BLCD 电机 Flash 单片机

BD66FM5250

版本 : V1.00　日期 : 2022-04-19

www.holtek.com

目录

特性 7
CPU 特性 7
周边特性 7
概述 8
方框图 8
引脚图 9
引脚说明 11
极限参数 16
直流电气特性 17
工作电压特性 17
工作电流特性 17
待机电流特性 17
交流电气特性 18
内部高速振荡器 – HIRC – 频率精准度 18
内部低速振荡器电气特性 – LIRC 18
工作频率电气特性曲线图 18
系统上电时间电气特性 19
输入 / 输出口电气特性 19
存储器电气特性 20
LVR/LVD 电气特性 21
A/D 转换器电气特性 21
过电流保护电气特性 22
比较器电气特性 22
上电复位特性 23
系统结构 23
时序和流水线结构 23
程序计数器 24
堆栈 24
算术逻辑单元 – ALU 25
Flash 程序存储器 26
结构 26
特殊向量 26
查表 26
查表范例 27
在线烧录 – ICP 28
片上调试 – OCDS 28
在线应用编程 – IAP 29
数据存储器 44
结构 44

数据存储器寻址 44
通用数据存储器 45
特殊功能数据存储器 45
特殊功能寄存器 47
间接寻址寄存器 – IAR0, IAR1, IAR2 47
存储器指针 – MP0, MP1L, MP1H, MP2L, MP2H 47
累加器 – ACC 48
程序计数器低字节寄存器 – PCL 48
查表寄存器 – TBLP, TBHP, TBLH 49
Option 存储器映射寄存器 – ORMC 49
状态寄存器 – STATUS 49
EEPROM 数据存储器 51
EEPROM 数据存储器结构 51
EEPROM 寄存器 51
EEPROM 读操作 53
EEPROM 页擦操作 53
EEPROM 写操作 54
写保护 55
EEPROM 中断 55
编程注意事项 55
振荡器 58
振荡器概述 58
系统时钟配置 58
内部高速 RC 振荡器 – HIRC 59
内部 32kHz 振荡器 – LIRC 59
工作模式和系统时钟 59
系统时钟 59
系统工作模式 60
控制寄存器 61
工作模式切换 62
待机电流的注意事项 65
唤醒 65
看门狗定时器 66
看门狗定时器时钟源 66
看门狗定时器控制寄存器 66
看门狗定时器操作 67
复位和初始化 68
复位功能 68
复位初始状态 70
输入 / 输出端口 76
上拉电阻 76
PA 口唤醒 77
输入 / 输出端口控制寄存器 77
引脚共用功能 77

输入 / 输出引脚结构 84
编程注意事项 84
定时器模块 – TM 85
简介 85
TM 操作 85
TM 时钟源 85
TM 中断 85
TM 外部引脚 85
编程注意事项 86
周期型 TM – PTM 87
周期型 TM 操作 88
周期型 TM 寄存器介绍 88
周期型 TM 工作模式 93
捕捉定时器模块 – CAPTM 100
捕捉定时器简介 100
捕捉定时器寄存器介绍 100
捕捉定时器操作 103
A/D 转换器 105
A/D 转换器简介 105
A/D 转换器寄存器介绍 105
A/D 转换器操作 109
A/D 转换器输入信号 110
A/D 转换率及时序图 111
A/D 转换步骤 111
编程注意事项 112
A/D 转换功能 112
A/D 转换应用范例 113
比较器 115
比较器操作 115
比较器寄存器 115
过流检测 117
过流检测功能介绍 117
过流检测寄存器 117
无刷直流电机控制电路 119
功能简介 119
PWM 计数器控制电路 120
Mask 功能 125
其它功能 130
霍尔传感器译码电路 132
电机保护功能 140
UART 接口 145
UART 外部引脚 146
UART 单线模式 146
UART 数据传输方案 147

UART 状态和控制寄存器 147
波特率发生器 153
UART 模块的设置与控制 154
UART 发送器 156
UART 接收器 157
接收错误处理 158
UART 模块中断结构 159
UART 模块暂停和唤醒 160
16 位乘除法单元 – MDU 161
MDU 寄存器 161
乘除法单元操作 162
低电压检测 – LVD 163
LVD 寄存器 163
LVD 操作 164
中断 164
中断寄存器 164
中断操作 173
霍尔传感器中断 174
外部中断 0 175
比较器 0 中断 175
PROTECTOC 中断 175
多功能中断 175
A/D 转换器中断 176
CAPTM 中断 176
PWM 模块中断 176
TM 中断 176
时基中断 177
LVD 中断 178
UART 中断 178
EEPROM 中断 178
中断唤醒功能 178
编程注意事项 178
应用电路 179
指令集 180
简介 180
指令周期 180
数据的传送 180
算术运算 180
逻辑和移位运算 180
分支和控制转换 181
位运算 181
查表运算 181
其它运算 181

指令集概要 182
惯例 182
扩展指令集 185
指令定义 187
扩展指令定义 199
封装信息 209
24-pin SSOP (150mil) 外形尺寸 210
28-pin SSOP (150mil) 外形尺寸 211
SAW Type 32-pin QFN (4mm×4mm×0.75mm) 外形尺寸 212

特性

CPU 特性

- 工作电压
  - ♦ f_{SYS}=32kHz~20MHz：4.5V~5.5V
- V_{DD}=5V，系统时钟为 20MHz 时，指令周期为 0.2μs
- 提供暂停和唤醒功能，以降低功耗
- 振荡器类型
  - ♦ 内部高速 20MHz RC – HIRC
  - ♦ 内部低速 32kHz RC – LIRC
- 内部集成振荡器，无需外接元件
- 多种工作模式：快速、低速、空闲和休眠
- 所有指令可在 1~3 个指令周期完成
- 查表指令
- 115 条指令
- 8 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器：8K×16
- RAM 数据存储器：2048×8
- True EEPROM 存储器：512×8
- 在线应用编程功能 – IAP
- 看门狗定时器功能
- 30 个双向 I/O 口
- 4 个与 I/O 口共用的外部中断输入 – H1、H2、H3 和 INT0
- 多个定时器模块用于时间测量、比较匹配输出、PWM 输出及单脉冲输出
- 1 个 16-bit CAPTM 用于电机保护
- 1 个 10-bit PWM 具有 3 组输出，支持边沿 / 中心对齐模式
- 10+1 通道 12-bit 分辨率的 A/D 转换器
- 过流检测 – 运算放大器、比较器 0 和 8-bit D/A 转换器
- 多达 4 个比较器功能
- 1 个全双工或半双工异步通信接口 – UART
- 内建乘除法单元 – MDU
- 单个时基功能，用于产生固定时间的中断信号
- 低电压复位功能
- 低电压检测功能
- 封装类型：32-pin QFN，24/28-pin SSOP

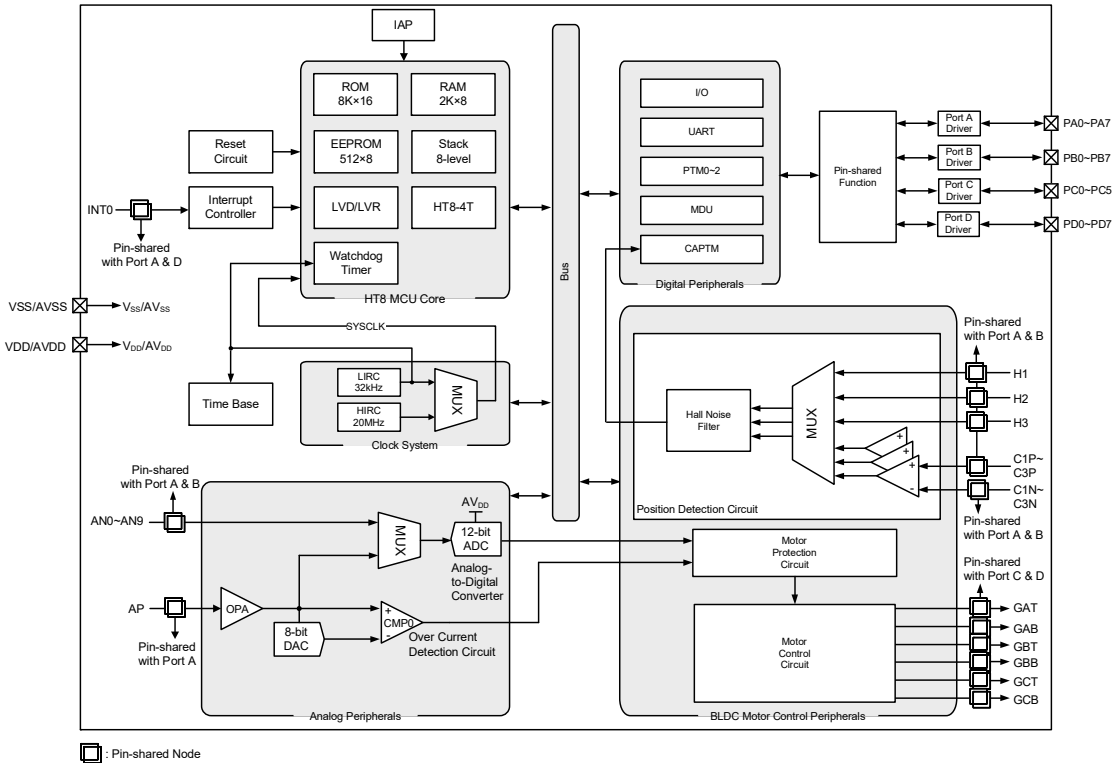
概述

BD66FM5250 是一款 8 位高性能精简指令集 Flash 单片机，它高度集成多种特性，专门为无刷直流电机应用而设计。

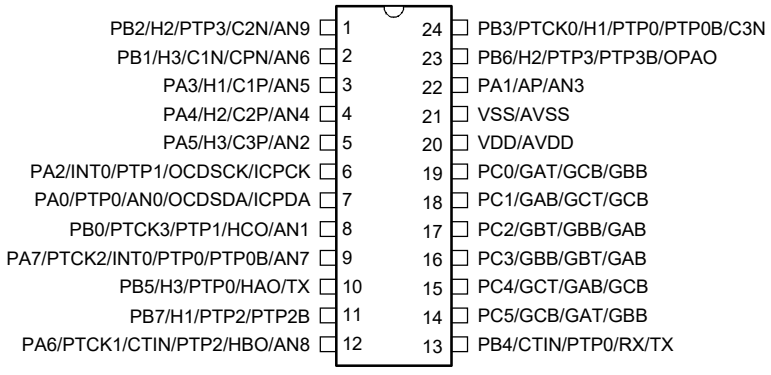
在存储器特性方面，Flash 存储器可多次编程的特性给用户提供了较大的方便。除了 Flash 程序存储器，还包括 RAM 数据存储器和用于存储序列数据、校准数据等非易失性数据的 True EEPROM 存储器。此外通过使用 IAP 功能，便于用户直接将测量的数据存储至程序存储器中或进行应用程序更新。

此单片机具有性能良好、功耗低、I/O 使用灵活的特性，外加集成有定时器模块、内部振荡器、多通道 A/D 转换器、D/A 转换器、运算放大器、脉宽调制功能、16 位捕捉定时器模块、比较器、电机保护模块、霍尔传感器位置检测功能、UART 接口功能、16 位 MDU、时基、低电压复位、低电压检测、暂停和唤醒功能。虽是一款专为无刷直流电机应用而设计的单片机，但其强大的功能使得它可以广泛应用于 A/D 产品中，例如传感器信号处理、电机驱动、工业控制、消费类产品和子系统控制等。

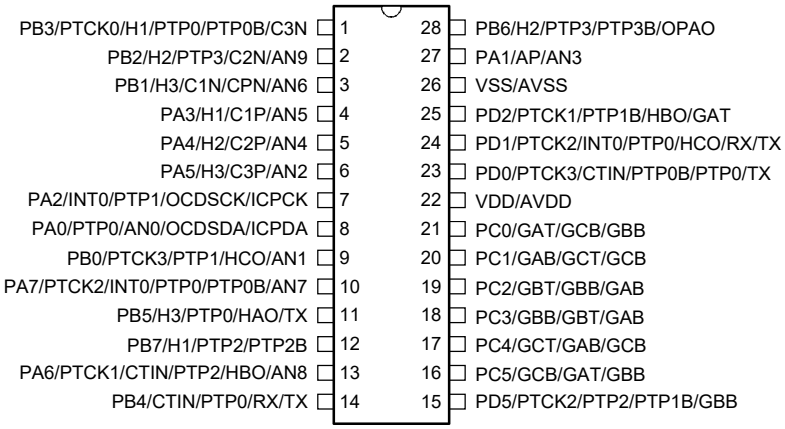
方框图



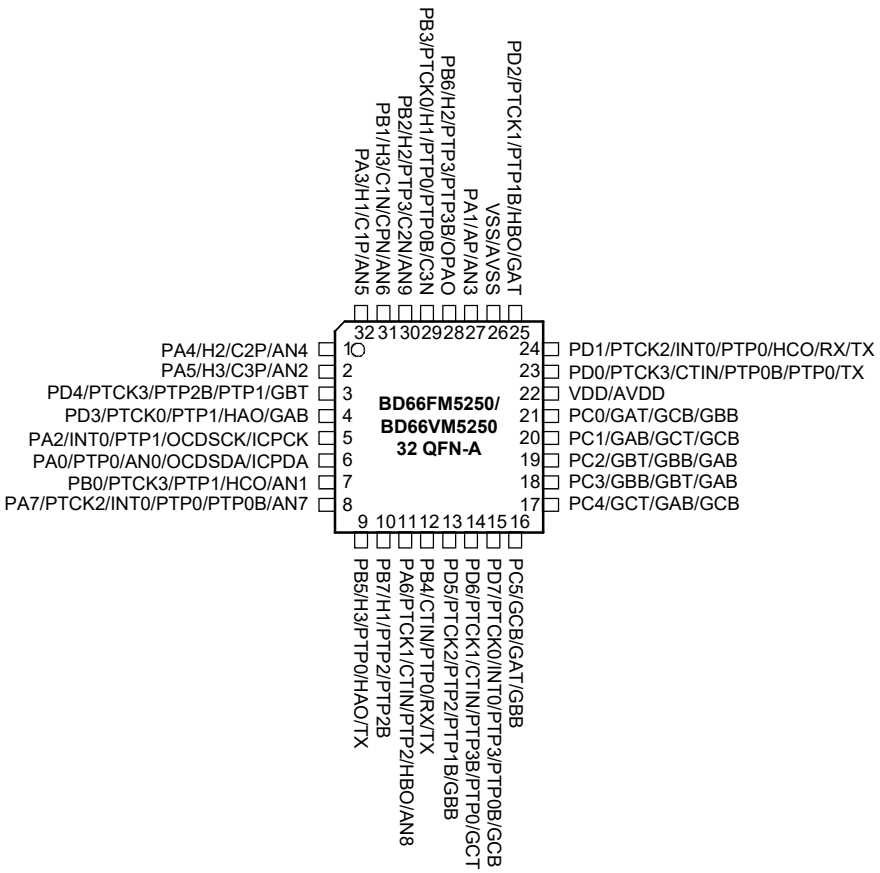
引脚图



BD66FM5250/BD66VM5250
24 SSOP-A



BD66FM5250/BD66VM5250
28 SSOP-A



注：1. 若共用脚有多种输出，所需引脚共用功能通过引脚共用寄存器中相应的软件控制位控制。
2. BD66VM5250 是 BD66FM5250 的 EV 芯片，OCDSCK 和 OCDSDA 引脚仅存在于 OCDS EV 芯片。
3. 在较小封装中可能含有未引出的引脚，需合理设置其状态以避免输入浮空造成额外耗电，详见“待机电流注意事项”和“输入 / 输出端口”章节。

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。由于该单片机存在多种封装类型，该表格反映的是较大封装的情况。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/PTP0/AN0/OCDSDA/ICPDA	PA0	PAWU PAPU 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
	PTP0	PAS0	—	CMOS	PTM0 输出
	AN0	PAS0	AN	—	A/D 转换器外部输入通道
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址引脚，仅用于 EV 芯片
	ICPDA	—	ST	CMOS	ICP 数据 / 地址引脚
PA1/AP/AN3	PA1	PAWU PAPU 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
	AP	PAS0	AN	—	OPA 正端输入
	AN3	PAS0	AN	—	A/D 转换器外部输入通道
PA2/INT0/PTP1/OCDSCK/ICPCK	PA2	PAWU PAPU 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
	INT0	PAS0 IFS1 INTEG INTC0	ST	—	外部中断输入 0
	PTP1	PAS0	—	CMOS	PTM1 输出
	OCDSCK	—	ST	CMOS	OCDS 时钟总线，仅用于 EV 芯片
	ICPCK	—	ST	CMOS	ICP 时钟总线
PA3/H1/C1P/AN5	PA3	PAWU PAPU 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
	H1	PAS0 IFS0	ST	—	霍尔传感器输入
	C1P	PAS0	AN	—	比较器 1 正端输入
	AN5	PAS0	AN	—	A/D 转换器外部输入通道
PA4/H2/C2P/AN4	PA4	PAWU PAPU 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
	H2	PAS1 IFS0	ST	—	霍尔传感器输入
	C2P	PAS1	AN	—	比较器 2 正端输入
	AN4	PAS1	AN	—	A/D 转换器外部输入通道

引脚名称	功能	OPT	I/T	O/T	说明
PA5/H3/C3P/AN2	PA5	PAWU PAPU 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
	H3	PAS1 IFS0	ST	—	霍尔传感器输入
	C3P	PAS1	AN	—	比较器 3 正端输入
	AN2	PAS1	AN	—	A/D 转换器外部输入通道
PA6/PTCK1/CTIN/ PTP2/HBO/AN8	PA6	PAWU PAPU 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
	PTCK1	PAS1 IFS3	ST	—	PTM1 时钟输入
	CTIN	PAS1 IFS0	ST	—	CAPTM 输入
	PTP2	PAS1	—	CMOS	PTM2 输出
	HBO	PAS1	—	CMOS	FHB 测试引脚
	AN8	PAS1	AN	—	A/D 转换器外部输入通道
PA7/PTCK2/INT0/ PTP0/PTP0B/AN7	PA7	PAWU PAPU 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
	PTCK2	PAS1 IFS3	ST	—	PTM2 时钟输入
	INT0	PAS1 IFS1 INTEG INTC0	ST	—	外部中断输入 0
	PTP0	PAS1	—	CMOS	PTM0 输出
	PTP0B	PAS1	—	CMOS	PTM1 反相输出
	AN7	PAS1	AN	—	A/D 转换器外部输入通道
PB0/PTCK3/PTP1/ HCO/AN1	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK3	PBS0 IFS3	ST	—	PTM3 时钟输入
	PTP1	PBS0	—	CMOS	PTM1 输出
	HCO	PBS0	—	CMOS	FHC 测试引脚
	AN1	PBS0	AN	—	A/D 转换器外部输入通道
PB1/H3/C1N/CPN/ AN6	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	H3	PBS0 IFS0	ST	—	霍尔传感器输入
	C1N	PBS0	AN	—	比较器 1 负端输入
	CPN	PBS0	AN	—	比较器 1&2&3 负端输入
	AN6	PBS0	AN	—	A/D 转换器外部输入通道

引脚名称	功能	OPT	I/T	O/T	说明
PB2/H2/PTP3/C2N/AN9	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	H2	PBS0 IFS0	ST	—	霍尔传感器输入
	PTP3	PBS0	—	CMOS	PTM3 输出
	C2N	PBS0	AN	—	比较器 2 负端输入
	AN9	PBS0	AN	—	A/D 转换器外部输入通道
PB3/PTCK0/H1/PTP0/PTP0B/C3N	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK0	PBS0 IFS3	ST	—	PTM0 时钟输入
	H1	PBS0 IFS0	ST	—	霍尔传感器输入
	PTP0	PBS0	—	CMOS	PTM0 输出
	PTP0B	PBS0	—	CMOS	PTM0 反相输出
	C3N	PBS0	AN	—	比较器 3 负端输入
PB4/CTIN/PTP0/RX/TX	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CTIN	PBS1 IFS0	ST	—	CAPTM 输入
	PTP0	PBS1	—	CMOS	PTM0 输出
	RX/TX	PBS1 IFS1	ST	CMOS	UART 串行数据输入 (全双工通信) 或 UART 串行数据输入 / 输出 (单线通信)
PB5/H3/PTP0/HAO/TX	PB5	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	H3	PBS1 IFS0	ST	—	霍尔传感器输入
	PTP0	PBS1	—	CMOS	PTM0 输出
	HAO	PBS1	—	CMOS	FHA 测试引脚
	TX	PBS1	—	CMOS	UART 串行数据输出
PB6/H2/PTP3/PTP3B/OPAO	PB6	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	H2	PBS1 IFS0	ST	—	霍尔传感器输入
	PTP3	PBS1	—	CMOS	PTM3 输出
	PTP3B	PBS1	—	CMOS	PTM3 反相输出
	OPAO	PBS1	—	AN	OPA 输出
PB7/H1/PTP2/PTP2B	PB7	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	H1	PBS1 IFS0	ST	—	霍尔传感器输入
	PTP2	PBS1	—	CMOS	PTM2 输出
	PTP2B	PBS1	—	CMOS	PTM2 反相输出

引脚名称	功能	OPT	I/T	O/T	说明
PC0/GAT/GCB/GBB	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	GAT	PCS0	—	CMOS	脉冲宽度调制互补输出
	GCB	PCS0	—	CMOS	脉冲宽度调制互补输出
	GBB	PCS0	—	CMOS	脉冲宽度调制互补输出
PC1/GAB/GCT/GCB	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	GAB	PCS0	—	CMOS	脉冲宽度调制互补输出
	GCT	PCS0	—	CMOS	脉冲宽度调制互补输出
	GCB	PCS0	—	CMOS	脉冲宽度调制互补输出
PC2/GBT/GBB/GAB	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	GBT	PCS0	—	CMOS	脉冲宽度调制互补输出
	GBB	PCS0	—	CMOS	脉冲宽度调制互补输出
	GAB	PCS0	—	CMOS	脉冲宽度调制互补输出
PC3/GBB/GBT/GAB	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	GBB	PCS0	—	CMOS	脉冲宽度调制互补输出
	GBT	PCS0	—	CMOS	脉冲宽度调制互补输出
	GAB	PCS0	—	CMOS	脉冲宽度调制互补输出
PC4/GCT/GAB/GCB	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	GCT	PCS1	—	CMOS	脉冲宽度调制互补输出
	GAB	PCS1	—	CMOS	脉冲宽度调制互补输出
	GCB	PCS1	—	CMOS	脉冲宽度调制互补输出
PC5/GCB/GAT/GBB	PC5	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	GCB	PCS1	—	CMOS	脉冲宽度调制互补输出
	GAT	PCS1	—	CMOS	脉冲宽度调制互补输出
	GBB	PCS1	—	CMOS	脉冲宽度调制互补输出
PD0/PTCK3/CTIN/PTP0B/PTP0/TX	PD0	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK3	PDS0 IFS3	ST	—	PTM3 时钟输入
	CTIN	PDS0 IFS0	ST	—	CAPTM 输入
	PTP0B	PDS0	—	CMOS	PTM0 反相输出
	PTP0	PDS0	—	CMOS	PTM0 输出
	TX	PDS0	—	CMOS	UART 串行数据输出

引脚名称	功能	OPT	I/T	O/T	说明
PD1/PTCK2/INT0/PTP0/HCO/RX/TX	PD1	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK2	PDS0 IFS3	ST	—	PTM2 时钟输入
	INT0	PDS0 IFS1 INTEG INTC0	ST	—	外部中断输入 0
	PTP0	PDS0	—	CMOS	PTM0 输出
	HCO	PDS0	—	CMOS	SC 测试引脚
	RX/TX	PDS0 IFS1	ST	CMOS	UART 串行数据输入 (全双工通信) 或 UART 串行数据输入 / 输出 (单线通信)
PD2/PTCK1/PTP1B/HBO/GAT	PD2	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK1	PDS0 IFS3	ST	—	PTM3 时钟输入
	PTP1B	PDS0	—	CMOS	PTM0 反相输出
	HBO	PDS0	—	CMOS	SA 测试引脚
	GAT	PDS0	—	CMOS	脉冲宽度调制互补输出
PD3/PTCK0/PTP1/HAO/GAB	PD3	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK0	PDS0 IFS3	ST	—	PTM0 时钟输入
	PTP1	PDS0	—	CMOS	PTM1 输出
	HAO	PDS0	—	CMOS	FHA 测试引脚
	GAB	PDS0	—	CMOS	脉冲宽度调制互补输出
PD4/PTCK3/PTP2B/PTP1/GBT	PD4	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK3	PDS1 IFS3	ST	—	PTM3 时钟输入
	PTP2B	PDS1	—	CMOS	PTM2 反相输出
	PTP1	PDS1	—	CMOS	PTM1 输出
	GBT	PDS1	—	CMOS	脉冲宽度调制互补输出
PD5/PTCK2/PTP2/PTP1B/GBB	PD5	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK2	PDS1 IFS3	ST	—	PTM2 时钟输入
	PTP2	PDS1	—	CMOS	PTM2 输出
	PTP1B	PDS1	—	CMOS	PTM1 反相输出
	GBB	PDS1	—	CMOS	脉冲宽度调制互补输出

引脚名称	功能	OPT	I/T	O/T	说明
PD6/PTCK1/CTIN/PTP3B/PTP0/GCT	PD6	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK1	PDS1 IFS3	ST	—	PTM1 时钟输入
	CTIN	PDS1 IFS0	ST	—	CAPTM 输入
	PTP3B	PDS1	—	CMOS	PTM3 反相输出
	PTP0	PDS1	—	CMOS	PTM0 输出
	GCT	PDS1	—	CMOS	脉冲宽度调制互补输出
PD7/PTCK0/INT0/PTP3/PTP0B/GCB	PD7	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTCK0	PDS1 IFS3	ST	—	PTM0 时钟输入
	INT0	PDS1 IFS1 INTEG INTC0	ST	CMOS	外部中断输入 0
	PTP3	PDS1	—	CMOS	PTM3 输出
	PTP0B	PDS1	—	CMOS	PTM0 反相输出
	GCB	PDS1	—	CMOS	脉冲宽度调制互补输出
VDD/AVDD	VDD	—	PWR	—	数字正电源
	AVDD	—	PWR	—	模拟正电源
VSS/AVSS	VSS	—	PWR	—	数字负电源，接地
	AVSS	—	PWR	—	模拟负电源，接地

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

极限参数

电源供应电压 ..... V_{SS}-0.3V~6.0V
端口输入电压 ..... V_{SS}-0.3V~V_{DD}+0.3V
存储温度 ..... -60°C~150°C
工作温度 ..... -40°C~105°C
I_{OH} 总电流 ..... -80mA
I_{OL} 总电流 80mA
总功耗 500mW

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

直流电气特性

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

工作电压特性

Ta=-40°C~105°C

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	$f_{SYS}=f_{HIRC}=20MHz$	4.5	—	5.5	V
	工作电压 – LIRC	$f_{SYS}=f_{LIRC}=32kHz$	4.5	—	5.5	

工作电流特性

Ta=-40°C~105°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	工作电流 – HIRC	5V	$f_{SYS}=f_{HIRC}=20MHz$	—	9	12	mA
	工作电流 – LIRC	5V	$f_{SYS}=f_{LIRC}=32kHz$	—	30	50	μA

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

待机电流特性

Ta=25°C，除非另有说明

符号	待机模式	测试条件		最小	典型	最大	Max. @105°C	单位
		V_{DD}	条件					
I_{STB}	休眠模式	5V	WDT off	—	0.5	2.0	10.0	μA
	休眠模式	5V	WDT on	—	1.2	2.4	2.9	
	空闲模式 0 – LIRC	5V	f_{SUB} on	—	5	10	12	
	空闲模式 1 – HIRC	4.5V	f_{SUB} on，$f_{SYS}=f_{HIRC}=20MHz$	—	1.32	1.98	2.37	mA
		5V		—	2.50	3.30	3.96	

注：当使用该表格电气特性数据时，以下几点需注意：
1. 任何数字输入都设置为非浮空的状态。
2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有待机电流数值都是在 HALT 指令执行后测得，因此 HALT 后停止执行所有指令。

交流电气特性

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

内部高速振荡器 – HIRC – 频率精准度

程序烧录时，烧录器会依据用户选择的 HIRC 频率和工作电压 (5V) 对 HIRC 进行频率精准度调整。

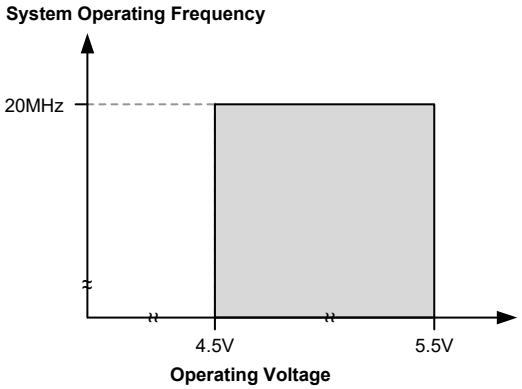
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{HIRC}	通过烧录器调整后的 20MHz HIRC 频率	5V	25°C	-1%	20	+1%	MHz
			-40°C~105°C	-2.5%	20	+2.5%	
		4.5V~5.5V	25°C	-2.5%	20	+2.5%	
			-40°C~105°C	-3%	20	+3%	

注：1. 烧录器可在 5V 这个固定电压下对 HIRC 频率进行调整，在此提供 V_{DD}=5V 时的参数值。
2. 5V 表格列下面提供的是全压条件下的参数值。

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 频率	4.5V~5.5V	25°C	-10%	32	+10%	kHz
			-40°C~105°C	-50%	32	+60%	
t_{START}	LIRC 启动时间	—	-40°C~105°C	—	—	500	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~105°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	14	16	18	ms
	系统复位延迟时间 (LVRC/WDTC 软件复位)	—	—				
	系统复位延迟时间 (WDT 溢出复位)	—	—				
t_{SRESET}	软件复位最小延迟脉宽	—	—	45	90	120	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} 和 f_{SYS} 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，则上面表格中对应的 t_{SST} 数值还需加上 LIRC 频率表格里提供的 LIRC 启动时间 t_{START}。
4. 系统速度切换时间实际上是指新使能的振荡器的启动时间。

输入 / 输出口电气特性

Ta=-40°C~105°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
V_{IH}	I/O 口高电平输入电压	5V	—	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	V
I_{OL}	I/O 口灌电流	4.5V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	mA
I_{OH}	I/O 口源电流	4.5V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	mA
R_{PH}	I/O 口上拉电阻 (注)	4.5V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
I_{LEAK}	输入漏电流	5V	$V_{IN}=V_{DD}$ 或 $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TCK}	PTCKn 输入引脚最小脉宽	—	—	0.3	—	—	μs
f_{TMCLK}	PTMn 计数时钟源最大频率	5V	—	—	—	1	f_{SYS}

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{INT}	外部中断输入引脚最小脉宽	—	—	10	—	—	μs
t_{H1}	H1 输入引脚最小脉宽	—	—	20	—	—	ns
t_{H2}	H2 输入引脚最小脉宽	—	—	20	—	—	ns
t_{H3}	H3 输入引脚最小脉宽	—	—	20	—	—	ns
t_{CTIN}	CTIN 输入引脚最小脉宽	—	—	20	—	—	ns

注：R_{PH} 内部上拉电阻值的计算方法是：引脚接地并设置为输入且使能上拉电阻功能，然后在特定电源电压下测量该引脚上电流，最后电压除以测量的电流值从而得到此上拉电阻值。

存储器电气特性

Ta=-40℃~105℃，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{RW}	读 / 写工作电压	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash 程序存储器							
t_{FER}	擦除时间	—	—	2.273	2.500	2.778	ms
t_{FWR}	写时间	—	—	1.364	1.500	1.667	ms
E_P	存储单元耐受性	—	—	100K	—	—	E/W
t_{RETD}	ROM 数据保存时间	—	Ta=25℃	—	40	—	Year
t_{ACTV}	ROM 激活时间 – 从暂停模式唤醒（注）	—	—	32	—	64	μs
数据 EEPROM 存储器							
t_{EEER}	擦除时间	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
t_{EEWR}	写时间 (字节模式)	—	EWERTS=0	—	2.2	2.7	ms
		—	EWERTS=1	—	3.0	3.6	
	写时间 (页模式)	—	EWERTS=0	—	3.2	3.9	
		—	EWERTS=1	—	3.7	4.5	
E_P	存储单元耐受性	—	—	100K	—	—	E/W
t_{RETD}	ROM 数据保存时间	—	Ta=25℃	—	40	—	Year
RAM 数据存储器							
V_{DR}	RAM 数据保存电压	—	—	1	—	—	V

注：1. 在计算从暂停模式唤醒的系统总启动时间时，还需加上 ROM 激活时间 t_{ACTV}。
2.“E/W”表示擦 / 写次数。

LVR/LVD 电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{LVR}	低电压复位电压	—	LVR 使能，电压选择 3.15V	-5%	3.15	+5%	V
V_{LVD}	低电压检测电压	—	LVD 使能，电压选择 3.3V	-5%	3.3	+5%	V
		—	LVD 使能，电压选择 3.6V	-5%	3.6	+5%	V
		—	LVD 使能，电压选择 4.0V	-5%	4.0	+5%	V
I_{LVRLVD}	工作电流	5V	LVD 使能，LVR 使能，V_{LVR}=3.15V，V_{LVD}=3.6V	—	10	15	μA
t_{LVDS}	LVDO 稳定时间	—	LVR 使能，LVD off → on	—	—	18	μs
t_{LVR}	产生 LVR 复位的低电压最短保持时间	—	—	120	240	480	μs
t_{LVD}	产生 LVD 中断的低电压最短保持时间	—	—	60	120	240	μs
I_{LVR}	LVR 使能的额外电流	—	LVD 除能	—	—	14	μA

A/D 转换器电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{ADI}	A/D 转换器输入电压	—	—	0	—	V_{REF}	V
V_{REF}	A/D 转换器参考电压	—	—	2	—	AV_{DD}	V
N_R	分辨率	—	—	—	—	12	Bit
DNL	非线性微分误差	—	V_{REF}=AV_{DD}=V_{DD}, t_{ADCK}=16/f_{SYS}	-3	—	3	LSB
INL	非线性积分误差	—	V_{REF}=AV_{DD}=V_{DD}, t_{ADCK}=16/f_{SYS}	-4	—	4	LSB
I_{ADC}	A/D 转换器使能的额外电流	5V	无负载 (t_{ADCK}=16/f_{SYS})	—	500	700	μA
t_{ON2ST}	A/D 转换器 On-to-Start 时间	—	—	4	—	—	μs
t_{ADC}	A/D 转换时间 (包括采样和保持时间)	—	—	—	16	—	t_{ADCK}

过电流保护电气特性

Ta=25°C

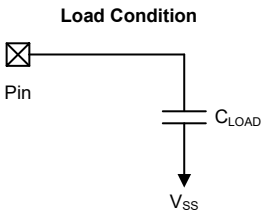
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OCP}	工作电流	5V	C0HYEN=0 OPAVS[2:0]=111	—	450	600	μA
V_{REF}	DAC 参考电压	5V	—	2.2	—	V_{DD}	V
V_{OS_CMP}	比较器输入失调电压	5V	未校准	-15	—	15	mV
V_{HYS0}	迟滞	5V	比较器 0	50	100	150	mV
V_{CM_CMP}	比较器共模电压范围	5V	—	V_{SS}	—	V_{DD}-1.0	V
V_{OS_OPA}	OPA 输入失调电压	5V	未校准 (AOF[4:0]=10000B)	-15	—	15	mV
		5V	已校准	-2	—	2	mV
V_{CM_OPA}	OPA 共模电压范围	5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{OR}	OPA 最大输出电压范围	5V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
Ga	PGA 增益精准度	5V	所有增益	-5	—	5	%
DNL	非线性微分误差	5V	DAC V_{REF}=V_{DD}	—	—	±1	LSB
INL	非线性积分误差	5V	DAC V_{REF}=V_{DD}	—	—	±1.5	LSB

比较器电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{CMP}	比较器工作电流	5V	—	—	300	450	μA
I_{CSTB}	比较器暂停电流	5V	比较器除能	—	—	0.1	μA
V_{HYS}	迟滞	5V	比较器 0	50	100	150	mV
			比较器 1、2、3	20	40	60	mV
V_{CM}	输入共模电压范围	5V	—	V_{SS}	—	V_{DD}-1	V
A_{OL}	比较器开环增益	5V	—	60	80	—	dB
t_{RP}	比较器响应时间	5V	100mV 过载 (1) (2)	—	200	400	ns

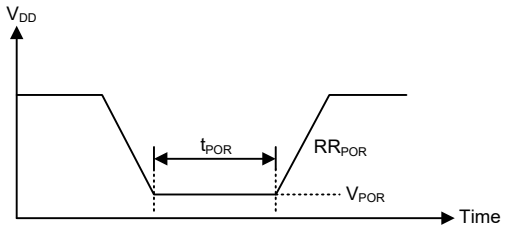
注：1. 测量方式为：当一只输入引脚的输入电压为 V_M=(V_{DD}-1.0)/2 时，另一只输入引脚的输入电压从 V_{SS} 到 (V_M+100mV) 或从 V_{DD} 到 (V_M-100mV) 转变。
2. 负载条件：C_{LOAD}=50pF



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



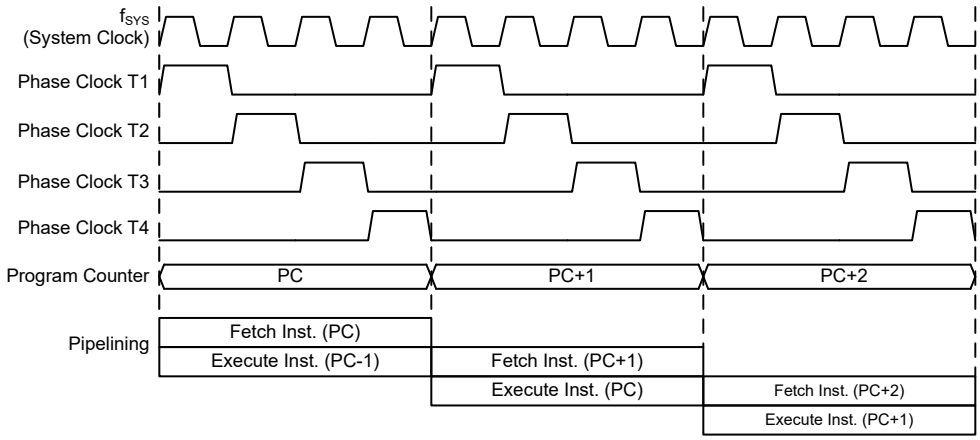
系统结构

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，该单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的获取和执行以及数据回写操作同时进行，此举使得除了跳转和调用指令需多一个指令周期外，其它大部分标准指令或扩展指令都能分别在一个或两个指令周期内完成。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 和 A/D 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大量生产的控制应用。

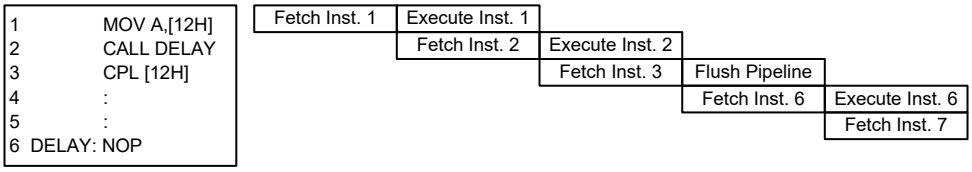
时序和流水线结构

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



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	
高字节	低字节 (PCL)
PC12~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。

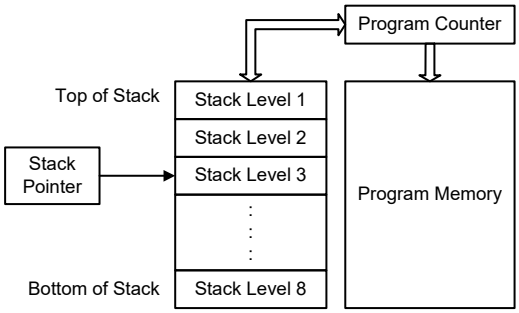
堆栈

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。该单片机有 8 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，
  LADD，LADDM，LADC，LADCM，LSUB，LSUBM，LSBC，LSBCM，
  LDAA
- 逻辑运算

  AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA，
  LAND，LANDM，LOR，LORM，LXOR，LXORM，LCPL，LCPLA
- 移位运算

  RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC，
  LRR，LRRA，LRRCA，LRRC，LRLA，LRL，LRLCA，LRLC
- 递增和递减

  INCA，INC，DECA，DEC，
  LINCA，LINC，LDECA，LDEC
- 分支判断

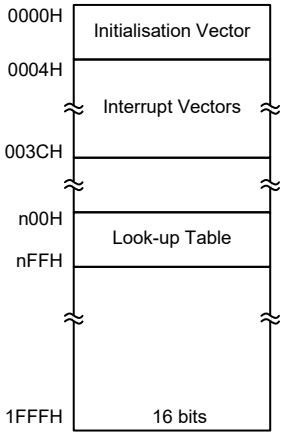
  JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI，
  LSNZ，LSZ，LSZA，LSIZ，LSIZA，LSDZ，LSDZA

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 8K×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

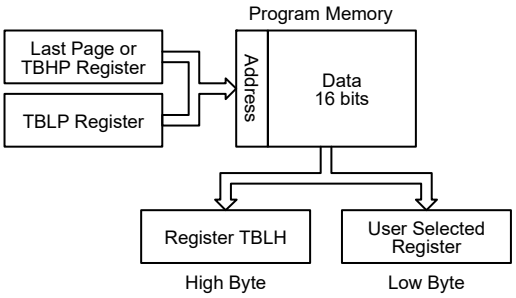
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

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，当数据存储器 [m] 位于 Sector 0，表格数据可以使用如“TABRD [m]”或“TABRDL [m]”等指令分别从程序存储器查表读取。如果存储器 [m] 位于其它 Sector，表格数据可以使用如“LTABRD [m]”或“LTABRDL [m]”等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。

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 8K 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 1F06H，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]”指令或“LTABRD [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]”指令或“LTABRD [m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

TBLH 寄存器为可读 / 写寄存器，且能重新储存，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           ; initialise low table pointer - note that this address
                    ; is referenced
mov tblp,a          ; to the last page or the page that tbhp pointed
mov a,1fh           ; initialise high table pointer
mov tbhp,a          ; it is not necessary to set tbhp if executing tabrdl
                    ; or ltabrdl
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer
                    ; data at program memory address "1F06H" 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 "1F05H" 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 1F00h           ; set initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

在线烧录 – ICP

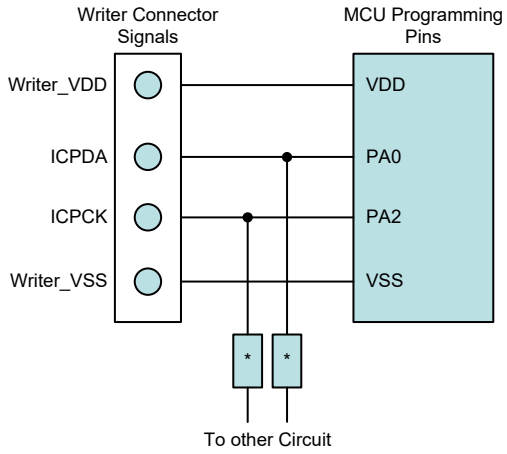
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

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	引脚描述
ICPDA	PA0	串行数据 / 地址烧录
ICPCK	PA2	时钟烧录
VDD	VDD	电源 (5V)
VSS	VSS	地

程序存储器可以通过 4 线的接口在线进行烧录。其中一条线用于数据串行下载或上传、一条线用于串行时钟、剩下两条用于提供电源。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。

烧录过程中，用户必须确保 ICPDA 和 ICPCK 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 芯片用于单片机仿真。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，单片机和 EV 芯片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 芯片对单片机的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能对 EV 芯片无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“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 (5V)
VSS	VSS	地

在线应用编程 – IAP

Flash 型程序存储器便于用户在同一芯片上对程序进行更新和修改。单片机提供的 IAP 功能使用户可以方便地对 Flash 程序存储器进行多次编程。IAP 功能可以通过内部固件进行程序的更新，而无需外接烧录器或 PC。此外，IAP 接口通过 I/O 引脚可以设置为任何类型的通信协议，例如 UART。关于内部固件，用户可以选择 Holtek 提供的版本或创建自己的内部固件。以下章节说明了如何实现 IAP 固件程序。

Flash 存储器读 / 写大小

Flash 存储器以页为单位进行擦 / 写操作，以字为单位进行读出操作。页的大小和写入缓冲器的大小都为 32 字。注意，在执行写入操作之前必须先执行擦除操作。

Flash 存储器擦 / 写功能成功使能时 CFWEN 位会被硬件置高，当该位被置高，便可写入数据到“写入缓冲器”。FWT 位用于启动写入程序，并指示写入操作的状态。当该位由应用程序置高时将开始一个写入程序，当写入操作结束后该位将由硬件清零。

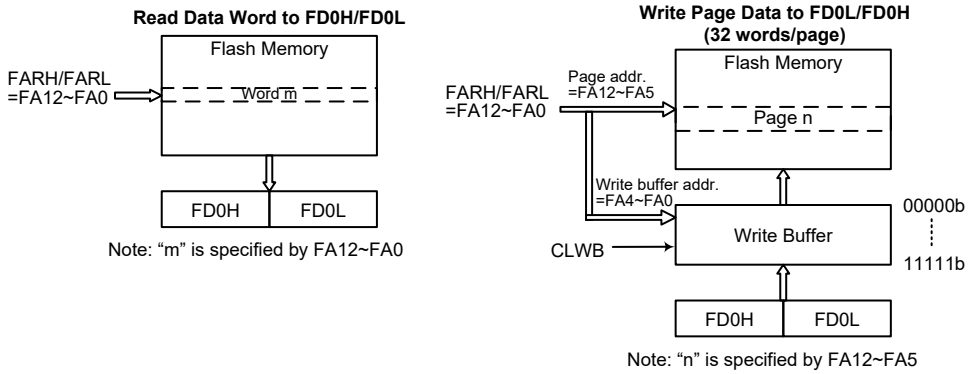
读出操作是通过一个特定的读出程序来执行的。FRDEN 位用于使能读出功能，由应用程序设置 FRD 位来启动读出程序，并指示读出操作的状态。当读出操作结束后该位将由硬件清零。

操作	格式
擦除	32 字 / 页
写入	32 字 / 次
读出	1 字 / 次
注：页大小 = 写入缓冲器大小 = 32 字。	

IAP 操作格式

页	FARH	FARL[7:5]	FARL[4:0]
0	0000 0000	000	标记地址
1	0000 0000	001	
2	0000 0000	010	
3	0000 0000	011	
4	0000 0000	100	
: :	: :	: :	
254	0001 1111	110	
255	0001 1111	111	

页序号和地址选择



Flash 存储器 IAP 读 / 写结构

写入缓冲器

执行写入操作时写入缓冲器用于临时存储写入的数据。通过执行 Flash 存储器擦 / 写使能程序成功使能 Flash 存储器擦 / 写功能后，才可将要写入的数据填入到“写入缓冲器”。通过配置 FC2 寄存器中的 CLWB 位可以清除写入缓冲器。置高 CLWB 位可以使能清除写入缓冲器程序，完成后该位会被硬件自动清零。建议第一次使用写入缓冲器或更新写入缓冲器内的数据时，应先置高 CLWB 位将写入缓冲器清零。

写入缓冲器的大小为 32 字，与页的大小一致。写入缓冲器的地址与存储器地址位 FA12~FA5 指定的 Flash 存储器页的地址相对应。写入到 FD0L 和 FD0H 寄存器的数据会被加载到写入缓冲器。当写入数据到高字节数据寄存器 FD0H，会使存储在高 / 低字节数据寄存器内的数据都写入到“写入缓冲器”，并使 Flash 存储器地址自动加一，之后新的地址会被加载到 FARH 和 FARL 地址寄存器。当 Flash 存储器地址到达当前页的最大地址，即 32 字的页为 11111b，地址将不再增加，并停在该页的最后一个地址，此时需要再设定一个新的页地址才可进行擦 / 写操作。

写入程序结束后，硬件会自动清除写入缓冲器。注意，如果数据比对步骤发现写入到 Flash 存储器的数据不正确时，需通过应用程序手动清除写入缓冲器，在写入缓冲器被清零之后再重新对其写入数据。

