多路 RGB LED Flash 单片机

HT45F0063

版本：V1.10　日期：2022-09-06

www.holtek.com

目录

特性

CPU 特性

- 工作电压
  - ♦ f_{SYS}=8MHz：2.2V~5.5V
- V_{DD}=5V，系统时钟为 8MHz 时，指令周期为 0.5μs
- 提供暂停和唤醒功能，以降低功耗
- 振荡器类型
  - ♦ 内部高速 8MHz RC – HIRC
  - ♦ 内部低速 32kHz RC – LIRC
- 多种工作模式：快速、低速、空闲和休眠
- 内部集成的振荡器无需外接元件
- 所有指令都可在 1 或 2 个指令周期内完成
- 查表指令
- 61 条功能强大的指令系统
- 4 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器：4K×16
- RAM 数据存储器：256×8
- 看门狗定时器功能
- 20 个双向 I/O 口
- 一个 10-bit CTM 用于时间测量、比较匹配输出及 PWM 输出功能
- 恒定电流 LED 驱动器
- COM 扫描输出驱动器
- 级联式收发器接口
- I^2C 接口
- 时基功能可提供固定时间的中断信号
- 低电压复位功能
- Flash 程序存储器烧录可达 10,000 次
- Flash 程序存储器数据可保存 10 年以上
- 封装类型：24-pin QFN/SSOP-EP

概述

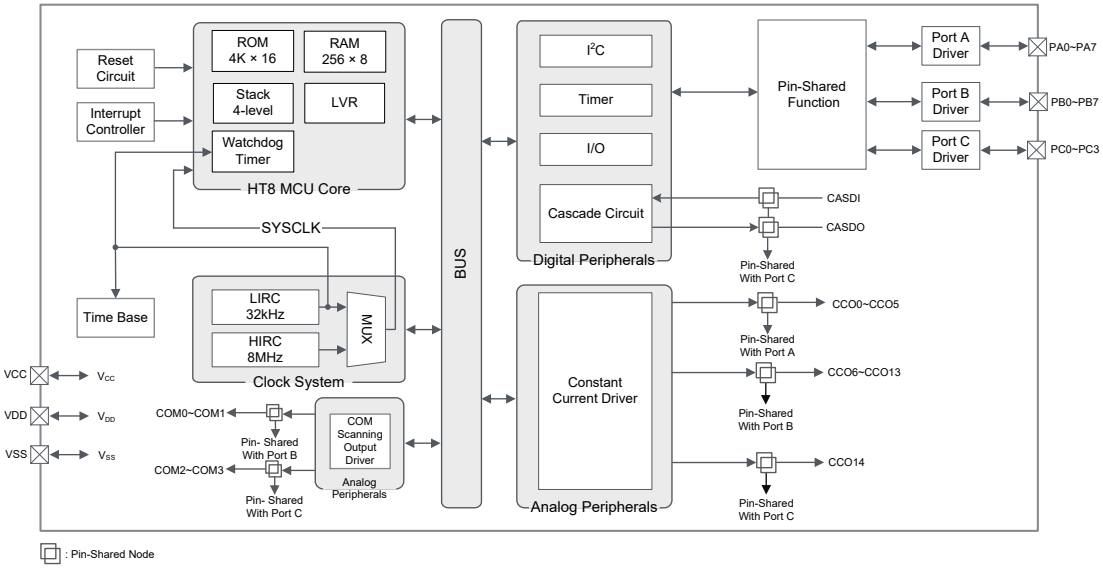
HT45F0063 单片机是一款 8 位高性能精简指令集的 Flash 单片机，专门用于多路 RGB 调光 LED 控制应用。它具有一系列功能和特性，其 Flash 存储器可多次编程的特性给用户提供了较大的方便。存储器方面，还包含了一个 RAM 数据存储器。

该单片机带有一个使用灵活的定时器模块，可提供定时功能、比较匹配输出功能及 PWM 产生等功能。内建 I^2C 接口为设计者提供了一个易与外部硬件通信的方法。内部看门狗定时器和低电压复位等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

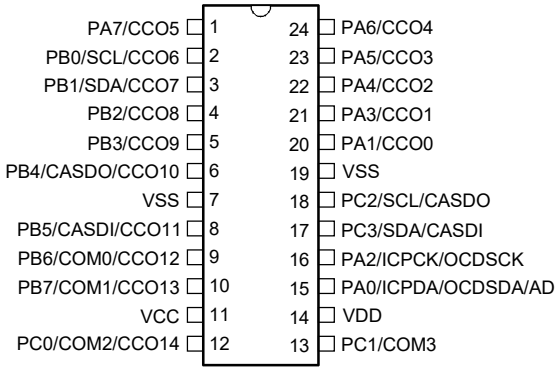
该单片机提供了内部高速和低速振荡器作为系统振荡器功能选项，无需外接元件。其在不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

针对 RGB 调光应用，该单片机还内置了多个相关产品所需的电路，包括 COM 扫描输出驱动器、PWM 功能、恒定电流 LED 驱动器以及级联式收发器接口。外加 I/O 使用灵活、时基功能和其它特性，使该单片机可以广泛应用于各种产品中，例如 LED 呼吸灯、圣诞灯、灯条及情景灯等。

方框图



引脚图



HT45F0063/HT45V0063
24 SSOP-EP-A

注：1. 若共用脚有多种输出，所需引脚共用功能由相应的软件控制位决定。
2. HT45V0063 是 HT45F0063 的 OCDS EV 芯片，OCDSDA 和 OCDSCK 引脚为 OCDS 专用引脚，仅存在于 OCDS EV 芯片。

引脚说明

每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/ICPDA/ OCDSDA/AD	PA0	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片
	AD	—	ST	—	I²C 接口设备地址选择引脚
PA1/CCO0	PA1	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	CCO0	PAS0	—	AN	LED PWM 恒定电流输出
PA2/ICPCK/ OCDSCK	PA2	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	ICPCK	—	ST	—	ICP 时钟
	OCDSCK	—	ST	—	OCDS 时钟，仅用于 EV 芯片
PA3/CCO1	PA3	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	CCO1	PAS0	—	AN	LED PWM 恒定电流输出
PA4/CCO2	PA4	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	CCO2	PAS1	—	AN	LED PWM 恒定电流输出
PA5/CCO3	PA5	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	CCO3	PAS1	—	AN	LED PWM 恒定电流输出
PA6/CCO4	PA6	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	CCO4	PAS1	—	AN	LED PWM 恒定电流输出
PA7/CCO5	PA7	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	CCO5	PAS1	—	AN	LED PWM 恒定电流输出
PB0/SCL/CCO6	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SCL	PBS0 IFS	ST	NMOS	I²C 时钟线
	CCO6	PBS0	—	AN	LED PWM 恒定电流输出

引脚名称	功能	OPT	I/T	O/T	说明
PB1/SDA/CCO7	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDA	PBS0 IFS	ST	NMOS	I²C 数据线
	CCO7	PBS0	—	AN	LED PWM 恒定电流输出
PB2/CCO8	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CCO8	PBS0	—	AN	LED PWM 恒定电流输出
PB3/CCO9	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CCO9	PBS0	—	AN	LED PWM 恒定电流输出
PB4/CASDO/ CCO10	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CASDO	PBS1	—	CMOS	级联式收发器接口输出
	CCO10	PBS1	—	AN	LED PWM 恒定电流输出
PB5/CASDI/ CCO11	PB5	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CASDI	PBS1 IFS	ST	—	级联式收发器接口输入
	CCO11	PBS1	—	AN	LED PWM 恒定电流输出
PB6/COM0/ CCO12	PB6	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	COM0	PBS1	—	AN	COM 扫描输出
	CCO12	PBS1	—	AN	LED PWM 恒定电流输出
PB7/COM1/ CCO13	PB7	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	COM1	PBS1	—	AN	COM 扫描输出
	CCO13	PBS1	—	AN	LED PWM 恒定电流输出
PC0/COM2/ CCO14	PC0	PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	COM2	PCS0	—	AN	COM 扫描输出
	CCO14	PCS0	—	AN	LED PWM 恒定电流输出
PC1/COM3	PC1	PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	COM3	PCS0	—	AN	COM 扫描输出
PC2/SCL/CASDO	PC2	PCS0 IFS	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SCL	PCS0 IFS	ST	NMOS	I²C 时钟线
	CASDO	PCS0	—	CMOS	级联式收发器接口输出
PC3/SDA/CASDI	PC3	PCS0 IFS	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDA	PCS0	ST	NMOS	I²C 数据线
	CASDI	PCS0	ST	—	级联式收发器接口输入
VDD	VDD	—	PWR	—	正电源电压，$V_{DD} \leq V_{CC}$

引脚名称	功能	OPT	I/T	O/T	说明
VCC	VCC	—	PWR	—	COM 驱动器正电源电压，$V_{DD} \leq V_{CC}$
VSS	VSS	—	PWR	—	负电源电压，接地

注：I/T：输入类型；
OPT：通过寄存器选项来配置；
ST：施密特触发输入；
CMOS：CMOS 输出；
O/T：输出类型；
PWR：电源；
NMOS：NMOS 输出；
AN：模拟信号

极限参数

电源供应电压 ..... V_{SS}-0.3V ~ 6.0V
输入电压 ..... V_{SS}-0.3V ~ V_{DD}+0.3V
储存温度 ..... -60°C ~ 150°C
工作温度 ..... -40°C ~ 85°C
I_{OH} 总电流 ..... -80mA
I_{OL} 总电流 80mA
最大结点温度 (Tj) ..... 125°C
热阻 (Rth)
24 QFN ..... 47°C/W
24 SSOP-EP ..... 31.5°C/W
功耗 (PD)
24 QFN @Ta=25°C 2.13W
24 QFN @Ta=85°C 0.85W
24 SSOP-EP @Ta=25°C 3.17W
24 SSOP-EP @Ta=85°C 1.27W
总功耗 500mW

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

直流电气特性

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

工作电压特性

Ta=25°C

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	$f_{SYS}=f_{HIRC}$=8MHz	2.2	—	5.5	V
	工作电压 – LIRC	$f_{SYS}=f_{LIRC}$=32kHz	2.2	—	5.5	V

待机电流特性

Ta=25°C

符号	待机模式	测试条件		最小	典型	最大	最大 @85°C	单位
		V_{DD}	条件					
I_{STB}	休眠模式	3V	WDT off	—	0.11	0.15	2.00	μA
		5V		—	0.18	0.38	2.90	
		3V	WDT on	—	3	5	6	μA
		5V		—	5	10	12	
	空闲模式 0	3V	f_{SUB} on	—	3	5	6	μA
		5V		—	5	10	12	
	空闲模式 1 – HIRC	3V	f_{SUB} on，$f_{SYS}=f_{HIRC}$=8MHz	—	360	500	600	μA
		5V		—	600	800	960	

注：当使用该表格电气特性数据时，以下几点需要注意：

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

工作电流特性

Ta=25°C

符号	工作模式	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	工作电流 (LIRC)	3V	$f_{SYS}=f_{LIRC}$=32kHz	—	10	20	μA
		5V		—	30	50	
	工作电流 (HIRC)	3V	$f_{SYS}=f_{HIRC}$=8MHz	—	1.0	2.0	mA
		5V		—	2.0	3.0	
		3V	$f_{SYS}=f_{HIRC}/2$，f_{HIRC}=8MHz	—	1.0	1.5	mA
		5V		—	1.5	2.0	
		3V	$f_{SYS}=f_{HIRC}/4$，f_{HIRC}=8MHz	—	0.9	1.3	mA
		5V		—	1.3	1.8	
		3V	$f_{SYS}=f_{HIRC}/8$，f_{HIRC}=8MHz	—	0.8	1.1	mA
		5V		—	1.1	1.6	
		3V	$f_{SYS}=f_{HIRC}/16$，f_{HIRC}=8MHz	—	0.7	1.0	mA
		5V		—	1.0	1.4	
		3V	$f_{SYS}=f_{HIRC}/32$，f_{HIRC}=8MHz	—	0.6	0.9	mA
		5V		—	0.9	1.2	
		3V	$f_{SYS}=f_{HIRC}/64$，f_{HIRC}=8MHz	—	0.5	0.8	mA
		5V		—	0.8	1.1	

注：当使用该表格电气特性数据时，以下几点需要注意：

1. 任何数字输入都设置为非浮空的状态。

2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有工作电流数值通过一个连续的 NOP 指令循环程序测得。

交流电气特性

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

内部高速振荡器 – HIRC – 频率精确度

程序烧录时，烧录器会调整 HIRC 振荡器使其工作在用户选择的 HIRC 频率和工作电压 (3V 或 5V) 条件下。

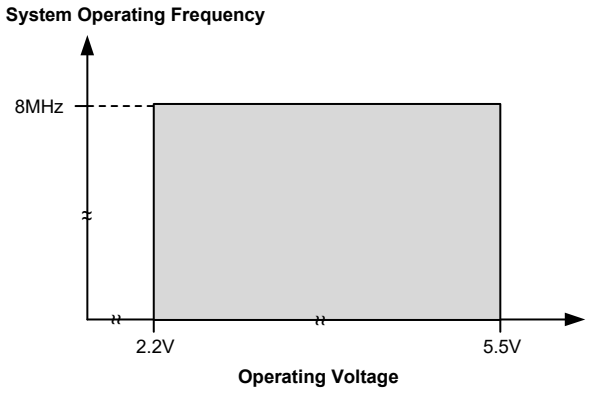
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{HIRC}	通过烧录器调整后的 8MHz HIRC 频率	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	

注：1. 烧录器可在 3V/5V 这两个可选的固定电压下对 HIRC 频率进行调整，在此提供 V_{DD}=3V/5V 时的参数值。
2. 3V/5V 表格列下面提供的是全压条件下的参数值。对于电压范围在 2.2V~3.6V 的应用，建议调整电压固定在 3V；对于电压范围在 3.3V~5.5V 的应用，建议调整电压固定在 5V。

内部低速振荡器电气特性 – LIRC

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{LIRC}	LIRC 振荡器频率	5V	25°C	25.6	32.0	38.4	kHz
		2.2V~5.5V	25°C	12.8	32.0	41.6	
			-40°C~85°C	8	32	60	
t_{START}	LIRC 启动时间	—	-40°C~85°C	—	—	100	μs

工作频率电气特性曲线图



系统上电时间电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{SST}	系统启动时间 (从 f_{SYS} off 的状态下唤醒)	—	$f_{SYS}=f_H\sim f_H/64$，$f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	系统启动时间 (从 f_{SYS} on 的状态下唤醒)	—	$f_{SYS}=f_H\sim f_H/64$，$f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	—	f_{HIRC} off → on	—	16	—	t_{HIRC}
t_{RSTD}	系统复位延迟时间 (上电复位或 LVR 硬件复位)	—	RR_{POR}=5V/ms	25	50	150	ms
	系统复位延迟时间 (LVRC/WDTC 软件复位)	—	—				
	系统复位延迟时间 (WDT 溢出)	—	—	8.3	16.7	50.0	ms
t_{SRESET}	软件复位最小延迟脉宽	—	—	45	90	250	μs

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

2. t_{HIRC} 等符号所表示的时间单位，是对应频率值的倒数，相关频率值在前面表格有说明。例如，$t_{HIRC}=1/f_{HIRC}$，$t_{SYS}=1/f_{SYS}$ 等等。

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

4. 系统速度切换时间实际上是指新使能的振荡器的启动时间。

输入 / 输出电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IL}	I/O 口低电平输入电压	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	I/O 口高电平输入电压	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	I/O 口灌电流	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	I/O 口源电流	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
R_{PH}	I/O 口上拉电阻 (注)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	I/O 口输入漏电流	5V	$V_{IN}=V_{DD}$ 或 $V_{IN}=V_{SS}$	—	—	±1	μA

注：R_{PH} 内部上拉电阻值的计算方法是：将引脚接地并使能 CCEN 位和 CCOPUx 寄存器以及 RGBn 寄存器且使能上拉电阻功能，然后在特定电源电压下测量输出源电流，最后电压除以测量的电流值从而得到此上拉电阻值。

LVR 电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{LVR}	低电压复位电压	—	LVR 使能，电压选择 2.1V	-5%	2.1	+5%	V
I_{LVR}	工作电流	3V	LVR 使能，V_{LVR}=2.1V	—	—	15	μA
		5V		—	15	25	
t_{LVR}	产生 LVR 复位的低电压最短保持时间	—	—	140	600	1000	μs

COM 驱动器和恒定电流特性

$V_{DD} \leq V_{CC}$，Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{CC}	工作电压	—	—	3.0	—	5.5	V
I_{CCS}	恒流功能使能的额外电流	3V	CCOn=off	—	3.8	5.0	mA
		5V		—	5.0	6.5	
I_{CCO}	CCOn 输出灌电流	5V	电流范围，V_{CCOn}=1.5V CCG[3:0]=0000，BC[7:0]=11111111	-13%	3.2	+13%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=0001，BC[7:0]=11111111	-10%	6.2	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=0010，BC[7:0]=11111111	-10%	9.2	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=0011，BC[7:0]=11111111	-10%	12.2	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=0100，BC[7:0]=11111111	-10%	15	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=0101，BC[7:0]=11111111	-10%	18	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=0110，BC[7:0]=11111111	-10%	21	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=0111，BC[7:0]=11111111	-10%	24	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=1000，BC[7:0]=11111111	-10%	27	+10%	mA

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{CCO}	CCOn 输出灌电流	5V	电流范围，V_{CCOn}=1.5V CCG[3:0]=1001， BC[7:0]=11111111	-10%	30	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=1010， BC[7:0]=11111111	-10%	33	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=1011， BC[7:0]=11111111	-10%	36	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=1100， BC[7:0]=11111111	-10%	39	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=1101， BC[7:0]=11111111	-10%	42	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=1110， BC[7:0]=11111111	-10%	45	+10%	mA
			电流范围，V_{CCOn}=1.5V CCG[3:0]=1111， BC[7:0]=11111111	-10%	48	+10%	mA
dI_{CCO1}	电流偏移率 (通道)	3V/5V	I_{CCOn}=33mA，V_{CCOn}=0.7V	—	±1.5	±3	%
dI_{CCO2}	电流偏移率 (IC)	3V/5V	I_{CCOn}=33mA，V_{CCOn}=0.7V	—	±3	±6	%
R_{PH}	I/O 口上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
$\%/dV_{CCO}$	输出电流 vs. 输出电压调整率	3V/5V	V_{CCOn}=0.7V~3.0V， I_{CCOn}=33mA	—	±0.1	—	%/V
$\%/dV_{DD}$	输出电流 vs. 电源电压调整率	—	V_{DD}=3.0V~5.5V， V_{CCOn}=0.7V	—	±1.0	±8.0	%/V
I_{COM_OH}	COM 驱动器源电流	4.5V	V_{OH}=4V	-364	-430	—	mA
R_{COM_PL}	COM 驱动器下拉电阻	4.5V	COMmOEN=1， COMm_data=0	6	10	18	kΩ

注：1. $\%/dV_{CCO}$={[I_{CCOn}(@ V_{CCOn}=3.0V) - I_{CCOn}(@V_{CCOn}=0.7V)] / I_{CCOn}(@ V_{CCOn}=1.5V)}×100% / (3.0V - 0.7V)

2. $\%/dV_{DD}$={[I_{CCOn}(@ V_{DD}=5.5V) - I_{CCOn}(@ V_{DD}=3.0V)] / I_{CCOn}(@ V_{DD}=4.0V)}×100% / (5.5V - 3.0V)

3. R_{COM_PL} 内部下拉电阻值的计算方法是：将引脚接到电源电压并设置COMmOEN=1，COMm_data=0，然后在特定电源电压下测量输入电流，最后电压除以测量的电流值从而得到此电阻值。

级联式收发器特性

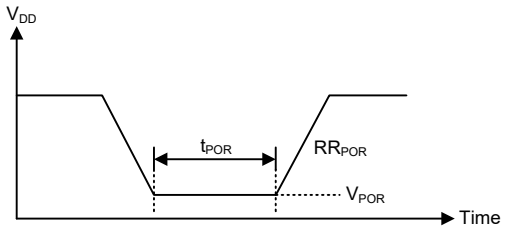
Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{CASDI}	CASDI 输入引脚最小延迟脉宽	—	—	0.3	—	—	μs
$f_{CASCLKI}$	级联式收发器接口时钟输入最大频率	5V	—	—	—	20	MHz

上电复位特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{POR}	上电复位电压	—	—	—	—	100	mV
RR_{POR}	上电复位电压速率	—	—	0.035	—	—	V/ms
t_{POR}	V_{DD} 保持为 V_{POR} 的最小时间	—	—	1	—	—	ms



系统结构

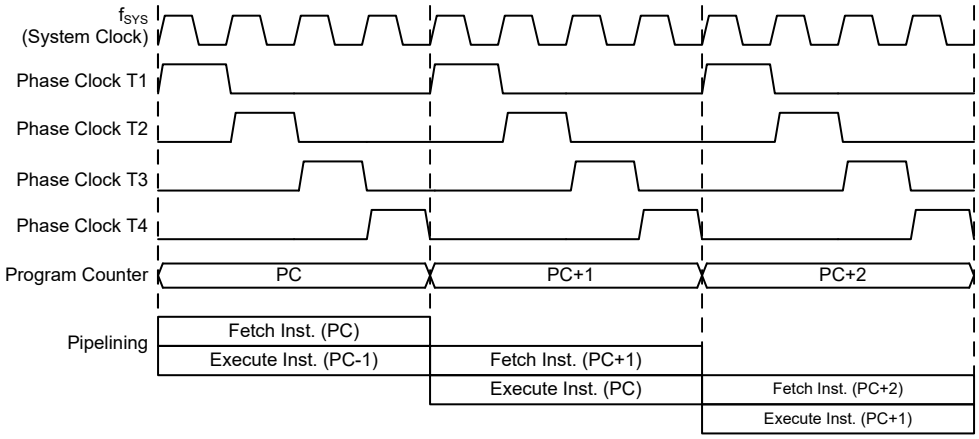
内部系统结构是 Holtek 单片机具有良好性能的主要因素。由于采用 RISC 结构，此单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令外，其它指令都能在一个指令周期内完成。8 位 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、递增、递减和分支等功能，而内部的数据路径则是以通过累加器和 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有较大可靠度和灵活性的 I/O 控制系统时，仅需要少数的外部器件。使得该单片机适用于低成本和批量生产的控制应用。

时序和流水线结构

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

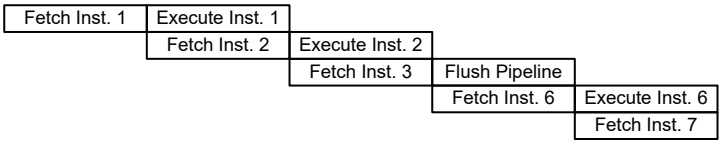
如果指令牵涉到分支，例如跳转或调用等指令，则需要两个指令周期才能完成指令执行。需要一个额外周期的原因是程序先用一个周期取出实际要跳转或调

用的地址，再用另一个周期去实际执行分支动作，因此用户需要特别考虑额外周期的问题，尤其是在执行时间要求较严格的时候。



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

在程序执行期间，程序计数器用来指向下一个要执行的指令地址。除了“JMP”和“CALL”指令需要跳转到一个非连续的程序存储器地址之外，它会在每条指令执行完成以后自动加一。只有较低的 8 位，即所谓的程序计数器低字节寄存器 PCL，可以被用户直接读写。

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

程序计数器	
程序计数器高字节	PCL 寄存器
PC11~PC8	PCL7~PCL0

程序计数器

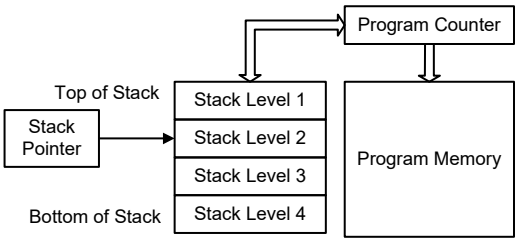
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。程序计数器的低字节可由程序直接进行读取，PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。此单片机有 4 层堆栈，堆栈既不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针 (SP) 加以指示，同样也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入到堆栈中。当子程序或中断响应结束时，返回指令 (RET 或 RETI) 使程序计数器从堆栈中重新得到它以前的值。当一个芯片复位后，堆栈指针将指向堆栈顶部。

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

若堆栈溢出，则首个存入堆栈的程序计数器数据将会丢失。



算术逻辑单元 – ALU

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

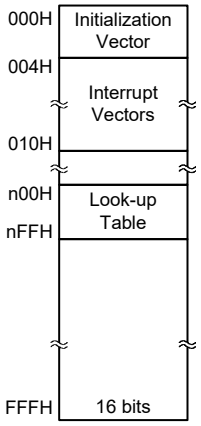
- 算术运算：ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA
- 逻辑运算：AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA
- 移位运算：RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC
- 递增和递减：INCA，INC，DECA，DEC
- 分支判断：JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI

Flash 程序存储器

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

结构

程序存储器的容量为 4K×16 位，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。



程序存储器结构

特殊向量

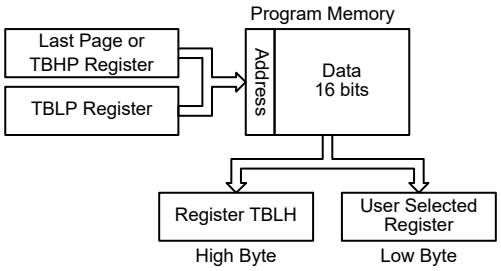
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

查表

程序存储器中的任何地址都可以定义成一个表格，以便储存固定的数据。使用表格时，表格指针必须先行设定，其方式是将表格的地址放在表格指针寄存器 TBLP 和 TBHP 中。这些寄存器定义表格总的地址。

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

下图是查表中寻址 / 数据流程：



查表范例

以下范例说明表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器中。ORG 指令的值“0F00H”指向的地址是 4K 程序存储器中最后一页的起始地址。表格指针低字节寄存器的初始值设为 06H，这可保证从数据表格读取的第一笔数据位于程序存储器地址 0F06H，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRD [m]”指令被使用，则表格指针指向 TBLP 和 TBHP 指定的地址。在这个例子中，表格数据的高字节等于零，而当“TABRD [m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

