器的 READ、WRITE、ERASE、EWEN 等。内部 EEPROM 和三线制 EEPROM 结构相似，4 个引脚用来传输指令，地址和数据信息，即芯片片选 CS，串行时钟 SK，数据输入 DI 和数据输出 DO。通过间接寻址 BANK1 区块 EECR 寄存器，控制 EEPROM 的 CS、SK，DI 和 DO 信号，以软件的方式产生 EEPROM 的操作时序，进而达到读写 EEPROM 的目的。

位	符号	EEPROM 功能
0~3	—	未用，读取为“0”
4	CS	EEPROM 数据寄存器选择位
5	SK	串行时钟位：用来将数据写入或读出 EEPROM
6	DI	数据输入位：用来将指令，地址和数据信息写入内部 EEPROM
7	DO	数据输出位：用来将内部 EEPROM 数据读出 没有数据读出时此位将是高阻态

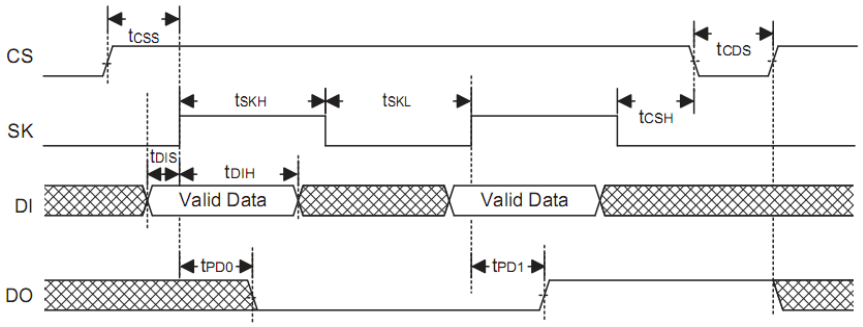
EECR 寄存器 — 控制位功能

当执行 READ 指令，数据从内部 EEPROM 数据寄存器读取，SK 上升沿时 DO 数据有效，否则 DO 为高阻态。DI 线上的串行数据会在 SK 上升沿写入内部 EEPROM 数据寄存器。当指令，地址和数据信息发送完成后，必须马上将 CS 置低。上电复位后，操作内部 EEPROM 数据寄存器必须重新进行初始化。

寄存器 EECR 的操作只能通过 MP1 和 IAR1 间接寻址方式进行，由于位于 BANK1 区块 40H 地址，对寄存器 EECR 操作前必须将 MP1 设置为 40H 且将存储区指针 BP 设置为 1。

EEPROM 数据寄存器指令设置

控制内部 EEPROM 的所有操作指令共有 7 条，比如读、写，禁止和使能等。当 CS 引脚置位高电平且将起始位“1”发出后，相关的指令通过 DI 引脚写入内部 EEPROM 数据寄存器。对于 READ，WRITE 和 ERASE 指令有两位操作码对应，然后写入地址信息，对于容量为 128×8 的单片机，地址是 7 位。对于容量为 256×8 的单片机，9 位的地址信息会被写入，然而高位是空位，可以为任意值。地址以高位在前的格式发出。



EEPROM 数据输入和输出时序

对于 EWEN、EWDS，ERAL 和 WRAL 这 4 条指令，起始位“1”发出后，紧接着发出两位的操作码“00”， 对于 EEPROM 存储器容量 128×8 的单片机，然后写入 7 位长度的地址信息，而对于 EEPROM 存储器容量 256×8 的单片机，则写入 9 位长度的地址信息。地址信息高两位定义可参考下面表格，地址信息其余位为任意值，可设置为 0 或 1。

当 WRITE 或 ERASE 指令发出后，内部 EEPROM 寄存器写周期将数据写入芯片，内部写数据期间

EEPROM 无法接收指令，直至内部写周期结束。上电复位后，对内部 EEPROM 数据寄存器操作前必须进行初始化，以确保操作正确进行。

指令	描述	起始位	指令码	地址	数据
READ	Read Out Data Byte(s)	1	10	A6~A0	D7~D0
ERASE	Erase Single Data Byte	1	11	A6~A0	—
WRITE	Write Single Data Byte	1	01	A6~A0	D7~D0
EWEN	Erase/Write Enable	1	00	11 XXXXX	—
EWDS	Erase/Write Disable	1	00	00 XXXXX	—
ERAL	Erase All	1	00	10 XXXXX	—
WRAL	Write All	1	00	01 XXXXX	D7~D0

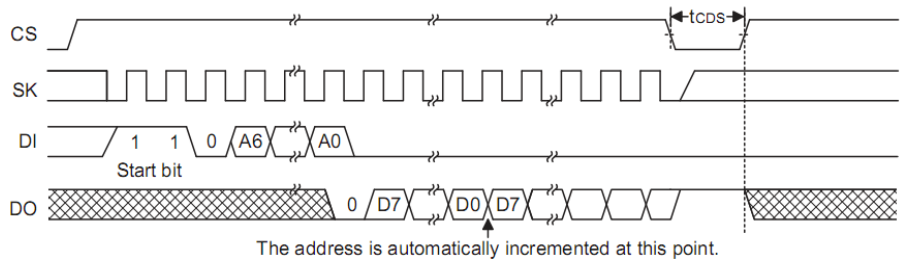
指令设置摘要 — HT46F49E 除外

指令	描述	起始位	指令码	地址	数据
READ	Read Out Data Byte(s)	1	10	X,A7~A0	D7~D0
ERASE	Erase Single Data Byte	1	11	X,A7~A0	—
WRITE	Write Single Data Byte	1	01	X,A7~A0	D7~D0
EWEN	Erase/Write Enable	1	00	11 XXXXXXX	—
EWDS	Erase/Write Disable	1	00	00 XXXXXXX	—
ERAL	Erase All	1	00	10 XXXXXXX	—
WRAL	Write All	1	00	01 XXXXXXX	D7~D0

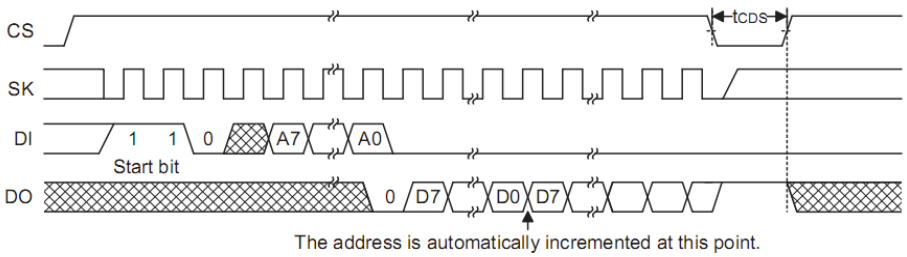
指令设置摘要 — HT46F49E

READ

READ 指令用于从内部 EEPROM 读取一个或者多个数据。发送 READ 指令前必须将 CS 置位高电平，紧接着为起始位“1”和两位操作码“10”，所有数据由 DI 发送。随后地址信息以高位在前的模式发出。对于单片机 HT46F49E，最后一位指令操作码和地址信息高位之间必须插入 1 个空位。在最后一位地址，即 A0 发出后，数据信息以 D7 开始，以高位在前模式在 SK 上升沿由 DO 读取。然而，在首位数据 D7 之前有一个逻辑“0”信号。一个字节数据被读取之后，EEPROM 内地址会自动增加，允许下一个连续的数据被读取，而不需要输入下一数据的地址。只要 CS 保持为高电平，下一个地址的数据 D7 将紧接着上一个地址的数据 D0 之后被读取，两数据间也不会再有逻辑“0”插入。地址会循环累加直到 CS 从高电平变为低电平。READ 指令执行完后 DO 为高阻态，注意的是 READ 指令不受 EWEN 和 EWDS 影响，无论 EWEN 或者 EWDS 命令执行与否，READ 命令都可以有效执行。



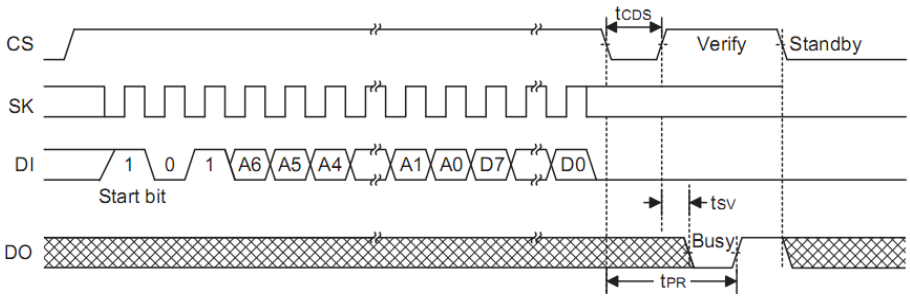
READ 时序 — HT46F49E 除外



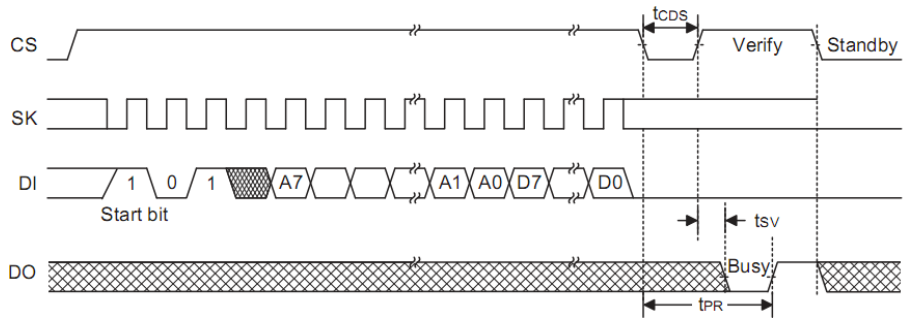
READ 时序 — HT46F49E

WRITE

WRITE 指令用于向内部 EEPROM 写入一个数据。发送 WRITE 指令前必须将 CS 置位高电平，紧接着为起始位“1”和两位操作码“01”，所有数据由 DI 发送。随后地址信息以高位在前的模式发出。在最后一位地址，即 A0 发出后，数据信息以高位在前模式发出。对于单片机 HT46F49E，最后一位指令操作码和地址信息高位之间必须插入 1 个空位。当 WRITE 指令、地址和数据送出后，EEPROM 会在 CS 下降沿进入内部写周期。由于内部写周期包括先擦除再写入的过程，因此在写入数据前不需要执行擦除命令。内部写周期时钟由 EEPROM 时钟发生器提供，并不需要 SK 信号。内部写数据期间 EEPROM 无法接收指令，直至内部写周期结束。EEPROM 执行内部写周期期间，可通过设置 CS 为高电平，循环检测 DO 位状态决定内部写周期是否结束，DO 位保持低电平说明内部写周期仍在进行；当 DO 恢复到高电平，内部写周期结束。执行 WRITE 指令前必须先发出 EWEN 指令，以使能内部 EEPROM 的擦写。



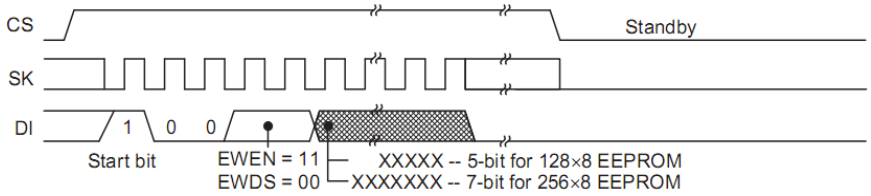
WRITE 时序 — HT46F49E 除外



WRITE 时序 — HT46F49E

EWEN/EWDS

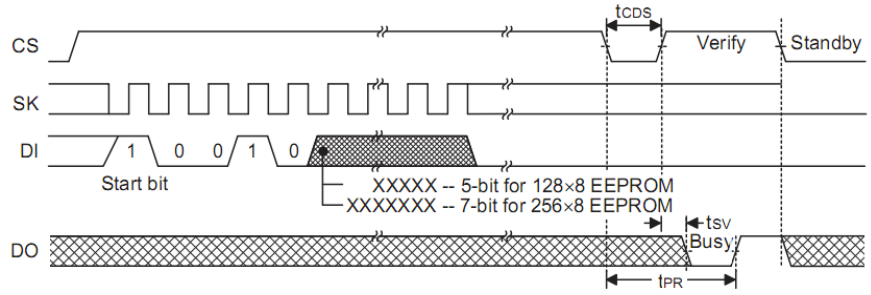
EWEN/EWDS 指令可以使能或禁止擦写动作有效执行。发送 EWEN 或 EWDS 指令前必须将 CS 置位高电平，紧接着为起始位“1”和两位操作码“00”。执行 EWEN 指令将继续发送“11”，执行 EWDS 指令将继续发送“00”。最后依据内置 EEPROM 为 128×8 或者 256×8 的存储器容量，分别将 5 位或者 7 位任意数据发出以结束指令过程。当内部 EEPROM 寄存器处于禁止擦写状态时，EEPROM 存储器内数据无法被改写，这样 EEPROM 内数据处于写保护状态。执行 WRITE、ERASE、WRAL 或 ERAL 指令之前，必须先执行 EWEN 命令使能擦写，擦写使能直到芯片掉电或 EWDS 指令被执行。



EWEN/EWDS 时序

ERAL

在擦写使能模式下，ERAL 命令可以将所有存储器单元擦除为逻辑“1”状态。发送 ERAL 指令前必须将 CS 置位高电平，紧接着为起始位“1”和两位操作码“00”，继续发送“10”，最后依据内置 EEPROM 为 128×8 或者 256×8 的存储器容量，分别将 5 位或者 7 位任意数据发出以结束指令过程。ERAL 命令送出后，EEPROM 会在 CS 下降沿执行内部擦除动作。内部擦除动作时钟由 EEPROM 时钟发生器提供，并不需要 SK 信号。内部擦除数据期间 EEPROM 无法接收指令，直至内部擦除周期结束。EEPROM 执行内部写周期期间，可通过设置 CS 为高电平，循环检测 DO 位状态决定内部写周期是否结束，DO 位保持低电平说明内部写周期仍在进行；当 DO 恢复到高电平，内部写周期结束。执行 ERAL 指令之前，必须先执行 EWEN 命令使能擦写。

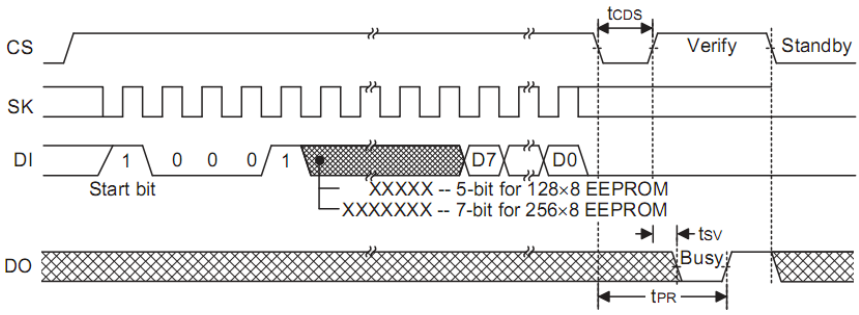


ERAL 时序

WRAL

在擦写使能模式下，WRAL 指令可以将相同数据写入内部 EEPROM 寄存器所有单元。发送 WRAL 指令前必须将 CS 置位高电平，紧接着为起始位“1”和两位操作码“00”，继续发送“01”，最后依据内置 EEPROM 为 128×8 或者 256×8 的存储器容量，分别将 5 位或者 7 位任意数据发出以结束指令过程。WRAL 命令送出后，数据信息以高位在前模式发出，EEPROM 会在 CS 下降沿进入内部写周期。内部写周期时钟由 EEPROM 时钟发生器提供，并不需要 SK 信号。内部写数据期间 EEPROM 无法接收指令，直至内部写周期结束。EEPROM 执行内部写周期期间，可通过设置 CS 为高电平，循环检测 DO 位状态决定内部写周期是否结束，DO 位保持低电平说明内部写周期仍在进行；当 DO 恢复到高电平，内部写周期结束。执行

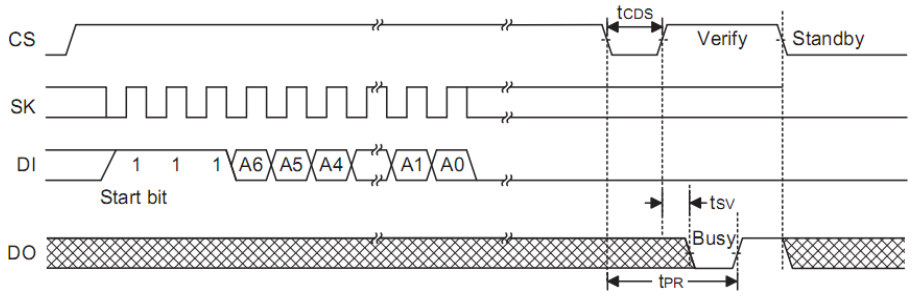
WRAL 指令之前，必须先执行 EWEN 命令使能擦写。执行 WRAL 指令时内部写周期包括先擦除再写入的过程，因此在写入数据前不需要执行擦除命令。