IAP Flash 程序存储器寄存器

IAP Flash 程序存储器有两个地址寄存器、四个 16 位数据寄存器和三个控制寄存器，且都位于 Sector 1。使用地址、数据和控制寄存器可以实现对 Flash 存储器的 16 位数据读写操作。这几个寄存器控制了内部 Flash 程序存储器所有操作，即地址寄存器 FARL 和 FARH，数据寄存器 FDnL 和 FDnH，控制寄存器 FC0、FC1 和 FC2。由于这些寄存器都位于 Sector 1 中，可通过扩展指令直接被访问，或者通过存储器指针对 MP1H/MP1L 或 MP2H/MP2L 和间接寻址寄存器 IAR1 或 IAR2 进行间接读取或写入。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	FWERTS	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	—	—	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP 寄存器列表

- **FC0 寄存器**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
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 **CFWEN**：Flash 存储器擦 / 写功能使能控制位
0：Flash 存储器擦 / 写功能除能
1：Flash 存储器擦 / 写功能已成功使能
当该位由应用程序清零时，Flash 存储器擦 / 写功能除能。注意该位不能由应用程序置高，对该位直接写“1”不会使能擦 / 写功能。此位用于指示 Flash 存储器擦 / 写功能的状态。硬件置高此位，表示 Flash 存储器擦 / 写功能成功使能，否则此位为 0，表示 Flash 存储器擦 / 写功能除能。

Bit 6~4 **FMOD2~FMOD0**：Flash 存储器模式选择位
000：写入模式
001：页擦除模式
010：保留
011：读出模式
100：保留
101：保留
110：Flash 存储器擦 / 写使能模式
111：保留
这几位用于选择 Flash 存储器的操作模式。注意在执行擦 / 写 Flash 存储器操作之前必须先成功使能“Flash 存储器擦 / 写使能模式”。

Bit 3 **FWPEN**：Flash 存储器擦 / 写使能程序触发控制位
0：擦 / 写使能程序未被触发或程序定时器溢出
1：擦 / 写使能程序被触发且程序定时器开始计时
该位用于启动 Flash 存储器擦 / 写使能程序和内部定时器。此位由应用程序置高，当内部定时器计时溢出后由硬件清零。需在 FWPEN 置高后尽快写入正确数据序列到 FD1L/FD1H、FD2L/FD2H 和 FD3L/FD3H 寄存器

Bit 2 **FWT**：Flash 存储器写入控制位
0：不启动 Flash 存储器写入程序或写入程序已完成
1：启动 Flash 存储器写入程序
此位由软件置高，当 Flash 存储器写入程序结束后由硬件清零。

Bit 1 **FRDEN**：Flash 存储器读出使能位
0：除能
1：使能
此位为 Flash 存储器读出使能位，在执行 Flash 存储器读出操作之前需将此位置高。将此位清零则禁止 Flash 存储器读出操作。

Bit 0 **FRD**：Flash 存储器读出控制位
0：不启动 Flash 存储器读出程序或读出程序已完成
1：启动 Flash 存储器读出程序
此位由软件置高，当 Flash 存储器读出程序结束后由硬件清零。

注：1. 在同一条指令中 FWT、FRDEN 和 FRD 位不可同时设置为“1”。
2. 确保 f_{SUB} 时钟在执行擦或写动作前已稳定。
3. 当读、擦或写动作成功启动后，CPU 相关操作将停止。
4. 确保读、擦或写动作成功完成后才可执行其它操作。

- **FC1 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**：整个芯片复位
当用户写入特定值“55H”到该寄存器，将产生一个复位信号将整个单片机复位。

- **FC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	FWERTS	CLWB
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1 **FWERTS**：擦除时间和写入时间选择位
0：擦除时间为 3.2ms (t_{FER}) / 写入时间为 2.2ms (t_{FWR})
1：擦除时间为 3.7ms (t_{FER}) / 写入时间为 3.0ms (t_{FWR})

Bit 0 **CLWB**：Flash 存储器写入缓冲器清除控制位
0：不启动清除写入缓冲器程序或清除程序已完成
1：启动清除写入缓冲器程序
此位由软件置高，当清除写入缓冲器程序结束后由硬件清零。

- **FARL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
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 **FA7~FA0**：Flash 程序存储器地址 bit 7 ~ bit 0

- **FARH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	FA12	FA11	FA10	FA9	FA8
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5　未定义，读为“0”

Bit 4~0　**FA12~FA8**：Flash 程序存储器地址 bit 12 ~ bit 8

- **FD0L 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**：第一个 Flash 存储器数据 bit 7 ~ bit 0

注意写入低字节数据寄存器 FD0L 的数据只能存储在 FD0L 寄存器，不能加载到低 8 位写入缓冲器。

- **FD0H 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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　**D15~D8**：第一个 Flash 存储器数据 bit 15 ~ bit 8

注意当写入 8 位数据到高字节数据寄存器 FD0H 时，存储在 FD0H 和 FD0L 寄存器内的 16 位数据将同时加载到 16 位写入缓冲器，此时 Flash 存储器地址寄存器 FARH 和 FARL 的内容将自动加一。

- **FD1L 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**：第二个 Flash 存储器数据 bit 7 ~ bit 0

- **FD1H 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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　**D15~D8**：第二个 Flash 存储器数据 bit 15 ~ bit 8

- **FD2L 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**：第三个 Flash 存储器数据 bit 7 ~ bit 0

- **FD2H 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**：第三个 Flash 存储器数据 bit 15 ~ bit 8

- **FD3L 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**：第四个 Flash 存储器数据 bit 7 ~ bit 0

- **FD3H 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**：第四个 Flash 存储器数据 bit 15 ~ bit 8

Flash 存储器擦 / 写流程

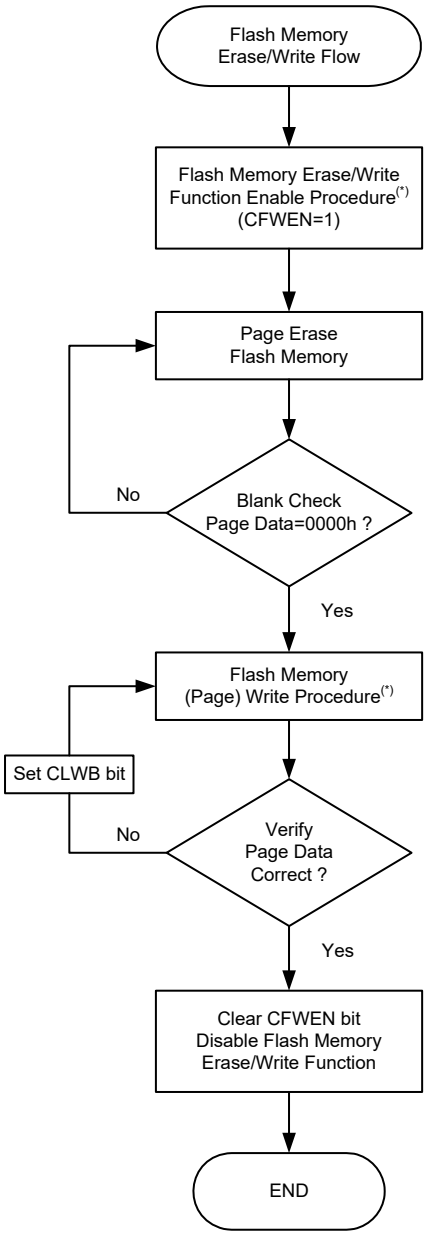
在开始更新 Flash 存储器之前，先了解 Flash 存储器擦 / 写流程操作是很重要的，用户可参考下列步骤进行程序开发，以确保 IAP 功能擦 / 写 Flash 存储器内容更新正确。

Flash 存储器擦 / 写流程说明

1. 先启动“Flash 存储器擦 / 写使能程序”。当 Flash 存储器擦 / 写功能成功使能后，FC0 寄存器中的 CFWEN 位会由硬件自动置高，此时才可执行擦 / 写 Flash 存储器操作。详细内容请参考“Flash 存储器擦 / 写使能程序”。
2. 配置 Flash 存储器地址指定要擦除的页，标记地址，然后擦除此页。

   对于页擦除操作，首先设置 FARL 和 FARH 寄存器来指定要擦除页的起始地址，然后写入任意数据到 FD0H 寄存器以标记地址。每写入一个任意数据到 FD0H 寄存器，当前地址将自动加一。当地址自动递增到页的最大地址，即 11111b，地址将不再增加，并停在该页的最后一个地址。注意写 FD0H 是为了标记地址，这一操作必须执行以确定要擦除哪些地址。
3. 查空确认是否擦除成功，可采用 TABRD 指令进行读取并比对是否为“0000h”，如果擦除不成功返回步骤 2 再擦除一次。

4. 写入数据至该页，详细内容请参考“Flash 存储器写入程序”。
5. 采用 TABRD 指令进行读取并比对写入数据是否正确，如果写入不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 4，再写入相同数据。
6. 完成当前页擦 / 写步骤后，如果无需擦 / 写其它页，可清除 CFWEN 位来解除“Flash 存储器擦 / 写使能模式”。



Flash 存储器擦 / 写流程图

注：“Flash 存储器擦 / 写使能程序”及“Flash 存储器写入程序”详见后续章节介绍。

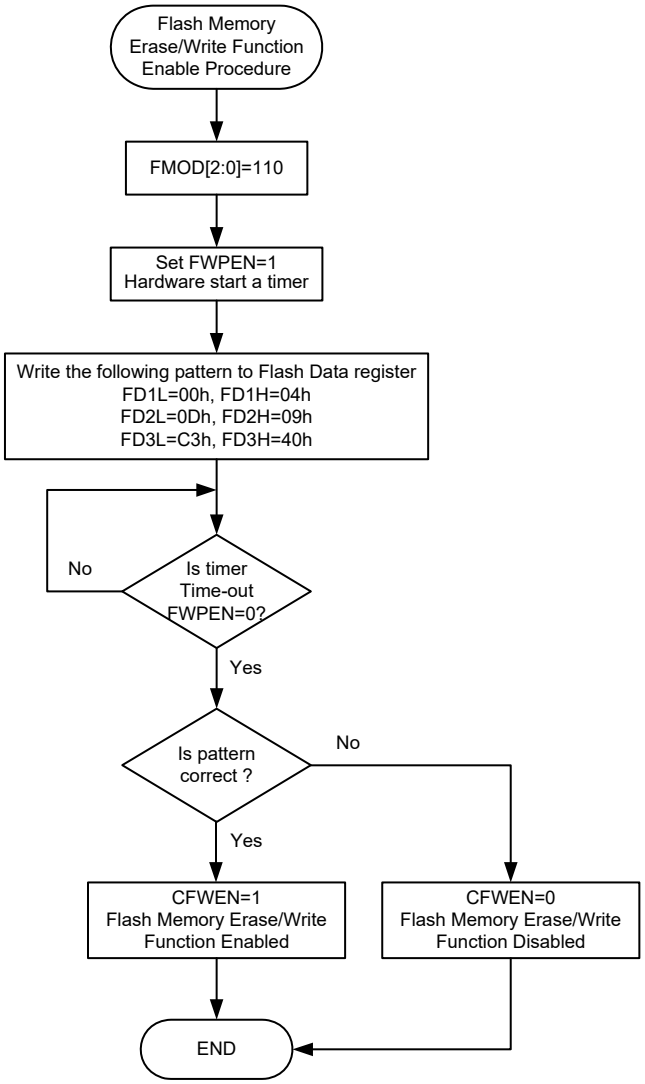
Flash 存储器擦 / 写使能程序

Flash 存储器擦 / 写使能模式是专门为保护 Flash 存储器内容不被轻易地修改而设计的。用户必须先使能 Flash 存储器擦 / 写功能，才能通过 IAP 控制寄存器来更改 Flash 存储器数据。

Flash 存储器擦 / 写使能程序说明

1. 写入数值“110”至 FC0 寄存器中的 FMOD[2:0] 位，选择 Flash 存储器擦 / 写使能模式。
2. 设置 FC0 寄存器中的 FWPEN 位为“1”，启动 Flash 存储器擦 / 写使能程序，此时内部硬件线路会启动内部定时器。
3. 使用者必须在 FWPEN 位置高后尽快填入正确数据序列至 FD1L~FD3L 和 FD1H~FD3H 寄存器中，数据序列对应为FD1L=00h、FD1H=04h、FD2L=0Dh、FD2H=09h、FD3L=C3h、FD3H=40h。
4. 一旦定时器计时结束，无论写入的数据序列是否正确，FWPEN 位将由硬件自动清零。
5. 如果写入的数据序列不正确，表示 Flash 存储器擦 / 写功能没有成功使能，需重复以上步骤。如果写入的数据序列正确，表示 Flash 存储器擦 / 写功能成功使能。
6. 一旦 Flash 存储器擦 / 写功能成功使能，即可通过 IAP 控制寄存器进行页擦 / 写操作来更新 Flash 存储器内容。

将 FC0 寄存器中的 CFWEN 位清零，可除能 Flash 存储器擦 / 写功能，不必再执行以上步骤。



Flash 存储器擦 / 写使能程序

Flash 存储器写入程序

当 Flash 擦 / 写功能成功使能后，CFWEN 位会被硬件置高，此时要写入 Flash 存储器的数据才能加载到“写入缓冲器”。在开始写入程序之前，应先正确配置 IAP 控制寄存器，将所选的 Flash 存储器页的数据擦除。

写入缓冲器的大小为每页 32 字，其地址与 FA12~FA5 指定的 Flash 存储器页的地址为相对应关系。注意“写入缓冲器”的地址与对应存储器的地址必须在相同页。

Flash 存储器连续地址写入程序说明

对于写入操作每次写入的数据为 32 字。多笔连续地址的数据写入时，写入缓冲器的地址将自动加“1”。用户只需将第一笔数据的地址填入 FARL、FARH，并将第一笔数据依序填入 FD0L、FD0H (先写 FD0L 再写 FD0H，才会将 FD0L、FD0H 数据一起填入“写入缓冲器”)，写入缓冲器的地址将自动加“1”，要填入第二笔数据时，可不用再指定地址 FARL、FARH。当连续地址到达当前页的最后一个地址时，写入缓冲器的地址将不会再自动加“1”，地址将保持为最后一个地址。

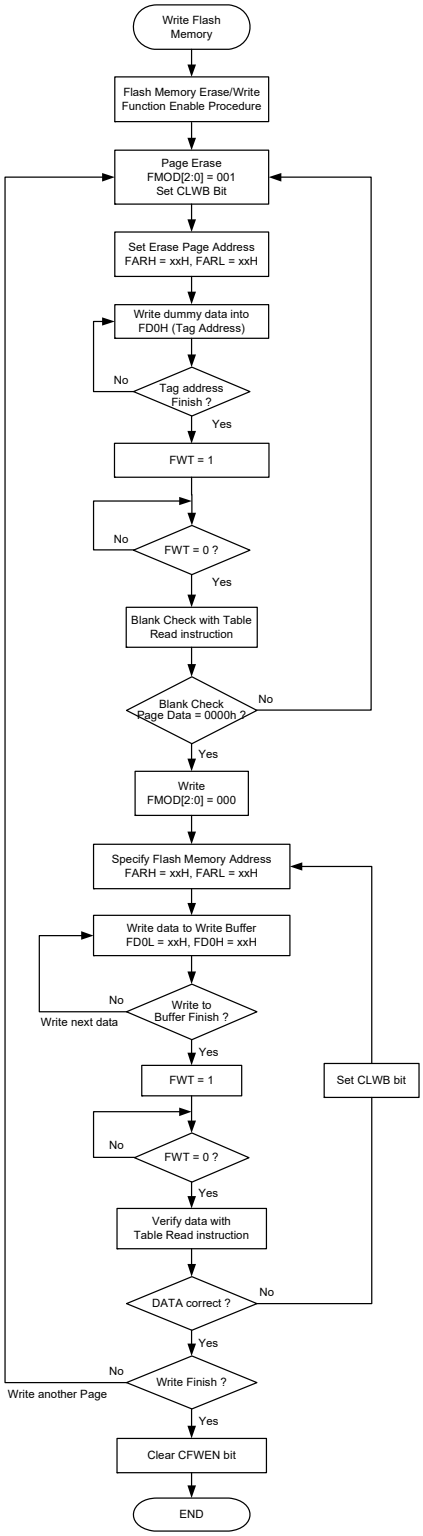
1. 启动“Flash 存储器擦 / 写使能程序”，确认 CFWEN 的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦 / 写使能程序”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式，并设置 CLWB 位为“1”清除“写入缓冲器”。设定 FWT 位为“1”，擦除由 FARH 和 FARL 指定且已标记地址的目标页，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。

   如果擦除操作不成功则返回步骤 2。

   如果擦除操作成功则接着执行步骤 4。
4. 设定 FMOD[2:0] 为“000”，选择写入模式。
5. 先将目标起始地址写入 FARL、FARH 寄存器中，将要往连续地址所在页写入的数据依序写入 FD0L、FD0H 寄存器。最多可写入 32 字。
6. 设定 FWT 位为“1”，将“写入缓冲器”的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
7. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。

   如果写入操作不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 5。

   如果写入操作成功则接着执行步骤 8。
8. 将 CFWEN 位清零以除能 Flash 存储器擦 / 写功能。



Flash 存储器连续地址写入程序

注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。
2. FWT 位由高变低所需时间可以通过 FC2 寄存器中的 FWERTS 位选择。

Flash 存储器非连续地址写入程序说明

连续地址写入操作与非连续地址写入操作的主要差别在于要写入的数据是否位于连续地址。如果要写入的数据不是位于连续的地址，当一笔数据成功写入到 Flash 存储器后需重新配置另一个目标地址。

以两笔非连续的数据写入操作为例，说明如下：

1. 启动“Flash 存储器擦 / 写使能程序”，确认 CFWEN 位的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦 / 写使能程序”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式，并设置 CLWB 位为“1”清除“写入缓冲器”。设定 FWT 位为“1”，擦除由 FARH 和 FARL 指定且已标记地址的目标页，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。

   如果擦除操作不成功则返回步骤 2。

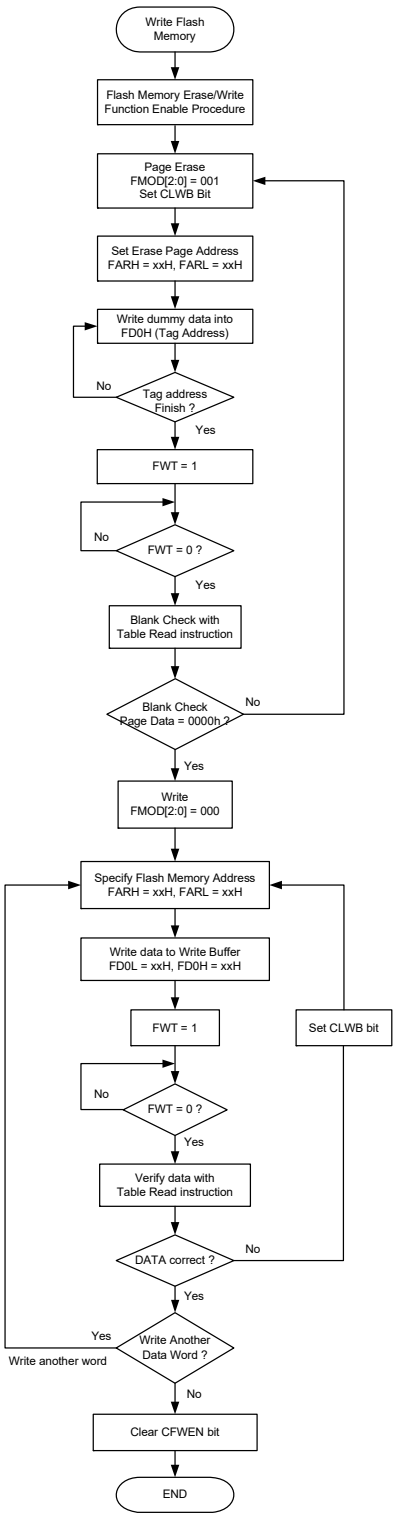
   如果擦除操作成功则接着执行步骤 4。
4. 设定 FMOD[2:0] 为“000”，选择写入模式。
5. 先将目标地址 ADDR1 写入 FARL、FARH 寄存器中，将要写入的数据 DATA1 先写入 FD0L 寄存器再写入 FD0H 寄存器。
6. 设定 FWT 位为“1”，将“写入缓冲器”的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
7. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。

   如果写入操作不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 5。

   如果写入操作成功则接着执行步骤 8。
8. 再将目标地址 ADDR2 写入 FARL、FARH 寄存器中，将要写入的数据 DATA2 先写入 FD0L 寄存器再写入 FD0H 寄存器。
9. 设定 FWT 位为“1”，将“写入缓冲器”的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
10. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。

    如果写入操作不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 8。

    如果写入操作成功则接着执行步骤 11。
11. 将 CFWEN 位清零以除能 Flash 存储器擦 / 写功能。



Flash 存储器非连续地址写入程序

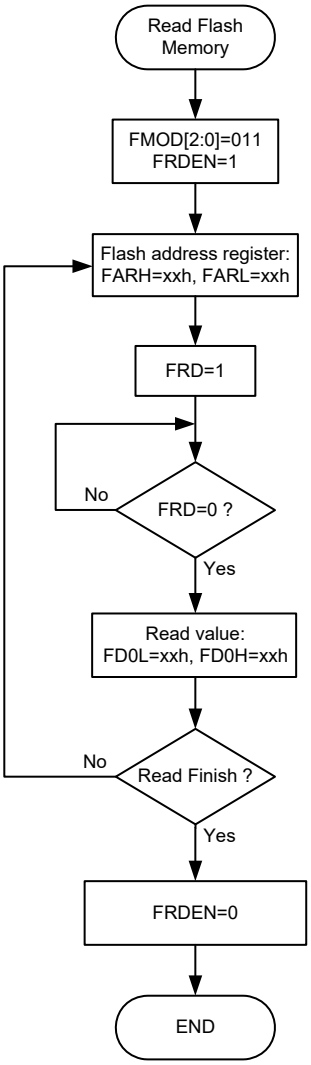
注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。
2. 在擦除或写入操作中，FWT 位由高变低所需时间可以通过 FC2 寄存器中的 FWERTS 位选择。

Flash 存储器写入操作注意事项

1. 要开始对 Flash 存储器进行 IAP 擦 / 写操作之前，必须先完成“Flash 存储器擦 / 写使能程序”。
2. Flash 存储器擦除操作以页作为擦除单位。
3. “写入缓冲器”以页为单位对 Flash 存储器写入数据，且写入时不可跨页填写。
4. 数据写入 Flash 存储器后，必须以查表指令“TABRD”读出方式比对所写数据是否正确，若比对写入数据不正确时，通过置高 CLWB 位将“写入缓冲器”清除后重新写入数据，且不清除 Flash 存储器直接再写入，然后再比对，直到写入正确。
5. 执行 IAP 数据写入与数据比对时，系统频率需与最高应用频率相同。

Flash 存储器读出程序

要启动 Flash 存储器读出程序，需将 FMOD[2:0] 位设为“011”选择 Flash 存储器读出模式，将 FRDEN 位设为“1”使能读出功能。将要读出的地址填入 FARH、FARL 地址寄存器中，并将 FRD 位设为“1”，然后便可开始 Flash 存储器读出操作。当 FRD 被硬件清为“0”时，则可在 FD0H、FD0L 取得 Flash 存储器中该地址数据。进行 Flash 存储器读出操作前，不需经过 Flash 存储器擦 / 写使能程序。



Flash 存储器读出程序

注：1. 当读动作成功启动后，所有 CPU 相关操作将暂停。
2. FRD 位由高变低所需时间为 3 个指令周期 (典型值)。

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。

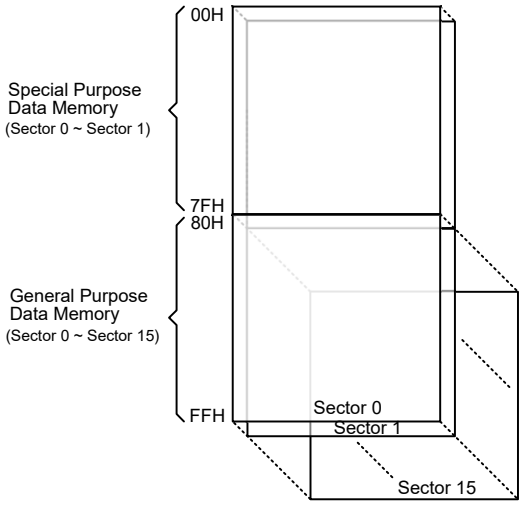
数据存储器分为两种类型，第一种是特殊功能数据存储器。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二种数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。当使用间接寻址方式时，切换不同的数据存储器 Sector 可通过设置正确的间接寻址指针值实现。

结构

数据存储器被分为两个 Sector，都是 8 位存储器。大多特殊功能寄存器可在 Sector 0 被访问，除了部分寄存器只能在 Sector 1 中被访问到。特殊功能数据存储器地址范围为 00H~7FH，而通用数据存储器地址范围为 80H~FFH。

特殊功能数据存储器	通用数据存储器	
位于 Sector	容量	**Sector：地址**
0，1	2048×8	0：80H~FFH 1：80H~FFH ： 15：80H~FFH

数据存储器概要



数据存储器结构

数据存储器寻址

此单片机支持扩展指令架构，它并没有可用于数据存储器的存储区指针。对于数据存储器，使用间接寻址访问方式时所需的 Sector 是通过 MP1H 或 MP2H 寄存器指定，而所选 Sector 的某一数据存储器地址是通过 MP1L 或 MP2L 寄存器指定。

直接寻址可用于所有 Sector，通过扩展指令可以寻址所有可用的数据存储器空间。当所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector 时，扩展指令可代替间接寻址方式用来访问数据存储器。标准指令和扩展指令的主要区别在于扩展指令中的数据存储器地址“m”有 12 个有效位，高字节表示 Sector，低字节表示指定的地址。

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”。

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	HIRCC	
11H	LVDC	
12H	LVRC	
13H	SCC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	WDTC	
19H	TBC	
1AH	INTEG	
1BH	INTC0	
1CH	INTC1	
1DH	INTC2	
1EH	INTC3	
1FH	PWMCS	
20H	PB	
21H	PBC	
22H	PBPU	
23H	PC	
24H	PCC	
25H	PCPU	
26H	SADC0	
27H	SADC1	
28H		
29H	CAPTC0	
2AH	CAPTC1	
2BH	CAPTMDL	
2CH	CAPTMDH	PAS0
2DH	CAPTMAL	PAS1
2EH	CAPTMAH	PBS0
2FH	CAPTMCL	PBS1
30H	CAPTMCH	PCS0
31H	ADCR2	PCS1
32H	ADDL	PDS0
33H	CMPSEL	PDS1
34H	CMPC	PTM2C0
35H	OPOMS	PTM2C1
36H	OPCM	PTM2DL
37H	OPACAL	PTM2DH
38H	SADOH	PTM2AL
39H	SADOL	PTM2AH
3AH	ADLVDH	PTM2RPL
3BH	ADLVDL	PTM2RPH
3CH	ADHVDH	
3DH	ADHVDL	EEAL
3EH	HINTEG	EEAH
3FH	PSCR	EED

	Sector 0	Sector 1
40H	PWMC	EEC
41H	OCPS	ORMC
42H	HDCR	
43H	HDCD	
44H	MPTC1	
45H	MPTC2	
46H	PTM1C0	
47H	PTM1C1	USR
48H	PTM1DL	UCR1
49H	PTM1DH	UCR2
4AH	PTM1AL	UCR3
4BH	PTM1AH	BRDH
4CH	PTM1RPL	BRDL
4DH	PTM1RPH	UFCR
4EH	PTM0C0	TXR_RXR
4FH	PTM0C1	RxCNT
50H		
51H	PTM0DL	FC0
52H	PTM0DH	FC1
53H	PTM0AL	FC2
54H	PTM0AH	FARL
55H		FARH
56H		FD0L
57H	PTM0RPL	FD0H
58H	PTM0RPH	FD1L
59H	MDUWR0	FD1H
5AH	MDUWR1	FD2L
5BH	MDUWR2	FD2H
5CH	MDUWR3	FD3L
5DH	MDUWR4	FD3H
5EH	MDUWR5	IFS0
5FH	MDUWCTRL	IFS1
60H	DUTR0L	
61H	DUTR0H	IFS3
62H	DUTR1L	HDCT0
63H	DUTR1H	HDCT1
64H	DUTR2L	HDCT2
65H	DUTR2H	HDCT3
66H	PRDRL	HDCT4
67H	PRDRH	HDCT5
68H	PWMRL	HDCT6
69H	PWMRH	HDCT7
6AH	PWMME	HDCT8
6BH	PWMMD	HDCT9
6CH	MCF	HDCT10
6DH	MCD	HDCT11
6EH	DTS	
6FH	PLC	
70H	MFI0	
71H	MFI1	PTM3C0
72H	MFI2	PTM3C1
73H	MFI3	PTM3DL
74H	MFI4	PTM3DH
75H	MFI5	PTM3AL
76H	MFI6	PTM3AH
77H	PD	PTM3RPL
78H	PDC	PTM3RPH
79H	PDPU	
7AH		
7BH		
7CH	HCHK_NUM	
7DH	HNF_MSEL	
7EH		
7FH		

: Unused, read as 00H

: Reserved, cannot be changed

特殊功能数据存储器结构

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, IAR2

间接寻址寄存器 IAR0、IAR1 和 IAR2 的地址虽位于数据存储区，但不同于普通寄存器，它们没有实际的物理地址。与定义实际存储器地址的直接存储器寻址不同，间接寻址是使用间接寻址寄存器和存储器指针来执行存储器数据操作。在间接寻址寄存器 IAR0、IAR1 和 IAR2 上的任何动作，将对存储器指针 MP0、MP1L/MP1H 或 MP2L/MP2H 所指定的存储器地址产生对应的读 / 写操作。它们总是成对出现，IAR0 和 MP0 可以访问 Sector 0，而 IAR1 和 MP1L/MP1H、IAR2 和 MP2L/MP2H 可以访问任何 Sector。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, MP1L, MP1H, MP2L, MP2H

该单片机提供五个存储器指针，即 MP0、MP1L、MP1H、MP2L 和 MP2H。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 Sector 0，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 可根据 MP1H 或 MP2H 寄存器访问所有的 Sector。使用扩展指令可对所有的数据 Sector 进行直接寻址。

以下例子说明如何清除一个具有 4 个 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 1

```
data .section ´data´
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 ´code´
org 00h
start:
    mov a, 04h             ; setup size of block
    mov block, a
    mov a, offset adres1   ; Accumulator loaded with first RAM address
    mov mp0, a             ; setup memory pointer with first RAM address
loop:
    clr IAR0               ; clear the data at address defined by MP0
    inc mp0                ; increment memory pointer
    sdz block              ; check if last memory location has been cleared
    jmp loop
continue:
```

间接寻址程序举例 2

```
data .section ´data´
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
```

```
code .section at 0 ´code´
org 00h
start:
    mov a, 04h            ; setup size of block
    mov block, a
    mov a, 01h            ; setup the memory sector
    mov mp1h, a
    mov a, offset adres1  ; Accumulator loaded with first RAM address
    mov mp1l, a           ; setup memory pointer with first RAM address
loop:
    clr IAR1              ; clear the data at address defined by MP1L
    inc mp1l              ; increment memory pointer MP1L
    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。

使用扩展指令直接寻址程序举例

```
data .section ´data´
temp db ?
code .section at 0 ´code´
org 00h
start:
    lmov a, [m]           ; move [m] data to acc
    lsub a, [m+1]         ; compare [m] and [m+1] data
    snz  c                ; [m]>[m+1]?
    jmp  continue         ; no
    lmov a, [m]           ; yes, exchange [m] and [m+1] data
    mov  temp, a
    lmov a, [m+1]
    lmov [m], a
    mov  a, temp
    lmov [m+1], a
continue:
```

注：“m”是位于任何数据存储器 Sector 的某一地址。例如，m=1F0H 表示 Sector 1 中的地址 0F0H。

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。

查表寄存器 – 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。

Option 存储器映射寄存器 – ORMC

ORMC 寄存器用于使能 Option 存储器映射功能。Option 存储器的容量为 64 个字。当连续写入特定数据序列 55H 和 AAH 到该寄存器，Option 存储器映射功能将使能，通过使用查表指令即可读到 Option 存储器的内容，Option 存储器的 00H~3FH 地址会一一对应到程序存储器最后一页的 C0H~FFH 地址。

要成功使能 Option 存储器映射功能，该特定的数据序列 55H 和 AAH 必须在两个指令周期内连续写入。建议在写入该特定数据序列前应当先将总中断位 EMI 清零，在数据序列成功写入后，根据用户的需求在适当的时间再将其置高。当数据序列成功写入时会启动内部定时器，$4\times t_{LIRC}$ 时间之后会自动结束映射。因此，用户需及时读出数据，否则需要重新启动 Option 存储器映射功能。每次 ORMC 寄存器被连续写入后，定时器都会重新计数。

当使用查表指令来读取 Option 存储器内容时，“TABRD [m]”和“TABRDL [m]”指令皆可使用。然而，若使用“TABRD [m]”指令来读取，必须配置 TBHP 寄存器将表格指针设定在最后一页。更多查表的描述请参考相关章节。

- **ORMC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ORMC7	ORMC6	ORMC5	ORMC4	ORMC3	ORMC2	ORMC1	ORMC0
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 **ORMC7~ORMC0**：Option 存储器映射特定数据序列
当将特定数据序列 55H 和 AAH 连续写入该寄存器，会使能 Option 存储器映射功能。需注意，单片机从空闲 / 休眠模式唤醒后，该寄存器的内容将被清除。

状态寄存器 – STATUS

这 8 位的状态寄存器由 SC 标志位、CZ 标志位、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、C、SC 和 CZ 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。
- CZ：不同指令不同标志位的操作结果。详细资料请参考寄存器定义部分。
- SC：当 OV 与当前指令操作结果 MSB 执行“XOR”所得结果。

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	SC	CZ	TO	PDF	OV	Z	AC	C
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	x	x	0	0	x	x	x	x

“x”：未知

Bit 7 **SC**：当 OV 与当前指令操作结果 MSB 执行“XOR”所得结果

Bit 6 **CZ**：不同指令不同标志位的操作结果
对于 SUB/SUBM/LSUB/LSUBM 指令，CZ 等于 Z 标志位。
对于 SBC/SBCM/LSBC/LSBCM 指令，CZ 等于上一个 CZ 标志位与当前零标志位执行“AND”所得结果。
对于其它指令，CZ 标志位无影响。

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未发生从高四位借位

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未发生借位
C 标志位也受循环移位指令的影响。

EEPROM 数据存储器

此单片机的一个特性是内建 EEPROM 数据存储器，由于其非易失的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。这种存储区扩展了存储空间，对设计者来说增加了许多新的应用机会。EEPROM 可以用来存储产品编号、校准值、用户特定数据、系统配置参数或其它产品信息等。EEPROM 的数据读取和写入过程也会变的更简单。

EEPROM 数据存储器结构

EEPROM 数据存储器容量为 512×8 位。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。使用 Sector 1 中的一对地址寄存器、一个数据寄存器以及一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

有四个寄存器控制内部 EEPROM 数据存储器总的操作，地址寄存器 EEAL 和 EEAH、数据寄存器 EED 及控制寄存器 EEC。所有寄存器都位于 Sector 1 中，可通过 MP1L/MP1H 和 IAR1 或 MP2L/MP2H 和 IAR2 进行间接读取或写入。由于 EEC 控制寄存器位于 Sector 1 中的“40H”，在 EEC 寄存器上的任何操作被执行前，MP1L 或 MP2L 必须先设为“40H”，MP1H 或 MP2H 被设为“01H”。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	—	—	—	EEAH0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

EEPROM 寄存器列表

- **EEAL 寄存器**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
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 **EEAL7~EEAL0**：数据 EEPROM 地址低字节 Bit 7 ~ Bit 0

- **EEAH 寄存器**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	—	—	—	—	—	—	—	EEAH0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 未定义，读为“0”

Bit 0 **EEAH0**：数据 EEPROM 地址高字节 Bit 0

- **EED 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**：数据 EEPROM 数据 Bit 7 ~ Bit 0

- **EEC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD
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 **EWERTS**：数据 EEPROM 擦除时间和写入时间选择位
0：擦除时间为 3.2ms (t_{EEER}) / 写入时间为 2.2ms (t_{EEWR})
1：擦除时间为 3.7ms (t_{EEER}) / 写入时间为 3.0ms (t_{EEWR})

Bit 6 **EREN**：数据 EEPROM 擦使能位
0：除能
1：使能
此位用来使能数据 EEPROM 擦功能，向数据 EEPROM 擦操作之前需将此位置高。擦周期结束后，硬件自动将此位清零。将此位清零时，则禁止向数据 EEPROM 擦操作。

Bit 5 **ER**：数据 EEPROM 擦控制位
0：擦周期结束
1：开始擦周期
此位为数据 EEPROM 擦控制位，由应用程序将此位置高将激活擦周期。擦周期结束后，硬件自动将此位清零。当 EREN 未先置高时，此位置高无效。

Bit 4 **MODE**：数据 EEPROM 操作模式选择位
0：字节操作模式
1：页操作模式
此位为数据 EEPROM 操作模式选择位。当此位为高，则选择页写、擦或读操作模式。当此位为 0，则选择字节写或读操作模式。EEPROM 页缓存器大小为 16 字节。

Bit 3 **WREN**：数据 EEPROM 写使能位
0：除能
1：使能
此位为数据 EEPROM 写使能位，向数据 EEPROM 写操作之前需将此位置高。将此位清零时，则禁止向数据 EEPROM 写操作。无论通过 MODE 位选择了何种写模式，在写操作结束后硬件都会自动将 WREN 位清零。

Bit 2 **WR**：数据 EEPROM 写控制位
0：写周期结束
1：开始写周期
此位为数据 EEPROM 写控制位，由应用程序将此位置高将激活写周期。写周期结束后，硬件自动将此位清零。当 WREN 未先置高时，此位置高无效。

Bit 1 **RDEN**：数据 EEPROM 读使能位
0：除能
1：使能
此位为数据 EEPROM 读使能位，向数据 EEPROM 读操作之前需将此位置高。将此位清零时，则禁止向数据 EEPROM 读操作。

Bit 0 **RD**：数据 EEPROM 读控制位
0：读周期结束
1：开始读周期
此位为数据 EEPROM 读控制位，由应用程序将此位置高将激活读周期。读周期结束后，硬件自动将此位清零。当 RDEN 未首先置高时，此位置高无效。

注：1. 在同一条指令中 EREN、ER、WREN、WR、RDEN 和 RD 不能同时置为“1”。
2. 确保 f_{SUB} 时钟在执行擦 / 写动作前已稳定。
3. 确保擦 / 写动作完成后才可改写 EEPROM 相关寄存器或启动 IAP 功能。

EEPROM 读操作

此单片机有两种模式可实现从 EEPROM 中读取数据，即字节读模式和页读模式，可通过 EEC 寄存器中的 EEPROM 操作模式选择位 MODE 选择。

字节读模式

当模式选择位 MODE 为 0 时，可执行 EEPROM 字节读操作。为了实现字节读操作，EEPROM 中要读取数据的地址需先放入 EEAH 和 EEAL 寄存器中，EEC 寄存器中的读使能位 RDEN 也要置高以使能读功能，然后再置高 RD 位以开始 EEPROM 字节读操作。注意，若 RD 位已置高而 RDEN 位还未被置高则不能开始读操作。若读周期结束，RD 位将自动清零，EEPROM 数据可以从 EED 寄存器中读取。读到的数据在其它读或写操作执行前将一直保留在 EED 寄存器中。应用程序可轮询 RD 位以确定数据可以有效地被读取。

页读模式

当模式选择位 MODE 为 1 时，可执行 EEPROM 页读操作。页读操作中页大小可达 16 个字节。为了实现页读操作，EEPROM 中要读取页的起始地址需先放入 EEAH 和 EEAL 寄存器中，EEC 寄存器中的读使能位 RDEN 也要置高以使能读功能，然后再置高 RD 位以开始 EEPROM 页读操作。注意，若 RD 位已置高而 RDEN 位还未被置高则不能开始读操作。当前字节读周期结束时，RD 位将自动清零，此时 EEPROM 数据可以从 EED 寄存器中读取，而且当前地址将由硬件自动加一。只要再置高 RD 位无需重新配置 EEPROM 地址和 RDEN 控制位，就可以连续读取下一个 EEPROM 地址的数据。应用程序可轮询 RD 位以确定数据可以有效地被读取。

EEPROM 地址高 5 位用来指定要读取页的位置，而低 4 位用来指向实际的地址。在页操作模式低 4 位地址将自动加一，而高 5 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到当前页的最大地址，即 0FH，EEPROM 地址低 4 位的值会停止在 0FH，EEPROM 地址将不会再增加。

EEPROM 页擦操作

当模式选择位 MODE 为 1 时，可执行 EEPROM 页擦操作。EEPROM 一页可擦除 16 个字节。上电复位后内部页缓存器将由硬件清零。当 EEPROM 擦使能控制位 EREN 由 1 变为 0 时，内部页缓存器也会被清零。注意当 EREN 位由 0 变为 1 时，内部页缓存器不会清零。EEPROM 地址高 5 位用来指定要擦除页的位置，而低 4 位用来指向实际的地址。在页擦操作模式每写入一字节任意数据到 EED 寄存器，低 4 位地址将自动加一，而高 5 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到当前页的最大地址，即 0FH，EEPROM 地址低 4 位的值会停止在 0FH，EEPROM 地址将不会再增加。

为了实现页擦操作，EEPROM 中要擦除页的起始地址需先放入 EEAH 和 EEAL 寄存器中，要写入的任意数据也需存入 EED 寄存器中。一页的最大数据长度为 16 字节。注意写 EED 是为了标记地址，这一操作必须执行以确定要擦除哪些

地址。当一整页的任意数据都写入 EED 寄存器后，EEC 寄存器中的 EREN 位先置高以使能擦功能，然后 EEC 寄存器中的 ER 位需立即置高以开始擦操作。这两条指令必须在两个指令周期内连续执行才可成功启动一个擦除操作。进行擦除操作之前应先将总中断使能位 EMI 清零，在一个有效的擦启动步骤完成之后再将其使能。

由于控制 EEPROM 擦除周期是一个内部时钟，与单片机的系统时钟异步，所以擦除 EEPROM 数据的时间将有所延迟。可通过轮询 EEC 寄存器中的 ER 位或判断 EEPROM 中断以侦测擦周期是否完成。若擦除周期完成，ER 位将自动清零，通知用户数据已擦除。因此，应用程序将轮询 ER 位以确定擦除周期是否结束。擦操作结束后，EREN 位将会被硬件置低。执行完一个页擦操作后，EEPROM 被擦除页的内容将全为零。

EEPROM 写操作

此单片机有两种模式可实现写数据到 EEPROM，即字节写模式和页写模式，可通过 EEC 寄存器中的 EEPROM 操作模式选择位 MODE 选择。

字节写模式

当模式选择位 MODE 为 0 时，可执行 EEPROM 字节写操作。为了实现字节写操作，EEPROM 中要写入数据的地址需先放入 EEAH 和 EEAL 寄存器中，写入的数据也需存入 EED 寄存器中。EEC 寄存器中的写使能位 WREN 先置高以使能写功能，然后 EEC 寄存器中的 WR 位需立即置高以开始写操作。这两条指令必须在两个指令周期内连续执行才可成功启动一个写操作。进行写操作之前应先将总中断使能位 EMI 清零，在一个有效的写启动步骤完成之后再将其使能。若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。

由于控制 EEPROM 写周期是一个内部时钟，与单片机的系统时钟异步，所以数据写入 EEPROM 的时间将有所延迟。可通过轮询 EEC 寄存器中的 WR 位或判断 EEPROM 写中断以侦测写周期是否完成。若写周期完成，WR 位将自动清零，通知用户数据已写入 EEPROM。因此，应用程序将轮询 WR 位以确定写周期是否结束。写操作结束后，WREN 位将会被硬件置低。注意，字节写操作被成功启动前会自动执行字节擦除操作。

页写模式

在执行页写操作之前，务必确保已成功执行了相关的页擦除操作。当模式选择位 MODE 为 1 时，可执行 EEPROM 页写操作。EEPROM 一页可写入 16 个字节。上电复位后内部页缓存器将由硬件清零。当 EEPROM 写使能控制位 WREN 由 1 变为 0 时，内部页缓存器也会被清零。注意当 WREN 位由 0 变为 1 时，内部页缓存器不会清零。除了最多可以写入 16 字节 EEPROM 数据以外，页写操作启动的方法与字节写操作相同。EEPROM 地址高 5 位用来指定要写入页的位置，而低 4 位用来指向实际的地址。在页写操作模式每写入一字节数据到 EED 寄存器，低 4 位地址将自动加一，而高 5 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到当前页的最大地址，即 0FH，EEPROM 地址低 4 位的值会停止在 0FH，EEPROM 地址将不会再增加。此时再对 EED 寄存器写入数据也将无效。

为了实现页写操作，EEPROM 中要写入页的起始地址需先放入 EEAH 和 EEAL 寄存器中，要写入的数据也需存入 EED 寄存器中。一页的最大数据长度为 16 字节。注意当写入一字节数据到 EED 寄存器，EED 中的数据会加载到内部页缓存器中，然后当前地址值会自动加一。当一页数据被全部写入页缓存器，EEC 寄存器中的写使能位 WREN 先置高以使能写功能，然后 EEC 寄存器中的 WR 位需立即置高以开始写操作。这两条指令必须在两个指令周期内连续执行

才可成功启动一个写操作。进行写操作之前应先将总中断使能位 EMI 清零，在一个有效的写启动步骤完成之后再将其使能。若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。

由于控制 EEPROM 写周期是一个内部时钟，与单片机的系统时钟异步，所以数据写入 EEPROM 的时间将有所延迟。可通过轮询 EEC 寄存器中的 WR 位或判断 EEPROM 写中断以侦测写周期是否完成。若写周期完成，WR 位将自动清零，通知用户数据已写入 EEPROM。因此，应用程序将轮询 WR 位以确定写周期是否结束。写操作结束后，WREN 位将会被硬件置低。

写保护

防止误写入的写保护有以下几种。单片机上电后控制寄存器中的写使能位将被清除以杜绝任何写入操作。上电后存储器指针高字节寄存器 MP1H 或 MP2H 将重置为“0”，这意味着数据存储区 Sector 0 被选中。由于 EEPROM 控制寄存器位于 Sector 1 中，这增加了对写操作的保护措施。在正常程序操作中确保控制寄存器中的写使能位被清除将能防止不正确的写操作。

EEPROM 中断

EEPROM 擦 / 写周期结束后将产生 EEPROM 擦 / 写中断，需先通过设置相关中断寄存器的 DEE 位使能 EEPROM 中断。当 EEPROM 擦 / 写周期结束，DEF 请求标志位将被置位。若总中断和 EEPROM 中断使能且堆栈未满的情况下将跳转到相应的 EEPROM 中断向量中执行。当中断被响应，EEPROM 中断标志位将自动复位。详情请参考中断章节。

编程注意事项

必须注意的是数据不会无意写入 EEPROM。在没有写动作时写使能位被正常清零可以增强保护功能。存储器指针高字节寄存器 MP1H 或 MP2H 也可以正常清零以阻止进入 EEPROM 控制寄存器存在的 Sector 1。尽管没有必要，写一个简单的读回程序以检查新写入的数据是否正确还是应该考虑的。

EREN 位置位后，EEC 寄存器中的 ER 位需立即置位，以确保擦周期正确地执行。WREN 位置位后，EEC 寄存器中的 WR 位需立即置位，以确保写周期正确地执行。写或擦周期开始前总中断位 EMI 应先清零，在一个有效的写或擦启动步骤完成之后再将此位重新使能。注意，单片机不应在 EEPROM 读、擦或写操作完全完成之前进入空闲或休眠模式，否则 EEPROM 读、擦或写操作将失败。

程序举例

从 EEPROM 中读取一个字节数据 – 轮询法

```
MOV A, 040H                ; setup memory pointer low byte MP1L
MOV MP1L, A                ; MP1L points to EEC register
MOV A, 01H                 ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4                 ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H      ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L      ; user defined low byte address
MOV EEAL, A
SET IAR1.1                 ; set RDEN bit, enable read operations
SET IAR1.0                 ; start Read Cycle - set RD bit
```

```
BACK:
SZ  IAR1.0                ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read function
CLR MP1H
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

