Sub-1GHz 低电流 RF 收发器感烟探测器 Flash 单片机

BA45F5650

版本 : V1.30　日期 : 2022-08-11

www.holtek.com

目录

特性

CPU 特性

- 工作电压：
  - ♦ f_{SYS}=2MHz：2.2V~3.6V
  - ♦ f_{SYS}=4MHz：2.2V~3.6V
  - ♦ f_{SYS}=8MHz：2.2V~3.6V
- V_{DD}=5V，系统时钟为 8MHz 时，指令周期为 0.5μs
- 提供暂停和唤醒功能，以降低功耗
- 振荡器类型：
  - ♦ 内部高速 2/4/8MHz RC – HIRC
  - ♦ 内部低速 32kHz RC – LIRC
- 多种工作模式：快速、低速、空闲和休眠
- 内部集成的振荡器无需外接元件
- 所有指令都可在 1~3 个指令周期内完成
- 查表指令
- 115 条功能强大的指令系统
- 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：1024×8
- True EEPROM 存储器：128×8
- 看门狗定时器功能
- 支持在线应用编程 – IAP
- 17 个双向 I/O 口
- 两个与 I/O 口共用的外部中断引脚
- 可编程 I/O 口源电流用于 LED 应用
- 用于恒定电流输出的灌电流发生器
- 感烟探测器 AFE，带有 2 个运算放大器
- 电源线数据收发器，内建 2 个比较器，1 个运算放大器和 3 个 D/A 转换器
- 多个定时器模块用于时间测量、捕捉输入、比较匹配输出、PWM 输出及单脉冲输出功能
- 双时基功能，可提供固定时间的中断信号
- 串行接口模块 – SIM，用于 SPI 或 I^2C 通信
- 全双工通用异步收发器接口 – UART
- 5 个外部通道 12-bit 分辨精度的 A/D 转换器，具有内部参考电压 V_{BGREF}
- 16-bit 语音 D/A 转换器

- 低电压复位功能
- 低电压检测功能
- 封装类型：46-pin QFN

RF 特性

- 频率带宽：315/433/470/868/915MHz
- FSK/GFSK 调制
- 支持 3 线或 4 线 SPI 接口
- 可编程数据速率：2Kbps~250Kbps
- 可编程 TX 输出功率：0dBm~13dBm
- 低电流损耗
  - Deep Sleep 模式电流 0.5μA，支持数据保存
  - RX 电流损耗 (AGC 开启 & 低数据速率) @ 433.92MHz：4.2mA
  - RX 电流损耗 (AGC 开启 & 低数据速率) @ 868.3MHz：5.5mA
  - TX 电流损耗 @ 433.92MHz：22mA@10dBm P_{OUT}
  - TX 电流损耗 @ 868.3MHz：24mA@10dBm P_{OUT}
- 高 RX 灵敏度 (433.92MHz)
  - -119dBm@2Kbps 无线数据速率
  - -109dBm@50Kbps 无线数据速率
  - -100dBm@250Kbps 无线数据速率
- 高 RX 灵敏度 (868.3MHz)
  - -118dBm@2Kbps 无线数据速率
  - -108dBm@50Kbps 无线数据速率
  - -100dBm@250Kbps 无线数据速率
- 片上 VCO 以及带内置回路滤波器的小数 N 分频合成器
- 支持低成本 16MHz 晶振，内置负载电容
- 可编程数字通道滤波器，实现各种数据速率条件下较佳性能
- AGC (自动增益控制) 功能实现宽输入范围，高达 +10dBm
- AFC (自动频率补偿) 功能用于补偿晶振老化造成的频漂
- 片上低功率 RC 振荡器用于 WOR (从 RX 唤醒) 和 WOT (从 TX 唤醒) 功能
- 物理 TX/RX FIFO 缓冲器：TX 64 字节，RX 64 字节
- Simple FIFO/Block FIFO/Extend FIFO (高达 255 字节)/Infinite FIFO 模式
- 可编程载波检测阈值
- FIFO 模式和 Direct 模式支持帧同步识别
- 数据包处理
  - FEC (正向纠错)
  - 数据白化
  - 曼彻斯特编码
  - CRC-16 校验
- ATR (自动发送 / 接收)
  - 自动重发
  - 自动应答

  - ♦ WOT + 自动重发
  - ♦ WOR + 自动应答
- 数据包过滤
  - ♦ CRC 过滤
  - ♦ Address 过滤

概述

BA45F5650 是一款 A/D 型具有 8 位高性能精简指令集的 Flash 单片机，专门为感烟探测器产品而设计。

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。此外还包含了一个 RAM 数据存储器和一个可用于存储序号、校准数据等非易失性数据的 True EEPROM 存储器。

在模拟特性方面，该单片机包含一个多通道 A/D 转换器、两个比较器、三个运算放大器和四个 D/A 转换器。在内部定时器方面，带有多个使用灵活的定时器模块，可提供定时功能、脉冲产生功能及 PWM 产生功能。内建完整的 SPI、I^2C 和 UART 接口，为设计者提供了一个易与外部硬件通信的方法。内部看门狗定时器、低电压复位和低电压检测等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

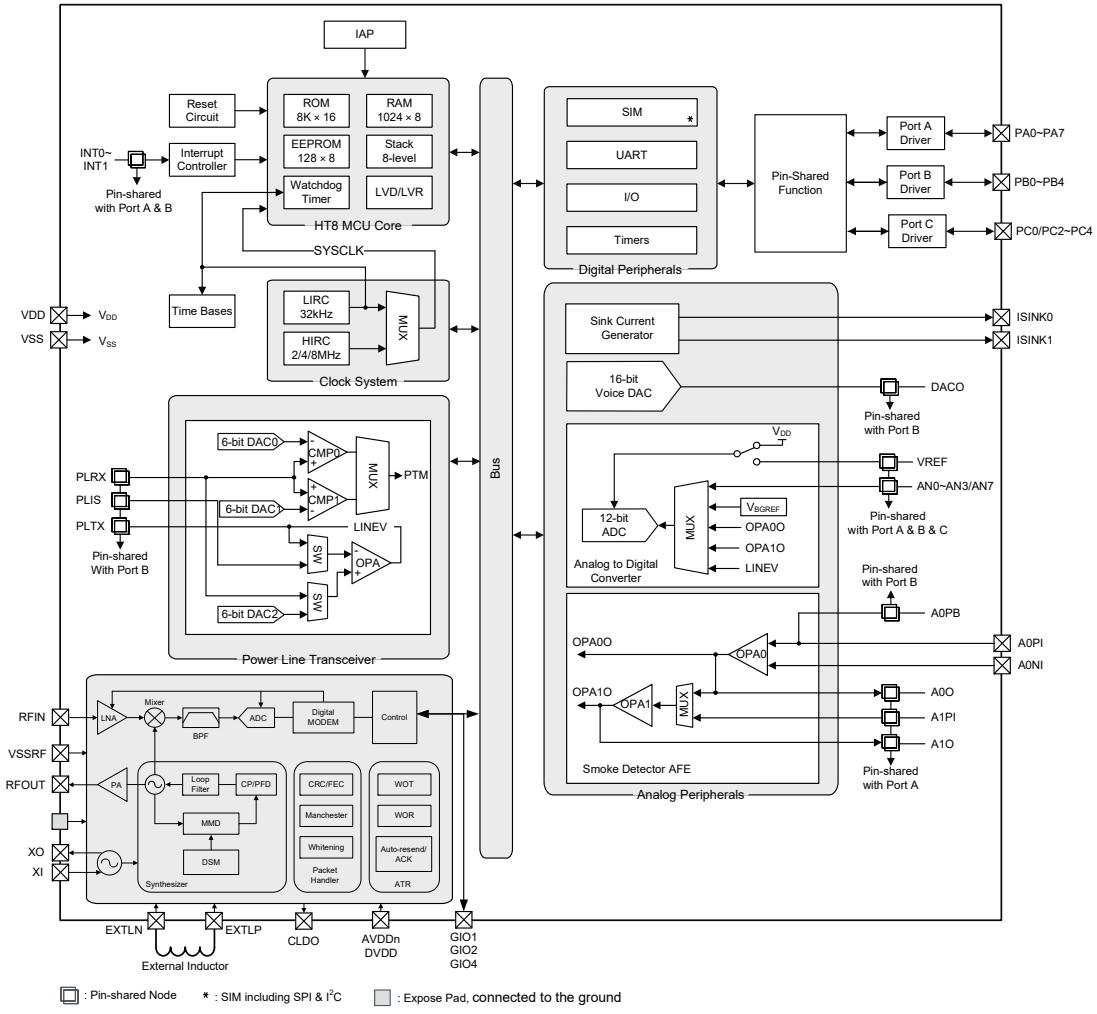
该单片机提供内部高速和低速振荡器功能选项，可灵活应用于不同程序。其在不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

内建 RF 模块为高性能、低成本 FSK/GFSK 收发器，可用于 315MHz、433MHz、470MHz、868MHz 和 915MHz 频段的无线应用。该芯片内置一个高度集成的 sub-1GHz 收发器和一个基带调制解调器，可编程数据速率范围是 2Kbps~250Kbps。数据处理特性包括 64 字节 TX/RX FIFO 和数据包处理如 CRC 生成、正向纠错、数据白化和曼彻斯特编码。

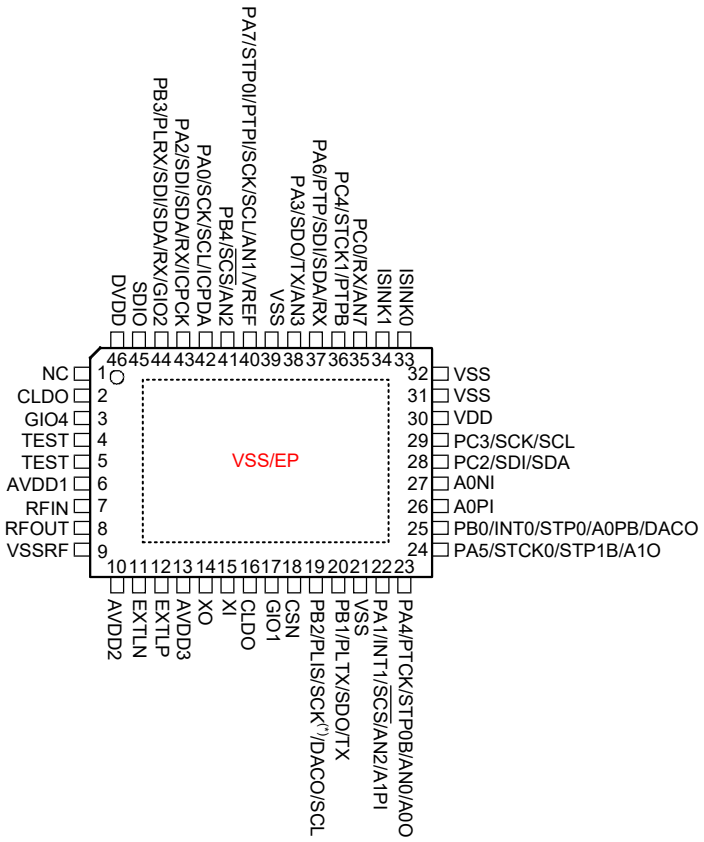
内建 RF 模块针对低功耗应用优化性能。在 433MHz 频段，其 RX 模式工作电流为 4.2mA，提供 +10dBm TX 输出功率时的电流损耗为 22mA。低功耗低中频接收器可在 433/868MHz 频段数据速率为 2Kbps 时实现 -119dBm 的灵敏度。E 类功率放大器可在 433/868MHz 频段提供高达 +13dBm 的输出功率。完全内置的小数 N 分频合成器所支持的频率范围宽，分辨精度高。芯片内部还内置有回路滤波器和晶振负载电容，减少外部元器件需求。

该单片机包含了可编程 I/O 口源电流用于 LED 应用。外加 I/O 使用灵活、时基功能和灌电流发生器等其它特性，使该单片机可以广泛应用于感烟探测器、IoT (智能小区)、无线抄表 (水表 / 气表)、农业 / 工业控制等。

方框图


IAP
Reset Circuit
ROM 8K × 16
RAM 1024 × 8
EEPROM 128 × 8
Stack 8-level
Watchdog Timer
LVD/LVR
HT8 MCU Core
INT0~INT1
Interrupt Controller
Pin-shared with Port A & B
SYSCLK
Time Bases
LIRC 32kHz
HIRC 2/4/8MHz
MUX
Clock System
VDD
VSS
V_{DD}
V_{SS}
Bus
SIM *
UART
I/O
Timers
Digital Peripherals
Pin-Shared Function
Port A Driver
Port B Driver
Port C Driver
PA0~PA7
PB0~PB4
PC0/PC2~PC4
Sink Current Generator
ISINK0
ISINK1
16-bit Voice DAC
DACO
Pin-shared with Port B
V_{DD}
VREF
AN0~AN3/AN7
Pin-shared with Port A & B & C
12-bit ADC
MUX
V_{BGREF}
OPA0O
OPA1O
LINEV
Analog to Digital Converter
Pin-shared with Port B
A0PB
A0PI
A0NI
OPA0O
OPA0
OPA1O
OPA1
MUX
A0O
A1PI
A1O
Pin-shared with Port A
Smoke Detector AFE
Analog Peripherals
6-bit DAC0
CMP0
CMP1
MUX
PTM
PLRX
PLIS
PLTX
6-bit DAC1
LINEV
SW
OPA
SW
6-bit DAC2
Pin-shared With Port B
Power Line Transceiver
RFIN
VSSRF
RFOUT
XO
XI
LNA
Mixer
BPF
ADC
Digital MODEM
Control
PA
Loop Filter
CP/PFD
MMD
DSM
Synthesizer
CRC/FEC
Manchester
Whitening
Packet Handler
WOT
WOR
Auto-resend/ ACK
ATR
EXTLN
EXTLP
External Inductor
CLDO
AVDDn
DVDD
GIO1
GIO2
GIO4
: Pin-shared Node
* : SIM including SPI & I2C
: Expose Pad, connected to the ground

引脚图



BA45F5650
46 QFN-A

注：1. 若共用引脚同时有多种输出，所需引脚共用功能通过相应的软件控制位决定。
2. SCK 和 GIO2 引脚，分别与 MCU 的 PB2 和 PB3 引脚共用。
3. *：与 PB2 共用的 SCK 引脚是 RF 的 SPI 串行时钟线或 MCU 的 SPI 串行时钟线。请勿将此引脚与 MCU 的 SPI 串行时钟线 (即与 PA 或 PC 端口共用的 SCK 引脚) 混淆。
4. 未引出的引脚 PB5，PB6，PB7，PC1 以及 PC5，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。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/SCK/SCL/ICPDA	PA0	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SCK	PAS0 IFS0	ST	CMOS	SPI 串行时钟
	SCL	PAS0 IFS0	ST	NMOS	I^2C 时钟线
	ICPDA	—	ST	CMOS	ICP 数据 / 地址

引脚名称	功能	OPT	I/T	O/T	说明
PA1/INT1/$\overline{SCS}$/A1O/A1PI	PA1	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT1	PAS0 INTC0 INTEG	ST	—	外部中断 1
	$\overline{SCS}$	PAS0 IFS0	ST	CMOS	SPI 从机选择
	A1O	PAS0	—	AN	运算放大器 1 输出
	A1PI	PAS0	AN	—	SD 运算放大器 1 同相输入端
PA2/SDI/SDA/RX/ICPCK	PA2	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SDI	PAS0 IFS0	ST	—	SPI 串行数据输入
	SDA	PAS0 IFS0	ST	NMOS	I^2C 数据线
	RX	PAS0 IFS1	ST	—	UART 接收脚
	ICPCK	—	ST	—	ICP 时钟引脚
PA3/SDO/TX/AN3	PA3	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SDO	PAS0	—	CMOS	SPI 串行数据输出
	TX	PAS0	—	CMOS	UART 发送脚
	AN3	PAS0	AN	—	A/D 转换器外部输入通道 3
PA4/PTCK/STP0B/AN0/A0O	PA4	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	PTCK	PAS1	ST	—	PTM 时钟输入或捕捉输入
	STP0B	PAS1	—	CMOS	STM0 反相输出
	AN0	PAS1	AN	—	A/D 转换器外部输入通道 0
	A0O	PAS1	—	AN	SD 运算放大器 0 输出
PA5/STCK0/STP1B/A1O	PA5	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	STCK0	PAS1	ST	—	STM0 时钟输入
	STP1B	PAS1	—	CMOS	STM1 反相输出
	A1O	PAS1	—	AN	SD 运算放大器 1 输出

引脚名称	功能	OPT	I/T	O/T	说明
PA6/PTP/SDI/SDA/RX	PA6	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	PTP	PAS1	—	CMOS	PTM 输出
	SDI	PAS1 IFS0	ST	—	SPI 串行数据输入
	SDA	PAS1 IFS0	ST	NMOS	I^2C 数据线
	RX	PAS1 IFS1	ST	—	UART 接收脚
PA7/STP0I/PTPI/SCK/SCL/AN1/VREF	PA7	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	STP0I	PAS1	ST	—	STM0 捕捉输入
	PTPI	PAS1 IFS0	ST	—	PTM 捕捉输入
	SCK	PAS1 IFS0	ST	CMOS	SPI 串行时钟
	SCL	PAS1 IFS0	ST	NMOS	I^2C 时钟线
	AN1	PAS1	AN	—	A/D 转换器外部输入通道 1
	VREF	PAS1	AN	—	A/D 转换器外部参考电压
PB0/INT0/STP0/A0PB/DACO	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	INT0	PBS0 INTC0 INTEG IFS1	ST	—	外部中断 0
	STP0	PBS0	—	CMOS	STM0 输出
	A0PB	PBS0	AN	—	SD 运算放大器 0 偏压输入
	DACO	PBS0	—	AN	16-bit D/A 转换器输出
PB1/PLTX/SDO/TX	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PLTX	PBS0	—	AN	电源线数据收发器 TX
	SDO	PBS0	—	CMOS	SPI 串行数据输出
	TX	PBS0	—	CMOS	UART 发送脚
PB2/PLIS/SCK/SCL/DACO	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PLIS	PBS0	AN	—	电源线数据收发器 IS 输入
	SCK	PBS0 IFS0	ST	CMOS	MCU SPI 串行时钟 RF SPI 串行时钟
	SCL	PBS0 IFS0	ST	NMOS	I^2C 时钟线
	DACO	PBS0	—	AN	16-bit 语音 D/A 转换器输出

引脚名称	功能	OPT	I/T	O/T	说明
PB3/PLRX/SDI/SDA/RX/GIO2	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PLRX	PBS0	AN	—	电源线数据收发器 RX
	SDI	PBS0 IFS0	ST	—	SPI 串行数据输入
	SDA	PBS0 IFS0	ST	NMOS	I^2C 数据线
	RX	PBS0 IFS1	ST	—	UART 接收脚
	GIO2	—	ST	CMOS	RF 多功能 I/O 2
PB4/$\overline{SCS}$/AN2	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	$\overline{SCS}$	PBS1 IFS0	ST	CMOS	SPI 从机选择
	AN2	PBS1	AN	—	A/D 转换器外部输入通道 2
PC0/RX/AN7	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	RX	PCS0 IFS1	ST	—	UART 接收脚
	AN7	PCS0	AN	—	A/D 转换器外部输入通道 7
PC2/SDI/SDA	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDI	PCS0 IFS0	ST	—	SPI 串行数据输入
	SDA	PCS0 IFS0	ST	NMOS	I^2C 数据线
PC3/SCK/SCL	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SCK	PCS0 IFS0	ST	CMOS	SPI 串行时钟
	SCL	PCS0 IFS0	ST	NMOS	I^2C 时钟线
PC4/STCK1/PTPB	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	STCK1	PCS1	ST	CMOS	STM1 时钟输入
	PTPB	PCS1	—	CMOS	PTM 反相输出
ISINK0	ISINK0	—	—	AN	灌电流源 0
ISINK1	ISINK1	—	—	AN	灌电流源 1
A0NI	A0NI	—	AN	—	SD 运算放大器 0 反相输入
A0PI	A0PI	—	AN	—	SD 运算放大器 0 同相输入
CLDO	CLDO	—	—	PWR	LDO 输出，连接旁路电容
GIO4	GIO4	—	ST	CMOS	RF 多功能 I/O 4
AVDD1	AVDD1	—	PWR	—	RF 模拟电源
RFIN	RFIN	—	AN	—	RF LNA 输入

引脚名称	功能	OPT	I/T	O/T	说明
RFOUT	RFOUT	—	—	AN	RF 功率放大器输出
VSSRF	VSSRF	—	PWR	—	RF 地
AVDD2	AVDD2	—	PWR	—	RF 模拟电源
EXTLN	EXTLN	—	AN	—	连接到外部电感
EXTLP	EXTLP	—	AN	—	连接到外部电感
AVDD3	AVDD3	—	PWR	—	RF 模拟电源
XO	XO	—	—	HXT	晶振输出
XI	XI	—	HXT	—	晶振输入
CLDO	CLDO	—	—	PWR	LDO 输出，连接旁路电容
GIO1	GIO1	—	ST	CMOS	RF 多功能 I/O 1
CSN	CSN	—	ST	—	SPI 芯片选择输入，低有效
SDIO	SDIO	—	ST	CMOS	SPI 数据输入 / 输出
DVDD	DVDD	—	PWR	—	RF 数字电源
GND	GND	—	PWR	—	地
VDD	VDD	—	PWR	—	数字正电源
VSS	VSS	—	PWR	—	数字负电源，接地
NC	NC	—	—	—	未连接
TEST	—	—	—	—	未连接，使其浮空
VSS/EP	VSS	—	PWR	—	裸露焊盘，接地

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

极限参数

电源供应电压 ..... V_{SS}-0.3V~3.6V
端口输入电压 ..... V_{SS}-0.3V~V_{DD}+0.3V
储存温度 ..... -60°C~150°C
工作温度 ..... -40°C~85°C
I_{OL} 总电流 80mA
I_{OH} 总电流 ..... -80mA
总功耗 500mW
ESD HBM ..... ±2kV

* 该芯片对 ESD 敏感。人体模式 (Human Body Mode) 符合 MIL-STD-883 标准。

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、工作温度、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~85°C

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	$f_{SYS}=f_{HIRC}$=2MHz	2.2	—	3.6	V
		$f_{SYS}=f_{HIRC}$=4MHz	2.2	—	3.6	
		$f_{SYS}=f_{HIRC}$=8MHz	2.2	—	3.6	
	工作电压 – LIRC	$f_{SYS}=f_{LIRC}$=32kHz	2.2	—	3.6	V

工作电流特性

Ta=-40°C~85°C

符号	工作模式	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	低速模式 – LIRC	2.2V	f_{SYS}=32kHz	—	8	16	μA
		3V		—	10	20	
	快速模式 – HIRC	2.2V	f_{SYS}=2MHz	—	0.15	0.20	mA
		3V		—	0.2	0.3	
		2.2V	f_{SYS}=4MHz	—	0.3	0.5	mA
		3V		—	0.4	0.6	
		2.2V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	

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=-40°C~85°C，除非另有说明

符号	待机模式	测试条件		最小	典型	最大	最大 @85°C	单位
		V_{DD}	条件					
I_{STB}	休眠模式	2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
	空闲模式 0 – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
	空闲模式 1 – HIRC	2.2V	f_{SUB} on，f_{SYS}=2MHz	—	60	120	140	μA
		3V		—	70	140	160	
		2.2V	f_{SUB} on，f_{SYS}=4MHz	—	90	200	220	μA
		3V		—	110	220	240	
		2.2V	f_{SUB} on，f_{SYS}=8MHz	—	150	300	340	μA
		3V		—	180	360	400	

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 指令执行后即停止执行所有指令后测得。

交流电气特性

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

内部高速振荡器 HIRC 频率精准度

程序烧录时，烧录器会依据用户选择的 HIRC 频率和工作电压 (3V) 对 HIRC 进行频率精准度调整。

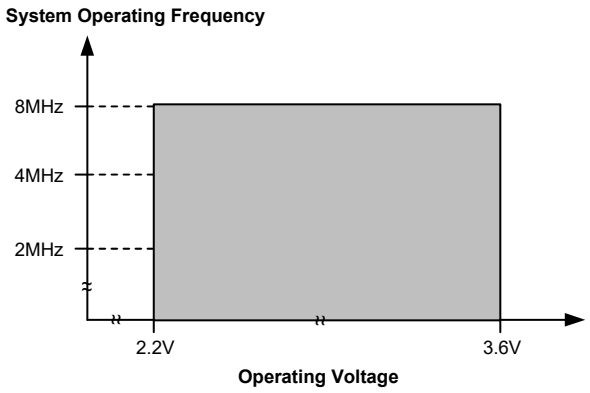
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{HIRC}	通过烧录器调整后的 2MHz HIRC 频率	3V	25°C	-1%	2	+1%	MHz
			-20°C~60°C	-2%	2	+2%	
	通过烧录器调整后的 4MHz HIRC 频率	3V	25°C	-1%	4	+1%	MHz
		2.4V~3.3V	-20°C~50°C	-2%	4	+2%	
	通过烧录器调整后的 8MHz HIRC 频率	3V	25°C	-1%	8	+1%	MHz

注：1. 烧录器可在 3V 这个固定电压下对 HIRC 频率进行调整，在此提供 V_{DD}=3V 时的参数值。
2. 3V 表格列下面提供的是全压条件下的参数值。对于电压范围在 2.2V~3.6V 的应用，建议烧录器电压固定在 3V。
3. 表格中提供的最小和最大误差值仅在对应的烧录器调整频率下有效。当烧录器已对所选的频率进行调整，此后再通过程序中振荡器控制位将其频率改为其它值时，频率误差范围将增加到 ±20%。

内部低速振荡器 LIRC 电气特性

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	2.2V~3.6V	-40°C~85°C	-7%	32	+7%	kHz
t_{START}	LIRC 启动时间	3V	-40°C~85°C	—	—	100	μs

工作频率电气特性曲线图



系统上电时间电气特性

Ta=-40°C~85°C

符号	参数	测试条件	最小	典型	最大	单位
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	42	48	54	ms
	系统复位延迟时间 (WDTC 软件复位)	—				
	系统复位延迟时间 (WDT 溢出复位)	—	14	16	18	
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} 等符号所表示的时间单位，是对应频率值的倒数，相关频率值在前面表格有说明。例如，$t_{HIRC}=1/f_{HIRC}$，$t_{SYS}=1/f_{SYS}$ 等等。
3. 若 LIRC 被选择作为系统时钟源且在休眠模式下 LIRC 关闭，则上面表格中对应 t_{SST} 数值还需加上 LIRC 频率表格里提供的 LIRC 启动时间 t_{START}。
4. 系统速度切换时间实际上是指新使能的振荡器的启动时间。

输入 / 输出口电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IL}	I/O 口低电平输入电压	—	—	0	—	$0.2V_{DD}$	V
V_{IH}	I/O 口高电平输入电压	—	—	$0.8V_{DD}$	—	V_{DD}	V
I_{OL}	I/O 口灌电流	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
I_{OH}	I/O 口源电流	3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=00B (n=0, 1; m=0, 2, 4, 6)	-0.7	-1.5	—	mA
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=01B (n=0, 1; m=0, 2, 4, 6)	-1.3	-2.5	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=10B (n=0, 1; m=0, 2, 4, 6)	-1.8	-3.6	—	
I_{OH}	I/O 口源电流	3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=11B (n=0, 1; m=0, 2, 4, 6)	-4	-8	—	mA
R_{PH}	I/O 口上拉电阻 (注)	3V	—	20	60	100	kΩ
I_{TOL}	所有 I/O 口灌电流总和	3V	—	40	—	—	mA

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{TOH}	所有 I/O 口源电流总和	3V	—	-40	—	—	mA
t_{TPI}	STM STP0I 输入引脚最小脉宽	—	—	0.3	—	—	μs
	PTM PTPI 输入引脚最小脉宽			0.1	—	—	
t_{TCK}	STM STCKn 输入引脚最小脉宽	—	—	0.3	—	—	μs
	PTM PTCK 输入引脚最小脉宽			0.3	—	—	
t_{INT}	中断引脚最小脉宽	—	—	10	—	—	μs
t_{CPW}	PTM 捕捉输入最小脉宽	—	—	2	—	—	t_{TMCLK}

注：R_{PH} 内部上拉电阻值的计算方法是：将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的电流，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°C~85°C，除非另有说明。

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{RW}	读 / 写工作电压	—	—	V_{DDmin}	—	V_{DDmax}	V
程序存储器 / 数据 EEPROM 存储器							
t_{DEW}	擦除 / 写周期时间 – Flash 程序存储器	—	—	—	2	3	ms
	写周期时间 – 数据 EEPROM 存储器	—	—	—	4	6	ms
I_{DDPGM}	V_{DD} 电压下烧录 / 擦除电流	—	—	—	—	5.0	mA
E_P	存储单元耐受性 – Flash 程序存储器	—	—	10K	—	—	E/W
	存储单元耐受性 – 数据 EEPROM 存储器	—	—	100K	—	—	
t_{RETD}	程序存储器数据保存时间	—	Ta=25°C	—	40	—	Year
RAM 数据存储器							
V_{DR}	RAM 数据保存电压	—	单片机处于 SLEEP 模式	1.0	—	—	V

注：“E/W”表示擦 / 写次数。

LVD & LVR 电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{LVR}	低电压复位电压	—	LVR 使能	-5%	2.1	+5%	V
V_{LVD}	低电压检测电压	—	LVD 使能，电压选择 2.0V	-5%	2.0	+5%	V
			LVD 使能，电压选择 2.2V		2.2		
			LVD 使能，电压选择 2.4V		2.4		
			LVD 使能，电压选择 2.7V		2.7		
			LVD 使能，电压选择 3.0V		3.0		
			LVD 使能，电压选择 3.3V		3.3		
			LVD 使能，电压选择 3.6V		3.6		
			LVD 使能，电压选择 4.0V		4.0		
$I_{LVRLVDBG}$	工作电流	3V	LVD 使能，LVR 使能，VBGEN=0	—	—	20	μA
		3V	LVD 使能，LVR 使能，VBGEN=1	—	—	25	μA
t_{LVDS}	LVDO 稳定时间	—	LVR 使能时，VBGEN=0，LVD off → on	—	—	18	μs
			LVR 除能时，VBGEN=0，LVD off → on	—	—	150	
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，VBGEN=0	—	—	24	μA
I_{LVD}	LVD 使能的额外电流	—	LVR 除能，VBGEN=0	—	—	24	μA

A/D 转换器电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{ADI}	输入电压	—	—	0	—	V_{REF}	V
V_{REF}	参考电压	—	—	2	—	V_{DD}	V
N_R	分辨精度	—	—	—	—	12	Bit
DNL	非线性微分误差	—	$V_{REF}=V_{DD}$，$t_{ADCK}=0.5\mu s$	-3	—	3	LSB
INL	非线性积分误差	—	$V_{REF}=V_{DD}$，$t_{ADCK}=0.5\mu s$	-4	—	4	LSB
I_{ADC}	A/D 转换器使能的额外电流	2.2V	无负载，$t_{ADCK}=0.5\mu s$	—	300	420	μA
		3V		—	340	500	
t_{ADCK}	时钟周期	—	—	0.5	—	10.0	μs
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}

内部参考电压特性

Ta=-40°C~85°C，除非另有说明。

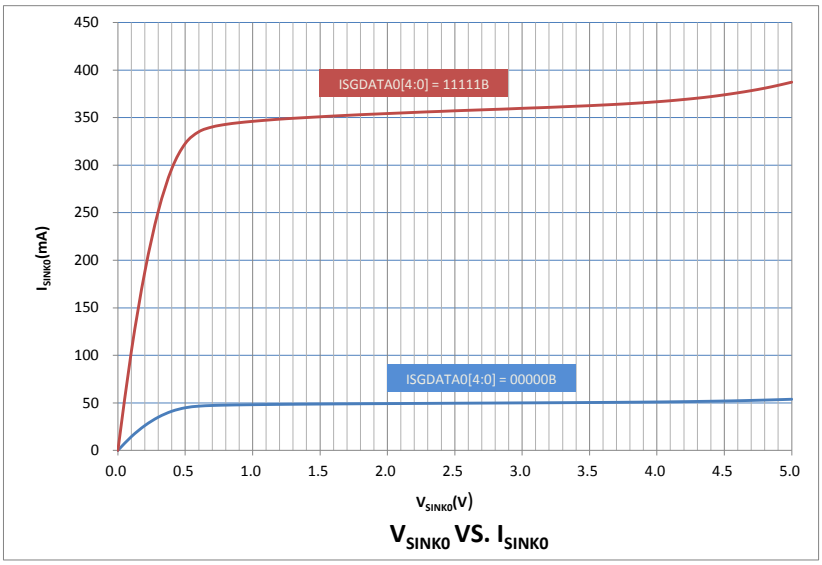
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.2	—	3.6	V
V_{BGREF}	Bandgap 参考电压	—	Ta=25°C	-0.3%	1.2	+0.3%	V
			—	-1%	1.2	+1%	
PSRR	电源电压抑制比	—	Ta=25°C，V_{RIPPLE}=1V_{P-P}，f_{RIPPLE}=100Hz	75	—	—	dB
En	输出噪声	—	Ta=25°C，无负载电流，f=0.1Hz~10Hz	—	300	—	μV_{RMS}
I_{DRV}	缓冲器驱动能力	—	ΔV_{BGREF}=-1%	1	—	—	mA
I_{SD}	关机电流	—	VBGREN=0	—	—	0.1	μA
t_{START}	启动时间	2.2V~3.6V	Ta=25°C	—	—	400	μs

注：1. 以上参数均在无负载的条件下测得，除非有特别说明。
2. VDD 引脚需连接一个 0.1μF 陶瓷电容到地。
3. V_{BGREF} 电压可以用作 A/D 转换器内部信号输入。

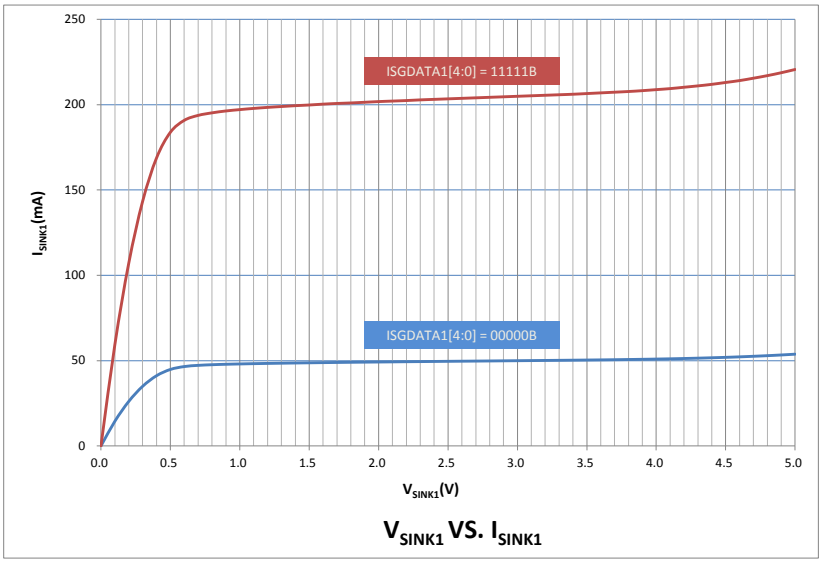
灌电流发生器电气特性

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{SINK0}	ISINK0 引脚灌电流	—	Ta=-40°C~85°C，V_{ISINK0}=1.0V~3.6V，ISGDATA0[4:0]=00000B	41	50	59	mA
		—	Ta=-40°C~85°C，V_{ISINK0}=0.7V~1.0V，ISGDATA0[4:0]=00000B	37.5	50.0	50.0	
		—	Ta=-40°C~85°C，V_{ISINK0}=1.0V~3.6V，ISGDATA0[4:0]=11111B	295	360	425	
		—	Ta=-40°C~85°C，V_{ISINK0}=0.7V~1.0V，ISGDATA0[4:0]=11111B	270	360	360	
I_{SINK1}	ISINK1 引脚灌电流	—	Ta=-40°C~85°C，V_{ISINK1}=1.0V~3.6V，ISGDATA1[4:0]=00000B	41	50	59	mA
		—	Ta=-40°C~85°C，V_{ISINK1}=0.7V~1.0V，ISGDATA1[4:0]=00000B	37.5	50.0	50.0	
		—	Ta=-40°C~85°C，V_{ISINK1}=1.0V~3.6V，ISGDATA1[4:0]=11111B	168	205	242	
		—	Ta=-40°C~85°C，V_{ISINK1}=0.7V~1.0V，ISGDATA1[4:0]=11111B	154	205	205	

灌电流发生器电气特性曲线图



I_{SINK0} 电气特性曲线图



I_{SINK1} 电气特性曲线图

运算放大器电气特性

感烟探测器 AFE

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.2	—	3.6	V
I_{OPA}	工作电流	3V	SDAmBW[1:0]=00B (m=0, 1)，无负载	—	3.0	5.0	μA
			SDAmBW[1:0]=01B (m=0, 1)，无负载	—	10	16	
			SDAmBW[1:0]=10B (m=0, 1)，无负载	—	80	128	
			SDAmBW[1:0]=11B (m=0, 1)，无负载	—	200	320	
V_{OS}	输入失调电压	3V	未校准 SDAmOF[5:0]=100000B (m=0, 1)	-15	—	15	mV
			已校准	-4	—	4	
I_{OS}	输入失调电流	3V	$V_{IN}=1/2V_{CM}$	—	1	10	nA
V_{CM}	共模电压范围	3V	SDAmBW[1:0]=00, 01, 10, 11 (m=0, 1)	V_{SS}	—	V_{DD}-1.4	V
PSRR	电源电压抑制比	3V	SDAmBW[1:0]=00, 01, 10, 11 (m=0, 1)	50	70	—	dB
CMRR	共模抑制比	3V	SDAmBW[1:0]=00, 01, 10, 11 (m=0, 1)	50	80	—	dB
A_{OL}	开环增益	3V	SDAmBW[1:0]=00, 01, 10, 11 (m=0, 1)	60	80	—	dB
SR	转换速率	3V	R_{LOAD}=1MΩ，C_{LOAD}=60pF，SDAmBW[1:0]=00 (m=0, 1)	0.5	1.0	—	V/ms
			R_{LOAD}=1MΩ，C_{LOAD}=60pF，SDAmBW[1:0]=01 (m=0, 1)	5	10	—	
			R_{LOAD}=1MΩ，C_{LOAD}=60pF，SDAmBW[1:0]=10 (m=0, 1)	180	300	—	
			R_{LOAD}=1MΩ，C_{LOAD}=60pF，SDAmBW[1:0]=11 (m=0, 1)	600	1200	—	
GBW	增益带宽	3V	R_{LOAD}=1MΩ，C_{LOAD}=60pF，SDAmBW[1:0]=00 (m=0, 1)	2.0	3.5	—	kHz
		3V	R_{LOAD}=1MΩ，C_{LOAD}=60pF，SDAmBW[1:0]=01 (m=0, 1)	15	30	—	
		3V	R_{LOAD}=1MΩ，C_{LOAD}=60pF，SDAmBW[1:0]=10 (m=0, 1)	250	500	—	
		3V	R_{LOAD}=1MΩ，C_{LOAD}=60pF，SDAmBW[1:0]=11 (m=0, 1)	800	1600	—	

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{OR}	最大输出电压范围	3V	SDAmBW[1:0]=00, 01 (m=0, 1) R_{LOAD}=5kΩ 接到 V_{DD}/2 处	V_{SS}+140	—	V_{DD}-160	mV
			SDAmBW[1:0]=10, 11 (m=0, 1) R_{LOAD}=5kΩ 接到 V_{DD}/2 处	V_{SS}+120	—	V_{DD}-140	
I_{SC}	输出短路电流	3V	R_{LOAD}=5.1Ω， SDAmBW[1:0]=00, 01 (m=0, 1)	±1.2	±5.0	—	mA
			R_{LOAD}=5.1Ω， SDAmBW[1:0]=10, 11 (m=0, 1)	±2	±10	—	

注：表格中为特征值，未经实测。

电源线数据收发器

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.2	—	3.6	V
I_{OPA}	工作电流	3V	PLTABW=0，无负载	—	80	128	μA
			PLTABW=1，无负载	—	200	320	
V_{OS}	输入失调电压	3V	未校准 (PLTAOF[5:0]=100000B)	-15	—	15	mV
			已校准	-2	—	2	
I_{OS}	输入失调电流	3V	V_{IN}=1/2V_{CM}	—	1	10	nA
V_{CM}	共模电压范围	3V	PLTABW=0, 1	V_{SS}	—	V_{DD}-1.4	V
PSRR	电源电压抑制比	3V	PLTABW=0, 1	50	70	—	dB
CMRR	共模抑制比	3V	PLTABW=0, 1	50	80	—	dB
A_{OL}	开环增益	3V	PLTABW=0, 1	60	80	—	dB
SR	转换速率	3V	R_{LOAD}=1MΩ，C_{LOAD}=60pF， PLTABW=0	180	500	—	V/ms
			R_{LOAD}=1MΩ，C_{LOAD}=60pF， PLTABW=1	600	1800	—	
GBW	增益带宽	3V	R_{LOAD}=1MΩ，C_{LOAD}=60pF， PLTABW=0	400	600	—	kHz
		3V	R_{LOAD}=1MΩ，C_{LOAD}=60pF， PLTABW=1	1300	2000	—	
V_{OR}	最大输出电压范围	3V	PLTABW=0, 1 R_{LOAD}=5kΩ 接到 V_{DD}/2 处	V_{SS} +140	—	V_{DD} -160	mV
I_{SC}	输出短路电流	3V	R_{LOAD}=5.1Ω，PLTABW=0, 1	±2	±10	—	mA

注：表格中为特征值，未经实测。

D/A 转换器电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.2	—	3.6	V
V_{DACO}	输出电压范围	—	—	V_{SS}	—	V_{REF}	V
V_{REF}	参考电压	—	—	2	—	V_{DD}	V
I_{DAC}	DAC 使能的额外电流 (DAC0&DAC1)	3V	—	—	—	12	μA
	DAC 使能的额外电流 (DAC2)	3V	—	—	—	360	μA
t_{ST}	建立时间	3V	C_{LOAD}=50pF	—	—	5	μs
DNL	非线性微分误差	3V	$V_{REF}=V_{DD}$	-1	—	+1	LSB
INL	非线性积分误差	3V	$V_{REF}=V_{DD}$	-1.5	—	+1.5	LSB
R_O	电阻串输出电阻 (DAC0&DAC1)	3V	—	—	1000	—	kΩ
	R2R 输出电阻	3V	—	—	10	—	kΩ
OSRR	偏置误差	3V	—	—	—	6	mV
GERR	增益误差	3V	—	—	—	12	mV

16-bit 语音 D/A 转换器电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.2	3.0	3.6	V
I_{DAC}	带缓冲器 D/A 转换器使能的额外电流	3V	—	—	—	3	mA
$I_{STB(DAC)}$	待机电流	3V	DACEN=0	—	—	1	μA
THD+N	总谐波失真 + 噪音 (注)	3V	10kΩ 负载	—	-55	—	dB
V_{OUT}	输出电压范围	3V	无负载	0.01	—	0.99	V_{DD}
t_{DACS}	D/A 转换器启动稳定时间	3V	—	—	—	1	ms

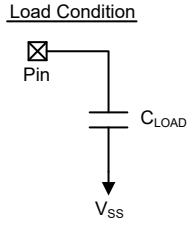
注：正弦波输入 @ 1kHz，-6dBFS。

比较器电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	比较器工作电压	—	—	2.2	—	3.6	V
I_{CMP}	比较器使能的额外电流	—	无负载，PLTCmIS[1:0]=00B (m=0, 1)	—	1.7	2.7	μA
			无负载，PLTCmIS[1:0]=01B (m=0, 1)	—	14	22	
			无负载，PLTCmIS[1:0]=10B (m=0, 1)	—	36	57	
			无负载，PLTCmIS[1:0]=11B (m=0, 1)	—	58	92	
V_{CM}	比较器共模电压范围	—	—	V_{SS}	—	V_{DD}-1.4	V
t_{RP}	响应时间	3V	10mV 过驱动电压（注） 无去抖，PLTCmIS[1:0]=00B (m=0, 1)	—	—	35	μs
		3V	10mV 过驱动电压（注） 无去抖，PLTCmIS[1:0]=01B (m=0, 1)	—	—	2.5	
		3V	10mV 过驱动电压（注） 无去抖，PLTCmIS[1:0]=10B (m=0, 1)	—	—	1	
		3V	10mV 过驱动电压（注） 无去抖，PLTCmIS[1:0]=11B (m=0, 1)	—	—	0.7	
V_{HYS}	迟滞宽度	3V	PLTCmHYS[1:0]=00， PLTCmIS[1:0]=00 (m=0, 1)	0	0	5	mV
		3V	PLTCmHYS[1:0]=01， PLTCmIS[1:0]=01 (m=0, 1)	20	40	60	
		3V	PLTCmHYS[1:0]=10， PLTCmIS[1:0]=10 (m=0, 1)	50	100	150	
		3V	PLTCmHYS[1:0]=11， PLTCmIS[1:0]=11 (m=0, 1)	80	160	240	

注：以上参数是在比较器输入电压 = (V_{DD}-1.4)/2 且保持不变的条件下测量。
负载条件：C_{LOAD}=50pF



RF 电气特性

Ta=25°C，V_{DD}=3.3V，f_{XTAL}=16MHz，FSK 调制 (含匹配电路和低 / 高通滤波器)，RF 输出由 V_{DD} (3.3V) 供电，除非另有说明

符号	参数	测试条件	最小	典型	最大	单位
T_{OP}	工作温度	—	-40	—	85	°C
V_{DD}	电源电压	—	2.2	3.3	3.6	V
数字输入 / 输出						
V_{IH}	高电平输入电压	—	0.7 ×V_{DD}	—	V_{DD}	V
V_{IL}	低电平输入电压	—	0	—	0.3 ×V_{DD}	V
V_{OH}	高电平输出电压	I_{OH}=-5mA	0.8 ×V_{DD}	—	V_{DD}	V
V_{OL}	低电平输出电压	I_{OL}=5mA	0	—	0.2 ×V_{DD}	V
电流损耗						
I_{Sleep}	Deep Sleep 模式电流损耗	—	—	0.4	1.0	μA
I_{IL}	Idle 模式电流损耗	LIRC 开启，晶振关闭	—	1.8	—	μA
	Light Sleep 模式电流损耗	晶振开启	—	0.55	—	mA
$I_{Standby}$	Standby 模式电流损耗 @ 315/433MHz	晶振开启，合成器开启	—	2.2	—	mA
	Standby 模式电流损耗 @ 868/915MHz		—	3.0	—	
I_{RX} 或 I_{TX}	315MHz 频段电流损耗	RX 模式 @50Kbps	—	4.1	—	mA
		RX 模式 @250Kbps	—	4.4	—	
		TX 模式 @0dBm P_{OUT}	—	14	—	
		TX 模式 @10dBm P_{OUT}	—	24	—	
		TX 模式 @13dBm P_{OUT}	—	30	—	
	433MHz 频段电流损耗	RX 模式 @50Kbps	—	4.2	—	mA
		RX 模式 @250Kbps	—	4.6	—	
		TX 模式 @0dBm P_{OUT}	—	14	—	
		TX 模式 @10dBm P_{OUT}	—	22	—	
		TX 模式 @13dBm P_{OUT}	—	30	—	
	868MHz 频段电流损耗	RX 模式 @50Kbps	—	5.5	—	mA
		RX 模式 @250Kbps	—	6.1	—	
		TX 模式 @0dBm P_{OUT}	—	15	—	
		TX 模式 @10dBm P_{OUT}	—	24	—	
		TX 模式 @13dBm P_{OUT}	—	32	—	
	915MHz 频段电流损耗	RX 模式 @50Kbps	—	6	—	mA
		RX 模式 @250Kbps	—	6.5	—	
		TX 模式 @0dBm P_{OUT}	—	18	—	
		TX 模式 @10dBm P_{OUT}	—	24	—	
		TX 模式 @13dBm P_{OUT}	—	32	—	

符号	参数	测试条件	最小	典型	最大	单位
R_{PH}	I/O 口上拉电阻	3.3V	—	33	—	kΩ
RF 特性						
f_{RF}	RF 频段	315MHz 频段	—	315	—	MHz
		433MHz 频段	—	433.92	—	
		470~510MHz 频段	—	490	—	
		868MHz 频段	—	868.3	—	
		915MHz 频段	—	915	—	
DR	数据速率	GFSK 调制	2	—	250	Kbps
发送器						
P_{OUT}	TX 输出功率	433MHz 频段	0	—	13	dBm
		868MHz 频段	0	—	13	
S.E.$_{TX}$	TX 杂散 (P_{OUT}=10dBm)	f < 1GHz	—	—	-36	dBm
		47MHz < f < 74MHz	—	—	-54	
		87.5MHz < f < 118MHz				
		174MHz < f < 230MHz				
		470MHz < f < 862MHz				
		二次谐波，三次谐波	—	—	-30	
接收器						
$t_{ST, RX}$	RX 稳定时间	Light Sleep 模式到 RX 模式	—	150	—	μs
P_{Sens}	433MHz RX 灵敏度 @ BER=0.1%	2Kbps (f_{DEV}=8kHz)	—	-119	—	dBm
		10Kbps (f_{DEV}=40kHz)	—	-112	—	
		50Kbps (f_{DEV}=18.75kHz)	—	-109	—	
		125Kbps(f_{DEV}=46.875kHz)	—	-104	—	
		250Kbps (f_{DEV}=93.75kHz)	—	-100	—	
	868MHz RX 灵敏度 @ BER=0.1%	2Kbps (f_{DEV}=8kHz)	—	-118	—	dBm
		10Kbps (f_{DEV}=40kHz)	—	-112	—	
		50Kbps (f_{DEV}=18.75kHz)	—	-108	—	
		125Kbps (f_{DEV}=46.875kHz)	—	-104	—	
		250Kbps (f_{DEV}=93.75kHz)	—	-100	—	
$P_{IN, max}$	最大输入功率	@ BER<0.1%	—	—	10	dBm
IR	镜像抑制	—	—	25	—	dB
S.E.$_{RX}$	RX 杂散	25MHz~1GHz	—	—	-57	dBm
		大于 1GHz	—	—	-47	
	RSSI 范围	AGC 开启	-110	—	-10	dBm
LO 特性						
f_{LO}	RF 频率范围	315MHz 频段	290	—	335	MHz
		433MHz 频段	415	—	490	
		470~510MHz 频段	470	—	510	
		868MHz 频段	830	—	1000	
		915MHz 频段	870	—	1050	
f_{STEP}	LO 频率分辨精度	—	—	—	1	kHz

符号	参数	测试条件	最小	典型	最大	单位
PN_{LO}	315MHz 相位噪声	@ 100kHz 偏移	—	-86	—	dBc/Hz
		@ 1MHz 偏移	—	-107	—	
	433MHz 相位噪声	@ 100kHz 偏移	—	-85	—	
		@ 1MHz 偏移	—	-106	—	
	868MHz 相位噪声	@ 100kHz 偏移	—	-82	—	
		@ 1MHz 偏移	—	-103	—	
	915MHz 相位噪声	@ 100kHz 偏移	—	-82	—	
		@ 1MHz 偏移	—	-103	—	
晶振						
f_{XTAL}	晶振频率	—	—	16	—	MHz
ESR	晶振等效串联电阻	—	—	—	100	Ω
C_{LOAD}	晶振电容负载	—	12	16	20	pF
TOL	晶振容差 (注)	—	-20	—	+20	dBc/Hz
t_{SU}	晶振启动时间	49US XO	—	—	1	ms

注：当数据率 = 2Kbps @ 315/433.92MHz 时，晶振容差需选择 ±10ppm。
当数据率 = 2Kbps @ 868/915MHz 时，晶振容差需选择 ±5ppm。

SPI 电气特性

Ta=-40°C~85°C，除非另有说明。

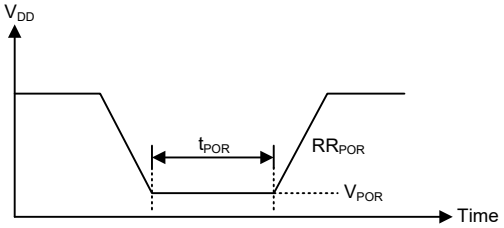
符号	参数	测试条件	最小	典型	最大	单位
f_{SCK}	SCK 频率	—	—	4	—	MHz
t_{SCKH}	SCK 高电平时间	—	$1/f_{XCLK}$	—	—	s
t_{SCKL}	SCK 低电平时间	—	$1/f_{XCLK}$	—	—	s
t_{S_SDIO}	SDIO 输入设置时间	—	20	—	—	ns
t_{H_SDIO}	SDIO 输入保持时间	—	20	—	—	ns
t_{S_CSN}	CSN 有效到 SCK 有效时间间隔	—	30	—	—	ns
t_{H_CSN}	SCK 无效到 CSN 无效时间间隔	—	30	—	—	ns

注：$f_{XCLK}= f_{XTAL}/(XODIV2+1)$。

上电复位特性

Ta=-40°C~85°C

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



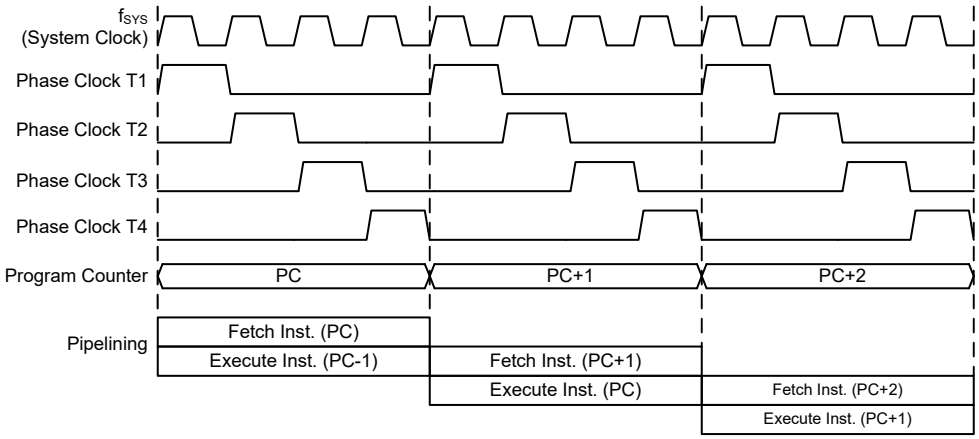
系统结构

内部系统结构是 Holtek 单片机具有良好性能的主要因素。由于采用 RISC 结构，此单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令需要一个以上指令周期外，大部分的标准指令或扩展指令分别能在一个指令周期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。使得这些单片机适用于低成本和批量生产的控制应用。

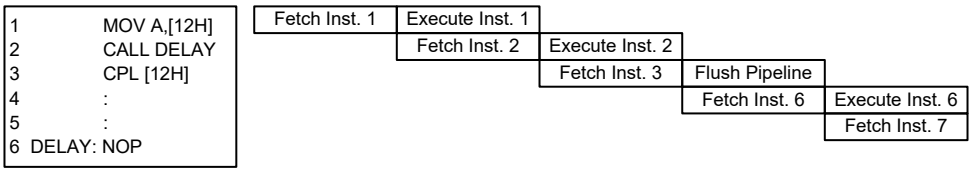
时序和流水线结构

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

如果指令牵涉到分支，例如跳转或调用等指令，则需要两个指令周期才能完成指令执行。需要一个额外周期的原因是程序先用一个周期取出实际要跳转或调用的地址，再用另一个周期去实际执行分支动作，因此用户需要特别考虑额外周期的问题，尤其是在执行时间要求较严格的时候。



系统时序和流水线



指令捕捉

程序计数器

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

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

程序计数器	
程序计数器高字节	PCL 寄存器
PC12~PC8	PCL7~PCL0

程序计数器

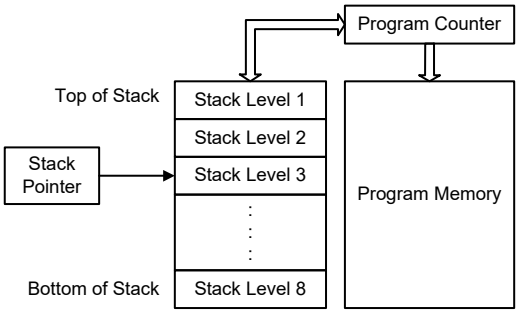
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

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：

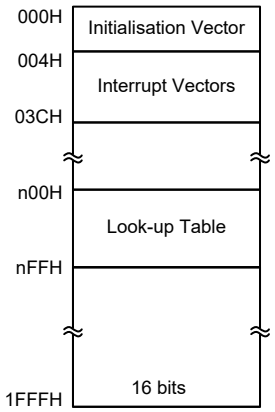
- 算术运算：
  ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA
  LADD，LADDM，LADC，LADCM，LSUB，LSUBM，LSBC，LSBCM，LDAA
- 逻辑运算：
  AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA
  LAND，LOR，LXOR，LANDM，LORM，LXORM，LCPL，LCPLA
- 移位运算：
  RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC
  LRRA，LRR，LRRCA，LRRC，LRLA，LRL，LRLCA，LRLC
- 递增和递减：
  INCA，INC，DECA，DEC
  LINCA，LINC，LDECA，LDEC
- 分支判断：
  JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI
  LSZ，LSZA，LSNZ，LSIZ，LSDZ，LSIZA，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

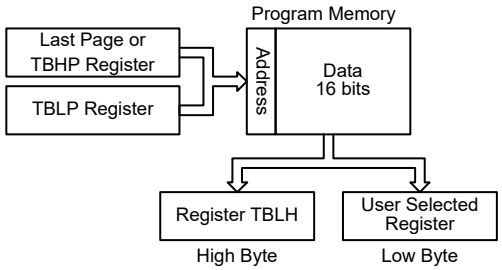
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

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“1F00H”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]”或“LTABRD”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]” 或“LTABRD”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 / 可写寄存器，且能重复储存，若主程序和中断服务程序都使用表格读取指令，应该注意它的保护。使用表格读取指令，中断服务程序可能会改变 TBLH 的值，若随后在主程序中再次使用这个值，则会发生错误，因此建议避免同时使用表格读取指令。然而在某些情况下，如果同时使用表格读取指令是不可避免的，则在执行任何主程序的表格读取指令前，中断应该先除能，另外要注意的是所有与表格相关的指令，都需要两个指令周期去完成操作。

