Freetech 北京飞锐泰克科技有限公司

MC9S08SH 中文数据手册

目 录

MC9S08SH4/8: 特征

8 位 HCS08 中央处理器(CPU)

- 40-MHz CPU 时钟
- HC08 指令集＋新增的 BGND 指令
- 可支持多达 32 个中断/复位源

存储器

MC9S08SH8

- 8KB FLASH，在芯片工作电压(2.7V~5.5V)范围内可擦除编程
- 512 字节片上 RAM

MC9S08SH4

- 4KB FLASH，在芯片工作电压(2.7V~5.5V)范围内可擦除编程
- 256 字节片上 RAM

节电模式

- 两种极低功耗的停止（STOP）模式
- 节电的等待（WAIT）模式
- 低功耗实时时钟可在正常，WAIT 和 STOP 模式下运行

时钟源

- ICS－内部时钟源模块，包含一个由内部参考或外部参考时钟控制的锁频环(FLL)。内部参考时钟精度可调，分辨率 0.2%；芯片全工作温度和电压范围内时钟频率仅 2%误差。总线频率 2MHz~20MHz。
- XOSC－精确的环控晶体振荡模块，软件选择外接晶体或陶瓷振荡器范围：31.25kHz~38.4kHz；1MHz~16MHz。

系统保护机制

- 看门狗复位，看门狗时钟可以是专门的 1-kHz 时钟或总线时钟
- 具有复位或中断功能的低电压检测模块
- 非法操作码错误检测复位
- 非法地址检测复位
- FLASH 块保护

开发支持

- 单线后台调试接口
- 在线调试可设置一个断点（外加片上调试模块两个断点），故可设置 3 个断点
- 片上实时总线捕捉在线仿真(ICE)

片上外设模块

- ADC－12 通道，10 位模数转换器，2.5us 转换时间，具有自动比较功能，内置温度传感器，内部能隙参考电压；可在 STOP3 模式下运行
- ACMP－具有中断功能的模拟比较器，比较器输出可为上升沿，下降沿或任意边沿；比较器输入端可连接到内部参考电压；比较器输出可连接到 TPM 模块；可在 STOP3 模式下运行。
- SCI－异步通信串口（UART），全双工 NRZ;可作为 LIN 总线主机产生 BREAK 字符;可作为 LIN 总线从机检测 BREAK 字符；可由有效的边沿信号唤醒。
- SPI－同步串行通信接口;发送和接收双缓冲；主机或从机模式；可选的高位在前或低位在前数据格式。
- IIC－波特率高达 100kbps；支持多主机操作；可编程的从机地址；数据传送可产生中断，支持广播指令和 10 位寻址。
- MTIM－8 位模计数器，8 位分频系数；溢出中断功能
- TPMx－2 个 2 通道 16 位计数器。每个通道可用于输入捕捉，比较输出，边缘对齐 PWM 或具有缓冲中心对齐 PWM
- RTC－8 位实时时钟模计数器；采用外部时钟源可用于计时或任务调度功能；采用内部低功耗的 1kHz 时钟可作为定时唤醒源；可运行于单片机的所有工作模式。

输入/输出

- 通用输入/输出(I/O)引脚 17 个和一个只输出功能的引脚
- 8 个极性可选的中断引脚
- 一次写操作就可以更改 PTB[5:2]和 PTC[3:0]这 8 个引脚状态
- 用作输入端时，端口可软件选择上拉 ；用作输出端时，端口可软件选择驱动能力和压摆率(slew rate)

封装

- 24 引脚 QFN
- 20 引脚 TSSOP
- 20 引脚 PDIP
- 16 引脚 TSSOP
- 8 引脚 SOIC

第一章 芯片概述

1.1 简介

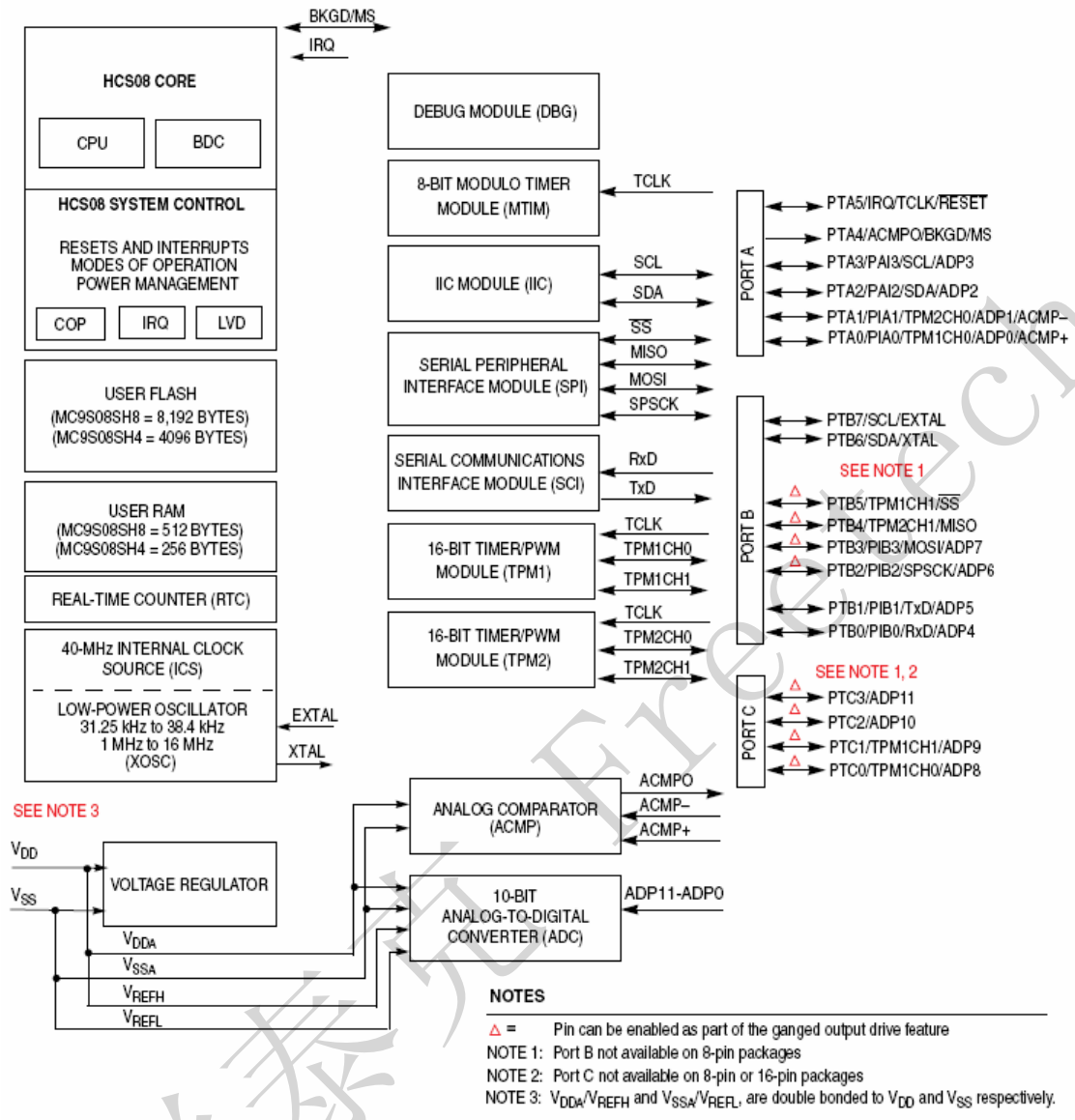
MC9S08SH8 是低功耗，高性能 HCS08 内核单片机家族的成员。

Feature	9S08SH8				9S08SH4			
FLASH size (bytes)	8192				4096			
RAM size (bytes)	512				256			
Pin quantity	24	20	16	8	24	20	16	8
ACMP	yes							
ADC channels	12	12	8	4	12	12	8	4
DBG	yes							
ICS	yes	yes	yes	yes[1]	yes	yes	yes	yes [1]
IIC	yes							
MTIM	yes							
Pin Interrupts	8	8	8	4	8	8	8	4
Pin I/O [2]	17	17	13	5	17	17	13	5
RTC	yes							
SCI	yes	yes	yes	no	yes	yes	yes	no
SPI	yes	yes	yes	no	yes	yes	yes	no
TPM1 channels	2	2	2	1	2	2	2	1
TPM2 channels	2	2	2	1	2	2	2	1
XOSC	yes	yes	yes	no	yes	yes	yes	no

表 1.1 芯片封装与特征

注 1）8 引脚芯片无 FBE 和 FEE 模式

1.2 内部原理图



注：1）8 引脚芯片无 PORT B

2）8 引脚或 16 引脚无 PORT C

图 1.1 内部原理图

1.3 内部时钟分布图

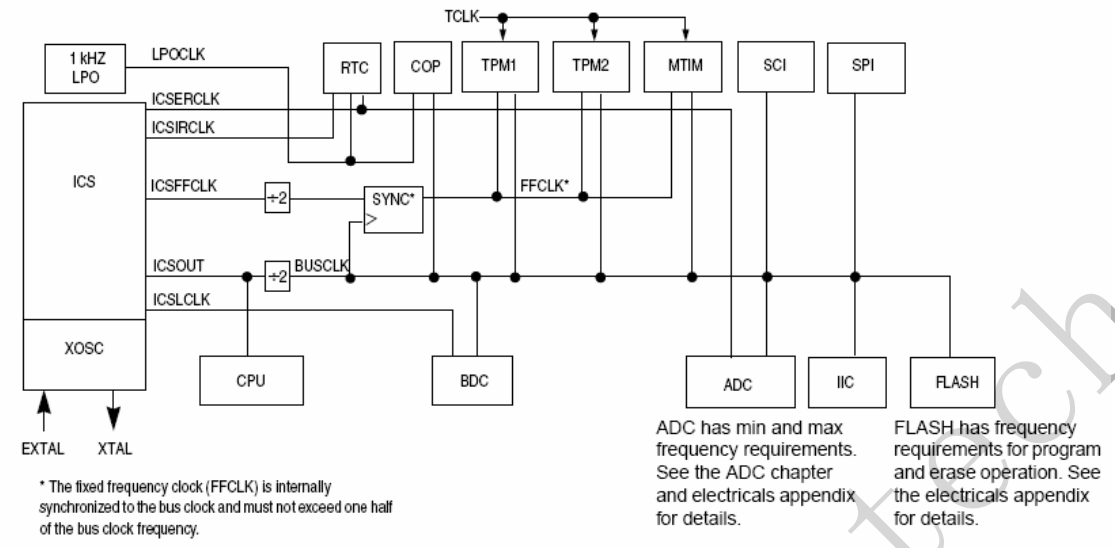


图 1.3 内部时钟分布图

ICG 模块为 MCU 提供以下几种时钟信号：

- ICGOUT：ICG 输出时钟。
- BUSCLK：总新频率，ICSOUT 的一半。
- ICGLCLK：当系统总线时钟低时，开发工具可以选择自时钟（～8MHz）用于 BDC 通信。
- ICGERCLK：外部参考时钟，实时时钟可以选择该时钟源。该时钟源也可以作 ADC 模块的 ALTCLK 时钟。
- ICSIRCLK：内部参考时钟，可为 RTC 提供时钟源
- ICSFFCLK：固定频率时钟，可为 TPM1,TPM2,MTIM 提供时钟源
- TCLK：为 TPM1,TPM2,MTIM 提供外部时钟

第二章 引脚与外部连接

2.1 简介

本章介绍单片机与外部连接细节

2.2 引脚分布

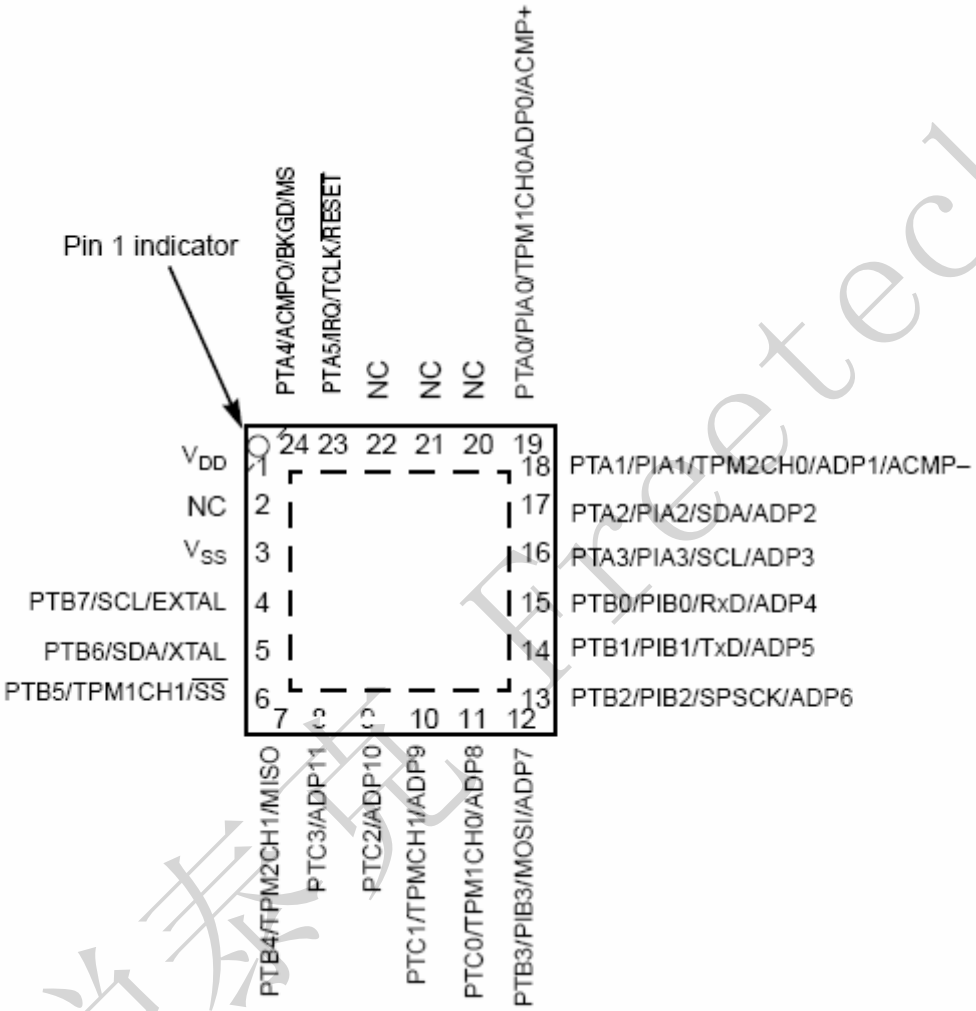


图 2.1 24-引脚 QFN 封装

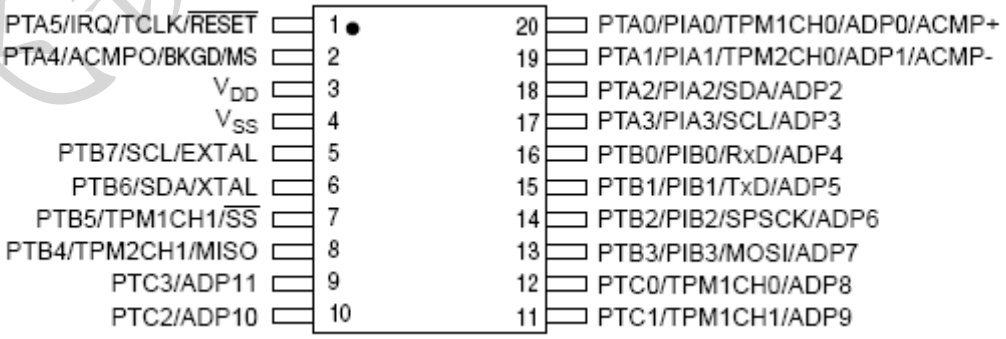


图 2.2 20-引脚 TSSOP 封装

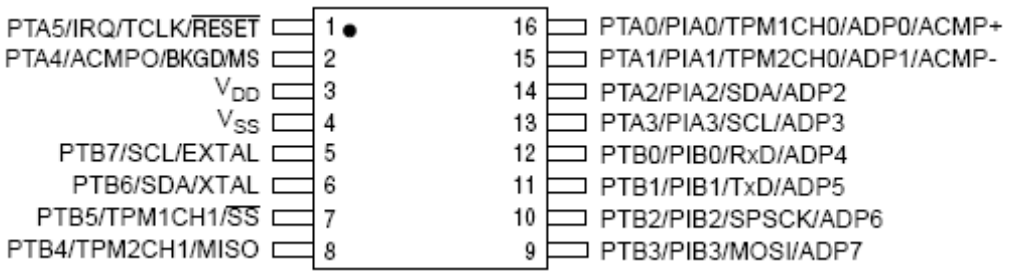


图 2.3 16 引脚 TSSOP 封装

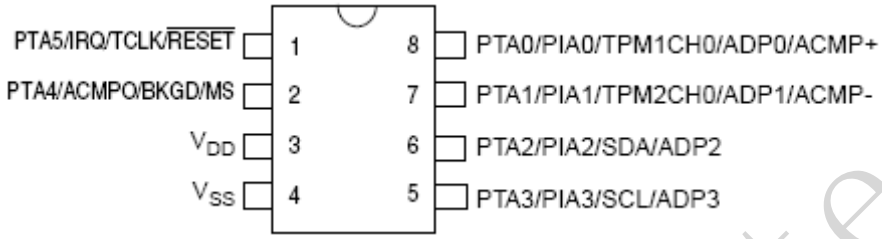
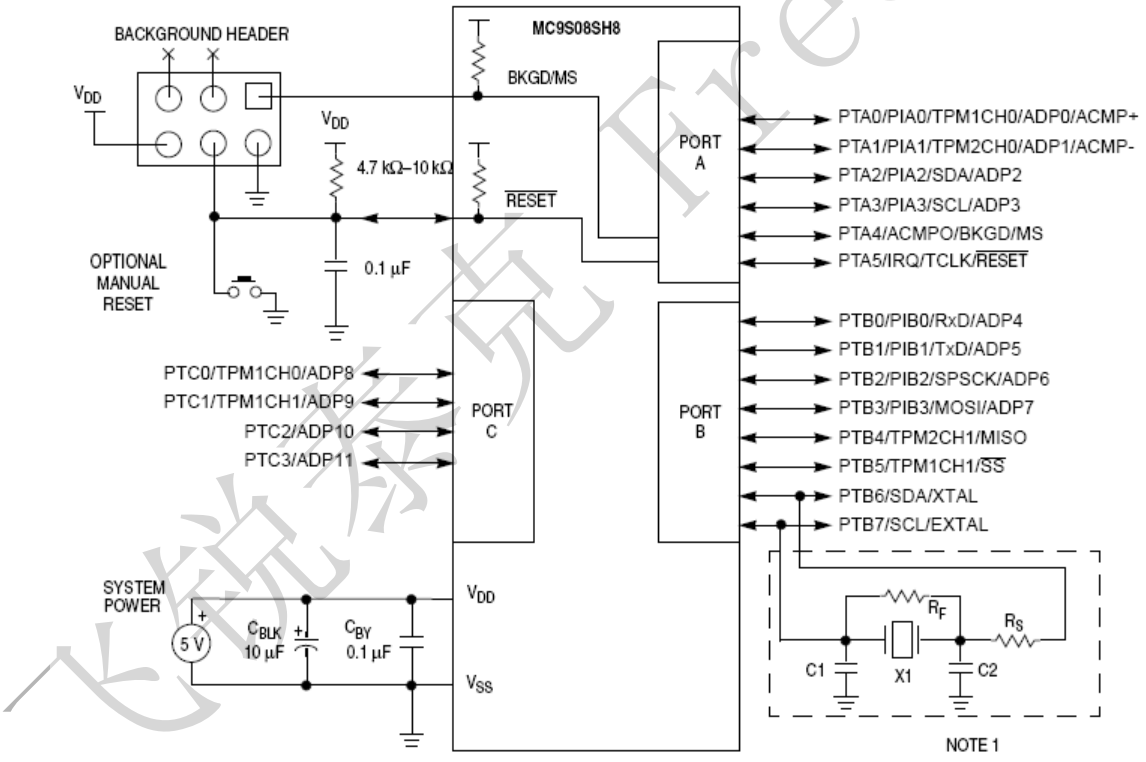


图 2.4 8 引脚 SOIC 封装

2.3 推荐连接图



NOTES:
1. External crystal circuit not required if using the internal clock option.
2. RESET pin can only be used to reset into user mode, you can not enter BDM using RESET pin. BDM can be entered by holding MS low during POR or writing a 1 to BDFR in SBDFR with MS low after issuing BDM command.
3. RC filter on RESET pin recommended for noisy environments.

图 2.5 推荐原理图

注：

1）当使用内部时钟时，外部晶振电路可省去。

2）RESET 引脚只用于将单片机进入用户模式；在上电复位期间将 MS 引脚拉低可进入 BDM

模式或发出 BDM 命令后，MS 引脚为低，向 SBDFR 寄存器的 BDFR 位写 1。
3）复位引脚的 RC 滤波抑制噪声干扰。

2.3.1 电源（V_{DD},V_{SS},V_{DDAD},V_{SSAD}）

V_{DD},V_{SS} 是 MCU 的主电源

V_{DDAD},V_{SSAD} 是 MCU 的模拟电源，给 ADC 供电。

建议在尽量靠近电源引脚处放置一个 10uF 和 0.1uF 的电容。

2.3.2 Oscillator(XTAL，EXTAL)

图 2.5 中，RS 和 RF 应选用低固有电感的电阻，例如炭膜电阻，而绕线和一些金属膜电阻有太大的固有电感，避免使用。C1 和 C2 选用高品质高频陶瓷电容。

RF 用于晶振起振过程中提供一个偏置路径，以使 EXTAL 输入在它的线性范围内。可以选用 1M~10M 欧姆的电阻，该电阻不是很关键。

C1~C2 典型值 5-pF~25-pF,同时应考虑电路板和芯片的固有容性，通常在 EXTAL 和 XTAL 引脚上各有 10pF 的固有电容。

2.3.3 $\overline{\text{RESET}}$

上电复位之后（POR），PTA5/IRQ/TCLK/RESET 引脚默认为通用 IO，PTA5。设置 SOPT1 寄存器中的 RSTPE 为 1，则该引脚配置为复位功能引脚。

由于具有内部上电复位和低电压检测复位功能，所以没必要给 MCU 施加一个外部复位信号。

复位(非上电复位)发生后，复位引脚被拉低，持续大概 66 个总线周期。复位电路会在 SRS 寄存器中记录复位源。

2.3.4 后台调试/模式选择(BKGD/MS)

在复位期间或后台调试强制复位时，该引脚作为模式选择。复位信号上升沿之后，该引脚立即作为后台调试引脚，用于后台调试的数据通信。该引脚作为 BKGD/MS 引脚时（BKGDPE=1），内部上拉自动使能。

SOPT1 寄存器中的 BKGDPE 使能时，后台调试模块使能。任意复位后 BKGDPE 会被置位，故如果 PTA4/ACMPO/BKGD/MS 引脚作为通用 IO 的话，该位必须清除。

如果该引脚未连接，MCU 在复位信号的上升沿之后进入正常运行模式(用户模式)。

在复位信号上升沿期间，BKGD/MS 被拉低，强制 MCU 进入后台调试模式。

在后台调试模式下，通信协议为 16 个目标单片机 BDC 时钟一位数据。目标 MCU 的 BDC 时钟可以为目标 MCU 的总线时钟，故不应有太大容性负载连接到 BKGD 引脚。由于后台调试协议可以确保信号的上升时间，所以导线等寄生容性不会影响通信。

2.3.5 通信 I/O 和外围模块引脚

MC9S08SH8 多大 17 个通用 IO 和 1 个只输出引脚。

当引脚作为通用输出或外围模块的输出功能时，可以软件设置两种驱动强度之一和是否使能输出压摆率。当引脚作为通用输入或外围模块的输入功能时，软件可以使能上拉。

复位之后，除了只输出功能引脚，所有引脚均为高阻抗通用输入，内部上拉未使能。而输出引脚配置为低驱动能力，输出压摆率控制使能的状态。

当片上外围模块系统控制了复用引脚，该引脚的数据方向控制寄存器仍会决定端口数据寄存器读取源，尽管外围模块控制了引脚方向。

注意：为避免引脚被灌入电流，用户的初始化程序应使能引脚的内部上拉或将该引脚设置为输出脚。

Pin Number				Priority (Lowest → Highest)					
24-pin	20-pin	16-pin	8-pin	Port Pin	Alt 1	Alt 2	Alt 3	Alt 4	Alt5
1	3	3	3						V_{DD}
2	—	—	—						
3	4	4	4						V_{SS}
4	5	5	—	PTB7	SCL[1]	EXTAL			
5	6	6	—	PTB6	SDA[1]	XTAL			
6	7	7	—	PTB5	TPM1CH1[2]	$\overline{SS}$	PTC0[3]		
7	8	8	—	PTB4	TPM2CH1	MISO	PTC0[3]		
8	9	—	—	PTC3			PTC0[3]	ADP11	
9	10	—	—	PTC2			PTC0[3]	ADP10	
10	11	—	—	PTC1		TPM1CH1[2]	PTC0[3]	ADP9	
11	12	—	—	PTC0		TPM1CH0[2]	PTC0[3]	ADP8	
12	13	9	—	PTB3	PIB3	MOSI	PTC0[3]	ADP7	
13	14	10	—	PTB2	PIB2	SPSCK	PTC0[3]	ADP6	
14	15	11	—	PTB1	PIB1	TxD		ADP5	
15	16	12	—	PTB0	PIB0	RxD		ADP4	
16	17	13	5	PTA3	PIA3	SCL[1]		ADP3	
17	18	14	6	PTA2	PIA2	SDA[1]		ADP2	
18	19	15	7	PTA1	PIA1	TPM2CH0		ADP1[4]	ACMP–[4]
19	20	16	8	PTA0	PIA0	TPM1CH0[2]		ADP0[4]	ACMP+[4]
20	—	—	—						
21	—	—	—						
22	—	—	—						
23	1	1	1	PTA5[5]	IRQ	TCLK			$\overline{RESET}$
24	2	2	2	PTA4	ACMPO			BKGD	MS

1） SOPT2 中的 IICPS 位控制 IIC 引脚使能

2） SOPT2 中的 TPM1PS 位控制 TPM1CHx 引脚的使能

3） 群输出功能，这些引脚的输出值，驱动能力和压摆率控制与 PTC0 一样。

4） 如果 ACMP 和 ADC 都使能，那么 ACMP 和 ADC 模块都可以访问该引脚。

5） 该引脚作为输出时，为开漏。

第三章 运行模式

3.1 简介

介绍每种模式的进入，退出及每种模式下的功能

3.2 特点

- 用于代码调试的后台调试模式
- 等待模式
  - －CPU 挂起
  - －系统时钟运行
  - －稳压模块维持
- 停止模式
  - －CPU 和总线停止
  - －STOP2：部分内部电路进入节电模式，RAM 数据保持
  - －STOP3：所有内部电路仍供电，可用于快速恢复

3.3 正常运行模式

MCU 复位之后，BKGD/MS 为高，进入正常运行模式。CPU 预取 0XFFE:0XFFFF 处代码。

3.4 后台调试模式

BDC：后台调试器

DBG：片上调试模块

BDC 和 DBG 一起工作提供了分析 MCU 操作的一种方式。

5 种进入后台调试模式的方式：

- 上电复位期间或发出一个后台调试强制复位时 BKGD/MS 引脚为低
- BKGD 引脚接收到 BACKGROUND 命令
- 执行 BGND 命令
- 遇到 BDC 断点
- 遇到 DBG 断点

进入后台调试模式后，CPU 挂起等待后台调试命令而不会执行用户程序。按后台调试指令按是否影响到 CPU 可以分为两种类型后台调试命令：

- 非干扰指令（硬 BDM 指令）：用户程序正常运行下的 BDM 指令，这些也可以在后台调试下运行。包括：
  - －存储器访问指令
  - －带状态返回存储器访问指令
  - －BDC 寄存器访问指令
  - －BACKGROUND 指令
- 干扰指令：只能在后台调试模式下执行
  - －CPU 寄存器指令
  - －单步跟踪用户程序
  - －离开后台调试模式，进入正常运行模式(GO)

3.5 等待模式

执行 WAIT 指令进入等待模式。

执行完 WAIT 指令，CPU 进入低功耗状态。进入 WAIT 模式之前，设置 CCR 寄存器的 I=0，使能等待模式下中断功能，从而使该中断唤醒 CPU，执行中断服务程序。

在等待模式下，BACKGROUND 和带状态返回访问存储器指令可以执行。BACKGROUND 命令可以唤醒 CPU 进入后台调试模式；带状态返回访问存储器命令不会访问存储器，而是返回一个错误标志，表明 MCU 处于停止状态或等待状态。BACKGROUND 命令可以唤醒 WAIT 状态的 MCU，从而进入后台调试模式。

3.6 停止模式

设置 SOPT1 中 STOPE=1，执行 STOP 指令，MCU 进入 2 种停止模式之一。任何一种方式下，总线和 CPU 时钟均停止。如果执行 STOP 命令之前，STOPE 位未置位，那么执行 STOP 命令不会进入 STOP 模式，而是作为非法操作码而发生复位。通过设置 SPMSC2 中的相应位可以选择不同的停止模式。

STOPE	ENBDM [1]	LVDE	LVDSE	PPDC	Stop Mode
0	x	x		x	Stop modes disabled; illegal opcode reset if STOP instruction executed
1	1	x		x	Stop3 with BDM enabled [2]
1	0	Both bits must be 1		0	Stop3 with voltage regulator active
1	0	Either bit a 0		0	Stop3
1	0	Either bit a 0		1	Stop2

[1] ENBDM is located in the BDCSCR, which is only accessible through BDC commands, see Section 17.4.1.1, "BDC Status and Control Register (BDCSCR)".

[2] When in Stop3 mode with BDM enabled, The S_{IDD} will be near R_{IDD} levels because internal clocks are enabled.

注：1）ENBDM 位于 BDCSCR 寄存器，只能通过 BDC 命令访问

2）BDM 使能情况下进入 STOP3 模式，功耗和正常运行模式接近。

表 3.1 不同的 STOP 模式选择

3.6.1 STOP2 模式

该模式下大多少内部电路电源关闭。但 RAM 电路电源不会关闭。一旦进入 STOP2 模式，所有的 IO 控制信号均被栓锁。

STOP2 模式下 MCU 功耗很低，同时 RAM 中的数据和 I/O 引脚的当前状态处于维持状态。在想进入 STOP2 模式，在执行 STOP 指令之前设置 PPDC=1，STOPE=1。如果在执行 STOP 命令之前，LVD 使能，那么执行 STOP 命令不会进入 STOP2 模式，而是进入 STOP3 模式。

在进入 STOP2 模式之前，用户必须保存 I/O 端口寄存器中的数据到 RAM 中。一旦从 STOP2 模式退出后，用户通过软件将保存的值恢复。

当 MCU 处于 STOP2 模式，所有的由电压稳压器供电的内部电路均关闭，RAM 除外。I/O 引脚闩锁。当从 STOP2 模式退出后，I/O 状态仍处于闩锁状态，直到向 SPMSC2 寄存器中的 PPDACK 位写 1。

当由于复位引脚上的复位信号或外部中断或 RTI 中断可以将 MCU 从 STOP2 模式退出。当 MCU 处于 STOP2 模式，IRQ 总是对低电平敏感而不管进入 STOP2 模

式之前的配置。

一旦 MCU 从 STOP2 模式退出，那么 MCU 将像上电复位那样运行，当然除了引脚状态仍处于闩锁状态外。CPU 运行复位向量处的程序。MCU 的系统和所有外围模块处于其默认状态必须进行初始化。

当 MCU 从 STOP2 模式唤醒，SPMSC2 寄存器中的 PPDF 位置位。通过该标志位用户可以执行 STOP2 恢复服务程序。

当 I/O 配置为通用 I/O 时，在进入 STOP2 模式之前，将 I/O 寄存器的状态保存到 RAM 中，在 MCU 退出 STOP2 模式后，用户软件从 RAM 中恢复保存的状态。如果在写 PPDACK 之前未恢复的话，那么 I/O 引脚转变为复位状态。

3.6.2 STOP3 模式

在执行 STOP 命令之前，设置 PPDC=0，STOPE=1，那么执行 STOP 指令，MCU 进入 STOP3 模式。一旦进入 STOP3 模式，MCU 内的所有时钟包括振荡器均停止。ICG 处于待命状态。内部寄存器和逻辑电路以及 RAM 中的数据均维持。I/O 引脚状态不被闩锁。

复位引脚上的复位信号，异步中断或实时中断可以唤醒 STOP3 模式下的 MCU。

复位引脚上的复位信号唤醒 MCU 的话，MCU 按照复位状态运行。而通过异步中断或实时中断唤醒的话，MCU 执行相应的中断复位程序。

一个单独的自时钟源（大约 1kHz）为实时中断提供时钟。当 RTIS2:RTIS1:RTIS0=0:0:0，实时中断和 1kHz 时钟源均关闭，从而进一步降低功耗。

3.6.3 STOP 模式下的后台调试模块

在 CPU 执行 STOP 指令之前，设置 BDCSCR 寄存器中的 ENBDM=1，那么在 MCU 进入停止模式之后，MCU 的后台调试通信仍能进行，而且 MCU 内部电压稳压模块不会进入低功耗模式，而是处于正常工作状态。因此当 ENBDM 置位，用户试图进入 STOP2 模式时，MCU 将进入 STOP3 模式。但在该模式下，大多数的后台调试命令不能使用。带状态返回存储器访问命令不会访问存储器，但会返回一个错误标志，表明 MCU 处于停止模式或等待模式。BACKGROUND 命令可使 MCU 从停止模式进入后台调试模式，一旦进入后台调试模式，所以的后台调试命令均可使用。

Mode	PPDC	CPU, Digital Peripherals, FLASH	RAM	ICG	ADC	Regulator	I/O Pins	RTI
Stop3	0	Standby	Standby	Active	Optionally on	Active	States held	Optionally on

表 3.2 BDM 使能时 STOP 模式下的运行情况

3.6.4 STOP 模式下的 LVD

当供电电源电压跌落到门限电压，LVD 模块会产生中断或复位。CPU 在执行 STOP 指令之前，设置 SPMSCQ 寄存器中的 LVDE=1，LVDSE=1，那么 LVD 在 STOP 模式下处于工作状态。用户在 LVD 使能的前提下，试图进入 STOP2 将被阻止，MCU 会进入 STOP3 模式。

Mode	PPDC	CPU, Digital Peripherals, FLASH	RAM	ICG	ADC	Regulator	I/O Pins	RTI
Stop3	0	Standby	Standby	Off	Optionally on	Active	States held	Optionally on

表 3.3 LVD 使能时 STOP 模式下的运行情况

3.6.5 STOP 模式下的外围模块

无论何种 STOP 模式，内部外围模块的时钟都会停止。在停止模式下，如果 ENBDM=1，那么后台调试逻辑电路时钟仍运行。

I/O 引脚

- 当 MCU 进入 STOP3 模式，所有的 I/O 状态保持。
- 如何 MCU 配置进入 STOP2 模式，那么在进入 STOP 模式之前，所有 I/O 引脚被闩锁。

MEMORY

- 在 STOP3 模式下，所有 RAM 和寄存器内容都保留
- 在任何停止模式下，FLASH 存储器的数据不会丢失
- 当 MCU 从 STOP2 模式唤醒，所有寄存器处于复位状态。RAM 中的内容不会丢失，引脚仍处于闩锁状态（直到 PPDACK 位被写）。

ICG

在 STOP3 模式，ICG 进入低功耗模式。通过设置 OSCSTEN 位，振荡器可保持运行。在 STOP2 模式，ICG 关闭。

TPM

当 MCU 进入 STOP 模式，TPM1 和 TPM2 的时钟停止。如何 MCU 从 STOP2 模式唤醒，TPM 模块将复位，必须进行初始化。

ADC

当 MCU 进入 STOP 模式，ADC 进入低功耗待命状态（如何 ADACK 使能，ADC 会仍运行）。如果 MCU 从 STOP2 模式唤醒，那么用户程序必须对 ADC 模块进行初始化。

KBI

在 STOP3 模式下，KBI 功能可以用于唤醒 MCU。而在 STOP2 模式下，KBI 关闭。从 STOP2 唤醒后，应对 KBI 重新初始化。

SCI

当 MCU 进入 STOP 模式，SCI1 和 SCI2 的时钟停止。当 MCU 从 STOP2 模式唤醒，用户必须对 SCI 进行初始化。

SPI

同 SCI

IIC

同 SCI

电源稳压器

如果 LVD 未使能，那么电源稳压器在 STOP 模式下处于低功耗待命状态。

第四章 存储器

4.1 MC9S08SH8 存储映象

寄存器分 3 个区域：

- 低页寄存器（0x0000～0x007F）
- 高页寄存器（0x1800～0x185F）
- 非易失寄存器（$FFB0～0xFFBF）

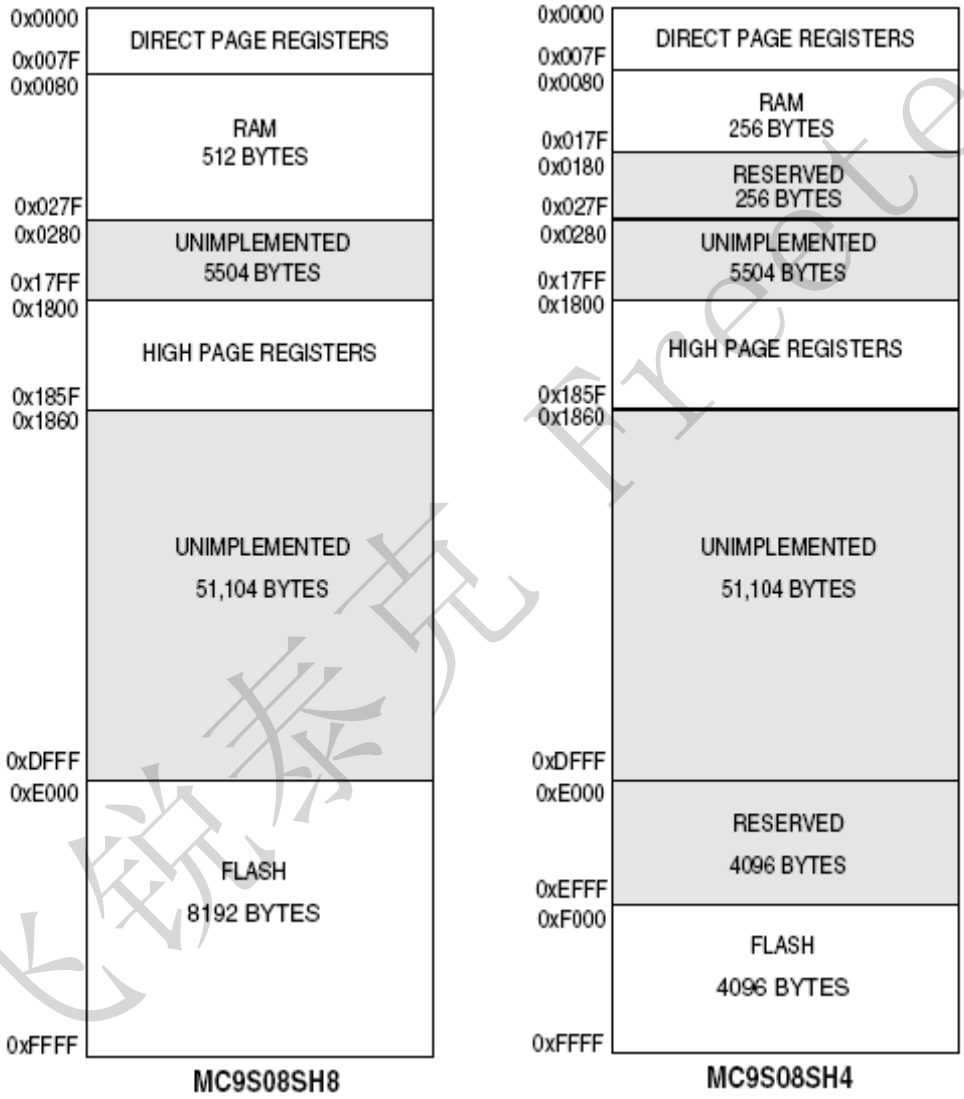


图 4.1MC9S08SH8/4 存储器映射

4.2 复位，中断向量表

Address (High/Low)	Vector	Vector Name
0xFFC0:0xFFC1	Reserved	—
0xFFC2:0xFFC3	ACMP	Vacmp
0xFFC4:0xFFC5	Reserved	—
0xFFC6:0xFFC7	Reserved	—
0xFFC8:0xFFC9	Reserved	—
0xFFCA:0xFFCB	MTIM Overflow	Vmtim
0xFFCC:0xFFCD	RTC	Vrtc
0xFFCE:0xFFCF	IIC	Viic
0xFFD0:0xFFD1	ADC Conversion	Vadc
0xFFD2:0xFFD3	Reserved	—
0xFFD4:0xFFD5	Port B Pin Interrupt	Vportb
0xFFD6:0xFFD7	Port A Pin Interrupt	Vporta
0xFFD8:0xFFD9	Reserved	—
0xFFDA:0xFFDB	SCI Transmit	Vscitx
0xFFDC:0xFFDD	SCI Receive	Vscirx
0xFFDE:0xFFDF	SCI Error	Vscierr
0xFFE0:0xFFE1	SPI	Vspi
0xFFE2:0xFFE3	TPM2 Overflow	Vtpm2ovf
0xFFE4:0xFFE5	TPM2 Channel 1	Vtpm2ch1
0xFFE6:0xFFE7	TPM2 Channel 0	Vtpm2ch0
0xFFE8:0xFFE9	TPM1 Overflow	Vtpm1ovf
0xFFEA:0xFFEB	Reserved	—
0xFFEC:0xFFED	Reserved	—
0xFFEE:0xFFEF	Reserved	—
0xFFF0:0xFFF1	Reserved	—
0xFFF2:0xFFF3	TPM1 Channel 1	Vtpm1ch1
0xFFF4:0xFFF5	TPM1 Channel 0	Vtpm1ch0
0xFFF6:0xFFF7	Reserved	—
0xFFF8:0xFFF9	Low Voltage Detect	Vlvd
0xFFFA:0xFFFB	IRQ	Virq
0xFFFC:0xFFFD	SWI	Vswi
0xFFFE:0xFFFF	Reset	Vreset

表 4.1 复位，中断向量表

4.3 寄存器地址和位分配

寄存器分为 3 组：

- 低页（直接页）寄存器：位于存储器的头 128 个位置，可以直接寻址访问，效率高
- 高页寄存器：不经常访问，位于 0x1800 开始的位置，存放变量和寄存器

- 非易失性寄存器：位于 FLASH 的 0XFFB0~0XFFBF 位置，包括：
  －NVPROT 和 NVOPT：MCU 复位后，它们被装载到相应的工作寄存器中去。
  －8 字节后门比较密匙：允许用户获得访问加密存储器的控制权

非易失性寄存器位于 FLASH 区域，所以它们必须被擦除编程。

直接页（包括低页和高页）寄存器可以被高效的直接寻址模式指令访。位操作指令可以访问直接页寄存器中的任何位。直接页寄存器之所以可以高效访问，是因为直接寻址模式只要求提供地址的低 8 位。

假如 KEYEN=1，那么 8 字节比较密匙可以用于临时取消存储器加密保护。这种密匙机制只能在用户程序运行过程中运用。通过编程可以设置 KEYEN=0，临时解除加密功能被禁止，唯一取消加密的方法是整块擦除 FLASH 并查空。同时为避免下次复位进入加密模式，可以编程 SEC01:SEC00＝1：0。

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	PTAD	0	0	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	PTADD	0	0	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
0x0002	PTBD	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0003	PTBDD	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0004	PTCD	0	0	0	0	PTCD3	PTCD2	PTCD1	PTCD0
0x0005	PTCDD	0	0	0	0	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x0006–0x000D	Reserved	—	—	—	—	—	—	—	—
0x000E	ACMPSC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0
0x000F	Reserved	—	—	—	—	—	—	—	—
0x0010	ADSC1	COCO	AIEN	ADCO	ADCH				
0x0011	ADSC2	ADACT	ADTRG	ACFE	ACFGT	—	—	—	—
0x0012	ADRH	0	0	0	0	0	0	ADR9	ADR8
0x0013	ADRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
0x0014	ADCVH	0	0	0	0	0	0	ADCV9	ADCV8
0x0015	ADCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
0x0016	ADCFG	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0017	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x0018	APCTL2	0	0	0	0	ADPC11	ADPC10	ADPC9	ADPC8
0x0019	Reserved	—	—	—	—	—	—	—	—
0x001A	IRQSC	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x001B	Reserved	—	—	—	—	—	—	—	—
0x001C	MTIMSC	TOF	TOIE	TRST	TSTP	0	0	0	0
0x001D	MTIMCLK	0	0	CLKS		PS			
0x001E	MTIMCNT	CNT							
0x001F	MTIMMOD	MOD							
0x0020	TPM1SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0021	TPM1CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0022	TPM1CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0023	TPM1MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0024	TPM1MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0025	TPM1C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0026	TPM1C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0027	TPM1C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0028	TPM1C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0029	TPM1C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x002A	TPM1C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x002B–0x0037	Reserved	—	—	—	—	—	—	—	—

表 4.2 直接寄存器表（3-1）

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0038	SCIBDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0039	SCIBDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x003A	SCIC1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x003B	SCIC2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x003C	SCIS1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x003D	SCIS2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x003E	SCIC3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x003F	SCID	Bit 7	6	5	4	3	2	1	Bit 0
0x0040–0x0047	Reserved	—	—	—	—	—	—	—	—
0x0048	ICSC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x0049	ICSC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
0x004A	ICSTRIM	TRIM							
0x004B	ICSSC	0	0	0	IREFST	CLKST		OSCINIT	FTRIM
0x004C–0x004F	Reserved	—	—	—	—	—	—	—	—
0x0050	SPIC1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0051	SPIC2	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x0052	SPIBR	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
0x0053	SPIS	SPRF	0	SPTEF	MODF	0	0	0	0
0x0054	Reserved	0	0	0	0	0	0	0	0
0x0055	SPID	Bit 7	6	5	4	3	2	1	Bit 0
0x0056–0x0057	Reserved	—	—	—	—	—	—	—	—
0x0058	IICA	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
0x0059	IICF	MULT		ICR					
0x005A	IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
0x005B	IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x005C	IICD	DATA							
0x005D	IICC2	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
0x005E–0x005F	Reserved	—	—	—	—	—	—	—	—
0x0060	TPM2SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0061	TPM2CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0062	TPM2CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0063	TPM2MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0064	TPM2MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0065	TPM2C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0066	TPM2C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0067	TPM2C0VL	Bit 7	6	5	4	3	2	1	Bit 0

表 4.3 直接寄存器表（3-2）

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0068	TPM2C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0069	TPM2C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x006A	TPM2C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x006B	Reserved	—	—	—	—	—	—	—	—
0x006C	RTCSC	RTIF	RTCLKS		RTIE	RTCPS			
0x006D	RTCCNT	RTCCNT							
0x006E	RTCMOD	RTCMOD							
0x006F - 0x007F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —

