LGT8F08A/04A/02A/01A

高性能低功耗 8 位 FLASH 微控制器

编程手册

LOGICGreen Technologies
Shenzhen, CHINA
2012-1-11

MBX: biz@mcugreen.com
www.mcugreen.com

产品特性

高性能，低功耗的 8 位 MVR8X 微处理内核

先进的 RISC 构架

- 3 级流水线设计
- 131 条指令，大多数指令执行时间为单个时钟周期
- 32 个 8 位通用工作寄存器
- 工作于 16MHz 时性能高达 16MIPS
- 单周期的硬件乘法器(8x8)

非易失性程序和数据存储器

- 8/4/2/1K 字节系统内可编程 FLASH
- 数据保护功能
- 504 字节数据 FLASH， 支持字节读写（EEPROM）
- 256/512 字节片内 SRAM
- 独立的用户数据区实现系统配置功能

SWD 双线调试接口

- 支持扩展的片内在线调试功能
- 通过 SWD 接口实现对 FLASH, EEPROM，系统配置区的编程

外设特点

- 8 通道 10bit 250KSPS 模数转换器(ADC)
- 2 通道模拟比较器，~~支持 ADC 通道输入功能~~
- 一个具有独立预分频和比较器功能的 8 位定时器/计数器
- 一个具有预分频器，比较器功能和捕捉功能的 16 位定时器/计数器
- 三通道 PWM 脉宽调制控制器
- 可编程的串行 USART
- 可工作于主/从模式的 SPI 串行接口
- 可编程看门狗定时器
- 最多 25 个可编程 I/O （LGT8F08A）

特殊的处理器特点

- 每个芯片具有独立的 32 位 GUID
- 具有掉电保护功能的片内 POR
- ±1%精度 16MHz 内部低温漂 RC 振荡器
- 片内/片外中断源
- 1KHz 低功耗 RC 实现更低的待机功耗
- 4 种睡眠模式，内部电源设计实现 uA 级待机功耗，可通过外部专用 I/O 或内部 1KHz RC 唤醒

封装类型

- SOP28L/SOP24L/SOP20L/SOP14L/SOP8L

工作电压

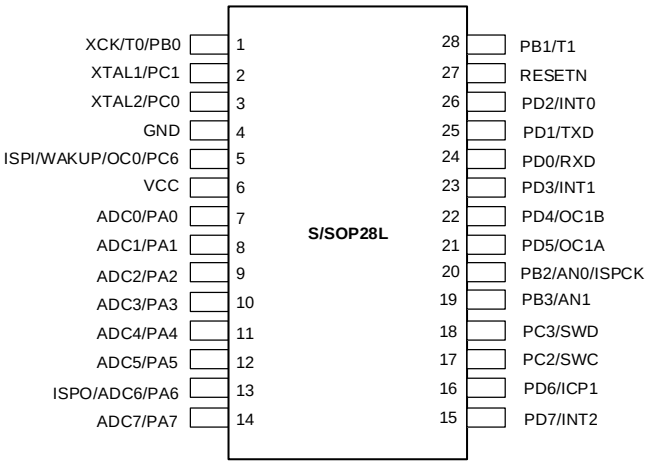
- 1.8V ~ 3.6V

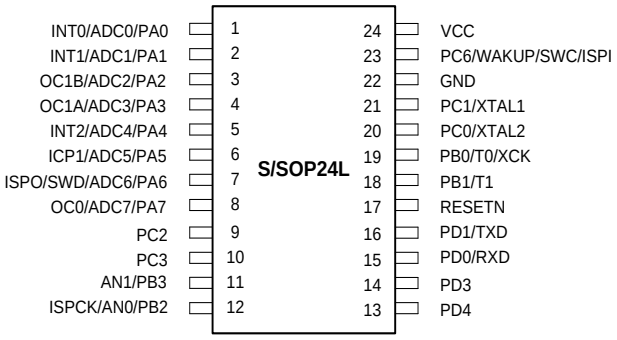
速度等级

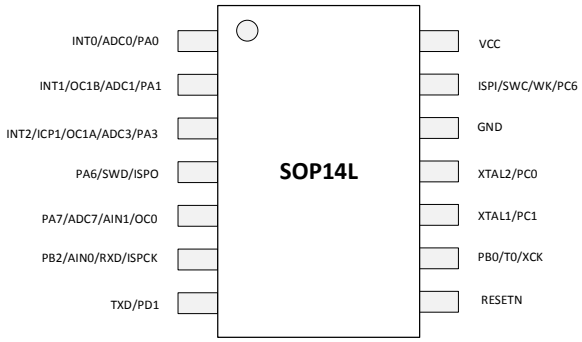
- 0 ~ 16MHz @1.8V ~ 3.0V
- 0 ~ 25MHz @3.0V ~ 3.6V

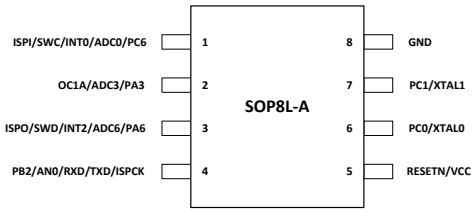
引脚配置

S/SOP 类型封装定义







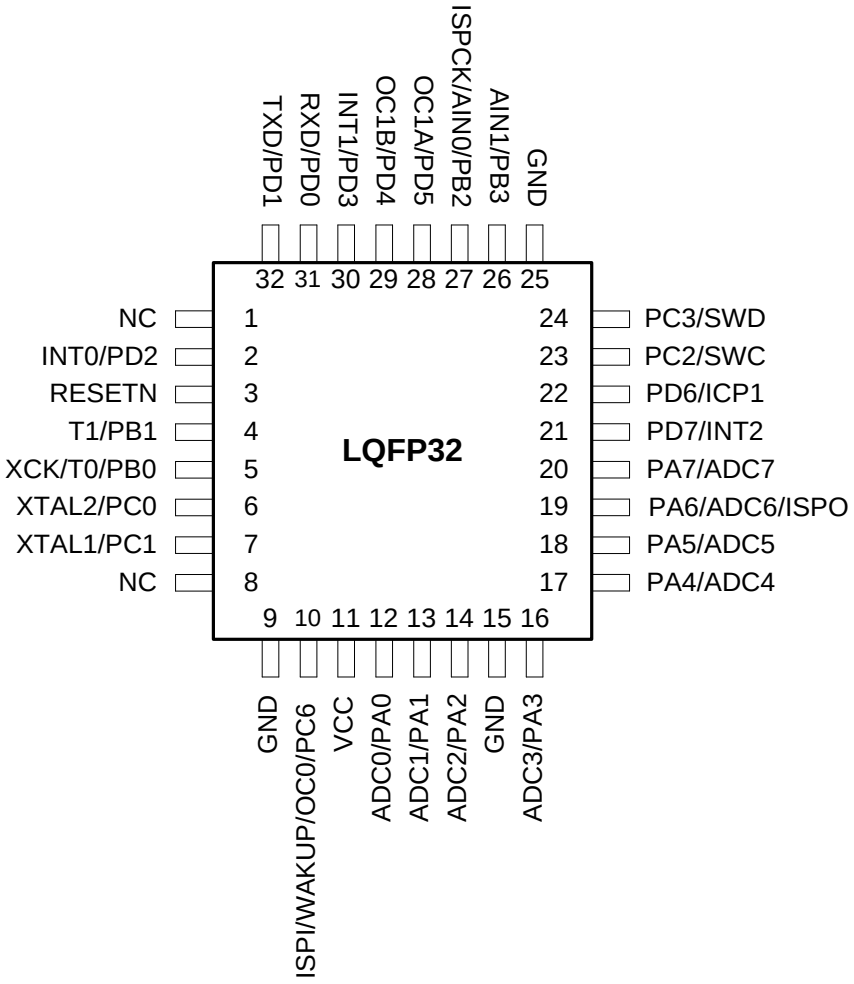


注意事项：

1. *对于 S/SOP24L 封装的 LGT8F08A 系列，PC2/3 以及 PD3/4 只能作为输出 I/O*
2. *SOP8L-A 在以 S/SOP24L 为基础封装，功能配置和 S/SOP24 一致。其中 SOP8L-A 的一些引脚是 S/SOP24L 中几个引脚封装到一起的结果。下面表格说明 SOP8L 和 S/SOP24L 引脚直接的对应关系，请在使用时特别注意：*

SOP8L-A	*S/SOP24L*
PIN1	*PIN1 + PIN23*
PIN3	*PIN5 + PIN7*
PIN4	*PIN12 + PIN15 + PIN16*
PIN5	*PIN17*

LQFP32 封装定义


LQFP32
1 NC
2 INT0/PD2
3 RESETN
4 T1/PB1
5 XCK/T0/PB0
6 XTAL2/PC0
7 XTAL1/PC1
8 NC
9 GND
10 ISPI/WAKUP/OC0/PC6
11 VCC
12 ADC0/PA0
13 ADC1/PA1
14 ADC2/PA2
15 GND
16 ADC3/PA3
17 PA4/ADC4
18 PA5/ADC5
19 PA6/ADC6/ISPO
20 PA7/ADC7
21 PD7/INT2
22 PD6/ICP1
23 PC2/SWC
24 PC3/SWD
25 GND
26 AIN1/PB3
27 ISPCK/AIN0/PB2
28 OC1A/PD5
29 OC1B/PD4
30 INT1/PD3
31 RXD/PD0
32 TXD/PD1

引脚说明

端口	功能说明
VCC	3.3V 电源输入
GND	地
端口 A (PA7...PA0)	端口 A 为 8 位双向 I/O 口，具有可编程的内部上拉电阻。其输出缓冲器具有对称的驱动特性，可以输出和吸收大电流。作为输入使用时，若内部上拉电阻使能，端口被外部电路拉低时将输出电流。在复位过程中，即使系统时钟还未起振，端口 A 处于高阻状态。
端口 B (PB3...PB0)	端口 B 为 4 位双向 I/O 口，具有可编程的内部上拉电阻。其输出缓冲器具有对称的驱动特性，可以输出和吸收大电流。
端口 C (PC6,PC3…PC0)	端口 C 为 5 位双向 I/O 口，具有可编程的内部上拉电阻。其输出缓冲器具有对称的驱动特性，可以输出和吸收大电流。作为输入使用时，若内部上拉电阻使能，端口被外部电路拉低时将输出电流。在复位过程中，即使系统时钟还未起振，端口 C 处于高阻状态。
端口 D (PD7...PD0)	端口 D 为 8 位双向 I/O 口，具有可编程的内部上拉电阻。其输出缓冲器具有对称的驱动特性，可以输出和吸收大电流。作为输入使用时，若内部上拉电阻使能，则端口被外部电路拉低时将输出电流。
RESETN	复位输入引脚。持续时间超过最小门限时间的低电平将引起系统复位。持续时间小于门限间的脉冲不能保证可靠复位
XTAL1	反向振荡放大器与片内时钟操作电路的输入端
XTAL2	反向振荡放大器的输出端
ADC7...ADC0	0~VCC 模拟 ADC 输入，当使用模拟输入功能时，对应的 PA 端口应该设置为输入高阻，关闭上拉电阻，避免数字模块对模拟输入的干扰。
AN1…AN0	0~VCC 模拟比较器外部输入，当使用外部模拟比较器输入时，对应的 PB3/PB2 端口应设置为输入高阻，关闭上拉电阻，避免数字模块对模拟输入产生干扰。
SWD **SWC**	双线串行调试接口数据线 双线串行调试接口时钟线 用户可以通过寄存器配置或全局配置位关闭 SWD 接口
TXD	UART 数据输出引脚，同步 UART 模式（SPI）数据输出
RXD	UART 数据输入引脚，同步 UART 模式（SPI）数据输入
XCK	同步 UART 模式（SPI）时钟，主模式为时钟输出，从模式为时钟输入。
T1	定时器 1 的外部时钟输入
ICP1	定时器 1 输入俘获
OC1B	定时器 1 比较器 A 输出（PWM 输出）
OC1A	定时器 1 比较器 B 输出（PWN 输出）
T0	定时器 0 外部时钟输入
OC0	定制器 0 比较器输出（PWM 输出）
INT1…INT0	外部中断输入
WAKUP	掉电模式下的专用唤醒引脚
ISPI	ISP 编程数据输入引脚
ISPO	ISP 编程数据输出引脚
ISPCK	ISP 编程时钟 ISP 需要与 RESETN 复位输入一同配合工作，拉低 RESETN 进入 ISP 模式 拉高 RESETN 后，ISP 模块被禁止，系统进入正常工作模式

重要说明

关于工程样片的不同

LGT8F0XA 系列第一版 LGT8F08A-16C 为前期工程样片，封装为 SOP28L。该工程样片与其他所有 LGT8F0XA 后续型号之间有一点使用上的差别，主要是外部晶振 I/O 的控制寄存器定义不同，请参考“系统时钟与功耗管理”章节中关于 **PMCR** 寄存器定义中对 **OSCMEN** 控制位的描述。

关于 SBIC/SBIS 指令的使用

由于 LGT8F0XA 的 I/O 端口为异步设计，所以可靠的使用方法是首先把端口的数据读入到内部寄存器，然后根据寄存器的值进行后续处理。所以不建议直接在端口寄存器上使用 SBIC/SBIS 指令。汇编模式下，用户可以使用 SBRC/SBRS 指令代替 SBIC/SBIS 指令。对于基于 C 语言的应用， 请参考“通用 I/O 端口”章节中关于“I/O 应用注意事项”的说明。

关于堆栈指针的初始化

LGT8F08A/04A 系列内部包含 512 字节 SRAM，地址映射为 0x100 ~ 0x2FF
LGT8F02A/01A 系列内部包含 256 字节 SRAM，地址映射为 0x100 ~ 0x1FF
用户需要注意在程序运行前正确的初始化内部堆栈指针寄存器(SP)。对于 LGT8F08A/04A 系列，使用 M164 兼容模式编译程序，SP 指针默认初始化为 0x4FF，这个值对于 LGT8F08A/04A 通用有效，内部偏移地址会保证 SP 指针指向 SRAM 的最高地址处，用户可以使用这种默认设置。对于 LGT8F02A/01A，需要用户使用程序代码或配置编译环境的方式正确的初始化 SP 指针到有效的 SRAM 空间。

关于 I/O 端口的复位状态

LGT8F0XA 系列 I/O 端口在复位阶段都默认为 GPIO 功能，全部复位为输入状态，并关闭内部上拉电阻。需要用户注意的是 PB0 (只针对 S/SOP28/24/20 封装) 在复位过程中作为 ISP 的片选引脚，内部上拉是打开的。

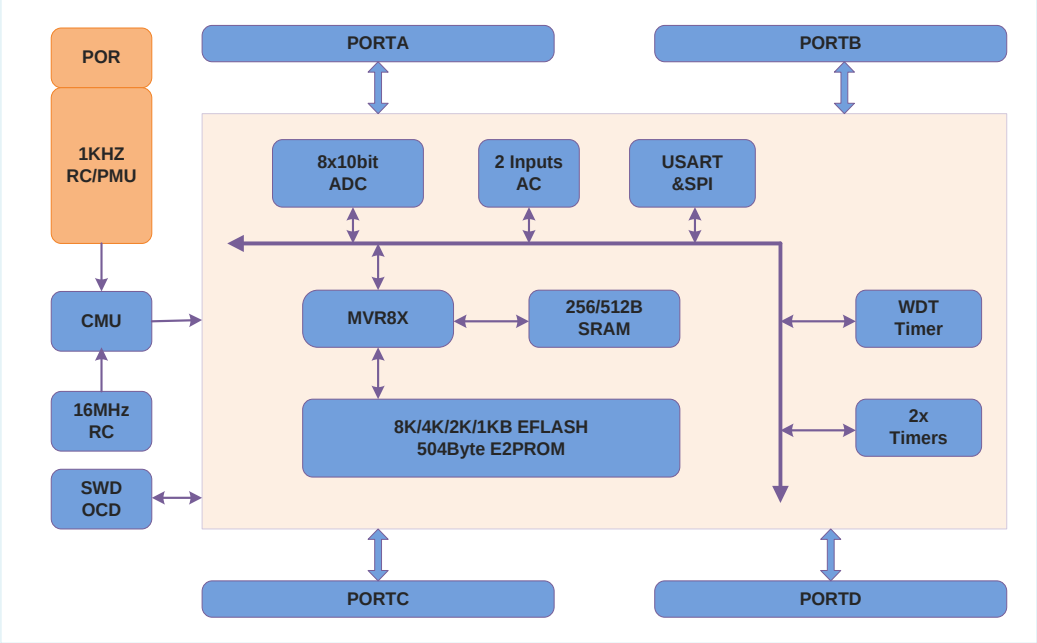
系统概述

LGT8F0XA 是基于增强的 8 位低功耗 MVR8X RISC 三级流水线构架设计。由于其先进的指令集以及单时钟周期指令执行时间，LGT8F0XA 的数据吞吐率高达 1MIPS/MHz，从而可以平衡系统在功耗和处理速度之间的矛盾。关于 MVR8X RISC 构架的更多细节，请参考本文档 MVR8X 内核介绍部分。

LGT8F0XA 引入的独特的电源设计方法，从而在系统待机功耗方面比同类产品有更加优异的表现，系统中为低功耗设计集成一个内部 1KHz RC 振荡器和电源管理模块，能够在系统空闲的时候由软件选择进入待机模式，在最高级别的待机模式下，电源管理系统将关闭系统工作电源，仅保持 1KHz RC 和电源管理模块的工作，从而实现微安级的待机功耗。

LGT8F0XA 基于 EFLASH 工艺设计，可以提供 8K/4K/2K/1K 四种不同的型号供选择，LGT8F0XA 集成了 504 字节内部数据 FLASH 以及 EEPROM 控制器单元，实现一个更加简易的 EEPROM 访问接口。同时 LGT8F0XA 也集成了 256/512 字节的 SRAM，可以让用户在实现不同应用时有更多的灵活性。

Figure3: 系统构架图



MVR8X 内核

MVR8X 内核具有丰富的指令集和 32 个通用工作寄存器，其中 R26~R31 可组合为三个 16 位通用寄存器 X/Y/Z。MVR8X 内部集成一个单周期的 8X8 乘法器，可以胜任简单的数据运算；MVR8X 针对中断响应进行了特别的优化，可以在中断发生后 3 个周期内进入中断服务程序，中断完成后，仅需 2 个周期便可从中断返回。MVR8X 同时也对 I/O 控制系统进行了更大的优化，并具有针对 I/O 直接操作的指令，可以仅仅使用一条指令，一个周期完成对单个 I/O 或一组 I/O 的读、写操作，这些特点使得 MVR8X 构架的 MCU 比同类型任何 MCU 更加适合控制类的应用。MVR8X 内核实现了片上调试功能，用户可以通过双线 SWD 接口以及专用的 USB 调试器，配合业界成熟的开发环境，轻松的实现产品的研发与调试。

存储单元

LGT8F0XA 系列 MCU 最多集成了 8KB EFLASH，504 字节的数据 FLASH，可以实现 E2PROM 功能。用户可通过 ISP 在线编程工具实现对 FLASH 的读写访问。LGT8F0XA 中集成了 E2PROM 接口控制逻辑，用户可以像访问 SRAM 一样访问 E2PROM 功能，提高了读写以及擦除操作的效率，同时也减少了实现 E2PROM 功能所需的代码量。

数据保护	LGT8F0XA 实现为保护用户程序代码实现了接口加密功能，用户在编程完成后，可以通过设置 LOCK 位，禁止 ISP 以及 SWD 接口访问 EFLASH 以及 E2PROM 的功能，LOCK 位禁止 ISP 后，必须通过整片擦除操作才能恢复 FLASH 以及 E2PROM 区域的读写操作。
时钟源	LGT8F0XA 内部集成了一个低温漂，误差±1%的 16MHz RC 振荡器，配合内部分频器，可以为系统运行提供 16MHz, 8MHz, 4MHz, 2MHz, 1MHz 最低至 125K 的 8 种运行频率，满足不同应用的需求，节省了外部晶振。同时内部也集成了一个低功耗的 1KHz RC 振荡器，可以在低功耗模式下维持系统的工作，用户可选择关闭 16MHz RC 振荡器，这样可以在系统运行的同时，得到更低的功耗。对于特殊的需求，LGT8F0XA 也支持外部晶振，这样用户可以禁用内部 16MHz RC 振荡器，完全依赖外部晶振工作。
模数转换器 **模拟比较器**	LGT8F0XA 全系列都集成了一个 10bit 250KSPS 采用率的 SAR-ADC 模数转换器，通过配置内部的 ADC 转换控制器，可以实现非常灵活的自动触发转换功能。LGT8F0XA 全系列内部也实现了一个两通道的模拟比较器，可以高速精准的判断两路模拟输入电压的细微差别，对一些需要快速电压检测的电路十分有效。LGT8F0XA 内部有一个 1.25V 的参考电压源，可以为 SAR-ADC 以及模拟比较器提供内部参考输入。模拟比较器的输入可以为两路专用的外部输入~~，也可以来自 SAR-ADC 的模拟输入，这样可以十分方便的实现对两通道模拟比较器的更多通道扩展，满足更为复杂的应用~~。
串行控制器 **SPI 控制器**	LGT8F0XA USART 是一个通用的串行控制器，支持通用的 PC 串口协议，可以通过串口实现与 PC 以及其他 UART 外设之间的通讯，USART 同时也支持并行模式，在并行模式下，用户可以实现 SPI 协议，通过相关的寄存器配置，选择并行模式下的 SPI 工作于主模式或从模式。通过 SPI 接口，用户可以实现对更多外设的兼容。
定时器 **PWM**	LGT8F0XA 实现两个多功能定时器，分别具有独立的定时预分频器，可以保证两个定时器的同时独立工作。定时计数器宽度分别为 8 位，16 位；可以满足不同的应用需求。定时器实现了通用的输入俘获，比较器输入等功能。通过对定时器的配置，可以轻松的实现三路 PWM 脉宽调制输出。为实现 PWM 相关的控制器算法提供了更加实用的解决方案。
看门狗 **定时器**	LGT8F0XA WDT 是一个 16 位宽的看门狗专用定时器，可以通过预分频实现从 1ms 到 512ms 的宽范围复位间隔宽度。MVR8X 内核实现了一个专门用于 WDT 复位的 WDR 指令，用户可以使用 WDR 指令方便的进行‘喂狗’操作。
可编程 I/O	LGT8F0XA 的端口中除去 2 个电源引脚，所有的其他 I/O 都可以工作在 GPIO 模式下，配合 MVR8X 独有的高效 I/O 操作指令，可以让用户用更少的代码，实现复杂的设计，这是其他同类 MCU 所不具备的。 LGT8F0XA 具有一整套编程与系统开发工具，包括：C 语言编译器，宏汇编，程序调试器/软件仿真器，以及评估版。

MVR8X 内核简介

本节从总体上讨论 MVR8X 内核的结构。CPU 的主要任务是保证程序的正确执行。因此它必须能够访问存储器、执行运算、控制外设以及处理中断。MVR8X 构架基于优化的 3 级流水线结构，在中断响应，跳转指令效率上有相当优秀的表现，MVR8X 流水线也针对程序空间以及数据空间的访问上做了更多的优化，保证了 90%以上的指令可以在一个周期完成，大部分常用的跳转，程序调用，以及所有的存储空间访问指令，也都是在一个周期内完成。而其他少数条件跳转指令也仅仅是最多在两个周期内完成。整体性能上，在同等频率下，MVR8X 内核比基于 1T C51 构架的 MCU 在执行效率上高出 90%以上。具体指令的执行周期信息，请参考本资料相关部分。

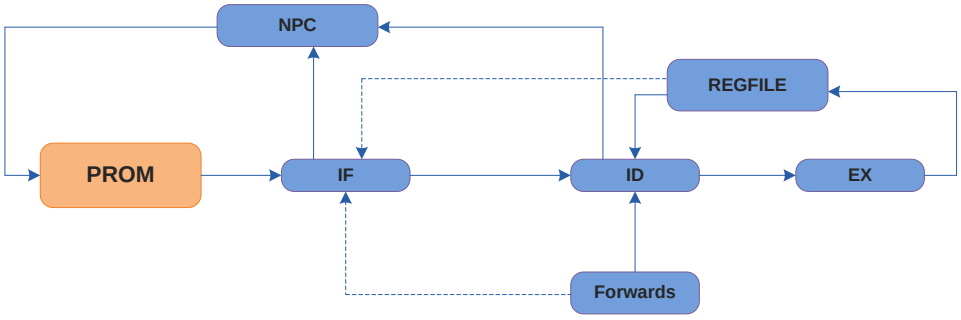


Figure 3. MVR8X 结构图

MVR8X

为了获得最高的性能以及并行性， MVR8X 采用了哈佛结构，具有独立的数据和程序总线。CPU 在执行一条指令的同时读取下一条指令。这个概念实现了指令的单时钟周期运行。程序存储器是可以在线编程的 FLASH。

快速访问寄存器文件包括 32 个 8 位通用工作寄存器，访问时间为一个时钟周期。从而实现了单时钟周期的 ALU 操作。在典型的 ALU 操作中，两个位于寄存器文件中的操作数同时被访问，然后执行运算，结果再被送回到寄存器文件。整个过程仅需一个时钟周期。寄存器文件里有 6 个寄存器可以组合为 3 个 16 位的间接寻址寄存器指针以寻址整个数据空间，实现高效的地址运算。其中一个指针还可以作为程序存储器查询表的地址指针。这些附加的功能寄存器即为 16 位的 X、Y、Z 寄存器。

ALU 支持寄存器之间以及寄存器和常数之间的算术和逻辑运算。ALU 也可以执行单寄存器操作。运算完成之后状态寄存器的内容得到更新以反映操作结果。

程序流程通过有/无条件的跳转指令和调用指令控制，从而直接寻址整个程序空间。大多数指令长度为 16 位，亦即每个程序存储器地址都包含一条 16 位或 32 位的指令。

在中断和调用子程序时返回地址的程序计数器(PC)保存于堆栈之中。堆栈位于内部数据 SRAM，因此其深度仅受限于 SRAM 的大小。在系统初始化过程中，用户首先要初始化堆栈指针寄存器 SP。这个寄存器位于 I/O 空间，可以进行读写访问。数据 SRAM 支持 5 种不同的寻址模式。

中断系统由位于 I/O 空间的控制寄存器和一个位于状态寄存器中的全局中断使能位构成。 所有的中断在中断向量表中都有独立的向量，中断优先级与中断向量在向量表中的位置对应，向量地址越低， 中断优先级越高。

状态寄存器 (SREG)

状态寄存器保存了最新的算术运算指令的执行结果。这个信息可用于程序的流程控制以及条件执行。状态寄存器的值在 ALU 操作之后更新。

进入中断之后，状态寄存器并不会自动保存，这个工作必须由软件完成。

MVR8X 状态寄存器 – SREG

Bit	7	6	5	4	3	2	1	0	
0x3F(0x5F)	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**	SREG
Read/Write	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	
Initial Value	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	

Bit7 – I : 全局中断使能

全局中断使能后，MVR8X 内核才能接收并处理中断。各个模块独立的中断使能用于控制各个模块的中断请求功能。当全局中断清零后，所有的中断将不会被系统响应。中断被响应后，全局中断使能位(I)将被硬件自动清除。在执行 RETI 指令后，全局中断使能被自动置位，从而可以顺利的响应其他中断请求。I 位也可以使用 SEI 与 CLI 指令进行控制，具体请参考指令集相关介绍。

Bit6 – T: 位拷贝存储

位拷贝指令 BLD (Bit LoaD) 以及 BST (Bit Store)使用 T 位作为操作目标。可以使用 BST 将任意通用寄存器中的某一位拷贝到 T 位，也可以将 T 位拷贝到任意通用寄存器中的指定位（BLD）。

Bit5 – H: 半进位标志

H 标志指示某些算术操作的半进位状态，半进位在 BCD 运算中非常有用，请参考具体指令中的说明。

Bit4 – S: 符号位， $S = N \oplus V$

符号位是负数位与二进制补码溢出位的异或值。

Bit3 – V: 二进制补码溢出位

二进制补码溢出用于支持二进制补码运算。请参考指令集描述部分。

Bit2 – N: 负数标志

负数标志指示了一个算术或逻辑运算的结果为负数。

Bit1 – Z: 零标志

Z 标志指示了算术或逻辑运算的结果为零。请参考指令集描述部分。

Bit0 – C: 进位标志

C 表示指示算术或逻辑运算中产生了进位或借位操作。请参考指令集描述部分。

通用寄存器 (R0~R31)

MVR8X 内核对通用寄存器文件做了特殊优化。为获得需要的性能和灵活性，寄存器文件被设计为支持以下几种访问类型：

输出一个 8 位操作数和输入一个 8 位的运算结果
输出两个 8 位操作数和输入一个 8 位的运算结果
输出两个 8 位操作数和输入一个 16 位的运算结果
输出一个 16 位的操作数和一个 16 位的运算结果

MVR8X CPU 通用寄存器文件结构：

	7 0	Addr.	
通用寄存器文件	**R0**	0x00	
	R1	0x01	
	…		
	R13	0x0D	
	R14	0x0E	
	R15	0x0F	
	R16	0x10	
	R17	0x11	
	…		
	R26	0x1A	X 寄存器低地址
	R27	0x1B	X 寄存器高地址
	R28	0x1C	Y 寄存器低地址
	R29	0x1D	Y 寄存器高地址
	R30	0x1E	Z 寄存器低地址
	R31	0x1F	Z 寄存器高地址

大部分操作寄存器文件的指令都可以访问到所有的寄存器，而且这些指令大部分都是单周期指令。
如上图所示，每个寄存器都被关联到一个数据内存地址，并映射到数据空间的开始 32 个位置。这种组织方式给访问通用寄存器提供了灵活的方式， 可以使用 X-, Y-以及 Z-指令可以作为索引访问任意寄存器。

寄存器 R26..R31 可以两两组合为一个 16 位的地址指针寄存器，可用作间接寻址所有的数据空间。这三个间接寄存器的 X,Y 以及 Z 的定义如下：

	15 XH		XL 0
X 寄存器	7 0	7 0	
	R27(0x1B)	R26(0x1A)	

	15 YH		YL 0
Y 寄存器	7 0	7 0	
	R29(0x1D)	R28(0x1C)	

	15 ZH		ZL 0
Z 寄存器	7 0	7 0	
	R31(0x1F)	R30(0x1E)	

不同的寻址模式使用 X/Y/Z 作为偏移地址，支持访问后地址的自动偏移增加，自动递增以及自动递减。
具体实现细节，请参考不同的指令集部分。

堆栈指针(SP)

堆栈主要用于保持临时数据，局部变量以及中断和子程序调用的返回地址。必须注意的是，堆栈指针被设计为从高地址向低地址递减的模式。堆栈指针(SP)总是指向堆栈的顶端。堆栈指针指向数据空间(SRAM)。

位于 SRAM 空间的堆栈必须在所有子程序和中断之前正确的初始化。初始的 SP 指针总是指向 SRAM 的最后一个可用地址。请参考内存分布部分的定义。

Table： 堆栈指针操作指令

指令	堆栈指针	操作描述
PUSH	减一	数据压栈
CALL **ICALL** **RCALL**	减二	返回地址压栈
POP	加一	数据出栈
RET **RETI**	加二	返回地址出栈

MVR8X 堆栈指针由两个位于 I/O 空间的 10 位寄存器构成。SP 指针的有效位数与实际的实现有关。某些 MVR8X 实现的 MCU 只含有非常小的 SRAM， 因此只有 SPL 是有效的。在这种情况下，SPH 寄存器实际上并不存在。

SPH 与 SPL – 堆栈指针的高低位寄存器

Bit	15	14	13	12	11	10	9	8	
0x3E(0x5E)	-	-	-	-	-	-	**SP9**	**SP8**	SPH
0x3D(0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	1	
	1	1	1	1	1	1	1	1	

Table : **LGT8F0XA** 系列堆栈指针有效位数：

设备	堆栈指针大小
LGT8F01A	SP[8:0]
LGT8F02A	SP[8:0]
LGT8F04A	SP[9:0]
LGT8F08A	SP[9:0]

关于堆栈指针初始化的说明：

LGT8F08A/04A 系列，初始化 SP 寄存器在 0x100 ~ 0x2FF 以内的有效空间；

LGT8F02A/01A 系列，初始化 SP 寄存器到 0x100 ~ 0x1FF 以内的有效地址空间。

对于 LGT8F08A/04A， 内部 SRAM 为 512 字节，映射到从 0x100 开始的数据存储空间。使用 M164 兼容模式编译代码时， SP 默认初始化为 0x4FF， 这个默认值对 LGT8F08A/04A 一样有效，内部的地址偏移机制会把 SP 准确定位到 SRAM 的最高地址上。但对于 LGT8F02A/01A，需要用户正确配置 SP 指针。

复位与中断处理

MVR8X 支持 22 种不同的中断源，这些中断以及复位向量分别对应于程序空间的向量地址。大部分中断都对应一个单独的使能控制，这些控制位并连同全局中断使能置位后，MVR8X 内核才能响应相应的中断请求。