表格读取程序范例

```
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
:
:
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 register
                    ; tempreg2
:
:
org 1F00h           ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

在线烧录 – ICP

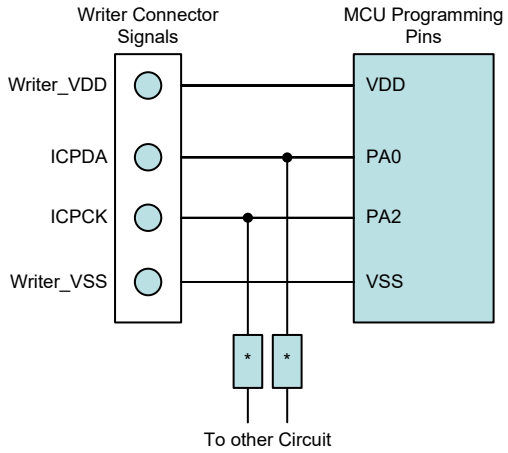
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。另外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧写，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

Holtek Flash MCU 与烧录器引脚对应表如下：

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

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执行 IAP 固件程序。

Flash 存储器读取 / 写入容量

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。

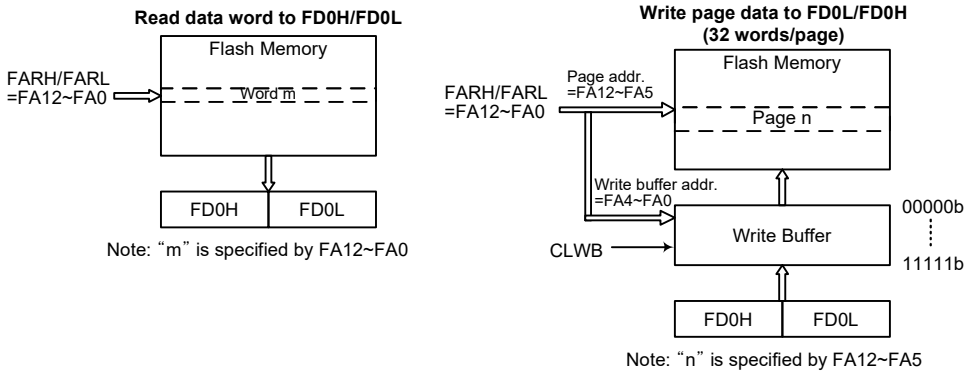
操作	格式
擦除	1 页 / 次
写入	32 字 / 次
读出	1 字 / 次
注：页大小 = 写入缓冲器大小 = 32 字	

IAP 操作格式

擦除页	FARH	FARL[7:5]	FARL[4:0]
0	0000 0000	000	x xxxx
1	0000 0000	001	x xxxx
2	0000 0000	010	x xxxx
3	0000 0000	011	x xxxx
4	0000 0000	100	x xxxx
: :	: :	: :	: :
254	0001 1111	110	x xxxx
255	0001 1111	111	x xxxx

“x”：无关

擦除页序号及选择



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 / 写使能步骤成功使能 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每页 32 字。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 时，会将存储在 FD0L 和 FD0H 数据寄存器内的数据都加载到写入缓冲器，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其它擦 / 写操作。

写入程序结束后，硬件会自动清除写入缓冲器。注意，如果在比对步骤时发现写入到 Flash 存储器的数据不正确，则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

与 IAP 相关的 Flash 存取寄存器有两个地址寄存器、四对 16-bit 数据寄存器和三个控制寄存器。使用地址、数据和控制寄存器可以对 Flash 存储器执行 16 位数据读 / 写操作。内部 Flash 程序存储器所有操作由一系列寄存器控制，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。

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	—	—	—	—	—	—	—	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

- **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 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 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，不会加载到低 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 位写入缓冲器中，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

- **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
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
当此位由应用程序清零后，Flash 存储器擦 / 写功能除能。注意，对此位直接写“1”不会使能擦 / 写功能。此位可用于指示 Flash 存储器擦 / 写功能状态。当此位由硬件置为“1”时，表明 Flash 存储器擦 / 写功能已经成功使能，若为“0”，表明 Flash 存储器擦 / 写功能除能。

Bit 6~4 **FMOD2~FMOD0**：Flash 存储器模式选择
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发生溢出
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：未开始 Flash 存储器写入程序或 Flash 存储器写入程序已完成
1：开始 Flash 存储器写入程序
此位由软件置“1”，当 Flash 存储器写入程序完成后由硬件清零。

Bit 1 **FRDEN**：Flash 存储器读出使能位
0：Flash 存储器读出除能
1：Flash 存储器读出使能
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：未开始 Flash 存储器读出程序或 Flash 存储器读出程序已完成
1：开始 Flash 存储器读出程序
此位由软件置“1”，当 Flash 存储器读出程序完成后由硬件清零。

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
当写入“55H”到该寄存器中，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	—	—	—	—	—	—	—	CLWB
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 未定义，读为“0”

Bit 0 **CLWB**：Flash 存储器写入缓冲器清除控制位
0：未开始写入缓冲器清除或写入缓冲器清除程序已完成
1：开始写入缓冲器清除程序
此位由软件置“1”，当写入缓冲区清除过程完成后由硬件清零。

Flash 存储器擦 / 写流程

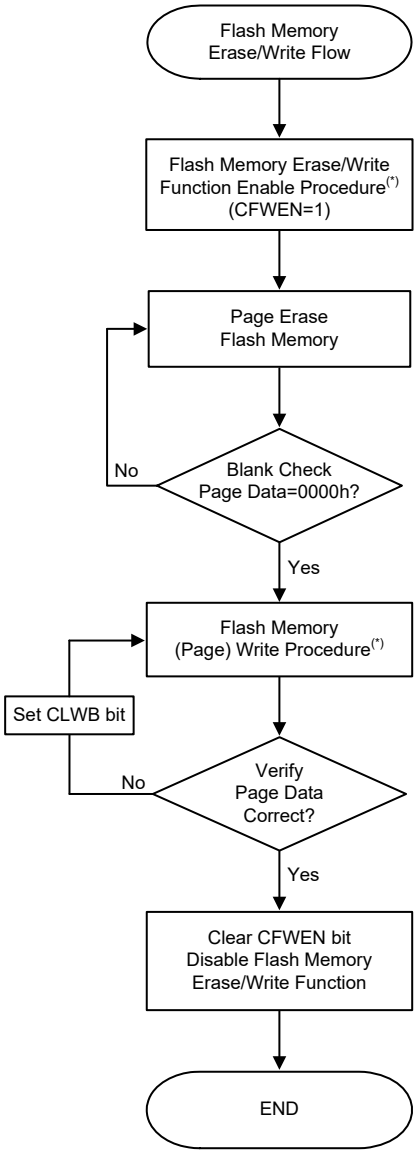
在开始更新 Flash 存储器之前，先了解 Flash 存储器擦 / 写流程操作是很重要的，用户可参考下列步骤进行 IAP 程序开发，以确保 Flash 存储器内容更新正确。

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 Flash 存储器擦或写操作。详细内容请参考“Flash 存储器擦 / 写使能步骤”。
2. 配置 Flash 存储器地址以指定要擦除的页，然后擦除此页。
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 再执行页擦除。

4. 写入数据至该页，详细内容请参考“Flash 存储器写入步骤”。
5. 采用 TABRD 指令进行读取并比对写入数据是否正确，如果读出的数据与写入数据不符，即写入不成功，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 / 写后，如果无需擦 / 写其它页，可清除 CFWEN 位来除能“Flash 存储器擦 / 写使能模式”。



Flash 存储器擦 / 写流程

注：*“Flash 存储器擦 / 写使能步骤”和“Flash 存储器写入步骤”将在后面介绍。

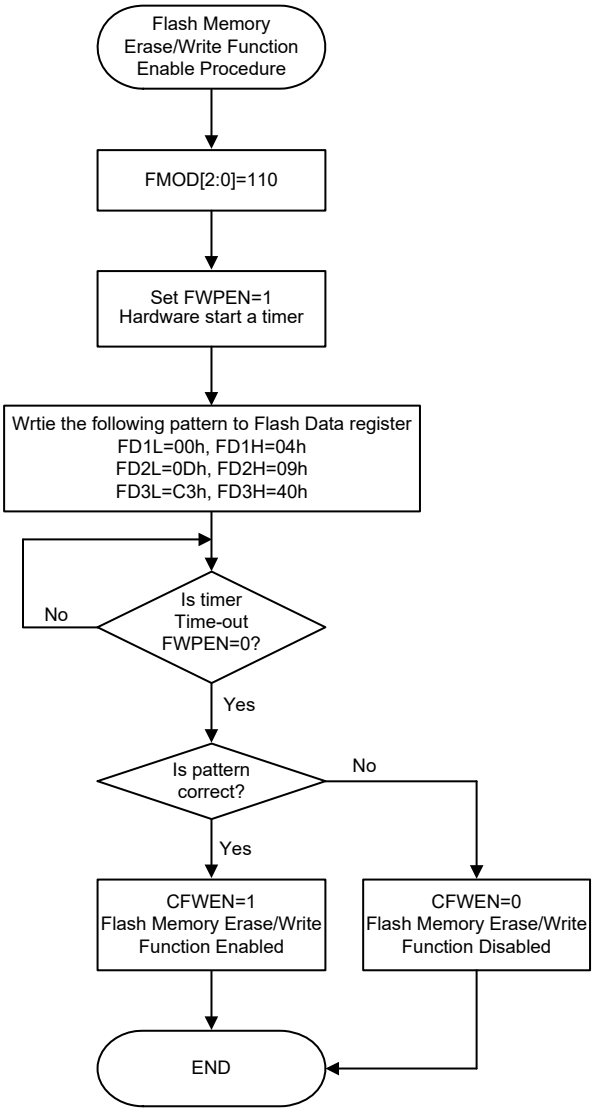
Flash 存储器擦 / 写使能步骤

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 / 写使能步骤说明

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一个内部定时器。
3. 使用者必须在 FWPEN 位置高后尽快填入以下数据序列至 FD1L~FD3L 和 FD1H~FD3H 寄存器中，数据序列依次为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，此时不必再执行以上步骤。



Flash 存储器擦 / 写使能步骤

Flash 存储器写入步骤

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 个字，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步骤说明

对于写入操作每次写入的数据最多为 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 和 FARH，并将第一笔数据依序填入 FD0L 和 FD0H 寄存器。先写 FD0L 再写 FD0H，才会将 FD0L 和 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，因此，要填入第二笔数据时，可不用修改 FARL 和 FARH 重新指定地址。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”，保持在最后一个地址。

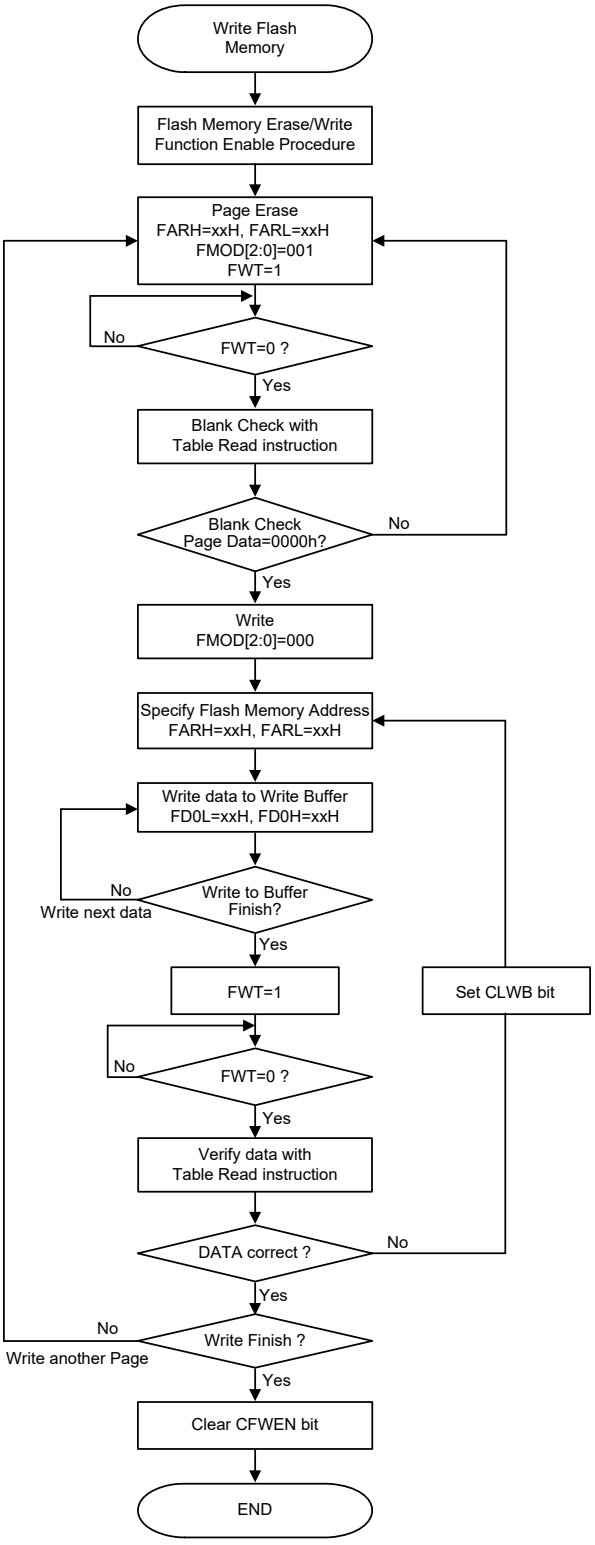
1. 启动 Flash 存储器擦 / 写使能程序，确认 CFWEN 的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦 / 写使能步骤”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式。设定 FWT 位为“1”，擦除 FARH 和 FARL 指定的目标页，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。

   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 和 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 和 FD0H 寄存器。最多可写入 32 个字。
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步骤

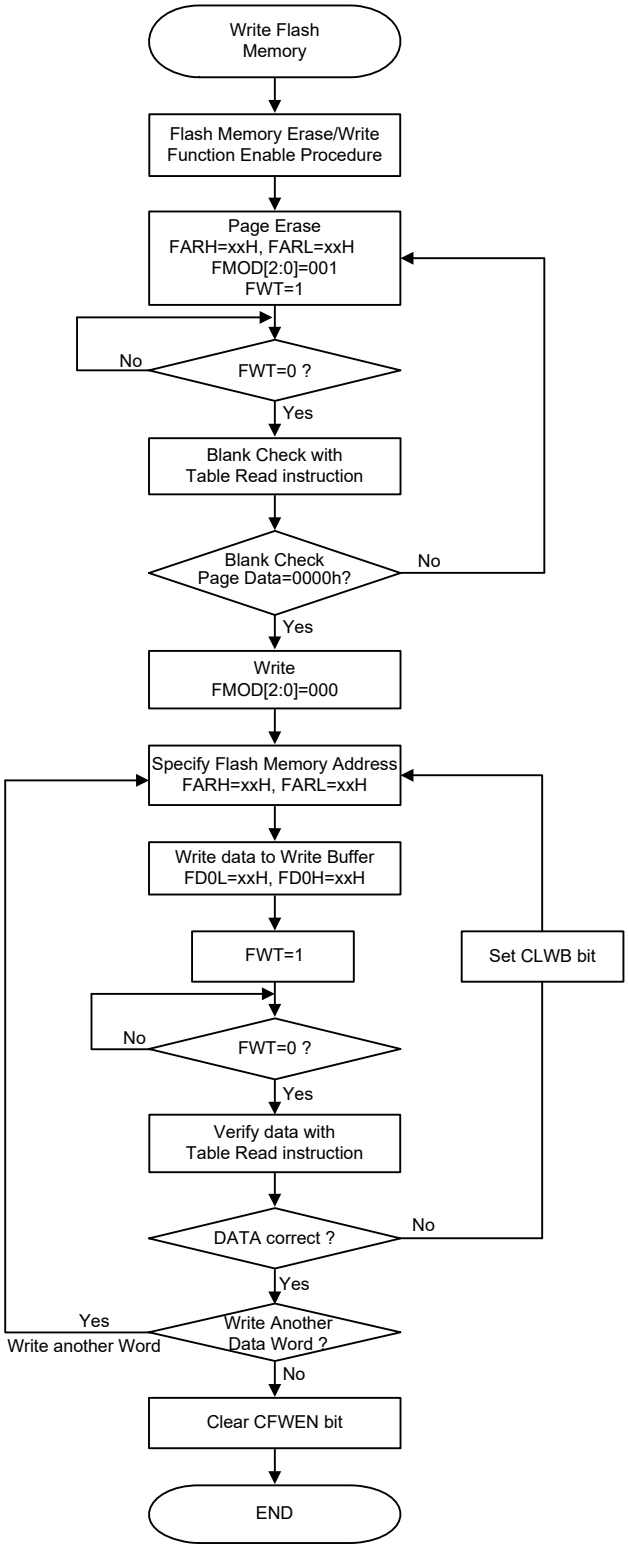
注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。
2. FWT 位由高变低所需时间为 2.2ms (典型值)。

Flash 存储器非连续地址写入步骤说明

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 存储器擦写使能步骤”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式。设定 FWT 位为“1”，擦除 FARH 和 FARL 指定的目标页，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 和 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 和 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步骤

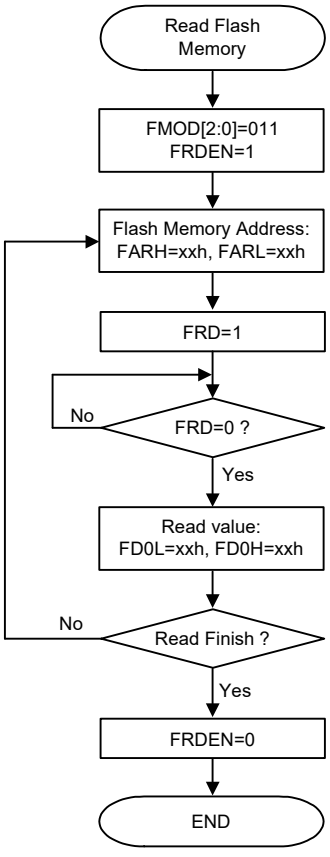
注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。
2. FWT 位由高变低所需时间为 2.2ms (典型值)。

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 / 写使能步骤”。
2. Flash 存储器擦除操作以页为单位进行擦除。
3. 写入缓冲器中的数据填入 Flash 存储器是以页为单位进行的，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，若比对发现写入数据不正确时，通过置高 CLWB 位将写入缓冲器清除，然后重新写入数据。无需清除对应的 Flash 存储器页，直接再写入，然后再比对，直到写入正确。
5. IAP 写入与数据比对时需与最高应用频率相同。

Flash 存储器读出步骤

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 和 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，则可从 FD0H 和 FD0L 寄存器中取得 Flash 存储器中该地址数据。进行 Flash 存储器读出操作前，无需执行 Flash 存储器擦 / 写使能步骤。



Flash 存储器读出步骤

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-bit RAM 内部存储器，用来储存临时数据。

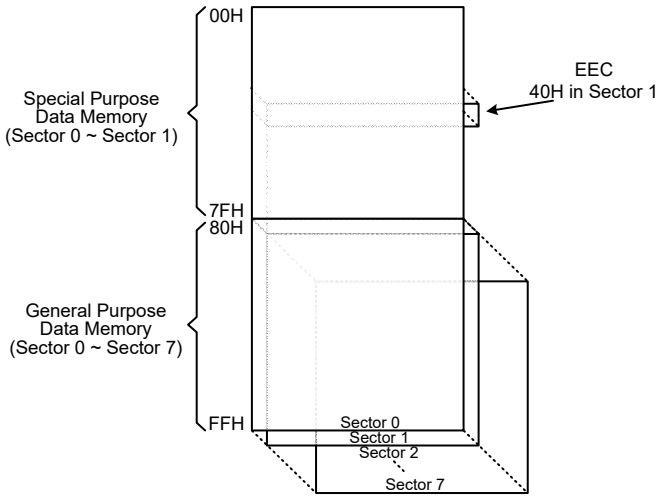
数据存储器分为两部分，第一部分是特殊功能数据存储器。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部分数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。

若用间接寻址方式切换不同的数据存储器 Sector 可通过设置正确的存储器指针值实现。

结构

数据存储器被分为多个 Sector，都位于 8 位存储器中。每个数据存储器 Sector 分为两类，特殊功能数据存储器和通用数据存储器。特殊功能数据存储器地址范围为 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	
所在 Sector	容量	Sector：地址
0：00H~7FH 1：40H (仅 EEC)	1024×8	0：80H~FFH 1：80H~FFH ： 7：80H~FFH



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使用间接寻址访问方式通过 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。所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector，扩展指令可代替间接寻址方式用来访问数据存储器。对于该单片机而言，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 11 个有效位，高字节表示某一 Sector，低字节表示某一指定地址。

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	VBGRC	
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	TB0C	
11H	TB1C	
12H	SCC	
13H	HIRCC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	PB	
19H	PBC	
1AH	PBPU	
1BH	SLEDC0	
1CH	SLEDC1	
1DH	PSCR	
1EH	LVDC	
1FH		
20H	SDSW	
21H	SDPGAC0	
22H	SDPGAC1	
23H	SDA0C	
24H	SDA0VOS	
25H	SDA1C	
26H	SDA1VOS	
27H	STM0C0	
28H	STM0C1	
29H	STM0DL	
2AH	STM0DH	
2BH	STM0AL	
2CH	STM0AH	
2DH	SADOL	
2EH	SADOH	
2FH	SADC0	
30H	SADC1	
31H	PLTSW	
32H	PLTDACC	
33H	PLTDA0L	
34H	PLTDA1L	
35H	PLTDA2L	
36H	PLTC0C	
37H	PLTC0VOS	
38H	PLTC1C	
39H	PLTC1VOS	
3AH	PLTCHYC	
3BH	PLTAC	
3CH	PLTAVOS	
3DH	WDTC	
3EH	EEA	
3FH	EED	

	Sector 0	Sector 1
40H		EEC
41H	SIMC0	
42H	SIMC1	
43H	SIMD	
44H	SIMA/SIMC2	
45H	SIMTOC	
46H	MFI	
47H	INTEG	
48H	INTC0	
49H	INTC1	
4AH	INTC2	
4BH	INTC3	
4CH	PAS0	
4DH	PAS1	
4EH	PBS0	
4FH	PBS1	
50H	PCS0	
51H	PCS1	
52H	PTMC0	
53H	PTMC1	
54H	PTMC2	
55H	PTMDL	
56H	PTMDH	
57H	PTMAL	
58H	PTMAH	
59H	PTMBL	
5AH	PTMBH	
5BH	PTMRPL	
5CH	PTMRPH	
5DH	ISGENC	
5EH	ISGDATA0	
5FH	ISGDATA1	
60H	STM1C0	
61H	STM1C1	
62H	STM1DL	
63H	STM1DH	
64H	STM1AL	
65H	STM1AH	
66H	PC	
67H	PCC	
68H	PCPU	
69H	USR	
6AH	UCR1	
6BH	UCR2	
6CH	TXR_RXR	
6DH	BRG	
6EH	DAH	
6FH	DAL	
70H	DACC	
71H	IFS0	
72H	IFS1	
73H	FC0	
74H	FC1	
75H	FC2	
76H	FARL	
77H	FARH	
78H	FD0L	
79H	FD0H	
7AH	FD1L	
7BH	FD1H	
7CH	FD2L	
7DH	FD2H	
7EH	FD3L	
7FH	FD3H	

: Unused, read as 00H

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，但其并没有实际的物理地址。间接寻址的方法准许使用存储器指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器 IAR0、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 RAM 地址的区块，它们已事先定义成地址 adres1 到 adres4。

间接寻址程序范例 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:
```

间接寻址程序范例 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范例

```
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。

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

该单片机内建 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器空间，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

该单片机的 EEPROM 数据存储器容量为 128×8 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 0 中的一个地址寄存器和一个数据寄存器以及 Sector 1 中的一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

有三个寄存器控制内部 EEPROM 数据存储器总的操作。地址寄存器 EEA、数据寄存器 EED 及控制寄存器 EEC。EEA 和 EED 位于 Sector 0 中，它们能像其它特殊功能寄存器一样直接被访问。EEC 位于 Sector 1 中，只能通过 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
EEA	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM 寄存器列表

- **EEA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7　　未定义，读为“0”

Bit 6~0　　**EEA6~EEA0：**数据 EEPROM 地址 Bit 6 ~ Bit 0

- **EED 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
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　　**EED7~EED0：**数据 EEPROM 数据 Bit 7 ~ Bit 0

- **EEC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WREN	WR	RDEN	RD
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 **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。

Bit 2 **WR**：EEPROM 写控制位
0：写周期结束
1：写周期有效
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**：EEPROM 读控制位
0：读周期结束
1：读周期有效
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 WREN、WR、RDEN 和 RD 不能同时置为“1”。
2. 确保 f_{SUB} 时钟在执行写动作前已稳定。
3. 确保写动作完成后才可改写 EEC 寄存器。

从 EEPROM 中读取数据

从 EEPROM 中读取数据，EEPROM 中读取数据的地址要先放入 EEA 寄存器中。EEC 寄存器中的读使能位 RDEN 先置为高以使能读功能。若 EEC 寄存器中的 RD 位被置高，一个读周期将开始。若 RD 位已置为高而 RDEN 位还未被设置则不能开始读操作。若读周期结束，RD 位将自动清除为“0”，数据可以从 EED 寄存器中读取。数据在其它读或写操作执行前将一直保留在 EED 寄存器中。应用程序将轮询 RD 位以确定数据可以有效地被读取。

写数据到 EEPROM

写数据至 EEPROM，EEPROM 中写入数据的地址要先放入 EEA 寄存器中，写入的数据需存入 EED 寄存器中。EEC 寄存器中的写使能位 WREN 先置为高以使能写功能，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。总中断位 EMI 在写周期开始前应当被清零，写周期开始后再将其使能。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 位将自动清除为“0”，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 及 MP2H 将重置为“0”，这意味着数据存储器 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 写周期结束后将产生 EEPROM 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 写周期结束，DEF 请求标志位将被置位。若总中断和 EEPROM 使能且堆栈未满的情况下将跳转到 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，更多内容请参考“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。

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。写周期执行前总中断位 EMI 应先清零，写周期开始执行后再将此位重新使能。注意，单片机不应在 EEPROM 读或写操作完全完成之前进入空闲或休眠模式，否则 EEPROM 读或写操作将失败。

程序举例

从 EEPROM 中读取数据 — 轮询法

```
MOV  A, EEPROM_ADRES        ; user defined address
MOV  EEA, A
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  IAR1.1                 ; 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 if no more read operations
                            ; are required
CLR  MP1H
MOV  A, EED                 ; move read data to register
MOV  READ_DATA, A
```

注：对于每一个读操作，即使地址是连续的，都必须重新设置地址寄存器，接着再将 RD 位置高开启一个读周期。

写数据到 EEPROM — 轮询法

```
MOV  A, EEPROM_ADRES        ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA         ; user defined data
MOV  EED, A
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  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。振荡器选择是通过应用程序控制寄存器完成的。

振荡器概述

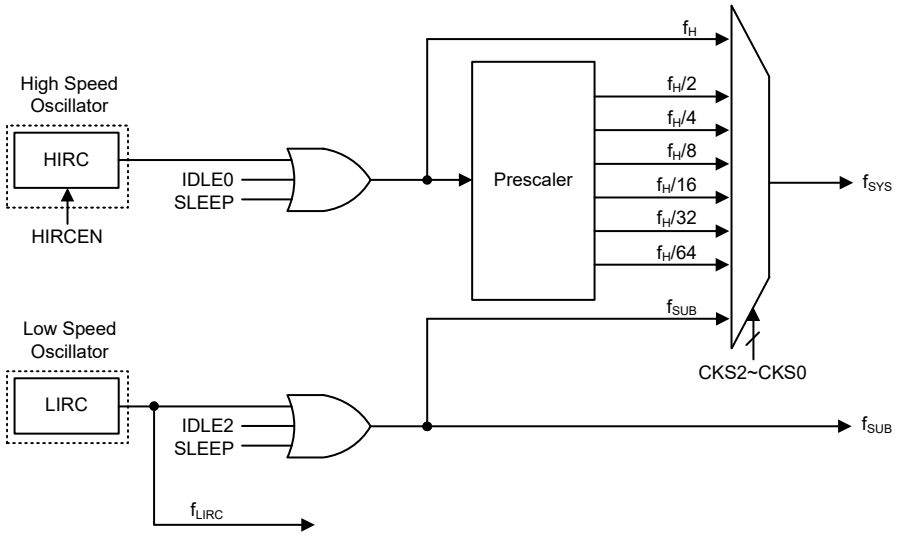
振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。两个完全集成的内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

类型	名称	频率
内部高速 RC	HIRC	2/4/8MHz
内部低速 RC	LIRC	32kHz

振荡器类型

系统时钟配置

此单片机有两个系统振荡器，包括一个高速振荡器和一个低速振荡器。高速振荡器有内部 2/4/8MHz RC 振荡器 – HIRC，低速振荡器有内部 32kHz 低速振荡器 – LIRC。使用高速或低速振荡器作为系统时钟的选择是通过设置 SCC 寄存器中的 CKS2~CKS0 位决定的，系统时钟可动态选择。



系统时钟配置

内部高速 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有三种固定的频率：2MHz，4MHz，8MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD}、温度以及芯片制成工艺不同的影响较大程度地降低。

内部 32kHz 振荡器 – LIRC

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

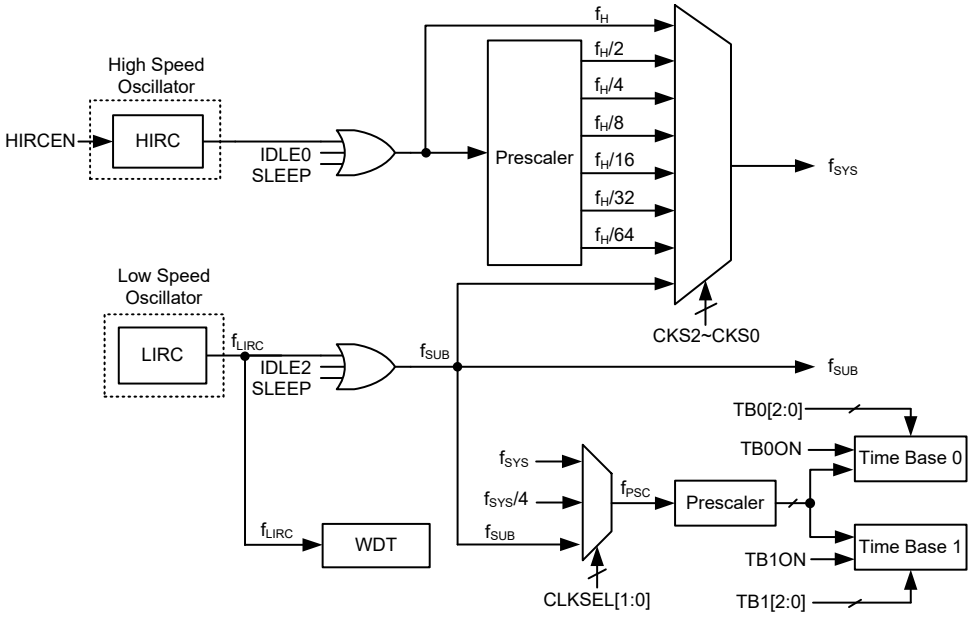
工作模式和系统时钟

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

系统时钟

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

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



单片机时钟配置

注：当系统时钟源 f_{SYS} 由 f_H 切换为 f_{SUB} 时，可以通过设置相应的高速振荡器使能控制位，选择停止以节省耗电，或者继续振荡，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(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} 时钟会开启因为 WDT 功能始终使能。

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 位选择的。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

在 HALT 指令执行后且 FHIDEN 位和 FSIDEN 位为低时，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。由于看门狗定时器功能使能，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 HIRC 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	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN

系统工作模式控制寄存器列表

- **SCC 寄存器**

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

Bit 7~5 **CKS2~CKS0**：系统时钟选择位

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

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

Bit 4~2　未定义，读为“0”

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

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

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	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4　未定义，读为“0”

Bit 3~2　**HIRC1~HIRC0**：HIRC 频率选择位
00：2MHz
01：4MHz
10：8MHz
11：2MHz
当 HIRC 振荡器使能或通过应用程序改变 HIRC 频率选择位时，在 HIRCF 标志位置高后时钟频率会自动改变。

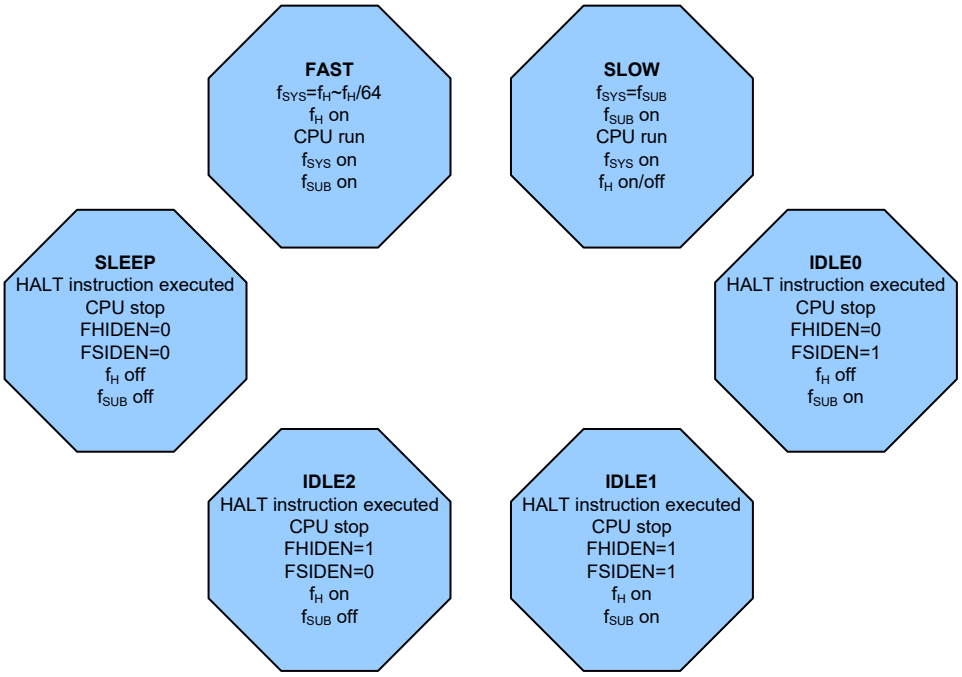
Bit 1　**HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 未稳定
1：HIRC 稳定
此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器或通过应用程序改变 HIRC 振荡器频率选择位，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

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

工作模式切换

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

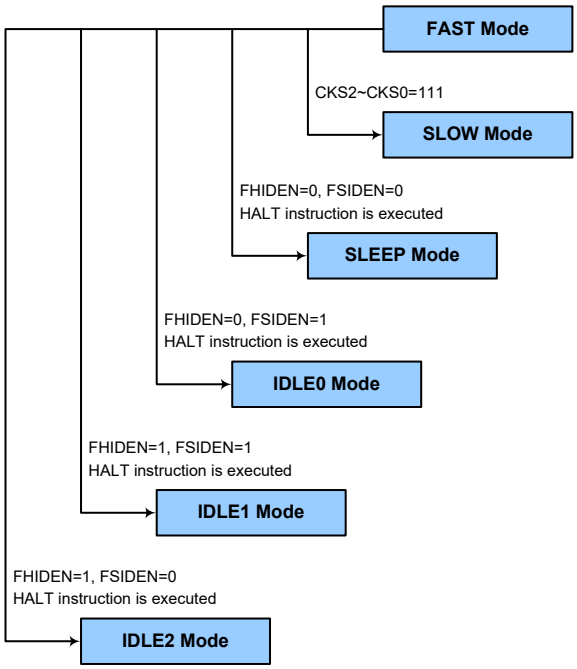
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



快速模式切换到低速模式

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

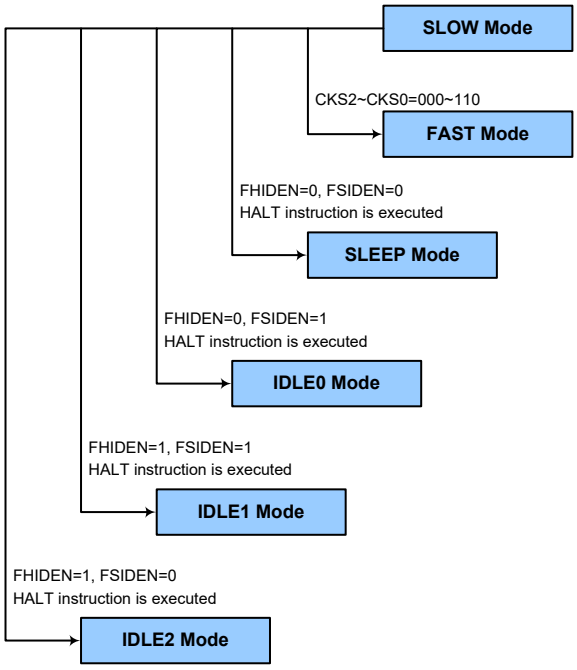
低速模式的时钟源来自 LIRC 振荡器，因此要求此振荡器在所有模式切换动作发生前稳定下来。



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 将被清除。
- 由于 WDT 功能总是使能，WDT 将被清零并重新开始计数。

进入空闲模式 0

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

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

- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 由于 WDT 功能总是使能，WDT 将被清零并重新开始计数。

进入空闲模式 1

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

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

进入空闲模式 2

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

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

待机电流注意事项

由于单片机进入休眠或空闲模式的主要原因是将单片机的电流降低到尽可能低，空闲模式 0 和休眠模式可能到只有几个微安的级别，所以如果要将电路的电流进一步降低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

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

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

唤醒

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

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- PA 口下降沿
- 系统中断
- WDT 溢出

单片机执行 HALT 指令，系统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。

若系统由看门狗定时器溢出唤醒，则会发生看门狗定时器复位，TO 将被置位。看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。

PA 口中的每个引脚都可以通过 PAWU 寄存器使能下降沿唤醒功能。PA 端口唤醒后，程序将在“HALT”指令后继续执行。

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源由内部低速振荡器 LIRC 提供。内部振荡器 LIRC 的频率大约为 32kHz，这个特殊的内部时钟周期会随 V_{DD}、温度和制成的不同而变化。看门狗定时器的时钟源可分频为 2^8~2^{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 寄存器用于选择溢出周期、控制 WDT 功能的使能和 MCU 复位操作。

- **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 功能软件控制位
10101 或 01010：使能
其它：MCU 复位
若因外部环境噪声或软件设置使单片机复位，复位动作发生在一段延迟时间 t_{SRESET} 后，复位后 RSTFC 寄存器中的 WRF 标志位会被置位。

Bit 2~0 **WS2~WS0**：WDT 溢出周期选择位
000：$2^8/f_{LIRC}$
001：$2^{10}/f_{LIRC}$
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	—	WRF
R/W	—	—	—	—	—	R/W	—	R/W
POR	—	—	—	—	—	x	—	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　未定义，读为“0”

Bit 0　**WRF**：WDT 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”，且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，这个清除指令不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。看门狗定时器控制寄存器 WDTC 中的 WE4~WE0 位可提供使能控制以及控制看门狗定时器复位单片机。如果 WE4~WE0 为 01010B 或 10101B 则 WDT 使能；而当设置为“01010B”和“10101B”以外的值时，单片机将在一段延迟时间 t_{SRESET} 后复位。上电后这些位初始化为“01010B”。

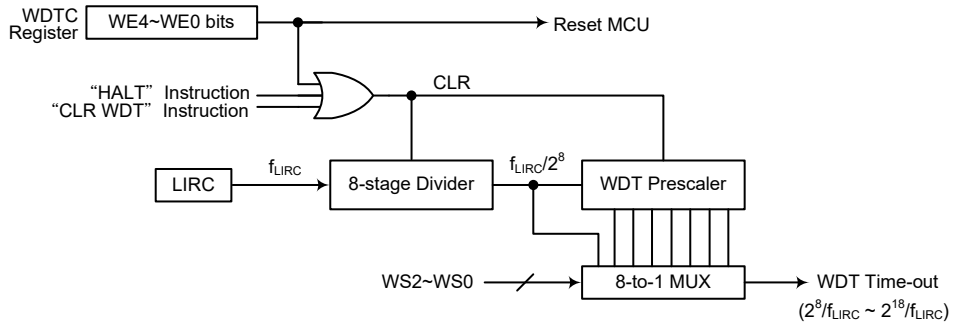
WE4~WE0 位	WDT 功能
01010B 或 10101B	使能
其它值	单片机复位

看门狗定时器使能 / 复位控制

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 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 WDTC 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 WDT 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软件指令清看门狗。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。

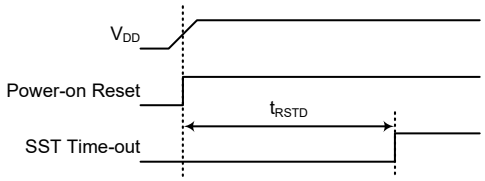
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。另一种复位为看门狗溢出单片机复位，不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机包含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。



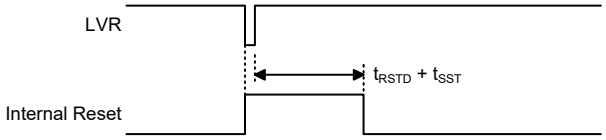
上电复位时序图

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。当电源电压低于某一预定值时，它将复位单片机。

正常运行时 LVR 始终使能，并会设定一个电源复位低电压 V_{LVR}。该单片机的 V_{LVR} 值固定为 2.1V。如果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 LVD & LVR 电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 LVR 将会忽略它且不会执行复位功能。

需要注意的是，当单片机进入空闲或休眠模式，LVR 功能将自动除能。



低电压复位时序图

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~3　未定义，读为“0”

Bit 2　**LVRF**：LVR 功能复位标志位
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　未定义，读为“0”

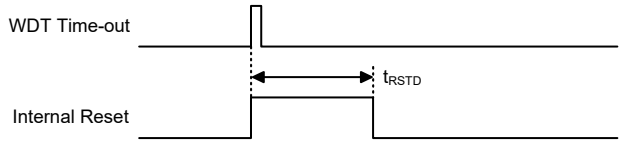
Bit 0　**WRF**：WDT 控制寄存器软件复位标志位
具体描述见看门狗定时器控制寄存器章节。

IAP 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。详见在线应用编程章节。

正常运行时看门狗溢出复位

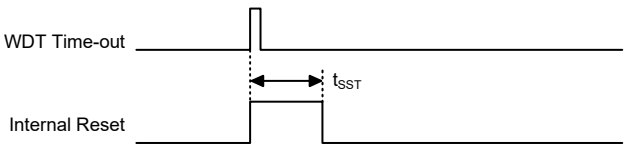
在正常运行即快速模式和低速模式下看门狗溢出复位后 TO 将被设为“1”。



正常运行时看门狗溢出复位时序图

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零及 TO 位被设为 1 外，绝大部份的条件保持不变。图中 t_{SST} 的详细说明请参考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，即存放在状态寄存器中的 PDF 和 TO 位，由休眠或空闲模式功能或看门狗计数器等几种控制器操作控制。复位标志位如下所示：

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

“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

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

不同的复位形式对单片机内部寄存器的影响是不同的。为保证复位后程序能正常执行，了解寄存器在特定条件复位后的设置是非常重要的。下表即为不同方式复位后内部寄存器的状况。

寄存器	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	xx1u uuuu	uu11 uuuu
VBGRC	---- ---0	---- ---0	---- ---u
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -x-0	---- -u-u	---- -u-u
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
SCC	000- --00	000- --00	uuu- --uu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
HIRCC	---- 0001	---- 0001	---- uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
SLEDC0	0000 0000	0000 0000	uuuu uuuu
SLEDC1	---- 0000	---- 0000	---- uuuu
PSCR	---- --00	---- --00	---- --uu
LVDC	--00 0000	--00 0000	--uu uuuu
SDSW	-000 0000	-000 0000	-uuu uuuu
SDPGAC0	--00 0000	--00 0000	--uu uuuu
SDPGAC1	0000 0000	0000 0000	uuuu uuuu
SDA0C	-00- --00	-00- --00	-uu- --uu
SDA0VOS	0010 0000	0010 0000	uuuu uuuu
SDA1C	-00- --00	-00- --00	-uu- --uu
SDA1VOS	0010 0000	0010 0000	uuuu uuuu
STM0C0	0000 0000	0000 0000	uuuu uuuu
STM0C1	0000 0000	0000 0000	uuuu uuuu
STM0DL	0000 0000	0000 0000	uuuu uuuu
STM0DH	---- --00	---- --00	---- --uu
STM0AL	0000 0000	0000 0000	uuuu uuuu
STM0AH	---- --00	---- --00	---- --uu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
PLTSW	---- -001	---- -001	---- -uuu
PLTDACC	---- -000	---- -000	---- -uuu
PLTDA0L	--00 0000	--00 0000	--uu uuuu
PLTDA1L	--00 0000	--00 0000	--uu uuuu
PLTDA2L	--00 0000	--00 0000	--uu uuuu
PLTC0C	000- 0000	000- 0000	uuu- uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
PLTC0VOS	-001 0000	-001 0000	-uuu uuuu
PLTC1C	000- 0000	000- 0000	uuu- uuuu
PLTC1VOS	-001 0000	-001 0000	-uuu uuuu
PLTCHYC	-000 0000	-000 0000	-uuu uuuu
PLTAC	-00- ---0	-00- ---0	-uu- ---u
PLTAVOS	0010 0000	0010 0000	uuuu uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
EEA	-000 0000	-000 0000	-uuu uuuu
EED	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- uuuu
SIMC0	111- 0000	111- 0000	uuu- uuuu
SIMC1	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA	0000 0000	0000 0000	uuuu uuuu
SIMC2	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	uuuu uuuu
MFI	-000 -000	-000 -000	-uuu -uuu
INTEG	---- 0000	---- 0000	---- uuuu
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
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
PTMC0	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	uuuu uuuu
PTMC2	---- -000	---- -000	---- -uuu
PTMDL	0000 0000	0000 0000	uuuu uuuu
PTMDH	---- --00	---- --00	---- --uu
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	---- --00	---- --00	---- --uu
PTMBL	0000 0000	0000 0000	uuuu uuuu
PTMBH	---- --00	---- --00	---- --uu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	---- --00	---- --00	---- --uu
ISGENC	0--- --00	0--- --00	u--- --uu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
ISGDATA0	---0 0000	---0 0000	---u uuuu
ISGDATA1	---0 0000	---0 0000	---u uuuu
STM1C0	0000 0000	0000 0000	uuuu uuuu
STM1C1	0000 0000	0000 0000	uuuu uuuu
STM1DL	0000 0000	0000 0000	uuuu uuuu
STM1DH	---- --00	---- --00	---- --uu
STM1AL	0000 0000	0000 0000	uuuu uuuu
STM1AH	---- --00	---- --00	---- --uu
PC	--11 1111	--11 1111	--uu uuuu
PCC	--11 1111	--11 1111	--uu uuuu
PCPU	--00 0000	--00 0000	--uu uuuu
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	uuuu uuuu
DAH	0000 0000	0000 0000	uuuu uuuu
DAL	0000 0000	0000 0000	uuuu uuuu
DACC	---- ---0	---- ---0	---- ---u
IFS0	0000 0000	0000 0000	uuuu uuuu
IFS1	---- 0000	---- 0000	---- uuuu
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu
FC2	---- ---0	---- ---0	---- ---u
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

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。

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	PC5	PC4	PC3	PC2	PC1	PC0
PCC	—	—	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0

“—”：未定义，读为“0”

输入 / 输出逻辑功能寄存器列表

上拉电阻

许多产品应用在端口处于输入状态时需要外加一个上拉电阻来实现上拉的功能。为了免去外部上拉电阻，当引脚规划为输入时，可由内部连接到一个上拉电阻。这些上拉电阻可通过相应的上拉控制寄存器 PAPU~PCPU 来设置，它用一个弱 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 Port x 引脚上拉功能控制位
0：除能
1：使能

PxPUn 位用于控制引脚上拉功能。此处的“x”可为 A、B 或 C。每个端口实际的有效位是不同的，具体信息可参考 I/O 逻辑功能寄存器列表。

应注意，PB5，PB6，PB7，PC1 和 PC5 引脚未与外部引脚相连，建议将其相应的 PxPUn 位置高以使能 I/O 引脚的上拉电阻。

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或休眠模式时，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

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

输入 / 输出端口控制寄存器

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

- **PxC 寄存器**

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

PxCn：I/O Port x 引脚输入 / 输出类型选择位
0：输出
1：输入

PxCn 位用于选择引脚输入 / 输出类型。此处的“x”可为 A、B 或 C。每个端口实际的有效位是不同的，具体信息可参考 I/O 逻辑功能寄存器列表。

需要注意的是，如果 RF 引脚功能分别与 MCU 的 PB2 和 PB3 引脚共用，PBC 寄存器中的 PBC2 和 PBC3 位必须正确配置。如果同时使用 MCU 和 RF 引脚功能，建议将 PB2 设置为输出。

对于 PB5，PB6，PB7，PC1 和 PC5 引脚，建议将其相应的 PxCn 位置高以使 I/O 引脚设为输入。

输入 / 输出端口源电流选择

该单片机的每个引脚都支持不同的源电流驱动能力，通过相应的源电流选择位控制。仅当对应的引脚被设为 CMOS 输出时，其源电流选择位才有效。否则，这些选择位无效。用户可参考输入 / 输出口电气特性章节为不同应用选择所需的源电流。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	—	—	—	—	SLEDC13	SLEDC12	SLEDC11	SLEDC10

I/O 口源电流选择寄存器列表

- **SLEDC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
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 **SLEDC07~SLEDC06**：PB4 源电流选择位
00：源电流 = Level 0 (最小)
01：源电流 = Level 1
10：源电流 = Level 2
11：源电流 = Level 3 (最大)

Bit 5~4 **SLEDC05~SLEDC04**：PB3~PB0 源电流选择位
00：源电流 = Level 0 (最小)
01：源电流 = Level 1
10：源电流 = Level 2
11：源电流 = Level 3 (最大)

Bit 3~2 **SLEDC03~SLEDC02**：PA7~PA4 源电流选择位
00：源电流 = Level 0 (最小)
01：源电流 = Level 1
10：源电流 = Level 2
11：源电流 = Level 3 (最大)

Bit 1~0 **SLEDC01~SLEDC00**：PA3~PA0 源电流选择位
00：源电流 = Level 0 (最小)
01：源电流 = Level 1
10：源电流 = Level 2
11：源电流 = Level 3 (最大)

- **SLEDC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SLEDC13	SLEDC12	SLEDC11	SLEDC10
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 **SLEDC13~SLEDC12**：PC4 源电流选择位
00：源电流 = Level 0 (最小)
01：源电流 = Level 1
10：源电流 = Level 2
11：源电流 = Level 3 (最大)

Bit 1~0 **SLEDC11~SLEDC10**：PC3, PC2, PC0 源电流选择位
00：源电流 = Level 0 (最小)
01：源电流 = Level 1
10：源电流 = Level 2
11：源电流 = Level 3 (最大)

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。该单片机包含端口引脚输出功能选择寄存器 PxSn，和输入功能输入引脚选择寄存器 IFSn，这些寄存器可以用来对引脚上的功能进行配置。

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 INTn、xTCKn、xTPnI 等，与对应的通用 I/O 口共用同一个引脚共用设置选项。要选择这个引脚功能，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
IFS0	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00
IFS1	—	—	—	—	IFS13	IFS12	IFS11	IFS10

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
01：SDO
10：TX
11：AN3

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

Bit 3~2 **PAS03~PAS02**：PA1 引脚共用功能选择
00：PA1/INT1
01：SCS
10：A1O
11：A1PI

Bit 1~0 **PAS01~PAS00**：PA0 引脚共用功能选择
00：PA0
01：SCL/SCK
10：PA0
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/STP0I/PTPI
01：SCK/SCL
10：AN1
11：VREF

Bit 5~4 **PAS15~PAS14**：PA6 引脚共用功能选择
00：PA6
01：PTP
10：SDI/SDA
11：RX

Bit 3~2 **PAS13~PAS12**：PA5 引脚共用功能选择
00：PA5/STCK0
01：STP1B
10：A1O
11：PA5/STCK0

Bit 1~0 **PAS11~PAS10**：PA4 引脚共用功能选择
00：PA4/PTCK
01：STP0B
10：AN0
11：A0O

- **PBS0 寄存器**

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

Bit 7~6 **PBS07~PBS06**：PB3 引脚共用功能选择
00：PB3
01：PLRX
10：SDI/SDA
11：RX

由于这些引脚功能和 RF 收发器的 GIO2 引脚共用相同的引脚位置，通过设置 PBS07~PBS06 位使用任何一种引脚功能时，建议通过断开 RF 收发器电源，或者首先确保 RF 收发器处在深度睡眠模式以避免使用 RF 收发器。如果要使用 RF 收发器的 GIO2 引脚，PBS07~PBS06 位必须固定在 00。

Bit 5~4 **PBS05~PBS04**：PB2 引脚共用功能选择
00：PB2
01：PLIS
10：SCK/SCL
11：DACO
由于这些引脚功能和 RF 收发器的 SCK 引脚共用相同的引脚位置，通过设置 PBS05~PBS04 位使用任何一种引脚功能时，建议通过断开 RF 收发器电源，或者首先确保 RF 收发器处在深度睡眠模式以避免使用 RF 收发器。如果要使用 RF 收发器的 SCK 引脚，PBS05~PBS04 位必须固定在 10。

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

Bit 1~0 **PBS01~PBS00**：PB0 引脚共用功能选择
00：PB0/INT0
01：STP0
10：A0PB
11：DACO

- **PBS1 寄存器**

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

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

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

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

Bit 1~0 **PBS11~PBS10**：PB4 引脚共用功能选择
00：PB4
01：$\overline{\text{SCS}}$
10：AN2
11：PB4

- **PCS0 寄存器**

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

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

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

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

Bit 1~0 **PCS01~PCS00**：PC0 引脚共用功能选择
00：PC0
01：RX
10：AN7
11：PC0

- **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：保留
10：保留
11：PC5

Bit 1~0 **PCS11~PCS10**：PC4 引脚共用功能选择
00：PC4/STCK1
01：PTPB
10：PC4/STCK1
11：PC4/STCK1

- **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**：PTPI 输入源选择
00：CXCAP
01：PA7
10：CXCAP
11：CXCAP
注：CXCAP 信号来源于电源线数据收发器的输出信号。

Bit 5~4 **IFS05~IFS04**：$\overline{SCS}$ 输入源引脚选择
00：PB4
01：保留
10：PA1
11：PB4

Bit 3~2 **IFS03~IFS02**：SCK/SCL 输入源引脚选择
00：PB2
01：PA0
10：PA7
11：PC3
注：如果选择 SPI 主机模式，当 SIMEN 位设置为高电平时，PA0、PA7、PB2 和 PC3 引脚都可以用作 SCK 引脚功能，忽略 IFS0[3:2] 位的设置。

Bit 1~0 **IFS01~IFS00**：SDI/SDA 输入源引脚选择
00：PB3
01：PA2
10：PA6
11：PC2

- **IFS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IFS13	IFS12	IFS11	IFS10
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

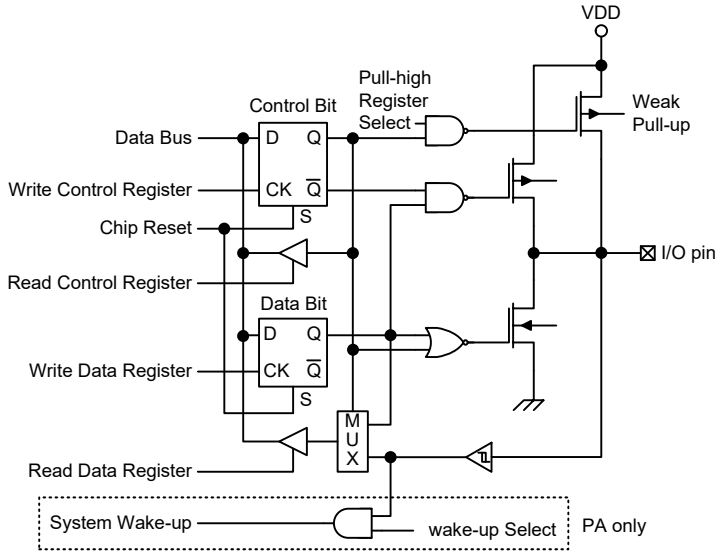
Bit 7~4 未定义，读为“0”

Bit 3~2 **IFS13~IFS12**：INT0 输入源引脚选择
00：PB0
01：PA3
10：PB0
11：PB0

Bit 1~0 **IFS11~IFS10**：RX 输入源引脚选择
00：PB3
01：PA2
10：PA6
11：PC0

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置为输出状态，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适当的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读 - 修改 - 写的操作。单片机需要先读入整个端口上的数据，修改个别的位，然后重新把这些数据写入到输出端口。

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，捕捉输入，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。

这里只介绍各种 TM 的共性，更多详细资料请参考标准型和周期型定时器章节。

简介

该单片机包含 3 个 TM。每个 TM 可被划分为一个特定的类型，即标准型 TM 和周期型 TM。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍标准型和周期型 TM 的共性，更多详细资料分别见后面各章。两种类型 TM 的特性和区别见下表。

功能	STM	PTM
定时 / 计数器	√	√
捕捉输入	√	√
比较匹配输出	√	√
PWM 输出	√	√
单脉冲输出	√	√
PWM 对齐方式	边沿对齐	边沿对齐
PWM 调节周期 & 占空比	占空比或周期	占空比或周期

TM 功能概要

TM 操作

两种不同类型的 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 xTMn 控制寄存器的 xTnCK2~xTnCK0 位，选择所需的时钟源，其中 x 代表 S 或 P 类型，n 代表同一类型 TM 中每个 TM 的编号。由于该单片机只包含一个 PTM，因此这类型的 TM 相关的引脚和控制位不带编号。该时钟源来自系统时钟 f_{SYS} 或内部高速时钟 f_H 的分频比或 f_{SUB} 时钟源或外部 xTCKn 引脚。xTCKn 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

标准型和周期型 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

无论哪种类型的 TM，都有两个 TM 输入引脚 xTCKn 和 xTPnI。xTMn 输入引脚 xTCKn 作为 xTMn 时钟源输入脚，通过设置 xTMnC0 寄存器中的 xTnCK2~

xTnCK0 位进行选择。外部时钟源可通过该引脚来驱动内部 TM。xTCKn 引脚可选择上升沿有效或下降沿有效。xTCKn 引脚还可分别用作 xTMn 单脉冲模式的外部触发引脚。

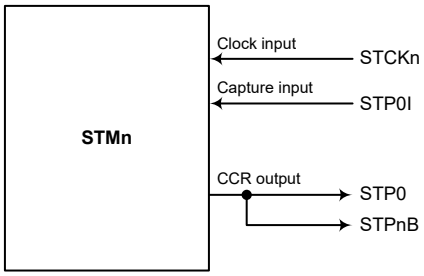
另一种 TM 输入引脚 xTPnI 作为捕捉输入脚，其有效边沿有上升沿、下降沿和双沿，通过设置 xTMnC1 寄存器中的 xTnIO1~xTnIO0 位来选择有效边沿类型。除了 PTPI 引脚外，PTCK 引脚也可用作 PTM 捕捉输入模式的外部触发引脚。