从 EEPROM 中读取一页数据 – 轮询法

```
MOV A, 040H               ; setup memory pointer low byte MP1L
MOV MP1L, A               ; MP1L points to EEC register
MOV A, 01H                ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H     ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L     ; user defined low byte address
MOV EEAL, A
SET IAR1.1                ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0                ; check for read cycle end
JMP BACK
MOV A, EED                ; move read data to register
MOV READ_DATA, A
RET
:
PAGE_READ_FINISH:
CLR IAR1                  ; disable EEPROM read function
CLR MP1H
```

擦除 EEPROM 的一页数据 – 轮询法

```
MOV A, 040H               ; setup memory pointer low byte MP1L
MOV MP1L, A               ; MP1L points to EEC register
MOV A, 01H                ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H     ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L     ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
```

```
MOV A, EEPROM_DATA          ; user defined data, erase mode don't care data
                            ; value
MOV EED, A
RET
:
Erase_START:
CLR EMI
SET IAR1.6                  ; set EREN bit, enable erase operations
SET IAR1.5                  ; start Erase Cycle - set ER bit - executed
                            ; immediately after setting EREN bit
SET EMI
BACK:
SZ  IAR1.5                  ; check for erase cycle end
JMP BACK
CLR MP1H
```

写入一个字节数据到 EEPROM – 轮询法

```
MOV A, 040H                 ; setup memory pointer low byte MP1L
MOV MP1L, A                 ; MP1L points to EEC register
MOV A, 01H                  ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4                  ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H       ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L       ; user defined low byte address
MOV EEAL, A
MOV A, EEPROM_DATA          ; user defined data
MOV EED, A
CLR EMI
SET IAR1.3                  ; set WREN bit, enable write operations
SET IAR1.2                  ; start Write Cycle - set WR bit - executed
                            ; immediately after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                  ; check for write cycle end
JMP BACK
CLR MP1H
```

写入一页数据到 EEPROM – 轮询法

```
MOV A, 040H                 ; setup memory pointer low byte MP1L
MOV MP1L, A                 ; MP1L points to EEC register
MOV A, 01H                  ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                  ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H       ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L       ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA          ; user defined data
```

```
MOV EED, A
RET
:
WRITE_START:
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed
                          ; immediately after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                ; check for write cycle end
JMP BACK
CLR MP1H
```

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。振荡器操作是通过相关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中断的时钟源。集成的内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换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。

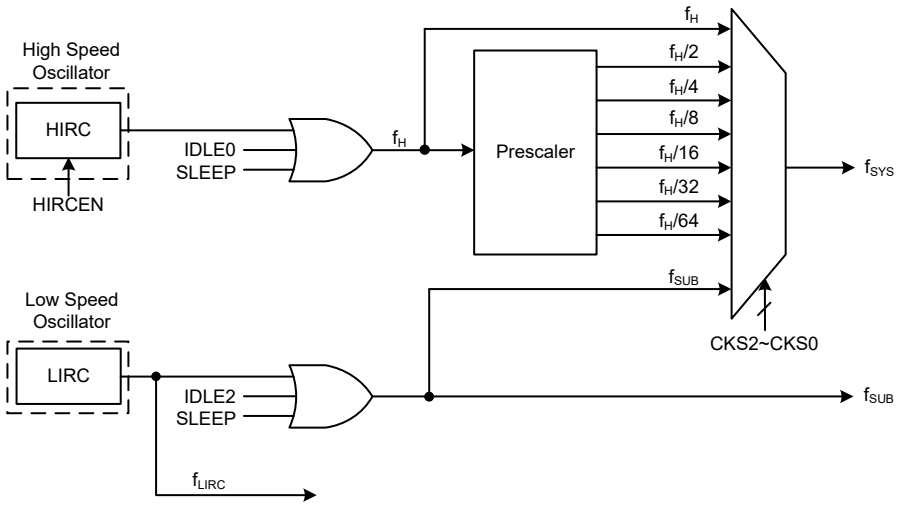
类型	名称	频率
内部高速 RC	HIRC	20MHz
内部低速 RC	LIRC	32kHz

振荡器类型

系统时钟配置

该单片机有两个振荡器，包括一个高速振荡器和一个低速振荡器。高速振荡器为内部 20MHz 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。

低速或高速系统时钟频率由 SCC 寄存器的 CKS2~CKS0 位决定的。请注意，两个振荡器必须做出选择，即一个高速振荡器和一个低速振荡器。



系统时钟配置

内部高速 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，无需其它外部器件。内部 RC 振荡器频率固定为 20MHz。芯片在制造时进行调整且内置频率补偿电路，使得振荡频率因 V_{DD}、温度以及芯片制程工艺不同的影响较大程度地降低。

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器是一个完全集成的低频 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。

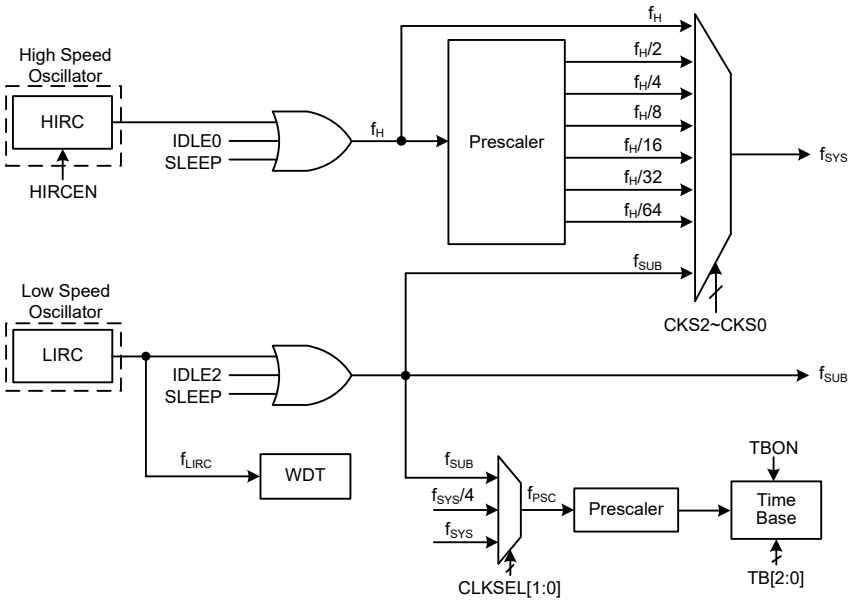
工作模式和系统时钟

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

此单片机为 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。低频系统时钟源来自 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。



单片机时钟选项

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。该低速时钟源可来自 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，f_{SUB} 停止为外围功能提供时钟。若看门狗定时器功能由 WDTC 寄存器设定为使能，f_{LIRC} 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确保一些外围功能继续工作。

空闲模式 2

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

控制寄存器

寄存器 SCC 和 HIRCC 用于控制系统时钟和相应的振荡器配置。

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

系统工作模式控制寄存器列表

- **SCC 寄存器**

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

Bit 7~5 **CKS2~CKS0**：系统时钟选择位

000：f_H
001：$f_H/2$
010：$f_H/4$
011：$f_H/8$
100：$f_H/16$
101：$f_H/32$
110：$f_H/64$
111：f_{SUB}

这三位用于选择系统时钟源。除了 f_H 或 f_{SUB} 提供的系统时钟源外，也可使用高频振荡器的分频作为系统时钟。

Bit 4~2　未定义，读为“0”

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

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

注：使用 CKS2~CKS0 位进行时钟切换设置之后，在相关时钟成功切换至目标时钟源之前需要一定的延时。因此，若接下来执行的操作需要目标时钟源立即响应，则在此之前必须规划适当的延迟时间。

时钟切换延迟时间 = $4\times t_{SYS} + [0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$，其中 $t_{Curr.}$ 指代当前的时钟周期，$t_{Tar.}$ 指代目标时钟周期，t_{SYS} 指代当前系统时钟周期。

- **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”

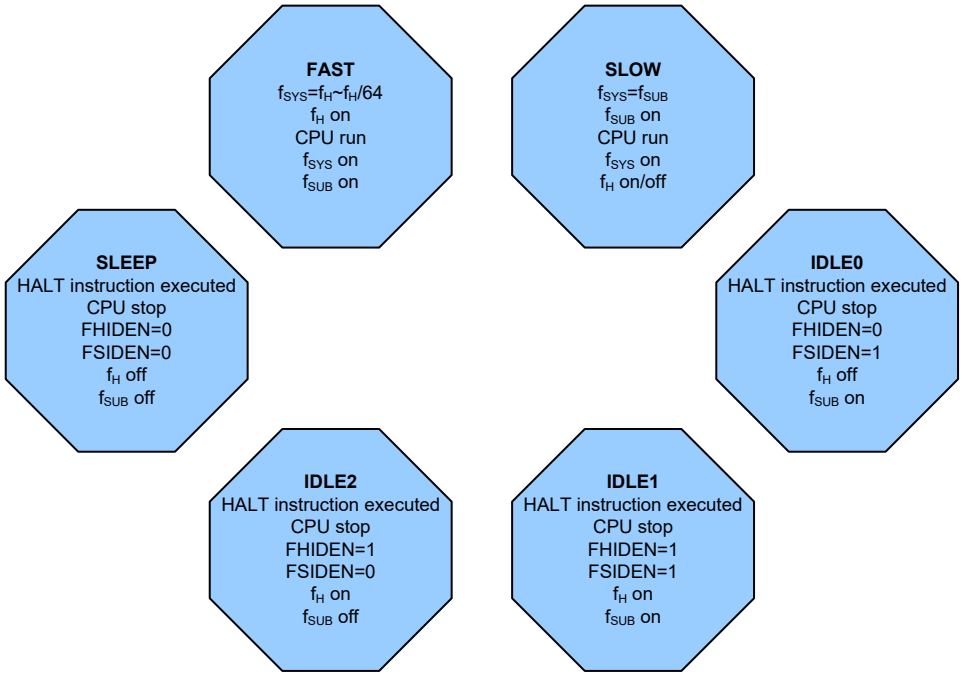
Bit 1　**HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 未稳定
1：HIRC 稳定
此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

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。

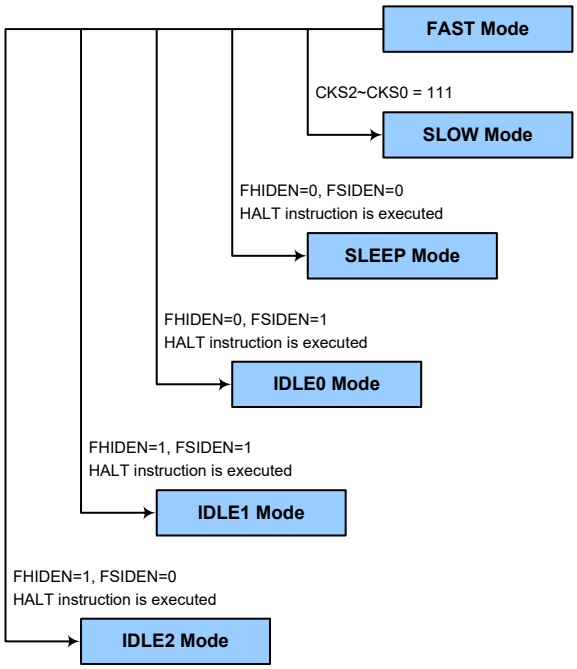
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



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。

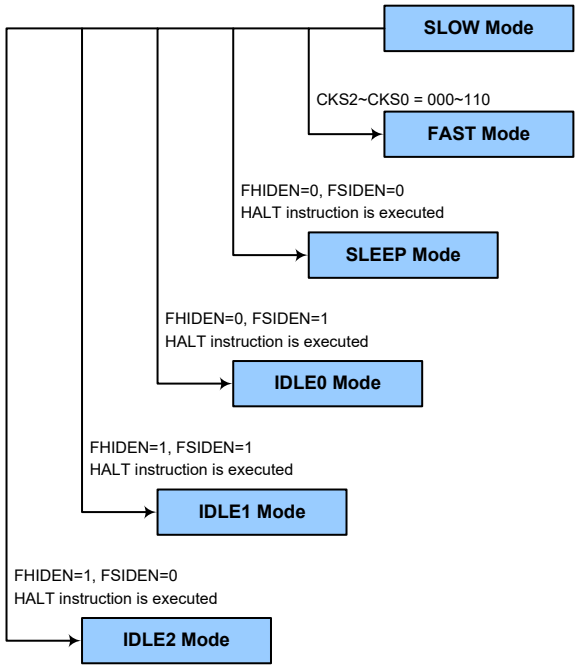
低速模式的系统时钟源自 LIRC 振荡器，因此要求此振荡器在所有模式切换动作发生前稳定下来。



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”。在这种模式下，除了 WDT 以外的所有时钟和功能都将关闭。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 将被清零。
- 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 将被清零。
- 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 将被清零。
- 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 将被清零。
- 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。

待机电流的注意事项

由于单片机进入休眠或空闲模式的主要原因是将单片机的电流降低到尽可能低，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。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

另外还需注意单片机设为输出的 I/O 引脚上的负载。应将它们设置在有最小拉电流的状态或将它们和其它的 CMOS 输入一样接到没有拉电流的外部电路上。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

执行 HALT 指令后单片机将进入空闲或休眠模式，PDF 将被置位。系统上电或执行清除看门狗的指令，PDF 将被清零。若系统由看门狗计数器溢出唤醒，将会启动看门狗复位并置位 TO 标志，这种复位只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}，由内部振荡器 LIRC 提供。内部振荡器 LIRC 的频率大约为 32kHz。需要注意的是，具体的内部时钟周期随 V_{DD}、温度和制程的不同而变化。看门狗定时器的时钟源可分频为 2^8~2^{18} 以提供更大的溢出周期，分频比由 WDTC 寄存器中的 WS2~WS0 位来决定。

看门狗定时器控制寄存器

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

- **WDTC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	WE4	WE3	WE2	WE1	WE0	WS2	WS1	WS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	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
如果由于不利的环境因素使这些位变为其它值，单片机将复位。复位动作发生在一段延迟时间 t_{SRESET} 后，且 RSTFC 寄存器的 WRF 位将置为“1”。

Bit 2~0 **WS2~WS0**：WDT 溢出周期选择位
000：$2^8/f_{LIRC}$
001：$2^{10}/f_{LIRC}$
010：$2^{12}/f_{LIRC}$
011：$2^{14}/f_{LIRC}$
100：$2^{15}/f_{LIRC}$

101：$2^{16}/f_{LIRC}$
110：$2^{17}/f_{LIRC}$
111：$2^{18}/f_{LIRC}$
这三位控制 WDT 时钟源的分频比，从而实现对 WDT 溢出周期的控制。

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~3　未定义，读为“0”

Bit 2　**LVRF**：LVR 复位标志位
详见“低电压复位”章节

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

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

看门狗定时器操作

当 WDT 溢出时，它产生一个单片机复位的动作。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，清除指令都不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。看门狗定时器控制寄存器 WDTC 中的 WE4~WE0 位可提供看门狗定时器使能 / 除能控制以及单片机复位操作。当 WE4~WE0 设置为“10101B”时除能 WDT 功能，而当设置为“01010B”时使能 WDT 功能。如果 WE4~WE0 设置为除“01010B”和“10101B”以外的值时，单片机将在 t_{SRESET} 延迟时间后复位。上电后这些位初始化为“01010B”。

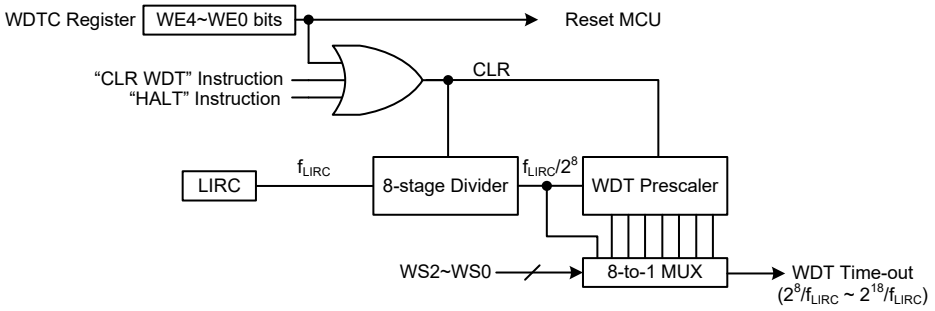
WE4~WE0 位	WDT 功能
10101B	除能
01010B	使能
其它值	单片机复位

看门狗定时器功能控制

程序正常运行时，WDT 溢出将导致单片机复位，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 和 PDF 位应置位，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 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^{18} 时，溢出周期最大。例如，时钟源为 32kHz LIRC 振荡器，分频比为 2^{18} 时最大溢出周期约 8s，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。

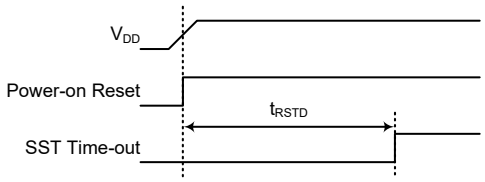
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。此外还有一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机包含下面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。



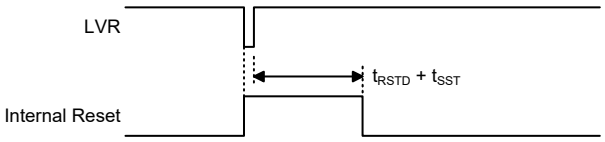
上电复位时序图

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产生单片机复位。

LVR 功能在快速 / 低速模式时始终使能，并会设定一个低电压复位阈值，V_{LVR}。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。有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过 LVR/LVD 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 LVS 位段设置固

定为 3.15V。若受到干扰 LVS7~LVS0 变为其它值时，LVR 将在 t_{SRESET} 时间后复位单片机。此时 RSTFC 寄存器中的 LRF 位被置位。上电复位后寄存器的值为 01100110B。注意当单片机进入空闲或休眠模式，LVR 功能将自动关闭。



低电压复位时序图

- **LVRC 寄存器**

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

Bit 7~0 **LVS7~LVS0**：LVR 电压选择
01100110：3.15V
01010101：3.15V
00110011：3.15V
10011001：3.15V
10101010：3.15V
11110000：3.15V
其它值：MCU 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。此低电压保持时间需要超过 t_{LVR} 才会响应复位。此时复位后 RSTFC 寄存器中的 LVRF 标志位会被置位，其它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 t_{SRESET} 时间才响应复位。经此种复位后 RSTFC 寄存器中的 LRF 标志位会被置位，其它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。

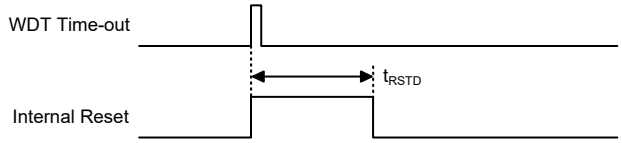
Bit 0 **WRF**：WDT 控制寄存器软件复位标志位
详见“看门狗定时器控制寄存器”章节

在线应用编程复位

当写值“55H”至 FC1 寄存器时，将产生一个复位信号将整个单片机复位。详见 IAP 章节。

正常运行时看门狗溢出复位

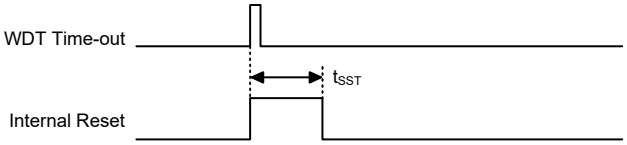
正常运行时看门狗溢出复位，看门狗溢出标志位 TO 将被设为“1”。



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 与 PDF 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	空闲模式或休眠模式时的 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
WDT，时基	复位后清零，且 WDT 开始计数
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。此单片机支持多个封装类型，该表格呈现的是较大封装类型的情况。

名称	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
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	---x xxxx	---u uuuu	---u uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -000	---- -uuu	---- -uuu
HIRCC	---- --01	---- --01	---- --uu
LVDC	--00 -000	--00 -000	--uu -uuu
LVRC	0110 0110	0110 0110	uuuu uuuu
SCC	000- --00	000- --00	uuu- --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
WDTC	0101 0011	0101 0011	uuuu uuuu
TBC	0--- -000	0--- -000	u--- -uuu
INTEG	---- --00	---- --00	---- -- uu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
PWMCS	---- 0000	---- 0000	---- uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
PC	--11 1111	--11 1111	--uu uuuu
PCC	--11 1111	--11 1111	-- uu uuuu
PCPU	--00 0000	--00 0000	-- uu uuuu
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	000- -000	000- -000	uuu- -uuu
CAPTC0	0000 0-00	0000 0-00	uuuu u-uu
CAPTC1	0000 0000	0000 0000	uuuu uuuu

名称	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
CAPTMDL	0000 0000	0000 0000	uuuu uuuu
CAPTMDH	0000 0000	0000 0000	uuuu uuuu
CAPTMAL	0000 0000	0000 0000	uuuu uuuu
CAPTMAH	0000 0000	0000 0000	uuuu uuuu
CAPTMCL	xxxx xxxx	xxxx xxxx	uuuu uuuu
CAPTMCH	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR2	0000 0-00	0000 0-00	uuu u-uu
ADDL	0000 0000	0000 0000	uuuu uuuu
CMPSEL	0000 000-	0000 000-	uuuu uuu-
CMPC	1111 0000	1111 0000	uuuu uuuu
OPOMS	110- -010	110- -010	uuu- -uuu
OPCM	0000 0000	0000 0000	uuuu uuuu
OPACAL	00-1 0000	00-1 0000	uu-u uuuu
SADOH	xxxx xxxx (ADRFS=0)	xxxx xxxx (ADRFS=0)	uuuu uuuu (ADRFS=0)
	---- xxxx (ADRFS=1)	---- xxxx (ADRFS=1)	---- uuuu (ADRFS=1)
SADOL	xxxx ---- (ADRFS=0)	xxxx ---- (ADRFS=0)	uuuu ---- (ADRFS=0)
	xxxx xxxx (ADRFS=1)	xxxx xxxx (ADRFS=1)	uuuu uuuu (ADRFS=1)
ADLVDH	0000 0000 (ADRFS=0)	0000 0000 (ADRFS=0)	uuuu uuuu (ADRFS=0)
	---- 0000 (ADRFS=1)	---- 0000 (ADRFS=1)	---- uuuu (ADRFS=1)
ADLVDL	---- 0000 (ADRFS=0)	---- 0000 (ADRFS=0)	uuuu ---- (ADRFS=0)
	0000 0000 (ADRFS=1)	0000 0000 (ADRFS=1)	uuuu uuuu (ADRFS=1)
ADHVDH	1111 1111 (ADRFS=0)	1111 1111 (ADRFS=0)	uuuu uuuu (ADRFS=0)
	---- 1111 (ADRFS=1)	---- 1111 (ADRFS=1)	---- uuuu (ADRFS=1)
ADHVDL	1111 ---- (ADRFS=0)	1111 ---- (ADRFS=0)	uuuu ---- (ADRFS=0)
	1111 1111 (ADRFS=1)	1111 1111 (ADRFS=1)	uuuu uuuu (ADRFS=1)
HINTEG	-000 0000	-000 0000	-uuu uuuu
PSCR	---- --00	---- --00	---- --uu
PWMC	0000 0000	0000 0000	uuuu uuuu
OCPS	--00 0000	--00 0000	--uu uuuu
HDCR	0001 0000	0001 0000	uuuu uuuu
HDCD	---- -000	---- -000	---- -uuu

名称	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
MPTC1	0000 0000	0000 0000	uuuu uuuu
MPTC2	-000 0000	-000 0000	-uuu uuuu
PTM1C0	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	uuuu uuuu
PTM1DH	0000 0000	0000 0000	uuuu uuuu
PTM1AL	0000 0000	0000 0000	uuuu uuuu
PTM1AH	0000 0000	0000 0000	uuuu uuuu
PTM1RPL	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	0000 0000	0000 0000	uuuu uuuu
PTM0C0	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	uuuu uuuu
PTM0DH	0000 0000	0000 0000	uuuu uuuu
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	0000 0000	0000 0000	uuuu uuuu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	0000 0000	0000 0000	uuuu uuuu
MDUWR0	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR1	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR2	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR3	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR4	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR5	xxxx xxxx	0000 0000	uuuu uuuu
MDUWCTRL	00-- ----	00-- ----	uu-- ----
DUTR0L	0000 0000	0000 0000	uuuu uuuu
DUTR0H	---- --00	---- --00	---- --uu
DUTR1L	0000 0000	0000 0000	uuuu uuuu
DUTR1H	---- --00	---- --00	---- --uu
DUTR2L	0000 0000	0000 0000	uuuu uuuu
DUTR2H	---- --00	---- --00	---- --uu
PRDRL	0000 0000	0000 0000	uuuu uuuu
PRDRH	---- --00	---- --00	---- --uu
PWMRL	0000 0000	0000 0000	uuuu uuuu
PWMRH	---- --00	---- --00	---- --uu
PWMME	--00 0000	--00 0000	--uu uuuu
PWMMD	--00 0000	--00 0000	--uu uuuu
MCF	0--- 0100	0--- 0100	u--- uuuu
MCD	--00 0xxx	--00 0 xxx	--uu uuuu
DTS	0000 0000	0000 0000	uuuu uuuu
PLC	--00 0000	--00 0000	--uu uuuu

名称	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
MFI0	-000 -000	-000 -000	-uuu -uuu
MFI1	0000 0000	0000 0000	uuuu uuuu
MFI2	0000 0000	0000 0000	uuuu uuuu
MFI3	--00 --00	--00 --00	--uu --uu
MFI4	--00 --00	--00 --00	--uu --uu
MFI5	--00 --00	--00 --00	--uu --uu
MFI6	--00 --00	--00 --00	--uu --uu
PD	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	uuuu uuuu
PDPU	0000 0000	0000 0000	uuuu uuuu
HCHK_NUM	---0 0000	---0 0000	---u uuuu
HNF_MSEL	---- 0000	---- 0000	---- uuuu
PAS0	0000 0000	0000 0000	uuuu 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
PCS1	---- 0000	---- 0000	---- uuuu
PDS0	0000 0000	0000 0000	uuuu uuuu
PDS1	0000 0000	0000 0000	uuuu uuuu
PTM2C0	0000 0---	0000 0---	uuuu u---
PTM2C1	0000 0000	0000 0000	uuuu uuuu
PTM2DL	0000 0000	0000 0000	uuuu uuuu
PTM2DH	---- --00	---- --00	---- --uu
PTM2AL	0000 0000	0000 0000	uuuu uuuu
PTM2AH	---- --00	---- --00	---- --uu
PTM2RPL	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	---- --00	---- --00	uuuu uuuu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	---- ---0	---- ---0	---- ---u
EED	0000 0000	0000 0000	uuuuuuuu
EEC	0000 0000	0000 0000	uuuu uuuu
ORMC	0000 0000	0000 0000	0000 0000
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu
UCR3	---- ---0	---- ---0	---- ---u
BRDH	0000 0000	0000 0000	uuuu uuuu
BRDL	0000 0000	0000 0000	uuuu uuuu
UFCR	--00 0000	--00 0000	--uu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu

名称	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
RxCNT	---- -000	---- -000	---- -uuu
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu
FC2	---- --00	---- --00	---- --uu
FARL	0000 0000	0000 0000	uuuu uuuu
FARH	---0 0000	---0 0000	---u uuuu
FD0L	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	uuuu uuuu
IFS0	0000 0000	0000 0000	uuuu uuuu
IFS1	---0 0000	---0 0000	---u uuuu
IFS3	0000 0000	0000 0000	uuuu uuuu
HDCT0	--00 0000	--00 0000	--uu uuuu
HDCT1	--00 0000	--00 0000	--uu uuuu
HDCT2	--00 0000	--00 0000	--uu uuuu
HDCT3	--00 0000	--00 0000	--uu uuuu
HDCT4	--00 0000	--00 0000	--uu uuuu
HDCT5	--00 0000	--00 0000	--uu uuuu
HDCT6	--00 0000	--00 0000	--uu uuuu
HDCT7	--00 0000	--00 0000	--uu uuuu
HDCT8	--00 0000	--00 0000	--uu uuuu
HDCT9	--00 0000	--00 0000	--uu uuuu
HDCT10	--00 0000	--00 0000	--uu uuuu
HDCT11	--00 0000	--00 0000	--uu uuuu
PTM3C0	0000 0---	0000 0---	uuuu u---
PTM3C1	0000 0000	0000 0000	uuuu uuuu
PTM3DL	0000 0000	0000 0000	uuuu uuuu
PTM3DH	---- --00	---- --00	---- --uu
PTM3AL	0000 0000	0000 0000	uuuu uuuu
PTM3AH	---- --00	---- --00	---- --uu
PTM3RPL	0000 0000	0000 0000	uuuu uuuu
PTM3RPH	---- --00	---- --00	---- --uu

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~PD 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	—	—	PC5	PC4	PC3	PC2	PC1	PC0
PCC	—	—	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0

“—”：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相应的上拉控制寄存器 PAPU~PDPU 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，当 I/O 引脚设为数字输入或 NMOS 输出时，上拉功能才会受 PxPU 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 Px 引脚上拉电阻控制位
0：除能
1：使能

PxPUn 位用于控制上拉电阻功能。这里的 x 可以是端口 A、B、C 或 D。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 功能输入类型且单片机处于空闲和休眠模式时，唤醒功能才会受 PAWU 控制开启，其它状态下此唤醒功能不可用。

- **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

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

- **PxC 寄存器**

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

PxCn：I/O Px 引脚输入 / 输出类型选择位
0：输出
1：输入
PxCn 位用于控制引脚类型。这里的 x 可以是端口 A、B、C 或 D。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”，记为 PxSn，和输入功能选择寄存器，记为 IFSi，这些寄存器可以用来选择共用引脚的特定功能。

当选择引脚共用输入功能，对应的输入和输出功能要进行合理设定。例如，CAPTM CTIN 引脚功能被选择，需通过 IFS0 寄存器选择 CTIN 信号的输入引脚，且所选择引脚上的共用功能要通过寄存器 PxSn 设置为 CTIN 功能。如果选择外部中断功能，相关的输出引脚共用功能应选择作为 I/O 功能，且也要选择中断输入信号。

要注意的最重要一点是，确保所需的引脚共用功能被正确地选择和取消。对于大部分共用功能，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位域时，一些数字输入引脚如 INT0、PTCKn 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	PAS05	PAS04	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
PCS1	—	—	—	—	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
IFS0	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00
IFS1	—	—	—	IFS14	D3	D2	IFS11	IFS10
IFS3	IFS37	IFS36	IFS35	IFS34	IFS33	IFS32	IFS31	IFS30

“—”：未定义，读为“0”

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	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
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 **PAS07~PAS06**：PA3 引脚共用功能选择
00：PA3/H1
01：C1P
10：AN5
11：PA3/H1

Bit 5~4 **PAS05~PAS04**：PA2 引脚共用功能选择
00：PA2/INT0
01：PTP1
10：PA2/INT0
11：PA2/INT0

Bit 3~2 **PAS03~PAS02**：PA1 引脚共用功能选择
00：PA1
01：AN3
10：PA1
11：AP

Bit 1~0 **PAS01~PAS00**：PA0 引脚共用功能选择
00：PA0
01：PTP0
10：AN0
11：PA0

- **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/PTCK2/INT0
01：AN7
10：PTP0
11：PTP0B

Bit 5~4 **PAS15~PAS14**：PA6 引脚共用功能选择
00：PA6/PTCK1/CTIN
01：AN8
10：PTP2
11：HBO

Bit 3~2 **PAS13~PAS12**：PA5 引脚共用功能选择
00：PA5/H3
01：C3P
10：AN2
11：PA5/H3

Bit 1~0 **PAS11~PAS10**：PA4 引脚共用功能选择
00：PA4/H2
01：C2P
10：AN4
11：PA4/H2

- **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/PTCK0/H1
01：C3N
10：PTP0
11：PTP0B

Bit 5~4 **PBS05~PBS04**：PB2 引脚共用功能选择
00：PB2/H2
01：C2N
10：AN9
11：PTP3

Bit 3~2 **PBS03~PBS02**：PB1 引脚共用功能选择
00：PB1/H3
01：C1N
10：CPN
11：AN6
当这些位设为 10B 选择 CPN 功能时，C1N、C2N 和 C3N 会由内部短接至 CPN 引脚，更多细节请参阅比较器章节。

Bit 1~0 **PBS01~PBS00**：PB0 引脚共用功能选择
00：PB0/PTCK3
01：AN1
10：HCO
11：PTP1

- **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/H1
01：PTP2
10：PTP2B
11：PB7/H1

Bit 5~4 **PBS15~PBS14**：PB6 引脚共用功能选择
00：PB6/H2
01：PTP3
10：PTP3B
11：OPAO

Bit 3~2 **PBS13~PBS12**：PB5 引脚共用功能选择
00：PB5/H3
01：PTP0
10：HAO
11：TX

Bit 1~0 **PBS11~PBS10**：PB4 引脚共用功能选择
00：PB4/CTIN
01：PTP0
10：RX/TX
11：PB4/CTIN

- **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 7~4 **PCS07~PCS06**：PC3 引脚共用功能选择
00：PC3
01：GBB
10：GBT
11：GAB

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

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

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

- **PCS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCS13	PCS12	PCS11	PCS10
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

Bit 3~2 **PCS13~PCS12**：PC5 引脚共用功能选择
00：PC5
01：GCB
10：GAT
11：GBB

Bit 1~0 **PCS11~PCS10**：PC4 引脚共用功能选择
00：PC4
01：GCT
10：GAB
11：GCB

- **PDS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
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 **PDS07~PDS06**：PD3 引脚共用功能选择
00：PD3/PTCK0
01：PTP1
10：HAO
11：GAB

Bit 5~4 **PDS05~PDS04**：PD2 引脚共用功能选择
00：PD2/PTCK1
01：PTP1B
10：HBO
11：GAT

Bit 3~2 **PDS03~PDS02**：PD1 引脚共用功能选择
00：PD1/PTCK2/INT0
01：PTP0

10：HCO
11：RX/TX

Bit 1~0 **PDS01~PDS00**：PD0 引脚共用功能选择
00：PD0/PTCK3/CTIN
01：PTP0B
10：PTP0
11：TX

- **PDS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
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 **PDS17~PDS16**：PD7 引脚共用功能选择
00：PD7/PTCK0/INT0
01：PTP3
10：PTP0B
11：GCB

Bit 5~4 **PDS15~PDS14**：PD6 引脚共用功能选择
00：PD6/PTCK1/CTIN
01：PTP3B
10：PTP0
11：GCT

Bit 3~2 **PDS13~PDS12**：PD5 引脚共用功能选择
00：PD5/PTCK2
01：PTP2
10：PTP1B
11：GBB

Bit 1~0 **PDS11~PDS10**：PD4 引脚共用功能选择
00：PD4/PTCK3
01：PTP2B
10：PTP1
11：GBT

- **IFS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00
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 **IFS07~IFS06**：H3 输入源引脚选择
00：PA5
01：PB1
10：PB5
11：PA5

Bit 5~4 **IFS05~IFS04**：H2 输入源引脚选择
00：PA4
01：PB2
10：PB6
11：PA4

Bit 3~2 **IFS03~IFS02**：H1 输入源引脚选择
00：PA3
01：PB3

10：PB7
11：PB3

Bit 1~0 **IFS01~IFS00**：CTIN 输入源引脚选择
00：PA6
01：PB4
10：PD0
11：PD6

- **IFS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	IFS14	D3	D2	IFS11	IFS10
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 未定义，读为“0”

Bit 4 **IFS14**：RX/TX 输入源引脚选择
0：PB4
1：PD1

Bit 3~2 **D3~D2**：保留，必须固定为 00

Bit 1~0 **IFS11~IFS10**：INT0 输入源引脚选择
00：PA7
01：PD1
10：PA2
11：PD7

- **IFS3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IFS37	IFS36	IFS35	IFS34	IFS33	IFS32	IFS31	IFS30
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 **IFS37~IFS36**：PTCK0 输入源引脚选择
00：PB3
01：PD3
10：PD7
11：PB3

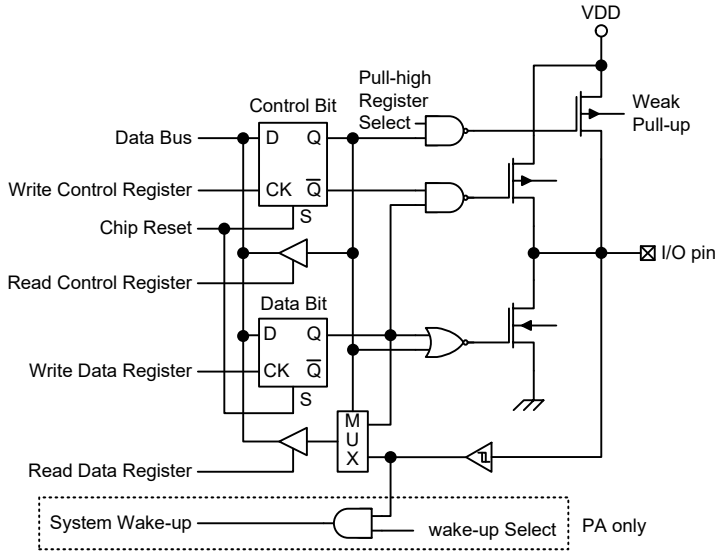
Bit 5~4 **IFS35~IFS34**：PTCK1 输入源引脚选择
00：PA6
01：PD2
10：PD6
11：PA6

Bit 3~2 **IFS33~IFS32**：PTCK2 输入源引脚选择
00：PA7
01：PD1
10：PD5
11：PA7

Bit 1~0 **IFS31~IFS30**：PTCK3 输入源引脚选择
00：PB0
01：PD0
10：PD4
11：PB0

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。该单片机提供几个定时器模块 (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 / 事件计数器，比较匹配输出，单脉冲输出以及 PWM 输出等功能。每个定时器模块有两个独立中断。每个 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 4 个 TM，各个 TM 的名称为 PTMn (n=0~3)。PTM0 和 PTM1 为 16-bit 周期型 TM，而 PTM2 和 PTM3 为 10-bit 周期型 TM。本章介绍 16-bit 和 10-bit 周期型 TM 的共性，更多详细资料详见后面章节。TM 的特性见下表。

功能	PTM
定时 / 计数器	√
比较匹配输出	√
PWM 输出	√
单脉冲输出	√
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

PTM0	PTM1	PTM2	PTM3
16-bit PTM	16-bit PTM	10-bit PTM	10-bit PTM

TM 名称 / 类型参考

TM 操作

不同类型的 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。用户选择内部时钟或外部时钟来驱动内部 TM 计数器。

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 PTMn 控制寄存器的 PTnCK2~PTnCK0 位，选择所需的时钟源，其中 n 代表具体 TM 编号。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} 时钟源或外部 PTCKn 引脚。PTCKn 引脚时钟源用于允许外部信号作为 TM 时钟源或用于事件计数。

TM 中断

每个周期型 TM 都有两个内部中断，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 输出引脚的状态。

TM 外部引脚

每个 TM 都有一个 TM 输入引脚，PTCKn。通过设置 PTMnC0 寄存器中的 PTnCK2~PTnCK0 位可选择 PTCKn 作为 PTMn 时钟源输入脚。外部时钟源可通过该引脚来驱动内部 TM。PTCKn 引脚可选择上升沿有效或下降沿有效。PTCKn 引脚

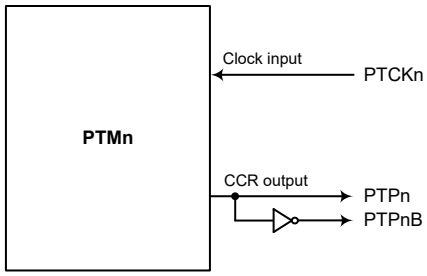
还可分别用作 PTMn 单脉冲输出模式的外部触发引脚。

每个 TM 都有两个输出引脚 PTPn 和 PTPnB。当 TM 工作在比较匹配输出模式且比较匹配发生时，PTPn 引脚会由 TM 控制切换到高电平或低电平或翻转。PTPnB 信号为 PTPn 输出的反相信号。当 TM 工作在 PWM 输出模式时，外部 PTPn 和 PTPnB 输出引脚也用于输出 PTMn 产生的 PWM 输出波形。

因 TM 输入和输出引脚与其它功能共用时，TM 输入和输出功能需要事先通过相关引脚共用功能选择寄存器进行设置。更多引脚共用功能选择详见引脚共用功能章节。

PTM3	
输入	输出
PTCKn	PTPn, PTPnB

TM 外部引脚

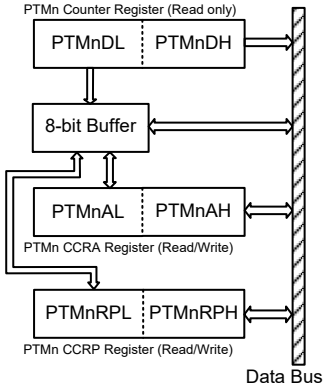


PTMn 功能引脚框图 (n=0~3)

编程注意事项

TM 计数寄存器和比较寄存器 CCRA 和 CCRP，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 和 CCRP 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 和 CCRP 低字节寄存器，即 PTMnAL 和 PTMnRPL，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 或 CCRP
  - ♦ 步骤 1. 写数据至低字节寄存器 PTMnAL 或 PTMnRPL

– 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 PTMnAH 或 PTMnRPH

– 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 或 CCRP 中读取数据

♦ 步骤 1. 由高字节寄存器 PTMnDH、PTMnAH 或 PTMnRPH 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 PTMnDL、PTMnAL 或 PTMnRPL 读取数据

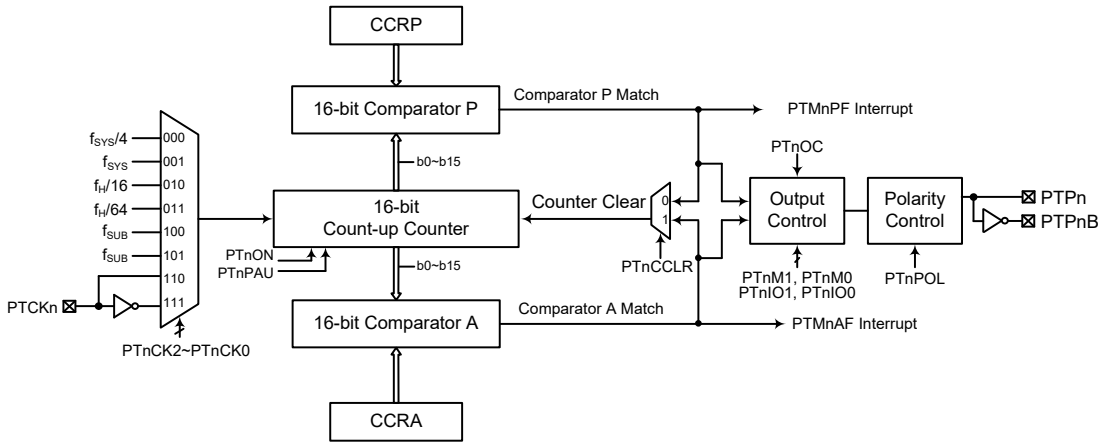
– 注意，此时读取 8-bit 缓存器中的数据。

周期型 TM – PTM

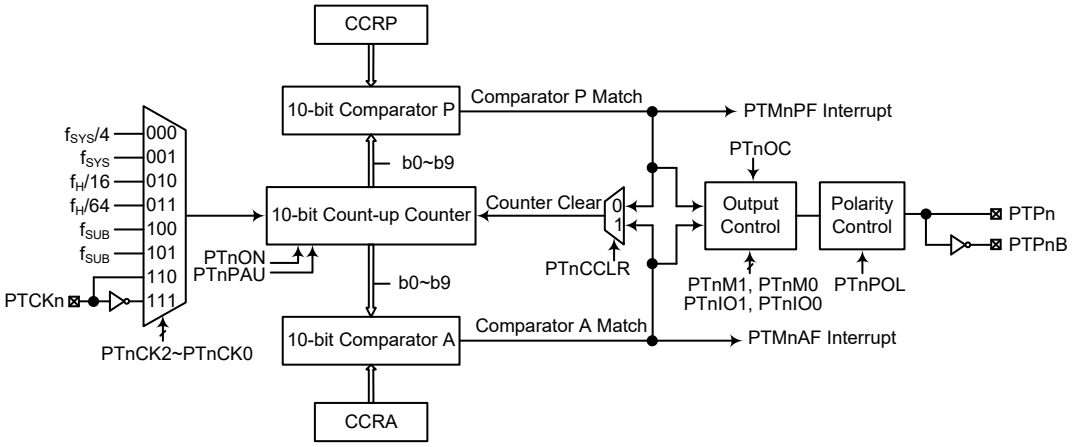
周期型 TM 包括 4 种工作模式，即比较匹配输出、定时 / 事件计数器、单脉冲输出和 PWM 输出模式。周期型 TM 由一个外部输入脚控制并驱动两个外部输出脚。

需注意的是，因 PTMn 功能引脚与其它功能共用引脚，并有多个引脚可同时用作 PTMn 信号的输出，因此在使用 PTMn 功能时，应先合理配置相关的引脚共用功能选择寄存器以确保 PTMn 功能的正常输入或输出。对于 PTCKn 引脚还需设置相应的端口控制寄存器，将该引脚设置为输入口。

类型	名称	TM 输入引脚	TM 输出引脚
16-bit PTM	PTM0，PTM1	PTCK0，PTCK1	PTP0，PTP1；PTP0B，PTP1B
10-bit PTM	PTM2，PTM3	PTCK2，PTCK3	PTP2，PTP3；PTP2B，PTP3B



16-Bit 周期型 TM 方框图 (n=0~1)



10-Bit 周期型 TM 方框图 (n=2~3)

周期型 TM 操作

周期型 TM 是 10 位或 16 位宽度。周期型 TM 核心是一个由用户选择的内部时钟源驱动的 10 位或 16 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 和 CCRA 是 10 位或 16 位的，与计数器的所有位比较。

通过应用程序改变 10 位或 16 位计数器值的唯一方法是使 PTnON 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 PTMn 中断信号。周期型 TM 可工作在不同的模式，可由不同时钟源驱动。所有工作模式的设定都是通过设置相关寄存器来实现的。

周期型 TM 寄存器介绍

周期型 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 位或 16 位计数器的值，两对读 / 写寄存器存放内部 10 位或 16 位 CCRA 和 CCRP 的值。剩下的几个寄存器用来设置不同的操作和控制模式。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	D1	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	D15	D14	D13	D12	D11	D10	D9	D8

16-Bit 周期型 TM 寄存器列表 (n=0~1)

寄存器名称	位							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	D1	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

10-Bit 周期型 TM 寄存器列表 (n=2~3)

- **PTMnC0 寄存器 (n=0~3)**

Bit	7	6	5	4	3	2	1	0
Name	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTnPAU**：PTMn 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。当处于暂停情况时，PTMn 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 **PTnCK2~PTnCK0**：选择 PTMn 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：PTCKn 上升沿
111：PTCKn 下降沿
此三位用于选择 PTMn 的时钟源。外部引脚时钟源能被选择在上升沿或下降沿有效。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 **PTnON**：PTMn 计数器 On/Off 控制位
0：Off
1：On
此位控制 PTMn 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 PTMn。清零此位将停止计数器并关闭 PTMn 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 PTMn 处于比较匹配输出模式、PWM 输出模式或单脉冲输出模式时，当 PTnON 位经由低到高转换时，PTMn 输出脚将复位至 PTnOC 位指定的初始值。

Bit 2~0 未定义，读为“0”

- **PTMnC1 寄存器 (n=0~3)**

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	D1	PTnCCLR
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 **PTnM1~PTnM0**：选择 PTMn 工作模式位
00：比较匹配输出模式
01：未定义
10：PWM 输出模式或单脉冲输出模式
11：定时 / 计数器模式
这两位设置 PTMn 需要的工作模式。为了确保操作可靠，PTMn 应在 PTnM1 和 PTnM0 位有任何改变前先关掉。在定时 / 计数器模式，PTMn 输出脚状态未定义。