程序空间的最低地址默认定义为复位以及其他中断的向量地址空间。完整的向量地址分配，请参考中断控制器描述部分。中断向量地址的位置，也同时决定了中断响应的优先级。向量地址越低，中断的优先级就会越高。因此复位（RESET）具有最高的优先级，然后是外部中断 0（INT0）。

中断发生后，全局中断使能位（I 位）被硬件自动清零。之后的中断将不在被响应。用户可以在中断服务程序中打开全局使能位，从而实现中断嵌套。所有的中断都可以被嵌套。执行中断返回指令(RETI)，将会同时置位全局使能位，从而可以继续响应其他中断。

中断基本上分为两种模式。第一种为事件触发，设置中断标志位。对于这些中断，程序计数器(PC)直接指向向量地址，执行该地址处的中断服务程序。响应中断后，硬件自动清除中断标志位。中断标志位也可以通过软件写 1 清除。如果发生中断的同时，中断不能被立刻响应，中断标志将会保持，直到中断得到响应，或者被用户通过软件主动清除。同样，如果中断发生的同时，全局中断被清除，中断标志位也将继续保持，直到全局使能被置位后，中断得到响应。同时发生的中断，将依据他们的优先级被先后执行。
第二种中断为状态中断，只要中断条件成立，中断就将一直保持。此类中断不需要相应的标志位。如果在中断被使能前中断条件消失， 中断将不会被响应。当 MVR8X 从中断中返回后，通常会执行一条或几条指令， 然后才会响应中断请求。

需要注意的是，状态寄存器并不会在中断响应的过程中被自动保存和恢复。这个工作必须由软件完成。
当用户使用 CLI 指令禁止中断响应，中断将会被立即禁止，即使是与 CLI 同时发生的中断也将无法得到响应。下面的程序例子描述如何在编程 EEPROM 时禁止中断产生的干扰。

汇编代码样例

```
in   r16, SREG          ; store SREG value
cli                     ; disable interrupts during timed sequence
sbi  EECR, EEMPE        ; start EEPROM write
sbi  EECR, EEPE
out  SREG, r16          ; restore SREG value (I-bit)
```

C 语言代码

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
__disable_interrupt();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

中断响应时间

当使用 SEI 指令使能中断，SEI 之后的指令将会被首先执行，然后中断才会得到响应。
当中断使能打开后，中断将会在最多 4 个周期内被响应。四个周期后，中断向量执行的指令进入执行期。中断服务程序开始执行。在这四个周期内，当前程序地址 PC 将会被压入堆栈。中断向量一般为一条跳转指令，跳转指令一般为 1 或 2 个周期。如果中断发生在 MCU 处于休眠模式，中断将首先唤醒 MCU，然后才能响应响应的中断，因此中断响应时间将会增加 3 个周期。从中断服务返回需要 2 个周期。在这两个周期内，返回地址从堆栈中弹出，堆栈指针加 2，全局中断使能位被硬件自动置位。

内存分配

本章主要介绍 LGT8F0XA 系列 MCU 的不同存储空间。MVR8X 结构包含了两种内存空间， 数据存储空间以及程序存储空间。另外 LGT8F0XA 系列也包含了一个 504 字节的数据 FLASH，支持字节读写功能，无需用户擦除管理，用户存放需要掉电保持的持久性数据。所有这三种存储空间都是线性的

系统可编程 FLASH 程序存储内存

LGT8F0XA 包含 8K/4K/2K/1K 字节的片上系统可编程 FLASH 存储器。用于存储系统程序。所有的 MVR8X 指令为 16 或 32 位宽，FLASH 的数据宽度统一为 x16 位宽。

FLASH 存储器的使用寿命为至少 20,000 次擦写周期。LGT8F0XA 系列程序计数器分别为 13/12/11/10 位，可以寻址到 8K/4K/2K/1K 的空间范围。FLASH 空间可以通过 ISP 接口（SPI）和在线调试接口（SWD）进行编程和读取操作。LGT8KF0XA 也支持通过用户程序编程内部 FLASH，用户可以通过这个功能实现自定义的在线编程和在线程序升级功能。

LGT8F0XA 系列均包含了 504 字节的数据 FLASH， 用于保存用户数据。LGT8F0XA 内部实现了一个专用的 EEPROM 控制器，用于管理这部分数据空间。通过这个专用的控制器，可以实现基于字节的读写访问，从而节省了大量的管理代码。详细操作细节，请参考 EEPROM 相关章节。

256/512BSRAM 数据存储空间

LGT8F0XA 系列是一种复杂微控制器，集成了可以通过 IN/OUT 指令访问的外设控制器。这部分包括了 I/O 空间的最低 64 字节范围。对于从 0x60 到 0xFF 之间的扩展 I/O 空间，被分配到数据存储空间，只能通过 ST/STS/STD 以及 LD/LDS/LDD 指令访问。

数据存储空间包含了寄存器文件，I/O 空间，扩展 I/O 空间以及内部 SRAM。开始的 32 个字节映射了 MVR8X 内部的 32 个通用寄存器，然后是 64 字节的标准 I/O 空间，160 字节的扩展 I/O 空间，最后是 256/512 字节的内部数据 SRAM 空间。

程序可以通过 5 中不同的寻址方式访问数据存储空间：直接寻址，带偏移的间接寻址，间接寻址，带预递减的间接寻址以及支持访问后地址递增的间接寻址模式。在通用寄存器中，寄存器 R26 到 R31 可用于间接寻址的地址指针寄存器。

直接寻址可以寻址到全部的地址空间。

带偏移的间接寻址模式支持从基地址开始的 63 个地址控制，基地址通过 Y 或 Z 寄存器指定。

当使用支持访问前地址递减或访问后地址递增的寄存器间接寻址模式时，地址指令寄存器 X,Y 或 Z 将自动发生递增或递减。

LGT8F0XA 数据空间分布：

数据存储空间

32 个通用寄存器	0x0000 ~ 0x001F
64 I/O 寄存器	0x0020 ~ 0x005F
160 扩展 I/O 寄存器	0x0060 ~ 0x00FF
256/512 字节 内部 SRAM 空间	0x0100 0x01FF/0x02FF

EEPROM 数据存储空间

LGT8F0XA 系列均包含了 504 字节的数据 EFLASH。配合内部的专用 EEPROM 控制器，实现了基于字节访问的 EEPROM 功能。内部 EEPROM 控制器针对用户的写操作进行了优化，可以避免不必要的重写和多余的擦除操作，极大的提高了写操作的平均速度和使用寿命。配合 EEPROM 内部的写均衡机制，可以将数据 FLASH 的使用寿命平均提高 2~3 倍以上。

用户也可以通过专用 ISP 通道(SPI 接口), 串行在线调试接口(SWD)对这部分区域进行访问，具体细节请参考相关章节。软件可以通过 I/O 空间访问 EEPROM 控制器，从而实现对 EEPROM 空间的读写。细节请参考下面的寄存器描述部分。为了防止误操作，对 EEPROM 空间的访问需要遵循一个特定的时序。细节请参 EEPROM 控制寄存器。当对 EEPROM 进行读写访问时，MVR8X 内核将处于待机状态。读写操作结束后，CPU 在 2 个系统时钟后继续运行，执行待机前的指令。

I/O 空间

LGT8F0XA 包含两种 I/O 空间。标准 I/O 空间以及扩展 I/O 空间。细节请参考寄存器定义部分。
所有外设控制器的寄存器都被分配到 I/O 空间。所有的 I/O 地址都可以使用 LD/LDS/LDD 以及 ST/STS/STD 指令访问，以及在这些 I/O 寄存器与 32 个通用寄存器之间传递数据。**地址从 0x00 到 0x1F 范围内的 I/O 寄存器支持位访问指令 SBI/CBI。这些寄存器的中的某一位也可以使用 SBIS/SBIC 指令检测，控制程序执行流程。当使用 IN/OUT 指令访问时，必须使用 0x00~0x3F 作为目标地址。当使用 LD/ST 指令访问 I/O 寄存器控制时，必须加上 0x20 的基地址。此基地址为 I/O 空间在数据存储空间的映射基地址。从 0x60 开始到 0xFF 之间的扩展 I/O 空间，被系统映射到数据空间，因此只能够通过 ST/STS/STD 以及 LD/LDS/LDD 指令访问。**

寄存器描述

EEARH/EEARL – EEPROM/EFLASH 地址寄存器

Bit	15	14	13	12	11	10	9	8	
0x22(0x42)	-	-	-	EEAR12	EEAR11	EERA10	EEAR9	EEAR8	**EEARH**
0x21(0x41)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	**EEARL**
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Bit 15:13 – Reserved

这些位在 LGT8F0XA 系列 MCU 中并没有使用，读返回零，写操作无任何意义。

Bit 12:0 – EEAR[12:0] : EEPROM 地址寄存器

LGT8F0XA 系列中包含 504 字节的 EEPROM 空间，因此只有 EEAR[8:0]实际用于访问 EEPROM 数据空间。EEAR[12:0]只用于通过 EEPROM 控制器接口访问程序 FLASH 空间。无论是访问 EEPROM 还是访问程序空间，这里指定的地址均为以字节寻址的地址值。

EEDR – EEPROM/EFLASH 数据寄存器

Bit	7	6	5	4	3	2	1	0	
0x20(0x40)	MSB							LSB	**EEDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7:0 – EEDR[7:0] : EEPROM/EFLASH 数据寄存器

EEPROM 空间是以字节为单位读写访问，所以对于 EEPROM 空间的访问，这里存放了读写操作的字节数据。FLASH 空间是以双字节(16 位)为单位访问，当通过 EEPROM 控制器接口访问 FLASH 空间时，这里存放读写数据的低字节部分。高字节部分请参考 EEDHR 寄存器定义。

EECR – EEPROM/EFLASH 控制寄存器

Bit	7	6	5	4	3	2	1	0	
0x1F(0x3F)	EEPM2	-	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	**EECR**
Read/Write	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – EEPM2 : FLASH 访问使能位

EEPM2 用于使能访问程序空间的控制逻辑。软件可以通过设置此位，实现通过 EEPROM 控制器对 FLASH 空间进行读/写/擦除操作。需要注意的是对 FLASH 空间的读写操作是以双字节为单位，擦除操作是以页为单位。FLASH 空间的页面大小为 256x16。

Bit 6 – Reserved : 保留位

Bit [5:4] – EEPM[1:0] : FLASH 操作模式控制位

这两位只在 EEPM2 设置为 1 的前提下才会起作用。用于控制对 FLASH 空间的操作模式。

EEPM 模式定义如下：

EEPM1	EEPM0	操作模式
0	1	FLASH 空间页擦除操作，页定制由 EEAR[12:9]指定 EEAR[8:0]需设置为全零。
1	0	FLASH 空间编程操作，编程以双字节为单位，编程地址由 EEAR[12:1]指定，数据通过 EEDR/EEDHR 指定
其他配置		FLASH 空间读操作，读操作是以字为单位，读地址由 EEAR[12:1]指定。数据存放在 EEDR/EEDHR

Bit 3 – EERIE : EEPROM/EFLASH 控制器就绪中断使能

设置 EERIE 位将使能 EEPROM/EFLASH 控制器准备就绪中断请求。清除此位将禁止就绪中断请求。控制器检测到 EEPE 为零后，将会产生一个持续的 EEPROM/EFLASH 控制器就绪中断请求。

Bit2 – EEMPE : EEPROM/EFLASH 编程时序使能控制

EEMPE 用于产生一个 EEPROM/EFLASH 编程时序使能控制。当 EEMPE 被设置后，必须在之后的四个周期内设置 EEPE 位，同时清除 EEMPE 位，数据才能被正确的编程。因此 EEPROM/EFLASH 编程应该满足下面列出的时序，否则将不会产生任何动作。

1. 等待 EEPE 清零（EEPROM/EFLASH 就绪）
2. 写入编程目标地址到 EEAR（可选）
3. 写入数据到 EEDR/EEDHR（可选）
4. EEMPE 写 1，同时 EEPE 清零
5. 在 EEMPE 写入之后的四个系统时钟内，置位 EEPE

需要特别注意的是， 在以上操作的 4 和 5 过程中，如果发生了中断，将导致编程无效。因为中断将破坏四个周期的时序约束。所以建议在进行步骤 4 和 5 时，关闭全局中断使能。

当编程操作完成后，EEPE 位将会被硬件自动清零。用户可以通过查询此位的状态，决定是否可以继续下一次写操作。在对 EEPROM/EFLASH 操作的过程中，CPU 将处于待机状态。

Bit 0 – EERE: EEPROM/EFLASH 读使能

当地址被设置到 EEAR 寄存器后，通过设置 EERE 位为 1 启动 EEPROM/EFLASH 读操作。在读操作的过程中，CPU 处于待机模式。由于 EEPROM 与程序空间公用数据总线，所以读数据将会在写 EERE 寄存器后 2 个周期有效。软件需要在 EERE 置位后插入至少两个 NOP 后，才能正确的读到数据。

下面将给出通过 EEPROM 控制器访问 EEPROM 空间的代码示例。假设用户在操作之前关闭了全局中断使能。

EEPROM 读操作汇编代码

```
EEPROM_read:
    ; Wait for completion of previous write
    sbic EECR,EEPE
    rjmp EEPROM_read
    ; Set up address (r18:r17) in address register
    out EEARH, r18
    out EEARL, r17
    ; Start eeprom read by writing EERE
    sbi EECR,EERE
    nop
    nop
    ; Read data from Data Register
    in r16,EEDR
    ret
```

EEPROM 读操作 C 语言代码

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEPE));
    /* Set up address register */
    EEAR = uiAddress;
    /* Start eeprom read by writing EERE */
    EECR |= (1<<EERE);
    __nop(); __nop();
    /* Return data from Data Register */
    return EEDR;
}
```

EEPROM 编程操作汇编代码

```
EEPROM_write:
    ; Wait for completion of previous write
    sbic    EECR, EEPE
    rjmp    EEPROM_write
    ; Set up address (r18:r17) in address register
    out     EEARH, r18
    out     EEARL, r17
    ; Write data (r16) to Data Register
    out     EEDR, r16
    ; Write logical one to EEMPE
    sbi     EECR, EEMPE
    ; Start eeprom write by setting EEPE
    sbi     EECR, EEPE
    ret
```

EEPROM 编程操作 C 语言代码

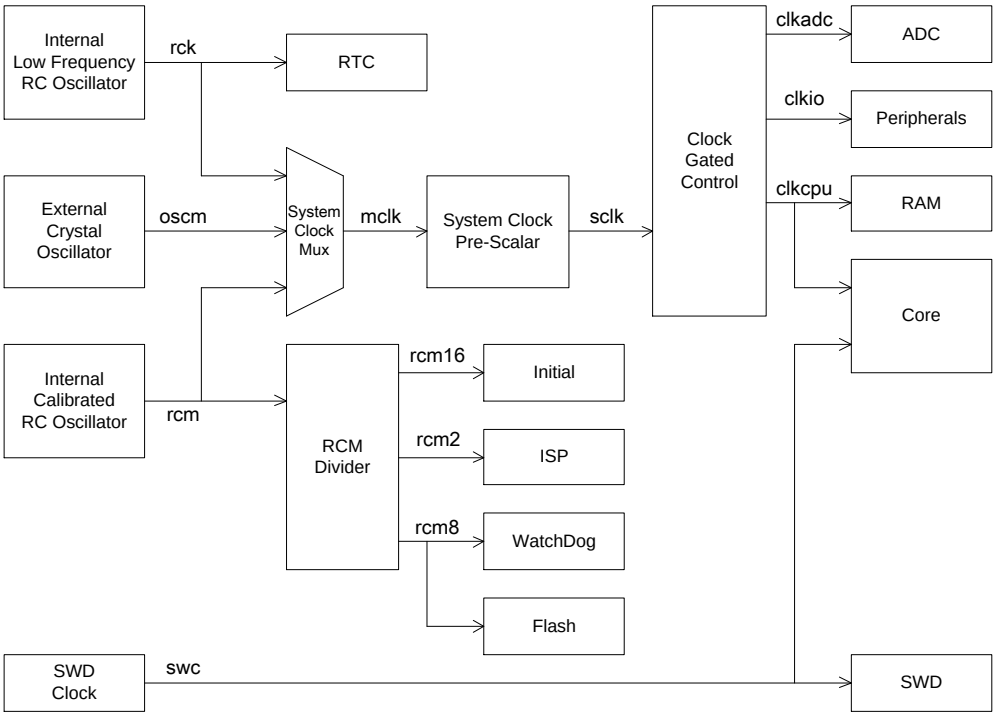
```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEPE));
    /* Set up address and Data Registers */
    EEAR = uiAddress;
    EEDR = ucData;
    /* Write logical one to EEMPE */
    EECR |= (1<<EEMPE);
    /* Start eeprom write by setting EEPE */
    EECR |= (1<<EEPE);
}
```

特别提醒：当使用 C 语言编程的时候，需要注意编译器的优化级别。保证在 EEMPE 和 EEPE 之间的四个系统周期限制。可以通过反汇编，查看 EEMPE 与 EEPE 之间的指令周期数。

系统时钟与功耗管理

本章主要介绍 LGT8F0XA 系列 MCU 的时钟分布以及时钟管理。LGT8F0XA 支持多种时钟源，但并不是所有的时钟都需要同时工作。为了节省功耗，不需要的时钟模块可以通过寄存器关闭。

LGT8F0XA 系统时钟结构：



时钟概述

CPU 时钟(clkcpu)主要用于驱动 MVR8X 内核以及内部 SRAM。MVR8X 内核组成包含了通用寄存器文件，状态寄存器，以及存放堆栈的 SRAM。当系统进入休眠模式后，CPU 时钟将被关闭，CPU 将处于待机模式。

I/O 时钟(clkio)用于驱动所有的 I/O 外设。比如定时器，SPI/USART 控制器，以及 ADC 控制器接口逻辑。I/O 时钟也用于外部中断模块。但是有些外部中断模式工作于异步模式，这样可以确保 I/O 时钟关闭后仍然可以检测到中断请求。不同的 I/O 外设也配有单独的时钟控制寄存器，可以在不需要相关模块时关闭该 I/O 模块的工作时钟。

FLASH 编程时钟(clkflash)。因为 FLASH 编程需要相对精确的时序，系统使用内部可校准 RC 振荡器时钟的 8 分频作为 FLASH 读写工作时钟。由于内部 RC 振荡器校准之后可以达到 1%的精度，其低温漂效应以及与外部工作电压无关的特性，保证了在各种条件下 FLASH 的正常操作。

ADC 时钟(clkadc)为 ADC 控制器接口提供工作时钟。ADC 独立的时钟控制可以实现在关闭 CPU 时钟和 I/O 时钟后 ADC 仍然可以继续工作，从而减小了系统对 ADC 模块的噪声干扰。这种工作机制可以提高 ADC 转换的精度。

RTC 时钟(rck)。LGT8F0XA 系列内部均集成了一个 128Hz 左右 RC 振荡器，可以在系统掉电的情况下继续工作。RTC 内部包含了一个 24 位的计数器，可用于实现更长时间的计时/计数应用。此 RTC 也可以用于掉电模式的唤醒工作。内部低频 RC 振荡器时钟也可以作为系统工作时钟。

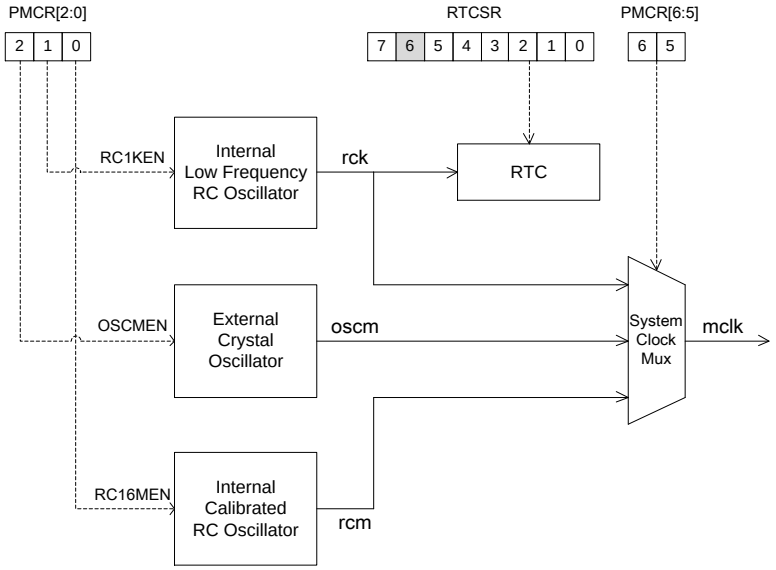
时钟源

LGT8F0XA 系列 MCU 主要支持三种时钟源，分别为内部可校准 RC 振荡器，内部低频 RC 振荡器以及外部晶振时钟。对于大部分应用，内部可校准 RC 提供 16MHz 的精准时钟，配合内部的系统时钟预分频器，可以产生从 16MHz 到 125KHz 一共 8 种频率，基本上能满足大部分应用。从而节省了外部晶振。在一些对频率特征有特殊要求的应用，LGT8F0XA 也支持外部晶振，给系统工作带来更大的灵活性。
不同时钟源的切换可以通过系统功耗管理控制寄存器实现(PMCR)。 具体细节请参考先关章节。

系统上电后，默认情况下，LGT8F0XA 工作于内部可校准 RC 振荡器的两分频，一般情况下为 8MHz 左右。系统启动后，软件可以通过相关的时钟控制寄存器，选择合适的时钟源以及时钟频率。

另外需要注意， 外部晶振 XTAL1/XTAL2 是与 PC0/PC1 复用的，在配置选择外部晶振工作前，需要软件保证 PC0/1 的为输入状态，并禁用 PC0/1 上的上拉电阻。同样，如果要使用 PC0/PC1 的 GPIO 功能，也需要首先禁止外部晶振。具体细节请参考寄存器定义部分。

时钟源控制结构如下图所示：



重要提醒：系统上电后，内部可校准 RC 振荡器处于工作状态。内部低频率 RC 以及外部晶振 I/O 处于关闭状态。用户在切换主时钟源之前，必须首先开启切换的目标时钟源。等待时钟稳定后，才可以执行切换时钟的操作。各种时钟源的启动时间，请参考如下列表：

时钟源模块启动时间：

时钟源	启动时间
内部低频 RC 振荡器	1ms
内部可校准 16MHz RC 振荡器	10us
外部晶振 I/O	-

睡眠模式

LGT8F0XA 共有 4 种睡眠模式，包括空闲模式，ADC 噪声抑制模式，掉电模式和断电模式。下面分别进行说明。
空闲模式：当设置 SMOD 为 0 时，执行 SLEEP 指令将使 MCU 进入空闲模式。在此模式下，CPU 停止运行，而 USART/USPI，AC，ADC，TC0，TC1，WDT，外部中断，电平改变中断和中断系统继续工作。这个休眠模式停止了 clkcpu 和 clkflash，其他时钟继续工作。所有中断均可以唤醒 MCU。

ADC 噪声抑制模式：当设置 SMOD 为 1 时，执行 SLEEP 指令将使 MCU 进入 ADC 噪声抑制模式。在此模式下，CPU 停止运行，USART/USPI，AC，TC0，TC1 等外设停止工作，而 ADC，WDT，外部中断，电平改变中断和中断系统继续工作。此模式提高了 ADC 的噪声环境，使得转换精度提高。当 ADC 使能时，进入此模式将自动启动一次 AD 转换。这个休眠模式停止了 clkio，clkcpu 和 clkflash，其他时钟继续工作。ADC 中断,看门狗复位，外部中断，电平改变中断均可以唤醒 MCU。

掉电模式：当设置 SMOD 为 2 时，执行 SLEEP 指令将使 MCU 进入掉电模式。在此模式下，CPU 停止运行，USART/USPI，AC，ADC，TC0，TC1 等外设停止工作，而 WDT，外部中断，电平改变中断和中断系统继续工作。这个休眠模式停止了 clkadc，clkio，clkcpu 和 clkflash，其他时钟继续工作。看门狗复位，外部中断，电平改变中断均可以唤醒 MCU。

断电模式：当设置 POWOFF 位为“1”时，将使 MCU 进入断电模式。在此模式下，只有实时时钟模块继续工作，其它模块均关断电源，此模式下系统功耗进入最低。外部引脚 PC6 和 RTC 计数溢出可以唤醒系统。

下表为不同睡眠模式下的活动时钟以及唤醒源

睡眠模式	工作时钟					唤醒源				
	clkcpu	clkflash	clkio	clkadc	rtc	INTn PCINTn	WDT	ADC	USART USPI TC	PC6 RTC
空闲模式			√	√		√	√	√	√	
ADC 噪声抑制模式				√		√	√	√		
掉电模式						√	√			
断电模式					√					√

寄存器定义

PMCR – 功耗管理控制寄存器

地址： 0xF2			初始值： 0x11	
Bit	**Name**	**R/W**	**Description**	
7	-	-	保留位	
6	LFEN	R/W	低频 RC 选择控制位, 当设置 LFEN 位为“1”时，选择内部低频 RC 时钟为系统工作时钟。 当设置 LFEN 位为“0“时，依据 EXTMSEL 位来选择系统工作时钟。	
5	EXTMSEL	R/W	外部晶振选择控制位, 当设置 EXTMSEL 位为“1”且 LFEN 为“0”时，选择外部晶振为系统工作时钟。 当设置 EXTMSEL 位为“0”且 LFEN 为“0”时，选择内部 16MHz 晶振为系统工作时钟。	
4	CFDS	R/W	I/O 驱动强度配置位	
			0	低电平输出电流: 5.4mA，最大可到 7.4mA
				高电平输出电流: 7.8mA，最大可到 12.9mA
			1	低电平输出电流: 10.7mA，最大可到 14.7mA
				高电平输出电流: 15.6mA，最大可到 25.8mA
3	-	-	保留位	
2	OSCMEN	R/W	**工程样品 LGT8F08A-16C， 为 0 时，禁用外部晶振，默认为 0**	
			其他所有 LGT8F0XA 型号，为 1 时，禁用外部晶振， 默认为 0	
1	RC1KEN	R/W	内部低频 RC 振荡器使能控制	
0	RC16MEN	R/W	内部 16MHz 振荡器使能控制	

CLKPR – 主时钟预分频寄存器

<table>
<tr><td colspan="3">地址： 0x61</td><td colspan="5">初始值： 0x01</td></tr>
<tr><th>Bit</th><th>Name</th><th>R/W</th><th colspan="5">Description</th></tr>
<tr><td>7</td><td>CLKPCE</td><td>R/W</td><td colspan="5">时钟分频改变使能位
改变时钟分频之前，必须先设置此位，软件必须在设置此位之后的 4 个系统周期内改变时钟分频设置，否则设置将不会产生作用</td></tr>
<tr><td>6:3</td><td>-</td><td>-</td><td colspan="5">保留位</td></tr>
<tr><td rowspan="10">2:0</td><td rowspan="10">CLKPS[2:0]</td><td rowspan="10">R/W</td><td colspan="5">预分频系数</td></tr>
<tr><td>CLKPS2</td><td>CLKPS1</td><td>CLKPS0</td><td colspan="2">分频系数</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td colspan="2">1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td colspan="2">2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td colspan="2">4</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td colspan="2">8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td colspan="2">16</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td colspan="2">32</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td colspan="2">64</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td colspan="2">128</td></tr>
</table>

OSCCAL – RC 精调控制寄存器

地址： 0x66			初始值： 0x11
Bit	**Name**	**R/W**	**Description**
7	-	-	保留位
6	PRESETN	R/W	Pad 复位功能选择，只在 14 脚和 8 脚封装时有效 当设置 PRESETN 位为“1”时，复用复位功能的引脚使能复位功能。 当设置 PRESETN 位为“0”时，复用复位功能的引脚禁止复位功能。
5:0	RCCAL[5:0]	R/W	16MHz RC 精调控制位 片内 16MHz 晶振的精调过程： 1） 测量 16MHz 晶振的输出时钟频率 Fm，可通过测量 PWM 波形的频率来计算系统时钟的频率，默认设置下系统时钟为 16MHz 晶振输出时钟的二分频，并且晶振的可调参数 RCCAL = 0x0； 2） 比较 Fm 和目标时钟频率 Ft（如 16MHz），若相等则晶振参数不用调整即保持 RCCAL 值不变，若 Fm > Ft 进入步骤 3），若 Fm < Ft 进入步骤 4）； 3） 当 Fm > Ft，取 F_{LSB} = 1% * Fm，Y = (Fm – Ft)/F_{LSB}，Z = 64 – ROUND(Y)； 4） 当 Fm < Ft，取 F_{LSB} = 1% * Fm，Y = (Ft – Fm)/F_{LSB}，Z = ROUND(Y)； 5） 计算出的 Z 值即为晶振精调参数，把 Z 值写入 RCCAL 即可。

PRR – 省电控制寄存器

地址: 0x64			初始值: 0x00
Bit	**Name**	**R/W**	**Description**
7:6	-	-	保留位
5	PRTC0	R/W	TC0 省电使能控制位 当设置 PRTC0 位为“1”时，关闭 TC0 相应模块的工作时钟。当不需要使用 TC0 模块时置位 PRTC0 来减少不必要的功耗。 当设置 PRTC0 位为“0”时，打开 TC0 相应模块的工作时钟。启用 TC0 正常工作前必须先清零 PRTC0 位来开启时钟。
4	-	-	保留位
3	PRTC1	R/W	TC1 省电使能控制位 当设置 PRTC1 位为“1”时，关闭 TC1 相应模块的工作时钟。当不需要使用 TC1 模块时置位 PRTC1 来减少不必要的功耗。 当设置 PRTC1 位为“0”时，打开 TC1 相应模块的工作时钟。启用 TC1 正常工作前必须先清零 PRTC1 位来开启时钟。
2	-	-	保留位
1	PRUSART	R/W	USART 省电使能控制位 当设置 PRUSART 位为“1”时，关闭 USART/USPI 相应模块的工作时钟。当不需要使用 USART/USPI 模块时置位 PRUSART 来减少不必要的功耗。 当设置 PRADC 位为“0”时，打开 USART/USPI 相应模块的工作时钟。启用 USART/USPI 正常工作前必须先清零 PRUSART 位来开启时钟。
0	PRADC	R/W	ADC 省电使能控制位 当设置 PRADC 位为“1”时，关闭 ADC 相应模块的工作时钟。当不需要使用 ADC 模块时先清零 ADC 使能位再置位 PRADC 来减少不必要的功耗。 当设置 PRADC 位为“0”时，打开 ADC 相应模块的工作时钟。启用 ADC 正常工作前必须先清零 PRADC 位来开启时钟。

SMCR – 睡眠模式控制寄存器

地址: 0x33			初始值: 0x00			
Bit	**Name**	**R/W**	**Description**			
7:4	-	-	保留位			
3:1	SMOD[2:0]	R/W	睡眠模式控制位			
			SMOD2	SMOD1	SMOD0	睡眠模式
			0	0	0	空闲模式
			0	0	1	ADC 噪声抑制模式
			0	1	0	掉电模式
			0	1	1	保留
			1	0	0	保留
			1	0	1	保留
			1	1	0	保留
			1	1	1	保留

SMCR – 睡眠模式控制寄存器(接上页)			
地址: 0x33			初始值: 0x00
Bit	**Name**	**R/W**	**Description**
0	SE	R/W	睡眠使能控制位 当设置 SE 位为“1”时，使能睡眠模式，执行 SLEEP 指令 CPU 将按照 SMOD 的设置进入相应的睡眠模式。为了防止 CPU 误入睡眠模式，建议在即将执行 SLEEP 指令前置位 SE 并在系统唤醒之后立即清除 SE 位。 当设置 SE 位为“0”时，禁止睡眠模式。

RTCSR – RTC 控制和状态寄存器

地址: 0xD0			默认值: 0x80
Bit	Name	R/W	描述
7	WREN	R/W	写使能标志位。 当 WREN 位为“1”时，允许 CPU 写 RTCSR 寄存器。 当 WREN 位为“0”时，写 RTCSR 寄存器操作失效。 CPU 配置 RTCSR 寄存器之前最好先读取 WREN 位的状态。
6:5	-	-	保留。
4	LOAD	R/W	加载使能控制位。 当设置 LOAD 位为“1”时，RTC 计数器加载计数最大值寄存器 RTCTOPH，RTCTOPM 和 RTCTOPL 的数值。加载完毕后，硬件自动清理 LOAD 位。
3	PWEN	R/W	外部引脚唤醒使能控制位。 当设置 PWEN 位为“1”时，外部引脚唤醒功能被使能。系统工作在 Power Off 模式下，可通过外部引脚和 RTC 计数溢出来唤醒。使用外部引脚来唤醒时，需要保持 PC6 为高电平并持续超过 500us。 当设置 PWEN 位为“0”时，外部引脚唤醒功能被禁止。
2	CWEN	R/W	计数溢出唤醒使能控制位。 当设置 CWEN 位为“1”时，计数溢出唤醒功能被使能。系统工作在 Power Off 模式下，可通过外部引脚和 RTC 计数溢出来唤醒。使用计数溢出来唤醒时，CPU 须预设计数最大值并使能计数，当计数到最小值发生溢出即可唤醒系统。 当设置 CWEN 位为“0”时，计数溢出唤醒功能被禁止。
1	CEN	R/W	计数使能控制位。 当设置 CEN 位为“1”时，RTC 开始计数。 当设置 CEN 位为“0”时，RTC 停止计数。
0	POWOFF	R/W	Power Off 使能控制位。 当 CPU 写 POWOFF 位为“1”时，系统进入 Power Off 模式，只有 RTC 模块工作，系统功耗最低。Power Off 模式下，可通过外部引脚和 RTC 计数溢出来唤醒。系统唤醒后 CPU 须清零 POWOFF 位。

MCUCR – MCU 控制寄存器

地址： 0x35			初始值： 0x00
Bit	**Name**	**R/W**	**Description**
7	SWDD	R/W	SWD 串行调试接口禁止控制位 当设置 SWDD 位为“1”时，禁止 SWD 串行调试接口。 当设置 SWDD 位为“0”时，使能 SWD 串行调试接口。 为了避免无意的禁止或使能 SWD 接口，必须通过下面的时序来改变 SWDD 位：软件必须在 4 个时钟周期内将期望的数值两次写入 SWDD 位。调试过程中，请避免置位 SWDD 位。 **SWD 调试接口与两个通用 I/O 复用，如果用户需要使用这两个 I/O，需要设置 SWDD 为 1，禁止 SWD 调试接口。禁止 SWD 接口后，芯片将无法正常调试，用户需要在设计时考虑这两个 I/O 的功能分配，建议将最简单的功能分配到这里，保证其他重要功能可以进行正常的调试。**
6:5	-	-	保留位
4	PUD	R/W	上拉禁止控制位 当设置 PUD 位为“1”时，禁止端口上拉。 当设置 PUD 位为“0”，且端口寄存器的设置满足上拉条件时（DDxn = 0，PORTxn = 1），使能端口上拉。
3:0	-	-	保留位

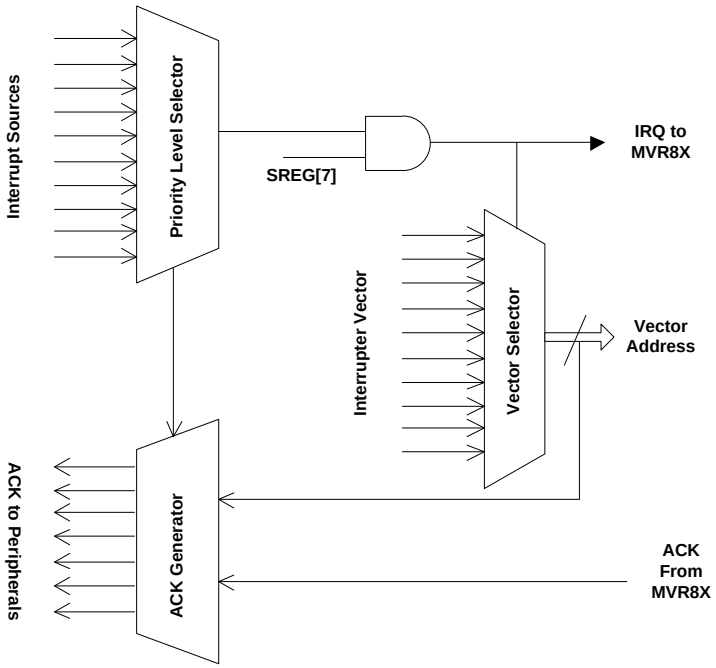
中断管理

本章主要介绍 LGT8F0XA 系列 MCU 的中断资源以及中断管理机制。**MVR8X 内核对中断响应机制做了专门的优化，使得 MVR8X 内核能够在 4 个周期之内执行中断服务程序。中断返回指令也仅需 2 个周期。**MVR8X 中断优先级与中断向量地址分布相对应，中断地址越低，中断优先级就越高。系统复位后，MVR8X 从 0x0000 地址开始执行复位向量代码，复位中断拥有最高优先级。

中断向量

MVR8X 内核包含了一个专用的中断管理器，用于管理不同优先级别的中断响应，产生最终的中断向量，并在中断得到响应后产生响应的中断应答信号。外设在收到中断应答信号后，硬件自动清除中断标志位。

MVR8X 内核中断控制器框架图：



LG8F0XA 系列 MCU 内外共 22 个中断源，各个中断源有独立的中断使能控制位。用户需要在开启外设的中断功能前，根据中断向量地址，初始化中断服务子程序。与子程序调用类似，系统在响应中断后，程序计数器指向中断向量地址，从该地址取出将要执行的向量代码。于此同时，MVR8X 内核将被中断的程序地址压入系统堆栈。因此，软件需要提前初始化堆栈指针，否则中断程序将不能正确的返回。

用户在配置完外设中断后，必须打开 MVR8X 状态寄存器中的全局中断使能控制位，内核才能够正确的响应中断。MVR8X 内核各个中断向量之间地址间隔 2 个字节，一般用户需要在此地址放置一个远跳转指令(JMP)，跳转到真正的中断服务子程序。具体细节请参考本章下面提供的示例代码部分。

MVR8X 内核中断设计为支持无限制中断嵌套。在中断请求得到响应后，全局中断使能被硬件自动关闭，从而禁止响应之后的中断请求。用户可以通过在中断服务程序中设置状态寄存器中的全局中断使能位，重新让 MVR8X 内核响应新的中断请求。中断嵌套不仅仅会影响中断服务的执行时间，也会对堆栈产生影响，用户需要在使用中断嵌套功能时特别注意。

中断向量地址分配

复位与中断向量表

Vector No	Program Address	Source	Interrupt Definition
1	0x0000	RESET	外部复位，上电复位，看门狗复位，JTAG 调试复位以及软件复位
2	0x0002	INT0	外部中断请求 0
3	0x0004	INT1	外部中断请求 1
4	0x0006	INT2	外部中断请求 2
5	0x0008	PCINT0	I/O 电平变化中断请求 0
6	0x000A	PCINT1	I/O 电平变化中断请求 1
7	0x000C	PCINT2	I/O 电平变化中断请求 2
8	0x000E	PCINT3	I/O 电平变化中断请求 3
9	0x0010	WDT_INTR	看门狗定时器溢出中断
向量 10~12 保留不用			
13	0x0018	ICF1_INTR	定时器 1 输入俘获中断
14	0x001A	OCA1_INTR	定时器 1 比较器 A 匹配中断
15	0x001C	OCB1_INTR	定时器 1 比较器 B 匹配中断
16	0x001E	TOV1_INTR	定时器 1 溢出中断
17	0x0020	OC0_INTR	定时器 0 比较器匹配中断
向量 18 保留不用			
19	0x0024	TOV0_INTR	定时器 0 溢出中断
向量 20 保留不用			
21	0x0028	RXC0_INTR	USART 接收完成中断
22	0x002A	UDR0_INTR	USART 数据寄存器空
23	0x002C	TXC0_INTR	USART 发送完成中断
24	0x002E	ACP_INTR	比较器中断请求
25	0x0030	ADC_INTR	ADC 转换完成中断请求
26	0x0032	EEP_INTR	EEPROM 就绪
向量 27 保留不用			
28	0x0036	RTC_INTR	实时定时器中断请求

中断向量初始化程序代码示例

Address	Labels	Code		Comments
0x0000		Jmp	RESET	;RESET
0x0002		Jmp	INT0	;IRQ0
0x0004		Jmp	INT1	;IRQ1
0x0006		Jmp	INT2	;IRQ2
0x0008		Jmp	PCINT0	;PCINT0
0x000A		Jmp	PCINT1	;PCINT1
0x000C		Jmp	PCINT2	;PCINT2

中断向量初始化程序代码示例（接上页）

Address	Labels	Code		Comments
0x000E		Jmp	PCINT3	;PCINT3
0x0010		Jmp	WDT	;Watch dog Timeout
		.DB	0, 0, 0	;Reserved
0x0018		Jmp	ICF1	;Timer1 Input Capture
0x001A		Jmp	COM1A	;Timer1 CompareA
0x001C		Jmp	COM1B	;Timer1 CompareB
0x001E		Jmp	TOV1	;Timer1 Overflow
0x0020		Jmp	OC0	;Timer0 Compare
0x0022		.DB	0	;Reserved
0x0024		Jmp	TOV0	;Timer0 Overflow
0x0026		.DB	0	;Reserved
0x0028		Jmp	RXC0	;USART RX Complete
0x002A		Jmp	UDR0	;USART UDR Empty
0x002C		Jmp	TXC0	;USART TX Complete
0x002E		Jmp	ACP	;Analog Compare
0x0030		Jmp	ADC	;ADC Convert Complete
0x0032		Jmp	EEP	;EEPROM Ready
0x0034		.DB	0	; Reserved
0x0036		Jmp	RTC	;RTC Underflow
			……	
	RESET:	Ldi	R16, high(RAMEND)	;Main program start
		Out	SPH, r16	;Set Stack pointer to top of
		Ldi	R16, low(RAMEND)	;RAM
		Out	SPL, r16	
		SEI		;Enable interrupts
			…….	

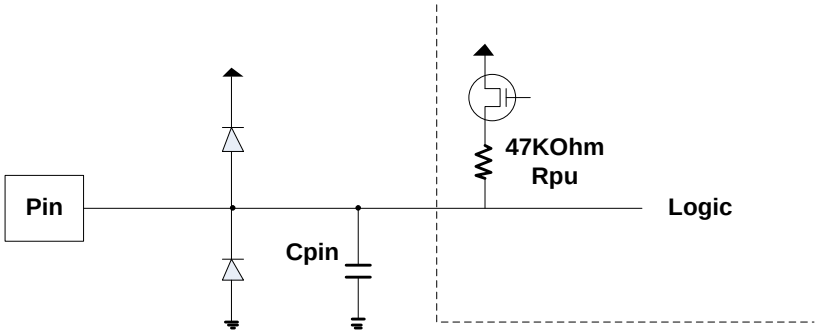
通用 I/O 端口

本章介绍 LGT8F0XA 系列的可编程通用 I/O 资源，LGT8F0XA 系列最大封装为 SOP28L，除去电源、地以及外部复位专用引脚外，其余 I/O 均可作为通用 I/O 使用。系统上电后，所有 I/O 默认配置为通用 I/O 模式。用户使能外设功能后，相应的 I/O 将会被外设专用的功能所替代。 **MVR8X 内核从指令上对 I/O 控制做了深入的优化，仅需一条指令即可改变任意一个 I/O 的输出状态，仅需一条指令即可根据当前端口的状态进行程序流程控制。这些针对 I/O 的优化，使得 LGT8F0XA 系列 MCU 非常适用于控制类应用，减少了程序代码，提高的效率。这是其他同类 MCU 无法相比的。**

概述

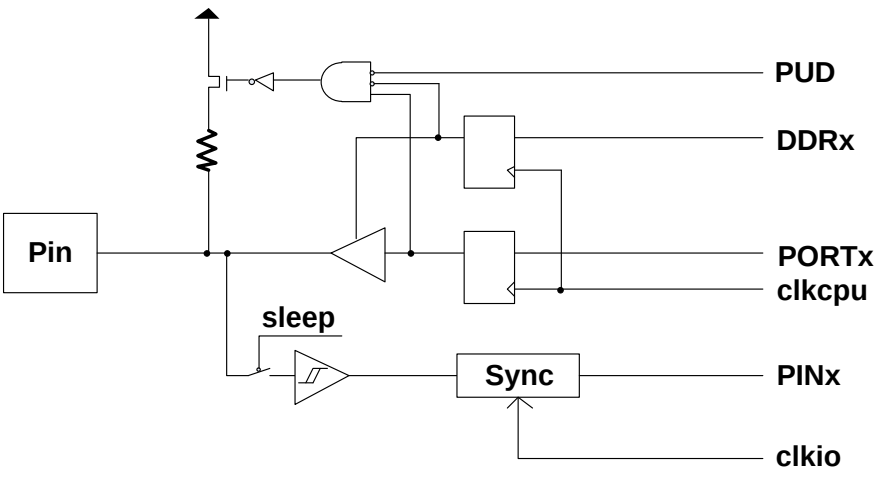
LGT8F0XA 系列所有的端口，当用作通用 I/O 时，支持**单周期读-改-写**操作。可以使用 SBI/CBI 指令在不改变其他任何端口状态的情况下，修改某一个 I/O 的状态（包括方向和驱动）。这种特性同样适用于修改单个端口的输出和上拉电阻(配置为输入)。端口的驱动能力可以根据应用配置为强驱动或弱驱动。在配置为强驱动时，端口可以驱动高达 15mA 的电流，可直接用于驱动 LED。所有的端口都具有典型值为 47KOhm 的可编程上拉电阻，这个电阻的值会随着供电电压略有变化。所有的 I/O 均支持施密特触发输入，提高了系统抗干扰的能力。

端口等效功能示意图：



LGT8F0XA 系列 MCU 最多包含 4 组通用可编程 I/O，分被位 PA、PB、PC 以及 PD。每组 I/O 有三个专用寄存器用于控制端口驱动，读取端口状态。这三个寄存器被分配到 I/O 空间的最低地址开始，因此可以使用指令集中的位操作指令。这三个寄存器分别为：端口输出数据寄存器 – PORTx， 端口方向寄存器- DDRx， 以及端口数据寄存器 – PINx，(x = A/B/C/D)。 其中端口数据寄存器 PINx 为只读寄存器，实时反映端口电平的变化。另外，系统还设有一个全局的上拉禁止控制位 – PUD，PUD 位于 MCUCR 寄存器。用户设置此位为 1 后，所有端口的上拉电阻将被强制关闭。

端口驱动结构图



端口配置

LGT8F0XA 系列 MCU 的每个通用可编程端口均为双向端口。每个端口对应三个寄存器位：DDRxn、PORTxn 以及 PINxn。寄存器细节请参考寄存器描述部分。DDRxn 通过 DDRx 寄存器访问，PORTxn 通过 PORTx 寄存器访问，PINxn 通过寄存器 PINx 访问。（x=A/B/C/D， n=0/1/2/3/4/5/6/7）。

DDRxn 用于控制端口的方向。当 DDRxn 为设置为 1，Pxn 端口将被配置为输出端口。如果 DDRxn 设置为 0，Pxn 端口将被配置为输入端口。

当 Pxn 被设置为输入端口后，PORTxn 写 1 将打开端口的上拉电阻。可以通过将 PORTxn 清零或将端口设为输出将上拉电阻关闭。系统上电复位后，所有端口均默认为输入三态，输出高阻。

当端口 Pxn 被配置为输出端口，PORTxn 位写 1 将端口驱动为高电平；将 PORTxn 清零将端口驱动为低电平。

当用户需要改变端口的输入输出状态时，需要特别注意，需要首先设置端口的输入输出状态(DDRx)，然后修改端口的驱动(PORTx)。否则将会在端口上产生一个窄的脉冲，对系统电路造成干扰。
在配置端口的上拉电阻时，也会遇到同样的问题，用户需要在配置好端口的输入输出状态后，才能改变其他端口的控制。

端口配置与状态对照表：

DDRxn	PORTxn	PUD	I/O 方向	上拉电阻	说明
0	0	X	输入	关闭	三态(Hi-Z)
0	1	0	输入	打开	扇入
0	1	1	输入	关闭	三态(Hi-Z)
1	0	X	输出	关闭	输出低（扇入电流）
1	1	X	输出	关闭	输出高（扇出电流）

无论 DDRxn 的设置如何，端口的状态都可以通过 PINx 寄存器读出。PINx 上的同步器仅在 I/O 时钟的下降沿对信号进行了锁存，使得 PINx 能够及时的反映 PORTx 的设置以及端口的状态更新。
端口在设置了 PORTx 后，可以在下一个周期及时的读到端口的状态变化，而不需要任何的等待周期。

端口配置实例代码：

汇编代码实例

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi  r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
ldi  r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
out  PORTB, r16
out  DDRB, r17
; Read port pins
in   r16, PINB
...
```