每个 TM 都有一个或两个输出引脚 xTPn 和 xTPnB。xTPnB 信号为 xTPn 输出的反相信号。当 TM 工作在比较匹配输出模式且比较匹配发生时，这些引脚会由 TM 控制切换到高电平或低电平或翻转。外部输出引脚也被 TM 用来产生 PWM 输出波形。

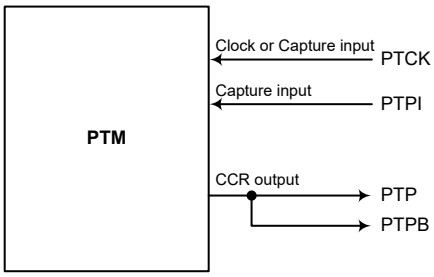
因 TM 输入和输出引脚与其它功能共用，TM 输入和输出功能需要事先通过相关引脚共用功能选择位进行设置。更多引脚共用功能选择详见引脚共用功能章节。

STMn		PTM	
输入	输出	输入	输出
STCKn, STP0I	STP0, STPnB	PTCK, PTPI	PTP, PTPB

TM 外部引脚



STMn 功能引脚方框图 (n=0~1)



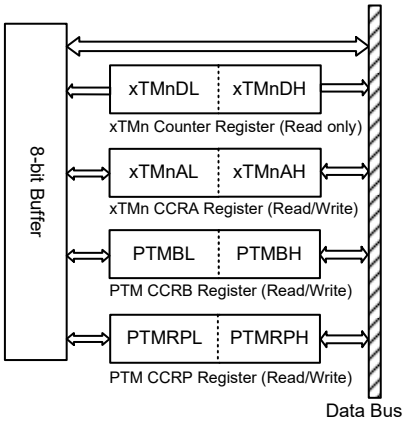
PTM 功能引脚方框图

编程注意事项

TM 计数寄存器和捕捉 / 比较寄存器 CCRA、CCRP 和 PTM CCRB 寄存器，都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。读写这些成对的寄存器需通过特殊的方式。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、CCRP 和 CCRB 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上述的特殊方式。建议使用“MOV”指令按照以下步骤访问 CCRA、

CCRP 和 CCRB 低字节寄存器，即 xTMnAL、PTMRPL 和 PTMBL，否则可能导致无法预期的结果。

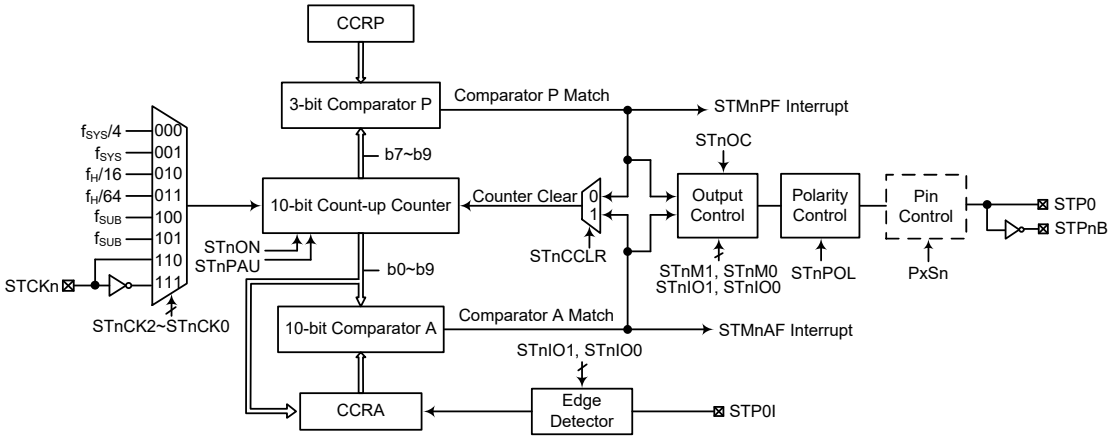


读写流程如下步骤所示：

- 写数据至 CCRA、CCRB 或 CCRP
  - ♦ 步骤 1. 写数据至低字节寄存器 xTMnAL、PTMBL 或 PTMRPL
    – 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 xTMnAH、PTMBH 或 PTMRPH
    – 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、CCRA、CCRB 或 CCRP 中读取数据
  - ♦ 步骤 1. 由高字节寄存器 xTMnDH、xTMnAH、PTMBH 或 PTMRPH 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 xTMnDL、xTMnAL、PTMBL 或 PTMRPL 读取数据
    – 注意，此时读取 8-bit 缓存器中的数据。

标准型 TM – STM

标准型 TM 包括 5 种工作模式，即比较匹配输出，定时 / 事件计数器，捕捉输入，单脉冲输出和 PWM 输出模式。标准型 TM 由两个外部输入脚控制并驱动两个外部输出脚。



注：1. STPnB 信号为 STP0 输出的反相信号。
2. STMn 外部引脚与其它功能共用，所以在使用 STMn 功能前引脚共用功能寄存器必须正确设置。
3. STP1I 和 STP1 这两个引脚不存在。

标准型 TM 方框图 (n=0~1)

标准型 TM 操作

标准型 STMn 核心是一个由用户选择的内部或外部时钟源驱动的 10-bit 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 CCRP 和 CCRA 寄存器中的值进行比较。CCRP 是 3 位宽度，与计数器的高 3 位比较；而 CCRA 是 10 位的，与计数器的所有位比较。

通过应用程序改变 10-bit 计数器值的唯一方法是使 STnON 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 STMn 中断信号。标准型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

标准型 TM 寄存器介绍

标准型 TM 的所有工作模式由一系列寄存器控制。一对只读寄存器用来存放 10 位计数器的值，一对读 / 写寄存器存放 10 位 CCRA 的值。剩下两个控制寄存器设置不同的操作和控制模式以及 3 位 CCRP 的值。

寄存器名称	位							
	7	6	5	4	3	2	1	0
STMnC0	STnPAU	STnCK2	STnCK1	STnCK0	STnON	STnRP2	STnRP1	STnRP0
STMnC1	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
STMnDL	D7	D6	D5	D4	D3	D2	D1	D0
STMnDH	—	—	—	—	—	—	D9	D8
STMnAL	D7	D6	D5	D4	D3	D2	D1	D0
STMnAH	—	—	—	—	—	—	D9	D8

10-bit 标准型 TM 寄存器列表 (n=0~1)

- **STMnC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	STnPAU	STnCK2	STnCK1	STnCK0	STnON	STnRP2	STnRP1	STnRP0
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 **STnPAU**：STMn 计数器暂停控制位
0：运行
1：暂停
通过设置此位为高可使计数器暂停，清零此位恢复正常计数器操作。当处于暂停条件时，STMn 保持上电状态并继续耗电。当此位由低到高转换时，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 **STnCK2~STnCK0**：选择 STMn 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：STCKn 上升沿时钟
111：STCKn 下降沿时钟
此三位用于选择 STMn 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 **STnON**：STMn 计数器 On/Off 控制位
0：Off
1：On
此位控制 STMn 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 STMn。清零此位将停止计数器并关闭 STMn 减少耗电。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 STMn 处于比较匹配输出模式时，当 STnON 位经由低到高的转换时，STMn 输出脚将复位至 STnOC 位指定的初始值。

Bit 2~0 **STnRP2~STnRP0**：STMn CCRP 3-bit 寄存器，与 STMn 计数器 bit 9~bit 7 比较
比较器 P 匹配周期 =
000：1024 个 STMn 时钟周期
001：128 个 STMn 时钟周期
010：256 个 STMn 时钟周期
011：384 个 STMn 时钟周期
100：512 个 STMn 时钟周期
101：640 个 STMn 时钟周期
110：768 个 STMn 时钟周期
111：896 个 STMn 时钟周期
此三位设定内部 CCRP 3-bit 寄存器的值，然后与内部计数器的高三位进行比较。如果 STnCCLR 位设为 0 时，选择 CCRP 比较匹配后清除内部计数器。STnCCLR 位设为低，CCRP 比较匹配结果将重置内部计数器。由于 CCRP 只与计数器高三位比较，比较结果是 128 时钟周期的倍数。CCRP 被清零时，实际上会使得计数器在最大值溢出。

- **STMnC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
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 **STnM1~STnM0**：选择 STMn 工作模式位
00：比较匹配输出模式
01：捕捉输入模式
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 STMn 需要的工作模式。为了确保操作可靠，STMn 应在 STnM1 和 STnM0 位有任何改变前先关掉。在定时 / 计数器模式，STMn 输出引脚状态未定义。

Bit 5~4 **STnIO1~STnIO0**：选择 STMn 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：PWM 输出无效状态
01：PWM 输出有效状态
10：PWM 输出
11：单脉冲输出
捕捉输入模式
00：在 STP0I 上升沿输入捕捉
01：在 STP0I 下降沿输入捕捉
10：在 STP0I 双沿输入捕捉
11：输入捕捉除能
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 STMn 外部引脚如何改变状态。这两位值的选择取决于 STMn 运行在哪种模式下。
在比较匹配输出模式下，STnIO1 和 STnIO0 位决定当从比较器 A 比较匹配输出发生时 STMn 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 STMn 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。STMn 输出脚的初始值通过 STMnC1 寄存器的 STnOC 位设置取得。注意，由 STnIO1 和 STnIO0 位得到的输出电平必须与通过 STnOC 位设置的初始值不同，否则当比较匹配发生时，STMn 输出脚将不会发生变化。在 STMn 输出脚改变状态后，通过 STnON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，STnIO1 和 STnIO0 决定比较匹配条件发生时怎样改变 STP0 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 STMn 关闭时改变 STnO1 和 STnIO0 位的值是很有必要的。若在 STMn 运行时改变 STnIO1 和 STnIO0 的值，PWM 输出的值将无法预料。

Bit 3 **STnOC**：STMn 输出脚 STP0 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 STMn 输出脚输出控制位。它取决于 STMn 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 STMn 处于定时 / 计数器模式，则

其无效。在比较匹配输出模式时，比较匹配发生前其决定 STMn 输出脚 STP0 的逻辑电平值。在 PWM 输出模式 / 单脉冲输出模式时，其决定 PWM 信号是高有效还是低有效。在单脉冲输出模式，其决定 STnON 位由低变高时 STMn 输出脚的逻辑电平。

Bit 2 **STnPOL**：STP0 输出极性控制位
0：同相
1：反相
此位控制 STP0 输出脚的极性。此位为高时 STP0 输出脚反相，为低时 STP0 输出脚同相。若 STMn 处于定时 / 计数器模式时其无效。

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

Bit 0 **STnCCLR**：选择 STMn 计数器清零条件位
0：STMn 比较器 P 匹配
1：STMn 比较器 A 匹配
此位用于选择清除计数器的方法。标准型 TM 包括两个比较器 - 比较器 A 和比较器 P。这两个比较器每个都可以用作清除内部计数器。STnCCLR 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。STnCCLR 位在 PWM 输出，单脉冲输出或输入捕捉模式时未使用。

- **STMnDL 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 STMn 计数器低字节寄存器 bit 7~bit 0
STMn 10-bit 计数器 bit 7~bit 0

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

- **STMnAL 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 STMn CCRA 低字节寄存器 bit 7~bit 0
STMn 10-bit CCRA bit 7~bit 0

- **STMnAH 寄存器**

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

Bit 7~2　未定义，读为“0”

Bit 1~0　STMn CCRA 高字节寄存器 bit 1 ~ bit 0
STMn 10-bit CCRA bit 9 ~ bit 8

标准型 TM 工作模式

标准型 TM 有五种工作模式，即比较匹配输出模式，PWM 输出模式，单脉冲输出模式，捕捉输入模式或定时 / 计数器模式。通过设置 STMnC1 寄存器的 STnM1 和 STnM0 位选择任意模式。

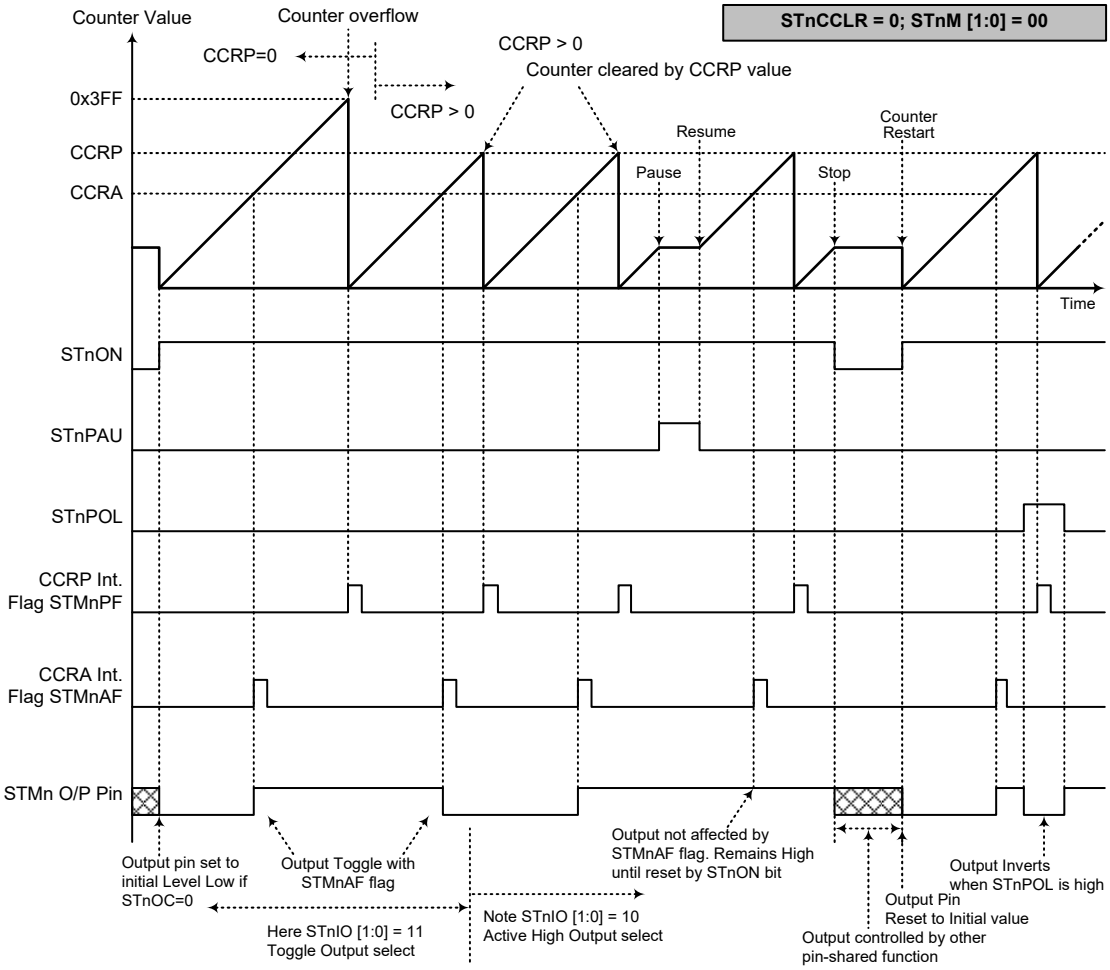
比较匹配输出模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 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 STnCCLR 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 STMnAF 和 STMnPF 将分别置位。

如果 STMnC1 寄存器的 STnCCLR 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，仅产生 STMnAF 中断请求标志。所以当 STnCCLR 为高时 nbvc，不会产生 STMnPF 中断请求标志。在比较匹配输出模式下，CCRA 不能设为“0”。

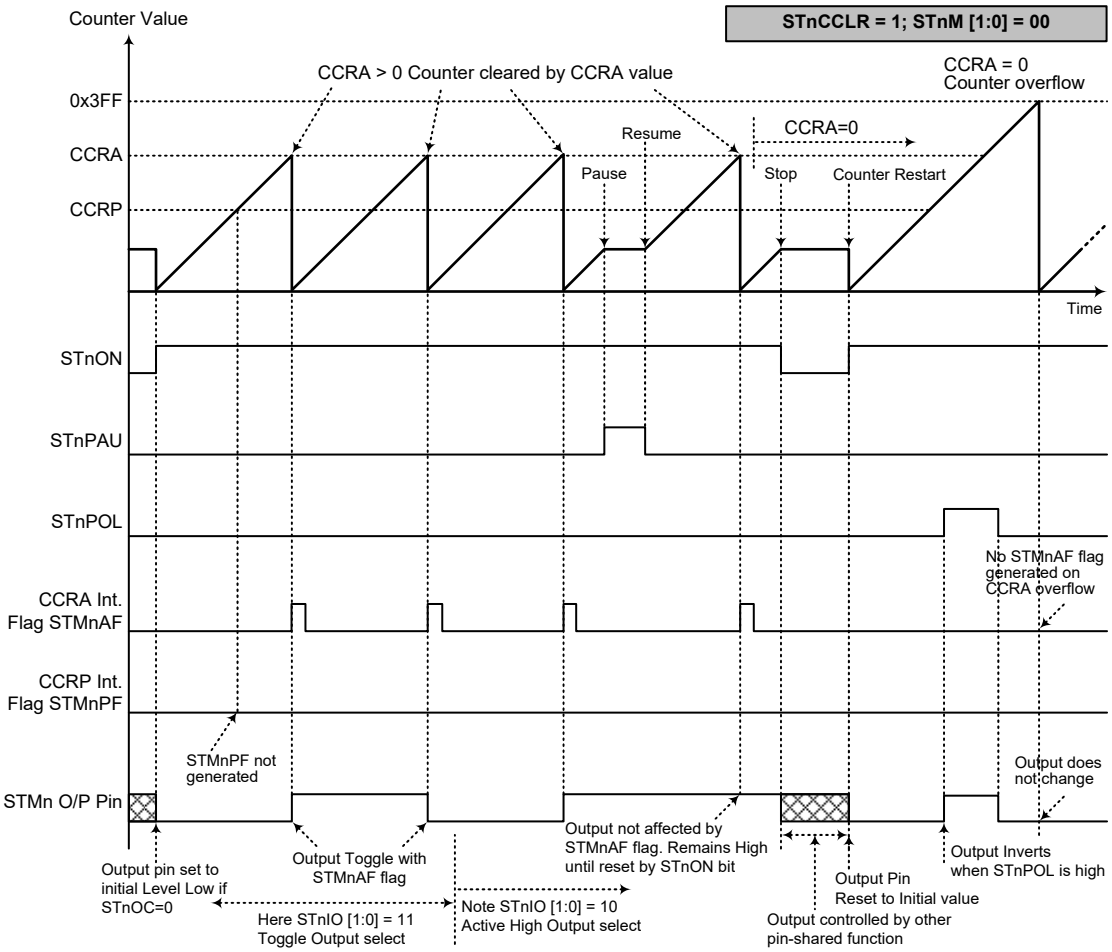
如果 CCRA 位都清为零，当计数器的值达到 10 位最大值 3FFH 时将溢出，但此时不会产生 STMnAF 中断请求标志。

正如该模式名所言，当比较匹配发生后，STMn 输出脚状态改变。当比较器 A 比较匹配发生后 STMnAF 标志产生时，STMn 输出脚状态改变。比较器 P 比较匹配发生时产生的 STMnPF 标志不影响 STMn 输出脚。STMn 输出脚状态改变方式由 STMnC1 寄存器中 STnIO1 和 STnIO0 位决定。当比较器 A 比较匹配发生时，STnIO1 和 STnIO0 位决定 STMn 输出脚输出高，低或翻转当前状态。STMn 输出脚初始值，在 STnON 位由低到高电平的变化后通过 STnOC 位设置。注意，若 STnIO1 和 STnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – STnCCLR=0 (n=0~1)

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



比较匹配输出模式 – STnCCLR=1 (n=0~1)

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

定时 / 计数器模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 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 STMn 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 STMn 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“10”，且 STnIO1 和 STnIO0 位也需要设置为“10”。STMn 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 STMn 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，其波形的选择就较为灵活。在 PWM 输出模式中，STnCCLR 位不影响 PWM 周期。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 STMnC1 寄存器的 STnDPX 位。所以 PWM 波形由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。STMnC1 寄存器中的 STnOC 位决定 PWM 波形的极性，STnIO1 和 STnIO0 位使能 PWM 输出或将 TM 输出脚置为逻辑高或逻辑低。STnPOL 位对 PWM 输出波形的极性取反。

- **10-bit STMn，PWM 输出模式，边沿对齐模式，STnDPX=0**

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

若 f_{SYS}=4MHz，STMn 时钟源为 f_{SYS}/4，CCRP=4，CCRA=128，

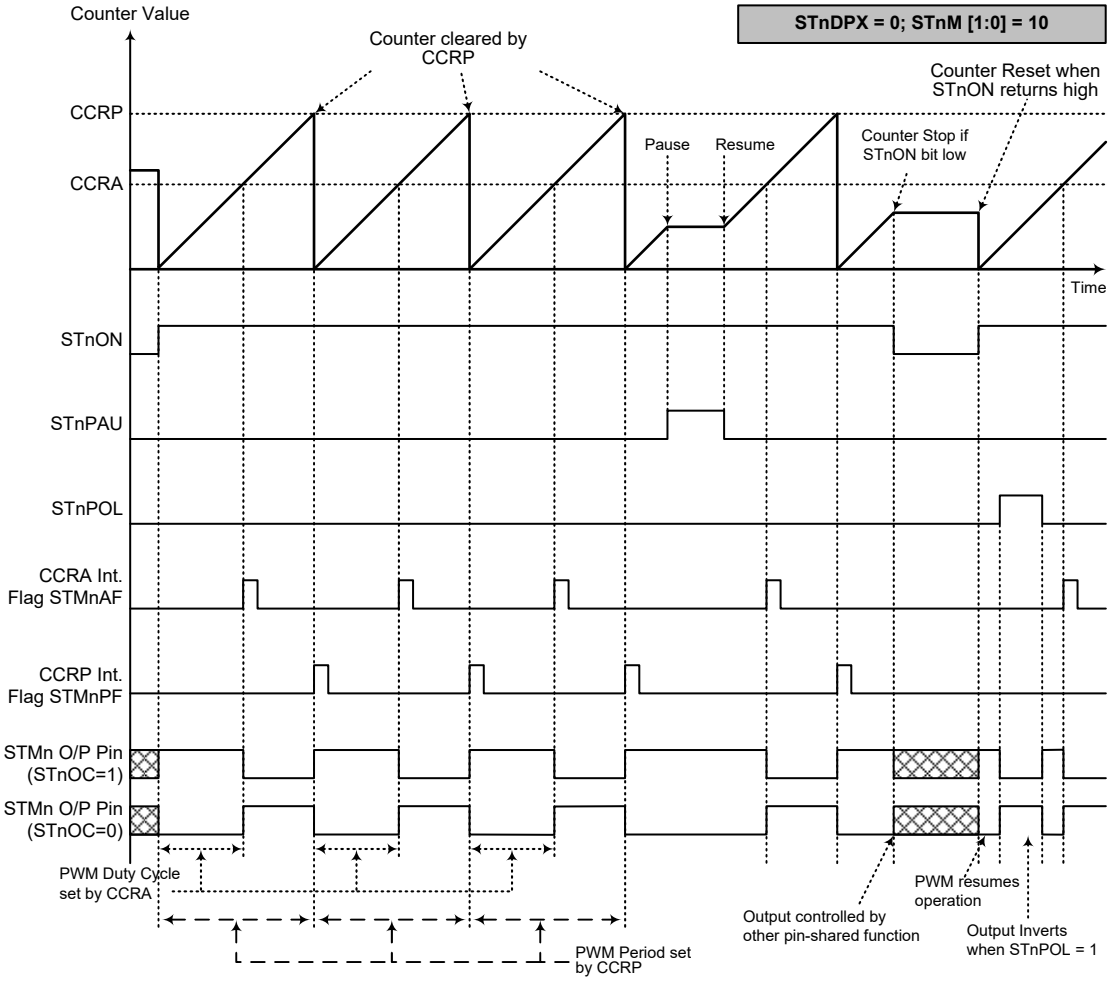
STMn PWM 输出频率 = (f_{SYS}/4)/(4×128)=f_{SYS}/2048=2kHz，duty=128/(4×128)=25%，

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%。

- **10-bit STMn，PWM 输出模式，边沿对齐模式，STnDPX=1**

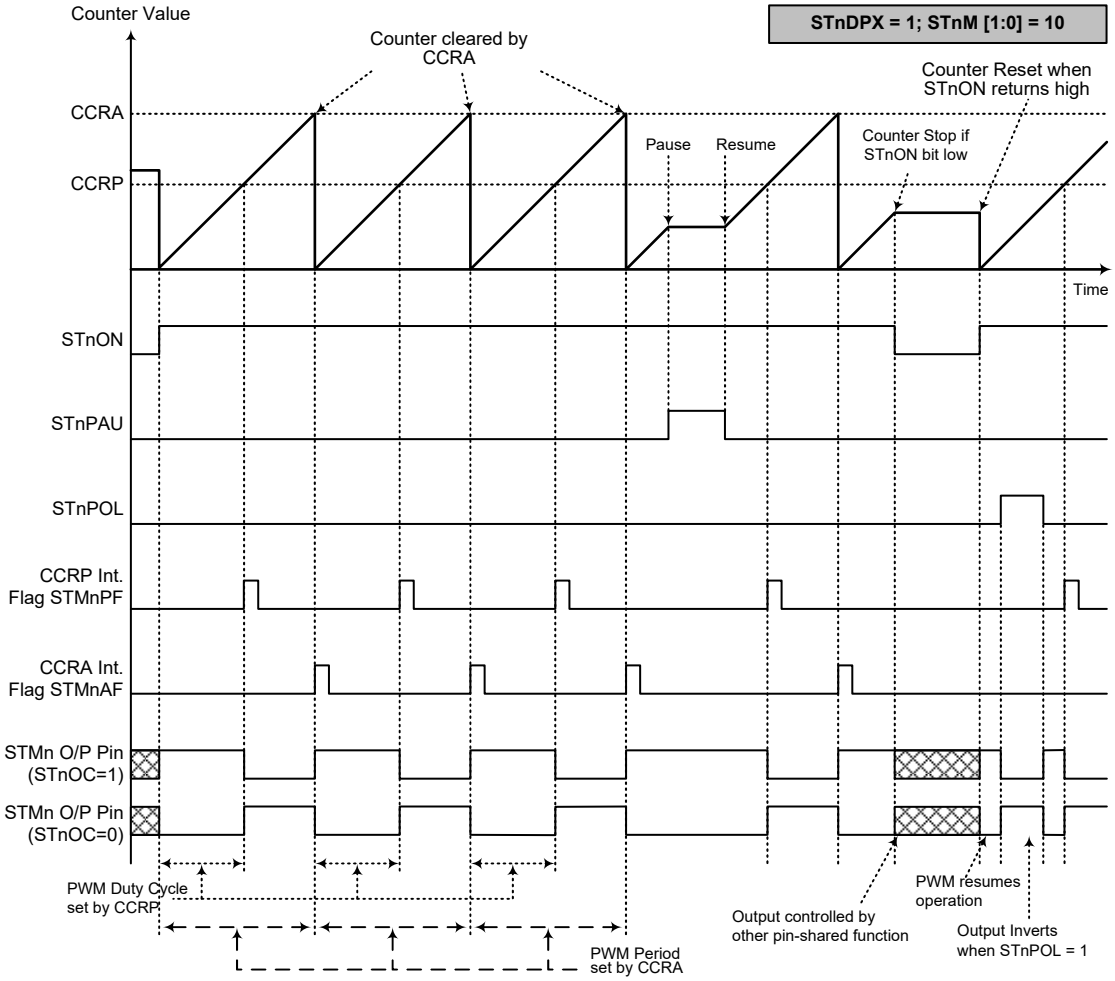
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

PWM 的输出周期由 CCRA 寄存器的值与 STMn 的时钟共同决定，PWM 的占空比由 CCRP×256 (除了 CCRP 为“0”外) 的值决定。



PWM 输出模式 – STnDPX=0 (n=0~1)

注：1. STnDPX=0，CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 STnIO[1: 0]=00 或 01，PWM 功能不变
4. STnCCLR 位不影响 PWM 操作



PWM 输出模式 – STnDPX=1 (n=0~1)

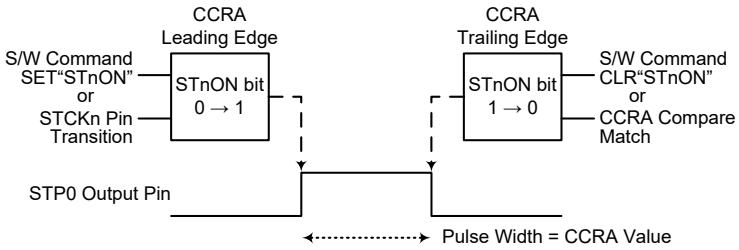
注：1. STnDPX=1，CCRA 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 STnIO[1:0]=00 或 01，PWM 功能不变
4. STnCCLR 位不影响 PWM 操作

单脉冲输出模式

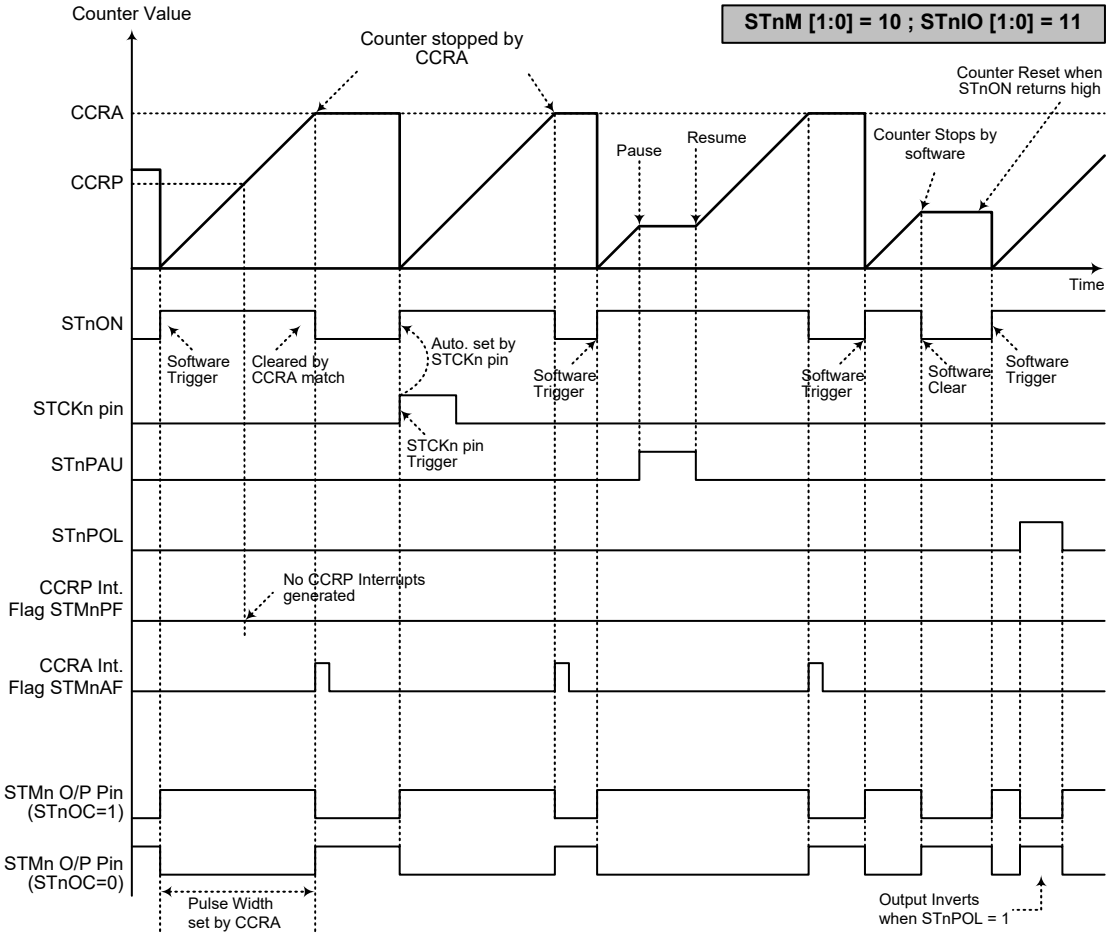
为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“10”，同时 STnIO1 和 STnIO0 位需要设置为“11”。正如模式名所言，单脉冲输出模式，在 STMn 输出脚将产生一个脉冲输出。

脉冲输出可以通过应用程序控制 STnON 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，STnON 位可由 STnCK 脚自动由低转变为高，进而开始单脉冲输出。当 STnON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。当脉冲有效时 STnON 位保持高电平。通过应用程序使 STnON 位清零或比较器 A 比较匹配发生时，产生脉冲下降沿。

然而，比较器 A 比较匹配发生时，会自动清除 STnON 位并产生单脉冲输出边沿跳变。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 STMn 中断。STnON 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，STnCCLR 和 STnDPX 位未使用。



单脉冲产生示意图



单脉冲输出模式 (n=0~1)

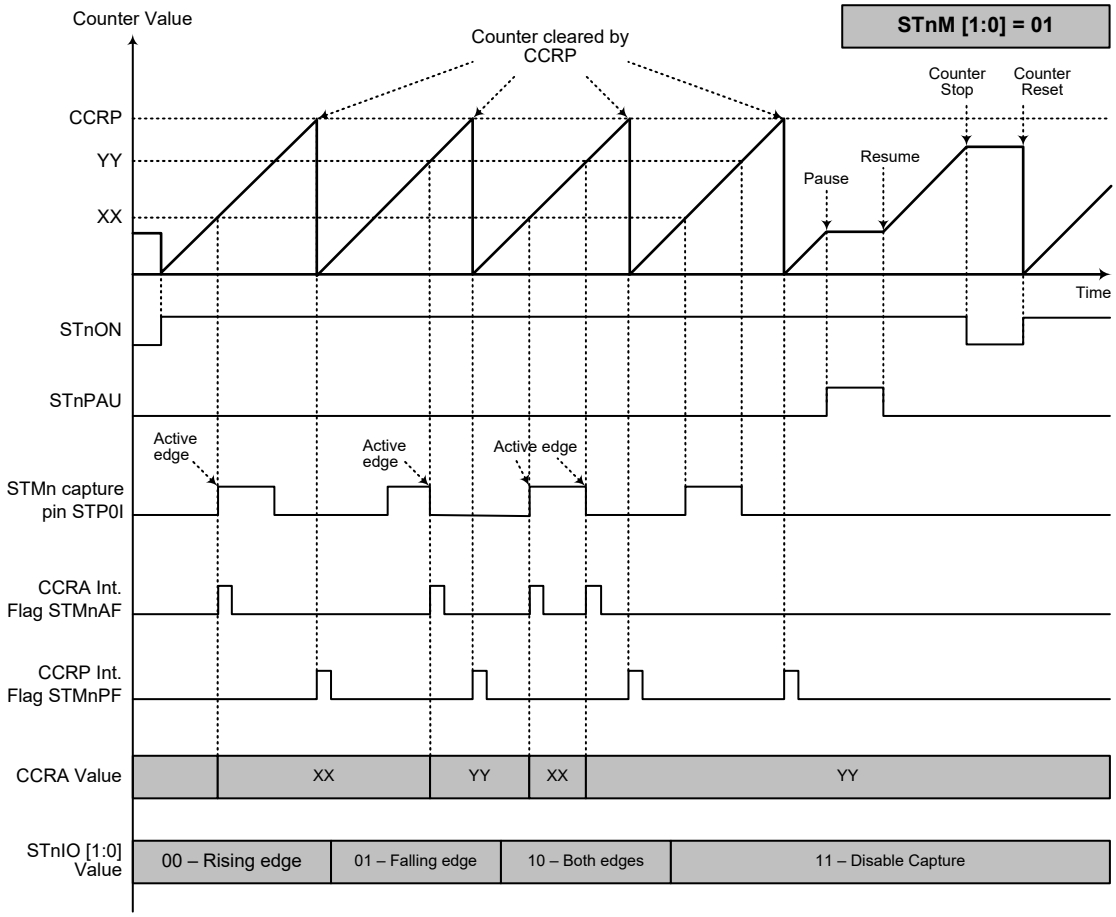
注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 STCKn 脚或设置 STnON 位为高来触发脉冲
4. STCKn 脚有效沿会自动置高位 STnON
5. 单脉冲输出模式中，STnIO[1:0] 需置位“11”，且不能更改

捕捉输入模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“01”。此模式使能外部信号捕捉并保存内部计数器当前值，因此被用于诸如脉冲宽度测量的应用中。STP0I 脚上的外部信号，通过设置 STMnC1 寄存器的 STnIO1 和 STnIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 STnON 位由低置为高时，计数器启动。

当 STP0I 脚出现有效边沿转换时，计数器当前值被锁存到 CCRA 寄存器，并产生 STMn 中断。无论 STP0I 引脚发生哪种边沿转换，计数器继续工作直到 STnON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；CCRP 的值通过这种方式控制计数器的最大值。当比较器 P CCRP 比较匹配发生时，也会产生 STMn 中断。记录 CCRP 溢出中断信号的值可以测量长脉宽。通过设置 STnIO1 和 STnIO0 位选择 STP0I 引脚为上升沿，下降沿或双沿有效。如果 STnIO1 和 STnIO0 都设置为高，无论 STP0I 引脚发生哪种边沿转换都不会产生

捕捉操作，但计数器仍会继续运行。当 STP0I 引脚与其它功能共用，STMn 工作在输入捕捉模式时需多加注意。这是因为如果引脚被设为输出，那么该引脚上的任何电平转变都可能执行输入捕捉操作。STnCCLR 和 STnDPX 位在此模式中未使用。

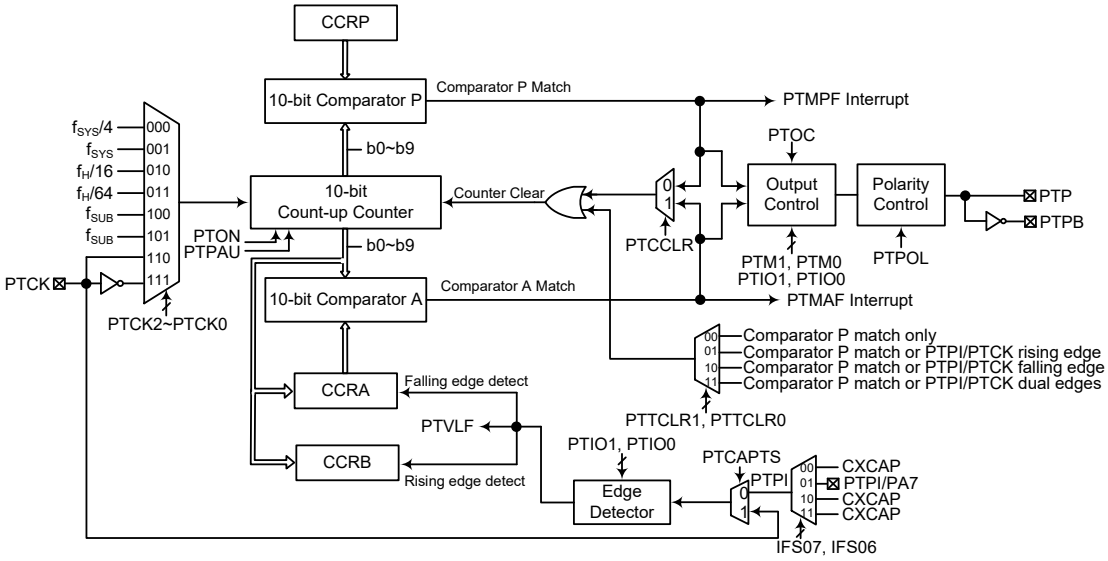


捕捉输入模式 (n=0~1)

注：1. STnM[1:0]=01 并通过 STnIO1 和 STnIO0 位设置有效边沿
2. STMn 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. STnCCLR 位未使用
4. 无输出功能 – STnOC 和 STnPOL 位未使用
5. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大

周期型 TM – PTM

周期型 TM 包括 5 种工作模式，即比较匹配输出、定时 / 事件计数器、捕捉输入、单脉冲输出和 PWM 输出模式。周期型 TM 由两个外部输入脚控制并驱动两个外部输出脚。



注：1. CXCAP 为电源线数据收发器的比较器输出信号。
2. PTM PTPI 信号可通过 IFS0[7:6] 位选择来自外部 PTPI 引脚还是来自内部 CXCAP 信号。
3. PTM 外部引脚与其它功能共用，所以在使用 PTM 功能前引脚共用功能寄存器必须正确设置。

周期型 TM 方框图

周期型 TM 操作

周期型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 10-bit 位向上计数器，它还包括两个内部比较器即比较器 A 和比较器 P。这两个比较器将计数器的值与 CCRA 和 CCRP 寄存器中的值进行比较。CCRP 是 10-bit 的宽度。

通过应用程序改变 10-bit 计数器值的唯一方法是使 PTON 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或在捕捉输入模式下通过设置 PTTCLR[1:0] 位选择有效触发沿也会自动清除计数器。上述条件发生时，通常情况会产生 PTM 中断信号。周期型 TM 可工作在不同的模式，可由包括来自两个输入脚的不同时钟源驱动，也可以控制多个输出脚。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-bit 计数器的值，三对读 / 写寄存器存放 10-bit CCRA 值、CCRP 值和 CCRB 值。剩下三个控制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
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMC2	—	—	—	—	—	PTTCLR1	PTTCLR0	PTVLF
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMBL	D7	D6	D5	D4	D3	D2	D1	D0
PTMBH	—	—	—	—	—	—	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	—	—	—	—	—	—	D9	D8

10-bit 周期型 TM 寄存器列表

- **PTMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTPAU**：PTM 计数器暂停控制位
0：运行
1：暂停
通过设置此位为高可使计数器暂停，清零此位恢复正常计数器操作。当处于暂停条件时，PTM 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 **PTCK2~PTCK0**：PTM 计数器时钟选择位
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{SUB}
101：f_{SUB}
110：PTCK 上升沿
111：PTCK 下降沿
此三位用于选择 PTM 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 **PTON**：PTM 计数器 On/Off 控制位
0：Off
1：On
此位控制 PTM 整体 On/Off 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 PTM。清零此位将停止计数器并关闭 PTM 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 PTM 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 PTON 位经由低到高转换时，PTM 输出脚将复位至 PTOC 位指定的初始值。

Bit 2~0 未定义，读为“0”

- **PTMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
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 **PTM1~PTM0**：PTM 工作模式选择位
00：比较匹配输出模式
01：捕捉输入模式
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 PTM 需要的工作模式。为了确保操作可靠，PTM 应在 PTM1 和 PTM0 位有任何改变前先关掉。在定时 / 计数器模式，PTM 输出引脚状态未知。

Bit 5~4 **PTIO1~PTIO0**：选择 PTM 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：PWM 输出无效状态
01：PWM 输出有效状态
10：PWM 输出
11：单脉冲输出
捕捉输入模式
PTTCLR[1:0]=00B：
00：在 PTPI 或 PTCK 上升沿输入捕捉，计数器值将锁存至 CCRA
01：在 PTPI 或 PTCK 下降沿输入捕捉，计数器值将锁存至 CCRA
10：在 PTPI 或 PTCK 双沿输入捕捉，计数器值将锁存至 CCRA
11：输入捕捉除能
PTTCLR[1:0]=01B、10B 或 11B：
00：在 PTPI 或 PTCK 上升沿输入捕捉，计数器值将锁存至 CCRB
01：在 PTPI 或 PTCK 下降沿输入捕捉，计数器值将锁存至 CCRA
10：在 PTPI 或 PTCK 双沿输入捕捉，下降沿计数器值将锁存至 CCRA，上升沿计数器值将锁存至 CCRB
11：输入捕捉除能
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 PTM 功能如何改变状态。这两位值的选择取决于 PTM 运行在哪种模式下。
在比较匹配输出模式下，PTIO1 和 PTIO0 位决定当从比较器 A 比较匹配输出发生时 PTM 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 PTM 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。PTM 输出脚的初始值通过 PTMC1 寄存器的 PTOC 位设置取得。注意，由 PTIO1 和 PTIO0 位得到的输出电平必须与通过 PTOC 位设置的初始值不同，否则当比较匹配发生时，PTM 输出脚将不会发生变化。在 PTM 输出脚改变状态后，通过 PTON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，PTIO1 和 PTIO0 用于决定比较匹配条件发生时怎样改变 PTM 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 PTM 关闭时改变 PTIO1 和 PTIO0 位的值是很有必要的。若在 PTM 运行时改变 PTIO1 和 PTIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **PTOC**：PTM PTP 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 PTM 输出脚输出控制位。它取决于 PTM 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 PTM 处于定时 / 计数器模式，则其不受影响。在比较匹配输出模式时，其决定比较匹配发生前 PTM 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 PTON 位由低变高时 PTM 输出脚的逻辑电平。

Bit 2 **PTPOL**：PTM PTP 输出极性控制位
0：同相
1：反相
此位控制 PTP 输出脚的极性。此位为高时 PTM 输出脚反相，为低时 PTM 输出脚同相。若 PTM 处于定时 / 计数器模式时其不受影响。

Bit 1 **PTCAPTS**：PTM 捕捉触发源控制位
0：来自 PTPI 输入信号
1：来自 PTCK 引脚

Bit 0 **PTCCLR**：PTM 计数器清零条件选择位
0：PTM 比较器 P 匹配
1：PTM 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。PTCCLR 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。PTCCLR 位在 PWM 输出模式、单脉冲输出模式或输入捕捉模式时未使用。

- **PTMC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PTTCLR1	PTTCLR0	PTVLF
R/W	—	—	—	—	—	R/W	R/W	R
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”

Bit 2~1 **PTTCLR1~PTTCLR0**：捕捉输入模式时 PTM 计数器清零条件选择位
00：比较器 P 比较匹配
01：比较器 P 比较匹配或 PTCK/PTPI 上升沿
10：比较器 P 比较匹配或 PTCK/PTPI 下降沿
11：比较器 P 比较匹配或 PTCK/PTPI 双沿
注意，PTTCLR1~PTTCLR0 位仅在 PTM 处于捕捉输入模式时可用。

Bit 0 **PTVLF**：PTM 计数器值锁存边沿标志位
0：下降沿触发计数器值锁存
1：上升沿触发计数器值锁存
当 PTTCLR1~PTTCLR0 位为 00B 时，忽略该标志位状态。

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

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

- **PTMAL 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**：PTM CCRA 低字节寄存器 bit 7 ~ bit 0
PTM 10-bit CCRA bit 7 ~ bit 0

- **PTMAH 寄存器**

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

Bit 7~2 未定义，读为“0”
Bit 1~0 **D9~D8**：PTM CCRA 高字节寄存器 bit 1 ~ bit 0
PTM 10-bit CCRA bit 9 ~ bit 8

- **PTMBL 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**：PTM CCRB 低字节寄存器 bit 7 ~ bit 0
PTM 10-bit CCRB bit 7 ~ bit 0

- **PTMBH 寄存器**

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

Bit 7~2　未定义，读为“0”

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

- **PTMRPL 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**：PTM CCRP 低字节寄存器 bit 7 ~ bit 0
PTM 10-bit CCRP bit 7 ~ bit 0

- **PTMRPH 寄存器**

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

Bit 7~2　未定义，读为“0”

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

周期型 TM 工作模式

周期型 TM 有五种工作模式，即比较匹配输出模式、PWM 输出模式、单脉冲输出模式、捕捉输入模式或定时 / 计数器模式。通过设置 PTMC1 寄存器的 PTM1 和 PTM0 位选择任意模式。

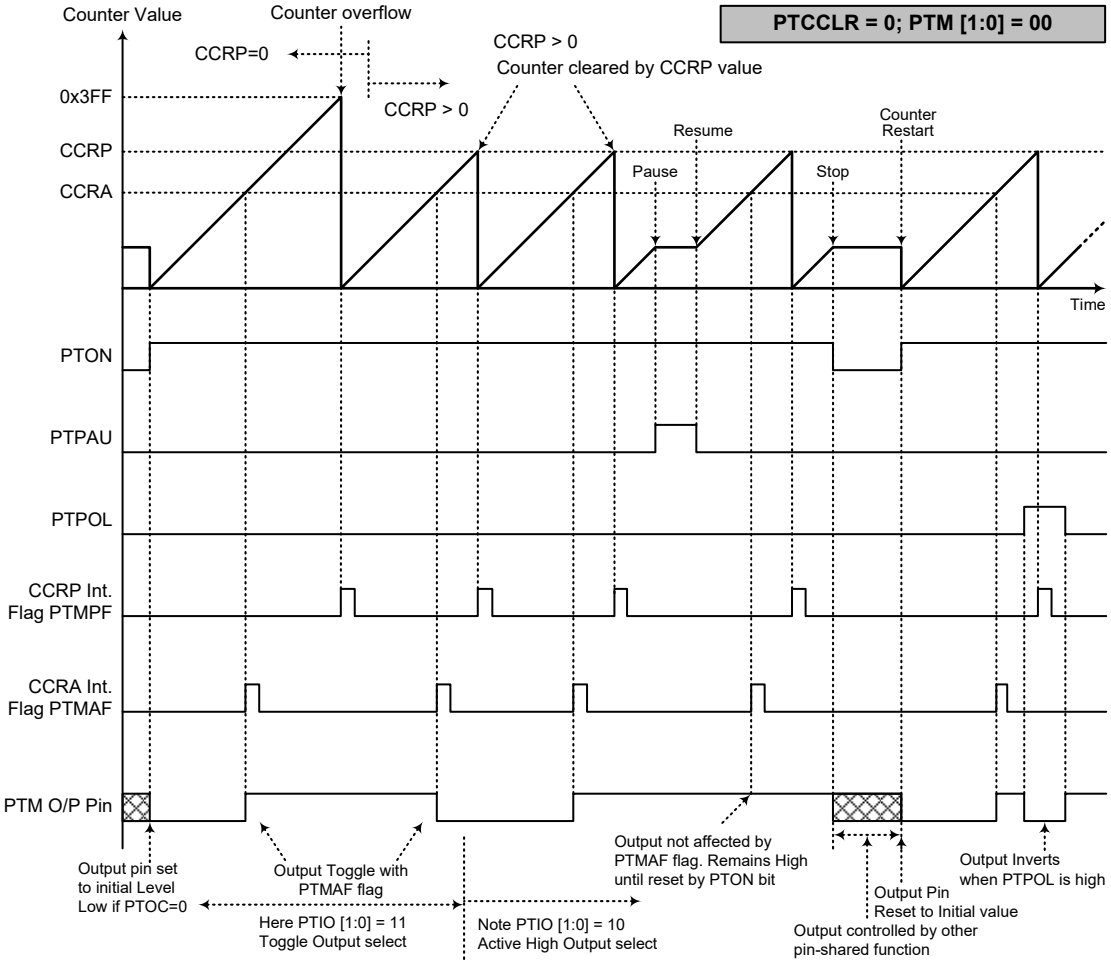
比较匹配输出模式

为使 PTM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 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 PTCCLR 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 PTMAF 和 PTMPF 将分别置起。

如果 PTMC1 寄存器的 PTCCLR 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 PTMAF 中断请求标志产生。所以当 PTCCLR 为高时，不会产生 PTMPF 中断请求标志。在比较匹配输出模式中，CCRA 寄存器值不能设为“0”。

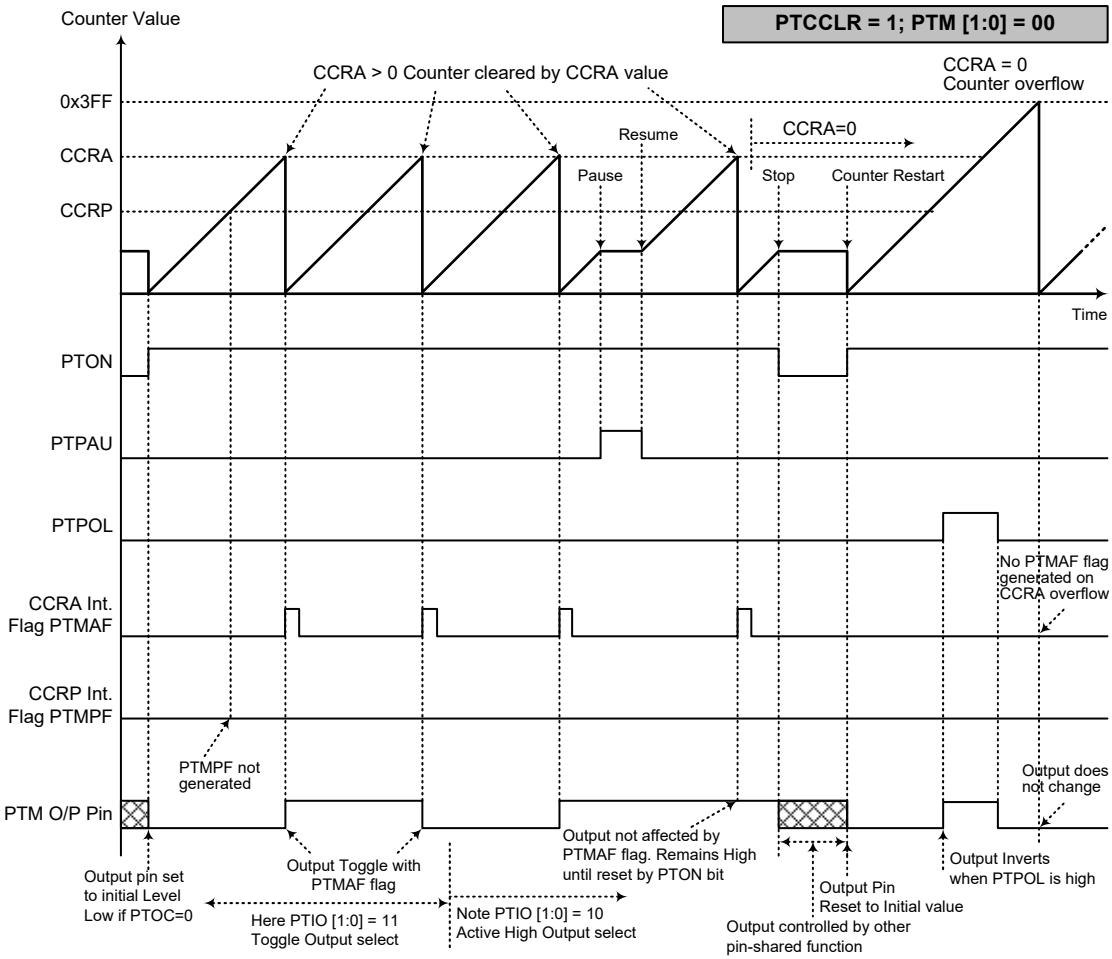
如果 CCRA 位都清除为零，当计数器的值达到 10 位最大值 3FFH 时将溢出，但此时不会产生 PTMAF 中断请求标志。

正如该模式名所言，当比较匹配发生后，PTM 输出脚状态改变。当比较器 A 比较匹配发生后 PTMAF 中断请求标志产生时，PTM 输出脚状态改变。比较器 P 比较匹配发生时产生的 PTMPF 标志不影响 PTM 输出脚。PTM 输出脚状态改变方式由 PTMC1 寄存器中 PTIO1 和 PTIO0 位决定。当比较器 A 比较匹配发生时，PTIO1 和 PTIO0 位决定 PTM 输出脚输出高，低或翻转当前状态。PTM 输出脚初始值，在 PTON 位由低到高电平的变化后通过 PTOC 位设置。注意，若 PTIO1 和 PTIO0 位同时为 0 时，引脚输出不变。



比较器匹配输出模式 – PTCCLR=0

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



比较器匹配输出模式 – PTCCLR=1

注：1. PTCCLR=1，比较器 A 匹配将清除计数器
2. PTM 输出脚仅由 PTMAF 标志位控制
3. 在 PTON 上升沿 PTM 输出脚复位至初始值
4. 当 PTCCLR=1，不产生 PTMPF 标志位

定时 / 计数器模式

为使 PTM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 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 PTM 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 PTM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 PTM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“10”。PTM 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 PTM 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，其波形的选择就较为灵活。在 PWM 输出模式中，PTCCLR 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。PTMC1 寄存器的 PTOC 位选择 PWM 波形的极性，PTIO1 和 PTIO0 位使能 PWM 输出或强制 PTM 输出脚为高电平或低电平。PTPOL 位用于 PWM 输出波形的极性反相控制。

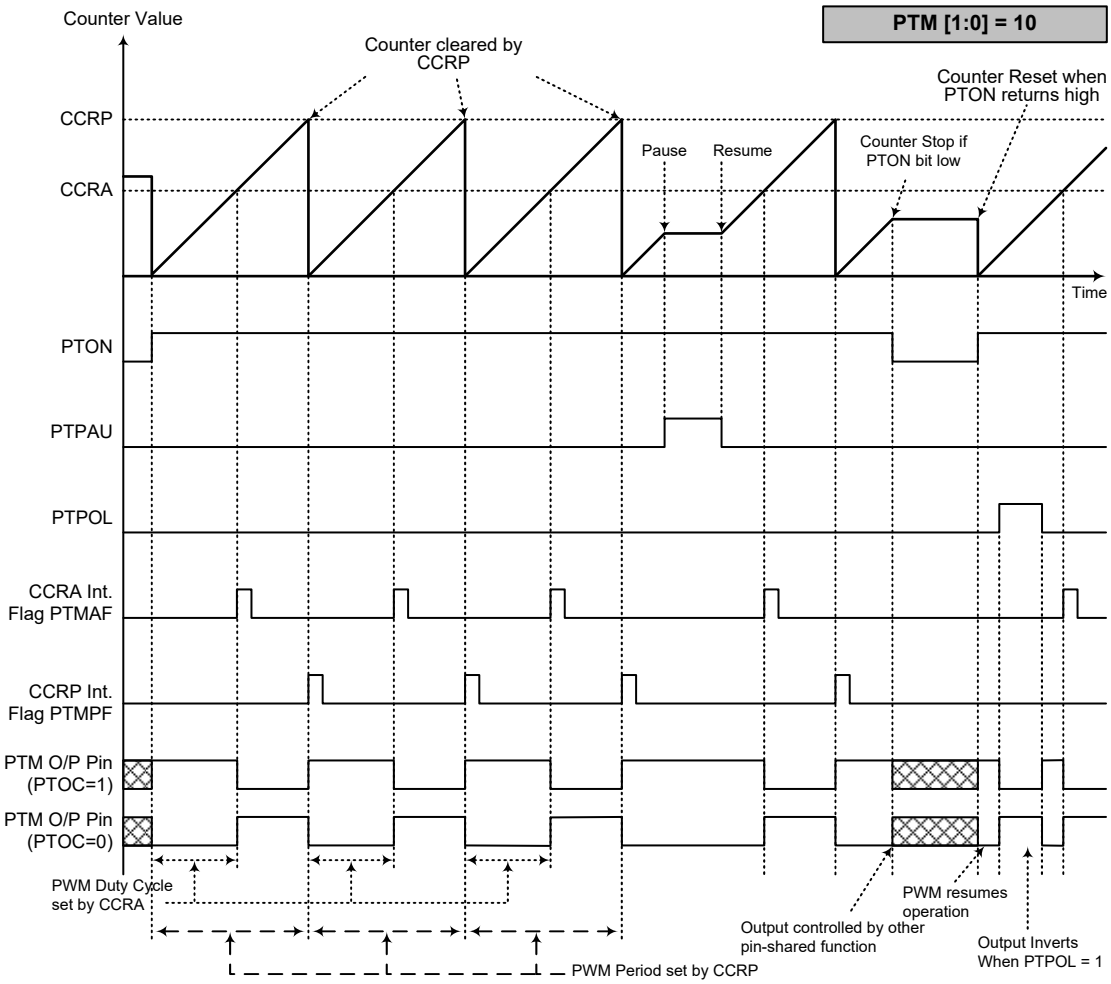
- **10-bit PTM，PWM 输出模式，边沿对齐模式**

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

若 f_{SYS}=8MHz，PTM 时钟源选择 f_{SYS}/4，CCRP=512 且 CCRA=128，

PTM PWM 输出频率 = (f_{SYS}/4)/512=f_{SYS}/2048=4kHz，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