表 4.4 直接寄存器表（3-3）

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
0x1801	SBDFR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	COPT		STOPE	0	0	IICPS	BKGDPE	RSTPE
0x1803	SOPT2	COPCLKS	COPW	0	ACIC	0	0	T1CH1PS	T1CH0PS
0x1804 – 0x1805	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x1806	SDIDH	0	—	—	—	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	Reserved	—	—	—	—	—	—	—	—
0x1809	SPMSC1	LVWF	LVWACK	LVWIE	LVDRE	LVDSE	LVDE	0	BGBE
0x180A	SPMSC2	0	0	LVDV	LVWV	PPDF	PPDACK	—	PPDC
0x180B– 0x180F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x1810	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1811	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0
0x1812	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1813	DBGCBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1814	DBGFH	Bit 15	14	13	12	11	10	9	Bit 8
0x1815	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0
0x1816	DBGC	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
0x1817	DBGT	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
0x1818	DBGS	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
0x1819– 0x181F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x1820	FCDIV	DIVLD	PRDIV8	DIV					
0x1821	FOPT	KEYEN	FNORED	0	0	0	0	SEC	
0x1822	Reserved	—	—	—	—	—	—	—	—
0x1823	FCNFG	0	0	KEYACC	0	0	0	0	0
0x1824	FPROT	FPS							FPDIS
0x1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x1826	FCMD	FCMD							
0x1827– 0x183F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x1840	PTAPE	0	0	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x1841	PTASE	0	0	—	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x1842	PTADS	0	0	—	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x1843	Reserved	—	—	—	—	—	—	—	—
0x1844	PTASC	0	0	0	0	PTAIF	PTAACK	PTAIE	PTAMOD

表 4.5 高页寄存器（2-1）

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1845	PTAPS	0	0	0	PTAPS4	PTAPS3	PTAPS2	PTAPS1	PTAPS0
0x1846	PTAES	0	0	0	PTAES4	PTAES3	PTAES2	PTAES1	PTAES0
0x1847	Reserved	—	—	—	—	—	—	—	—
0x1848	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1849	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x184A	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x184B	Reserved	—	—	—	—	—	—	—	—
0x184C	PTBSC	0	0	0	0	PTBIF	PTBACK	PTBIE	PTBMOD
0x184D	PTBPS	0	0	0	0	PTBPS3	PTBPS2	PTBPS1	PTBPS0
0x184E	PTBES	0	0	0	0	PTBES3	PTBES2	PTBES1	PTBES0
0x184F	Reserved	—	—	—	—	—	—	—	—
0x1850	PTCPE	0	0	0	0	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x1851	PTCSE	0	0	0	0	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x1852	PTCDS	0	0	0	0	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x1853	GNGC	GNGPS7	GNGPS6	GNGPS5	GNGPS4	GNGPS3	GNGPS2	GNGPS1	GNGEN
0x1854–0x185F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —

表 4.6 高页寄存器（2-2）

注：复位之后，NVPROT 和 NVOPT 寄存器的值转移到相应的 FPROT 和 FOPT 工作寄存器，用力控制加密和保护功能。

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0xFFAE	Reserved for storage of FTRIM	0	0	0	0	0	0	0	FTRIM
0xFFAF	Reserved for storage of ICSTRIM	TRIM							
0xFFB0–0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8–0xFFBC	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0xFFBD	NVPROT	FPS							FPDIS
0xFFBE	Reserved	—	—	—	—	—	—	—	—
0xFFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC	

表 4.7 非易失性寄存器

4.3 RAM

位于 0X100 以下的 RAM 区域可以被高效的直接寻址模式访问，且该区域可以位寻址。用于存放经常使用的变量。

在 WAIT，STOP2，STOP3 模式下，RAM 数据保持。上电复位后 RAM 中的数据未初始化。只要电源电压不低于 RAM 维持电压值，RAM 的数据不会受任何复位的影响。

为了兼容 M68HC05 系列，HCS08 复位后，堆栈指针指向 0X00FF。通常初始化堆栈指针，指向 RAM 的顶，这样低页 RAM 完全可以用于变量和位寻址变量。可执行下面的语句：

LDHX #RamLast + 1 TXS

当加密使能，RAM 是一个加密的存储区域，不同通过后台调试或无未加密的存储区代码访问。

4.4 FLASH

通过 BDM 调试接口，可以进行在线编程。

4.4.1 特征

- FLASH 大小：
  －MC9S08SH8：8192 字节（每页 512 字节，共 16 页）
  －MC9S08SH4：4096 字节（每页 512 字节，共 8 页）
- 单电源擦除编程
- 快速擦除编程命令
- 在典型电压和温度下，多达 100，000 次编程擦除次数
- 灵活的块保护
- 低频读取时自动降低功耗

4.4.2 编程和擦除时间

任何编程或擦除命令执行之前，FLASH 时钟分频寄存器（FCDIV）必须将内部时钟设置设置好，即 f_{FCLK} 在 150kHz~200kHz 之间。这个寄存器只能写入一次，所以在复位初始化过程中设置好该寄存器。如果在 FSTAT 寄存器中的 FACCERR＝1，FCDIV 不允许写入。故在写入 FCDIV 之前，应查询 FACCERR 标志。执行时间如下：

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 µs
Byte program (burst)	4	20 µs[1]
Page erase	4000	20 ms
Mass erase	20,000	100 ms

[1] Excluding start/end overhead

表 4.9 编程和擦除时间表

4.4.3 编程擦除命令执行过程

在执行下面操作之前，应清除所有的错误标志，并设置好 FCDIV。

执行步骤：

1. 向 FLASH 阵列的一个地址写入一个数据。这次写的地址和数据信息被闩锁进 FLASH 接口。该预写步骤是任何命令序列的第一个步。
   对于擦除和查空命令，数据值并不重要。
   对于页擦除命令，地址应为要擦除的 512 字节 FLASH 块中任意一个位置地址。
   对于块擦除名，地址应为 FLASH 存储器地址空间中任意一个位置地址。
   对于字节编程，那么该地址为要编程的地址，数据为要编程的数据。
   注意：正确擦除操作之后，只允许编程一次。对未擦除的地址进行编程会破坏该地址中的数据。
2. FCMD 写入命令。命令如下：
   $05:查空
   $20:字节编程

$25:快速编程

$40:页擦除

$41:块擦除

3. FSTAT 寄存器中 FCBEF 位写 1，启动命令（包括地址和数据信息）。

在第 1 步和第 3 步骤之间，可以手动的设置 FCBEF=0，可中止该次操作。在开始新的命令之前，必须清除 FACCERR 访问错误标志。

必须遵守这一严格过程，否则命令不会被接受。FCCF 位表明命令是否执行完成。

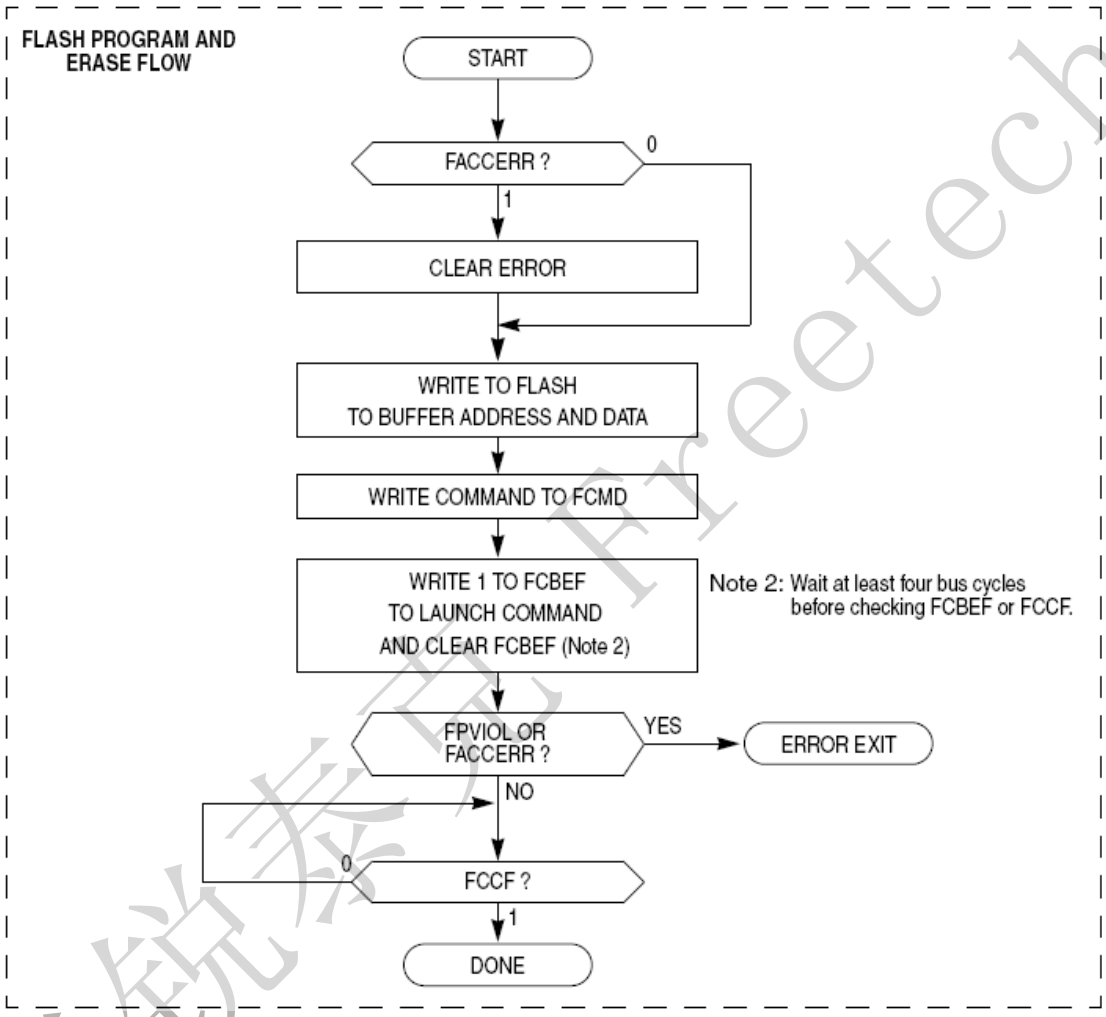


图4.2 FLASH擦除和编程流程图

4.4.4 快速编程过程

快速编程用于编程连续几个字节的数据，时间上少于通过标准编程命令方式。这是由于在两次编程操作之间，不必关闭高压。通常，当发出一个擦除或编程命令，内部电压泵被打开为FLASH阵列提供高压。一旦命令执行完成，电压泵关闭。当发出一个快速编程命令，电压泵使能并保持（在一次快速编程完成之后，遇到以下情形）

- 在当前编程操作完成之前，下一个快速编程命令已经列队好
- 下一个连续地址与当前编程地址为同一行。一行FLASH包括64个字节。一行内的地址通过A0~A5来定位。当A0~A5全为0时，开始新的一行。

快速编程模式下，第一个字节编程时间与标准编程模式时间相同。接下的字节编程时

间为快速编程时间，前提是满足上面两个条件之一。

当连续字节的地址为新的一行的第一个字节，其编程时间与标准编程时间相同，这是因为，高压必须关闭，给另一个FLASH阵列供电。在当前命令完成之前，没有新的快速编程命令，电荷泵将关闭，高压移除该阵列。

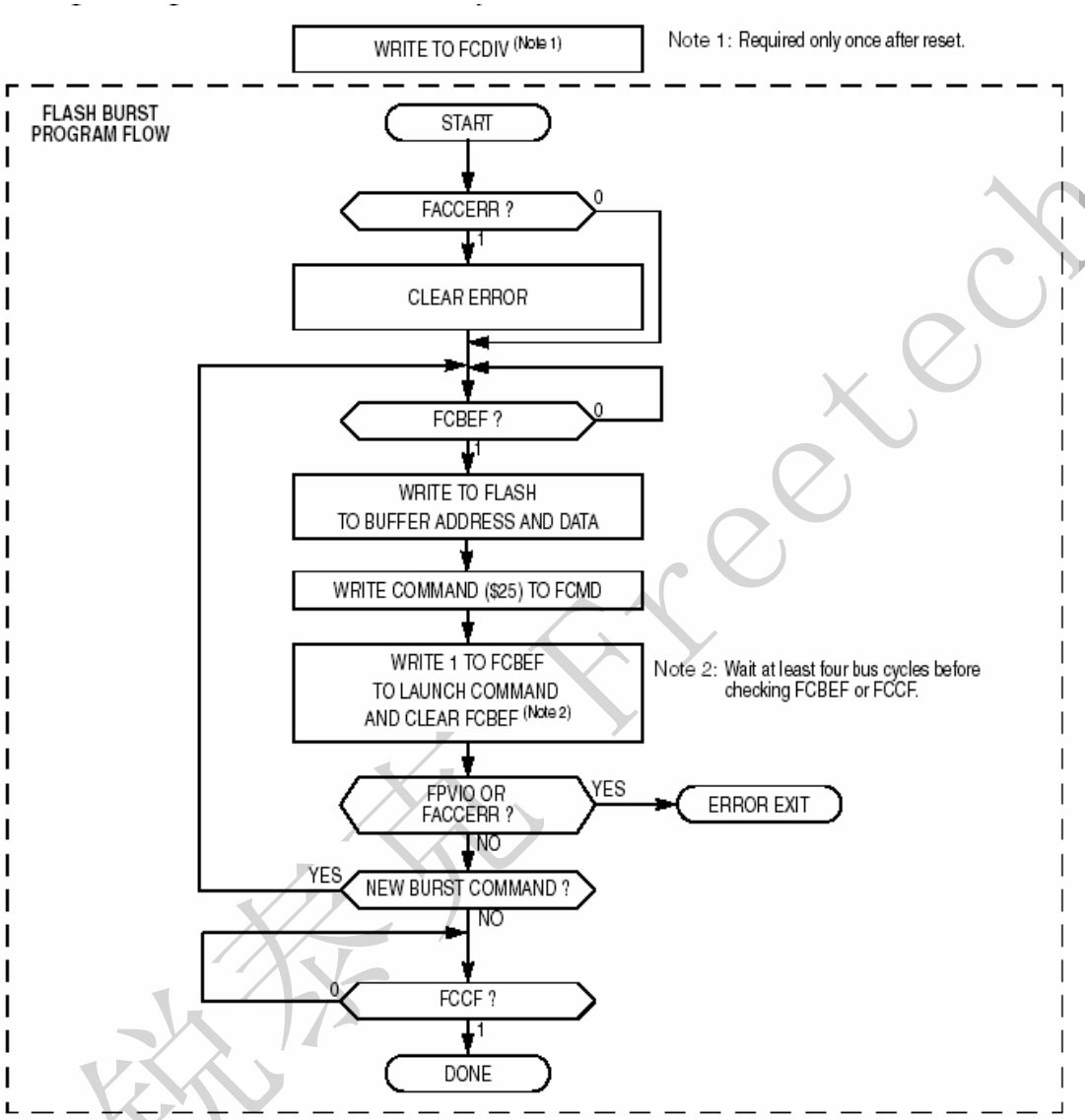


图4.3 FLASH快速编程流程图

4.4.5 错误访问

当违反命令执行协议时，就会产生一个访问错误。即FACCERR=1。在开始任何一个命令之前，FACCERR必须清除，清除方法是向FACCERR写1。

- 在FLASH时钟频率未设置之前向一个FLASH地址写操作
- 未向FLASH地址写操作之前，FCBEF设置为1
- 在发起前一个命令之前，再次向一个FLASH地址写数据（对于每个命令执行之前只能进行一次对FLASH的写操作）
- 在发起前一个命令之前，再次写FCMD（对于每个命令操作，只能对FCMD写一次）
- 在向一个FLASH地址写数据之后，除了FCMD，向任何其他FLASH控制寄存器写

操作

- 除了$05,$20,$25,$40,$41，向FCMD写入其他任何命令代码操作
- 在向FCMD写入命令之后，除了写FSTAT寄存器，访问任何FLASH控制寄存器
- 擦除或编程进行中，MCU进入STOP模式
- MCU处于加密状态，后台调试命令进行字节编程，快速编程或页擦除命令（后台调试命令只能进行查空和块擦除命令，当MCU处于保密状态）
- 向FCBEF写0，取消命令操作

4.4.6 FLASH 块保护

块保护是指被保护的FLASH区域可以避免被编程或擦除。块保护是通过FLASH保护寄存器（FPROT）控制的。当块保护使能，块保护将保护FLASH地址0XFFFF以下任何以512个字节为单位的块。

复位之后，NVPROT内的值装载到FPROT中。FPROT不能被应用软件直接改变，所以用户程序不能改变块保护设置。因为NVPROT位于FLASH的最后512个字节，如果任何数量的存储器被保护，那么NVPROT自身也受保护，不允许修改。

后台调试命令可以修改FPORT，所以后台调试命令可以擦除和再次编程受保护的FLASH存储器。

例如为了保护0XE000~0XFFFF存储器，应设置FPS＝0XDF。所以0XDFFF表示从0X0000开始未保护区域。同时为了能够修改FPS值，首先应该设置FPDIS=0。

用户可以对存放引导程序的区域进行块保护，这样引导程序就可以对其他FLASH空间擦除和编程。由于引导程序受到保护，所以在擦除和编程期间，即使电源掉电，也不会受到破坏。

4.4.7 向量重指

当任意块受到保护，那么复位和中断向量也将受到保护。向量重指允许用户程序调整中断向量信息（不包括引导程序区域和复位向量）。设置NVOPT寄存器中的FNORED=0，允许向量重指。为了能够重指，至少一部分FLASH空间受到保护。所有中断向量（0XFFC0~0XFFD）都被重指，而复位向量不改变。

例如，如果FLASH的512字节即（0XFE00~0XFFFF）受到保护。那么中断向量（0XFFC0~0XFFFD）被重指到0XFDC0~0XFDFD。如果现在SPI中断发生，那么0XFDE0:FDE1处的中断向量被使用。利用这一特点，用户可以对未保护的区域进行编程（包括新的中断向量）。

4.5 加密

MC9S08AC16可以阻止对FLASH和RAM的未授权访问。当加密机制使能，FLASH和RAM认为是加密的区域。而低页，高页寄存器和后台调试控制器被认为是未加密区域。在加密的存储器执行的程序可以访问任何MCU存储器和资源。反之，在非安全区域的程序或后台调试接口试图访问一个加密区域的话，将被阻止（写操作忽略，读操作返回0值）。

加密机制使能或禁止是由FOPT寄存器中的非易失性位SEC01:SEC00位决定。复位期间，NVOPT中的值装载到FOPT中。因此在程序烧写到存储器时，可以设置好NVOPT寄存器。1：0的组合取消加密机制，其他3种组合均使能加密机制。故擦除之后，组合（1：1）会使MCU

加密。在开发过程中，当FLASH被擦除，编程SEC01:SEC00=1:0，从而MCU处于未加密状态，方便程序的开发。

当MCU处于加密状态时，片上调试模块不能使能。但独立的后台调试控制器仍用于后台存储器访问命令，但MCU不能进入后台调试模式（除非在复位的上升沿，BKGD/MS维持低）。

通过后门密匙，可以临时解密。如果NVOPT/FOPT中的KEYEN位为0，则后门密匙不允许，除了完全擦除FLASH外，没有其他方法取消加密。如果KEYEN=1，一个加密的用户程序可以临时取消加密状态，方法如下：

1. 向FCNFG寄存器中的KEYACC写1。从而对FLASH的NVBACKKEY~NVBACKKEY+7位置的写操作是比较后门密匙，而不是作为FLASH编程或擦除命令的第一步。
2. 向NVBACKKEY～NVBACKKEY+7位置写入用户输入的密匙值。写过程必须按一定顺序进行，开始于NVBACKKEY，结束于NVBACKKEY+7。STHX指令不应用于该写过程，因为对NVBACKKEY～NVBACKKEY+7写操作不能在接近总线周期完成。用户软件通常从通过通信接口从外部系统获得密匙。
3. 向FCNFG寄存器的KEYACC位写0。如果刚写入的8字节密匙与存储在FLASH的密匙相同，SEC01:SEC00自动变为1：0，直到下次复位，那么MCU的加密状态取消。

加密密匙只能由加密存储区域（RAM或FLASH）的程序修改，因此不能通过后台调试命令输入。

后门比较密匙位于FLASH空间，与中断和复位向量同属一个512字节块。

通过后台调试接口执行以下步骤可以取消加密机制：

1. 写FPROT位取消任何块保护。FPROT只能通过后台调试命令修改，而不能通过应用软件。
2. 块擦除FLASH
3. 查空。

为避免下次复位之后返回安全状态，应编程SEC01:SEC00=1:0。

4.6 FLASH寄存器和控制位

4.6.1 FLASH 时钟分频寄存器（FCDIV）

	7	6	5	4	3	2	1	0
R	DIVLD	PRDIV8	DIV5	DIV4	DIV3	DIV2	DIV1	DIV0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

7 DIVLD	0:FCDIV 寄存器在复位之后未被写过；擦除和编程操作禁止； 1：FCDIV 寄存器复位之后被写过；可以对 FLASH 进行擦除和编程
6 PRDIV8	预分频系数 8 允许位 0：总线时钟直接输入 FLASH 时钟驱动器 1：总线时钟 8 分频后再输入 FLASH 时钟驱动器
5 DIV[5:0]	分频系数：FLASH 时钟有总线时钟驱动。见下表

if PRDIV8 = 0 — $f_{FCLK} = f_{Bus} \div ([DIV5:DIV0] + 1)$ *Eqn. 4-1*

if PRDIV8 = 1 — $f_{FCLK} = f_{Bus} \div (8 \times ([DIV5:DIV0] + 1))$ *Eqn. 4-2*

f_{Bus}	PRDIV8 (Binary)	DIV5:DIV0 (Decimal)	f_{FCLK}	Program/Erase Timing Pulse (5 μs Min, 6.7 μs Max)
20 MHz	1	12	192.3 kHz	5.2 μs
10 MHz	0	49	200 kHz	5 μs
8 MHz	0	39	200 kHz	5 μs
4 MHz	0	19	200 kHz	5 μs
2 MHz	0	9	200 kHz	5 μs
1 MHz	0	4	200 kHz	5 μs
200 kHz	0	0	200 kHz	5 μs
150 kHz	0	0	150 kHz	6.7 μs

表 4.10 flash 时钟分频器设置

4.6.2 FLASH 选择寄存器（FOPT 和 NVOPT）

	7	6	5	4	3	2	1	0
R	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
W								
Reset	This register is loaded from nonvolatile location NVOPT during reset.							

= Unimplemented or Reserved

7 KEYEN	后门密匙机制允许位 0：后门密匙访问禁止；1：通过输入密匙值并与 NVBACKKEY－NVBACKEY+7 匹配，加密机制被屏蔽直到下次复位
6 FNORED	向量重指允许位 0：向量重指允许；1：禁止向量重指
1：0 SEC0[1:0]	安全状态代码，见下表

SEC01:SEC00	Description
0:0	secure
0:1[1]	secure
1:0	unsecured
1:1	secure

表 11 加密状态

4.6.3 FLASH 配置寄存器（FCNFG）

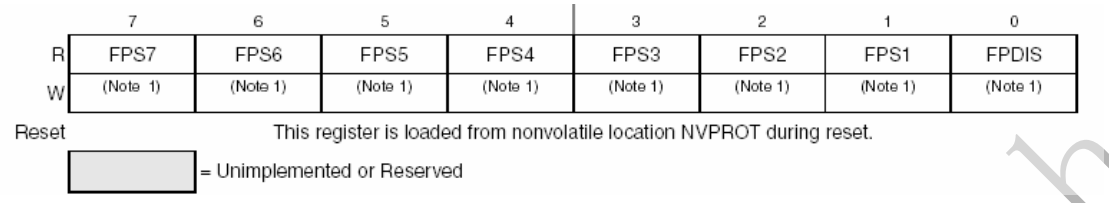
	7	6	5	4	3	2	1	0
R	0	0	KEYACC	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

5	访问密匙允许位

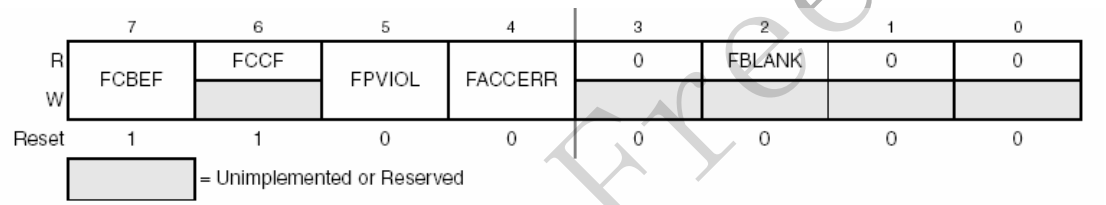
KEYACC	0：写 0Xffb0～0Xffb7 操作被认为是擦除或编程 FLASH 的开始 1：写 0Xffb0～0Xffb7 操作被认为是比较密匙

4.6.4 FLASH 保护寄存器（FPROT 和 NVPROT）



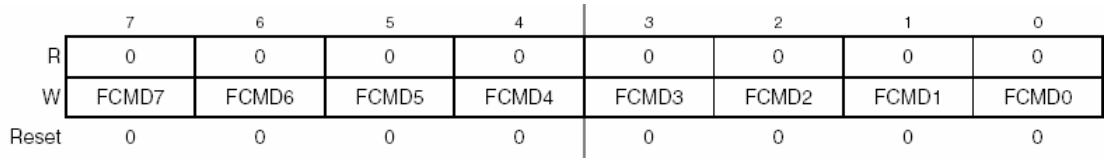
7：1 FPS[7:1]	FLASH 保护选择位－当 FPDIS=0，该 7 位决定未被保护区域的截至地址
0 FPDIS	FLASH 保护允许位 0：FLASH 保护机制使能；1：禁止保护 FLASH

4.6.5 FLASH 状态寄存器（FSTAT）



7 FCBEF	FLASH 命令缓冲区空标志 0：命令缓冲区满；1：一个新的快速编程命令可以写入命令缓冲区
6 FCCF	FLASH 命令完成标志 0：命令执行中；1：所有命令执行完成
5 FPVIOL	FLASH 保护冲突标志 0：无保护冲突；1：试图擦除或编程一个受保护的区域，发生冲突
4 FACCERR	FLASH 访问错误标志 0：无访问错误发生 1：发生一个访问错误
2 FBLANK	FLASH 查空核实标志 0：查空命令之后，FCCF=1，FBLANK=0 表明 FLASH 阵列未完全擦除； 1：查空命令之后，FCCF=1, FBLANK=1 表明 FLASH 阵列完成擦除

4.6.6 FLASH 命令寄存器（FCMD）



Command	FCMD	Equate File Label
Blank check	$05	mBlank
Byte program	$20	mByteProg
Byte program — burst mode	$25	mBurstProg
Page erase (512 bytes/page)	$40	mPageErase
Mass erase (all FLASH)	$41	mMassErase

表 4.12 FLASH 操作命令

第五章 复位，中断，系统配置

5.1 介绍

本章讨论MCU的复位和中断机制以及各种复位和中断源。

5.2 特征

- 多种复位源，系统配置灵活，可靠
  - －上电检测
  - －低电压检测
  - －复位引脚上施加复位信号
  - －看门狗复位
  - －非法操作码
  - －后台调试复位
- 复位状态寄存器(SRS)记录最近一次发生的复位源
- 每个片上外围功能模块有独立的中断向量

5.3 MCU复位

复位MCU操作为用户提供一种预知的设置状态。复位期间，大多数控制和状态寄存器被强制复位到初始值，复位向量装载到程序计数器中。片上外围模块禁止，I/O初始化为通用的高阻抗输入，且内部上拉取消。中断禁止。用户程序初始化程序可以初始化堆栈指针SP和系统控制配置。

注意：SP复位之后为0X00FF。

7个复位源：

- 上电复位（POR）
- 低电压复位（LVD）
- 看门狗复位（COP）
- 非法操作码复位
- 后台调试强制复位
- 复位引脚
- 时钟发生器锁定失败和时钟丢失复位

除了后台调试强制复位，其他每个复位源与复位状态寄存器相关联。MCU进入复位状态，内部时钟发生器(ICG)模块打开，时钟频率为f_{self_reset}，复位引脚拉低，持续34个内部总线时钟周期，然后该引脚被释放，内部上拉。再经过38个周期，采样该引脚，决定是否是该引脚引发的复位。

5.4 看门狗（COP）

当用户程序“跑飞”，看门狗产生一个强制系统复位。为了阻止看门狗复位，用户程序必须定期的复位COP计数器。

复位之后，COP使能。如果想关闭COP功能，只要清除SOPT1中的COPT位。

向SRS寄存器写0X0055或0X00AA，COP计数器会复位。但该写操作并不会影响只读寄

存器SRS内容。如果向SRS写入其他之，那么MCU会立即复位。

SOPT2寄存器中的COPCLKS位用于选择COP的时钟源。

Control Bits		Clock Source	COP Window[1] Opens (COPW = 1)	COP Overflow Count
COPCLKS	COPT[1:0]			
N/A	0:0	N/A	N/A	COP is disabled
0	0:1	1 kHz	N/A	2^5 cycles (32 ms[2])
0	1:0	1 kHz	N/A	2^8 cycles (256 ms[1])
0	1:1	1 kHz	N/A	2^{10} cycles (1.024 s[1])
1	0:1	Bus	6144 cycles	2^{13} cycles
1	1:0	Bus	49,152 cycles	2^{16} cycles
1	1:1	Bus	196,608 cycles	2^{18} cycles

表 5.1 看门狗时钟源配置表

当选择总线时钟作为 COP 的时钟时，用户应在所设定的 COP 时间间隔的后 25%的时间内对 SRS 写操作，复位 COP，如果提前对 SRS 写操作将复位 MCU。如果选择的 1-kHz 的时钟源，那么不会出现上面的问题。

用户程序可以使用COPW，COPCLKS和COPT的复位默认值，因为这两个寄存器只一次写有效，建议用户程序复位初始化过程中写一次SOPT1和SOPT2寄存器，可以避免在用户程序运行过程中意外修改它们。

写SRS寄存器操作（清看门狗操作）不能放在中断服务程序（ISR）中，这是因为尽管主程序已经“跑飞”，但ISR仍会定期执行，从而失去看门狗监测的功能。

当MCU处于后台调试模式下，COP计数器不会递增。

当MCU处于处于STOP模式，看门狗时钟源选择的是总线时钟的情况下，COP计数器会停止计数。一旦MCU脱离后台调试模式，COP计数器继续递增式计数。

当MCU处于STOP模式，看门狗时钟源选择的是1－kHz的情况下，COP计数器被初始化为0。一旦MCU脱离STOP模式，COP计数器从0开始递增式计数。

5.5 中断

中断是指保存当前CPU状态和寄存器，执行中断服务程序，中断服务程序执行完毕后恢复CPU状态，MCU从被中断打断的地方继续执行。IRQ引脚的边沿事件或计数器溢出事件等产生硬件中断。用户应用程序执行特定指令可以产生软件中断（SWI），后台调试模块在一定条件下也可以产生SWI。

当某一中断事件发生，与该事件相对应的只读标志将被置位。如果该中断未使能，那么CPU不会对该中断作出反映。CCR寄存器中的全局中断屏蔽位I在复位之后为1，屏蔽掉所有中断事件。在用户程序完成堆栈指针和其他设置之后，清除I，允许CPU响应中断事件。

当CPU接受到一个有效的中断请求，在响应之前，它会执行完当前运行的指令，然后逐周期执行以下过程：

- 保存 CPU 寄存器到堆栈中
- 设置 CCR 寄存器中的 I=1，禁止全局中断
- 获取当前所有挂起的待处理中断事件中最高优先级中断向量
- 预取中断向量的地址排列到指令队列中

当CPU响应中断后，I自动置为1，避免中断嵌套。通常，ISR执行完毕后，I恢复为0（CCR从堆栈弹出）。如果在ISR中，清除I（即设置I=0），那么CPU有可能在当前中断事件未完

成之前响应其他中断事件，即中断潜套。中断潜套可能对用户程序调试带来麻烦。

当ISR执行到RTI指令，CCR，A，X，PC从堆栈恢复。

注意：为兼容MC68HC08，H寄存器不会自动压栈和出栈。用户程序完成对H的压栈出栈。

5.5.1 中断堆栈结构

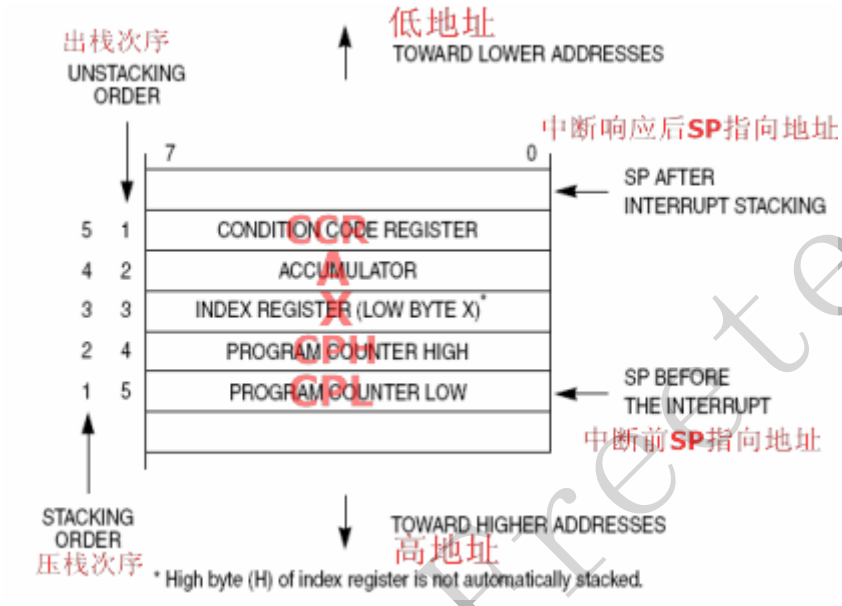


图 5.1 中断压栈

在中断服务程序的开始，用户程序应清掉产生该中断的标志位，其目的是同一中断源产生另外一个中断事件能够被记录，并在完成当前中断服务程序之后能够得到CPU的响应。

5.5.2 外部中断引脚

外部中断由IRQ的状态和控制寄存器IRQSC管理。当IRQ使能，同步逻辑电路监测该引脚是否边沿事件或边沿－电平事件发生。IRQ可以用于唤醒STOP模式下的MCU。

5.5.2.1 引脚配置选项

设置IRQSC寄存器中的IRQPE＝1，那么该引脚作为IRQ输入功能。

IRQMOD寄存器决定引脚中断模式：沿触发中断或边沿－电平触发中断。

IRQEDG寄存器决定引脚中断信号边沿或电平的极性。

当IRQ引脚作为外部中断输入，根据选择的极性内部上拉或下拉；当用户打算在外部对IRQ引脚进行上拉或下拉，那么IRQPDD位应设置为1，关闭内部的上拉或下拉。

BIH和BIL指令可用于检测引脚信号电平。

5.5.2.2 边沿和电平触发模式

IRQMOD控制位设置外部中断输入信号边沿有效还是电平有效。

在边沿－电平有效模式下，当检测到有效的边沿信号，IRQF标志就会置位，如果该引脚一直处于有效的触发电平状态，那么该中断标志会连续被置位。

5.5.3 中断向量，中断源和中断屏蔽

下表列出所有中断向量和中断源以及中断优先级。

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
Lowest	31	0xFFC0/0xFFC1	—	—	—	—	—
	30	0xFFC2/0xFFC3	Vacmp	ACMP	ACF	ACIE	Analog comparator
	29	0xFFC4/0xFFC5	—	—	—	—	—
	28	0xFFC6/0xFFC7	—	—	—	—	—
	27	0xFFC8/0xFFC9	—	—	—	—	—
	26	0xFFCA/0xFFCB	Vmtim	MTIM	TOF	TOIE	MTIM overflow
	25	0xFFCC/0xFFCD	Vrtc	RTC	RTIF	RTIE	Real-time interrupt
	24	0xFFCE/0xFFCF	Viic	IIC	IICIF	IICIE	IIC control
	23	0xFFD0/0xFFD1	Vadc	ADC	COCO	AIEN	ADC
	22	0xFFD2/0xFFD3	—	—	—	—	—
	21	0xFFD4/0xFFD5	Vportb	Port B	PTBIF	PTBIE	Port B Pins
	20	0xFFD6/0xFFD7	Vporta	Port A	PTAIF	PTAIE	Port A Pins
	19	0xFFD8/0xFFD9	—	—	—	—	—
	18	0xFFDA/0xFFDB	Vscitx	SCI	TDRE, TC	TIE, TCIE	SCI transmit
	17	0xFFDC/0xFFDD	Vscirx	SCI	IDLE, RDRF, LBKDIE, RXEDGIE	ILIE, RIE, LDBKDIF, RXEDGE	SCI receive
	16	0xFFDE/0xFFDF	Vscierr	SCI	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI error
	15	0xFFE0/0xFFE1	Vspi	SPI	SPIF, MODF, SPTEF	SPIE, SPIE, SPTIE	SPI
	14	0xFFE2/0xFFE3	Vtpm2ovf	TPM2	TOF	TOIE	TPM2 overflow
	13	0xFFE4/0xFFE5	Vtpm2ch1	TPM2	CH1F	CH1IE	TPM2 channel 1
	12	0xFFE6/0xFFE7	Vtpm2ch0	TPM2	CH0F	CH0IE	TPM2 channel 0
	11	0xFFE8/0xFFE9	Vtpm1ovf	TPM1	TOF	TOIE	TPM1 overflow
	10	0xFFEA/0xFFEB	—	—	—	—	—
	9	0xFFEC/0xFFED	—	—	—	—	—
	8	0xFFEE/0xFFEF	—	—	—	—	—
	7	0xFFF0/0xFFF1	—	—	—	—	—
	6	0xFFF2/0xFFF3	Vtpm1ch1	TPM1	CH1F	CH1IE	TPM1 channel 1
	5	0xFFF4/0xFFF5	Vtpm1ch0	TPM1	CH0F	CH0IE	TPM1 channel 0
	4	0xFFF6/0xFFF7	—	—	—	—	—
	3	0xFFF8/0xFFF9	Vlvd	System control	LVWF	LVWIE	Low-voltage warning
	2	0xFFFA/0xFFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin
	1	0xFFFC/0xFFFD	Vswi	Core	SWI Instruction	—	Software interrupt
Highest	0	0xFFFE/0xFFFF	Vreset	System control	COP, LVD, $\overline{\text{RESET}}$ pin, Illegal opcode, Illegal address	COPT LVDRE — — —	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address

表 5.2 中断向量列表

5.6 低电压检测系统

低电压检测系统用于保护处于低电压条件下的存储器中数据和在电压波动条件下控制 MCU 系统状态。该系统包括上电复位（POR）和低电压检测(LVD)电路。

SPMSC1 寄存器中的 LVDE＝1，LVD 使能。

SPMSC2 寄存器中的 LVDV 位选择跌落电压。

当 MCU 处于 STOP 模式下，LVDSE 未设置，那么 LVD 功能关闭。

当 LVDSE 和 LVDE 位都被置位，那么 MCU 不能进入 STOP2 模式。

5.6.1 上电复位

上电或电源跌落到 V_{POR} 以下，POR 电路将产生复位条件。当电源电压上升，LVD 电路使 MCU 处于复位状态，直到电源电压大于 V_{LVDL} 。所以上电复位之后，SRS 寄存器中的 POR 和 LVD 为均为 1。

5.6.2 LVD 复位

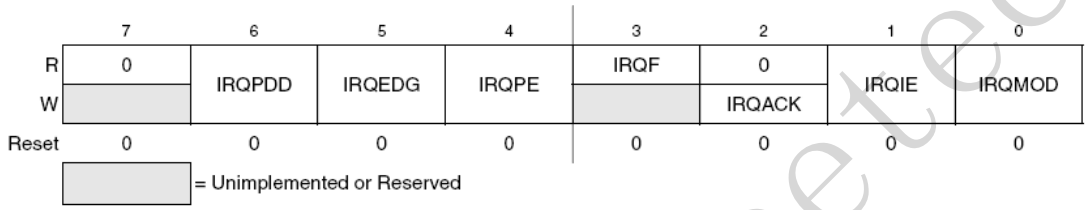
如果 LVDRE=1，那么 LVD 电路检测到低电压情况下将产生一个复位。直到电源电压大于 V_{LVDV} ，MCU 一直处于复位状态。LVD 复位或 POR 后，SRS 的 LVD 位均被置位。

5.6.3 LVD 中断

设置 LVDE=1，LVDIE=1，LVDRE=0，LVD 电路检测到低电压产生一个中断，LVDF 被置为 1。

5.7 复位，中断和系统控制寄存器和控制位

5.7.1 外部中断引脚状态和控制寄存器（IRQSC）



6 IRQPDD	IRQ 引脚内部上拉控制位：可读写 0：IRQ 引脚内部上拉使能 1：IRQ 引脚内部上拉取消
5 IRQEDG	IRQ 边沿选择：可读写； 0：下降沿或下降沿/低电平有效 1：上升沿或上升沿/高电平有效
4 IRQPE	IRQ 使能：可读写 0：关闭 IRQ 功能 1：IRQ 使能
3 IRQF	IRQ 中断标志 0：无 IRQ 请求 1：有 IRQ 请求
2 IRQACK	IRQ 确认：只写 写 1 清除 IRQF，但在边沿－电平触发模式下，如果 IRQ 引脚处于中断有效电平状态，向 IRQACK 写 1 不会清除 IRQF
1 IRQIE	IRQ 中断使能：可读写 0：关闭 IRQ 中断功能 1：IRQ 中断使能
0 IRQMOD	IRQ 触发方式：可读写 0：IRQ 边沿触发方式 1：上升沿/高电平或下降沿/低电平触发方式

5.7.2 系统复位状态寄存器（SRS）

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	0	ICG	LVD	0
W	Writing any value to SIMRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVR:	u	0	0	0	0	0	1	0
Any other reset:	0	Note(1)	Note(1)	Note(1)	0	Note(1)	0	0

u = Unaffected by reset

SRS 寄存器

注 1：MCU 复位时，任何有效的或未激活复位源将导致其对应的标志位置 1，无效的或未激活的其他复位源标志将被清除。

7 POR	上电复位 0：不是 POR 引起复位 1：POR 导致复位
6 PIN	外部复位引脚 0：不是外部复位引脚引起复位 1：外部复位引脚导致复位
5 COP	看门狗复位 0：不是看门狗引起复位 1：看门狗引起复位
4 ILOP	非法操作码复位 0：不是非法操作码引起复位 1：非法操作码导致复位
3 ILAD	非法地址 0：不是非法地址引起中断 1：非法地址导致中断
2 ICG	内部时钟发生器模块复位 0：不是内部时钟发生器引起复位 1：内部时钟发生器导致复位
1 LVD	低电压检测复位 0：不是低电压检测模块或上电复位引起复位 1：低电压检测或上电导致复位

5.7.3 系统后台调试强制复位寄存器（SBDFR）

该寄存器只包含一个只写位。像 WRITE_BYTE 后台调试指令必须用于写 SBDFR 寄存器。

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								BDFR[1]
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

0 BDFR	后台调试强制复位：通过后台调试命令写 1，强制 MCU 复位。 该位不能被用户程序写。

5.7.4 系统选项寄存器（SOPT1）

	7	6	5	4[1]	3	2	1	0
R	COPT		STOPE		0	IICPS	BKGDPE	RSTPE
W								
Reset:	1	1	0	0	0	0	1	u[2]
POR:	1	1	0	0	0	0	1	0
LVR:	1	1	0	0	0	0	1	u

= Unimplemented or Reserved

该寄存器在任何时候均可以读取。复位之后，只允许写一次。用户在复位之后初始化过程中，应写一次该寄存器，尽管写入的值与该寄存器复位之后相同。

7:6 COPT	看门狗溢出时间
5 STOPE	STOP 模式使能 0：禁止进入 STOP 模式；1：允许进入 STOP 模式
2 IICPS	IIC 引脚选择 0：SDA 为 PTA2 引脚，SCL 为 PTA3 引脚 1：SDA 为 PTB6 引脚，SCL 为 PTB7 引脚
1 BKGDPE	BKGD/MS 引脚是否用于后台调试。复位之后，该引脚默认为 BKGD/MS 0：不用于 BKGD 功能；1：用于 BKGD 功能
0 RSTPE	PTB2/RESET 引脚是否用于外部复位功能。POR 后，该引脚默认为只输入功能。 0：用作只输入功能；1：用作复位输入

5.7.5 系统选项寄存器 2（SOPT2）

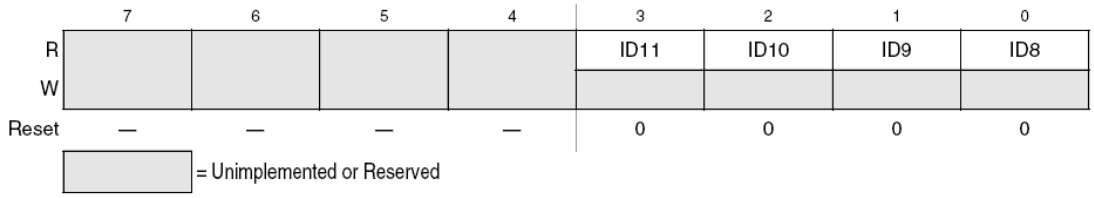
	7	6	5	4	3	2	1	0
R	COPCLKS[1]	COPW[1]	0	ACIC	0	0	T1CH1PS	T1CH0PS
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

7 COPCLKS	COP 时钟选择位 0：内部 1-kHz 时钟源 1：总线时钟
6 COPW	COP 窗口控制 0：正常操作 1：COP 窗口控制使能
4 ACIC	模拟比较器输出到输入捕捉 0：ACMP 输出不连接到 TPM1 的通道 0 1：ACMP 输出连接到 TPM1 的通道 0
1 T1CH1PS	TPM1CH1 引脚选择 0：TPM1CH1 连接到 PTB5 1：TPM1CH1 连接到 PTC1
0 T1CH0PS	TPM1CH0 引脚选择 0：TPM1CH0 连接到 PTA0

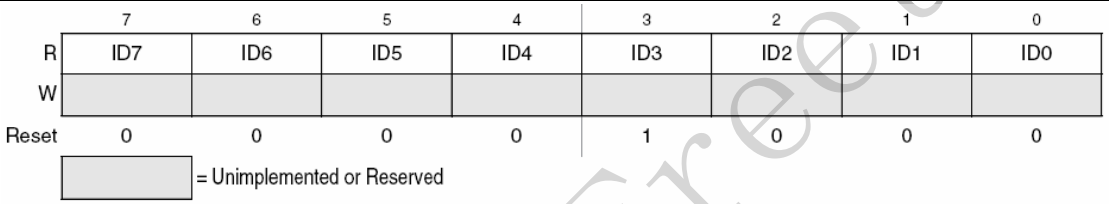
	1：TPM1CH0 连接到 PTC0

5.7.5 设备 ID 寄存器（SDIDH，SDIDL）



SDIDH

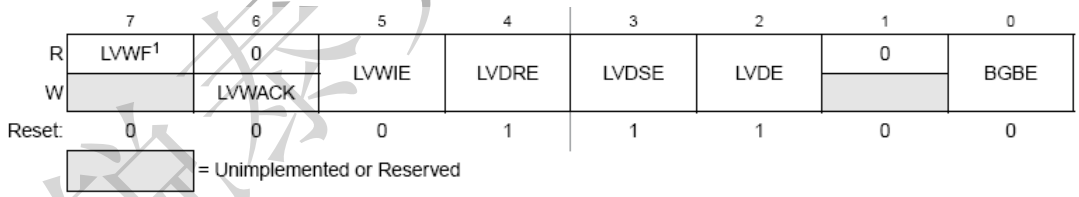
7：4 保留	保留位
3：0 ID[11:8]	设备 ID 高 4 位，均为 0



SDIDL

7：0 ID[7:0]	设备 ID，固定值 0X08

5.7.7 系统电源管理状态和控制寄存器 1（SPMSC1）



7 LVDF	低电压检测标志，只读 1：表明低电压事件发生
6 LVDACK	低电压检测确认，只写 写 1 清除 LVDF
5 LVDIE	低电压检测中断使能 0：中断禁止；1：中断使能
4 LVDRE	低电压检测复位使能 0：LVDF 不产生硬件复位；1：LVDF=1 时产生强制复位
3 LVDSE	低电压检测是否在停止模式可用 0：STOP 模式下 LVD 禁止；1：STOP 模式下 LVD 可运行
2 LVDE	LVD 使能 0：LVD 关闭；1：LVD 使能

0 BGBE	能系缓冲使能：ADC 模块的每一通道上释放使能内部带隙电压缓冲 0：关闭能系参考电压；1：使能能系参考电压

5.7.8 系统电源管理状态和控制寄存器 2（SPMSC2）

	7	6	5	4	3	2	1	0
R	0	0	LVDV[1]	LVWV	PPDF	0	0	PPDC[2]
W						PPDACK		
Power-on Reset:	0	0	0	0	0	0	0	0
LVD Reset:	0	0	u	u	0	0	0	0
Any other Reset:	0	0	u	u	0	0	0	0

= Unimplemented or Reserved　　u = Unaffected by reset

5 LVDV	低电压检测电压选择 0：VLVD= V_{LVDL} 1：VLVD= V_{LVDH}
4 LVWV	低电压报警电压选择 0：VLVW= V_{LVWL} 1：VLVW= V_{LVWH}
3 PDF	全功耗降低标志，只读 0：MCU 从 STOP1 模式唤醒；1：MCU 不是从 STOP1 模式唤醒
2 PPDF	该位表明 MCU 是从 STOP2 模式恢复 0：MCU 不是从 STOP2 模式唤醒 1：MCU 从 STOP2 模式唤醒
2 PPDACK	清除 PPDF 位 写 1 清除 PPDF
0 PPDC	该位用于选择 STOP2 或 STOP3 模式 0：STOP3 模式允许 1：STOP2 模式允许

LVDV:LVWV	LVW Trip Point	LVD Trip Point
0:0	V_{LVW0} = 2.74 V	V_{LVD0} = 2.56 V
0:1	V_{LVW1} = 2.92 V	
1:0	V_{LVW2} = 4.3 V	V_{LVD1} = 4.0 V
1:1	V_{LVW3} = 4.6 V	

[1] See Electrical Characteristics appendix for minimum and maximum values.