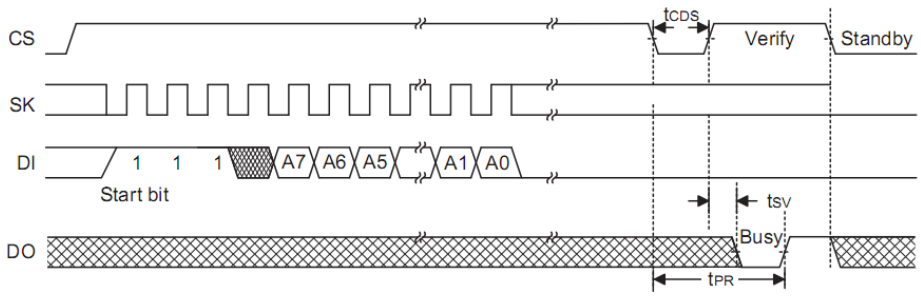
WRAL 时序

ERASE

在擦写使能模式下，ERASE 指令用来擦除指定地址的数据，指定地址的数据将设置为“1”。发送 ERASE 指令前必须将 CS 置位高电平，紧接着为起始位“1”和两位操作码“11”，所有数据由 DI 发送。随后地址信息以高位在前的模式发出。对于单片机 HT46F49E，最后一位指令操作码和地址信息高位之间必须插入 1 个空位。ERASE 指令和指定地址发出后，EEPROM 会在 CS 下降沿执行内部擦除动作，将指定地址的数据设置为全 1。内部擦除动作时钟由 EEPROM 时钟发生器提供，并不需要 SK 信号。EEPROM 内部写数据期间无法接收指令，直至内部写周期结束。EEPROM 执行内部写周期期间，可通过设置 CS 为高电平，循环检测 DO 位状态决定内部写周期是否结束，DO 位保持低电平说明内部写周期仍在进行；当 DO 恢复到高电平，内部写周期结束。执行 ERASE 指令之前，必须先执行 EWEN 命令使能擦写。



ERASE 时序 — HT46F49E 除外

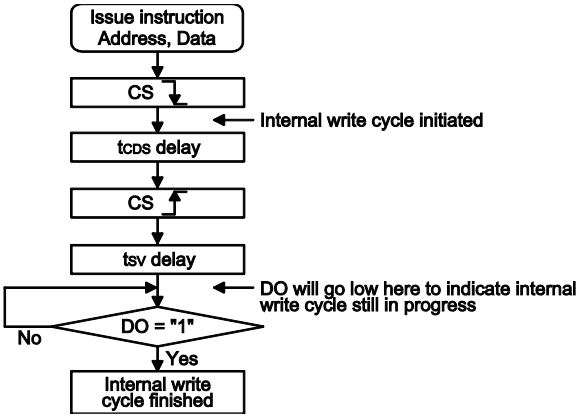


ERASE 时序 — HT46F49E

内部写周期

执行 WRITE、ERASE，ERAL 和 WRAL 指令后，内部 EEPROM 寄存器进入内部写或擦除过程，内部擦除动作时钟由 EEPROM 时钟发生器提供，并不需要 SK 信号。虽然单片机无法控制 EEPROM 内部写的时序，但是仍然有一些方法检测出内部写周期是否已经结束。在这个内部写周期过程中，EEPROM 不会执行其他任何的命令，直至内部写周期结束。

检测 EEPROM 内部写周期是否结束的一种方法就是在 CS 下降沿检查 DO 信号。CS 下降沿将初始化 EEPROM 内部写周期，如果内部写周期没有结束，DO 为低电平；如果内部写周期结束，DO 为高电平，内部 EEPROM 寄存器可以接收其他指令。



内部写周期巡检时序

EEPROM 初始化

当单片机上电复位后，用户对内部 EEPROM 寄存器执行操作之前必须执行一段初始化程序，比如执行 WRITE 指令将随机数据写入某一地址前必须首先发出 EWEN 指令，然后执行 EWDS 指令结束内部 EEPROM 的操作（此步骤为可选）。

下列范例程序实现如何上电后初始化 EEPROM：

```
mov     A, 01h
mov     BP, A           ;set to bank1
mov     A,40h
mov     MP1,A           ; set MP1 to EECR address
call    EWEN            ;subroutine to run EWEN instructions
mov     A, 7Fh
mov     EEADDR, A
mov     A, 55h
mov     EEDATA, A
call    WRITE           ;subroutine to run WRITE instructions
                        ;write 55h data to address 7FH
call    EWDS            ;optional subroutine to run EWDS instruction
```

EEPROM 程序范例

下面几个范例程序列出如何将指令发出，对EEPROM进行读写操作。这些程序能形成对EEPROM存取数据的原理的基本的理解。这些程序供有相同的256×8位容量的内部EEPROM存储器的HT46F49E单片机使用。对于另外的有更小的128×8位容量的内部EEPROM存储器的单片机，插入在最后一位指令操作码和地址信息高位之间1个空位没有发出。

范例 1-定义和发送 EEPROM 操作指令

```
_CS             EQU     IAR1.4          ;EEPROM lines setup to have corresponding
_SK             EQU     IAR1.5          ;Bit in the Indirect Addressing Register IAR1
_DI             EQU     IAR1.6          ;EEPROM can only be indirectly addressed using MP1
_DO             EQU     IAR1.7          ;
_EECR           EQU     40H             ;Setup address of the EEPROM control register
C_Addr_Length           EQU 8           ;Address length-8bits
C_Data_Length           EQU 8           ;Data length-always 8bits

DATA    .SECTION  at 70H  'DATA'
EE_command              DB      ?       ;Stores the read or write instruction information
ADDR                    DB      ?       ;Store Write data or read data address
WR_Data                 DB      ?       ;Store read or write data
COUNT                   DB      ?       ;Temporary counter
WriteCommand:                           ;Write instruction code subroutine
        MOV     A,3                     ;Read,Write and Erase instruction are 3bits long
        MOV     COUNT,A
WriteCommand_0:
        CLR     _DI                     ;Prepare the transmitted bit
        SZ      EE_command.7            ;Check value of highest instruction code bit
        SET     _DI
        SET     _SK
        CLR     _SK
        CLR     C
        RLC     EE_command              ;Get next bit of instruction code
        SDZ     COUNT                   ;Check if last bit has been transmitted
        JMP     WriteCommand_0
        CLR     _DI
        RET
```

范例 2-向 EEPROM 写地址

```
WriteAddr:                              ;Write address subroutine
            MOV     A,C_Addr_Length     ;Setup address length
            MOV     COUNT,A
            SET     _SK                 ;Dummy bit transmission for HT46F49E only
            CLR     _SK                 ;Not requied for other devices
WriteAddr_0:
            CLR     _DI
            SZ      ADDR.7              ;Check value of address MSB
            SET     _DI
            CLR     C
            RLC     ADDR                ; GET next address bit
            SET     _SK
            CLR     _SK
            SDZ     COUNT               ;Check if address lsb has been written
            JMP     WriteAddr_0
            CLR     _DI
            RET
```

范例 3-向 EEPROM 写数据

```
WriteData:
            MOV     A,C_Data_Length     ;Setup data length
            MOV     COUNT,A
WriteData_0:
            CLR     _DI
            SZ      WR_Data.7           ;Check value of data MSB
            SET     _DI
            CLR     C
            RLC     WR_Data             ;Get next address bit
            SET     _SK
            CLR     _SK
            SDZ     COUNT               ;Check if data lsb has been written
            JMP     WriteData_0
            CLR     _CS                 ;CS low edge initiates internal write cycle
            SET     _CS                 ;CS high edge allows DO to be used to indicate
                                        ;end of write cycle
            SNZ     _DO                 ;Poll for DO high to indicate end of write cycle
            JMP     $-1
            RET
```

范例 4-从 EEPROM 读数据

```
ReadData:
        MOV     A,C_Data_Length     ;Setup data length
        MOV     COUNT,A
        CLR     WR_Data
ReadData _0:
        CLR     C
        RLC     WR_Data
        SET     _SK
        SZ      _DO                 ;Check value of data MSB
        SET     WR_Data.0
        CLR     _SK
        SDZ     COUNT               ;Check if lsb has been received
        JMP     ReadData _0
        MOV     A,WR_Data
        RET
```

输入/输出端口

盛群单片机的输入/输出端口控制具有很大的灵活性。这体现在每一个引脚在使用者的程序控制下可以被指定为输入或输出、所有引脚的上拉电阻选项、以及指定引脚的唤醒选择，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要求。

依据所选单片机及封装类型的不同，该系列单片机提供从 13 到 23 个不等双向输入/输出口，标示为 PA、PB、PC 和 PD，这些输入/输出端口在数据存储器的对应指定地址如表所示。所有输入/输出口都可作为输入及输出之用。作为输入操作时，输入/输出引脚无锁存功能，输入数据必须在指令“MOV A,[m]”T2 上升沿准备好，m 表示端口地址。对于输出操作，所有数据是锁存的，而且持续到输出锁存被重写。

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。为了免去这个外加的电阻，当引脚设置为输入时，可由内部连接上拉电阻，这些上拉电阻可通过掩膜选项来加以选择，它用一个 PMOS 晶体管来实现。

PA 口唤醒

此系列的单片机都具有暂停功能，使得单片机进入暂停模式以降低功耗，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 某位引脚从高电平转为低电平。当使用暂停指令“HALT”迫使单片机进入暂停状态以后，单片机将保持闲置即低功耗状态，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 PA 口的每个引脚都可单独的选择具有唤醒的功能。

输入/输出端口控制寄存器

每一个输入/输出端口都具有各自的控制寄存器（PAC、PBC、PCC 和 PDC）控制输入/输出状态。利用此控制寄存器，每一个 CMOS 输出或者带或不带上拉电阻的输入，均可利用软件控制方式加以动态的重新设置。所有输入/输出端口的引脚都各自对应于输入/输出端口控制寄存器的某一位。若输入/输出引脚要实现输入功能，则对应的控制寄存器位必须设定为“1”，这时程序指令可以直接读出输入引脚的逻辑状态。如果引脚的控制寄存器位被设定为“0”，则此引脚被设置为 CMOS 输出。当引脚被设置为输出状态，程序指令读取的是输出端口寄存器的内容。请注意当输入/输出端口被设置为输出状态时，此时如果对输出口做读取的动作，则会读取到内部数据寄存器中的锁存值，而不是输出引脚实际的逻辑状态。

引脚共用功能

引脚的共用功能可以增加单片机的灵活程度。有限的引脚个数会严重的限制设计者，但是引脚的复用功能特性，可以很好的解决此类问题。复用功能输入/输出引脚的功能选择，有些是由掩膜选项进行设定，有些则是在应用程序中进行控制。

- 外部中断输入
  外部中断引脚 $\overline{\text{INT}}$ 与输入/输出引脚 PA5 共用。如果不需要外部中断输入，此引脚可当作一般的输入/输出引脚使用；此时，外部中断控制寄存器 INTC 中的外部中断使能位必须除能。

- 外部时钟输入
  外部时钟输入引脚 TMR 与 PA4 共用。如果 PA4/TMR 引脚被设定为计数器输入，则 TMRC 控制寄存器中相应的控制位也必须正确设置。在不需要外部计数器输入的时候，此引脚也可以作为一般 I/O 引脚使用。对于此种应用，TMRC 控制寄存器中的定时器模式位必须选为定时器模式(内部时钟源)，以避免输入/输出引脚与计数器操作的冲突。

- PFD 输出

  此系列每款单片机包含 PFD 功能，PFD 信号输出引脚与 PA3 共用。PFD 输出功能可以通过掩膜选项进行选择，并且在程序烧录后保持不变。需要注意的是，必须将 PAC.3 设为输出以使能 PFD 输出。若 PAC 设置为带上拉电阻的输入，即使选择了 PFD 输出功能，此引脚仍将作为带上拉电阻的一般输入引脚使用。

- PWM 输出

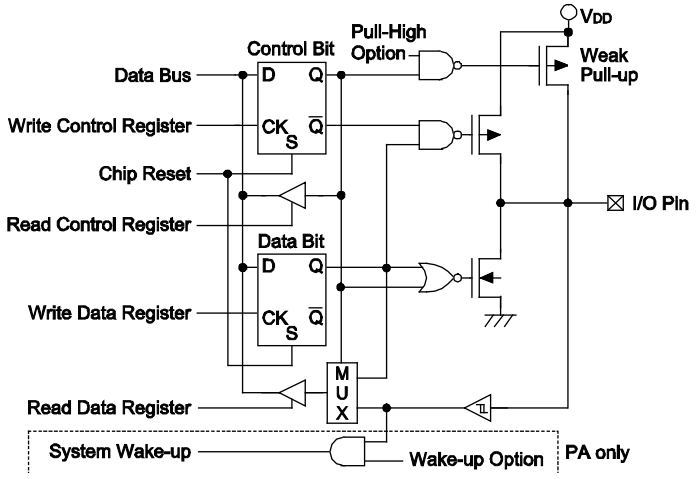
  此系列单片机均提供 1 个或者 2 个 PWM 输出，分别与 PD0 和 PD1 共用引脚。PWM 输出功能可以通过掩膜选项进行选择，并且在程序烧录后保持不变。需要注意的是，必须将 PDC 中相应的位设为输出以使能 PWM 输出。若 PDC 设置为带上拉电阻的输入，即使选择了 PWM 输出功能，此引脚仍将作为带上拉电阻的一般输入引脚使用。

- A/D 输入

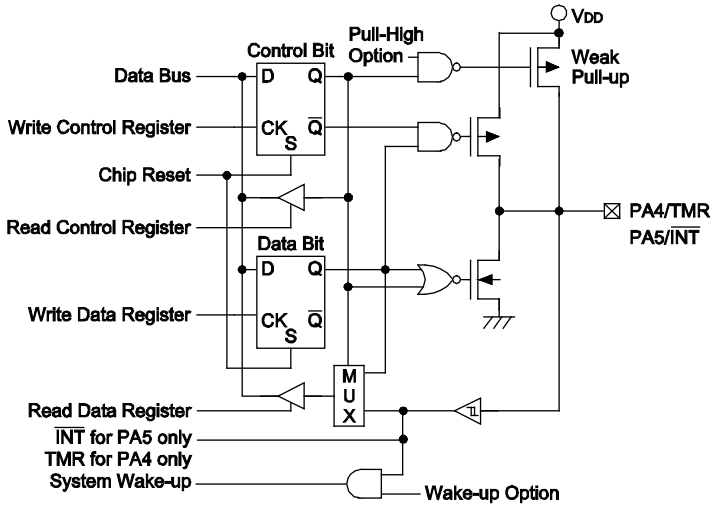
  此系列的单片机都具有 4 个 A/D 转换器输入。所有的模拟输入与 PB 口的 I/O 引脚共用。如果这些引脚被用来作为 A/D 输入而不是一般的 I/O 引脚，则 A/D 转换控制寄存器 ADCR 中相应的位必须被正确的设定。掩膜选项内不包含 A/D 功能。如果这些引脚作为 I/O 引脚使用，仍可以通过掩膜选项选择是否要接上拉电阻。然而如果作为 A/D 输入使用，则这些引脚上的上拉电阻会自动失效。

输入/输出端口结构

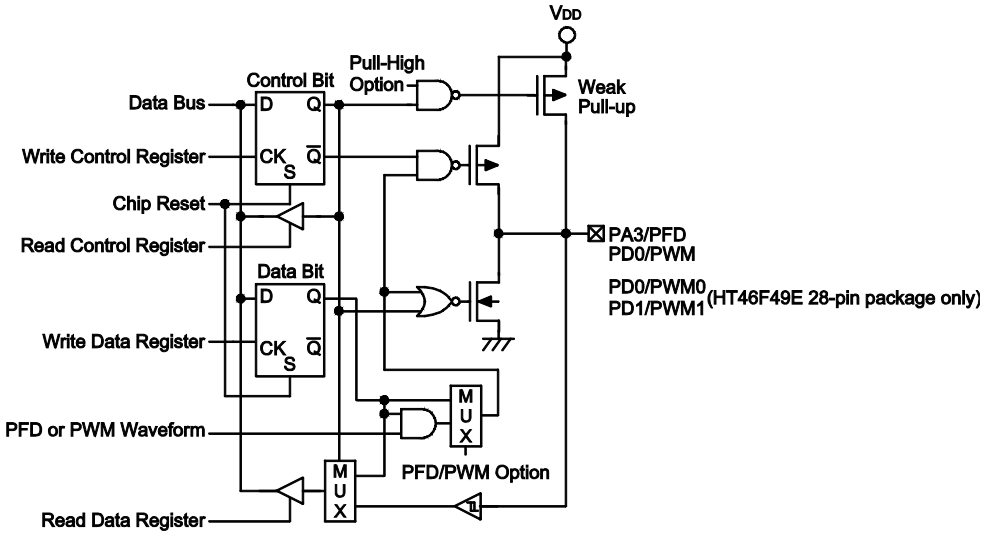
下列为输入/输出端口的内部结构图，可能与芯片内部实际的结构并不完全相同，只是用于帮助用户理解输入/输出端口。



