

# KEA128 子系列参考手册

**支持:** S9KEAZ64AMLK(R)、S9KEAZ128AMLK(R)、  
S9KEAZ64AVLK(R)、S9KEAZ128AVLK(R)、S9KEAZ64ACLK(R)、  
S9KEAZ128ACLK(R)、S9KEAZ64AMLH(R)、S9KEAZ128AMLH(R)、  
S9KEAZ64AVLH(R)、S9KEAZ128AVLH(R)、S9KEAZ64ACLH(R)和  
S9KEAZ128ACLH(R)

Document Number: S9KEA128Z80M48SF0RM  
Rev 2, July 2014





## 内容

小节编号	标题	页
<b>第 1 章 关于本文档</b>		
1.1	概述.....	37
1.1.1	目的.....	37
1.1.2	受众.....	37
1.2	惯例.....	37
1.2.1	编码系统.....	37
1.2.2	印刷符号.....	38
1.2.3	特殊术语.....	38
<b>第 2 章 简介</b>		
2.1	概述.....	39
2.2	模块功能类别.....	39
2.2.1	ARM Cortex-M0+内核模块.....	40
2.2.2	系统模块.....	41
2.2.3	存储器和存储器接口.....	41
2.2.4	时钟.....	42
2.2.5	安全性和完整性模块.....	42
2.2.6	模拟模块.....	42
2.2.7	定时器模块.....	43
2.2.8	通信接口.....	43
2.2.9	人机接口.....	44
2.2.10	可订购部件编号.....	44
<b>第 3 章 芯片配置</b>		
3.1	简介.....	45
3.2	模块间互连.....	45
3.2.1	互连概述.....	45

小节编号	标题	页
3.2.2	模拟基准选项.....	49
3.2.3	ACMP 输出捕捉.....	49
3.2.4	UART0_TX 调制.....	50
3.2.5	UART0/1/2_RX 捕捉.....	50
3.2.6	UART0_RX 滤波器.....	51
3.2.7	RTC 捕捉.....	51
3.2.8	FTM2 软件同步.....	51
3.2.9	ADC 硬件触发.....	52
3.3	内核模块.....	52
3.3.1	ARM Cortex-M0+内核配置.....	52
3.3.1.1	ARM Cortex M0+内核 .....	53
3.3.1.2	总线、互连和接口.....	54
3.3.1.3	系统节拍定时器.....	54
3.3.1.4	内核特权级别.....	54
3.3.1.5	缓存器.....	55
3.3.2	可嵌套向量的中断控制器(NVIC)配置.....	55
3.3.2.1	中断优先级.....	55
3.3.2.2	不可屏蔽中断.....	55
3.3.2.3	中断通道分配.....	56
3.3.3	异步唤醒中断控制器(AWIC)配置.....	58
3.3.3.1	唤醒源.....	58
3.4	系统模块.....	59
3.4.1	SIM 配置.....	59
3.4.2	PMC 配置.....	60
3.4.3	MCM 配置.....	61
3.4.4	简化交叉开关配置.....	61
3.4.4.1	简化交叉开关主机分配.....	62
3.4.4.2	交叉开关从机分配.....	62

小节编号	标题	页
3.4.5	外设桥配置.....	63
3.4.5.1	外设桥的数量.....	63
3.4.5.2	存储器映像.....	63
3.5	系统安全性.....	64
3.5.1	CRC 配置.....	64
3.5.2	WDOG 配置.....	64
3.5.2.1	WDOG 时钟.....	65
3.5.2.2	WDOG 操作.....	65
3.6	时钟模块.....	66
3.6.1	ICS 配置.....	66
3.6.1.1	时钟门控.....	67
3.6.2	OSC 配置.....	67
3.7	存储器和存储器接口.....	68
3.7.1	Flash 存储器配置.....	68
3.7.1.1	Flash 存储器大小.....	68
3.7.1.2	Flash 存储器映像.....	69
3.7.1.3	备用非易失性 IRC 用户微调说明.....	69
3.7.1.4	Flash 加密.....	70
3.7.1.5	擦除所有 Flash 内容.....	70
3.7.2	Flash 存储器控制器配置.....	70
3.7.3	SRAM 配置.....	71
3.7.3.1	SRAM 大小.....	71
3.7.3.2	SRAM 范围.....	72
3.7.3.3	SRAM 位操作.....	73
3.8	模拟.....	74
3.8.1	12 位模数转换器(ADC)配置.....	74
3.8.1.1	ADC 实例化信息.....	74
3.8.1.2	ADC0 连接/通道分配.....	75
3.8.1.3	ADC 模拟电源和基准连接.....	76

小节编号	标题	页
3.8.1.4	温度传感器和带隙基准.....	76
3.8.1.5	备选时钟.....	76
3.8.2	ACMP 配置.....	77
3.8.2.1	ACMP 概述.....	77
3.8.2.2	ACMP 互相连接.....	78
3.8.2.3	Stop 模式下的 ACMP.....	78
3.9	定时器.....	79
3.9.1	FlexTimer 配置.....	79
3.9.1.1	FTM 概述.....	79
3.9.1.2	FTM 时钟选项.....	81
3.9.1.3	FTM 互连.....	81
3.9.1.4	FTM 中断.....	82
3.9.2	PIT 配置.....	82
3.9.2.1	PIT 概述.....	82
3.9.2.2	PIT 互相连接.....	83
3.9.3	RTC 配置.....	83
3.9.3.1	RTC 概述.....	83
3.9.3.2	RTC 互连.....	83
3.9.4	PWT 配置.....	84
3.9.4.1	PWT 概述.....	84
3.9.4.2	PWT 互连.....	85
3.10	通信接口.....	85
3.10.1	SPI 配置.....	85
3.10.1.1	SPI 概述.....	86
3.10.2	I2C 配置.....	86
3.10.2.1	I2C 概述.....	86
3.10.2.2	I2C0 4 线式接口特性.....	87
3.10.3	UART 配置.....	87
3.10.3.1	UART 概述.....	88

小节编号	标题	页
3.10.3.2	UART 互连.....	88
3.10.4	MSCAN 配置.....	89
3.10.4.1	MSCAN 概述.....	89
3.10.4.2	MSCAN 时钟源.....	89
3.10.4.3	MSCAN 唤醒中断和去抖滤波器.....	90
3.11	人机接口(HMI).....	90
3.11.1	GPIO 配置.....	90
3.11.1.1	GPIO 概述.....	90
3.11.2	KBI 配置.....	91
3.11.2.1	KBI 概述.....	91
3.11.2.2	KBI 分配.....	91
3.11.3	IRQ 配置.....	92
3.11.3.1	IRQ 分配.....	92

## 第 4 章 存储器映像

4.1	简介.....	93
4.2	系统存储器映像.....	93
4.3	位带区别名.....	94
4.4	位操作引擎.....	95
4.5	系统 ROM 存储器映像.....	95
4.5.1	入口 (ROM_ENTRY $n$ ).....	97
4.5.2	表格标记寄存器结束 (ROM_TABLEMARK).....	97
4.5.3	系统访问寄存器 (ROM_SYSACCESS).....	98
4.5.4	外设 ID 寄存器 (ROM_PERIPHID $n$ ).....	98
4.5.5	组件 ID 寄存器 (ROM_COMPID $n$ ).....	99
4.6	外设桥(AIPS-Lite)存储器映像.....	99
4.6.1	先写后读序列和存储器操作所需的串行化.....	100
4.6.2	外设桥(AIPS-Lite)存储器映像.....	100
4.7	专用外设总线(PPB)存储器映像.....	104

小节编号	标题	页
<b>第 5 章 时钟分布</b>		
5.1	简介.....	105
5.2	编程模型.....	105
5.3	器件时钟连接示意图.....	105
5.4	时钟定义.....	106
5.4.1	器件时钟汇总.....	107
5.4.2	时钟分布.....	108
5.5	内部时钟源.....	109
5.6	外部时钟源.....	109
5.7	时钟选通.....	110
5.8	模块时钟.....	110
5.8.1	FTM 和 PWT 计时.....	112
<b>第 6 章 复位与引导</b>		
6.1	简介.....	115
6.2	复位.....	115
6.2.1	上电复位(POR).....	115
6.2.2	系统复位源.....	116
6.2.2.1	外部引脚复位(RESET).....	116
6.2.2.2	低压检测(LVD).....	116
6.2.2.3	WDOG 定时器.....	117
6.2.2.4	ICS 时钟丢失(LOC).....	117
6.2.2.5	Stop 模式应答错误(SACKERR) .....	117
6.2.2.6	软件复位(SW).....	117
6.2.2.7	死锁复位(LOCKUP).....	118
6.2.2.8	MDM-AP 系统复位请求.....	118
6.2.3	MCU 复位.....	118
6.2.3.1	仅 POR .....	118

小节编号	标题	页
6.2.3.2	芯片 POR .....	118
6.2.3.3	早期芯片复位 .....	118
6.2.3.4	芯片复位 .....	118
6.3	引导.....	119
6.3.1	引导源.....	119
6.3.2	引导序列.....	119
<b>第 7 章 电源管理</b>		
7.1	简介.....	121
7.2	功耗模式.....	121
7.3	进入和退出低功耗模式.....	122
7.4	低功耗模式下的模块操作.....	122
<b>第 8 章 加密</b>		
8.1	简介.....	125
8.2	Flash 加密.....	125
8.3	与其他模块之间的安全性交互.....	125
8.3.1	与调试之间的加密交互.....	125
<b>第 9 章 调试</b>		
9.1	简介.....	127
9.2	调试端口引脚说明.....	127
9.3	SWD 状态和控制寄存器.....	127
9.3.1	MDM-AP 状态寄存器.....	129
9.3.2	MDM-AP 控制寄存器.....	129
9.4	调试复位.....	130
9.5	低功耗模式下的调试.....	130
9.6	调试和加密.....	131

小节编号	标题	页
<b>第 10 章 信号多路复用和信号说明</b>		
10.1	简介	133
10.2	引脚分配	133
10.2.1	信号多路复用和引脚分配	133
10.2.2	器件引脚分配	136
10.3	模块信号描述表	137
10.3.1	内核模块	137
10.3.2	系统模块	138
10.3.3	时钟模块	138
10.3.4	模拟	138
10.3.5	定时器模块	139
10.3.6	通信接口	140
10.3.7	人机接口(HMI)	141
<b>第 11 章 端口控制(PORT)</b>		
11.1	简介	143
11.2	端口数据和数据方向	145
11.3	内部上拉使能	146
11.4	输入去抖滤波器设置	146
11.5	大电流驱动	147
11.6	Stop 模式下的引脚特性	147
11.7	端口数据寄存器	147
11.7.1	端口滤波寄存器 0 (PORT_IOFLT0)	148
11.7.2	端口滤波寄存器 1 (PORT_IOFLT1)	151
11.7.3	端口上拉使能寄存器 0 (PORT_PUE0)	152
11.7.4	端口上拉使能寄存器 1 (PORT_PUE1)	157
11.7.5	端口上拉使能寄存器 2 (PORT_PUE2)	161
11.7.6	端口大电流驱动使能寄存器 (PORT_HDRVE)	162

小节编号	标题	页
<b>第 12 章 系统集成模块 (SIM)</b>		
12.1	简介 .....	165
12.1.1	特性 .....	165
12.2	存储器映像和寄存器定义 .....	165
12.2.1	系统复位状态和 ID 寄存器 (SIM_SRSID) .....	166
12.2.2	系统选项寄存器 0 (SIM_SOPT0) .....	169
12.2.3	系统选项寄存器 (SIM_SOPT1) .....	172
12.2.4	引脚选择寄存器 0 (SIM_PINSEL0) .....	174
12.2.5	引脚选择寄存器 1 (SIM_PINSEL1) .....	176
12.2.6	系统时钟选通控制寄存器 (SIM_SCGC) .....	178
12.2.7	通用唯一标识符低位寄存器 (SIM_UUIDL) .....	182
12.2.8	通用唯一标识符中低位寄存器 (SIM_UUIDML) .....	182
12.2.9	通用唯一标识符中高位寄存器 (SIM_UUIDMH) .....	183
12.2.10	时钟分频器寄存器 (SIM_CLKDIV) .....	183
12.3	功能说明 .....	184
<b>第 13 章 电源管理控制器(PMC)</b>		
13.1	简介 .....	185
13.2	低电压检测(LVD)系统 .....	185
13.2.1	上电复位(POR)操作 .....	186
13.2.2	LVD 复位操作 .....	186
13.2.3	Stop 模式下的 LVD 使能 .....	186
13.2.4	低压警报(LVW) .....	186
13.3	带隙基准源 .....	187
13.4	存储器映像和寄存器说明 .....	187
13.4.1	系统电源管理状态和控制 1 寄存器 (PMC_SPMSC1) .....	187
13.4.2	系统电源管理状态和控制 2 寄存器 (PMC_SPMSC2) .....	189

小节编号	标题	页
<b>第 14 章 杂项控制模块(MCM)</b>		
14.1	简介 .....	191
14.1.1	特性 .....	191
14.2	存储器映像/寄存器说明 .....	191
14.2.1	交叉开关(AXBS)从机配置 (MCM_PLASC) .....	192
14.2.2	交叉开关(AXBS)主机配置 (MCM_PLAMC) .....	192
14.2.3	平台控制寄存器 (MCM_PLACR) .....	193
<b>第 15 章 外设桥(AIPS-Lite)</b>		
15.1	简介 .....	197
15.1.1	特性 .....	197
15.1.2	一般操作 .....	197
15.2	功能说明 .....	198
15.2.1	访问支持 .....	198
<b>第 16 章 WDOG</b>		
16.1	简介 .....	199
16.1.1	特性 .....	199
16.1.2	结构框图 .....	200
16.2	存储器映像和寄存器定义 .....	201
16.2.1	WDOG 控制和状态寄存器 1 (WDOG_CS1) .....	201
16.2.2	WDOG 控制和状态寄存器 2 (WDOG_CS2) .....	203
16.2.3	WDOG 计数器寄存器：高位 (WDOG_CNTH) .....	203
16.2.4	WDOG 计数器寄存器：低位 (WDOG_CNTL) .....	204
16.2.5	WDOG 定时溢出值寄存器：高位 (WDOG_TOVALH) .....	205
16.2.6	WDOG 定时溢出值寄存器：低位 (WDOG_TOVALL) .....	205
16.2.7	WDOG 窗口寄存器：高位 (WDOG_WINH) .....	206
16.2.8	WDOG 窗口寄存器：低位 (WDOG_WINL) .....	206

小节编号	标题	页
16.3 功能说明.....		206
16.3.1 WDOG 刷新机制.....		207
16.3.1.1 窗口模式.....		207
16.3.1.2 刷新 WDOG.....		208
16.3.1.3 示例代码：刷新 WDOG.....		208
16.3.2 配置 WDOG.....		208
16.3.2.1 重新配置 WDOG.....		209
16.3.2.2 解锁 WDOG.....		209
16.3.2.3 代码示例：重新配置 WDOG.....		209
16.3.3 时钟源.....		210
16.3.4 使用中断延迟复位.....		210
16.3.5 备用复位.....		211
16.3.6 Debug 模式及低功耗模式下的功能.....		211
16.3.7 WDOG 快速测试.....		211
16.3.7.1 测试计数器的每一个字节.....		212
16.3.7.2 进入用户模式.....		213

## 第 17 章 位操作引擎(BME)

17.1 简介.....	215
17.1.1 概述.....	215
17.1.2 特性.....	216
17.1.3 工作模式.....	217
17.2 存储器映像和寄存器定义.....	217
17.3 功能说明.....	217
17.3.1 BME 已修饰存储.....	217
17.3.1.1 已修饰存储逻辑与(AND).....	219
17.3.1.2 已修饰存储逻辑或(OR).....	220
17.3.1.3 已修饰存储逻辑异或(XOR).....	221
17.3.1.4 已修饰存储位字段插入(BFI).....	222

小节编号	标题	页
17.3.2	BME 已修饰加载.....	223
17.3.2.1	已修饰加载：1 位加载 - 置位(LAS1).....	226
17.3.2.2	已修饰加载无符号位字段提取(UBFX).....	227
17.3.3	已修饰地址和 GPIO 访问的更多详情.....	228
17.4	应用信息.....	229

## 第 18 章 Flash 存储器模块(FTMRE)

18.1	简介.....	231
18.2	特性.....	231
18.2.1	Flash 存储器特性.....	231
18.2.2	Flash 模块其他特性.....	232
18.3	功能说明.....	232
18.3.1	工作模式.....	232
18.3.1.1	Wait 模式.....	232
18.3.1.2	Stop 模式.....	232
18.3.2	Flash 存储器映射.....	232
18.3.3	系统复位之后的 Flash 初始化.....	232
18.3.4	Flash 命令操作.....	233
18.3.4.1	写 FCLKDIV 寄存器.....	234
18.3.4.2	命令写入序列.....	236
18.3.5	Flash 中断.....	238
18.3.5.1	Flash 中断操作的说明.....	238
18.3.6	保护.....	238
18.3.7	加密.....	242
18.3.7.1	使用后门密钥访问解密 MCU.....	243
18.3.7.2	使用 SWD 解密 MCU.....	244
18.3.7.3	模式与加密状态对 Flash 命令可用性的影响.....	244
18.3.8	Flash 命令.....	244
18.3.8.1	Flash 命令.....	244

小节编号	标题	页
18.3.9	Flash 命令汇总.....	245
18.3.9.1	“擦除验证所有数据块”命令.....	246
18.3.9.2	“擦除验证数据块”命令.....	246
18.3.9.3	“擦除检验 Flash 段”命令.....	247
18.3.9.4	“读取一次”命令.....	248
18.3.9.5	“编程 Flash”命令.....	248
18.3.9.6	“编程一次”命令.....	249
18.3.9.7	“擦除所有区块”命令.....	250
18.3.9.8	调试器整体擦除请求.....	251
18.3.9.9	“擦除 Flash 区块”命令.....	252
18.3.9.10	“擦除 Flash 扇区”命令.....	252
18.3.9.11	“解密 Flash”命令.....	253
18.3.9.12	“检验后门访问密钥”命令.....	254
18.3.9.13	“设置用户裕量水平”命令.....	255
18.3.9.14	“设置出厂裕量水平”命令.....	256
18.3.9.15	“配置 NVM”命令.....	257
18.4	存储器映像和寄存器定义.....	259
18.4.1	Flash CCOB 索引寄存器 (FTMRE_FCCOBIX).....	260
18.4.2	Flash 安全寄存器 (FTMRE_FSEC).....	260
18.4.3	Flash 时钟分频器寄存器 (FTMRE_FCLKDIV).....	261
18.4.4	Flash 状态寄存器 (FTMRE_FSTAT).....	262
18.4.5	Flash 配置寄存器 (FTMRE_FCNFG).....	263
18.4.6	Flash 通用命令对象寄存器：低位 (FTMRE_FCCOBLO).....	264
18.4.7	Flash 通用命令对象寄存器：高位 (FTMRE_FCCOBHI).....	264
18.4.8	Flash 保护寄存器 (FTMRE_FPROT).....	265
18.4.9	Flash 选项寄存器 (FTMRE_FOPT).....	266

小节编号	标题	页
<b>第 19 章 Flash 存储器控制器(FMC)</b>		
19.1	简介 .....	267
19.1.1	概述 .....	267
19.1.2	特性 .....	267
19.2	工作模式 .....	268
19.3	外部信号说明 .....	268
19.4	存储器映像和寄存器说明 .....	268
19.5	功能说明 .....	268
<b>第 20 章 内部时钟源(ICS)</b>		
20.1	简介 .....	271
20.1.1	特性 .....	271
20.1.2	结构框图 .....	271
20.1.3	工作模式 .....	272
20.1.3.1	FLL 内部启用(FEI) .....	272
20.1.3.2	FLL 外部启用(FEE)模式 .....	272
20.1.3.3	FLL 内部旁路(FBI) .....	272
20.1.3.4	FLL 内部旁路低功耗(FBILP) .....	273
20.1.3.5	FLL 外部旁路(FBE) .....	273
20.1.3.6	FLL 外部旁路低功耗(FBELP) .....	273
20.1.3.7	停止 (STOP) .....	273
20.2	外部信号说明 .....	273
20.3	寄存器定义 .....	273
20.3.1	ICS 控制寄存器 1 (ICS_C1) .....	274
20.3.2	ICS 控制寄存器 2 (ICS_C2) .....	275
20.3.3	ICS 控制寄存器 3 (ICS_C3) .....	276
20.3.4	ICS 控制寄存器 4 (ICS_C4) .....	276
20.3.5	ICS 状态寄存器 (ICS_S) .....	277

小节编号	标题	页
20.4 功能说明.....		278
20.4.1 工作模式.....		278
20.4.1.1 FLL 内部启用(FEI).....		279
20.4.1.2 FLL 外部启用(FEE).....		279
20.4.1.3 FLL 内部旁路(FBI).....		280
20.4.1.4 FLL 内部旁路低功耗(FBILP).....		280
20.4.1.5 FLL 外部旁路(FBE).....		280
20.4.1.6 FLL 外部旁路低功耗(FBELP).....		280
20.4.1.7 停止.....		281
20.4.2 模式切换.....		281
20.4.3 总线频率分频器.....		281
20.4.4 低功耗字段的使用.....		281
20.4.5 内部参考时钟.....		282
20.4.6 固定频率时钟.....		282
20.4.7 FLL 锁定和时钟监视器.....		282
20.4.7.1 FLL 时钟锁定.....		282
20.4.7.2 外部基准时钟监视器.....		283
20.5 初始化/应用信息.....		283
20.5.1 初始化 FEI 模式.....		283
20.5.2 初始化 FBI 模式.....		284
20.5.3 初始化 FEE 模式.....		284
20.5.4 初始化 FBE 模式.....		284
<b>第 21 章</b> <b>振荡器(OSC)</b>		
21.1 简介.....		287
21.1.1 概述.....		287
21.1.2 特性和模式.....		287
21.1.3 结构框图.....		287
21.2 信号说明.....		288

小节编号	标题	页
21.3	外部晶振/谐振器连接.....	289
21.4	外部时钟连接.....	290
21.5	存储器映像和寄存器说明.....	290
21.5.1	OSC 控制寄存器 (OSC_CR).....	291
21.6	功能说明.....	292
21.6.1	OSC 模块状态.....	292
21.6.1.1	关.....	293
21.6.1.2	振荡器启动.....	293
21.6.1.3	振荡器稳定.....	293
21.6.1.4	外部时钟模式.....	293
21.6.2	OSC 模块模式.....	293
21.6.2.1	低频、高增益模式.....	294
21.6.2.2	低频、低功耗模式.....	294
21.6.2.3	高频、高增益模式.....	294
21.6.2.4	高频、低功耗模式.....	295
21.6.3	计数器寄存器.....	295
21.6.4	基准时钟的引脚要求.....	295

## 第 22 章 循环冗余校验(CRC)

22.1	简介.....	297
22.1.1	特性.....	297
22.1.2	结构框图.....	297
22.1.3	工作模式.....	298
22.1.3.1	Run 模式.....	298
22.1.3.2	低功耗模式 (Wait 或 Stop 模式) .....	298
22.2	存储器映像和寄存器说明.....	298
22.2.1	CRC 数据寄存器 (CRC_DATA).....	299
22.2.2	CRC 多项式寄存器 (CRC_GPOLY).....	300
22.2.3	CRC 控制寄存器 (CRC_CTRL).....	300

小节编号	标题	页
22.3 功能说明.....		301
22.3.1 CRC 初始化/重新初始化.....		301
22.3.2 CRC 计算.....		302
22.3.2.1 16 位 CRC.....		302
22.3.2.2 32 位 CRC.....		302
22.3.3 转置特性.....		303
22.3.3.1 转置类型.....		303
22.3.4 CRC 结果补码.....		304
<b>第 23 章 外部中断(IRQ)</b>		
23.1 简介.....		305
23.2 特性.....		305
23.2.1 引脚配置选项.....		306
23.2.2 边沿和电平有效.....		307
23.3 外部中断引脚请求寄存器.....		307
23.3.1 外部中断引脚请求状态和控制寄存器 (IRQ_SC).....		307
<b>第 24 章 模数转换器(ADC)</b>		
24.1 简介.....		309
24.1.1 特性.....		309
24.1.2 结构框图.....		310
24.2 外部信号说明.....		310
24.2.1 模拟电源(VDDA).....		311
24.2.2 模拟接地(VSSA).....		311
24.2.3 高基准电压 (VREFH).....		311
24.2.4 低基准电压 (VREFL).....		311
24.2.5 模拟通道输入(ADx).....		311
24.3 ADC 控制寄存器.....		312
24.3.1 状态和控制寄存器 1 (ADC_SC1).....		312

小节编号	标题	页
24.3.2	状态和控制寄存器 2 (ADC_SC2).....	314
24.3.3	状态和控制寄存器 3 (ADC_SC3).....	316
24.3.4	状态和控制寄存器 4 (ADC_SC4).....	317
24.3.5	转换结果寄存器 (ADC_R).....	318
24.3.6	比较值寄存器 (ADC_CV).....	319
24.3.7	引脚控制 1 寄存器 (ADC_APCTL1).....	319
24.3.8	状态和控制寄存器 5 (ADC_SC5).....	320
24.4	功能说明.....	321
24.4.1	时钟选择和分频控制.....	321
24.4.2	输入选择和引脚控制.....	321
24.4.3	硬件触发.....	322
24.4.4	转换控制.....	322
24.4.4.1	发起转换.....	322
24.4.4.2	完成转换.....	323
24.4.4.3	中止转换.....	323
24.4.4.4	功率控制.....	323
24.4.4.5	采样时间和总转换时间.....	323
24.4.5	自动比较功能.....	325
24.4.6	FIFO 操作.....	325
24.4.7	MCUWait 模式下的操作.....	330
24.4.8	MCU Stop 模式下的操作.....	330
24.4.8.1	采用 Stop 模式并禁用 ADACK.....	330
24.4.8.2	采用 Stop 模式并启用 ADACK.....	330
24.5	初始化信息.....	331
24.5.1	ADC 模块初始化示例.....	331
24.5.1.1	初始化序列.....	331

小节编号	标题	页
24.5.1.2	伪代码示例.....	331
24.5.2	ADC FIFO 模块初始化示例.....	332
24.5.2.1	伪代码示例.....	332
24.6	应用信息.....	333
24.6.1	外部引脚和布线.....	333
24.6.1.1	模拟电源引脚.....	333
24.6.1.2	模拟基准引脚.....	334
24.6.1.3	模拟输入引脚.....	334
24.6.2	误差来源.....	335
24.6.2.1	采样误差.....	335
24.6.2.2	引脚漏电误差.....	335
24.6.2.3	噪声性误差.....	335
24.6.2.4	编码宽度和量化误差.....	336
24.6.2.5	线性误差.....	337
24.6.2.6	编码抖动、非单调性和失码.....	337

## 第 25 章 模拟比较器 (ACMP)

25.1	简介.....	339
25.1.1	特性.....	339
25.1.2	工作模式.....	339
25.1.2.1	Wait 模式下的操作.....	339
25.1.2.2	Stop 模式下的操作.....	340
25.1.2.3	Debug 模式下的操作.....	340
25.1.3	结构框图.....	340
25.2	外部信号说明.....	341
25.3	存储器映像和寄存器定义.....	341
25.3.1	ACMP 控制和状态寄存器 (ACMPx_CS).....	341
25.3.2	ACMP 控制寄存器 0 (ACMPx_C0).....	342
25.3.3	ACMP 控制寄存器 1 (ACMPx_C1).....	343

小节编号	标题	页
25.3.4	ACMP 控制寄存器 2 (ACMPx_C2).....	343
25.4	功能说明.....	344
25.5	ACMP 的设置和操作.....	345
25.6	复位.....	345
25.7	中断.....	345

## 第 26 章 FlexTimer 模块(FTM)

26.1	简介.....	347
26.1.1	FlexTimer 的基本理念.....	347
26.1.2	特性.....	348
26.1.3	工作模式.....	349
26.1.4	结构框图.....	349
26.2	FTM 信号说明.....	351
26.3	存储器映像和寄存器定义.....	351
26.3.1	存储器映像.....	351
26.3.2	寄存器说明.....	351
26.3.3	状态和控制寄存器 (FTMx_SC).....	355
26.3.4	计数器寄存器 (FTMx_CNT).....	356
26.3.5	模数寄存器 (FTMx_MOD).....	357
26.3.6	通道(n)状态和控制寄存器 (FTMx_CnSC).....	358
26.3.7	通道(n)值寄存器 (FTMx_CnV).....	360
26.3.8	计数器初始值寄存器 (FTMx_CNTIN).....	360
26.3.9	捕捉和比较状态寄存器 (FTMx_STATUS).....	361
26.3.10	特性模式选择寄存器 (FTMx_MODE).....	363
26.3.11	同步寄存器 (FTMx_SYNC).....	364
26.3.12	通道输出的初始状态寄存器 (FTMx_OUTINIT).....	366
26.3.13	输出屏蔽寄存器 (FTMx_OUTMASK).....	368
26.3.14	已连接通道功能寄存器 (FTMx_COMBINE).....	370
26.3.15	死区时间插入控制 (FTMx_DEADTIME).....	374

小节编号	标题	页
26.3.16	FTM 外部触发寄存器 (FTMx_EXTTRIG).....	375
26.3.17	通道极性寄存器 (FTMx_POL).....	377
26.3.18	故障模式状态寄存器 (FTMx_FMS).....	380
26.3.19	输入捕捉滤波器控制寄存器 (FTMx_FILTER).....	382
26.3.20	故障控制寄存器 (FTMx_FLTCTRL).....	383
26.3.21	配置寄存器 (FTMx_CONF).....	385
26.3.22	FTM 故障输入极性寄存器 (FTMx_FLTPOL).....	386
26.3.23	同步配置寄存器 (FTMx_SYNCONF).....	387
26.3.24	FTM 反相控制寄存器 (FTMx_INVCTRL).....	389
26.3.25	FTM 软件输出控制寄存器 (FTMx_SWOCTRL).....	390
26.3.26	FTM PWM 加载寄存器 (FTMx_PWMLOAD).....	392
26.4	功能说明.....	393
26.4.1	时钟源.....	394
26.4.1.1	计数器时钟源.....	394
26.4.2	预分频器.....	395
26.4.3	计数器寄存器.....	395
26.4.3.1	向上计数.....	395
26.4.3.2	向上-向下计数.....	398
26.4.3.3	自由运行计数器.....	399
26.4.3.4	计数器复位.....	400
26.4.3.5	TOF 置位.....	400
26.4.4	输入捕捉模式.....	401
26.4.4.1	输入捕捉模式的滤波器.....	401
26.4.5	输出比较模式.....	403
26.4.6	边沿对齐 PWM (EPWM) 模式.....	404
26.4.7	中心对齐 PWM (CPWM) 模式.....	406
26.4.8	组合模式.....	407
26.4.8.1	不对称 PWM.....	415
26.4.9	互补模式.....	415

小节编号	标题	页
26.4.10	通过写缓存更新的寄存器.....	416
26.4.10.1	CNTIN 寄存器更新.....	416
26.4.10.2	MOD 寄存器更新.....	417
26.4.10.3	CnV 寄存器更新.....	417
26.4.11	PWM 同步.....	417
26.4.11.1	硬件触发.....	418
26.4.11.2	软件触发.....	419
26.4.11.3	边界周期和加载点.....	419
26.4.11.4	MOD 寄存器同步.....	420
26.4.11.5	CNTIN 寄存器同步.....	423
26.4.11.6	C(n)V 和 C(n+1)V 寄存器同步.....	423
26.4.11.7	OUTMASK 寄存器同步.....	424
26.4.11.8	INVCTRL 寄存器同步.....	427
26.4.11.9	SWOCTRL 寄存器同步.....	428
26.4.11.10	FTM 计数器同步.....	430
26.4.12	反相.....	432
26.4.13	软件输出控制.....	434
26.4.14	死区插入.....	436
26.4.14.1	死区插入个别案例.....	437
26.4.15	输出屏蔽.....	438
26.4.16	故障控制.....	439
26.4.16.1	自动故障清除.....	441
26.4.16.2	手动故障清除.....	441
26.4.16.3	故障输入极性控制.....	442
26.4.17	极性控制.....	442
26.4.18	初始化.....	443
26.4.19	特性优先级.....	443
26.4.20	通道触发器输出.....	444
26.4.21	初始化触发.....	445

小节编号	标题	页
26.4.22	捕获测试模式.....	447
26.4.23	双沿捕捉模式.....	448
26.4.23.1	一次性捕捉模式.....	449
26.4.23.2	连续捕捉模式.....	450
26.4.23.3	脉宽测量.....	450
26.4.23.4	周期测量.....	452
26.4.23.5	读取一致性机制.....	454
26.4.24	Debug 模式.....	455
26.4.25	中间加载.....	456
26.4.26	全局时基(GTB).....	458
26.4.26.1	使能全局时基(GTB).....	459
26.5	复位概述.....	459
26.6	FTM 中断.....	460
26.6.1	定时器溢出中断.....	461
26.6.2	通道(n)中断.....	461
26.6.3	故障中断.....	461
26.7	初始化流程.....	461

## 第 27 章 脉冲宽度定时器(PWT)

27.1	简介.....	463
27.1.1	特性.....	463
27.1.2	工作模式.....	463
27.1.3	功能框图.....	464
27.2	PWT 信号说明.....	465
27.2.1	PWTIN[3:0] - 脉宽定时器捕捉输入.....	465
27.2.2	ALTCLK- 计数器的备用时钟源.....	465
27.3	存储器映像和寄存器说明.....	465
27.3.1	脉宽定时器寄存器 1 (PWT_R1).....	466
27.3.2	脉宽定时器寄存器 2 (PWT_R2).....	468

小节编号	标题	页
27.4 功能说明.....		469
27.4.1 PWT 计数器和 PWT 时钟预分频器.....		469
27.4.2 边沿检测和捕捉控制 .....		469
27.5 复位.....		472
27.5.1 通用.....		472
27.5.2 复位操作说明.....		472
27.6 中断.....		472
27.6.1 中断操作说明.....		472
27.6.2 应用示例.....		473
<b>第 28 章 周期性中断定时器(PIT)</b>		
28.1 简介.....		475
28.1.1 结构框图.....		475
28.1.2 特性.....		476
28.2 信号说明.....		476
28.3 存储器映像/寄存器说明.....		477
28.3.1 PIT 模块控制寄存器 (PIT_MCR).....		477
28.3.2 定时器加载值寄存器 (PIT_LDVAL $n$ ).....		478
28.3.3 当前定时器值寄存器 (PIT_CVAL $n$ ).....		479
28.3.4 定时器控制寄存器 (PIT_TCTRL $n$ ).....		479
28.3.5 定时器标志寄存器 (PIT_TFLG $n$ ).....		480
28.4 功能说明.....		481
28.4.1 常规操作.....		481
28.4.1.1 定时器.....		481
28.4.1.2 Debug 模式.....		482
28.4.2 中断.....		482
28.4.3 链接定时器.....		482
28.5 初始化和应用信息.....		483
28.6 链接定时器配置示例.....		484

小节编号	标题	页
	<b>第 29 章 实时计数器 (RTC)</b>	
29.1	简介.....	485
29.2	特性.....	485
29.2.1	工作模式.....	485
29.2.1.1	Wait 模式.....	485
29.2.1.2	Stop 模式.....	486
29.2.2	结构框图.....	486
29.3	外部信号说明.....	486
29.4	寄存器定义.....	486
29.4.1	RTC 状态和控制寄存器 (RTC_SC).....	487
29.4.2	RTC 模数寄存器 (RTC_MOD).....	488
29.4.3	RTC 计数器寄存器 (RTC_CNT).....	489
29.5	功能说明.....	489
29.5.1	RTC 操作示例.....	490
29.6	初始化/应用信息.....	491
	<b>第 30 章 串行外设接口(SPI)</b>	
30.1	简介.....	493
30.1.1	特性.....	493
30.1.2	工作模式.....	494
30.1.3	结构框图.....	494
30.1.3.1	SPI 系统结构框图.....	495
30.1.3.2	SPI 模块结构框图.....	495
30.2	外部信号说明.....	496
30.2.1	SPSCK — SPI 串行时钟.....	496
30.2.2	MOSI — 主机数据输出, 从机数据输入.....	497
30.2.3	MISO — 主机数据输入, 从机数据输出.....	497
30.2.4	SS — 从机选择.....	497

小节编号	标题	页
30.3	存储器映像/寄存器定义.....	497
30.3.1	SPI 控制寄存器 1 (SPIx_C1).....	498
30.3.2	SPI 控制寄存器 2 (SPIx_C2).....	500
30.3.3	SPI 波特率寄存器 (SPIx_BR).....	501
30.3.4	SPI 状态寄存器 (SPIx_S).....	502
30.3.5	SPI 数据寄存器 (SPIx_D).....	503
30.3.6	SPI 匹配寄存器 (SPIx_M).....	504
30.4	功能说明.....	504
30.4.1	综述.....	504
30.4.2	主机模式.....	505
30.4.3	从机模式.....	506
30.4.4	SPI 时钟格式.....	507
30.4.5	SPI 波特率生成.....	510
30.4.6	特殊功能.....	510
30.4.6.1	SS 输出.....	510
30.4.6.2	双向模式 (MOMI 或 SISO) .....	511
30.4.7	错误条件.....	512
30.4.7.1	模式故障错误.....	512
30.4.8	低功耗模式选项.....	512
30.4.8.1	Run 模式下的 SPI.....	512
30.4.8.2	Wait 模式下的 SPI.....	512
30.4.8.3	Stop 模式下的 SPI.....	513
30.4.9	复位.....	514
30.4.10	中断.....	514
30.4.10.1	MODF.....	514
30.4.10.2	SPRF.....	514
30.4.10.3	SPTEF.....	515
30.4.10.4	SPMF.....	515
30.4.10.5	低功耗模式下的异步中断.....	515

小节编号	标题	页
30.5 初始化/应用信息.....		515
30.5.1 初始化序列.....		516
30.5.2 伪代码示例.....		516
<b>第 31 章 I2C 模块</b>		
31.1 简介.....		519
31.1.1 特性.....		519
31.1.2 工作模式.....		520
31.1.3 结构框图.....		520
31.2 I2C 信号说明.....		521
31.3 存储器映像/寄存器定义.....		522
31.3.1 I2C 地址寄存器 1 (I2Cx_A1).....		523
31.3.2 I2C 分频器寄存器 (I2Cx_F).....		523
31.3.3 I2C 控制寄存器 1 (I2Cx_C1).....		524
31.3.4 I2C 状态寄存器 (I2Cx_S).....		525
31.3.5 I2C 数据 I/O 寄存器 (I2Cx_D).....		527
31.3.6 I2C 控制寄存器 2 (I2Cx_C2).....		528
31.3.7 I2C 可编程输入去抖滤波器寄存器 (I2Cx_FLT).....		529
31.3.8 I2C 范围地址寄存器 (I2Cx_RA).....		530
31.3.9 I2C SMBus 控制和状态寄存器 (I2Cx_SMB).....		530
31.3.10 I2C 地址寄存器 2 (I2Cx_A2).....		532
31.3.11 I2C SCL 低位定时溢出寄存器高电平 (I2Cx_SLTH).....		532
31.3.12 I2C SCL 低位定时溢出寄存器低电平 (I2Cx_SLTL).....		533
31.4 功能说明.....		533
31.4.1 I2C 协议.....		533
31.4.1.1 START 信号.....		534
31.4.1.2 从机地址发送.....		534
31.4.1.3 数据传输.....		535
31.4.1.4 停止信号.....		535

小节编号	标题	页
31.4.1.5	重复开始信号.....	535
31.4.1.6	仲裁程序.....	535
31.4.1.7	时钟同步.....	536
31.4.1.8	握手.....	536
31.4.1.9	时钟拉伸.....	537
31.4.1.10	I2C 分频器和保持值.....	537
31.4.2	10 位地址.....	538
31.4.2.1	主发送器对从接收器进行寻址.....	538
31.4.2.2	主机-发送器对从机-接收器进行寻址.....	539
31.4.3	地址匹配.....	539
31.4.4	系统管理总线规范.....	540
31.4.4.1	定时溢出.....	540
31.4.4.2	FAST ACK 和 NACK.....	541
31.4.5	复位.....	542
31.4.6	中断.....	542
31.4.6.1	字节传输中断.....	543
31.4.6.2	地址检测中断.....	543
31.4.6.3	停止检测中断.....	543
31.4.6.4	退出低功耗/Stop 模式.....	543
31.4.6.5	仲裁丢失中断.....	543
31.4.6.6	SMBus 中的定时溢出中断.....	544
31.4.7	可编程输入去抖滤波器.....	544
31.4.8	地址匹配唤醒.....	545
31.5	初始化/应用信息.....	545

## 第 32 章 MSCAN

32.1	简介.....	549
32.1.1	术语表.....	549
32.1.2	结构框图.....	549

小节编号	标题	页
32.1.3	特性.....	550
32.1.4	工作模式.....	551
32.1.4.1	系统正常工作模式.....	551
32.1.4.2	系统特殊工作模式.....	551
32.1.4.3	仿真模式.....	551
32.1.4.4	仅监听模式.....	551
32.1.4.5	MSCAN 初始化模式.....	551
32.2	外部信号说明.....	552
32.2.1	CAN 系统.....	553
32.3	存储器映像和寄存器定义.....	553
32.3.1	程序员报文存储模型.....	553
32.3.2	MSCAN 控制寄存器 0 (MSCAN_CANCTL0).....	558
32.3.3	MSCAN 控制寄存器 1 (MSCAN_CANCTL1).....	560
32.3.4	MSCAN 总线定时寄存器 0 (MSCAN_CANBTR0).....	561
32.3.5	MSCAN 总线定时寄存器 1 (MSCAN_CANBTR1).....	562
32.3.6	MSCAN 接收器标志寄存器 (MSCAN_CANRFLG).....	563
32.3.7	MSCAN 接收器中断使能寄存器 (MSCAN_CANRIER).....	565
32.3.8	MSCAN 发送器标志寄存器 (MSCAN_CANTFLG).....	566
32.3.9	MSCAN 发送器中断使能寄存器 (MSCAN_CANTIER).....	567
32.3.10	MSCAN 发送器报文中止请求寄存器 (MSCAN_CANTARQ).....	568
32.3.11	MSCAN 发送器报文中止应答寄存器 (MSCAN_CANTAAK).....	568
32.3.12	MSCAN 发送缓冲区选择寄存器 (MSCAN_CANTBSEL).....	569
32.3.13	MSCAN 标识符验收控制寄存器 (MSCAN_CANIDAC).....	570
32.3.14	MSCAN 其他寄存器 (MSCAN_CANMISC).....	571
32.3.15	MSCAN 接收错误计数器 (MSCAN_CANRXERR).....	572
32.3.16	MSCAN 发送错误计数器 (MSCAN_CANTXERR).....	572
32.3.17	第一群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDARn).....	573
32.3.18	第一群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMRn).....	574
32.3.19	第二群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDARn).....	574

小节编号	标题	页
32.3.20	第二群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMRn).....	575
32.3.21	接收扩展标识符寄存器 0 (MSCAN_REIDR0).....	576
32.3.22	接收标准标识符寄存器 0 (MSCAN_RSIDR0).....	576
32.3.23	接收扩展标识符寄存器 1 (MSCAN_REIDR1).....	577
32.3.24	接收标准标识符寄存器 1 (MSCAN_RSIDR1).....	578
32.3.25	接收扩展标识符寄存器 2 (MSCAN_REIDR2).....	579
32.3.26	接收扩展标识符寄存器 3 (MSCAN_REIDR3).....	579
32.3.27	接收扩展数据段寄存器 N (MSCAN_REDSTRn).....	580
32.3.28	接收数据长度寄存器 (MSCAN_RDLR).....	580
32.3.29	接收时间标志寄存器高位 (MSCAN_RTSRH).....	581
32.3.30	接收时间标志寄存器低位 (MSCAN_RTSRL).....	581
32.3.31	发送扩展标识符寄存器 0 (MSCAN_TEIDR0).....	582
32.3.32	发送标准标识符寄存器 0 (MSCAN_TSIDR0).....	583
32.3.33	发送扩展标识符寄存器 1 (MSCAN_TEIDR1).....	583
32.3.34	发送标准标识符寄存器 1 (MSCAN_TSIDR1).....	584
32.3.35	发送扩展标识符寄存器 2 (MSCAN_TEIDR2).....	585
32.3.36	发送扩展标识符寄存器 3 (MSCAN_TEIDR3).....	585
32.3.37	发送扩展数据段寄存器 N (MSCAN_TEDSTRn).....	586
32.3.38	发送数据长度寄存器 (MSCAN_TDLR).....	586
32.3.39	发送缓冲区优先级寄存器 (MSCAN_TBPR).....	587
32.3.40	发送时间标志寄存器高位 (MSCAN_TTSRH).....	588
32.3.41	发送时间标志寄存器低位 (MSCAN_TTSRL).....	588
32.4	功能说明.....	590
32.4.1	报文存储.....	590
32.4.2	报文发送后台.....	591
32.4.3	发送结构.....	591
32.4.4	接收结构.....	592
32.4.5	标识符验收滤波器.....	593
32.4.5.1	协议违规保护.....	595

小节编号	标题	页
	32.4.5.2 时钟系统.....	596
32.4.6 低功耗选项.....		598
	32.4.6.1 Run 模式下的操作.....	598
	32.4.6.2 Wait 模式下的操作.....	599
	32.4.6.3 Stop 模式下的操作.....	599
	32.4.6.4 MSCAN 正常模式.....	599
	32.4.6.5 MSCAN 睡眠模式.....	599
	32.4.6.6 MSCAN 降耗模式.....	601
	32.4.6.7 禁用模式.....	601
	32.4.6.8 可编程唤醒功能.....	601
32.4.7 复位初始化.....		602
32.4.8 中断.....		602
	32.4.8.1 中断操作说明.....	602
	32.4.8.2 发送中断.....	602
	32.4.8.3 接收中断.....	602
	32.4.8.4 唤醒中断.....	602
	32.4.8.5 错误中断.....	603
	32.4.8.6 中断应答.....	603
32.4.9 初始化/应用信息.....		603
	32.4.9.1 MSCAN 初始化.....	603
	32.4.9.2 总线关闭恢复.....	604

### 第 33 章 通用异步收发器(UART)

33.1 简介.....	605
33.1.1 特性.....	605
33.1.2 工作模式.....	605
33.1.3 结构框图.....	606
33.2 UART 信号说明.....	608
33.2.1 详细信号说明.....	608

小节编号	标题	页
33.3 寄存器定义.....		608
33.3.1 UART 波特率寄存器：高位 (UARTx_BDH).....		609
33.3.2 UART 波特率寄存器：低位 (UARTx_BDL).....		610
33.3.3 UART 控制寄存器 1 (UARTx_C1).....		611
33.3.4 UART 控制寄存器 2 (UARTx_C2).....		612
33.3.5 UART 状态寄存器 1 (UARTx_S1).....		613
33.3.6 UART 状态寄存器 2 (UARTx_S2).....		615
33.3.7 UART 控制寄存器 3 (UARTx_C3).....		617
33.3.8 UART 数据寄存器 (UARTx_D).....		618
33.4 功能说明.....		619
33.4.1 波特率生成.....		619
33.4.2 发送器功能说明.....		619
33.4.2.1 发送分隔和已排队的闲置.....		620
33.4.3 接收器功能说明.....		621
33.4.3.1 数据采样技术.....		621
33.4.3.2 接收器唤醒操作.....		622
33.4.4 中断和状态标志.....		623
33.4.5 波特率公差.....		624
33.4.5.1 慢速数据公差.....		625
33.4.5.2 快速数据公差.....		626
33.4.6 其他 UART 功能.....		627
33.4.6.1 8 位和 9 位数据模式.....		627
33.4.6.2 Stop 模式操作.....		627
33.4.6.3 循环模式.....		627
33.4.6.4 单线操作.....		628

## 第 34 章 通用输入/输出 (GPIO)

34.1 简介.....	629
34.1.1 特性.....	629

小节编号	标题	页
34.1.2	工作模式.....	629
34.1.3	GPIO 信号说明.....	630
	34.1.3.1    详细信号说明.....	630
34.2	存储器映像和寄存器定义.....	631
34.2.1	GPIO/FGPIO 寄存器位分配.....	631
34.2.2	端口数据输出寄存器 (GPIO <sub>x</sub> _PDOR).....	633
34.2.3	端口置位输出寄存器 (GPIO <sub>x</sub> _PSOR).....	633
34.2.4	端口清零输出寄存器 (GPIO <sub>x</sub> _PCOR).....	634
34.2.5	端口跳变输出寄存器 (GPIO <sub>x</sub> _PTOR).....	634
34.2.6	端口数据输入寄存器 (GPIO <sub>x</sub> _PDIR).....	635
34.2.7	端口数据方向寄存器 (GPIO <sub>x</sub> _PDDR).....	635
34.2.8	端口输入禁用寄存器 (GPIO <sub>x</sub> _PIDR).....	636
34.3	FGPIO 存储器映像和寄存器定义.....	636
34.3.1	GPIO/FGPIO 寄存器位分配.....	636
34.3.2	端口数据输出寄存器 (FGPIO <sub>x</sub> _PDOR).....	638
34.3.3	端口置位输出寄存器 (FGPIO <sub>x</sub> _PSOR).....	638
34.3.4	端口清零输出寄存器 (FGPIO <sub>x</sub> _PCOR).....	639
34.3.5	端口跳变输出寄存器 (FGPIO <sub>x</sub> _PTOR).....	639
34.3.6	端口数据输入寄存器 (FGPIO <sub>x</sub> _PDIR).....	640
34.3.7	端口数据方向寄存器 (FGPIO <sub>x</sub> _PDDR).....	640
34.3.8	端口输入禁用寄存器 (FGPIO <sub>x</sub> _PIDR).....	640
34.4	功能说明.....	641
34.4.1	通用输入.....	641
34.4.2	通用输出.....	641
34.4.3	IOPORT.....	641

## 第 35 章 键盘中断(KBI)

35.1	简介.....	643
35.1.1	特性.....	643

小节编号	标题	页
35.1.2	工作模式.....	643
35.1.2.1	Wait 模式下的 KBI.....	643
35.1.2.2	Stop 模式下的 KBI.....	644
35.1.3	结构框图.....	644
35.2	外部信号说明.....	644
35.3	寄存器定义.....	645
35.4	存储器映像和寄存器.....	645
35.4.1	KBI 引脚使能寄存器 (KBI <sub>x</sub> _PE).....	646
35.4.2	KBI 边沿选择寄存器 (KBI <sub>x</sub> _ES).....	646
35.4.3	KBI 状态和控制寄存器 (KBI <sub>x</sub> _SC).....	647
35.4.4	KBI 源引脚寄存器 (KBI <sub>x</sub> _SP).....	648
35.5	功能说明.....	648
35.5.1	边沿触发.....	649
35.5.2	边沿和电平触发.....	649
35.5.3	KBI 上拉电阻.....	649
35.5.4	KBI 初始化.....	649

## 第 36 章 第 2 版的版本记录

# 第 1 章 关于本文档

## 1.1 概述

### 1.1.1 目的

本文档介绍 Freescale KEA128 微控制器的特性、架构和编程模型。

### 1.1.2 受众

本文档主要面向在系统中使用或打算使用 KEA128 微控制器的系统架构和软件应用程序开发人员。

## 1.2 惯例

### 1.2.1 编码系统

通过以下后缀可识别不同的编码系统：

后缀	用来识别
b	二进制数。例如，数字 5 的二进制数写为 101b。在某些情况下，二进制数冠以前缀 0b。
d	十进制数。当有可能出现混淆时才为十进制数加后缀。通常，十进制数没有后缀。
h	十六进制数。例如，数字 60 的十六进制数写为 3Ch。在某些情况下，十六进制数冠以前缀 0x。

## 1.2.2 印刷符号

以下印刷符号广泛应用于本文档：

示例	说明
占位符, $x$	斜体项目是您所提供的信息的占位符。斜体文本还用作出版物的标题，用于强调。无格式的小写字母还用作单个字母和数字的占位符。
代码	固定宽度类型指的是文本必须严格按照所示格式输入。它可用于指令助记符、指令符、符号、子命令、参数和运算符。固定宽度类型也用于示例代码。文本和表格中的指令助记符和指令符必须全部大写，例如：BSR。
SR[SCM]	括号中的助记符代表寄存器中的命名字段。此示例指的是状态寄存器(SR)中的缩放模式(SCM)字段。
REVNO[6:4]和 XAD[7:0]	数字放在方括号中，数字之间用冒号隔开可代表以下两种意思： <ul style="list-style-type: none"> <li>寄存器命名字段的子集 例如，REVNO[6:4]指的是 6-4 位，其在 REVNO 寄存器中占据 6-0 位的 COREREV 字段的一部分。</li> <li>独立总线信号的连续范围 例如，XAD[7:0]指的是 XAD 总线的信号 7-0。</li> </ul>

## 1.2.3 特殊术语

以下术语具有特殊含义：

术语	含义
电平有效	指的是以下信号状态： <ul style="list-style-type: none"> <li>高电平有效信号为高电平(1)时，其电平有效。</li> <li>低电平有效信号为低电平(0)时，其电平有效。</li> </ul>
电平无效	指的是以下信号状态： <ul style="list-style-type: none"> <li>高电平有效信号为低电平(0)时，其电平无效。</li> <li>高电平有效信号为高电平(1)时，其电平无效。</li> </ul> 在某些情况下，电平无效的信号还被描述为电平 <i>Negated</i> 。
保留	指某存储空间、寄存器、或字段。它们或者为以后使用保留，或者是当写入值后会导致模块或芯片行为不可预知。

## 第 2 章 简介

### 2.1 概述

本章概述了 Kinetis KEA128 等 ARM<sup>®</sup> Cortex<sup>®</sup>-M0+ MCU 产品系列。此外，还对本文档所涵盖器件上的可用模块进行了概要说明。

Kinetis EA 系列 MCU 是针对汽车市场开发的 MCU。它的内核基于 32 位 ARM Cortex-M0+，具有高度可扩展性。此款产品系列对于成本敏感的应用进行了优化，可提供低引脚的选择，且具有极低功耗的工作状态。2.7~5.5v 供电及优异的 EMC/ESD 健壮性，使 Kinetis EA 系列 MCU 非常适合从车身电子到车身安全或通用传感器节点等应用。在车身电子应用里面，Kinetis EA 系列 MCU 针对于如下应用，例如入门级的车身控制、网关模块、车窗/天窗、车锁、座椅/后视镜控制，是个很好的选择。对于有特殊安全性要求的相关应用，可咨询飞思卡尔当地的人员，以获得相关建议。Kinetis EA 系列 MCU 产品共用类似的外设且引脚兼容，丰富的存储器尺寸大小的划分，为开发者从冗余过多的设计过度到 kinetis 紧凑的设计提供了方便。Kinetis EA 系列 MCU 的兼容性和扩展性，可使开发者标准化地开发其产品，最大化地重用硬件和软件，从而缩短开发周期。Kinetis EA 系列 MCU 支持很多第三方或飞思卡尔的开发工具，例如 Code Warrior、Keil、iAR、处理器专家(Processor Expert)和 MQX。开发者可在上述的开发环境中，快速且方便的进行设计。

### 2.2 模块功能类别

该器件上的模块按照功能进行归类。下列章节更详细地说明了每一类的模块。

表 2-1. 模块功能类别

模块类别	说明
ARM Cortex-M0+内核	<ul style="list-style-type: none"> <li>ARM Cortex-M 级 32 位 MCU 内核，1.77 CoreMark<sup>®</sup>/MHz，单周期访问存储器，48 MHz CPU 频率</li> </ul>
系统	<ul style="list-style-type: none"> <li>系统集成模块(SIM)</li> <li>电源管理和模式控制器(PMC)</li> <li>其他控制模块(MCM)</li> </ul>

下一页继续介绍此表...

表 2-1. 模块功能类别 (继续)

模块类别	说明
	<ul style="list-style-type: none"> <li>位操作引擎(BME)</li> <li>外设桥(AIPS)</li> <li>WDOG</li> </ul>
存储器	<ul style="list-style-type: none"> <li>内部存储器包括:           <ul style="list-style-type: none"> <li>高达 128 KB 的 Flash 存储器</li> <li>高达 16 KB 的 SRAM</li> </ul> </li> </ul>
时钟	<ul style="list-style-type: none"> <li>外部晶振或谐振器           <ul style="list-style-type: none"> <li>低范围: 31.25–39.0625 kHz</li> <li>高范围: 4–24 MHz</li> </ul> </li> <li>外部方波输入时钟</li> <li>内部时钟基准           <ul style="list-style-type: none"> <li>31.25 至 39.0625 kHz 振荡器</li> <li>1 kHz LPO 振荡器</li> </ul> </li> <li>锁频环(FLL)范围: 40–50 MHz</li> </ul>
安全性	<ul style="list-style-type: none"> <li>采用独立时钟源的 WDOG</li> <li>用于误差检测的循环冗余校验(CRC)模块</li> </ul>
模拟	<ul style="list-style-type: none"> <li>一个 12 位模数转换器(ADC)，具有多达 16 个通道</li> <li>两个模拟比较器(ACMP)，带内部 6 位数模转换器(DAC)</li> </ul>
定时器	<ul style="list-style-type: none"> <li>一个全功能型 6 通道 FlexTimer (FTM)</li> <li>两个双通道 FTM，具有基本 TPM 功能</li> <li>双通道周期性中断定时器(PIT)</li> <li>实时时钟(RTC)</li> <li>一个脉宽定时器(PWT)</li> <li>系统节拍定时器(SysTick)</li> </ul>
通信	<ul style="list-style-type: none"> <li>两个 8 位串行外设接口(SPI)</li> <li>两个 I<sup>2</sup>C 模块</li> <li>三个通用异步收发器(UART)模块</li> <li>一个 MSCAN</li> </ul>
人机界面(HMI)	<ul style="list-style-type: none"> <li>通用输入/输出(GPIO)控制器</li> <li>两个键盘中断(KBI)</li> <li>外部中断(IRQ)</li> </ul>

## 2.2.1 ARM Cortex-M0+内核模块

此器件提供下列内核模块。

表 2-2. 内核模块

模块	说明
ARM Cortex-M0+	<p>ARM Cortex-M0+是 Cortex M 系列处理器的最新产品，面向重点关注对成本极为敏感、具有确定性且由中断驱动的应用。Cortex M0+处理器基于 ARMv6 架构和 Thumb®-2 ISA，其指令集完全兼容其前代产品（即 Cortex-M0 内核），同时向上兼容 Cortex-M3 和 M4 内核。</p> <p>ARM Cortex-M0+的改进包括一个 ARMv6 Thumb-2 DSP，该 DSP 采用 ARMv6-A/R 配置架构，提供 32 位指令，具有 SIMD（单指令多数据）DSP 类型的乘法/加法和饱和算法，支持单周期 32x32 乘法器。</p>

下一页继续介绍此表...

表 2-2. 内核模块 (继续)

模块	说明
可嵌套向量的中断控制器(NVIC)	ARMv6-M 异常模型和可嵌套向量的中断控制器(NVIC)集成可重定位的向量表，支持许多外部中断、一个非屏蔽中断(NMI)和优先级。 NVIC 用等效系统和简化编程性能代替影子寄存器。NVIC 包含中断响应的函数的地址。该地址通过指令端口提取，允许并行寄存器堆栈与查找。前 16 个入口分配至 ARM 内部，其余入口则映射至 MCU 定义的中断。
异步唤醒中断控制器(AWIC)	异步唤醒中断控制器(AWIC)的主要功能是检测 Stop 模式下的异步唤醒事件，并向时钟控制逻辑发送信号以恢复系统时钟。时钟重启后，NVIC 观察处于挂起状态的中断，并执行正常中断或事件处理。
单周期 I/O 端口(IOPORT)	为了实现对外设的高速单周期访问，Cortex-M0+处理器集成了专用的单周期 I/O 端口。在该器件上，快速 GPIO (FGPIO) 实施在 IOPORT 接口上。
调试接口	该器件的大部分调试都基于 ARM CoreSight™ 架构。支持的调试接口为： <ul style="list-style-type: none"><li>• 串行线调试(SWD)</li></ul>

## 2.2.2 系统模块

此器件提供下列系统模块。

表 2-3. 系统模块

模块	说明
系统集成模块(SIM)	SIM 包括集成逻辑和多个模块配置设置。
电源管理控制器(PMC)	PMC 为用户提供多种电源选项。有多种模式可支持，用户因此可针对所需的功能水平优化功耗。包括上电复位(POR)和集成的低压检测(LVD)功能，具有复位(掉电)能力和可选 LVD 跳变点。
其他控制模块(MCM)	MCM 包括集成逻辑和详细信息。
外设桥(AIPS-Lite)	外设桥将 ARM AHB 接口转换为一个可访问器件上大多数外设的接口。
WDOG	WDOG 监控内部系统操作并在发生故障时强制进行复位。它工作时可以采用独立的 1 kHz 低功耗振荡器，具有可检测程序流或系统频率中偏差的可编程刷新窗口。
位操作引擎(BME)	BME 针对基于 Cortex-M0+ 的微控制器中对外设地址空间的读取-修改-写入存储器原子操作提供硬件支持。

## 2.2.3 存储器和存储器接口

该器件提供下列存储器和存储器接口。

表 2-4. 存储器和存储器接口

模块	说明
Flash 存储器(FTMRE)	Flash 存储器 — 高达 64/128 的非易失性 Flash 存储器，可执行程序代码。

下一页继续介绍此表...

表 2-4. 存储器和存储器接口 (继续)

模块	说明
Flash 存储器控制器(FMC)	FMC 是存储器加速单元，提供 Cortex M0+内核和 32 位 P-Flash 之间的接口。FMC 包含一个 32 位推测缓冲区和一个 32 字节高速缓存，可加快指令提取和 Flash 访问速度。
SRAM	高达 16 KB 的内部系统 RAM，支持通过 BME 模块或混叠位段域进行位操作。

## 2.2.4 时钟

此器件提供下列时钟模块。

表 2-5. 时钟模块

模块	说明
内部时钟源(ICS)	ICS 模块，包含内部基准时钟(ICSIRCLK)以及锁频环(FLL)。
系统振荡器(OSC)	系统振荡器与外部晶体或谐振器一同生成 MCU 的基准时钟。
低功耗振荡器(LPO)	PMC 模块集成一个在所有模式下都可用作独立低频时钟源的 1 kHz 低功耗振荡器。

## 2.2.5 安全性和完整性模块

以下安全性和完整性模块可用于此设备：

表 2-6. 安全性和完整性模块

模块	说明
循环冗余校验(CRC)	CRC 生成可用于错误检测的 16/32 位 CRC 码。
WDOG	WDOG 监控内部系统操作并在发生故障时强制进行复位。它工作时可以采用独立的 1 kHz 低功耗振荡器，具有可检测程序流或系统频率中偏差的可编程刷新窗口。

## 2.2.6 模拟模块

此器件提供下列模拟模块：

表 2-7. 模拟模块

模块	说明
模数转换器(ADC)	12 位逐次逼近型 ADC 模块，具有多达 16 个通道。
模拟比较器(ACMP)	两个比较器，支持全电源电压范围内的模拟输入电压以及 CPU 中断。ACMP0/1 还可进一步触发 ADC 采样和 FTM 更新。
6 位数模转换器(DAC)	64-tap 电阻阶梯网络，能够为比较器提供可选基准电压。

## 2.2.7 定时器模块

此器件提供下列定时器模块：

表 2-8. 定时器模块

模块	说明
FlexTimer 模块(FTM)	<ul style="list-style-type: none"> <li>可选 FTM 源时钟，支持高达 48 MHz 的单独定时器时钟，可编程预分频器</li> <li>16 位计数器，支持自由运行或初始/最终值，可向上或上下计数</li> <li>输入捕捉、输出比较以及边沿对齐与中心对齐 PWM 模式</li> <li>FTM 通道可采用具有相同输出或互补输出的成对工作方式，或者以具有独立输出的独立通道方式工作</li> <li>死区时间插入可用于每一对互补通道</li> <li>PWM 输出的软件控制</li> <li>对于全局故障控制最多有 2 个故障输入</li> <li>可配置通道极性</li> <li>可编程的中断可用于输入捕捉、基准比较、溢出的计数器或检测到的故障条件</li> </ul>
周期性中断定时器(PIT)	<ul style="list-style-type: none"> <li>一个通用中断定时器</li> <li>适用于触发 ADC 转换</li> <li>32 位计数器分辨率</li> <li>根据总线时钟频率计时</li> </ul>
实时计数器(RTC)	<ul style="list-style-type: none"> <li>16 位上行计数器           <ul style="list-style-type: none"> <li>16 位模数匹配限制</li> <li>软件可控周期性匹配中断</li> </ul> </li> <li>可通过软件选择预分频器输入的时钟源；集成可编程 16 位预分频器           <ul style="list-style-type: none"> <li>总线时钟</li> <li>IRC 时钟(31.25~39.0625 kHz)</li> <li>LPO (约 1 kHz)</li> <li>系统振荡器输出时钟</li> </ul> </li> </ul>
脉宽定时器(PWT)	<ul style="list-style-type: none"> <li>分辨率为 16 位的脉冲宽度的自动测量</li> <li>正脉冲和负脉冲的单独脉宽测量</li> <li>可编程的开始测量的触发沿</li> <li>逐次交替型边沿、上升沿或下降沿之间的可编程测量时间</li> <li>时钟输入可通过编程预分频器作为 16 位计数器时基</li> <li>两个可选时钟源，支持高达 48 MHz 的单独定时器时钟</li> <li>四个可选脉冲输入</li> <li>可设定在脉冲宽度值更新及计数器溢出时生成中断</li> </ul>

## 2.2.8 通信接口

该器件提供下列通信接口：

表 2-9. 通信模块

模块	说明
串行外设接口(SPI)	两个 SPI 同步串行总线，用于与外部器件通信。

下一页继续介绍此表...

表 2-9. 通信模块 (继续)

模块	说明
I2C	两个 I2C 模块，用于内部器件通信。还支持系统管理总线(SMBus)规范（第 2 版）。I2C0 支持四线式接口特性。
通用异步收发器(UART)	三个异步串行总线通信接口 (UART) 模块，支持可选 13 位断点、全双工不归零(NRZ) 和 LIN 扩展支持。
MSCAN	一个 MSCAN。

## 2.2.9 人机接口

该器件提供下列人机接口(HMI):

表 2-10. HMI 模块

模块	说明
通用输入(GPIO)	多达 71 个通用输入或输出(GPIO)引脚。
键盘中断(KBI)	两个支持引脚中断的 KBI 模块。
外部中断(IRQ)	IRQ 模块提供可屏蔽中断输入。

## 2.2.10 可订购部件编号

下表汇总了本文档所涉及器件的部件编号。

表 2-11. 可订购部件编号汇总

Freescale 部件编号	CPU 频率	引脚数	封装	Flash 存储器总量	RAM	温度范围
S9KEAZ64AMLH(R)	48 MHz	64	LQFP	64 KB	8 KB	-40 至 125 °C
S9KEAZ128AMLH(R)	48 MHz	64	LQFP	128 KB	16 KB	-40 至 125 °C
S9KEAZ64AMLK(R)	48 MHz	80	LQFP	64 KB	8 KB	-40 至 125 °C
S9KEAZ128AMLK(R)	48 MHz	80	LQFP	128 KB	16 KB	-40 至 125 °C
S9KEAZ64AVLK(R)	48 MHz	80	LQFP	64 KB	8 KB	-40 至 125 °C
S9KEAZ128AVLK(R)	48 MHz	80	LQFP	128 KB	16 KB	-40 至 125 °C
S9KEAZ64ACLK(R)	48 MHz	80	LQFP	64 KB	8 KB	-40 至 125 °C
S9KEAZ128ACLK(R)	48 MHz	80	LQFP	128 KB	16 KB	-40 至 125 °C
S9KEAZ64AVLH(R)	48 MHz	64	LQFP	64 KB	8 KB	-40 至 125 °C
S9KEAZ128AVLH(R)	48 MHz	64	LQFP	128 KB	16 KB	-40 至 125 °C
S9KEAZ64ACLH(R)	48 MHz	64	LQFP	64 KB	8 KB	-40 至 125 °C
S9KEAZ128ACLH(R)	48 MHz	64	LQFP	128 KB	16 KB	-40 至 125 °C

# 第3章 芯片配置

## 3.1 简介

本章提供微控制器各模块的详细信息。包括：

- 模块框图展示其内部的连接
- 单独的模块章节中无需讨论具体的模块间交互
- 更多信息的链接

## 3.2 模块间互连

### 3.2.1 互连概述

下表包含了该器件的模块间互连。

表 3-1. 模块间互连

外设	信号	—	到外设	用例	控制	备注
FTM2	INITTRG、MatchTRG	到	ADC（触发器）	ADC 触发	SIM_SOPT0[ADHWT]	—
PIT CHx	TIF0、TIF1	到	ADC（触发器）	ADC 触发	SIM_SOPT0[ADHWT]	—
RTC	RTC 溢出	到	ADC（触发器）	ADC 触发	SIM_SOPT0[ADHWT]	—
FTM0	INITTRG	到	ADC（触发器）	ADC 触发	SIM_SOPT0[ADHWT]	—
ACMP0	CMP0_OUT	到	ADC（触发器）	ADC 触发	SIM_SOPT0[ADHWT]	—
ACMP1	CMP1_OUT	到	ADC（触发器）	ADC 触发	SIM_SOPT0[ADHWT]	—
UART0	TX0	到	通过 FTM0 CH0 调制	UART 调制	SIM_SOPT0[TXDME]	—
UART0	RX0	到	FTM0 CH1 输入捕捉	UART 调制	SIM_SOPT0[RXDCE]	—
ACMP0	CMP0_OUT	到	UART0 RxD	UART RxD 滤波器	SIM_SOPT0[RXDFE]	—
ACMP1	CMP1_OUT	到	UART0 RxD	UART RxD 滤波器	SIM_SOPT0[RXDFE]	—

下一页继续介绍此表...

表 3-1. 模块间互连 (继续)

外设	信号	—	到外设	用例	控制	备注
ACMP0	CMP0_OUT	到	FTM1 CH0 输入捕捉	ACMP0 输出捕捉	SIM_SOPT0[ACIC]	—
RTC	RTC 溢出	到	FTM1 CH1 输入捕捉	RTC 溢出捕捉	SIM_SOPT0[RTCC]	—
ACMP0	CMP0_OUT	到	FTM2 Trigger0	FTM2 触发输入	SIM_SOPT0[ACTRG] & FTM2_SYNC[TRIG0]	—
ACMP1	CMP1_OUT	到	FTM2 Trigger0	FTM2 触发输入	SIM_SOPT0[ACTRG] & FTM2_SYNC[TRIG0]	—
FTM0	FTM0 CH0 输出	到	FTM2 Trigger1	FTM2 触发输入	FTM2_SYNC[TRIG1]	—
SIM	FTMSYNC	到	FTM2 Trigger2	FTM2 触发输入	FTM2_SYNC[TRIG2]	—
ACMP0	CMP0_OUT	到	FTM2 Fault0	FTM2 故障输入	FTM2_FLTCTRL[FAULT0EN]	—
EXTFT_IN	EXTFT_IN	到	FTM2 Fault1	FTM2 故障输入	FTM2_FLTCTRL[FAULT1EN]	PTA6
EXTFT_IN	EXTFT_IN	到	FTM2 Fault2	FTM2 故障输入	FTM2_FLTCTRL[FAULT2EN]	PTA7
ACMP1	CMP1_OUT	到	FTM2 Fault3	FTM2 故障输入	FTM2_FLTCTRL[FAULT3EN]	—
EXT_PWT_I_N0	PWT_IN0	到	PWT	PWT_IN0	SIM_PINSEL1[PWTIN0PS] & PWT_R1[PINSEL]	PTD5 或 PTE2
EXT_PWT_I_N1	PWT_IN1	到	PWT	PWT_IN1	SIM_PINSEL1[PWTIN1PS] & PWT_R1[PINSEL]	PTB0 或 PTH7
ACMP0	ACMP0_OUT	到	PWT	PWT_IN2	SIM_SOPT1[ACPWTS] & PWT_R1[PINSEL]	—
ACMP1	ACMP1_OUT	到	PWT	PWT_IN2	SIM_SOPT1[ACPWTS] & PWT_R1[PINSEL]	—
UART0	RX	到	PWT	PWT_IN3	SIM_SOPT1[UARTPWTS] & PWT_R1[PINSEL]	—
UART1	RX	到	PWT	PWT_IN3	SIM_SOPT1[UARTPWTS] & PWT_R1[PINSEL]	—
UART2	RX	到	PWT	PWT_IN3	SIM_SOPT1[UARTPWTS] & PWT_R1[PINSEL]	—

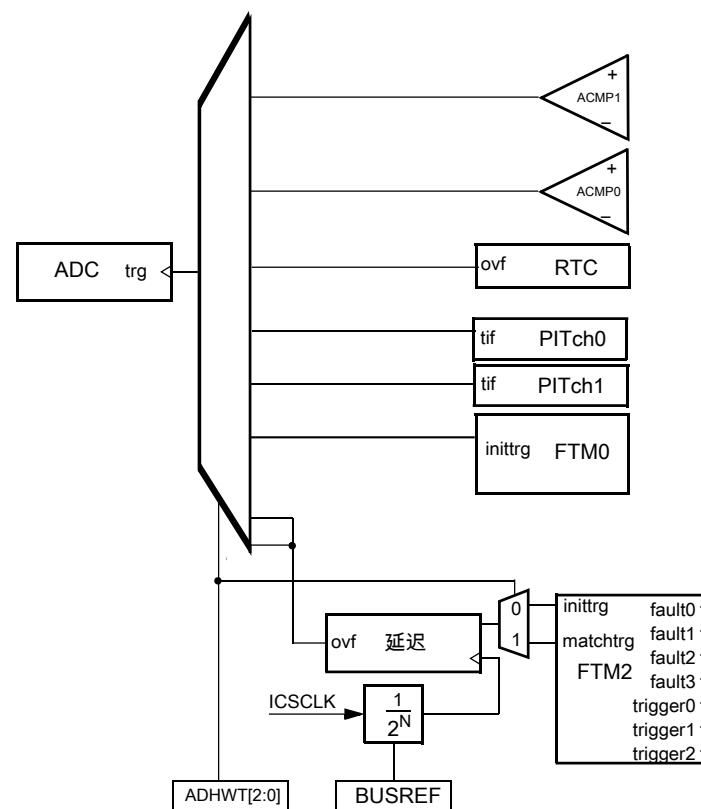


图 3-1. ADC 硬件触发连接图

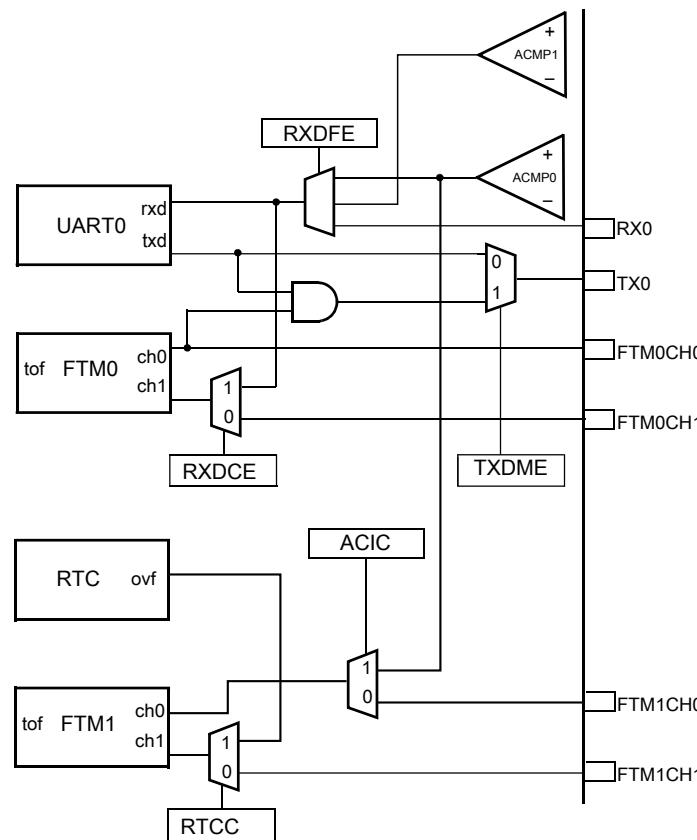


图 3-2. FTMx/ACMPx/UARTx 连接图

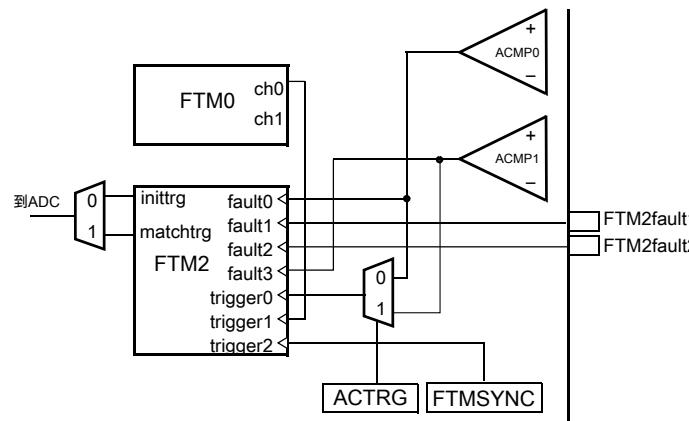


图 3-3. FTM2 互连示意图

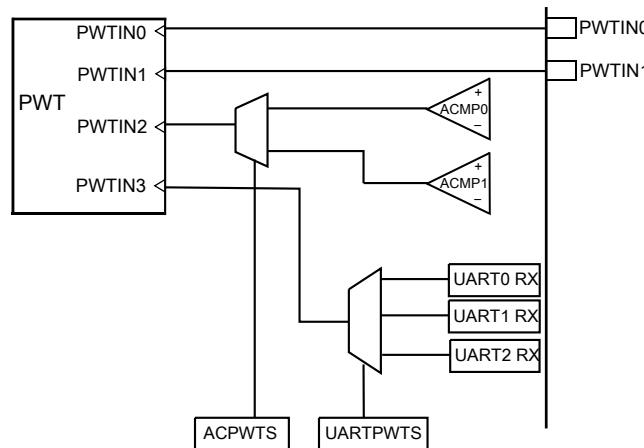


图 3-4. PWT 互连示意图

### 3.2.2 模拟基准选项

多个模拟数据块拥有可选择的基准电压，如表 3-2 所示。这些选项允许模拟外设共享或拥有单独的模拟基准。需要慎重选择模拟基准以避免串扰噪声。

表 3-2. 模拟基准选项

模块	基准选项	备注/基准选择
12 位 ADC	<ul style="list-style-type: none"> <li>• VREFH</li> <li>• VREFL</li> </ul>	为了提高 ADC 性能，80LQFP 上提供了 VREFH 引脚，它与 64QFP/LQFP、44LQFP 上的 VDDA 在内部相连，各封装上均提供了 VREFL 引脚。
ACMP0 ACMP1	<ul style="list-style-type: none"> <li>• VDDA</li> <li>• 带隙基准</li> </ul>	由 ACMP0_C1[DACREF] 或 ACMP1_C1[DACREF] 选择

### 3.2.3 ACMP 输出捕捉

置位时，SIM\_SOPT0[ACIC]位使能 ACMP0 的输出连接到 FTM1\_CH0，FTM1\_CH0 引脚被释放，可以被用作其他共享功能。

SIM\_SOPT0[RXDFE]置位为 01b 时，可选择 ACMP0 输出，使其连接到 UART0 的接收器通道。SIM\_SOPT0[RXDFE]置位为 10b 时，可选择 ACMP1 输出，使其连接到 UART1 的接收器通道。

ACMP0 和 ACMP1 输出还选择性地连接到 PWT input2(PWT\_IN2)，或可用作 FTM2 触发/故障输入和 ADC 硬件触发。

### 3.2.4 UART0\_TX 调制

UART0\_TX 可通过 FTM0 通道 0 输出调制。SIM\_SOPT0[TXDME]置位时，UART0\_TX 由 FTM0 通道 0 输出通过与门进行选通，然后映射到 UART0\_TX 管脚。将该字段清零后，UART0\_TX 会直接映射到管脚上。要使能 IR 调制功能，FTM0\_CH0 和 UART 都必须处于有效状态。FTM0\_CNT 和 FTM0\_MOD 寄存器指定 PWM 的周期，而 FTM0\_C0V 寄存器指定 PWM 的占空比。那么，当 SIM\_SOPT0[TXDME]使能时，每个通过 UART0\_TX 从 UART0 发送的数据都通过 FTM0 通道 0 输出调制，且无论 FTM0 引脚重新分配的配置如何，FTM\_CH0 引脚让与其他共享功能。

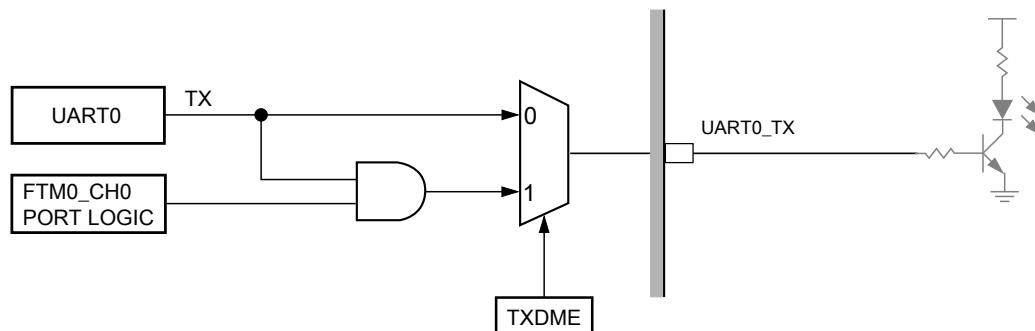


图 3-5. IR 解调制图

### 3.2.5 UART0/1/2\_RX 捕捉

UART0\_RX 引脚可被选择性地直接连接到 UART0 模块或同时连接到 FTM0 通道 1。SIM\_SOPT0[RXDCE]置位时，UART0\_RX 引脚连接 UART0 模块和 FTM0 通道 1，而无论 FTM0 引脚重新分配的配置如何，FTM0\_CH1 引脚让与其他共享功能。将该字段清零后，UART0\_RX 引脚仅连接 UART0。

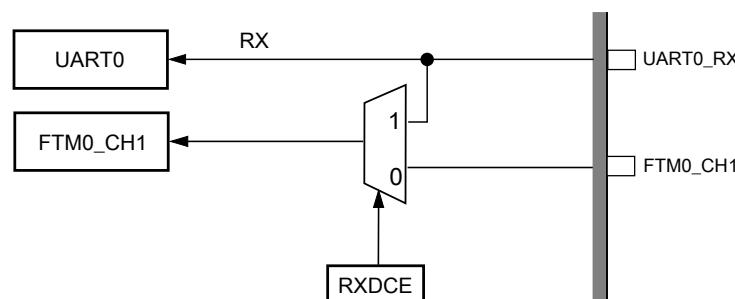


图 3-6. UART0\_RX 捕捉功能图

UART0、UART 和 UART2 的 RX 信号可由 PWT 捕捉。通过配置 SIM\_SOPT1[UARTPWTS]，可将 UART0\_RX、UART1\_RX 或 UART2\_RX 连接至 PWT 输入 3。

### 3.2.6 UART0\_RX 滤波器

SIM\_SOPT0[RXDFE]清零后，UART0\_RX 引脚直接连接至 UART0 模块。该字段正确置位后，可将 ACMP0 输出连接至 UART0 的接收通道。要使能 UART0\_RX 滤波器功能，UART0 和 ACMP0 都必须处于有效状态。如果该功能处于有效状态，则无论 UART0 引脚重新分配的配置如何，UART0 外部 UART0\_RX 引脚让与其他共享功能。当 UART0\_RX 捕捉功能处于有效状态时，ACMP0 输出也将注入 FTM0 通道 1。

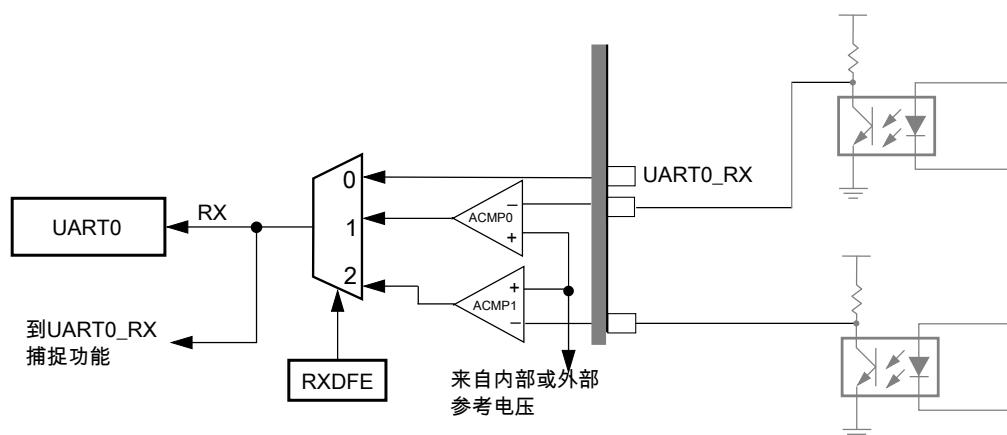


图 3-7. IR 解调制图

### 3.2.7 RTC 捕捉

RTC 溢出可通过设置 SIM\_SOPT0[RTCC]位由 FTM1 通道 1 捕捉。该位置位后，RTC 溢出连接到 FTM1 通道 1 以便进行捕捉，而 FTM1\_CH1 引脚则释放给其他共享功能。

### 3.2.8 FTM2 软件同步

FTM2 包含三个同步输入触发器，其中一个触发器通过将 1 写入 SIM\_SOPT0[FTMSYNC]来软件触发。将 0 写入此字段不起任何作用。该字段始终读到的是 0。

### 3.2.9 ADC 硬件触发

ADC 模块可以通过硬件触发发起转换。下表显示的是通过设置 SIM\_SOPT0[ADHWT]而可用的 ADC 硬件触发源。

表 3-3. ADC 硬件触发设置

ADHWT	ADC 硬件触发
000	RTC 溢出
001	FTM0 初始化触发器
010	具有 8 位可编程延迟的 FTM2 初始化触发器
011	具有 8 位可编程延迟的 FTM2 匹配触发器
100	PIT ch0 溢出
101	PIT ch1 溢出
110	ACMP0 输出
111	ACMP1 输出

当 ADC 硬件触发选择 FTM2 触发器的输出时，将会启用 8 位延迟块。该逻辑可以对 FTM2 的触发进行延迟，延迟值由一个 8 位计数器 SIM\_SOPT0[DELAY]指定。该模块的基准时钟是总线时钟，其可选择预分频器由 SIM\_SOPT0[BUSREF]指定。

## 3.3 内核模块

### 3.3.1 ARM Cortex-M0+内核配置

本节总结本模块在芯片级是如何配置的。本模块的完整文档由 ARM 提供，请访问：[arm.com](http://arm.com)。

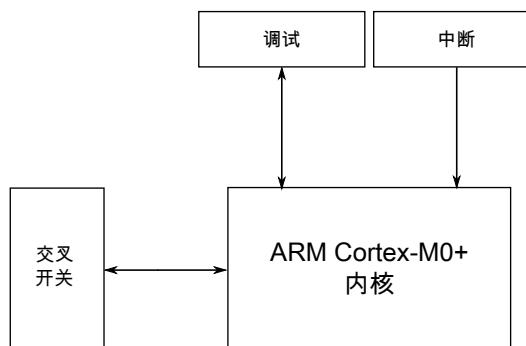


图 3-8. 内核配置

表 3-4. 相关信息的参考链接

主题	相关模块	参考
完整说明	ARM Cortex-M0+内核, r0p0	<a href="#">ARM Cortex-M0+技术参考手册, r0p0</a>
系统存储器映像		<a href="#">系统存储器映像</a>
时钟		<a href="#">时钟分布</a>
电源管理		<a href="#">电源管理</a>
系统/指令/数据总线模块	交叉开关	<a href="#">交叉开关</a>
调试	串行线调试(SWD)	<a href="#">调试</a>
中断	可嵌套向量的中断控制器(NVIC)	<a href="#">NVIC</a>
	其他控制模块(MCM)	<a href="#">MCM</a>

### 3.3.1.1 ARM Cortex M0+内核

ARM Cortex M0+参数设置如下：

表 3-5. ARM Cortex-M0+参数设置

参数	Verilog 名称	值	说明
架构时钟门控	ACG	1 = 存在	实现架构时钟门控
DAP 从端口支持	AHBSLV	1	支持任何 AHB 调试访问端口（例如 CM4 DAP）
DAP ROM 表基地址	BASEADDR	0xF000_2003	DAP ROM 表基地址
字节顺序	BE	0	数据传输的小端字节控制
断点	BKPT	2	实现了 2 个断点
调试支持	DBG	1 = 存在	—
停止事件支持	HALTEV	1 = 存在	—
I/O 端口	IOP	1 = 存在	实现对特殊地址空间的单周期加载/存储访问
IRQ 屏蔽使能	IRQDIS	0x0	—
调试端口协议	JTAGnSW	0 = SWD	SWD 协议, 非 JTAG

下一页继续介绍此表...

表 3-5. ARM Cortex-M0+参数设置 (继续)

参数	Verilog 名称	值	说明
内核存储器保护	MPU	0 = 不存在	无 MPU
IRQ 数量	NUMIRQ	32	假设完整的 NVIC 请求向量
复位所有寄存器	RAR	0 = 标准	不要强制对所有寄存器进行异步复位
乘法器	SMUL	0 = 快速乘法器	实现单周期乘法器
多点支持	SWMD	0 = 不存在	不包括串行线支持多点
系统节拍定时器	SYST	1 = 存在	实现系统节拍定时器 (以便兼容 CM4)
DAP 目标 ID	TARGETID	0	—
用户/特权	USER	1 = 存在	实现处理器工作模式
向量表偏移寄存器	VTOR	1 = 存在	实现异常向量表的重定位
WIC 支持	WIC	1 = 存在	实现 WIC 接口
WIC 请求	WICLINES	34	唤醒 IRQ 的确切数量为 34
监测点	WPT	2	实现 2 个监测点

有关 ARM Cortex-M0+处理器内核的详情，请参见 ARM 网站：[arm.com](http://arm.com)。

### 3.3.1.2 总线、互连和接口

ARM Cortex-M0+内核有两个总线接口：

- 单个 32 位 AMBA-3 AHB-Lite 系统接口，提供到外设和所有系统存储器的连接，包括 Flash 和 RAM。
- 单个 32 位 I/O 端口总线(IOPORT)，可以对 GPIO 进行单周期加载和存储。

### 3.3.1.3 系统节拍定时器

SysTick 控制和状态寄存器中的 CLKSOURCE 字段选择内核时钟 (CLKSOURCE = 1 时) 或 16 分频内核时钟 (CLKSOURCE = 0 时)。由于定时基准是一个可变频率，因此 SysTick 校准值寄存器中的 TENMS 字段将始终为零。

### 3.3.1.4 内核特权级别

本器件的内核实现了特权和非特权级别。ARM 文档使用不同于本文档中的术语以区分特权级别。

如果遇到术语...	它也意味着该术语...
特权	超级用户或者管理员
非特权或者用户	用户

### 3.3.1.5 缓存器

该器件没有与处理器相关的缓存器内存，但是 Flash 控制器有用于 Flash 访问的内部 32 字节缓存器。

### 3.3.2 可嵌套向量的中断控制器(NVIC)配置

本节总结本模块在芯片级是如何配置的。本模块的完整文档由 ARM 提供，请访问：[arm.com](http://arm.com)。

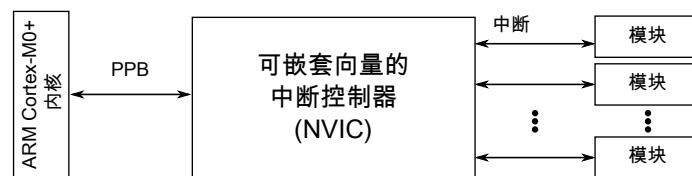


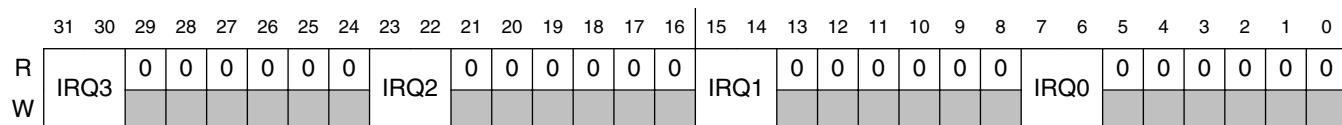
图 3-9. NVIC 配置

表 3-6. 相关信息的参考链接

主题	相关模块	参考
完整说明	可嵌套向量的中断控制器(NVIC)	<a href="#">ARM Cortex-M0+技术参考手册</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分配</a>
电源管理		<a href="#">电源管理</a>
专用外设总线(PPB)	ARM Cortex-M0+内核	<a href="#">ARM Cortex-M0+内核</a>

### 3.3.2.1 中断优先级

该器件支持 4 个中断优先级。因此，在 NVIC 中，IPR 寄存器中的各个源都包含 2 位。例如，IPR0 如下所示：



### 3.3.2.2 不可屏蔽中断

对 NVIC 的不可屏蔽中断请求由外部 **NMI** 信号控制。**NMI** 信号进行多路复用所在的引脚必须配置 **NMI** 功能，以产生不可屏蔽中断请求。

### 3.3.2.3 中断通道分配

下表中定义了中断向量分配。

- 向量编号 - 中断处于响应状态时，存储在堆栈上的值。
- IRQ 编号 - 非内核中断源编号，其为向量编号减 16。

IRQ 编号用于 ARM 的 NVIC 文档中。

表 3-8. 中断向量分配

地址	向量	IRQ <sup>1</sup>	NVIC IPR 寄存 器编号 <sup>2</sup>	来源模块	来源描述
<b>ARM 内核系统处理程序向量</b>					
0x0000_0000	0	—	—	ARM 内核	初始堆栈指针
0x0000_0004	1	—	—	ARM 内核	初始程序计数
0x0000_0008	2	—	—	ARM 内核	不可屏蔽中断(NMI)
0x0000_000C	3	—	—	ARM 内核	硬故障
0x0000_0010	4	—	—	—	—
0x0000_0014	5	—	—	—	—
0x0000_0018	6	—	—	—	—
0x0000_001C	7	—	—	—	—
0x0000_0020	8	—	—	—	—
0x0000_0024	9	—	—	—	—
0x0000_0028	10	—	—	—	—
0x0000_002C	11	—	—	ARM 内核	管理程序调用(SVCall)
0x0000_0030	12	—	—	—	—
0x0000_0034	13	—	—	—	—
0x0000_0038	14	—	—	ARM 内核	可挂起的系统服务请求(PendableSrvReq)
0x0000_003C	15	—	—	ARM 内核	系统节拍定时器(SysTick)
<b>非内核向量</b>					
0x0000_0040	16	0	0	—	—
0x0000_0044	17	1	0	—	—
0x0000_0048	18	2	0	—	—
0x0000_004C	19	3	0	—	—
0x0000_0050	20	4	1	—	—
0x0000_0054	21	5	1	FTMRE	命令完成
0x0000_0058	22	6	1	PMC	低压检测

下一页继续介绍此表...

表 3-8. 中断向量分配 (继续)

地址	向量	IRQ <sup>1</sup>	NVIC IPR 寄存器编号 <sup>2</sup>	来源模块	来源描述
0x0000_005C	23	7	1	IRQ	外部中断
0x0000_0060	24	8	2	I <sup>2</sup> C0	所有源的单个中断向量
0x0000_0064	25	9	2	I <sup>2</sup> C1	所有源的单个中断向量
0x0000_0068	26	10	2	SPI0	所有源的单个中断向量
0x0000_006C	27	11	2	SPI1	所有源的单个中断向量
0x0000_0070	28	12	3	UART0	状态和错误
0x0000_0074	29	13	3	UART1	状态和错误
0x0000_0078	30	14	3	UART2	状态和错误
0x0000_007C	31	15	3	ADC0	ADC 转换完成中断
0x0000_0080	32	16	4	ACMP0	模拟比较器 0 中断
0x0000_0084	33	17	4	FTM0	所有源的单个中断向量
0x0000_0088	34	18	4	FTM1	所有源的单个中断向量
0x0000_008C	35	19	4	FTM2	所有源的单个中断向量
0x0000_0090	36	20	5	RTC	RTC 溢出
0x0000_0094	37	21	5	ACMP1	模拟比较器 1 中断
0x0000_0098	38	22	5	PIT_CH0	PIT CH0 溢出
0x0000_009C	39	23	5	PIT_CH1	PIT CH1 溢出
0x0000_00A0	40	24	6	KBI0 (32 位)	键盘中断 0 (32 位)
0x0000_00A4	41	25	6	KBI1 (32 位)	键盘中断 1 (32 位)
0x0000_00A8	42	26	6	—	
0x0000_00AC	43	27	6	ICS	时钟失锁
0x0000_00B0	44	28	7	WDOG	WDOG 定时溢出
0x0000_00B4	45	29	7	PWT	所有源的单个中断向量
0x0000_00B8	46	30	7	MSCAN	MSCAN 接收中断
0x0000_00BC	47	31	7	MSCAN	MSCAN 发送、错误和唤醒中断

1. 表示 NVIC 的中断源编号。

2. 表示用于该 IRQ 的 NVIC 的 IPR 寄存器编号。计算该值的公式为 : IRQ 除 4

### 3.3.2.3.1 确定字段和寄存器位置, 以配置特定中断

假设您需要配置 SPI0 中断。下表是中断优先级中 SPI0 行的摘录。

表 3-9. 中断向量分配

地址	向量	IRQ <sup>1</sup>	NVIC IPR 寄存器 编号 <sup>2</sup>	来源模块	来源描述
0x0000_0068	26	10	2	SPI0	所有源的单个中断向量

1. 表示 NVIC 的中断源编号。

2. 表示用于该 IRQ 的 NVIC 的 IPR 寄存器编号。计算该值的公式为 : IRQ 除 4

- 您将用于配置中断的 NVIC 寄存器为：

- NVICIPR2

- 要确定这些特定寄存器中特定 IRQ 的字段位置：

- NVICIPR2 字段起始位置 =  $8 * (\text{IRQ mod } 4) + 6 = 22$

由于 NVICIPR 字段为 2 位宽 (4 个优先级), NVICIPR2 字段范围为位 22–23。

因此, 字段位置 NVICIPR2[23:22]用于配置 SPI0 中断。

### 3.3.3 异步唤醒中断控制器(AWIC)配置

本节总结本模块在芯片级是如何配置的。本模块的完整文档由 ARM 提供, 请访问: [arm.com](http://arm.com)。

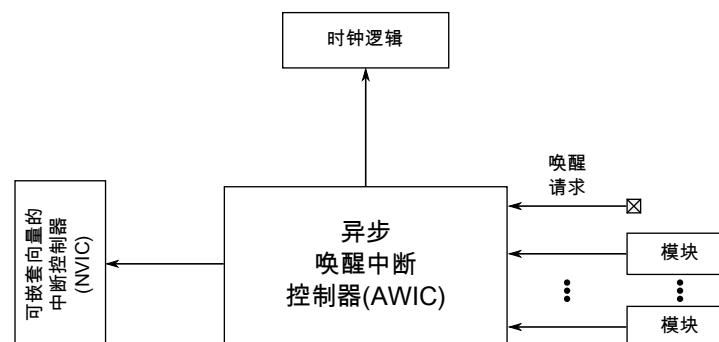


图 3-10. 异步唤醒中断控制器配置

表 3-10. 相关信息的参考链接

主题	相关模块	参考
系统存储器映像		<a href="#">系统存储器映像</a>
时钟		<a href="#">时钟分布</a>
电源管理		<a href="#">电源管理</a>
中断控制	可嵌套向量的中断控制器(NVIC)	<a href="#">NVIC</a>
唤醒请求		<a href="#">AWIC 唤醒源</a>

### 3.3.3.1 唤醒源

该器件使用下列 AWIC 模块的内部输入和外部输入。

表 3-11. AWIC 停止唤醒源

唤醒源	说明
可用系统复位	LPO 是其时钟源时的 RESET 引脚
低电压警告	电源管理控制器
IRQ	IRQ 引脚
引脚中断	KBI - 任何使能的引脚中断都能唤醒系统。
ADC	ADC 在使用内部时钟源时可在 Stop 模式下运行。
ACMP	正常中断
I <sup>2</sup> C	地址匹配唤醒
SPI	SPI 从机模式中断
UART	UART_RX 引脚处的 UART 有效边沿检测
RTC	警告中断
不可屏蔽中断	NMI 引脚
MSCAN	MSCAN 唤醒中断

## 3.4 系统模块

### 3.4.1 SIM 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

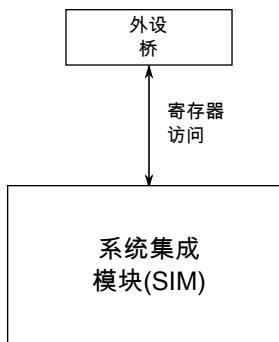


图 3-11. SIM 配置

表 3-12. 相关信息的参考链接

主题	相关模块	参考
完整说明	SIM	<a href="#">SIM</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>

### 3.4.2 PMC 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

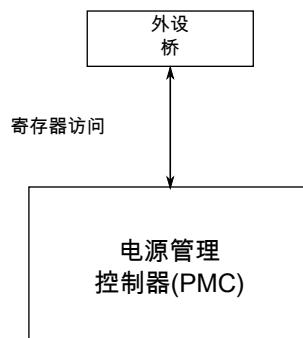


图 3-12. PMC 配置

表 3-13. 相关信息的参考链接

主题	相关模块	参考
完整说明	PMC	<a href="#">PMC</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>

### 3.4.3 MCM 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。



图 3-13. MCM 配置

表 3-14. 相关信息的参考链接

主题	相关模块	参考
完整说明	其他控制模块(MCM)	<a href="#">MCM</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>
专用外设总线(PPB)	ARM Cortex-M0+内核	<a href="#">ARM Cortex-M0+内核</a>
传输	Flash 存储器控制器	<a href="#">Flash 存储器控制器</a>

### 3.4.4 简化交叉开关配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

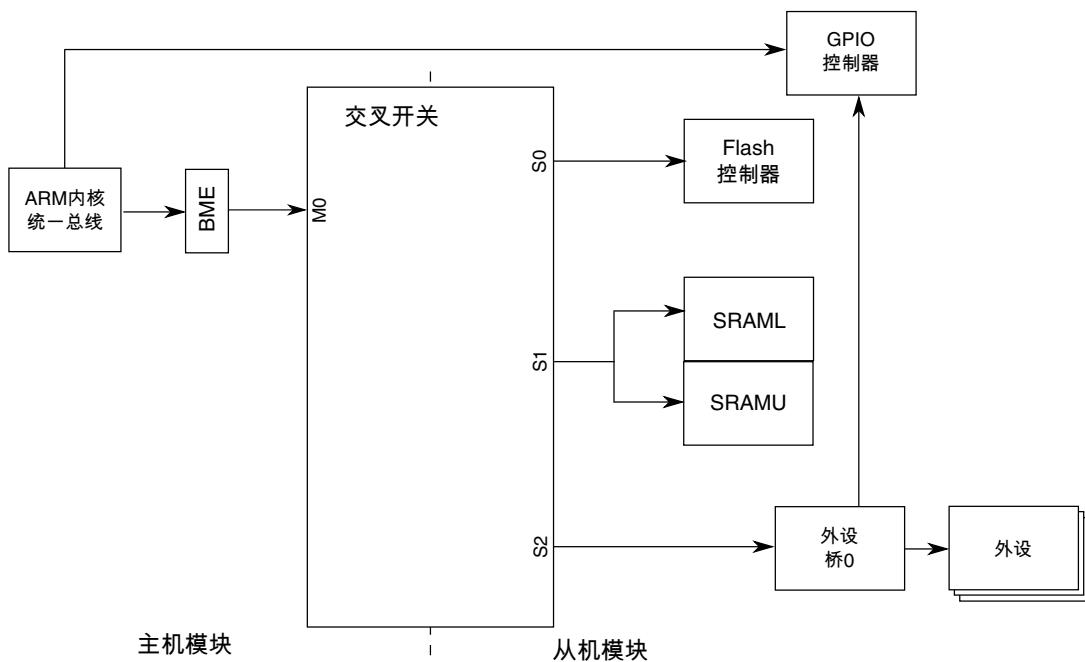


图 3-14. 简化交叉开关集成

表 3-15. 相关信息的参考链接

主题	相关模块	参考
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
交叉开关主机	ARM Cortex-M0+内核	<a href="#">ARM Cortex-M0+内核</a>
交叉开关从机	Flash 存储器控制器	<a href="#">Flash 存储器控制器</a>
交叉开关从机	SRAM 控制器	<a href="#">SRAM 配置</a>
交叉开关从机	外设桥	<a href="#">外设桥</a>
2 端口外设	GPIO 控制器	<a href="#">GPIO 控制器</a>

### 3.4.4.1 简化交叉开关主机分配

与交叉开关相连的主机按以下方式分配：

主机模块	主机端口号
ARM 内核统一总线	0

### 3.4.4.2 交叉开关从机分配

该器件包含 3 个与交叉开关相连的从机。

从机分配如下所示：

从机模块	从机端口号
Flash 存储器控制器	0
SRAM 控制器	1
外设联接器	2

### 3.4.5 外设桥配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。



图 3-15. 外设桥配置

表 3-16. 相关信息的参考链接

主题	相关模块	参考
完整说明	外设桥(AIPS-Lite)	<a href="#">外设桥(AIPS-Lite)</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>

#### 3.4.5.1 外设桥的数量

该器件包含一个外设桥。

#### 3.4.5.2 存储器映像

该外设桥用于访问该器件上大多数模块的寄存器。有关每个模块存储器位置分配的信息，请参见 [AIPS-Lite 存储器映像](#)。

## 3.5 系统安全性

### 3.5.1 CRC 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

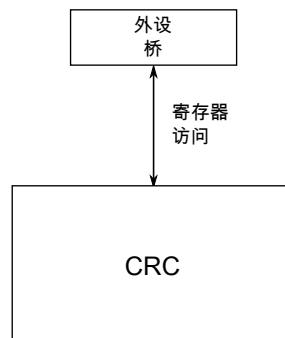


图 3-16. CRC 配置

表 3-17. 相关信息的参考链接

主题	相关模块	参考
完整说明	CRC	<a href="#">CRC</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
电源管理	—	<a href="#">电源管理</a>

### 3.5.2 WDOG 配置

本节总结本模块在芯片级是如何配置的。

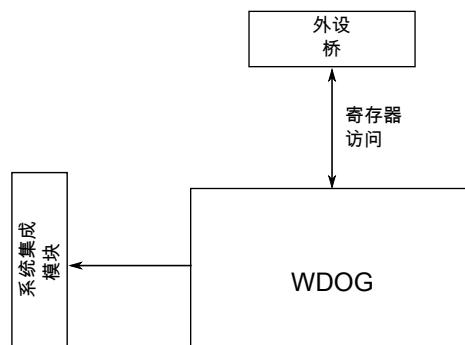


图 3-17. WDOG 配置

表 3-18. 相关信息的参考链接

主题	相关模块	参考
完整说明	WDOG	<a href="#">WDOG</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>
编程模型	系统集成模块(SIM)	<a href="#">SIM</a>

### 3.5.2.1 WDOG 时钟

该 WDOG 具有四个可选时钟源：

- 1 kHz 内部低功耗振荡器(LPOCLK)
- 内部 32 kHz 基准时钟(ICSIRCLK)
- 外部时钟(OSCERCLK)
- 总线时钟

### 3.5.2.2 WDOG 操作

WDOG 模块能提供一套保险机制以确保在系统出现故障时能复位至已知操作状态，如 CPU 时钟停止或在软件代码中出现跑飞现象。WDOG 计数器连续执行可选时钟源并定期使用（刷新）。如果没有，它（WDOG）将复位系统。

任何复位之后，都将使能 WDOG 监视器。如果在应用中未使用 WDOG 监视器，则可通过清零 WDOG\_CS1[EN]将其禁用。

刷新写入序列就是写入 0xA602，然后将 0xB480 写入到 WDOG\_CNT:H:L 寄存器。0xB480 的写入必须在 0xA602 写入后的 16 个总线时钟内进行；否则，监视器将复位 MCU。

监视器计数器有四个时钟源选项，可通过编程 WDOG\_CS2[CLK]进行选择。这四个时钟源选项是总线时钟、内部 1 kHz 时钟、外部时钟或内部 32 kHz 时钟源。

刷新定时溢出时间由 WDOG\_TOVALH:L 定义。此外，如果使用了窗口模式，那么只有在时间值超过（大于）WDOG\_WINH:L 寄存器设置值后，软件才能开始执行刷新序列。

适用于所有时钟源的可选固定预分频器允许较长的定时溢出周期。

WDOG\_CS2[PRES]置位后，对时钟源先进行 256 预分频，随后对 WDOG 计数器进行计时。

WDOG 计数寄存器 CNTH:L 可访问自由运行 WDOG 计数器的值。此软件可随时读取计数寄存器，但不能直接写入 WDOG 计数器。刷新序列将 WDOG 计数器复位至 0x0000。在 16 个总线时钟内依次将 0xC520 和 0xD928 写入 WDOG\_CNTH:L 寄存器可启动解锁序列。完成解锁序列后，用户必须在 128 个总线时钟内重新配置 WDOG，否则 WDOG 会强制对 MCU 进行复位。

默认情况下，WDOG 在“Debug”、“Wait”或“Stop”模式下不起作用。

WDOG\_CS1[DBG]、WDOG\_CS1[WAIT]或 WDOG\_CS1[STOP]置位可激活“Debug”、“Wait”或“Stop”模式下的 WDOG。

## 3.6 时钟模块

### 3.6.1 ICS 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

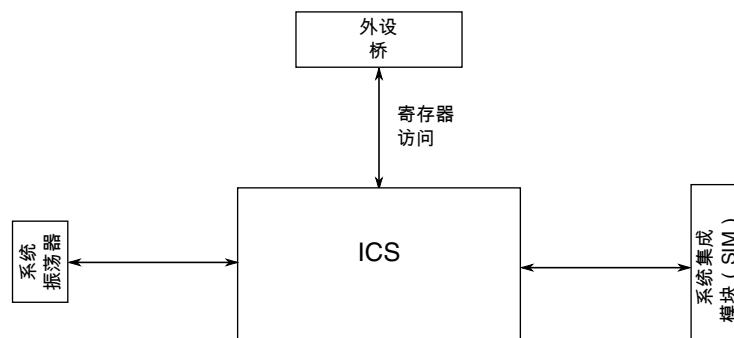


图 3-18. ICS 配置

表 3-19. 相关信息的参考链接

主题	相关模块	参考
完整说明	ICS	<a href="#">ICS</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>

下一页继续介绍此表...

表 3-19. 相关信息的参考链接 (继续)

主题	相关模块	参考
电源管理	—	电源管理

### 3.6.1.1 时钟门控

该系列器件包括每个外设的时钟门控控制，即每个外设的时钟均可使用 SIM\_SCGC 寄存器中的时钟门控控制位直接打开或关闭。

### 3.6.2 OSC 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

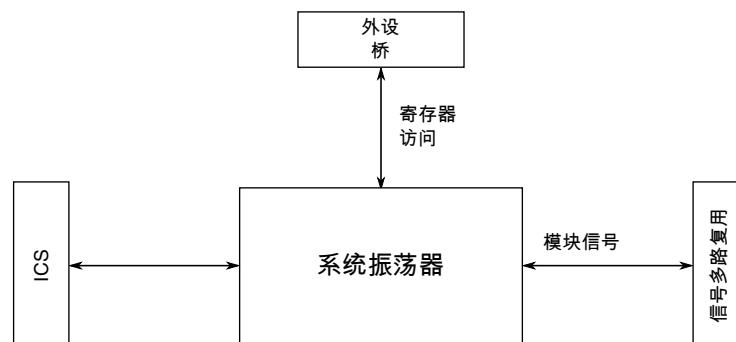


图 3-19. OSC 配置

表 3-20. 相关信息的参考链接

主题	相关模块	参考
完整说明	OSC	OSC
系统存储器映像	—	系统存储器映像
时钟	—	时钟分布
电源管理	—	电源管理
完整说明	ICS	ICS

## 3.7 存储器和存储器接口

### 3.7.1 Flash 存储器配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

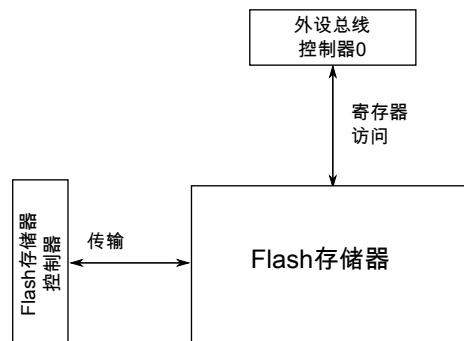


图 3-20. Flash 存储器配置

表 3-21. 相关信息的参考链接

主题	相关模块	参考
完整说明	Flash 存储器	<a href="#">Flash 存储器</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
传输	Flash 存储器控制器	<a href="#">Flash 存储器控制器</a>
寄存器访问	外设桥	<a href="#">外设桥</a>

#### 3.7.1.1 Flash 存储器大小

本文档所涉及器件的 Flash 和存储器数量如下所示：

表 3-22. KEA128 Flash 存储器大小

器件	Flash (KB)	数据块 0 (Flash) 地址范围
S9KEAZ64AMLK(R)	64	0x0000_0000 – 0x0000_FFFF
S9KEAZ128AMLK(R)	128	0x0000_0000 – 0x0001_FFFF
S9KEAZ64AVLK(R)	64	0x0000_0000 – 0x0000_FFFF
S9KEAZ128AVLK(R)	128	0x0000_0000 – 0x0001_FFFF
S9KEAZ64ACLK(R)	64	0x0000_0000 – 0x0000_FFFF
S9KEAZ128ACLK(R)	128	0x0000_0000 – 0x0001_FFFF

下一页继续介绍此表...

表 3-22. KEA128 Flash 存储器大小 (继续)

器件	Flash (KB)	数据块 0 (Flash) 地址范围
S9KEAZ64AMLH(R)	64	0x0000_0000 – 0x0000_FFFF
S9KEAZ128AMLH(R)	128	0x0000_0000 – 0x0001_FFFF
S9KEAZ64AVLH(R)	64	0x0000_0000 – 0x0000_FFFF
S9KEAZ128AVLH(R)	128	0x0000_0000 – 0x0001_FFFF
S9KEAZ64ACLH(R)	64	0x0000_0000 – 0x0000_FFFF
S9KEAZ128ACLH(R)	128	0x0000_0000 – 0x0001_FFFF

### 3.7.1.2 Flash 存储器映像

Flash 存储器和 Flash 寄存器位于不同的基址，如此处图示。

各自对应的基址在 [系统存储器映像](#) 中指定。

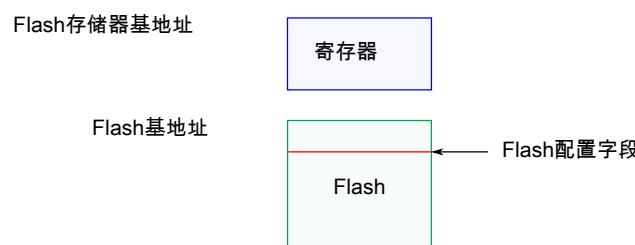


图 3-21. Flash 存储器映像

片上 Flash 存储器在一部分已分配的 Flash 范围中实现，从而在起始地址为 0x0000\_0000 的存储器映像中形成一个相邻块。有关支持范围的详情，请参见 [Flash 存储器大小](#)。

访问超出器件上 Flash 数量的 Flash 存储器范围会导致总线周期终止，同时在发出请求的总线主机中进行相应响应后会出现错误。

### 3.7.1.3 备用非易失性 IRC 用户微调说明

表 3-23 中所示的非易失性位置 (2 字节) 保留用于某些开发和编程工具支持的自定义 ICS 内部参考时钟 (IRC) 调整值。对出厂加载调整值所作的备用 IRC 调整可存储在这些位置。要覆盖该出厂调整，用户软件必须将这些位置的自定义调整值复制到 ICS\_C3 和 ICS\_C4 的 ICS 调整字段中。

表 3-23. 备用非易失性 IRC 调整

非易失性字节地址	备用 IRC 调整值
0x0000_03FE (位 0)	SCFTRIM
0x0000_03FF	SCTRIM

### 3.7.1.4 Flash 加密

有关如何在该器件上实施 Flash 加密的信息，请参见[芯片加密](#)。

### 3.7.1.5 擦除所有 Flash 内容

除软件外，整个 Flash 存储器也可通过 SW-DP 调试端口外部擦除，方法是置位 MDM-AP CONTROL[0] (MDM-AP 控制寄存器的位 0)。MDM-AP STATUS[0] (MDM-AP 状态寄存器的位 0) 置位以说明已接受整体擦除命令。MDM-AP CONTROL[0] 在整体擦除完成时清零。

## 3.7.2 Flash 存储器控制器配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

有关 FMC 复位配置的详情，请参见[平台控制寄存器 \(MCM\\_PLACR\) 寄存器描述](#)。

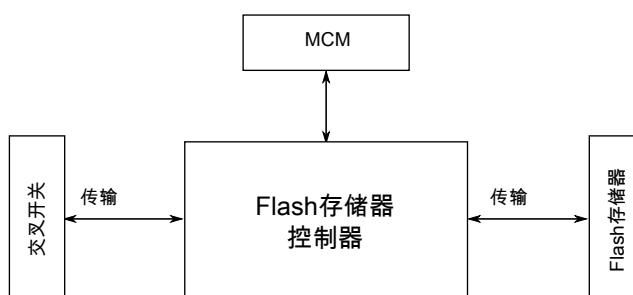


图 3-22. Flash 存储器控制器配置

表 3-24. 相关信息的参考链接

主题	相关模块	参考
完整说明	Flash 存储器控制器	<a href="#">Flash 存储器控制器</a>
系统存储器映像		<a href="#">系统存储器映像</a>

下一页继续介绍此表...

表 3-24. 相关信息的参考链接 (继续)

主题	相关模块	参考
时钟		时钟分布
传输	Flash 存储器	Flash 存储器
传输	交叉开关	交叉开关
寄存器访问	MCM	MCM

### 3.7.3 SRAM 配置

本节总结本模块在芯片级是如何配置的。

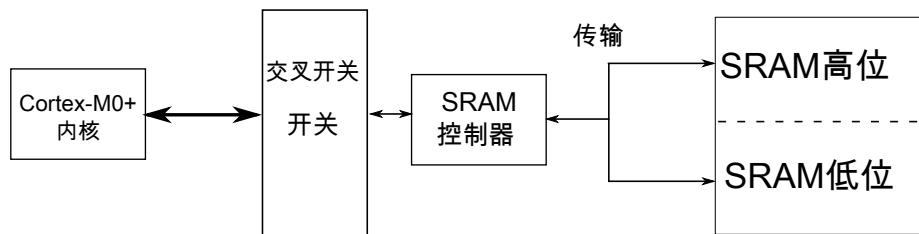


图 3-23. SRAM 配置

表 3-25. 相关信息的参考链接

主题	相关模块	参考
完整说明	SRAM	SRAM
系统存储器映像	—	系统存储器映像
时钟	—	时钟分布
ARM Cortex-M0+内核	—	ARM Cortex-M0+内核

#### 3.7.3.1 SRAM 大小

SRAM 支持在所有内核频率下单周期访问 (零等待状态)。

本文档所涉及器件的 SRAM 数量如下所示：

表 3-26. SRAM 大小

Freescale 部件编号	SRAM
S9KEAZ64AMLH(R)	8 KB
S9KEAZ128AMLH(R)	16 KB
S9KEAZ64AMLK(R)	8 KB

下一页继续介绍此表...

表 3-26. SRAM 大小 (继续)

Freescale 部件编号	SRAM
S9KEAZ128AMLK(R)	16 KB
S9KEAZ64AVLK(R)	8 KB
S9KEAZ128AVLK(R)	16 KB
S9KEAZ64ACLK(R)	8 KB
S9KEAZ128ACLK(R)	16 KB
S9KEAZ64AVLH(R)	8 KB
S9KEAZ128AVLH(R)	16 KB
S9KEAZ64ACLH(R)	8 KB
S9KEAZ128ACLH(R)	16 KB

### 3.7.3.2 SRAM 范围

片上 SRAM 分为两个范围；1/4 被分配到 SRAM\_L，3/4 被分配到 SRAM\_U。

片上 SRAM 的实现还可以让 SRAM\_L 和 SRAM\_U 范围在存储器映射中形成一个相邻块。例如：

- SRAM\_L 固定至 0x1FFF\_FFFF 并占用该结束地址前的空间。
- SRAM\_U 固定至 0x2000\_0000 并占用该起始地址后的空间。

SRAM\_L 和 SRAM\_U 的有效地址范围随后被定义为：

- $\text{SRAM\_L} = [0x2000\_0000 - (\text{SRAM\_size}/4)] \text{ 至 } 0x1FFF\_FFFF$
- $\text{SRAM\_U} = 0x2000\_0000 \text{ 至 } [0x2000\_0000 + (\text{SRAM\_size} * (3/4)) - 1]$

如下图所示。

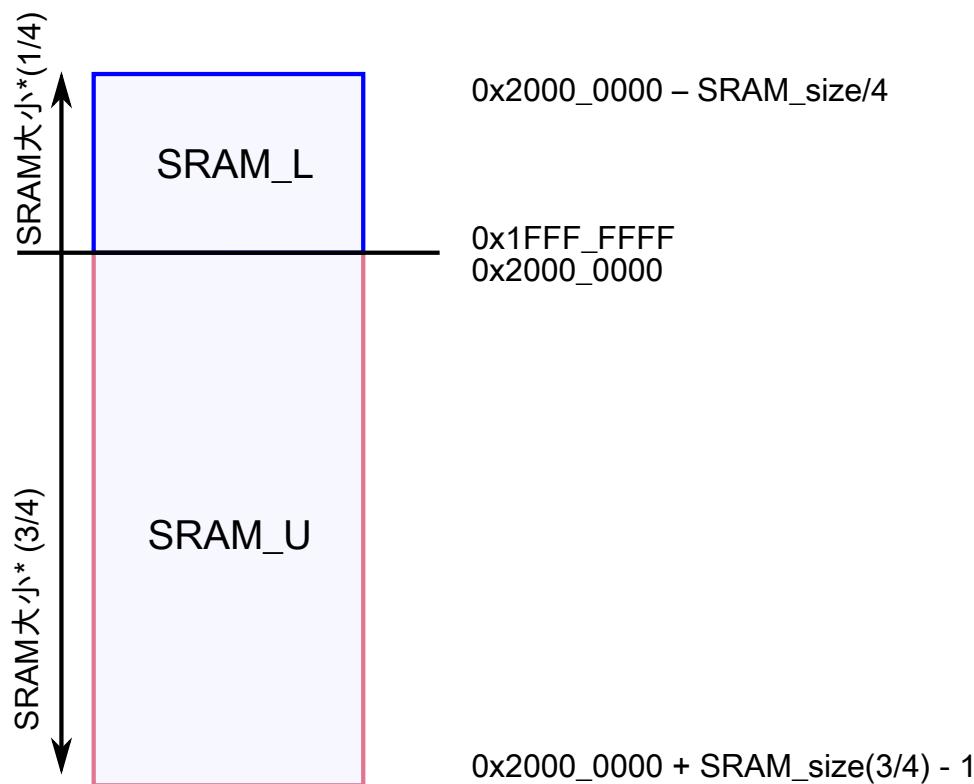


图 3-24. SRAM 块存储器映射

例如，对于包含 16 KB SRAM 的器件，其范围为：

- SRAM\_L: 0x1FFF\_F000 – 0x1FFF\_FFFF
- SRAM\_U: 0x2000\_0000 – 0x2000\_2FFF

### 3.7.3.3 SRAM 位操作

片上 SRAM 分为两个范围：SRAM\_L 和 SRAM\_U。SRAM\_U 范围通过两种方式支持该器件上的位操作：

- 位带区别名
- 位操作引擎（BME）

该位带区别名中的 32 位写操作与对 SRAM\_U 区中目标位进行的读-修改-写操作的作用相同。详情请参见[位带区别名](#)。

位带区别名仅支持简单置位或清零操作。通过 BME 引擎可进一步支持更复杂的位操作（AND、OR、XOR 等）。详情参见[位操作引擎](#)。

## 3.8 模拟

### 3.8.1 12 位模数转换器(ADC)配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

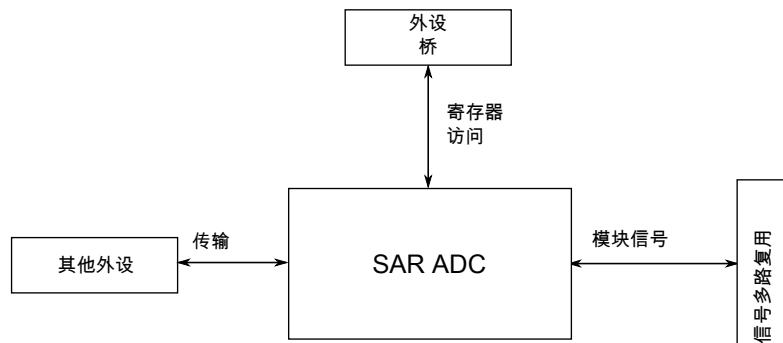


图 3-25. 12 位 SAR ADC 配置

表 3-27. 相关信息的参考链接

主题	相关模块	参考
完整说明	12 位 SAR ADC	<a href="#">12 位 SAR ADC</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>

#### 3.8.1.1 ADC 实例化信息

该器件包含一个 12 位逐次逼近型 ADC，具有多达 16 个通道。

表 3-28. ADC 通道

Freescale 部件编号	ADC 通道
S9KEAZ64AMLH(R)	16
S9KEAZ128AMLH(R)	16
S9KEAZ64AMLK(R)	16
S9KEAZ128AMLK(R)	16
S9KEAZ64AVLK(R)	16
S9KEAZ128AVLK(R)	16
S9KEAZ64ACLK(R)	16

下一页继续介绍此表...

表 3-28. ADC 通道 (继续)

Freescale 部件编号	ADC 通道
S9KEAZ128ACLK(R)	16
S9KEAZ64AVLH(R)	16
S9KEAZ128AVLH(R)	16
S9KEAZ64AC LH(R)	16
S9KEAZ128AC LH(R)	16

ADC 支持软件和硬件触发。ADC 硬件触发 (ADHWT) 可选择性来自 ACMP0、ACMP1、FTM0 初始化触发器、FTM2 初始化触发器、FTM2 匹配触发器、RTC 溢出或 PITCHO/1 溢出。硬件触发可配置为在 MCU “Run”、“Wait”和“Stop”模式下引起硬件触发。

硬件触发源的详细信息列示于“[模块到模块](#)”一节。

### 3.8.1.2 ADC0 连接/通道分配

该器件的 ADC 通道分配如下表所示。已保留的通道转换成未知值。

表 3-29. ADC 通道分配

ADCH	通道	输入
00000	AD0	PTA0/ADP0
00001	AD1	PTA1/ADP1
00010	AD2	PTA6/ADP2
00011	AD3	PTA7/ADP3
00100	AD4	PTB0/ADP4
00101	AD5	PTB1/ADP5
00110	AD6	PTB2/ADP6
00111	AD7	PTB3/ADP7
01000	AD8	PTC0/ADP8
01001	AD9	PTC1/ADP9
01010	AD10	PTC2/ADP10
01011	AD11	PTC3/ADP11
01100	AD12	PTF4/ADP12
01101	AD13	PTF5/ADP13
01110	AD14	PTF6/ADP14
01111	AD15	PTF7/ADP15
10000	AD16	Vss
10001	AD17	Vss
10010	AD18	Vss

下一页继续介绍此表...