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

表格读取程序范例

```
tempreg1 db?        ; temporary register #1
tempreg2 db?        ; temporary register #2
:
:
mov a,06h           ; initialize table pointer - note that this address is
                    ; referenced
mov tblp,a          ; to the last page or the page that tbhp pointed
mov a,0Fh           ; initialize high table pointer
mov tbhp,a          ; it is not necessary to set tbhp if executing tabrdl
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer
                    ; data at program memory address "F06H" transferred to
                    ; tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrd tempreg2      ; transfers value in table referenced by table pointer
                    ; data at program memory address "F05H" transferred to
                    ; tempreg2 and TBLH
                    ; in this example the data "1AH" is transferred to
                    ; tempreg1 and data "0FH" to tempreg2
                    ; the value "00H" will be transferred to the high byte
                    ; register TBLH
:
:
org 0F00h           ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

在线烧录 – ICP

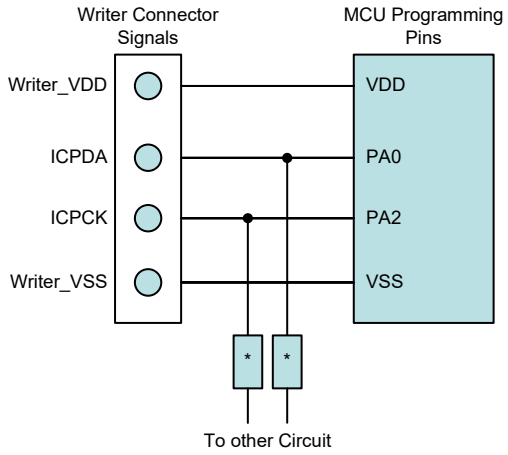
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

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

Holtek 烧录器引脚名称	MCU 在线烧录引脚名称	功能
ICPDA	PA0	串行数据 / 地址烧录
ICPCK	PA2	时钟烧录
VDD	VDD	电源
VSS	VSS	地

单片机内部程序存储器可以通过 4 线的接口在线进行烧录。其中一条用于数据串行下载或上传、一条用于串行时钟、另外两条用于提供电源。芯片在线烧写的详细使用说明超出此文档的描述范围，将由专门的参考文献提供。

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



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

片上调试 – OCDS

EV 芯片 HT45V0063 用于单片机 HT45F0063 仿真。此 EV 芯片提供片上调试功能 (OCDS) 用于开发过程中的单片机调试。除了片上调试功能方面，EV 芯片和实际 MCU 在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 Holtek HT-IDE 开发工具，从而实现 EV 芯片对实际 IC 的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，实际单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 使用手册”文件。

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

数据存储器

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

数据存储器分为两种类型，第一种是特殊功能数据存储器。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二种数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。

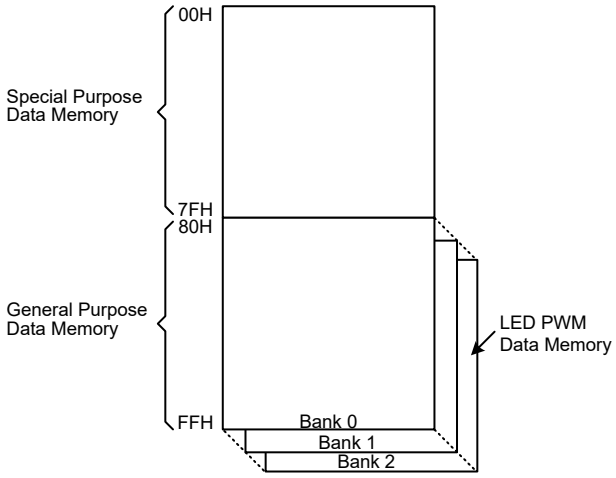
切换不同区域可通过正确配置存储器指针实现。

结构

数据存储器被分为三个 Bank，都可在 8 位的存储器中实现。每个数据存储器 Bank 分为两种类型，即特殊功能数据存储器和通用数据存储器。特殊功能数据存储器的地址范围为 00H~7FH，而通用数据存储器的地址范围为 80H~FFH。LED PWM 数据缓存器位于 Bank 2 的 80H~FFH。

特殊功能数据存储器	通用数据存储器		LED PWM 数据存储器		
所在 Bank	容量	Bank：地址	容量		Bank：地址
0	256×8	0, 1: 80H~FFH	128×8	Buffer_A	2: 80H~BFH
				Buffer_B	2: C0H~FFH

数据存储器概要



数据存储器结构

通用数据存储器

所有的单片机程序需要一个读 / 写的存储区，让临时数据可以被储存和再使用，该 RAM 区域就是通用数据存储器。这个数据存储区可让使用者进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，较大地方便了用户在数据存储器内进行位操作。

特殊功能数据存储器

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

	Bank 0
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	
0CH	SCC
0DH	HIRCC
0EH	WDTC
0FH	RSTFC
10H	INTC0
11H	INTC1
12H	MFI
13H	
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	PB
19H	PBC
1AH	PBPU
1BH	PSCR
1CH	TBC
1DH	PAS0
1EH	PAS1
1FH	PBS0
20H	PBS1
21H	PCS0
22H	IFS
23H	CTMC0
24H	CTMC1
25H	CTMDL
26H	CTMDH
27H	CTMAL
28H	CTMAH
29H	CCS
2AH	IICC0
2BH	IICC1
2CH	IICD
2DH	IICA
2EH	IICTOC
2FH	CCOPU0
30H	CCOPU1
31H	PC
32H	PCC
33H	PCPU
34H	LVRC
35H	BC
36H	CASCON
37H	CASPRE
38H	CASTH
39H	D0CNT
3AH	D1CNT
3BH	PCNT
3CH	RCNT
3DH	CASDNB
3EH	CASD0
3FH	CASD1

	Bank 0
40H	CASD2
41H	CASD3
42H	CASD4
43H	CASD5
44H	CASD6
45H	CASD7
46H	CASD8
47H	CASD9
48H	CASD10
49H	CASD11
4AH	CASD12
4BH	CASD13
4CH	CASD14
4DH	CASD15
4EH	CASD16
4FH	CASD17
50H	CASD18
51H	CASD19
52H	CASD20
53H	CASD21
54H	CASD22
55H	CASD23
56H	CASD24
57H	CASD25
58H	CASD26
59H	CASD27
5AH	CASD28
5BH	CASD29
5CH	CASD30
5DH	CASD31
5EH	CASD32
5FH	CASD33
60H	CASD34
61H	CASD35
62H	CASD36
63H	CASD37
64H	CASD38
65H	CASD39
66H	CASD40
67H	CASD41
68H	CASD42
69H	CASD43
6AH	CASD44
6BH	CASD45
6CH	CASD46
6DH	CASD47
6EH	INTCON0
6FH	INTCON1
70H	COM_PWM
71H	PWM0DATA
72H	PWM1DATA
73H	PWM2DATA
74H	PWM3DATA
75H	PWM4DATA
76H	PWM5DATA
77H	PWM6DATA
78H	PWM7DATA
79H	PWM8DATA
7AH	PWM9DATA
7BH	PWM10DATA
7CH	PWM11DATA
7DH	PWM12DATA
7EH	PWM13DATA
7FH	PWM14DATA

: Unused, read as 00H

特殊功能数据存储器

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1

间接寻址寄存器 IAR0 和 IAR1 的地址虽位于数据存储区，但其并没有实际的物理地址。间接寻址的方法准许使用存储器指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器 IAR0 和 IAR1 上的任何动作，将对存储器指针 MP0 和 MP1 所指定的存储器地址产生对应的读 / 写操作。它们总是成对出现，IAR0 和 MP0 只可以访问 Bank 0，而 IAR1 和 MP1 可以访问所有 Bank。因为这些间接寻址寄存器不是实际存在的，直接读取将返回“00H”的结果，而直接写入此寄存器则不做任何操作。

存储器指针 – MP0, MP1

该单片机提供两个存储器指针，即 MP0 和 MP1。由于这些指针在数据存储器中能像普通的寄存器一般被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由存储器指针所指定的地址。MP0、IAR0 用于访问 Bank 0，而 MP1 和 IAR1 可通过 BP 寄存器的 DMBP1~DMBP0 位访问所有的 Bank。直接寻址仅可以用在 Bank 0 中，Bank 1 和 Bank 2 只可使用 MP1 和 IAR1 进行间接寻址。

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

间接寻址程序举例

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

存储区指针 – BP

数据存储器被分为三个 Bank，即 Bank 0，Bank 1 和 Bank 2。可以通过设置存储区指针 (Bank Pointer) 值来访问不同的数据存储区。BP 指针的 DMBP1~DMBP0 位用于选择数据存储器的 Bank。

复位后，数据存储器会初始化到 Bank 0，但是在空闲或休眠模式下的 WDT 溢出复位，不会改变通用数据存储器的存储区号。数据存储器的直接寻址总是访

问 Bank 0，不影响存储区指针的值。要访问 Bank 0 之外的存储区，则必须要使用间接寻址方式。

- **BP 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	DMBP1	DMBP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**DMBP1 ~ DMBP0**：数据存储区选择位
00：Bank 0
01：Bank 1
10：Bank 2
11：保留

累加器 – ACC

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

程序计数器低字节寄存器 – PCL

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

表格寄存器 – TBLP, TBHP, TBLH

这三个特殊功能寄存器对存储在程序存储器中的表格进行操作。TBLP 和 TBHP 为表格指针，指向表格数据存储的地址。它们的值必须在任何表格读取指令执行前加以设定，由于它们的值可以被如“INC”或“DEC”的指令所改变，这就提供了一种简单的方法对表格数据进行读取。表格读取数据指令执行之后，表格数据高字节存储在 TBLH 中。其中要注意的是，表格数据低字节会被传送到使用者指定的地址。

状态寄存器 – STATUS

这个 8 位的状态寄存器由零标志位 (Z)、进位标志位 (C)、辅助进位标志位 (AC)、溢出标志位 (OV)、暂停标志位 (PDF) 和看门狗定时器溢出标志位 (TO) 组成。这些算术 / 逻辑操作和系统运行标志位是用来记录单片机的运行状态。

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

Z、OV、AC 和 C 标志位通常反映最近运算的状态。

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

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

- **STATUS 寄存器**

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

“x”：未知

Bit 7~6　未定义，读为“0”

Bit 5　**TO**：看门狗溢出标志位
0：系统上电或执行“CLR WDT”或“HALT”指令后
1：看门狗溢出发生

Bit 4　**PDF**：暂停标志位
0：系统上电或执行“CLR WDT”指令后
1：执行“HALT”指令

Bit 3　**OV**：溢出标志位
0：无溢出
1：运算结果高两位的进位状态异或结果为 1

Bit 2　**Z**：零标志位
0：算术或逻辑运算结果不为 0
1：算术或逻辑运算结果为 0

Bit 1　**AC**：辅助进位标志位
0：无辅助进位
1：在加法运算中低四位产生了向高四位进位，或减法运算中低四位不发生从高四位借位

Bit 0　**C**：进位标志位
0：无进位
1：如果在加法运算中结果产生了进位，或在减法运算中结果不发生借位
C 也受循环移位指令的影响。

振荡器

不同的振荡器选择可以让使用者在不同的应用需求中实现更大范围的功能。振荡器的灵活性使得在速度和功耗方面可以达到较佳的优化。振荡器选择和操作是通过应用程序和相应的控制寄存器完成的。

振荡器概述

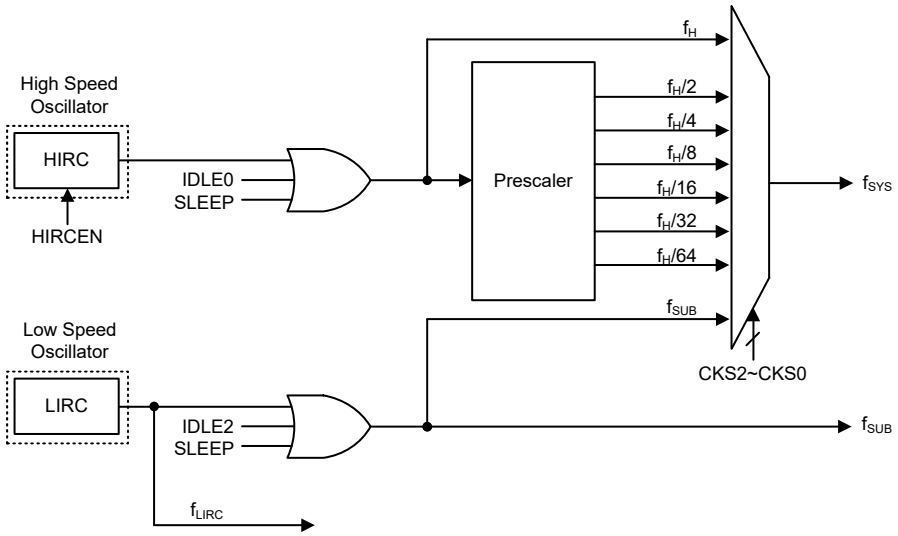
振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。集成的两个内部振荡器不需要任何外围器件。它们提供高速和低速系统振荡器。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

类型	名称	频率
内部高速 RC	HIRC	8MHz
内部低速 RC	LIRC	32kHz

振荡器类型

系统时钟配置

此单片机有两个系统振荡器，包括一个高速振荡器和一个低速振荡器。高速振荡器为内部 8MHz RC 振荡器 HIRC。低速振荡器为内部 32kHz RC 振荡器 LIRC。使用高速或低速振荡器作为系统时钟的选择是通过设置 SCC 寄存器中的 CKS2~CKS0 位决定的，系统时钟可动态选择。



系统时钟配置

内部 RC 振荡器 – HIRC

内部高速 RC 振荡器是一个完全集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有一个 8MHz 的固定频率。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因电源电压、温度以及芯片制成工艺不同的影响较大程度地降低。如果选择了该内部时钟，无需额外的引脚。

内部 32kHz 振荡器 – LIRC

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

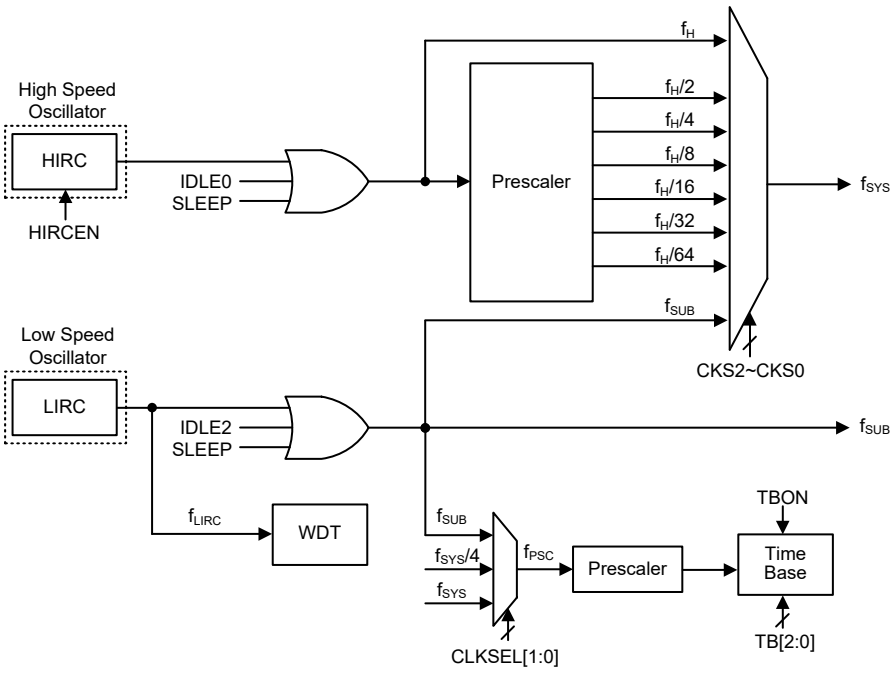
工作模式和系统时钟

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

系统时钟

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

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



单片机时钟配置

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

系统工作模式

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

工作模式	CPU	寄存器设置			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
快速模式	On	x	x	000~110	f_H~f_H/64	On	On	On
低速模式	On	x	x	111	f_{SUB}	On/Off(1)	On	On
空闲模式 0	Off	0	1	000~110	Off	Off	On	On
				111	On			
空闲模式 1	Off	1	1	xxx	On	On	On	On
空闲模式 2	Off	1	0	000~110	On	On	Off	On
				111	Off			
休眠模式	Off	0	0	xxx	Off	Off	Off	On/Off(2)

“x”：无关

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

快速模式

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

低速模式

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

休眠模式

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

空闲模式 0

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

空闲模式 1

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但高速和低速振荡器都会开启以保持一些外围功能继续工作。

空闲模式 2

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 位为高、FSIDEN 位为低时，系统进入空闲模式 2。在空闲模式 2 中，CPU 停止，但高速振荡器会开启以保持一些外围功能继续工作。

控制寄存器

寄存器 SCC 和 HIRCC 用于控制系统时钟和相应的振荡器配置。

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

系统工作模式控制寄存器列表

- **SCC 寄存器**

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

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

Bit 4~2 未定义，读为“0”

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

Bit 0 **FSIDEN**：CPU 关闭时低频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后低速振荡器是被激活还是停止。

- **HIRCC 寄存器**

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

Bit 7~2 未定义，读为“0”

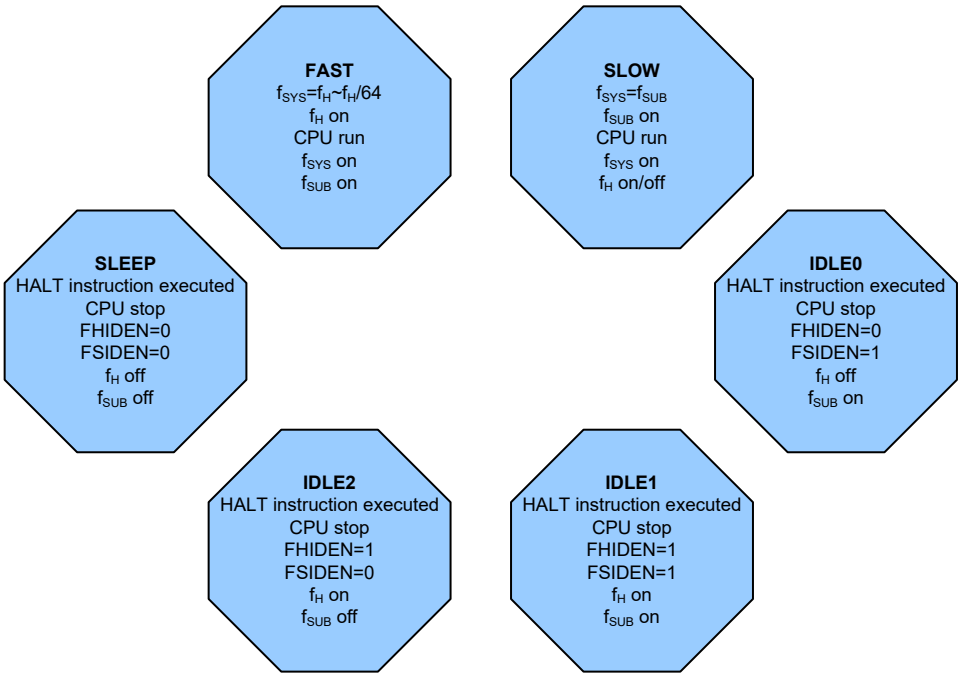
Bit 1 **HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 未稳定
1：HIRC 稳定
此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

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

工作模式切换

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

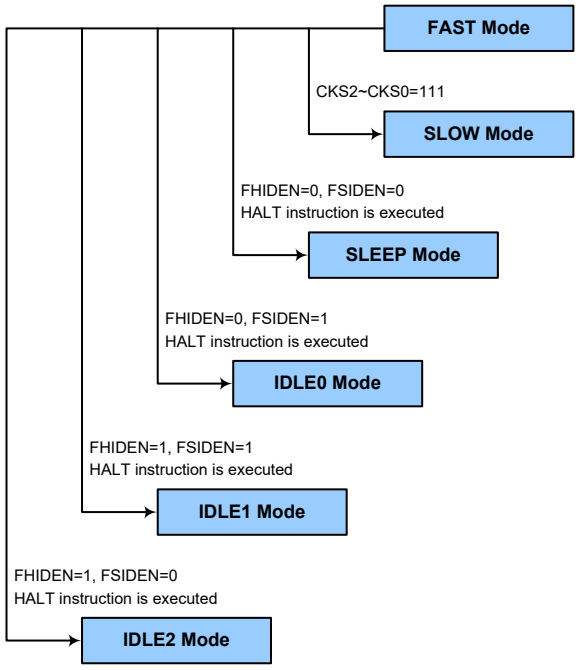
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



快速模式切换到低速模式

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

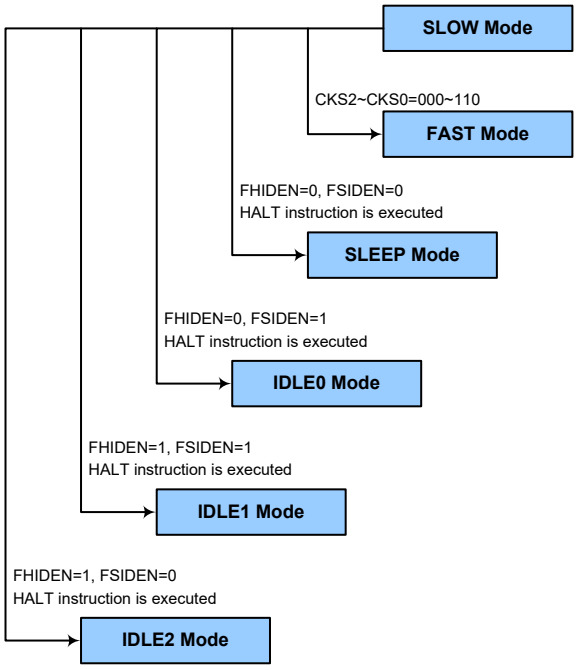
低速模式的时钟源来自 LIRC 振荡器，因此要求这个振荡器在所有模式切换动作发生前稳定下来。



低速模式切换到快速模式

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

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



进入休眠模式

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

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

进入空闲模式 0

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

- f_H 时钟停止运行，应用程序停止在“HALT”指令处，但 f_{SUB} 时钟将继续运行。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。

- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 如果 WDT 功能使能，WDT 将被清零并重新开始计数。如果 WDT 功能除能，WDT 将被清零并停止计数。

进入空闲模式 1

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

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

进入空闲模式 2

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

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

待机电流注意事项

由于单片机进入休眠或空闲模式的主要原因是将 MCU 的电流降低到尽可能低，可能到只有几个微安的级别 (空闲模式 1 和空闲模式 2 除外)，所以如果要将电路的电流进一步降低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这些引脚也必须设为输出或带有上拉电阻的输入。

另外还需注意单片机设为输出的 I/O 引脚上的负载。应将它们设置在有最小拉电流的状态或将它们和其它的 CMOS 输入一样接到没有拉电流的外部电路上。还应注意的是，如果选择 LIRC 振荡器，会导致耗电增加。

在空闲模式 1 和空闲模式 2 中，高速振荡器开启。若外围功能时钟源来自高速振荡器，额外的待机电流也可能会有几百微安。

唤醒

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

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- PA 口下降沿

- 系统中断
- WDT 溢出

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

PA 口中的每个引脚都可以通过 PAWU 寄存器使能下降沿唤醒功能。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。第一种情况是：相关中断除能或是中断使能且堆栈已满，则程序会在“HALT”指令之后继续执行。这种情况下，唤醒系统的中断会等到相关中断使能或有堆栈层可以使用之后才执行。第二种情况是：相关中断使能且堆栈未满，则中断可以马上执行。如果在进入休眠或空闲模式之前中断标志位已经被设置为“1”，则相关中断的唤醒功能将无效。