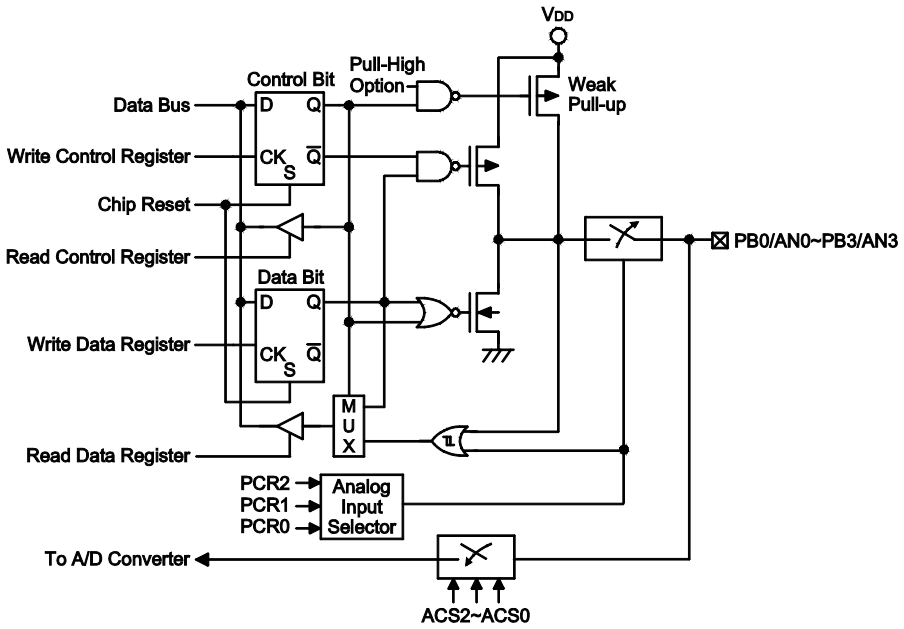
没有引脚共用功能的输入/输出端口



PA4/PA5 输入/输出端口



PA3/PFD 和 PD/PWM 输入/输出端口

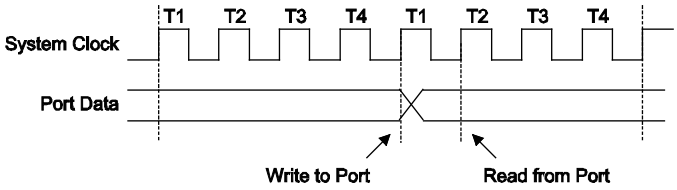


PB 输入/输出端口

编程注意事项

在使用者的程序中，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选项。如果 PAC、PBC、PCC 和 PDC 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 PA、PB、PC 和 PD 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。注意的是当使用这些位控制指令时，一个读-修改-写的操作将

会发生。单片机必须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

PA 口有唤醒的额外功能，当芯片在 HALT 状态时有很多方法去唤醒此单片机，其中之一就是 PA 口任一个引脚电平由高到低的转换，可以设定 PA 口的一个或多个引脚有这项功能。

注意，有些型号芯片具有不同封装类型，导致某些端口没有引出。若这此引脚被设为输入，就会产生振荡并增加系统功耗，特别在 HALT 模式下更明显。建议在使用中将未引出的端口设为输出或设为带上拉的输入。

定时/计数器

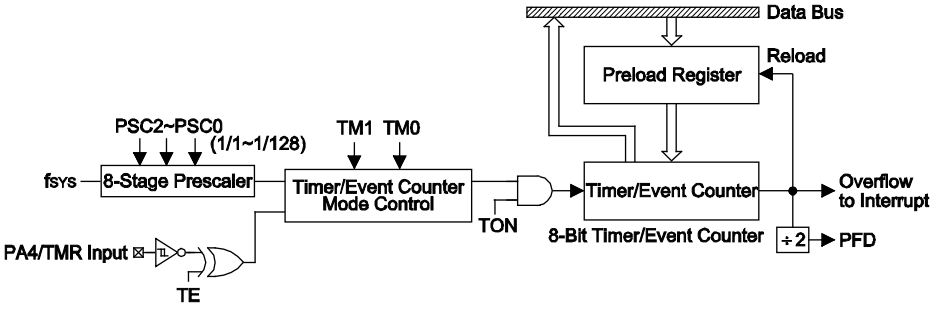
定时/计数器在任何单片机中都是一个很重要的部分，提供程序设计者一种实现和时间有关功能的方法。每款单片机提供一个 8 位的向上定时/计数器。每个定时/计数器有三种不同的工作模式，可以被当作普通定时器、外部事件计数器、或者脉冲宽度测量器使用。定时器里的 8 级预分频器扩大了定时的范围。

有两个和定时/计数器相关的寄存器，分别为 TMR 和 TMRC。TMR 是存储实际的计数值，赋值给此寄存器可以设定初始计数值，读取此寄存器可获得定时/计数器的内容。而 TMRC 是定时/计数器的控制寄存器，此寄存器设置定时/计数器的选项，控制定时/计数器的工作模式。定时/计数器的时钟源可掩膜设置来自内部系统时钟或外部引脚输入（PA4/TMR）。

配置定时/计数器输入时钟源

内部定时/计数器的时钟源可以来自系统时钟或外部时钟源。当定时/计数器在定时器模式或者在脉冲宽度测量模式时，使用系统时钟作为计时来源。内部时钟还可以通过预分频器进行分频，预分频值由 PSC0、PSC1 及 PSC2 三位决定。

定时/计数器在事件计数器模式时使用外部时钟源，时钟源由外部定时/计数器引脚 PA4/TMR 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 TE 位决定）跳变时，计数器值加一。



8 位定时/计数器结构

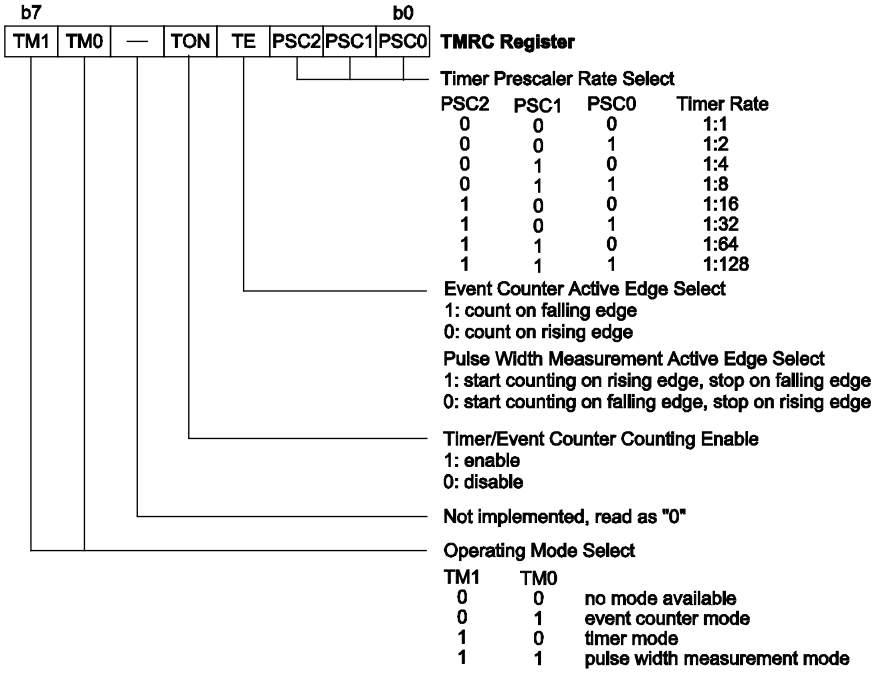
定时/计数寄存器 — TMR

TMR 是一个 8 位特殊功能寄存器，用于储存实际定时器值。在收到一内部计时脉冲时或外部定时/计数器引脚 PA4/TMR 状态发生跳变时，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，直到计满 FFH，此时定时器溢出且会产生一个内部中断信号。定时器自动重新加载计数器初值并继续计数。为了得到定时/计数器 00H 至 FFH 的最大计数范围，预置寄存器必须先清除为 00H。注意的是，上电后预置寄存器处于未知状态。定时/计数器在 OFF 条件下，如果把数据写入预置寄存器，这数据将被立即写入实际的定时器。而如果定时/计数器已经 ON 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，只有在下一个溢出发生时才被写入实际定时器。

定时/计数控制寄存器 — TMRC

定时/计数器能工作在三种不同的模式，至于选择工作在哪一种模式则是由控制寄存器的内容决定。TMRC 和 TMR 寄存器控制定时/计数器的全部操作。在使用定时器之前，必须正确地设定定时/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。

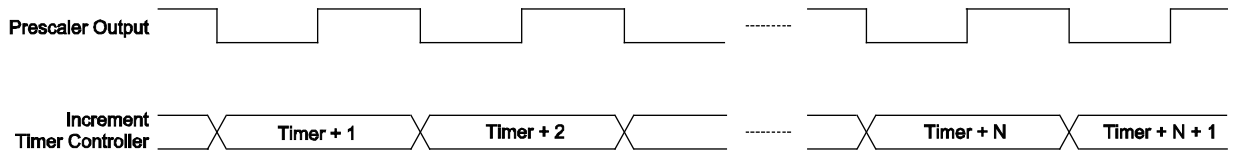
为了确定定时器工作在哪一种模式，定时器模式，外部事件计数模式或脉冲宽度测量模式，TM0 和 TM1 位必须设定到要求的逻辑电平。定时器打开位 TON，即定时/计数控制寄存器的第 4 位，是定时器控制的开关，设定为逻辑高时，计数器开始计数，而清零时则停止计数。定时/计数控制寄存器的第 0~2 位决定输入定时预分频器中的分频系数。如果使用外部计时源，预分频器的位将不起作用。如果定时器工作在事件计数或脉冲宽度测量模式，TE 的逻辑电平，即 TMRC 寄存器的第 3 位将可用来选择上升沿或下降沿触发。



定时/计数控制寄存器

设置定时器模式

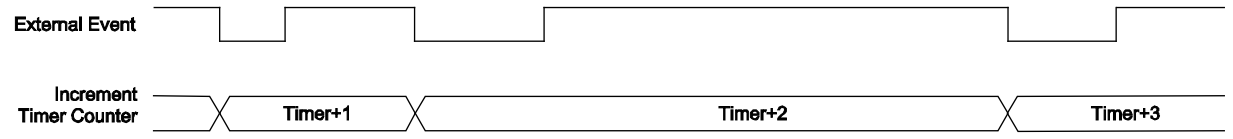
在这个模式下，定时器可以用来测量固定时间间隔，当定时器发生溢出时，就会产生一个内部中断信号。要工作在这个模式，TMRC 寄存器中位 TM1 和 TM0 必须分别设为 1 和 0。在这个模式下，定时器的计数源来自内部时钟。定时器的输入计时频率是 f_{SYS} 除以定时器预分频器的值，这个值由定时器控制寄存器的 PSC2~PSC0 位决定。定时器使能位 TON 必须被设为逻辑高，才能使定时器工作。每次内部时钟由高到低的电平跳变都会使定时器值增加一。当定时器已满即溢出时，会产生中断信号且定时器会重新载入已经载入到预置寄存器的值，然后继续向上计数。若定时器中断允许，将会产生一个中断信号。寄存器 INTC 中 ETI 位清零，则定时器中断禁止。注意的是定时器溢出中断也是唤醒暂停模式的一种方法。



定时器模式时序图

设置事件计数模式

在这个模式下，发生在 PA4/TMR 引脚的外部逻辑事件改变的次数，可以通过内部计数器来记录。为使定时/计数器工作在事件计数器模式，TMRC 寄存器中 TM1 和 TM0 位必须分别设为 0 和 1。计数器打开位 TON 必须设为逻辑高，使计数器开始计数。当 TE 为逻辑低时，每次外部定时/计数器引脚 PA4/TMR 接收到由低到高的电平跳变将使计数器值加一。而当 TE 为逻辑高时，每次外部定时/计数器引脚接收到由高到低的电平跳变将使计数器值加一。与另外两个模式一样，当计数器计满时，计数器将溢出且自动从预置寄存器中加载初值。若定时中断允许，将会产生一个中断信号。定时器中断可以通过清除 INTC 寄存器中 ETI 位禁止。为了确保定时/计数器工作在事件计数器模式，必须注意两点，首先是要将 TM0 和 TM1 位设定在事件计数器模式，其次是确定端口控制寄存器将这个引脚设定为输入状态。计数器的溢出是唤醒暂停模式的一种方法。在事件计数模式下，即使系统进入 HALT 状态，定时/计数器仍可记录外部引脚的变化。因此当定时/计数器溢出将会唤醒系统，若中断允许将会产生一个定时器中断信号。

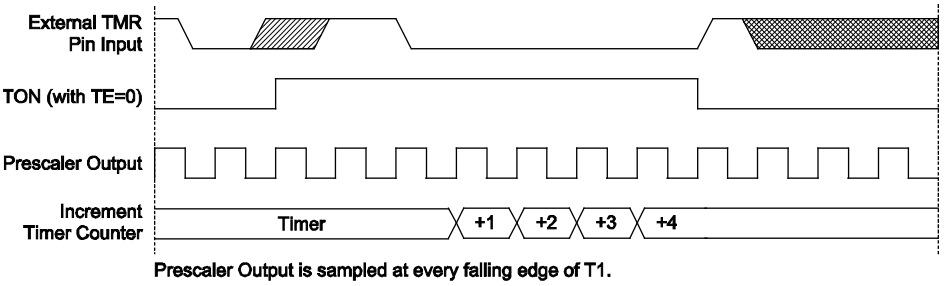


计数器模式时序图

设置脉冲宽度测量模式

在这个模式下，可以测量外部 PA4/TMR 引脚上的外部脉冲宽度。在脉冲宽度测量模式中，定时/计数器的时钟源由内部时钟提供，TM0 和 TM1 位则必须都设为逻辑高。如果 TE 位是逻辑低，当外部 PA4/TMR 引脚接收到一个由高到低的电平跳变时，定时/计数器将开始计数直到外部 PA4/TMR 引脚回到它原来的高电平。此时 TON 位将自动清除为零，且定时/计数器停止计数。而如果 TE 位是逻辑高，则当外部 PA4/TMR 引脚接收到一个由低到高的电平跳变时，定时/计数器开始计数直到外部 PA4/TMR 引脚回到原来的低电平。如上所述，TON 位将自动清除为 0，且定时/计数器停止计数。注意，在脉冲宽度测量模式中，当外部定时器引脚上的外部控制信号回到它原来的电平时，TON 位将自动地清除为 0。而在其它两种模式下，TON 位只能在程序控制下才会被清除为 0。这时定时/计数器中的值可被程序读取，并由此得知外部定时/计数器引脚接收到的脉冲的长度。当 TON 位被清除时，任何在外部定时/计数器引脚的电平跳变将被忽略。直到 TON 位再次被程序设定为逻辑高，定时/计数器才开始脉冲宽度测量。用这种方法可完成单个脉冲的测量，注意的是在这种模式下，定时/计数器是通过外部定时/计数器引脚上的逻辑跳变来控制，而不是通过逻辑电平。

与另外两个模式一样，当定时/计数器计满产生溢出，定时/计数器将清零并载入预置寄存器的值。若定时器中断允许，将会产生一个内部中断信号。为了确保 PA4/TMR 工作在脉冲宽度测量模式，必须注意两点，首先是要将 TM0 与 TM1 位设定在脉冲宽度测量模式，其次是确定此引脚的输入/输出端口控制寄存器对应位被设定为输入状态。定时器的溢出中断也是唤醒暂停模式的一种。