表 3-29. ADC 通道分配 (继续)

ADCH	通道	输入
10011	AD19	Vss
10100	AD20	保留
10101	AD21	保留
10110	AD22	温度传感器
10111	AD23	带隙基准
11000	AD24	保留
11001	AD25	保留
11010	AD26	保留
11011	AD27	保留
11100	AD28	保留
11101	AD29	VREFH
11110	AD30	VREFL
11111	模块已禁用	无

### 3.8.1.3 ADC 模拟电源和基准连接

该器件包括 80LQFP 封装上专用的 VDDA 引脚、VSSA 引脚、VREFH 引脚和 VREFL 引脚。专用 VREFL 引脚在 64QFP\LQFP 封装和 LQFP44 封装上可用，而 VREFH 引脚与 VDDA 内部相连。

### 3.8.1.4 温度传感器和带隙基准

ADC 模块集成了片上温度传感器。要使用此温度传感器，必须执行以下操作。

- 用最高 1 MHz 的时钟配置 ADC，以便进行长时间采样
- 转换带隙基准电压通道(AD23)
  - 通过使用  $V_{BG}$  的值转换带隙基准电压通道的数字值，用户就可确定  $V_{DD}$ 。
- 转换温度传感器通道(AD22)
  - 通过使用计算所得的  $V_{DD}$  值将 AD22 的数字值转换成电压， $V_{TEMP}$

### 3.8.1.5 备选时钟

ADC 模块能够使用 MCU 总线时钟、总线时钟 2 分频、模块中的本地异步时钟 (ADACK)或备选时钟 ALTCLK 来执行转换。这些器件的备选时钟是外部振荡器输出(OSC\_OUT)。

ADC 选定的时钟源在经过 ADC\_SC3[ADIV]分频后，必须符合  $f_{ADC_K}$  指定的时钟范围。

假如上述条件均已满足，则 ALTCLK 在 MCU 处于 Wait 模式时处于有效状态。这样 ALTCLK 就能在 MCU 处于 Wait 模式时用作 ADC 的转换时钟源。

MCU 处于 Stop 模式时，ALTCLK 不能用作 ADC 的转换时钟源。

### 3.8.2 ACMP 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

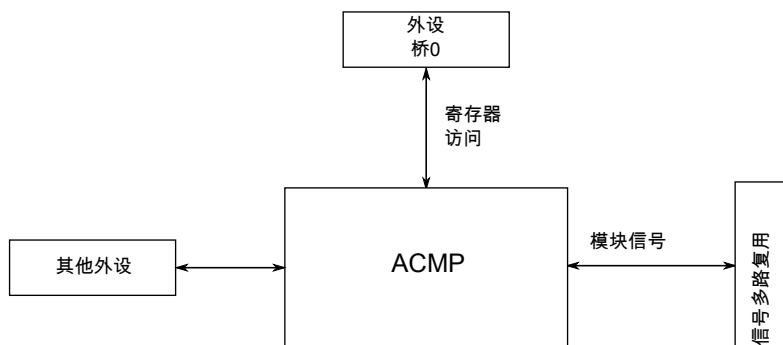


图 3-26. ACMP 配置

表 3-30. 相关信息的参考链接

主题	相关模块	参考
完整说明	模拟比较器(ACMP)	<a href="#">比较器</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>

#### 3.8.2.1 ACMP 概述

该器件包含两个模拟比较器模块(ACMP)，该模块提一个电路，用于比较两个模拟输入电压或比较一个模拟输入电压与一个内部基准电压。该比较器电路适用于在整个供电电压范围内操作（全摆幅操作）。

ACMP 具有四种不同的输入，由 ACMP 的正输入和负输入多路复用。其中一个固定连接到内置 DAC 输出，其他则在外部映射到管脚。

ACMP 模块支持内部带隙基准电压。使用带隙基准时，用户必须先使能 PMC 带隙基准缓冲器。

如果已使能“Wait”和“Stop”模式，则 ACMP 模块就能继续在这些模式下操作，并且能够在发生比较事件时唤醒 MCU。

### 3.8.2.2 ACMP 互相连接

通过设置 SIM\_SOPT0[ACIC]可将 ACMP0 输出配置为与 FTM1 输入捕捉通道 0 相连。在 ACIC 字段的电平为有效的情况下，FTM1\_CH0 引脚在外部不可用，无论通道 0 的FTM1 模块配置如何。

ACMP0 和 ACMP1 输出也是内部连接到 FTM2 触发输入和故障输入。通过配置 SIM\_SOPT0[ACTRG]，用户可将 ACMP0\_OUT 或 ACMP1\_OUT 连接到 FTM2 触发输入 0。ACMP0\_OUT 连接到 FTM2 故障输入 0，而 ACMP1\_OUT 则连接到 FTM2 故障输入 3。

通过设置 SIM\_SOPT0[RXDCE]可将 ACMP0 输出直接投射至 UART0\_RX。在该模式下，UART0\_RX 管脚不工作。任何连接到 ACMP0 输入的外部信号都可被视为输入引脚。

ACMP0 和 ACMP1 输出可在 PWT 模块上进行测量。通过配置 SIM\_SOPT1[ACPWTS]位，用户可将 ACMP0\_OUT 或 ACMP1\_OUT 连接到 PWT 输入 2。

下表显示的是 ACMP0 和 ACMP1 的输入连接：

**表 3-31. ACMP0 输入连接**

ACMP0 通道	连接
0	PTA0/ACMP0_IN0
1	PTA1/ACMP0_IN1
2	PTC4/ACMP0_IN2
3	DAC 输出

**表 3-32. ACMP1 输入连接**

ACMP1 通道	连接
0	PTA6/ACMP1_IN0
1	PTA7/ACMP1_IN1
2	PTB4/ACMP1_IN2
3	DAC 输出

### 3.8.2.3 Stop 模式下的 ACMP

如已使能，ACMP 将继续在 Stop 模式下工作。ACMPx\_SC[ACOPE]如已使能，则比较器输出将以正常工作模式工作并将控制 ACMPx\_OUT 引脚。当发生比较事件且 ACMPx\_CS[ACIE]使能时，MCU 会退出 Stop 模式；ACMPx\_CS[ACF]标志相应地置位。

## 3.9 定时器

### 3.9.1 FlexTimer 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

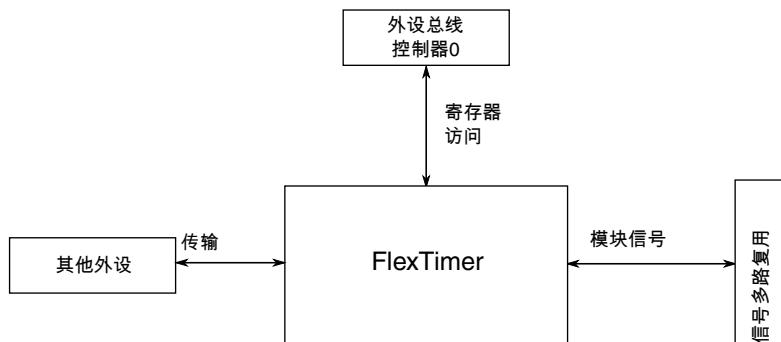


图 3-27. FlexTimer 配置

表 3-33. 相关信息的参考链接

主题	相关模块	参考
完整说明	FlexTimer	<a href="#">FlexTimer</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>
信号多路复用	端口控制	<a href="#">信号多路复用</a>

### 3.9.1.1 FTM 概述

FTM 定时器最多包含六个通道，这些通道支持输入捕捉、输出比较和 PWM 信号的生成以控制电动机和电源管理应用。FTM 时间基准为 16 位计数器，它可用作无符号的计数器或有符号的计数器。

该器件最多包含具有全部功能的一个 6 通道 FTM 和具有基本 TPM 功能的两个 2 通道 FTM 的三个 FTM 模块。每个 FTM 模块都可以使用独立的外部时钟输入。下表总结了 FTM 模块的配置。

表 3-34. FTM 模块功能

特性	FTM0/FTM1	FTM2
通道数	2	6
初始计数值	否	是
周期性 TOF	否	是
输入捕捉模式	是	是
通道输入滤波器	否	通道 0、1、2 和 3
输出比较模式	是	是
边沿对齐 PWM (EPWM)	是	是
中心对齐 PWM (CPWM)	是	是
组合模式	否	是
互补模式	否	是
PWM 同步	否	是
反相	否	是
软件输出控制(SWOC)	否	是
死区时间插入	否	是
输出掩码	否	是
故障控制	否	是
故障输入数	0	4
故障输入滤波器	否	故障输入 0、1、2 和 3
极性控制	否	是
初始化	否	是
通道匹配触发器	否	是
初始化触发器	是	是
捕捉测试模式	否	是
DMA	否	否
双边沿捕捉模式	否	是
正交解码器模式	否	否
正交解码器输入滤波器	否	否
Debug 模式	否	是
中间负载	否	是
全局时基使能 <sup>1</sup>	否	是
可用的寄存器	FTM_SC、FTM_CNT、 FTM_MOD、 FTM_C0SC、 FTM_C0V、FTM_C1SC 以及 FTM_C1V、 FTM_EXTTRIG	FTM_SC、FTM_CNT、FTM_MOD、FTM_C0SC、 FTM_C0V、FTM_C1SC 以及 FTM_C1V、 FTM_C2SC、FTM_C2V、FTM_C3SC、 FTM_C3V、FTM_C4SC、FTM_C4V、 FTM_C5SC、FTM_C5V、FTM_CNTIN、 FTM_STATUS、FTM_MODE、FTM_SYNC、 FTM_OUTINIT、FTM_OUTMASK、 FTM_COMBINE、FTM_DEADTIME、

表 3-34. FTM 模块功能

特性	FTM0/FTM1	FTM2
		FTM_EXTTRIG、FTM_POL、FTM_FMS、 FTM_FILTER、FTM_FLCTRL、FTM_CONF、 FTM_FLTPOL、FTM_SYNCONF、 FTM_INVCTRL、FTM_SWOCTRL 以及 FTM_PWMLOAD

1. 全局时基(GTB)允许在一个芯片上同步多个 FTM 模块。它需要由所有相关的 FTM 模块支持的 GTB 功能。在该器件上，仅一个 FTM 模块(FTM2)支持 GTB 功能，因此，GTB 功能实际不可用。

### 3.9.1.2 FTM 时钟选项

可选择的 FTM 源时钟包括、定时器时钟（最高 48 MHz）固定频率时钟或外部时钟。选定的控制源由 FTMx\_SC[CLKS]控制。

- 当 FTMx\_SC[CLKS]等于 00 时，没有选择任何时钟（这实际上会禁用 FTM 计数器）。
- 当 FTMx\_SC[CLKS]等于 01 时，选择定时器时钟。
- 当 FTMx\_SC[CLKS]等于 10 时，选择固定频率时钟(ICSFFCLK)。
- 当 FTMx\_SC[CLKS]等于 11 时，选择外部时钟。

### 3.9.1.3 FTM 互连

FTM0 有下列互连：

- UART0\_TX 信号可通过 FTM0 通道 0 PWM 输出调制。
- UART0\_RX 信号可通过将 1 写入 SIM\_SOPT0[RXDCE]由 FTM0 通道 1 输入捕捉功能标记。

FTM1 拥有以下互相连接：

- ACMP0 输出可通过将 1 写入 SIM\_SOPT0[ACIC]在内部与 FTM1 通道 0 捕捉输入相连。
- RTC 溢出可通过将 1 写入 SIM\_SOPT0[RTCC]与 FTM1 通道 1 捕捉输入相连。

FTM2 支持三种 PWM 同步源：

- Trigger0 可通过将 0 或 1 写入 SIM\_SOPT0[ACTRG]与 ACMP0 或 ACMP1 的输出相连。
- Trigger1 连接到 FTM0 通道 0 输出。
- Trigger2 是采用将 1 写入 SIM\_SOPT0[FTMSYNC]的方法而进行触发的软件触发。

FTM2 支持四种 FTM 故障源：

- 故障 0 连接到 ACMP0 输出。
- 故障 1 连接到 PTA6。
- 故障 2 连接到 PTA7。
- 故障 3 连接到 ACMP1 输出。

### 3.9.1.4 FTM 中断

FlexTimer 有多个中断源。但是，每个中断源都能对中断控制器生成单个中断请求。当出现 FTM 中断时，读取 FTM 状态寄存器以确定确切的中断源。

## 3.9.2 PIT 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

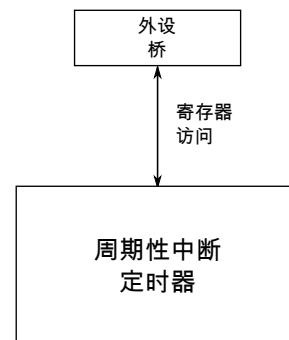


图 3-28. PIT 配置

表 3-35. 相关信息的参考链接

主题	相关模块	参考
完整说明	PIT	<a href="#">PIT</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>

### 3.9.2.1 PIT 概述

PIT 模块是一系列可用于发起中断和触发的定时器。

该器件包含一个支持链式定时器模式的双通道 PIT 模块。

### 3.9.2.2 PIT 互相连接

通过 SIM\_SOPT0[ADHWT]置位, 可将 PIT 通道 0 和通道 1 触发器输出用作 ADC 硬件触发。

### 3.9.3 RTC 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述, 请参见各模块相关章节。

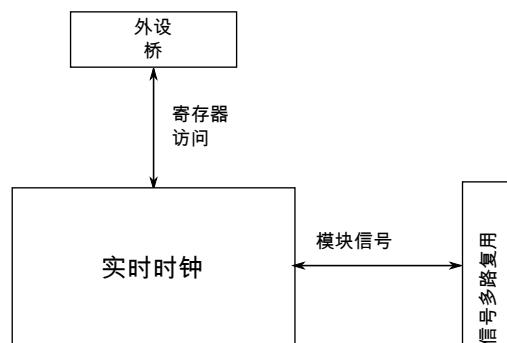


图 3-29. RTC 配置

表 3-36. 相关信息的参考链接

主题	相关模块	参考
完整说明	RTC	<a href="#">RTC</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>

### 3.9.3.1 RTC 概述

在该器件上使用的实时计数器(RTC)包括一个 16 位计数器、一个 16 位比较器、若干个二进制和十进制预分频器、四个时钟源和一个可编程周期中断。该模块可用于当日时间、日历或任何任务调度功能。该模块可以在无外部组件的情况下实现低功耗周期性的自动唤醒。

### 3.9.3.2 RTC 互连

四个软件可选择时钟源可以被选择作为基于二进制或者十进制的预分频器的输入源

- 1 kHz 内部低功耗振荡器(LPOCLK)
- 外部时钟(OSCERCLK)
- 32 kHz 内部基准时钟(ICSIRCLK)
- 总线时钟

RTC 溢出触发器通过配置 SIM\_SOPT0[ADHWT]可用作 ADC 的硬件触发，通过配置SIM\_SOPT0[RTCC]还能由 FTM1 通道 1 捕捉。

### 3.9.4 PWT 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

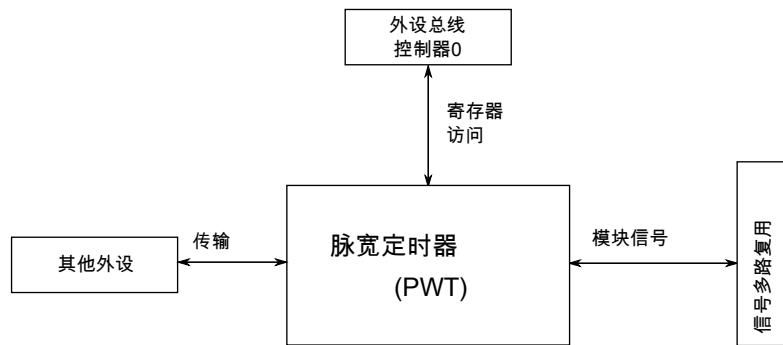


图 3-30. PWT 配置

表 3-37. 相关信息的参考链接

主题	相关模块	参考
完整说明	PWT	<a href="#">PWT</a>
系统存储器映像		<a href="#">系统存储器映像</a>
时钟		<a href="#">时钟分布</a>
电源管理		<a href="#">电源管理</a>
信号多路复用	端口控制	<a href="#">信号多路复用</a>

#### 3.9.4.1 PWT 概述

该器件上的脉冲宽度定时器(PWT)模块由一个 16 位计数器构成，可用于捕捉或测量映射在其输入通道上的脉冲宽度。

PWT 的计数器具有两个与 FTM 模块共享的可选择时钟源，支持频率高达 48MHz 的内部定时器时钟。PWT 模块支持可编程的正脉冲边沿或负脉冲边沿，以及基于脉冲宽度或计数器溢出的可编程中断生成。

### 3.9.4.2 PWT 互连

有两个软件可选择时钟源可用于 PWT 模块预分频器的输入：

- 定时器时钟：最多 48 MHz，也是 FTM 模块的时钟源选项
- TCLK：来自面板的外部时钟

PWT 模块具有四个输入通道，其连接如下所示：

表 3-38. PWT 输入连接

PWT 输入通道	连接
0	PTD5 或 PTE2
1	PTB0 或 PTH7
2	ACMPO 输出或 ACMP1 输出
3	UART0_RX、UART1_RX 或 UART2_RX

## 3.10 通信接口

### 3.10.1 SPI 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

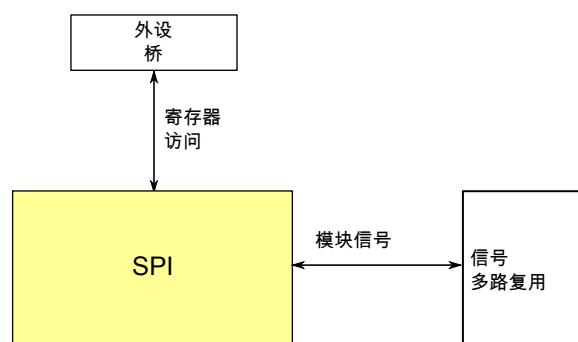


图 3-31. SPI 配置

表 3-39. 相关信息的参考链接

主题	相关模块	参考
完整说明	SPI	<a href="#">SPI</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>

### 3.10.1.1 SPI 概述

该器件包含两个支持 8 位数据长度的 SPI 模块。

串行外设接口(SPI)模块为 MCU 和外设之间的全双工、同步和串行通信而提供。这些外设可包括其他微控制器、模数转换器、移位寄存器、传感器和存储器等。

SPI 在主机模式下的运行波特率高达总线时钟除以 2, 在从机模式中高达总线时钟除以 4。软件可以轮询该状态标志，或可使用中断方式驱动 SPI 操作。

### 3.10.2 I<sup>2</sup>C 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

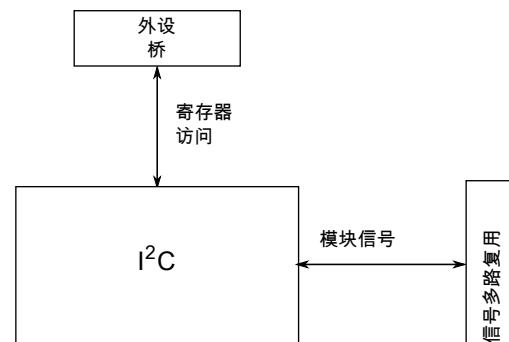


图 3-32. I<sup>2</sup>C 配置

表 3-40. 相关信息的参考链接

主题	相关模块	参考
完整说明	I <sup>2</sup> C	<a href="#">I<sup>2</sup>C</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>

### 3.10.2.1 I<sup>2</sup>C 概述

该器件包含两个具有 SMBus 特性的 I<sup>2</sup>C 模块。I<sup>2</sup>C 用于多个器件之间的通信。该接口在总线负荷和时序最高的情况下以高达 100 kb/s 的速度运行。该器件能够以更高的波特率运行，最高可达时钟/20 的最大值，同时总线负荷更低。最大通信长度和可连接器件数受限于 400 pF 的最大总线电容。

I2C0 还提供 4 线式接口选项。

### 3.10.2.2 I2C0 4 线式接口特性

I2C0 4 线式接口凭借 4 个单向信号而非标准 I2C 的 SCL 和 SDA 这 2 个双向信号来提高 I2C 的抗噪性。

I2C0 提供 4 线式接口选项。SIM\_SOPT1[I2C04WEN]置位时，SDA/SCL 输入来自 SDA\_IN/SCL\_IN，同时 SDA\_OUT/SCL\_OUT 上存在 SDA/SCL 输出。

在 4 线式接口特性因 SIM\_SOPT1[I2C04WEN]位置位而使能后，用户可使 SIM\_SOPT1[I2C0OINV]置位，这样在输出之前 SDA\_OUT/SCL\_OUT 就会被反相。该特性仅当未重新映射 I2C0 管脚时可用。

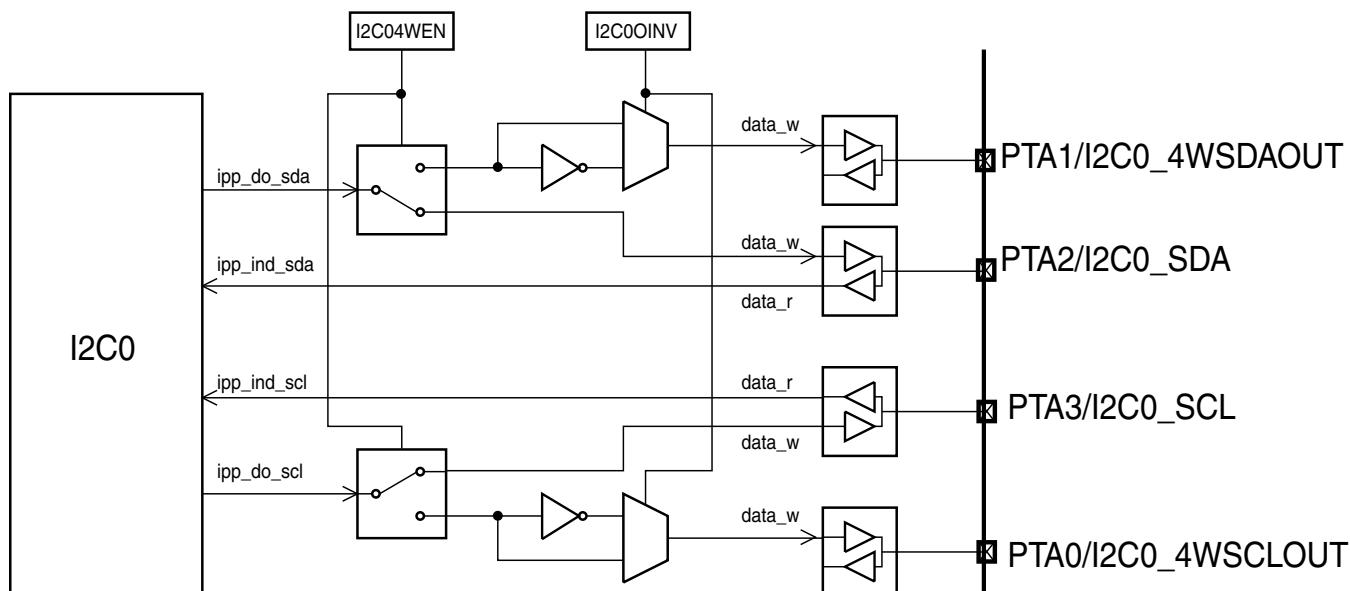


图 3-33. I2C0 4 线式接口图

### 3.10.3 UART 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

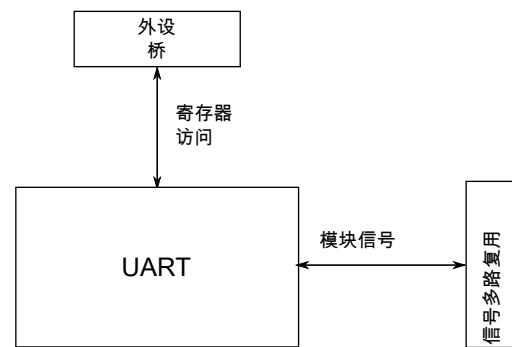


图 3-34. UART 配置

表 3-41. 相关信息的参考链接

主题	相关模块	参考
完整说明	UART	<a href="#">UART</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>

### 3.10.3.1 UART 概述

该器件包含三通用异步收发器(UART) 模块。通常，这些系统可连接到个人计算机或工作站的 RS232 串行输入/输出端口。这些系统还可用于与其他嵌入式控制器通信。

一个可变并基于模数的 13 位波特率生成器支持 115.2 kbaud 以上的大范围标准波特率。在相同的 UART 中发送和接收可使用常见的波特率，每个 UART 模块都具有单独的波特率生成器。

此 UART 系统提供在其他嵌入式控制器的异步串行 I/O 外设上许多不常见的高级功能。该接收器采用一种高级数据采样技术确保通信和噪音检测可靠。还包括发送和接收中的硬件奇偶校验、接收器唤醒和双缓冲。

### 3.10.3.2 UART 互连

UART0 可通过以下技巧实现红外功能：

UART0\_TX 调制：

- UART0\_TX 输出可通过 FTMO 通道 0 PWM 输出调制

UART0\_RX 标签：

- UART0\_RX 输入可标记到 FTM0 通道 1 或交由 ACMP0 或 ACMP1 模块滤波

UART0\_RX、UART1\_RX 和 UART2\_RX 可通过 PWT 测定。用户通过配置 SIM\_SOPT1[UARTPWTS]，可将 UART0\_RX、UART1\_RX 或 UART2\_RX 连接到 PWT\_IN3。

### 3.10.4 MSCAN 配置

本节总结本模块在芯片级是如何配置的。有关模块本身的全面阐述，请参见各模块相关章节。

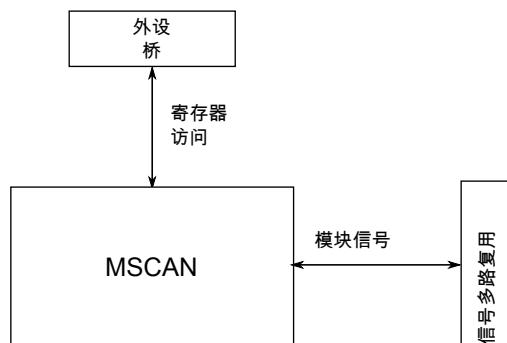


图 3-35. MSCAN 配置

表 3-42. 相关信息的参考链接

主题	相关模块	参考
完整说明	MSCAN	<a href="#">MSCAN</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>

#### 3.10.4.1 MSCAN 概述

该器件包含 CAN 模块。它使用 MSCAN 模块，后者是采用 CAN 2.0A/B 协议的通信控制器，具体请参见 1991 年 9 月定义的 Bosch 规范。

#### 3.10.4.2 MSCAN 时钟源

MSCAN 模块具有可编程时钟源。它可由总线时钟或外部振荡器时钟(OSCERCLK)计时。用户可对 MSCAN\_CANCTL1[CLKSRC]进行配置以选择使用过的时钟。

将 OSCERCLK 选作 MSCAN 时钟时，其频率不得高于 24 MHz。

### 3.10.4.3 MSCAN 唤醒中断和去抖滤波器

当检测到 CAN 总线活动时，可将 MSCAN 编程为从睡眠或 Stop 模式唤醒。对现有 CAN 总线动作的敏感度是可以修改的，方法是：通过使 MSCAN\_CANCTL1[WUPM] 置位，将低通过滤波器功能应用到 RXCAN 输入线路。该特性可防止 MSCAN 因 CAN 总线线路上的短时毛刺而唤醒。

## 3.11 人机接口(HMI)

### 3.11.1 GPIO 配置

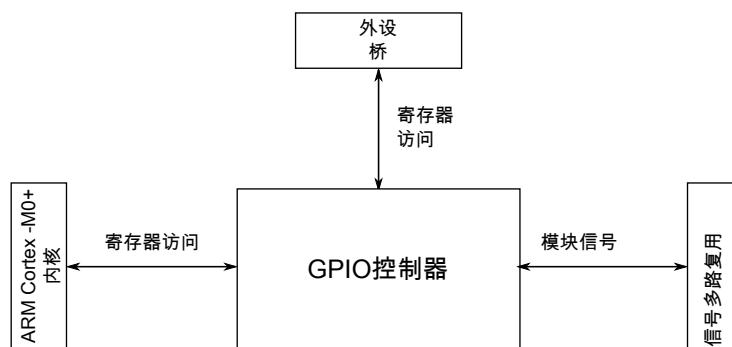


图 3-36. GPIO 配置

表 3-43. 相关信息的参考链接

主题	相关模块	参考
完整说明	GPIO	<a href="#">GPIO</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>
交叉开关	交叉开关	<a href="#">交叉开关</a>

### 3.11.1.1 GPIO 概述

GPIO 具有多个端口，内核可以对基址 0xF800\_0000(FGPIO)进行零等待状态直接访问。另外还可由内核通过 0x400F\_F000 处的交叉开关/AIPS 接口以及地址 0x4000\_F000 处的别名插槽(15)对其进行访问。对 GPIO 空间的所有 BME 操作都可通过引用地址 0x4000\_F000 处的别名插槽(15)完成。只有部分 BME 操作可通过引用地址 0x400F\_F000 处的 GPIO 完成。

### 3.11.2 KBI 配置

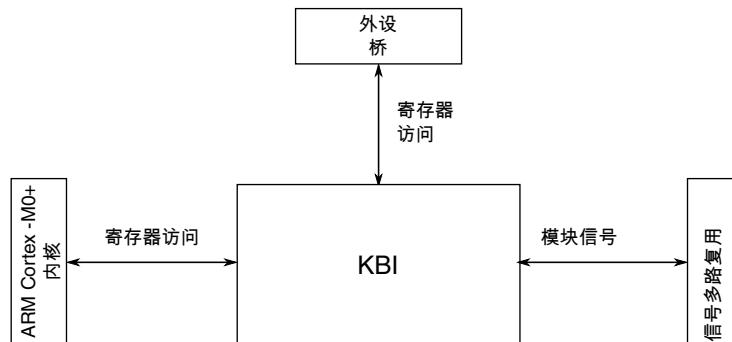


图 3-37. KBI 配置

表 3-44. 相关信息的参考链接

主题	相关模块	参考
完整说明	KBI	<a href="#">KBI</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>
交叉开关	交叉开关	<a href="#">交叉开关</a>

#### 3.11.2.1 KBI 概述

该器件具有两个 32 位键盘中断模块(KBI)，并且最多有 64 个键盘中断输入可用，具体取决于封装。

#### 3.11.2.2 KBI 分配

KBI 端口分配如下表所示。

表 3-45. KBI 端口分配

KBI	输入自
KBI0P0 ~ KBI0P7	PTA0 ~ PTA7
KBI0P8 ~ KBI0P15	PTB0 ~ PTB7
KBI0P16 ~ KBI0P23	PTC0 ~ PTC7
KBI0P24 ~ KBI0P31	PTD0 ~ PTD7
KBI1P0 ~ KBI1P7	PTE0 ~ PTE7

下一页继续介绍此表...

表 3-45. KBI 端口分配 (继续)

KBI	输入自
KBI1P8 ~ KBI1P15	PTF0 ~ PTF7
KBI1P16 ~ KBI1P23	PTG0 ~ PTG7
KBI1P24 ~ KBI1P31	PTH0 ~ PTH7

### 3.11.3 IRQ 配置

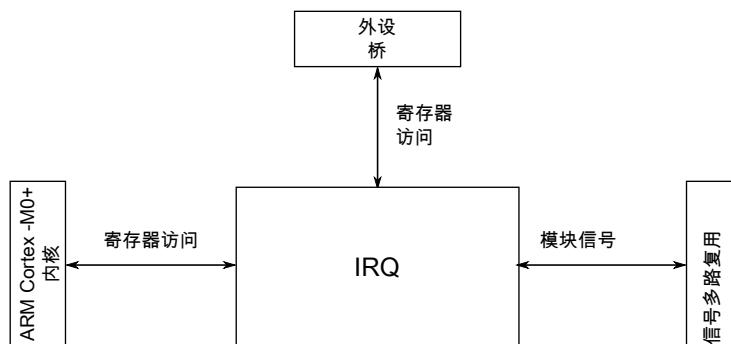


图 3-38. IRQ 配置

表 3-46. 相关信息的参考链接

主题	相关模块	参考
完整说明	IRQ	<a href="#">IRQ</a>
系统存储器映像	—	<a href="#">系统存储器映像</a>
时钟	—	<a href="#">时钟分布</a>
电源管理	—	<a href="#">电源管理</a>
交叉开关	交叉开关	<a href="#">交叉开关</a>

#### 3.11.3.1 IRQ 分配

默认情况下，IRQ 分配至引脚 PTA5。通过配置 SIM\_PINSEL0[IRQPS]，可将 IRQ 重新分配至引脚 PTI0、PTI1、PTI2、PTI3、PTI4、PTI5 或 PTI6。

## 第 4 章 存储器映像

### 4.1 简介

本设备包含多种存储器和内存映射外设，并且都在 4GB 的存储空间之内。本章介绍该存储器空间中存储器和外设的位置。

### 4.2 系统存储器映像

下表显示的是器件存储器映像概要。

表 4-1. 系统存储器映像

系统 32 位地址范围	目标从机	访问
0x0000_0000–0x07FF_FFFF <sup>1</sup>	程序 Flash 和只读数据 (包括前 196 字节的异常向量)	所有主机
0x0800_0000–0x0FFF_FFFF	保留	—
0x1000_0000–0x1FFF_EFFF	保留	—
0x1FFF_F000–0x1FFF_FFFF <sup>2</sup>	SRAM_L: 低位 SRAM	所有主机
0x2000_0000–0x2000_2FFF	SRAM_U: 高位 SRAM (位带区)	所有主机
0x2000_3000–0x21FF_FFFF	保留	—
0x2200_0000–0x2205_FFFF	SRAM_U 位带区别名	Cortex-M0+内核
0x2206_0000–0x23FF_FFFF	保留	—
0x2400_0000–0x3FFF_FFFF	对 SRAM_U 的位操作引擎(BME)访问	Cortex-M0+内核
0x4000_0000–0x4007_FFFF	AIPS 外设	Cortex-M0+ 内核
0x4008_0000–0x400F_EFFF	保留	—
0x400F_F000–0x400F_FFFF	通用输入/输出(GPIO)	Cortex-M0+ 内核
0x4010_0000–0x43FF_FFFF	保留	—
0x4400_0000–0x5FFF_FFFF	对 AIPS 外设的位操作引擎(BME)访问，用于插槽 0–127 <sup>3</sup>	Cortex-M0+内核
0x6000_0000–0xDFFF_FFFF	保留	—
0xE000_0000–0xE00F_FFFF	专用外围设备	Cortex-M0+内核
0xE010_0000–0xFFFF_FFFF	保留	—

下一页继续介绍此表...

表 4-1. 系统存储器映像 (继续)

系统 32 位地址范围	目标从机	访问
0xF000_0000–0xF000_0FFF	保留	-
0xF000_1000–0xF000_1FFF	保留	-
0xF000_2000–0xF000_2FFF	系统 ROM 表 <sup>4</sup>	Cortex-M0+内核
0xF000_3000–0xF000_3FFF	其他控制模块(MCM)	Cortex-M0+内核
0xF000_4000–0xF7FF_FFFF	保留	-
0xF800_0000–0xFFFF_FFFF	IOPORT: GPIO (单周期)	Cortex-M0+内核

- 程序 Flash 开始于 0x0000\_0000，但 Flash 的结束位置则不尽相同，具体取决于特定器件所配备的 Flash 的大小。详情请参见 [Flash 存储器大小](#)。
- 该范围不尽相同，具体取决于 SRAM 大小。详情请参见 [SRAM 大小](#)。
- 包括插槽 15 处 GPIO ( 基地址为 0x4000\_F000 ) 的 BME 运算。
- 该器件集成系统 ROM 表，用于在 CoreSight 调试系统中重定位至 ARM Cortex M0+ (Flycatcher) ROM 表。详情请参见 [系统 ROM 存储器映像](#)。

## 4.3 位带区别名

该器件支持 Cortex M0+内核的 SRAM\_U 位带区别名。在位带区别名中进行的 32 位写操作的结果与在位带区中目标位上进行的读取-修改-写入操作的结果相同，但只需一个周期的时间。位带区别名中的位操作更为高效。

写入位带区别名的位 0 数值确定哪个数值写入目标位：

- 位 0 置位时，向目标位写入 1。
- 位 0 清零时，向目标位写入 0。

位带区别名中进行的 32 位读操作返回以下任意一种数值：

- 数值为 0x0000\_0000 表示目标位清零
- 数值为 0x0000\_0001 表示目标位置位

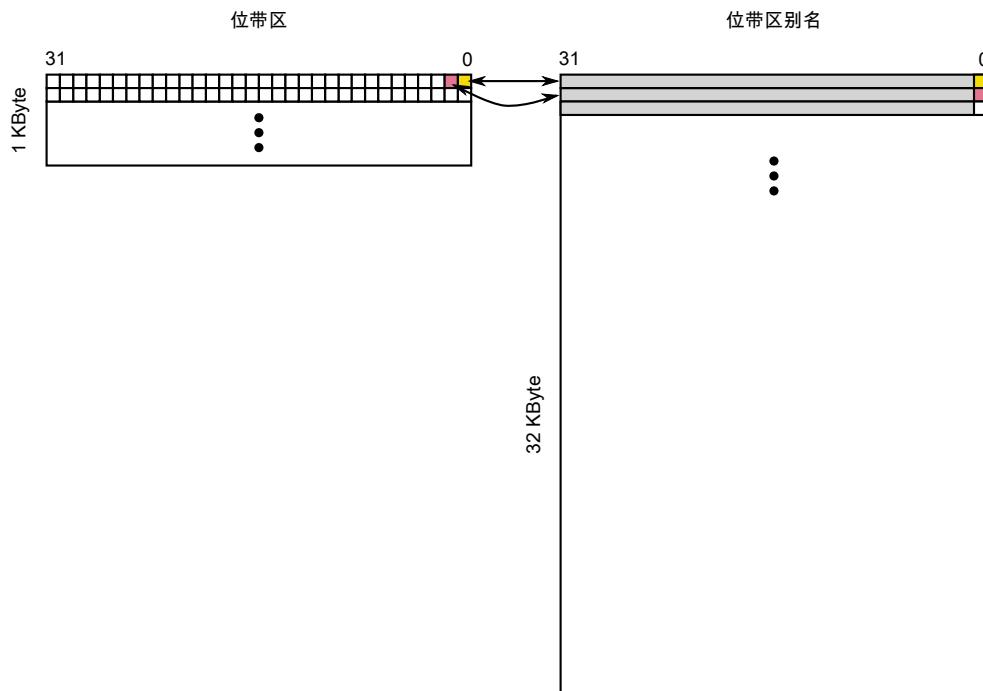


图 4-1. 位带区别名映射

## 4.4 位操作引擎

位处理引擎(BME)针对外设和 SRAM\_U 地址空间的原子“读取-修改-写入”存储器操作提供硬件支持。通过将 Cortex-M 指令集架构中的基本加载和存储指令支持与 BME 提供的修饰存储的概念相结合，为此类超低端微控制器提供了强大和高效的“读取-修改-写入”功能。有关其功能的详细描述，参见[位处理引擎\(BME\)](#)。

## 4.5 系统 ROM 存储器映像

通过 ARM CoreSight 调试基础架构来识别芯片上的组件，系统 ROM 表只是可选项目。

对于内核配置（比如 Cortex-M0+所支持的那些），ARM 建议调试器通过 CoreSight 调试基础架构识别并连接至调试元件。

ARM 建议调试器遵循下图所示流程来识别 CoreSight 调试基础架构中的组件。本例中，调试器为 CoreSight 系统中的每个 CoreSight 组件读取外设和组件 ID 寄存器。

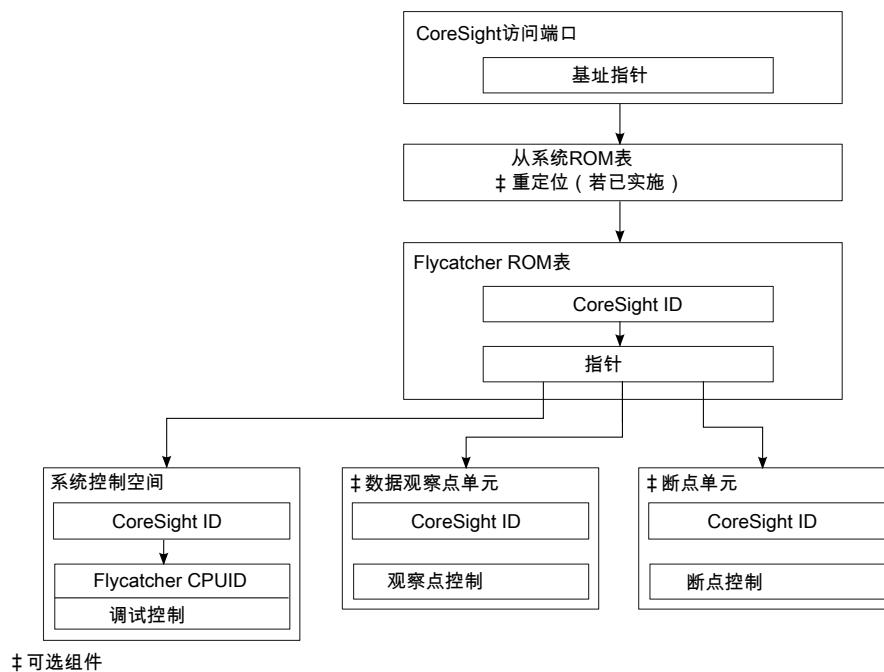


图 4-2. CoreSight 发现过程

下表显示的是 Freescale 系统 ROM 表存储器映像。它包括 ARM CoreSight 调试基础架构所需的 ROM 入口、外设 ID 和组件 ID。

### 注

该器件仅集成 Flycatcher ROM 表中定义的标准 ARM M0+内核调试组件。未集成自定义调试组件。

### ROM 存储器映射

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
F000_2000	入口 (ROM_ENTRY0)	32	R	参见章节	4.5.1/97
F000_2004	表格标记寄存器结束 (ROM_TABLEMARK)	32	R	0000_0000h	4.5.2/97
F000_2FCC	系统访问寄存器 (ROM_SYSACCESS)	32	R	0000_0001h	4.5.3/98
F000_2FD0	外设 ID 寄存器 (ROM_PERIPHID4)	32	R	参见章节	4.5.4/98
F000_2FD4	外设 ID 寄存器 (ROM_PERIPHID5)	32	R	参见章节	4.5.4/98
F000_2FD8	外设 ID 寄存器 (ROM_PERIPHID6)	32	R	参见章节	4.5.4/98
F000_2FDC	外设 ID 寄存器 (ROM_PERIPHID7)	32	R	参见章节	4.5.4/98
F000_2FE0	外设 ID 寄存器 (ROM_PERIPHID0)	32	R	参见章节	4.5.4/98
F000_2FE4	外设 ID 寄存器 (ROM_PERIPHID1)	32	R	参见章节	4.5.4/98
F000_2FE8	外设 ID 寄存器 (ROM_PERIPHID2)	32	R	参见章节	4.5.4/98
F000_2FEC	外设 ID 寄存器 (ROM_PERIPHID3)	32	R	参见章节	4.5.4/98
F000_2FF0	组件 ID 寄存器 (ROM_COMPID0)	32	R	参见章节	4.5.5/99
F000_2FF4	组件 ID 寄存器 (ROM_COMPID1)	32	R	参见章节	4.5.5/99

下一页继续介绍此表...

## ROM 存储器映射 (继续)

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
F000_2FF8	组件 ID 寄存器 (ROM_COMPID2)	32	R	参见章节	<a href="#">4.5.5/99</a>
F000_2FFC	组件 ID 寄存器 (ROM_COMPID3)	32	R	参见章节	<a href="#">4.5.5/99</a>

### 4.5.1 入口 (ROM\_ENTRYn)

系统 ROM 表始于“n”个相关的 32 位地址，器件中的每个调试组件均对应一个地址，并以一个全零值结束以表示 ROM 表在第“n+1”个值处终止。

它硬连线到特定值。外部调试代理在自动发现过程中会使用到这个值。

Address: F000\_2000h base + 0h offset + (4d × i), where i=0d to 0d

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
	ENTRY																															
W																																

复位 x\* | x\* x\*

\* 注:

- See field descriptions for reset values.x = 复位时未定义。

### ROM\_ENTRYn 字段描述

字段	描述
ENTRY	入口 入口 0 ( CM0+ ROM 表 ) 被硬连接到 0xF00F_D003。

### 4.5.2 表格标记寄存器结束 (ROM\_TABLEMARK)

该寄存器指示表格标记结束。它硬连线到特定值。外部调试代理在自动发现过程中会使用到这个值。

地址: F000\_2000h 基准 + 4h 偏移 = F000\_2004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
	MARK																															
W																																

### ROM\_TABLEMARK 字段描述

字段	描述
MARK	硬连线至 0x0000_0000

### 4.5.3 系统访问寄存器 (ROM\_SYSACCESS)

该寄存器指示系统访问。它硬连线到特定值。外部调试代理在自动发现过程中会使用到这个值。

地址: F000\_2000h 基准 + FCCh 偏移 = F000\_2FCCh

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	SYSACCESS															
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	

#### ROM\_SYSACCESS 字段描述

字段	描述
SYSACCESS	硬连线至 0x0000_0001

### 4.5.4 外设 ID 寄存器 (ROM\_PERIPHIDn)

这些寄存器指示外设 ID。它硬连线到特定值。外部调试代理在自动发现过程中会使用到这个值。

Address: F000\_2000h base + FD0h offset + (4d × i), where i=0d to 7d

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	PERIPHID															
W																																
复位	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*																	

\* 注:

- See field descriptions for reset values.x = 复位时未定义。

#### ROM\_PERIPHIDn 字段描述

字段	描述
PERIPHID	外设 ID1 硬连线至 0x0000_00E0 ; ID2 硬连线至 0x0000_0008 ; 其他硬连线至 0x0000_0000。

#### 4.5.5 组件 ID 寄存器 (ROM\_COMPIDn)

这些寄存器指示组件 ID。它硬连线到特定值。外部调试代理在自动发现过程中会使用到这个值。

Address: F000 2000h base + FF0h offset + (4d × i), where i=0d to 3d

\* 注・

- See field descriptions for reset values. $x$  = 复位时未定义。

## ROM COMPID<sub>n</sub> 字段描述

字段	描述
COMPID	组件 ID 组件 ID0 硬连线至 0x0000_000D ; ID1 硬连线至 0x0000_0010 ; ID2 硬连线至 0x0000_0005 ; ID3 硬连线至 0x0000_00B1。

## 4.6 外设桥(AIPS-Lite)存储器映像

外设桥存储器映像可通过 0x4000\_0000–0x400F\_FFFF 区域中交叉开关上的某个从机端口访问。该器件集成了一个可定义 1024 KB 地址空间的外设桥。

与此空间有关的三个区域为：

- 一个 128 KB 区域，分区为 32 个空间，每个空间大小为 4 KB，保留供平台上的外设使用。AIPS 控制器生成供所有 32 个空间使用的独特模块。
  - 一个 384 KB 区域，分区为 96 个空间，每个空间大小为 4 KB，保留供平台外的外设模块使用。AIPS 控制器为所有 96 个空间生成唯一的模块启用。
  - 最后一个插槽是始于 0x400F\_F000 的 4 KB 区域，用于访问 GPIO 模块。GPIO 插槽（插槽 128）是插槽 15 的别名。此块还直接与内核相连，无需经过与访问相关的等待状态即可通过 AIPS 控制器直接访问。

如果通过对应的 SIM 寄存器时钟选通控制位禁用了某模块，那么该模块的 AIPS 插槽也将被禁用。访问任何未实施或禁用的外设桥插槽中的地址都会导致传输错误终止。

通过外设桥进行编程模型访问，在 4 KB 插槽中通常实现的只占很小一部分。访问未在外设中实现的地址会导致传输错误终止。

## 4.6.1 先写后读序列和存储器操作所需的串行化

在某些情况下，执行后续操作前外设先前的写入操作必须全部完成。这类情况示例有：

- 退出中断服务程序(ISR)
- 更改模式
- 配置函数

在这类情况下，应用软件必须执行先写后读序列，以确保正确执行存储器操作所需的串行化：

1. 对外设寄存器进行写操作。
2. 对已进行写操作的外设寄存器进行读操作，以便验证写操作。
3. 继续后续操作。

## 4.6.2 外设桥(AIPS-Lite)存储器映像

### 注

- 插槽 0-95 和插槽 128 是 32 位数据宽的模块，例外情况是：插槽 49、82 是 8 位数据宽的模块 (IRQ、WDOG)，插槽 36 是 16 位数据宽的模块(MSCAN)。
- 插槽 96-127 是 8 位数据宽的模块。而插槽 121 和 122 为 32 位数据宽的模块 (KBI0、KBI1)

表 4-2. 外设桥 0 插槽分配

系统 32 位基地址	插槽编号	模块
0x4000_0000	0	—
0x4000_1000	1	—
0x4000_2000	2	—
0x4000_3000	3	—
0x4000_4000	4	—
0x4000_5000	5	—
0x4000_6000	6	—
0x4000_7000	7	—
0x4000_8000	8	—
0x4000_9000	9	—
0x4000_A000	10	—
0x4000_B000	11	—
0x4000_C000	12	—
0x4000_D000	13	—

下一页继续介绍此表...

表 4-2. 外设桥 0 插槽分配 (继续)

系统 32 位基地址	插槽编号	模块
0x4000_E000	14	—
0x4000_F000	15	GPIO 控制器 (别名地址 0x400F_F000)
0x4001_0000	16	—
0x4001_1000	17	—
0x4001_2000	18	—
0x4001_3000	19	—
0x4001_4000	20	—
0x4001_5000	21	—
0x4001_6000	22	—
0x4001_7000	23	—
0x4001_8000	24	—
0x4001_9000	25	—
0x4001_A000	26	—
0x4001_B000	27	—
0x4001_C000	28	—
0x4001_D000	29	—
0x4001_E000	30	—
0x4001_F000	31	—
0x4002_0000	32	Flash 存储器(FTMRE)
0x4002_1000	33	—
0x4002_2000	34	—
0x4002_3000	35	—
0x4002_4000	36	MSCAN
0x4002_5000	37	—
0x4002_6000	38	—
0x4002_7000	39	—
0x4002_8000	40	—
0x4002_9000	41	—
0x4002_A000	42	—
0x4002_B000	43	—
0x4002_C000	44	—
0x4002_D000	45	—
0x4002_E000	46	—
0x4002_F000	47	—
0x4003_0000	48	—
0x4003_1000	49	IRQ 控制器(IRQ)
0x4003_2000	50	循环冗余校验(CRC)
0x4003_3000	51	脉宽定时器(PWT)
0x4003_4000	52	—

下一页继续介绍此表...

表 4-2. 外设桥 0 插槽分配 (继续)

系统 32 位基地址	插槽编号	模块
0x4003_5000	53	—
0x4003_6000	54	—
0x4003_7000	55	周期性中断定时器(PIT)
0x4003_8000	56	Flex 定时器 0 (FTM0)
0x4003_9000	57	Flex 定时器 1 (FTM1)
0x4003_A000	58	Flex 定时器 2 (FTM2)
0x4003_B000	59	模数转换器(ADC)
0x4003_C000	60	—
0x4003_D000	61	实时时钟(RTC)
0x4003_E000	62	—
0x4003_F000	63	—
0x4004_0000	64	—
0x4004_1000	65	—
0x4004_2000	66	—
0x4004_3000	67	—
0x4004_4000	68	—
0x4004_5000	69	—
0x4004_6000	70	—
0x4004_7000	71	—
0x4004_8000	72	系统集成模块(SIM)
0x4004_9000	73	端口控制器
0x4004_A000	74	—
0x4004_B000	75	—
0x4004_C000	76	—
0x4004_D000	77	—
0x4004_E000	78	—
0x4004_F000	79	—
0x4005_0000	80	—
0x4005_1000	81	—
0x4005_2000	82	看门狗(WDOG)
0x4005_3000	83	—
0x4005_4000	84	—
0x4005_5000	85	—
0x4005_6000	86	—
0x4005_7000	87	—
0x4005_8000	88	—
0x4005_9000	89	—
0x4005_A000	90	—
0x4005_B000	91	—

下一页继续介绍此表...

表 4-2. 外设桥 0 插槽分配 (继续)

系统 32 位基地址	插槽编号	模块
0x4005_C000	92	—
0x4005_D000	93	—
0x4005_E000	94	—
0x4005_F000	95	—
0x4006_0000	96	—
0x4006_1000	97	—
0x4006_2000	98	—
0x4006_3000	99	—
0x4006_4000	100	内部时钟源(ICS)
0x4006_5000	101	系统振荡器(OSC)
0x4006_6000	102	I <sup>2</sup> C0
0x4006_7000	103	I <sup>2</sup> C1
0x4006_8000	104	—
0x4006_9000	105	—
0x4006_A000	106	通用异步收发器 0 (UART0)
0x4006_B000	107	通用异步收发器 1 (UART1)
0x4006_C000	108	通用异步收发器 2 (UART2)
0x4006_D000	109	—
0x4006_E000	110	—
0x4006_F000	111	—
0x4007_0000	112	—
0x4007_1000	113	—
0x4007_2000	114	—
0x4007_3000	115	模拟比较器 0 (ACMP0)
0x4007_4000	116	模拟比较器 1 (ACMP1)
0x4007_5000	117	—
0x4007_6000	118	串行外设接口 0 (SPI0)
0x4007_7000	119	串行外设接口 1 (SPI1)
0x4007_8000	120	—
0x4007_9000	121	键盘中断 0 (KBI0)
0x4007_A000	122	键盘中断 1 (KBI1)
0x4007_B000	123	—
0x4007_C000	124	—
0x4007_D000	125	电源管理控制器(PMC)
0x4007_E000	126	—
0x4007_F000	127	—
0x400F_F000	128	GPIO 控制器

## 4.7 专用外设总线(PPB)存储器映像

PPB 是已定义 ARM 总线架构的一部分，提供对选择处理器-本地模块的访问。这些资源仅可从内核访问；其他系统主机无法访问。

表 4-3. PPB 存储器映像

系统 32 位地址范围	资源	更多范围详情	资源
0xE000_0000–0xE000_DFFF	保留位		
0xE000_E000–0xE000_EFFF	系统控制空间(SCS)	0xE000_E000–0xE000_E00F	保留位
		0xE000_E010–0xE000_E0FF	SysTick
		0xE000_E100–0xE000_ECFF	NVIC
		0xE000_ED00–0xE000_ED8F	系统控制数据块
		0xE000_ED90–0xE000_EDEF	保留位
		0xE000_EDF0–0xE000_EEFF	调试
		0xE000_EF00–0xE000_EFFF	保留
0xE000_F000–0xE00F_EFFF	保留		
0xE00F_F000–0xE00F_FFFF	内核 ROM 空间(CRS)		

# 第 5 章 时钟分布

## 5.1 简介

本章介绍该器件的时钟架构以及时钟概述，并提供术语部分。

Cortex M0+位于同步内核平台上，可对处理器和总线主机、Flash 和外设时钟进行独立配置。

ICS 模块被用于生成主时钟。由 ICS 模块来选择某个时钟源（内部基准、外部晶体或外部时钟信号）产生系统时钟源。

## 5.2 编程模型

系统时钟源的选择和多路复用通过 [ICS 模块](#) 控制和配置。该系统的时钟分频器设置和模块时钟选通通过 [SIM 模块](#) 配置。有关寄存器和位的详细说明，请参见相关部分。

## 5.3 器件时钟连接示意图

该器件包含下列片上时钟源：

- 内部时钟源(ICS)模块：向外设提供总线时钟和其他基准时钟的主时钟源发生器
- 系统振荡器(OSC)模块：向内部时钟源(ICS)、实时时钟计数器时钟模块(RTC)和其他 MCU 子系统提供基准时钟的系统振荡器
- 低功耗振荡器(LPO)模块：向 RTC 和 WDOG 提供 1 kHz 基准时钟的片上低功耗振荡器

[图 5-1](#) 显示的是来自 ICS 模块和 OSC 模块的时钟如何分配给微控制器的其他功能单元。微控制器中的某些模块具有可选时钟输入。

下列系统振荡器、ICS 和 SIM 模块的寄存器控制多路复用器、分频器和时钟脉冲门，如图所示：

表 5-1. 控制多路复用器、分频器和时钟脉冲门的寄存器

	OSC	ICS	SIM
复用器	OSC_CR	ICS_C1	SIM_SOPT
分频器	—	ICS_C2	SIM_CLKDIV
时钟门控	OSC_CR	ICS_C1	SIM_SCGC

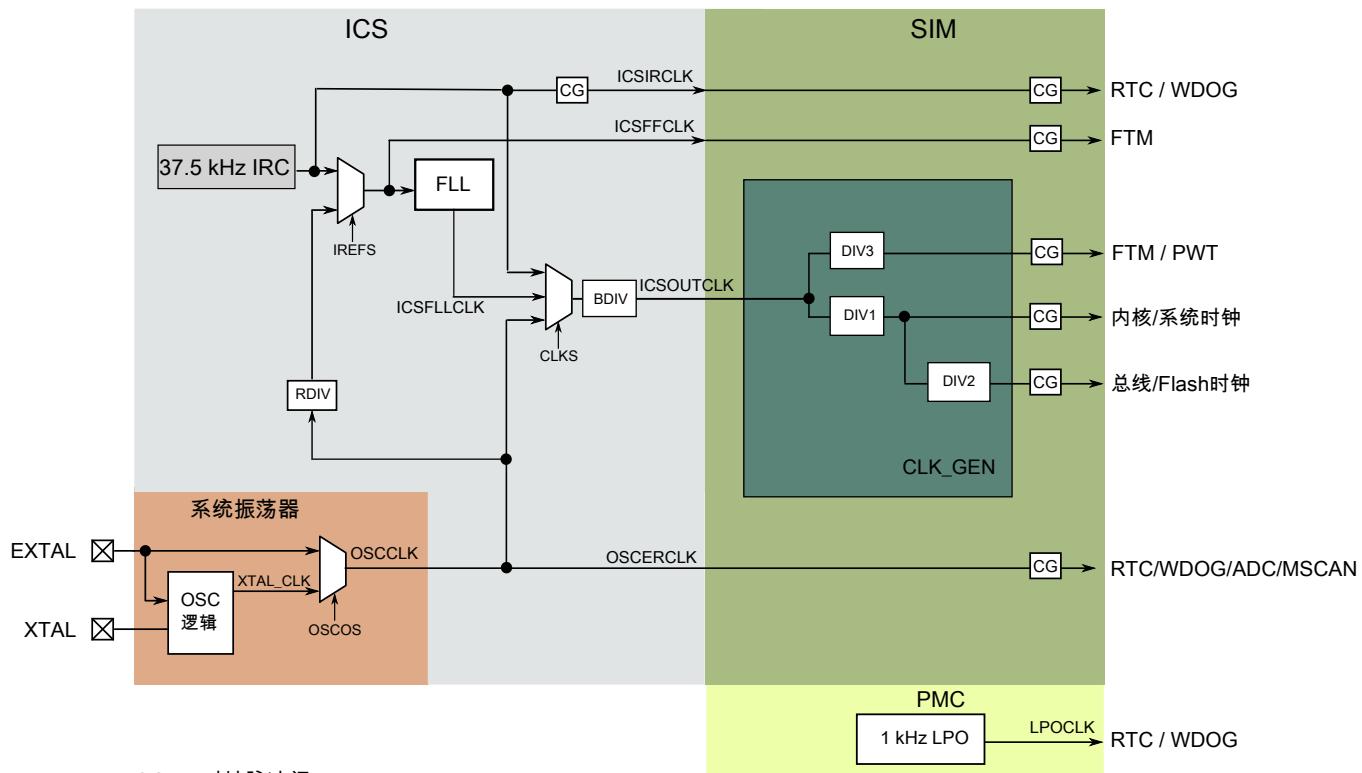


图 5-1. 时钟示意图

## 5.4 时钟定义

下表说明的是图 5-1 中的时钟。

表 5-2. 时钟定义

时钟名称	说明
内核时钟	由 DIV1 分频的 ICSOUTCLK，为 ARM Cortex-M0+ 内核提供时钟。它是 CPU HCLK
平台时钟	由 DIV1 分频的 ICSOUTCLK，为交叉开关和 NVIC 提供时钟。它是自由运行的 FCLK

下一页继续介绍此表...

表 5-2. 时钟定义 (继续)

时钟名称	说明
系统时钟	由 DIV1 分频的 ICSOUTCLK, 直接为总线主机提供时钟
总线时钟	由 DIV2 分频的系统时钟, 为总线从机和外设提供时钟
Flash 时钟	由 DIV2 分频的系统时钟, 为 Flash 存储器模块提供时钟。它与该器件中的总线时钟相同
定时器时钟	由 DIV3 分频的 ICSOUTCLK, 为 FTM 和 PWT 模块提供时钟
调试时钟	调试逻辑时钟。就该器件而言, 它来源于平台时钟
SWD 时钟	DAP 接口时钟。SWD 时钟通常由外部调试器驱动, 且完全与内核时钟和平台时钟异步
ICSIRCLK	内部 32 kHz IRC 基准时钟的 ICS 输出。ICSIRCLK 可作为 RTC 或 WDOG 模块的时钟源
ICSOOUTCLK	IRC、ICSFLLCLK 或 ICS 外部基准时钟的 ICS 输出, 是内核、系统、总线和 Flash 时钟的时钟源
ICSFLLCLK	FLL 的输出, FLL 将频率锁定为 1280 乘以内部或外部基准频率
ICSFCLK	固定频率时钟的 ICS 输出。ICSFCLK 可作为 FTM 模块的时钟源。ICSFCLK 的频率由 ICS 的设置决定
OSCCLK	内部振荡器的系统振荡器输出, 或由 EXTAL 直接提供时钟源。用作 ICS 外部基准时钟
OSCERCLK	由 OSCCLK 提供时钟源的系统振荡器输出, 可作为 MSCAN <sup>1</sup> 、RTC、WDOG 或 ADC 模块的时钟源
LPOCLK	PMC 1 kHz 输出, LPOCLK 可作为 RTC 或 WDOG 模块的时钟源

1. 当 OSCERCLK 用作 MSCAN 时钟时, 其频率不得超过 24 MHz

## 5.4.1 器件时钟汇总

下表提供有关片上时钟的更多信息。

表 5-3. 时钟汇总

时钟名称	Run 模式频率	时钟源	时钟禁用条件
内核时钟	最高 48 MHz	ICSOOUTCLK 时钟分频器	在 Wait 和 Stop 模式下
平台时钟	最高 48 MHz	ICSOOUTCLK 时钟分频器	在 Stop 模式下
系统时钟	最高 48 MHz	ICSOOUTCLK 时钟分频器	在 Stop 模式下
定时器时钟	最高 48 MHz	ICSOOUTCLK 时钟分频器	在 Stop 模式下
总线时钟	最高 24 MHz	ICSOOUTCLK 时钟分频器	在 Stop 模式下
调试时钟	最高 24 MHz	来自平台时钟	调试未使能
SWD 时钟	高达 24 MHz	SWD_CLK 引脚	外部时钟的输入, 因此不会禁用。
Flash 时钟	最高 24 MHz	ICSOOUTCLK 时钟分频器	在 Stop 模式下
内部基准时钟 (ICSIRCLK)	31.25–39.0625 kHz IRC	IRC	ICS_C1[IRCLKEN]=0, 或

下一页继续介绍此表...

表 5-3. 时钟汇总 (继续)

时钟名称	Run 模式频率	时钟源	时钟禁用条件
			在 Stop 模式下且 ICS_C1[IREFSTEN]=0
外部基准时钟 (OSCERCLK)	DC 最高达到 48 MHz (旁路), 31.25–39.0625 kHz 或 4–24 MHz (晶体)	系统 OSC	OSC_CR[OSCEN]=0, 或 在 Stop 模式下且 OSC_CR[OSCSTEN]=0
FLL 外部时钟 (ICSPLLCLK)	40–50 MHz	系统 OSC 或 IRC	在 Stop 模式下, 或 FLL 未使能
ICS 固定频率时钟 (ICSFFCLK)	31.25–39.0625 kHz	系统 OSC 或 IRC	在 Stop 模式下
LPOCLK	1 kHz	PMC	在所有电源模式下都可用

## 5.4.2 时钟分布

下图显示的是时钟分布简化示意图

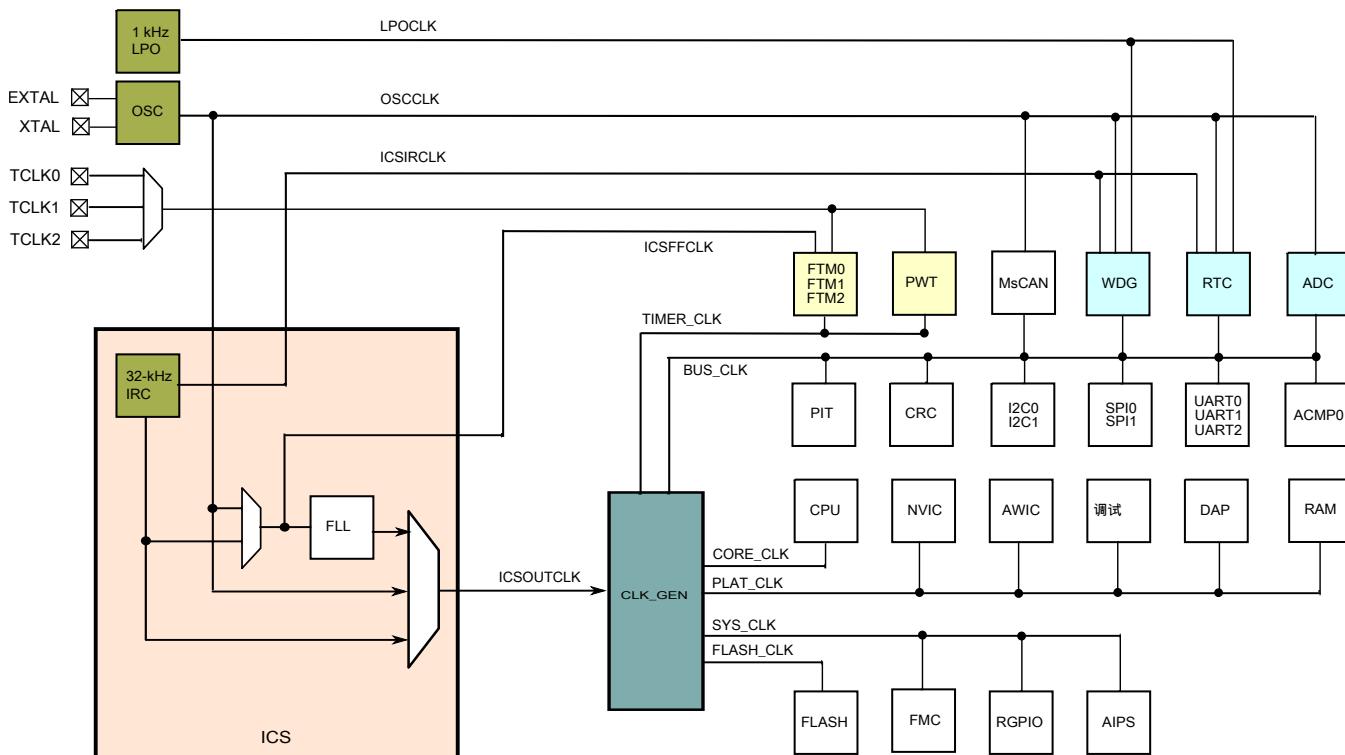


图 5-2. 时钟分布概要示意图

**注**

此时钟分布图中未显示时钟分频和门控。

## 5.5 内部时钟源

该器件的内部时钟源如下：

- 片上 RC 振荡器，范围为 31.25–39.0625 kHz，作为 FLL 输入的基准。
- 片上内部 1 kHz 振荡器，作为 RTC 和 WDOG 的低频低功耗源，符合特定用例要求。

下表显示的是该器件的频率可用性：

**表 5-4. 基于内部基准的可用 ICS 总线频率**

基准		ICSOUTCLK
FEI (高范围)	BDIV = 0	40 MHz ~ 50 MHz <sup>1</sup>
	BDIV = 1	20 MHz ~ 25 MHz
	BDIV = 2	10 MHz ~ 12.5 MHz
	BDIV = 4	5 MHz ~ 6.25 MHz
	BDIV = 8	2.5 MHz ~ 3.125 MHz
	BDIV = 16	1.25 MHz ~ 1.5625 MHz
	BDIV = 32	625 kHz ~ 781.25 kHz
	BDIV = 64	312.5 kHz ~ 390.625 kHz
	BDIV = 128	156.25 kHz ~ 195.3125 kHz

- 请仔细配置 SIM\_CLKDIV 和 BDIV，确保任何时钟频率均不得高于 48 MHz。

## 5.6 外部时钟源

该器件支持下列两个外部时钟源：

- 外部方波输入时钟，DC 最高达到 48 MHz。
- 外部晶振或谐振器：
  - 小范围：31.25–39.0625 kHz
  - 大范围：4–24 MHz

**注**

外部方波输入时钟仅在 OSC 模块设置为外部时钟模式下使用（旁路）。凭借外部方波时钟源，用户可使用 FLL 被禁用的 FBE 模式来实现更低功耗或精密时钟源。

下表显示的是该器件采用 OSC 时钟作为时钟源时的可用频率。OSC 外部时钟模式未列出。

表 5-5. 可用的 OSC 频率

ICS 配置	外部基准	RDIV
FBE	31.25 kHz ~ 39.0625 kHz	-
	4 MHz ~ 24 MHz	-
FEE <sup>1</sup>	31.25 kHz ~ 39.0625 kHz	RDIV = 1
	62.5 kHz ~ 78.125 kHz	RDIV = 2
	125 kHz ~ 56.25 kHz	RDIV = 4
	250 kHz ~ 312.5 kHz	RDIV = 8
	500 kHz ~ 625 kHz	RDIV = 16
	1 MHz ~ 1.25 MHz	RDIV = 32
	2 MHz ~ 2.5 MHz	RDIV = 64
	4 MHz ~ 5 MHz	RDIV = 128
	8 MHz ~ 10 MHz	RDIV = 256
	16 MHz ~ 20 MHz	RDIV = 512

- 在 FEE 模式下，FLL 输出频率 = OSC/RDIV \*1280。仔细选择 OSC 和 RDIV，确保将 FLL 输出频率保持在限定内。

## 5.7 时钟选通

每个模块的时钟都可通过配置系统时钟选通控制寄存器 (SIM\_SCGC) 单独地实现开/关的选通。进行模块初始化之前，配置系统时钟选通控制寄存器 (SIM\_SCGC) 中相应位以使能时钟。关闭时钟前，务必禁用该模块。

当外设的时钟被关闭的时候，任何通过总线对这个外设进行的访问都会产生错误而终止。

## 5.8 模块时钟

下表汇总与各模块相关的时钟。

表 5-6. 模块时钟

模块	总线接口时钟	内部时钟	I/O 接口时钟
内核模块			
ARM Cortex-M0+内核	平台时钟	内核时钟	-
NVIC	平台时钟	-	-
DAP	平台时钟	-	SWD_CLK
系统模块			

下一页继续介绍此表...

表 5-6. 模块时钟 (继续)

模块	总线接口时钟	内部时钟	I/O 接口时钟
端口控制	总线时钟	—	—
交叉开关	平台时钟	—	—
外设桥	系统时钟	总线时钟	—
PMC、SIM	总线时钟	LPOCLK	—
MCM	平台时钟	—	—
CRC	总线时钟	—	—
WDOG 定时器	总线时钟	总线时钟 LPOCLK ICSIRCLK OSCERCLK	—
时钟			
ICS	总线时钟	ICSOOUTCLK ICSFLLCLK ICSIRCLK OSCERCLK	—
OSC	总线时钟	OSCERCLK	—
存储器和存储器接口			
Flash 控制器	系统时钟	—	—
Flash 存储器	Flash 时钟	—	—
SRAM	平台时钟	—	—
模拟			
ADC	总线时钟	总线时钟 OSCERCLK ADACK	—
ACMPO	总线时钟	—	—
ACMP1	总线时钟	—	—
定时器			
PIT	总线时钟	—	—
PWT	定时器时钟	—	TCLK0/1/2
FTM0	定时器时钟	定时器时钟 ICSFFCLK	TCLK0/1/2
FTM1	定时器时钟	定时器时钟 ICSFFCLK	TCLK0/1/2
FTM2	定时器时钟	定时器时钟 ICSFFCLK	TCLK0/1/2
RTC	总线时钟	总线时钟 LPOCLK ICSIRCLK OSCERCLK	RTC_CLKOUT

下一页继续介绍此表...

表 5-6. 模块时钟 (继续)

模块	总线接口时钟	内部时钟	I/O 接口时钟
通信接口			
SPI0	总线时钟	—	SPI0_SCK
SPI1	总线时钟	—	SPI1_SCK
I <sup>2</sup> C0	总线时钟	—	I <sup>2</sup> C0_SCL
I <sup>2</sup> C1	总线时钟	—	I <sup>2</sup> C1_SCL
UART0/SCI0	总线时钟	—	—
UART1/SCI1	总线时钟	—	—
UART2/SCI2	总线时钟	—	—
MSCAN	总线时钟	OSCERCLK <sup>1</sup>	—
人机接口			
GPIO	系统时钟	—	—
KBI0	总线时钟	—	—
KBI1	总线时钟	—	—

1. OSCERCLK 用作 MSCAN 时钟时，其频率不得高于 24 MHz

## 5.8.1 FTM 和 PWT 计时

FTM 模块和 PWT 模块的计数器具有可选时钟，如下图所示。TCLK0 TCLK1 和 TCLK2 是定时器的可选外部时钟输入。

### 注

如果选择 TCLK0、TCLK1 或 TCLK2 作为 FTM 或 PWT 的计数器时钟，则仍然需要使用片上定时器时钟 (TIMER\_CLK) 来同步计数器结果。而且片上定时器时钟 (TIMER\_CLK) 必须至少比来自 TCLK0、TCLK1 或 TCLK2 的外部时钟快 4 倍。

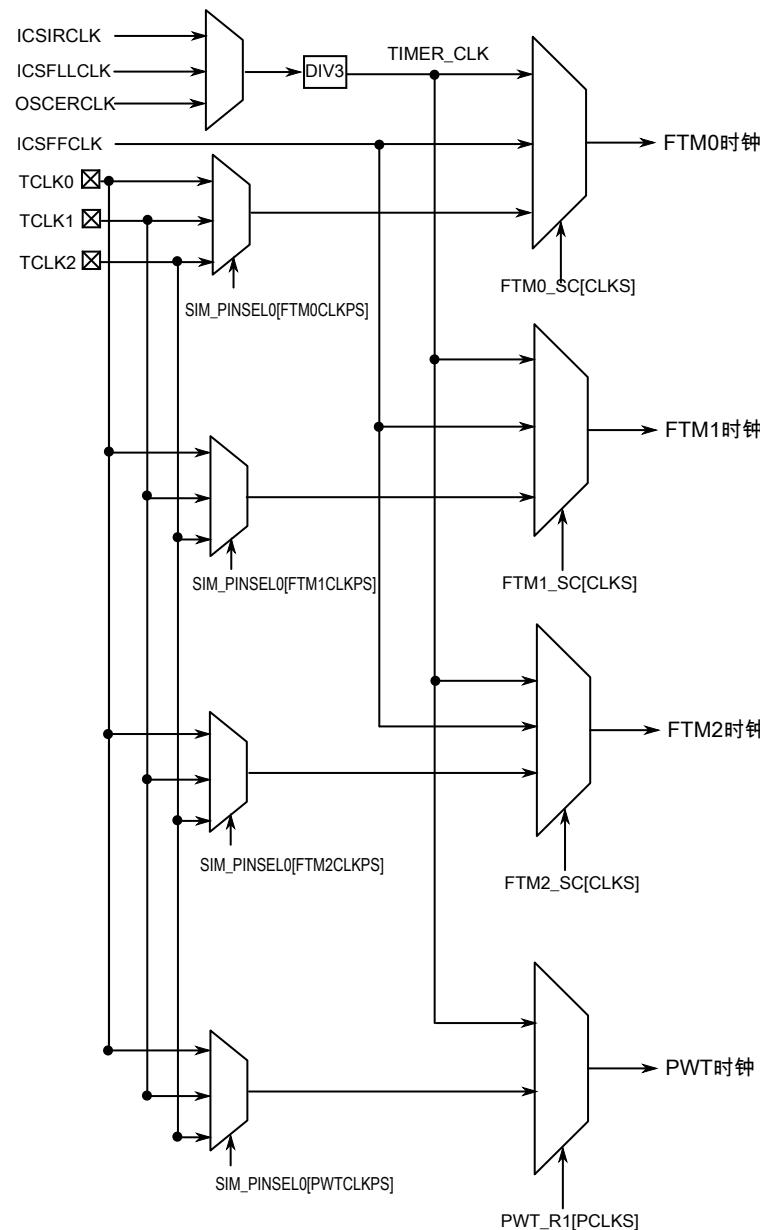


图 5-3. FTM 和 PWT 时钟生成



# 第 6 章 复位与引导

## 6.1 简介

该 MCU 支持下列复位源：

表 6-1. 复位源

复位源	说明
POR 复位	<ul style="list-style-type: none"> <li>上电复位(POR)</li> </ul>
系统复位	<ul style="list-style-type: none"> <li>外部引脚复位(PIN)</li> <li>低压检测(LVD)</li> <li>WDOG 定时器</li> <li>ICS 时钟丢失(LOC)复位</li> <li>Stop 模式应答错误(SACKERR)</li> <li>软件复位(SW)</li> <li>死锁复位(LOCKUP)</li> <li>MDM DAP 系统复位</li> </ul>

每个系统复位源在系统复位状态和 ID 寄存器 (SIM\_SRSID) 中都有相应的位。

MCU 在功能模式下可以退出或复位，此时 CPU 处于执行代码或者调试挂起的状态。MCU 有多种引导选项可进行配置。更多详情，请参见[引导](#)章节。

## 6.2 复位

本节讨论基本复位机制和复位源。某些能导致复位的模块可以通过配置引起中断来代替复位。更多信息，请参考独立的外设章节。

### 6.2.1 上电复位(POR)

MCU 初次上电或电源电压下降到低于上电复位电压电平( $V_{POR}$ )时，POR 电路将产生 POR 复位条件。

随着供电电压升高，LVD 电路将 MCU 保持在复位状态直至电源超过 LVD 最低阈值 ( $V_{LVDL}$ )。系统复位状态和 ID 寄存器 (SIM\_SRSID) 的 POR 和 LVD 字段 (SIM\_SRSID[POR] 和 SIM\_SRSID[LVD]) 在 POR 之后置位。

## 6.2.2 系统复位源

复位 MCU 提供一种从已知的初始条件开始处理的方法。系统复位开始时，片上稳压器处于完全稳压状态，系统时钟来源于内部基准时钟。当处理器退出复位时将执行以下操作：

- 从向量表偏移 0 处读取 SP(SP\_main)初始值
- 从向量表偏移 4 处读取程序计数器(PC)初始值
- 链接寄存器(LR)设置为 0xFFFF\_FFFF。

片上外设模块禁用且非模拟 I/O 引脚初始配置为禁用 (SWD\_DIO/SWD\_CLK、 $\overline{NMI}$  和  $\overline{RESET}$  引脚可根据[系统选项寄存器 0 \(SIM\\_SOPT0\)](#) 的设置在系统复位后使能的情况除外)。具有模拟功能的引脚在复位后默认为模拟功能。

### 6.2.2.1 外部引脚复位( $\overline{RESET}$ )

该引脚具有内部上拉电阻。 $\overline{RESET}$  可将器件从任意模式唤醒。

POR 复位后，PTA5 默认功能为  $\overline{RESET}$ 。必须对 SIM\_SOPT0[RSTPE]进行编程以使能其他功能。该字段清零后，此引脚可用作 PTA5 或其他替代功能。

#### 6.2.2.1.1 复位引脚滤波器

$\overline{RESET}/\overline{IRQ}$  引脚滤波器支持用 1 kHz LPO 时钟和总线时钟进行滤波。它可用作简单的低通滤波器，从而过滤由  $\overline{RESET}/\overline{IRQ}$  引脚产生的毛刺。

毛刺宽度阈值可根据端口滤波寄存器 0 (PORT\_IOFLT0) 来调节，设置[端口滤波寄存器 0 \(PORT\\_IOFLT0\)](#) 范围在 1~4096 BUSCLK (或 1~128 LPOCLK) 之间。该可配置去抖滤波器可替代板载外部模拟滤波器，并且能极大地提高 EMC 性能。设置[端口滤波寄存器 0 \(PORT\\_IOFLT0\)](#) 可配置所有端口的滤波器。

### 6.2.2.2 低压检测(LVD)

该器件集成了一个可避免低压条件的系统，以便在电源电压发生变化期间保护存储器内容和控制 MCU 系统状态。该系统由上电复位(POR)电路和 LVD 电路组成，LVD 具有用户可选的触发电压，可以是高电平( $V_{LVDH}$ )或低电平( $V_{LVDL}$ )。

PMC\_SPMSC1[LVDE]置位且 PMC\_SPMSC2[LVDV]选定触发电压后，LVD 电路使能。除非 PMC\_SPMSC1[LVDSE]置位或处于串行线调试(SWD)模式，否则 LVD 一进入 Stop 模式就会被禁用。若 PMC\_SPMSC1[LVDSE]和 PMC\_SPMSC1[LVDE]均置位，在 LVD 使能的情况下，Stop 模式下的电流消耗将更高。

### 6.2.2.3 WDOG 定时器

WDOG 通过定期的软件通讯来对系统操作进行监控。此通讯一般被称为处理(或刷新)WDOG。如果此周期性刷新没有发生，则 WDOG 将产生一个系统复位的事件。WDOG 复位会使 SIM\_SRSID[WDOG]置位。

### 6.2.2.4 ICS 时钟丢失(LOC)

该芯片上的 ICS 支持带复位功能的外部基准时钟监控器。

在 FBE 或 FEE 模式下，如将 1 写入 ICS\_C4[CME]，则时钟监控器使能。若外部基准频率下降到低于某个特定频率(比如： $f_{loc\_high}$  或  $f_{loc\_low}$ ，具体取决于 OSC\_CR[RANGE])，则 MCU 将复位。SIM\_SRSID[LOC]将置位以显示出错误。

在 FBILP 模式，FLL 不启用，因此即便将 1 写入 ICS\_C4[CME]，外部基准时钟监视器也不会运行。

外部基准时钟监控器将 FLL 用作内部基准时钟。FLL 必须在 ICS\_C4[CME]置位前能够工作。

### 6.2.2.5 Stop 模式应答错误(SACKERR)

在内核试图进入 Stop 模式时，在 1025 个 1kHz LPO 时钟周期内，仍然有外设模块没有对该模式进行响应，则会产生该复位。

如果发生错误条件，模块对进入 Stop 模式可能不会作出应答。该错误可能是由模块的外部时钟输入故障引起的。

### 6.2.2.6 软件复位(SW)

通过设置 NVIC 应用中断及复位控制寄存器中的 SYSRESETREQ 字段可以强制使设备产生软件复位。(有关寄存器字段的完整说明，尤其是 VECTKEY 字段的要求，请参见 ARM 的 NVIC 文档)。设置 SYSRESETREQ 可产生软件复位请求。软件复位对除调试模块以外的所有主要模块强制执行系统复位。

### 6.2.2.7 死锁复位(LOCKUP)

LOCKUP 可立即表明内核出现严重的软件错误。这是激活处理器内置系统状态保护硬件而产生的不可恢复的异常进而内核被锁定的结果。

LOCKUP 条件可导致系统复位，也可导致 SIM\_SRSID[LOCKUP]置位。

### 6.2.2.8 MDM-AP 系统复位请求

设置 MDM-AP 控制寄存器中的“系统复位请求”字段产生系统复位。这是通过 SWD 接口进行复位的主要方法。在清除此位前系统一直保持复位状态。

设置 MDM-AP 控制寄存器中的“内核保持复位”字段，将内核保持在复位状态，而系统剩余部分退出复位状态。

## 6.2.3 MCU 复位

MCU 产生各种复位来复位不同模块。

### 6.2.3.1 仅 POR

“仅 POR 复位”仅由 POR 复位源产生。它复位 PMC 和 RTC。

“仅 POR 复位”还会产生所有其他的复位类型。

### 6.2.3.2 芯片 POR

“芯片 POR”由 POR 和 LVD 复位源产生。它复位“复位引脚滤波器”寄存器和部分 SIM 和 ICS 的寄存器。

“芯片 POR”还能导致芯片复位（包括“早期芯片复位”）产生。

### 6.2.3.3 早期芯片复位

“早期芯片复位”在所有复位源上都产生。它仅复位 Flash 存储器模块和 ARM 平台。它在 Flash 存储器开始初始化之前撤除（“早于”芯片复位撤除）。

### 6.2.3.4 芯片复位

“芯片复位”在所有复位源上都产生，并且仅在  $\overline{\text{RESET}}$  引脚撤除后才撤除。它复位剩余的模块（未由其他复位类型复位的模块）。

## 6.3 引导

本小节介绍引导顺序，包括来源和选项。

引导过程自动加载某些配置信息，如存在工厂编程 Flash 位置中的时钟调整值。

### 6.3.1 引导源

CM0+内核加入对可配置向量表偏移寄存器(VTOR<sup>1</sup>)的支持，可重新定位中断向量表。此器件支持从内部闪存和 RAM 引导。

该器件支持从复位向量位于地址 0x0 (初始 SP\_main)、0x4 (初始 PC) 的内部Flash 引导，以及从中断向量表重定位至 RAM 的 RAM 引导。

### 6.3.2 引导序列

上电时，片上稳压器将系统保持在 POR 状态，直到输入电源电压高于 POR 阈值。系统持续处于该静态，直到内部稳压电源达到 LVD 所决定的安全操作电压。复位控制器逻辑随后执行以下序列以退出复位。

1. 系统复位遵循一定的内部逻辑。 $\overline{\text{RESET}}$  引脚被驱动至输出低电平（约 4.2  $\mu\text{s}$ ），并且 ICS 按默认配置使能。
2. 释放  $\overline{\text{RESET}}$  引脚。如果  $\overline{\text{RESET}}$  引脚的电平继续为有效值（表示  $\overline{\text{RESET}}$  引脚上升时间缓慢或外部驱动低电平），则系统继续保持在复位状态。一旦检测到  $\overline{\text{RESET}}$  引脚处于高电平，内核时钟使能且系统解除复位状态。
3. NVM 开始内部初始化。Flash 控制器解除复位状态并开始执行初始化操作，而内核在 Flash 初始化完成之前依然保持停止状态。
4. Flash 初始化完成(16  $\mu\text{s}$ )后，内核设置堆栈、程序计数器(PC)和链接寄存器(LR)。处理器从向量表偏移 0 读取 SP(SP\_main)初始值。内核从向量表偏移 4 读取 PC 初始值。LR 设为 0xFFFF\_FFFF。CPU 在 PC 位置开始执行。

后续系统复位遵循相同的复位流程。

---

1. VTOR: 指的是 ARMv6-M 架构参考手册中的向量表偏移寄存器。



# 第 7 章 电源管理

## 7.1 简介

本章介绍多种芯片电源模式以及各模块在这些模式下的功能。

## 7.2 功耗模式

电源管理控制器(PMC)为用户提供多种功耗选项。支持各种工作模式，来允许用户针对所需的功能等级优化功耗。

该器件支持 Run、Wait 和 Stop 三种模式，无论是不同的功耗水平还是功能要求，客户都能轻松使用。所有模式下都能保持 I/O 状态。

- Run 模式—CPU 时钟全速运行，内部电源处于全稳压状态。
- Wait 模式—关断 CPU 以降低功耗；此时系统时钟和总线时钟依然运行，保持完全稳压状态。
- Stop 模式—可选 LVD 使能，稳压器处于待机状态。

三种工作模式为：Run、Wait 和 Stop。WFI 指令激活芯片的 Wait 和 Stop 模式。

表 7-1. 芯片功率模式

功耗模式	说明	内核模式	一般恢复方法
正常运行	允许芯片工作在最高性能下。复位后的默认状态 片上稳压器开启。	运行	—
通过 WFI 指令实现 正常待机	允许外设工作，同时让内核处于睡眠模式，可降低功耗。NVIC 依然监测中断；继续提供外设时钟。	睡眠	中断
通过 WFI 指令实现 正常停止	将芯片置于静态。最低功耗模式保持全部寄存器值，并保留 LVD 保护功能（可选）。NVIC 禁用；使用 AWIC 从中断唤醒；外设时钟停止。	深度睡眠	中断

## 7.3 进入和退出低功耗模式

芯片通过 WFI 指令进入 Wait 和 Stop 模式。处理器通过中断退出低功耗模式。

### 注

WFE 指令可能会对进入低功耗模式产生副作用，但这并非其预期用途。有关 WFE 指令的更多信息，请参见 ARM 文档。

## 7.4 低功耗模式下的模块操作

下表说明了该芯片处于各低功耗模式时每个模块的功能。其标准特性如表所示，其中包括某些例外情况。

表 7-2. 低功耗模式下的模块操作

模块	运行	等待	停止
内核模块			
CPU	开启	待机	待机
NVIC	开启	开启	待机
系统模块			
PMC	完全稳压	完全稳压	宽松稳压
WDOG	开启	开启	可选开启
LVD	开启	开启	可选开启
CRC	开启	开启	待机
时钟			
ICS	开启	开启	可选开启
OSC	开启	开启	可选开启
LPO	开启	开启	始终开启
存储器			
Flash	开启	开启	待机
RAM	开启	待机 <sup>1</sup>	待机
定时器			
FTM	开启	开启	待机
PIT	开启	开启	待机
PWT	开启	开启	待机
RTC	开启	开启	可选开启
模拟			
ADC	开启	开启	可选开启
ACMP	开启	开启	可选开启

下一页继续介绍此表...

表 7-2. 低功耗模式下的模块操作 (继续)

模块	运行	等待	停止
通信接口			
UART	开启	开启	待机 <sup>2</sup>
SPI	开启	开启	待机 <sup>3</sup>
IIC	开启	开启	待机 <sup>4</sup>
MSCAN	开启	开启	待机 <sup>2</sup>
人机接口			
KBI	开启	开启	待机 <sup>5</sup>
IRQ	开启	开启	待机 <sup>5</sup>
I/O	开启	开启	状态保持

- SRAM 使能信号保持在低电平时，禁用内部时钟信号并屏蔽地址和数据输入；芯片中的 RAM 时钟可在 Wait 模式下处于激活状态。
- 支持 Stop 模式下的边沿唤醒
- 支持 Stop 模式下的从机模式接收和唤醒
- 支持 Stop 模式下的地址匹配唤醒
- 支持 Stop 模式下的引脚中断唤醒



## 第 8 章 加密

### 8.1 简介

本器件基于从 Flash 模块中选择的模式实施加密。下面的章节介绍 Flash 加密概述和非 Flash 模块上加密影响的详细信息。

### 8.2 Flash 加密

Flash 模块根据 FTMRE\_FSEC[SEC] 保持的状态为 MCU 提供加密信息。而 MCU 确认加密请求，并限制对 Flash 资源的访问。复位期间，Flash 模块使用从 Flash 配置字段读取的加密字节对 Flash 安全寄存器 (FTMRE\_FSEC) 进行初始化。

#### 注

加密特性仅适用于外部访问；CPU 对 Flash 的访问不受 Flash 安全寄存器 (FTMRE\_FSEC) 状态的影响。

在未加密状态中，编程接口支持来自调试端口(SWD)或用户执行代码的所有 Flash 命令。当 Flash 受加密保护时 (FTMRE\_FSEC[SEC] = 00、01 或 11)，仅允许编程器接口执行批量擦除操作。此外，在该模式下，调试端口无法访问存储器位置。

### 8.3 与其他模块之间的安全性交互

系统通过 Flash 加密设置确定可用资源。下面的章节介绍模块和 Flash 加密设置之间的交互或 Flash 加密对非 Flash 数据块的影响。

#### 8.3.1 与调试之间的加密交互

当 Flash 处于加密状态时，SWD 端口无法访问 MCU 的存储器资源。

虽然大部分调试功能被禁用，但是调试器可以写 MDM-AP 控制寄存器的“FLASH 整体擦除进度”字段以触发整体擦除（擦除所有数据块）命令。即使某些存储器位置受保护，也能通过调试器进行整体擦除。

# 第 9 章 调试

## 9.1 简介

该器件基于 ARM CoreSight 架构进行调试，配置为在引脚分配和其他可用资源所允许的最大限度内提供最大的灵活性。

它通过外部调试器接口、基本运行/停止控制加上 2 个断点和 2 个观察点，提供对寄存器和存储器的访问。

该器件仅支持一个调试接口，即串行线调试(SWD)。

## 9.2 调试端口引脚说明

上电复位(POR)后，调试端口引脚默认为其 SWD 功能。

表 9-1. 串行线调试引脚说明

引脚名称	类型	说明
SWD_CLK	输入	串行线时钟。该引脚在串行线调试模式下作为调试逻辑的时钟。 <sup>1</sup>
SWD_DIO	输入/输出	串行线调试数据输入/输出。外部调试工具通过 SWD_DIO 引脚进行通信和器件控制。此引脚在内部上拉。

1. 该器件不支持片上下拉；SWD\_CLK 引脚仅支持由 PTAP0 控制的上拉，完全支持 SWD 协议需要外部下拉电阻。

## 9.3 SWD 状态和控制寄存器

通过 ARM 调试访问端口(DAP)，调试器可访问以寄存器形式在 DAP 总线(如图 9-1 所示)上实施的状态和控制元件。这类寄存器针对低功耗模式恢复和典型运行-控制场景提供额外的控制和状态。凭借状态寄存器位，调试器无需通过交叉开关发起总线操作即能获得内核的已更新状态，从而确保调试期间更少的干扰。

该器件上包括了辅助调试模块(MDM)，其中包含 DAP 控制和状态寄存器。值得注意的是，这些 DAP 控制和状态寄存器并非在系统存储器映像中进行地址映射，并且只能通过使用 SWD 的调试访问端口进行访问。调试访问端口 1 的 MDM-AP 的可访问寄存器如下表所示。

表 9-2. MDM-AP 寄存器汇总

地址	寄存器	说明
0x0100_0000	状态	参见 <a href="#">MDM-AP 状态寄存器</a>
0x0100_0004	控制	参见 <a href="#">MDM-AP 控制寄存器</a>
0x0100_00FC	IDR	只读识别寄存器，始终以 0x001C_0020 读取

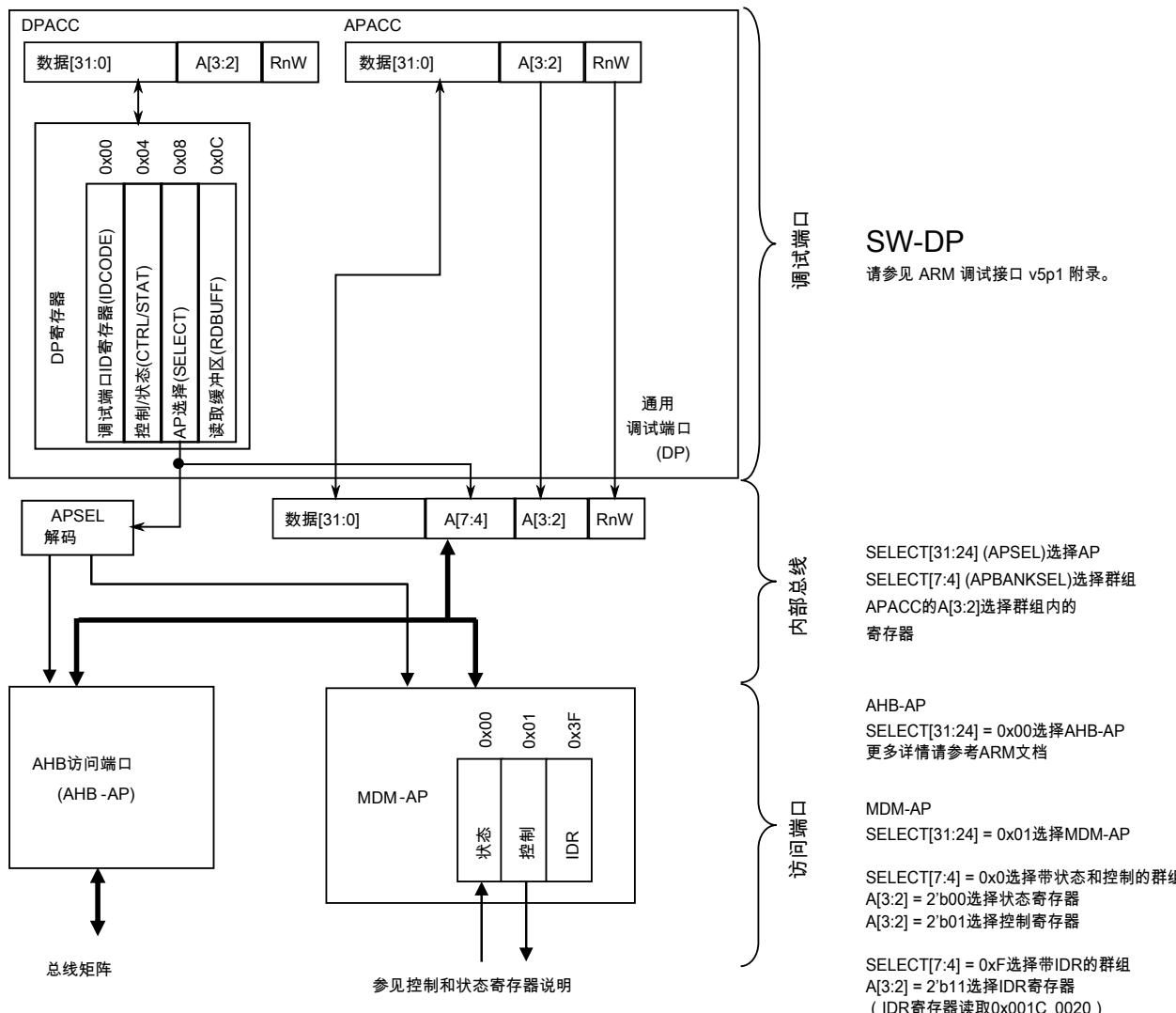


图 9-1. MDM AP 寻址

### 9.3.1 MDM-AP 状态寄存器

表 9-3. MDM-AP 状态寄存器分配

位	名称	说明
0	Flash 整体擦除应答	Flash 整体擦除应答字段在 POR 复位后清零。通过对 MDM AP 控制寄存器的“正在执行 Flash 整体擦除”字段执行写操作而发出整体擦除命令时该字段也会清零。在 Flash 控制逻辑开始了整体擦除操作后，将设置该 Flash 整体擦除应答。
1	Flash 就绪	表示 Flash 存储器已完成初始化，调试器可配置，即使调试器继续保持系统在复位状态。 0 Flash 正在初始化。 1 Flash 就绪。
2	系统加密	表示加密状态。处于加密状态时，调试器无法访问系统总线或任何存储器映射外设。该字段表示器件锁定且无法进行系统总线访问。 注：该位在 Flash 就绪位置位前无效。 0 器件未处于加密状态。 1 器件处于加密状态。
3	系统复位	表示系统复位状态。 0 系统已处于复位状态。 1 系统未处于复位状态。
4	保留	
5 – 15	保留供将来使用	始终读取 0。
16	内核暂停	表示内核已进入调试暂停模式 0 内核未暂停。 1 内核已暂停。
17	内核 SLEEPDEEP	SLEEPDEEP=1 表示内核已进入 Stop 模式。
18	内核 SLEEPING	SLEEPING=1 表示内核已进入 Wait 模式。
19 – 31	保留供将来使用	始终读取 0。

### 9.3.2 MDM-AP 控制寄存器

表 9-4. MDM-AP 控制寄存器分配

位	名称	加密 <sup>1</sup>	说明
0	正在执行 Flash 整体擦除	是	置位以导致整体擦除。批量擦除操作后由硬件进行的清零完成。
1	调试禁用	否	置位禁用调试。清零允许调试操作。置位时，它覆盖 DHCSR 中的 C_DEBUGEN 字段 <sup>2</sup> 并强制禁用调试逻辑。
2	调试请求	否	置位强制暂停内核。

下一页继续介绍此表...

表 9-4. MDM-AP 控制寄存器分配 (继续)

位	名称	加密 <sup>1</sup>	说明
			若内核处于 Wait 或 Stop 模式，该字段可用于唤醒内核并转变到暂停状态。
3	系统复位请求	是	置位强制进行系统复位。系统在该字段清零前都保持在复位状态。该位置位后，RESET 引脚不反映系统复位的状态并且不保持低电平。
4	内核保持	否	配置字段，可控制系统复位序列结束时的内核操作。 0 正常运行—在系统复位序列结束时，解除内核与系统的其他部分的复位。 1 挂起运行—在复位序列结束后将内核保持在复位状态。一旦系统进入挂起状态，该控制位清零可立即解除内核的复位状态并且 CPU 开始工作。
5–31	保留供将来使用	否	

1. 加密状态下可用的命令
2. DHCSR：指的是 ARMv6-M 架构参考手册中的“调试暂停控制和状态寄存器”。

## 9.4 调试复位

调试系统将收到以下复位源：

- 系统 POR 复位

相反，调试系统能够使用以下机制产生系统复位：

- DAP 控制寄存器中的系统复位，允许调试器保持系统在复位状态。
- 将 1 写入 NVIC 应用中断和复位控制寄存器中的 SYSRESETREQ 字段
- DAP 控制寄存器中的系统复位，允许调试器保持内核在复位状态。

## 9.5 低功耗模式下的调试

在调试模块保持为静态或掉电状态的低功耗模式下，调试器无法在低功耗模式时间段内采集任何调试数据。

- 如果调试器保持为静态，那么只要退出低功耗模式且系统返回到某个具有主动调试的状态，调试器端口就会恢复所有功能。
- 如果调试器逻辑处于掉电状态，那么调试器在恢复时就会复位并且在退出低功耗模式时必须重新配置。

主动调试将防止芯片进入低功耗模式。如果芯片已处于低功耗模式，那么来自 MDM-AP 控制寄存器的调试请求将从低功耗模式唤醒该芯片。

## 9.6 调试和加密

使能 Flash 加密后，调试端口的功能将受限，以防对安全数据的不当利用。在加密状态下，调试器仍然可以访问状态寄存器，并且可以确定该器件当前的加密状态。如果器件处于加密状态，那么调试器仅可以执行整体擦除操作。



# 第 10 章

## 信号多路复用和信号说明

### 10.1 简介

为了优化小型封装的功能，引脚通过信号多路复用技术实现多种可用功能。本章说明该器件的哪些信号在哪个外部引脚上多路复用。

[引脚选择寄存器 0 \(SIM\\_PINSEL0\)](#)和[引脚选择寄存器 1 \(SIM\\_PINSEL1\)](#)控制该外部引脚上有哪些信号。有关特定多路复用引脚控制操作的详细信息，请参考该寄存器的相关内容。

### 10.2 引脚分配

#### 10.2.1 信号多路复用和引脚分配

下表显示的是各引脚上的信号以及这些引脚在本文档支持的器件上的位置。“端口控制模块”负责选择每个引脚上的 ALT 功能。

##### 注

VSS 与 VSSA 在内部相连。

VREFH 与 VDDA 在 64 引脚封装中内部相连。

PTB4、PTB5、PTD0、PTD1、PTE0、PTE1、PTH0 和 PTH1 用作输出时为大电流驱动引脚。

PTA2 和 PTA3 在用作输出时为有效的开漏引脚。

80 LQFP	64 LQFP /QFP	Pin Name	默认值	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
1	1	PTD1	禁用	PTD1	KBIO_P25	FTM2_CH3	SPI1_MOSI				
2	2	PTD0	禁用	PTD0	KBIO_P24	FTM2_CH2	SPI1_SCK				

80 LQFP	64 LQFP /QFP	Pin Name	默认值	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
3	3	PTH7	禁用	PTH7	KBI1_P31	PWT_IN1					
4	4	PTH6	禁用	PTH6	KBI1_P30						
5	—	PTH5	禁用	PTH5	KBI1_P29						
6	5	PTE7	禁用	PTE7	KBI1_P7	TCLK2		FTM1_CH1	CANO_TX		
7	6	PTH2	禁用	PTH2	KBI1_P26	BUSOUT		FTM1_CH0	CANO_RX		
8	7	VDD	VDD							VDD	
9	8	VDDA	VDDA						VREFH	VDDA	
10	—	VREFH	VREFH							VREFH	
11	9	VREFL	VREFL							VREFL	
12	10	VSS/ VSSA	VSS/ VSSA						VSSA	VSS	
13	11	PTB7	EXTAL	PTB7	KBI0_P15	I2C0_SCL				EXTAL	
14	12	PTB6	XTAL	PTB6	KBI0_P14	I2C0_SDA				XTAL	
15	13	PTI4	禁用	PTI4		IRQ					
16	—	PTI1	禁用	PTI1		IRQ	UART2_TX				
17	—	PTI0	禁用	PTI0		IRQ	UART2_RX				
18	14	PTH1	禁用	PTH1	KBI1_P25	FTM2_CH1					
19	15	PTH0	禁用	PTH0	KBI1_P24	FTM2_CH0					
20	16	PTE6	禁用	PTE6	KBI1_P6						
21	17	PTE5	禁用	PTE5	KBI1_P5						
22	18	PTB5	禁用	PTB5	KBI0_P13	FTM2_CH5	SPI0_PCS	ACMP1_OUT			
23	19	PTB4	NMI_b	PTB4	KBI0_P12	FTM2_CH4	SPI0_MISO	ACMP1_IN2	NMI_b		
24	20	PTC3	ADC0_SE11	PTC3	KBI0_P19	FTM2_CH3		ADC0_SE11			
25	21	PTC2	ADC0_SE10	PTC2	KBI0_P18	FTM2_CH2		ADC0_SE10			
26	22	PTD7	禁用	PTD7	KBI0_P31	UART2_TX					
27	23	PTD6	禁用	PTD6	KBI0_P30	UART2_RX					
28	24	PTD5	禁用	PTD5	KBI0_P29	PWT_IN0					
29	—	PTI6	禁用	PTI6	IRQ						
30	—	PTI5	禁用	PTI5	IRQ						
31	25	PTC1	ADC0_SE9	PTC1	KBI0_P17	FTM2_CH1		ADC0_SE9			
32	26	PTC0	ADC0_SE8	PTC0	KBI0_P16	FTM2_CH0		ADC0_SE8			
33	—	PTH4	禁用	PTH4	KBI1_P28	I2C1_SCL					
34	—	PTH3	禁用	PTH3	KBI1_P27	I2C1_SDA					
35	27	PTF7	ADC0_SE15	PTF7	KBI1_P15			ADC0_SE15			
36	28	PTF6	ADC0_SE14	PTF6	KBI1_P14			ADC0_SE14			
37	29	PTF5	ADC0_SE13	PTF5	KBI1_P13			ADC0_SE13			
38	30	PTF4	ADC0_SE12	PTF4	KBI1_P12			ADC0_SE12			
39	31	PTB3	ADC0_SE7	PTB3	KBI0_P11	SPI0_MOSI	FTM0_CH1	ADC0_SE7			
40	32	PTB2	ADC0_SE6	PTB2	KBI0_P10	SPI0_SCK	FTM0_CH0	ADC0_SE6			
41	33	PTB1	ADC0_SE5	PTB1	KBI0_P9	UART0_TX		ADC0_SE5			
42	34	PTB0	ADC0_SE4	PTB0	KBI0_P8	UART0_RX	PWT_IN1	ADC0_SE4			

80 LQFP	64 LQFP /QFP	Pin Name	默认值	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
43	35	PTF3	禁用	PTF3	KBI1_P11	UART1_TX					
44	36	PTF2	禁用	PTF2	KBI1_P10	UART1_RX					
45	37	PTA7	ADC0_SE3	PTA7	KBI0_P7	FTM2_FLT2	ACMP1_IN1	ADC0_SE3			
46	38	PTA6	ADC0_SE2	PTA6	KBI0_P6	FTM2_FLT1	ACMP1_IN0	ADC0_SE2			
47	39	PTE4	禁用	PTE4	KBI1_P4						
48	40	VSS	VSS							VSS	
49	41	VDD	VDD							VDD	
50	—	PTG7	禁用	PTG7	KBI1_P23	FTM2_CH5	SPI1_PCS				
51	—	PTG6	禁用	PTG6	KBI1_P22	FTM2_CH4	SPI1_MISO				
52	—	PTG5	禁用	PTG5	KBI1_P21	FTM2_CH3	SPI1_MOSI				
53	—	PTG4	禁用	PTG4	KBI1_P20	FTM2_CH2	SPI1_SCK				
54	42	PTF1	禁用	PTF1	KBI1_P9	FTM2_CH1					
55	43	PTF0	禁用	PTF0	KBI1_P8	FTM2_CH0					
56	44	PTD4	禁用	PTD4	KBI0_P28						
57	45	PTD3	禁用	PTD3	KBI0_P27	SPI1_PCS					
58	46	PTD2	禁用	PTD2	KBI0_P26	SPI1_MISO					
59	47	PTA3	禁用	PTA3	KBI0_P3	UART0_TX	I2C0_SCL				
60	48	PTA2	禁用	PTA2	KBI0_P2	UART0_RX	I2C0_SDA				
61	49	PTA1	ADC0_SE1	PTA1	KBI0_P1	FTM0_CH1	I2C0_4WSDAOUT	ACMP0_IN1	ADC0_SE1		
62	50	PTA0	ADC0_SE0	PTA0	KBI0_P0	FTM0_CH0	I2C0_4WSCLOUT	ACMP0_IN0	ADC0_SE0		
63	51	PTC7	禁用	PTC7	KBI0_P23	UART1_TX			CAN0_TX		
64	52	PTC6	禁用	PTC6	KBI0_P22	UART1_RX			CAN0_RX		
65	—	PTI3	禁用	PTI3	IRQ						
66	—	PTI2	禁用	PTI2	IRQ						
67	53	PTE3	禁用	PTE3	KBI1_P3	SPI0_PCS					
68	54	PTE2	禁用	PTE2	KBI1_P2	SPI0_MISO	PWT_IN0				
69	—	VSS	VSS							VSS	
70	—	VDD	VDD							VDD	
71	55	PTG3	禁用	PTG3	KBI1_P19						
72	56	PTG2	禁用	PTG2	KBI1_P18						
73	57	PTG1	禁用	PTG1	KBI1_P17						
74	58	PTG0	禁用	PTG0	KBI1_P16						
75	59	PTE1	禁用	PTE1	KBI1_P1	SPI0_MOSI		I2C1_SCL			
76	60	PTE0	禁用	PTE0	KBI1_P0	SPI0_SCK	TCLK1	I2C1_SDA			
77	61	PTC5	禁用	PTC5	KBI0_P21		FTM1_CH1		RTC_CLKOUT		
78	62	PTC4	SWD_CLK	PTC4	KBI0_P20	RTC_CLKOUT	FTM1_CH0	ACMP0_IN2	SWD_CLK		
79	63	PTA5	RESET_b	PTA5	KBI0_P5	IRQ	TCLK0	RESET_b			
80	64	PTA4	SWD_DIO	PTA4	KBI0_P4		ACMP0_OUT	SWD_DIO			

## 10.2.2 器件引脚分配

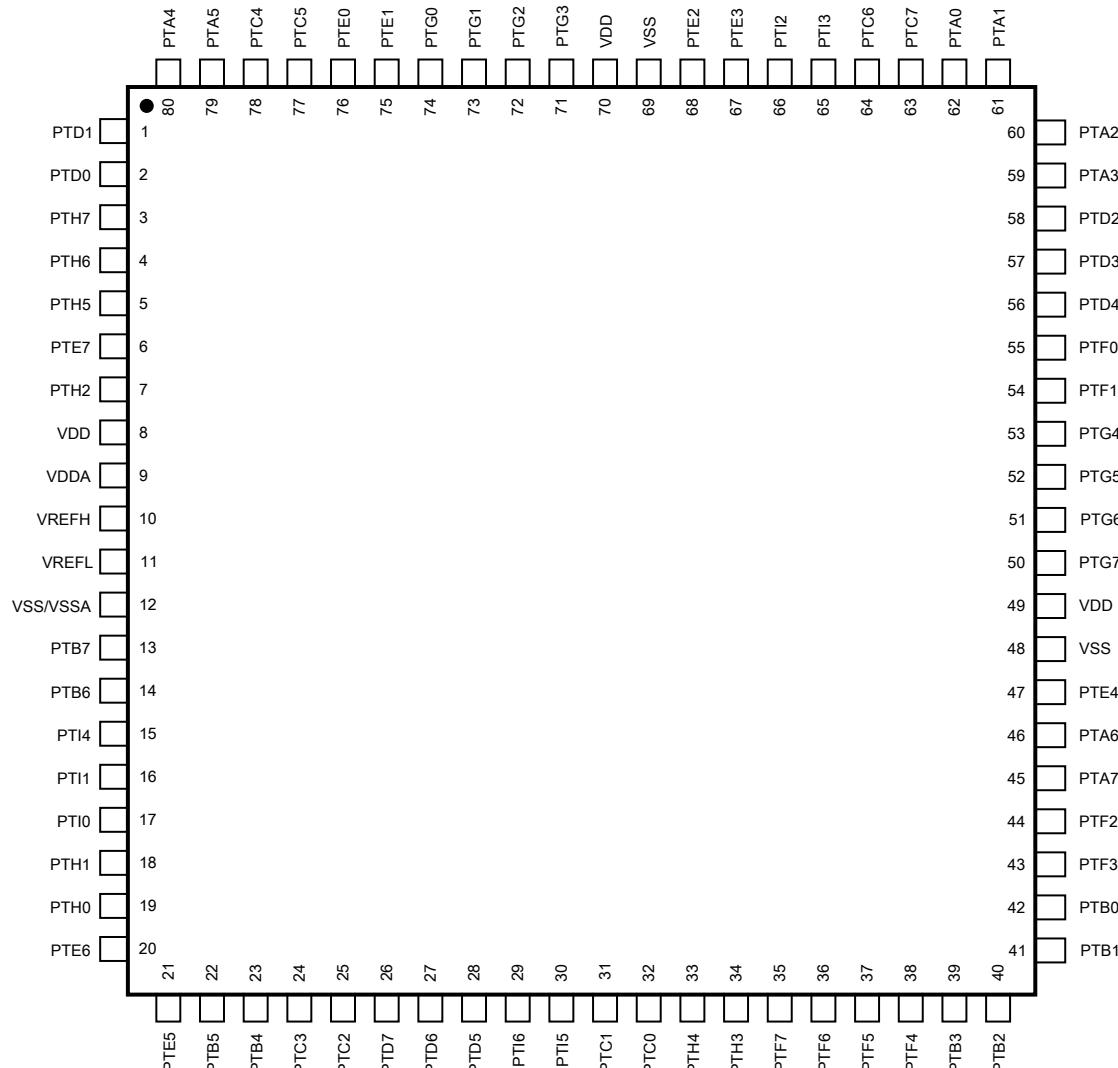


图 10-1. 80 引脚 LQFP 封装

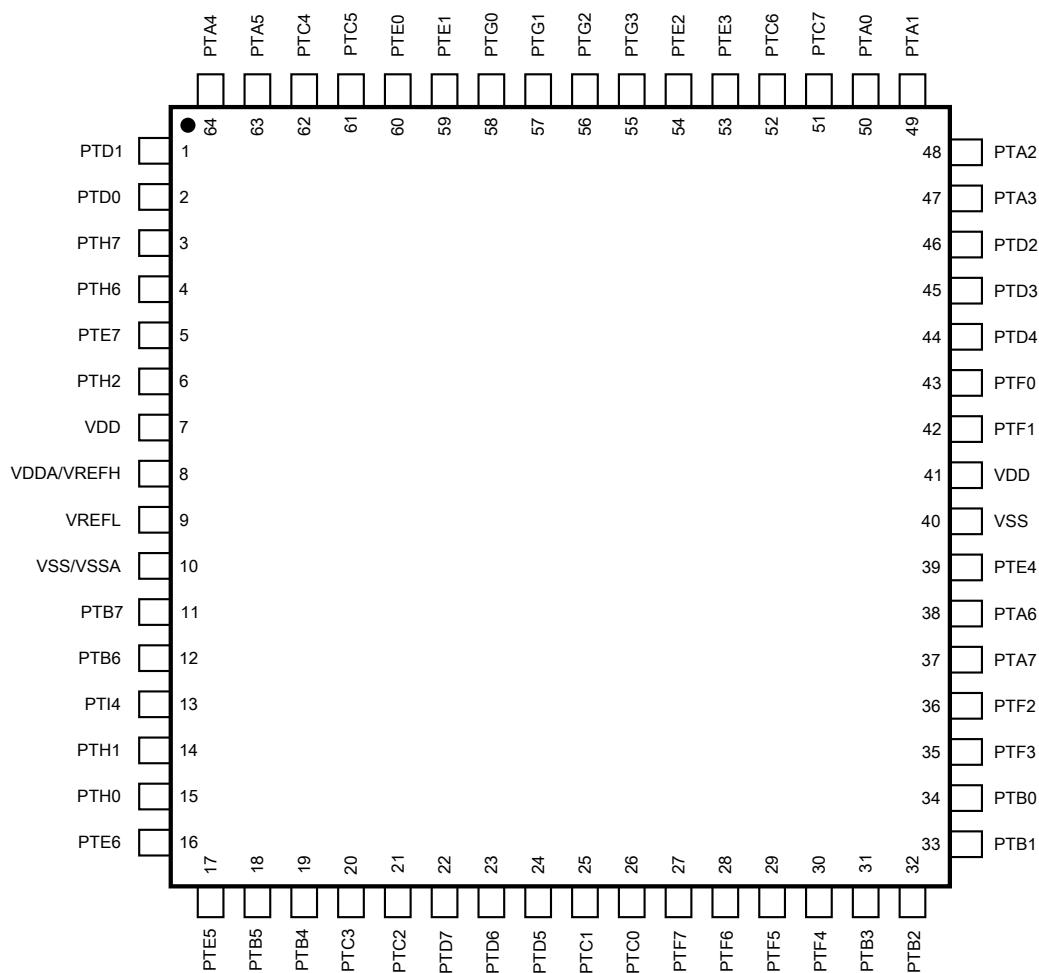


图 10-2. 64 引脚 LQFP 封装

## 10.3 模块信号描述表

下面的章节描述芯片级信号名称与模块章节中使用的信号名称关联。还简要介绍信号功能和方向。

### 10.3.1 内核模块

表 10-1. SWD 信号说明

芯片信号名称	模块信号名称	说明	I/O
SWD_DIO	SWD_DIO	串行线调试数据输入/输出。外部调试工具通过 SWD_DIO 引脚进行通信和器件控制。此引脚在内部上拉。	输入/输出
SWD_CLK	SWD_CLK	串行线时钟。该引脚在串行线调试模式下作为调试逻辑的时钟。 <sup>1</sup>	输入

1. 该器件不支持片上下拉；SWD\_CLK 引脚仅支持由 PTAPE0 控制的上拉，完全支持 SWD 协议需要外部下拉电阻。

## 10.3.2 系统模块

表 10-2. 系统信号说明

芯片信号名称	模块信号名称	说明	I/O
NMI	—	非屏蔽中断  注：如果相应引脚选择 NMI 功能，那么将 NMI 信号驱动至低电平会强制生成非屏蔽中断。	I
RESET	—	复位双向信号	I/O
VDD	—	MCU 电源	I
VSS	—	MCU 接地	I

## 10.3.3 时钟模块

表 10-3. OSC 信号说明

芯片信号名称	模块信号名称	说明	I/O
EXTAL	EXTAL	外部时钟/振荡器输入	模拟输入
XTAL	XTAL	振荡器输出	模拟输出

## 10.3.4 模拟

表 10-4. ADC0 信号说明

芯片信号名称	模块信号名称	说明	I/O
ADC0_SEn	AD15-AD0	模拟通道输入	I
VDD/VREFH	VDDA/VREFH	模拟电源/基准电压源高电平	I
VSS/VREFL	VSSA/VREFL	模拟电源地/基准电压源低电平	I

表 10-5. ACMP0 信号说明

芯片信号名称	模块信号名称	说明	I/O
ACMP0_INn	ACMP0_IN[2:0]	模拟电压输入	I
ACMP0_OUT	ACMP0_OUT	比较器输出	O

表 10-6. ACMP1 信号说明

芯片信号名称	模块信号名称	说明	I/O
ACMP1_INn	ACMP1_IN[2:0]	模拟电压输入	I
ACMP1_OUT	ACMP1_OUT	比较器输出	O

### 10.3.5 定时器模块

表 10-7. FTM0 信号说明

芯片信号名称	模块信号名称	说明	I/O
TCLKn	EXTCLK	FTM 外部时钟	I
FTM0_CH[1:0]	CHn	FTM 通道	I/O

表 10-8. FTM1 信号说明

芯片信号名称	模块信号名称	说明	I/O
TCLKn	EXTCLK	FTM 外部时钟	I
FTM1_CH[1:0]	CHn	FTM 通道	I/O

表 10-9. FTM2 信号说明

芯片信号名称	模块信号名称	说明	I/O
TCLKn	EXTCLK	FTM 外部时钟	I
FTM2_CHn	CHn	FTM 通道	I/O
FTM2_FLT1	FAULT1	故障输入(1)	I
FTM2_FLT2	FAULT2	故障输入(2)	I

表 10-10. RTC 信号说明

芯片信号名称	模块信号名称	说明	I/O
RTC_CLKOUT	RTCO	RTC 时钟输出	O

表 10-11. PWT 信号说明

芯片信号名称	模块信号名称	说明	I/O
TCLKn	ALTCLK	PWT 备用外部时钟	O
PWT_IN0	PWTIN[0]	PWT 输入通道 0	I
PWT_IN1	PWTIN[1]	PWT 输入通道 1	I

### 10.3.6 通信接口

表 10-12. SPI0 信号说明

芯片信号名称	模块信号名称	说明	I/O
SPI0_MISO	MISO	主机数据输入, 从机数据输出	I/O
SPI0_MOSI	MOSI	主机数据输出, 从机数据输入	I/O
SPI0_SCK	SPSCK	SPI 串行时钟	I/O
SPI0_PCS	SS	从机选择	I/O

表 10-13. SPI1 信号说明

芯片信号名称	模块信号名称	说明	I/O
SPI1_MISO	MISO	主机数据输入, 从机数据输出	I/O
SPI1_MOSI	MOSI	主机数据输出, 从机数据输入	I/O
SPI1_SCK	SPSCK	SPI 串行时钟	I/O
SPI1_PCS	SS	从机选择	I/O

表 10-14. I<sup>2</sup>C0 信号说明

芯片信号名称	模块信号名称	说明	I/O
I2C0_SCL	SCL	I <sup>2</sup> C 系统的双向串行时钟线路。	I/O
I2C0_SDA	SDA	I <sup>2</sup> C 系统的双向串行数据线路。	I/O
I2C0_4WSCLOUT		采用四线式接口配置的 I <sup>2</sup> C0 串行时钟线路输出	O
I2C0_4WSDAOOUT		采用四线式接口配置的 I <sup>2</sup> C0 串行数据线路输出	O

表 10-15. I<sup>2</sup>C1 信号说明

芯片信号名称	模块信号名称	说明	I/O
I2C1_SCL	SCL	I <sup>2</sup> C 系统的双向串行时钟线路。	I/O
I2C1_SDA	SDA	I <sup>2</sup> C 系统的双向串行数据线路。	I/O

表 10-16. UART0 信号说明

芯片信号名称	模块信号名称	说明	I/O
UART0_TX	TxD	发送数据	I/O
UART0_RX	RxD	接收数据	I

表 10-17. UART1 信号说明

芯片信号名称	模块信号名称	说明	I/O
UART1_TX	TxD	发送数据	I/O
UART1_RX	RxD	接收数据	I

表 10-18. UART2 信号描述

芯片信号名称	模块信号名称	说明	I/O
UART2_TX	TxD	发送数据	I/O
UART2_RX	RxD	接收数据	I

表 10-19. MSCAN 信号说明

芯片信号名称	模块信号名称	说明	I/O
CAN0_RX	RXCAN	发送数据	I
CAN0_TX	TXCAN	接收数据	O

### 10.3.7 人机接口(HMI)

表 10-20. GPIO 信号说明

芯片信号名称	模块信号名称	说明	I/O
PTA[7:0] <sup>1</sup>	PORATA7-PORATA0	通用输入/输出	I/O
PTB[7:0] <sup>1</sup>	PORATA15-PORATA8	通用输入/输出	I/O
PTC[7:0] <sup>1</sup>	PORATA23-PORATA16	通用输入/输出	I/O
PTD[7:0] <sup>1</sup>	PORATA31-PORATA24	通用输入/输出	I/O
PTE[7:0] <sup>1</sup>	PORTB7-PORTB0	通用输入/输出	I/O
PTF[7:0] <sup>1</sup>	PORTB15-PORTB8	通用输入/输出	I/O
PTG[7:0] <sup>1</sup>	PORTB23-PORTB16	通用输入/输出	I/O
PTH[7:0] <sup>1</sup>	PORTB31-PORTB24	通用输入/输出	I/O
PTI[6:0] <sup>1</sup>	PORTC6-PORTC0	通用输入/输出	I/O

1. 可用 GPIO 引脚取决于特定封装。有关具体哪些 GPIO 信号可用，请参见“信号多路复用”部分。

表 10-21. KBI0 信号说明

芯片信号名称	模块信号名称	说明	I/O
KBI0_Pn	KBI0Pn	键盘中断引脚, n 可以是 0 ~ 31	I/O

**表 10-22. KBI1 信号说明**

芯片信号名称	模块信号名称	说明	I/O
KBI1_Pn	KBI1Pn	键盘中断引脚, n 可以是 0 ~ 31	I/O

**表 10-23. IRQ 信号说明**

芯片信号名称	模块信号名称	说明	I/O
IRQ	IRQ	IRQ 输入	I

# 第 11 章 端口控制(PORT)

## 11.1 简介

该器件具有 9 组 I/O 端口，最多可支持 71 个通用 I/O 引脚。

并非所有器件都提供全部引脚。

很多 I/O 引脚都共用片上外设功能。外设模块的优先级高于 I/O，因此当启用外设时，会禁用相关的 I/O 功能。

复位后，共用的外设功能被禁用，因此这些引脚通过并行 I/O 控制，但默认关联至 SWD\_DIO、SWD\_CLK、 $\overline{NMI}$  和  $\overline{RESET}$  功能的 PTA4、PTA5、PTB4 和 PTC4 除外。所有并行 I/O 都配置为高阻抗状态(Hi-Z)。每个引脚的控制功能配置如下：

- 输入禁用(GPIOx\_PIDR[PID] = 1)，
- 输出禁用(GPIOx\_PDDR[PDD] = 0)，以及
- 内部上拉禁用(PORT\_PUE(0/1/2)[PTxPEn] = 0)。

此外，支持大电流驱动的并行 I/O 在复位后禁用(HDRVE = 0x00)。

以下三张图显示每个 I/O 引脚的结构。

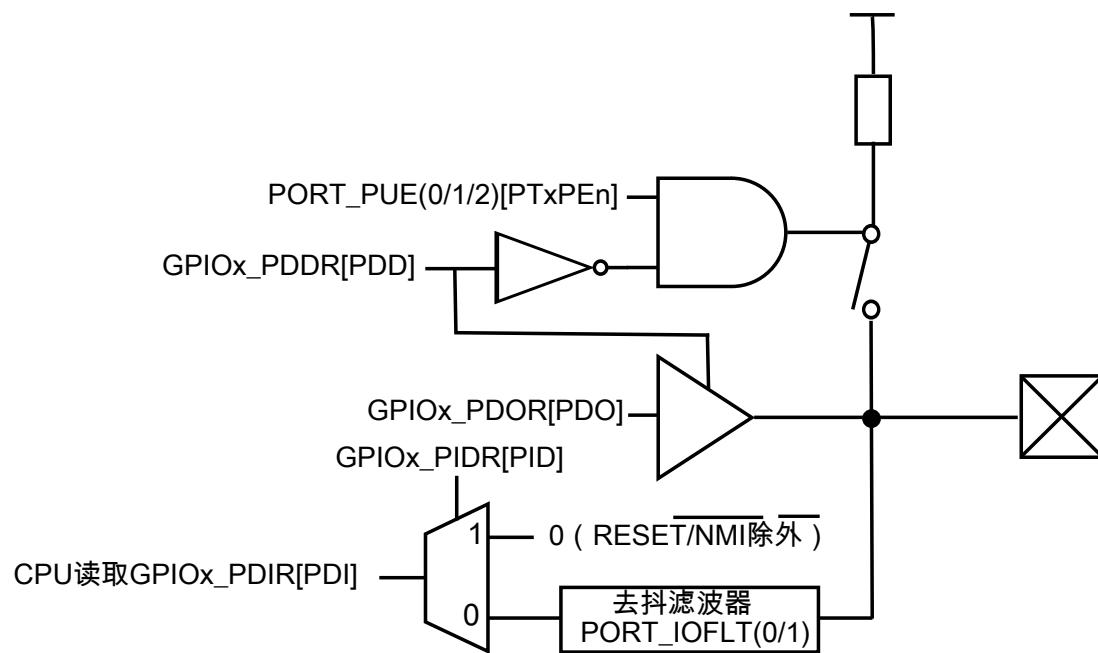


图 11-1. 普通 I/O 结构

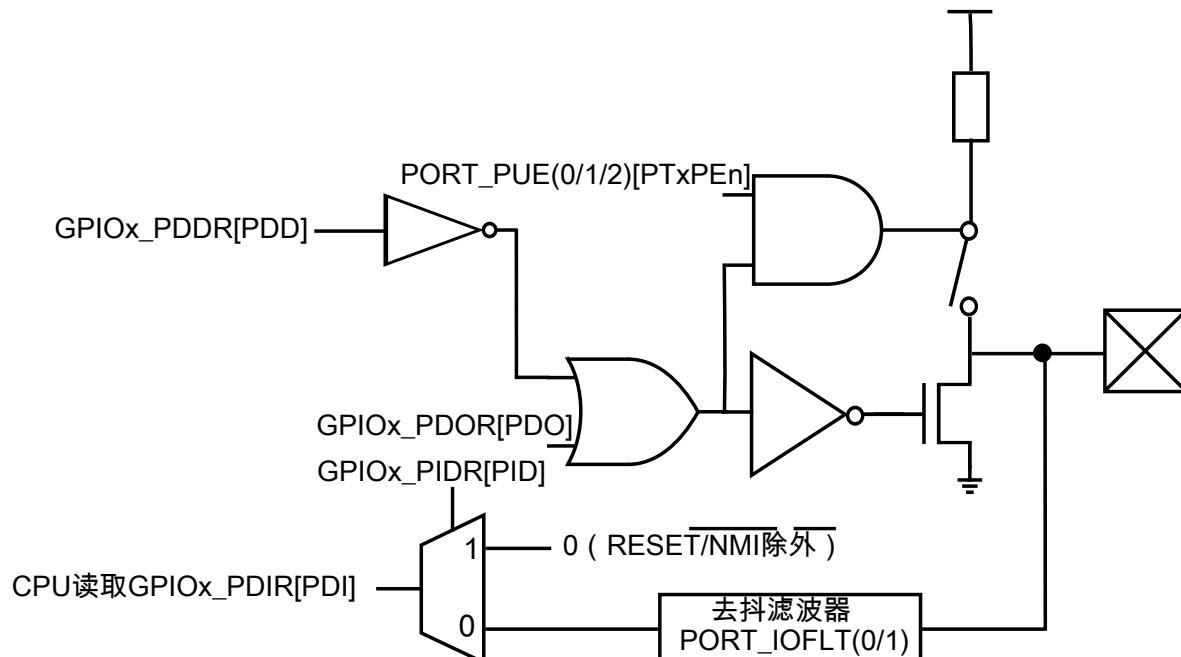


图 11-2. SDA(PTA2)/SCL(PTA3) 结构

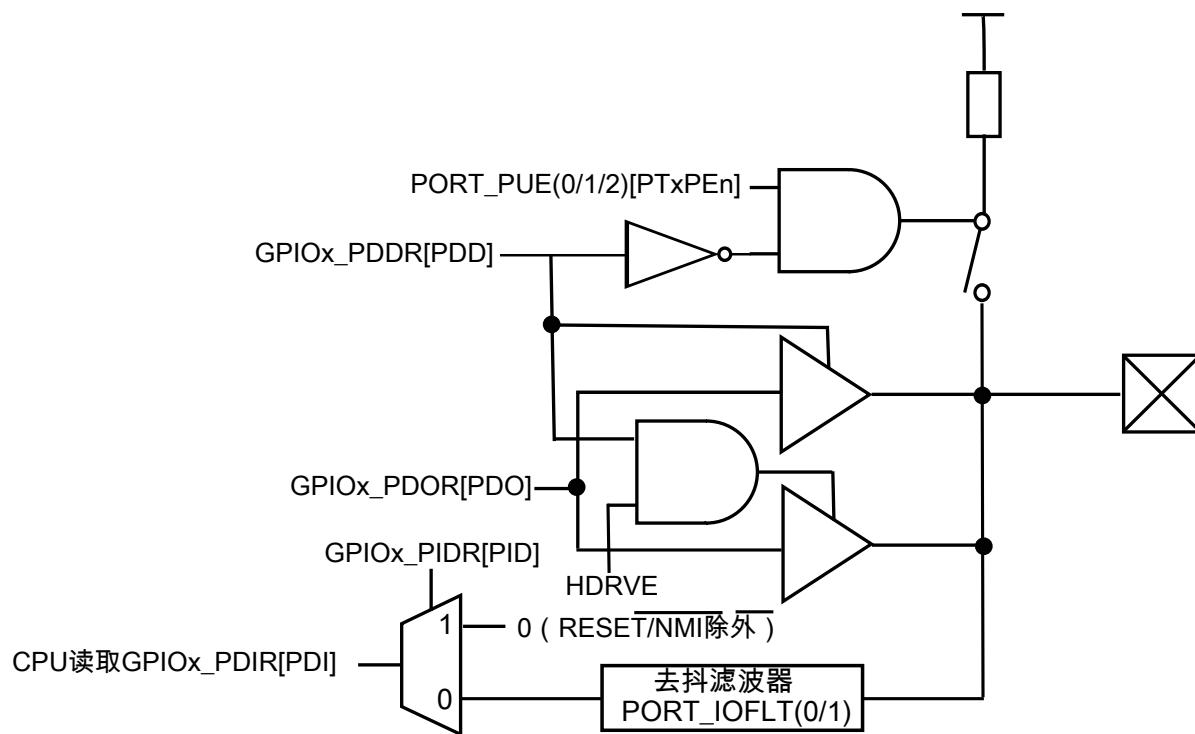


图 11-3. 大电流驱动 I/O 结构

## 11.2 端口数据和数据方向

并行 I/O 的读取和写入通过端口数据寄存器(GPIOx\_PDIR/PDOR)实现。数据方向(输入或输出)通过输入禁用寄存器(GPIOx\_PIDR)和数据方向寄存器(GPIOx\_PDDR)进行控制。

复位后，所有并行 I/O 均默认为 Hi-Z 状态。必须配置端口数据方向寄存器(GPIOx\_PDDR)或输入禁用寄存器(GPIOx\_PIDR)中的相应位以便进行输出或输入操作。每个端口引脚都有一个输入禁用位和一个输出启用位。当 GPIOx\_PIDR[PID] = 0 时，读取 GPIOx\_PDIR[PDI]将返回对应引脚的输入值；当 GPIOx\_PIDR[PID] = 1 时，读取 GPIOx\_PDIR[PDI]将返回 0，但 RESET/NMI 除外。

### 注

当引脚用作输入功能时必须将对应引脚的 GPIOx\_PDDR 清零，以防冲突。如果同时设置 GPIOx\_PDDR 和 GPIOx\_PIDR 位，则读取 GPIOx\_PDIR 将始终读取引脚状态。

当外设模块或系统功能控制端口引脚时，即使外设对实际引脚的数据方向控制有更高的优先级，数据方向寄存器位也仍然控制针对端口数据寄存器的读操作所返回的内容。

当针对某个引脚使能某个共用模拟功能时，这个引脚的数字功能都被禁用。对于任何使能了共用的模拟功能的端口而言，对端口数据寄存器进行读操作将返回数值 0。通常情况下，只要某个引脚由数字功能和模拟功能共用，那么模拟功能就具有这样的优先级：在数字功能和模拟功能都使能的情况下，模拟功能将控制该引脚。

向端口数据寄存器写入有效数据的操作必须在设置相关端口引脚的输出使能位之前完成。这样可确保该引脚不会写入错误的数据值。

### 11.3 内部上拉使能

通过设置上拉使能寄存器(PORT\_PUE(0/1/2))中的相应位，可以使能对应引脚的内部上拉电阻。如果此引脚通过并行 I/O 控制逻辑或任何共享外设功能被配置为输出，那么无论对应的上拉使能寄存器位的状态如何，此引脚的内部上拉电阻都会被禁用。若引脚由模拟功能控制，内部上拉电阻同样会被禁用。

#### 注

当配置 I2C0 时要使用“SDA(PTA2/PTB6)”和“SCL(PTA3/PTB7)”引脚，或配置 I2C1 时要使用“SDA(PTE0/PTH3)”和“SCL(PTE1/PTH4)”引脚，并且采用的是内部上拉电阻而非外部上拉电阻，那么当这些引脚配置为输出时，内部上拉电阻仍保持当前设置，不过在输出低电平时会被自动禁用以省电。

### 11.4 输入去抖滤波器设置

滤波器适用于每个配置成数字输入的引脚。它可用作简单的低通滤波器，对 GPIO、FTM0、FTM1、I2C0、I2C1、PWT、IRQ、 $\overline{\text{RESET}}$ 、 $\overline{\text{NMI}}$  和 KBI 引脚产生的任何毛刺进行滤波。毛刺宽度阈值可在 1~4096 BUSCLK (或 1~128 LPOCLK) 的区间通过设置 PORT\_IOFLT0[FLTDIVn]轻松调节。这个可配置的毛刺滤波电路可替代板载外部模拟滤波器，并且能极大地提高 EMC 性能，因为任何毛刺都不会被错误地采样或忽略。

设置寄存器 PORT\_IOFLT(0/1)可配置整个端口或外设输入的滤波器。例如，设置 PORT\_IOFLT0[FLTA]将会影响所有 PTAn 引脚。

脉宽比所选时钟周期更短的毛刺会被滤除；脉宽比所选时钟周期长两倍以上的毛刺将不会被滤除，并且会进入内部电路。

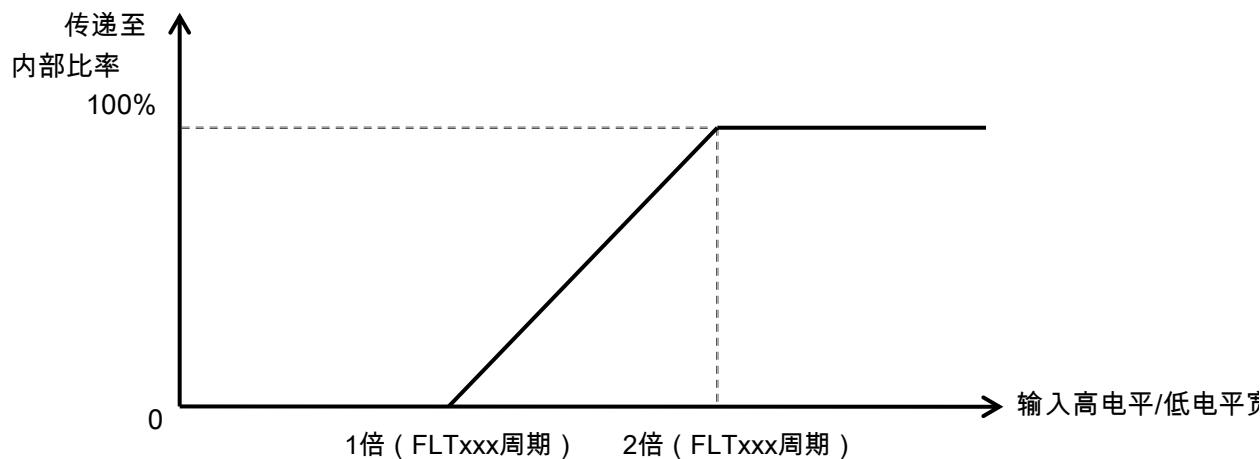


图 11-4. 输入去抖滤波器

## 11.5 大电流驱动

通过使能 HDRVE 寄存器中的相关位，可以使能的大电流驱动功能，作为输出灌电流或拉电流。这些引脚可用作输出和输入；引脚作为输出时，输出的是大电流功能的灌电流或拉电流。

- 若引脚通过并行 I/O 控制逻辑配置为输入，则大电流驱动功能禁用。
- 当配置为任何共用外设功能时，大电流驱动功能对这些引脚依然有效，但仅当这些引脚配置为输出时才有效。

## 11.6 Stop 模式下的引脚特性

在 Stop 模式下，会保持所有 I/O 状态，因为内部逻辑电路保持上电状态。只要一恢复，正常的 I/O 功能就可供用户使用。

## 11.7 端口数据寄存器

### PORT 存储器映射

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4004_9000	端口滤波寄存器 0 (PORT_IOFLT0)	32	R/W	00C0_0000h	11.7.1/148
4004_9004	端口滤波寄存器 1 (PORT_IOFLT1)	32	R/W	0000_0000h	11.7.2/151
4004_9008	端口上拉使能寄存器 0 (PORT_PUE0)	32	R/W	0010_0000h	11.7.3/152

下一页继续介绍此表...

## PORT 存储器映射 (继续)

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4004_900C	端口上拉使能寄存器 1 (PORT_PUE1)	32	R/W	0000_0000h	11.7.4/157
4004_9010	端口上拉使能寄存器 2 (PORT_PUE2)	32	R/W	0000_0000h	11.7.5/161
4004_9014	端口大电流驱动使能寄存器 (PORT_HDRVE)	32	R/W	0000_0000h	11.7.6/162

## 11.7.1 端口滤波寄存器 0 (PORT\_IOFLT0)

这个寄存器是配置输入引脚的滤波器，可以配置高或低电平毛刺的脉宽阈值。脉宽比所选时钟周期更短的毛刺将被滤除；脉宽比所选时钟周期长两倍以上的毛刺将不会被滤除，并且会进入内部电路。

地址: 4004\_9000h 基准 + 0h 偏移 = 4004\_9000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FLTDIV3			FLTDIV2			FLTDIV1		FLTNMI		FLTKBI1		FLTKBI0		FLTRST	
复位	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FLTH		FLTG		FLTF		FLTE		FLTD		FLTC		FLTB		FLTA	
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## PORT\_IOFLT0 字段描述

字段	描述
31-29 FLTDIV3	滤波器分组 3 端口滤波器分组 3  000 LPOCLK 001 LPOCLK/2 010 LPOCLK/4 011 LPOCLK/8 100 LPOCLK/16 101 LPOCLK/32 110 LPOCLK/64 111 LPOCLK/128
28-26 FLTDIV2	滤波器分组 2 端口滤波器分组 2  000 BUSCLK/32 001 BUSCLK/64 010 BUSCLK/128 011 BUSCLK/256 100 BUSCLK/512 101 BUSCLK/1024

下一页继续介绍此表...