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟 f_{LIRC}，而 f_{LIRC} 的时钟源由内部 RC 振荡器 LIRC 提供。内部振荡器 LIRC 的频率大约为 32kHz，这个特殊的内部时钟周期随 V_{DD}、温度和制成的不同而变化。看门狗定时器的时钟源可分频为 2^8~2^{15} 以提供更大的溢出周期，分频比由 WDTC 寄存器中的 WS2~WS0 位来决定。

看门狗定时器控制寄存器

WDTC 寄存器用于控制 WDT 功能的使能 / 除能，选择溢出周期以及复位单片机功能。

- **WDTC 寄存器**

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

Bit 7~3 **WE4~WE0**：WDT 软件控制位
10101：除能
01010：使能
其它值：MCU 复位
若因外部环境噪声使 WE4~WE0 值发生改变，则单片机会在一段延迟时间 t_{SRESET} 后产生复位，复位后 RSTFC 寄存器中的 WRF 标志位会被置位。

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

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~3　未定义，读为“0”

Bit 2　**LVRF**：LVR 复位标志位
详见“低电压复位”章节

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

Bit 0　**WRF**：WDTC 控制寄存器软件复位标志
0：未发生
1：发生
当 WDT 控制寄存器软件复位时此位设置为“1”。此位只能通过程序清零。

看门狗定时器操作

当 WDT 溢出时，它产生一个芯片复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗溢出前有策略地清看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。无论什么原因，程序失常跳转到一个未知的地址或进入一个死循环，这个清除指令不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。WDTC 寄存器中的 WE4~WE0 位可用来控制看门狗定时器的使能 / 除能和复位操作。当 WE4~WE0 设置为“10101B”时除能 WDT 功能，而当设置为“01010B”时使能 WDT 功能。如果 WE4~WE0 设置为除 01010B 和 10101B 以外的其它值，将会在一段延迟时间 t_{SRESET} 后复位单片机。上电后 WE4~WE0 的默认值为 01010B。

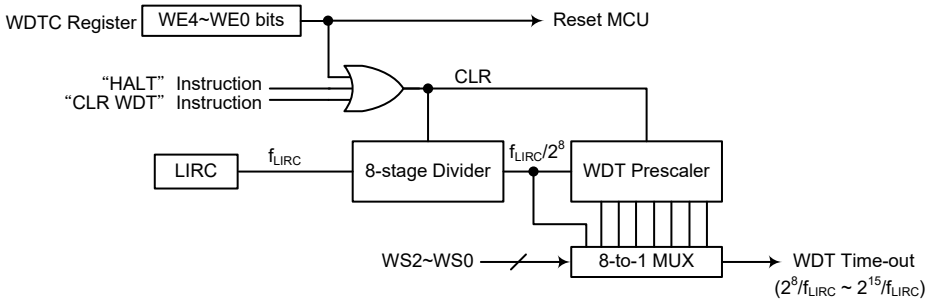
WE4~WE0 位	WDT 功能
10101B	除能
01010B	使能
其它值	MCU 复位

看门狗定时器使能 / 除能控制

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

该单片机只使用一条清看门狗指令“CLR WDT”。因此只要执行“CLR WDT”便清除 WDT。

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



看门狗定时器

复位和初始化

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

另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设置值时，系统会产生 LVR 复位。

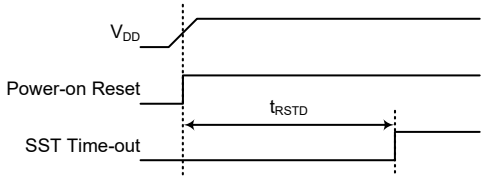
另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机包含几种由内部事件触发的复位方式。

上电复位

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

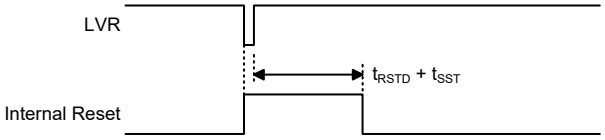


上电复位时序图

低电压复位 – LVR

单片机具有低电压复位电路，用来监测它的电源电压。当电源电压低于预设值时，它将复位单片机。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 RSTFC 寄存器中的 LVRF 标志位置位。所谓有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过 LVD/LVR 电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 LVR 将会忽略它且不会执行复位功能。V_{LVR} 参数值可通过 LVRC 寄存器中的 LVS7~LVS0 位设置。若由于受到干扰 LVS7~LVS0 变为其它值时，需经过一

段延迟时间 t_{SRESET} 后才响应复位。此时 RSTFC 寄存器的 LRF 位被置位。上电后寄存器的值为 01010101B。LVR 会于休眠或空闲时自动除能关闭。



低电压复位时序图

- **LVRC 寄存器**

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

Bit 7~0 **LVS7~LVS0**：LVR 电压选择
01011010：2.1V
10100101：除能
其它值：单片机复位 – 寄存器复位为 POR 值
若低电压情况发生且满足以上定义的低电压复位值，则单片机复位。当低电压状况保持时间大于 t_{LVR} 后，响应复位。此时复位后的寄存器内容保持不变。
除了以上定义的低电压复位值外，其它值也能导致单片机复位。需要经过一段延迟时间 t_{SRESET} 才响应复位。但此时寄存器内容将复位为 POR 值。

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~3 未定义，读为“0”

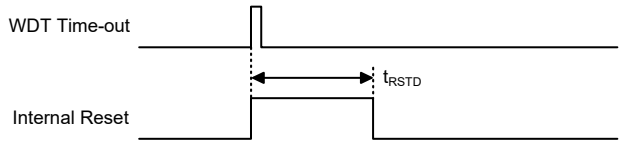
Bit 2 **LVRF**：LVR 复位标志位
0：未发生
1：发生
当特定的低电压复位条件发生时，此位被置为“1”，且只能通过应用程序清零。

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

Bit 0 **WRF**：WDT 控制寄存器软件复位标志位
详见“看门狗定时器控制寄存器”章节

正常运行时看门狗溢出复位

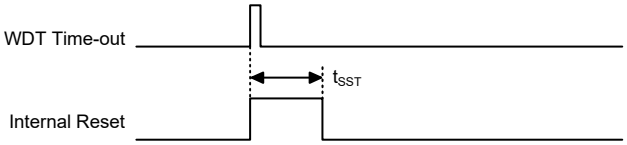
除了看门狗溢出标志位 TO 将被设为“1”之外，快速或低速模式下看门狗溢出复位和上电复位相同。



正常运行时看门狗溢出时序图

休眠或空闲时看门狗溢出复位

休眠或空闲时看门狗溢出复位和其它种类的复位有些不同。除了程序计数器与堆栈指针将被清“0”及 TO 位被设为“1”外，绝大部分的条件保持不变。图中 t_{SST} 的详细说明请参考系统上电时间电气特性。



休眠或空闲时看门狗溢出复位时序图

复位初始状态

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

TO	PDF	复位条件
0	0	上电复位
u	u	快速模式或低速模式时的 LVR 复位
1	u	快速模式或低速模式时的 WDT 溢出复位
1	1	空闲或休眠模式时的 WDT 溢出复位

“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

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

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

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
IAR0	xxxx xxxx	xxxx xxxx	xuuu uuuu
MP0	xxxx xxxx	xxxx xxxx	xuuu uuuu
IAR1	xxxx xxxx	xxxx xxxx	xuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xuuu uuuu
BP	---- --00	---- --00	---- --uu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- uuuu	---- uuuu
STATUS	--00 xxxx	--1u uuuu	--11 uuuu
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
WDTC	0101 0011	0101 0011	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	---0 ---0	---0 ---0	---u ---u
MFI	--00 --00	--00 --00	--uu --uu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
PSCR	---- --00	---- --00	---- --uu
TBC	0--- -000	0--- -000	u--- -uuu
PAS0	00-- 0000	00-- 0000	uu-- uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
PBS1	0000 0000	0000 0000	uuuu uuuu
PCS0	0000 0000	0000 0000	uuuu uuuu
IFS	---- -000	---- -000	---- -uuu
CTMC0	0000 0000	0000 0000	uuuu uuuu
CTMC1	0000 0000	0000 0000	uuuu uuuu
CTMDL	0000 0000	0000 0000	uuuu uuuu
CTMDH	---- --00	---- --00	---- --uu
CTMAL	0000 0000	0000 0000	uuuu uuuu
CTMAH	---- --00	---- --00	---- --uu
CCS	0010 1010	0010 1010	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
IICC0	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 0000	0000 0000	uuuu uuuu
IICTOC	0000 0000	0000 0000	uuuu uuuu
CCOPU0	0000 0000	0000 0000	uuuu uuuu
CCOPU1	-000 0000	-000 0000	-uuu uuuu
PC	---- 1111	---- 1111	---- uuuu
PCC	---- 1111	---- 1111	---- uuuu
PCPU	---- 0000	---- 0000	---- uuuu
LVRC	0101 1010	0101 1010	uuuu uuuu
BC	1111 1111	1111 1111	uuuu uuuu
CASCON	-100 0000	-100 0000	-uuu uuuu
CASPRE	---- -000	---- -000	---- -uuu
CASTH	---0 0111	---0 0111	---u uuuu
D0CNT	---0 0100	---0 0100	---u uuuu
D1CNT	---0 1010	---0 1010	---u uuuu
PCNT	0001 1000	0001 1000	uuuu uuuu
RCNT	1000 0000	1000 0000	uuuu uuuu
CASDNB	---- 0011	---- 0011	---- uuuu
CASD0	0000 0000	0000 0000	uuuu uuuu
CASD1	0000 0000	0000 0000	uuuu uuuu
CASD2	0000 0000	0000 0000	uuuu uuuu
CASD3	0000 0000	0000 0000	uuuu uuuu
CASD4	0000 0000	0000 0000	uuuu uuuu
CASD5	0000 0000	0000 0000	uuuu uuuu
CASD6	0000 0000	0000 0000	uuuu uuuu
CASD7	0000 0000	0000 0000	uuuu uuuu
CASD8	0000 0000	0000 0000	uuuu uuuu
CASD9	0000 0000	0000 0000	uuuu uuuu
CASD10	0000 0000	0000 0000	uuuu uuuu
CASD11	0000 0000	0000 0000	uuuu uuuu
CASD12	0000 0000	0000 0000	uuuu uuuu
CASD13	0000 0000	0000 0000	uuuu uuuu
CASD14	0000 0000	0000 0000	uuuu uuuu
CASD15	0000 0000	0000 0000	uuuu uuuu
CASD16	0000 0000	0000 0000	uuuu uuuu
CASD17	0000 0000	0000 0000	uuuu uuuu
CASD18	0000 0000	0000 0000	uuuu uuuu
CASD19	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
CASD20	0000 0000	0000 0000	uuuu uuuu
CASD21	0000 0000	0000 0000	uuuu uuuu
CASD22	0000 0000	0000 0000	uuuu uuuu
CASD23	0000 0000	0000 0000	uuuu uuuu
CASD24	0000 0000	0000 0000	uuuu uuuu
CASD25	0000 0000	0000 0000	uuuu uuuu
CASD26	0000 0000	0000 0000	uuuu uuuu
CASD27	0000 0000	0000 0000	uuuu uuuu
CASD28	0000 0000	0000 0000	uuuu uuuu
CASD29	0000 0000	0000 0000	uuuu uuuu
CASD30	0000 0000	0000 0000	uuuu uuuu
CASD31	0000 0000	0000 0000	uuuu uuuu
CASD32	0000 0000	0000 0000	uuuu uuuu
CASD33	0000 0000	0000 0000	uuuu uuuu
CASD34	0000 0000	0000 0000	uuuu uuuu
CASD35	0000 0000	0000 0000	uuuu uuuu
CASD36	0000 0000	0000 0000	uuuu uuuu
CASD37	0000 0000	0000 0000	uuuu uuuu
CASD38	0000 0000	0000 0000	uuuu uuuu
CASD39	0000 0000	0000 0000	uuuu uuuu
CASD40	0000 0000	0000 0000	uuuu uuuu
CASD41	0000 0000	0000 0000	uuuu uuuu
CASD42	0000 0000	0000 0000	uuuu uuuu
CASD43	0000 0000	0000 0000	uuuu uuuu
CASD44	0000 0000	0000 0000	uuuu uuuu
CASD45	0000 0000	0000 0000	uuuu uuuu
CASD46	0000 0000	0000 0000	uuuu uuuu
CASD47	0000 0000	0000 0000	uuuu uuuu
INTCON0	0000 0000	0000 0000	uuuu uuuu
INTCON1	---0 ---0	---0 ---0	---u ---u
COM_PWM	0-00 0000	0-00 0000	u-uu uuuu
PWM0DATA	0000 0000	0000 0000	uuuu uuuu
PWM1DATA	0000 0000	0000 0000	uuuu uuuu
PWM2DATA	0000 0000	0000 0000	uuuu uuuu
PWM3DATA	0000 0000	0000 0000	uuuu uuuu
PWM4DATA	0000 0000	0000 0000	uuuu uuuu
PWM5DATA	0000 0000	0000 0000	uuuu uuuu
PWM6DATA	0000 0000	0000 0000	uuuu uuuu
PWM7DATA	0000 0000	0000 0000	uuuu uuuu
PWM8DATA	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
PWM9DATA	0000 0000	0000 0000	uuuu uuuu
PWM10DATA	0000 0000	0000 0000	uuuu uuuu
PWM11DATA	0000 0000	0000 0000	uuuu uuuu
PWM12DATA	0000 0000	0000 0000	uuuu uuuu
PWM13DATA	0000 0000	0000 0000	uuuu uuuu
PWM14DATA	0000 0000	0000 0000	uuuu uuuu

注：“u”表示不改变
“x”表示未知
“-”表示未定义

输入 / 输出端口

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

该单片机提供 PA~PC 双向输入 / 输出口。这些寄存器在数据存储器有特定的地址。所有 I/O 口用于输入输出操作。作为输入操作，输入引脚无锁存功能，也就是说输入数据必须在执行“MOV A, [m]”，T2 的上升沿准备好，m 为端口地址。对于输出操作，所有数据都是被锁存的，且保持不变直到输出锁存被重写。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	—	—	PC3	PC2	PC1	PC0
PCC	—	—	—	—	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	—	—	PCPU3	PCPU2	PCPU1	PCPU0

“—”：未定义，读为“0”

I/O 逻辑功能寄存器列表

上拉电阻

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

需要注意的是，只有当 I/O 引脚设为数字输入或 NMOS 输出时，上拉功能才会受相应的上拉控制寄存器控制开启，其它状态下上拉功能不可用。

- **PxPU 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PxPU7	PxPU6	PxPU5	PxPU4	PxPU3	PxPU2	PxPU1	PxPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

PxPUn：I/O Port x 引脚上拉功能控制位
0：除能
1：使能
PxPUn 位用于控制对应引脚的上拉功能。此处的“x”可为 A、B 或 C。但是，每个 I/O 端口实际有效位可能不同。

PA 口唤醒

当使用暂停指令“HALT”迫使单片机进入休眠或空闲模式，单片机的系统时钟将会停止以降低功耗，此功能对于电池及低功耗应用很重要。唤醒单片机有很多种方法，其中之一就是使 PA 口的其中一个引脚从高电平转为低电平。这个功能特别适合于通过外部开关来唤醒的应用。PA 口的每个引脚可以通过设置 PAWU 寄存器来单独选择是否具有唤醒功能。

需要注意的是，只有当引脚被设置为通用 I/O 功能且 MCU 处于空闲或休眠模式时，唤醒功能才会受 PAWU 控制开启，其它状态下唤醒功能不可用。

- **PAWU 寄存器**

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

Bit 7~0 **PAWU7~PAWU0**：PA7~PA0 唤醒功能控制位
0：除能
1：使能

输入 / 输出端口控制寄存器

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

- **PxC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC6	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

PxCn：I/O Port x 引脚输入 / 输出类型选择位
0：输出
1：输入
PxCn 位用于选择对应引脚的输入 / 输出类型。此处的“x”可为 A、B 或 C。但是，每个 I/O 端口实际有效位可能不同。

引脚共用功能

引脚的多功能可以增加单片机应用的灵活性。有限的引脚个数将会限制设计者，而引脚的多功能将会解决很多此类问题。此外，这些引脚功能可以通过一系列寄存器进行设定。

引脚共用功能选择寄存器

封装中有限的引脚个数会对某些单片机功能造成影响。然而，引脚功能共用和引脚功能选择，使得小封装单片机具有更多不同的功能。单片机包含端口“x”输出功能选择寄存器“n”，即 PxSn，和输入功能选择寄存器，即 IFS，这些寄存器可以用来选择共用引脚的特定功能。

当要使用引脚上的输入功能，对应的输入和输出功能要进行合理设定。例如，若要使用级联式收发器接口，对应的共用功能要通过 PxSn 寄存器设置为级联式收发器接口功能且还需通过 IFS 寄存器合理选择级联式收发器接口信号输入源。

需要特别注意的是，要确保所需的引脚共用功能被正确地选择和取消。对于大部分的引脚功能，要选择所需的引脚共用功能，首先应通过相应的引脚共用控制寄存器正确地选择该功能，然后再配置相应的外围功能设置以使能外围功能。但是，在设置相关引脚控制字段时，一些数字输入引脚如 INT 等，与对应的通用 I/O 口共用同一个引脚共用设置选项。要选择这些引脚功能，除了上述的必要的引脚共用控制和外围功能设置外，还必须将其对应的端口控制寄存器位设置为输入。要正确地取消引脚共用功能，首先应除能外围功能，然后再修改相应的引脚共用控制寄存器以选择其它的共用功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	—	—	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
IFS	—	—	—	—	—	IFS2	IFS1	IFS0

引脚共用功能选择寄存器列表

- **PAS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	—	—	PAS03	PAS02	PAS01	PAS00
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	0	0

Bit 7~6 **PAS07~PAS06**：PA3 引脚共用功能选择
00：PA3
01：PA3
10：PA3
11：CCO1

Bit 5~4 未定义，读为“0”

Bit 3~2 **PAS03~PAS02**：PA1 引脚共用功能选择
00：PA1
01：PA1

10：PA1
11：CCO0

Bit 1~0 **PAS01~PAS00**：PA0 引脚共用功能选择
00：PA0
01：PA0
10：SDA
11：AD

- **PAS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS17~PAS16**：PA7 引脚共用功能选择
00：PA7
01：PA7
10：PA7
11：CCO5

Bit 5~4 **PAS15~PAS14**：PA6 引脚共用功能选择
00：PA6
01：PA6
10：PA6
11：CCO4

Bit 3~2 **PAS13~PAS12**：PA5 引脚共用功能选择
00：PA5
01：PA5
10：PA5
11：CCO3

Bit 1~0 **PAS11~PAS10**：PA4 引脚共用功能选择
00：PA4
01：PA4
10：PA4
11：CCO2

- **PBS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PBS07~PBS06**：PB3 引脚共用功能选择
00：PB3
01：PB3
10：PB3
11：CCO9

Bit 5~4 **PBS05~PBS04**：PB2 引脚共用功能选择
00：PB2
01：PB2
10：PB2
11：CCO8

Bit 3~2 **PBS03~PBS02**：PB1 引脚共用功能选择
00：PB1
01：PB1
10：SDA