第六章 并行输入/输出

MC9S08SH8 有 3 个并行 I/O，即 17 个 IO 和一个只输出。很多外围模块与 IO 复用，外围模块的优先级要高于通用 IO。复位之后，复用的外围模块功能禁止，IO 配置为输入。

6.1 端口数据和数据方向

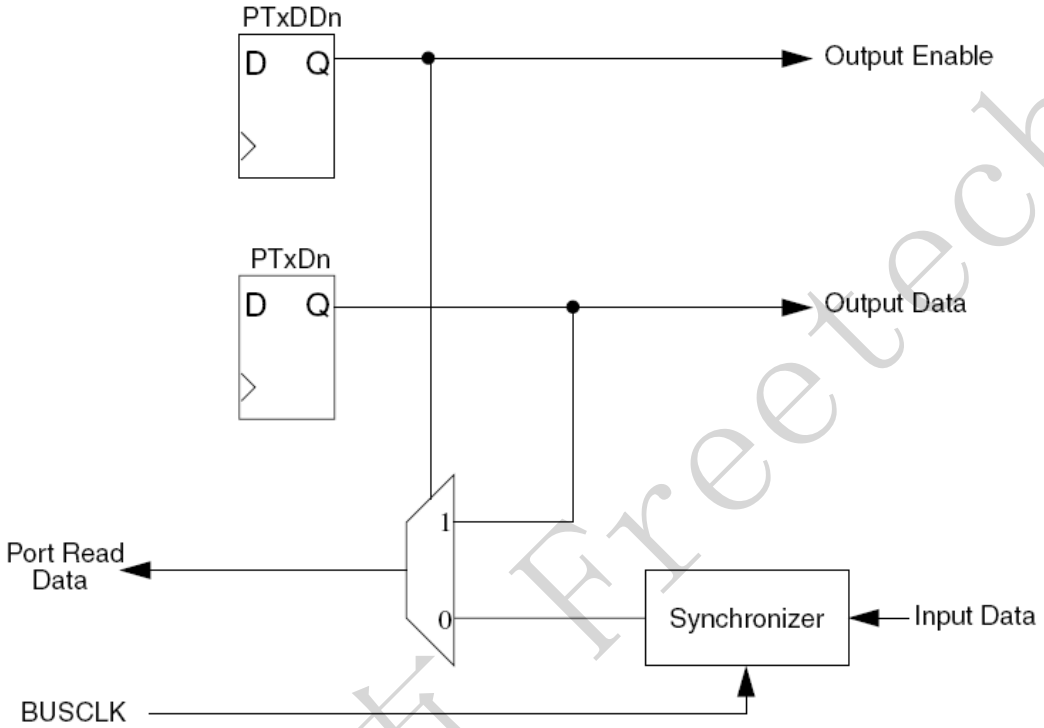


图 6.1 I/O 框图

方向控制位（PTxDDn）决定与引脚相关的输出缓冲器是否使能，同时控制端口数据寄存器读取源。每一个引脚都有一个方向控制位。当 PTxDDn=0，相应的引脚作为输入引脚，读取 PTxD 将返回引脚值。当 PTxDDn=1，相应的引脚作为输出引脚，读取 PTxD 将返回最后一次写入 PTxD 的值。

当复用的数字功能模块使能，那么输出缓冲由复用模块控制。然而，数据方向寄存器仍可以控制端口数据寄存器的读取源。

当引脚复用为模拟功能引脚，那么该引脚的输入和输出缓冲关闭。如果该引脚的数字功能和模拟功能复用，模拟功能优先级较高，当模拟功能和数字功能同时使能时，模拟功能将控制引脚。

在改变端口方向由输入到输出之前，先修改端口数据寄存器的值是一个好的编程习惯。从而避免旧值错误的驱动引脚。

6.2 上拉，压摆率，驱动强度

6.2.1 内部上拉使能

当该引脚作为通用输入功能，PTxPEn 位控制上拉使能。当该引脚作为模拟功能，外围模块功能或通用输出引脚，那么上拉禁止。

6.2.2 压摆率控制使能位

每一个引脚都有一个相应位控制压摆率。当该引脚压摆率使能，那么该引脚的输出

压摆率将被限制，从而降低 EMC 对外干扰。当然，当该引脚作为输入引脚时，压摆率控制不再起作用。

6.2.3 输出驱动强度选择

PTxDSn 位控制引脚驱动强度，当 PTxDSn=1，相应的引脚可以吸收和提供更多电流。

6.3 绑定输出

MC9S08SH8 允许 8 个端口引脚绑定一起输出。绑定输出驱动控制寄存器（GNGC）为只一次写入寄存器。该寄存器控制使能绑定输出，选择哪些端口引脚作为绑定输出。GNGEN 位使能绑定输出。GNGPS[7:1]位控制哪些引脚绑定输出。

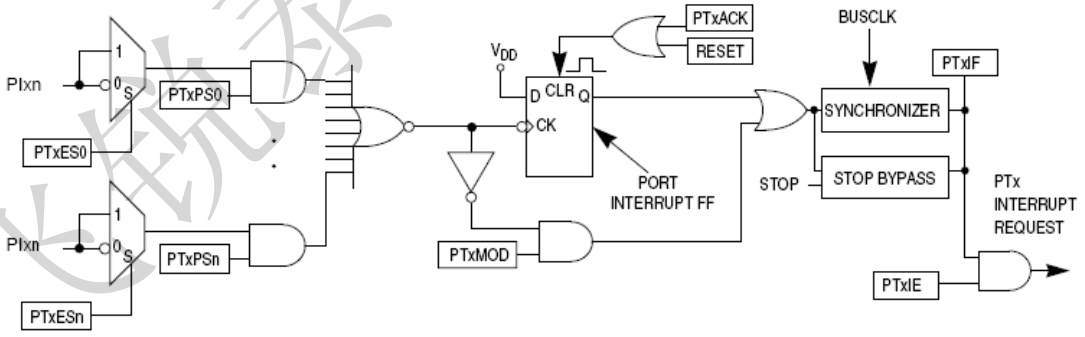
当 GNGEN 位置位，绑定输出的引脚行为与 PTC0 一致。

	GNGC Register Bits							
	GNGPS7	**GNGPS6**	**GNGPS5**	**GNGPS4**	**GNGPS3**	**GNGPS2**	**GNGPS1**	**GNGEN[1]**
Port Pin [2]	PTB5	PTB4	PTB3	PTB2	PTC3	PTC2	PTC1	PTC0
Data Direction Control	Pin is automatically configured as output when pin is enabled as ganged output.							
Data Control	PTCD0 in PTCD controls data value of output							
Drive Strength Control	PTCDS0 in PTCDS controls drive stength of output							
Slew Rate Control	PTCSE0 in PTCSE controls slew rate of output							

注 1：8 引脚封装的芯片没有绑定输出。

2：当 GNGEN=1，PTC0 强制为输出。

6.4 引脚中断



引脚中断框图

端口中断使能寄存器（PTxPS）中的 PTxPSn 位可单独对每个端口引脚中断进行控制。每个端口可配置为边沿敏感或边沿电平敏感方式。

引脚中断初始化过程：

1） 清零 PTxSC 中的 PTxIE 位

2） 设置 PTxES 中的 PTxESn 位，选择引脚边沿极性

3） 如果使用内部上拉或下拉电阻，应设置 PTxPE 中的相应位。

4） 使能中断引脚（PTxPS 中的 PTxPSn 位）

5） 向 PTxSC 中的 PTxACK 写操作清除任何错误中断

6） 设置 PTxSC 中的 PTxIE 位使能中断。

6.5 STOP 模式下引脚工作情况

- Stop2 模式是一种部分降低功耗模式，I/O 锁存器处于维持态。在执行 STOP 指令之前，CPU 寄存器和 I/O 寄存器必须保存到 RAM 中。从 STOP2 模式唤醒后，访问任何 I/O 之前，用户必须检查在 SPMSC2 寄存器的 PPDF 位的状态。如果 PPDF=0，I/O 必须进行初始化，因为是上电复位唤醒 STOP2 模式下的 CUP。如果 PPDF=1，I/O 数据可以从先前保存到 RAM 的数据恢复，然后向 SPMSC2 寄存器的 PPDACK 位写 1，目的就是用户程序可以访问引脚。
- STOP3 模式下，所有引脚处于维持状态，因此脱离 STOP3 模式后，引脚功能与进入 STOP3 模式前一样。

6.6 并行 T/O 寄存器

6.6.1 端口 A 寄存器

6.6.1.1 端口 A 数据寄存器（PTAD）

	7	6	5	4	3	2	1	0
R	0	0	PTAD5	PTAD4[1]	PTAD3	PTAD2	PTAD1	PTAD0
W								
Reset:	0	0	0	0	0	0	0	0

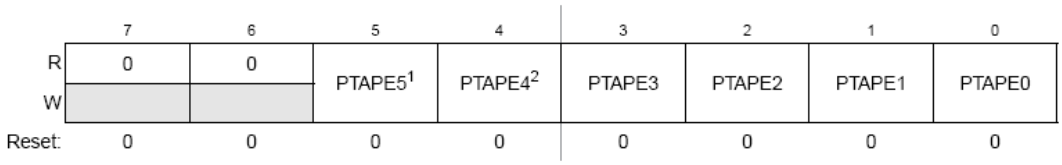
5:0 PTAD[5:0]	端口 A 数据寄存器位：A 为输入端口时，读取该寄存器将返回引脚的逻辑电平值；A 为输出端口时，读取该寄存器将返回上一次写入该寄存器的值。 写入的数据将被闩锁。 复位后，该寄存器各位为 0，高阻输入状态

6.6.1.2 端口 A 方向寄存器（PTADD）

	7	6	5	4	3	2	1	0
R	0	0	PTADD5	PTADD4[1]	PTADD3	PTADD2	PTADD1	PTADD0
W								
Reset:	0	0	0	0	0	0	0	0

5:0 PTADD[5:0]	端口 A 方向寄存器：可读写位。 0：输入；1：输出

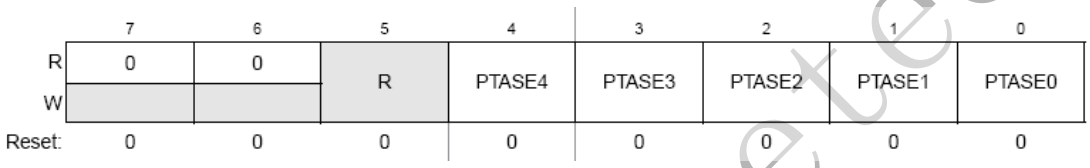
6.6.1.3 内部上拉使能(PTAPE)



	7	6	5	4	3	2	1	0
R	0	0	PTAPE5[1]	PTAPE4[2]	PTAPE3	PTAPE2	PTAPE1	PTAPE0
W								
Reset:	0	0	0	0	0	0	0	0

5：0 PTAPE[5:0]	端口 A 上拉使能位：当端口 A 作为输入引脚时，该寄存器各位控制相应引脚内部上拉是否使能；当端口 A 作为输出时，内部上拉关闭（不管该寄存器各位的状态） 0：内部上拉关闭；1：内部上拉使能

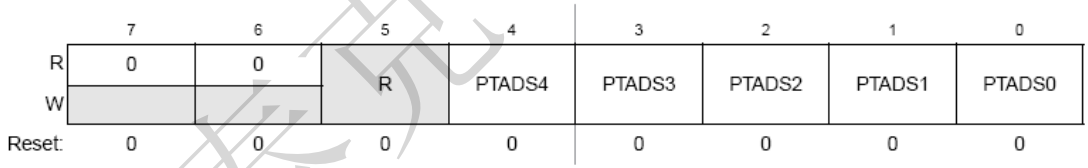
6.6.1.4 输出压摆率控制寄存器(PTASE)



	7	6	5	4	3	2	1	0
R	0	0	R	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
W								
Reset:	0	0	0	0	0	0	0	0

4：0 PTASE[4:0]	端口 A 压摆率控制位：当端口 A 作为输出时，该可读写位控制是否使能压摆率控制；当端口 A 作为输入时，该位不再起作用。 0：压摆率控制关闭；1：压摆率控制使能。

6.6.1.5 输出驱动强度选择（PTADS）

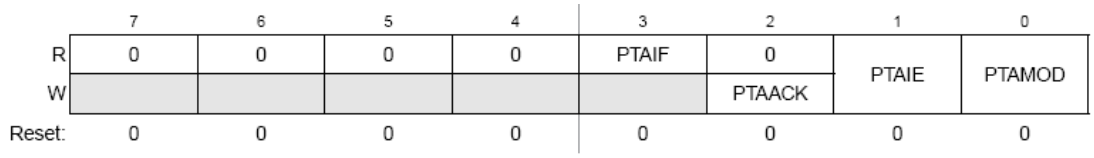


	7	6	5	4	3	2	1	0
R	0	0	R	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
W								
Reset:	0	0	0	0	0	0	0	0

引脚配置为高驱动强度时，引脚能够驱动和吸收更大电流。当用户必须确保芯片总的灌电流和拉电流不能超过芯片的极限。驱动强度选择将影响到引脚的 DC 和 AC 特性。

4：0 PTADS[4:0]	输出驱动强度选择位 0：低驱动能力；1：高驱动能力

6.6.1.6 端口 A 中断状态和控制寄存器（PTASC）



	7	6	5	4	3	2	1	0
R	0	0	0	0	PTAIF	0	PTAIE	PTAMOD
W						PTAACK		
Reset:	0	0	0	0	0	0	0	0

3	端口 A 中断标志：PTAIF 表明端口 A 检测到一个中断事件。

PTAIF	0：无中断事件 1：发生中断事件
2 PTAACK	端口 A 中断确认位：向该位写 1 清除中断。
1 PTAIE	端口 A 中断使能： 0：端口 A 中断禁止 1：端口 A 中断允许
0 PTAMOD	端口 A 检测模式 0：端口 A 只检测边沿 1：端口 A 检测边沿电平

6.6.1.7 端口 A 中断引脚选择寄存器（PTASPS）

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTAPS3	PTAPS2	PTAPS1	PTAPS0
W								
Reset:	0	0	0	0	0	0	0	0

3:0 PTAPS[3:0]	端口 A 中断引脚选择 0：引脚不作为中断 1：引脚作为中断

6.6.1.8 端口 A 中断边沿选择寄存器（PTAES）

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTAES3	PTAES2	PTAES1	PTAES0
W								
Reset:	0	0	0	0	0	0	0	0

3:0 PTAE[3:0]	端口 A 边沿选择 0：上拉使能，下降沿或低电平产生中断 1：下拉使能，上升沿或高电平产生中断

6.6.2 端口 B 寄存器

6.6.2.1 端口 B 数据寄存器（PTBD）

	7	6	5	4	3	2	1	0
R W	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
Reset:	0	0	0	0	0	0	0	0

7:0 PTBD[7:0]	端口 B 数据寄存器位：B 为输入端口时，读取该寄存器将返回引脚的逻辑电平值；B 为输出端口时，读取该寄存器将返回上一次写入该寄存器的值。写入的数据将被闩锁。 复位后，该寄存器各位为 0，高阻输入状态

6.6.2.2 端口 B 方向寄存器（PTBDD）

	7	6	5	4	3	2	1	0
R W	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
Reset:	0	0	0	0	0	0	0	0

7:0 PTBDD[7:0]	端口 B 方向寄存器：可读写位。 0：输入；1：输出

6.6.2.3 内部上拉使能(PTBPE)

	7	6	5	4	3	2	1	0
R W	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
Reset:	0	0	0	0	0	0	0	0

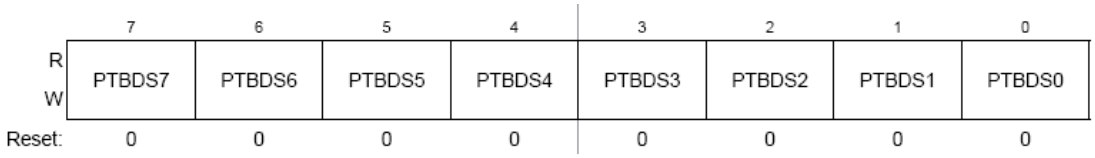
7：0 PTBPE[7:0]	端口 B 上拉使能位：当端口 B 作为输入引脚时，该寄存器各位控制相应引脚内部上拉是否使能；当端口 B 作为输出时，内部上拉关闭（不管该寄存器各位的状态） 0：内部上拉关闭；1：内部上拉使能

6.6.2.4 输出压摆率控制寄存器(PTBSE)

	7	6	5	4	3	2	1	0
R W	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
Reset:	0	0	0	0	0	0	0	0

7：0 PTBSE[7:0]	端口 B 压摆率控制位：当端口 B 作为输出时，该可读写位控制是否使能压摆率控制；当端口 B 作为输入时，该位不再起作用。 0：压摆率控制关闭；1：压摆率控制使能。

6.6.2.5 输出驱动强度选择（PTBDS）

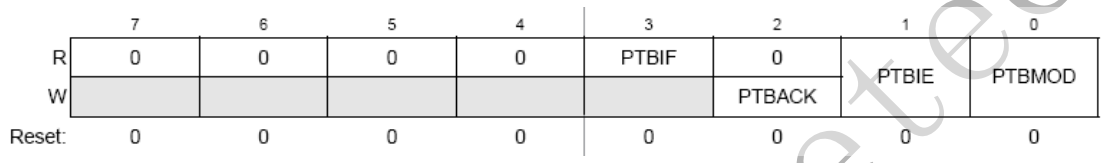


	7	6	5	4	3	2	1	0
R W	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
Reset:	0	0	0	0	0	0	0	0

引脚配置为高驱动强度时，引脚能够驱动和吸收更大电流。当用户必须确保芯片总的灌电流和拉电流不能超过芯片的极限。驱动强度选择将影响到引脚的 DC 和 AC 特性。

7：0 PTBDS[7:0]	输出驱动强度选择位 0：低驱动能力；1：高驱动能力

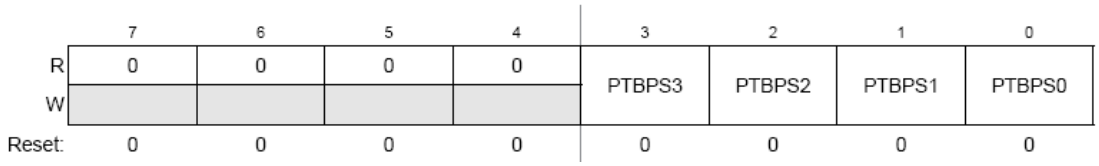
6.6.2.6 端口 B 中断状态和控制寄存器（PTBSC）



	7	6	5	4	3	2	1	0
R	0	0	0	0	PTBIF	0	PTBIE	PTBMOD
W						PTBACK		
Reset:	0	0	0	0	0	0	0	0

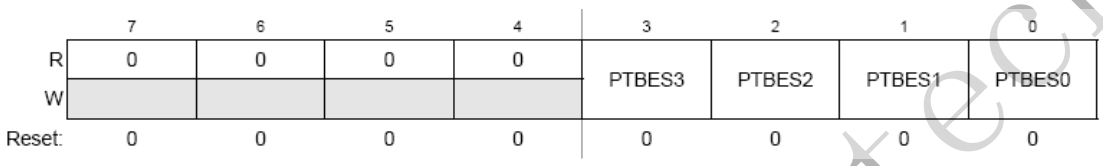
3 PTBIF	端口 B 中断标志：PTBIF 表明端口 B 检测到一个中断事件。 0：无中断事件 1：发生中断事件
2 PTBACK	端口 B 中断确认位：向该位写 1 清除中断。
1 PTBIE	端口 B 中断使能： 0：端口 B 中断禁止 1：端口 B 中断允许
0 PTAMOD	端口 B 检测模式 0：端口 B 只检测边沿 1：端口 B 检测边沿电平

6.6.2.7 端口 B 中断引脚选择寄存器（PTBSPS）



3:0 PTBPS[3:0]	端口 B 中断引脚选择 0：引脚不作为中断 1：引脚作为中断

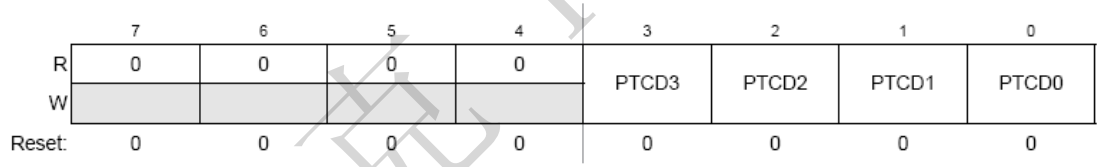
6.6.2.8 端口 B 中断边沿选择寄存器（PTBES）



3:0 PTBE[3:0]	端口 B 边沿选择 0：上拉使能，下降沿或低电平产生中断 1：下拉使能，上升沿或高电平产生中断

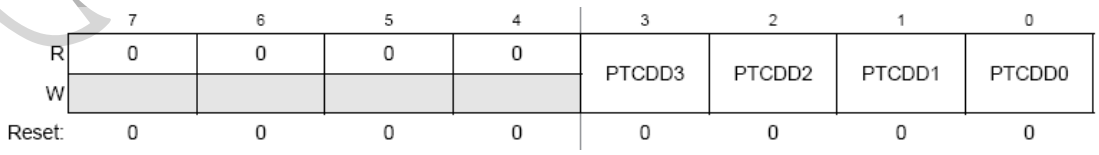
6.6.3 端口 C 寄存器

6.6.3.1 端口 C 数据寄存器（PTCD）



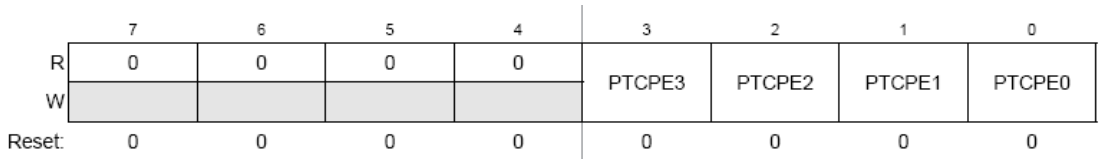
3:0 PTCD[3:0]	端口 C 数据寄存器位：C 为输入端口时，读取该寄存器将返回引脚的逻辑电平值；C 为输出端口时，读取该寄存器将返回上一次写入该寄存器的值。写入的数据将被闩锁。 复位后，该寄存器各位为 0，高阻输入状态

6.6.3.2 端口 C 方向寄存器（PTCDD）



3:0 PTCDD[7:0]	端口 C 方向寄存器：可读写位。 0：输入；1：输出

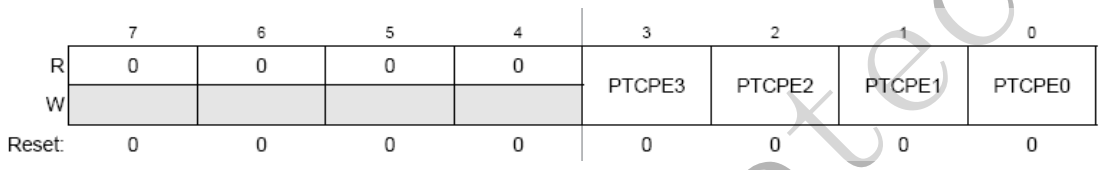
6.6.3.3 内部上拉使能(PTCPE)



	7	6	5	4	3	2	1	0
R	0	0	0	0	PTCPE3	PTCPE2	PTCPE1	PTCPE0
W								
Reset:	0	0	0	0	0	0	0	0

3：0 PTCPE[C:0]	端口 C 上拉使能位：当端口 C 作为输入引脚时，该寄存器各位控制相应引脚内部上拉是否使能；当端口 C 作为输出时，内部上拉关闭（不管该寄存器各位的状态） 0：内部上拉关闭；1：内部上拉使能

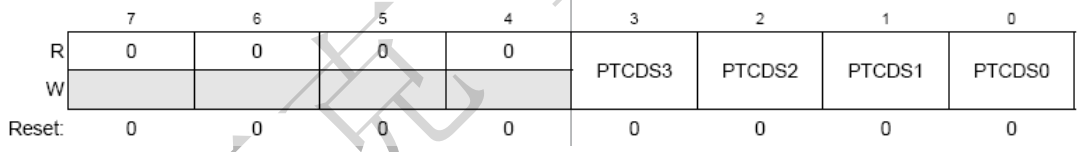
6.6.3.4 输出压摆率控制寄存器(PTCSE)



	7	6	5	4	3	2	1	0
R	0	0	0	0	PTCPE3	PTCPE2	PTCPE1	PTCPE0
W								
Reset:	0	0	0	0	0	0	0	0

C：0 PTCSE[3:0]	端口 C 压摆率控制位：当端口 C 作为输出时，该可读写位控制是否使能压摆率控制；当端口 C 作为输入时，该位不再起作用。 0：压摆率控制关闭；1：压摆率控制使能。

6.6.3.5 输出驱动强度选择（PTCDS）

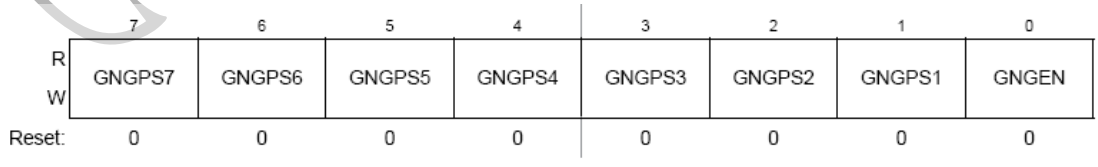


	7	6	5	4	3	2	1	0
R	0	0	0	0	PTCDS3	PTCDS2	PTCDS1	PTCDS0
W								
Reset:	0	0	0	0	0	0	0	0

引脚配置为高驱动强度时，引脚能够驱动和吸收更大电流。当用户必须确保芯片总的灌电流和拉电流不能超过芯片的极限。驱动强度选择将影响到引脚的 DC 和 AC 特性。

3：0 PTCDS[3:0]	输出驱动强度选择位 0：低驱动能力；1：高驱动能力

6.6.3.6 绑定输出驱动控制寄存器（GNGC）



	7	6	5	4	3	2	1	0
R	GNGPS7	GNGPS6	GNGPS5	GNGPS4	GNGPS3	GNGPS2	GNGPS1	GNGEN
W								
Reset:	0	0	0	0	0	0	0	0

7：1 GNGP[7:1]	绑定输出引脚选择位 0：对应的引脚不作为绑定输出 1：对应的引脚作为绑定输出
0 GNGEN	绑定输出驱动使能控制位 0：绑定输出驱动禁止 1：绑定输出驱动使能

第七章 CPU

7.1 简介

7.1.1 特点

- 向上兼容 M68HC05 和 M68HC08
- 所有寄存器和存储器物理地址映射到单一的 64K 字节空间
- 16 位堆栈指针
- 16 位索引寄存器(H:X)
- 8 位累加器 A
- 许多指令把 X 作为第二个通用寄存器
- 七种寻址模式

  －隐含寻址模式：操作数位于内部寄存器

  －相对寻址模式：8 位有符号偏移量用于转移到目的分支

  －立即寻址模式：操作数紧接在操作码后面

  －直接寻址模式：操作数位于存储器的 0X0000~0X00FF 空间

  －扩展寻址模式：操作数位于存储器的 64K 空间内

  －变址寻址方式：包括自动递增的 5 种子模式

  －堆栈寻址模式：提高 C 语言效率
- 存储器到存储器寻址：4 种寻址方式
- 溢出，半进位，负数，零和进位条件码
- 高效的位操作指令
- 快速 8 位乘 8 位指令和 16 位除以 8 位指令
- STOP 和 WAIT 指令：用于进入低功耗模式

7.2 编程模型和 CPU 寄存器

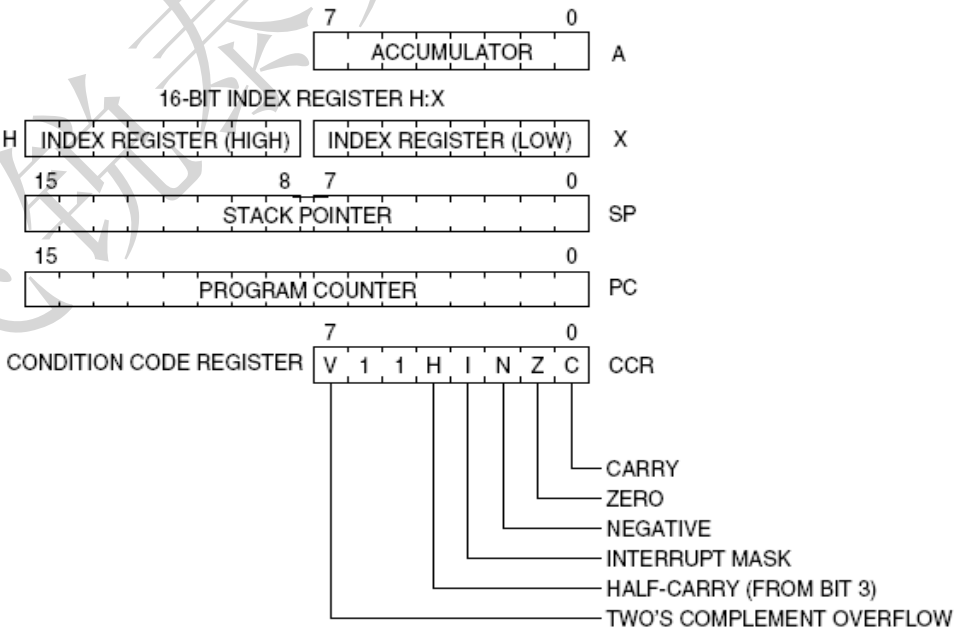


图 7.1 CPU 寄存器

7.2.1 累计器 A

累计器是一个通用的 8 位寄存器。用于存放 ALU 的输入和保存 ALU 的运算结果。通过不同的寻址方式，A 可以与存储器中的数据进行交换。

7.2.2 变址寄存器（H:X）

变址寄存器实际由两个独立的 8 位寄存器（H 和 X）组成的 16 位地址指针。H 为高字节，X 为地址低字节。在变址寻址方式下，CPU 使用 H:X 存放操作数的地址。

许多指令把 X 当作第二个通用寄存器，可以存放 8 位数据，可以对 X 进行清除，递增，递减，移位，求反，求补或翻转操作。

为了兼容 M68HC05，复位后 H 为 0。

7.2.3 堆栈指针

16 位地址指针寄存器指向后进先出型堆栈的下一个单元位置。堆栈用于自动保存子程序返回地址，中断子程序的返回地址和 CPU 寄存器，局部变量。

复位后，SP 的值为 0x00FF。用户在复位后的初始化程序中，重新初始化 SP，将堆栈空间移出 0x0000～0x00FF 空间。

7.2.4 程序计数器（PC）

PC 是一个 16 位的寄存器，用于存放下一条指令或预取操作数的地址。通常，每当一个指令码或操作数被预取后，程序计数器中的地址会自动增量指向下一个。但当遇到转移，分支或中断操作时就不同了，这时会将下一个地址压入堆栈，而新的转移地址装入 PC。

PC 在复位后自动装入 0xFFFE 和 0xFFFF 这两个存储器单元中的值作为程序的入口地址，即复位向量值。

7.2.5 条件码寄存器

7 V	溢出标志：当执行指令的结果发生了二进制补码溢出时，CPU 会置该位。 0：无溢出；1：有溢出
4 H	半进位标志：执行 ADD 或 ADC 操作时，累计器的 bit3 向 bit4 进位时，该位会置位。该标志主要用于二进制编码的十进制运算（BCD。累计器的十进制调整指令 DAA 会利用该位和 C 位来确定相应的修正值。 0：bit3 向 bit4 无进位；1：bit3 向 bit4 进位
3 I	中断屏蔽位：当某个中断发生时，在 CPU 寄存器压入堆栈后，中断向量被预取之前，中断屏蔽位将自动置 1。 在执行完清除 I 位指令（例如 CLI 或 TAP）之后，CPU 会执行完紧接在他们下面的指令。 0：开放 CPU 中断；1：禁止 CPU 中断
2 N	运算结果为负值：对于带符号的数据操作，最高位为 1 表示负数，为 0 表示正数。 0：运算结果为正值；1：运算结果为负值
1 Z	零标志 0：运算结果非 0；1：运算结果为 0
0 C	进位标志：加法操作使累加器的最高位产生进位或者减法操作产生借位时置该位为 1。部分逻辑操作和数据操作也会影响该位。

	0：无进位或借位；1：产生进位或借位

7.3 寻址模式

7.3.1 隐含寻址模式（INH）

在隐含寻址方式中，指令的所有有关信息均在操作码中，它可访问累加器、变址寄存器和条件码寄存器，均为单字节指令。比如 DAA， CLRA， DIV。指令中只有操作码，没有操作数。指令的意义是明显的，不需要任何操作数。例如： CLRA 功能：把累加器 A 清零。

7.3.2 相对寻址模式（REL）

相对寻址主要用于相对转移指令。一般来讲，转移指令（不包括位操作数转移指令）由二个字节组成，其中一字节为操作码，一字节为相对偏移量。相对偏移量为有符号二进制数。如果转移条件为真，则把指令的第二字节的 8 位存在的偏移量加到 PC 中，形成有效转移地址。当转移条件为假时，执行转移指令后的指令。相对寻址的范围为操作码地址开始的-126 到+129 字节。在使用汇编程序时，用户不必要自己计算偏移量，汇编程序会计算偏移量，并进行校验，看它是否在转移范围内。有四个新的转移指令(BLT，BGT，BLE，BGE)通过访问标志位 N，Z，V 来确定相关的有符号操作数的值。 BEQ $80 如果条件成立：PC= $80; 否则顺序执行。

BRA ABC 无条件转向标号 ABC 处；

7.3.3 立即寻址模式（IMM）

在立即寻址方式中，实际操作数紧跟在指令码之后，字节数与寄存器大小一致，这类指令含有 2、3 或 4 个（当需前置字节时）字节，紧跟在操作码后面的 # 是说明符，说明后面紧跟的是操作数值而不是操作数地址。立即数限制为 1 或 2 个字节，它取决于指令所使用的寄存器大小。例如：LDA #$FF 功能：把十六进制值$FF 送到累加器 A 中。

7.3.4 直接寻址模式（DIR）

在直接寻址方式中，操作数的有效地址（EA，16 位）的低字节放在操作码后面的一个字节中，地址的高位字节默认为$00，不包含在指令中。直接寻址指令（两字节）能访问存储器中最低的 256 字节（$0000-$00FF），绝大部分指令都具有直接寻址方式。它节省了指令空间，提高了指令的执行速度。直接寻址方式只能对$0000-$00FF 内存空间中的操作数进行操作。有时也将直接寻址方式称为零寻址方式。直接寻址方式指令比效果相当的扩展寻址指令少用了一个字节的程序内存空间，由于不需访问扩展内存，指令的执行时间减少了一个时钟周期。对于一个大型程序，这样的节省是很可观的。大部分单片机将$0000-$00FF 的内存空出，以便设计者放置那些经常需要调用的数据。

例如：LDA $50 功能：把地址为$0050 的单元内容送到累加器 A 中。

7.3.5 扩展寻址方式（EXT）

扩展寻址方式，操作数的有效地址为操作码后面的两个字节。因此，大多数扩展地址指令为 3 个字节：一个字节是操作码，两个字节是有效地址。在汇编语言中，在单字节地

址前加符号<表示直接寻址。否则，编译时会在该地址字节前加上一个$00 字节，就成了扩展寻址。

例如， LDA <$40 表示汇编时用直接寻址。LDA $40 汇编时将生成两字节操作数$0040，属扩展寻址。扩展变址中，操作数占两个字节。操作数表示的是地址，故寻址范围是 64K。例如：LDA $0400 功能：把地址为$0400 的单元内容送到累加器 A 中。

7.3.6 变址寻址方式

在变址寻址方式中，操作数的有效地址由 16 位变址寄存器 H:X 中的内容和跟在操作码后的无符号偏移量决定。

7.3.6.1 无偏移量变址寻址（IX）

操作数的有效地址在 16 位变址寄存器 H:X 中。

7.3.6.2 无偏移量变址后加 1 寻址（IX+）

操作数的有效地址在 16 位变址寄存器 H:X 中，操作数预取后，H:X=H:X+0x0001。

7.3.6.3 8 位偏移量变址寻址方式（IX1）

操作数的有效地址使 16 位变址寄存器 H:X 中的无符号数与操作码后一个字节无符号整数之和。

7.3.6.4 8 位偏移量变址后加 1 寻址方式（IX1+）

操作数的有效地址使 16 位变址寄存器 H:X 中的无符号数与操作码后一个字节无符号整数之和，操作数预取后，H:X=H:X+0x0001。

7.3.6.5 16 位偏移量寻址方式（IX2）

操作数有效地址为 16 位变址寄存器 H:X 中的无符号整数与操作码后的两字节无符号整数之和。

7.3.6.6 8 位偏移量堆栈寻址方式（SP1）

操作数有效地址为 16 位堆栈指针 SP 中的内容和跟在操作码后的 8 位无符号偏移量的和决定.

7.3.6.7 16 位偏移量堆栈寻址方式(SP2)

操作数有效地址为 16 位堆栈指针 SP 中的内容和跟在操作码后的 16 位无符号偏移量的和决定.