## PORT\_IOFLT0 字段描述 (继续)

字段	描述
	110 BUSCLK/2048 111 BUSCLK/4096
25–24 FLTDIV1	滤波器分组 1 端口滤波器分组 1 00 BUSCLK/2 01 BUSCLK/4 10 BUSCLK/8 11 BUSCLK/16
23–22 FLTNMI	针对 NMI 输入的滤波器选择 00 无滤波器。 01 选择 FLTDIV1，在 Stop 模式下会自动切换到 FLTDIV3。 10 选择 FLTDIV2，在 Stop 模式下会自动切换到 FLTDIV3。 11 FLTDIV3
21–20 FLTKBI1	针对 KBI1 输入的滤波器选择 00 无滤波器 01 选择 FLTDIV1，在 Stop 模式下会自动切换到 FLTDIV3。 10 选择 FLTDIV2，在 Stop 模式下会自动切换到 FLTDIV3。 11 FLTDIV3
19–18 FLTKBI0	针对 KBI0 输入的滤波器选择 00 无滤波器。 01 选择 FLTDIV1，在 Stop 模式下会自动切换到 FLTDIV3。 10 选择 FLTDIV2，在 Stop 模式下会自动切换到 FLTDIV3。 11 FLTDIV3
17–16 FLTRST	针对 RESET/IRQ 输入的滤波器选择 00 无滤波器。 01 选择 FLTDIV1，在 Stop 模式下会自动切换到 FLTDIV3。 10 选择 FLTDIV2，在 Stop 模式下会自动切换到 FLTDIV3。 11 FLTDIV3
15–14 FLTH	针对 PTH 输入的滤波器选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
13–12 FLTG	针对 PTG 输入的滤波器选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
11–10 FLTF	针对 PTF 输入的滤波器选择 00 BUSCLK

下一页继续介绍此表...

**PORT\_IOFLTO 字段描述 (继续)**

字段	描述
	01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
9–8 FLTE	针对 PTD 输入的滤波器选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
7–6 FLTD	针对 PTD 输入的滤波器选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
5–4 FLTC	针对 PTC 输入的滤波器选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
3–2 FLTB	针对 PTB 输入的滤波器选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
FLTA	针对 PTA 输入的滤波器选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3

## 11.7.2 端口滤波寄存器 1 (PORT\_IOFLT1)

这个寄存器是配置输入引脚的滤波器，可以配置高或低电平毛刺的脉宽阈值。脉宽比所选时钟周期更短的毛刺将被滤除；脉宽比所选时钟周期长两倍以上的毛刺将不会被滤除，并且会进入内部电路。

地址: 4004\_9000h 基准 + 4h 偏移 = 4004\_9004h

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	0																
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	FLTI2C1	FLTI2C0	FLTPWT	FLTFTM1	FLTFTM0	FLTIRQ	0	FLTI									
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

PORT\_IOFLT1 字段描述

字段	描述
31–16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15–14 FLTI2C1	针对 SCL1/SDA1 输入的滤波器选择  00 无滤波器 01 选择 FLTDIV1 10 选择 FLTDIV2 11 选择 BUSCLK
13–12 FLTI2C0	针对 SCL0/SDA0 输入的滤波器选择  00 无滤波器 01 选择 FLTDIV1 10 选择 FLTDIV2 11 选择 BUSCLK
11–10 FLTPWT	针对 PWT_IN1/PWT_IN0 输入的滤波器选择  00 无滤波器 01 选择 FLTDIV1 10 选择 FLTDIV2 11 选择 FLTDIV3
9–8 FLTFTM1	针对 FTM1CH0/FTM1CH1 输入的滤波器选择  00 无滤波器 01 选择 FLTDIV1 10 选择 FLTDIV2 11 选择 FLTDIV3
7–6 FLTFTM0	针对 FTM0CH0/FTM0CH1 输入的滤波器选择  00 无滤波器

下一页继续介绍此表...

## PORT\_IOFLT1 字段描述 (继续)

字段	描述
	01 选择 FLTDIV1 10 选择 FLTDIV2 11 选择 FLTDIV3
5–4 FLTIRQ	针对 IRQ 输入的滤波器选择 00 无滤波器 01 选择 FLTDIV1，在 Stop 模式下会自动切换到 FLTDIV3。 10 选择 FLTDIV2，在 Stop 模式下会自动切换到 FLTDIV3。 11 FLTDIV3
3–2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
FLTI	针对 PTI 输入的滤波器选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3

## 11.7.3 端口上拉使能寄存器 0 (PORT\_PUE0)

地址: 4004\_9000h 基准 + 8h 偏移 = 4004\_9008h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
W	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
复位	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
W	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## PORT\_PUE0 字段描述

字段	描述
31 PTDPE7	端口 D 位 7 上拉使能 该字段决定是否使能端口 D 的内部上拉电阻。对于配置为输出或高阻态的端口 D 引脚，该字段无效。 0 端口 D 位 7 上拉禁用。 1 端口 D 位 7 上拉使能。

下一页继续介绍此表...

## PORT\_PUE0 字段描述 (继续)

字段	描述
30 PTDPE6	端口 D 位 6 上拉使能  该字段决定是否使能端口 D 的内部上拉电阻。对于配置为输出或高阻态的端口 D 引脚，该字段无效。  0 端口 D 位 6 上拉禁用。 1 端口 D 位 6 上拉使能。
29 PTDPE5	端口 D 位 5 上拉使能  该字段决定是否使能端口 D 的内部上拉电阻。对于配置为输出或高阻态的端口 D 引脚，该字段无效。  0 端口 D 位 5 上拉禁用。 1 端口 D 位 5 上拉使能。
28 PTDPE4	端口 D 位 4 上拉使能  该位决定是否使能端口 D 的内部上拉电阻。对于配置为输出或高阻态的端口 D 引脚，这些位无效。  0 端口 D 位 4 上拉禁用。 1 端口 D 位 4 上拉使能。
27 PTDPE3	端口 D 位 3 上拉使能  该字段决定是否使能端口 D 的内部上拉电阻。对于配置为输出或高阻态的端口 D 引脚，该字段无效。  0 端口 D 位 3 上拉禁用。 1 端口 D 位 3 上拉使能。
26 PTDPE2	端口 D 位 2 上拉使能  该字段决定是否使能端口 D 的内部上拉电阻。对于配置为输出或高阻态的端口 D 引脚，该字段无效。  0 端口 D 位 2 上拉禁用。 1 端口 D 位 2 上拉使能。
25 PTDPE1	端口 D 位 1 上拉使能  该字段决定是否使能端口 D 的内部上拉电阻。对于配置为输出或高阻态的端口 D 引脚，该字段无效。  0 端口 D 位 1 上拉禁用。 1 端口 D 位 1 上拉使能。
24 PTDPE0	端口 D 位 0 上拉使能  该字段决定是否使能端口 D 的内部上拉电阻。对于配置为输出或高阻态的端口 D 引脚，该字段无效。  0 端口 D 位 0 上拉禁用。 1 端口 D 位 0 上拉使能。
23 PTCPE7	端口 C 位 7 上拉使能  该字段决定是否使能端口 C 的内部上拉电阻。对于配置为输出或高阻态的端口 C 引脚，该字段无效。  0 端口 C 位 7 上拉禁用。 1 端口 C 位 7 上拉使能。
22 PTCPE6	端口 C 位 6 上拉使能  该字段决定是否使能端口 C 的内部上拉电阻。对于配置为输出或高阻态的端口 C 引脚，该字段无效。

下一页继续介绍此表...

## PORT\_PUE0 字段描述 (继续)

字段	描述
	0 端口 C 位 6 上拉禁用。 1 端口 C 位 6 上拉使能。
21 PTCPE5	端口 C 位 5 上拉使能  该字段决定是否使能端口 C 的内部上拉电阻。对于配置为输出或高阻态的端口 C 引脚，该字段无效。  0 端口 C 位 5 上拉禁用。 1 端口 C 位 5 上拉使能。
20 PTCPE4	端口 C 位 4 上拉使能  该字段决定是否使能端口 C 的内部上拉电阻。对于配置为输出或高阻态的端口 C 引脚，该字段无效。  0 端口 C 位 4 上拉禁用。 1 端口 C 位 4 上拉使能。
19 PTCPE3	端口 C 位 3 上拉使能  该字段决定是否使能端口 C 的内部上拉电阻。对于配置为输出或高阻态的端口 C 引脚，该字段无效。  0 端口 C 位 3 上拉禁用。 1 端口 C 位 3 上拉使能。
18 PTCPE2	端口 C 位 2 上拉使能  该字段决定是否使能端口 C 的内部上拉电阻。对于配置为输出或高阻态的端口 C 引脚，该字段无效。  0 端口 C 位 2 上拉禁用。 1 端口 C 位 2 上拉使能。
17 PTCPE1	端口 C 位 1 上拉使能  该字段决定是否使能端口 C 的内部上拉电阻。对于配置为输出或高阻态的端口 C 引脚，该字段无效。  0 端口 C 位 1 上拉禁用。 1 端口 C 位 1 上拉使能。
16 PTCPE0	端口 C 位 0 上拉使能  该字段决定是否使能端口 C 的内部上拉电阻。对于配置为输出或高阻态的端口 C 引脚，该字段无效。  0 端口 C 位 0 上拉禁用。 1 端口 C 位 0 上拉使能。
15 PTBPE7	端口 B 位 7 上拉使能  该字段决定是否使能端口 B 的内部上拉电阻。对于配置为输出或高阻态的端口 B 引脚，该字段无效。  0 端口 B 位 7 上拉禁用。 1 端口 B 位 7 上拉使能。
14 PTBPE6	端口 B 位 6 上拉使能  该字段决定是否使能端口 B 的内部上拉电阻。对于配置为输出或高阻态的端口 B 引脚，该字段无效。  0 端口 B 位 6 上拉禁用。 1 端口 B 位 6 上拉使能。

下一页继续介绍此表...

## PORT\_PUE0 字段描述 (继续)

字段	描述
13 PTBPE5	<p>端口 B 位 5 上拉使能</p> <p>该字段决定是否使能端口 B 的内部上拉电阻。对于配置为输出或高阻态的端口 B 引脚，该字段无效。</p> <p>0 端口 B 位 5 上拉禁用。 1 端口 B 位 5 上拉使能。</p>
12 PTBPE4	<p>端口 B 位 4 上拉使能</p> <p>该字段决定是否使能端口 B 的内部上拉电阻。对于配置为输出或高阻态的端口 B 引脚，该字段无效。</p> <p>0 端口 B 位 4 上拉禁用。 1 端口 B 位 4 上拉使能。</p>
11 PTBPE3	<p>端口 B 位 3 上拉使能</p> <p>该字段决定是否使能端口 B 的内部上拉电阻。对于配置为输出或高阻态的端口 B 引脚，该字段无效。</p> <p>0 端口 B 位 3 上拉禁用。 1 端口 B 位 3 上拉使能。</p>
10 PTBPE2	<p>端口 B 位 2 上拉使能</p> <p>该字段决定是否使能端口 B 的内部上拉电阻。对于配置为输出或高阻态的端口 B 引脚，该字段无效。</p> <p>0 端口 B 位 2 上拉禁用。 1 端口 B 位 2 上拉使能。</p>
9 PTBPE1	<p>端口 B 位 1 上拉使能</p> <p>该字段决定是否使能端口 B 的内部上拉电阻。对于配置为输出或高阻态的端口 B 引脚，该字段无效。</p> <p>0 端口 B 位 1 上拉禁用。 1 端口 B 位 1 上拉使能。</p>
8 PTBPE0	<p>端口 B 位 0 上拉使能</p> <p>该字段决定是否使能端口 B 的内部上拉电阻。对于配置为输出或高阻态的端口 B 引脚，该字段无效。</p> <p>0 端口 B 位 0 上拉禁用。 1 端口 B 位 0 上拉使能。</p>
7 PTAPE7	<p>端口 A 位 7 上拉使能</p> <p>该字段决定是否使能端口 A 的内部上拉电阻。对于配置为输出或高阻态的端口 A 引脚，该字段无效。</p> <p>0 端口 A 位 7 上拉禁用。 1 端口 A 位 7 上拉使能。</p>
6 PTAPE6	<p>端口 A 位 6 上拉使能</p> <p>该字段决定是否使能端口 A 的内部上拉电阻。对于配置为输出或高阻态的端口 A 引脚，该字段无效。</p> <p>0 端口 A 位 6 上拉禁用。 1 端口 A 位 6 上拉使能。</p>
5 PTAPE5	<p>端口 A 位 5 上拉使能</p> <p>该字段决定是否使能端口 A 的内部上拉电阻。对于配置为输出或高阻态的端口 A 引脚，该字段无效。</p>

下一页继续介绍此表...

## PORT\_PUE0 字段描述 (继续)

字段	描述
	<p>0 端口 A 位 5 上拉禁用。 1 端口 A 位 5 上拉使能。</p>
4 PTAPE4	<p>端口 A 位 4 上拉使能 该字段决定是否使能端口 A 的内部上拉电阻。对于配置为输出或高阻态的端口 A 引脚，该字段无效。</p> <p>0 端口 A 位 4 上拉禁用。 1 端口 A 位 4 上拉使能。</p>
3 PTAPE3	<p>端口 A 位 3 上拉使能 该字段决定是否使能端口 A 的内部上拉电阻。对于配置为输出或高阻态的端口 A 引脚，该字段无效。</p> <p>注：当此引脚被配置为 IIC 功能且输出高电平时，内部上拉电阻在 PTAP3 置位时保持激活状态。当输出为低电平时，它会自动禁用以降低功耗。</p> <p>0 端口 A 位 3 上拉禁用。 1 端口 A 位 3 上拉使能。</p>
2 PTAPE2	<p>端口 A 位 2 上拉使能 该字段决定是否使能端口 A 的内部上拉电阻。对于配置为输出或高阻态的端口 A 引脚，该字段无效。</p> <p>注：当此引脚被配置为 IIC 功能且输出高电平时，内部上拉电阻在 PTAP2 置位时保持激活状态。当输出为低电平时，它会自动禁用以降低功耗。</p> <p>0 端口 A 位 2 上拉禁用。 1 端口 A 位 2 上拉使能。</p>
1 PTAPE1	<p>端口 A 位 1 上拉使能 该字段决定是否使能端口 A 的内部上拉电阻。对于配置为输出或高阻态的端口 A 引脚，该字段无效。</p> <p>0 端口 A 位 1 上拉禁用。 1 端口 A 位 1 上拉使能。</p>
0 PTAPE0	<p>端口 A 位 0 上拉使能 该字段决定是否使能端口 A 的内部上拉电阻。对于配置为输出或高阻态的端口 A 引脚，该字段无效。</p> <p>0 端口 A 位 0 上拉禁用。 1 端口 A 位 0 上拉使能。</p>

### 11.7.4 端口上拉使能寄存器 1 (PORT\_PUE1)

地址: 4004\_9000h 基准 + Ch 偏移 = 4004\_900Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PTHPE7	PTHPE6	PTHPE5	PTHPE4	PTHPE3	PTHPE2	PTHPE1	PTHPE0	PTGPE7	PTGPE6	PTGPE5	PTGPE4	PTGPE3	PTGPE2	PTGPE1	PTGPE0
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PORT\_PUE1 字段描述

字段	描述
31 PTHPE7	端口 H 位 7 上拉使能 该字段决定是否使能端口 H 的内部上拉电阻。对于配置为输出或高阻态的端口 H 引脚，该字段无效。 0 端口 H 位 7 上拉禁用。 1 端口 H 位 7 上拉使能。
30 PTHPE6	端口 H 位 6 上拉使能 该字段决定是否使能端口 H 的内部上拉电阻。对于配置为输出或高阻态的端口 H 引脚，该字段无效。 0 端口 H 位 6 上拉禁用。 1 端口 H 位 6 上拉使能。
29 PTHPE5	端口 H 位 5 上拉使能 该字段决定是否使能端口 H 的内部上拉电阻。对于配置为输出或高阻态的端口 H 引脚，该字段无效。 0 端口 H 位 5 上拉禁用。 1 端口 H 位 5 上拉使能。
28 PTHPE4	端口 H 位 4 上拉使能 该字段决定是否使能端口 H 的内部上拉电阻。对于配置为输出或高阻态的端口 H 引脚，该字段无效。 0 端口 H 位 4 上拉禁用。 1 端口 H 位 4 上拉使能。
27 PTHPE3	端口 H 位 3 上拉使能 该字段决定是否使能端口 H 的内部上拉电阻。对于配置为输出或高阻态的端口 H 引脚，该字段无效。 0 端口 H 位 3 上拉禁用。 1 端口 H 位 3 上拉使能。

下一页继续介绍此表...

## PORT\_PUE1 字段描述 (继续)

字段	描述
26 PTHPE2	端口 H 位 2 上拉使能 该字段决定是否使能端口 H 的内部上拉电阻。对于配置为输出或高阻态的端口 H 引脚，该字段无效。 0 端口 H 位 2 上拉禁用。 1 端口 H 位 2 上拉使能。
25 PTHPE1	端口 H 位 1 上拉使能 该字段决定是否使能端口 H 的内部上拉电阻。对于配置为输出或高阻态的端口 H 引脚，该字段无效。 0 端口 H 位 1 上拉禁用。 1 端口 H 位 1 上拉使能。
24 PTHPE0	端口 H 位 0 上拉使能 该字段决定是否使能端口 H 的内部上拉电阻。对于配置为输出或高阻态的端口 H 引脚，该字段无效。 0 端口 H 位 0 上拉禁用。 1 端口 H 位 0 上拉使能。
23 PTGPE7	端口 G 位 7 上拉使能 该字段决定是否使能端口 G 的内部上拉电阻。对于配置为输出或高阻态的端口 G 引脚，该字段无效。 0 端口 G 位 7 上拉禁用。 1 端口 G 位 7 上拉使能。
22 PTGPE6	端口 G 位 6 上拉使能 该字段决定是否使能端口 G 的内部上拉电阻。对于配置为输出或高阻态的端口 G 引脚，该字段无效。 0 端口 G 位 6 上拉禁用。 1 端口 G 位 6 上拉使能。
21 PTGPE5	端口 G 位 5 上拉使能 该字段决定是否使能端口 G 的内部上拉电阻。对于配置为输出或高阻态的端口 G 引脚，该字段无效。 0 端口 G 位 5 上拉禁用。 1 端口 G 位 5 上拉使能。
20 PTGPE4	端口 G 位 4 上拉使能 该字段决定是否使能端口 G 的内部上拉电阻。对于配置为输出或高阻态的端口 G 引脚，该字段无效。 0 端口 G 位 4 上拉禁用。 1 端口 G 位 4 上拉使能。
19 PTGPE3	端口 G 位 3 上拉使能 该字段决定是否使能端口 G 的内部上拉电阻。对于配置为输出或高阻态的端口 G 引脚，该字段无效。 0 端口 G 位 3 上拉禁用。 1 端口 G 位 3 上拉使能。
18 PTGPE2	端口 G 位 2 上拉使能 该字段决定是否使能端口 G 的内部上拉电阻。对于配置为输出或高阻态的端口 G 引脚，该字段无效。

下一页继续介绍此表...

## PORT\_PUE1 字段描述 (继续)

字段	描述
	0 端口 G 位 2 上拉禁用。 1 端口 G 位 2 上拉使能。
17 PTGPE1	端口 G 位 1 上拉使能  该字段决定是否使能端口 G 的内部上拉电阻。对于配置为输出或高阻态的端口 G 引脚，该字段无效。  0 端口 G 位 1 上拉禁用。 1 端口 G 位 1 上拉使能。
16 PTGPE0	端口 G 位 0 上拉使能  该字段决定是否使能端口 G 的内部上拉电阻。对于配置为输出或高阻态的端口 G 引脚，该字段无效。  0 端口 G 位 0 上拉禁用。 1 端口 G 位 0 上拉使能。
15 PTFPE7	端口 F 位 7 上拉使能  该字段决定是否使能端口 F 的内部上拉电阻。对于配置为输出或高阻态的端口 F 引脚，该字段无效。  0 端口 F 位 7 上拉禁用。 1 端口 F 位 7 上拉使能。
14 PTFPE6	端口 F 位 6 上拉使能  该字段决定是否使能端口 F 的内部上拉电阻。对于配置为输出或高阻态的端口 F 引脚，该字段无效。  0 端口 F 位 6 上拉禁用。 1 端口 F 位 6 上拉使能。
13 PTFPE5	端口 F 位 5 上拉使能  该字段决定是否使能端口 F 的内部上拉电阻。对于配置为输出或高阻态的端口 F 引脚，该字段无效。  0 端口 F 位 5 上拉禁用。 1 端口 F 位 5 上拉使能。
12 PTFPE4	端口 F 位 4 上拉使能  该字段决定是否使能端口 F 的内部上拉电阻。对于配置为输出或高阻态的端口 F 引脚，该字段无效。  0 端口 F 位 4 上拉禁用。 1 端口 F 位 4 上拉使能。
11 PTFPE3	端口 F 位 3 上拉使能  该字段决定是否使能端口 F 的内部上拉电阻。对于配置为输出或高阻态的端口 F 引脚，该字段无效。  0 端口 F 位 3 上拉禁用。 1 端口 F 位 3 上拉使能。
10 PTFPE2	端口 F 位 2 上拉使能  该字段决定是否使能端口 F 的内部上拉电阻。对于配置为输出或高阻态的端口 F 引脚，该字段无效。  0 端口 F 位 2 上拉禁用。 1 端口 F 位 2 上拉使能。

下一页继续介绍此表...

## PORT\_PUE1 字段描述 (继续)

字段	描述
9 PTFPE1	端口 F 位 1 上拉使能 该字段决定是否使能端口 F 的内部上拉电阻。对于配置为输出或高阻态的端口 F 引脚，该字段无效。 0 端口 F 位 1 上拉禁用。 1 端口 F 位 1 上拉使能。
8 PTFPE0	端口 F 位 0 上拉使能 该字段决定是否使能端口 F 的内部上拉电阻。对于配置为输出或高阻态的端口 F 引脚，该字段无效。 0 端口 F 位 0 上拉禁用。 1 端口 F 位 0 上拉使能。
7 PTEPE7	端口 E 位 7 上拉使能 该字段决定是否使能端口 E 的内部上拉电阻。对于配置为输出或高阻态的端口 E 引脚，该字段无效。 0 端口 E 位 7 上拉禁用。 1 端口 E 位 7 上拉使能。
6 PTEPE6	端口 E 位 6 上拉使能 该字段决定是否使能端口 E 的内部上拉电阻。对于配置为输出或高阻态的端口 E 引脚，该字段无效。 0 端口 E 位 6 上拉禁用。 1 端口 E 位 6 上拉使能。
5 PTEPE5	端口 E 位 5 上拉使能 该字段决定是否使能端口 E 的内部上拉电阻。对于配置为输出或高阻态的端口 E 引脚，该字段无效。 0 端口 E 位 5 上拉禁用。 1 端口 E 位 5 上拉使能。
4 PTEPE4	端口 E 位 4 上拉使能 该字段决定是否使能端口 E 的内部上拉电阻。对于配置为输出或高阻态的端口 E 引脚，该字段无效。 0 端口 E 位 4 上拉禁用。 1 端口 E 位 4 上拉使能。
3 PTEPE3	端口 E 位 3 上拉使能 该字段决定是否使能端口 E 的内部上拉电阻。对于配置为输出或高阻态的端口 E 引脚，该字段无效。 0 端口 E 位 3 上拉禁用。 1 端口 E 位 3 上拉使能。
2 PTEPE2	端口 E 位 2 上拉使能 该字段决定是否使能端口 E 的内部上拉电阻。对于配置为输出或高阻态的端口 E 引脚，该字段无效。 0 端口 E 位 2 上拉禁用。 1 端口 E 位 2 上拉使能。
1 PTEPE1	端口 E 位 1 上拉使能 该字段决定是否使能端口 E 的内部上拉电阻。对于配置为输出或高阻态的端口 E 引脚，该字段无效。

下一页继续介绍此表...

## PORT\_PUE1 字段描述 (继续)

字段	描述
	0 端口 E 位 1 上拉禁用。 1 端口 E 位 1 上拉使能。
0 PTEPE0	端口 E 位 0 上拉使能  该字段决定是否使能端口 E 的内部上拉电阻。对于配置为输出或高阻态的端口 E 引脚，该字段无效。  0 端口 E 位 0 上拉禁用。 1 端口 E 位 0 上拉使能。

## 11.7.5 端口上拉使能寄存器 2 (PORT\_PUE2)

地址: 4004\_9000h 基准 + 10h 偏移 = 4004\_9010h

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									0		PTIPE6	PTIPE5	PTIPE4	PTIPE3	PTIPE2	PTIPE1	PTIPE0
W										0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

## PORT\_PUE2 字段描述

字段	描述
31–7 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
6 PTIPE6	端口 I 位 6 上拉使能  该字段决定是否使能端口 I 的内部上拉电阻。对于配置为输出或高阻态的端口 I 引脚，该字段无效。  0 端口 I 位 6 上拉禁用。 1 端口 I 位 6 上拉使能。
5 PTIPE5	端口 I 位 5 上拉使能  该字段决定是否使能端口 I 的内部上拉电阻。对于配置为输出或高阻态的端口 I 引脚，该字段无效。  0 端口 I 位 5 上拉禁用。 1 端口 I 位 5 上拉使能。
4 PTIPE4	端口 I 位 4 上拉使能  该字段决定是否使能端口 I 的内部上拉电阻。对于配置为输出或高阻态的端口 I 引脚，该字段无效。

下一页继续介绍此表...

## PORT\_PUE2 字段描述 (继续)

字段	描述
	0 端口 I 位 4 上拉禁用。 1 端口 I 位 4 上拉使能。
3 PTIPE3	端口 I 位 3 上拉使能  该字段决定是否使能端口 I 的内部上拉电阻。对于配置为输出或高阻态的端口 I 引脚，该字段无效。  0 端口 I 位 3 上拉禁用。 1 端口 I 位 3 上拉使能。
2 PTIPE2	端口 I 位 2 上拉使能  该字段决定是否使能端口 I 的内部上拉电阻。对于配置为输出或高阻态的端口 I 引脚，该字段无效。  0 端口 I 位 2 上拉禁用。 1 端口 I 位 2 上拉使能。
1 PTIPE1	端口 I 位 1 上拉使能  该字段决定是否使能端口 I 的内部上拉电阻。对于配置为输出或高阻态的端口 I 引脚，该字段无效。  0 端口 I 位 1 上拉禁用。 1 端口 I 位 1 上拉使能。
0 PTIPE0	端口 I 位 0 上拉使能  该字段决定是否使能端口 I 的内部上拉电阻。对于配置为输出或高阻态的端口 I 引脚，该字段无效。  0 端口 I 位 0 上拉禁用。 1 端口 I 位 0 上拉使能。

## 11.7.6 端口大电流驱动使能寄存器 (PORT\_HDRVE)

地址: 4004\_9000h 基准 + 14h 偏移 = 4004\_9014h

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R					0				PTH1	PTH0	PTE1	PTE0	PTD1	PTD0	PTB5	PTB4	
W										0	0	0	0	0	0	0	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

## PORT\_HDRVE 字段描述

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 PTH1	PTH1 大电流驱动能力

下一页继续介绍此表...

## PORT\_HDRVE 字段描述 (继续)

字段	描述
	该字段使能 PTH1 的大电流驱动能力。 0 PTH1 禁用以提供大电流驱动能力。 1 PTH1 使能以提供大电流驱动能力。
6 PTH0	PTH0 大电流驱动能力 该字段使能 PTH0 的大电流驱动能力。 0 PTH0 禁用以提供大电流驱动能力。 1 PTH0 使能以提供大电流驱动能力。
5 PTE1	PTE1 大电流驱动能力 该字段使能 PTE1 的大电流驱动能力。 0 PTE1 禁用以提供大电流驱动能力。 1 PTE1 使能以提供大电流驱动能力。
4 PTE0	PTE0 大电流驱动能力 该字段使能 PTE0 的大电流驱动能力。 0 PTE0 禁用以提供大电流驱动能力。 1 PTE0 使能以提供大电流驱动能力。
3 PTD1	PTD1 大电流驱动能力 该字段使能 PTD1 的大电流驱动能力。 0 PTD1 禁用以提供大电流驱动能力。 1 PTD1 使能以提供大电流驱动能力。
2 PTD0	PTD0 大电流驱动能力 该字段使能 PTD0 的大电流驱动能力。 0 PTD0 禁用以提供大电流驱动能力。 1 PTD0 使能以提供大电流驱动能力。
1 PTB5	PTB5 大电流驱动能力 该字段使能 PTB0 的大电流驱动能力。 0 PTB5 禁用以提供大电流驱动能力。 1 PTB5 使能以提供大电流驱动能力。
0 PTB4	PTB4 大电流驱动能力 该字段使能 PTB4 的大电流驱动能力。 0 PTB4 禁用以提供大电流驱动能力。 1 PTB4 使能以提供大电流驱动能力。



# 第 12 章

## 系统集成模块 (SIM)

### 12.1 简介

系统集成模块(SIM)提供系统控制和芯片配置寄存器。

#### 12.1.1 特性

SIM 模块的特性列示如下。

- 复位状态和器件 ID 信息
- 系统互连配置和特殊引脚使能
- 引脚重映射控制
- 系统时钟选通控制和时钟分频

### 12.2 存储器映像和寄存器定义

SIM 模块包含许多字段，用来为不同的模块时钟选择时钟源和分频器。

**SIM 存储器映射**

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4004_8000	系统复位状态和 ID 寄存器 (SIM_SRSID)	32	R	参见章节	<a href="#">12.2.1/166</a>
4004_8004	系统选项寄存器 0 (SIM_SOPT0)	32	R/W	参见章节	<a href="#">12.2.2/169</a>
4004_8008	系统选项寄存器 (SIM_SOPT1)	32	R/W	0000_0000h	<a href="#">12.2.3/172</a>
4004_800C	引脚选择寄存器 0 (SIM_PINSEL0)	32	R/W	0000_0000h	<a href="#">12.2.4/174</a>
4004_8010	引脚选择寄存器 1 (SIM_PINSEL1)	32	R/W	0000_0000h	<a href="#">12.2.5/176</a>
4004_8014	系统时钟选通控制寄存器 (SIM_SCGC)	32	R/W	0000_3000h	<a href="#">12.2.6/178</a>
4004_8018	通用唯一标识符低位寄存器 (SIM_UUIDL)	32	R	未定义	<a href="#">12.2.7/182</a>
4004_801C	通用唯一标识符中低位寄存器 (SIM_UUIDML)	32	R	未定义	<a href="#">12.2.8/182</a>

下一页继续介绍此表...

**SIM 存储器映射 (继续)**

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4004_8020	通用唯一标识符中高位寄存器 (SIM_UUIDMH)	32	R	未定义	12.2.9/183
4004_8024	时钟分频器寄存器 (SIM_CLKDIV)	32	R/W	参见章节	12.2.10/183

**12.2.1 系统复位状态和 ID 寄存器 (SIM\_SRSID)**

地址: 4004\_8000h 基准 + 0h 偏移 = 4004\_8000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FAMID				SUBFAMID				RevID				PINID			
W																
复位	0	0	0	0	0	1	1	0	*	*	*	*	*	*	*	*
LVD	0	0	0	0	0	1	1	0	*	*	*	*	*	*	*	*
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SACKERR	0	MIDMAP	SW	LOCKUP	0	POR	PIN	WDOG	0	LOC	LVD	0		
W																
复位	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
LVD	0	0	0	0	0	0	0	0	u*	0	0	0	0	0	1	0

\* 注:

- RevID 字段: Decided by device revision number.
- PINID 字段: Decided by device pin number.
- u = 不受复位影响。

**SIM\_SRSID 字段描述**

字段	描述
31–28 FAMID	Kinetis 系列 ID

下一页继续介绍此表...

**SIM\_SRSID 字段描述 (继续)**

字段	描述
	0000 KE0x 系列。 其他 保留。
27–24 SUBFAMID	Kinetis 子系列 ID  0100 KEx4 子系列 0110 KEx6 子系列 其他 保留
23–20 RevID	器件版本号
19–16 PINID	器件引脚 ID  0000 8 引脚 0001 16 引脚 0010 20 引脚 0011 24 引脚 0100 32 引脚 0101 44 引脚 0110 48 引脚 0111 64 引脚 1000 80 引脚 1010 100 引脚 其他 保留
15–14 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
13 SACKERR	Stop 模式应答错误复位  指示尝试进入 Stop 模式后，已因一个或多个 IIC 未能在约一秒时间内对此作出应答而引起复位。  0 复位不是因为外设未能对尝试进入 Stop 模式作出应答而引起。 1 复位是因为外设未能对尝试进入 Stop 模式作出应答而引起。
12 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
11 MDMAP	MDM-AP 系统复位请求  指示复位是由主调试器系统设置 MDM-AP 控制寄存器中的系统复位请求字段而引起。  0 复位不是由主调试器设置系统复位请求位而引起。 1 复位是由主调试器系统设置系统复位请求位而引起。
10 SW	软件  指示复位是由软件对 ARM 内核中的应用中断和复位控制寄存器的 SYSRESETREQ 位进行设置而引起。  0 复位不是由软件设置 SYSRESETREQ 位引起。 1 复位是由软件设置 SYSRESETREQ 位引起。
9 LOCKUP	内核锁定  指示 LOCKUP 事件的 ARM 内核指示已引起复位。

下一页继续介绍此表...

**SIM\_SRSID 字段描述 (继续)**

字段	描述
	0 复位不是由内核 LOCKUP 事件引起。 1 复位是由内核 LOCKUP 事件引起。
8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 POR	<b>上电复位</b> 通过上电检测逻辑引起复位。当内部供电电压斜升时，低压复位(LVR)状态字段也在此时置位，指示已因内部供电电压低于 LVR 阈值而发生复位。 注：该位在 POR 时复位为 1，在 LVR 时复位为不定值，在任何其他情况下复位为 0。 0 复位不是由 POR 引起。 1 POR 引起复位。
6 PIN	<b>外部复位引脚</b> 通过外部复位引脚的有效低电平引起复位。 0 复位不是由外部复位引脚引起。 1 复位来源于外部复位引脚。
5 WDOG	<b>WDOG</b> 通过 WDOG 定时器定时溢出引起复位。该复位源可通过设置 WDOG_CS1[EN] = 0 加以阻断。 0 复位不是由 WDOG 定时溢出引起。 1 复位是由 WDOG 定时溢出引起。
4–3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 LOC	<b>内部时钟源模块复位</b> 通过 ICS 模块复位引起复位。 0 复位不是由 ICS 模块引起。 1 复位是由 ICS 模块引起。
1 LVD	<b>低电压检测</b> 如果 PMC_SPMSC1[LVDRE]在 Run 模式下置位或 PMC_SPMSC1[LVDRE]和 PMC_SPMSC1[LVDSE]在 Stop 模式下均置位，并且供电电压下降低于 LVD 触发电压，那么将发生 LVD 复位。该字段也会因 POR 而置位。 注：该字段在 POR 和 LVR 时复位为 1，在其他复位时复位为 0。 0 复位不是由 LVD 跳变或 POR 引起。 1 复位是由 LVD 跳变或 POR 引起。
0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

## 12.2.2 系统选项寄存器 0 (SIM\_SOPT0)

### 注

RSTPE 和 NMIE 在每次复位时只能写入一次。

地址: 4004\_8000h 基准 + 4h 偏移 = 4004\_8004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									DLYACT							
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
POR/ LVD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXDME	0	0	RXDCE	ACIC	RTCC	RXDDE		0		0					0
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	1	u*	u*	0
POR/ LVD	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

\* 注:

- u = 不受复位影响。

### SIM\_SOPT0 字段描述

字段	描述
31-24 DELAY	FTM2 触发延迟

下一页继续介绍此表...

**SIM\_SOPT0 字段描述 (继续)**

字段	描述
	指定将 1 写入 ADHWT 时从 FTM2 初始或匹配触发到 ADC 硬件触发的延迟。该 8 位模数值允许 0 到 255 的延迟，具体取决于 BUSREF 时钟设置。这是一个一次性计数器，当触发到达时开始计数，当计数器值达到所定义的模数值时停止计数。
23 DLYACT	<p>FTM2 触发延迟有效</p> <p>该只读字段指定有关 FTM2 初始或匹配延迟是否有效的状态。该字段在 FTM2 触发到达且延迟计数器正在计数时置位。否则，该字段清零。</p> <p>0 延迟无效。 1 延迟有效。</p>
22–20 ADHWT	<p>ADC 硬件触发源</p> <p>选择 ADC 硬件触发源。所有触发源都是在上升沿开始 ADC 转换。</p> <p>000 RTC 溢出用作 ADC 硬件触发 001 FTMO 用作 ADC 硬件触发 010 带 8 位可编程计数器延迟的 FTM2 初始触发 011 带 8 位可编程计数器延迟的 FTM2 匹配触发 100 PIT 通道 0 溢出用作 ADC 硬件触发 101 PIT 通道 1 溢出用作 ADC 硬件触发 110 ACMP0 输出用作 ADC 硬件触发 111 ACMP1 输出用作 ADC 硬件触发</p>
19 CLKOE	<p>总线时钟输出使能</p> <p>在上使能总线时钟输出</p> <p>0 总线时钟输出在 PTH2 上禁用。 1 总线时钟输出在 PTH2 上使能。</p>
18–16 BUSREF	<p>总线时钟输出选择</p> <p>通过可选预分频器在使能总线时钟输出。</p> <p>000 总线 001 总线 2 分频 010 总线 4 分频 011 总线 8 分频 100 总线 16 分频 101 总线 32 分频 110 总线 64 分频 111 总线 128 分频</p>
15 TXDME	<p>UART0_TX 调制选择</p> <p>使能由 FTMO 通道 0 调制的 UART0_TX 输出。</p> <p>0 UART0_TX 输出直接连接到引出线。 1 UART0_TX 输出到引出线前由 FTMO 通道 0 调制。</p>
14 FTMSYNC	<p>FTM2 同步选择</p> <p>将 1 写入该字段时生成 FTM2 模块的 PWM 同步触发。</p>

下一页继续介绍此表...

**SIM\_SOPT0 字段描述 (继续)**

字段	描述
	0 未触发任何同步。 1 生成 FTM2 模块的 PWM 同步触发。
13 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
12 RXDCE	<b>UART0_RX 捕捉选择</b> 使能 UART0_RX 由 FTM0 通道 1 捕捉。 0 UART0_RX 输入信号仅连接到 UART0 模块。 1 UART0_RX 输入信号连接到 UART0 模块和 FTM0 通道 1。
11 ACIC	<b>模拟比较器至输入捕捉使能</b> 将 ACMP0 的输出连接到 FTM1 输入通道 0。 0 ACMP0 输出未连接到 FTM1 输入通道 0。 1 ACMP0 输出连接到 FTM1 输入通道 0。
10 RTCC	<b>实时计数器捕捉</b> 允许实时计数器(RTC)溢出由 FTM1 通道 1 捕捉。 0 RTC 溢出未连接到 FTM1 输入通道 1。 1 RTC 溢出连接到 FTM1 输入通道 1。
9–8 RXDFE	<b>UART0 RxD 滤波器选择</b> 使能 UART0 RxD 输入由 ACMP 滤波。该功能使能时，任何具有 ACMP 输入标记的信号都可被视作 UART0。 00 RXD0 输入信号直接连接到 UART0 模块。 01 RXD0 输入信号由 ACMP0 滤波，然后注入 UART0。 10 RXD0 输入信号由 ACMP1 滤波，然后注入 UART0。 11 保留。
7–6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5 ACTRG	<b>ACMP 触发 FTM2 选择</b> 选择两个 ACMP 输出作为 FTM2 的触发 0 输入。 0 ACMP0 输出 1 ACMP1 输出
4 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
3 SWDE	<b>单线调试端口引脚使能</b> 使能 PTA4/KBI0_P4/ACMP0_OUT/SWD_DIO 引脚用作 SWD_DIO，使能 PTC4/KBI0_P20/RTC_CLKOUT/FTM1_CH0/ACMP0_IN2/SWD_CLK 引脚用作 SWD_CLK。清零时，两个引脚用作 PTA4 和 PTC4。该引脚在任何 MCU 复位之后默认用作 SWD_DIO 和 SWD_CLK。

下一页继续介绍此表...

**SIM\_SOPT0 字段描述 (继续)**

字段	描述
	<p>0 PTA4/KBI0_P4/ACMP0_OUT/SWD_DIO 作为 PTA4 或 ACMP0_OUT 功能 , PTC4/KBI0_P20/RTC_CLKOUT/FTM1_CH0/ACMP0_IN2/SWD_CLK 用作 PTC4、KBI0_P20、RTC_CLKOUT、FTM1_CH0 或 ACMP0_IN2 功能。</p> <p>1 PTA4/KBI0_P4/ACMP0_OUT/SWD_DIO 用作 SWD_DIO 功能 , PTC4/KBI0_P20/RTC_CLKOUT/FTM1_CH0/ACMP0_IN2/SWD_CLK 用作 SWD_CLK 功能。</p>
2 RSTPE	<p>RESET 引脚使能</p> <p>在任何复位后都可对该一次写入字段进行写操作。RSTPE 置位时 , PTA5/KBI0_P5/IRQ/TCLK0/RESET 引脚用作 RESET。清零时 , 该引脚用作复用功能之一。该引脚在 MCU POR 之后默认用作 RESET。其他复位不会影响该字段。RSTPE 置位时 , RESET 上的内部上拉器件使能。</p> <p>0 PTA5/KBI0_P5/IRQ/TCLK0/RESET 引脚用作 PTA5/KBI0_P5/IRQ/TCLK0。</p> <p>1 PTA5/KBI0_P5/IRQ/TCLK0/RESET 引脚用作 RESET。</p>
1 NMIE	<p>NMI 引脚使能</p> <p>在任何复位后都可对该一次写入字段进行写操作。NMIE 置位时 , PTB4/KBI0_P12/FTM2_CH4/SPI0_MISO/ACMP1_IN2/NMI 引脚用作 NMI。清零时 , 该引脚用作复用功能之一。该引脚在 MCU POR 之后默认用作 NMI。其他复位不会影响该位。NMIE 置位时 , NMI 上的内部上拉器件使能。</p> <p>0 PTB4/KBI0_P12/FTM2_CH4/SPI0_MISO/ACMP1_IN2/NMI 引脚用作 PTB4、KBI0_P12、FTM2_CH4、SPI0_MISO 或 ACMP1_IN2。</p> <p>1 PTB4/KBI0_P12/FTM2_CH4/SPI0_MISO/ACMP1_IN2/NMI 引脚用作 NMI。</p>
0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

**12.2.3 系统选项寄存器 (SIM\_SOPT1)****注**

RSTPE 和 NMIE 在每次复位时只能写入。

地址: 4004\_8000h 基准 + 8h 偏移 = 4004\_8008h

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									0						0		
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

**SIM\_SOPT1 字段描述**

字段	描述
31–6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5–4 UARTPWTS	PWT UART RX 选择  该字段选择 PWTIN3 输入信号  00 UART0 RX 连接到 PWTIN3。 01 UART1 RX 连接到 PWTIN3。 10 UART2 RX 连接到 PWTIN3。 11 保留。
3 ACPWTS	PWT ACMP_OUT 选择  该字段选择 PWTIN2 输入信号。  0 ACMP1_OUT 连接到 PWTIN2。 1 ACMP0_OUT 连接到 PWTIN2。
2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
1 I2C0OINV	I2C0 输出反相  该字段控制 I2C0 输出的反相。  注： 该位在 I2C0 4 线式接口禁用(I2C04WEN = 0)时不起作用。  0 在 I2C0 4 线式接口配置下，SDA_OUT 和 SCL_OUT 在输出前不反相 1 在 I2C0 4 线式接口配置下，SDA_OUT 和 SCL_OUT 在输出前反相
0 I2C04WEN	I2C0 4 线式接口使能  该字段控制 I2C0 4 线式接口配置。  注： 该字段仅在 SIM_PINSEL0[I2C0PS]为 0 时起作用。  0 I2C0 4 线式接口配置禁用。 1 I2C0 4 线式接口配置使能。

## 12.2.4 引脚选择寄存器 0 (SIM\_PINSEL0)

地址: 4004\_8000h 基准 + Ch 偏移 = 4004\_800Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PWTCLKPS	FTM2CLKPS	FTM1CLKPS	FTM0CLKPS					0							
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		0		FTM1PS1	FTM1PS0	FTM0PS1	FTM0PS0	UART0PS	SPI0PS	I2C0PS	RTCP0S	0				IRQPS
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**SIM\_PINSEL0** 字段描述

字段	描述
31–30 PWTCLKPS	PWT TCLK 引脚选择 选择 TCLK 引脚分配。  00 选择 TCLK0 用于 PWT 模块。 01 选择 TCLK1 用于 PWT 模块。 10 选择 TCLK2 用于 PWT 模块。 11 保留。
29–28 FTM2CLKPS	FTM2 TCLK 引脚选择 选择 TCLK 引脚分配。  00 选择 TCLK0 用于 FTM2 模块。 01 选择 TCLK1 用于 FTM2 模块。 10 选择 TCLK2 用于 FTM2 模块。 11 保留。
27–26 FTM1CLKPS	FTM1 TCLK 引脚选择 选择 TCLK 引脚分配。  00 选择 TCLK0 用于 FTM1 模块。 01 选择 TCLK1 用于 FTM1 模块。 10 选择 TCLK2 用于 FTM1 模块。 11 保留。
25–24 FTM0CLKPS	FTM0 TCLK 引脚选择 选择 TCLK 引脚分配。  00 选择 TCLK0 用于 FTM0 模块。 01 选择 TCLK1 用于 FTM0 模块。

下一页继续介绍此表...

## SIM\_PINSEL0 字段描述 (继续)

字段	描述
	10 选择 TCLK2 用于 FTM0 模块。 11 保留。
23–12 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
11 FTM1PS1	FTM1_CH1 端口引脚选择 选择 FTM1_CH1 通道引脚分配。 0 FTM1_CH1 通道映射到 PTC5 上。 1 FTM1_CH1 通道映射到 PTE7 上。
10 FTM1PS0	FTM1_CH0 端口引脚选择 选择 FTM1_CH0 通道引脚分配。 0 FTM1_CH0 通道映射到 PTC4 上。 1 FTM1_CH0 通道映射到 PTH2 上。
9 FTM0PS1	FTM0_CH1 端口引脚选择 选择 FTM0_CH1 通道引脚分配。 0 FTM0_CH1 通道映射到 PTA1 上。 1 FTM0_CH1 通道映射到 PTB3 上。
8 FTM0PS0	FTM0_CH0 端口引脚选择 选择 FTM0_CH0 通道引脚分配。 0 FTM0_CH0 通道映射到 PTA0 上。 1 FTM0_CH0 通道映射到 PTB2 上。
7 UART0PS	UART0 引脚选择 选择 UART0 引脚分配。 0 UART0_RX 和 UART0_TX 映射到 PTB0 和 PTB1 上。 1 UART0_RX 和 UART0_TX 映射到 PTA2 和 PTA3 上。
6 SPI0PS	SPI0 引脚选择 选择 SPI0 引脚分配。 0 SPI0_SCK、SPI0_MOSI、SPI0_MISO 和 SPI0_PCS 映射到 PTB2、PTB3、PTB4 和 PTB5 上。 1 SPI0_SCK、SPI0_MOSI、SPI0_MISO 和 SPI0_PCS 映射到 PTE0、PTE1、PTE2 和 PTE3 上。
5 I2C0PS	I2C0 端口引脚选择 选择 I2C0 端口引脚。 0 I2C0_SCL 和 I2C0_SDA 分别映射到 PTA3 和 PTA2 上。 1 I2C0_SCL 和 I2C0_SDA 分别映射到 PTB7 和 PTB6 上。
4 RTCPs	RTCO 引脚选择 选择 RTCO 端口引脚。

下一页继续介绍此表...

**SIM\_PINSEL0 字段描述 (继续)**

字段	描述
	0 RTCO 映射到 PTC4 上。 1 RTCO 映射到 PTC5 上。
3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
IRQPS	IRQ 端口引脚选择  选择 IRQ 端口引脚。  000 IRQ 映射到 PTA5 上。 001 IRQ 映射到 PTI0 上。 010 IRQ 映射到 PTI1 上。 011 IRQ 映射到 PTI2 上。 100 IRQ 映射到 PTI3 上。 101 IRQ 映射到 PTI4 上。 110 IRQ 映射到 PTI5 上。 111 IRQ 映射到 PTI6 上。

**12.2.5 引脚选择寄存器 1 (SIM\_PINSEL1)**

地址: 4004\_8000h 基准 + 10h 偏移 = 4004\_8010h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									0							
W																MSCANPS
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PWTIN1PS	PWTIN0PS	UART2PS	UART1PS	SPI1PS	I2C1PS	FTM2PS5	FTM2PS4	FTM2PS3	FTM2PS2	FTM2PS1	FTM2PS0				
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**SIM\_PINSEL1 字段描述**

字段	描述
31–17 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
16 MSCANPS	MSCAN 引脚选择  选择 MSCAN 引脚分配。  0 CAN_TX 在 PTC7 上, CAN_RX 在 PTC6 上。 1 CAN_TX 在 PTE7 上, CAN_RX 在 PTH2 上。

下一页继续介绍此表...

## SIM\_PINSEL1 字段描述 (继续)

字段	描述
15 PWTIN1PS	PWTIN1 引脚选择 选择 PWTIN1 引脚分配。 0 PWTIN1 在 PTB0 上。 1 PWTIN1 在 PTH7 上。
14 PWTIN0PS	PWTIN0 引脚选择 选择 PWTIN0 引脚分配。 0 PWTIN0 在 PTD5 上。 1 PWTIN0 在 PTE2 上。
13 UART2PS	UART2 引脚选择 选择 UART2 引脚分配。 0 UART2_TX 在 PTD7 上 , UART2_RX 在 PTD6 上。 1 UART2_TX 在 PTI1 上 , UART2_RX 在 PTI0 上。
12 UART1PS	UART1 引脚选择 选择 UART1 引脚分配。 0 UART1_TX 在 PTC7 上 , UART1_RX 在 PTC6 上。 1 UART1_TX 在 PTF3 上 , UART1_RX 在 PTF2 上。
11 SPI1PS	SPI1 引脚选择 选择 SPI1 引脚分配。 0 SPI1_SCK、SPI1_MOSI、SPI1_MISO 和 SPI1_PCS 映射到 PTD0、PTD1、PTD2 和 PTD3 上。 1 SPI1_SCK、SPI1_MOSI、SPI1_MISO 和 SPI1_PCS 映射到 PTG4、PTG5、PTG6 和 PTG7 上。
10 I2C1PS	I2C1 引脚选择 选择 I2C1 引脚分配。 0 I2C1_SCL 在 PTE1 上 , I2C1_SDA 在 PTE0 上。 1 I2C1_SCL 在 PTH4 上 , I2C1_SDA 在 PTH3 上。
9 FTM2PS5	FTM2 通道 5 引脚选择 选择 FTM2 通道 5 引脚分配。 0 FTM2 CH5 映射到 PTB5 上。 1 FTM2 CH5 映射到 PTG7 上。
8 FTM2PS4	FTM2 通道 4 引脚选择 选择 FTM2 通道 4 引脚分配。 0 FTM2 CH4 映射到 PTB4 上。 1 FTM2 CH4 映射到 PTG6 上。
7–6 FTM2PS3	FTM2 通道 3 引脚选择 选择 FTM2 通道 3 引脚分配。

下一页继续介绍此表...

**SIM\_PINSEL1 字段描述 (继续)**

字段	描述
	00 FTM2 CH3 映射到 PTC3 上。 01 FTM2 CH3 映射到 PTD1 上。 10 FTM2 CH3 映射到 PTG5 上。 11 保留。
5–4 FTM2PS2	<b>FTM2 通道 2 引脚选择</b> 选择 FTM2 通道 2 引脚分配。  00 FTM2 CH2 映射到 PTC2 上。 01 FTM2 CH2 映射到 PTD0 上。 10 FTM2 CH2 映射到 PTG4 上。 11 保留。
3–2 FTM2PS1	<b>FTM2 通道 1 引脚选择</b> 选择 FTM2 通道 1 引脚分配。  00 FTM2 CH1 映射到 PTC1 上。 01 FTM2 CH1 映射到 PTH1 上。 10 FTM2 CH1 映射到 PTF1 上。 11 保留。
FTM2PS0	<b>FTM2 通道 0 引脚选择</b> 选择 FTM2 通道 0 引脚分配。  00 FTM2 CH0 映射到 PTC0 上。 01 FTM2 CH0 映射到 PTH0 上。 10 FTM2 CH0 映射到 PTF0 上。 11 保留。

**12.2.6 系统时钟选通控制寄存器 (SIM\_SCGC)**

地址: 4004\_8000h 基准 + 14h 偏移 = 4004\_8014h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ACMP1	ACMP0	ADC	0	IRQ	0	KBI1	KBI0	0	UART2	UART1	UART0	SPI1	SPI0	I2C1	I2C0
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MSCAN	0	SWD	FLASH	0	CRC	0	FTM2	FTM1	FTM0	PWT	0	PIT	RTC		
W																
复位	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

**SIM\_SCGC 字段描述**

字段	描述
31 ACMP1	ACMP1 时钟选通控制  控制 ACMP1 模块的时钟选通。  0 ACMP1 模块的总线时钟禁用。 1 ACMP1 模块的总线时钟使能。
30 ACMP0	ACMP0 时钟选通控制  控制 ACMP0 模块的时钟选通。  0 ACMP0 模块的总线时钟禁用。 1 ACMP0 模块的总线时钟使能。
29 ADC	ADC 时钟选通控制  控制 ADC 模块的时钟选通。  0 ADC 模块的总线时钟禁用。 1 ADC 模块的总线时钟使能。
28 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
27 IRQ	IRQ 时钟选通控制  控制 IRQ 模块的时钟选通。  0 IRQ 模块的总线时钟禁用。 1 IRQ 模块的总线时钟使能。
26 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
25 KBI1	KBI1 时钟选通控制  控制 KBI1 模块的时钟选通。  0 KBI1 模块的总线时钟禁用。 1 KBI1 模块的总线时钟使能。
24 KBI0	KBI0 时钟选通控制  控制 KBI0 模块的时钟选通。  0 KBI0 模块的总线时钟禁用。 1 KBI0 模块的总线时钟使能。
23 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
22 UART2	UART2 时钟选通控制  控制 UART2 模块的时钟选通。  0 UART2 模块的总线时钟禁用。 1 UART2 模块的总线时钟使能。
21 UART1	UART1 时钟选通控制  控制 UART1 模块的时钟选通。

下一页继续介绍此表...

**SIM\_SCGC 字段描述 (继续)**

字段	描述
	0 UART1 模块的总线时钟禁用。 1 UART1 模块的总线时钟使能。
20 UART0	UART0 时钟选通控制  控制 UART0 模块的时钟选通。  0 UART0 模块的总线时钟禁用。 1 UART0 模块的总线时钟使能。
19 SPI1	SPI1 时钟选通控制  控制 SPI1 模块的时钟选通。  0 SPI1 模块的总线时钟禁用。 1 SPI1 模块的总线时钟使能。
18 SPI0	SPI0 时钟选通控制  控制 SPI0 模块的时钟选通。  0 SPI0 模块的总线时钟禁用。 1 SPI0 模块的总线时钟使能。
17 I2C1	I2C1 时钟选通控制  控制 I2C1 模块的时钟选通。  0 I2C1 模块的总线时钟禁用。 1 I2C1 模块的总线时钟使能。
16 I2C0	I2C0 时钟选通控制  控制 I2C0 模块的时钟选通。  0 I2C0 模块的总线时钟禁用。 1 I2C0 模块的总线时钟使能。
15 MSCAN	MSCAN 时钟选通控制  控制 MSCAN 模块的时钟选通。  0 MSCAN 模块的总线时钟禁用。 1 MSCAN 模块的总线时钟使能。
14 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
13 SWD	SWD ( 单线调试器 ) 时钟选通控制  控制 SWD 模块的时钟选通。  0 SWD 模块的总线时钟禁用。 1 SWD 模块的总线时钟使能。
12 FLASH	Flash 时钟选通控制  控制 Flash 模块的时钟选通。

下一页继续介绍此表...

**SIM\_SCGC 字段描述 (继续)**

字段	描述
	0 Flash 模块的总线时钟禁用。 1 Flash 模块的总线时钟使能。
11 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
10 CRC	CRC 时钟选通控制  控制 CRC 模块的时钟选通。  0 CRC 模块的总线时钟禁用。 1 CRC 模块的总线时钟使能。
9–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 FTM2	FTM2 时钟选通控制  控制 FTM2 模块的时钟选通。  0 FTM2 模块的总线时钟禁用。 1 FTM2 模块的总线时钟使能。
6 FTM1	FTM1 时钟选通控制  控制 FTM1 模块的时钟选通。  0 FTM1 模块的总线时钟禁用。 1 FTM1 模块的总线时钟使能。
5 FTM0	FTM0 时钟选通控制  控制 FTM0 模块的时钟选通。  0 FTM0 模块的总线时钟禁用。 1 FTM0 模块的总线时钟使能。
4 PWT	PWT 时钟选通控制  控制 PWT 模块的时钟选通。  0 PWT 模块的定时器时钟禁用。 1 PWT 模块的定时器时钟使能。
3–2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
1 PIT	PIT 时钟选通控制  控制 PIT 模块的时钟选通。  0 PIT 模块的总线时钟禁用。 1 PIT 模块的总线时钟使能。
0 RTC	RTC 时钟选通控制  控制 RTC 模块的时钟选通。  0 RTC 模块的总线时钟禁用。 1 RTC 模块的总线时钟使能。

## 12.2.7 通用唯一标识符低位寄存器 (SIM\_UIDL)

只读 SIM\_UIDL 寄存器包含一系列数字，用以识别该系列中的不同器件。

地址: 4004\_8000h 基准 + 18h 偏移 = 4004\_8018h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	ID[31:0]															
W																																

复位 x\* x\*

\* 注:

- x = 复位时未定义。

### SIM\_UIDL 字段描述

字段	描述
ID[31:0]	通用唯一标识符

## 12.2.8 通用唯一标识符中低位寄存器 (SIM\_UUIDML)

只读 SIM\_UUIDML 寄存器包含一组数字，用于标识系列中的唯一器件。

地址: 4004\_8000h 基准 + 1Ch 偏移 = 4004\_801Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	ID[63:32]															
W																																

复位 x\* x\*

\* 注:

- x = 复位时未定义。

### SIM\_UUIDML 字段描述

字段	描述
ID[63:32]	通用唯一标识符

## 12.2.9 通用唯一标识符中高位寄存器 (SIM\_UUIDMH)

只读 SIM\_UUIDMH 寄存器包含一系列数字，用以识别该系列中的不同器件。

地址: 4004\_8000h 基准 + 20h 偏移 = 4004\_8020h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															ID[80:64]																
W																																

复位 X\* X\*

\* 注:

- X = 复位时未定义。

### SIM\_UUIDMH 字段描述

字段	描述
31–16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
ID[80:64]	通用唯一标识符

## 12.2.10 时钟分频器寄存器 (SIM\_CLKDIV)

该寄存器设置时钟的分频值。

### 注

小心配置 OUTDIV1 和 OUTDIV2，避免总线时钟频率高于 24 MHz。

地址: 4004\_8000h 基准 + 24h 偏移 = 4004\_8024h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0		OUTDIV1				0		OUTDIV2				0		0		
W																	
复位	0	0	u*	u*	0	0	0	u*	0	0	0	u*	0	0	0	0	
POR/ LVD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0																
W																	
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

POR/ LVD 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0

- \* 注:
  - $u$  = 不受复位影响。

## SIM\_CLKDIV 字段描述

字段	描述
31–30 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
29–28 OUTDIV1	时钟 1 输出分频器值  该字段设置内核/系统时钟的分频值。  00 同 ICSOUTCLK。 01 ICSOUTCLK 除以 2。 10 ICSOUTCLK 除以 3。 11 ICSOUTCLK 除以 4。
27–25 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
24 OUTDIV2	时钟 2 输出分频器值  该字段设置总线/FLASH 的分频值，跟随 OUTDIV1。  0 不根据分频器 1 进行分频。 1 根据分频器 1 进行 2 分频。
23–21 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
20 OUTDIV3	时钟 3 输出分频器值  该字段设置定时器 (FTM0、FTM1、FTM2、PWT) 的分频值。  0 同 ICSOUTCLK。 1 ICSOUTCLK 除以 2。
保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

### 12.3 功能说明

参见简介部分。

# 第 13 章 电源管理控制器(PMC)

## 13.1 简介

本章介绍芯片低功耗模式下各模块的功能以及电源管理控制器模块的操作。

## 13.2 低电压检测(LVD)系统

该设备包含一个低电压保护系统，以便在供电电压变动时保护存储器中的内容并控制 MCU 系统的状态。此系统包含一个上电复位(POR)电路，和具有可供用户选择触发电压 (高电压( $V_{LVDH}$ )或低电压( $V_{LVDL}$ )) 的 LVD 电路。SPMSC1[LVDE]置位且 SPMSC2[LVDV]选择触发电压后，即可启用 LVD 电路。除非设置 SPMSC1[LVDSE]，否则在输入 Stop 模式时将禁用 LVD。如果 SPMSC1[LVDSE] 和 SPMSC1[LVDE] 均已设置，则启用 LVD 系统时，电流消耗量将高于 Stop 模式中的电流消耗量。

下图呈现的是低电压检测(LVD)系统的结构框图。

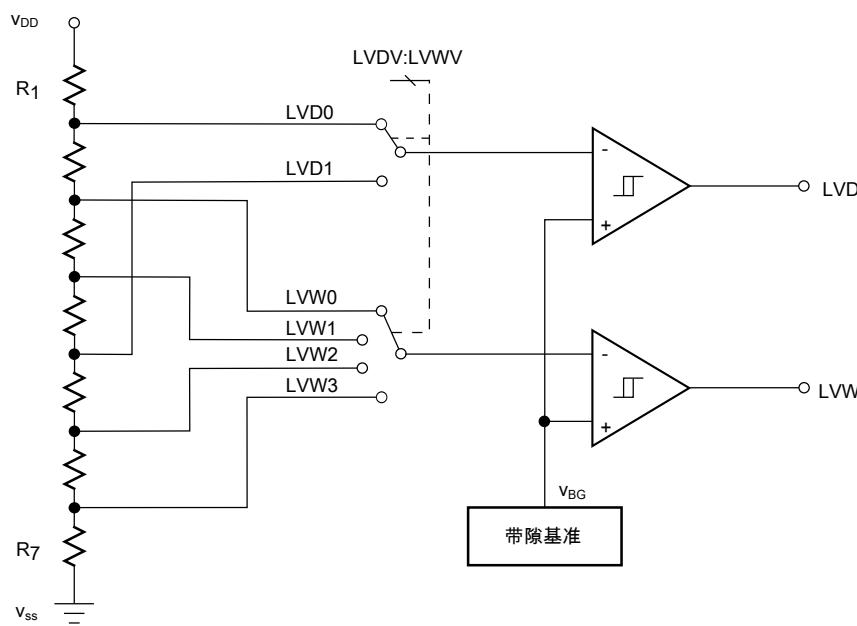


图 13-1. 低电压检测(LVD)结构框图

### 13.2.1 上电复位(POR)操作

MCU 初次上电或电源电压下降低于  $V_{POR}$  电平时, POR 电路将触发 MCU 复位。随着供电电压的升高, LVD 电路将芯片保持在复位状态, 直至电源超过  $V_{LVDL}$  电平。SIM\_SRSID[POR]和 SIM\_SRSID[LVD]在 POR 之后都将置位。

### 13.2.2 LVD 复位操作

通过将 SPMSC1[LVDRE]置 1, LVD 可配置为一旦检测到低压条件就生成复位信号。LVD 复位后, LVD 系统会将 MCU 保持在复位状态, 直到电源电压上升高于 LVDV 电平。SIM\_SRSID[LVD]在 LVD 复位或 POR 之后置位。

### 13.2.3 Stop 模式下的 LVD 使能

电源电压下降低于 LVD 电压时, LVD 系统能生成复位信号。若 LVD 在 CPU 执行 STOP 指令时在 Stop 模式 (SPMSC1[LVDE]和 SPMSC1[LVDSE]都置 1) 下使能, 则稳压器在 Stop 模式期间仍保持激活状态。

### 13.2.4 低压警报(LVW)

LVD 系统具有低压警报标志，可提示电源电压正在接近 LVW 电压。检测到低压条件且 LVD 电路已配置为可中断操作 (SPMSC1[LVDE]置位, SPMSC1[LVWIE]置位) 时，SPMSC1[LVWF]将置位，并且将发生 LVW 中断。每次 LVDV 配置时，LVW 都有四个用户可选触发电压。触发电压根据 SPMSC2[LVWV]进行选择。

### 13.3 带隙基准源

该器件集成了一个连接至 ADC 通道的片上带隙基准源( $\approx 1.2 \text{ V}$ )。即使工作电压正在下降，带隙基准电压也不会下降低于全工作电压。该基准电压用作精确测量的理想基准电压。

### 13.4 存储器映像和寄存器说明

#### PMC 存储器映射

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4007_D000	系统电源管理状态和控制 1 寄存器 (PMC_SPMSC1)	8	R/W	1Ch	<a href="#">13.4.1/187</a>
4007_D001	系统电源管理状态和控制 2 寄存器 (PMC_SPMSC2)	8	R/W	00h	<a href="#">13.4.2/189</a>

#### 13.4.1 系统电源管理状态和控制 1 寄存器 (PMC\_SPMSC1)

该寄存器包含状态位和控制位，支持低压检测功能，可使能带隙基准电压供 ADC 模块使用。即使所需设置与复位设置相同，该寄存器也必须在用户复位初始化程序运行期间写入数据以设置所需的控制。

地址: 4007\_D000h 基准 + 0h 偏移 = 4007\_D000h

位	7	6	5	4	3	2	1	0
读	LVWF	0	LVWIE	LVDRE	LVDSE	LVDE		BGBE
写		LVWACK					0	
复位	0	0	0	1	1	1	0	0

#### PMC\_SPMSC1 字段描述

字段	描述
7 LVWF	低压警报标志 指示低压警报状态。

下一页继续介绍此表...

## PMC\_SPMSC1 字段描述 (继续)

字段	描述
	<p>注：如果 <math>V_{Supply}</math> 转换低于跳变点或在复位后 <math>V_{Supply}</math> 已低于 <math>V_{LVW}</math>，那么 LVWF 将置位。上电复位后，LVWF 可能为 1；因此，为了使用 LVW 中断功能，在使能 LVWIE 前，LVWF 必须首先通过写入 LVWACK 清零。</p> <p>0 不存在低压警报。 1 存在或曾经存在低压警报。</p>
6 LVWACK	<p>低压警报应答</p> <p>若 <math>LVWF = 1</math>，则已发生低压条件。为了应答该低压警报，向 LVWACK 写入 1，如果低压警报不再存在，那么就会自动清零 LVWF。</p>
5 LVWIE	<p>低压警报中断使能</p> <p>使能 LVWF 的硬件中断请求。</p> <p>0 禁用硬件中断（使用轮询）。 1 <math>LVWF = 1</math> 时请求硬件中断。</p>
4 LVDRE	<p>低压检测复位使能</p> <p>该一次性写入位可使能 LVD 事件以产生硬件复位（假设 <math>LVDE = 1</math>）。</p> <p>注：复位后，该字段仅可写入一次。其他写操作会被忽略。</p> <p>若 <math>LVDRE = 0</math>，由于没有标志置位，则可用 LVW 去监控状态。</p> <p>0 LVD 事件不产生硬件复位。 1 发生有效的低压检测事件时强制进行 MCU 复位</p>
3 LVDSE	<p>低压检测在 Stop 模式下的使能</p> <p>假设 <math>LVDE = 1</math>，则该读/写字段可确定低压检测功能在 MCU 处于 Stop 模式时是否工作。</p> <p>0 低压检测在 Stop 模式期间禁用。 1 低压检测在 Stop 模式期间使能。</p>
2 LVDE	<p>低压检测使能</p> <p>该一次性写入位可以使能低压检测逻辑并认证该寄存器中其他字段的操作。</p> <p>注：复位后，该字段仅可写入一次。其他写操作会被忽略。</p> <p>0 LVD 逻辑禁用。 1 LVD 逻辑使能。</p>
1 Reserved	此字段为保留字段。
0 BGBE	<p>带隙基准缓冲区使能</p> <p>使能内部缓冲区以便带隙基准电压可供 ADC 模块在其内部通道上作为基准电压或选作 ACMP 基准的带隙基准。</p> <p>0 带隙基准缓冲区禁用。 1 带隙基准缓冲区使能。</p>

### 13.4.2 系统电源管理状态和控制 2 寄存器 (PMC\_SPMSC2)

该寄存器用于报告低压警报功能的状态，以及配置 MCU 的 Stop 模式特性。即使所需设置与复位设置相同，该寄存器也应当在用户复位初始化程序运行期间写入数据以设置所需的控制。

地址: 4007\_D000h 基准 + 1h 偏移 = 4007\_D001h

位	7	6	5	4	3	2	1	0
读写	0	LVDV	LVWV		0	0	0	0
复位	0	0	0	0	0	0	0	0

PMC\_SPMSC2 字段描述

字段	描述
7 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
6 LVDV	低压检测电压选择  该一次性写入位选择低压检测(LVD)跳变点设置。更多详情请参见数据手册。  0 选择低电平跳变点( $V_{LVD} = V_{LVDL}$ )。 1 选择高电平跳变点( $V_{LVD} = V_{LVDH}$ )。
5–4 LVWV	低压警报电压选择  选择低压警报(LVW)跳变点电压。更多详情请参见数据手册。  00 选择低电平跳变点( $V_{LVW} = V_{LVW1}$ )。 01 选择中间电平 1 跳变点( $V_{LVW} = V_{LVW2}$ )。 10 选择中间电平 2 跳变点( $V_{LVW} = V_{LVW3}$ )。 11 选择高电平跳变点( $V_{LVW} = V_{LVW4}$ )。
保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。



# 第 14 章 杂项控制模块(MCM)

## 14.1 简介

### 注

关于与具体芯片相关的本模块详细操作示例, 请参见芯片配置信息。

杂项控制模块(MCM)可提供多种杂项控制功能。

### 14.1.1 特性

MCM 包含下列特性:

- 程序可见的有关平台配置信息
- Flash 控制器推测缓冲区和高速缓存配置

## 14.2 存储器映像/寄存器说明

下文的存储器映像和寄存器的描述介绍了使用字节地址的寄存器。这些寄存器仅可在管理模式下写入。

**MCM 存储器映射**

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
F000_3008	交叉开关(AXBS)从机配置 (MCM_PLASC)	16	R	0007h	<a href="#">14.2.1/192</a>
F000_300A	交叉开关(AXBS)主机配置 (MCM_PLAMC)	16	R	0001h	<a href="#">14.2.2/192</a>
F000_300C	平台控制寄存器 (MCM_PLACR)	32	R/W	0000_0800h	<a href="#">14.2.3/193</a>

### 14.2.1 交叉开关(AXBS)从机配置 (MCM\_PLASC)

PLASC 是一个 16 位只读寄存器，用于识别是否存在总线从连接至设备交叉开关。

地址: F000\_3000h 基准 + 8h 偏移 = F000\_3008h

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读					0									ASC		
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

**MCM\_PLASC 字段描述**

字段	描述
15–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
ASC	ASC 字段中的每个位表示交叉开关的从输入端口是否存在对应连接。  0 不存在至 AXBS 输入端口 $n$ 的总线从连接。 1 存在至 AXBS 输入端口 $n$ 的总线从连接。

### 14.2.2 交叉开关(AXBS)主机配置 (MCM\_PLAMC)

PLAMC 是一个 16 位只读寄存器，用于识别是否存在总线主连接至设备交叉开关。

地址: F000\_3000h 基准 + Ah 偏移 = F000\_300Ah

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读					0									AMC		
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

**MCM\_PLAMC 字段描述**

字段	描述
15–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
AMC	AMC 字段中的每个位表示 AXBS 主输入端口是否存在对应连接。  0 不存在至 AXBS 输入端口 $n$ 的总线主连接。 1 存在至 AXBS 输入端口 $n$ 的总线主连接。

### 14.2.3 平台控制寄存器 (MCM\_PLACR)

Flash 存储器控制器中的推测缓冲区和高速缓存可通过 PLACR[15:10]配置。

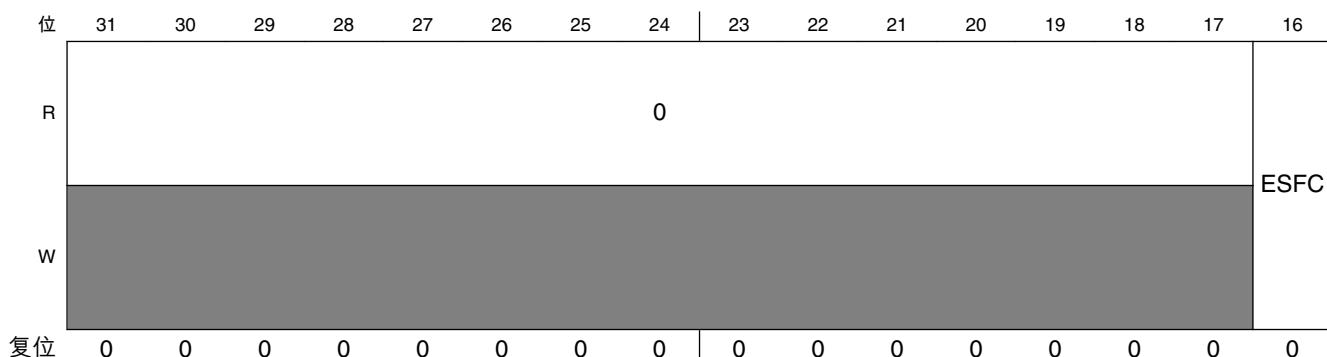
复位后，推测缓冲器仅对指令有效。推测缓冲区可能具有以下状态：

DFCS	EFDS	说明
0	0	推测缓冲区对于指令为开，对于数据为关。
0	1	推测缓冲区对于指令为开，对于数据为开。
1	X	推测缓冲区关闭。

Flash 控制器中的高速缓存在复位后，已使能且可以对指令和数据进行提取。高速缓存可能具有以下状态：

DFCC	DFCIC	DFCDA	说明
0	0	0	高速缓存对于指令和数据均为开。
0	0	1	高速缓存对于指令为开，对于数据为关。
0	1	0	高速缓存对于指令为关，对于数据为开。
0	1	1	高速缓存对于指令和数据均为关。
1	X	X	高速缓存关闭。

地址: F000\_3000h 基准 + Ch 偏移 = F000\_300Ch



位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	DFCS	EFDS	DFCC	DFCIC	DFCDA	0							0				
W							CFCC										
复位	0	0	0	0	1	0	0	0		0	0	0	0	0	0	0	0

**MCM\_PLACR 字段描述**

字段	描述
31-17 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
16 ESFC	使能阻塞 Flash 控制器  当 Flash 正忙时使能阻塞 Flash 控制器。  当 Flash 存储器资源正在通过 Flash 命令使用时，如果软件需要访问 Flash 存储器，则该软件可以使能阻塞机制，以避免读取碰撞。阻塞机制允许该软件执行来自与执行 Flash 操作的相同区块的代码。但是，该软件必须能够确保在其上执行 Flash 操作的扇区与执行代码的扇区不同。  ESFC 使能阻塞机制。该位仅可在执行 Flash 操作之前置位，并且必须在操作完成时清除。  0 当 Flash 正忙时禁用阻塞 Flash 控制器。 1 当 Flash 正忙时使能阻塞 Flash 控制器。
15 DFCS	禁用 Flash 控制器的推测  禁用 Flash 控制器的推测。  0 使能 Flash 控制器的推测。 1 禁用 Flash 控制器的推测。
14 EFDS	使能 Flash 数据的推测  使能 Flash 数据的推测。  0 禁用 Flash 数据的推测。 1 使能 Flash 数据的推测。
13 DFCC	禁用 Flash 控制器高速缓存  禁用 Flash 控制器高速缓存。  0 使能 Flash 控制器高速缓存。 1 禁用 Flash 控制器高速缓存。
12 DFCIC	禁用 Flash 控制器指令高速缓存  禁用 Flash 控制器指令高速缓存。  0 使能 Flash 控制器指令高速缓存。 1 禁用 Flash 控制器指令高速缓存。
11 DFCDA	禁用 Flash 控制器数据高速缓存

下一页继续介绍此表...

**MCM\_PLACR 字段描述 (继续)**

字段	描述
	禁用 Flash 控制器数据高速缓存。 0 使能 Flash 控制器数据高速缓存 1 禁用 Flash 控制器数据高速缓存。
10 CFCC	清除 Flash 控制器高速缓存 将 1 写入该字段会清除高速缓存。将 0 写入该字段会被忽略。该字段始终读取为 0。
保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。



# 第 15 章 外设桥(AIPS-Lite)

## 15.1 简介

### 注

关于与具体芯片相关的本模块详细操作示例，请参见芯片配置信息。

外设桥将交叉开关接口转连至一个能访问片上大多数从机外设的接口。

外设桥占用 64 MB 地址空间（划分为许多 4 KB 的外设插槽）。（所有的外设插槽也可能都未使用。有关插槽分配的详情，请参见“存储器映像”章节。）该联接器包含单独的各时钟使能输入，使每个插槽都能支持速度较低的外设。

### 15.1.1 特性

外设桥的主要特性：

- 支持 8 位、16 位和 32 位数据路径宽度的外设插槽

### 15.1.2 一般操作

连接外设桥的从设备模块均有可编程的控制和状态寄存器。系统主机通过外设桥对这些寄存器进行读写操作。外设桥执行主机事务的总线协议转换并产生下列信号作为外设的输入：

- 模块使能
- 模块地址
- 传输属性
- 字节使能
- 写入数据

外设桥从外设接口选择并捕捉读取的数据，并将其返回至交叉开关。

外设的寄存器映像位于 4 KB 边界上。每个外设均分配了一个或多个 4-KB 存储器映像数据块。

AIPS-Lite 模块使用已访问外设的数据宽度执行适当的数据字节通道路由；访问尺寸大于外设数据宽度时执行总线分解（总线尺寸调整）。

## 15.2 功能说明

外设桥用作交叉开关和从机外设总线之间的总线协议转换器。

外设桥管理面向连接的从设备的所有事务，并通过对连接的地址空间的访问进行解码，从而为外设总线上的模块生成选择信号。

### 15.2.1 访问支持

支持访问大小和外设数据端口宽度的所有组合。大于目标外设数据宽度的访问将被分解成多个小一些的访问。总线分解会由访问空寄存器空间的传输错误而终止。

# 第 16 章 WDOG

## 16.1 简介

WDOG 定时器模块是一个可供系统使用的独立定时器。它提供了安全特性，可确保软件按计划执行，且 CPU 不会陷入死循环中或者执行错误的代码。若 WDOG 模块在规定时间内未得到服务（刷新），它会复位 MCU。

### 16.1.1 特性

WDOG 模块特性包括：

- 独立的可配置时钟源，与下述无关：
  - 总线时钟
  - 内部 32 kHz RC 振荡器
  - 内部 1 kHz RC 振荡器
  - 外部时钟源
- 可编程的定时溢出周期
  - 可编程的 16 位定时溢出值
  - 如需更长的定时溢出周期，可选用固定 256 倍的时钟预分频器
- 稳健的计数器刷新写入序列
  - 在 16 个总线时钟内，先后写入 0x02A6 以及 0x80B4 进行刷新
- 可选的窗口刷新模式
  - 可编程的 16 位窗口值

- 监测程序流程是否快于预期
- 过早刷新会触发复位
- 可选的定时溢出中断，允许后期诊断处理
  - 根据中断向量指向的中断服务程序，CPU 响应相应的中断请求
  - 在中断产生 128 个总线时钟后，强制复位
- WDOG 配置位在复位后只能被写入一次，防止误修改
- 解锁 WDOG 单次写入配置位
  - 在 16 个总线时钟内，先写入 0x20C5 再写入 0x28D9 的解锁序列，可实现对 WDOG 配置位的更新
  - 软件必须在解锁后、且 WDOG 解锁窗口关闭前的 128 个总线时钟内完成配置更新的操作

## 16.1.2 结构框图

下图为 WDOG 模块的结构框图。

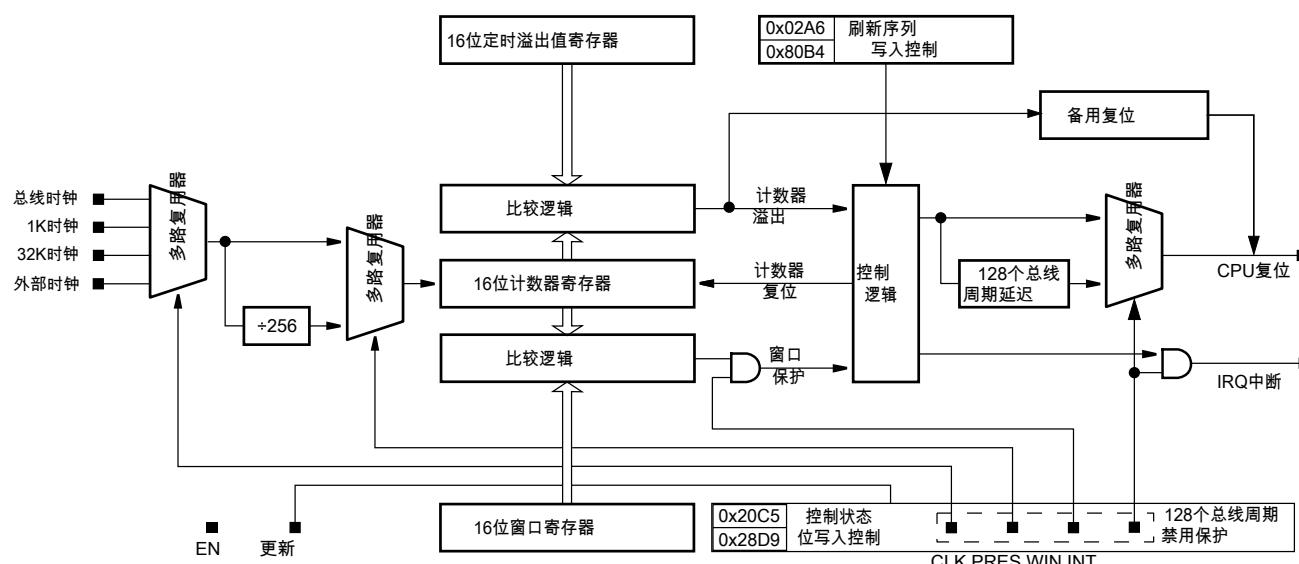


图 16-1. WDOG 结构框图

## 16.2 存储器映像和寄存器定义

### 注

如果使用半字访问 WDOG\_CNT、WDOG\_TOVAL 和 WDOG\_WIN，则必须遵循“低字节：高字节”的转置 16 位字节格式。因此推荐使用 8 位的读/写操作方式。

### WDOG 存储器映射

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4005_2000	WDOG 控制和状态寄存器 1 (WDOG_CS1)	8	R/W	80h	<a href="#">16.2.1/201</a>
4005_2001	WDOG 控制和状态寄存器 2 (WDOG_CS2)	8	R/W	01h	<a href="#">16.2.2/203</a>
4005_2002	WDOG 计数器寄存器：高位 (WDOG_CNTH)	8	R	00h	<a href="#">16.2.3/203</a>
4005_2003	WDOG 计数器寄存器：低位 (WDOG_CNTL)	8	R	00h	<a href="#">16.2.4/204</a>
4005_2004	WDOG 定时溢出值寄存器：高位 (WDOG_TOVALH)	8	R/W	00h	<a href="#">16.2.5/205</a>
4005_2005	WDOG 定时溢出值寄存器：低位 (WDOG_TOVALL)	8	R/W	04h	<a href="#">16.2.6/205</a>
4005_2006	WDOG 窗口寄存器：高位 (WDOG_WINH)	8	R/W	00h	<a href="#">16.2.7/206</a>
4005_2007	WDOG 窗口寄存器：低位 (WDOG_WINL)	8	R/W	00h	<a href="#">16.2.8/206</a>

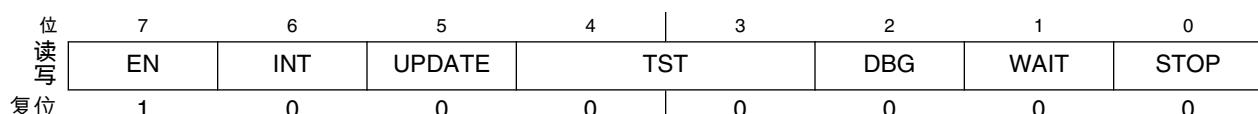
### 16.2.1 WDOG 控制和状态寄存器 1 (WDOG\_CS1)

本节说明 WDOG 控制和状态寄存器 1 的功能。

### 注

TST 仅在 POR 上才清零(0:0)。任何其他复位都不会影响该字段的数值。

地址: 4005\_2000h 基准 + 0h 偏移 = 4005\_2000h



### WDOG\_CS1 字段描述

字段	描述
7 EN	<p>WDOG 使能</p> <p>该一次写入位使能 WDOG 计数器以开始计数。</p> <p>0 WDOG 禁用。 1 WDOG 使能。</p>
6 INT	WDOG 中断

下一页继续介绍此表...

## WDOG\_CS1 字段描述 (继续)

字段	描述
	<p>该一次写入位配置 WDOG，使得在发生触发复位的事件（定时溢出或 WDOG 非法写入）时，先产生一个中断请求，然后再强制进行复位。中断产生后的 128 个总线时钟延迟之后发生强制复位。</p> <p>0 WDOG 中断禁用。WDOG 复位无延迟。 1 WDOG 中断使能。WDOG 复位有 128 个总线时钟延迟。</p>
5 UPDATE	<p>允许更新</p> <p>该一次写入位使系统无需复位即能重新配置 WDOG。</p> <p>0 不允许更新。完成初始配置后，除非强制复位，否则无法修改 WDOG 配置。 1 允许更新。执行解锁写入序列后，软件可以在 128 个总线时钟内修改 WDOG 配置寄存器。</p>
4–3 TST	<p>WDOG 测试</p> <p>使能快速测试模式。测试模式允许软件检查计数器的所有位，来验证 WDOG 工作正常。参见 <a href="#">WDOG 快速测试</a> 部分。</p> <p>该一次写入字段仅在 POR 时才清零(0:0)。任何其他复位都不会影响该字段的数值。</p> <p>00 WDOG 测试模式禁用。 01 WDOG 用户模式使能（WDOG 测试模式禁用）。完成 WDOG 测试后，软件应使用这项测试来表明 WDOG 在用户模式下工作正常。 10 WDOG 测试模式使能，仅使用低位字节。将 WDOG_CNTL 与 WDOG_TOVALL 进行比较。 11 WDOG 测试模式使能，仅使用高位字节。将 WDOG_CNTH 与 WDOG_TOVALH 进行比较。</p>
2 DBG	<p>Debug 模式使能</p> <p>该一次写入位在芯片处于 Debug 模式时使能 WDOG 的操作。</p> <p>0 WDOG 在芯片 Debug 模式下禁用。 1 WDOG 在芯片 Debug 模式下使能。</p>
1 WAIT	<p>Wait 模式使能</p> <p>该一次写入位在芯片处于 Wait 模式时使能 WDOG 的操作。</p> <p>0 WDOG 在芯片 Wait 模式下禁用。 1 WDOG 在芯片 Wait 模式下使能。</p>
0 STOP	<p>Stop 模式使能</p> <p>该一次写入位在芯片处于 Stop 模式时使能 WDOG 的操作。</p> <p>0 WDOG 在芯片 Stop 模式下禁用。 1 WDOG 在芯片 Stop 模式下使能。</p>

## 16.2.2 WDOG 控制和状态寄存器 2 (WDOG\_CS2)

本节说明 WDOG 控制和状态寄存器 2 的功能。

地址: 4005\_2000h 基准 + 1h 偏移 = 4005\_2001h

位	7	6	5	4	3	2	1	0
读	WIN	FLG	0	PRES	0			
写	w1c						CLK	
复位	0	0	0	0	0	0	0	1

**WDOG\_CS2 字段描述**

字段	描述
7 WIN	WDOG 窗口工作模式 该一次写入位使能窗口模式。参见 <a href="#">窗口模式</a> 部分。 0 窗口模式禁用。 1 窗口模式使能。
6 FLG	WDOG 中断标志 当 INT 在控制和状态寄存器 1 中置位后，该位是中断产生的标志位。写入 1 将其清零。 0 未发生中断。 1 发生一次中断。
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
4 PRES	WDOG 预分频器 该一次写入位使能 WDOG 计数器基准时钟的 256 倍预分频器。( 功能框图中显示了该时钟分频器选项。) 0 256 预分频器禁用。 1 256 预分频器使能。
3–2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
CLK	WDOG 时钟 该一次写入字段标示 WDOG 计数器的时钟源。参见 <a href="#">时钟源</a> 部分。 00 总线时钟。 01 1 kHz 内部低功耗振荡器(LPOCLK)。 10 32 kHz 内部振荡器(ICSIRCLK)。 11 外部时钟源。

## 16.2.3 WDOG 计数器寄存器：高位 (WDOG\_CNTH)

本节说明 WDOG 计数器寄存器：高位(CNTH)和低位(CNTL)。

WDOG 计数器寄存器 CNTH 和 CNTL 提供对 WDOG 计数器数值的访问。软件可在任意时刻对计数器寄存器进行读操作。

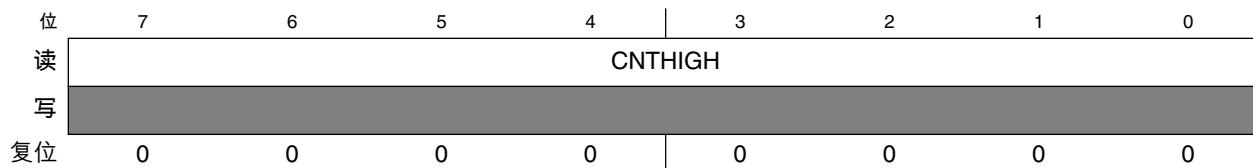
软件无法直接写入 WDOG 计数器；然而，针对这两个寄存器的两个写入序列具有特殊功能：

1. 刷新序列可将 WDOG 计数器复位至 0x0000。参见[刷新 WDOG 部分](#)。
2. 解锁序列使 WDOG 能在不强制复位 (WDOG\_CS1[UPDATE] = 1 时) 的情况下得到重新配置。参见[代码示例：重新配置 WDOG 部分](#)。

### 注

对这些寄存器的所有其他写操作均无效，并且会强制复位。

地址: 4005\_2000h 基准 + 2h 偏移 = 4005\_2002h



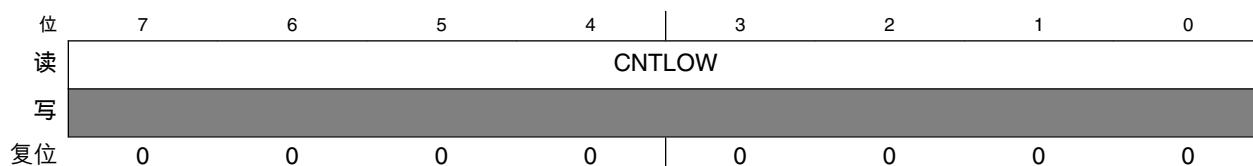
**WDOG\_CNTH 字段描述**

字段	描述
CNTHIGH	WDOG 计数器的高位字节

## 16.2.4 WDOG 计数器寄存器：低位 (WDOG\_CNTL)

参见 WDOG\_CNTH 寄存器说明。

地址: 4005\_2000h 基准 + 3h 偏移 = 4005\_2003h



**WDOG\_CNTL 字段描述**

字段	描述
CNTLOW	WDOG 计数器的低位字节

### 16.2.5 WDOG 定时溢出值寄存器：高位 (WDOG\_TOVALH)

本节说明 WDOG 定时溢出值寄存器：高位(WDOG\_TOVALH)和低位(WDOG\_TOVALL)。WDOG\_TOVALH 和 WDOG\_TOVALL 包含 16 位数值，用于设置 WDOG 的定时溢出周期。

WDOG 计数器(WDOG\_CNTH 和 WDOG\_CNTL)与定时溢出值(WDOG\_TOVALH 和 WDOG\_TOVALL)不断地进行比较。若计数器达到定时溢出值，则 WDOG 会强制进行复位。

#### 注

不要将 0 写入 WDOG 定时溢出值寄存器，否则 WDOG 一直产生复位信号。

地址: 4005\_2000h 基准 + 4h 偏移 = 4005\_2004h

位 读写	7	6	5	4		3	2	1	0
	TOVALHIGH								
复位	0	0	0	0		0	0	0	0

#### WDOG\_TOVALH 字段描述

字段	描述
TOVALHIGH	定时溢出值的高位字节

### 16.2.6 WDOG 定时溢出值寄存器：低位 (WDOG\_TOVALL)

参见 WDOG\_TOVALH 寄存器说明。

#### 注

有读操作时，所有位复位为 0。

地址: 4005\_2000h 基准 + 5h 偏移 = 4005\_2005h

位 读写	7	6	5	4		3	2	1	0
	TOVALLOW								
复位	0	0	0	0		0	1	0	0

#### WDOG\_TOVALL 字段描述

字段	描述
TOVALLOW	定时溢出值的低位字节

### 16.2.7 WDOG 窗口寄存器：高位 (WDOG\_WINH)

本节说明 WDOG 窗口寄存器 高位(WDOG\_WINH)和低位(WDOG\_WINL)。窗口模式使能后 (WDOG\_CS2[WIN]置位)，WDOG\_WINH 和 WDOG\_WINL 决定刷新序列被视为有效的最早时间。参见 [WDOG 刷新机制](#) 部分。

WDOG\_WINH 和 WDOG\_WINL 必须小于 WDOG\_TOVALH 和 WDOG\_TOVALL。

地址: 4005\_2000h 基准 + 6h 偏移 = 4005\_2006h

位 读写	7	6	5	4		3	2	1	0
WINHIGH									
复位	0	0	0	0		0	0	0	0

**WDOG\_WINH 字段描述**

字段	描述
WINHIGH	WDOG 窗口的高位字节

### 16.2.8 WDOG 窗口寄存器：低位 (WDOG\_WINL)

参见 WDOG\_WINH 寄存器说明。

地址: 4005\_2000h 基准 + 7h 偏移 = 4005\_2007h

位 读写	7	6	5	4		3	2	1	0
WINLOW									
复位	0	0	0	0		0	0	0	0

**WDOG\_WINL 字段描述**

字段	描述
WINLOW	WDOG 窗口的低位字节

## 16.3 功能说明

WDOG 模块提供故障安全机制，确保系统在出现故障时（比如 CPU 时钟停摆或软件代码出现跑飞）能够复位至已知状态。WDOG 计数器采用选定的时钟源工作，并监测是否定期得到更新（喂狗）。若非如此，它便会复位系统。

定时溢出周期、窗口模式和时钟源均可以编程配置，但必须在复位后的 128 个总线时钟内进行配置。

### 16.3.1 WDOG 刷新机制

若 WDOG 计数器未得到刷新，则 WDOG 会复位 MCU。鲁棒性刷新机制使跑飞的代码不太可能刷新 WDOG。

要刷新 WDOG 计数器，软件必须在定时溢出周期终止前执行刷新写序列。此外，如果使用了窗口模式，那么只有在 WDOG\_WINH 和 WDOG\_WINL 的窗口时间值设置好之后，软件才能开始执行刷新序列。请参见下图。

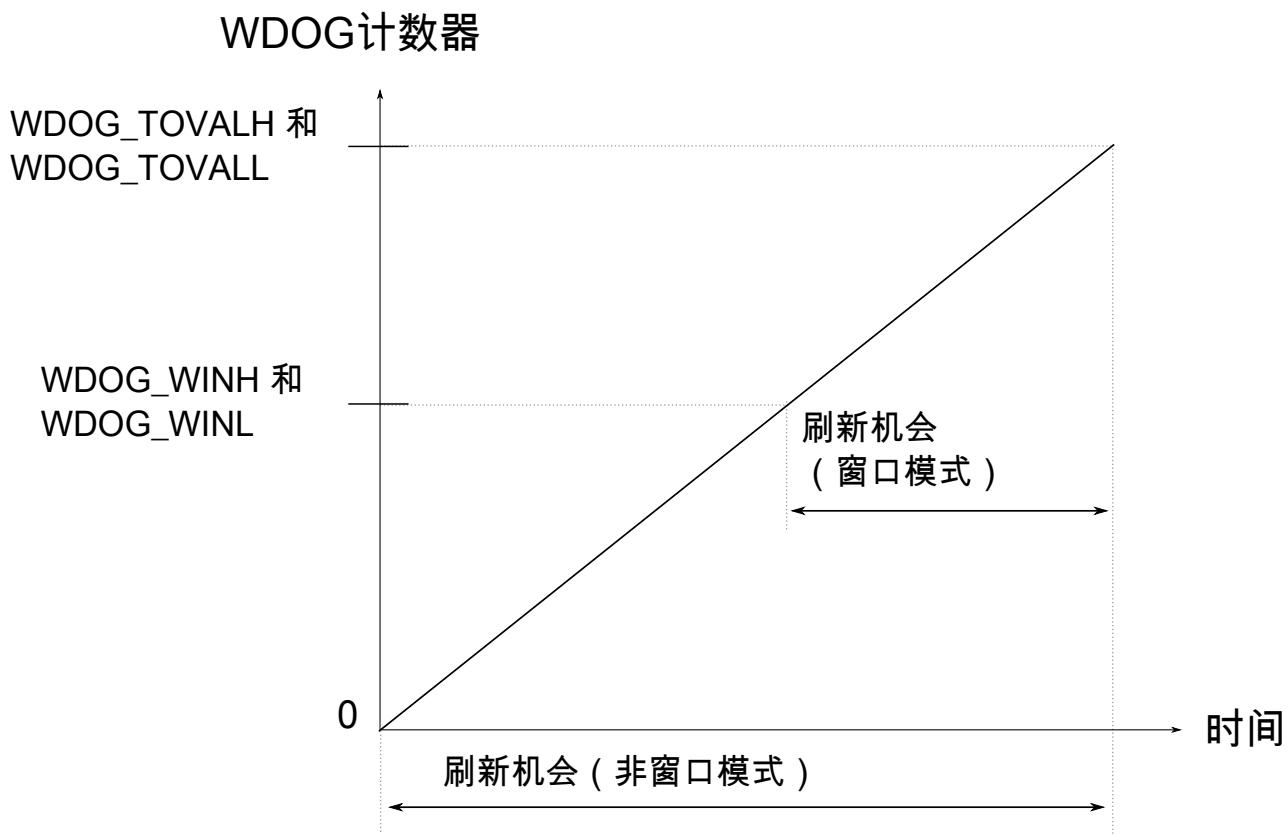


图 16-2. WDOG 计数器的刷新机会

#### 16.3.1.1 窗口模式

软件比预期更快地完成主控制循环，则表明可能存在系统问题。根据用户应用的具体要求，可以配置 WDOG，在提前刷新 WDOG 的时候，强制系统复位。

窗口模式使能时，必须在计数器达到最小窗口时间值后刷新 WDOG；否则，WDOG 会复位 MCU。最小窗口时间值在 WDOG\_WINH:L 寄存器中进行设置。设置 CS1[WIN] 即可启用窗口模式。

### 16.3.1.2 刷新 WDOG

刷新写入序列为：先将 0x02A6 写入 WDOG\_CNTH 和 WDOG\_CNTL 寄存器，然后再将 0x80B4 写入。写入 0x80B4 的操作必须在写入 0x02A6 后的 16 个总线时钟内执行；否则，看门狗会复位 MCU。

#### 注

在开始执行刷新序列前，先禁用全局中断。否则，如果产生中断，写入四个字节所需的时间就会长于 16 个总线周期，从而刷新序列失效。在刷新序列完成后重新使能全局中断。

### 16.3.1.3 示例代码：刷新 WDOG

下面的代码段显示了 WDOG 模块的刷新写序列。

#### 注

下列示例代码的组合情况为：8 位 WDOG\_CNTH 和 WDOG\_CNTL 组合为 16 位 WDOG\_CNT；8 位 WDOG\_TOVALH 和 WDOG\_TOVALL 组合为 16 位 WDOG\_TOVAL；WDOG\_WINH 和 WDOG\_WINL 组合为 WDOG\_WIN，并采用 16 位访问。

```
/* Refresh watchdog */

for (;;) // main loop
{
    ...
    DisableInterrupts; // disable global interrupt
    WDOG_CNT = 0x02A6; // write the 1st refresh word
    WDOG_CNT = 0x80B4; // write the 2nd refresh word to refresh counter
    EnableInterrupts; // enable global interrupt
    ...
}
```

### 16.3.2 配置 WDOG

所有 WDOG 控制位、定时溢出值和窗口值都只能在复位后一次写入。这表示，除非复位，否则它们的内容在写操作后无法更改。这就为配置 WDOG 提供了可靠的机制，确保软件跑飞时无法误禁用或误修改已配置过的 WDOG。

用户首先配置窗口和定时溢出值，然后再配置其他控制位并确保 CS1[UPDATE]也置 0，从而实现上述功能。该新配置仅在复位后一次写入除 WDOG\_CNT:H:L 外的所有寄存器后生效。否则，WDOG 默认使用复位值。如果未使用窗口模式(CS2[WIN]为 0)，那么不需要将数值写入 WDOG\_WINH:L 来使新配置生效。

### 16.3.2.1 重新配置 WDOG

某些情况下（比如支持 bootloader 功能时），用户可能想要在不先强制复位的情况下重新配置或禁用 WDOG。这时可以通过复位后进行 WDOG 初始配置时将 CS1[UPDATE]置 1，从而使用户可采用执行解锁序列的方法，在任意时刻重新配置 WDOG。（相反，如果 CS1[UPDATE]保持 0，重新配置 WDOG 的唯一方法是发起复位。）解锁序列与刷新序列类似，但使用不同的数值。

### 16.3.2.2 解锁 WDOG

解锁序列是指在配置完 WDOG 之后，在 16 个总线时钟内先对 WDOG\_CNT:H:L 寄存器写入 0x20C5，然后再写入 0x28D9。完成解锁序列后，用户必须在 128 个总线时钟内重新配置 WDOG；。

#### 注

由于重新配置 WDOG 有 128 个总线时钟要求，因此在重新配置 WDOG 之后、执行 STOP 或 WAIT 指令之前必须插入一定的延迟。这样可确保 WDOG 的新配置在 MCU 进入低功耗模式之前生效。否则，可能无法将 MCU 从低功耗模式中唤醒。

### 16.3.2.3 代码示例：重新配置 WDOG

下面的代码段显示了 WDOG 模块的重新配置示例。

```
/* Initialize watchdog with ~1-kHz clock source, ~1s time-out */

DisableInterrupts; // disable global interrupt

WDOG_CNT = 0x20C5; // write the 1st unlock word
WDOG_CNT = 0x28D9; // write the 2nd unlock word

WDOG_TOVAL = 1000; // setting timeout value
WDOG_CS2 = WDOG_CS2_CLK_MASK; // setting 1-kHz clock source
WDOG_CS1 = WDOG_CS1_EN_MASK; // enable counter running
EnableInterrupts; // enable global interrupt
```

### 16.3.3 时钟源

WDOG 计数器有四个时钟源选项，可以通过对 CS2[CLK]进行编程来选择：

- 总线时钟
- 运行频率约为 1 kHz 的内部低功耗振荡器(LPO) (为默认源)
- 内部 32 kHz 时钟
- 外部时钟

这些选项允许软件选择独立于总线时钟的时钟源，以满足安全要求更严格的应用需要。如果总线时钟由于某些原因而暂停，那么使用非总线时钟的时钟源可确保 WDOG 计数器继续运行；参见[备用复位](#)。

可用于所有时钟源的分频数固定的可选预分频器可以实现较长的定时溢出周期。CS2[PRES]置位后，时钟源先进行 256 倍预分频，随后对 WDOG 计数器进行计时。

下表总结了可用的不同 WDOG 定时溢出周期。

**表 16-1. WDOG 定时溢出可用性**

基准时钟	预分频器	WDOG 定时溢出可用性
内部 约 1 kHz (LPO)	直通式	约 1 ms–65.5 s <sup>1</sup>
	÷256	约 256 ms–16,777 s
内部 约 32 kHz	直通式	约 31.25 µs–2.048 s
	÷256	约 8 ms–524.3 s
1 MHz (来自总线或外部)	直通式	1 µs–65.54 ms
	÷256	256 µs–16.777 s
20 MHz (来自总线或外部)	直通式	50 ns–3.277 ms
	÷256	12.8 µs–838.8 ms

1. 复位后的默认定时溢出值约为 4 ms。

#### 注

当程序员在重新配置期间切换时钟源时，WDOG 硬件将计数器保持在零，持续时间为配置时间周期（128 个总线周期）后，保持上一个时钟源的 2.5 个周期和新时钟源的 2.5 个周期。该延迟可确保采用新配置重新启动计数器前的平稳过渡。

### 16.3.4 使用中断延迟复位

中断使能( $\text{CS1[INT]} = 1$ )时，WDOG 在发生复位触发事件（如计数器定时溢出或无效刷新尝试）时首先生成中断请求。WDOG 延迟 128 个总线时钟进行强制复位，从而允许中断服务程序(ISR)执行任务，如分析堆栈以调试代码。

中断禁用( $\text{CS1[INT]} = 0$ )时，WDOG 不会延迟进行强制复位。

### 16.3.5 备用复位

#### 注

必须使用总线时钟以外的其它时钟源作为计数器的基准时钟；否则，备用复位功能不可用。

备用复位功能是一种安全保护特性，在 WDOG 主逻辑丢失其时钟（总线时钟）而无法再监控计数器时，独立生成复位。如果 WDOG 计数器连续两次溢出（没有强制复位），那么备用复位功能生效并生成复位。

### 16.3.6 Debug 模式及低功耗模式下的功能

默认情况下，WDOG 在 Active Background 模式、Wait 模式或 Stop 模式下不工作。但是，WDOG 在如下这些模式中仍然工作：

- 对于 Active Background 模式，置位 CS1[DBG]。（通过这种方式，即使 CPU 状态由调试模块保持，WDOG 也可在 Active Background 模式下工作。）
- 对于 Wait 模式，置位 CS1[WAIT]。
- 对于 Stop 模式，置位 CS1[STOP]。

#### 注

WDOG 无法在 Stop 模式下生成中断——即使 CS1[STOP] 已置位——且不会将 MCU 从 Stop 模式中唤醒。它可以在 Stop 模式期间生成复位。

对于 Active Background 模式和 Stop 模式，除了上述配置外，必须使用总线时钟以外的其他时钟源作为计数器的基准时钟；否则，WDOG 无法运行。

## 16.3.7 WDOG 快速测试

在安全性能很关键的应用中，执行应用程序代码之前，用户需要测试 WDOG 是否按预期进行工作并复位 MCU。对 16 位计数器进行完全测试的方法是使其运行至溢出值，这会花费相对较长的时间（64 k 个时钟周期）。

为了最大限度降低复位后应用程序代码的启动延迟，WDOG 提供了一种测试方法，即通过将计数器分成字节宽度的各级，来更快速地测试 WDOG。低位和高位字节独立运行，并针对定时溢出值寄存器的对应字节执行定时溢出测试。（为了全面测试计数器的高位字节，测试时通过低位字节的第 8 位将输入时钟送入，从而确保低位字节到高位字节的进位溢出性能被测试。）

使用该测试特性可将测试时间缩短至 512 个时钟（不包括用户配置和复位向量提取等开销）。为进一步加快测试速度，可以使用速度更快的时钟（如总线时钟）作为计数器基准。

上电复位时，系统复位寄存器中的 POR 位置位，指示用户应当执行 WDOG 快速测试。

### 16.3.7.1 测试计数器的每一个字节

测试步骤如下：

1. WDOG 配置时间周期内，在 WDOG\_TOVALH 和 WDOG\_TOVALL 寄存器中写入设计的数值。
2. 选择待测试的计数器的字节：对于低位字节，使用 WDOG\_CS1[TST] = 10b；对于高位字节，使用 WDOG\_CS1[TST] = 11b。
3. 等待 WDOG 出现定时溢出。或者在闲置循环中，使 RAM 位置增量，以便用作后续比较的并行软件计数器。由于 RAM 不受 WDOG 复位影响，因此 WDOG 计数器的定时溢出周期能与软件计数器作比较，以验证定时溢出周期是否如预期发生。
4. WDOG 计数器定时溢出并强制复位。
5. 确认系统复位寄存器中的 WDOG 标志已置位，表明 WDOG 导致复位。（POR 标志位保持清零状态。）
6. 确认 WDOG\_CS1[TST]显示测试（10b 或 11b）已执行。

若得到确认，对选定的字节执行计数功能和比较功能。重复该流程，选择步骤 2 中的另一字节。

#### 注

WDOG\_CS1[TST]仅通过 POR 清零，并且不受其他复位影响。

### 16.3.7.2 进入用户模式

顺利完成对 WDOG 计数器低位和高位字节的测试后，用户可将 WDOG\_CS1[TST] 配置为 01b，以指示 WDOG 准备用于应用用户模式。因此，如果再次发生复位，软件可将复位触发视为软件跑飞或故障应用代码导致的真正 WDOG 复位。

作为一个持续进行的测试，在使用默认的 1 kHz 时钟源时，软件可定期对 WDOG\_CNTH 和 WDOG\_CNTL 寄存器进行读操作以确保计数器处于递增状态。



# 第 17 章 位操作引擎(BME)

## 17.1 简介

位操作引擎(BME)针对基于 Cortex-M0+的微控制器中对外设地址空间的读取-修改-写入存储器原子操作提供硬件支持。

这种架构能力亦称为“已修饰存储(decorated storage)”，因其定义了一种机制，可为外设的加载和存储(load-store)操作提供额外的语义，而不只是对目标地址的读和写。在 BME 定义中，“修饰”(即额外的语义信息)经编码后进入外设地址，用于存储器基准。

通过组合 Cortex-M 指令集架构 (v6M, v7M) 的基本加载和存储指令，加上 BME 提供的已修饰存储概念，这种架构为低端微控制器提供了可靠且高效的读取-修改-写入能力。该内核平台所定义的架构能够对外设寄存器提供 n-bit 字段操作，并与内置的 C 语言标准 I/O 硬件寻址相一致。就大多数 BME 命令而言，单个内核读写总线周期转换为一个原子读取-修改-写入，即一个透明的“先读后写”总线序列。

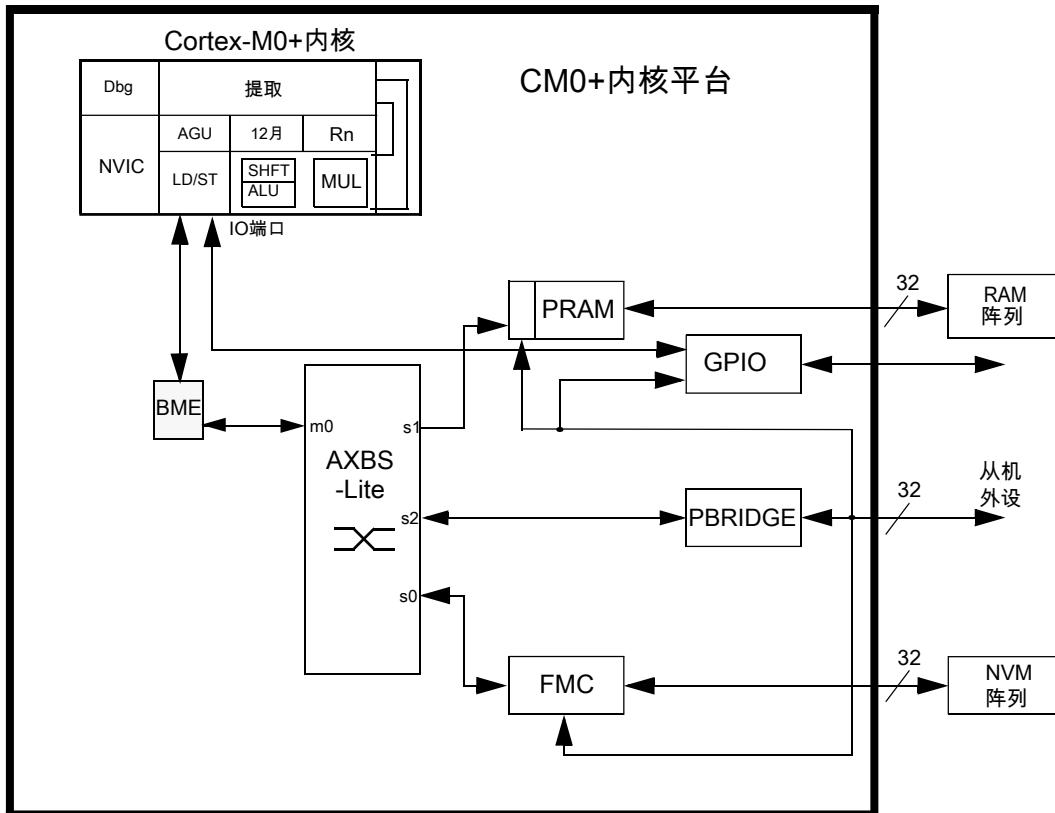
BME 已修饰基准只能用于处理器内核产生的系统总线操作，并且其目标地址应是以 0x4000\_0000 为基地址的 512KB 标准外设地址空间<sup>1</sup>或以 0x2000\_0000 为基地址的 SRAM\_U 空间。修饰语义嵌入地址位[28:19]，在地址 0x4400\_0000–0x5FFF\_FFFF 处为外设桥创建448MB 的空间；在地址 0x2400\_0000–0x3FFF\_FFFF 为 SRAM\_U 创建 448MB 的空间；这些位在发送至外设总线控制器的实际地址中被略去了，BME 用它们来定义并控制其操作。

### 17.1.1 概述

下图是该类超低端微控制器的处理器内核和平台的一般结构框图。

---

1. 为精确起见，外设地址空间占用 516 KB 区域：基地址为 0x4000\_0000 的 512 KB 加上基地址为 F\_F000 的 4 KB 空间，用于 GPIO 访问。此结构与 Kinetis K Family 兼容。尝试访问 0x4008\_0000 - 0x400F\_EFFF 之间的存储器空间会因地址非法而导致错误终止。



注意：只能通过内核访问BME。

图 17-1. Cortex-M0+内核平台结构框图

如该结构框图所示，BME 模块与交叉开关上的主端口连接，从而使其支持对 SRAM\_U (图中所示的平台 RAM(PRAM)) 和外设桥(PBRIDGE)控制器进行的读取-修改-写入(read-modify-wirte)的原子操作。BME 硬件的微架构采用二级流水线设计，符合 AMBA-AHB 系统总线接口协议。PBRIDGE 模块将 AHB 系统总线协议转换为 IPS/APB 协议，用于连接的从外设。

## 17.1.2 特性

BME 的主要特性包括：

- 针对选定地址空间进行的已修饰存储的轻量级实施
- 将附加访问语义编码到基准地址中
- 位于处理器内核和开关主端口之间
- 符合 AHB 系统总线协议的两级流水线设计
- 可将非修饰访问组合传输至从设备总线控制器
- 可将来自处理器内核的已修饰加载和存储转换为原子读取-修改-写入
- 已修饰加载支持无符号位字段提取、加载和{置位，清零} 1 位操作
- 已修饰存储支持位字段插入、逻辑“与”、“或”和“异或”操作

- 支持字节、半字和字大小的修饰操作
- 支持在 AHB 输出总线上执行最少的信号切换，以降低功耗

### 17.1.3 工作模式

BME 模块不支持任何特殊工作模式。作为位于交叉主 AHB 系统总线端口上的内存映射器件，BME 严格按照内存地址作出响应，以便访问 SRAM\_U 和外设桥总线控制器。

所有 BME 模块的相关功能都位于内核平台时钟域内，包括它与交叉主端口、SRAM\_U 和 PBRIDGE 总线控制器的连接。

## 17.2 存储器映像和寄存器定义

BME 模块提供存储器映像功能，且不包含任何编程模型寄存器。

[功能说明](#) 中将详细介绍 BME 支持的确切功能。

外设地址空间占据 516 KB 区域：基址为 0x4000\_0000 的 512 KB 加上基址为 0x400F\_F000 的 4 KB GPIO 访问空间；已修饰地址空间映射至位于 0x4400\_0000–0x5FFF\_FFFF 的 448 MB 区域。与 SRAM\_U 有关的已修饰地址空间为映射在 0x2400\_0000 - 0x3FFF\_FFFF 的 448 MB 区域。

## 17.3 功能说明

此处信息详细说明该 BME 支持的功能。

如前文所述，Cortex-M 指令集架构 (v6M, v7M) 的基本加载和存储指令组合加上 BME 提供的已修饰存储概念，这种实施方法为这类超低端微控制器提供了可靠且高效的读取-修改-写入能力。由该内核平台功能所定义的架构能力可在外设寄存器和 RAM 中实现 n 位字段操作，并与内置的 C 语言标准 I/O 硬件寻址相一致。就大多数 BME 命令而言，单个内核读写总线周期转换为一个原子读取-修改-写入，即一个透明的“先读后写”总线序列。

应当首先执行已修饰存储操作，然后执行已修饰加载。

## 17.3.1 BME 已修饰存储

BME 已修饰存储支持的功能包括三个逻辑运算符（AND、OR、XOR）以及一个位字段插入。

对于所有这些操作而言，BME 可将单个已修饰 AHB 存储事务转换为一个双周期原子读取-修改-写入序列，其中读取-修改组合操作在第一个 AHB 数据阶段执行，然后写入操作在第二个 AHB 数据阶段执行。

显示外设位字段插入操作的已修饰存储的一般时序示意图如下所示：

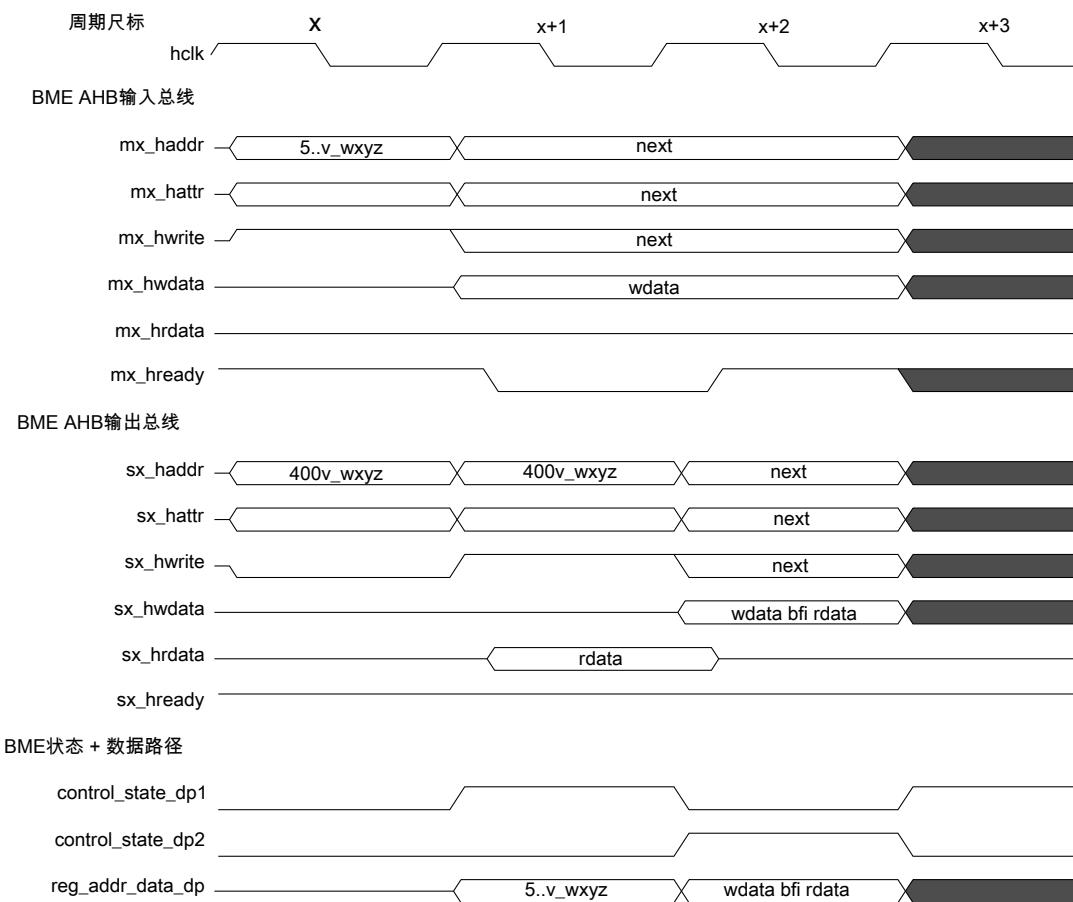


图 17-2. 已修饰存储：位字段插入时序示意图

所有已修饰存储操作遵循图 17-2 中所示的相同执行模板，双周期读取-修改-写入操作：

1. 周期 x，首个 AHB 地址阶段：来自输入总线的写操作被转换成输出总线上实际地址（修饰已移除）的读操作，且该地址被捕捉到一个寄存器中。
2. 周期 x+1，第二个 AHB 地址阶段：对捕捉的（实际）内存地址的写访问被输出。
3. 周期 x+1，首个 AHB 数据阶段：使用输入总线上的写入数据和修饰定义的功能对存储器读取数据进行修改，其结果被捕捉到一个数据寄存器中；输入总线周期停止。

4. 周期 x+2, 第二个 AHB 数据阶段: 捕捉到数据寄存器的数据被输出到输出写数据总线上。

### 注

由从设备插入的任何等待状态直接通过 BME 传回主设备输入总线, 从而逐周期停止 AHB 事务。

#### 17.3.1.1 已修饰存储逻辑与(AND)

该命令执行基准存储器位置的原子读取-修改-写入

1. 首先, 读取该位置;
2. 然后, 通过使用来自系统总线周期的写数据操作数执行逻辑与(AND)操作对其进行修改
3. 最后, AND 操作结果写回基准存储器位置。

数据尺寸由写入操作定义, 可以是字节(8位)、半字(16位)或字(32位)。进行字节和半字传输时, 内核执行所需的写入数据通道复制操作。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ioandb	0	*	*	0	0	1	-	-	-	-	-	-									mem_addr											
ioandh	0	*	*	0	0	1	-	-	-	-	-	-									mem_addr									0		
ioandw	0	*	*	0	0	1	-	-	-	-	-	-									mem_addr									0 0		

图 17-3. 已修饰存储地址: 逻辑与(AND)

参见图 17-3, 其中 addr[30:29] = 01 (SRAM\_U), addr[30:29] = 10(外设), addr[28:26]=001 (指定 AND 操作), mem\_addr[19:0]指定空间偏移地址 (对于 SRAM\_U, 基址为 0x2000\_0000; 对于外设, 基址为 0x4000\_0000)。"-”表示地址位无关。

已修饰 AND 写操作采用下列伪代码定义:

```
ioand<sz>(accessAddress, wdata)           // decorated store AND
tmp    = mem[accessAddress & 0xE00FFFFF, size] // memory read
tmp    = tmp & wdata                          // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp  // memory write
```

其中, 操作数大小<sz>定义为 b(字节, 8位)、h(半字, 16位)和w(字, 32位)。本文通篇使用这种标记方式。

在周期定义表中, 标记 AHB\_ap 和 AHB\_dp 表示 BME AHB 事务的地址和数据阶段。下表详细介绍各周期的 BME 操作。

表 17-1. 已修饰存储的周期定义：逻辑与(AND)

流水线级	周期		
	x	x+1	x+2
BME AHB_ap	地址转发到存储器; 解码修饰; 将 master_wt (主设备_写) 转换为 slave_rd (从设备_读); 捕捉地址, 属性	将捕捉到的地址以及属性作为 slave_wt (从设备_写) 转发给存储器	<下一个>
BME AHB_dp	<上一个>	执行存储器读取 在寄存器中形成形成(rdata & wdata)并捕捉目标数据	执行写操作, 将寄存器中的数据发送至存储器

### 17.3.1.2 已修饰存储逻辑或(OR)

该命令执行基准存储器位置的原子读取-修改-写入

- 首先, 读取该位置。
- 然后, 通过使用来自系统总线周期的写数据操作数执行逻辑或(OR)操作对其进行修改。
- 最后, OR 操作结果写回基准存储器位置。

数据尺寸由写入操作定义, 可以是字节 (8 位)、半字 (16 位) 或字 (32 位)。进行字节和半字传输时, 内核执行所需的写入数据通道复制操作。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ioorb	0	*	*	0	1	0	-	-	-	-	-	-									mem_addr											
ioorh	0	*	*	0	1	0	-	-	-	-	-	-									mem_addr									0		
ioorw	0	*	*	0	1	0	-	-	-	-	-	-									mem_addr									0		

图 17-4. 已修饰地址存储: 逻辑或(OR)

参见图 17-4, 其中 addr[30:29] = 01 (SRAM\_U), addr[30:29] = 10 (外设), addr[28:26] = 010 (指定 OR 操作), mem\_addr[19:0]指定空间偏移地址 (对于 SRAM\_U, 基地址为 0x2000\_0000; 对于外设, 基地址为 0x4000\_0000)。"-" 表示地址位无关。

已修饰 OR 写操作采用下列伪代码定义:

```
ioor<sz>(accessAddress, wdata) // decorated store OR
tmp    = mem[accessAddress & 0xE00FFFFF, size] // memory read
tmp    = tmp | wdata // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp // memory write
```

逐周期 BME 操作详情见下表。

表 17-2. 已修饰存储的周期定义：逻辑或(OR)

流水线级	周期		
	x	x+1	x+2
BME AHB_ap	地址转发到存储器; 解码修饰; 将 master_wt (主设备_写) 转换为 slave_rd (从设备_读); 捕捉地址, 属性	将捕捉到的地址以及属性作为 slave_wt (从设备_写) 转发给存储器	<下一个>
BME AHB_dp	<上一个>	执行存储器读取 在寄存器中形成(rdata   wdata)并捕捉目标数据	执行写操作, 将寄存器中的数据发送至存储器

### 17.3.1.3 已修饰存储逻辑异或(XOR)

该命令执行基准存储器位置的原子读取-修改-写入

- 首先, 读取该位置。
- 然后, 通过使用来自系统总线周期的写数据操作数执行逻辑异或(XOR)操作对其进行修改。
- 最后, XOR 操作结果写回基准存储器位置。

数据尺寸由写入操作定义, 可以是字节 (8 位)、半字 (16 位) 或字 (32 位)。进行字节和半字传输时, 内核执行所需的写入数据通道复制操作。

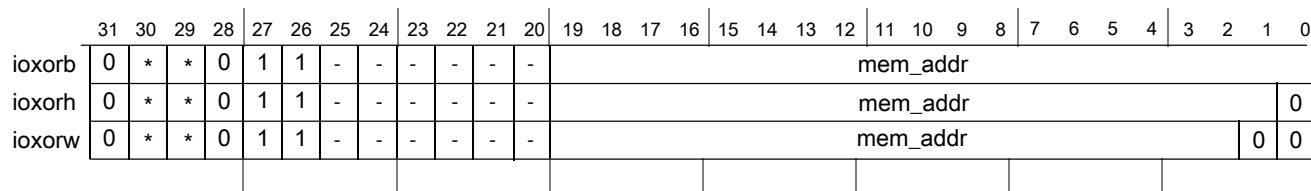


图 17-5. 已修饰地址存储: 逻辑异或(XOR)

参见图 17-5, 其中 addr[30:29] = 01 (SRAM\_U), addr[30:29] = 10 (外设), addr[28:26] = 011 (指定 XOR 操作), mem\_addr[19:0]指定外设空间偏移地址 (对于 SRAM\_U, 基址为 0x2000\_0000; 对于外设, 基址为 0x4000\_0000)。"-" 表示地址位无关。

已修饰 XOR 写操作采用下列伪代码定义:

```
ioxor<sz>(accessAddress, wdata) // decorated store XOR
tmp = mem[accessAddress & 0xE00FFFFF, size] // memory read
tmp = tmp ^ wdata // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp // memory write
```

逐周期 BME 操作详情见下表。

表 17-3. 已修饰存储的周期定义：逻辑 XOR

流水线级	周期		
	x	x+1	x+2
BME AHB_ap	地址转发到存储器; 解码修饰; 将 master_wt (主设备_写) 转换为 slave_rd (从设备_读); 捕捉地址, 属性	将捕捉到的地址以及属性作为 slave_wt (从设备_写) 转发给存储器	<下一个>
BME AHB_dp	<上一个>	执行存储器读取 在寄存器中形成(rdata ^ wdata)并捕捉目标数据	执行写操作, 将寄存器中的数据发送至存储器

### 17.3.1.4 已修饰存储位字段插入(BFI)

该命令使用一个原子读取-修改-写入序列, 将由 LSB(b)和位字段宽度(w+1)定义的写数据操作数中所含的位字段插入由与存储指令有关的访问尺寸定义的存储器“容器”中。

数据尺寸由写入操作定义, 可以是字节 (8 位)、半字 (16 位) 或字 (32 位)。

#### 注

就字大小的操作而言, 最大位字段宽度为 16 位。进行字节和半字传输时, 内核执行所需的写入数据通道复制操作。

BFI 操作可用于将单个位插入外设。就本例而言, w 字段置 0, 表示位字段宽度为 1。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
iobfib	0	*	*	1	-	-	b	b	b	b	-	w	w	w	w																		mem_addr
iobfih	0	*	*	1	-	b	b	b	b	w	w	w	w	w	w																		mem_addr
iobfiw	0	*	*	1	b	b	b	b	b	w	w	w	w	w	w																		mem_addr

图 17-6. 修饰地址存储: 位字段插入

其中,  $\text{addr}[30:29] = 01$  (SRAM\_U),  $\text{addr}[30:29] = 10$  (外设),  $\text{addr}[28] = 1$  (表示 BFI 操作),  $\text{addr}[27:23] = "b"$  (LSB 标识符),  $\text{addr}[22:19] = "w"$  (位字段宽度减 1 标识符),  $\text{addr}[18:0]$ 指定外设空间偏移地址 (对于 SRAM\_U, 基址为 0x2000\_0000; 对于外设, 基址为 0x4000\_0000)。"-" 表示地址位无关。注意, 与其他已修饰保存操作不同, BFI 使用  $\text{addr}[19]$ 作为"w"指示符中的最低有效位, 而非地址位。

已修饰 BFI 写操作采用下列伪代码定义:

```
iofbfi<sz>(accessAddress, wdata) // decorated bit field insert
tmp = mem[accessAddress & 0xE007FFFF, size] // memory read
```

```

mask  = ((1 << (w+1)) - 1) << b           // generate bit mask
tmp   = tmp & ~mask                           // modify
      | wdata & mask
mem[accessAddress & 0xE007FFFF, size] = tmp    // memory write

```

与存储指令有关的写数据操作数(wdata)包含待插入的位字段。它必须在右对齐的容器内正确排列，即字节操作为最低 8 位内，半字操作为最低 16 位内，字操作为整个 32 位内。

为便于说明，举例如下：向 8 位存储器容器中插入 3 位字段"xyz"，其初始设置为"abcd\_efgh"。在所有情况下，w 都为 2，表示位字段宽度是 3。

```

if b = 0 and the decorated store (strb) Rt register[7:0] = -----_xyz,
then destination is "abcd_efxyz"
if b = 1 and the decorated store (strb) Rt register[7:0] = -----_xyz|,
then destination is "abcd_xyzh"
if b = 2 and the decorated store (strb) Rt register[7:0] = ---x_yz--,
then destination is "abcx_yzgh"
if b = 3 and the decorated store (strb) Rt register[7:0] = --xy_z---,
then destination is "abxy_zfgh"
if b = 4 and the decorated store (strb) Rt register[7:0] = -xyz_-----,
then destination is "axyz_efgh"
if b = 5 and the decorated store (strb) Rt register[7:0] = xyz-_-----,
then destination is "xyzd_efgh"
if b = 6 and the decorated store (strb) Rt register[7:0] = yz--_-----,
then destination is "yzcd_efgh"
if b = 7 and the decorated store (strb) Rt register[7:0] = z---_-----,
then destination is "zbcd_efgh"

```

该例中，需注意若起始位的位置加上字段宽度超出存储器大小，则仅向目标存储器位置插入一部分位字段源。若状态不同，假设 $(b + w+1) > \text{container\_width}$ ，则实际只有低阶" $\text{container\_width} - b$ "位被插入。

下表详细介绍各周期的 BME 操作。

表 17-4. 已修饰存储的周期定义：位字段插入

流水线级	周期		
	x	x+1	x+2
BME AHB_ap	地址转发到存储器；解码修饰；将 master_wt (主设备_写) 转换为 slave_rd (从设备_读)；捕捉地址，属性	将捕捉到的地址以及属性作为 slave_wt (从设备_写) 转发给存储器	<下一个>
BME AHB_dp	<上一个>	执行存储器读取；形成位屏蔽；在寄存器中形成按位数据 $((\text{mask}) ? \text{wdata} : \text{rdata})$ 并捕捉目标数据	执行写操作，将寄存器中的数据发送至存储器

### 17.3.2 BME 已修饰加载

BME 已修饰加载支持的功能包括：单 bit 加载和置位操作，单 bit 加载和清零操作，以及无符号位字段提取。

对于 2 个加载-{置位, 清零}操作, BME 可将单个已修饰 AHB 加载事务转换为一个双周期原子读取-修改-写入序列, 其中读取-修改组合操作在首个 AHB 数据阶段执行, 然后写入操作在第二个 AHB 数据阶段执行, 此时初始读取数据已返回至处理器内核。对于无符号位字段提取, 已修饰加载事务将在 BME 中停止一个周期——因为数据字段已被提取——然后在第二个 AHB 数据阶段进行对齐并返回至处理器。这是唯一一个非原子读取-修改-写入操作的已修饰事务, 因为它只是一次简单的数据读取。

显示外设加载-置位 1 位操作的已修饰加载的一般时序示意图如下所示。

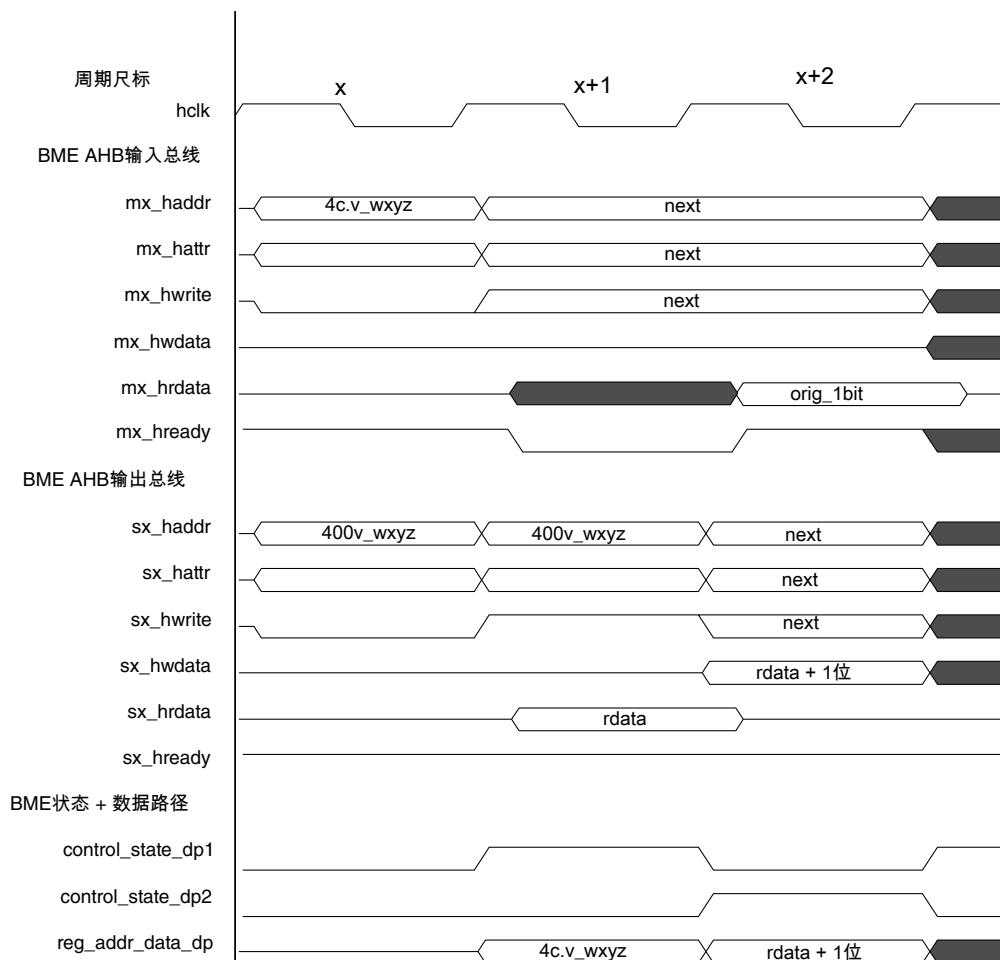


图 17-7. 已修饰加载: 加载-置位 1 位字段插入时序示意图

已修饰加载-{置位, 清零}1 位操作遵循上图所示的执行模板: 双周期读取-修改-写入操作:

1. 周期 x, 首个 AHB 地址阶段: 根据实际存储器地址 (修饰已移除) 将从输入总线的读取转换为输出总线上的读操作, 然后该地址被捕捉到一个寄存器中
2. 周期 x+1, 第二个 AHB 地址阶段: 对捕捉的 (实际的) 存储器地址的写访问被输出

3. 周期  $x+1$ , 首个 AHB 数据阶段: 在寄存器中捕捉“原始”1 位存储器读取数据, 根据修饰 (采用寄存器中捕捉到的已修改数据) 定义的功能进行 1 位字段的置位或清零; 输入总线周期停止
4. 周期  $x+2$ , 第二个 AHB 数据阶段: 将选定的原始 1 位数据右对齐、零填充, 然后驱动至输入读取数据总线上, 而已寄存的写入数据被发送到输出写数据总线上

### 注

由从设备插入的任何等待状态直接通过 BME 传回主设备输入总线, 从而逐周期停止 AHB 事务。

显示无符号外设位字段操作的已修饰加载的一般时序示意图如下图所示。

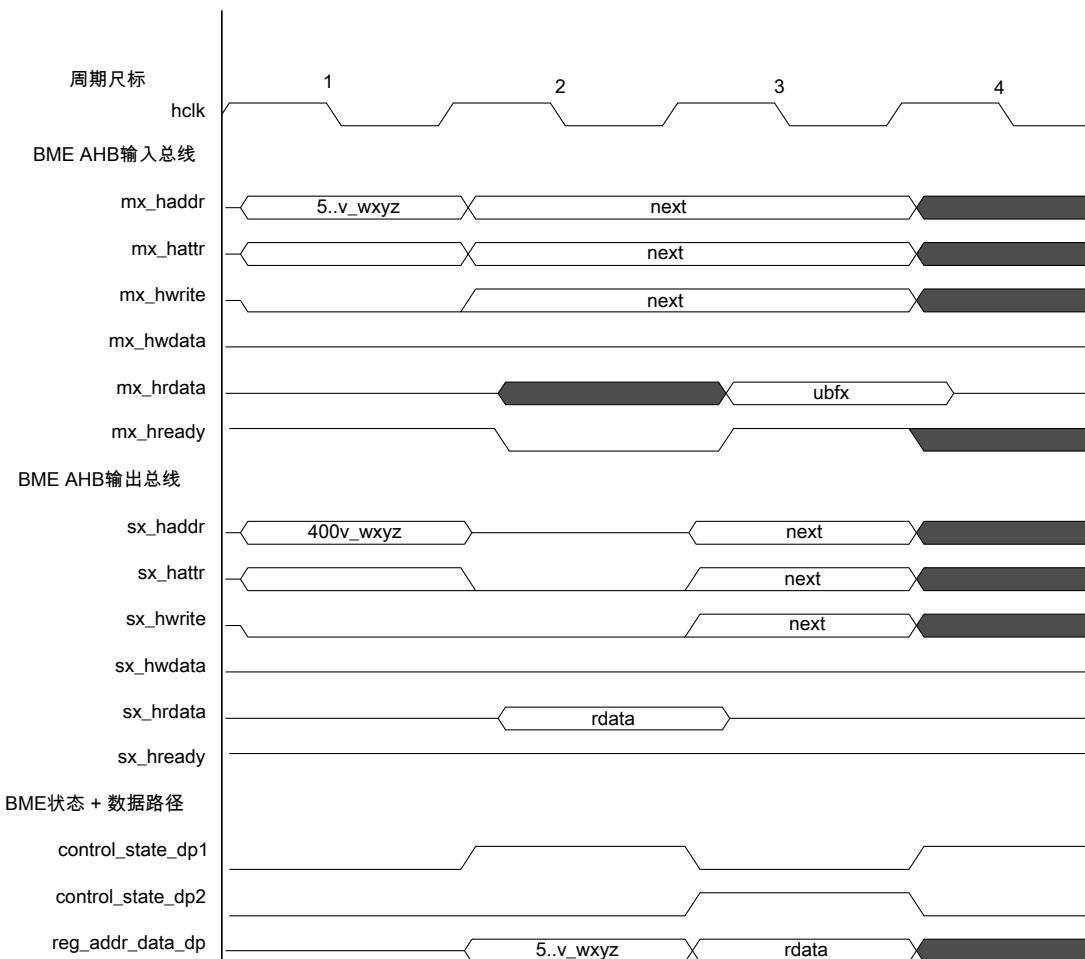


图 17-8. 已修饰加载: 无符号位字段提取时序示意图

已修饰无符号位字段提取遵循与上图中双周期读取操作相同的执行模板:

- 周期  $x$ , 首个 AHB 地址阶段: 根据实际存储器地址 (修饰已移除) 将从输入总线的读取转换为输出总线上的读操作, 然后该地址被捕捉到一个寄存器中。
- 周期  $x+1$ , 第二个 AHB 地址阶段: 空闲周期

- 周期 x+1, 第一个 AHB 数据阶段: 根据起始位的位置和字段宽度生成一个位屏蔽; 屏蔽位与存储器读取数据之间执行 AND 运算以隔离位字段; 计算结果被捕获到一个数据寄存器中; 输入总线周期停止
- 周期 x+2, 第二个 AHB 数据阶段: 将已寄存数据逻辑右对齐并驱动至输入读取数据总线

### 注

由从设备插入的任何等待状态直接通过 BME 传回主设备输入总线, 从而逐周期停止 AHB 事务。

#### 17.3.2.1 已修饰加载: 1 位加载-置位(LAS1)

该命令将 LSB 位(b)定义的 1 位字段加载至内核通用目标寄存器(Rt), 并在执行原子读取-修改-写入序列后将存储器空间中的这 1 位置位。

从存储器地址中提取的 1 位数据字段在返回至内核的操作数中右对齐和零填充。

数据尺寸由读取操作指定, 可以是字节 (8 位)、半字 (16 位) 或字 (32 位)。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
iolaslb	0	*	*	0	1	1	-	-	b	b	b	-																					mem_addr	
iolaslh	0	*	*	0	1	1	-	b	b	b	b	-																					mem_addr	0
iolaslw	0	*	*	0	1	1	b	b	b	b	b	-																					mem_addr	0 0

图 17-9. 已修饰加载地址: 1 位加载-置位

其中,  $\text{addr}[30:29] = 01$  (SRAM\_U),  $\text{addr}[30:29] = 10$  (外设),  $\text{addr}[28:26] = 011$  (指定 1 位加载 - 置位操作),  $\text{addr}[25:21] = "b"$  (位标识符),  $\text{mem\_addr}[19:0]$ 指定空间偏移地址 (对于 SRAM\_U, 基地址为 0x2000\_0000; 对于外设, 基地址为 0x4000\_0000)。"-" 表示地址位无关。

已修饰 1 位加载-置位读操作采用下列伪代码定义:

```
rdata = iolas1<sz>(accessAddress)           // decorated load-and-set 1
tmp   = mem[accessAddress & 0xE00FFFFF, size] // memory read
mask  = 1 << b                            // generate bit mask
rdata = (tmp & mask) >> b                  // read data returned to core
tmp   = tmp | mask                           // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp // memory write
```

逐周期 BME 操作详情见下表。

表 17-5. 已修饰加载的周期定义：1 位加载-置位

流水线级	周期		
	x	x+1	x+2
BME AHB_ap	地址转发到存储器; 解码修饰; 捕捉地址, 属性	将捕捉到的地址以及属性作为 slave_wt (从设备_写) 转发给存储器	<下一个>
BME AHB_dp	<上一个>	执行存储器读取; 形成位屏蔽从 rdata 提取位; 在寄存器中形成(rdata   mask)并捕捉目标数据	将已提取位返回至主设备; 执行写操作, 从而将已寄存的数据发送到存储器

### 17.3.2.2 已修饰加载无符号位字段提取(UBFX)

该命令使用一个双周期读取序列, 从与加载指令有关的访问尺寸所定义的存储器“容器”中提取 LSB 位(b)定义的位字节和位字段宽度(w+1)。

从存储器地址中提取的位字段在返回至内核的操作数中右对齐和零填充。如前文所述, 这是唯一一个不执行存储器写入的已修饰操作, 即 UBFX 仅执行读取操作。

数据尺寸由写入操作定义, 可以是字节 (8 位)、半字 (16 位) 或字 (32 位)。注意, 就字大小的操作而言, 最大位字段宽度为 16 位。

建议使用 UBFX 操作提取单个位。就本例而言, w 字段置 0, 表示位字段宽度为 1。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ioubfxb	0	*	*	1	-	-	b	b	b	-	w	w	w																				mem_addr
ioubfxh	0	*	*	1	-	b	b	b	b	w	w	w	w																			0	mem_addr
ioubfxw	0	*	*	1	b	b	b	b	b	w	w	w	w																		0	0	mem_addr

图 17-10. 已修饰加载地址: 无符号位字段提取

参见图 17-10, 其中: addr[30:29] = 01 (SRAM\_U), addr[30:29] = 10 (外设), addr[28]=1 (指定无符号位字段提取操作), addr[27:23] = "b" (LSB 标识符), addr[22:19] = "w" (位字段宽度减 1 标识符), mem\_addr[18:0]指定空间偏移地址 (对于 SRAM\_U, 基址为 0x2000\_0000; 对于外设, 基址为 0x4000\_0000。"-表示地址位无关。注意, 与其他已修饰加载操作不同, UBFX 使用 addr[19]作为"w"指示器中的最低有效位, 而非地址位。

已修饰无符号位字段提取读操作采用下列伪代码定义:

```
rdata = ioubfx<sz>(accessAddress)           // unsigned bit field extract
tmp   = mem[accessAddress & 0xE007FFFF, size] // memory read
mask  = ((1 << (w+1)) - 1) << b           // generate bit mask
rdata = (tmp & mask) >> b                   // read data returned to core
```

像 BFI 操作一样，如果起始位的位置加上字段宽度超出容器大小，则仅从目标存储器单元提取源位字段的一部分。也就是说，如果  $(b + w + 1) > \text{container\_width}$ ，则实际只有低阶 " $\text{container\_width} - b$ " 位被提取。下表详细介绍各周期的 BME 操作。

表 17-6. 已修饰加载的周期定义：无符号位字段提取

流水线级	周期		
	x	x+1	x+2
BME AHB_ap	地址转发到存储器；解码修饰；捕捉地址，属性	空闲 AHB 地址阶段	<下一个>
BME AHB_dp	<上一个>	执行存储器读取；形成位屏蔽，在寄存器中形成(rdata & mask)并捕捉目标数据	已寄存的数据逻辑右移，将对齐的 rdata 返回至主机

### 17.3.3 已修饰地址和 GPIO 访问的更多详情

如前文所述，外设地址空间占用 516 KB 区域：基地址为 0x4000\_0000 的 512 KB 加上基地址为 0x400F\_F000 的 4 KB 空间，用于 GPIO 访问。此存储器布局与 Kinetis K Family 兼容，可提供 129 个地址插槽，每个大小为 4 KB。

GPIO 地址空间由硬件映射到多个地址：它在“标准”系统地址 0x400F\_F000 处出现，并位于与地址 0x4000\_F000 对应的地址插槽中。已修饰加载和数据存储使访问 GPIO 的情况稍显复杂。地址[19]的使用随修饰运算而有所不同；对于 AND、OR、XOR、LAC1 和 LAS1，此位相当于真正的地址位，而对于 BFI 和 UBX，此位定义了“w”位字段说明符中最不重要的位。

因此，未修饰的 GPIO 访问和已修饰的 AND、OR、XOR、LAC1、LAS1 运算可使用标准 0x400F\_F000 基地址，而已修饰的 BFI 和 UBX 运算则必须使用替代的 0x4000\_F000 基地址。另一种实施可以只将 0x400F\_F000 用作所有未修饰 GPIO 访问的基地址，以及将 0x4000\_F000 用作所有已修饰访问的基地址。两种实施都有硬件支持

表 17-7. 已修饰外设和 GPIO 地址详情

外设地址空间	说明
0x4000_0000–0x4007_FFFF	未修饰（正常）外设访问
0x4008_0000–0x400F_EFFF	非法地址；尝试的访问被终止并且以错误结束
0x400F_F000–0x400F_FFFF	使用标准地址的未修饰（正常）GPIO 访问
0x4010_0000–0x43FF_FFFF	非法地址；尝试的访问被终止并且以错误结束
0x4400_0000–0x4FFF_FFFF	以 0x4000_F000 或 0x400F_F000 为基地址对外设和 GPIO 的已修饰 AND、OR、XOR、LAC1、LAS1 操作。
0x5000_0000–0x5FFF_FFFF	以 0x4000_F000 为基地址对外设和 GPIO 的已修饰 BFI、UBFX 操作。

## 17.4 应用信息

在本节中，以带 C 语言表达式操作数的 GNU 汇编器宏为例介绍执行已修饰运算所需的指令。

本节专门介绍 bme.h 文件的一部分，该文件定义的是已修饰逻辑存储的汇编语言表达式：AND、OR 和 XOR。BFI 和已修饰加载的对应函数更加复杂，可在完整的 BME 头文件中查看。

这些宏使用[功能说明](#) 中介绍的相同的函数名。

```
#define IOANDW(ADDR, WDATA)          \
    __asm("ldr    r3, =(1<<26);"      \
          "orr    r3, %[addr];"        \
          "mov    r2, %[wdata];"       \
          "str    r2, [r3];"           \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");\n\n#define IOANDH(ADDR, WDATA)          \
    __asm("ldr    r3, =(1<<26);"      \
          "orr    r3, %[addr];"        \
          "mov    r2, %[wdata];"       \
          "strh   r2, [r3];"           \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");\n\n#define IOANDB(ADDR, WDATA)          \
    __asm("ldr    r3, =(1<<26);"      \
          "orr    r3, %[addr];"        \
          "mov    r2, %[wdata];"       \
          "strb   r2, [r3];"           \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");\n\n#define IOORW(ADDR, WDATA)           \
    __asm("ldr    r3, =(1<<27);"      \
          "orr    r3, %[addr];"        \
          "mov    r2, %[wdata];"       \
          "str    r2, [r3];"           \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");\n\n#define IOORH(ADDR, WDATA)           \
    __asm("ldr    r3, =(1<<27);"      \
          "orr    r3, %[addr];"        \
          "mov    r2, %[wdata];"       \
          "strh   r2, [r3];"           \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");\n\n#define IOORB(ADDR, WDATA)           \
    __asm("ldr    r3, =(1<<27);"      \
          "orr    r3, %[addr];"        \
          "mov    r2, %[wdata];"       \
          "strb   r2, [r3];"           \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");\n\n#define IOXORW(ADDR, WDATA)          \
    __asm("ldr    r3, =(3<<26);"      \
          "orr    r3, %[addr];"        \
          "mov    r2, %[wdata];"       \
          "str    r2, [r3];"           \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");\n\n#define IOXORH(ADDR, WDATA)          \
    __asm("ldr    r3, =(3<<26);"
```

```
"orr      r3, %[addr];"      \
"mov      r2, %[wdata];"      \
"strh    r2, [r3];"          \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");\n\n#define IOXORB(ADDR,WDATA)           \
__asm("ldr      r3, =(3<<26);"      \
"orr      r3, %[addr];"              \
"mov      r2, %[wdata];"            \
"strb    r2, [r3];"                \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
```

# 第 18 章

## Flash 存储器模块(FTMRE)

### 18.1 简介

FTMRE 模块可以实现以下功能：

- 对 Flash (Flash)存储器编程

Flash 存储器非常适合单电源应用，允许字段重编程操作，无需任何外部高压电源即可进行编程或擦除操作。Flash 模块包含一个存储器控制器，用于执行命令以修改 Flash 内容。与存储器控制器连接的用户接口包含 Flash 通用命令对象(FCCOB)寄存器，通过填入执行命令，全局地址，数据等参数对该寄存器进行写入操作。存储器控制器必须先完成某个命令的执行，然后才能以新的命令对 FCCOB 寄存器进行写入操作。

#### 警告

在编程之前，Flash 字节或长字必须处于已擦除状态。不允许对 Flash 中已经编程（写 0）的区域重复编程（写 0）。

按字节对 Flash 存储器进行读取操作。字节的读取访问时间是一个总线周期。对于 Flash 存储器，已擦除的位读取为 1，已编程的位读取为 0。

### 18.2 特性

#### 18.2.1 Flash 存储器特性

Flash 存储器具有以下特性：

- 64KB Flash 存储器由一个 64KB 的 Flash 块组成，它分成 128 个扇区，每个扇区 512 字节
- 带有校验功能的自动化编程，擦除算法
- 快速扇区擦除和长字编程操作
- 灵活的保护方案以防止意外的编程和擦除操作

## 18.2.2 Flash 模块其他特性

Flash 存储器模块的其他特性如下：

- 无需外部高电压电源即可进行 Flash 存储器的编程和擦除操作
- 完成 Flash 命令
- 利用加密机制防止未经授权访问 Flash 存储器

## 18.3 功能说明

### 18.3.1 工作模式

Flash 存储器模块提供正常的用户工作模式。工作模式取决于模块级的输入，同时会影响、FCNFG 和 FCLKDIV 寄存器。

#### 18.3.1.1 Wait 模式

如果 MCU 进入 Wait 模式，闪存 模块不受影响。Flash 模块可通过 CCIF 中断从 Wait 模式中恢复 MCU。参见 [Flash 中断](#)。

#### 18.3.1.2 Stop 模式

如果在 MCU 请求 Stop 模式时 Flash 命令正在执行 (FSTAT[CCIF] = 0)，则当前 NVM 操作完成之后，MCU 才被允许进入 Stop 模式。

### 18.3.2 Flash 存储器映射

Flash 存储器在 MCU 地址空间的布置如下表所示。

表 18-1. Flash 存储器地址

全局地址	Flash 大小	说明
0x0000_0000–0x0000_FFFF	64 KB	Flash 块包含 Flash 配置字段。
0x0000_0000–0x0001_FFFF	128 KB	Flash 块包含 Flash 配置字段。

### 18.3.3 系统复位之后的 Flash 初始化

每次系统复位时，Flash 模块都会执行一个初始化序列，为 Flash 区块配置参数、FPROT 保护寄存器以及 FOPT 和 FSEC 寄存器建立初始值。如果在执行复位过程中遇到错误，初始化过程会将上述寄存器恢复为内置的默认值，使模块处于受到充分保护的加密状态。如果在复位序列期间检测到错误，两个 FSTAT[MGSTAT]位都将置位。

FSTAT[CCIF]在整个初始化序列期间都会清零。NVM 模块在初始化过程中的一段时间内不会对 CPU 发来的任何请求作出响应。去掉这种阻截之后，将允许读取 Flash。FSTAT[CCIF]设置为高电平标志着初始化序列完成，此时即可使能用户命令。FSTAT[CCIF]保持清零状态时，无法对寄存器 FCCOBIX 或 FCCOB 进行写入操作。

如果在任何 Flash 命令处理过程中发生复位，将立即中止该命令。对于正在编程的字或正在擦除的扇区/区块，无法保证其状态。

### 18.3.4 Flash 命令操作

Flash 命令操作用于修改 Flash 内存内容。

Flash 命令的操作包含以下三个步骤：

1. 为 Flash 编程和擦除命令配置时钟。
2. 使用命令写入序列以设置 Flash 命令参数并启动命令执行。
3. 根据 MCU 功能模式和 MCU 加密状态执行合法的 Flash 命令。

下图显示了 Flash 命令写入序列的通用流程图。

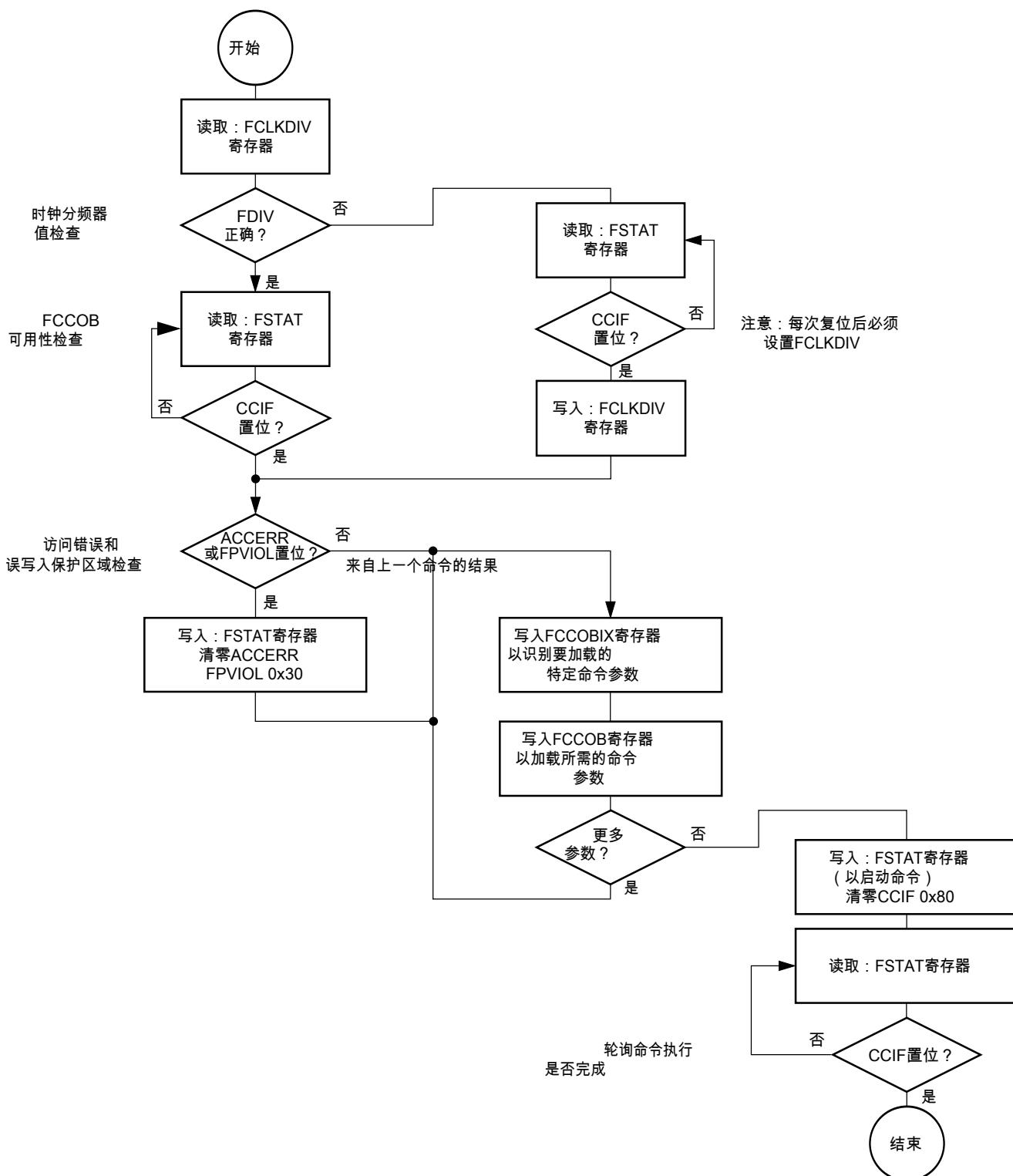


图 18-1. 通用 Flash 命令写入序列流程图

### 18.3.4.1 写 FCLKDIV 寄存器

复位后，在发出任何 Flash 程序或擦除命令之前，用户需对 FCLKDIV 寄存器进行写操作，以将 BUSCLK 分频为 1 MHz 的目标 FCLK。下表展示了基于 BUSCLK 频率建议的 FCLKDIV[FDIV]值。

**表 18-2. 各 BUSCLK 频率对应的 FDIV 值**

BUSCLK 频率 (MHz)		FDIV[5:0]
最小值 <sup>1</sup>	最大值 <sup>2</sup>	
1.0	1.6	0x00
1.6	2.6	0x01
2.6	3.6	0x02
3.6	4.6	0x03
4.6	5.6	0x04
5.6	6.6	0x05
6.6	7.6	0x06
7.6	8.6	0x07
8.6	9.6	0x08
9.6	10.6	0x09
10.6	11.6	0xA
11.6	12.6	0xB
12.6	13.6	0xC
13.6	14.6	0xD
14.6	15.6	0xE
15.6	16.6	0xF
16.6	17.6	0x10
17.6	18.6	0x11
18.6	19.6	0x12
19.6	20.6	0x13
20.6	21.6	0x14
21.6	22.6	0x15
22.6	23.6	0x16
23.6	24.6	0x17
24.6	25.6	0x18

1. BUSCLK 大于此值。

2. BUSCLK 小于或等于此值。

## 警告

如果总线时钟的运行频率低于 0.8 MHz，则无法执行编程或擦除 Flash 存储器的操作。将 FCLKDIV[FDIV]置位得过高会造成电气负载过大，从而破坏 Flash 存储器。将 FCLKDIV[FDIV]置位得太低会导致 Flash 存储器单元的编程或擦除不完整。

对 FCLKDIV 寄存器采取写入操作后，FCLKDIV[FDIVLD]将自动置位。如果 FCLKDIV[FDIVLD]为 0，则说明自上次复位后未对 FCLKDIV 寄存器采取写入操作。如果尚未对 FCLKDIV 寄存器进行写操作，则写入序列期间加载的任何 Flash 编程或擦除命令都不会执行且 FSTAT[ACCERR]将置位。

### 18.3.4.2 命令写入序列

存储器控制器会启动所有通过命令写入序列输入的合法的 Flash 命令。

启动某个命令前，必须清零 FSTAT[ACCERR]和 FSTAT[FPVIOL]，而且将会测试 FSTAT[CCIF]标志，以便确定当前命令写入序列的状态。如果 FSTAT[CCIF]为 0，则表明上一个命令写入序列仍然有效，不能开始新的命令写入序列，将忽略对 FCCOB 寄存器采取的所有写入操作。

被执行 Flash 命令的所有参数都必须加载到 FCCOB 参数字段。对 FCCOB 参数字段的访问是通过 FCCOBIX[CCOBIX]控制的。

Flash 命令模式使用编入索引的 FCCOB 寄存器向存储器控制器提供命令代码及其相关参数。首先，用户必须设置所有必需的 FCCOB 字段。然后，用户才能通过将 1 写入 FSTAT[CCIF]来启动命令的执行。此操作可将 CCIF 命令完成标志清 0。当用户清零 FSTAT[CCIF]时，所有 FCCOB 参数字段均锁定，在命令完成（存储器控制器将 FSTAT[CCIF]恢复为 1）之前不能由用户更改。有些命令向 FCCOB 寄存器阵列中返回信息。

下表展示了 Flash 命令模式中 FCCOB 参数字段的一般格式。FSTAT[CCIF]标志由存储器控制器恢复为 1 之后，即可读取返回值。将忽略对未实施的参数字段采取的写入操作 (FCCOBIX[CCOBIX] = 110b 和 FCCOBIX[CCOBIX] = 111b)，从这些字段读取到的值会返回为 0x0000。

表 18-3 展示了一般 Flash 命令格式。CCOB 阵列中第一个字的高字节包含命令代码，随后是此特定 Flash 命令的参数。有关每个命令所需的 FCCOB 设置详情，请参见 [Flash 命令汇总](#) 中的 Flash 命令说明。

表 18-3. FCCOB – Flash 命令模式的典型用法

CCOBIX[2:0]	字节	Flash 命令模式中的 FCCOB 参数字段
000	HI	FCMD[7:0]定义 Flash 命令
	LO	全局地址[23:16]
001	HI	全局地址[15:8]
	LO	全局地址[7:0]
010	HI	数据 0 [15:8]
	LO	数据 0 [7:0]
011	HI	数据 1 [15:8]
	LO	数据 1 [7:0]
100	HI	数据 2 [15:8]
	LO	数据 2 [7:0]
101	HI	数据 3 [15:8]
	LO	数据 3 [7:0]

用户通过写入 1 清零 FSTAT[CCIF]命令完成标志时，FCCOB 参数字段的内容将传输到存储器控制器中。在 Flash 命令完成之前，CCIF 标志一直保持清零状态。完成后，存储器控制器将 FSTAT[CCIF]恢复为 1，FCCOB 寄存器则用于传达结果。

下表展示了合法的 Flash 命令，这些功能（命令）可能受到 MCU 工作模式和加密状态的限制。

MCU 加密状态由 FSEC[SEC]选择。

表 18-4. 按模式和加密状态执行的 Flash 命令

FCMD	命令	解密	加密
		U <sup>1</sup>	U <sup>2</sup>
0x01	擦除检验所有数据块	*	*
0x02	擦除检验数据块	*	*
0x03	擦除检验 Flash 段	*	*
0x04	读取一次	*	*
0x06	编程 Flash	*	*
0x07	编程一次	*	*
0x08	擦除所有数据块	*	*
0x09	擦除 Flash 数据块	*	*
0x0A	擦除 Flash 扇区	*	*
0x0B	解密的 Flash	*	*
0x0C	检验后门访问密钥	*	*
0x0D	设置用户裕量水平	*	*
0x0E	设置出厂裕量水平	*	*
0x0F	配置 NVM	*	*

#### 1. 解密的用户模式

### 18.3.5 Flash 中断

当 Flash 命令操作完成时或，Flash 模块可产生中断。

表 18-5. Flash 中断源

中断源	中断标志	本地使能	全局(CCR)掩码
Flash 命令完成	CCIF (FSTAT 寄存器)	CCIE (FCNFG 寄存器)	1 位

#### 18.3.5.1 Flash 中断操作的说明

Flash 模块结合使用 FSTAT[CCIF]标志与 FCNFG[CCIE]中断使能位，以生成 Flash 命令中断请求。

下图展示了生成 Flash 模块中断所用的逻辑。

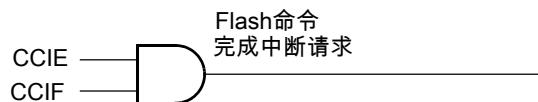


图 18-2. Flash 模块中断的实现

### 18.3.6 保护

可通过设置 FPROT 寄存器来保护 Flash 存储器中的某些区域，防止意外的编程或擦除操作。有三个单独的存储器区域：一个从 Flash 存储器中的全局地址 0x0000 向上增长，称为低位区域；一个从 Flash 存储器中的全局地址 0x7FFF 向下增长，称为高位区域；最后一个是 Flash 存储器中的其余地址，可激活进行保护。这些可受保护的区域所覆盖的 Flash 存储器地址显示在 Flash 存储器映射中。

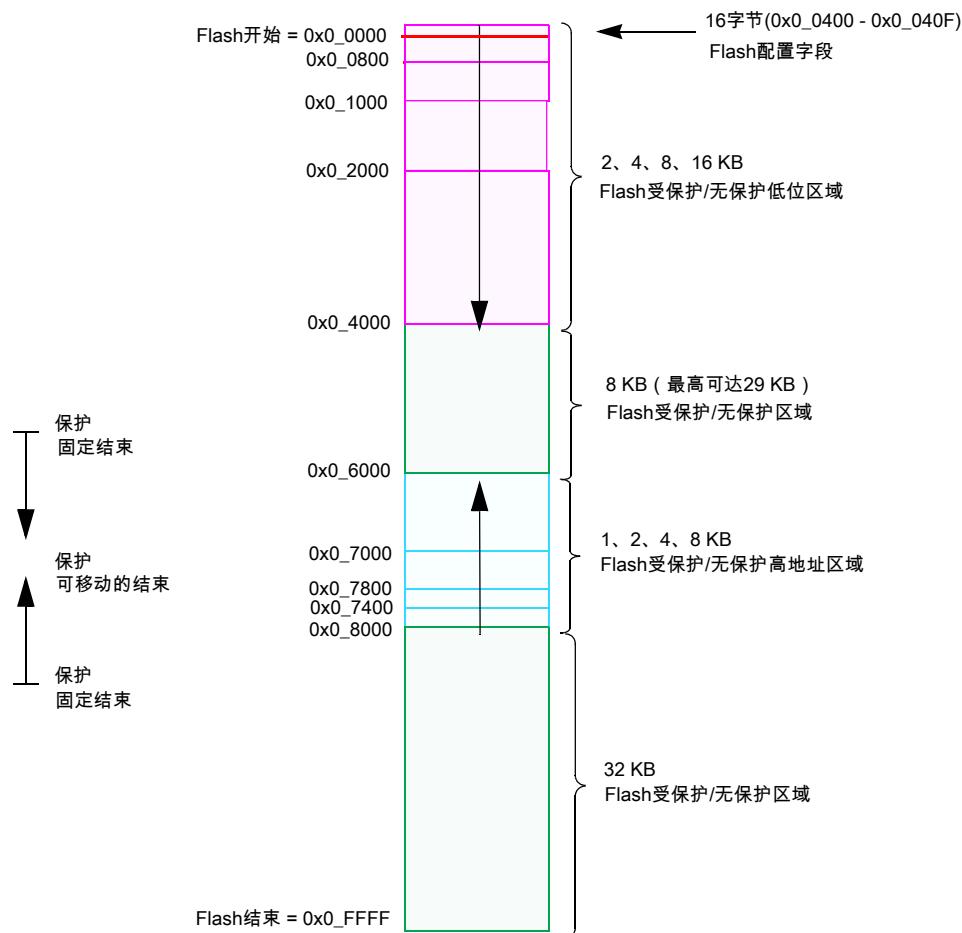


图 18-3. 64 KB 的 Flash 保护存储器映射

如下表所述, Flash 配置字段中存储了默认保护设置和加密信息, 这些设置和信息允许 MCU 限制对 Flash 模块的访问。

表 18-6. Flash 配置字段

全局地址	大小 (字节)	说明
0x0400–0x0407	8	后门比较密钥。参见“检验后门访问密钥”命令 和 使用后门密钥访问解密 MCU。
0x0408–0x040B 1	4	保留
0x040C–0x040F <sup>1</sup>	1	Flash 非易失字节 - 数据[31:24]
	1	Flash 加密字节 - 数据[23:16]
	1	Flash 保护字节 - 数据[15:8]
	1	保留位 - 数据[7:0]

1. 0x0\_0408–0x040B 和 0x040C–0x0\_040F 形成每个地址范围内的 Flash 长字, 必须以一个命令写入序列进行编程。这些标记为保留的长字中的每个字节都必须编程为 0xFF。或者, 也可以通过一个命令写入序列对 Flash 双字 0x0408–0x040F 进行编程。

Flash 存储器模块为 MCU 提供保护。在复位序列期间，Flash 存储器中全局地址 0x040D 处的 Flash 配置字段中的 Flash 保护字节将被载入 FPROT 寄存器。保护功能取决于 FPROT 寄存器中的位设置配置。

表 18-7. Flash 保护功能

FPOOPEN	FPHDIS	FPLDIS	功能 1
1	1	1	无 Flash 保护
1	1	0	受保护的低位范围
1	0	1	受保护的高位范围
1	0	0	受保护的高位和低位范围
0	1	1	整个 Flash 存储器受保护
0	1	0	无保护的低位范围
0	0	1	无保护的高位范围
0	0	0	无保护的高位和低位范围

- 有关范围大小，请参见 [Table 4](#) 和 [Table 5](#)

有些应用需要在单芯片模式下进行重编程操作，同时在无需重编程的 Flash 区域提供尽可能多的保护，此时即可使用 Flash 保护方案。

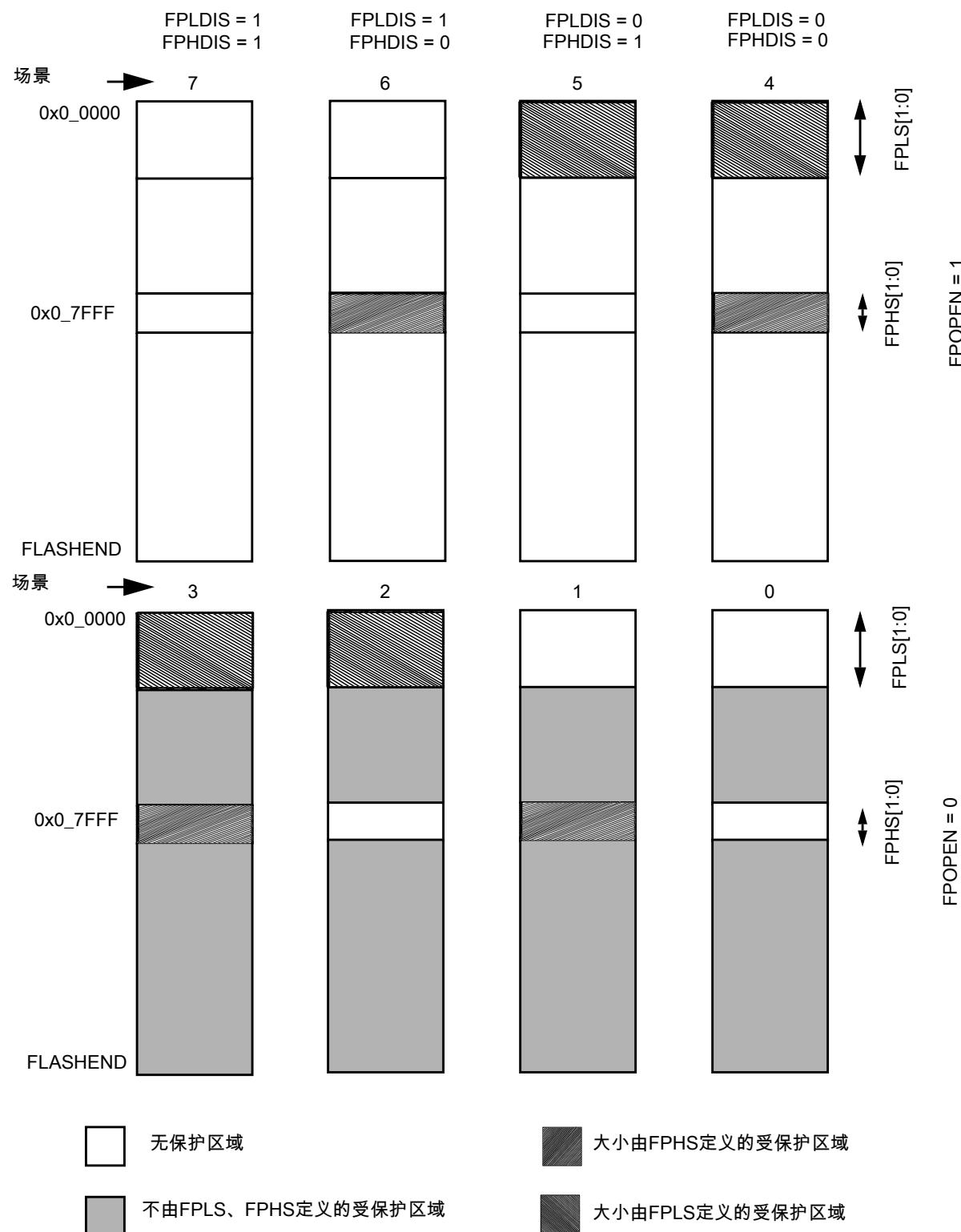


图 18-4. Flash 保护场景

一般准则是只能增加而不能移除 Flash 保护。下表指定了 Flash 保护场景之间所有的有效转换。将忽略任何尝试对 FPROT 寄存器写入无效场景的操作都将被忽略。FPROT 寄存器的内容反映了有效的保护场景。有关其他限制，请参见 FPROT[FPHS]和 FPROT[FPLS]字段说明。

表 18-8. Flash 保护场景的转换

从保护场景	到保护场景 <sup>1</sup>							
	0	1	2	3	4	5	6	7
0	×	×	×	×				
1		×		×				
2			×	×				
3				×				
4				×	×			
5			×	×	×	×		
6		×		×	×		×	
7	×	×	×	×	×	×	×	×

1. 允许的转换标记为 X。

下面两个表列出了与上表中的场景相关的 Flash 保护地址范围。

表 18-9. Flash 保护较高地址范围

FPHS[1:0]	全局地址范围	受保护的大小
00	0x7C00–0x7FFF	1 KB
01	0x7800–0x7FFF	2 KB
10	0x7000–0x7FFF	4 KB
11	0x6000–0x7FFF	8 KB

表 18-10. Flash 保护较低地址范围

FPLS[1:0]	全局地址范围	受保护的大小
00	0x0000–0x07FF	2 KB
01	0x0000–0x0FFF	4 KB
10	0x0000–0x1FFF	8 KB
11	0x0000–0x3FFF	16 KB

图 18-4 中展示了所有可能的 Flash 保护场景。尽管保护方案是在复位序列期间从 Flash 存储器的全局地址 0x040D 加载的，但用户仍然可对其进行更改。

### 18.3.7 加密

Flash 模块向 MCU 提供安全信息。Flash 安全状态由 FSEC[SEC]定义。在复位期间，Flash 模块使用从 Flash 配置字段的安全字节读取的数据初始化 FSEC 寄存器。假设在 MCU 启动时的模式中需要的 Flash 擦除和编程命令都可用且 Flash 的高地址区域未受到保护，可通过对安全字节进行编程来对复位后的安全状态进行永久性的更改。如果成功地对 Flash 安全字节进行了编程，其新值将在下一次 MCU 复位后生效。

以下小节介绍了这些与安全相关的主题：

- 使用后门秘钥来解密 MCU
- 使用 SWD 来解密 MCU
- MCU 模式和加密状态对 Flash 命令的影响

#### 18.3.7.1 使用后门密钥访问解密 MCU

利用后门密钥访问特性可以解密 MCU，该特性要求了解后门密钥的内容，也就是在地址 0x400–0x407 编程的四个 16 位字。如果 KEYEN[1:0]位处于使能状态，“检验后门访问密钥”命令（参见“[检验后门访问密钥”命令](#)）将允许用户呈现四个可能的密钥，以便通过存储器控制器与 Flash 存储器中存储的密钥进行比较。如果“检验后门访问密钥”命令中呈现的密钥与 Flash 存储器中存储的密钥匹配，则 FSEC[SEC]会被改变从而解密 MCU。不允许将 0x0000 和 0xFFFF 的密钥值作为后门密钥。“检验后门访问密钥”命令处于有效状态时，无法对 Flash 存储器进行读取访问，尝试这次操作将返回无效数据。

Flash 存储器中存储的用户代码必须有方法从外部刺激接收后门密钥。通常情况下，可以使用串口接收外部密钥输入。

如果 KEYEN[1:0]位处于使能状态，则可通过下述后门密钥访问序列解密 MCU：

1. 按“[检验后门访问密钥”命令](#) 中的说明遵循“检验后门访问密钥”命令的命令序列。
2. 如果“检验后门访问密钥”命令成功，将会解密 MCU，并且 FSEC[SEC]会强制为解密的状态 10。

“检验后门访问密钥”命令由存储器控制器监控，非法密钥会导致禁止继续使用“检验后门访问密钥”命令。要重新使能“检验后门访问密钥”命令，唯一的方法就是复位 MCU。使用“检验后门访问密钥”序列不会更改 Flash 安全字节中定义的安全状态。

地址 0x400–0x407 中存储的后门密钥不受“检验后门访问密钥”命令序列影响。“检验后门访问密钥”命令序列不影响 Flash 保护寄存器 FPROT 中定义的编程和擦除保护。

正确匹配后门密钥之后，将解密 MCU。解密 MCU 之后，可以擦除包含 Flash 安全字节的扇区，Flash 安全字节可以重新编程为解密的状态（如果需要）。在解密的状态下，用户可通过对 Flash 配置字段中的地址 0x400–0x407 进行编程来充分控制后门密钥的内容。

### 18.3.7.2 使用 SWD 解密 MCU

可用以下方式使安全状态下的 MCU 处于不安全状态，以擦除闪存内存：

1. 通过将 RESET 引脚或 DAP\_CTRL[3] 的电平变为有效值使器件复位。
2. 使 DAP\_CTRL[0] 位置位，以便通过 SWD 激活调试整体擦除。
3. 通过 SWD 使 RESET 引脚或 DAP\_CTRL[3] 位的电平变为无效值可解除复位。
4. 一直等到 DAP\_CTRL[0] 位清零（整体擦除完成后，DAP\_CTRL[0] 位将自动清零）。此时，CPU 将处于保持状态，整体擦除完成，该器件处于解密状态（Flash 配置字段中的 Flash 安全字节重编程为 0xFE）。
5. 复位该器件。

### 18.3.7.3 模式与加密状态对 Flash 命令可用性的影响

MCU 的工作模式和加密状态会对 Flash 模块的命令造成影响，如表 18-4 所示。

## 18.3.8 Flash 命令

### 18.3.8.1 Flash 命令

下表总结了有效的 Flash 命令以及这些命令对 Flash 区块及 Flash 模块内其他资源的影响。

表 18-11. Flash 命令

FCMD	命令	Flash 存储器上的功能
0x01	擦除检验所有区块	检验所有 Flash 区块是否都已擦除
0x02	擦除检验数据块	检验某个 Flash 数据块是否已被擦除
0x03	擦除检验 Flash 段	检验从所提供的地址开始的给定数量的字是否已被擦除

下一页继续介绍此表...

表 18-11. Flash 命令 (继续)

FCMD	命令	Flash 存储器上的功能
0x04	读取一次	读取 Flash 数据块内非易失性信息寄存器中的某个专用 64 字节字段，该字段之前已通过“编程一次”命令进行了编程
0x06	编程 Flash	在 Flash 区块内对最多两个长字进行编程
0x07	编程一次	对 Flash 数据块内非易失性信息寄存器中的某个专用 64 字节字段进行编程，该字段只允许编程一次
0x08	擦除所有区块	擦除所有 Flash 区块 只有在启动命令之前已置位 FPROT[FPHDIS]、FPROT[FPLDIS]及 FPROT[FPOEN]位的情况下，才可能擦除所有 Flash 区块
0x09	擦除 Flash 区块	擦除某个 Flash 区块 只有在启动命令之前已置位 FPROT[FPLDIS]、FPROT[FPHDIS]及 FPROT[FPOEN]的情况下，才可能擦除整个 Flash 区块。
0x0A	擦除 Flash 扇区	擦除某个 Flash 扇区中的所有字节
0x0B	解密的 Flash	支持通过擦除所有 Flash 区块并检验是否所有 Flash 区块都已擦除来退出 MCU 加密状态
0x0C	检验后门访问密钥	支持通过检验一组安全密钥来退出 MCU 加密状态
0x0D	设置用户裕量水平	为所有 Flash 区块指定用户裕量读取水平
0x0E	设置出厂裕量水平	为所有 Flash 区块指定出厂裕量读取水平
0x0F	配置 NVM	配置 NVM 参数，以便使能或禁用 NVM 阵列中的某些特性，从而在某些特定场合达到省电的目的。

### 18.3.9 Flash 命令汇总

本节详细介绍由命令写入序列启动的所有可用 Flash 命令。如果执行了以下任一非法步骤，导致存储器控制器不处理命令，FSTAT[ACCERR]将在命令写入序列期间置位：

- 启动任何对 Flash 存储器进行编程或擦除操作的命令写入序列没有对 FLCKDIV 初始化。
- 写入无效命令作为命令写入序列的一部分。
- 有关其他可能的错误，请参见针对每个命令提供的错误处理表。

在对某个 Flash 区块执行算法(FSTAT[CCIF] = 0)期间对该区块进行读取操作的情况下，。这样还会触发非法访问异常情况。

如果 FSTAT[ACCERR]或 FSTAT[FPVIOL]已置位，则用户必须在启动任何命令写入序列之前清空这些字段。

## 警告

在编程之前，Flash 长字必须处于已擦除状态。已编程的 Flash 长字不允许被再次编程。

### 18.3.9.1 “擦除验证所有数据块”命令

“擦除验证所有数据块”命令将验证所有闪存模块都已被擦除。

**表 18-12. “擦除验证所有数据块”命令的 FCCOB 要求**

CCOBIX[2:0]	FCCOBHI 参数	FCCOBLO 参数
000	0x01	不需要

清零 FSTAT[CCIF]以启动“擦除验证所有数据块”命令后，存储器控制器将验证整个 Flash 存储器空间是否都已被擦除。FSTAT[CCIF]在擦除验证所有数据块操作完成后将置位。如果未擦除所有数据块，则表示空白检查失败且 FSTAT[MGSTAT]位都将置位。

**表 18-13. “擦除验证所有数据块”命令错误处理**

寄存器	错误位	错误条件
FSTAT	ACCERR	启动命令时，如果 CCOBIX[2:0] != 000，则置位
	FPVIOL	无
	MGSTAT1	读取期间发生任何错误时置位 <sup>1</sup> 或空白检查失败时，则此位将置位
	MGSTAT0	读取过程中发生任何非可纠正性错误，或者空白检查失败时，则此位将置位

1. 参见 NVM 存储器映射

### 18.3.9.2 “擦除验证数据块”命令

“擦除验证数据块”命令使用户可验证整个 Flash 模块已被擦除。FCCOB 全局地址[23:0]位用于确定必须验证哪个数据块。

**表 18-14. “擦除验证数据块”命令的 FCCOB 要求**

CCOBIX[2:0]	FCCOBHI 参数	FCCOBLO 参数
000	0x02	用于确定 Flash 数据块的全局地址[23:16]
001	Flash 数据块中要验证的全局地址[15:0]	

清除 FSTAT[CCIF]以启动“擦除验证模块”命令后，内存控制器将验证选定的 Flash 模块已被擦除。FSTAT[CCIF]标志在擦除验证数据块操作完成后将置位。如果未擦除该数据块，则意味着空白检查失败且 FSTAT[MGSTAT]位都会置位。

表 18-15. “擦除验证数据块”命令错误处理

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] != 000 时，则此位将置位
		提供无效的全局地址[23:0]时，则此位将置位 <sup>1</sup>
	FPVIOL	无
	MGSTAT1	读取过程中发生任何错误，或空白检查失败时，则此位将置位
	MGSTAT0	读取过程中发生任何非可纠正性错误，或者空白检查失败时，则此位将置位

1. 如同在 NVM 存储器映像中发现的一样

### 18.3.9.3 “擦除检验 Flash 段”命令

“擦除检验 Flash 段”命令将检验 Flash 存储器中的某个代码段是否已擦除。“擦除检验 Flash 段”命令定义了待检验代码的起始点和长字数。

表 18-16. “擦除检验 Flash 段”命令的 FCCOB 要求

CCOBIX[2:0]	FCCOBHI 参数	FCCOBLO 参数
000	0x03	Flash 区块的全局地址[23:16]
001		要检验的第一个长字的全局地址[15:0]
010		要检验的长字数

清零 FSTAT[CCIF]以启动“擦除检验 Flash 段”命令时，存储器控制器将检验 Flash 存储器的所选段是否已擦除。“擦除检验 Flash 段”操作完成后，将置位 FSTAT[CCIF]标志。如果该段未擦除，也就意味着空白检查失败，两个 FSTAT[MGSTAT]位都将置位。

表 18-17. “擦除检验 Flash 段”命令错误处理

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] != 010 时置位
		设置命令在当前模式下是否可用（参见表 18-4）
		如果提供了无效的全局地址 [23:0]，则设置此参数（请参见 表 18-1） <sup>1</sup>
		提供未对齐的长字地址时置位（全局地址[1:0] != 00）
		请求的段越过 Flash 地址边界时置位
	FPVIOL	无
	MGSTAT1	读取期间发生任何错误时置位 <sup>2</sup> 或空白检查失败时
	MGSTAT0	读取过程中发生任何无法纠正的错误时 <sup>2</sup> 或空白检查失败时置位

1. 和 NVM 存储器映像定义的一样
2. 参见 NVM 存储器映射

### 18.3.9.4 “读取一次”命令

“读取一次”命令对 Flash 内非易失性信息寄存器中保留的 64 字节字段（8 个双字）提供读取访问。“读取一次”字段只能编程一次，无法擦除。可用于存储产品 ID 或其他任何只能写入一次的信息。使用“[编程一次”命令](#) 中介绍的“编程一次”命令进行编程。为避免代码跑飞，不得从包含“编程一次”保留字段的 Flash 数据块执行“读取一次”命令。

**表 18-18. “读取一次”命令的 FCCOB 要求**

CCOBIX[2:0]	FCCOB 参数	
000	0x04	不需要
001	“读取一次”双字索引(0x0000 – 0x0007)	
010	“读取一次”字 0 值	
011	“读取一次”字 1 值	
100	“读取一次”字 2 值	
101	“读取一次”字 3 值	

清零 FSTAT[CCIF]以启动“读取一次”命令时，会提取一个“读取一次”双字并将其存储在 FCCOB 寄存器数组中。“读取一次”操作完成后，FSTAT[CCIF]标志将置位。“读取一次”命令的有效双字索引值范围是 0x0000 到 0x0007。执行“读取一次”命令期间，任何尝试对 Flash 数据块内的地址进行读取的操作都将返回无效数据。

**表 18-19. “读取一次”命令错误处理**

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] != 001 时置位
		设置命令在当前模式下是否可用（参见 <a href="#">表 18-4</a> ）
		提供无效的双字索引时置位
	FPVIOL	无
	MGSTAT1	读取期间发生任何错误时置位
	MGSTAT0	读取期间发生任何无法纠正的错误时置位

### 18.3.9.5 “编程 Flash”命令

“编程 Flash”操作将利用嵌入式算法对 Flash 存储器中之前已擦除的最多两个长字进行编程。

## 注

Flash 双字在被编程前必须处于已擦除状态。不允许对 Flash 双字内的位进行累计编程。

**表 18-20. “编程 Flash”命令的 FCCOB 要求**

CCOBIX[2:0]	FCCOBHI 参数	FCCOBLO 参数
000	0x06	用于确定 Flash 数据块的全局地址[23:16]
001		要进行编程的长字位置的全局地址[15:0] <sup>1</sup>
010		字 0 (长字 0) 编程值
011		字 1 (长字 0) 编程值
100		字 2 (长字 1) 编程值
101		字 3 (长字 1) 编程值

- 全局地址[1:0]必须为 00。

清零 FSTAT[CCIF]以启动“编程 Flash”命令时,存储器控制器将数据字编程到提供的全局地址,接下来检验数据字是否按预期方式读回。FSTAT[CCIF]标志将在“编程 Flash”操作完成后置位。

**表 18-21. “编程 Flash”命令错误处理**

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] ≠ 011 或 101 时置位
		设置命令在当前模式下是否可用 (参见表 18-4 )
		提供无效的全局地址[23:0]时置位 (参见表 18-1 。 <sup>1</sup> )
		提供未对齐的长字地址 (全局地址[1:0] != 00) 时置位
		请求的字组超过 Flash 数据块末尾时置位。
	FPVIOL	全局地址[23:0]指向某个受保护数据时置位
	MGSTAT1	检验操作期间发生任何错误时置位。
	MGSTAT0	检验操作期间发生任何无法纠正的错误时置位

- 和 NVM 存储器映像定义的一样。

### 18.3.9.6 “编程一次”命令

“编程一次”命令仅限于对 Flash 内非易失性信息寄存器中保留的 64 字节字段 (8 个双字) 进行的编程。如“[读取一次”命令](#)所述, 可使用“读取一次”命令读取“编程一次”保留字段。“编程一次”命令只能发出一次, 因为无法擦除 Flash 内的非易失性信息寄存器。为避免代码跑飞, 不得从包含“编程一次”保留字段的 Flash 区块执行“编程一次”命令。

**表 18-22. “编程一次”命令的 FCCOB 要求**

CCOBIX[2:0]	FCCOB 参数	
000	0x07	不需要
001	“编程一次”双字索引(0x000 – 0x0007)	
010	“编程一次”字 0 值	
011	“编程一次”字 1 值	
100	“编程一次”字 2 值	
101	“编程一次”字 3 值	

清除 FSTAT[CCIF]以启动“编程一次”命令时，存储器控制器首先会检验所选双字是否已被擦除。如已被擦除，则将对所选双字进行编程，然后进行读回检验。FSTAT[CCIF]标志会保持清零状态，只有在“编程一次”操作完成后才会置位。

无法擦除“编程一次”命令访问的保留非易失性信息寄存器，不允许再次对任一双字进行编程。“编程一次”命令的有效双字索引值范围是 0x0000 到 0x0007。执行“编程一次”命令期间，任何尝试对 Flash 内的地址进行读取的操作都将返回无效数据。

**表 18-23. “编程一次”命令错误处理**

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下满足 CCOBIX[2:0] != 101 时置位
		设置命令在当前模式下是否可用（参见表 18-4）
		提供无效的双字索引时置位
		目标地址已经被编程的情况下置位 <sup>1</sup>
	FPVIOL	无
	MGSTAT1	检验操作期间发生任何错误时置位。
	MGSTAT0	检验操作期间发生任何无法纠正的错误时置位

- 如果某个“编程一次”双字初次编程为 0xFFFF\_FFFF\_FFFF\_FFFF，将允许再次对该双字执行“编程一次”命令。

### 18.3.9.7 “擦除所有区块”命令

“擦除所有区块”操作将擦除整个 Flash 存储器空间。

**表 18-24. “擦除所有区块”命令的 FCCOB 要求**

CCOBIX[2:0]	FCCOBHI 参数	FCCOBLO 参数
000	0x08	不需要

清零 FSTAT[CCIF]以启动“擦除所有数据块”命令时，存储器控制器将擦除整个 NVM 存储器空间，并检验该空间是否已被擦除。如果存储器控制器检验到整个 NVM 存储器空间已被正确擦除，就会退出加密状态。因此，器件处于解密状态。执行此命令期间(FSTAT[CCIF] = 0)，用户不得对任何 NVM 模块寄存器进行写入操作。FSTAT[CCIF]标志将在“擦除所有数据块”操作完成后置位。

表 18-25. “擦除所有数据块”命令错误处理

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] ≠ 000 时置位 设置命令在当前模式下是否可用（参见表 18-4）
	FPVIOL	有任何 Flash 存储器区域处于受保护状态时置位
	MGSTAT1	检验操作期间发生任何错误时置位。 <sup>1</sup>
	MGSTAT0	检验操作期间发生任何无法纠正的错误时置位 <sup>1</sup>

1. 参见 NVM 存储器映像。

### 18.3.9.8 调试器整体擦除请求

“擦除所有数据块”命令的功能也可通过调试器整体擦除请求特性以非命令的方式实现。

调试器整体擦除请求要求在调用这一功能之前先初始化时钟分频器寄存器 FCLKDIV。如果调用该特性时不允许直接向寄存器 FCLKDIV 写入值，则请查阅参考手册以了解有关 FCLKDIV 默认值的信息。如果 FCLKDIV 未初始化，则将无法执行调试器整体擦除请求，FSTAT[ACCERR]标志将会置位。执行整体擦除功能后，将复位 FCLKDIV 寄存器，并且必须在启动任何其他命令前先初始化 FCLKDIV 寄存器的值。

调用“擦除所有”功能之前，必须清零 FSTAT[ACCERR]和 FSTAT[FPVIOL]标志。调用后，无论保护设置如何，调试器整体擦除请求都会擦除所有 Flash 存储器空间。如果擦除后检验通过，随后会将 FSEC[SEC]设置为解密状态，由此退出加密状态。“Flash 配置”字段中的安全字节将设置为解密状态。调试器整体擦除请求的状态反映在 FCNFG[ERSAREQ]中。一旦此操作完成，将清零 FCNFG[ERSAREQ]，且正常的 FSTAT 错误报告将可用，如下表中所述。

整体擦除序列结束时，保护的配置情况仍然与执行整体擦除功能之前一样。如果应用程序要求在整体擦除功能结束后对 P-Flash 进行编程，则必须考虑现有的保护限制。如果需要禁用保护，用户可能需要在完成整体擦除功能之后立即复位系统。

表 18-26. 调试器整体擦除请求错误处理

寄存器	错误位	错误条件
FSTAT	ACCERR	命令在当前模式下不可用时置位

下一页继续介绍此表...

表 18-26. 调试器整体擦除请求错误处理 (继续)

寄存器	错误位	错误条件
	MGSTAT1	擦除检验操作期间或编程检验操作期间发生任何错误时置位。
	MGSTAT0	擦除检验操作期间或编程检验操作期间发生任何无法纠正的错误时置位。

### 18.3.9.9 “擦除 Flash 区块”命令

“擦除 Flash 区块”操作将擦除某个 Flash 区块中的所有地址。

表 18-27. “擦除 Flash 区块”命令的 FCCOB 要求

CCOBIX[2:0]	FCCOB 参数	
000	0x09	全局地址[23:16], 以识别 Flash 区块
001	要擦除的 Flash 数据块中的全局地址[15:0]	

清零 FSTAT[CCIF]以启动“擦除 Flash 数据块”命令时, 存储器控制器将擦除所选的 Flash 数据块, 并检验该数据块是否已被擦除。FSTAT[CCIF]标志会在“擦除 Flash 数据块”操作完成后置位。

表 18-28. “擦除 Flash 数据块”命令错误处理

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] != 001 时置位
		设置命令在当前模式下是否可用 (参见表 18-4 )
		提供无效的全局地址[23:16]时置位 <sup>1</sup>
	FPVIOL	所选 Flash 数据块的某个区域处于受保护状态时置位
	MGSTAT1	检验操作期间发生任何错误时置位。 <sup>2</sup>
	MGSTAT0	检验操作期间发生任何无法纠正的错误时置位 <sup>2</sup>

1. 和 NVM 存储器映像定义的一样。
2. 参见 NVM 存储器映像。

### 18.3.9.10 “擦除 Flash 扇区”命令

“擦除 Flash 扇区”操作将擦除某个 Flash 扇区中的所有地址。

**表 18-29. “擦除 Flash 扇区”命令的 FCCOB 要求**

CCOBIX[2:0]	FCCOB 参数	
000	0x0A	用于确定要擦除的 Flash 数据块的全局地址 [23:16]
001	待擦除的扇区中任意位置上的全局地址 [15:0]。有关 Flash 扇区大小, 请参见 <a href="#">概述</a>	

清除 FSTAT[CCIF]以启动“擦除 Flash 扇区”命令时, 存储器控制器将擦除所选的 Flash 扇区, 并检验该扇区是否已被擦除。FSTAT[CCIF]标志在“擦除 Flash 扇区”操作完成后置位。

**表 18-30. “擦除 Flash 扇区”命令错误处理**

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] != 001 时置位
		设置命令在当前模式下是否可用 (参见 <a href="#">表 18-4</a> )
		提供无效的全局地址[23:16]时置位。 <sup>1</sup> (请参见 <a href="#">表 18-1</a> )
		提供未对齐的长字地址 (全局地址[1:0] != 00) 时置位
	FPVIOL	所选 Flash 扇区处于受保护状态时置位
	MGSTAT1	检验操作期间发生任何错误时置位。
	MGSTAT0	检验操作期间发生任何无法纠正的错误时置位

1. 参见 NVM 存储器映射

### 18.3.9.11 “解密 Flash”命令

“解密 Flash”命令会擦除整个 Flash 存储器空间, 如果擦除成功, 将退出加密状态。

**表 18-31. “解密 Flash”命令的 FCCOB 要求**

CCOBIX[2:0]	FCCOB 参数	
000	0x0B	不需要

清零 FSTAT[CCIF]以启动“解密 Flash”命令时, 存储器控制器会擦除整个 Flash 存储器空间, 并检验该空间是否已擦除。如果存储器控制器检验到整个 Flash 存储器空间都已正确擦除, 将会退出加密状态。如果擦除检验不成功, “解密 Flash”操作将置位 FSTAT[MGSTAT1]并终止但不更改加密状态。执行此命令期间(FSTAT[CCIF] = 0), 用户不得对任何 Flash 模块寄存器进行写入操作。“解密 Flash”操作完成后, FSTAT[CCIF]标志将置位。

表 18-32. “解密 Flash”命令错误处理

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] != 000 时置位 设置命令在当前模式下是否可用 (参见表 18-4)
	FPVIOL	设置是否有任何 Flash 存储器区域处于受保护状态
	MGSTAT1	检验操作期间发生任何错误时置位。 <sup>1</sup>
	MGSTAT0	检验操作期间发生任何无法纠正的错误时置位 <sup>1</sup>

1. 参见 NVM 存储器映射

### 18.3.9.12 “检验后门访问密钥”命令

只有在 FSEC[KEYEN]位使能的情况下，才能执行“检验后门访问密钥”命令。如果用户提供的密钥与 Flash 配置字段的 Flash 安全字节中存储的密钥匹配，“检验后门访问密钥”命令将使 MCU 退出加密状态。详情参见表 18-1。必须从 RAM 运行用于执行“检验后门访问密钥”命令的代码。

表 18-33. “检验后门访问密钥”命令的 FCCOB 要求

CCOBIX[2:0]	FCCOBHI 参数	FCCOBLO 参数
000	0x0C	不需要
001		密钥 0
010		密钥 1
011		密钥 2
100		密钥 3

清零 FSTAT[CCIF]以启动“检验后门访问密钥”命令时，存储器控制器将检查 FSEC[KEYEN]位，以检验该命令是否已使能。如果未使能，存储器控制器将使 FSTAT[ACCERR]位置位。如果该命令已使能，存储器控制器会将 FCCOB 中提供的密钥与 Flash 配置字段中的后门比较密钥进行对比，即密钥 0 与 0x0400 相比较，以此类推。如果后门密钥匹配，将退出加密状态。如果后门密钥不匹配，就不会退出加密状态，将来所有尝试执行“检验后门访问密钥”命令的操作都会中止（置位 FSTAT[ACCERR]），直到发生复位为止。“检验后门访问密钥”操作完成后，FSTAT[CCIF]标志将置位。

表 18-34. “检验后门访问密钥”命令错误处理

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] ≠ 100 时置位
		提供不正确的后门密钥时置位
		未使能后门密钥访问(KEYEN[1:0] ≠ 10)时置位

下一页继续介绍此表...

表 18-34. “检验后门访问密钥”命令错误处理 (继续)

寄存器	错误位	错误条件
		自上次复位以来后门密钥不匹配时置位
	FPVIOL	无
	MGSTAT1	无
	MGSTAT0	无

### 18.3.9.13 “设置用户裕量水平”命令

用户裕量是一个基于正常读取基准水平的小差值，实际上是一个最小安全裕量。换言之，如果读取操作在容差更严格的用户裕量条件下通过了，那么在用户遭遇数据丢失之前，正常的读取操作至少拥有这么多的安全裕量。

“设置用户裕量水平”命令导致存储器控制器设置为 Flash 区块将来的读取操作指定的裕量水平。

表 18-35. “设置用户裕量水平”命令的 FCCOB 要求

CCOBIX[2:0]	FCCOBHI 参数	FCCOBLO 参数
000	0x0D	全局地址[23:16]，以识别 Flash 区块
001	用于确定 Flash 数据块的全局地址[15:0]	
010	裕量水平设置	

清零 FSTAT[CCIF]以启动“设置用户裕量水平”命令时，存储器控制器将设置目标区块的用户裕量水平，然后置位 FSTAT[CCIF]标志。

#### 注

下表定义了“设置用户裕量水平”命令的有效裕量水平设置。

表 18-36. “设置用户裕量水平”的有效设置

CCOB (CCOBIX = 010)	水平说明
0x0000	返回正常水平
0x0001	用户裕量 1 级 <sup>1</sup>
0x0002	用户裕量 0 级 <sup>2</sup>

1. 读取裕量能够擦除状态
2. 读取裕量能够编程状态

表 18-37. “设置用户裕量水平”命令错误处理

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] != 010 时置位
		设置命令在当前模式下是否可用 (参见表 18-4)
		提供无效的全局地址[23:0]时置位
		提供无效的裕量水平设置时置位
	FPVIOL	无
	MGSTAT1	无
	MGSTAT0	无

**注**

用户裕量水平可用于检查 NVM 存储器内容是否拥有合适的裕量，以便进行正常水平的读取操作。如果在检查用户裕量水平的 NVM 存储器内容时出现了意外结果，则说明检测到潜在的信息丢失。

**18.3.9.14 “设置出厂裕量水平”命令**

“设置出厂裕量水平”命令导致存储器控制器设置为 Flash 区块将来的读取操作指定的裕量水平。

表 18-38. “设置出厂裕量水平”命令的 FCCOB 要求

CCOBIX[2:0]	FCCOBHI 参数	FCCOBLO 参数
000	0x0E	全局地址[23:16], 以识别 Flash 区块
001	用于确定 Flash 数据块的全局地址[15:0]	
010	裕量水平设置	

清零 FSTAT[CCIF]以启动“设置出厂裕量水平”命令时，存储器控制器将设置目标区块的出厂裕量水平，然后置位 FSTAT[CCIF]标志。

**注**

下表定义了“设置出厂裕量水平”命令的有效裕量水平设置。

表 18-39. “设置出厂裕量水平”的有效设置

CCOB (CCOBIX = 010)	水平说明
0x0000	返回正常水平

下一页继续介绍此表...

表 18-39. “设置出厂裕量水平”的有效设置 (继续)

CCOB (CCOBIX = 010)	水平说明
0x0001	用户裕量 1 级 <sup>1</sup>
0x0002	用户裕量 0 级 <sup>2</sup>
0x0003	出厂裕量 1 级 <sup>1</sup>
0x0004	出厂裕量 0 级 <sup>2</sup>

1. 读取裕量能够擦除状态
2. 读取裕量能够编程状态

表 18-40. “设置出厂裕量水平”命令的错误处理

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] != 010 时置位
		设置命令在当前模式下是否可用 (参见表 18-4)
		提供无效的全局地址[23:0]时置位
		提供无效的裕量水平设置时置位
	FPVIOL	无
	MGSTAT1	无
	MGSTAT0	无

## 警告

只有在检验初始出厂编程的过程中，才能使用出厂裕量水平。

## 注

出厂裕量水平可用于检查 Flash 存储器内容是否拥有合适的裕量，以便在正常水平的设置条件下保留数据。如果在检查出厂裕量水平的 Flash 存储器内容时出现了意外结果，则必须擦除 Flash 存储器内容并重新编程。

### 18.3.9.15 “配置 NVM”命令

“配置 NVM”命令让用户能够控制 NVM 阵列中的某些特性，可以在某些特定场合通过使能或禁用这些特性达到省电的目的。通过“配置 NVM”命令修改的设置不是永久的，因此如果在用户应用中需要更改这些设置，必须在每次复位后调用此命令。

表 18-41. “配置 NVM”命令的 FCCOB 要求

CCOBIX[2:0]	FCCOBHI 参数	FCCOBLO 参数
000	0x0F	下表所示的“配置 NVM”控制字节

表 18-42. NVM 控制字节 - 字段说明

字段	说明
7 QUERY	查询/驱动模式 - 此位控制“配置 NVM”命令是否用于驱动或查询 FLPHV 和 FLPLF 位的当前状态。 0 表示将 FLPHV 和 FLPLF 位上的选项驱动到 NVM 阵列中 1 表示查询 FLPHV 和 FLPLF 的当前状态。执行命令后，可以从 FCCOB[0]读到 FLPHV 和 FLPLV 的当前状态。
6-2 RNV	保留位 - RNV 位应该是 1'b0，留作今后改进。
1 FLPHV	<b>Flash</b> 高供电电压低功耗控制 - FLPHV 位控制 NVM 阵列中的某个电路，该电路可生成内部基准电压源，当器件在外部供电电压规范的较低范围工作时该基准电压用于 Flash 读取操作。在特定环境下，可禁用该电路，以便省电。 0 表示使能该电路（推荐，默认值）- 这样可确保 NVM 阵列能在全范围外部供电电压内正常进行读取操作。 1 表示禁用该电路 - 只有在保证外部供电电压产生的驱动电压是在最低水平之上才设置。
0 FLPLF	<b>Flash</b> 低频率低功耗控制 - FLPLF 位控制 NVM 阵列中的某个电路，如果器件正在低频率条件下运行，该电路可在 Flash 读取操作期间限制电流消耗。该电路本身也会耗一些功率，建议在典型频率设置条件下禁用该电路。在特定环境下（总线频率较低时），可使能该电路以便省电。 0 表示禁用该电路（推荐，默认值 <sup>1</sup> ）- 对于典型总线频率值，可保留该电路的禁用状态，以便最大限度减少能耗。 1 表示使能该电路 - 在总线频率设置为低于最低阈值时可采用此设置，这样 Flash 即可在低频率工作状态下省电。

- 如果器件特别用于低频率工作范围，Freescale 可能会为 NVM 阵列中的这种控制提供具有不同初始设置的器件，最终使用不同型号器件。因此，如果用户不确定这种控制的初始设置，建议先在查询模式下运行“配置 NVM”命令，以便检索 FLPLF 的初始状态。

LPHV 控制着一个内部电路，该电路可保证在外部电源的整个范围内实现正确的 Flash 读取操作，特别是在较低水平的电压规格下。如果外部电源驱动的电压高于 NVM 电气参数中指定的最低电平，则可使用 FLPHV 禁用该电路 - 如果可禁用该电路，请参见参考手册中的相应章节确认此最小电平和省电情况。

### 警告

如果用户错误地禁用了 FLPHV 控制的电路，导致工作电压降至指定的最小供电电压以下，就会有一定的风险。这种情况下，Flash 不会读取到有意义的数据，从而导致不可预知的结果。恢复 Flash 功能的唯一方法可能是将器件复位，这样 NVM 阵列就会恢复到默认状况（也就是使能 FLPHV 控制的电路）。

FLPLF 控制着一个内部电路，如果 Flash 在较低的总线频率(BUSCLK)条件下运行，该电路可限制 Flash 读取期间的电流消耗。对于典型频率值，建议禁用该电路，以便省电；但是对于在低频率条件下运行的应用，则可使能该电路，以便最大限度地减少电流消耗。了解总线频率阈值的详情，在此阈值条件下使能该电路可以省电。

**注**

FLPLF 位与读取操作相关，此操作受总线频率(BUSCLK)影响；编程和擦除操作则受与 FLPLF 无关的 FCLK 时基影响。

清空 CCIF 以启动“配置 NVM”命令时，MGATE 会相应地设置 NVM 阵列参数（条件是在驱动模式下启动命令 - QUERY=0），然后设置 CCIF 标志。执行命令后，始终可在 FCCOB[0]中读取 FLPHV 和 FLPLV 的当前状态。可在查询模式下 (QUERY=1) 执行“配置 NVM”命令，以便直接检索这些位的状态，而无需将其驱动到 NVM 阵列中。

用户无需运行“配置 NVM”命令即可正确操作 Flash 阵列。复位之后保留默认值就是适合 Flash 操作的建议/安全状态。如上文所述，“配置 NVM”命令只适合在能够省电的情况下调用。

**表 18-43. “配置 NVM”命令错误处理**

寄存器	错误位	错误条件
FSTAT	ACCERR	命令启动情况下 CCOBIX[2:0] != 000 时置位
		设置命令在当前模式下是否不可用（参见表 18-4）
		提供无效的全局地址[23:0]时置位
		提供无效的裕量水平设置时置位
	FPVIOL	无
	MGSTAT1	无
	MGSTAT0	无

## 18.4 存储器映像和寄存器定义

本节从较高层面概要介绍了寄存器以及这些寄存器的映射方式。寄存器能够以 32 位、16 位（对齐到[31:16]数据或[15:0]数据）或 8 位存取。如果是可写的寄存器，在 Flash 命令执行期间将会禁止写入访问。有关更多详情，请参见 [Flash 存储器映射](#) 中的警告说明。

### FTMRE 存储器映射

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4002_0001	Flash CCOB 索引寄存器 (FTMRE_FCCOBIX)	8	R/W	00h	<a href="#">18.4.1/260</a>
4002_0002	Flash 安全寄存器 (FTMRE_FSEC)	8	R	未定义	<a href="#">18.4.2/260</a>
4002_0003	Flash 时钟分频器寄存器 (FTMRE_FCLKDIV)	8	R/W	00h	<a href="#">18.4.3/261</a>
4002_0005	Flash 状态寄存器 (FTMRE_FSTAT)	8	R/W	80h	<a href="#">18.4.4/262</a>
4002_0007	Flash 配置寄存器 (FTMRE_FCNFG)	8	R/W	00h	<a href="#">18.4.5/263</a>
4002_0008	Flash 通用命令对象寄存器：低位 (FTMRE_FCCOBLO)	8	R/W	00h	<a href="#">18.4.6/264</a>

下一页继续介绍此表...

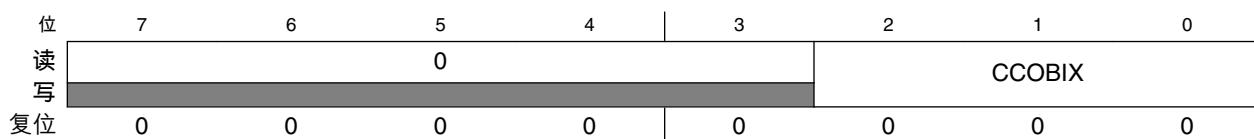
## FTMRE 存储器映射 (继续)

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4002_0009	Flash 通用命令对象寄存器 : 高位 (FTMRE_FCCOBHI)	8	R/W	00h	<a href="#">18.4.7/264</a>
4002_000B	Flash 保护寄存器 (FTMRE_FPROT)	8	R	参见章节	<a href="#">18.4.8/265</a>
4002_000F	Flash 选项寄存器 (FTMRE_FOPT)	8	R	未定义	<a href="#">18.4.9/266</a>

### 18.4.1 Flash CCOB 索引寄存器 (FTMRE\_FCCOBIX)

FCCOBIX 寄存器用于对 FCCOB 寄存器编制索引，以便进行 NVM 存储器操作。

地址: 4002\_0000h 基准 + 1h 偏移 = 4002\_0001h



**FTMRE\_FCCOBIX 字段描述**

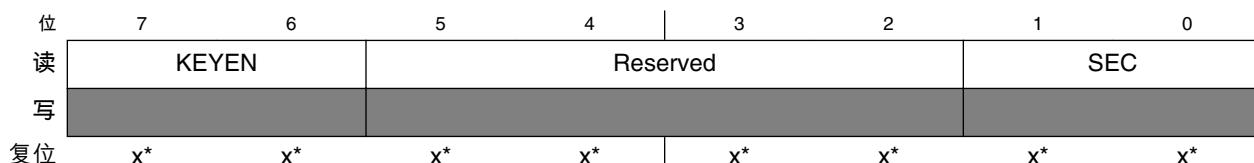
字段	描述
7-3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
CCOBIX	通用命令寄存器索引  对 FCCOB 寄存器组正在读写操作的字进行选择。

### 18.4.2 Flash 安全寄存器 (FTMRE\_FSEC)

FSEC 寄存器控制着与 MCU 及 NVM 模块加密相关的所有位。FSEC 寄存器中的所有字段均可读，但不可写。在复位序列期间，会将 Flash 存储器中 Flash 配置字段内 Flash 安全字节的内容加载到 FSEC 寄存器。

有关加密功能，请参见[加密](#)。

地址: 4002\_0000h 基准 + 2h 偏移 = 4002\_0002h



\* 注:

- x = 复位时未定义。

**FTMRE\_FSEC 字段描述**

字段	描述
7–6 KEYEN	<p>后门密钥安全使能位</p> <p>KEYEN[1:0]位定义 Flash 模块后门密钥访问权限的使能。</p> <p>注：01 是用于禁用后门密钥访问权限的 KETEN 的首选状态。</p> <p>00 禁用 01 禁用 10 使能 11 禁用</p>
5–2 Reserved	此字段为保留字段。
SEC	<p>Flash 安全位</p> <p>定义 MCU 的加密状态。如果 Flash 模块未利用后门密钥访问权限进行加密保护，则 SEC 字段强制为 10。</p> <p>注：00 是用于将 MCU 设置到加密状态的 SEC 的首选状态。</p> <p>00 加密 01 加密 10 解密 11 加密</p>

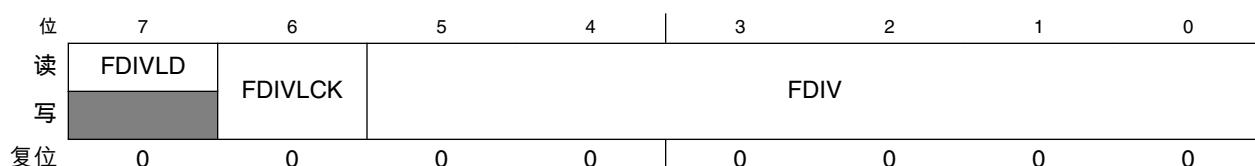
**18.4.3 Flash 时钟分频器寄存器 (FTMRE\_FCLKDIV)**

FCLKDIV 寄存器用于控制编程和擦除算法中的定时事件。

**注**

执行 Flash 命令(FSTAT[CCIF] = 0)时不得对 FCLKDIV 寄存器采取写入操作

地址: 4002\_0000h 基准 + 3h 偏移 = 4002\_0003h

**FTMRE\_FCLKDIV 字段描述**

字段	描述
7 FDIVLD	<p>时钟分频器已加载</p> <p>0 自上次复位后未对 FCLKDIV 寄存器采取写入操作。 1 自上次复位后已对 FCLKDIV 寄存器采取写入操作。</p>
6 FDIVLCK	时钟分频器已锁定

下一页继续介绍此表...

**FTMRE\_FCLKDIV 字段描述 (继续)**

字段	描述
	0 FDIV 字段已开放，可进行写入操作。 1 FDIV 值已锁定，不能更改。锁定位设置为高电平后，只有复位才能将该位清零并恢复 FDIV 字段在用户模式下的可写性。
FDIV	时钟分频位  必须将 FDIV[5:0]合理地设置，才能有效地将 BUSCLK 分频到 1MHz，以便在 Flash 编程和擦除算法期间控制定时事件。有关基于 BUSCLK 频率的 FDIV 建议值，请参见 <a href="#">写 FCLKDIV 寄存器</a> 中的表格。

**18.4.4 Flash 状态寄存器 (FTMRE\_FSTAT)**

FSTAT 寄存器报告 Flash 模块的操作状态。

地址: 4002\_0000h 基准 + 5h 偏移 = 4002\_0005h

位	7	6	5	4	3	2	1	0
读	CCIF	0	ACCERR	FPVIOL	MGBUSY	0	MGSTAT	
写								

复位      1      0      0      0      0      0      0      0

**FTMRE\_FSTAT 字段描述**

字段	描述
7 CCIF	命令完成中断标志  指明某个 Flash 命令已完成。将 1 写入 CCIF 可将 CCIF 标志清零以启动某个命令，CCIF 在命令完成或命令出错之前会一直保持低电平。  0 Flash 命令正在处理中。 1 Flash 命令已完成。
6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5 ACCERR	Flash 访问错误标志  指明 Flash 存储器遭遇了非法访问，其原因要么是违反了命令写入序列，要么是发出了非法 Flash 命令。ACCERR 置位时，将 CCIF 标志清零无法启动某个命令。将 1 写入该字段可将其清零，写入 0 则没有影响。  0 没有检测到访问错误。 1 检测到访问错误。
4 FPVIOL	Flash 保护违反标志  指明在命令写入序列期间某个受保护的 Flash 存储器区域中的地址遭遇了试图对其进行编程或擦除的操作。向 FPVIOL 写入 1 可将该字段清空，写入 0 则没有影响。FPVIOL 置位时，无法启动命令或开始某个命令写入序列。  0 没有检测到保护违反情况。 1 检测到保护违反情况。

下一页继续介绍此表...

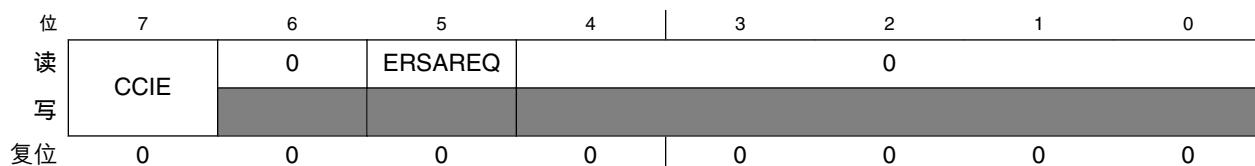
**FTMRE\_FSTAT 字段描述 (继续)**

字段	描述
3 MGBUSY	<p>存储器控制器忙标志</p> <p>反映存储器控制器的有效状态。</p> <p>0 存储器控制器空闲。 1 存储器控制器忙于执行某个 Flash 命令(CCIF = 0)。</p>
2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
MGSTAT	<p>存储器控制器命令完成状态标志</p> <p>执行某个 Flash 命令期间或 Flash 复位序列期间，如果检测到错误，将有一个或多个 MGSTAT 标志位置位。</p> <p>注：如果在复位序列期间检测到双位错误，则复位值可能会与表中显示的值不同。</p>

**18.4.5 Flash 配置寄存器 (FTMRE\_FCNFG)**

FCNFG 寄存器可以使能 Flash 命令完成中断，并在从 CPU 进行 Flash 阵列读访问时强制生成 ECC 故障。

地址: 4002\_0000h 基准 + 7h 偏移 = 4002\_0007h

**FTMRE\_FCNFG 字段描述**

字段	描述
7 CCIE	<p>命令完成中断使能</p> <p>在完成某个 Flash 命令时控制中断的生成。</p> <p>0 命令完成中断已禁用。 1 无论何时，只要 FSTAT 寄存器中的 CCIF 标志置位，就会请求中断。</p>
6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5 ERSAREQ	<p>调试器整体擦除请求</p> <p>请求 MGATE 执行“擦除所有区块”命令并解除加密状态。无法直接对 ERSAREQ 采取写入操作，但可由用户间接控制。</p> <p>MGATE 开始执行序列时，ERSAREQ 字段将置 1。操作结束后，ERSAREQ 将由 MGATE 复位为 0。</p> <p>0 无请求或请求完成 1 请求 <ul style="list-style-type: none"> <li>• 运行“擦除所有区块”命令</li> <li>• 检验擦除状态</li> </ul> </p>

下一页继续介绍此表...

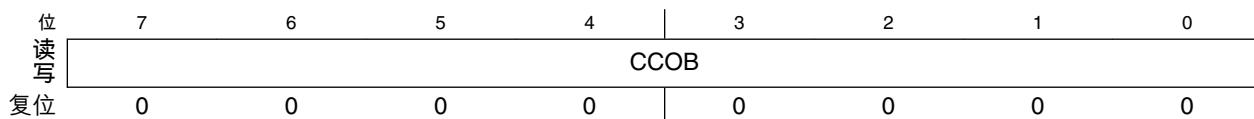
**FTMRE\_FCNFG 字段描述 (继续)**

字段	描述
	<ul style="list-style-type: none"> <li>将 Flash 配置字段中的安全字节编程为解密状态</li> <li>通过将 FSEC[SEC]设置为解密状态以解除 MCU 加密状态</li> </ul>
保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

**18.4.6 Flash 通用命令对象寄存器：低位 (FTMRE\_FCCOBLO)**

FCCOB 是六个字组成的阵列，通过 FCCOBIX 寄存器中找到 CCOBIX 索引进行寻址。允许对 FCCOB 寄存器进行字节范围的读写操作。

地址: 4002\_0000h 基准 + 8h 偏移 = 4002\_0008h

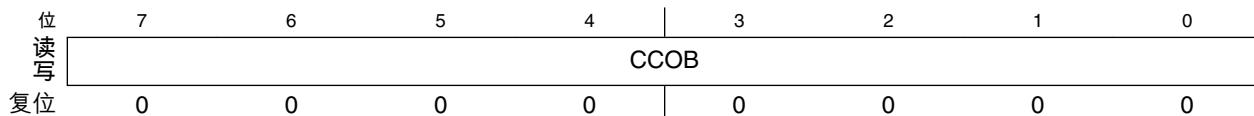
**FTMRE\_FCCOBLO 字段描述**

字段	描述
CCOB	通用命令对象位 7:0 通用命令对象寄存器的低 8 位

**18.4.7 Flash 通用命令对象寄存器：高位 (FTMRE\_FCCOBHI)**

FCCOB 是六个字组成的阵列，通过 FCCOBIX 寄存器中找到 CCOBIX 索引进行寻址。允许对 FCCOB 寄存器进行字节范围的读写操作。

地址: 4002\_0000h 基准 + 9h 偏移 = 4002\_0009h

**FTMRE\_FCCOBHI 字段描述**

字段	描述
CCOB	通用命令对象位 15:8 通用命令对象寄存器的高 8 位

### 18.4.8 Flash 保护寄存器 (FTMRE\_FPROT)

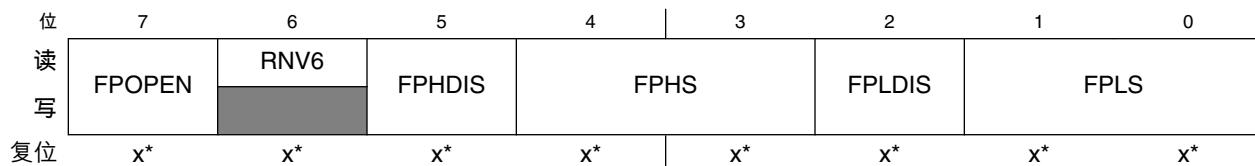
FPROT 寄存器定义了针对编程和擦除操作，为哪些 Flash 扇区提供保护。

FPROT 寄存器的未保留位是可写的，但限制条件是受保护区域的大小只能增加（参见[保护](#)）。

在复位序列期间，会以 Flash 存储器中 Flash 配置字段内 Flash 保护字节的内容加载 FPROT 寄存器，该字段的全局地址为 0x40D。要更改复位序列期间加载的 Flash 保护，必须取消保护 Flash 存储器的高位扇区，然后必须对 Flash 保护字节重新编程。

尝试更改 Flash 存储器中任何受保护区域的数据都会导致保护违反错误，并且 FSTAT 寄存器中的 FPVIOL 位将置位。如果 Flash 数据块中包含的任何 Flash 扇区处于受保护状态，将无法进行该 Flash 数据块的擦除。

地址: 4002\_0000h 基准 + Bh 偏移 = 4002\_000Bh



**FTMRE\_FPROT 字段描述**

字段	描述
7 FPOOPEN	Flash 保护操作使能 FPOOPEN 位确定是否针对编程或擦除操作使用保护功能。 0 FPOOPEN 清零时，FPHDIS 和 FPLDIS 字段用于定义未受保护区域（由 FPHS 和 FPLS 指定地址范围）。 1 FPOOPEN 置位时，FPHDIS 和 FPLDIS 字段用于定义受保护区域（由 FPHS 和 FPLS 指定地址范围）。
6 RNV6	保留的非易失位 RNV 位必须保持在已擦除状态。
5 FPHDIS	Flash 保护较高地址范围禁用 FPHDIS 位确定 Flash 存储器中某个特定区域内是否有以全局地址 0x7FFF 结束的受保护/无保护的区域。 0 使能保护/无保护。 1 禁用保护/无保护。
4-3 FPHS	Flash 保护较高地址大小 FPHS 位确定 Flash 存储器中受保护/无保护区域的大小。只有在 FPHDIS 位已置位的情况下，才能对 FPHS 位采取写入操作。
2 FPLDIS	Flash 保护较低地址范围禁用 FPLDIS 位确定 Flash 存储器中某个特定区域内是否有以全局地址 0x0_0000 开始的受保护/无保护的区域。 0 使能保护/无保护。 1 禁用保护/无保护。

下一页继续介绍此表...

**FTMRE\_FPROT 字段描述 (继续)**

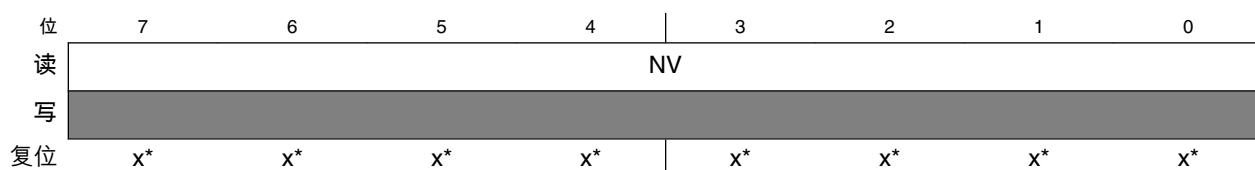
字段	描述
FPLS	Flash 保护较低地址大小 FPLS 位确定 Flash 存储器中受保护/无保护区域的大小。只有在 FPLDIS 位已置位的情况下，才能对 FPLS 位采取写入操作。

**18.4.9 Flash 选项寄存器 (FTMRE\_FOPT)**

FOPT 寄存器是 Flash 选项寄存器。

在复位序列期间，会按复位条件所指示的那样从 Flash 存储器中 Flash 配置字段内的 Flash 非易失字节加载 FOPT 寄存器，该字段的全局地址为 0x040F。

地址: 4002\_0000h 基准 + Fh 偏移 = 4002\_000Fh



\* 注:

- x = 复位时未定义。

**FTMRE\_FOPT 字段描述**

字段	描述
NV	非易失位 NV[7:0]位可以是非易失位。在复位序列期间，会从 Flash 存储器中 Flash 配置字段内的 Flash 非易失字节加载 FOPT 寄存器，该字段的全局地址为 0x40F。

# 第 19 章

## Flash 存储器控制器(FMC)

### 19.1 简介

Flash 存储器控制器(FMC)是一个存储器加速单元。下文列出了 FMC 具备的特性。

- 它是总线上的主设备与 32 位 P-Flash。
- 具有一个缓冲区和一个高速缓存(Cache)，可用于加速 P-Flash 的数据传输。

#### 19.1.1 概述

Flash 存储器控制器管理总线主设备与 32 位 P-Flash。FMC 接收详细配置 Flash 存储器的状态信息，并利用此信息确保接口合适。FMC 支持对 P-Flash 进行 8 位、16 位及 32 位读取操作。对 P-Flash 进行任何写入操作都会导致总线错误。

此外，FMC 还提供两套独立的机制用于加速总线主设备与 P-Flash 之间的接口。32 位推测缓冲器可预取下一个 32 位 Flash 存储器位置，而 4 路、2 组的 P-Flash 高速缓存可存储之前访问过的 P-Flash 数据，以便快速访问。

#### 19.1.2 特性

FMC 模块特性包括：

- 总线主设备与 32 位 P-Flash 之间的接口：
  - 对非易失性 Flash 存储器（包括用作数据 Flash 存储器的 FlexNVM）。
- 加快从 P-Flash 到该器件的数据传输：
  - 有一个 32bit 预取推测缓冲区用于 P-Flash 访问，且可控制它用于指令或是数据访问
  - 4 路、2 组、每行 32bit 的 P-Flash 高速缓存，用于总计八个 32 位带失效控制的入口

## 19.2 工作模式

FMC 只在某个总线主设备访问 P-Flash 时工作。

从芯片功耗模式角度来看：

- FMC 只在 Run 和 Wait 模式下工作。
- 对于任何无法访问 P-Flash 的功耗模式，FMC 都将禁用。

## 19.3 外部信号说明

FMC 没有外部（片外）信号。

## 19.4 存储器映像和寄存器说明

MCM 的编程模型提供了有关 FMC 特性的控制和配置信息。

有关详情，请参见 MCM 平台控制寄存器(PLACR)的说明。

## 19.5 功能说明

FMC 是一个 Flash 加速单元，拥有可供用户配置的灵活缓冲区。

除了管理总线主设备与 P-Flash 之间的接口以外，FMC 还可用于定制 P-Flash 高速缓存和缓冲区，以便提供单周期系统时钟数据访问次数。无论何时，只要命中预取推测缓冲区或高速缓存（已使能的情况下），请求的数据就会在一个系统时钟周期内完成传输。

系统复位时，FMC 配置如下：

- Flash 高速缓存已使能。
- 指令推测和高速缓存已使能。
- 数据推测已禁用。
- 数据高速缓存已使能。

尽管默认配置提供了 Flash 加速，高级用户仍然可能希望定制 FMC 缓冲区配置，以便最大限度地提高其用例的吞吐量。例如，用户可能会调节控制，从而根据访问类型（数据或指令）使能缓冲。

**注**

重新配置 FMC 时，请勿在访问 P-Flash 的同时对 FMC 的控制和配置输入进行编程。相反，请在监控器模式下从 RAM 执行例程，由此达到更改目的。



# 第 20 章 内部时钟源(ICS)

## 20.1 简介

内部时钟源(ICS)模块为 MCU 提供时钟源选择。该模块包含一个锁频环(FLL)作为时钟源，该锁频环可以采用内部或外部的基准时钟。ICS 模块可以把 FLL 时钟、内部基准时钟或外部基准时钟作为 MCU 系统时钟的时钟源。另外还提供了一些信号来控制低功耗振荡器(OSC)模块。这些信号配置并使能 OSC 模块以生成其外部晶体/谐振器时钟(OSC\_OUT)，提供给其他外部模块使用，同时也作为 ICS 外部基准时钟源。ICS 外部基准时钟可以是 OSC 提供的外部晶体/谐振器(OSC\_OUT)，也可以是另一个外部时钟源。

选定的 ICS 时钟源通过可编程分频器 (BDIV) 传送出去，最终输出较低的时钟频率。

### 20.1.1 特性

下面是 ICS 模块的主要特性：

- 内部基准时钟的准确度可以校正。
- 可使用内部或外部基准时钟来控制 FLL。
- 针对外部时钟提供了基准分频器，以提供给 FLL 恰当的输入频率。
- 内部基准时钟具有 9 个调整位。
- 内部或外部基准时钟都可以被选作 MCU 的时钟源。
- “FLL 内部启用”模式在芯片复位后被自动选中。
- FLL 锁定检测器和外部时钟监视器
  - 具有中断功能的 FLL 锁定检测器
  - 具有复位功能的外部基准时钟监视器
- 数字控制振荡器针对 40-48 MHz 频率范围进行了优化

### 20.1.2 结构框图

下图是 ICS 结构框图。

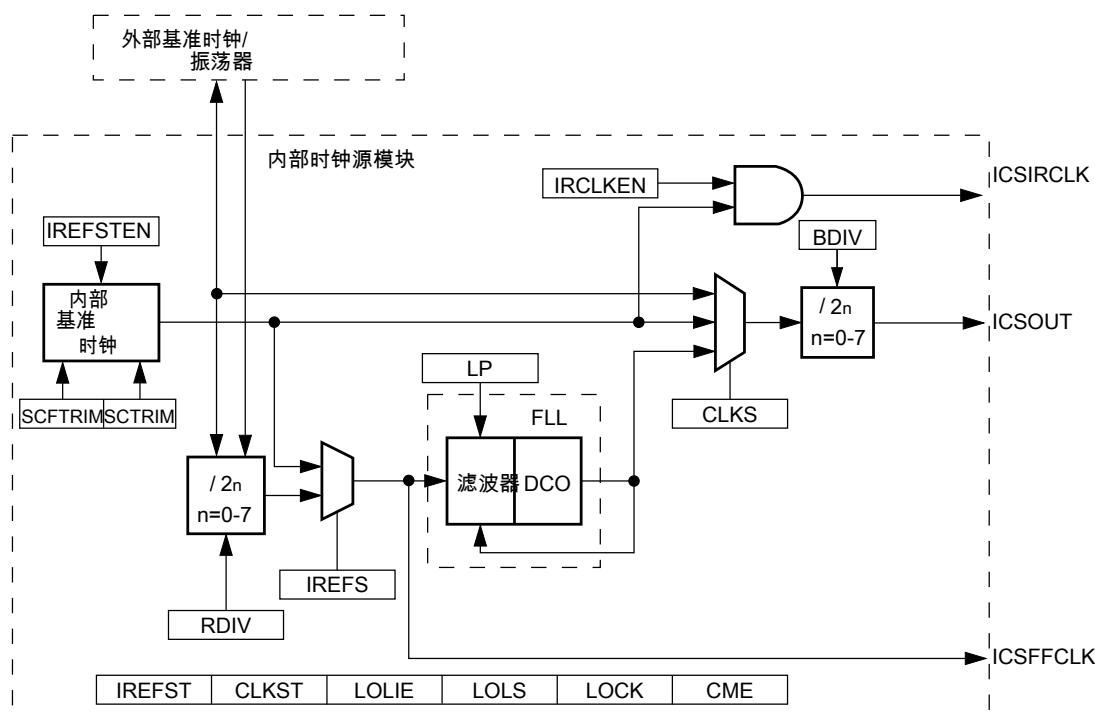


图 20-1. 内部时钟源(ICS)结构框图

### 20.1.3 工作模式

ICS 有七种工作模式：FEI、FEE、FBI、FBILP、FBE、FBELP 和 STOP。以下各小节会分别对每一种工作模式作简要说明。

#### 20.1.3.1 FLL 内部启用(FEI)

“FLL 内部启用”模式是默认模式，在该模式下，ICS 提供一个来源于 FLL 的时钟，FLL 时钟由内部基准时钟控制。

#### 20.1.3.2 FLL 外部启用(FEE)模式

在 FLL 外部启用模式下，ICS 提供一个来源于 FLL 的时钟，该 FLL 时钟由外部基准时钟控制。

#### 20.1.3.3 FLL 内部旁路(FBI)

在 FLL 内部旁路模式下，FLL 使能并通过内部基准时钟来控制，但处于旁路状态。ICS 提供的输出是从内部基准时钟分频而来的时钟。

#### 20.1.3.4 FLL 内部旁路低功耗(FBILP)

在 FLL 内部旁路低功耗模式下，FLL 被禁用并旁路，ICS 提供的输出是由内部基准时钟分频而来的时钟。

#### 20.1.3.5 FLL 外部旁路(FBE)

在 FLL 外部旁路模式下，FLL 使能并通过外部基准时钟来控制，但处于旁路状态。ICS 提供的输出是从外部基准时钟源分频而来的时钟。

#### 20.1.3.6 FLL 外部旁路低功耗(FBELP)

在 FLL 外部旁路低功耗模式下，FLL 禁用且旁路，ICS 提供的输出是由外部基准时钟分频而来的时钟。

#### 20.1.3.7 停止(STOP)

在 Stop 模式下，FLL 被禁用，可以选择使能或者禁用内部或 ICS 外部基准时钟源(OSC\_OUT)。ICS 不提供任何 MCU 时钟源。

##### 注

DCO 频率从预停止值变为其复位值，FLL 需要在频率稳定前重新获取锁定。时序敏感的操作在执行前，必须等待长度为 FLL 获取时间  $t_{Acquire}$  的一段时间。

### 20.2 外部信号说明

没有任何 ICS 信号被连接到芯片外部。

### 20.3 寄存器定义

#### ICS 存储器映射

绝对地址(十六进制)	寄存器名称	宽度(单位:位)	访问	复位值	小节/页
4006_4000	ICS 控制寄存器 1 (ICS_C1)	8	R/W	04h	<a href="#">20.3.1/274</a>

下一页继续介绍此表...

## ICS 存储器映射 (继续)

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4006_4001	ICS 控制寄存器 2 (ICS_C2)	8	R/W	20h	20.3.2/275
4006_4002	ICS 控制寄存器 3 (ICS_C3)	8	R/W	未定义	20.3.3/276
4006_4003	ICS 控制寄存器 4 (ICS_C4)	8	R/W	参见章节	20.3.4/276
4006_4004	ICS 状态寄存器 (ICS_S)	8	R	10h	20.3.5/277

## 20.3.1 ICS 控制寄存器 1 (ICS\_C1)

地址: 4006\_4000h 基准 + 0h 偏移 = 4006\_4000h

位 读写	7	6	5	4	3	2	1	0
	CLKS		RDIV		IREFS	IRCLKEN	IREFSTEN	
复位	0	0	0	0	0	1	0	0

## ICS\_C1 字段描述

字段	描述																											
7–6 CLKS	时钟源选择 选择控制总线频率的时钟源。实际总线频率取决于 ICS_C2[BDIV]的值。 00 选择 FLL 的输出。 01 选择内部基准时钟。 10 选择外部基准时钟。 11 保留，默认值为 00。																											
5–3 RDIV	基准分频器 修改 RDIV 寄存器可以改变 FLL 的基准时钟，在 FEE 和 FBE 模式下，不允许修改 RDIV 寄存器。 通过 IREFS 寄存器位可以为 FLL 选择适当的降频参数。输出的结果频率必须在 31.25 kHz 到 39.0625 kHz 的范围内。  <table border="1"> <thead> <tr> <th>RDIV</th> <th>OSC_CR[RANGE ]= 0</th> <th>OSC_CR[RANGE ]= 1</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>1<sup>1</sup></td> <td>32</td> </tr> <tr> <td>001</td> <td>2</td> <td>64</td> </tr> <tr> <td>010</td> <td>4</td> <td>128</td> </tr> <tr> <td>011</td> <td>8</td> <td>256</td> </tr> <tr> <td>100</td> <td>16</td> <td>512</td> </tr> <tr> <td>101</td> <td>32</td> <td>1024</td> </tr> <tr> <td>110</td> <td>64</td> <td>保留</td> </tr> <tr> <td>111</td> <td>128</td> <td>保留</td> </tr> </tbody> </table> 1. 复位默认值	RDIV	OSC_CR[RANGE ]= 0	OSC_CR[RANGE ]= 1	000	1 <sup>1</sup>	32	001	2	64	010	4	128	011	8	256	100	16	512	101	32	1024	110	64	保留	111	128	保留
RDIV	OSC_CR[RANGE ]= 0	OSC_CR[RANGE ]= 1																										
000	1 <sup>1</sup>	32																										
001	2	64																										
010	4	128																										
011	8	256																										
100	16	512																										
101	32	1024																										
110	64	保留																										
111	128	保留																										
2 IREFS	内部基准选择																											

下一页继续介绍此表...

**ICS\_C1 字段描述 (继续)**

字段	描述
	选择 FLL 的基准时钟源。 0 选择外部基准时钟。 1 选择内部基准时钟。
1 IRCLKEN	内部基准时钟使能 使能内部基准时钟用作 ICSIRCLK。 0 ICSIRCLK 无效。 1 ICSIRCLK 有效。
0 IREFSTEN	内部基准停止使能 控制内部基准时钟在 ICS 进入 Stop 模式时是否保持使能。 0 内部基准时钟在 Stop 模式下禁用。 1 如果 IRCLKEN 置位，或者 ICS 在进入 Stop 模式之前处于 FEI、FBI 或 FBILP 模式，则内部基准时钟在 Stop 模式下保持使能。

## 1. 复位默认值

**20.3.2 ICS 控制寄存器 2 (ICS\_C2)**

地址: 4006\_4000h 基准 + 1h 偏移 = 4006\_4001h

位	7	6	5	4	3	2	1	0
读写	BDIV		LP		0		0	
复位	0	0	1	0	0	0	0	0

**ICS\_C2 字段描述**

字段	描述
7–5 BDIV	总线分频器 选择 ICS_C1[CLKS]选定的时钟源的分频比。该字段控制总线频率。 000 编码 0—选定时钟 1 分频。 001 编码 1—选定时钟 2 分频 (复位默认值)。 010 编码 2—选定时钟 4 分频。 011 编码 3—选定时钟 8 分频。 100 编码 4—选定时钟 16 分频。 101 编码 5—选定时钟 32 分频。 110 编码 6—选定时钟 64 分频。 111 编码 7—选定时钟 128 分频。
4 LP	低功耗选择 控制 FLL 在 FLL 旁路模式下是否禁用。 0 FLL 在旁路模式下不禁用。 1 FLL 在旁路模式下禁用 (除非调试有效)。

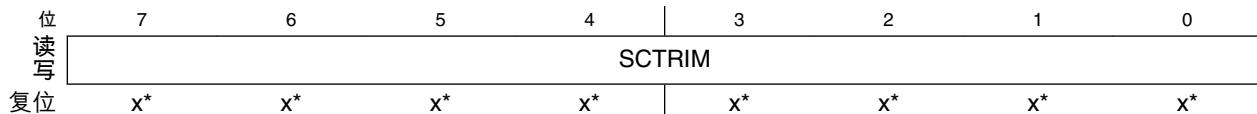
下一页继续介绍此表...

**ICS\_C2 字段描述 (继续)**

字段	描述
保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

**20.3.3 ICS 控制寄存器 3 (ICS\_C3)**

地址: 4006\_4000h 基准 + 2h 偏移 = 4006\_4002h



\* 注:

- x = 复位时未定义。

**ICS\_C3 字段描述**

字段	描述
SCTRIM	<p>慢速内部基准时钟调整设置</p> <p>通过控制内部基准时钟周期来控制慢速内部基准时钟频率。这些位采用二进制加权。换言之，位 1 的调整量是位 0 的两倍。提高 SCTRIM 的二进制值会加长周期，降低该值会缩短周期。还有一个精密调整位 ICS_C4[SCFTRIM]。</p> <p>除了 Debug 模式外，ICS_C3 在复位后会自动载入工厂校准值。这个值是在芯片生产出厂前校准的，以使内部时钟满足 fint_ft 数据手册的描述。用户可以在 fint_t 范围内使用自己的校正值，这个值需要被写入 Flash 的保留空间 0x0000_03FF 中，并且在初始化代码中复制到 ICS_C3 寄存器。</p>

**20.3.4 ICS 控制寄存器 4 (ICS\_C4)**

地址: 4006\_4000h 基准 + 3h 偏移 = 4006\_4003h



\* 注:

- x = 复位时未定义。

**ICS\_C4 字段描述**

字段	描述
7 LOLIE	<p>失锁中断</p> <p>决定在失锁指示出现后是否生成中断请求。该字段仅在 ICS_S[LOLS]置位时有效。</p>

下一页继续介绍此表...

**ICS\_C4 字段描述 (继续)**

字段	描述
	0 失锁时不生成中断请求。 1 失锁时生成中断请求。
6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5 CME	时钟监控使能  决定在外部时钟丢失指示出现后是否生成复位请求。仅当 ICS 处于使用外部时钟的工作模式 ( FEE、FBE 或 FBELP ) 时，才必须将该字段设置为逻辑 1。  0 时钟监控禁用。 1 丢失外部时钟时生成复位请求。
4-1 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
0 SCFTRIM	慢速内部基准时钟精密调整  控制内部基准时钟频率的最小调整量。SCFTRIM 置位将加长周期，SCFTRIM 清零将缩短周期，调整量为最小可能的量。  除了 Debug 模式外，ICS_C4[SCFTRIM]在复位后会自动载入工厂校准值。这个值是在芯片生产出厂前校准的，以使内部时钟满足 fint_ft 数据手册的描述。用户可以在 fint_t 范围内使用自己的校正值，这个值需要被写入 Flash 的保留空间 0x0000_03FE 中，并且在初始化代码中复制到 ICS_C4 寄存器。

**20.3.5 ICS 状态寄存器 (ICS\_S)**

地址: 4006\_4000h 基准 + 4h 偏移 = 4006\_4004h

位	7	6	5	4	3	2	1	0
读	LOLS	LOCK	0	IREFST	CLKST		0	
写	w1c							
复位	0	0	0	1	0	0	0	0

**ICS\_S 字段描述**

字段	描述
7 LOLS	失锁状态  指示 FLL 的锁定状态。在使能了锁定检测并且已经锁定后，如果 FLL 输出频率已不在±4.7%至±5.97%的锁定退出频率容差范围内，LOLS 会置位。ICS_C4[LOLIE]决定置位时是否生成中断请求。LOLS 在置位时通过复位或写入逻辑 1 清零。将逻辑 0 写入 LOLS 无效。  0 FLL 自上次 LOLS 清零以来未失锁。 1 FLL 自上次 LOLS 清零以来已失锁。
6 LOCK	锁定状态  指示 FLL 是否已获取锁定。锁定检测在 FLL 禁用时也禁用。锁定状态位置位时，如果更改 IREFS、RDIV[2:0] 或 SCTRIM[7:0] ( 处于 FEI 或 FBI 模式时 ) 中任一字段的值，则会导致锁定状态位清零，清零状态将保持到 FLL 重新获取锁定为止。进入 Stop 模式也会导致锁定状态位清零，清零状态将保持到 FLL 重新获取锁定为止。

下一页继续介绍此表...

**ICS\_S 字段描述 (继续)**

字段	描述
	0 FLL 当前未锁定。 1 FLL 当前已锁定。
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
4 IREFST	内部基准状态  指示基准时钟的当前来源。由于时钟域之间的内部同步，该字段不会在写入 ICS_C1[IREFS]后立即更新。  0 基准时钟来源于外部时钟。 1 基准时钟来源于内部时钟。
3-2 CLKST	时钟模式状态  指示当前时钟模式。由于时钟域之间的内部同步，该字段不会在写入 ICS_C1[CLKS]后立即更新。  00 选择 FLL 的输出。 01 FLL 旁路，选择内部基准时钟。 10 FLL 旁路，选择外部基准时钟。 11 保留。
保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

## 20.4 功能说明

### 20.4.1 工作模式

状态图中显示了 ICS 的 7 种状态，介绍如下。箭头表示各状态间允许的切换。

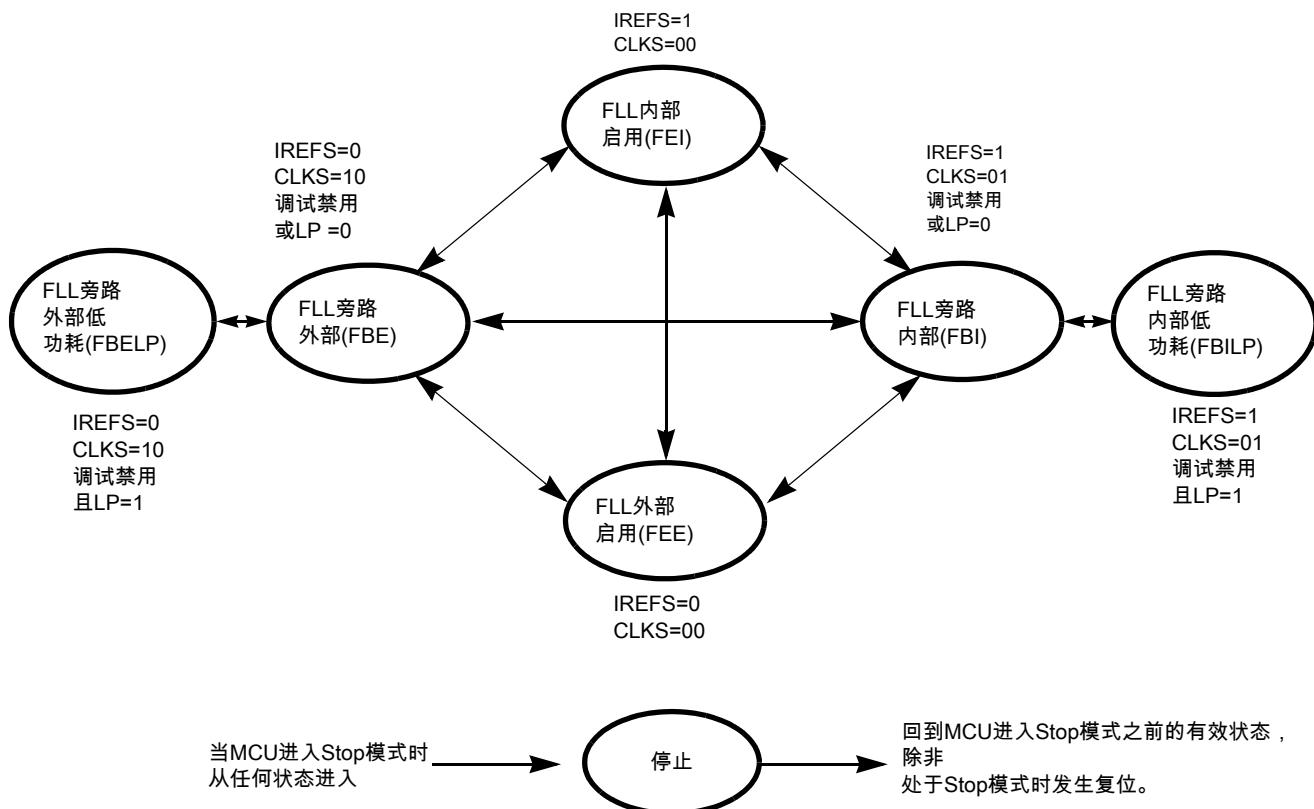


图 20-2. 时钟切换模式

#### 20.4.1.1 FLL 内部启用(FEI)

FLL 内部启用(FEI)模式是默认工作模式，当出现以下所有条件时进入该模式：

- 00b 写入 ICS\_C1[CLKS]。
- 1b 写入 ICS\_C1[IREFS]。

在“FLL 内部启用”模式下，ICSOUT 时钟从 FLL 时钟获得，FLL 时钟由内部基准时钟控制。FLL 环路将频率锁定到内部基准频率的 1280 倍。内部基准时钟使能。

#### 20.4.1.2 FLL 外部启用(FEE)

出现以下所有条件时，进入 FLL 外部启用(FEE)模式：

- 00b 写入 ICS\_C1[CLKS]。
- 0b 写入 ICS\_C1[IREFS]。
- 对 ICS\_C1[RDIV]和 OSC\_CR[RANGE] 进行写操作可将外部基准时钟分频到 31.25 kHz 至 39.0625 kHz 范围内。

在“FLL 外部启用”模式下，ICSOUT 时钟从 FLL 时钟获得，FLL 时钟由外部基准时钟源控制。FLL 环路将 FLL 频率锁定到外部基准频率（由 ICS\_C1[RDIV] 和 OSC\_CR[RANGE] 选择）的 1280 倍。可以使用外部基准时钟也被启用。

#### 20.4.1.3 FLL 内部旁路(FBI)

出现以下所有条件时，进入 FLL 内部旁路(FBI)模式：

- 01b 写入 ICS\_C1[CLKS]。
- 1b 写入 ICS\_C1[IREFS]。

在“FLL 内部旁路”模式下，ICSOUT 时钟从内部基准时钟获得。FLL 时钟由内部基准时钟控制，FLL 环路将 FLL 频率锁定到内部基准频率的1280 倍。内部基准时钟使能。

#### 20.4.1.4 FLL 内部旁路低功耗(FBILP)

出现以下所有条件时，进入 FLL 内部旁路低功耗(FBILP)模式：

- 01b 写入 ICS\_C1[CLKS]。
- 1b 写入 ICS\_C1[IREFS]。

在“FLL 内部旁路低功耗”模式下，ICSOUT 时钟从内部基准时钟获得，FLL 禁用。内部基准时钟使能。

#### 20.4.1.5 FLL 外部旁路(FBE)

出现以下所有条件时，进入 FLL 外部旁路(FBE)模式：

- 10b 写入 ICS\_C1[CLKS]。
- 0b 写入 ICS\_C1[IREFS]。
- 对 ICS\_C1[RDIV] 和 OSC\_CR[RANGE] 字段进行写操作可将外部基准时钟分频到 31.25 kHz 至 39.0625 kHz 范围内。

在 FLL 外部旁通模式下，ICSOUT 时钟由外部基准时钟源分频而来。FLL 时钟由外部基准时钟控制，FLL 环路将 FLL 频率锁定到外部基准频率（由 ICS\_C1[RDIV] 和 OSC\_CR[RANGE]）。

#### 20.4.1.6 FLL 外部旁路低功耗(FBELP)

出现以下所有条件时，进入 FLL 外部旁路低功耗(FBELP)模式：

- 10b 写入 ICS\_C1[CLKS]。
- 0b 写入 ICS\_C1[IREFS]。

在“FLL 外部旁路低功耗”模式下，ICSOUT 时钟从外部基准时钟源获得，FLL 禁用。外部基准时钟源使能。

#### 20.4.1.7 停止

##### 注

DCO 频率从预停止值变为其复位值，FLL 需要在频率稳定前重新获取锁定。时序敏感的操作在执行前，必须等待长度为 FLL 获取时间  $t_{Acquire}$  的一段时间。

只要 MCU 进入 STOP 状态，就会进入 Stop 模式。在该模式下，所有 ICS 时钟信号都保持静态，但以下情况例外：

在满足以下所有条件时，ICSIRCLK 将在 Stop 模式下激活：

- 1b 写入 ICS\_C1[IRCLKEN]。
- 1b 写入 ICS\_C1[IREFSTEN]。

#### 20.4.2 模式切换

ICS\_C1[IREFS]可以随时更改，但由 ICS\_S[IREFST]指示到新选择的时钟的实际切换。在 FLL 内部使用 (FEI) 模式和 FLL 外部使用 (FEE) 模式间进行切换时，当切换完成后，FLL 将再次开始锁定。

ICS\_C1[CLKS]也可以随时更改，但由 ICS\_S[CLKST]指示到新选择的时钟的实际切换。如果新选择的时钟不可用，将继续选择上一个时钟。

##### 注

从 FEE、FBE 或 FBELP 切换到 FEI 模式时，建议先等待 IREFST 切换完成，然后再更改 ICS\_C1[CLKS]。

#### 20.4.3 总线频率分频器

ICS\_C2[BDIV]可以随时更改，并且实际转换到新频率会立即发生。

#### 20.4.4 低功耗字段的使用

ICS\_C2 寄存器中的低功耗(LP)字段在 FLL 不使用时可将其禁用从而降低功耗。

但在某些应用中，在切换到 FLL 使用模式前，可能需要启用 FLL 并锁定它，以最大限度提高准确度。为此，将 0b 写入 ICS\_C2[LP]。

## 20.4.5 内部参考时钟

ICS\_C1[IRCLKEN]置位时，内部基准时钟信号显示为 ICSIRCLK，它可以用作附加时钟源。要重新设定 ICSIRCLK 频率，请将新值写入 ICS\_C3[SCTRIM]位来调整内部基准时钟的周期：

- 如果写入更大的值，将减慢 ICSIRCLK 频率。
- 将较小的值写入 ICS\_C3 寄存器会提高 ICSIRCLK 频率。

如果 ICS 处于 FLL 内部使用(FEI)模式、FLL 内部旁通(FBI)模式或 FLL 内部旁通低功耗(FBILP)模式，则 TRIM 位会影响 ICSOUT 频率。

在调整 ICSIRCLK 之前，写入较低的总线分频(ICS\_C2[BDIV])系数可能导致 ICSOUT 频率超过最大芯片级频率，违反芯片级时钟的时序规范。

如果 ICS\_C1[IREFSTEN]置位且 1b 写入 ICS\_C1[IRCLKEN]，那么内部基准时钟在 Stop 模式下继续运行，以便能在退出 Stop 模式时快速恢复。

所有 MCU 器件都在保留的存储器区间写入了出厂校正值。该值在任何的复位初始化期间被复制到 ICS\_C3 寄存器和 ICS\_C4[SCFTRIM]。为了提高精度，在应用中可以重新校准内部振荡器并相应地设置 ICS\_C4[SCFTRIM]。

## 20.4.6 固定频率时钟

ICS 提供经过分频的 FLL 基准时钟作为 ICSFFCLK，以便用作附加时钟源。

ICSFFCLK 频率不能超过 ICSOUT 频率的 1/4 才视为有效。由于这一要求，在旁路模式下，ICSFFCLK 仅对满足以下条件 (ICS\_C2[BDIV]、ICS\_C1[RDIV] 分频比和 OSC\_CR[RANGE] 值) 的外部旁路模式 (FBE 和 FBELP) 有效：

如果 OSC\_CR[RANGE] 为高电平，

- ICS\_C2[BDIV] = 000, ICS\_C2[RDIV] ≥ 010
- ICS\_C2[BDIV] = 001 (2 分频), ICS\_C2[RDIV] ≥ 011
- ICS\_C2[BDIV] = 010 (4 分频), ICS\_C2[RDIV] ≥ 100
- ICS\_C2[BDIV] = 011 (8 分频), ICS\_C2[RDIV] ≥ 101

## 20.4.7 FLL 锁定和时钟监视器

### 20.4.7.1 FLL 时钟锁定

在 FBE 和 FEE 模式下，时钟检测器源使用外部参考作为自身的参考。当检测到 FLL 从锁定变为失锁时，ICS\_S[LOLS]置位。如果 ICS\_C4[LOLIE]置位，则会产生中断。ICS\_S[LOLS]在置位时通过复位或向 ICS\_S[LOLS]写入逻辑 1 清零 ICS\_S[LOLS]。将逻辑 0 写入 ICS\_S[LOLS]无效。

在 FBI 和 FEI 模式下，时钟检测器源使用内部基准作为自身的基准。当检测到 FLL 从锁定变为失锁时，ICS\_S[LOLS]置位。如果 ICS\_C4[LOLIE]置位，则会产生中断。ICS\_S[LOLS]在置位时通过复位或向 ICS\_S[LOLS]写入逻辑 1 清零 ICS\_S[LOLS]。将逻辑 0 写入 ICS\_S[LOLS]无效。

在 FBELP 和 FBILP 模式下，FLL 未启用，因此锁定检测功能不可用。

### 20.4.7.2 外部基准时钟监视器

在 FBE、FEE 或 FBELP 模式下，如将 1 写入 ICS\_C4[CME]，会使能时钟监控器。如果外部基准下降到低于某个特定频率，那么 MCU 将复位。SIM\_SRSID[LOC]将置位以指示错误。

## 20.5 初始化/应用信息

本节通过代码示例为用户介绍如何初始化和配置 ICS 模块。软件示例采用 C 语言实施。

### 20.5.1 初始化 FEI 模式

以下代码段显示如何将 ICS 设置为 FEI 模式。

#### 示例: 20.5.1.1 FEI 模式初始化程序

```
/* the following code segment demonstrates setting the ICS to FEI mode using the factory
trim value. The resulting ICSOUT frequency is fint_ft*1280/BDIV. */
ICS_C2 = 0x20; // BDIV=divide by 2 - use default until clock dividers configured
ICS_C1 = 0x04; // internal reference clock as source to FLL
while ((ICS_S & ICS_S_LOCK_MASK) == 0); // wait for FLL to lock
SIM_CLKDIV = 0x01100000; // core clock = ICSOUT/1 and bus clock = core clock/2
ICS_C2 = 0x00; // BDIV=divide by 1 - allows max core and bus clock frequencies

/* the following code segment demonstrates setting the ICS to FEI mode using a custom trim
value provided by a programming tool. The resulting ICSOUT frequency is fint_t*1280/BDIV. */
ICS_C3 = *((uint8_t*) 0x03FF); // trim internal reference clock
ICS_C4 = *((uint8_t*) 0x03FE); // fine trim internal reference clock
ICS_C2 = 0x20; // BDIV=divide by 2 - use default until clock dividers configured
ICS_C1 = 0x04; // internal reference clock as source to FLL
while ((ICS_S & ICS_S_LOCK_MASK) == 0); // wait for FLL to lock
```

```
SIM_CLKDIV = 0x01100000; // core clock = ICSOUT/1 and bus clock = core clock/2
ICS_C2 = 0x00; // BDIV=divide by 1 - allows max core and bus clock frequencies
```

## 20.5.2 初始化 FBI 模式

以下代码段显示如何将 ICS 设置为 FBI 模式。

### 示例: 20.5.2.1 FBI 模式初始化程序

```
/* the following code segment demonstrates setting the ICS to FBI mode using the factory
trim value. The resulting ICSOUT frequency is fint_ft/BDIV. Note that the FLL will be
running at a frequency of fint_ft*1280/BDIV even though the FLL is bypassed. */
ICS_C2 = 0x20; // BDIV=divide by 2 - use default until clock dividers configured
ICS_C1 = 0x44; // internal reference clock as source for ICSOUT
while ((ICS_S & 0x0C) != 0x04); // wait until internal reference is selected
SIM_CLKDIV = 0x00000000; // core clock = ICSOUT/1; bus clock = core clock/1
ICS_C2 = 0x00; // BDIV=divide by 1 - allows max core and bus clock frequencies
```

## 20.5.3 初始化 FEE 模式

以下代码段显示如何将 ICS 设置为 FEE 模式。

### 示例: 20.5.3.1 FEE 模式初始化程序

```
/* the following code segment demonstrates setting the ICS to FEE mode generating a 40MHZ
core clock frequency using an external 8MHz crystal */
OSC_CR = 0x96; // high-range, high-gain oscillator selected
while ((OSC_CR & OSC_CR_OSCINIT_MASK) == 0); // wait until oscillator is ready
ICS_C2 = 0x20; // BDIV=divide by 2 - use default until clock dividers configured
ICS_C1 = 0x18; // 8MHz external reference clock/256 as source to FLL
while ((ICS_S & ICS_S_IREFST_MASK) == 1); // wait for external source selected
while ((ICS_S & ICS_S_LOCK_MASK) == 0); // wait for FLL to lock
SIM_CLKDIV = 0x01100000; // core clock = ICSOUT/1 and bus clock = core clock/2
ICS_C2 = 0x00; // BDIV=divide by 1 - allows max core and bus clock frequencies
```

## 20.5.4 初始化 FBE 模式

以下代码段显示如何将 ICS 设置为 FBE 模式。

### 示例: 20.5.4.1 FBE 模式初始化程序

```
/* the following code segment demonstrates setting the ICS to FBE mode generating 20MHZ core
clock frequency using an external 20MHz crystal */
OSC_CR = 0x96; // high-range, high-gain oscillator selected
while ((OSC_CR & OSC_CR_OSCINIT_MASK) == 0); // wait until oscillator is ready
ICS_C2 = 0x20; // BDIV=divide by 2 - use default until clock dividers configured
ICS_C1 = 0xA0; // 20MHz external clock as ICSOUT source; FLL source = 20MHz/512
while ((ICS_S & ICS_S_IREFST_MASK) == 1); // wait for external source selected
while ((ICS_S & 0x0C) != 0x08); // wait until FBE mode is selected
```

```
SIM_CLKDIV = 0x00000000; // core clock = ICSOUT/1 and bus clock = core clock/1  
ICS_C2 = 0x00; // BDIV=divide by 1 - allows max core and bus clock frequencies
```



# 第 21 章 振荡器(OSC)

## 21.1 简介

### 21.1.1 概述

OSC 模块为 MCU 提供时钟源。OSC 模块可与外部晶振或谐振器一起产生一个时钟，这个时钟可以被 MCU 用作基准时钟或总线时钟。

### 21.1.2 特性和模式

OSC 模块的主要特性有：

- 支持 32 kHz 晶振（低范围模式）
- 支持 4–24 MHz 晶振和谐振器（高范围模式）
- 使用低功耗模式（低增益模式）可通过自动增益控制(AGC)优化两种频率范围内的功耗
- 两种频率范围内均具有高增益选项：32 kHz, 4–24 MHz
- 提供电压和频率过滤功能，确保时钟的频率和稳定性
- 通过 ICS 可使能支持。

### 21.1.3 结构框图

OSC 模块结构框图请参见下图。

OSC 模块使用晶振或谐振器生成三个经过滤波的振荡器时钟信号(XTL\_CLK)。XTL\_CLK 可在 Stop 模式下工作，因为它们来自始终通电的硬件模块。

OSCos 决定 OSC\_OUT 是来自内部振荡器(XTL\_CLK)还是直接来自 EXTAL 引脚上设定的外部时钟。OSCos 信号允许 XTAL 引脚用作 I/O 或测试时钟。

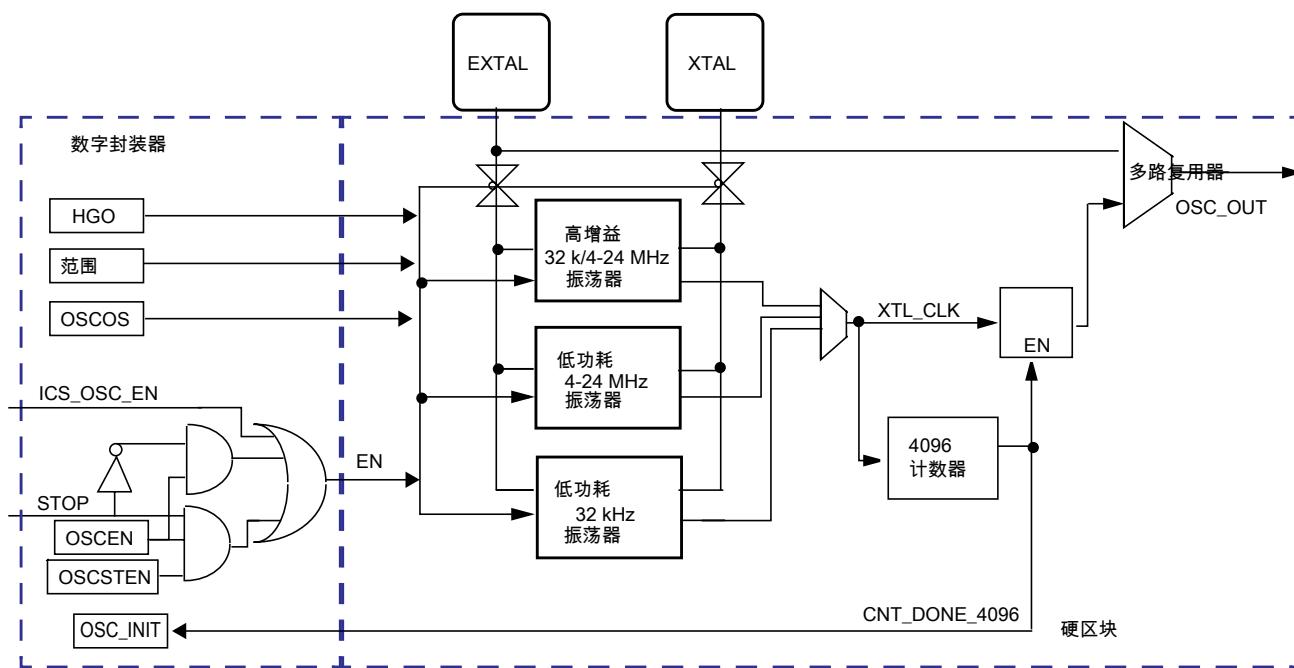


图 21-1. OSC 模块结构框图

## 21.2 信号说明

下表介绍用户可访问的 OSC 模块信号。参见芯片级规格，了解外部引脚上实际连接了哪些信号。

表 21-1. OSC 信号说明

信号	说明	I/O
EXTAL	外部时钟/振荡器输入	模拟输入
XTAL	振荡器输出	模拟输出

## 21.3 外部晶振/谐振器连接

晶振/谐振器频率基准的连接如图 21-2 和图 21-3 所示。在使用低频、低功耗模式时，唯一的外部组件就是晶振或谐振器本身。在其他振荡器模式下，需要使用负载电容器( $C_x, C_y$ )和反馈电阻( $R_F$ )。另外，高增益模式下还要使用串联电阻( $R_S$ )。数据手册中列出了建议的组件值。

表 21-2. 外部晶振/谐振器连接

振荡器模式	连接
低频、高增益	Connection2
低频、低功耗	Connection1
高频、高增益(4–20 MHz)	Connection2
高频、低功耗(4–20 MHz)	Connection2

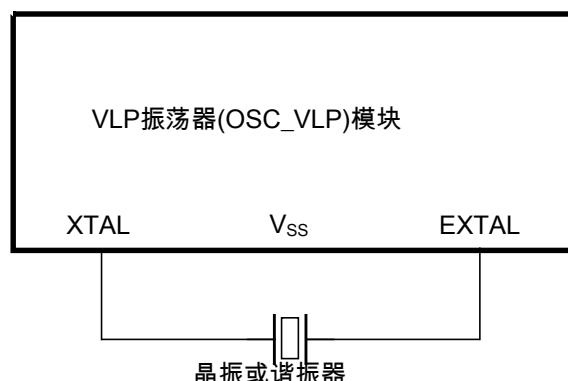


图 21-2. 晶振/谐振器连接 - 连接 1

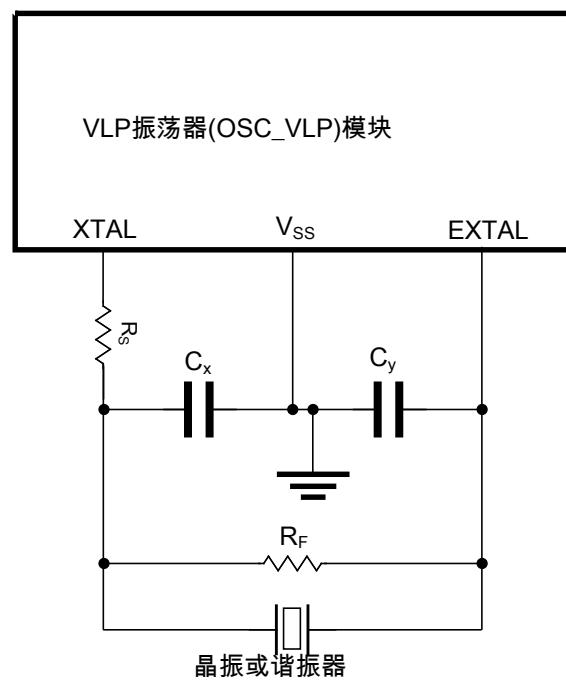


图 21-3. 晶振/谐振器连接 - 连接 2

## 21.4 外部时钟连接

在外部时钟模式下(OSC\_CR[OSCOS] = 0)，引脚可如下图所示连接。

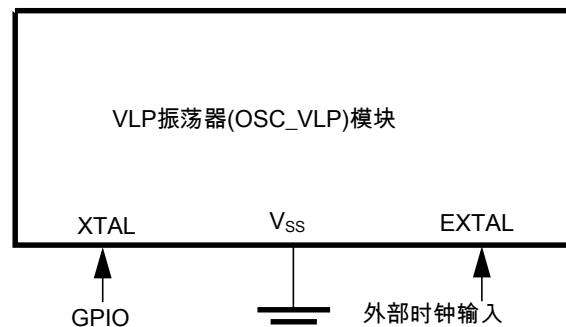


图 21-4. 外部时钟连接

## 21.5 存储器映像和寄存器说明

### OSC 存储器映射

绝对地址(十六进制)	寄存器名称	宽度(单位:位)	访问	复位值	小节/页
4006_5000	OSC 控制寄存器 (OSC_CR)	8	R/W	00h	21.5.1/291

## 21.5.1 OSC 控制寄存器 (OSC\_CR)

地址: 4006\_5000h 基准 + 0h 偏移 = 4006\_5000h

位	7	6	5	4	3	2	1	0
读	OSCEN	0	OSCSTEN	OSCOS	0	RANGE	HGO	OSCINIT
写	0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0

**OSC\_CR 字段描述**

字段	描述
7 OSCEN	OSC 使能 使能 OSC 模块。 OSC 模块也可通过 ICS 模块使能。 0 OSC 模块禁用。 1 OSC 模块使能。
6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5 OSCSTEN	Stop 模式下的 OSC 使能 控制 OSC 时钟在 MCU 进入 Stop 模式且 OSCEN 置位时是否保持使能。 如果 ICS 请求 OSC 使能，则 OSCSTEN 无效。 0 OSC 时钟在 Stop 模式下禁用。 1 OSC 时钟在 Stop 模式下保持使能。
4 OSCOS	OSC 输出选择 选择 OSC 模块的输出时钟。 0 选择来自 EXTAL 引脚的外部时钟源。 1 选择振荡器时钟源。
3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 RANGE	频率范围选择 选择 OSC 模块的频率范围。 此位必须在 OSC 使能之前配置，且在 OSC 使能后不能更改 0 32 kHz 的低频范围。 1 4–24 MHz 的高频范围。
1 HGO	高增益振荡器选择 控制 OSC 工作模式。 0 低功耗模式 1 高增益模式
0 OSCINIT	OSC 初始化 该字段在振荡器初始化周期完成后置位。

下一页继续介绍此表...