11：CCO7

Bit 1~0 **PBS01~PBS00**：PB0 引脚共用功能选择
00：PB0
01：PB0
10：SCL
11：CCO6

- **PBS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PBS17~PBS16**：PB7 引脚共用功能选择
00：PB7
01：PB7
10：COM1
11：CCO13

Bit 5~4 **PBS15~PBS14**：PB6 引脚共用功能选择
00：PB6
01：PB6
10：COM0
11：CCO12

Bit 3~2 **PBS13~PBS12**：PB5 引脚共用功能选择
00：PB5
01：PB5
10：CASDI
11：CCO11

Bit 1~0 **PBS11~PBS10**：PB4 引脚共用功能选择
00：PB4
01：PB4
10：CASDO
11：CCO10

- **PCS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PCS07~PCS06**：PC3 引脚共用功能选择
00：PC3
01：PC3
10：SDA
11：CASDI

Bit 5~4 **PCS05~PCS04**：PC2 引脚共用功能选择
00：PC2
01：PC2
10：SCL
11：CASDO

Bit 3~2 **PCS03~PCS02**：PC1 引脚共用功能选择
00：PC1
01：PC1
10：PC1
11：COM3

Bit 1~0 **PCS01~PCS00**：PC0 引脚共用功能选择
00：PC0
01：PC0
10：COM2
11：CCO14

- **IFS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	IFS2	IFS1	IFS0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”

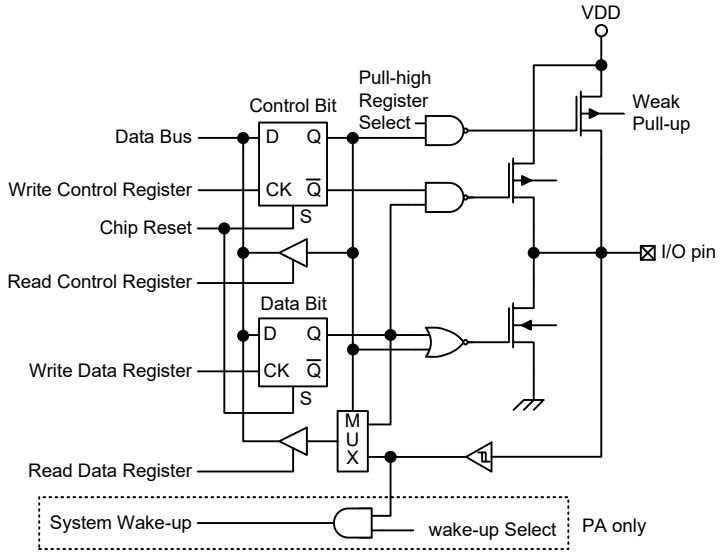
Bit 2 **IFS2**：SDA 输入源引脚选择
0：PC3
1：PB1

Bit 1 **IFS1**：SCL 输入源引脚选择
0：PC2
1：PB0

Bit 0 **IFS0**：CASDI 输入源引脚选择
0：PC3
1：PB5

输入 / 输出引脚结构

下图为输入 / 输出引脚逻辑功能的内部结构图。输入 / 输出引脚的准确逻辑结构图可能与此图不同，这里只是为了方便对 I/O 引脚逻辑功能的理解提供一个参考。由于存在诸多的引脚共用结构，在此不方便提供所有类型引脚功能结构图。



逻辑功能输入 / 输出端口结构

编程注意事项

在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入 / 输出数据及端口控制寄存器都将被设为逻辑高。所有输入 / 输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器将某些引脚设定为输出状态，这些输出引脚会有初始高电平输出，除非端口数据寄存器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对应的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读 - 修改 - 写的操作。单片机需要先读入整个端口上的数据，修改个别的位，然后重新把这些数据写入到输出端口。

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

定时器模块 – TM

控制和测量时间在任何单片机中都是一个很重要的部分。该单片机提供一个定时器模块 (简称 TM)，来实现和时间有关的功能。定时器模块是包括多种操作的定时单元，提供的操作有：定时 / 计数器，比较匹配输出以及输出 PWM 信号等功能。该定时器模块有两个独立中断。TM 外加的输入输出引脚，扩大了定时器的灵活性，便于用户使用。

简介

该单片机包含一个简易型 TM，记为 CTM。本章介绍简易型 TM 的一些特性，更多详细资料见后面章节。CTM 的基本功能见下表。

功能	CTM
定时 / 计数器	√
比较匹配输出	√
PWM 输出	√
PWM 对齐方式	边沿对齐
PWM 调节周期 & 占空比	占空比或周期

TM 功能概要

TM 操作

简易型 TM 提供从简单的定时操作到 PWM 信号产生等多种功能。理解 TM 操作的关键是比较 TM 内独立运行的计数器的值与内部比较器的预置值。当计数器的值与比较器的预置值相同时，则比较匹配，TM 中断信号产生，清零计数器并改变 TM 输出引脚的状态。用户选择内部时钟来驱动内部 TM 计数器。

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 CTM 控制寄存器的 CTCK2~CTCK0 位，选择所需的时钟源。该时钟源来自系统时钟 f_{SYS} 的分频比或内部高速时钟 f_H 或 f_{SUB} 时钟源。

TM 中断

简易型 TM 拥有两个内部中断，分别是内部比较器 A 或比较器 P，当比较匹配发生时产生 TM 中断。当 TM 中断产生时，计数器清零并改变 TM 输出信号的状态。

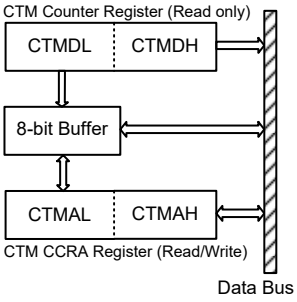
TM 输出信号

该 TM 有两个输出信号 CTP 和 CTPB。CTPB 是 CTP 输出的反相信号。当 TM 工作在比较匹配输出模式且比较匹配发生时，这些信号会由 TM 控制切换到高电平或低电平或翻转。外部 CTP 或 CTPB 输出也被 TM 用来产生 PWM 输出波形。

编程注意事项

TM 计数寄存器和捕捉 / 比较寄存器 CCRA 为 10-bit 寄存器，含有低字节和高字节结构。高字节可直接访问，低字节则仅能通过一个内部 8-bit 的缓存器进行访问。值得注意的是 8-bit 缓存器的存取数据及相关低字节的读写操作仅在其相应的高字节读取操作执行时发生。

CCRA 寄存器访问方式如下图所示，读写这些成对的寄存器需通过特殊的方式。建议使用“MOV”指令按照以下步骤访问 CCRA 低字节寄存器，即 CTMAL，否则可能导致无法预期的结果。

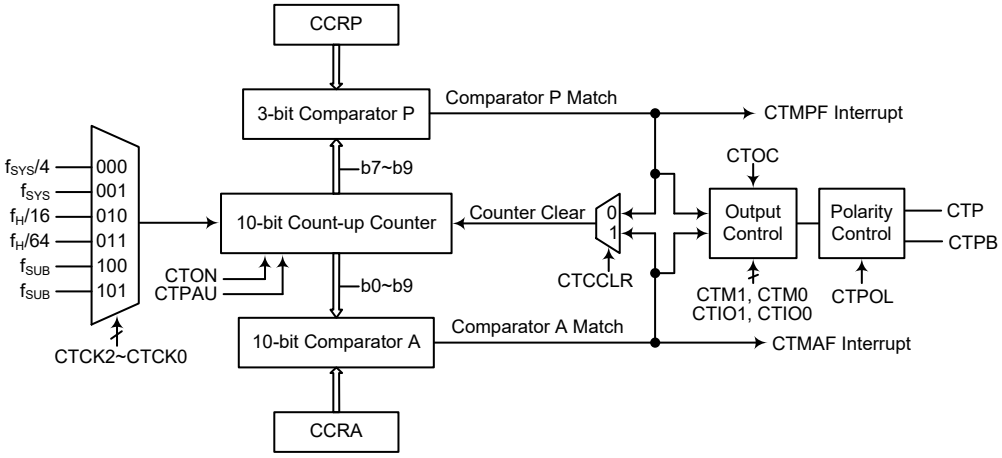


读写流程如下步骤所示：

- 写数据至 CCRA
  - ♦ 步骤 1. 写数据至低字节寄存器 CTMAL
    – 注意，此时数据仅写入 8-bit 缓存器。
  - ♦ 步骤 2. 写数据至高字节寄存器 CTMAH
    – 注意，此时数据直接写入高字节寄存器，同时锁存在 8-bit 缓存器中的数据写入低字节寄存器。
- 由计数器寄存器和 CCRA 中读取数据
  - ♦ 步骤 1. 由高字节寄存器 CTMDH 或 CTMAH 读取数据
    – 注意，此时高字节寄存器中的数据直接读取，同时由低字节寄存器读取的数据锁存至 8-bit 缓存器中。
  - ♦ 步骤 2. 由低字节寄存器 CTMDL 或 CTMAL 读取数据
    – 注意，此时读取 8-bit 缓存器中的数据。

简易型 TM – CTM

简易型 TM 包括 3 种工作模式，即比较匹配输出、定时 / 事件计数器和 PWM 输出模式。



注：CTPB 是 CTP 输出的反相信号，均未连接至外部封装引脚。

10-bit 简易型 TM 方框图

简易型 TM 操作

简易型 TM 核心是一个由用户选择的内部时钟源驱动的 10 位向上计数器，它还包括两个内部比较器即比较器 A 和比较器 P。这两个比较器将计数器的值与 CCRA 和 CCRP 寄存器中的值进行比较。CCRP 是 3-bit 的，与计数器的高 3 位比较；而 CCRA 是 10-bit 的，与计数器的所有位比较。

通过应用程序改变 10 位计数器值的唯一方法是使 CTON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况会产生 CTM 中断信号。简易型 TM 可工作在不同的模式，由不同的时钟源驱动。所有工作模式的设定都是通过设置相关寄存器来实现的。

简易型 TM 寄存器介绍

简易型 TM 的所有操作由一系列寄存器控制。一对只读寄存器用来存放 10 位计数器的值，两对读 / 写寄存器存放 10 位 CCRA 的值。剩下两个控制寄存器用来设置不同的操作和控制模式，以及 3 位 CCRP 的值。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CTMC0	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
CTMC1	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
CTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMDH	—	—	—	—	—	—	D9	D8
CTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMAH	—	—	—	—	—	—	D9	D8

10-bit 简易型 TM 寄存器列表

- **CTMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CTPAU**：CTM 计数器暂停控制位
0：运行
1：暂停
通过设置此位为高可使计数器暂停，清零此位恢复正常计数器操作。当处于暂停条件时，CTM 保持上电状态并继续耗电。当此位由低到高转变时，计数器将保留其剩余值，直到此位再次改变为低电平，并从此值开始继续计数。

Bit 6~4 **CTCK2~CTCK0**：选择 CTM 计数时钟位
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{SUB}
101：f_{SUB}
110：未定义，不能被选择
111：未定义，不能被选择
此三位用于选择 CTM 的时钟源。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的内部时钟源，细节方面请参考振荡器章节。

Bit 3 **CTON**：CTM 计数器 On/Off 控制位
0：Off
1：On
此位控制 CTM 的总开关功能。设置此位为高则使能计数器使其运行，清零此位则除能 CTM。清零此位将停止计数器并关闭 CTM 减少耗电。当此位经由低到高转变时，内部计数器将复位清零；当此位经由高到低转换时，内部计数器将保持其剩余值，直到此位再次改变为高电平。
若 CTM 处于比较匹配输出模式或 PWM 输出模式时，当 CTON 位经由低到高转换时，CTM 输出将复位至 CTOC 位指定的初始值。

Bit 2~0 **CTRP2~CTRP0**：CTM CCRP 3-bit 寄存器，与 CTM 计数器 bit 9 ~ bit 7 比较
比较器 P 匹配周期
000：1024 个 CTM 时钟周期
001：128 个 CTM 时钟周期
010：256 个 CTM 时钟周期
011：384 个 CTM 时钟周期
100：512 个 CTM 时钟周期
101：640 个 CTM 时钟周期
110：768 个 CTM 时钟周期
111：896 个 CTM 时钟周期
此三位设定内部 CCRP 3-bit 寄存器的值，然后与内部计数器的高三位进行比较。如果 CTCCLR 位设定为 0 时，此比较结果可用于清除内部计数器。CTCCLR 位设为低，内部计数器在比较器 P 比较匹配发生时被重置；由于 CCRP 只与计数器高三位比较，比较结果是 128 时钟周期的倍数。CCRP 被清零时，实际上会使得计数器在最大值溢出。

- **CTMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **CTM1~CTM0**：选择 CTM 工作模式位
00：比较匹配输出模式
01：未定义
10：PWM 输出模式
11：定时 / 计数器模式
这两位设置 CTM 需要的工作模式。为了确保操作可靠，CTM 应在 CTM1 和 CTM0 位有任何改变前先关掉。在定时 / 计数器模式，CTM 输出状态为未知。

Bit 5~4 **CTIO1~CTIO0**：选择 CTP 输出功能位
比较匹配输出模式
00：无变化
01：输出低
10：输出高
11：输出翻转
PWM 输出模式
00：PWM 输出无效状态
01：PWM 输出有效状态
10：PWM 输出
11：未定义
定时 / 计数器模式
未使用
此两位用于决定在一定条件达到时 CTM 输出如何改变状态。这两位值的选择取决于 CTM 运行在哪种模式下。
在比较匹配输出模式下，CTIO1 和 CTIO0 位决定当从比较器 A 比较匹配输出发生时 CTM 输出如何改变状态。当从比较器 A 比较匹配输出发生时 CTM 输出能设为切换高、切换低或翻转当前状态。若此两位同时为 0 时，这个输出将不会改变。CTM 输出的初始值通过 CTMC1 寄存器的 CTOC 位设置取得。注意，由 CTIO1 和 CTIO0 位得到的输出电平必须与通过 CTOC 位设置的初始值不同，否则当比较匹配发生时，CTM 输出将不会发生变化。在 CTM 输出改变状态后，通过 CTON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，CTIO1 和 CTIO0 用于决定比较匹配条件发生时怎样改变 CTM 输出的状态。PWM 输出功能通过这两位的变化进行更新。仅在 CTM 关闭时改变 CTIO1 和 CTIO0 位的值是很有必要的。若在 CTM 运行时改变 CTIO1 和 CTIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **CTOC**：CTM CTP 输出控制位
比较匹配输出模式
0：初始低
1：初始高
PWM 输出模式
0：低有效
1：高有效
这是 CTM 输出控制位。它取决于 CTM 此时正运行于比较匹配输出模式还是 PWM 输出模式。若 CTM 处于定时 / 计数器模式，则其不受影响。在比较匹配输出模式时，比较匹配发生前其决定 CTM 输出的逻辑电平值。在 PWM 输出模式时，其决定 PWM 信号是高有效还是低有效。

Bit 2 **CTPOL**：CTP 输出极性控制位
0：同相
1：反相
此位控制 CTP 输出的极性。此位为高时 CTM 输出反相，为低时 CTM 输出同相。若 CTM 处于定时 / 计数器模式时其不受影响。

Bit 1 **CTDPX**：CTM PWM 周期 / 占空比控制位
0：CCRP – 周期；CCRA – 占空比
1：CCRP – 占空比；CCRA – 周期
此位决定 CCRA 与 CCRP 寄存器哪个被用于 PWM 波形的周期和占空比控制。

Bit 0 **CTCCLR**：选择 CTM 计数器清零条件位
0：CTM 比较器 P 匹配
1：CTM 比较器 A 匹配
此位用于选择清除计数器的方法。周期型 CTM 包括两个比较器 – 比较器 A 和比较器 P，两者都可以用作清除内部计数器。CTCCLR 位设为高，计数器在比较器 A 比较匹配发生时被清除；此位设为低，计数器在比较器 P 比较匹配发生或计数器溢出时被清除。计数器溢出清除的方法仅在 CCRP 被清除为 0 时才能生效。CTCCLR 位在 PWM 输出模式时未使用。

- **CTMDL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：CTM 计数器低字节寄存器 bit 7 ~ bit 0
CTM 10-bit 计数器 bit 7 ~ bit 0

- **CTMDH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **D9~D8**：CTM 计数器高字节寄存器 bit 1 ~ bit 0
CTM 10-bit 计数器 bit 9 ~ bit 8

- **CTMAL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：CTM CCRA 低字节寄存器 bit 7 ~ bit 0
CTM 10-bit CCRA bit 7 ~ bit 0

- **CTMAH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **D9~D8**：CTM CCRA 高字节寄存器 bit 1 ~ bit 0
CTM 10-bit CCRA bit 9 ~ bit 8

简易型 TM 工作模式

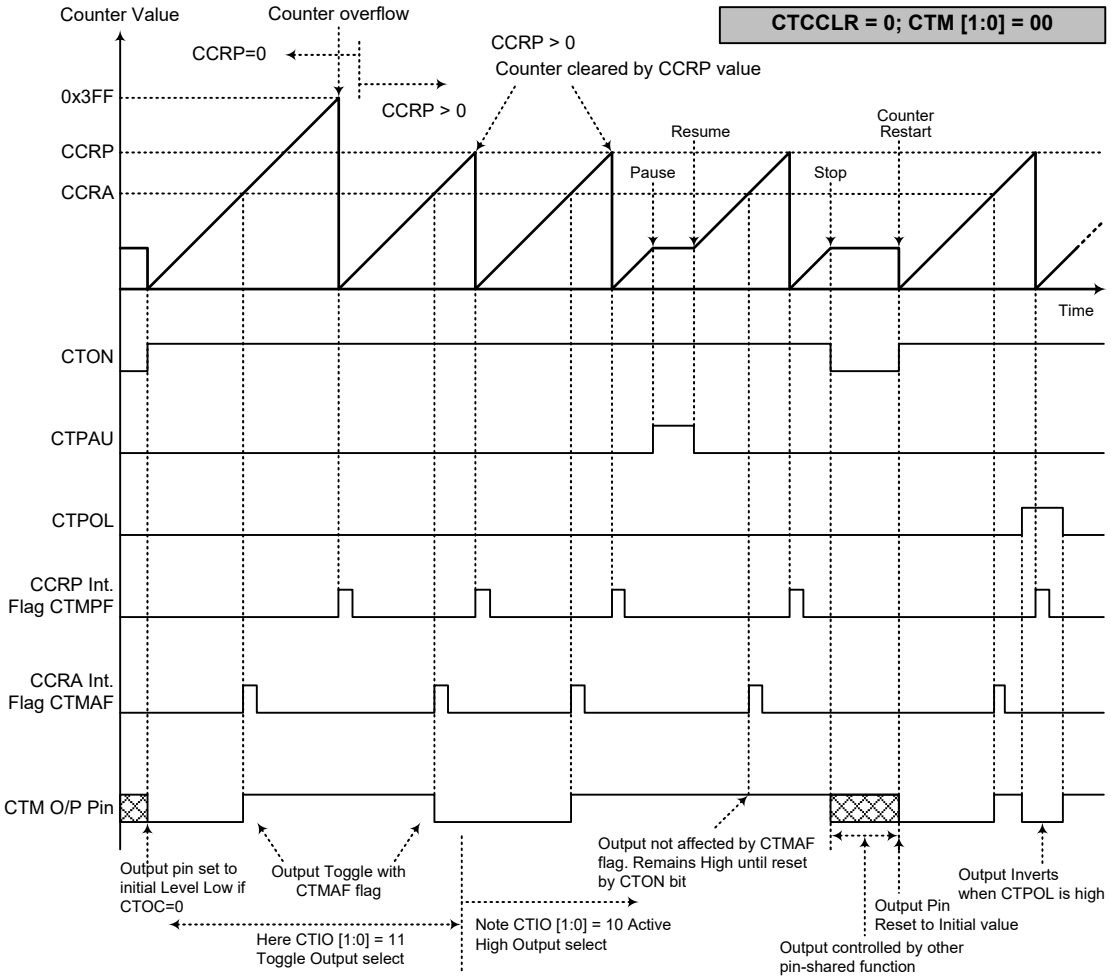
简易型 TM 有三种工作模式，即比较匹配输出模式、PWM 输出模式或定时 / 计数器模式。通过设置 CTMC1 寄存器的 CTM1 和 CTM0 位选择任意模式。

比较匹配输出模式

为使 CTM 工作在此模式，CTMC1 寄存器的 CTM1 和 CTM0 位需要设置为“00”。当工作在该模式，一旦计数器使能并开始计数，有三种方法来清零，分别是：计数器溢出，比较器 A 比较匹配发生和比较器 P 比较匹配发生。当 CTCCLR 位为低，有两种方法清除计数器。一种是比较器 P 比较匹配发生，另一种是 CCRP 所有位设置为零并使得计数器溢出。此时，比较器 A 和比较器 P 的请求标志位 CTMAF 和 CTMPF 将分别置起。

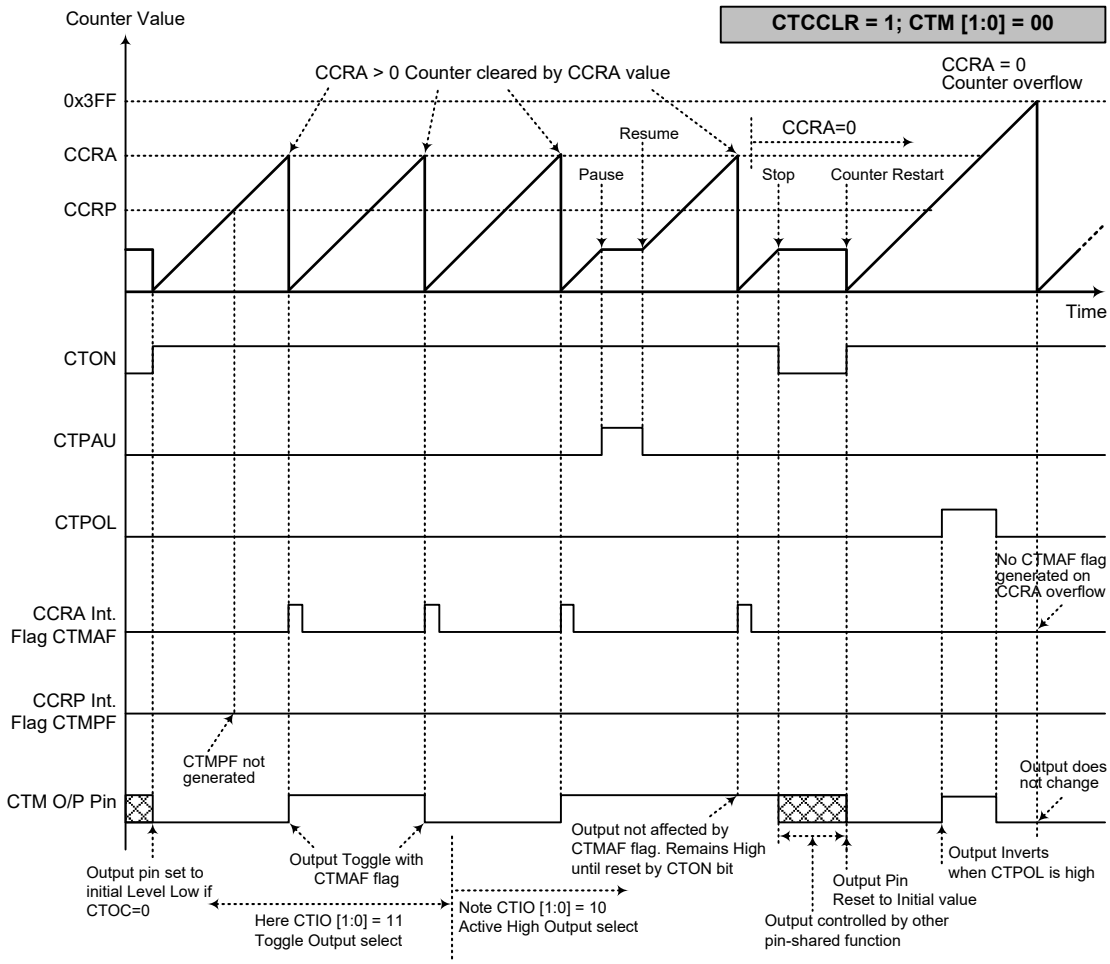
如果 CTMC1 寄存器的 CTCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 CTMAF 中断请求标志产生。所以当 CTCCLR 为高时，不会产生 CTMPF 中断请求标志。在比较匹配输出模式中，CCRA 寄存器值不能设为“0”。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 CTMAF 请求标志。

正如该模式名所言，当比较匹配发生后，CTM 输出状态改变。当比较器 A 比较匹配发生后 CTMAF 中断请求标志产生时，CTM 输出状态改变。比较器 P 比较匹配发生时产生的 CTMPF 标志不影响 CTM 输出。CTM 输出状态改变方式由 CTMC1 寄存器中 CTIO1 和 CTIO0 位决定。当比较器 A 比较匹配发生时，CTIO1 和 CTIO0 位决定 CTM 输出高，低或翻转当前状态。在 CTON 位由低到高电平的变化后，CTM 输出初始状态为 CTOC 位所指定的电平。注意，若 CTIO1 和 CTIO0 位同时为 0 时，CTM 输出不变。



比较器匹配输出模式 – CTCCLR=0

注：1. CTCCLR=0，比较器 P 匹配将清除计数器
2. CTM 输出仅由 CTMAF 标志位控制
3. 在 CTON 上升沿 CTM 输出复位至初始值



比较器匹配输出模式 – CTCCLR=1

注：1. CTCCLR=1，比较器 A 匹配将清除计数器
2. CTM 输出仅由 CTMAF 标志位控制
3. 在 CTON 上升沿 CTM 输出复位至初始值
4. 当 CTCCLR=1 时，不会产生 CTMPF 标志位

定时 / 计数器模式

为使 CTM 工作在此模式，CTMC1 寄存器的 CTM1 和 CTM0 位需要设置为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 CTM 输出未使用。因此，比较匹配输出模式中的描述和时序图可以适用于此功能。该模式中未使用的 CTM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 CTM 工作在此模式，CTMC1 寄存器的 CTM1 和 CTM0 位需要设置为“10”，且 CTIO1 和 CTIO0 位也需要设置为“10”。CTM 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 CTM 输出提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 输出模式中，CTCCLR 位对 PWM 周期无影响。CCRP 和 CCRA 寄存器都用于控制 PWM 波形。一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 CTMC1 寄存器的 CTDPX 位。PWM 波形的周期和占空比由 CCRP 和 CCRA 寄存器的值控制。

当比较器 A 或比较器 P 比较匹配发生时，CCRA 和 CCRP 中断标志位分别产生。CTMC1 寄存器的 CTOC 位选择 PWM 波形的极性，CTIO1 和 CTIO0 位使能 PWM 输出或强制 CTM 输出为高电平或低电平。CTPOL 位用于 PWM 输出波形的极性反相控制。

- **10-bit CTM，PWM 输出模式，边沿对齐模式，CTDPX=0**

CCRP	1~7	0
Period	CCRP×128	1024
Duty	CCRA	

若 f_{SYS}=8MHz，CTM 时钟源选择 $f_{SYS}/4$，CCRP=2，CCRA=128，

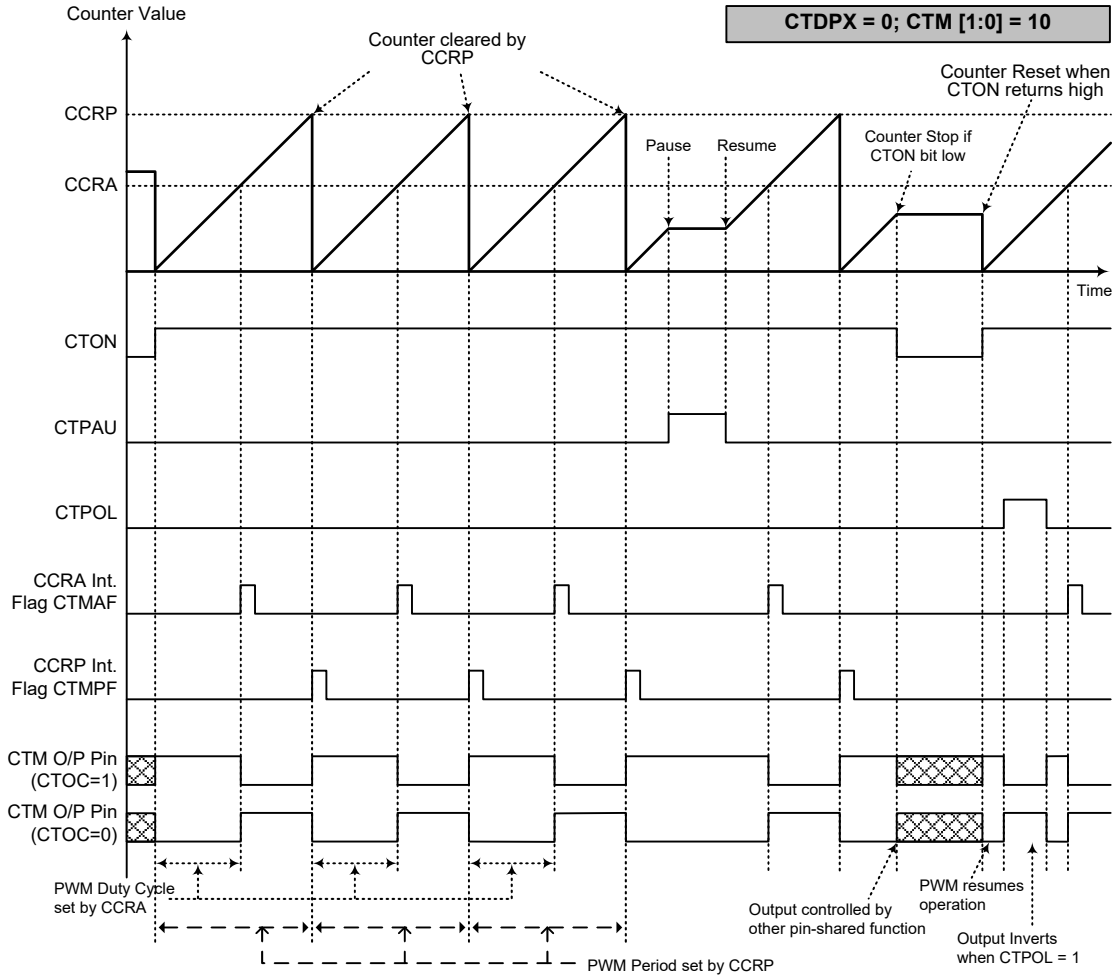
CTM PWM 输出频率 = $(f_{SYS}/4)/(2\times128)=f_{SYS}/1024$=7.812kHz，duty=128/(2×128)=50%。

若由 CCRA 寄存器定义的 Duty 值等于或大于 Period 值，PWM 输出占空比为 100%。

- **10-bit CTM，PWM 输出模式，边沿对齐模式，CTDPX=1**

CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

PWM 的输出周期由 CCRA 寄存器的值与 CTM 的时钟共同决定，PWM 的占空比由 CCRP 寄存器的值决定。



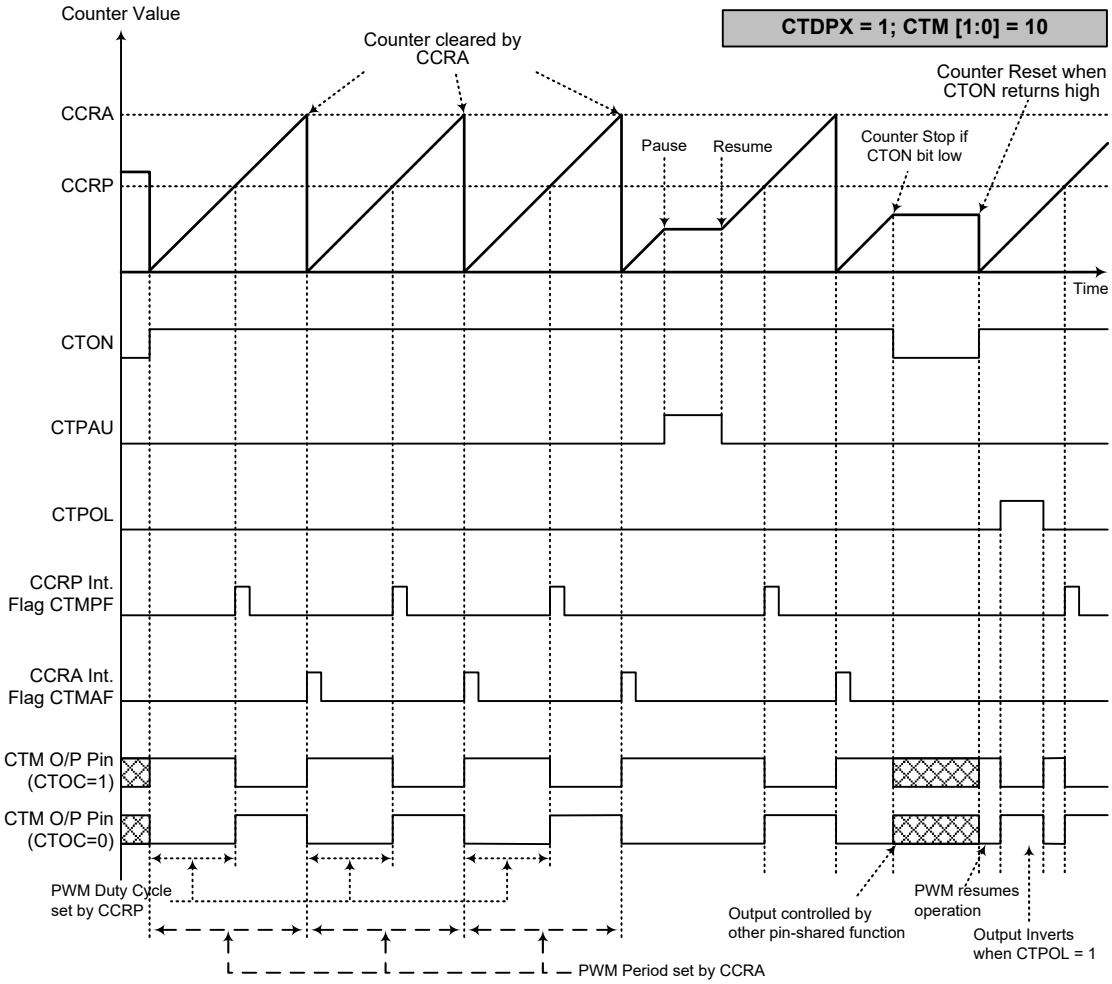
PWM 输出模式 – CTDPX=0

注：1. 这里的 CTDPX=0 – 计数器由 CCRP 清除

2. 计数器清零并设置 PWM 周期

3. 即使在 CTIO[1:0]=00 或 01 时，内部 PWM 功能继续运行

4. CTCCLR 位对 PWM 操作没有影响



PWM 输出模式 – CTDPX=1

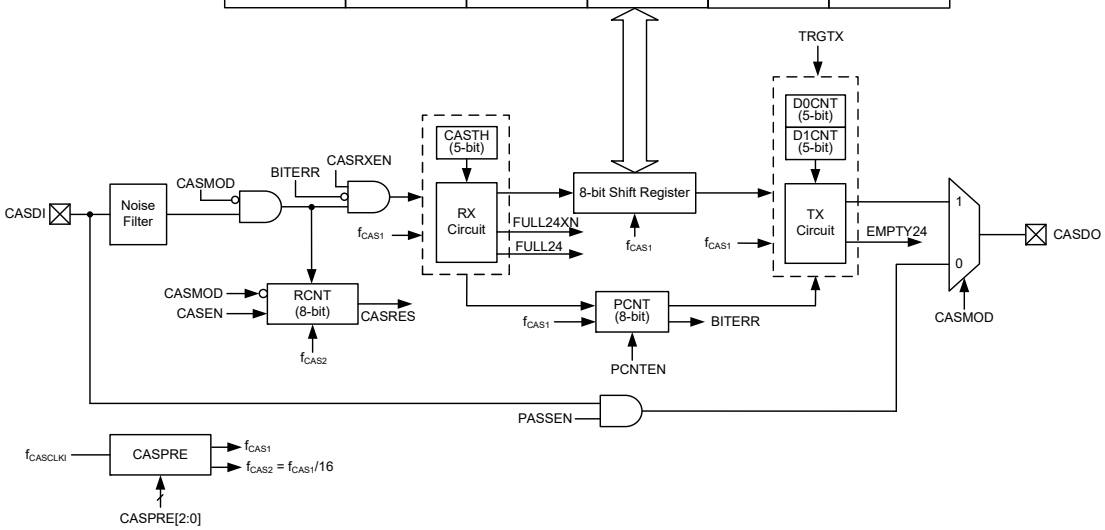
注：1. 这里的 CTDPX=1 – 计数器由 CCRA 清除
2. 计数器清零并设置 PWM 周期
3. 即使在 CTIO[1:0]=00 或 01 时，内部 PWM 功能继续运行
4. CTCCLR 位对 PWM 操作无影响

级联式收发器接口

级联式收发器功能为 LED 灯带的一个重要特性。它由主控单片机产生，并通过单线级联式收发器接口传送 PWM 数据用于 RGB LED 色调。传输速率应尽可能快以确保 RGB LED 色调平缓地变化。需要注意的是，当级联式收发器为主控单片机时，TX 功能才有效。

级联式收发器是单线单向的传输接口，包含一个 CASDI 输入脚和 CASDO 输出脚。RX 电路每接收到 24 位数据就可以开启一组 RGB LED。当 RX 电路已经收满 24×N 位数据时，会绕开路径到 TX 电路，再传送到下一颗单片机，由下一颗单片机读取接下来的 24×N 位数据，其中 N=1~16，根据应用需求并通过 CASDNB 寄存器来选择。因此这种依序截取 24×N 位数据的模式，形成所谓的串接模式，即级联电路。

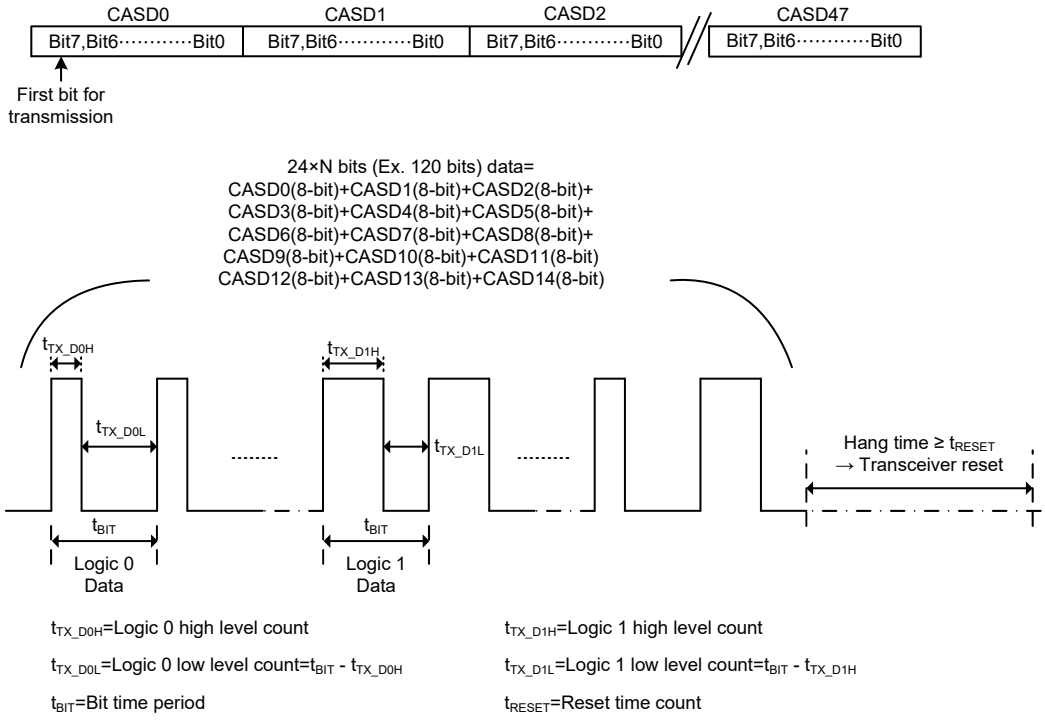
CASD8[7:0]	CASD17[7:0]	CASD26[7:0]	CASD35[7:0]	CASD44[7:0]	
CASD7[7:0]	CASD16[7:0]	CASD25[7:0]	CASD34[7:0]	CASD43[7:0]	
CASD6[7:0]	CASD15[7:0]	CASD24[7:0]	CASD33[7:0]	CASD42[7:0]	
CASD5[7:0]	CASD14[7:0]	CASD23[7:0]	CASD32[7:0]	CASD41[7:0]	
CASD4[7:0]	CASD13[7:0]	CASD22[7:0]	CASD31[7:0]	CASD40[7:0]	
CASD3[7:0]	CASD12[7:0]	CASD21[7:0]	CASD30[7:0]	CASD39[7:0]	
CASD2[7:0]	CASD11[7:0]	CASD20[7:0]	CASD29[7:0]	CASD38[7:0]	CASD47[7:0]
CASD1[7:0]	CASD10[7:0]	CASD19[7:0]	CASD28[7:0]	CASD37[7:0]	CASD46[7:0]
CASD0[7:0]	CASD9[7:0]	CASD18[7:0]	CASD27[7:0]	CASD36[7:0]	CASD45[7:0]



注：1. 级联式收发器时钟 $f_{CASCLKI}$ 来源于系统时钟 f_{SYS}。

2. TX 缓存器：CASD0~CASD2；
   RX 缓存器用于直推：CASD0~CASD14；
   RX 缓存器用于 3×COM 扫描：CASD0~CASD35；
   RX 缓存器用于 4×COM 扫描：CASD0~CASD47。