7.4 特殊操作

7.4.1 复位时序

上电复位,看门狗复位或外部复位引脚施加低电平引发 MCU 复位。当复位事件发生，CPU 立刻停止运行。

7.4.2 中断时序

当一个中断事件向 CPU 请求时，CPU 首先完成当前正在执行的指令。然后 CPU 执行以下操作：

1. 依次将 PCL,PCH,X,A,CCR 压入堆栈
2. 设置 CCR 中的 I=1
3. 预取中断向量的高 8 位
4. 预取中断向量的低 8 位
5. 延时 1 个总线周期
6. 中断服务程序首条指令预取

软件中断指令（SWI）是不可屏蔽中断。

7.4.3 WAIT 模式

WAIT 指令将清除 CCR 中的 I 位，允许中断，然后停止 CPU 时钟，CPU 等待中断或复位事件。当中断或复位事件发生，CPU 时钟恢复，中断或复位事件被正常执行。

该模式下，BACKGROUND 命令可以唤醒 CPU，CPU 进入后台调试活动模式。从而其他的调试命令可以被执行。

7.4.4 STOP 模式

通常，系统的所有时钟均关闭，但可以通过设置，HCS08 维持一个最小频率的时钟，该时钟允许一个定期的信号来唤醒 CPU。

当调试系统连接到目标系统的 BKGD 引脚，通过后台调试接口，ENBDM 控制位被设置为 1，那么 MCU 进入 STOP 模式后，振荡器强制处于活动状态。这种情况下，BACKGROUND 命令将唤醒 STOP 模式下的 MCU，CPU 时钟恢复，CPU 进入后台调试活动模式。

7.4.5 后台调试指令

后台调试指令不能用在用户的程序中，当目标 MCU 处于后台调试模式下，只有复位或主调试系统发送 GO,TRACE1,TAGGO 命令，用户程序才能运行。

通过用 BGND 操作码替换在设定断点地址处的操作码，形成基于软件的断点。当程序运行到断点位置，CPU 强制进入后台调试活动模式。

8.5 HCS08 指令集

操作码

（）＝ 寄存器或存储器中的内容
← ＝ 装载
& ＝ 与运算
| ＝ 或运算
X ＝ 乘运算
÷ ＝ 除运算
：＝ 连接
＋ ＝ 加运算
－ ＝ 求反

CPU 寄存器

A = 累加器
CCR＝条件码寄存器
H=变址寄存器高 8 位
X＝变址寄存器低 8 位
PC=程序计数器
PCH＝程序计数器高 8 位
PCL＝程序计数器低 8 位
SP=堆栈指针

存储器或地址

M = 存储器地址或绝对数据

条件码寄存器位

V = 溢出标志
H = 半进位标志
I = 中断屏蔽位
N = 负标志
Z = 零标志
C ＝ 进位/借位标志

机器码符号

dd = 直接地址的低 8 位
ee = 16 位偏移量的高 8 位
ff = 16 位偏移量的低 8 位或 8 位偏移量
ii ＝ 一字节立即数
jj = 16 位立即数的高 8 位
kk ＝ 16 位立即数的低 8 位
hh ＝ 16 位扩展地址的高 8 位
ll ＝ 16 位扩展地址的低 8 位
rr ＝ 相对偏移量

源操作数格式

n－任意标号或表达式（0～7）
opr8i－任意标号或表达式（8 位立即数）
opr16i－任意标号或表达式（16 位立即数）
opr8a－任意标号或表达式（8 位值）
opr16a－任意标号或表达式（16 位值）
oprx8－任意标号或表达式（8 位值）
oprx16－任意标号或表达式（16 位值）
rel－任意标号或表达式（－128～＋127）

寻址模式

见寻址模式章节

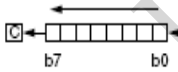
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
ADC #*opr8i* ADC *opr8a* ADC *opr16a* ADC *oprx16*,X ADC *oprx8*,X ADC ,X ADC *oprx16*,SP ADC *oprx8*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 B9 C9 D9 E9 F9 9ED9 9EE9	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
ADD #*opr8i* ADD *opr8a* ADD *opr16a* ADD *oprx16*,X ADD *oprx8*,X ADD ,X ADD *oprx16*,SP ADD *oprx8*,SP	Add without Carry	A ← (A) + (M)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	AB BB CB DB EB FB 9EDB 9EEB	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
AIS #*opr8i*	Add Immediate Value (Signed) to Stack Pointer	SP ← (SP) + (M) M is sign extended to a 16-bit value	–	–	–	–	–	–	IMM	A7	ii	2
AIX #*opr8i*	Add Immediate Value (Signed) to Index Register (H:X)	H:X ← (H:X) + (M) M is sign extended to a 16-bit value	–	–	–	–	–	–	IMM	AF	ii	2
AND #*opr8i* AND *opr8a* AND *opr16a* AND *oprx16*,X AND *oprx8*,X AND ,X AND *oprx16*,SP AND *oprx8*,SP	Logical AND	A ← (A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 B4 C4 D4 E4 F4 9ED4 9EE4	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
ASL *opr8a* ASLA ASLX ASL *oprx8*,X ASL ,X ASL *oprx8*,SP	Arithmetic Shift Left (Same as LSL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	5 1 1 5 4 6
ASR *opr8a* ASRA ASRX ASR *oprx8*,X ASR ,X ASR *oprx8*,SP	Arithmetic Shift Right		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	5 1 1 5 4 6
BCC *rel*	Branch if Carry Bit Clear	Branch if (C) = 0	–	–	–	–	–	–	REL	24	rr	3

表 7.1 指令集（7－1）

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
BCLR *n,opr8a*	Clear Bit n in Memory	Mn ← 0	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ *rel*	Branch if Equal	Branch if (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE *rel*	Branch if Greater Than or Equal To (Signed Operands)	Branch if (N ⊕ V) = 0	–	–	–	–	–	–	REL	90	rr	3
BGND	Enter Active Background if ENBDM = 1	Waits For and Processes BDM Commands Until GO, TRACE1, or TAGGO	–	–	–	–	–	–	INH	82		5+
BGT *rel*	Branch if Greater Than (Signed Operands)	Branch if (Z) ∣ (N ⊕ V) = 0	–	–	–	–	–	–	REL	92	rr	3
BHCC *rel*	Branch if Half Carry Bit Clear	Branch if (H) = 0	–	–	–	–	–	–	REL	28	rr	3
BHCS *rel*	Branch if Half Carry Bit Set	Branch if (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI *rel*	Branch if Higher	Branch if (C) ∣ (Z) = 0	–	–	–	–	–	–	REL	22	rr	3
BHS *rel*	Branch if Higher or Same (Same as BCC)	Branch if (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if IRQ Pin High	Branch if IRQ pin = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if IRQ Pin Low	Branch if IRQ pin = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT *#opr8i* BIT *opr8a* BIT *oprx16a* BIT *oprx16,X* BIT *oprx8,X* BIT ,X BIT *oprx16,SP* BIT *oprx8,SP*	Bit Test	(A) & (M) (CCR Updated but Operands Not Changed)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 B5 C5 D5 E5 F5 9ED5 9EE5	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
BLE *rel*	Branch if Less Than or Equal To (Signed Operands)	Branch if (Z) ∣ (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO *rel*	Branch if Lower (Same as BCS)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	Branch if (C) ∣ (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *rel*	Branch if Less Than (Signed Operands)	Branch if (N ⊕ V) = 1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	Branch if (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	Branch if (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	Branch if (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	Branch if (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	Branch if (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	No Test	–	–	–	–	–	–	REL	20	rr	3

表 7.2 指令集（7－2）

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
BRCLR *n,opr8a,rel*	Branch if Bit *n* in Memory Clear	Branch if (Mn) = 0	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	Uses 3 Bus Cycles	–	–	–	–	–	–	REL	21	rr	3
BRSET *n,opr8a,rel*	Branch if Bit *n* in Memory Set	Branch if (Mn) = 1	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n,opr8a*	Set Bit *n* in Memory	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BSR *rel*	Branch to Subroutine	PC ← (PC) + 0x0002 push (PCL); SP ← (SP) – 0x0001 push (PCH); SP ← (SP) – 0x0001 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	5
CBEQ *opr8a,rel* CBEQA *#opr8i,rel* CBEQX *#opr8i,rel* CBEQ *oprx8,X+,rel* CBEQ *,X+,rel* CBEQ *oprx8,SP,rel*	Compare and Branch if Equal	Branch if (A) = (M) Branch if (A) = (M) Branch if (X) = (M) Branch if (A) = (M) Branch if (A) = (M) Branch if (A) = (M)	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 5 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask Bit	I ← 0	–	–	0	–	–	–	INH	9A		1
CLR *opr8a* CLRA CLRX CLRH CLR *oprx8,X* CLR *,X* CLR *oprx8,SP*	Clear	M ← 0x00 A ← 0x00 X ← 0x00 H ← 0x00 M ← 0x00 M ← 0x00 M ← 0x00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	5 1 1 1 5 4 6
CMP *#opr8i* CMP *opr8a* CMP *opr16a* CMP *oprx16,X* CMP *oprx8,X* CMP *,X* CMP *oprx16,SP* CMP *oprx8,SP*	Compare Accumulator with Memory	(A) – (M) (CCR Updated But Operands Not Changed)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	A1 B1 C1 D1 E1 F1 9ED1 9EE1	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
COM *opr8a* COMA COMX COM *oprx8,X* COM *,X* COM *oprx8,SP*	Complement (One's Complement)	M ← (M̄)= 0xFF – (M) A ← (Ā) = 0xFF – (A) X ← (X̄) = 0xFF – (X) M ← (M̄) = 0xFF – (M) M ← (M̄) = 0xFF – (M) M ← (M̄) = 0xFF – (M)	0	–	–	↕	↕	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	5 1 1 5 4 6
CPHX *opr16a* CPHX *#opr16i* CPHX *opr8a* CPHX *oprx8,SP*	Compare Index Register (H:X) with Memory	(H:X) – (M:M + 0x0001) (CCR Updated But Operands Not Changed)	↕	–	–	↕	↕	↕	EXT IMM DIR SP1	3E 65 75 9EF3	hh ll jj kk dd ff	6 3 5 6

表 7.3 指令集（7—3）

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
CPX #*opr8i* CPX *opr8a* CPX *opr16a* CPX *oprx16*,X CPX *oprx8*,X CPX ,X CPX *oprx16*,SP CPX *oprx8*,SP	Compare X (Index Register Low) with Memory	(X) – (M) (CCR Updated But Operands Not Changed)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	A3 B3 C3 D3 E3 F3 9ED3 9EE3	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	$(A)_{10}$	U	–	–	↕	↕	↕	INH	72		1
DBNZ *opr8a,rel* DBNZA *rel* DBNZX *rel* DBNZ *oprx8,X,rel* DBNZ ,X,*rel* DBNZ *oprx8*,SP,*rel*	Decrement and Branch if Not Zero	Decrement A, X, or M Branch if (result) ≠ 0 DBNZX Affects X Not H	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	7 4 4 7 6 8
DEC *opr8a* DECA DECX DEC *oprx8,X* DEC ,X DEC *oprx8*,SP	Decrement	M ← (M) – 0x01 A ← (A) – 0x01 X ← (X) – 0x01 M ← (M) – 0x01 M ← (M) – 0x01 M ← (M) – 0x01	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	5 1 1 5 4 6
DIV	Divide	A ← (H:A)÷(X) H ← Remainder	–	–	–	–	↕	↕	INH	52		6
EOR #*opr8i* EOR *opr8a* EOR *opr16a* EOR *oprx16*,X EOR *oprx8*,X EOR ,X EOR *oprx16*,SP EOR *oprx8*,SP	Exclusive OR Memory with Accumulator	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 B8 C8 D8 E8 F8 9ED8 9EE8	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
INC *opr8a* INCA INCX INC *oprx8,X* INC ,X INC *oprx8*,SP	Increment	M ← (M) + 0x01 A ← (A) + 0x01 X ← (X) + 0x01 M ← (M) + 0x01 M ← (M) + 0x01 M ← (M) + 0x01	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	5 1 1 5 4 6
JMP *opr8a* JMP *opr16a* JMP *oprx16,X* JMP *oprx8,X* JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	3 4 4 3 3
JSR *opr8a* JSR *opr16a* JSR *oprx16,X* JSR *oprx8,X* JSR ,X	Jump to Subroutine	PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – 0x0001 Push (PCH); SP ← (SP) – 0x0001 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	5 6 6 5 5
LDA #*opr8i* LDA *opr8a* LDA *opr16a* LDA *oprx16,X* LDA *oprx8,X* LDA ,X LDA *oprx16*,SP LDA *oprx8*,SP	Load Accumulator from Memory	A ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A6 B6 C6 D6 E6 F6 9ED6 9EE6	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
LDHX #*opr16i* LDHX *opr8a* LDHX *opr16a* LDHX ,X LDHX *oprx16,X* LDHX *oprx8,X* LDHX *oprx8*,SP	Load Index Register (H:X) from Memory	H:X ← (M:M + 0x0001)	0	–	–	↕	↕	–	IMM DIR EXT IX IX2 IX1 SP1	45 55 32 9EAE 9EBE 9ECE 9EFE	jj kk dd hh ll ee ff ff ff	3 4 5 5 6 5 5

表 7.4 指令集（7－4）

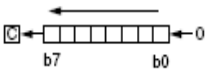
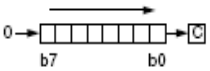
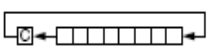
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
LDX #*opr8i* LDX *opr8a* LDX *opr16a* LDX *oprx16*,X LDX *oprx8*,X LDX ,X LDX *oprx16*,SP LDX *oprx8*,SP	Load X (Index Register Low) from Memory	X ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE BE CE DE EE FE 9EDE 9EEE	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
LSL *opr8a* LSLA LSLX LSL *oprx8*,X LSL ,X LSL *oprx8*,SP	Logical Shift Left (Same as ASL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	5 1 1 5 4 6
LSR *opr8a* LSRA LSR*X* LSR *oprx8*,X LSR ,X LSR *oprx8*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	5 1 1 5 4 6
MOV *opr8a*,*opr8a* MOV *opr8a*,X+ MOV #*opr8i*,*opr8a* MOV ,X+,*opr8a*	Move	$(M)_{destination} \leftarrow (M)_{source}$ H:X ← (H:X) + 0x0001 in IX+/DIR and DIR/IX+ Modes	0	–	–	↕	↕	–	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E 5E 6E 7E	dd dd dd ii dd dd	5 5 4 5
MUL	Unsigned multiply	X:A ← (X) × (A)	–	0	–	–	–	0	INH	42		5
NEG *opr8a* NEGA NEGX NEG *oprx8*,X NEG ,X NEG *oprx8*,SP	Negate (Two's Complement)	M ← – (M) = 0x00 – (M) A ← – (A) = 0x00 – (A) X ← – (X) = 0x00 – (X) M ← – (M) = 0x00 – (M) M ← – (M) = 0x00 – (M) M ← – (M) = 0x00 – (M)		–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	5 1 1 5 4 6
NOP	No Operation	Uses 1 Bus Cycle	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap Accumulator	A ← (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	62		1
ORA #*opr8i* ORA *opr8a* ORA *opr16a* ORA *oprx16*,X ORA *oprx8*,X ORA ,X ORA *oprx16*,SP ORA *oprx8*,SP	Inclusive OR Accumulator and Memory	A ← (A) | (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA BA CA DA EA FA 9EDA 9EEA	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
PSHA	Push Accumulator onto Stack	Push (A); SP ← (SP) – 0x0001	–	–	–	–	–	–	INH	87		2
PSHH	Push H (Index Register High) onto Stack	Push (H); SP ← (SP) – 0x0001	–	–	–	–	–	–	INH	8B		2
PSHX	Push X (Index Register Low) onto Stack	Push (X); SP ← (SP) – 0x0001	–	–	–	–	–	–	INH	89		2
PULA	Pull Accumulator from Stack	SP ← (SP + 0x0001); Pull (A)	–	–	–	–	–	–	INH	86		3
PULH	Pull H (Index Register High) from Stack	SP ← (SP + 0x0001); Pull (H)	–	–	–	–	–	–	INH	8A		3
PULX	Pull X (Index Register Low) from Stack	SP ← (SP + 0x0001); Pull (X)	–	–	–	–	–	–	INH	88		3
ROL *opr8a* ROLA ROLX ROL *oprx8*,X ROL ,X ROL *oprx8*,SP	Rotate Left through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	5 1 1 5 4 6

表 7.5 指令集（7－5）

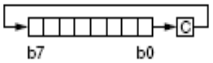
Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Bus Cycles[1]
ROR *opr8a* RORA RORX ROR *oprx8*,X ROR ,X ROR *oprx8*,SP	Rotate Right through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	5 1 1 5 4 6
RSP	Reset Stack Pointer	SP ← 0xFF (High Byte Not Affected)	–	–	–	–	–	–	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 0x0001; Pull (CCR) SP ← (SP) + 0x0001; Pull (A) SP ← (SP) + 0x0001; Pull (X) SP ← (SP) + 0x0001; Pull (PCH) SP ← (SP) + 0x0001; Pull (PCL)	↕	↕	↕	↕	↕	↕	INH	80		9
RTS	Return from Subroutine	SP ← SP + 0x0001; Pull (PCH) SP ← SP + 0x0001; Pull (PCL)	–	–	–	–	–	–	INH	81		6
SBC *#opr8i* SBC *opr8a* SBC *opr16a* SBC *oprx16*,X SBC *oprx8*,X SBC ,X SBC *oprx16*,SP SBC *oprx8*,SP	Subtract with Carry	A ← (A) – (M) – (C)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	A2 B2 C2 D2 E2 F2 9ED2 9EE2	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
SEC	Set Carry Bit	C ← 1	–	–	–	–	–	1	INH	99		1
SEI	Set Interrupt Mask Bit	I ← 1	–	–	1	–	–	–	INH	9B		1
STA *opr8a* STA *opr16a* STA *oprx16*,X STA *oprx8*,X STA ,X STA *oprx16*,SP STA *oprx8*,SP	Store Accumulator in Memory	M ← (A)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP2 SP1	B7 C7 D7 E7 F7 9ED7 9EE7	dd hh ll ee ff ff ee ff ff	3 4 4 3 2 5 4
STHX *opr8a* STHX *opr16a* STHX *oprx8*,SP	Store H:X (Index Reg.)	(M:M + 0x0001) ← (H:X)	0	–	–	↕	↕	–	DIR EXT SP1	35 96 9EFF	dd hh ll ff	4 5 5
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation	I bit ← 0; Stop Processing	–	–	0	–	–	–	INH	8E		2+
STX *opr8a* STX *opr16a* STX *oprx16*,X STX *oprx8*,X STX ,X STX *oprx16*,SP STX *oprx8*,SP	Store X (Low 8 Bits of Index Register) in Memory	M ← (X)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP2 SP1	BF CF DF EF FF 9EDF 9EEF	dd hh ll ee ff ff ee ff ff	3 4 4 3 2 5 4
SUB *#opr8i* SUB *opr8a* SUB *opr16a* SUB *oprx16*,X SUB *oprx8*,X SUB ,X SUB *oprx16*,SP SUB *oprx8*,SP	Subtract	A ← (A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 B0 C0 D0 E0 F0 9ED0 9EE0	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
SWI	Software Interrupt	PC ← (PC) + 0x0001 Push (PCL); SP ← (SP) – 0x0001 Push (PCH); SP ← (SP) – 0x0001 Push (X); SP ← (SP) – 0x0001 Push (A); SP ← (SP) – 0x0001 Push (CCR); SP ← (SP) – 0x0001 I ← 1; PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	–	–	1	–	–	–	INH	83		11

表 7.6 指令集（7－6）

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
TAP	Transfer Accumulator to CCR	CCR ← (A)	↕	↕	↕	↕	↕	↕	INH	84		1
TAX	Transfer Accumulator to X (Index Register Low)	X ← (A)	–	–	–	–	–	–	INH	97		1
TPA	Transfer CCR to Accumulator	A ← (CCR)	–	–	–	–	–	–	INH	85		1
TST *opr8a* TSTA TSTX TST *oprx8*,X TST ,X TST *oprx8*,SP	Test for Negative or Zero	(M) – 0x00 (A) – 0x00 (X) – 0x00 (M) – 0x00 (M) – 0x00 (M) – 0x00	0	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	4 1 1 4 3 5
TSX	Transfer SP to Index Reg.	H:X ← (SP) + 0x0001	–	–	–	–	–	–	INH	95		2
TXA	Transfer X (Index Reg. Low) to Accumulator	A ← (X)	–	–	–	–	–	–	INH	9F		1
TXS	Transfer Index Reg. to SP	SP ← (H:X) – 0x0001	–	–	–	–	–	–	INH	94		2
WAIT	Enable Interrupts; Wait for Interrupt	I bit ← 0; Halt CPU	–	–	0	–	–	–	INH	8F		2+

表 7.7 指令集（7—7）

Bit-Manipulation		Branch	Read-Modify-Write					Control		Register/Memory					
00 5 BRSET0 3 DIR	10 5 BSET0 2 DIR	20 3 BRA 2 REL	30 5 NEG 2 DIR	40 1 NEGA 1 INH	50 1 NEGX 1 INH	60 5 NEG 2 IX1	70 4 NEG 1 IX	80 9 RTI 1 INH	90 3 BGE 2 REL	A0 2 SUB 2 IMM	B0 3 SUB 2 DIR	C0 4 SUB 3 EXT	D0 4 SUB 3 IX2	E0 3 SUB 2 IX1	F0 3 SUB 1 IX
01 5 BRCLR0 3 DIR	11 5 BCLR0 2 DIR	21 3 BRN 2 REL	31 5 CBEQ 3 DIR	41 4 CBEQA 3 IMM	51 4 CBEQX 3 IMM	61 5 CBEQ 3 IX1+	71 5 CBEQ 2 IX+	81 6 RTS 1 INH	91 3 BLT 2 REL	A1 2 CMP 2 IMM	B1 3 CMP 2 DIR	C1 4 CMP 3 EXT	D1 4 CMP 3 IX2	E1 3 CMP 2 IX1	F1 3 CMP 1 IX
02 5 BRSET1 3 DIR	12 5 BSET1 2 DIR	22 3 BHI 2 REL	32 5 LDHX 3 EXT	42 5 MUL 1 INH	52 6 DIV 1 INH	62 1 NSA 1 INH	72 1 DAA 1 INH	82 5+ BGND 1 INH	92 3 BGT 2 REL	A2 2 SBC 2 IMM	B2 3 SBC 2 DIR	C2 4 SBC 3 EXT	D2 4 SBC 3 IX2	E2 3 SBC 2 IX1	F2 3 SBC 1 IX
03 5 BRCLR1 3 DIR	13 5 BCLR1 2 DIR	23 3 BLS 2 REL	33 5 COM 2 DIR	43 1 COMA 1 INH	53 1 COMX 1 INH	63 5 COM 2 IX1	73 4 COM 1 IX	83 11 SWI 1 INH	93 3 BLE 2 REL	A3 2 CPX 2 IMM	B3 3 CPX 2 DIR	C3 4 CPX 3 EXT	D3 4 CPX 3 IX2	E3 3 CPX 2 IX1	F3 3 CPX 1 IX
04 5 BRSET2 3 DIR	14 5 BSET2 2 DIR	24 3 BCC 2 REL	34 5 LSR 2 DIR	44 1 LSRA 1 INH	54 1 LSRX 1 INH	64 5 LSR 2 IX1	74 4 LSR 1 IX	84 1 TAP 1 INH	94 2 TXS 1 INH	A4 2 AND 2 IMM	B4 3 AND 2 DIR	C4 4 AND 3 EXT	D4 4 AND 3 IX2	E4 3 AND 2 IX1	F4 3 AND 1 IX
05 5 BRCLR2 3 DIR	15 5 BCLR2 2 DIR	25 3 BCS 2 REL	35 4 STHX 2 DIR	45 3 LDHX 3 IMM	55 4 LDHX 2 DIR	65 3 CPHX 3 IMM	75 5 CPHX 2 DIR	85 1 TPA 1 INH	95 2 TSX 1 INH	A5 2 BIT 2 IMM	B5 3 BIT 2 DIR	C5 4 BIT 3 EXT	D5 4 BIT 3 IX2	E5 3 BIT 2 IX1	F5 3 BIT 1 IX
06 5 BRSET3 3 DIR	16 5 BSET3 2 DIR	26 3 BNE 2 REL	36 5 ROR 2 DIR	46 1 RORA 1 INH	56 1 RORX 1 INH	66 5 ROR 2 IX1	76 4 ROR 1 IX	86 3 PULA 1 INH	96 5 STHX 3 EXT	A6 2 LDA 2 IMM	B6 3 LDA 2 DIR	C6 4 LDA 3 EXT	D6 4 LDA 3 IX2	E6 3 LDA 2 IX1	F6 3 LDA 1 IX
07 5 BRCLR3 3 DIR	17 5 BCLR3 2 DIR	27 3 BEQ 2 REL	37 5 ASR 2 DIR	47 1 ASRA 1 INH	57 1 ASRX 1 INH	67 5 ASR 2 IX1	77 4 ASR 1 IX	87 2 PSHA 1 INH	97 1 TAX 1 INH	A7 2 AIS 2 IMM	B7 3 STA 2 DIR	C7 4 STA 3 EXT	D7 4 STA 3 IX2	E7 3 STA 2 IX1	F7 2 STA 1 IX
08 5 BRSET4 3 DIR	18 5 BSET4 2 DIR	28 3 BHCC 2 REL	38 5 LSL 2 DIR	48 1 LSLA 1 INH	58 1 LSLX 1 INH	68 5 LSL 2 IX1	78 4 LSL 1 IX	88 3 PULX 1 INH	98 1 CLC 1 INH	A8 2 EOR 2 IMM	B8 3 EOR 2 DIR	C8 4 EOR 3 EXT	D8 4 EOR 3 IX2	E8 3 EOR 2 IX1	F8 3 EOR 1 IX
09 5 BRCLR4 3 DIR	19 5 BCLR4 2 DIR	29 3 BHCS 2 REL	39 5 ROL 2 DIR	49 1 ROLA 1 INH	59 1 ROLX 1 INH	69 5 ROL 2 IX1	79 4 ROL 1 IX	89 2 PSHX 1 INH	99 1 SEC 1 INH	A9 2 ADC 2 IMM	B9 3 ADC 2 DIR	C9 4 ADC 3 EXT	D9 4 ADC 3 IX2	E9 3 ADC 2 IX1	F9 3 ADC 1 IX
0A 5 BRSET5 3 DIR	1A 5 BSET5 2 DIR	2A 3 BPL 2 REL	3A 5 DEC 2 DIR	4A 1 DECA 1 INH	5A 1 DECX 1 INH	6A 5 DEC 2 IX1	7A 4 DEC 1 IX	8A 3 PULH 1 INH	9A 1 CLI 1 INH	AA 2 ORA 2 IMM	BA 3 ORA 2 DIR	CA 4 ORA 3 EXT	DA 4 ORA 3 IX2	EA 3 ORA 2 IX1	FA 3 ORA 1 IX
0B 5 BRCLR5 3 DIR	1B 5 BCLR5 2 DIR	2B 3 BMI 2 REL	3B 7 DBNZ 3 DIR	4B 4 DBNZA 2 INH	5B 4 DBNZX 2 INH	6B 7 DBNZ 3 IX1	7B 6 DBNZ 2 IX	8B 2 PSHH 1 INH	9B 1 SEI 1 INH	AB 2 ADD 2 IMM	BB 3 ADD 2 DIR	CB 4 ADD 3 EXT	DB 4 ADD 3 IX2	EB 3 ADD 2 IX1	FB 3 ADD 1 IX
0C 5 BRSET6 3 DIR	1C 5 BSET6 2 DIR	2C 3 BMC 2 REL	3C 5 INC 2 DIR	4C 1 INCA 1 INH	5C 1 INCX 1 INH	6C 5 INC 2 IX1	7C 4 INC 1 IX	8C 1 CLRH 1 INH	9C 1 RSP 1 INH		BC 3 JMP 2 DIR	CC 4 JMP 3 EXT	DC 4 JMP 3 IX2	EC 3 JMP 2 IX1	FC 3 JMP 1 IX
0D 5 BRCLR6 3 DIR	1D 5 BCLR6 2 DIR	2D 3 BMS 2 REL	3D 4 TST 2 DIR	4D 1 TSTA 1 INH	5D 1 TSTX 1 INH	6D 4 TST 2 IX1	7D 3 TST 1 IX		9D 1 NOP 1 INH	AD 5 BSR 2 REL	BD 5 JSR 2 DIR	CD 6 JSR 3 EXT	DD 6 JSR 3 IX2	ED 5 JSR 2 IX1	FD 5 JSR 1 IX
0E 5 BRSET7 3 DIR	1E 5 BSET7 2 DIR	2E 3 BIL 2 REL	3E 6 CPHX 3 EXT	4E 5 MOV 3 DD	5E 5 MOV 2 DIX+	6E 4 MOV 3 IMD	7E 5 MOV 2 IX+D	8E 2+ STOP 1 INH	9E Page 2	AE 2 LDX 2 IMM	BE 3 LDX 2 DIR	CE 4 LDX 3 EXT	DE 4 LDX 3 IX2	EE 3 LDX 2 IX1	FE 3 LDX 1 IX
0F 5 BRCLR7 3 DIR	1F 5 BCLR7 2 DIR	2F 3 BIH 2 REL	3F 5 CLR 2 DIR	4F 1 CLRA 1 INH	5F 1 CLRX 1 INH	6F 5 CLR 2 IX1	7F 4 CLR 1 IX	8F 2+ WAIT 1 INH	9F 1 TXA 1 INH	AF 2 AIX 2 IMM	BF 3 STX 2 DIR	CF 4 STX 3 EXT	DF 4 STX 3 IX2	EF 3 STX 2 IX1	FF 2 STX 1 IX

INH	Inherent	REL	Relative	SP1	Stack Pointer, 8-Bit Offset
IMM	Immediate	IX	Indexed, No Offset	SP2	Stack Pointer, 16-Bit Offset
DIR	Direct	IX1	Indexed, 8-Bit Offset	IX+	Indexed, No Offset with Post Increment
EXT	Extended	IX2	Indexed, 16-Bit Offset	IX1+	Indexed, 1-Byte Offset with Post Increment
DD	DIR to DIR	IMD	IMM to DIR		
IX+D	IX+ to DIR	DIX+	DIR to IX+		

Opcode in Hexadecimal — F0 3 SUB — HCS08 Cycles / Instruction Mnemonic / Addressing Mode
Number of Bytes — 1 IX

表 7.8 操作码映射表（2-1）

操作码HEX码 指令助记符 HCS08周期
Opcode in Hexadecimal — F0 3 SUB — HCS08 Cycles / Instruction Mnemonic / Addressing Mode
Number of Bytes — 1 IX
字节数 寻址模式

Bit-Manipulation		Branch	Read-Modify-Write					Control		Register/Memory					
						9E60 6 NEG 3 SP1							9ED0 5 SUB 4 SP2	9EE0 4 SUB 3 SP1	
						9E61 6 CBEQ 4 SP1							9ED1 5 CMP 4 SP2	9EE1 4 CMP 3 SP1	
													9ED2 5 SBC 4 SP2	9EE2 4 SBC 3 SP1	
						9E63 6 COM 3 SP1							9ED3 5 CPX 4 SP2	9EE3 4 CPX 3 SP1	9EF3 6 CPHX 3 SP1
						9E64 6 LSR 3 SP1							9ED4 5 AND 4 SP2	9EE4 4 AND 3 SP1	
													9ED5 5 BIT 4 SP2	9EE5 4 BIT 3 SP1	
						9E66 6 ROR 3 SP1							9ED6 5 LDA 4 SP2	9EE6 4 LDA 3 SP1	
						9E67 6 ASR 3 SP1							9ED7 5 STA 4 SP2	9EE7 4 STA 3 SP1	
						9E68 6 LSL 3 SP1							9ED8 5 EOR 4 SP2	9EE8 4 EOR 3 SP1	
						9E69 6 ROL 3 SP1							9ED9 5 ADC 4 SP2	9EE9 4 ADC 3 SP1	
						9E6A 6 DEC 3 SP1							9EDA 5 ORA 4 SP2	9EEA 4 ORA 3 SP1	
						9E6B 8 DBNZ 4 SP1							9EDB 5 ADD 4 SP2	9EEB 4 ADD 3 SP1	
						9E6C 6 INC 3 SP1									
						9E6D 5 TST 3 SP1									
										9EAE 5 LDHX 2 IX	9EBE 6 LDHX 4 IX2	9ECE 5 LDHX 3 IX1	9EDE 5 LDX 4 SP2	9EEE 4 LDX 3 SP1	9EFE 5 LDHX 3 SP1
						9E6F 6 CLR 3 SP1							9EDF 5 STX 4 SP2	9EEF 4 STX 3 SP1	9EFF 5 STHX 3 SP1

INH	Inherent	REL	Relative	SP1	Stack Pointer, 8-Bit Offset
IMM	Immediate	IX	Indexed, No Offset	SP2	Stack Pointer, 16-Bit Offset
DIR	Direct	IX1	Indexed, 8-Bit Offset	IX+	Indexed, No Offset with Post Increment
EXT	Extended	IX2	Indexed, 16-Bit Offset	IX1+	Indexed, 1-Byte Offset with Post Increment
DD	DIR to DIR	IMD	IMM to DIR		
IX+D	IX+ to DIR	DIX+	DIR to IX+		

Note: All Sheet 2 Opcodes are Preceded by the Page 2 Prebyte (9E)

Prebyte (9E) and Opcode in Hexadecimal
Number of Bytes
9E60 6 NEG 3 SP1
HCS08 Cycles
Instruction Mnemonic
Addressing Mode

表 7.9 操作码映射表（2-2）

第八章 5V 模拟比较器

8．1 简介

模拟比较器模块 ACMP 用于比较两路模拟输入电压或者一路模拟输入电源与内部参考电压相比较。该模拟比较器可工作于供电电压的全电压范围。

8．1．1 ACMP 设置信息

当使用内部能隙电压作为 ACMP＋的输入时，用户应设置 SPMSC1 寄存器中的 BGBE 位为 1。

8．1．2 STOP 模式下的 ACMP

ACMP 可以运行在 STOP3 模式下。如果 ACOPE＝1，那么比较器的输出将工作于正常操作模式，并且控制 ACMPO 引脚。ACMP 可唤醒处于 STOP3 模式下的 MCU。

8．1．3 ACMP/TPM 配置信息

通过设置 SOPT2 寄存器中的 ACIC 位，可以将 ACMP 的输出连接到 TPM1 的通道 0。

8．1．4 特点

- 全电压范围内工作
- 比较器的输出可为上升沿，下降沿或任意边沿
- 可选的内部参考电压
- 比较器输出结果可在 ACMPO 引脚呈现
- 可工作于 STOP3 模式

8．1．5 工作模式

8.1.5.1 WAIT 模式下

在执行 WAIT 指令进入 WAIT 模式之前使能 ACMP，那么当 MCU 进入 WAIT 模式后，ACMP 仍会运行。因此，ACMP 的中断可唤醒处于 WAIT 模式的 MCU。

8.1.5.2 STOP 模式下

8.1.5.2.1 STOP3 模式下

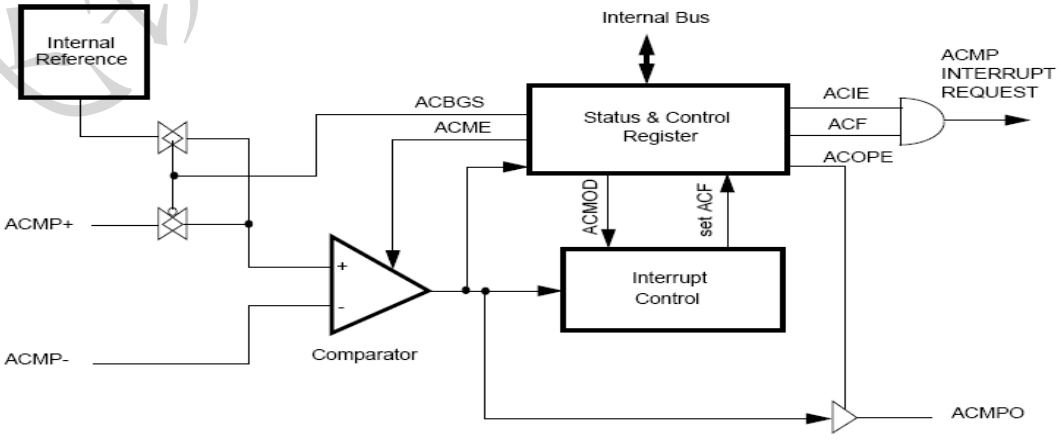
ACMP 可以运行在 STOP3 模式下。如果 ACOPE＝1，那么比较器的输出将工作于正常操作模式，并且控制 ACMPO 引脚。ACMP 可唤醒处于 STOP3 模式下的 MCU。

8.1.5.2.2 STOP2 和 STOP1 模式下

这两种模式下 ACMP 不再运行。

8.1.5.3 ACMP 在后台调试模式下

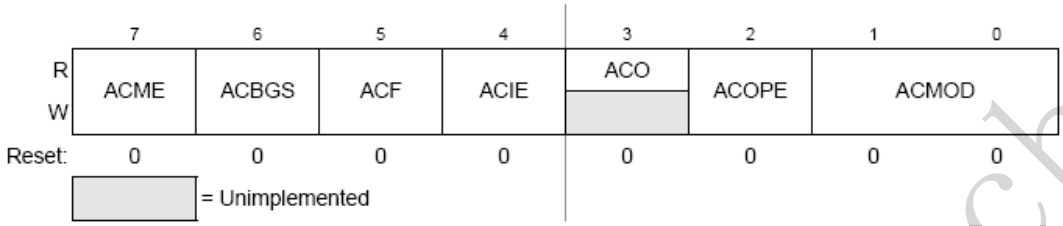
该模式下 ACMP 正常运行。



8.2 外部信号

信号	功能	输入/输出
ACMP-	反向输入	输入
ACMP+	正向输入	输入
ACMPO	ACMP 输出	输出

8.3 寄存器



7 ACME	比较器使能控制位 0：ACMP 禁止 1：ACMP 使能
6 ACBGS	比较器参考电压选择 0：ACMP+作为正向输入 1：内部参考电压连接到 ACMP+
5 ACF	比较器标志位：当比较事件发生，ACF 位置位。向 ACF 位写 1 可清除该位。 0：未发生比较事件 1：发生比较事件
4 ACIE	比较器中断使能 0：中断禁止 1：中断使能
3 ACO	比较器输出：读取 ACO 将得到当前模拟比较器输出值。
2 ACOPE	比较器输出引脚使能 0：比较器输出在 ACMPO 引脚不呈现 1：比较器输出在 ACMPO 引脚呈现
1:0 ACMOD	ACF 位置位条件 00：比较器输出为下降沿 01：比较器输出为上升沿 10：比较器输出为下降沿 11：比较器输出为任意边沿

8．4 功能

当比较器的正向输入大于反向输入，比较器输出为高；反之为低。而 ACMOD 可控制 ACF 位置位条件。

第九章模数转换器

9.1 概述

本章将介绍 10 位 ADC 模块及其操作。

9.1.1 通道

ADCH	Channel	Input	Pin Control
00000	AD0	PTB0/ADC1P0	ADPC0
00001	AD1	PTB1/ADC1P1	ADPC1
00010	AD2	PTB2/ADC1P2	ADPC2
00011	AD3	PTB3/ADC1P3	ADPC3
00100	AD4	PTB4/ADC1P4	ADPC4
00101	AD5	PTB5/ADC1P5	ADPC5
00110	AD6	PTB6/ADC1P6	ADPC6
00111	AD7	PTB7/ADC1P7	ADPC7
01000	AD8	V_{REFL}	N/A
01001	AD9	V_{REFL}	N/A
01010	AD10	V_{REFL}	N/A
01011	AD11	V_{REFL}	N/A
01100	AD12	V_{REFL}	N/A
01101	AD13	V_{REFL}	N/A
01110	AD14	V_{REFL}	N/A
01111	AD15	V_{REFL}	N/A

ADCH	Channel	Input	Pin Control
10000	AD16	V_{REFL}	N/A
10001	AD17	V_{REFL}	N/A
10010	AD18	V_{REFL}	N/A
10011	AD19	V_{REFL}	N/A
10100	AD20	V_{REFL}	N/A
10101	AD21	V_{REFL}	N/A
10110	AD22	Reserved	N/A
10111	AD23	Reserved	N/A
11000	AD24	Reserved	N/A
11001	AD25	Reserved	N/A
11010	AD26	Reserved	N/A
11011	AD27	Internal Bandgap	N/A
11100		Reserved	N/A
11101	V_{REFH}	V_{REFH}	N/A
11110	V_{REFL}	V_{REFL}	N/A
11111	module disabled	None	N/A

表 14-1 ADC 通道

9.1.2 ADC 模块时钟

ADC 模块使用 MCU 总线时钟进行 AD 转换，也可由 ICSERCLK 时钟作为 ADC 模块时钟。

当 MCU 处于等待模式时，ALTCLK 可作为 ADC 模块的时钟源。而在 STOP3 或 STOP2 模式下，ALTCLK 停止。

9.1.3 硬件触发机制

ADC 硬件触发器－ADHWT：RTI 的输出。RTI 计数器可由 ICSERCLK 时钟或 1-kHz 的时钟源驱动。1-kHz 时钟源可在正常运行模式，等待状态或 STOP3 模式下运行。

RTI 的定时时间由输入时钟频率和 RTCPS 位决定。RTI 自由运行，产生溢出中断。当 ADC 硬件触发使能，那么 RTI 的计数溢出中断将启动 ADC 转换。

9.1.4 温度传感器

9.1.5 特点

- 10 位线性逐次逼近算法
- 多大 28 路模拟输入
- 输出格式 10，或 8 位
- 单次或连续转换

- 可配置抽样时间和转换速率
- 转换完成标志和中断
- 输入时钟可选，多达 4 个时钟源
- 可工作于 STOP3 和 wait 模式
- 异步时钟源可用于低噪声操作
- 可选的异步硬件转换触发器
- 自动比较中断（当小于，大于或等于预定值）

9.1.6 ADC 框图

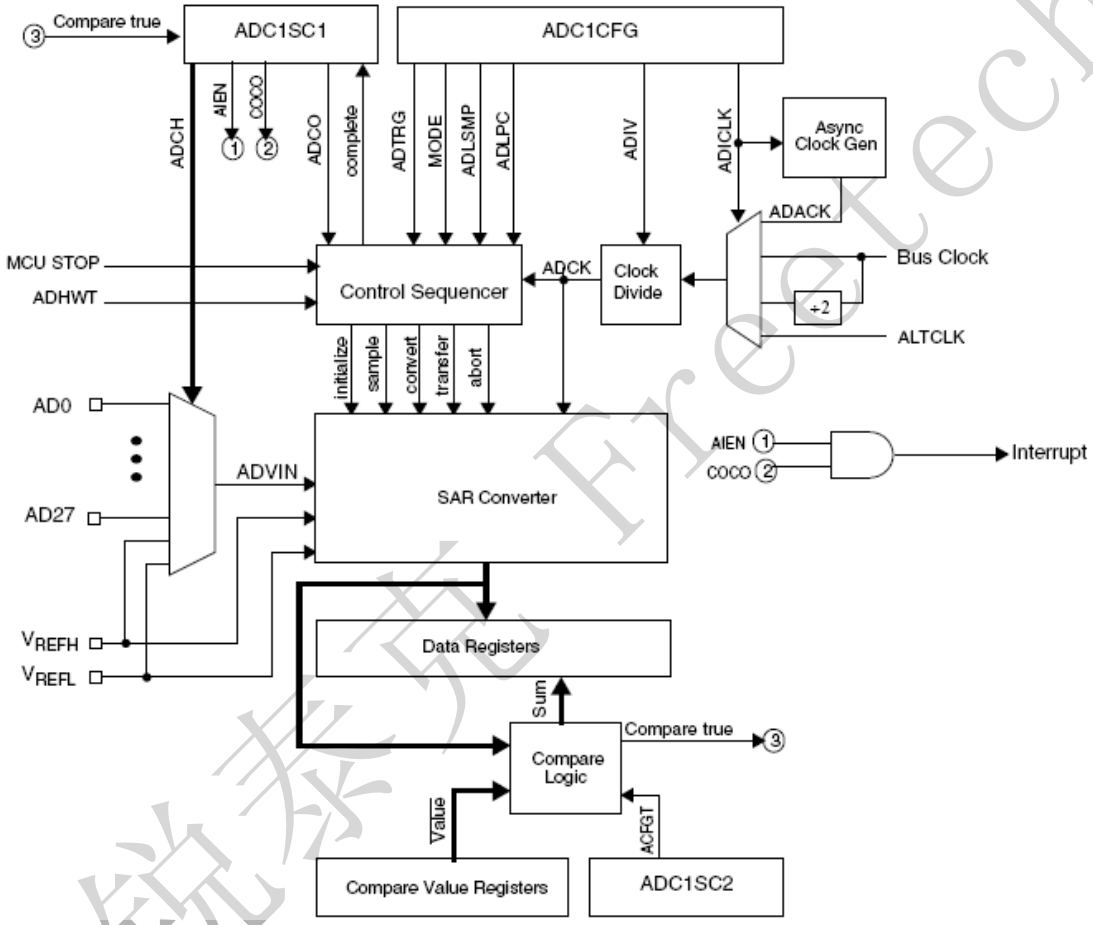


图 9-2 ADC 原理框图

9.2 外部信号描述

Name	Function
AD27–AD0	Analog Channel inputs
V_{REFH}	High reference voltage
V_{REFL}	Low reference voltage
V_{DDAD}	Analog power supply
V_{SSAD}	Analog ground

表 9.1 ADC 外部信号表

9.2.1 模拟电压（V_{DDAD}）

一般 V_{DDAD} 连接到 V_{DD}，加滤波电容。

9.2.2 模拟地（V_{SSAD}）

V_{SSAD} 可连接到 V_S

9.2.3 参考电压高（V_{REFH}）

一般连接到 V_{DDAD}

9.2.4 参考电压低（V_{REFL}）

一般连接到 V_{SSAD}

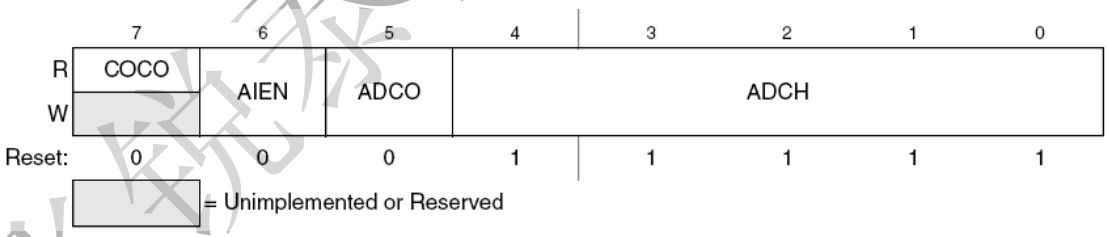
9.2.5 模拟输入通道（ADx）

ADC 模块支持多达 28 个独立的模拟输入。

9.3 寄存器定义

- 状态与控制寄存器，ADCSC1
- 状态与控制寄存器，ADCSC2
- 数据结果寄存器，ADCRH，AD1RL
- 比较值寄存器，ADCCVH，ADCCVL
- 配置寄存器，ADCCFG
- 引脚使能寄存器，APCTL1,APCTL2,APCTL3

9.3.1 状态和控制寄存器（ADCSC1）



7 COCO	转换完成标志－只读，比较功能禁止时（ACFE=0），当每次转换完成后，该位置位。当比较功能允许时（ACFE=1），只有比较结果为真时，COCO 置位。清除该位方法：任意时刻写 ADCSC1 或读 ADCRL 0：转换未完成；1：转换完成
6 AIEN	中断允许位 0：禁止转换完成中断；1：允许转换完成中断
5 ADCO	连续转换允许位 0：通过向写 ADCSC1 启动一次转换（选择的为软件触发）或通过 ADHWT 触发启动一次转换（选择的为硬件触发）； 1：通过向写 ADCSC1 启动连续转换（选择的为软件触发）或通过 ADHWT 触发启动连续转换（选择的为硬件触发）

4：0 ADCH	通道选择位一见下表。 当通道选择位均为 1 时，ADC 的转换系统关闭。利用这一特点可直接禁止 ADC 功能。当中止连续转换时将阻止再次转换。 当 ADC 未使能连续转换，那么没有必要将通道选择位都置 1 使 ADC 进入低功耗模式，因为 ADC 模块在完成转换后会自动进入低功耗模式

ADCH	Input Select
00000	AD0
00001	AD1
00010	AD2
00011	AD3
00100	AD4
00101	AD5
00110	AD6
00111	AD7

ADCH	Input Select
10000	AD16
10001	AD17
10010	AD18
10011	AD19
10100	AD20
10101	AD21
10110	AD22
10111	AD23

ADCH	Input Select
01000	AD8
01001	AD9
01010	AD10
01011	AD11
01100	AD12
01101	AD13
01110	AD14
01111	AD15

ADCH	Input Select
11000	AD24
11001	AD25
11010	AD26
11011	AD27
11100	Reserved
11101	V_{REFH}
11110	V_{REFL}
11111	Module disabled

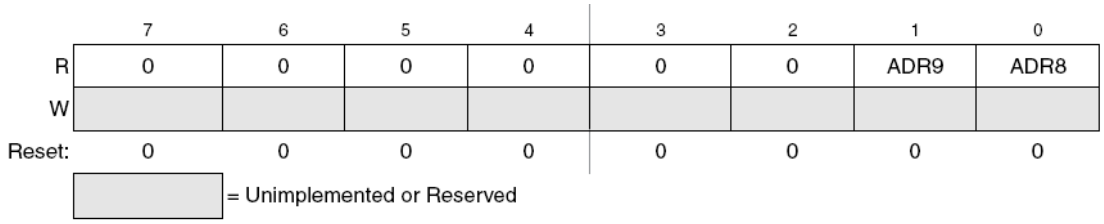
9.3.2 状态和控制寄存器 2（ADCSC2）

	7	6	5	4	3	2	1	0
R	ADACT	ADTRG	ACFE	ACFGT	0	0	R^1	R^1
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

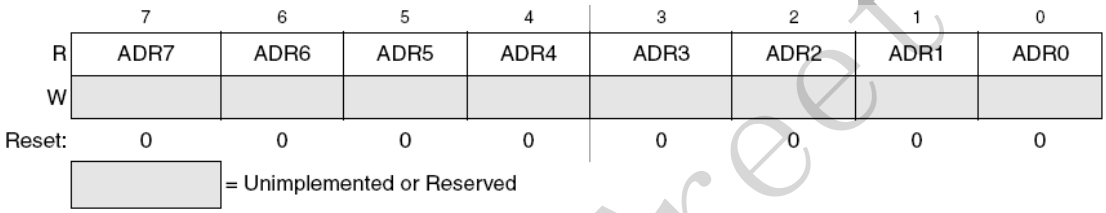
7 ADACT	转换活动标志：当开始转换时该位置位，当转换完成或中止该位清零 0：转换未进行中；1：转换进行中
6 ADTRG	转换触发器选择：软件触发（写 ADC1SC1）或硬件触发 0：软件触发；1：硬件触发
5 ACFE	比较功能允许位 0：禁止比较功能；1：允许比较功能
4 ACFGT	比较功能触发方式选择位 0：当输入小于比较值时比较器触发；1：当输入大于或等于比较值时触发

9.3.3 数据转换结果高（ADCRH）



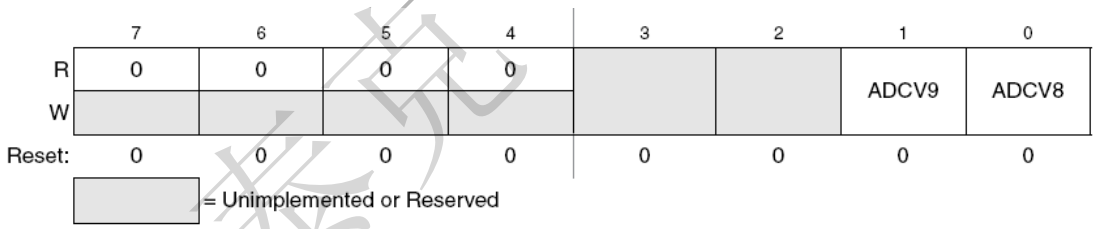
ADCRH 包含 10 位转换结果的最高两位。当配置为 8 位转换模式时，ADR8 和 ADR9 等于 0。ADCRH 在每次转换完成后均更新，除非比较功能使能而且比较条件未满足。在 10 位模式下，读取 ADCRH 将阻止下一次的转换结果转移到结果寄存器中，直到 ADCRL 被读取。如果下一次转换已经完成，而 ADCRL 未读取那么这次转换结果丢失。在 8 位模式下，ADCRL 没有互锁机制。当 MODE 位发生改变，那么在 ADCRH 中的任何数据将无效。