Bit 5~4 **PTnIO1~PTnIO0**：选择 PTMn 外部引脚功能
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 / 单脉冲输出模式
00：强制无效状态
01：强制有效状态
10：PWM 输出
11：单脉冲输出
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 PTMn 外部引脚如何改变状态。这两位值的选择取决于 PTMn 运行在哪种模式下。
在比较匹配输出模式下，PTnIO1 和 PTnIO0 位决定当从比较器 A 比较匹配输出发生时 PTMn 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 PTMn 输出脚能设为切换高、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。PTMn 输出脚的初始值通过 PTMnC1 寄存器的 PTnOC 位设置取得。注意，由 PTnIO1 和 PTnIO0 位得到的输出电平必须与通过 PTnOC 位设置的初始值不同，否则当比较匹配发生时，PTMn 输出脚将不会发生变化。在 PTMn 输出脚改变状态后，通过 PTnON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，PTnIO1 和 PTnIO0 用于决定比较匹配条件发生时怎样改变 PTMn 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。仅在 PTMn 关闭时改变 PTnIO1 和 PTnIO0 位的值是很有必要的。若在 PTMn 运行时改变 PTnIO1 和 PTnIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **PTnOC**：PTMn 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 / 单脉冲输出模式
0：低有效
1：高有效
这是 PTMn 输出脚输出控制位。它取决于 PTMn 此时正运行于比较匹配输出模式还是 PWM 输出模式 / 单脉冲输出模式。若 PTMn 处于定时 / 计数器模式，则其无效。在比较匹配输出模式时，其决定比较匹配发生前 PTMn 输出脚的逻辑电平值。在 PWM 输出模式时，其决定 PWM 信号是高有效还是低有效。在单脉冲输出模式时，其决定 PTnON 位由低变高时 PTMn 输出脚的逻辑电平。

Bit 2 **PTnPOL**：PTPn 输出极性控制位
0：同相
1：反相
此位控制输出脚的极性。此位为高时 PTMn 输出脚反相，为低时 PTMn 输出脚同相。若 PTMn 处于定时 / 计数器模式时其无效。

Bit 1 **D1**：保留，读为“0”

Bit 0 **PTnCCLR**：选择 PTMn 计数器清零条件位
0：PTMn 比较器 P 匹配
1：PTMn 比较器 A 匹配
此位用于选择清除计数器的方法。周期型 TM 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。PTnCCLR 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。PTnCCLR 位在 PWM 输出模式或单脉冲输出模式时未使用。

- **PTMnDL 寄存器 (n=0~3)**

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**：PTMn 计数器低字节寄存器 bit 7 ~ bit 0
PTMn 10-bit/16-bit 计数器 bit 7 ~ bit 0

- **PTMnDH 寄存器 (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8**：PTMn 计数器高字节寄存器 bit 7 ~ bit 0
PTMn 16-bit 计数器 bit 15 ~ bit 8

- **PTMnDH 寄存器 (n=2~3)**

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**：PTMn 计数器高字节寄存器 bit 1 ~ bit 0
PTMn 10-bit 计数器 bit 9 ~ bit 8

- **PTMnAL 寄存器 (n=0~3)**

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**：PTMn CCRA 低字节寄存器 bit 7 ~ bit 0
PTMn 10-bit/16-bit CCRA bit 7 ~ bit 0

- **PTMnAH 寄存器 (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**：PTMn CCRA 高字节寄存器 bit 7 ~ bit 0
PTMn 16-bit CCRA bit 15 ~ bit 8

- **PTMnAH 寄存器 (n=2~3)**

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**：PTMn CCRA 高字节寄存器 bit 1 ~ bit 0
PTMn 10-bit CCRA bit 9 ~ bit 8

- **PTMnRPL 寄存器 (n=0~3)**

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**：PTMn CCRP 低字节寄存器 bit 7 ~ bit 0
PTMn 10-bit/16-bit CCRP bit 7 ~ bit 0

- **PTMnRPH 寄存器 (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**：PTMn CCRP 高字节寄存器 bit 7 ~ bit 0
PTMn 16-bit CCRP bit 15 ~ bit 8

- **PTMnRPH 寄存器 (n=2~3)**

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**：PTMn CCRP 高字节寄存器 bit 1 ~ bit 0
PTMn 10-bit CCRP bit 9 ~ bit 8

周期型 TM 工作模式

周期型 TM 有四种工作模式，即比较匹配输出模式、PWM 输出模式、单脉冲输出模式或定时 / 计数器模式。通过设置 PTMnC1 寄存器的 PTnM1 和 PTnM0 位选择任意模式。

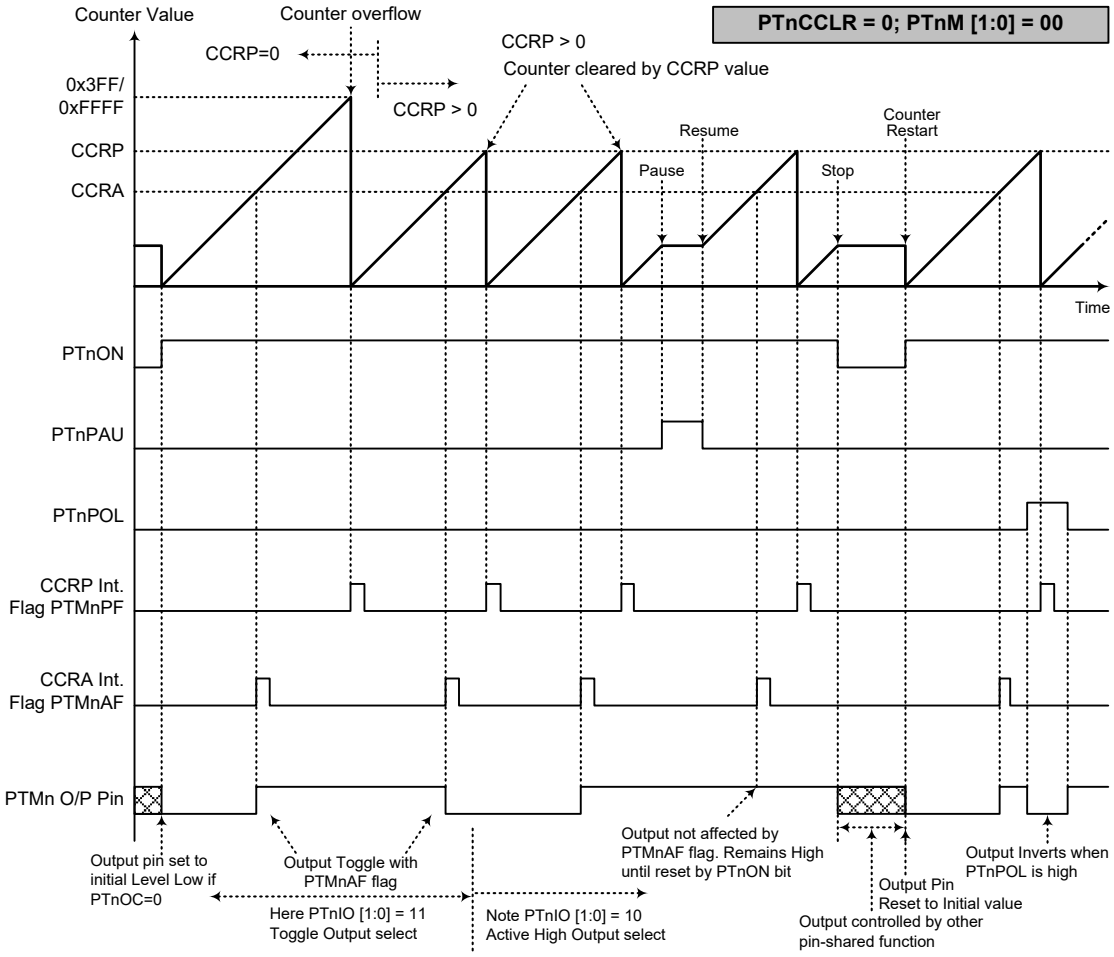
比较匹配输出模式

为使 PTMn 工作在此模式，PTMnC1 寄存器的 PTnM1 和 PTnM0 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 PTnCCLR 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 PTMnAF 和 PTMnPF 将分别置起。

如果 PTMnC1 寄存器的 PTnCCLR 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 PTMnAF 中断请求标志产生。所以当 PTnCCLR 为高时，不会产生 PTMnPF 中断请求标志。在比较匹配输出模式中，CCRA 寄存器值不能被清为“0”。

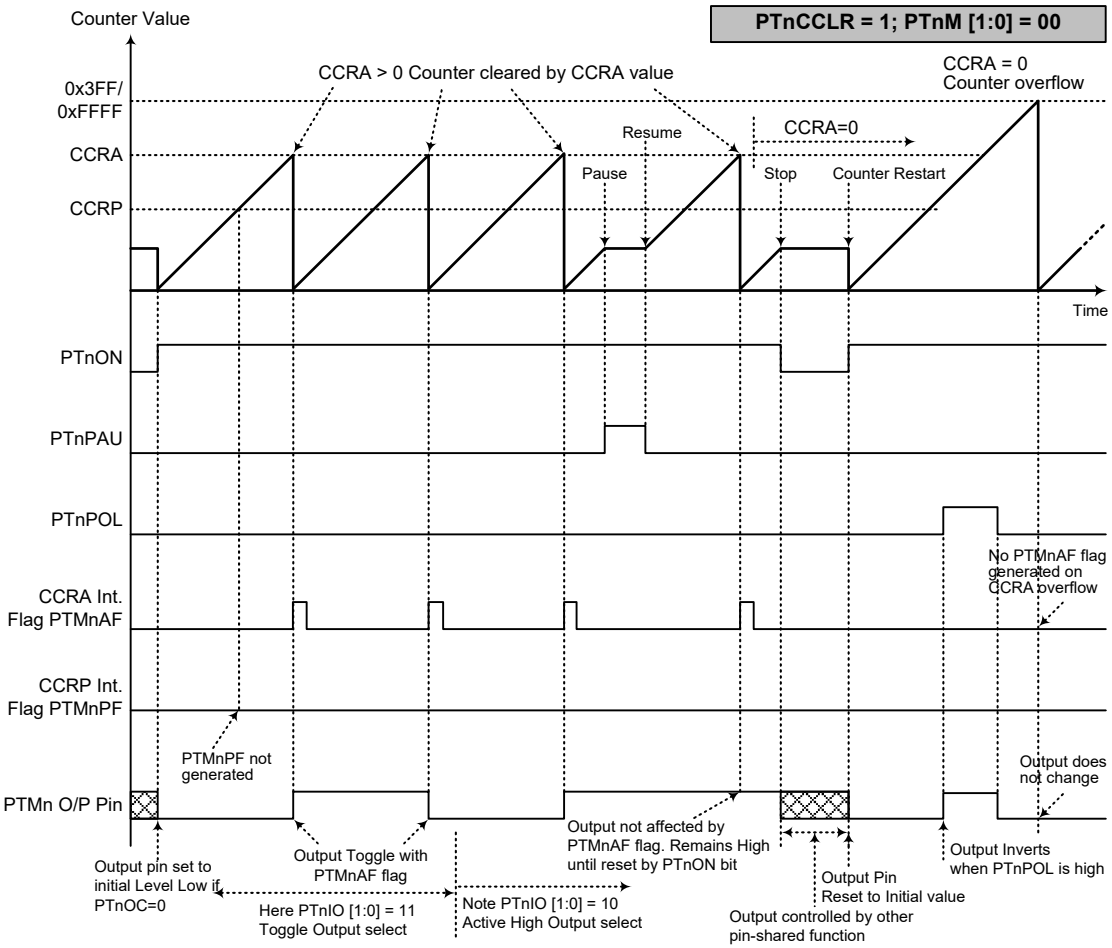
如果 CCRA 位都清除为零，当计数器的值达到 10 位最大值 3FFH 时或 16 位最大值 FFFFH 时将溢出，但此时不会产生 PTMnAF 中断请求标志。

正如该模式名所言，当比较匹配发生后，PTMn 输出脚状态改变。当比较器 A 比较匹配发生后 PTMnAF 中断请求标志产生时，PTMn 输出脚状态改变。比较器 P 比较匹配发生时产生的 PTMnPF 标志不影响 PTMn 输出脚。PTMn 输出脚状态改变方式由 PTMnC1 寄存器中 PTnIO1 和 PTnIO0 位决定。当比较器 A 比较匹配发生时，PTnIO1 和 PTnIO0 位决定 PTMn 输出脚输出高，低或翻转当前状态。在 PTnON 位由低到高后，PTMn 输出脚初始状态为 PTnOC 位所指定的电平。注意，若 PTnIO1 和 PTnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – PTnCCLR=0 (n=0~3)

注：1. PTnCCLR=0，比较器 P 匹配将清除计数器
2. PTMn 输出脚仅由 PTMnAF 标志位控制
3. 在 PTnON 上升沿 PTMn 输出脚复位至初始值



比较匹配输出模式 – PTnCCLR=1 (n=0~3)

注：1. PTnCCLR=1，比较器 A 匹配将清除计数器
2. PTMn 输出脚仅由 PTMnAF 标志位控制
3. 在 PTnON 上升沿 PTMn 输出脚复位至初始值
4. 当 PTnCCLR=1 时，不会产生 PTMnPF 标志

定时 / 计数器模式

为使 PTMn 工作在此模式，PTMnC1 寄存器的 PTnM1 和 PTnM0 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 PTMn 输出脚未使用。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 PTMn 输出脚用作普通 I/O 引脚或其它功能。

PWM 输出模式

为使 PTMn 工作在此模式，PTMnC1 寄存器的 PTnM1 和 PTnM0 位需要设置为“10”，且 PTnIO1 和 PTnIO0 位也需要设置为“10”。PTMn 的 PWM 功能在电机控制，加热控制，照明控制等方面十分有用。给 PTMn 输出脚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，其波形的选择就极其灵活。在 PWM 输出模式中，PTnCCLR 位对 PWM 周期无影响。CCRP 和 CCRA 寄存器都用于控制 PWM 方波。CCRP 寄存器通过清除内部计数从而控制 PWM 周期，CCRA 寄存器设置 PWM 的占空比。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。PTMnC1 寄存器的 PTnOC 位选择 PWM 波形的极性，PTnIO1 和 PTnIO0 位使能 PWM 输出或强制 PTMn 输出脚为高电平或低电平。PTnPOL 位用于 PWM 输出波形的极性反相控制。

- **16-bit PTMn，PWM 输出模式，边沿对齐模式**

CCRP	1~65535	0
Period	1~65535	65536
Duty	CCRA	

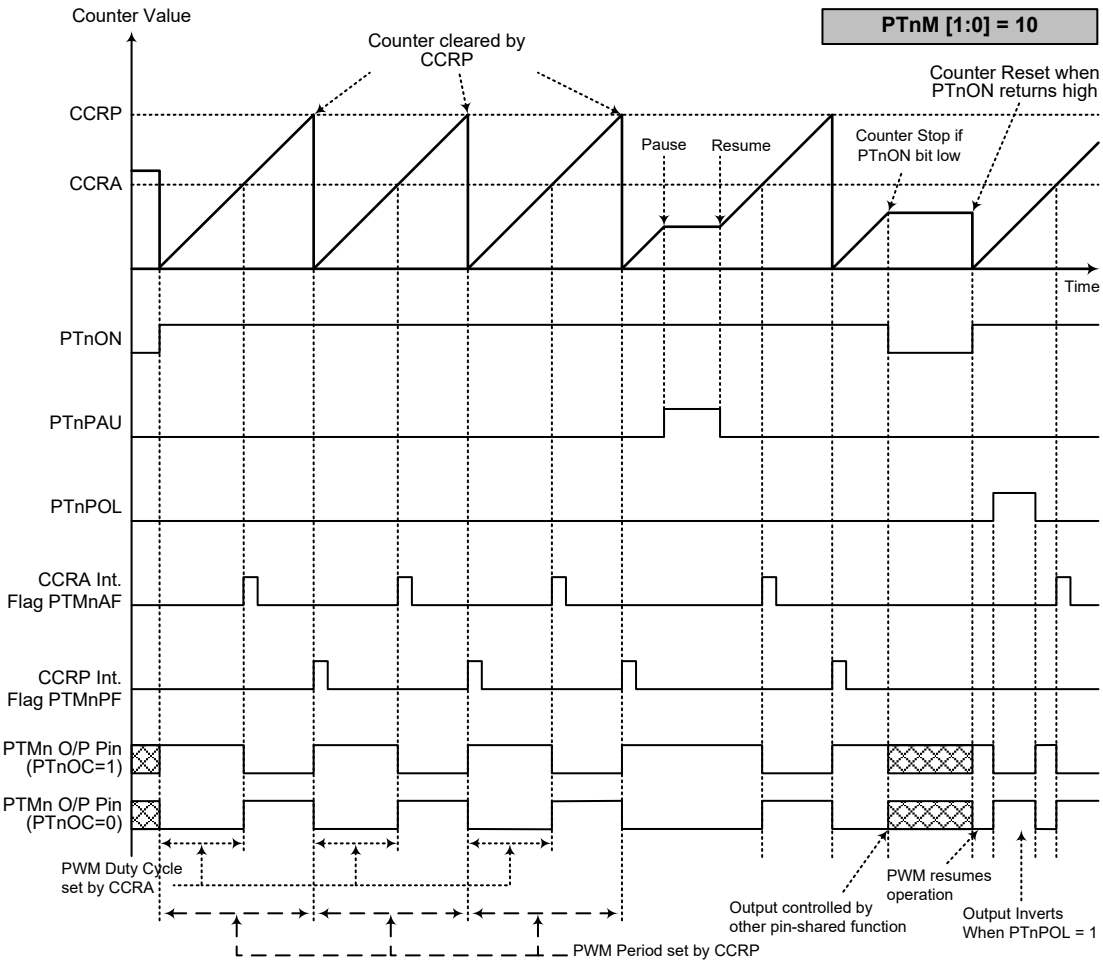
- **10-bit PTMn，PWM 输出模式，边沿对齐模式**

CCRP	1~1023	0
Period	1~1023	1024
Duty	CCRA	

若 f_{SYS}=4MHz，PTMn 时钟源选择 f_{SYS}/4，CCRP=512 且 CCRA=128，

PTMn PWM 输出频率 = (f_{SYS}/4)/512=f_{SYS}/2048=2kHz，duty=128/512=25%。

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%。



PWM 输出模式 (n=0~3)

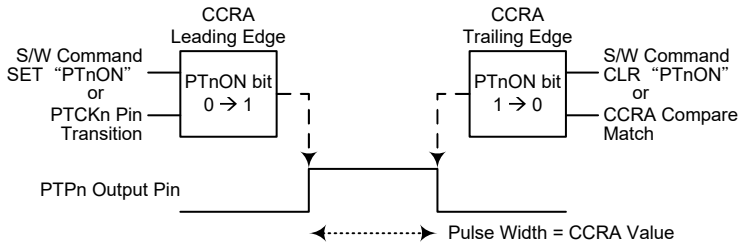
注：1. CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 PTnIO[1:0]=00 或 01，PWM 功能不变
4. PTnCCLR 位对 PWM 功能无影响

单脉冲输出模式

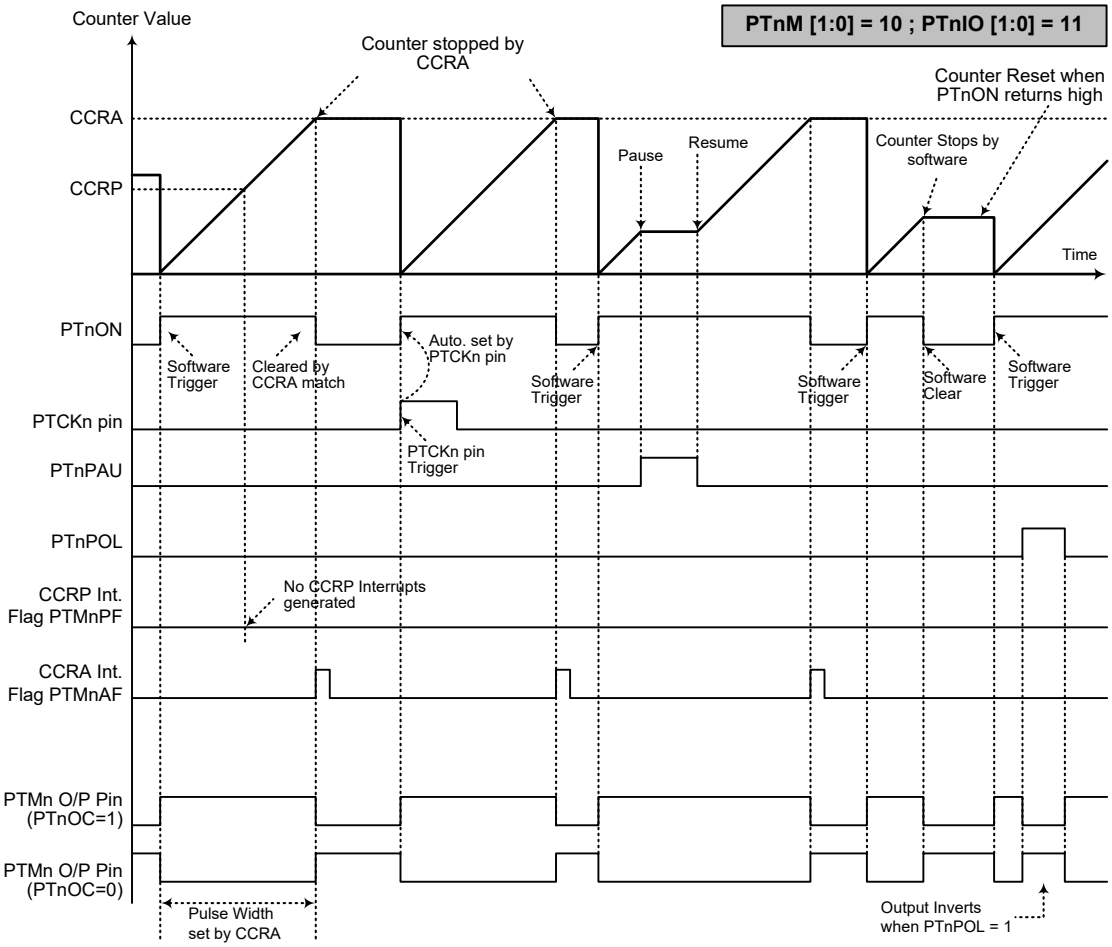
为使 PTMn 工作在此模式，PTMnC1 寄存器中的 PTnM1 和 PTnM0 位需要设置为“10”，并且相应的 PTnIO1 和 PTnIO0 需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTMn 输出脚将产生一个脉冲输出。

通过应用程序控制 PTnON 位由低到高的转变来触发脉冲前沿输出。而处于单脉冲输出模式时，PTnON 位可在 PTCKn 脚发生有效边沿跳转时自动由低转变为高，进而开始单脉冲输出。当 PTnON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。通过应用程序使 PTnON 位清零或比较器 A 比较匹配发生时，产生脉冲后沿。

而比较器 A 比较匹配发生时，会自动清除 PTnON 位并产生单脉冲输出边沿跳转。CCRA 的值通过这种方式控制脉冲宽度。比较器 A 比较匹配发生时，也会产生 PTMn 中断。PTnON 位在计数器重启时会发生由低到高的转变，此时计数器才复位至零。在单脉冲输出模式中，CCRP 寄存器和 PTnCCLR 位未使用。



单脉冲产生示意图 (n=0~3)



单脉冲输出模式 (n=0~3)

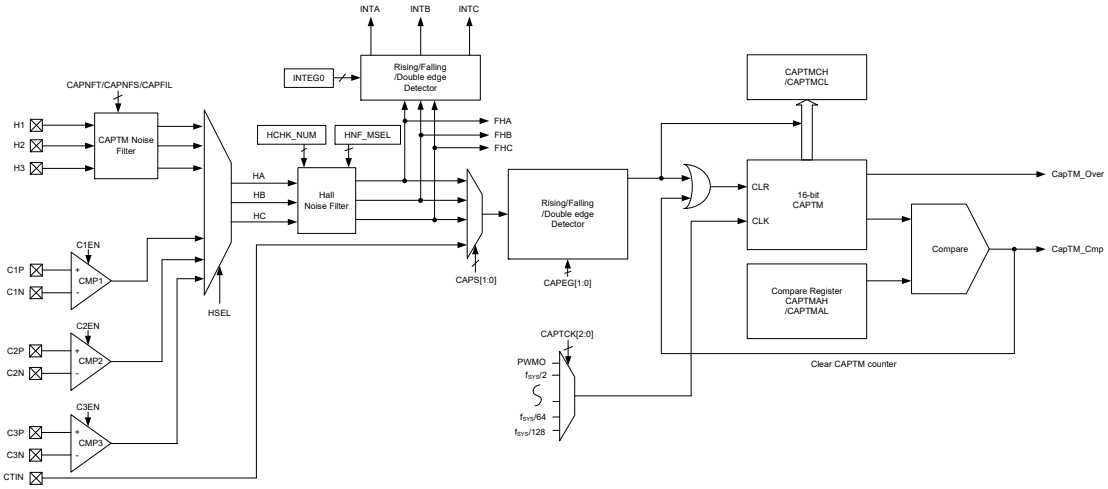
注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCKn 脚或设置 PTnON 位为高来触发脉冲
4. PTCKn 脚有效沿会自动置位 PTnON
5. 单脉冲输出模式中，PTnIO[1:0] 需置位“11”，且不能更改

捕捉定时器模块 – CAPTM

捕捉定时器模块是一个专门用于电机控制的定时单元。CAPTM 由程序选择的时钟源控制。

捕捉定时器简介

捕捉定时器核心是一个由用户选择的内部时钟源驱动的 16 位向上计数器，时钟源可来自系统时钟分频或 PWMO 周期信号。该捕捉定时器包含一个内部比较器用于将计数器的值与预先存储在两个寄存器中的值进行比较。它有两个基本工作模式，即比较模式和捕捉模式，每一个都可以用来复位内部计数器。当到达比较匹配情况时，会产生一个信号将内部计数器复位。当由 FHA，FHB，FHC 和 CTIN 其中一个触发捕捉时，计数器也将被清除。



注：关于霍尔噪声滤波器的控制及输入选择，请参阅霍尔传感器噪声滤波器章节。

捕捉定时器方框图

捕捉定时器寄存器介绍

捕捉定时器的所有操作由 8 个寄存器控制。一对只读寄存器用来存放 16 位计数器的值，一对读 / 写寄存器用来存放 16 位比较值，另外一对只读寄存器用来存放捕捉值，剩下两个控制寄存器用于设置不同的操作和控制模式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CAPTC0	CAPTPAU	CAPTCK2	CAPTCK1	CAPTCK0	CAPTON	—	CAPS1	CAPS0
CAPTC1	CAPEG1	CAPEG0	CAPEN	CAPNFT	CAPNFS	CAPFIL	CAPCLR	CAMCLR
CAPTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMDH	D15	D14	D13	D12	D11	D10	D9	D8
CAPTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMAH	D15	D14	D13	D12	D11	D10	D9	D8
CAPTMCL	D7	D6	D5	D4	D3	D2	D1	D0
CAPTMCH	D15	D14	D13	D12	D11	D10	D9	D8

捕捉定时器寄存器列表

- **CAPTC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CAPTPAU	CAPTCK2	CAPTCK1	CAPTCK0	CAPTON	—	CAPS1	CAPS0
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 **CAPTPAU**：CAPTM 计数器暂停控制位
0：运行
1：暂停
将该位设置为高可以暂停计数器，清为零可以将计数器重启操作。当处于暂停状态时，CAPTM 将保持上电状态且继续消耗功耗。当该位从低到高的转变时，计数器将会保持原来的值；当该位又从高到低转变时，则计数器将从原来的值重新开始计数。

Bit 6~4 **CAPTCK2~CAPTCK0**：选择 CAPTM 计数时钟
000：PWMO 周期性信号
001：$f_{SYS}/2$
010：$f_{SYS}/4$
011：$f_{SYS}/8$
100：$f_{SYS}/16$
101：$f_{SYS}/32$
110：$f_{SYS}/64$
111：$f_{SYS}/128$
这三位用于选择 CAPTM 时钟源。

Bit 3 **CAPTON**：CAPTM 计数器开 / 关控制位
0：关闭
1：开启
该位为 CAPTM 总的开 / 关控制位。将该位设为高，将使能计数器，清为零将除能 CAPTM。将该位清为零将停止计数器计数并关闭 CAPTM 以节省功耗。当该位从低到高的转变时，内部计数器值将复位到零；当该位从高到低时，内部计数器将保持原来的值。

Bit 2 未定义，读为“0”

Bit 1~0 **CAPS1~CAPS0**：CAPTM 捕捉源选择
00：FHA
01：FHB
10：FHC
11：CTIN 引脚

- **CAPTC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CAPEG1	CAPEG0	CAPEN	CAPNFT	CAPNFS	CAPFIL	CAPCLR	CAMCLR
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 **CAPEG1~CAPEG0**：CAPTM 捕捉有效边沿选择位
00：除能 CAPTM 捕捉
01：上升沿捕捉
10：下降沿捕捉
11：双边沿捕捉

Bit 5 **CAPEN**：CAPTM 捕捉输入控制
0：除能
1：使能

Bit 4 **CAPNFT**：CAPTM 噪声滤波采样次数
0：2 次
1：4 次
CAPTM 噪声滤波电路需连续 2 次或 4 次采样 CAPTM 输入信号，当多次采样都相同时，信号才会被承认，采样的频率由 CAPNFS 决定。

Bit 3 **CAPNFS**：CAPTM 噪声滤波时钟源选择位
0：f_{SYS}
1：$f_{SYS}/4$
CAPTM 噪声滤波器的时钟来自 f_{SYS} 或 $f_{SYS}/4$。

Bit 2 **CAPFIL**：CAPTM 捕捉输入噪声滤波器控制
0：除能
1：使能
该位用于使能/除能 CAPTM 捕捉输入噪声滤波器。若 CAPTON=1 且 CAPFIL=1，则 CAPTM 捕捉输入噪声滤波器将被使能。

Bit 1 **CAPCLR**：CAPTM 计数器捕捉自动复位控制
0：除能
1：使能
当此位置 1 且当 FHA/FHB/FHC/CTIN 产生有效触发边沿时，硬件自动将 CAPTMDL 和 CAPTMDH 的值传送到捕捉寄存器 CAPTMCL 和 CAPTMCH，然后自动复位 CAPTM 计数器。

Bit 0 **CAMCLR**：CAPTM 计数器比较匹配自动复位控制
0：除能
1：使能
当此位置 1 且比较匹配发生时，硬件将自动复位 CAPTM 计数器。当 CAPTMAH/CAPTMAL=000H，也可产生 CAPTM 比较匹配中断。

- **CAPTMDL 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**：CAPTM 计数器低字节寄存器 bit 7 ~ bit 0
CAPTM 16-bit 计数器 bit 7 ~ bit 0

- **CAPTMDH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8**：CAPTM 计数器高字节寄存器 bit 7 ~ bit 0
CAPTM 16-bit 计数器 bit 15 ~ bit 8

- **CAPTMAL 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**：CAPTM 比较低字节寄存器 bit 7 ~ bit 0
CAPTM 16-bit 比较寄存器 bit 7 ~ bit 0

- **CAPTMAH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**：CAPTM 比较高字节寄存器 bit 7 ~ bit 0
CAPTM 16-bit 比较寄存器 bit 15 ~ bit 8

- **CAPTMCL 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	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0 **D7~D0**：CAPTM 捕捉低字节寄存器 bit 7 ~ bit 0
CAPTM 16-bit 捕捉寄存器 bit 7 ~ bit 0

- **CAPTMCH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x

“x”：未知

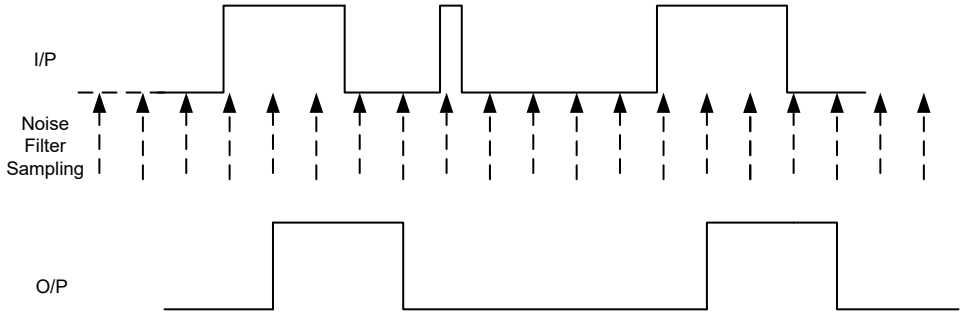
Bit 7~0 **D15~D8**：CAPTM 捕捉高字节寄存器 bit 7 ~ bit 0
CAPTM 16-bit 捕捉寄存器 bit 15 ~ bit 8

捕捉定时器操作

捕捉定时器模块可以用于检测和测量输入信号的脉冲宽度和周期。它可工作在捕捉模式或比较模式。定时器有 4 个捕捉输入，即 FHA，FHB，FHC 和 CTIN。每个捕捉输入都有自己的有效边沿选择位，可选择上升沿，下降沿或者双边沿捕捉。

CAPTON 位用来控制整个捕捉定时器使能 / 除能功能。若不使用捕捉定时器模块时，除能该模块可减少单片机耗电。通过 CAPEN 控制位来使能 / 除能捕捉输入。通过 CAPEG1 和 CAPEG0 设置触发边沿选项来选择在上升沿，下降沿或者双边沿捕捉。

该定时器还包含一个内部捕捉定时器噪声滤波器，用来滤除 H1，H2 和 H3 输入脚上不必要的干扰或脉冲。噪声滤波器可以通过 CAPFIL 位使能。如果噪声滤波器使能，CAPTM 噪声滤波电路可对引脚上输入的信号进行连续 2 次或 4 次采样，从而过滤出真正的有效信号。采样次数是基于采样时钟选择，采样的时钟由 CAPNFS 选择为 f_{SYS} 或 $f_{SYS}/4$。



噪声滤波器时序 – CAPNFT=0 & CAPNFS=0

捕捉模式操作

捕捉定时器模块包含两个捕捉寄存器，CAPTMCL 和 CAPTMCH，用于存储捕捉到的数据。当捕捉输入使能，捕捉输入源上发生有效的触发信号时，存储于寄存器 CAPTMDL 和 CAPTMDH 内的 16 位向上计数值将被捕捉到捕捉寄存器 CAPTMCL 和 CAPTMCH 内。若计数溢出，中断控制寄存器中的 CAPTM 捕捉溢出中断请求标志位 CAPOF 位将被置位。如果设置 CAPCLR 位为高，捕捉溢出事件发生后将自动复位 CAPTM 计数器。

比较模式操作

当定时器工作在比较模式时，寄存器 CAPTMAL 和 CAPTMAH 用来存储 16 位比较值。当 CAPTM 计数器向上计数达到 CAPTMAL 和 CAPTMAH 中的值时，比较匹配发生。中断控制寄存器中的 CAPTM 比较匹配中断请求标志位 CAPCF 位将被置位。若 CAMCLR 位为高，当比较匹配发生时，可以自动复位 CAPTM 计数器。

在电机应用中，通过设置比较寄存器，动态比对捕捉信号边沿转换时间，可检测转子的移动速度，据此判断电机是否发生堵转情况。若有堵转情形，将产生比较中断，可以选择由硬件或软件模式将 PWM 电机驱动电路关闭，以保护电机不会被烧毁。

A/D 转换器

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

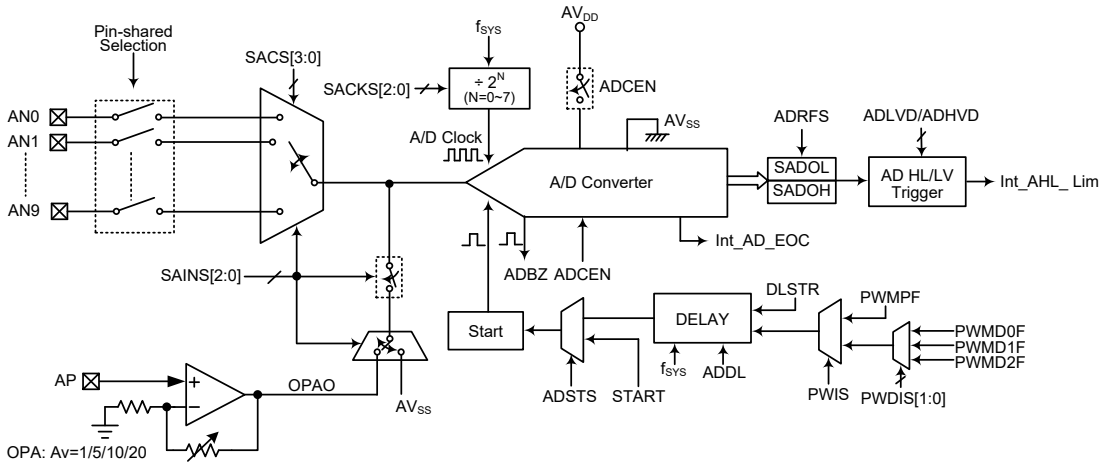
A/D 转换器简介

此单片机包含一个多通道的 A/D 转换器，10 个通道可以直接接入外部模拟信号 (来自传感器或其它控制信号) 或内部模拟信号 (比如过流检测运算放大器输出) 并直接将这些信号转换成 12 位的数字量。另一个通道来自 AP 引脚 (运算放大器正端输入)，由 AP 引脚输入的信号经过运算放大器放大后可输出到 A/D 转换器。选择转换外部或内部模拟信号由 SAINS3~SAINS0 位和 SACS3~SACS0 位共同控制。应注意的是，若通过 SAINS2~SAINS0 位选择内部模拟信号，则外部通道模拟输入将被自动关闭以避免冲突。关于 A/D 转换器输入信号的详细描述可参考“A/D 转换器输入信号”章节进行了解。

此单片机提供两对边界寄存器，用于定义转换的上限值和下限值。实际的 A/D 转换结果值与预置的边界寄存器中的数值进行比较，当比较结果满足设定的条件时，则触发对应的 A/D 转换值比较结果中断。另外，提供了一个延迟触发电路。当选择 PWM 中断来启动 A/D 转换时，此延迟电路可在 PWM 中断与 A/D 转换开始之前加入一段延迟时间，以减少误采样的发生。

外部输入通道	内部输入信号	A/D 通道选择位
AN0~AN9	OPAO，AV_{SS}	SAINS2~SAINS0 SACS3~SACS0

下图显示了 A/D 转换器内部结构和相关的寄存器。



A/D 转换器结构

A/D 转换器寄存器介绍

A/D 转换器的整个工作由一系列寄存器控制。SADOH 和 SADOL 这对只读寄存器用来存放 12 位的 A/D 转换数据的值。两对寄存器，ADHVDH/ADHVDL 和 ADLVDH/ADLVDL 用来存放 A/D 转换器触发的上下边界值。ADDL 寄存器用来设置 PWM 中断触发后 A/D 转换启动延迟时间。剩下的控制寄存器设置 A/D 转换器的操作和控制功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADLVDL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADLVDL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADLVDH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADLVDH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADHVDL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADHVDL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADHVDH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADHVDH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	—	—	SACKS2	SACKS1	SACKS0
ADCR2	ADSTS	DLSTR	PWIS	ADCHVE	ADCLVE	—	PWDIS1	PWDIS0
ADDL	D7	D6	D5	D4	D3	D2	D1	D0

A/D 转换器寄存器列表

A/D 转换器数据寄存器

对于 12 位的 A/D 转换器，需要两个数据寄存器存放转换结果。一个高字节寄存器 SADOH 和一个低字节寄存器 SADOL。在 A/D 转换完毕后，单片机可以直接读取这些寄存器以获得转换结果。由于寄存器只使用了 16 位中的 12 位，其数据存储格式由 SADC0 寄存器的 ADRFS 位控制，如下表所示。D0~D11 是 A/D 换转数据结果位。未使用的位读为“0”。当 A/D 转换器除能时，数据寄存器的内容不变。

ADRFS	SADOH								SADOL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D 转换器数据寄存器

A/D 转换器控制寄存器 – SADC0, SADC1, ADCR2, ADDL

寄存器 SADC0、SADC1 和 ADCR2 用来控制 A/D 转换器的功能和操作。这些 8 位的寄存器定义包括选择连接至内部 A/D 转换器的模拟通道，数据转换格式，A/D 时钟源，并控制和监视 A/D 转换器的开始和转换结束状态。由于每个单片机只包含一个实际的模数转换电路，因此这些外部和内部模拟信号只能单个的被发送到转换器。SADC0 寄存器中的 SACS3~SACS0 位用于选择哪个外部模拟输入通道被连接到内部 A/D 转换器。SADC1 寄存器中的 SAINS3~SAINS0 位用于选择外部模拟输入通道或内部模拟信号被连接到内部 A/D 转换器。若选择 DELAY 电路触发 A/D 转换，ADDL 寄存器用于设置延迟时间。

引脚共用功能控制寄存器的相关位用来定义 I/O 端口中的哪些引脚为 A/D 转换器的模拟输入，哪些引脚不作为 A/D 转换输入。当引脚作为 A/D 输入时，其原来的通用 I/O 功能或其它引脚共用功能消失，此外，其内部上拉电阻也将自动断开。

- **SADC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **START**：启动 A/D 转换位
0→1→0：启动 A/D 转换
此位用于启动 A/D 转换过程。通常此位为低，但如果设为高再被清零，将启动 A/D 转换过程。

Bit 6 **ADBZ**：A/D 转换忙碌标志位
0：A/D 转换结束或未开始转换
1：A/D 转换中
此位用于表明 A/D 转换过程是否完成。当 START 位由低变为高再变为低时，ADBZ 位为高，表明 A/D 转换已初始化。A/D 转换结束后，此位被清零。

Bit 5 **ADCEN**：A/D 转换器使能 / 除能控制位
0：除能
1：使能
此位控制 A/D 内部功能。该位被置高将使能 A/D 转换器。如果该位设为低将关闭 A/D 转换器以降低功耗。当 A/D 转换器除能时，A/D 数据寄存器 SADOH 和 SADOL 的值保持不变。

Bit 4 **ADRFS**：A/D 转换数据格式选择位
0：A/D 转换数据格式 → SADOH=D[11:4]，SADOL=D[3:0]
1：A/D 转换数据格式 → SADOH=D[11:8]，SADOL=D[7:0]
此位控制存放在两个 A/D 数据寄存器中的 12 位 A/D 转换结果的格式。细节方面请参考 A/D 数据寄存器章节。

Bit 3~0 **SACS3~SACS0**：A/D 转换器输入通道选择位
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：AN4
0101：AN5
0110：AN6
0111：AN7
1000：AN8
1001：AN9
1010~1111：未定义，输入浮空

- **SADC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	—	—	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7~5 **SAINS2~SAINS0**：A/D 输入信号选择位
000：外部来源 – 外部模拟通道输入 ANn
001：内部来源 – 内部 OPA 输出信号 OPAO
010：内部来源 – 接地
011：内部来源 – 接地
100~111：外部来源 – 外部模拟通道输入 ANn
当选择转换内部模拟信号时，无论 SACS2~SACS0 为何值，外部通道输入信号都会自动关闭。此举预防了外部通道输入与内部模拟信号连接从而导致的不可预期的后果。

Bit 4~3 未定义，读为“00”

Bit 2~0 **SACKS2~SACKS0**：A/D 时钟源选择位
000：f_{SYS}
001：$f_{SYS}/2$
010：$f_{SYS}/4$
011：$f_{SYS}/8$
100：$f_{SYS}/16$
101：$f_{SYS}/32$
110：$f_{SYS}/64$
111：$f_{SYS}/128$

- **ADCR2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ADSTS	DLSTR	PWIS	ADCHVE	ADCLVE	—	PWDIS1	PWDIS0
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 **ADSTS**：选择 A/D 转换开始触发电路
0：选择 START 启动位触发 A/D 转换开始
1：选择通过 DELAY 电路触发 A/D 转换开始

Bit 6 **DLSTR**：DELAY 启动电路控制位
0：除能
1：使能
当此位为零，ADDL 寄存器值必须清零。当此位置位，ADDL 寄存器必须设置为除“00H”以外的值。

Bit 5 **PWIS**：选择 PWM 模块中断源
0：PWM 周期匹配中断 (PWMP_Int)
1：PWM 占空比匹配中断 (PWMD0~2_Int)

Bit 4~3 **ADCHVE~ADCLVE**：A/D 比较结果中断触发条件配置位 (仅用于 SAIN[2:0]=001)
00：ADLVD[11:0] < SADO[11:0] < ADHVD[11:0]
01：SADO[11:0] ≤ ADLVD[11:0]
10：SADO[11:0] ≥ ADHVD[11:0]
11：SADO[11:0] ≤ ADLVD[11:0] 或 SADO[11:0] ≥ ADHVD[11:0]

Bit 2 未定义，读为“0”

Bit 1~0 **PWDIS1~PWDIS0**：PWM 占空比匹配中断源选择位 (PWIS=1)
00：PWM0 占空比匹配中断触发 PWM 中断
01：PWM1 占空比匹配中断触发 PWM 中断
10：PWM2 占空比匹配中断触发 PWM 中断
11：PWM2 占空比匹配中断触发 PWM 中断

- **ADDL 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**：A/D 转换器 DELAY 电路延迟时间控制位 (对系统时钟计数)
延迟时间 = $(1/f_{SYS})\times D[7:0]$
请注意，PWMPF 或 PWMnDF 由 0 变 1 时触发 DELAY 电路开始计数。

A/D 转换器边界寄存器

此单片机包含两对边界寄存器存放预定的上限和下限值用于与存储在 SADOH/SADOL 寄存器中的 A/D 转换结果进行比较。当转换结果达到限定条件时，会触发 A/D 转换比较结果中断。预置的上限和下限边界值需依照 ADRFS 指定的格式分别存储在 ADHVDH/ADHVDL 和 ADLVDH/ADLVDL 寄存器对中。

ADRFS	ADLVDH								ADLVDL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D 下限值数据寄存器

ADRFS	ADHVDH								ADHVDL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D 上限值数据寄存器

A/D 转换器操作

有两种方法可以启动 A/D 转换周期，由 ADSTS 位进行选择。第一种方法是通过 SADC0 寄存器中的 START 启动位启动 A/D 转换。当单片机设定此位从逻辑低到逻辑高，然后再到逻辑低，就会开始一个模数转换周期。第二种方法是由 PWM 中断信号触发 A/D 转换周期。PWM 中断信号可通过 PWIS 位选择来自 PWM 周期匹配中断或 PWM 占空比匹配中断。当选择来自 PWM 占空比匹配中断时，实际的 PWM 占空比匹配中断触发源可通过 ADCR2 寄存器中的 PWDIS1 和 PWDIS0 位选择来自 PWMD0~2_Int 中断信号中的一个。DLSTR 位可使能延迟功能，通过延迟电路可在 PWM 中断信号与实际 A/D 转换开始之间插入一段延迟时间，此延迟时间由 ADDL 寄存器设置。实际延迟时间等于 ADDL 寄存器值乘以系统时钟周期。当电机驱动晶体管瞬间释放大电流时，该延迟时间可减少误采样的发生。应注意，如果 DLSTR 位选择不延迟，则 ADDL 寄存器必须被清除为零，如果 DLSTR 位选择延迟，则必须对 ADDL 寄存器写入非零的数值。

SADC0 寄存器中的 ADBZ 位用于表明模数转换过程是否正在进行。A/D 转换成功启动后，ADBZ 位会被单片机自动置为“1”。在转换周期结束后，ADBZ 位会被单片机自动清零。此外，也会置位中断控制寄存器内相应的 A/D 中断请求标志位，如果中断使能，就会产生对应的内部中断信号。A/D 内部中断信号将引导程序到相应的 A/D 内部中断入口。如果 A/D 内部中断除能，可以让单片机轮询 SADC0 寄存器中的 ADBZ 位，检查此位是否被清除，以作为另一种侦

测 A/D 转换周期结束的方法。

A/D 转换器的时钟源为系统时钟 f_{SYS} 或其分频，而分频系数由 SADC1 寄存器中的 SACKS2~SACKS0 位决定。虽然 A/D 时钟源是由系统时钟 f_{SYS} 和 SACKS2~SACKS0 位决定，但可选择的最大 A/D 时钟源则有一些限制。由于允许的 A/D 时钟周期 t_{ADCK} 的范围为 0.5μs~10μs，所以选择系统时钟速度时就必须小心。例如，如果系统时钟速度为 8MHz 时，SACKS2~SACKS0 位不能设为“000”、“001”或“111”。必须保证设置的 A/D 转换时钟周期不小于时钟周期的最小值或不大于时钟周期的最大值。使用者可以参考下面的表格，被标上星号 * 的数值是不允许的，因为它们超出了 A/D 转换时钟周期规定的范围。

f_{SYS}	A/D 时钟周期 (t_{ADCK})							
	SACKS[2:0] = 000 (f_{SYS})	SACKS[2:0] = 001 ($f_{SYS}/2$)	SACKS[2:0] = 010 ($f_{SYS}/4$)	SACKS[2:0] = 011 ($f_{SYS}/8$)	SACKS[2:0] = 100 ($f_{SYS}/16$)	SACKS[2:0] = 101 ($f_{SYS}/32$)	SACKS[2:0] = 110 ($f_{SYS}/64$)	SACKS[2:0] = 111 ($f_{SYS}/128$)
1MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
12MHz	83ns *	167ns *	333ns *	667ns	1.33μs	2.67μs	5.33μs	10.67μs *
16MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs
20MHz	50ns *	100ns *	200ns *	400ns *	800ns	1.6μs	3.2μs	6.4μs