C 语言代码实例

```
unsigned char   i;
/* Define pull-ups and set outputs high and directions for port pins */
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
/* Read port pins */
i = PINB;
```

I/O 应用注意事项

对于没有使用的引脚，建议给这些引脚一个确定的电平。及时我们在系统处于休眠模式时，内部关闭的这些引脚的输入使能，但是浮空的引脚仍然会在系统处于其他模式时产生漏电。

最简单的方法是打开这些引脚的内部上拉电阻。在系统上电或复位过程中，内部上拉的关闭的，如果用户需要在系统上电时仍然保持较低的功耗，建议使用外部的上拉或下拉。直接将不使用的引脚接到 VCC 或 GND 上的做法是不建议的，因为不正确的引脚配置可能会产生非常大的电流，导致芯片功能损坏。

SWD 双线调试接口默认为使能状态，建议用户在最终产品中，通过设置 MCUCR 寄存器中的 SWDD 位关闭 SWD 调试功能，以避免外部不可预知的干扰。

考虑到软件能够及时获得端口信号的变化， PINA/B/C/D 输入为异步设计，直接反应端口的状态。但这样的做法也同时带来了一些信号同步的问题。建议用户在直接使用端口状态进行程序流程控制时，采用必要的软件滤波算法。 同时也避免直接使用 PINA/B/C/D 寄存器的值进行程序流程控制，而是首先使用 IN 指令将其读到内部寄存器，然后使用寄存器的值作后续处理。下面将给出建议的使用端口作为程序流程控制的代码

汇编代码实例

```
Label:
    in    r16, PINA
    ; Skip if PINA[1] is clear, avoid to use sbic directly
    sbrc    r16, 1
    ; Skip if PINA[1] is set, avoid to use sbis directly
    sbrs    r16,1
     .....
```

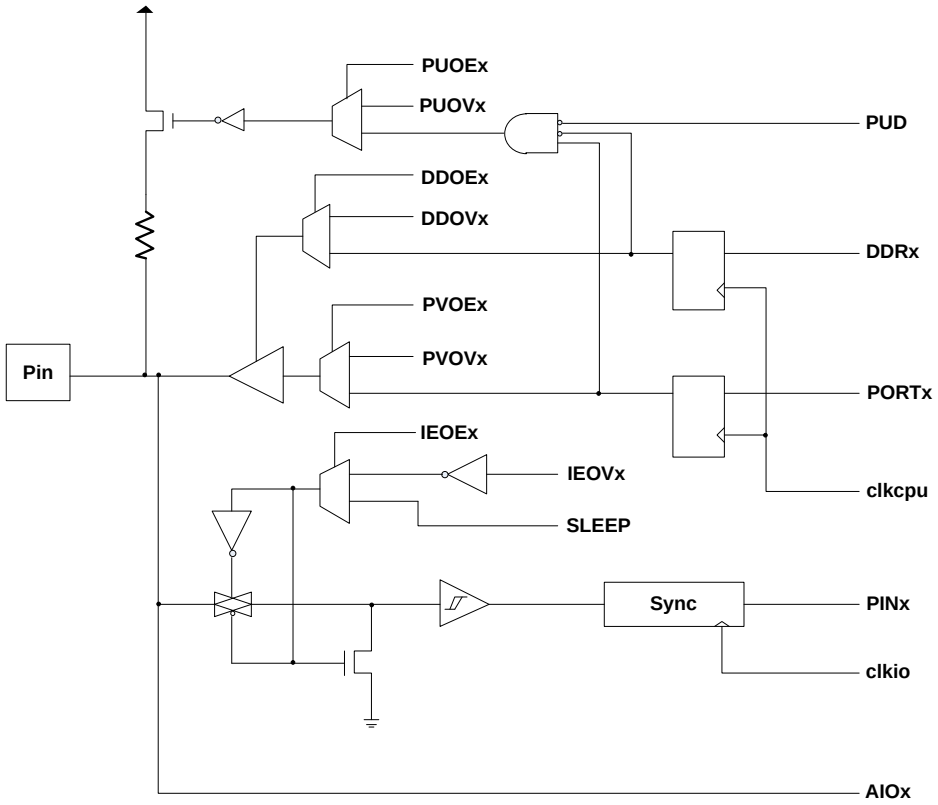
C 语言代码实例

```
volatile unsigned char pv; /*volatile is necessary to disable optimize on ‘pv’ */
/* wait until PINA[1] is set */
do {
      pv = PINA;
} while ((pv&0x2) != 0x02);    /* avoid to use while(PINA&0x2) */
```

端口功能复用

LGT8F0XA 系列微处理器的大部分引脚都具有多重功能。下图说明了端口功能与对应的控制信号之间的关系。并不是所有的复用控制信号都会出现在全部的端口上，下图只是用来简单的说明 LGT8F0XA 系列微处理器的端口一般关系。

端口复用控制图



下面的表格概况了端口复用控制信号的具体功能。这些复用控制信号由控制器内部自动产生。用户在正确的配置功能模块后，相应的端口复用控制就会生效。

端口复用功能控制信号的描述

信号	功能	描述
PUOE	上拉重载使能	当 PUOE 置位时，上拉使能由 PUOV 控制。 当 PUOE 清零时，上拉使能由{DDRxn, PORTxn, PUD}控制。
PUOV	上拉重载值	若 PUOE 置位，当 PUOV 置位时，上拉使能；当 PUOV 清零时，上拉禁止。 若 PUOE 清零，PUOV 的值无效。
DDOE	数据方向重载使能	当 DDOE 置位时，数据方向由 DDOV 控制。 当 DDOE 清零时，数据方向由 DDRxn 控制。
DDOV	数据方向重载值	若 DDOE 置位，当 DDOV 置位时，数据方向为输出；当 DDOV 清零时，数据方向为输入。 若 DDOE 清零时，DDOV 的值无效。
PVOE	端口数据重载使能	当 PVOE 置位时，端口数据由 PVOV 控制。 当 PVOE 清零时，端口数据由 PORTxn 控制。
PVOV	端口数据重载值	若 PVOE 置位，当 PVOV 置位时，端口数据为高；当 PVOV 清零时，端口数据为低。 若 PVOE 清零，PVOV 的值无效。
IEOE	输入使能重载使能	当 IEOE 置位时，输入使能由 IEOV 控制。 当 IEOE 清零时，输入使能由 MCU 的状态（睡眠与否）控制。
IEOV	输入使能重载值	若 IEOE 置位，当 IEOV 置位时，输入使能；当 IEOV 清零时，输入禁止。
DI	数字输入	复用功能的数字输入。
AIO	模拟输入输出	模拟输入输出。信号直接与引脚相连，可用作双向端口。

端口 A 引脚复用功能的控制 PA3-PA0

信号	PA3/ADC3/PCINT3	PA2/ADC2/PCINT2	PA1/ADC1/PCINT1	PA0/ADC0/PCINT0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
IEOE	PCIE0 & PCINT3 & ~ADC3D	PCIE0 & PCINT2 & ~ADC2D	PCIE0 & PCINT1 & ~ADC1D	PCIE0 & PCINT0 & ~ADC0D
IEOV	1	1	1	1
DI	PCINT3	PCINT2	PCINT1	PCINT0
AIO	ADC3	ADC2	ADC1	ADC0

端口 A 引脚复用功能的控制 PA7-PA4

信号	PA7/ADC7/PCINT7	PA6/ADC6/PCINT6	PA5/ADC5/PCINT5	PA4/ADC4/PCINT4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
IEOE	PCIE0 & PCINT7 & ~ADC7D	PCIE0 & PCINT6 & ~ADC6D	PCIE0 & PCINT5 & ~ADC5D	PCIE0 & PCINT4 & ~ADC4D
IEOV	1	1	1	1
DI	PCINT7	PCINT6	PCINT5	PCINT4
AIO	ADC7	ADC6	ADC5	ADC4

端口 B 引脚复用功能的控制 PB3-PB0

信号	PB3/AIN1/PCINT11	PB2/AIN0/PCINT10	PB1/ PCINT9	PB0/ PCINT8
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	XCKO_EN
PVOV	0	0	0	XCKO
IEOE	PCIE1 & PCINT11 & ~AIN1D	PCIE1 & PCINT10 & ~AIN0D	PCIE1 & PCINT9 + EXT1_EN	PCIE1 & PCINT8 + EXT0_EN + XCKI_EN
IEOV	1	1	1	1
DI	PCINT11	PCINT10	PCINT9/T1	PCINT8/T0/XCKI
AIO	AIN1	AIN0	-	-

端口 C 引脚复用功能的控制 PC3-PC0

信号	PC3/ PCINT19	PC2/ PCINT18	PC1/ PCINT17	PC0/ PCINT16
PUOE	SWD_EN	SWD_EN	0	0
PUOV	1	0	0	0
DDOE	SWD_EN	SWD_EN	0	0
DDOV	~SWDO	0	0	0
PVOE	SWD_EN	0	0	0
PVOV	0	0	0	0
IEOE	PCIE2 & PCINT19 + SWD_EN	PCIE2 & PCINT18	PCIE2 & PCINT17 & ~OSCM_EN	PCIE2 & PCINT16 & ~OSCM_EN

端口 C 引脚复用功能的控制 PC3-PC0 (接上页)

IEOV	1	1	1	1
DI	PCINT19 SWDI	PCINT18 SWCI	PCINT17	PCINT16
AIO	-	-	XTAL2	XTAL1

端口 C 引脚复用功能的控制 PC6

信号	PC6/ PCINT22
PUOE	0
PUOV	0
DDOE	0
DDOV	0
PVOE	OC0_EN
PVOV	OC0
IEOE	PCIE2 & PCINT22 + WAKUP_EN
IEOV	1
DI	PCINT22 WAKUP
AIO	-

端口 D 引脚复用功能的控制 PD3-PD0

信号	PD3/PCINT27	PD2/PCINT26	PD1/PCINT25	PD0/PCINT24
PUOE	0	0	TXEN	RXEN
PUOV	0	0	0	PORTD0 & ~PUD
DDOE	0	0	TXEN	RXEN
DDOV	0	0	1	0
PVOE	0	0	TXEN	0
PVOV	0	0	TXD	0
IEOE	PCIE0 & PCINT27 + EINT1_EN	PCIE0 & PCINT26 + EINT0_EN	PCIE0 & PCINT25	PCIE0 & PCINT24 + RXEN
IEOV	1	1	1	1
DI	PCINT27 INT1	PCINT26 INT0	PCINT25	PCINT24 RXD
AIO	-	-	-	-

端口 D 引脚复用功能的控制 PD7-PD4

信号	PD7/ PCINT31	PD6/PCINT30	PD5/PCINT29	PD4/PCINT28
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	OC1A_EN	OC1B_EN
PVOV	0	0	OC1A	OC1B
IEOE	PCIE0 & PCINT31 + EINT2_EN	PCIE0 & PCINT30 + ICP1_EN	PCIE0 & PCINT29	PCIE0 & PCINT28
IEOV	1	1	1	1
DI	PCINT31 INT2	PCINT30 ICP1	PCINT29	PCINT28
AIO	-	-	-	-

寄存器定义

PORTA – 端口 A 驱动输出寄存器

地址: 0x02			默认值: 0x00
Bit	Name	W/R	描述
7:0	PORTA[7:0]	W/R	A 组端口输出驱动寄存器

DDRA – 端口 A 输入/输出方向控制寄存器

地址: 0x01			默认值: 0x00
Bit	Name	W/R	描述
7:0	DDRA[7:0]	W/R	控制 A 组端口的输入、输出方向

PINA – 端口 A 输入寄存器

地址: 0x00			默认值: N/A
Bit	Name	W/R	描述
7:0	PINA[7:0]	W/R	直接反应端口 A 的电平状态

PORTB – 端口 B 驱动输出寄存器

地址: 0x05			默认值: 0x00
Bit	Name	W/R	描述
3:0	PORTB[3:0]	W/R	B 组端口输出驱动寄存器
7:4	PORTB[7:4]	-	端口 B 位 4 位端口，此处为保留位

DDRB – 端口 B 输入/输出方向控制寄存器

地址: 0x04			默认值: 0x00
Bit	Name	W/R	描述
3:0	DDRB[3:0]	W/R	控制 B 组端口的输入、输出方向
7:4	DDRB[7:4]	-	端口 B 位 4 位端口，此处为保留位

PINB – 端口 B 输入寄存器

地址: 0x03			默认值: N/A
Bit	Name	W/R	描述
3:0	PINB[3:0]	W/R	直接反应端口 B 的电平状态
7:4	PINB[7:4]	-	端口 B 位 4 位端口，此处为保留位

PORTC – 端口 C 驱动输出寄存器

地址: 0x08			默认值: 0x00
Bit	Name	W/R	描述
3:0	PORTC[3:0]	W/R	C 组端口输出驱动寄存器
5:4	PORTC[5:4]	-	保留位
6	PORTC[6]	W/R	C 组端口 6 输出驱动寄存器
7	PORTC[7]	-	保留位

DDRC – 端口 C 驱动输出寄存器

地址: 0x07			默认值: 0x00
Bit	Name	W/R	描述
3:0	DDRC[3:0]	W/R	C 组端口输入/输出方向寄存器
5:4	DDRC[5:4]	-	保留位
6	DDRC[6]	W/R	C 组端口 6 输入/输出方向寄存器
7	DDRC[7]	-	保留位

PINC – 端口 C 驱动输出寄存器

地址: 0x06			默认值: N/A
Bit	Name	W/R	描述
3:0	PINC[3:0]	W/R	C 组端口输入寄存器
5:4	PINC[5:4]	-	保留位
6	PINC[6]	W/R	C 组端口 6 输入寄存器
7	PINC[7]	-	保留位

PORTD – 端口 D 驱动输出寄存器

地址: 0x05			默认值: 0x00
Bit	Name	W/R	描述
7:0	PORTD[7:0]	W/R	D 组端口输出驱动寄存器

DDRD – 端口 D 输入/输出方向控制寄存器

地址: 0x04			默认值: 0x00
Bit	Name	W/R	描述
7:0	DDRD[7:0]	W/R	控制 D 组端口的输入、输出方向

PIND – 端口 D 输入寄存器

地址: 0x03			默认值: N/A
Bit	Name	W/R	描述
7:0	PIND[7:0]	W/R	直接反应端口 D 的电平状态

通用同步/异步串行收发器(USART)

主要特点

- 全双工操作（独立的串行接收和发送寄存器）
- 异步或同步操作
- 主机或从机操作
- 高精度的波特率发生器
- 支持 5，6，7，8，或 9 个数据位和 1，或 2 个停止位
- 硬件支持的奇偶产生和校验机制
- 数据过速检测
- 帧错误检测
- 噪声滤波，包括错误的起始位检测以及数字低通滤波器
- 三个独立的中断，包括：发送结束中断，发送数据寄存器空中断，以及接收结束中断
- 多处理器通信模式
- 倍速异步通信模式

综述

当设置 UCSRC 的 UMSEL1 位为“0”时，使能 USART 工作模式。USART 占用 USPI 的资源，主要包括三个部分：时钟发生器，发送器和接收器。控制和状态寄存器由这三个部分共享。时钟发生器由波特率发生器和同步从机操作模式下外部输入时钟的同步逻辑组成。XCK 引脚只用于同步传输模式。发送器包括一个写数据缓冲器，串行移位寄存器，奇偶发生器以及处理不同帧格式所需的控制逻辑。写数据缓冲器允许连续发送数据而不会在数据帧之间引入延迟。接收器具有时钟和数据恢复单元，用于异步数据的接收。除了恢复单元，接收器还包括奇偶校验，控制逻辑，串行移位寄存器和一个两级接收缓冲器 UDR。接收器支持与发送器相同的帧格式，而且可以检测帧错误，数据过速和奇偶校验错误。

时钟发生器

时钟产生逻辑为发送器和接收器产生基础时钟。USART 支持 4 种模式的时钟：正常的异步模式，倍速的异步模式，主机同步模式，以及从机同步模式。USCRC 的 UMSEL0 位用于选择同步或异步模式。USCRA 的 U2X 位控制异步模式下的倍速使能。仅在同步模式下有效的 XCK 引脚的数据方向寄存器(与 IO 复用)决定了时钟源是由内部产生(主机模式)还是外部产生(从机模式)。

波特率发生器

波特率寄存器 UBRR 和降序计数器连接在一起作为 USART 的可编程的预分频器或波特率发生器。降序计数器工作在系统时钟(f_{sys})下，当其计数到零或 UBRRL 寄存器被写时，会自动加载 UBRR 寄存器的值。当计数到零时产生一个时钟，该时钟作为波特率发生器的输出时钟，频率为 f_{sys}/(UBRR+1)。

下表给出了各种工作模式下计算波特率（位/秒）以及 UBRR 值的公式。

工作模式	波特率计算公式(1)	UBRR 值计算公式
异步正常模式	BAUD = f_{sys}/(16*(UBRR+1))	UBRR = f_{sys}/(16*BAUD) - 1
异步倍速模式	BAUD = f_{sys}/(8*(UBRR+1))	UBRR = f_{sys}/(8*BAUD) - 1
同步主机模式	BAUD = f_{sys}/(2*(UBRR+1))	UBRR = f_{sys}/(2*BAUD) - 1

说明：

1. 波特率定义为每秒的位传输速度（bps）；
2. BUAD 为波特率，f_{sys} 为系统时钟，UBRR 为波特率寄存器 UBRRH 和 UBRRL 的组合值。

倍速工作模式

通过设定 UCSRA 寄存器的 U2X 位可以是传输速率加倍，该位只在异步工作模式下有效，同步工作模式下置该位为“0”。

设置该位将会把波特率分频器的分频值减半，有效地加倍异步通信的传输速率。在这种情况下，接收器只使用一半的采样数来对数据进行采样及时钟恢复，因此需要更精准的波特率设置和系统时钟。发送器则没有变化。