9.3.4 数据结果寄存器低（ADCRL）

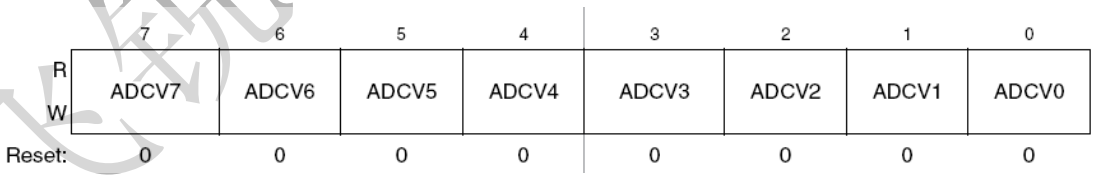


该寄存器功能可参照 ADCRH。

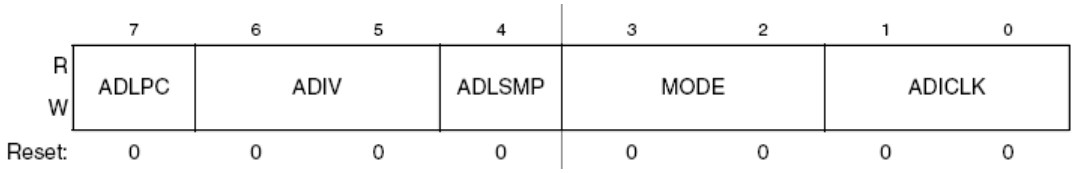
9.3.5 比较值高位寄存器（ADCCVH）



9.3.6 比较值低位寄存器（ADCCVL）



9.3.7 配置寄存器（ADCCFG）



7 ADLPC	低功耗配置-控制转换器转换速度和功耗 0：高速转换；1：低速转换

6：5 ADIV	时钟分频系数-用于选择分频系数从而产生 ADCK，见下表
4 ADLSMP	抽样间隔配置：通过调整抽样间隔以允许高阻抗输入可以被精确抽样或低阻抗输入时加快转换速度。当选择的为连续转换模式并且对转换速度要求不高的前提下，较长的抽样间隔可以降低整体功耗。 0：短抽样间隔；1：长抽样间隔
3：2 MODE	转换模式选择：见下表
1：0 ADICLK	输入时钟选择：见下表

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

表 9.2 时钟分频选择表

ADICLK	Selected Clock Source
00	Bus clock
01	Bus clock divided by 2
10	Alternate clock (ALTCLK)
11	Asynchronous clock (ADACK)

表 9.3 时钟源选择位

MODE	Mode Description
00	8-bit conversion (N=8)
01	Reserved
10	10-bit conversion (N=10)
11	Reserved

表 9.4 模式选择表

9.3.8 引脚控制寄存器 1(APCTL1)

	7	6	5	4	3	2	1	0
R W	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
Reset:	0	0	0	0	0	0	0	0

7 ADPC7	ADC 引脚控制位 7 0：AD7 引脚作为 ADC 的输入 1：AD7 引脚不作为 ADC 的输入
6	ADC 引脚控制位 6

ADPC6	0：AD6 引脚作为 ADC 的输入 1：AD6 引脚不作为 ADC 的输入
5 ADPC5	ADC 引脚控制位 5 0：AD5 引脚作为 ADC 的输入 1：AD5 引脚不作为 ADC 的输入
4 ADPC4	ADC 引脚控制位 4 0：AD4 引脚作为 ADC 的输入 1：AD4 引脚不作为 ADC 的输入
3 ADPC3	ADC 引脚控制位 3 0：AD3 引脚作为 ADC 的输入 1：AD3 引脚不作为 ADC 的输入
2 ADPC2	ADC 引脚控制位 2 0：AD2 引脚作为 ADC 的输入 1：AD2 引脚不作为 ADC 的输入
1 ADPC1	ADC 引脚控制位 1 0：AD1 引脚作为 ADC 的输入 1：AD1 引脚不作为 ADC 的输入
0 ADPC0	ADC 引脚控制位 0 0：AD0 引脚作为 ADC 的输入 1：AD0 引脚不作为 ADC 的输入

9.3.9 引脚控制寄存器 2（APCTL2）

	7	6	5	4	3	2	1	0
R W	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
Reset:	0	0	0	0	0	0	0	0

7 ADPC15	ADC 引脚控制位 15 0：AD15 引脚作为 ADC 的输入 1：AD15 引脚不作为 ADC 的输入
6 ADPC14	ADC 引脚控制位 14 0：AD14 引脚作为 ADC 的输入 1：AD14 引脚不作为 ADC 的输入
5 ADPC13	ADC 引脚控制位 13 0：AD13 引脚作为 ADC 的输入 1：AD13 引脚不作为 ADC 的输入
4 ADPC12	ADC 引脚控制位 12 0：AD12 引脚作为 ADC 的输入 1：AD12 引脚不作为 ADC 的输入
3 ADPC11	ADC 引脚控制位 11 0：AD11 引脚作为 ADC 的输入 1：AD11 引脚不作为 ADC 的输入
2 ADPC10	ADC 引脚控制位 10 0：AD10 引脚作为 ADC 的输入 1：AD10 引脚不作为 ADC 的输入
1	ADC 引脚控制位 9

ADPC9	0：AD9 引脚作为 ADC 的输入 1：AD9 引脚不作为 ADC 的输入
0 ADPC8	ADC 引脚控制位 8 0：AD8 引脚作为 ADC 的输入 1：AD8 引脚不作为 ADC 的输入

9.3.10 引脚控制寄存器 3（APCTL3）

	7	6	5	4	3	2	1	0
R W	ADPC23	ADPC22	ADPC21	ADPC20	ADPC19	ADPC18	ADPC17	ADPC16
Reset:	0	0	0	0	0	0	0	0

7 ADPC23	ADC 引脚控制位 23 0：AD23 引脚作为 ADC 的输入 1：AD23 引脚不作为 ADC 的输入
6 ADPC22	ADC 引脚控制位 22 0：AD22 引脚作为 ADC 的输入 1：AD22 引脚不作为 ADC 的输入
5 ADPC21	ADC 引脚控制位 21 0：AD21 引脚作为 ADC 的输入 1：AD21 引脚不作为 ADC 的输入
4 ADPC20	ADC 引脚控制位 20 0：AD20 引脚作为 ADC 的输入 1：AD20 引脚不作为 ADC 的输入
3 ADPC19	ADC 引脚控制位 19 0：AD19 引脚作为 ADC 的输入 1：AD19 引脚不作为 ADC 的输入
2 ADPC18	ADC 引脚控制位 18 0：AD18 引脚作为 ADC 的输入 1：AD18 引脚不作为 ADC 的输入
1 ADPC17	ADC 引脚控制位 17 0：AD17 引脚作为 ADC 的输入 1：AD17 引脚不作为 ADC 的输入
0 ADPC16	ADC 引脚控制位 16 0：AD16 引脚作为 ADC 的输入 1：AD16 引脚不作为 ADC 的输入

9.4 功能描述

在复位或 ADCH 各位都为 1 时，ADC 模块关闭。当本次转换完成，而另一个转换未启动时，模块处于空闲状态。模块处于空闲状态时为最低功耗。

ADC 模块可以根据软件选择的通道执行模数转换。所选择的通道通过逐次逼进算法将结果转换为 11 位数据。在 8 位模式，会将结果转换为 9 位数据结果。

当转换完成，结果存放在数据寄存器（ADCRH 和 ADCRL）。10 位模式下，10 位结果放置在 ADCRH 和 ADCRL。8 位模式下，结果存放在 ADCRL。转换完成标志 COCO 在转换完成后置位，可以产生中断。

ADC 模块还具有自动将转换结果与比较寄存器值进行比较功能。

9.4.1 时钟选择和分频控制

可以选择的时钟源有四种，时钟源经过分频产生转换时钟。

1) 总线时钟，也就是软件执行的时钟。复位之后的默认时钟源
2) 总线时钟 2 分频
3) ALTCLK
4) 异步时钟 ADACK。该时钟由 ADC 模块内部产生。选择该时钟源，那么当 MCU 处于等待或 STOP3 模式时，ADC 仍工作。

无论选择哪种时钟源，但应符合 ADCK 范围。如果时钟太低，ADC 也不会工作。时钟太快，那么时钟必须分频到合适的值。分频值可以为 1，2，4，8。

9.4.2 输入选择和引脚控制

寄存器 APCTL3，APCTL2，APCTL1 控制 ADC 输入通道。

9.4.3 硬件触发转换

ADC 模块可采用异步触发－ADHWT，当 ADTRG=1 时。ADC 模块在 ADHWT 的上升沿开始转换。转换过程中，再次出现的上升沿将被忽略。在连续转换模式下，只有第一次启动转换的上升沿被检测。

9.4.4 转换控制

MODE 位决定转换结果为 10 位或 8 位模式。可以软件或硬件启动转换。而且 ADC 模块可以配置为低功耗，长抽样间隔时间，连续转换，自动比较转换结果的功能。

9.4.4.1 启动转换

- 在软件触发操作模式下，写 ADCSC1 寄存器将启动转换。
- 在硬件触发模式下，ADHWT 触发事件
- 在连续转换模式下，当转换结果转移到数据寄存器中。

在连续转换模式中，在完成当前转换完成后，自动启动下一次转换。在软件启动转换模式下，在写 ADCSC1 之后连续转换开始，直到被中止。在硬件触发模式下，硬件触发事件发生后开始连续转换，直到被中止。

9.4.4.2 转换完成

当转换结果转移到数据结果寄存器后，转换完成。此时 COCO 置位，用于表征转换完成。如果 AIEN=1，则产生中断。

在读取 ADCRH 和 ADCRL 寄存器过程中，使用阻止机制以避免新的转换结果覆盖先前的数据。当阻止机制发生，那么数据转移被阻止，COCO 不会被置位，新的转换结果丢失。

在单次转换，比较功能使能，但比较条件未满足时，阻止机制不产生作用，ADC 操作中止。在其他操作模式下，当数据传输被阻止，下一次转换仍会启动。

在单次转换模式下，必须等到转换完成后，再去读取数据寄存器中的值。

9.4.4.3 转换中止

当发生下面的操作将中止转换：

1） MCU 复位

2） MCU 进入 STOP 模式并且 ADACK=0

3） 写 ADCSC1 寄存器（当前转换被中止，启动一个新的转换，前提 ADCH 不全为 1）

4） 写 ADCSC2,ADCCFG,ADCCVH,ADCCVL。表明操作模式发生变化，当前转换因此无效。

当转换被中止，ADCRH 和 ADCRL 中的寄存器的内容不会改变，而是保持为最近上一次成功转换的结果。如果是由于复位中止的转换，那么 ADCRH 和 ADCRL 为复位默认值。

9.4.4.4 功耗控制

ADC 模块处于空闲状态，直到开始转换。如果 ADACK 被选择为转换时钟，ADACK 时钟发生器使能。

设置 ADLPC 位，可降低 ADC 功耗。

9.4.4.5 总转换时间

总的转换时间依赖于抽样时间（ADLSMP 位决定），MCU 总线频率，转换模式（8 位或 10 位），和转换时钟（f_{ADCK}）。

当模块进入活动状态，开始抽样模拟输入。ADLSMP 用于选择短的抽样时间和长的抽样时间。当抽样完成，转换器将与输入通道隔离开，执行逐次逼近算法。该算法执行完成后，转换结果转移到 ADCRH 和 ADCRL。

如果总线频率小于 f_{ADCK} 频率时，在连续转换模式下，采用短抽样的话，不能保证精确抽样。当总线频率小于 f_{ADCK} 的 1/11，在连续转换模式下，采用长抽样时间，也不能保证精确抽样。

Conversion Type	ADICLK	ADLSMP	Max Total Conversion Time
Single or first continuous 8-bit	0x, 10	0	20 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 µs + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	0	5 µs + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 µs + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	1	5 µs + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

表 9.5 总转换时间和控制条件

最大的转换时间由所选时钟源和分频系数决定。例如，10 位模式，总线频率作为输入时钟

源，分频系数为 1，总线频率 8MHz。单次转换时间为：

$$\text{Conversion time} = \frac{23 \text{ ADCK cyc}}{8 \text{ MHz}/1} + \frac{5 \text{ bus cyc}}{8 \text{ MHz}} = 3.5 \ \mu\text{s}$$

$$\text{Number of bus cycles} = 3.5 \ \mu\text{s} \times 8 \text{ MHz} = 28 \text{ cycles}$$

9.4.5 自动比较功能

比较功能可用于检查上限或下限。输入抽样后，转换完后，结果与 ADCCVH 和 ADCCVL 中的值比较。当 ACFGT=1，转换结果大于或等于比较值，COCO 置位。当 ACFGT=0，转换结果小于比较值，COCO 置位。

一旦转换完成，此时比较功能使能，如何比较条件未满足，那么 COCO 不会置位，没有数据转移到结果寄存器中。

注：比较功能可以用于处于 WAIT 或 STOP3 模式下 MCU 监测通道电压。当比较条件满足，ADC 中断可以唤醒 MCU。

9.4.6 WAIT 模式下的操作

MCU 执行 WAIT 指令，MCU 可以进入低功耗模式，在这种模式下时钟源仍运行，故可以快速恢复。当 MCU 进入 WAIT 模式后，转换正进行中，那么它将继续转换直到完成。在 WAIT 模式下，可以通过硬件触发方式启动转换或仍继续连续转换模式。

总线时钟，总线时钟的二分频，ADACK 在 WAIT 模式下仍可作为转换时钟源。如果选择 ALTCLK 作为转换时钟的话，应查看每种 MCU 的 ALTCLK 定义。

转换完成，COCO 位设置为 1，产生 ADC 中断，可以唤醒 WAIT 模式下的 MCU。

9.4.7 MCU 在 STOP3 模式下

9.4.7.1 STOP3 模式下 ADACK 关闭

如果 ADACK 未被选择为转换时钟，那么执行 STOP 指令将中断当前转换，ADC 进入空闲状态。ADCRH 和 ADCRL 的内容不会被改变。从 STOP3 模式唤醒后，软件或硬件触发器用于恢复转换。

9.4.7.2 STOP3 模式下，ADACK 运行

如果选择 ADACK 作为转换时钟，那么在 STOP3 模式中，ADC 仍继续运行。为确保 ADC 的运行，MCU 稳压器仍处于活动状态。

进入 STOP3 模式后，正在运行的转换将继续转换，直到完成。通过硬件触发或连续转换将启动 ADC 转换。

转换完成后，COCO 位置位，如果 AIEN=1，将产生中断，MCU 从 STOP3 模式唤醒。

9.4.8 STOP1 和 STOP2 模式下

在这两种模式下，ADC 将自动关闭。当从 STOP1 或 STOP2 模式唤醒后，所有的模块寄存器将恢复到其复位值，因此 ADC 模块必须重新使能和配置。

9.5 初始化信息

9.5.1 ADC 模块初始化例子

9.5.1.1 初始化过程

典型初始化过程如下：

1. 更新配置寄存器 ADCCFG 选择时钟源和分频系数产生内部时钟，ADCK。同时该寄存器用于选择抽样时间和低功耗配置。
2. 更新 ADCSC2 选择触发方式和比较器是否使能
3. 更新 ADCSC1 选择单次转换还是连续转换，打开或关闭转换完成中断以及输入通道。

9.5.1.2 实例

该例子中，中断功能使能，单次 10 位转换，低功耗，长抽样时间，输入通道 1，ADCK 时钟由总线时钟驱动，分频系数为 1。

ADCCFG=0X98

BIT7 ADLPC 1 配置为低功耗模式

BIT6:5 ADIV 00 设置 ADCK 为输入时钟的 1 分频

BIT4 ADLSMP 1 配置长抽样时间

BIT3:2 MODE 10 设置为 10 位转换

BIT1:0 ADICLK 00 选择总线时钟作为输入时钟源

ADCSC2=0X00

BIT7 ADACT 0 标识转换进行中

BIT6 ADTRG 0 软件触发方式

BIT5 ACFE 0 关闭比较功能

BIT4 ACFGT 0 未使用

BIT3:2 00

BIT1:0 00

ADCSC1=0X41

BIT7 COCO 0

BIT6 AIEN 1 中断使能

BIT5 ADCO 0 单次转换

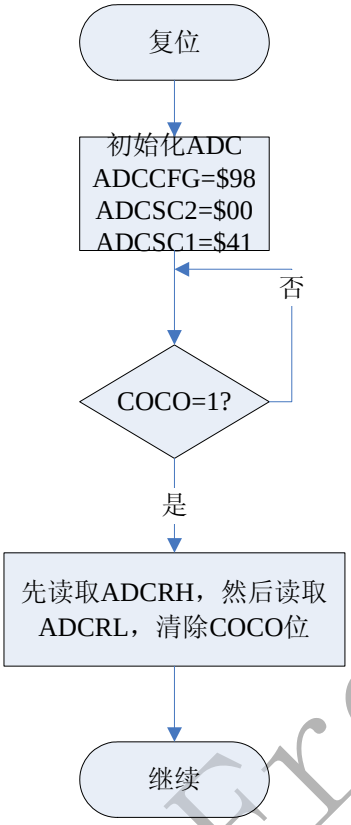
BIT4:0 ADCH 00001 通道 1 作为模拟输入

ADCRH/L=0xxx

ADCCVH/L=0xxx

APCTL1=0X02

APCTL2=0X00



9.6 应用信息

9.6.1 外部引脚

9.6.1.1 模拟电源引脚

大多少情况，模拟电源和数字电源独立，但模拟电源和数字电源通过一点共地。

9.6.1.2 模拟参考引脚

该两个引脚应加 0.1u 的电容，并且靠近引脚。

9.6.1.3 模拟输入引脚

外部模拟输入与 I/O 引脚复用。建议引脚作为模拟输入时，该引脚的控制寄存器一直置位。尽量靠近输入引脚与 V_{SSA} 之间一个高频 0.01uF 电容，可以提高 ADC 性能。

为了能够正确转换，输入电压必须在 V_{REFH} 和 V_{REFL} 之间。如果输入信号等于或大于 VREFH，那么数据寄存器将为全 1。如果输入信号等于或小于 VREFL，数据寄存器将全为 0。

9.6.2 误差源

9.6.2.1 抽样错误

为了得到正确的转换结果，输入模拟信号必须被正确的抽样。假如输入最大阻值接近 7k，输入电容接近 5.5pF，如果外部模拟信号源的阻抗在 5k 以下，那么最小的抽样

窗口可以满足精度要求。

如果信号源阻抗较大或要求精度高，那么可以采用较长的抽样时间。

9.6.2.2 引脚漏电误差

如果外部模拟信号源内阻很大，那么 I/O 引脚的漏电会产生转换误差。

如果该误差不能被应用所接受，那么应使 R_{AS} 小于 V_{DDAD} / (2^N * I_{LEAK})，N 等于 8 或 10。

9.6.2.3 噪声引入的误差

在抽样或转换过程中，系统噪声会影响转换的精度。

- V_{REFH} 和 V_{REFL} 之间放置一个 0.1uf 的低等效串连电阻电容
- V_{DDAD} 和 V_{SSAD} 之间放置一个 0.1uf 的低等效串连电阻电容
- 如果 V_{DDAD} 通过电感与主电源隔离，那么 V_{DDAD} 和 V_{SSAD} 之间放置一个 1uF 电容
- 如果 V_{SSAD} 和 V_{REFL} 连接到 V_{SS} 那么应在一个地平面的一个免受干扰的点。
- 在转换期间，没有 I/O 进行开关操作。

9.6.2.4 位宽和量化误差

8 位或 10 位 ADC 有±1/2LSB 的误差。

$$1LSB = (V_{REFH} - V_{REFL}) / 2^N$$

9.6.2.5 线性误差

线性误差有几种形式：零点误差；满刻度误差；微分非线性误差；积分非线性误差；总未调误差。

第十章 内部时钟源

10.1 简介

ICS 模块为 MCU 提供时钟。模块包括一个锁频环 FLL。模块为 MCU 系统时钟提供 FLL 时钟，内部或外部参考时钟。

总新频率为 ICSOUT 频率的一半。

10.1.1 模块配置

STOP 模式下内部参考时钟使能，那么电压稳压器也必须在 STOP 模式下使能。

10.1.2 特性

- FLL 精度可调

－内部 32kHz 参考频率分辨率 0.2%

－内部 32kHz 参考频率全温度全工作电压偏差 2%

- 内部或外部参考时钟可达 5MHz
- 内部参考时钟有 9 位调整位
- 内部或外部参考时钟可选择为 MCU 的系统时钟
- 可选的分频系数

－1，2，4，8

－BDC 时钟的分频系数为常数 2

- 低功耗的外部参考时钟
- 复位之后 FLL 使能内部模式自动使能

10.1.3ICS 内部原理框图

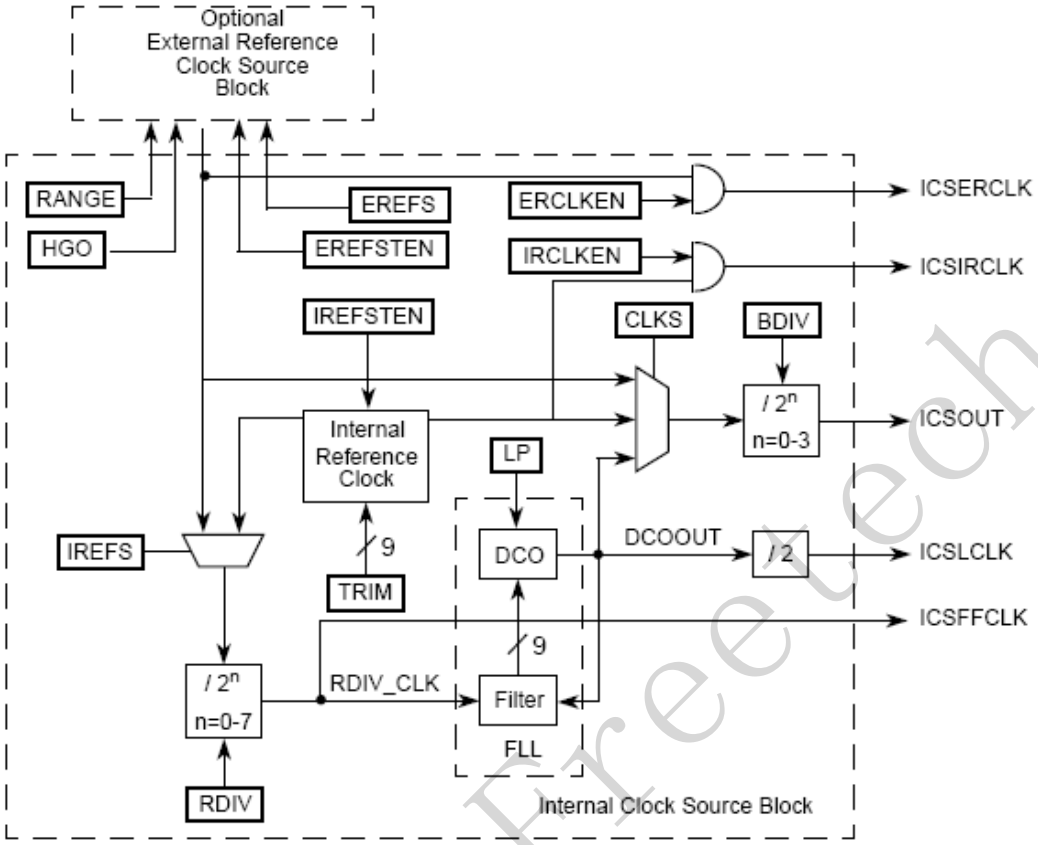


图 10.1 ICS 内部原理框图

10.1.4 操作模式

● 模式 1－FLL 使能内部参考时钟（FEI）

该模式下，ICS 的 FLL 倍频内部参考时钟。

BDC 时钟有 FLL 提供。

● 模式 2－FLL 使能外部参考时钟（FEE）

● FLL 旁路，内部参考时钟模式（FBI）

FLL 由内部参考时钟控制，但 ICS 为系统提供的时钟是由内部参考时钟驱动的。BDC 时钟由 FLL 提供。

● FLL 旁路，内部参考时钟，低功耗模式（FBILP）

FLL 关闭。ICS 为系统提供的时钟由内部参考时钟驱动。BDC 无时钟驱动。

● FLL 旁路，外部参考时钟模式（FBE）

FLL 由外部参考时钟控制，但 ICS 为系统提供的时钟是由外部参考时钟驱动的。BDC 时钟由 FLL 提供

● FLL 旁路，外部参考时钟，低功耗模式（FBELP）

FLL 关闭。ICS 为系统提供的时钟是由外部参考时钟驱动的。BDC 无时钟。

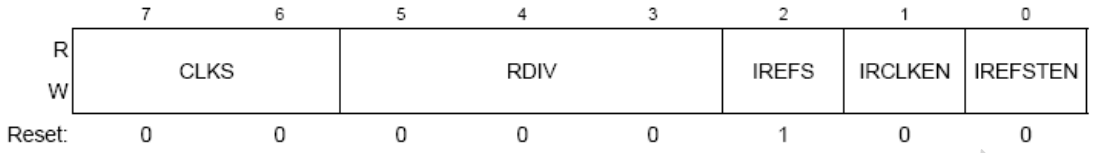
● STOP 模式

FLL 关闭。ICS 不为系统提供时钟。

10.2 外部信号描述

10.3 寄存器定义

10.3.1 ICS 控制寄存器 1（ICSC1）



7:6 CLKS	时钟源选择：选择总线时钟源 00：FLL 输出 01：内部参考时钟 10：外部参考时钟 11：预留
5：3 RDIV	参考时钟分频系数：FLL 参考频率范围 31.25kHz～39.0625kHz 000：分频系数 1 001：分频系数 2 010：分频系数 4 011：分频系数 8 100：分频系数 16 101：分频系数 32 110：分频系数 64 111：分频系数 128
2 IREFS	内部参考时钟源选择：FLL 参考源选择位 1：选择内部参考时钟源 0：选择外部参考时钟源
1 IRCLKEN	内部参考时钟使能 1：ICSIRCLK激活 0：ICSIRCLK关闭
0 IREFSTEN	内部参考时钟在 STOP 模式下运行控制位 1：STOP 模式下内部参考时钟仍运行 0：STOP 模式下内部参考时钟关闭

10.3.2 ICS 控制寄存器 2(ICSC2)

7:6 BDIV	总线分频系数：根据 CLKS 位所选择的时钟源，对该时钟源进行分频，从而得到总线频率。

	00：分频系数 1 01：分频系数 2（复位后的默认值） 10：分频系数 4 11：分频系数 8
5 RANGE	频率范围选择 1：外接低频晶振 0：外接高频晶振
4 HGO	高增益晶振选择控制位 1：高增益 0：低功耗
3 LP	低功耗选择 1：FLL 在旁路模式下，FLL 关闭（BDM 活动状态除外） 0：FLL 旁路模式下，FLL 仍运行
2 EREFS	外部参考选择控制位 1：晶振 0：外部时钟源
1 ERCLKEN	外部参考使能 1：ICSERCLK 由外部参考时钟驱动 0：ICSERCLK 关闭
0 EREFSTEN	STOP 模式下外部参考使能控制位 1：STOP 模式下仍运行 0：STOP 模式下关闭

10.3.3 ICS 调整寄存器（ICSTRM）

	7	6	5	4	3	2	1	0
R W	TRIM							
POR:	1	0	0	0	0	0	0	0
Reset:	U	U	U	U	U	U	U	U

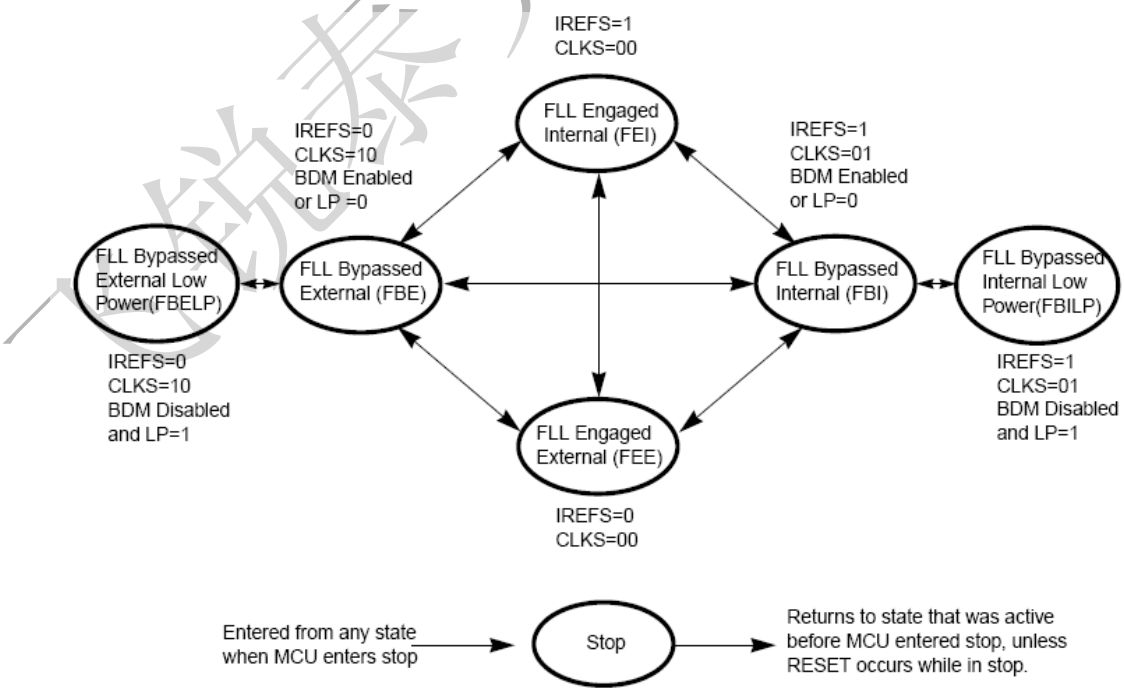
7:0 **TRIM**	TRIM 位控制内部参考时钟的频率。该寄存器的值增加，则周期增加，频率降低；反之，周期缩短，频率提高。

10.3.4 ICS 状态和控制寄存器（ICSSC）

	7	6	5	4	3	2	1	0
R	0	0	0	IREFST	CLKST		OSCINIT	FTRIM
W								
POR:	0	0	0	1	0	0	0	0
Reset:	0	0	0	1	0	0	0	U

7：5	预留
4 IREFST	内部参考状态 0：参考时钟为外部时钟 1：参考时钟为内部时钟
3：2 CLKST	时钟模式状态：表明当前时钟模式 00：FLL 输出 01：FLL 使能，内部参考时钟 10：FLL 旁路，外部参考时钟 11：预留
1	OSC 初始化完成标志
0	ICS 细调位：该位置位将增加内部参考时钟周期，该位清零将减小内部参考时钟周期。

10.4 功能描述



10.4.1 FEI

FEI 为默认的工作模式，当下面所有条件都满足时进入该模式：

- CLKS 被写为 00
- IREFS 被写为 1
- RDIV 被写，从而使参考时钟在 31.25kHz~39.0625kHz 之间

  该模式下，ICSOUT 时钟由 FLL 时钟驱动，FLL 会将内部参考时钟倍频 1024 倍。ICSLCLK 可用于 BDC 通信，内部参考时钟使能。

10.4.2 FEE

当下面所有条件都满足时进入该模式：

- CLKS 被写为 00
- IREFS 被写为 0
- RDIV 被写，从而使参考时钟在 31.25kHz~39.0625kHz 之间

  该模式下，ICSOUT 时钟由 FLL 时钟驱动，FLL 会将内部参考时钟倍频 1024 倍。ICSLCLK 可用于 BDC 通信，外部参考时钟使能。

10.4.3 FBI

当下面所有条件都满足时进入该模式：

- CLKS 被写为 01
- IREFS 被写为 1
- BDM 激活或 LP 位被写入 0

  该模式下，ICSOUT 时钟由内部参考时钟驱动，FLL 会将内部参考时钟倍频 1024 倍。ICSLCLK 可用于 BDC 通信，内部参考时钟使能。

10.4.4 FBILP

当下面所有条件都满足时进入该模式：

- CLKS 被写为 01
- IREFS 被写为 1
- BDM 关闭或 LP 位被写入 1

  该模式下，ICSOUT 时钟由内部参考时钟驱动，FLL 关闭。ICSLCLK 不再用于 BDC 通信，内部参考时钟使能。

10.4.5 FBE

当下面所有条件都满足时进入该模式：

- CLKS 被写为 10
- IREFS 被写为 0
- BDM 激活或 LP 位被写入 0

  该模式下，ICSOUT 时钟由外部参考时钟驱动，FLL 会将外部参考时钟倍频 1024 倍。ICSLCLK 可用于 BDC 通信，外部参考时钟使能。

10.4.6 FBELP

当下面所有条件都满足时进入该模式：

- CLKS 被写为 10
- IREFS 被写为 0
- BDM 关闭或 LP 位被写入 1

  该模式下，ICSOUT 时钟由外部参考时钟驱动，FLL 关闭。ICSLCLK 不再用于 BDC 通信，外部参考时钟使能。

10.4.7 STOP

当执行 STOP 指令，ICS 进入 STOP 模式。

10.4.8 模式切换

FEI 和 FEE 模式之间进行切换，IREFS 位在任何时刻都可以改变，但 RDIV 位也应同时改变，保持参考频率在 31.25kHz～390.625kHz 之间。

第十一章 IIC

11.1 介绍

11.1.1 模块配置

IICPS in SOPT1	Port Pin for SDA	Port Pin for SCL
0 (default)	PTA2	PTA3
1	PTB6	PTB7

11.1.2 特征

- 与 IIC 标准总线兼容
- 多主机操作
- 软件可编程选择为 64 个不同时钟频率
- 软件可选的确认位
- 数据传输产生中断
- 仲裁失败中断
- 地址匹配中断
- 开始和停止信号产生/检测
- 重开始信号产生
- 确认位产生/检测
- 总线忙检测
- 兼容 10 位地址

11.1.3 运行模式

- 正常运行模式－基本的运行模式。
- 等待模式－模块继续运行
- 停止模式－在 STOP3 模式下，IIC 不再活动，但 STOP 指令不会影响 IIC 寄存器状态。STOP1 和 STOP2 模式下将复位寄存器内容。

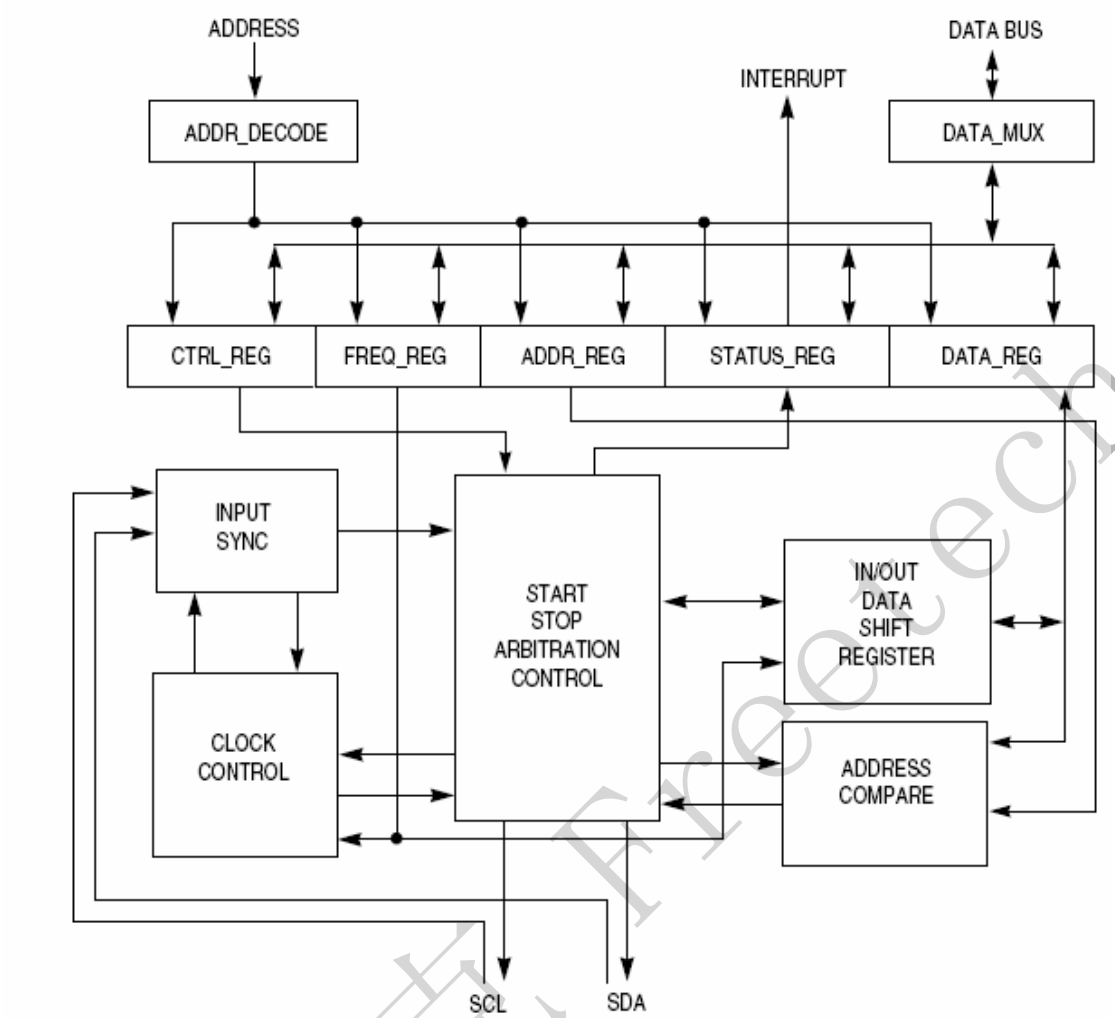


图 13.1 IIC 原理框图

11.2 外部信号描述

11.2.1 SCL－串行时钟线

SCL 为双向的串行时钟线

11.2.2 SDA－串行数据线

SDA 为双向的串行数据线

11.3 寄存器定义

11.3.1 IIC 地址寄存器（IIC1A）

7：1 ADDR[7:1]	从机地址－该 7 位为 IIC 模块的从机地址

11.3.2 IIC 频率分频寄存器（IIC1F）

7：6 MULT	IIC 倍频因子－用于产生波特率 00 mul=01 01 mul=02 10 mul=04 11 未用
5：0 ICR	IIC 时钟波特率 IIC 波特率＝总线频率/（mul * SCL divider） SDA 维持时间定义为从 SCL（IIC 时钟）的下降沿到 SDA 发生改变的延时时间。ICR 用于决定 SDA 的维持时间值。 IIC 维持时间＝总线周期*SDA 维持时间值。 例如： 总线频率＝8MHZ MULT=01(mul＝2) 要是 IIC 波特率＝100kbps IIC 波特率＝总线频率/(mul * SCL 分频系数) 100000＝8000000/(2 * SCL 分频系数) SCL 分频系数＝40 此时，假如 ICR=0X0B SDA 维持时间＝总线周期 * SDA 维持时间值 SDA 维持时间＝1/8000000 * 9 = 1.125us。

ICR (hex)	SCL Divider	SDA Hold Value
00	20	7
01	22	7
02	24	8
03	26	8
04	28	9
05	30	9
06	34	10
07	40	10
08	28	7
09	32	7
0A	36	9
0B	40	9
0C	44	11
0D	48	11
0E	56	13
0F	68	13
10	48	9
11	56	9
12	64	13
13	72	13
14	80	17
15	88	17
16	104	21
17	128	21
18	80	9
19	96	9
1A	112	17
1B	128	17
1C	144	25
1D	160	25
1E	192	33
1F	240	33

ICR (hex)	SCL Divider	SDA Hold Value
20	160	17
21	192	17
22	224	33
23	256	33
24	288	49
25	320	49
26	384	65
27	480	65
28	320	33
29	384	33
2A	448	65
2B	512	65
2C	576	97
2D	640	97
2E	768	129
2F	960	129
30	640	65
31	768	65
32	896	129
33	1024	129
34	1152	193
35	1280	193
36	1536	257
37	1920	257
38	1280	129
39	1536	129
3A	1792	257
3B	2048	257
3C	2304	385
3D	2560	385
3E	3072	513
3F	3840	513

表 11.1 IIC 分频与维持值

11.3.3 IIC 控制寄存器（IICC1）

	7	6	5	4	3	2	1	0
R	IICEN	IICIE	MST	TX	TXAK	0	0	0
W						RSTA		
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

7 IICEN	IIC 模块允许控制位 0：IIC 模块关闭；1：IIC 模块使能
6 IICIE	IIC 中断允许控制位 0：禁止 IIC 中断；1：允许 IIC 中断
5 MST	主设备模式选择－当总线上产生一个起始信号，MST 位由 0 变为 1，则选择为主设备；在总线产生一个停止信号，该位由 1 变为 0，则操作模式由

	主变为从 0：从机模式 1：主机模式
4 TX	发送模式选择－TX 位选择主和从传送方向。在主模式下，该位被置位根据传送要求。因此在寻址周期，该位应始终为高。当作为从设备被寻址时，软件根据 SRW 位置位。 0：接收；1：发送
3 TXAK	发送确认使能位 0：接收完一字节数据后，确认信号发送到总线 1：不发送确认信号
2 RSTA	重复启动－假设它是当前主设备，向该位写 1 将产生一个重复开始信号。读该位始终为低。而试图在错误时间发送重复开始信号将产生仲裁失败错误。

11.3.4 IIC 状态寄存器（IICS）

	7	6	5	4	3	2	1	0
R	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
W								
Reset	1	0	0	0	0	0	0	0

= Unimplemented or Reserved

7 TCF	传输完成标志－当完成一字节传送后该位置位。注意该位只在向 IIC 模块发送数据或接收数据期间或紧随其后有效。在接收模式时，读取 IIC1D 寄存器或在发送模式时，写 IIC1D 寄存器可清除该位。 0：正在传送；1：传送完成
6 IAAS	作为从设备被寻址－当寻址地址与从设备的可编程地址匹配时，该位置位。 0：未被寻址到；1：作为从设备被寻址到
5 BUSY	总线忙标志－无论主设备模式还是从设备模式，该位表明总线的状态。当起始信号被检测到，BUSY 位置位，检测到 STOP 信号该位清除。 0：总线空闲；1：总线忙
4 ARBL	仲裁失败标志－清除该位方法：向该位写 1。 0：标准总线操作；1：仲裁失败
2 SRW	从设备读/写标志位 0：从设备接收；主设备写从设备； 1：从设备发送；主设备读取从设备数据
1 IICIF	IIC 中断标志 0：无中断事件；1：有中断； 该位必须通过软件清除；方法为向该为写 1 一字节传送完毕 地址匹配 总裁丢失
0 RXAK	接收确认状态位－当 RXAK 位为低，表明在总线传送完一个字节后收到一个确认信号。 0：接收到一个确认信号；1：未收到确认信号

11.3.5 IIC 数据寄存器（IICD）

7：0 DATA	数据－在主设备发送模式，当数据写入到 IIC1D，数据传送开始；最高位先被发送。在主设备接收模式，读取该寄存器开始接收下一个字节数据

当主机由发送模式转为接收模式前，不应对 IIC1D 进行读取，避免错误的启动一次主机接收数据的传送。

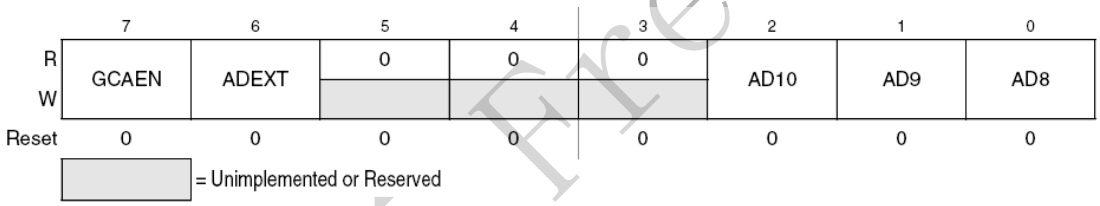
在从机模式，当地址匹配后，IIC1D 作为数据发送和读取源。

注意在 IIC1C 寄存器中的 TX 位必须正确反映主设备模式和从设备模式下传送方向。例如 IIC 配置为主设备发送，但想进行主设备接收过程，那么读取 IIC1D 并不会启动接收。

当 IIC 配置为主设备接收或从设备接收模式，读取 IIC1D 将返回上一次接收到的字节。IIC1D 不会总线上发送的每一字节也不会校验已经写入到 IIC1D 数据的准确性。

在主机发送模式，主机应先将 MST 由 0 变为 1，发送 7 位从机地址。

11.3.6 IIC 控制寄存器 2（IICC2）



7 GCAEN	General call 寻址使能控制位 0：禁止 1：允许
6 ADEXT	地址扩展控制位 0：7 位地址 1：10 位地址
2:0 AD[10:8]	从机设备的高 3 位。

11.4 功能描述

11.4.1 IIC 协议

IIC 总线使用串行数据线 SDA 和串行时钟线 SCL 进行数据传输。所有设备连接到总线上，漏极开路或集电极开路。使用外部电阻上拉，线与功能。

正常情况下，标准通信由 4 部分组成：

主机发送启动信号

主机发送从设备地址

主机发送数据

停止信号

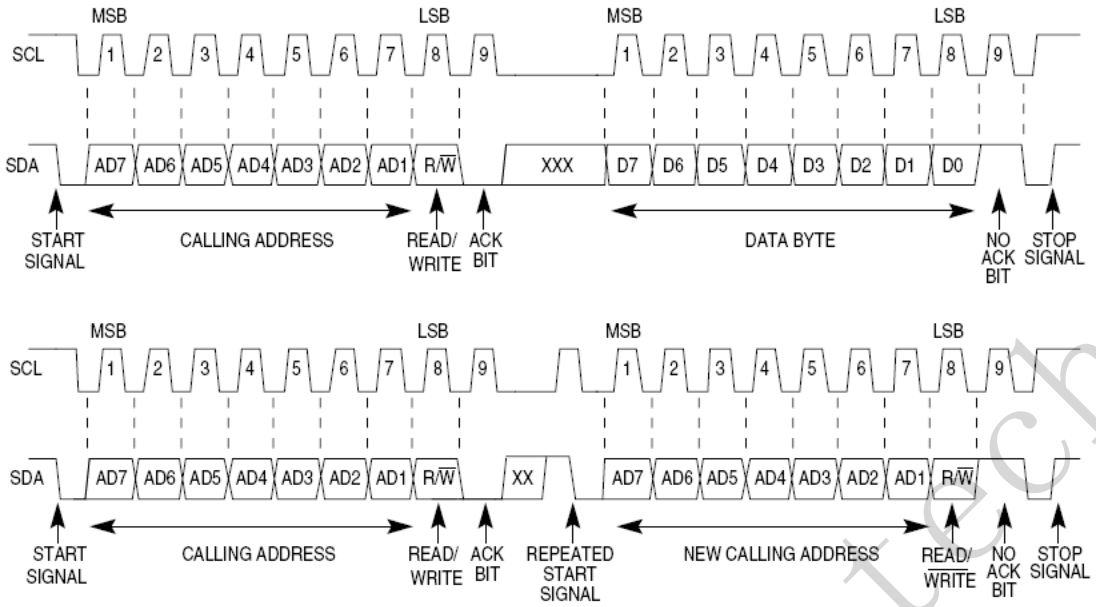


图 11.2 IIC 传送协议

11.4.1.1 启动信号

当总线空闲时，SCL 和 SDA 线都为高电平。主设备通过发送 START 信号启动通信。启动信号定义为：SCL 为高，SDA 由高变为低。该信号通知一个新数据传输开始，唤醒总线上所有其他空闲状态的设备。