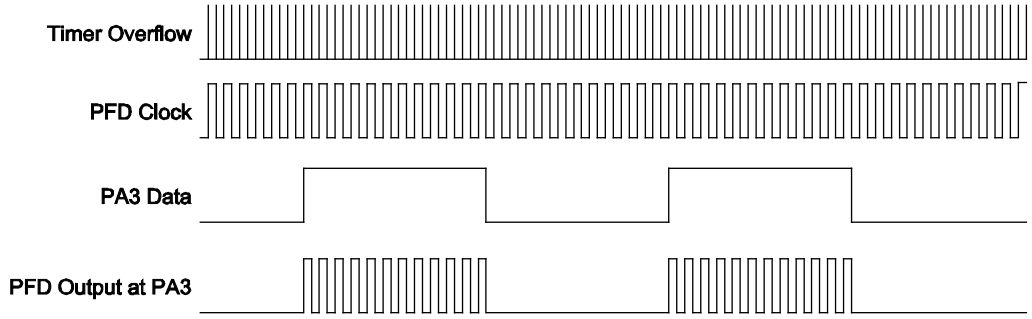
脉冲宽度测量模式时序图

可编程分频器 – PFD

PFD 输出引脚与 PA3 引脚共用。此功能通过掩膜选项选择，如果不选择该功能，则这个引脚作为普通的输入/输出引脚使用。PFD 电路使用定时器溢出信号作为它的时钟源。载入合适的值到定时器预分频器，可以产生需要时钟源的分频系数，由此来控制输出的频率。系统时钟被预分频器分频后的时钟源，进入定时器计时，定时器从预置寄存器的值开始往上计数，直到计数值满而产生溢出信号，导致 PFD 输出改变状态。定时器将自动地重新载入预置寄存器的值，并继续向上计数。

要使 PFD 正确运作，必须将 PA 控制寄存器 PAC 的第 3 位设置为输出。如果把它设置为输入，则 PFD 输出不工作，该引脚仍是作为普通的输入引脚使用。只有把 PA3 位置“1”，PFD 输出引脚才会有输出。这个输出数据位被用作 PFD 输出的开/关控制。注意，如果 PA3 输出数据位被清为“0”，PFD 输出将为低电平。

假如系统时钟使用晶体振荡器，则使用这种频率产生的方法可以产生非常精确的频率值。



PFD 输出控制

预分频器

TMRC 控制寄存器的 PSC0~PSC2 可以用来定义定时/计数器中内部时钟源的预分频级数。定时/计数器溢出信号可用来驱动 PFD 或产生定时器中断。

输入/输出接口

当工作在事件计数或脉冲宽度测量模式时，定时/计数器需要使用外部 PA4/TMR 引脚以确保正确的动作。由于此端口为共用引脚，必须将其设置为定时/计数器的外部输入而不是普通的输入/输出，这需要设置 TMRC 寄存器内的相应的位为事件计数或脉宽测量模式。另外端口控制寄存器 PAC 的第 4 位必须为高，以保证被设为输入引脚。即使定时/计数器使用了此引脚，掩膜选项的上拉仍有效。

编程注意事项

当定时/计数器工作在定时器模式时，定时器的时钟源使用内部系统时钟且与单片机所有运算都能同步。在这个模式下，当定时器寄存器溢出时，单片机将产生一个内部中断信号，使程序进入相应的内部中断向量。对于脉冲宽度测量模式，定时器的时钟源也是使用内部系统时钟，但定时器只有在正确的逻辑条件出现在外部 PA4/TMR 引脚时开始工作。由于外部事件和内部定时器时钟无法同步，只有当下一个定时器时钟到达时，单片机才会发现这个外部事件，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/计数器配置为外部事件计数模式时，它的时钟源是外部事件，与内部系统时钟或者定时器时钟不同步。

当读取定时/计数器或写数据到预置寄存器时，计数时钟会被阻隔以避免错误发生，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必须正确的设置，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必须正确的设定，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，必须确保先载入定时/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。在打开定时器之前，必须先确定定时器模式是否已设定。通过写定时/计数控制寄存器，即同时改变模式并打开定时器，可能会导致错误结果。

当发生定时/计数器溢出，相应的中断请求标志将置位。若中断允许，将会产生一个中断信号。不管中断是否允许，在 HALT 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 HALT 模式下，不需要定时器中断唤醒系统，可以在进入 HALT 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/计数为定时模式，时钟来源于内部的系统时钟。

```
org 04h                 ; external interrupt vector
reti
org 08h                 ; timer/event counter interrupt vector
jmp tmrint               ;jmp here when timer overflows
      :
org 20h                 ; main program
                        ; internal timer/event counter interrupt routine
tmrint:
      :
                        ;timer/event counter main program placed here
      :
reti
      :
      :
begin:
                        ; setup timer registers
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,005h              ; enable master interrupt and timer interrupts
mov intc,a
set tmrc.4              ; start timer – note mode bits must be previously setup
```

脉冲宽度调制器

此系列每款单片机都提供 1 个或 2 个脉冲宽度调制(PWM)输出。这在马达速度控制等应用中十分有用，通过给相应的 PWM 寄存器设置特定的值，PWM 功能可提供占空比可调而频率固定的波形。

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。对于只有一个 PWM 输出的单片机，这个寄存器为 PWM。对于有两个 PWM 输出的单片机，寄存器则为 PWM0 和 PWM1。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，每一个调制周期被调制两个或四个独立的调制子区段，即分别是 7＋1 或 6+2 模式。每个单片机可以通过设置适当的掩膜选项来选择使用哪种模式。当选择一种掩膜选项模式后，此模式将应用于此芯片的所有 PWM 输出。注意的是，当使用 PWM 时，只要将所需的值写入相应的 PWM 寄存器内，单片机的内部硬件会自动地将波形细分为子调制周期。

对所有的单片机而言，PWM 时钟源就是系统时钟 f_{SYS}。

单片机型号	通道	PWM 模式	输出引脚	PWM 寄存器名称
HT46F49E	2	6+2 或 7＋1	PD0/PD1	PWM0/PWM1
其它芯片	1	6+2 或 7＋1	PD0	PWM

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。只要产生的 PWM 脉冲周期小于负载的时间常数，PWM 输出就比较合适，这是因为长时间常数负载将会平均 PWM 输出的脉冲。使用者必须理解 PWM 频率与 PWM 调制频率的不同之处。当 PWM 时钟为系统时钟 f_{SYS}，而 PWM 值为 8 位时，整个 PWM 周期的频率为 $f_{SYS}/256$。然而工作在 7+1 模式时，PWM 调制频率将会是 $f_{SYS}/128$，而工作在 6+2 模式时，PWM 调制频率将会是 $f_{SYS}/64$。

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

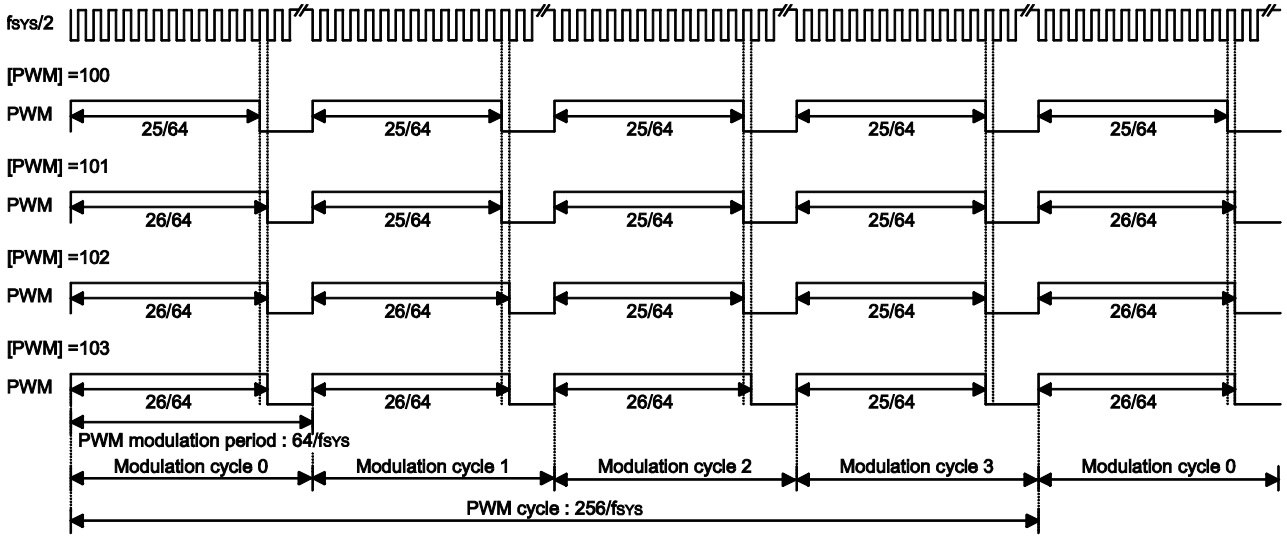
6+2 PWM 模式

通过一个 8 位的 PWM，PWM0 或 PWM1 寄存器控制，每个完整的 PWM 周期由 256 个时钟周期组成。在 6+2 PWM 模式中，每个 PWM 周期又被分成四个独立的子周期，称为调制周期 0~调制周期 3，在表格中以 “i” 表示。四个子周期各包含 64 个时钟周期。在这个模式下，得到以 4 为因数增加的调制频率。8 位的 PWM 寄存器被分成两个部分，这个寄存器的值表明整个 PWM 波形的占空比。第一部分包括第 2 位~第 7 位，表示 DC 值，第二部分为第 0 位~第 1 位，表示 AC 值。在 6+2 PWM 模式中，四个调制子周期的占空比，分别如下表所示。

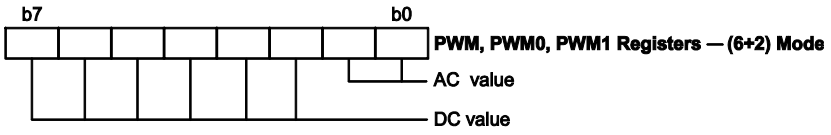
参数	AC (0~3)	DC (占空比)
调制周期 i (i=0~3)	i<AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$

6+2 模式调制周期值

下图表示 6+2 模式下 PWM 输出的波形。请特别注意单个的 PWM 周期是如何分成四个独立的调制周期以及 AC 值与 PWM 值的关系。



6+2 PWM 模式



6+2 模式 PWM 寄存器

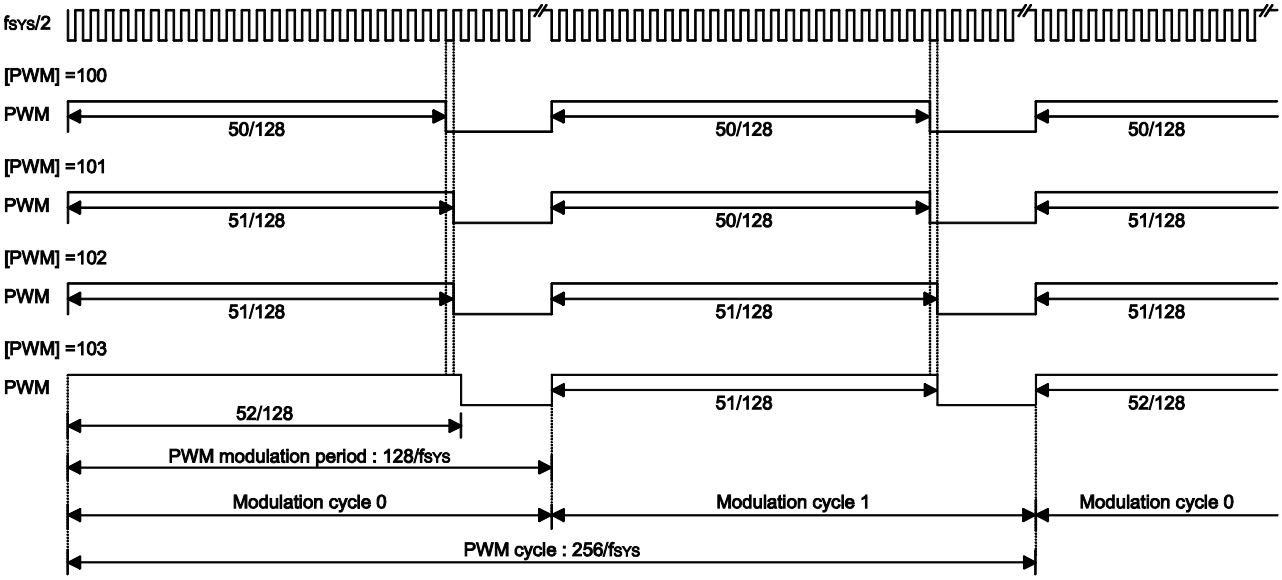
7+1 模式

通过一个 8 位的 PWM，PWM0 或 PWM1 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。

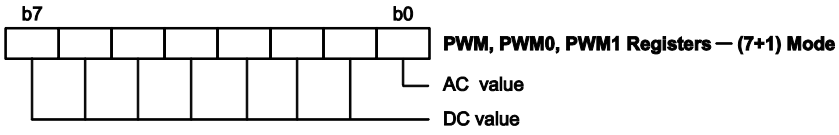
参数	AC (0~1)	DC (占空比)
调制周期 i (i=0~1)	i<AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

7+1 模式调制周期值

下图表示 7+1 模式下 PWM 输出的波形。请特别注意单个的 PWM 周期是如何分成两个独立的调制周期以及 AC 值与 PWM 值的关系。



7+1 PWM 模式



7+1 模式 PWM 寄存器

PWM 输出控制

此系列所有单片机，PWM 输出与 PD0 或 PD1 端口引脚共用。要使某个引脚作为 PWM 输出而非普通的 I/O 引脚，必须选择正确的 PWM 掩膜选项。I/O 端口控制寄存器 PDC 中相应的位也必须写“0”，以确保所需要的 PWM 输出引脚设置为输出状态。在完成这两个初始化步骤，以及将所要求的 PWM 值写入 PWM 寄存器之后，将“1”写入到 PD 输出数据寄存器的相应位，则 PWM 数据就会出现在引脚上。将“0”写入到 PD 输出数据寄存器的相应位，则会除能 PWM 输出功能并强制输出低电平。通过这种方式，PD 数据输出寄存器作为 PWM 功能的开/关控制来使用。假如掩膜选项已经选择 PWM 功能，但是在 PDC 控制寄存器中相应的位又写入“1”，使其成为输入引脚，则此引脚仍是作为带上拉电阻的正常输入端使用。

PWM 应用范例

下面的范例程序表明如何设置和控制 PWM 输出。在使用相应 PWM 输出前需要在掩膜选项中使能 PWM 输出。

```
clr     pdc.0           ;set pin PD0 as output
clr     pdc.1           ;set pin PD1 as output
set     pd.0            ;pd.0=1;enable pin “PD0/PWM0” to be the PWM channel 0
mov     a,64h           ; PWM0=100D=64H
```

```
mov     pwm0,a

set     pd.1            ;pd.1=1;enable pin "PD1/PWM1" to be the PWM channel 1
mov     a,65h           ; PWM1=101D=65H
mov     pwm1,a

clr     pd.0            ; disable the PWM0 output – PD.0 will remain low
clr     pd.1            ; disable the PWM1 output – PD.1 will remain low
```

A/D 转换器

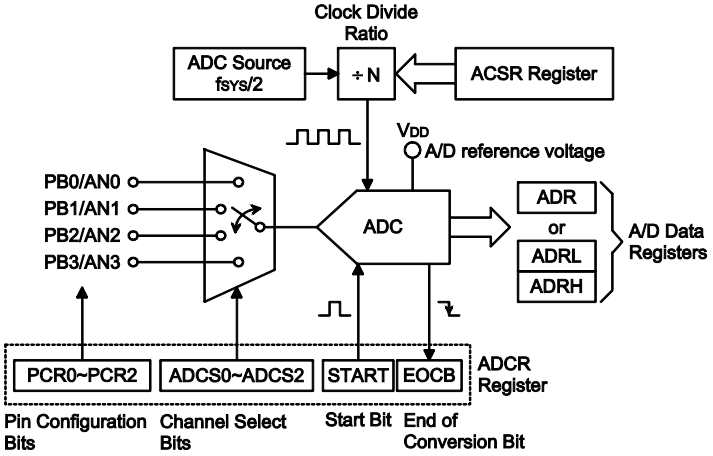
对于大多数电子系统而言，处理现实世界中模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先必须通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 简介

此系列每款单片机都包含了四个通道的 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 8 位或 9 位的数字量。

单片机型号	输入通道	转换位数	输入引脚
HT46F46E	4	8	PB0~PB3
HT46F47E	4	9	PB0~PB3
HT46F48E	4	9	PB0~PB3
HT46F49E	4	9	PB0~PB3

下图显示了 A/D 转换器整个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 — ADR，ADRL，ADRH