外部时钟

同步从机操作模式由外部时钟驱动。外部时钟经过同步寄存器和边沿检测器之后才被发送器和接收器使用，这一过程会引入两个系统时钟的延时，因此外部 XCK 的最大时钟频率由以下公式限制：

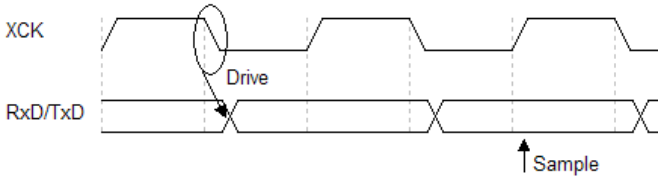
$$f_{XCK} < f_{sys}/4$$

要注意 fsys 有系统时钟的稳定性决定，为了防止因频率漂移而丢失数据，建议保留足够的裕量。

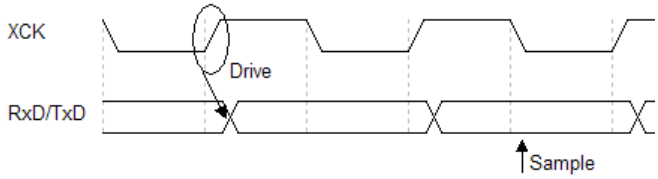
同步时钟操作

同步模式下，XCK 引脚被用于时钟输入（从机模式）或时钟输出（主机模式）。时钟的边沿与数据采样和数据变化关系的基本规律是：对数据输入端（RxD）采样所使用的时钟沿与数据输出端变化所使用的时钟沿是相反的。

UCPOL = 1



UCPOL = 0



同步模式下的 XCK 时序

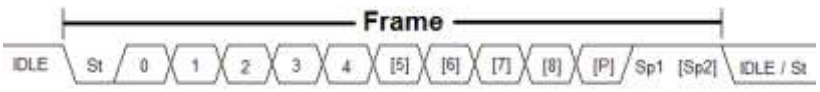
如上图所示，当 UCPOL 值为“1”时，在 XCK 的下降沿改变数据输出，在 XCK 的上升沿进行数据采样；当 UCPOL 值为“0”时，在 XCK 的上升沿改变数据输出，在 XCK 的下降沿进行数据采样。

帧格式

一个串行数据帧由数据字加上同步位（起始位和停止位）以及用于纠错的奇偶校验位构成。USART 接受以下 30 种组合的数据帧格式：

- 1 个起始位
- 5、6、7、8 或 9 个数据位
- 无校验位、奇校验位或偶校验位
- 1 或 2 个停止位

数据帧以起始位开始，紧接着是数据字的最低位，接着是其它数据位，以数据字的最高位结束，最多成功传输 9 位数据。如果使能了校验，校验位将紧接着数据字，最后是停止位。当一个完整的数据帧传输后，可以立即传输下一个新的数据帧，或者使传输线处于空闲（高电平）状态。下图为可能的数据帧结构，方括号中的位是可选的。



说明：

1）IDLE 通信线（RxD 或 TxD）上没有数据传输，线路空闲时必须为高电平

2）St 起始位，总是为低电平

3）0-8 数据位

4）P　　校验位，
奇校验或偶校验
5）Sp　　停止位，总是为高电平
数据帧的结构由 UCSRB 和 UCSRC 寄存器中的 UCSZ[2:0]、UPM[1:0]和 USBS 设定。接收与发送使用相同的设置。设置的任何改变都可能破坏正在进行的数据传输。其中，UCSZ[2:0]确定了数据帧的数据位数，UPM[1:0]用于使能和确定校验的类型，USBS 设置帧有一位或两位结束位。接收器会忽略第二个停止位，因此帧错误只在第一个结束位为“0”时被检测到。

校验位计算

校验位的计算是对数据的各个位进行异或运算。如果选择了奇校验，则异或结果还需要取反。校验位与数据位的关系如下：

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

说明：
1）P_{even} 偶校验结果
2）P_{odd} 奇校验结果
3）d_n　第 n 个数据位

USART 初始化

进行通信之前首先要对 USART 进行初始化。初始化过程通常包括波特率的设定，帧结构的设定，以及根据需要使能接收器或发送器。对于中断驱动的 USART 操作，在初始化时要清零全局中断标志并禁止 USART 的所有中断。在进行重新初始化比如改变波特率或帧结构时，必须确保没有数据传输。TXC 标志位可以用来检测发送器是否完成了所有传输，RXC 标志位可以用来检测接收缓冲器中是否还有数据未被读出。如果 TXC 标志位用作此用途，在每次发送数据之前（写 UDR 寄存器之前）必须清零 TXC 标志位。

发送器

置位 UCSRB 寄存器的 TXEN 位将使能 USART 的数据发送。使能后 TxD 引脚的通用 IO 功能即被 USART 功能所取代，成为发送器的串行输出。发送数据之前要设置好波特率、工作模式与帧格式。如果使用同步发送模式，施加于 XCK 引脚上的时钟信号即为数据发送的时钟。

发送 5 到 8 位数据的帧

将需要发送的数据加载到发送缓冲器中来启动数据发送。CPU 通过写 UDR 寄存器来加载数据。当发送移位寄存器可以发送新一帧数据的时候，缓冲器中的数据将转移到移位寄存器中。当移位寄存器处于空闲状态（没有正在进行的数据传输），或者前一帧数据的最后一个停止位发送完毕，它将加载新的数据。一旦移位寄存器加载了新的数据，它将按照既定的设置传输一个完整的帧。

发送 9 位数据的帧

如果发送 9 位数据的帧，应先将数据的第 9 位写入寄存器 UCSRB 的 TXB8 位，然后再将低 8 位数据写入发送数据寄存器 UDR。第 9 位数据在多机通信中用于表示地址帧，在同步通信中可以用于协议处理。

发送奇偶校验位

奇偶校验产生电路为串行数据帧生成相应的校验位。当校验位使能时（UPM1 = 1），发送控制逻辑电路会在数据字的最后一位与第一个停止位之间插入奇偶校验位。

发送标志位与中断处理

USART 发送器有两个标志位：USART 数据寄存器空标志 UDRE 和传输结束标志 TXC，两个标志位都可以产生中断。数据寄存器空标志 UDRE 用来表示发送缓冲器是否可以写入一个新的数据。该位在发送缓冲器空时被置“1”，满时被置“0”。当 UDRE 位为“1”时，CPU 可以往数据寄存器 UDR 写入新的数据，反之则不能。
当 UCSRB 寄存器中的数据寄存器空中断使能位 UDRIE 为“1”时，只要 UDRE 被置位（且全局中断使能），就将产生 USART 数据寄存器空中断请求。对寄存器 UDR 执行写操作将清零 UDRE。当采用中断方式传输数据时，在数据寄存器空中断服务程序中必须写入一个新的数据到 UDR 以清零 UDRE，或者是禁止数据寄存器空中断。否则一旦该中断服务程序结束，一个新的中断将再次产生。

禁止发送器

整个数据帧被移出发送移位寄存器，同时发送寄存器中又没有新的数据时，发送结束标志 TXC 将被置位。当 UCSRB 上的发送结束中断使能位 TXCIE（且全局中断使能）置“1”时，随着 TXC 标志位被置位，USART 发送结束中断将被执行。一旦进入中断服务程序，TXC 标志位即被自动清零，CPU 也可以对该位写“1”来清零。

接收器

当 TXEN 清零后，只有等所有的数据都发送完成以后发送器才能够真正禁止，即发送移位寄存器与发送缓冲寄存器中都没有要传送的数据。发送器禁止以后，TxD 引脚恢复其通用 IO 功能。

置位 UCSRB 寄存器的接收允许位（RXEN）即可启动 USART 接收器。使能后 RxD 引脚的通用 IO 功能被 USART 功能所取代，成为接收器的串行输入口。进行数据接收之前首先要设置好波特率、操作模式及帧格式。如果使用同步接收模式，XCK 引脚上的时钟被用为传输时钟。

接收 5 到 8 位数据的帧

一旦接收器检测到一个有效的起始位，便开始接受数据。起始位后的每一位数据都将以所设定的波特率或 XCK 时钟来进行接收，直到收到一帧数据的第一个停止位，第二个停止位会被接收器忽略。接收到的每一位数据被送入接收移位寄存器，收到第一个停止位以后，接收器置位位于 UCSRA 寄存器的接收数据完成标志 RXC 位，并把移位寄存器中完整的数据帧转移到接收缓冲器中，CPU 通过读取 UDR 寄存器就可以获得接收到的数据。

接收 9 位数据的帧

如果设定了 9 位数据的数据帧，在从 UDR 读取低 8 位数据之前必须首先读取寄存器 UCSRB 的 RXB8 位来获得第 9 位数据。这个规则同样适用于状态标志位 FE、DOR 以及 PE。读取 UDR 存储单元会改变接收缓冲器的状态，进而改变同样存储于缓冲器中的 TXB8、FE、DOR 及 PE 位。

接收结束及中断

USART 接收器有一个标志位：接收结束标志 RXC，用来表明接收缓冲器中是否有未被读出的数据。当接收缓冲器中有未被读出的数据时，此位为“1”，反之为“0”。如果接收器被禁止，接收缓冲器会被刷新，RXC 也会被清零。置位 UCSRB 的接收结束中断使能位 RXCIE 后，只要 RXC 标志被置位（且全局中断被使能），就会产生 USART 接收结束中断。使用中断方式进行数据接收时，数据接收结束中断服务程序必须从 UDR 读取数据来清零 RXC 标志，否则只要中断处理程序一结束，一个新的中断就会产生。

接收器错误标志

USART 接收器有三个错误标志：帧错误 FE、数据溢出 DOR 及奇偶校验错误 PE。它们都位于 UCSRA 寄存器。错误标志与数据帧一起保存在接收缓冲器当中。所有的错误标志都不能产生中断。

帧错误标志 FE 表明存储在接收缓冲器中的下一个可读帧的第一个停止位的状态。停止位正确（值为“1”）则 FE 标志为“0”，否则 FE 标志为“1”。这个标志可用来检测同步丢失、传输中断，也可用于协议处理。

数据溢出标志 DOR 表明由于接收缓冲器满造成了数据丢失。当接收缓冲器为满，接收移位寄存器中已有数据，若此时检测到一个新的起始位，数据溢出就产生了。DOR 标志被置位即表明在最近一次读取 UDR 和下一次读取 UDR 之间丢失了一个或多个数据帧。当数据帧成功地从移位寄存器转入接收缓冲器后，DOR 标志被清零。

奇偶校验错标志 PE 表明接收缓冲器中的下一帧数据在接收时有奇偶错误。如果不使能奇偶校验，PE 被清零。

奇偶校验器

置位奇偶校验模式位 UPM1 将启动奇偶校验器。校验的模式（偶校验或奇校验）由 UPM0 决定。奇偶校验使能后，校验器将计算输入数据的奇偶并把结果与数据帧的奇偶位进行比较。校验结果将与数据和停止位一起存储在接收缓冲器中。CPU 通过读取 PE 位来检查接收的帧当中是否有奇偶错误。如果下一个从接收缓冲器中读出的数据有奇偶错误，并且奇偶校验使能，则 UPE 被置位，一直有效到接收缓冲器 UDR 被读取。

禁止接收器

与发送器相比，禁止接收器即刻起作用。正在接收的数据将丢失。禁止接收器（RXEN 清零）后，接收器将不再占用 RxD 引脚，接收缓冲器也会被刷新。

异步数据接收

USART 有一个时钟恢复单元和数据恢复单元来处理异步数据接收。时钟恢复逻辑用于同步从 RxD 引脚输入的异步串行数据和内部的波特率时钟。数据恢复逻辑用于采集数据，并通过低通滤波器过滤所输入的每一位数据，从而提高接收器的抗干扰性能。异步接收的工作范围依赖于内部波特率时钟的精度、帧输入的速率及一帧所包含的数据位数。

异步工作范围

接收器的工作范围依赖于接收到的数据速率与内部波特率之间的不匹配程度。如果发送器以过快或过慢的比特率传输数据，或者接收器内部产生的波特率没有相同的频率，那么接收器就无法与起始位同步。为了确保接收器不会错过下一帧起始位的采样，数据输入速率和内部接收器波特率不能相差太大，用它们之间的比值来描述波特率的误差范围。下面两个表格分别给出了普通模式下和倍速模式下容许的最大波特率误差范围。

普通模式下最大接收器波特率误差范围

数据位+奇偶位长度和	最大误差范围（%）	推荐误差范围（%）
5	+6.7/-6.8	±3.0
6	+5.8/-5.9	±2.5
7	+5.1/-5.2	±2.0
8	+4.6/-4.5	±3.0
9	+4.1/-4.2	±1.5
10	+3.8/-3.8	±1.5

倍速模式下最大接收器波特率误差范围

数据位+奇偶位长度和	最大误差范围（%）	推荐误差范围（%）
5	+5.7/-5.9	±2.5
6	+4.9/-5.1	±2.0
7	+4.4/-4.5	±1.5
8	+3.9/-4.0	±1.5
9	+3.5/-3.6	±1.0
10	+3.2/-3.3	±1.0

从表中可以看出，普通模式下波特率允许有更大的变化范围。上述推荐的波特率误差范围是假定接收器和发送器对最大总误差具有同等贡献的前提下得出的。产生接收器波特率误差的可能原因有两个。首先，接收器系统时钟的稳定性与工作电压和温度有关。使用晶振来产生系统时钟时一般不会有此问题，但使用内部振荡器时，系统时钟可能会有偏差。第二个原因是波特率发生器不一定能通过对系统时钟的分频来得到恰好想要的波特率。此时可以调整 UBRR 的值，使得误差低至可以接受。

波特率设置及引入误差

对于标准晶振及谐振器频率来说，异步模式下的实际通信的波特率可通过波特率计算公式来获得，它与常用通信波特率之间的误差可用如下公式来计算：

$$Error[\%] = (Baud_{real}/Baud - 1)*100\%$$

其中，Baud 为常用的通信波特率，$Baud_{real}$ 为通过计算公式算出来的波特率，带入波特率计算公式即可得到波特率误差与系统时钟 f_{sys} 和波特率寄存器 UBRR 值之间的关系如下：

普通模式：

$$Error[\%] = (f_{sys}/(16*(UBRR+1))/Baud - 1)*100\%$$

倍速模式：

$$Error[\%] = (f_{sys}/(8*(UBRR+1))/Baud - 1)*100\%$$

当不考虑通信两边的时钟误差，即系统时钟 f_{sys} 为标准时钟时，即可得到波特率误差 UBRR 值之间的关系。下表即为 16MHz 系统时钟下不同 UBRR 值设置下的波特率误差。

16MHz 系统时钟下设置 UBRR 值所产生的误差

波特率(bps)	f_{sys} = 16.000MHz			
	普通模式(U2X = 0)		倍速模式(U2X = 1)	
	UBRR	误差	UBRR	误差
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4K	68	0.6%	138	-0.1%
19.2K	51	0.2%	103	0.2%
28.8K	34	-0.8%	68	0.6%
38.4K	25	0.2%	34	-0.8%
57.6K	16	2.1%	51	0.2%
76.8K	12	0.2%	25	0.2%
115.2K	8	-3.5%	16	2.1%
230.4K	3	8.5%	8	-3.5%
250K	3	0%	7	0%
0.5M	1	0%	3	0%
1M	0	0%	1	0%

多处理器通信模式

置位 UCSRA 的多处理器通信模式（MPCM）位可以对 USART 接收器接收到的数据帧进行过滤。那些没有地址信息的帧将被忽略，也不会存入接收缓冲器。在一个多处理器系统中，各处理器通过相同的串行总线进行通信，这种过滤有效的减少了需要 CPU 处理的数据帧的数量。MPCM 位的设置不影响发送器的工作，但在多处理器通信的系统中，它的使用方法会有所不同。

如果接收器所接收的数据帧长度为 5 到 8 位，那么第一个停止位会用来表示当前帧包含的是数据还是地址信息。如果接收器所接收的数据帧长度是 9 位，那么由第 9 位来确定是数据还是地址信息。如果帧类型标志位为“1”，那么这是地址帧，否则为数据帧。

在多处理器通信模式下，允许多个从处理器从一个主处理器接收数据。首先要通过解码地址帧来确定所寻址的是哪一个从处理器。被寻址的从处理器将正常接收后续的数据，而其他的从处理器则会忽略这些数据帧直到接收到下一个地址帧。

对于一个作为主机的处理器来说，它可以使用 9 位数据帧格式，并用第 9 位数据来标识帧格式。在这种通信模式下，从处理器也必须工作于 9 位数据帧格式。

下面即为多处理器通信模式下进行数据交换的步骤：

1. 所有从处理器都工作在多处理器通信模式（置位 MPCM）；
2. 主处理器发送地址帧，所有从处理器都接收此帧。从处理器 UCSRA 寄存器的 RXC 位正常置位；
3. 每个从处理器都读取 UDR 寄存器的内容，解码地址帧来确定是否被选中。如果选中，就清零 UCSRA 寄存器的 MPCM 位，未被选中，则保持 MPCM 为“1”并等待下一个地址帧的到来；
4. 被寻址的从处理器接收所有的数据帧，直到收到一个新的地址帧。未被寻址的从处理器忽略这些数据帧；
5. 被寻址的从处理器收到最后一个数据帧后，置位 MPCM 位，并等待下一个地址帧的到来。然后从第二步骤重复进行。

使用 5 到 8 位数据的帧格式是可以的，但是不切实际，因为接收器必须在使用 n 和 n+1 帧格式之间进行切换。由于接收器和发送器使用相同的字符长度设置，这种设置使得全双工操作变得很困难。如果使用 5 到 8 位数据的帧格式，发送器应该设置两个停止位，其中第一个停止位被用于判断帧类型。

寄存器描述

UCSRA – USART 控制和状态寄存器 A

地址: 0xC0			默认值: 0x20
Bit	Name	R/W	描述
7	RXC	R/W	接收结束。 当 RXC 的值为“1”时，表明接收缓冲器中有未读出的数据。当 RXC 的值为“0”时，表明接收缓冲器中没有未读出的数据。接收器禁止时，接收缓冲器被刷新，导致 RXC 被清零。当接收结束中断使能位 RXCIE 为“1”时，RXC 可用来产生接收结束中断。
6	TXC	R	发送结束。 发送移位寄存器中的数据被送出，且发送缓冲器为空时 TXC 置位。执行发送结束中断时 TXC 自动清零，也可以通过对 TXC 写“1”来进行清零。当发送结束中断使能位 TXCIE 为“1”时，TXC 可用来产生发送结束中断。
5	UDRE	R	数据寄存器空。 当 UDRE 为“1”时，表明 USART 发送数据缓冲器为空，可以写入数据。当 UDRE 为“0”时，表明 USART 发送数据缓冲器为满，不能写入数据。当数据寄存器空中断使能位 UDRIE 为“1”时，UDRE 可用来产生数据寄存器空中断。
4	FE	R	帧错误。 当 FE 为“1”时，表明接收数据缓冲器接收到的数据有帧错误，即第一个停止位为“0”。当 FE 为“0”时，表明接收数据缓冲器接收到的数据没有帧错误，即第一个停止位为“1”。FE 被置位后会一直有效到 UDR 被读取。对 UCSRA 进行写入时，FE 这一位要写“0”。
3	DOR	R	数据溢出。 当接收缓冲器为满（包含了两个数据），接收移位寄存器中有数据，若此时检测到一个新的起始位，数据溢出产生，DOR 被置位，一直有效到 UDR 被读取。对 UCSRA 进行写入时，DOR 这一位要写“0”。
2	PE	R	奇偶校验错误。 当奇偶校验使能（UPM1 为“1”）时，且接收缓冲器中所接收到的数据帧有奇偶校验错误，PE 被置位，一直有效到 UDR 被读取。对 UCSRA 进行写入时，PE 这一位要写“0”。
1	U2X	W	倍速发送。 当 U2X 为“1”时，异步通信模式的传输速率加倍。当 U2X 为“0”时，异步通信模式的传输速率为普通速率。 这一位仅在异步操作模式下有效，使用同步操作模式时将此位清零。
0	MPCM	R/W	多处理器通信模式。 设置 MPCM 位将启动多处理器通信模式。MPCM 置位后，USART 接收器接收到的那些不包含地址信息的输入帧都将被忽略。发送器不受 MPCM 设置的影响。

UCSRB – USART 控制和状态寄存器 B

地址: 0xC1			默认值: 0x00
Bit	Name	R/W	描述
7	RXCIE	R/W	接收结束中断使能。 置位后使能 RXC 中断，清零后禁止 RXC 中断。当 RXCIE 为“1”，全局中断使能，UCSRA 寄存器的 RXC 为“1”时可以产生 USART 接收结束中断。
6	TXCIE	R/W	发送结束中断使能。 置位后使能 TXC 中断，清零后禁止 TXC 中断。当 TXCIE 为“1”，全局中断使能，UCSRA 寄存器的 TXC 为“1”时可以产生 USART 发送结束中断。
5	UDRIE	R/W	数据寄存器空中断使能。 置位后使能 UDRE 中断，清零后禁止 UDRE 中断。当 UDRIE 为“1”，全局中断使能，UCSRA 寄存器的 UDRE 为“1”时可以产生 USART 数据寄存器空中断。
4	RXEN	R/W	接收使能。 置位后启动 USART 接收器。RxD 引脚的通用 IO 功能被 USART 接收所取代。禁止接收器将刷新接收缓冲器，并使 FE、DOR 及 PE 标志无效。
3	TXEN	R/W	发送使能。 置位后启动 USART 发送器。TxD 引脚的通用 IO 功能被 USART 发送所取代。TXEN 清零后，只有等到所有的数据发送完成后才能够真正禁止 USART 发送。
2	UCSZ2	R/W	字符长度控制第 2 位。 UCSZ2 与 UCSRC 寄存器的 UCSZ1:0 结合在一起设置数据帧所包含的数据位数。
1	RXB8	R/W	接收数据第 8 位。 当数据帧长度为 9 位时，RXB8 是接收数据的最高位。读取 UDR 所包含的低 8 位数据之前要先读取 RXB8。
0	TXB8	R/W	发送数据第 8 位。 当数据帧长度为 9 位时，TXB8 是发送数据的最高位。写入 UDR 所包含的低 8 位数据之前要先写入 TXB8。

UCSRC– USART 控制和状态寄存器 C

地址: 0xC2			默认值: 0x86	
Bit	Name	R/W	描述	
7:6	UMSEL1:0	R/W	USART 模式选择。 高位 UMSEL1 选择 USART 或 SPI 操作模式，UMSEL0 选择同步或异步操作模式或者主机从机操作模式。	
			UMSEL1:0	模式
			0	USART 异步操作模式
			1	USART 同步操作模式
			2	SPI 从机操作模式
			3	SPI 主机操作模式

UCSRC– USART 控制和状态寄存器 C (接上页)			
地址: 0xC2			默认值: 0x86
Bit	Name	R/W	描述
5:4	UPM1:0	R/W	奇偶校验模式选择。 高位 UPM1 选择使能或禁止奇偶校验，低位 UPM0 选择奇校验或偶校验。 UPM1:0 — 模式 0 — 禁止奇偶校验 1 — 保留 2 — 使能偶校验 3 — 使能奇校验
3	USBS	R/W	停止位选择。 选择停止位的位数。 USBS — 停止位位数 0 — 1 1 — 2
2:1	UCSZ1:0	R/W	数据帧字符长度选择。 UCSZ1:0 与 UCSRB 寄存器的 UCSZ2 结合起来设置数据帧包含的数据位数。 UCSZ2:0 — 数据帧长度 0 — 5 位 1 — 6 位 2 — 7 位 3 — 8 位 4 — 保留 5 — 保留 6 — 保留 7 — 9 位
0	UCPOL	R/W	时钟极性选择。 在 USART 同步工作模式下，UCPOL 设置了输出数据的改变和输入数据的采样与同步时钟 XCK 之间的关系。使用异步工作模式下与 UCPOL 无关，将这一位清零 UCPOL — 发送数据改变 — 接收数据采样 0 — XCK 的上升沿 — XCK 的下降沿 1 — XCK 的下降沿 — XCK 的上升沿

UBRRL – USART 波特率寄存器低字节

地址: 0xC4			默认值: 0x00
Bit	Name	R/W	描述
7:0	UBRRL	R/W	USART 波特率寄存器的低字节部分。 USART 波特率寄存器包含 UBRRL 和 UBRRH 两部分，结合在一起用来设置通信的波特率。

UBRRH – USART 波特率寄存器高字节

<table>
<tr><td colspan="3">地址: 0xC5</td><td colspan="2">默认值: 0x00</td></tr>
<tr><td>Bit</td><td>Name</td><td>R/W</td><td colspan="2">描述</td></tr>
<tr><td rowspan="5">7:0</td><td rowspan="5">UBRRH</td><td rowspan="5">R/W</td><td colspan="2">USART 波特率寄存器的高字节部分。
USART 波特率寄存器包含 UBRRL 和 UBRRH 两部分，结合在一起用来设置通信的波特率。
UBRR = {UBRRH, UBRRL}</td></tr>
<tr><td>工作模式</td><td>波特率计算公式</td></tr>
<tr><td>异步正常模式</td><td>BAUD = f_{sys}/(16*(UBRR+1))</td></tr>
<tr><td>异步倍速模式</td><td>BAUD = f_{sys}/(8*(UBRR+1))</td></tr>
<tr><td>同步主机模式</td><td>BAUD = f_{sys}/(2*(UBRR+1))</td></tr>
</table>

UDR – USART 数据寄存器

地址: 0xC6			默认值: 0x00
Bit	Name	R/W	描述
7:0	UDR	R/W	USART 发送和接收的数据。 USART 发送数据缓冲器和接收数据缓冲器共享 USART 数据寄存器 UDR。将数据写入 UDR 即写入发送数据缓冲器，从 UDR 读取数据即读取接收数据缓冲器。 在 5 到 8 位数据帧模式下，未使用的第 9 位被发送器忽略，而接收器则将它们设置为 0。 只有当 UCSRA 寄存器的 UDRE 标志为“1”时才能对发送缓冲器进行写操作，否则发送器的操作会出错。当发送移位寄存器为空时，发送器会把发送缓冲器中的数据加载到发送移位寄存器中，然后数据串行地从 TxD 引脚输出。 接收缓冲器包含一个两级 FIFO，一旦接收缓冲器被读取，FIFO 就会改变它的状态。

通用同步串行接口(USPI)

主要特点

- 全双工，三线同步数据传输
- 主机或从机操作模式
- 支持全部四种工作模式（模式 0、1、2 和 3）
- 低位或高位首先传输（可配置的数据传输顺序）
- 队列操作（双缓冲器）
- 高分辨率波特率产生器

综述

当设置 USCRC 的 UMSEL1 位为“1”时，使能 SPI 工作模式。SPI 占用 USPI 的资源，包括发送和接收移位寄存器和缓冲器，以及波特率发生器。奇偶校验产生和检查逻辑，数据和时钟恢复逻辑均无效。控制和状态寄存器的地址是一样的，不过寄存器位的功能会随着 SPI 工作模式的需要而发生改变。

时钟发生器

当 SPI 工作在主机模式时，需要提供通信用的时钟，复用 USART 的波特率发生器来产生这个时钟。该时钟从 XCK 引脚输出，因此 XCK 引脚的数据方向寄存器（DDR_XCK）必须设置为“1”。

时钟频率有以下计算公式决定：

$$BAUD = f_{sys}/(2*(UBRR+1))$$

当 SPI 工作在从机模式时，通信时钟由外部主机提供，从 XCK 引脚输入，因此 XCK 引脚的数据方向寄存器（DDR_XCK）必须设置为“0”。

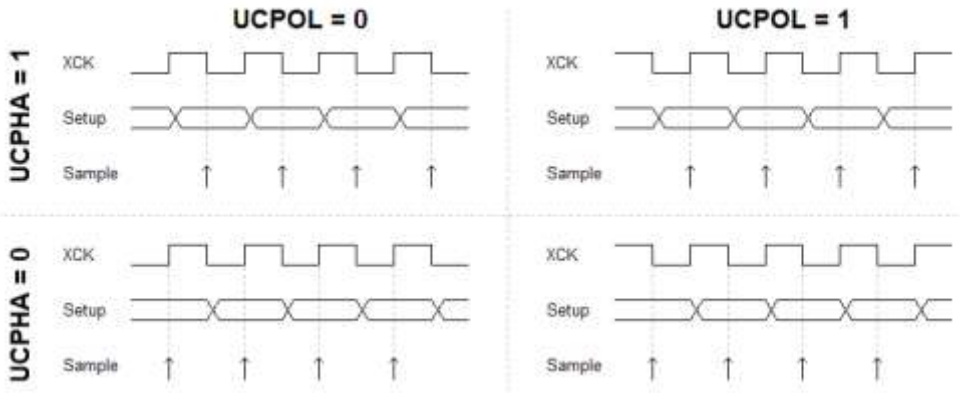
SPI 模式与时序

SPI 数据模式和时序

SPI 有四种时钟相位和极性的组合方式，有控制位 UCPHA 和 UCPOL 来决定，具体的控制如下表和下图所示：

SPI 工作模式

SPI 模式	UCPOL	UCPHA	起始沿	结束沿
0	0	0	上升沿采样	下降沿设置
1	0	1	上升沿设置	下降沿采样
2	1	0	下降沿采样	上升沿设置
3	1	1	下降沿设置	上升沿采样



SPI 工作模式图示

帧格式

SPI 的一个串行帧可以由最低位或最高位开始，到最高位或最低位结束，总共 8 位数据。一帧结束以后，可以紧接着传输新的一帧，传输结束即可拉高数据线为空闲状态。

数据传输

SPI 置 UCSRB 寄存器的 TXEN 位为“1”来使能发送器，TxD 引脚被发送器占用来发送串行输出数据。此时接收器可以不使能。SPI 置 UCSRB 寄存器的 RXEN 位为“1”来使能接收器，RxD 引脚被接收器占用来接收串行输入数据。此时发送器须使能。
SPI 发送和接收都使用 XCK 来当作传输时钟。

进行通信之前首先要对 SPI 进行初始化。初始化过程通常包括波特率的设定，帧数据位传输顺序的设定，以及根据需要使能接收器或发送器。对于中断驱动的 SPI 操作，在初始化时要清零全局中断标志并禁止 SPI 的所有中断。在进行重新初始化比如改变波特率或帧结构时，必须确保没有数据传输。TXC 标志位可以用来检测发送器是否完成了所有传输，RXC 标志位可以用来检测接收缓冲器中是否还有数据未被读出。如果 TXC 标志位用作此用途，在每次发送数据之前（写 UDR 寄存器之前）必须清零 TXC 标志位。

初始化 SPI 以后，往 UDR 寄存器写入数据即可开始数据传输。由于发送器控制着传输时钟，发送和接收数据均是如此操作。当发送移位寄存器准备好发送新一帧数据的时候，发送器就会把写入到 UDR 寄存器的数据从发送缓冲器移到发送移位寄存器里并发送出去。为了保证输入缓冲器和发送数据同步，每发送一个字节的数据后都必须读取一次 UDR 寄存器。当发生数据溢出时，最近收到的数据将会丢失，而不是最早收到的数据。

发送标志位与中断

SPI 发送器有两个标志位：SPI 数据寄存器空标志 UDRE 和传输结束标志 TXC，两个标志位都可以产生中断。
数据寄存器空标志 UDRE 用来表示发送缓冲器是否可以写入一个新的数据。该位在发送缓冲器空时被置“1”，满时被置“0”。当 UDRE 位为“1”时，CPU 可以往数据寄存器 UDR 写入新的数据，反之则不能。
当 UCSRB 寄存器中的数据寄存器空中断使能位 UDRIE 为“1”时，只要 UDRE 被置位（且全局中断使能），就将产生 SPI 数据寄存器空中断请求。对寄存器 UDR 执行写操作将清零 UDRE。当采用中断方式传输数据时，在数据寄存器空中断服务程序中必须写入一个新的数据到 UDR 以清零 UDRE，或者是禁止数据寄存器空中断。否则一旦该中断服务程序结束，一个新的中断将再次产生。
当整个数据帧被移出发送移位寄存器，同时发送寄存器中又没有新的数据时，发送结束标志 TXC 将被置位。当 UCSRB 上的发送结束中断使能位 TXCIE（且全局中断使能）置“1”时，随着 TXC 标志位被置位，SPI 发送结束中断将被执行。一旦进入中断服务程序，TXC 标志位即被自动清零，CPU 也可以对该位写“1”来清零

禁止发送器

当 TXEN 清零后，只有等所有的数据都发送完成以后发送器才能够真正禁止，即发送移位寄存器与发送缓冲寄存器中都没有要传送的数据。发送器禁止以后，TxD 引脚恢复其通用 IO 功能。