11.4.1.2 从设备地址发送

起始信号后的第一个字节是从设备地址。其中高 7 位为地址，第 8 位为 R/W 位。R/W 位告诉从设备数据传送方向。

1＝主机读：从设备发送数据到主设备

0＝主机写：主设备发送数据到从设备

只有和该地址匹配的从设备作出响应，从设备会发一个确认信号。即从设备在第 9 个时钟将 SDA 拉低。

在系统中，从设备地址是唯一的。如果 IIC 模块为主设备，那么它不能发送一个等于它作为从设备时的地址。IIC 模块不能同时作为主设备和从设备。然而，如果在一个寻址期间内，仲裁失败，那么 IIC 模块将转变为从机模式。那么其他主机如果寻址正确的话，就可以进行通信。

11.4.1.3 数据传输

每个数据长度位 8 位。数据只能在 SCL 为低期间改变，而在高电平期间必须维持。每一位数据对应一个时钟脉冲。先发送最高位。每字节数据后有一个确认位，该位由接收方发出。即在第 9 个时钟周期内，将 SDA 线拉低。所以一个完整字节的传输需要 9 个时钟脉冲。

如果在主设备的第 9 个脉冲时间，接收器没有确认。那么 SDA 线必须由从设备维持高电平。主设备会由于未得到确认而产生中断。

如果在主设备接收时，当接收完一字节数据后，主机未发确认位，那么从设备认为传送数据完成，释放 SDA 线。

中断数据传输方法：主设备进行一下任意一种操作：

- 通过产生一个停止信号放弃总线。
- 通过产生一个重复的起始信号开始一个新的寻址。

11.4.1.4 停止信号

主设备通过发送停止信号来释放总线。然而主设备可以产生一个开始信号，然后发寻址命令，这种操作称为重复开始。

停止信号的定义：SCL 为 1，SDA 由低变高。

在数据传输过程中，从设备发出确认信号后，主机可以产生停止信号，该停止信号告诉从机释放总线。

13.4.1.5 重复开始信号

重复开始信号定义：一个新的开始信号（前一个开始信号后未发停止信号）。通常用于在不释放总线的情况下主设备与另一个从设备通信或者同一个从设备（不同模式下）。

11.4.1.6 仲裁过程

IIC 是真正的多主总线，因此允许多个主设备连接在总线上。如果两个以上的主设备试图同时控制总线时，一个时钟同步过程决定总线时钟，低电平持续时间等于时钟信号中的最长的低电平时间，而高电平时间等于最短的高电平时间。

多主机竞争的相对优先级由数据仲裁过程确定，在仲裁工程中，如果一个主机发送 1，而其他主机发送 0，那么该主机仲裁失败。仲裁失败的主机立刻转变为从机接收模式，停止驱动 SDA。在这种情况下，主设备转变为从设备不会产生停止信号。

11.4.4.7 时钟同步

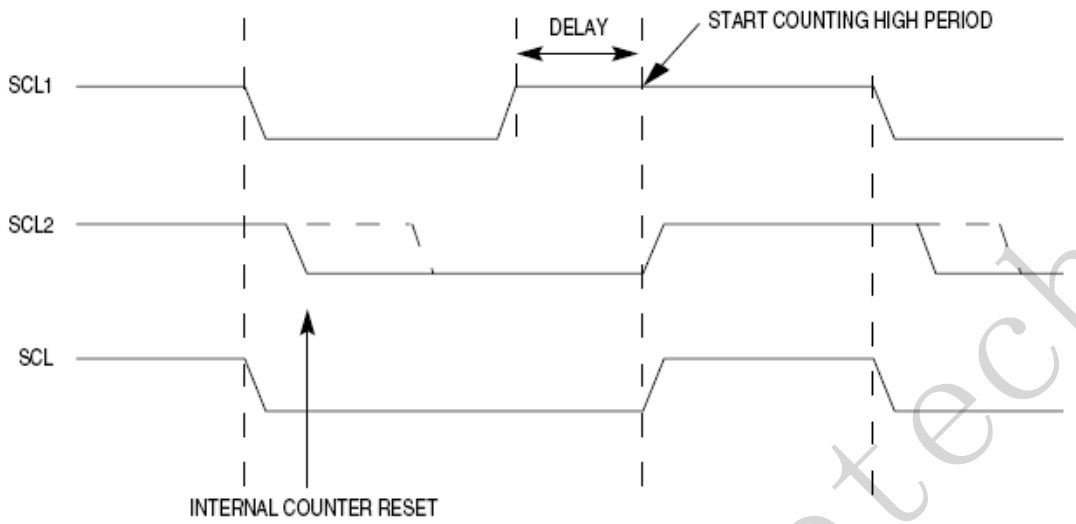


图 11.3 时钟同步图

因为 SCL 线具有线与逻辑功能，SCL 线上一个由高到低转变会影响到总线上所有设备。当某个设备开始计数它们的低电平间隔，此时该设备将使 SCL 维持低电平，直到遇到时钟高电平。然而，该设备时钟由低变高时，其他设备时钟仍处于低电平状态。那么它不能将 SCL 驱动到高电平。因此，同步时钟 SCL 会被低电平最长的设备控制。

时间较短的低电平设备在该期间进入高电平等待状态。当所有的设备都结束它们的低电平时间，那么同步时钟 SCL 线释放，被拉高。同理，第一个完成它高电平的设备将 SCL 线先拉低。

11.4.1.8 握手

时钟同步机制可以用于数据传输的握手。完成一字节传送之后，从设备可能维持 SCL 为低，在这种情况下，它将暂停总线时钟，迫使主机时钟时钟进入等待模式直到从设备释放 SCL 线。

11.4.1.9 时钟扩展

从设备利用时钟同步机制可用于降低传送波特率。在主设备将 SCL 驱动低后，从设备可以驱动 SCL 低并持续一个必需的时间然后释放。如果从设备驱动 SCL 为低的时间大于主设备的，结果 SCL 总线信号的低电平时间加长。

11.4.2 10 位寻址地址

11.4.2.1 主机（发送）寻址从机（接收）

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Data	A	...	Data	A/A	P

如上图所示，可能不止一个从设备向主机发送确认信号 A1。而向主机发送确认信号 A2 的从设备是唯一的。

11.4.2.2 主机（接收）寻址从机（发送）

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Sr	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 1	A3	Data	A	...	Data	A	P

上图所示，A2 之前的过程同主机寻址从机一样。在主机发送完 Sr 信号之后，已经被寻址到的从机会核实 Sr 信号之后的 11110+AD10+AD9 和 R/W 位，如果该从机比较 11110+AD10+AD9 与自己的一致，并且 R/W 位为 1。那么从机就处于主动发送状态，一直到它接收到 STOP 信号活 Sr 信号。

11.4.3 General Call 寻址

如果 GCAEN=1，那么当 general call 寻址地址匹配 MCU 的从机地址时，IIC 模块会做出响应，并且作为从机（接收），寻址周期之后 IAAS=1。当寻址地址的第一个字节传送后，软件必须读取 IICD 寄存器，并确定地址匹配是自己的从机地址或 general call 地址。如果 IICD 为“00”，那么匹配的为 general call 地址。如果 GCAEN 为被清除，那么 IIC 模块会忽律 general call 寻址。

11.5 复位

复位之后，IIC 关闭。IIC 不能产生 MCU 复位。

11.6 中断

Interrupt Source	Status	Flag	Local Enable
Complete 1-byte transfer	TCF	IICIF	IICIE
Match of received calling address	IAAS	IICIF	IICIE
Arbitration Lost	ARBL	IICIF	IICIE

图 13.2 IIC 中断表

11.6.1 字节传送中断

在第 9 个时钟的下降沿，TCF 位置位，表明该字节传送完毕。

11.6.2 地址检测中断

当寻址地址与 IIC 地址配合或 GCAEN＝1 并且接收到 general call 地址时，IAAS 位置位。如果 IICIE 使能，那么 CPU 被中断。此时 CPU 必须检查 SRW 位，设置它相应的 Tx 模式。

11.6.3 仲裁失败中断

IIC 为真正的多主总线。如果两个或更多主机试图同时控制总线，那么通过一个数据仲裁的过程来解决竞争。下列情况下会发生仲裁失败：

- 在地址或数据发送周期期间，当主设备驱动 SDA 为高，SDA 被抽样为低。
- 在数据接收周期的确认位期间，当主设备驱动 SDA 为高，SDA 被抽样为低。
- 当总线忙时，试图发起始信号

在从设备模式下，发一个重复开始信号

第十二章 模计数器

12.1 简介

MTIM 为一个简单的 8 位计数器，有几种软件可选的时钟源和一个中断。

MTIM 中心器件为一个 8 位计数器；可运行在自由计数或模计数方式。计数溢出中断可产生定期中断。

12.1.1MTIM 配置信息

MTIM 可由外部时钟信号驱动－TCLK，该引脚与 PTA0 复用。

12.1.2 特点

- 8 位向上计数器

－自由计数或 8 位模计数

－溢出中断

－计数复位位－TRST

－计数停止位 TSTP

- 4 种软件可选的时钟源

－系统总线时钟

－固定频率时钟

－外部时钟

- 9 种分频值

－1，2，4，8，16，32，64，128，256

12.1.3 运行模式

12.1.3.1WAIT 模式下

可以运行在该模式下。

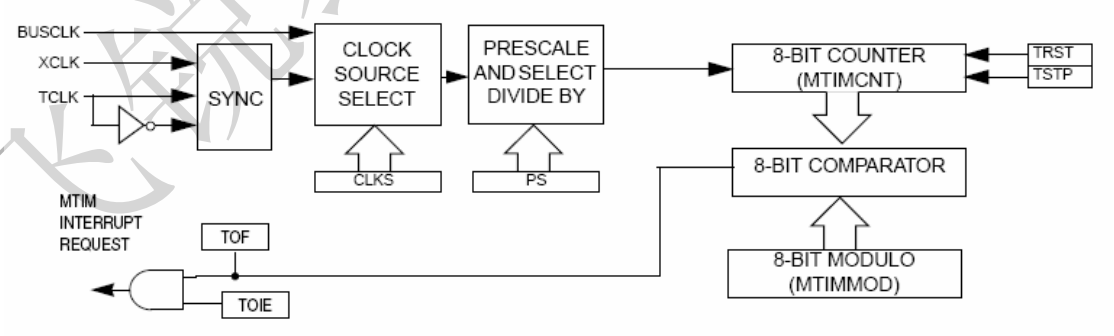
12.1.3.2STOP 模式下

该模式下 MTIM 不再运行。

12.1.3.3BDM 模式下

该模式下，MTIM 挂起。

12.1.4 框图

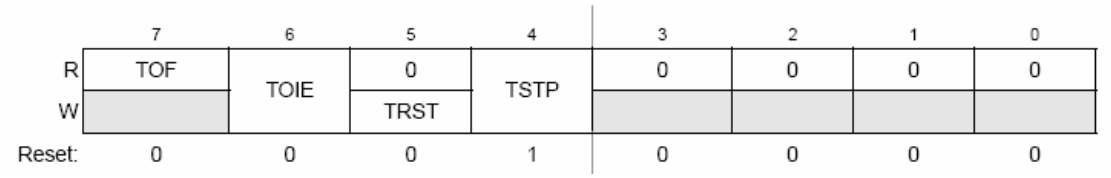


12.2 外接信号

TCLK：MTIM 外部时钟源

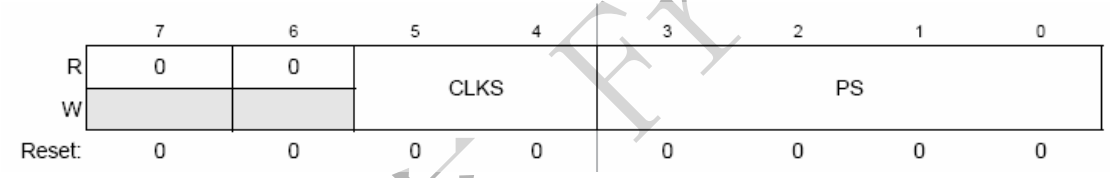
12.3 寄存器定义

12.3.1MTIM 状态和控制寄存器（MTIMSC）



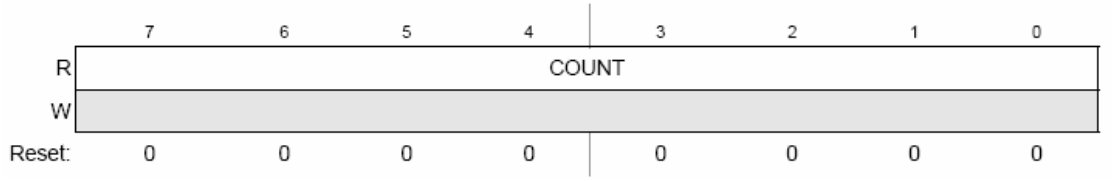
7 TOF	MTIM 溢出标志－当 MTIM 计数寄存器溢出变为 0X00 清除 TOF 方法：读 MTIMSC 的状态和控制寄存器，然后向 TOF 位写 0。 向 TRST 位写 1 或向 MTIMMOD 寄存器写任意值都可以将 TOF 位清零。 0：未发生溢出；1：溢出。
6 TOIE	MTIM 溢出中断允许位：可读写 0：不允许溢出中断；1：溢出中断允许
5 TRST	MTIM 计数器复位控制位 0：无影响 1：MTIM 计数值复位为 0X00
4 TSTP	MTIM 计数停止位 0：MTIM 计数 1：MTIM 停止计数
3:0	预留

12.3.2 MTIM 时钟配置寄存器（MTIMCLK）



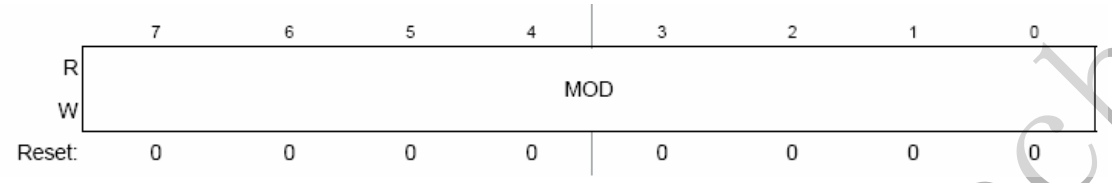
7:6	预留
5：4 CLKS	时钟源选择： 00：总线时钟 BUSCLK 01：固定频率时钟 XCLK 10：外部时钟源 TCK 引脚，上升沿 11：外部时钟源 TCK 引脚，下降沿
3：0 PS	时钟源分频器 0000：MTIM 时钟源/1 0001: MTIM 时钟源/2 0010:MTIM 时钟源/4 0011: MTIM 时钟源/8 0100: MTIM 时钟源/16 0101: MTIM 时钟源/32 0110: MTIM 时钟源/64 0111: MTIM 时钟源/128 1000:MTIM 时钟源/256 其他值：MTIM 时钟源/256

12.3.3MTIM 计数寄存器（MTIMCNT）



7:0 COUNT	MTIM 计数寄存器

12.3.4MTIM 模寄存器（MTIMMOD）



7:0 MOD	MTIM 模寄存器 该寄存器值为 0X00，运行于自由计数模式 向该寄存器写操作将复位 COUNT 寄存器的值为 0，并且清除 TOF 位。 复位后该寄存器为 0X00。

12.4 功能描述

MTIM 有一个 8 位向上计数器和一个模寄存器，时钟源可选，9 种分频选择。MTIM 有 3 种运行模式：停止，自由计数，模计数模式。复位之后，计数处于停止状态。当模寄存器的值为 0 时，计数器运行于自由计数模式；当模寄存器的值不为 0 时，计数器运行于模计数模式。

MCU 复位之后，MTIM 处于停止状态，计数器寄存器的值为 0X00，模寄存器的值为 0x00。总线频率为默认的驱动时钟，分频系数为 1。只要向 MTIMSC 寄存器的 TSTP 位清零，MTIM 就可以开始自由计数。

MTIM 有 4 种时钟源，如果 MTIM 正在运行，而时钟源发生改变，MTIM 仍会继续计数。当分频系数发生改变，MTIM 也会继续计数。

第十三章 实时时钟计数器

13.1 介绍

RTC 包含一个 8 位计数器和一个 8 位比较器，时钟分频器，2 种时钟源选择和一个可编程的中断。

13.1.1 特点

－8 位递增计数器

－1-kHz 的内部低功耗振荡器

－外部时钟（ERCLK）

－32-kHz 的内部时钟（IRCLK）

13.1.2 运行模式

13.1.2.1WAIT 模式下

该模式下，RTC 可以运行，可作为 MCU 的唤醒源

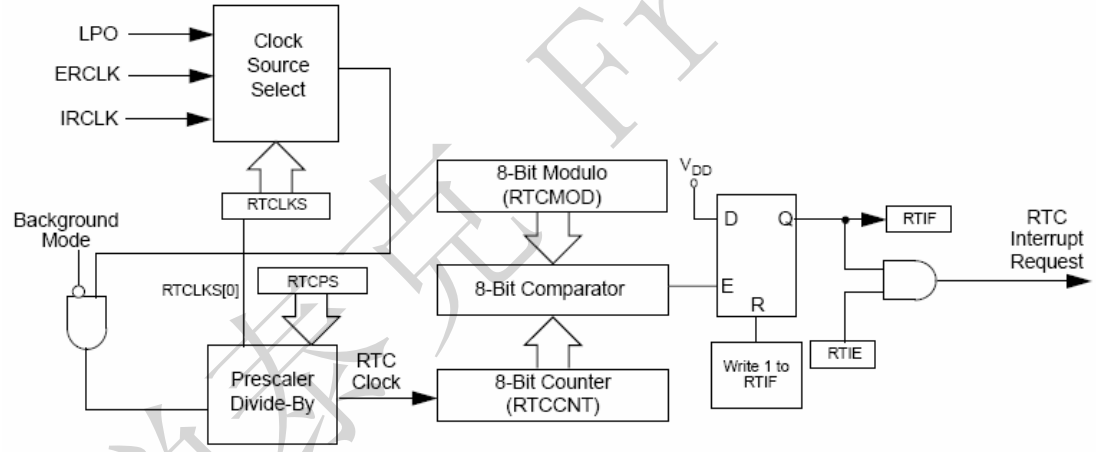
13.2.1.2STOP 模式下

可在 STOP2 或 STOP3 模式下运行。

13.1.2.3BDM 模式下

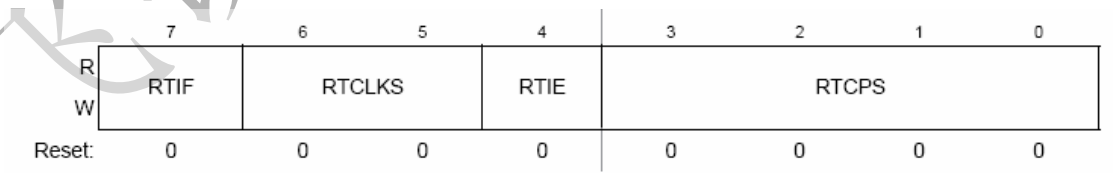
该模式下，RTC 挂起

13.1.3 原理框图



13.3 寄存器

13.3.1RTC 状态和控制寄存器（RTCSC）

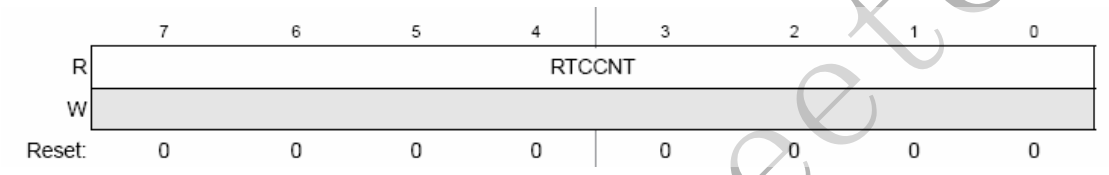


7 RTIF	实时时钟中断标志位 0：RTC 计数值未达到模寄存器中设定的值 1：RTC 计数值等于模寄存器中的设定值
6：5 RTCLKS	实时时钟时钟源选择 00：内部 1kHz 低功耗振荡器 01：外部时钟源

	1x：内部时钟 IRCLK
4 RTIE	实时时钟中断允许位 0：实时时钟中断禁止 1：实时时钟中断允许
3：0 RTCPS	实时时钟分频器：见下表

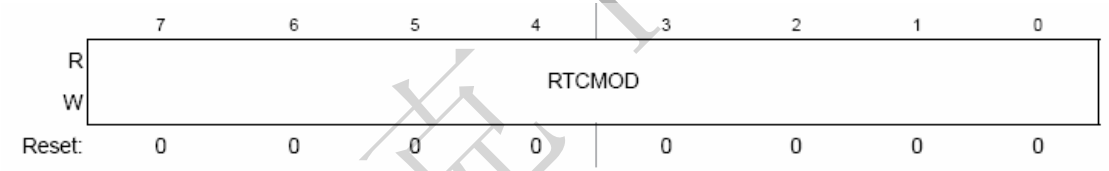
RTCLKS[0]	RTCPS															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Off	2^3	2^5	2^6	2^7	2^8	2^9	2^{10}	1	2	2^2	10	2^4	10^2	$5x10^2$	10^3
1	Off	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	10^3	$2x10^3$	$5x10^3$	10^4	$2x10^4$	$5x10^4$	10^5	$2x10^5$

13.3.2RTC 计数寄存器（RTCCNT）



7:0 RTCCNT	RTC 计数寄存器

13.3.3RTC 模寄存器



7:0 RTCMOD	RTC 模寄存器

13.4 功能描述

RTC 包括一个 8 位递增计数器和一个 8 位的模寄存器，一个时钟选择器，一个分频器。复位之后，RTC 停止，模寄存器的值为 0，分频器关闭。内部 1kHz 时钟为默认时钟。

有 3 种时钟源选择，当时钟源发生改变时，分频器和 RTCCNT 计数复位到 0X00。

RTCPS	1-kHz Internal Clock (RTCLKS = 00)	1-MHz External Clock (RTCLKS = 01)	32-kHz Internal Clock (RTCLKS = 10)	32-kHz Internal Clock (RTCLKS = 11)
0000	Off	Off	Off	Off
0001	8 ms	1.024 ms	250 µs	32 ms
0010	32 ms	2.048 ms	1 ms	64 ms
0011	64 ms	4.096 ms	2 ms	128 ms
0100	128 ms	8.192 ms	4 ms	256 ms
0101	256 ms	16.4 ms	8 ms	512 ms
0110	512 ms	32.8 ms	16 ms	1.024 s
0111	1.024 s	65.5 ms	32 ms	2.048 s
1000	1 ms	1 ms	31.25 µs	31.25 ms
1001	2 ms	2 ms	62.5 µs	62.5 ms
1010	4 ms	5 ms	125 µs	156.25 ms
1011	10 ms	10 ms	312.5 µs	312.5 ms
1100	16 ms	20 ms	0.5 ms	0.625 s
1101	0.1 s	50 ms	3.125 ms	1.5625 s
1110	0.5 s	0.1 s	15.625 ms	3.125 s
1111	1 s	0.2 s	31.25 ms	6.25 s

第十四章 串行通信接口 SCI

14.1 介绍

14.1.1 特点

- 全双工，标准的NRZ格式
- 双缓冲发送和接收，可单独使能
- 可编程波特率
- 中断或轮询操作

－发送数据寄存器空，发送完成
－接收数据寄存器满
－接收超限，奇偶校验错误，帧错误，噪声错误
－空闲接收检测
－接收引脚上有效信号
－BREAK检测，支持LIN总线

- 硬件奇偶校验位生成和校验
- 数据长度8位或9位
- 接收器被空闲线或地址标记唤醒
- 可配置位13位break字符产生或11位break字符
- 可选择的发送器输出极性

14.1.2 操作模式

- 8位和9位数据模式
- 停止模式
- 单线模式
- 循环模式

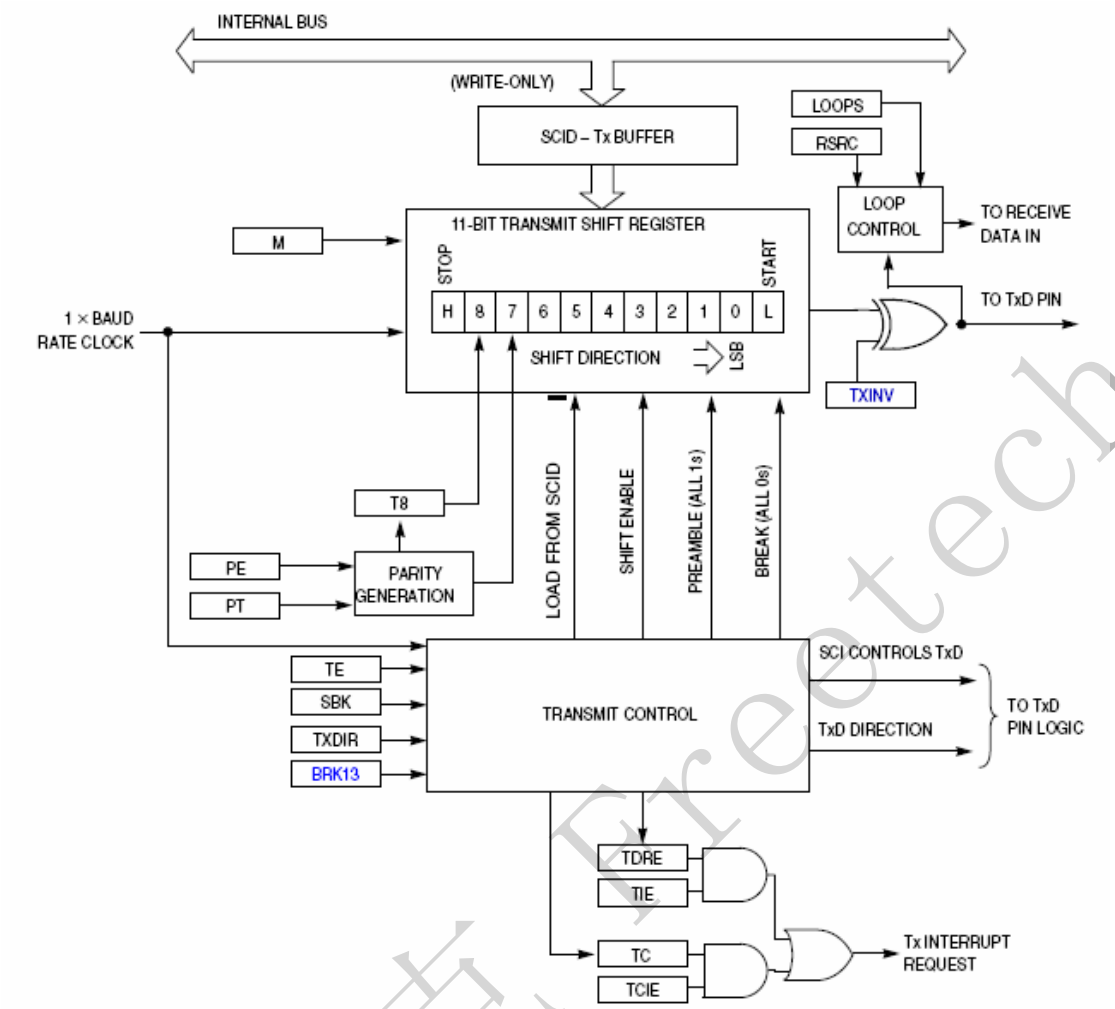


图14.1 SCI发送原理框图

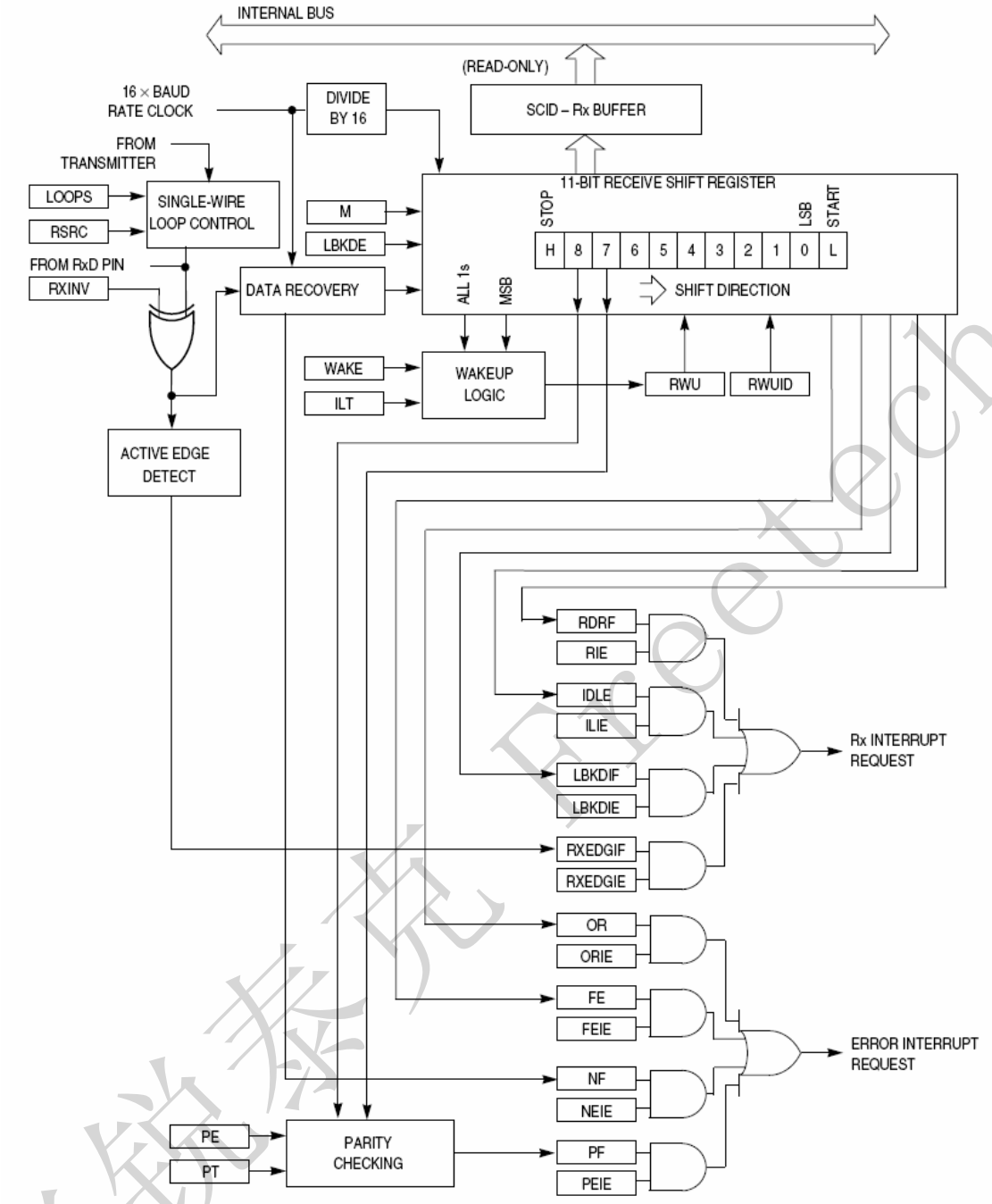


图14.2 SCI接收原理框图

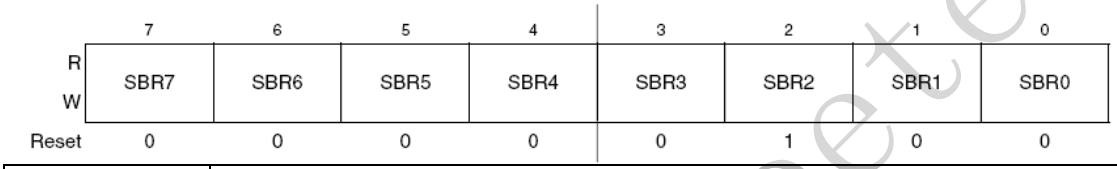
14.2 寄存器定义

14.2.1 SCI波特率寄存器（SCIxBDH,SCIxBDL）

这一对寄存器控制 SCI 的波特率。为了更新 13 位波特率设置位[SBR12:SBR0]，首先向 SCIxBDH 寄存器写入新值，然后再写入 SCIxBDL。只有 SCIxBDL 寄存器写完成，新的波特率才起作用。

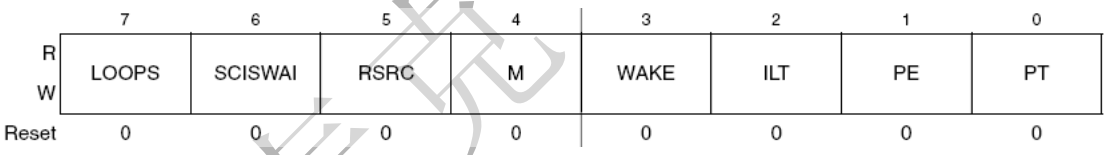
复位之后，SCIxBDL 寄存器中的内容非 0，但波特率发生器未工作(设置 SCIxC2 寄存器中 RE 或 TE 为 1，使能接收或发送功能)。

7 LBKDIE	LIN 总线 break 检测中断允许控制位 0：硬件中断不允许；1：硬件中断允许当 LBKDIF＝1
6 RXEDGIE	RxD 输入有限沿中断允许位控制位 0：硬件中断不允许；1：硬件中断允许当 RXEDGIF=1
4:0 SBR[12:8]	波特率控制位高 5 位 当 BR＝0，SCI 波特率发生器关闭进而降低功耗；当 BR=1~8191，SCI 波特率＝BUSCLK/(16 X BR)



7:0 SBR[7:0]	波特率控制位低 8 位 当 BR＝0，SCI 波特率发生器关闭进而降低功耗；当 BR=1~8191，SCI 波特率＝BUSCLK/(16 X BR)

14.2.2 SCI控制寄存器1（SCIxC1）



7 LOOPS	循环模式选择位 0：正常操作模式－RxD 和 TxD 使用单独的引脚 1：循环模式或单线模式（在 MCU 内部，发送器的输出直接连接到接收器的输入）
6 SCISWAI	WAIT 模式下 SCI 时钟控制位 0：SCI 仍正常运行，此时 SCI 的中断能够唤醒 CPU；1：SCI 时钟停止
5 RSRC	接收源选择位：此位仅在 LOOPS=1 的模式下有意义。 0：内部短接，不经过外部引脚；1：RxD 和 TxD 引脚短接
4 M	字符长度选择位 0：正常方式－起始位＋8 位数据（低位在前）＋停止位 1：起始位＋8 位数据（低位在前）＋第 9 位数据＋停止位
3 WAKE	唤醒条件选择位：只在接收器中有效 0：空闲线唤醒；1：地址唤醒
2 ILT	空闲方式选择位： 0：空闲字符位从"起始位"开始计数；1：空闲字符位从"停止位"开始计数；
1 PE	奇偶校验允许位 0：硬件不产生或校验奇偶校验位；1：硬件产生并校验奇偶校验位

0 PT	校验类型选择位 0：偶校验；1：奇校验

14.2.3 SCI控制寄存器2（SCIxC2）

	7	6	5	4	3	2	1	0
R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

7 TIE	SCI 发送中断允许位：与 TDRE 配合使用 0：不允许中断；1：允许中断
6 TCIE	SCI 发送完成中断允许位：与 TC 配合使用 0：不允许中断；1：允许中断
5 RIE	SCI 接收中断允许位：与 RDRF 配合使用 0：不允许中断；1：允许中断
4 ILIE	接收线空闲中断允许位：与 IDLE 配合使用 0：不允许中断；1：允许中断
3 TE	发送允许位 0：禁止发送；1：允许发送 通常，TE=1，TxD 引脚作为 SCI 系统的输出； LOOPS=1,RSRC=0：TxD 引脚作为通用 I/O； LOOPS=1,RSRC=1：TXDIR 控制单线 SCI 通信线方向； 在发送期间，TE=0->TE=1 的操作可用于产生一个空闲字符；
2 RE	接收允许位 0：SCI 接收禁止；1：SCI 接收允许
1 RWU	接收唤醒控制位 接收器唤醒条件： 1）空闲线唤醒：WAKE=0，消息之间的空闲字符 2）地址唤醒：WAKE=1 0:正常的 SCI 接收操作；1：SCI 接收器处于待命状态，等待唤醒条件
0 SBK	发送 break－向 SBK 位写 1，接着再向该位写 0，其结果是将 BREAK 字符列队在发送数据流中。通过设置 SBK13 位，可以设置 BREAK 字符为 10/11 位或 13/14 位。 0：正常发送操作 1：队列一个 BREAK 字符，等待发送

14.2.4 SCI状态寄存器1（SCIxS1）

	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
Reset	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

7 TDRE	发送数据寄存器空标志－当发送数据寄存器的内容转移到发送移位器后，该位置位。读取 SCIxS1，然后向 SCIxD 写数据可清除该标志 0：发送数据寄存器非空；1：发送数据寄存器空
6 TC	发送完成标志－当 TDRE=1 并且无其他数据，BREAK 字符，前导码发送时置位 0：正在发送；1：发送完成 读取 SCIxS1，然后进行一下任意一种操作 TC 位将自动清除： ● 向 SCIxD 寄存器写入新的数据 ● 通过向 TE 写 0，接着向 TE 写 1，列队一个前导符 ● 通过向 SBK 写 1，列队一个 BREAK 符
5 RDRF	接收数据寄存器满标志－当一个字符从接收移位器转移到接收数据寄存器后，该位置位。读取 SCIxS1，然后读取 SCIxD 可清除该标志位 0：接收数据寄存器空；1：接收数据寄存器满
4 IDLE	接收线空闲标志－当 SCI 接收线空闲时间持续一个字符时间该位置位。 当 ILT=0，接收器从开始位计空闲位时间。因此如果接收到的字符全 1，那么这些位的时间加上停止位的时间是接收器检测空闲线的时间。 当 ILT=1，接收器从停止位开始计时空闲位时间。因此停止位和刚发送字符中的任意高电平位的时间不作为接收器检测空闲线的时间。 读取SCIxS1，然后读取SCIxD可以清除该标志位。该位被清除以后，不会再次置位，直到接收到一个新的字符，并且RDRF=1。IDLE只置位一次，即使接收线处于空闲状态时间大于空闲线时间。 0：未检测到空闲线 1：检测到空闲线
3 OR	接收器溢出标志－当一个新的串行字符准备转移到接收数据寄存器时，但以前的接收到的字符未从 SCIxD 读取。 清除该位方法：读取 SCIxS1 寄存器，然后读取 SCIxD。 0：未溢出；1：溢出
2 NF	噪声标志位：对起始位进行 7 次抽样，对数据位和停止位进行 3 次抽样。如果任意一次抽样与其他抽样不同，NF 将被置位。 清除该位方法：读取 SCIxS1，然后读取 SCIxD 0：未检测到噪声；1：SCID 数据中有噪声。
1 FE	帧错误标志－在停止位的位置上检测到 0，将置位该位。 清除该位方法：读取 SCIxS1，然后读取 SCIxD。 0：未检测到帧错误，但不能保证该帧是正确的；1：帧错误
0 PF	奇偶校验错误： 清除该位方法：先读取 SCIxS1，然后读取 SCIxD 0：无校验错误；1：校验错误

11.2.5 SCI状态寄存器2（SCIxS2）

	7	6	5	4	3	2	1	0
R	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

7 LBKDIF	LIN 总线 BREAK 字符检测中断标志－当 LIN 总线 BREAK 检测电路使能，并且 LIN 总线 BREAK 字符被检测到，该位置位。 向该位写 1，清除该位 0：未检测到 LIN 总线 BREAK 字符 1：检测到 LIN 总线 BREAK 字符
6 RXEDGIF	RxD 引脚有效沿中断标志－如果 RXINV=0，那么当引脚上出现下降沿时该位置位。如果 RXINV=1，那么当引脚上出现上升沿时该位置位。 向该位写 1 将清除该位。 0：引脚未检测到有效边沿；1：该引脚检测到有效边沿
4 RXINV	接收数据反向控制位：接收器有效 0：接收数据不必反向 1：接收数据反向
3 RWUID	接收唤醒空闲检测－该位控制空闲字符是否设置 IDLE 位。 0：在接收待命期间，检测到一个空闲字符不会置 IDLE 位 1：在接收待命期间，检测到一个空闲字符会置 IDLE 位
2 BRK13	BREAK 字符长度：发送器有效 0：BREAK 字符长度为 10 位时间 1：BREAK 字符长度为 13 位时间
1 LBKDE	LIN 总线 BREAK 字符检测使能 0：BREAK 字符按 10 位时间检测 1：BREAK 字符按 11 位时间检测
0 RAF	接收器活动标志－当接收器检测到一个有效的起始位时该位置位。当接收器检测到一个空闲线状态，该位自动清除。在 MCU 进入 STOP 模式之前，可以查询该位，从而确定 SCI 接收器是否正在接收字符。 0：SCI 接收器待命，等待起始位 1：SCI 接收器处于活动状态

当在 LIN 系统中使用内部振荡器时，有必要提高 BREAK 检测门限。
建议在 LIN 总线通信时，LBKDE 设置为 1，避免对数据 0x00 错误的当作 BREAK 字符。

14.2.6 SCI控制寄存器3（SCIxC3）

7 R8	接收器的第 9 位数据－当 SIC 配置为 9 位数据模式时，R8 认为是第 9 位数据。当读取这 9 位数据时，必须先读取 R8，然后读取 SCIxD。
6 T8	发送器的第 9 位发送数据－当 SCI 配置为 9 位数据模式时，T8 被认为第 9 位发送数据。先向 T8 写数据，然后向 SCIxD 寄存器写数据。如果 T8 没必要更新，那么每次就没必要写。
5 TXDIR	TxD 引脚方向－当 SCI 配置为单线半双工模式（LOOPS=RSRC=1）,该位将决定 TxD 引脚的方向。 0：TxD 引脚在单线模式下作为输入；1：TxD 引脚在单线模式下作为输出。
4 TXINV	发送数据反向控制位： 0：发送数据不反向；1：发送数据反向
3 ORIE	接收器中断溢出允许位：与 OR 位配合使用 0：不允许；1：允许中断
2 NEIE	噪声错误中断允许位：与 NF 位配合使用 0：不允许；1：允许

1 FEIE	帧错误中断允许位：与 FE 位配合使用 0：不允许；1：允许中断
0 PEIE	奇偶校验错误中断允许位：与 PF 位配合使 0：不允许；1：允许

14.2.7 SCI数据寄存器

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

该寄存器实际上是两个单独的寄存器。读取时，返回只读接收数据缓冲器的内容；写时，修改的是只写发送数据缓冲器。读和写该寄存器通常用于清除一些标志位。

14.3 功能描述

14.3.1 波特率发生器

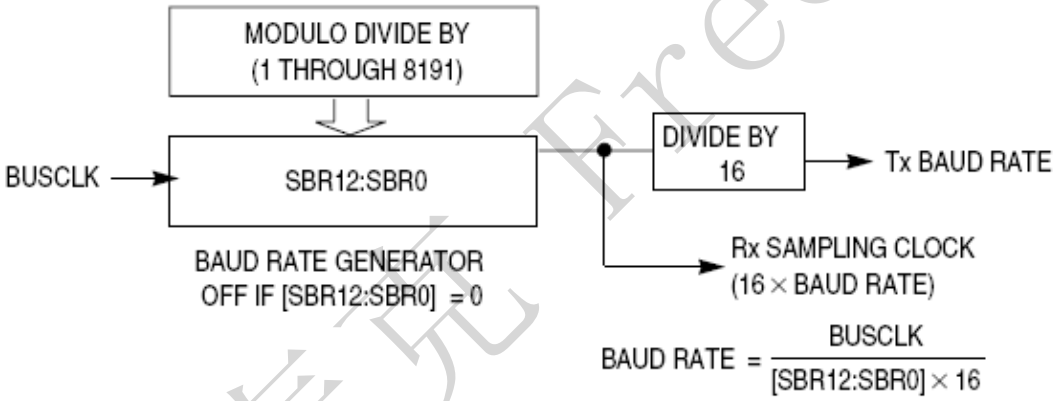


图 11.3 SCI 波特率发生器

SCI通信要求发送和接收使用同一波特率。Freescale半导体的SCI系统增强了波特率匹配能力。

14.3.2 发送器功能描述

发送器引脚默认为高电平。通过配置TXINV＝1，可使发送器输出反向。配置SCIxC2寄存器中的TE位为1，使能发送器的发送功能。发送器处于空闲状态，直到发送数据缓冲中有一个有效数据。

发送器的核心部分是10位或11位的发送移位寄存器。在8位数据模式中，移位寄存器中包含起始位，8个数据位，和一个停止位。当发送移位寄存器对新的字符可用时，在发送数据寄存器中的值转移到移位寄存器后，RDRE置位表明另一个字符可以写入发送数据缓冲器。

当停止位移出TxD引脚后，没有新的字符在发送数据寄存器中等待，那么发送器会置位发送完成标志位，进入空闲模式，TxD引脚为高，等待发送另一个字符。

如果发送器正在发送一个数据，空闲符，BREAK符的过程中，设置TE=0，并不会立

即释放TxD引脚。

14.3.2.1 发送BREAK字符和列队空闲

『前导符：由全1逻辑构成，包括起始位也是1。也就是说，当TE位由0跳变至1启动发送时，发送器控制逻辑自动装载一个由逻辑1组成的前导符至发送移位寄存器，作为数据头。前导符移出后，控制逻辑才将SCIxD的数据传至发送移位寄存器。

中止符：由全0逻辑组成，且无起始位，停止位和奇偶位。中止符长度取决于M位。将SBK位置1即会装载中止符至发送移位寄存器，只要SBK保持逻辑1，那么发送器将继续装载中止符至移位寄存器。待软件清除SBK后，移位寄存器中止最后一个中止符且至少发送1个逻辑1，这个1的自动插入能保障下一个字符的起始位识别。

空闲符：由全1逻辑构成，且没有起始位，停止位，和奇偶位。其长度取决于M。每次发送开始时的前导符就是一个用作同步的空闲符。如果发送期间的TE位清0，那么在当前发送结束后TxD引脚将处于空闲态。在当前字符发送后，当发送队等待1个空闲符发送时TE清0然后置1。』

SCIxC2中的SBK位用于控制发送中止符，该中止符最初用于引起老的电报接收器的注意。中止符由全0逻辑构成，包括起始和停止位。如果BRK13=1,那么中止符的长度为13位。

正常情况下：等待TDRE=1(表明报文的最后一个字符转移到移位寄存器)，然后向SBK位写1，然后向SBK位写0。只要移位器可用，中止符队列就被发送。如果SBK一直为1，当队列好的中止符移到移位器后，另一个中止符被队列。如果接受设备是另一个SCI，那么接收器将接收到8位全0，同时FE=1。

当使用空闲唤醒模式时，在两报文之间，一个全1字符将唤醒处于睡眠状态的接收器。正常情况下，当TDRE＝1，向TE位写0，然后向TE位写1。只要移位器可用，空闲符队列被发送。当设置TE=0，只要空闲符未发送完成，SCI发送器不会释放对TxD引脚的控制。

设置TE=0，移位器发送完成时，TxD引脚设置为通用I/O，且输出高电平。

14.3.3 接收器

接收移位器接收到停止位后，假如接收数据寄存器未满，数据向接收数据寄存器转移，RDRF被置位。表明接收数据寄存器已经满了，如果OR位置1，表明接收溢出，新数据丢失。SCI接收器是双缓冲，在RDRF置1后，在接收数据缓冲器被读取之前，程序有一个整个字符时间避免接收器溢出。