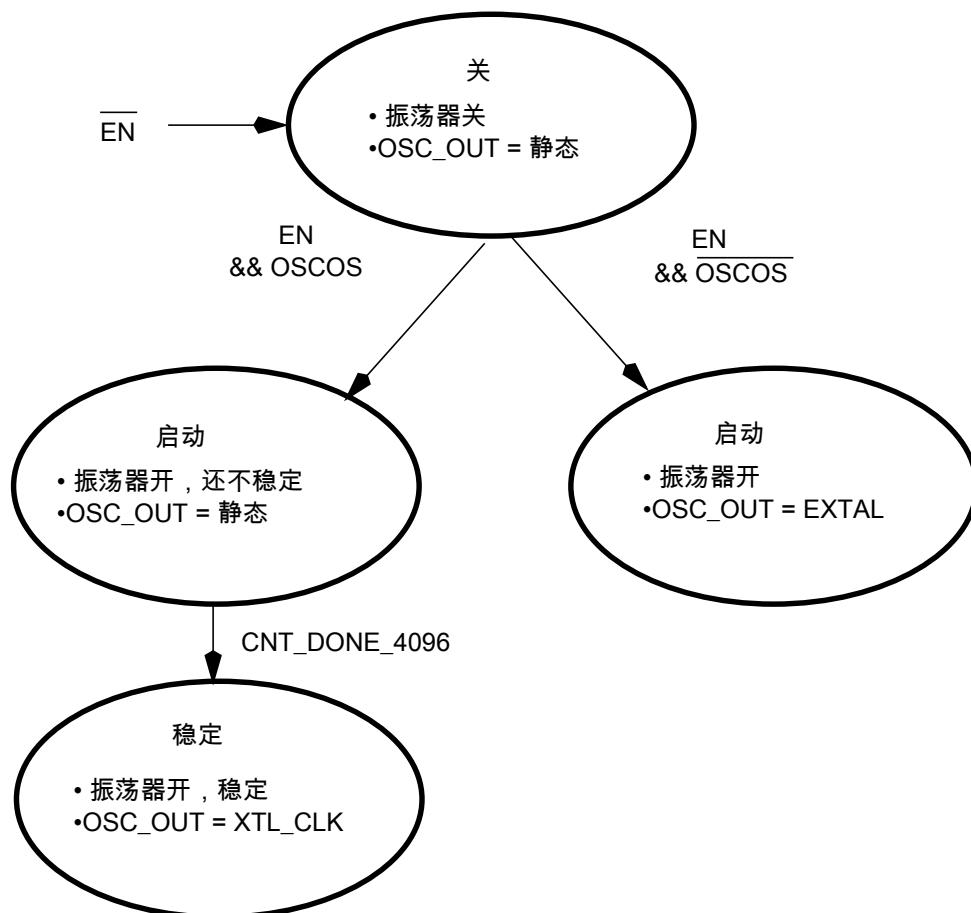
**OSC\_CR 字段描述 (继续)**

字段	描述
0	振荡器初始化未完成。
1	振荡器初始化已完成。

## 21.6 功能说明

### 21.6.1 OSC 模块状态

OSC 模块有三种状态。状态图可参见图 21-5。本节说明各状态以及相互之间的转换。

**图 21-5. OSC 模块状态图**

EN 由 OSC\_CR[OSCEN]、停止、OSC\_CR[OSCSTEN]和外部请求(ICS\_OSC\_EN)所确定。详情可参见下表。

表 21-3. EN 状态

EN	ICS_OSC_EN	OSC_CR[OSCEN]	OSC_CR[OSCSTEN]	停止
1	1	-	-	-
1	0	1	-	0
1	0	1	1	1
0	0	1	0	1
0	0	0	-	-

### 21.6.1.1 关

只要 EN 信号被否定，便进入关状态。进入该状态后，XTL\_CLK 和 OSC\_OUT 为静态。在此状态下，EXTAL 和 XTAL 引脚还会与其他所有振荡器电路解除耦合。OSC 模块电路配置为消耗最小的电流。

### 21.6.1.2 振荡器启动

只要振荡器首次使能，振荡器便进入启动状态（EN 转换为高电平），并且 OSC\_CR[OSCOS] 为高电平。在此状态下，OSC 模块启用并开始振荡，但是还不稳定。当振荡幅度变得足够大，可以通过输入缓冲区时，XTL\_CLK 开始为计数器提供时钟源。当计数器看到 XTL\_CLK 的 4096 个周期时，振荡器可视作稳定，并且 XTL\_CLK 传输至输出时钟 OSC\_OUT。

### 21.6.1.3 振荡器稳定

在满足了振荡器使能（EN 高电平），OSC\_CR[OSCOS] 为高电平，并且计数器能看到 4096 个 XTL\_CLK 周期（CNT\_DONE\_4096 为高电平），振荡器便进入稳定状态。在此状态下，OSC 模块在 OSC\_OUT 上产生稳定的输出时钟。其频率由所使用的外部组件确定。

### 21.6.1.4 外部时钟模式

使能振荡器（EN 高电平）且 OSC\_CR[OSCOS] 为低电平时，进入外部时钟状态。此状态下，OSC 模块设置为将来自 EXTAL 的时钟缓冲（有迟滞）至 OSC\_OUT。其频率由提供的外部时钟确定。

## 21.6.2 OSC 模块模式

此振荡器是 Pierce 类型的振荡器，支持外部晶振或谐振器以下表中显示的频率范围运行。这些模式假定 EN = 1、OSC\_CR[OSCOS] = 1。

表 21-4. 振荡器模式

范围	HGO	模式	频率范围
0	1	低频率、高增益	$f_{lo}(min)$ 到最高 $f_{lo}(max)$
0	0	低频率、低功耗(VLP)	
1	1	高频率模式 1、高增益	$f_{hi}(min)$ 到最高 $f_{hi}(max)$
1	0	高频率模式 1、低功耗	

### 21.6.2.1 低频、高增益模式

在低频率、高增益模式下 (OSC\_CR[RANGE] = 0、OSC\_CR[HGO] = 1)，振荡器使用简单的逆变器类放大器。增益设置为可获得全摆幅振荡幅度。此模式下的振荡器输入缓冲区为单端式。它为电压滤波提供低通频率过滤和迟滞功能，并将输出转换为逻辑电平。

### 21.6.2.2 低频、低功耗模式

在低频率、低功耗模式下 (OSC\_CR[RANGE] = 0、OSC\_CR[HGO] = 0)，振荡器使用增益控制环路来最大程度地降低功耗。当振幅增大时，放大器电流减小。这种状况将持续，直到所需的幅度到达稳态。

此模式下的振荡器输入缓冲区为单端式。它为电压滤波提供低通频率过滤和迟滞功能，并将输出转换为逻辑电平。

在该模式下，放大器输入、增益控制输入和输入缓冲区输入均为容性耦合，以提升漏电容差（对 EXTAL 的直流电平不敏感）。

该模式下，集成了除谐振器本身以外的所有外部元器件，包括：负载电容和反馈电阻，可偏置 EXTAL。

### 21.6.2.3 高频、高增益模式

在高频率、高增益模式下 (OSC\_CR[RANGE] = 1、OSC\_CR[HGO] = 1)，振荡器使用简单的逆变器类放大器。增益设置为可获得全摆幅振荡幅度。

此模式下的振荡器输入缓冲区为单端式。它为电压滤波提供低通频率过滤和迟滞功能，并将输出转换为逻辑电平。

#### 21.6.2.4 高频、低功耗模式

在高频率、低功耗模式下 ( $\text{OSC\_CR[RANGE]} = 1$ 、 $\text{OSC\_CR[HGO]} = 0$ )，振荡器使用增益控制环路来最大程度地降低功耗。当振幅增大时，放大器电流减小。这种状况将持续，直到所需的幅度到达稳态。

此模式下的振荡器输入缓冲区为差分式。它为电压滤波提供低通频率过滤和迟滞功能，并将输出转换为逻辑电平。

#### 21.6.3 计数器寄存器

振荡器输出时钟( $\text{OSC\_OUT}$ )将被门控关闭，直到计数器检测到 4096 个输入时钟周期( $\text{XTL\_CLK}$ )。一旦 4096 个周期完成后，计数器会将  $\text{XTL\_CLK}$  发送到  $\text{OSC\_OUT}$  上。此计数定时溢出可用于确保输出时钟的稳定性。

#### 21.6.4 基准时钟的引脚要求

在振荡器模式下，OSC 模块需要同时使用 EXTAL 和 XTAL 引脚来生成输出时钟，但在外部时钟模式下，只需要使用 EXTAL 引脚。只要满足数据手册中列出的规格，便可将 EXTAL 和 XTAL 引脚用于 I/O 或测试时钟用途。



# 第 22 章 循环冗余校验(CRC)

## 22.1 简介

### 注

关于与具体芯片相关的本模块详细操作示例，请参见芯片配置信息。

循环冗余校验(CRC)模块生成 16/32 位 CRC 码以便进行误差检测。

CRC 模块提供多项的 16 位或 32 位 CRC 标准所需的可编程多项式、WAS 和其他参数。

对于 32 位数据，每次都计算该 16/32 位代码。

### 22.1.1 特性

CRC 模块特性包括：

- 硬件电路 CRC 产生器用到一个 16 位或者 32 位可编程移位寄存器
- 可编程初始化起始值和多项式
- 逐位或逐字节可选择转换输入数据或输出数据 (CRC 结果)。某些 CRC 标准要求提供该选项。以 8 位读取操作访问 CRC 数据寄存器时，无法执行逐字节转置操作。这种情况下，用户软件必须执行逐字节转置功能。
- 提供最终 CRC 结果反转选项
- 32 位 CPU 寄存器编程接口

### 22.1.2 结构框图

下图为 CRC 结构框图。

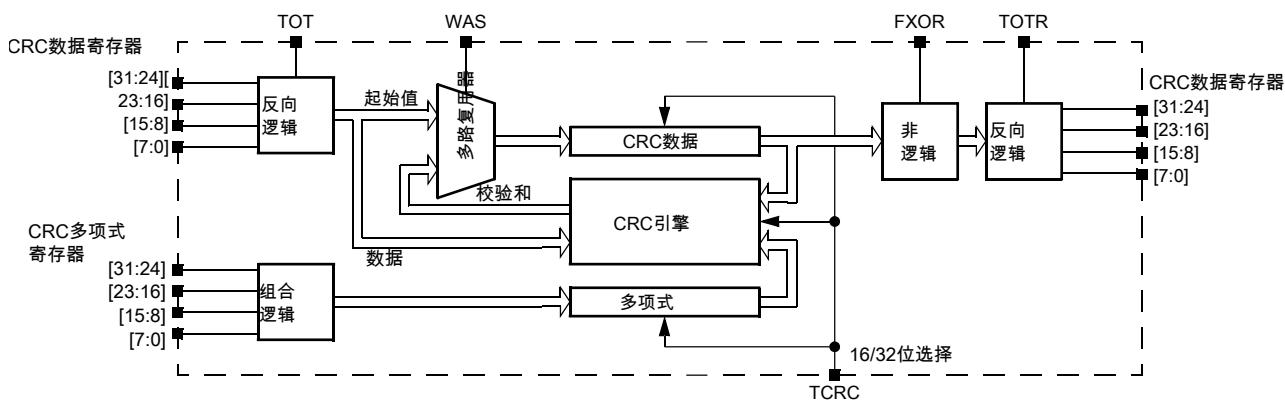


图 22-1. 可编程循环冗余校验(CRC)结构框图

## 22.1.3 工作模式

不同 MCU 模式都会影响 CRC 模块的功能。

### 22.1.3.1 Run 模式

这是基本工作模式。

### 22.1.3.2 低功耗模式 (Wait 或 Stop 模式)

MCU 进入低功耗模式后禁用模块时钟，任何进行中的 CRC 计算都会停止。CRC 计算在时钟使能后或通过系统复位退出低功耗模式后恢复。该模块的时钟选通取决于 MCU。

## 22.2 存储器映像和寄存器说明

### CRC 存储器映射

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4003_2000	CRC 数据寄存器 (CRC_DATA)	32	R/W	FFFF_FFFFh	<a href="#">22.2.1/299</a>
4003_2004	CRC 多项式寄存器 (CRC_GPOLY)	32	R/W	0000_1021h	<a href="#">22.2.2/300</a>
4003_2008	CRC 控制寄存器 (CRC_CTRL)	32	R/W	0000_0000h	<a href="#">22.2.3/300</a>

## 22.2.1 CRC 数据寄存器 (CRC\_DATA)

CRC 数据寄存器包含起始值、数据及校验和。如果设置了 CTRL[WAS]，则对数据寄存器执行的任何写操作都视为起始值。如果 CTRL[WAS]清零，则对数据寄存器进行的任何写操作都被视为用于一般 CRC 计算的数据。

在 16 位 CRC 模式下，不使用 HU 字段和 HL 字段来设定起始值，对这些字段进行读操作会返回不确定的值。在 32 位 CRC 模式中，所有字段均用于设定起始值。

进行数据数值编程时，如果所有字节都是连续的，那么可一次写入 8 位、16 位或 32 位的数值；首先写入的是 MSB 数据数值。

写入所有数据数值后，可从该数据寄存器中读取 CRC 结果。在 16 位 CRC 模式下，LU 字段和 LL 字段提供 CRC 结果。在 32 位 CRC 模式下，所有字段均包含此结果。随时对此寄存器执行读访问，以返回中间的 CRC 值，假设已配置 CRC 模块。

地址: 4003\_2000h 基准 + 0h 偏移 = 4003\_2000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
R																																				
W	HU								HL								LU								LL											
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					

### CRC\_DATA 字段描述

字段	描述
31–24 HU	CRC 高字段高位字节  在 16 位 CRC 模式 ( CTRL[TCRC] 为 0 ) 下，该字段并不用于起始值的编程。在 32 位 CRC 模式 ( CTRL[TCRC] 为 1 ) 下，当 CTRL[WAS] 为 1 时，写入该字段的值是起始值的一部分。当 CTRL[WAS] 为 0 时，写入该字段的数据用于在 16 位 CRC 模式和 32 位 CRC 模式下生成 CRC 校验和。
23–16 HL	CRC 高字段低位字节  在 16 位 CRC 模式 ( CTRL[TCRC] 为 0 ) 下，该字段并不用于设定起始值。在 32 位 CRC 模式 ( CTRL[TCRC] 为 1 ) 下，当 CTRL[WAS] 为 1 时，写入该字段的值是起始值的一部分。当 CTRL[WAS] 为 0 时，写入该字段的数据用于在 16 位 CRC 模式和 32 位 CRC 模式下生成 CRC 校验和。
15–8 LU	CRC 低字段高位字节  当 CTRL[WAS] 为 1 时，写入该字段的数值是起始值的一部分。当 CTRL[WAS] 为 0 时，写入该字段的数据用于生成 CRC 校验和。
LL	CRC 低字段低位字节  当 CTRL[WAS] 为 1 时，写入该字段的数值是起始值的一部分。当 CTRL[WAS] 为 0 时，写入该字段的数据用于生成 CRC 校验和。

## 22.2.2 CRC 多项式寄存器 (CRC\_GPOLY)

该寄存器含有 CRC 计算所需的多项式值。HIGH 字段含有 CRC 多项式的高 16 位，仅在 32 位 CRC 模式下使用。在 16 位 CRC 模式下会忽略对 HIGH 字段的写操作。LOW 字段包含在 16 位 CRC 模式和 32 位 CRC 模式下都使用的 CRC 多项式的低 16 位。

地址: 4003\_2000h 基准 + 4h 偏移 = 4003\_2004h

	位 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R	HIGH	LOW
W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1

复位 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

### CRC\_GPOLY 字段描述

字段	描述
31–16 HIGH	多项式高 16 位 32 位 CRC 模式下可读写 ( CTRL[TCRC] 为 1 )。该字段在 16 位 CRC 模式下不可写 ( CTRL[TCRC] 为 0 )。
LOW	多项式低 16 位 在 32 位 CRC 模式和 16 位 CRC 模式下都可读写。

## 22.2.3 CRC 控制寄存器 (CRC\_CTRL)

该寄存器控制 CRC 模块的配置和工作。开始进行新的 CRC 计算前，相应位必须置位。初始化新的 CRC 计算的方法是：使 CTRL[WAS] 的电平变为有效，然后将起始值写入 CRC 数据寄存器。

地址: 4003\_2000h 基准 + 8h 偏移 = 4003\_2008h

	位 31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16					
R	TOT	TOTR	0	FXOR	WAS	TCRC	0
W	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0					
复位	0 0 0 0 0 0 0						
位	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0					
R	0						
W	0 0 0 0 0 0 0						
复位	0 0 0 0 0 0 0	0 0 0 0 0 0 0					

**CRC\_CTRL 字段描述**

字段	描述
31–30 TOT	写入的转置类型  定义写入 CRC 数据寄存器的数据的转置配置。有关可用的转置选项，请参见转置特性说明。  00 无转置。 01 字节中的位转置；字节不转置。 10 字节中的位和字节均转置。 11 仅字节转置；字节中的位不转置。
29–28 TOTR	读取的转置类型  识别从 CRC 数据寄存器读取的数值的转置配置。有关可用的转置选项，请参见转置特性说明。  00 无转置。 01 字节中的位转置；字节不转置。 10 字节中的位和字节均转置。 11 仅字节转置；字节中的位不转置。
27 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
26 FXOR	CRC 数据寄存器的补充读取  某些 CRC 协议要求最终校验和与 0xFFFFFFFF 或 0xFFFF 进行异或运算。使该位的电平变为有效可使能已读取数据的即时补充。  0 读取时不执行异或运算。 1 反转或补充 CRC 数据寄存器的读取值。
25 WAS	作为起始值写入 CRC 数据寄存器  电平变为有效后，写入 CRC 数据寄存器的值被视为起始值。电平变为无效后，写入 CRC 数据寄存器的值用作 CRC 计算中的数据。  0 写入 CRC 数据寄存器的是数据值。 1 写入 CRC 数据寄存器的是起始值。
24 TCRC	CRC 协议宽度。  0 16 位 CRC 协议。 1 32 位 CRC 协议。
保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

## 22.3 功能说明

### 22.3.1 CRC 初始化/重新初始化

要使能 CRC 计算，用户必须在应用的寄存器内对 CRC\_CTRL[WAS]、CRC\_GPOLY、转置所必需的参数以及 CRC 结果反转进行编程。使 CRC\_CTRL[WAS]的电平变为有效可实现将起始值编入 CRC\_DATA 寄存器。

完成 CRC 计算后，使 CRC\_CTRL[WAS]的电平再次变为有效并进行起始值的编程，且无论其值是全新数值还是之前使用过的起始值，都重新初始化 CRC 模块以便进行新的 CRC 计算。所有其他参数都必须设置在进行起始值以及后续数据值的程序之前。

## 22.3.2 CRC 计算

在 16 位 CRC 模式和 32 位 CRC 模式下，如果所有字节都是连续的，可进行 8 位、16 位或 32 位数据值的一次性编程。非连续字节可能导致 CRC 计算错误。

### 22.3.2.1 16 位 CRC

如需计算 16 位 CRC：

1. 清零 CRC\_CTRL[TCRC]以使能 16 位 CRC 模式。
2. 按 CRC 计算要求对转置进行编程，并在 CTRL 寄存器中补全选项位。更多详情，请参见[转置特性](#) 和 [CRC 结果补码](#)。
3. 将 16 位多项式写入 CRC\_GPOLY[LOW]字段。CRC\_GPOLY[HIGH]字段在 16 位 CRC 模式下不可用。
4. 置位 CRC\_CTRL[WAS]以进行起始值的编程。
5. 将 16 位起始值写入 CRC\_DATA[LU:LL]。CRC\_DATA[HU:HL]不使用。
6. 清零 CRC\_CTRL[WAS]以开始写入数据值。
7. 将数据值写入 CRC\_DATA[HU:HL:LU:LL]。每次执行数据值写入操作便计算 CRC，并将 CRC 中间值结果存回至 CRC\_DATA[LU:LL]。
8. 完成所有数值的写入操作后，从 CRC\_DATA[LU:LL]读取最终的 CRC 结果。

转置和补充操作在读取或写入数值的同时执行。更多详情，请参见[转置特性](#) 和 [CRC 结果补码](#)。

### 22.3.2.2 32 位 CRC

如需计算 32 位 CRC：

1. 置位 CRC\_CTRL[TCRC]以使能 32 位 CRC 模式。
2. 按 CRC 计算要求对转置进行编程，并在 CTRL 寄存器中补全选项位。更多详情，请参见[转置特性](#) 和 [CRC 结果补码](#)。
3. 将 32 位多项式写入 CRC\_GPOLY[HIGH:LOW]。
4. 置位 CRC\_CTRL[WAS]以进行起始值的编程。
5. 将 32 位起始值写入 CRC\_DATA[HU:HL:LU:LL]。
6. 清零 CRC\_CTRL[WAS]以开始写入数据值。

7. 将数据值写入 CRC\_DATA[HU:HL:LU:LL]。每次执行数据值写入操作便计算 CRC，并将 CRC 中间值结果存回至 CRC\_DATA[HU:HL:LU:LL]。
8. 完成所有数值的写入操作后，从 CRC\_DATA[HU:HL:LU:LL]读取最终的 CRC 结果。CRC 计算逐字节执行，且需要两个时钟周期完成一次 CRC 计算。

转置和补充操作在读取或写入数值的同时执行。更多详情，请参见[转置特性](#) 和 [CRC 结果补码](#)。

### 22.3.3 转置特性

默认情况下，转置特性未使能。然而，某些 CRC 标准要求对输入数据和/或最终校验和进行转置。用户软件可根据 CRC 标准需要选择单独配置每个转置操作。数据在读写的同时进行转置。

某些协议计算 CRC 时对数据流采用低字节序格式。在这种情况下，转置特性非常有用，可以翻转位。该转置选项是 CRC 模块支持的功能类型之一。

#### 22.3.3.1 转置类型

CRC 模块提供可翻转位和/或字节的多种转置功能类型，以便根据采用的 CRC 计算方法分别使用 CTRL[TOT]或 CTRL[TOTR]字段写入输入数据和读取 CRC 结果。

下列转置功能类型可用于对 CRC 数据寄存器进行读写操作：

1. CTRL[TOT]或 CTRL[TOTR]为 00。

不发生转置。

2. CTRL[TOT]或 CTRL[TOTR]为 01。

字节中的位转置，而字节不转置。

`reg[31:0]`变为`{reg[24:31], reg[16:23], reg[8:15], reg[0:7]}`

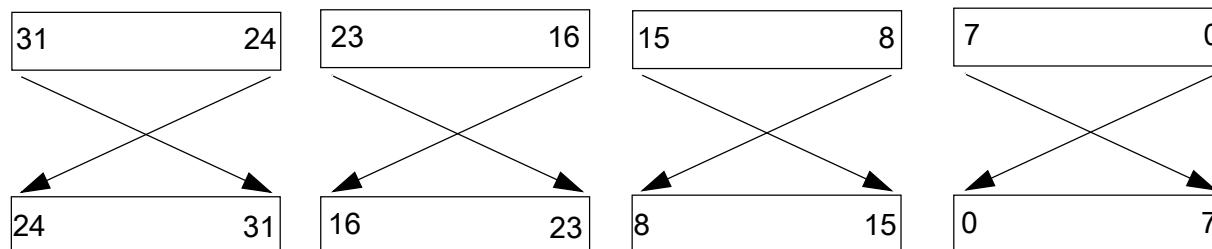


图 22-2. 转置类型 01

3. CTRL[TOT]或 CTRL[TOTR]为 10。

字节中的位和字节均转置。

reg[31:0]变为 = {reg[0:7], reg[8:15], reg[16:23], reg[24:31]}

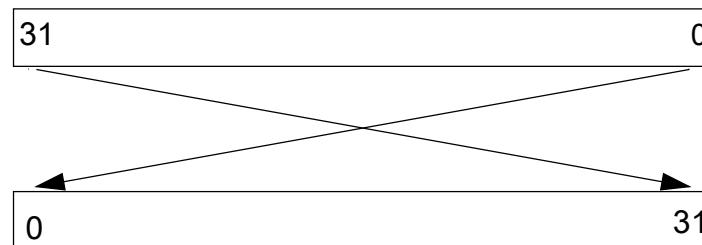


图 22-3. 转置类型 10

4. CTRL[TOT]或 CTRL[TOTR]为 11。

字节转置，但位不转置。

reg[31:0]变为{reg[7:0], reg[15:8], reg[23:16], reg[31:24]}

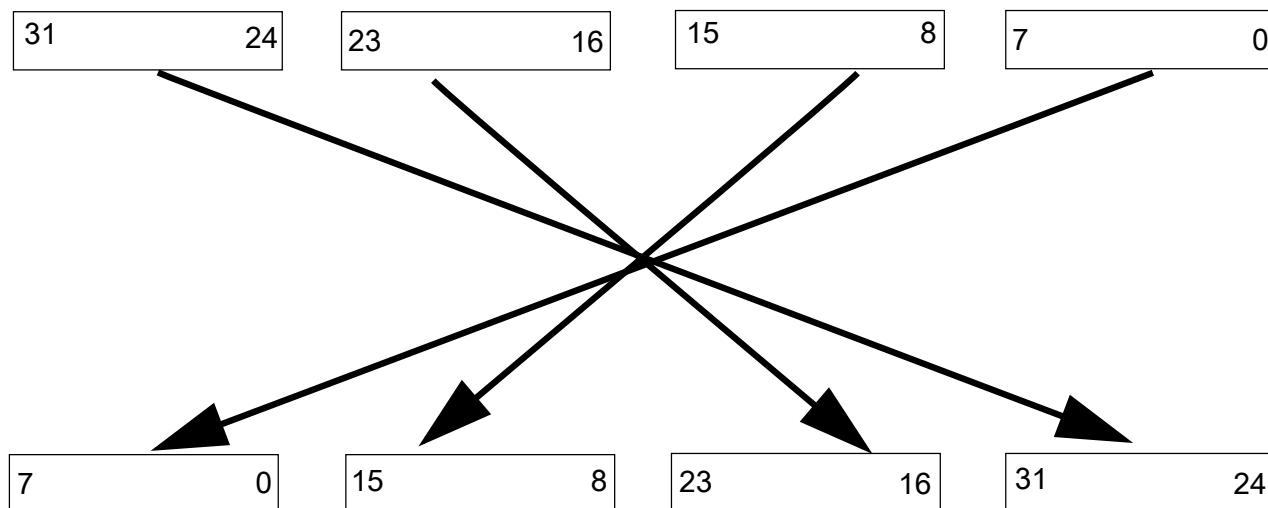


图 22-4. 转置类型 11

### 注

为了对 CRC 数据寄存器进行 8 位或 16 位的写访问, 对不使用的字节该数据转置为零 (将 32 位作为一个整体), 但对于有效字节仅计算 CRC。读取 CRC 数据寄存器中的 16 位 CRC 结果并使用转置选项 10 和 11 时, 转置后的结果数值位于 CRC[HU:HL]字段。读取 16 位 CRC 结果时, 用户软件必须将该情形考虑在内, 因此优先读取 32 位。

### 22.3.4 CRC 结果补码

CTRL[FXOR]置位后, 对校验和进行补码。CRC 结果补码功能可在每次对 CRC 数据寄存器进行读操作时输出存储在 CRC 数据寄存器中的校验和数值的补码。CTRL[FXOR]清零后, 对 CRC 数据寄存器进行读操作会访问原始校验和数值。

# 第 23 章 外部中断(IRQ)

## 23.1 简介

外部中断(IRQ)模块提供一路可屏蔽中断输入。

## 23.2 特性

IRQ 模块特性包括：

- IRQ 中断控制位
- 可编程中断由边沿触发或边沿和电平触发
- 自动中断应答
- 内部上拉电阻

施加在外部中断请求(IRQ)引脚上的低电平可锁存 CPU 中断请求。下图显示的是 IRQ 模块结构：

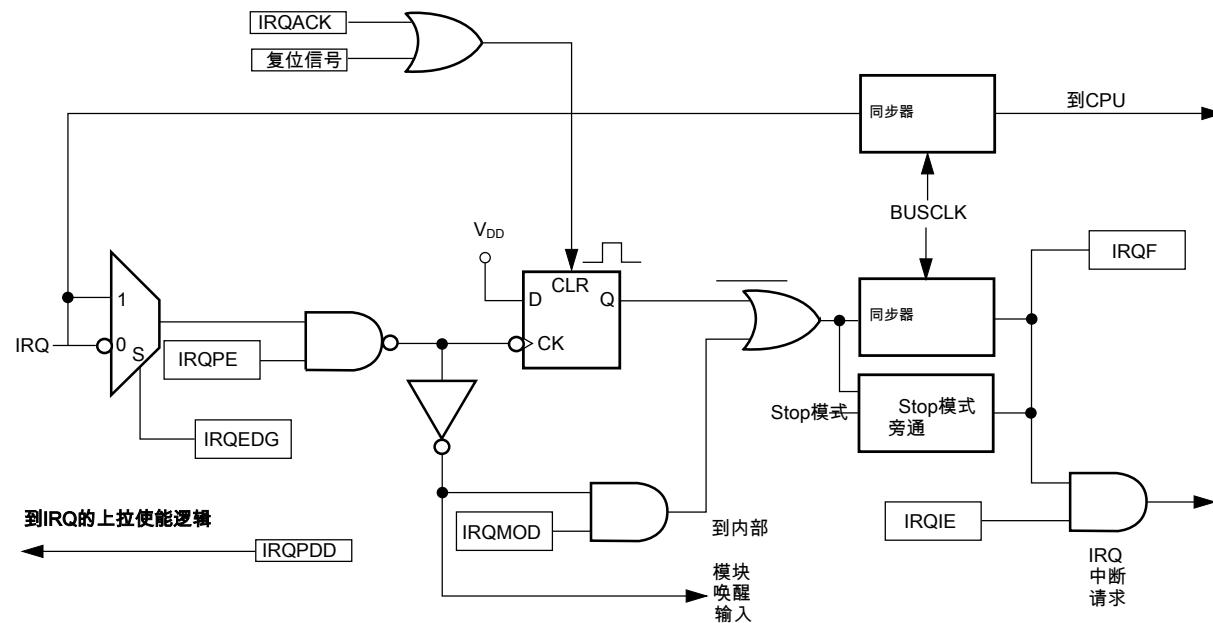


图 23-1. IRQ 模块结构框图

外部中断由 IRQ\_SC 状态和控制寄存器管理。使能 IRQ 功能后，同步逻辑监测引脚，以便确定是仅边沿事件还是边沿和电平事件。当 MCU 处于 Stop 模式且系统时钟被关断时，就会使用单独的异步路径，从而 IRQ（若使能）能唤醒 MCU。

### 23.2.1 引脚配置选项

IRQ 引脚使能(IRQ\_SC[IRQPE])控制字段必须置 1，IRQ 引脚才能用作 IRQ 输入。用户可以选择有效的边沿或电平的极性(IRQEDG)，引脚是只检测边沿还是检测边沿和电平(IRQMOD)，或者事件是触发中断还是仅置位 IRQ\_SC[IRQF]标志，再通过软件轮询。

使能时，IRQ 引脚默认使用内部上拉电阻(IRQ\_SC[IRQPDD] = 0)。若用户使用外部上拉或下拉电阻，则可将 IRQ\_SC[IRQPDD]置 1 以关断内部设备。

在 IRQ 引脚配置为 IRQ 输入时，BIH 和 BIL 指令可用于检测该引脚上的电平。

#### 注

该引脚不含钳位至  $V_{DD}$  的钳位二极管，且不得输入超过  $V_{DD}$ 。内部上拉的 IRQ 引脚上测得的电压可能低至  $V_{DD} - 0.7$  V。连接该引脚的内部栅极一路上拉至  $V_{DD}$ 。

使能 IRQ 引脚以供使用后，IRQ\_SC[IRQF]将置位，并且在使能中断前必须清零。针对 3V 系统，当管脚配置为下降沿或电平有效时，在清标志位和使能中断之前须等待几个机器周期。

## 23.2.2 边沿和电平有效

IRQ\_SC[IRQMOD]控制字段可以配置检测逻辑，以检测边沿事件和引脚电平。在该检测模式下，如果 IRQ 引脚从无效电平变为有效电平，则在检测到边沿时 IRQ\_SC[IRQF]状态标志置位；但只要 IRQ 引脚保持在有效电平，该标志就会继续置位并且无法清零。

## 23.3 外部中断引脚请求寄存器

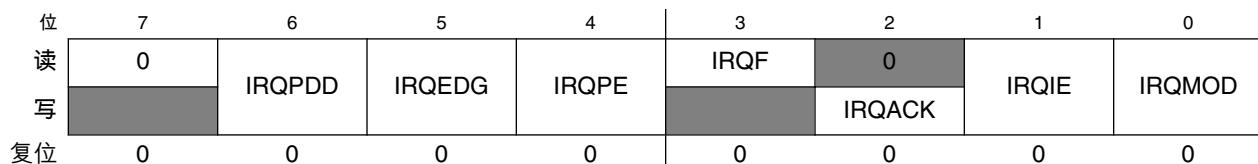
**IRQ 存储器映射**

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4003_1000	外部中断引脚请求状态和控制寄存器 (IRQ_SC)	8	R/W	00h	23.3.1/307

### 23.3.1 外部中断引脚请求状态和控制寄存器 (IRQ\_SC)

该直接页面寄存器包括状态位和控制位，用于配置 IRQ 功能、报告状态和应答 IRQ 事件。

地址: 4003\_1000h 基准 + 0h 偏移 = 4003\_1000h



**IRQ\_SC 字段描述**

字段	描述
7 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
6 IRQPDD	IRQ 内部上拉电阻禁用  当 IRQ 引脚使能时(IRQPE = 1)，该读/写控制位用于禁用内部上拉电阻，以配合外部器件。  0 若 IRQPE = 1，则 IRQ 内部上拉电阻使能。 1 若 IRQPE = 1，则 IRQ 内部上拉电阻禁用。
5 IRQEDG	IRQ 边沿选择  该读/写控制字段用于选择使 IRQF 置位的 IRQ 引脚上的边沿或电平的极性。IRQMOD 控制字段确定 IRQ 引脚是对边沿和电平都敏感还是只对边沿敏感。当 IRQ 引脚使能用作 IRQ 输入并配置为检测上升沿时，可选上拉电阻是禁用的。

下一页继续介绍此表...

**IRQ\_SC 字段描述 (继续)**

字段	描述
	0 IRQ 对下降沿或下降沿/低电平敏感。 1 IRQ 对上升沿或上升沿/高电平敏感。
4 IRQPE	<b>IRQ 引脚使能</b> 该读/写控制字段使能 IRQ 引脚功能。该字段置位后，IRQ 引脚可用作中断请求。 0 IRQ 引脚功能禁用。 1 IRQ 引脚功能使能。
3 IRQF	<b>IRQ 标志</b> 该只读状态字段指示是否发生了中断请求事件。 0 无 IRQ 请求 1 检测到 IRQ 事件。
2 IRQACK	<b>IRQ 应答</b> 该只写字段用于应答中断请求事件 ( 写入 1 可清零 IRQF )。写入 0 无效。读取操作始终返回 0。如果选择边沿和电平敏感(IRQMOD = 1) , 当 IRQ 引脚保持在有效电平时无法清零 IRQF。
1 IRQIE	<b>IRQ 中断使能</b> 该读/写控制字段确定 IRQ 事件是否生成中断请求。 0 IRQF 置位时禁止发送中断请求 ( 使用轮询 ) 1 只要 IRQF = 1 就发送中断请求。
0 IRQMOD	<b>IRQ 检测模式</b> 该读/写控制字段选择仅边沿检测或边沿和电平敏感。 0 只检测下降沿/上升沿。 1 检测下降沿/上升沿和低电平/高电平。

# 第 24 章

## 模数转换器(ADC)

### 24.1 简介

12 位模数转换器 (ADC) 是一种逐次逼近型 ADC，应用于集成化的 MCU 片上系统。

#### 24.1.1 特性

ADC 模块特性包括：

- 采用 8 位、10 位或 12 位分辨率的线性逐次逼近算法
- 最多 16 个外部模拟输入、外部引脚输入以及 5 个内部模拟输入，包括内部带隙基准、温度传感器和基准电压
- 8 位、10 位或 12 位的右对齐无符号格式输出
- 单次或连续转换（单次转换后自动返回到空闲状态）
- 结果 FIFO 深度可选，最多可保存 8 个转换结果
- 可配置采样时间和转换速度/功耗
- 转换完成标志和中断
- 可从最多 4 个来源中选择输入时钟
- 可在 Wait 或 Stop 模式下工作，降低转换时的噪声
- 提供异步时钟源，降低转换时的噪声
- 可选的异步硬件转换触发源
- 自动与设定值进行比较（小于、大于或等于），根据结果产生中断

## 24.1.2 结构框图

本图是 ADC 模块的结构框图。

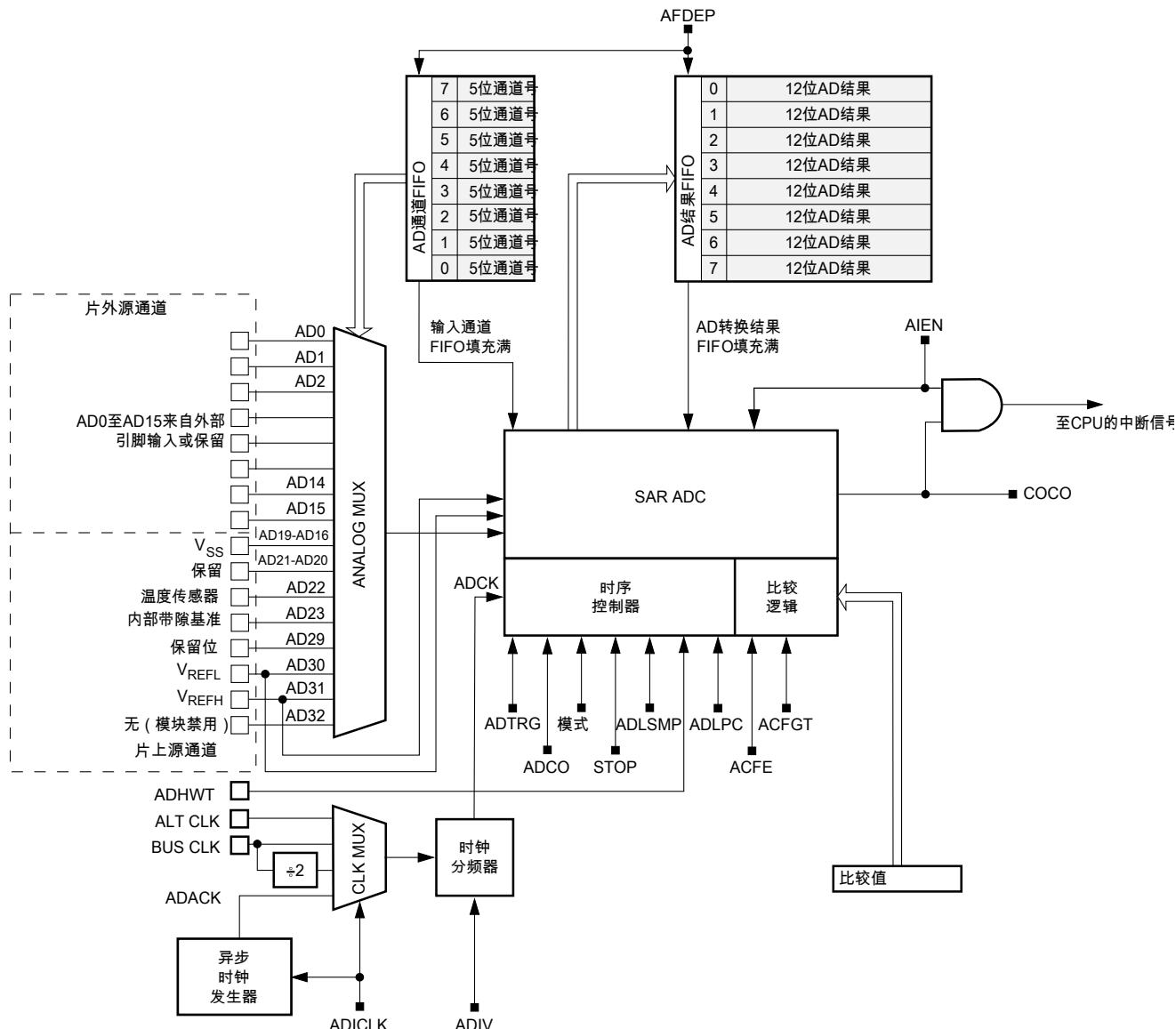


图 24-1. ADC 方框图

## 24.2 外部信号说明

ADC 模块支持多达 24 个单独的模拟输入。它还要求四个电源/基准/接地连接。

表 24-1. 信号属性

名称	功能
AD23–AD0	模拟通道输入
$V_{REFH}$	高基准电压
$V_{REFL}$	低基准电压
$V_{DDA}$	模拟电源
$V_{SSA}$	模拟接地

## 24.2.1 模拟电源( $V_{DDA}$ )

ADC 模拟部分将  $V_{DDA}$  用作其电源连接。在某些封装中， $V_{DDA}$  内部连接到  $V_{DD}$ 。如果可以外部连接，则将  $V_{DDA}$  引脚连接到与  $V_{DD}$  相同的电势。可能必须进行外部滤波确保  $V_{DDA}$  干净以便获得良好的结果。

## 24.2.2 模拟接地( $V_{SSA}$ )

ADC 模拟部分将  $V_{SSA}$  用作其接地连接。在某些封装中， $V_{SSA}$  内部连接到  $V_{SS}$ 。如果可以外部连接，则将  $V_{SSA}$  引脚连接到与  $V_{SS}$  相同的电势。

## 24.2.3 高基准电压 ( $V_{REFH}$ )

$V_{REFH}$  是转换器的高基准电压。在某些封装中， $V_{REFH}$  内部连接到  $V_{DDA}$ 。如果可从外部连接，可将  $V_{REFH}$  连接到与  $V_{DDA}$  相同的电势，或由介于数据手册中指定的最小  $V_{DDA}$  与  $V_{DDA}$  电势之间的外部电压源驱动 ( $V_{REFH}$  不能超过  $V_{DDA}$ )。

## 24.2.4 低基准电压 ( $V_{REFL}$ )

$V_{REFL}$  是转换器的低基准电压。在有些封装中， $V_{REFL}$  在内部与  $V_{SSA}$  相连。如果也可从外部连接，则将  $V_{REFL}$  引脚连接到与  $V_{SSA}$  相同的电压引脚上。

## 24.2.5 模拟通道输入(ADx)

ADC 模块支持多达 24 个单独的模拟输入。通过 ADCH 通道选择位选择某个输入进行转换。

## 24.3 ADC 控制寄存器

### ADC 存储器映射

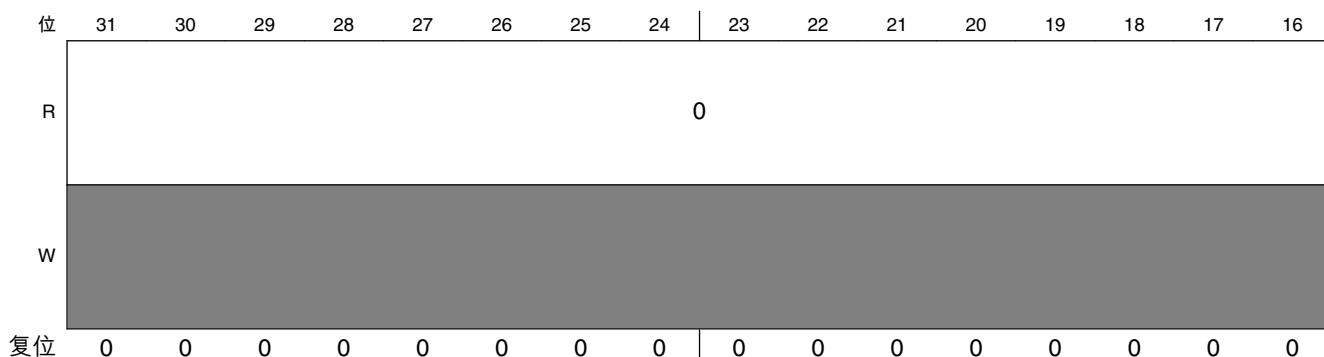
绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4003_B000	状态和控制寄存器 1 (ADC_SC1)	32	R/W	0000_001Fh	24.3.1/312
4003_B004	状态和控制寄存器 2 (ADC_SC2)	32	R/W	0000_0008h	24.3.2/314
4003_B008	状态和控制寄存器 3 (ADC_SC3)	32	R/W	0000_0000h	24.3.3/316
4003_B00C	状态和控制寄存器 4 (ADC_SC4)	32	R/W	0000_0000h	24.3.4/317
4003_B010	转换结果寄存器 (ADC_R)	32	R	0000_0000h	24.3.5/318
4003_B014	比较值寄存器 (ADC_CV)	32	R/W	0000_0000h	24.3.6/319
4003_B018	引脚控制 1 寄存器 (ADC_APCTL1)	32	R/W	0000_0000h	24.3.7/319
4003_B01C	状态和控制寄存器 5 (ADC_SC5)	32	R/W	0000_0000h	24.3.8/320

### 24.3.1 状态和控制寄存器 1 (ADC\_SC1)

本节介绍 ADC 状态和控制寄存器(ADC\_SC)的功能。对 ADC\_SC1 进行写操作会中止当前转换并在 ADCH 位非全 1 时启动新的转换。

FIFO 使能时，通过 ADCH 对模拟输入通道 FIFO 进行写操作。FIFO 中的模拟输入通道队列必须通过 ADCH 连续写入。生成的 FIFO 顺序与写入的模拟输入通道顺序相同。当输入通道 FIFO 中填充的内容达到 ADC\_SC4[AFDEP]指示的深度时，ADC 开始转换。在输入通道 FIFO 有效时将 0x1F 写入这些位就会使 FIFO 复位并停止转换。

地址: 4003\_B000h 基准 + 0h 偏移 = 4003\_B000h



位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R								0	COCO							
W								0		AIEN		ADCO				ADCH
复位	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

**ADC\_SC1 字段描述**

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 COCO	转换完成标志  转换完成标志。COCO 标志是一个只读位，当比较功能禁用时(ADC_SC2[ACFE] = 0)，COCO 会在每次转换完成时置位。如果比较功能使能(ADC_SC2[ACFE] = 1)，则仅当比较结果为真时，COCO 标志才会在转换完成时置位。当使能 FIFO 功能时(ADC_SC4[AFDEP] > 0)，COCO 标志在 FIFO 转换全部完成时置位。对 ADC_SC1 进行写操作或对 ADC_R 进行读操作时，该位清零。  0 转换未完成。 1 转换已完成。
6 AIEN	中断使能  AIEN 使能转换完成中断。如果在 COCO 置位的同时 AIEN 为高电平，那么会产生中断。  0 转换完成中断禁用。 1 转换完成中断使能。
5 ADCO	连续转换使能  ADCO 使能连续转换。  0 选择软件触发操作时，对 ADC_SC1 进行写操作后启动一次转换；选择硬件触发操作时，ADHWT 的电平变为有效值后启动一次转换。使能 FIFO 功能时(AFDEP > 0)，如果 ADC_SC2[ADTRG]=0 或 ADC_SC2[ADTRG]=1 且 ADC_SC4[HTRGME]=1，一次触发操作会触发一组转换。 1 选择软件触发操作时，对 ADC_SC1 进行写操作后启动连续转换。选择硬件触发操作时，由 ADHWT 事件启动连续转换。FIFO 功能使能时(AFDEP > 0)，一次触发操作会循环触发一组转换。
ADCH	输入通道选择  ADCH 位形成一个 5 位字段，用于选择一个输入通道。  00000-01111 AD0-AD15 10000-10011 V <sub>SS</sub> 10100-10101 保留位 10110 温度传感器

下一页继续介绍此表...

## ADC\_SC1 字段描述 (继续)

字段	描述
	10111 带隙基准 11000-11100 保留位 11101 $V_{REFH}$ 11110 $V_{REFL}$ 11111 禁用 ADC 模块  注：在 FIFO 模式下复位 FIFO。

## 24.3.2 状态和控制寄存器 2 (ADC\_SC2)

ADC\_SC2 寄存器用于控制 ADC 模块的比较功能、转换触发以及转换进行标志位。

地址: 4003\_B000h 基准 + 4h 偏移 = 4003\_B004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									0							
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									ADACT	ADTRG	ACFE	ACFGT	FEMPTY	FFULL		REFSEL
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

## ADC\_SC2 字段描述

字段	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

下一页继续介绍此表...

## ADC\_SC2 字段描述 (继续)

字段	描述
7 ADACT	<p>转换进行标志位</p> <p>指示转换正在进行。ADACT 在转换启动时置位，在转换完成或中止时清零。</p> <p>0 转换未在进行。 1 转换正在进行。</p>
6 ADTRG	<p>转换触发类型选择</p> <p>选择用于启动转换的触发类。有两类触发可选择：软件触发和硬件触发。选择软件触发时，对 ADC_SC1 进行写操作后启动转换。选择硬件触发时，ADHWT 输入的电平变为有效值后启动转换。</p> <p>0 选择软件触发。 1 选择硬件触发。</p>
5 ACFE	<p>比较功能使能</p> <p>使能比较功能。</p> <p>0 比较功能禁用。 1 比较功能使能。</p>
4 ACFGT	<p>大于等于比较功能使能</p> <p>将比较功能的触发条件配置为当被检测输入的转换结果大于或等于比较值时。默认的触发条件为当被检测输入的转换结果小于比较值时。</p> <p>0 输入小于比较电平时，比较触发。 1 输入大于或等于比较电平时，比较触发。</p>
3 FEMPTY	<p>结果 FIFO 空标志</p> <p>0 指示 ADC 结果 FIFO 至少有一个有效新数据。 1 指示 ADC 结果 FIFO 没有有效新数据。</p>
2 FFULL	<p>结果 FIFO 满标志</p> <p>0 指示 ADC 结果 FIFO 未满，下一转换数据仍可存储在 FIFO 中。 1 指示 ADC 结果 FIFO 已满，如果不读取 FIFO，下一转换数据将覆盖旧数据。</p>
REFSEL	<p>基准电压源选择</p> <p>选择用于转换的基准电压源。</p> <p>00 默认基准电压引脚对(<math>V_{REFH}/V_{REFL}</math>)。 01 模拟供电引脚对(<math>V_{DDA}/V_{SSA}</math>)。 10 保留位。 11 保留位 - 选择默认基准电压(<math>V_{REFH}/V_{REFL}</math>)引脚对。</p>

### 24.3.3 状态和控制寄存器 3 (ADC\_SC3)

ADC\_SC3 用于选择工作模式、时钟源、时钟分频，并配置低功耗或长采样时间。

地址: 4003\_B000h 基准 + 8h 偏移 = 4003\_B008h

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	0																
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	0									ADLPC	ADIV	ADLSMP	MODE	ADICLK			
W										0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

ADC\_SC3 字段描述

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 ADLPC	低功耗配置  ADLPC 用于配置逐次逼近转换器的速度和功耗。在无需更高采样率时，这能优化功耗。  0 高速模式。 1 低功耗模式：减小功耗的同时会降低时钟频率所允许的最大值。
6–5 ADIV	时钟分频选择  ADIV 选择 ADC 用于生成内部时钟 ADCK 的分频比。  00 分频比 = 1，时钟速率 = 输入时钟。 01 分频比 = 2，时钟速率 = 输入时钟 ÷ 2。 10 分频比 = 3，时钟速率 = 输入时钟 ÷ 4。 11 分频比 = 4，时钟速率 = 输入时钟 ÷ 8。
4 ADLSMP	长采样时间配置  ADLSMP 选择长或短采样时间。调整采样周期可以使高阻抗输入的采样更精确或使低阻抗输入的转换速度更快。当连续转换使能且无需高转换速率时，使用长采样时间还能降低整体功耗。  0 短采样时间。 1 长采样时间。
3–2 MODE	转换模式选择  MODE 位用于选择 12 位、10 位或 8 位操作。  00 8 位转换(N = 8)

下一页继续介绍此表...

## ADC\_SC3 字段描述 (继续)

字段	描述
	01 10 位转换( $N = 10$ ) 10 12 位转换( $N = 12$ ) 11 保留
ADICLK	<b>输入时钟选择</b>  ADICLK 位选择用于生成内部时钟 ADCK 的输入时钟源。  00 总线时钟 01 总线时钟 2 分频 10 备用时钟(ALTCLK) 11 异步时钟(ADACK)

## 24.3.4 状态和控制寄存器 4 (ADC\_SC4)

该寄存器用于配置 ADC 模块的 FIFO 扫描模式、FIFO 比较功能和 FIFO 深度。

地址: 4003\_B000h 基准 + Ch 偏移 = 4003\_B00Ch

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									HTRGME	0	ASCANE	ACFSEL	0				
W										0	0	0	0	0	0	0	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

## ADC\_SC4 字段描述

字段	描述
31–9 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
8 HTRGME	硬件触发多次转换功能使能  该字段使能硬件触发多次转换。  0 一个硬件触发脉冲触发一次转换。 1 一个硬件触发脉冲触发 FIFO 模式下的多次转换。
7 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

下一页继续介绍此表...

## ADC\_SC4 字段描述 (继续)

字段	描述
6 ASCANE	FIFO 扫描模式使能  FIFO 在使能时总是使用第一个通道单元中的内容。该位置位且 FIFO 功能使能时，ADC 将重复使用该第一个 FIFO 通道作为转换通道，直至结果 FIFO 填满。在连续模式(ADCO = 1)下，当 COCO 置位时，ADC 将利用相同通道开始下一转换。  0 FIFO 扫描模式禁用。 1 FIFO 扫描模式使能。
5 ACFSEL	比较功能选择  当 FIFO 功能使能时( $AFDEP > 0$ )，将比较功能配置为对 FIFO 中所有转换的比较触发进行“或”或者与“运算”。当该字段清零时，ADC 将对所有比较触发作“或”运算，如果至少有一个比较触发有效，就会置位 COCO。当该字段置位时，ADC 将对所有比较触发作“与”运算，如果所有比较触发都有效，就会置位 COCO。  0 对所有比较触发作“或”运算。 1 对所有比较触发作“与”运算。
4–3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
AFDEP	FIFO 深度  用于使能 FIFO 功能和设置 FIFO 的深度。AFDEP 清零时，FIFO 禁用。AFDEP 设置为非零值时，FIFO 功能使能，深度由 AFDEP 位指示。FIFO 功能使能时，FIFO 模式必须访问 ADC_SC1[ADCH]和 ADC_R。当模拟输入通道 FIFO 写入的内容达到 AFDEP 位指示的深度时，ADC 启动转换。当 FIFO 中的一组转换完成且结果 FIFO 中的内容达到 AFDEP 位指示的深度时，COCO 位置位。  000 FIFO 禁用。 001 2 级 FIFO 使能。 010 3 级 FIFO 使能。 011 4 级 FIFO 使能。 100 5 级 FIFO 使能。 101 6 级 FIFO 使能。 110 7 级 FIFO 使能。 111 8 级 FIFO 使能。

## 24.3.5 转换结果寄存器 (ADC\_R)

在 12 位工作模式下，ADC\_R 包含 12 位转换的 12 位结果。

在 10 位模式下，ADC\_R 包含 10 位转换的 10 位结果。

在 8 位模式下，ADC\_R 包含 8 位转换的 8 位结果。

除非使能了自动比较且不满足比较条件，ADC\_R 会在每次转换时更新。

当 FIFO 使能时，结果 FIFO 通过 ADC\_R 读取。当在 AFDEP 指定的深度执行输入通道 FIFO 时，ADC 转换完成。AD 结果 FIFO 可以按照模拟输入通道 ADCH 设置的顺序通过 ADC\_R 连续读取。

如果 MODE 位改变，ADC\_R 中的所有数据都会变为无效。

地址: 4003\_B000h 基准 + 10h 偏移 = 4003\_B010h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	0															
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

#### ADC\_R 字段描述

字段	描述
31-12 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
ADR	转换结果

### 24.3.6 比较值寄存器 (ADC\_CV)

该寄存器保存比较值。12 位模式下的转换完成后，位 ADCV11:ADCV0 与 12 位结果进行比较。

地址: 4003\_B000h 基准 + 14h 偏移 = 4003\_B014h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	0															
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

#### ADC\_CV 字段描述

字段	描述
31-12 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
CV	转换结果[11:0]

### 24.3.7 引脚控制 1 寄存器 (ADC\_APCTL1)

该引脚控制寄存器禁用用作模拟输入的 MCU 引脚的 I/O 端口控制。APCTL1 用于控制与 ADC 模块的通道 0-31 相关的引脚。

地址: 4003\_B000h 基准 + 18h 偏移 = 4003\_B018h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	0															
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

## ADC\_APCTL1 字段描述

字段	描述
31–16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
ADPC	ADC 引脚控制  ADPCx 控制与通道 ADx 相关的引脚。  0 ADx 引脚 I/O 控制使能。 1 ADx 引脚 I/O 控制禁用。

## 24.3.8 状态和控制寄存器 5 (ADC\_SC5)

ADC\_SC5 选择硬件触发屏蔽。

地址: 4003\_B000h 基准 + 1Ch 偏移 = 4003\_B01Ch

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									0								
W																HTRGMASKE	HTRGMASKSEL
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

## ADC\_SC5 字段描述

字段	描述
31–2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
1 HTRGMASKE	硬件触发屏蔽使能  当 HTRGMASKSEL 为低电平时，该字段使能硬件触发屏蔽。  0 硬件触发屏蔽禁用。 1 使能硬件触发屏蔽，硬件触发无法触发 ADC 转换。
0 HTRGMASKSEL	硬件触发屏蔽模式选择  该字段选择硬件触发屏蔽模式。  0 用 HTRGMASKE 屏蔽硬件触发。 1 当数据 FIFO 非空时，自动屏蔽硬件触发。

**ADC\_SC5 字段描述 (继续)**

字段	描述
----	----

## 24.4 功能说明

ADC 模块在复位期间 ADC\_SC1[ADCH]位均为高电平时禁用。该模块在转换完成后以及另一个转换还未发起前处于空闲状态。空闲时，该模块处于其功耗最低状态。

ADC 可在任何软件可选的通道上进行模数转换。在 12 位模式下，逐次逼近算法可将选定的通道电压转换成 12 位数字结果。在 10 位模式下，逐次逼近算法可将选定的通道电压转换成 10 位数字结果。在 8 位模式下，逐次逼近算法可将选定的通道电压转换成 8 位数字结果。

转换完成后，结果置于数据寄存器 (ADC\_R) 中。在 10 位模式下，结果四舍五入为 10 位并置于数据寄存器 (ADC\_R) 中。在 8 位模式下，结果四舍五入为 8 位并置于数据寄存器 ADC\_R 中。转换完成标志(ADC\_SC1[COCO])随后置位并且在转换完成中断已使能(ADC\_SC1[AIEN] = 1)的情况下会生成一个中断。

ADC 模块能够自动将转换结果与其比较寄存器的内容作比较。比较功能在 ADC\_SC2[ACFE]位置位的情况下使能，而且能在任何转换模式和配置下工作。

### 24.4.1 时钟选择和分频控制

ADC 模块的时钟源可以从四个时钟源中选择。然后用一个可配置的值将该时钟源分频，生成转换器的输入时钟(ADCK)。利用 ADC\_SC3[ADICLK]位从下列时钟源选择时钟：

- 总线时钟 2 分频：总线时钟速率较高时，最多可以对总线时钟进行 16 分频。
- ALTCLK，也就是备用时钟 OSC\_OUT
- 异步时钟(ADACK)：该时钟从 ADC 模块内部的时钟源生成。选择该时钟作为时钟源时，当 MCU 处于 Wait 或 Stop 模式时，它仍然有效，转换仍可进行，而且噪声很低。

无论选择何种时钟，其频率都必须在 ADCK 的指定频率范围以内。如果可用时钟太慢，ADC 将不能以额定性能工作。如果可用时钟太快，必须将其分频到合适的频率。该分频器由 ADC\_SC3[ADIV]位指定，可执行 1、2、4 或 8 分频。

## 24.4.2 输入选择和引脚控制

引脚控制寄存器 (ADC\_APCTL1) 用来禁用用作模拟输入的引脚的 I/O 端口控制。当引脚控制寄存器位置位时，将强制使相关的 MCU 引脚处于以下状态：

- 输出缓冲强制进入高阻抗状态。
- 输入缓冲禁用。对于其输入缓冲被禁用的任何引脚，对 I/O 端口进行读操作会返回 0。
- 上拉禁用。

## 24.4.3 硬件触发

ADC 模块具有一个可选的异步硬件转换触发源 ADHWT，在 ADC\_SC2[ADTRG]位置位后会被启用。此来源不适用于所有 MCU。有关本 MCU 特有的 ADHWT 来源的信息，请参见相关模块的简介。

如果 ADHWT 存在有效的来源并且选择硬件触发(ADC\_SC2[ADTRG] = 1)，则会在 ADHWT 的上升沿发起转换。如果出现上升沿时转换正在进行中，则该上升沿会被忽略。在连续转换配置中，仅观察最开始启动连续转换的上升沿。硬件触发功能可在任何转换模式及配置下运行。

## 24.4.4 转换控制

转换可在 12 位模式、10 位模式或 8 位模式下进行，具体取决于 ADC\_SC3[MODE] 位。转换可由软件或硬件触发发起。此外，可通过配置 ADC 模块实现更多的功能，包括低功耗转换、长采样时间、连续转换以及自动将转换结果与软件确定的比较值作比较。

### 24.4.4.1 发起转换

以下情形时会发起转换：

- 选择软件触发时，对 ADC\_SC1 执行一次写操作或者在 FIFO 模式下对 ADC\_SC1 执行一组写操作 (ADCH 非全1)。
- 选择硬件触发时，发生硬件触发 (ADHWT) 事件。
- 连续转换模式开启时，结果被传输到转换结果寄存器。

如果连续转换模式开启，则在当前转换完成后会自动发起新转换。在软件触发操作中，连续转换在写入 ADC\_SC1 后开始，并一直继续直到中止。在硬件触发操作中，连续转换在硬件触发事件后开始，并一直继续直到中止。

#### 24.4.4.2 完成转换

当转换结果传输到转换结果寄存器 ADC\_R 时，ADC\_SC1[COCO]置位表明 ADC 完成转换。如果 ADC\_SC1[AIEN]在 ADC\_SC1[COCO]置位时为高电平，则会生成中断。

#### 24.4.4.3 中止转换

正在进行中的任何转换在下列情况中都会中止：

- 发生对 ADC\_SC1 的写操作。
  - 如果 ADC\_SC1[ADCH]非全 1 且 ADC\_SC4[AFDEP]为全 0，那么将中止当前转换并启动新的转换。
  - 如果 ADC\_SC4[AFDEP]非全 0，那么将中止当前转换和其余转换，并且不会启动新的转换。
  - 当 FIFO 被再次填充至 ADC\_SC4[AFDEP]指定的深度时，将会发起新转换。
- 发生对 ADC\_SC2、ADC\_SC3、ADC\_SC4、ADC\_CV 的写操作。这表明工作模式已发生改变，因此当前和其余转换（当 ADC\_SC4[AFDEP]非全 0 时）无效。
- MCU 复位。
- MCU 进入 Stop 模式，且 ADACK 未使能。

转换中止后，数据寄存器 ADC\_R 的内容保持不变。然而，这些内容仍然是完成最后一次成功转换之后传输的值。如果转换因复位而中止，ADC\_R 会恢复到复位状态。

#### 24.4.4.4 功率控制

ADC 模块在启动转换前一直保持在其空闲状态。如果选择 ADACK 作为转换时钟源，则还会使能 ADACK 时钟产生器。

模块为活动状态时，其功耗可通过置位 ADC\_SC3[ADLPC]来降低。这会降低  $f_{ADCK}$  的最大值（参见数据手册）。

## 24.4.4.5 采样时间和总转换时间

总转换时间取决于采样时间（由 ADC\_SC3[ADLSMP]决定）、MCU 总线频率、转换模式（8 位、10 位或 12 位）及转换时钟频率( $f_{ADCK}$ )。该模块激活后，便开始对输入进行采样。ADC\_SC3[ADLSMP]选择短（3.5 ADCK 周期）或长（23.5 ADCK 周期）采样时间。采样完成时，转换器与输入通道隔离开来，执行逐次逼近算法以确定模拟信号的数字值。转换算法完成时，转换结果转移到 ADC\_R。

如果总线频率低于  $f_{ADCK}$  频率，使能短采样时间时(ADC\_SC3[ADLSMP] = 0)，就无法保证连续转换采样时间的准确。如果总线频率低于  $f_{ADCK}$  频率的 1/11，使能长采样时间时(ADC\_SC3[ADLSMP] = 1)，就无法保证连续转换采样时间的准确。

下表总结了不同条件下的最大总转换时间。

表 24-2. 总转换时间与控制条件

转换类型	ADICLK	ADLSMP	最长总转换时间
8 位模式下，单次转换模式或者连续转换模式下的第一次转换	0x, 10	0	20 个 ADCK 周期 + 5 个总线时钟周期
10 位或 12 位模式下，单次转换模式或者连续转换模式下的第一次转换	0x, 10	0	23 个 ADCK 周期 + 5 个总线时钟周期
8 位模式下，单次转换模式或者连续转换模式下的第一次转换	0x, 10	1	40 个 ADCK 周期 + 5 个总线时钟周期
10 位或 12 位模式下，单次转换模式或者连续转换模式下的第一次转换	0x, 10	1	43 个 ADCK 周期 + 5 个总线时钟周期
8 位模式下，单次转换模式或者连续转换模式下的第一次转换	11	0	5 $\mu$ s + 20 个 ADCK + 5 个总线时钟周期
10 位或 12 位模式下，单次转换模式或者连续转换模式下的第一次转换	11	0	5 $\mu$ s + 23 个 ADCK + 5 个总线时钟周期
8 位模式下，单次转换模式或者连续转换模式下的第一次转换	11	1	5 $\mu$ s + 40 个 ADCK + 5 个总线时钟周期
10 位或 12 位模式下，单次转换模式或者连续转换模式下的第一次转换	11	1	5 $\mu$ s + 43 个 ADCK + 5 个总线时钟周期
8 位模式下，连续转换模式下的后续转换; $f_{BUS} > f_{ADCK}$	xx	0	17 个 ADCK 周期
10 位或 12 位模式下，连续转换模式下的后续转换; $f_{BUS} > f_{ADCK}$	xx	0	20 个 ADCK 周期
8 位模式下，连续转换模式下的后续转换; $f_{BUS} > f_{ADCK}/11$	xx	1	37 个 ADCK 周期
10 位或 12 位模式下，连续转换模式下的后续转换; $f_{BUS} > f_{ADCK}/11$	xx	1	40 个 ADCK 周期

最长总转换时间由选定的时钟源和分频比决定。时钟源可通过 ADC\_SC3[ADICLK]位选择，分频比通过 ADC\_SC3[ADIV]位指定。例如，在 10 位模式下，选择总线时钟作为输入时钟源，选定的输入时钟分频比为 1，总线频率为 8 MHz，则单次转换的转换时间通过下式计算：

$$\text{转换时间} = \frac{23\text{ADCKCyc}}{8\text{MHz}/1} + \frac{5\text{busCyc}}{8\text{MHz}} = 3.5\mu\text{s}$$

8MHz 时，单次转换时间的总线周期数为：

$$\text{总线周期} = 3.5\mu\text{s} \times 8\text{MHz} = 28$$

### 注

为了满足 ADC 规格要求，ADCK 频率必须介于  $f_{ADCK}$  最小值和  $f_{ADCK}$  最大值之间。

## 24.4.5 自动比较功能

比较功能经配置后可用于检查上限或下限。对输入进行采样和转换后，结果与比较值 (ADC\_CV) 的补值相加。与上限(ADC\_SC2[ACFGT] = 1)比较时，如果结果大于或等于比较值，则 ADC\_SC1[COCO]置位。与下限(ADC\_SC2[ACFGT] = 0)比较时，如果结果小于比较值，则 ADC\_SC1[COCO]置位。转换结果与比较值的补值相加所得到的值被传输到 ADC\_R。

比较功能使能，且完成转换后，如果比较条件不成立，则 ADC\_SC1[COCO]不置位且无任何数据传输到结果寄存器。如果 ADC 中断使能(ADC\_SC1[AIEN] = 1)，则在 ADC\_SC1[COCO]置位后会生成一个 ADC 中断。

当比较功能和 FIFO 使能，且所有转换完成时，如果 ADC\_SC4[ACFSEL]为低电平且所有比较条件均不成立；或 ADC\_SC4[ACFSEL]为高电平且并非所有比较条件都成立，则 ADC\_SC1[COCO]不置位。无论比较条件成立或不成立，只要 FIFO 使能，比较数据就会传输到结果寄存器中。

### 注

在 MCU 处于 Wait 或 Stop 模式时，比较功能可监测通道上的电压。在比较条件得到满足时，ADC 中断会唤醒 MCU。

### 注

比较功能在 FIFO 使能时无法在连续转换模式下工作。

## 24.4.6 FIFO 操作

ADC 模块支持 FIFO 操作以最大程度地减少 CPU 中断，从而降低 CPU 处理 ADC 中断服务程序的负荷。该模块包含两个 FIFO，分别缓存模拟输入通道号和结果。

FIFO 功能在 ADC\_SC4[AFDEP]位置位为非零值时使能。可通过这些位来指定 FIFO 深度。FIFO 最多可支持 8 个级别的缓冲区。

FIFO 功能使能时，模拟输入通道 FIFO 通过 ADC\_SC1[ADCH]位进行访问。必须按顺序将模拟输入通道号写入该 FIFO。如果通道 FIFO 的填充水平低于 ADC\_SC4[AFDEP]位指示的水平，无论设置的是软件触发还是硬件触发，ADC 都不会开始转换。对 ADC\_SC1[ADCH]进行读操作将读取目前正在转换的通道值。对 ADC\_SC1[ADCH]进行写操作将重新填充通道 FIFO 以开始新的转换。此操作将中止当前转换以及尚未开始的其他所有转换。当所有转换完成或 ADC 处于空闲状态时才可对 ADC\_SC1 执行写操作。

FIFO 功能使能时，FIFO 的结果通过 ADC\_R 寄存器进行访问。必须通过上述两个寄存器并且按照与模拟输入通道 FIFO 相同的顺序读取结果，这样才能获得正确结果。在 FIFO 模式下，对 ADC\_R 的读操作应等到所有转换都完成之后进行。无论设置的是软件触发还是硬件触发，只有模拟输入通道 FIFO 中的所有转换都完成之后，ADC\_SC1[COCO]位才会置位。当 FIFO 转换完成且 ADC\_SC1[COCO]位置位时，如果 ADC\_SC1[AIEN]置位，则将向 CPU 提交中断请求。

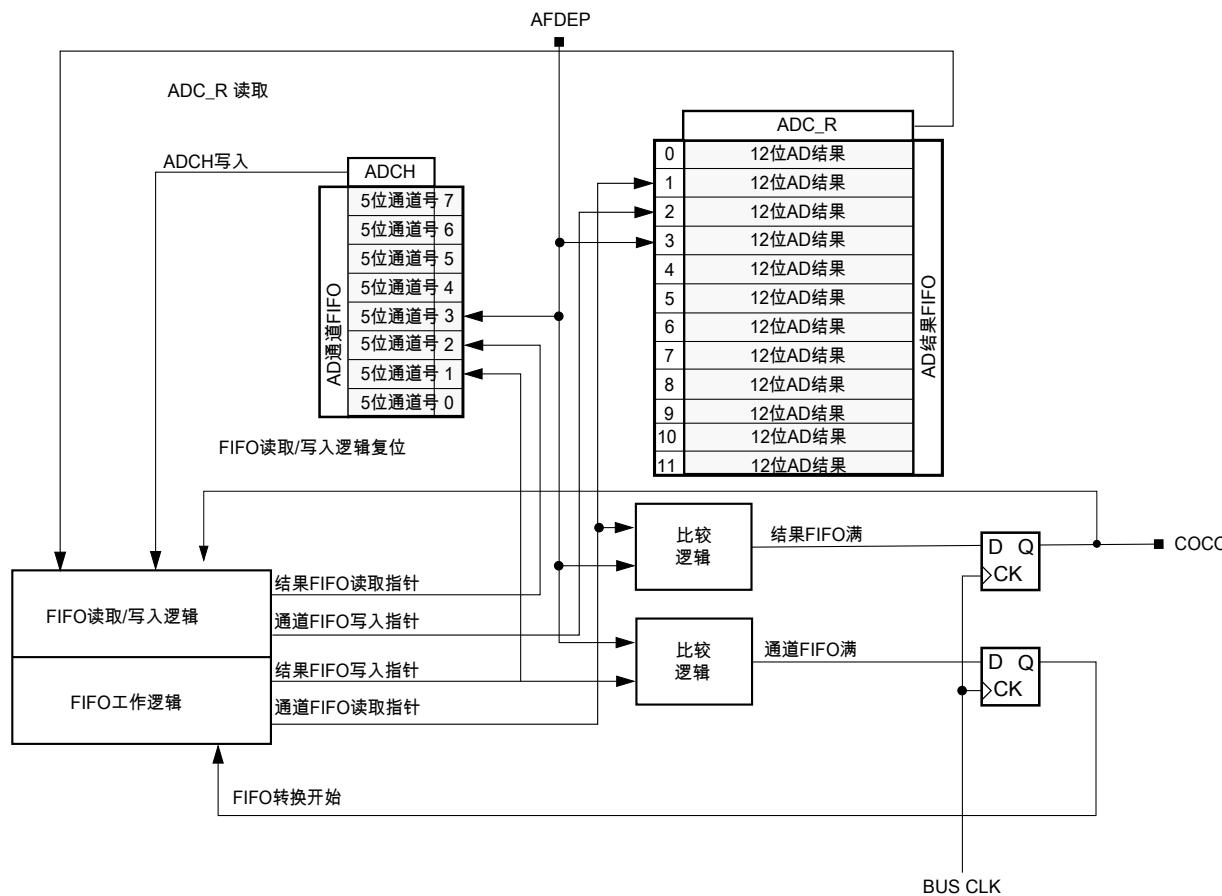


图 24-2. FADC FIFO 结构

如果使能软件触发模式，则当一个转换完成并且其结果存储在结果 FIFO 后，就会立即从模拟输入通道 FIFO 中提取下一个模拟通道。模拟输入通道 FIFO 中设置的所有转换都完成后，ADC\_SC1[COCO]位置位，如果 ADC\_SC1[AIEN]位置位，则将向 CPU 提交中断请求。

如果使能单次硬件触发模式 (ADC\_SC2[ADTRG]=1 且 ADC\_SC4[HTRGME]=0)，则只有当前转换完成，其结果已存储到结果 FIFO，并且将下一个硬件触发脉冲馈入 ADC 模块之后，才会从模拟输入通道 FIFO 中提取下一个模拟通道。如果使能多次硬件触发模式 (ADC\_SC2[ADTRG]=1 且 ADC\_SC4[HTRGME]=1)，则只有当前转换完成且其结果已存储到结果 FIFO 之后，才会从模拟输入通道 FIFO 中提取下一个模拟通道，下一转换无需等待下一个硬件触发脉冲便可开始。模拟输入通道 FIFO 中设置的所有转换都完成后，ADC\_SC1[COCO]位置位，如果 ADC\_SC1[AIEN]位置位，则将向 CPU 提交中断请求。

在单次转换模式下 (ADC\_SC1[ADCO]位清零)，当 ADC\_SC1[COCO]位置位时，ADC 停止转换，直至再次达到通道 FIFO 的填充要求或出现新的硬件触发脉冲。

FIFO 还提供扫描模式来简化输入通道 FIFO 的重复性工作。在 FIFO 模式下，当 ADC\_SC4[ASCANE]位置位时，无论输入通道 FIFO 为何值，FIFO 都只会使用其第一个通道单元中的内容。一旦第一个通道单元被写入，ADC 转换便开始在 FIFO 模

式下工作。对通道 FIFO 执行的后续写操作会不断覆盖该 FIFO 中的第一个通道元素。在扫描 FIFO 模式下，当结果 FIFO 中填充的内容达到 ADC\_SC1[COCO]位指示的深度时，ADC\_SC4[AFDEP]位置位。

在连续转换模式 (ADC\_SC1[ADCO]位置位) 下，当所有转换完成时，ADC 立即开始下一转换。ADC 模块从模拟输入通道 FIFO 一开始就会获取模拟输入通道。

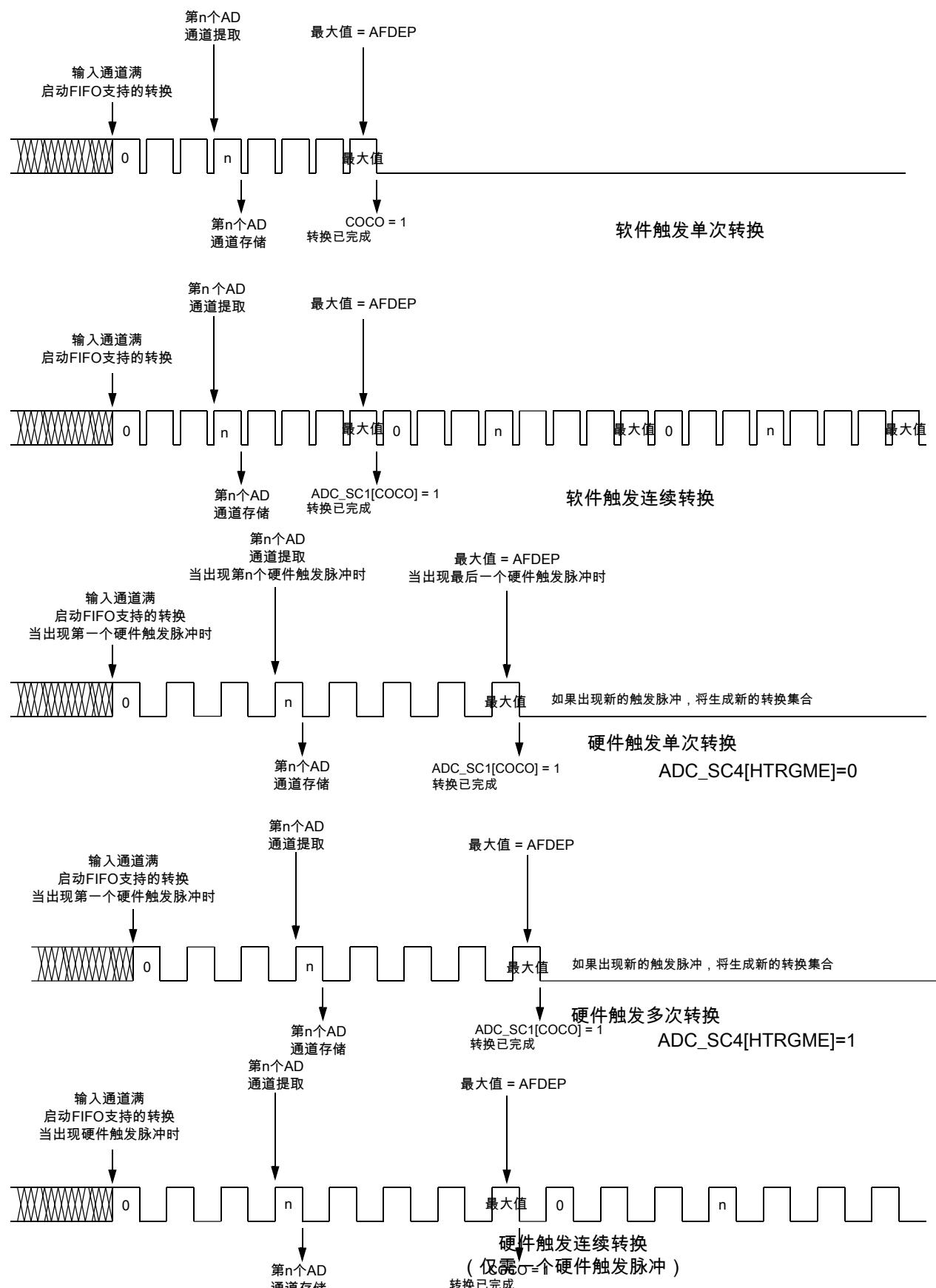


图 24-3. ADC FIFO 转换序列

KEA128 子系列参考手册, Rev. 2, July 2014

## 24.4.7 MCUWait 模式下的操作

Wait 模式是低功耗的待机模式，通过该模式可迅速恢复，因为时钟源保持有效。如果转换在 MCU 进入 Wait 模式时正在进行，则它会一直继续，直到完成。当 MCU 处于 Wait 模式时，如果硬件触发或者连续转换模式已使能，可发起新的转换。

在 Wait 模式下，可选择总线时钟、总线时钟 2 分频、ALTCLK 和 ADACK 作为转换时钟源。

ADC 中断已使能(ADC\_SC1[AIEN] = 1)的情况下，转换完成事件会使 ADC\_SC1[COCO]置位，并生成 ADC 中断将 MCU 从 Wait 模式中唤醒。

## 24.4.8 MCU Stop 模式下的操作

Stop 模式是低功耗待机模式，在此模式下，MCU 上的大多数甚至全部时钟源都被禁用。

### 24.4.8.1 采用 Stop 模式并禁用 ADACK

如果没有选择异步时钟 ADACK 作为转换时钟，则执行 STOP 指令会中止当前转换并使 ADC 处于空闲状态。ADC\_R 的内容不受 Stop 模式的影响。退出 Stop 模式后，需要一次软件或硬件来使转换继续。

### 24.4.8.2 采用 Stop 模式并启用 ADACK

如果选择 ADACK 作为转换时钟，则 ADC 在 Stop 模式下将连续运行。为保证 ADC 的运行，MCU 的电压调节器必须在 Stop 模式期间保持有效。有关此 MCU 的配置信息，请参见相关模块简介。

如果转换在 MCU 进入 Stop 模式时正在进行，则它会一直继续，直到完成。当 MCU 处于 Stop 模式时，如果硬件触发或者连续转换模式已使能，可发起新的转换。

ADC 中断已使能(ADC\_SC1[AIEN] = 1)的情况下，转换完成事件会使 ADC\_SC1[COCO]置位，并生成 ADC 中断将 MCU 从 Stop 模式中唤醒。在 FIFO 模式下，ADC 无法彻底完成转换操作，也无法将 MCU 从 Stop 模式中唤醒。

## 注

ADC 模块可将系统从低功耗停止模式中唤醒，并让 MCU 开始消耗运行级别的电流，但不生成系统级中断。为了阻止这种情况的发生，进入 Stop 模式并继续执行 ADC 转换时，必须消除数据传输阻塞机制。

## 24.5 初始化信息

本节将举例说明 ADC 模块的初始化和配置步骤。您可以将模块配置为 8 位、10 位或 12 位分辨率、单一或连续转换以及轮询或中断方法，还有其他许多选项可供选择。有关本例中使用的信息，请参见 ADC\_SC3 寄存器。

## 注

十六进制值的前缀是 0x，二进制值的前缀是%，十进制值无前缀字符。

### 24.5.1 ADC 模块初始化示例

在使用 ADC 模块完成转换前，首先必须对它进行初始化。下面介绍 ADC 模块的初始化方法。

#### 24.5.1.1 初始化序列

常见的初始化序列如下：

1. 更新配置寄存器(ADC\_SC3)，以选择输入时钟源以及生成内部时钟 ADCK 所使用的分频比。此寄存器还可用于选择采样时间和低功耗配置。
2. 更新状态和控制寄存器 2 (ADC\_SC2)，以选择硬件或软件转换触发、比较功能选项（如果启用）。
3. 更新状态和控制寄存器 1 (ADC\_SC1)，以选择转换是连续进行还是仅完成一次，以及是启用还是禁用转换完成中断。还可在此处选择执行转换的输入通道。

#### 24.5.1.2 伪代码示例

在本例中，ADC 模块设置为启用中断，并以低功率、长采样时间在输入通道 1 上执行单一的 10 位转换，其中内部的 ADCK 时钟是总线时钟的 1 分频。

## 示例: 24.5.1.2.1 常规 ADC 初始化程序

```
void ADC_init(void)
{
    /* The following code segment demonstrates how to initialize ADC by low-power mode,
long
    sample time, bus frequency, software triggered from AD1 external pin without FIFO
enabled
    */
    ADC_APCTL1 = ADC_APCTL1_ADPC1_MASK;
    ADC_SC3 = ADC_SC3_ADLPC_MASK | ADC_SC3_ADLSMP_MASK | ADC_SC3_MODE0_MASK;
    ADC_SC2 = 0x00;
    ADC_SC1 = ADC_SC1_AIEN_MASK | ADC_SC1_ADCH0_MASK;
}
```

## 24.5.2 ADC FIFO 模块初始化示例

在使用 ADC 模块执行 FIFO 转换前，首先必须执行初始化过程。常见序列如下：

1. 更新配置寄存器(ADC\_SC3)，以选择输入时钟源以及生成内部时钟 ADCK 所使用的分频比。此寄存器还可用于选择采样时间和低功耗配置。
2. 更新配置寄存器(ADC\_SC4)，以选择 FIFO 扫描模式、FIFO 比较功能模式 (OR 或 AND 运算) 和 FIFO 深度。
3. 更新状态和控制寄存器 2 (ADC\_SC2)，以选择硬件或软件转换触发、比较功能选项 (如果启用)。
4. 更新状态和控制寄存器 1 (ADC\_SC1)，以选择转换是连续进行还是仅完成一次，以及是启用还是禁用转换完成中断。还可在此处选择执行转换的输入通道。

### 24.5.2.1 伪代码示例

本例中，ADC 模块设置如下：使能中断，以低功耗和长采样时间对输入通道 1、3、5、7 执行单次硬件触发的 10 位 4 级 FIFO 转换。内部 ADCK 时钟从总线时钟获得 (1 分频)。

## 示例: 24.5.2.1.1 FIFO ADC 初始化程序

```
void ADC_init(void)
{
/* The following code segment demonstrates how to initialize ADC by low-power mode, long
sample time, bus frequency, hardware triggered from AD1, AD3, AD5, and AD7 external pins
with 4-level FIFO enabled */

    ADC_APCTL1 = ADC_APCTL1_ADPC6_MASK | ADC_APCTL1_ADPC5_MASK | ADC_APCTL1_ADPC3_MASK |
ADC_APCTL1_ADPC1_MASK;

    ADC_SC3 = ADC_SC3_ADLPC_MASK | ADC_SC3_ADLSMP_MASK | ADC_SC3_MODE1_MASK;
```

```
// setting hardware trigger  
ADC_SC2 = ADC_SC2_ADTRG_MASK ;  
  
//4-Level FIFO  
ADC_SC4 = ADC_SC4_AFDEP1_MASK | ADC_SC4_AFDEP0_MASK;  
  
// dummy the 1st channel  
ADC_SC1 = ADC_SC1_ADCH0_MASK;  
  
// dummy the 2nd channel  
ADC_SC1 = ADC_SC1_ADCH1_MASK | ADC_SC1_ADCH0_MASK;  
  
// dummy the 3rd channel  
ADC_SC1 = ADC_SC1_ADCH2_MASK | ADC_SC1_ADCH0_MASK;  
  
// dummy the 4th channel and ADC starts conversion  
ADC_SC1 = ADC_SC1_AIEN_MASK | ADC_SC1_ADCH2_MASK | ADC_SC1_ADCH1_MASK | ADC_SC1_ADCH0_MASK;  
}
```

### 示例: 24.5.2.1.2 FIFO ADC 中断服务程序

```
unsigned short buffer[4];  
interrupt VectorNumber_Vadc void ADC_isr(void)  
{  
    /* The following code segment demonstrates read AD result FIFO */  
    // read conversion result of channel 1 and COCO bit is cleared  
    buffer[0] = ADC_R;  
    // read conversion result of channel 3  
    buffer[1] = ADC_R;  
    // read conversion result of channel 5  
    buffer[2] = ADC_R;  
    // read conversion result of channel 7  
    buffer[3] = ADC_R;  
}
```

#### 注

ADC\_R 是 16 位 ADC 结果寄存器，由 ADC\_RH 和 ADC\_RL 合并得来

## 24.6 应用信息

本节介绍 ADC 模块的应用信息。ADC 已集成到微控制器中，可用在需要 A/D 转换器的嵌入式控制应用中。

### 24.6.1 外部引脚和布线

下面几节介绍与 ADC 模块相关的外部引脚以及如何使用它们才能达到最佳效果。

## 24.6.1.1 模拟电源引脚

ADC 模块上的模拟电源和接地电源 ( $V_{DDA}$  和  $V_{SSA}$ ) 在某些设备上可用作单独的引脚。在有些设备上,  $V_{SSA}$  与 MCU 数字  $V_{SS}$  共享同一引脚。在其他设备上,  $V_{SSA}$  和  $V_{DDA}$  共享 MCU 数字电源引脚。在这些情况下, 在引脚上 (与对应的数字电源相同) 为模拟电源粘合单独的芯片, 以确保模拟电源和数字电源间保持一定的隔离。

如果用在单独的引脚上,  $V_{DDA}$  和  $V_{SSA}$  必须连接到与其对应的 MCU 数字电源 ( $V_{DD}$  和  $V_{SS}$ ) 相同的电势上, 并且布线时必须非常小心, 以最大限度提高抗噪性并且旁路电容必须尽可能靠近封装放置。

如果分别使用单独的模拟电源和数字电源, 则两者之间的接地连接必须位于  $V_{SSA}$  引脚上。这应该是两个电源之间唯一的接地连接 (如果可能)。 $V_{SSA}$  引脚是一个很好的单点接地位置。

## 24.6.1.2 模拟基准引脚

除模拟电源之外, ADC 模块还具有用于两个基准电压输入的连接。高基准电压为  $V_{REFH}$ , 在某些器件上可能与  $V_{DDA}$  引脚共用。低基准电压为  $V_{REFL}$ , 在某些器件上可能与  $V_{SSA}$  引脚共用。

如果  $V_{REFH}$  存在单独的引脚, 可将其连接到与  $V_{DDA}$  相同的电势, 或由介于数据手册中指定的最小  $V_{DDA}$  与  $V_{DDA}$  电势之间的外部电压源驱动 ( $V_{REFH}$  不能超过  $V_{DDA}$ )。如果  $V_{REFL}$  存在单独的引脚, 可将其连接到与  $V_{SSA}$  相同的电势。必须小心安排  $V_{REFH}$  和  $V_{REFL}$  以便实现最大抗噪性, 而且旁路电容必须尽可能靠近封装放置。

通过  $V_{REFH}$  和  $V_{REFL}$  循环引出电流尖峰形式的交流电流, 用来为每个连续近似步骤中的电容器阵列供电。满足此电流需求的最佳外部组件是  $0.1 \mu F$  的优质高频电容器。此电容器连接在  $V_{REFH}$  和  $V_{REFL}$  之间, 位置必须尽量靠近封装引脚。不建议在路径中使用电阻, 因为电流产生的压降会导致转换错误。此路径中的电感必须最小 (仅限干扰)。

## 24.6.1.3 模拟输入引脚

外部模拟输入通常由 MCU 器件上的数字 I/O 引脚共用。设置引脚控制寄存器的控制位可以禁用相应的引脚 I/O 控制。虽然引脚控制位的设置与否并不影响对输入信号的模数转换, 但当某个引脚用作模拟输入时, 建议将引脚控制寄存器的相应控制位始终置 1。首先这能避免争用问题, 因为输出缓冲区处于其高阻抗状态且上拉电阻禁用。另外, 输入缓冲区在其输入不处于  $V_{DD}$  或  $V_{SS}$  时会消耗直流电流。所以应当将作为模拟输入的引脚所对应的引脚控制寄存器控制位置 1, 以实现最低的工作电流。

经验数据表明，当存在噪声或者源阻抗较高时，模拟输入上的电容可以改善性能。使用具有良好高频特性的  $0.01 \mu\text{F}$  电容就足够了。在某些情况下并不一定要使用这类电容，但是使用，则必须将其尽可能地靠近封装引脚放置并将  $V_{SSA}$  作为基准电压。

为确保转换正确完成，输入电压必须介于  $V_{REFH}$  和  $V_{REFL}$  之间。如果输入电压等于或大于  $V_{REFH}$ ，转换器电路会将信号转换为 0xFFFF (满刻度 12 位表示法)、0x3FF (满刻度 10 位表示法) 或 0xFF (满刻度 8 位表示法)。如果输入电压等于或小于  $V_{REFL}$ ，则转换器电路会将其转换为 0x000。在  $V_{REFH}$  和  $V_{REFL}$  之间的输入电压属于直线线性转换。采用电容进行充电时，存在与  $V_{REFL}$  有关的短时电流。ADC\_SC3[ADLSMP] 为低电平时，对该输入进行采样的时间为 ADCK 源的 3.5 个周期，ADC\_SC3[ADLSMP] 为高电平时，进行采样的时间为 23.5 个周期。

为了尽量减小电流注入造成的准确度损失，转换过程中不能对与模拟输入引脚相邻的引脚进行电平转换操作。

## 24.6.2 误差来源

A/D 转换出现误差的来源很多。下面几节将详细介绍一下。

### 24.6.2.1 采样误差

为正确转换，输入的采样时间必须长到能实现正确的精度。假设最大输入电阻约为  $7 \text{ k}\Omega$ 、输入电容约为  $5.5 \text{ pF}$ ，则在最小采样窗口 (3.5 个 ADCK 周期，而 ADCK 的最大频率为 8MHz) 内可将采样误差控制到 1/4 LSB 内 (12 位分辨率)，同时还假设外部模拟源 ( $R_{AS}$ ) 的电阻始终小于  $2 \text{ k}\Omega$ 。

通过设置 ADC\_SC3[ADLSMP] (以将采样窗口增加到 23.5 个周期)，或减小 ADCK 频率以增加采样时间，可提高源电阻或采样精度。

### 24.6.2.2 引脚漏电误差

如果外部模拟源电阻 ( $R_{AS}$ ) 偏高，I/O 引脚上的漏电流可能会导致转换出错。如果应用程序无法接受这一误差，则使  $R_{AS}$  低于  $V_{DDA}/(2^N * I_{LEAK})$  以便将漏电流误差控制在 1/4LSB 内 ( $N = 8$  (8 位),  $N = 10$  (10 位),  $N = 12$  (12 位) 模式)。

### 24.6.2.3 噪声性误差

采样或转换过程中出现的系统噪音会影响转换的准确度。只有在满足以下条件时，才能保证 ADC 准确度值达到预期：

- $V_{REFH}$  到  $V_{REFL}$  之间有一个  $0.1 \mu F$  的低 ESR 电容器。
- $V_{DDA}$  到  $V_{SSA}$  之间有一个  $0.1 \mu F$  的低 ESR 电容器。
- 如果主电源使用了电感隔离，则  $V_{DDA}$  到  $V_{SSA}$  之间还要放一个  $1 \mu F$  的电容器。
- $V_{SSA}$  (和  $V_{REFL}$ , 如果连接) 连接到接地层中某安静点处的  $V_{SS}$ 。
- 在发起 (硬件触发转换) ADC 转换前，或发起 (硬件或软件触发转换) ADC 转换后立即让 MCU 在 Wait 或 Stop 模式下运行。
  - 对于软件触发的转换，对 ADC\_SC1 执行写操作后立即使用停止指令。
  - 对于 Stop 模式操作，选择 ADACK 作为时钟源。Stop 模式下的操作可减少  $V_{DD}$  噪音，但也会因停止恢复而增加有效转换时间。
- 转换期间，MCU 上无 I/O 切换、输入或输出。

在某些情况下，外部系统的活动会引起辐射噪声或传导噪声的传播，或使过大的  $V_{DD}$  噪声耦合到 ADC 中。在这类情况下，如果无法将 MCU 置于 Wait 或 Stop 模式，或者无法暂停 I/O 活动，建议通过以下动作降低噪声对精度的影响：

- 在  $V_{REFL}$  或  $V_{SSA}$  的选定输入通道上放一个  $0.01 \mu F$  的电容器 ( $C_{AS}$ ) (此举可改善噪音问题，但会影响基于外部模拟源阻抗的采样率)。
- 将模拟输入接连转换多次并除以结果总数得出结果的平均值。消除一次 1LSB 的错误所带来的影响需要四个样本。
- 通过使用异步时钟 (ADACK) 并计算平均值来降低同步噪音的影响。与 ADCK 同步的噪音不能计算平均值。

### 24.6.2.4 编码宽度和量化误差

ADC 将理想的线性转换函数量化为 4096 步 (在 12 位模式下)。每一步在理想情况下都具有相同的高度 (一个编码) 和宽度。宽度定义为某个编码的转变点与下一个编码的转变点之间的差值。定义为 1LSB 的 N 位转换器的理想编码宽度 (在本例中，N 可以是 8、10 或 12) 是：

$$1 \text{ lsb} = (V_{REFH}) / 2^{N_{REFL}}$$

结果的数字化导致存在固有的量化误差。对于 8 位或 10 位转换，线性转换函数与实际的转换函数在一些离散点上精确重合，编码会在电压位于这些点之间的中点时发生切换。因此，量化误差在 8 位模式或 10 位模式下将为  $\pm 1/2$  lsb。并且，第一个编码 (0x000) 宽度只有 1/2 lsb，最后一个编码 (0xFF 或 0x3FF) 的宽度为 1.5 lsb。

对于 12 位转换，编码只有在电压变化了一个完整的编码宽度后才会切换，因此量化误差为 -1 lsb 到 0 lsb，每个振幅的编码宽度为 1 lsb。

### 24.6.2.5 线性误差

ADC 可能还会呈现几种非线性形式。虽然已经尽全力减少这些误差，但是系统必须知道误差的存在，因为它们会影响总体的准确度。这些误差包括：

- 零刻度误差 ( $E_{ZS}$ ) (有时也称为偏移量) — 该误差是指第一个编码的实际编码宽度和理想编码宽度 (8 位或 10 位模式下为 1/2 lsb; 12 位模式下为 1 lsb) 之间的差值。如果第一个转换为 0x001，则零刻度误差为实际的 0x001 编码宽度与其理想宽度 (1 lsb) 之间的差值。
- 满刻度误差 ( $E_{FS}$ ) — 该误差是指最后一个编码的实际编码宽度与理想编码宽度 (8 位或 10 位模式下为 1.5 lsb; 12 位模式下为 1 lsb) 之间的差值。如果最后一个转换为 0x3FE，则满刻度误差为实际的 0x3FE 编码宽度与其理想宽度 (1 lsb) 之间的差值。
- 差分非线性 (DNL) — 该误差是指所有转换的实际编码宽度与理想编码宽度之间的最差情况差值。
- 积分非线性 (INL) — 该误差是指 DNL 累加和的绝对值所达到的最大值。更简而言之，INL 是指所有编码中，实际切换电压与其对应的理想切换电压之间的最差情况差值。
- 未调整总误差 (TUE) — 该误差是指实际转换函数与理想线性转换函数之间的差值，包括所有形式的误差。

### 24.6.2.6 编码抖动、非单调性和失码

模数转换器对三种特殊形式的错误特别敏感。它们是编码抖动、非单调性和失码。

编码抖动发生在给定的输入电压在某些点经过重复采样后转换为两个值中的某一个值时。理想情况下，当输入电压小于但无限接近切换电压时，转换器会产生较低的编码 (反之亦然)。然而，当输入电压接近切换电压时，即使少量的系统噪声也可能使转换器在两个编码间变得不确定。在 8 位或 10 位模式下，此范围一般约为  $\pm 1/2$  lsb；在 12 位模式下，则约为 2 lsb，随噪声的增大而增大。

通过对输入重复采样并求结果的平均值可以减少出现此错误的几率。此外，[噪声性误差](#) 中介绍的方法也可以减少出现此错误的几率。

除了编码抖动外，当转换器因较高的输入电压而切换到较低的编码时，会出现非单调性问题。失码是指任何输入都不会转换为的那些值。

在 8 位或 10 位模式下，ADC 保证是单调的并且无失码。

# 第 25 章 模拟比较器 (ACMP)

## 25.1 简介

模拟比较器模块(ACMP)提供一个用于比较两个模拟输入电压的电路。此比较器电路适用于在整个供电电压范围内操作 (全摆幅操作)。

模拟复用器提供一个用于从 4 个通道中选择模拟输入信号的电路。一个信号由 6 位 DAC 提供。多路复用器电路适用于在整个供电电压范围内操作。6 位 DAC 是一个 64-tap 梯形电阻网络，可为需要电压基准的应用提供可选的电压基准。此 64-tap 梯形电阻网络将电源基准  $V_{in}$  分成 64 个电压电平。6 位数字信号输入可选择产生不同的输出电压，范围从  $V_{in}$  至  $V_{in}/64$ 。可从两个电压源选择  $V_{in}$ 。

### 25.1.1 特性

ACMP 特性包括：

- 可在 2.7 V 至 5.5 V 的整个电源电压范围内操作
- 片上 6 位分辨率 DAV，基准电压源可以选择  $V_{DD}$  或内部带隙基准
- 可配置迟滞
- 可在比较器输出上升沿、下降沿或者任何边沿时选择产生中断
- 可选择翻转比较器输出
- 最多 4 个可选择比较器输入
- 可在 Stop 模式中操作

### 25.1.2 工作模式

本节说明 ACMP 在 Wait、Stop 和 Background Debug 模式下的操作。

### 25.1.2.1 Wait 模式下的操作

在 Wait 模式下，如果使能，ACMP 将继续工作。如果使能，中断可唤醒 MCU。

### 25.1.2.2 Stop 模式下的操作

如果已启用，ACMP（包括 DAC 和 CMP）将继续在 Stop 模式中操作。如果 ACMP\_CS[ACIE]置位，则可生成 ACMP 中断将 MCU 从 Stop 模式中唤醒。

如果通过中断而退出 Stop 模式，那么在进入 Stop 模式之前 ACMP 设置仍将保持。如果通过复位而退出 Stop 模式，那么 ACMP 将进入复位状态。

由于 DAC 消耗额外的电能，因此如果 DAC 输出不用作 ACMP 的基准输入，那么用户必须关闭 DAC 以省电。

### 25.1.2.3 Debug 模式下的操作

当 MCU 处于 Debug 模式时，ACMP 继续正常工作。

### 25.1.3 结构框图

下图是 ACMP 模块的结构框图。

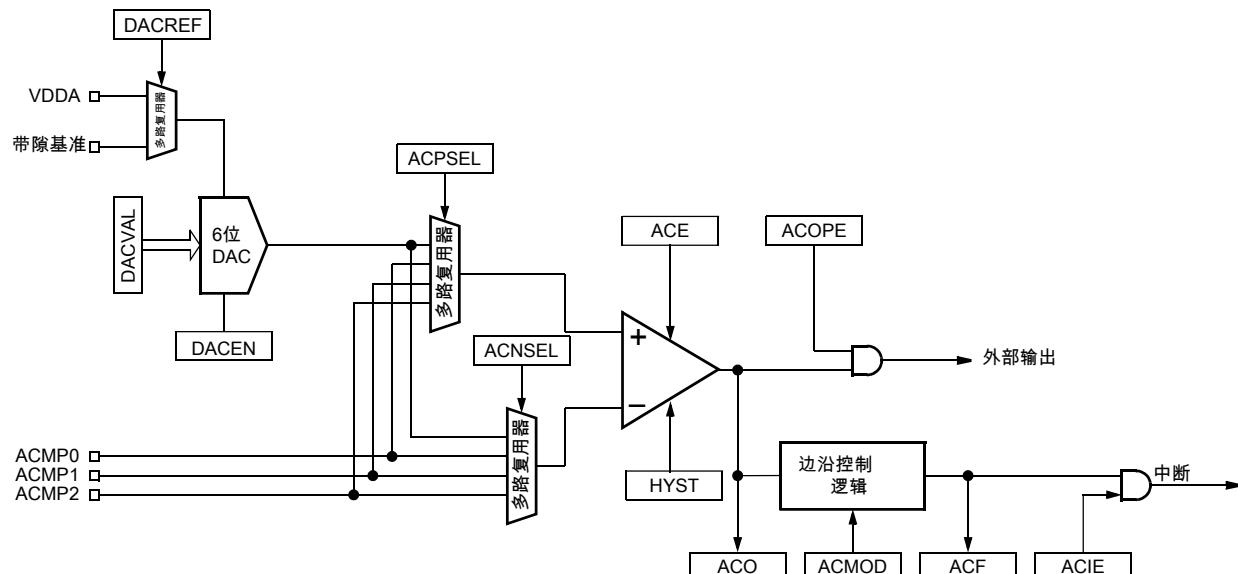


图 25-1. ACMP 结构框图

## 25.2 外部信号说明

ACMP 的输出也可映射到外部引脚。当该输出映射到外部引脚时，ACMP\_CS[ACOPE]控制该引脚来使能/禁用 ACMP 输出功能。

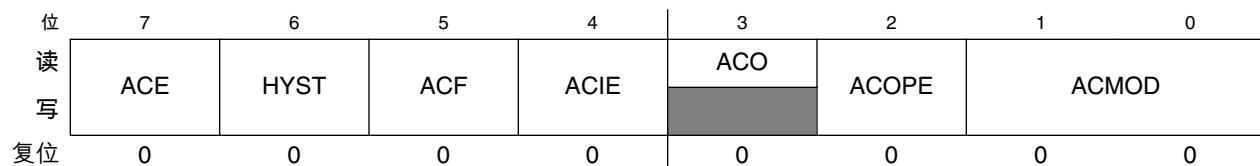
## 25.3 存储器映像和寄存器定义

### ACMP 存储器映射

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4007_3000	ACMP 控制和状态寄存器 (ACMP0_CS)	8	R/W	00h	<a href="#">25.3.1/341</a>
4007_3001	ACMP 控制寄存器 0 (ACMP0_C0)	8	R/W	00h	<a href="#">25.3.2/342</a>
4007_3002	ACMP 控制寄存器 1 (ACMP0_C1)	8	R/W	00h	<a href="#">25.3.3/343</a>
4007_3003	ACMP 控制寄存器 2 (ACMP0_C2)	8	R/W	00h	<a href="#">25.3.4/343</a>
4007_4000	ACMP 控制和状态寄存器 (ACMP1_CS)	8	R/W	00h	<a href="#">25.3.1/341</a>
4007_4001	ACMP 控制寄存器 0 (ACMP1_C0)	8	R/W	00h	<a href="#">25.3.2/342</a>
4007_4002	ACMP 控制寄存器 1 (ACMP1_C1)	8	R/W	00h	<a href="#">25.3.3/343</a>
4007_4003	ACMP 控制寄存器 2 (ACMP1_C2)	8	R/W	00h	<a href="#">25.3.4/343</a>

### 25.3.1 ACMP 控制和状态寄存器 (ACMPx\_CS)

地址: 基址 基准 + 0h 偏移



#### ACMPx\_CS 字段描述

字段	描述
7 ACE	模拟比较器使能 使能 ACMP 模块。 0 ACMP 禁用。 1 ACMP 使能。
6 HYST	模拟比较器迟滞选择 选择 ACMP 迟滞。

下一页继续介绍此表...

## ACMPx\_CS 字段描述 (继续)

字段	描述
	0 20 mV。 1 30 mV。
5 ACF	ACMP 中断标志位  当 ACMP 输出有一个由 ACMOD 定义的有效边沿时，由硬件同步置位。该位的置位滞后于 ACMPO 至总线时钟。将 0 写入 ACF 位可将该位清零。将 1 写入该位无效。
4 ACIE	ACMP 中断使能  使能 ACMP CPU 中断。  0 禁用 ACMP 中断。 1 使能 ACMP 中断。
3 ACO	ACMP 输出  读取 ACO 将返回模拟比较器输出的当前值。ACMP 禁用 ( ACE=0 ) 时，ACO 复位并读作 0。
2 ACOPE	ACMP 输出引脚使能  ACOPE 使能焊盘逻辑，以便能将输出置于外部引脚上。  0 不允许 ACMP 输出置于外部引脚上。 1 允许 ACMP 输出置于外部引脚上。
ACMOD	ACMP MOD  确定中断触发器的触发模式。  00 ACMP 中断在输出下降沿发生。 01 ACMP 中断在输出上升沿发生。 10 ACMP 中断在输出下降沿发生。 11 ACMP 中断在输出下降沿或上升沿发生。

## 25.3.2 ACMP 控制寄存器 0 (ACMPx\_C0)

地址: 基址 基准 + 1h 偏移



## ACMPx\_C0 字段描述

字段	描述
7–6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5–4 ACPSEL	ACMP 正输入选择  00 外部基准 0 01 外部基准 1

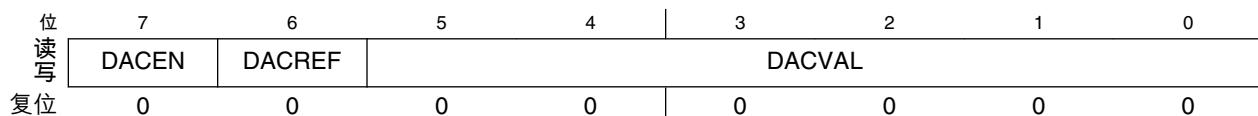
下一页继续介绍此表...

## ACMPx\_C0 字段描述 (继续)

字段	描述
	10 外部基准 2 11 DAC 输出
3–2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
ACNSEL	ACMP 负输入选择  00 外部基准 0 01 外部基准 1 10 外部基准 2 11 DAC 输出

## 25.3.3 ACMP 控制寄存器 1 (ACMPx\_C1)

地址: 基址 基准 + 2h 偏移



## ACMPx\_C1 字段描述

字段	描述
7 DACEN	DAC 使能  使能 6 位 DAC 的输出。  0 DAC 禁用。 1 DAC 使能。
6 DACREF	DAC 基准选择  0 DAC 选择带隙基准作为基准。 1 DAC 选择 V <sub>DDA</sub> 作为基准。
DACVAL	DAC 输出电平选择  使用以下公式选择输出电压 : V <sub>output</sub> = (V <sub>in</sub> /64) × (DACVAL[5:0]+1)。V <sub>output</sub> 范围为 V <sub>in</sub> /64 至 V <sub>in</sub> ，步进为 V <sub>in</sub> /64

## 25.3.4 ACMP 控制寄存器 2 (ACMPx\_C2)

地址: 基址 基准 + 3h 偏移



**ACMPx\_C2 字段描述**

字段	描述
7–3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
ACIPE	ACMP 输入引脚使能  该 3 位字段控制对应的 ACMP 外部引脚是否能由模拟输入驱动。  0 不允许对应的外部模拟输入。 1 允许对应的外部模拟输入。

## 25.4 功能说明

ACMP 模块就功能而言由两部分组成：数模转换器(DAC)和比较器(CMP)。

DAC 包括 64 级 DAC (数模转换器) 和相关控制逻辑。通过置位 ACMP\_C1[DACREF]，DAC 可选择  $V_{DD}$  或片上带隙基准源这两个基准输入的其中之一作为 DAC 输入  $V_{in}$ 。DAC 使能后，将 ACMP\_C1[DACVAL]中设置的数据转换为步进式模拟输出，然后馈入 ACMP 作为内部基准输入。此递阶模拟输出还将在此模块之外映射。输出电压范围为  $V_{in}/64$  到  $V_{in}$ 。步进大小为  $V_{in}/64$ 。

ACMP 可以实现正输入与负输入的模拟比较，然后提供一个数字输出和相关的中断。ACMP 的正负输入均可从四个通用输入中选择：三个外部基准输入和一个来自 DAC 输出的内部基准输入。ACMP 的正输入由 ACMP\_C0[ACPSEL]选定，其负输入由 ACMP\_C0[ACNSEL]选定。可通过使用适当的值配置 ACMPC0 来比较 8 个输入的任何配对。

通过置位 ACMP\_CS[ACE]使能 ACMP 之后，比较结果以数字输出呈现。只要 ACMP\_CS[ACMOD]中定义的有效边沿出现，ACMP\_CS[ACF]的电平就会变为有效值。如果 ACMP\_CS[ACIE]置位，那么就会发生 ACMP CPU 中断。有效边沿由 ACMP\_CS[ACMOD]定义。当 ACMP\_CS[ACMOD] = 00b 或 10b 时，只有 ACMP 输出的下降沿有效。当 ACMP\_CS[ACMOD] = 01b 时，只有 ACMP 输出的上升沿有效。当 ACMP\_CS[ACMOD] = 11b 时，ACMP 输出的上升沿和下降沿均有效。

ACMP 输出经总线时钟同步后产生 ACMP\_CS[ACO]，以便 CPU 能读取比较结果。在 stop3 模式下，即使 ACMP 的输出改变，ACMPO 也无法及时更新。因为输出信号的同步以及 ACMP\_CS[ACO]的更新都需要 CPU 被唤醒，因为此时总线时钟有效。ACMP\_CS[ACO]根据比较结果而改变，因此它可以用作一个跟踪标志，连续指示输入的电压变化。

如果选择芯片外部的基准输入作为 ACMP 的输入，那么必须置位对应的 ACMP\_C2[ACIPE]位以使能来自焊盘接口的输入。如果需要将 ACMP 的输出置于外部引脚上，那么必须置位 ACMP\_CS[ACOPE]位以使能焊盘逻辑的 ACMP 引脚功能。

## 25.5 ACMP 的设置和操作

ACMP 两部分 (DAC 和 CMP) 的设置和操作可以独立进行。但是，如果 DAC 用作 CMP 的输入，那么在使能 ACMP 之前必须对 DAC 进行配置。

由于输入转换会导致 ACMP 输入上出现问题，因此用户应在启用 ACMP 之前完成输入选择，并且不能在启用 ACMP 时更改选择输入以避免意外输出。同样，更改 ACMP\_C1[DACVAL]之后，DAC 会有一定的设置延迟，因此用户应当在使能 DAC 之前完成 ACMP\_C1[DACVAL]的设置。

## 25.6 复位

复位期间，ACMP 会按照默认模式配置，此时 CMP 和 DAC 都被禁用。

## 25.7 中断

如果 ACMP\_CS[ACMOD]定义的有效边沿出现时总线时钟可用，那么 ACMP\_CS[ACF]的电平就会变为有效值。如果 ACMP\_CS[ACIE]置位，那么会发生 ACMP 中断事件。ACMP\_CS[ACF]位的电平将一直保持有效值，直到 ACMP 中断被软件清除。在 Stop 模式下，ACMP 输入的有效边沿会生成一个异步中断，它可将 MCU 从 Stop模式中唤醒。将 0 写入 ACMP\_CS[ACF]位可清除该中断。



# 第 26 章

## FlexTimer 模块(FTM)

### 26.1 简介

#### 注

关于与具体芯片相关的本模块详细操作示例，请参见芯片配置信息。

FlexTimer 模块(FTM)是一种两通道至八通道的定时器，支持输入捕捉、输出比较和生成用于控制电机和电源管理的 PWM 信号。FTM 的时间基准是一个 16 位计数器，可用作无符号或有符号计数器。

#### 26.1.1 FlexTimer 的基本理念

FTM 是基于一个简单的定时器（即 HCS08 定时器 PWM 模块--TPM，这个定时器在飞思卡尔 8 位微处理器上已经使用多年）而设计的。FTM 在功能方面实现了延伸，可以更好的满足电机控制、数字照明解决方案及功率转换的需求。另外，带有 TPM 模块的 FTM 成本低，并支持向下兼容。

其增强的主要功能如下：

- 带符号的向上计数器
- 支持硬件死区插入
- 故障控制输入
- 增强型触发功能
- 初始化和极性控制

与 TPM 一致的功能均分配有可以完全向下兼容的寄存器。FTM 还可在同一内核平台上使用代码而无需修改，即可实现相同功能。

已通过一组专用的寄存器添加了电机控制和功率转换特性，所有全新特性均默认为关断。硬件死区插入、极性控制、故障控制以及输出强制和屏蔽等全新特性，可极大地减轻执行软件的负担，其中每一项特性通常都由一组寄存器来控制。

FTM 输入触发源可从比较器、ADC 或其他子模块来自动启动定时器功能。这些触发源在子模块集成期间可以各种方式链接，所以需要使用适用于当前 FTM 配置的触发源选项。

假设每个 FTM 中的初始化、输入时钟、初始和最终计数值都相同，则通过同步一个以上 FTM 模块（计数器一致增加），可以得到计数器更大的定时器。

所有主要的用户访问寄存器都经过缓冲，以缓解当前运行软件的负载。根据用户定义的数据，通过触发源各可用选项可以确定需要更新的寄存器。

## 26.1.2 特性

FTM 特性包括：

- FTM 时钟源是可选的
  - 时钟源可以是系统时钟、固定频率时钟或外部时钟
  - 固定频率时钟是一种附加时钟输入，允许在系统时钟以外选择一个片上时钟源
  - 选择外部时钟可将 FTM 时钟连接到芯片级输入引脚，从而能够将 FTM 计数器与片外时钟源同步
- 预分频器（1、2、4、8、16、32、64 或 128 分频）
- 16 位计数器
  - 可以是自由运行的计数器，也可以是采用初始和最终值的计数器
  - 计数可以是向上计数或者先向上后向下计数
- 每个通道均可配置为输入捕捉、输出比较或边沿对齐 PWM 模式
- 在输入捕捉模式下：
  - 捕捉可以在上升沿、下降沿或两个边沿同时发生
  - 可为某些通道选择一个输入滤波器
- 在输出比较模式下，可以对输出信号采取置位、清除或匹配切换操作
- 所有通道均可配置为中心对齐 PWM 模式
- 每对通道均可组合起来生成一个 PWM 信号，并且独立控制 PWM 信号的两个边沿

- FTM 通道可采用具有同等输出或互补输出的成对工作方式，或者作为独立的通道独立输出信号
- 死区插入可用于每一对互补通道
- 生成匹配触发器
- 软件控制 PWM 输出
- 对于全局故障控制最多有 4 个故障输入
- 每个通道的极性都是可配置的
- 按通道生成中断
- 当计数器溢出时生成中断
- 检测到故障条件时生成中断
- 同步加载写缓冲 FTM 寄存器
- 关键寄存器的写入保护
- 向下兼容 TPM
- 输入捕捉测试可检测固定为零和固定为一的状况
- 用于脉冲和周期宽度测量的双重边沿捕捉

### 26.1.3 工作模式

当 MCU 处于有效 Debug 模式时，FTM 会临时挂起所有计数，直到 MCU 返回正常用户工作模式。在 Stop 模式下，所有 FTM 输入时钟都将停止，因此，在时钟恢复前，FTM 会一直有效地禁用。在 Wait 模式下，FTM 继续正常工作。如果 FTM 无需生成实时基准或提供将 MCU 从 Wait 模式唤醒所需的中断源，则可在进入 Wait 模式之前通过禁用 FTM 功能省电。.

### 26.1.4 结构框图

FTM 为每个通道使用一个输入/输出(I/O)引脚，CH<sub>n</sub> (FTM 通道(n)) 中的 n 为通道编号(0–7)。

下图展示了 FTM 结构。FTM 模块的核心组件是一个初始值和终值可编程的 16 位计数器，该计数器可向上或者先向上后向下计数。

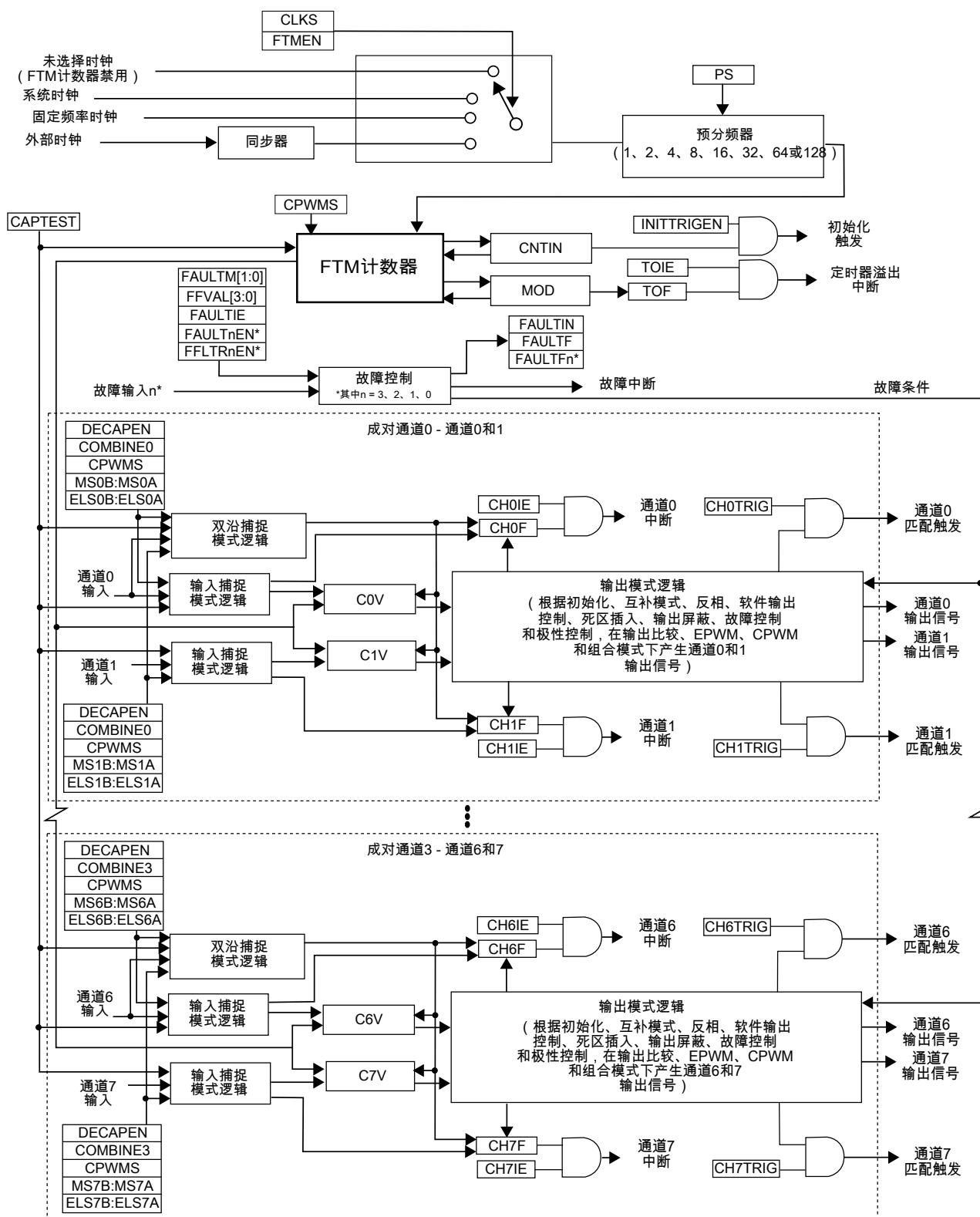


图 26-1. FTM 结构框图

## 26.2 FTM 信号说明

表 26-1 显示了用户可访问的 FTM 信号。。

表 26-1. FTM 信号说明

信号	说明	I/O	功能
EXTCLK	外部时钟。可选择 FTM 外部时钟用来驱动 FTM 计数器。	I	如果通过 SC 寄存器中的 CLKS[1:0]位选择了外部时钟输入信号，则该信号可用作 FTM 计数器时钟。该时钟信号不得超过系统时钟频率的 1/4。选择了外部时钟后，还会用到 FTM 计数器预分频器的选择和设置功能。
CHn	FTM 通道(n)，其中 n 可以是 7-0	I/O	每个 FTM 通道均可配置为输入或输出。每个通道（无论是输入还是输出）关联的方向是根据分配给该通道的模式选择的。
FAULTj	故障输入(j)，其中 j 可以是 3-0	I	故障输入信号用于控制 CHn 通道输出状态。如果检测到故障，FAULTj 信号变为有效，且通道输出进入安全状态。故障逻辑的行为方式由 MODE 寄存器中的 FAULTM[1:0]控制位和 COMBINEm 寄存器中的 FAULTEN 位定义。注意，由于 FAULTM[1:0]和 FAULTEN 控制位是为每一对通道定义的，因此每个 FAULTj 输入都有可能有选择地影响所有通道。因为有多个 FAULTj 输入 (FTM 模块最多有 4 个)，所以这些输入中的每一个都会由 FLTCTRL 寄存器中的 FAULTjEN 位激活。

## 26.3 存储器映像和寄存器定义

### 26.3.1 存储器映像

本节从较高层面概要介绍了 FTM 寄存器以及这些寄存器的映射方式。

FTM 中不可用功能的寄存器和比特仍然保留在存储器映像中以及复位值中，但不具备有效功能。

#### 注

FTMEN = 0 时，请勿在 CNTIN 寄存器到 PWMLOAD 寄存器这一区域内进行写入操作。

### 26.3.2 寄存器说明

访问保留地址会导致传输错误。用于不存在的通道的寄存器即视为保留寄存器。

## FTM 存储器映射

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4003_8000	状态和控制寄存器 (FTM0_SC)	32	R/W	0000_0000h	26.3.3/355
4003_8004	计数器寄存器 (FTM0_CNT)	32	R/W	0000_0000h	26.3.4/356
4003_8008	模数寄存器 (FTM0_MOD)	32	R/W	0000_0000h	26.3.5/357
4003_800C	通道(n)状态和控制寄存器 (FTM0_C0SC)	32	R/W	0000_0000h	26.3.6/358
4003_8010	通道(n)值寄存器 (FTM0_C0V)	32	R/W	0000_0000h	26.3.7/360
4003_8014	通道(n)状态和控制寄存器 (FTM0_C1SC)	32	R/W	0000_0000h	26.3.6/358
4003_8018	通道(n)值寄存器 (FTM0_C1V)	32	R/W	0000_0000h	26.3.7/360
4003_801C	通道(n)状态和控制寄存器 (FTM0_C2SC)	32	R/W	0000_0000h	26.3.6/358
4003_8020	通道(n)值寄存器 (FTM0_C2V)	32	R/W	0000_0000h	26.3.7/360
4003_8024	通道(n)状态和控制寄存器 (FTM0_C3SC)	32	R/W	0000_0000h	26.3.6/358
4003_8028	通道(n)值寄存器 (FTM0_C3V)	32	R/W	0000_0000h	26.3.7/360
4003_802C	通道(n)状态和控制寄存器 (FTM0_C4SC)	32	R/W	0000_0000h	26.3.6/358
4003_8030	通道(n)值寄存器 (FTM0_C4V)	32	R/W	0000_0000h	26.3.7/360
4003_8034	通道(n)状态和控制寄存器 (FTM0_C5SC)	32	R/W	0000_0000h	26.3.6/358
4003_8038	通道(n)值寄存器 (FTM0_C5V)	32	R/W	0000_0000h	26.3.7/360
4003_803C	通道(n)状态和控制寄存器 (FTM0_C6SC)	32	R/W	0000_0000h	26.3.6/358
4003_8040	通道(n)值寄存器 (FTM0_C6V)	32	R/W	0000_0000h	26.3.7/360
4003_8044	通道(n)状态和控制寄存器 (FTM0_C7SC)	32	R/W	0000_0000h	26.3.6/358
4003_8048	通道(n)值寄存器 (FTM0_C7V)	32	R/W	0000_0000h	26.3.7/360
4003_804C	计数器初始值寄存器 (FTM0_CNTIN)	32	R/W	0000_0000h	26.3.8/360
4003_8050	捕捉和比较状态寄存器 (FTM0_STATUS)	32	R/W	0000_0000h	26.3.9/361
4003_8054	特性模式选择寄存器 (FTM0_MODE)	32	R/W	0000_0004h	26.3.10/363
4003_8058	同步寄存器 (FTM0_SYNC)	32	R/W	0000_0000h	26.3.11/364
4003_805C	通道输出的初始状态寄存器 (FTM0_OUTINIT)	32	R/W	0000_0000h	26.3.12/366
4003_8060	输出屏蔽寄存器 (FTM0_OUTMASK)	32	R/W	0000_0000h	26.3.13/368
4003_8064	已连接通道功能寄存器 (FTM0_COMBINE)	32	R/W	0000_0000h	26.3.14/370
4003_8068	死区时间插入控制 (FTM0_DEADTIME)	32	R/W	0000_0000h	26.3.15/374
4003_806C	FTM 外部触发寄存器 (FTM0_EXTRIG)	32	R/W	0000_0000h	26.3.16/375
4003_8070	通道极性寄存器 (FTM0_POL)	32	R/W	0000_0000h	26.3.17/377
4003_8074	故障模式状态寄存器 (FTM0_FMS)	32	R/W	0000_0000h	26.3.18/380
4003_8078	输入捕捉滤波器控制寄存器 (FTM0_FILTER)	32	R/W	0000_0000h	26.3.19/382
4003_807C	故障控制寄存器 (FTM0_FLTCTRL)	32	R/W	0000_0000h	26.3.20/383
4003_8084	配置寄存器 (FTM0_CONF)	32	R/W	0000_0000h	26.3.21/385
4003_8088	FTM 故障输入极性寄存器 (FTM0_FLTPOL)	32	R/W	0000_0000h	26.3.22/386
4003_808C	同步配置寄存器 (FTM0_SYNCONF)	32	R/W	0000_0000h	26.3.23/387
4003_8090	FTM 反相控制寄存器 (FTM0_INVCTRL)	32	R/W	0000_0000h	26.3.24/389
4003_8094	FTM 软件输出控制寄存器 (FTM0_SWOCTRL)	32	R/W	0000_0000h	26.3.25/390
4003_8098	FTM PWM 加载寄存器 (FTM0_PWMLOAD)	32	R/W	0000_0000h	26.3.26/392
4003_9000	状态和控制寄存器 (FTM1_SC)	32	R/W	0000_0000h	26.3.3/355

下一页继续介绍此表...