接收结束及中断

SPI 接收器有一个标志位：接收结束标志 RXC，用来表明接收缓冲器中是否有未被读出的数据。当接收缓冲器中有未被读出的数据时，此位为“1”，反之为“0”。如果接收器被禁止，接收缓冲器会被刷新，RXC 也会被清零。
置位 UCSRB 的接收结束中断使能位 RXCIE 后，只要 RXC 标志被置位（且全局中断被使能），就会产生 SPI 接收结束中断。使用中断方式进行数据接收时，数据接收结束中断服务程序必须从 UDR 读取数据来清零 RXC 标志，否则只要中断处理程序一结束，一个新的中断就会产生。

禁止接收器

与发送器相比，禁止接收器即刻起作用。正在接收的数据将丢失。禁止接收器（RXEN 清零）后，接收器将不再占用 RxD 引脚，接收缓冲器也会被刷新。

此 SPI 模块为三线 SPI 工作模式，与四线 SPI 模式相比，缺少从机选择线，其它三根线均一致。

寄存器描述

SPI 的寄存器和 USART 的寄存器共用，因此采用相同的寄存器名称和地址，但寄存器位的具体功能和含义不相同，参考下面各寄存器的详细描述。

UCSRA – SPI 控制和状态寄存器 A

地址: 0xC0			默认值: 0x20
Bit	Name	W/R	描述
7	RXC	RO	接收结束。 当 RXC 的值为“1”时，表明接收缓冲器中有未读出的数据。当 RXC 的值为“0”时，表明接收缓冲器中没有未读出的数据。接收器禁止时，接收缓冲器被刷新，导致 RXC 被清零。当接收结束中断使能位 RXCIE 为“1”时，RXC 可用来产生接收结束中断。
6	TXC	W/R	发送结束。 发送移位寄存器中的数据被送出，且发送缓冲器为空时 TXC 置位。执行发送结束中断时 TXC 自动清零，也可以通过对 TXC 写“1”来进行清零。当发送结束中断使能位 TXCIE 为“1”时，TXC 可用来产生发送结束中断。
5	UDRE	RO	数据寄存器空。 当 UDRE 为“1”时，表明 SPI 发送数据缓冲器为空，可以写入数据。当 UDRE 为“0”时，表明 SPI 发送数据缓冲器为满，不能写入数据。当数据寄存器空中断使能位 UDRIE 为“1”时，UDRE 可用来产生数据寄存器空中断。
4:0	-	-	SPI 模式下保留。

UCSRB – SPI 控制和状态寄存器 B

地址: 0xC1			默认值: 0x00
Bit	Name	W/R	描述
7	RXCIE	W/R	接收结束中断使能。 置位后使能 RXC 中断，清零后禁止 RXC 中断。当 RXCIE 为“1”，全局中断使能，UCSRA 寄存器的 RXC 为“1”时可以产生 SPI 接收结束中断。
6	TXCIE	W/R	发送结束中断使能。 置位后使能 TXC 中断，清零后禁止 TXC 中断。当 TXCIE 为“1”，全局中断使能，UCSRA 寄存器的 TXC 为“1”时可以产生 SPI 发送结束中断。
5	UDRIE	W/R	数据寄存器空中断使能。 置位后使能 UDRE 中断，清零后禁止 UDRE 中断。当 UDRIE 为“1”，全局中断使能，UCSRA 寄存器的 UDRE 为“1”时可以产生 SPI 数据寄存器空中断。
4	RXEN	W/R	接收使能。 置位后启动 SPI 接收器。RxD 引脚的通用 IO 功能被 SPI 接收所取代。禁止接收器将刷新接收缓冲器。
3	TXEN	W/R	发送使能。 置位后启动 SPI 发送器。TxD 引脚的通用 IO 功能被 SPI 发送所取代。TXEN 清零后，只有等到所有的数据发送完成后才能够真正禁止 SPI 发送。
2:0	-	-	SPI 模式下保留。

UCSRC– USART 控制和状态寄存器 C

地址: 0xC2			默认值: 0x86		
Bit	Name	W/R	描述		
7:6	UMSEL1:0	W/R	SPI 模式选择。 高位 UMSEL1 选择 USART 或 SPI 操作模式，UMSEL0 选择同步或异步操作模式或者主机从机操作模式。		
			UMSEL1:0	模式	
			0	USART 异步操作模式	
			1	USART 同步操作模式	
			2	SPI 从机操作模式	
			3	SPI 主机操作模式	
5:3	-	-	SPI 模式下保留。		
2	DORD	W/R	数据传输顺序选择。		
			DORD	数据顺序	
			0	高位先传输	
			1	低位先传输	
1	UCPHA	W/R	时钟相位选择。 UCPHA 选择数据采样发生在起始沿或结束沿。		
			UCPHA	采样时刻	
			0	起始沿	
			1	结束沿	
0	UCPOL	W/R	时钟极性选择。 UCPOL 选择数据改变和采样发生在上升沿或下降沿。		
			UCPOL	发送数据的改变	接收数据的采样
			0	XCK 的上升沿	XCK 的下降沿
			1	XCK 的下降沿	XCK 的上升沿

UBRRL – SPI 波特率寄存器低字节

地址: 0xC4			默认值: 0x00
Bit	Name	W/R	描述
7:0	UBRRL	W/R	SPI 波特率寄存器的低字节部分。 SPI 波特率寄存器包含 UBRRL 和 UBRRH 两部分，结合在一起用来设置通信的波特率。

UBRRH – SPI 波特率寄存器高字节

地址: 0xC5			默认值: 0x00	
Bit	Name	W/R	描述	
7:4	-	-	保留。	
3:0	UBRRH	W/R	SPI 波特率寄存器的高字节部分。 SPI 波特率寄存器包含 UBRRL 和 UBRRH 两部分，结合在一起用来设置通信的波特率。UBRR = {UBRRH, UBRRL}	
			工作模式	波特率计算公式
			主机模式	BAUD = f_{sys}/(2*(UBRR+1))
			从机模式	波特率由外部主机决定

UDR – SPI 数据寄存器

地址: 0xC6			默认值: 0x00
Bit	Name	W/R	描述
7:0	UDR	W/R	SPI 发送和接收的数据。 SPI 发送数据缓冲器和接收数据缓冲器共享 SPI 数据寄存器 UDR。将数据写入 UDR 即写入发送数据缓冲器，从 UDR 读取数据即读取接收数据缓冲器。 只有当 UCSRA 寄存器的 UDRE 标志为“1”时才能对发送缓冲器进行写操作，否则发送器的操作会出错。当发送移位寄存器为空时，发送器会把发送缓冲器中的数据加载到发送移位寄存器中，然后数据串行地从 TxD 引脚输出。 接收缓冲器包含一个两级 FIFO，一旦接收缓冲器被读取，FIFO 就会改变它的状态。

定时计数器 0 (TC0)

主要特点

- 单通道计数器
- 比较匹配发生时自动清零计数器并自动加载
- 无干扰脉冲的相位修正的 PWM 输出
- 频率发生器
- 外部事件计数器
- 10 位的时钟预分频器
- 溢出和比较匹配中断

综述

TC0 是一个通用 8 位定时计数器模块，支持 PWM 输出，可以精确地产生波形。TC0 包含 1 个 8 位计数器，波形产生模式控制单元和 1 个输出比较单元。波形产生模式控制单元控制着计数器的工作模式和比较输出波形的产生。根据不同的工作模式，计数器对每一个计数时钟 Clkt0 实现清零、加一或减一操作。Clkt0 可以由内部时钟源或外部时钟源产生。当计数器的计数值 TCNT0 到达最大值（等于极大值 0xFF 或输出比较寄存器 OCR0，定义为 TOP，定义极大值为 MAX 以示区别）时，计数器会进行清零或减一操作。当计数器的计数值 TCNT0 到达最小值（等于 0x00，定义为 BOTTOM）时，计数器会进行加一操作。当计数器的计数值 TCNT0 到达 OCR0 时，也被称为发生比较匹配时，会清零或置位输出比较信号 OC0，来产生 PWM 波形。

工作模式

定时计数器 0 有四种不同的工作模式，包括普通模式（Normal），比较匹配时清零（CTC）模式，快速脉冲宽度调制（FPWM）模式和相位修正脉冲宽度调制（PCPWM）模式，由波形产生模式控制位 WGM0[2:0]来选择。下面具体来描述这四种模式。

普通模式

普通模式是定时计数器最简单的工作模式，此时波形产生模式控制位 WGM0[2:0]=0，计数的最大值 TOP 为 MAX（0xFF）。在这种模式下，计数方式为每一个计数时钟加一递增，当计数器到达 TOP 溢出后就回到 BOTTOM 重新开始累加。在计数值 TCNT0 变成零的同一个计数时钟里置位定时计数器溢出标志 TOV0。这种模式下 TOV0 标志就像是第 9 计数位，只是只会被置位不会被清零。溢出中断服务程序会自动清除 TOV0 标志，软件可以用它来提高定时计数器的分辨率。普通模式下没有特殊情形需要考虑，可以随时写入新的计数值。

设置 OC0 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC0 的波形。当 COM0=1 时，发生比较匹配时会翻转 OC0 信号，这种情况下波形的频率可以用下面的公式来计算：

$$f_{oc0normal} = f_{sys}/(2*N*256)$$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

输出比较单元可以用来产生中断，但是在普通模式下不推荐使用中断，这样会占用太多 CPU 的时间。

CTC 模式

设置 WGM0[2:0]=2 时，定时计数器 0 进入 CTC 模式，计数的最大值 TOP 为 OCR0。在这个模式下，计数方式为每一个计数时钟加一递增，当计数器的数值 TCNT0 等于 TOP 时计数器清零。OCR0 定义了计数的最大值，亦即计数器的分辨率。这个模式使得用户可以很容易的控制比较匹配输出的频率，也简化了外部事件计数的操作。

当计数器到达计数的最大值时，输出比较匹配标志 OCF0 被置位，相应的中断使能置位时将会产生中断。在中断服务程序里可以更新 OCR0 寄存器即计数的最大值。在这个模式下 OCR0 没有使用双缓冲，在计数器以无预分频器或很低的预分频器工作下将最大值更新为接近最小值的时候要小心。如果写入 OCR0 的数值小于当时的 TCNT0 值时，计数器将丢失一次比较匹配。在下一次比较匹配发生之前，计数器不得不先计数到 TOP，然后再从 BOTTOM 开始计数到 OCR0 值。和普通模式一样，计数值回到 BOTTOM 的计数时钟里置位 TOV0 标志。

设置 OC0 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC0 的波形。当 COM0=1 时，发生比较匹配时会翻转 OC0 信号，这种情况下波形的频率可以用下面的公式来计算：

$$f_{oc0ctc} = f_{sys}/(2*N*(1+OCR0))$$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

从公式可以看出，当设置 OCR0 为 0x0 且无预分频器时，可以获得最大频率为 $f_{sys}/2$ 的输出波形。

快速 PWM 模式

设置 WGM0[1:0]=3 时，定时计数器 0 进入快速 PWM 模式，可以用来产生高频的 PWM 波形。当 WGM0[2]=0 时，计数最大值 TOP 为 MAX（0xFF），当 WGM0[2]=1 时，计数最大值 TOP 为 OCR0。快速 PWM 模式和其他 PWM 模式不同在于它是单向操作。计数器从最小值 0x00 累加到 TOP 后又回到 BOTTOM 重新计数。当计数值 TCNT0 到达 OCR0 或 BOTTOM 时，输出比较信号 OC0 会被置位或清零，取决于比较输出模式 COM0 的设置，详情见寄存器描述。由于采用单向操作，快速 PWM 模式的操作频率是采用双向操作的相位修正 PWM 模式的两倍。高频特性使得快速 PWM 模式适用于功率调节，整流以及 DAC 应用。高频信号可以减小外部元器件（电感电容等）的尺寸，从而降低系统成本。

当计数值到达最大值时，定时计数器溢出标志 TOV0 将会被置位，并把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR0 寄存器。

设置 OC0 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC0 的波形。波形的频率可用下面的公式来计算：

$$f_{oc0fpwm} = f_{sys}/(N*256)$$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

当 TCNT0 和 OCR0 发生比较匹配时，波形产生器就置位（清零）OC0 信号，当 TCNT0 被清零时，波形产生器就清零（置位）OC0 信号，以此来产生 PWM 波。由此 OCR0 的极值将会产生特殊的 PWM 波形。当 OCR0 设置为 0x00 时，输出的 PWM 为每 256 个计数时钟里有一个窄的尖峰脉冲。当 OCR0 设置为最大值时，输出的波形为持续的高电平或低电平。

相位修正 PWM 模式

当设置 WGM0[1:0]=1 时，定时计数器 0 进入相位修正 PWM 模式。当 WGM0[2]=0 时，计数的最大值 TOP 为 MAX（0xFF），当 WGM0[2]=1 时，计数的最大值 TOP 为 OCR0。计数器采用双向操作，由 BOTTOM 递增到 TOP，然后又递减到 BOTTOM，再重复此操作。计数到达 TOP 和 BOTTOM 时均改变计数方向，计数值在 TOP 或 BOTTOM 上均只停留一个计数时钟。在递增或递减过程中，计数值 TCNT0 与 OCR0 匹配时，输出比较信号 OC0 将会被清零或置位，取决于比较输出模式 COM0 的设置。与单向操作相比，双向操作可获得的最大频率要小，但其极好的对称性更适合于电机控制。

相位修正 PWM 模式下，当计数到达 BOTTOM 时置位 TOV0 标志，当计数到达 TOP 时把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR0 寄存器。

设置 OC0 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC0 的波形。波形的频率可用下面的公式来计算：

$$f_{oc0pcpwm} = f_{sys}/(N*510)$$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

在递增计数过程中，当 TCNT0 与 OCR0 匹配时，波形产生器就清零（置位）OC0 信号。在递减计数过程中，当 TCNT0 与 OCR0 匹配时，波形产生器就置位（清零）OC0 信号。由此 OCR0 的极值会产生特殊的 PWM 波。当 OCR0 设置为最大值或最小值时，OC0 信号输出会一直保持低电平或高电平。

为了保证输出 PWM 波在最小值两侧的对称性，在没有发生比较匹配时，有两种情况下也会翻转 OC0 信号。第一种情况是，当 OCR0 的值由最大值 0xFF 改变为其他数据时。当 OCR0 为最大值，计数值达到最大时，OC0 的输出与前面降序计数时比较匹配的结果相同，即保持 OC0 不变。此时会更新比较值为新的 OCR0 的值（非 0xFF），OC0 的值会一直保持，直到升序计数时发生比较匹配而翻转。此时 OC0 信号并不以最小值为中心对称，因此需要在 TCNT0 到达最大值时翻转 OC0 信号，此即没有发生比较匹配时翻转 OC0 信号的第一种情况。第二种情况是，当 TCNT0 从比 OCR0 高的值开始计数时，因而会丢失一次比较匹配，从而引起不对称情形的产生。同样需要翻转 OC0 信号去实现最小值两侧的对称性。

寄存器描述

TCCR0A – TC0 控制寄存器 A

地址: 0x24			默认值: 0x00
Bit	Name	W/R	描述
7:6	COM0[1:0]	W/R	比较匹配输出模式控制位。 COM0 用来控制输出比较波形。如果 COM0 的 1 位或者 2 位都置位，输出比较波形占据着 OC0 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM0 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
5:2	-	-	保留。
1:0	WGM0[1:0]	W/R	波形产生模式控制低两位。 WGM0[1:0]和 WGM0[2]一起组成波形产生模式控制 WGM0，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。

下表为非 PWM 模式（即普通模式和 CTC 模式）下，比较输出模式对输出比较波形的控制。

非 PWM 模式下比较输出模式控制

COM0[1:0]	描述
0	OC0 断开，通用 IO 口操作
1	比较匹配时翻转 OC0 信号
2	比较匹配时清零 OC0 信号
3	比较匹配时置位 OC0 信号

下表为快速 PWM 模式下比较输出模式对输出比较波形的控制。

快速 PWM 模式下比较输出模式控制

COM0[1:0]	描述
0	OC0 断开，通用 IO 口操作
1	WGM0[2] = 0：OC0 断开，通用 IO 口操作 WGM0[2] = 1：比较匹配时翻转 OC0 信号
2	比较匹配时清零 OC0 信号，最大值匹配时置位 OC0 信号
3	比较匹配时置位 OC0 信号，最大值匹配时清零 OC0 信号

下表为快速 PWM 模式下比较输出模式对输出比较波形的控制。

快速 PWM 模式下比较输出模式控制

COM0[1:0]	描述
0	OC0 断开，通用 IO 口操作
1	WGM0[2] = 0：OC0 断开，通用 IO 口操作 WGM0[2] = 1：比较匹配时翻转 OC0 信号
2	升序计数下比较匹配时清零 OC0 信号，降序计数下比较匹配时置位 OC0 信号
3	升序计数下比较匹配时置位 OC0 信号，降序计数下比较匹配时清零 OC0 信号

TCCR0B – TC0 控制寄存器 B

地址: 0x25			默认值: 0x00	
Bit	Name	W/R	描述	
7	FOC0	W/R	强制输出比较。 工作于非 PWM 模式时，可以通过对强制输出比较位 FOC0 写“1”的方式来产生比较匹配。强制比较匹配不会置位 OCF0 标志，也不会重载或清零定时器，但是输出引脚 OC0 将被按照 COM0 的设置相应的更新，就跟真的发生了比较匹配一样。 工作于 PWM 模式时，写 TCCR0B 寄存器时要对其清零。 读取 FOC0 的返回值一直为零。	
6:4	-	-	保留。	
3	WGM0[2]	W/R	波形产生模式控制高位。 WGM0[1:0]和 WGM0[2]一起组成波形产生模式控制 WGM0，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。	
2:0	CS0[2:0]	W/R	时钟选择控制位。 用于选择定时计数器 0 的时钟源	
			CS0[2:0]	描述
			0	无时钟源，停止计数
			1	clk_{sys}
			2	clk_{sys}/8，来自预分频器
			3	clk_{sys}/64，来自预分频器
			4	clk_{sys}/256，来自预分频器
			5	clk_{sys}/1024，来自预分频器
			6	外部时钟 T0 引脚，下降沿触发
			7	外部时钟 T0 引脚，上升沿触发

下表为波形产生模式控制。

波形产生模式控制

WGM0[2:0]	工作模式	TOP 值	更新 OCR0 时刻	置位 TOV0 时刻
0	Normal	0xFF	立即	MAX
1	PCPWM	0xFF	TOP	BOTTOM
2	CTC	OCR0	立即	MAX
3	FPWM	0xFF	TOP	MAX
4	保留	-	-	-
5	PCPWM	OCR0	TOP	BOTTOM
6	保留	-	-	-
7	FPWM	OCR0	TOP	TOP

TCNT0 – TC0 计数值寄存器

地址: 0x26			默认值: 0x00
Bit	Name	W/R	描述
7:0	TCNT0[7:0]	W/R	计数值寄存器。 通过 TCNT0 寄存器可以直接对计数器的 8 为计数值进行读写访问。CPU 对 TCNT0 寄存器的写操作会在下一个定时器时钟周期阻止比较匹配的发生，即使定时器已经停止。这就允许初始化 TCNT0 寄存器的值与 OCR0 的值一致而不会引发中断。 如果写入 TCNT0 的数值等于或绕过 OCR0 值时，比较匹配就会丢失，造成不正确的波形发生结果。 没有选择时钟源时定时器停止计数，但 CPU 仍可以访问 TCNT0。CPU 写计数器比清零或加减操作的优先级高。

OCR0A – TC0 输出比较寄存器

地址: 0x27			默认值: 0x00
Bit	Name	W/R	描述
7:0	OCR0A[7:0]	W/R	输出比较寄存器。 OCR0A 包含一个 8 位的数据，不间断地与计数器数值 TCNT0 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC0 引脚上产生波形。 当使用 PWM 模式时，OCR0A 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR0A 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR0A 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR0A 本身。

TIMSK0 – TC0 中断屏蔽寄存器

地址: 0x6E			默认值: 0x00
Bit	Name	W/R	描述
7:2	-	-	保留。
1	OCIE0A	W/R	输出比较匹配中断使能位。 当 OCIE0A 位为“1”，且全局中断置位，TC0 输出比较匹配中断使能。当比较匹配发生时，即 TIFR 中 OCF0A 位被置位时，中断产生。 当 OCIE0A 位为“0”时，TC0 输出比较匹配中断被禁止。
0	TOIE0	W/R	溢出中断使能位。 当 TOIE0 位为“1”，且全局中断置位，TC0 溢出中断使能。当 TC0 发生溢出，即 TIFR 中的 TOV0 位被置位时，中断产生。 当 TOIE0 位为“0”时，TC0 溢出中断被禁止。

TIFR0 – TC0 中断标志寄存器

地址: 0x15			默认值: 0x00
Bit	Name	W/R	描述
7:2	-	-	保留。
1	OCF0A	W/R	输出比较匹配标志位。 当 TCNT0 等于 OCR0A 时，比较单元就给出匹配信号，并置位比较标志 OCF0A。若此时输出比较中断使能 OCIE0 为“1”且全局中断标志置位，则会产生输出比较中断。执行此中断服务程序时 OCF0A 将自动清零，或对 OCF0A 位写“1”也可清零该位。
0	TOV0	W/R	溢出标志位。 当计数器发生溢出时，置位溢出标志 TOV0。若此时溢出中断使能 TOIE0 为“1”且全局中断标志置位，则会产生溢出中断。执行此中断服务程序时 TOV0 将自动清零，或对 TOV0 位写“1”也可清零该位。

定时计数器 1 (TC1)

主要特点

- 真正的 16 位设计，允许 16 位的 PWM
- 2 个独立的输出比较单元
- 双缓冲的输出比较寄存器
- 1 个输入捕捉单元
- 输入捕捉噪声抑制器
- 比较匹配时自动清零计数器并自动加载
- 无干扰脉冲的相位修正的 PWM
- 可变的 PWM 周期
- 频率发生器
- 外部事件计数器
- 4 个独立的中断源

综述

TC1 是一个通用 16 位定时计数器模块，支持 PWM 输出，可以精确地产生波形。TC1 包含 1 个 16 位计数器，波形产生模式控制单元，2 个独立的输出比较单元和 1 个输入捕捉单元。波形产生模式控制单元控制着计数器的工作模式和比较输出波形的产生。根据不同的工作模式，计数器对每一个计数时钟 Clkt1 实现清零、加一或减一操作。Clkt1 可以由内部时钟源或外部时钟源产生。当计数器的计数值 TCNT1 到达最大值（等于极大值 0xFFFF 或固定值或输出比较寄存器 OCR1A 或输入捕捉寄存器 ICR1，定义为 TOP，定义极大值为 MAX 以示区别）时，计数器会进行清零或减一操作。当计数器的计数值 TCNT1 到达最小值（等于 0x0000，定义为 BOTTOM）时，计数器会进行加一操作。当计数器的计数值 TCNT1 到达 OCR1A 或 OCR1B 时，也被称为发生比较匹配时，会清零或置位输出比较信号 OC1A 或 OC1B，来产生 PWM 波形。当开启输入捕捉功能时，计数器被触发即开始或停止计数，ICR1 寄存器会记录捕捉信号触发周期内的计数值。

工作模式

定时计数器 1 有六种不同的工作模式，包括普通模式（Normal），比较匹配时清零（CTC）模式，快速脉冲宽度调制（FPWM）模式，相位修正脉冲宽度调制（PCPWM）模式，相位频率修正脉冲宽度调制（PFCPWM）模式，和输入捕捉（ICP）模式。由波形产生模式控制位 WGM1[3:0]来选择。下面具体来描述这六种模式。由于有两个独立的输出比较单元，分别用“A”和“B”来表示，用小写的“x”来表示这两个输出比较单元通道。

普通模式

普通模式是定时计数器最简单的工作模式，此时波形产生模式控制位 WGM1[3:0]=0，计数的最大值 TOP 为 MAX（0xFFFF）。在这种模式下，计数方式为每一个计数时钟加一递增，当计数器到达 TOP 溢出后就回到 BOTTOM 重新开始累加。在计数值 TCNT1 变成零的同一个计数时钟里置位定时计数器溢出标志 TOV1。这种模式下 TOV1 标志就像是第 17 计数位，只是只会被置位不会被清零。溢出中断服务程序会自动清除 TOV1 标志，软件可以用它来提高定时计数器的分辨率。普通模式下没有特殊情形需要考虑，可以随时写入新的计数值。

设置 OC1x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 的波形。当 COM1x=1 时，发生比较匹配时会翻转 OC1x 信号，这种情况下波形的频率可以用下面的公式来计算：

$$f_{oc1xnormal} = f_{sys}/(2*N*65536)$$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

输出比较单元可以用来产生中断，但是在普通模式下不推荐使用中断，这样会占用太多 CPU 的时间。

CTC 模式

设置 WGM1[3:0]=4 或 12 时，定时计数器 1 进入 CTC 模式。当 WGM1[3]=0 时，计数最大值 TOP 为 OCR1A，当 WGM1[3]=1 时，计数最大值 TOP 为 ICR1。下面以 WGM1[3:0]=4 为例来描述 CTC 模式在这个模式下，计数方式为每一个计数时钟加一递增，当计数器的数值 TCNT1 等于 TOP 时计数器清零。这个模式使得用户可以很容易的控制比较匹配输出的频率，也简化了外部事件计数的操作。

当计数器到达 TOP 时，输出比较匹配标志 OCF1 被置位，相应的中断使能置位时将会产生中断。在中断服务程序里可以更新 OCR1A 寄存器。在这个模式下 OCR1A 没有使用双缓冲，在计数器以无预分频器或很低的预分频器工作下将最大值更新为接近最小值的时候要小心。如果写入 OCR1A 的数值小于当时的 TCNT1 值时，计数器将丢失一次比较匹配。在下一次比较匹配发生之前，计数器不得不先计数到 MAX，然后再从 BOTTOM 开始计数到 OCR1A。和普通模式一样，计数值回到 0x0 的计数时钟里置位 TOV1 标志。

设置 OC1x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 的波形。波形的频率可以用下面的公式来计算：

$$f_{oc1xctc} = f_{sys}/(2*N*(1+OCR1A))$$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

从公式可以看出，当设置 OCR1A 为 0x0 且无预分频器时，可以获得最大频率为 $f_{sys}/2$ 的输出波形。

当 WGM1[3:0]=12 时与 WGM1[3:0]=4 类似，只是把与 OCR1A 相关的换成 ICR1 即可。

快速 PWM 模式

设置 WGM1[3:0]=5，6，7，14 或 15 时，定时计数器 1 进入快速 PWM 模式，计数最大值 TOP 分别为 0xFF，0x1FF，0x3FF，ICR1 或 OCR1A，可以用来产生高频的 PWM 波形。快速 PWM 模式和其他 PWM 模式不同在于它是单向操作。计数器从 BOTTOM 累加到 TOP 后又回到 BOTTOM 重新计数。当计数值 TCNT1 到达 TOP 或 BOTTOM 时，输出比较信号 OC1x 会被置位或清零，取决于比较输出模式 COM1 的设置，详情见寄存器描述。由于采用单向操作，快速 PWM 模式的操作频率是采用双向操作的相位修正 PWM 模式的两倍。高频特性使得快速 PWM 模式适用于功率调节，整流以及 DAC 应用。高频信号可以减小外部元器件（电感电容等）的尺寸，从而降低系统成本。

当计数值到达 TOP 时，定时计数器溢出标志 TOV1 将会被置位，并把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新 OCR1A 寄存器。

设置 OC1x 引脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 的波形。波形的频率可用下面的公式来计算：

$$f_{oc1xfpwm} = f_{sys}/(N*(1+TOP))$$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

当 TCNT1 和 OCR1x 发生比较匹配时，波形产生器就置位（清零）OC1x 信号，当 TCNT1 被清零时，波形产生器就清零（置位）OC1x 信号，以此来产生 PWM 波。由此 OCR1x 的极值将会产生特殊的 PWM 波形。当 OCR1x 设置为 0x00 时，输出的 PWM 为每(1+TOP)个计数时钟里有一个窄的尖峰脉冲。当 OCR1x 设置为 TOP 时，输出的波形为持续的高电平或低电平。如果用 OCR1A 作为 TOP 并设置 COM1A=1，输出比较信号 OC1A 会产生占空比为 50%的 PWM 波。

相位修正 PWM 模式

当设置 WGM0[3:0]=1，2，3，10 或 11 时，定时计数器 1 进入相位修正 PWM 模式，计数的最大值 TOP 分别为 0xFF，0x1FF，0x3FF，ICR1 或 OCR1A。计数器采用双向操作，由 BOTTOM 递增到 TOP，然后又递减到 BOTTOM，再重复此操作。计数到达 TOP 和 BOTTOM 时均改变计数方向，计数值在 TOP 或 BOTTOM 上均只停留一个计数时钟。在递增或递减过程中，计数值 TCNT1 与 OCR1x 匹配时，输出比较信号 OC1x 将会被清零或置位，取决于比较输出模式 COM1 的设置。与单向操作相比，双向操作可获得的最大频率要小，但其极好的对称性更适合于电机控制。

相位修正 PWM 模式下，当计数到达 BOTTOM 时置位 TOV1 标志，当计数到达 TOP 时把比较缓冲器的值更新到比较值。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR1x 存器。

设置 OC1x 脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 波形。波形的频率可用下面的公式来计算：

$$f_{oc1xcpcpwm} = f_{sys}/(N*TOP*2)$$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

在递增计数过程中，当 TCNT1 与 OCR1x 匹配时，波形产生器就清零（置位）OC1x 信号。在递减计数过程中，当 TCNT1 与 OCR1x 匹配时，波形产生器就置位（清零）OC1x 信号。由此 OCR1x 的极值会产生特殊的 PWM 波。当 OCR1x 设置为 TOP 或 BOTTOM 时，OC1x 信号输出会一直保持低电平或高电平。如果用 OCR1A 作为 TOP 并设置 COM1A=1，输出比较信号 OC1A 会产生占空比为 50%的 PWM 波。

为了保证输出 PWM 波在 BOTTOM 两侧的对称性，在没有发生比较匹配时，有两种情况下也会翻转 OC1x 信号。第一种情况是，当 OCR1x 的值由 TOP 改变为其他数据时。当 OCR1x 为 TOP，计数值达到 TOP 时，OC1x 的输出与前面降序计数时比较匹配的结果相同，即保持 OC1x 不变。此时会更新比较值为新的 OCR1x 的值（非 TOP），OC1x 的值会一直保持，直到升序计数时发生比较匹配而翻转。此时 OC1x 信号并不以最小值为中心对称，因此需要在 TCNT1 到达最大值时翻转 OC1x 信号，此即没有发生比较匹配时翻转 OC1x 信号的第一种情况。第二种情况是，当 TCNT1 从比 OCR1x 高的值开始计数时，因而会丢失一次比较匹配，从而引起不对称情形的产生。同样需要翻转 OC1x 信号去实现最小值两侧的对称性。

相位频率修 PWM 模式

当设置 WGM0[3:0]=8 或 9 时，定时计数器 1 进入相位频率修正 PWM 模式，计数的最大值 TOP 分别为 ICR1 或 OCR1A。计数器采用双向操作，由 BOTTOM 递增到 TOP，然后又递减到 BOTTOM，再重复此操作。计数到达 TOP 和 BOTTOM 时均改变计数方向，计数值在 TOP 或 BOTTOM 上均只停留一个计数时钟。在递增或递减过程中，计数值 TCNT1 与 OCR1x 匹配时，输出比较信号 OC1x 将会被清零或置位，取决于比较输出模式 COM1 的设置。与单向操作相比，双向操作可获得的最大频率要小，但其极好的对称性更适合于电机控制。

相位频率修正 PWM 模式下，当计数到达 BOTTOM 时置位 TOV1 标志，并且把比较缓冲器的值更新到比较值，更新比较值的时间是相位频率修正 PWM 模式和相位修正 PWM 模式的最大不同点。如果中断使能，在中断服务程序中可以更新比较缓冲器 OCR1x 存器。当 CPU 改变 TOP 值即 ORC1A 或 ICR1 的值时，必须保证新的 TOP 值不小于已经在使用的 TOP 值，否则比较匹配将不会再发生。

设置 OC1x 脚的数据方向寄存器为输出时才能得到输出比较信号 OC1x 波形。波形的频率可用下面的公式来计算：

$$f_{oc1xcpfcpwm} = f_{sys}/(N*TOP*2)$$

其中，N 表示的是预分频因子（1，8，64，256 或者 1024）。

在递增计数过程中，当 TCNT1 与 OCR1x 匹配时，波形产生器就清零（置位）OC1x 信号。在递减计数过程中，当 TCNT1 与 OCR1x 匹配时，波形产生器就置位（清零）OC1x 信号。由此 OCR1x 的极值会产生特殊的 PWM 波。当 OCR1x 设置为 TOP 或 BOTTOM 时，OC1x 信号输出会一直保持低电平或高电平。如果用 OCR1A 作为 TOP 并设置 COM1A=1，输出比较信号 OC1A 会产生占空比为 50%的 PWM 波。

因为 OCR1x 寄存器是在 BOTTOM 时刻更新的，所以 TOP 值两边升序和降序的计数长度是一样的，也就产生了频率和相位都正确的对称波形。

当使用固定 TOP 值时，最好采用 ICR1 寄存器作为 TOP 值，即设置 WGM1[3:0]=8，此时 OCR1A 寄存器只需用来产生 PWM 输出。如果要产生频率变化的 PWM 波，必须通过改变 TOP 值， OCR1A 的双缓冲特性会更适合于这个应用。