对于具有 8 位 A/D 转换器的 HT46F46E，使用寄存器 ADR 保存模数转换值；对于其它具有 9 位 A/D 转换器的芯片，则需要两个寄存器，一个高字节寄存器 ADRH 和一个低字节寄存器 ADRL。在 A/D 转换完毕后，单片机可以直接读取这些寄存器以获得转换结果。对于具有 2 个 A/D 转换结果寄存器的单片机，要注意的是，只有高位寄存器 ADRH 完全利用了 8 位。而低位寄存器 ADRL 只利用了 8 位中的 1 位，它包含的只是 9 位转换值中低的 1 位。

在下表中，D0~D8 是 A/D 换转数据结果位。

寄存器	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADR	D7	D6	D5	D4	D3	D2	D1	D0

A/D 数据寄存器 — HT46F46E

寄存器	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRL	D0	—	—	—	—	—	—	—
ADRH	D8	D7	D6	D5	D4	D3	D2	D1

A/D 数据寄存器 — 其它型号

A/D 转换控制寄存器 – ADCR

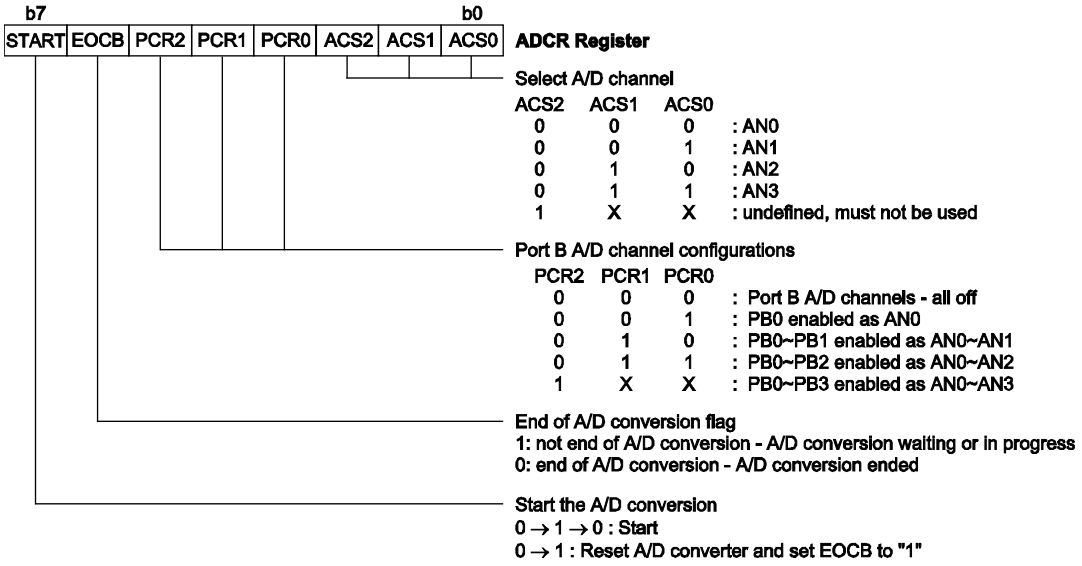
寄存器 ADCR 用来控制 A/D 转换器的功能和操作。这个 8 位的寄存器所定义的功能包括选择哪一个模拟通道连接至内部 A/D 转换器，哪个引脚是模拟输入，哪个引脚是正常 I/O，并控制和监视 A/D 转换器的开始和复位功能。

寄存器 ADCR 包含 ACS2~ACS0 位，它们定义通道的编号。由于每个单片机只包含一个实际的模数转换电路，因此这 4 个模拟输入中的每一个都必须分别被发送到转换器。ADCR 寄存器中 ACS2~ACS0 位的功能正是决定哪个模拟通道真正连接到内部 A/D 转换器。需要注意的是， ACS2 位必须保持为“0”。

ADCR 寄存器中的 PCR2~PCR0 位，用来定义 PB 端口上哪些引脚为 A/D 转换器的模拟输入，哪些引脚为正常的 I/O。如果 PCR2~PCR0 这 3 位地址的值等于或大于“100”，也就是 AN0、AN1、AN2 和 AN3 都将被设定为模拟输入。注意的是，如果 PCR2~PCR0 全都设为“0”，则所有 PB 端口的引脚都被设定为正常的 I/O，这时内部 A/D 转换器电路的电源将被关闭以减少功耗。

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，就会产生适当的内部中断信号。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，单片机会轮询 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。

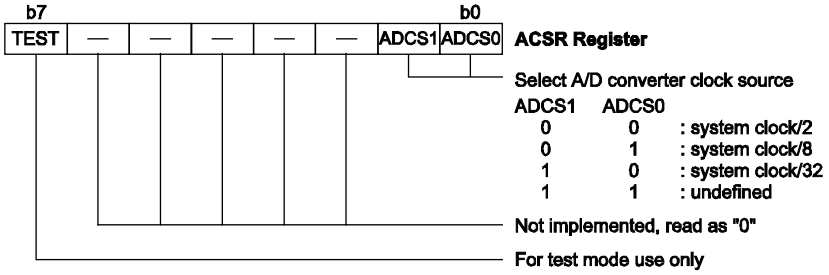


A/D 转换控制寄存器

A/D 转换时钟源寄存器 – ACSR

A/D 转换器时钟源为系统时钟 f_{SYS} 的分频，分频系数由 ACSR 寄存器中的 ADCS1 和 ADCS0 位决定。

虽然 A/D 时钟源是由系统时钟 f_{SYS}、ADCS1 和 ADCS0 决定，但可选择的最大 A/D 时钟源速度则有一些限制。对于 HT46F46E 允许的 A/D 时钟周期 t_{AD} 的最小值为 0.5μs，因此，当系统时钟速度超过 4MHz 时，ADCS1 和 ADCS0 位不能设为“00”。而对于其它芯片则为 1μs，当系统时钟速度超过 2MHz 时，ADCS1 和 ADCS0 位不能设为“00”，否则将会产生不准确的 A/D 转换值。参考下面表格中的一些例子，被标上星号*的数值是不允许的，因为它们的 A/D 时钟周期小于规定的最小值。



A/D 转换时钟控制寄存器

f_{SYS}	A/D 时钟周期(t_{AD})			
	ADCS1, ADCS0=00 (f_{SYS}/2)	ADCS1, ADCS0=01 (f_{SYS}/8)	ADCS1, ADCS0=10 (f_{SYS}/32)	ADCS1, ADCS0=11
1MHz	2μs	8μs	32μs	未定义
2MHz	1μs	4μs	16μs	未定义
4MHz	500ns*	2μs	8μs	未定义
8MHz	250ns*	1μs	4μs	未定义
12MHz	166.67ns*	0.67μs	2.67μs	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 PB 端口的 I/O 引脚共用。ADCR 寄存器中的 PCR2~PCR0 位，决定是将输入引脚设置为普通的 PB 输入/输出引脚，还是将它们设置为模拟输入引脚，而不是由掩膜选项来决定。通过这种方式，引脚的功能可由程序来控制，从普通 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 I/O 引脚使用时，可通过掩膜选项设置上拉电阻，若设置为 A/D 输入，则上拉电阻会自动断开。请注意，PB 端口控制寄存器并不需要为使能 A/D 输入，而先设定 A/D 引脚为输入引脚，当 PCR2~PCR0 位使能 A/D 输入时，不考虑端口控制寄存器的状态。电源脚 VDD 为 A/D 转换器的参考电压，模拟输入电压不可超过此电压值。另外须适当的量测 VDD，以确保该电压的稳定及减少噪声。

A/D 转换初始化

内部的 AD 转换器必须经过一个特殊的方法初始化。当 PB 端口转换通道选择位被改变，AD 转换器必须重新初始化。如果通道选择位改变后没有重新初始化，那么 EOCB 标志位可能会处于不确定的状态，这样可能会导致一个错误的转换结束信号。当通道选择位改变后，寄存器 ADCR 中的 START 位必须在 1~10 个指令周期内先置位再立即清零。这样可以保证 EOCB 被正确的置位。

A/D 转换步骤

下面总结实现 A/D 转换过程的各个步骤。

- 步骤 1

  通过 ACSR 寄存器中的 ADCS1 和 ADCS0 位，选择所需的 A/D 转换时钟。

- 步骤 2

  通过 ADCR 寄存器中的 ACS2~ACS0 位，选择连接至内部 A/D 转换器的通道。

- 步骤 3

  通过 ADCR 寄存器中的 PCR2~PCR0 位，选择 PB 端口的 A/D 输入引脚，并将它们设置为 A/D 输入引脚。此步骤也可在第二步写 ADCR 寄存器时完成。

- 步骤 4

  如果要使用中断，则中断控制寄存器必须正确地设置，以确保 A/D 功能的动作。中断控制寄存器 INTC 里总中断控制位 EMI 必须置位为“1”，A/D 转换器的中断使能位 EADI 也必须置位为“1”。

- 步骤 5

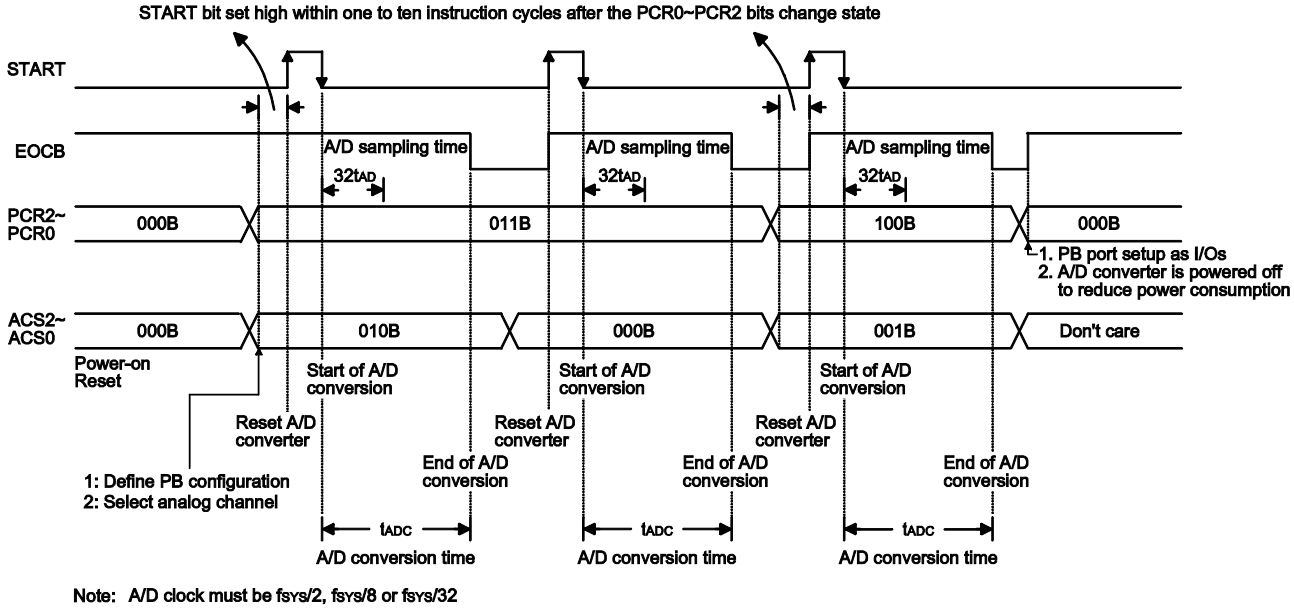
  通过设定 ADCR 寄存器中的 START 位从“0”到“1”再回到“0”，可以开始模数转换的过程。该位需初始化为“0”。

- 步骤 6

  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，则转换完成后，程序会进入 A/D 中断服务子程序。

注意：若使用轮询 ADCR 寄存器中 EOCB 位的状态的方法来检查转换过程是否结束时，步骤 4 可以省略。

下列时序图表示模数转换过程中不同阶段的图形与时序。



A/D 转换时序图

A/D 转换器没有对应的掩膜选项,它的功能设定与操作完全由应用程序控制。由应用程序控制开始 A/D 转换过程后，单片机的内部硬件就会开始进行转换，在这个过程中，程序可以继续其它功能。A/D 转换时间取决于使用的单片机型号，下表为 A/D 转换所需要的周期数。

型号	A/D 转换时间
HT46F46E	$64t_{AD}$
其它	$76t_{AD}$

A/D 转换时间

编程注意事项

在编写程序时，必须特别注意寄存器 ADCR 中的 AD 转换通道选择位。如果这些全部为零，那么没有外部引脚连接到 AD 转换器上，此时的外部引脚可作为普通的 IO 口使用。这样可以减少 AD 转换部分的功耗，也降低了整个芯片的工作电流。关闭 AD 转换器以降低功耗，这一点对电池供电的系统非常重要。

另一个编程注意事项是，当 AD 转换通道选择位改变后，AD 转换器必须重新初始化。当通道选择位改变后，给寄存器 ADCR 的 START 位一个脉冲即可初始化 AD 转换器。当选择位全部被清零，由于不需要进行 AD 转换，可以不重新初始化 AD 转换器。

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 EOCB 轮询方法侦测转换的结束，此范例适用于 HT46F46E

```
        clr EADI                ; disable ADC interrupt
        mov a,00000001B
        mov ACSR,a              ; setup the ACSR register to select fSYS/8 as the A/D clock
        mov a,00100000B         ; setup ADCR register to configure Port PB0~PB3
                                ; as A/D inputs and select AN0 to be
        mov ADCR,a              ; connected to the A/D converter
            :                   ; As the Port B channel bits have changed the  following
                                ; START signal (0-1-0) must be issued within 10 instruction cycles
            :
Start_conversion:
        clr START
        set START               ; reset the A/D
        clr START               ; start the 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,ADR               ; read conversion result high byte value from  ADR register
        mov adr_buffer,a        ; save result to user defined memory
            :
            :
        jmp Start_conversion    ; start next A/D conversion
```

范例：使用中断方式侦测转换的结束，此范例适用于 HT46F46E

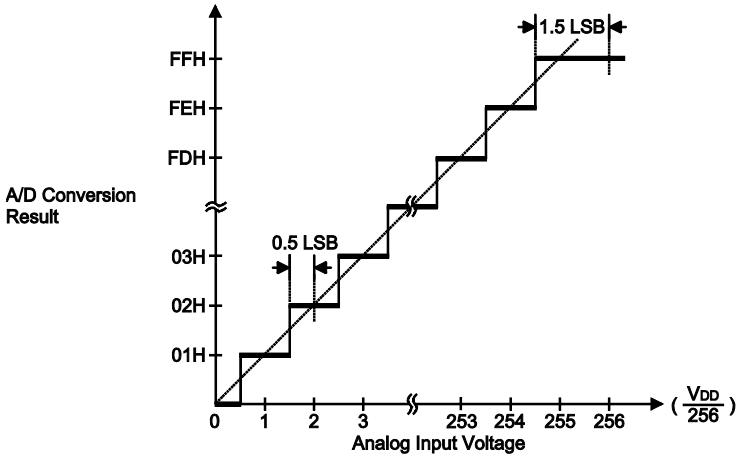
```
        clr EADI                ; disable ADC interrupt
        mov a,00000001B
        mov ACSR,a              ; setup the ACSR register to select fSYS/8  as the A/D clock
        mov a,00100000B         ; setup ADCR register to configure Port PB0~PB3
                                ; as A/D inputs and select input AN0 to be
        mov ADCR,a              ; connected to the A/D converter
            :                   ; As the Port B channel bits have changed the  following
                                ; START signal (0-1-0) must be issued within 10 instruction cycles
            :
Start_conversion:
        clr START
        set START               ; reset the A/D
        clr START               ; start the A/D
        clr ADF                 ; clear ADC interrupt request flag
```

```
        set EADI                ; enable ADC interrupt
        set EMI                 ; enable global interrupt
            :
            :
ADC_ISR:                        ; ADC interrupt service routine
        mov acc_stack,a         ; save ACC to user defined memory
        mov a,STATUS
        mov status_stack,a      ; save STATUSto user defined memory
            :
            :
        mov a,ADR               ; read conversion result value from the ADR register
        mov adr_buffer,a        ; save result to user defined register
            :
            :
EXIT_INT_ISR:
        mov a,status_stack
        mov STATUS,a            ;restore STATUS from user difined memory
        mov a,acc_stack         ;restore ACC from user difined memory
        reti
```