A/D 转换时钟周期范例

SADC0 寄存器中的 ADCEN 位用于控制 A/D 转换电路电源的开启和关闭。该位必须置高以开启 A/D 转换器电源。当设置 ADCEN 位为高开启 A/D 转换器内部电路时，在 A/D 转换成功开启前需一段延时，如时序图中所示。即使通过相关引脚共用控制位选择无引脚作为 A/D 输入，如果 ADCEN 设为“1”，那么仍然会产生功耗。因此在功耗敏感的应用中，当未使用 A/D 转换器功能时，建议设置 ADCEN 为低以减少功耗。

边界寄存器对 ADHVDH/ADHVDL 和 ADLVDH/ADLVDL 用于存储预设值，该值与寄存器 SADOH/SADOL 中存储的值作比较。ADCLVE 和 ADCHVE 位可用于定义各种比较条件。当转换结果满足限定的条件时，系统会产生中断。这个功能用来确保电机工作在安全限制电流内。

A/D 转换器输入信号

所有的 A/D 模拟输入引脚都与 I/O 口及其它功能共用。使用 PxS0 和 PxS1 寄存器中的相应位，可以将它们设置为 A/D 转换器模拟输入脚或具有其它功能。如果对应的引脚作为 A/D 转换输入，那么它原来的引脚功能将除能。通过这种方式，引脚的功能可由程序来控制，灵活地切换引脚功能。如果将引脚设为 A/D 输入，则通过寄存器编程设置的所有上拉电阻会自动断开。请注意，端口控制寄存器不需要为使能 A/D 输入而先设定为输入模式，当 A/D 输入功能选择位使能 A/D 输入时，端口控制寄存器的状态将被重置。

另外还有一个内部模拟信号可作为 A/D 转换器的模拟输入信号，来自运算放大器输出信号 OPAO 通过设置 SAINS2~SAINS0 位来选择。若 SAINS2~SAINS0 位为“000”或“100~111”，则选择转换外部模拟输入信号，具体通道编号由 SACS3~SACS0 位决定。若选择内部模拟信号时，无论 SACS 位域为何值，外部通道输入将自动断开。此举将预防外部通道输入将与内部模拟信号连接从而导致的不可预期的后果。

SAINS[2:0]	SACS[3:0]	输入信号	描述
000, 100~111	0000~1001	AN0~AN9	外部模拟通道输入
	1010~1111	—	未选择外部通道，输入浮空
001	xxxx	OPAO	AP 引脚输入的信号经过运算放大器输出的信号
010~011	xxxx	AV_{SS}	接地

“x”：无关

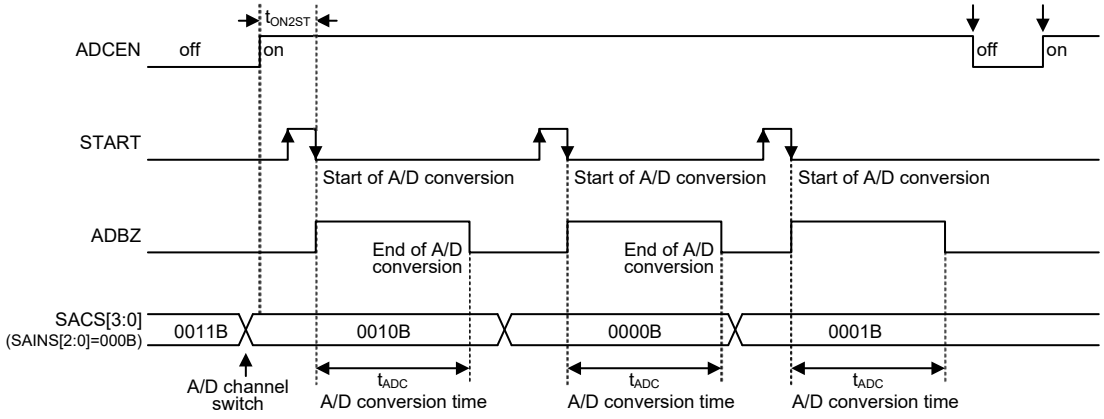
A/D 转换器输入信号选择

A/D 转换率及时序图

一个完整的 A/D 转换包含两部分，数据采样和数据转换。一个完整的 A/D 转换时间，t_{ADC}，一共需要 16 个 A/D 时钟周期。

最大 A/D 转换率 = 1/(A/D 时钟周期 × 16)

下列时序图表示一个外部通道输入信号模数转换过程中不同阶段的图形与时序。由应用程序控制开始 A/D 转换过程后，单片机的内部硬件就会开始进行转换，在这个过程中，程序可以继续其它功能。A/D 转换时间为 $16t_{ADCK}$，t_{ADCK} 为 A/D 时钟周期。



A/D 转换时序图 – 外部通道输入

A/D 转换步骤

下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1
  通过 SADC1 寄存器中的 SACKS2~SACKS0 位，选择所需的 A/D 转换时钟。
- 步骤 2
  将 SADC0 寄存器中的 ADCEN 位置高使能 A/D 转换器。
- 步骤 3
  通过配置 SACS 和 SAINS 位域，选择连接至内部 A/D 转换器的信号。
  若选择外部通道输入，接着执行步骤 4。
  若选择内部模拟信号，接着执行步骤 5。
- 步骤 4
  若配置 SAINS 位域为“000”，“100~111”，将选择外部通道输入信号为 A/D 输入信号。所需的外部通道输入由 SACS 位域确定。当 A/D 输入信号来自

外部通道输入，相应引脚应先通过配置相关的引脚共用功能控制位选择；若 SAINS 位域的值为“001”时选中相关内部模拟信号，则外部通道模拟输入将被自动断开。

- 步骤 5

  通过 ADCR2 寄存器中的 ADSTS 位选择通过哪种方式启动 A/D 转换。

- 步骤 6

  设置 ADRFS 位选择 A/D 转换器输出数据格式。

- 步骤 7

  如果要使用中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是有效的。总中断控制位 EMI 以及 A/D 转换器中断位 ADE 需要预先置位为“1”。

- 步骤 8

  如果在步骤 5 中选择通过 START 触发电路，则需设置 START 位从“0”到“1”再回到“0”，开始模数转换的过程。注意，该位需初始化为 0。如果在步骤 5 中选择通过 PWM 中断延迟触发 A/D 转换周期，则需设置 ADCR2 寄存器中的 DLSTR 位为高以使能 DELAY 开始功能。同时合理设置 ADDL 寄存器以实现时间延迟。

- 步骤 9

  如果 A/D 转换正在进行中，ADBZ 位会被置为逻辑高。A/D 转换完成后，ADBZ 位会被置为逻辑低，并可从 SADOH 和 SADOL 寄存器中读取输出数据。

注：若使用轮询 SADC0 寄存器中 ADBZ 位的状态的方法来检查转换过程是否结束时，则中断使能的步骤可以省略。

编程注意事项

在编程时，如果 A/D 转换器未使用，通过设置 SADC0 寄存器中的 ADCEN 为低高，关闭 A/D 内部电路以减少电源功耗。此时，不考虑输入脚的模拟电压，内部 A/D 转换器电路不产生功耗。如果 A/D 转换器输入脚用作普通 I/O 脚，必须特别注意，输入电压为无效逻辑电平也可能增加功耗。

A/D 转换功能

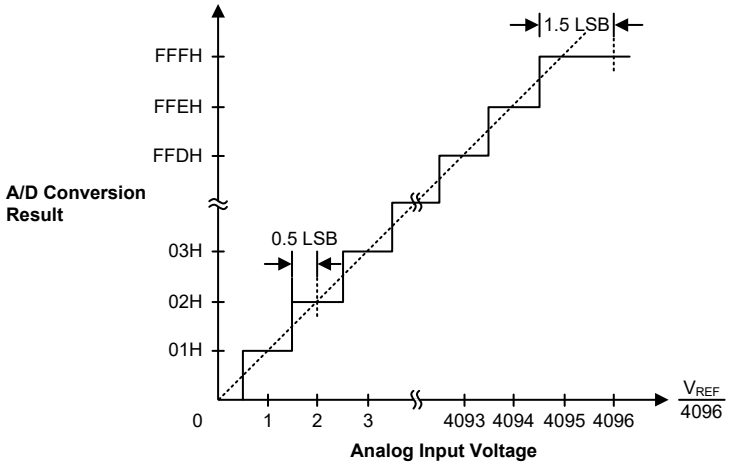
单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于实际输入的 A/D 转换器参考电压 V_{REF} 的电压值，因此每一位可表示 $V_{REF}/4096$ 的模拟输入值。

$$1\ LSB=V_{REF}\div 4096$$

通过下面的等式可估算 A/D 转换器输入电压值：

$$\text{A/D 输入电压} = \text{A/D 数字输出值} \times (V_{REF}\div 4096)$$

下图显示 A/D 转换器模拟输入值和数字输出值之间理想的转换功能。除了数字化数值 0，其后的数字化数值会在精确点之前的 0.5 LSB 处改变，而数字化数值的最大值将在 V_{REF} 之前的 1.5 LSB。



理想的 A/D 转换功能

A/D 转换应用范例

下面两个范例程序用来说明怎样使用 A/D 转换。第一个范例是轮询 SADC0 寄存器中的 ADBZ 位来判断 A/D 转换是否完成；第二个范例则使用中断的方式判断。

范例 1：使用查询 ADBZ 的方式来检测转换结束

```
clr ADE                    ; disable ADC interrupt
mov a,03H                  ; select fSYS/8 as A/D clock and A/D input
mov SADC1,a                ; signal comes from external channel
mov a,02H                  ; setup PAS0 to configure pin AN0
mov PAS0,a
mov a,20H                  ; enable A/D converter and select AN0 as the A/D
                           ; external channel input
mov SADC0,a
:
start_conversion:
clr START                  ; high pulse on start bit to initiate conversion
set START                  ; reset A/D
clr START                  ; start A/D
:
polling_EOC:
sz  ADBZ                   ; poll the SADC0 register ADBZ bit to detect end
                           ; of A/D conversion
jmp polling_EOC            ; continue polling
:
mov a,SADOL                ; read low byte conversion result value
mov SADOL_buffer,a         ; save result to user defined register
mov a,SADOH                ; read high byte conversion result value
mov SADOH_buffer,a         ; save result to user defined register
:
jmp start_conversion       ; start next A/D conversion
```

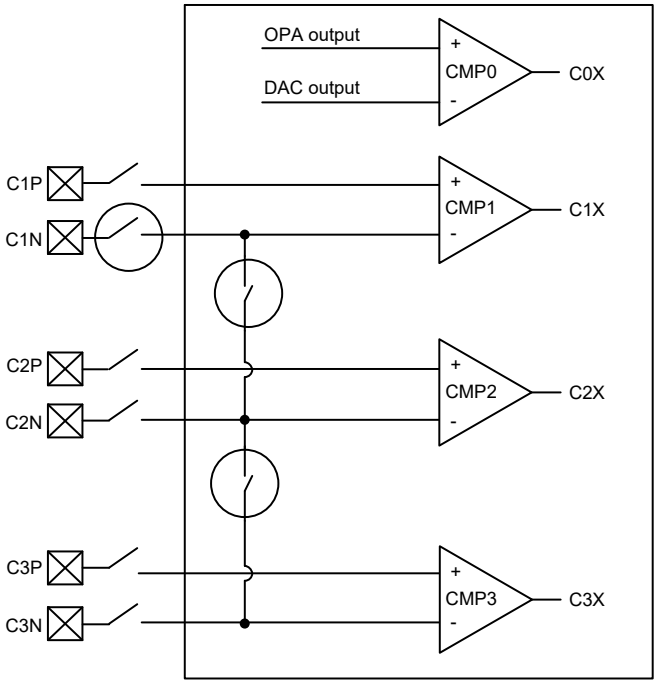
范例 2：使用中断的方式来检测转换结束

```
clr ADE                    ; disable ADC interrupt
mov a,03H                  ; select fSYS/8 as A/D clock and A/D input
mov SADC1,a                ; signal comes from external channel
mov a,02h                  ; setup PACS0 to configure pin AN0
mov PAS0,a
mov a,20h
mov SADC0,a                ; enable A/D converter and select AN0 as the A/D
                           ; external channel input
:
Start_conversion:
clr START                  ; high pulse on START bit to initiate conversion
set START                  ; reset A/D
clr START                  ; start A/D
clr ADF                    ; clear ADC interrupt request flag
set ADE                    ; enable ADC interrupt
set EMI                    ; enable global interrupt
:
:
ADC_ISR:                   ; ADC interrupt service routine
mov acc_stack,a            ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a         ; save STATUS to user defined memory
:
mov a,SADOL                ; read low byte conversion result value
mov SADOL_buffer,a         ; save result to user defined register
mov a,SADOH                ; read high byte conversion result value
mov SADOH_buffer,a         ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a               ; restore STATUS from user defined memory
mov a,acc_stack            ; restore ACC from user defined memory
reti
```

比较器

该单片机中含有四个独立的模拟比较器。它们具有暂停和迟滞等功能，可通过寄存器进行灵活配置。比较器的引脚与普通 I/O 引脚共用，当比较器功能未使用时，此引脚可做普通引脚使用而不浪费 I/O 资源。

由于比较器引脚与其它功能引脚共用，比较器引脚功能需要事先通过相关引脚共用功能选择寄存器进行设置。更多引脚共用功能选择详见引脚共用功能章节。



注：当 PBS0 寄存器中的 PBS03~PBS02 位设置为 10B 时，方框图中圈出来的开关将导通，此时 C1N、C2N 和 C3N 将内部短接，并连接至 CPN 引脚。

比较器方框图

比较器操作

该单片机包含四个比较器功能，用于比较两个模拟电压，基于它们的差值上提供一个输出。当比较器使能时，连接到与比较器输入共用的引脚的上拉电阻将自动失效。当比较器输入接近其切换电压时，由于输入信号上升或下降速度较慢，比较器输出端可能会产生一些伪输出信号。通过选择迟滞功能提供少量正反馈给比较器可使此种情况的发生率进一步降低。理想情况下正负输入信号在同一个电压点时比较器将发生开关动作，但是不可避免的输入失调电压会导致情况不确定。若迟滞功能使能，也可增加切换偏差值。

比较器寄存器

CMPC 控制寄存器用于控制四个比较器的迟滞功能和开 / 关控制。CnHYEN 为比较器 n 迟滞控制位，该位为 1 时，比较器有一定量的迟滞，具体见比较器电气特性表。滞后产生的正反馈将减少比较器门槛附近的伪开关效应的影响。CnEN 为比较器 n 开 / 关控制位，该位为 0 时，比较器关闭，即使其引脚上加有模拟电压也不会产生功耗。对功耗要求严格的应用中，当比较器未使用或单片机进入空闲 / 休眠模式之前，此位应清零。

由于比较器 0 用于过流检测，关于它的更多描述请参考其相关章节。

- **CMPC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C3HYEN	C2HYEN	C1HYEN	C0HYEN	C3EN	C2EN	C1EN	C0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	0	0	0	0

Bit 7 **C3HYEN**：比较器 3 迟滞控制位
0：关闭
1：开启

Bit 6 **C2HYEN**：比较器 2 迟滞控制位
0：关闭
1：开启

Bit 5 **C1HYEN**：比较器 1 迟滞控制位
0：关闭
1：开启

Bit 4 **C0HYEN**：比较器 0 迟滞控制位
0：关闭
1：开启

Bit 3 **C3EN**：比较器 3 开 / 关控制位
0：关闭
1：开启

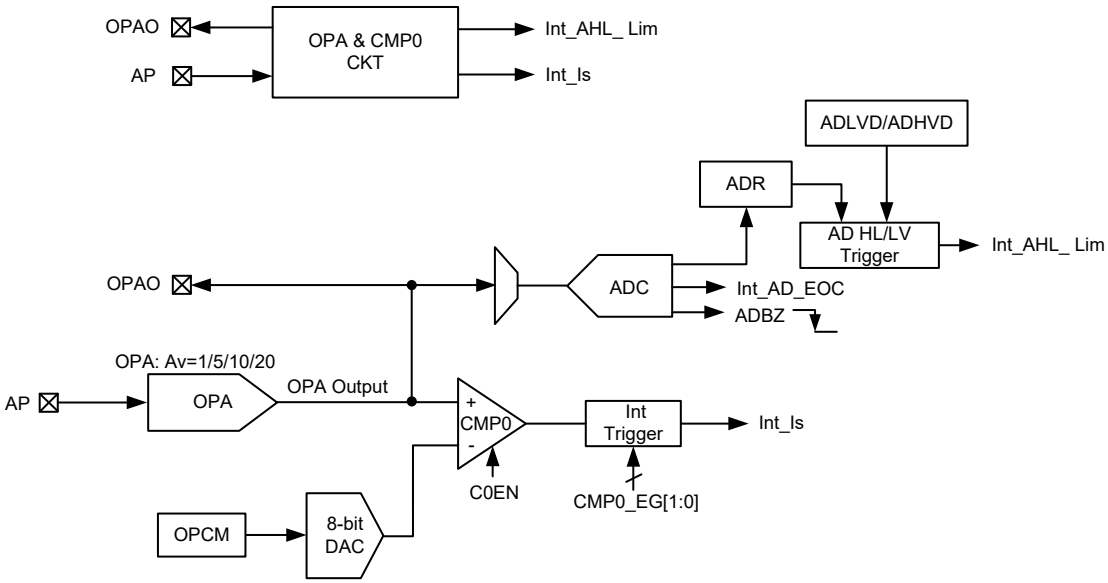
Bit 2 **C2EN**：比较器 2 开 / 关控制位
0：关闭
1：开启

Bit 1 **C1EN**：比较器 1 开 / 关控制位
0：关闭
1：开启

Bit 0 **C0EN**：比较器 0 开 / 关控制位
0：关闭
1：开启

过流检测

该单片机包含一个完全集成的过流检测电路用于保护电机。



过流检测电路方框图

过流检测功能介绍

过流检测功能由一系列电路实现，包括一个放大器 OPA、A/D 转换器、8 位 D/A 转换器和比较器 0。如果检测到过流情况，可立即关闭电机外部驱动电路，以避免电机被烧毁。

两种中断用来检测过流情况：

1. A/D 转换器比较结果中断 – Int_AHL_Lim
2. 比较器 0 中断 – Int_Is

过流检测寄存器

有四个寄存器 OPOMS、OPCM、OPACAL 和 CMPC 用来控制整个过流检测电路的功能和操作。这些 8 位寄存器定义了比较器使能 / 除能，迟滞功能及中断边沿控制、OPA 工作模式选择和 OPA 校准等等。OPCM 寄存器为 8 位 DAC 寄存器，用于和 OPAO 进行比较。CMPC 寄存器内容参考“比较器寄存器”章节。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OPOMS	CMP0_EG1	CMP0_EG0	CMP1P_SEL	—	—	OPAVS2	OPAVS1	OPAVS0
OPCM	D7	D6	D5	D4	D3	D2	D1	D0
OPACAL	ARS	AOFM	—	AOF4	AOF3	AOF2	AOF1	AOF0
CMPC	C3HYEN	C2HYEN	C1HYEN	C0HYEN	C3EN	C2EN	C1EN	C0EN

过流检测寄存器列表

- **OPOMS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CMP0_EG1	CMP0_EG0	CMP1P_SEL	—	—	OPAVS2	OPAVS1	OPAVS0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	1	1	0	—	—	0	1	0

Bit 7~6 **CMP0_EG1~CMP0_EG0**：比较器 0 中断触发有效边沿控制
00：除能比较器 0 和 D/A 转换器
01：上升沿触发
10：下降沿触发
11：双边沿触发
注意，当设置为“11”时，发生边沿跳转时比较器 0 中断可正常发生，但不能作为触发保护电路的信号。

Bit5 **CMP1P_SEL**：比较器 1 正端输入源选择
0：C1P 引脚
1：OPA 输出

Bit 4~3 未定义，读为“0”

Bit 2~0 **OPAVS2~OPAVS0**：OPA 增益选择
000：OPA 除能
001：Av=5
010：Av=10
011：Av=20
100~110：未定义
111：Av=1
注：当使用 OPA 功能时，相关引脚共用控制位应合理设置以使能 AP 引脚功能。

- **OPCM 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**：8-bit D/A 转换器控制码 bit 7 ~ bit 0
将 OPOMS 寄存器的 CMP0_EG1~CMP0_EG0 位设置为“00”外的任意值可使能 D/A 转换器，所以在设置 OPCM 寄存器值之前应先合理设置 CMP0_EG1~CMP0_EG0 位，以确保 D/A 转换器电压成功输出。
DAC $V_{OUT}=(AV_{DD}/256)\times D[7:0]$

- **OPACAL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ARS	AOFM	—	AOF4	AOF3	AOF2	AOF1	AOF0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	1	0	0	0	0

Bit 7 **ARS**：OPA 失调校准时参考电压输入选择
0：OPA 反相输入端
1：OPA 同相输入端

Bit 6 **AOFM**：正常或校准模式选择
0：正常模式
1：失调校准模式

Bit 5 未定义，读为“0”

Bit 4~0 **AOF4~AOF0**：OPA 输入失调电压校准控制码
00000：最小值
10000：中间值
11111：最大值

无刷直流电机控制电路

该章节介绍单片机如何控制无刷直流电机，即 BLDC 电机。高度的功能集成和灵活性为电机提供更大范围的驱动特性。

功能简介

PWM 计数器电路：产生 PWM 信号输出即 PWMO，可线性调整不同占空比，用以调整电机的输出功率进而调整转速，并可线性调整频率以适应电机特性，以提升电机效率或降低电机运作时所产生的共振或噪声。

内部 Mask 电路：Mask 电路决定电机控制时哪些 PWM 调制信号使能或除能。可软件控制选择采用外部预驱的晶体管对的上侧 (GAT/GBT/GCT) 或是下侧 (GAB/GBB/GCB) 来输出 PWM 调整信号。

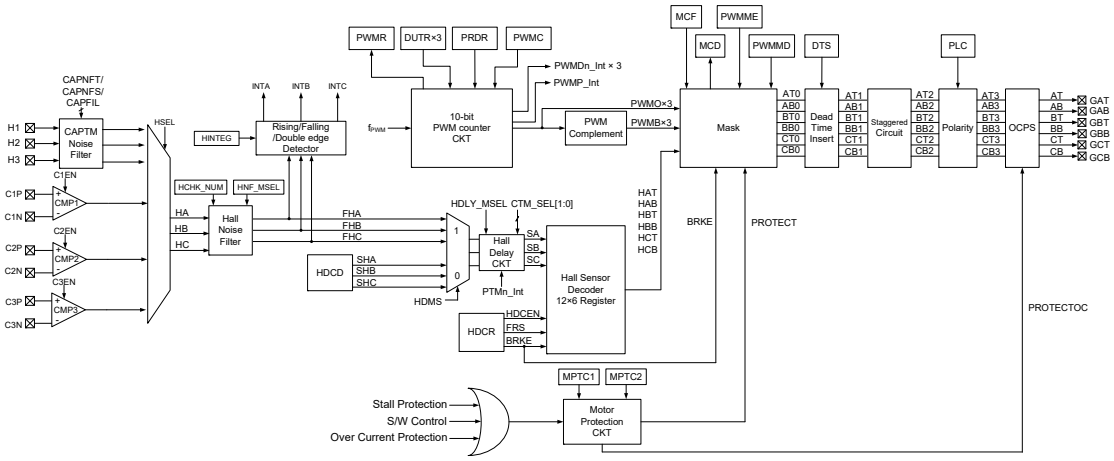
死区时间插入电路：确保外部预驱电路晶体管对在转态时不会瞬间导通 (上下侧 MOS 都开启)，产生短路电流。死区时间由软件控制，控制在 0.3μs~5μs 范围内。

防呆电路：若发生软件错误，或因外力因素如 ESD 问题造成外部预驱晶体管对的上侧和下侧的输出 MOS 同时开启，防呆电路可迫使所有输出都是关闭状态。

极性电路：调整 BLDC 输出控制接口的输出极性，以支持多种外部 MOS 预驱电路组合。

电机保护电路包括多种检测电路用来检测堵转情况，过流情况等等。如果检测到，可即时关闭电机外部驱动电路，以避免电机被烧毁。

霍尔传感器译码电路：支持 6 步骤通信的电机方向控制的方法，利用 12 个寄存器 (每个寄存器 6 个位) 控制电机方向。通过 HDCD/HDCR 寄存器控制电机的转向前、转向后、刹车、空转。霍尔传感器译码电路的输入可以选择来自 HA/HB/HC 或 SHA/SHB/SHC。

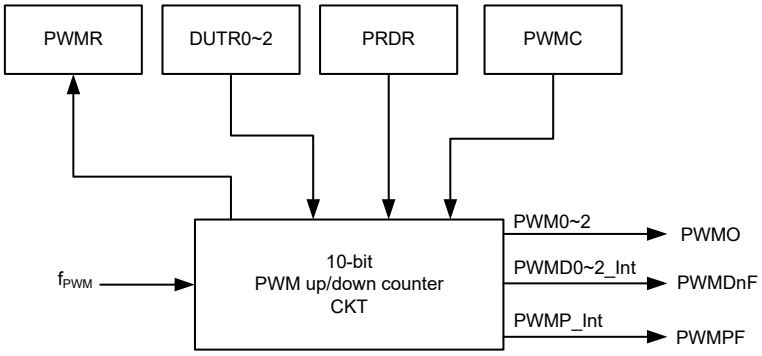


注：GAT、GAB、GBT、GBB、GCT、GCB = PWM0H、PWM0L、PWM1H、PWM1L、PWM2H、PWM2L

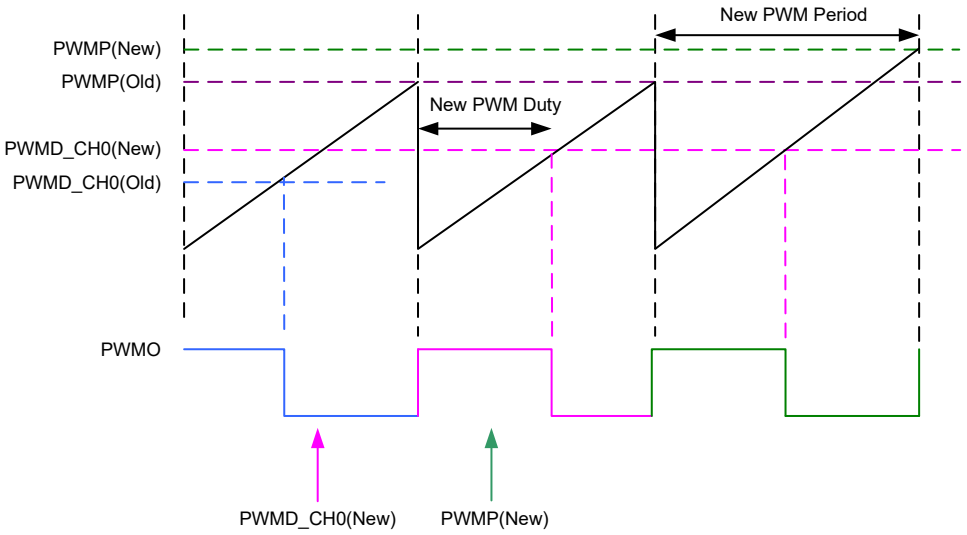
BLDC 电机控制方框图

PWM 计数器控制电路

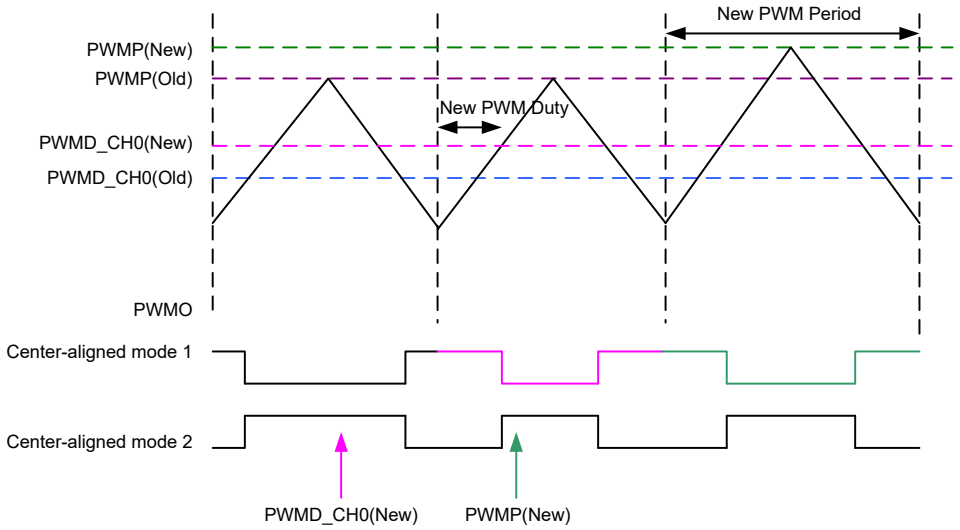
此单片机提供一个 10 位 PWM 发生器。通过给相应的 PWM 寄存器设置特定的 10 位值，PWM 功能可提供占空比和频率可调的波形。



PWM 方框图



PWM 边沿对齐模式时序图



PWM 中心对齐模式时序图

PWM 占空比同步更新模式

在高速的 BLDC 应用中 PWM 中断更新占空比有可能会造成三路的 PWM 占空比无法同步更新，使得 PWM 占空比输出值与预期不相同而造成控制误差，为了改善这个问题 PWM 占空比的更新方式提供了 2 种三路 PWM 同步输出功能。

DUTR0~DUTR2 PWM 占空比相同

通常在方波控制中三路 PWM 占空比会是相同的。此时通过设置 PWMSV 为 1 可以使 DUTR0H 和 DUTR0L 被赋值的同时将占空比值同步更新至 DUTR1H/DUTR1L 和 DUTR2H/DUTR2L。这样即可实现三路 PWM 占空比同步更新并减少 PWM 更新所需指令。

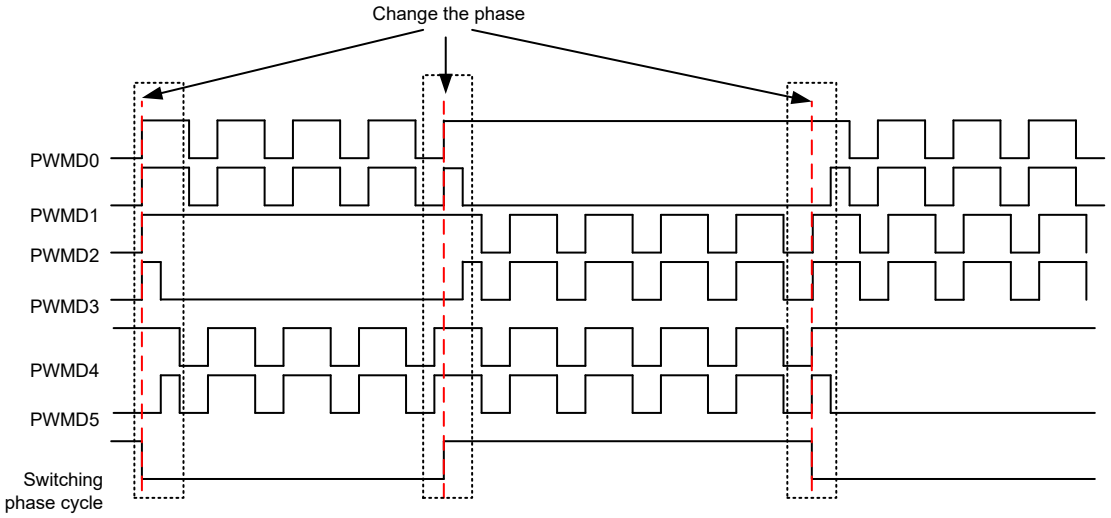
DUTR0~DUTR2 PWM 占空比不同

若三路 PWM 的占空比不同且需要同步更新 PWM 时，可以通过设置 PWMSU 为 1 来开启同步更新三路 PWM 占空比的功能。当三路占空比同步更新请求标志位 PWMSUF 置 1 时，硬件会同时更新 DUTR0、DUTR1 和 DUTR2 的值。当硬件完成更新后会将此标志位自动清零。

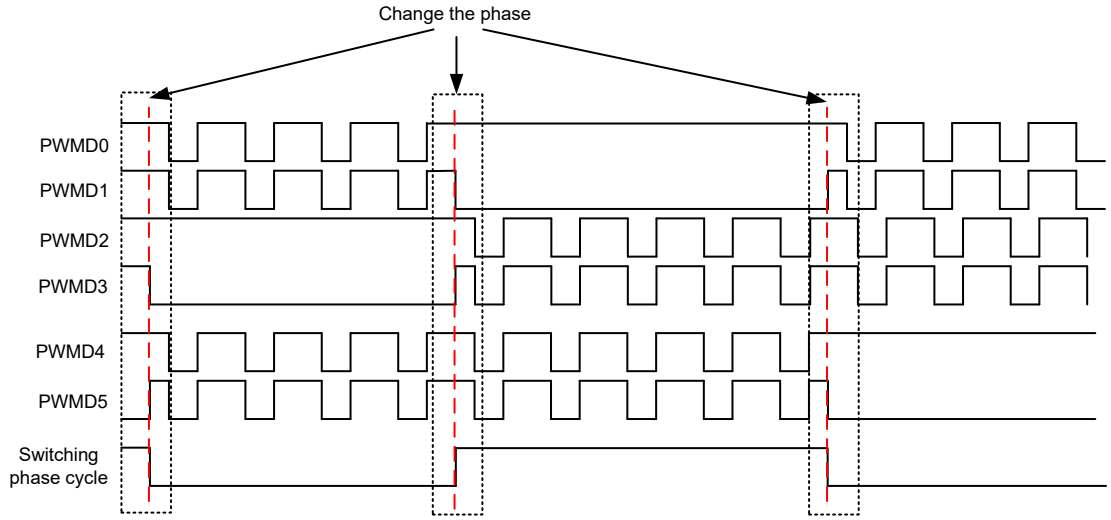
PWM 寄存器介绍

整个 PWM 操作由一系列寄存器控制。寄存器 DUTRnL/DUTRnH 用来改变 PWM 占空比以调整电机的输出功率。寄存器 PRDRL/PRDRH 组成 10 位值用来设置 PWM 周期以调整 PWM 频率。线性调整 PWM 频率，以适应电机的电机特性，降低噪声或共振等。寄存器 PWMRL/PWMRH 用来监控 PWM 计数器的动态。寄存器 PWMC 的 PWMON 位用来控制 10 位 PWM 计数器的开 / 关。寄存器 PWMC 中的 PCKS1~PCKS0 位用来选择 PWM 计数器的时钟源。寄存器 PWMC 中的 PWMMS[1:0] 位域决定 PWM 对齐类型为边沿对齐还是中心对齐。

PWMCS 寄存器用于控制 PWM DUTR0~DUTR2 占空比同步更新及 PWM 在换相时是否能快速更新输出状态。如果换相发生，当 PWMMEDO 位为零时，PWM 会等到计数周期结束后，才对下一个相位做 PWM 输出；此行为在软件 Mask 模式时，只针对 PWMME 寄存器的 PMEn=0 的情况有效，在硬件 Mask 模式时 HDCTn 皆有效。当 PWMMEDO 位为高时，PWM 会在换相后马上对下一个相位做 PWM 输出。时序如下。



换相时序图 – PWMMEDO=0



换相时序图 – PWMMEDO=1

DUTRn 及 PRDR 寄存器为只写寄存器，写入数据到这些寄存器对需遵循以下流程。

- 写数据至 DUTRn 或 PRDR
  - ♦ 步骤 1. 写数据至高字节寄存器 DUTRnH 或 PRDRH
    – 注意，此时数据仅写入 2-bit 缓存器。
  - ♦ 步骤 2. 写数据至低字节寄存器 DUTRnL 或 PRDRL
    – 此时数据直接写入低字节寄存器，同时锁存在 2-bit 缓存器中的数据写入高字节寄存器。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PWMC	PWMMS1	PWMMS0	PCKS1	PCKS0	PWMON	ITCMS1	ITCMS0	PWMLD
PWMCS	—	—	—	—	PWMMEDO	PWMSUF	PWMSU	PWMSV
DUTR0L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR0H	—	—	—	—	—	—	D9	D8
DUTR1L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR1H	—	—	—	—	—	—	D9	D8
DUTR2L	D7	D6	D5	D4	D3	D2	D1	D0
DUTR2H	—	—	—	—	—	—	D9	D8
PRDRL	D7	D6	D5	D4	D3	D2	D1	D0
PRDRH	—	—	—	—	—	—	D9	D8
PWMRL	D7	D6	D5	D4	D3	D2	D1	D0
PWMRH	—	—	—	—	—	—	D9	D8

PWM 寄存器列表

- **PWMC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PWMMS1	PWMMS0	PCKS1	PCKS0	PWMON	ITCMS1	ITCMS0	PWMLD
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 **PWMMS1~PWMMS0**：PWM 对齐模式选择
00：边沿对齐模式
01：边沿对齐模式
10：中心对齐模式 1
11：中心对齐模式 2

Bit 5~4 **PCKS1~PCKS0**：PWM 计数器时钟 f_{PWM} 来源选择
00：f_{SYS}
01：$f_{SYS}/2$
10：$f_{SYS}/4$
11：$f_{SYS}/8$

Bit 3 **PWMON**：PWM 电路开 / 关控制位
0：关闭
1：开启
该位是 PWM 功能总的开 / 关位。该位为高，则 PWM 计数器开启，清零则 PWM 除能。将该位清为零将会使正在计数的计数器停止计数且关闭 PWM 以降低功耗。

Bit 2~1 **ITCMS1~ITCMS0**：PWM 中心对齐模式占空比中断控制位
00：除能中心对齐模式占空比中断
01：向上计数时发生中心对齐模式占空比中断
10：向下计数时发生中心对齐模式占空比中断
11：向上或向下计数时发生中心对齐模式占空比中断

Bit 0 **PWMLD**：PWM PRDR 和 DUTRn (n=0~2) 寄存器更新控制
0：不载入 PRDR 周期值与 DUTRn (n=0~2) 占空比值
1：计数器溢出 (上溢或下溢) 后，载入 PRDR 周期值与 DUTRn (n=0~2) 占空比值

- **PWMCS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PWMMEDO	PWMSUF	PWMSU	PWMSV
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

Bit 3 **PWMMEDO**：PWM 换相时快速更新输出状态控制位
0：除能
1：使能

Bit 2 **PWMSUF**：PWM DUTR0~2 占空比同步更新请求标志位 (PWMSU=1 时有效)
0：无请求
1：请求 DUTR0~2 占空比同步更新
设置此位为 1 将请求 DUTR0~2 同步更新。当同步更新完成后，此位由硬件自动清零。

Bit 1 **PWMSU**：PWM DUTR0~2 占空比同步更新控制 (PWMLD=1 时有效)
0：除能
1：使能

Bit 0 **PWMSV**：PWM 占空比写入 DUTR0 时同步至 DUTR1 和 DUTR2 (PWMLD=1 时有效)
0：除能，DUTR0~2 分开独立设置
1：使能，PWM 占空比写入 DUTR0 时也同时写入 DUTR1 和 DUTR2

- **DUTRnL 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	W	W	W	W	W	W	W	W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：10-bit PWMn 占空比低字节寄存器 bit 7 ~ bit 0
10-bit DUTRn 寄存器 bit 7 ~ bit 0

- **DUTRnH 寄存器 (n=0~2)**

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

Bit 7~2 未定义，读为“0”

Bit 1~0 **D9~D8**：10-bit PWMn 占空比高字节寄存器 bit 1 ~ bit 0
10-bit DUTRn 寄存器 bit 9 ~ bit 8

- **PRDRL 寄存器**

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

Bit 7~0 **D7~D0**：10-bit PWM 周期低字节寄存器 bit 7 ~ bit 0
10-bit PRDR 寄存器 bit 7 ~ bit 0

- **PRDRH 寄存器**

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

Bit 7~2　未定义，读为“0”

Bit 1~0　**D9~D8**：10-bit PWM 周期高字节寄存器 bit 1 ~ bit 0
10-bit PRDR 寄存器 bit 9 ~ bit 8

- **PWMRL 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**：10-bit PWM 计数器低字节寄存器 bit 7 ~ bit 0
10-bit PWM 计数器 bit 7 ~ bit 0

- **PWMRH 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**：10-bit PWM 计数器高字节寄存器 bit 1 ~ bit 0
10-bit PWM 计数器 bit 9 ~ bit 8

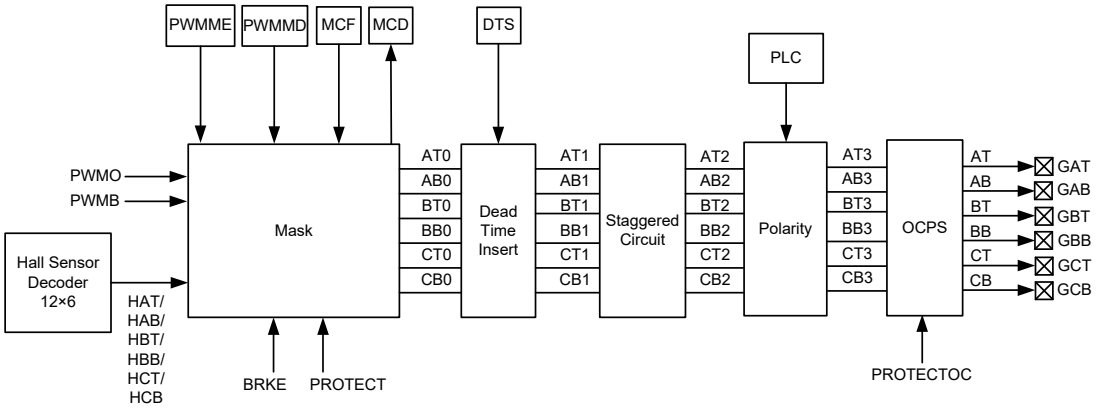
边沿对齐模式：PWM 周期 = (PRDR+1)/f_{PWM} (PRDR 不能设置为 000H)
PWM 占空比 = DUTRn/f_{PWM}

中心对齐模式：PWM 周期 = (PRDR×2)/f_{PWM} (PRDR 不能设置为 000H)
PWM 占空比 = (DUTRn×2-1)/f_{PWM}

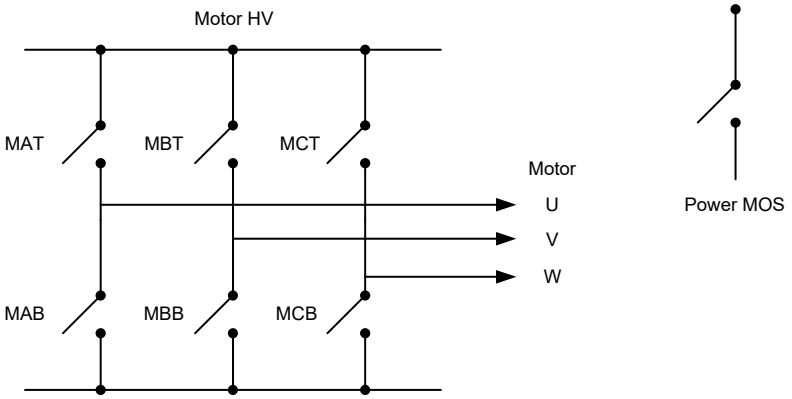
Mask 功能

该单片机具有电机控制 Mask 功能用来增加电机控制灵活性。

内部 Mask 电路有三种工作模式：正常模式，刹车模式，电机保护模式。正常模式包含两个子模式：硬件 Mask 模式和软件 Mask 模式。



Mask 功能方框图



Mask 开关电路

正常模式

正常模式下，通过寄存器 MCF 的 PWMS 和 MPWE 位决定电机调速的控制方式。

- 当 PWMS=0 时，PWM 输出选择晶体管对的下侧 (GAB/GBB/GCB)。
- 当 PWMS=1 时，PWM 输出选择晶体管对的上侧 (GAT/GBT/GCT)。
- 当 MPWE=0 时，PWM 输出除能 (AT0/BT0/CT0/AB0/BB0/CB0 全通)。
- 当 MPWE=1 时，PWM 输出使能 (AT0/BT0/CT0/AB0/BB0/CB0 可输出 PWM 调速)。
- 当 MPWMS=0 时，PWM 互补式输出。
- 当 MPWMS=1 时，PWM 非互补式输出。
- 当 MSKMS=0 时，选择硬件 Mask 模式。
- 当 MSKMS=1 时，选择软件 Mask 模式。

- **MCF 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MSKMS	—	—	—	MPWMS	MPWE	FMOS	PWMS
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	1	0	0

Bit 7 **MSKMS**：Mask 模式选择位
0：硬件 Mask 模式
1：软件 Mask 模式

Bit 6~4 未定义，读为“0”

Bit 3 **MPWMS**：硬件 Mask 模式 PWM 输出选择位
0：互补式输出
1：非互补式输出
此位的选择在软件 Mask 模式下无效，软件 Mask 模式下的 PWM 输出取决于 PWMME 和 PWMMD 寄存器设定。

Bit 2 **MPWE**：PWM 输出控制
0：PWM 输出除能 (AT0/BT0/CT0/AB0/BB0/CB0 不输出 PWM)
1：PWM 输出使能

Bit 1 **FMOS**：PROTECT 被触发时 Mask 输出控制
0：AT0/BT0/CT0=0，AB0/BB0/CB0=0
1：AT0/BT0/CT0=0，AB0/BB0/CB0=1

Bit 0 **PWMS**：硬件 Mask 模式上侧 / 下侧输出 PWM 选择
0：下侧输出 PWM
1：上侧输出 PWM

- **MCD 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	GAT	GAB	GBT	FHC	FHB	FHA
R/W	—	—	R	R	R	R	R	R
POR	—	—	0	0	0	x	x	x

“x”：未知

Bit 7~6 未定义，读为“0”

Bit 5~3 **GAT/GAB/GBT**：预驱输出

Bit 2~0 **FHC/FHB/FHA**：HC/HB/HA 滤波输出
这些信号表示 HC/HB/HA 经过霍尔噪声滤波器滤波后的信号。

硬件 Mask 模式

互补式控制：MPWMS=0

	HAT	HAB	AT0	AB0		HAT	HAB	AT0	AB0
PWMS=0	0	0	0	0	**PWMS=1**	0	0	0	0
	0	1	PWMB	PWMO		0	1	0	1
	1	0	1	0		1	0	PWMO	PWMB
	1	1	0	0		1	1	0	0

	HBT	HBB	BT0	BB0		HBT	HBB	BT0	BB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	PWMB	PWMO	**PWMS=1**	0	1	0	1
	1	0	1	0		1	0	PWMO	PWMB
	1	1	0	0		1	1	0	0

	HCT	HCB	CT0	CB0		HCT	HCB	CT0	CB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	PWMB	PWMO	**PWMS=1**	0	1	0	1
	1	0	1	0		1	0	PWMO	PWMB
	1	1	0	0		1	1	0	0

非互补式控制：MPWMS=1

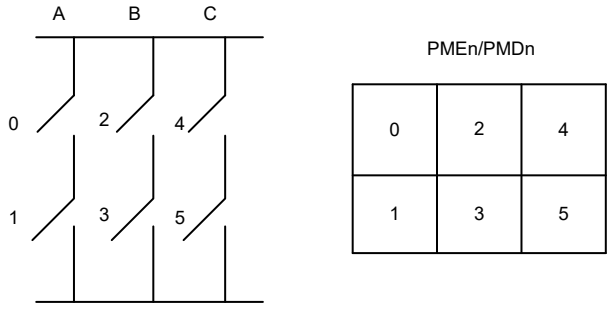
	HAT	HAB	AT0	AB0		HAT	HAB	AT0	AB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	0	PWMO	**PWMS=1**	0	1	0	1
	1	0	1	0		1	0	PWMO	0
	1	1	0	0		1	1	0	0

	HBT	HBB	BT0	BB0		HBT	HBB	BT0	BB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	0	PWMO	**PWMS=1**	0	1	0	1
	1	0	1	0		1	0	PWMO	0
	1	1	0	0		1	1	0	0