输入捕捉模式

输入捕捉用来捕获外部事件，并为其赋予时间标记以说明此事件发生的时刻，可以在前面的计数模式下进行，不过要除去使用 ICR1 值作为计数 TOP 值的波形产生模式。

外部事件发生的触发信号由引脚 ICP1 输入，也可以通过模拟比较器单元来实现。当引脚 ICP1 上的逻辑电平发生变化，或模拟比较器的输出 ACO 电平发生变化，并且这个电平变化被输入捕捉单元所捕获，输入捕捉即被触发，此时 16 位的计数值 TCNT1 数据被复制到输入捕捉寄存器 ICR1，同时输入捕捉标志 ICF1 置位，若 ICIE1 位为“1”，输入捕捉标志将产生输入捕捉中断。

通过设置模拟比较控制与状态寄存器 ACSR 的模拟比较输入捕捉控制位 ACIC 来选择输入捕捉触发源 ICP1 或 ACO。需注意的是，改变触发源有可能造成一次输入捕捉，因此在改变触发源后必须对 ICF1 进行一次清零操作来避免出现错误的结果。

输入捕捉信号经过
一个可选的噪声抑制器之后送入边沿检测器，根据输入捕捉选择控制位 ICES1 的配置，看检测到的边沿是否满足触发条件。噪声抑制器是一个简单的数字滤波，对输入信号进行 4 次采样，只有当 4 次采样值都相等时其输出才会送入边沿检测器。噪声抑制器由 TCCR1B 寄存器的 ICNC1 位控制其使能或禁止。
使用输入捕捉功能时，当 ICF1 被置位后，应尽可能早的读取 ICR1 寄存器的值，因为下一次捕捉事件发生后 ICR1 的值将会被更新。推荐使能输入捕捉中断，在任何输入捕捉工作模式下，都不推荐在操作过程中改变计数 TOP 值。

输入捕捉到的时间标记可用来计算频率、占空比及信号的其它特征，以及为触发事件创建日志。测量外部信号的占空比时要求每次捕捉后都要改变触发沿，因此读取 ICR1 值以后须尽快改变触发的信号边沿。

寄存器描述

TCCR1A – TC1 控制寄存器 A

地址: 0x80			默认值: 0x00
Bit	Name	R/W	描述
7:6	COM1A[1:0]	R/W	比较匹配输出 A 模式控制位。 COM1A 用来控制输出比较波形 OC1A。如果 COM1A 的 1 位或者 2 位都置位，输出比较波形占据着 OC1A 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM1A 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
5:4	COM1B[1:0]	R/W	比较匹配输出 B 模式控制位。 COM1B 用来控制输出比较波形 OC1B。如果 COM1B 的 1 位或者 2 位都置位，输出比较波形占据着 OC1B 引脚，不过该引脚的数据方向寄存器必须置高才能输出此波形。在不同工作模式下，COM1B 对输出比较波形的控制也不同，具体见比较输出模式控制表格描述。
3	FOC1A	R/W	强制输出比较 A。 工作于非 PWM 模式时，可以通过对强制输出比较位 FOC1A 写“1”的方式来产生比较匹配。强制比较匹配不会置位 OCF1A 标志，也不会重载或清零定时器，但是输出引脚 OC1A 将被按照 COM1A 的设置相应的更新，就跟真的发生了比较匹配一样。 工作于 PWM 模式时，写 TCCR1A 寄存器时要对其清零。 读取 FOC1A 的返回值一直为零。
2	FOC1B	R/W	强制输出比较 B。 工作于非 PWM 模式时，可以通过对强制输出比较位 FOC1B 写“1”的方式来产生比较匹配。强制比较匹配不会置位 OCF1B 标志，也不会重载或清零定时器，但是输出引脚 OC1B 将被按照 COM1B 的设置相应的更新，就跟真的发生了比较匹配一样。 工作于 PWM 模式时，写 TCCR1A 寄存器时要对其清零。 读取 FOC1B 的返回值一直为零。
1:0	WGM1[1:0]	R/W	波形产生模式控制低两位。 WGM1[1:0]和 WGM1[3:2]一起组成波形产生模式控制 WGM1，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。

下表为非 PWM 模式（即普通模式和 CTC 模式）下，比较输出模式对输出比较波形的控制。

非 PWM 模式下比较输出模式控制

COM1x[1:0]	描述
0	OC1x 断开，通用 IO 口操作
1	比较匹配时翻转 OC1x 信号
2	比较匹配时清零 OC1x 信号
3	比较匹配时置位 OC1x 信号

下表为快速 PWM 模式下比较输出模式对输出比较波形的控制。

快速 PWM 模式下比较输出模式控制

COM1x[1:0]	描述
0	OC1x 断开，通用 IO 口操作
1	WGM1[3] = 0：OC1x 断开，通用 IO 口操作 WGM1[3] = 1：比较匹配时翻转 OC1x 信号
2	比较匹配时清零 OC1x 信号，最大值匹配时置位 OC1x 信号
3	比较匹配时置位 OC1x 信号，最大值匹配时清零 OC1x 信号

下表为相位修正模式下比较输出模式对输出比较波形的控制。

相位修正和相位频率修正 PWM 模式下比较输出模式控制

COM1x[1:0]	描述
0	OC1x 断开，通用 IO 口操作
1	WGM1[3] = 0：OC1x 断开，通用 IO 口操作 WGM1[3] = 1：比较匹配时翻转 OC1x 信号
2	升序计数下比较匹配清零 OC1x 信号，降序计数下比较匹配置位 OC1x 信号
3	升序计数下比较匹配置位 OC1x 信号，降序计数下比较匹配清零 OC1x 信号

TCCR1B – TC1 控制寄存器 B

地址: 0x81			默认值: 0x00
Bit	Name	R/W	描述
7	ICNC1	R/W	输入捕捉噪声抑制器使能控制位。 当设置 ICNC1 位为“1”时，使能输入捕捉噪声抑制器，此时外部引脚 ICP1 的输入被滤波，连续 4 个采样值相等时输入信号才有效，该功能使得输入捕捉被延迟了 4 个时钟周期。 当设置 ICNC1 位为“0”时，禁止输入捕捉噪声抑制器，此时外部引脚 ICP1 的输入直接有效。
6	ICES1	R/W	输入捕捉触发沿选择控制位。 当设置 ICES1 位为“1”时，选择电平的上升沿触发输入捕捉；当设置 ICES1 位为“0”时，选择电平的下降沿触发输入捕捉。 当捕获到一个事件后，计数器的数值被复制到 ICR1 寄存器，同时置位输入捕捉标志 ICF1。如果中断使能，产生输入捕捉中断。
5	-	-	保留。
4:3	WGM1[3:2]	R/W	波形产生模式控制高位。 WGM1[1:0]和 WGM1[3:2]一起组成波形产生模式控制 WGM1，控制计数器的计数方式和波形产生方式，具体见波形产生模式表格描述。

<table>
<tr><th colspan="5">TCCR1B – TC1 控制寄存器 B (接上页)</th></tr>
<tr><td colspan="3">地址: 0x81</td><td colspan="2">默认值: 0x00</td></tr>
<tr><th>Bit</th><th>Name</th><th>R/W</th><th colspan="2">描述</th></tr>
<tr><td rowspan="9">2:0</td><td rowspan="9">CS1[2:0]</td><td rowspan="9">R/W</td><td colspan="2">时钟选择控制位。
用于选择定时计数器 1 的时钟源</td></tr>
<tr><td>CS1[2:0]</td><td>描述</td></tr>
<tr><td>0</td><td>无时钟源，停止计数</td></tr>
<tr><td>1</td><td>clk_{sys}</td></tr>
<tr><td>2</td><td>clk_{sys}/8，来自预分频器</td></tr>
<tr><td>3</td><td>clk_{sys}/64，来自预分频器</td></tr>
<tr><td>4</td><td>clk_{sys}/256，来自预分频器</td></tr>
<tr><td>5</td><td>clk_{sys}/1024，来自预分频器</td></tr>
<tr><td>6</td><td>外部时钟 T1 引脚，下降沿触发</td></tr>
<tr><td>7</td><td>外部时钟 T1 引脚，上升沿触发</td></tr>
</table>

下表为波形产生模式控制。

波形产生模式控制

WGM1[3:0]	工作模式	TOP 值	更新 OCR0 时刻	置位 TOV0 时刻
0	Normal	0xFFFF	立即	MAX
1	8 位 PCPWM	0x00FF	TOP	BOTTOM
2	9 位 PCPWM	0x01FF	TOP	BOTTOM
3	10 位 PCPWM	0x03FF	TOP	BOTTOM
4	CTC	OCR1A	立即	MAX
5	8 位 FPWM	0x00FF	BOTTOM	TOP
6	9 位 FPWM	0x01FF	BOTTOM	TOP
7	10 位 FPWM	0x03FF	BOTTOM	TOP
8	PFCPWM	ICR1	BOTTOM	BOTTOM
9	PFCPWM	OCR1A	BOTTOM	BOTTOM
10	PCPWM	ICR1	TOP	BOTTOM
11	PCPWM	OCR1A	TOP	BOTTOM
12	CTC	ICR1	立即	MAX
13	保留	-	-	-
14	FPWM	ICR1	TOP	TOP
15	FPWM	OCR1A	TOP	TOP

TCNT1L – TC1 计数值寄存器低字节

地址: 0x84			默认值: 0x00
Bit	Name	R/W	描述
7:0	TCNT1[7:0]	R/W	TC1 计数值的低字节。 TCNT1H 和 TCNT1L 结合到一起组成 TCNT1，通过 TCNT1 寄存器可以直接对计数器的 16 位计数值进行读写访问。读写 16 位寄存器需要两次操作。写 16 位 TCNT1 时，应先写入 TCNT1H。读 16 位 TCNT1 时，应先读取 TCNT1L。 CPU 对 TCNT1 寄存器的写操作会在下一个定时器时钟周期阻止比较匹配的发生，即使定时器已经停止。这就允许初始化 TCNT1 寄存器的值与 OCR1x 的值一致而不会引发中断。 如果写入 TCNT1 的数值等于或绕过 OCR1x 值时，比较匹配就会丢失，造成不正确的波形发生结果。 没有选择时钟源时定时器停止计数，但 CPU 仍可以访问 TCNT1。CPU 写计数器比清零或加减操作的优先级高。

TCNT1H – TC1 计数值寄存器高字节

地址: 0x85			默认值: 0x00
Bit	Name	R/W	描述
7:0	TCNT1[15:8]	R/W	TC1 计数值的高字节。 TCNT1H 和 TCNT1L 结合到一起组成 TCNT1，通过 TCNT1 寄存器可以直接对计数器的 16 位计数值进行读写访问。读写 16 位寄存器需要两次操作。写 16 位 TCNT1 时，应先写入 TCNT1H。读 16 位 TCNT1 时，应先读取 TCNT1L。 CPU 对 TCNT1 寄存器的写操作会在下一个定时器时钟周期阻止比较匹配的发生，即使定时器已经停止。这就允许初始化 TCNT1 寄存器的值与 OCR1x 的值一致而不会引发中断。 如果写入 TCNT1 的数值等于或绕过 OCR1x 值时，比较匹配就会丢失，造成不正确的波形发生结果。 没有选择时钟源时定时器停止计数，但 CPU 仍可以访问 TCNT1。CPU 写计数器比清零或加减操作的优先级高。

ICR1L – TC1 输入捕捉值寄存器低字节

地址: 0x86			默认值: 0x00
Bit	Name	R/W	描述
7:0	ICR1[7:0]	R/W	TC1 输入捕捉值的低字节。 ICR1H 和 ICR1L 结合到一起组成 16 位的 ICR1。读写 16 位寄存器需要两次操作。写 16 位 ICR1 时，应先写入 ICR1H。读 16 位 ICR1 时，应先读取 ICR1L。 当输入捕捉被触发时，计数值 TCNT1 就会更新复制到 ICR1 寄存器里。ICR1 寄存器也可用来定义计数的 TOP 值。

OCR1AL –TC1 输出比较寄存器 A 低字节

地址: 0x88			默认值: 0x00
Bit	Name	R/W	描述
7:0	OCR1A[7:0]	R/W	输出比较寄存器 A 的低字节。 OCR1AL 和 OCR1AH 结合到一起组成 16 位的 OCR1A。读写 16 位寄存器需要两次操作。写 16 位 OCR1A 时，应先写入 OCR1AH。读 16 位 OCR1A 时，应先读取 OCR1AL。 OCR1A 不间断地与计数器数值 TCNT1 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC1A 引脚上产生波形。 当使用 PWM 模式时，OCR1A 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR1A 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR1A 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR1A 本身。

OCR1AH –TC1 输出比较寄存器 A 高字节

OCR1AH –TC1 输出比较寄存器 A 高字节			
地址: 0x89			默认值: 0x00
Bit	Name	R/W	描述
7:0	OCR1A[15:8]	R/W	输出比较寄存器 A 的高字节。 OCR1AL 和 OCR1AH 结合到一起组成 16 位的 OCR1A。读写 16 位寄存器需要两次操作。写 16 位 OCR1A 时，应先写入 OCR1AH。读 16 位 OCR1A 时，应先读取 OCR1AL。 OCR1A 不间断地与计数器数值 TCNT1 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC1A 引脚上产生波形。 当使用 PWM 模式时，OCR1A 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR1A 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR1A 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR1A 本身。

OCR1BL –TC1 输出比较寄存器 B 低字节

地址: 0x8A			默认值: 0x00
Bit	Name	R/W	描述
7:0	OCR1B[7:0]	R/W	输出比较寄存器 B 的低字节。 OCR1BL 和 OCR1BH 结合到一起组成 16 位的 OCR1B。读写 16 位寄存器需要两次操作。写 16 位 OCR1B 时，应先写入 OCR1BH。读 16 位 OCR1B 时，应先读取 OCR1BL。 OCR1B 不间断地与计数器数值 TCNT1 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC1B 引脚上产生波形。 当使用 PWM 模式时，OCR1B 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR1B 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR1B 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR1B 本身。

OCR1BH –TC1 输出比较寄存器 B 高字节

地址: 0x8B			默认值: 0x00
Bit	Name	R/W	描述
7:0	OCR1B[15:8]	R/W	输出比较寄存器 B 的高字节。 OCR1BL 和 OCR1BH 结合到一起组成 16 位的 OCR1B。读写 16 位寄存器需要两次操作。写 16 位 OCR1B 时，应先写入 OCR1BH。读 16 位 OCR1B 时，应先读取 OCR1BL。 OCR1B 不间断地与计数器数值 TCNT1 进行比较。比较匹配可以用来产生输出比较中断，或者用来在 OC1B 引脚上产生波形。 当使用 PWM 模式时，OCR1B 寄存器使用双缓冲寄存器。而普通工作模式和匹配清零模式下，双缓冲功能是禁止的。双缓冲可以将更新 OCR1B 寄存器与计数最大值或最小值时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。 使用双缓冲功能时，CPU 访问的是 OCR1B 缓冲寄存器，禁止双缓冲功能时 CPU 访问的是 OCR1B 本身。

TIMSK1 – TC1 中断屏蔽寄存器

地址: 0x6F			默认值: 0x00
Bit	Name	R/W	描述
7:6	-	-	保留。
5	ICIE1	R/W	输入捕捉中断使能控制位。 当 ICIE1 位为“1”时，且全局中断置位，TC1 输入捕捉中断被使能。当输入捕捉触发时，即 TIFR1 的 ICF1 标志被置位，中断发生。 当 ICIE1 位为“0”时，TC1 输入捕捉中断被禁止。
4:3	-	-	保留。
2	OCIE1B	R/W	输出比较 B 匹配中断使能位。 当 OCIE1B 位为“1”，且全局中断置位，TC1 输出比较 B 匹配中断使能。当比较匹配发生时，即 TIFR 中 OCF1B 位被置位时，中断产生。 当 OCIE1B 位为“0”时，TC1 输出比较 B 匹配中断被禁止。
1	OCIE1A	R/W	输出比较 A 匹配中断使能位。 当 OCIE1A 位为“1”，且全局中断置位，TC1 输出比较 A 匹配中断使能。当比较匹配发生时，即 TIFR 中 OCF1A 位被置位时，中断产生。 当 OCIE1A 位为“0”时，TC1 输出比较 A 匹配中断被禁止。
0	TOIE1	R/W	溢出中断使能位。 当 TOIE1 位为“1”，且全局中断置位，TC1 溢出中断使能。当 TC1 发生溢出，即 TIFR 中的 TOV1 位被置位时，中断产生。 当 TOIE1 位为“0”时，TC1 溢出中断被禁止。

TIFR1 – TC1 中断标志寄存器

地址: 0x16			默认值: 0x00
Bit	Name	R/W	描述
7:6	-	-	保留。
5	ICF1	R/W	输入捕捉标志位。 当输入捕捉事件发生时，ICF1 标志被置位。当 ICR1 被用作计数的 TOP 值，且计数值到达 TOP 值时，ICF1 标志被置位。若 ICIE1 为“1”且全局中断标志置位，则会产生输入捕捉中断。执行此中断服务程序时 ICF1 将自动清零，或对 ICF1 位写“1”也可清零该位。
4:3	-	-	保留。
2	OCF1B	R/W	输出比较 B 匹配标志位。 当 TCNT1 等于 OCR1B 时，比较单元就给出匹配信号，并置位比较标志 OCF1B。若此时输出比较中断使能 OCIE1B 为“1”且全局中断标志置位，则会产生输出比较中断。执行此中断服务程序时 OCF1B 将自动清零，或对 OCF1B 位写“1”也可清零该位。
1	OCF1A	R/W	输出比较 A 匹配标志位。 当 TCNT1 等于 OCR1A 时，比较单元就给出匹配信号，并置位比较标志 OCF1A。若此时输出比较中断使能 OCIE1A 为“1”且全局中断标志置位，则会产生输出比较中断。执行此中断服务程序时 OCF1A 将自动清零，或对 OCF1A 位写“1”也可清零该位。
0	TOV1	R/W	溢出标志位。 当计数器发生溢出时，置位溢出标志 TOV1。若此时溢出中断使能 TOIE1 为“1”且全局中断标志置位，则会产生溢出中断。执行此中断服务程序时 TOV1 将自动清零，或对 TOV1 位写“1”也可清零该位。

TC0/TC1 预分频器

主要特点

- 10 位预分频器
- TC0 和 TC1 复用
- 支持软件复位

综述

TC0 和 TC1 共用一个 10 位的预分频器，但它们有不同的分频设置。预分频器只对系统时钟进行分频，可以输出 5 个不同的分频信号，即 $f_{sys}/1$，$f_{sys}/8$，$f_{sys}/64$，$f_{sys}/256$ 和 $f_{sys}/1024$，通过 TC0 和 TC1 的时钟选择源来控制。

通过复位预分频器来同步定时计数器与程序运行是可能的，但是必须要注意另外一个定时计数器是否也在使用这一预分频器，复位将会影响计数。

寄存器描述

GTCCR –通用定时计数器控制寄存器

GTCCR –通用定时计数器控制寄存器			
地址: 0x23			默认值: 0x00
Bit	Name	R/W	描述
7	TSM	R/W	定时计数器同步模式使能位。 当设置 TSM 位为“1”时，激活定时计数器同步模式。在这种模式下，写入 PSR10 位的值将会保持，因此一直复位预分频器。这可以停止定时计数器来配置所有的计数器。
6:1	-	-	保留。
0	PSR10	R/W	预分频器复位。 置位 PSR10 时将会复位 TC0 和 TC1 的预分频器。当 TSM=0 时，复位操作完成以后由硬件自动清理这一位。当 TSM=1 时，PSR10 的值将保持。

外部中断

主要特点

- 3 个外部引脚中断源，可配置的触发方式
- 25 个独立的引脚电平改变中断源
- 中断源可用作唤醒

综述

外部中断包括 3 个外部引脚中断源 INTx 和 25 个外部引脚电平改变中断源 PCINTx。如果中断使能，即使中断源的引脚配置成输出也能触发中断，即支持软件产生中断方式。

外部引脚中断有 3 个中断源，由寄存器位来配置触发方式，支持上升沿，下降沿或低电平来触发。当外部中断使能并采用低电平触发时，外部引脚持续保持低电平，中断也会一直持续。

外部引脚电平改变中断由四组中断源 PCINTA，PCINTB，PCINTC 和 PCINTD 组成，并分别由 PCIE0，PCIE1， PCIE2 和 PCIE3 寄存器位来使能和禁止，每一个中断源又有独立的寄存器位来使能和禁止。

外部中断采用异步检测，可以用作系统唤醒。

寄存器描述

EICRA –外部引脚中断控制寄存器

地址: 0x69			默认值: 0x00
Bit	Name	R/W	描述
7:6	-	-	保留。
5:4	ISC2[1:0]	R/W	外部引脚 2 中断触发方式控制位。 详见外部中断触发方式表格描述。
3:2	ISC1[1:0]	R/W	外部引脚 1 中断触发方式控制位。 详见外部中断触发方式表格描述。
1:0	ISC0[1:0]	R/W	外部引脚 0 中断触发方式控制位。 详见外部中断触发方式表格描述。

外部中断触发方式见下表。

外部中断触发方式控制

ISCn[1:0]	描述
0	外部引脚 INTn 低电平触发
1	外部引脚 INTn 上升沿或下降沿触发
2	外部引脚 INTn 下降沿触发
3	外部引脚 INTn 上升沿触发

说明：

ISCn 和 INTn 中的“n”表示 0，1 或 2。

EIMSK –外部引脚中断屏蔽寄存器

地址: 0x1D			默认值: 0x00
Bit	Name	R/W	描述
7:3	-	-	保留。
2	INT2EN	R/W	外部引脚 2 中断使能控制位。 当设置 INT2EN 位为“1”时，且全局中断置位，外部引脚 2 中断被使能，唤醒功能被使能。 当设置 INT2EN 位为“0”时，外部引脚 2 中断被禁止，唤醒功能也被禁止。
1	INT1EN	R/W	外部引脚 1 中断使能控制位。 当设置 INT1EN 位为“1”时，且全局中断置位，外部引脚 1 中断被使能，唤醒功能被使能。 当设置 INT1EN 位为“0”时，外部引脚 1 中断被禁止，唤醒功能也被禁止。
0	INT0EN	R/W	外部引脚 0 中断使能控制位。 当设置 INT0EN 位为“1”时，且全局中断置位，外部引脚 0 中断被使能，唤醒功能被使能。 当设置 INT0EN 位为“0”时，外部引脚 0 中断被禁止，唤醒功能也被禁止。

EIFR –外部引脚中断标志寄存器

地址: 0x1C			默认值: 0x00
Bit	Name	R/W	描述
7:3	-	-	保留。
2	INTF2	R/W	外部引脚 2 中断标志位。 当边沿触发外部引脚 2 中断时，INTF2 被置位。当低电平触发外部引脚 2 中断时，不会置位 INTF2 位。若此时外部引脚 2 中断使能 INT2EN 位为“1”且全局中断标志置位，则会产生外部引脚 2 中断。执行此中断服务程序时 INTF2 将自动清零，或对 INTF2 位写“1”也可清零该位。
1	INTF1	R/W	外部引脚 1 中断标志位。 当边沿触发外部引脚 1 中断时，INTF1 被置位。当低电平触发外部引脚 1 中断时，不会置位 INTF1 位。若此时外部引脚 1 中断使能 INT1EN 位为“1”且全局中断标志置位，则会产生外部引脚 1 中断。执行此中断服务程序时 INTF1 将自动清零，或对 INTF1 位写“1”也可清零该位。
0	INTF0	R/W	外部引脚 0 中断标志位。 当边沿触发外部引脚 0 中断时，INTF0 被置位。当低电平触发外部引脚 0 中断时，不会置位 INTF0 位。若此时外部引脚 0 中断使能 INT0EN 位为“1”且全局中断标志置位，则会产生外部引脚 0 中断。执行此中断服务程序时 INTF0 将自动清零，或对 INTF0 位写“1”也可清零该位。

PCICR –外部引脚电平改变中断控制寄存器

地址: 0x68			默认值: 0x00
Bit	Name	R/W	描述
7:4	-	-	保留。
3	PCIE3	R/W	外部引脚 3（PDx）电平改变中断使能控制位。 当设置 PCIE3 位为“1”时，且全局中断置位，外部引脚 3 电平改变中断使能。当 PDx 引脚的电平改变时将会触发中断。 当设置 PCIE3 位为“0”时，外部引脚 3 电平改变中断禁止。
2	PCIE2	R/W	外部引脚 2（PCx）电平改变中断使能控制位。 当设置 PCIE2 位为“1”时，且全局中断置位，外部引脚 2 电平改变中断使能。当 PCx 引脚的电平改变时将会触发中断。 当设置 PCIE2 位为“0”时，外部引脚 2 电平改变中断禁止。
1	PCIE1	R/W	外部引脚 1（PBx）电平改变中断使能控制位。 当设置 PCIE1 位为“1”时，且全局中断置位，外部引脚 1 电平改变中断使能。当 PBx 引脚的电平改变时将会触发中断。 当设置 PCIE1 位为“0”时，外部引脚 1 电平改变中断禁止。
0	PCIE0	R/W	外部引脚 0（PAx）电平改变中断使能控制位。 当设置 PCIE0 位为“1”时，且全局中断置位，外部引脚 0 电平改变中断使能。当 PAx 引脚的电平改变时将会触发中断。 当设置 PCIE0 位为“0”时，外部引脚 0 电平改变中断禁止。

PCIFR –外部引脚电平改变中断标志寄存器

地址: 0x1B			默认值: 0x00
Bit	Name	R/W	描述
7:4	-	-	保留。
3	PCIF3	R/W	外部引脚 3 电平改变中断标志位。 当外部引脚 PDx 电平改变触发外部引脚 3 电平改变中断时，PCIF3 被置位。若此时外部引脚 3 中断使能 PCIE3 位为“1”且全局中断标志置位，则会产生外部引脚 3 电平改变中断。执行此中断服务程序时 PCIF3 将自动清零，或对 PCIF3 位写“1”也可清零该位。
2	PCIF2	R/W	外部引脚 2 电平改变中断标志位。 当外部引脚 PCx 电平改变触发外部引脚 2 电平改变中断时，PCIF2 被置位。若此时外部引脚 2 中断使能 PCIE2 位为“1”且全局中断标志置位，则会产生外部引脚 2 电平改变中断。执行此中断服务程序时 PCIF2 将自动清零，或对 PCIF2 位写“1”也可清零该位。
1	PCIF1	R/W	外部引脚 1 电平改变中断标志位。 当外部引脚 PBx 电平改变触发外部引脚 1 电平改变中断时，PCIF1 被置位。若此时外部引脚 1 中断使能 PCIE3 位为“1”且全局中断标志置位，则会产生外部引脚 1 电平改变中断。执行此中断服务程序时 PCIF1 将自动清零，或对 PCIF1 位写“1”也可清零该位。
0	PCIF0	R/W	外部引脚 0 电平改变中断标志位。 当外部引脚 PAx 电平改变触发外部引脚 0 电平改变中断时，PCIF0 被置位。若此时外部引脚 0 中断使能 PCIE0 位为“1”且全局中断标志置位，则会产生外部引脚 0 电平改变中断。执行此中断服务程序时 PCIF0 将自动清零，或对 PCIF0 位写“1”也可清零该位。

PCMSK0 –外部引脚 0 电平改变中断屏蔽控制寄存器

地址: 0x6B			默认值: 0x00
Bit	Name	R/W	描述
7:0	PCINT0EN[7:0]	R/W	外部引脚 0 电平改变中断使能控制位。 外部引脚 PAx 的电平改变中断都有独立的使能控制位。当设置 PCINT0EN[x]位为“1”时，且 PCIE0 位和全局中断置位，外部引脚 PAx 的电平改变中断被使能。 当设置 PCINT0EN[x]位“0”时，外部引脚 PAx 的电平改变中断被禁止。

PCMSK1 –外部引脚 1 电平改变中断屏蔽控制寄存器

地址: 0x6C			默认值: 0x00
Bit	Name	R/W	描述
7:0	PCINT1EN[7:0]	R/W	外部引脚 1 电平改变中断使能控制位。 外部引脚 PBx 的电平改变中断都有独立的使能控制位。当设置 PCINT1EN[x]位为“1”时，且 PCIE1 位和全局中断置位，外部引脚 PBx 的电平改变中断被使能。 当设置 PCINT1EN[x]位“0”时，外部引脚 PBx 的电平改变中断被禁止。

PCMSK2 –外部引脚 2 电平改变中断屏蔽控制寄存器

地址: 0x6D			默认值: 0x00
Bit	Name	R/W	描述
7:0	PCINT2EN[7:0]	R/W	外部引脚 2 电平改变中断使能控制位。 外部引脚 PCx 的电平改变中断都有独立的使能控制位。当设置 PCINT2EN[x]位为“1”时，且 PCIE2 位和全局中断置位，外部引脚 PCx 的电平改变中断被使能。 当设置 PCINT2EN[x]位“0”时，外部引脚 PCx 的电平改变中断被禁止。

PCMSK3 –外部引脚 3 电平改变中断屏蔽控制寄存器

地址: 0x73			默认值: 0x00
Bit	Name	R/W	描述
7:0	PCINT3EN[7:0]	R/W	外部引脚 3 电平改变中断使能控制位。 外部引脚 PDx 的电平改变中断都有独立的使能控制位。当设置 PCINT3EN[x]位为“1”时，且 PCIE3 位和全局中断置位，外部引脚 PDx 的电平改变中断被使能。 当设置 PCINT3EN[x]位“0”时，外部引脚 PDx 的电平改变中断被禁止。

模拟比较器

主要特点

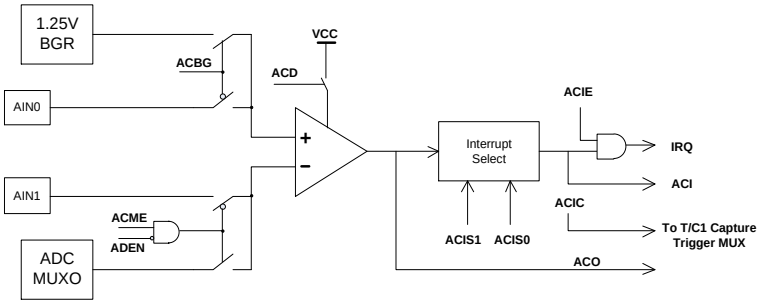
- 6mV 比较精度
- 支持 2 路外部独立模拟输入
- ~~支持 ADC 通道输入~~
- 支持内部 1.25V 参考电压输入

综述

模拟比较器对正极的值与负极的值进行比较，当正极上的电压比负极上的电压高时，模拟比较器的输出 ACO 被置位。当 ACO 的电平发生变化时，信号的边沿可用来触发中断。输出信号 ACO 还可用来触发定时计数器 1 的输入捕捉功能。

~~请注意，在使用 ADC 通道输入前，必须将 I/O 外设功耗管理寄存器中的 PRADC 位清零，具体设置请参考 PRR 寄存器说明。~~

模拟比较器构架图：



寄存器描述

ACSR –模拟比较器的控制和状态寄存器

地址: 0x30			默认值: 0x80	
Bit	Name	R/W	描述	
7	ACD	R/W	模拟比较器禁止位。 当设置 ACD 位为“1”时，模拟比较器被关闭。 当设置 ACD 位为“0”时，模拟比较器被开启。	
6	ACBG	R/W	模拟比较器正极选择控制位。 当设置 ACBG 位为“1”时，正极选择内部 1.25V 参考电压作为输入。 当设置 ACBG 位为“0”时，正极选择外部引脚 AIN0 作为输入。	
5	ACO	R/W	模拟比较器的输出状态位。 模拟比较器的输出经过同步之后直接连到 ACO 位。软件可读取 ACO 位的值来获取模拟比较器的输出值。	
4	ACIF	R/W	模拟比较器的中断标志位。 当模拟比较器的输出事件触发了由 ACIS 位定义的中断模式时，ACIF 位被置位。 当中断使能位 ACIE 为“1”且全局中断置位时，中断产生。执行模拟比较器中断服务程序时，ACIF 将自动清零，或对 ACIF 位写“1”也可清零该位。	
3	ACIE	R/W	模拟比较器的中断使能位。 当 ACIE 位为“1”，且全局中断置位时，模拟比较器的中断被使能。 当 ACIE 位为“0”时，模拟比较器的中断被禁止。	
2	ACIC	R/W	模拟比较器输入捕捉使能位。 当 ACIC 位为“1”时，定时计数器 1 的输入捕捉源来自模拟比较器的输出 ACO。 当 ACIC 位为“0”时，定时计数器 1 的输入捕捉源来自外部引脚 ICP1。	
1:0	ACIS[1:0]	R/W	模拟比较器中断模式控制位。 ACIS 控制模拟比较器的中断触发方式。	
			ACIS[1:0]	中断模式
			0	ACO 的上升沿或下降沿触发
			1	保留。
			2	ACO 的下降沿触发
			3	ACO 的上升沿触发