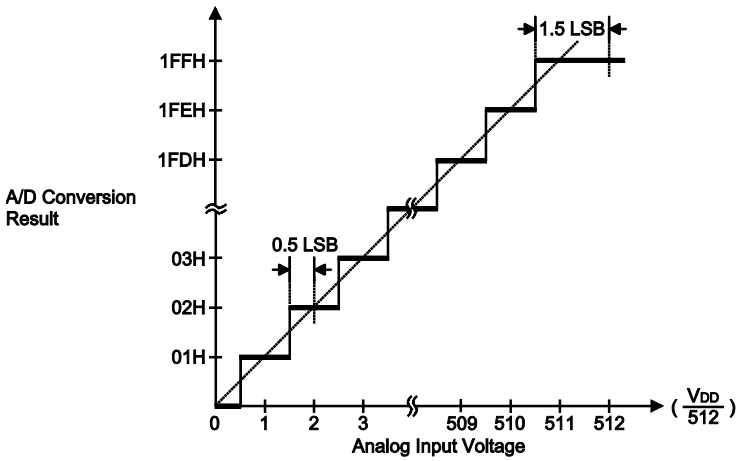
A/D 转换功能

HT46F46E 单片机含有一组 8 位的 A/D 转换器，它们转换的最大值可达 0FFH。由于模拟输入最大值等于 VDD 的电压值，因此每一位可表示 $V_{DD}/256$ 的模拟输入值。而对于其它型号单片机均含有一组 9 位的 A/D 转换器，它们转换的最大值可达 1FFH，每一位表示 $V_{DD}/512$ 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。



理想的 A/D 转换功能 – HT46F46E



理想的 A/D 转换功能 – 其它型号

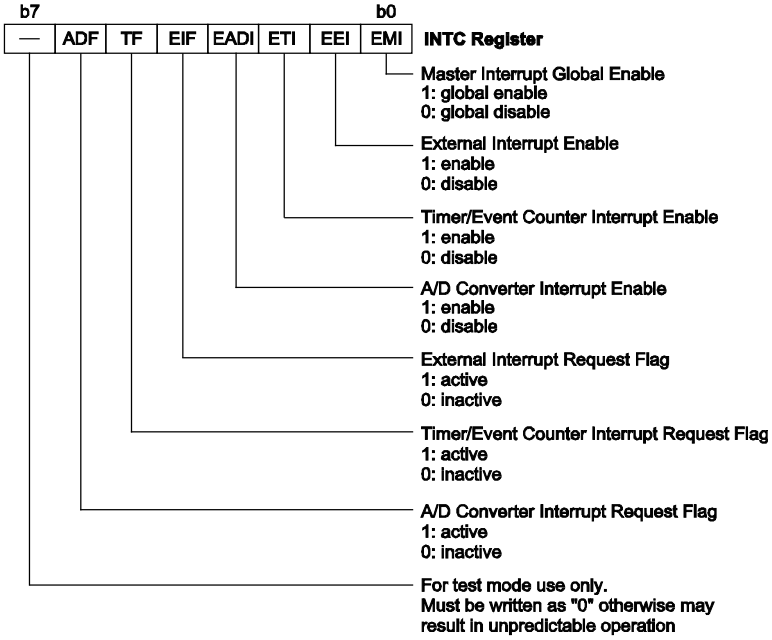
注意的是，为了减少量化错误，A/D 转换器输入端会加入 0.5LSB 的偏移量。除了数字化数值 0，其后的数字化数值会在精确点之前的 0.5LSB 处改变，而数字化数值的最大值将在 VDD 之前的 1.5 LSB 处改变。

中断

中断是单片机一个重要功能。当外部中断或内部中断如定时/计数器和 A/D 转换有效，系统会中止当前的程序，而转到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两个内部中断，外部中断与 $\overline{\text{INT}}$ 引脚相关，而内部中断与定时/计数器和 A/D 转换相关。

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允许位使能或禁止相关的中断。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禁止所有中断允许。

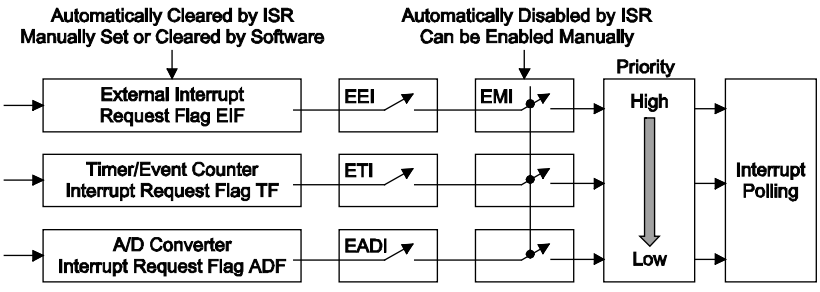


中断控制寄存器

中断操作

如果相应的中断允许，定时/计数器溢出、A/D 转换结束或外部中断引脚下降沿都会产生一个中断请求，系统将下条指令的地址压入堆栈，相应的中断向量地址加载至 PC 中，并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，将堆栈中先前的 PC 值恢复，以继续执行中断发生处的主程序。

各个中断使能位以及相应的请求标志位，以优先级的顺序如下图所示。



中断结构图

一旦中断子程序被响应，所有其它的中断将被屏蔽（系统自动清除 EMI 位）。这个方式可以防止任何进一步的中断嵌套。其它的中断请求可能发生在此期间，但只有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，此时有另一个中断要求响应，EMI 位和 INTC 相关的位可以被置位，以允许此中断被响应。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。

中断优先级

中断发生在两个连续的 T2 脉冲上升沿之间，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所提供的优先级。这个可以通过重新设定 EMI 位来加以屏蔽。

中断源	优先级
外部中断	1
定时/计数溢出中断	2
A/D 转换中断	3

假使外部中断和内部中断均被使能，且同时发生，则外部中断永远优先处理，首先被响应。使用 INTC 寄存器适当地屏蔽个别中断，可以防止中断同时发生的情况。

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 EEI 必须先被置位。外部中断通过 $\overline{\text{INT}}$ 端口由高到低的电平跳变来触发，相应的中断请求标志位 EIF 被设定。外部中断与 PA5 共用引脚， INTC 中相应允许位被置位，此引脚将被作为外部中断使用，此时必须通过 PAC.5 将其设为输入。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 EIF，EMI 位会被清零以除能其它中断。外部中断使能时此引脚的上拉电阻选项仍然有效。

定时/计数器中断

要使定时/计数器中断发生，总中断控制位 EMI、定时/计数器中断使能位 ETI 必须先置位。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，将调用位于地址 08H 处的子程序。当响应定时/计数器中断服务子程序时，中断请求标志位 TF，EMI 位会被清零以除能其它中断。

A/D 转换中断

要使 A/D 转换中断发生，总中断控制位 EMI、A/D 转换中断使能位 EADI 必须先被置位。当 A/D 转换结束，相应的中断请求标志位 ADF 将置位并触发 A/D 转换中断。若中断使能，堆栈未满，当发生 A/D 转换中断时，将调用位于地址 0CH 处的子程序。当响应 AD 转换中断服务子程序时，中断请求标志位 ADF，EMI 位会被清零以禁止其它中断。

编程注意事项

通过除能中断使能位，可以屏蔽中断请求，然而，一旦中断请求标志位被设定，它们会被保留在 INTC 寄存器内，直到相应的中断服务子程序执行或被软件指令清除。

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。

在暂停模式下所有中断都具有唤醒功能。

当进入中断服务程序，系统只会将程序计数器的内容压入堆栈。如果有破坏主程序流程的寄存器或状态寄存器被中断服务程序改变，应事先将这些数据保存起来。

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。最重要的是，复位条件在初次提供电源给单片机后，经短暂延迟，内部电路将使得单片机被定义在良好的状态且准备执行第一条程序语句。上电复位之后，在程序未开始执行前，部分重要的内部寄存器将会被预先设定状态。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。

除了上电复位外，即使单片机正在执行状态，有些情况的发生也迫使单片机必须加以复位。其中一个例子是当提供电源给单片机以执行程序后，$\overline{\text{RES}}$ 引脚被强制拉至低电平。这个例子为正常操作复位，单片机中只有一些寄存器受影响，而大部分寄存器不受影响，以便复位引脚回复至高电平后，单片机仍可以正常操作。复位的另一种形式是看门狗定时器溢出而复位单片机，所有复位操作类型导致不同的寄存器条件被加以设定。

另外一种复位以低电压复位，即 LVR 的型态存在，在电源提供电压低于某一临界值的情况下，一种和 $\overline{\text{RES}}$ 引脚复位类似的完全复位将会被执行。

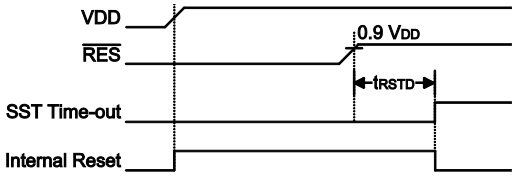
复位

通过内部与外部事件触发复位，单片机共有五种复位方式：

- 上电复位

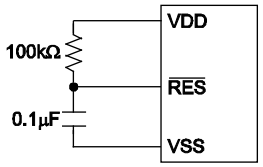
这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器会从起始地址开始执行，上电复位也使得其它寄存器被设定在预设条件，所有的输入/输出端口寄存器和输入/输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设为输入状态。

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



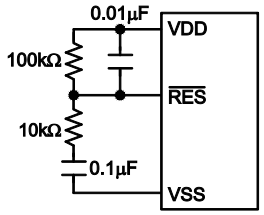
上电复位时序图

在许多应用场合，可以在 VDD 与 $\overline{\text{RES}}$ 之间连接电阻，而在 $\overline{\text{RES}}$ 与 VSS 之间连接电容作为复位电路，为了减少干扰影响，至 $\overline{\text{RES}}$ 引脚的连线应尽量短。



基本复位电路

当系统在较强干扰的场合工作时，强烈建议使用增强型复位电路。

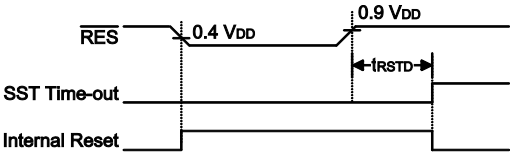


增强型复位电路

欲知更多外部复位电路的相关信息可参考 HOLTEK 网站应用范例 HA0075S。

- **$\overline{\text{RES}}$** 引脚复位

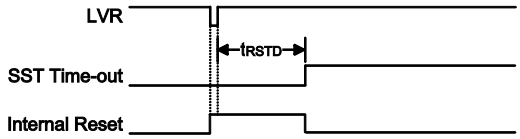
当单片机正常工作，而 $\overline{\text{RES}}$ 引脚通过外部硬件(如外部开关)被强迫拉至低电平时，此种复位形式即会发生。这种复位形式与其它复位的例子一样，程序计数器会被清除为零且程序从头开始执行。



$\overline{\text{RES}}$ 引脚复位时序图

- 低电压复位

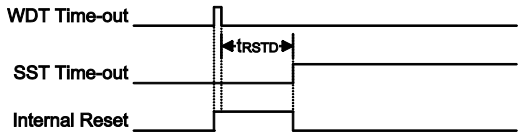
单片机具有低电压复位电路，用来监测它的电源电压，可通过掩膜选项进行选择。例如在更换电池的情况下，单片机供应的电压可能会落在 0.9V~V_{LVR} 的范围内，这时 LVR 将会自动复位单片机。有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过交流电气特性中 t_{LVR} 参数的值。如果低电压存在时间不超过此值，则 LVR 将会忽略它且不会执行复位功能。



低电压复位时序图

- 正常操作时看门狗溢出复位

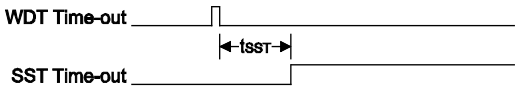
除了看门狗溢出标志位 TO 将被设为 1 之外，正常操作时看门狗溢出复位和 $\overline{RES}$ 复位相同。



正常操作时看门狗溢出复位时序图

- 暂停时看门狗溢出复位

暂停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除为 0 及 TO 标志位被设为 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。



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，被放在状态寄存器中，由如暂停功能或看门狗计数器等几种控制器操作控制。复位标志位如下所示：

TO	PDF	复位条件
0	0	上电时的 $\overline{RES}$ 复位
u	u	正常运行时的 $\overline{RES}$ 复位或 LVR 低电压复位
1	u	正常运行时的 WDT 溢出复位
1	1	暂停时的 WDT 溢出复位

注意：“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	所有中断被禁止
看门狗定时器	WDT 清除并重新计时
定时/计数器	定时/计数器关闭
预分频器	定时/计数器之预分频器内容清除
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

不同的复位形式以不同的途径影响单片机中的内部寄存器。为保证复位发生后程序的正常执行，在特定的复位发生后，了解单片机内的情况是非常重要的。下表描述了不同的复位如何影响单片机的内部寄存器。若芯片有多种封装类型，表格反映较大的封装的情况。

HT46F46E

寄存器	$\overline{RES}$复位 (上电时)	$\overline{RES}$或 LVR 复位	WDT 溢出复位 (正常运行时)	WDT 溢出复位 (暂停模式)
MP0	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu
BP	xxxx xxx0	xxxx xxx0	xxxx xxx0	xxxx xxxu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	---- 1111	---- 1111	---- 1111	---- uuuu
PBC	---- 1111	---- 1111	---- 1111	---- uuuu
PD	---- ---1	---- ---1	---- ---1	---- ---u
PDC	---- ---1	---- ---1	---- ---1	---- ---u
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	u--- --uu
EECR	1000 ----	1000 ----	1000 ----	uuuu ----

“u”表示不变化
“x”表示不确定
“-”表示不存在

HT46F47E

寄存器	$\overline{RES}$复位 (上电时)	$\overline{RES}$或 LVR 复位	WDT 溢出复位 (正常运行时)	WDT 溢出复位 (暂停模式)
MP0	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu
BP	xxxx xxx0	xxxx xxx0	xxxx xxx0	xxxx xxxu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	---- 1111	---- 1111	---- 1111	---- uuuu
PBC	---- 1111	---- 1111	---- 1111	---- uuuu
PD	---- ---1	---- ---1	---- ---1	---- ---u
PDC	---- ---1	---- ---1	---- ---1	---- ---u
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	x--- ----	x--- ----	x--- ----	u--- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	u--- --uu
EECR	1000 ----	1000 ----	1000 ----	uuuu ----

"u" 表示不变化
"x" 表示不确定
"-" 表示不存在

HT46F48E

寄存器	$\overline{RES}$复位 (上电时)	$\overline{RES}$或 LVR 复位	WDT 溢出复位 (正常运行时)	WDT 溢出复位 (暂停模式)
MP0	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu
BP	xxxx xxx0	xxxx xxx0	xxxx xxx0	xxxx xxxu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	---- --11	---- --11	---- --11	---- --uu
PCC	---- --11	---- --11	---- --11	---- --uu
PD	---- ---1	---- ---1	---- ---1	---- ---u
PDC	---- ---1	---- ---1	---- ---1	---- ---u
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	x--- ----	x--- ----	x--- ----	u--- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	u--- --uu
EECR	1000 ----	1000 ----	1000 ----	uuuu ----

"u" 表示不变化
"x" 表示不确定
"-" 表示不存在

HT46F49E

寄存器	$\overline{RES}$复位 (上电时)	$\overline{RES}$或 LVR 复位	WDT 溢出复位 (正常运行时)	WDT 溢出复位 (暂停模式)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	xxxx xxx0	xxxx xxx0	xxxx xxx0	xxxx xxxu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	---1 1111	---1 1111	---1 1111	---u uuuu
PCC	---1 1111	---1 1111	---1 1111	---u uuuu
PD	---- --11	---- --11	---- --11	---- --uu
PDC	---- --11	---- --11	---- --11	---- --uu
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	x--- ----	x--- ----	x--- ----	u--- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	u--- --uu
EECR	1000 ----	1000 ----	1000 ----	uuuu ----

"u" 表示不变化
"x" 表示不确定
"-" 表示不存在

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。有两种系统时钟可供选择，而看门狗定时器又有多种时钟源选项，提供了使用者较大的灵活性。所有的振荡器选项都是通过掩膜选项来完成。

系统时钟的产生有两种方法：

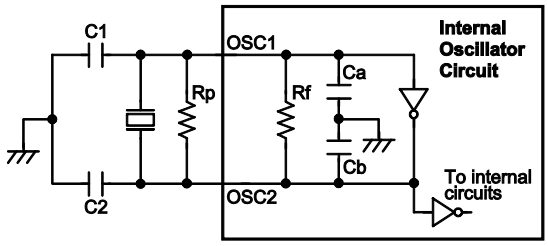
- 外部晶体/陶瓷振荡器
- 外部 RC 振荡器

系统时钟的设置可以通过掩膜选项来选择。

欲知更多振荡器电路的相关信息可参考 HOLTEK 网站应用范例 HA0075S。

外部晶体/谐振器

对于晶体振荡器的结构配置，只要简单地将晶体连接至 OSC1 和 OSC2，则会产生所需的相移及反馈，而不需其它外部的器件。然而为了保证起振和精确频率的产生，某些晶体振荡器和所有谐振器，建议使用两个小容量电容 C1 和 C2，具体数值与客户选择的晶体/谐振器有关。在许多应用场合，外部并联反馈 Rp 并不一定需要，但在某些应用中可能需要帮助振荡器的起振。