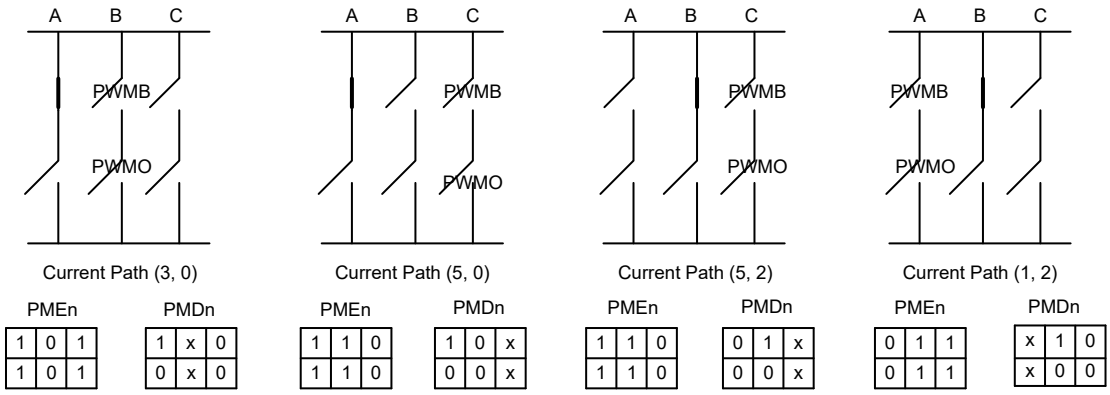
	HCT	HCB	CT0	CB0		HCT	HCB	CT0	CB0
	0	0	0	0		0	0	0	0
PWMS=0	0	1	0	PWMO	**PWMS=1**	0	1	0	1
	1	0	1	0		1	0	PWMO	0
	1	1	0	0		1	1	0	0

软件 Mask 模式

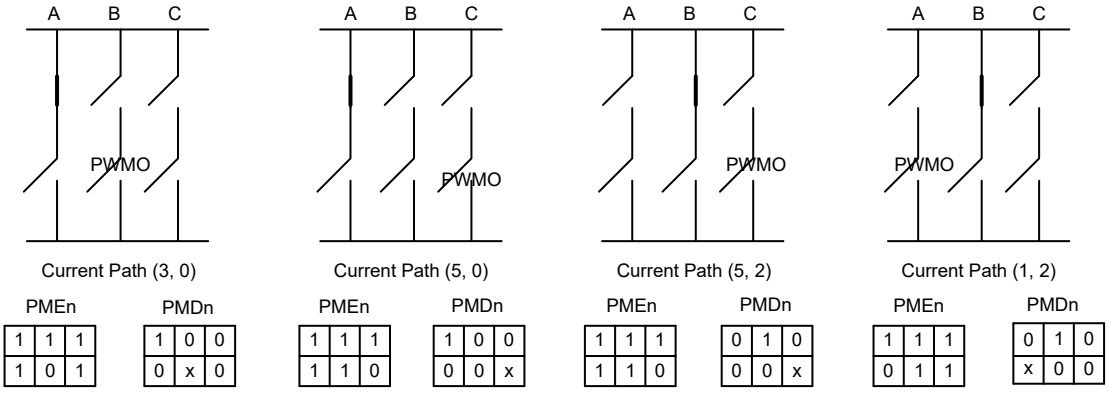
寄存器 PWMME 和 PWMMD 用来控制软件 Mask 电路。寄存器 PWMME 用来控制 PWM 信号是否被 Mask，寄存器 PWMMD 用来决定 MOS 预驱电路开或关。



三相反相器符号及控制位



Mask 互补模式示例



单独 Mask 模式示例

注：1. 若 Mask 使能，当 PWMxH 和 PWMxL 同时 Mask 时，PMD0 和 PMD1，PMD2 和 PMD3，PMD4 和 PMD5，不可同时设置“1”。若是同时为“1”，则会强制其都输出都为“0”。
2. 若 PWM 和互补式 PWM 同时使能，则 PWMxH 和 PWMxL 其中一个为 PWM 输出，另外一个不可 Mask 为“1”，否则硬件自动将输出设为“0”。
3. 若 GxT 和 GxB (x 为 A，B 或 C) 引脚被设置为 I/O 功能，则 PWM Mask 功能将无效。

- **PWMME 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PME5	PME4	PME3	PME2	PME1	PME0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6　未定义，读为“0”

Bit 5~0　**PMEn**：PWM Mask 使能位 (n=0~5)
0：除能，PWM 发生器信号输出到下一级
1：使能，PWM 发生器信号被 Mask
设置此位为高，PWM 信号将被 Mask，相应通道输出由 PMDn 指定的逻辑电平。

- **PWMMD 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PMD5	PMD4	PMD3	PMD2	PMD1	PMD0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6　未定义，读为“0”

Bit 5~0　**PMDn**：PWM Mask 数据位 (n=0~5)
0：输出逻辑低
1：输出逻辑高
当对应的 PMEn=1，该数据位才可用于控制 PWMn 输出状态。

刹车模式

刹车模式有最高优先级。当刹车模式启动时，外部预驱晶体管对上侧关闭，下侧开启。刹车模式下外部预驱晶体管对状态的真值表如下所示。

BRKE=1	AT0	BT0	CT0	AB0	BB0	CB0
	0	0	0	1	1	1

电机保护模式

当 PROTECT 位置高，电机保护模式启动时，外部预驱晶体管对可选择刹车 (上侧关闭，下侧开启) 或是空转 (上侧和下侧都关闭)。保护解码表如下所示。

PROTECT=1	GAT	GBT	GCT	GAB	GBB	GCB
FMOS=0	0	0	0	0	0	0
FMOS=1	0	0	0	1	1	1

其它功能

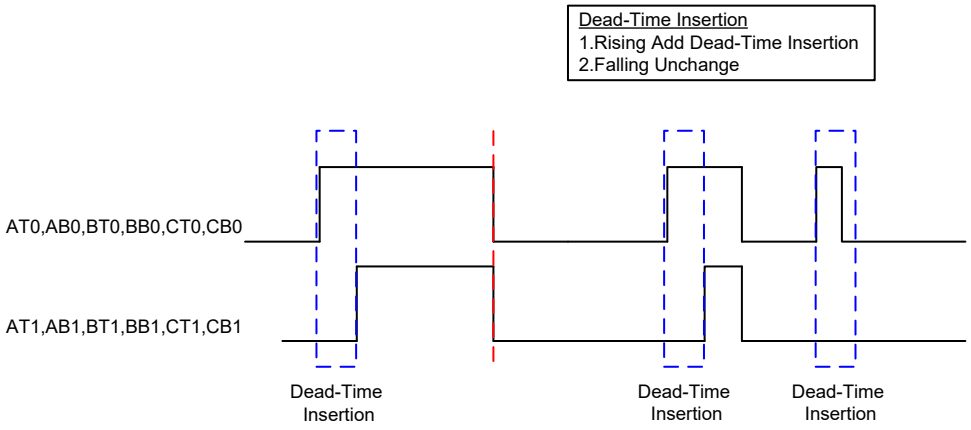
一些其它功能增加了电机控制驱动信号的灵活性。它们包括死区时间功能，防呆功能和极性控制功能。

死区时间插入功能

死区时间功能确保外部预驱电路晶体管对在转态时不会瞬间导通 (上下侧 MOS 都开启)，以防产生短路电流。死区时间可通过应用程序控制在 0.3μs~5μs 左右。

- 当 AT0/AB0/BT0/BB0/CT0/CB0 输出上升沿时，可插入死区时间。
- 当 AT0/AB0/BT0/BB0/CT0/CB0 输出下降沿时，输出不变。

死区时间插入电路只在电机控制时使用，死区时间功能由 DTS 寄存器控制。



死区时间插入时序图

- **DTS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	DTCKS1	DTCKS0	DTE	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 7~6 **DTCKS1~DTCKS0**：死区时间时钟源选择
00：$f_{DT}=f_{SYS}$
01：$f_{DT}=f_{SYS}/2$
10：$f_{DT}=f_{SYS}/4$
11：$f_{DT}=f_{SYS}/8$

Bit 5 **DTE**：死区时间插入控制
0：除能
1：使能

Bit 4~0 **D4~D0**：死区时间寄存器 bit 4 ~ bit 0
5-bit 死区时间计数器。
死区时间 = (DTS[4:0]+1)/f_{DT}

防呆功能

若发生软件错误，或是外界因素如 ESD，造成外部预驱晶体管对的上侧与下侧的输出都是 MOS 开启的状态，防呆电路强迫输出都是 MOS 关闭。

AT1	AB1	AT2	AB2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

注：BLDC 电机控制电路中都默认把外部预驱晶体管对设计为 N 型晶体管对。以上表格中值为 1 代表晶体管开启，0 代表晶体管关闭。

极性控制功能

该功能用来设置外部预驱晶体管开 / 关极性状态。寄存器 PLC 用于整个功能控制。

- **PLC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PCBC	PCTC	PBBC	PBTC	PABC	PATC
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6　未定义，读为“0”

Bit 5　**PCBC**：GCB 输出极性控制
0：同相输出
1：反相输出

Bit 4　**PCTC**：GCT 输出极性控制
0：同相输出
1：反相输出

Bit 3　**PBBC**：GBB 输出极性控制
0：同相输出
1：反相输出

Bit 2　**PBTC**：GBT 输出极性控制
0：同相输出
1：反相输出

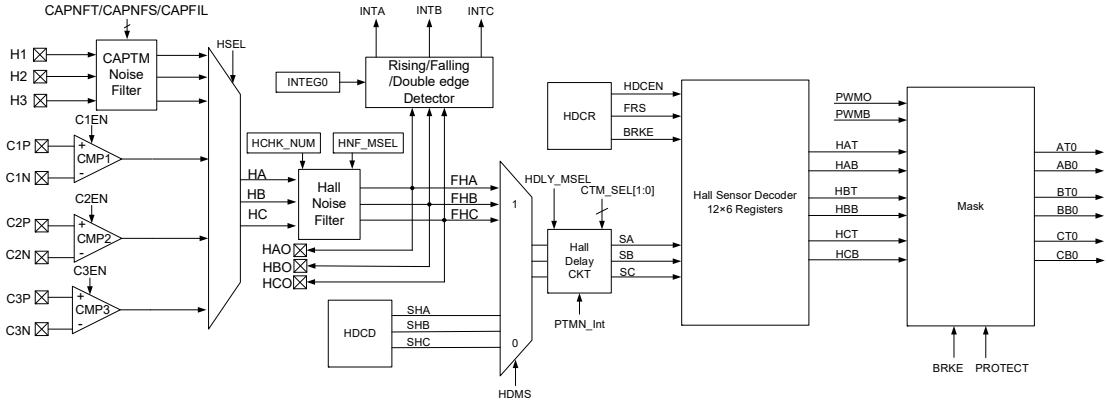
Bit 1　**PABC**：GAB 输出极性控制
0：同相输出
1：反相输出

Bit 0　**PATC**：GAT 输出极性控制
0：同相输出
1：反相输出

应注意，默认输出引脚 GAT/GAB/GBT/GBB/GCT/GCB 状态为高阻抗。

霍尔传感器译码电路

该单片机包含一个完全集成的霍尔传感器译码电路，连接到霍尔传感器用于 BLDC 电机的方向和转速控制。

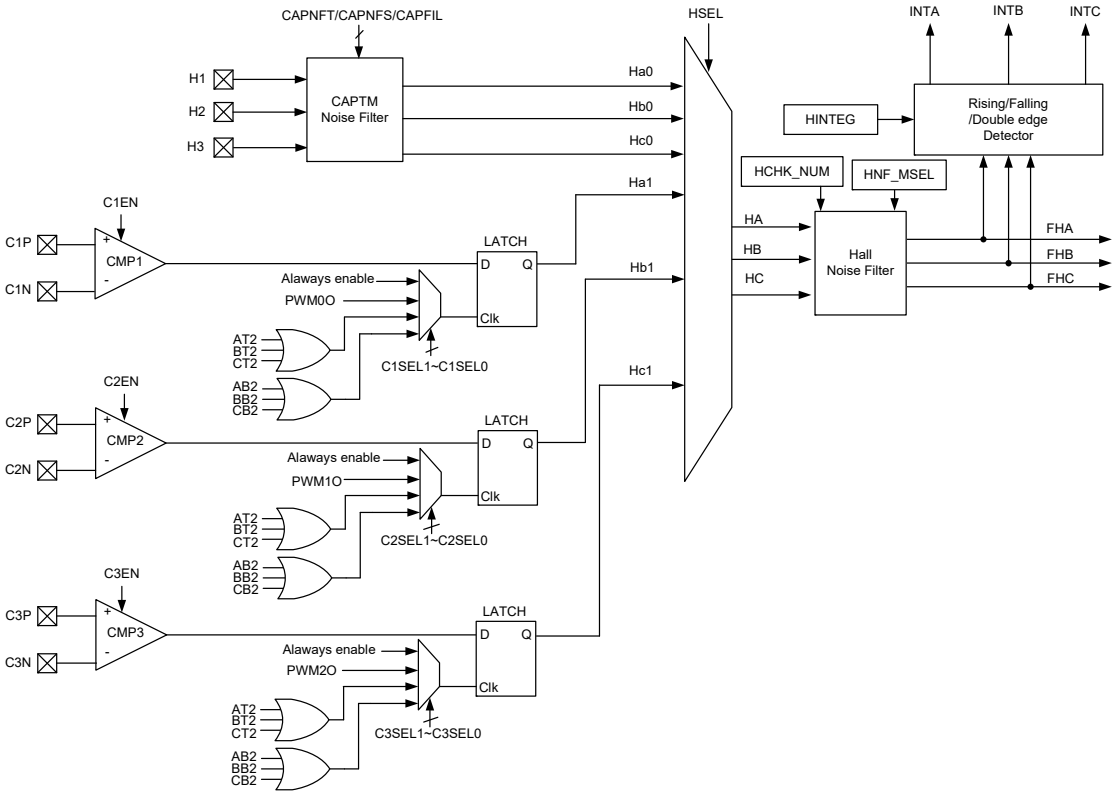


霍尔传感器译码电路方框图

通过设置 HDMS 位为高来选择实际的霍尔传感器译码电路信号输入。如果 HDMS 位为 0，则 HDCD 寄存器中设定的 SHA/SHB/SHC 取代实际的霍尔传感器信号输入。如果 HDMS 的位高，则 HA/HB/HC 信号源由 HINTEG 寄存器中的 HSEL 位选择。

霍尔传感器噪声滤波器

该单片机内建霍尔噪声滤波器以滤除电机驱动器因转换电流过大产生的噪声。该噪声干扰可能影响霍尔传感器输入 (HA/HB/HC)，造成霍尔传感器解码输出有误动作。



注：关于 CAPTM 噪声滤波器的详细描述请参阅捕捉定时器模块章节。

霍尔传感器噪声滤波器方框图

一些寄存器用来控制噪声滤波器。寄存器 HNF_MSEL 中的 HNF_EN 位用来使能 / 除能噪声滤波器。

HNF_EN 位	状态
0	噪声滤波器关闭 – HA/HB/HC 绕过噪声滤波器
1	噪声滤波器开启

霍尔传感器噪声滤波器使能

通过 HFR_SEL[2:0] 位设置霍尔传感器噪声滤波器的采样频率。

HCHK_NUM 寄存器中的 HCK_N[4:0] 位用来设置霍尔传感器输入比较次数。

HCK_N[4:0] × 采样间距 = 抗噪声能力 = 霍尔延迟时间

要注意的是，霍尔延迟时间越长，转子速度反馈失真越严重。

霍尔传感器延迟功能

霍尔传感器电路内建霍尔延迟电路以支持霍尔相位超前 / 相位落后功能。下列步骤显示了该功能如何运作，这些步骤必须在开启霍尔译码器使能前先执行。

- 步骤 1
  设定霍尔译码表，以决定相位超前或相位落后。
- 步骤 2
  通过 CTM_SEL1~CTM_SEL0 位选择哪一个 PTM 来作延迟时间，并将其设置成比较匹配输出模式，以决定延迟时间。
- 步骤 3
  通过 HDLY_MSEL 位选择霍尔延迟电路工作模式。HDLY_MSEL 位的默认值为 0，霍尔延迟电路除能。如果 HDLY_MSEL 位置为高，霍尔延迟电路使能。
- 步骤 4
  通过 HDCEN 位开启霍尔译码器。

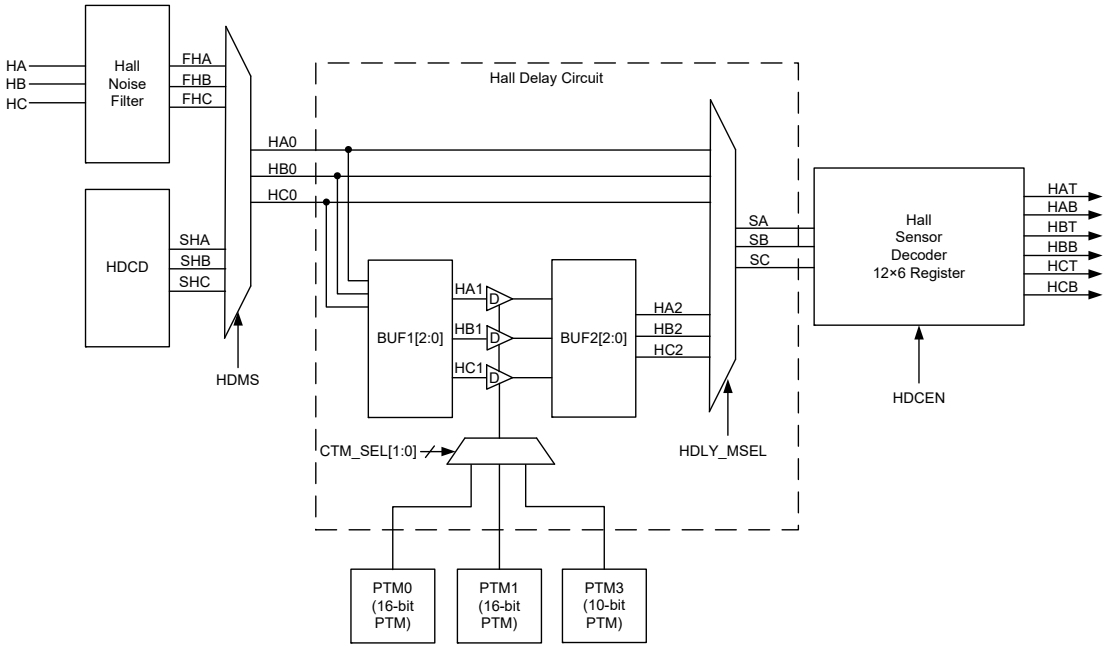
关于 HDLY_MSEL 位必须注意以下几点：

1. 当 HDLY_MSEL 位为 0 时，BUF1[2:0] 和 BUF2[2:0] 被清零。
2. 当 HDLY_MSEL 位为 0 时，PTM0、PTM1 和 PTM3 维持原本的 TM 功能。
3. 当 HDLY_MSEL 位为 1 时，被选中用于延迟功能的 PTM 将专门用于霍尔延迟电路计时。除了 PTnON 位将由硬件来自动控制外，原 PTM 功能仍然有效。在开启延迟功能前，用户需适当配置 PTM 功能。

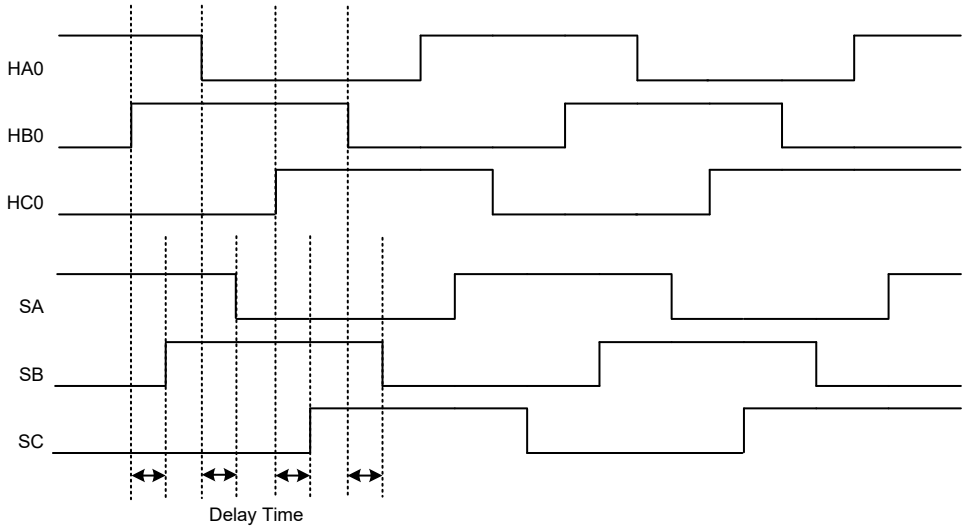
在使能延迟功能前，设置所选 PTM 功能时需注意以下几点：

1. 保持 PTnON=0 和 PTnPAU=0。
2. 将 PTM 设置为比较匹配输出模式。
3. 设置 PTnCCLR=1，即当比较器 A 比较匹配时清除 PTM 计数器。
4. 依需要的延迟时间合理设置 PTMn CCRA 值和选择计数器时钟。

当 HDLY_MSEL 位从低变为高使能延迟电路后，延迟时间不得大于霍尔输入一步的时间 (一个霍尔周期共有 6 步)，否则输出无法预期，造成控制失步。



霍尔延迟功能方框图



延迟功能时序

电机控制驱动信号

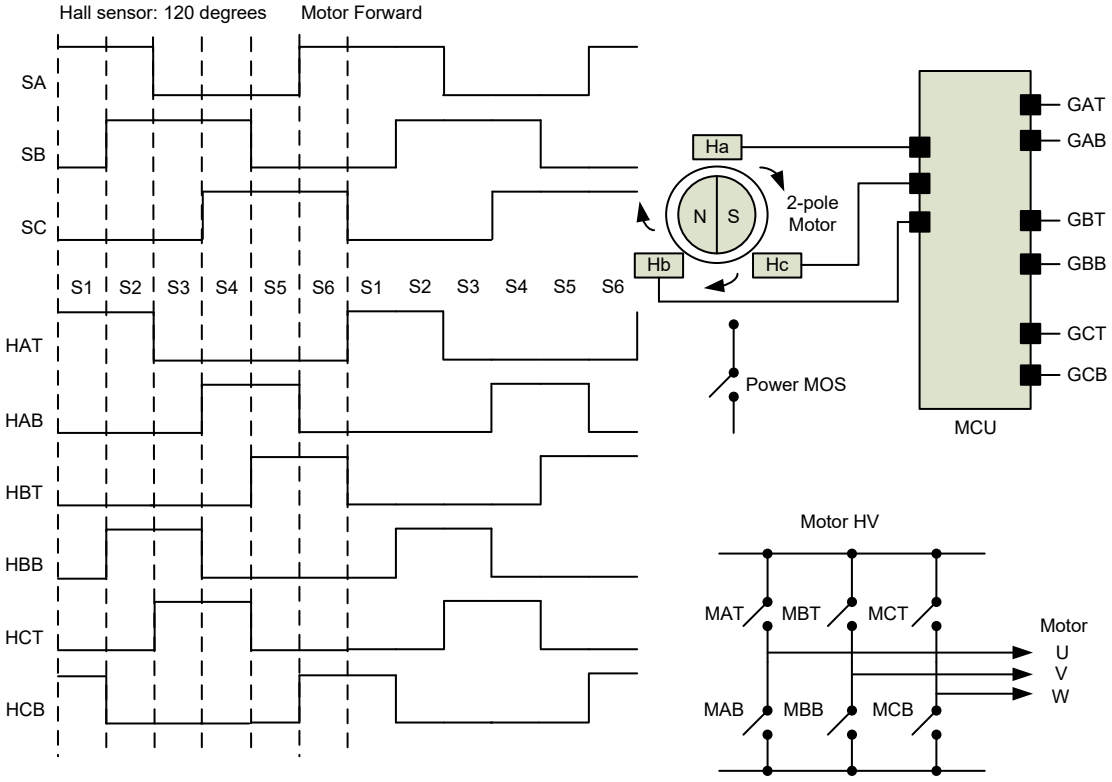
通过 HDCR 和 HDCD 寄存器以及一系列 HDCT 寄存器 HDCT0~HDCT11 来控制 BLDC 电机的方向。当使用霍尔传感器译码功能时，分别通过 HDCR 寄存器中的 FRS 位和 BRKE 位决定电机的方向或者是否刹车。HDCT0~HDCT5 为电机向前转输出状态控制；HDCT6~HDCT11 为电机向后转输出状态控制。

霍尔传感器解码真值表如下。

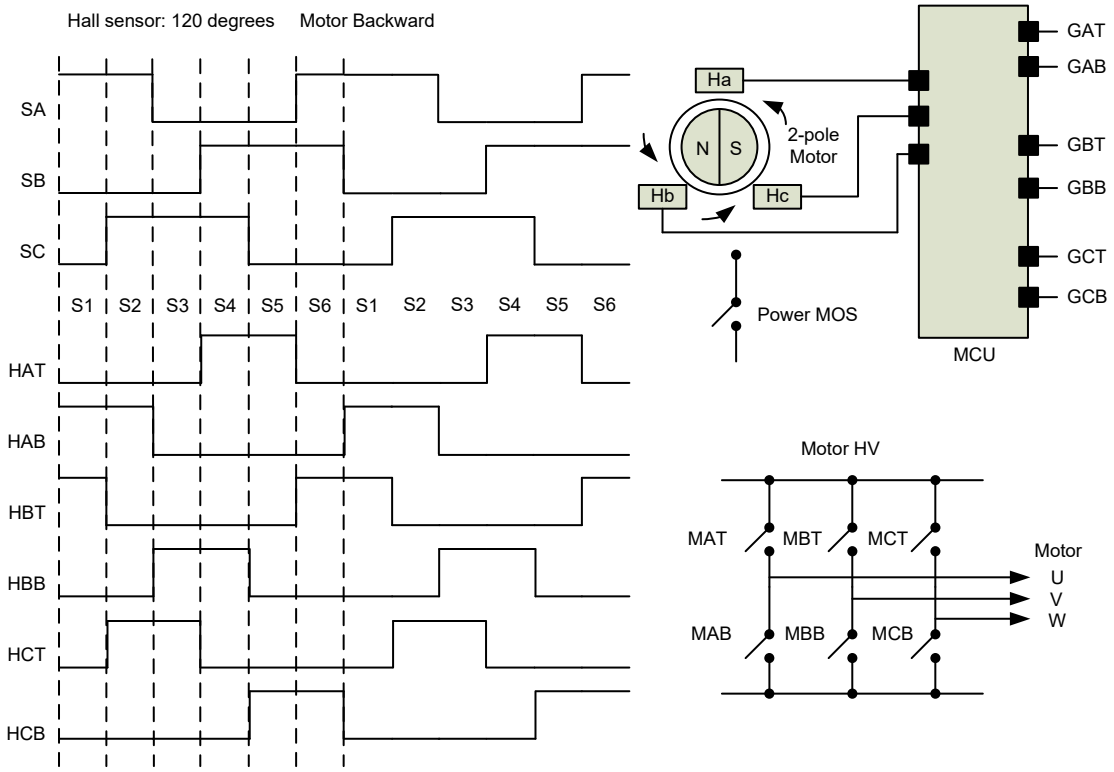
	60 度			120 度			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	**SB**	**SC**	**SA**	**SB**	**SC**	**HAT**	**HAB**	**HBT**	**HBB**	**HCT**	**HCB**
向前转 (HDCEN=1，FRS=0，BRKE=0)	1	0	0	1	0	0	HDCT0[5:0]					
	1	1	0	1	1	0	HDCT1[5:0]					
	1	1	1	0	1	0	HDCT2[5:0]					
	0	1	1	0	1	1	HDCT3[5:0]					
	0	0	1	0	0	1	HDCT4[5:0]					
	0	0	0	1	0	1	HDCT5[5:0]					
	60 度			120 度			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	**SB**	**SC**	**SA**	**SB**	**SC**	**HAT**	**HAB**	**HBT**	**HBB**	**HCT**	**HCB**
向后转 (HDCEN=1，FRS=1，BRKE=0)	1	0	0	1	0	0	HDCT6[5:0]					
	1	1	0	1	1	0	HDCT7[5:0]					
	1	1	1	0	1	0	HDCT8[5:0]					
	0	1	1	0	1	1	HDCT9[5:0]					
	0	0	1	0	0	1	HDCT10[5:0]					
	0	0	0	1	0	1	HDCT11[5:0]					
刹车 (BRKE=1，HDCEN=x，FRS=x)	60 度			120 度			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	**SB**	**SC**	**SA**	**SB**	**SC**	**HAT**	**HAB**	**HBT**	**HBB**	**HCT**	**HCB**
	V	V	V	V	V	V	0	1	0	1	0	1
霍尔译码器除能 (HDCEN=0) 空转	60 度			120 度			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	V	V	V	V	V	V	0	0	0	0	0	0

霍尔译码器出错 (HDCEN=x)	60 度			120 度			Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	SA	SB	SC	SA	SB	SC	HAT	HAB	HBT	HBB	HCT	HCB
	1	0	1	1	1	1	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0

真值表中的数据与实际电机驱动信号的关系如下面时序图所示。以典型 6 步骤通信为例，下图是电机向前转 / 电机向后转时序图。



电机驱动信号时序图 – 向前转



电机驱动信号时序图 – 向后转

霍尔传感器译码相关寄存器介绍

寄存器 HDCR 为霍尔传感器解码控制寄存器；寄存器 HDCD 为霍尔传感器译码输入数据寄存器；寄存器 HDCT0~HDCT11 为霍尔传感器译码表。寄存器 HCHK_NUM 为霍尔噪声滤波器检查次数寄存器；寄存器 HNF_MSEL 为霍尔噪声滤波器模式选择寄存器。寄存器 HINTEG 为霍尔传感器输入源选择和中断边沿控制寄存器。

- **HINTEG 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	HSEL	INTCS1	INTCS0	INTBS1	INTBS0	INTAS1	INTAS0
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　**HSEL**：HA/HB/HC 来源选择
0：H1/H2/H3
1：CMP1/CMP2/CMP3 输出

Bit 5~4　**INTCS1~INTCS0**：FHC 触发 INTC 中断之边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双边沿触发

Bit 3~2 **INTBS1~INTBS0**：FHB 触发 INTB 中断之边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双边沿触发

Bit 1~0 **INTAS1~INTAS0**：FHA 触发 INTA 中断之边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双边沿触发

- **HDCR 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTM_SEL1	CTM_SEL0	HDLY_MSEL	HALS	HDMS	BRKE	FRS	HDCEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7~6 **CTM_SEL1~CTM_SEL0**：霍尔延迟电路 TM 选择
00：PTM0 (16-bit PTM)
01：PTM1 (16-bit PTM)
10：PTM3 (10-bit PTM)
11：未使用

Bit 5 **HDLY_MSEL**：霍尔延迟电路选择
0：选择原始路径 (绕过延迟电路)
1：选择霍尔延迟电路

Bit 4 **HALS**：霍尔传感器译码角度配置选择
0：霍尔传感器 60 度配置
1：霍尔传感器 120 度配置

Bit 3 **HDMS**：霍尔传感器译码模式选择
0：软件模式 (位于 HDCD 寄存器的 SHA/SHB/SHC)
1：霍尔传感器模式 (噪声滤波器输出 FHA/FHB/FHC)

Bit 2 **BRKE**：电机刹车控制
0：AT/BT/CT/AB/BB/CB=V
1：AT/BT/CT=0，AB/BB/CB=1

Bit 1 **FRS**：电机向前 / 向后转选择
0：向前转
1：向后转

Bit 0 **HDCEN**：霍尔传感器译码器使能
0：除能
1：使能

- **HDCD 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	SHC	SHB	SHA
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”

Bit 2 **SHC**：软件 Hall C

Bit 1 **SHB**：软件 Hall B

Bit 0 **SHA**：软件 Hall A

- **HDCTn 寄存器 (n=0~11)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	HATDn	HABDn	HBTDn	HBBDn	HCTDn	HCBDn
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6　未定义，读为“0”

Bit 5　**HATDn**：HAT 输出状态控制
0：输出 0
1：输出 1

Bit 4　**HABDn**：HAB 输出状态控制
0：输出 0
1：输出 1

Bit 3　**HBTDn**：HBT 输出状态控制
0：输出 0
1：输出 1

Bit 2　**HBBDn**：HBB 输出状态控制
0：输出 0
1：输出 1

Bit 1　**HCTDn**：HCT 输出状态控制
0：输出 0
1：输出 1

Bit 0　**HCBDn**：HCB 输出状态控制
0：输出 0
1：输出 1

- **HCHK_NUM 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	HCK_N4	HCK_N3	HCK_N2	HCK_N1	HCK_N0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5　未定义，读为“0”

Bit 4~0　**HCK_N4~HCK_N0**：霍尔噪声滤波器检测次数

- **HNF_MSEL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HNF_EN	HFR_SEL2	HFR_SEL1	HFR_SEL0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

Bit 3　**HNF_EN**：霍尔噪声滤波器控制
0：除能 (旁路)
1：使能

Bit 2~0　**HFR_SEL2~HFR_SEL0**：霍尔噪声滤波器时钟源选择
000：$f_{SYS}/2$
001：$f_{SYS}/4$
010：$f_{SYS}/8$
011：$f_{SYS}/16$
100：$f_{SYS}/32$

101：$f_{SYS}/64$
110：$f_{SYS}/128$
111：未使用

- **CMPSEL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C3SEL1	C3SEL0	C2SEL1	C2SEL0	C1SEL1	C1SEL0	CPNSW	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

Bit7~6 **C3SEL1~C3SEL0**：比较器 3 输出锁存控制
00：始终使能
01：PWM2O 信号
10：AT2/BT2/CT2 三组执行“OR”逻辑运算后的信号
11：AB2/BB2/CB2 三组执行“OR”逻辑运算后的信号

Bit5~4 **C2SEL1~C2SEL0**：比较器 2 输出锁存控制
00：始终使能
01：PWM1O 信号
10：AT2/BT2/CT2 三组执行“OR”逻辑运算后的信号
11：AB2/BB2/CB2 三组执行“OR”逻辑运算后的信号

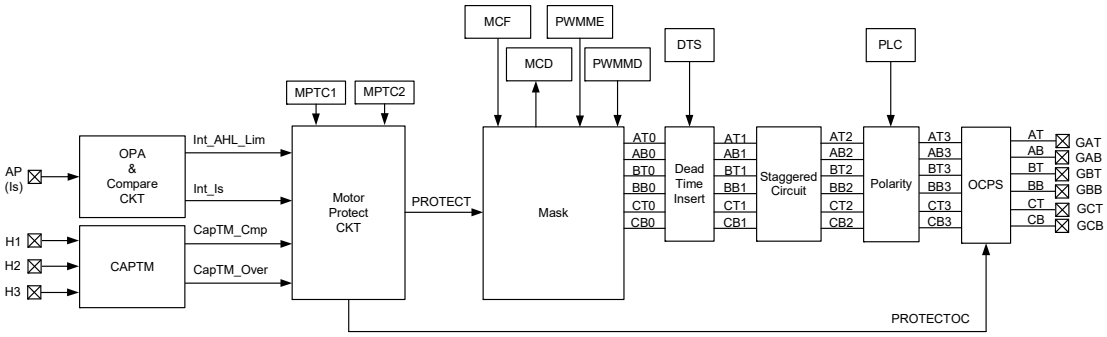
Bit3~2 **C1SEL1~C1SEL0**：比较器 1 输出锁存控制
00：始终使能
01：PWM0O 信号
10：AT2/BT2/CT2 三组执行“OR”逻辑运算后的信号
11：AB2/BB2/CB2 三组执行“OR”逻辑运算后的信号

Bit1 **CPNSW**：CPN 使能控制
0：C1N、C2N、C3N 使能 /CPN 除能
1：C1N、C2N、C3N 除能 /CPN 使能

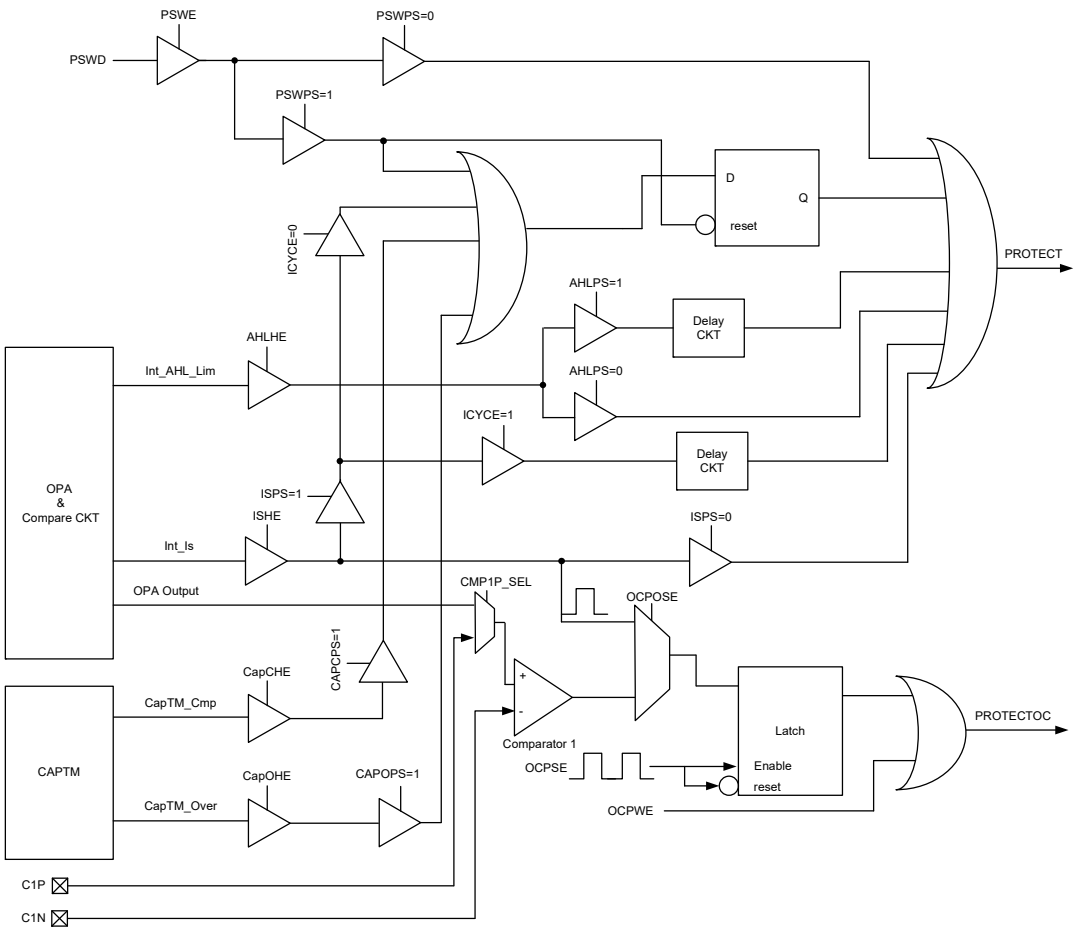
Bit0 未定义，读为“0”

电机保护功能

电机运转一般需要较大电流，为了减少电机破坏或基于安全考虑，需要保护电路以防过电流和电机堵转等情况发生。该单片机包括多个安全和保护特性。



电机保护电路方框图



保护功能控制逻辑

电机保护功能介绍

该单片机的 PROTECT 保护机制提供 3 种电机保护电路，可即时关闭电机，以避免电机被烧毁或提供额外安全保护。

保护特性有：

- 堵转检测
- 过流保护
- 软件关闭电机

当电机保护电路启动时，即 PROTECT=1，外部预驱晶体管对可选择刹车模式 (上侧关闭，下侧开启) 或是空转模式 (上侧和下侧都关闭)，由 MCF 寄存器的 FMOS 位决定。

电机保护电路有两种模式可以选择，一种是 Fault 模式，一种是 Pause 模式，由 MPTC2 寄存器决定。Fault 模式：PROTECT 保护机制启动由触发源启动状态决定；结束由触发源解除状态决定。Pause 模式：PROTECT 保护机制启动由触发源决定，保护功能结束则由软件决定。

另外，PROTECTOC 保护机制则是用来快速关断驱动信号时使用，提供了更立即的电流保护机制。当过流情况发生时，会忽略原本防呆电路后输出到极性控制电路的信号，转而送出 OCPS 寄存器提供的过电流保护电平来立即改变驱动

信号，以达到保护功率晶体管的目的。PROTECTOC 提供了 OCPSE (硬件) 或 OCPWE (软件) 触发机制，可在 MPTC1 寄存器中设置。当结束 PROTECTOC 保护时，无论是由硬件还是软件触发，都需要由软件方式解除。

电流保护机制

该单片机包含一个内部 OPA，一个 12 位高速 A/D 转换器、一个 8 位 D/A 转换器和一个比较器 0 以测量电机电流以及监控电机驱动过程中有无过电流的情形发生。如果检测到过流情况，可即时关闭电机外部驱动电路，以避免电机被烧毁。更多细节请参阅过流检测章节。

由于电机驱动过程 PCB 系统上噪声较大，加上使用内建 OPA 放大电路会将噪声放大，易导致误触发，故电流保护机制必须采用 Fault 模式。

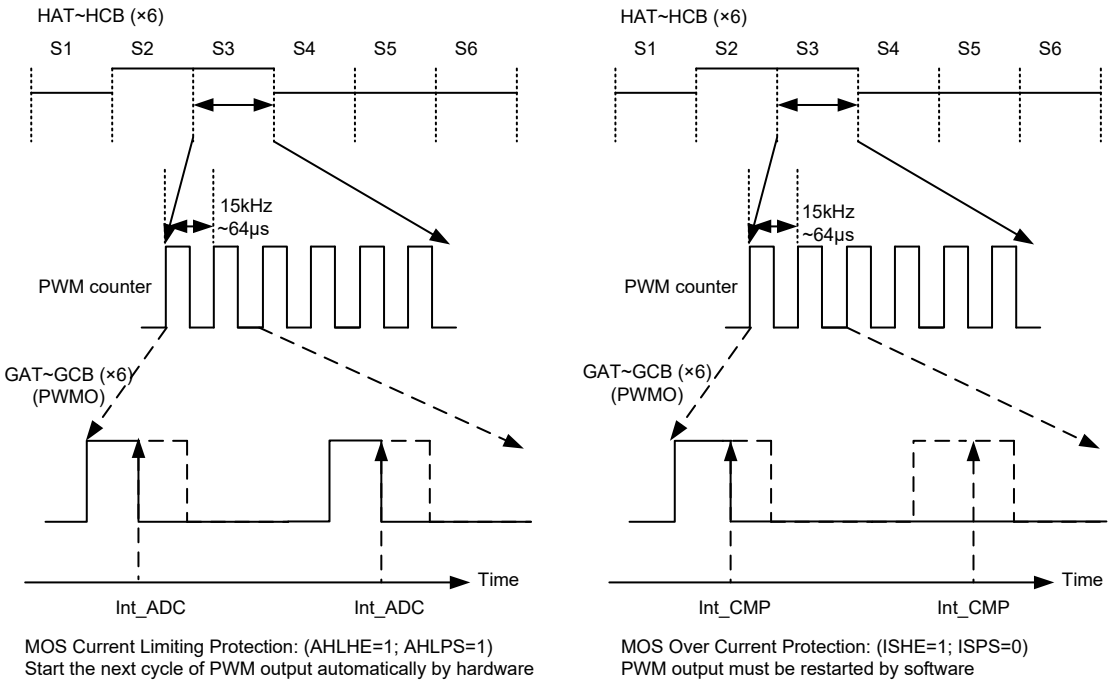
比较器 0 输出经过 2 个系统时钟的滤波电路后才产生 Int_Is 信号触发保护电路。

MOS 限电流保护机制 Int_AHL_Lim：当 AHLHE=0，除能硬件保护模式；当 AHLHE=1，使能硬件保护模式。限电流电路为纯硬件电路，且仅当 A/D 转换器通道选择了运算放大器输出时才有效。

AHLPS=0：一旦 Int_AHL_Lim 中断保护解除，保护电路立即响应产生 PWM 输出。

AHLPS=1：Int_AHL_Lim 中断保护解除时，PWM 输出会等待到下一个 PWM 周期到来才响应。

MOS 过电流保护机制 Int_Is：当 ISHE=0，除能硬件触发模式。当 ISHE=1，使能硬件触发模式。当 ISPS=0，选择 Fault 模式。当 ISPS=1，选择 Pause 模式。



过流保护时序

PROTECTOC 保护可选择软件或硬件两种触发机制，通过 MPTC1 寄存器的 OCPSE 或 OCPWE 位设置。

OCPSE=1：硬件直接过流触发，软件解除

必须确认已设置 ISHE 为高，使能硬件 Int_Is 过电流保护后再将 OCPSE 位置 1。之后当过电流发生就可以直接利用过电流比较中断 Int_Is 直接断开驱动信号。由于 Int_Is 是一个脉冲信号，所以必须通过锁存器来锁住该过流触发信号。当 PROTECTOC 为逻辑高时会忽略极性控制阶段的驱动信号，转而使用 OCPS 寄存器的过流保护逻辑来立即关断驱动信号，以达到保护功率晶体管的目的。若要解除 PROTECTOC 过流保护机制，需要利用 OCPSE 位由低到高来触发软件复位功能，使 PROTECTOC 变为逻辑低，此时可回到正常的 Polarity 驱动信号。

OCPWE=1：软件触发，软件解除

这是一种更为快速的过电流保护机制，可以直接设定 OCPWE 位为 1 来触发 PROTECTOC 机制。若需要解除过电流保护机制，直接将 OCPWE 清零来解除 PROTECTOC 信号，就可以回到正常的 Polarity 驱动信号。

电机堵转检测功能

针对霍尔传感器三相无刷直流应用，内建的 16 位的 CAPTM 定时器用来监控 H1，H2 和 H3 输入以判断转子的移动速度。通过设置寄存器 CAPTMAH 和 CAPTMAL 来侦测霍尔传感器输入 H1，H2 和 H3，以监控转子速度。若发生异常现象将产生中断 CapTM_Cmp 或 CapTM_Over 中断。可参考 CAPTM 章节。

CapTM_Cmp 堵转检测机制：当 CapCHE=0 时，除能硬件触发模式；当 CapCHE=1 时，使能硬件触发模式。电机堵转检查机制必须采用 Pause 模式，通过 CAPCPS 位选择。

CapTM_Over 堵转检测机制：当 CapOHE=0 时，除能硬件触发模式；当 CapOHE=1 时，使能硬件触发模式。接着通过置位 CAPOPS 位选择 Pause 模式。

电机保护寄存器介绍

寄存器 MPTC1、MPTC2 和 OCPS 均用于电机保护控制。

- **MPTC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PSWD	PSWE	CapOHE	CapCHE	ISHE	AHLHE	OCPWE	OCPSE
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 **PSWD**：电机保护软件模式数据
0：PSWD=0
1：PSWD=1
Pause 模式时，PSWD 可用来触发保护与解除保护。Fault 模式时，PSWD 只能用来解除保护，若使用 PSWD 解除保护机制，需设定 PSWE=1。

Bit 6 **PSWE**：电机保护软件模式控制
0：除能
1：使能
当电机保护软件模式已使能并被触发，用户需设置此位为 1→0→1 才能解除保护。

Bit 5 **CapOHE**：CapTM_Over 硬件触发模式控制
0：除能
1：使能

Bit 4 **CapCHE**：CapTM_Cmp 硬件触发模式控制
0：除能
1：使能

Bit 3 **ISHE**：Int_Is 硬件触发模式控制
0：除能
1：使能

Bit 2 **AHLHE**：Int_AHL_Lim 硬件触发模式控制
0：除能
1：使能

Bit 1 **OCPWE**：PROTECTOC 软件触发 & 软件解除设置
0：除能
1：使能

Bit 0 **OCPSE**：PROTECTOC 过流硬件触发 & 软件解除设置
0：除能
1：使能
当此功能已使能且 PROTECTOC 被触发，用户需设置此位清零再置高，才能解除保护。

- **MPTC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	OCPOSE	ICYCE	PSWPS	AHLPS	ISPS	CAPCPS	CAPOPS
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 **OCPOSE**：PTOTECTOC 硬件触发源选择
0：Int_Is
1：比较器 1 输出上升沿

Bit 5 **ICYCE**：过电流硬件逐周期保护
0：除能
1：使能
一旦 Int_Is 出现过流情况，该保护电路将关闭 PWM 输出，并在下一个 PWM 周期恢复 PWM 输出。但若仍存在过流情况，将立即再次关闭 PWM 输出。