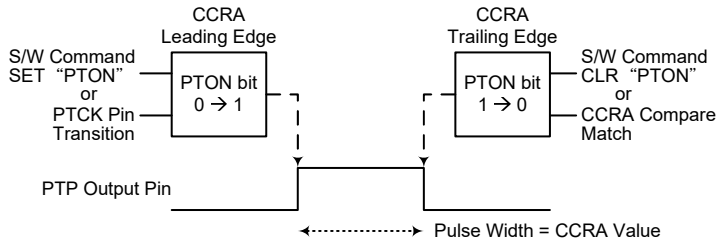
注：1. CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 PTIO[1:0]=00 或 01，PWM 功能不变
4. PTCCLR 位对 PWM 功能无影响

单脉冲输出模式

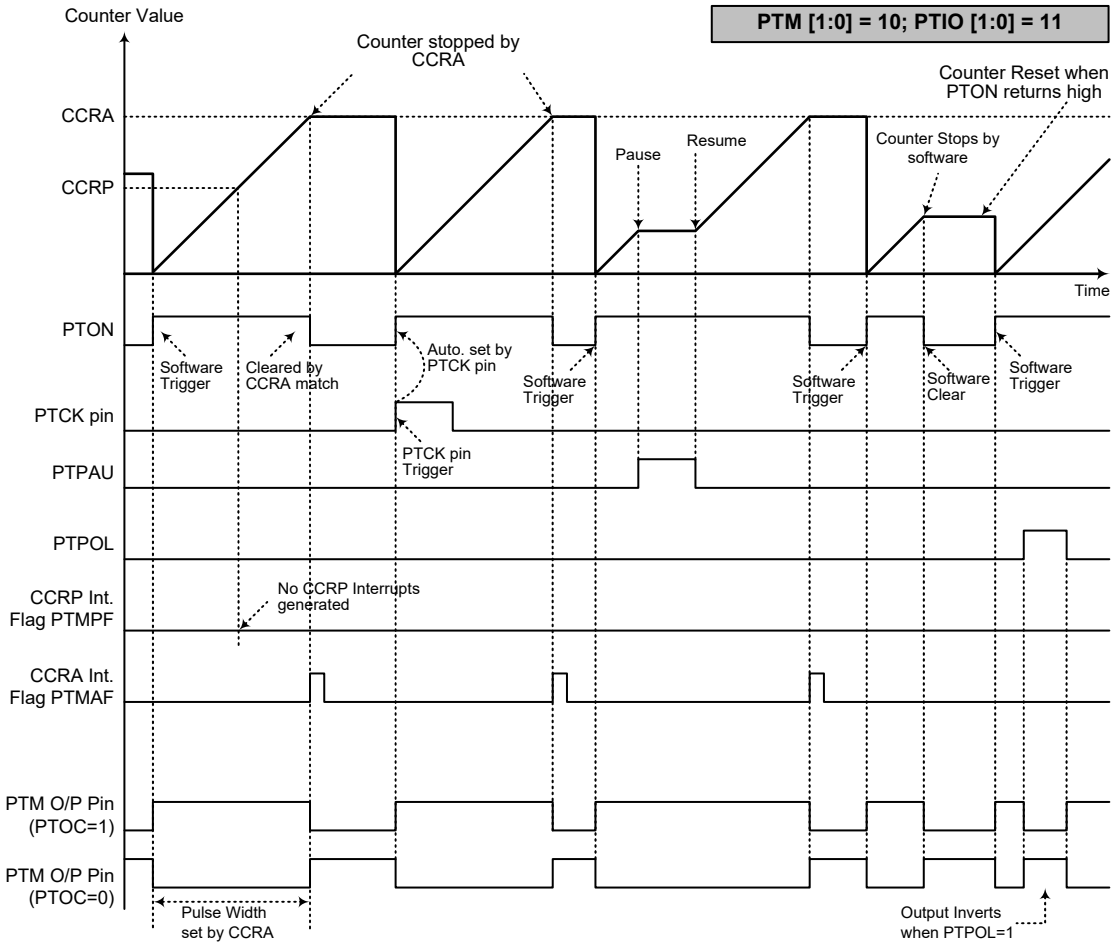
为使 PTM 工作在此模式，PTMC1 寄存器中的 PTM1 和 PTM0 位需要设置为“10”，并且相应的 PTIO1 和 PTIO0 需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTM 输出脚将产生一个脉冲输出。

通过应用程序控制 PTON 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，PTON 位可在 PTCK 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 PTON 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 PTON 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 PTON 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 PTM 中断。PTON 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 PTCCLR 位未使用。



单脉冲产生示意图



单脉输出冲模式

注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCK 脚或设置 PTON 位为高来触发脉冲
4. PTCK 脚有效沿会自动置位 PTON
5. 在单脉冲输出模式，PTIO[1:0] 需设置为“11”且不可改变

捕捉输入模式

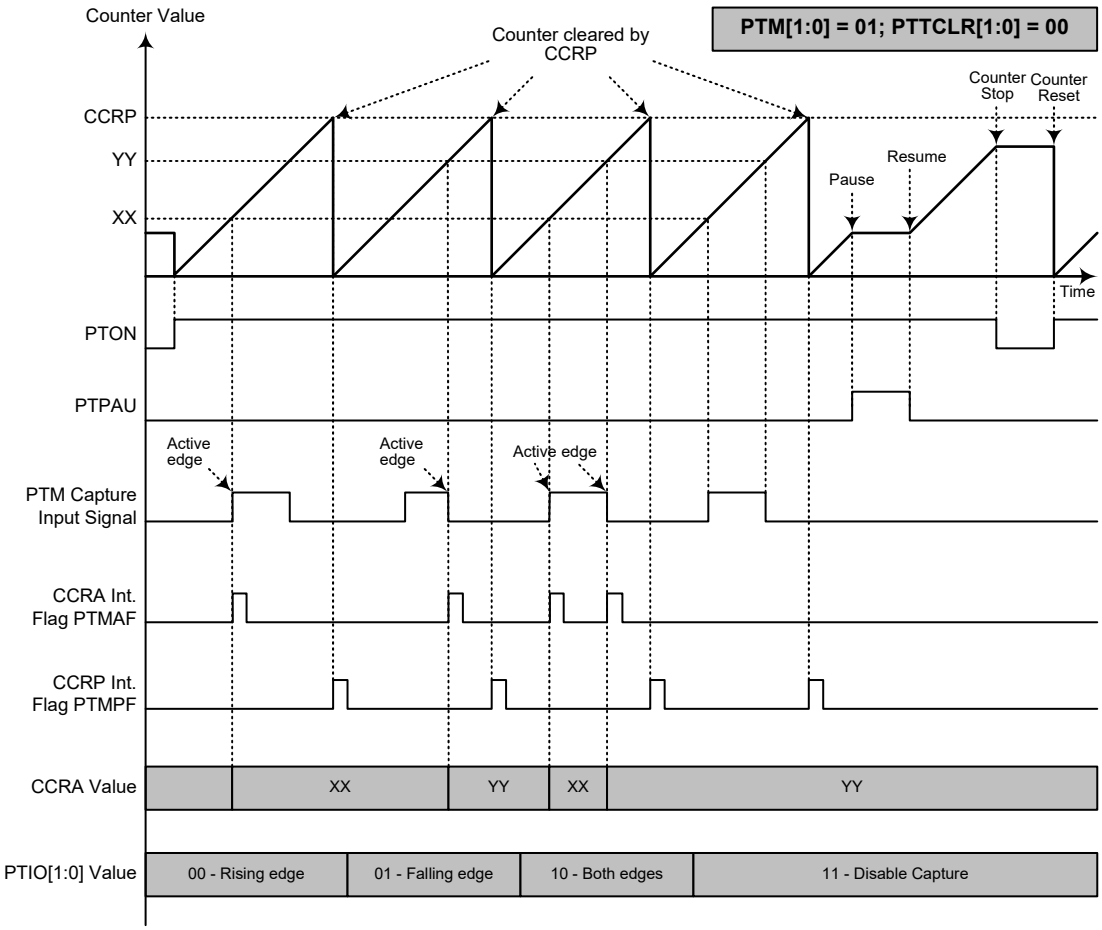
为使 PTM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“01”。此模式使能外部或内部信号捕捉并保存内部计数器当前值，因此被用于诸如脉冲宽度测量的应用中。通过设置 PTMC1 寄存器的 PTCAPTS 位选择捕捉输入信号来自 PTPI 信号还是 PTCK 引脚，其中 PTPI 信号可由 IFS0 寄存器的 IFS07~IFS06 位选择来自内部信号或外部引脚输入。可通过设置 PTMC1 寄存器的 PTIO1 和 PTIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 PTON 位由低变为高时，计数器启动。

PTIO1 和 PTIO0 位决定触发产生中断且被锁存的有效边沿。PTTCLR1 和 PTTCLR0 位决定计数器复位至零的条件。当前计数器值是锁存至 CCRA 还是 CCRB 取决于 PTIO1~PTIO0 位和 PTTCLR1~PTTCLR0 位的设置。PTIO1~PTIO0 位和 PTTCLR1~PTTCLR0 位的设置是相互独立且不相互影响的。

当选择的输入信号出现有效边沿转换时，计数器当前值被锁存到 CCRA 或 CCRB 寄存器，并产生 PTM 中断。无论选择的输入信号发生哪种边沿转换，计数器将继续工作直到 PTON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；通过这种方式 CCRP 的值可控制计数器的最大值。当比较器 P CCRP 比较匹配发生时，也会产生 PTM 中断。记录 CCRP 溢出中断信号的值可以测量长脉宽。通过设置 PTIO1 和 PTIO0 位选择输入信号为上升沿，下降沿或双沿有效。如果 PTIO1 和 PTIO0 位都设置为高，无论选择的输入信号发生哪种边沿转换都不会产生捕捉操作，但计数器仍会继续运行。

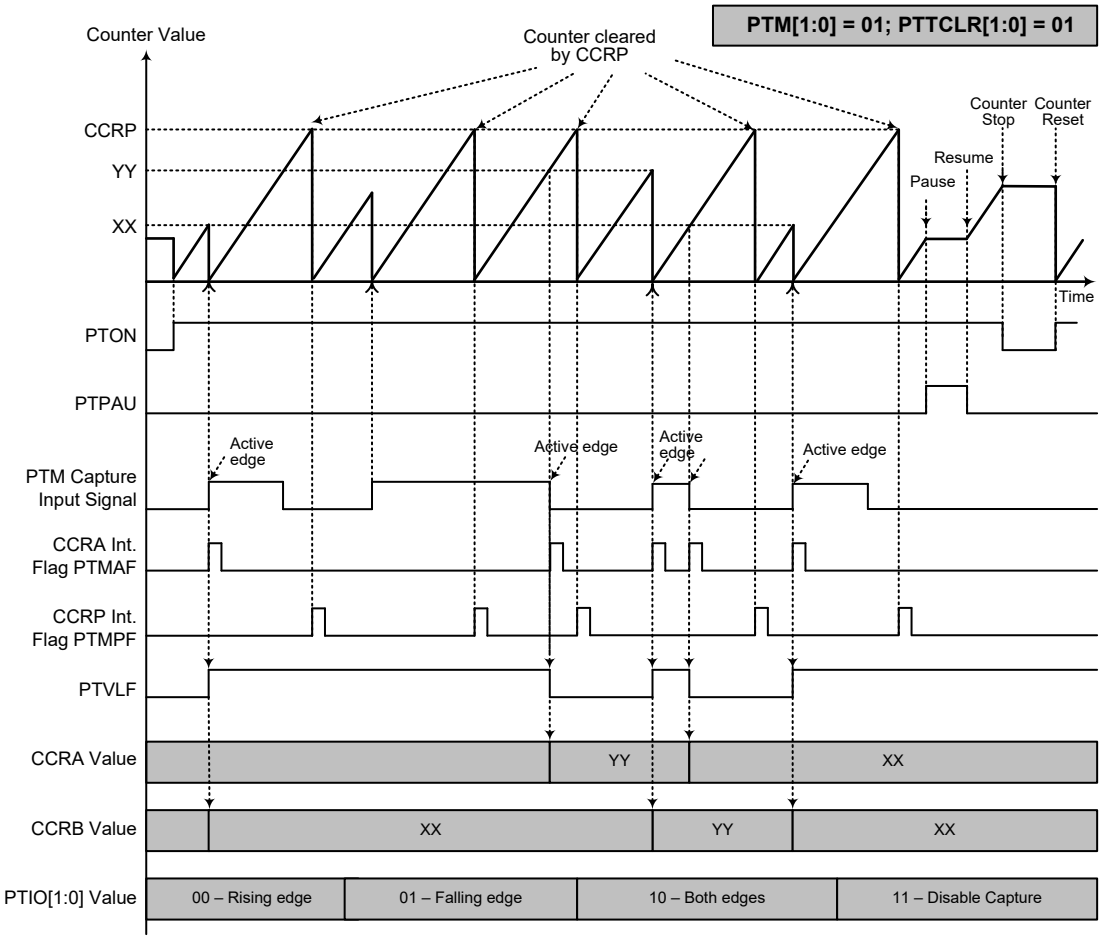
若捕捉脉宽小于两个 PTM 时钟周期，硬件可能会忽略该脉冲。PTM 时钟必须小于或等于 50MHz，否则计数器可能计数失败。

PTCCLR、PTOC 和 PTPOL 位在此模式中未使用。



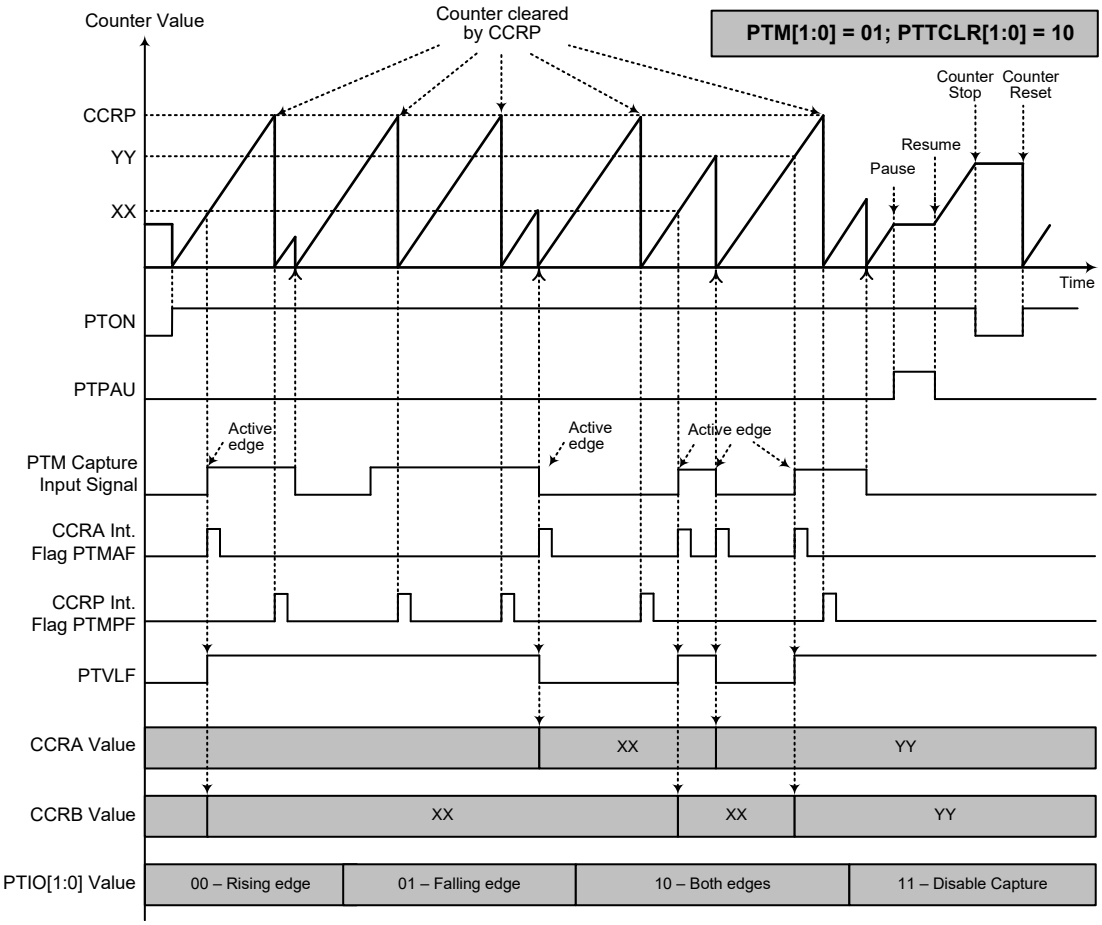
捕捉输入模式 – PTTCLR[1:0]=00

注：1. PTM[1:0]=01，PTTCLR[1:0]=00 并通过 PTIO[1:0] 位设置有效边沿
2. PTM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. 比较器 P 比较匹配，计数器清零。
4. PTCCLR 位未使用
5. 无输出功能 – PTOC 和 PTPOL 位未使用
6. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大
7. 当 PTTCLR[1:0]=00 时忽略 PTVLF 位的状态



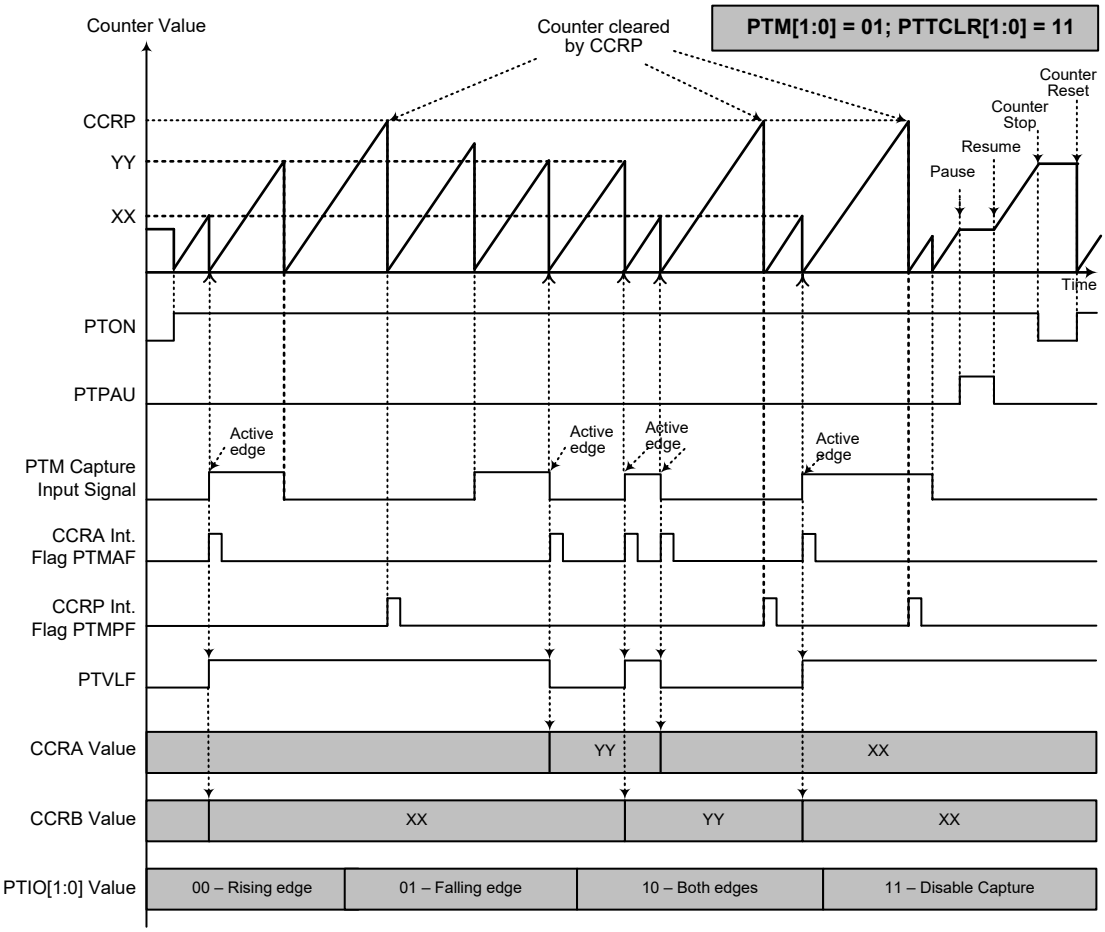
捕捉输入模式 – PTTCLR[1:0]=01

注：1. PTM[1:0]=01，PTTCLR[1:0]=01 并通过 PTIO[1:0] 位设置有效边沿
2. PTM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 或 CCRB 中
3. 比较器 P 比较匹配或 PTM 捕捉输入上升沿时，计数器清零。
4. PTCCLR 位未使用
5. 无输出功能 – PTOC 和 PTPOL 位未使用
6. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大



捕捉输入模式 – PTTCLR[1:0]=10

注：1. PTM[1:0]=01，PTTCLR[1:0]=10 并通过 PTIO[1:0] 位设置有效边沿
2. PTM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 或 CCRB 中
3. 比较器 P 比较匹配或 PTM 捕捉输入下降沿时，计数器清零。
4. PTCCLR 位未使用
5. 无输出功能 – PTOC 和 PTPOL 位未使用
6. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大

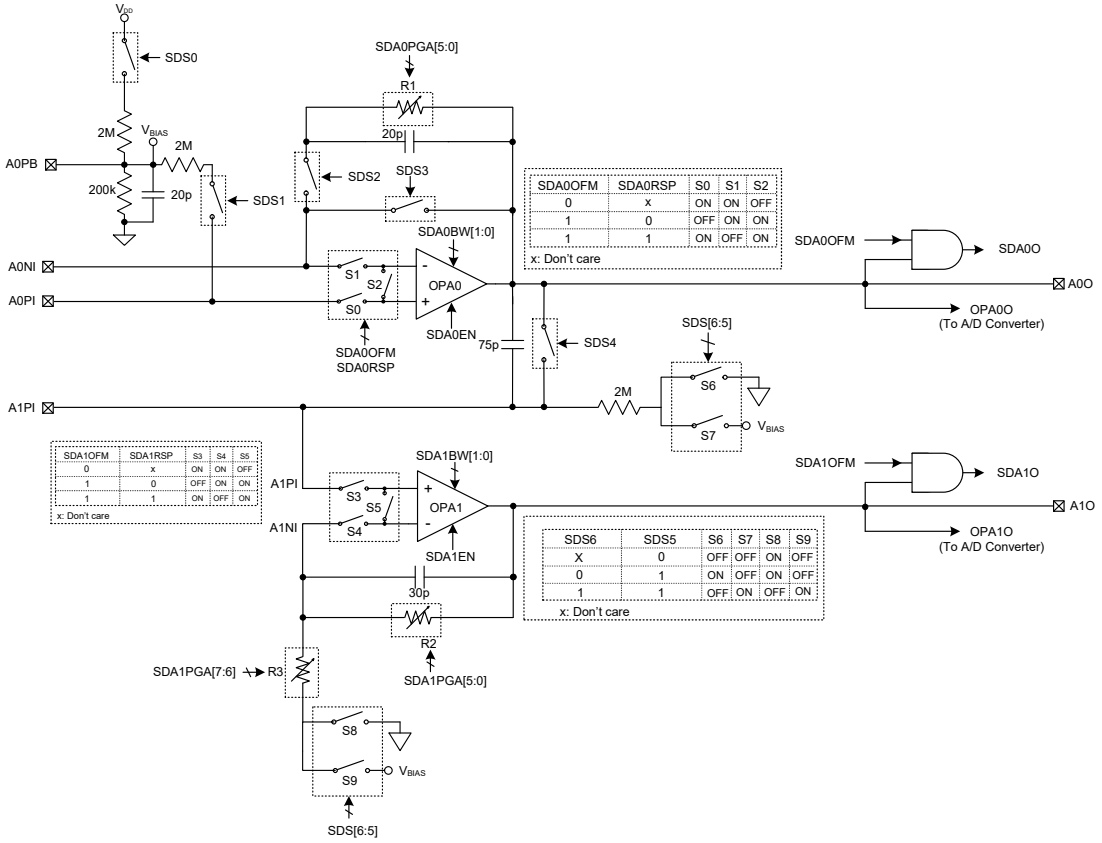


捕捉输入模式 – PTTCLR[1:0]=11

注：1. PTM[1:0]=01，PTTCLR[1:0]=11 并通过 PTIO[1:0] 位设置有效边沿
2. PTM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 或 CCRB 中
3. 比较器 P 比较匹配或 PTM 捕捉输入上升沿或下降沿时，计数器清零。
4. PTCCLR 位未使用
5. 无输出功能 – PTOC 和 PTPOL 位未使用
6. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大

感烟探测器 AFE

该单片机提供一个感烟探测器 AFE 电路，可用于感烟探测器应用的光信号检测。该电路包含两个完全集成的运算放大器。光信号可通过这两个运算放大器检测和处理。



SDA0OFM	SDA0RSP	S0	S1	S2
0	x	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

x: Don't care

SDA1OFM	SDA1RSP	S3	S4	S5
0	x	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

x: Don't care

SDS6	SDS5	S6	S7	S8	S9
X	0	OFF	OFF	ON	OFF
0	1	ON	OFF	ON	OFF
1	1	OFF	ON	OFF	ON

x: Don't care

感烟探测器 AFE 方框图

应注意，虽然 SD OPAm 带宽由 SDAmBW1~SDAmBW0 位决定，但 OPAm 搭配 A/D 转换器使用时则有一些限制。由于 OPAm 带宽导致输出电流较小，所以选择 SD OPAm 带宽就必须小心。使用者可以参考下面的表格，被标上 √ 的数值是可使用的。应确保 12-bit A/D 转换器读取到的数值已小于 1 LSB。

SD OPAm 带宽选择	A/D 转换器时钟频率 (kHz)							
	15.625	31.25	62.5	125	250	500	1000	2000
SDAmBW[1:0]=00	√	—	—	—	—	—	—	—
SDAmBW[1:0]=01	√	√	√	√	—	—	—	—
SDAmBW[1:0]=10	√	√	√	√	√	√	√	√
SDAmBW[1:0]=11	√	√	√	√	√	√	√	√

感烟探测器 AFE SD OPAm 带宽范例 (m=0~1)

感烟探测器 AFE 寄存器

感烟探测器 AFE 电路的所有操作由一系列寄存器控制。SDSW 寄存器用于控制开关 On/Off，进而控制运算放大器的工作模式。SDPGAC0 和 SDPGAC1 寄存器用于选择 R1、R2 和 R3 的阻值。SDAnC 寄存器用于控制 SD OPAn 使能 / 除能、带宽以及储存输出状态，其中 n=0~1。SDAnVOS 寄存器用于控制 SD OPAn 输入失调电压校准功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SDSW	—	SDS6	SDS5	SDS4	SDS3	SDS2	SDS1	SDS0
SDPGAC0	—	—	SDA0PGA5	SDA0PGA4	SDA0PGA3	SDA0PGA2	SDA0PGA1	SDA0PGA0
SDPGAC1	SDA1PGA7	SDA1PGA6	SDA1PGA5	SDA1PGA4	SDA1PGA3	SDA1PGA2	SDA1PGA1	SDA1PGA0
SDA0C	—	SDA0EN	SDA0O	—	—	—	SDA0BW1	SDA0BW0
SDA1C	—	SDA1EN	SDA1O	—	—	—	SDA1BW1	SDA1BW0
SDA0VOS	SDA0OFM	SDA0RSP	SDA0OF5	SDA0OF4	SDA0OF3	SDA0OF2	SDA0OF1	SDA0OF0
SDA1VOS	SDA1OFM	SDA1RSP	SDA1OF5	SDA1OF4	SDA1OF3	SDA1OF2	SDA1OF1	SDA1OF0

感烟探测器 AFE 寄存器列表

- **SDSW 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	SDS6	SDS5	SDS4	SDS3	SDS2	SDS1	SDS0
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 6~5　**SDS6~SDS5**：模式控制位
00：外部模式
01：AC 耦合模式
10：外部模式
11：DC 耦合模式 (SDS1 开关不能同时 On)

Bit 4　**SDS4**：SDS4 On/Off 控制位
0：Off
1：On

Bit 3　**SDS3**：SDS3 On/Off 控制位
0：Off
1：On

Bit 2　**SDS2**：SDS2 On/Off 控制位
0：Off
1：On

Bit 1　**SDS1**：SDS1 On/Off 控制位
0：Off
1：On

Bit 0　**SDS0**：SDS0 On/Off 控制位
0：Off
1：On

- **SDPGAC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SDA0PGA5	SDA0PGA4	SDA0PGA3	SDA0PGA2	SDA0PGA1	SDA0PGA0
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　**SDA0PGA5~SDA0PGA0**：R1 阻值控制位
R1=SDA0PGA[5:0]×100kΩ
这些位用于选择 R1 阻值，应注意当这些位设置为“000000”时 R1≠0Ω。

- **SDPGAC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SDA1PGA7	SDA1PGA6	SDA1PGA5	SDA1PGA4	SDA1PGA3	SDA1PGA2	SDA1PGA1	SDA1PGA0
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　**SDA1PGA7~SDA1PGA6**：R3 阻值控制位
00：10kΩ
01：20kΩ
10：30kΩ
11：40kΩ

Bit 5~0　**SDA1PGA5~SDA1PGA0**：R2 阻值控制位
R2=SDA1PGA[5:0]×100kΩ
这些位用于选择 R2 阻值，应注意当这些位设置为“000000”时 R2≠0Ω。

- **SDA0C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	SDA0EN	SDA0O	—	—	—	SDA0BW1	SDA0BW0
R/W	—	R/W	R	—	—	—	R/W	R/W
POR	—	0	0	—	—	—	0	0

Bit 7　未定义，读为“0”

Bit 6　**SDA0EN**：SD OPA0 使能 / 除能控制位
0：除能
1：使能

Bit 5　**SDA0O**：SD OPA0 输出状态 (正逻辑电平)
该位是只读位。
当 SDA0OFM=1，SDA0O 定义 SD OPA0 输出状态，详细内容参考“运算放大器输入失调校准”章节。
当 SDA0OFM=0，该位固定为低电平。

Bit 4~2　未定义，读为“0”

Bit 1~0　**SDA0BW1~SDA0BW0**：SD OPA0 带宽控制位
00：5kHz
01：40kHz
10：600kHz
11：2MHz
详细内容参考“运算放大器电气特性”。

- **SDA1C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	SDA1EN	SDA1O	—	—	—	SDA1BW1	SDA1BW0
R/W	—	R/W	R	—	—	—	R/W	R/W
POR	—	0	0	—	—	—	0	0

Bit 7　未定义，读为“0”

Bit 6　**SDA1EN**：SD OPA1 使能 / 除能控制位
0：除能
1：使能

Bit 5　**SDA1O**：SD OPA1 输出状态 (正逻辑电平)
该位是只读位。
当 SDA1OFM=1，SDA1O 定义 SD OPA1 输出状态，详细内容参考“运算放大器输入失调校准”章节。
当 SDA1OFM=0，该位固定为低电平。

Bit 4~2　未定义，读为“0”

Bit 1~0　**SDA1BW1~SDA1BW0**：SD OPA1 带宽控制位
00：5kHz
01：40kHz
10：600kHz
11：2MHz
详细内容参考“运算放大器电气特性”。

- **SDA0VOS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SDA0OFM	SDA0RSP	SDA0OF5	SDA0OF4	SDA0OF3	SDA0OF2	SDA0OF1	SDA0OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7　**SDA0OFM**：SD OPA0 正常操作或输入失调电压校准模式选择位
0：正常操作
1：失调校准模式

Bit 6　**SDA0RSP**：SD OPA0 输入失调电压校准参考选择位
0：输入参考电压来自 A0NI
1：输入参考电压来自 A0PI

Bit 5~0　**SDA0OF5~SDA0OF0**：SD OPA0 输入失调电压校准控制位
这 6 位用于执行运算放大器的输入失调校准操作，并重新储存 SD OPA0 的输入失调校准值。更多详细资料请参考“运算放大器输入失调校准”章节。

- **SDA1VOS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SDA1OFM	SDA1RSP	SDA1OF5	SDA1OF4	SDA1OF3	SDA1OF2	SDA1OF1	SDA1OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7　**SDA1OFM**：SD OPA1 正常操作或输入失调电压校准模式选择位
0：正常操作
1：失调校准模式

Bit 6　**SDA1RSP**：SD OPA1 输入失调电压校准参考选择位
0：输入参考电压来自 A1NI
1：输入参考电压来自 A1PI

Bit 5~0 **SDA1OF5~SDA1OF0**：SD OPA1 输入失调电压校准控制位
这 6 位用于执行运算放大器的输入失调校准操作，并重新储存 SD OPA1 的输入失调校准值。更多详细资料请参考“运算放大器输入失调校准”章节。

运算放大器操作

单片机内部集成两个运算放大器，OPA0 和 OPA1，可根据用户的特定需求对信号进行放大。它们的使能或除能通过软件设置内部寄存器来完成。通过控制特殊寄存器，OPA 相关的应用可以很容易的实现，例如单位增益缓冲器、同相放大器、反相放大器和各种各样的滤波器等。

运算放大器输入失调校准

应注意，由于 SD 运算放大器输入引脚与 I/O 引脚共用，在输入失调校准前应先将引脚配置为 SD 运算放大器输入引脚。

步骤 1：设置 SDAnOFM=1 和 SDAnRSP=1，使 SD 运算放大器 n 工作于失调校准模式，S0 和 S2 都 On。为了确保校准后的 V_{AnOS} 尽可能小，校准模式下的输入参考电压应该跟正常模式下的输入直流工作电压相同。

步骤 2：设置 SDAnOF[5:0]=000000，读取 SDAnO 位。

步骤 3：使 SDAnOF[5:0]=SDAnOF[5:0]+1，读取 SDAnO 位。

如果 SDAnO 位状态不变，重复步骤 3 直到 SDAnO 位状态改变。

如果 SDAnO 位状态改变，记录此时的 SDAnOF[5:0] 值为 V_{AnOS1} 然后转到步骤 4。

步骤 4：设置 SDAnOF[5:0]=111111，读取 SDAnO 位。

步骤 5：使 SDAnOF[5:0]=SDAnOF[5:0]-1，读取 SDAnO 位。

如果 SDAnO 位状态不变，重复步骤 5 直到 SDAnO 位状态改变。

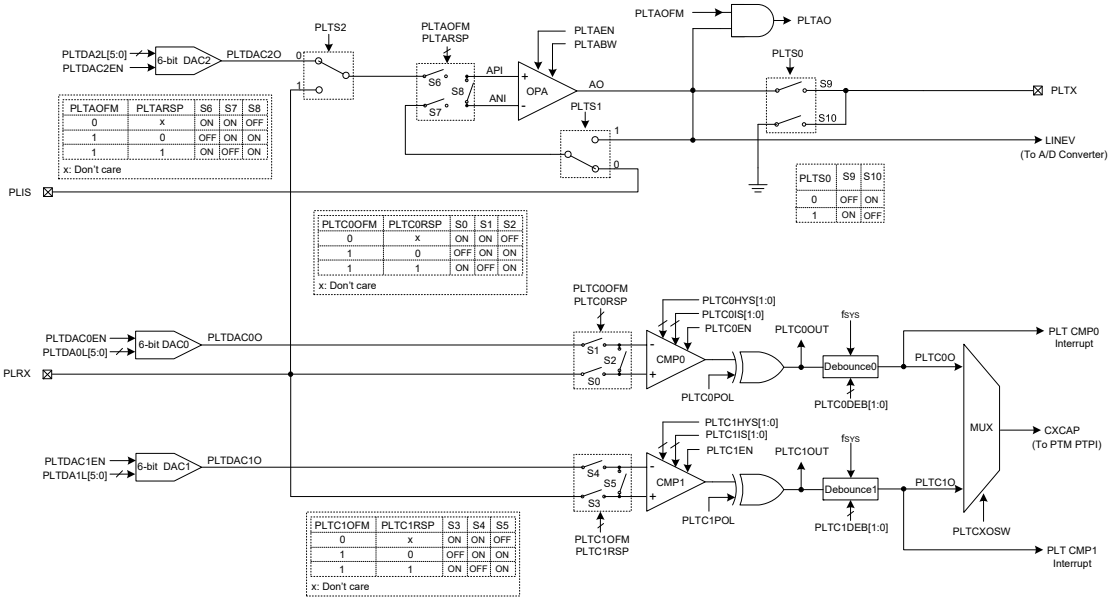
如果 SDAnO 位状态改变，记录此时的 SDAnOF[5:0] 值为 V_{AnOS2} 然后转到步骤 6。

步骤 6：将 SD 运算放大器 n 输入失调校准值 V_{AnOS} 存入 SDAnOF[5:0] 位中，校准结束。

其中 $V_{AnOS}=(V_{AnOS1}+V_{AnOS2})/2$。如果 $(V_{AnOS1}+V_{AnOS2})/2$ 不是整数，舍弃小数。

电源线数据收发器 – PLT

该单片机包含一个电源线数据收发器电路，可用于电源线数据发送和接收。该电路由三个 6-bit D/A 转换器、一个完全集成的运算放大器和两个比较器组成。



电源线数据收发器方框图

电源线数据收发器寄存器

电源线数据收发器电路的所有操作由一系列寄存器控制。通过应用程序设置这些寄存器可以控制 DACn 输出，运算放大器、比较器输入信号的选择，工作模式以及输出信号等。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PLTSW	—	—	—	—	—	PLTS2	PLTS1	PLTS0
PLTDACC	—	—	—	—	—	PLTDAC2EN	PLTDAC1EN	PLTDAC0EN
PLTDA0L	—	—	D5	D4	D3	D2	D1	D0
PLTDA1L	—	—	D5	D4	D3	D2	D1	D0
PLTDA2L	—	—	D5	D4	D3	D2	D1	D0
PLTC0C	PLTC0OUT	PLTC0EN	PLTC0O	—	PLTC0DEB1	PLTC0DEB0	PLTC0IS1	PLTC0IS0
PLTC0VOS	—	PLTC0OFM	PLTC0RSP	PLTC0OF4	PLTC0OF3	PLTC0OF2	PLTC0OF1	PLTC0OF0
PLTC1C	PLTC1OUT	PLTC1EN	PLTC1O	—	PLTC1DEB1	PLTC1DEB0	PLTC1IS1	PLTC1IS0
PLTC1VOS	—	PLTC1OFM	PLTC1RSP	PLTC1OF4	PLTC1OF3	PLTC1OF2	PLTC1OF1	PLTC1OF0
PLTCHYC	—	PLTCXOSW	PLTC1POL	PLTC0POL	PLTC1HYS1	PLTC1HYS0	PLTC0HYS1	PLTC0HYS0
PLTAC	—	PLTAEN	PLTAO	—	—	—	—	PLTABW
PLTAVOS	PLTAOFM	PLTARSP	PLTAOF5	PLTAOF4	PLTAOF3	PLTAOF2	PLTAOF1	PLTAOF0

电源线数据收发器寄存器列表

- **PLTSW 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PLTS2	PLTS1	PLTS0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	1

Bit 7~3　未定义，读为“0”

Bit 2　**PLTS2**：PLTS2 开关选择位
0：连接到 PLTDAC2O
1：连接到 PLRX

Bit 1　**PLTS1**：PLTS1 开关选择位
0：连接到 PLIS
1：连接到 LINEV

Bit 0　**PLTS0**：PLTX 开关选择位
0：连接到地
1：连接到 AO

- **PLTDACC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PLTDAC2EN	PLTDAC1EN	PLTDAC0EN
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3　未定义，读为“0”

Bit 2　**PLTDAC2EN**：PLT DAC2 使能 / 除能控制位
0：除能 (PLTDAC2O 处于高阻抗状态)
1：使能

Bit 1　**PLTDAC1EN**：PLT DAC1 使能 / 除能控制位
0：除能 (PLTDAC1O 处于高阻抗状态)
1：使能

Bit 0　**PLTDAC0EN**：PLT DAC0 使能 / 除能控制位
0：除能 (PLTDAC0O 处于高阻抗状态)
1：使能

- **PLTDA0L 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	D5	D4	D3	D2	D1	D0
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　**D5~D0**：PLT DAC0 输出控制码
PLTDAC0O=(DAC $AV_{DD}/2^6$)×PLTDA0L[5:0]

- **PLTDA1L 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	D5	D4	D3	D2	D1	D0
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　**D5~D0**：PLT DAC1 输出控制码
PLTDAC1O=(DAC $AV_{DD}/2^6$)×PLTDA1L[5:0]

- **PLTDA2L 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	D5	D4	D3	D2	D1	D0
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　**D5~D0**：PLT DAC2 输出控制码
PLTDAC2O=(DAC $V_{DD}/2^6$)×PLTDA2L[5:0]

- **PLTC0C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PLTC0OUT	PLTC0EN	PLTC0O	—	PLTC0DEB1	PLTC0DEB0	PLTC0IS1	PLTC0IS0
R/W	R	R/W	R	—	R/W	R/W	R/W	R/W
POR	0	0	0	—	0	0	0	0

Bit 7　**PLTC0OUT**：PLT 比较器 0 输出位
若 PLTC0POL=0 且比较器输入电压为
C0PI>C0NI → PLTC0OUT=1
C0NI>C0PI → PLTC0OUT=0
若 PLTC0POL=1 且比较器输入电压为
C0PI<C0NI → PLTC0OUT=1
C0NI<C0PI → PLTC0OUT=0

Bit 6　**PLTC0EN**：PLT 比较器 0 使能 / 除能控制
0：除能
1：使能
此位为 PLT 比较器 0 开关控制位。若此位除能，比较器输出为 0。因此当 PLTC0POL=0 时 PLTC0OUT 为 0，当 PLTC0POL=1 时 PLTC0OUT 为 1。

Bit 5　**PLTC0O**：PLT 比较器 0 去抖输出
PLTC0O 为 PLTC0OUT 去抖后的输出。
若 PLTC0POL=0，仅当 PLTC0OUT 当前及之前的 N 个采样都为“1”时 PLTC0O 才输出“1”。若 PLTC0POL=1，仅当 PLTC0OUT 当前及之前的 N 个采样都为“0”时 PLTC0O 才输出“0”。采样频率取决于 PLTC0DEB[1:0] 位的配置。

Bit 4　未定义，读为“0”

Bit 3~2　**PLTC0DEB1~PLTC0DEB0**：PLT 比较器 0 去抖时间控制
00：无去抖
01：(31~32)×t_{SYS}
10：(63~64)×t_{SYS}
11：(126~127)×t_{SYS}
注：$t_{SYS}=1/f_{SYS}$。

Bit 1~0　**PLTC0IS1~PLTC0IS0**：PLT 比较器 0 电流控制
详细内容参考“比较器电气特性”表格。

- **PLTC1C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PLTC1OUT	PLTC1EN	PLTC1O	—	PLTC1DEB1	PLTC1DEB0	PLTC1IS1	PLTC1IS0
R/W	R	R/W	R	—	R/W	R/W	R/W	R/W
POR	0	0	0	—	0	0	0	0

Bit 7 **PLTC1OUT**：PLT 比较器 1 输出位
若 PLTC1POL=0 且比较器输入电压为
C1PI>C1NI → PLTC1OUT=1
C1NI>C1PI → PLTC1OUT=0
若 PLTC1POL=1 且比较器输入电压为
C1PI<C1NI → PLTC1OUT=1
C1NI<C1PI → PLTC1OUT=0

Bit 6 **PLTC1EN**：PLT 比较器 1 使能 / 除能控制
0：除能
1：使能
此位为 PLT 比较器 1 开关控制位。若此位除能，比较器输出为 0。因此当 PLTC1POL=0 时 PLTC1OUT 为 0，当 PLTC1POL=1 时 PLTC1OUT 为 1。

Bit 5 **PLTC1O**：PLT 比较器 1 去抖输出
PLTC1O 为 PLTC1OUT 去抖后的输出。
若 PLTC1POL=0，仅当 PLTC1OUT 当前及之前的 N 个采样都为“1”时 PLTC1O 才输出“1”。若 PLTC1POL=1，仅当 PLTC1OUT 当前及之前的 N 个采样都为“0”时 PLTC1O 才输出“0”。采样频率取决于 PLTC1DEB[1:0] 位的配置。

Bit 4 未定义，读为“0”

Bit 3~2 **PLTC1DEB1~PLTC1DEB0**：PLT 比较器 1 去抖时间控制
00：无去抖
01：(31~32)×t_{SYS}
10：(63~64)×t_{SYS}
11：(126~127)×t_{SYS}
注：$t_{SYS}=1/f_{SYS}$。

Bit 1~0 **PLTC1IS1~PLTC1IS0**：PLT 比较器 1 电流控制
详细内容参考“比较器电气特性”表格。

- **PLTC0VOS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PLTC0OFM	PLTC0RSP	PLTC0OF4	PLTC0OF3	PLTC0OF2	PLTC0OF1	PLTC0OF0
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 未定义，读为“0”

Bit 6 **PLTC0OFM**：PLT 比较器 0 正常操作或输入失调电压校准模式选择位
0：正常操作
1：失调校准模式

Bit 5 **PLTC0RSP**：PLT 比较器 0 输入失调电压校准参考选择位
0：输入参考电压来自 C0NI
1：输入参考电压来自 C0PI

Bit 4~0 **PLTC0OF4~PLTC0OF0**：PLT 比较器 0 输入失调电压校准控制位
这 5 位用于执行 PLT 比较器 0 的输入失调校准操作，并重新储存 PLT 比较器 0 的输入失调校准值。更多详细资料请参考“比较器输入失调校准”章节。

- **PLTC1VOS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PLTC1OFM	PLTC1RSP	PLTC1OF4	PLTC1OF3	PLTC1OF2	PLTC1OF1	PLTC1OF0
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　未定义，读为“0”

Bit 6　**PLTC1OFM**：PLT 比较器 1 正常操作或输入失调电压校准模式选择位
0：正常操作
1：失调校准模式

Bit 5　**PLTC1RSP**：PLT 比较器 1 输入失调电压校准参考选择位
0：输入参考电压来自 C1NI
1：输入参考电压来自 C1PI

Bit 4~0　**PLTC1OF4~PLTC1OF0**：PLT 比较器 1 输入失调电压校准控制位
这 5 位用于执行 PLT 比较器 1 的输入失调校准操作，并重新储存 PLT 比较器 1 的输入失调校准值。更多详细资料请参考“比较器输入失调校准”章节。

- **PLTCHYC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PLTCXOSW	PLTC1POL	PLTC0POL	PLTC1HYS1	PLTC1HYS0	PLTC0HYS1	PLTC0HYS0
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　**PLTCXOSW**：比较器 0 或比较器 1 输出选择位
0：比较器 0 输出
1：比较器 1 输出
该位为比较器 0 或比较器 1 输出控制位。若该位为 0 则 PLTC0O 将会输出；若该位为 1 则 PLTC1O 位将会输出。

Bit 5　**PLTC1POL**：PLT 比较器 1 输出极性控制位
0：同相
1：反相
该位为 PLT 比较器 1 输出极性控制位。若该位为 0 则 PLTC1OUT 位为比较器 1 的同相输出；若该位为 1 则 PLTC1OUT 为比较器 1 的反相输出。

Bit 4　**PLTC0POL**：PLT 比较器 0 输出极性控制位
0：同相
1：反相
该位为 PLT 比较器 0 输出极性控制位。若该位为 0 则 PLTC0OUT 位为比较器 0 的同相输出；若该位为 1 则 PLTC0OUT 为比较器 0 的反相输出。

Bit 3~2　**PLTC1HYS1~PLTC1HYS0**：PLT 比较器 1 迟滞电压窗口控制位
详细内容参考“比较器电气特性”表格。

Bit 1~0　**PLTC0HYS1~PLTC0HYS0**：PLT 比较器 0 迟滞电压窗口控制位
详细内容参考“比较器电气特性”表格。

- **PLTAC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PLTAEN	PLTAO	—	—	—	—	PLTABW
R/W	—	R/W	R	—	—	—	—	R/W
POR	—	0	0	—	—	—	—	0

Bit 7　未定义，读为“0”

Bit 6　**PLTAEN**：PLT OPA 使能 / 除能控制位
0：除能 (AO 处于高阻抗状态)
1：使能

Bit 5　**PLTAO**：PLT OPA 输出状态 (正逻辑电平)
该位是只读位。
当 PLTAOFM 为 1，PLTAO 定义 PLT OPA 输出状态，详细内容参考“失调校准步骤”章节。当 PLTAOFM 为 0，该位固定为低电平。

Bit 4~1　未定义，读为“0”

Bit 0　**PLTABW**：PLT OPA 增益带宽控制位
0：600kHz
1：2MHz
详细内容参考“运算放大器电气特性”表格。

- **PLTAVOS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PLTAOFM	PLTARSP	PLTAOF5	PLTAOF4	PLTAOF3	PLTAOF2	PLTAOF1	PLTAOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7　**PLTAOFM**：PLT OPA 正常操作或输入失调电压校准模式选择位
0：正常操作
1：失调校准模式

Bit 6　**PLTARSP**：PLT OPA 输入失调电压校准参考选择位
0：输入参考电压来自 ANI
1：输入参考电压来自 API

Bit 5~0　**PLTAOF5~PLTAOF0**：PLT OPA 输入失调电压校准控制位
这 6 位用于执行 PLT 运算放大器的输入失调校准操作，并重新储存 PLT OPA 的输入失调校准值。更多详细资料请参考“运算放大器输入失调校准”章节。

失调校准步骤

PLTAOFM 或 PLTCnOFM 位先设置为“1”使 PLT 运算放大器或 PLT 比较器 n 处于输入失调校准模式。应注意，由于 PLT 运算放大器输入引脚 PLIS 或 PLT 比较器 n 输入引脚 PLRX 与 I/O 引脚共用，应先将引脚配置为 PLT 运算放大器或 PLT 比较器 n 输入引脚。

比较器输入失调校准

- 步骤 1
设置 PLTCnOFM=1 和 PLTCnRSP=1，使 PLT 比较器 n 工作于失调校准模式，开关 S0 和 S2 都 ON 或 S3 和 S5 都 ON。为了确保校准后的 V_{CnOS} 尽可能小，校准模式下的输入参考电压应该跟正常模式下的输入直流工作电压相同。
- 步骤 2
设置 PLTCnOF[4:0]=00000，读取 PLTCnOUT 位。

- 步骤 3
  使 PLTCnOF[4:0]=PLTCnOF[4:0]+1，读取 PLTCnOUT 位。
  如果 PLTCnOUT 位状态不变，重复步骤 3 直到 PLTCnOUT 位状态改变。
  如果 PLTCnOUT 位状态改变，记录此时的 PLTCnOF[4:0] 值为 V_{CnOS1} 然后转到步骤 4。
- 步骤 4
  设置 PLTCnOF[4:0]=11111，读取 PLTCnOUT 位。
- 步骤 5
  使 PLTCnOF[4:0]=PLTCnOF[4:0]-1，读取 PLTCnOUT 位。
  如果 PLTCnOUT 位状态不变，重复步骤 5 直到 PLTCnOUT 位状态改变。
  如果 PLTCnOUT 位状态改变，记录此时的 PLTCnOF[4:0] 值为 V_{CnOS2} 然后转到步骤 6。
- 步骤 6
  将 PLT 比较器 n 输入失调校准值 V_{CnOS} 存入 PLTCnOF[4:0] 位中，校准结束。
  其中 $V_{CnOS}=(V_{CnOS1}+V_{CnOS2})/2$。如果 $(V_{CnOS1}+V_{CnOS2})/2$ 不是整数，舍弃小数。

运算放大器输入失调校准

- 步骤 1
  设置 PLTAOFM=1 且 PLTARSP=1，使 PLT 运算放大器工作于失调校准模式，开关 S6 和 S8 都 ON。为了确保校准后的 V_{AOS} 尽可能小，校准模式下的输入参考电压应该跟正常模式下的输入直流工作电压相同。
- 步骤 2
  设置 PLTAOF[5:0]=000000，读取 PLTAO 位。
- 步骤 3
  使 PLTAOF[5:0]=PLTAOF[5:0]+1，读取 PLTAO 位。
  如果 PLTAO 位状态不变，重复步骤 3 直到 PLTAO 位状态改变。
  如果 PLTAO 位状态改变，记录此时的 PLTAOF[5:0] 值为 V_{AOS1} 然后转到步骤 4。
- 步骤 4
  设置 PLTAOF[5:0]=111111，读取 PLTAO 位。
- 步骤 5
  使 PLTAOF[5:0]=PLTAOF[5:0]-1，读取 PLTAO 位。
  如果 PLTAO 位状态不变，重复步骤 5 直到 PLTAO 位状态改变。
  如果 PLTAO 位状态改变，记录此时的 PLTAOF[5:0] 值为 V_{AOS2} 然后转到步骤 6。
- 步骤 6
  将 PLT 运算放大器输入失调校准值 V_{AOS} 存入 PLTAOF[5:0] 位中，校准结束。
  其中 $V_{AOS}=(V_{AOS1}+V_{AOS2})/2$。如果 $(V_{AOS1}+V_{AOS2})/2$ 不是整数，舍弃小数。

A/D 转换器

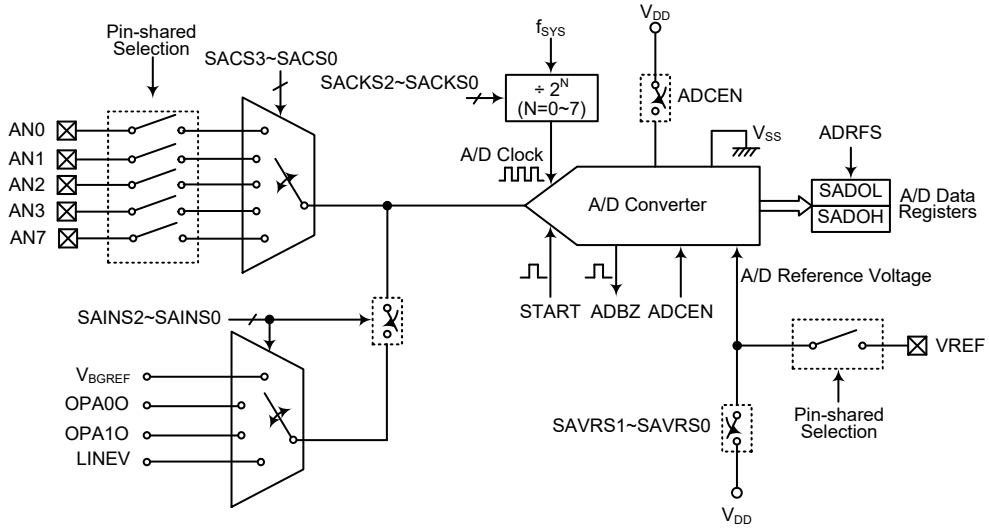
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 简介

此单片机包含一个多通道 12-bit A/D 转换器，它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 (高性能 Bandgap 参考电压 V_{BGREF}、SD 运算放大器 0 输出信号 OPA0O、SD 运算放大器 1 输出信号 OPA1O 和 PLT 运算放大器输出信号 LINEV) 并直接将这些信号转换成 12-bit 的数字量。选择转换外部或内部模拟信号由 SAINS2~SAINS0 位和 SACS3~SACS0 位共同控制。关于 A/D 输入信号的详细描述请参考“A/D 转换器控制寄存器”和“A/D 转换器输入信号”两节内容。

外部输入通道	内部信号	通道选择位
5：AN0~AN3, AN7	4：V_{BGREF}, OPA0O, OPA1O, LINEV	SAINS2~SAINS0, SACS3~SACS0

图显示了 A/D 转换器整体的内部结构和相关的寄存器。



A/D 转换器结构

A/D 转换寄存器介绍

A/D 转换器的所有工作由一系列寄存器控制。一对只读寄存器来存放 12-bit A/D 转换数据的值。剩下两个控制寄存器设置 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
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0

寄存器名称	位							
	7	6	5	4	3	2	1	0
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D 转换寄存器列表

A/D 转换器数据寄存器 – SADOL, SADOH

对于具有 12 位 A/D 转换器的芯片，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”。应注意，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 数据寄存器

A/D 转换器控制寄存器 – SADC0, SADC1

寄存器 SADC0 和 SADC1 用来控制 A/D 转换器的功能和操作。这些 8-bit 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，数字化数据格式，A/D 时钟源，并控制和监视 A/D 转换器的忙碌状态。由于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中的每一个都需要分别被发送到转换器。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 SAINS2~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。

引脚共用功能选择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 I/O 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
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
此位只读标志位用于表明 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 转换已开始。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 的内容将保持不变。

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 外部模拟通道输入选择位
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：保留
0101：保留
0110：保留
0111：AN7
1000~1111：未定义，输入浮空

- **SADC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
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 7~5 **SAINS2~SAINS0**：A/D 输入信号选择位
000：外部输入 – 外部模拟通道输入 ANn
001：内部输入 – 内部高性能 Bandgap 参考电压 V_{BGREF}
010：内部输入 – 内部 SD 运算放大器 0 输出信号 OPA0O
011：内部输入 – 内部 SD 运算放大器 1 输出信号 OPA1O
100：内部输入 – 内部 PLT 运算放大器输出信号 LINEV
101~111：外部输入 – 外部模拟通道输入 ANn
当 SAINS2~SAINS0 被设为“001”~“100”选择转换内部模拟信号时，需特别注意。当选择内部模拟信号时，应正确设置 SACS3~SACS0 位，避免外部通道输入作为 A/D 输入信号。否则，外部输入通道会和内部模拟信号一起连接至内部 A/D 转换器，这将导致无法预期的损害。

Bit 4~3 **SAVRS1~SAVRS0**：A/D 转换器参考电压选择位
00：外部 VREF 引脚
01：内部 A/D 转换器电源 V_{DD}
1x：外部 VREF 引脚
这两位用于选择 A/D 转换器参考电压。当 SAVRS1~SAVRS0 为被设为“01”选择内部参考电压时需特别注意。当选择 A/D 转换器电源作为 A/D 转换器参考电压时，外部 VREF 引脚需通过引脚共用控制位选择作为其它共用的引脚功能。否则，外部 VREF 输入电压会和内部参考电压一起连接至内部 A/D 转换器，可能会导致内部电路的损坏。