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

外部晶体/谐振器

内部 Ca，Cb，Rf 的典型值@5V，25℃		
Ca	Cb	Rf
11~13pF	13~15pF	470kΩ

振荡器内部元件参数值

晶体振荡器 C1 和 C2 参数值			
晶体频率	C1	C2	CL
8MHz	TBD	TBD	TBD
4MHz	TBD	TBD	TBD
1MHz	TBD	TBD	TBD
注：1、C1，C2 值仅作为参考。 2、CL 是晶体振荡器制造商指定负载电容值。			

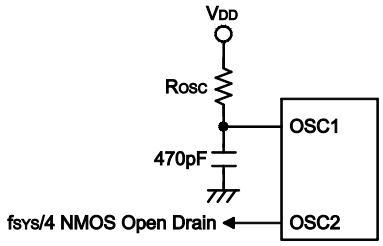
晶体振荡器推荐电容参数值

谐振器 C1 和 C2 参数值		
谐振器频率	C1	C2
3.58MHz	TBD	TBD
1MHz	TBD	TBD
455KHz	TBD	TBD
注：C1，C2 值仅作为参考。		

谐振器推荐电容参数值

外部 RC 振荡器

使用外部 RC 电路作为系统振荡器，需要在 OSC1 和 VDD 之间连接一个阻值约在 15KΩ到 750KΩ之间的电阻，OSC1 与地之间连接一个电容。产生的系统时钟 4 分频后提供给 OSC2 作输出，以达到与外部同步化的目的。由于 OSC2 为 NMOS 开漏输出，测量 RC 振荡频率时，则需要加上拉电阻。虽然此振荡器配置成本较低，但振荡频率会因 VDD、温度和芯片本身的制成而改变，因此不适合用来做计时严格或需要精确振荡器频率的场合。对于外部电阻 R_{OSC} 的阻值，请参考附录章节中典型 RC 振荡器对温度以及对 V_{DD} 特性曲线分析。注意：内部电容和外部电阻 R_{OSC} 共同作用决定频率值，图中显示的外部电容并不会影响振荡器的频率值。



RC 振荡器

看门狗定时振荡器

WDT 振荡器是一个完全独立且自由动作的内建振荡器，它在 5V 时的周期时间典型值为 65μs，且不需外部的器件搭配。WDT 振荡器通过掩膜选项进行选择。当选择 WDT 振荡器，芯片进入暂停模式后，系统时钟停止振荡，但 WDT 振荡器仍继续工作，看门狗功能有效。然而，由于看门狗的工作使得系统在暂停模式下功耗增加，所以在低功耗应用场合，WDT 振荡器可以通过掩膜选项来关闭。

暂停模式下的暂停和唤醒

暂停模式

所有 HOLTEK 单片机都具有暂停功能，即 HALT 模式或者睡眠模式。当单片机进入此模式，由于系统停止振荡，芯片的工作电流会降到较低静态模式。由于单片机保存了当前工作的一些内部条件，它可以被唤醒并继续运行，而不需要重新进行复位。这个特性对于电池供电系统等供电容量受限但需要单片机保存当前工作状态的应用场合特别重要。

进入暂停模式

单片机进入暂停模式仅能通过应用程序执行“HALT”指令实现，且造成如下结果：

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

静态电流注意事项

要使系统静态电流降到毫安级，除了需要单片机进入 HALT 模式，还要考虑到电路的设计，特别要注意输入/输出口的状态。所有高阻抗输入引脚必须接高电平或低电平，否则会造成引脚浮空而引起内部振荡。这同样适用于芯片具有不同封装类型的场合，将未引出的引脚必须设置为输出或带上拉电阻的输入。另外还需要注意单片机输出端口上的负载，当外部电路为 CMOS 输入时，可设置为拉电流。也要注意若内部看门狗振荡器使能，也将消耗部分额外的电流。

唤醒

当系统进入 HALT 模式，可以通过以下几种方式唤醒：

- 外部复位
- PA 口下降沿
- 系统中断
- WDT 溢出

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

端口 PA 中的每个位都可以通过掩膜选项独立选择唤醒功能。PA 端口唤醒后，程序将在“HALT”指令后继续执行。

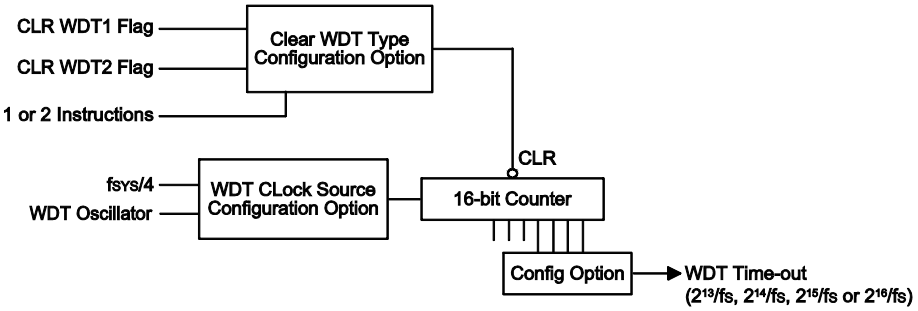
如果系统是通过中断唤醒，则有两种可能发生，假如中断除能或中断使能但堆栈已满，程序将在 HALT 指令下一条处继续执行，但相应的中断服务程序只有在中断使能或堆栈空闲后被执行；假如中断使能且堆栈未满，则正常的中断响应将会发生。假设在进入暂停模式之前外部中断请求标志位被设为“1”，则相关中断的唤醒功能将无效。

无论何种唤醒源，一旦唤醒事件发生，回到正常运行将需要 1024 个系统时钟周期。如果唤醒由中断发生，则真实的中断子程序执行将延迟一个或数个周期。如果唤醒后接着去执行“HALT”下一条指令，则它将在 1024 个系统时钟周期结束后立刻执行。

看门狗计数器

看门狗定时器的功能在于防止如电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。当 WDT 溢出时，它产生一个芯片复位的动作。WDT 时钟通过选择掩膜选项中两个时钟源之一提供：单片机内建 WDT 振荡器或指令时钟（系统时钟 4 分频）。要注意的是假如 WDT 掩膜选项设为除能，则任何相关的指令将无效。

在此系列单片机中，所有看门狗定时器的选项，如使能/除能、WDT 时钟源和清除看门狗指令条数都是通过掩膜选项来选择。没有与 WDT 相关的内部寄存器。WDT 的时钟源之一是内部振荡器，在供应电压为 5V 时周期近似为 65μs。必须注意的是，这个内部时钟的周期可以随着 VDD、温度和制作工艺而改变。另一个 WDT 时钟源选项是系统时钟的四分频。无论 WDT 时钟源是来自它内部的 WDT 振荡器或是来自指令时钟，都经过 2^{13}~2^{16} 分频（通过掩膜选项获得 WDT 溢出周期）。当设置 2^{16} 时，可使溢出周期最大为 4.3S。溢出周期可以随着温度、VDD 和制作工艺而改变。由于清除指令只复位计数链的最后一级，因此实际的分频系数和相应的 WDT 溢出时间会有因数为 2 的变化。看门狗精确的分频系数可以根据清除 WDT 指令到 WDT 计满所需的时间确定。必须明白没有单独的内部寄存器或掩膜选项与看门狗溢出时间相应，WDT 溢出时间完全取决于 f_{SYS}/4 或内部 WDT 振荡器。



看门狗定时器

注意的是，如果使用指令时钟作为时钟源，当系统进入暂停模式后，指令时钟会停止且 WDT 将失去其保护目的。在这种情况下，系统只能通过外部逻辑重新复位。当系统操作在干扰严重的环境时，建议使用内部 WDT 振荡器。

系统在正常运行状态下，WDT 溢出将导致芯片复位，并置位 TO 状态标志位。然而如果系统处于暂停模式，当发生 WDT 溢出时，状态寄存器中的 TO 位将被置位，同时仅复位程序计数器和 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($\overline{\text{RES}}$ 引脚低电平)，第二种是通过软件指令，而第三种是通过执行 HALT 指令。

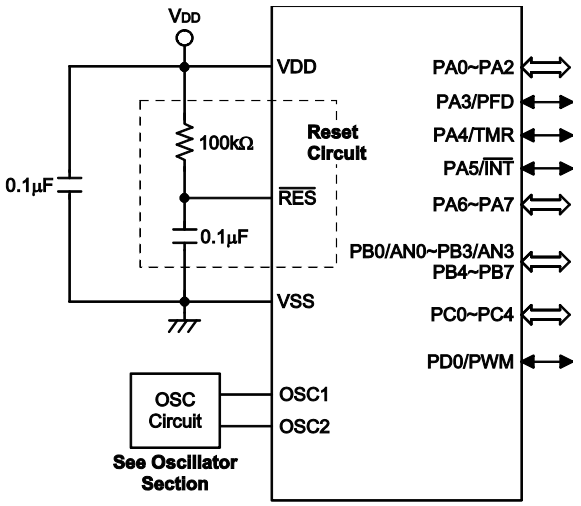
使用软件指令有两种方法去清除看门狗寄存器，必须由掩膜选项选择。第一种选择是使用单一“CLR WDT”指令，而第二种是使用“CLR WDT1”和“CLR WDT2”两个指令。对于第一种选择，只要执行“CLR WDT”便清除 WDT。而第二种选择，必须交替执行“CLR WDT1”和“CLR WDT2”两者才能成功的清除 WDT。关于第二种选择要注意的是，如果“CLR WDT1”正被使用来清除 WDT，接着再执行这条指令将是无效的，只有执行“CLR WDT2”指令才能清除 WDT。同样的“CLR WDT2”指令已经执行后，只有接着执行“CLR WDT1”指令才可以清除看门狗定时器。

掩膜选项

掩膜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掩膜选项。当掩膜选项烧入单片机后，无法再通过应用程序修改。所有位必须按系统的需要定义，具体内容可参考下表：

编号	选项
1	WDT 时钟源：WDT OSC 或 $f_{SYS}/4$
2	WDT 功能：使能或禁止
3	WDT溢出周期：$2^{13}/f_S$，$2^{14}/f_S$，$2^{15}/f_S$或$2^{16}/f_S$
4	CLR WDT 指令条数：1 或 2 条指令
5	OSC 类型选择： 晶体或 RC
6	PA、PB 和 PD 上拉电阻：有或无 PC 上拉电阻：有或无 — 仅对于 HT46F48E 和 HT46F49E 有效
7	PWM：使能或禁止 — HT46F49E 除外 PWM0、PWM1：使能或禁止 — 仅对 HT46F49E 有效 PWM 模式：选择 6+2 或者 7+1 模式
8	PA0~PA7：唤醒使能或除能 — 位选项
9	PFD：正常输入/输出或 PFD 输出
10	LVR 功能：使能或禁止 LVR 电压：2.1V，3.15V，4.2V
11	外部中断 INT 触发沿：除能，上升，下降，或双向（上升或下降）

应用电路



指令集

简介

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

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

大部分的操作均只需要一个指令周期来执行。分支、调用或查表则需要两个指令周期。一个指令周期相当于四个系统时钟周期，因此如果在 8MHz 的系统时钟振荡器下，大部分的操作将在 0.5μs 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者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。

分支和控制的转换

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，或者是内部数据位的值。

位运算

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

查表运算

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

其它运算

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

指令集概要

下列表格根据功能描述了指令集的分类，利用下表列出的惯例可以作为基本的指令参考。

表格惯例：

x：立即数

m：数据存储器地址

A：累加器

i：第 0～7 位

addr：程序存储器地址

助记符	说明	指令周期	影响标志位
算术运算			
ADD A,[m]	ACC 与数据存储器相加，结果放入 ACC	1	Z,C,AC,OV
ADDM A,[m]	ACC 与数据存储器相加，结果放入数据存储器	1 注	Z,C,AC,OV
ADD A,x	ACC 与立即数相加，结果放入 ACC	1	Z,C,AC,OV
ADC A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	1	Z,C,AC,OV
ADCM A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	1 注	Z,C,AC,OV
SUB A,x	ACC 与立即数相减，结果放入 ACC	1	Z,C,AC,OV
SUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z,C,AC,OV
SUBM A,[m]	ACC 与数据存储器相减，结果放入数据存储器	1 注	Z,C,AC,OV
SBC A,[m]	ACC 与数据存储器、进位标志的反相减，结果放入 ACC	1	Z,C,AC,OV
SBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1 注	Z,C,AC,OV
DAA [m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1 注	C
逻辑运算			
AND A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1 注	Z
ORM A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1 注	Z
XORM A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1 注	Z
AND A,x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR A,x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR A,x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL [m]	对数据存储器取反，结果放入数据存储器	1 注	Z
CPLA [m]	对数据存储器取反，结果放入 ACC	1	Z
递增和递减			
INCA [m]	递增数据存储器，结果放入 ACC	1	Z
INC [m]	递增数据存储器，结果放入数据存储器	1 注	Z
DECA [m]	递减数据存储器，结果放入 ACC	1	Z
DEC [m]	递减数据存储器，结果放入数据存储器	1 注	Z

助记符	说明	指令周期	影响标志位
移位			
RRA [m]	数据存储器右移一位，结果放入 ACC	1	无
RR [m]	数据存储器右移一位，结果放入数据存储器	1 注	无
RRCA [m]	带进位将数据存储器右移一位，结果放入 ACC	1	C
RRC [m]	带进位将数据存储器右移一位，结果放入数据存储器	1 注	C
RLA [m]	数据存储器左移一位，结果放入 ACC	1	无
RL [m]	数据存储器左移一位，结果放入数据存储器	1 注	无
RLCA [m]	带进位将数据存储器左移一位，结果放入 ACC	1	C
RLC [m]	带进位将数据存储器左移一位，结果放入数据存储器	1 注	C
数据传送			
MOV A,[m]	将数据存储器送至 ACC	1	无
MOV [m],A	将 ACC 送至数据存储器	1 注	无
MOV A,x	将立即数送至 ACC	1	无
位运算			
CLR [m].i	清除数据存储器的位	1 注	无
SET [m].i	置位数据存储器的位	1 注	无
转移			
JMP addr	无条件跳转	2	无
SZ [m]	如果数据存储器为零，则跳过下一条指令	1 注	无
SZA [m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	1 注	无
SZ [m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	1 注	无
SNZ [m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	1 注	无
SIZ [m]	递增数据存储器，如果结果为零，则跳过下一条指令	1 注	无
SDZ [m]	递减数据存储器，如果结果为零，则跳过下一条指令	1 注	无
SIZA [m]	递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1 注	无
SDZA [m]	递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1 注	无
CALL addr	子程序调用	2	无
RET	从子程序返回	2	无
RET A,x	从子程序返回，并将立即数放入 ACC	2	无
RETI	从中断返回	2	无
查表			
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	TO,PDF
CLR WDT2	预清除看门狗定时器	1	TO,PDF
SWAP [m]	交换数据存储器的高低字节，结果放入数据存储器	1 注	无
SWAPA [m]	交换数据存储器的高低字节，结果放入 ACC	1	无
HALT	进入暂停模式	1	TO,PDF

注： 1、对跳转指令而言，如果比较的结果牵涉到跳转即需 2 个周期，如果没有发生跳转，则只需一个周期。

2、任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

3、对于“CLR WDT1”或“CLR WDT2”指令而言，TO 和 PDF 标志位也许会受执行结果影响，“CLR WDT1”和 “ CLR WDT2”被连续地执行后，TO 和 PDF 标志位会被清除，除此之外 TO 和 PDF 标志位保持不变。

指令定义

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

ADCM A, [m] Add the accumulator and carry to the accumulator
说明： 将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。
运算过程： [m]←ACC+[m]+C
影响标志位： OV、Z、AC、C

ADD A, [m] Add data memory to the accumulator
说明： 将指定的数据存储器和累加器内容相加，结果存放到累加器。
运算过程： ACC←ACC+[m]
影响标志位： OV、Z、AC、C

ADD A, x Add immediate data to the accumulator
说明： 将累加器和立即数相加，结果存放到累加器。
运算过程： ACC←ACC+x
影响标志位： OV、Z、AC、C

ADDM A, [m] Add the accumulator to the data memory
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。
运算过程： [m]←ACC+[m]
影响标志位： OV、Z、AC、C