## FTM 存储器映射 (继续)

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4003_9004	计数器寄存器 (FTM1_CNT)	32	R/W	0000_0000h	26.3.4/356
4003_9008	模数寄存器 (FTM1_MOD)	32	R/W	0000_0000h	26.3.5/357
4003_900C	通道(n)状态和控制寄存器 (FTM1_C0SC)	32	R/W	0000_0000h	26.3.6/358
4003_9010	通道(n)值寄存器 (FTM1_C0V)	32	R/W	0000_0000h	26.3.7/360
4003_9014	通道(n)状态和控制寄存器 (FTM1_C1SC)	32	R/W	0000_0000h	26.3.6/358
4003_9018	通道(n)值寄存器 (FTM1_C1V)	32	R/W	0000_0000h	26.3.7/360
4003_901C	通道(n)状态和控制寄存器 (FTM1_C2SC)	32	R/W	0000_0000h	26.3.6/358
4003_9020	通道(n)值寄存器 (FTM1_C2V)	32	R/W	0000_0000h	26.3.7/360
4003_9024	通道(n)状态和控制寄存器 (FTM1_C3SC)	32	R/W	0000_0000h	26.3.6/358
4003_9028	通道(n)值寄存器 (FTM1_C3V)	32	R/W	0000_0000h	26.3.7/360
4003_902C	通道(n)状态和控制寄存器 (FTM1_C4SC)	32	R/W	0000_0000h	26.3.6/358
4003_9030	通道(n)值寄存器 (FTM1_C4V)	32	R/W	0000_0000h	26.3.7/360
4003_9034	通道(n)状态和控制寄存器 (FTM1_C5SC)	32	R/W	0000_0000h	26.3.6/358
4003_9038	通道(n)值寄存器 (FTM1_C5V)	32	R/W	0000_0000h	26.3.7/360
4003_903C	通道(n)状态和控制寄存器 (FTM1_C6SC)	32	R/W	0000_0000h	26.3.6/358
4003_9040	通道(n)值寄存器 (FTM1_C6V)	32	R/W	0000_0000h	26.3.7/360
4003_9044	通道(n)状态和控制寄存器 (FTM1_C7SC)	32	R/W	0000_0000h	26.3.6/358
4003_9048	通道(n)值寄存器 (FTM1_C7V)	32	R/W	0000_0000h	26.3.7/360
4003_904C	计数器初始值寄存器 (FTM1_CNTIN)	32	R/W	0000_0000h	26.3.8/360
4003_9050	捕捉和比较状态寄存器 (FTM1_STATUS)	32	R/W	0000_0000h	26.3.9/361
4003_9054	特性模式选择寄存器 (FTM1_MODE)	32	R/W	0000_0004h	26.3.10/363
4003_9058	同步寄存器 (FTM1_SYNC)	32	R/W	0000_0000h	26.3.11/364
4003_905C	通道输出的初始状态寄存器 (FTM1_OUTINIT)	32	R/W	0000_0000h	26.3.12/366
4003_9060	输出屏蔽寄存器 (FTM1_OUTMASK)	32	R/W	0000_0000h	26.3.13/368
4003_9064	已连接通道功能寄存器 (FTM1_COMBINE)	32	R/W	0000_0000h	26.3.14/370
4003_9068	死区时间插入控制 (FTM1_DEADTIME)	32	R/W	0000_0000h	26.3.15/374
4003_906C	FTM 外部触发寄存器 (FTM1_EXTRIG)	32	R/W	0000_0000h	26.3.16/375
4003_9070	通道极性寄存器 (FTM1_POL)	32	R/W	0000_0000h	26.3.17/377
4003_9074	故障模式状态寄存器 (FTM1_FMS)	32	R/W	0000_0000h	26.3.18/380
4003_9078	输入捕捉滤波器控制寄存器 (FTM1_FILTER)	32	R/W	0000_0000h	26.3.19/382
4003_907C	故障控制寄存器 (FTM1_FLTCTRL)	32	R/W	0000_0000h	26.3.20/383
4003_9084	配置寄存器 (FTM1_CONF)	32	R/W	0000_0000h	26.3.21/385
4003_9088	FTM 故障输入极性寄存器 (FTM1_FLTPOL)	32	R/W	0000_0000h	26.3.22/386
4003_908C	同步配置寄存器 (FTM1_SYNCONF)	32	R/W	0000_0000h	26.3.23/387
4003_9090	FTM 反相控制寄存器 (FTM1_INVCTRL)	32	R/W	0000_0000h	26.3.24/389
4003_9094	FTM 软件输出控制寄存器 (FTM1_SWOCTRL)	32	R/W	0000_0000h	26.3.25/390
4003_9098	FTM PWM 加载寄存器 (FTM1_PWMLOAD)	32	R/W	0000_0000h	26.3.26/392
4003_A000	状态和控制寄存器 (FTM2_SC)	32	R/W	0000_0000h	26.3.3/355
4003_A004	计数器寄存器 (FTM2_CNT)	32	R/W	0000_0000h	26.3.4/356

下一页继续介绍此表...

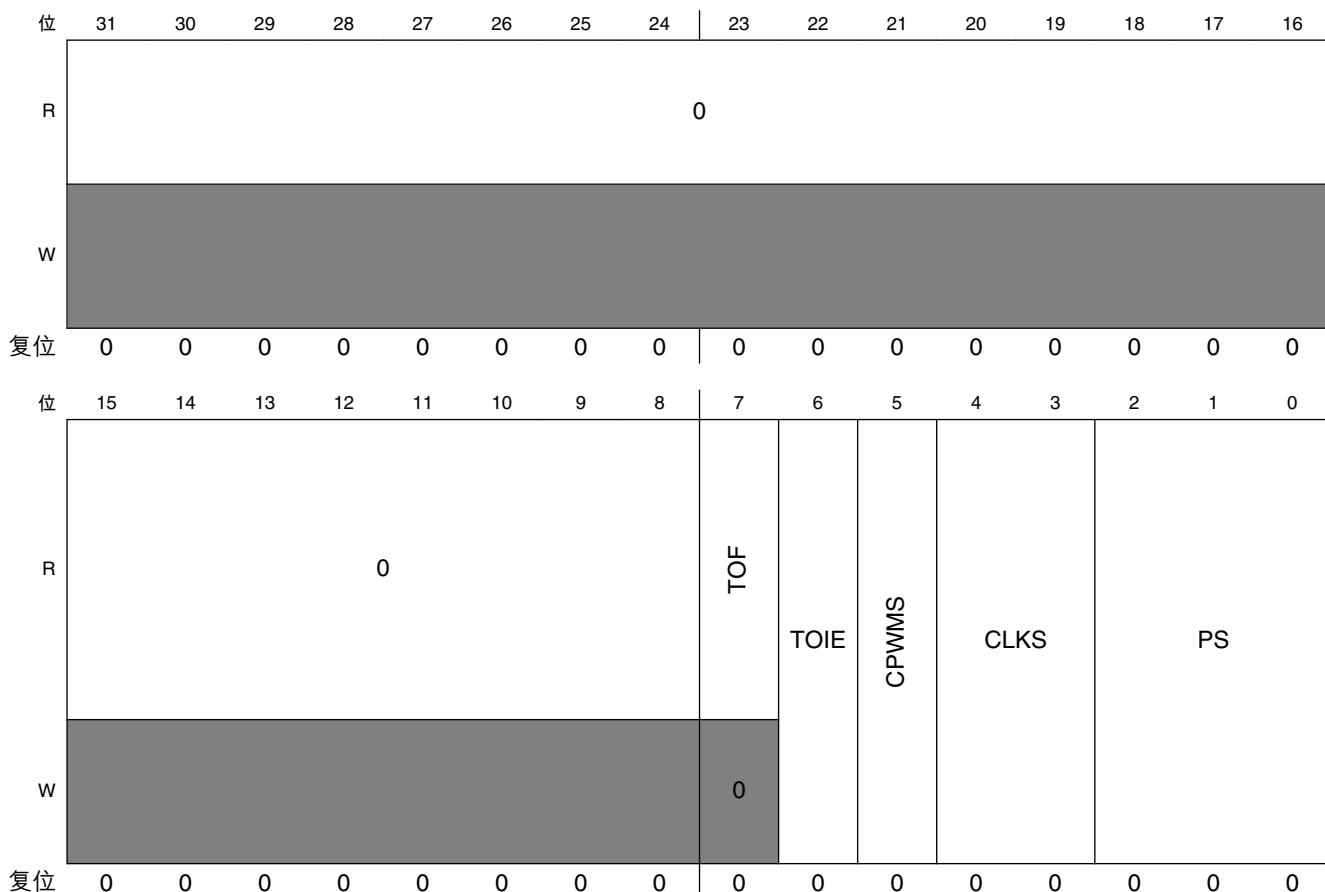
## FTM 存储器映射 (继续)

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4003_A008	模数寄存器 (FTM2_MOD)	32	R/W	0000_0000h	26.3.5/357
4003_A00C	通道(n)状态和控制寄存器 (FTM2_C0SC)	32	R/W	0000_0000h	26.3.6/358
4003_A010	通道(n)值寄存器 (FTM2_C0V)	32	R/W	0000_0000h	26.3.7/360
4003_A014	通道(n)状态和控制寄存器 (FTM2_C1SC)	32	R/W	0000_0000h	26.3.6/358
4003_A018	通道(n)值寄存器 (FTM2_C1V)	32	R/W	0000_0000h	26.3.7/360
4003_A01C	通道(n)状态和控制寄存器 (FTM2_C2SC)	32	R/W	0000_0000h	26.3.6/358
4003_A020	通道(n)值寄存器 (FTM2_C2V)	32	R/W	0000_0000h	26.3.7/360
4003_A024	通道(n)状态和控制寄存器 (FTM2_C3SC)	32	R/W	0000_0000h	26.3.6/358
4003_A028	通道(n)值寄存器 (FTM2_C3V)	32	R/W	0000_0000h	26.3.7/360
4003_A02C	通道(n)状态和控制寄存器 (FTM2_C4SC)	32	R/W	0000_0000h	26.3.6/358
4003_A030	通道(n)值寄存器 (FTM2_C4V)	32	R/W	0000_0000h	26.3.7/360
4003_A034	通道(n)状态和控制寄存器 (FTM2_C5SC)	32	R/W	0000_0000h	26.3.6/358
4003_A038	通道(n)值寄存器 (FTM2_C5V)	32	R/W	0000_0000h	26.3.7/360
4003_A03C	通道(n)状态和控制寄存器 (FTM2_C6SC)	32	R/W	0000_0000h	26.3.6/358
4003_A040	通道(n)值寄存器 (FTM2_C6V)	32	R/W	0000_0000h	26.3.7/360
4003_A044	通道(n)状态和控制寄存器 (FTM2_C7SC)	32	R/W	0000_0000h	26.3.6/358
4003_A048	通道(n)值寄存器 (FTM2_C7V)	32	R/W	0000_0000h	26.3.7/360
4003_A04C	计数器初始值寄存器 (FTM2_CNTIN)	32	R/W	0000_0000h	26.3.8/360
4003_A050	捕捉和比较状态寄存器 (FTM2_STATUS)	32	R/W	0000_0000h	26.3.9/361
4003_A054	特性模式选择寄存器 (FTM2_MODE)	32	R/W	0000_0004h	26.3.10/363
4003_A058	同步寄存器 (FTM2_SYNC)	32	R/W	0000_0000h	26.3.11/364
4003_A05C	通道输出的初始状态寄存器 (FTM2_OUTINIT)	32	R/W	0000_0000h	26.3.12/366
4003_A060	输出屏蔽寄存器 (FTM2_OUTMASK)	32	R/W	0000_0000h	26.3.13/368
4003_A064	已连接通道功能寄存器 (FTM2_COMBINE)	32	R/W	0000_0000h	26.3.14/370
4003_A068	死区时间插入控制 (FTM2_DEADTIME)	32	R/W	0000_0000h	26.3.15/374
4003_A06C	FTM 外部触发寄存器 (FTM2_EXTRIG)	32	R/W	0000_0000h	26.3.16/375
4003_A070	通道极性寄存器 (FTM2_POL)	32	R/W	0000_0000h	26.3.17/377
4003_A074	故障模式状态寄存器 (FTM2_FMS)	32	R/W	0000_0000h	26.3.18/380
4003_A078	输入捕捉滤波器控制寄存器 (FTM2_FILTER)	32	R/W	0000_0000h	26.3.19/382
4003_A07C	故障控制寄存器 (FTM2_FLCTRL)	32	R/W	0000_0000h	26.3.20/383
4003_A084	配置寄存器 (FTM2_CONF)	32	R/W	0000_0000h	26.3.21/385
4003_A088	FTM 故障输入极性寄存器 (FTM2_FLTPOL)	32	R/W	0000_0000h	26.3.22/386
4003_A08C	同步配置寄存器 (FTM2_SYNCONF)	32	R/W	0000_0000h	26.3.23/387
4003_A090	FTM 反相控制寄存器 (FTM2_INVCTRL)	32	R/W	0000_0000h	26.3.24/389
4003_A094	FTM 软件输出控制寄存器 (FTM2_SWOCTRL)	32	R/W	0000_0000h	26.3.25/390
4003_A098	FTM PWM 加载寄存器 (FTM2_PWMLOAD)	32	R/W	0000_0000h	26.3.26/392

### 26.3.3 状态和控制寄存器 (FTMx\_SC)

状态和控制寄存器含有溢出状态标志和控制位，用来配置中断使能、FTM、时钟源和预分频系数。这些控制与模块内的全部通道有关。

地址: 基址 基准 + 0h 偏移



FTMx\_SC 字段描述

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 TOF	定时器溢出标志  当 FTM 计数器达到 MOD 寄存器中的值时，通过硬件置位。在 TOF 置位的情况下读取 SC 寄存器，然后向 TOF 位写入 0，可清零 TOF 位。将 1 写入 TOF 无效。  若在读取和写入操作之间发生另一次 FTM 溢出，则写操作无效；因此，TOF 保持置位状态，表示发生了溢出。在此情况下，TOF 中断请求不会因为之前的 TOF 清零序列而丢失。  0 FTM 计数器未溢出。 1 FTM 计数器已溢出。
6 TOIE	定时器溢出中断使能

下一页继续介绍此表...

**FTMx\_SC 字段描述 (继续)**

字段	描述
	使能 FTM 溢出中断。 0 禁用 TOF 中断。使用软件轮询。 1 使能 TOF 中断。TOF 等于 1 时，产生中断。
5 CPWMS	中心对齐 PWM 选择 选择 CPWM 模式。该模式配置 FTM 在先增后减计数模式中的操作。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 FTM 计数器可工作在向上计数模式中。 1 FTM 计数器可工作在先增后减计数模式中。
4–3 CLKS	时钟源选择 从三个 FTM 计数器时钟源选择。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 00 未选定时钟。该设置生效后可禁用 FTM 计数器。 01 系统时钟 10 固定频率时钟 11 外部时钟
PS	预分频系数选择 从 8 个分频系数中选择，用于 CLKS 所选择的时钟源。新的预分频系数将会在新数值更新至寄存器位后，于下一个系统时钟周期影响时钟源。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 000 1 分频 001 2 分频 010 4 分频 011 8 分频 100 16 分频 101 32 分频 110 64 分频 111 128 分频

**26.3.4 计数器寄存器 (FTMx\_CNT)**

CNT 寄存器含有 FTM 计数器值。

复位操作可清零 CNT 寄存器。向 COUNT 写入任何值都将用初始值 CNTIN 更新计数器。

激活 Debug 时，FTM 计数器冻结。这是您可能读取的值。

地址: 基址 基准 + 4h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																0																
W																												COUNT				
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

#### FTMx\_CNT 字段描述

字段	描述
31–16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
COUNT	计数器值

### 26.3.5 模数寄存器 (FTMx\_MOD)

模数寄存器含有 FTM 计数器的模数值。FTM 计数器到达模数值后, 溢出标志(TOF)在下一个时钟变成置位状态, 且 FTM 计数器的下一个数值取决于选定的计数方式; 参见[计数器寄存器](#)。

写入 MOD 寄存器可将数值锁存到缓冲区中。将根据[通过写缓存更新的寄存器](#)用 MOD 寄存器的写缓冲器值更新此 MOD 寄存器。

若 FTMEN = 0, 此写入一致性机制便可手动复位, 方法是将 Debug 是否激活写入 SC 寄存器。

在写入到 MOD 寄存器之前, 通过写 CNT 以初始化 FTM 计数器, 从而避免对计数器首次发生溢出的时间造成混淆。

地址: 基址 基准 + 8h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R																Reserved																	
W																												MOD					
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

#### FTMx\_MOD 字段描述

字段	描述
31–16 Reserved	此字段为保留字段。
MOD	模数值

### 26.3.6 通道(n)状态和控制寄存器 (FTMx\_CnSC)

CnSC 含有通道中断状态标志和控制位，用来配置中断使能、通道和引脚功能。

表 26-2. 模式、边沿和电平选择

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	模式	配置
X	X	X	XX	00	未用于 FTM 的引脚 — 将通道引脚恢复为通用 I/O 或其他外围控件	
0	0	0	00	01	输入捕捉	仅在上升沿捕捉
				10		仅在下降沿捕捉
				11		在上升沿或下降沿上捕捉
			01	01	输出比较	匹配时切换输出
				10		匹配时清零输出
				11		匹配时置位输出
			1X	10	边沿对齐 PWM	高电平真脉冲(匹配时清零输出)
				X1		低真脉冲(匹配时置位输出)
			1	10	中心对齐 PWM	高真脉冲(向上匹配时清零输出)
				X1		低真脉冲(向上匹配时置位输出)
			0	10	组合 PWM	高真脉冲 (通道(n)匹配时置位, 通道(n+1)匹配时清零)
				X1		低真脉冲 (通道(n)匹配时清零, 通道(n+1)匹配时置位)
1	0	0	X0	参见下表( <a href="#">表 26-3</a> )。	双沿捕捉	单次捕捉模式
			X1			连续捕捉模式

表 26-3. 双边沿捕捉模式 - 边沿极性选择

ELSnB	ELSnA	通道端口使能	已检测边沿
0	0	禁用	无边沿
0	1	使能	上升沿
1	0	使能	下降沿
1	1	使能	上升沿和下降沿

地址: 基址 + Ch 偏移 + (8d × i), 其中 i=0d 到 7d

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									CHF	CHIE	MSB	MSA	ELSB	ELSA	0	0	
W									0								
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

### FTMx\_CnSC 字段描述

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 CHF	通道标志  通道上有事件发生时，通过硬件置位。在 CHnF 置位的情况下读取 CSC 寄存器，然后向 CHF 位写入 0，可清零 CHF。将 1 写入 CHF 无效。  若在读取和写入操作之间发生另一次事件，则写操作无效；因此，CHF 保持置位状态，表示发生了一次事件。在此情况下，CHF 中断请求不会因为之前的 CHF 清零序列而丢失。  0 未发生通道事件。 1 发生通道事件。
6 CHIE	通道中断使能  使能通道中断。  0 禁用通道中断。使用软件轮询。 1 使能通道中断。
5 MSB	通道模式选择  用来在通道逻辑中作进一步选择。其功能取决于通道模式。参见表 26-2。  该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。
4 MSA	通道模式选择  用来在通道逻辑中作进一步选择。其功能取决于通道模式。参见表 26-2。  该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。
3 ELSB	边沿或电平选择  ELSB 和 ELSA 功能取决于通道模式。参见表 26-2。  该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。
2 ELSA	边沿或电平选择  ELSB 和 ELSA 功能取决于通道模式。参见表 26-2。  该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。
1 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

### 26.3.7 通道(n)值寄存器 (FTMx\_CnV)

这些寄存器包含输入模式下捕获的 FTM 计数器值或输出模式下的匹配值。

在输入捕捉、捕捉测试和双边沿捕捉模式下，任何对 CnV 寄存器的写入操作都将被忽略。

在输出模式下，向 CnV 寄存器写入值时，会将此值锁存到缓冲器中。将根据[通过写缓存更新的寄存器](#)用 CnV 寄存器的写入缓冲器值更新此 CnV 寄存器。

若 FTMEN = 0，此写入一致性机制便可手动复位，方法是将 Debug 模式是否激活写入 CnSC 寄存器。

地址: 基址 + 10h 偏移 + (8d × i), 其中 i=0d 到 7d

位	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R	0	VAL
W		
复位	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

**FTMx\_CnV** 字段描述

字段	描述
31–16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
VAL	通道值  输入模式下捕捉的 FTM 计数器值或用于输出模式的匹配值

### 26.3.8 计数器初始值寄存器 (FTMx\_CNTIN)

计数器初始值寄存器包含用于 FTM 计数器的初始数值。

写入 CNTIN 寄存器可将数值锁存到缓冲区中。CNTIN 寄存器根据[通过写缓存更新的寄存器](#)所示，更新为其写入缓冲区的内容。

初始选定 FTM 时钟时，通过向 CLKS 位写入非零值，会让 FTM 计数器以数值 0x0000 起始。为了避免发生这种情况，在首次写入以选择 FTM 时钟前，向 CNTIN 寄存器写入新数值，然后向 CNT 寄存器写入任意值可初始化 FTM 计数器。

地址: 基址 基准 + 4Ch 偏移

位	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R	Reserved	INIT
W		
复位	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

**FTMx\_CNTIN 字段描述**

字段	描述
31–16 Reserved	此字段为保留字段。
INIT	FTM 计数器初始值

**26.3.9 捕捉和比较状态寄存器 (FTMx\_STATUS)**

状态寄存器含有 CnSC 中状态标志位 CHnF 的副本，可用于各 FTM 通道，为软件提供方便。

状态寄存器中每一个 CHnF 位都是 CnSC 中 CHnF 位的镜像。对状态寄存器进行一次读取，即可检查所有 CHnF 位。读取状态寄存器，然后向其写入 0x00，即可清零所有 CHnF 位。

通道上发生事件时，硬件将置位各通道标志。在 CHnF 置位的情况下，读取状态寄存器，然后向 CHnF 位写入 0，可清零 CHnF。将 1 写入 CHnF 无效。

若在读取和写入操作之间发生另一次事件，则写操作无效；因此，CHnF 保持置位状态，表示发生了一次事件。在这种情况下，由于之前 CHnF 的清除序列，CHnF 中断请求将不会丢失。

地址: 基址 基准 + 50h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									0							
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W									0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**FTMx\_STATUS 字段描述**

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 CH7F	通道 7 标志  参见寄存器说明。  0 未发生通道事件。 1 发生通道事件。
6 CH6F	通道 6 标志  参见寄存器说明。  0 未发生通道事件。 1 发生通道事件。
5 CH5F	通道 5 标志  参见寄存器说明。  0 未发生通道事件。 1 发生通道事件。
4 CH4F	通道 4 标志  参见寄存器说明。  0 未发生通道事件。 1 发生通道事件。
3 CH3F	通道 3 标志  参见寄存器说明。  0 未发生通道事件。 1 发生通道事件。
2 CH2F	通道 2 标志  参见寄存器说明。  0 未发生通道事件。 1 发生通道事件。
1 CH1F	通道 1 标志  参见寄存器说明。  0 未发生通道事件。 1 发生通道事件。
0 CH0F	通道 0 标志  参见寄存器说明  0 未发生通道事件。 1 发生通道事件。

### 26.3.10 特性模式选择寄存器 (FTMx\_MODE)

该寄存器含有 FTM 特有的全局使能位，以及用来配置的控制位：

- 故障控制模式和中断
- 捕捉测试模式
- PWM 同步
- 写保护
- 通道输出初始化

这些控制与模块内的全部通道有关。

地址: 基址 基准 + 54h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									0							
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									FAULTIE		FAULTM	CAPTEST	PWM_SYNC	WPDIS	INIT	FTMEN
W									0	0	0	0	0	1	0	0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

FTMx\_MODE 字段描述

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 FAULTIE	故障中断使能  FTM 检测到故障，且 FTM 故障控制使能时，使能中断生成。  0 故障控制中断禁用。 1 故障控制中断使能。
6–5 FAULTM	故障控制模式  定义 FTM 故障控制模式。  该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。  00 所有通道禁用故障控制。 01 仅针对偶数通道（通道 0、2、4 和 6）使能故障控制，所选模式为手动故障清零。 10 针对所有通道使能故障控制，所选模式为手动故障清零。 11 针对所有通道使能故障控制，所选模式为自动故障清零。

下一页继续介绍此表...

**FTMx\_MODE 字段描述 (继续)**

字段	描述
4 CAPTEST	<p>捕捉测试模式使能 使能捕捉测试模式。</p> <p>该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 捕捉测试模式禁用。 1 捕捉测试模式使能。</p>
3 PWMSYNC	<p>PWM 同步模式 选择可用于 MOD、CnV、OUTMASK 和 FTM 计数器同步的触发。参见 <a href="#">PWM 同步</a>。当 SYNCMODE 为 0 时，PWMSYNC 位进行同步配置。</p> <p>0 无限制。选择可用于 MOD、CnV、OUTMASK 和 FTM 计数器同步的软件和硬件触发。 1 软件触发只能用于 MOD 和 CnV 同步，硬件触发只能用于 OUTMASK 和 FTM 计数器同步。</p>
2 WPDIS	<p>写保护禁用 写保护使能时(WPDIS = 0)，无法向写保护位写入内容。写保护禁用后(WPDIS = 1)，可以写入写保护位。WPDIS 位与 WPEN 位相反。向 WPEN 写入 1 可清零 WPDIS。WPDIS 置位：WPEN 位读取为 1，然后向 WPDIS 写入 1。向 WPDIS 中写入 0 无效。</p> <p>0 写保护使能。 1 写保护禁用。</p>
1 INIT	<p>初始化通道输出 INIT 位写入 1 后，通道输出根据 OUTINIT 寄存器中的对应位状态进行初始化。将 0 写入 INIT 位无效。 INIT 位始终读取为 0。</p>
0 FTMEN	<p>FTM 使能 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 TPM 兼容性。自由运行计数器并与 TPM 同步兼容。 1 自由运行计数器并且同步与 TPM 行为不同。</p>

**26.3.11 同步寄存器 (FTMx\_SYNC)**

该寄存器配置 PWM 同步性能。

同步事件可采用 MOD、CV 和 OUTMASK 寄存器的写入缓冲区数值和 FTM 计数器初始化数值执行同步更新。

**注**

当 SYNCMODE = 0 时，同时使用软件触发 (SWSYNC 位) 和硬件触发 (TRIG0、TRIG1 和 TRIG2 位) 可能会产生冲突。同一时间应当仅使用硬件或软件触发 (而非两者同时使用)，否则可能产生无法预测的结果。

选择加载点 (CNTMAX 和 CNTMIN 位) 以便在全部使能通道中同时提供 MOD、CNTIN 和 CnV 寄存器更新。使用加载点选择配合 SYNCMODE = 0 和硬件触发选择 (TRIG0、TRIG1 或 TRIG2 位) 可能产生不可预测的结果。

同步事件选择还取决于 PWMSYNC (MODE 寄存器) 和 SYNCMODE (SYNCONF 寄存器) 位。参见 [PWM 同步](#)。

地址: 基址 基准 + 58h 偏移

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									SWSYNC	TRIG2	TRIG1	TRIG0	SYNCHOM	REINIT	CNTMAX	CNTMIN	
W									0	0	0	0	0	0	0	0	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

#### FTMx\_SYNC 字段描述

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 SWSYNC	PWM 同步软件触发  选择软件触发作为 PWM 同步触发。向 SWSYNC 位写入 1 时，发生软件触发。  0 软件触发未选定。 1 软件触发已选定。
6 TRIG2	PWM 同步硬件触发 2  使能 PWM 同步的硬件触发 2。在触发 2 输入信号中检测到上升沿时，产生硬件触发 2。  0 触发禁用。 1 触发使能。
5 TRIG1	PWM 同步硬件触发 1  使能 PWM 同步的硬件触发 1。在触发 1 输入信号中检测到上升沿时，产生硬件触发 1。  0 触发禁用。 1 触发使能。
4 TRIG0	PWM 同步硬件触发 0  使能 PWM 同步的硬件触发 0。在触发 0 输入信号中检测到上升沿时，产生硬件触发 0。

下一页继续介绍此表...

**FTMx\_SYNC 字段描述 (继续)**

字段	描述
	0 触发禁用。 1 触发使能。
3 SYNCHOM	输出屏蔽同步  选择 OUTMASK 寄存器何时以其缓冲区的数值更新。  0 OUTMASK 寄存器将在所有系统时钟的上升沿以其缓冲区的数值进行更新。 1 OUTMASK 寄存器仅通过 PWM 同步以其缓冲区值进行更新。
2 REINIT	通过同步对 FTM 计数器进行重新初始化( <a href="#">FTM 计数器同步</a> )  检测到用于同步的特定触发时，确定 FTM 计数器是否重新初始化。当 SYNCMODE 为 0 时，REINIT 位进行同步配置。  0 FTM 计数器继续正常计数。 1 当检测到特定的触发时，FTM 计数器以其初始值更新。
1 CNTMAX	最大加载点使能  选择 PWM 同步的最大加载点。参见 <a href="#">边界周期和加载点</a> 。若 CNTMAX 为 1，则选定的加载点为 FTM 计数器达到其最大值 (MOD 寄存器) 的时刻。  0 最大加载点禁用。 1 最大加载点使能。
0 CNTMIN	最小加载点使能  选择 PWM 同步的最小加载点。参见 <a href="#">边界周期和加载点</a> 。若 CNTMIN 为 1，则选定的加载点为 FTM 计数器达到其最小值 (CNTIN 寄存器) 的时刻。  0 最小加载点禁用。 1 最小加载点使能。

**26.3.12 通道输出的初始状态寄存器 (FTMx\_OUTINIT)**

地址: 基址 基准 + 5Ch 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									0							
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									CH7OI	CH6OI	CH5OI	CH4OI	CH3OI	CH2OI	CH1OI	CH0OI
W									0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**FTMx\_OUTINIT 字段描述**

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 CH7OI	通道 7 输出初始值  进行初始化时，选择强制进入通道输出的数值。  0 初始值为 0。 1 初始值为 1。
6 CH6OI	通道 6 输出初始值  进行初始化时，选择强制进入通道输出的数值。  0 初始值为 0。 1 初始值为 1。
5 CH5OI	通道 5 输出初始值  进行初始化时，选择强制进入通道输出的数值。  0 初始值为 0。 1 初始值为 1。
4 CH4OI	通道 4 输出初始值  进行初始化时，选择强制进入通道输出的数值。  0 初始值为 0。 1 初始值为 1。
3 CH3OI	通道 3 输出初始值  进行初始化时，选择强制进入通道输出的数值。  0 初始值为 0。 1 初始值为 1。
2 CH2OI	通道 2 输出初始值  进行初始化时，选择强制进入通道输出的数值。  0 初始值为 0。 1 初始值为 1。
1 CH1OI	通道 1 输出初始值  进行初始化时，选择强制进入通道输出的数值。  0 初始值为 0。 1 初始值为 1。
0 CH0OI	通道 0 输出初始值  进行初始化时，选择强制进入通道输出的数值。  0 初始值为 0。 1 初始值为 1。

### 26.3.13 输出屏蔽寄存器 (FTMx\_OUTMASK)

该寄存器为每个 FTM 通道提供屏蔽。通道屏蔽确定其输出是否进行响应；也就是说，当发生匹配时，它是否屏蔽。该特性可用于 BLDC 控制；这类应用在特定时间内，电机上存在 PWM 信号，以提供电子换向。

对 OUTMASK 寄存器采取的任意写操作都会将数值保存在其写入缓冲区中。寄存器根据 [PWM 同步](#) 所示，更新为其写入缓冲区的内容。

地址: 基址 基准 + 60h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
R									0									
W																		
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R									0		CH7OM	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM	CH0OM
W											0	0	0	0	0	0	0	
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

FTMx\_OUTMASK 字段描述

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 CH7OM	通道 7 输出屏蔽  定义通道输出为屏蔽或未屏蔽。  0 通道输出未屏蔽。它连续正常工作。 1 通道输出已屏蔽。它被强制处于不活跃状态。
6 CH6OM	通道 6 输出屏蔽  定义通道输出为屏蔽或未屏蔽。  0 通道输出未屏蔽。它连续正常工作。 1 通道输出已屏蔽。它被强制处于不活跃状态。
5 CH5OM	通道 5 输出屏蔽  定义通道输出为屏蔽或未屏蔽。  0 通道输出未屏蔽。它连续正常工作。 1 通道输出已屏蔽。它被强制处于不活跃状态。
4 CH4OM	通道 4 输出屏蔽

下一页继续介绍此表...

**FTMx\_OUTMASK 字段描述 (继续)**

字段	描述
	定义通道输出为屏蔽或未屏蔽。 0 通道输出未屏蔽。它连续正常工作。 1 通道输出已屏蔽。它被强制处于不活跃状态。
3 CH3OM	通道 3 输出屏蔽 定义通道输出为屏蔽或未屏蔽。 0 通道输出未屏蔽。它连续正常工作。 1 通道输出已屏蔽。它被强制处于不活跃状态。
2 CH2OM	通道 2 输出屏蔽 定义通道输出为屏蔽或未屏蔽。 0 通道输出未屏蔽。它连续正常工作。 1 通道输出已屏蔽。它被强制处于不活跃状态。
1 CH1OM	通道 1 输出屏蔽 定义通道输出为屏蔽或未屏蔽。 0 通道输出未屏蔽。它连续正常工作。 1 通道输出已屏蔽。它被强制处于不活跃状态。
0 CH0OM	通道 0 输出屏蔽 定义通道输出为屏蔽或未屏蔽。 0 通道输出未屏蔽。它连续正常工作。 1 通道输出已屏蔽。它被强制处于不活跃状态。

### 26.3.14 已连接通道功能寄存器 (FTMx\_COMBINE)

该寄存器包含控制位，可配置故障控制、同步、死区时间插入、双沿捕捉模式、互补和组合模式，用于通道对(n)和(n+1)，其中 n 等于 0、2、4 和 6。

地址: 基址 基准 + 64h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	FAULTEN3	SYNCEN3	DTEN3	DECAP3	DECAPEN3	COMP3	COMBINE3	0	FAULTEN2	SYNCEN2	DTEN2	DECAP2	DECAPEN2	COMP2	COMBINE2
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FAULTEN1	SYNCEN1	DTEN1	DECAP1	DECAPEN1	COMP1	COMBINE1	0	FAULTENO	SYNCENO	DTENO	DECAP0	DECAPENO	COMP0	COMBINE0
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx\_COMBINE 字段描述

字段	描述
31 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
30 FAULTEN3	故障控制使能( $n = 6$ ) 使能通道(n)和(n+1)中的故障控制。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用该通道对中的故障控制。 1 使能该通道对中的故障控制。
29 SYNCEN3	同步使能( $n = 6$ ) 使能寄存器 C(n)V 和 C(n+1)V 的 PWM 同步。 0 禁用该通道对中的 PWM 同步。 1 使能该通道对中的 PWM 同步。
28 DTEN3	死区时间使能( $n = 6$ ) 使能通道(n)和(n+1)中的死区时间插入。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用该通道对中的死区时间插入。 1 使能该通道对中的死区时间插入。

下一页继续介绍此表...

## FTMx\_COMBINE 字段描述 (继续)

字段	描述
27 DECAP3	<p>双沿捕捉模式捕捉(<math>n = 6</math>)</p> <p>使能根据通道(<math>n</math>)输入事件捕捉 FTM 计数器值的功能和双沿捕捉位配置。</p> <p>仅在 DECAPEN = 1 时应用此字段。</p> <p>如果选定了双边沿捕捉 — 一次性模式，且执行了通道 (<math>n+1</math>) 事件的捕捉时，将由硬件自动清除 DECAP 位。</p> <p>0 双沿捕捉未激活。 1 双沿捕捉已激活。</p>
26 DECAPEN3	<p>双沿捕捉模式使能(<math>n = 6</math>)</p> <p>使能通道(<math>n</math>)和(<math>n+1</math>)的双沿捕捉模式。此位将根据 表 26-2 在双边沿捕捉模式中重新配置 MSnA、ELSnB:ELSnA 和 ELS(<math>n+1</math>)B:ELS(<math>n+1</math>)A 位的功能。</p> <p>此字段受写入保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 该通道对中的双沿捕捉模式禁用。 1 该通道对中的双沿捕捉模式使能。</p>
25 COMP3	<p>通道(<math>n</math>)互补(<math>n = 6</math>)</p> <p>使能组合通道的互补模式。在互补模式下，通道(<math>n+1</math>)输出与通道(<math>n</math>)输出相反。</p> <p>该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 通道(<math>n+1</math>)输出与通道(<math>n</math>)输出相同。 1 通道(<math>n+1</math>)输出与通道(<math>n</math>)输出互补。</p>
24 COMBINE3	<p>组合通道(<math>n = 6</math>)</p> <p>使能通道(<math>n</math>)和(<math>n+1</math>)的组合功能。</p> <p>该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 通道(<math>n</math>)和(<math>n+1</math>)独立。 1 通道(<math>n</math>)和(<math>n+1</math>)组合。</p>
23 保留	<p>此字段为保留字段。</p> <p>此只读字段为保留字段且值始终为 0。</p>
22 FAULTEN2	<p>故障控制使能(<math>n = 4</math>)</p> <p>使能通道(<math>n</math>)和(<math>n+1</math>)中的故障控制。</p> <p>该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 禁用该通道对中的故障控制。 1 使能该通道对中的故障控制。</p>
21 SYNCEN2	<p>同步使能(<math>n = 4</math>)</p> <p>使能寄存器 C(<math>n</math>)V 和 C(<math>n+1</math>)V 的 PWM 同步。</p> <p>0 禁用该通道对中的 PWM 同步。 1 使能该通道对中的 PWM 同步。</p>
20 DTEN2	<p>死区时间使能(<math>n = 4</math>)</p> <p>使能通道(<math>n</math>)和(<math>n+1</math>)中的死区时间插入。</p> <p>该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p>

下一页继续介绍此表...

**FTMx\_COMBINE 字段描述 (继续)**

字段	描述
	<p>0 禁用该通道对中的死区时间插入。 1 使能该通道对中的死区时间插入。</p>
19 DECAP2	<p>双沿捕捉模式捕捉(<math>n = 4</math>) 使能根据通道(<math>n</math>)输入事件捕捉 FTM 计数器值的功能和双沿捕捉位配置。 仅在 DECAPEN = 1 时应用此字段。 如果选定了双边沿捕捉 — 一次性模式，且执行了通道 (<math>n+1</math>) 事件的捕捉时，将由硬件自动清除 DECAP 位。</p> <p>0 双沿捕捉未激活。 1 双沿捕捉已激活。</p>
18 DECAPEN2	<p>双沿捕捉模式使能(<math>n = 4</math>) 使能通道(<math>n</math>)和(<math>n+1</math>)的双沿捕捉模式。此位将根据 表 26-2 在双边沿捕捉模式中重新配置 MSnA、ELSnB:ELSnA 和 ELS(<math>n+1</math>)B:ELS(<math>n+1</math>)A 位的功能。 此字段受写入保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 该通道对中的双沿捕捉模式禁用。 1 该通道对中的双沿捕捉模式使能。</p>
17 COMP2	<p>通道(<math>n</math>)互补(<math>n = 4</math>) 使能组合通道的互补模式。在互补模式下，通道(<math>n+1</math>)输出与通道(<math>n</math>)输出相反。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 通道(<math>n+1</math>)输出与通道(<math>n</math>)输出相同。 1 通道(<math>n+1</math>)输出与通道(<math>n</math>)输出互补。</p>
16 COMBINE2	<p>组合通道(<math>n = 4</math>) 使能通道(<math>n</math>)和(<math>n+1</math>)的组合功能。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 通道(<math>n</math>)和(<math>n+1</math>)独立。 1 通道(<math>n</math>)和(<math>n+1</math>)组合。</p>
15 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
14 FAULTEN1	<p>故障控制使能(<math>n = 2</math>) 使能通道(<math>n</math>)和(<math>n+1</math>)中的故障控制。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 禁用该通道对中的故障控制。 1 使能该通道对中的故障控制。</p>
13 SYNCEN1	<p>同步使能(<math>n = 2</math>) 使能寄存器 C(<math>n</math>)V 和 C(<math>n+1</math>)V 的 PWM 同步。</p> <p>0 禁用该通道对中的 PWM 同步。 1 使能该通道对中的 PWM 同步。</p>
12 DTEN1	<p>死区时间使能(<math>n = 2</math>) 使能通道(<math>n</math>)和(<math>n+1</math>)中的死区时间插入。</p>

下一页继续介绍此表...

## FTMx\_COMBINE 字段描述 (继续)

字段	描述
	<p>该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 禁用该通道对中的死区时间插入。 1 使能该通道对中的死区时间插入。</p>
11 DECAP1	<p>双沿捕捉模式捕捉(<math>n = 2</math>)</p> <p>使能根据通道(<math>n</math>)输入事件捕捉 FTM 计数器值的功能和双沿捕捉位配置。</p> <p>仅在 DECAPEN = 1 时应用此字段。</p> <p>如果选定了双边沿捕捉 — 一次性模式，且执行了通道 (<math>n+1</math>) 事件的捕捉时，将由硬件自动清除 DECAP 位。</p> <p>0 双沿捕捉未激活。 1 双沿捕捉已激活。</p>
10 DECAPEN1	<p>双沿捕捉模式使能(<math>n = 2</math>)</p> <p>使能通道(<math>n</math>)和(<math>n+1</math>)的双沿捕捉模式。此位将根据 表 26-2 在双边沿捕捉模式中重新配置 MSnA、ELSnB:ELSnA 和 ELS(<math>n+1</math>)B:ELS(<math>n+1</math>)A 位的功能。</p> <p>此字段受写入保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 该通道对中的双沿捕捉模式禁用。 1 该通道对中的双沿捕捉模式使能。</p>
9 COMP1	<p>通道(<math>n</math>)互补(<math>n = 2</math>)</p> <p>使能组合通道的互补模式。在互补模式下，通道(<math>n+1</math>)输出与通道(<math>n</math>)输出相反。</p> <p>该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 通道(<math>n+1</math>)输出与通道(<math>n</math>)输出相同。 1 通道(<math>n+1</math>)输出与通道(<math>n</math>)输出互补。</p>
8 COMBINE1	<p>组合通道(<math>n = 2</math>)</p> <p>使能通道(<math>n</math>)和(<math>n+1</math>)的组合功能。</p> <p>该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 通道(<math>n</math>)和(<math>n+1</math>)独立。 1 通道(<math>n</math>)和(<math>n+1</math>)组合。</p>
7 保留	<p>此字段为保留字段。</p> <p>此只读字段为保留字段且值始终为 0。</p>
6 FAULTENO	<p>故障控制使能(<math>n = 0</math>)</p> <p>使能通道(<math>n</math>)和(<math>n+1</math>)中的故障控制。</p> <p>该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 禁用该通道对中的故障控制。 1 使能该对通道中的故障控制。</p>
5 SYNCENO	<p>同步使能(<math>n = 0</math>)</p> <p>使能寄存器 C(<math>n</math>)V 和 C(<math>n+1</math>)V 的 PWM 同步。</p> <p>0 禁用该通道对中的 PWM 同步。 1 使能该通道对中的 PWM 同步。</p>

下一页继续介绍此表...

**FTMx\_COMBINE 字段描述 (继续)**

字段	描述
4 DTENO	<p>死区时间使能(<math>n = 0</math>) 使能通道(<math>n</math>)和(<math>n+1</math>)中的死区时间插入。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 禁用该通道对中的死区时间插入。 1 使能该通道对中的死区时间插入。</p>
3 DECAP0	<p>双沿捕捉模式捕捉(<math>n = 0</math>) 使能根据通道(<math>n</math>)输入事件捕捉 FTM 计数器值的功能和双沿捕捉位配置。 仅在 DECAPEN = 1 时应用此字段。 如果选定了双边沿捕捉 — 一次性模式，且执行了通道 (<math>n+1</math>) 事件的捕捉时，将由硬件自动清除 DECAP 位。</p> <p>0 双沿捕捉未激活。 1 双沿捕捉已激活。</p>
2 DECAPEN0	<p>双沿捕捉模式使能(<math>n = 0</math>) 使能通道(<math>n</math>)和(<math>n+1</math>)的双沿捕捉模式。此位将根据 表 26-2 在双边沿捕捉模式中重新配置 MSnA、ELSnB:ELSnA 和 ELS(n+1)B:ELS(n+1)A 位的功能。 此字段受写入保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 该通道对中的双沿捕捉模式禁用。 1 该通道对中的双沿捕捉模式使能。</p>
1 COMP0	<p>通道(<math>n</math>)互补(<math>n = 0</math>) 使能组合通道的互补模式。在互补模式下，通道(<math>n+1</math>)输出与通道(<math>n</math>)输出相反。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 通道(<math>n+1</math>)输出与通道(<math>n</math>)输出相同。 1 通道(<math>n+1</math>)输出与通道(<math>n</math>)输出互补。</p>
0 COMBINE0	<p>组合通道(<math>n = 0</math>) 使能通道(<math>n</math>)和(<math>n+1</math>)的组合功能。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。</p> <p>0 通道(<math>n</math>)和(<math>n+1</math>)独立。 1 通道(<math>n</math>)和(<math>n+1</math>)组合。</p>

**26.3.15 死区时间插入控制 (FTMx\_DEADTIME)**

该寄存器选择死区时间预分频器系数和死区时间值。对于死区时间插入而言，所有 FTM 通道均使用此时钟预分频器和该死区时间值。

地址: 基址 基准 + 68h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R																0																		
W																																		
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

**FTMx\_DEADTIME 字段描述**

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7–6 DTPS	死区时间预分频器值  选择系统时钟的分频系数。死区时间计数器使用此预分频时钟。  该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。  0x 系统时钟 1 分频。 10 系统时钟 4 分频。 11 系统时钟 16 分频。
DTVAL	死区时间值  选择死区时间插入值，用于死区时间计数器。死区时间计数器由系统时钟的一部分提供时钟源。参考 DTPS 说明。  死区时间插入数值 = (DTPS × DTVAL)。  DTVAL 选择插入的死区时间计数数目，如下所示： 若 DTVAL 等于 0，则不插入计数数目。 若 DTVAL 等于 1，则插入 1 个计数数目。 若 DTVAL 等于 2，则插入 2 个计数数目。 以此类推，最高可达 63 个计数数目。  该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。

**26.3.16 FTM 外部触发寄存器 (FTMx\_EXTTRIG)**

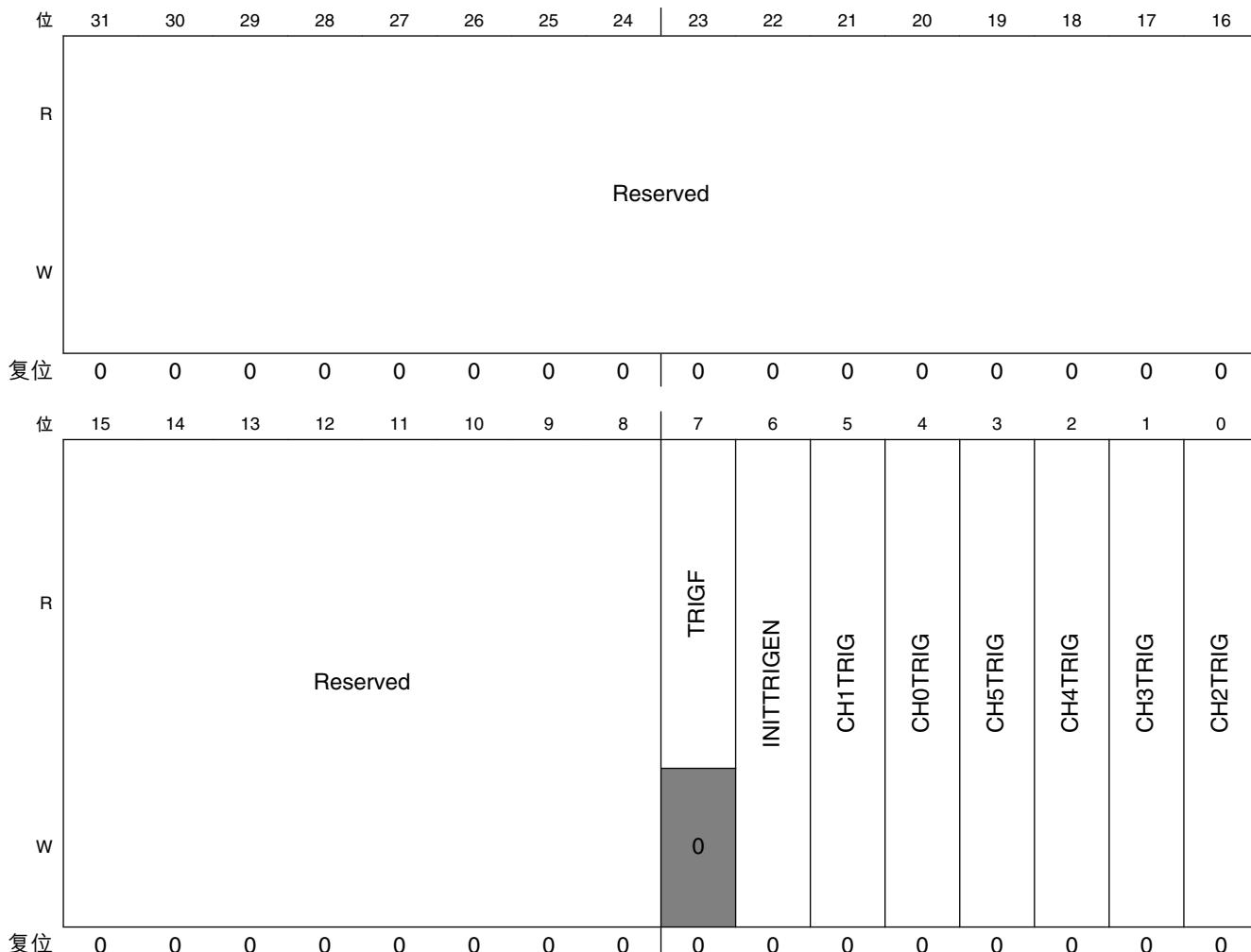
该寄存器：

- 指示何时产生通道触发
- 当 FTM 计数器等于其初始值时，使能触发生成
- 产生通道触发时，选择所用的通道

可选择多个通道，以便在一个 PWM 周期内产生多个触发。

通道 6 和 7 未用来产生通道触发。

地址: 基址 基准 + 6Ch 偏移

**FTMx\_EXTTRIG 字段描述**

字段	描述
31–8 Reserved	此字段为保留字段。
7 TRIGF	<p>通道触发标志</p> <p>生成通道触发时，由硬件置位。在 TRIGF 置位的情况下通过读取 EXTTRIG，然后向 TRIGF 写入 0，可清零 TRIGF。将 1 写入 TRIGF 无效。</p> <p>如果在完成清零序列之前生成另一个通道触发，则序列复位，从而 TRIGF 在完成针对先前 TRIGF 的清零序列之后将保持置位状态。</p> <p>0 未生成通道触发。 1 生成一个通道触发。</p>
6 INITTRIGEN	<p>初始化触发使能</p> <p>当 FTM 计数器等于 CNTIN 寄存器时，使能触发生成。</p> <p>0 禁用初始化触发生成。 1 使能初始化触发生成。</p>

下一页继续介绍此表...

**FTMx\_EXTTRIG 字段描述 (继续)**

字段	描述
5 CH1TRIG	通道 1 触发使能 当 FTM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成。 1 使能通道触发生成。
4 CH0TRIG	通道 0 触发使能 当 FTM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成。 1 使能通道触发生成。
3 CH5TRIG	通道 5 触发使能 当 FTM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成。 1 使能通道触发生成。
2 CH4TRIG	通道 4 触发使能 当 FTM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成。 1 使能通道触发生成。
1 CH3TRIG	通道 3 触发使能 当 FTM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成。 1 使能通道触发生成。
0 CH2TRIG	通道 2 触发使能 当 FTM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成。 1 使能通道触发生成。

**26.3.17 通道极性寄存器 (FTMx\_POL)**

该寄存器定义 FTM 通道的输出极性。

**注**

当故障控制使能且检测到故障条件时，通道输出的安全值就是通道的无效状态。也就是说，通道安全值是其 POL 位的值。

地址: 基址 基准 + 70h 偏移

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R W	Reserved																
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R W	Reserved									POL7	POL6	POL5	POL4	POL3	POL2	POL1	POL0
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

**FTMx\_POL 字段描述**

字段	描述
31–8 Reserved	此字段为保留字段。
7 POL7	通道 7 极性 定义通道输出的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 通道极性为高电平有效。 1 通道极性为低电平有效。
6 POL6	通道 6 极性 定义通道输出的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 通道极性为高电平有效。 1 通道极性为低电平有效。
5 POL5	通道 5 极性 定义通道输出的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 通道极性为高电平有效。 1 通道极性为低电平有效。
4 POL4	通道 4 极性 定义通道输出的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 通道极性为高电平有效。 1 通道极性为低电平有效。
3 POL3	通道 3 极性 定义通道输出的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 通道极性为高电平有效。 1 通道极性为低电平有效。
2 POL2	通道 2 极性 定义通道输出的极性。

下一页继续介绍此表...

**FTMx\_POL 字段描述 (继续)**

字段	描述
	该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 通道极性为高电平有效。 1 通道极性为低电平有效。
1 POL1	通道 1 极性 定义通道输出的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 通道极性为高电平有效。 1 通道极性为低电平有效。
0 POL0	通道 0 极性 定义通道输出的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 通道极性为高电平有效。 1 通道极性为低电平有效。

### 26.3.18 故障模式状态寄存器 (FTMx\_FMS)

该寄存器含有故障检测标志位、写保护使能位和使能故障输入的逻辑 OR。

地址: 基址 基准 + 74h 偏移

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									FAULTF		FAULTIN		0	FAULTF3	FAULTF2	FAULTF1	FAULTFO
W									0	WPEN			0	0	0	0	0
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

FTMx\_FMS 字段描述

字段	描述
31–8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 FAULTF	故障检测标志  代表单个 FAULTF <sub>j</sub> 位的逻辑 OR，其中 : j = 3、2、1、0。在 FAULTF 置位的情况下，当已使能的故障输入端不存在故障条件时，读取 FMS 寄存器，然后向 FAULTF 写入 0 可清零 FAULTF。将 1 写入 FAULTF 无效。  如果在清零序列结束前检测到已使能故障输入的另一个故障条件，则序列复位，因此 FAULTF 将在完成针对上一个故障条件的清零序列之后保持置位状态。当 FAULTF <sub>j</sub> 位独立清零时，FAULTF 位亦清零。  0 未检测到故障条件。 1 检测到故障条件。
6 WPEN	写保护使能

下一页继续介绍此表...

## FTMx\_FMS 字段描述 (继续)

字段	描述
	WPEN 位与 WPDIS 位相反。将 1 写入 WPEN 时，可将其置位。WPEN 清零：WPEN 位读取为 1，然后向 WPDIS 写入 1。向 WPEN 中写入 0 无效。 0 写保护禁用。可写入写保护位。 1 写保护使能。不可写入写保护位。
5 FAULTIN	故障输入 使能故障控制时，它代表滤波（假设滤波器使能）之后已使能故障输入的逻辑 OR。 0 已使能故障输入的逻辑 OR 为 0。 1 已使能故障输入的逻辑 OR 为 1。
4 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
3 FAULTF3	故障检测标志 3 故障控制使能时由硬件置位，对应的故障输入使能，且在故障输入端检测到故障条件。 当对应的故障输入无故障条件时，在 FAULTF3 置位的情况下读取 FMS 寄存器，然后将 0 写入 FAULTF3 即可清零 FAULTF3。将 1 写入 FAULTF3 无效。FAULTF 位清零时，也会清零 FAULTF3 位。 如果在清零序列结束前检测到对应故障输入端存在另一个故障条件，则序列复位，因此 FAULTF3 将在完成针对上一个故障条件的清零序列之后保持置位状态。 0 故障输入端未检测到故障条件。 1 故障输入端检测到故障条件。
2 FAULTF2	故障检测标志 2 故障控制使能时由硬件置位，对应的故障输入使能，且在故障输入端检测到故障条件。 当对应的故障输入无故障条件时，在 FAULTF2 置位的情况下读取 FMS 寄存器，然后将 0 写入 FAULTF2 即可清零 FAULTF2。将 1 写入 FAULTF2 无效。FAULTF 位清零时，也会清零 FAULTF2 位。 如果在清零序列结束前检测到对应故障输入端存在另一个故障条件，则序列复位，因此 FAULTF2 将在完成针对上一个故障条件的清零序列之后保持置位状态。 0 故障输入端未检测到故障条件。 1 故障输入端检测到故障条件。
1 FAULTF1	故障检测标志 1 故障控制使能时由硬件置位，对应的故障输入使能，且在故障输入端检测到故障条件。 当对应的故障输入无故障条件时，在 FAULTF1 置位的情况下读取 FMS 寄存器，然后将 0 写入 FAULTF1 即可清零 FAULTF1。将 1 写入 FAULTF1 无效。FAULTF 位清零时，也会清零 FAULTF1 位。 如果在清零序列结束前检测到对应故障输入端存在另一个故障条件，则序列复位，因此 FAULTF1 将在完成针对上一个故障条件的清零序列之后保持置位状态。 0 故障输入端未检测到故障条件。 1 故障输入端检测到故障条件。
0 FAULTF0	故障检测标志 0 故障控制使能时由硬件置位，对应的故障输入使能，且在故障输入端检测到故障条件。 当对应的故障输入无故障条件时，在 FAULTF0 置位的情况下读取 FMS 寄存器，然后将 0 写入 FAULTF0 即可清零 FAULTF0。将 1 写入 FAULTF0 无效。FAULTF 位清零时，也会清零 FAULTF0 位。 如果在清零序列结束前检测到对应故障输入端存在另一个故障条件，则序列复位，因此 FAULTF0 将在完成针对上一个故障条件的清零序列之后保持置位状态。

下一页继续介绍此表...

## FTMx FMS 字段描述 (继续)

字段	描述
	0 故障输入端未检测到故障条件。 1 故障输入端检测到故障条件。

### 26.3.19 输入捕捉滤波器控制寄存器 (FTMx\_FILTER)

该寄存器选择用于通道输入的滤波器值。

通道4、5、6、7没有输入滤波器。

注

写入 FILTER 寄存器的内容即刻生效，且必须在通道 0、1、2 和 3 为非输入模式时才能完成。如果无法做到，则可能错过有效信号。

地址: 基址 基准 + 78h 偏移

## FTMx FILTER 字段描述

字段	描述
31–16 Reserved	此字段为保留字段。
15–12 CH3FVAL	通道 3 输入滤波器  选择用于通道输入的滤波器值。  数值为 0 时，滤波器禁用。
11–8 CH2FVAL	通道 2 输入滤波器  选择用于通道输入的滤波器值。  数值为 0 时，滤波器禁用。
7–4 CH1FVAL	通道 1 输入滤波器  选择用于通道输入的滤波器值。  数值为 0 时，滤波器禁用。
CH0FVAL	通道 0 输入滤波器  选择用于通道输入的滤波器值。  数值为 0 时，滤波器禁用。

### 26.3.20 故障控制寄存器 (FTMx\_FLTCTRL)

该寄存器选择故障输入的滤波器值，使能故障输入和故障输入滤波器。

地址: 基址 基准 + 7Ch 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				FFVAL				FFLTR3EN	FFLTR2EN	FFLTR1EN	FFLTROEN	FAULT3EN	FAULT2EN	FAULT1EN	FAULT0EN
W									0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx\_FLTCTRL 字段描述

字段	描述
31–12 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
11–8 FFVAL	故障输入滤波器 选择用于故障输入的滤波器值。 数值为 0 时，故障滤波器禁用。 注：写操作即时生效，且必须在禁用故障控制或全部故障输入时完成。如果无法做到，则可能错过故障检测。
7 FFLTR3EN	故障输入 3 滤波器使能 使能故障输入的滤波器。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用故障输入滤波器。 1 使能故障输入滤波器。
6 FFLTR2EN	故障输入 2 滤波器使能 使能故障输入的滤波器。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用故障输入滤波器。 1 使能故障输入滤波器。
5 FFLTR1EN	故障输入 1 滤波器使能 使能故障输入的滤波器。

下一页继续介绍此表...

**FTMx\_FLTCTRL 字段描述 (继续)**

字段	描述
	该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用故障输入滤波器。 1 使能故障输入滤波器。
4 FFLTROEN	故障输入 0 滤波器使能 使能故障输入的滤波器。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用故障输入滤波器。 1 使能故障输入滤波器。
3 FAULT3EN	故障输入 3 使能 使能故障输入。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用故障输入。 1 使能故障输入。
2 FAULT2EN	故障输入 2 使能 使能故障输入。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用故障输入。 1 使能故障输入。
1 FAULT1EN	故障输入 1 使能 使能故障输入。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用故障输入。 1 使能故障输入。
0 FAULT0EN	故障输入 0 使能 使能故障输入。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 禁用故障输入。 1 使能故障输入。

### 26.3.21 配置寄存器 (FTMx\_CONF)

该寄存器选择应当在 TOF 位置位前发生的 FTM 计数器溢出次数、FTM 在 Debug 模式下的特性、外部全局时基的使用以及全局时基信号生成。

地址: 基址 基准 + 84h 偏移

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									GTBEOUT	GTBEEN	0						
W										0		0	0	0	0	0	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

FTMx\_CONF 字段描述

字段	描述
31–11 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
10 GTBEOUT	全局时基输出  使能到其它 FTM 的全局时基信号生成。  0 禁用全局时基信号生成。 1 将启用全局时基的信号生成。
9 GTBEEN	全局时基使能  配置 FTM，以使用另一个 FTM 产生的外部全局时基信号。  0 禁用外部全局时基。 1 使能外部全局时基。
8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7–6 BDMMODE	Debug 模式  选择 B Debug 模式下的 FTM 特性。请参见 <a href="#">Debug 模式</a> 。
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
NUMTOF	TOF 频率

下一页继续介绍此表...

**FTMx\_CONF 字段描述 (继续)**

字段	描述
	<p>选择计数器溢出数量与 TOF 位置位次数之比。</p> <p>NUMTOF = 0 : TOF 位针对每一个计数器溢出进行置位。</p> <p>NUMTOF = 1 : TOF 位针对第一个计数器溢出进行置位，但不会对下一个溢出置位。</p> <p>NUMTOF = 2 : TOF 位针对第一个计数器溢出进行置位，但不对之后的两个溢出置位。</p> <p>NUMTOF = 3 : TOF 位针对第一个计数器溢出进行置位，但不对之后的三个溢出置位。</p> <p>以此类推，最高可达 31 个。</p>

**26.3.22 FTM 故障输入极性寄存器 (FTMx\_FLTPOL)**

该寄存器定义故障输入极性。

地址: 基址 基准 + 88h 偏移

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									0					FLT3POL	FLT2POL	FLT1POL	FLT0POL
W														0	0	0	0
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

**FTMx\_FLTPOL 字段描述**

字段	描述
31–4 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
3 FLT3POL	故障输入 3 极性 定义故障输入的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 故障输入极性为高电平有效。故障输入端的 1 表示故障。 1 故障输入极性为低电平有效。故障输入端的 0 表示故障。
2 FLT2POL	故障输入 2 极性 定义故障输入的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。

下一页继续介绍此表...

**FTMx\_FLTPOL 字段描述 (继续)**

字段	描述
	0 故障输入极性为高电平有效。故障输入端的 1 表示故障。 1 故障输入极性为低电平有效。故障输入端的 0 表示故障。
1 FLT1POL	故障输入 1 极性 定义故障输入的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 故障输入极性为高电平有效。故障输入端的 1 表示故障。 1 故障输入极性为低电平有效。故障输入端的 0 表示故障。
0 FLT0POL	故障输入 0 极性 定义故障输入的极性。 该字段为写保护。仅当 MODE[WPDIS] = 1 时，才能写操作。 0 故障输入极性为高电平有效。故障输入端的 1 表示故障。 1 故障输入极性为低电平有效。故障输入端的 0 表示故障。

**26.3.23 同步配置寄存器 (FTMx\_SYNCONF)**

当检测到硬件触发 j 时，若 FTM 清零 TRIGj 位(j=0,1,2)，该寄存器将选择 PWM 同步配置、SWOCTRL、INVCTRL 和 CNTIN 寄存器进行同步。

地址: 基址 基准 + 8Ch 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									0			HWSOC	HWNVC	HWOM	HWWRBUF	HWRSTCNT
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			0		SWSOC	SWINVNC	SWOM	SWWRBUF	SWRSTCNT	SYNCODE	0	SWOC	INVNC	0	CNTINC	0
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**FTMx\_SYNCONF 字段描述**

字段	描述
31–21 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

下一页继续介绍此表...

## FTMx\_SYNCONF 字段描述 (继续)

字段	描述
20 HWSOC	软件输出控制同步通过硬件触发激活。 0 硬件触发不激活 SWOCTRL 寄存器同步。 1 硬件触发激活 SWOCTRL 寄存器同步。
19 HWINVC	反相控制同步通过硬件触发激活。 0 硬件触发不激活 INVCTRL 寄存器同步。 1 硬件触发激活 INVCTRL 寄存器同步。
18 HWOM	输出屏蔽同步通过硬件触发激活。 0 硬件触发不激活 OUTMASK 寄存器同步。 1 硬件触发激活 OUTMASK 寄存器同步。
17 HWRBUF	MOD、CNTIN 和 CV 寄存器同步通过硬件触发激活。 0 硬件触发不激活 MOD、CNTIN 和 CV 寄存器同步。 1 硬件触发激活 MOD、CNTIN 和 CV 寄存器同步。
16 HWRSTCNT	FTM 计数器同步通过硬件触发激活。 0 硬件触发不激活 FTM 计数器同步。 1 硬件触发激活 FTM 计数器同步。
15–13 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
12 SWSOC	软件输出控制同步通过软件触发激活。 0 软件触发不激活 SWOCTRL 寄存器同步。 1 软件触发激活 SWOCTRL 寄存器同步。
11 SWINVC	反相控制同步通过软件触发激活。 0 软件触发不激活 INVCTRL 寄存器同步。 1 软件触发激活 INVCTRL 寄存器同步。
10 SWOM	输出屏蔽同步通过软件触发激活。 0 软件触发不激活 OUTMASK 寄存器同步。 1 软件触发激活 OUTMASK 寄存器同步。
9 SWWRBUF	MOD、CNTIN 和 CV 寄存器同步通过软件触发激活。 0 软件触发不激活 MOD、CNTIN 和 CV 寄存器同步。 1 软件触发激活 MOD、CNTIN 和 CV 寄存器同步。
8 SWRSTCNT	FTM 计数器同步通过软件触发激活。 0 软件触发不激活 FTM 计数器同步。 1 软件触发激活 FTM 计数器同步。
7 SYNCMODE	同步模式 选择 PWM 同步模式。 0 选择传统 PWM 同步。 1 选择增强 PWM 同步。

下一页继续介绍此表...

## FTMx\_SYNCONF 字段描述 (继续)

字段	描述
6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5 SWOC	SWOCTRL 寄存器同步 0 SWOCTRL 寄存器将在所有系统时钟的上升沿以其缓冲区的数值进行更新。 1 SWOCTRL 寄存器通过 PWM 同步以其缓冲区值进行更新。
4 INV C	INVCTRL 寄存器同步 0 INVCTRL 寄存器将在所有系统时钟的上升沿以其缓冲区的数值进行更新。 1 INVCTRL 寄存器通过 PWM 同步以其缓冲区值进行更新。
3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 CNTINC	CNTIN 寄存器同步 0 CNTIN 寄存器将在所有系统时钟的上升沿以其缓冲区的数值进行更新。 1 CNTIN 寄存器通过 PWM 同步以其缓冲区值进行更新。
1 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
0 HWTRIGMODE	硬件触发模式 0 检测到硬件触发 j 时 ( $j = 0, 1, 2$ ), FTM 清零 TRIGj 位。 1 检测到硬件触发 j 时 ( $j = 0, 1, 2$ ), FTM 不清零 TRIGj 位。

## 26.3.24 FTM 反相控制寄存器 (FTMx\_INVCTRL)

该寄存器控制通道(n)输出何时变为通道(n+1)输出，以及通道(n+1)输出何时变为通道(n)输出。每个 INVmEN 位使能针对相应通道对 m 的反相操作。

该寄存器具有写缓冲区。INVmEN 位通过 INVCTRL 寄存器同步进行更新。

地址: 基址 基准 + 90h 偏移

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

**FTMx\_INVCTRL 字段描述**

字段	描述
31–4 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
3 INV3EN	通道对 3 反相使能 0 禁用反相。 1 使能反相。
2 INV2EN	通道对 2 反相使能 0 禁用反相。 1 使能反相。
1 INV1EN	通道对 1 反相使能 0 禁用反相。 1 使能反相。
0 INV0EN	通道对 0 反相使能 0 禁用反相。 1 使能反相。

**26.3.25 FTM 软件输出控制寄存器 (FTMx\_SWOCTRL)**

该寄存器使能通道(n)输出的软件控制，并定义强制进入通道(n)输出的数值：

- CHnOC 位通过软件使能对应通道(n)输出的控制。
- CHnOCV 位选择强制进入对应通道(n)输出的数值。

该寄存器具有写缓冲区。该字段通过 SWOCTRL 寄存器同步进行更新。

地址：基址 基准 + 94h 偏移

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	CH7OCV	CH6OCV	CH5OCV	CH4OCV	CH3OCV	CH2OCV	CH1OCV	CH0OCV		CH7OC	CH6OC	CH5OC	CH4OC	CH3OC	CH2OC	CH1OC	CH0OC
W	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

**FTMx\_SWOCTRL 字段描述**

字段	描述
31–16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15 CH7OCV	通道 7 软件输出控制值 0 软件输出控制强制 0 进入通道输出。 1 软件输出控制强制 1 进入通道输出。
14 CH6OCV	通道 6 软件输出控制值 0 软件输出控制强制 0 进入通道输出。 1 软件输出控制强制 1 进入通道输出。
13 CH5OCV	通道 5 软件输出控制值 0 软件输出控制强制 0 进入通道输出。 1 软件输出控制强制 1 进入通道输出。
12 CH4OCV	通道 4 软件输出控制值 0 软件输出控制强制 0 进入通道输出。 1 软件输出控制强制 1 进入通道输出。
11 CH3OCV	通道 3 软件输出控制值 0 软件输出控制强制 0 进入通道输出。 1 软件输出控制强制 1 进入通道输出。
10 CH2OCV	通道 2 软件输出控制值 0 软件输出控制强制 0 进入通道输出。 1 软件输出控制强制 1 进入通道输出。
9 CH1OCV	通道 1 软件输出控制值 0 软件输出控制强制 0 进入通道输出。 1 软件输出控制强制 1 进入通道输出。
8 CH0OCV	通道 0 软件输出控制值 0 软件输出控制强制 0 进入通道输出。 1 软件输出控制强制 1 进入通道输出。
7 CH7OC	通道 7 软件输出控制使能 0 通道输出不受软件输出控制的影响。 1 通道输出受软件输出控制的影响。
6 CH6OC	通道 6 软件输出控制使能 0 通道输出不受软件输出控制的影响。 1 通道输出受软件输出控制的影响。
5 CH5OC	通道 5 软件输出控制使能 0 通道输出不受软件输出控制的影响。 1 通道输出受软件输出控制的影响。
4 CH4OC	通道 4 软件输出控制使能

下一页继续介绍此表...

**FTMx\_SWOCTRL 字段描述 (继续)**

字段	描述
	0 通道输出不受软件输出控制的影响。 1 通道输出受软件输出控制的影响。
3 CH3OC	通道 3 软件输出控制使能 0 通道输出不受软件输出控制的影响。 1 通道输出受软件输出控制的影响。
2 CH2OC	通道 2 软件输出控制使能 0 通道输出不受软件输出控制的影响。 1 通道输出受软件输出控制的影响。
1 CH1OC	通道 1 软件输出控制使能 0 通道输出不受软件输出控制的影响。 1 通道输出受软件输出控制的影响。
0 CH0OC	通道 0 软件输出控制使能 0 通道输出不受软件输出控制的影响。 1 通道输出受软件输出控制的影响。

**26.3.26 FTM PWM 加载寄存器 (FTMx\_PWMLOAD)**

当 FTM 计数器从 MOD 寄存器值变更至其下一个值，或者发生通道(j)匹配时，使能 MOD、CNTIN、C(n)V 和 C(n+1)V 寄存器的加载，数值为它们写入缓冲区数值。当 FTM 计数器 = C(j)V 时，通道(j)发生匹配。

地址: 基址 基准 + 98h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									0							
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R				0			LDOK	0	CH7SEL	CH6SEL	CH5SEL	CH4SEL	CH3SEL	CH2SEL	CH1SEL	CH0SEL
W									0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**FTMx\_PWMLOAD 字段描述**

字段	描述
31–10 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

下一页继续介绍此表...

## FTMx\_PWMLOAD 字段描述 (继续)

字段	描述
9 LDOK	加载使能 使能 MOD、CNTIN 和 CV 寄存器的加载，数值为它们的写缓冲区内容。 0 禁用更新值加载。 1 使能更新值加载。
8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 CH7SEL	通道 7 选择 0 匹配过程中不包括通道。 1 匹配过程中包括通道。
6 CH6SEL	通道 6 选择 0 匹配过程中不包括通道。 1 匹配过程中包括通道。
5 CH5SEL	通道 5 选择 0 匹配过程中不包括通道。 1 匹配过程中包括通道。
4 CH4SEL	通道 4 选择 0 匹配过程中不包括通道。 1 匹配过程中包括通道。
3 CH3SEL	通道 3 选择 0 匹配过程中不包括通道。 1 匹配过程中包括通道。
2 CH2SEL	通道 2 选择 0 匹配过程中不包括通道。 1 匹配过程中包括通道。
1 CH1SEL	通道 1 选择 0 匹配过程中不包括通道。 1 匹配过程中包括通道。
0 CH0SEL	通道 0 选择 0 匹配过程中不包括通道。 1 匹配过程中包括通道。

## 26.4 功能说明

下图展示了本文档中用来表示计数器和信号生成的各种用法。

FTM计数为向上计数。  
通道(n)处于高真EPWM模式。

PS[2:0] = 001  
CNTIN = 0x0000  
MOD = 0x0004  
CnV = 0x0002

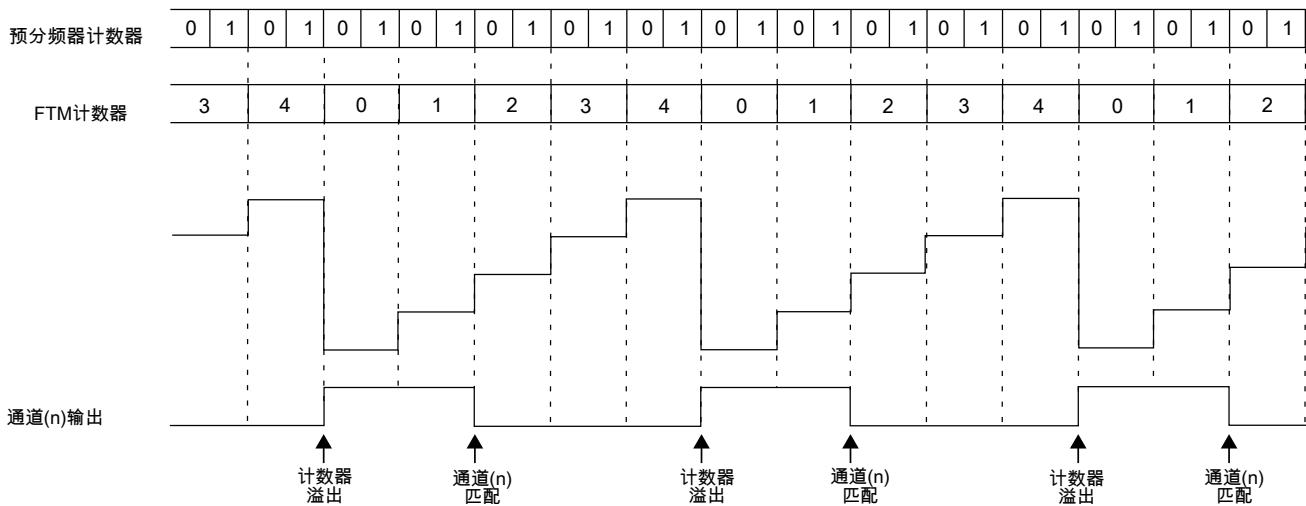


图 26-2. 所用的用法

#### 26.4.1 时钟源

FTM 只有一个时钟域：系统时钟。

#### 26.4.1.1 计数器时钟源

SC 寄存器中的 CLKS[1:0]位可以从三种可能的时钟源中为 FTM 计数器选择一种，或者禁用 FTM 计数器。发生任意 MCU 复位后，CLKS[1:0] = 0:0，这意味着未选择任何时钟源。

可在任意时间对 CLKS[1:0]位进行读取或写入操作。通过向 CLKS[1:0]位写入 0:0 禁用 FTM 计数器不会影响 FTM 计数器值或其他寄存器。

固定频率时钟 ICSFFCLK 是一种可以用于 FTM 计数器的备选时钟源，允许在系统时钟或外部时钟以外另选一种时钟源。这种时钟输入源集成在芯片内部。有关更多信息，请参见具体的芯片文档。由于 FTM 硬件的实施限制，固定频率时钟的频率不得超过系统时钟频率的 1/2。

外部时钟会经过一个由系统时钟计时的同步器，以确保计数器转换与系统时钟转换恰当的同步。因此，为符合奈奎斯特准则，并考虑到时钟抖动，外部时钟源的频率不得超过系统时钟频率的  $1/4$ 。

## 26.4.2 预分频器

所选的计数器时钟源会经过一个预分频器，该预分频器是一个 7 位计数器。由 PS[2:0]位选择预分频器的值。下图给出了预分频器计数器和 FTM 计数器的示例。

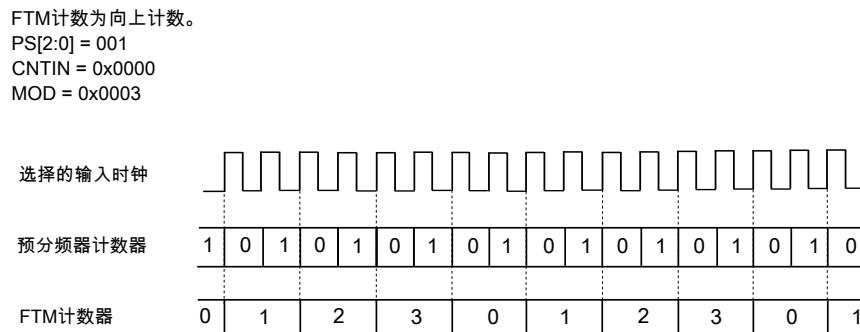


图 26-3. 预分频器计数器示例

## 26.4.3 计数器寄存器

FTM 有一个 16 位计数器，通道可以选择用于输入或输出模式。FTM 计数器时钟是由选定的输入时钟经过预分频得到的。

FTM 计时器有这些工作模式：

- 向上计数
- 向上-向下计数

### 26.4.3.1 向上计数

在以下情形中会选择向上计数：

- CPWMS = 0

CNTIN 定义计数的起始值，MOD 定义计数的最终值，参见下图。CNTIN 的值加载到 FTM 计数器中，计数器的值递增，直至达到 MOD 的值，此时计数器将重新加载 CNTIN 的值。

采用向上计数时的 FTM 周期为 $(MOD - CNTIN + 0x0001) \times FTM$  计数器时钟的周期。

FTM 计数器从 MOD 变为 CNTIN 时，TOF 位将置位。

FTM计数为向上计数。  
 CNTIN = 0xFFFF ( 在二的补码中等于-4 )  
 MOD = 0x0004

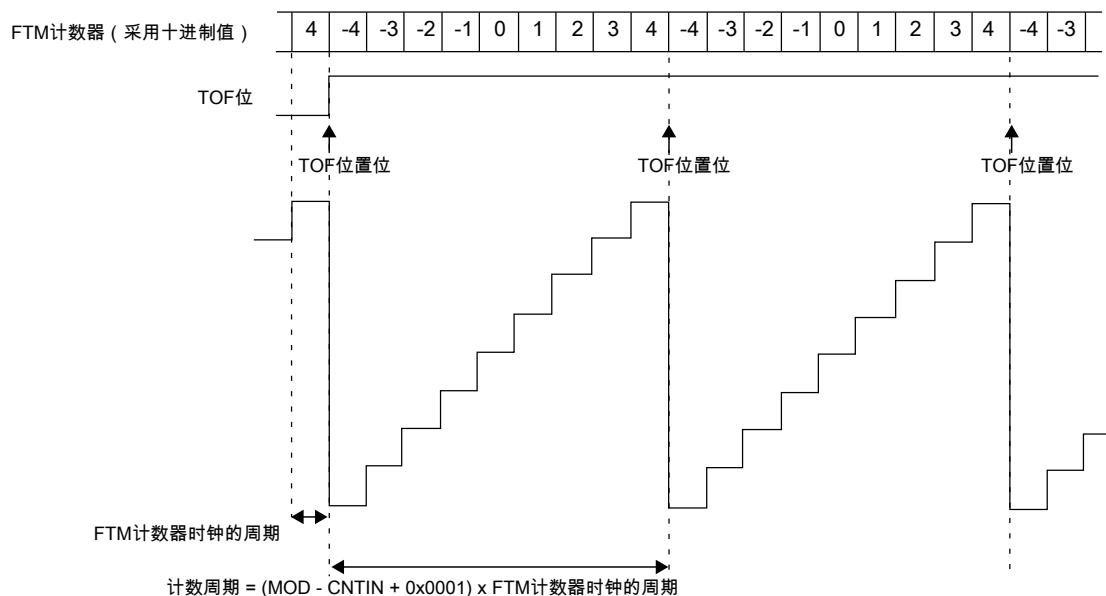


图 26-4. FTM 向上有符号计数示例

表 26-4. 基于 CNTIN 值的 FTM 计数

满足条件	结果
CNTIN = 0x0000	FTM 计数与 TPM 向上计数等效，即向上无符号计数。参见下图。
CNTIN[15] = 1	FTM 计数器的初始值是一个二补码负数，因此 FTM 计数为向上有符号计数。
CNTIN[15] = 0 且 CNTIN ≠ 0x0000	FTM 计数器的初始值是一个正数，因此 FTM 计数为向上无符号计数。

FTM 计数向上数  
CNTIN = 0x0000  
MOD = 0x0004

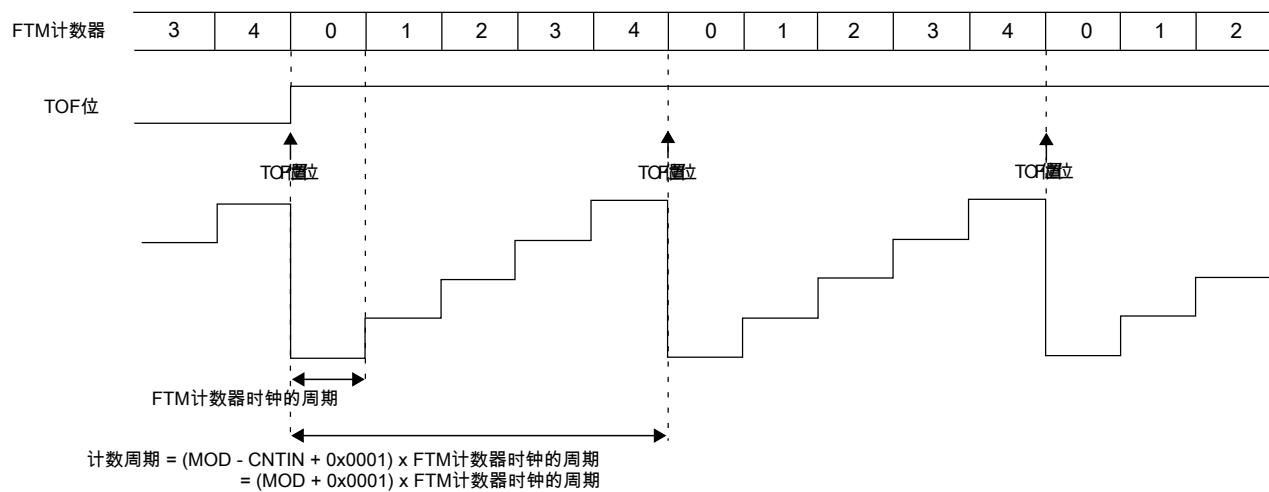


图 26-5. CNTIN = 0x0000 时的 FTM 向上计数示例

### 注

- 无论是无符号计数还是有符号计数，FTM 工作只有在 CNTIN 寄存器的值小于 MOD 寄存器的值时才有效。软件负责确保 CNTIN 和 MOD 寄存器中的值满足此要求。CNTIN 和 MOD 如有任何不满足此要求的值，将导致意外现象发生。
- MOD = CNTIN 是一个冗余条件。这种情况下，FTM 计数器始终等于 MOD，将在 FTM 计数器时钟的每一个上升沿置位 TOF 位。
- MOD = 0x0000、CNTIN = 0x0000（例如，复位后）且 FTMEN = 1 时，FTM 计数器将在 0x0000 保持停止，直到向 MOD 或 CNTIN 寄存器中写入非零值。
- 不建议将 CNTIN 设置为大于 MOD 的值，因为这种非寻常设置可能导致 FTM 的操作难以理解。然而，对于这种配置没有任何限制，下图给出了示例。

FTM计数为向上计数  
MOD = 0x0005  
CNTIN = 0x0015

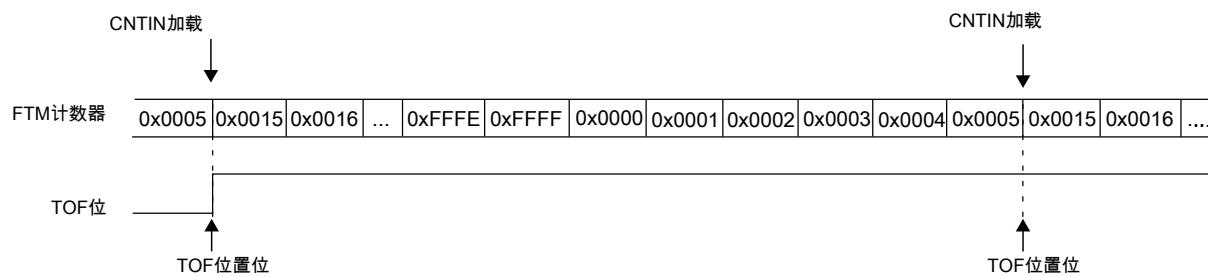


图 26-6. CNTIN 值大于 MOD 值时的向上计数示例

### 26.4.3.2 向上-向下计数

在以下情形中会选择自上而下计数：

- CPWMS = 1

CNTIN 定义计数的起始值，MOD 定义计数的最终值。CNTIN 的值加载到 FTM 计数器中，计数器的值一直增加，直至达到 MOD 的值，紧接着计数器的值一直减少，直至回到 CNTIN 的值，然后计数器将重新开始自上而下计数。

采用向上-向下计数时的 FTM 周期为  $2 \times (MOD - CNTIN) \times FTM$  计数器时钟的周期。

FTM 计数器从 MOD 更改为(MOD - 1)时，TOF 位将被置位。

如果(CNTIN = 0x0000)，FTM 计数与 TPM 向上-向下计数等效，即无符号向上-向下计数。参见下图。

FTM 计数向上-向下  
CNTIN = 0x0000  
MOD = 0x0004

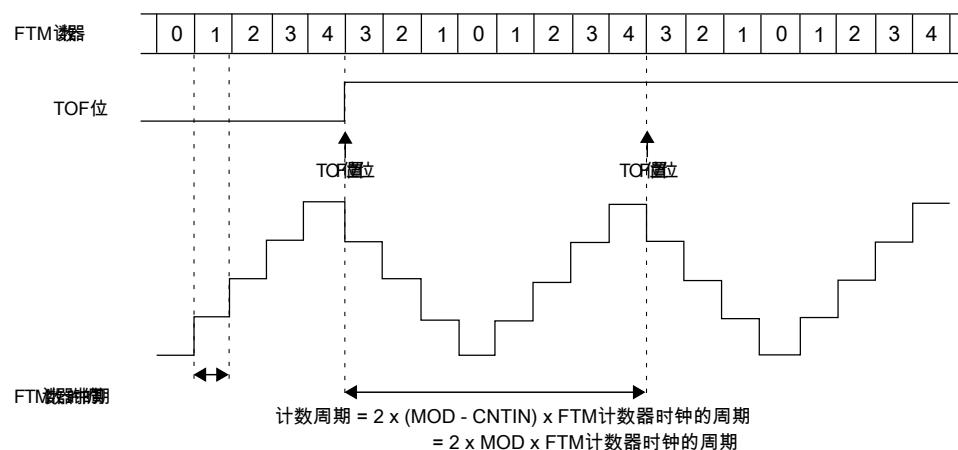


图 26-7. CNTIN = 0x0000 时的向上-向下计数示例

### 注

当 CNTIN 在向上-向下计数中为非零时，将会生成有效的 CPWM 信号：

- 如果  $CnV > CNTIN$ ，或者
- 如果  $CnV = 0$  或  $CnV[15] = 1$ 。在此情况下，将生成 0% CPWM。

### 26.4.3.3 自由运行计数器

如果( $\text{FTMEN} = 0$ )且 ( $\text{MOD} = 0x0000$  或  $\text{MOD} = 0xFFFF$ )，则 FTM 计数器为自由运行计数器。这种情况下，FTM 计数器从  $0x0000$  到  $0xFFFF$  自由运行，并且当 FTM 计数器从  $0xFFFF$  更改为  $0x0000$  时，TOF 位将置位。参见下图。

FTMEN = 0  
MOD = 0x0000

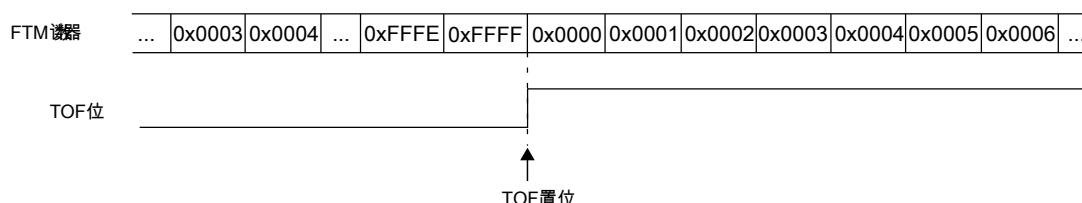


图 26-8. FTM 计数器自由运行时的示例

FTM 计数器在以下情形中也是自由运行计数器：

- $\text{FTMEN} = 1$

- CPWMS = 0
- CNTIN = 0x0000, 且
- MOD = 0xFFFF

#### 26.4.3.4 计数器复位

以下任一情形都会导致 FTM 计数器复位为 CNTIN 寄存器中的值，并且通道输出重置为其初始值，但处于输出比较模式的通道例外。

- 对 CNT 采取的任何写入操作。
- FTM 计数器同步。

#### 26.4.3.5 TOF 置位时

NUMTOF[4:0]位定义了 TOF 置位之前 FTM 计数器溢出应该发生的次数。如果 NUMTOF[4:0] = 0x00，则在每次 FTM 计数器溢出时置位 TOF 位。

对 NUMTOF[4:0]位采取写入操作之后，再通过对 FTM 计数器的 CNT 进行写入操作来初始化 FTM 计数器，这样可避免混淆何时发生第一次计数器溢出。

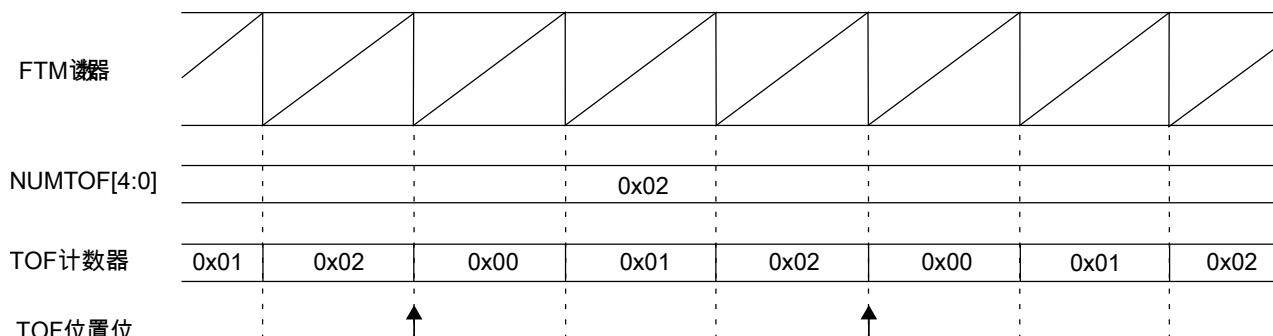


图 26-9. NUMTOF = 0x02 条件下的周期性 TOF

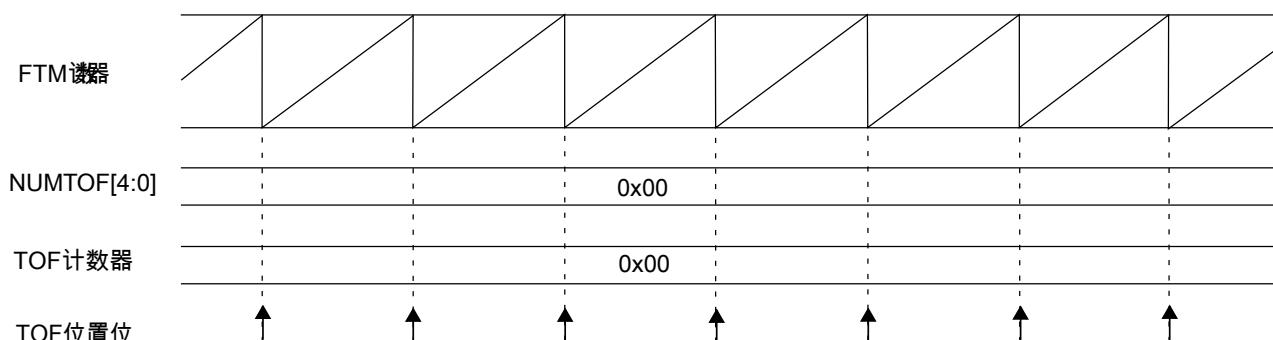


图 26-10. NUMTOF = 0x00 条件下的周期性 TOF

## 26.4.4 输入捕捉模式

在以下情形中会选择输入捕捉模式：

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0
- MSnB:MSnA = 0:0, 且
- ELSnB:ELSnA ≠ 0:0

通道输入中发生所选边沿时, FTM 计数器的当前值将会捕捉到 CnV 寄存器中, 同时还会将 CHnF 位置位并生成通道中断 (如果已通过 CHnIE = 1 使能)。参见下图。

如果某个通道已配置为输入捕捉模式, 则 FTMxCHn 引脚为边沿触发的输入。

ELSnB:ELSnA 控制位可确定是哪个边沿 (上升沿或下降沿) 触发输入捕捉事件。注意, 可以被正确检测到输入信号的最大频率为系统时钟的四分之一, 这是满足信号采样的奈奎斯特定律所必需的条件。

在输入捕捉模式下, 将忽略对 CnV 寄存器采取的写入操作。

在 Debug 模式下, 输入捕捉功能按配置正常工作。发生所选边沿事件时, 因调试而冻结的 FTM 计数器值将捕捉到 CnV 寄存器中, 并且 CHnF 位会置位。

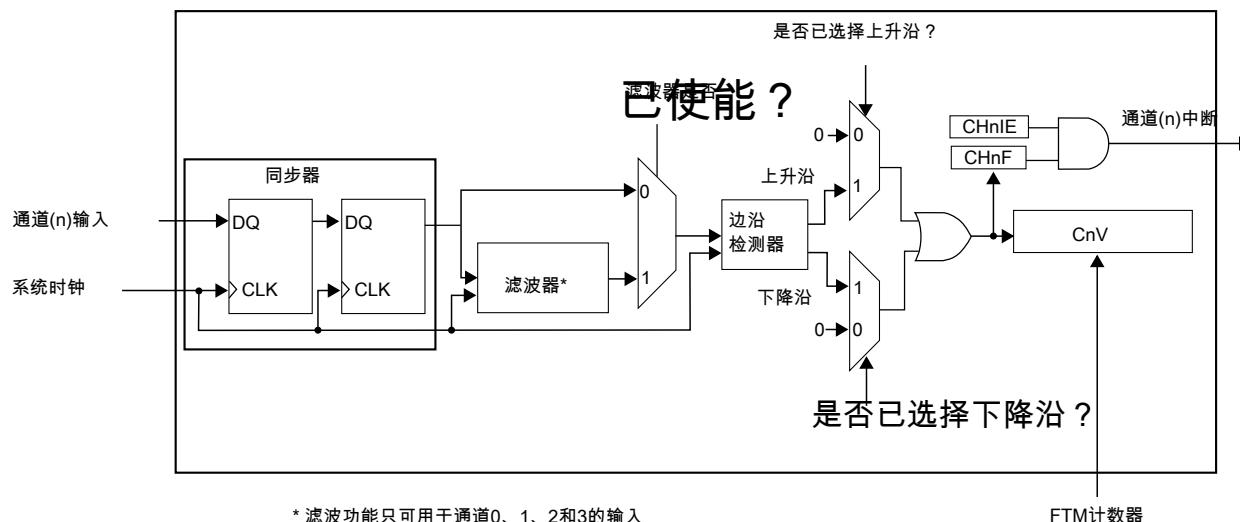


图 26-11. 输入捕捉模式

如果通道输入中没有使能滤波器, 则输入信号总是会被延迟 3 个系统时钟的上升沿, 也就是说两个上升沿到同步器, 再加一个上升沿到边沿检测器。换言之, 将在通道输入中发生有效边沿后, CHnF 位在系统时钟的第三个上升沿上置位。

### 26.4.4.1 输入捕捉模式的滤波器

滤波器功能只可用于通道 0、1、2 和 3。

首先，由系统时钟同步输入信号。同步之后，输入信号进入滤波器区块。参见下图。

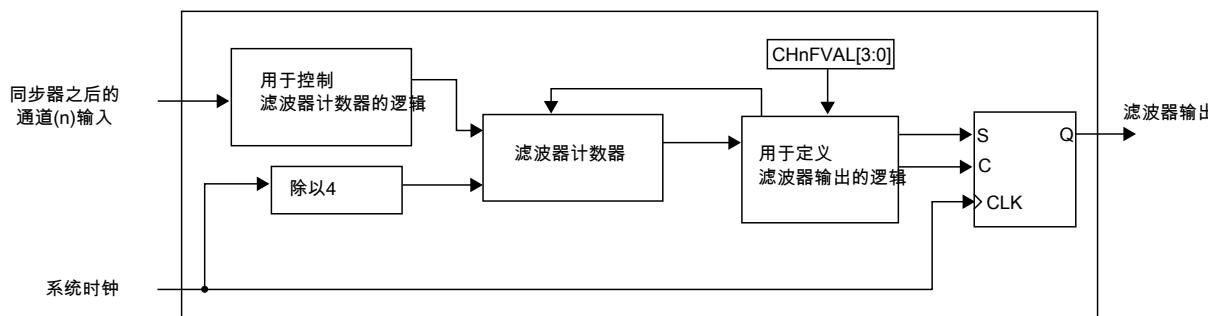


图 26-12. 通道输入滤波器

当输入信号中发生状态更改时，计数器将复位并开始向上计数。只要新状态在输入端保持稳定，计数器就会继续增加数值。当计数器等于 CHnFVAL[3:0] 时，输入信号的状态更改得以验证。然后就会作为脉冲边沿发送到边沿检测器。

如果在得以验证之前相对边沿出现在输入信号上，计数器就会复位。下一次输入转换时，计数器再次开始计数。任何比 CHnFVAL[3:0] ( $x 4$  个系统时钟) 所选的最小值短的脉冲都会被认作是毛刺，不会传递到边沿检测器上。下图为输入滤波器的时序图。

CHnFVAL[3:0] 位为零时，将禁用滤波器功能。这种情况下，输入信号将按系统时钟的 3 个上升沿延迟。如果( $CHnFVAL[3:0] \neq 0000$ )，则输入信号按最小脉宽 ( $CHnFVAL[3:0] \times 4$  个系统时钟) 延迟，外加系统时钟的 4 个上升沿：两个上升沿到同步器，一个上升沿到滤波器输出，另外一个到边沿检测器。换言之，有效边沿出现在通道输入端之后， $CHnF$  将在( $4+4 \times CHnFVAL[3:0]$ )个系统时钟周期后被置位。

通道输入滤波器中计数器的时钟是系统时钟的 4 分频。

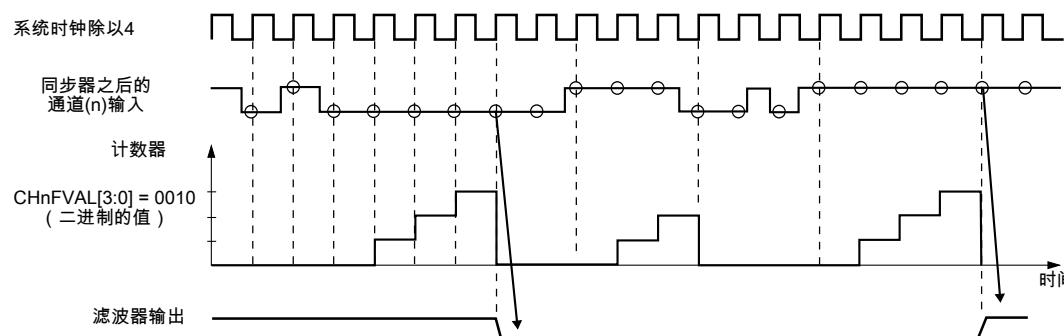


图 26-13. 通道输入滤波器示例

## 26.4.5 输出比较模式

在以下情形中会选择输出比较模式：

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, 且
- MSnB:MSnA = 0:1

在输出比较模式下, FTM 可生成具有可编程位置、极性、持续时间和频率的定时脉冲。当计数器与某个输出比较通道的 CnV 寄存器中的值匹配时, 可以设置、清除或翻转通道(n)输出。

当某个通道初次配置为翻转模式时, 将保持通道输出先前的值, 直到发生第一个输出比较事件。

通道(n)匹配 (FTM 计数器 = CnV) 时, CHnF 位将置位并生成通道(n)中断 (条件是 CHnIE = 1)。

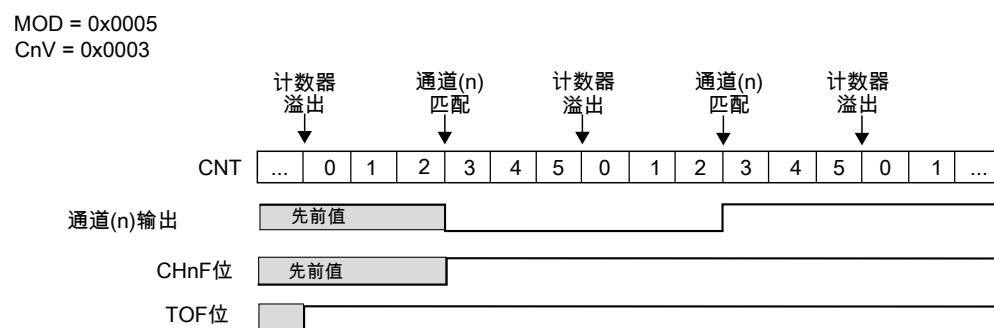


图 26-14. 匹配翻转通道输出时的输出比较模式示例

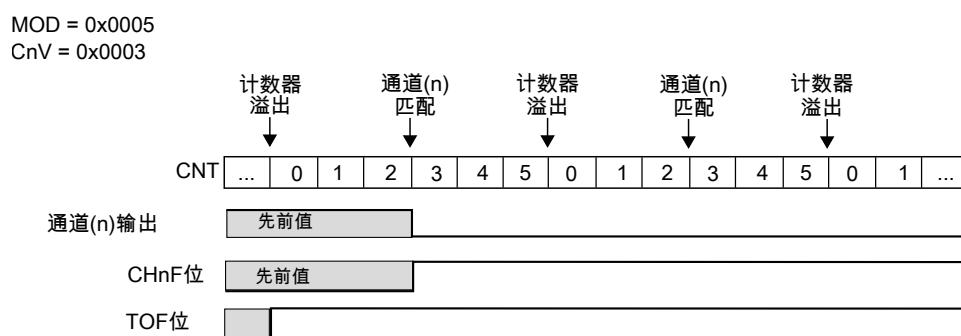


图 26-15. 匹配清空通道输出时的输出比较模式示例

MOD = 0x0005  
CnV = 0x0003

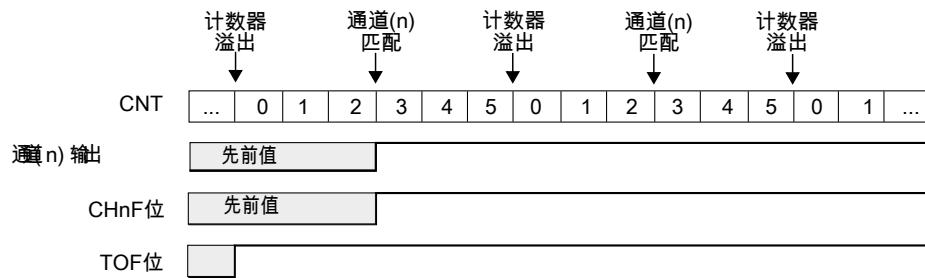


图 26-16. 匹配置位通道输出时的输出比较模式示例

如果计数器达到 CnV 寄存器中的值时满足(ELSnB:ELSnA = 0:0)条件，则 CHnF 位置位并生成通道(n)中断（如果 CHnIE = 1），但是，通道(n)输出不由 FTM 修改和控制。

## 26.4.6 边沿对齐 PWM (EPWM)模式

在以下情形中会选择边沿对齐模式：

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, 以及
- MSnB = 1

EPWM 周期取决于(MOD – CNTIN + 0x0001)，脉宽(占空比)取决于(CnV – CNTIN)。

通道(n)匹配 (FTM 计数器 = CnV) (即脉宽结束) 时，CHnF 位将置位并生成通道(n)中断 (条件是 CHnIE = 1)。

这种类型的 PWM 信号称为边沿对齐，因为所有 PWM 信号的前沿都与周期的开始对齐，这对 FTM 内的所有通道都一样。

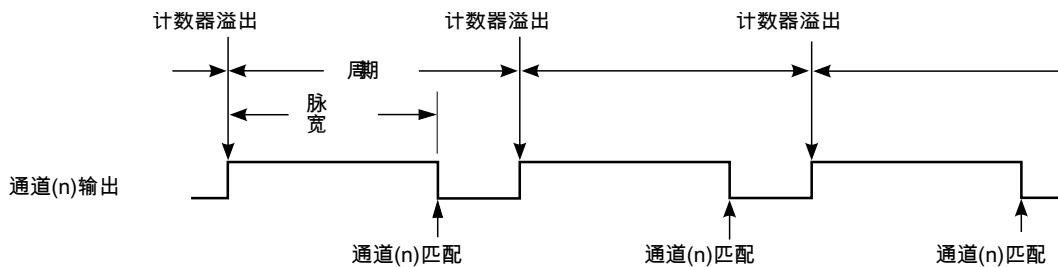


图 26-17. ELSnB:ELSnA = 1:0 条件下的 EPWM 周期和脉宽

如果计数器达到 CnV 寄存器中的值时满足(ELSnB:ELSnA = 0:0)条件，则 CHnF 位置位并生成通道(n)中断（如果 CHnIE = 1），但是，通道(n)输出不由 FTM 控制。

如果( $\text{ELSnB:ELSnA} = 1:0$ )，那么，通道(n)输出会在 CNTIN 寄存器值加载到 FTM 计数器时强制为高电平，在通道(n)匹配 (FTM 计数器 = CnV) 的情况下强制为低电平。参见下图。

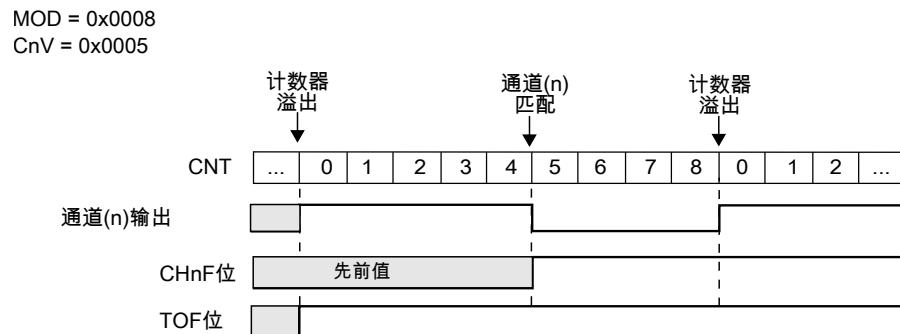


图 26-18.  $\text{ELSnB:ELSnA} = 1:0$  条件下的 EPWM 信号

如果( $\text{ELSnB:ELSnA} = X:1$ )，那么，通道(n)输出会在 CNTIN 寄存器值加载到 FTM 计数器时强制为低电平，在通道(n)匹配 (FTM 计数器 = CnV) 的情况下强制为高电平。参见下图。

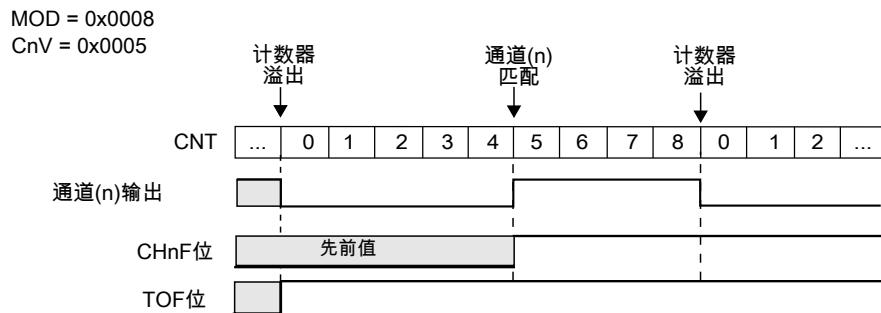


图 26-19.  $\text{ELSnB:ELSnA} = X:1$  条件下的 EPWM 信号

如果( $\text{CnV} = 0x0000$ )，则通道(n)输出为 0% 占空比 EPWM 信号，而且 CHnF 位不置位，即便存在通道(n)匹配也依然如此。如果( $\text{CnV} > \text{MOD}$ )，则通道(n)输出为 100% 占空比 EPWM 信号，而且 CHnF 位不置位，即便存在通道(n)匹配也依然如此。因此，要获得 100% 占空比 EPWM 信号，MOD 必须小于 0xFFFF。

## 注

当 CNTIN 为非零时，将生成以下 EPWM 信号：

- 0% EPWM 信号 (如果  $\text{CnV} = \text{CNTIN}$ )
- 0% 至 100% 之间的 EPWM 信号 (如果  $\text{CNTIN} < \text{CnV} \leq \text{MOD}$ )
- 100% EPWM 信号 (如果  $\text{CNTIN} > \text{CnV}$  或  $\text{CnV} > \text{MOD}$ )。

## 26.4.7 中心对齐 PWM (CPWM)模式

在以下情形中会选择中心对齐模式：

- DECAPEN = 0
- COMBINE = 0, 且
- CPWMS = 1

CPWM 脉宽（占空比）取决于  $2 \times (CnV - CNTIN)$ , 周期取决于  $2 \times (MOD - CNTIN)$ 。参见下图。MOD 必须保持在 0x0001 到 0x7FFF 范围内，因为此范围以外的值会产生不明确的结果。

在 CPWM 模式下，FTM 计数器在达到 MOD 之前会一直向上计数，然后一直向下计数，直至达到 CNTIN。

当 FTM 向下计数（脉宽开始）和 FTM 向上计数（脉宽结束）时，会在通道(n)匹配（FTM 计数器 = CnV）情况下置位 CHnF 位并生成通道(n)中断。

这种类型的 PWM 信号称为中心对齐，因为所有通道的脉宽中心都与 CNTIN 的值对齐。

其他通道模式与自上而下的计数器不兼容(CPWMS = 1)。因此，当(CPWMS = 1)时，必须在 CPWM 模式下使用所有 FTM 通道。

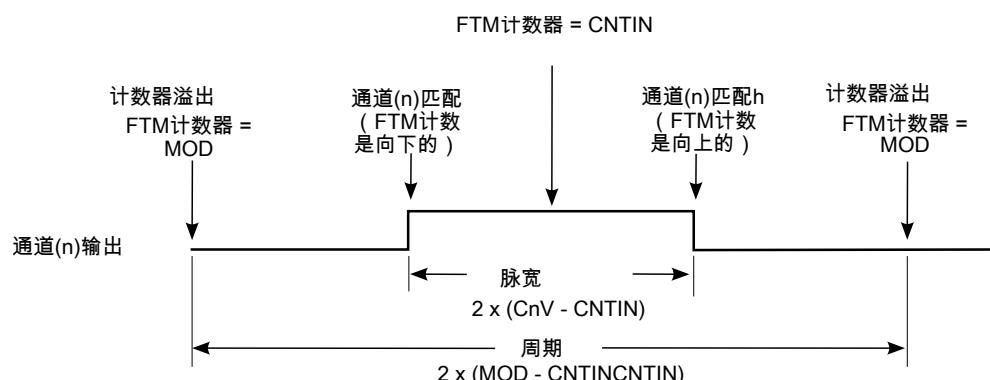


图 26-20. ELSnB:ELSnA = 1:0 条件下的 CPWM 周期和脉宽

如果 FTM 计数器达到 CnV 寄存器中的值时满足(ELSnB:ELSnA = 0:0)条件，则 CHnF 位置位并生成通道(n)中断（如果 CHnIE = 1），但是，通道(n)输出不由 FTM 控制。

如果(ELSnB:ELSnA = 1:0)，那么，通道(n)输出会在向下计数时与通道(n)匹配（FTM 计数器 = CnV）的情况下强制为高电平，在向上计数时与通道(n)匹配的情况下强制为低电平。参见下图。



图 26-21. ELSnB:ELSnA = 1:0 条件下的 CPWM 信号

如果( $\text{ELSnB}:\text{ELSnA} = X:1$ )，那么，通道(n)输出会在向下计数时与通道(n)匹配 (FTM 计数器 = CnV) 的情况下强制为低电平，在向上计数时与通道(n)匹配的情况下强制为高电平。参见下图。



图 26-22. ELSnB:ELSnA = X:1 条件下的 CPWM 信号

如果( $\text{CnV} = 0x0000$ )或  $\text{CnV}$  为负值，即( $\text{CnV}[15] = 1$ )，则通道(n)输出为 0% 占空因数 CPWM 信号，而且  $\text{CHnF}$  位不置位，即便存在通道(n)匹配也依然如此。

如果  $\text{CnV}$  为正值，即( $\text{CnV}[15] = 0$ )、( $\text{CnV} \geq \text{MOD}$ )且( $\text{MOD} \neq 0x0000$ )，则通道(n)输出为 100% 占空因数的 CPWM 信号，而且  $\text{CHnF}$  位不置位，即便存在通道(n)匹配也依然如此。这就意味着， $\text{MOD}$  设置的周期可用范围是 0x0001 到 0x7FFE，如果不生成 100% 占空因数 CPWM 信号，则为 0x7FFF。这并非一项重大限制，因为生成的周期比正常应用所需的周期要长得多。

FTM 计数器为自由运行计数器时，不得使用 CPWM 模式。

## 26.4.8 组合模式

在以下情形中会选择组合模式：

- DECAPEN = 0
- COMBINE = 1，且
- CPWMS = 0

在组合模式下，一个偶数通道(n)和相邻的奇数通道(n+1)组合起来生成一个 PWM 信号供通道(n)输出。

在组合模式下，PWM 周期取决于( $\text{MOD} - \text{CNTIN} + 0x0001$ )，PWM 脉宽（占空比）取决于( $|C(n+1)V - C(n)V|$ )。

通道(n)匹配 (FTM 计数器 =  $C(n)V$ ) 时， $\text{CHnF}$  位将置位并生成通道(n)中断 (条件是  $\text{CHnIE} = 1$ )。通道(n+1)匹配 (FTM 计数器 =  $C(n+1)V$ ) 时， $\text{CH}(n+1)\text{F}$  位将置位并生成通道(n+1)中断 (条件是  $\text{CH}(n+1)\text{IE} = 1$ )。

如果( $\text{ELSnB:ELSnA} = 1:0$ )，则在周期开始 (FTM 计数器 =  $\text{CNTIN}$ )、通道(n+1)匹配 (FTM 计数器 =  $C(n+1)V$ ) 时强制通道(n)输出为低电平。在通道(n)匹配 (FTM 计数器 =  $C(n)V$ ) 时强制为高电平。参见下图。

如果( $\text{ELSnB:ELSnA} = X:1$ )，则在周期开始 (FTM 计数器 =  $\text{CNTIN}$ )、通道(n+1)匹配 (FTM 计数器 =  $C(n+1)V$ ) 时强制通道(n)输出为高电平。在通道(n)匹配 (FTM 计数器 =  $C(n)V$ ) 时强制为低电平。参见下图。

在组合模式下，生成通道(n)和(n+1)输出时不使用  $\text{ELS}(n+1)\text{B}$  和  $\text{ELS}(n+1)\text{A}$  位。但是，如果( $\text{ELSnB:ELSnA} = 0:0$ )，则通道(n)输出不由 FTM 控制；如果( $\text{ELS}(n+1)\text{B:ELS}(n+1)\text{A} = 0:0$ )，则通道(n+1)输出不由 FTM 控制。

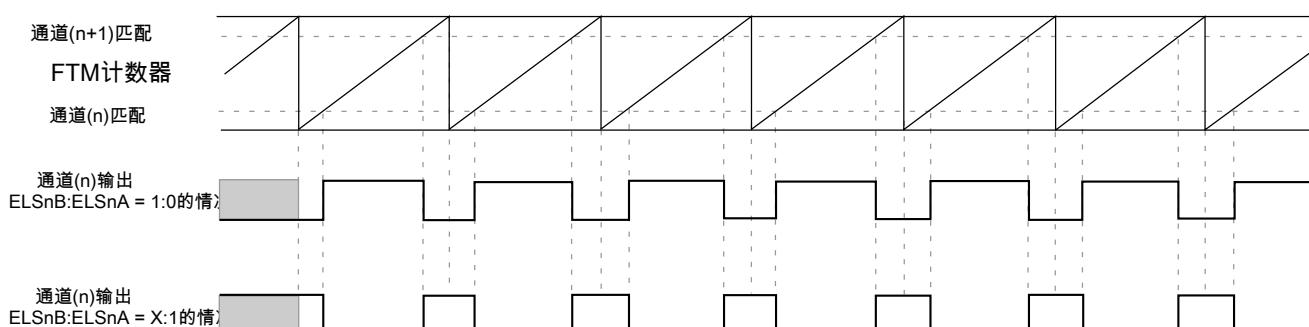


图 26-23. 组合模式

下图展示了利用组合模式生成 PWM 信号的过程。

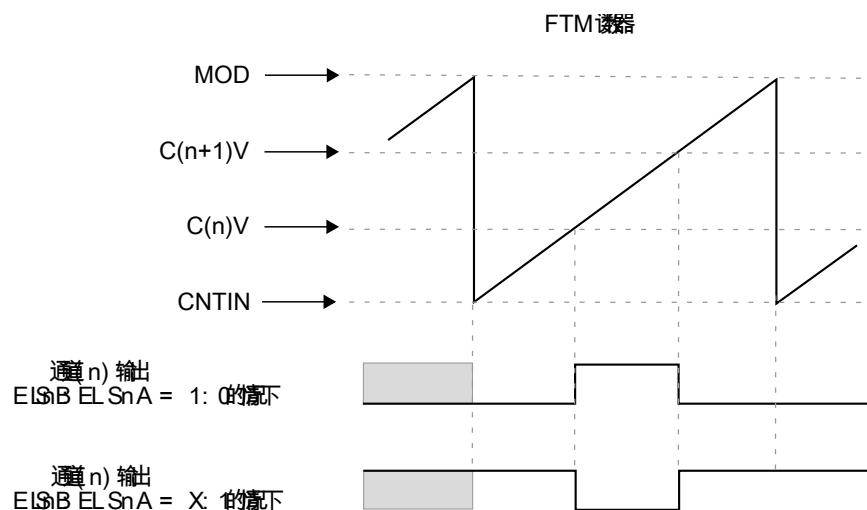


图 26-24. (CNTIN < C(n)V < MOD)、(CNTIN < C(n+1)V < MOD)且(C(n)V < C(n+1)V)条件下的通道(n)输出

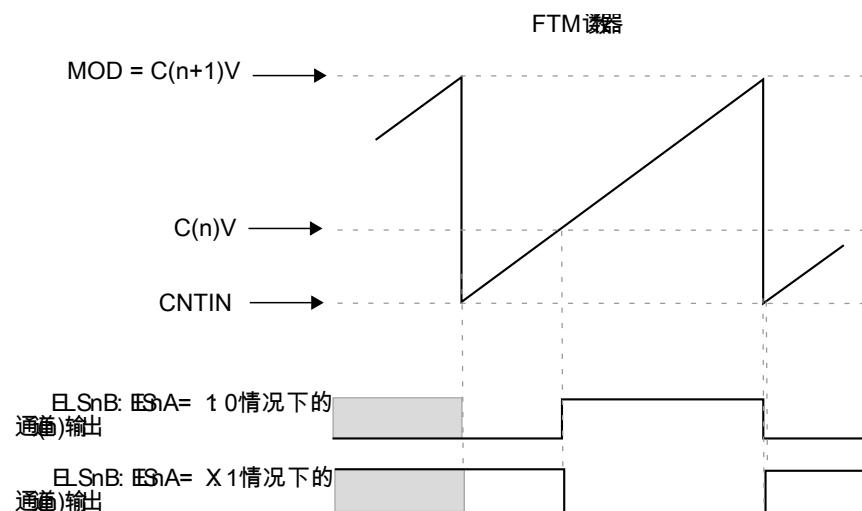


图 26-25. (CNTIN < C(n)V < MOD)且(C(n+1)V = MOD)条件下的通道(n)输出

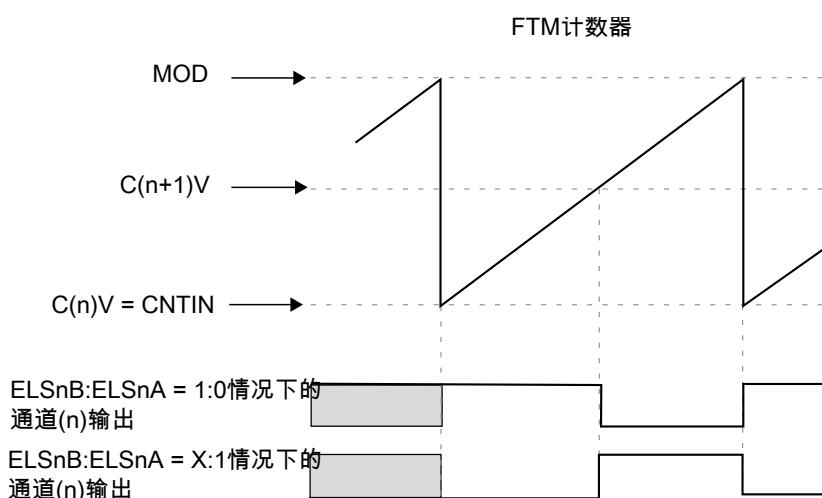


图 26-26. (C(n)V = CNTIN)且(CNTIN < C(n+1)V < MOD)条件下的通道(n)输出

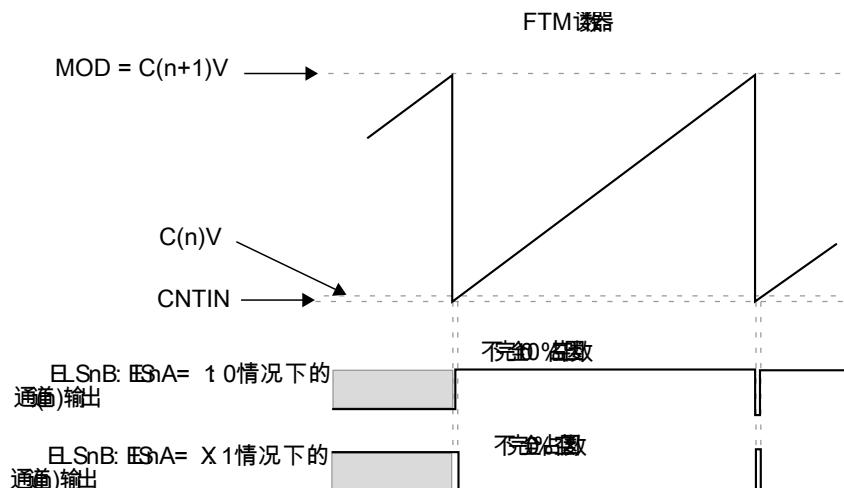


图 26-27. ( $CNTIN < C(n)V < MOD$ )、( $C(n)V$  约等于  $CNTIN$ ) 且 ( $C(n+1)V = MOD$ ) 条件下的通道(n)输出

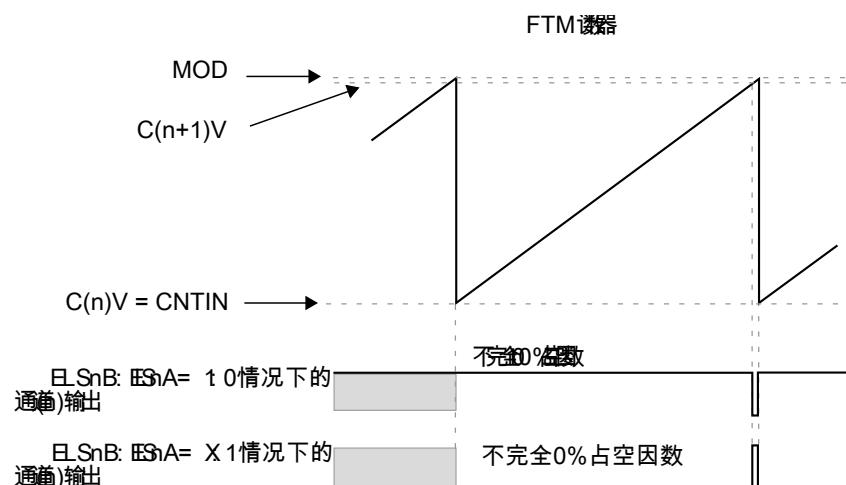


图 26-28. ( $C(n)V = CNTIN$ )、( $CNTIN < C(n+1)V < MOD$ ) 且 ( $C(n+1)V$  约等于  $MOD$ ) 条件下的通道(n)输出

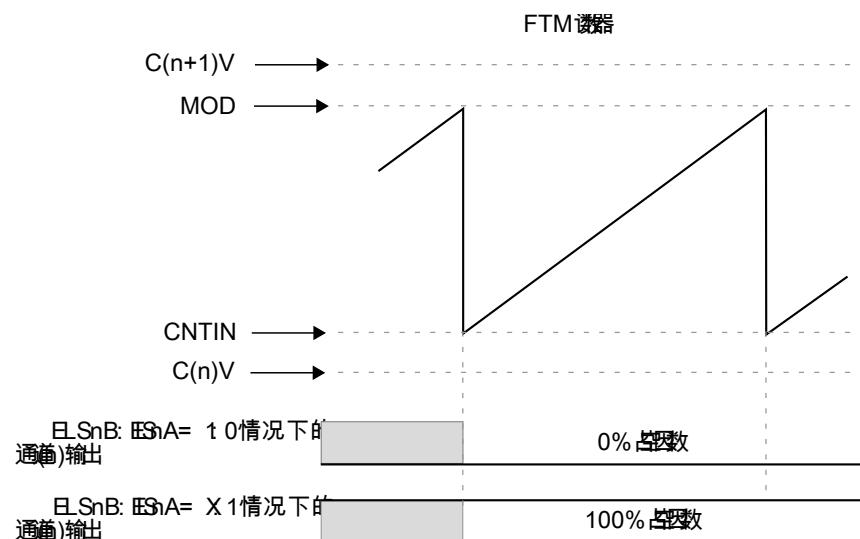


图 26-29. C(n)V 和 C(n+1)V 不在 CNTIN 和 MOD 之间条件下的通道(n)输出

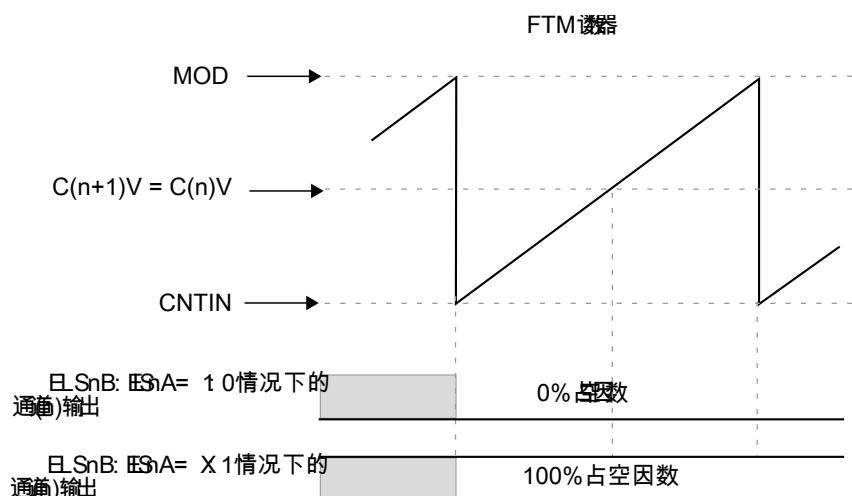


图 26-30. (CNTIN < C(n)V < MOD)、(CNTIN < C(n+1)V < MOD) 且 (C(n)V = C(n+1)V) 条件下的通道(n)输出

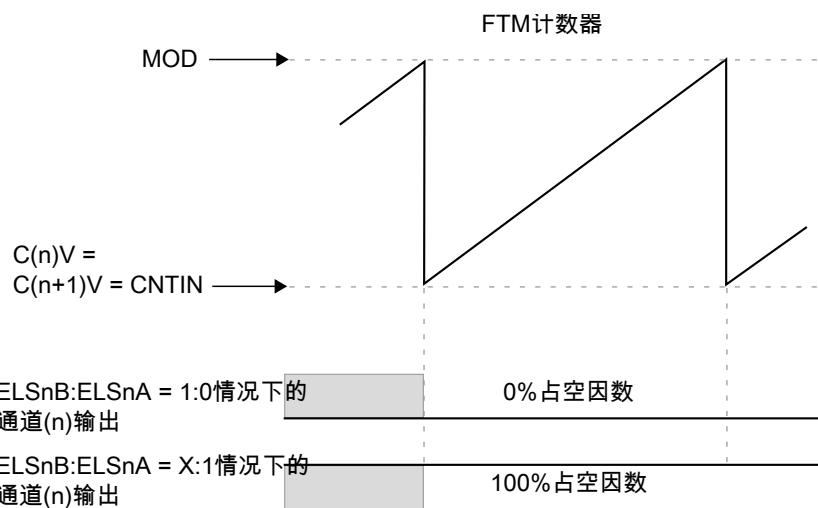


图 26-31. ( $C(n)V = C(n+1)V = CNTIN$ ) 条件下的通道(n)输出

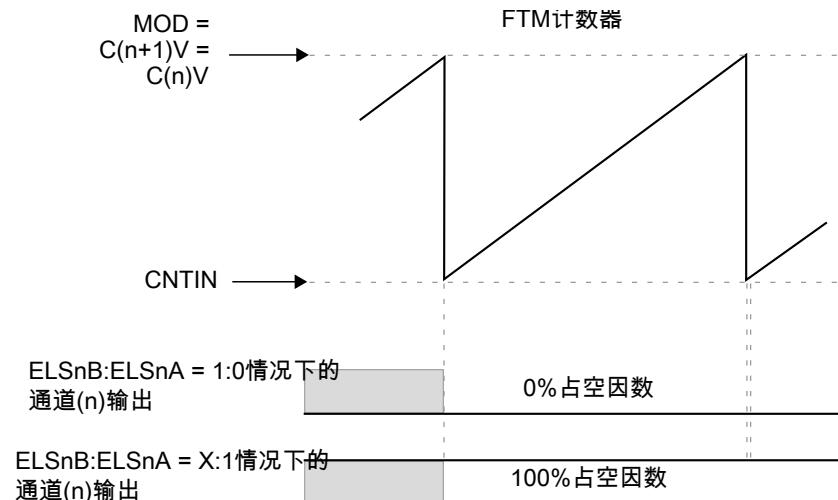


图 26-32. ( $C(n)V = C(n+1)V = MOD$ ) 条件下的通道(n)输出

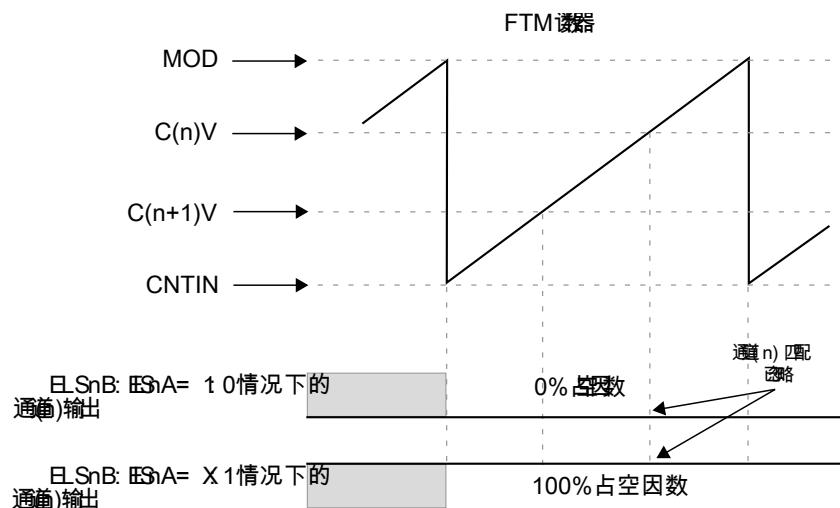


图 26-33. ( $CNTIN < C(n)V < MOD$ )、( $CNTIN < C(n+1)V < MOD$ ) 且 ( $C(n)V > C(n+1)V$ ) 条件下的通道(n)输出

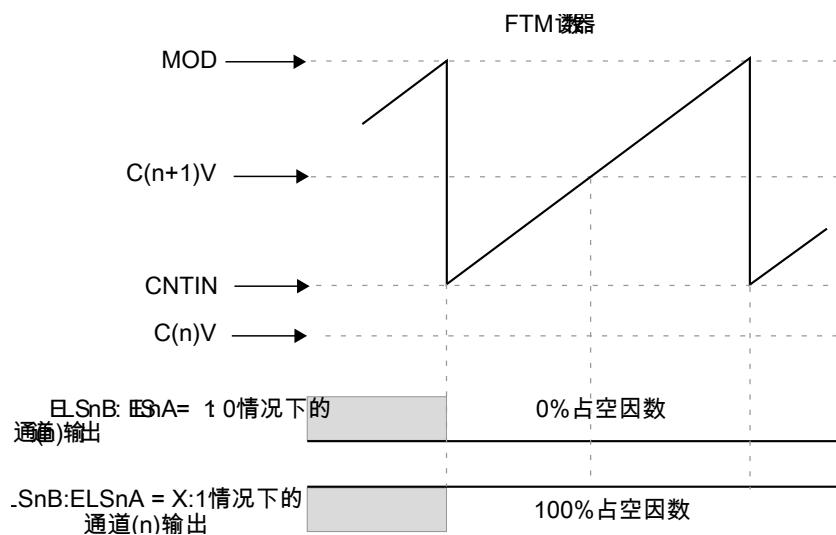


图 26-34. (C<sub>(n)V</sub> < CNTIN)且(CNTIN < C<sub>(n+1)V</sub> < MOD)条件下的通道(n)输出

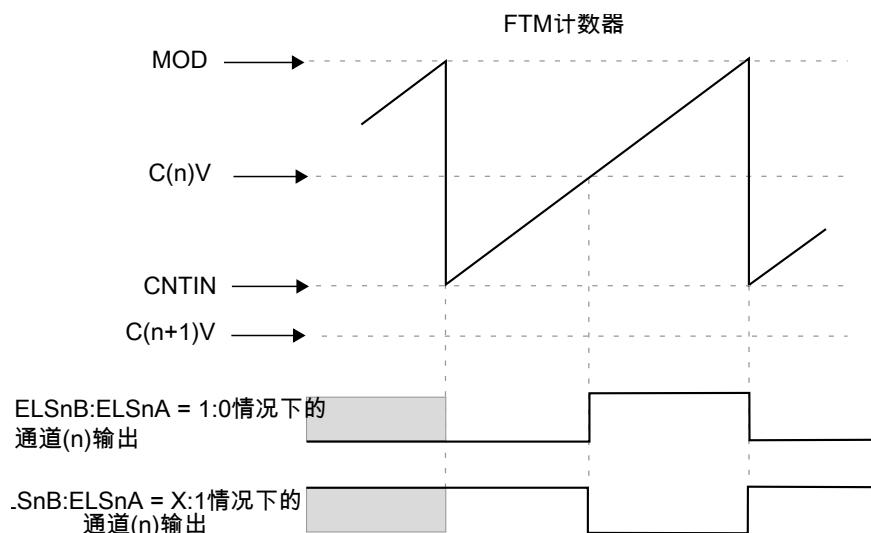


图 26-35. (C<sub>(n+1)V</sub> < CNTIN)且(CNTIN < C<sub>(n)V</sub> < MOD)条件下的通道(n)输出