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：$f_{SYS}/128$
这三位用于选择 A/D 转换器的时钟源。

A/D 转换器操作

SADC0 寄存器中的 START 位，用于开启 A/D 转换器。当单片机设置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。

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 位会自动置为“0”。此外，也会置位中断控制寄存器内相应的 A/D 中断请求标志位，如果中断使能，就会产生对应的内部中断信号。A/D 内部中断信号将引导程序跳转到相应的 A/D 内部中断地址。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，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 *

A/D 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。

A/D 转换器参考电压

A/D 转换器参考电压来自正电源电压 V_{DD} 或外部参考源引脚 VREF，通过 SAVRS1 和 SAVRS0 位选择。当 SAVRS1~SAVRS0 位为“01”时，A/D 转换器参考电压来自 V_{DD}。当 SAVRS1~SAVRS0 位为“01”以外的任意值时，A/D 转换器参考电压来自 VREF 引脚。由于 VREF 引脚与其它功能共用，当选择 VREF 引脚作为参考电压源时，需先正确设置引脚共用选择位将 VREF 引脚配置为参考电压输入功能。然而，当内部 A/D 转换器电源被选作参考电压源时，相关的引脚共用控制位不可选择 VREF 参考电压输入功能，避免 VREF 引脚电压跟内部参考电压 V_{DD} 一起接入 A/D 转换器。模拟输入值一定不能超过所选的参考电压值。

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。

另外还有几个内部模拟信号可作为 A/D 转换器的模拟输入信号，分别来自高性能 Bandgap 参考电压 V_{BGREF}、SD 运算放大器 0 输出信号 OPA0O、SD 运算放大器 1 输出信号 OPA1O 和 PLT 运算放大器输出信号 LINEV，通过设置 SAINS2~SAINS0 位来选择。若 SAINS2~SAINS0 位为“000”或“101~111”，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，应将 SACS3~SACS0 位设置为一个适当的值，以关闭外部模拟通道输入。否则，外部通道输入会与内部模拟信号一起接入从而导致错误。

V_{BGREF} 是具有驱动能力的高性能 Bandgap 的参考电压。当输入电源电压 V_{DD} 改变或温度变化时，Bandgap 会输出精准的参考电压。而且该 Bandgap 在低压下便可启动。因此，该参考电压对于存在低压降线性稳压器 LDO 的应用具有较高的电源电压抑制比 PSRR。

SAINS[2:0]	SACS[3:0]	输入信号	说明
000, 101~111	0000~0011, 0111	AN0~AN3, AN7	外部模拟通道输入
	1000~1111	—	未选择外部通道，输入浮空
001	1000~1111	V_{BGREF}	内部高性能 Bandgap 参考电压
010	1000~1111	OPA0O	内部 SD 运算放大器 0 输出信号
011	1000~1111	OPA1O	内部 SD 运算放大器 1 输出信号
100	1000~1111	LINEV	内部 PLT 运算放大器输出信号

A/D 转换器输入信号选择

- **VBGRC 寄存器**

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

Bit 7~1　未定义，读为“0”

Bit 0　**VBGREN**：Bandgap 使能 / 除能控制位
0：除能
1：使能
当 VBGREN 位被清零时，Bandgap 输出处于高阻抗状态。

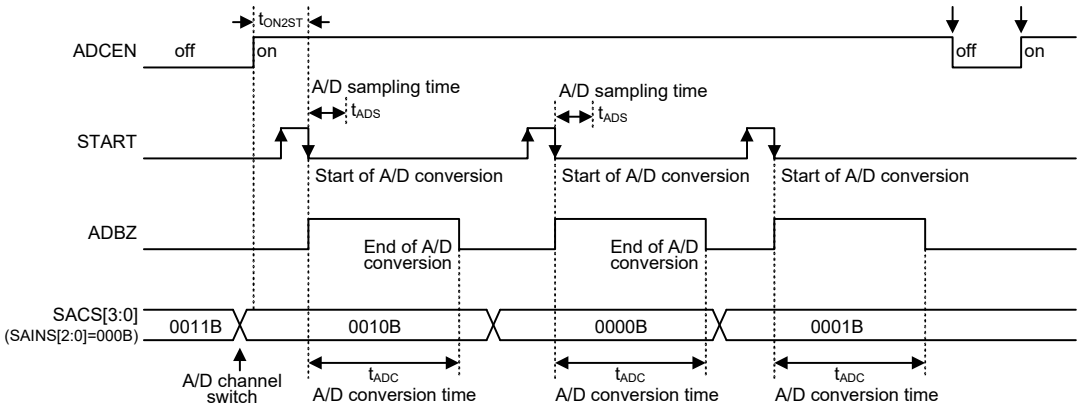
A/D 转换率及时序图

一个完整的 A/D 转换包含两部分，数据采样和数据转换。数据采样时间定义为 t_{ADS}，需要 4 个 A/D 时钟周期，而数据转换需要 12 个 A/D 时钟周期。所以一个完整的 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 转换步骤概述

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 SAINS2~SAINS0 位，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
  若已通过 SAINS2~SAINS0 位选择 A/D 输入信号来自外部通道输入，接着应设置相关的引脚共用控制位将该引脚规划为 A/D 输入引脚。通过设置 SACS3~SACS0 位选择哪个外部通道接至 A/D 转换器。接着执行步骤 6。
- 步骤 5
  选择内部模拟信号前，应正确设置 SACS3~SACS0 位，将外部通道输入切换到无通道输入。然后再设置 SAINS2~SAINS0 位选择所需的内部模拟信号。接着执行步骤 6。
- 步骤 6
  通过 SADC1 寄存器中的 SAVRS1~SAVRS0 位选择参考电压。
- 步骤 7
  设置 SADC0 寄存器中的 ADRFS 位选择 A/D 转换器输出数据格式。
- 步骤 8
  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 中断功能是激活的。总中断控制位 EMI 需要置位为“1”，以及 A/D 转换器中断位 ADE 也需要置位为“1”。
- 步骤 9
  现在可以通过设置 SADC0 寄存器中的 START 位从“0”到“1”再回到“0”，开始模数转换的过程。

- 步骤 10

  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，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

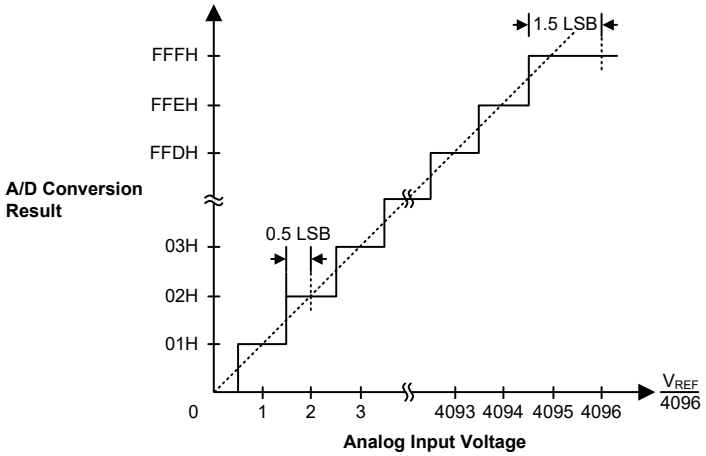
单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于实际 A/D 转换器参考电压值 V_{REF}，因此每一位可表示 V_{REF}/4096 的模拟输入值。

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

通过下面的等式可估算 A/D 转换器输入电压值：

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

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 处改变。注意，这里的 V_{REF} 电压指的是通过 SAVRS[1:0] 位选择的实际输入 A/D 转换器的参考电压。



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：使用查询 ADBZ 的方式来检测转换结束

```
clr  ADE                    ; disable ADC interrupt
mov  a,03H
mov  SADC1,a                ; select fSYS/8 as A/D clock
set  ADCEN
```

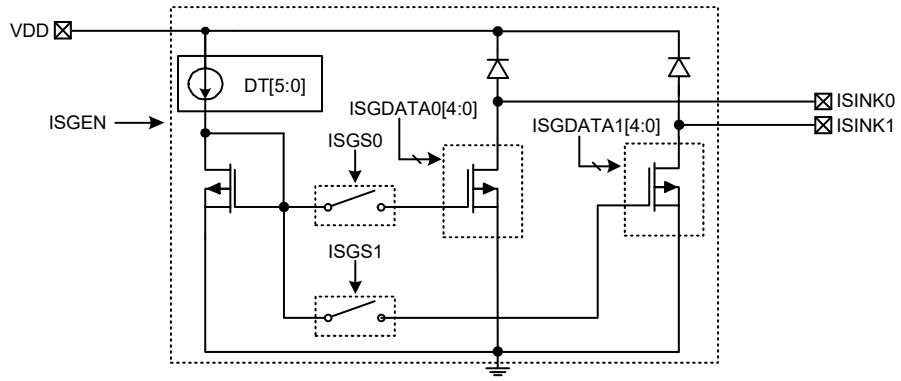
```
mov a,02h                  ; setup PAS1 register to configure pin AN0
mov PAS1,a
mov a,20h
mov SADC0,a                ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START                  ; high pulse on start bit to initiate conversion
set START                  ; reset 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：使用中断的方式来检测转换结束