AND A, [m] Logical AND accumulator with data memory
说明： 将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。
运算过程： ACC←ACC “AND” [m]
影响标志位： Z

AND A, x Logical AND immediate data to the accumulator

说明： 将累加器中的数据和立即数做逻辑与，结果存放到累加器。
运算过程： ACC←ACC “AND” x
影响标志位： Z

ANDM A, [m] Logical AND data memory with the accumulator
说明： 将指定数据存储器内容和累加器中的数据做逻辑与，结果存放到数据存储器。
运算过程： [m]←ACC “AND” [m]
影响标志位： Z

CALL addr Subroutine call

说明： 无条件的调用指定地址的子程序，此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈，接着载入指定地址并从新地址执行程序。由于指令需要额外的运算，所以此指令为 2 个周期。

运算过程： Stack←Program Counter+1
Program Counter ← addr

影响标志位： 无

CLR [m] Clear data memory

说明： 将指定数据存储器的内容清零。

运算过程： [m] ←00H

影响标志位： 无

CLR [m] . i Clear bit of data memory

说明： 将指定数据存储器的 i 位内容清零。

运算过程： [m].i ←0

影响标志位： 无

CLR WDT Clear Watchdog Timer

说明： WDT 计数器、暂停标志位 PDF 和看门狗溢出标志位 TO 清零。

运算过程： WDT ← 00H
PDF & TO ← 0

影响标志位： TO、PDF

CLR WDT1 Preclear Watchdog Timer

说明： PDF 和 TO 标志位都被清 0。必须配合 CLR WDT2 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT1，而没有执行 CLR WDT2 时，PDF 与 TO 保留原状态不变。

运算过程： WDT ← 00H
PDF & TO ← 0

影响标志位： TO、PDF

CLR WDT2 Preclear Watchdog Timer

说明： PDF 和 TO 标志位都被清 0。必须配合 CLR WDT1 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT2，而没有执行 CLR WDT1 时，PDF 与 TO 保留原状态不变。

运算过程： WDT ← 00H
PDF & TO ← 0

影响标志位： TO、PDF

CPL [m] Complement data memory

说明： 将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或从 0 变 1。

运算过程： [m] ← [$\overline{m}$]

影响标志位： Z

CPLA [m] Complement data memory
说明： 将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或从 0 变 1，结果被存放回累加器且数据寄存器的内容保持不变。
运算过程： ACC ← [$\overline{m}$]
影响标志位： Z

DAA [m] Decimal-Adjust accumulator for addition
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。
操作： [m] ←ACC+00H 或 [m] ←ACC+06H
[m] ←ACC+60H 或 [m] ←ACC+66H
影响标志位： C

DEC [m] Decrement data memory
说明： 将指定数据存储器的内容减 1。
运算过程： [m] ← [m]-1
影响标志位： Z

DECA [m] Decrement data memory and place result in the accumulator
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。
运算过程： ACC ← [m]-1
影响标志位： Z

HALT Enter power down mode
说明： 此指令终止程序执行并关掉系统时钟，RAM 和寄存器的内容保持原状态，WDT 计数器和分频器被清“0”，暂停标志位 PDF 被置位 1，WDT 溢出标志位 TO 被清 0。
运算过程： PDF ← 1
TO ← 0
影响标志位： TO、PDF

INC [m] Increment data memory

说明： 将指定数据存储器的内容加 1。
运算过程： [m] ← [m]+1
影响标志位： Z

INCA [m] Increment data memory and place result in the accumulator
说明： 将指定数据存储器的内容加 1，结果存放回累加器并保持指定的数据存储器内容不变。
运算过程： ACC ← [m]+1
影响标志位： Z

JMP addr Directly jump

说明： 程序计数器的内容无条件地由被指定的地址取代，程序由新的地址继续执行。当新的地址被加载时，必须插入一个空指令周期，所以此指令为 2 个周期的指令。
运算过程： PC ← addr
影响标志位： 无

MOV A, [m] Move data memory to the accumulator
说明： 将指定数据存储器的内容复制到累加器。
运算过程： ACC ← [m]
影响标志位： 无

MOV A, x Move immediate data to the accumulator
说明： 将 8 位立即数载入累加器。
运算过程： ACC ← x
影响标志位： 无

MOV [m], A Move the accumulator data to memory
说明： 将累加器的内容复制到指定的数据存储器。
运算过程： [m] ← ACC
影响标志位： 无

NOP No operation
说明： 空操作，顺序执行下一条指令。
运算过程： PC ← PC+1
影响标志位： 无

OR A, [m] Logical OR accumulator with data memory
说明： 将累加器中的数据和指定的数据存储器内容逻辑或，结果存放到累加器。
运算过程： ACC←ACC “OR” [m]
影响标志位： Z

OR A, x Logical OR immediate data to the accumulator
说明： 将累加器中的数据和立即数逻辑或，结果存放到累加器。
运算过程： ACC←ACC “OR” x
影响标志位： Z

ORM A, [m] Logical OR data memory with accumulator
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。
运算过程： [m]←ACC “OR” [m]
影响标志位： Z

RET Return from subroutine
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。
运算过程： PC ← Stack
影响标志位： 无

RET A, x Return and place immediate data in the accumulator

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。

运算过程： PC ← Stack
ACC ← x

影响标志位： 无

RETI Return from interrupt

说明： 将堆栈寄存器中的程序计数器值恢复且中断功能通过设置 EMI 位重新使能。EMI 是控制中断使能的主控制位。如果在执行 RETI 指令之前还有中断未被相应，则这个中断将在返回主程序之前被相应。

运算过程： PC ← Stack
EMI ← 1

影响标志位： 无

RL [m] Rotate data memory left

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。

运算过程： [m].(i+1) ←[m].i (i=0~6)
[m].0 ←[m].7

影响标志位： 无

RLA [m] Rotate data memory left and place result in the accumulator

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。

运算过程： 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

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。

运算过程： [m].(i+1) ←[m].i (i=0~6)
[m].0 ←C
C ← [m].7

影响标志位： C

RLCA [m] Rotate left through carry and place result in the accumulator

说明： 将指定数据存储器的内容连同进位标志左移 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。

运算过程： 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

说明： 将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。

运算过程： [m].i ←[m].(i+1) (i=0~6)
[m].7 ←[m].0,

影响标志位： 无

RRA [m] Rotate right and place result in the accumulator

说明： 将指定数据存储器的内容循环右移 1 位，第 0 位移到第 7 位，移位结果存放到累加器，而指定数据存储器的内容保持不变。

运算过程： ACC.i ←[m].(i+1) (i=0~6)
ACC.7 ←[m].0

影响标志位： 无

RRC [m] Rotate data memory right through carry

说明： 将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位。

运算过程： [m].i ←[m].(i+1) (i=0~6)
[m]. 7 ←C
C ← [m].0

影响标志位： C

RRCA [m] Rotate right through carry and place result in the accumulator

说明： 将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位,移位结果送回累加器，但是指定数据寄存器的内容保持不变。

运算过程： ACC.i ←[m].(i+1) (i=0~6)
ACC.7 ←C
C ← [m].0

影响标志位： C

SBC A,[m] Subtract data memory and carry from the accumulator

说明： 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

运算过程： ACC←ACC－[m]－$\overline{C}$

影响标志位： OV、Z、AC、C

SBCM A,[m] Subtract data memory and carry from the accumulator

说明： 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

运算过程： ACC←ACC－[m]－$\overline{C}$

影响标志位： OV、Z、AC、C

SDZ [m] Skip if decrement data memory is 0
说明： 将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
运算过程： [m] ← [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
说明： 将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
运算过程： ACC ← [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
说明： 将指定数据存储器的每一位设置为 1。
运算过程： [m] ← FFH
影响标志位： 无

SET [m]. i Set bit of data memory
说明： 将指定数据存储器的第 i 位设置为 1。
运算过程： [m].i ←1
影响标志位： 无

SIZ [m] Skip if increment data memory is 0
说明： 将指定的数据存储器的内容加 1，判断是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
运算过程： [m] ←[m]+1，如果[m]=0 跳过下一条指令执行
影响标志位： 无

SIZA [m] Increment data memory and place result in ACC,skip if 0
说明： 将指定数据存储器的内容加 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果会被存放到累加器，但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
运算过程： ACC ←[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 the data memory is not 0
说明： 判断指定数据存储器的第 i 位，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果为 0，则程序继续执行下一条指令。
运算过程： 如果[m].i≠0，跳过下一条指令执行
影响标志位： 无

SUB A, [m] Subtract data memory from the accumulator
说明： 将累加器的内容减去指定的数据存储器的数据，把结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
运算过程： ACC←ACC－[m]
影响标志位： OV、Z、AC、C

SUBM A, [m] Subtract data memory from the accumulator
说明： 将累加器的内容减去指定数据存储器的数据，结果存放到指定的数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
运算过程： [m]←ACC－[m]
影响标志位： OV、Z、AC、C

SUB A, x Subtract immediate data from the accumulator
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。
运算过程： ACC←ACC－x
影响标志位： OV、Z、AC、C

SWAP [m] Swap nibbles within the data memory
说明： 将指定数据存储器的低 4 位和高 4 位互相交换。
运算过程： [m].3~[m].0↔[m].7~[m].4
影响标志位： 无

SWAPA [m] Swap data memory and place result in the accumulator
说明： 将指定数据存储器的低 4 位和高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。
运算过程： 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
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。
运算过程： 如果[m] = 0, 跳过下一条指令执行
影响标志位： 无

SZA [m] Move data memory to ACC,skip if 0
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。
运算过程： 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 the data memory is 0
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。
运算过程： 如果[m].i = 0，跳过下一条指令执行
影响标志位： 无

TABRDC [m] Move the ROM code(current page) to TBLH and data memory
说明： 将表格指针 TBLP 所指的程序代码低字节（当前页）移至指定的数据存储器且将高字节移至 TBLH。
运算过程： [m] ←程序代码（低字节）
TBLH←程序代码（高字节）
影响标志位： 无

TABRDL [m] Move the ROM code(last page) to TBLH and data memory
说明： 将表格指针 TBLP 所指的程序代码低字节（最后一页）移至指定的数据存储器且将高字节移至 TBLH。
运算过程： [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 accumulator with data memory
说明： 将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
运算过程： ACC←ACC “XOR” [m]
影响标志位： Z

XORM A, [m] Logical XOR data memory with accumulator
说明： 将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。
运算过程： [m]←ACC “XOR” [m]
影响标志位： Z

XOR A, x Logical XOR immediate data to the accumulator
说明： 将累加器的数据与立即数逻辑异或，结果存放到累加器。
运算过程： ACC←ACC “XOR” x
影响标志位： Z

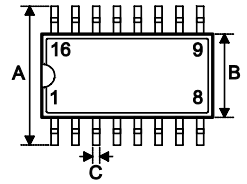
封装信息

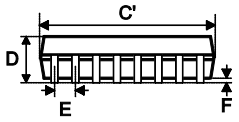
请注意，这里提供的封装信息仅作为参考。由于这个信息经常更新，提醒用户咨询 Holtek 网站 以获取最新版本的封装信息。

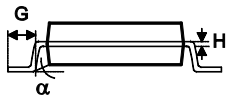
封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

16-pin NSOP(150mil)外形尺寸



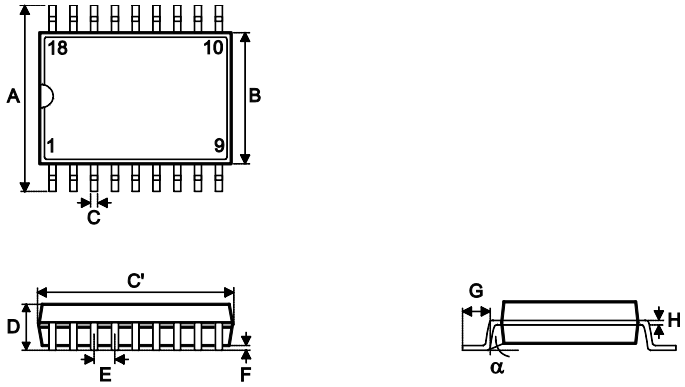




符号	尺寸（单位：inch）		
	最小	典型	最大
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸（单位：mil）		
	最小	典型	最大
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°

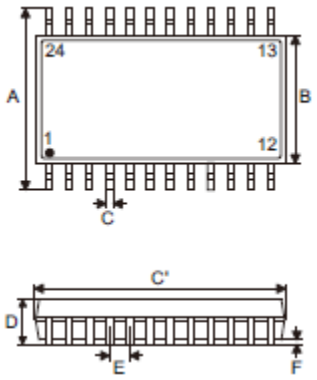
18-pin SOP(300mil)外形尺寸



符号	尺寸 （单位：inch）		
	最小	典型	最大
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C’	—	0.455 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸 （单位：mil）		
	最小	典型	最大
A	—	10.30 BSC	—
B	—	7.5 BSC	—
C	0.31	—	0.51
C’	—	11.55 BSC	—
D	—	—	2.65
E	—	1.27 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

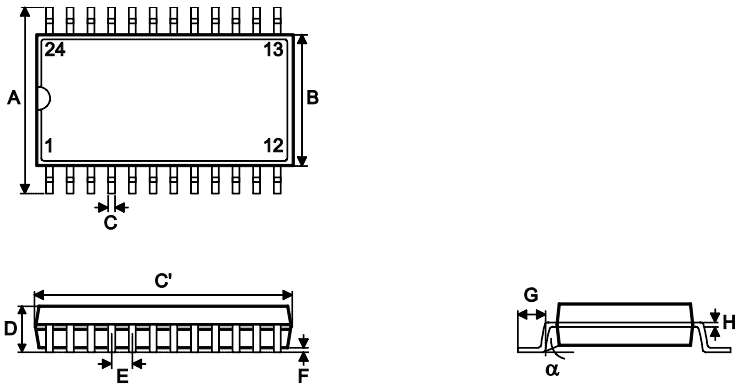
20-pin SSOP(150mil)外形尺寸



符号	尺寸（单位：inch）		
	最小	典型	最大
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：mil）		
	最小	典型	最大
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°

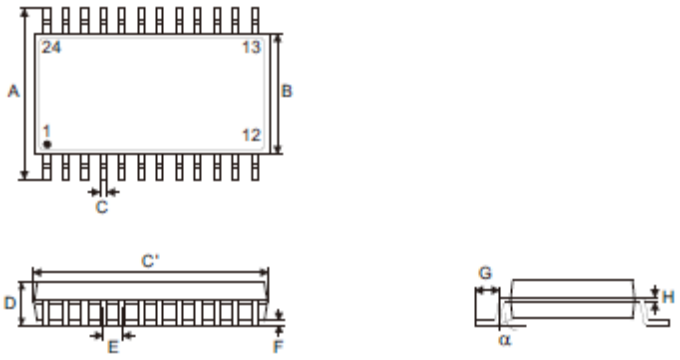
24-pin SOP(300mil)外形尺寸



符号	尺寸（单位：inch）		
	最小	典型	最大
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C’	—	0.606 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸（单位：mil）		
	最小	典型	最大
A	—	10.30 BSC	—
B	—	7.50 BSC	—
C	0.31	—	0.51
C’	—	15.40 BSC	—
D	—	—	2.65
E	—	1.27 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

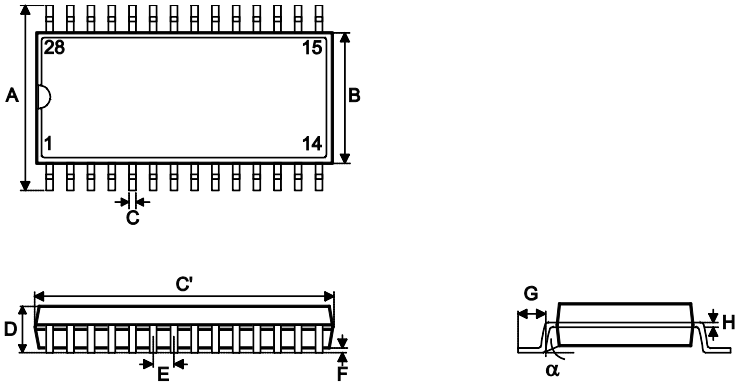
24-pin SSOP(150mil)外形尺寸



符号	尺寸（单位：inch）		
	最小	典型	最大
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：mil）		
	最小	典型	最大
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°

28-pin SOP(300mil)外形尺寸



符号	尺寸（单位：inch）		
	最小	典型	最大
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C’	—	0.705 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸（单位：mil）		
	最小	典型	最大
A	—	10.30 BSC	—
B	—	7.50 BSC	—
C	0.31	—	0.51
C’	—	17.90 BSC	—
D	—	—	2.65
E	—	1.27 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°