级联式收发器接口方框图



单线级联式收发器接口数据序列

级联式收发器接口寄存器介绍

级联式收发器接口的所有操作由一系列寄存器控制。寄存器 CASCON、INTCON0 和 INTCON1 用于级联式收发器各种 RX 和 TX 功能控制和中断控制。寄存器 CASPRE 用于选择级联式收发器的时钟。寄存器 CASTH 用来指定级联式收发器 RX 功能输入数据判断阈值。寄存器 D0CNT 和 D1CNT 用来控制级联式收发器 TX 功能输出数据为逻辑 0 和逻辑 1。寄存器 PCNT 和 RCNT 用来控制级联式收发器数据位时间周期和复位时间周期。寄存器 CASDNB 寄存器用于选择 RX 模式中要接收数据的长度。寄存器 CASD0~CASD47 用来存放接收到的数据或将要发送的数据。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CASCON	—	PCNTEN	CASRXEN	D4	TRGTX	PASSEN	CASMOD	CASEN
CASPRE	—	—	—	—	—	CASPRE2	CASPRE1	CASPRE0
CASTH	—	—	—	THS4	THS3	THS2	THS1	THS0
D0CNT	—	—	—	LCNT4	LCNT3	LCNT2	LCNT1	LCNT0
D1CNT	—	—	—	HCNT4	HCNT3	HCNT2	HCNT1	HCNT0
PCNT	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
RCNT	RS7	RS6	RS5	RS4	RS3	RS2	RS1	RS0
CASDNB	—	—	—	—	D3	D2	D1	D0
CASDn	D7	D6	D5	D4	D3	D2	D1	D0
INTCON0	BITERR	CASRES	EMPTY24	FULL24XN	BERINTEN	RESINTEN	EPTINTEN	FULINTEN
INTCON1	—	—	—	FULL24	—	—	—	INT24EN

级联式收发器接口寄存器列表 **(n=0~47)**

- **CASCON 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PCNTEN	CASRXEN	D4	TRGTX	PASSEN	CASMOD	CASEN
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	1	0	0	0	0	0	0

Bit 7　未定义，读为“0”

Bit 6　**PCNTEN**：RX/TX 传输位的时间计数器控制
0：除能
1：使能
该位用来计算 RX 或 TX 传输的数据位时间周期。当 PCNTEN 位被清零时，RX 模式下的位时间计数器 PCNT 功能将被除能。这表示将不检查每个 bit 的传输时间，BITERR 位将始终为零。而在 TX 模式下，PCNT 总是使能，PCNTEN 位对 PCNT 功能无影响。
当 PCNTEN 置为 1 时，位时间计数器 PCNT 功能将使能。对于 TX 功能，位时间计数器用来定义每个位传输的时间。对于 RX 功能，位时间计数器用来定义每个位传输的最长时间。若在位时间计数器溢出前，CASDI 引脚还未出现下一个上升沿，BITERR 位将置 1。

Bit 5　**CASRXEN**：级联式收发器 RX 功能使能控制
0：除能
1：使能
该位用来控制级联式收发器 RX 功能。当该位被置 1 时，级联式收发器 RX 功能将使能。当 RX 移位寄存器满标志位 FULL24XN 置高时，该位将自动清零。当级联式收发器复位标志位 CASRES 被置高时，该位又将自动置为 1。当该位为 0 时，级联式收发器 RX 功能将除能。然而，即使 RX 功能除能，级联式收发器复位信号仍将被解码且被识别。需要注意的是，若 CASRXEN 位被硬件置 1，PASSEN 位将自动清零，反之亦然。

Bit 4　**D4**：保留位，需固定为“0”

Bit 3　**TRGTX**：级联式收发器 TX 输出缓存触发控制
0：无动作或数据已传输完成
1：已触发 TX 缓存输出，数据将继续传输
当 CASEN 位为 1 时，该位仅可被写入 1。当 TX 移位寄存器空标志位 EMPTY24 置高，TRGTX 位将被硬件清零。若 CASEN 位为低时，该位也将被清零。需要注意的是，若 TRGTX 位被软件置高时，无论级联式收发器工作在哪种模式下，EMPTY24 位都将清零。而在 RX 模式下设置 TRGTX 位将无效。

Bit 2　**PASSEN**：级联式收发器输入信号绕开 RX 电路的使能控制位
0：除能 – 未绕开 RX 电路
1：使能 – 绕开 RX 电路
该位用来控制级联式收发器输入信号绕开 RX 电路的功能。该位可被硬件自动置位和清零。当 CASRXEN 位被硬件置位，级联式收发器 RX 功能使能，PASSEN 位将被硬件自动清零，RX 电路将解码级联式收发器的输入信号。若 CASRXEN 位被硬件清零，级联式收发器 RX 功能将被除能，PASSEN 位将被硬件自动置位。此时，级联式收发器输入信号将绕开 RX 电路，直接连接至 CASDO 引脚。

Bit 1　**CASMOD**：级联式收发器 TX 或 RX 模式选择位
0：RX 模式
1：TX 模式
仅当 CASEN 位为低时，该位才可被修改。应先选择级联式收发器工作模式，然后再设置 CASEN 位为高。

Bit 0　**CASEN**：级联式收发器使能控制位
0：除能
1：使能
该位用来使能或除能级联式收发器功能。当该位被清零，级联式收发器功能将除能，内部控制电路、INTCON0 和 INTCON1 寄存器中相应的只读位和

TRGTX 位将复位。其它寄存器内容保持不变。需要注意的是，在 RX 模式下，若 CASEN 位为低，寄存器 CASD0~CASD47 中的内容是未知的。若 CASEN 位为低，CASDI 输入路径将关闭，CASDO 输出将会浮空。

- **CASPRE 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	CASPRE2	CASPRE1	CASPRE0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3　未定义，读为“0”

Bit 2~0　**CASPRE2~CASPRE0**：级联式收发器时钟 f_{CAS1} 分频比选择

000：$f_{CAS1}=f_{CASCLKI}$
001：$f_{CAS1}=f_{CASCLKI}/2$
010：$f_{CAS1}=f_{CASCLKI}/4$
011：$f_{CAS1}=f_{CASCLKI}/8$
100：$f_{CAS1}=f_{CASCLKI}/16$
101：$f_{CAS1}=f_{CASCLKI}/32$
110：$f_{CAS1}=f_{CASCLKI}/64$
111：$f_{CAS1}=f_{CASCLKI}/128$

这些位用于级联式收发器时钟 f_{CAS1} 的分频比选择。$f_{CASCLKI}$ 时钟为级联式收发器输入时钟，来源于系统时钟 f_{SYS}。f_{CAS1} 时钟被用于驱动除了复位时间计数器以外的所有级联式收发器电路。复位时间计数器 RCNT 由时钟 f_{CAS2} 驱动，这里 $f_{CAS2}=f_{CAS1}/16$。

- **CASTH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	THS4	THS3	THS2	THS1	THS0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	1	1	1

Bit 7~5　未定义，读为“0”

Bit 4~0　**THS4~THS0**：级联式收发器 RX 功能数据判断阈值

逻辑 0 阈值：$t_{RX\_DT0HS}=(THS[4:0] - 2) \times t_{CAS1}$，
逻辑 1 阈值：$t_{RX\_DT1HS}=(THS[4:0] + 1) \times t_{CAS1}$，
这里 $t_{CAS1}=1/f_{CAS1}$

这些位用来指定级联式收发器 RX 功能输入数据判断阈值。当输入信号的高电平时间大于或等于 t_{RX_DT1HS} 阈值时，输入数据将被认为逻辑 1。若输入信号的高电平时间小于或等于 t_{RX_DT0HS} 阈值，输入数据将被认为为逻辑 0。接收的数据将被存储在寄存器 CASD0~CASD47 中。

- **D0CNT 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	LCNT4	LCNT3	LCNT2	LCNT1	LCNT0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	1	0	0

Bit 7~5　未定义，读为“0”

Bit 4~0　**LCNT4~LCNT0**：级联式收发器 TX 功能输出数据逻辑 0 的高电平脉冲计数值，t_{TX_D0H}

$t_{TX\_D0H}=LCNT[4:0] \times t_{CAS1}$，这里 $t_{CAS1}=1/f_{CAS1}$

这些位用来指定级联式收发器 TX 功能输出数据逻辑 0 的高电平脉冲计数值，由 f_{CAS1} 时钟驱动。若存储在 CASDn 寄存器的传输数据为逻辑 0，CASDO 引脚

将输出一个信号，其高电平脉宽为 t_{TX_D0H}，低电平脉宽为 (t_{BIT} - t_{TX_D0H})，这里位时间 t_{BIT} 由位时间计数器 PCNT 指定。LCNT 字段的最小值应根据级联式收发器输入时钟频率 $f_{CASCLKI}$ 来为应用正确配置。

- **D1CNT 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	HCNT4	HCNT3	HCNT2	HCNT1	HCNT0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	1	0	1	0

Bit 7~5　未定义，读为“0”

Bit 4~0　**HCNT4~HCNT0**：级联式收发器 TX 功能输出数据逻辑 1 的高电平脉冲计数值，t_{TX_D1H}

t_{TX_D1H}=HCNT[4:0] × t_{CAS1}，这里 t_{CAS1}=1/f_{CAS1}

这些位用来指定级联式收发器 TX 功能输出数据逻辑 1 的高电平脉冲计数值，由 f_{CAS1} 时钟驱动。若存储在 CASDn 寄存器的传输数据为逻辑 1，CASDO 引脚将输出一个信号，其高电平脉宽为 t_{TX_D1H}，低电平脉宽为 (t_{BIT} - t_{TX_D1H})，这里的位时间 t_{BIT} 由位时间计数器 PCNT 指定。HCNT 字段的最小值应根据级联式收发器输入时钟频率 $f_{CASCLKI}$ 来为应用正确配置。

- **PCNT 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	1	0	0	0

Bit 7~0　**PS7~PS0**：级联式收发器位时间计数器

TX 模式：t_{BIT}=PS[7:0] × t_{CAS1}，

RX 模式：t_{BIT} < (PS[7:0]-1) × t_{CAS1}，

这里 t_{CAS1}=1/f_{CAS1}

该计数器用来指定级联式收发器数据位时间周期的计数值。位时间计数器由 f_{CAS1} 时钟驱动。对于 TX 功能，无论 PCNTEN 位状态如何，位时间计数器始终使能。TX 模式下的位时间可由以上公式计算出来。在 TX 模式下，若 TRGTX 置为 1，PCNT 和 D0CNT 或 D1CNT 计数器将开始计数。CASDO 引脚将输出逻辑 0 或者逻辑 1，逻辑 0 由高电平脉冲 t_{TX_D0H} 和低电平脉冲 (t_{BIT} - t_{TX_D0H}) 组成。逻辑 1 由高电平脉冲 t_{TX_D1H} 和低电平脉冲 (t_{BIT} - t_{TX_D1H}) 组成。

对于 RX 功能来说，位时间计数器除了用来指定位时间，还用来检查接收的数据是否出现错误。若通过将 PCNTEN 位置高使能 PCNT 功能，且 CASDI 脚出现一个上升沿，位时间计数器 PCNT 将开始向下计数。接收数据 bit 0 ~ bit (24×N-2) 期间，若在 PCNT 计数器向下计数到零前，CASDI 脚还未出现第二个上升沿，位错误标志位 BITERR 将自动置 1，表示位传输发生错误。对于 bit (24×N-1) 的接收，若在 PCNT 计数器向下计数到零前，CASDI 脚出现一个下降沿，RX 移位寄存器满标志位 FULL24XN 和 FULL24 将置 1。这就意味着，24×N 位数据已经全部接收完成。此时 PASSEN 位将被硬件自动置 1。当接收数据 bit (24×N-1) 时，在 PCNT 计数器向下计数到零前，若 CASDI 脚未出现下降沿，位错误标志位 BITERR 也将被硬件置 1，表示位传输发生错误。此时 FULL24XN 和 FULL24 标志位将被清零，而 PASSEN 位将保持在低电平不变。

- **RCNT 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	RS7	RS6	RS5	RS4	RS3	RS2	RS1	RS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	0	0	0	0

Bit 7~0 **RS7~RS0**：级联式收发器复位时间计数器

t_{RESET}=RS[7:0] × t_{CAS2}，这里 t_{CAS2}=1/f_{CAS2}=16/f_{CAS1}=16 × t_{CAS1}

该向下计数器用来指定级联式收发器 RX 功能复位时间周期的计数值。复位时间计数器由 f_{CAS2} 时钟驱动，f_{CAS2} 时钟为 f_{CAS1} 时钟的 16 除频。若先将 CASMOD 位设为 0 来选择 RX 模式，将 CASEN 位设为 1 来使能级联式收发器功能，RCNT 计数器将开始计数。若 CASDI 信号在一段特定时间保持高电平或低电平不变，RCNT 向下计数到零，级联式收发器复位标志位 CASRES 将置 1，表示级联式收发器发生复位。若 CASRES 位设为 1，BITERR、FULL24XN、FULL24 和 PASSEN 位将清零，CASRXEN 位将置 1。

- **CASDNB 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D3	D2	D1	D0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	1	1

Bit 7~4 未定义，读为“0”

Bit 3~0 **D3~D0**：级联 24×N 位，仅用于 RX 模式

0000：N=1
0001：N=2
0010：N=3
0011：N=4
0100：N=5
0101：N=6
0110：N=7
0111：N=8
1000：N=9
1001：N=10
1010：N=11
1011：N=12
1100：N=13
1101：N=14
1110：N=15
1111：N=16

- **CASDn 寄存器 (n=0~2)**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：数据字节 n

该寄存器用来存储 RX 模式下接收的或者 TX 模式下传输的数据字节 n。

- **CASDn 寄存器 (n=3~47)**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：数据字节 n
该寄存器用来存储 RX 模式下接收的数据字节 n。

- **INTCON0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	BITERR	CASRES	EMPTY24	FULL24XN	BERINTEN	RESINTEN	EPTINTEN	FULINTEN
R/W	R	R	R	R	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **BITERR**：RX 接收的数据位溢出标志位
0：未发生位溢出错误
1：发生位溢出错误
该位用来表明接收的位是否溢出。当接收的数据位时间大于 PCNT 寄存器指定的 t_{BIT} 时间，BITERR 位将被硬件置 1 表示接收的位发生溢出。若发生位溢出错误，直到 CASRES 位被置高，即级联式收发器发生复位，RX 输入数据才会被解码。若 CASRES 位被置高，BITERR 位将被硬件自动清零。

Bit 6 **CASRES**：级联式收发器复位标志位
0：未发生复位
1：发生复位
该位用来表明级联式收发器是否发生复位。若 CASDI 信号保持高电平或低电平的时间超过 t_{RESET}，则 CASRES 位将被硬件置 1，表明 RX 功能发生复位，这里的 t_{RESET} 由 RCNT 寄存器设置。若 CASRES 位被置高，BITERR、FULL24XN、FULL24 和 PASSEN 位将被自动清零，而 CASRXEN 位被硬件置高。若 CASDI 引脚出现一个上升沿，CASRES 位将清零。

Bit 5 **EMPTY24**：级联式收发器 24-bit TX 移位寄存器空标志位
0：TX 移位寄存器不为空
1：TX 移位寄存器为空
该位用来表明级联式收发器 24-bit TX 移位寄存器是否为空。若 TX 电路完成 24 位数据传输，24-bit 移位寄存器将为空，EMPTY24 位将置 1。若 EMPTY24 位被置高，TRGTX 位将被硬件自动清零。当 TRGTX 位被置高时，数据将继续传输，EMPTY24 位将自动清零。

Bit 4 **FULL24XN**：级联式收发器 24×N 位 RX 移位寄存器满标志位
0：24×N 位 RX 移位寄存器未满
1：24×N 位 RX 移位寄存器已满
该位用来表明级联式收发器 24×N 位 RX 移位寄存器是否已满。若 RX 电路接收满 24×N 位数据，24×N 位移位寄存器将会被填满，FULL24XN 位将置 1 而 BITERR 位将被清零。若 FULL24XN 位被置高，PASSEN 位将置 1，CASRXEN 位将被硬件清零。这使 CASDI 信号绕开级联式收发器，直接输出至 CASDO 脚。若 CASRES 位被置高，FULL24XN 位将自动清零。

Bit 3 **BERINTEN**：RX 接收的数据位错误中断控制
0：除能
1：使能
该位用来控制 RX 接收的数据位错误中断功能。当 BERINTEN 位被置高时，若 BITERR 位也被置高，级联式收发器将产生一个位错误中断信号来通知单片机。

Bit 2 **RESINTEN**：级联式收发器复位中断控制
0：除能
1：使能
该位用来控制级联式收发器复位中断功能。当 RESINTEN 位被置高时，若 CASRES 位也被置高，级联式收发器将会产生一个复位中断信号来通知单片机。

Bit 1 **EPTINTEN**：级联式收发器 24-bit TX 移位寄存器空中断控制
0：除能
1：使能
该位用来控制级联式收发器 TX 移位寄存器空中断功能。当 EPTINTEN 位被置高时，若 EMPTY24 位也被置高，级联式收发器将会产生一个 TX 移位寄存器空中断信号来通知单片机。

Bit 0 **FULINTEN**：级联式收发器 24×N 位 RX 移位寄存器满中断控制
0：除能
1：使能
该位用来控制级联式收发器 24×N 位 RX 移位寄存器满中断功能。当 FULINTEN 位被置高时，若 FULL24XN 位也被置高，级联式收发器将会产生一个 24×N 位 RX 移位寄存器满中断信号来通知单片机。

- **INTCON1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	FULL24	—	—	—	INT24EN
R/W	—	—	—	R/W	—	—	—	R/W
POR	—	—	—	0	—	—	—	0

Bit 7~5 未定义，读为“0”

Bit 4 **FULL24**：级联式收发器 24-bit RX 移位寄存器满标志位
0：24-bit RX 移位寄存器未满
1：24-bit RX 移位寄存器已满
该位用来表明级联式收发器 24-bit RX 移位寄存器是否已满。若 RX 电路接收满 24 位数据，24-bit 移位寄存器将会被填满，FULL24 位将置 1。如果还需要接收更多的数据，应由软件将该标志位清零。
若 CASRES 位被置高，FULL24 位将自动清零。需要注意的是该标志位不可由软件置高。

Bit 3~1 未定义，读为“0”

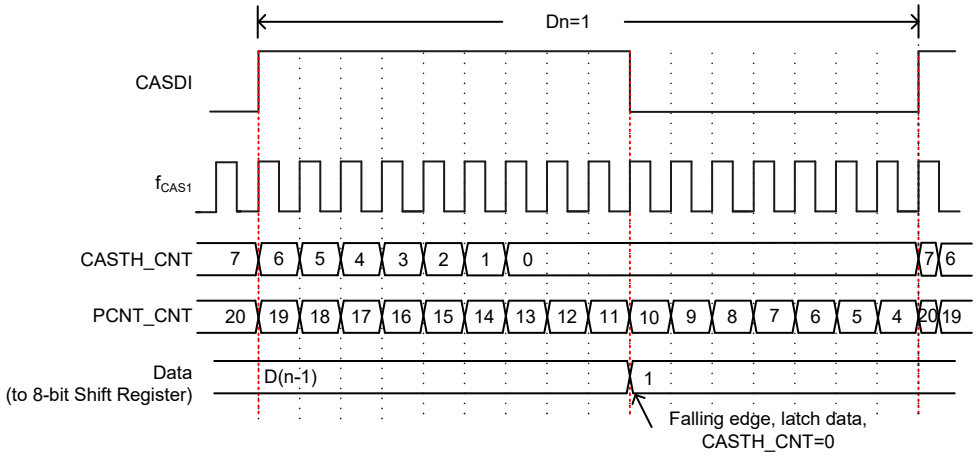
Bit 0 **INT24EN**：RX 收完 24 位数据中断控制
0：除能
1：使能
该位用来控制级联式收发器 24-bit RX 移位寄存器满中断功能。当 INT24EN 位被置高时，若 FULL24 位也被置高，级联式收发器将会产生一个 24-bit RX 移位寄存器满中断信号来通知单片机。

级联式收发器 RX 功能操作

级联式收发器 RX 功能用来解码 CASDI 脚的脉冲是逻辑高还是逻辑低。

级联式收发器功能逻辑高

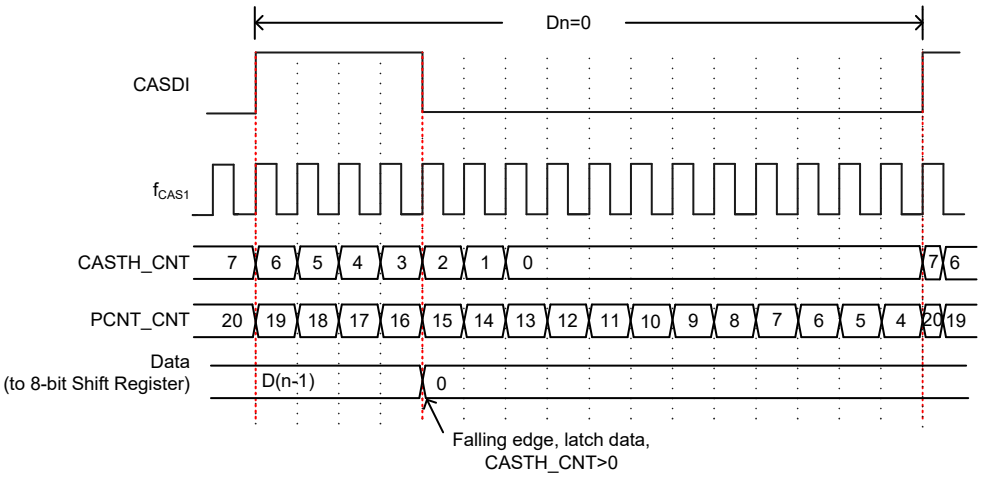
若级联式收发器时钟高电平计数值大于或等于 RX 功能数据判断阈值 (CASTH+1)，这意味着解码为逻辑高。



在 RX 模式下解码为逻辑 1 (假设 CASTH=7)

级联式收发器功能逻辑低

若级联式收发器时钟高电平计数值小于或等于RX功能数据判断阈值(CASTH-2)，这意味着解码为逻辑低。



在 RX 模式下解码为逻辑 0 (假设 CASTH=7)

级联式收发器 RX 步骤

级联式收发器功能开始执行前，若 CASDI 脚的信号在特定的时间周期内保持高电平或低电平不变，RCNT 计数器向下计数到零，这时级联式收发器复位标志位 CASRES 将置 1，表示级联式收发器发生复位。

若 CASRES 位被置高，多路复用器直接与 CASDO 脚相连的 bypass 路径将会被除能，RX 电路仅解码 CASDI 脚的信号。

步骤 1：若主控单片机复位级联式收发器单线总线，FULL24XN、FULL24、BITERR 和 PASSEN 位将清零，CASRES 和 CASRXEN 位将置 1，RX 电路仅为 CASDI 脚的信号准备。

步骤 2：若 CASDI 引脚上出现上升沿，RX 电路将开始计数高电平个数。若在 CASDI 引脚出现下降沿前，高电平计数值小于或等于阈值(CASTH-2)，将解码为逻辑低。若在 CASDI 引脚出现下降沿前，高电平计数值大于或等于阈值 (CASTH+1)，将解码为逻辑高。

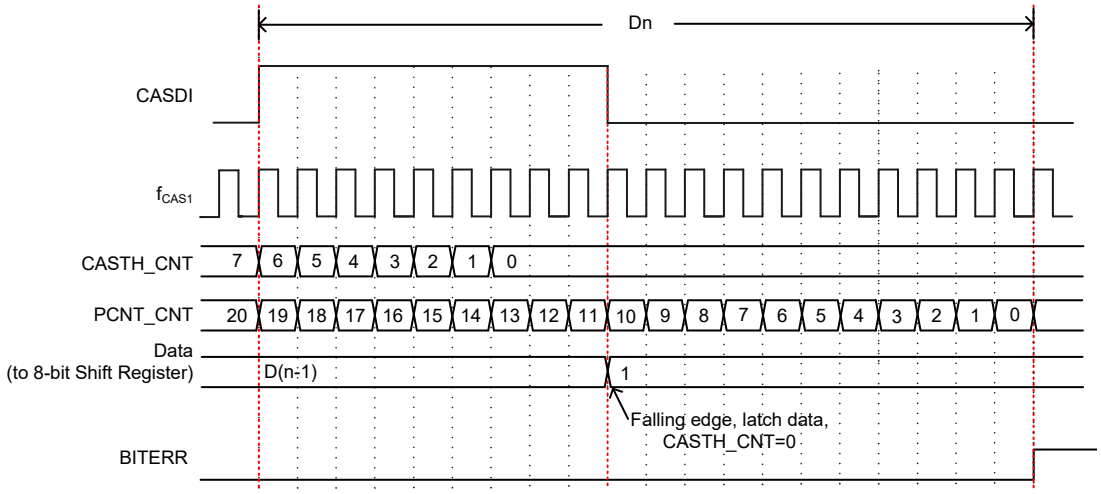
步骤 3：若 24×N 位移位寄存器已满，FULL24XN 位将置 1，总线将自动绕开 RX 电路。因此若 PASSEN 位置为 1，且 CASRXEN 位清零，系统将产生一个 CASINT 中断。

步骤 4：CASDI 引脚的输入信号将通过多路复用器传送到 CASDO 引脚，然后传送到下一颗单片机。

步骤 5：当主控单片机再次复位级联总线，将再次从步骤 1 重新开始。

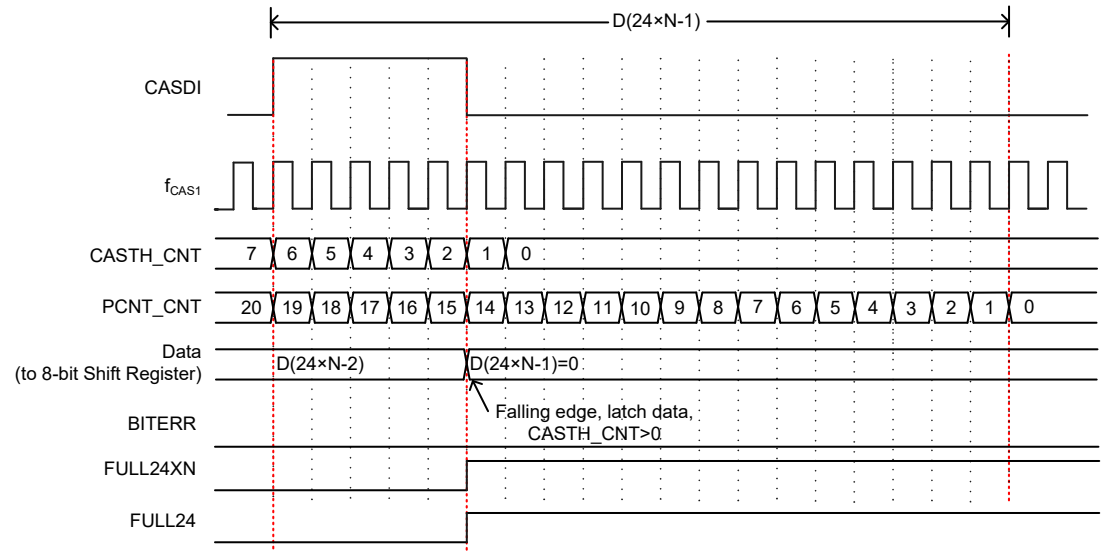
步骤 6：当 CASDI 引脚出现第一个上升沿时，CASRES 位将自动清零。

注：级联式收发器时钟 f_{CAS1} 可因不同的波特率调整。



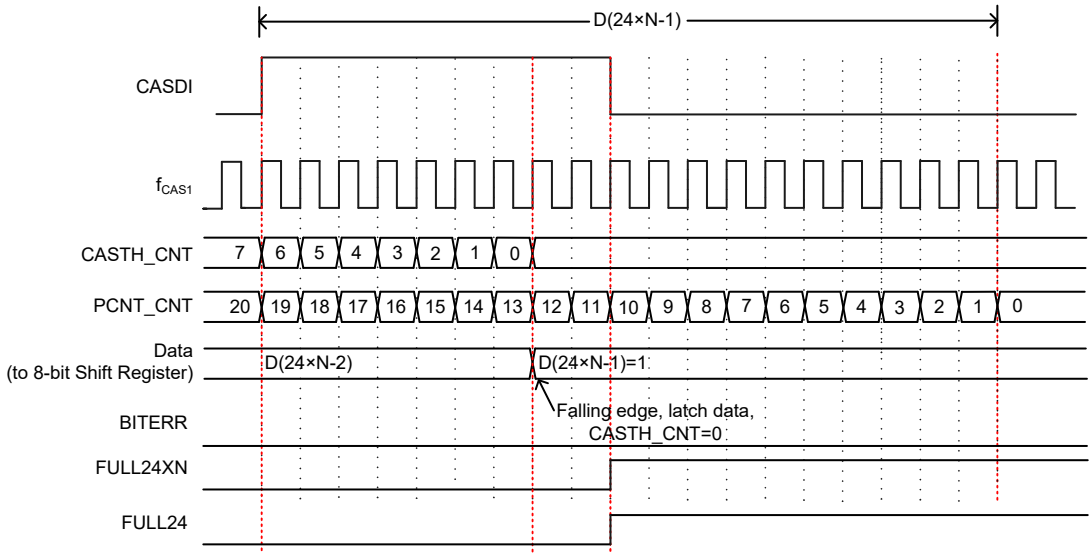
在 RX 模式下 bit 0~bit (24×N-2) 数据位发生错误

注：接收数据 bit 0~bit (24×N-2) 期间，若在 PCNT 计数器向下计数到零前，CASDI 脚还未出现第二个上升沿，位错误标志位 BITERR 将自动置为 1，表示发生位传输错误。



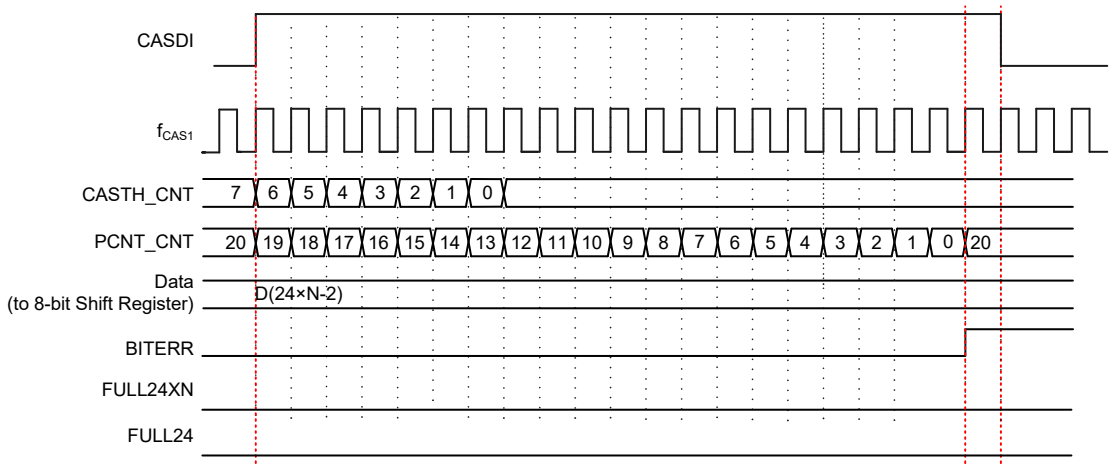
在 RX 模式下 bit (24×N-1) 数据解码为逻辑 0

注：对于数据 bit (24×N-1) 的接收，如果在 PCNT 计数器向下计数到零前，CASDI 脚出现一个下降沿，RX 移位寄存器满标志位 FULL24XN 和 FULL24 将置 1。在 CASTH 计数器向下计数为 0 前，当 CASDI 引脚出现一个下降沿时，数据逻辑 0 将被读出。



在 RX 模式下 bit (24×N-1) 解码为逻辑 1

注：对于数据 bit (24×N-1) 的接收，若 PCNT 计数器向下计数到零前，CASDI 脚出现一个下降沿，RX 移位寄存器满标志位 FULL24XN 和 FULL24 将置 1。当 CASTH 计数器向下计数为 0 时，数据逻辑 1 将存储在 8-bit 移位寄存器，且直到 CASDI 引脚出现下降沿时才被读出。



在 RX 模式下 bit (24×N-1) 数据异常情况

注：这是一个异常情况，若在接收第 (24×N-1) 位数据位期间，CASDI 引脚信号维持高电平不变，直到 PCNT 计数器溢出，CASDI 引脚都未出现下降沿，这表示 24×N 位数据没有接收完全，BITERR 位将置高，FULL24XN 和 FULL24 位将清零，而 PASSEN 位不变。