当检测到RDRF=1，程序应该读取SCIxD，从接收数据寄存器获取数据。

14.3.3.1 数据抽样技术

SCI使用16倍波特率的抽样。

14.3.3.2 接收器唤醒操作

接收器唤醒是一个硬件机制。允许SCI接收器忽略报文中的字符，该报文是发送给其他SCI接收器的。在这样的系统下，所有接收器评估报文的第一个字符，只要确定了该报文是发送给其他接收器的，那么会向SCIC2寄存器中的RWU位写1。当RWU被置位，

与接收器相关的状态标志将被屏蔽，因此减轻了软件处理不重要报文的工作。该报文发送完后，在下一个报文发送前，所有的接收器自动强制RWU清零，及时唤醒，查询下一个报文的第一个字符。

14.3.3.2.1 线路空闲唤醒

当WAKE=0，接收器配置为空闲线唤醒。在该模式下，当接收器检测到一个完整的空闲字符时间，RWU自动清零

14.3.3.2.2 地址唤醒

当WAKE=1，接收器配置为地址唤醒。在该模式下，当接收器检测到接收到的字符的最高位为1，RWU位自动清零。

14.3.4 中断和状态标志

SCI系统有三个独立的中断向量，从而降低软件对中断进行判断工作量。一个中断向量与发送器的TDRE和TC事件对应；一个中断向量和接收器的RDRF，IDLE，RXEDGIF，LBKDIE事件对应；第3个中断向量与OR，NF，FE，PF事件对应。

SCI发送器有两个状态标志产生硬件中断请求。发送数据寄存器空（TDRE）表明可以向SCIxD寄存器写入新的发送数据。如果TIE位置位，当TDRE位为1时，产生硬件中断。发送完成TC位表明发送器完成所有数据发送。该位经常用作安全关断数据传送的判断条件。同TDRE一样，如果TCIE位置位，当TC位为1时，产生硬件中断。如果TIE或TCIE位为0，那么可以通过软件轮询的方式查询TDRE和TC状态位。

当程序检测到接收数据寄存器满（RDRF=1），可以通过读取SCIxD寄存器获取数据。清除该状态位的方法：先读取SCIxS1，然后读取SCIxD。当然，用户采用轮询方式时，自然的遵循了上面所说的步骤。如果用户采用中断方式，那么在中断服务程序的一开始，SCIxI必须被读取。通常，中断服务程序中，读取SCIxS1，对错误标志位进行判断。

IDLE状态位置位后，RxD线仍处于空闲状态，IDLE不会再次置位。清除IDLE的方法：读取SCIxS1，然后读取SCIxD。IDLE被清除后，该位不会再次置位直到接收器至少接收一次字符之后。

如果RDRF位已经置位，此时一个新的数据准备从接收移位器转移到接收数据缓冲器中，那么OR位被置位。那么接收到的数据和与该数据相关的NF,FE,PF位均丢失。

在任何时候，RxD引脚上的有效边沿都会导致RXEDGIF位置位。向RXEDGIF位写1可以清除该位。

14.3.5 其他SCI功能

14.3.5.1 8位和9位数据模式

SCI系统通过设置SCIxC1寄存器中的M位，可以配置为9位数据模式。在9位模式中，在SCI数据寄存器的最高位后，作为第九位数据。发送时该位数据存放在T8位中；接收时第9位数据存放在R8中。

为保持发送数据缓冲器数据一致性，在写SCIxD之前，先写T8位。

14.3.5.2 STOP模式操作

所有STOP模式下，SCI模块的时钟均停止。

在STOP1和STOP2模式下，所有SCI寄存器数据均丢失，因此从这两种模式恢复后，必须进行初始化。而在STOP3模式下，所有的寄存器不会受到影响。

在STOP3模式下，接收输入有效边沿信号可以唤醒该模式下的MCU。但在STOP2模式下，该功能不再有效。

因此当SCI模块将要发送或接收字符时，用户软件应确保未进入STOP模式。

14.3.5.3 LOOP模式

当LOOPS=1，RSRC=0时，SCI配置为LOOP模式。在该模式下，发送器的输出内部连接到接收器的输入。因此RxD引脚没有被SCI使用，可作为通用I/O。

14.3.5.4 单线模式

当LOOPS=1，RSRC=1，SCI配置为单线模式。接收器内部连接到发送器的输出。因此RxD引脚没有被SCI使用，可作为通用I/O。

在单线模式下，SCIxC3寄存器的TXDIR位控制TxD引脚的方向。当TXDIR=0，TxD引脚作为SCI接收器的输入，发送器临时与外部设备断开；当TXDIR=1，TxD引脚被发送器的输出驱动。

第十五章 SPI

15.1 概述

15.1.1 SPI 特性

- 可设置为主机或从机工作方式
- 全双工模式或单线双向模式
- 可编程的传送速率
- 使用相互独立的发送和接受数据寄存器，实现双缓存冲操作
- 可设置时钟极性和相位
- 从机选择输出
- 可选择从最高位或最低位开始传送
- 可配置为 8 位或 16 位数据传送长度
- 接收数据缓冲硬件匹配特征

15.1.2 系统框图

SPI 系统最常见的应用是一个单片机做主机。在这种配置下，主机发起并控制数据的传送和流向，只有主机发出通知后，从机才能从主机读取数据或向主机发送数据。

15.1.2.1 系统框图

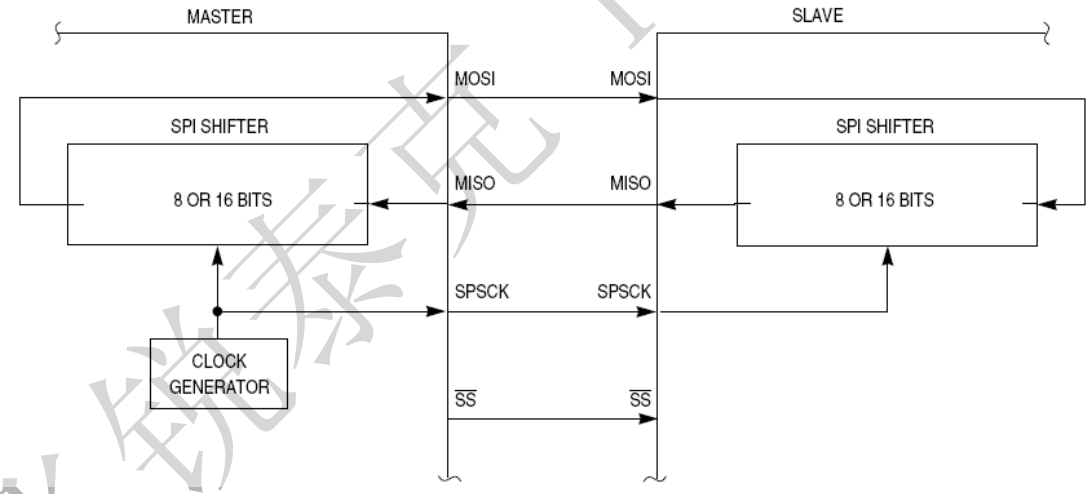


图 15.1 SPI 系统连接图

主机初始化 SPI 接口。在一次 SPI 传送发生时，一个字节通过主出从入（MOSI）引脚移位输出；同时另一个字节从主入从出（MISO）引脚输入。这个传送过程可以理解为主从 SPI 移位寄存器相互交换数据。SPSCK 为主设备输入给从设备的时钟信号。从设备的 SS(低有效)引脚必须拉低。

15.1.2.2 SPI 原理框图

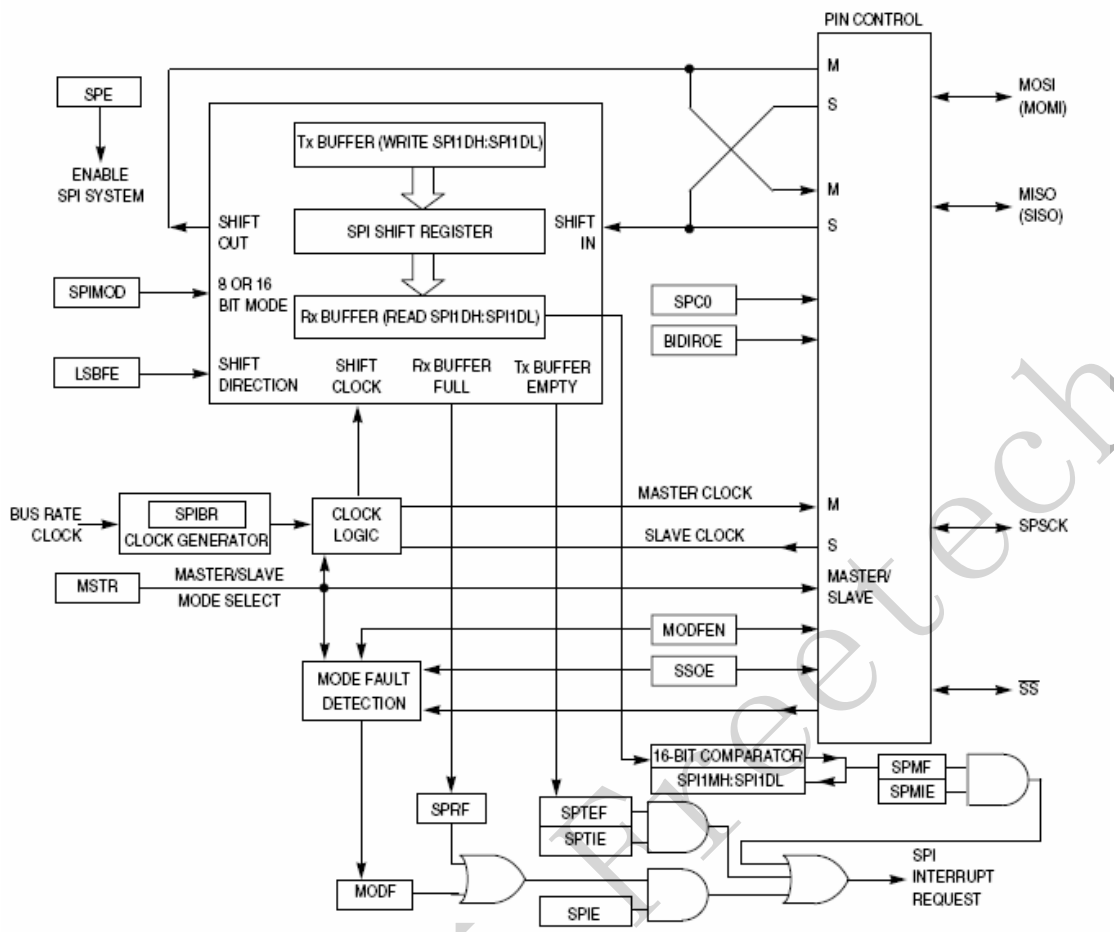


图 15.2 SPI 原理框图

SPI 的中心元件为 SPI 移位寄存器。CPU 将数据写到（SPI1DH：SPI1DL）寄存器，该数据装载到 SPI 移位寄存器，开始数据转移。当字节数据移位完毕后，数据转移到（SPI1DH：SPI1DL）中，可以读取该数据。

当 SPI 配置为主设备，SPSCK 引脚作为时钟输出；MOSI 作为移位寄存器的输出；MISO 作为移位寄存器的输入。

当 SPI 配置为从设备，SPSCK 引脚作为时钟输入，MISO 作为移位寄存器的输出；MOSI 作为移位寄存器的输入。

SPI 的外部连接：主从设备的 SPSCK 相连；MISO 相连；MOSI 相连。

15.1.3 SPI 波特率发生器

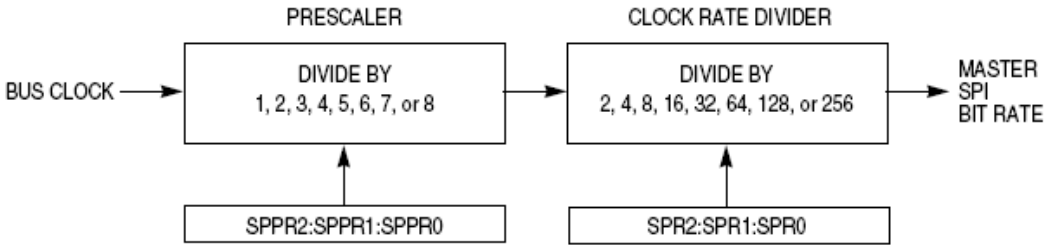


图 12.3 SPI 波特率发生器

时钟源为总线时钟，经过两级分频得到 SPI 通信的 1 位时钟。

15.2 外部引脚

SPI 的 4 个引脚与通用 I/O 复用。

15.2.1 SPSCK-SPI 串行时钟

串行时钟输出。

15.2.2 MOSI－主出从入

当作为主机，且 SPC0=0 时，该引脚作为输出；当作为从机，且 SPC0=0，该引脚作为数据输入。当 SPC0=1，且作为主机时，该引脚作为双向数据端口。在双向模式下，BIDIROE=0 时，该引脚为输入；BIDIROE=1 时，该引脚作为输出。

如果 SPC0=1,且作为从机，那么该引脚作为通用 IO 使用。

15.2.3 MISO－主入从出

当作为主机，且 SPC0=0,那么该引进作为数据输入；如果作为从机，且 SPC0=0，那么该引脚作为数据输出。如果 SPC0=1，且作为从机，那么该引脚作为双向数据端口。此时 BIDIROE=0，该引脚作为输入；BIDIROE=1，该引脚作为输出。

如果 SPC0=1,且作为主机，那么该引脚作为通用 IO 使用。

15.2.4 SS-从机选择

当作为从机，该引脚地电平有效。当作为主机，MODFEN=0，该引脚作为通用 IO。当作为主机，MODFEN=1，从机选择输出使能位 SSOE=1。

15.3 操作模式

15.3.1 SPI 在 STOP 模式

在 STOP 模式下，SPI 停止运行。其中 STOP1 和 STOP2 摸索下，SPI 完全断电。STOP3 模式下，SPI 的时钟挂起，寄存器不会改变。

15.4 寄存器定义

有 5 个寄存器

15.4.1 SPI 控制寄存器 1（SPIC1）

	7	6	5	4	3	2	1	0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0

7 SPIE	SPI 中断允许位 0：不允许由 SPRF 和 MODF 产生中断 1：允许 SPI 接收缓存满标志 SPRF 或模式错误标志 MODF 产生中断
6 SPE	SPI 系统使能 0:停止 SPI 1：允许 SPI

5 SPTIE	SPI 传送中断允许位 0：不允许 SPTEF 产生中断；1：允许 SPI 发送缓冲器空标志 SPTEF 产生中断
4 MSTR	主/从模式选择控制位 0：SPI 模块配置为从机模式；1：配置为主机模式
3 CPOL	时钟极性控制位 0：选择高电平有效时钟，SPSCK 空闲状态为低电平； 1：选择低电平有效时钟，SPSCK 空闲状态为高电平
2 CPHA	时钟相位控制位 0：在 SPI 数据传送的第 1 个周期的中间时刻产生第一个 SPSCK 跳变沿 1：在 SPI 的数据传送的第 1 个周期的开始时刻产生第一个 SPSCK 跳变沿
1 SSOE	从机选通控制位 该控制位与 SPIC2 中的 MODFEN 控制位一起控制 SS 引脚，见表 12.1
0 LSBFE	SPI 传送方向控制位 0：SPI 数据传送从最高位开始 1：SPI 数据传送从最低位开始

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input

表 15.1　SS 引脚功能

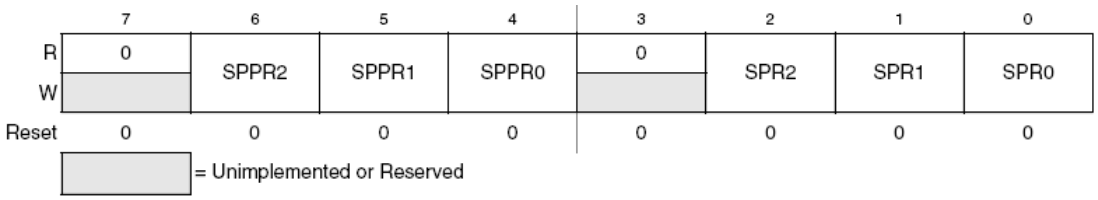
15.4.2　SPI 控制寄存器 2（SPIC2）

	7	6	5	4	3	2	1	0
R W	SPMIE	0	0	MODFEN	BIDIROE	SPIMODE	SPISWAI	SPC0
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

7 SPMIE	SPI 匹配中断使能控制位： 0：SPIMF=0 位不产生中断 1：SPIMF=1 可产生中断
4 MODFEN	主机模式检错功能允许控制位。该位对于从设备没有意义。作为主设备与 SPIxC1 的 SSOE 位一起作用。见表 12.1
3 BIDIROE	双向模式输出允许控制位。该位只有在 SPC0=1 时有效 0：SPI 的 I/O 作为输入端口；1：SPI 的 I/O 作为输出端口
1 SPISWAI	SPI 在 WAIT 模式下的停止控制位 0：在 WAIT 模式仍继续运行；1：在 WAIT 模式下，停止 SPI 时钟
0 SPC0	SPI 引脚控制位 0：SPI 进入全双工模式（输入/输出采用独立的信号线） 1：SPI 进入半双工工作模式（主机模式下使用 MOSI 作为双向数据线，从机模式下使用 MISO 作为双向数据线）

15.4.3 SPI 波特率寄存器（SPIBR）



6：4 SPPR[2:0]	SPI 波特率预分频器 见表 15.2
2：0 SPR[2:0]	SPI 分频因子 见表 15.3

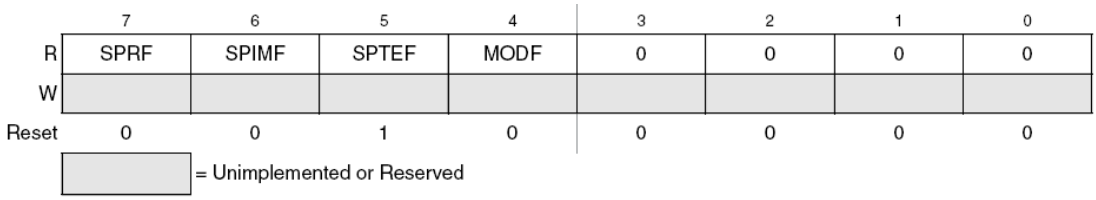
SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

表 15.2 SPI 波特率预分频器

SPR2:SPR1:SPR0	Rate Divisor
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

表 15.3 SPI 波特率分频器

15.4.4 SPI 状态寄存器（SPIS）



7 SPRF	SPI 接收缓冲满标志位－当 SPI 数据传送完成后，SPRF 被置 1，说明接收到的数据可以从 SPI 数据寄存器读出。清除该位方法：先读取 SPRF 标志位，然后读取 SPI 数据寄存器 0：接收数据缓冲器中无有效数据；1：接收数据缓冲器中数据有效
6 SPIMF	SPI 匹配标志：当接收数据缓冲器的值与 SPIMH:SPIML 的值一致时，该位置位。清除该位方法：先读取 SPIMF，然后向该位写 1。 0：接收数据缓冲器的值与 SPI1MH:SPI1ML 寄存器的值不一致 1：接收数据缓冲器的值与 SPI1MH:SPI1ML 寄存器的值一致
5 SPTEF	SPI 发送缓存空标志位。当 SPI 发送缓存空时，该标志置 1。清除方法：通过读取 SPTEF 标志位，然后把将要发送的数据写入到数据寄存器中。在向 SPI1DH:SPI1DL 写数据时前，必须先读取 SPI1S，否则向 SPI1DH:SPI1DL 写数据被忽律。 如果 SPTIE 位置位，SPTEF 产生一个 SPTEF 中断。 SPTEF 会自动设置当数据从发送缓冲转移到发送移位寄存器。 空闲模式下，当数据从发送缓存转移到移位寄存器，几乎同时该位被置位。 0：SPI 发送缓存非空 1：SPI 发送缓存空
4 MODF	主机模式错误标志－当 SPI 配置为主机模式，从机选择引脚变低，表明其他 SPI 设备也配置为主机模式。SS 引脚作为模式错误标志当 MSTR=1,MODFEN=1,SSOE=0；否则 MODF 不会置位。通过读取 MODF 位，然向 SPI1C1 寄存器写数据，可以清除该位。 0：无模式错误 1：检测到模式错误

15.4.5 SPI 数据寄存器(SPID)

读取 SPI 高低位数据寄存器返回接收数据缓存中的数据。写该寄存器将数据写入发送数据缓存。当 SPI 配置为主机，向发送数据缓存写数据将启动 SPI 发送。

在启动发送数据之前，应检查 SPTEF 位。如果 SPTEF 置位，表明在发送缓冲器中有一个空间接收新的发送数据，可以启动发生发送。

SPRF 位置位之后，在传送另一个数据之前，任何时刻可以读取 SPID 中的数据。如果在转移一个新数据之前，未从接收数据缓冲器中读出数据，那么将导致接收溢出，造成数据丢失。

15.5 功能描述

主机模式下，启动 SPI 数据转移的方法为：检查 SPTEF 是否为 1，然后向 SPI 数据寄存器写入要发送的数据。当 SPI 的移位寄存器可用，该数据从发送数据缓冲转移到移位器，此时 SPTEF 会置位，表明发送数据缓冲可以接收另一个数据。SPI 数据发送开始。

在 SPI 发送期间，在 SPSCK 的边沿对 MIMO 引脚进行抽样，移位，半个 SPSCK 时钟周期后，MOSI 引脚的电平改变为下一位值。8 或 16 个 SPSCK 周期后，主机移位寄存器中的数据从 MOSI 引脚移出，而从机将数据从 MISO 引脚移入到数据移位寄存器。在传送的最后，主机接收到的数据从移位器转移到接收数据缓冲，同时 SPRF 置位，表明数据可被读取。如果此时在发送缓冲有一个等待发送的数据，那么该数据转移到移位器中，SPTEF 置位，开始一个新的发送。

通常，SPI 数据先发送最高位。如果 LSBFE 置位，SPI 数据先发送最低位。

当 SPI 配置为从机模式，它的 SS 引脚在传送前必须被驱动到低电平，在整个传送期间，SS 必须处于低电平状态。当 CPHA=0，在连续传送之间，SS 必须被拉高。

如果 CPHA=1，SS 在连续传送之间可保持低电平。

由于发送器和接收器是双缓冲，所以先前接收到的数据可以存储在接收数据缓冲器中，同时一个新的字符正在移位。SPTEF 标志表明当发送缓冲有为新数据的空间。SPRF 标志表明在接收数据缓存中有一个有效数据。在下一次传送之前，接收到的字符必须从接收缓存中读出否则产生超载错误。

在超载情况下，新数据会丢失，因为接收缓存会一直保存着以前接收到的字符。因此应用程序开发者必须确保先前接收到的数据必须从接收缓存读出，才能启动下一次发送。

15.5.1 SPI 时钟格式

为了适应各生产厂家的不同 SPI 芯片，SPI 系统有一个时钟极性位和时钟相位控制位，用于选择 4 中时钟形式之一。

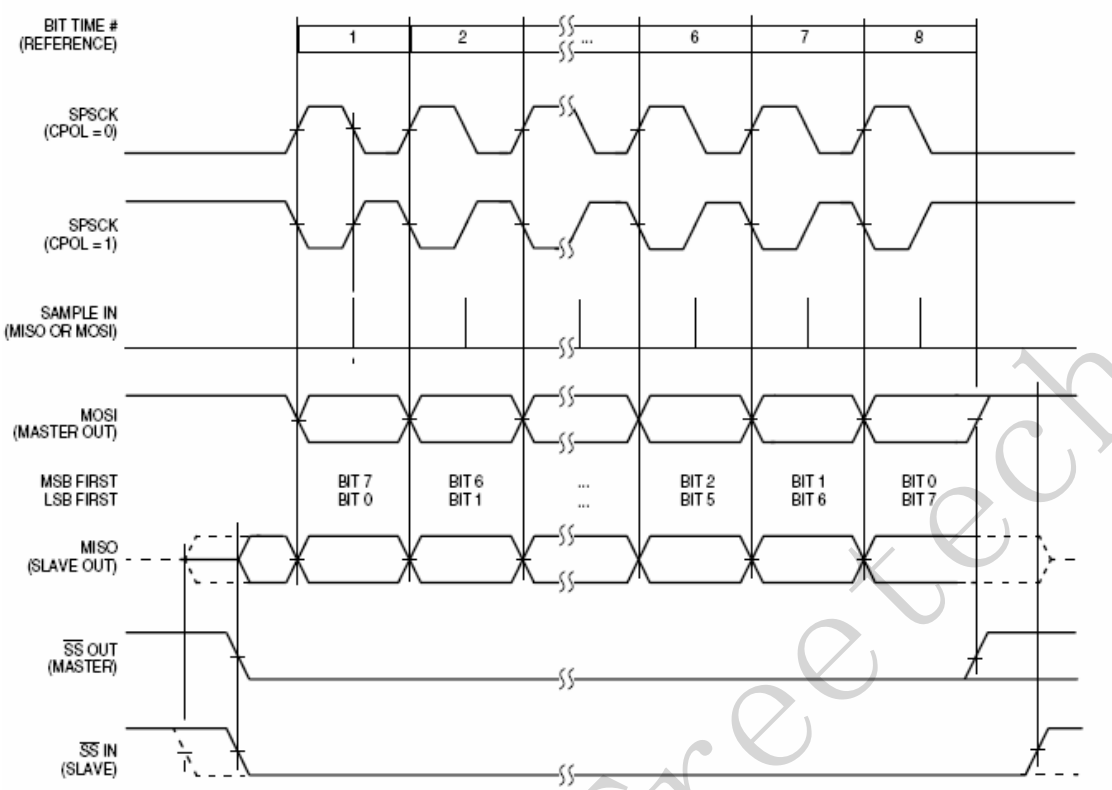


图 15.4 SPI 时钟（CPHA=1）

图 15.4 的时序图（SPIMODE=0，CPHA=1）。

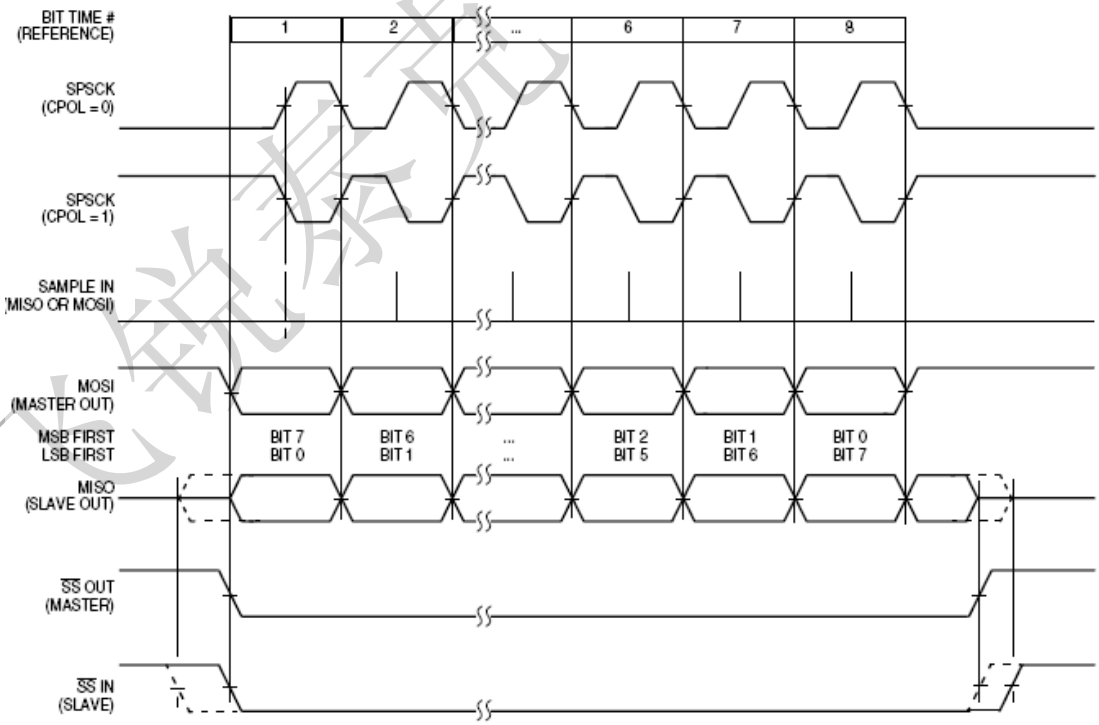


图 15.5 SPI 时序（CPHA=0）

图 15.5 的时序图（SPIMODE=0，CPHA=0）。

15.5.2 SPI 中断

SPI 系统有一个中断向量，4 个标志位和 3 个中断屏蔽位。

SPIE 位用于控制 SPI 接收器满标志（SPRF）和模式错误标志（MODF）是否引发中断。SPTIE 位用于控制 SPI 传送缓冲空标志 SPTEF 是否引发中断。SPIMIE 位用于控制 SPI 匹配标志（SPIMF）标志是否引发中断。当其中一个标志位置位，并且相应的中断屏蔽位使能，那么该标志将引发一个硬件中断。如果中断屏蔽位被清除，那么软件可通过轮询方式查询相应的标志位。

SPI 中断服务程序应该检查引发中断的标志位，该中断服务程序也应清除该标志位。在 8 位数据模式，SPRF，SPTEF，SPIMF 位设置为 8 位数据的接收，发送和匹配。

15.5.3 模式错误检测

当 SPI 主设备检测到 SS 引脚的错误信号(假设 SS 引脚配置为模式错误输入信号)，那么模式错误发生并且 MODF 位置位。当 MSTR＝1，MODFEN=1，SSOE=1 时，SS 引脚配置为模式错误输入信号。

模式错误检测特性可用于在同一个时刻可能有多于一个 SPI 设备成为主设备。当主设备的 SS 引脚被拉低，就会检测到错误，表明有其他设备试图寻址该设备。

当检测到模式错误，MODF 被置位。MSTR 会被清除，从而 SPI 配置为从设备。

第十六章 计数器/脉宽调制

16.1 介绍

MC9S08SH8 包括 2 个独立的定时器/PWM 模块。

Feature	MC9S08SH8/4		
Pin quantity	20	16	8
TPM1 channels	2	2	1[1]
TPM2 channels	2	2	1[1]

[1] The 8-pin device does not have TPM1 or TPM2 channel 1 bonded out, but those timer channels are available to the user to use as software compares.

表 16.1 不同引脚芯片的 TPM 通道数量

16.1.1 ACMP/TPM 配置信息

通过设置 SOPT2 寄存器中的 ACIC 位，ACMP 模块的输出可以连接到 TPM1 的输入捕捉通道 0。

16.1.2 TPM 配置信息

设置 TPMxSC 寄存器中的 CLKS[B:A]=1:1，那么 PTA0 引脚作为 TCLK 输入，为 TPM 和 MTIM 模块同时提供时钟源。

16.1.3 特点

- 每一个通道

－每一个通道可以为输入捕捉，输出比较，或边沿对齐 PWM

－上升沿，下降沿，或任意边缘触发输入捕捉

－设置，清除或触发比较输出功能

－可选择 PWM 输出极性

- 每一个 TPM 的通道都能设置为缓冲的，中心对齐脉宽调制输出 CPWM
- 每一个 TPM 的时钟源可以独立设定
- 时钟源包括：总线时钟分频，固定频率时钟，和外部引脚输入时钟

－时钟分频系数：1，2，4，8，16，32，64，128

－固定系统时钟源与总线时钟是同步的

－外部时钟引脚与时钟通道复用

- 16-位计数可工作于自由计数或递增/递减计数
- 16 位模计数寄存器用来控制计数范围
- 计时系统使能
- 每一个 TPM 模块每一个通道有一个中断外加一个计数溢出中断

16.1.4 操作模式

通常，TPM 通道可单独配置为输入捕捉，输出比较，或边沿对齐 PWM。有一个控制位允许 TPM 的所有通道工作于中心对齐 PWM 模式。

当单片机处于后台调试模式，TPM 临时挂起直到返回用户模式。停止模式下，所有的系统时钟包括晶振均停止，因此 TPM 也会禁止。等待模式下，TPM 正常继续运行。

- 输入捕捉模式

当一个有效的边沿信号施加到响应的 MCU 引脚，16 位计数器的当前值会被捕捉到该通道值寄存器，同时产生一个中断标志。上升沿，下降沿，任意边沿，或无边沿（关闭通道）都可以作为输入捕捉的触发源。

- 输出比较模式

  当定时器计数寄存器中的值与通道值寄存器值相等时，中断标志位置位，相关的 MCU 引脚输出一个信号。该信号可以为 1，0，或取反，或无输

- 边沿对齐 PWM 模式

  16 位模寄存器用于设置 PWM 输出信号的周期。通道值寄存器设置 PWM 输出信号的占空比。用户也可以选择 PWM 输出信号的极性。边沿对齐的 PWM 模式是指一个 TPM 模块的所有的 PWM 信号的起始沿在一个周期的开始时间点对齐。

- 中心对齐 PWM 模式

  16 位模寄存器值的 2 倍为 PWM 设置 PWM 信号周期,通道值寄存器设置占空比的一半。计数器递增计数直到计数值与模寄存器值相等，然后递减计数直到计数值为 0。当计数值与通道值寄存器相等（递减过程中），PWM 输出信号才激活。

16.2.1 框图

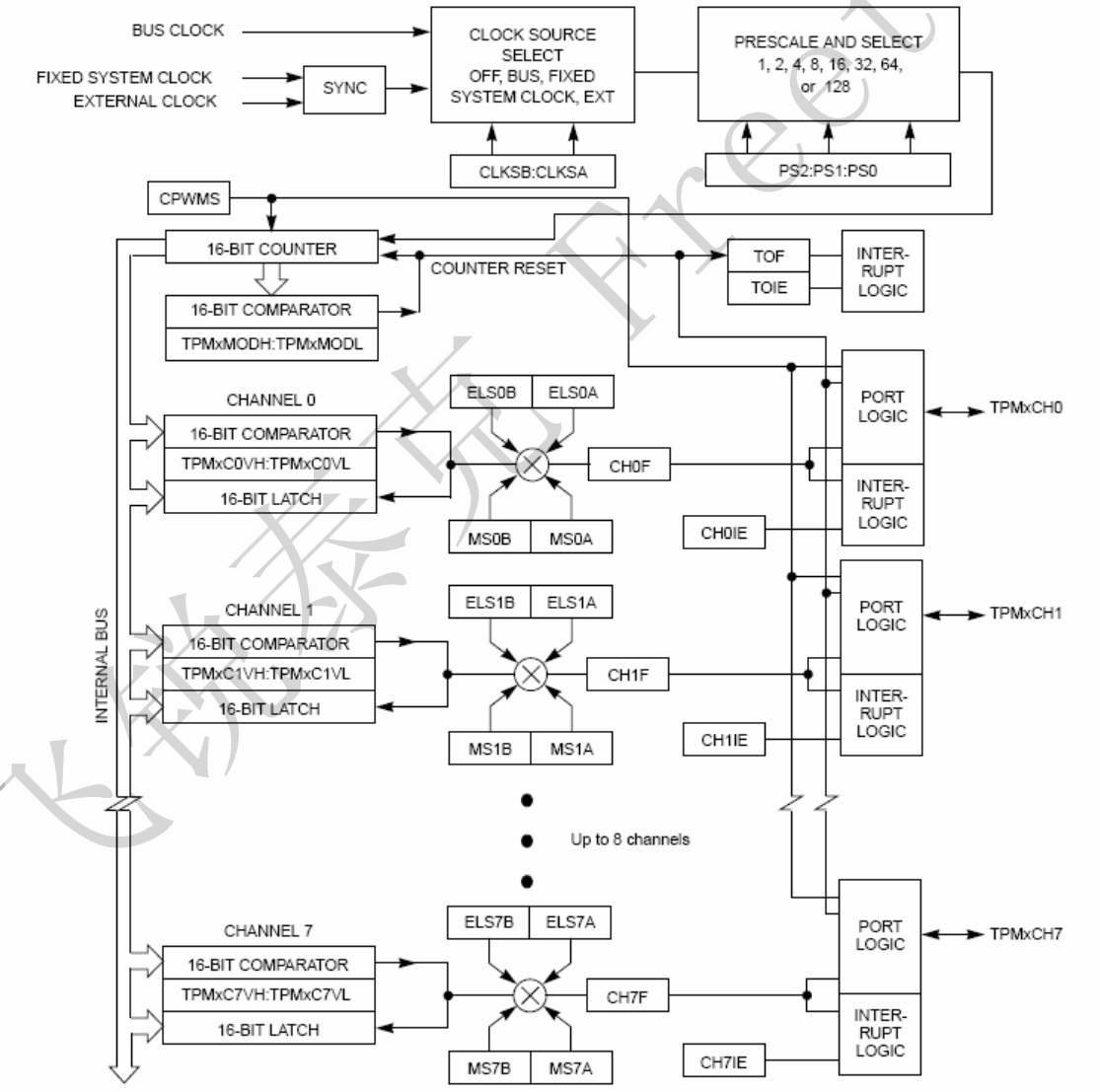


图 16.1 TPM 原理框图

TPM 通道可独立的作为输入捕捉，输出比较或边沿对齐 PWM。当产生中心对齐 PWM 时，所有通道均工作于该模式。

当一个通道配置为输入捕捉，那么内部上拉设备可以启动。

由于中心对齐 PWMs 通常用于驱动 3 相 AC 电机和无刷 DC 电机。

16.2 外部信号描述

EXTCLK	外部时钟源：可选择为 TPM 计数的时钟源
TPMxCHn	**I/O** 引脚与 TPM 通道 n

16.2.1 外部 TPM 时钟源

当 CLKSB:CLKSA＝1：1 时，TPMx 时钟由外部时钟源驱动。在外部时钟和 TPM 之间有一个同步器，总线时钟驱动该同步器。外部时钟源的最大频率极限为总线频率的 1/4。

外部时钟输入与 TPM 的某一个通道复用。当该引脚作为外部时钟输入时，相应的 TPM 通道不能再使用该引脚。

16.2.2 TPMxCHn－TPMx 通道 n 的输入/输出引脚

TPM 通道引脚与 IO 复用，当 ELSnB:ELSnA=0:0 或者 CLKSB:CLKSA=0:0 时，TPM 通道不再控制 IO。当 CPWMS=1，ELSnB:ELSnA 不为 0:0，一个 TPM 的所有通道引脚配置为中心对齐 PWM。而当 CPWMS=0，MSnB:MSnA 可以配置通道引脚为输入捕捉，输出比较或边沿对齐 PWM 模式。

当一个通道配置为输入捕捉模式时，该引脚作为一个边沿信号敏感输入到 TPM。ELSnB:ELSnA 可以设置边沿极性。此时引脚的端口数据和方向控制无效。

当一个通道配置为输出比较模式，相应的数据方向控制无效。

当一个通道配置为边沿对齐 PWM 模式，数据方向控制无效。TPMxCHn 引脚强制为输出，ELSnA 控制 PWM 输出信号的极性。当 ELSnB:ELSnA=1:0，TPMxCHn 引脚在每一个新周期的开始（TPMxCNT=0X0000）强制为高电平，当通道值寄存器与计数器值匹配时，引脚强制为低电平。当 ELSnA=1，TPMxCHn 引脚在每一个周期的开始强制为低，当通道值寄存器与计数器值匹配时，该引脚强制为高。

TPMxMODH:TPMxMODL=0x0008

TPMxCnVH:TPMxCnVL=0x0005

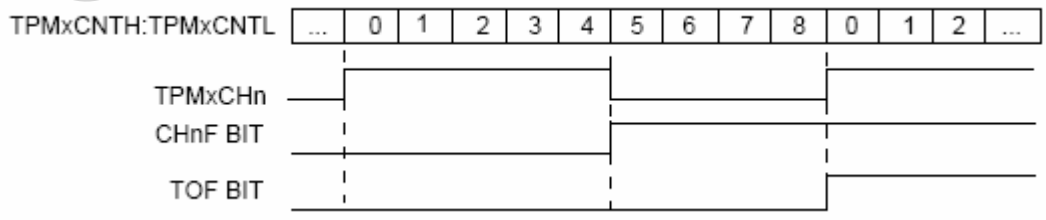


图 16.2 边沿对齐时高电平

TPMxMODH:TPMxMODL=0x0008

TPMxCnVH:TPMxCnVL=0x0005

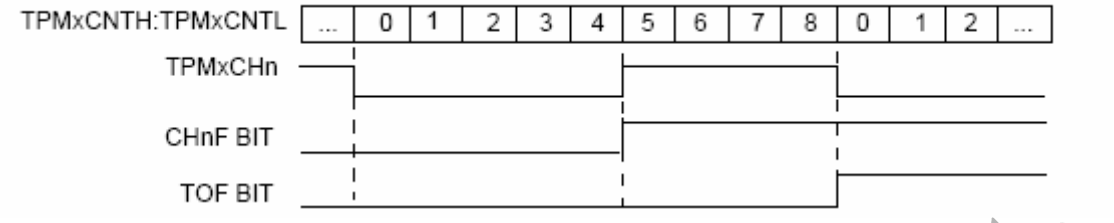


图 16.3 边沿对齐时低电平

当 TPM 配置为中心对齐的 PWM 模式时，所有通道的数据方向寄存器关闭。TPMxCHn 强制为输出状态。

TPMxMODH:TPMxMODL=0x0008

TPMxCnVH:TPMxCnVL=0x0005

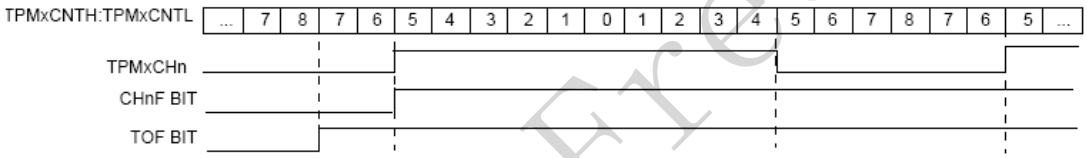


图 16.4 高电平型中心对齐 PWM

TPMxMODH:TPMxMODL=0x0008

TPMxCnVH:TPMxCnVL=0x0005

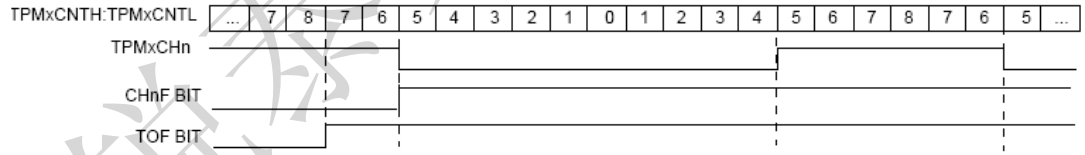
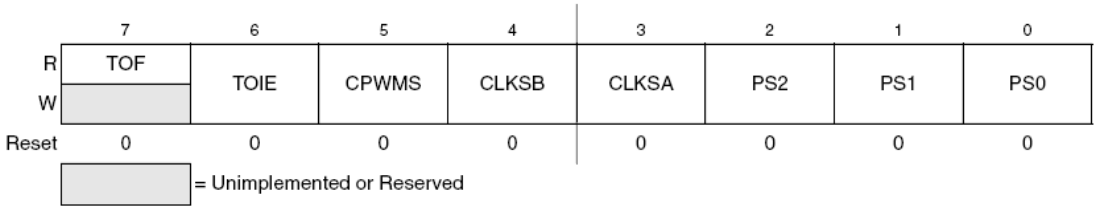


图 16.5 低电平型中心对齐 PWM

16.3 寄存器定义

16.3.1 计时器 n 状态和控制寄存器（TPMxSC）



7 TOF	定时器溢出标志－当定时器的计数达到$FFFF（自由计数模式）或预置计数器的值时，该位置 1。

	清除 TOF 方法：读 TPM 的状态和控制寄存器，然后向 TOF 位写 0。 如果在清除该标志位之前又发生了下一次溢出，那么这一次清零将无法完成。 程序必须保证每一次溢出发生后，都要在下一次溢出发生前清除该溢出标志位。在使用溢出中断的情况下，要求在中断返回之前必须清除溢出中断标志，并且该中断的处理时间在最不利的情形下也不能超过一次计数溢出时间。 0：未发生溢出；1：溢出。
6 TOIE	定时器溢出中断允许位：可读写 0：不允许溢出中断；1：运行溢出中断
5 CPWMS	中心对齐 PWM 0::：由 MSnB：MSnA 位决定所有的 TPMx 通道工作于输入捕捉，比较输出，或边沿对齐 PWM 模式 1：所有的 TPMx 通道工作于中心对齐 PWM 模式
4：3 CLKS[B:A]	时钟源选择：见下表
2：0 PS[2:0]	时钟源分频因子选择控制位：见下表

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disable)
01	Bus rate clock
10	Fixed system clock
11	External source

表 16.1 TPM 时钟源选择

PS2:PS1:PS0	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

表 16.2 时钟源分频列表

16.3.2 定时器 x 计数寄存器（TPMxCNTH:TPMxCNTL）

只读；当读取任意一个字节时，会将这两个寄存器的内容闩锁到一个缓冲器中，它们处于闩锁状态，直到另一个字节被读取；从而保证了数据一致性。

	7	6	5	4	3	2	1	0
R	Bit 15	14	13	12	11	10	9	Bit 8
W	Any write to TPMxCNTH clears the 16-bit counter.							
Reset	0	0	0	0	0	0	0	0

TPMxCNTH

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	Any write to TPMxCNTL clears the 16-bit counter.							
Reset	0	0	0	0	0	0	0	0

TPMxCNTL

当 MCU 的 BDM 激活，定时器计数器计数被冻结。

当后台调试器被激活，定时器的计数功能和一致性机制冻结。即使读取计数器的一个或两个寄存器值，缓冲器仍为拴锁状态。

16.3.3 定时器 x 预置数寄存器（TPMxMODH:TPMxMODL）

当定时器计数值等于预置数时，并且 CPWMS=0，那么计数器寄存器的值恢复为 0x0000。如 CPWMS=1，开始向下计数；同时 TOF 被置为 1。向这两个寄存器之一的写操作将屏蔽 TOF 和中断功能，直到另一个寄存器被写。复位会使这两个寄存器变为 0x0000。定时器运行于自由计数模式。

写 TPMxMODH 或 TPMxMODL 之一只是把要写入的值栓锁到一个缓冲中，根据 CLKSB:CLKA 位的配置，分为以下两种情况来更新这两个寄存器的值。

1、 如果 CLKSB:CLKSA=0:0，那么这两个寄存器会在这两个寄存器全被写后，立即更新。

2、 如果 CLKSB:CLKSA 不等于 0:0，当这两个寄存器都被写后，TPM 计数值从 TPMxMODH:TPMxMODL-1 变为 TPMxMODH:TPMxMODL 时更新。如果 TPM 计数于自由运行模式，那么当 TPM 计数值从 0xFFFE 变为 0xFFFF 时更新。