```
clr ADE                    ; disable ADC interrupt
mov a,03H
mov SADC1,a                ; select fSYS/8 as A/D clock
set ADCEN
mov a, 02h                 ; setup PAS1 register to configure pin AN0
mov PAS1,a
mov a,20h
mov SADC0,a                ; enable and connect AN0 channel to A/D converter
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 interrupt service routine
ADC_ISR:
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
```

灌电流发生器

无论 V_{ISINK} 电压为 1.0V~3.6V 内何值，灌电流发生器都可以提供恒定的电流。恒定电流值通过 ISGDATA0/ISGDATA1 寄存器控制，灌电流范围为 50mA~360mA。



灌电流发生器寄存器

灌电流发生器的所有工作由一系列寄存器控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
ISGENC	ISGEN	—	—	—	—	—	ISGS1	ISGS0
ISGDATA0	—	—	—	D4	D3	D2	D1	D0
ISGDATA1	—	—	—	D4	D3	D2	D1	D0

灌电流发生器寄存器列表

- **ISGENC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ISGEN	—	—	—	—	—	ISGS1	ISGS0
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	0	0

Bit 7 **ISGEN**：灌电流发生器使能控制
0：除能
1：使能
当 ISGEN 位为 0 时将除能灌电流发生器，此时 ISINK0 和 ISINK1 引脚状态为：V_{ISINK0}&V_{ISINK1} = 浮空，I_{SINK0}&I_{SINK1}=0。

Bit 6~2 未定义，读为“0”

Bit 1 **ISGS1**：ISINK1 引脚灌电流使能 / 除能控制
0：除能
1：使能

Bit 0 **ISGS0**：ISINK0 引脚灌电流使能 / 除能控制
0：除能
1：使能

- **ISGDATA0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	D4	D3	D2	D1	D0
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　**D4~D0**：ISINK0 引脚灌电流发生器控制
灌电流值 (mA)=50+10×(ISGDATA0[4:0])
更多内容请参考“灌电流发生器电气特性”。

- **ISGDATA1 寄存器**

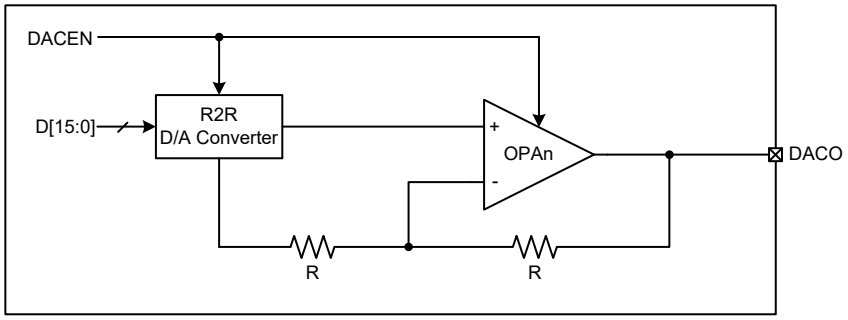
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	D4	D3	D2	D1	D0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5　未定义，读为“0”

Bit 4~0　**D4~D0**：ISINK1 引脚灌电流发生器控制
灌电流值 (mA)=50+5×(ISGDATA1[4:0])
更多内容请参考“灌电流发生器电气特性”。

16-bit 语音 D/A 转换器

此单片机内置一个 10-bit D/A 转换器，该电路是用于音频应用的 16-bit R2R D/A 转换器。16-bit 语音 D/A 转换器的参考电压源仅来自模拟电源并能够降压以降低功耗。16-bit 语音 D/A 转换器适用于语音或音频应用。虽然该 D/A 转换器不是将通常的数据一对一地转换为模拟信号，但它提供了良好及等同的音频质量，无论声音音量的大小。应注意，16-bit 语音 D/A 转换器的电压通过 OPAn 放大并缓存后输出。



16-bit 语音 D/A 转换器结构

16-bit 语音 D/A 转换器寄存器

16-bit 语音 D/A 转换器所有的操作由三个寄存器控制。一个 16-bit 语音 D/A 转换器数据高字节寄存器 DAH，一个 16-bit 语音 D/A 转换器数据低字节寄存器 DAL 和一个用于控制 16-bit 语音 D/A 转换器使能或除能的控制寄存器 DACC。

寄存器名称	位							
	7	6	5	4	3	2	1	0
DAH	D15	D14	D13	D12	D11	D10	D9	D8
DAL	D7	D6	D5	D4	D3	D2	D1	D0
DACC	—	—	—	—	—	—	—	DACEN

16-bit 语音 D/A 转换器寄存器列表

- **DAH 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**：16-bit 语音 D/A 转换器数据高字节

如果需要更新 16-bit 语音 D/A 转换器数据应先更新 16-bit 语音 D/A 转换器数据低字节寄存器 DAL 再进行 DAH 寄存器的更新。每次写入 DAH 寄存器时，16-bit 的数据都将加载到 D/A 转换器中并启动转换循环。应注意，在更新 D/A 转换器数据之前需先使能 D/A 转换器。

- **DAL 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~D**：16-bit 语音 D/A 转换器数据低字节

写数据至该寄存器时，数据仅写入影子缓存器；写数据至 DAH 寄存器时，数据直接写入高字节寄存器，同时锁存在影子缓存器中的数据写入到 DAL 寄存器。应注意，在更新 D/A 转换器数据之前需先使能 D/A 转换器。

- **DACC 寄存器**

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

Bit 7~1 未定义，读为“0”

Bit 0 **DACEN**：16-bit 语音 D/A 转换器使能 / 除能控制位

0：除能

1：使能

16-bit 语音 D/A 转换器使能后，电路稳定需要一定的延时 t_{DACS}，需确保 16-bit 语音 D/A 转换器电路在 16-bit 数据寄存器更新前已稳定。

串行接口模块 – SIM

此单片机内有一个串行接口模块，包括两种易与外部设备通信的串行接口：四线 SPI 或两线 I^2C 接口。这两种接口具有相当简单的通信协议，单片机可以通过这些接口与传感器、闪存或 EEPROM 内存等硬件设备通信。因 SIM 接口引脚是与其它 I/O 引脚共用，因此在使用 SIM 功能前，要先通过相应的引脚共用功能选择寄存器选定 SIM 引脚功能。因为这两种接口 SPI 和 I^2C 共用引脚和寄存器，所以要通过一个 SIMC0 寄存器中的 SIM2~SIM0 位来选择哪一种通信接口。若 SIM 功能使能，可通过上拉电阻控制寄存器选择与输入 / 输出口共用的 SIM 输入引脚的上拉电阻。

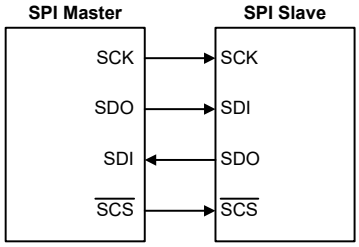
SPI 接口

SPI 接口常用于与外部设备如传感器、闪存或 EEPROM 内存等通信。四线 SPI 接口最初是由摩托罗拉公司研制，是一个有相当简单的通信协议的串行数据接口，这个协议可以简化与外部硬件的编程要求。

SPI 通信模式为全双工模式，且能以主 / 从模式的工作方式进行通信，单片机既可以做为主机，也可以做为从机。虽然 SPI 接口理论上允许一个主机控制多个从机，但此处的 SPI 中只有一个片选信号引脚 $\overline{SCS}$。若主机需要控制多个从机，可使用输入 / 输出引脚选择从机。

SPI 接口操作

SPI 接口是一个全双工串行数据传输器。SPI 接口的四线为：SDI、SDO、SCK 和 $\overline{SCS}$。SDI 和 SDO 是数据的输入和输出线。SCK 是串行时钟线，$\overline{SCS}$ 是从机的选择线。SPI 的接口引脚与普通 I/O 口和 I^2C 的功能脚共用。通过设定 SIMC0/SIMC2 寄存器的对应位，来使能 SPI 接口。SPI 可以通过 SIMC0 寄存器中的 SIMEN 位来除能或使能。连接到 SPI 接口的单片机以主机 / 从机模式进行通信，且所有的数据传输由主机发起，并控制时钟信号。由于单片机只有一个 $\overline{SCS}$ 引脚，所以只能拥有一个从机设备。可通过软件控制 $\overline{SCS}$ 引脚使能与除能，设置 CSEN 位为“1”使能 $\overline{SCS}$ 功能，设置 CSEN 位为“0”，$\overline{SCS}$ 引脚将处于浮空状态。

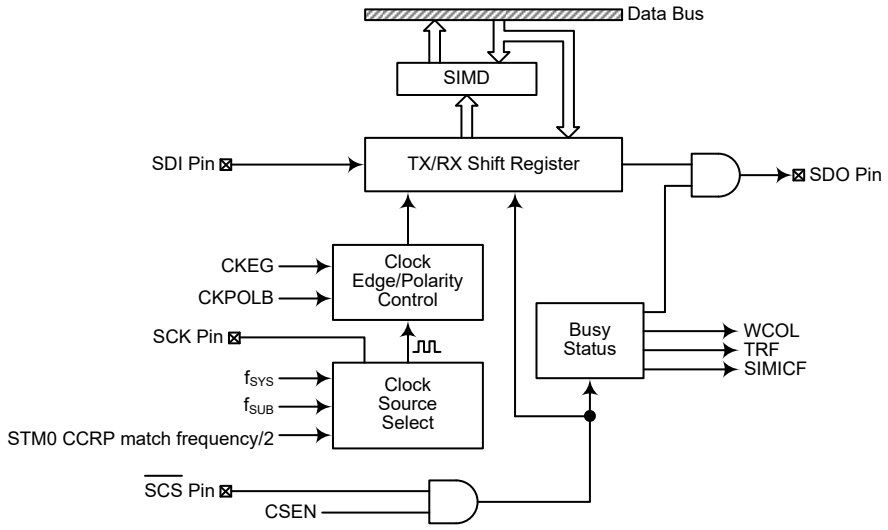


SPI 主 / 从机连接方式

该单片机的 SPI 功能具有以下特点：

- 全双工同步数据传输
- 主从模式
- 最低有效位先传或最高有效位先传的数据传输模式
- 传输完成标志位
- 时钟源上升沿或下降沿有效

SPI 接口状态受很多因素的影响，如单片机处于主机或从机的工作模式和 CSEN，SIMEN 位的状态。



SPI 方框图

SPI 寄存器

有三个内部寄存器用于控制 SPI 接口的所有操作，其中有一个数据寄存器 SIMD、两个控制寄存器 SIMC0 和 SIMC2。注意，SIMC1 寄存器仅用于 I²C 接口控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0

SPI 寄存器列表

SPI 数据寄存器

SIMD 用于存储发送和接收的数据。这个寄存器由 SPI 和 I²C 功能所共用。在单片机将数据写入到 SPI 总线之前，要传输的数据应先存在 SIMD 中。SPI 总线接收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 SPI 传输或接收的数据都必须通过 SIMD 实现。

- **SIMD 寄存器**

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

“x”：未知

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

SPI 控制寄存器

单片机中也有两个控制 SPI 接口功能的寄存器，SIMC0 和 SIMC2。应注意的是 SIMC2 与 I²C 接口功能中的寄存器 SIMA 是同一个寄存器。SPI 功能不会用到寄存器 SIMC1，SIMC1 寄存器仅在工作于 I²C 接口时才被使用。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC2 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

- **SIMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0**：SIM 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 STM0 CCRP 匹配频率 / 2
101：SPI 从机模式
110：I²C 从机模式
111：未使用模式
这几位用于设置 SIM 功能的工作模式，除了选择 I²C 或 SPI 功能，还可选择 SPI 的主从模式和 SPI 的主机时钟频率。SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 STM0。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 未定义，读为“0”

Bit 3~2 **SIMDEB1~SIMDEB0**：I²C 去抖时间选择位
这些位只有在 SIM 设置成 I²C 接口模式时才有效。请参考 I²C 寄存器部分。

Bit 1 **SIMEN**：SIM 使能控制位
0：除能
1：使能
此位为 SIM 接口的开 / 关控制位。此位为“0”时，SIM 接口除能，SDI、SDO、SCK 和 $\overline{SCS}$ 或 SDA 和 SCL 脚将失去 SPI 或 I²C 功能，SIM 工作电流减小到最小值。此位为“1”时，SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I²C 接口，当 SIMEN 位由低到高转变时，I²C 控制寄存器中的设置，如 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 **SIMICF**：SIM SPI 未完成标志位
0：SPI 传输未完成状况未发生
1：SPI 传输未完成状况发生
此位仅当 SIM 配置在 SPI 从机模式时有效。如果 SPI 工作在从机模式且 SIMEN 和 CSEN 位都为“1”，但在 SPI 数据传输完全结束前 $\overline{SCS}$ 线被外部主机拉高，SIMICF 和 TRF 位都会被置高。在这种情况下，如果相应的中断功能使能将产生一个中断。然而，如果 SIMICF 位是由软件应用程序设为 1，那么 TRF 位将不会置高。

- **SIMC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
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 **D7~D6**：未定义位
用户可通过软件程序对这两位进行读写。

Bit 5 **CKPOLB**：SPI 时钟线的基础状态位
0：当时钟无效时，SCK 引脚为高电平
1：当时钟无效时，SCK 引脚为低电平
此位决定了时钟线的基础状态，若此位为高，当时钟无效时 SCK 为低电平，若此位为低，当时钟无效时 SCK 为高电平。

Bit 4 **CKEG**：SPI 的 SCK 有效时钟边沿类型位
CKPOLB=0
0：SCK 为高电平且在 SCK 上升沿抓取数据
1：SCK 为高电平且在 SCK 下降沿抓取数据
CKPOLB=1
0：SCK 为低电平且在 SCK 下降沿抓取数据
1：SCK 为低电平且在 SCK 上升沿抓取数据
CKEG 和 CKPOLB 位用于设置 SPI 总线上时钟信号输入和输出方式。这两位必须在执行数据传输前先被设置好，否则将产生错误的时钟边沿信号。CKPOLB 位决定时钟线的基本状态，若时钟无效且此位为高，则 SCK 为低电平，若时钟无效且此位为低，则 SCK 为高电平。CKEG 位决定有效时钟边沿类型，取决于 CKPOLB 的状态。

Bit 3 **MLS**：SPI 数据移位顺序控制位
0：LSB 优先
1：MSB 优先
数据移位顺序选择位，用于选择数据传输时高位优先传输还是低位优先传输。此位设置为高时高位优先传输，为低时低位优先传输。

Bit 2 **CSEN**：SPI $\overline{\text{SCS}}$ 引脚控制位
0：除能
1：使能
CSEN 位用于 $\overline{\text{SCS}}$ 引脚的使能 / 除能控制。此位为低时，$\overline{\text{SCS}}$ 除能并处于浮空状态。此位为高时，$\overline{\text{SCS}}$ 作为选择脚。

Bit 1 **WCOL**：SPI 写冲突标志位
0：无冲突
1：冲突
WCOL 标志位用于监测数据冲突的发生。此位为高时，表示在传输过程中有数据被写入 SIMD 寄存器。若数据正在被传输时，此写操作无效。此位可被应用程序清零。

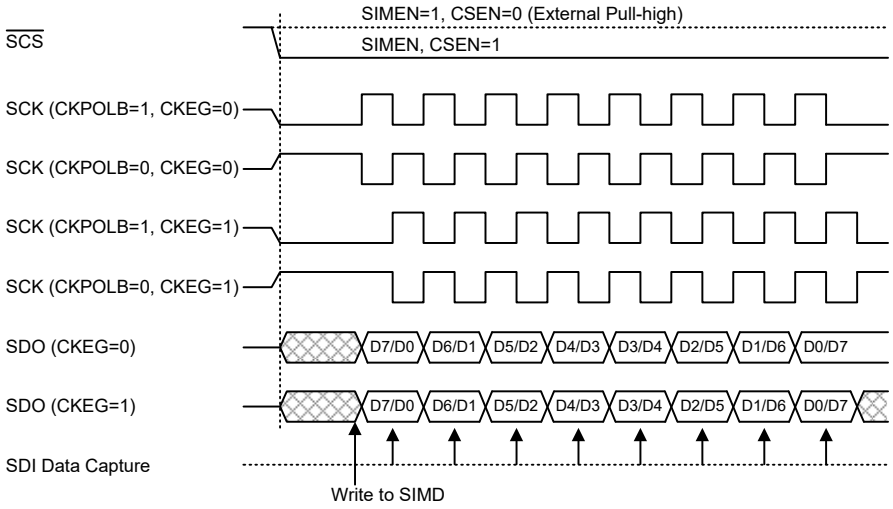
Bit 0 **TRF**：SPI 发送 / 接收结束标志位
0：数据正在发送
1：数据发送结束
TRF 位为发送 / 接收结束标志位，当 SPI 数据传输结束时，此位自动置为高，但须通过应用程序设置为“0”。此位也可用于产生中断。

SPI 通信

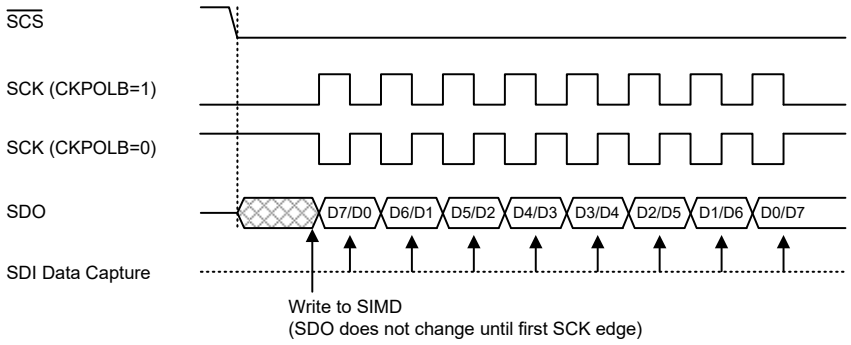
将 SIMEN 设置为高，使能 SPI 功能之后，单片机处于主机模式，当数据写入到寄存器 SIMD 的同时传输 / 接收开始进行。数据传输完成时，TRF 位将自动被置位但只能通过应用程序清零。单片机处于从机模式时，收到主机发来的信号之后，会传输 SIMD 中的数据，而且在 SDI 引脚上的数据也会被移位到 SIMD

寄存器中。主机应在输出时钟信号之前先输出一个 $\overline{SCS}$ 信号以使能从机，从机的数据传输功能也应在与 SCK 信号相关的适当时候准备就绪，这由 CKPOLB 和 CKEG 位决定。所附时序图表明了在 CKPOLB 和 CKEG 位各种设置情况下从机数据与 SCK 信号的关系。

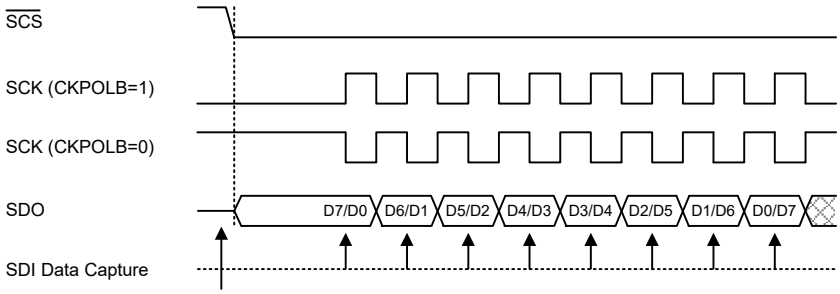
即使在单片机处于空闲模式时，若 SPI 接口使用的时钟源仍开启，SPI 功能仍将继续执行。



SPI 主机模式时序

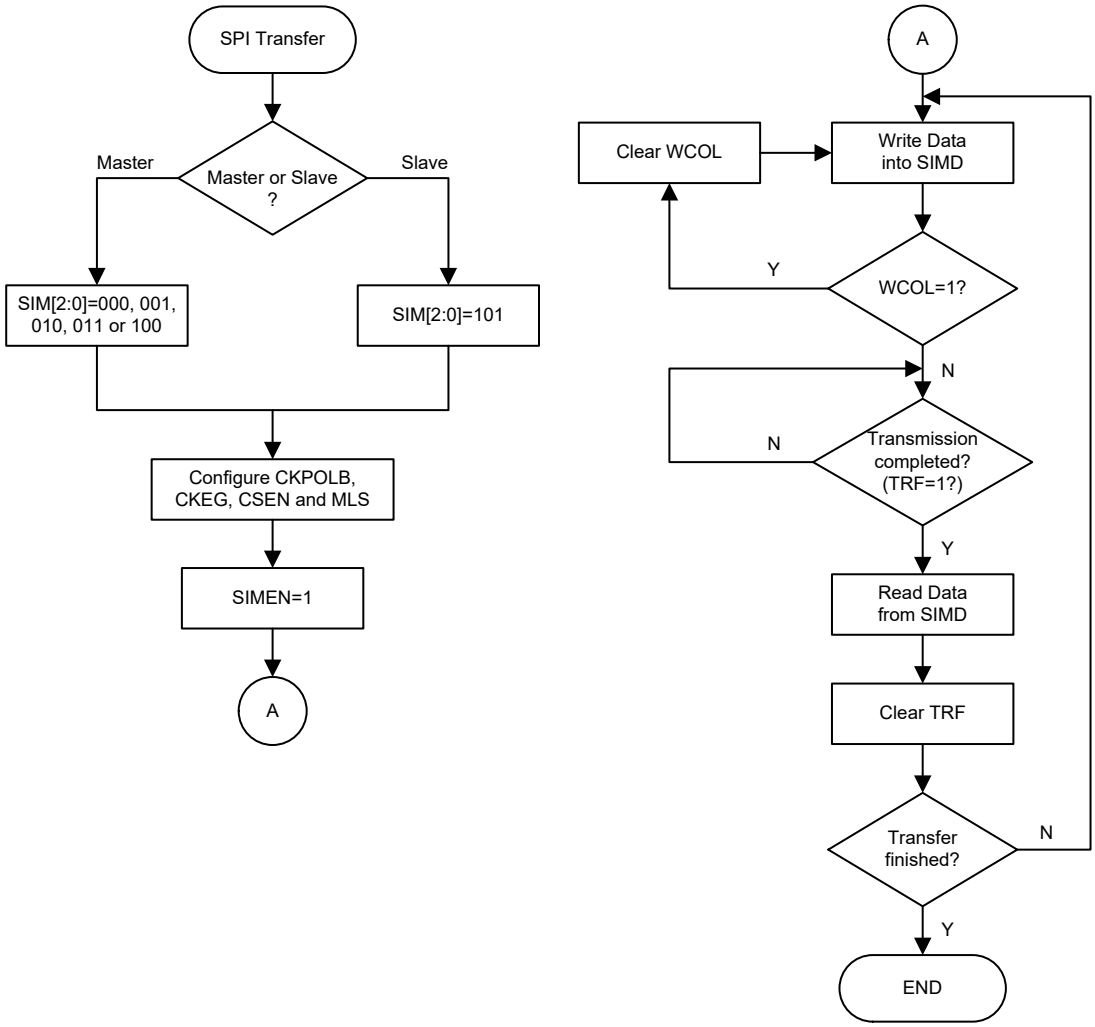


SPI 从机模式时序 – CKEG=0



Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignores the $\overline{SCS}$ level.

SPI 从机模式时序 – CKEG=1



SPI 传输控制流程图

SPI 使能 / 除能

设置 CSEN=1、$\overline{SCS}$=0 将使能 SPI 总线，然后等待写数据到 SIMD 寄存器 (TXRX 缓存器)。单片机处于主机模式，数据写入 SIMD 寄存器后，自动开始数据传输或接收操作。数据传输完成时，TRF 位将自动被置位。单片机处于从机模式，SCK 引脚上收到脉冲信号之后，会传输 TXRX 中的数据，或将 SDI 引脚上的数据移入。

当 SPI 总线除能时，通过设置相应共用功能选择控制位，SCK、SDI、SDO、$\overline{SCS}$ 可作为 I/O 口或其它功能引脚使用。

SPI 操作步骤

四线制 SPI 接口可完成所有主 / 从模式通信工作。

在 SIMC2 寄存器中，CSEN 位控制 SPI 接口的所有功能。设置此位为高，$\overline{SCS}$ 信号线有效将使能 SPI 接口。设置此位为低，SPI 接口将除能，$\overline{SCS}$ 信号线处于浮空状态因此不能控制 SPI 接口。CSEN 位和 SIMC0 寄存器中的 SIMEN 位设置为高，使得 SDI 信号线处于浮空状态且 SDO 信号线为高电平。主机模式中，如果 SCK 信号线为高还是低取决于 SIMC2 寄存器中的时钟极性选择位

CKPOLB。从机模式中，SCK 信号线处于浮空状态。如果 SIMEN 位设置为低，SPI 接口被除能，通过设置相应引脚共用控制位，$\overline{SCS}$、SDI、SDO 和 SCK 可作为 I/O 口或其它功能引脚使用。主机模式中，当数据被写入 SIMD 寄存器后，主机发起数据传输，并控制时钟信号。从机模式中，由外部主机发出数据传送 / 接收时钟信号。下面介绍主从模式中数据传输步骤。

主机模式

- 步骤 1
  设置 SIMC0 控制寄存器中的 SIM2~SIM0 位，选择 SPI 主机模式和时钟源。
- 步骤 2
  设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，这必须与从机设备一致。
- 步骤 3
  设置 SIMC0 控制寄存器中的 SIMEN 位，使能 SPI 接口功能。
- 步骤 4
  对于写操作：写数据到 SIMD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。再使用 SCK 和 SDO 信号线将数据输出。跳至步骤 5。
  对于读操作：从 SDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMD 寄存器。
- 步骤 5
  检测 WCOL 位，若此位为高，则发生数据冲突并跳回至步骤 4；若为低，则继续执行下面的步骤。
- 步骤 6
  检测 TRF 位或等待 SPI 串行总线中断发生。
- 步骤 7
  从 SIMD 寄存器中读数据。
- 步骤 8
  清除 TRF。
- 步骤 9
  跳回至步骤 4。

从机模式

- 步骤 1
  设置 SIMC0 控制寄存器中的 SIM2~SIM0 位，选择 SPI 从机模式。
- 步骤 2
  设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，这必须与主机设备一致。
- 步骤 3
  设置 SIMC0 控制寄存器中的 SIMEN 位，使能 SPI 接口功能。
- 步骤 4
  对于写操作：写数据到 SIMD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。等待主机时钟 SCK 信号和 $\overline{SCS}$ 信号。跳至步骤 5。
  对于读操作：从 SDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMD 寄存器。

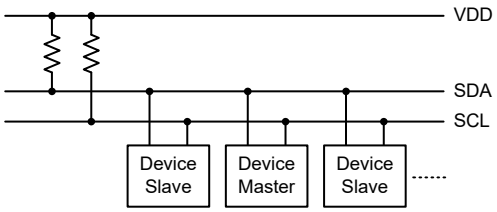
- 步骤 5
  检测 WCOL 位，若此位为高，则发生数据冲突并跳回至步骤 4；若为低，则继续执行下面的步骤。
- 步骤 6
  检测 TRF 位或等待 SPI 串行总线中断发生。
- 步骤 7
  从 SIMD 寄存器中读数据。
- 步骤 8
  清除 TRF。
- 步骤 9
  跳回至步骤 4。

错误侦测

SIMC2 寄存器中的 WCOL 位用于数据传输期间监测数据冲突的发生。此位由 SPI 串行接口设置为高，而由应用程序来清除为零。在数据传输期间如果写数据到 SIMD，此位被置高提示数据冲突发生，并阻止数据继续被写入。

I²C 接口

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

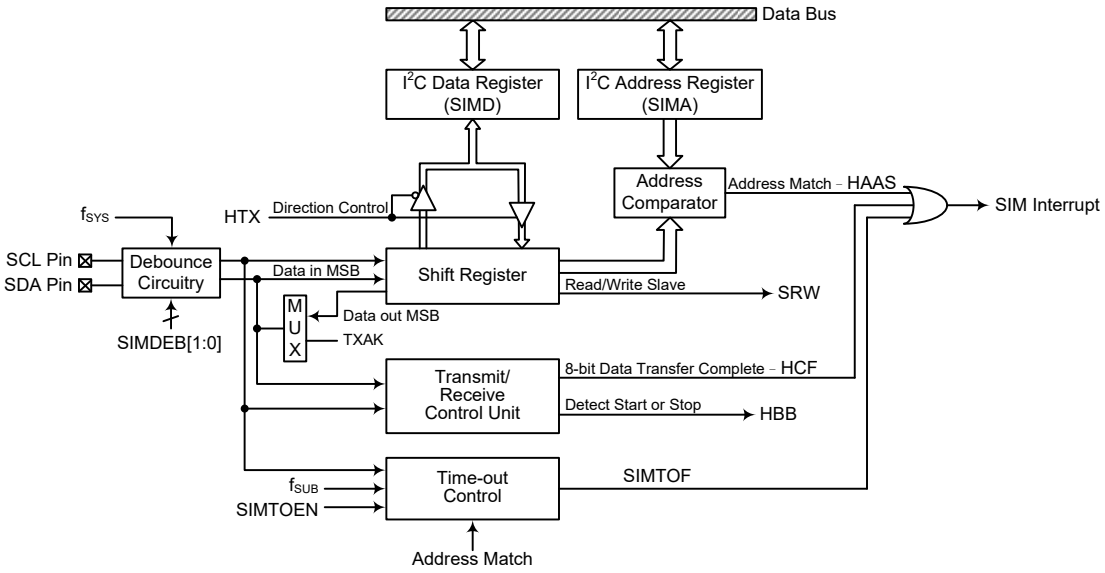


I²C 主从总线连接图

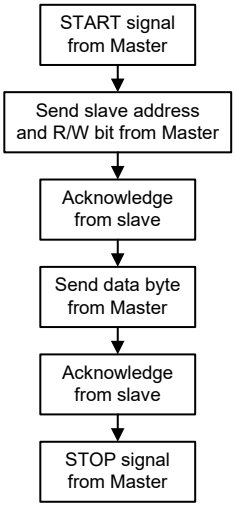
I²C 接口操作

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

如果有两个设备通过双向的 I²C 总线进行通信，那么就存在一个主机和一个从机。主机和从机都可以用于传输和接收数据，但只有主机才可以控制总线动作。那些处于从机模式的设备，要在 I²C 总线上传输数据只有两种方式，一是从机发送模式，二是从机接收模式。即使 I²C 设备被激活，与 SCL/SDA 引脚共用的 I/O 口上拉电阻控制功能仍有效，其上拉电阻功能由相应的端口上拉电阻控制寄存器控制。



I²C 方框图



I²C 接口操作

SIMDEB1 和 SIMDEB0 位决定 I²C 接口的去抖时间。这个功能可以使用内部时钟在外部时钟上增加一个去抖间隔，减小时钟线上毛刺发生的可能性，以避免单片机发生误动作。如果选择了这个功能，去抖时间可以选择 2 个或 4 个系统时钟。为了达到需要的 I²C 数据传输速度，系统时钟 f_{SYS} 和 I²C 去抖时间之间存在一定的关系。I²C 标准模式或者快速模式下，用户需注意所选的系统时钟频率与标准匹配去抖时间的设置，其具体关系如下表所示。

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

I²C 最小 f_{SYS} 频率要求

I²C 寄存器

I²C 总线有三个控制寄存器 SIMC0、SIMC1 和 SIMTOC，一个地址寄存器 SIMA 以及一个数据寄存器 SIMD。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C 寄存器列表

I²C 数据寄存器

SIMD 用于存储发送和接收的数据。这个寄存器由 SPI 和 I²C 功能所共用。在单片机将数据写入到 I²C 总线之前，要传输的数据应先存在 SIMD 中。I²C 总线接收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 I²C 传输或接收的数据都必须通过 SIMD 实现。

- **SIMD 寄存器**

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

“x”：未知

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

I²C 地址寄存器

SIMA 寄存器也在 SPI 接口功能中使用，但其名称改为 SIMC2。SIMA 寄存器用于存放 7 位从机地址，寄存器 SIMA 中的 bit 7 ~ bit 1 是单片机的从机地址，bit 0 未定义。

如果接至 I²C 的主机发送出的地址和寄存器 SIMA 中存储的地址相符，那么就选中了这个从机。应注意的是寄存器 SIMA 和 SPI 接口使用的寄存器 SIMC2 共用同一个寄存器地址。

- **SIMA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	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 7~1　**SIMA6~SIMA0**：I²C 从机地址位
SIMA6~SIMA0 是 7 位从机地址 bit 6 ~ bit 0。

Bit 0　**D0**：保留位，此位可通过软件程序进行读或写

I²C 控制寄存器

单片机中有三个控制 I²C 接口功能的寄存器，SIMC0、SIMC1 和 SIMTOC。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC1 包括多个用于指示 I²C 传输状态的相关标志位。SIMTOC 寄存器用于控制 I²C 超时功能，此寄存器在 I²C 超时章节介绍。

- **SIMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0**：SIM 工作模式控制位
000：SPI 主机模式；SPI 时钟为 f_{SYS}/4
001：SPI 主机模式；SPI 时钟为 f_{SYS}/16
010：SPI 主机模式；SPI 时钟为 f_{SYS}/64
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 STM0 CCRP 匹配频率 / 2
101：SPI 从机模式
110：I²C 从机模式
111：未使用模式
这几位用于设置 SIM 功能的工作模式，用于选择 SPI 的主从模式和 SPI 的主机时钟频率及 I²C 或 SPI 功能。SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 STM0。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 未定义，读为“0”

Bit 3~2 **SIMDEB1~SIMDEB0**：I²C 去抖时间选择位
00：无去抖时间
01：2 个系统时钟去抖时间
1x：4 个系统时钟去抖时间
当设置 SIM2~SIM0 位为“110”将 SIM 设置为 I²C 接口功能时，这两个位用于选择 I²C 去抖时间。

Bit 1 **SIMEN**：SIM 使能控制位
0：除能
1：使能
此位为 SIM 接口的开 / 关控制位。此位为“0”时，SIM 接口除能，SDI、SDO、SCK 和 $\overline{SCS}$ 或 SDA 和 SCL 脚将失去 SPI 或 I²C 功能，SIM 工作电流减小到最小值。此位为“1”时，SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I²C 接口，当 SIMEN 位由低到高转变时，I²C 控制寄存器中的设置，如 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 **SIMICF**：SIM SPI 未完成标志位
此位仅当 SIM 配置在 SPI 从机模式时有效。请参考 SPI 寄存器部分。

- **SIMC1 寄存器**

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

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

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

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

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

Bit 3 **TXAK**：I²C 总线发送应答标志位
0：从机发送应答标志
1：从机没有发送应答标志
从机接收完 8 位数据之后，该位将在第九个从机时钟时被传到总线上。如果从机想要接收更多的数据，则应在接收数据之前将此位设置为“0”。

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

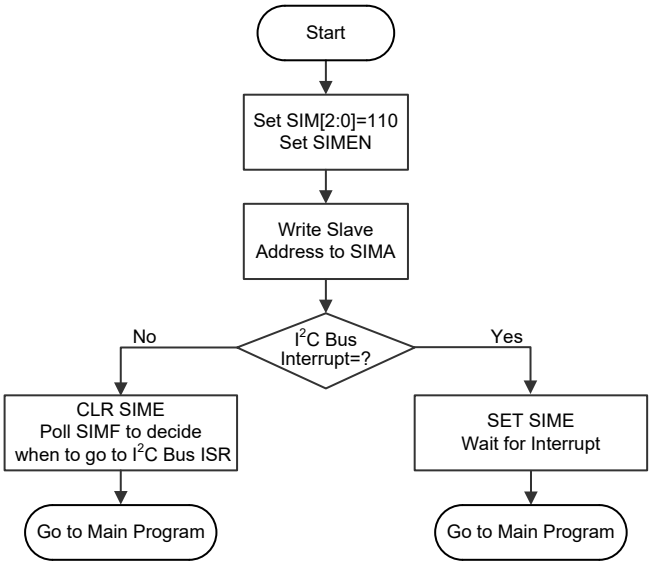
Bit 1 **IAMWU**：I²C 地址匹配唤醒控制位
0：除能
1：使能
此位设置为“1”则使能 I²C 地址匹配使系统从休眠或空闲模式中唤醒的功能。若进入休眠或空闲模式前 IAMWU 已经置高以使能 I²C 地址匹配唤醒功能，在系统唤醒后须软件清除此位以确保单片机正确地运行。

Bit 0 **RXAK**：I²C 总线接收应答标志位
0：从机接收到应答标志
1：从机没有接收到应答标志
RXAK 位是接收应答标志位。如果 RXAK 位为“0”，即表示 8 位数据传输之后，从机在第九个时钟有接收到一个应答信号。如果从机处于发送状态，从机作为发送方会检查 RXAK 位来判断主机接收方是否愿意继续接收下一个字节。因此发送方会一直发送数据，直到 RXAK 为“1”时才停止发送数据。这时，发送方将释放 SDA 线，主机方可发出停止信号从而释放 I²C 总线。

I²C 总线通信

I²C 总线上的通信需要四步完成，一个起始信号，一个从机地址发送，一个数据传输，还有一个停止信号。当起始信号被写入 I²C 总线时，总线上的所有从机都会接收到这个起始信号并且被通知总线上即将有数据到达。数据的前 7 位是从机地址，高位在前，低位在后。如果发出的地址和从机地址匹配，SIMC1 寄存器的 HAAS 位会被置位，同时产生 I²C 中断。进入中断服务程序后，系统要检测 HAAS 位和 SIMTOF 位，以判断 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕，或是来自 I²C 超时。在数据传递中，要注意的是，在 7 位从机地址被发送后，接下来的一位，即第 8 位，是读 / 写控制位，该位的值会反映到 SRW 位中。从机通过检测 SRW 位以确定自己是要进入发送模式还是接收模式。在 I²C 总线开始传送数据前，需要先初始化 I²C 总线，初始化 I²C 总线步骤如下：

- 步骤 1
  设置 SIMC0 寄存器中 SIM2~SIM0 位为“110”和 SIMEN 位为“1”，以使能 I²C 总线。
- 步骤 2
  向 I²C 总线地址寄存器 SIMA 写入从机地址。
- 步骤 3
  设置相关中断使能控制位使能 SIM 中断。



I²C 总线初始化流程图

I²C 总线起始信号

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

从机地址

总线上的所有从机都会侦测由主机发出的起始信号。发送起始信号后，紧接着主机会发送从机地址以选择要进行数据传输的从机。所有在 I²C 总线上的从机

接收到 7 位地址数据后，都会将其与各自内部的地址进行比较。如果从机从主机上接收到的地址与自身内部的地址相匹配，则会产生一个 I²C 总线中断信号。地址位接下来的一位为读 / 写状态位 (即第 8 位)，将被保存到 SIMC1 寄存器的 SRW 位，从机随后发出一个低电平应答信号 (即第 9 位)。当从机地址匹配时，从机会将状态标志位 HAAS 置位。

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

I²C 总线读 / 写信号

SIMC1 寄存器的 SRW 位用来表示主机是要从 I²C 总线上读取数据还是要将数据写到 I²C 总线上。从机通过检测该位以确定自己是作为发送方还是接收方。当 SRW 置“1”，表示主机要从 I²C 总线上读取数据，从机则作为发送方，将数据写到 I²C 总线；当 SRW 清“0”，表示主机要写数据到 I²C 总线上，从机则做为接收方，从 I²C 总线上读取数据。

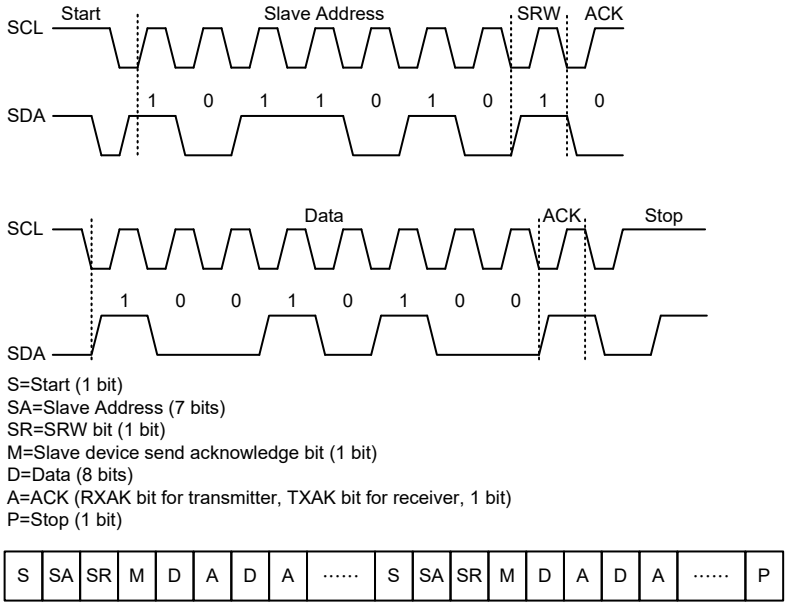
I²C 总线从机地址应答信号

主机发送呼叫地址后，当 I²C 总线上的任何从机内部地址与其匹配时，会发送一个应答信号。此应答信号会通知主机有从机已经接收到了呼叫地址。如果主机没有收到应答信号，则主机必须发送停止 (STOP) 信号以结束通信。当 HAAS 为高时，表示从机接收到的地址与自己内部地址匹配，则从机需检查 SRW 位，以确定自己是作为发送方还是作为接收方。如果 SRW 位为高，从机须设置成发送方，这样会置位 SIMC1 寄存器的 HTX 位。如果 SRW 位为低，从机须设置成接收方，这样会清零 SIMC1 寄存器的 HTX 位。

I²C 总线数据和应答信号

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

当接收器想要继续接收下一个数据时，必须在第 9 个时钟发出应答信号 (TXAK)。被设为发送方的从机将检测寄存器 SIMC1 中的 RXAK 位以判断是否传输下一个字节的数据，如果从机不传输下一个字节，那么它将释放 SDA 线并等待接收主机的停止信号。

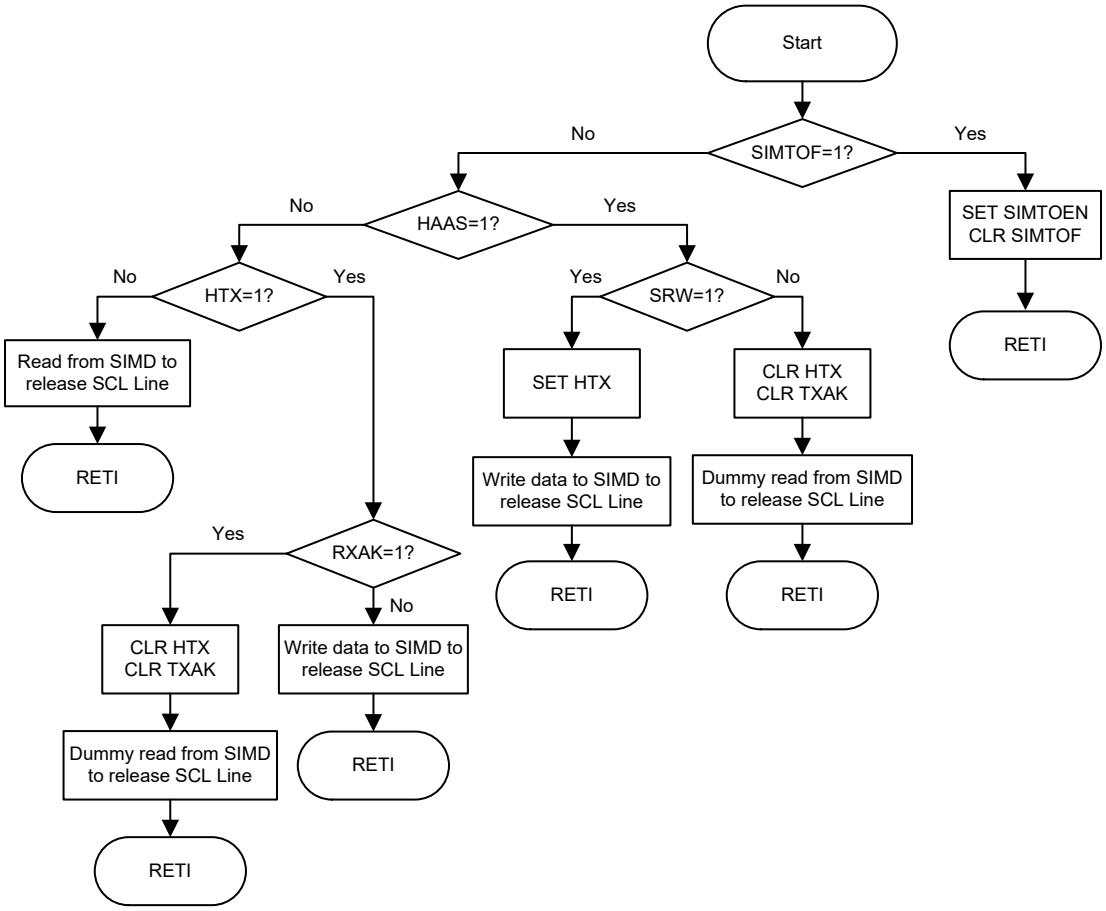


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

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

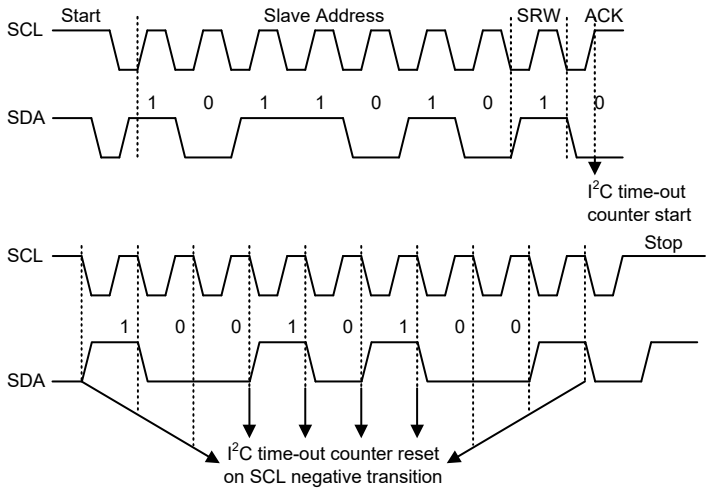
I^2C 通信时序图



I²C 总线 ISR 流程图

I²C 超时控制

超时功能可减少 I²C 接收错误的时钟源而引起的锁死问题。如果连接到 I²C 总线的时钟源经过一段时间还未接收到，则在一定的超时周期后，I²C 电路和寄存器将复位。超时计数器在 I²C 总线“START”和“地址匹配”条件下开始计数，且在 SCL 下降沿清零。在下一个 SCL 下降沿到来之前，如果超时时间大于 SIMTOC 寄存器指定的超时周期，则超时发生。I²C“STOP”条件发生时超时功能终止。



I²C 超时时序图

当 I²C 超时计数器溢出时，计数器将停止计数，SIMTOEN 位被清零，且 SIMTOF 位被置高以表明超时计数器中断发生。超时计数器中断使用的也是 I²C 中断向量。当 I²C 超时发生时，I²C 内部电路会被复位，寄存器也将发生如下复位情况。

寄存器	I²C 超时发生后
SIMD, SIMA, SIMC0	保持不变
SIMC1	复位至 POR

超时发生后的 I²C 寄存器

SIMTOF 标志位由应用程序清零。共有 64 个超时周期，可通过 SIMTOC 寄存器的 SIMTOS 字段进行选择。超时周期可通过公式计算：$((1\sim64)\times(32/f_{SUB}))$。由此可得超时周期范围为 1ms~64ms。

- **SIMTOC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
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 **SIMTOEN**：SIM I²C 超时控制位
0：除能
1：使能

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

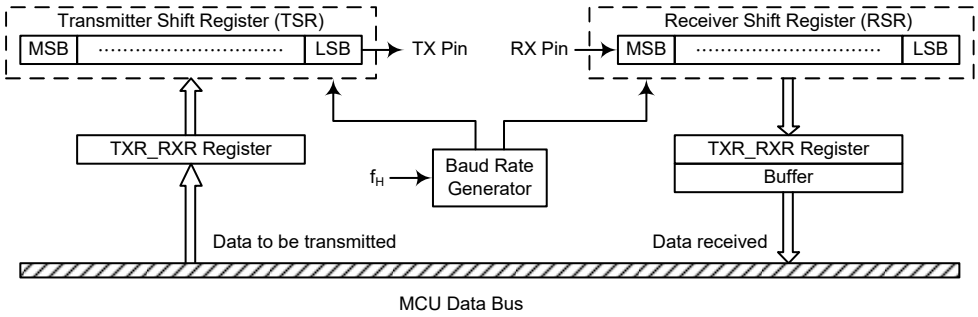
Bit 5~0 **SIMTOS5~SIMTOS0**：SIM I²C 超时时间选择位
I²C 超时时钟源是 $f_{SUB}/32$
I²C 超时时间计算方法：$(SIMTOS[5:0]+1)\times(32/f_{SUB})$

UART 接口

该单片机具有一个全双工的异步串行通信接口，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 UART 中断。

内置的 UART 功能包含以下特性：

- 全双工通用异步接收器 / 发送器
- 8 位或 9 位传输格式
- 奇校验、偶校验或无校验
- 1 位或 2 位停止位
- 8 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
- 2-byte FIFO 接收数据缓冲器
- RX 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

UART 外部引脚

内部 UART 有两个外部引脚 TX 和 RX，可与外部串行接口进行通信。TX 和 RX 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 引脚功能。当 UARTEN 和 TXEN/RXEN 位置高时，将自动设置这些 I/O 脚或其它共用功能脚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 引脚功能后，TX 或 RX 引脚将处于浮空状态。这时 TX 或 RX 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 数据传输方案

前面方框图显示了 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 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 功能相关的有五个寄存器，包括控制 UART 模块整体功能的 USR、UCR1 和 UCR2 寄存器，控制波特率的 BRG 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	PRT	STOPS	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0

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 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
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，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 和 UCR2 是 UART 的两个控制寄存器，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等等。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT	STOPS	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 脚处于浮空状态
1：UART 使能，TX 和 RX 脚作为 UART 功能引脚
此位为 UART 的使能位。UARTEN=0，UART 除能，RX 和 TX 处于浮空状态；UARTEN=1，UART 使能，TX 和 RX 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 清零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 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**：发送数据位数选择位
0：8-bit 传输数据
1：9-bit 传输数据
BNO 是发送数据位数选择位。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。

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 **PRT**：奇偶校验选择位
0：偶校验
1：奇校验
奇偶校验选择位。PRT=1，奇校验；PRT=0，偶校验。

Bit 3 **STOPS**：停止位的长度选择位
0：有一位停止位
1：有两位停止位
此位用来设置停止位的长度。STOP=1，有两位停止位；STOP=0，只有一位停止位。

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。它也可用来控制波特率，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	BRGH	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 引脚将处于浮空状态。若 RXEN=1 且 UARTEN=1，则接收将被使能，RX 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 引脚将处于浮空状态。

Bit 5 **BRGH**：波特率发生器高低速选择位
0：低速波特率
1：高速波特率
此位为波特率发生器高低速选择位，它和 BRG 寄存器一起控制 UART 的波特率。BRGH=1，为高速模式；BRGH=0，为低速模式。

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 脚下降沿唤醒 UART 功能使能位
0：RX 脚下降沿唤醒 UART 功能除能
1：RX 脚下降沿唤醒 UART 功能使能
此位用于控制 RX 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，则无 RX 引脚唤醒 UART 功能无效。若此位置高且 UART 时钟 f_H 关闭，当 RX 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 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 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。

- **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 引脚正在接收的数据。

Bit	7	6	5	4	3	2	1	0
Name	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
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 **TXRX7~TXRX0**：UART 发送 / 接收数据位 Bit 7~Bit 0

- **BRG 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
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 **BRG7~BRG0**：波特率值
软件设置 BRGH 位 (设置波特率发生器的速度) 和 BRG 寄存器 (设置波特率的值)，一起控制 UART 的波特率。
注：若 BRGH=0，波特率 = $f_H/[64\times(N+1)]$；
若 BRGH=1，波特率 = $f_H/[16\times(N+1)]$。

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 8 位计数器产生，它由 BRG 寄存器和 UCR2 寄存器的 BRGH 位来控制。BRGH 是决定波特率发生器处于高速模式还是低速模式，从而决定计算公式的选用。BRG 寄存器的值 N 可根据下表中的公式计算，N 的范围是 0 到 255。

UCR2 的 BRGH 位	0	1
波特率 (BR)	$f_H/[64(N+1)]$	$f_H/[16(N+1)]$

为得到相应的波特率，首先需要设置 BRGH 来选择相应的计算公式从而算出 BRG 的值。由于 BRG 的值不连续，所以实际波特率和理论值之间有一个偏差。下面举例怎样计算 BRG 寄存器中的值 N 和误差。

波特率和误差的计算

若选用 4MHz 时钟频率且 BRGH=0，若期望的波特率为 4800，计算它的 BRG 寄存器的值 N，实际波特率和误差。

根据上表，波特率 $BR=f_H/[64(N+1)]$

转换后的公式 $N=[f_H/(BR\times64)]-1$

带入参数 $N=[4000000/(4800\times64)]-1=12.0208$

取最接近的值，十进制 12 写入 BRG 寄存器，实际波特率如下

$BR=4000000/[64\times(12+1)]=4808$

因此，误差 = (4808-4800)/4800=0.16%

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、偶校验和无校验三种格式。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 UCR1 寄存器的 BNO、PRT、PREN 和 STOPS 设定。用于数据发送和接收的波特率由一个内部的 8 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 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，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 清零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 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。

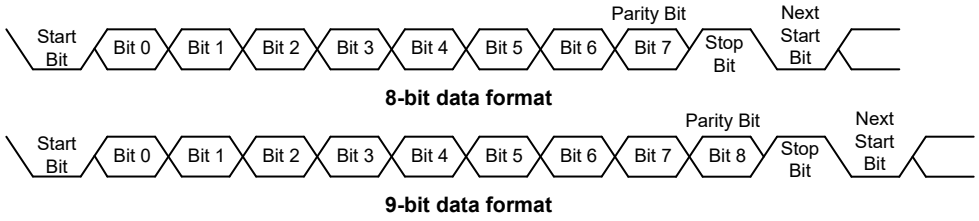
数据位、停止位位数以及奇偶校验的选择

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 寄存器的各个位控制的。BNO 决定数据传输是 8 位还是 9 位；PRT 决定校验类型；PREN 决定是否选择奇偶校验；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。

发送器的启动可由如下步骤完成：

- 正确地设置 BNO、PRT、PREN 和 STOPS 位以确定数据长度、校验类型和停止位长度。
- 设置 BRG 寄存器，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

清除 TXIF 和 TIDLE 软件执行次序相同。

发送暂停字

若 TXBRK=1 保持超过 [(BRG+1)×t_H] 时间且 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 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 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 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 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 寄存器是一个两层的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 TXR_RXR 寄存器，否则忽略第三帧数据并且发生溢出错误。

接收器的启动可由如下步骤完成：

- 正确地设置 BNO、PRT 和 PREN 位以确定数据长度和校验类型。
- 设置 BRG 寄存器，选择期望的波特率。
- 置高 RXEN，使能 UART 接收器且使 RX 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，溢出错误发生之前至多还有一帧数据可读。
- 若 RIE=1，数据从 RSR 寄存器加载到 TXR_RXR 寄存器中将产生中断。
- 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 寄存器是一个两层的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 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

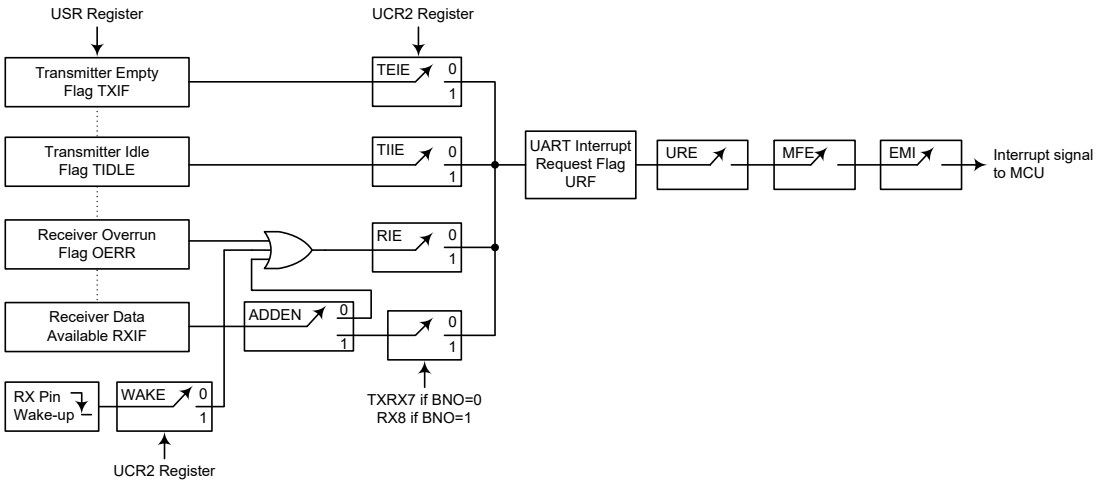
若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 USR 寄存器和 TXR_RXR 寄存器中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 USR 寄存器中的 FERR 和 PERR 错误标志位。

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、接收器数据有效、溢出和地址检测和 RX 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 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 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，其中断请求由 UART 模块决定。



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、接收 / 发送寄存器以及 BRG 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 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 引脚的下降沿可触发产生 RX 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 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。

低电压检测 – LVD

此单片机具有低电压检测功能，即 LVD。该功能使能用于监测电源电压 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 8 个固定电压中的一个参考点。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	VBGEN	VLVD2	VLVD1	VLVD0
R/W	—	—	R	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　**LVDO**：LVD 输出标志位
0：未检测到低电压
1：检测到低电压

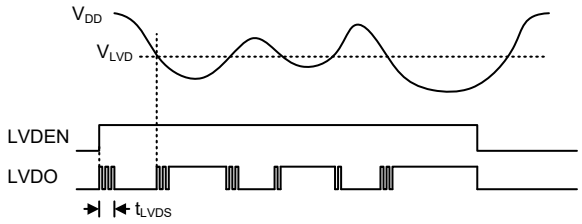
Bit 4　**LVDEN**：低电压检测控制位
0：除能
1：使能

Bit 3　**VBGEN**：Bandgap 缓冲器控制位
0：除能
1：使能
应注意，当 LVD 或 LVR 功能使能或此位置位时，Bandgap 电路使能。

Bit 2~0　**VLVD2~VLVD0**：选择 LVD 电压位
000：2.0V
001：2.2V
010：2.4V
011：2.7V
100：3.0V
101：3.3V
110：3.6V
111：4.0V

LVD 操作

通过比较电源电压 V_{DD} 与存储在 LVDC 寄存器中的预置电压值的结果，低电压检测功能工作。其设置的范围为 2.0V~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。低电压检测功能由一个自动使能的参考电压提供。当单片机处于休眠模式时，即使 LVDEN 位为高，低电压检测器除能。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。中断条件产生置位 LVDO 并延时 t_{LVD} 后，中断产生。此种情况下，若 V_{DD} 降至小于 LVD 预置电压值时，中断请求标志位 LVF 将被置位，中断产生，单片机将被从空闲模式中唤醒。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 LVF 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 INT0~INT1 引脚动作产生，而内部中断由各种内部功能，如定时器模块、时基、LVD、EEPROM 和 A/D 转换器等产生。

中断寄存器

中断控制基本上是在一定单片机条件发生时设置请求标志位，应用程序中中断使能位的设置是通过位于特殊功能数据存储器中的一系列寄存器控制的。寄存器总的分为两类。第一类是 INTC0~INTC3 寄存器，用于设置基本的中断；第二类是 MFI 寄存器，用于设置多功能中断；最后一种是 INTEG 寄存器，用于设置外部中断边沿触发类型。

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，紧接着中断号 (可选)，最后的字母“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	注释
总中断	EMI	—	—
INTn 引脚	INTnE	INTnF	n=0~1
PLT 比较器	PLTCnE	PLTCnF	n=0~1
A/D 转换器	ADE	ADF	—
时基	TBnE	TBnF	n=0~1
SIM	SIME	SIMF	—
UART	URE	URF	—
多功能中断	MFE	MFF	—
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
STMn	STMnPE	STMnPF	n=0~1
	STMnAE	STMnAF	
PTM	PTMPE	PTMPF	—
	PTMAE	PTMAF	

中断寄存器位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	INT1F	INT0F	PLTC0F	INT1E	INT0E	PLTC0E	EMI
INTC1	DEF	ADF	LVF	SIMF	DEE	ADE	LVE	SIME
INTC2	STM0AF	STM0PF	PTMAF	PTMPF	STM0AE	STM0PE	PTMAE	PTMPE
INTC3	MFF	PLTC1F	TB1F	TB0F	MFE	PLTC1E	TB1E	TB0E
MFI	—	URF	STM1AF	STM1PF	—	URE	STM1AE	STM1PE

中断寄存器列表

- **INTEG 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
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 **INT1S1~INT1S0**：INT1 脚中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

Bit 1~0 **INT0S1~INT0S0**：INT0 脚中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

- **INTC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	INT1F	INT0F	PLTC0F	INT1E	INT0E	PLTC0E	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 **INT1F**：INT1 中断请求标志位
0：无请求
1：中断请求

Bit 5 **INT0F**：INT0 中断请求标志位
0：无请求
1：中断请求

Bit 4 **PLTC0F**：PLT 比较器 0 中断请求标志位
0：无请求
1：中断请求

Bit 3 **INT1E**：INT1 中断控制位
0：除能
1：使能

Bit 2 **INT0E**：INT0 中断控制位
0：除能
1：使能

Bit 1 **PLTC0E**：PLT 比较器 0 中断控制位
0：除能
1：使能

Bit 0 **EMI**：总中断控制位
0：除能
1：使能

- **INTC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	DEF	ADF	LVF	SIMF	DEE	ADE	LVE	SIME
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 **DEF**：数据 EEPROM 中断请求标志位
0：无请求
1：中断请求

Bit 6 **ADF**：A/D 转换器中断请求标志位
0：无请求
1：中断请求

Bit 5 **LVF**：LVD 中断请求标志位
0：无请求
1：中断请求

Bit 4 **SIMF**：SIM 中断请求标志位
0：无请求
1：中断请求

Bit 3 **DEE**：数据 EEPROM 中断控制位
0：除能
1：使能

Bit 2 **ADE**：A/D 转换器中断控制位
0：除能
1：使能

Bit 1 **LVE**：LVD 中断控制位
0：除能
1：使能

Bit 0 **SIME**：SIM 中断控制位
0：除能
1：使能

- **INTC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	STM0AF	STM0PF	PTMAF	PTMPF	STM0AE	STM0PE	PTMAE	PTMPE
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 **STM0AF**：STM0 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 6 **STM0PF**：STM0 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求

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

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

Bit 3 **STM0AE**：STM0 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 2 **STM0PE**：STM0 比较器 P 匹配中断控制位
0：除能
1：使能

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

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

- **INTC3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MFF	PLTC1F	TB1F	TB0F	MFE	PLTC1E	TB1E	TB0E
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 **MFF**：多功能中断请求标志位
0：无请求
1：中断请求

Bit 6 **PLTC1F**：PLT 比较器 1 中断请求标志位
0：无请求
1：中断请求

Bit 5 **TB1F**：时基 1 中断请求标志位
0：无请求
1：中断请求

Bit 4 **TB0F**：时基 0 中断请求标志位
0：无请求
1：中断请求