Bit 4 **PSWPS**：软件模式下 Pause/Fault 模式选择
0：Fault 模式
1：Pause 模式

Bit 3 **AHLPS**：Int_AHL_Lim 模式选择
0：当保护电路解除保护时，立即重新开始 PWM 输出
1：当保护电路解除保护时，需等到下一个 PWM 周期才重新开始 PWM 输出

Bit 2 **ISPS**：Int_Is Pause/Fault 模式选择
0：Fault 模式
1：Pause 模式

Bit 1 **CAPCPS**：CapTM_Cmp Pause 模式选择
0：未定义
1：Pause 模式
注意，对于 CapTM_Cmp 触发保护必须置高 CAPCPS 位选择 Pause 模式。

Bit 0 **CAPOPS**：CapTM_Over Pause 模式选择
0：未定义
1：Pause 模式
注意，对于 CapTM_Over 触发保护必须置高 CAPOPS 选择 Pause 模式。

- **OCPS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	OCPCB	OCPCT	OCPBB	OCPBT	OCPAB	OCPAT
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6　未定义，读为“0”

Bit 5　**OCPCB**：GCB 过流保护输出选择
0：输出 0
1：输出 1

Bit 4　**OCPCT**：GCT 过流保护输出选择
0：输出 0
1：输出 1

Bit 3　**OCPBB**：GBB 过流保护输出选择
0：输出 0
1：输出 1

Bit 2　**OCPBT**：GBT 过流保护输出选择
0：输出 0
1：输出 1

Bit 1　**OCPAB**：GAB 过流保护输出选择
0：输出 0
1：输出 1

Bit 0　**OCPAT**：GAT 过流保护输出选择
0：输出 0
1：输出 1

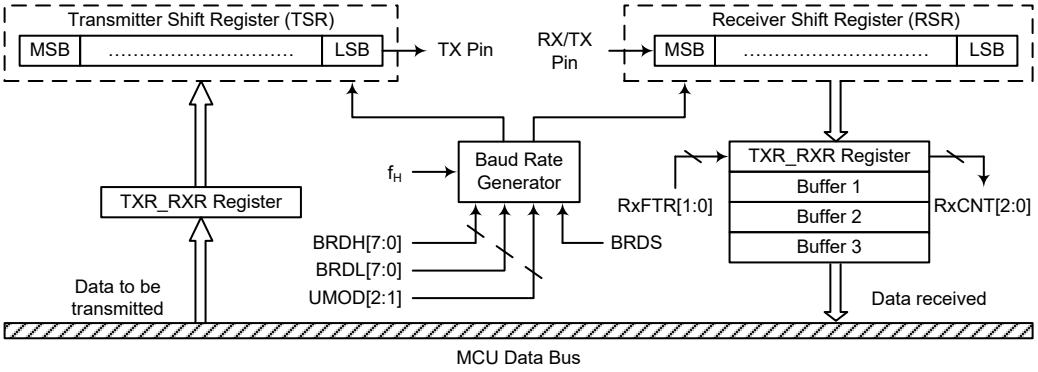
UART 接口

该单片机具有两个全双工或半双工的异步串行通信接口 – UART，可以很方便的与其它具有串行口的芯片通信。UART 具有许多功能特性，发送或接收串行数据时，将数据组成一个 8 位或 9 位的数据块，连同数据特征位一并传输。具有检测数据覆盖或帧错误等功能。UART 功能占用一个内部中断向量，当接收到数据或数据发送结束，触发中断。

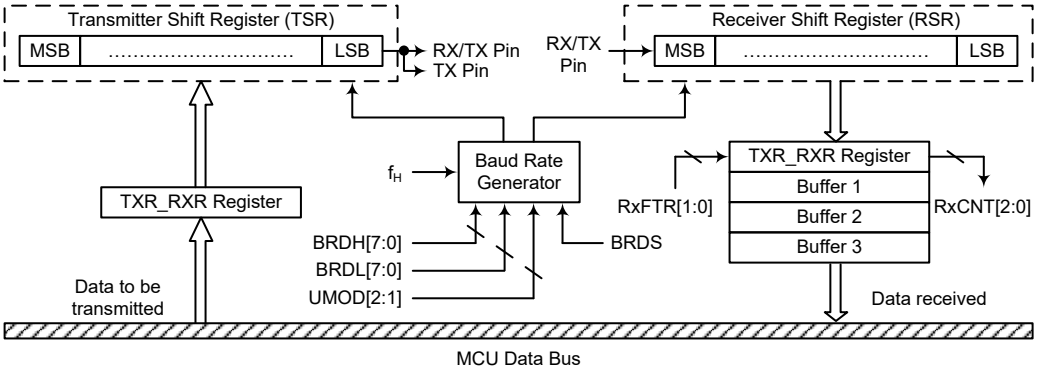
内置的 UART 功能包含以下特性：

- 全双工或半双工 (单线模式) 通用异步接收器 / 发送器
- 8 位或 9 位传输格式
- 奇校验、偶校验、Mark 校验、Space 校验或无校验
- 1 位或 2 位停止位
- 16 位预分频的波特率发生器
- 奇偶、帧、噪声和溢出检测
- 支持地址匹配中断 (最后一位 = 1)
- 独立的发送和接收使能
- 4-byte FIFO 接收缓冲器
- 1-byte FIFO 发送缓冲器
- RX/TX 引脚唤醒功能
- 发送和接收中断

- 中断可由下列条件触发：
  - ♦ 发送器为空
  - ♦ 发送器空闲
  - ♦ 接收完成
  - ♦ 接收器溢出
  - ♦ 地址匹配



UART 数据传输方框图 – SWM=0



UART 数据传输方框图 – SWM=1

UART 外部引脚

内部 UART 有两个外部引脚 TX 和 RX/TX，可与外部串行接口进行通信。TX 和 RX/TX 与 I/O 口或其它功能共用引脚。在使用 UART 功能前，应先通过相应的引脚共用功能选择寄存器，选择 TX 和 RX/TX 引脚功能。当 UARTEN、TXEN 和 RXEN 位置高时，将自动设置这些 I/O 脚或其它共用功能脚作为发送输出和接收输入。此时，用作发送输出的引脚其内部上拉电阻被除能，而用作接收输入的引脚其内部上拉电阻由相应的上拉电阻控制位控制。当 UARTEN、TXEN 或 RXEN 位清零除能 TX 或 RX/TX 引脚功能后，TX 或 RX/TX 引脚将处于浮空状态。这时 TX 或 RX/TX 引脚是否连接内部上拉电阻是由相应的 I/O 上拉电阻控制位决定的。

UART 单线模式

UART 功能支持单线模式通信，通过 UCR3 寄存器中的 SWM 位选择。当设置该位为高时，UART 功能将工作在单线模式。在单线模式下，单个 RX/TX 引脚通过相关控制位的不同设置即可完成数据的发送与接收。设置 RXEN 位为高，RX/TX 引脚用作接收引脚。将 RXEN 位清零，同时设置 TXEN 位为高，RX/TX

引脚将作为发送引脚。

在单线模式下建议不要将 RXEN 位和 TXEN 位同时设置为高。若 RXEN 位和 TXEN 位同时为高，RXEN 位具有更高的优先级，此时 UART 为接收器状态。

需特别注意的是，UART 章节所有内容是基于 UART 全双工通信来对 UART 功能进行描述，相关的说明除引脚的使用外，对半双工通信 (单线模式) 同样适用。在理解单线模式通信时，全双工通信中使用的 TX 引脚需取代为 RX/TX 引脚。

单线模式下，通过合理的软件配置，数据也可在 TX 引脚进行发送。因此数据可通过 RX/TX 和 TX 引脚输出。

UART 数据传输方案

UART 数据传输方框图显示了 UART 的整体结构。需要发送的数据首先写入 TXR_RXR 寄存器，接着此数据被传输到发送移位寄存器 TSR 中，然后在波特率发生器的控制下将 TSR 寄存器中数据一位位地移到 TX 引脚上，低位在前。TXR_RXR 寄存器被映射到单片机的数据存储器中，而发送移位寄存器没有实际地址，所以发送移位寄存器不可直接操作。

数据在波特率发生器的控制下，低位在前高位在后，从外部引脚 RX/TX 进入接收移位寄存器 RSR。当数据接收完成，数据从接收移位寄存器移入可被用户程序操作的 TXR_RXR 寄存器中。TXR_RXR 寄存器被映射到单片机数据存储器中，而接收移位寄存器没有实际地址，所以接收移位寄存器不可直接操作。

需要注意的是，发送和接收都是共用同一个数据存储器地址的数据寄存器，即 TXR_RXR 寄存器。

UART 状态和控制寄存器

与 UART 功能相关的有九个寄存器。UCR3 寄存器的 SWM 位用于使能 / 除能 UART 单线模式。其它包括控制 UART 模块整体功能的 USR、UCR1、UCR2、UFCR 和 RxCNT 寄存器，控制波特率的 BRDH 和 BRDL 寄存器，管理发送和接收数据的数据寄存器 TXR_RXR。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
UCR3	—	—	—	—	—	—	—	SWM
TXR_RXR	D7	D6	D5	D4	D3	D2	D1	D0
BRDH	D7	D6	D5	D4	D3	D2	D1	D0
BRDL	D7	D6	D5	D4	D3	D2	D1	D0
UFCR	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
RxCNT	—	—	—	—	—	D2	D1	D0

UART 寄存器列表

- **USR 寄存器**

寄存器 USR 是 UART 的状态寄存器，可以通过程序读取。所有 USR 位是只读的。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **PERR**：奇偶校验出错标志位
0：奇偶校验正确
1：奇偶校验出错
PERR 是奇偶校验出错标志位。若 PERR=0，奇偶校验正确；若 PERR=1，接收到的数据奇偶校验出错。只有使能了奇偶校验，选择了校验类型，此位才有效。可使用软件清除该标志位，即先读取 USR 寄存器再读 TXR_RXR 寄存器来清除此位。

Bit 6 **NF**：噪声干扰标志位
0：未检测到噪声
1：检测到噪声
NF 是噪声干扰标志位。若 NF=0，没有受到噪声干扰；若 NF=1，UART 接收数据时受到噪声干扰。它与 RXIF 在同周期内置位，但不会与溢出标志位同时置位。可使用软件清除该标志位，即先读取 USR 寄存器再读 TXR_RXR 寄存器将清除此标志位。

Bit 5 **FERR**：帧错误标志位
0：无帧错误发生
1：有帧错误发生
FERR 是帧错误标志位。若 FERR=0，没有帧错误发生；若 FERR=1，当前的数据发生了帧错误。可使用软件清除该标志位，即先读取 USR 寄存器再读 TXR_RXR 寄存器来清除此位。

Bit 4 **OERR**：溢出错误标志位
0：无溢出错误发生
1：有溢出错误发生
OERR 是溢出错误标志位，表示接收缓冲器是否溢出。若 OERR=0，没有溢出错误；若 OERR=1，发生了溢出错误，它将禁止下一组数据的接收。可通过软件清除该标志位，即先读取 USR 寄存器再读 TXR_RXR 寄存器将清除此标志位。

Bit 3 **RIDLE**：接收状态标志位
0：正在接收数据
1：接收器空闲
RIDLE 是接收状态标志位。若 RIDLE=0，正在接收数据；若 RIDLE=1，接收器空闲。在接收到停止位和下一个数据的起始位之间，RIDLE 被置位，表明 UART 空闲，RX/TX 脚处于逻辑高状态。

Bit 2 **RXIF**：接收寄存器状态标志位
0：TXR_RXR 寄存器为空
1：TXR_RXR 寄存器含有有效数据且达到接收 FIFO 触发等级
RXIF 是接收寄存器状态标志位。当 RXIF=0，TXR_RXR 寄存器为空；当 RXIF=1，TXR_RXR 寄存器接收到新数据。当数据从移位寄存器加载到 TXR_RXR 寄存器中，且达到接收 FIFO 触发等级，如果 UCR2 寄存器中的 RIE=1，则会触发中断。当接收数据时检测到一个或多个错误时，相应的标志位 NF、FERR 或 PERR 会在同一周期内置位。读取 USR 寄存器再读 TXR_RXR 寄存器，如果 TXR_RXR 寄存器中没有新的数据，那么将清除 RXIF 标志。

Bit 1 **TIDLE**：数据发送完成标志位
0：数据传输中
1：无数据传输
TIDLE 是数据发送完成标志位。若 TIDLE=0，数据传输中。当 TXIF=1 且数据

发送完毕或者暂停字被发送时，TIDLE 置位。TIDLE=1，TX 引脚空闲且处于逻辑高状态。读取 USR 寄存器再写 TXR_RXR 寄存器将清除 TIDLE 位。数据字符或暂停字就绪时，不会产生该标志位。

Bit 0 **TXIF**：发送数据寄存器 TXR_RXR 状态位
0：数据还没有从缓冲器加载到移位寄存器中
1：数据已从缓冲器加载到移位寄存器中 (TXR_RXR 数据寄存器为空)
TXIF 是发送数据寄存器为空标志位。若 TXIF=0，数据还没有从缓冲器加载到移位寄存器中；若 TXIF=1，数据已从缓冲器中加载到移位寄存器中。读取 USR 存器再写 TXR_RXR 寄存器将清除 TXIF。当 TXEN 被置位，由于发送缓冲器未满，TXIF 也会被置位。

- **UCR1 寄存器**

UCR1、UCR2 和 UCR3 是 UART 的三个控制寄存器，用来定义各种 UART 功能，例如 UART 的使能与除能、奇偶校验控制和传输数据的长度以及单线模式通信等等。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”：未知

Bit 7 **UARTEN**：UART 功能使能位
0：UART 除能，TX 和 RX/TX 脚处于浮空状态
1：UART 使能，TX 和 RX/TX 脚作为 UART 功能引脚
此位为 UART 的使能位。UARTEN=0，UART 除能，RX/TX 和 TX 处于浮空状态；UARTEN=1，UART 使能，TX 和 RX/TX 将分别由 TXEN 和 RXEN 控制。
当 UART 被除能将清除缓冲器，所有缓冲器中的数据将被忽略，另外波特率计数器、错误和状态标志位被复位，TXEN、RXEN、TXBRK、RXIF、OERR、FERR、PERR 和 NF 位以及 RxCNT 寄存器清零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2、UCR3、UFCR、BRDH 和 BRDL 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

Bit 6 **BNO**：数据传输位数选择位
0：8-bit 数据传输
1：9-bit 数据传输
BNO 是数据传输位数选择位。BNO=1，传输数据为 9 位；BNO=0，传输数据为 8 位。若选择了 9 位数据传输格式，RX8 和 TX8 将分别存储接收和发送数据的第 9 位。
需要注意的是，若 BNO=1，奇偶校验使能时，数据的第 9 位为奇偶校验位，不会传送到 RX8。若 BNO=0，奇偶校验使能时，数据的第 8 位为奇偶校验位，不会传送到 TXR_RXR.7。

Bit 5 **PREN**：奇偶校验使能位
0：奇偶校验除能
1：奇偶校验使能
此位为奇偶校验使能位。PREN=1，使能奇偶校验；PREN=0，除能奇偶校验。

Bit 4~3 **PRT1~PRT0**：奇偶校验选择位
00：偶校验
01：奇校验
10：Mark 校验
11：Space 校验
奇偶校验选择位。PRT[1:0]=00，偶校验；PRT[1:0]=01，奇校验；PRT[1:0]=10，Mark 校验，校验位始终为 1；PRT[1:0]=11，Space 校验，校验位为 0。

Bit 2 **TXBRK**：暂停字发送控制位
0：没有暂停字要发送
1：发送暂停字
TXBRK 是暂停字发送控制位。TXBRK=0，没有暂停字要发送，TX 引脚正常操作；TXBRK=1，将会发送暂停字，发送器将发送逻辑“0”。若 TXBRK 为高，缓冲器中数据发送完毕后，发送器输出将至少保持 13 位宽的低电平直至 TXBRK 复位。

Bit 1 **RX8**：接收 9-bit 数据传输格式中的第 9 位 (只读)
此位只有在传输数据为 9 位的格式中有效，用来存储接收数据的第 9 位。BNO 是用来控制传输位数是 8 位还是 9 位。

Bit 0 **TX8**：发送 9-bit 数据传输格式中的第 9 位 (只写)
此位只有在传输数据为 9 位的格式中有效，用来存储发送数据的第 9 位。BNO 是用来控制传输位数是 8 位还是 9 位。

- **UCR2 寄存器**

UCR2 是 UART 的第二个控制寄存器，它的主要功能是控制发送器、接收器以及各种 UART 中断源的使能或除能。它也可用来选择停止位的长度，使能接收唤醒和地址侦测。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
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 **TXEN**：UART 发送使能位
0：UART 发送除能
1：UART 发送使能
此位为发送使能位。TXEN=0，发送将被除能，发送器立刻停止工作。另外发送缓冲器将被复位，此时 TX 引脚将处于浮空状态。若 TXEN=1 且 UARTEN=1，则发送将被使能，TX 引脚将由 UART 来控制。在数据传输时清除 TXEN 将中止数据发送且复位发送器，此时 TX 引脚将处于浮空状态。

Bit 6 **RXEN**：UART 接收使能位
0：UART 接收除能
1：UART 接收使能
此位为接收使能位。RXEN=0，接收将被除能，接收器立刻停止工作。另外接收缓冲器将被复位，此时 RX/TX 引脚将处于浮空状态。若 RXEN=1 且 UARTEN=1，则接收将被使能，RX/TX 引脚将由 UART 来控制。在数据传输时清除 RXEN 将中止数据接收且复位接收器，此时 RX/TX 引脚将处于浮空状态。

Bit 5 **STOPS**：发送器停止位的长度选择位
0：有一位停止位
1：有两位停止位
此位用来设置发送器停止位的长度。STOP=1，有两位停止位；STOP=0，只有一位停止位。

Bit 4 **ADDEN**：地址检测使能位
0：地址检测除能
1：地址检测使能
此位为地址检测使能和除能位。ADDEN=1，地址检测使能，此时数据的第 8 位 (BNO=0) 或第 9 位 (BNO=1) 为高，那么接到的是地址而非数据。若相应的中断使能且接收到的值最高位为 1，那么中断请求标志将会被置位，若地址检测功能使能且最高位为 0，那么将不会产生中断且收到的数据也会被忽略。

Bit 3 **WAKE**：RX/TX 脚下降沿唤醒 UART 功能使能位
0：RX/TX 脚下降沿唤醒 UART 功能除能
1：RX/TX 脚下降沿唤醒 UART 功能使能
此位用于控制 RX/TX 引脚下降沿时是否唤醒 UART 功能。此位仅当 UART 时

钟源 f_H 关闭时有效。若 UART 时钟源 f_H 还开启，则 RX/TX 引脚唤醒 UART 功能无效。若此位置高且 UART 时钟 f_H 关闭，当 RX/TX 引脚发生下降沿时会产生 UART 唤醒请求。若相应的中断使能，将产生 RX/TX 引脚唤醒 UART 的中断，以告知单片机使其通过应用程序开启 UART 时钟源 f_H，从而唤醒 UART 功能。否则，若此位为低，即使 RX/TX 引脚发生下降沿也无法恢复 UART 功能。

Bit 2 **RIE**：接收中断使能位
0：接收中断除能
1：接收中断使能
此位为接收中断使能或除能位。若 RIE=1，当 OERR 或 RXIF 置位时，UART 的中断请求标志置位；若 RIE=0，UART 中断请求标志不受 OERR 和 RXIF 影响。

Bit 1 **TIIE**：发送器空闲中断使能位
0：发送器空闲中断除能
1：发送器空闲中断使能
此位为发送器空闲中断的使能或除能位。若 TIIE=1，当发送器空闲触发 TIDLE 置位时，UART 的中断请求标志置位；若 TIIE=0，UART 中断请求标志不受 TIDLE 的影响。

Bit 0 **TEIE**：发送寄存器为空中断使能位
0：发送寄存器为空中断除能
1：发送寄存器为空中断使能
此位为发送寄存器为空中断的使能或除能位。若 TEIE=1，当发送器为空触发 TXIF 置位时，UART 的中断请求标志置位；若 TEIE=0，UART 中断请求标志不受 TXIF 的影响。

- **UCR3 寄存器**

UCR3 寄存器用于使能 UART 单线模式通信。顾名思义，在单线模式下 UART 只需要使用一条线，RX/TX，在 UCR2 寄存器中的 RXEN 和 TXEN 位控制下即可完成通信。

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

Bit 7~1 未定义，读为“0”

Bit 0 **SWM**：单线模式使能控制位
0：除能，RX/TX 引脚仅用于 UART 接收功能
1：使能，RX/TX 引脚在 RXEN 和 TXEN 位控制下可用于 UART 接收或发送功能
注意，单线模式使能，若将 RXEN 和 TXEN 位同时设置为高，RX/TX 引脚仅用作接收功能。

- **TXR_RXR 寄存器**

TXR_RXR 是一个数据寄存器，用来存储 TX 引脚将要发送或 RX/TX 引脚正在接收的数据。

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**：UART 发送 / 接收数据位 bit 7~bit 0

- **BRDH 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**：波特率分频器高字节

波特率分频器 BRD (BRDH/BRDL) 用来定义 UART 时钟的分频比率。

波特率 = f_H / (BRD+UMOD/8)

BRD=16~65535 或 8~65535，取决于 BRDS

注：1. 当 BRDS=0 时，BRD 值不应小于 16；当 BRDS=1 时，BRD 值不应小于 8，否则可能发生错误。

2. 必须先对 BRDL 写值，再对 BRDH 写值，否则可能发生错误。

- **BRDL 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**：波特率分频器低字节

波特率分频器 BRD (BRDH/BRDL) 用来定义 UART 时钟的分频比率。

波特率 = f_H / (BRD+UMOD/8)

BRD=16~65535 或 8~65535，取决于 BRDS

注：1. 当 BRDS=0 时，BRD 值不应小于 16；当 BRDS=1 时，BRD 值不应小于 8，否则可能发生错误。

2. 必须先对 BRDL 写值，再对 BRDH 写值，否则可能发生错误。

- **UFCR 寄存器**

UFCR 寄存器是 FIFO 控制寄存器，用于 UART 调制控制、BRD 范围选择、RXIF 和中断的触发电平选择。

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 未定义，读为“0”

Bit 5~3 **UMOD2~UMOD0**：UART 调制控制位

该调制控制位用于校正接收到的或发送出的 UART 信号的波特率。这几位决定是否应该在一个 UART 位时间内加入额外的 UART 时钟周期。每个 UART 位时间 UMOD2~UMOD0 将被加入到内部累加器中。直到进位到 bit 3，对应的 UART 位时间增加一个 UART 时钟周期。

Bit 2 **BRDS**：BRD 范围选择

0：BRD=16~65535

1：BRD=8~65535

BRDS 位用于控制 UART 位时间内的采样点。若 BRDS=0，则在一个 UART 位时间内采样点为 BRD/2、BRD/2+1×f_H 和 BRD/2+2×f_H。若 BRDS=1，则在一个 UART 位时间内采样点为 BRD/2-1×f_H、BRD/2、BRD/2+2×f_H。

Bit 1~0 **RxFTR1~RxFTR0**：接收器 FIFO 触发等级 (字节数)

00：接收器 FIFO 中有 4 个字节

01：接收器 FIFO 中有 1 个以上字节

10：接收器 FIFO 中有 2 个以上字节
11：接收器 FIFO 中有 3 个以上字节

对于接收器，这几位用于定义接收器 FIFO 中接收到的数据字节数，达到设定字节数将触发 RXIF 位置高，若 RIE 位使能，还将产生一个中断。复位后接收器 FIFO 为空。

- **RxCNT 寄存器**

RxCNT 寄存器是一个计数器，用来表示未被 MCU 读取的接收器 FIFO 中接收的数据字节数。这个寄存器是只读的。

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

Bit 7~3　未定义，读为“0”

Bit 2~0　**D2~D0**：接收器 FIFO 计数器

RxCNT 寄存器是一个计数器，用来表示未被 MCU 读取的接收器 FIFO 中接收的数据字节数。当接收器 FIFO 接收到一个字节数据时，RxCNT 将自动加一；当 MCU 从接收器 FIFO 中读取一个字节数据时，RxCNT 将自动减一。如果接收器 FIFO 中有 4 个字节的数据，那么第 5 个数据将保存在移位寄存器中。如果有第 6 个数据，第 6 个数据将保存在移位寄存器中。但是 RxCNT 的值仍然是 4。当复位发生或 UARTEN=1 时，RxCNT 将被清零。这个寄存器是只读的。

波特率发生器

UART 自身具有一个波特率发生器，通过它可以设定数据传输速率。波特率是由一个独立的内部 16 位计数器产生，它由 BRDH/BRDL 寄存器和 UART 调制控制位 UMOD2~UMOD0 来控制。如果由 UART 时钟 f_H 生成所需的波特率 BR，则：

f_H/BR = 整数部分 + 小数部分

整数部分载入 BRD (BRDH/BRDL)，小数部分乘以 8，四舍五入后载入 UMOD 位域，如下：

$BRD = TRUNC(f_H/BR)$

$UMOD = ROUND[MOD(f_H/BR)\times 8]$

因此，实际波特率如下：

波特率 = $f_H/[BRD+(UMOD/8)]$

波特率和误差的计算

若选用 4MHz 时钟频率且期望的波特率为 230400，计算 BRDH/BRDL 寄存器的值，实际波特率和误差。

根据上述公式，$BRD = TRUNC(f_H/BR) = TRUNC(17.36111) = 17$

$UMOD = ROUND[MOD(f_H/BR)\times 8] = ROUND(0.36111\times 8) = ROUND(2.88888) = 3$

实际波特率 = $f_H/[BRD+(UMOD/8)] = 230215.83$

因此，误差 = (230215.83-230400)/230400 = -0.08%

调制控制范例

为了得到 UART 调制控制位 UMOD2~UMOD0 的最佳拟合位序列，可以采用以下算法：首先，将理论除法因子的小数部分乘以 8。然后将结果四舍五入，并写入 UMOD2~UMOD0 位。每个 UART 位时间 UMOD2~UMOD0 将被加入到内

部累加器中。直到进位到 bit 3，对应的 UART 位时间增加一个 UART 时钟周期。下面以之前计算的小数 0.36111 为例来做说明：UMOD[2:0]=ROUND(0.36111×8)=011b。

小数叠加	进位到 Bit 3	UART 位时间序列	额外的 UART 时钟周期
0000b + 0011b = 0011b	No	起始位	No
0011b + 0011b = 0110b	No	D0	No
0110b + 0011b = 1001b	Yes	D1	Yes
1001b + 0011b = 1100b	No	D2	No
1100b + 0011b = 1111b	No	D3	No
1111b + 0011b = 0010b	Yes	D4	Yes
0010b + 0011b = 0101b	No	D5	No
0101b + 0011b = 1000b	Yes	D6	Yes
1000b + 0011b = 1011b	No	D7	No
1011b + 0011b = 1110b	No	校验位	No
1110b + 0011b = 0001b	Yes	停止位	Yes

波特率校正范例

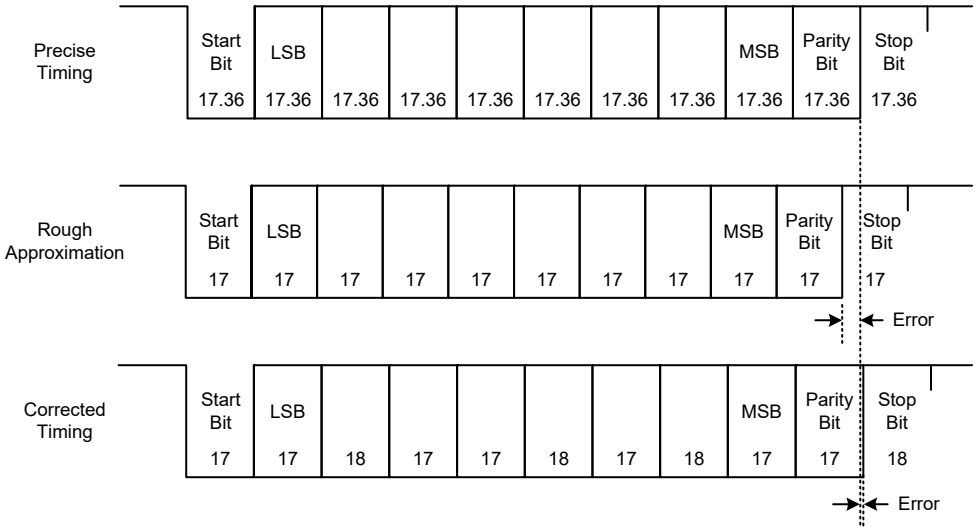
下图为一个使用 UART 时钟 f_H 生成的波特率为 230400 的示例，数据格式是：8 位数据位，奇偶校验使能，无地址位，2 位停止位。

下图显示了三个不同的帧：

上帧为准确帧，位长为 17.36 个 f_H 时钟周期 (400000/230400=17.36)。

中间帧采用粗略估计，位长为 17 个 f_H 时钟周期。

下帧显示的是校正后的帧，采用 UART 调制控制位 UMOD2~UMOD0 的最佳拟合算法。



UART 模块的设置与控制

UART 采用标准的不归零码传输数据，这种方法通常被称为 NRZ 法。它由 1 位起始位，8 位或 9 位数据位和 1 位或者两位停止位组成。奇偶校验是由硬件自动完成的，可设置成奇校验、偶校验、Mark 校验、Space 校验和无校验五种格

式。常用的数据传输格式由 8 位数据位，1 位停止位，无校验组成，用 8、N、1 表示，它是系统上电的默认格式。数据位数、停止位数和奇偶校验由 BNO、PRT1~PRT0、PREN 和 STOPS 设定。用于数据发送和接收的波特率由一个内部的 16 位波特率发送器产生，数据传输时低位在前高位在后。尽管 UART 发送器和接收器在功能上相互独立，但它们使用相同的数据传输格式和波特率，在任何情况下，停止位是必须的。

UART 的使能和除能

UART 是由 UCR1 寄存器的 UARTEN 位来使能和除能的。若 UARTEN、TXEN 和 RXEN 都为高，则 TX 和 RX/TX 分别为 UART 的发送端口和接收端口。若没有数据发送，TX 引脚默认状态为高电平。

UARTEN 清零将除能 TX 和 RX/TX，通过设置相关引脚共用控制位，这两个引脚可用作普通 I/O 口或其它引脚共用功能。当 UART 被除能时将清空缓冲器，所有缓冲器中的数据将被忽略，另外一些使能控制、错误标志和状态标志将被复 位， 如 TXEN、RXEN、TXBRK、RXIF、OERR、FERR、PERR 和 NF 位以及 RxCNT 寄存器清零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2、UCR3、UFCR、BRDH 和 BRDL 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，UART 也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

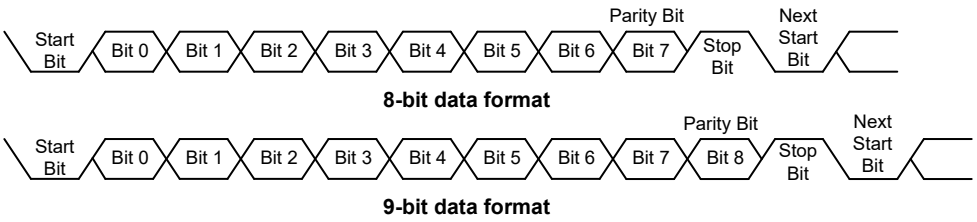
数据位、停止位位数以及奇偶校验的选择

数据传输格式由数据长度、是否校验、校验类型、地址位以及停止位长度组成。它们都是由 UCR1 和 UCR2 寄存器的各个位控制的。BNO 决定数据传输是 8 位还是 9 位；PRT1~PRT0 决定校验类型；PREN 决定是否选择奇偶校验、Mark 校验或者 Space 校验；而 STOPS 决定选用 1 位还是 2 位停止位。下表列出了各种数据传输格式。若地址检测功能使能，地址位，即数据字节的最高位，用来确定此帧是地址还是数据。停止位的长度和数据位的长度无关，且只有发送器需设置停止位长度。接收器只接收一个停止位。

起始位	数据位	地址位	校验位	停止位
8 位数据位				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
9 位数据位				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

发送和接收数据格式

下图是传输 8 位和 9 位数据的波形。



UART 发送器

UCR1 寄存器的 BNO 位是控制数据传输的长度。BNO=1 其长度为 9 位，第 9 位 MSB 存储在 UCR1 寄存器的 TX8 中。发送器的核心是发送移位寄存器 TSR，它的数据由发送寄存器 TXR_RXR 提供，应用程序只须将发送数据写入 TXR_RXR 寄存器。上组数据的停止位发出前，TSR 寄存器禁止写入。如果还有新的数据要发送，一旦停止位发出，待发数据将会从 TXR_RXR 寄存器加载到 TSR 寄存器。TSR 不像其它寄存器一样映射到数据存储器，所以应用程序不能对其进行读写操作。TXEN=1，发送使能，但若 TXR_RXR 寄存器没有数据或者波特率没有设置，发送器将不会工作。先写 TXR_RXR 寄存器再置高 TXEN 也会触发发送。当发送器使能，若 TSR 寄存器为空，数据写入 TXR_RXR 寄存器将会直接加载到 TSR 寄存器中。发送器工作时，TXEN 清零，发送器将立刻停止工作并且复位，此时通过设置相关引脚共用控制位，TX 引脚用作普通 I/O 口或其它引脚共用功能。

发送数据

当 UART 发送数据时，数据从移位寄存器中移到 TX 引脚上，其低位在前高位在后。在发送模式中，TXR_RXR 寄存器在内部总线和发送移位寄存器间形成一个缓冲。如果选择 9 位数据传输格式，最高位 MSB 取自 UCR1 寄存器的 TX8。

发送器初始化可由如下步骤完成：

- 正确地设置 BNO、PRT1~PRT0、PREN 和 STOPS 位以确定数据长度、校验类型和停止位长度。
- 设置 BRDH, BRDL 寄存器和 UMOD2~UMOD0 位，选择期望的波特率。
- 置高 TXEN，使能 UART 发送器且使 TX 作为 UART 的发送端。
- 读取 USR 寄存器，然后将待发数据写入 TXR_RXR 寄存器。注意，此步骤会清除 TXIF 标志位。

如果要发送多个数据只需重复上一步骤。

当 TXIF=0 时，数据将禁止写入 TXR_RXR 寄存器。可以通过以下步骤来清除 TXIF：

1. 读取 USR 寄存器
2. 写 TXR_RXR 寄存器

只读标志位 TXIF 由 UART 硬件置位。若 TXIF=1，TXR_RXR 寄存器为空，其它数据可以写入而不会覆盖之前的数据。若 TEIE=1，TXIF 标志位会产生中断。在数据传输时，写 TXR_RXR 指令会将待发数据暂存在 TXR_RXR 寄存器中，当前数据发送完毕后，待发数据被加载到发送移位寄存器中。当发送器空闲时，写 TXR_RXR 指令会将数据直接加载到 TSR 寄存器中，数据传输立刻开始且 TXIF 置位。当发送完停止位或暂停帧后，表示一帧数据已发送完毕，此时 TIDLE 位将被置位。

可以通过以下步骤来清除 TIDLE：

1. 读取 USR 寄存器
2. 写 TXR_RXR 寄存器

请注意，清除 TXIF 和 TIDLE 软件执行次序相同。

发送暂停字

若 TXBRK=1 且此状态保持时间超过 [(BRD+1) × t_H]，且 TIDLE=1，下一帧将

会发送暂停字。它是由一个起始位、13×N (N=1，2……) 位逻辑 0 组成。置位 TXBRK 将会发送暂停字，而清除 TXBRK 将产生停止位，传输暂停字不会产生中断。需要注意的是，暂停字至少 13 位宽。若 TXBRK 持续为高，那么发送器会一直发送暂停字；当应用程序将 TXBRK 清零后，发送器结束最后一帧暂停字的发送后接着发送一位或两位停止位。最后一帧暂停字的结尾自动为高电平，以确保下一帧数据起始位的检测。

UART 接收器

UART 接收器支持 8 位或者 9 位数据接收。若 BNO=1，数据长度为 9 位，而最高位 MSB 存放在 UCR1 寄存器的 RX8 中。接收器的核心是串行移位寄存器 RSR。RX/TX 引脚上的数据送入数据恢复器中，它在 16 倍波特率的频率下工作，而串行移位器工作在正常波特率下。当在 RX/TX 引脚上检测到停止位，若 TXR_RXR 寄存器为空，数据从 RSR 寄存器中加载到 TXR_RXR 寄存器。RX/TX 引脚上的每一位数据会被采样三次以判断其逻辑状态。RSR 不像其它寄存器一样映射在数据存储器，所以应用程序不能对其进行读写操作。

接收数据

当 UART 接收数据时，数据低位在前高位在后，连续地从 RX/TX 引脚进入移位寄存器。TXR_RXR 寄存器在内部总线和接收移位寄存器间形成一个缓冲。TXR_RXR 寄存器是一个四字节深度的 FIFO 缓冲器，它能保存四字节数据的同时接收第五字节数据，应用程序必须保证在接收完第五字节前读取 TXR_RXR 寄存器，否则忽略第五字节数据并且发生溢出错误。

接收器的初始化可由如下步骤完成：

- 正确地设置 BNO、PRT1~PRT0 和 PREN 位以确定数据长度和校验类型。
- 设置 BRDH、BRDL 寄存器以及 UMOD2~UMOD0 位，选择期望的波特率。
- 置高 RXEN，使能 UART 接收器且使 RX/TX 作为 UART 的接收端。

此时接收器被使能并检测起始位。

接收数据将会发生如下事件：

- 当 TXR_RXR 寄存器中包含有效数据时，USR 寄存器中的 RXIF 位将会置位，可以通过轮询 RxCNT 寄存器的内容来检查有效数据字节数。
- 数据从 RSR 寄存器加载到 TXR_RXR 寄存器中，并且达到接收器 FIFO 触发字节数，若 RIE=1，将产生中断。
- 若接收器检测到帧错误、噪声干扰错误、奇偶出错或溢出错误，那么相应的错误标志位置位。

可以通过如下步骤来清除 RXIF：

1. 读取 USR 寄存器
2. 读取 TXR_RXR 寄存器

接收暂停字

UART 接收任何暂停字都会当作帧错误处理。接收器只根据 BNO 位的设置外加一个停止位来确定一帧数据的长度。若暂停字位数大于 BNO 位指定的长度外加一个停止位，接收器认为接收已完毕，RXIF 和 FERR 置位，TXR_RXR 寄存器清 0，若相应的中断允许且 RIDLE 为高将会产生中断。暂停字只会被认为包含信息 0 且会置位 FERR 标志位。如果检测到较长的暂停信号，接收器会将此信号视为包含一个起始位、数据位和无效的停止位的数据帧并且置位 FERR 标志位。在下个开始位到来之前，接收器必须等待一个有效的停止位。接收器不

会假定线上的暂停信号是下一个开始位。暂停字将会加载到缓冲器中，在接收到停止位前不会再接收数据，没有检测到停止位也会置位只读标志位 RIDLE。

UART 接收到暂停字会产生以下事件：

- 帧错误标志位 FERR 置位。
- TXR_RXR 寄存器清零。
- OERR、NF、PERR、RIDLE 或 RXIF 可能会置位。

空闲状态

当 UART 接收数据时，即在起初位和停止位之间，USR 寄存器的接收状态标志位 RIDLE 清零。在停止位和下一帧数据的起始位之间，RIDLE 被置位，表示接收器空闲。

接收中断

USR 寄存器的只读标志位 RXIF 由接收器的边沿触发置位。若 RIE=1，数据从移位寄存器 RSR 加载到 TXR_RXR 寄存器时产生中断，同样地，溢出也会产生中断。

接收错误处理

UART 会产生几种接收错误，下面部分将描述各错误以及怎样处理。

溢出 – OERR 标志

TXR_RXR 寄存器是一个四字节深度的 FIFO 缓冲器，它能保存四字节数据的同时接收第五字节数据，应用程序必须保证在接收完第五字节前读取 TXR_RXR 寄存器，否则发生溢出错误。

产生溢出错误时将会发生以下事件：

- USR 寄存器中 OERR 被置位。
- TXR_RXR 寄存器中数据不会丢失。
- RSR 寄存器数据将会被覆盖。
- 若 RIE=1，将会产生中断。

先读取 USR 寄存器再读取 TXR_RXR 寄存器可将 OERR 清零。

噪声干扰 – NF 标志

数据恢复时多次采样可以有效的鉴别出噪声干扰。当检测到数据受到噪声干扰时将会发生以下事件：

- 在 RXIF 上升沿，USR 寄存器中只读标志位 NF 置位。
- 数据从 RSR 寄存器加载到 TXR_RXR 寄存器中。
- 不产生中断，但此位置位发生在 RXIF 置位产生中断的同周期内。

先读取 USR 寄存器再读取 TXR_RXR 寄存器可将 NF 清零。

帧错误 – FERR 标志

若在停止位上检测到 0，USR 寄存器中只读标志 FERR 置位。若选择两位停止位，此两位都必须为高，否则将置位 FERR。此标志位同接收的数据分别记录在 USR 寄存器和 TXR_RXR 寄存器中，此标志位可被任何复位清零。

奇偶校验错误 – PERR 标志

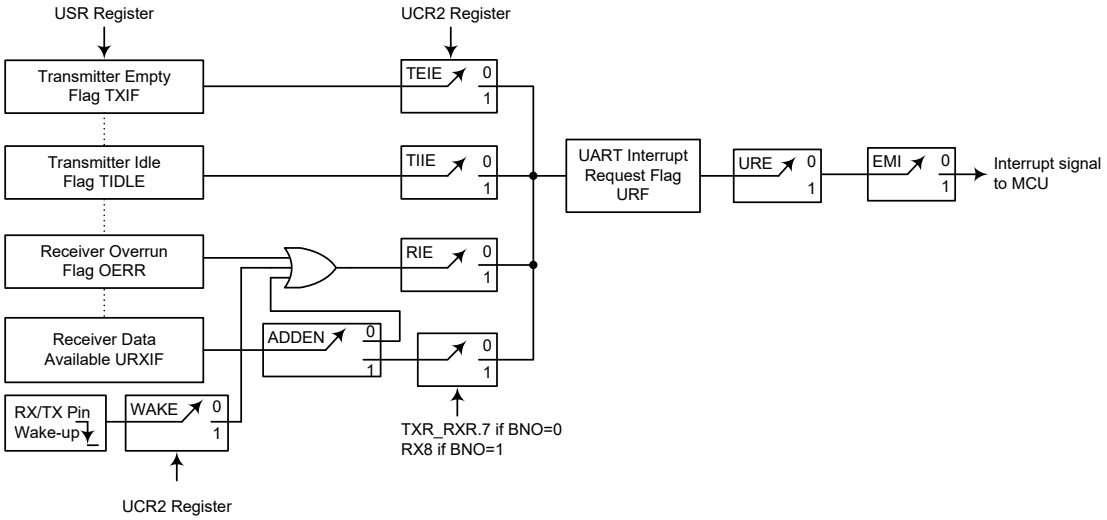
若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 USR 寄存器和 TXR_RXR 寄存器中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 USR 寄存器中的 FERR 和 PERR 错误标志位。

UART 模块中断结构

几个独立的 UART 条件可以产生一个 UART 中断。当条件满足时，会产生一个低脉冲信号。发送寄存器为空、发送器空闲、接收器达到 FIFO 触发字节数、溢出和地址检测和 RX/TX 引脚唤醒都会产生中断。若总中断使能位及相应的中断控制位使能且堆栈未满，程序将会跳转到相应的中断向量执行中断服务程序，而后再返回主程序。其中四种情况，若其 UCR2 寄存器中相应中断允许位被置位，则 USR 寄存器中对应中断标志位将产生 UART 中断。发送器相关的两个中断情况有各自对应的中断允许位，而接收器相关的两个中断情况共用一个中断允许位。这些允许位可用于禁止个别的 UART 中断源。

地址检测也是 UART 的中断源，它没有相应的标志位，若 UCR2 寄存器中 ADDEN=1，当检测到地址将会产生 UART 中断。RX/TX 引脚唤醒也可以产生 UART 中断，它没有相应的标志位，当 UART 时钟源 f_H 关闭且 UCR2 中的 WAKE 和 RIE 位被置位，RX/TX 引脚上有下降沿时会产生 UART 中断。

注意，USR 寄存器标志位为只读状态，软件不能对其进行设置，和其它一些中断一样，在进入相应中断服务程序时也不能清除这些标志位。这些标志位仅在 UART 特定动作发生时才会自动被清除，详细解释见 UART 寄存器章节。整体 UART 中断的使能或除能可由中断控制寄存器中的相关中断使能控制位控制，以决定是否响应或屏蔽 UART 模块的中断请求。



UART 中断结构图

地址检测模式

置位 UCR2 寄存器中的 ADDEN 将启动地址检测模式。若此位为“1”，可产生接收数据有效中断，其请求标志位为 RXIF。若 ADDEN 有效，只有在接收到数据最高位为 1 才会产生中断，注意 URE 和 EMI 中断使能位也要使能才会产生中断。地址的最高位为第 9 位 (BNO=1) 或第 8 位 (BNO=0)，若此位为高，则

接收到的是地址而非数据。只有接收的数据的最后一位为高才会产生中断。若 ADDEN 除能，每接收到一个有效数据便会置位 RXIF，而不用考虑数据的最后一位。地址检测和奇偶校验在功能上相互排斥，若地址检测模式使能，为了确保操作正确，必须将奇偶校验使能位清零以除能奇偶校验。

ADDEN	9th Bit (BNO=1) 8th Bit (BNO=0)	产生 UART 中断
0	0	√
	1	√
1	0	×
	1	√

ADDEN 位功能

UART 模块暂停和唤醒

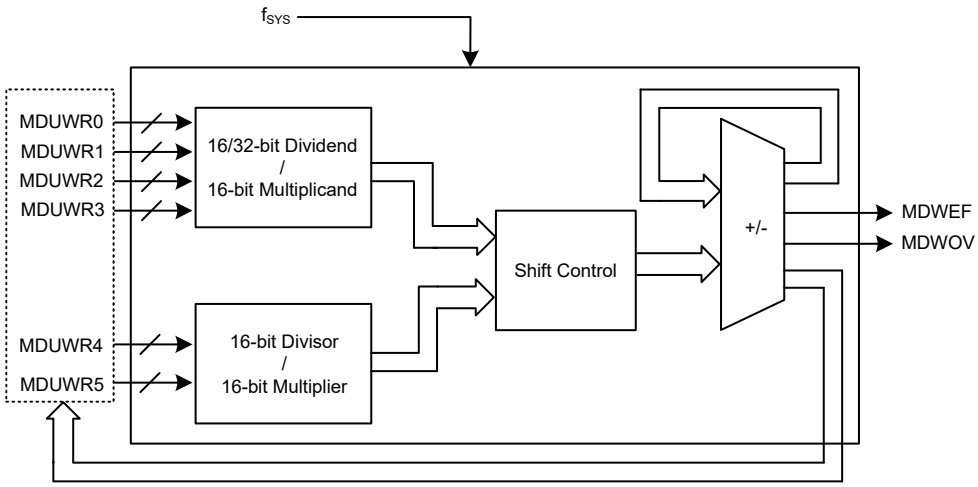
UART 时钟 f_H 关闭后 UART 模块将停止运行。当传送数据时 UART 时钟 f_H 关闭，发送将停止直到 UART 模块时钟再次使能。同样地，当接收数据时单片机进入空闲或休眠模式，数据接收也会停止。当单片机进入空闲或休眠模式，USR、UCR1、UCR2、UCR3、UFCR、TXR_RXR 以及 BRDH 和 BRDL 寄存器都不会受到影响。建议在单片机进入空闲或休眠模式前先确保数据发送或接收已完成。

UART 功能中包括了 RX/TX 引脚的唤醒功能，由 UCR2 寄存器中 WAKE 位控制。当单片机进入空闲或休眠模式且 UART 时钟 f_H 关闭时，若 WAKE 位与 UART 允许位 UARTEN、接收器允许位 RXEN 和接收器中断允许位 RIE 都被置位，则 RX/TX 引脚的下降沿可触发产生 RX/TX 引脚唤醒 UART 的中断。唤醒后系统需延时一段时间才能正常工作，在此期间，RX/TX 引脚上的任何数据将被忽略。

若要唤醒并产生 UART 中断，除了唤醒使能控制位和接收中断使能控制位需置位外，总中断使能位 EMI 和 UART 中断使能控制位 URE 也必须置位；若这三个控制位没有被置位，那么单片机将可以被唤醒但不会产生中断。同样唤醒后系统需一定的延时才能正常工作，然后才会产生 UART 中断。