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

TPMxMODH

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

TPMxMODL

在修改模寄存器值之前，应复位 TPM 计数值，从而避免当模计数值改变后，计数器的不正确溢出。

16.3.4 定时器 x 通道 n 状态和控制寄存器（TPMxCnSC）

	7	6	5	4	3	2	1	0
R	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

7 CHnF	通道 n 标志： 当通道 n 配置为输入捕捉模式时，通道 n 上的有效边沿触发该位置位。 当通道 n 配置为比较输出或边沿对齐/中心对齐 PWM 模式时，TPM 计数寄存器中的值等于 TPM 通道 n 的寄存器值时，CHnF 置位。 当通道 n 为边沿对齐/中心对齐 PWM 通道，占空比为 0%或 100%，CHnF 不会置位（当 TPM 计数值与通道寄存器值相等）。 如果 CHnIE=1,那么当 CHnF 置位，就会产生相应中断事件。CHnF 不会自动清除，人工清除方法：读取 TPMxCnSC 寄存器，然后向 CHnF 位写 0。 如果在清除 CHnF 位过程中，另一个中断被请求，那么清除过程被复位，CHnF 仍保持置位状态。之所以这样做，是因为为了保证中断事件不丢失。 0：在通道 n 上无输入捕捉或输出比较事件发生 1：通道 n 发生输入捕捉或输出比较事件
6 CHnIE	通道 n 中断允许位： 0：通道 n 中断请求不允许；1：通道 n 中断请求允许
5 MSnB	通道模式选择位，见下表
4 MSnA	通道模式选择位，见下表
3：2 ELSn[B:A]	边沿/电平选择位，见下表

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin not used for TPM channel; use as an external clock for the TPM or revert to general-purpose I/O	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	00	Output compare	Software compare only
		01		Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)
		X1		Low-true pulses (set output on compare)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

在改变通道配置之后，在使能通道中断之前，清除状态标志，以避免发生不期望的事件。

16.3.5 计时器 x 通道值寄存器（TPMxCnVH:TPMxCnVL）

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

TPMxCnVH

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

TPMxCnVL

在输入捕捉模式下，读取任一个寄存器值将闩锁这两个寄存器的值到一个缓冲器中，这两个值维持闩锁状态，直到另一个字节被读取。当 TPMxCnSC 寄存器被改写时，这种闩锁机制复位。在输入捕捉模式下，对通道寄存器的任意写将视为无效操作。

在比较输出或 PWM 模式中，写任意一个寄存器，将被闩锁到一个缓冲器中，只有这两个字节都被写，它们才真正传送到通道寄存器中。根据 CLKSB:CLKSA 配置，有以下两种情况更新这两个寄存器的值。

第一种情况：CLKSB:CLKSA=0:0，当这两个寄存器全部被写后，寄存器的值立即更新。

第二种情况：当 CLKSB:CLKSA 不等于 0:0，TPM 工作于只输出模式，那么在这两个寄存器都被写操作后，TPM 计数器下一次计数后，寄存器值更新。

第三种情况：CLKSB:CLKSA 不等于 0:0，TPM 工作于 EPWM 或 CPWM 模式，当 TPM 计数值由 TPMxMODH:TPMxMODL-1 变为 TPMxMODH:TPMxMODL 后，这两个寄存器值更新。

16.4 功能描述

所有的 TPM 功能与主要部件 16 位计数器相关联。

16.4.1 计数

复位后，CLKSB:CLKSA=0:0；无时钟源，TPM 禁止。任何复位后，CLKSB:CLKSA=0:0 因此没有选择时钟源。这两个控制位可以任何时刻被读取或改写。关闭计数器不会影响计数器或其他寄存器的值。

一个中断标志位和使能位与 16 位计数器相关。TOF 位用于表明计数是否溢出。TOIE 位控制是否使能硬件中断。根据 TPM 工作模式，有几种不同的溢出方式。最简单的一种就是计数器从 0x0000 计数到 0xFFFF，然后在下一个计数时钟变为 0x0000。那么当计数值从 0xFFFF 变为 0x0000 时，TOF 置位。当模计数使能，那么计数器由模值转变为 0 后，TOF 置位。当 TPM 工作于中心对齐 PWM 模式，当计数值的达到终点值时，TOF 置位。

作为递增计数，16 位计数器从 0x0000 到终点值，之后继续从 0x0000 计起。终点值为 0XFFFF 或预置值。

当处于中心对齐 PWM 模式时，重复从 0x0000 递增到终点值，然后递减到 0x0000 的过程。

通过对 TPMxCNTH 或 TPMxCNTL 之一的写任意值，那么计数就会被复位。

16.4.2 通道模式选择

16.4.2.1 输入捕捉模式

用于捕捉发生在外部引脚事件时间。当一个有效的边沿触发输入捕捉通道，TPM 将 TPM 计数寄存器中的值锁存到通道寄存器中。有效边沿包括上升沿，下降沿或任意边沿。

该模式下，TPMxCnVH 和 TPMxCnVL 寄存器是只读的。

当对读取这两个寄存器其中一个后，另一个寄存器的值就会栓锁到一个缓冲中。当然用户可以通过对 TPMxCnSC 寄存器的写操作复位一致性机制。

CHnF 标志可向 CPU 提出中断请求。

MCU 处于 BDM 模式，输入捕捉工作于用户配置方式下。当外部信号发生，TPM 会将 TPM 计数值锁存到通道值寄存器中，同时置通道标志位。

16.4.2.2 比较输出模式

用于产生定时脉冲。

当计数器计数值等于通道寄存器值时，TPM 可置位，清除或反转通道引脚电平信号。

修改通道寄存器值遵循一致性原则

可以产生中断。

16.4.2.3 边沿 PWM 模式

PWM 输出的一种典型方式为计数器正常递增计数（CPWMS=0），同时同一 TPM 模块的其他通道配置为输入捕捉或输出比较功能。PWM 信号的周期由模寄存器（TPMxMODH:TPMxMODL＋1）决定。占空比由通道值寄存器（TPMxCnVH:TPMxCnCL）决定。而 PWM 的信号极性由 ELSnA 控制位决定。占空比可设置为 0%或 100%。下图所示：

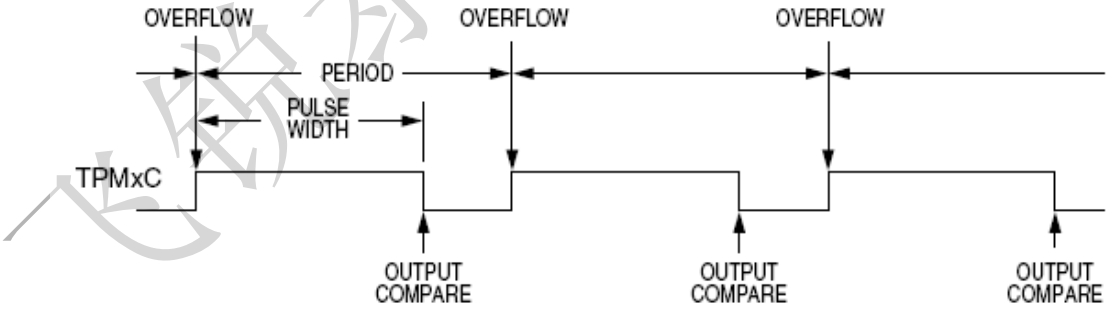


图 16.6 边沿 PWM 周期与占空比（ELSnA=0）

TPMxMODH:TPMxMODL 用于决定 PWM 周期。

TPMxCnVH:TPMxCnVL 用于决定一个周期内高电平时间，即脉宽时间 Tpulse。

ELSnA 决定信号变化。ELSnA＝0：计数溢出时引脚输出高电平，当比较输出时强制引脚为低电平。ELSnA＝1：情况相反。

当通道值寄存器设置为 0X0000，占空比为 0%。

如果通道值寄存器的值大于模寄存器的值，占空比为 100%。

工作过程（ELSnA＝0）：

当用户程序设置好 Tcycle 和 Tpulse 及 ELSnA 后，计数器从 0x0000 开始计数，此时通道引脚为高电平，当计数值等于 Tpulse 时，通道引脚变为低电平，计数值继续递增，当计数值等于 Tcycle 时，溢出，通道引脚变为高电平，重复该过程。

16.4.2.3 中心对齐 PWM 模式

当 CPWMS=1 时，计数器会递增/递减计数模式。TPMxCnVH:TPMxCnVL 中的值决定 PWM 的脉宽；TPMxMODH:TPMxMODL 决定 PWM 的周期。

TPMxMODH:TPMxMODL 的值设置范围为 0X0001~0X7FFF，超过这个范围产生不确定的结果。ELSnA 决定 CPWM 输出的极性。

pulse width = 2 x (TPMxCnVH:TPMxCnVL)

period = 2 x (TPMxMODH:TPMxMODL);
for TPMxMODH:TPMxMODL = 0x0001–0x7FFF

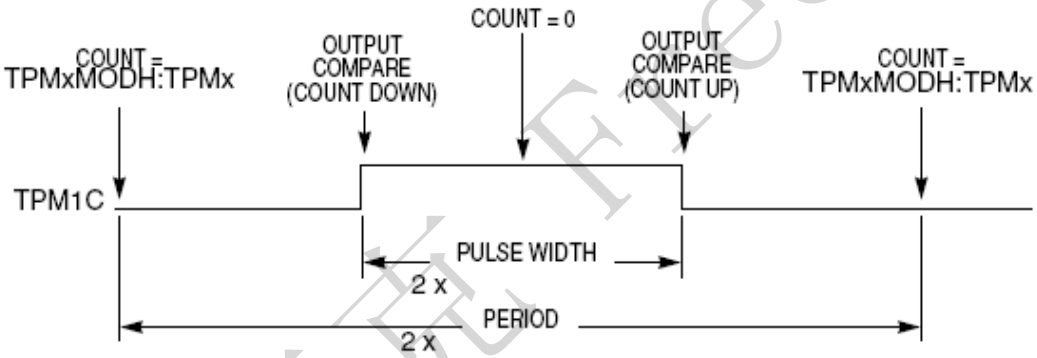


图 16.3 CPWM 周期与占空比

当 TPMxCnVH:TPMxCnVL＝0 或最高位＝1，占空比为 0%。

当 TPMxCnVH:TPMxCnVL（最高位＝0）大于 TPMxMODH:TPMxMODL（非 0），占空比为 100%。

工作过程（ELSnA=0）：

计数器从 0 开始递增，通道引脚为高电平，当计数值等于比较输出计数值(通道值寄存器的值)时，通道引脚强制为低电平，计数值继续递增；当计数值等于预置数值时，开始递减；当计数值递减到比较输出数值时，通道引脚强制为高电平，计数值继续递减；当计数值递减到零时，开始递增。

该模式下，模寄存器取值范围 0x0001～0x7FFE。

16.5 复位后 TPM 状态

复位将清除 TPMxSC 寄存器，TPM 时钟关闭，TPM 溢出中断禁止。

16.6 TPM 中断

16.6.1 概述

TPM在计数溢出或每个通道产生中断。如果通道配置为输入捕捉，当通道引脚出现有效边沿时，中断标志位置位。当通道配置为比较输出或PWM模式时，每当计数器匹配通道

寄存器中的值时中断标志位置位。

对于TPM中的每一个中断源，有一个相应的标志位进行标识。该标志位可被软件轮询到或向CPU申请中断。

16.6.2 中断操作

定时器溢出，通道输入捕捉或输出比较事件都会产生中断事件。当中断使能，发生中断事件后，CPU会执行相应的中断服务程序。用户在中断服务程序执行完之前要清除中断标志。

清除中断标志的方法：读取该标志位然后向该标志位写0。如何在这两步之间产生一个新的中断，那么该次清除过程不会清除标志位，从而避免遗漏新的中断事件。

16.6.3 定时器溢出中断

定时器自由运行模式时，当计数器的值由 0xFFFF 变为 0x0000 时，TOF 置位。

定时器运行于预置数模式时，当计数器的值由预置数变为 0x0000 时，TOF 置位。

定时器运行于递增/递减模式时，当改变计数方向时，该位置位。也就是当计数值等于模寄存器值时，该位置位。

16.6.4 通道事件中断

通道事件包括输入捕捉，比较输出，边沿对齐 PWM 或中心对齐 PWM。

当通道配置为边沿对齐 PWM，当计数器的值与通道寄存器的值匹配时，通道标志被置位。

当通道配置为中心对齐 PWM，计数器的值会与通道寄存器的值匹配两次，故通道标志在脉冲的起始和结束均置位。

第十七章 开发工具

17.1 介绍

本章介绍利用片内 BDC(后台调试控制器)实现单线后台调试，利用 DBG（片上调试模块）实现片上实时 ICE（在线仿真）。

在 HCS08 家族，地址和数据总线信号不能通过外部引脚被访问。通过单线后台调试接口发送给目标 MCU 的命令进行调试。调试模块提供可选择的触发或捕捉总线信息，从而外部系统可以观测到 MCU 内部的运行情况。BDC 的时钟源为 ICGLCLK。

17.1.1 特点

BDM 特性：

- 单线模式选择和后台调试
- BDC 寄存器不位于存储器映射空间
- SYNC 命令同步主机与目标 MCU 通信速率
- 非干扰调试命令访问存储器
- 后台调试命令可访问 CPU 寄存器
- 连续运行(GO)和单步跟踪(TRANCE1)命令
- BACKGROUND 命令可以唤醒处于等待或停止模式的 CPU
- BDC 支持一个断点
- 通过 BDC 设置，振荡器可运行在 STOP 模式
- 看门狗在后台调试模式下停止工作

ICE 特征：

- 两个触发比较器：两个地址＋读/写（R/W）或一个全地址＋数据＋R/W
- 灵活的 8 级 16 位 FIFO 缓冲，用于捕捉信息

－分支地址

－只数据事件

- 两种类型断点

－标记类型断点：指令操作码

－强制类型断点：访问任意地址

- 9 种触发模式

－基本：只 A，A 或 B

－连续：A 然后 B

－全：A 与 B 数据，A 与 B 的反

－事件：只事件 B，事件 A 任何只事件 B

－范围：A<= 地址 <=B，地址 < A，地址>B

17.2 后台调试控制器（BDC）

HCS08 系列单片机均包含一个单线后台调试接口，支持在线编程和复杂的非干扰调试能力。不同于早期的 8 位 MCU 调试接口，该系统不会干扰正常的应用资源，也不会使用任何用户存储器或存储器映射空间，也不与片上外围复用。

BDC 命令分为两组：

- 主动后台模式命令：要求目标MCU 处于后台调试模式下(用户程序不运行)，通过该组命令可以读写CPU寄存器，允许用户在某一时刻跟踪运行一条指令，或者全速脱离后台调试模式而运行用户的程序。
- 非干扰命令能：该命令在任何时候都能执行，甚至在用户程序运行时期。非干扰命令允许一个用户读写MCU存储器空间或访问在后台调试控制器(BDC) 内的状态和控制寄存器

主机和目标之间通过一个接口连接，通常连接目标的 GND，BKGD，RESET，VDD (可选)。RESET 接口用于主机强制目标系统复位，从而重新获得对目标系统可控制或在目标系统 FLASH 存储器未编程之前控制目标系统的建立。VDD 用于从目标系统为调试器供电，当调试器单独供电时，不会强制目标系统复位，也不会干扰目标系统的正常运行。

17.2.1 BKGD 引脚描述

BKGD 是单线后台调试接口引脚。该引脚主要功能是在后台调试中传送命令和数据，并且是双向的。复位期间，该引脚用于选择目标系统是进入后台调试模式还是用户程序正常运行模式。另外该引脚对同步命令作出一个定时的脉冲响应，从而使主机确定目标系统后台调试串行通信的时钟频率。

BDC 串口通信由主机发起和控制，主机驱动出一个由高到低的边沿作为每一位时间的开始。命令和数据的最高位首先被送出。

如果主机不知目标系统的 BDC 时钟频率，那么主机会向目标系统发送一个 SYNC 命令，目标系统会回复一个定时的同步响应信号，主机根据该信号确定正确的通信速度。

BKGD 引脚是一个伪漏级开路引脚，内部有一个上拉。不同于典型的漏级开路引脚，该引脚的外部 RC 时间对该引脚上信号的上升时间几乎没有影响。并且后台调试协议规定简短，驱动能力强的信号强制快速的上升时间。

当目标系统未连接调试器，那么 BKGD 引脚会工作于正常操作模式。当开发系统连接到目标系统，调试器会将 BKGD 和 RESET 引脚拉低，然后释放 RESET 引脚，从而使目标系统进入后台调试，然后释放 BKGD 引脚。因此当通过后台调试接口进行通信时，没有必要复位目标 MCU。

17.2.2 通信细节

BDC 串行通信接口要求外部调试器在每位的开始 BKGD 引脚上产生一个下降沿。无论调试器发送或接收数据，外部控制器都要提供该下降沿。

BKGD 引脚可以由自身 MCU 或调试器驱动。每位数据耗时 16 个 BDC 时钟周期最高位首先被传送。调试发出的两个下降沿之间的超时时间为 512 个 BDC 时钟周期。一旦发生超时，任何运行的 BDC 通信命令被中止，但不会影响目标系统的存储器和操作模式。

BDC 的状态和控制寄存器中的 CLKSW 位用于用户选择 BDC 时钟源。时钟源为总线或 BDC 时钟源。

BKGD 引脚可以接收高低电平或发送高低电平。

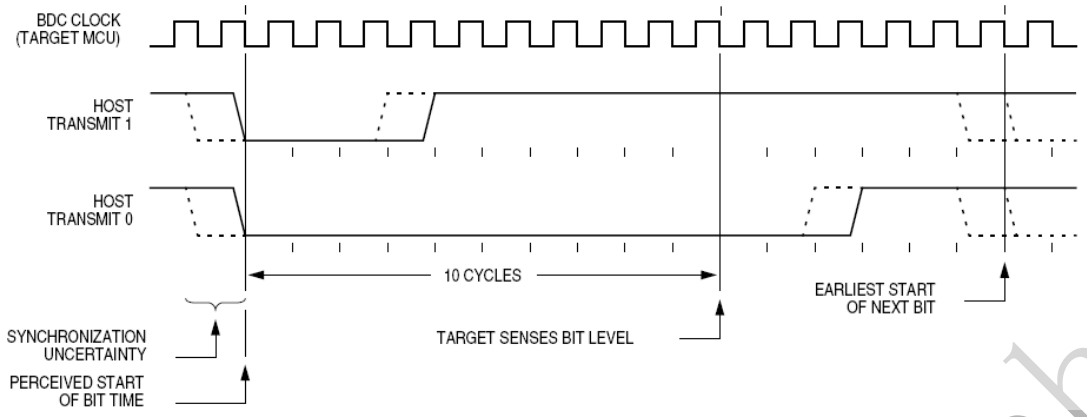


图17.1 主机－>目标一位数据时序

由于主机和目标是异步通信，因此目标会检测是否有主机的下降沿，作为一位数据传送的开始。10 个目标系统 BDC 时钟周期后，目标系统会检测 BKGD 引脚的电平值。典型情况。

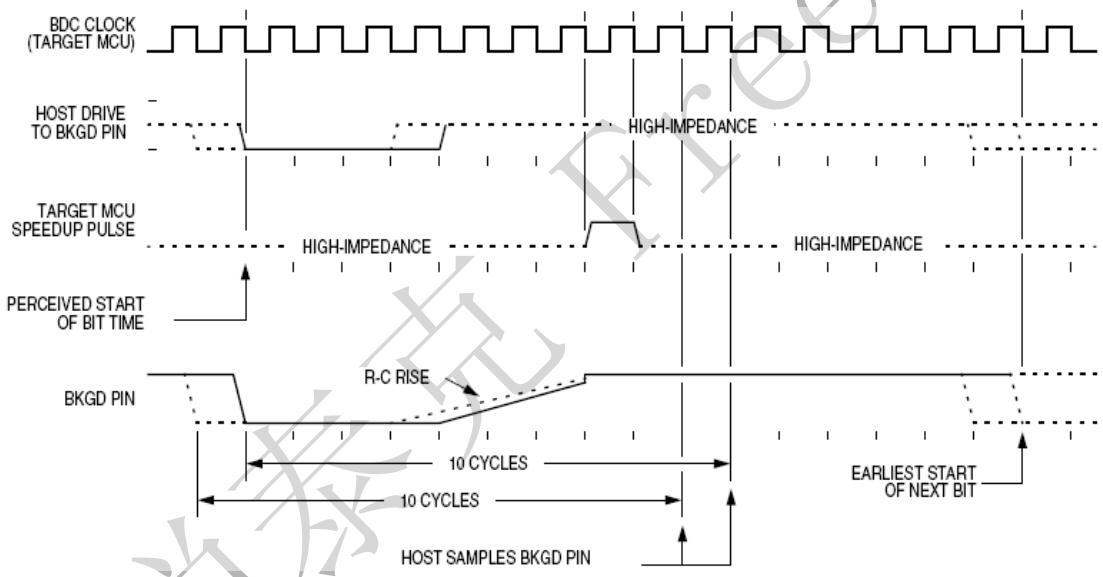


图17.2 目标－>主机一位数据（1）

从上图可以看出，在主机发送下降沿后，主机会控制 BKGD 引脚为低，时间至少两个目标 BDC 时钟周期。然后，主机释放 BKGD 引脚。目标 MCU 在起始后的第 7 个 BDC 周期会驱动 BKGD 引脚为低，接着在第 10 个周期，主机抽样 BKGD 引脚电平。

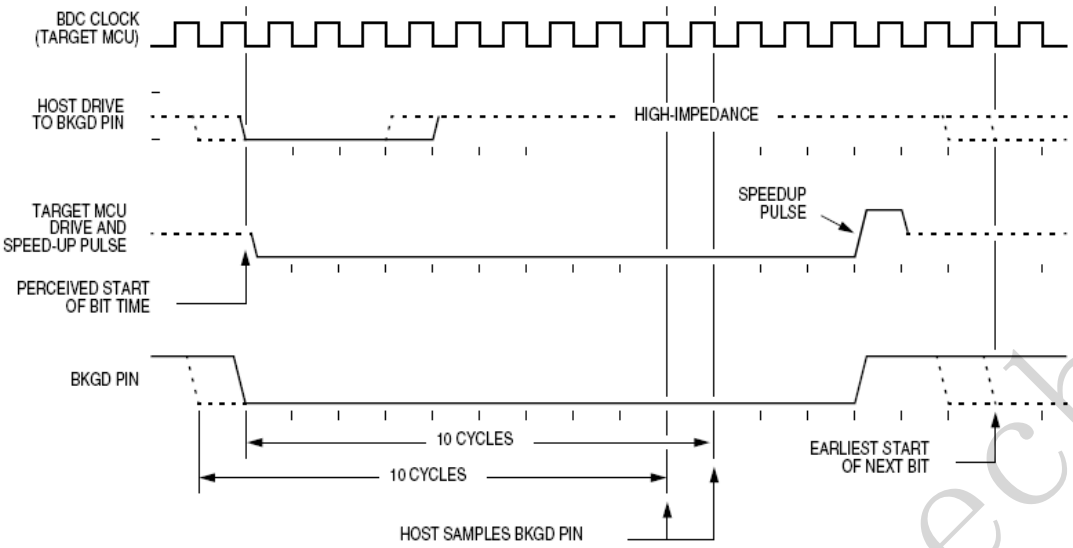


图15.3 目标－>主机一位数据（0）

从上图可以看出，主机启动位传送后，目标 MCU 会驱动 BKGD 引脚为低，在第 13 个 BDC 时钟周期，会产生一个简短脉冲。主机会在第 10 个周期采样 BKGD 引脚。

17.2.3 BDC 命令

/=分割命令符

d=延时 16 个目标 BDC 时钟周期

AAAA=16 位地址，主机－>目标

RD=8 位数据，目标－>主机

WD=8 位数据，主机－>目标

SS=BDCSCR 的内容，目标－>主机

CC=写 BDCSCR 的内容，主机－>目标

AAAA=16 位地址，主机－>目标

RBKP=BDCBKPT 断点寄存器 16 位内容，目标－>主机

WBKP=写入到 BDCBKPT 断点寄存器的 16 位内容，主机－>目标

RD16=16 位数据，目标－>主机

WD16=16 位数据，主机－>目标

命令名称	命令类型	命令码及格式	命令描述
SYNC	非干扰	无	请求一个定时参考用于确定目标 BDC 通信速度
ACK_ENABLE	非干扰	D5/d	使能确认机制协议
ACK_ENABLE	非干扰	D6/d	禁止确认机制协议
BACKGROUND	非干扰	90/d	进入后台调试模式
READ_STATUS	非干扰	E4/SS	从 BDCSCR 读取 BDC 状态
WRITE_CONTROL	非干扰	C4/CC	写 BDC 的 BDCSCR
READ_BYTE	非干扰	E0/AAAA/d/RD	从目标存储器读一个字节
READ_BYTE_WA	非干扰	E1/AAAA/d/SS/RD	从目标存储器一个字节，并返回 BDCSCR 内容

READ_LAST	非干扰	E8/SS/RD	重读刚读过的地址中的内容，并返回 BDCSCR 内容
WRITE_BYTE	非干扰	C0/AAAA/WD/d	向目标 MCU 存储器写一个数据
WRITE_BYTE_WS	非干扰	C1/AAAA/WD/d/SS	向目标 MCU 存储器写一个数据，目标 MCU 返回 BDCSCR 内容
READ_BKPT	非干扰	E2/RBKP	读取 BDCBKPT 断点寄存器内容
WRITE_BKPT	非干扰	C2/WBKP	向 BDCBKPT 寄存器写入数据
GO	主动式	08/d	从 PC 指向的应用程序开始执行
TRACE1	主动式	10/d	跟踪 PC 指向的一条指令，目标返回后台调试模式
TAGGO	主动式	18/d	同 GO
READ_A	主动式	68/d/RD	读取累加器 A
READ_CCR	主动式	69/d/RD	读取条件码寄存器 CCR
READ_PC	主动式	6B/d/RD16	读取 PC
READ_HX	主动式	6C/d/RD16	读取 H:X
READ_SP	主动式	6F/d/RD16	读取 SP
READ_NEXT	主动式	70/d/RD	H:X 加 1，读取 H:X 指向的存储器内容
READ_NEXT_WS	主动式	71/d/SS/RD	H:X 加 1，读取 H:X 指向的存储器内容，并且返回 BDCSCR
WRITE_A	主动式	48/WD/d	写寄存器 A
WRITE_CCR	主动式	49/WD/d	写条件码寄存器 CCR
WRITE_PC	主动式	4B/WD16/d	写 PC
WRITE_HX	主动式	4C/WD16/d	写 H:X
WRITE_SP	主动式	4F/WD16/d	写 SP
WRITE_NEXT	主动式	50/WD/d	H:X 加 1，然后写入 H:X 指向的地址
WRITE_NEXT_WS	主动式	51/WD/d/SS	H:X 加 1，然后写入 H:X 指向的地址，并且返回 BDCSRC 内容

SYNC 命令：

主机操作过程：

- 主机驱动 BKGD 引脚为低至少 128 个可能最慢的 BDC 时钟周期
- 主机驱动 BKGD 引脚一个简短快速的脉冲（该快速脉冲宽度典型为一个系统最快时钟周期）
- 主机释放 BKGD 引脚
- 主机监测 BKGD 引脚上的同步相应脉冲

目标操作过程（检测到 SYNC 要求）：

- 等待 BKGD 引脚返回高电平
- 等待 16 个周期
- 驱动 BKGD 引脚为低，持续 128 个 BDC 时钟周期
- 驱动一个脉冲宽度为 1 个 BDC 时钟周期的快速脉冲
- 释放 BKGD 引脚

主机通过测量 128 个周期的低电平从而确定目标 MCU 的通信速度。

17.2.4 BDC 硬件断点

BDC 包含一个相对简单的硬件断点，比较 CPU 地址总线和在 BDCBKPT 寄存器中的 16 位地址。该断点可以为一个强制类型断点或标记类型断点。

强制断点是指在执行断点处的下一条指令之前，强制 CPU 进入后台调试模式，该断点可以设置为任何地址。

标记断点是指断点地址处的操作码被标记，当 CPU 执行到该操作码时，CPU 进入后台调试模式，而不是执行该操作码；只能放置指令操作码地址。

在 BDCSCR 寄存器中的 BKPTEN 位用于控制使能断点逻辑功能

FTS 位用于选择强制类型断点或标记类型断点。

DBG 模块包含另外两个硬件断点电路，因此更灵活。

17.3 片上调试系统（DBG）

17.3.1 比较器 A 和 B

因为 HCS08 没有外部地址和数据总线，在线仿真内置在 MCU 中。该调试系统包括 8 级的 FIFO，用力存储地址和数据信息，灵活的触发系统决定何时捕捉总线信息和捕捉何种信息。

系统依赖单线后台调试系统来访问调试控制寄存器和读取 8 级 FIFO 的结果。

该调试模块包括控制和状态寄存器。

调试模块的最大用处是用于开发过程中，用户程序很少访问调试模块的任何控制和状态寄存器。

17.3.2 总线捕获信息和 FIFO 操作

使用 FIFO 通常方式是启动触发模式和其他控制选项，然后 ARM（启动）调试器。当 FIFO 满或者调试停止，存储数据到 FIFO 时，你可以读出存储到 FIFO 的数据。状态位表明存储在 FIFO 中有效的字数目。如果通过向 ARM 位写 0，在 CNT=1:0:0:0 之前，人为的停止跟踪运行，那么信息移入下一个位置，主机必须执行((8-CNT)-1)读出 FIFO 第一条存储数据。

在大多数触发模式下，存储在 FIFO 中的信息由 16 位 change of flow 地址组成。在这些情况下，读 DBGFH，然后读 DBGFL 会得到一个一致的字信息。读 DBGFL 导致 FIFO 指向下一个字信息。在只事件触发模式，8 位数据信息存储到 FIFO。在这些情况下，高 8 位 FIFO 没有用于存储数据。DBGFL 每次被读取，FIFO 转移到下一个可用数据。

在触发模式，FIFO 存储 change of flow 地址，在 CPU 地址和存入 FIFO 之间有一个延时。因为该个延时，如果触发事件本身是 change of flow 地址或在下两个总线周期出现的 change of flow 地址，在一个触发事件开始 FIFO 之后，它不会存储到 FIFO。在 END 类型跟踪中，如果触发事件是 change of flow，它将作为最后一个 change of flow 条目。

FIFO 也可以用于产生执行过的指令地址路径当调试器未 ARMED。当 ARM=0，读取 DBGFL 导致最近一次读取的操作码地址存储到 FIFO。为了使用该特性，主机调试器读出 FIFO 的地址，通过读取 DBGFH 然后 DBGFL，按照一个定期。最初的 8 个值将放弃因为他们对应于 8 个 DBFFL 读取，必须的初始填充 FIFO。

17.3.3 change of flow 信息

为了最小化存储在 FIFO 中的信息量，只有与指令相关的，导致正常执行顺序的变化被存储。由于知道存储在目标系统的源和目的代码，外部调试器可以重构执行路径通过许多指令从存储在 FIFO 中的转移地址信息。

对于条件分支指令当分支发生时，源地址被存储。因为 BRA 和 BRN 指令是无条件转移的，这些事件不会导致在 FIFO 中存储转移地址。

间接 JMP 和 JSR 指令用于当前 H:X 中的内容确定目的地址，因此调试系统存储运行期内的目的地址为间接的 JMP 和 JSR。例如中断，RTI，RTS，目的地址存储在 FIFO 中作为一个转移地址信息。

17.3.4 标记和强制断点和触发器

标记：标识一个指令操作码，该操作码预取到指令队列，但不会采取任何动作直到该指令实际被 CPU 执行。这种能差别很重要，因为任何从 jump，branch，subroutine 调用，中断中的任何 change of flow 导致一些指令，预取到指令队列中的，被舍弃，不会被执行。

强制类型断点等待当前指令完成，然后响应断点要求。通常进入后台调试模式，而不是执行下面的指令。

标记和强制技术在调试模块中用于两种环境。第一种情况是从 CPU 调试器的断点请求。第二种是匹配信号从调试器的比较器中。当一个标记型断点请求发给 CPU,一个信号进入指令队列和操作码，因此如果当操作码执行，CPU 将有效的用 BGND 操作码代替标记的操作码这样 CPU 进入后台调试模式。而不是执行该标记的操作码。当 TRGSEL 控制位置位，选择标记类型操作，从比较器 A 或 B 的输出被调试模块的一块电路使能，跟踪代码，对调试器产生一个触发如果在比较地址处的操作码被实际执行。有一个独立的操作码跟踪逻辑为每一个比较器，因此更多的比较事件可以跟踪通过某一时刻的指令队列。

17.3.5 触发模式

触发模式控制整个调试器的行为。4 位 TRG 用于选择 9 种触发模式之一。当 TRGSEL=1，比较器的输出必须通过一个操作码跟踪电路传播，在触发 FIFO 行为之前。BEGIN 位选择 FIFO 是否开始存储数据当有资格的触发器被检测到（BEGIN trace）或者 FIFO 存储数据在一个循环方式从它被 ARM，直到有资格的触发器被检测到。

向 ARM 位写 1，调试开始运行，在 DBGS 中的 ARMF 位置位，而 AF 和 BF，CNT 位清零。一个 BEGIN－trace 调试结束当 FIFO 被填满。一个 end trace 结束当选择的触发事件发生。任何调试可以被停止，人工的，通过向 ARM 位写 1。

在所有的触发器模式中除了只事件模式，FIFO 存储 change of flow 地址。在只事件模式中，FIFO 存储数据到 FIFO 的低 8 位。

BEGIN 位在 event－only 触发模式下忽略，所有调试运行为 begin 类型跟踪。当 TRGSEL=1 选择操作码预取触发器，没必要使用 R/W 在比较器因此操作码标记只应用于操作码预取，即总是读周期。不常用，TRGSEL=1 而运行一个 full 模式触发器因此操作码的值通常在一个特定地址。

下面介绍的触发模式主要的比较器条件导致触发。任意比较器通常通过设置 RWAEN 或 RWBEN 为进一步有资格。相应的 RWA 值应该和 R/W 位匹配。从比较器的信号用于请求 CPU 断点当 BRKEN=1，TAG 决定 CPU 要求是标记或强制。

A-only 当地址和比较器 A 中的匹配触发

A OR B 当地址和比较器 A 或 B 中的值匹配触发

A THEN B 当地址匹配寄存器 A 后，再经过 n 个周期，匹配 B 比较器触发；n 为任意值

A AND DATA(full mode) 该触发模式之所以称为 full mode，是因为只有当同一总线周期内，地址，数据和 R/W 都必须匹配时，发生触发事件。比较器 A 检查地址，比较器 B 的低 8 位数据比较数据（高 8 位未使用），R/W 与 RWA 进行比较（前提 RWAEN=1）。

在 full mode 情况下，设定为 tag 类型断点（BRKEN=TAG=1），那么比较器 B 的数据项匹配将被忽略，即当比较器 A 地址匹配时，就会向 CPU 请求 tag 类型断点

A AND NOT B DATA(full mode) 地址必须和比较器 A 匹配，数据必须和比较器 B 的低 8 位不匹配，R/W 位必须和 RWA 匹配（前提 RWAEN=1）。只有在同一总线周期内，该 3 个条件同时成立才会触发。

Event-only B(store data) 每当地址和比较器 B 匹配时触发。该触发事件导致被捕获的数据存储到 FIFO 中。而当 FIFO 被填满时，调试停止。

A THEN EVENT-ONLY B(store data) 当地址匹配比较器 B 后，每当地址和比较器 B 匹配发生触发事件。该触发事件导致被捕获的数据存储到 FIFO 中。而当 FIFO 被填满时，调试停止。

INSIDE RANGE(A<=ADDRESS<=B) 当地址在比较器 A 和比较器 B 范围内时触发事件发生。

OUTSIDE RANGE(ADRESS < A OR ADDRESS >B) 当地址在比较器 A 和比较器 B 范围之外时触发事件发生。

17.3.6 硬件断点

BRKEN=1，触发事件向 CPU 请求产生一个硬件断点。Tag 类型断点将当前操作码加上标记，如果该操作码将要被 CPU 执行时，CPU 执行 BGND 指令进入后台调试模式，而不是执行该指令。Force 类型断点导致 CPU 完成当前指令之后，立即进入后台调试模式。

如果后台调试模式禁止，那么通过 BKGD 引脚传送一个 WRITE_CONTROL 命令，CPU 将执行 SWI 指令，而不会进入后台调试模式

17.3.7 比较器 A 和 B

两个 16 位比较器（A 和 B）

独立的控制位允许你忽略对每个比较器的 R/W。操作码跟踪电路允许你指定一个触发器，如果被指定地址处的操作码实际在执行，

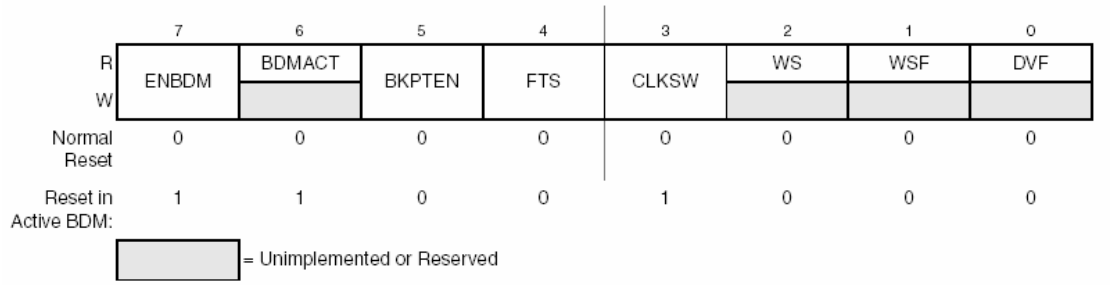
17.4 寄存器定义

17.4.1 BDC 寄存器和控制位

BDC 包括两个寄存器：

- BDC 状态和控制寄存器 BDCSCR
- BDC 断点匹配寄存器 BDCBKPT

17.4.1.1 BDCSCR



	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
W								
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	1	0	0	0

= Unimplemented or Reserved

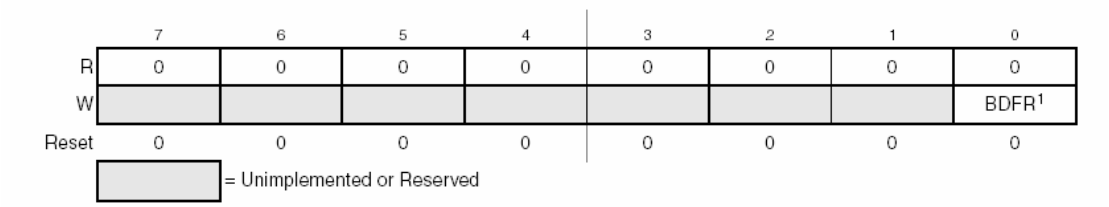
7 ENBDM	BDM 允许位－通常，该位在开始调试之前被主机设置为 1 0：禁止 BDM；1：BDM 通过后台调试命令可以激活
6 BDMACT	后台调试模式活动状态标志 0：BDM 未运行；1：BDM 被激活，等待串口命令
5 BKPTEN	BDC 断点功能允许位 0：禁止 BDC 断点；1：允许 BDC 断点
4 FTS	Force/tag 选择－ 0：与 BDCBKPT 寄存器中的值相匹配的 CPU 地址总线对应的操作码被标记，当该标记的操作码为 CPU 执行指令队列的最后一条指令时，CPU 进入后台调试模式，而不是执行该操作码 1：当 CPU 地址总线匹配 BDCBKPT 寄存器中的值时，向 CPU 请求一个断点，故地址不必有操作码。
3 CLKSW	BDC 通信时钟源选择位 0：BDC 时钟源；1：MCU 总线时钟
2 WS	等待或停止模式－当目标 MCU 处于等待或停止模式，大多数 BDC 命令不再起作用。然而 BACKGROUND 命令可用于强制目标 MCU 从等待或停止模式退出，进入后台调试模式。 0：目标 MCU 运行用户应用程序或处于后台调试模式 1：当目标 MCU 处于等待或停止模式时，BACKGROUND 命令可以唤醒 CPU
1 WSF	等待或停止失败状态位－当目标 CPU 执行 WAIT 或 STOP 命令，此时主机的存储器访问命令会失败，该位置位。 0：存储器访问与 WAIT 或 STOP 指令没有冲突 1：存储器访问命令失败，原因未 CPU 进入等待或停止模式
0 DVF	数据有效状态标志：MC9S08LC60/36/20 未使用该位。

17.4.1.2 BDC 断点匹配寄存器（BDCBKPT）

该 16 位寄存器存放的是 BDC 的硬件断点地址。BKPTEN 和 FTS 控制位用于配置断点逻辑。READ_BKPT 和 WRITE_BKPT 命令分别用于读和写 BDCBKPT 寄存器的值，该寄存器不会被用户程序访问到。

断点通常在运行用户程序之前，后台调试模式中设置。

17.4.2 系统后台调试强制复位寄存器（SBDFR）



0 BDFR	后台调试强制复位位－后台调试命令 WRITE_BYTE 允许主机强制目标系统复位。向该位写 1 强制复位。

17.4.3 DBG 寄存器器和控制位

17.4.3.1 调试比较器 A 高位寄存器（DBGCAH）

该寄存器为比较器 A 的高 8 位值。

17.4.3.2 调试比较器 A 低位寄存器（DBGCAL）

该寄存器为比较器 A 的低 8 位值。

17.4.3.3 调试比较器 B 高位寄存器（DBGCBH）

该寄存器为比较器 B 的高 8 位值。

17.4.3.4 调试比较器 B 低位寄存器（DBGCBL）

该寄存器为比较器 B 的高低位值。

17.4.3.5 调试 FIFO 高位寄存器（DBGFH）

该寄存器只读，用于访问 FIFO 的高 8 位。向该寄存器写操作没有意义。在只事件触发模式时，FIFO 只保存数据到 FIFO 的低 8 位，所以读取该寄存器将返回 0x00。

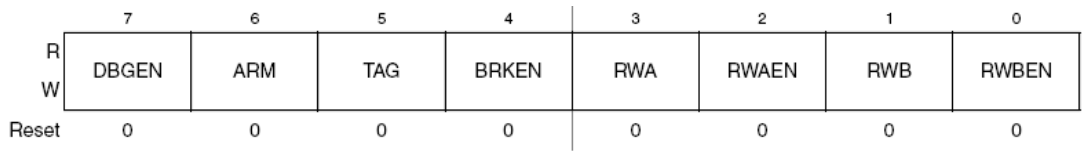
读取 DBGFH 不会导致 FIFO 转变到下一个字。当从 FIFO 读出 16 位的字时，在读取 DBGFL 之前读取 DBGFH。因为读取 DBGFL 导致 FIFO 推进下一字信息。

17.4.3.6 调试 FIFO 低位寄存器（DBGFL）

只读的 FIFO 的低 8 位。读 DBGFL 导致 FIFO 移向下一个可用的信息字。当调试模块工作于只事件模式，只有 8 位数据保存进 FIFO。当读取后，DBGFL 从 FIFO 装入下面的字节。

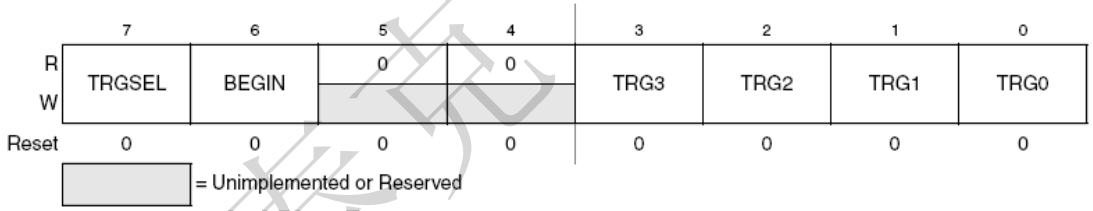
如果调试未 ARM，读取 DBGFL 导致 CPU 最近获取操作码的地址存储到 FIFO 的最后位置。通过定期读取 DBGFH，然后 DBGFL，外部主机软件可以获取程序执行轨迹。从 FIFO 读取 8 次之后，第 9 次读取时将返回信息，该信息最初的读取结果。

17.4.3.7 调试控制寄存器



7 DBGEN	DBG 模块允许位－如果 MCU 安全状态下，DBGEN 不能设置为 1。 0：禁止 DBG 模块；1：DBG 模块允许
6 ARM	Arm 控制位－控制调试器是否进行比较和存储信息到 FIFO。 0：调试器未 ARMED；1：调试器 ARMED
5 TAG	断点选择 0：强制 CPU 停止；1：TAG 类型断点
4 BRKEN	断点使能－触发事件是否对 CPU 产生一个中断。触发事件导致信息存储到 FIFO，不会产生一个断点要求。对于一个跟踪的结束，CPU 断点要求，当比较器和 R/W 条件满足。对于跟踪的开始，CPU 断点要求当 FIFO 变满。 0：CPU 断点要求禁止；1：触发一个断点要求对 CPU
3 RWA	A 比较器读/写允许位。当 REAEN＝1，该位决定是读还是写比较器 A 0：比较器 A 只能匹配写周期 1：比较器 A 只能匹配读周期
2 RWAEN	比较 A 读/写触发允许位 0：禁止；1：允许
1 RWB	B 比较器读/写允许位。当 REBEN＝1，该位决定是读还是写比较器 B 0：比较器 B 只能匹配写周期 1：比较器 B 只能匹配读周期
0 RWBEN	比较 B 读/写触发允许位 0：禁止；1：允许

17.4.3.8 调试触发寄存器



7 TRGSEL	触发类型－如果操作码在匹配地址被执行，触发事件只告知 FIFO 逻辑，如果 TRGSEL=1，从比较器 A 或 B 的 0：访问比较地址触发；1：如果比较地址内的操作码被执行
6 BEGIN	0：END 类型的 TRACE，此时，DBG 模块被启动后，将持续向 FIFO 里填充相关的调试信息，触发条件满足时产生一个 DBG 断点，FIFO 存储断点到达之前最新的 8 次数据。 1:触发器启动数据存储（BEGIN 型跟踪）：在触发器触发后，DBG 将进行满足条件的信息捕获，并将其保存到 FIFO 中，FIFO 填满时，产生一个 DBG 断点
3：0 TRG[3:0]	触发类型 0000：地址匹配 A 时 0001：匹配 A 或 B 时 0010：匹配 A 若干周期后，匹配 B 0011：匹配 B 时，存数据到 FIFO；FIFO 填满时，结束 0100：匹配 A 若干周期后，若匹配 B，将数据存储到 FIFO，此次调试结束 0101：A AND B data(full mode)

	0110：A AND NOT B data（full mode） 0111：A AND NOT B data（full mode） 0111：匹配 A 到 B 之间的地址 1000：匹配 A 和 B 之外的地址 1001～1111：无

17.4.3.9 调试状态寄存器（DBGS）

	7	6	5	4	3	2	1	0
R	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

7 AF	匹配 A 标志 0：比较器 A 不匹配；1：比较器 A 匹配
6 BF	匹配 B 标志 0：比较器 A 不匹配；1：比较器 A 匹配
5 ARMF	ARM 标志
3：0 CNT[3:0]	FIFO 有效数目－在调试运行开始时，该位清零，在调试结束表明 FIFO 中的字数。 0000：无有效字数 0001：有效字数＝1 0010：有效字数＝2 0011：有效字数＝3 0100：有效字数＝4 0101：有效字数＝5 0110：有效字数＝6 0111：有效字数＝7 1000：有效字数＝8

附录 A

电气参数

附录 B

订货信息及芯片封装尺寸