Bit 3 **MFE**：多功能中断控制位
0：除能
1：使能

Bit 2 **PLTC1E**：PLT 比较器 1 中断控制位
0：除能
1：使能

Bit 1 **TB1E**：时基 1 中断控制位
0：除能
1：使能

Bit 0 **TB0E**：时基 0 中断控制位
0：除能
1：使能

- **MFI 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	URF	STM1AF	STM1PF	—	URE	STM1AE	STM1PE
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　**URF**：UART 中断请求标志位
0：无请求
1：中断请求

Bit 5　**STM1AF**：STM1 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4　**STM1PF**：STM1 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求

Bit 3　未定义，读为“0”

Bit 2　**URE**：UART 中断控制位
0：除能
1：使能

Bit 1　**STM1AE**：STM1 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0　**STM1PE**：STM1 比较器 P 匹配中断控制位
0：除能
1：使能

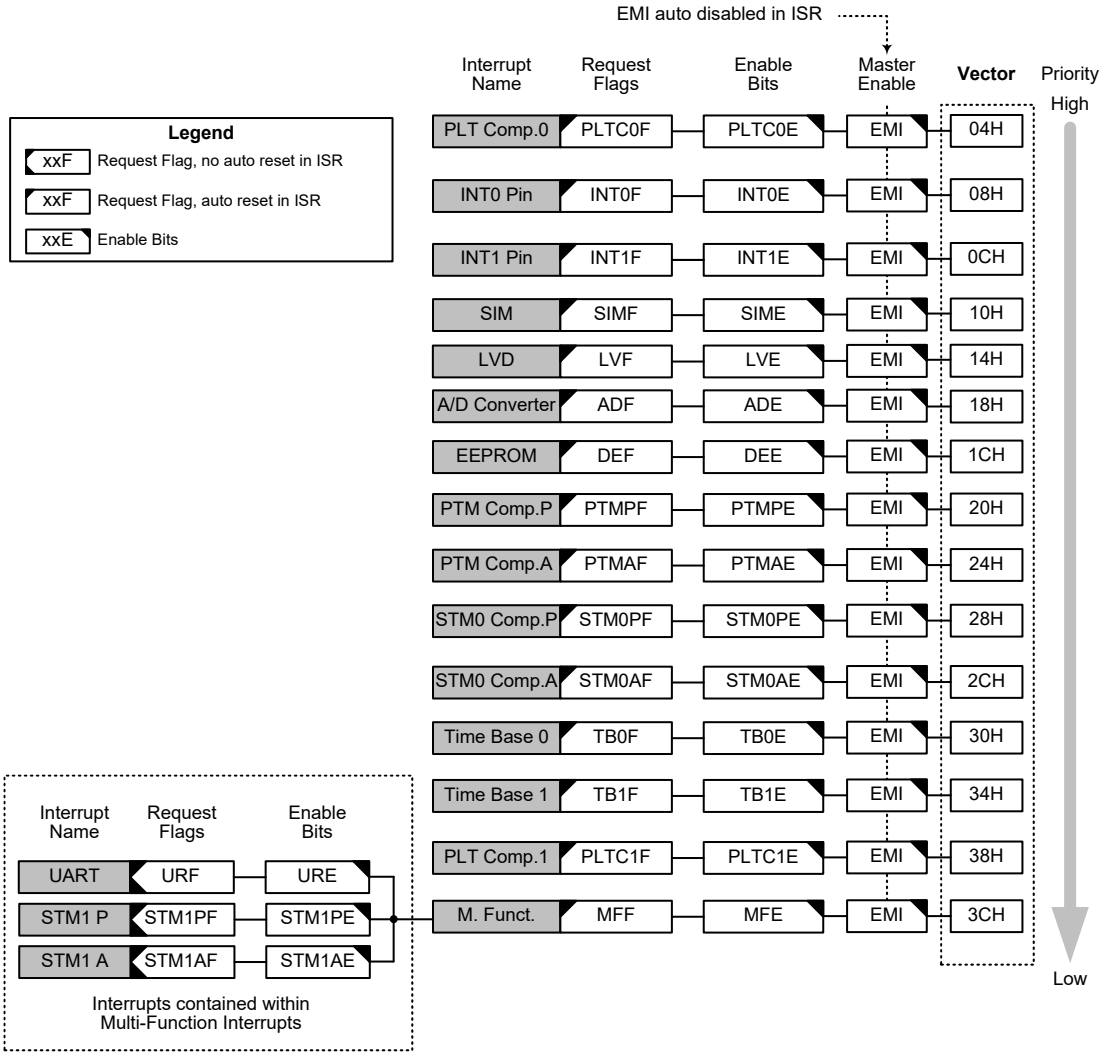
中断操作

若中断事件条件产生，如一个 TM 比较器 P、比较器 A 匹配或 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。中断向量处通常为跳转指令，以跳转到相应的中断服务程序。中断服务程序必须以“RETI”指令返回至主程序，以继续执行原来的程序。

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

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



中断结构

外部中断

通过 INT0~INT1 引脚上的信号变化可控制外部中断。当触发沿选择位设置好触发类型，INT0~INT1 引脚的状态发生变化，外部中断请求标志 INT0F~INT1F 被置位时外部中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和相应中断使能位 INT0E~INT1E 需先被置位。此外，必须使用 INTEG 寄存器使能外部中断功能并选择触发沿类型。外部中断引脚和普通 I/O 口共用，如果相应寄存器中的中断使能位被置位，并且通过引脚共用寄存器选择外部中断脚，此引脚将被作为外部中断脚使用。此时该引脚必须通过设置控制寄存器，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，中断请求标志位 INT0F~INT1F 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，其上拉电阻选项仍保持有效。

寄存器 INTEG 被用来选择有效的边沿类型，来触发外部中断。可以选择上升沿还是下降沿或双沿触发都产生外部中断。注意 INTEG 也可以用来除能外部中断功能。

PLT 比较器中断

PLT 比较器中断由电源线数据收发器电路的内部比较器控制。当 PLT 比较器 n 输出位状态改变，PLT 比较器 n 中断请求标志 PLTCnF 被置位，PLT 比较器 n 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 PLT 比较器 n 中断使能位 PLTCnE 需先被置位。当中断使能，堆栈未满并且 PLT 比较器 n 输入产生一个比较器输出位变化时，将调用 PLT 比较器 n 中断向量子程序。当响应中断服务子程序时，PLT 比较器 n 中断请求标志位 PLTCnF 会自动复位且 EMI 位会被清零以除能其它中断。

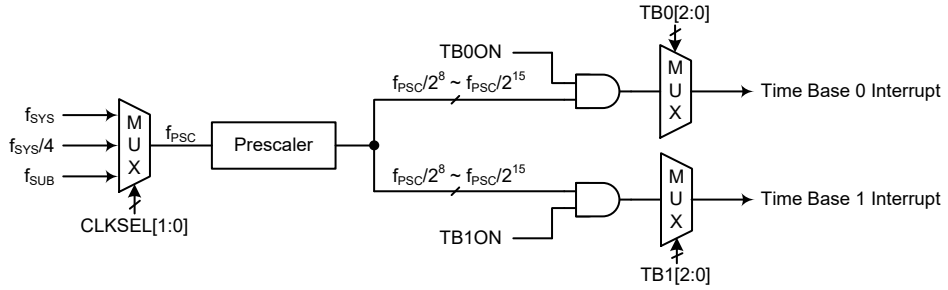
A/D 转换器中断

当 A/D 转换器中断请求标志 ADF 被置位，即 A/D 转换过程完成时，中断请求发生。当总中断使能位 EMI 和 A/D 中断使能位 ADE 被置位，允许程序跳转到各自的中断向量地址。当中断使能，堆栈未满且 A/D 转换动作结束时，将调用 A/D 转换器中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 ADF 会自动清零。EMI 位也会被清零以除能其它中断。

时基中断

时基中断提供一个固定周期的中断信号，由各自的定时器功能产生溢出信号控制。当各自的中断请求标志 TB0F~TB1F 被置位时，中断请求发生。当总中断使能位 EMI 和时基使能位 TB0E~TB1E 被置位，允许程序跳转到各自的中断向量地址。当中断使能，堆栈未满且时基溢出时，将调用它们各自的中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 TB0F~TB1F 会自动复位且 EMI 位会被清零以除能其它中断。

时基中断的目的是提供一个固定周期的中断信号，时钟源来自时钟源 f_{PSC}。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 TB0C~TB1C 寄存器相关位获取合适的分频值以提供更长的时基中断周期。相应的控制时基中断周期的时钟源可通过 PSCR 寄存器的 CLKSEL1~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**：预分频时钟源选择
00：f_{SYS}
01：$f_{SYS}/4$
1x：f_{SUB}

- **TBnC 寄存器 (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	TBnON	—	—	—	—	TBn2	TBn1	TBn0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7　**TBnON**：时基 n 控制位
0：除能
1：使能

Bit 6~3　未定义，读为“0”

Bit 2~0　**TBn2~TBn0**：选择时基 n 溢出周期
000：$2^8/f_{PSC}$
001：$2^9/f_{PSC}$
010：$2^{10}/f_{PSC}$
011：$2^{11}/f_{PSC}$
100：$2^{12}/f_{PSC}$
101：$2^{13}/f_{PSC}$
110：$2^{14}/f_{PSC}$
111：$2^{15}/f_{PSC}$

多功能中断

该单片机中有一种多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即 STM1 中断和 UART 中断。

当多功能中断请求标志 MFF 被置位，多功能中断请求产生。当中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，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。

串行接口模块中断

串行接口模块中断，即 SIM 中断。当一个字节数据已由 SIM 接口接收或发送完，或 I^2C 从机地址匹配，或 I^2C 超时，中断请求标志 SIMF 被置位，SIM 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和串行接口中断使能位 SIME 需先被置位。当中断使能，堆栈未满且以上任一种情况发生时，可跳转至相关多功能中断向量子程序中执行。当响应中断服务子程序时，串行接口中断标志位 SIMF 会自动复位且 EMI 将被自动清零以除能其它中断。

UART 中断

UART 中断属于多功能中断，由几种 UART 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、地址检测和 RX 引脚唤醒，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、UART 中断使能位 URE 和多功能使能位 MFE 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，EMI 位会被清零以除能其它中断，多功能中断请求标志位也将自动清零。而 URF 标志位无法自动清除，需通过应用程序清除。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，详情请参考 UART 章节。

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 LVF 被置位，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 LVE 需先被置位。当中断使能，堆栈未满且低电压条件发生时，将调用 LVD 中断向量子程序。当低电压中断响应，LVD 中断请求标志 LVF 自动清除，EMI 将被自动清零以除能其它中断。

EEPROM 中断

当写周期结束，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 写周期结束时，将调用 EEPROM 中断向量子程序。当 EEPROM 中断响应，相应中断请求标志位 DEF 会自动清零且 EMI 位也会被清零以除能其它中断。

TM 中断

标准型和周期型 TM 各有两个中断，分别来自比较器 P 和比较器 A 匹配。STM1 中断属于多功能中断，而 STM0 和 PTM 中断源有自己独立的向量。所有 TM 均带有两个中断请求标志位及两个使能位。当 TM 比较器 P、A 匹配情况发生时，相应 TM 中断请求标志被置位，TM 中断请求产生。

对于 STM1，若要程序跳转到相应中断向量地址，总中断控制位 EMI，STM1 中断使能位 STM1E 和多功能使能位 MFE 需先被置位。当中断使能，堆栈未满且 STM1 比较器匹配情况发生时，可跳转至相关中断向量子程序中执行。当 STM1 中断响应，仅 MFF 标志位会被自动清零，STM1 中断请求标志位 STM1F 需通过应用程序手动清零。

对于 STM0 或 PTM，若要程序跳转到相应中断向量地址，总中断控制位 EMI 和相应的中断使能位 STM0E 或 PTME 需先被置位。当中断使能，堆栈未满且 STM0 或 PTM 的比较器 P 或比较器 A 匹配情况发生时，可调用相应的 TM 中断向量子程序。当 STM0 或 PTM 中断响应，TM 中断标志位 STM0F 或 PTMF 自动清除，EMI 将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，低电压改变都可能导致其相应的中断标志被置位，由此产生中断，因此必须注意避免伪唤醒情况的发生。若中断唤醒功能被除能，单片机进入休眠或空闲模式前相应中断请求标志应被置起。中断唤醒功能不受中断使能位的影响。

编程注意事项

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

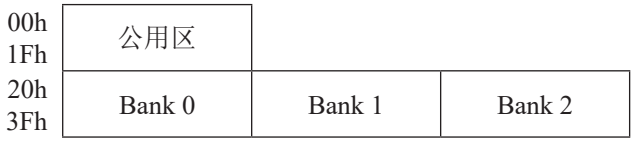
多功能中断中所含中断相应程序执行时，多功能中断请求标志 MFF 可以自动清零，但各自的请求标志需在应用程序中手动清除。

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

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

当进入中断服务程序，系统仅将程序计数器的内容压入堆栈，如果中断服务程序会改变状态寄存器或其它的寄存器的内容而破坏控制流程，应事先将这些数据保存起来。若从中断子程序中返回可执行 RET 或 RETI 指令。除了能返回至主程序外，RETI 指令还能自动设置 EMI 位为高，允许进一步中断。RET 指令只能返回至主程序，清除 EMI 位，除能进一步中断。

存储器映射

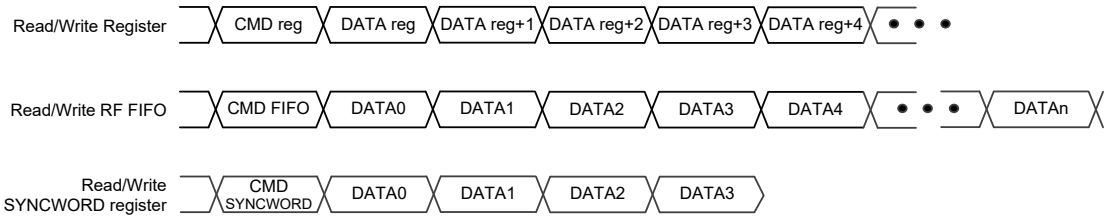


公用区：包含 32 字节空间。无论存储区指针为何值，访问地址 00h~1Fh 都意味着访问公用区。

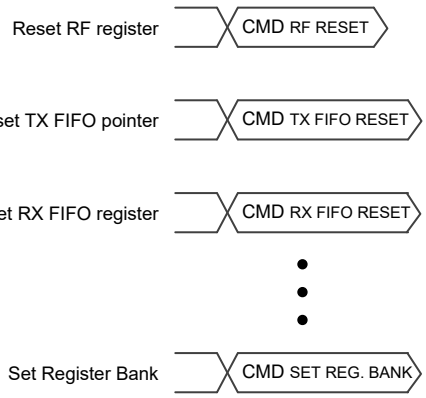
Bank 0~2：每个 Bank 包含 32 字节空间。通过存储区指针选择 Bank。

存储区指针，即 BANK[1:0]，定义在公用区，可通过设置寄存器存储区命令直接设置，且可通过控制寄存器命令进行读 / 写。

控制寄存器访问



Strobe 命令后加 n 字节数据 (CmdD)



仅 Strobe 命令 (CmdO)

SFR 映射和位定义

公用区控制寄存器

上电复位后，所有控制寄存器被设置为初始值。软件复位后，除了位于 RC1、IO1、IO2 和 IO3 寄存器的 FSYCK_EN、FSYCK_DIV[1:0]、PWRON、GIO1S[2:0]、GIO2S[2:0]、PADDS[1:0]、GIO4S[3:0]、GIOPU[4:1]、SPIPU、SDO_TEN 位之外，其它控制寄存器也将恢复至初始值，而前述控制位在软件复位后保持不变。

地址	名称	位							
		7	6	5	4	3	2	1	0
00h	CFG1	—	AGC_EN	RXCON_EN	DIR_EN	—	—	BANK[1:0]	
01h	RC1	PWRON	FSYCK_RDY	XCLK_RDY	XCLK_EN	FSYCK_DIV[1:0]		FSYCK_EN	RST_LL
02h	IRQ1	RXTO	RXFFOW	—	—	RXDETS[1:0]		IRQCPOR	IRQPOR
03h	IRQ2	ARKTFIE	ATRCTIE	FIFOLTIE	RXERRIE	RXDETIE	CALCMPIE	RXCMPIE	TXCMPIE
04h	IRQ3	ARKTFIF	ATRCTIF	FIFOLTIF	RXERRIF	RXDETIF	CALCMPIF	RXCMPIF	TXCMPIF
06h	IO1	PADDS[1:0]		GIO2S[2:0]			GIO1S[2:0]		
07h	IO2	GIO4S[3:0]				D3	D2	D1	D0
08h	IO3	SDO_TEN	SPIPU	—	GIOPU[4:1]				—
09h	FIFO1	—	—	TXFFSA[5:0]					
0Ah	FIFO2	—	—	—	RXPL2F_EN	FFINF_EN	FFMG_EN	FFMG[1:0]	
0Bh	PKT1	TXPMLEN[7:0]							
0Ch	PKT2	PID[1:0]		TRAILER_EN	WHTFMT	SYNCLEN[1:0]		RXPMLEN[1:0]	
0Dh	PKT3	MCH_EN	FEC_EN	CRC_EN	CRCFMT	PLLEN_EN	PLHAC_EN	PLHLEN	PLH_EN
0Eh	PKT4	WHT_EN	WHTSD[6:0]						
0Fh	PKT5	TXDLEN[7:0]							
10h	PKT6	RXDLEN[7:0]							
11h	PKT7	RXPID[1:0]		DLY_RXS[2:0]			DLY_TXS[2:0]		
12h	PKT8	—		PLHA[5:0]					
13h	PKT9	PLHEA[7:0]							
14h	MOD1	DTR[7:0]							

地址	名称	位							
		7	6	5	4	3	2	1	0
15h	MOD2	RXIFOS[11:8]				DITHER[1:0]		—	DTR[8]
16h	MOD3	RXIFOS[7:0]							
17h	DM1	—	—	MDIV[5:0]					
18h	DM2	PREAMBLE_CFO_EN1	PREAMBLE_CFO_EN0	SDR[5:0]					
19h	DM3	CSF_SW_EN	FD_MOD[6:0]						
1Ah	DM4	THOLD[3:0]				CFO_DSEL	—	PH_DIFF_MOD	PRE_CSF_EN
1Bh	DM5	FD_HOLD[7:0]							
1Eh	DM8	M_RATIO[7:0]							

注：地址 05h，1Ch，1Dh 和 1Fh 未列于此表格，是预留给将来使用。建议不要通过任何方式修改这几个地址的初始值。

下面寄存器说明表格里的复位值指的是 Strobe 命令软件复位后的结果。

- **CFG1：配置控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	—	AGC_EN	RXCON_EN	DIR_EN	—	—	BANK[1:0]	
R/W	—	R/W	R/W	R/W	—	—	R/W	
Reset	0	0	0	0	0	0	0	0

Bit 7　保留，必须设为“0”

Bit 6　**AGC_EN**：AGC 使能
0：除能
1：使能

Bit 5　**RXCON_EN**：RX 连续模式使能
0：除能
1：使能
设置此位只影响正常 RX 模式以及无 ARK 功能的 ATR RX 模式。

Bit 4　**DIR_EN**：Direct 模式使能
0：TX/RX 数据来自数据包处理硬件
1：TX/RX 数据直接来自 / 发至外部 MCU

Bit 3~2　保留，必须设为“00”

Bit 1~0　**BANK[1:0]**：控制寄存器 Bank 选择
00：Bank 0
01：Bank 1
10：Bank 2
11：保留
此选择可通过设置寄存器存储区命令和控制寄存器命令设置。

- **RC1：复位 / 时钟控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	PWRON	FSYCK_RDY	XCLK_RDY	XCLK_EN	FSYCK_DIV[1:0]		FSYCK_EN	RST_LL
R/W	R/W	R	R	R/W	R/W		R/W	R/W
POR	1	—	—	—	0	0	0	—
Reset	—	0	0	1	—	—	—	0

Bit 7 **PWRON**：3.3V 上电标志位
此位仅在上电复位后被设置为“1”，它不受 Strobe 命令软件复位影响。此标志位被置高后需通过软件清零。软件可先检查此标志位状态再决定在 Light Sleep 模式时是否进行自动校准。

Bit 6 **FSYCK_RDY**：FSYCK 时钟就绪标志位 (只读)
0：未就绪
1：就绪
此位用于指示 FSYCK 时钟是否就绪。当 FSYCK_EN=0、发生上电复位或接收到 Deep Sleep 或者 Idle 命令时，此位都会自动清零。

Bit 5 **XCLK_RDY**：XCLK 时钟就绪标志位 (只读)
0：未就绪
1：就绪
此位用于指示 XCLK 去抖计数器是否记满、XCLK 时钟是否就绪。当离开 Deep Sleep 状态时，需要一段时间后此标志位才被置高。当 XCLK_EN=0、RST_LL=1、发生上电复位或接收到软件复位命令、Deep Sleep 命令或者 Idle 命令时，此标志位都会自动清零。

Bit 4 **XCLK_EN**：XCLK 时钟使能
0：除能
1：使能
此位置高将使能 XCLK 至基带模块的路径。若有需要可将此位清零以减少功耗。当写数据到 FIFO 时，XCLK 时钟必须使能。

Bit 3~2 **FSYCK_DIV[1:0]**：FSYCK 时钟 (XCLK 分频) 选择
00：1/1 XCLK
01：1/2 XCLK
10：1/4 XCLK
11：1/8 XCLK

Bit 1 **FSYCK_EN**：FSYCK 时钟使能
0：除能
1：使能

Bit 0 **RST_LL**：低电压 (1.2V) 逻辑复位控制
0：不复位
1：复位

- **IRQ1：中断控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	RXTO	RXFFOW	—	—	RXDETS[1:0]		IRQCPOR	IRQPOR
R/W	R	R	—	—	R/W		R/W	R/W
Reset	0	0	0	0	1	0	0	1

Bit 7 **RXTO**：RX 溢出标志位
0：RX 溢出未发生
1：RX 溢出发生
当 RX 溢出发生时，此标志位将被硬件置高。当接收到 Light Sleep 命令、芯片进入 RX 连续模式、WOR/WOT 唤醒发生或芯片进入 ARK TX/RX 模式时，此标志位都会自动清零。

Bit 6 **RXFFOW**：RX FIFO 覆写标志位
0：RX FIFO 覆写未发生
1：RX FIFO 覆写发生
当 RX FIFO 覆写情况发生时，此标志位将被硬件置高。当接收到 RX FIFO 复位命令或 RX 命令时，此标志位都会自动清零。

Bit 5~4 保留，必须设为“00”

Bit 3~2 **RXDETS[1:0]**：RX 检测选择
00：检测载波 (Carry)
01：检测前导码 (Preamble)
10/11：检测同步码 (SYNCWORD)

Bit 1 **IRQCPOR**：IRQ 标志位清零极性选择
0：写 0 时对应 IRQ 标志位清零
1：写 1 时对应 IRQ 标志位清零

Bit 0 **IRQPOR**：IRQ 信号极性选择
0：低有效
1：高有效
当 IRQ3 寄存器里的 IRQ 标志位置高且对应的 IRQ 功能使能时，IRQ 信号的有效电平由此位决定。

- **IRQ2：中断控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	ARKTFIE	ATRCTIE	FIFOLTIE	RXERRIE	RXDETIE	CALCMPIE	RXCMPIE	TXCMPIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 **ARKTFIE**：ARK TX 失败 IRQ 使能
0：除能
1：使能

Bit 6 **ATRCTIE**：ATR 周期定时器 IRQ 使能
0：除能
1：使能

Bit 5 **FIFOLTIE**：FIFO 低阈值 IRQ 使能
0：除能
1：使能

Bit 4 **RXERRIE**：RX 错误 IRQ 使能
0：除能
1：使能

Bit 3 **RXDETIE**：RX 事件检测 IRQ 使能
0：除能
1：使能

Bit 2 **CALCMPIE**：校准完成 IRQ 使能
0：除能
1：使能

Bit 1 **RXCMPIE**：RX 完成 IRQ 使能
0：除能
1：使能

Bit 0 **TXCMPIE**：TX 完成 IRQ 使能
0：除能
1：使能

- **IRQ3：中断控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	ARKTFIF	ATRCTIF	FIFOLTIF	RXERRIF	RXDETIF	CALCMPIF	RXCMPIF	TXCMPIF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

当此寄存器中的各个标志位置高时，产生对应的中断请求。这些标志位通过写 0 还是写 1 清零由 IRQCPOR 位决定。

Bit 7 **ARKTFIF**：ARK TX 失败 IRQ 标志位
0：无请求
1：中断请求

Bit 6 **ATRCTIF**：ATR 周期定时器 IRQ 标志位
0：无请求
1：中断请求
当 ATRCT 定时器计满时，此标志位将置高。

Bit 5 **FIFOLTIF**：FIFO 低阈值 IRQ 标志位
0：无请求
1：中断请求
在 Burst TX 模式下，若此位置 1 则表示 TX FIFO 里数据长度小于 FFMG 设置的阈值且还有待写入 FIFO 的 TX 数据。在 Burst RX 模式下，若此位置 1 则表示 RX FIFO 剩余的空间小于 FFMG 设置的阈值且待接收的 RX 数据长度大于 FFMG 设置的阈值。

Bit 4 **RXERRIF**：RX 错误 IRQ 标志位
0：无请求
1：中断请求
所谓 RX 错误情况包含 RX 失败、CRC 失败 (CRC_EN=1) 或 RX FIFO 覆写。

Bit 3 **RXDETIF**：RX 事件检测 IRQ 标志位
0：无请求
1：中断请求
RX 事件包括载波、前导码和同步码，实际触发中断源取决于 RXDETS[1:0] 设置。

Bit 2 **CALCMPIF**：校准完成 IRQ 标志位
0：无请求
1：中断请求
当 ACAL_EN=0 时，LIRC 校准可由自己的使能位使能，当校准完成后会触发中断请求。当 ACAL_EN=1 时，VCO 和 RC 校准都使能，两者都完成后会触发中断请求。

Bit 1 **RXCMPIF**：RX 完成 IRQ 标志位
0：无请求
1：中断请求
当 RX 操作完成且无错误发生，此标志位将被硬件置高。

Bit 0 **TXCMPIF**：TX 完成 IRQ 标志位
0：无请求
1：中断请求

- **IO1：I/O 控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	PADDS[1:0]		GIO2S[2:0]			GIO1S[2:0]		
R/W	R/W		R/W			R/W		
POR	0	1	0	0	0	0	0	0

Bit 7~6 **PADDS[1:0]**：PAD 驱动强度选择 (仅通过 POR 复位)
00：0.5mA
01：1mA
10：5mA
11：10mA

Bit 5~3 **GIO2S[2:0]**：GIO2 引脚功能选择 (仅通过 POR 复位)
000/111：无功能，输入
001：SDO，4 线 SPI 数据，输出
010：TRXD，Direct 模式 TXD/RXD，输入 / 输出
011：TXD，Direct 模式 TXD，输入
100：RXD，Direct 模式 RXD，输出
101：IRQ，中断请求，输出
110：ROSCi，ATR 时钟外部输入

Bit 2~0 **GIO1S[2:0]**：GIO1 引脚功能选择 (仅通过 POR 复位)
000/111：无功能，输入
001：SDO，4 线 SPI 数据，输出
010：TRXD，Direct 模式 TXD/RXD，输入 / 输出
011：TXD，Direct 模式 TXD，输入
100：RXD，Direct 模式 RXD，输出
101：IRQ，中断请求，输出
110：ROSCi，ATR 时钟外部输入

- **IO2：I/O 控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	GIO4S[3:0]				D3	D2	D1	D0
R/W	R/W				R/W			
POR	0	0	0	0	0	0	0	0

Bit 7~4 **GIO4S[3:0]**：GIO4 引脚功能选择 (仅通过 POR 复位)
0000/0111/1111：无功能，输入
0001：SDO，4 线 SPI 数据，输出
0010：TRXD，Direct 模式 TXD/RXD，输入 / 输出
0011：TXD，Direct 模式 TXD，输入
0100：RXD，Direct 模式 RXD，输出
0101：IRQ，中断请求，输出
0110：ROSCi，ATR 时钟外部输入
1000：TBCLK，TX 位 (数据) 时钟，输出
1001：RBCLK，RX 位 (还原) 时钟，输出
1010：FSYCK，即 XCLK 1/1，1/2，1/4，1/8 输出
1011：LIRCCLK，内部 LIRC 去抖时钟，输出
1100：EPA_EN，外部功率放大器使能，输出
1101：ELAN_EN，外部 LNA 使能，输出
1110：TRBCLK，TX 模式的 TBCLK 或 RX 模式的 RBCLK，输出

Bit 3~0 **D3~D0**：保留，必须固定为 “0000”，“0111” 或 “1111”

- **IO3：I/O 控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	SDO_TEN	SPIPU	—	GIOPU[4:1]				—
R/W	R/W	R/W	—	R/W				—
POR	0	1	1	1	1	1	1	1

Bit 7 **SDO_TEN**：SDO 三态使能 (仅通过 POR 复位)
0：除能
1：使能

Bit 6 **SPIPU**：3 线 SPI 上拉使能 (仅通过 POR 复位)
0：除能
1：使能
此位置 1 仅控制 CSN、SCK 和 SDIO 引脚的上拉功能。注意，4 线 SPI 的 SDO 引脚上拉功能需通过 GIOPU[4:1] 对应位设置。

Bit 5 保留，必须设为“1”

Bit 4~1 **GIOPU[4:1]**：GIO 引脚功能上拉使能控制 (仅通过 POR 复位)
这些位分别控制 GIO4~GIO1 引脚的上拉功能。

Bit 0 保留，必须设为“1”

- **FIFO1：FIFO 控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	—	—	TXFFSA[5:0]					
R/W	—	—	R/W					
Reset	0	0	0	0	0	0	0	0

Bit 7~6 保留，必须设为“00”

Bit 5~0 **TXFFSA[5:0]**：TX FIFO 起始地址，用于 Block FIFO 模式

- **FIFO2：FIFO 控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	RXPL2F_EN	FFINF_EN	FFMG_EN	FFMG[1:0]	
R/W	—	—	—	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	1

Bit 7~5 保留，必须设为“000”

Bit 4 **RXPL2F_EN**：RX 有效载荷 (Payload) 长度字节载入 FIFO 使能
0：除能
1：使能
当此位置高，指示有效载荷长度的字节将被加入数据包并载入 RX FIFO。在 RX 连续模式下 (RXCON_EN=1)，RX FIFO 将支持多笔有效载荷，此位也必须置高。

Bit 3 **FFINF_EN**：FIFO 无限制长度模式使能
0：除能
1：使能

Bit 2 **FFMG_EN**：FIFO 长度边界检测使能
0：除能
1：使能

Bit 1~0 **FFMG[1:0]**：FIFO 长度边界选择
TX FIFO 剩余数据长度阈值：
00：4 字节
01：8 字节
10：16 字节
11：32 字节

RX FIFO 剩余空间长度阈值：
00：4 字节
01：8 字节
10：16 字节
11：32 字节
当 FFMG_EN 位置高使能 FIFO 长度边界检测功能，且已通过这些位选择所需检测的FIFO长度边界后，当所选的情况发生时，FIFOLTIF 标志位将被置高。此时，若对应的中断功能已使能，将产生中断。

- **PKT1：数据包控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	TXPMLEN[7:0]							
R/W	R/W							
Reset	0	0	0	0	0	0	0	1

Bit 7~0 **TXPMLEN[7:0]**：TX 前导码长度
发送前导码长度 = (TXPMLEN[7:0]+1) 字节

- **PKT2：数据包控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	PID[1:0]		TRAILER_EN	WHTFMT	SYNCLEN[1:0]		RXPMLEN[1:0]	
R/W	R/W		R/W	R/W	R/W		R/W	
Reset	0	0	1	0	0	1	1	0

Bit 7~6 **PID[1:0]**：TX 数据包 ID
当 PLH_EN 位置高使能头码选项时，此 ID 会被放入有效载荷头码字段的最高两位。

Bit 5 **TRAILER_EN**：连接码字段使能
0：除能
1：使能

Bit 4 **WHTFMT**：数据白化格式选择
0：$P(X)=X^7+X^6+X^5+X^4+1$
1：$P(X)=X^7+X^4+1$ (PN7)

Bit 3~2 **SYNCLEN[1:0]**：TX/RX 模式同步码长度选择
00：保留
01：4 字节
10：6 字节
11：8 字节

Bit 1~0 **RXPMLEN[1:0]**：RX 前导码检测长度选择
00：0 字节 – 无前导码检测
01：1 字节
10：2 字节
11：4 字节

- **PKT3：数据包控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	MCH_EN	FEC_EN	CRC_EN	CRCFMT	PLLEN_EN	PLHAC_EN	PLHLEN	PLH_EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 7 **MCH_EN**：曼彻斯特编码使能
0：除能
1：使能

Bit 6 **FEC_EN**：FEC 使能
0：除能
1：使能

Bit 5 **CRC_EN**：CRC 域使能
0：除能
1：使能

Bit 4 **CRCFMT**：CRC 格式选择
0：CCITT-16-CRC $G(X)=X^{16}+X^{12}+X^5+1$
1：IBC-16-CRC $G(X)=X^{16}+X^{15}+X^2+1$

Bit 3 **PLLEN_EN**：有效载荷长度域使能
0：除能
1：使能

Bit 2 **PLHAC_EN**：有效载荷头码地址校准使能控制
0：除能，PKT8 寄存器中的 PLHA[5:0] 字段可由用户自定义作为标志位使用
1：使能，TX 和 RX 设备的 PLHA[5:0] 字段必须包含相同地址，否则数据包将被视为无效数据包

Bit 1 **PLHLEN**：有效载荷头码长度选择
0：1 字节
1：2 字节

Bit 0 **PLH_EN**：有效载荷头码域使能
0：除能
1：使能

- **PKT4：数据包控制寄存器 4**

Bit	7	6	5	4	3	2	1	0
Name	WHT_EN	WHTSD[6:0]						
R/W	R/W	R/W						
Reset	0	0	1	1	0	1	1	0

Bit 7 **WHT_EN**：数据白化使能
0：除能
1：使能

Bit 6~0 **WHTSD[6:0]**：数据白化种子

- **PKT5：数据包控制寄存器 5**

Bit	7	6	5	4	3	2	1	0
Name	TXDLEN[7:0]							
R/W	R/W							
Reset	0	1	0	0	0	0	0	0

Bit 7~0 **TXDLEN[7:0]**：TX 数据长度 (单位：字节；仅用于 Burst 模式)

- **PKT6：数据包控制寄存器 6**

Bit	7	6	5	4	3	2	1	0
Name	RXDLEN[7:0]							
R/W	R/W							
Reset	0	1	0	0	0	0	0	0

Bit 7~0 **RXDLEN[7:0]**：RX 数据长度 (单位：字节；仅用于 Burst 模式)
当 PLLEN_EN 位清零时，接收的数据长度取决于此字段。当此寄存器被读取时，所读出的数值表示 RX FIFO 中的数据长度。此寄存器被读取出的默认值是 00h。

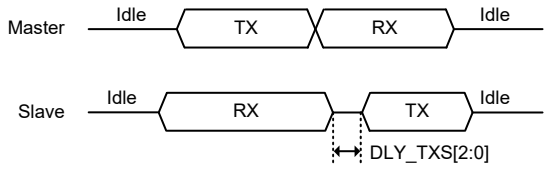
- **PKT7：数据包控制寄存器 7**

Bit	7	6	5	4	3	2	1	0
Name	RXPID[1:0]		DLY_RXS[2:0]			DLY_TXS[2:0]		
R/W	R		R/W			R/W		
Reset	0	0	1	0	0	0	0	0

Bit 7~6 **RXPID[1:0]**：接收数据包 PID (只读)

Bit 5~3 **DLY_RXS[2:0]**：RX 模块使能后稳定时间选择
000：4μs
001：8μs
010：12μs
011：16μs
100：20μs
101：32μs
110：64μs
111：100μs
这些位用于选择 RX 使能后到 RX 稳定前的等待时间。此时间应该大于 RX DCOC Turbo 模式默认延迟时间 (6μs)。

Bit 2~0 **DLY_TXS[2:0]**：进入 TX 模式前的 TX 启动 (延迟) 时间
000：0μs
001：10μs
010：20μs
011：40μs
100：60μs
101：80μs
110：100μs
111：120μs
该时间用于 ARK 模式下发送器和接收器间的时序调整。



- **PKT8：数据包控制寄存器 8**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PLHA[5:0]					
R/W	—	—	R/W					
Reset	0	0	0	0	0	0	0	0

Bit 7~6　保留，必须设为“00”

Bit 5~0　**PLHA[5:0]**：有效载荷头码地址，用于支持播送功能
RX 模式下若此地址为 0 表示不执行校验核对。
写：写数据到 TX PLHA[5:0]。读：从 RX PLHA[5:0] 读数据。

- **PKT9：数据包控制寄存器 9**

Bit	7	6	5	4	3	2	1	0
Name	PLHEA[7:0]							
R/W	R/W							
Reset	0	0	0	0	0	0	0	0

Bit 7~0　**PLHEA[7:0]**：有效载荷头码扩展地址，用于支持播送功能
RX 模式下若此地址为 0 表示不执行校验核对。

- **MOD1：调制器控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	DTR[7:0]							
R/W	R/W							
Reset	0	0	0	0	0	0	0	1

Bit 7~0　**DTR[7:0]**
DTR[8:0]：数据速率分频器，DTR[8] 位于 MOD2 寄存器。
数据速率 = f_{XTAL}/[(XODIV2+1)×32×(DTR[8:0]+1)]，其中 XODIV2=0，这里的数据速率表示 TBCLK。注意，DTR[8:0] 只能是奇数。

- **MOD2：调制器控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	RXIFOS[11:8]				DITHER[1:0]		—	DTR[8]
R/W	R/W				R/W		—	R/W
Reset	1	0	0	1	0	0	0	0

Bit 7~4　**RXIFOS[11:8]**
RXIFOS[11:0]：RX 中频偏移，RXIFOS[7:0] 位于 MOD3 寄存器。
要先写 RXIFOS[11:8] 再写 RXIFOS[7:0] 才可完全更新 RXIFOS[11:0]。
RXIFOS[11:0]=floor{f_{IF}/[f_{XTAL}/(XODIV2+1)]×2^{17}}，XODIV2=0

Bit 3~2　**DITHER[1:0]**：高频振动值

Bit 1　保留，必须设为“0”

Bit 0　**DTR[8]**
DTR[8:0]：数据速率分频，DTR[7:0] 位于 MOD1 寄存器。
数据速率 = f_{XTAL}/[(XODIV2+1)×32×(DTR[8:0] +1)]，其中 XODIV2=0，这里的数据速率表示 TBCLK。注意，DTR[8:0] 只能是奇数。

- **MOD3：调制器控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	RXIFOS[7:0]							
R/W	R/W							
Reset	1	0	0	1	1	0	1	0

Bit 7~0 **RXIFOS[7:0]**
RXIFOS[11:0]：RX 中频偏移，RXIFOS[11:8] 位于 MOD2 寄存器
要先写 RXIFOS[11:8] 再写 RXIFOS[7:0] 才可完全更新 RXIFOS[11:0]。
RXIFOS[11:0]=floor{(f_{IF}/[f_{XTAL}/(XODIV2+1)]×2^{17}}，XODIV2=0

- **DM1：解调器控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	—	—	MDIV[5:0]					
R/W	—	—	R/W					
Reset	0	0	0	0	0	0	1	1

Bit 7~6 保留，必须设为“00”

Bit 5~0 **MDIV[5:0]**：解调器工作时钟分频
DMCLK=ADCLK/(MDIV[5:0]+1)

- **DM2：解调器控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	PREAMBLE_CFO_EN1	PREAMBLE_CFO_EN0	SDR[5:0]					
R/W	R/W	R/W	R/W					
Reset	0	1	0	0	0	0	0	0

Bit 7 **PREAMBLE_CFO_EN1**：前导码第二阶 CFO 校准使能
0：除能
1：使能
仅当前导码为 4 个字节时，即 RXPMLEN[1:0]=11b 时，此位才能置 1。

Bit 6 **PREAMBLE_CFO_EN0**：前导码第一阶 CFO 校准使能
0：除能
1：使能

Bit 5~0 **SDR[5:0]**：相位提取后的解调器工作时钟
SDR[5:0]+1=DMCLK/(8×DATA_RATE)，这里的 DATA_RATE 表示 RBCLK。

- **DM3：解调器控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	CSF_SW_EN	FD_MOD[6:0]						
R/W	R/W	R/W						
Reset	1	1	1	1	0	0	0	0

Bit 7 **CSF_SW_EN**：通道选择滤波器自动频宽切换使能
0：除能
1：使能

Bit 6~0 **FD_MOD[6:0]**：频率偏移调节器
FD_MOD=round ((h/(SDR[5:0]+1))×128)；h = 解调系数
SDR[5:0]+1=DMCLK/(8×DATA_RATE)

- **DM4：解调器控制寄存器 4**

Bit	7	6	5	4	3	2	1	0
Name	THOLD[3:0]				CFO_DSEL	—	PH_DIFF_MOD	PRE_CSF_EN
R/W	R/W				R/W	—	R/W	R/W
Reset	0	0	0	1	1	0	0	0

Bit 7~4 **THOLD[3:0]**：检测错误阈值
THOLD[3:2]：前导码检测错误位数
THOLD[1:0]：同步码检测错误位数

Bit 3 **CFO_DSEL**：CFO 校准域选择
0：模拟域
1：数字域

Bit 2 保留，必须设为“0”

Bit 1 **PH_DIFF_MOD**：相位差提取模式设置
0：相位提取范围 [-pi/2, pi/2]
1：相位提取范围 [-pi, pi]

Bit 0 **PRE_CSF_EN**：前导码匹配时接收滤波器带宽切换控制
0：除能
1：使能

- **DM5：解调器控制寄存器 5**

Bit	7	6	5	4	3	2	1	0
Name	FD_HOLD[7:0]							
R/W	R/W							
Reset	0	0	1	1	0	0	0	0

Bit 7~0 **FD_HOLD[7:0]**：前导码检测频率偏差阈值

- **DM8：解调器控制寄存器 8**

Bit	7	6	5	4	3	2	1	0
Name	M_RATIO[7:0]							
R/W	R/W							
Reset	0	1	0	0	0	0	0	0

Bit 7~0 **M_RATIO[7:0]**：用于 CFO 计算
$M\_RATIO=round(1/(MDIV[5:0]+1)\times 2^8)$

Bank 0 控制寄存器

上电复位后，所有控制寄存器被设置为初始值。软件复位后，除了位于 XO3 寄存器的 LIRC_EN、LIRC_OP[4:0]、LIRC_OW 和 LIRCCAL_EN 位之外，其它控制寄存器也将恢复至初始值。而前述控制位在软件复位后保持不变。

地址	名称	位							
		7	6	5	4	3	2	1	0
20h	OM	PWR_SOFT	BAND_SEL[1:0]		—	ACAL_EN	RTX_EN	RTX_SEL	SX_EN
22h	SX1	—	D_N[6:0]						
23h	SX2	D_K[7:0]							
24h	SX3	D_K[15:8]							

地址	名称	位							
		7	6	5	4	3	2	1	0
25h	SX4	—	—	—	—	D_K[19:16]			
26h	STA1	—	—	—	CD_FLAG	—	OMST[2:0]		
28h	RSSI2	—				RSSI_CTHD[3:0]			
29h	RSSI3	RSSI_NEGDB[7:0]							
2Ah	RSSI4	RSSI_SYNC_OK[7:0]							
2Bh	ATR1	ATRCLK_DIV[1:0]		ATRCLKS	ATRTU	ATRCTM	ATRM[1:0]		ATR_EN
2Ch	ATR2	ATRCYC[7:0]							
2Dh	ATR3	ATRCYC[15:8]							
2Eh	ATR4	ATRRXAP[7:0]							
2Fh	ATR5	ATRRXEP[7:0]							
30h	ATR6	ATRRXEP[15:8]							
31h	ATR7	ARKNM[3:0]				—	ATR_WDLY[1:0]		ARK_EN
32h	ATR8	ARKRXAP[7:0]							
33h	ATR9	ATRCT[7:0]							
34h	ATR10	ATRCT[15:8]							
35h	ATR11	—					ATRRXAP[10:8]		
3Ch	XO1	XSHIFT[1:0]		—		XO_TRIM[4:0]			
3Dh	XO2	—	—	—	—	XODIV2	—		
3Eh	XO3	LIRCCAL_EN	LIRC_OW	LIRC_OP[4:0]					LIRC_EN
3Fh	TX2	—			CT_PAD[3:0]				

注：地址 21h、27h 和 36h~3Bh 未列于此表格，是预留给将来使用。建议不要通过任何方式修改这些地址的初始值。

下面寄存器说明表格里的复位值指的是 Strobe 命令软件复位后的结果。

- **OM：工作模式控制寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PWR_SOFT	BAND_SEL[1:0]		—	ACAL_EN	RTX_EN	RTX_SEL	SX_EN
R/W	R/W	R/W		—	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 7 **PWR_SOFT**：RF 工作模式选择
0：RF 正常工作模式
1：RF 工程模式

Bit 6~5 **BAND_SEL[1:0]**：频段选择 (当 PWR_SOFT=0)
00：315MHz 频段
01：433MHz 频段
10：470~510MHz 频段
11：868/915MHz 频段

Bit 4 保留，必须设为“0”

Bit 3 **ACAL_EN**：自动校准使能
0：除能
1：使能
当此位置高，VCO 和 RC 校准都使能。当 VCO 和 RC 校准都完成后，此位由硬件自动清零。

Bit 2 **RTX_EN**：RX 或 TX 模式使能
0：除能
1：使能
RX 或 TX 模式由 RTX_SEL 位选择之后，此位置高将会使能所选的模式。

Bit 1 **RTX_SEL**：RX 或 TX 模式选择
0：RX 模式
1：TX 模式

Bit 0 **SX_EN**：合成器使能 (Standby 模式使能控制)
0：除能
1：使能
此位置高将使能 PFD、CP 和 VCO 功能。

- **SX1：小数 N 分频合成器控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	—	D_N[6:0]						
R/W	—	R/W						
Reset	0	0	0	1	1	0	1	1

Bit 7 保留，必须设为“0”

Bit 6~0 **D_N[6:0]**：RF 通道整数代码
D_N[6:0]=floor{f_{RF}/[f_{XTAL}/(XODIV2+1)]}
例如，默认 XO=16MHz 且 RF 频段 = 433.92MHz：
→ 433.92MHz/16MHz=27.12
→ D_N=27
→ Dec2Hex(27)=1B
→ Dec2Bin(27)=001_1011

- **SX2：小数 N 分频合成器控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	D_K[7:0]							
R/W	R/W							
Reset	1	0	0	0	0	1	0	1

Bit 7~0 **D_K[7:0]**：RF 通道小数代码低字节

- **SX3：小数 N 分频合成器控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	D_K[15:8]							
R/W	R/W							
Reset	1	1	1	0	1	0	1	1

Bit 7~0 **D_K[15:8]**：RF 通道小数代码中间字节

- **SX4：小数 N 分频合成器控制寄存器 4**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D_K[19:16]			
R/W	—	—	—	—	R/W			
Reset	0	0	0	0	0	0	0	1

Bit 7~4 保留，必须设为“0000”

Bit 3~0 **D_K[19:16]**：RF 通道小数代码高字节

$D\_K[19:0]=floor\{(f_{RF}/[f_{XTAL}/(XODIV2+1)]-D\_N[6:0])\times 2^{20}\}$

例如，默认 XO=16MHz 且 RF 频段 = 433.92MHz：

→ 433.92MHz/16MHz=27.12

→ $D\_K=0.12\times 2^{20}=125829$

→ Dec2Hex(125829)=1EB85

→ Dec2Bin(125829)=0001_1110_1011_1000_0101

- **STA1：状态控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	CD_FLAG	—	OMST[2:0]		
R/W	—	—	—	R	—	R		
Reset	0	0	0	0	0	0	0	0

Bit 7~5 保留，必须设为“000”

Bit 4 **CD_FLAG**：载波检测标志位 (只读)

当 DEMOD_EN 拉高且载波检测没问题时，此标志位将被硬件置高。这里的 DEMOD_EN 高电平是内部信号，在 Direct 模式 (DIR_EN=1) 下由内部状态机产生，或在 Burst 模式 (DIR_EN=0) 下接收到 RX 命令后产生。此标志位在 RX_EN 上升沿时被自动清零。这里说的 RX_EN 上升沿在 Direct 模式下设置 RTX_SEL=0 且 RTX_EN=1 后由内部状态机产生，或在 Burst 模式下接收到 RX 命令后产生。

Bit 3 保留，必须设为“0”

Bit 2~0 **OMST[2:0]**：工作模式状态指示 (只读)

000：Deep Sleep 模式
001：Idle 模式
010：Light Sleep 模式
011：Standby 模式
100：TX 模式
101：RX 模式
110：VCO 校准模式
111：未定义

- **RSSI2：RSSI 控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	RSSI_CTHD[3:0]			
R/W	—	—	—	—	R/W			
Reset	0	0	0	0	1	0	1	0

Bit 7~4 保留，必须设为“0000”

Bit 3~0 **RSSI_CTHD[3:0]**：载波检测 RSSI 阈值

(RSSI_CTHD[3:0]×2+1)+74 = 载波检测 RSSI 阈值

- **RSSI3：RSSI 控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	RSSI_NEGDB[7:0]							
R/W	R							
Reset	0	0	0	0	0	0	0	0

Bit 7~0 **RSSI_NEGDB[7:0]**：RSSI 值 (单位：-dB)
此值为实时测量值。

- **RSSI4：RSSI 控制寄存器 4**

Bit	7	6	5	4	3	2	1	0
Name	RSSI_SYNC_OK[7:0]							
R/W	R							
Reset	0	0	0	0	0	0	0	0

Bit 7~0 **RSSI_SYNC_OK[7:0]**：同步码检测正确时的 RSSI 快摄值

- **ATR1：自动 TX/RX 控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	ATRCLK_DIV[1:0]		ATRCLKS	ATRTU	ATRCTM	ATRM[1:0]		ATR_EN
R/W	R/W		R/W	R/W	R/W	R/W		R/W
Reset	1	1	0	0	1	0	0	0

Bit 7~6 **ATRCLK_DIV[1:0]**：ATR 时钟频率分频
00：1/1，ATRCLK=32768Hz
01：1/4，ATRCLK=8192Hz
10：1/8，ATRCLK=4096Hz
11：1/16，ATRCLK=2048Hz

Bit 5 **ATRCLKS**：ATRCLK 时钟源选择
0：来自内部 LIRC 时钟
1：来自 GIOn 引脚的外部 ROSCi 时钟输入

Bit 4 **ATRTU**：自动 TRX 单位时间选择
0：250μs
1：1ms，用于支持低数据速率应用
此位用于选择 ATR RX 有效周期 (ATRRXAP[10:0])、ATR RX 扩展周期 (ATRRXEP[15:0]) 以及 ARK RX 有效周期 (ARKRXAP[7:0]) 的单位时间。

Bit 3 **ATRCTM**：自动 TRX 定时模式选择
0：单次模式，每次发生 ATR 事务时启动 ATRCT 定时器
1：连续模式，接收到 Idle 命令时启动 ATRCT 定时器，当 ATR_EN=0 或者 ATRCTM=0 时停止 ATRCT 定时器

Bit 2~1 **ATRM[1:0]**：自动 TRX 模式选择
00：ATR WOT 模式
01：ATR WOR 模式
10/11：ATR WTM 模式

Bit 0 **ATR_EN**：自动 TRX 使能
0：除能
1：使能
当工作模式状态由 Deep Sleep/Light Sleep 模式切换到 Idle 模式时 ATR 功能启动。

- **ATR2：自动 TX/RX 控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	ATRCYC[7:0]							
R/W	R/W							
Reset	1	1	1	1	1	1	1	1

Bit 7~0 **ATRCYC[7:0]**：ATRCT 定时器界限值低字节

- **ATR3：自动 TX/RX 控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	ATRCYC[15:8]							
R/W	R/W							
Reset	0	0	0	0	1	1	1	1

Bit 7~0 **ATRCYC[15:8]**：ATRCT 定时器界限值高字节
唤醒周期 = 周期 (ATRCLK)×(ATRCYC[15:0]) + 周期 (LIRCCLK)，ATRCYC!=0。
默认大约为 2 秒。

- **ATR4：自动 TX/RX 控制寄存器 4**

Bit	7	6	5	4	3	2	1	0
Name	ATRRXAP[7:0]							
R/W	R/W							
Reset	0	0	1	0	0	1	1	1

Bit 7~0 **ATRRXAP[7:0]**：ATR RX 有效周期低字节
ATR RX 有效周期高字节 ATRRXAP[10:8] 位于 ATR11 寄存器。
有效周期 = 单位时间 × (ATRRXAP[10:0]+1)；单位时间为 250μs 或 1ms，由 ATRTU 位决定。由于默认的单位时间是 250μs，默认的 ATR RX 有效周期为 10ms。

- **ATR5：自动 TX/RX 控制寄存器 5**

Bit	7	6	5	4	3	2	1	0
Name	ATRRXEP[7:0]							
R/W	R/W							
Reset	1	0	0	0	1	1	1	1

Bit 7~0 **ATRRXEP[7:0]**：ATR RX 扩展周期低字节

- **ATR6：自动 TX/RX 控制寄存器 6**

Bit	7	6	5	4	3	2	1	0
Name	ATRRXEP[15:8]							
R/W	R/W							
Reset	0	0	0	0	0	0	0	1

Bit 7~0 **ATRRXEP[15:8]**：ATR RX 扩展周期高字节
扩展周期 = 单位时间 × (ATRRXEP[15:0]+1)；单位时间为 250μs 或 1ms，由 ATRTU 位决定。由于默认的单位时间是 250μs，默认的 ATR RX 扩展周期为 100ms。

- **ATR7：自动 TX/RX 控制寄存器 7**

Bit	7	6	5	4	3	2	1	0
Name	ARKNM[3:0]				—	ATR_WDLY[1:0]		ARK_EN
R/W	R/W				—	R/W		R/W
Reset	0	1	1	1	0	0	1	0

Bit 7~4 **ARKNM[3:0]**：ARK 重复周期次数
最大重复周期次数 = ARKNM[3:0]+1

Bit 3 保留，必须设为“0”

Bit 2~1 **ATR_WDLY[1:0]**：自动唤醒延迟时间
00：244μs
01：488μs
10：732μs
11：976μs

Bit 0 **ARK_EN**：自动重发 /ACK 使能
0：除能
1：使能

- **ATR8：自动 TX/RX 控制寄存器 8**

Bit	7	6	5	4	3	2	1	0
Name	ARKRXAP[7:0]							
R/W	R/W							
Reset	0	0	1	0	0	1	1	1

Bit 7~0 **ARKRXAP[7:0]**：ARK RX 有效周期
有效周期 = 单位时间 × (ARKRXAP[7:0]+1)；单位时间为 250μs 或 1ms，由 ATRTU 位决定。由于默认的单位时间是 250μs，默认的 ARK RX 有效周期为 10ms。

- **ATR9：自动 TX/RX 控制寄存器 9**

Bit	7	6	5	4	3	2	1	0
Name	ATRCT[7:0]							
R/W	R/W							
Reset	0	0	0	0	0	0	0	0

Bit 7~0 **ATRCT[7:0]**：ATR 周期定时器低字节

- **ATR10：自动 TX/RX 控制寄存器 10**

Bit	7	6	5	4	3	2	1	0
Name	ATRCT[15:8]							
R/W	R/W							
Reset	0	0	0	0	0	0	0	0

Bit 7~0 **ATRCT[15:8]**：ATR 周期定时器高字节
读 ATRCT[15:0] 将得到当前计数值。由于 8 位 SPI 数据长度的限制，读取 ATR9 寄存器时会快速抓取完整的 16 位数据放入读取寄存器缓冲器。用户需连续读取 ATR9 和 ATR10 寄存器 (不被间断) 才能得到正确数据。
写数据到 ATRCT[15:0] 将更新计数值。先写 ATR9 寄存器再写 ATR10 寄存器才能触发 ATRCT 写功能。此定时器的更新机制用于双向 RF 系统调整主从端时隙。

- **ATR11：自动 TX/RX 控制寄存器 11**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	ATRRXAP[10:8]		
R/W	—	—	—	—	—	R/W		
Reset	0	0	0	0	0	0	0	0

Bit 7~3　保留，必须设为“00000”

Bit 2~0　**ATRRXAP[10:8]**：ATR RX 有效周期高字节
ATR RX 有效低字节 ATRRXAP[7:0] 位于 ATR4 寄存器。