级联式收发器 TX 步骤

步骤 1：首先将 CASEN 位清零，除能 RX 和 TX 功能。

步骤 2：将 CASMOD 位置高，将 PASSEN 位清零，使能 TX 功能，通过路径传给 TX 电路，然后将 CASEN 位置高。

步骤 3：对寄存器 CASD0~CASD2 写入 3 字节数据。

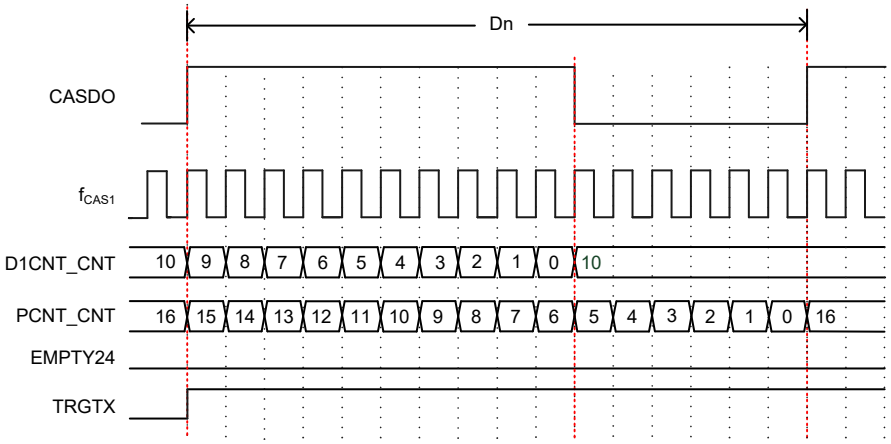
步骤 4：将 TRGTX 位置高来开始转移寄存器 CASD0~CASD2 的数据并输出。

步骤 5：当 24-bit TX 移位寄存器为空将 EMPTY24 位置高时，再次将寄存器 CASD0~CASD2 填满值。重复步骤 4。

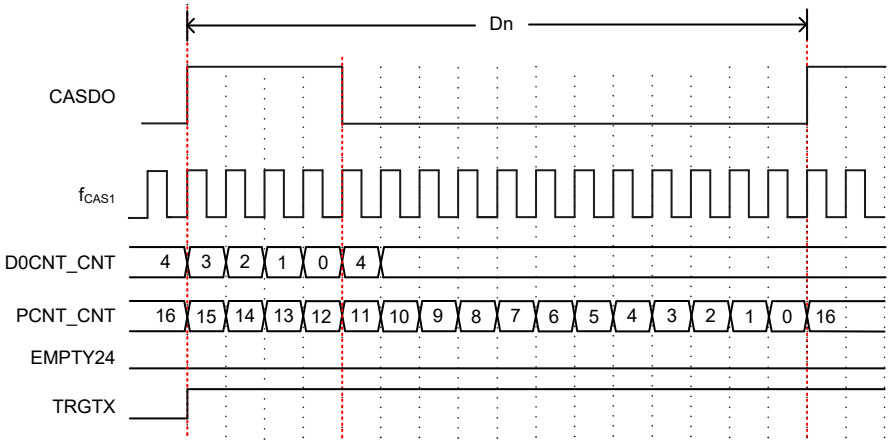
步骤 6：当每 24×N×M 位数据被移出时，TX 设备将发送 RESET 命令给从机，此时 RX 电路将通过软件把 CASRES 位置高，这里 M 表示 RX 设备级联的数量。

步骤 7：重复步骤 3，再次传送下一帧的 (24×N×M) 位数据。

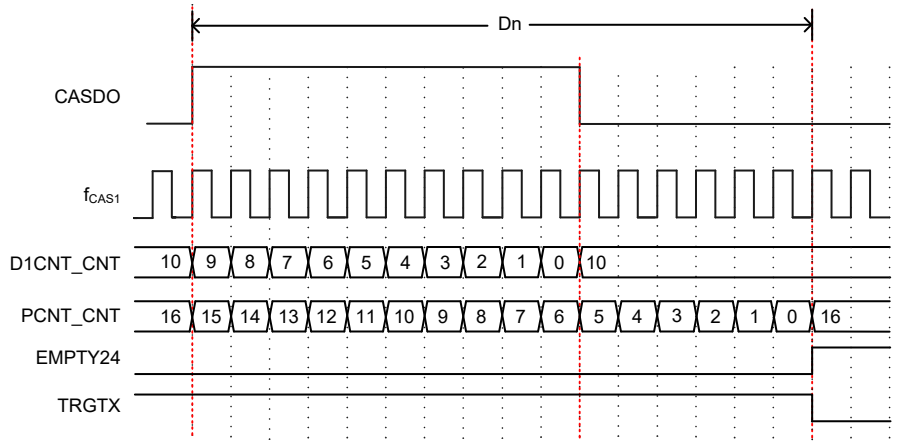
注：级联式收发器时钟 f_{CAS1} 可根据不同的波特率调整。



在 TX 模式下解码为 1 (n=0~24×N-2, D1CNT=10)



在 TX 模式下解码为 0 (n=0~24×N-2, D0CNT=4)

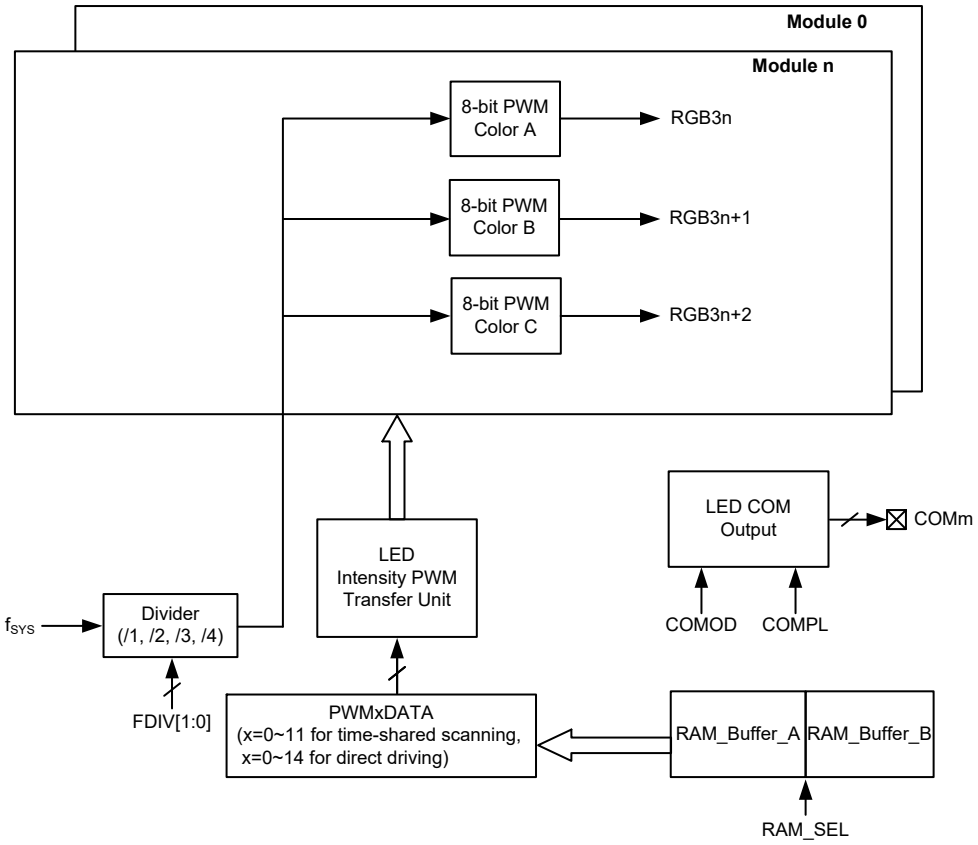


在 TX 模式下解码为 1 (n=24×N-1, D1CNT=10)

注：当 TX 移位寄存器空标志位 EMPTY24 置高时，TRGTX 位将由硬件清零。

用于 RGB LED 的 PWM 功能

该单片机具有一个可用于RGB LED应用的PWM功能，由四个LED PWM模块、两个 LED 数据缓存区和一个寄存器传输单元以及 LED COM 输出单元组成。



注：1. n=0~3 适用于分时扫描模式和直推模式，n=4 只适用于直推模式
2. m=0~3

用于 RGB LED 的 PWM 方框图

PWM 寄存器介绍

用于 RGB LED 的 PWM 功能的所有操作由一系列的寄存器控制。COM_PWM 寄存器用于 PWM 功能控制、PWM 时钟分频选择、扫描模式选择和配置。PWM0DATA~PWM14DATA 寄存器用于定义 PWM 输出的占空比。

寄存器名称	位							
	7	6	5	4	3	2	1	0
COM_PWM	PWMEN	—	FDIV1	FDIV0	RAM_SEL	COMPL	COMOD	SCANMOD
PWM0DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM1DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM2DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM3DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM4DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM5DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM6DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM7DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM8DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM9DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM10DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM11DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM12DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM13DATA	D7	D6	D5	D4	D3	D2	D1	D0
PWM14DATA	D7	D6	D5	D4	D3	D2	D1	D0

PWM 寄存器列表

- **COM_PWM 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PWMEN	—	FDIV1	FDIV0	RAM_SEL	COMPL	COMOD	SCANMOD
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

Bit 7 **PWMEN**：PWM 使能 / 除能控制位
0：PWM 除能
1：PWM 使能
当 PWMEN 置高时，禁止变更 FDIV1~FDIV0、COMPL、COMOD 和 SCANMOD 位，仅 RAM_SEL 可变更。当此位清零时，COM0~COM3 引脚固定输出无效数据，RGB0~RGB11 固定输出 1。

Bit 6 未定义，读为“0”

Bit 5~4 **FDIV1~FDIV0**：PWM 时钟分频选择位
f_{PWMCLK}=
00：$f_{SYS}/1$
01：$f_{SYS}/2$
10：$f_{SYS}/3$
11：$f_{SYS}/4$
当这几位设置为“10”时，正半周期 = 2×PWMCLK，负半周期 = 1×PWMCLK。

Bit 3 **RAM_SEL**：用于 COM 扫描显示的数据缓存器选择位 (仅适用于分时扫描模式)
0：选择数据缓存器 A 用于 COM 扫描
1：选择数据缓存器 B 用于 COM 扫描
需要注意的是数据缓存器的更改仅在当前帧结束后才有效。在当前帧结束瞬间抓取 RAM_SEL 位的值并重新选择数据缓存器。在分时扫描模式下，无法对所选的缓存器进行一般读写操作。

Bit 2 **COMPL**：COM 极性选择位 (仅适用于分时扫描模式)
0：低电平有效
1：高电平有效
在直推模式下，COM 引脚输出无效数据。在 PWM 除能状态下，应注意 COMPL 位的设置。若 PWMEN=0，COMPL=0，COM 引脚将输出高电平 (无效数据)。若 PWMEN=0，COMPL=1，COM 引脚将输出低电平 (无效数据)。建议先通过共用功能控制位选择 COM 引脚并配置好 COMPL 位，再使能 PWM 功能。

Bit 1 **COMOD**：COM 模式选择位 (仅适用于分时扫描模式)
0：3×COM
1：4×COM

Bit 0 **SCANMOD**：PWM 分时扫描或直推模式选择位
0：直推模式
1：分时扫描模式

- **PWM0DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED0 颜色 A
该寄存器在分时扫描模式下为只读寄存器。

- **PWM1DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED0 颜色 B
该寄存器在分时扫描模式下为只读寄存器。

- **PWM2DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED0 颜色 C
该寄存器在分时扫描模式下为只读寄存器。

- **PWM3DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED1 颜色 A
该寄存器在分时扫描模式下为只读寄存器。

- **PWM4DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED1 颜色 B
该寄存器在分时扫描模式下为只读寄存器。

- **PWM5DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED1 颜色 C
该寄存器在分时扫描模式下为只读寄存器。

- **PWM6DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED2 颜色 A
该寄存器在分时扫描模式下为只读寄存器。

- **PWM7DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED2 颜色 B
该寄存器在分时扫描模式下为只读寄存器。

- **PWM8DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED2 颜色 C
该寄存器在分时扫描模式下为只读寄存器。

- **PWM9DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED3 颜色 A
该寄存器在分时扫描模式下为只读寄存器。

- **PWM10DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED3 颜色 B
该寄存器在分时扫描模式下为只读寄存器。

- **PWM11DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED3 颜色 C
该寄存器在分时扫描模式下为只读寄存器。

- **PWM12DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED4 颜色 A
该寄存器在分时扫描模式下无效。

- **PWM13DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED4 颜色 B
该寄存器在分时扫描模式下无效。

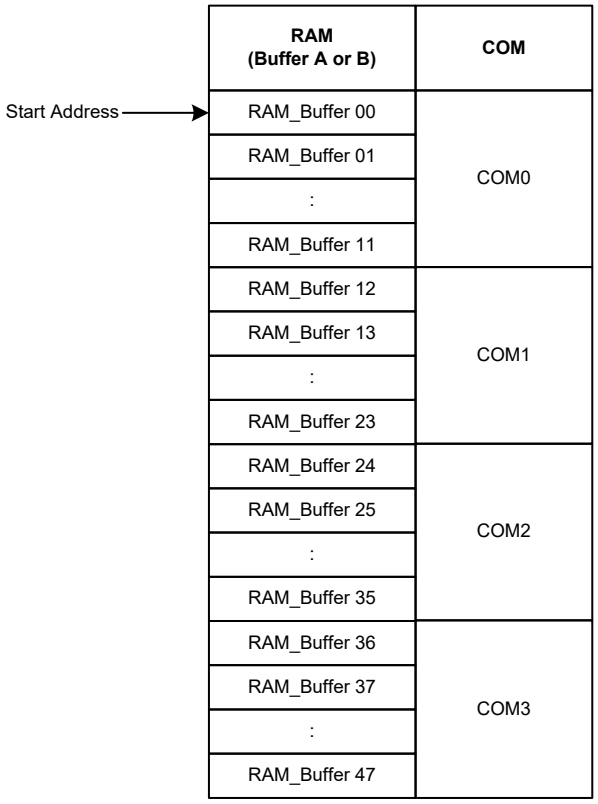
- **PWM14DATA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：LED4 颜色 C
该寄存器在分时扫描模式下无效。

PWM 数据缓存器

该单片机提供了 2 个 LED PWM 数据缓存器，即缓存器 A 和缓存器 B，每个缓存器都包含 64 字节。缓存器 A 位于 Bank 2 的 80H~BFH，缓存器 B 位于 Bank 2 的 C0H~FFH。然而，PWM 功能的实际可用大小为 48 字节，如下图所示。关于数据缓存器使用的详细内容将在 PWM 操作一节中描述。



PWM 操作

PWM 功能由 COM_PWM 寄存器中的 PWMEN 位控制。如果不使用 PWM 功能，则应将 PWMEN 位清零，以关闭 PWM 功能从而节省功耗。在 PWM 操作中，无论是直推模式还是分时扫描模式，如果执行 HALT 指令，但仍开启 f_{PWMCLK} 时钟，PWM 将继续工作并输出波形。然而，如果在执行 HALT 指令后关闭 f_{PWMCLK} 时钟，PWM 功能将停留在进入 HALT 的瞬时状态。

PWM 时钟来自于系统时钟 f_{SYS}，可以通过 FDIV1~FDIV0 位选择用于产生 f_{PWMCLK} 时钟的时钟分频器。

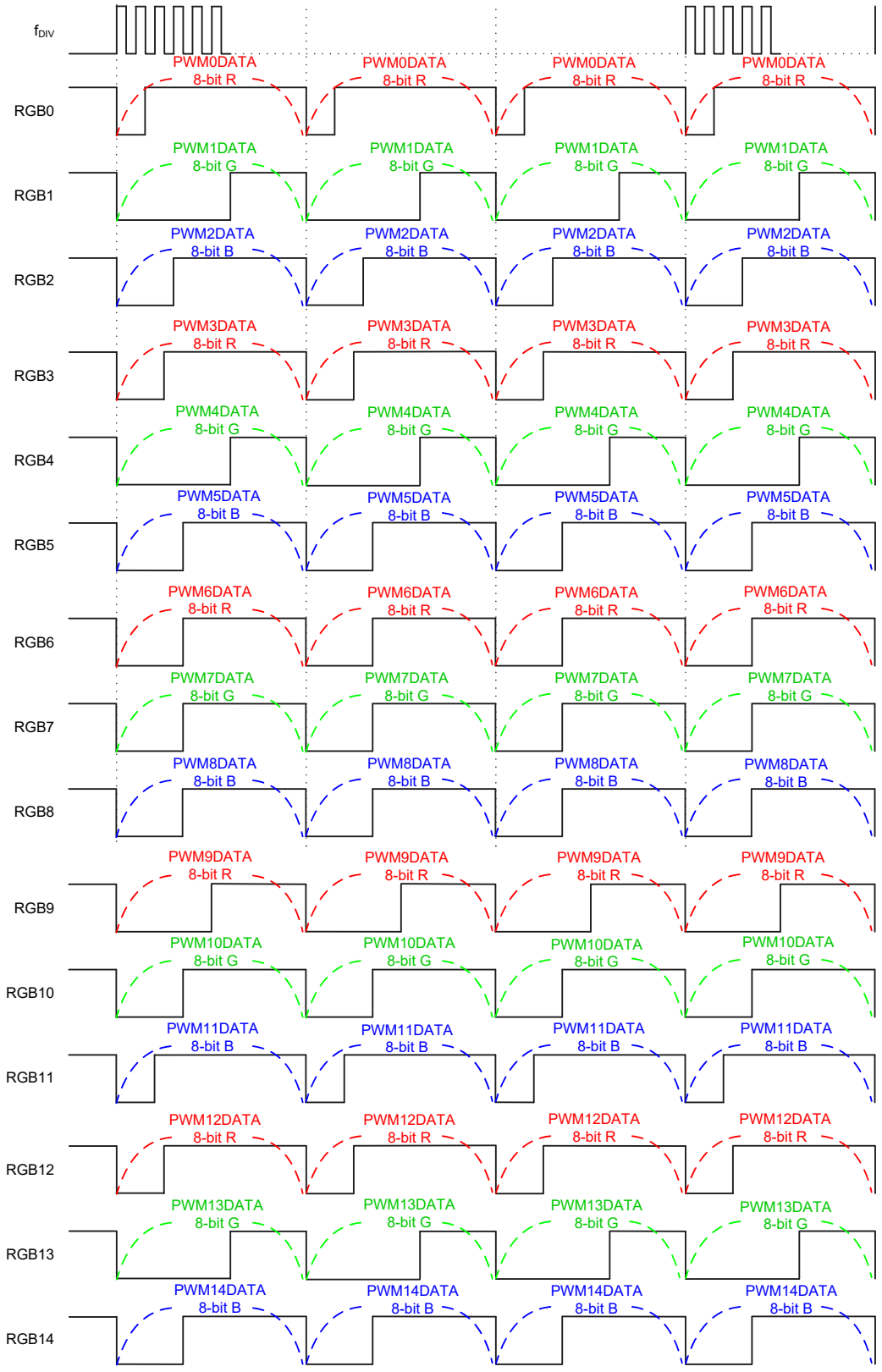
PWM 功能支持两种工作模式：直推模式和分时扫描模式，由 COM_PWM 寄存器中的 SCANMOD 位选择。

直推模式

如果将 SCANMOD 位清零来选择直推模式，则所有 4 个 PWM 模块都会被用来驱动 15 个 RGB LED。输出波形由 LED PWM 数据寄存器 PWM0DATA~PWM14DATA 确定。PWM 数据寄存器内容一旦发生改变，将直接反映在 RGBn 输出上，RGB 调光将立即生效。

如果 PWM 数据寄存器被修改，并不是将 PWM 功能复位再输出一个新的 PWM 占空比周期，而是在数据发生变化时，将新的 PWM 波形直接接续到旧的 PWM 波形上。这意味着数据变化前后的 PWM 波形使用同一个 ($256\times f_{PWMCLK}$) 周期。

在该模式下，RAM 缓存器 A 和 B 都可以用作通用数据存储器，并且可以读写。



直推模式的 **LED PWM** 时序

分时扫描模式

如果将 SCANMOD 位置高来选择分时扫描模式，那么只有 3 个 PWM 模块被用来驱动 12 个 RGB LED。输出波形由 LED PWM 数据寄存器 PWM0DATA~PWM11DATA 确定。PWM12DATA~PWM14DATA 寄存器在此模式下无效，无论寄存器内容如何，相应的 PWM 输出都浮空。

在该模式下，COM 模式可以通过 COMOD 位选择为 3×COM 或 4×COM，COM 极性由 COMPL 位确定，用于 COM 扫描显示的数据缓存器 A 或 B 由 RAM_SEL 位确定。RAM_Buffer_A/B 的 36 字节数据 (3×COM) 或 48 字节数据 (4×COM) 将由硬件每次传送 12 个字节到 PWM0DATA~PWM2DATA & PWM3DATA~PWM5DATA &PWM6DATA~PWM8DATA & PWM9DATA~PWM11DATA 寄存器，以产生相应的 PWM 波形。

如果 RAM_SEL 等于 0，则数据缓存器 A 被选中，无法由应用程序读写，但是数据缓存器 B 可读可写。相反，如果 RAM_SEL 等于 1，则数据缓存器 B 被选中，无法由应用程序读写，但是数据缓存器 A 可读可写。需要注意的是，如果数据缓存器的选择被更改，那么只有在当前帧完成后，新的数据缓存器数据才会被加载到 PWM 数据寄存器中。数据缓存器 A 或 B 中未使用的字节对 PWM 输出没有影响。

将数据从数据缓存器加载到 PWM 数据寄存器的时间为：

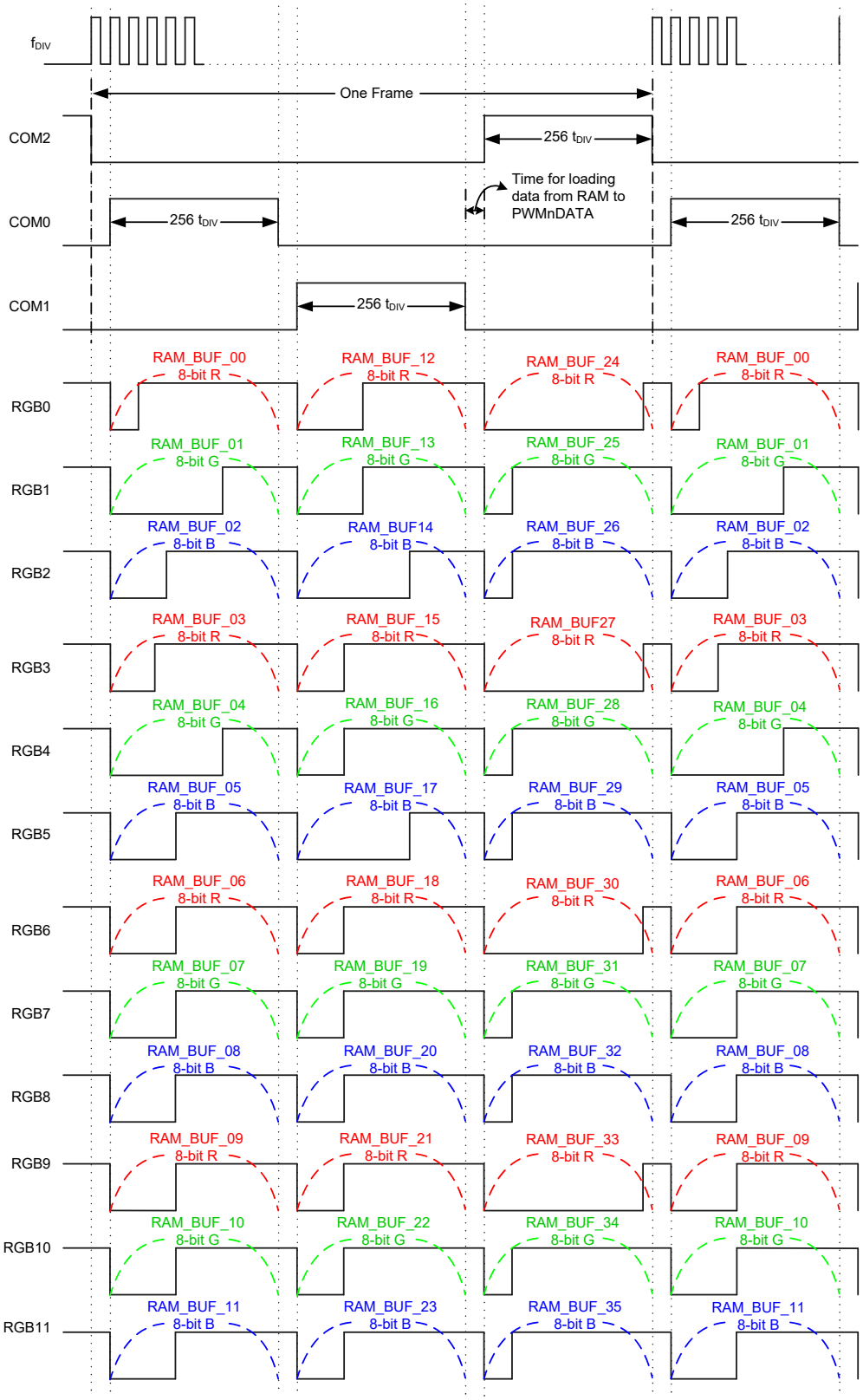
$4\times t_{PWMCLK}\times 12+(0\sim 1)\times t_{PWMCLK}+(0\sim 1)\times t_{DIV}$

3×COM 模式的帧周期为：

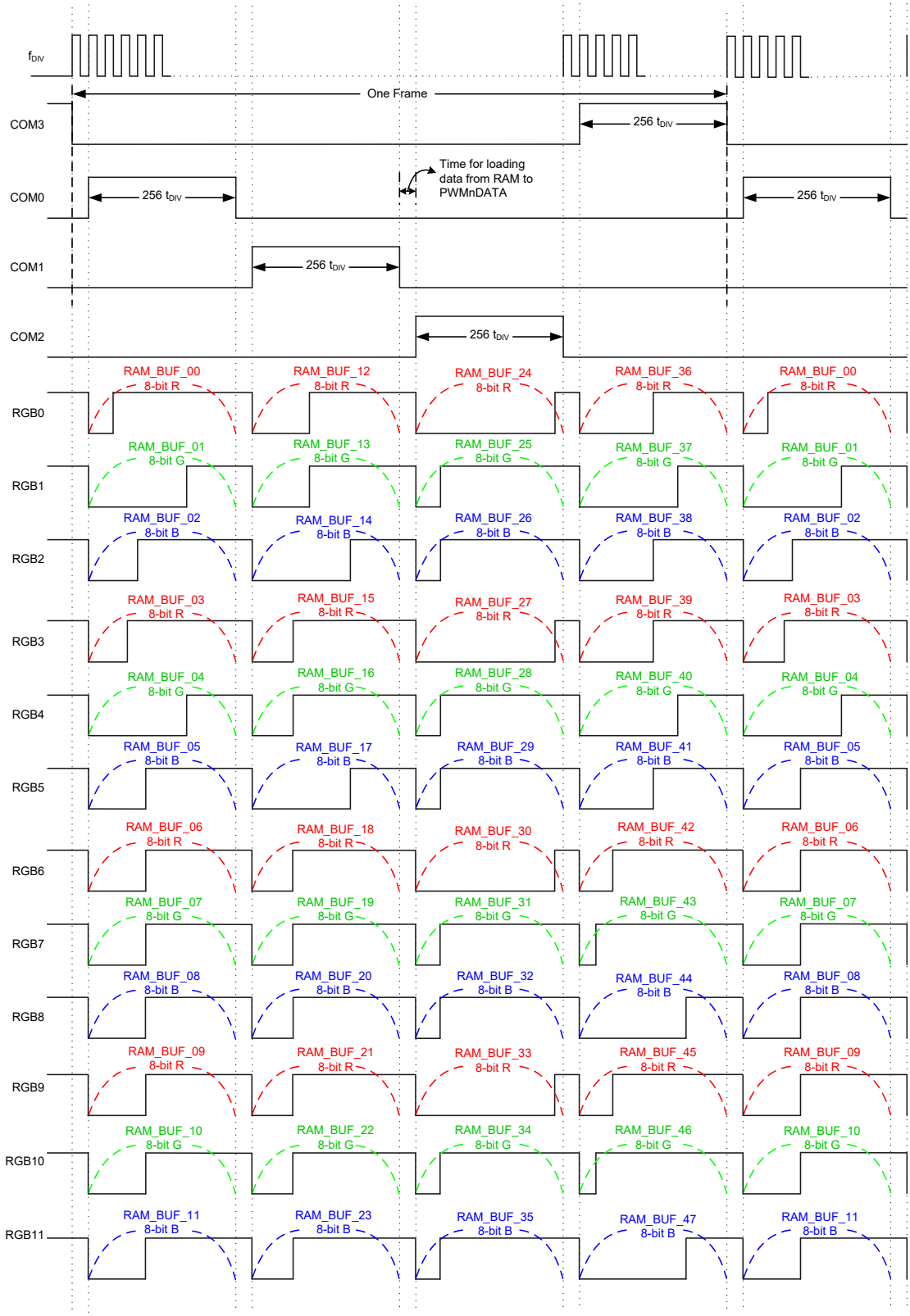
$3\times\{[4\times t_{PWMCLK}\times 12+(0\sim 1)\times t_{PWMCLK}+(0\sim 1)\times t_{DIV}]+256\times t_{DIV}\}$

4×COM 模式的帧周期为：

$4\times\{[4\times t_{PWMCLK}\times 12+(0\sim 1)\times t_{PWMCLK}+(0\sim 1)\times t_{DIV}]+256\times t_{DIV}\}$



分时扫描模式的 **LED PWM** 时序 – **3×COM**



分时扫描模式的 LED PWM 时序 – 4×COM

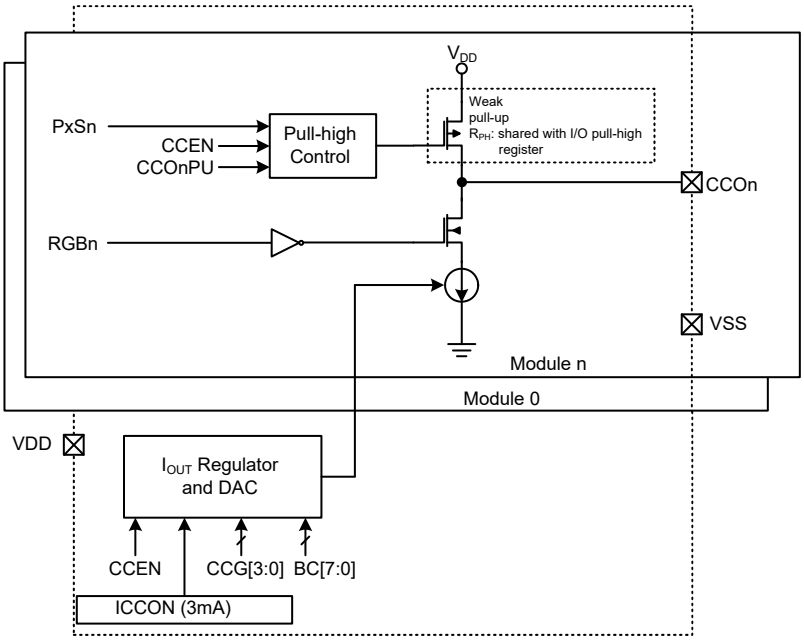
恒流 LED 驱动器

该单片机具有一个准确的恒流驱动器，专为 LED 显示器应用而设计，可提供 15 通道稳定的恒流输出来驱动 LED。

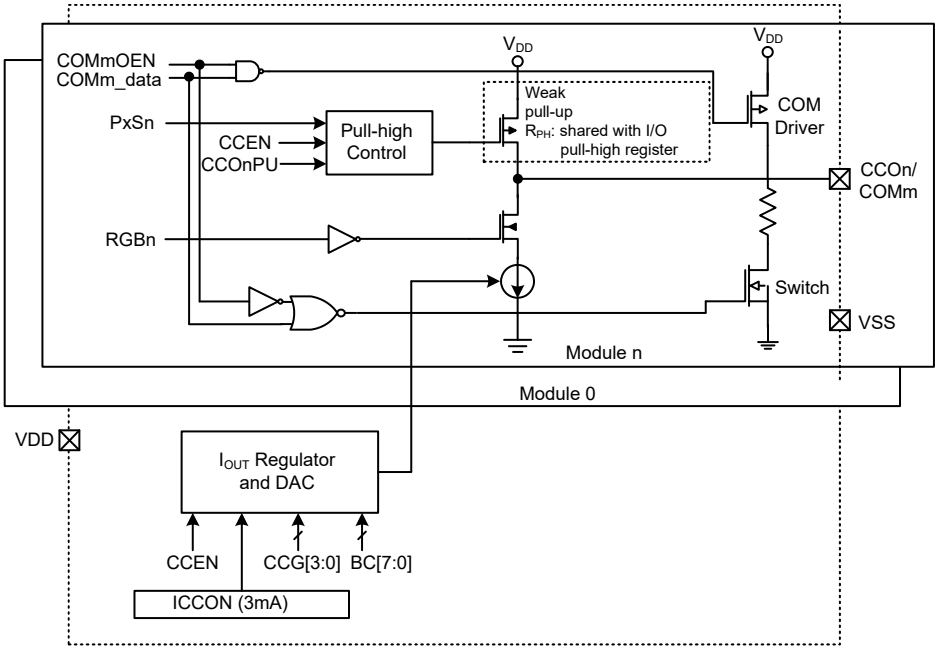
输出的恒流由电流增益选择位 CCG3~CCG0、RGBn PWM 输入和 CCOn 上拉功能确定。通道间电流变化小于 ±5%，不同设备之间的电流变化小于 ±10%。不论 LED 正向电压值是多少，输出电流都保持不变。

恒流可以用以下公式计算：

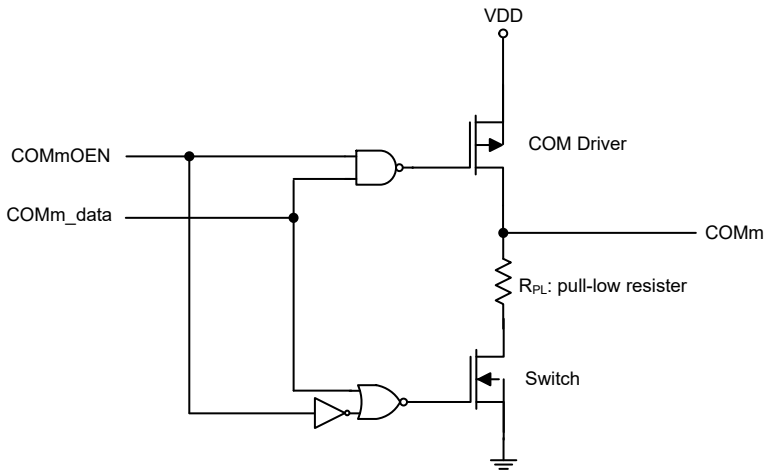
$$I_{CCOn}=3mA\times Gain$$



恒流 LED 驱动器方框图 (CCO)



恒流 LED 驱动器方框图 (CCO & COM)



恒流 LED 驱动器方框图 (COM)

注：1. n=0~14，m=0~3。
2. 模块 n 表示模块 0 ~ 模块 14。
3. RGBn 来源于 PWMnDATA 输出。
4. COMmOEN 来源于 PBS1 和 PCS0。
5. COMm_data 来源于 COMm 输出。

恒流 LED 驱动器方框图

恒流 LED 驱动器寄存器介绍

有三个与恒流 LED 驱动器相关的控制寄存器，即 CCS、CCOPU0 和 CCOPU1。CCS 寄存器用于恒流功能的控制和增益的选择。CCOPU0 和 CCOPU1 寄存器用于控制 CCOn 引脚的上拉功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CCS	CCEN	D6	D5	D4	CCG3	CCG2	CCG1	CCG0
CCOPU0	CCO7PU	CCO6PU	CCO5PU	CCO4PU	CCO3PU	CCO2PU	CCO1PU	CCO0PU
CCOPU1	—	CCO14PU	CCO13PU	CCO12PU	CCO11PU	CCO10PU	CCO9PU	CCO8PU
BC	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0

恒流 LED 驱动器寄存器列表

- **CCS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CCEN	D6	D5	D4	CCG3	CCG2	CCG1	CCG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	1	0	1	0

Bit 7 **CCEN**：恒流功能使能 / 除能控制位
0：除能
1：使能
该位用于控制恒流功能。当 CCEN 位被清零时，恒流功能将被除能，不管 RGBn 信号如何，CCOn 输出都将处于浮空状态。此时与 I/O 引脚共用的上拉电阻无法由 CCOnPU 位控制。
当引脚配置为 CCOn 输出功能，且 CCEN 位置高时，可通过相应的 CCOnPU 位来控制上拉电阻。当 CCO 功能使能且 RGBn 信号处于逻辑低状态时，CCOn 输出将作为一个恒流驱动器。如果 CCO 功能使能且 RGBn 信号处于逻辑高状态时，那么 CCOn 输出将处于高电平状态或浮空状态，这取决于上拉功能是否使能。

I/O 模式	CCEN	RGBn	CCOnPU	上拉使能 / 除能	CCOn 状态
Px	X	X	X	由 PxPU 控制	一般 I/O 状态
CCOn	0	X	X	除能	浮空
CCOn	1	0	0	由 CCOnPU 除能	输出低电平 (恒流)
CCOn	1	0	1	由 CCOnPU 使能	输出低电平 (影响恒流值)
CCOn	1	1	0	由 CCOnPU 除能	浮空
CCOn	1	1	1	由 CCOnPU 使能	上拉
注：Px：Port A1 & Port A[7:3] & Port B[7:0] & Port C0；X：无关					

Bit 6~4 **D6~D4**：保留位，这些位不可用且必须固定为“010”

Bit 3~0 **CCG3~CCG0**：最大恒流增益选择位
0000：I_{CCOn}=3mA
0001：I_{CCM}=6mA
0010：I_{CCM}=9mA
0011：I_{CCM}=12 mA
0100：I_{CCM}=15mA
0101：I_{CCM}=18mA
0110：I_{CCM}=21mA
0111：I_{CCM}=24 mA

1000：I_{CCM}=27mA
1001：I_{CCM}=30mA
1010：I_{CCM}=33mA
1011：I_{CCM}=36 mA
1100：I_{CCM}=39mA
1101：I_{CCM}=42mA
1110：I_{CCM}=45mA
1111：I_{CCM}=48 mA

- **CCOPU0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CCO7PU	CCO6PU	CCO5PU	CCO4PU	CCO3PU	CCO2PU	CCO1PU	CCO0PU
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **CCO7PU~CCO0PU**：CCO7~CCO0 上拉功能控制位
0：除能
1：使能

- **CCOPU1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	CCO14PU	CCO13PU	CCO12PU	CCO11PU	CCO10PU	CCO9PU	CCO8PU
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 未定义，读为“0”

Bit 6~0 **CCO14PU~CCO8PU**：CCO14~CCO8 上拉功能控制位
0：除能
1：使能

- **BC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **BC7~BC0**：恒流亮度控制位
0：除能
1：使能

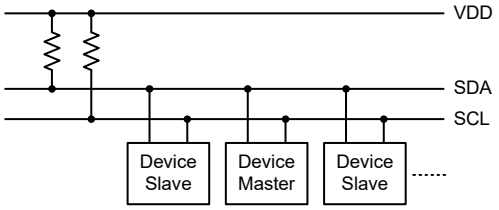
注：I_{CCOn}=I_{CCM} × 调光占空比。

BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0	数字调光占空比
0	0	0	0	0	0	0	0	I_{CCOn}=I_{CCM}×0/255
0	0	0	0	0	0	0	1	I_{CCOn}=I_{CCM}×1/255
0	0	0	0	0	0	1	0	I_{CCOn}=I_{CCM}×2/255
⋮								⋮
0	0	0	0	1	1	1	1	I_{CCOn}=I_{CCM}×15/255
0	0	0	1	0	0	0	0	I_{CCOn}=I_{CCM}×16/255
0	0	0	1	0	0	0	1	I_{CCOn}=I_{CCM}×17/255
⋮								⋮

BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0	数字调光占空比
0	1	1	1	1	1	1	1	$I_{CCOn}=I_{CCM}\times127/255$
1	0	0	0	0	0	0	0	$I_{CCOn}=I_{CCM}\times128/255$
1	0	0	0	0	0	0	1	$I_{CCOn}=I_{CCM}\times129/255$
⋮								⋮
1	1	1	1	1	1	0	0	$I_{CCOn}=I_{CCM}\times252/255$
1	1	1	1	1	1	0	1	$I_{CCOn}=I_{CCM}\times253/255$
1	1	1	1	1	1	1	0	$I_{CCOn}=I_{CCM}\times254/255$
1	1	1	1	1	1	1	1	$I_{CCOn}=I_{CCM}\times255/255$

I²C 接口

I²C 可以和传感器，EEPROM 内存等外部硬件接口进行通信。最初是由飞利浦公司研制，是适用于同步串行数据传输的双线式低速串行接口。I²C 接口具有两线通信，非常简单的通信协议和在同一总线上和多个设备进行通信的能力的优点，使之在很多的应用场合中大受欢迎。



I²C 主从总线连接图

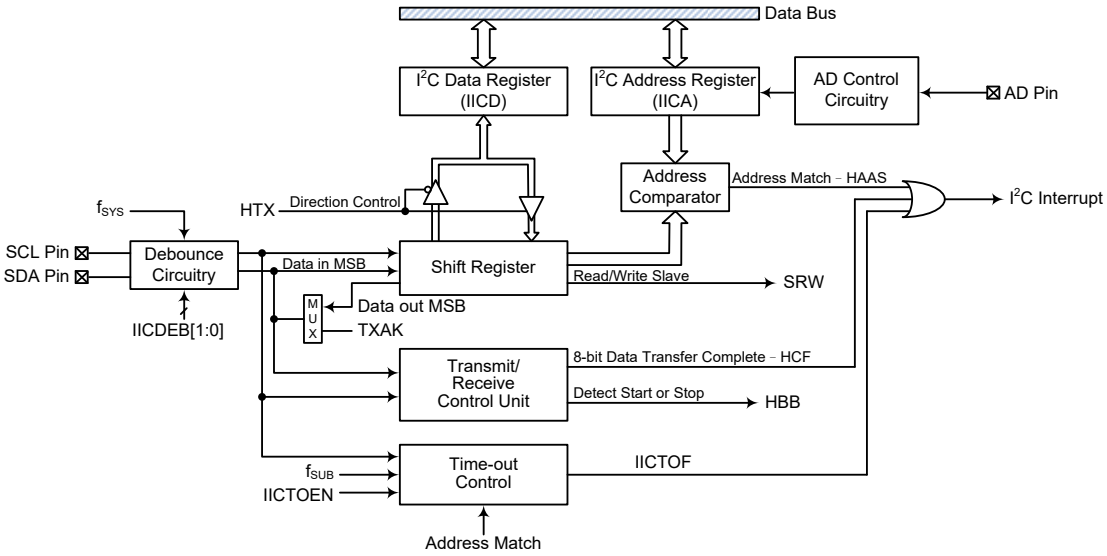
I²C 接口操作

I²C 串行接口是一个双线的接口，有一条串行数据线 SDA 和一条串行时钟线 SCL。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出口上都应加上拉电阻。应注意的是，I²C 总线上的每个设备都没有选择线，但分别与唯一的地址一一对应，用于 I²C 通信。

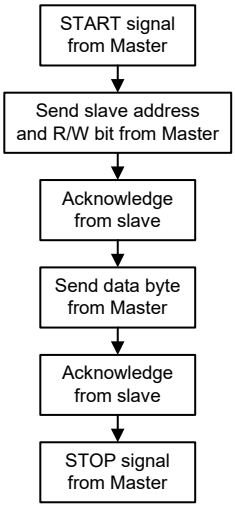
如果有两个设备通过双向的 I²C 总线进行通信，那么就存在一个主机和一个从机。主机和从机都可以用于发送和接收数据，但只有主机才可以控制总线动作。那些处于从机模式的设备，要在 I²C 总线上传输数据只有两种方式，一是从机发送模式，二是从机接收模式。

建议单片机不应在 I²C 通信期间进入空闲或休眠模式。

即使 I²C 设备被激活，与 SCL/SDA 引脚共用的 I/O 口上拉电阻控制功能仍有效，其上拉电阻功能由相应的上拉电阻控制寄存器控制。



I²C 方框图



I²C 接口操作

IICDEB1 和 IICDEB0 位决定 I²C 接口的去抖时间。这个功能可以使用内部时钟在外部时钟上增加一个去抖间隔，会减小时钟线上毛刺发生的可能性，以避免单片机发生误动作。如果选择了这个功能，去抖时间可以选择 2 个或 4 个系统时钟。为了达到需要的 I²C 数据传输速度，系统时钟 f_{SYS} 和 I²C 去抖时间之间存在一定的关系。I²C 标准模式或者快速模式下，用户需注意所选的系统时钟频率与标准匹配去抖时间的设置，其具体关系如下表所示。

I²C 去抖时间选择	I²C 标准模式 (100kHz0	I²C 快速模式 (400kHz)
无去抖时间	f_{SYS} > 2MHz	f_{SYS} > 5MHz
2 个系统时钟去抖时间	f_{SYS} > 4MHz	f_{SYS} > 10MHz
4 个系统时钟去抖时间	f_{SYS} > 8MHz	f_{SYS} > 20MHz

I²C 最小 f_{SYS} 频率要求

I²C 寄存器