DIDR1 –数字输入禁止控制寄存器 1

地址: 0x7F			默认值: 0x00
Bit	Name	R/W	描述
7:2	-	-	保留。
1:0	AIND[1:0]	R/W	数字输入禁止控制位。 当设置 AINDx 位为“1”时，引脚 AINx 的数字输入端被禁止，并一直为零。当使能模拟比较器时，AINx 的数字输入端功能不需要，因此须置位 AINDx。 当设置 AINDx 位为“0”时，引脚 AINx 的数字输入端被使能，引脚上的信号可输入到内部数字逻辑，此时须置位 ACD 位，及关闭模拟比较器。

模数转换器(ADC)

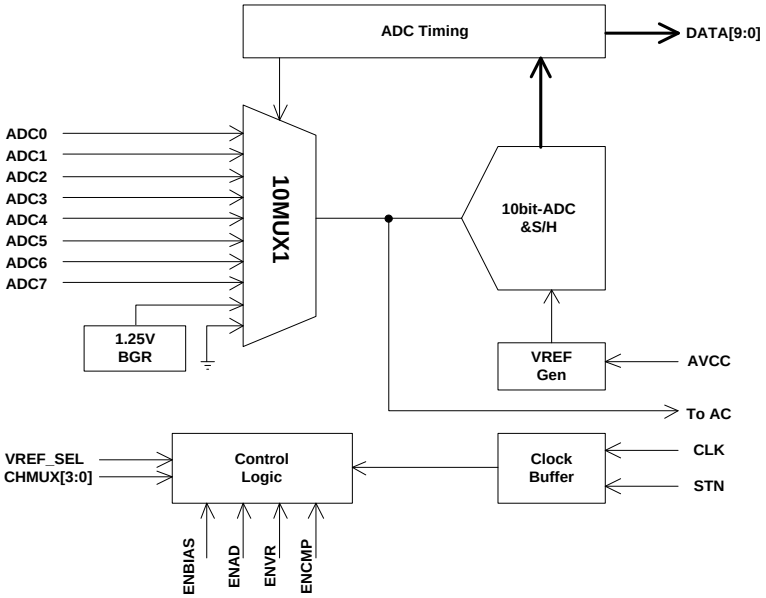
主要特点

- 10 位分辨率，DNL 为±1LSB，INL 为±1.5LSB
- 最高分辨率时采样率高达 250KSPS
- 8 路复用的单端输入通道
- 连续转换或单次转换模式
- 可选内部 1.25V 参考电压或外部 VCC 为参考电压
- 基于中断源的自动开始转换触发模式
- 转换结果支持可选左对齐模式
- ADC 转换结束中断

综述

LGT8F08X 系列均包含一个 10 位的逐次逼近型 ADC，将输入的模拟电压转换成一个 10 位的数字量。最小值代表 GND，最大值代表基准电压减去 1LSB。

ADC 结构图



通过设置 ADCSRA 寄存器的 ADEN 位即可开启 ADC，ADEN 位清零时 ADC 进入休眠状态。

ADC 与一个 8 通道的模拟多路复用器连接，能对端口 A 的 8 路单端输入电压进行采样，还可选择 GND 与内部基准电压源作为输入源，输入通道的选择通过寄存器位来控制。在启动转换之前先选定通道。在 ADSC 置位后的一个 ADC 时钟周期后就可以选择新的模拟通道。如果在转换过程中选择了另外一个通道，那么 ADC 会完成当前的转换之后再改变通道。最简单的办法是等待转换结束后再改变通道。

ADC 转换结果为 10 位数据，存放在数据寄存器 ADCH 及 ADCL 中，通过设置 ADLAR 位可以选择数据对齐方式。当只需要 8 位的转换精度时，可以置位 ADLAR 来设置成数据左对齐，只需要读取 ADCH 寄存器就足够了。当清零 ADLAR 位时数据为右对齐存放，此时需要先读取 ADCL 寄存器，再读取 ADCH 寄存器，以保证数据寄存器中的内容为同一次转换的结果。一旦读取 ADCL 后，数据寄存器 ADCL 和 ADCH 被锁存，读取 ADCH 后转换结果即可再更新到数据寄存器 ADCL 和 ADCH。

当设置 ADC 启动转换位 ADSC 为“1”时，即可启动单次转换。在转换过程中此位保持为高，直到转换结束，然后被硬件清零。

ADC 的参考电压源反映了 ADC 的转换范围。若输入通道电压超过了参考电压，其转换结果将接近最大值。参考电压源可以是 AVCC 或内部 1.25V 基准电压源，通过 REFS 寄存器位选择。

寄存器描述

ADMUX – ADC 多路选择控制寄存器

地址: 0x7C			默认值: 0x00	
Bit	Name	R/W	描述	
7:6	REFS[1:0]	R/W	参考电压选择控制位。 通过设置 REFS 控制位来选择参考电压，若在转换过程中改变 REFS 的设置，只有等到当前的转换结束之后改变才会起作用。	
			REFS[1:0]	参考电压选择
			0	AVCC
			1	AVCC
			2	1.25V 片内基准电压源
			3	保留。
5	ADLAR	R/W	转换结果左对齐使能控制位。 当 ADLAR 位为“1”时，转换结果在 ADC 数据寄存器中为左对齐。 当 ADLAR 位为“0”时，转换结果在 ADC 数据寄存器中为右对齐。	
4:0	CHMUX[4:0]	R/W	输入通道源选择控制位。	
			CHMUX[4:0]	单端输入源
			0	PA0
			1	PA1
			2	PA2
			3	PA3
			4	PA4
			5	PA5
			6	PA6
			7	PA7
			8-24	保留。
			25	1.25V 片内基准电压源
			26	GND
			27-31	保留。

ADCSRA –ADC 控制与状态寄存器 A

地址: 0x7A			默认值: 0x00
Bit	Name	R/W	描述
7	ADEN	R/W	ADC 使能控制位。 当设置 ADEN 位为“1”时，ADC 被使能。 当设置 ADEN 位为“0”时，ADC 被禁止。

ADCSRA –ADC 控制与状态寄存器 A (接上页)			
地址: 0x7A			默认值: 0x00
Bit	Name	R/W	描述
6	ADSC	R/W	ADC 开始转换。 在单次转换模式下，ADSC 置位将启动一次转换。在连续转换模式下，ADSC 置位将启动首次转换。
5	ADATE	R/W	ADC 自动触发使能控制位。 当 ADATE 位为“1”时，自动触发功能被使能。所选中触发信号的上升沿开启一次转换。触发源的选择由 ADCSRB 寄存器的 ADTS 来控制。 当设置 ADATE 位为“0”时，自动触发功能被禁止。
4	ADIF	R/W	ADC 中断标志位。 当 ADC 完成一次转换并更新数据寄存器后置位 ADIF。若 ADC 中断使能位 ADIE 为“1”且全局中断置位，ADC 中断产生。执行 ADC 中断会清零 ADIF 位，也可对该位写“1”来清零。
3	ADIE	R/W	ADC 中断使能控制位。 当设置 ADIE 位为“1”且全局中断置位时，ADC 中断被使能。 当设置 ADIE 位为“0”时，ADC 中断被禁止。
2:0	ADPS[2:0]	R/W	ADC 预分频器选择控制位。 ADPS 选择系统时钟产生 ADC 时钟的预分频因子。

ADPS[2:0]	预分频因子
0	2
1	2
2	4
3	8
4	16
5	32
6	64
7	128

ADCSRB – ADC 控制与状态寄存器 B

地址: 0x7B			默认值: 0x00
Bit	Name	R/W	描述
7	-	-	保留。
6	ACME	R/W	模拟比较器多路复用器使能控制位。 当设置 ACME 位为“1”且 ADC 处于关闭状态（ADEN=0）时，ADC 多路复用器的输出作为模拟比较器的负极输入。 当设置 ACME 位为“0”时，引脚 AIN1 作为模拟比较器的负极输入。
5:3	-	-	保留。
2:0	ADTS[2:0]	R/W	(转下页)

Bit	Name	R/W	描述
2:0	ADTS[2:0]	R/W	ADC 自动触发源选择控制位。 当设置 ADATE 位为“1”时，自动触发功能被使能，触发源的选择由 ADTS 来控制。当设置 ADATE 位为“0”时，ADTS 的设置无效。所选中触发信号中断标志的上升沿开启一次转换。当从一个中断标志清零的触发源切换到中断标志置位的触发源会使触发信号产生一个上升沿，如果此时 ADEN 置位，ADC 也会开启一次转换。当切换到连续转换模式（ADTS=0）时，自动触发功能被禁止。

ADTS[2:0]	触发源
0	连续转换模式
1	模拟比较器
2	外部中断 0
3	定时计数器 0 比较匹配
4	定时计数器 0 溢出
5	定时计数器 1 比较匹配 B
6	定时计数器 1 溢出
7	定时计数器 1 输入捕捉事件

ADCL – ADC 数据低字节寄存器

地址: 0x78			默认值: 0x00
Bit	Name	R/W	描述
7:0	ADCDOL	RO	ADC 数据低字节寄存器。 当 ADLAR 位为“0”时，ADC 输出数据 ADCDO 在寄存器中的存放按低位对齐，此时 ADCDOL = ADCDO[7:0]；当 ADLAR 位为“1”时，ADC 输出数据 ADCDO 在寄存器中的存放按高位对齐，此时 ADCDOL[7:6] = ADCDO[1:0]，低 6 位 ADCDOL[5:0]无意义。

ADCH– ADC 数据高字节寄存器

地址: 0x79			默认值: 0x00
Bit	Name	R/W	描述
7:0	ADCDOH	RO	ADC 数据高字节寄存器。 当 ADLAR 位为“0”时，ADC 输出数据 ADCDO 在寄存器中的存放按低位对齐，此时 ADCDOH[1:0] = ADCDO[9:8]，高 6 位 ADCDOL[15:10]无意义；当 ADLAR 位为“1”时，ADC 输出数据 ADCDO 在寄存器中的存放按高位对齐，此时 ADCDOH = ADCDO[9:2]。

DIDR0 –数字输入禁止控制寄存器 0

地址: 0x7E			默认值: 0x00
Bit	Name	R/W	描述
7:0	ADCD[7:0]	R/W	数字输入禁止控制位。 当 ADCDx 位为“1”，引脚 ADCx 的数字输入端被禁止，并一直为零。当使能模拟比较器时，ADCx 的数字输入端功能不需要，因此须置位 ADCDx。 当 ADCDx 位为“0”，引脚 ADCx 的数字输入端被使能，引脚上的信号可输入到内部数字逻辑，此时须清 ADEN 位，关闭模拟比较器。

看门狗定时器(WDT)

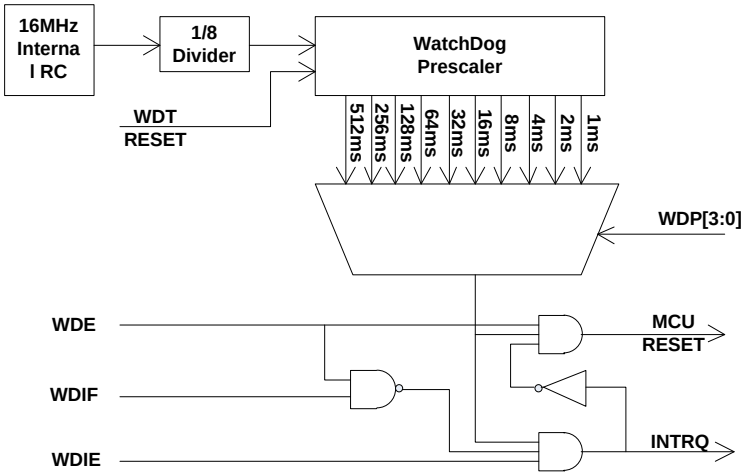
主要特点

- 时钟来自内部 16MHz 可校准 RC 振荡器
- 支持中断模式，复位模式以及复位中断模式
- 定时器超时可从 1ms 到 512ms

综述

看门狗定时器是一个 20 位的计数器，由 2M 的片内时钟驱动，支持 1ms 到 512ms 的定时溢出周期。

看门狗定时器内部结构图



看门狗定时器有 3 种操作模式，中断模式，系统复位模式和中断及系统复位模式。在中断模式下，定时器溢出时 WDT 会给出溢出中断，可用作休眠模式下的唤醒，或者作为一个通用的定时器。其中一个应用是限制某个操作的最大允许时间，当该操作运行时间超过预期时进行中断。在系统复位模式下，定时器溢出时 WDT 给出复位信号来复位系统。典型的应用是防止系统因代码跑飞而挂起。中断及系统复位模式结合了前面两种模式，当定时器溢出时先给出中断，再次溢出时则复位系统。这个模式可用作系统安全关机，它允许在系统复位之前先保存好关键的数据。

寄存器描述

WDTCSR –看门狗定时器控制和状态寄存器

地址: 0x60			默认值: 0x0
Bit	Name	R/W	描述
7	WDIF	R/W	看门狗中断标志位。 当 WDT 工作在中断模式并发生溢出时会置位 WDIF 位。当 WDT 中断使能位 WDIE 为“1”且全局中断置位时，WDT 中断产生。执行 WDT 中断时会清零 WDIF 位，对 WDIF 位写“1”也可清零该位。 （转下页）

<table>
<tr><th colspan="6">WDTCSR –看门狗定时器控制和状态寄存器 (接上页)</th></tr>
<tr><td rowspan="6">6</td><td rowspan="6">WDIE</td><td rowspan="6">R/W</td><td colspan="3">看门狗中断使能控制位。
当设置 WDIE 位为“1”，且全局中断置位时，WDT 中断被使能。
当设置 WDIE 位为“0”时，WDT 中断被禁止。
WDIE 位和 WDE 位决定看门狗的工作模式，如下表所示。</td></tr>
<tr><td>WDE</td><td>WDIE</td><td>模式</td><td>溢出后动作</td></tr>
<tr><td>0</td><td>0</td><td>停止</td><td>无</td></tr>
<tr><td>0</td><td>1</td><td>中断模式</td><td>中断</td></tr>
<tr><td>1</td><td>0</td><td>复位模式</td><td>复位</td></tr>
<tr><td>1</td><td>1</td><td>中断及复位模式</td><td>中断后复位</td></tr>
<tr><td>5</td><td>WDP[3]</td><td>R/W</td><td colspan="3">WDT 预分频因子选择控制第 3 位。
WDP[3]和 WDP[2:0]组成 WDT 预分频因子选择位 WDP[3:0]，用来设置 WDT 的溢出周期。详见 WDP[2:0]描述。</td></tr>
<tr><td>4</td><td>WDCE</td><td>R/W</td><td colspan="3">WDT 改变配置使能控制位。
当设置 WDCE 位为“1”时，WDE 和 WDP 位的配置允许改变，CPU 须在 4 个时钟周期内清零 WDE 位或写入新的 WDP 值，否则硬件会清零 WDCE 位。
当设置 WDCE 位为“0”时，WDE 和 WDP 位的配置不允许改变，即清零 WDE 位和更改 WDP 值的操作不会被执行。</td></tr>
<tr><td>3</td><td>WDE</td><td>R/W</td><td colspan="3">WDT 复位系统使能控制位。
当设置 WDE 位为“1”时，WDT 溢出时复位系统被使能。当设置 WDE 位为“0”时，WDT 溢出时不会复位系统。
置位 WDE 时，CPU 可直接往 WDE 位写“1”，清零 WDE 时，CPU 须先置位 WDCE，接着在 4 个时钟周期内清零 WDE 位。
当 WDE 位为“1”且 WDT 溢出复位系统后会置位 WDT 位于 MCUSR 寄存器的复位系统标志 WDRF。当 WDRF 位处于置位状态时会置位 WDE 位。因此要清零 WDE 位，必须先清零 WDRF 位。</td></tr>
<tr><td rowspan="12">2:0</td><td rowspan="12">WDP[2:0]</td><td rowspan="12">R/W</td><td colspan="3">WDT 预分频因子选择控制低 3 位。
WDP[3]和 WDP[2:0]组成 WDT 预分频因子选择位 WDP[3:0]，用来设置 WDT 的溢出周期。</td></tr>
<tr><td>WDP[3:0]</td><td>计数时钟数</td><td>溢出周期</td></tr>
<tr><td>0</td><td>2K Cycle</td><td>1ms</td></tr>
<tr><td>1</td><td>4K Cycle</td><td>2ms</td></tr>
<tr><td>2</td><td>8K Cycle</td><td>4ms</td></tr>
<tr><td>3</td><td>16K Cycle</td><td>8ms</td></tr>
<tr><td>4</td><td>32K Cycle</td><td>16ms</td></tr>
<tr><td>5</td><td>64K Cycle</td><td>32ms</td></tr>
<tr><td>6</td><td>128K Cycle</td><td>64ms</td></tr>
<tr><td>7</td><td>256K Cycle</td><td>128ms</td></tr>
<tr><td>8</td><td>512K Cycle</td><td>256ms</td></tr>
<tr><td>9</td><td>1024K Cycle</td><td>512ms</td></tr>
<tr><td>10-15</td><td>64K Cycle</td><td>32ms</td></tr>
</table>

MCUSR –MCU 状态寄存器

地址: 0x54			默认值: 0x00
Bit	Name	R/W	描述
7: 6	-	-	保留。
5	SWRF	R/W	软件复位标志位。 发生软件复位时置位 SWRF。上电复位将其清零，也可以通过写“0”来清除。
4	OCDRF	R/W	调试复位标志位。 发生调试复位时置位 OCDRF。上电复位将其清零，也可以通过写“0”来清除。
3	WDRF	R/W	看门狗复位标志位。 发生看门狗复位时置位 WDRF。上电复位将其清零，也可以通过写“0”来清除。
2	-	-	保留。
1	EXRF	R/W	外部复位标志位。 发生外部复位时置位 EXRF。上电复位将其清零，也可以通过写“0”来清除。
0	PORF	R/W	上电复位标志位。 发生上电复位时置位 PORF。只能通过写“0”来清除。

实时计数器 (RTC)

主要特点

- 24 位定时计数器
- 内部 128Hz 低频 RC 时钟源
- 计数溢出时自动加载并产生中断
- 支持 Power Down 模式下计数溢出唤醒

综述

RTC 模块是一个 24 位的定时计数器，工作在内部低频时钟(经分频后约为 128Hz 左右)，最大支持 36 小时的计数。CPU 设定计数器的计数最大值(TOP)，并使能计数，每一个计数时钟计数器自动减一，当计数到最小值 0x0(BOTTOM)时，计数器自动加载 TOP 值，同时置位溢出标志，若溢出中断使能位置位，则产生溢出中断。

RTC 模块可工作在 Power Down 模式下，并可用计数溢出来唤醒系统。同时也支持外部引脚 PC6 唤醒，持续超过 500us 的高电平即可。在配置 RTC 模块寄存器及使能 RTC 计数器之前，必须先使能 RTC 模块的工作时钟，即 CPU 先置位 PMCR 寄存器的 RCKEN 位来使能内部 RCK 时钟产生模块，并等待 1ms 的时间直到 RCK 时钟稳定。

寄存器描述

RTCSR – RTC 控制和状态寄存器

地址: 0xD0			默认值: 0x80
Bit	Name	R/W	描述
7	WREN	R/W	写使能标志位。 当 WREN 位为“1”时，允许 CPU 写 RTCSR 寄存器。 当 WREN 位为“0”时，写 RTCSR 寄存器操作失效。 CPU 配置 RTCSR 寄存器之前最好先读取 WREN 位的状态。
6:5	-	-	保留。
4	LOAD	R/W	加载使能控制位。 当设置 LOAD 位为“1”时，RTC 计数器加载计数最大值寄存器 RTCTOPH，RTCTOPM 和 RTCTOPL 的数值。加载完毕后，硬件自动清理 LOAD 位。
3	PWEN	R/W	外部引脚唤醒使能控制位。 当设置 PWEN 位为“1”时，外部引脚唤醒功能被使能。系统工作在 Power Off 模式下，可通过外部引脚和 RTC 计数溢出来唤醒。使用外部引脚来唤醒时，需要保持 PC6 为高电平并持续超过 500us。 当设置 PWEN 位为“0”时，外部引脚唤醒功能被禁止。
2	CWEN	R/W	计数溢出唤醒使能控制位。 当设置 CWEN 位为“1”时，计数溢出唤醒功能被使能。系统工作在 Power Off 模式下，可通过外部引脚和 RTC 计数溢出来唤醒。使用计数溢出来唤醒时，CPU 须预设计数最大值并使能计数，当计数到最小值发生溢出即可唤醒系统。 当设置 CWEN 位为“0”时，计数溢出唤醒功能被禁止。
1	CEN	R/W	计数使能控制位。 当设置 CEN 位为“1”时，RTC 开始计数。 当设置 CEN 位为“0”时，RTC 停止计数。
0	POWOFF	R/W	Power Off 使能控制位。 当 CPU 写 POWOFF 位为“1”时，系统进入 Power Off 模式，只有 RTC 模块工作，系统功耗最低。Power Off 模式下，可通过外部引脚和 RTC 计数溢出来唤醒。系统唤醒后 CPU 须清零 POWOFF 位。

RTCISR – RTC 中断控制和状态寄存器

地址: 0xD1			默认值: 0x00
Bit	Name	R/W	描述
7	WKUF	R/W	唤醒标志位。 当系统从 Power Off 模式唤醒时，WKUF 位被置位。
6:2	-	-	Reserved.
1	IF	R/W	RTC 溢出中断标志位。 当计数器发生计数溢出时，置位 IF 位。若 RTC 溢出中断使能 IEN 位为“1”且全局中断置位，产生 RTC 溢出中断。执行 RTC 溢出中断服务程序清零该位，也可对该位写“1”来清零。
0	IEN	R/W	RTC 溢出中断使能控制位。 当设置 IEN 位为“1”且全局中断置位时，RTC 溢出中断被使能。 当设置 IEN 位为“0”时，RTC 溢出中断被禁止。

RTCTOPL – RTC 计数最大值低位寄存器

地址: 0xD2			默认值: 0x00
Bit	Name	R/W	描述
7:0	RTCTOP[7:0]	R/W	RTC 计数最大值低位寄存器。 RTC 计数最大值由 RTCTOPH，RTCTOPM 和 RTCTOPL 组成，TOP = {RTCTOPH, RTCTOPM, RTCTOPL}，当 RTCSR 寄存器中的 LOAD 位被置位时，TOP 值被加载到计数器。

RTCTOPM – RTC 计数最大值中位寄存器

地址: 0xD3			默认值: 0x00
Bit	Name	R/W	描述
7:0	RTCTOP[15:8]	R/W	RTC 计数最大值中位寄存器。 RTC 计数最大值由 RTCTOPH，RTCTOPM 和 RTCTOPL 组成，TOP = {RTCTOPH, RTCTOPM, RTCTOPL}，当 RTCSR 寄存器中的 LOAD 位被置位时，TOP 值被加载到计数器。

RTCTOPH – RTC 计数最大值高位寄存器

地址: 0xD4			默认值: 0x00
Bit	Name	R/W	描述
7:0	RTCTOP[23:16]	R/W	RTC 计数最大值高位寄存器。 RTC 计数最大值由 RTCTOPH，RTCTOPM 和 RTCTOPL 组成，RTCTOP = {RTCTOPH, RTCTOPM, RTCTOPL}，当 RTCSR 寄存器中的 LOAD 位被置位时，TOP 值被加载到计数器。

RTCNTL – RTC 计数器最低字节寄存器

地址: 0xD5			默认值: 0x00
Bit	Name	R/W	描述
7:0	RTCNT[7:0]	R/O	RTC 计数值低位寄存器。 RTC 计数值由 RTCNTH，RTCNTM 和 RTCNTL 组成，RTCNT = {RTCNTH, RTCNTM, RTCNTL}。

RTCNTM – RTC 计数器中间字节寄存器

地址: 0xD6			默认值: 0x00
Bit	Name	R/W	描述
7:0	RTCNT[15:8]	R/O	RTC 计数器中间字节寄存器。 RTC 计数值由 RTCNTH，RTCNTM 和 RTCNTL 组成，RTCNT = {RTCNTH, RTCNTM, RTCNTL}。

RTCNTH – RTC 计数器最高字节寄存器

地址: 0xD7			默认值: 0x00
Bit	Name	R/W	描述
7:0	RTCNT[23:16]	R/O	RTC 计数器高字节寄存器。 RTC 计数值由 RTCNTH，RTCNTM 和 RTCNTL 组成，RTCNT = {RTCNTH, RTCNTM, RTCNTL}。

寄存器速查表

地址	寄存器	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit	Bit0
$FF~$F7	**RESERVED**	-	-	-	-	-	-	-	-
$F6	**GUID3**	GUID 第 3 字节							
$F5	**GUID2**	GUID 第 2 字节							
$F4	**GUID1**	GUID 第 1 字节							
$F3	**GUID0**	GUID 第 0 字节							
$F2	**PMCR**	-	LFEN	EXTMSE	CFDS	-	OSCMEN	RC1KEN	RC16MEN
				.....					
$D7	**RTCNTH**	RTC 计数器最高字节							
$D6	**RTCNTM**	RTC 计数器中间字节							
$D5	**RTCNTL**	RTC 计数器最低字节							
$D4	**RTCTOPH**	RTC 定时寄存器最高字节							
$D3	**RTCTOPM**	RTC 定时寄存器中间字节							
$D2	**RTCTOPL**	RTC 定时寄存器最低字节							
$D1	**RTCISR**	WKUF	-	-	-	-	-	IF	IEN
$D0	**RTCSR**	WREN	-	-	LOAD	PWEN	CWEN	EN	POWOFF
				.....					
$C6	**UDR**	USART IO 数据寄存器							
$C5	**UBRRH**	-	-	-	-	USART 波特率高 4 位			
$C4	**UBRRL**	USART 波特率低 8 位							
$C3	**RESERVED**					-			
$C2	**UCSRC**	UMSEL1	UMSEL0	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
$C1	**UCSRB**	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
$C0	**UCSRA**	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
				.....					
$8B	**OCR1BH**	定时计数器 1 输出比较寄存器 B 高字节							
$8A	**OCR1BL**	定时计数器 1 输出比较寄存器 B 低字节							
$89	**OCR1AH**	定时计数器 1 输出比较寄存器 A 高字节							
$88	**OCR1AL**	定时计数器 1 输出比较寄存器 A 低字节							
$87	**ICR1H**	定时计数器 1 俘获寄存器高字节							
$86	**ICR1L**	定时计数器 1 俘获寄存器低字节							
$85	**TCNT1H**	定时计数器 1 计数器高字节							
$84	**TCNT1L**	定时计数器 1 计数器低字节							
$83	**RESERVED**	-	-	-	-	-	-	-	-
$82	**TCCR1C**	FOC1A	FOC1B	-	-	-	-	-	-
$81	**TCCR1B**	ICNC1	ICES1	-	WGM13	MGM12	CS12	CS11	CS10
$80	**TCCR1A**	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10
$7F	**DIDR1**	-	-	-	-	-	-	AIN1D	AIN0D
$7E	**DDIR0**	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D

$7D	**ADTM**	-	-	-	-	-	ADCTM2	ADCTM1	ADCTM0
$7C	**ADMUX**	REFSX1	REFSX0	ADLAR	CHMUX4	CHMUX3	CHMUX2	CHUMX1	CHMUX0
$7B	**ADCSRB**	-	ACME	ADICTL	ADGAIN1	ADGAIN0	ADTSX2	ADTSX1	ADTSX0
$7A	**ADCSRA**	ADCEN	ADSC	ADATE	ADIF	ADIE	ADPSX2	ADPSX1	ADPSX0
$79	**ADCH**	ADC 转换结果寄存器高字节							
$78	**ADCL**	ADC 转换结果寄存器低字节							
$77~$74	**RESERVED**	-	-	-	-	-	-	-	-
$73	**PCMSK3**	PCINT31	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24
$72~$70	**RESERVED**	-	-	-	-	-	-	-	-
$6F	**TIMSK1**	-	-	TICIE1	-	-	OCIE1B	OCIE1A	TOIE1
$6E	**TIMSK0**	-	-	-	-	-	-	OCIE0A	TOIE0
$6D	**PCMSK2**	-	PCINT22	-	-	PCINT19	PCINT18	PCINT17	PCINT16
$6C	**PCMSK1**	-	-	-	-	PCINT11	PCINT10	PCINT9	PCINT8
$6B	**PCMSK0**	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
$6A	**RESERVED**	-	-	-	-	-	-	-	-
$69	**EICRA**	-	-	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00
$68	**PCICR**	-	-	-	-	PCIE3	PCIE2	PCIE1	PCIE0
$67	**RESERVED**	-	-	-	-	-	-	-	-
$66	**OSCCAL**	-	PRESTEN	内部 16MHz RC 校准寄存器（6 位）					
$65	**RESERVED**	-	-	-	-	-	-	-	-
$64	**PRR**	-	-	PRTC0	-	PRTC1	-	PRUSART	PRADC
$63~$62	**RESERVED**	-	-	-	-	-	-	-	-
$61	**CLKPR**	CLKPCE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0
$60	**WDTCSR**	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
$5F ($3F)	**SREG**	I	T	H	S	V	N	Z	C
$5E ($3E)	**SPH**	堆栈指针高字节							
$5D ($3D)	**SPL**	堆栈指针低字节							
$5C ($3C)	**RESERVED**	-	-	-	-	-	-	-	-
$5B ($3B)	**RESERVED**	-	-	-	-	-	-	-	-
$5A ($3A)	**RESERVED**	-	-	-	-	-	-	-	-
$59 ($39)	**RESERVED**	-	-	-	-	-	-	-	-
$58 ($38)	**RESERVED**	-	-	-	-	-	-	-	-
$57 ($37)	**RESERVED**	-	-	-	-	-	-	-	-
$56 ($36)	**RESERVED**	-	-	-	-	-	-	-	-
$55 ($35)	**MCUCR**	SWDD	-	-	PUD	-	-	-	-
$54 ($34)	**MCUCSR**	-	-	SWRF	OCDRF	WDTRF	BORF	EXTRF	PORF
$53 ($33)	**SMCR**	-	-	-	-	SMOD2	SMOD1	SMOD0	SE
$52 ($32)	**RESERVED**	-	-	-	-	-	-	-	-
$51 ($31)	**RESERVED**	-	-	-	-	-	-	-	-
$50 ($30)	**ACSR**	ACD	ACBG	ACO	ACIF	ACIE	ACIC	ACIS1	ACIS0
$4F ($2F)	**RESERVED**	-	-	-	-	-	-	-	-
$4E ($2E)	**RESERVED**	-	-	-	-	-	-	-	-
$4D ($2D)	**RESERVED**	-	-	-	-	-	-	-	-
$4C ($2C)	**RESERVED**	-	-	-	-	-	-	-	-

$4B ($2B)	**GPIOR2**	可编程 I/O 寄存器 2							
$4A ($2A)	**GPIOR1**	可编程 I/O 寄存器 1							
$49 ($29)	**EEDHR**	EFLASH 数据寄存器高字节							
$48 ($28)	**RESERVED**	-	-	-	-	-	-	-	-
$47 ($27)	**OCR0A**	定时计数器 0 输出比较寄存器							
$46 ($26)	**TCNT0**	定时计数器 0 计数器							
$45 ($25)	**TCCR0B**	FOC0A	-	-	-	WGM02	CS02	CS01	CS00
$44 ($24)	**TCCR0A**	COM0A1	COM0A0	-	-	-	-	WGM01	WGM00
$43 ($23)	**GTCCR**	TSM	-	-	-	-	-	-	PSR10
$42 ($22)	**EEARH**	-	-	-	EEPROM/EFLASH 地址寄存器高 4 位				
$41 ($21)	**EEARL**	EEPROM/EFLASH 地址寄存器低 8 位							
$40 ($20)	**EEDR**	EEPROM/EFLASH 数据寄存器低 8 位							
$3F ($1F)	**EECR**	EEPM2	-	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE
$3E ($1E)	**GPIOR0**	可编程 I/O 寄存器 0							
$3D ($1D)	**EIMSK**	-	-	-	-	-	INT2	INT1	INT0
$3C ($1C)	**EIFR**	-	-	-	-	-	INTF2	INTF1	INTF0
$3B ($1B)	**PCIFR**	-	-	-	-	PCIF3	PCIF2	PCIF1	PCIF0
$3A ($1A)	**RESERVED**	-	-	-	-	-	-	-	-
$39 ($19)	**RESERVED**	-	-	-	-	-	-	-	-
$38 ($18)	**RESERVED**	-	-	-	-	-	-	-	-
$37 ($17)	**RESERVED**	-	-	-	-	-	-	-	-
$36 ($16)	**TIFR1**	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1
$35($15)	**TIFR0**	-	-	-	-	-	-	OCF0A	TOV0
$34 ($14)	**RESERVED**	-	-	-	-	-	-	-	-
				.....					
$2C ($0C)	**RESERVED**	-	-	-	-	-	-	-	-
$2B ($0B)	**PORTD**	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
$2A ($0A)	**DDRD**	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
$29 ($09)	**PIND**	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
$28 ($08)	**PORTC**	-	PORTC6	-	-	PORTC3	PORTC2	PORTC1	PORTC0
$27 ($07)	**DDRC**	-	DDRC6	-	-	DDRC3	DDRC2	DDRC1	DDRC0
$26 ($06)	**PINC**	0	PINC6	0	0	PINC3	PINC2	PINC1	PINC0
$25 ($05)	**PORTB**	-	-	-	-	PORTB3	PORTB2	PORTB1	PORTB0
$24 ($04)	**DDRB**	-	-	-	-	DDRB3	DDRB2	DDRB1	DDRB0
$23 ($03)	**PINB**	0	0	0	0	PINB3	PINB2	PINB1	PINB0
$22 ($02)	**PORTA**	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
$21 ($01)	**DDRA**	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
$20 ($00)	**PINA**	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0