有效周期 = 单位时间 × (ATRRXAP[10:0]+1)；单位时间为 250μs 或 1ms，由 ATRTU 位决定。由于默认的单位时间是 250μs，默认的 ATR RX 有效周期为 10ms。

- **XO1：XO 控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	XSHIFT[1:0]		—	XO_TRIM[4:0]				
R/W	R/W		—	R/W				
Reset	0	0	0	1	0	0	0	0

Bit 7~6　**XSHIFT[1:0]**：电容负载粗调

Bit 5　保留，必须设为“0”

Bit 4~0　**XO_TRIM[4:0]**：晶振内部电容负载细调
默认设置 = 2.4pF，step=0.15pF。此调整值越大电容负载越大，反之同理。

XO1 寄存器的建议值如下表所示：

CLOAD	12pF	16pF	20pF
XO1	1Eh	50h	92h

- **XO2：XO 控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	XODIV2	—	—	—
R/W	—	—	—	—	R/W	—	—	—
Reset	0	0	0	0	0	0	1	1

Bit 7~4　保留，必须设为“0000”

Bit 3　**XODIV2**：XO 输出除以 2 使能控制
0：除能
1：使能
注：f_{XTAL}=16MHz，XODIV2 必须为“0”。

Bit 2~0　保留，必须设为“011”

- **XO3：XO 控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	LIRCCAL_EN	LIRC_OW	LIRC_OP[4:0]					LIRC_EN
R/W	R/W	R/W	R/W					R/W
POR	0	0	0	1	1	0	1	0

Bit 7　**LIRCCAL_EN**：LIRC 校准使能控制
0：除能
1：使能

Bit 6 **LIRC_OW**：LIRC 覆写控制
0：LIRC_OP[4:0] 来自校准引擎
1：LIRC_OP[4:0] 来自控制寄存器

Bit 5~1 **LIRC_OP[4:0]**：LIRC 调整
写数据到 LIRC_OP[4:0] 之后，此值在 LIRC_OW 位置高后生效。当从 LIRC_OP[4:0] 读取数据时，实际数据源取决于 LIRC_OW 位的设置。

Bit 0 **LIRC_EN**：LIRC 使能控制
0：除能
1：使能

- **TX2：TX 控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	CT_PAD[3:0]			
R/W	—	—	—	—	R/W			
Reset	0	0	0	0	1	0	0	0

Bit 7~4 保留，必须设为“0000”

Bit 3~0 **CT_PAD[3:0]**：TX PAD 线性功率控制

Bank 1 控制寄存器

上电复位后，所有控制寄存器被设置为初始值。软件复位后，这些控制寄存器也将恢复至初始值。

地址	名称	位							
		7	6	5	4	3	2	1	0
21h	AGC2	SAT_SEL[1:0]		—				AGC_CMP_THD[1:0]	
22h	AGC3	CDRST_THD_SEL[1:0]		ENVAVG_SEL[1:0]		—	IF_DETOK_THD[2:0]		
23h	AGC4	GAIN_SEL[3:0]				—	AGC_ST[2:0]		
24h	AGC5	—						AGC_FSEL[1:0]	
26h	AGC7	GAIN_STB[7:0]							
2Ch	FCF1	—		SFRATIO[1:0]		—			
2Dh	FCF2	FSCALE[7:0]							
2Eh	FCF3	—				FSCALE[11:8]			
2Fh	FCF4	CF_B12[7:0]							
30h	FCF5	—						CF_B12[9:8]	
31h	FCF6	CF_B13[7:0]							
32h	FCF7	—						CF_B13[9:8]	
33h	FCF8	CF_A12[7:0]							
34h	FCF9	—						CF_A12[9:8]	
35h	FCF10	CF_A13[7:0]							
36h	FCF11	—						CF_A13[9:8]	
37h	FCF12	CF_B22[7:0]							
38h	FCF13	—						CF_B22[9:8]	
39h	FCF14	CF_B23[7:0]							
3Ah	FCF15	—						CF_B23[9:8]	
3Bh	FCF16	CF_A22[7:0]							
3Ch	FCF17	—						CF_A22[9:8]	
3Dh	FCF18	CF_A23[7:0]							

地址	名称	位							
		7	6	5	4	3	2	1	0
3Eh	FCF19	—						CF_A23[9:8]	

注：地址 20h、25h、27h~2Bh 和 3Fh 未列于此表格，是预留给将来使用。建议不要通过任何方式修改这些地址的初始值。

下面寄存器说明表格里的复位值指的是 Strobe 命令软件复位后的结果。

- **AGC2：AGC 控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	SAT_SEL[1:0]		—	—	—	—	AGC_CMP_THD[1:0]	
R/W	R/W		—	—	—	—	R/W	
Reset	0	1	0	0	0	0	0	0

Bit 7~6 **SAT_SEL[1:0]**：饱和检测阈值选择
00：-6dBFS
01：-8dBFS
10：-10dBFS
11：-12dBFS
注：“FS”表示 ADC 输出满额。

Bit 5~2 保留，必须设为“0000”

Bit 1~0 **AGC_CMP_THD[1:0]**：AGC 比较次数阈值
00：连续的 AGC 比较直到检测到 SYNCWORD
01~11：比较次数阈值

- **AGC3：AGC 控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	CDRST_THD_SEL[1:0]		ENVAVG_SEL[1:0]		—	IF_DETOK_THD[2:0]		
R/W	R/W		R/W		—	R/W		
Reset	0	0	1	0	0	1	0	0

Bit 7~6 **CDRST_THD_SEL[1:0]**：复位 AGC 的载波信号阈值

CDRST_THD_SEL[1:0]	GAIN_SEL[3:0]		
	0010b	0011b	其它值
00b	-32dBFS	-41dBFS	-48dBFS
01b	-35dBFS	-44dBFS	-48dBFS
10b	-38dBFS	-47dBFS	-48dBFS
11b	-41dBFS	-48dBFS	-48dBFS

若 AGC 操作完成且检测到的信号强度低于预设值时，AGC 进程复位并重新开始。

Bit 5~4 **ENVAVG_SEL[1:0]**：包络检测平均比率选择
00：1/16
01：1/32
10：1/64
11：1/128

Bit 3 保留，必须设为“0”

Bit 2~0 **IF_DETOK_THD[2:0]**：IF 检测正常阈值
经过一段增益稳定时间 (由 AGC7 寄存器决定) 后，AGC 电路在开始检测 IF 信号强度前会先等待 (IF_DETOK_THD×8) 个 ADCLK 周期。

- **AGC4：AGC 控制寄存器 4**

Bit	7	6	5	4	3	2	1	0
Name	GAIN_SEL[3:0]				—	AGC_ST[2:0]		
R/W	R				—	R		
Reset	0	0	0	1	0	0	0	1

Bit 7~4 **GAIN_SEL[3:0]**：增益曲线选择
0000：未选择增益曲线
0001：选择最大增益
0111：选择最小增益
有效值范围为 0000~0111。硬件自动选择增益。该字段和 CDRST_THD_SEL[1:0] 字段共同决定复位 AGC 的载波信号强度阈值，详见 AGC3 寄存器描述。

Bit 3 保留，必须设为“0”

Bit 2~0 **AGC_ST[2:0]**：AGC 状态机的状态
000~001：AGC 未完成
111：AGC 完成

- **AGC5：AGC 控制寄存器 5**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	AGC_FSEL[1:0]	
R/W	—	—	—	—	—	—	R/W	
Reset	0	0	0	0	0	0	0	0

Bit 7~2 保留，必须设为“000000”

Bit 1~0 **AGC_FSEL[1:0]**：AGC 滤波器配置
AGC_FSEL[1]：HPF 通带设定点
0：5/32 ADCLK
1：6/32 ADCLK
AGC_FSEL[0]：LPF 通带设定点
0：17/320 ADCLK
1：17/256 ADCLK
ADCLK=0.5×XCLK。XCLK=16MHz，建议设置 AGC_FSEL[1:0]=00b。

- **AGC7：AGC 控制寄存器 7**

Bit	7	6	5	4	3	2	1	0
Name	GAIN_STB[7:0]							
R/W	R/W							
Reset	0	0	1	1	0	0	0	0

Bit 7~0 **GAIN_STB[7:0]**：增益稳定计数
增益稳定延迟计数 (单位：ADCLK 周期)=GAIN_STB[7:0]×2

- **FCF1：滤波器系数控制寄存器 1**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SFRATIO[1:0]		—	—	—	—
R/W	—	—	R/W		—	—	—	—
Reset	0	0	0	0	0	1	1	0

Bit 7~6　保留，必须设为“00”

Bit 5~4　**SFRATIO[1:0]**：平滑滤波器比率选择
00：1/1
01：1/16
10：1/64
11：1/128

Bit 3~0　保留，必须设为“0110”

- **FCF2：滤波器系数控制寄存器 2**

Bit	7	6	5	4	3	2	1	0
Name	FSCALE[7:0]							
R/W	R/W							
Reset	0	1	0	0	0	1	0	0

Bit 7~0　**FSCALE[7:0]**：频率偏移比例参数低字节

- **FCF3：滤波器系数控制寄存器 3**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	FSCALE[11:8]			
R/W	—	—	—	—	R/W			
Reset	0	0	0	0	0	1	0	0

Bit 7~4　保留，必须设为“0000”

Bit 3~0　**FSCALE[11:8]**：频率偏移比例参数高字节
如果数据率为 100Kbps~250Kbps，则 FSCALE 参考 Lookup Table 建议设定值。
如果数据率 < 100Kbps，则 FSCALE 的值如下：
FSCALE[11:0]=round ((h×f_S/f_{XTAL}/(XODIV2+1))×2^{15})，其中 h 为解调系数，通过频率偏移和数据字符率计算：h=(2 × 频率偏移)/(数据字符率)。

- **FCF4：滤波器系数控制寄存器 4**

Bit	7	6	5	4	3	2	1	0
Name	CF_B12[7:0]							
R/W	R/W							
Reset	1	0	0	0	0	1	0	1

- **FCF5：滤波器系数控制寄存器 5**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CF_B12[9:8]	
R/W	—	—	—	—	—	—	R/W	
Reset	0	0	0	0	0	0	1	0

- **FCF6：滤波器系数控制寄存器 6**

Bit	7	6	5	4	3	2	1	0
Name	CF_B13[7:0]							
R/W	R/W							
Reset	1	0	0	0	1	0	1	0

- **FCF7：滤波器系数控制寄存器 7**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CF_B13[9:8]	
R/W	—	—	—	—	—	—	R/W	
Reset	0	0	0	0	0	0	0	0

- **FCF8：滤波器系数控制寄存器 8**

Bit	7	6	5	4	3	2	1	0
Name	CF_A12[7:0]							
R/W	R/W							
Reset	0	0	0	1	0	0	1	0

- **FCF9：滤波器系数控制寄存器 9**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CF_A12[9:8]	
R/W	—	—	—	—	—	—	R/W	
Reset	0	0	0	0	0	0	0	0

当数据速率在 49Kbps~2Kbps 范围内时，需要进行以下平滑滤波。

$CF\_A12[9:0]=mod(2^{10}+[(SFRATIO[1:0]-1)\times 2^{8}], 2^{10})$

- **FCF10：滤波器系数控制寄存器 10**

Bit	7	6	5	4	3	2	1	0
Name	CF_A13[7:0]							
R/W	R/W							
Reset	0	0	1	0	1	0	1	1

- **FCF11：滤波器系数控制寄存器 11**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CF_A13[9:8]	
R/W	—	—	—	—	—	—	R/W	
Reset	0	0	0	0	0	0	1	1

- **FCF12：滤波器系数控制寄存器 12**

Bit	7	6	5	4	3	2	1	0
Name	CF_B22[7:0]							
R/W	R/W							
Reset	0	0	0	1	0	1	0	0

- **FCF13：滤波器系数控制寄存器 13**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CF_B22[9:8]	
R/W	—	—	—	—	—	—	R/W	
Reset	0	0	0	0	0	0	0	1

- **FCF14：滤波器系数控制寄存器 14**

Bit	7	6	5	4	3	2	1	0
Name	CF_B23[7:0]							
R/W	R/W							
Reset	0	0	1	0	0	0	0	1

- **FCF15：滤波器系数控制寄存器 15**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CF_B23[9:8]	
R/W	—	—	—	—	—	—	R/W	
Reset	0	0	0	0	0	0	0	0

- **FCF16：滤波器系数控制寄存器 16**

Bit	7	6	5	4	3	2	1	0
Name	CF_A22[7:0]							
R/W	R/W							
Reset	0	1	1	1	1	0	0	0

- **FCF17：滤波器系数控制寄存器 17**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CF_A22[9:8]	
R/W	—	—	—	—	—	—	R/W	
Reset	0	0	0	0	0	0	0	0

- **FCF18：滤波器系数控制寄存器 18**

Bit	7	6	5	4	3	2	1	0
Name	CF_A23[7:0]							
R/W	R/W							
Reset	0	0	1	0	1	0	0	0

- **FCF19：滤波器系数控制寄存器 19**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CF_A23[9:8]	
R/W	—	—	—	—	—	—	R/W	
Reset	0	0	0	0	0	0	0	0

FCF4~FCF19 寄存器定义了 8 组 IIR 系数，不同 XTAL 时钟情况下这些寄存器的建议设定值如下面表格所示。

f_{XTAL}	16MHz	16MHz	16MHz	16MHz	16MHz
f_S	250Kbps	125Kbps	50Kbps	10Kbps	2Kbps
f_D	93.75kHz	46.875kHz	18.75kHz	40kHz	8kHz
D_K[19:0] (H)	$f_{RF}/[f_{XTAL}/(XODIV2+1)]$，取小数				
D_N[6:0] (H)	$f_{RF}/[f_{XTAL}/(XODIV2+1)]$，取整数				
SFRATIO[1:0] (D)	0	0	0	1	3
FSCALE[11:0] (H)	294	119	4C	A4	20
CF_B12[9:0] (H)	2CA	01D	0	0	0
CF_B13[9:0] (H)	062	346	0	0	0
CF_A12[9:0] (H)	358	022	0	310	302
CF_A13[9:0] (H)	3E9	331	0	0	0
CF_B22[9:0] (H)	3B3	386	0	0	0
CF_B23[9:0] (H)	03E	012	0	0	0
CF_A22[9:0] (H)	3E9	008	0	0	0
CF_A23[9:0] (H)	039	008	0	0	0

Bank 2 控制寄存器

上电复位后，所有控制寄存器被设置为初始值。软件复位后，这些控制寄存器也将恢复至初始值。

地址	名称	位							
		7	6	5	4	3	2	1	0
26h	RSV1	保留							
27h	RSV2	保留							
28h	RSV3	保留							
29h	RSV4	保留							
2Dh	RSV5	保留							
2Eh	RSV6	保留							
2Fh	RVS7	保留							
30h	RSV8	保留							
31h	RSV9	保留							
34h	RSV10	保留							
3Ah	RSV11	保留							

注：未列于此表格的地址，是预留给将来使用。建议不要通过任何方式修改这些地址的初始值。

Bank 2 寄存器的建议设置值如下标所示。

地址	名称	频段	
		433MHz	868MHZ
26h	RSV1	03h	
27h	RSV2	88h	
28h	RSV3	A3h	
29h	RSV4	80h	
2Dh	RSV5	16h	
2Eh	RSV6	64h	74h
2Fh	RSV7	44h/54h (≥ 100Kbps: 54h)	
30h	RSV8	00h	
31h	RSV9	64h	
34h	RSV10	BCh	9Ch
3Ah	RSV11	94h	

特殊功能说明

Sub-1GHz RF 收发器

BA45F5650 采用完全集成的低中频接收器架构。接收到的信号先经过一个低噪声放大器 (LNA) 进行放大，接着通过一个正交混频器将频率向下转换为中频。混频器输出信号经过通道选择滤波器进行滤波，滤除不必要的带外干扰和图像信号。经过滤波后，中频信号经过一个模拟可编程增益放大器 (PGA) 进行放大。接着使用一个 10 位 ΣΔ A/D 转换器将放大后的中频信号数字化。

RF 内置一个自动增益控制 (AGC) 单元，可根据数字调制解调器产生的 RSSI 来调节接收器增益。此 AGC 功能使得 RF 可在灵敏度级别至 +10dBm 输入功率范围之内工作。

BA45F5650 采用完全内置的小数 N 分频合成器，包含 RF VCO、回路滤波器、内带负载电容的数字控制晶振。在 PCB 上装 VCO 负载电感可降低 VCO 谐振频率，从而实现 4.2mA 的 RX 模式电流损耗。小数 N 分频合成器架构允许用户将其潜在应用扩展至更广泛的频率范围。

传输会话采用 VCO 直接调制架构。与传统的直接上变频发送器不同，这里利用小数 N 分频合成器的优势直接将 GFSK 调制信号接入 VCO。因此，布局面积和电流损耗都比传统直接上变频发送器的小很多。精细的分辨率可产生低 FSK 误差的 GFSK 信号。调制好的信号接入一个 E 类功率放大器 (PA)，最大输出功率可达 +13dBm。

串行接口

RF 通过一个 3 线 SPI 接口 (CSN, SCK, SDIO) 或一个 4 线串行接口 (SDO 位于 GIO1 或 GIO2) 与 MCU 通信，数据速率高达 4Mbps。一笔 SPI 传输其实就是一个 (8+8×n) 位的序列，包含一个 8 位的命令和 n×8 位数据，其中 n 可以是 0 或者任何自然数。若 n 大于地址边界，则会返回地址 0。BA45F5650 要访问 RF 部分时应将 CSN (SPI 芯片选择) 引脚拉低。用户可通过 SPI 接口访问控制寄存器并发出 Strobe 命令。当写数据到 RF 芯片时，SPI 数据会在 SCK 信号上升沿时存入对应寄存器。若从 RF 芯片寄存器读取数据，当输入目标寄存器地址后，每一个位数据会在 SCK 信号下降沿时传出。

命令 (8 位)								数据 (8 位)							
C7	C6	C5	C4	C3	C2	C1	C0	D7	D6	D5	D4	D3	D2	D1	D0

SPI 命令格式

有两种命令，一种是只有 1 个字节的命令，即 CmdO；另一种是 1 个字节命令加 n 个字节数据，即 CmdD。

C7	C6	C5	C4	C3	C2	C1	C0	说明	CmdO	CmdD
0	1	A5	A4	A3	A2	A1	A0	写入控制寄存器		√
1	1	A5	A4	A3	A2	A1	A0	读取控制寄存器		√
0	0	1	x	x	x	B1	B0	设置寄存器存储区	√	
0	0	0	1	x	x	x	0	写同步码命令		√
1	0	0	1	x	x	x	0	读同步码命令		√
0	0	0	1	x	x	x	1	TX FIFO 写命令		√
1	0	0	1	x	x	x	1	RX FIFO 读命令		√
1	0	0	1	1	1	1	1	读取芯片 ID 命令		√
0	0	0	0	1	0	0	0	软件复位命令	√	
0	0	0	0	1	0	0	1	TX FIFO 地址指针复位命令	√	
1	0	0	0	1	0	0	1	RX FIFO 地址指针复位命令	√	
0	0	0	0	1	0	1	0	Deep Sleep 模式	√	
0	0	0	0	1	0	1	1	Idle 模式	√	
0	0	0	0	1	1	0	0	Light Sleep 模式	√	
0	0	0	0	1	1	0	1	Standby 模式	√	
0	0	0	0	1	1	1	0	TX 模式	√	
1	0	0	0	1	1	1	0	RX 模式	√	

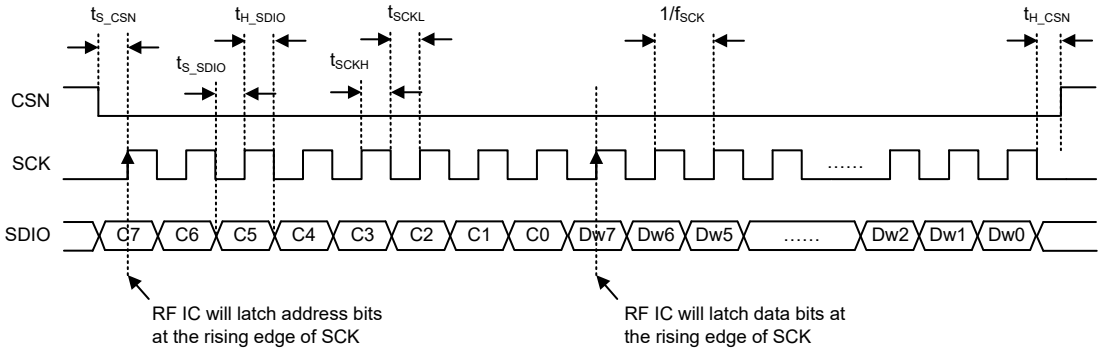
A5~A0：控制寄存器地址；

x：硬件上无关，但建议软件设置为 0；

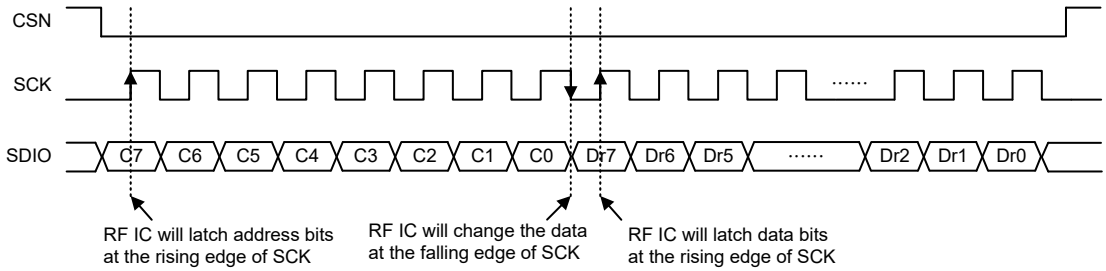
B1~B0：Bank 编号

注：1. 此芯片支持多字节读 / 写操作，每次读或写操作后地址自动递增。
2. 在单个 CSN 使能周期内，每个读 / 写命令之后允许软件读 / 写多个字节。
3. 在 Sleep 模式下，GIO 引脚维持上一个工作模式时的电平状态。
4. 芯片 ID 是一个 2 字节数据。

SPI 时序



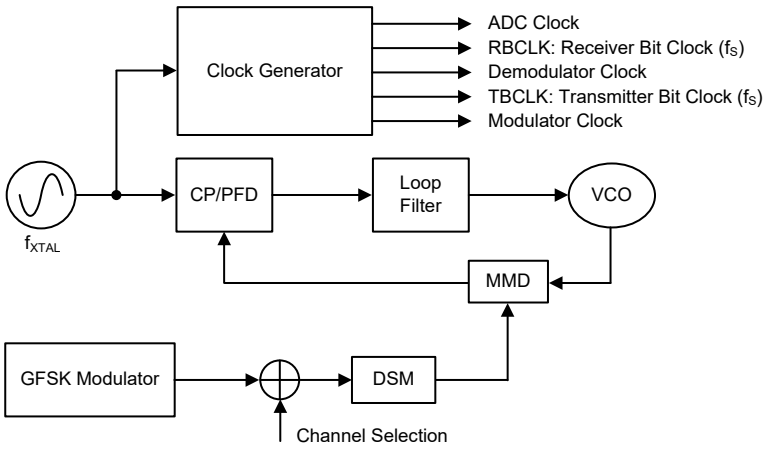
3 线 SPI 接口写入 1 个字节数据



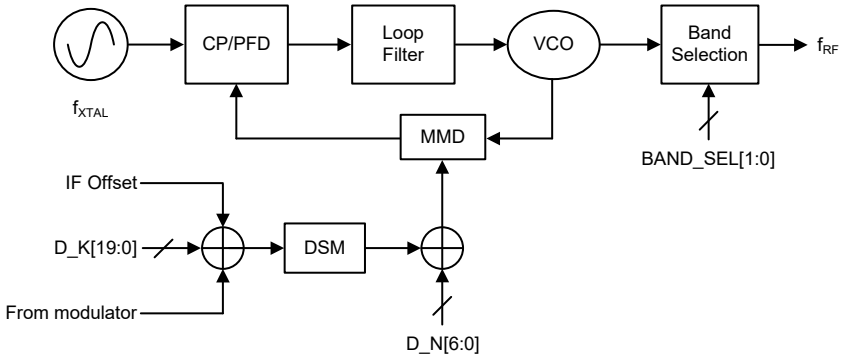
3 线 SPI 接口读取 1 个字节数据

系统时钟

RF 的主要系统时钟来自 XTAL 晶振。各种功能模块的所有内部操作时钟都来自此晶振。



频率合成器



RF 收发器频率由一个高分辨率的小数 N Delta Sigma 频率合成器产生。通过合理设置 D_N[6:0] 和 D_K[19:0]，可产生一个低噪声 LO 频率，适用于各种无线电规范标准，如 ETSI EN 和 FCC 等等。在 RX 模式下，此合成器可为 RX 混频器操作提供一个 LO-IF 频率，通过配置 RXIFOS[11:0] 产生所需的 IF 偏移量。当数据速率大于等于 200Kbps 时，IF 须设为 300kHz，否则 IF 应设置为 200kHz。在 TX 模式下，调制器可提供额外的基带数据频率偏移波。

$$D\_N[6:0]=Floor\left(\frac{f_{RF}}{f_{XTAL}/(XODIV2+1)}\right)$$

$$D\_K[19:0]=Floor\left(\left(\frac{f_{RF}}{f_{XTAL}/(XODIV2+1)}-D\_N[6:0]\right)\times 2^{20}\right)$$

$$RXIFOS[11:0]=Floor\left(\left(\frac{f_{IF}}{f_{XTAL}/(XODIV2+1)}\right)\times 2^{17}\right)$$

调制器

BA45F5650 支持 GFSK 调制。RF 部分内置一个 BT=0.5 的高斯滤波器以实现脉冲平滑的目的。发送器的频率偏移 f_{DEV} 由 FSCALE[11:0] 位设定。FSCALE[11:0] 的值取决于解调器系数 h、XO 输出除以 2 控制位 XODIV2、数据速率 f_S 以及 f_{XTAL}。

$$h=\frac{2\times f_{DEV}}{f_S}$$

$$FSCALE[11:0]=round\left(\left(h\times\frac{f_S}{f_{XTAL}/(XODIV2+1)}\right)\times 2^{15}\right)$$，取最低 12 位

对于低数据速率 (≤10Kbps) 应用，解调系数建议值为 8。对于高数据速率 (≥50Kbps) 应用，解调器系数建议为 0.75。当数据速率介于前面这两个值之间时，确保频率偏移高于 20kHz。

当数据速率大于等于 100Kbps 时，FSCALE 字段需乘以一个比例因数。数据速率大于等于 100Kbps 时 FSCALE 的建议值，在滤波器系数控制寄存器之后的表格里有提供。

状态机

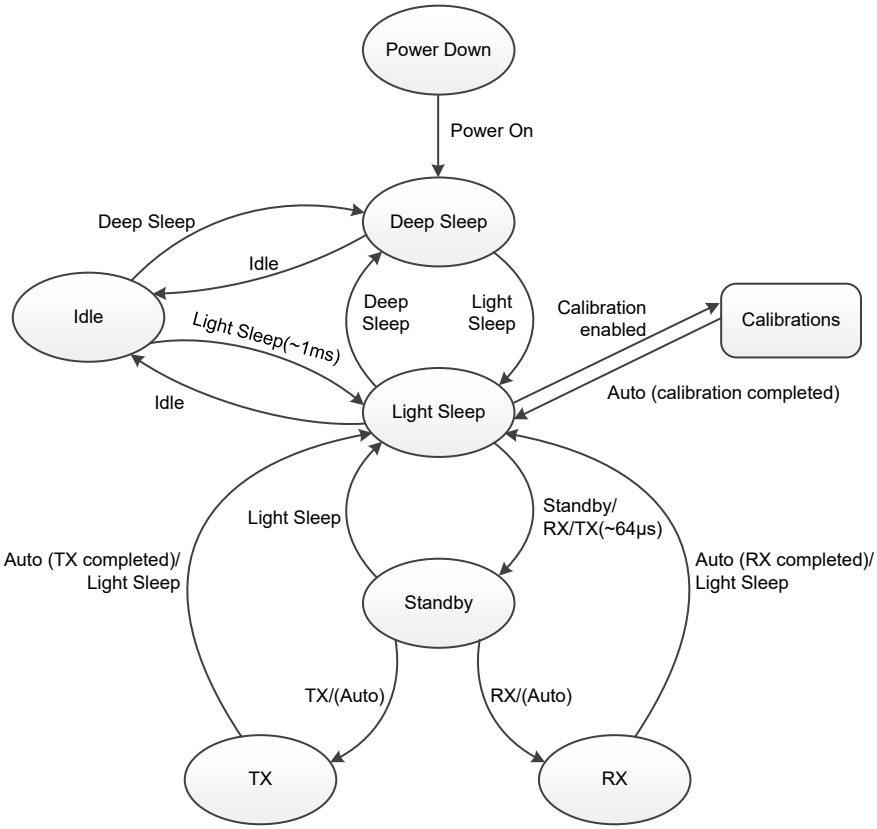
RF 有 7 种工作模式。所有工作模式以及对应的关键功能开启 / 关闭状态如下所示。

1. Power Down 模式
2. Deep Sleep 模式
3. Light Sleep 模式
4. Standby 模式
5. Idle 模式
6. TX 模式
7. RX 模式

模式	寄存器保存	3.3V	LIRC	稳压器	XO	Standby+VCO	TX	RX	Strobe 命令
Power Down	No	OFF	OFF	OFF	OFF	OFF	OFF	OFF	—
Deep Sleep	Yes	ON	OFF	OFF	OFF	OFF	OFF	OFF	0000_1010
Light Sleep	Yes	ON	OFF	ON	ON	OFF	OFF	OFF	0000_1100
Idle	Yes	ON	ON	OFF	OFF	OFF	OFF	OFF	0000_1011
Standby	Yes	ON	OFF	ON	ON	ON	OFF	OFF	0000_1101
TX	Yes	ON	OFF	ON	ON	ON	ON	OFF	0000_1110
RX	Yes	ON	OFF	ON	ON	ON	OFF	ON	1000_1110

TX/RX FIFO 模式 (DIR_EN=0) 状态机

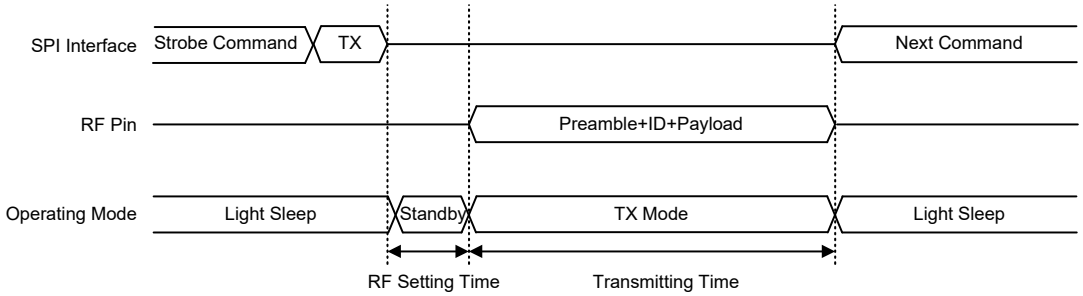
若 DIR_EN 位为 0，芯片模式切换通过 MCU 发送 Strobe 命令来实现，且 TX/RX 数据来自数据包处理硬件。



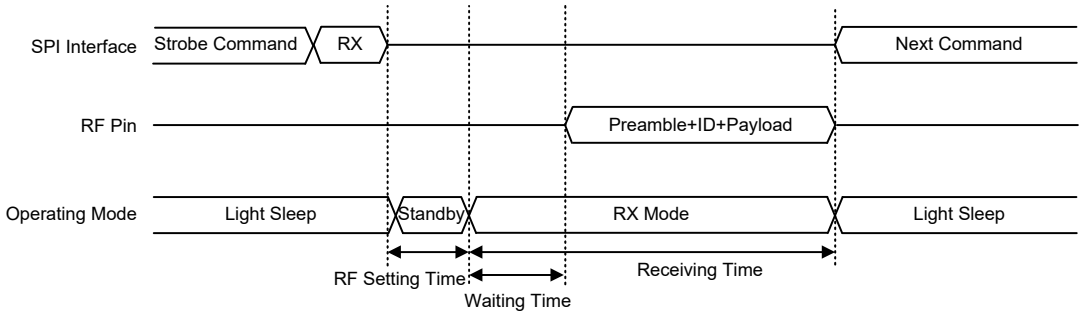
FIFO 模式状态方框图

RF 初始状态为 Power Down 模式。芯片完成内部上电复位后先进入 Deep Sleep 模式，等待来自 MCU 的 Strobe 命令。若接收到 Light Sleep 命令，芯片将使能内部 LDO、起振 XO 并进入 Light Sleep 模式。在此模式下，若有需要 MCU 可让 RF 执行校准功能。若要进行正常的 TRX 操作，MCU 可发送 RX 或 TX 命令给芯片。当芯片接收到 TX 或 RX 命令后，会先进入 Standby 模式并持续一段时间，此时间称为 TX/RX 设置时间。经过这段设置时间后，芯片将进入 RX 或者 TX 模式。芯片保持 TX/RX 状态直到 TX/RX 操作完成。这之后芯片自动返回 Light Sleep 模式。

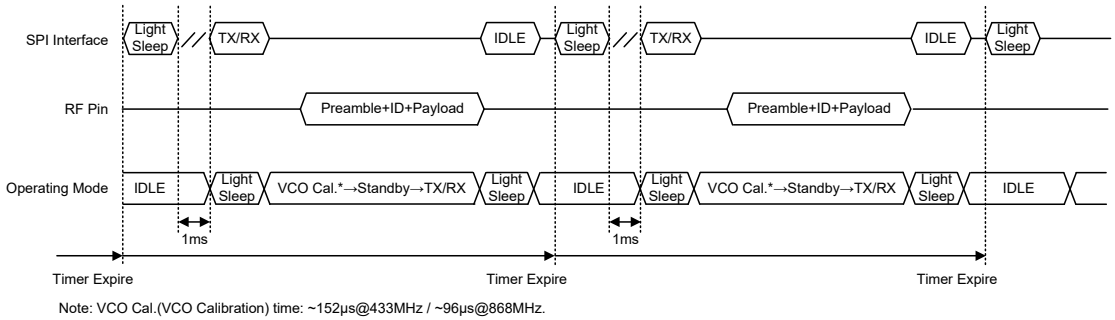
针对低功率周期性无线传输，该芯片支持低功耗 Idle 模式，在此模式下 LIRC 和唤醒定时器开启。合理设置定时器并发送 Idle 命令，芯片将关闭 LDO 和 XO 并进入 Idle 模式。芯片保持 Idle 模式直到达到定时器定时时间，与此同时芯片通过 GIO 发送一个中断请求以唤醒 MCU。接着，MCU 可让芯片进入 Light Sleep 模式，接着再执行 TX/RX 相关操作。当 TX/RX 操作完成，MCU 可发送 Idle 命令给芯片使其再次进入 Idle 模式。



FIFO 模式 TX 时序图



FIFO 模式 RX 时序图



Note: VCO Cal.(VCO Calibration) time: ~152μs@433MHz / ~96μs@868MHz.

周期性 TX/RX 时序

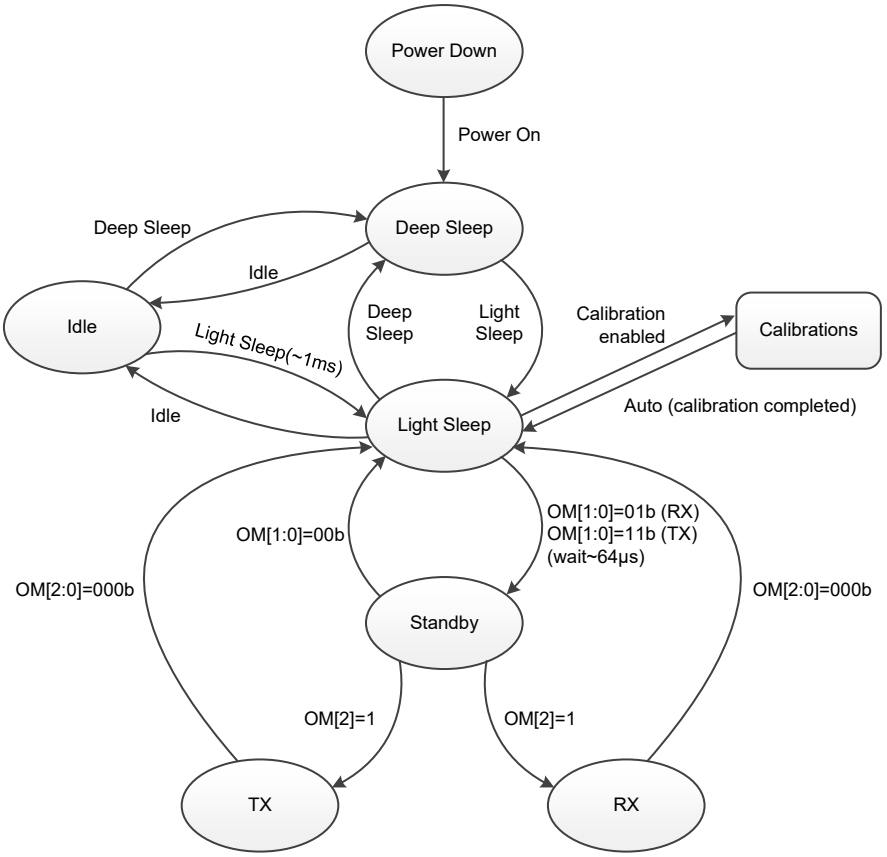
TX/RX Direct 模式 (DIR_EN=1) 状态机

若设置 DIR_EN 为 1，TX 数据由主控 MCU 直接发送给 RF，RX 数据由 RF 直接发送给 MCU。为了简化 RF 与 MCU 之间的数据位时钟同步，设置 GIO4S[3:0]，RF 便可从 GIO4 输出 TBCLK/RBCLK 时钟。TBCLK 和 RBCLK 都是 50/50 占空比周期。在发送模式下，MCU 在 TBCLK 信号上升沿时输出位数据，RF 在 TBCLK 信号的下降沿时采样 TX 位数据。在接收模式下，MCU 在 RBCLK 信号上升沿时接收数据，RF 在 RBCLK 信号下降沿时输出位数据。MCU 可设置 GIO1S[2:0]/GIO2S[2:0] 选择 GIO1/GIO2 用于 TX/RX 位数据传输。

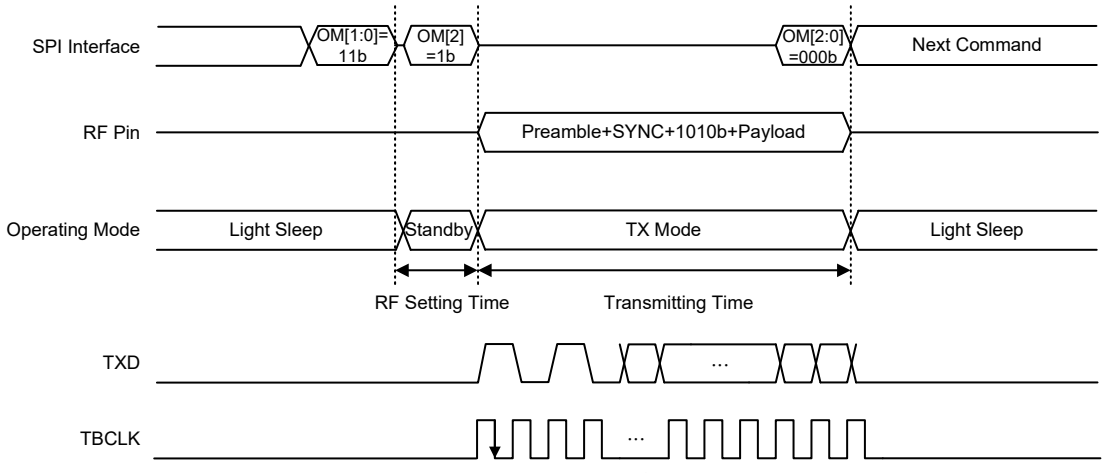
若要在 Direct 模式下进行 TX 操作，MCU 需设置 OM[1:0]，即 RTX_SEL 和 SX_EN 位，为“11”以选中 TX 模式并先让 RF 进入 Standby 模式。接着设置 OM[2]，即 RTX_EN 位，为“1”使 RF 开始发送 TX 数据。一旦 MCU 将 OM[2:0] 位设置为“000”，RF 将返回 Light Sleep 模式。

若要在 Direct 模式下进行 RX 操作，MCU 需先设置 OM[1:0] 位为“01”，接着设置 OM[2] 为“1”使 RF 开始接收数据。当芯片接收到匹配的同步码时，会输

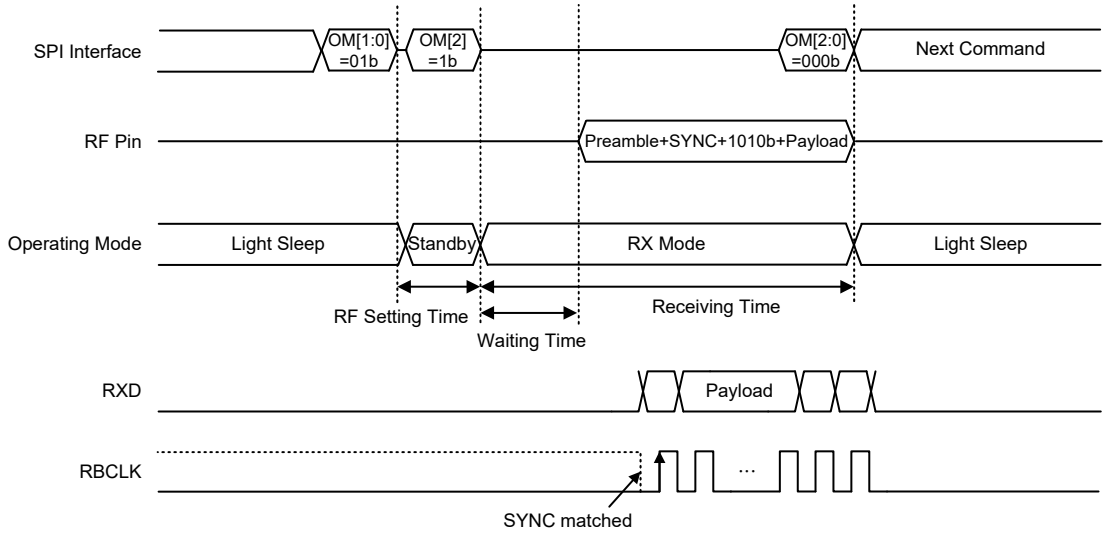
出 RBCLK 时钟，接收数据位，即有效载荷部分，然后再发送给 MCU。
在 Direct 模式下，对传输的数据长度无限制。



Direct 模式状态方框图



Direct 模式 TX 时序图



Direct 模式 RX 时序图

校准

此芯片有三种校准功能，即 VCO，RC 和 LIRC 校准，帮助用户自动选择合适的设置来实现 PVT (制程 - 电压 - 温度) 变化补偿。ACAL_EN 控制位可同时使能 VCO 和 RC 校准功能，当此位置高后两种校准功能将自动执行。当两个校准功能都完成后，ACAL_EN 位由硬件自动清零。MCU 可检查 ACAL_EN 位状态或者通过校准完成中断标志位 CALCMPF 位确认校准状态。LIRC 校准功能有自己独立的使能位，LIRCCAL_EN，允许单独执行 LIRC 校准功能。

LIRC 校准

RF 内置一个低频 RC 晶振，在 Idle 模式下可为唤醒定时器提供时钟源。校准后，该低频 RC 晶振可对 PVT 变化提供 ±2% 补偿。校准过程通过 LIRC 频率曲线设置使其频率接近 32768Hz。接着，另外再进行一个 LIRC 校正步骤以微调唤醒定时器使其精度误差小于 ±1%。

在进行 LIRC 校准之前，MCU 需先设置 LIRC_OW 为“0”，LIRC_EN 为“1”。在 Light Sleep 模式下，当 LIRCCAL_EN 位被 MCU 置高时，RF 将执行 LIRC 校准。当 LIRC 校准完成后，LIRCCAL_EN 位由硬件清零。LIRC 校准过程需要花大概 4ms 时间。

AGC & RSSI

为了加强接收动态范围，并确保信号的信噪比不低于解调器最小信噪比要求，该芯片还内置一个 AGC (自动增益控制) 功能模块。在 ADC 保持在设定点和 -26dBFS 之间之前，AGC 会先微调接收器增益以确保得到有效的信号电平。设定点范围是 -6dBFS 到 -12dBFS，由 SAT_SEL[1:0] 设置调整。FS 表示 A/D 转换器满额。

RF 还内建一个接收器信号强度指示器 (RSSI) 测量功能模块。RSSI 计算引擎用于计算经过 ADC 转换后的接收信号强度。结合计算出来的 ADC 信号强度值以及接收器链路总增益得出 RSSI 值。RSSI 有效读取值范围是 -110dBm 到 -10dBm。RSSI 测量误差通常低于 ±6dBm。读取值的单位是 -dBm。一共有两组 RSSI 读取值，一组是 RSSI_SYNC_OK[7:0]，此值为检测到有效同步码正常后的 RSSI 测量值快照。另一组是 RSSI_NEGDB[7:0]，为实时的 RSSI 计算结果。在接收

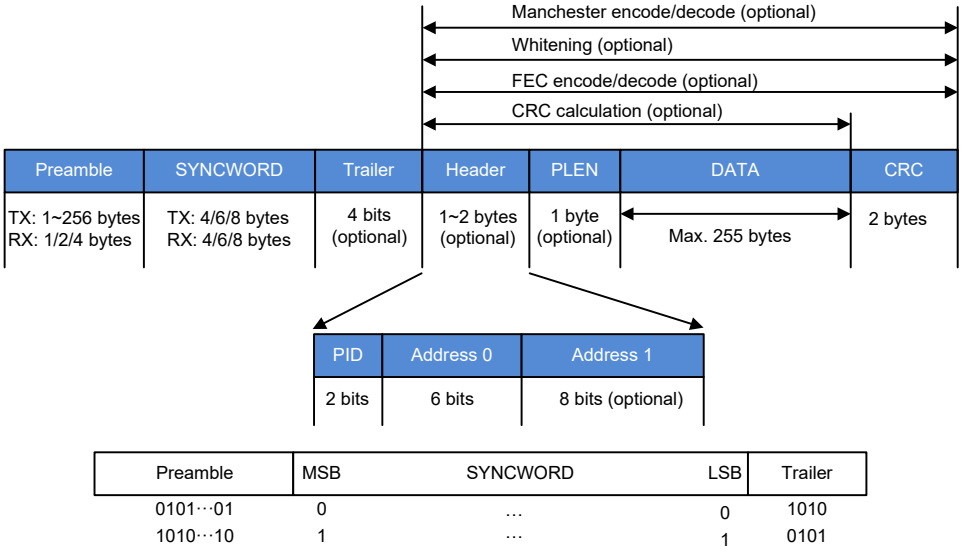
模式下，MCU 可访问 RSSI_NEGDB 字段。当接收完成时，MCU 可检查 RSSI_SYNC_OK[7:0] 以获取接收信号强度。

数据包处理器

在 TX 模式下，数据包处理器将要发送的数据从 FIFO 中移出并按照数据包格式执行通道编码，然后将数据包发送给调制器。在 RX 模式下，数据包处理器对来自解调器的数据进行通道解码，并将有效载荷存入 FIFO。

数据包处理器负责执行多个任务，如插入前导码和同步码、正向纠错、CRC 计算 / 校验、数据白化 / 去白以及曼彻斯特编码 / 解码。

数据包格式



Preamble	MSB	SYNCWORD	LSB	Trailer
0101…01	0	…	0	1010
1010…10	1	…	1	0101

注：1. 前导码格式依据同步码最高位进行反转
若 MSB=0，前导码格式 = 0101…01
若 MSB=1，前导码格式 = 1010…10
2. 连接码格式依据同步码最低位进行反转
若 LSB=0，连接码 = 1010
若 LSB=1，连接码 = 0101
3. Trailer 域包含 4 位，可由用户自选，通过 TRAILER_EN 位控制。

- 前导码 (Preamble)
数据包开头是一个 1~256 字节的前导码，TX 模式下此长度由 TXPMLEN[7:0] 决定。在 RX 模式下，前导码检测长度为 1、2 或 4 个字节，通过 RXPMLEN[1:0] 选择。
- 同步码 (SYNCWORD)
同步码长度由 SYNCLEN[1:0] 设置，TX 模式下可为 4、6 或 8 个字节。在 RX 模式下，同步码检测长度也是 4、6 或 8 个字节。当 RX 端接收到匹配的同步码数据包时，数据域部分将被存入 FIFO。
- 连接码 (Trailer)
连接码字段固定是 4 位，是介于同步码及后续有效载荷间的连接字段。
- 头码 (Header)
是否需要头码，即有效载荷头码，可由用户自选，通过 PLH_EN 位使能。有效载荷头码长度是 1 或 2 个字节，由 PLHLEN 位决定。当 PLHLEN 位为 0 时，

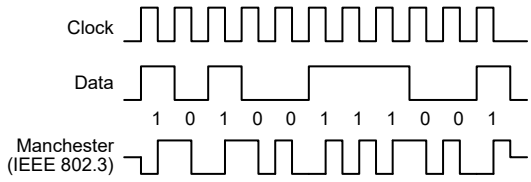
数据包中的头码仅包含 PID[1:0] 和 PLHA[5:0] (地址 0) 字段。此时 PID[1:0] 位于头码字段的 7~6 位。若 PLHAC_EN 位为 0，PLHA[5:0] 可用作软件标志位，实际功能可由用户自定义。若 PLHAC_EN 位为 1，芯片将对比自身 PLHA[5:0] 和接收到的 PLHA[5:0] 字段值。若匹配，则接收到的数据会移入 RX FIFO，否则紧接着的数据会被丢弃。PLHA[5:0] 字段的目的是用于支持播送功能，当 PLHA[5:0] 等于 0 时，允许 RF 不执行地址过滤机制。
当 PLHLEN 位设为 1 时，地址域的长度扩展到 14 位，由地址 0 (PLHA[5:0]) 和地址 1 (PLHEA[7:0]) 构成。

- 有效载荷长度 (PLEN)
  PLEN 域可由用户自选，一旦通过 PLLEN_EN 位使能后，PLEN 域固定为 1 个字节。当 PLLEN_EN 位置 1 时，数据域的长度是可变的，由每个 TX/RX 数据包中的 PLEN 域决定。
- 数据 (DATA)
  TX 模式下的 TX 数据长度取决于 TXDLEN[7:0]。在 Extend FIFO 模式下，最大长度可为 255 个字节。在 Infinite FIFO 模式下，长度无限制可超过 255 字节。若 PLLEN_EN 位等于 1，TX 数据包的 PLEN 域使能，此时 PLEN 值等于 TXDLEN[7:0]。在 RX 模式下，若 PLLEN_EN 等于 0，RX 数据长度由 RXDLEN[7:0] 决定；若 PLLEN_EN 等于 1，RX 数据长度由 PLEN 域决定。
- 循环冗余码校验 (CRC)
  CRC 域可由用户自选，通过 CRC_EN 位使能。建议始终设置 CRC_EN 位为 1 以检查数据的正确性。共有两种 CRC 格式，通过 CRCFMT 位选择。
  CRCFMT=0：CCITT-16-CRC $G(X)=X^{16}+X^{12}+X^{5}+1$
  CRCFMT=1：IBC-16-CRC $G(X)=X^{16}+X^{15}+X^{2}+1$
  注：CRC 初始值为 FFFF。
- 正向纠错 (FEC)
  该数据编码 / 解码功能由 FEC_EN 位使能。使用 (7, 4) 汉明码对每 4 位数据进行大于等于 1 位的错误检测。正向纠错之后，每笔数据的数据长度为 (4+3)×2=14 位。

- **汉明码功能表格**

位	7	6	5	4	3	2	1
传送位	D3	D2	D1	P2	D0	P1	P0
P0	Y	N	Y	N	Y	N	Y
P1	Y	Y	N	N	Y	Y	N
P2	Y	Y	Y	Y	N	N	N

- 数据白化 (Data Whitening)
  数据白化 / 去白功能通过 WHT_EN 位使能。使用 PN7 码与发送的数据进行异或运算。数据白化种子值由 WHTSD[6:0] 设置。
- 曼彻斯特码 (Manchester Code)
  曼彻斯特编码 / 解码功能由 MCH_EN 位使能。每一个位在经过曼彻斯特编码后变成两位，解码后再恢复至一位。



曼彻斯特码举例

FIFO 工作模式

Burst 模式下，RF 发送器要发送的数据来自 FIFO 且由 MCU 预先写好。共有四个 FIFO 模式可支持各种应用。这些模式为 Simple FIFO 模式、Block FIFO 模式、Extend FIFO 模式和 Infinite FIFO 模式。

FIFO 复位

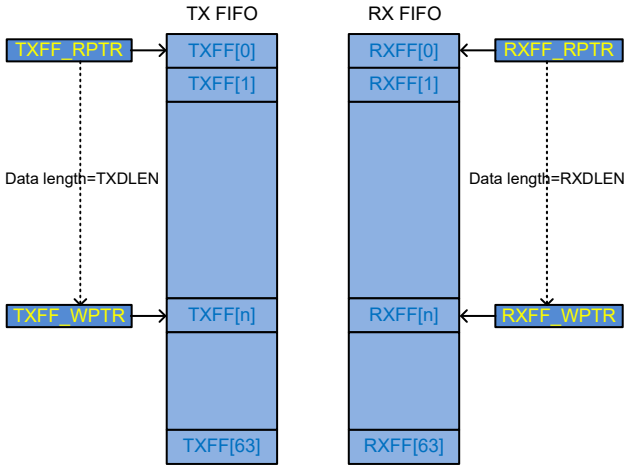
在 Burst 模式下使用 FIFO 之前，需先发送 TX FIFO 地址指针复位命令以及 RX FIFO 地址指针复位命令，以复位 FIFO 指针和缓冲器。在这之后，FIFO 将复位至初始状态。

Simple FIFO 模式

此 FIFO 模式用于 TX/RX 数据长度小于等于 64 字节的一般应用。注意，数据长度不可超过 64 个字节。要使用 Simple FIFO 模式，MCU 需通过 SPI 写 FIFO 命令将要发送的数据写入 FIFO。发送的顺序为先写的字节先发送，每个字节里的最高位先发送。用户需预先确定好所有发送数据包格式，包括前导码、同步码以及数据包编码如正向纠错、CRC 和数据白化。当 FIFO 数据填写完成后，将 TXFFSA[5:0] 字段清零，并将 TXDLEN[7:0]/RXDLEN[7:0] 字段设置为所需的发送 / 接收长度，单位为字节。接着发送 TX 命令开始数据传输。当前发送完成后，数据会被保存在 FIFO 中并等待下一次传输。

编程步骤：

1. 通过 SPI 复位 TX FIFO 命令复位 TX FIFO。
2. 通过 SPI 复位 RX FIFO 命令复位 RX FIFO。
3. TXFFSA[5:0] 必须清零。
4. 通过 SPI 写 FIFO 命令填写 TX FIFO。
5. 设置 TXDLEN[7:0]/RXDLEN[7:0] 控制 TX/RX 长度，单位为字节。
6. 发送 TX 命令给发送器，发送 RX 命令给接收器。
7. 通过 TX/RX 完成 IRQ 告知 TX/RX 完成状态。
8. 重新发送含相同数据的 TX 数据包会自动将 TXFF_RPTR 清零。

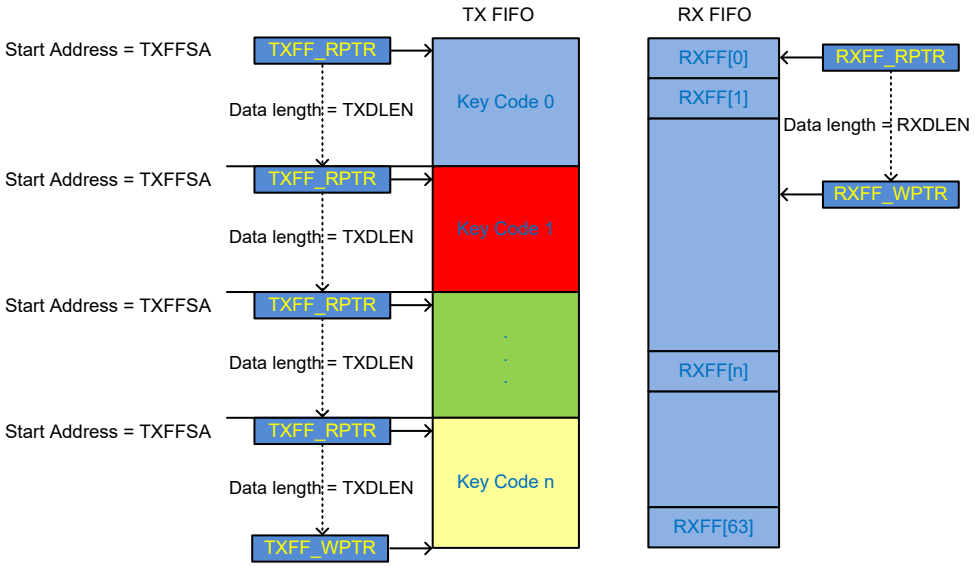


Block FIFO 模式

Block FIFO 模式可支持多键代码应用。用户需先将所有按键代码写入 FIFO。当有按键被按下时，MCU 会侦测对应哪个按键，将 TXFFSA[5:0] 设置为对应按键码的起始地址，并设置 TXDLEN[7:0] 指示按键码长度，接着发送 TX 命令开始传输。此模式最大 FIFO 长度限制为 64 个字节。

编程步骤：

1. TX：通过 SPI 写 FIFO 命令将按键 0~n 的代码写入 TX FIFO。
2. TX：设置 TXDLEN[7:0] 表示按键代码长度。
3. TX：当有按键按下时，MCU 会将 TXFFSA[5:0] 设置为对应按键代码的起始地址。
4. RX：将 RXDLEN[7:0] 设置为按键代码长度，然后借由 SPI 命令进入 RX 模式。
5. TX：发送 TX 命令给发送器。
6. RX：发送 RX 命令给接收器。
7. 通过 TX/RX 完成 IRQ 告知 TX/RX 完成状态。



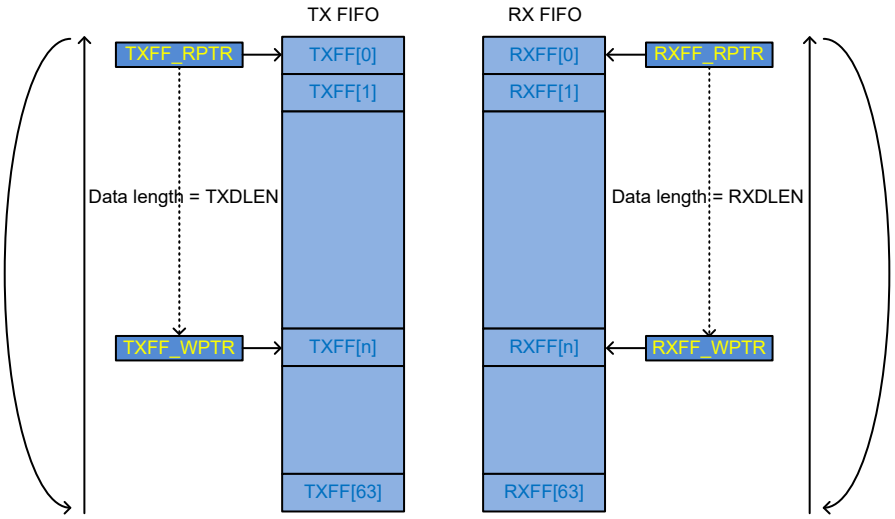
Extend FIFO 模式

Extend FIFO 模式适用于传输有效载荷数据长度较长的数据包。最大长度为 255 个字节。由于 FIFO 的物理长度为 64 字节，想要扩展每个数据包的可用发送长度，MCU 和 FIFO 控制器之间需要一个握手机制。

设置 FFMG[1:0] 决定 FIFO 数据长度边界，然后设置 FFMG_EN 位使能边界检测功能，当 TX FIFO 数据长度小于所选边界时通知 MCU。MCU 收到此提醒时应尽快往 TX FIFO 写数据以避免 TX FIFO 下溢迫使传输中断。

编程步骤：

1. 设置 FFMG_EN 使能 FIFO 低阈值检测功能，设置 FFMG[1:0] 选择边界值长度为 4、8、16 或 32 字节。
2. 设置 FIFOLTIE 位为 1 使能 FIFO 低阈值 IRQ。
3. 设置 GIOnS 字段 (n=1~4) 为“101”，则 IRQ 可从 GIO1~GIO4 输出。
4. TX：若检测到 FIFO 低阈值 IRQ 信号，MCU 会往 TX FIFO 写数据，写入长度小于等于 (64-FFMG[1:0]) 字节。接着 MCU 将 FIFO 低阈值 IRQ 标志位 FIFOLTIF 清零。MCU 重复这一步骤直到所有 TX 数据都完全写入 TX FIFO。
5. RX：若检测到 FIFO 低阈值 IRQ 信号，MCU 会从 RX FIFO 读取数据，读取长度等于 FFMG[1:0] 字节。接着 MCU 将 FIFO 低阈值 IRQ 标志位 FIFOLTIF 清零。MCU 重复这一步骤直到接收到 RX 完成 IRQ，并从 RX FIFO 读取剩余数据。



Infinite FIFO 模式

编程步骤：

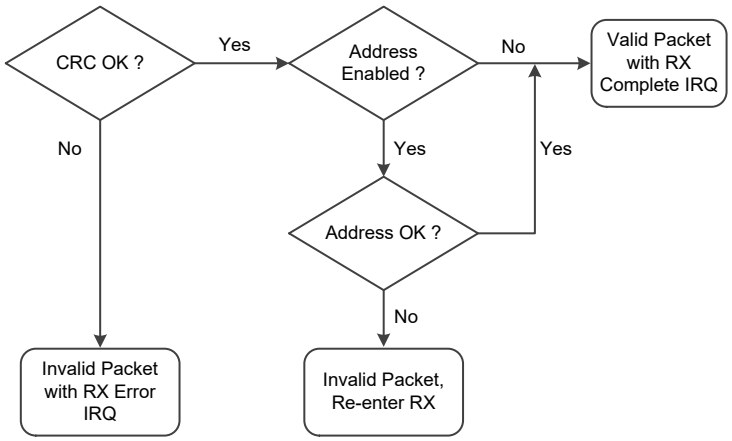
1. 设置 FFINF_EN 为 1 使能 Infinite FIFO 模式。
2. 此模式下的握手机制以及 IRQ 功能都与 Extend FIFO 模式下的一样。
3. TX：当接收到 FIFO 低阈值 IRQ 时，MCU 继续往 TX FIFO 写入 TX 数据，写入长度小于等于 (64-FFMG[1:0]) 字节。接着 MCU 将 FIFO 低阈值 IRQ 标志位 FIFOLTIF 清零。MCU 重复这一步骤直到它想结束 Infinite FIFO 模式。若要想结束此模式，当收到 IRQ 并往 TX FIFO 写入数据后，若剩余待发送数据的长度小于 192 字节并大于 64 字节，MCU 应将 FFINF_EN 位清零并将

TXDLEN[7:0] 设置为剩余发送数据的长度。针对每一次传输，此结束配置只需设置一次。当所有目标数据都完全发送完后，数据包发送终止。

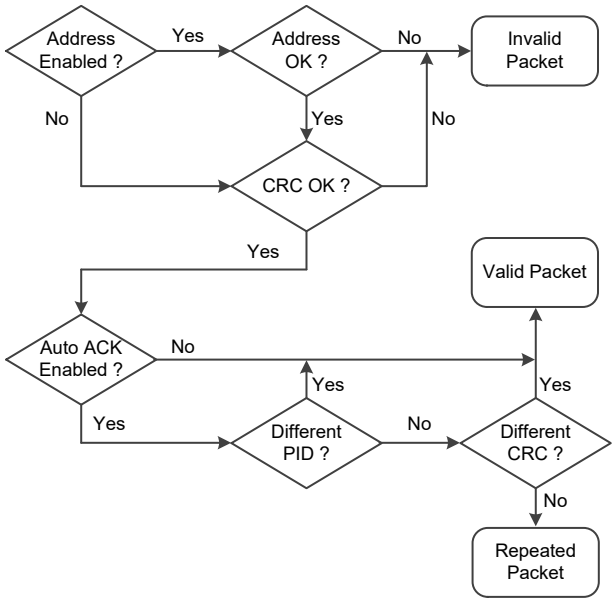
4. RX：当接收到 FIFO 低阈值 IRQ 时，MCU 从 RX FIFO 读取数据，读取长度等于 FFMG[1:0] 字节。接着 MCU 将 FIFO 低阈值 IRQ 标志位 FIFOLTIF 清零。MCU 重复这一步骤直到它想结束 Infinite FIFO 模式。若要想结束此模式，当收到 IRQ 并从 RX FIFO 读取数据后，若剩余待接收数据的长度小于 192 字节并大于 64 字节，MCU 应将 FFINF_EN 位清零并将 RXDLEN[7:0] 设置为剩余接收数据的长度。针对每一次传输，此结束配置只需设置一次。当所有目标数据都完全接收完后，数据包接收终止。

接收数据包判断

在正常 RX 工作模式下，数据包接收遵循以下判断标准。

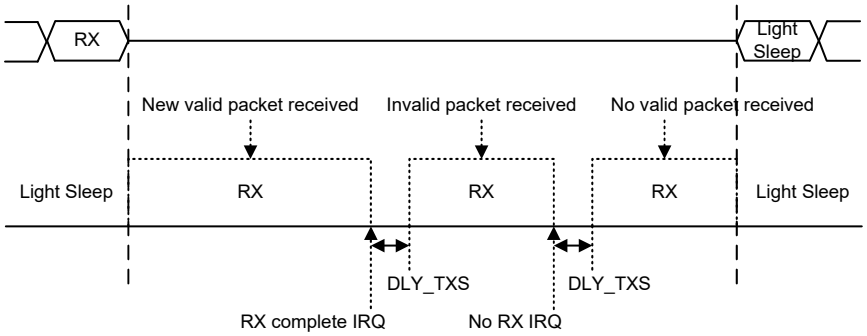


RF 针对连续 RX 模式以及自动应答模式采用额外的接收器数据包判断机制。这些特殊链接层函数存在的目的是为了在处理 TRX 数据包事务时减少 MCU 负载。



连续 RX 模式

RF 还支持特殊的连续 RX 工作模式。MCU 可设置 RXCON_EN 位为 1 来使能此模式并发送 RX 命令给芯片以启动此模式。若接收到一个有效的 RX 数据包，芯片将会向 MCU 发出 RX 完成中断请求。经过 DLY_TXS[2:0] 定义的一段时间后，芯片重复 RX 操作继续侦听后续传入的数据包。若接收到的是无效数据包，芯片只会重复 RX 操作但不会向 MCU 发出 RX 完成中断请求。MCU 发送 Light Sleep 命令给芯片可停止连续 RX 模式。在连续 RX 模式下，只可使用 Simple FIFO 模式。在 MCU 从 RX FIFO 读取数据之前，为了防止接收的数据包数据长度域因后续传入数据包而损坏，用户需将 RXPL2F_EN 和 PLLEN_EN 都设置为 1 从而将 PLEN 信息保存到 RX FIFO 中。由于 PLEN 域的存在，最大数据包数据长度变为 63 个字节。若 MCU 从 RX FIFO 读取数据之前有新传入的数据，则发生 FIFO 溢出错误，此时 RF 会将 RXERRIF 位置高并向 MCU 发出 RX 错误中断请求。此时，MCU 应该退出连续 RX 模式并将 RX FIFO 指针复位。

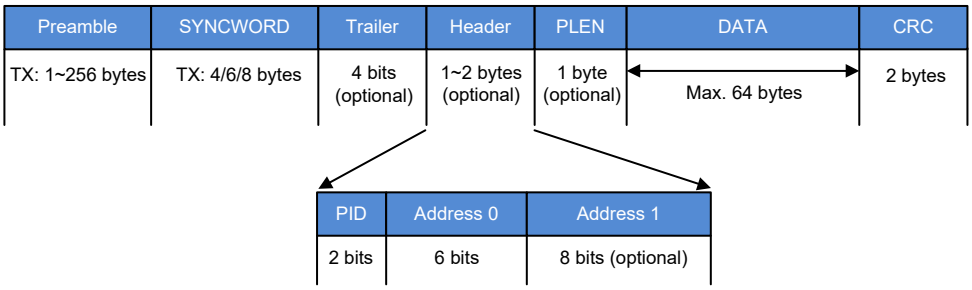


连续 RX 模式

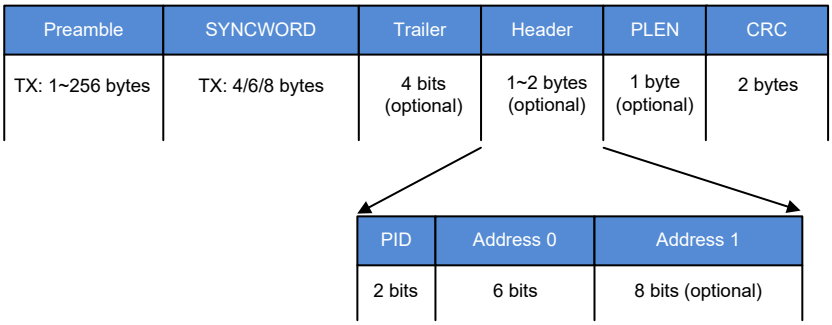
ARK 模式：自动重发和自动应答

RF 支持自动重发和自动应答机制，通过设置 ARK_EN 为 1 来使能。此机制支持简单的双向通信，但只可工作在 Simple FIFO 模式下。

设置 ARK_EN 为 1 使芯片进入自动重发和自动应答模式。当接收到来自 MCU 的 TX 命令时会触发自动重发功能，当收到 RX 命令时会触发自动应答功能。自动重发模式下主机发送给从机的数据包格式如下图所示。

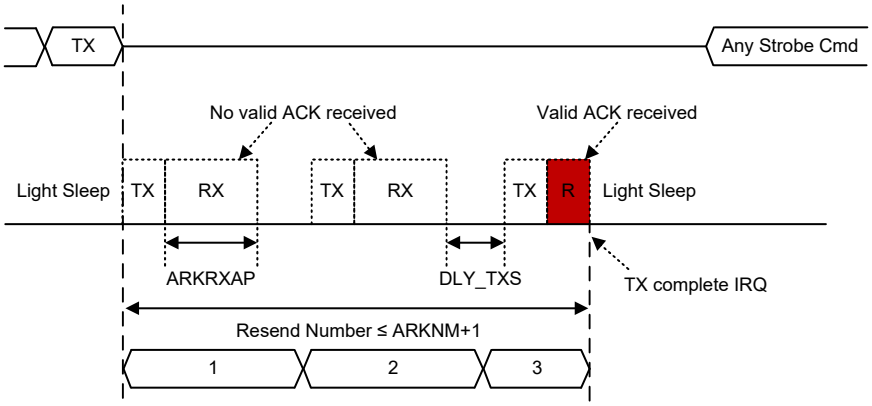


处于自动应答模式下的从机，使用下图所示的数据包格式发送应答数据包给主机。注意，此应答数据包不包含有效载荷数据域。

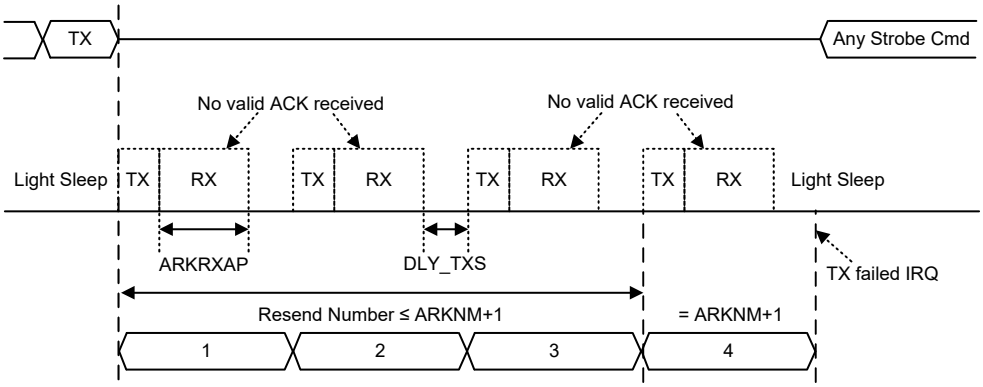


若在 ARK 模式下要使用地址域，则自动重发端 (主机) 设置的地址要与自动应答端 (从机) 的相同。

设置好 ARKNM[3:0]、ARK_EN 和 ARKRXAP[7:0] 之后，MCU 发出 TX 命令以开始自动重发进程。RF 开始发送 TX FIFO 中的数据并在 TX 操作完成后进入 RX 模式。RX 周期为 250μs 的倍数，此倍数等于 (ARKRXAP[7:0]+1)。若 RF 在 RX 周期内收到来自从机的 CRC 校验正确的有效应答包，芯片将返回 Light Sleep 模式并向 MCU 发送 TX 完成中断请求。否则，RF 会判断是否达到 ARKNM[3:0] 定义的自动重发次数。若未达到，芯片将进入 TX 模式继续发送相同的 TX 数据，此时自动重发次数加一。



自动重发：达到 ARKNM 限制次数前接收到 ACK 数据包



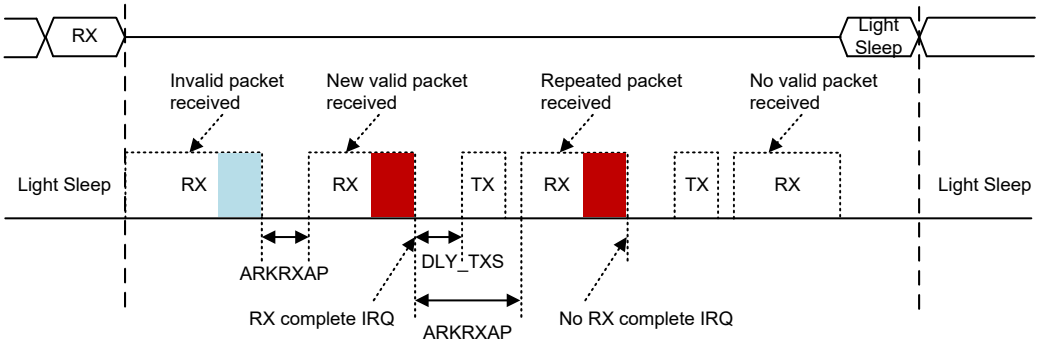
自动重发：达到 ARKNM 限制次数前未接收到有效数据包

从机自动应答方面，MCU 通过发送 RX 命令开启自动应答进程，通过发送 Light Sleep 命令停止自动应答进程。在自动应答模式下，从机端会启用 PID/

CRC 过滤功能以检查接收到的数据包。若新传入的数据包的 PID/CRC 与上一个数据包的 PID/CRC 相同，则新接收到的数据包将被视为重复的数据包。

在自动应答过程中，若接收到 PID/CRC 值不同且 CRC/ 地址检测正确的有效数据包，芯片会向 MCU 发出 RX 完成中断请求，并自动发送应答包给主机。若芯片接收到相同 PID/CRC 且 CRC/ 地址检测正确的数据包，会将其视为重复的数据包。接着，芯片不会向 MCU 发送 RX 完成中断请求但仍会自动发送应答包给主机。若芯片接收到的数据包 CRC/ 地址检测错误，不发送中断请求并且重新执行 RX 操作继续侦听后续数据包。

当前 RX 完成到重新下一次 RX 操作之间的时间间隔由 ARKRXAP[7:0] 决定。在一般应用里，MCU 接收到 RX 完成中断请求后需在此时间内从接收器 FIFO 读取数据。另外，在接收到 RX 完成中断请求后，MCU 若想退出 ARK 模式需等待同样一段时间。



自动应答进程

ATR 模式：自动发送 / 接收

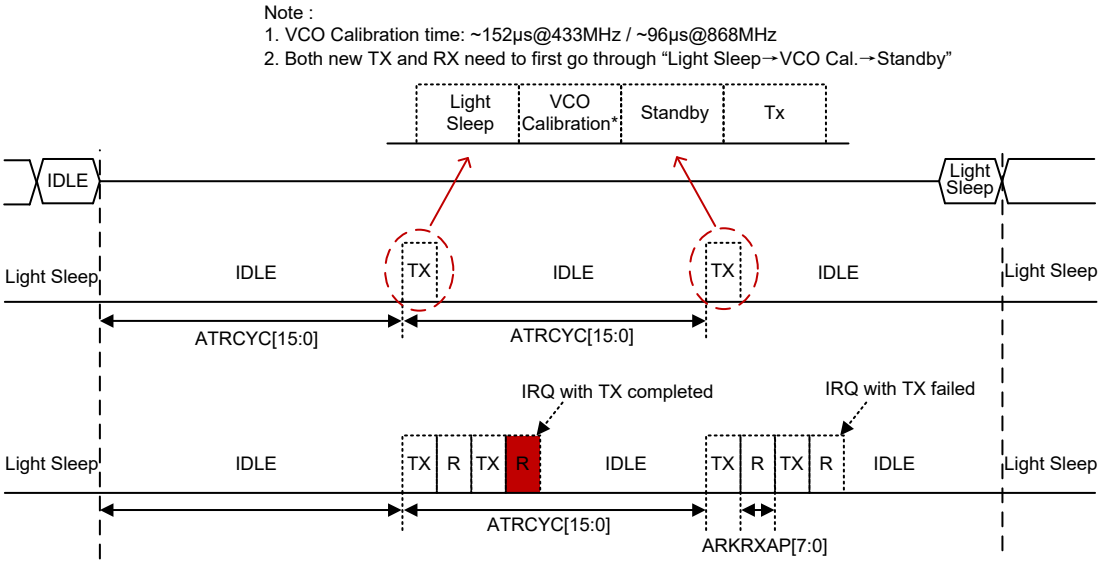
RF 支持特殊的 ATR 工作模式，可减少外部主机负载。此芯片含两种 ATR 功能，一种是 WOR，另一种是 WOT。这两种功能都只能工作在 Simple FIFO 模式下。这两种工作模式需要搭配使用一个以低频运行的 Idle 模式定时器。其低频时钟来自内部 LIRC 或外部 ROSCi 时钟，由 ATR1 寄存器中的 ATRCLKS 位选择。ATRCT 定时器有两种工作模式，通过 ATRCTM 位选择。若 ATRCTM 位清零则选择单次模式，此模式下当芯片进入 Idle 状态，每当发生 ATR 事务时 ATRCT 定时器重新启动，每当接收到 Light Sleep 命令时，ATRCT 定时器停止并退出 ATR 模式。若 ATRCTM 位置高则选择连续模式，此模式下一旦接收到 Idle 命令 ATRCT 定时器开始工作，并持续工作直到 ATR_EN 位或 ATRCTM 位清零。

进入 ATR 模式后，只有 Idle 命令、Light Sleep 命令、设置寄存器存储区命令和控制寄存器读 / 写命令可被 RF 识别。

WOT (从 TX 唤醒) 功能

当设置 ATR_EN 位为 1、ATRM[1:0] 为“00”使能 WOT 功能后，芯片将周期性地从 Idle 模式唤醒并在不与 MCU 互动的情况下发送 TX FIFO 中的数据。当接收到来自 MCU 的 Idle 命令时芯片开始 WOT 进程，当接收到来自 MCU 的 Light Sleep 命令时，芯片停止 WOT 进程。ATRCYC[15:0] 位用于设置 WOT 功能的唤醒周期。若达到定时器定时时间，唤醒定时器会触发芯片离开 Idle 状态并进入激活状态去发送数据，同时 ATRCYC[15:0] 的值会被载入定时器的计数器。完成 TX 操作后，芯片返回 Idle 模式并保持此状态直到下一次达到唤醒定时器定时时间。在激活状态下，芯片默认只执行一次唤醒传输。用户可结合 ARK 功能扩展唤醒传输机制。重复发送的次数由 ARK7 寄存器的 ARKNM[3:0]

位决定。在重复发送数据包的间隔内会插入一个 RX 时隙，此时隙由 ARK8 寄存器的 ARKRXAP[7:0] 位决定。若芯片在此时隙期间收到 ACK 信号，会向 MCU 发出 TX 完成中断请求。



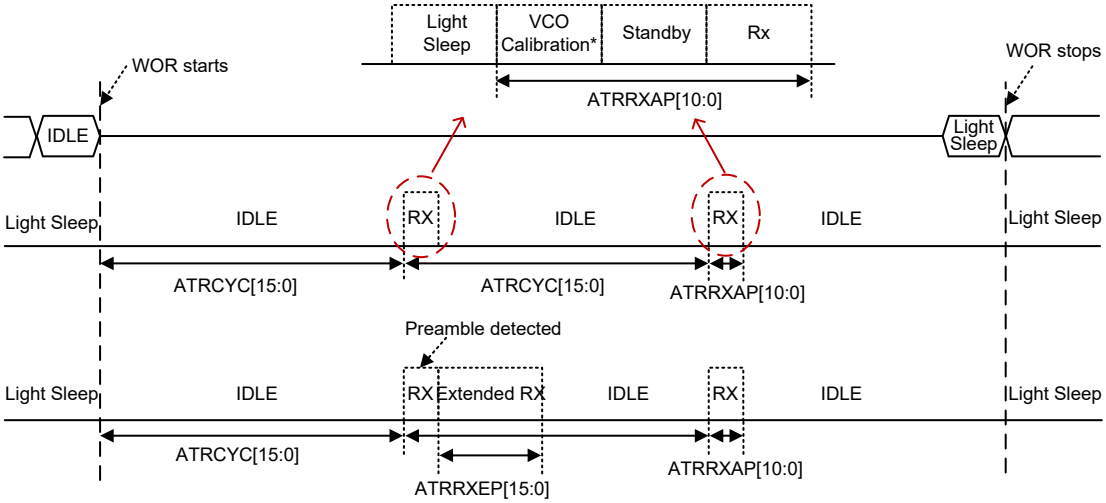
WOT 进程

WOR (从 RX 唤醒) 功能

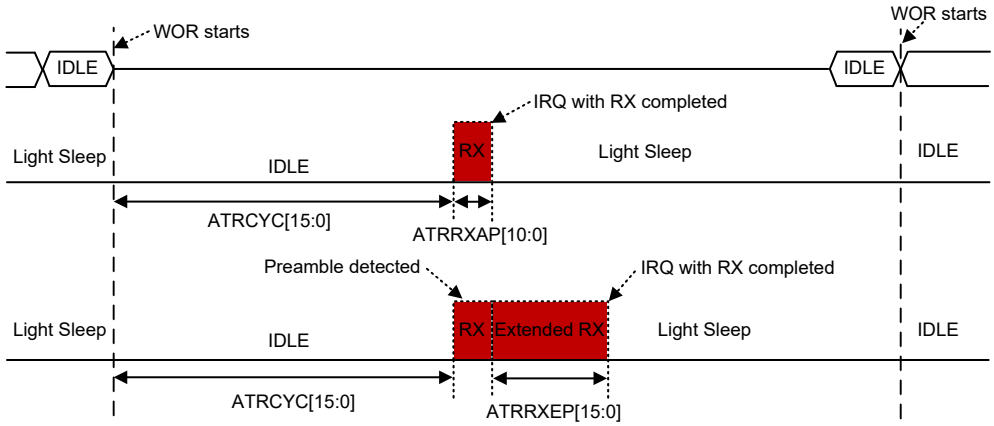
当设置 ATR_EN 位为 1、ATRM[1:0] 为“01”使能 WOR 功能后，芯片将周期性地从 Idle 模式唤醒并在不与 MCU 互动的情况下侦听传入的数据。当接收到来自 MCU 的 Idle 命令时芯片开始 WOR 进程，当接收到来自 MCU 的 Light Sleep 命令时，芯片停止 WOR 进程。ATRCYC[15:0] 位用于设置 WOR 功能的唤醒周期。若达到定时器定时时间，唤醒定时器会触发芯片离开 Idle 状态并进入激活状态去侦听传入的数据，同时 ATRCYC[15:0] 的值会被载入定时器的计数器。接收有效周期由 ATRRXAP[7:0] 位决定，是 250μs 的倍数，最少为 250μs。若在 RX 有效周期内未接收到数据包，芯片将返回 Idle 模式并等待下一轮 WOR 进程。

若检测到前导码，则有效周期将自动延长。延长时间由 ATRRXEP[15:0] 定义。延长时间是 250μs 的倍数，最少为 250μs。一旦接收到同步码，接收周期将会自动延长直到整个数据包被完全接收。当 RX 接收完成且 CRC 校验正确时，RF 会发送 RX 完成中断请求告知 MCU 并停留在 Light Sleep 模式。MCU 可从 RX FIFO 读取传入的数据并发出 Idle 命令从而开始新一轮 WOR 进程。若想退出 WOR 模式，MCU 还需发送 Light Sleep 命令给 RF。

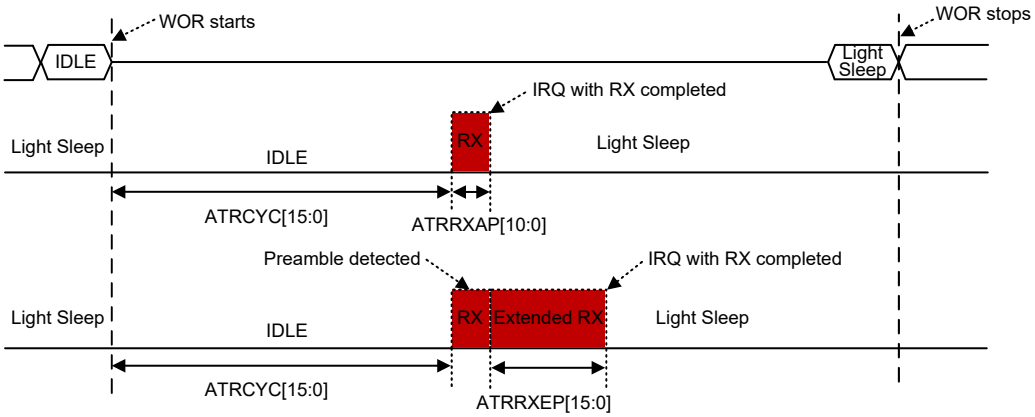
Note :
1. VCO Calibration time，433MHz(~152us)/868MHz(~96us)
2. Both new TX and RX need to first go through “Light Sleep→VCO Cal.→Standby”



WOR – 未接收到传入数据

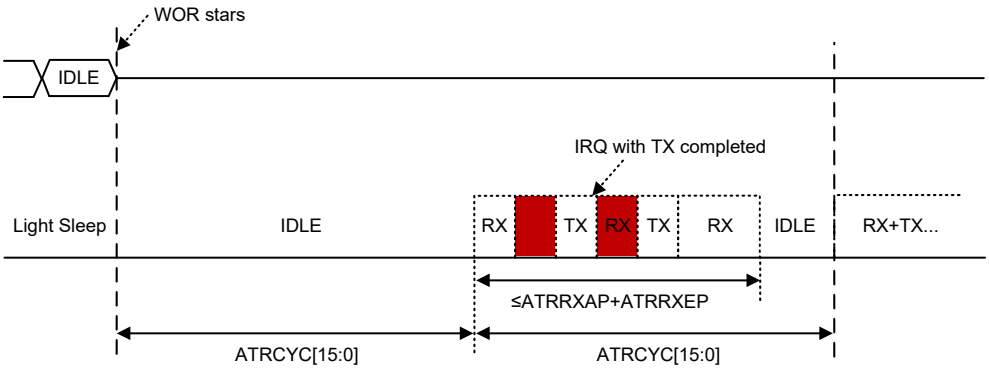


WOR – 接收到传入数据



接收到传入数据后停止 WOR

在 WOR 有效周期内，芯片默认只执行一次 RX 操作。用户可结合 ARK 功能扩展唤醒接收机制。在 WOR+ARK 模式下，在重复接收数据包的间隔内会插入一个 TX 时隙作为应答。TX 持续时间取决于发送数据速率。芯片保持 RX 模式的最大时间由 ATRRXAP 和 ATRRXEP 共同决定。若在达到定时器定时时间之前接收到 CRC 校验正确且包含不同 PID/CRC 值的有效数据包，芯片会发送一个 RX 完成中断请求给 MCU 并自动进入 TX 模式。若接收到 CRC 校验正确但 PID/CRC 值相同的重复数据包，芯片只会自动进入 TX 模式，不会向 MCU 发送中断请求。当 TX 操作完成后，芯片将再次返回 RX 模式并侦听传入的数据包直到无数据包传入为止。



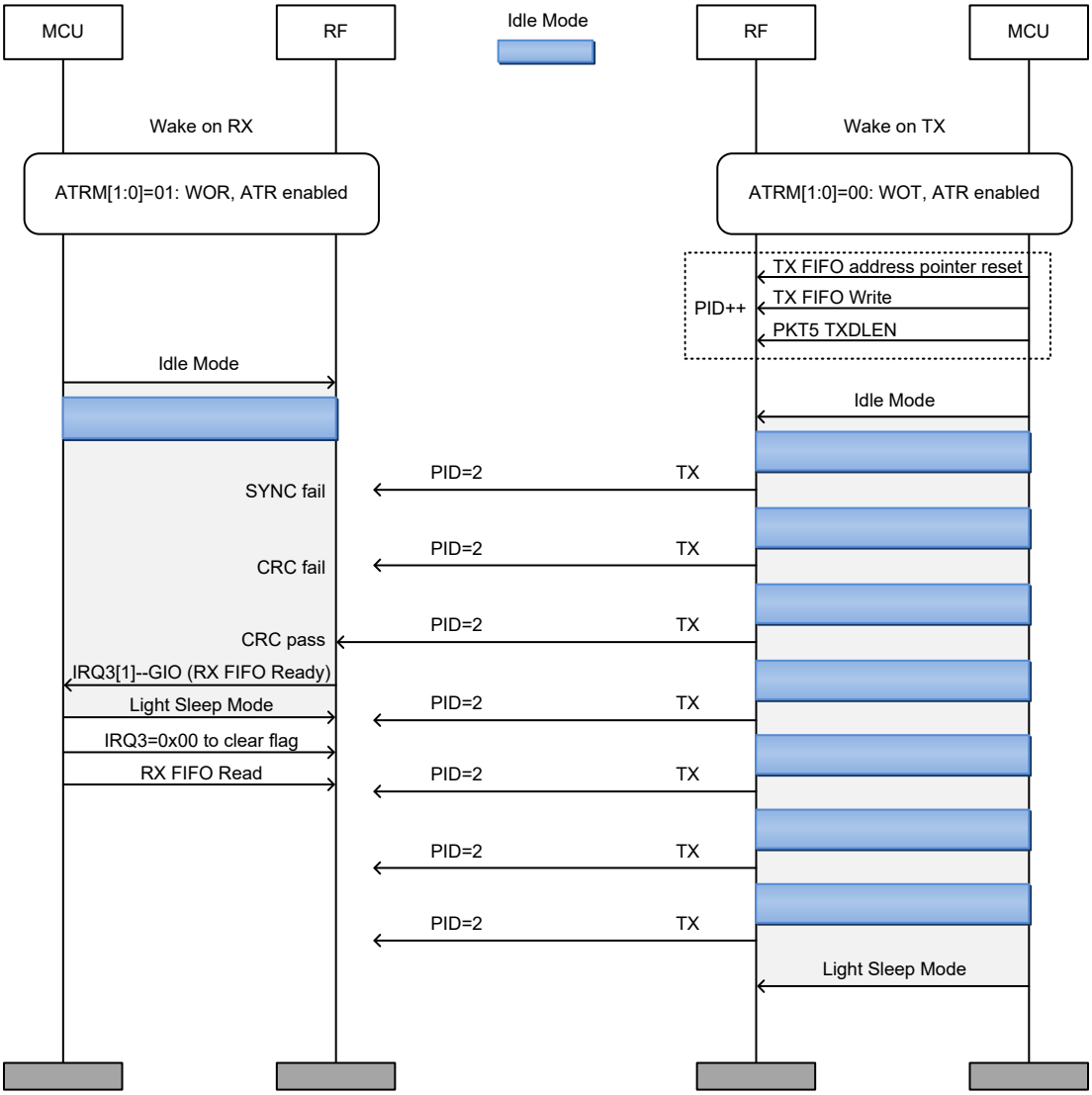
WOR+ARK 进程

WTM (唤醒定时模式)

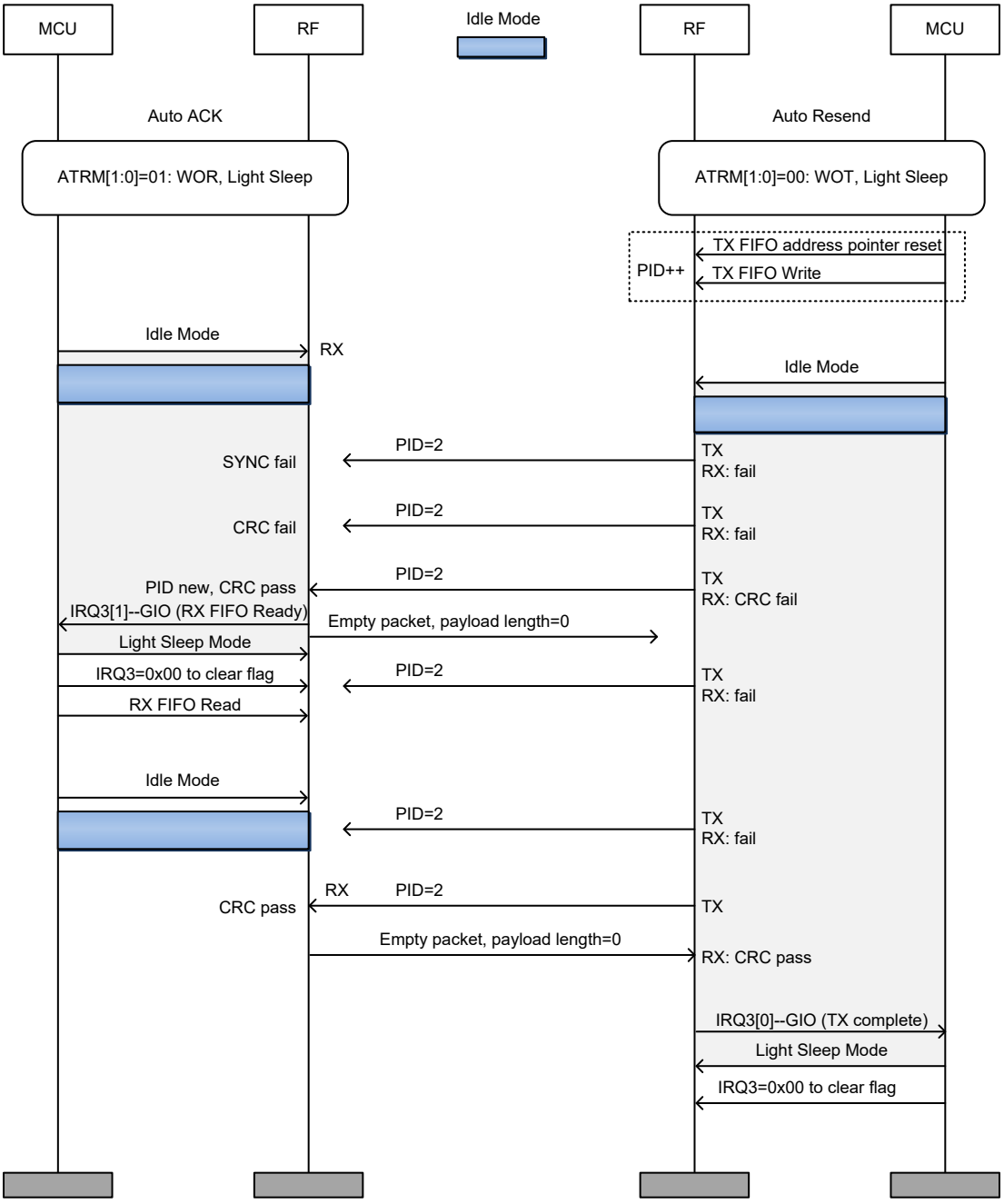
RF 可被设置作为一个可编程定时器从 GIO 口输出周期性波形。用户可使用此信号唤醒 CPU。设置 ATR_EN 为 1 和 ATRM=10/11 以使能 WTM 模式。当接收到来自 MCU 的 Idle 命令时芯片开始 WTM 进程，当接收到来自 MCU 的 Light Sleep 命令时，芯片停止 WTM 进程。在整个 WTM 进程内芯片都处于 Idle 模式。

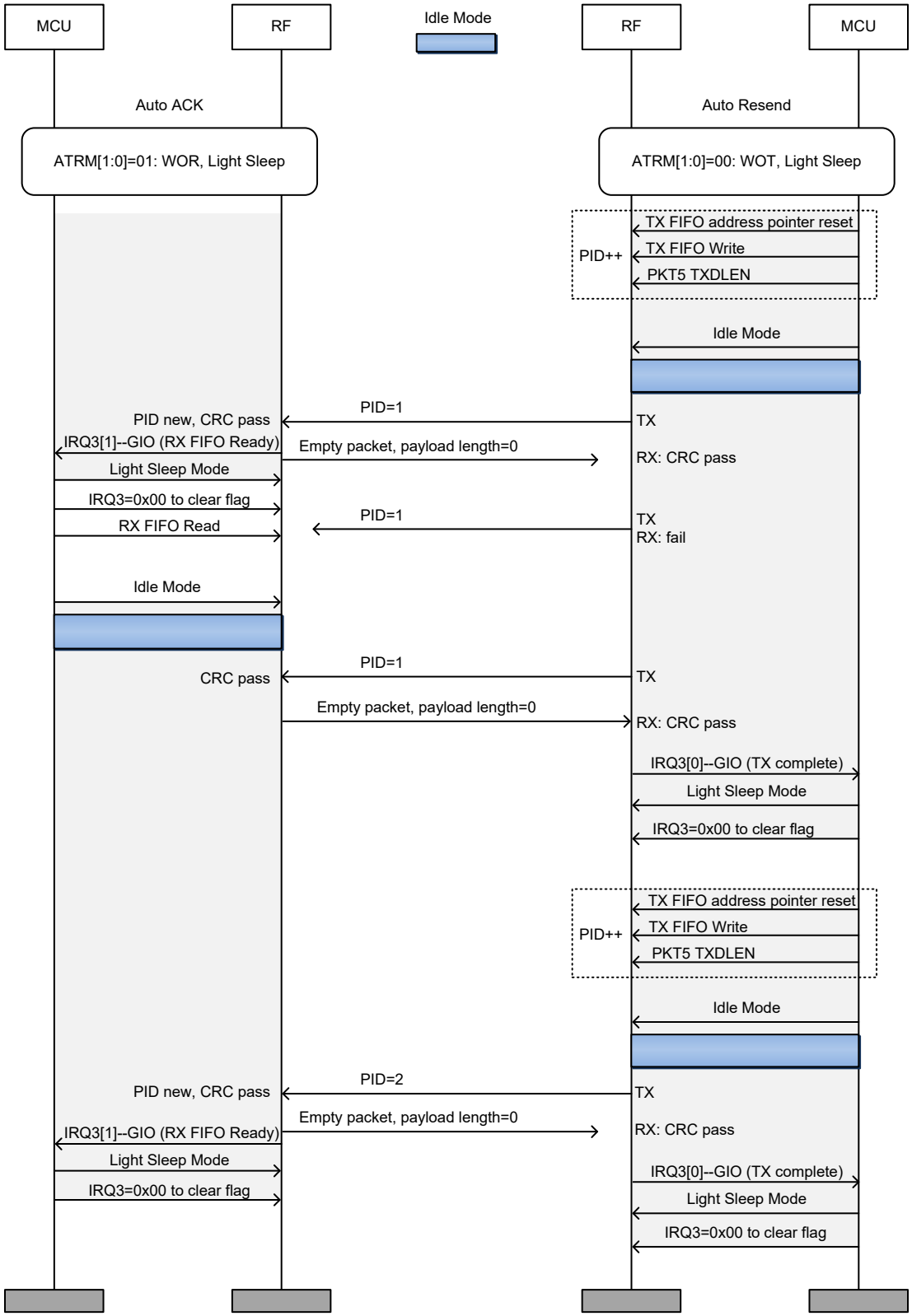
信息流程范例

ATR：WOT & WOR

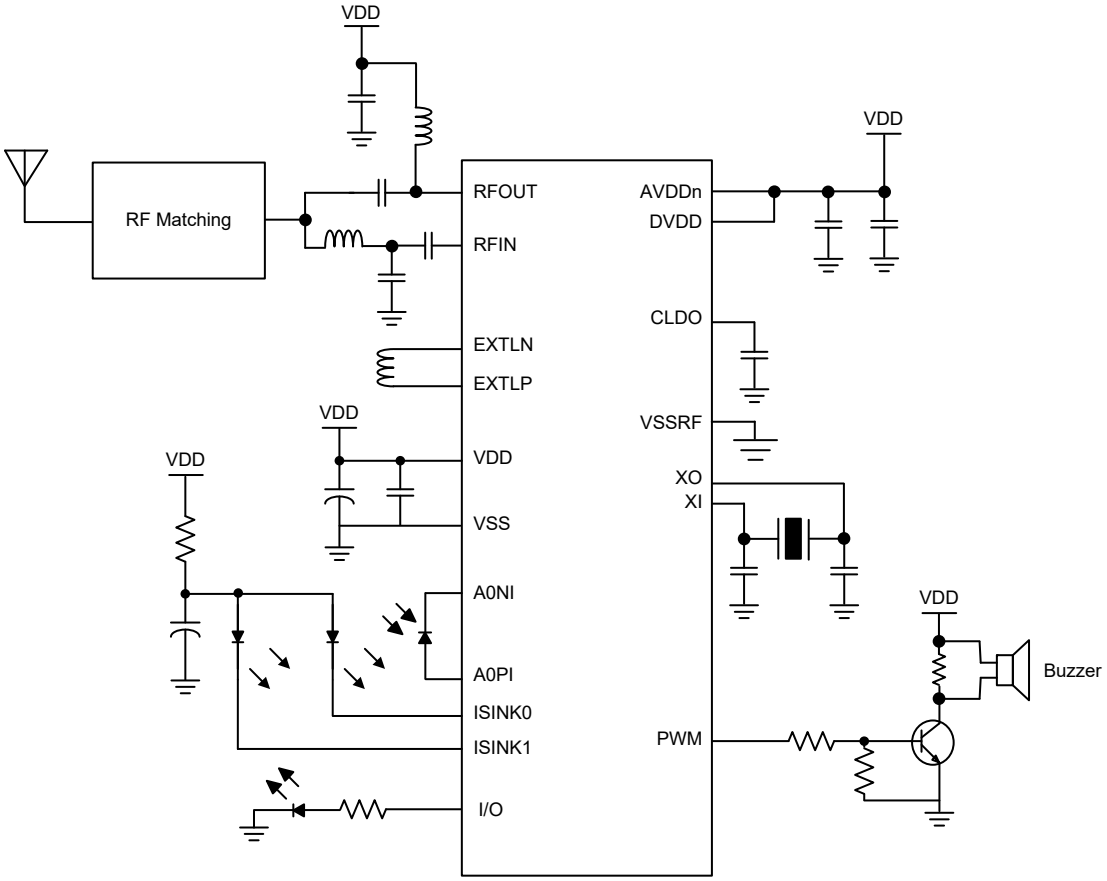


ATR+ARK：WOT + 自动重发 & WOR + 自动应答




MCU
RF
Idle Mode
RF
MCU
Auto ACK
ATRM[1:0]=01: WOR, Light Sleep
Auto Resend
ATRM[1:0]=00: WOT, Light Sleep
TX FIFO address pointer reset
PID++
TX FIFO Write
PKT5 TXDLEN
Idle Mode
PID=1
TX
PID new, CRC pass
IRQ3[1]--GIO (RX FIFO Ready)
Empty packet, payload length=0
RX: CRC pass
Light Sleep Mode
IRQ3=0x00 to clear flag
PID=1
TX
RX FIFO Read
RX: fail
Idle Mode
PID=1
TX
CRC pass
Empty packet, payload length=0
RX: CRC pass
IRQ3[0]--GIO (TX complete)
Light Sleep Mode
IRQ3=0x00 to clear flag
TX FIFO address pointer reset
PID++
TX FIFO Write
PKT5 TXDLEN
Idle Mode
PID=2
TX
PID new, CRC pass
Empty packet, payload length=0
IRQ3[1]--GIO (RX FIFO Ready)
RX: CRC pass
Light Sleep Mode
IRQ3[0]--GIO (TX complete)
IRQ3=0x00 to clear flag
Light Sleep Mode
IRQ3=0x00 to clear flag

应用电路


VDD
RF Matching
RFOUT
RFIN
EXTLN
EXTLP
VDD
VSS
A0NI
A0PI
ISINK0
ISINK1
I/O
AVDDn
DVDD
CLDO
VSSRF
XO
XI
PWM
Buzzer

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 $[m] \leftarrow ACC - [m] - \overline{C}$

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

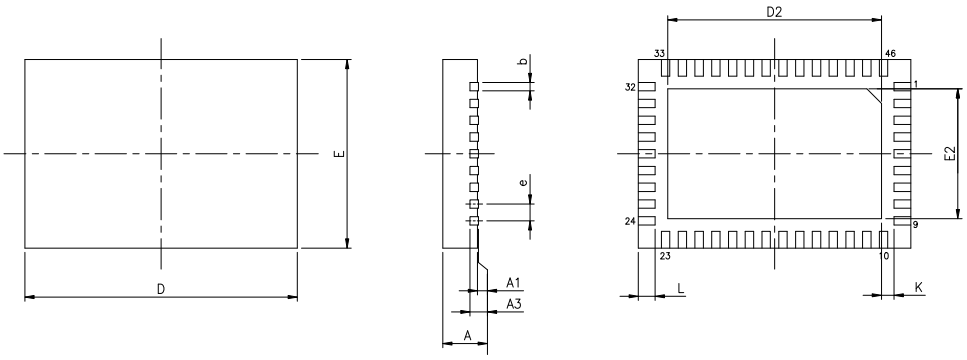
封装信息

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

SAW Type 46-pin QFN (6.5mm×4.5mm×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	—	0.256 BSC	—
E	—	0.177 BSC	—
e	—	0.016 BSC	—
D2	0.199	0.201	0.203
E2	0.120	0.122	0.124
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	—	6.50 BSC	—
E	—	4.50 BSC	—
e	—	0.40 BSC	—
D2	5.05	5.10	5.15
E2	3.05	3.10	3.15
L	0.35	0.40	0.45
K	0.20	—	—

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

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