I²C 总线有三个控制寄存器 IICC0、IICC1 和 IICTOC，一个从机地址寄存器 IICA 及一个数据寄存器 IICD。

寄存器名称	位							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
IICC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
IICD	D7	D6	D5	D4	D3	D2	D1	D0
IICA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	ADEN
IICTOC	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0

I²C 寄存器列表

I²C 数据寄存器

IICD 用于存储发送和接收的数据。在单片机尚未将数据写入到 I²C 总线中时，要传输的数据应存在 IICD 中。I²C 总线接收到数据之后，单片机就可以从 IICD 数据寄存器中读取。所有通过 I²C 传输或接收的数据都必须通过 IICD 实现。

- **IICD 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0 **D7~D0**：I²C 数据寄存器 bit 7 ~ bit 0

I²C 地址寄存器

IICA 寄存器用于存放 7 位从机地址，寄存器 IICA 中的 Bit 7~3 定义单片机从机高 5 位有效位地址。如果接至 I²C 的主机发送处的地址和寄存器 IICA 中存储的地址相符，那么就选中了这个从机。从机地址的低 2 位有效位由 IICA1~IICA0 或 AD 引脚连接状态决定，具体取决于 IICA 寄存器的 ADEN 位。

- **IICA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	ADEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~3 **IICA6~IICA2**：I²C 从机地址位 bit 7 ~ bit 3
IICA6~IICA2 是从机地址对应的 bit 6 ~ bit 2。

Bit 2~1 **IICA1~IICA0**：I²C 从机地址的低 2 位有效位 bit 1 ~ bit 0
ADEN=0，I²C 从机地址的低 2 位有效位 bit 1 ~ bit 0 由 IICA1 和 IICA0 决定。
IICA1 和 IICA0 可读可写。
ADEN=1，I²C 从机地址的低 2 位有效位 bit 1 ~ bit 0 由 AD 引脚状态决定
IICA1~IIAC0=
00：若 AD 引脚连接到 VSS
01：若 AD 引脚连接到 SCL
10：若 AD 引脚连接到 SDA
11：若 AD 引脚连接到 VDD

当 ADEN=1 时，从机地址的低 2 位有效位由 AD 引脚连接状态决定。IICA1 和 IICA0 只能被读取。IICA1 和 IICA0 可以从启动后的第二个 SCL 时钟的下降沿确定。

Bit 0 **ADEN**：I²C AD 引脚控制
0：除能 AD 引脚
1：使能 AD 引脚

I²C 控制寄存器

单片机中有三个控制 I²C 接口功能的寄存器，IICC0、IICC1 和 IICTOC。寄存器 IICC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 IICC1 包括多个用于表明 I²C 传输状态的相关标志位。另一个寄存器 IICTOC 用于控制 I²C 超时功能，将在相应章节描述。

- **IICC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 未定义，读为“0”

Bit 3~2 **IICDEB1~IICDEB0**：I²C 去抖时间选择位
00：无去抖时间
01：2 个系统时钟去抖时间
10：4 个系统时钟去抖时间
11：4 个系统时钟去抖时间
需要注意的是，如果系统时钟 f_{SYS} 来自 f_H 或者 IAMWU 位为 0，I²C 去抖电路才会正常工作。否则，SCL 和 SDA 脚会绕过去抖电路。

Bit 1 **IICEN**：I²C 控制位
0：除能
1：使能
此位为 I²C 接口的开 / 关控制位。此位为“0”时，I²C 接口除能，SDA 和 SCL 脚将失去 I²C 功能，I²C 工作电流减小到最小值。此位为“1”时，I²C 接口使能。当 IICEN 位由低到高转变时，I²C 控制寄存器中的设置，如 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 未定义，读为“0”

- **IICC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **HCF**：I²C 总线数据传输结束标志位
0：数据正在被传输
1：8 位数据传输完成
数据正在传输时该位为低。当 8 位数据传输完成时，此位为高并产生一个中断。

Bit 6 **HAAS**：I²C 地址匹配标志位
0：地址不匹配
1：地址匹配
此标志位用于决定从机地址是否与主机发送地址相同。若地址匹配此位为高，否则此位为低。

Bit 5 **HBB**：I²C 总线忙标志位
0：I²C 总线闲
1：I²C 总线忙
当检测到 START 信号时 I²C 忙，此位变为高电平。当检测到 STOP 信号时 I²C 总线停止，该位变为低电平。

Bit 4 **HTX**：从机处于发送或接收模式标志位
0：从机处于接收模式
1：从机处于发送模式

Bit 3 **TXAK**：I²C 总线发送确认标志位
0：从机发送确认标志
1：从机没有发送确认标志
单片机接收 8 位数据之后会将该位在第九个时钟传到总线上。如果单片机想要接收更多的数据，则应在接收数据之前将此位设置为“0”。

Bit 2 **SRW**：I²C 从机读 / 写位
0：从机应处于接收模式
1：从机应处于发送模式
SRW 位是从机读写位。决定主机是否希望传输或接收来自 I²C 总线的数据。当传输地址和从机的地址相同时，HAAS 位会被设置为高，从机将检测 SRW 位来决定进入发送模式还是接收模式。如果 SRW 位为高时，主机会请求从总线上读数据，此时从机处于发送模式。当 SRW 位为“0”时，主机往总线上写数据，从机处于接收模式以读取该数据。

Bit 1 **IAMWU**：I²C 地址匹配唤醒控制位
0：除能
1：使能
此位应设置为“1”使能 I²C 地址匹配以使系统从休眠或空闲模式中唤醒。若进入休眠或空闲模式前 IAMWU 已经置高以使能 I²C 地址匹配唤醒功能，在系统唤醒后须软件清除此位以确保单片机正确地运行。

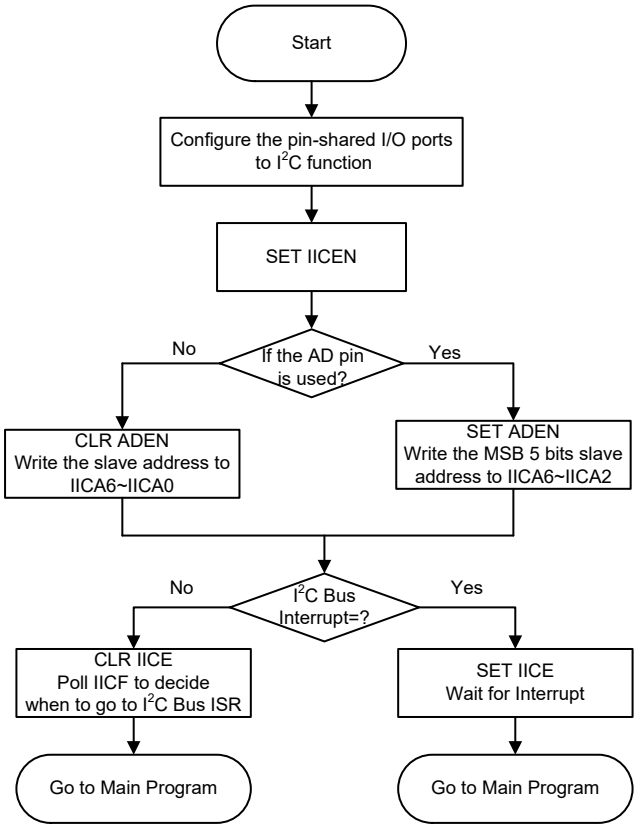
Bit 0 **RXAK**：I²C 总线接收确认标志位
0：从机接收到确认标志
1：从机没有接收到确认标志
RXAK 位是接收确认标志位。如果 RXAK 位被重设为“0”即 8 位数据传输之后，从机在第九个时钟有接收到一个正确的确认位。如果从机处于发送状态，发送方会检查 RXAK 位来判断接收方是否愿意继续接收下一个字节。因此直到 RXAK 为“1”时，传输方停止发送数据。这时，传输方将释放 SDA 线，主机发出停止信号。

I²C 总线通信

I²C 总线上的通信需要四步完成，一个起始信号，一个从机地址发送，一个数据传输，还有一个停止信号。当起始信号被写入 I²C 总线时，总线上的所有从机都会接收到这个起始信号并且被通知总线上会即将有数据到达。数据的前 7 位是从机地址，高位在前，低位在后。如果发出的地址和从机地址匹配，IICC1 寄存器的 HAAS 位会被置位，同时产生 I²C 中断。进入中断服务程序后，系统要检测 HAAS 位和 IICTOF 位，以判断 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕，或是来自 I²C 超时。在数据传递中，注意的是，在 7 位从机地址被发送后，接下来的一位，即第 8 位，是读 / 写控制位，该位的值会反映到 SRW 位中。从机通过检测 SRW 位以确定主控制器是要进入发送模式还是接收模式。在 I²C 总线开始传送数据前，需要先初始化 I²C 总线，初始化 I²C 总线步骤如下：

- 步骤 1
  将相应的共用 I/O 引脚配置为 I²C 引脚功能。设置 IICC0 寄存器中的 IICEN 位为“1”，以使能 I²C 总线。

- 步骤 2
  根据应用需求，设置单片机的从机地址。
  若使用 AD 引脚，跳至步骤 3
  若未使用 AD 引脚，跳至步骤 4
- 步骤 3
  设置 IICA 寄存器中的 ADEN 位为“1”，以使能 AD 引脚，然后根据应用需求，连接 AD 引脚至 VDD，VSS，SCL 或 SDA。向 IICA 寄存器的 IICA6~IICA2 位写入从机地址的高 5 位有效位。IICA1~IICA0 位的值由 AD 引脚连接状态决定。
- 步骤 4
  清除 IICA 寄存器中的 ADEN 位为“0”，以除能 AD 引脚。向 IICA 寄存器的 IICA6~IICA0 位写入从机地址。
- 步骤 5
  设置中断控制寄存器的 IICE 中断使能位以使能 I^2C 中断。



I^2C 总线初始化流程图

I²C 总线起始信号

起始信号只能由连接 I²C 总线主机产生，而不是由只做从机的 MCU 产生。总线上的所有从机都可以侦测到起始信号。如果有从机侦测到起始信号，则表明 I²C 总线处于忙碌状态，并会置位 HBB。起始信号是指在 SCL 为高电平时，SDA 线上发生从高到低的电平变化。

从机地址

总线上的所有从机都会侦测由主机发出的起始信号。发送起始信号后，紧接着主机会发送从机地址以选择要进行数据传输的从机。所有在 I²C 总线上的从机接收到 7 位地址数据后，都会将其与各自内部的地址进行比较。如果从机从主机上接收到的地址与自身内部的地址相匹配，则会产生一个 I²C 总线中断信号。地址位接下来的一位为读 / 写状态位 (即第 8 位)，将被保存到 IICC1 寄存器的 SRW 位，随后发出一个低电平应答信号 (即第 9 位)。当单片机从机的地址匹配时，会将状态标志位 HAAS 置位。

I²C 总线有三个中断源，当程序运行至中断服务子程序时，通过检测 HAAS 位和 IICTOF 位，以确定 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕，或是来自 I²C 超时。当是从机地址匹配发生中断时，则从机或是用于发送模式并将数据写进 IICD 寄存器，或是用于接收模式并从 IICD 寄存器中读取空值以释放 SCL 线。

I²C 总线读 / 写信号

IICC1 寄存器的 SRW 位用来表示主机是要从 I²C 总线上读取数据还是要将数据写到 I²C 总线上。从机则通过检测该位以确定自己是作为发送方还是接收方。当 SRW 置“1”，表示主机要从 I²C 总线上读取数据，从机则作为发送方，将数据写到 I²C 总线；当 SRW 清“0”，表示主机要写数据到 I²C 总线上，从机则做为接收方，从 I²C 总线上读取数据。

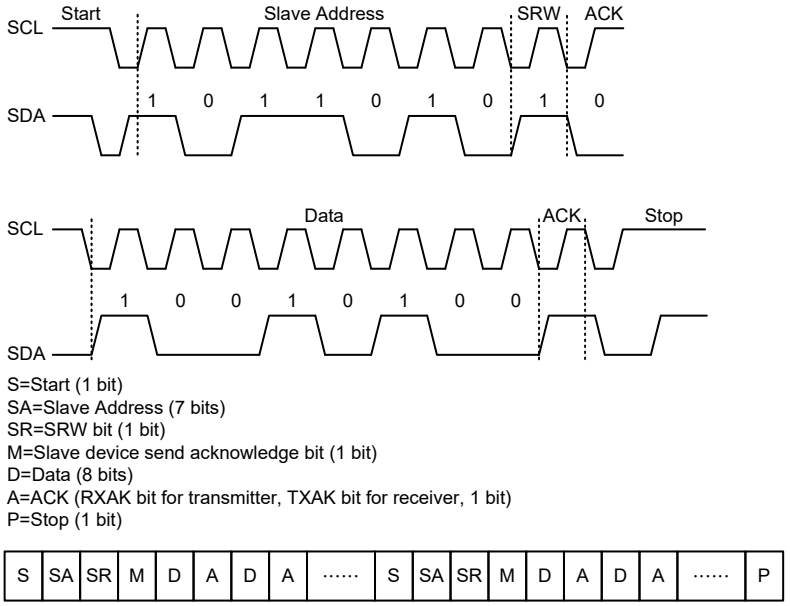
I²C 总线从机地址确认信号

主机发送呼叫地址后，当 I²C 总线上的任何从机内部地址与其匹配时，会发送一个应答信号。此应答信号会通知主机有从机已经接收到了呼叫地址。如果主机没有收到应答信号，则主机必须发送停止 (STOP) 信号以结束通信。当 HAAS 为高时，表示从机接收到的地址与自己内部地址匹配，则从机需检查 SRW 位，以确定自己是作为发送方还是作为接收方。如果 SRW 位为高，从机须设置成发送方，这样会置位 IICC1 寄存器的 HTX 位。如果 SRW 位为低，从机须设置成接收方，这样会清零 IICC1 寄存器的 HTX 位。

I²C 总线数据和确认信号

在从机确认接收到从地址后，会进行 8 位宽度的数据传输。这个数据传输顺序是的高位在前，低位在后。接收方在接收到 8 位数据后必须发出一个应答信号 (“0”) 以继续接收下一个数据。如果发送方没接收到应答信号，发送方将释放 SDA 线，同时，主机将发出 STOP 信号以释放 I²C 总线。所传送的数据存储在 IICD 寄存器中。如果设置成发送方，从机必须先将欲传输的数据写到 IICD 寄存器中；如果设置成接收方，从机必须从 IICD 寄存器读取数据。

当接收器想要继续接收下一个数据时，必须在第 9 个时钟发出应答信号 (TXAK)。被设为发送方的从机将检测寄存器 IICC1 中的 RXAK 位以判断是否传输下一个字节的数据，如果单片机不传输下一个字节，那么它将释放 SDA 线并等待接收主机的停止信号。

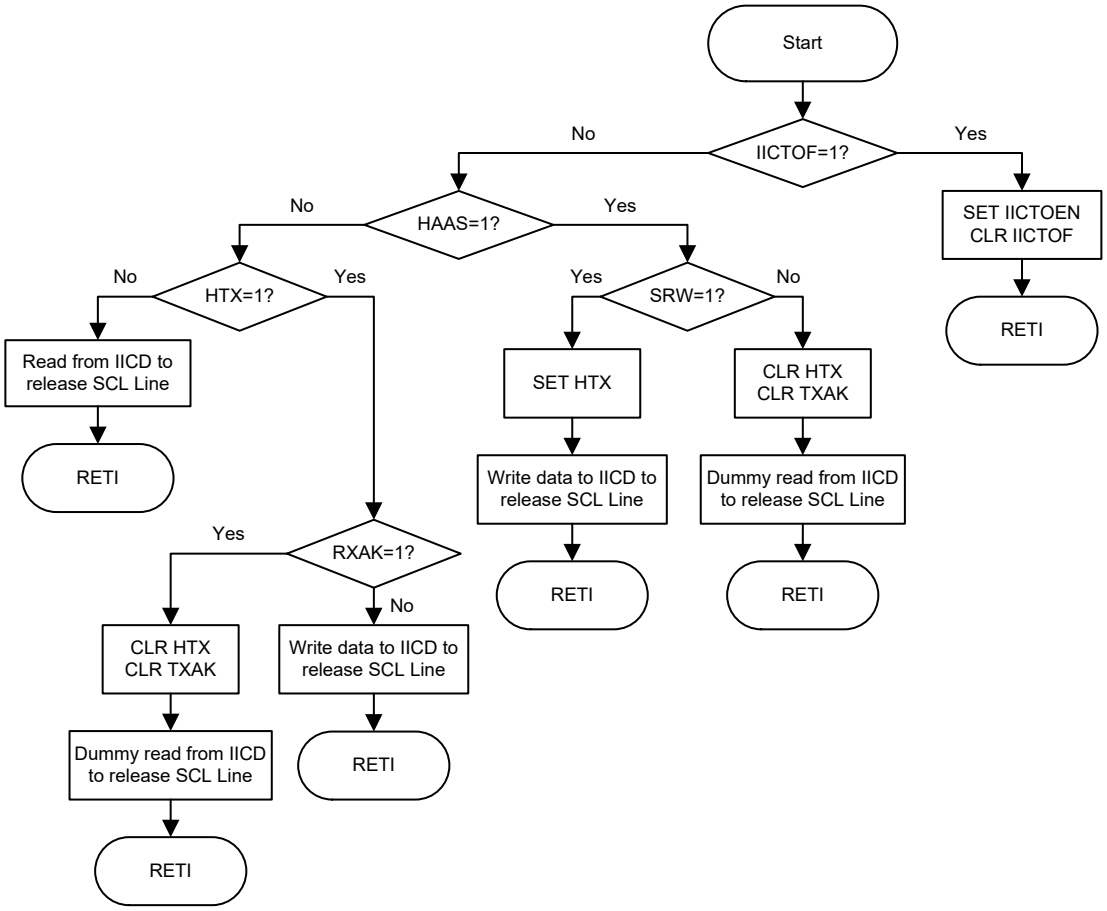


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	·····	S	SA	SR	M	D	A	D	A	·····	P

注：当从机地址匹配时，单片机必须选择设置为发送模式还是接收模式。若设置为发送模式，需写数据至 IICD 寄存器；若设置为接收模式，需立即从 IICD 寄存器中虚读数据以释放 SCL 线。

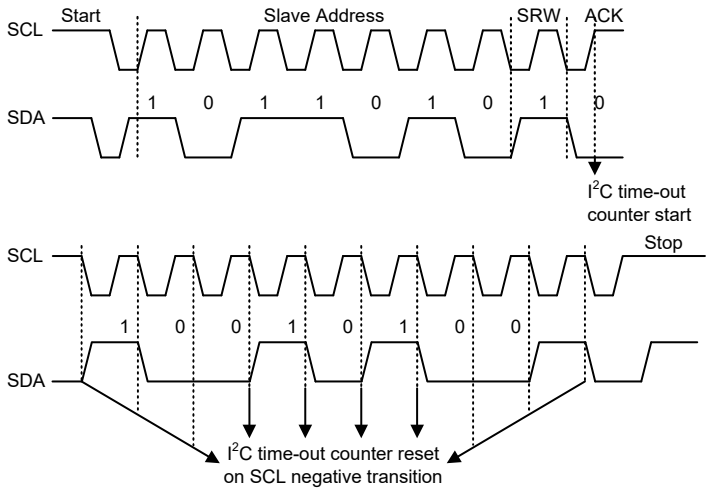
I^2C 通信时序图



I²C 总线 ISR 流程图

I²C 超时控制

超时功能可减少由于接收错误的时钟源而引起的 I²C 锁死问题。在一定时间内如果 I²C 总线未接收到时钟源，则 I²C 电路和寄存器将会复位。超时计数器在 I²C 总线接收到“START”信号和“地址匹配”条件时开始计数，并在 SCL 下降沿处清零。在下一个 SCL 下降沿来临之前，如果等待时间大于 IICTOC 寄存器设定的超时时间，则会发生超时现象。当 I²C “STOP”条件发生时，超时计数器将停止计数。



I²C 超时

当 I²C 超时计数器溢出时，计数器停止且 IICTOEN 位清零，且 IICTOF 位被置高以表明超时计数器中断发生。超时时将产生一个中断且共用 I²C 中断向量。当 I²C 超时发生时，超时计数器中断使用的也是 I²C 中断向量。当 I²C 超时发生时，I²C 内部电路会被复位，寄存器也将发生如下复位情况。

寄存器	I²C 超时发生后
IICD, IICA, IICC0	保持不变
IICC1	复位至 POR

超时发生后的 I²C 寄存器

IICTOF 标志位由应用程序清零。64 个超时时钟周期可由 IICTOC 寄存器里的 IICTOS 位段来设置。超时时间的计算方法如下：

$$((1\sim64)\times32)/f_{SUB}$$

由此可得超时周期范围为 1ms~64ms。

- **IICTOC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **IICTOEN**：I²C 超时控制位
0：除能
1：使能

Bit 6 **IICTOF**：I²C 超时标志位
0：超时未发生
1：超时发生

Bit 5~0 **IICTOS5~IICTOS0**：I²C 超时时间选择位
I²C 超时时钟源是 $f_{SUB}/32$
I²C 超时时间计算公式：(IICTOS[5:0]+1)×(32/f_{SUB})

中断

中断是单片机一个重要功能。当外部事件或内部功能如定时器模块有效，并且产生中断时，系统会暂时中止当前的程序而转到执行相对应的中断服务程序。此单片机仅提供内部中断功能。内部中断由各种内部功能，如定时器模块、时基、I²C 接口和级联式收发器接口等产生。

中断寄存器

中断控制基本上是在一定单片机条件发生时设置请求标志位，应用程序中中断使能位的设置是通过位于专用数据存储器中的一系列寄存器控制的，总的分为两类。第一类是 INTC0~INTC1 寄存器，用于设置基本的中断；第二类是 MFI 寄存器，用于设置多功能中断。

寄存器中含有中断控制位和中断请求标志位。中断控制位用于使能或除能各种中断，中断请求标志位用于存放当前中断请求的状态。它们都按照特定的模式命名，前面表示中断类型的缩写，紧接着的字母“E”代表使能 / 除能位，“F”代表请求标志位。

功能	使能位	请求标志
总中断	EMI	—
多功能	MFE	MFF
时基	TBE	TBF
I²C 接口	IICE	IICF
级联式收发器接口	CASINTE	CASINTF
CTM	CTMPE	CTMPF
	CTMAE	CTMAF

中断寄存器位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTC0	—	CASINTF	MFF	TBF	CASINTE	MFE	TBE	EMI
INTC1	—	—	—	IICF	—	—	—	IICE
MFI	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE

中断寄存器列表

- **INTC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	CASINTF	MFF	TBF	CASINTE	MFE	TBE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7　未定义，读为“0”

Bit 6　**CASINTF**：级联式收发器中断请求标志位
0：无请求
1：中断请求

Bit 5　**MFF**：多功能中断请求标志位
0：无请求
1：中断请求

Bit 4 **TBF**：时基中断请求标志位
0：无请求
1：中断请求

Bit 3 **CASINTE**：级联式收发器中断控制位
0：除能
1：使能

Bit 2 **MFE**：多功能中断控制位
0：除能
1：使能

Bit 1 **TBE**：时基中断控制位
0：除能
1：使能

Bit 0 **EMI**：总中断控制位
0：除能
1：使能

- **INTC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	IICF	—	—	—	IICE
R/W	—	—	—	R/W	—	—	—	R/W
POR	—	—	—	0	—	—	—	0

Bit 7~5 未定义，读为“0”

Bit 4 **IICF**：I^2C 接口中断请求标志位
0：无请求
1：中断请求

Bit 3~1 未定义，读为“0”

Bit 0 **IICE**：I^2C 接口中断控制位
0：除能
1：使能

- **MFI 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 未定义，读为“0”

Bit 5 **CTMAF**：CTM 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4 **CTMPF**：CTM 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求

Bit 3~2 未定义，读为“0”

Bit 1 **CTMAE**：CTM 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0 **CTMPE**：CTM 比较器 P 匹配中断控制位
0：除能
1：使能

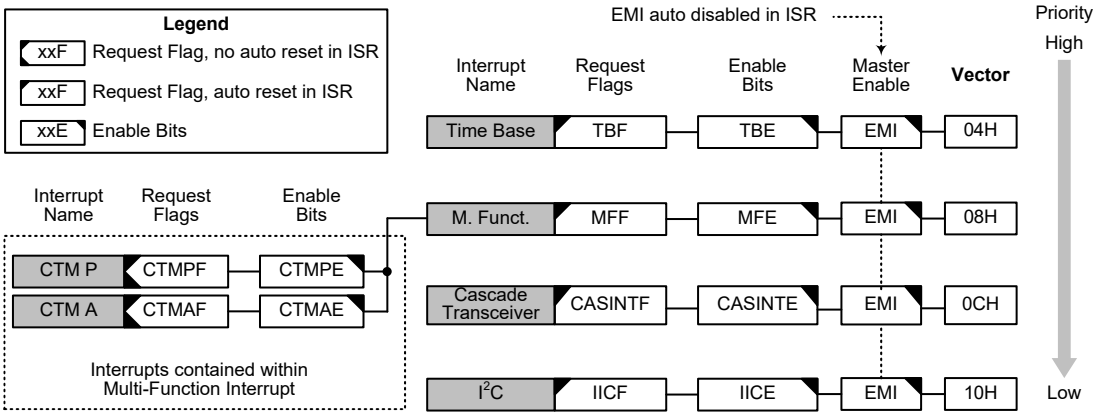
中断操作

若中断事件条件产生，如一个 TM 比较器 P、比较器 A 匹配等等，相关中断请求标志将置起。中断标志产生后程序是否会跳转至相关中断向量执行是由中断使能位的条件决定的。若使能位为“1”，程序将跳至相关中断向量中执行；若使能位为“0”，即使中断请求标志置起中断也不会发生，程序也不会跳转至相关中断向量执行。若总中断使能位为“0”，所有中断都将除能。

当中断发生时，下条指令的地址将被压入堆栈。相应的中断向量地址加载至 PC 中。系统将从此向量取下条指令。中断向量处通常为“JMP”指令，以跳转到相应的中断服务程序。中断服务程序必须以“RETI”指令返回至主程序，以继续执行原来的程序。

各个中断使能位以及相应的请求标志位，以优先级的次序显示在下图。一些中断源有自己的向量，但是有些中断却共用多功能中断向量。一旦中断子程序被响应，系统将自动清除 EMI 位，所有其它的中断将被屏蔽，这个方式可以防止任何进一步的中断嵌套。其它中断请求可能发生在此期间，虽然中断不会立即响应，但是中断请求标志位会被记录。

如果某个中断服务子程序正在执行时，有另一个中断要求立即响应，那么 EMI 位应在程序进入中断子程序后置位，以允许此中断嵌套。如果堆栈已满，即使此中断使能，中断请求也不会被响应，直到 SP 减少为止。如果要求立刻动作，则堆栈必须避免成为储满状态。请求同时发生时，执行优先级如下流程图所示。所有被置起的中断请求标志都可把单片机从休眠或空闲模式中唤醒，若要防止唤醒动作发生，在单片机进入休眠或空闲模式前应将相应的标志置起。



中断结构

I²C 中断

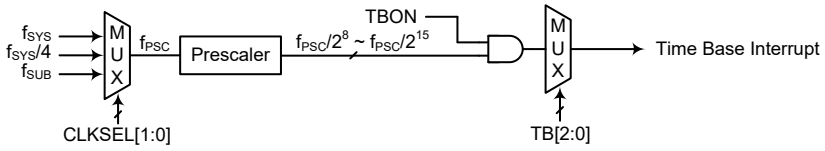
当一个字节数据已由 I²C 接口接收或发送完，或 I²C 从机地址匹配，或 I²C 超时，中断请求标志 IICF 被置位，I²C 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和串行接口中断使能位 IICE 需先被置位。当中断使能，堆栈未满且以上任一种情况发生时，可跳转至相关中断向量子程序中执行。当响应中断服务子程序时，串行接口中断标志位 IICF 会自动复位且 EMI 将被自动清零以除能其它中断。

时基中断

时基中断提供一个固定周期的中断信号，由它的定时器功能产生溢出信号控制。当中断请求标志 TBF 被置位时，中断请求发生。当总中断使能位 EMI 和时基

使能位 TBE 被置位，允许程序跳转到相应的中断向量地址。当中断使能，堆栈未满且时基溢出时，将调用它的中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 TBF 会自动复位且 EMI 位会被清零以除能其它中断。

时基中断的目的是提供一个固定周期的中断信号。其时钟源 f_{PSC} 来自内部时钟源 f_{SYS}、$f_{SYS}/4$ 或 f_{SUB}。f_{PSC} 输入时钟首先经过分频器，分频率由程序设置 TBC 寄存器相关位获取合适的分频值以提供更长的时基中断周期。相应的控制时基中断周期的时钟源可通过 PSCR 寄存器的 CLKSEL1 和 CLKSEL0 位选择。



时基中断

- **PSCR 寄存器**

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

Bit 7~2　未定义，读为“0”

Bit 1~0　**CLKSEL1~CLKSEL0**：分频器时钟源选择
00：f_{SYS}
01：$f_{SYS}/4$
1x：f_{SUB}

- **TBC 寄存器**

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

Bit 7　**TBON**：时基使能 / 除能控制位
0：除能
1：使能

Bit 6~3　未定义，读为“0”

Bit 2~0　**TB2~TB0**：选择时基溢出周期位
000：$2^8/f_{PSC}$
001：$2^9/f_{PSC}$
010：$2^{10}/f_{PSC}$
011：$2^{11}/f_{PSC}$
100：$2^{12}/f_{PSC}$
101：$2^{13}/f_{PSC}$
110：$2^{14}/f_{PSC}$
111：$2^{15}/f_{PSC}$

多功能中断

此单片机中有一个多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即 TM 中断。

当多功能中断中任何一种中断请求标志 MFF 被置位，多功能中断请求产生。当中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，相关的多功能请求标志位会自动复位且 EMI 位会自动清零以除能其它中断。

但必须注意的是，在中断响应时，虽然多功能中断标志会自动复位，但多功能中断源的请求标志位，即 TM 中断的请求标志位不会自动复位，必须由应用程序清零。

级联式收发器接口中断

当级联式收发器 RX 接收的数据格式发生错误、级联电路复位、级联式收发器 TX 移位寄存器为空、或级联式收发器 24 位或 24×N 位 RX 移位寄存器已满，级联式收发器接口中断请求标志 CASINTF 被置位，级联式收发器接口中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和级联式收发器接口中断使能位 CASINTE 需先被置位。当中断使能，堆栈未满且上述任一情况发生时，可跳转至级联式收发器接口中断向量子程序中执行。当级联式收发器接口中断响应时，相应的 CASINTF 请求标志会自动复位且 EMI 位也会被清零以除能其它中断。

TM 中断

简易型 TM 有两个中断，都属于多功能中断。该简易型 TM 有两个中断请求标志位 CTMPF 和 CTMAF 及两个使能位 CTMPE 和 CTMAE。当 CTM 比较器 P、A 匹配情况发生时，相应 CTM 中断请求标志被置位，CTM 中断请求产生。