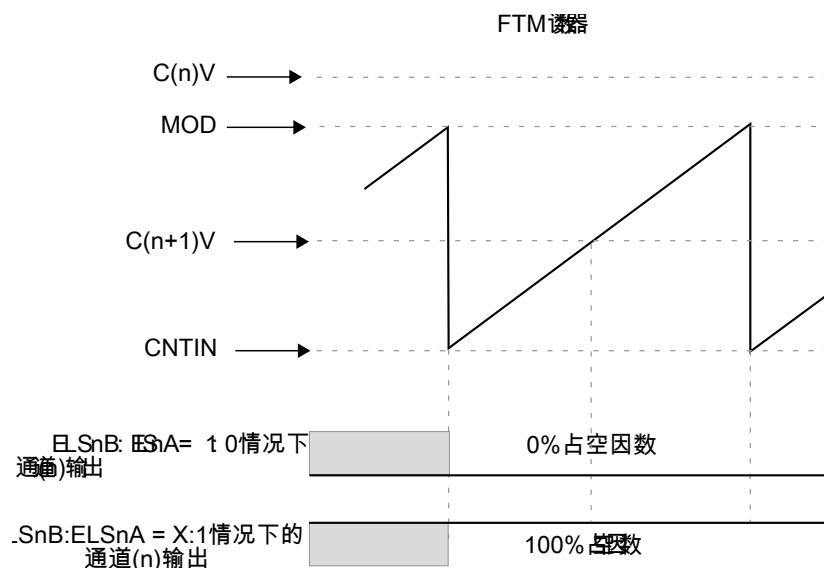


图 26-36. ( $C(n)V > MOD$ )且( $CNTIN < C(n+1)V < MOD$ )条件下的通道(n)输出

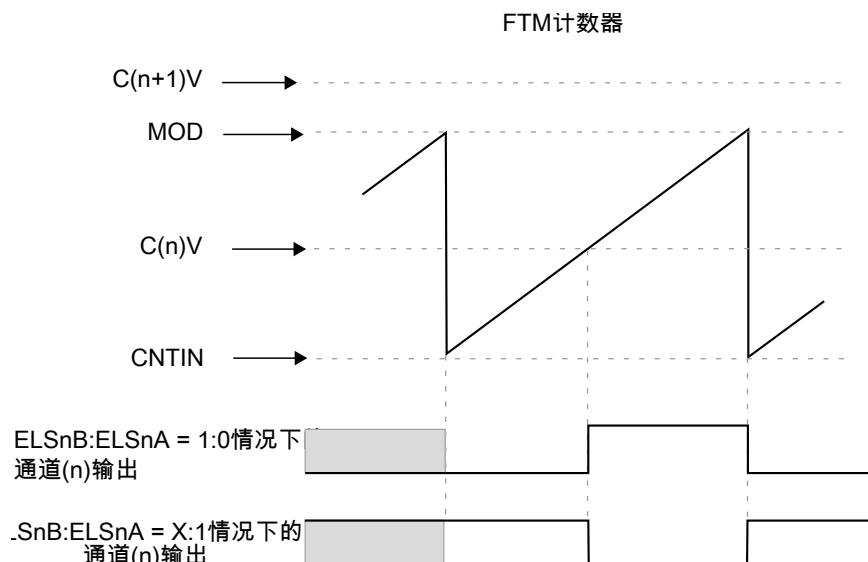


图 26-37. ( $C(n+1)V > MOD$ )且( $CNTIN < C(n)V < MOD$ )条件下的通道(n)输出

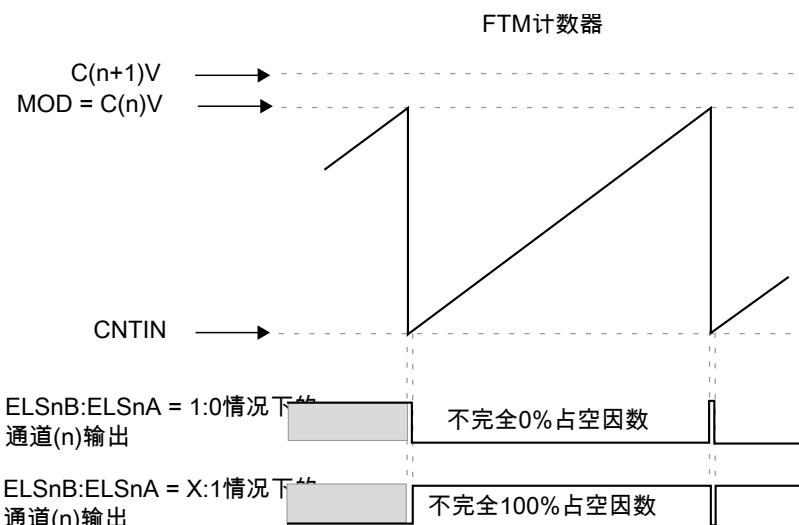


图 26-38. ( $C(n+1)V > MOD$ ) 且 ( $CNTIN < C(n)V = MOD$ ) 条件下的通道(n)输出

#### 26.4.8.1 不对称 PWM

在组合模式下，通道 n 发生匹配（即 FTM 计数器 =  $C(n)V$ ）控制 PWM 信号的第一边沿，通道  $n+1$  发生匹配（即 FTM 计数器 =  $C(n+1)V$ ）控制 PWM 信号的第二边沿，两者无关。因此，组合模式允许生成不对称的 PWM 信号。

#### 26.4.9 互补模式

选择互补模式的配置如下：

- DECAPE = 0
- COMP = 1

在互补模式下，通道( $n+1$ )输出与通道(n)输出电平相反。

通道( $n+1$ )输出与通道(n)输出相同的配置如下：

- DECAPE = 0
- COMP = 0

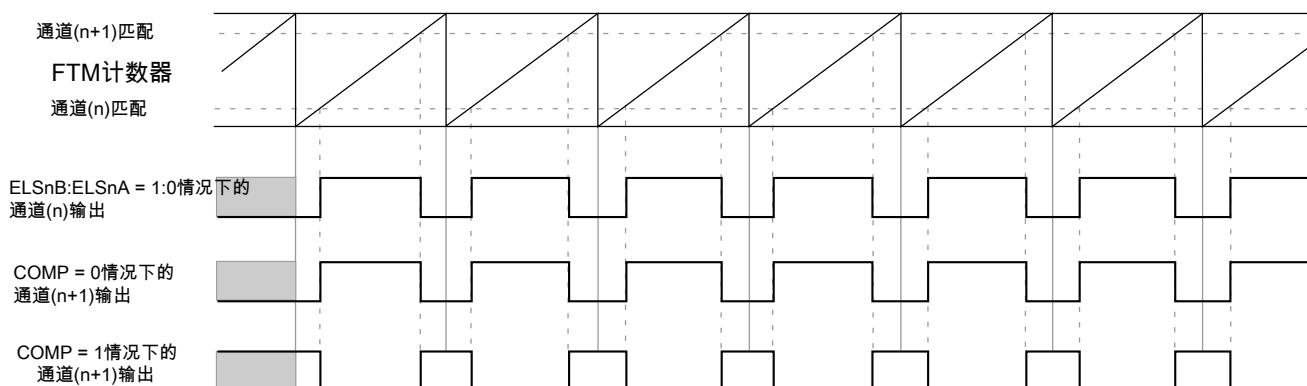


图 26-39. (ELSnB:ELSnA = 1:0) 条件下互补模式中的通道(n+1)输出

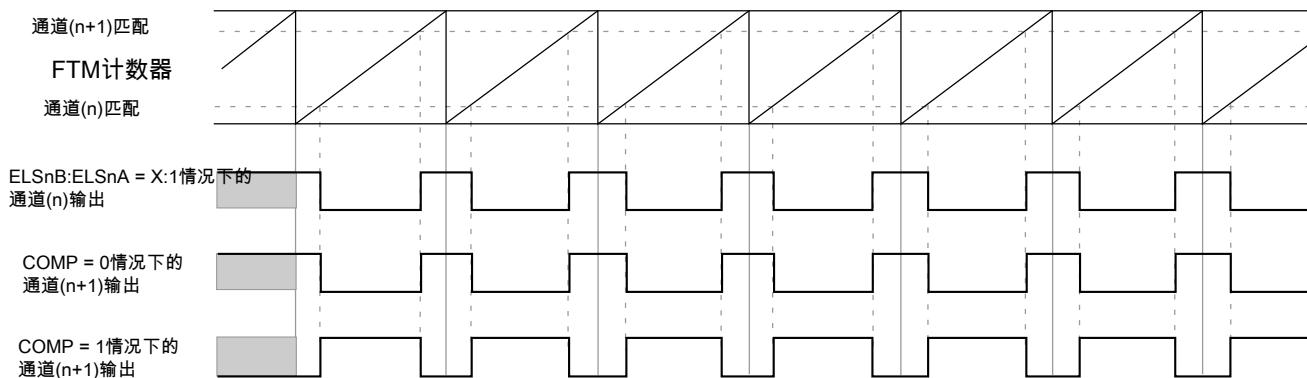


图 26-40. (ELSnB:ELSnA = X:1) 条件下互补模式中的通道(n+1)输出

## 注

互补模式不可用于输出比较模式。

### 26.4.10 通过写缓存更新的寄存器

#### 26.4.10.1 CNTIN 寄存器更新

下表介绍了何时更新 CNTIN 寄存器：

表 26-5. CNTIN 寄存器更新

满足条件	然后更新 CNTIN 寄存器
CLKS[1:0] = 0:0	对 CNTIN 寄存器采取写入操作时，与 FTMEN 位无关。
<ul style="list-style-type: none"> <li>• FTMEN = 0，或</li> <li>• CNTINC = 0</li> </ul>	对 CNTIN 采取写入操作之后的下一个系统时钟周期。
<ul style="list-style-type: none"> <li>• FTMEN = 1，</li> <li>• SYNCMODE = 1，且</li> <li>• CNTINC = 1</li> </ul>	参考 <a href="#">CNTIN 寄存器同步</a> 章节。

### 26.4.10.2 MOD 寄存器更新

下表介绍了何时更新 MOD 寄存器：

表 26-6. MOD 寄存器更新

满足条件	然后更新 MOD 寄存器
CLKS[1:0] = 0:0	对 MOD 寄存器采取写入操作时，与 FTMEN 位无关。
<ul style="list-style-type: none"> <li>• CLKS[1:0] ≠ 0:0, 且</li> <li>• FTMEN = 0</li> </ul>	<p>根据 CPWMS 位，也就是：</p> <ul style="list-style-type: none"> <li>• 如果选择的模式不是 CPWM，则 MOD 寄存器在对 MOD 寄存器采取写入操作且 FTM 计数器从 MOD 更改为 CNTIN 之后更新。如果 FTM 计数器处于自由运行计数器模式，则此更新在 FTM 计数器从 0xFFFF 更改为 0x0000 时发生。</li> <li>• 如果选择的模式为 CPWM，则 MOD 寄存器在对 MOD 寄存器采取写入操作且 FTM 计数器从 MOD 更改为(MOD – 0x0001)之后更新。</li> </ul>
<ul style="list-style-type: none"> <li>• CLKS[1:0] ≠ 0:0, 且</li> <li>• FTMEN = 1</li> </ul>	参考 <a href="#">MOD 寄存器同步</a> 章节。

### 26.4.10.3 CnV 寄存器更新

下表介绍了何时更新 CnV 寄存器：

表 26-7. CnV 寄存器更新

满足条件	然后更新 CnV 寄存器
CLKS[1:0] = 0:0	对 CnV 寄存器采取写入操作时，与 FTMEN 位无关。
<ul style="list-style-type: none"> <li>• CLKS[1:0] ≠ 0:0, 且</li> <li>• FTMEN = 0</li> </ul>	<p>根据选择的模式，即：</p> <ul style="list-style-type: none"> <li>• 如果选择的模式为输出比较，则 CnV 寄存器在下一次 FTM 计数器更改、预分频器计数结束时、对 CnV 寄存器采取写入操作之后更新。</li> <li>• 如果选择的模式为 EPWM，则 CnV 寄存器在对 CnV 寄存器采取写入操作且 FTM 计数器从 MOD 更改为 CNTIN 之后更新。如果 FTM 计数器处于自由运行计数器模式，则此更新在 FTM 计数器从 0xFFFF 更改为 0x0000 时发生。</li> <li>• 如果选择的模式为 CPWM，则 CnV 寄存器在对 CnV 寄存器采取写入操作且 FTM 计数器从 MOD 更改为(MOD – 0x0001)之后更新。</li> </ul>
<ul style="list-style-type: none"> <li>• CLKS[1:0] ≠ 0:0, 且</li> <li>• FTMEN = 1</li> </ul>	<p>根据选择的模式，即：</p> <ul style="list-style-type: none"> <li>• 如果选择的模式为输出比较，则 CnV 寄存器根据 SYNCEN 位更新。如果 (SYNCEN = 0)，则在下一次更改 FTM 计数器、预分频器计数结束时对 CnV 寄存器采取写入操作。如果(SYNCEN = 1)，则根据 <a href="#">C(n)V 和 C(n+1)V 寄存器同步</a> 更新 CnV 寄存器。</li> <li>• 如果选择的模式不是输出比较且(SYNCEN = 1)，则根据 <a href="#">C(n)V 和 C(n+1)V 寄存器同步</a> 更新 CnV 寄存器。</li> </ul>

## 26.4.11 PWM 同步

通过 PWM 同步将有机会以 MOD、CNTIN、CnV、OUTMASK、INVCTRL 和 SWOCTRL 寄存器各自的缓存值对这些寄存器进行更新，并强制 FTM 计数器为 CNTIN 寄存器值。

### 注

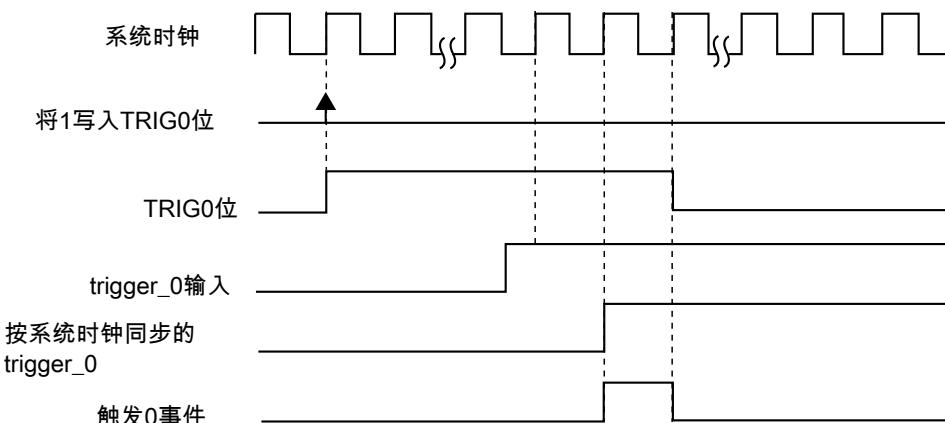
传统 PWM 同步(SYNCMODE = 0)是增强型 PWM 同步(SYNCMODE = 1)的子集。因此，只能使用增强型 PWM 同步。

### 26.4.11.1 硬件触发

$TRIG_n = 1$  (其中  $n = 0, 1$  或  $2$ , 各自对应于每一个输入信号) 时, 将使能 FTM 模块的三个硬件触发信号输入。硬件触发输入  $n$  由系统时钟同步。在使能的硬件触发输入上检测到上升沿时, 将启动采用硬件触发的 PWM 同步。

如果(HWTRIGMODE = 0), 则当 0 被写入或者触发  $n$  事件被检测到时, 对应的  $TRIG_n$  会被清零。

这种情况下, 如果使能了两个或更多硬件触发 (例如,  $TRIG_0$  和  $TRIG_1 = 1$ ), 但只发生了触发 1 事件, 则只清零  $TRIG_1$  位。如果在发生触发  $n$  事件的同时还通过写操作使  $TRIG_n$  位置位, 则会启动同步, 但  $TRIG_n$  位仍会因写操作而保持置位。



**注释**  
所有硬件触发输入都具有相同的行为方式。

图 26-41. HWTRIGMODE = 0 时的硬件触发事件

如果  $HWTRIGMODE = 1$ , 则  $TRIG_n$  位只在向其写 0 的情况下清零。

## 注

只有在采用增强型 PWM 同步的情况下(SYNCMODE = 1)，才能使 HWTRIGMODE 位为 1。

### 26.4.11.2 软件触发

向 SYNC[SWSYNC]位写入 1 时，将发生软件触发事件。向 SWSYNC 位写入 0 时即清零此位，由软件事件启动的 PWM 同步完成时，也会清零此位。

如果在上一个软件触发事件发起的 PWM 同步结束的同时（通过向 SWSYNC 位再次写入 1）再次发生软件触发事件，将开始新的 PWM 同步，且 SWSYNC 位保持等于 1。

如果 SYCMODE = 0，则 SWSYNC 位也会根据 PWMSYNC 和 REINIT 位由 FTM 清零。这种情况下，如果(PWMSYNC = 1)或 (PWMSYNC = 0 且 REINIT = 0)，SWSYNC 位将在软件触发事件发生后于所选的下一个加载点清零；参见[边界周期和加载点](#)和下图。如果(PWMSYNC = 0)且(REINIT = 1)，SWSYNC 位将在软件触发事件发生时清零。

如果 SYCMODE = 1，则 SWSYNC 位也会根据 SWRSTCNT 位由 FTM 清零。如果 SWRSTCNT = 0，SWSYNC 位将在软件触发事件发生后于所选的下一个加载点清零；参见下图。如果 SWRSTCNT = 1，SWSYNC 位将在软件触发事件发生时清零。

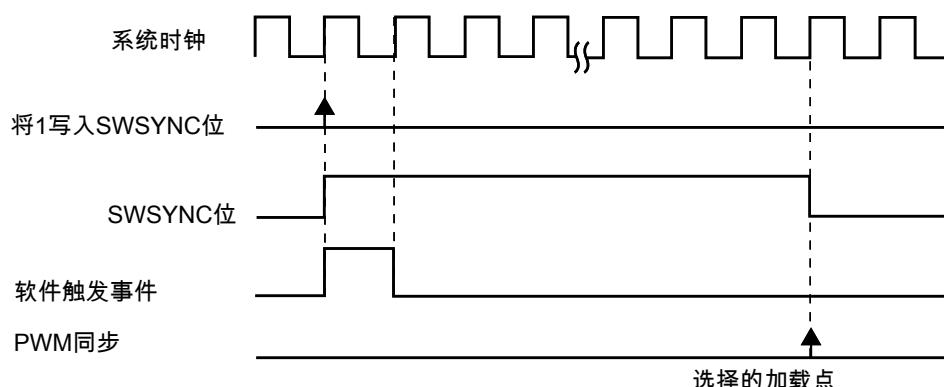


图 26-42. 软件触发事件

### 26.4.11.3 边界周期和加载点

边界周期定义对寄存器 MOD、CNTIN 和 C(n)V 的加载点很重要。

在[向上计数](#)模式下，边界周期定义为计数器变为其初始值(CNTIN)的时候。如果是在[向上-向下计数](#)模式下，边界周期则定义为计数器从向下计数变为向上计数的时候以及从向上计数变为向下计数的时候。

下表显示了寄存器的边界周期和加载点。在向上计数模式中，如果有一个 CNTMIN 或 CTMAX 位为 1，则使能加载点。在自上而下计数模式中，如图所示，由 CNTMIN 和 CNTMAX 位选择加载点。这些加载点对寄存器更新而言是一个安全之处，从而可在 PWM 波形生成过程中实现顺畅的过渡。

在这两种计数模式中，如果 CNTMIN 和 CNTMAX 都不是 1，则边界周期不用作寄存器更新的加载点。有关详细信息，请参见以下各节中的寄存器同步说明。

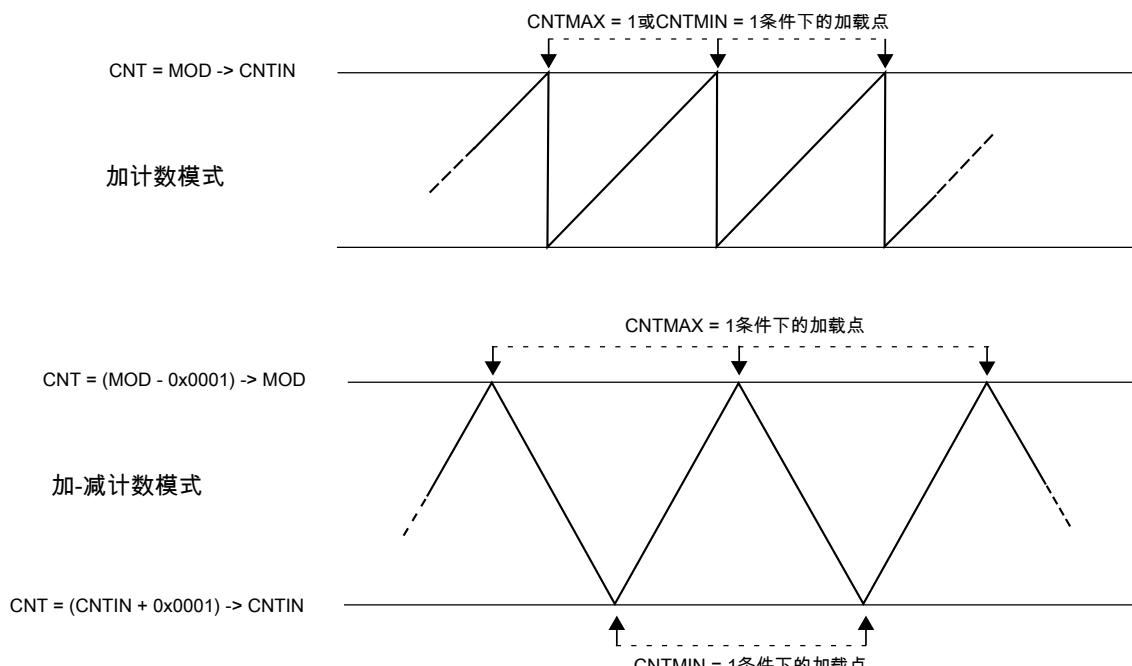


图 26-43. 边界周期和加载点

#### 26.4.11.4 MOD 寄存器同步

MOD 寄存器同步将以其缓存值对 MOD 寄存器进行更新。如果(FTMEN = 1)，将使能这种同步。

MOD 寄存器同步可通过增强型 PWM 同步(SYNCMODE = 1)或传统 PWM 同步(SYNCMODE = 0)完成。但是，更希望只通过增强型 PWM 同步来同步 MOD 寄存器。

采用增强型 PWM 同步时，依据下述流程图，MOD 寄存器同步取决于 SWWRBUF、SWRSTCNT、HWWRBUF 和 HWRSTCNT 位：

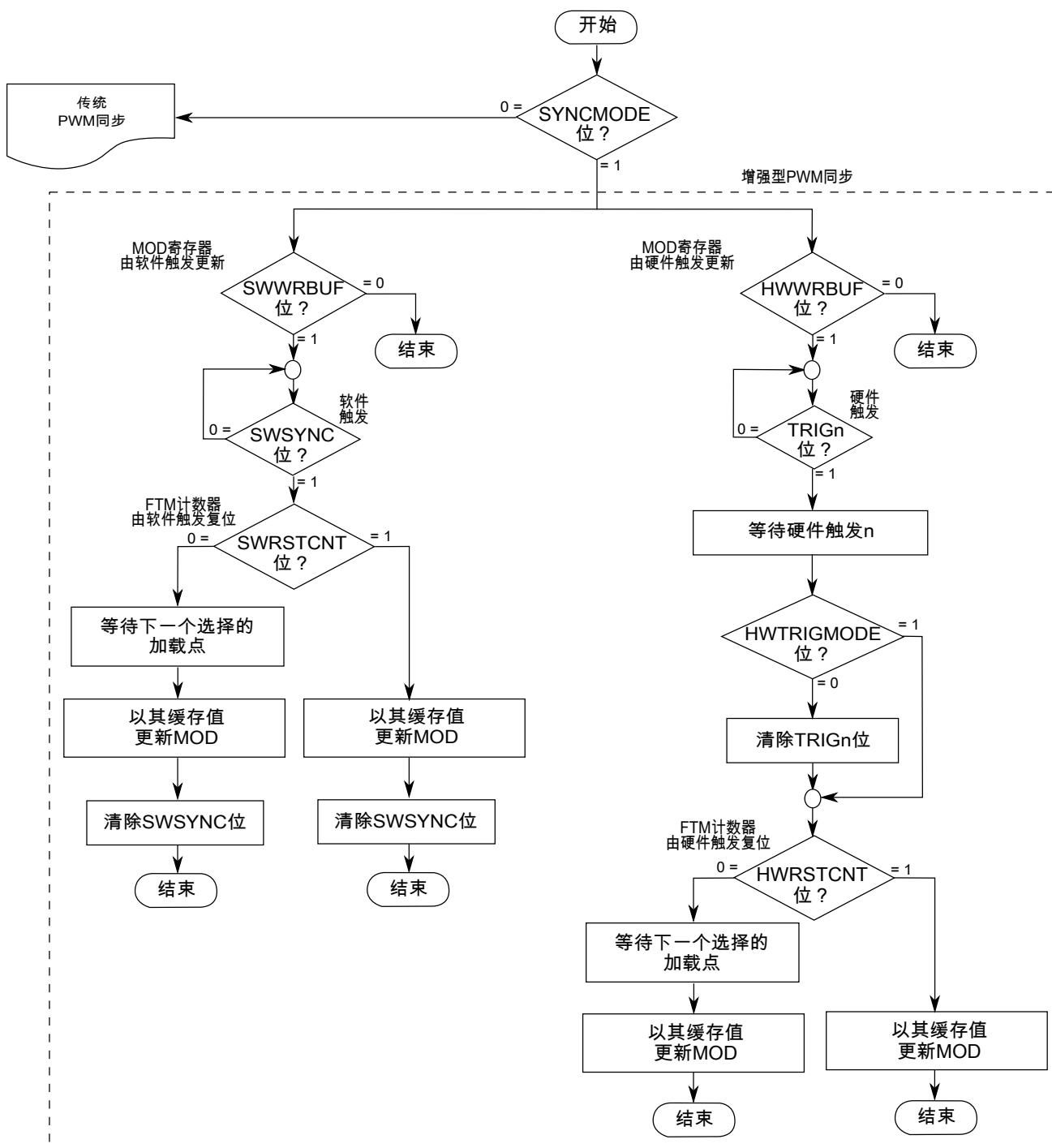


图 26-44. MOD 寄存器同步流程图

采用传统 PWM 同步时，依据下述说明，MOD 寄存器同步取决于 PWMSYNC 和 REINIT 位。

如果(SYNCMODE = 0)、(PWMSYNC = 0)且(REINIT = 0)，则在发生所使能的触发事件之后、在选择的下一个加载点进行此同步。如果触发事件为软件触发，则在选择的下一个加载点清除 SWSYNC 位。如果触发事件为硬件触发，则根据硬件触发清除触发器使能位(TRIGN)。以下为采用软件和硬件触发的示例。

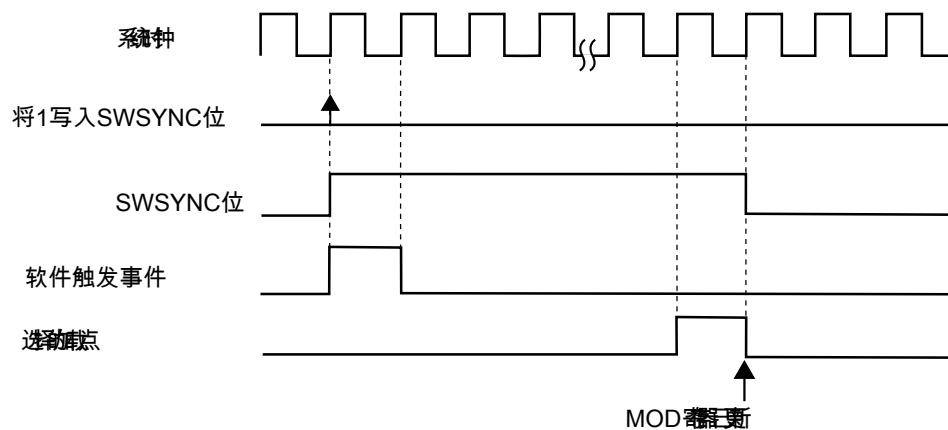


图 26-45. (SYNCMODE = 0)、(PWMSYNC = 0)、(REINIT = 0)且使用软件触发时的 MOD 同步

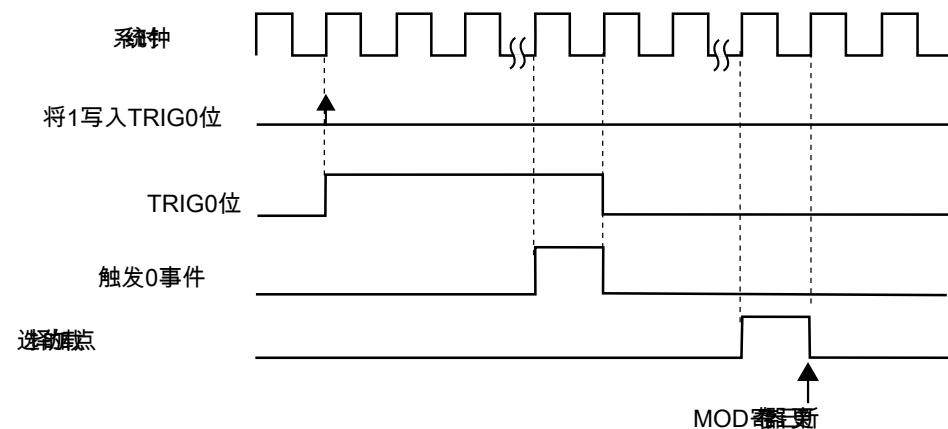


图 26-46. (SYNCMODE = 0)、(HWTRIGMODE = 0)、(PWMSYNC = 0)、(REINIT = 0)且使用硬件触发时的 MOD 同步

如果(SYNCMODE = 0)、(PWMSYNC = 0)且(REINIT = 1)，则在下一次发生所使能的触发事件时进行此同步。如果触发事件为软件触发，则根据以下示例清除 SWSYNC 位。如果触发事件为硬件触发，则根据硬件触发 清除 TRIGN 位。以下为采用软件和硬件触发的示例。

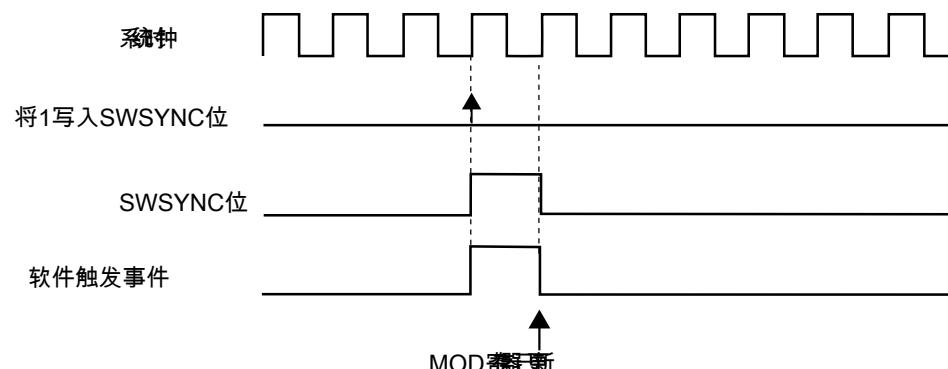


图 26-47. (SYNCMODE = 0)、(PWMSYNC = 0)、(REINIT = 1)且使用软件触发时的 MOD 同步

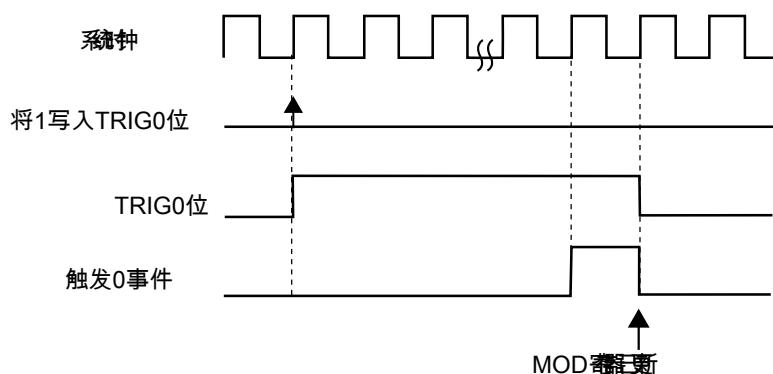


图 26-48. (SYNCMODE = 0)、(HWTRIGMODE = 0)、(PWMSYNC = 0)、(REINIT = 1)且使用硬件触发时的 MOD 同步

如果(SYNCMODE = 0)且(PWMSYNC = 1)，则在发生软件触发事件之后、在选择的下一个加载点进行此同步。在选择的下一个加载点清除 SWSYNC 位：

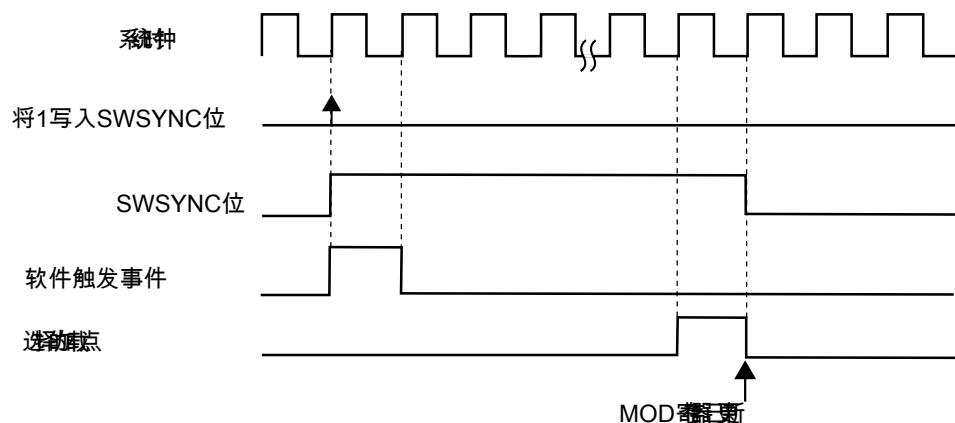


图 26-49. (SYNCMODE = 0)且(PWMSYNC = 1)时的 MOD 同步

### 26.4.11.5 CNTIN 寄存器同步

CNTIN 寄存器同步将以其缓存值对 CNTIN 寄存器进行更新。

如果(FTMEN = 1)、(SYNCMODE = 1)且(CNTINC = 1)，将使能这种同步。只能通过增强型 PWM 同步(SYNCMODE = 1)完成 CNTIN 寄存器同步。同步机制与通过增强型 PWM 同步完成的 MOD 寄存器同步相同；参见 [MOD 寄存器同步](#)。

### 26.4.11.6 C(n)V 和 C(n+1)V 寄存器同步

C(n)V 和 C(n+1)V 寄存器同步用其缓存值对 C(n)V 和 C(n+1)V 寄存器进行更新。

如果( $\text{FTMEN} = 1$ )且( $\text{SYNCEN} = 1$ )，则会使能该同步。同步机制与 [MOD 寄存器同步](#) 相同。但是，希望只通过增强型 PWM 同步( $\text{SYNCMODE} = 1$ )完成  $C(n)V$  和  $C(n+1)V$  寄存器的同步。

#### 26.4.11.7 OUTMASK 寄存器同步

OUTMASK 寄存器同步将以其缓存值对 OUTMASK 寄存器进行更新。

可通过增强型 PWM 同步 ( $\text{SYNCHOM} = 1$  且  $\text{SYNCMODE} = 1$ ) 或传统 PWM 同步 ( $\text{SYNCHOM} = 1$  且  $\text{SYNCMODE} = 0$ ) 在系统时钟的每个上升沿更新 OUTMASK 寄存器  $OM(m)$ : $\text{SYNCHOM} = 0$ 。但是，希望只通过增强型 PWM 同步来同步 OUTMASK 寄存器。

采用增强型 PWM 同步时, OUTMASK 寄存器同步取决于 SWOM 和 HWOM 位。参见以下流程图:

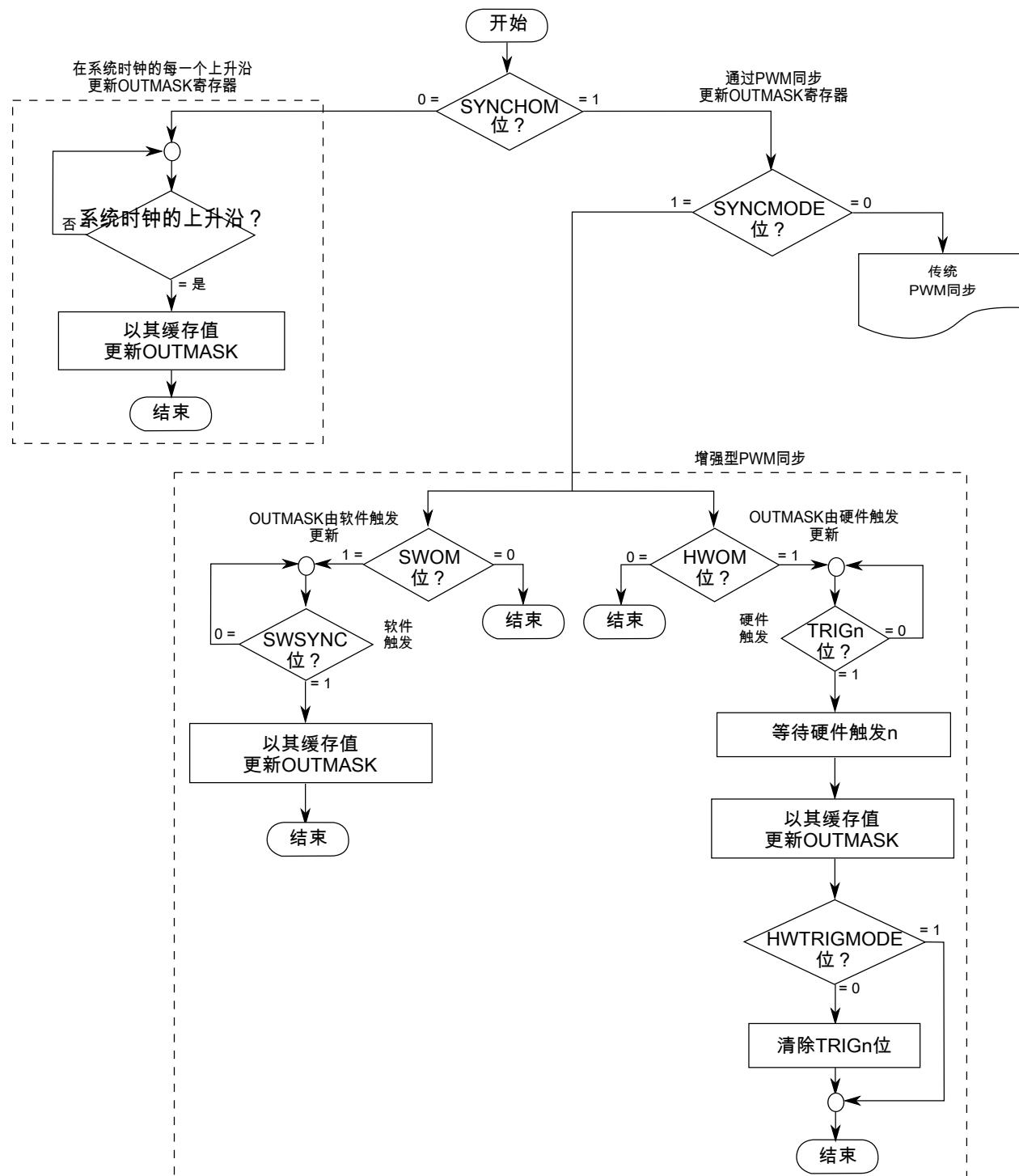


图 26-50. OUTMASK 寄存器同步流程图

采用传统 PWM 同步时，依据下述说明，OUTMASK 寄存器同步取决于 PWMSYNC 位。

如果(**SYNCMODE** = 0)、(**SYNCHOM** = 1)且(**PWMSYNC** = 0)，则在下一次发生所使能的触发事件时进行此同步。如果触发事件为软件触发，则在选择的下一个加载点清除 SWSYNC 位。如果触发事件为硬件触发，则根据**硬件触发** 清除 TRIGn 位。以下为采用软件和硬件触发的示例。

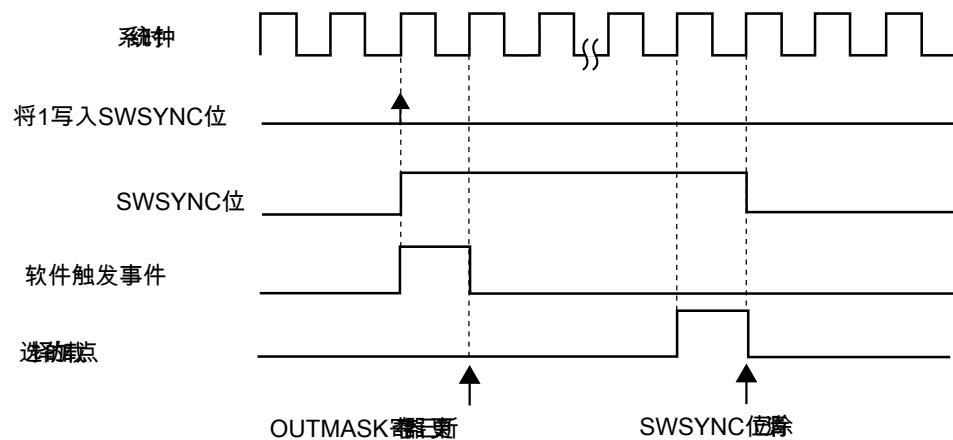


图 26-51. (**SYNCMODE** = 0)、(**SYNCHOM** = 1)、(**PWMSYNC** = 0)且使用软件触发时的 OUTMASK 同步

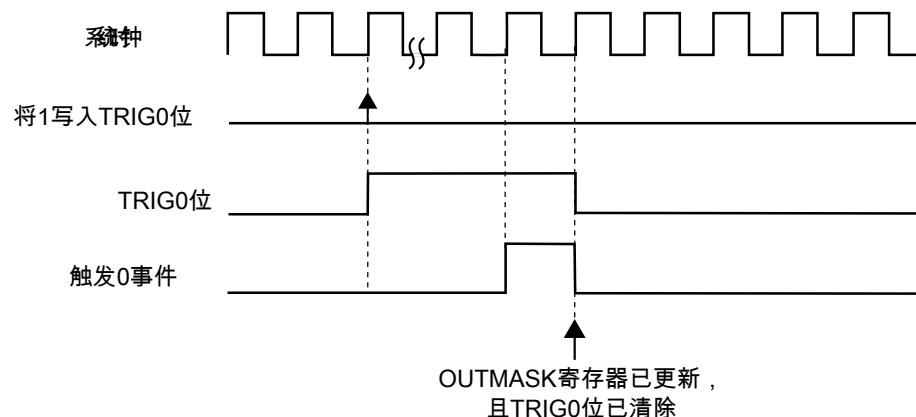


图 26-52. (**SYNCMODE** = 0)、(**HWTRIGMODE** = 0)、(**SYNCHOM** = 1)、(**PWMSYNC** = 0)且使用硬件触发时的 OUTMASK 同步

如果(**SYNCMODE** = 0)、(**SYNCHOM** = 1)且(**PWMSYNC** = 1)，则在下一次发生所使能的硬件触发事件时进行此同步。根据**硬件触发** 清除 TRIGn 位。以下是采用硬件触发的示例。

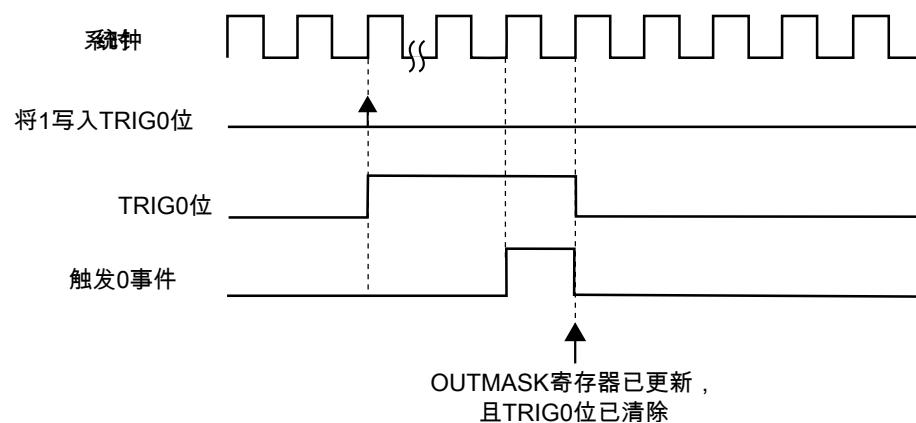


图 26-53. (SYNCMODE = 0)、(HWTRIGMODE = 0)、(SYNCHOM = 1)、(PWMSYNC = 1)且使用硬件触发时的 OUTMASK 同步

#### 26.4.11.8 INVCTRL 寄存器同步

INVCTRL 寄存器同步将通过其缓存值对 INVCTRL 寄存器进行更新。

INVCTRL 寄存器可在系统时钟的每个上升沿( $\text{INVC} = 0$ )更新，或者通过增强型 PWM 同步 ( $\text{INVC} = 1$  且  $\text{SYNCMODE} = 1$ ) 根据以下流程图更新。

采用增强型 PWM 同步时，INVCTRL 寄存器同步取决于 SWINVC 和 HWINVC 位。

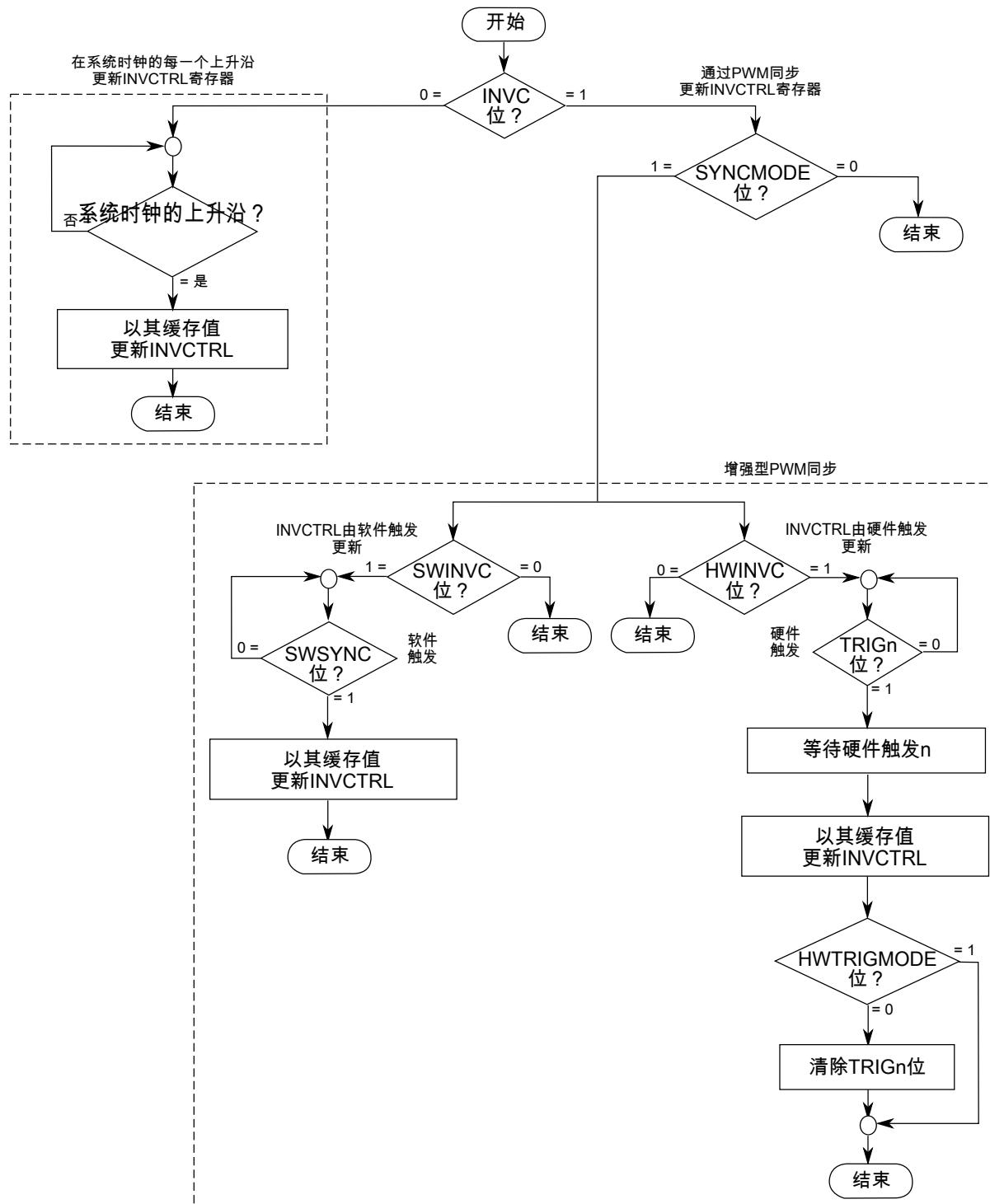


图 26-54. INVCTRL 寄存器同步流程图

#### 26.4.11.9 SWOCTRL 寄存器同步

SWOCTRL 寄存器同步将与其缓存值对 SWOCTRL 寄存器进行更新。

SWOCTRL 寄存器可在系统时钟的每个上升沿(SWOC = 0)更新，或者通过增强型 PWM 同步 (SWOC = 1 且 SYNCMODE = 1) 根据以下流程图更新。

采用增强型 PWM 同步时，SWOCTRL 寄存器同步取决于 SWSOC 和 HWSOC 位。

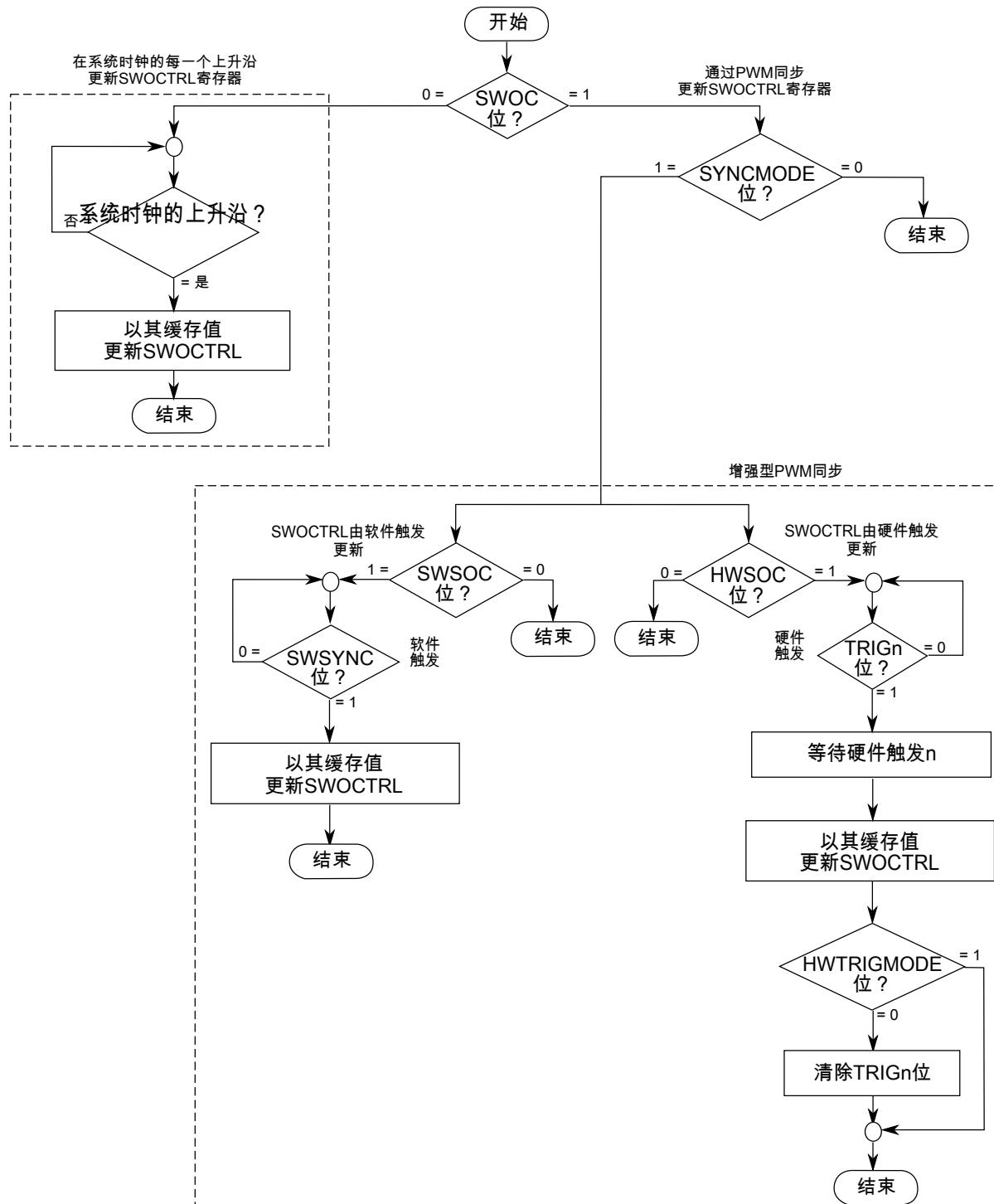


图 26-55. SWOCTRL 寄存器同步流程图

### 26.4.11.10 FTM 计数器同步

FTM 计数器同步是一种机制，通过这种机制 FTM 可以在 PWM 周期中的特定点重新开始生成 PWM。通道输出强制为各自的初始值（处于输出比较模式的通道则例外），FTM 计数器强制为 CNTIN 寄存器定义的初始计数值。

下图展示了 FTM 计数器同步情况。注意，同步事件发生后，通道(n)被设置为其初始值，通道(n+1)则不会被设置为其初始值，这是由此图中的特定时序造成的，在此图中死区插入阻止了该通道输出电平转换为 1。如果没有选择死区插入，则通道(n+1)在同步事件发生后会立即转换为逻辑值 1。

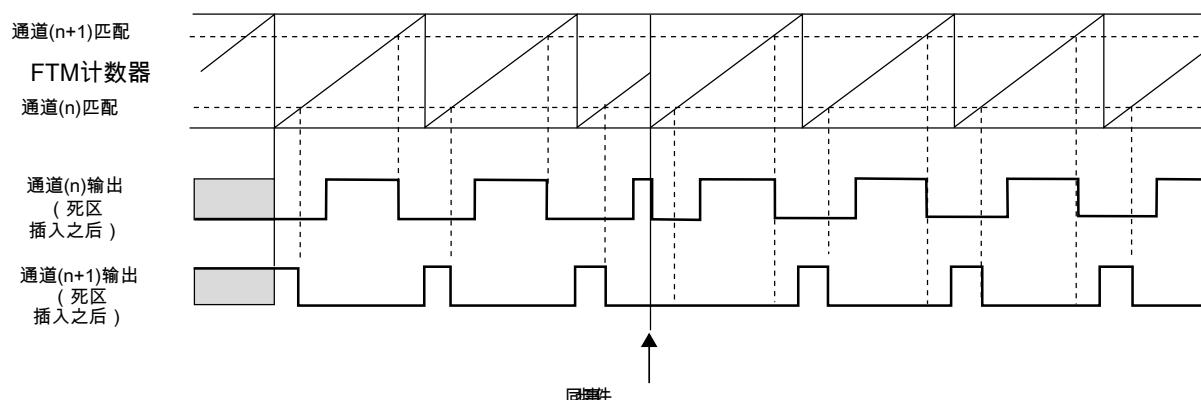


图 26-56. FTM 计数器同步

FTM 计数器同步可通过增强型 PWM 同步(SYNCMODE = 1)或传统 PWM 同步(SYNCMODE = 0)完成。但是，只能通过增强型 PWM 同步来同步 FTM 计数器。

采用增强型 PWM 同步时，依据下述流程图，FTM 计数器同步取决于 SWRSTCNT 和 HWRSTCNT 位。

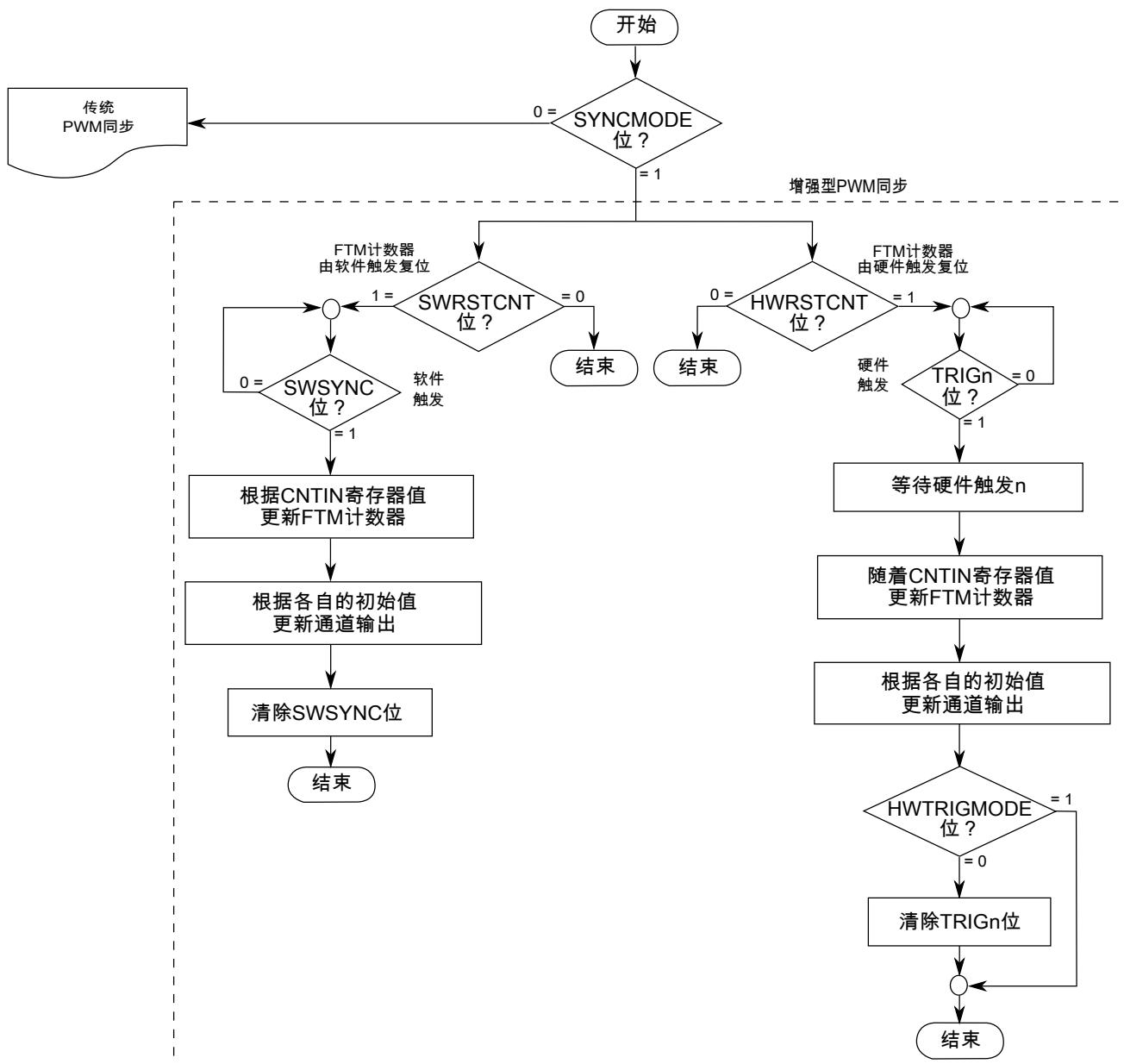


图 26-57. FTM 计数器同步流程图

采用传统 PWM 同步时，依据下述说明，FTM 计数器同步取决于 REINIT 和 PWMSYNC 位。

如果(**SYNCMODE** = 0)、(**REINIT** = 1)且(**PWMSYNC** = 0)，则在下一次触发事件使能时进行此同步。如果触发事件为软件触发，则根据以下示例清零 **SWSYNC** 位。如果触发事件为硬件触发，则根据**硬件触发** 清零 **TRIGn** 位。以下为采用软件和硬件触发的示例。

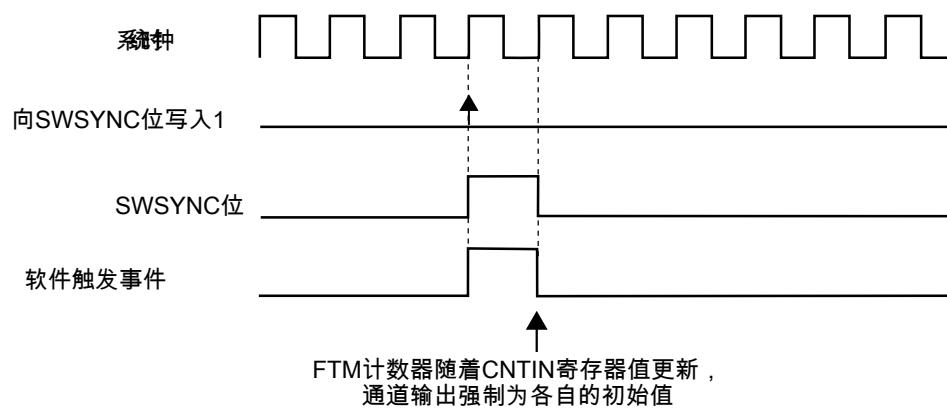


图 26-58. (SYNCMODE = 0)、(REINIT = 1)、(PWMSYNC = 0)且使用软件触发时的 FTM 计数器同步

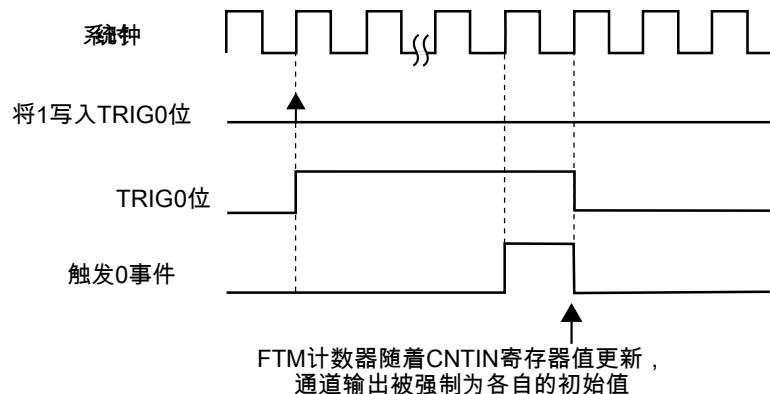


图 26-59. (SYNCMODE = 0)、(HWTRIGMODE = 0)、(REINIT = 1)、(PWMSYNC = 0)且使用硬件触发时的 FTM 计数器同步

如果(SYNCHMODE = 0)、(REINIT = 1)且(PWMSYNC = 1)，则在下一次硬件触发使能时进行此同步。根据[硬件触发](#)清零 TRIGn 位。

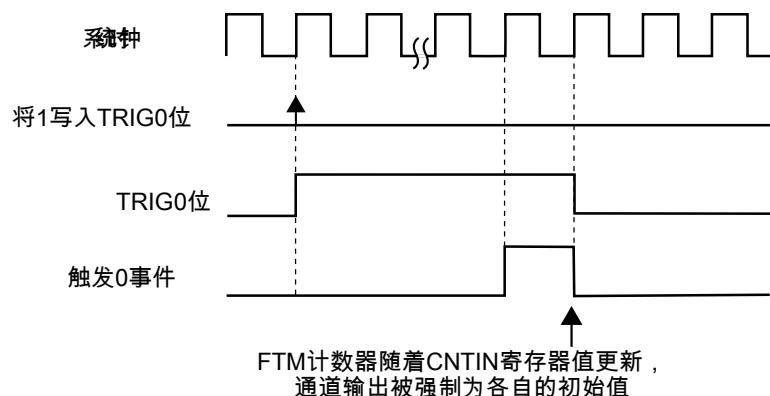


图 26-60. (SYNCHMODE = 0)、(HWTRIGMODE = 0)、(REINIT = 1)、(PWMSYNC = 1)且使用硬件触发时的 FTM 计数器同步

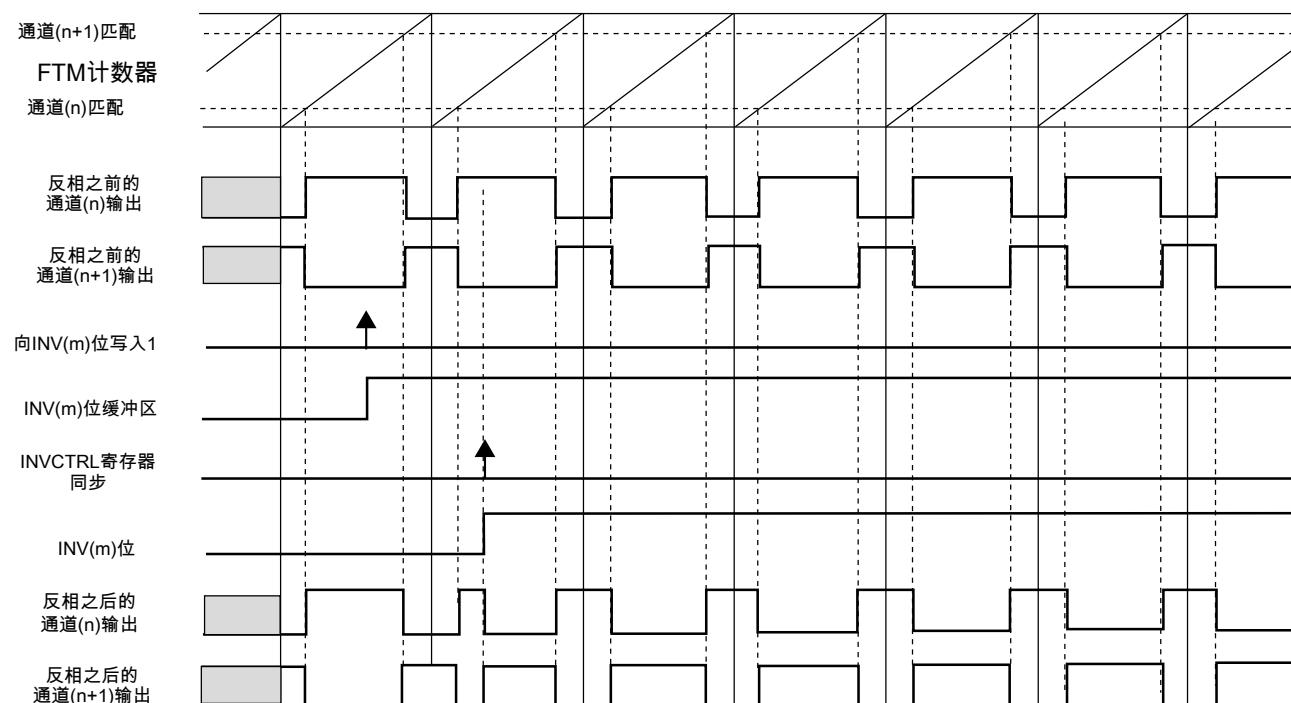
## 26.4.12 反相

反相功能在通道(n)和通道(n+1)输出之间交换信号。在以下情形中将选择反相操作：

- DECAPEN = 0
- COMP = 1 且
- INV<sub>m</sub> = 1 (其中 m 代表一对通道)

INVCTRL 寄存器中的 INV<sub>m</sub> 位将根据 [INVCTRL 寄存器同步](#) 以其缓冲值更新。

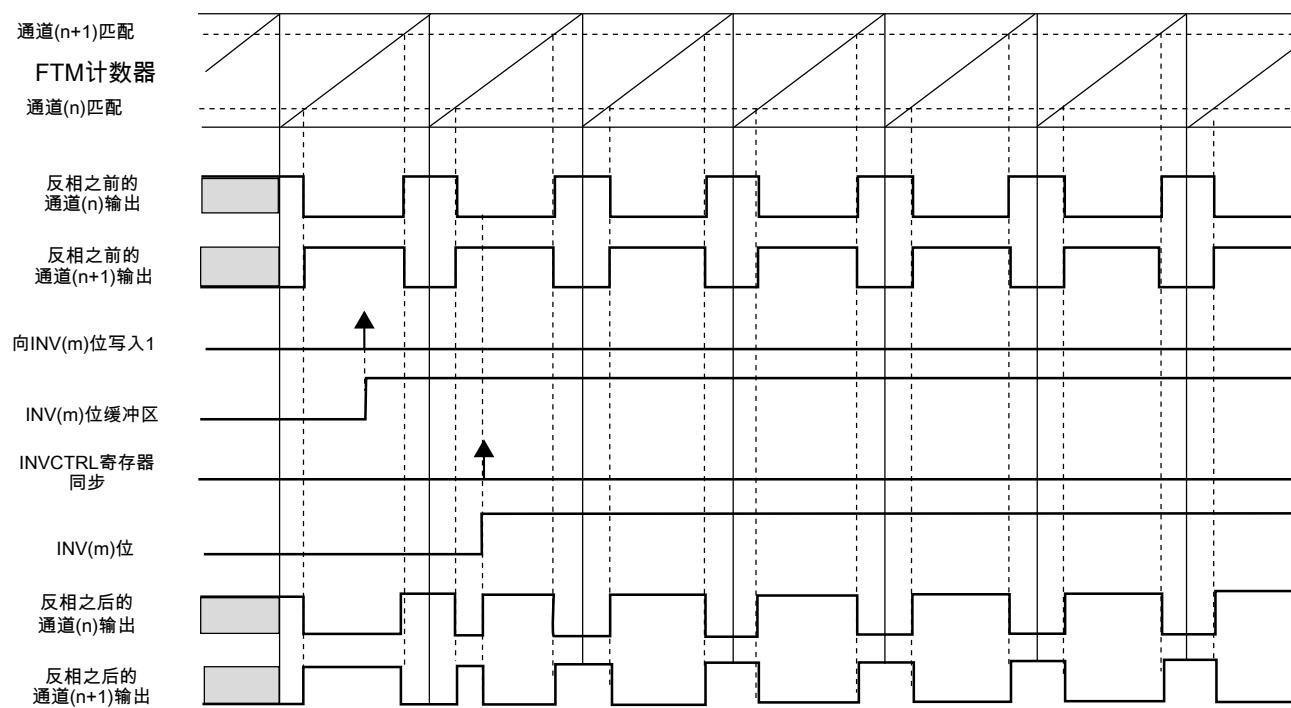
在高电平有效(ELSnB:ELSnA = 1:0)组合模式下，通道(n)输出在周期开始 (FTM 计数器 = CNTIN) 时被强制为低电平，在通道(n)匹配时被强制为高电平，在通道(n+1)匹配时被强制为低电平。如果选择了反相，通道(n)输出的行为方式将变更为：在 PWM 周期开始时被强制为高电平，在通道(n)匹配时被强制为低电平，在通道(n+1)匹配时被强制为高电平。参见下图。



**注释**  
INV(m)位选择通道对(n)和(n+1)的反相。

**图 26-61. 高电平有效(ELSnB:ELSnA = 1:0)组合模式下反相之后的通道(n)和(n+1)输出**

注意，应当考虑 ELSnB:ELSnA 位值，因为它们定义了通道输出的有效状态。在低电平有效(ELSnB:ELSnA = X:1)组合模式下，通道(n)输出在周期开始时被强制为高电平，在通道(n)匹配时被强制为低电平，在通道(n+1)匹配时被强制为高电平。选择反相后，通道(n)和(n+1)呈现的波形将如下图所示。



注释

INV(m)位选择通道对(n)和(n+1)的反相。

图 26-62. 低电平有效(ELSnB:ELSnA = X:1)组合模式下反相之后的通道(n)和(n+1)输出

### 注

反相功能不可用于输出比较模式。

#### 26.4.13 软件输出控制

软件输出控制可在 PWM 生成过程中于特定时间点强制通道输出软件定义的值。

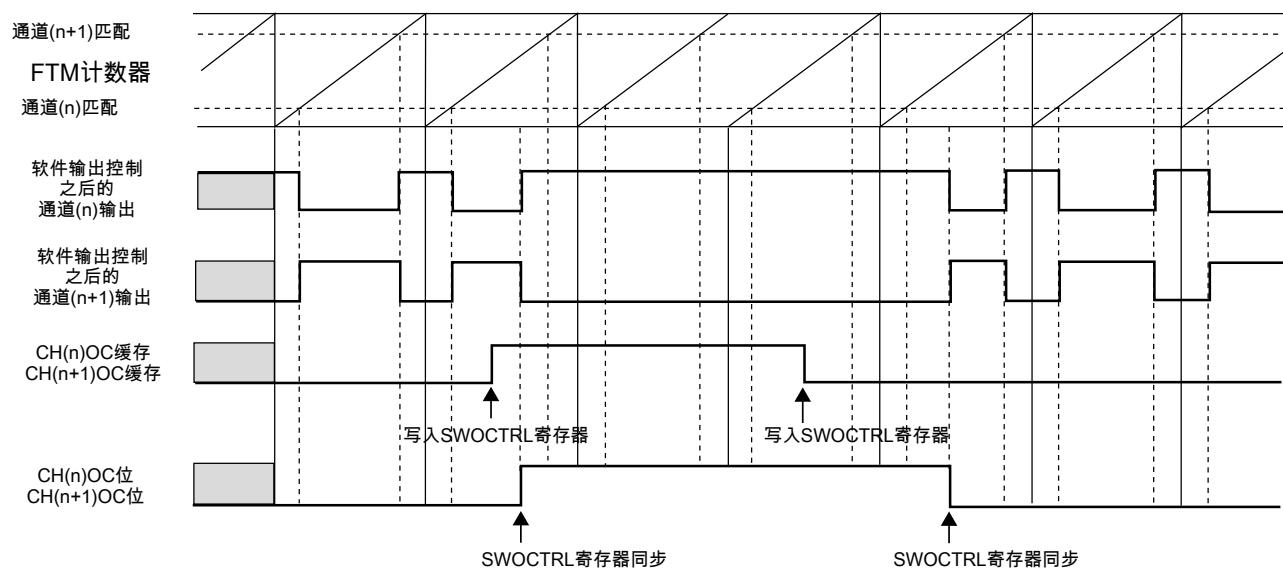
在以下情形中会选择软件输出控制：

- DECAPE = 0 且
- CHnOC = 1

CHnOC 位使能针对特定通道输出的软件输出控制，CHnOCV 选择强制此通道输出的值。

SWOCTRL 寄存器中的 CHnOC 和 CHnOCV 位均根据 SWOCTRL 寄存器同步 缓存，并以各自的缓冲区值更新。

下图展示了采用软件输出控制时的通道(n)和(n+1)输出信号。本例中，通道(n)和(n+1)设置为组合与互补模式。



注释

CH(n)OCV = 1 且 CH(n+1)OCV = 0。

图 26-63. 组合与互补模式下的软件输出控制示例

COMP 位为零时，软件输出控制强制通道(n)和(n+1)输出以下值。

表 26-8. (COMP = 0)时的软件输出控制行为方式

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	通道(n)输出	通道(n+1)输出
0	0	X	X	没有被 SWOC 修改	没有被 SWOC 修改
1	1	0	0	强制为零	强制为零
1	1	0	1	强制为零	强制为一
1	1	1	0	强制为一	强制为零
1	1	1	1	强制为一	强制为一

COMP 位为一时，软件输出控制强制通道(n)和(n+1)输出以下值。

表 26-9. (COMP = 1)时的软件输出控制行为方式

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	通道(n)输出	通道(n+1)输出
0	0	X	X	没有被 SWOC 修改	没有被 SWOC 修改
1	1	0	0	强制为零	强制为零
1	1	0	1	强制为零	强制为一
1	1	1	0	强制为一	强制为零
1	1	1	1	强制为一	强制为零

## 注

- CH(n)OC 和 CH(n+1)OC 位应该相等。

- 使能软件输出控制(即  $CH(n)OC = 1$  和/或  $CH(n+1)OC = 1$ )时, 不得修改 COMP 位。
- 软件输出控制的行为方式与处于禁用或使能状态的 FTM 计数器相同(参见状态和控制寄存器中的 CLKS 字段说明)。

## 26.4.14 死区插入

当( $DTEN = 1$ )且 ( $DTVAL[5:0]$ 为非零) 时, 将使能死区插入。

**DEADTIME** 寄存器定义可用于所有 FTM 通道的死区延迟。DTPS[1:0]位定义系统时钟的预分频器, DTVAL[5:0]位则定义死区模数, 即死区预分频器时钟的数量。

死区延迟插入可确保不会有两个互补信号(通道(n)和(n+1))同时驱动有效状态。

如果  $POL(n) = 0$ 、 $POL(n+1) = 0$  且死区处于使能状态, 那么在出现通道(n)匹配(FTM 计数器 =  $C(n)V$ )情况时, 通道(n)输出将保持低值, 直到通道(n)输出置位、死区延迟结束时。与此类似, 在出现通道(n+1)匹配(FTM 计数器 =  $C(n+1)V$ )情况时, 通道(n+1)输出将保持低值, 直到通道(n+1)输出置位、死区延迟结束时。参见下图。

如果  $POL(n) = 1$ 、 $POL(n+1) = 1$  且死区处于使能状态, 那么在出现通道(n)匹配(FTM 计数器 =  $C(n)V$ )情况时, 通道(n)输出将保持高值, 直到通道(n)输出清除、死区延迟结束时。与此类似, 在出现通道(n+1)匹配(FTM 计数器 =  $C(n+1)V$ )情况时, 通道(n+1)输出将保持高值, 直到通道(n+1)输出清除、死区延迟结束时。

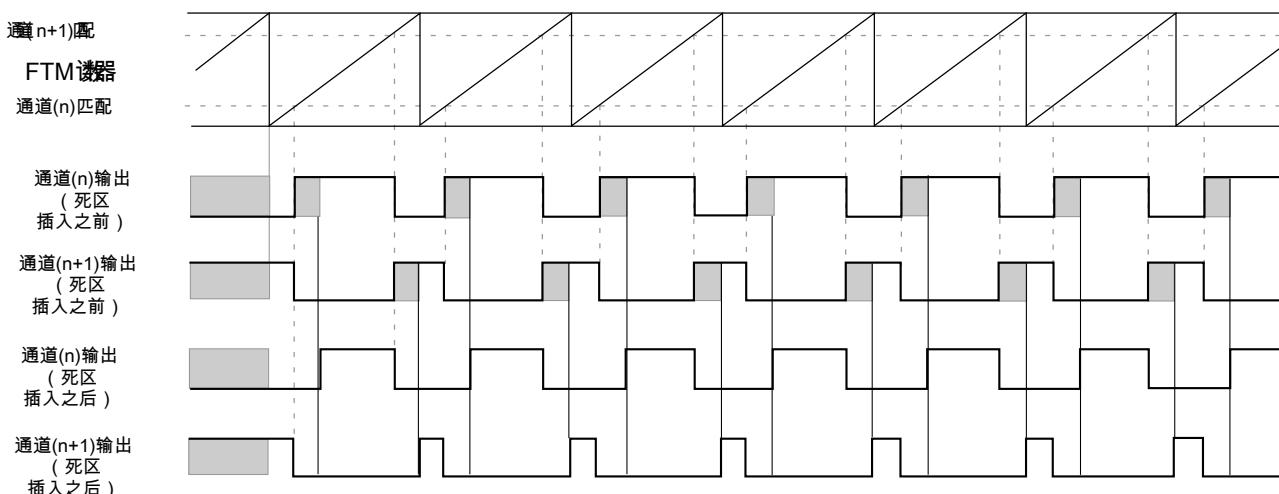


图 26-64. ELSnB:ELSnA = 1:0、 $POL(n) = 0$  且  $POL(n+1) = 0$  条件下的死区插入

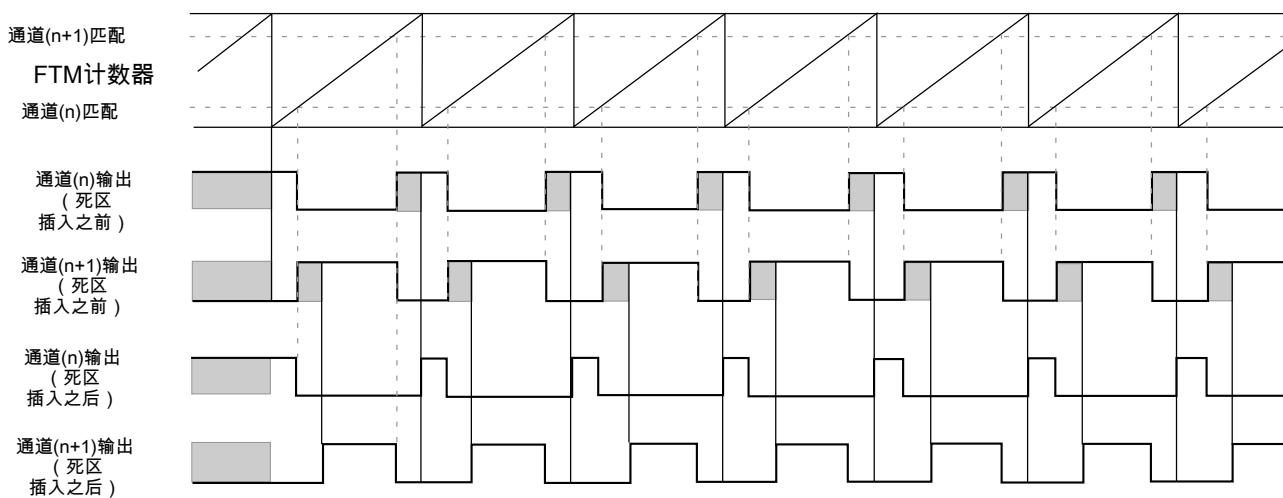


图 26-65.  $\text{ELSnB:ELSnA} = \text{X:1}$ 、 $\text{POL}(n) = 0$  且  $\text{POL}(n+1) = 0$  条件下的死区插入

### 注

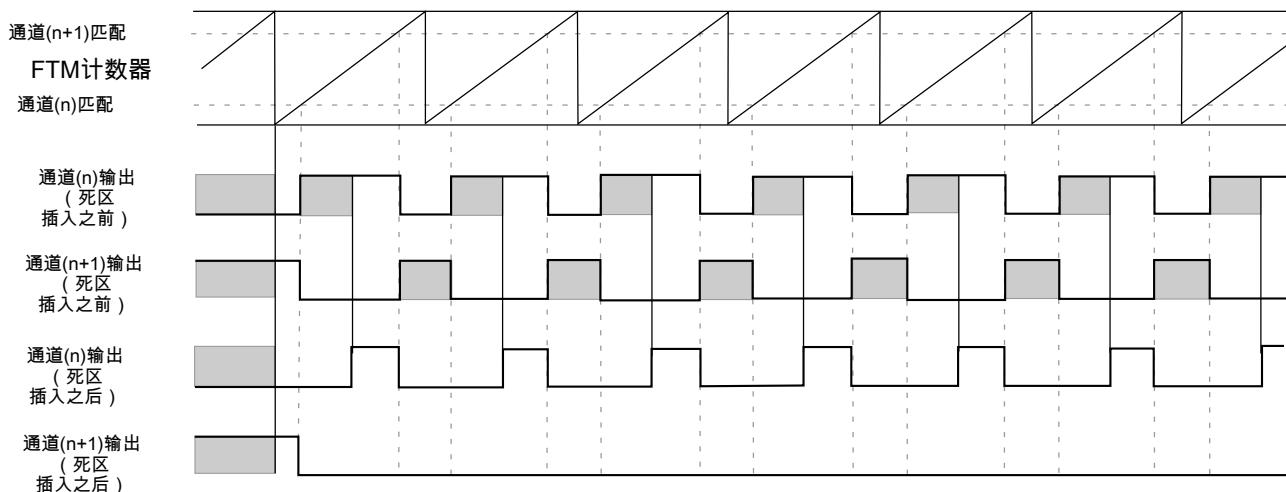
- 死区功能只能用于互补模式。
- 死区功能不可用于输出比较模式。

#### 26.4.14.1 死区插入个别案例

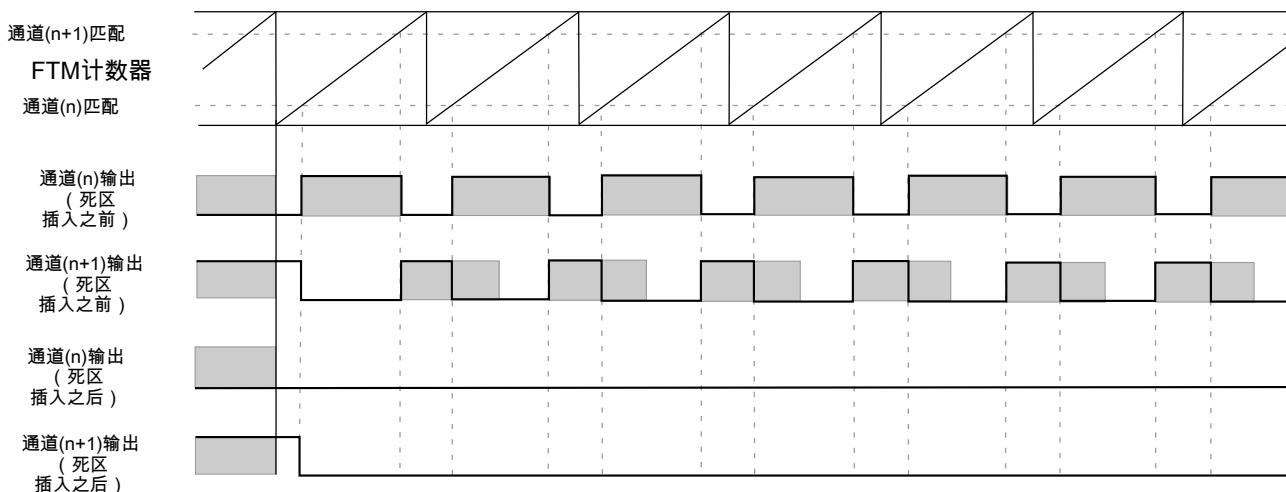
如果 ( $\text{PS}[2:0]$  已清零)、( $\text{DTFS}[1:0] = 0:0$  或  $\text{DTFS}[1:0] = 0:1$ ):

- 且死区延迟大于或等于通道(n)占空比  $((C(n+1)V - C(n)V) \times \text{系统时钟})$ , 则通道(n)输出始终为无效值 (POL(n)位值)。
- 且死区延迟大于或等于通道(n+1)占空比  $((\text{MOD} - \text{CNTIN} + 1 - (C(n+1)V - C(n)V)) \times \text{系统时钟})$ , 则通道(n+1)输出始终为无效值 (POL(n+1)位值)。

尽管大多数情况下死区延迟无法与通道(n)和(n+1)占空比相比较, 但下面这些图还是以示例说明了死区延迟可与占空比相比较的情况。



**图 26-66. 死区延迟可与通道(n+1)占空比相比较时的死区插入示例 (ELSnB:ELSnA = 1:0, POL(n) = 0 且 POL(n+1) = 0)**



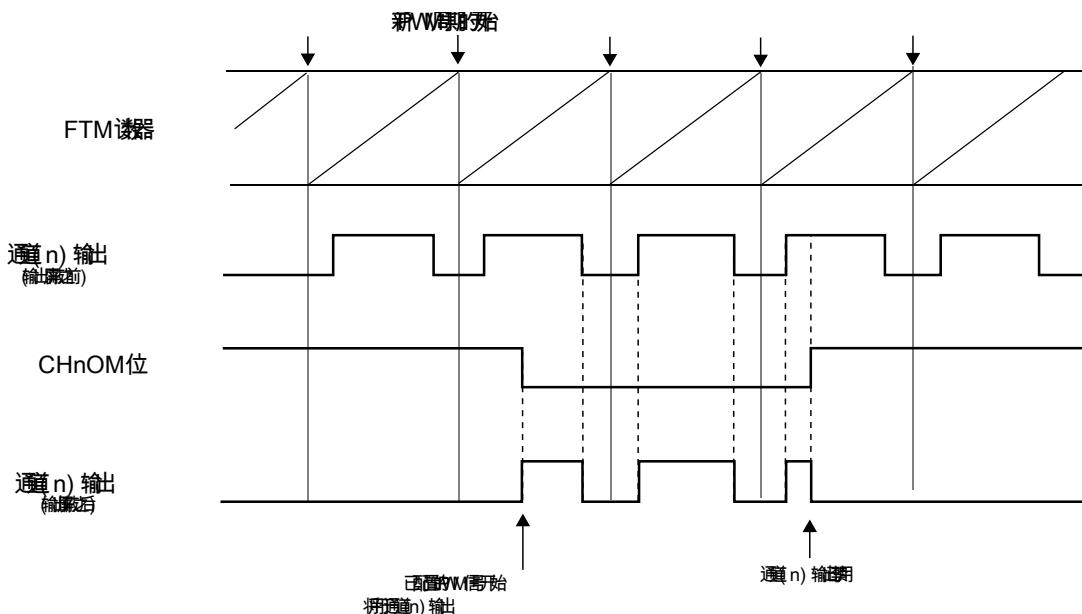
**图 26-67. 死区延迟可与通道(n)及(n+1)占空比相比较时的死区插入示例 (ELSnB:ELSnA = 1:0, POL(n) = 0 且 POL(n+1) = 0)**

## 26.4.15 输出屏蔽

输出屏蔽可用于通过软件强制通道输出为各自的无效状态。例如，用于控制 BLDC 电机。

对 OUTMASK 寄存器采取的任意写入操作都会更新其写入缓冲区。OUTMASK 寄存器通过 PWM 同步以其缓冲区值进行更新；参见 [OUTMASK 寄存器同步](#)。

如果 CHnOM = 1，则通道(n)输出强制为通道的无效状态 (POLn 位值)。如果 CHnOM = 0，则通道(n)输出不受输出屏蔽影响。参见下图。

图 26-68.  $\text{POL}_n = 0$  条件下的输出屏蔽

下表展示了极性控制之前的输出屏蔽结果。

表 26-10. 极性控制之前的通道(n)输出屏蔽结果

CHnOM	输出屏蔽输入	输出屏蔽结果
0	无效状态	无效状态
	有效状态	有效状态
1	无效状态	无效状态
	有效状态	

## 26.4.16 故障控制

如果( $\text{FAULT}_{\text{TM}}[1:0] \neq 0:0$ )，则使能故障控制。

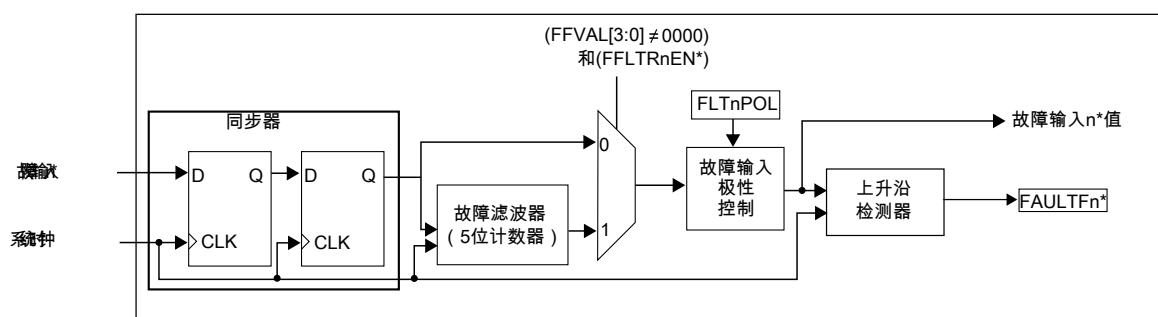
FTM 最多可拥有四个故障输入。 $\text{FAULT}_{\text{nEN}}$  位 (其中  $n = 0, 1, 2, 3$ ) 使能故障输入  $n$ ， $\text{FFLTR}_{\text{nEN}}$  位使能故障输入  $n$  滤波器。 $\text{FFVAL}[3:0]$  位选择已使能的每个故障输入中处于使能状态的滤波器的值。

首先，每个故障输入信号由系统时钟同步；参见下图中的同步器时钟。同步之后，故障输入  $n$  信号进入滤波器模块。当故障输入  $n$  信号中发生状态更改时，5 位计数器将复位并开始向上计数。只要新状态在故障输入  $n$  中保持稳定，计数器就会继续增加数值。如果 5 位计数器溢出，即计数器超出  $\text{FFVAL}[3:0]$  位的值，就表明新故障输入  $n$  值得以验证。然后就会作为脉冲边沿发送到边沿检测器。

如果在验证（计数器溢出）之前相反的边沿出现在故障输入 n 信号上，计数器就会复位。下一次输入转换时，计数器再次开始计数。任何比 FFVAL[3:0]位 (x 系统时钟) 所选的最小值短的脉冲都会被认作是毛刺，不会传递到边沿检测器上。

FFVAL[3:0]位为零或 FAULTnEN = 0 时，将禁用故障输入 n 滤波器。这种情况下，故障输入 n 信号将按系统时钟的 2 个上升沿延迟，而且在故障输入 n 中出现上升沿之后 FAULTFn 位在系统时钟的第 3 个上升沿上置位。

如果 FFVAL[3:0] ≠ 0000 且 FAULTnEN = 1，则故障输入 n 信号按系统时钟的(3 + FFVAL[3:0])个上升沿延迟，也就是在故障输入 n 中出现上升沿之后 FAULTFn 位在系统时钟的(4 + FFVAL[3:0])个上升沿上置位。



\* 其中 = 3, 2, 1, 0

图 26-69. 故障输入 n 控制功能框图

如果故障控制和故障输入 n 已使能，且在故障输入 n 信号上检测到上升沿，则表明已出现故障状况且 FAULTFn 位已置位。FAULTF 位是 FAULTFn[3:0]位的逻辑或 (OR)。参见下图。

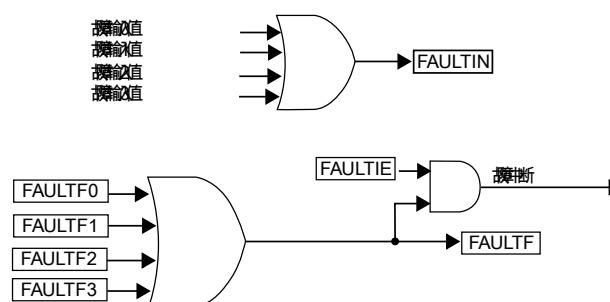


图 26-70. FAULTF 和 FAULTIN 位以及故障中断

如果已使能故障控制(FAULTM[1:0] ≠ 0:0)、已出现故障状况且(FAULTEN = 1)，则输出强制为各自的安全值：

- 通道(n)输出采用 POL(n)的值
- 通道(n+1)采用 POL(n+1)的值

当( $\text{FAULTF} = 1$ )且( $\text{FAULTIE} = 1$ )时，生成故障中断。此中断请求一直保持置位状态，直到出现以下情形：

- 软件通过读取 FAULTF 位为 1 并向其写入 0 来清除 FAULTF 位
- 软件清除 FAULTIE 位
- 发生复位

#### 26.4.16.1 自动故障清除

如果选择了自动故障清除( $\text{FAULTM}[1:0] = 1:1$ )，则当故障输入信号( $\text{FAULTIN}$ )恢复为零，新的 PWM 周期开始时，被故障控制禁用的通道输出将再次进入使能状态。参见下图。

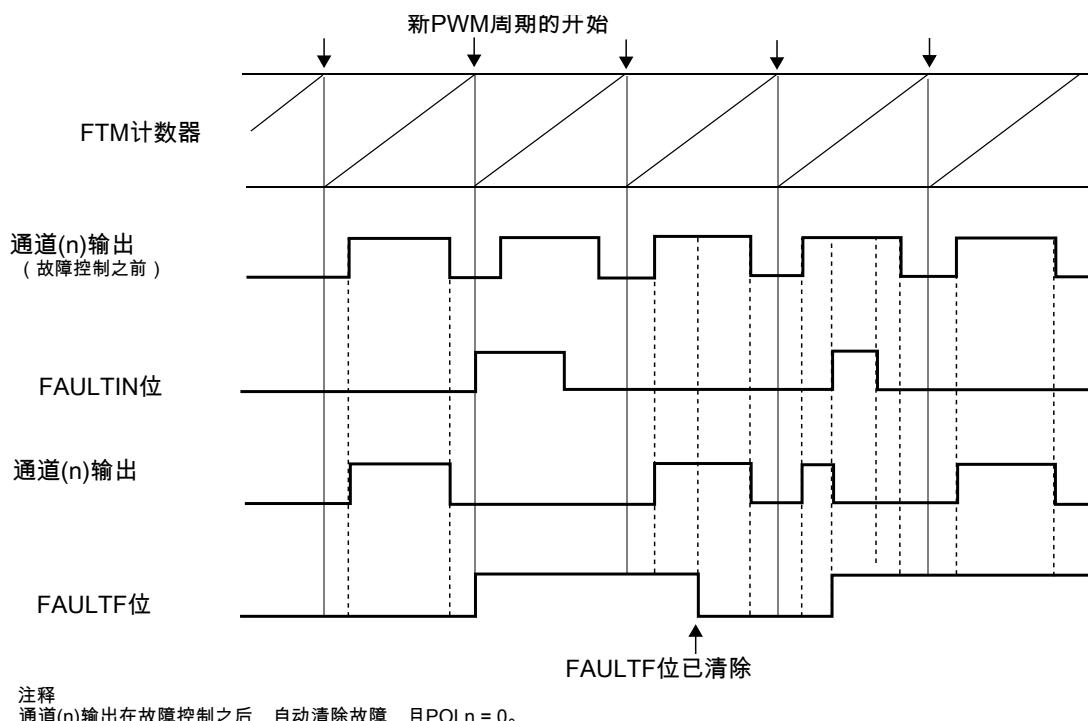


图 26-71. 采用自动故障清除的故障控制

#### 26.4.16.2 手动故障清除

如果选择了手动故障清除 ( $\text{FAULTM}[1:0] = 0:1$  或  $1:0$ )，则当 FAULTF 位清除，新的 PWM 周期开始时，被故障控制禁用的通道输出将再次进入使能状态。参见下图。

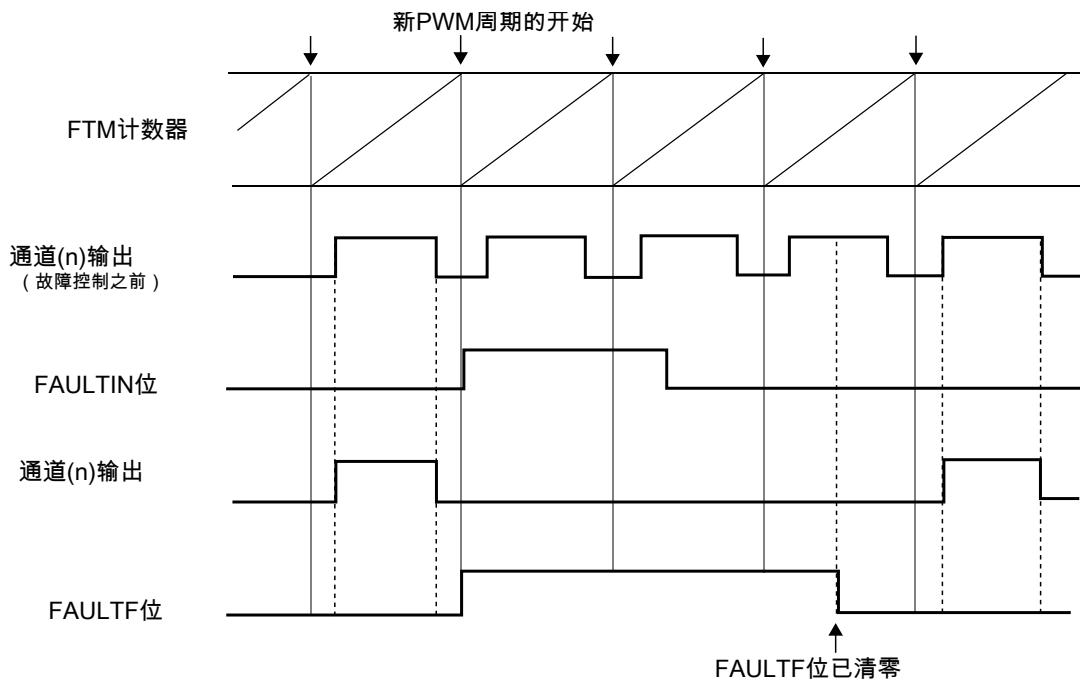


图 26-72. 采用手动故障清除的故障控制

### 26.4.16.3 故障输入极性控制

FLTjPOL 位选择故障输入 j 极性，其中  $j = 0, 1, 2, 3$ :

- 如果  $\text{FLTjPOL} = 0$ ，则故障 j 输入极性高，因此故障输入 j 处的逻辑一表示故障。
- 如果  $\text{FLTjPOL} = 1$ ，则故障 j 输入极性低，因此故障输入 j 处的逻辑零表示故障。

### 26.4.17 极性控制

POLn 位选择通道(n)输出极性:

- 如果  $\text{POLn} = 0$ ，则通道(n)输出极性为高，因此逻辑一为有效状态，逻辑零为无效状态。
- 如果  $\text{POLn} = 1$ ，则通道(n)输出极性为低，因此逻辑零为有效状态，逻辑一为无效状态。

### 26.4.18 初始化

向 INIT 位写入一时，初始化将强制通道(n)输出为 CHnOI 位的值。

初始化取决于 COMP 和 DTEN 位。下表展示了 COMP 和 DTEN 位为零时初始化条件下通道(n)和(n+1)输出的值。

表 26-11. (COMP = 0 且 DTEN = 0) 时的初始化结果

CH(n)OI	CH(n+1)OI	通道(n)输出	通道(n+1)输出
0	0	强制为零	强制为零
0	1	强制为零	强制为一
1	0	强制为一	强制为零
1	1	强制为一	强制为一

下表展示了(COMP = 1)或(DTEN = 1)时由初始化强制通道(n)和(n+1)输出的值。

表 26-12. (COMP = 1 或 DTEN = 1) 时的初始化行为方式

CH(n)OI	CH(n+1)OI	通道(n)输出	通道(n+1)输出
0	X	强制为零	强制为一
1	X	强制为一	强制为零

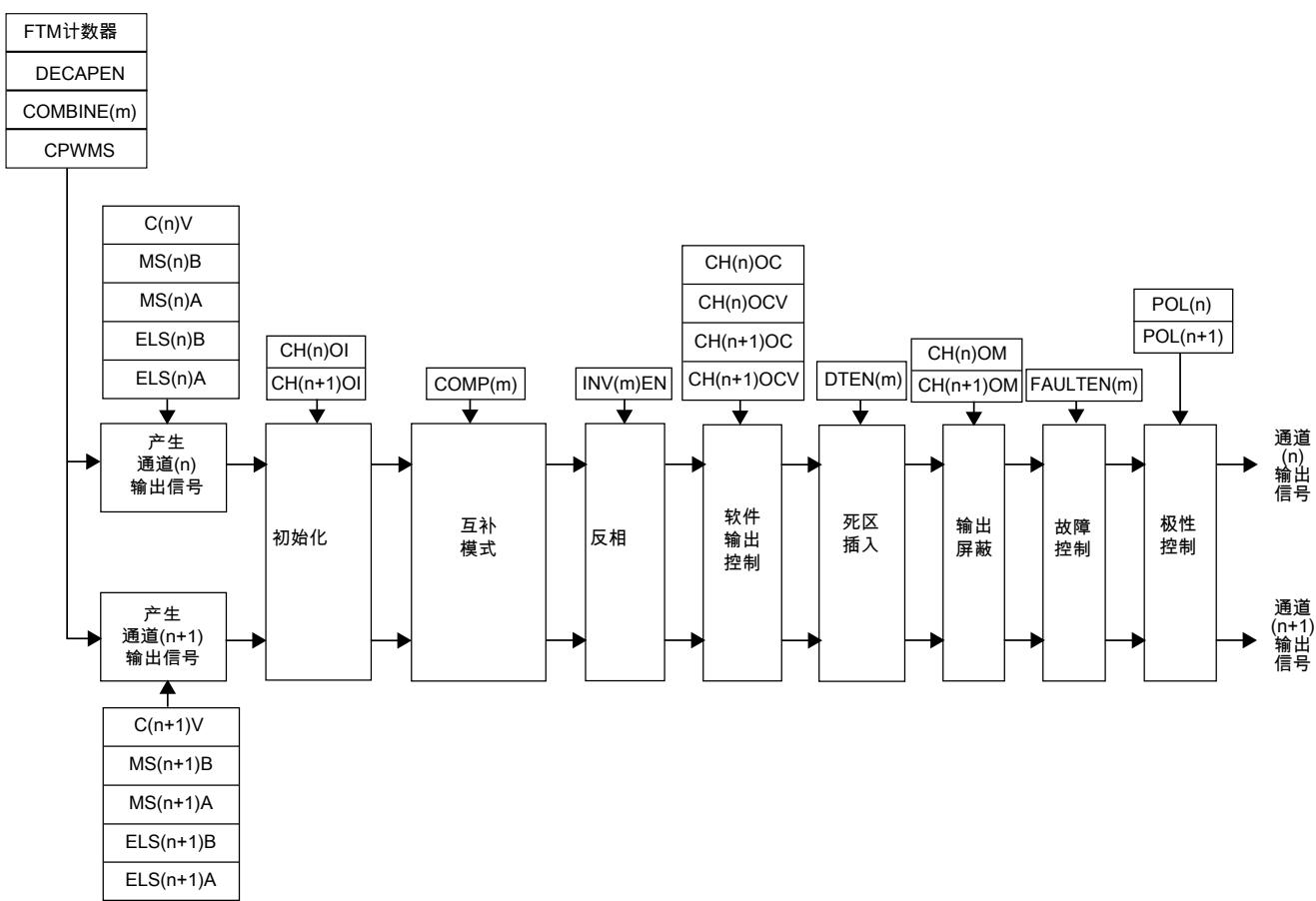
### 注

初始化特性只能与禁用的 FTM 计数器配合使用。参见状态和控制寄存器中关于 CLKS 字段的说明。

### 26.4.19 特性优先级

下图展示了生成通道(n)和(n+1)输出信号时所用特性的优先级。

通道对(m) - 通道(n)和(n+1)

**注意**

通道(n) 和 n+1 优先级比较 EPM/CPW 或互换。

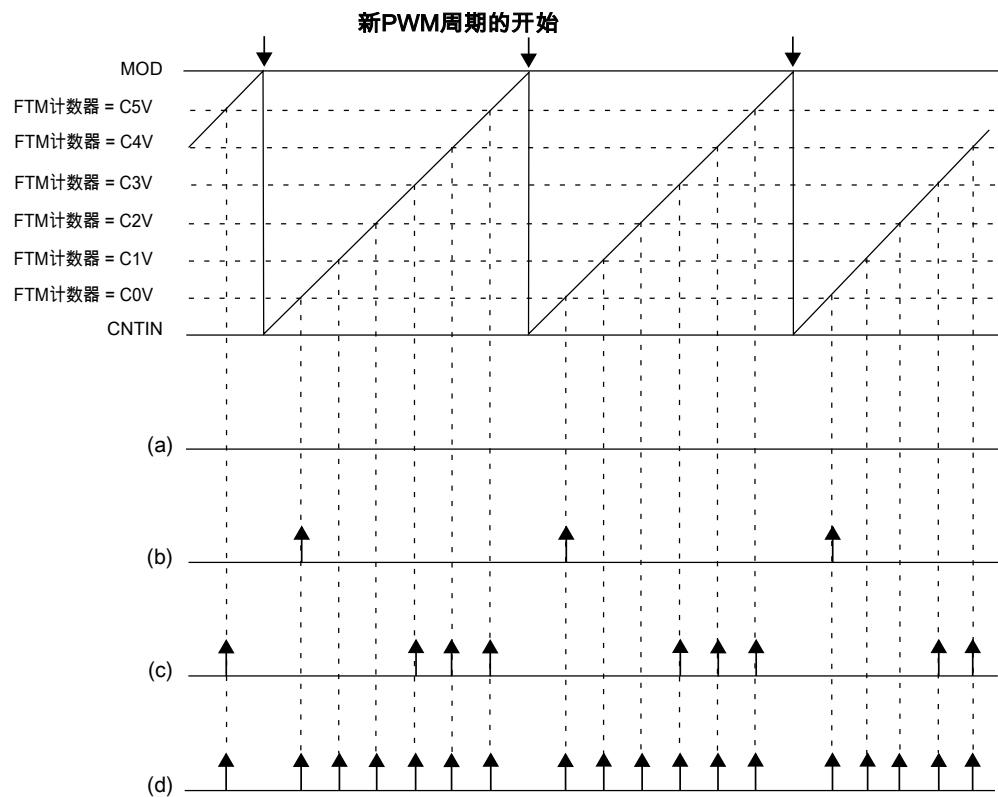
**图 26-73. 生成通道(n)和(n+1)输出信号时所用特性的优先级****注**初始化 特性不得与**反相** 和**软件输出控制** 特性一起使用。

## 26.4.20 通道触发器输出

如果  $CHjTRIG = 1$  (其中  $j = 0, 1, 2, 3, 4$  或  $5$ )，则当通道( $j$ )发生匹配事件 ( $FTM$  计数器= $C(j)V$ )， $FTM$  会产生触发信号。

通道触发器输出提供了一种可用于片上模块的触发信号。

$FTM$  能够在一个 PWM 周期内生成多个触发器。因为每个触发信号都是由一个特定通道生成的，所以要实现此功能需要有多个通道。下图描述了这种行为方式。

**注释**

- (a) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
- (b) CH0TRIG = 1, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
- (c) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1
- (d) CH0TRIG = 1, CH1TRIG = 1, CH2TRIG = 1, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1

**图 26-74. 通道匹配触发器**

### 26.4.21 初始化触发

如果 INITTRIGEN = 1，那么在以下情形中 FTM 计数器更新为 CNTIN 寄存器值时，FTM 将生成触发。

- FTM 计数器通过所选计数模式自动更新为 CNTIN 寄存器值。
- 对 CNT 寄存器采取写入操作时。
- 存在 **FTM 计数器同步** 时。
- 如果(CNT = CNTIN)、(CLKS[1:0] = 0:0)，并且向 CLKS[1:0]位写入了不为零的值。

以下各图展示了这些情形。

CNTIN = 0x0000  
MOD = 0x000F  
CPWMS = 0

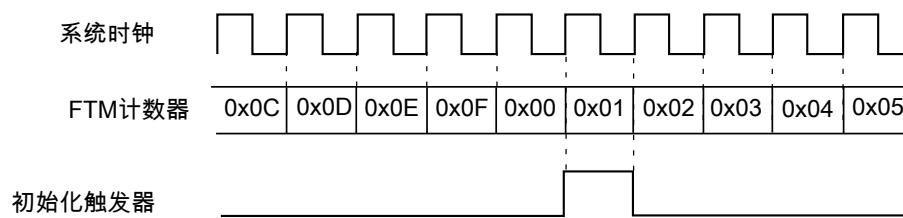


图 26-75. FTM 计数达到 CNTIN 寄存器值时生成初始化触发

CNTIN = 0x0000  
MOD = 0x000F  
CPWMS = 0

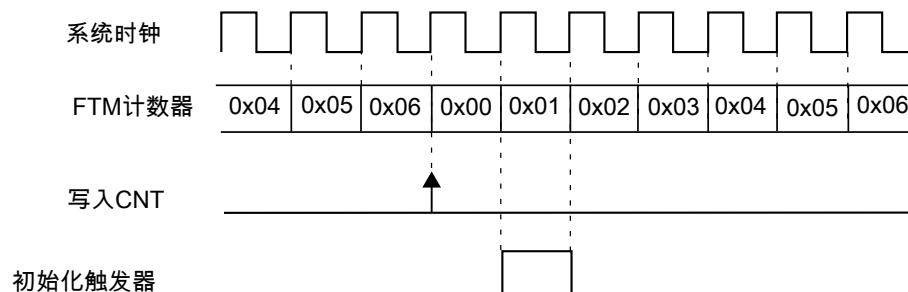


图 26-76. 对 CNT 寄存器采取写入操作时生成初始化触发

CNTIN = 0x0000  
MOD = 0x000F  
CPWMS = 0

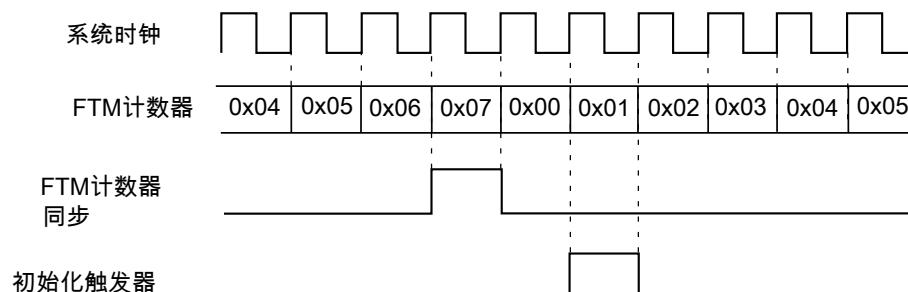
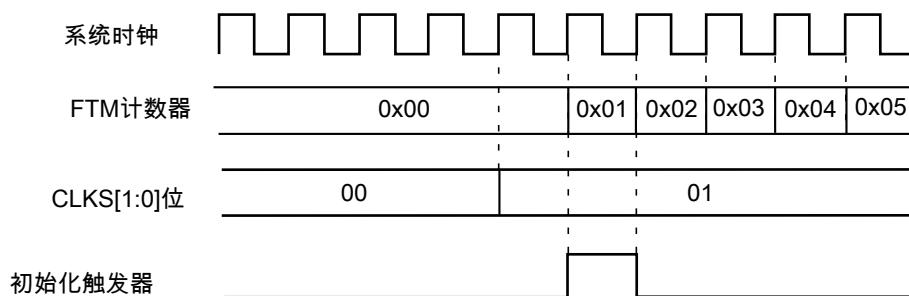


图 26-77. 存在 FTM 计数器同步时生成初始化触发

CNTIN = 0x0000  
MOD = 0x000F  
CPWMS = 0



**图 26-78. (CNT = CNTIN)、(CLKS[1:0] = 0:0)且向 CLKS[1:0]位写入了不为零的值时生成初始化触发**

初始化触发输出提供了一种可用于片上模块的触发信号。

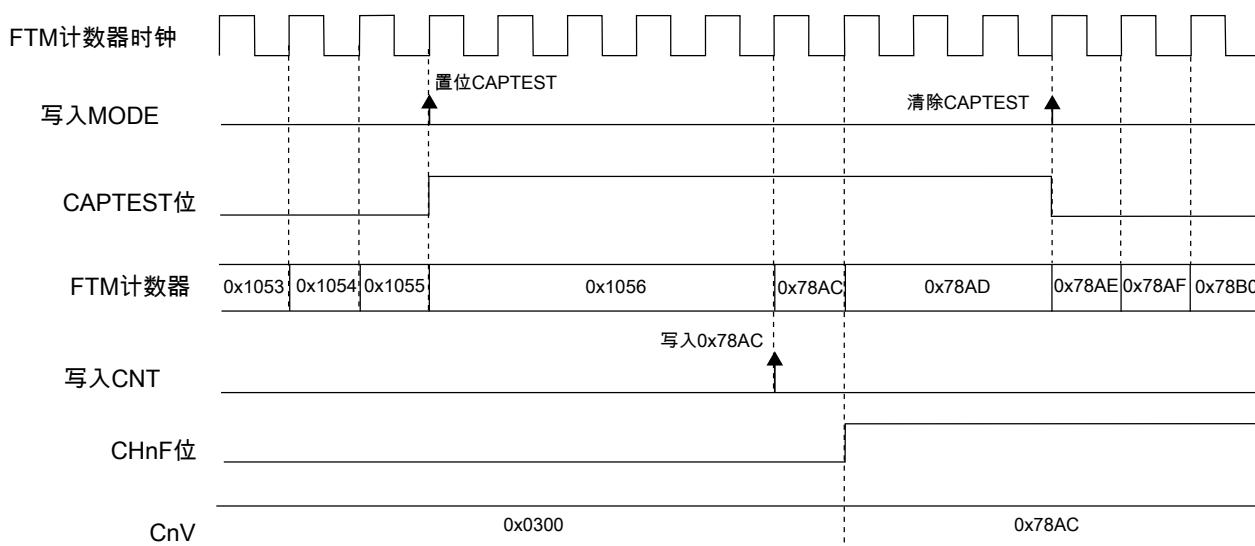
## 26.4.22 捕获测试模式

捕获测试模式允许测试 CnV 寄存器、FTM 计数器以及 FTM 计数器与 CnV 寄存器之间的互连逻辑。

在这种测试模式下，必须为[输入捕捉模式](#) 配置所有通道，且 FTM 计数器必须配置为[向上计数](#)。

使能捕获测试模式(CAPTEST = 1)后，FTM 计数器将冻结，对 CNT 寄存器进行的任何写入操作都会直接更新 FTM 计数器；参见下图。写入后，所有 CnV 寄存器都会更新为 CNT 寄存器中的值，同时 CHnF 位会置位。因此，FTM 计数器会根据其配置更新其下一个值。其下一个值取决于 CNTIN、MOD 以及写入 FTM 计数器的值。

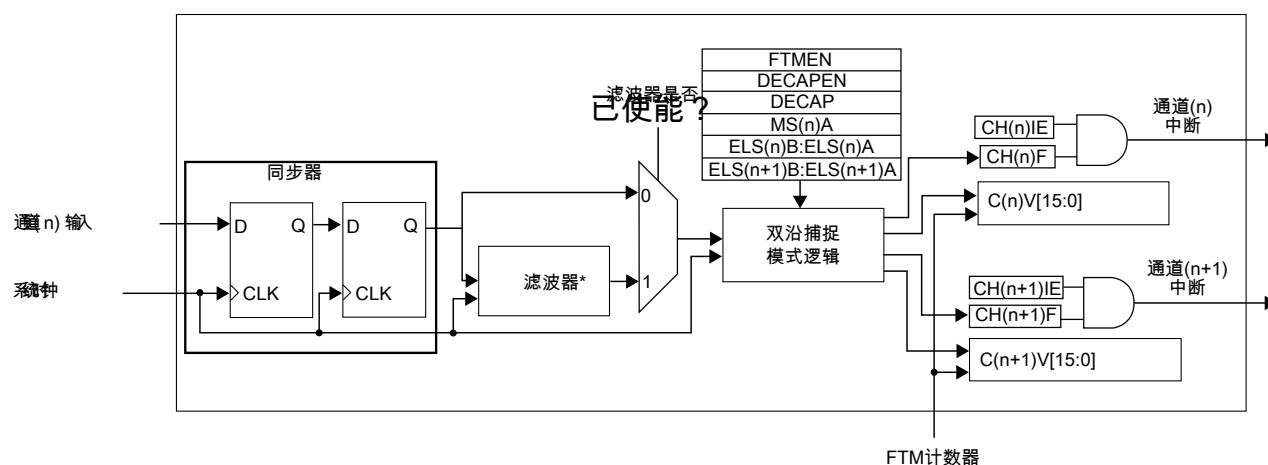
接下来的 CnV 寄存器读取操作会返回写入 FTM 计数器的值，接下来的 CNT 寄存器读取操作则返回 FTM 计数器的下一个值。

**注意**

- FTM计数器配置 : (FTMEN = 1) , (CAPTEST = 1) , (CPWMS = 0) , (CNTIN = 0x0000) , (MOD = 0xFFFF)
- FTM通道n配置 : 输入捕捉模式 - (DECAPEN = 0) , (COMBINE = 0) , (MSnB:MSnA = 0:0)

**图 26-79. 捕捉测试模式****26.4.23 双沿捕捉模式**

如果 DECAPEN = 1，将选择双沿捕捉模式。此模式允许在一对通道中通道(n)的输入端测量脉宽或信号周期。在此模式下，当 n 为 0 或 2 时，通道(n)滤波器可处于有效状态。

**图 26-80. 双沿捕捉模式结构框图**

MS(n)A 位定义双沿捕捉模式是单次还是持续。

ELS(n)B:ELS(n)A 位选择由通道(n)捕捉的边沿, ELS(n+1)B:ELS(n+1)A 位选择由通道(n+1)捕捉的边沿。如果 ELS(n)B:ELS(n)A 和 ELS(n+1)B:ELS(n+1)A 位选择同一边沿, 则为周期测量。如果这两个位选择不同边沿, 则为脉宽测量。

在双沿捕捉模式下, 只使用通道(n)输入, 忽略通道(n+1)输入。

如果在通道(n)输入端检测到由通道(n)位定义的边沿, 则 CH(n)F 位置位并生成通道(n)中断(条件是 CH(n)IE = 1)。如果在通道(n)输入端检测到由通道(n+1)位定义的边沿且(CH(n)F = 1), 则 CH(n+1)F 位置位并生成通道(n+1)中断(条件是 CH(n+1)IE = 1)。

C(n)V 寄存器存储了在通道(n)输入端检测到通道(n)所选边沿时 FTM 计数器的值。C(n+1)V 寄存器存储了在通道(n)输入端检测到通道(n+1)所选边沿时 FTM 计数器的值。

在此模式下, 有一个一致性机制可确保读取 C(n)V 和 C(n+1)V 寄存器时保持一致的数据。这个唯一的要求就是必须在 C(n+1)V 之前读取 C(n)V。

### 注

- CH(n)F、CH(n)IE、MS(n)A、ELS(n)B 和 ELS(n)A 位是与通道(n)有关的位。
- CH(n+1)F、CH(n+1)IE、MS(n+1)A、ELS(n+1)B 和 ELS(n+1)A 位是通道(n+1)位。
- 必须在 ELS(n)B:ELS(n)A = 0:1 或 1:0、ELS(n+1)B:ELS(n+1)A = 0:1 或 1:0 且 FTM 计数器为自由运行计数器的情况下使用双沿捕捉模式。

#### 26.4.23.1 一次性捕捉模式

当(DECAPEN = 1)且(MS(n)A = 0)时选择单次捕捉模式。在这种捕捉模式下, 只捕捉通道(n)输入上的一对边沿。ELS(n)B:ELS(n)A 位选择要捕捉的第一边沿, ELS(n+1)B:ELS(n+1)A 位选择要捕捉的第二边沿。

DECAP 位置位时将使能边沿捕捉。对于单次捕捉模式下的每一次新测量, 必须首先清除 CH(n)F 和 CH(n+1)位, 然后 DECAP 位必须置位。

在此模式下, 将在捕捉通道(n+1)所选的边沿后由 FTM 自动清除 DECAP 位。因此, DECAP 位置位后, 就会开始单次捕捉。清除此位后, 两个边沿都会捕捉, 捕捉的值也可用于在 C(n)V 和 C(n+1)V 寄存器中读取。

与此类似, CH(n+1)F 位置位后, 两个边沿都会捕捉, 捕捉的值也可用于在 C(n)V 和 C(n+1)V 寄存器中读取。

### 26.4.23.2 连续捕捉模式

当(DECAPEN = 1)且(MS(n)A = 1)时选择持续捕捉模式。在这种捕捉模式下，将持续捕捉通道(n)输入上的边沿。ELS(n)B:ELS(n)A 位选择要捕捉的初始沿，ELS(n+1)B:ELS(n+1)A 位选择要捕捉的最终沿。

DECAP 位置位时将使能边沿捕捉。初次使用时，首先必须清空 CH(n)F 和 CH(n+1)F 位，然后必须使 DECAP 位置位，以便开始持续测量。

CH(n+1)F 位置位后，两个边沿都会被捕捉，捕捉的值也可在 C(n)V 和 C(n+1)V 寄存器中读取。即便已清空 DECAP 位，最新捕捉到的值也仍然可用于这些寄存器。

在此模式下，可以只清空 CH(n+1)F 位。因此，CH(n+1)F 位再次置位后，最新捕捉到的值将可用在 C(n)V 和 C(n+1)V 寄存器中。

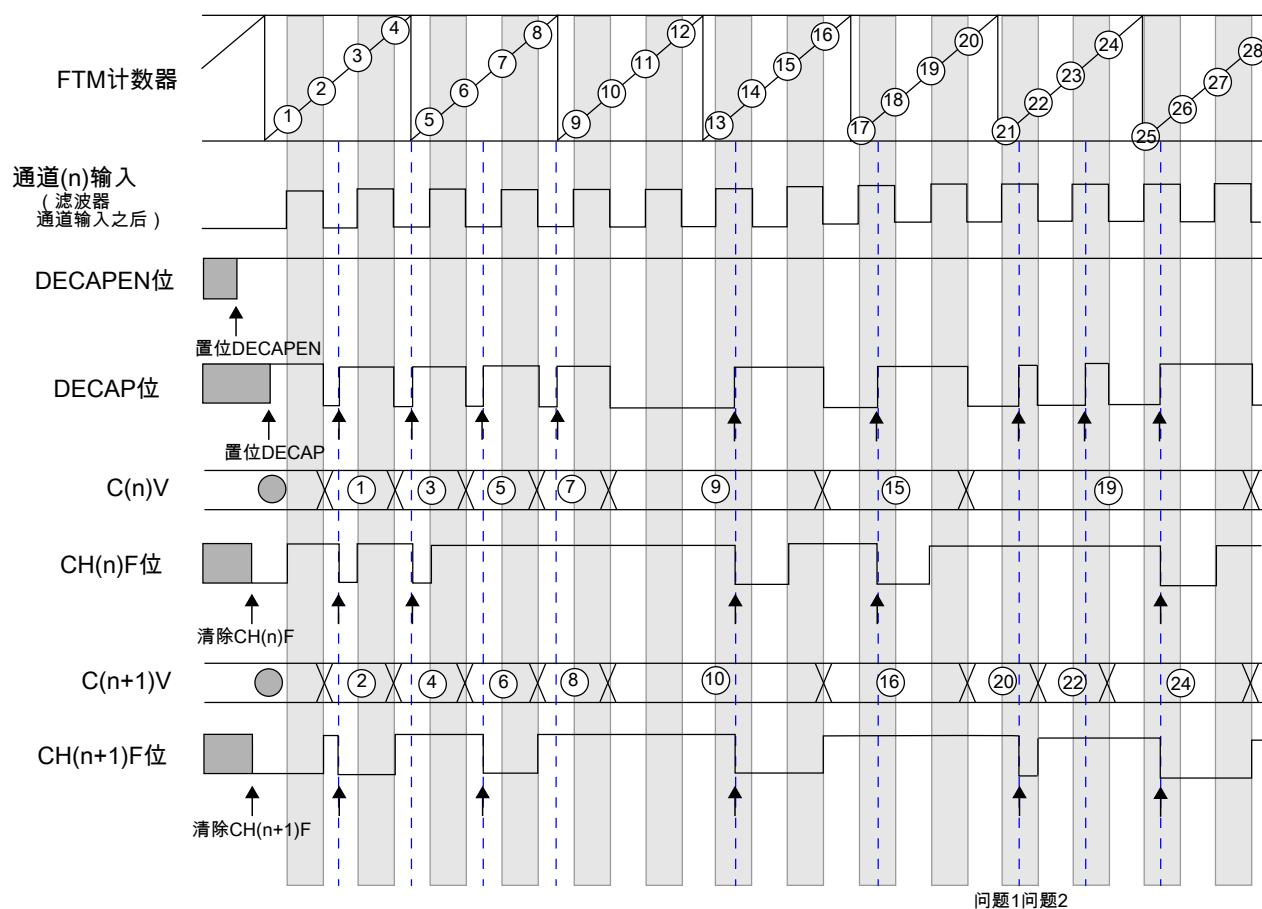
对于双沿捕捉 – 持续模式下的一系列新测量，清空 CH(n)F 和 CH(n+1)F 位便可开始新的测量。

### 26.4.23.3 脉宽测量

如果通道(n)配置为捕捉上升沿(ELS(n)B:ELS(n)A = 0:1)，通道(n+1)配置为捕捉下降沿(ELS(n+1)B:ELS(n+1)A = 1:0)，则测量正极性脉宽。如果通道(n)配置为捕捉下降沿(ELS(n)B:ELS(n)A = 1:0)，通道(n+1)配置为捕捉上升沿(ELS(n+1)B:ELS(n+1)A = 0:1)，则测量负极性脉宽。

可在[一次性捕捉模式](#) 或[连续捕捉模式](#) 下进行脉宽测量。

下图给出了双沿捕捉 – 单次模式的示例，用于测量正极性脉宽。DECAPEN 位选择双沿捕捉模式，因此保持置位。DECAP 位置位，以使能下一个正极性脉宽的测量。检测到此脉冲的第一个边沿（也就是由 ELS(n)B:ELS(n)A 位选择的边沿）时，将置位 CH(n)F 位。检测到此脉冲的第二个边沿（也就是由 ELS(n+1)B:ELS(n+1)A 位选择的边沿）时，将置位 CH(n+1)F 位并清除 DECAP 位。DECAP 和 CH(n+1)F 位指出何时捕捉到脉冲的两个边沿，以及 C(n)V 和 C(n+1)V 寄存器何时准备好进行读取。

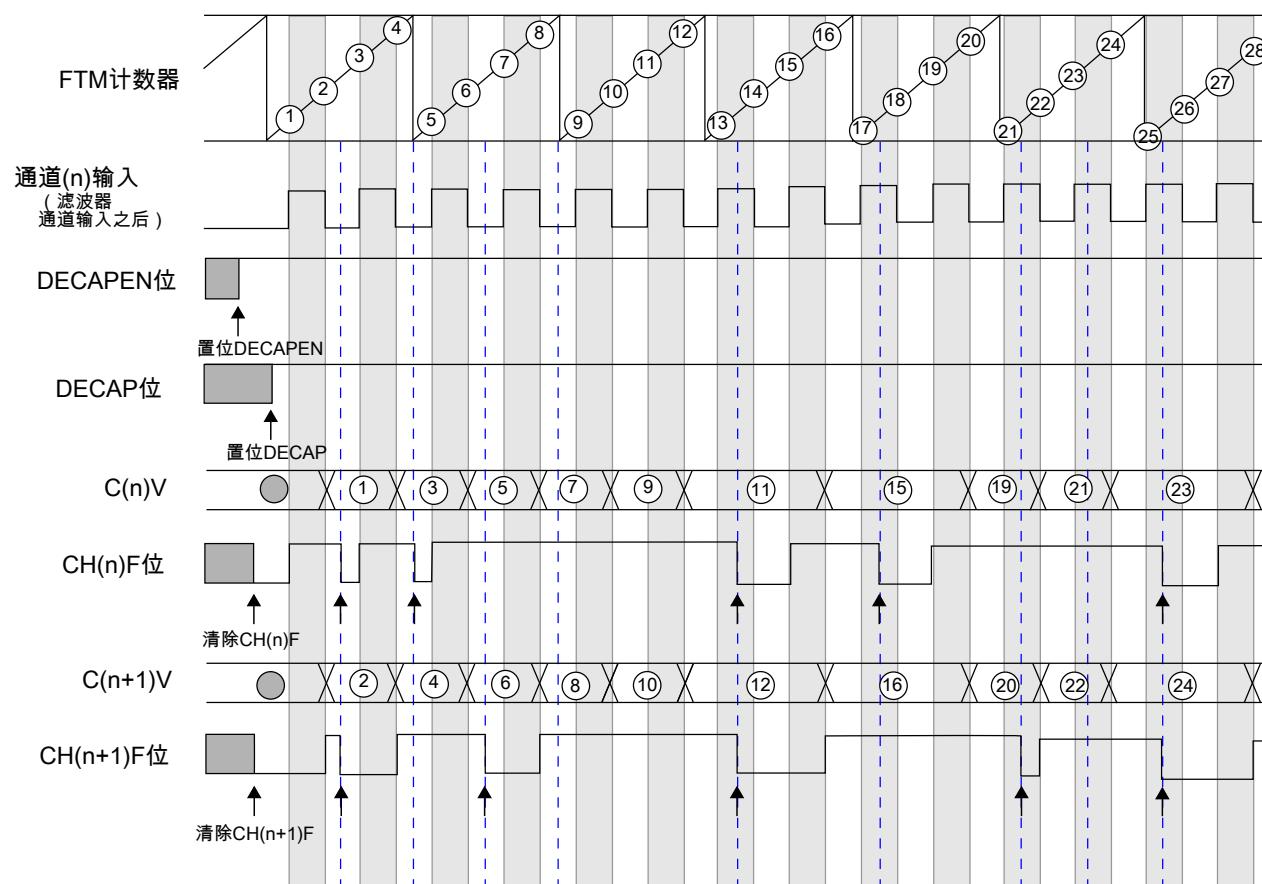


注释- 置位DECAPEN、置位DECAP、清除CH(n)F和清除CH(n+1)这些命令均由用户发出。

- 问题1：通道(n)输入 = 1，置位DECAP，不清除CH(n)F，清除CH(n+1)F。
- 问题2：通道(n)输入 = 1，置位DECAP，不清除CH(n)F，不清除CH(n+1)F。

图 26-81. 用于正极性脉宽测量的双沿捕捉 – 单次模式

下图给出了双沿捕捉 – 持续模式的示例，用于测量正极性脉宽。DECAPEN 位选择双沿捕捉模式，因此保持置位。DECAP 位置位时，将进行已配置的测量。检测到正极性脉冲的第一个边沿（也就是由 ELS(n)B:ELS(n)A 位选择的边沿）时，将置位 CH(n)F 位。检测到此脉冲的第二个边沿（也就是由 ELS(n+1)B:ELS(n+1)A 位选择的边沿）时，将置位 CH(n+1)F 位。CH(n+1)F 位指出何时捕捉到脉冲的两个边沿，以及 C(n)V 和 C(n+1)V 寄存器何时准备好进行读取。



注释- 置位DECAPEN、置位DECAP、清除CH(n)F和清除CH(n+1)这些命令均由用户发出。

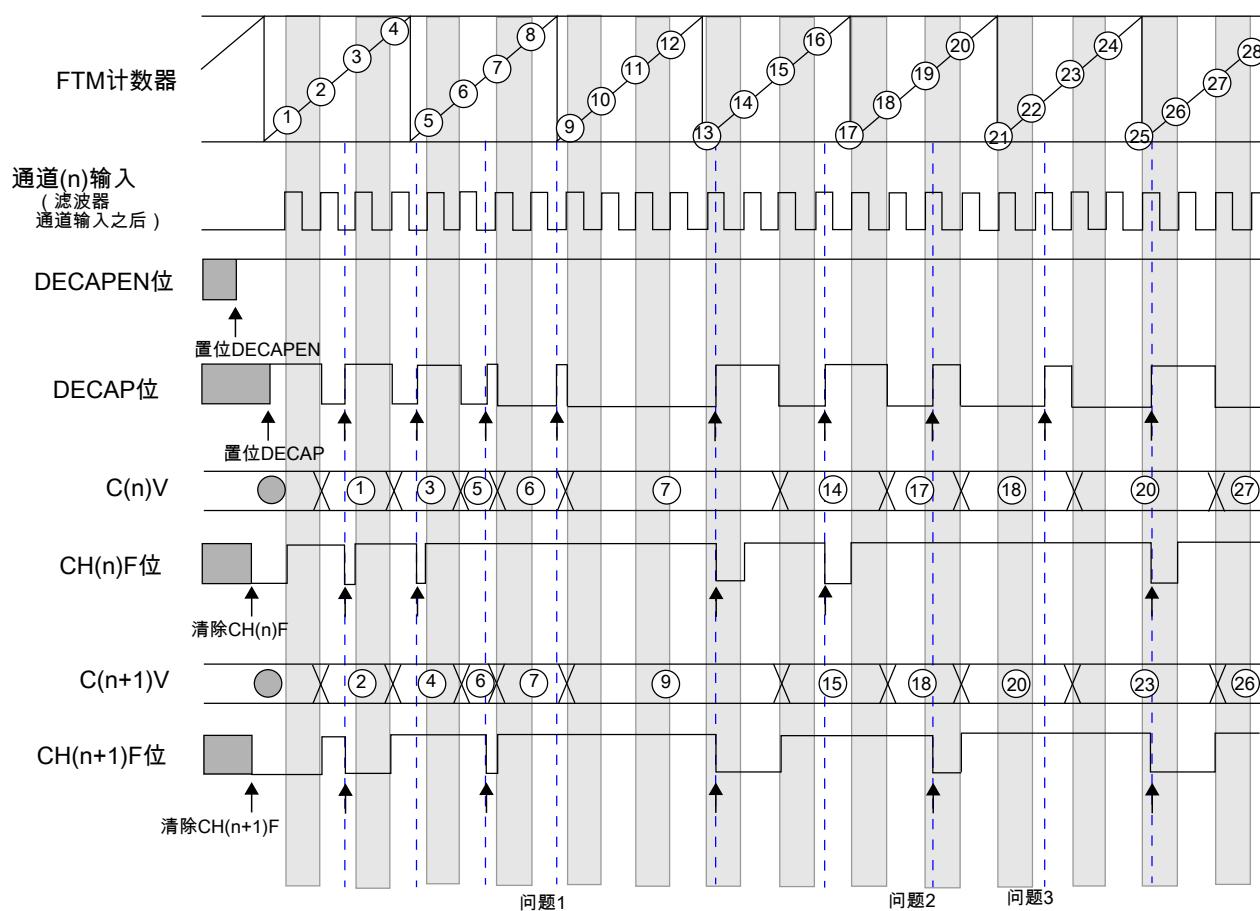
图 26-82. 用于正极性脉宽测量的双沿捕捉 – 持续模式

#### 26.4.23.4 周期测量

如果通道(n)和(n+1)配置为捕捉相同极性的连续边沿，则测量通道(n)输入信号的周期。如果通道(n)和(n+1)配置为捕捉上升沿 (ELS(n)B:ELS(n)A = 0:1 且 ELS(n+1)B:ELS(n+1)A = 0:1)，则测量两个连续上升沿之间的周期。如果通道(n)和(n+1)配置为捕捉下降沿 (ELS(n)B:ELS(n)A = 1:0 且 ELS(n+1)B:ELS(n+1)A = 1:0)，则测量两个连续下降沿之间的周期。