16 位乘除法单元 – MDU

该单片机内置了 16 位乘除法单元，即 MDU，是 16 位 unsigned 乘法与 32 位/16 位 unsigned 除法器。此乘除法器可取代软件乘除法的运算，节省大量运算时间、程序存储器和数据存储器空间，降低单片机负载，以达到提升整体系统性能。



16-bit MDU 方框图

MDU 寄存器

乘法和除法操作通过特定的步骤实现，即按照特定的顺序对一系列寄存器写值。状态寄存器 MDUWCTRL 用于指示运算操作的状态。六个数据寄存器每个寄存器用于存储不同 MDU 操作中的操作数。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MDUWR0	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR1	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR2	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR3	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR4	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR5	D7	D6	D5	D4	D3	D2	D1	D0
MDUWCTRL	MDWEF	MDWOV	—	—	—	—	—	—

MDU 寄存器列表

- **MDUWRn 寄存器 (n=0~5)**

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**：16-bit MDU 数据寄存器 n

- **MDUWCTRL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MDWEF	MDWOV	—	—	—	—	—	—
R/W	R	R	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7 **MDWEF**：16-bit MDU 错误标志位
0：正常
1：异常
如果在运算过程中 MDUWRn 寄存器被改写或被读取，MDWEF 位由硬件自动置位。当运算完成且 MDWEF 位为 1 时，可通过读取 MDUWCTRL 寄存器将此位清零。

Bit 6 **MDWOV**：16-bit MDU 溢出标志位
0：溢出未发生
1：乘法结果大于 FFFFH 或除数为 0
每完成一次运算，硬件自动更新此位以指示当前运算的情况。

Bit 5~0 未定义，读为“0”

乘除法单元操作

乘除法单元用于乘法运算还是除法运算取决于寄存器 MDUWR0~MDUWR5 的写入顺序。无论是被除数、被乘数、除数或乘数的低字节数据，必须在其高字节数据之前先写入对应的 MDU 数据寄存器中。所有的 MDU 操作都在已按照正确顺序写入 MDUWRn 寄存器且 MDUWR5 寄存器被写入后开始执行。不一定要连续往 MDU 数据寄存器中写数据，但必须要按照正确的顺序写入。因此，只要不影响写操作，在正确的写入顺序中间允许插入非写入 MDUWRn 的指令或中断。写入顺序与 MDU 运算的对应关系如下：

- 32-bit/16-bit 除法运算：依序从 MDUWR0 写到 MDUWR5
- 16-bit/16-bit 除法运算：依序写入 MDUWR0、MDUWR1、MDUWR4 和 MDUWR5，跳过 MDUWR2 和 MDUWR3 不写
- 16-bit×16-bit 乘法运算：依序写入 MDUWR0、MDUWR4、MDUWR1 和 MDUWR5，跳过 MDUWR2 和 MDUWR3 不写

写顺序确定后，MDU 开始执行对应的运算。不同的 MDU 运算所需的计算时间不同。在执行运算操作过程中，禁止对六个 MDU 数据寄存器执行读或写动作。当每次运算操作完成后，应通过 MDUWCTRL 寄存器确认该运算是否正确。若运算正确，可按照特定顺序从对应的 MDU 数据寄存器中读取运算结果。MDU 运算及其所需的计算时间如下所示：

- 32-bit/16-bit 除法运算：$17 \times t_{SYS}$
- 16-bit/16-bit 除法运算：$9 \times t_{SYS}$
- 16-bit×16-bit 乘法运算：$11 \times t_{SYS}$

运算操作完成后，结果存储在对应的 MDU 数据寄存器中，且需按照特定的顺序读取。不一定要连续从 MDU 数据寄存器中读取结果，但必须按照正确的顺序读取。因此，只要不影响读操作，在正确的读取顺序中间允许插入非读取 MDUWRn 的指令或中断。运算结果读取顺序与 MDU 运算的对应关系如下：

- 32-bit/16-bit 除法运算：依序从 MDUWR0 到 MDUWR3 读取商数，依序从 MDURW4 和 MDUWR5 读取余数
- 16-bit/16-bit 除法运算：依序从 MDUWR0 和 MDUWR1 读取商数，依序从 MDUWR4 和 MDUWR5 读取余数

- 16-bit×16-bit 乘法运算：依序从 MDUWR0 到 MDUWR3 读取乘积

MDU 读 / 写顺序以及计算时间的注意事项总结如下表所示。注意，在 MDU 操作未完成之前，单片机不可进入空闲或休眠模式，否则 MDU 操作失败。

运算 / 事项	32-bit / 16-bit 除法	16-bit / 16-bit 除法	16-bit × 16-bit 乘法
写顺序 最先写 ↓ ↓ ↓ ↓ 最后写	被除数字节 0 写入 MDUWR0 被除数字节 1 写入 MDUWR1 被除数字节 2 写入 MDUWR2 被除数字节 3 写入 MDUWR3 除数字节 0 写入 MDUWR4 除数字节 1 写入 MDUWR5	被除数字节 0 写入 MDUWR0 被除数字节 1 写入 MDUWR1 除数字节 0 写入 MDUWR4 除数字节 1 写入 MDUWR5	被乘数字节 0 写入 MDUWR0 乘数字节 0 写入 MDUWR4 被乘数字节 1 写入 MDUWR1 乘数字节 1 写入 MDUWR5
计算时间	17 × t_{SYS}	9 × t_{SYS}	11 × t_{SYS}
读顺序 最先读 ↓ ↓ ↓ ↓ 最后读	从 MDUWR0 读取商数字节 0 从 MDUWR1 读取商数字节 1 从 MDUWR2 读取商数字节 2 从 MDUWR3 读取商数字节 3 从 MDUWR4 读取余数字节 0 从 MDUWR5 读取余数字节 1	从 MDUWR0 读取商数字节 0 从 MDUWR1 读取商数字节 1 从 MDUWR4 读取余数字节 0 从 MDUWR5 读取余数字节 1	从 MDUWR0 读取乘积字节 0 从 MDUWR1 读取乘积字节 1 从 MDUWR2 读取乘积字节 2 从 MDUWR3 读取乘积字节 3

MDU 操作总结

低电压检测 – LVD

此单片机都具有低电压检测功能，即 LVD。该功能使能后用于监测电源电压 V_{DD}，若电源电压低于一定值可提供一个警告信号。此功能在电池类产品中非常有用，在电池电压较低时产生警告信号。低电压检测也可产生中断信号。

LVD 寄存器

低电压检测功能由 LVDC 寄存器控制。VLVD2~VLVD0 位用于选择三个固定的电压参考点。LVDO 位被置位时低电压情况发生，若 LVDO 位为低表明 V_{DD} 电压工作在当前所设置低电压水平值之上。LVDEN 位用于控制低电压检测功能的开启 / 关闭，设置此位为高使能此功能，反之，关闭内部低电压检测电路。低电压检测会有一定的功耗，在不使用时可考虑关闭此功能，此举在功耗要求严格的电池供电应用中值得考虑。

- **LVDC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6　未定义，读为“0”

Bit 5　**LVDO**：LVD 输出标志位
0：未检测到低电压
1：检测到低电压

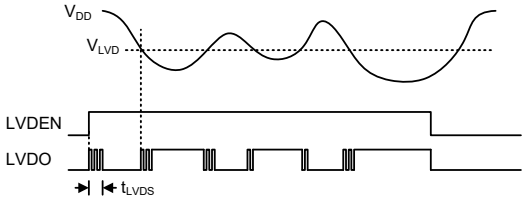
Bit 4　**LVDEN**：低电压检测控制位
0：除能
1：使能

Bit 3　未定义，读为“0”

Bit 2~0 **VLVD2~VLVD0**：LVD 电压选择位
000：保留
001：保留
010：保留
011：保留
100：保留
101：3.3V
110：3.6V
111：4.0V

LVD 操作

低电压检测功能的工作原理是比较电源电压 V_{DD} 与 LVDC 寄存器定义的预置电压值。其设置的范围为 3.3V~4.0V。当电源电压 V_{DD} 低于预置电压值时，LVDO 位被置为高，表明低电压产生。当单片机进入休眠模式时，即使 LVDEN 位为高，低电压检测器也会自动除能。低电压检测器使能后，读取 LVDO 位前，电路稳定需要一定的延时 t_{LVDS}。注意，V_{DD} 电压可能上升或下降比较缓慢，在 V_{LVD} 电压值附近时，LVDO 位可能有多种变化。



LVD 操作

低电压检测器也有自己的中断功能，它是除了轮询 LVDO 位之外的另一种检测低电压的方法。中断条件发生置位 LVDO 并延时 t_{LVD} 后，中断产生。若 V_{DD} 降至小于 LVD 预置电压值时，中断请求标志位 LVDF 将被置位，中断产生，单片机将从空闲模式中被唤醒。若不要求低电压检测的唤醒功能使能，在单片机进入空闲模式前应将 LVDF 标志置为高。

中断

中断是单片机一个重要功能。当外部事件或内部功能如定时器模块或 A/D 转换器有效，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。此单片机提供多个外部中断和内部中断功能，外部中断由 H1、H2、H3 和 INT0 引脚动作产生，而内部中断由各种内部功能产生，如定时器模块、比较器 0、16 位 CAPTM、时基、PWM、LVD、EEPROM、UART、A/D 转换器和霍尔传感器等等。

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。寄存器总的分为三类。第一类是 HINTEG 和 INTEG 寄存器，用于设置部分中断的中断边沿触发类型；第二类是 INTC0~INTC3 寄存器，用于设置基本的中断；第三类是 MFI0~MFI6 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	请求标志位	注释
总中断	EMI	—	—
霍尔传感器中断	HALAE	HALAF	霍尔噪声滤波器输出
	HALBE	HALBF	
	HALCE	HALCF	
外部中断 0	INT0E	INT0F	—
比较器 0	C0E	C0F	—
低电压检测功能	LVDE	LVDF	—
PROTECTOC	PROTOCE	PROTOCF	ProtectOC
时基	TBE	TBF	—
多功能中断	MFnE	MFnF	n=0~6
A/D 转换器	ADE	ADF	—
	ALIME	ALIMF	—
16 位捕捉定时器模块	CAPOE	CAPOF	—
	CAPCE	CAPCF	—
PWM 功能	PWMDnE	PWMDnF	n=0~2
	PWMPE	PWMPF	—
定时器模块	PTMnPE	PTMnPF	n=0~3
	PTMnAE	PTMnAF	
UART	URE	URF	—
EEPROM	DEE	DEF	—

中断相关位命名模式

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HINTEG	—	HSEL	INTCS1	INTCS0	INTBS1	INTBS0	INTAS1	INTAS0
INTEG	—	—	—	—	—	—	INT0S1	INT0S0
INTC0	—	C0F	INT0F	MF0F	C0E	INT0E	MF0E	EMI
INTC1	PROTOCF	LVDF	MF1F	D4	PROTOCE	LVDE	MF1E	D0
INTC2	MF5F	MF4F	MF3F	MF2F	MF5E	MF4E	MF3E	MF2E
INTC3	TBF	MF6F	DEF	URF	TBE	MF6E	DEE	URE
MFI0	—	HALCF	HALBF	HALAF	—	HALCE	HALBE	HALAE
MFI1	CAPCF	CAPOF	ALIMF	ADF	CAPCE	CAPOE	ALIME	ADE
MFI2	PWMPF	PWMD2F	PWMD1F	PWMD0F	PWMPE	PWMD2E	PWMD1E	PWMD0E
MFI3	—	—	PTM2AF	PTM2PF	—	—	PTM2AE	PTM2PE
MFI4	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
MFI5	—	—	PTM1AF	PTM1PF	—	—	PTM1AE	PTM1PE
MFI6	—	—	PTM3AF	PTM3PF	—	—	PTM3AE	PTM3PE

中断寄存器列表

- **HINTEG 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	HSEL	INTCS1	INTCS0	INTBS1	INTBS0	INTAS1	INTAS0
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 **HSEL**：HA/HB/HC 来源选择
0：H1/H2/H3
1：CMP1/CMP2/CMP3 输出

Bit 5~4 **INTCS1~INTCS0**：FHC 触发 INTC 中断之边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双边沿触发

Bit 3~2 **INTBS1~INTBS0**：FHB 触发 INTB 中断之边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双边沿触发

Bit 1~0 **INTAS1~INTAS0**：FHA 触发 INTA 中断之边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双边沿触发

- **INTEG 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INT0S1	INT0S0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **INT0S1~INT0S0**：INT0 脚中断触发边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双边沿触发

- **INTC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	C0F	INT0F	MF0F	C0E	INT0E	MF0E	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 **C0F**：比较器 0 中断请求标志位
0：无请求
1：中断请求

Bit 5 **INT0F**：外部中断 0 请求标志位
0：无请求
1：中断请求

Bit 4 **MF0F**：多功能中断 0 请求标志位
0：无请求
1：中断请求

Bit 3 **C0E**：比较器 0 中断控制位
0：除能
1：使能

Bit 2 **INT0E**：外部中断 0 中断控制位
0：除能
1：使能

Bit 1 **MF0E**：多功能中断 0 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	PROTOCF	LVDF	MF1F	D4	PROTOCE	LVDE	MF1E	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 **PROTOCF**：PROTECTOC 中断请求标志位
0：无请求
1：中断请求

Bit 6 **LVDF**：LVD 中断请求标志位
0：无请求
1：中断请求

Bit 5 **MF1F**：多功能中断 1 请求标志位
0：无请求
1：中断请求

Bit 4 **D4**：保留，必须固定为 0

Bit 3 **PROTOCE**：PROTECTOC 中断控制位
0：除能
1：使能

Bit 2 **LVDE**：LVD 中断控制位
0：除能
1：使能

Bit 1 **MF1E**：多功能中断 1 控制位
0：除能
1：使能

Bit 0 **D0**：保留，必须固定为 0

- **INTC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MF5F	MF4F	MF3F	MF2F	MF5E	MF4E	MF3E	MF2E
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 **MF5F**：多功能中断 5 请求标志位
0：无请求
1：中断请求

Bit 6 **MF4F**：多功能中断 4 请求标志位
0：无请求
1：中断请求

Bit 5 **MF3F**：多功能中断 3 请求标志位
0：无请求
1：中断请求

Bit 4 **MF2F**：多功能中断 2 请求标志位
0：无请求
1：中断请求

Bit 3 **MF5E**：多功能中断 5 控制位
0：除能
1：使能

Bit 2 **MF4E**：多功能中断 4 控制位
0：除能
1：使能

Bit 1 **MF3E**：多功能中断 3 控制位
0：除能
1：使能

Bit 0 **MF2E**：多功能中断 2 控制位
0：除能
1：使能

- **INTC3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TBF	MF6F	DEF	URF	TBE	MF6E	DEE	URE
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 **TBF**：时基中断请求标志位
0：无请求
1：中断请求

Bit 6 **MF6F**：多功能中断 6 请求标志位
0：无请求
1：中断请求

Bit 5 **DEF**：数据 EEPROM 中断请求标志位
0：无请求
1：中断请求

Bit 4 **URF**：UART 中断请求标志位
0：无请求
1：中断请求

Bit 3 **TBE**：时基中断控制位
0：除能
1：使能

Bit 2 **MF6E**：多功能中断 6 控制位
0：除能
1：使能

Bit 1 **DEE**：数据 EEPROM 中断控制位
0：除能
1：使能

Bit 0 **URE**：UART 中断控制位
0：除能
1：使能

- **MFI0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	HALCF	HALBF	HALAF	—	HALCE	HALBE	HALAE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 未定义，读为“0”

Bit 6 **HALCF**：霍尔传感器 C 中断 (INTC) 请求位
0：无请求
1：中断请求

Bit 5 **HALBF**：霍尔传感器 B 中断 (INTB) 请求位
0：无请求
1：中断请求

Bit 4 **HALAF**：霍尔传感器 A 中断 (INTA) 请求位
0：无请求
1：中断请求

Bit 3 未定义，读为“0”

Bit 2 **HALCE**：霍尔传感器 C 中断 (INTC) 控制位
0：除能
1：使能

Bit 1 **HALBE**：霍尔传感器 B 中断 (INTB) 控制位
0：除能
1：使能

Bit 0 **HALAE**：霍尔传感器 A 中断 (INTA) 控制位
0：除能
1：使能

- **MFI1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CAPCF	CAPOF	ALIMF	ADF	CAPCE	CAPOE	ALIME	ADE
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 **CAPCF**：CAPTM 比较匹配中断请求标志位
0：无请求
1：中断请求

Bit 6 **CAPOF**：CAPTM 捕捉溢出中断请求标志位
0：无请求
1：中断请求

Bit 5 **ALIMF**：A/D 转换值比较结果中断请求标志位
0：无请求
1：中断请求

Bit 4 **ADF**：A/D 转换完成中断请求标志位
0：无请求
1：中断请求

Bit 3 **CAPCE**：CAPTM 比较匹配中断控制位
0：除能
1：使能

Bit 2 **CAPOE**：CAPTM 捕捉溢出中断控制位
0：除能
1：使能

Bit 1 **ALIME**：A/D 转换值比较结果中断控制位
0：除能
1：使能

Bit 0 **ADE**：A/D 转换完成中断控制位
0：除能
1：使能

- **MFI2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PWMPF	PWMD2F	PWMD1F	PWMD0F	PWMPE	PWMD2E	PWMD1E	PWMD0E
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 **PWMPF**：PWM 周期匹配中断请求标志位
0：无请求
1：中断请求

Bit 6 **PWMD2F**：PWM2 占空比匹配中断请求标志位
0：无请求
1：中断请求

Bit 5 **PWMD1F**：PWM1 占空比匹配中断请求标志位
0：无请求
1：中断请求

Bit 4 **PWMD0F**：PWM0 占空比匹配中断请求标志位
0：无请求
1：中断请求

Bit 3 **PWMPE**：PWM 周期匹配中断控制位
0：除能
1：使能

Bit 2 **PWMD2E**：PWM2 占空比匹配中断控制位
0：除能
1：使能

Bit 1 **PWMD1E**：PWM1 占空比匹配中断控制位
0：除能
1：使能

Bit 0 **PWMD0E**：PWM0 占空比匹配中断控制位
0：除能
1：使能

- **MFI3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM2AF	PTM2PF	—	—	PTM2AE	PTM2PE
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 **PTM2AF**：PTM2 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4 **PTM2PF**：PTM2 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 **PTM2AE**：PTM2 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0 **PTM2PE**：PTM2 比较器 P 匹配中断控制位
0：除能
1：使能

- **MFI4 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
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 **PTM0AF**：PTM0 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4 **PTM0PF**：PTM0 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 **PTM0AE**：PTM0 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0 **PTM0PE**：PTM0 比较器 P 匹配中断控制位
0：除能
1：使能

- **MFI5 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM1AF	PTM1PF	—	—	PTM1AE	PTM1PE
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 **PTM1AF**：PTM1 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4 **PTM1PF**：PTM1 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 **PTM1AE**：PTM1 比较器 A 匹配中断请求标志位
0：除能
1：使能

Bit 0 **PTM1PE**：PTM1 比较器 P 匹配中断请求标志位
0：除能
1：使能

- **MFI6 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM3AF	PTM3PF	—	—	PTM3AE	PTM3PE
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 **PTM3AF**：PTM3 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4 **PTM3PF**：PTM3 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 **PTM3AE**：PTM3 比较器 A 匹配中断请求标志位
0：除能
1：使能

Bit 0 **PTM3PE**：PTM3 比较器 P 匹配中断请求标志位
0：除能
1：使能

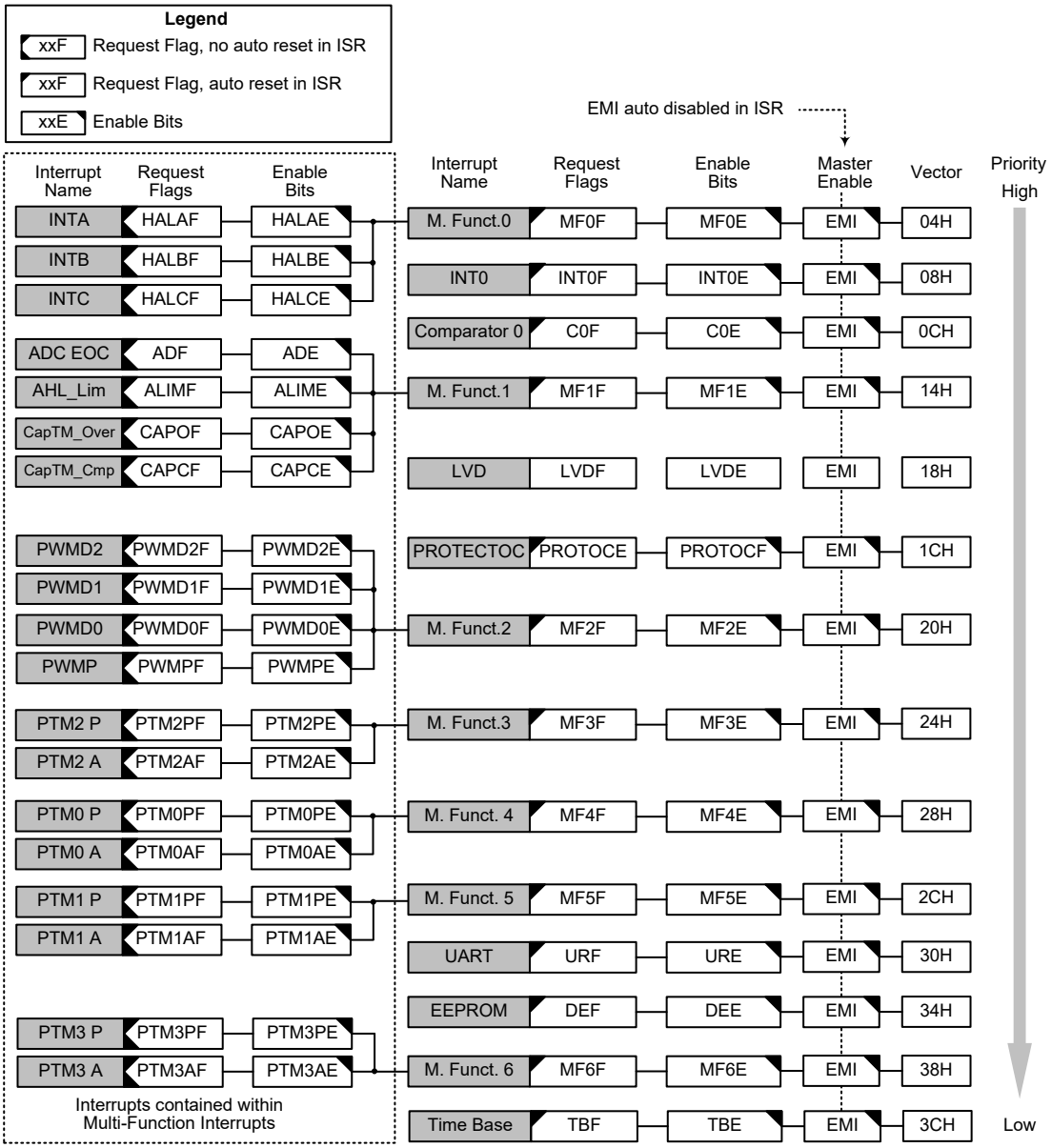
中断操作

若中断事件的条件产生，如一个 TM 比较器 P、比较器 A 匹配或 A/D 转换结束等等，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，而有些中断则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。所有中断请求标志位在被置起时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中断请求标志置起。



中断结构

霍尔传感器中断

霍尔传感器中断 INTA/INTB/INTC 属于多功能中断，由霍尔传感器噪声滤波器输出信号 FHA、FHB 和 FHC 控制。每当霍尔传感器滤波输出信号 FHA、FHB 或 FHC 发生预设的边沿变化时 (上升沿，下降沿或双边沿，通过 HINTEG 寄存器选择)，中断请求标志 HALAF、HALBF 或 HALCF 被置位，多功能中断 0 请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI、霍尔传感器总中断使能位 MF0E 以及对应的霍尔传感器中断控制位 HALAE、HALBE 或 HALCE 需先被置位。此外，必须使用 HINTEG 寄存器选择触发沿类型。若霍尔噪声滤波器输入源选择来自外部输入引脚 H1、H2 和 H3 时，由于它们和普通 I/O 口共用，在霍尔传感器中断使能之前必须先通过对应引脚共用控制位选择外部霍尔传感器输入引脚。当中断使能，堆栈未满并且霍尔传感器中断触发

边沿发生时，将调用对应多功能中断向量子程序。当响应霍尔传感器中断服务子程序时，EMI 位会被自动清零以除能其它中断，多功能中断 0 请求标志位 MF0F 也可自动清零，但霍尔传感器中断请求标志位 HALAF，HALBF 和 HALCF 标志需在应用程序中手动清零。

寄存器 HINTEG 被用来选择有效的边沿类型，来触发霍尔传感器中断。可以选择上升沿还是下降沿或双沿触发都产生霍尔传感器中断。注意 HINTEG 也可以用来除能霍尔传感器中断功能。

外部中断 0

通过 INT0 引脚上的信号变化可控制外部中断 0。当 INT0 引脚发生边沿变化时 (触发边沿类型由 INTEG 寄存器中的触发沿选择位决定)，对应中断请求标志位 INT0F 被置位，外部中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和外部中断 0 使能位 INT0E 需先被置位。由于外部中断引脚和普通 I/O 口共用，在使能外部引脚中断功能前需通过引脚共用寄存器选择外部中断引脚功能。此外必须通过设置对应的端口控制寄存器，将该引脚设置为输入口。当中断使能，堆栈未满并且外部中断引脚状态改变，将调用外部中断向量子程序。当响应外部中断服务子程序时，外部中断 0 请求标志位 INT0F 会自动复位且 EMI 位会被清零以除能其它中断。注意，即使此引脚被用作外部中断输入，其上拉电阻仍保持有效。

寄存器 INTEG 被用来选择有效的边沿类型，来触发外部中断 0。可以选择上升沿还是下降沿或双沿触发都产生外部中断 0。注意 INTEG 也可以用来除能外部中断 0 功能。

比较器 0 中断

比较器 0 中断由内部比较器 0 控制。当比较器输出发生预设的边沿变化时 (触发边沿类型由 OPOMS 寄存器中的 CMP0_EG1~CMP0_EG0 位选择)，比较器 0 中断请求标志 C0F 被置位，比较器 0 中断请求产生。若要跳转到相应中断向量地址，总中断使能位 EMI 和比较器 0 中断使能位 C0E 需先被置位。当中断使能，堆栈未满并且比较器输出发生中断触发边沿时，将调用比较器中断向量子程序。当响应中断服务子程序时，比较器 0 中断请求标志位会自动复位且 EMI 位会被清零以除能其它中断。

PROTECTOC 中断

PROTECTOC 中断由电机保护电路控制。当产生一个 PROTECTOC 信号，PROTECTOC 中断请求标志 PROTOCF 被置位，PROTECTOC 中断请求产生。若要跳转到相应中断向量地址，总中断使能位 EMI 和 PROTECTOC 中断使能位 PROTOCE 需先被置位。当中断使能，堆栈未满并且产生一个 PROTECTOC 信号时，将调用 PROTECTOC 中断向量子程序。当响应中断服务子程序时，PROTECTOC 中断请求标志位 PROTOCF 会自动复位且 EMI 位会被清零以除能其它中断。

多功能中断

此单片机中有多个多功能中断，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，分别为霍尔传感器中断，PWM 周期和占空比匹配中断，TM 中断，A/D 转换器中断和 CAPTM 中断。

当多功能中断中任何一个中断源请求发生，其所在的多功能中断请求标志位 MFnF 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不会自动复位，必须由应用程序清零。

A/D 转换器中断

A/D 转换器中断属于多功能中断，有两种中断源。第一种是 A/D 转换值比较结果中断，由 A/D 转换器转换值 (SADOH/SADOL) 与上下限值 (ADHVDH/ADHVDL 和 ADLVDH/ADLVDL) 进行比较控制，当转换结果满足由 ADCR2 寄存器中的 ADCHVE/ADCLVE 设置的中断条件时，发出中断请求。第二种是 A/D 转换完成中断由 A/D 转换动作的结束来控制。当 A/D 转换过程完成时，可发出中断请求。

当总中断控制位 EMI，相应多功能中断使能位和 A/D 中断使能位 ADE 或 ALIME 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 A/D 转换动作完成或在完成后产生比较结果时，将调用相关多功能中断向量子程序。当响应相关中断服务子程序时，EMI 位会被清零以除能其它中断，多功能中断请求标志也可自动清除。而 ADF 和 ALIMF 标志位需在应用程序中手动清零。

CAPTM 中断

捕捉定时器模块中断属于多功能中断，有两个中断，CapTM_Over 和 CapTM_Cmp。这两个中断分别是捕捉溢出中断 CapTM_Over 和比较匹配中断 CapTM_Cmp。当 CAPTM 捕捉溢出或比较匹配时，相关中断请求标志位 CAPOF 或 CAPCF 被置位，中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI、相应的多功能中断使能位和中断使能位 CAPOE 或 CAPCE 需先被置位。当中断使能，堆栈未满并且发生捕捉溢出或比较匹配时，将调用相应的中断向量子程序。当响应 CAPTM 模块中断服务子程序时，EMI 位会被清零以除能其它中断，多功能中断请求标志也可自动清零，但 CAPOF 或 CAPCF 标志位需在应用程序中手动清除。

PWM 模块中断

PWM 模块包含 4 个中断，一个 PWM 周期匹配中断即 PWMP 中断和 3 个 PWM 占空比匹配中断即 PWMDn (n=0~2) 中断。PWMP 和 PWMD0~PWMD2 中断属于多功能中断 2。当 PWM 占空比匹配或 PWM 周期匹配时，中断请求标志位 PWMDnF 或 PWMPF 以及对应多功能中断请求标志位被置位，PWM 中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI、相应多功能中断使能位、PWM 周期或占空比匹配中断使能位，PWMPE 或 PWMDnE，需先被置位。当中断使能，堆栈未满并且 PWM 占空比匹配或 PWM 周期匹配时，将调用相关多功能中断向量子程序。当响应 PWM 中断服务子程序时，EMI 位会被清零以除能其它中断，多功能中断请求标志也可自动清除，但 PWMDnF 或 PWMPF 标志需在应用程序中手动清零。

TM 中断

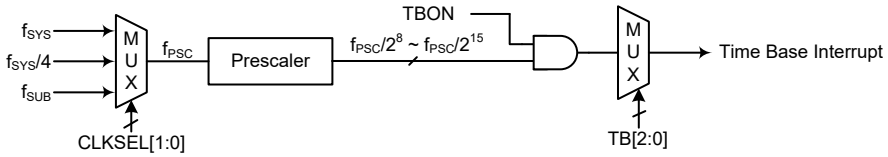
每个周期型 TM 有两个中断，一个来自比较器 A 匹配，一个来自比较器 P 匹配。所有的 TM 中断都属于多功能中断。周期型 TM 的两个中断请求标志位分别为 PTMnPF 和 PTMnAF，及两个使能位分别为 PTMnPE 和 PTMnAE。当 TM 比较器 P 或比较器 A 匹配情况发生时，任意 TM 中断请求标志和对应中断多功能请求标志被置位，TM 中断请求产生。

若要程序跳转到相应中断向量地址，总中断控制位 EMI、相应 TM 中断使能位和相关多功能中断使能位需先被置位。当中断使能，堆栈未满且 TM 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 TM 中断响应，EMI 将被自动清零以除能其它中断，相关多功能中断标志也可自动清除，但 TM 中断请求标志需在应用程序中手动清除。

时基中断

时基中断提供一个固定周期的中断信号，由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，允许程序跳转到中断向量地址。当中断使能，堆栈未满且时基溢出时，将调用中断向量子程序。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 位会被自动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分别通过 PSCR 寄存器的 CLKSEL1~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 f_{PSC} 选择
00：f_{SYS}
01：$f_{SYS}/4$
1x：f_{SUB}

- **TBC 寄存器**

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

Bit 7　**TBON**：时基控制位
0：除能
1：使能

Bit 6~3　未定义，读为“0”

Bit 2~0　**TB2~TB0**：时基溢出周期选择
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：$2^{15}/f_{PSC}$

LVD 中断

当低电压检测功能检测到一个低电压时，LVD 中断请求标志 LVDF 置位，LVD 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 和低电压中断使能位 LVDE 需先被置位。当中断使能，堆栈未满且低电压条件发生时，可跳转至相关中断向量子程序中执行。当低电压中断响应，EMI 将被自动清零以除能其它中断，同时其请求标志位 LVDF 也会被自动清零。

UART 中断

UART 传输中断由几种 UART 传输条件来控制。当发送器为空、发送器空闲、接收器数据有效、接收器溢出、地址检测或 RX/TX 引脚唤醒发生，UART 中断请求标志 URF 被置位，UART 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 UART 中断使能位 URE 需先被置位。当中断使能，堆栈未满且以上任何一种情况发生时，将调用 UART 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 URF 会自动复位且 EMI 位会被清零以除能其它中断。然而 USR 寄存器里的标志位只有在对 UART 执行特定动作时才会被清零，详细参考 UART 章节。

EEPROM 中断

当擦或写周期结束，EEPROM 中断请求标志 DEF 被置位，EEPROM 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 EEPROM 中断使能位 DEE 需先被置位。当中断使能，堆栈未满且 EEPROM 擦或写周期结束时，可跳转至相应的 EEPROM 中断向量子程序中执行。当响应中断服务子程序时，相应的中断请求标志位 DEF 会自动复位且 EMI 位会被自动清零以除能其它中断。

中断唤醒功能

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

编程注意事项

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

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。

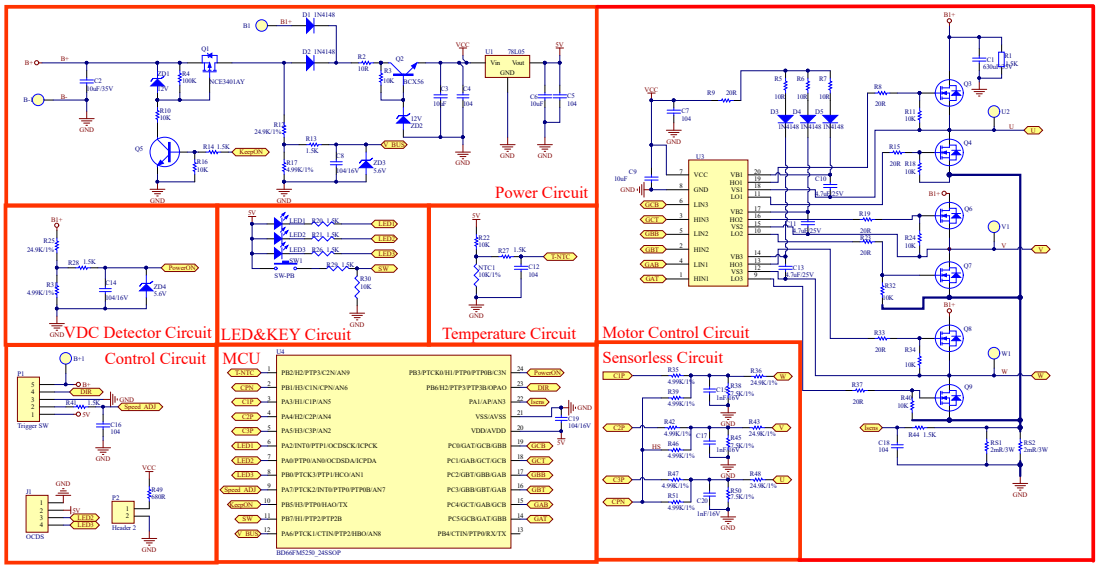
建议在中断服务子程序中不要使用“CALL 子程序”指令。中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

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中断请求标志置为高。

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



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

当要操作的数据存储器位于数据存储器 Sector 0 时，下表说明了与数据存储器存取有关的指令。

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

助记符	说明	指令周期	影响标志位
移位			
RRA [m]	数据存储器右移一位，结果放入 ACC	1	无
RR [m]	数据存储器右移一位，结果放入数据存储器	1[注]	无
RRCA [m]	带进位将数据存储器右移一位，结果放入 ACC	1	C
RRC [m]	带进位将数据存储器右移一位，结果放入数据存储器	1[注]	C
RLA [m]	数据存储器左移一位，结果放入 ACC	1	无
RL [m]	数据存储器左移一位，结果放入数据存储器	1[注]	无
RLCA [m]	带进位将数据存储器左移一位，结果放入 ACC	1	C
RLC [m]	带进位将数据存储器左移一位，结果放入数据存储器	1[注]	C
数据传送			
MOV A,[m]	将数据存储器送至 ACC	1	无
MOV [m],A	将 ACC 送至数据存储器	1[注]	无
MOV A, x	将立即数送至 ACC	1	无
位运算			
CLR [m].i	清除数据存储器的位	1[注]	无
SET [m].i	置位数据存储器的位	1[注]	无
转移			
JMP addr	无条件跳转	2	无
SZ [m]	如果数据存储器为零，则跳过下一条指令	1[注]	无
SZA [m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	1[注]	无
SNZ [m]	如果数据存储器不为零，则跳过下一条指令	1[注]	无
SZ [m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	1[注]	无
SNZ [m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	1[注]	无
SIZ [m]	递增数据存储器，如果结果为零，则跳过下一条指令	1[注]	无
SDZ [m]	递减数据存储器，如果结果为零，则跳过下一条指令	1[注]	无
SIZA [m]	递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1[注]	无
SDZA [m]	递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1[注]	无
CALL addr	子程序调用	2	无
RET	从子程序返回	2	无
RET A, x	从子程序返回，并将立即数放入 ACC	2	无
RETI	从中断返回	2	无
查表			
TABRD [m]	读取特定页的 ROM 内容，并送至数据存储器和 TBLH	2[注]	无
TABRDL [m]	读取最后页的 ROM 内容，并送至数据存储器和 TBLH	2[注]	无
ITABRD [m]	读表指针 TBLP 自加，读取特定页的 ROM 内容，并送至数据存储器和 TBLH	2[注]	无
ITABRDL [m]	读表指针 TBLP 自加，读取最后页的 ROM 内容，并送至数据存储器和 TBLH	2[注]	无
其它指令			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1[注]	无
SET [m]	置位数据存储器	1[注]	无

助记符	说明	指令周期	影响标志位
CLR WDT	清除看门狗定时器	1	TO, PDF
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。

扩展指令集

扩展指令用来提供更大范围的数据存储器寻址。当被存取的数据存储器位于 Sector 0 之外的任何数据存储器 Sector，扩展指令可直接存取数据存储器而无需使用间接寻址，此举不仅可节省 Flash 存储器空间的使用，同时可提高 CPU 执行效率。

助记符	说明	指令周期	影响标志位
算术运算			
LADD A,[m]	ACC 与数据存储器相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	ACC 与数据存储器相加，结果放入数据存储器	2注	Z, C, AC, OV, SC
LADC A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADCM A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	2注	Z, C, AC, OV, SC
LSUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	ACC 与数据存储器相减，结果放入数据存储器	2注	Z, C, AC, OV, SC, CZ
LSBC A,[m]	ACC 与数据存储器、进位标志相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	2注	Z, C, AC, OV, SC, CZ
LDAA [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	2注	C
逻辑运算			
LAND A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	2	Z
LOR A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	2	Z
LXOR A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	2	Z
LANDM A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	2注	Z
LORM A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	2注	Z
LXORM A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	2注	Z
LCPL [m]	对数据存储器取反，结果放入数据存储器	2注	Z
LCPLA [m]	对数据存储器取反，结果放入 ACC	2	Z
递增和递减			
LINCA [m]	递增数据存储器，结果放入 ACC	2	Z
LINC [m]	递增数据存储器，结果放入数据存储器	2注	Z
LDECA [m]	递减数据存储器，结果放入 ACC	2	Z
LDEC [m]	递减数据存储器，结果放入数据存储器	2注	Z
移位			
LRRA [m]	数据存储器右移一位，结果放入 ACC	2	无
LRR [m]	数据存储器右移一位，结果放入数据存储器	2注	无
LRRCA [m]	带进位将数据存储器右移一位，结果放入 ACC	2	C
LRRC [m]	带进位将数据存储器右移一位，结果放入数据存储器	2注	C
LRLA [m]	数据存储器左移一位，结果放入 ACC	2	无
LRL [m]	数据存储器左移一位，结果放入数据存储器	2注	无
LRLCA [m]	带进位将数据存储器左移一位，结果放入 ACC	2	C
LRLC [m]	带进位将数据存储器左移一位，结果放入数据存储器	2注	C
数据传送			
LMOV A,[m]	将数据存储器送至 ACC	2	无
LMOV [m],A	将 ACC 送至数据存储器	2注	无

助记符		说明	指令周期	影响标志位
位运算				
LCLR	[m].i	清除数据存储器的位	2 注	无
LSET	[m].i	置位数据存储器的位	2 注	无
转移				
LSZ	[m]	如果数据存储器为零，则跳过下一条指令	2 注	无
LSZA	[m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	2 注	无
LSNZ	[m]	如果数据存储器不为零，则跳过下一条指令	2 注	无
LSZ	[m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	2 注	无
LSNZ	[m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	2 注	无
LSIZ	[m]	递增数据存储器，如果结果为零，则跳过下一条指令	2 注	无
LSDZ	[m]	递减数据存储器，如果结果为零，则跳过下一条指令	2 注	无
LSIZA	[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	2 注	无
LSDZA	[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	2 注	无
查表				
LTABRD	[m]	读取特定页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LTABRDL	[m]	读取最后页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LITABRD	[m]	读表指针 TBLP 自加，读取特定页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LITABRDL	[m]	读表指针 TBLP 自加，读取最后页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
其它指令				
LCLR	[m]	清除数据存储器	2 注	无
LSET	[m]	置位数据存储器	2 注	无
LSWAP	[m]	交换数据存储器的高低字节，结果放入数据存储器	2 注	无
LSWAPA	[m]	交换数据存储器的高低字节，结果放入 ACC	2	无

注：1. 对扩展跳转指令而言，如果比较的结果牵涉到跳转即需 3 个周期，如果没有发生跳转，则只需两个周期。
2. 任何扩展指令若要改变 PCL 的内容将需要 3 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、SC

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、SC

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、SC

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、SC

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、SC

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、SC、CZ

SBC A, x Subtract immediate data from ACC with Carry
指令说明 将累加器减去立即数以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C、SC、CZ

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

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

SDZA [m] 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

SNZ [m] Skip if Data Memory is not 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。判断指定存储器，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]≠0，跳过下一条指令执行
影响标志位 无

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

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

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

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

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

SZ [m]	Skip if Data Memory is 0
指令说明	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) to TBLH and Data Memory
指令说明	将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无

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

ITABRD [m]	Increment table pointer low byte first and read table (specific page) to TBLH and data memory
指令说明	自加表格指针低字节 TBLP，将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无

ITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and data memory
指令说明	自加表格指针低字节 TBLP，将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。
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

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

LADC 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、SC

LADCM 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、SC

LADD A, [m] Add Data Memory to ACC
指令说明 将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、SC

LADDM 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、SC

LAND 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

LANDM 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

LCLR [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

LCLR [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

LCPL [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

LCPLA [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，结果被存放回累加器且数据寄存器的内容保持不变。
功能表示 ACC←$[\overline{m}]$
影响标志位 Z

LDAA [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 调整就执行对低四位加“6”，否则低四位保持不变；如果高四位的值大于“9”或C=1，那么BCD调整就执行对高四位加“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

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

LDECA [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

LINC [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

LINCA [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

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

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

LOR 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

LORM 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

LRL [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

LRLA [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

LRLC [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

LRLC 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

LRR [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

LRRA [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

LRRC [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

LRRCA [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

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

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

功能表示 ACC ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

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

LSDZ [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 3 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

LSDZA [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 3 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

LSET [m] Set Data Memory
指令说明 将指定数据存储器的每一个位置位为 1。
功能表示 [m] ← FFH
影响标志位 无

LSET [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

LSIZ [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 3 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

LSIZA [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 3 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

LSNZ [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 3 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

LSNZ [m] Skip if Data Memory is not 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，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 3 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]≠0，跳过下一条指令执行

影响标志位 无

LSUB A, [m] Subtract Data Memory from ACC

指令说明 将累加器的内容减去指定的数据存储器的数据，把结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

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

LSWAP [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

LSWAPA [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 位和高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。
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

LSZ [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 3 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

LSZA [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 3 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

LSZ [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 3 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

LTABRD [m] Move the ROM code (specific page) to TBLH and data memory
指令说明 将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)
影响标志位 无

LTABRDL [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

LITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and data memory
指令说明 自加表格指针低字节 TBLP，将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)
影响标志位 无

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and data memory
指令说明 自加表格指针低字节 TBLP，将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。
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

LXOR 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

LXORM 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

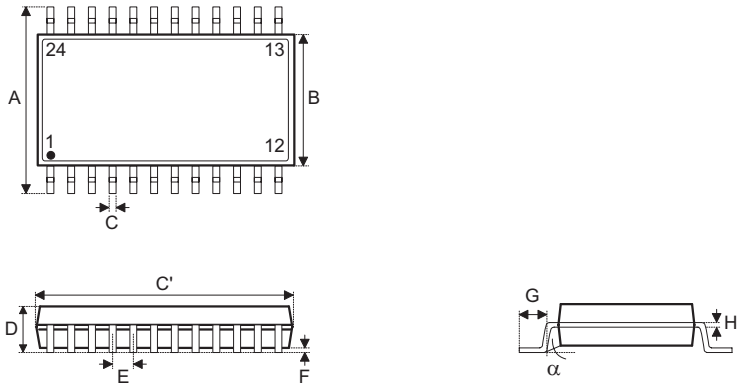
封装信息

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

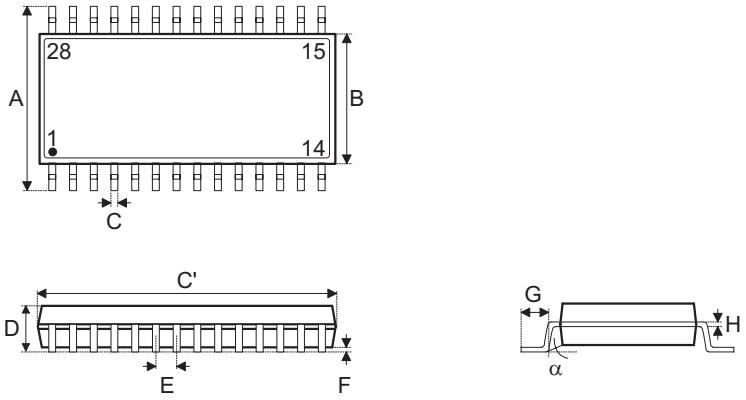
24-pin SSOP (150mil) 外形尺寸



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

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

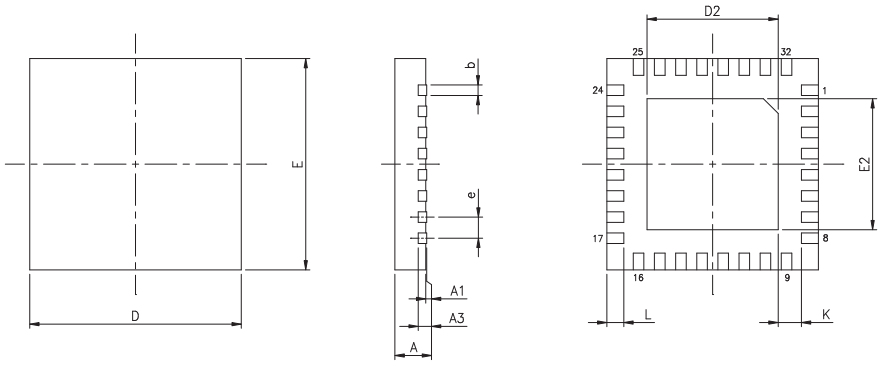
28-pin SSOP (150mil) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.20	—	0.30
C’	—	9.90 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

SAW Type 32-pin QFN (4mm×4mm×0.75mm) 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	0.006	0.008	0.010
D	—	0.157 BSC	—
E	—	0.157 BSC	—
e	—	0.016 BSC	—
D2	0.104	0.106	0.108
E2	0.104	0.106	0.108
L	0.014	0.016	0.018
K	0.008	—	—

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	0.15	0.20	0.25
D	—	4.00 BSC	—
E	—	4.00 BSC	—
e	—	0.40 BSC	—
D2	2.65	2.70	2.75
E2	2.65	2.70	2.75
L	0.35	0.40	0.45
K	0.20	—	—