若要程序跳转到相应中断向量地址，总中断控制位 EMI、相应 CTM 中断使能位和相关多功能中断使能位 MFE 需先被置位。当中断使能，堆栈未满且 CTM 比较器匹配情况发生时，可跳转至相关多功能中断向量子程序中执行。当 CTM 中断响应，EMI 将被自动清零以除能其它中断，相关 MFF 标志也可自动清除，但 CTM 中断请求标志需在应用程序中手动清除。

中断唤醒功能

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

编程注意事项

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

多功能中断中所含中断相应程序执行时，多功能中断请求标志 MFF 可以自动清零，但各自的请求标志需在应用程序中手动清除。

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

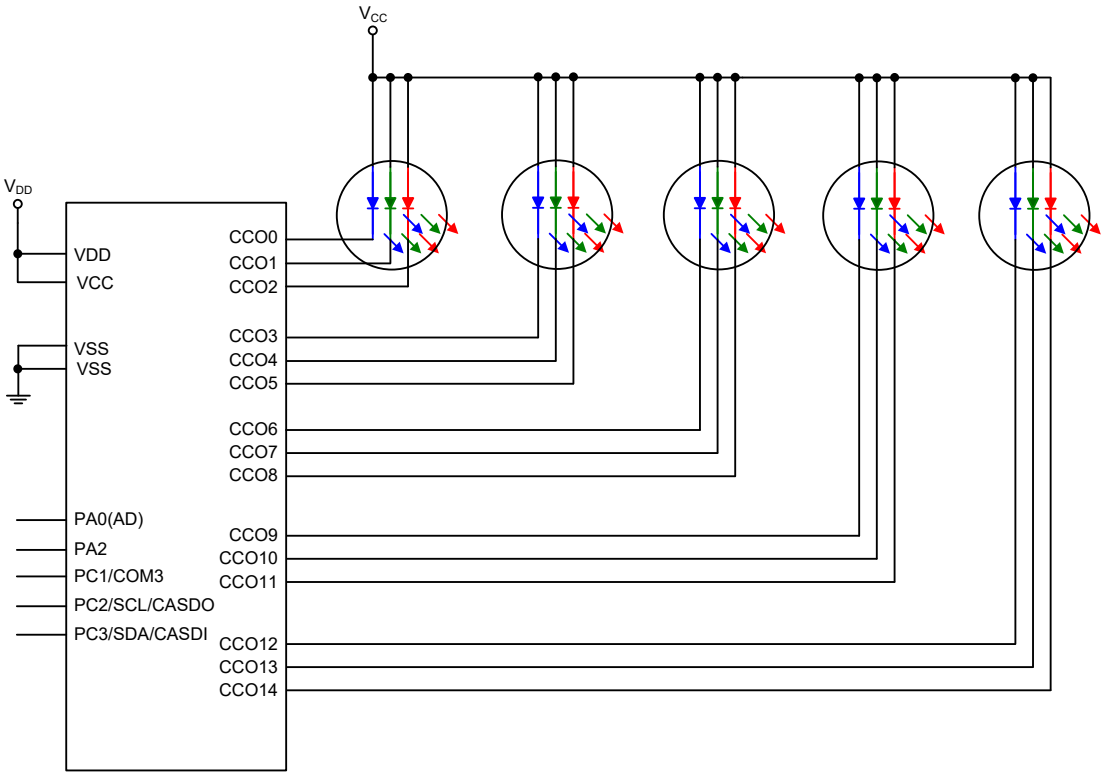
所有中断在休眠或空闲模式下都具有唤醒功能，当中断请求标志发生由低到高的转变时都可产生唤醒功能。若要避免相应中断产生唤醒动作，在单片机进入休眠或空闲模式前需先将相应请求标志置为高。

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

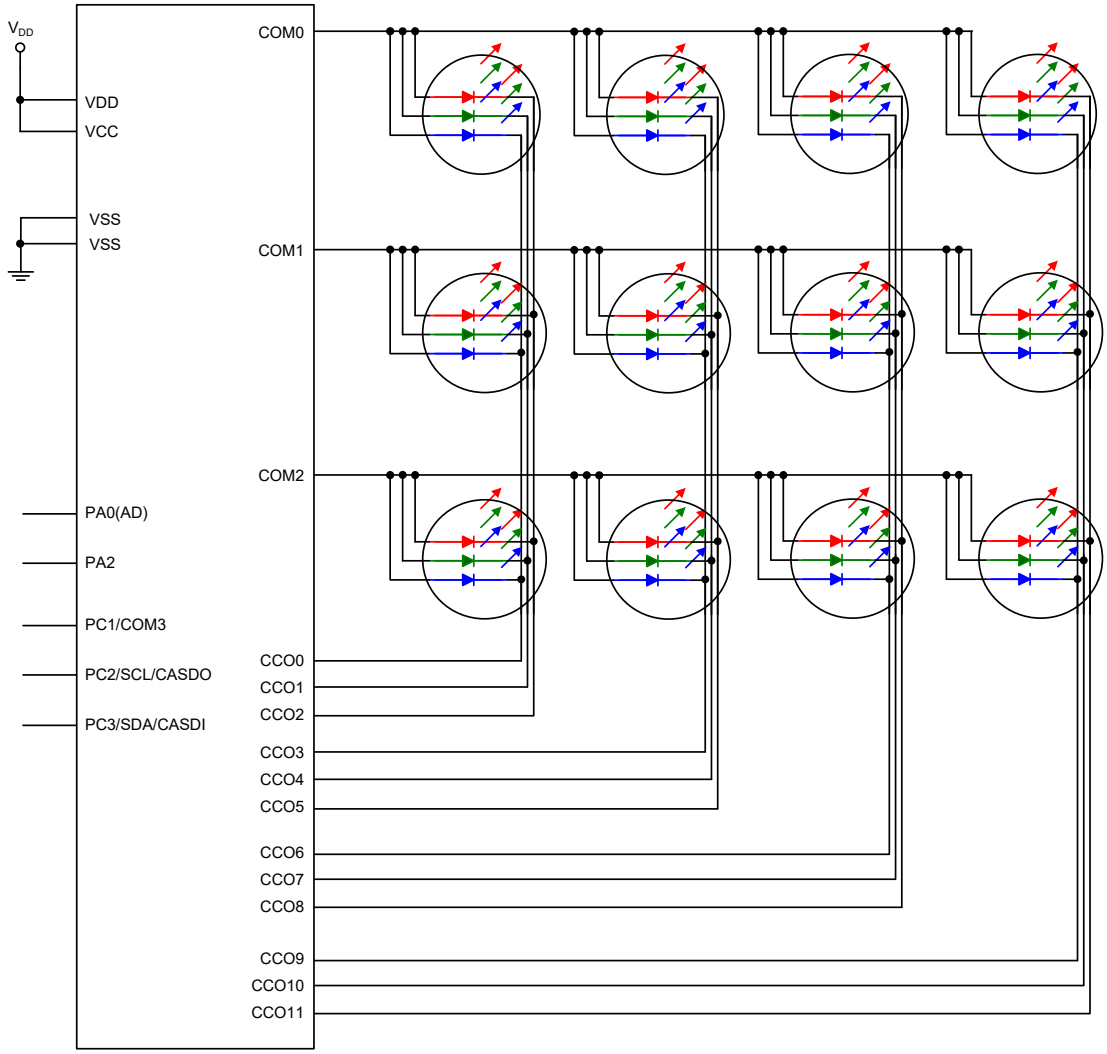
若从中断子程序中返回可执行 RET 或 RETI 指令。除了能返回至主程序外，RETI 指令还能自动设置 EMI 位为高，允许进一步中断。RET 指令只能返回至主程序，清除 EMI 位，除能进一步中断。

应用电路

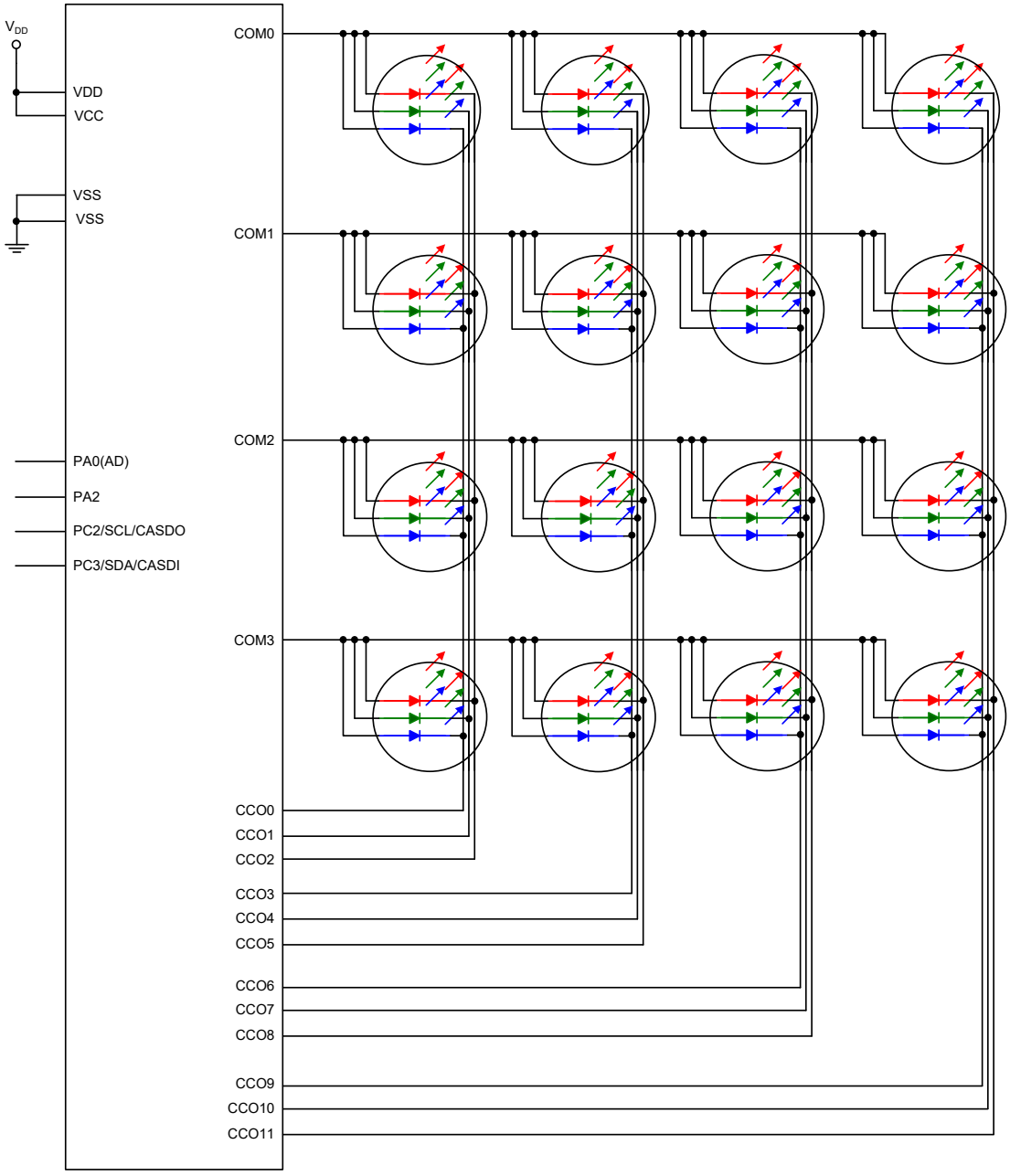
直推模式



分时扫描模式 – 3×COM


VDD
VDD
VCC
VSS
VSS
PA0(AD)
PA2
PC1/COM3
PC2/SCL/CASDO
PC3/SDA/CASDI
COM0
COM1
COM2
CCO0
CCO1
CCO2
CCO3
CCO4
CCO5
CCO6
CCO7
CCO8
CCO9
CCO10
CCO11

分时扫描模式 – 4×COM


VDD
VDD
VCC
VSS
VSS
PA0(AD)
PA2
PC2/SCL/CASDO
PC3/SDA/CASDI
COM0
COM1
COM2
COM3
CCO0
CCO1
CCO2
CCO3
CCO4
CCO5
CCO6
CCO7
CCO8
CCO9
CCO10
CCO11

指令集

简介

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

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

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

数据的传送

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

算术运算

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

逻辑和移位运算

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

分支和控制转换

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

位运算

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

查表运算

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

其它运算

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

指令集概要

下表中说明了按功能分类的指令集，用户可以将该表作为基本的指令参考。

惯例

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

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

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

注：1. 对跳转指令而言，如果比较的结果牵涉到跳转即需 2 个周期，如果没有发生跳转，则只需一个周期。
2. 任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

指令定义

ADC A, [m] Add Data Memory to ACC with Carry
指令说明 将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
功能表示 ACC ← ACC + [m] + C
影响标志位 OV、Z、AC、C

ADCM A, [m] Add ACC to Data Memory with Carry
指令说明 将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。
功能表示 [m] ←ACC + [m] + C
影响标志位 OV、Z、AC、C

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

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

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

AND A, [m] Logical AND Data Memory to ACC
指令说明 将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。
功能表示 ACC ← ACC “AND” [m]
影响标志位 Z

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

ANDM A, [m]	Logical AND ACC to Data Memory
指令说明	将指定数据存储器内容和累加器中的数据做逻辑与， 结果存放到数据存储器。
功能表示	[m] ← ACC“AND”[m]
影响标志位	Z

CALL addr	Subroutine call
指令说明	无条件地调用指定地址的子程序，此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈，接着载入指定 地址并从新地址继续执行程序，由于此指令需要额外的运 算，所以为一个 2 周期的指令。
功能表示	Stack ← Program Counter + 1 Program Counter ← addr
影响标志位	无

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

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

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

CPL [m] Complement Data Memory
指令说明 将指定数据存储器中的每一位取逻辑反，
相当于从 1 变 0 或 0 变 1。
功能表示 [m] ← $[\overline{m}]$
影响标志位 Z

CPLA [m] Complement Data Memory with result in ACC
指令说明 将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，而结果被储存回累加器且数据存储器中的内容不变。
功能表示 ACC←$[\overline{m}]$
影响标志位 Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory
指令说明 将累加器中的内容转换为 BCD (二进制转成十进制) 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对原值加“6”，否则原值保持不变；如果高四位的值大于“9”或 C=1，那么 BCD 调整就执行对原值加“6”。BCD 转换实质上是根据累加器和标志位执行 00H，06H，60H 或 66H 的加法运算，结果存放到数据存储器。只有进位标志位 C 受影响，用来指示原始 BCD 的和是否大于 100，并可以进行双精度十进制数的加法运算。
功能表示 [m] ← ACC + 00H 或
[m] ← ACC + 06H 或
[m] ← ACC + 60H 或
[m] ← ACC + 66H
影响标志位 C

DEC [m] Decrement Data Memory
指令说明 将指定数据存储器内容减 1。
功能表示 [m] ← [m] – 1
影响标志位 Z

DECA [m] Decrement Data Memory with result in ACC
指令说明 将指定数据存储器的内容减 1，把结果存放回累加器并保持指定数据存储器的内容不变。
功能表示 ACC ← [m] – 1
影响标志位 Z

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

INC [m]	Increment Data Memory
指令说明	将指定数据存储器的内容加 1。
功能表示	[m] ← [m] + 1
影响标志位	Z

INCA [m]	Increment Data Memory with result in ACC
指令说明	将指定数据存储器的内容加 1，结果存放回累加器并保持指定的数据存储器内容不变。
功能表示	ACC ← [m] + 1
影响标志位	Z

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

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

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

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

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

OR A, [m]	Logical OR Data Memory to ACC
指令说明	将累加器中的数据和指定的数据存储器内容逻辑或， 结果存放到累加器。
功能表示	ACC ← ACC“OR”[m]
影响标志位	Z

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

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

RET	Return from subroutine
指令说明	将堆栈寄存器中的程序计数器值恢复， 程序由取回的地址继续执行。
功能表示	Program Counter←Stack
影响标志位	无

RET A, x	Return from subroutine and load immediate data to ACC
指令说明	将堆栈寄存器中的程序计数器值恢复且累加器载入指定的 立即数，程序由取回的地址继续执行。
功能表示	Program Counter ← Stack ACC←x
影响标志位	无

RETI Return from interrupt

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

功能表示 Program Counter ←Stack
EMI ← 1

影响标志位 无

RL [m] Rotate Data Memory left

指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。

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

影响标志位 无

RLA [m] Rotate Data Memory left with result in ACC

指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。

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

影响标志位 无

RLC [m] Rotate Data Memory Left through Carry

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

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

影响标志位 C

RLC A [m] Rotate Data Memory left through Carry with result in ACC

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

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

影响标志位 C

RR [m] Rotate Data Memory right
指令说明 将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7 ← [m].0
影响标志位 无

RRA [m] Rotate Data Memory right with result in ACC
指令说明 将指定数据存储器的内容循环右移 1 位，第 0 位移到第 7 位，移位结果存放到累加器，而指定数据存储器的内容保持不变。
功能表示 ACC.i ← [m].(i+1) (i=0~6)
ACC.7 ← [m].0
影响标志位 无

RRC [m] Rotate Data Memory right through Carry
指令说明 将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位。
功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7← C
C ← [m].0
影响标志位 C

RRCA [m] Rotate Data Memory right through Carry with result in ACC
指令说明 将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。
功能表示 ACC.i ← [m].(i+1) (i=0~6)
ACC.7 ← C
C ← [m].0
影响标志位 C

SBC A, [m] Subtract Data Memory from ACC with Carry
指令说明 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C

SBCM A, [m] Subtract Data Memory from ACC with Carry and result in Data Memory
指令说明 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 [m] ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C

SDZ [m] Skip if Decrement Data Memory is 0
指令说明 将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 [m] ← [m] – 1，如果 [m]=0 跳过下一条指令执行
影响标志位 无

SDZA [m] Skip if decrement Data Memory is zero with result in ACC
指令说明 将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 ACC ← [m] – 1，如果 ACC=0 跳过下一条指令执行
影响标志位 无

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

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

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

SIZA [m] Skip if increment Data Memory is zero with result in ACC
指令说明 将指定数据存储器的内容加 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果会被存放到累加器，但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 ACC ←[m] + 1，如果 ACC=0 跳过下一条指令执行
影响标志位 无

SNZ [m].i Skip if bit i of Data Memory is not 0
指令说明 判断指定数据存储器的第 i 位，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果为 0，则程序继续执行下一条指令。
功能表示 如果 [m].i≠0，跳过下一条指令执行
影响标志位 无

SUB A, [m] Subtract Data Memory from ACC
指令说明 将累加器的内容减去指定的数据存储器的数据，把结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m]
影响标志位 OV、Z、AC、C

SUBM A, [m] Subtract Data Memory from ACC with result in Data Memory
指令说明 将累加器的内容减去指定数据存储器的数据，结果存放到指定的数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 [m] ← ACC – [m]
影响标志位 OV、Z、AC、C

SUB A, x Subtract immediate Data from ACC
指令说明 将累加器的内容减去立即数，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – x
影响标志位 OV、Z、AC、C

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

SWAPA [m] Swap nibbles of Data Memory with result in ACC
指令说明 将指定数据存储器的低 4 位与高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。
功能表示 ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0
影响标志位 无

SZ [m] Skip if Data Memory is 0
指令说明 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 如果 [m]=0，跳过下一条指令执行
影响标志位 无

SZA [m] Skip if Data Memory is 0 with data movement to ACC
指令说明 将指定数据存储器内容复制到累加器，并判断指定数据存储器的内容是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 ACC ←[m]，如果 [m]=0，跳过下一条指令执行
影响标志位 无

SZ [m].i	Skip if bit i of Data Memory is 0
指令说明	判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 [m].i=0，跳过下一条指令执行
影响标志位	无
TABRD [m]	Read table (specific page or current page) to TBLH and Data Memory
指令说明	将表格指针 (TBHP 和 TBLP，若无 TBHP 则仅 TBLP) 所指的程序代码低字节移至指定数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
TABRDL [m]	Read table (last page) to TBLH and Data Memory
指令说明	将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
XOR A, [m]	Logical XOR Data Memory to ACC
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
功能表示	ACC ← ACC“XOR”[m]
影响标志位	Z
XORM A, [m]	Logical XOR ACC to Data Memory
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。
功能表示	[m] ← ACC“XOR”[m]
影响标志位	Z
XOR A, x	Logical XOR immediate data to ACC
指令说明	将累加器的数据与立即数逻辑异或，结果存放到累加器。
功能表示	ACC ← ACC“XOR”x
影响标志位	Z

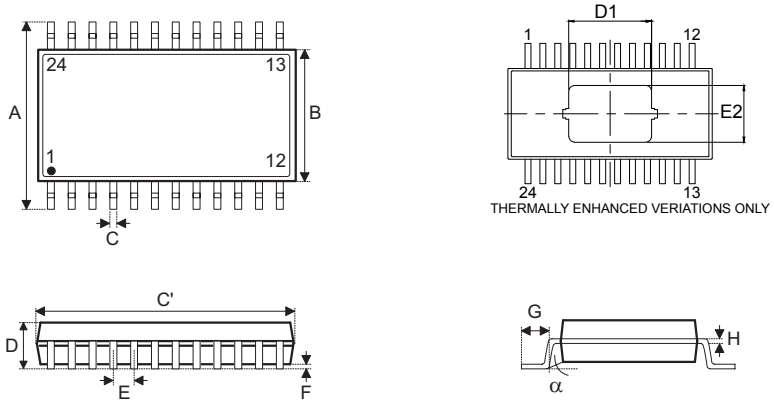
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

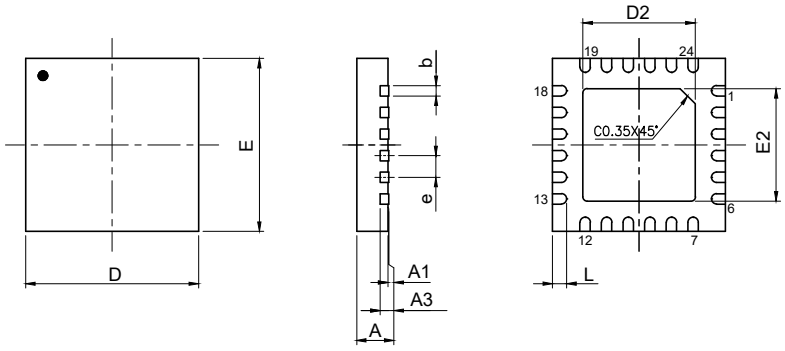
24-pin SSOP-EP (150mil) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
D1	—	0.140	—
E	—	0.025 BSC	—
E2	—	0.096	—
F	0.000	—	0.004
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.20	—	0.30
C’	—	8.66 BSC	—
D	—	—	1.75
D1	—	3.56	—
E	—	0.635 BSC	—
E2	—	2.44	—
F	0.00	—	0.10
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

SAW Type 24-pin QFN (4mm×4mm, lead: 0.325mm) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	—	0.008 BSC	—
b	0.007	0.010	0.012
D	—	0.157 BSC	—
E	—	0.157 BSC	—
e	—	0.020 BSC	—
D2	0.100	0.102	0.104
E2	0.100	0.102	0.104
L	0.011	0.013	0.015

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	—	0.203 BSC	—
b	0.18	0.25	0.30
D	—	4.00 BSC	—
E	—	4.00 BSC	—
e	—	0.50 BSC	—
D2	2.55	2.60	2.65
E2	2.55	2.60	2.65
L	0.275	0.325	0.375

Copyright© 2022 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

本文件出版时 HOLTEK 已针对所载信息为合理注意，但不保证信息准确无误。文中提到的信息仅是提供作为参考，且可能被更新取代。HOLTEK 不担保任何明示、默示或法定的，包括但不限于适合商品化、令人满意的质量、规格、特性、功能与特定用途、不侵害第三方权利等保证责任。HOLTEK 就文中提到的信息及该信息之应用，不承担任何法律责任。此外，HOLTEK 并不推荐将 HOLTEK 的产品使用在会由于故障或其他原因而可能会对人身安全造成危害的地方。HOLTEK 特此声明，不授权将产品使用于救生、维生或安全关键零部件。在救生 / 维生或安全应用中使用 HOLTEK 产品的风险完全由买方承担，如因该等使用导致 HOLTEK 遭受损害、索赔、诉讼或产生费用，买方同意出面进行辩护、赔偿并使 HOLTEK 免受损害。HOLTEK (及其授权方，如适用) 拥有本文件所提供信息 (包括但不限于内容、数据、示例、材料、图形、商标) 的知识产权，且该信息受著作权法和其他知识产权法的保护。HOLTEK 在此并未明示或暗示授予任何知识产权。HOLTEK 拥有不事先通知而修改本文件所载信息的权利。如欲取得最新的信息，请与我们联系。