指令集速查表

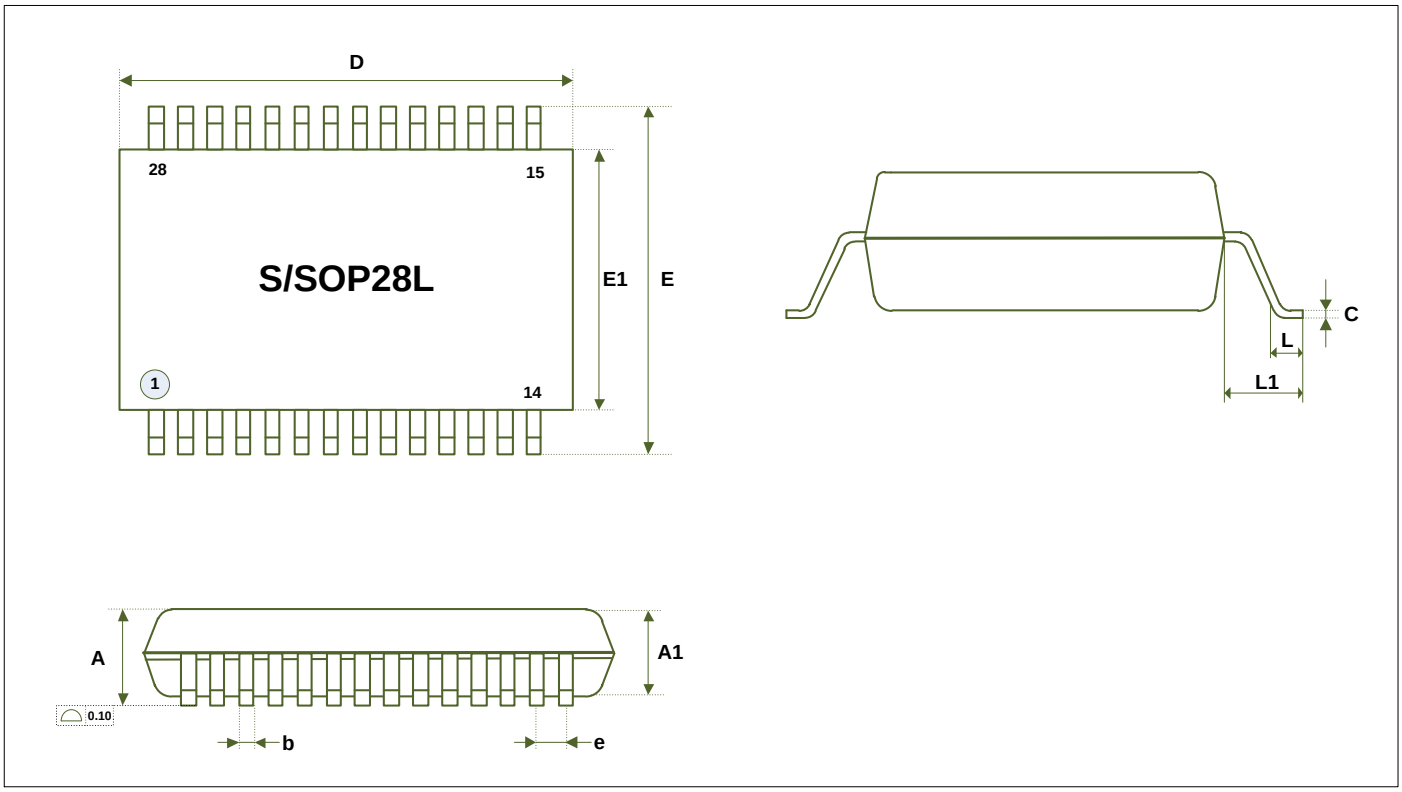
指令	操作数	描述	操作	标记位	周期
算术逻辑运算指令					
ADD	R_d, R_r	寄存器相加	$R_d \leftarrow R_d + R_r$	Z,C,N,V,H	1
ADC	R_d, R_r	带进位的寄存器相加	$R_d \leftarrow R_d + R_r + C$	Z,C,N,V,H	1
ADIW	R_{dl}, K	立即数与字相加	$R_{dh}:R_{dl} \leftarrow R_{dh}:R_{dl} + K$	Z,C,N,V,S	1
SUB	R_d, R_r	寄存器相加减	$R_d \leftarrow R_d - R_r$	Z,C,N,V,H	1
SUBI	R_d, K	寄存器减常数	$R_d \leftarrow R_d - K$	Z,C,N,V,H	1
SBC	R_d, R_r	带借位的寄存器相加减	$R_d \leftarrow R_d - R_r - C$	Z,C,N,V,H	1
SBCI	R_d, K	带借位的寄存器减常数	$R_d \leftarrow R_d – K - C$	Z,C,N,V,H	1
SBIW	R_{dl}, K	立即数与字相减	$R_{dh}:R_{dl} \leftarrow R_{dh}:R_{dl} - K$	Z,C,N,V,S	1
AND	R_d, R_r	逻辑与	$R_d \leftarrow R_d \& R_r$	Z,N,V	1
ANDI	R_d, K	寄存器逻辑与常数	$R_d \leftarrow R_d \& K$	Z,N,V	1
OR	R_d, R_r	逻辑或	$R_d \leftarrow R_d \mid R_r$	Z,N,V	1
ORI	R_d, K	寄存器逻辑或常数	$R_d \leftarrow R_d \mid K$	Z,N,V	1
EOR	R_d, R_r	寄存器异或	$R_d \leftarrow R_d \oplus R_r$	Z,N,V	1
COM	R_d	反码	$R_d \leftarrow \$FF - R_d$	Z,C,N,V	1
NEG	R_d	2 禁制补码	$R_d \leftarrow \$00 - R_d$	Z,C,N,V,H	1
SBR	R_d, K	设置寄存器中的位	$R_d \leftarrow R_d$ v K	Z,N,V	1
CBR	R_d, K	清寄存器中的位	$R_d \leftarrow R_d$ v ($FF – K)	Z,N,V	1
INC	R_d	递增	$R_d \leftarrow R_d + 1$	Z,N,V	1
DEC	R_d	递减	$R_d \leftarrow R_d - 1$	Z,N,V	1
TST	R_d	测试为 0 或负数	$R_d \leftarrow R_d \& R_d$	Z,N,V	1
CLR	R_d	清寄存器	$R_d \leftarrow R_d \oplus R_d$	Z,N,V	1
SER	R_d	寄存器全设置为 1	$R_d \leftarrow \$FF$	None	1
MUL	R_d, R_r	无符号乘法	R_1: $R_0 \leftarrow R_d \times R_r$	Z,C	1
MULS	R_d, R_r	有符号乘法	R_1: $R_0 \leftarrow R_d \times R_r$	Z,C	1
MULSU	R_d, R_r	有符号数乘无符号数	R_1: $R_0 \leftarrow R_d \times R_r$	Z,C	1
FMUL	R_d, R_r	无符号乘法，移位	R_1: $R_0 \leftarrow (R_d \times R_r) << 1$	Z,C	1
FMULS	R_d, R_r	有符号乘法，移位	R_1: $R_0 \leftarrow (R_d \times R_r) << 1$	Z,C	1
FMULSU	R_d, R_r	有符号数乘无符号数，移位	R_1: $R_0 \leftarrow (R_d \times R_r) << 1$	Z,C	1
跳转指令					
RJMP	K	相对跳转	PC ← PC + K + 1	None	1
IJMP		间接跳转（到 Z 指向地址）	PC ← Z	None	2
JMP	K	直接跳转	PC ← K	None	2
RCALL	K	相对地址子程序调用	PC ← PC + K + 1	None	1
ICALL		间接子程序调用（Z 指向地址）	PC ← Z	None	2
CALL	K	直接子程序调用	PC ← K	None	2
RET		子程序返回	PC ← Stack	None	2
RETI		中断返回	PC ← Stack	I	2

指令	操作数	描述	操作	标记位	周期
跳转指令（续）					
CPSE	R_d, R_r	相等即跳转	If(R_d=R_r) PC ← PC + 2 or 3	None	1/2
CP	R_d, R_r	比较	R_d - R_r	Z,N,V,C,H	1
CPC	R_d, R_r	带进位比较	R_d - R_r – C	Z,N,V,C,H	1
CPI	R_d, K	与立即数比较	R_d - K	Z,N,V,C,H	1
SBRC	R_r, b	位为 0 即跳过下一条指令	If(R_r(b)=0) PC ← PC + 2 or 3	None	1/2
SBRS	R_r, b	位为 1 即跳过下一条指令	If(R_r(b)=1) PC ← PC + 2 or 3	None	1/2
SBIC	P, b	I/O 位为 0 即跳过下一条指令	If(P(b)=0) PC ← PC + 2 or 3	None	1/2
SBIS	P, b	I/O 位为 1 即跳过下一条指令	If(P(b)=1) PC ← PC + 2 or 3	None	1/2
BRBS	s, k	状态标记为 1 即跳转	If(SREG(S)=1) PC ← PC + K + 1	None	1/2
BRBC	s, k	状态标记为 0 即跳转	If(SREG(S)=0) PC ← PC + K + 1	None	1/2
BREQ	k	相等即跳转	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	不等即跳转	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	进位则跳转	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	无进位则跳转	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	不小于则跳转	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	小于则跳转	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	为负则跳转	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	为正则跳转	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	有符号的不小于即跳转	if (N⊕V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	有符号的小于 0 即跳转	if (N⊕V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	半进位为 1 则跳转	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	半进位为 0 则跳转	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	T 置位则跳转	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	T 清零则跳转	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	溢出则跳转	f (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	不溢出则跳转	f (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	全局中断使能则跳转	f (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	全局中断禁止则跳转	f (I = 0) then PC ← PC + k + 1	None	1/2
数据传输指令					
MOV	Rd, Rr	寄存器之间移动数据	Rd ← Rr	None	1
MOVW	Rd, Rr	移动一个字的数据	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	加载立即数	Rd ← K	None	1
LD	Rd, X	间接加载	Rd ← (X)	None	1
LD	Rd, X+	间接加载，地址递增	Rd ← (X), X ← X + 1	None	1
LD	Rd, -X	地址递减，间接加载	X ← X - 1, Rd ← (X)	None	1
LD	Rd, Y	间接加载	Rd ← (Y)	None	1
LD	Rd, Y+	间接加载，地址递增	Rd ← (Y), Y ← Y + 1	None	1
LD	Rd, -Y	地址递减，间接加载	Y ← Y - 1, Rd ← (Y)	None	1
LDD	Rd, Y+q	带偏移量的间接加载	Rd ← (Y + q)	None	1
LD	Rd, Z	间接加载	Rd ← (Z)	None	1

指令	操作数	描述	操作	标记位	周期
数据传输指令（续）					
LD	Rd, Z+	间接加载，地址递增	Rd ← (Z), Z ← Z+1	None	1
LD	Rd, -Z	地址递减，间接加载	Z ← Z - 1, Rd ← (Z)	None	1
LDD	Rd, Z+q	带偏移量的间接加载	Rd ← (Z + q)	None	1
LDS	Rd, k	直接从 SRAM 中加载	Rd ← (k)	None	2
ST	X, Rr	间接存储	(X) ← Rr	None	1
ST	X+, Rr	间接存储，地址递增	(X) ← Rr, X ← X + 1	None	1
ST	-X, Rr	地址递减，间接存储	X ← X - 1, (X) ← Rr	None	1
ST	Y, Rr	间接存储	(Y) ← Rr	None	1
ST	Y+, Rr	间接存储，地址递增	(Y) ← Rr, Y ← Y + 1	None	1
ST	-Y, Rr	地址递减，间接存储	Y ← Y - 1, (Y) ← Rr	None	1
STD	Y+q, Rr	带偏移量的间接存储	(Y + q) ← Rr	None	1
ST	Z, Rr	间接存储	(Z) ← Rr	None	1
ST	Z+, Rr	间接存储，地址递增	(Z) ← Rr, Z ← Z + 1	None	1
ST	-Z, Rr	地址递减，间接存储	Z ← Z - 1, (Z) ← Rr	None	1
STD	Z+q, Rr	带偏移量的间接存储	(Z + q) ← Rr	None	1
STS	k, Rr	直接存储到 SRAM 中	(k) ← Rr	None	2
LPM		加载程序空间数据	R0 ← (Z)	None	2
LPM	Rd, Z	加载程序空间数据	Rd ← (Z)	None	2
LPM	Rd, Z+	加载程序数据，地址递增	Rd ← (Z), Z ← Z+1	None	2
IN	Rd, P	读端口	Rd ← P	None	1
OUT	P, Rr	写端口	P ← Rr	None	1
PUSH	Rr	压栈	STACK ← Rr	None	1
POP	Rd	出栈	Rd ← STACK	None	1
位操作指令					
SBI	P, b	设置 IO 寄存器	I/O(P, b) ← 1	None	1
CBI	P, b	清零 IO 寄存器	I/O(P, b) ← 0	None	1
LSL	Rd	逻辑左移	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	逻辑右移	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z	1
ROL	Rd	包含进位的循环左移	Rd(0)←C, Rd(n+1) ← Rd(n), C←Rd(7)	Z	1
ROR	Rd	包含进位的循环右移	Rd(7)←C, Rd(n) ← Rd(n+1), C←Rd(0)	Z	1
ASR	Rd	算术右移	Rd(n) ← Rd(n+1), n=0:6	Z	1
SWAP	Rd	位交换	Rd(3:0) ← Rd(7:4), Rd(7:4) ← Rd(3:0)	None	1
BSET	s	设置状态位	SREG(s) ← 1	SREG(s)	1
BCLR	s	清零状态位	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	存储到 T 位	T ← Rr(b)	T	1
BLD	Rd, b	读出 T 位到寄存器	Rd(b) ← T	None	1
SEC		设置进位标志	C ← 1	C	1
CLC		清楚进位标志	C ← 0	C	1
SEN		设置负数标志	N ← 1	N	1

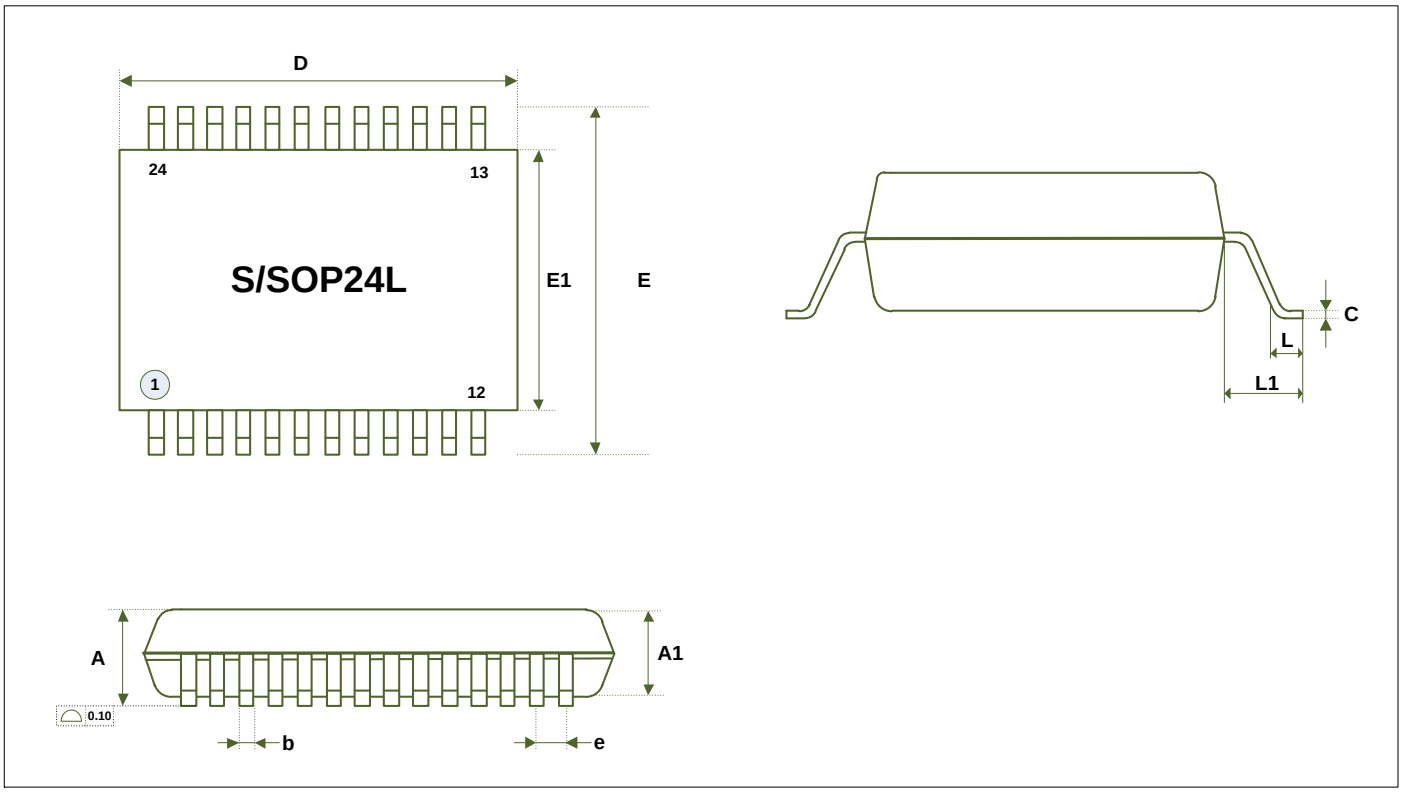
指令	操作数	描述	操作	标记位	周期
位操作指令（续）					
CLN		清除负数标志	N ← 0	N	1
SEZ		设置零标志	Z ← 1	Z	1
CLZ		清除零标志	Z ← 0	Z	1
SEI		使能全局中断	I ← 1	I	1
CLI		禁制全局中断	I ← 0	I	1
SES		设置符号测试标志	S ← 1	S	1
CLS		清除符号测试标志	S ← 0	S	1
SEV		设置二进制补码溢出标志	V ← 1	V	1
CLV		清除二进制补码溢出标志	V ← 0	V	1
SET		设置 T 位（SREG）	T ← 1	T	1
CLT		清除 T 位（SREG）	T ← 0	T	1
SEH		设置半进位标志（SREG）	H ← 1	H	1
CLH		清除半进位标志（SREG）	H ← 0	H	1
MCU 控制指令					
NOP		空指令		None	1
SLEEP		进入休眠模式		None	1
WDR		看门狗复位		None	1
BREAK		软断点	仅用于调试目的	None	N/A

封装参数



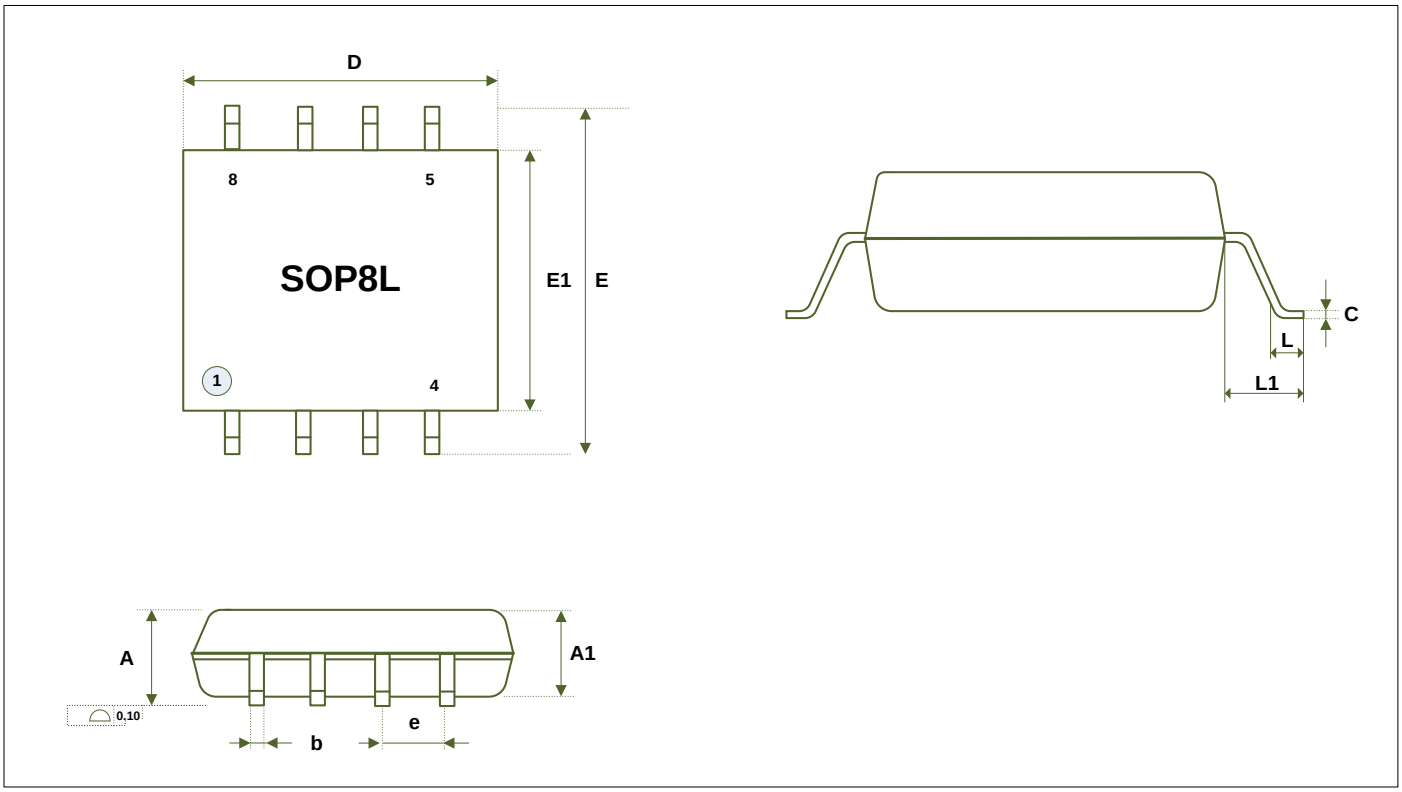
SOP28L 通用尺寸定义

字符代号	最小值	典型值	最大值	单位
A	2.35	2.55	2.80	毫米
A1	2.25	2.45	2.65	毫米
b	0.41	-	0.54	毫米
e	1.17	1.27	1.37	毫米
E	10.20	10.40	10.60	毫米
E1	7.50	7.60	7.70	毫米
D	17.40	17.50	17.60	毫米
C	0.15	-	0.20	毫米
L	0.40	0.60	0.80	毫米
L1	1.40REF			毫米



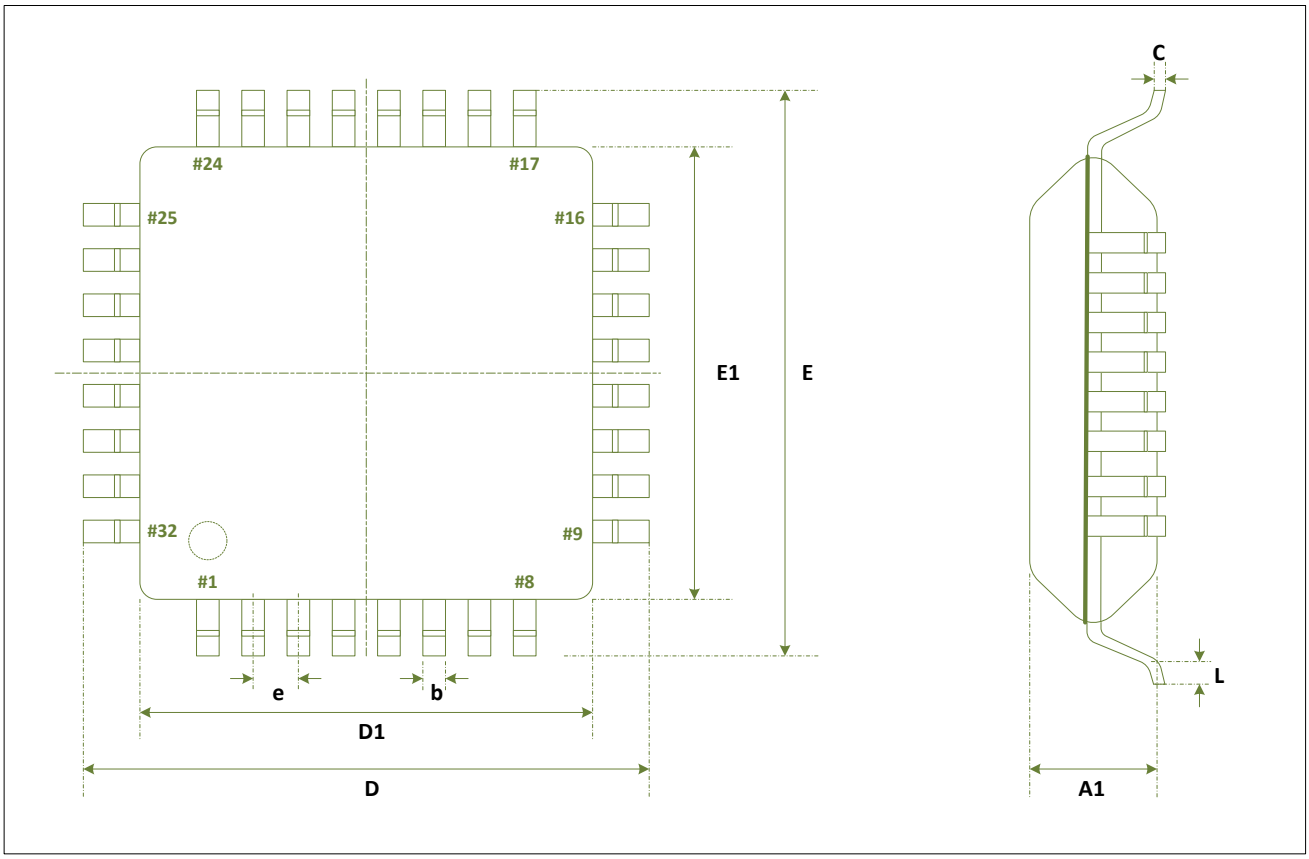
SSOP24L 通用尺寸定义

字符代号	最小值	典型值	最大值	单位
A	1.60	1.65	1.70	毫米
A1	1.40	1.50	1.60	毫米
b	0.25	0.30	0.35	毫米
e	-	0.65	-	毫米
E	7.65	7.80	7.95	毫米
E1	5.15	5.25	5.35	毫米
D	8.10	8.20	8.30	毫米
C	-	0.152	-	毫米
L	0.80	0.90	1.00	毫米
L1	1.15	1.25	1.35	毫米



SOP8L 通用尺寸定义

字符代号	最小值	典型值	最大值	单位
A	1.35	1.55	1.75	毫米
A1	1.25	1.40	1.65	毫米
b	0.38	-	0.51	毫米
e	-	1.27BSC	-	毫米
E	5.80	6.00	6.20	毫米
E1	3.80	3.90	4.00	毫米
D	4.80	4.90	5.00	毫米
C	0.17	-	0.25	毫米
L	0.45	0.60	0.8	毫米
L1	-	1.04REF	-	毫米



LQFP32L 通用尺寸定义

字符代号	最小值	典型值	最大值	单位
D	8.90	9.00	9.10	毫米
D1	6.90	7.00	7.10	毫米
b	0.15	0.20	0.25	毫米
e	0.75	0.80	0.85	毫米
E	8.90	9.00	9.10	毫米
E1	6.90	7.00	7.10	毫米
C	-	0.10	-	毫米
L	0.55	0.60	0.65	毫米
A1	-	1.40	-	毫米

电气特性

最大工作环境

工作温度	-40℃ ~ +85℃	**注意事项**
存储温度	-55℃ ~ +125℃	芯片工作在超出此处的工作环境，将有可能导致芯片工作不稳定，甚至会导致芯片本身的严重损坏。此处的最大工作环境仅仅在其他工作条件都在正常范围内时有效。将芯片长时间工作在最大环境下，可能会影响芯片的稳定性和使用寿命。
最大工作电压	3.6V	
I/O 最大驱动电流	25mA	
最大工作频率	20MHz	
电源地最大电流	200mA	

直流特性

T_A = -40℃ ~ +85℃， Vcc = 1.8V ~ 3.6V

符号	参数	条件	最小值	典型值	最大值	单位
V_{IL}	输入低电压	Vcc = 1.8V-3.6V	-0.3		0.1Vcc	V
V_{IH}	输入高电压	Vcc = 1.8V-3.6V	0.6Vcc		Vcc+0.5	
V_{IL1}	输入低电压 XTAL2 引脚	Vcc = 1.8V-3.6V	-0.5		0.1Vcc	
V_{IH1}	输入高电压 XTAL2 引脚	Vcc = 1.8V-3.6V	0.7Vcc		Vcc+0.5	
V_{OL}	输出低电压	IOL=6mA，Vcc=3.6V			0.5	
V_{OH}	输出高电压	IOH=-10mA, Vcc=3.6V	2.3			
I_{IL}	I/O 输入漏电	Vcc = 3.3V			1	uA
I_{IH}	I/O 输入漏电	Vcc = 3.3V			1	
R_{PU}	I/O 上拉电阻		28	41	62	KΩ
I_{CC}	工作电流	16MHz, 3.3V		5.9		mA
		16MHz, 1.8V		3.8		
		4MHz, 3.3V		3.3		
		4MHz, 1.8V		1.4		
		1MHz, 3.3V		2.4		
		1MHz, 1.8V		0.78		
	休眠模式			35		uA

内部 16MHz 可校准 RC 特性

参数	最小值	典型值	最大值	单位	说明
工作电压	1.62	1.8	1.98	V	
工作电流		300		uA	
休眠电流			10	uA	
输出频率		16		MHz	
输出频率精度		+/1		%	*+/1%为校准后的精度*
输出占空比	45		55	%	

ADC 电气参数

T_A = 25℃, Vcc = 3.3V, F_{CLK} = 3MHz (F_S = 250 KHz)

参数	最小值	典型值	最大值	单位	说明
工作电压	1.62	1.8	1.98	V	
工作电流		800		uA	
休眠电流		20		uA	
模拟输入特性					
参考输入阻抗		20		Kohm	
模拟输入阻抗		20		Kohm	
模拟输入电容			5	pF	
静态性能参数					
分辨率		10		Bit	
输出数据类型	最小值			000H	
	最大值			3FFH	
DNL	-1		+1	LSB	
INL	-1.5		+1.5	LSB	
增益误差		+/-6		%	
数字接口特性					
采样频率（F_S）	0.025	0.25	-	MHz	
工作频率	0.3	3		MHz	
数据转换周期		12		Cycles	
唤醒周期		50		Cycles	1,2
SNR	52	56		dB	
SFDR	60	62		dB	

[1]: 唤醒 ADC 模块需要 50 个采样周期的等待时间后，才能开始采样

[2]: 当禁止 ADC 超过 5 个采样周期后，ADC 需要重新唤醒

更新日志

V1.6.6 2015-02-03	更新 SOP14L 封装定义
V1.6.5 2012-12-14	更新了 SOP8L 封装定义和封装参数
V1.6.4 2012-11-22	修正了 RCCAL 寄存器描述部分的寄存器地址(正确地址为 0x66)
V1.6.2 2012-10-25	修改了 LQFP32L 封装定义的错误 PA7 对应的第二功能为 ADC7
V1.6.1 2012-10-15	增加了 LQFP32L 封装定义
V1.6 2012-9-05	更新了不同工作电压下的最大工作频率
V1.5 2012-8-07	去除模拟比较器对 ADC 通道输入的支持功能 LGT8F0XA 系列不支持比较器通道扩展功能 增加对 ISP 编程引脚的描述
V1.4 2012-6-26	重要更新 更新了部分寄存器命名的不一致 更新了部分内容描述的错误，主要为寄存器定义 **增加了 SOP24 封装的定义，更新 SOP20 的封装定义** **更新了 PMCR 寄存器中对外部晶振 I/O 使能控制的定义，请用户特别注意 PMCR 中关于 OSCMEN 的定义。** 增加了芯片的部分电气特性定义
V1.3 2012-4-26	修改了一些寄存器说明的错误 增加了对 I/O 使用的注意事项说明，涉及到 PINA/B/C/D 相关的操作 **建议用户特别注意(避免直接在 PINA/B/C/D 寄存器上使用 SBIC/SBIS 指令)**
V1.2 2012-4-18	更新了 PC0/1 的引脚定义 更新了端口复用控制部分描述

	增加了 SOP28L 封装参数 修改了部分寄存器定义错误
V1.1 2012-4-9	修改了一些寄存器定义的错误 增加了第一版遗漏的一些寄存器定义 修改了中断向量编号
V1.0	初始版本