可在[一次性捕捉模式](#)或[连续捕捉模式](#)下进行周期测量。

下图给出了双沿捕捉 – 单次模式的示例，用于测量两个连续上升沿之间的周期。DECAPEN 位选择双沿捕捉模式，因此保持置位。DECAP 位置位，以使能下一周期的测量。检测到第一个上升沿（也就是由 ELS(n)B:ELS(n)A 位选择的边沿）时，将置位 CH(n)F 位。检测到第二个上升沿（也就是由 ELS(n+1)B:ELS(n+1)A 位选择的边沿）时，将置位 CH(n+1)F 位并清空 DECAP 位。DECAP 和 CH(n+1)F 位指出何时捕捉到所选的两个边沿，以及 C(n)V 和 C(n+1)V 寄存器何时准备好进行读取。

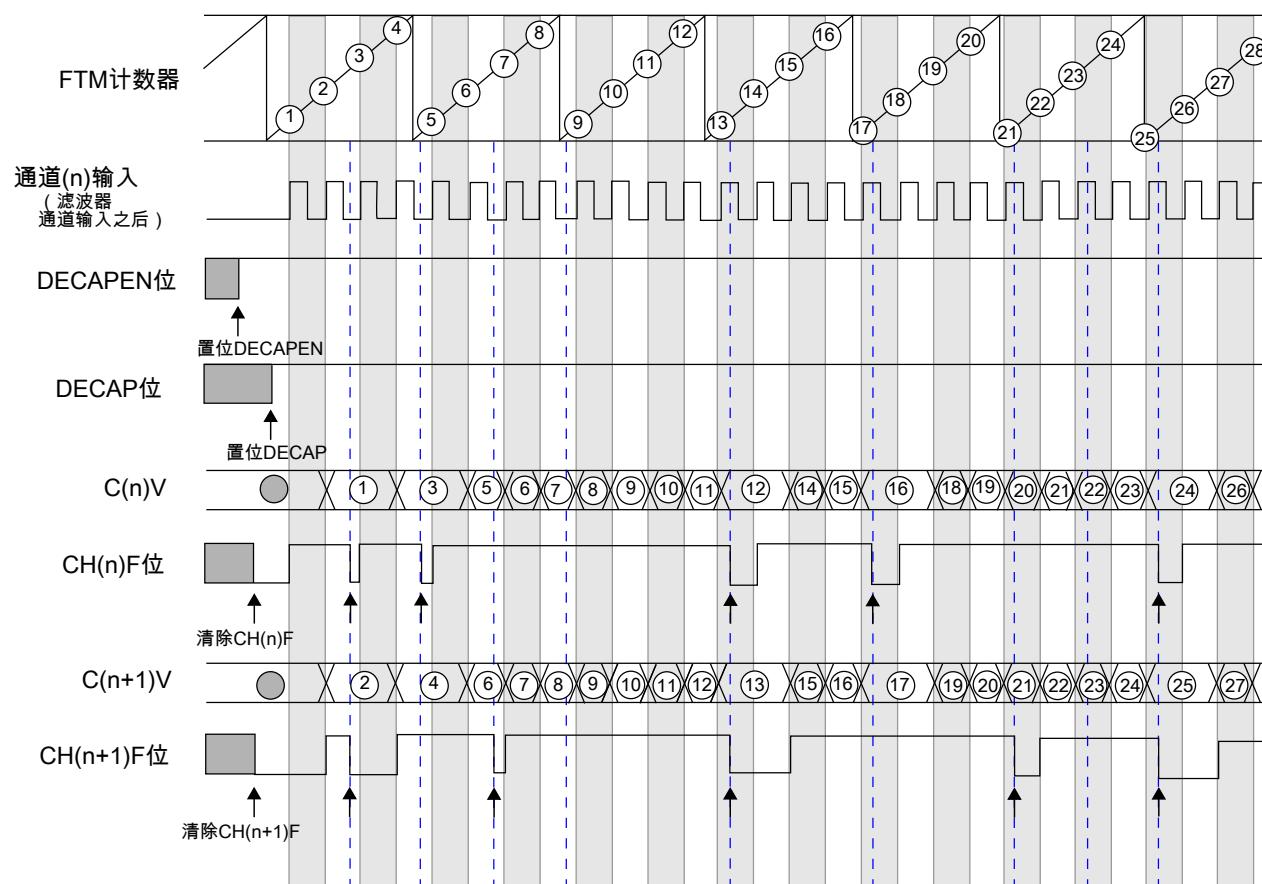


注释- 置位DECAPEN、置位DECAP、清除CH(n)F和清除CH(n+1)这些命令均由用户发出。

- 问题1：通道(n)输入 = 0，置位DECAP，不清除CH(n)F，不清除CH(n+1)F。
- 问题2：通道(n)输入 = 1，置位DECAP，不清除CH(n)F，清除CH(n+1)F。
- 问题3：通道(n)输入 = 1，置位DECAP，不清除CH(n)F，不清除CH(n+1)F。

**图 26-83. 用于测量两个连续上升沿之间周期的双沿捕捉 – 单次模式**

下图给出了双沿捕捉 – 持续模式的示例，用于测量两个连续上升沿之间的周期。DECAPEN 位选择双沿捕捉模式，因此保持置位。DECAP 位置位时，将进行已配置的测量。检测到第一个上升沿（也就是由 ELS(n)B:ELS(n)A 位选择的边沿）时，将置位 CH(n)F 位。检测到第二个上升沿（也就是由 ELS(n+1)B:ELS(n+1)A 位选择的边沿）时，将置位 CH(n+1)F 位。CH(n+1)F 位指出何时捕捉到周期的两个边沿，以及 C(n)V 和 C(n+1)V 寄存器何时准备好进行读取。



注释- 置位DECAPEN、置位DECAP、清除CH(n)F和清除CH(n+1)这些命令均由用户发出。

图 26-84. 用于测量两个连续上升沿之间周期的双沿捕捉 – 持续模式

### 26.4.23.5 读取一致性机制

对于在 C(n)V 和 C(n+1)V 寄存器中捕捉的 FTM 计数器值，双沿捕捉模式会实施一种读取一致性机制。下图对读取一致性机制进行了说明。此例中，通道(n)和(n+1)处于双沿捕捉 – 持续模式，以便进行正极脉宽测量。因此，通道(n)配置为在通道(n)输入信号中存在上升沿时捕捉 FTM 计数器值，通道(n+1)配置为在通道(n)输入信号中存在下降沿时捕捉 FTM 计数器值。

通道(n)输入信号中存在上升沿时，FTM 计数器值将捕捉到通道(n)捕捉缓存内。通道(n)输入信号中存在下降沿时，通道(n)捕捉缓存值将传输到 C(n)V 寄存器中。上一个上升沿发生时，C(n)V 寄存器拥有 FTM 计数器值；最后一个上升沿发生时，通道(n)捕捉缓存拥有 FTM 计数器值。

通道(n)输入信号中存在下降沿时，FTM 计数器值将捕捉到通道(n+1)捕捉缓存内。读取 C(n)V 寄存器后，通道(n+1)捕捉缓存值将传输到 C(n+1)V 寄存器中。

下图中，发生事件 1 时读取 C(n)V 将返回 FTM 计数器值，发生事件 2 时读取 C(n+1)V 将返回 FTM 计数器值。

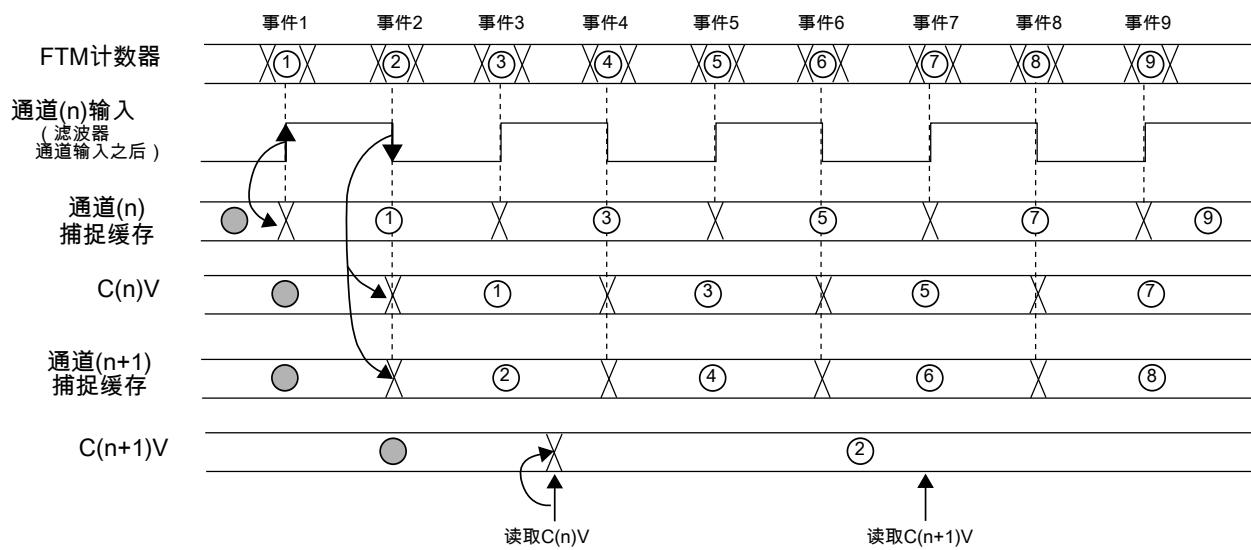


图 26-85. 双沿捕捉模式读取一致性机制

要让读取一致性机制正常工作，在单次双沿捕捉和持续双沿捕捉模式下，必须先读取 C(n)V 寄存器，然后再读取 C(n+1)V 寄存器。

## 26.4.24 Debug 模式

芯片处于 Debug 模式时，BDMMODE[1:0]位将根据下表选择 FTM 计数器、CH(n)F 位、通道输出以及对 MOD、CNTIN 和 C(n)V 寄存器的写入操作的行为方式。

表 26-13. 芯片处于 Debug 模式时的 FTM 行为方  
式

BDMMODE	FTM 计数器	CH(n)F 位	FTM 通道输出	对 MOD、CNTIN 和 C(n)V 寄存器的写入操作
00	已停止	可以设置	正常工作模式	对这些寄存器的写入操作会绕过寄存器缓冲区
01	已停止	未设置	通道输出会根据 POLn 位强制采取各自的安全值	对这些寄存器的写入操作会绕过寄存器缓冲区
10	已停止	未设置	芯片进入 Debug 模式时，通道输出会冻结	对这些寄存器的写入操作会绕过寄存器缓冲区
11	正常工作模 式	可以设置	正常工作模式	功能模式

注意，如果 BDMMODE[1:0] = 2'b00，通道输出将保持芯片进入 Debug 模式时的值，因为 FTM 计数器已停止。但是，以下情形会在这种 Debug 模式下修改通道输出。

- 向 CNT 寄存器写入任意值；参见[计数器复位](#)。这种情况下，FTM 计数器会根据 CNTIN 寄存器值更新，并且通道输出更新为初始值 – 设置为输出比较模式的通道则例外。
- FTM 计数器由 PWM 同步模式复位；参见[FTM 计数器同步](#)。这种情况下，FTM 计数器会根据 CNTIN 寄存器值更新，并且通道输出更新为初始值 – 处于输出比较模式的通道则例外。
- 在通道输出初始化过程中，向 INIT 位写入值 1 时，通道(n)输出将强制为 CH(n)OI 位值。参见[初始化](#)。

### 注

$\text{BDMMODE}[1:0] = 2'b00$  不得与[故障控制](#)一起使用。即便已使能故障控制且存在故障条件，通道输出值也仍然会按上文所述更新。

### 注

如果 BDM 中的  $\text{CLKS}[1:0] = 2'b00$ ，则会在 BDM 中的 CLKS 中写入一个非零值且  $\text{CnV} = \text{CNTIN}$  (BDM 禁用)，随后 CHnF 位置位（如果通道为 0%EPWM 信号）(BDM 禁用)。

## 26.4.25 中间加载

PWMLOAD 寄存器允许在定义的重载点根据寄存器缓冲区的内容更新 MOD、CNTIN 和 C(n)V 寄存器。这种情况下，不需要使用 PWM 同步。

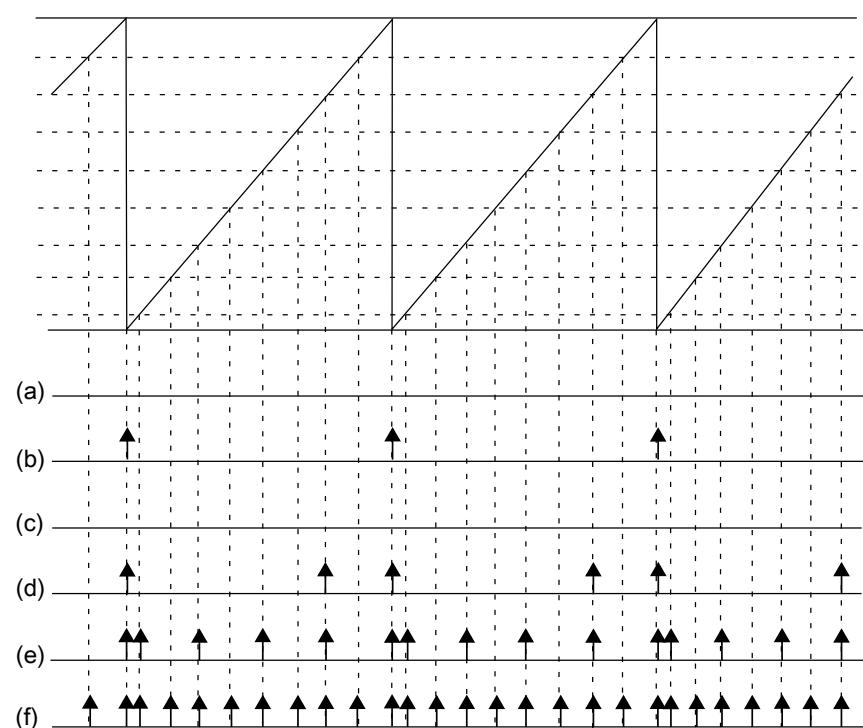
对于中间加载，有多种可能的加载点：

表 26-14. 什么时候使能可能的加载点

加载点	使能
FTM 计数器从 MOD 值变为 CNTIN 值时	始终
通道(j)匹配 (FTM 计数器 = C(j)V) 时	$\text{CHjSEL} = 1$ 时

下图给出了一些有关已使能加载点的示例。

FTM计数器 = MOD  
 FTM计数器 = C7V  
 FTM计数器 = C6V  
 FTM计数器 = C5V  
 FTM计数器 = C4V  
 FTM计数器 = C3V  
 FTM计数器 = C2V  
 FTM计数器 = C1V  
 FTM计数器 = C0V

**注释**

- (a) LDOK = 0 , CH0SEL = 0 , CH1SEL = 0 , CH2SEL = 0 , CH3SEL = 0 , CH4SEL = 0 , CH5SEL = 0 , CH6SEL = 0 , CH7
- (b) LDOK = 1 , CH0SEL = 0 , CH1SEL = 0 , CH2SEL = 0 , CH3SEL = 0 , CH4SEL = 0 , CH5SEL = 0 , CH6SEL = 0 , CH7
- (c) LDOK = 0 , CH0SEL = 0 , CH1SEL = 0 , CH2SEL = 0 , CH3SEL = 1 , CH4SEL = 0 , CH5SEL = 0 , CH6SEL = 0 , CH7
- (d) LDOK = 1 , CH0SEL = 0 , CH1SEL = 0 , CH2SEL = 0 , CH3SEL = 0 , CH4SEL = 0 , CH5SEL = 0 , CH6SEL = 1 , CH7
- (e) LDOK = 1 , CH0SEL = 1 , CH1SEL = 0 , CH2SEL = 1 , CH3SEL = 0 , CH4SEL = 1 , CH5SEL = 0 , CH6SEL = 1 , CH7
- (f) LDOK = 1 , CH0SEL = 1 , CH1SEL = 1 , CH2SEL = 1 , CH3SEL = 1 , CH4SEL = 1 , CH5SEL = 1 , CH6SEL = 1 , CH7

**图 26-86. 中间加载的加载点**

使能加载点之后，必须对要发生的加载设置 LDOK 位。这种情况下，加载将根据以下条件在下一个使能的加载点发生：

**表 26-15. 让加载在下一个使能的加载点发生的条件**

向以下寄存器写入了新的值时	结果
MOD 寄存器	MOD 寄存器将以其写入缓冲区值更新。
CNTIN 寄存器，且 CNTINC = 1	CNTIN 寄存器将以其写入缓冲区值更新。
C(n)V 寄存器，且 SYNCENm = 1 – 其中 m 表示通道对(n)和(n+1)	C(n)V 寄存器将以其写入缓冲区值更新。
C(n+1)V 寄存器，且 SYNCENm = 1 – 其中 m 表示通道对(n)和(n+1)	C(n+1)V 寄存器将以其写入缓冲区值更新。

**注**

- 如果 ELSjB 和 ELSjA 位不为零，则根据配置的输出模式生成通道(j)输出信号。如果 ELSjB 和 ELSjA 位为零，则生成的信号在通道(j)输出中不可用。

- 如果  $CHjIE = 1$ , 将在发生通道(j)匹配时生成通道(j)中断。
- 在中断加载, 通道输出和 FTM 计数器均不改变。软件必须及时在一个安全点设置中间加载。

### 26.4.26 全局时基(GTB)

全局时基(GTB)是一项 FTM 功能, 允许同步一个芯片上的多个 FTM 模块。下图以示例说明了如何利用 GTB 特性同步两个 FTM 模块。在本例中, FTM A 和 B 通道运作起来就好像只使用了一个 FTM 模块, 这就是全局时基。

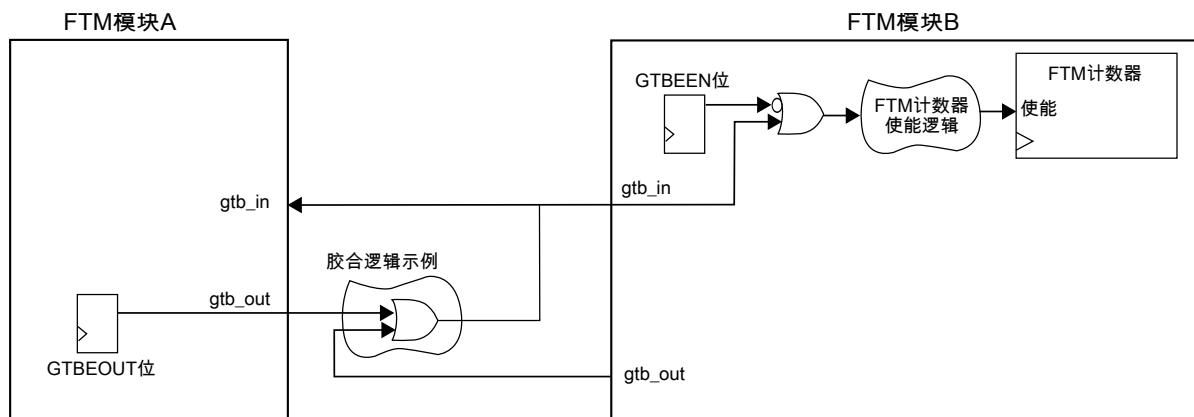


图 26-87. 全局时基(GTB)功能框图

GTB 功能可通过 CONF 寄存器中的 GTBEEN 和 GTBEOUT 位以及输入信号  $gtb\_in$  和输出信号  $gtb\_out$  实现。GTBEEN 位让  $gtb\_in$  能够控制 FTM 计数器使能信号:

- 如果  $GTBEEN = 0$ , 每一个 FTM 模块都将根据各自的配置模式独立工作。
- 如果  $GTBEEN = 1$ , 则只在  $gtb\_in$  为 1 的情况下使能 FTM 计数器更新。

在上图所述的配置中, 如果其中某个 FTM 模块的至少一个  $gtb\_out$  信号为 1, 则使能 FTM 模块 A 和 B 的 FTM 计数器。对于  $gtb\_in$  和  $gtb\_out$  信号的互连, 有多种可能的配置, 图中的示例胶合逻辑即体现了这一点。注意, 这些配置是独立于芯片的, 在 FTM 模块外部实施。有关芯片的具体实施, 请参见芯片专用 FTM 详情。

#### 注

- 为了使用 GTB 信号同步不同 FTM 模块的 FTM 计数器, 每个 FTM 模块的配置都应该确保其 FTM 计数器在  $gtb\_in$  信号为 1 时立刻开始计数。
- GTB 特性不提供 FTM 计数器的持续同步, 这就意味着在 FTM 工作期间 FTM 计数器可能丢失同步。GTB 特性只允许 FTM 计数器同时开始它们的工作。

### 26.4.26.1 使能全局时基(GTB)

要使能 GTB 特性, 请对每个参与的 FTM 模块执行以下步骤:

1. 停止 FTM 计数器: 向 SC[CLKS]写入 00b。
2. 将 FTM 编程为预期配置。FTM 计数器模式需要在所有参与模块之间保持一致。
3. 向 CONF[GTBEEN]写入 1, 同时向 CONF[GTBEOUT]写入 0。
4. 在 SC[CLKS]中选择预期的 FTM 计数器时钟源。时钟源需要在所有参与模块之间保持一致。
5. 复位 FTM 计数器: 向 CNT 寄存器中写入任意值。

要在上图所述配置中启动 GTB 特性, 请向用作时基的 FTM 模块中的 CONF[GTBEOUT]写入 1。

## 26.5 复位概述

无论何时, 只要有芯片复位, FTM 就会复位。

FTM 何时从复位状态中退出:

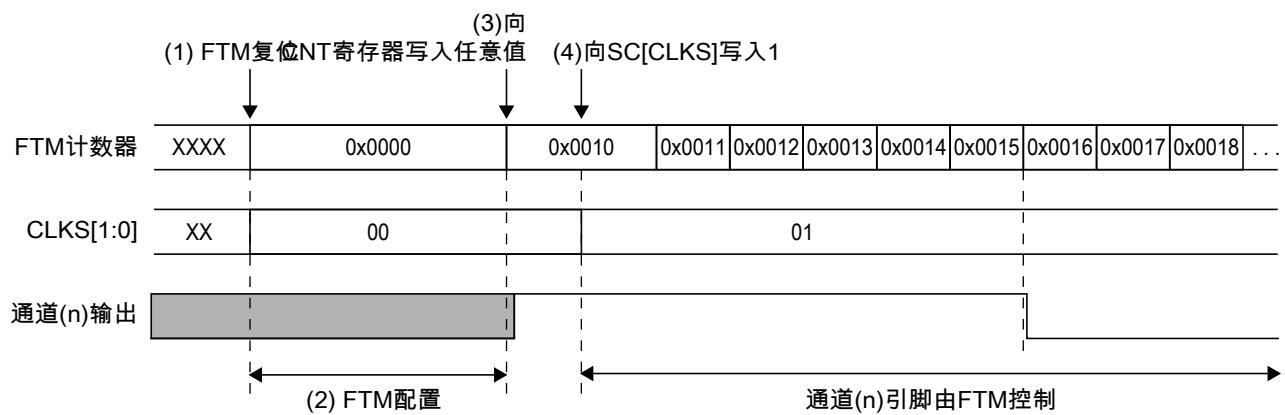
- FTM 计数器和预分频器计数器为零, 且已停止(CLKS[1:0] = 00b);
- 定时器溢出中断为零, 参见[定时器溢出中断](#);
- 通道中断为零, 参见[通道\(n\)中断](#);
- 故障中断为零, 参见[故障中断](#);
- 通道处于输入捕捉模式, 参见[输入捕捉模式](#);
- 通道输出为零;
- 通道引脚不由 FTM 控制(ELS(n)B:ELS(n)A = 0:0) (参见 CnSC 寄存器说明中的表格)。

下图展示了 FTM 在复位之后的行为方式。复位时 (项目 1), FTM 计数器将会禁用 (参见状态和控制寄存器中的 CLKS 字段说明), 其值将更新为零, 而且引脚不由 FTM 控制 (参见 CnSC 寄存器说明中的表格)。

复位之后, 应该对 FTM 进行配置 (项目 2)。有必要根据通道模式来定义 FTM 计数器模式、FTM 计数限制 (MOD 和 CNTIN 寄存器值)、通道模式和的 CnV 寄存器值。

因此, 建议向 CNT 寄存器中写入任意值 (项目 3)。采取这种写入操作之后, FTM 计数器会根据 CNTIN 寄存器值更新, 通道输出会根据其初始值更新 (处于输出比较模式的通道则例外) ([计数器复位](#))。

下一步是通过 CLKS[1:0]位选择 FTM 计数器时钟（项目 4）。有必要强调的一点是，CLKS[1:0]位不为零时，引脚将只由 FTM 控制（参见 CnSC 寄存器说明中的表格）。

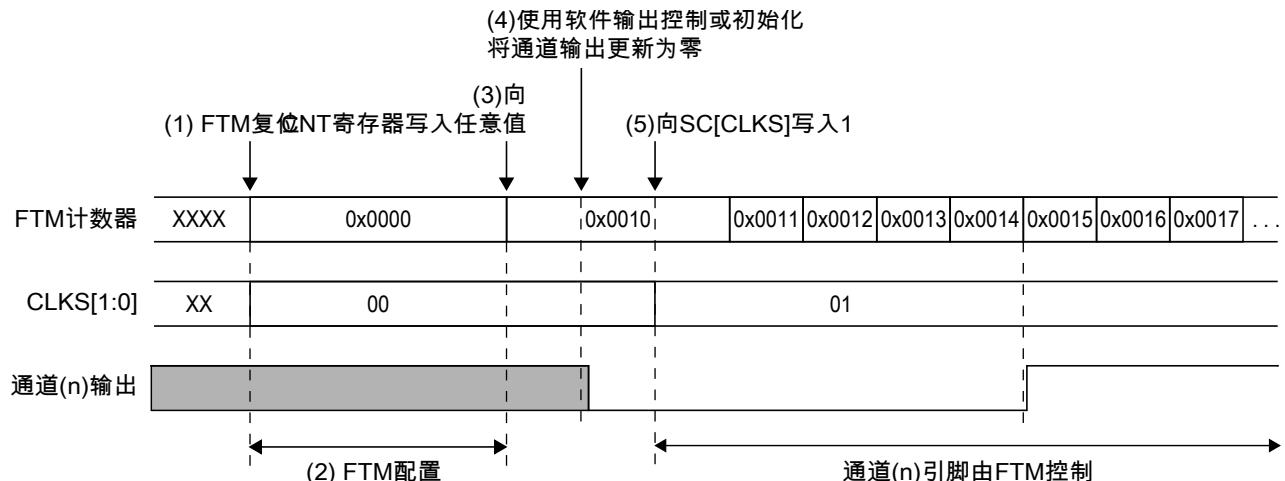


**注释：**

- CNTIN = 0x0010
- 通道(n)处于低真组合模式，CNTIN < C(n)V < C(n+1)V < MOD
- C(n)V = 0x0015

图 26-88. 通道(n)处于组合模式时复位之后的 FTM 行为方式

下图以示例说明了当通道(n)处于输出比较模式时，如果存在匹配，则切换通道(n)输出。在输出比较模式下，对 CNT 寄存器采取写入操作后（项目 3），通道输出不会更新为其初始值。这种情况下，利用软件输出控制（[软件输出控制](#)）或初始化（[初始化](#)）将通道输出更新为所选值（项目 4）。



**注释：**

- CNTIN = 0x0010
- 通道(n)处于输出比较模式，存在匹配时，将切换通道(n)输出
- C(n)V = 0x0014

图 26-89. 通道(n)处于输出比较模式时复位之后的 FTM 行为方式

## 26.6 FTM 中断

### 26.6.1 定时器溢出中断

(TOIE = 1)且(TOF = 1)时，将生成定时器溢出中断。

### 26.6.2 通道(n)中断

当(CHnIE = 1)且(CHnF = 1)时，会生成通道(n)中断。

### 26.6.3 故障中断

当FAULTIE = 1)且FAULTF = 1)时，生成故障中断。

## 26.7 初始化流程

建议遵照下列初始化流程对 FlexTimer 操作进行配置。该流程还可用于对 FlexTimer 操作进行全新配置。

- 定义 POL 位。
- 采用 SYNCHOM = 0 屏蔽通道输出。写入 OUTMASK 之后等待两个时钟周期再执行此操作，以便通道输出为安全值。
- (重新) 配置 FTM 计数器和通道以生成周期性信号 - 禁用时钟。若选定的模式为正交解调器模式，则禁用该模式。(重新) 配置的示例：
  - 写入 MOD。
  - 写入 CNTIN。
  - 针对所有待使用通道选择 OC、EPWM、CPWM、组合、互补模式
  - 选择高电平有效通道模式和低电平有效通道模式。
  - 针对所有待使用通道写入 CnV。
  - (重新) 配置死区和故障控制。
  - 不要在未进行软件同步的情况下使用 SWOC (见第 6 项)。
  - 不要在未进行软件同步的情况下使用反相 (见第 6 项)。
  - 不要使用初始化。
  - 不要改变极性控制。
  - 不要配置硬件同步
- 向 CNT 写入任意值。FTM 计数器复位，并根据新的配置更新通道输出。
- 使能时钟。将非 0 的值写入 CLKS[1:0]位。若在正交解调器模式下，则使能该模式。

- 针对 SWOC、反相和输出掩码配置软件同步（前两者为按需配置，后者为始终配置）
  - 针对“输出掩码写入 SYNC”选择同步 ( $SWSYNC = 0$ ,  $TRIG2 = 0$ ,  $TRIG1 = 0$ ,  $TRIG0 = 0$ ,  $SYNCHOM = 1$ ,  $REINIT = 0$ ,  $CNTMAX = 0$ ,  $CNTMIN = 0$ )
  - 写入 SYNCNF。
    - 硬件同步无法使能 ( $HWSOC = 0$ ,  $HWINVC = 0$ ,  $HWOM = 0$ ,  $HWWRBUF = 0$ ,  $HWRSTCNT = 0$ ,  $HWTRIGMODE = 0$ )。
    - 用于 SWOC 的软件同步（按需）:  $SWSOC = [0/1]$  且  $SWOC = [0/1]$ 。
    - 用于反相的软件同步（按需）:  $SWINVC = [0/1]$  且  $INVVC = [0/1]$ 。
    - 用于 SWOM 的软件同步（始终）:  $SWOM = 1$ 。不要为写入缓冲区使能软件同步（因为写入带写入缓冲区的寄存器是通过  $CLKS[1:0] = 2'b00$  实现的）:  $SWWRBUF = 0$  且  $CNTINC = 0$ 。
    - 用于计数器复位的软件同步（始终）:  $SWRSTCNT = 1$ 。
    - 增强同步（始终）:  $SYNCMODE = 1$
  - 如果使用了 SWOC ( $SWSOC = 1$  且  $SWOC = 1$ )，则写入 SWOCTRL 寄存器。
  - 如果使用了反相 ( $SWINVC = 1$  且  $INVVC = 1$ )，则写入 INVCTRL 寄存器。
  - 写入 OUTMASK 以使能屏蔽通道。
- 生成“软件触发写入 SYNC” ( $SWSYNC = 1$ ,  $TRIG2 = 0$ ,  $TRIG1 = 0$ ,  $TRIG0 = 0$ ,  $SYNCHOM = 1$ ,  $REINIT = 0$ ,  $CNTMAX = 0$ ,  $CNTMIN = 0$ )

# 第 27 章 脉冲宽度定时器(PWT)

## 27.1 简介

### 27.1.1 特性

脉宽定时器(PWT)包含下列特性：

- 采用 16 位分辨率自动测量脉冲宽度
- 正脉冲和负脉冲的单独脉宽测量
- 可编程的开始测量的触发沿
- 可编程测量逐次交错沿、上升沿或下降沿之间的时间
- 可编程预分频器，时钟输入作为 16 位计数器时基
- 两个可选时钟源—总线时钟和备用时钟
- 四个可选脉冲输入
- 可设定在脉冲宽度值更新及计数器溢出时生成中断

### 27.1.2 工作模式

下表说明了各种模式下 PWT 模块的工作情况。

模式	说明
Run	使能时，脉宽定时器模块处于激活状态。
Wait	使能时，若对应的中断使能，则脉宽定时器模块处于激活状态，并且可以执行唤醒功能。
Stop	当进入 Stop 模式且保留寄存器内容和工作状态时，脉宽定时器模块暂停。若以复位信号退出 Stop 模式，则模块复位。若以其他信号源退出 Stop 模式，则模块恢复退出时的工作状态。
Active Background	进入 Debug 模式后，PWT 会挂起所有计数和脉冲边沿检测操作，直到微控制器返回正常用户工作模式。只要 PWTSR 位（PWT 软件复位）未写入 1 且 PWT 模块仍然使能，那么当返回正常用户工作模式时，计数和边沿检测会恢复挂起时的数值。

### 27.1.3 功能框图

下图是脉宽定时器模块(PWT)的功能框图。

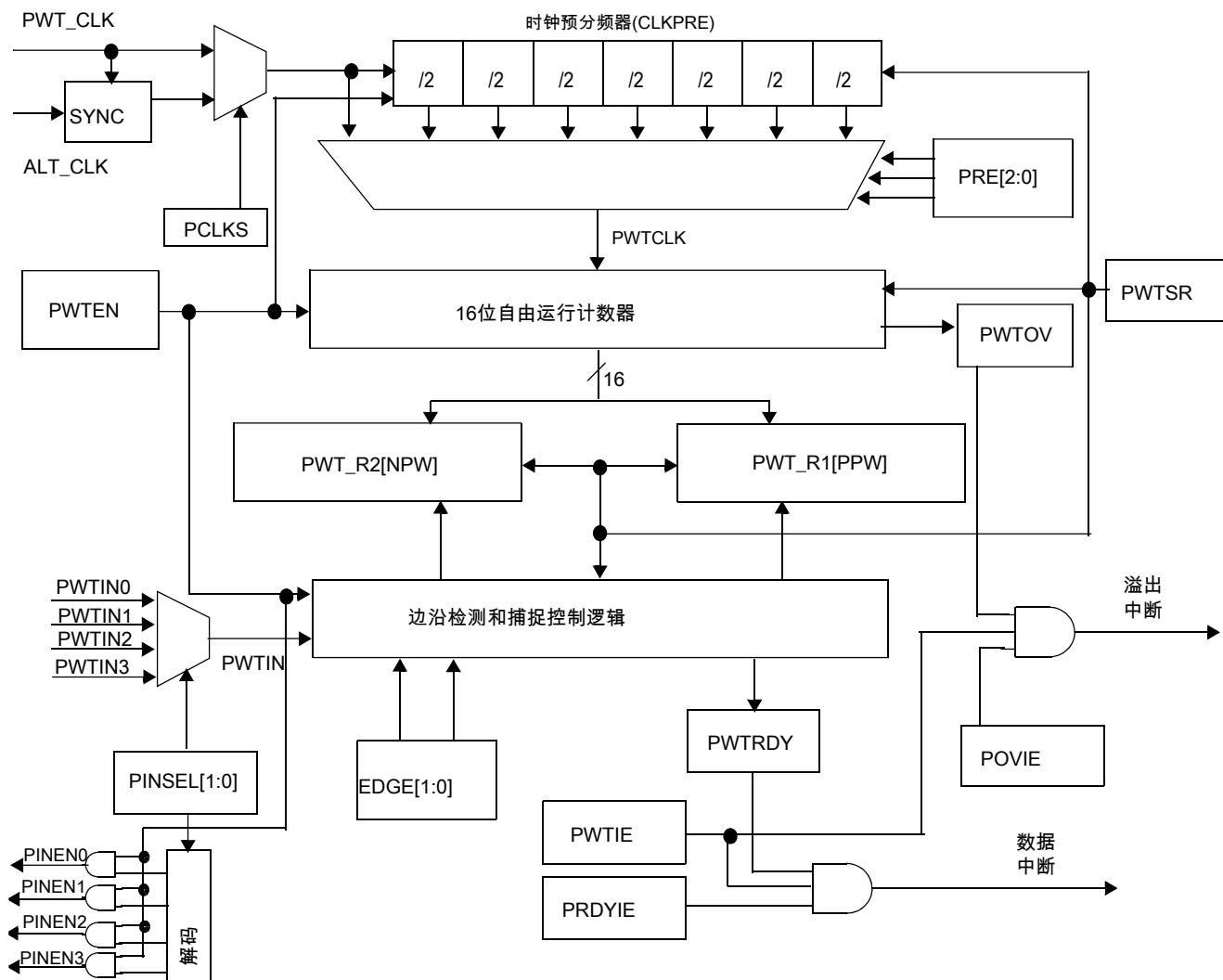


图 27-1. 脉冲宽度定时器 (PWT) 方框图

#### 注

PWT\_CLK 取决于芯片输入时钟。就该芯片而言，它是 TIMER\_CLK。

## 27.2 PWT 信号说明

表 27-1. PWT 信号说明

信号	I/O	上拉	说明
PWTIN[3:0]	I	否	脉冲输入
ALTCLK	I	否	计数器的备用时钟源

### 27.2.1 PWTIN[3:0] - 脉宽定时器捕捉输入

输入信号为可来自内部源或外部源的脉冲捕捉输入。PWT 输入由 PINSEL[1:0]选择并连接到脉宽定时器。如果输入来自外部源且选作 PWT 输入，那么输入端口由 PINSEL[1:0]自动使能为 PWT 功能。待测量脉宽的最小值为 1 个 PWTCLK 周期，任何比该数值窄的脉冲都会被 PWT 模块忽略。PWTCLK 周期时间取决于 PWT 时钟源选择和预分频器比例设置。

### 27.2.2 ALTCLK- 计数器的备用时钟源

PWT 具有备用时钟输入 ALTCLK，在 R1[PCLKS]置位后可将其选作计数器的时钟源。ALTCLK 输入必须由总线时钟同步。另外还必须适应占空比变化和时钟抖动，这样 ALTCLK 信号就一定不会超过总线频率的四分之一。通过通用端口引脚可共用 ALTCLK 引脚。有关此功能的引脚位置和优先级，请参见“引脚和连接”章节。

## 27.3 存储器映像和寄存器说明

PWT 存储器映射

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4003_3000	脉宽定时器寄存器 1 (PWT_R1)	32	R/W	0000_0000h	<a href="#">27.3.1/466</a>
4003_3004	脉宽定时器寄存器 2 (PWT_R2)	32	R	0000_0000h	<a href="#">27.3.2/468</a>

### 27.3.1 脉宽定时器寄存器 1 (PWT\_R1)

该寄存器定义 PWT 的通用控制和状态位。它还包含正脉宽内容。

地址: 4003\_3000h 基准 + 0h 偏移 = 4003\_3000h

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	PPW																
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	PCLKS	PINSEL		EDGE	PRE			PWTEN	PWTIE	PRDYIE	POVIE	0	0	PWTSR	PWTRDY	PWTOV	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

**PWT\_R1 字段描述**

字段	描述
31–16 PPW	正脉宽 捕捉到的正脉宽值。建议使用半字 ( 16 位 ) 或字 ( 32 位 ) 读出该值。
15 PCLKS	PWT 时钟源选择 控制 PWT 计数器时钟源的选择。 0 BUS_CLK 选作 PWT 计数器的时钟源。 1 备用时钟选作 PWT 计数器的时钟源。
14–13 PINSEL	PWT 脉冲输入选择 如果该 PWT 输入来自外部源，则使能相应的 PWT 输入端口。 00 PWTIN[0]使能。 01 PWTIN[1]使能。 10 PWTIN[2]使能。 11 PWTIN[3]使能。

下一页继续介绍此表...

## PWT\_R1 字段描述 (继续)

字段	描述
12–11 EDGE	<p>PWT 输入边沿灵敏度</p> <p>选择哪个边沿触发脉宽测量以及哪个边沿触发捕捉操作。如果用户需通过更改 EDGE[1:0]的值来改变触发和捕捉模式，则在更改 EDGE[1:0]的值后需进行 PWT 软件复位。清零 PWTEN 然后对其置位具有相同的效果。</p> <p>00 在第一个下降沿开始测量脉宽，在所有之后的下降沿上捕捉脉宽。      01 在第一个上升沿开始测量脉宽，在所有之后的上升沿和下降沿上捕捉脉宽。      10 在第一个下降沿开始测量脉宽，在所有之后的上升沿和下降沿上捕捉脉宽。      11 在第一个上升沿开始测量脉宽，在所有之后的上升沿上捕捉脉宽。</p>
10–8 PRE	<p>PWT 时钟预分频器(CLKPREG)设置</p> <p>选择为 PWT 计数器计时所需的时钟分频数值。</p> <p>000 时钟分频系数为 1。      001 时钟分频系数为 2。      010 时钟分频系数为 4。      011 时钟分频系数为 8。      100 时钟分频系数为 16。      101 时钟分频系数为 32。      110 时钟分频系数为 64。      111 时钟分频系数为 128。</p>
7 PWTEN	<p>PWT 模块使能</p> <p>使能/禁用 PWT 模块。为避免出现预料外的行为，只要 PWTEN 置位，就不要更改任何 PWT 配置。</p> <p>0 PWT 禁用。      1 PWT 使能。</p>
6 PWTIE	<p>PWT 模块中断使能</p> <p>使能 PWT 模块以产生中断。</p> <p>0 禁用 PWT 以产生中断。      1 使能 PWT 以产生中断。</p>
5 PRDYIE	<p>PWT 脉宽数据就绪中断使能</p> <p>只要 PWTIE 置位，则 PWTRDY 置位时，使能/禁用 PWT 以产生中断。</p> <p>0 当 PWTRDY 置位时，禁用 PWT 以产生中断。      1 当 PWTRDY 置位时，使能 PWT 以产生中断。</p>
4 POVIE	<p>PWT 计数器溢出中断使能</p> <p>当 PWTOV 因 PWT 计数器溢出而置位时，使能/禁用 PWT 以产生中断。</p> <p>0 当 PWTOV 置位时，禁用 PWT 以产生中断。      1 当 PWTOV 置位时，使能 PWT 以产生中断。</p>
3 PWTSR	<p>PWT 软复位</p> <p>执行 PWT 软复位。该字段始终读取为 0。</p> <p>0 未采取任何动作。      1 向该字段写 1 将执行 PWT 软复位。</p>

下一页继续介绍此表...

**PWT\_R1 字段描述 (继续)**

字段	描述
2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
1 PWTRDY	PWT 脉宽有效  指示 PWT 脉宽寄存器已更新且读取就绪。该字段通过读取 PWTRDY 清零，随后在 PWTRDY 置位时向 PWTRDY 位写入 0。向该字段写入 1 无效。PWTRDY 设置与 EDGE[1:0]位有关。  0 PWT 脉宽寄存器未更新。 1 PWT 脉宽寄存器已更新。
0 PWTOV	PWT 计数器溢出  指示 PWT 计数器已从 0x0000_0xFFFF 运行到 0x0000_0x0000。当 PWTOV 置位时，该字段通过向 PWTOV 写入 0 清零。向该字段写入 1 无效。如果清零该字段时发生另一次溢出事件，则清零失败。  0 PWT 计数器无溢出。 1 PWT 计数器从 0xFFFF 运行到 0x0000。

**27.3.2 脉宽定时器寄存器 2 (PWT\_R2)**

该寄存器包含负脉宽内容以及 16 位自由运行计数器内容。

地址: 4003\_3000h 基准 + 4h 偏移 = 4003\_3004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PWTC															NPW																
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**PWT\_R2 字段描述**

字段	描述
31–16 PWTC	PWT 计数器。建议使用半字 (16 位) 或字 (32 位) 读出该值。
NPW	负脉宽。建议使用半字 (16 位) 或字 (32 位) 读出该值。

## 27.4 功能说明

### 27.4.1 PWT 计数器和 PWT 时钟预分频器

脉宽定时器(PWT)通过 16 位自由运行计数器(PWT\_R2[PWTC])测量脉冲持续时间或 PWTIN 信号输入的周期。PWT 模块中有一个时钟预分频器(CLKPREG)，为 PWT\_R2[PWTC]提供分频时钟。时钟预分频器可通过 PWT\_R1[PCKLS]从总线时钟和备用时钟中选择时钟输入。

PWT 计数器使用 CLKPREG 中的分频时钟执行计数器。预分频器的频率可编程设置，时钟分频系数为 1、2、4、8、16、32、64、128（具体取决于 PRE[2:0]的设置）。

PWT 计数器一使能，就会使用选定且经过分频的时钟源开始计数；检测到第一个有效边沿（触发沿）时，计数器清零且不加载到寄存器。如果长时间未检测到有效的触发沿，那么计数器可能溢出。当 16 位自由运行计数器运行时，触发沿之后的任何待测边沿都会导致 PWT\_R2[PWTC]的数值上载至适当的脉宽寄存器。此时，PWT\_R2[PWTC]复位至 0x0000，并且时钟预分频器输出也将一同复位。

PWT\_R2[PWTC]随后会使用输入时钟再次开始递增。如果 PWT\_R2[PWTC]从 0xFFFF 运行到 0x0000，那么 PWTOV 位置位。

### 27.4.2 边沿检测和捕捉控制

边沿检测和捕捉控制部分检测测量触发沿，并控制更新脉宽寄存器的时间以及具体更新哪个。

根据 EDGE[1:0]的设置，边沿检测逻辑确定 PWTIN 上待测量脉宽的起始和结束沿，以及待更新的寄存器字段。详情请参见[边沿检测和捕捉控制](#)。

通过配置 PINSEL[1:0]可从四个源的其中之一选出 PWTIN。

必须注意的是，在边沿检测和捕捉逻辑内，PWT 使用系统从时钟以便采样并同步 PWTIN 脉冲，因此最小 PWTIN 脉宽由该从时钟频率确定。例如，如果该时钟频率为 50 MHz，那么最小 PWTIN 脉宽必须大于 20 ns (50 MHz 周期)，否则 PWT 将无法捕捉该脉冲且计数器将溢出。有关 PWT 的从时钟频率，请参见“芯片配置”章节。此外，如果长时间没有 PWTIN 宽度的任何有效边沿，那么 PWT 计数器也将溢出。

EDGE[1:0]为 00 时，第一个下降沿是开始测量脉宽的触发沿。计数器数值在每个后续下降沿上载至 PWT\_R1[PPW]。如果 EDGE[1:0] 设为 11，则第一个上升沿为开始测量脉冲宽度的触发沿。计数器数值在每个逐次上升沿上载至 PWT\_R2[NPW]。在这两种情况下，可测得 PWTIN 的周期。

如果 EDGE[1:0] 设为 01，则第一个上升沿为触发沿。脉宽测量从该边沿开始。PWT\_R1[PPW]在每个后续下降沿上载。PWT\_R2[NPW]在每个逐次上升沿上载。EDGE[1:0]为 10 时，第一个下降沿是测量脉宽的触发沿。PWT\_R2[NPW]在每个逐次上升沿上载，PWT\_R1[PPW]在每个逐次下降沿上载。这两种情况下，正脉冲和负脉冲的测量是单独进行的，且正脉宽上载至 PWT\_R1[PPW]。负脉宽上载至 PWT\_R2[NPW]。

下图阐明了 PWT 的触发沿检测和脉冲宽度寄存器更新过程。

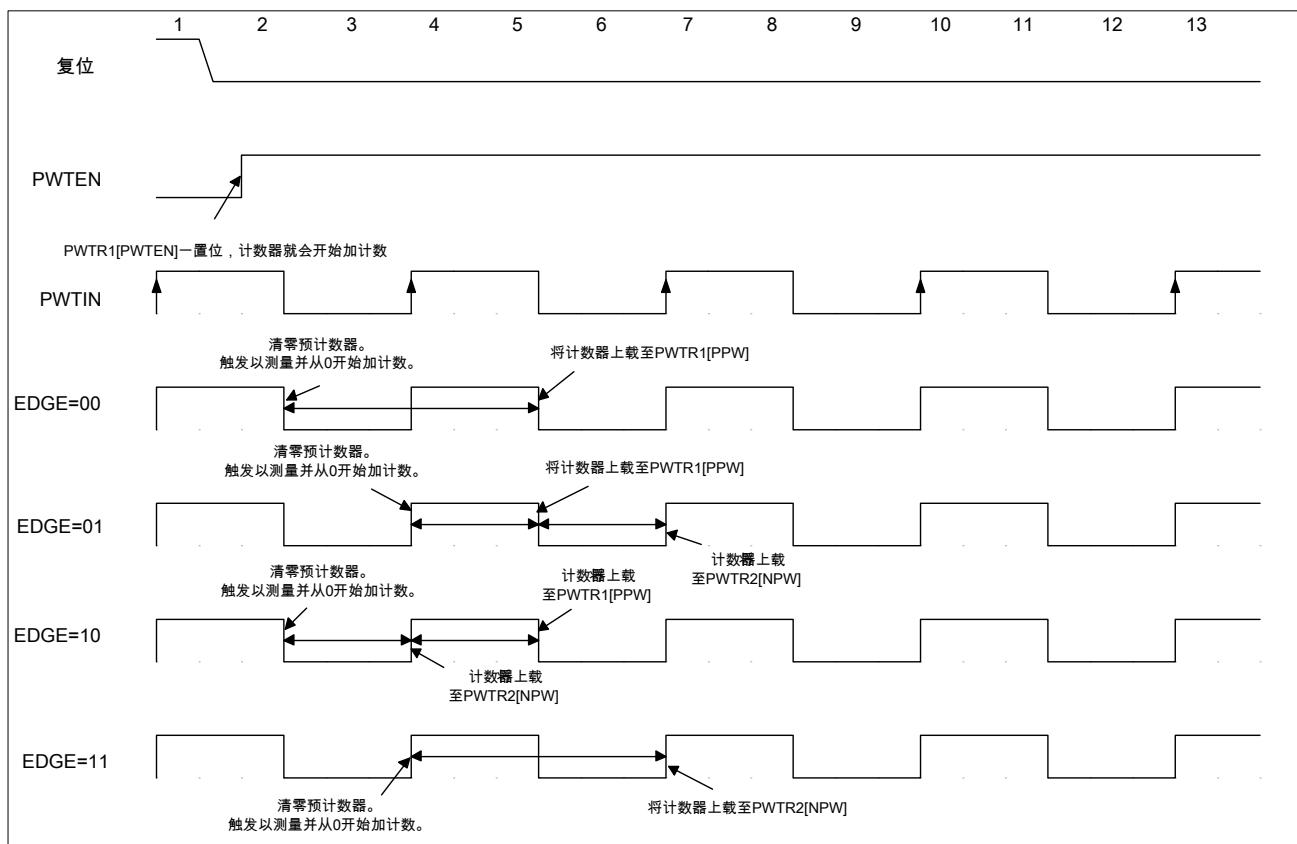


图 27-2. 触发沿检测和脉宽寄存器更新

PWT\_R1[PWTRDY]表明可根据 EDGE[1:0]的设置在 PWT\_R1[PPW]和/或 PWT\_R2[NPW]中读取数据。

- EDGE[1:0]为 00 时，只要一更新 PWT\_R1[PPW]，PWT\_R1[PWTRDY]就置位。
- EDGE[1:0]为 11 时，只要一更新 PWT\_R2[NPW]，PWT\_R1[PWTRDY]就置位。
- EDGE[1:0]为 01 时，只要一更新 PWTxPPH:L，PWT\_R1[PWTRDY]就置位，然后 PWT\_R2[NPW]更新。
- EDGE[1:0]为 10 时，只要一更新 PWT\_R2[NPW]，PWT\_R1[PWTRDY]就置位，然后 PWT\_R1[PPW]更新。

PWT\_R1[PWTRDY]置位后，已更新的脉宽寄存器将数据传输到对应的 16 位读缓冲区。只要芯片处于正常 Run 模式或 Debug 模式，脉宽寄存器的读取值就始终来自对应的读取缓冲区。在读取后向 PWT\_R1[PWTRDY]标志写入 0 可清零该字段。在清零 PWT\_R1[PWTRDY]前，都无法更新 16 位读缓冲区。但这并不影响 PWT 计数器的脉冲宽度寄存器上传。

如果完成了另一个脉冲测量并更新了脉宽寄存器，那么 PWT\_R1[PWTRDY]标志清零失败，也就是说，PWT\_R1[PWTRDY]仍然会置位，但是只要清零了该操作，就会再次更新 16 位读缓冲区。用户应当在清零 PWT\_R1[PWTRDY]之前完成脉宽数据读取以免数据缺失。此机制可确保当 MCU 没有足够的时间读第一个脉冲测量值时，第二个脉冲测量值不会丢失。该机制通过 MCU 复位而自动重启：向 PWT\_R1[PWTSR]写入 1 或者在向 PWT\_R1[PWTEN]写入 0 后向其写入 1。

下图阐明了脉冲宽度寄存器的缓冲机制：

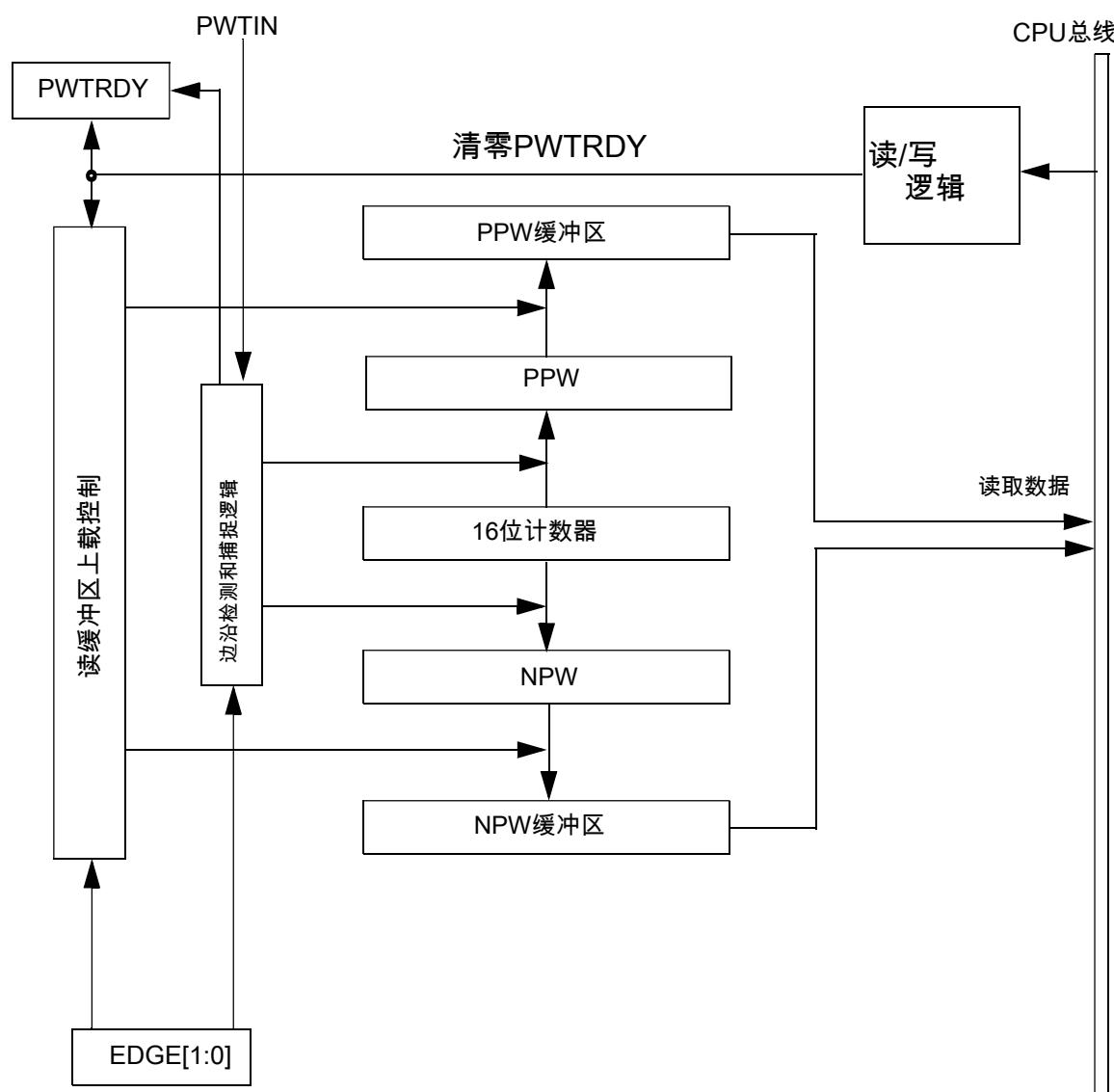


图 27-3. 脉宽寄存器的缓冲机制

如果 PWT 完成了任何脉宽测量，那么在该数据上载至脉宽寄存器后会生成一个使 PWT\_R2[CNTR] 和时钟预分频器输出复位的信号。为确保不遗漏计数，在完成脉宽测量后的一个总线时钟周期内会进行 PWT\_R2[CNTR] 和时钟预分频器输出的复位。

## 27.5 复位

### 27.5.1 通用

### 27.5.2 复位操作说明

PWT 软复位机制已嵌入到 PWT 中，用于复位/重新启动脉冲宽度定时器。向 PWT\_R1[PWTSR] 写入 1 即可触发 PWT 软复位。（该字段始终读取为 0）。不像 CPU 复位，PWT 复位不会将 PWT 内的所有内容恢复到其复位状态。下列步骤可用于说明复位操作。

1. PWT 计数器设为 0x0000。
2. PWT 计数器的 16 位缓冲区复位。
3. PWT 时钟预分频器输出复位。
4. 边沿检测逻辑复位。
5. 捕获逻辑复位，同时重新启动脉冲宽度寄存器的闭锁机制。
6. PWT\_R1[PPW] 和 PWT\_R2[NPW] 设为 0x0000。
7. PWT\_R1[PWTOV] 和 PWT\_R1[PWTRDY] 设为 0。
8. 所有其他 PWT 寄存器设置不变。

向 PWT\_R1[PWTEN] 写入 0 同样可以获得上述效果，除了复位状态将保持到 PWT\_R1[PWTEN] 置 1。

## 27.6 中断

### 27.6.1 中断操作说明

PWT 的另一个主要构成部分是中断控制逻辑。PWT\_R1[PWTOV] 和 PWT\_R1[POVIE] 置位后，可生成 PWT 溢出中断。PWT\_R1[PWTRDY] 位和 PWT\_R1[PRDYIE] 置位后，可生成脉宽数据就绪中断。PWT\_R1[PWTIE] 控制 PWT 模块的中断生成。生成中断并不影响 PWT 的功能。

## 27.6.2 应用示例

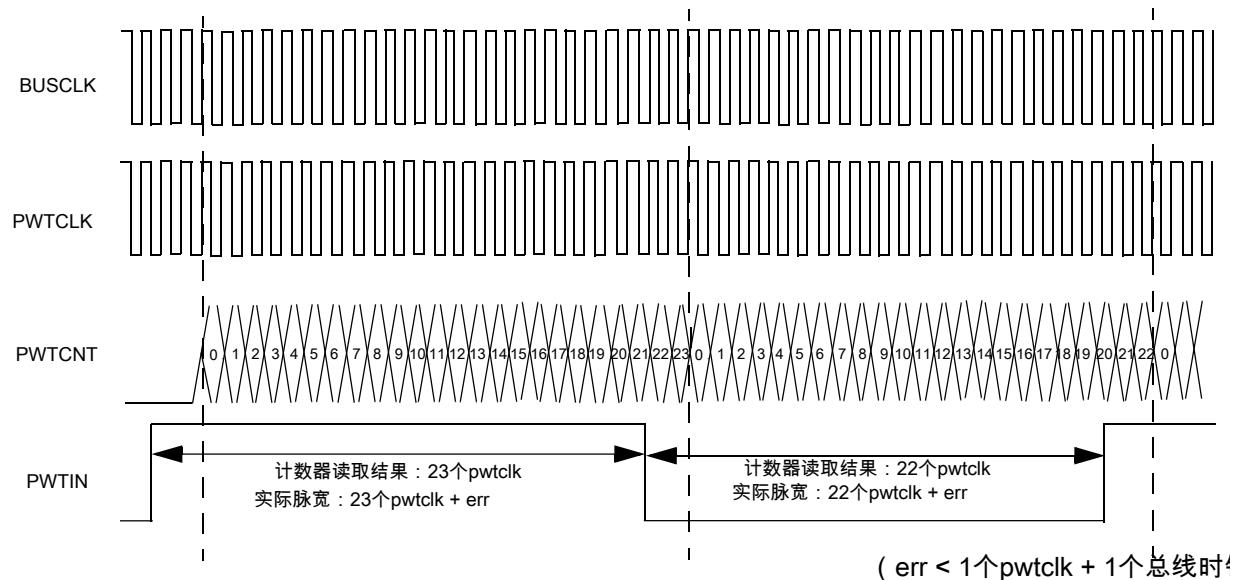


图 27-4. PWTCLK 为 1 分频总线时钟的示例

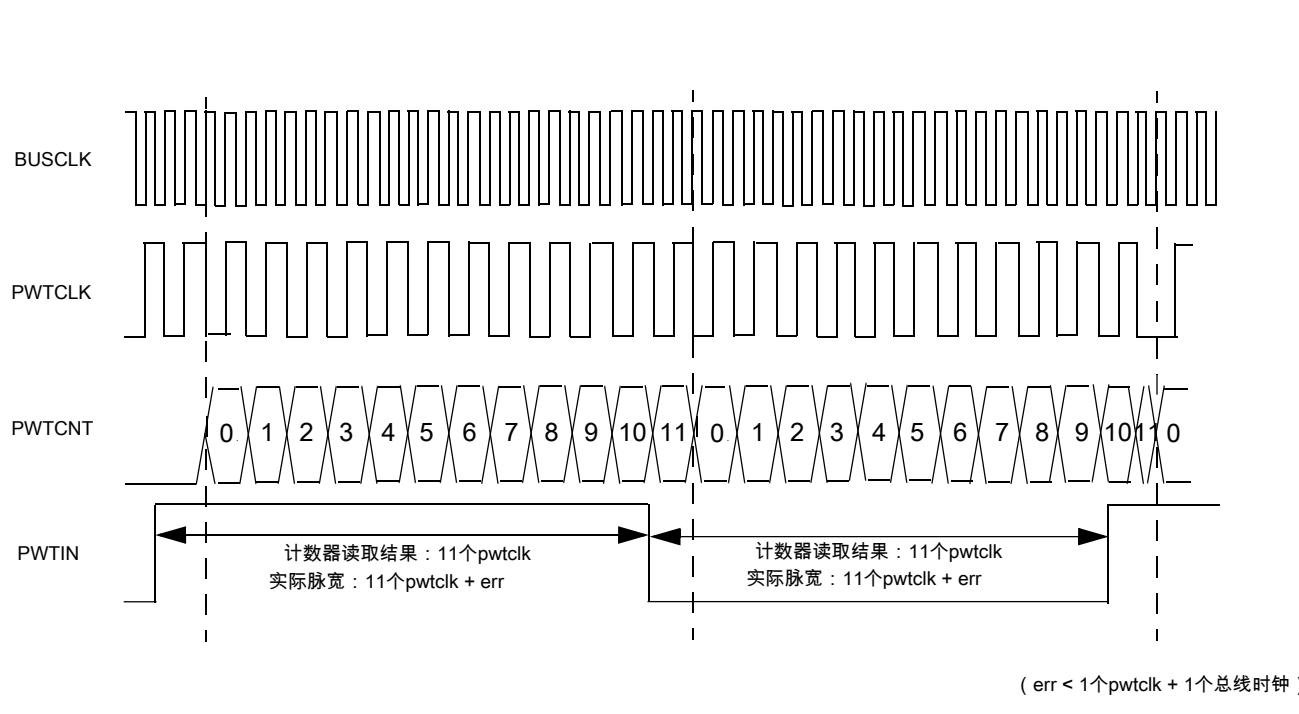


图 27-5. PWTCLK 为 2 分频总线时钟的示例

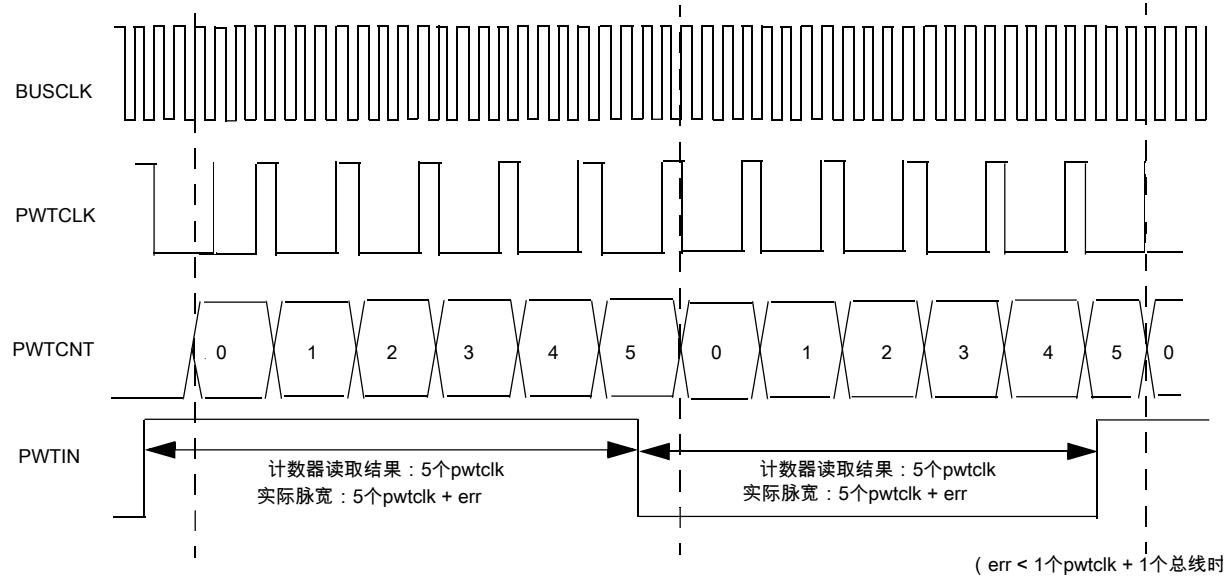


图 27-6. PWTCLK 为 4 分频总线时钟的示例

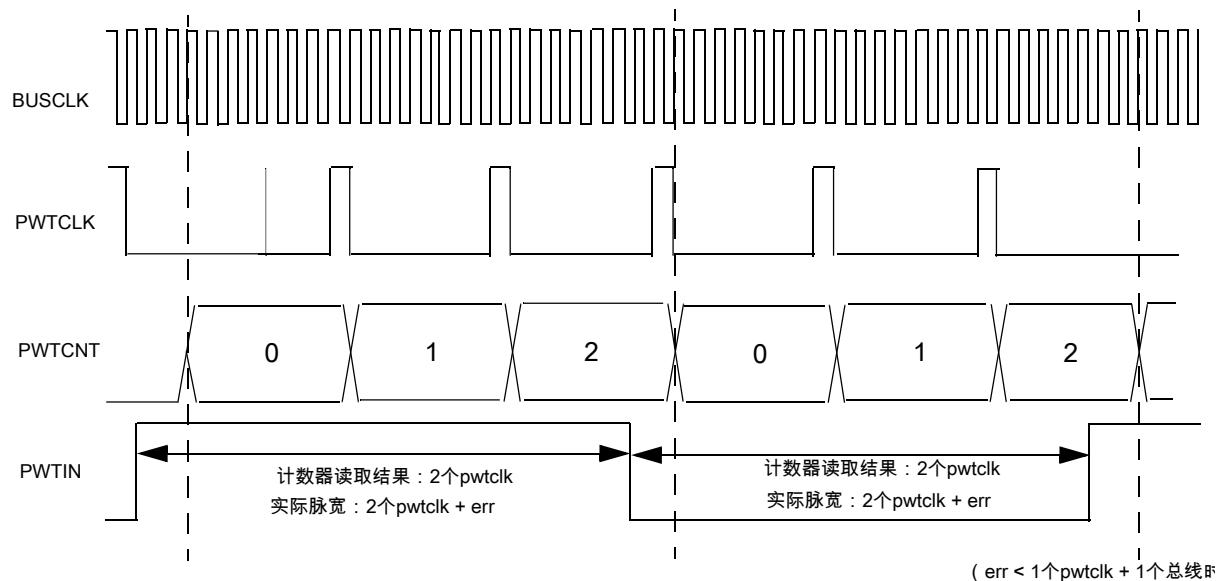


图 27-7. PWTCLK 为 8 分频总线时钟的示例

# 第 28 章 周期性中断定时器(PIT)

## 28.1 简介

### 注

关于与具体芯片相关的本模块详细操作示例，请参见芯片配置信息。

PIT 模块是一组定时器，用于生成中断和触发脉冲。

### 28.1.1 结构框图

下图是 PIT 模块的结构框图。

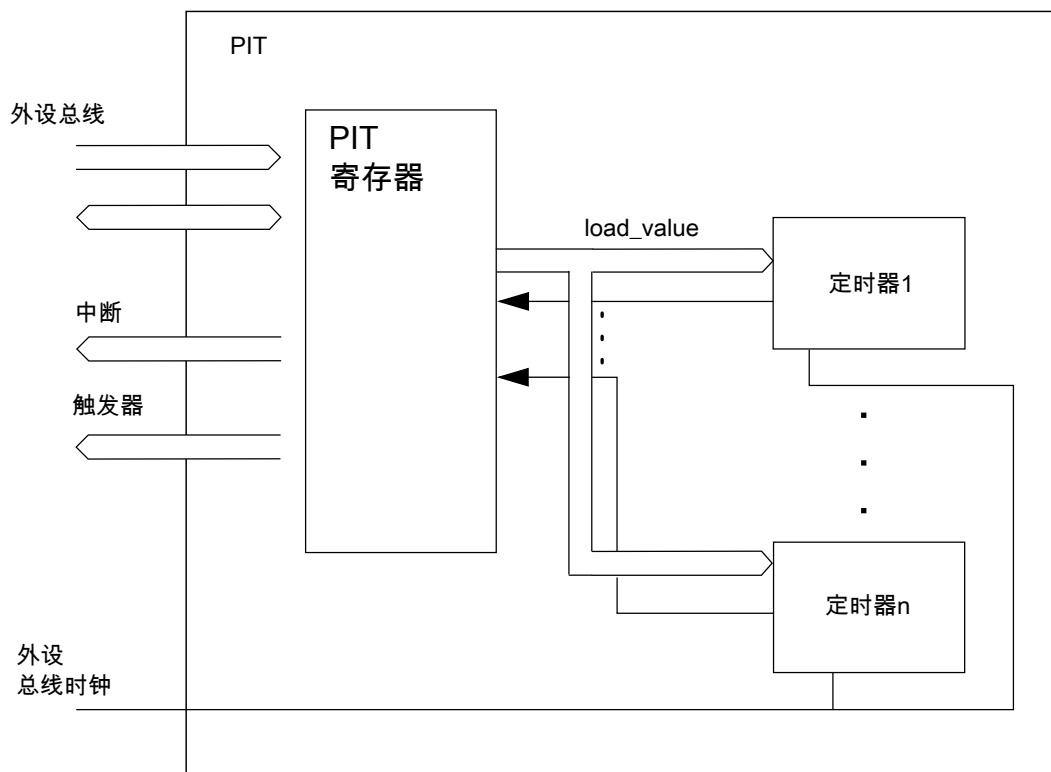


图 28-1. PIT 结构框图

### 注

有关该 MCU 使用的 PIT 通道数，请参见各芯片的 PIT 信息。

## 28.1.2 特性

该数据块具有如下主要特性：

- 定时器能够生成触发脉冲
- 定时器能够生成中断
- 可屏蔽中断
- 每个定时器都具有独立的超时周期

## 28.2 信号说明

PIT 模块没有外部引脚。

## 28.3 存储器映像/寄存器说明

本节详细说明 PIT 模块中可访问的所有寄存器。

- 保留寄存器将读取为 0，写操作将无效。
- 有关该 MCU 使用的 PIT 通道数，请参见各芯片的 PIT 信息。

**PIT 存储器映射**

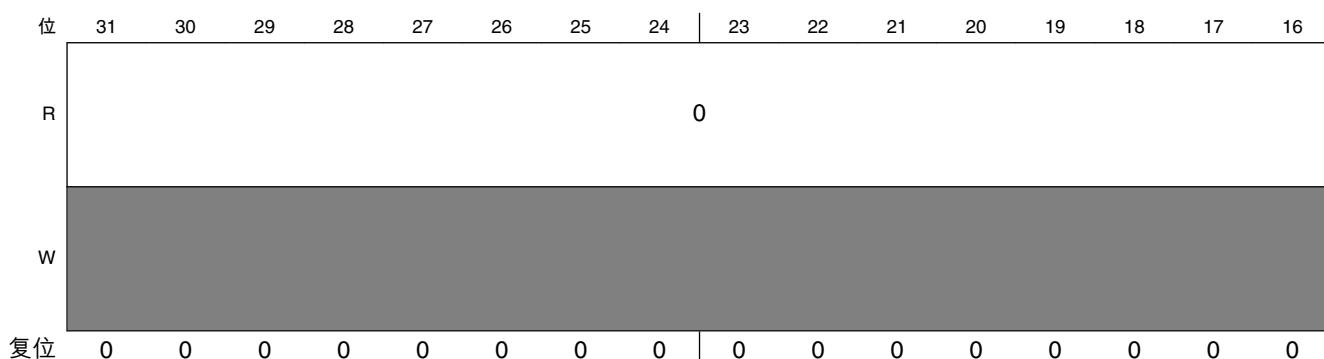
绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4003_7000	PIT 模块控制寄存器 (PIT_MCR)	32	R/W	0000_0006h	<a href="#">28.3.1/477</a>
4003_7100	定时器加载值寄存器 (PIT_LDVAL0)	32	R/W	0000_0000h	<a href="#">28.3.2/478</a>
4003_7104	当前定时器值寄存器 (PIT_CVAL0)	32	R	0000_0000h	<a href="#">28.3.3/479</a>
4003_7108	定时器控制寄存器 (PIT_TCTRL0)	32	R/W	0000_0000h	<a href="#">28.3.4/479</a>
4003_710C	定时器标志寄存器 (PIT_TFLG0)	32	R/W	0000_0000h	<a href="#">28.3.5/480</a>
4003_7110	定时器加载值寄存器 (PIT_LDVAL1)	32	R/W	0000_0000h	<a href="#">28.3.2/478</a>
4003_7114	当前定时器值寄存器 (PIT_CVAL1)	32	R	0000_0000h	<a href="#">28.3.3/479</a>
4003_7118	定时器控制寄存器 (PIT_TCTRL1)	32	R/W	0000_0000h	<a href="#">28.3.4/479</a>
4003_711C	定时器标志寄存器 (PIT_TFLG1)	32	R/W	0000_0000h	<a href="#">28.3.5/480</a>

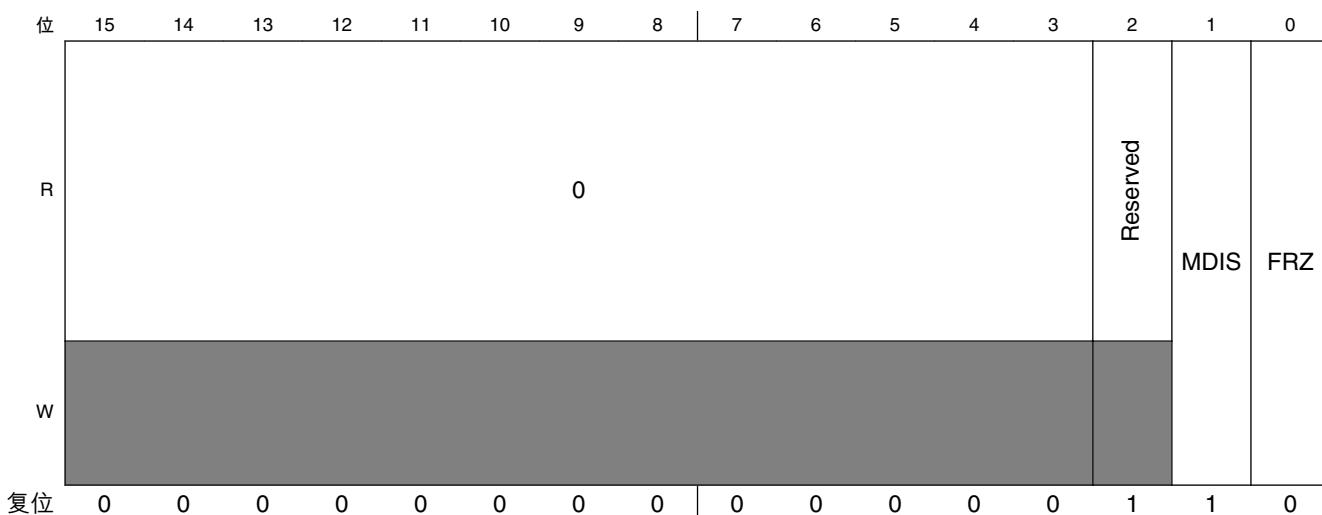
### 28.3.1 PIT 模块控制寄存器 (PIT\_MCR)

当 PIT 进入 Debug 模式时，此寄存器可用于启用或禁用 PIT 定时器时钟并控制定时器。

访问：用户读取/写入

地址: 4003\_7000h 基准 + 0h 偏移 = 4003\_7000h





### PIT\_MCR 字段描述

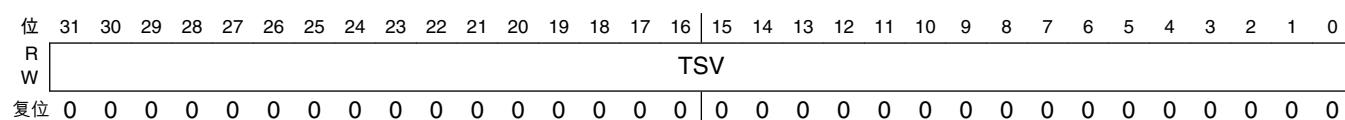
字段	描述
31–3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 Reserved	此字段为保留字段。
1 MDIS	模块禁用 - (PIT 部分) 禁用标准定时器。必须在执行任何其他设置之前使能该字段。 0 标准 PIT 定时器的时钟使能。 1 标准 PIT 定时器的时钟禁用。
0 FRZ	冻结 当器件进入 Debug 模式时，允许停止定时器。 0 定时器在 Debug 模式下继续运行。 1 定时器在 Debug 模式下停止运行。

### 28.3.2 定时器加载值寄存器 (PIT\_LDVAL $n$ )

这些寄存器选择定时器中断的定时溢出周期。

访问： 用户读取/写入

Address: 4003\_7000h base + 100h offset + (16d × i), where i=0d to 1d



**PIT\_LDVAL<sub>n</sub>** 字段描述

字段	描述
TSV	定时器起始值  设置定时器起始值。定时器将倒计时至 0，然后生成一个中断并再次加载该寄存器值。将新值写入该寄存器不会重启定时器，定时器到期后会加载新值。要中止当前周期并用新值开始一个定时器周期，必须先禁用该定时器然后再将其使能。

**28.3.3 当前定时器值寄存器 (PIT\_CVAL<sub>n</sub>)**

这些寄存器指示当前定时器位置。

访问：仅限用户读取

Address: 4003\_7000h base + 104h offset + (16d × i), where i=0d to 1d

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**PIT\_CVAL<sub>n</sub>** 字段描述

字段	描述
TVL	当前定时器值  代表当前定时器值（若定时器已使能）。  注： <ul style="list-style-type: none"> <li>• 若定时器已禁用，请勿使用该字段，因为其值不可靠。</li> <li>• 定时器使用向下计数器。如果 MCR[FRZ]置位，定时器在 Debug 模式下会被冻结。</li> </ul>

**28.3.4 定时器控制寄存器 (PIT\_TCTRL<sub>n</sub>)**

这些寄存器包含每个定时器的控制位。

访问：用户读取/写入

Address: 4003\_7000h base + 108h offset + (16d × i), where i=0d to 1d

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**PIT\_TCTRL*n*** 字段描述

字段	描述
31–3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 CHN	链模式  激活时，定时器 n-1 需先到期，定时器 n 才能递减 1。  不能链接定时器 0。  0 定时器不链接。 1 定时器链接到前一定时器。例如，对于通道 2，若该字段置位，则定时器 2 链接到定时器 1。
1 TIE	定时器中断使能  某个中断挂起或 TFLGn[TIF]置位时，使能该中断将立即引起中断事件。为避免这种情况，必须先清零相关的 TFLGn[TIF]。  0 定时器 n 的中断请求禁用。 1 只要 TIF 置位，就会请求中断。
0 TEN	定时器使能  使能或禁用定时器。  0 定时器 n 禁用。 1 定时器 n 使能。

**28.3.5 定时器标志寄存器 (PIT\_TFLG*n*)**

这些寄存器保存 PIT 中断标志。

访问：用户读取/写入

Address: 4003\_7000h base + 10Ch offset + (16d × i), where i=0d to 1d

位	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R									0								
W																	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R									0							TIF	
W																w1c	
复位	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

**PIT\_TFLG*n*** 字段描述

字段	描述
31–1 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

下一页继续介绍此表...

**PIT\_TFLGn 字段描述 (继续)**

字段	描述
0 TIF	<p>定时器中断标志</p> <p>在定时器周期结束时置 1。将 1 写入该标志可将其清零。写入 0 则无效。若使能或 TCTRLn[TIE] = 1 ,TIF 将引发中断请求。</p> <p>0 定时溢出尚未发生。 1 定时溢出已经发生。</p>

## 28.4 功能说明

本节说明该模块的功能。

### 28.4.1 常规操作

本节详细说明该模块的内部操作。每个定时器都可用于生成触发脉冲和中断。每个中断都可用于单独的中断线。

#### 28.4.1.1 定时器

定时器在使能时定期生成触发脉冲。定时器加载 LDVAL 寄存器指定的起始值，倒数至 0，然后再次加载相应的起始值。每次定时器达到 0 时，它就会生成一个触发脉冲并置位中断标志。

所有中断都可通过设置 TCTRLn[TIE] 来使能或屏蔽。新的中断只能在清除上一个中断之后生成。

需要时，定时器的当前计数值可通过 CVAL 寄存器读取。

通过 TCTRLn[TEN] 先禁用定时器再将其使能可重启计数周期。请参见下图。

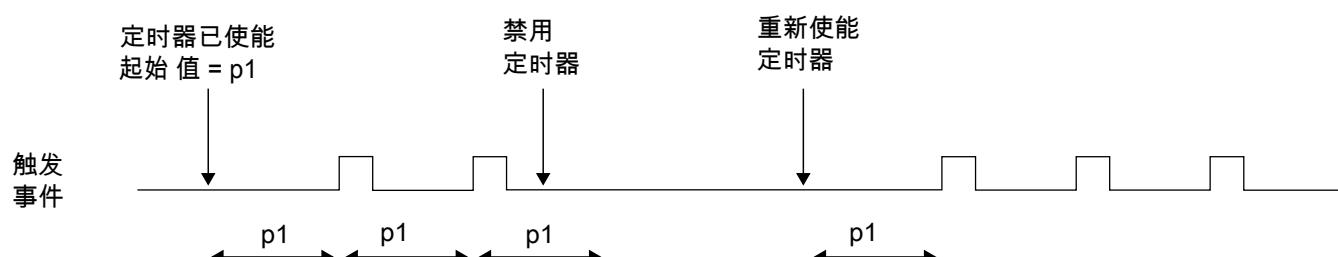


图 28-2. 停止和启动定时器

先禁用定时器，设置新加载值，然后再次使能定时器，可以修改运行中的定时器的计数周期。参见下图。

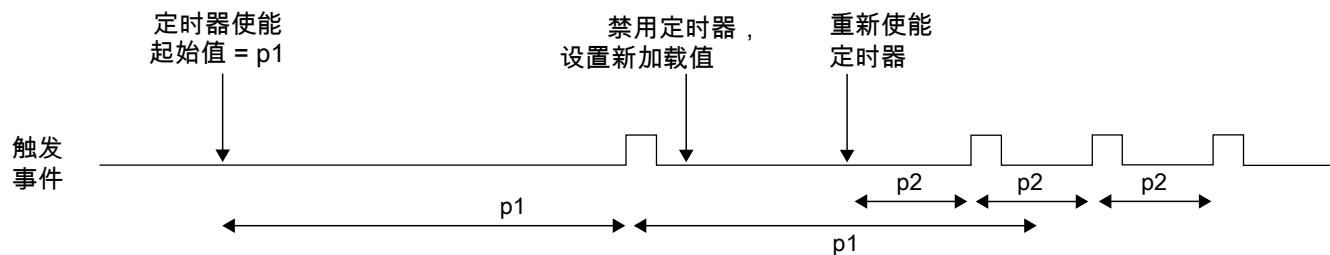


图 28-3. 修改运行中的定时器的周期

还可以在不重启定时器的情况下更改计数周期，方法是将新的加载值写入 LDVAL。该值将在下一个触发事件之后加载。参见下图。

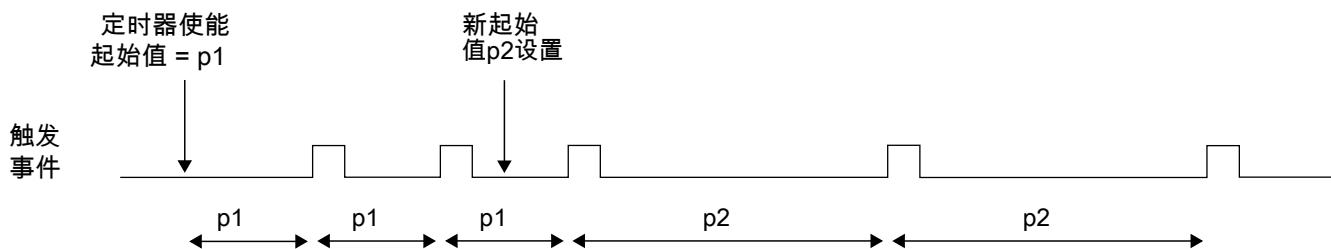


图 28-4. 动态设置新加载值

### 28.4.1.2 Debug 模式

在 Debug 模式下，定时器将根据 MCR[FRZ]决定是否冻结。其目的是协助软件开发，使开发人员能够暂停处理器，调查系统的当前状态，如定时器值等，然后继续运行。

### 28.4.2 中断

所有定时器都支持中断生成。相关的向量地址和优先级，请参见 MCU 规范。

定时器中断可通过置位 TCTRLn[TIE]来使能。相关定时器发生定时溢出时，TFLGn[TIF]置位为 1；将 1 写入对应的 TFLGn[TIF]可将其清零。

### 28.4.3 链接定时器

如果某个定时器使链模式处于使能状态，那么只有在上一个定时器溢出后，它才会开始计时。因此，如果定时器 n-1 已倒数至 0，定时器 n 的值将递减 1。这样就能将某些定时器链接起来形成更长的定时器。第一个定时器（定时器 0）不能链接至任何其他定时器。

## 28.5 初始化和应用信息

在配置示例中：

- PIT 时钟的频率为 50 MHz。
- 定时器 1 每隔 5.12 ms 生成一个中断。
- 定时器 3 每隔 30 ms 生成一个触发事件。

PIT 模块必须通过将 0 写入 MCR[MDIS] 来激活。

50 MHz 时候总频率等于时钟周期 20 ns。定时器 1 需要每隔  $5.12 \text{ ms} / 20 \text{ ns} = 256,000$  个周期触发一次，定时器 3 每隔  $30 \text{ ms} / 20 \text{ ns} = 1,500,000$  个周期触发一次。LDVAL 寄存器触发器的值计算如下：

LDVAL 触发器 = (周期 / 时钟周期) - 1

这意味着将 0x0003E7FF 和 0x0016E35F 分别写入 LDVAL1 和 LDVAL3。

定时器 1 的中断通过置位 TCTRL1[TIE] 来使能。定时器通过将 1 写入 TCTRL1[TEN] 来启动。

定时器 3 只能用于触发。因此，定时器 3 通过将 1 写入 TCTRL3[TEN] 来启动。TCTRL3[TIE] 保持为 0。

下面的示例代码与上述设置一致：

```
// turn on PIT
PIT_MCR = 0x00;

// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

## 28.6 链接定时器配置示例

在配置示例中：

- PIT 时钟的频率为 100 MHz。
- 定时器 1 和定时器 2 可用。
- 每隔 1 分钟应发生一次中断。

PIT 模块需要通过将 0 写入 MCR[MDIS]来激活。

100 MHz 时钟频率相当于 10 ns 的时钟周期，因此 PIT 需要计数 60 亿个周期，这不是单个定时器能够做到的。因此，将定时器 1 设置为每 6 s (6 亿个周期) 触发一次。定时器 2 链接至定时器 1，经过编程后触发 10 次。

LDVAL 寄存器触发器的值等于周期数减 1，因此 LDVAL1 接收值 0x23C345FF，LDVAL2 接收值 0x00000009。

定时器 2 的中断通过置位 TCTRL2[TIE]来使能，链模式通过置位 TCTRL2[CHN]来激活，定时器通过将 1 写入 TCTRL2[TEN]来启动。TCTRL1[TEN]需要置位，TCTRL1[CHN]和 TCTRL1[TIE]需要清零。

下面的示例代码与上述设置一致：

```
// turn on PIT
PIT_MCR = 0x00;

// Timer 2
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 |= TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2

// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```

# 第 29 章 实时计数器 (RTC)

## 29.1 简介

实时计数器(RTC)由一个 16 位计数器、一个 16 位比较器、若干个基于二进制和基于十进制的预分频器、三个时钟源、一个可编程周期性中断和一个可编程外部切换脉冲输出组成。该模块可用于当日时间、日历或任何任务调度功能。它还能充当周期唤醒，将 MCU 从低功耗模式、Stop 模式和 Wait 模式中唤醒而无需外部组件。

## 29.2 特性

RTC 模块特性包括：

- 16 位向上计数器
  - 16 位模数匹配限制
  - 软件可控制的周期性匹配中断
- 可通过软件选择预分频器输入的时钟源；集成可编程 16 位预分频器
  - OSC 32.768KHz (标称值)。
  - LPO ( $\sim 1$  kHz)
  - 总线时钟
  - 内部基准时钟(32 kHz)

### 29.2.1 工作模式

本节定义 RTC 在 Stop、Wait 和 Background Debug 模式下的工作。

#### 29.2.1.1 Wait 模式

如果 RTC 在执行 WAIT 指令前已使能，那么在 Wait 模式下会继续运行。因此，如果实时中断使能，就可利用 RTC 使 MCU 离开 Wait 模式。为实现最低电流消耗，如果 RTC 在 Wait 模式期间无需用作中断源，那么必须通过软件使其停止。

### 29.2.1.2 Stop 模式

如果在执行 STOP 指令之前已启用 RTC，则此 RTC 将继续在 Stop 模式下运行。因此，如果已使能实时中断，RTC 可用于使 MCU 退出 Stop 模式，而无需外部组件。

### 29.2.2 结构框图

下图显示的是 RTC 模块的结构框图。

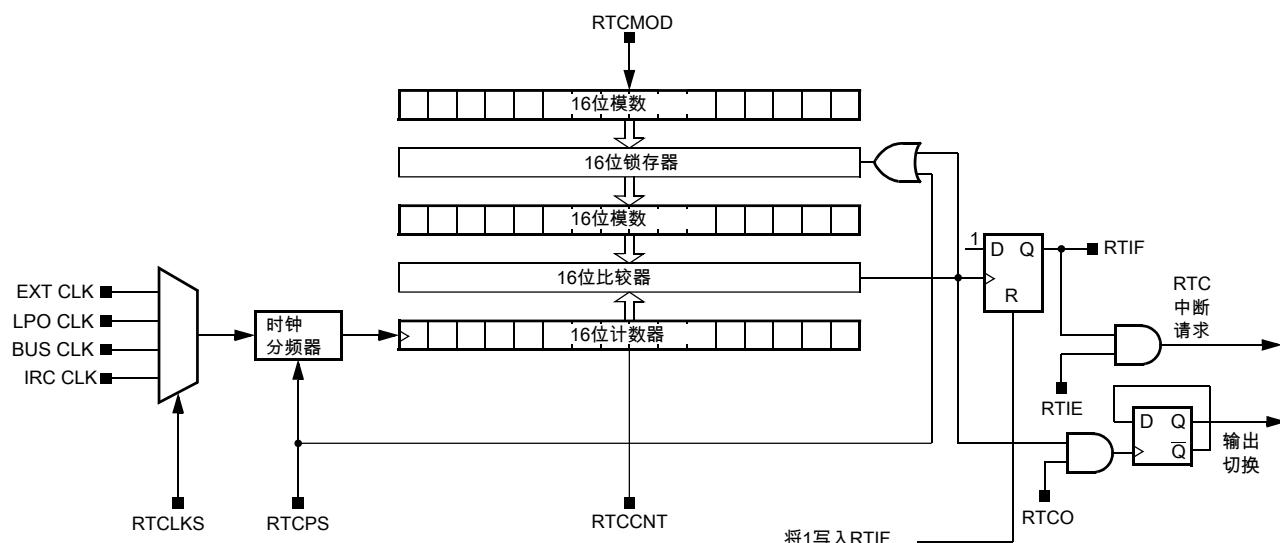


图 29-1. 实时计数器(RTC)结构框图

## 29.3 外部信号说明

RTCO 是 RTC 的输出。MCU 复位后，RTC\_SC[RTCO]置位为高电平。计数器溢出时，电平输出切换。

## 29.4 寄存器定义

RTC 包括状态和控制寄存器、16 位计数器寄存器和 16 位模数寄存器。

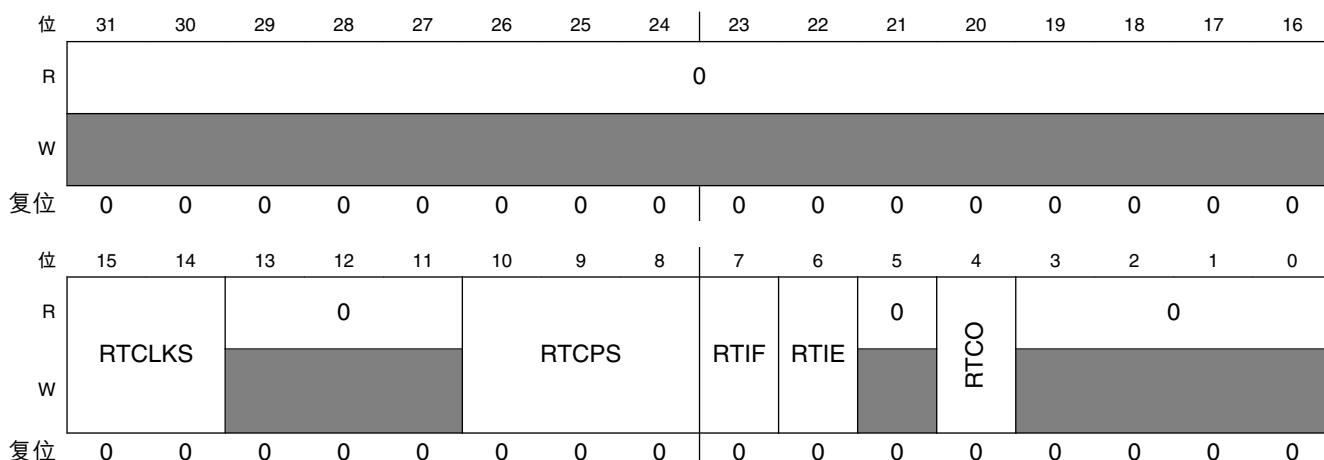
## RTC 存储器映射

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4003_D000	RTC 状态和控制寄存器 (RTC_SC)	32	R/W	0000_0000h	29.4.1/487
4003_D004	RTC 模数寄存器 (RTC_MOD)	32	R/W	0000_0000h	29.4.2/488
4003_D008	RTC 计数器寄存器 (RTC_CNT)	32	R	0000_0000h	29.4.3/489

### 29.4.1 RTC 状态和控制寄存器 (RTC\_SC)

RTC\_SC 包含实时中断状态标志(RTIF)和切换输出使能位(RTCO)。

地址: 4003\_D000h 基准 + 0h 偏移 = 4003\_D000h



#### RTC\_SC 字段描述

字段	描述
31–16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15–14 RTCLKS	实时时钟源选择  该读/写字段选择 RTC 预分频器的时钟源输入。更改时钟源会将预分频器和 RTCCNT 计数器清零。复位会将 RTCLKS 清除为 00。  00 外部时钟源。 01 实时时钟源为 1 kHz (LPOCLK)。 10 内部基准时钟(ICSIRCLK)。 11 总线时钟。
13–11 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
10–8 RTCPS	实时时钟预分频器选择

下一页继续介绍此表...

## RTC\_SC 字段描述 (继续)

字段	描述
	<p>该读/写字段为时钟源选择基于二进制或基于十进制的分频值。更改预分频器值会将预分频器和 RTCCNT 计数器清零。复位会将 RTCPS 清除为 000。</p> <p>000 关闭            001 如果 RTCLKS = x0 , 它为 1 ; 如果 RTCLKS = x1 , 它为 128。            010 如果 RTCLKS = x0 , 它为 2 ; 如果 RTCLKS = x1 , 它为 256。            011 如果 RTCLKS = x0 , 它为 4 ; 如果 RTCLKS = x1 , 它为 512。            100 如果 RTCLKS = x0 , 它为 8 ; 如果 RTCLKS = x1 , 它为 1024。            101 如果 RTCLKS = x0 , 它为 16 ; 如果 RTCLKS = x1 , 它为 2048。            110 如果 RTCLKS = x0 , 它为 32 ; 如果 RTCLKS = x1 , 它为 100。            111 如果 RTCLKS = x0 , 它为 64 ; 如果 RTCLKS = x1 , 它为 1000。</p>
7 RTIF	<p>实时中断标志</p> <p>该状态位指示 RTC 计数器寄存器已达到 RTC 模数寄存器中的值。写入逻辑 0 无效。写入逻辑 1 会将该位清零并清除实时中断请求。复位会将 RTIF 清除为 0。</p> <p>0 RTC 计数器未达到 RTC 模数寄存器中的值。            1 RTC 计数器已达到 RTC 模数寄存器中的值。</p>
6 RTIE	<p>实时中断使能</p> <p>该读/写位使能实时中断。如果 RTIE 置位，那么在 RTIF 置位时会生成中断。复位会将 RTIE 清除为 0。</p> <p>0 实时中断请求禁用。使用软件轮询。            1 实时中断请求使能。</p>
5 保留	<p>此字段为保留字段。</p> <p>此只读字段为保留字段且值始终为 0。</p>
4 RTCO	<p>实时计数器输出</p> <p>该读/写位使能实时计数器把切换输出到引脚上。如果该位置位，那么在 RTC 计数器溢出时，将切换 RTCO 至引脚。</p> <p>0 实时计数器输出禁用。            1 实时计数器输出使能。</p>
保留	<p>此字段为保留字段。</p> <p>此只读字段为保留字段且值始终为 0。</p>

## 29.4.2 RTC 模数寄存器 (RTC\_MOD)

RTC\_MOD 指示 16 位模数值。

地址: 4003\_D000h 基准 + 4h 偏移 = 4003\_D004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R																0																		
W																																		
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

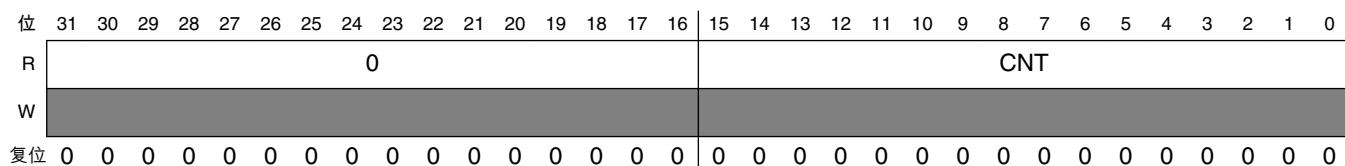
## RTC MOD 字段描述

字段	描述
31–16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
MOD	RTC 模数  该读/写字段包含模数值，在比较结果匹配时将计数值复位到 0x0000，以及置位 SC[RTIF]状态字段。值 0x0000 使 SC[RTIF]在预分频器输出的每个上升沿都置位。复位会将该模数设置为 0x0000。

### 29.4.3 RTC 计数器寄存器 (RTC\_CNT)

RTC CNT 指示 16 位计数器的当前 RTC 计数的只读值。

地址: 4003 D000h 基准 + 8h 偏移 = 4003 D008h



## RTC CNT 字段描述

字段	描述
31–16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
CNT	RTC 计数  该只读字段包含 16 位计数器的当前值，先读取 CNT[7:0]，再读取 CNT[15:8]。写操作对该寄存器无效。复位或将其他值写入 SC[RTCLKS]和 SC[RTCPMS]会将该计数清除为 0x0000。

## 29.5 功能说明

RTC 由一个主 16 位上加计数器并带有 16 位模数寄存器、一个时钟源选择器以及一个预分频器模块（可选择基于二进制或基于十进制的值）组成。该模块还包含用于引脚分配的软件可选中断逻辑和切换逻辑。

MCU 复位后，计数器停止并复位到 0x0000，模数寄存器设置到 0x0000，预分频器关闭。外部振荡器时钟被选作默认时钟源。要启动预分频器，需将非 0 值写入预分频器选择字段(RTC\_SC[RTCPS])。

时钟源可通过软件选择：外部振荡器(OSC)、片上低功耗振荡器(LPO)、32-kHz 内部基准时钟和总线时钟。RTC 时钟选择字段( RTC\_SC[RTCLKS] )用于选择预分频器所需的时钟源。如果将不同的值写入 RTC\_SC[RTCLKS]，预分频器和 CNT 计数器会复位到 0x00。

RTC\_SC[RTCP]和 RTC\_SC[RTCLKS]选择所需的分频值。如果将不同的值写入 RTC\_SC[RTCP]，预分频器和 RTCCNT 计数器会复位到 0x00。下表列出了不同的预分频器周期值。

表 29-1. 预分频器周期

RTCP	32768Hz OSC 时钟源 预分频器周期(RTCLKS = 00)	LPO 时钟(1 kHz)源预 分频器周期(RTCLKS = 01)	内部基准时钟(32.768 kHz)源预分频器周期 (RTCLKS = 10)	总线时钟(8 MHz)源预 分频器周期(RTCLKS = 11)
000	关闭	关闭	关闭	关闭
001	30.5176 μs	128 ms	30.5176 μs	16 μs
010	61.0351 μs	256 ms	61.0351 μs	32 μs
011	122.0703 μs	512 ms	122.0703 μs	64 μs
100	244.1406 μs	1024 ms	244.1406 μs	128 μs
101	488.28125 μs	2048 ms	488.28125 μs	256 μs
110	976.5625 μs	100 ms	976.5625 μs	12.5 μs
111	1.9531 ms	1 s	1.9531 μs	125 μs

RTC 模数寄存器(RTC\_MOD)允许将比较值设置为 0x0000 到 0xFFFF 之间的任意值。计数器处于有效状态时，会以所选比率递增，直到计数值与模数值匹配为止。当这些值匹配时，计数器复位到 0x0000 并继续计数。只要出现匹配情况，实时中断标志(RTC\_SC[RTIF])就会置位。该标志在模数值变为 0x0000 时置位。写入 RTC\_MOD 的模数值处于锁存状态，直到 RTC 计数器溢出或选择的 RTC\_SC[RTCP] 为非 0 值。

只要 RTC\_SC[RTIF] 置位，RTC 便允许生成一个中断。要使能实时中断，需将实时中断使能字段(RTC\_SC[RTIE])置位。将 1 写入 RTC\_SC[RTIF] 可清零 RTC\_SC[RTIF]。

通过电平翻转，RTC 还允许输出到外部管脚。必须置位 RTC\_SC[RTCO] 以使能外部管脚翻转。该功能有效时，电平取决于计数器溢出时管脚的前一状态。

## 29.5.1 RTC 操作示例

本节举例说明了计数器达到模数寄存器中匹配值时的 RTC 操作。

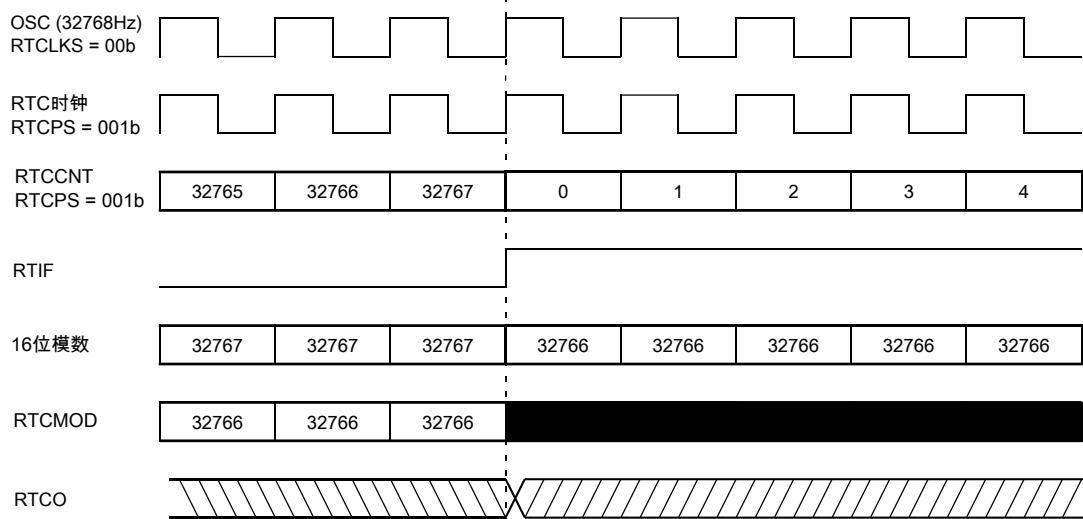


图 29-2. RTC 计数器溢出示例

在上例中,选择的是外部时钟源。预分频器设置为  $\text{RTC\_SC[RTCPS]} = 001\text{b}$  或直通。当  $\text{RTC\_MOD}$  寄存器中的模数值设置为 32766 时, 16 位比较器使用的实际模数值是 32767。当计数器  $\text{RTC\_CNT}$  达到模数值 32767 时, 计数器溢出到 0x00 并继续计数。模数值通过提取  $\text{RTC\_MOD}$  寄存器中的值来更新。当计数器值从 0x7FFF 变为 0x0000 时, 实时中断标志  $\text{RTC\_SC[RTIF]}$  置位。 $\text{RTC\_SC[RTCO]}$  在  $\text{RTC\_SC[RTIF]}$  置位时也翻转。

## 29.6 初始化/应用信息

本节给出代码示例旨在就如何进行 RTC 模块的初始化和配置向用户提供基本指引。软件示例采用 C 语言实现。

下述示例显示使用 OSC 时钟源实现可能的最低功耗时, 如何用 RTC 实现计时。

### 示例: 29.6.1 RTC ISR 中实现的软件日历

```
/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;

/* Configure RTC to interrupt every 1 second from OSC (32.768KHz) clock source */
RTC_MOD = 511; // overflow every 32 times
RTC_SC = RTC_SC_RTCPS_MASK; // external 32768 clock selected with 1/64 predivider.
RTC_SC = RTC_SC_RTIF_MASK | RTC_SC_RTIE_MASK; // interrupt cleared and enabled

*****
Function Name : RTC_ISR
Notes : Interrupt service routine for RTC module.
*****
```

```
void RTC_ISR(void)
{
/* Clears the interrupt flag, RTIF, and interrupt request */
RTC_SC |= RTC_SC_RTIF_MASK;

/* RTC interrupts every 1 Second */
Seconds++;

/* 60 seconds in a minute */
if (Seconds > 59)
{
Minutes++;
Seconds = 0;
}

/* 60 minutes in an hour */
if (Minutes > 59)
{
Hours++;
Minutes = 0;
}

/* 24 hours in a day */
if (Hours > 23)
{
Days++;
Hours = 0;
}
```

# 第 30 章 串行外设接口(SPI)

## 30.1 简介

### 注

关于与具体芯片相关的本模块详细操作示例，请参见芯片配置信息。

串行外设接口(SPI)模块提供 MCU 与外设之间的全双工同步串行通信。外设可以是其他微控制器、模数转换器、移位寄存器、传感器、存储器等。

在主机模式下，SPI 的波特率最高等于总线时钟除以 2；在从机模式下，SPI 的波特率最高等于总线时钟除以 4。软件可对状态标志轮询，或对 SPI 操作进行中断驱动。

### 注

有关实际最大的 SPI 波特率，请参见芯片配置详细信息和器件数据手册。

SPI 还并集成了用于接收数据缓冲区的硬件匹配特性。

### 30.1.1 特性

SPI 包括这些特性：

- 主机模式或从机模式工作
- 全双工或单线双向模式
- 可编程发送比特率
- 双缓冲发送和接收数据寄存器
- 串行时钟相位和极性选择

- 从机选择输出
- 带 CPU 中断功能的模式错误标志位
- Wait 模式期间的 SPI 操作控制
- 可供选择的 MSB 优先或 LSB 优先移位
- 接收数据缓冲器硬件匹配特性

### 30.1.2 工作模式

SPI 有如下三种工作模式。

- Run 模式

这是基本工作模式。

- Wait 模式

SPIWait 模式是可配置的低功耗模式,由 SPIx\_C2 寄存器中的 SPISWAI 位控制。在 Wait 模式下,如果 C2[SPISWAI]被清零,则 SPI 如同在 Run 模式下一样工作。如果 C2[SPISWAI]置位,则 SPI 进入降低功耗状态,SPI 时钟发生器关闭。如果 SPI 配置为主机,任何正在进行的发送都会停止,不过在 CPU 进入 Run 模式后会恢复。如果 SPI 配置为从机,字节的接收和发送会继续进行,因而从机仍然与主机同步。

- Stop 模式

为了降低功耗,SPI 在 Stop 模式(外设总线时钟停止但内部逻辑状态仍然保留)下无效。如果 SPI 配置为主机,任何正在进行的发送都会停止,不过在 CPU 进入 Run 模式后会恢复。如果 SPI 配置为从机,数据的接收和发送会继续进行,因而从机仍然与主机同步。

SPI 在 Stop 模式(外设总线时钟停止且不保留内部逻辑状态)下完全禁用。当 CPU 从这些 Stop 模式中唤醒时,所有 SPI 寄存器内容都会复位。

有关工作模式的详细说明,请参见[低功耗模式选项](#)。

### 30.1.3 结构框图

本节的结构框图表示了 SPI 系统连接、SPI 模块的内部结构以及控制主机模式比特率的 SPI 时钟分频器。

### 30.1.3.1 SPI 系统结构框图

下图表示的是采用主从配置进行连接的两个 MCU 的 SPI 模块。主机发起所有 SPI 数据传输。传输期间，主机将数据（通过 MOSI 引脚）移出到从机，同时将数据从从机移入（通过 MISO 引脚）。传输有效地交换两个 SPI 系统的 SPI 移位寄存器中的数据。SPSCK 信号是主机的时钟输出和从机的输入。从机必须通过用低电平信号对从机选择输入（SS 引脚）进行选择。该系统中，主机将其 SS 引脚配置为可选的从机选择输出。

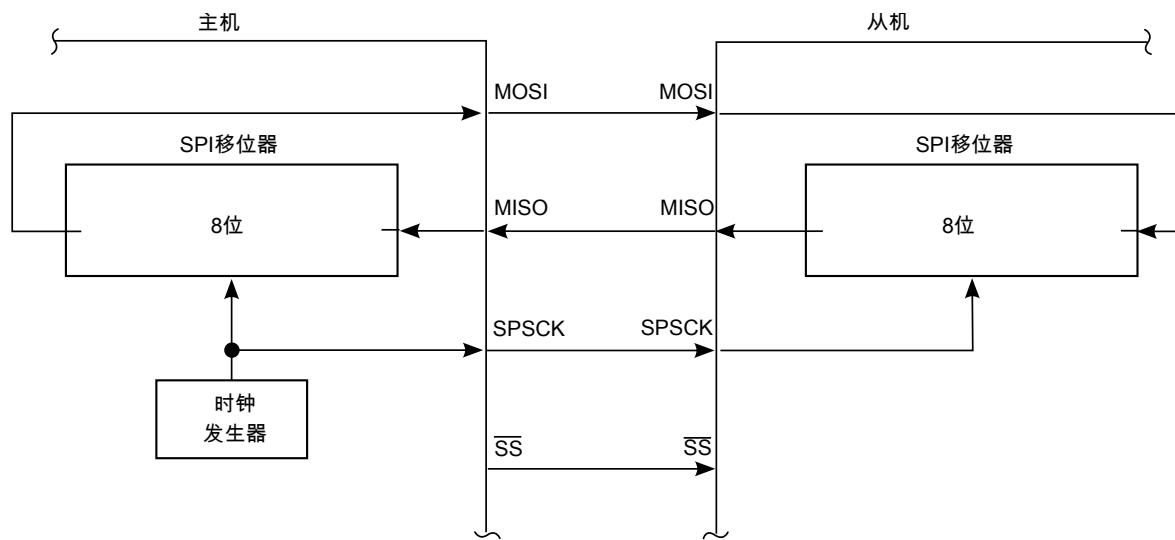


图 30-1. SPI 系统连接

### 30.1.3.2 SPI 模块结构框图

下图是 SPI 模块的结构框图。SPI 模块的核心部分是 SPI 移位寄存器。数据被写入具有双缓冲区的发送器（写入 SPIx\_D）并在数据传输开始时被转移到 SPI 移位寄存器。移入 8 位数据后，数据被转移到可从 SPIx\_D 读取数据的具有双缓冲区的接收器。通过引脚多路复用逻辑控制 MCU 引脚与 SPI 模块之间的连接。

当 SPI 配置为主机时，时钟输出连接到 SPSCK 引脚，移位器输出连接到 MOSI，移位器输入从 MISO 引脚接入。

当 SPI 配置为从机时，SPSCK 引脚连接到 SPI 的时钟输入，移位器输出连接到 MISO，移位器输入从 MOSI 引脚接入。

在外部 SPI 系统中，只需将所有 SPSCK 引脚彼此相连，将所有 MISO 引脚连在一起，并将所有 MOSI 引脚连在一起。对于这些引脚，外设经常使用略有不同的名称。

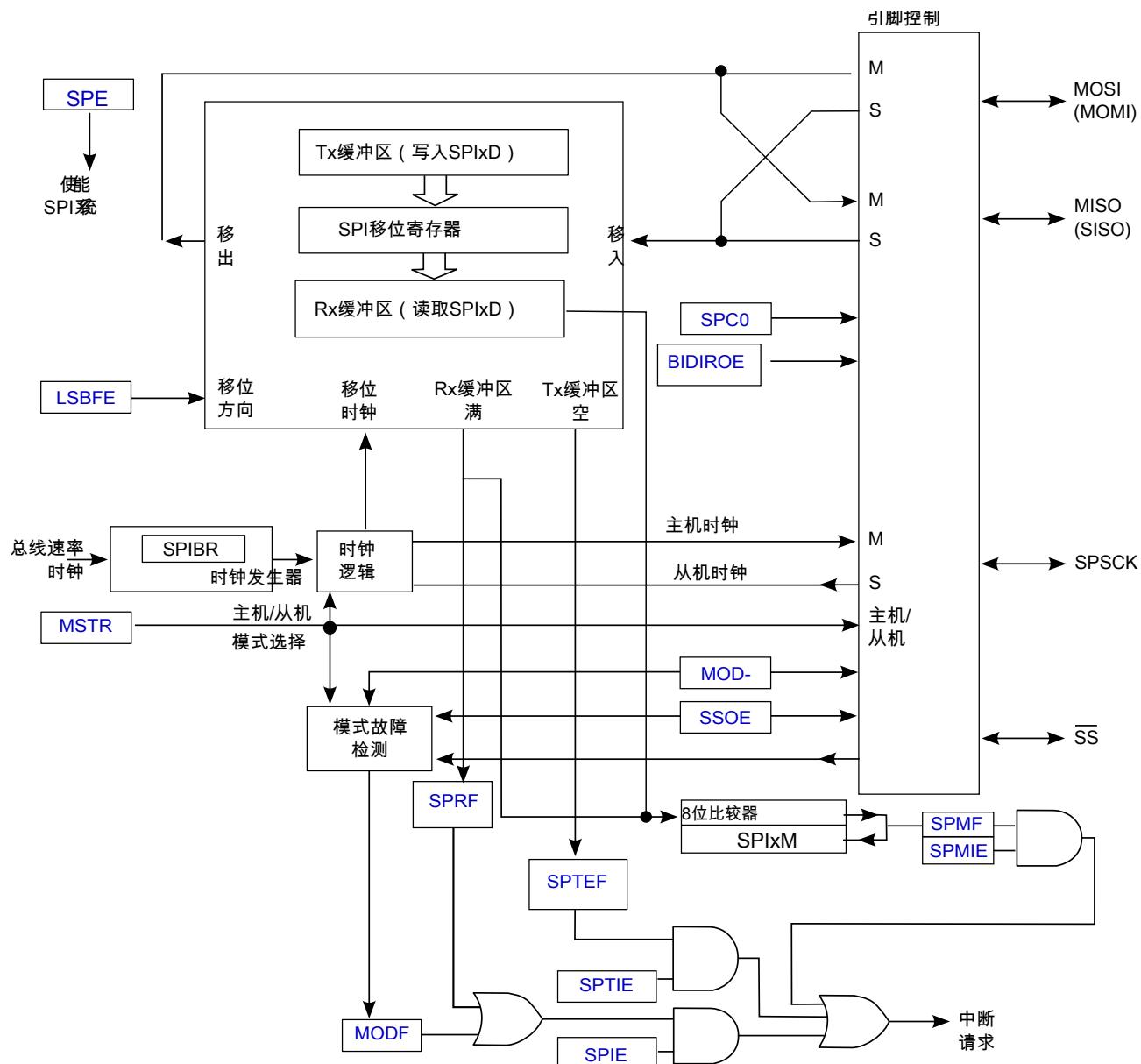


图 30-2. 无 FIFO 的 SPI 模块功能框图

## 30.2 外部信号说明

SPI 可视情况共用四个端口引脚。这些引脚的功能取决于 SPI 控制位的置位。SPI 禁用(SPE = 0)时，这四个引脚恢复到不受 SPI 控制的其他功能（基于芯片配置）。

### 30.2.1 SPSCK — SPI 串行时钟

当 SPI 使能为从机时，此引脚为串行时钟输入。当 SPI 使能为主机时，此引脚为串行时钟输出。

### 30.2.2 MOSI — 主机数据输出, 从机数据输入

当 SPI 被使能为主机且 SPI 引脚控制零(SPC0)为 0 时 (非双向模式)，此引脚为串行数据输出。当 SPI 被使能为从机且 SPC0 为 0 时，此引脚为串行数据输入。如果 SPC0 为 1，选择单线双向模式和主机模式，此引脚将成为双向数据 I/O 引脚(MOMI)。同时，双向模式输出使能位确定引脚用作输入 (BIDIROE 是 0) 还是用作输出 (BIDIROE 是 1)。如果 SPC0 是 1 且已选择从机模式，则 SPI 将不能使用此引脚，此引脚将恢复至其他功能 (基于芯片配置)。

### 30.2.3 MISO — 主机数据输入, 从机数据输出

当 SPI 被使能为主机且 SPI 引脚控制零(SPC0)为 0 时 (非双向模式)，此引脚为串行数据输入。当 SPI 被使能为从机且 SPC0 为 0 时，此引脚为串行数据输出。如果 SPC0 为 1，选择单线双向模式和从机模式，此引脚将变成双向数据 I/O 引脚(SISO)，而双向模式输出使能位将确定引脚用作输入 (BIDIROE 为 0) 还是用作输出 (BIDIROE 为 1)。如果 SPC0 为 1 且已选择主机模式，则 SPI 将不能使用此引脚，此引脚将恢复至其他功能 (基于芯片配置)。

### 30.2.4 SS — 从机选择

当 SPI 被使能为从机时，此引脚为低电平有效的从机选择输入。当 SPI 被使能为主机且模式故障使能关闭 (MODFEN 为 0) 时，SPI 将不能使用此引脚，而引脚将恢复至其他功能 (基于芯片配置)。当 SPI 被使能为主机且 MODFEN 为 1 时，从机选择输出使能位能够决定此引脚是用作模式故障输入 (SSOE 为 0) 或是用作从机选择输出 (SSOE 为 1)。

## 30.3 存储器映像/寄存器定义

SPI 具有可选择 SPI 选项、控制波特率、报告 SPI 状态、保持 SPI 数据匹配值以及用于发送/接收数据的 8 位寄存器。

## SPI 存储器映射

地址偏移量 (十六进制)	绝对地址(十 六进制)	寄存器名称	宽度(单 位: 位)	访问	复位值	小节/页
0	4007_6000	SPI 控制寄存器 1 (SPI0_C1)	8	R/W	04h	30.3.1/498
1	4007_6001	SPI 控制寄存器 2 (SPI0_C2)	8	R/W	00h	30.3.2/500
2	4007_6002	SPI 波特率寄存器 (SPI0_BR)	8	R/W	00h	30.3.3/501
3	4007_6003	SPI 状态寄存器 (SPI0_S)	8	R	20h	30.3.4/502
5	4007_6005	SPI 数据寄存器 (SPI0_D)	8	R/W	00h	30.3.5/503
7	4007_6007	SPI 匹配寄存器 (SPI0_M)	8	R/W	00h	30.3.6/504
0	4007_7000	SPI 控制寄存器 1 (SPI1_C1)	8	R/W	04h	30.3.1/498
1	4007_7001	SPI 控制寄存器 2 (SPI1_C2)	8	R/W	00h	30.3.2/500
2	4007_7002	SPI 波特率寄存器 (SPI1_BR)	8	R/W	00h	30.3.3/501
3	4007_7003	SPI 状态寄存器 (SPI1_S)	8	R	20h	30.3.4/502
5	4007_7005	SPI 数据寄存器 (SPI1_D)	8	R/W	00h	30.3.5/503
7	4007_7007	SPI 匹配寄存器 (SPI1_M)	8	R/W	00h	30.3.6/504

### 30.3.1 SPI 控制寄存器 1 (SPIx\_C1)

该读/写寄存器包括 SPI 使能控制、中断使能和配置选项。

地址: 4007\_6000h 基准 + 0h 偏移 = 4007\_6000h

位 读 写	7	6	5	4	3	2	1	0
	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
复位	0	0	0	0	0	1	0	0

#### SPI0\_C1 字段描述

字段	描述
7 SPIE	SPI 中断使能：用于 SPRF 和 MODF 使能用于 SPI 接收缓冲区满(SPRF)和模式故障(MODF)事件的中断。 0 来自 SPRF 和 MODF 的中断被禁止——使用轮询 1 SPRF 或 MODF 为 1 时请求硬件中断
6 SPE	SPI 系统使能 使能 SPI 系统并将 SPI 端口引脚专门用于 SPI 系统功能。如果 SPE 清零，则 SPI 禁用且强制进入空闲状态，S 寄存器中的所有状态位都复位。 0 SPI 系统不工作 1 SPI 系统使能
5 SPTIE	SPI 发送中断使能 这是 SPI 发送缓冲区空(SPTEF)的中断使能位。中断在 SPI 发送缓冲区为空 ( SPTEF 置位 ) 时发生。

下一页继续介绍此表...

**SPI0\_C1 字段描述 (继续)**

字段	描述
	<p>0 来自 SPTEF 的中断被禁止 ( 使用轮询 )      1 SPTEF 为 1 时 , 请求硬件中断。</p>
4 MSTR	<p>主机/从机模式选择      选择主机或从机模式操作。</p> <p>0 SPI 模块配置为 SPI 从机      1 SPI 模块配置为 SPI 主机</p>
3 CPOL	<p>时钟极性      选择反相或非反相 SPI 时钟。要在 SPI 模块之间发送数据 , SPI 模块必须具有相同的 CPOL 值。      该位有效地将一个反相器与来自 SPI 主机或传输至 SPI 从机的时钟信号串联。有关详细信息 , 请参见“SPI 时钟格式”的说明。</p> <p>0 高电平有效 SPI 时钟 ( 空闲时为低电平 )      1 低电平有效 SPI 时钟 ( 空闲时为高电平 )</p>
2 CPHA	<p>时钟相位      针对不同种类的同步串行外设选择两种时钟格式中的一种。有关详细信息 , 请参见“SPI 时钟格式”的说明。</p> <p>0 SPSCK 上的第一个边沿出现在数据传输的第一个周期的中间。      1 SPSCK 上的第一个边沿出现在数据传输的第一个周期的开始。</p>
1 SSOE	<p>从机选择输出使能      该位与 C2 寄存器中的模式故障使能(MODFEN)字段和主机/从机(MSTR)控制位共同确定 SS 引脚的功能。</p> <p>0 C2[MODFEN]为 0 时 : 在主机模式下 , SS 引脚功能为通用 I/O ( 非 SPI )。在从机模式下 , SS 引脚功能为从机选择输入。      C2[MODFEN]为 1 时 : 在主机模式下 , SS 引脚功能为模式故障的 SS 输入。在从机模式下 , SS 引脚功能为从机选择输入。</p> <p>1 C2[MODFEN]为 0 时 : 在主机模式下 , SS 引脚功能为通用 I/O ( 非 SPI )。在从机模式下 , SS 引脚功能为从机选择输入。      C2[MODFEN]为 1 时 : 在主机模式下 , SS 引脚功能为自动 SS 输出。在从机模式下 , SS 引脚功能为从机选择输入。</p>
0 LSBFE	<p>LSB 优先 ( 移位器方向 )      该位不影响数据寄存器中 MSB 和 LSB 的位置。在对数据寄存器的读写操作中 , MSB 始终是位 7。</p> <p>0 SPI 串行数据传输从最高有效位开始。      1 SPI 串行数据传输从最低有效位开始。</p>

### 30.3.2 SPI 控制寄存器 2 (SPIx\_C2)

该读/写寄存器用于控制 SPI 系统的可选特性。位 6 不实施，且始终读取为 0。

地址: 4007\_6000h 基准 + 1h 偏移 = 4007\_6001h

位 读 写	7	6	5	4		3	2	1	0
复位	SPMIE	保留	保留	MODFEN	BIDIROE	保留	SPISWAI	SPC0	0
	0	0	0	0	0	0	0	0	0

**SPI0\_C2 字段描述**

字段	描述
7 SPMIE	SPI 匹配中断使能 该位是 SPI 接收数据缓冲区硬件匹配(SPMF)功能的中断使能位。 0 来自 SPMF 的中断被禁止 ( 使用轮询 ) 1 SPMF 为 1 时 , 请求硬件中断。
6 Reserved	此字段为保留字段。 不要对该保留位进行写操作。
5 Reserved	此字段为保留字段。 不要对该保留位进行写操作。
4 MODFEN	主机模式故障功能使能 SPI 针对从机模式进行配置时 , 该位无意义或无效。( SS 引脚是从机选择输入。 ) 在主机模式下 , 该位决定如何使用 SS 引脚。有关详细信息 , 请参见 C1 寄存器中 SSOE 位的说明。 0 模式故障功能禁用 , 主机 SS 引脚恢复为不受 SPI 控制的通用 I/O 1 模式故障功能使能 , 主机 SS 引脚用作模式故障输入或从机选择输出
3 BIDIROE	双向模式输出使能 双向模式因 SPI 引脚控制 0 (SPC0)置 1 而使能时 , BIDIROE 决定是否对单一双向 SPI I/O 引脚的 SPI 数据输出驱动器使能。当 SPI 配置为主机或从机时 , 它分别使用 MOSI (MOMI) 或 MISO (SISO) 引脚作为单一 SPI 数据 I/O 引脚。SPC0 为 0 时 , BIDIROE 无意义或无效。 0 输出驱动器禁用 , SPI 数据 I/O 引脚用作输入 1 SPI I/O 引脚使能 , 用作输出
2 Reserved	此字段为保留字段。 不要对该保留位进行写操作。
1 SPISWAI	SPI 在 Wait 模式下停止 该位在器件处于 Wait 模式时用于省电。 0 SPI 时钟在 Wait 模式下继续工作。 1 SPI 时钟在 MCU 进入 Wait 模式时停止工作。
0 SPC0	SPI 引脚控制 0 启用双向引脚配置。

下一页继续介绍此表...

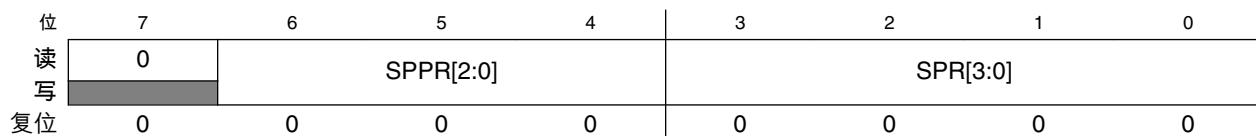
### SPI0\_C2 字段描述 (继续)

字段	描述
	<p>0 SPI 针对数据输入和数据输出使用独立的引脚 ( 引脚模式为正常 )。</p> <p>在主机工作模式下 : MISO 为主机输入 , MOSI 为主机输出。</p> <p>在从机工作模式下 : MISO 为从机输出 , MOSI 为从机输入。</p> <p>1 SPI 配置为单线双向操作 ( 引脚模式为双向 )。</p> <p>在主机工作模式下 : SPI 不使用 MISO ; MOSI 为主机输入 ( BIDIROE 为 0 时 ) 或主机 I/O ( BIDIROE 为 1 时 )。</p> <p>在从机工作模式下 : MISO 为从机输入 ( BIDIROE 为 0 时 ) 或从机 I/O ( BIDIROE 为 1 时 ); SPI 不使用 MOSI。</p>

### 30.3.3 SPI 波特率寄存器 (SPIx\_BR)

使用该寄存器设置 SPI 主机的预分频器和比特率分频因子。随时可对该寄存器进行读写操作。

地址: 4007\_6000h 基准 + 2h 偏移 = 4007\_6002h



### SPI0\_BR 字段描述

字段	描述
7 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
6–4 SPPR[2:0]	<p>SPI 波特率预分频因子</p> <p>该 3 位字段选择 SPI 波特率预分频器 8 个分频因子中的一个。该预分频器的输入为总线速率时钟 (BUSCLK)。该预分频器的输出驱动 SPI 波特率分频器的输入。有关详细信息 , 请参见“SPI 波特率生成”的说明。</p> <p>000 波特率预分频器分频因子为 1。</p> <p>001 波特率预分频器分频因子为 2。</p> <p>010 波特率预分频器分频因子为 3。</p> <p>011 波特率预分频器分频因子为 4。</p> <p>100 波特率预分频器分频因子为 5。</p> <p>101 波特率预分频器分频因子为 6。</p> <p>110 波特率预分频器分频因子为 7。</p> <p>111 波特率预分频器分频因子为 8。</p>
SPR[3:0]	<p>SPI 波特率分频因子</p> <p>该 4 位字段选择 SPI 波特率分频器 9 个分频因子中的一个。该分频器的输入来自 SPI 波特率预分频器。有关详细信息 , 请参见“SPI 波特率生成”的说明。</p>

下一页继续介绍此表...

**SPI0\_BR 字段描述 (继续)**

字段	描述							
	0000	波特率分频因子为 2。						
	0001	波特率分频因子为 4。						
	0010	波特率分频因子为 8。						
	0011	波特率分频因子为 16。						
	0100	波特率分频因子为 32。						
	0101	波特率分频因子为 64。						
	0110	波特率分频因子为 128。						
	0111	波特率分频因子为 256。						
	1000	波特率分频因子为 512。						
	所有其他值 保留							

**30.3.4 SPI 状态寄存器 (SPIx\_S)**

该寄存器包含只读状态位。写操作无意义或无效。

**注**

位 3 至位 0 不执行且始终读为 0。

地址: 4007\_6000h 基准 + 3h 偏移 = 4007\_6003h

位	7	6	5	4	3	2	1	0
读	SPRF	SPMF	SPTEF	MODF				0
写								
复位	0	0	1	0	0	0	0	0

**SPI0\_S 字段描述**

字段	描述
7 SPRF	SPI 读取缓冲区满标志  SPRF 在 SPI 传输完成时置位 , 指示可以从 SPI 数据(D)寄存器读取接收到的数据。SPRF 的方法是在其置位时对 SPRF 进行读操作 , 然后对 SPI 数据寄存器进行读操作。  0 在接收数据缓冲区中无数据 1 在接收数据缓冲区中有数据
6 SPMF	SPI 匹配标志  接收数据缓冲区中的值与 M 寄存中的值匹配时 , SPRF 为 1 , SPMF 随后置位。要将该标志清零 , 在 SPMF 置位时对其进行读操作 , 然后写入 1 。  0 接收数据缓冲区中的值与 M 寄存中的值不匹配 1 接收数据缓冲区中的值与 M 寄存中的值匹配
5 SPTEF	SPI 发送缓冲区空标志  该位在发送数据缓冲区为空时置位。清零 SPTEF 的方法是在 SPTEF 置位时对 S 寄存器进行读操作 , 然后将某个数据值写入 D 处的发送缓冲区。在向 D 寄存器中写入数据之前 , 必须在 SPTEF 被置 1 时读取 S 寄存器 ; 否则 , 将忽略 D 写入。SPTEF 在发送缓冲区中的所有数据都搬运到发送移位寄存器时自动置位。对于

下一页继续介绍此表...

**SPI0\_S 字段描述 (继续)**

字段	描述
	<p>空闲 SPI，写入 D 的数据几乎是立即传输到移位器，SPTEF 在两个总线周期内置位，从而使第二组数据可以排队进入发送缓冲区。移位寄存器中的数据传输完成后，发送缓冲区中的排队数据自动移入移位器，SPTEF 置位表示发送缓冲区中有空间来容纳新数据。如果发送缓冲器中没有处于等待状态的新数据，SPTEF 仍保持置位状态，且缓冲器中没有数据移至移位器。</p> <p>如果传输不停止，则会再次发出上一个发送的数据。</p> <p>0 SPI 发送缓冲区非空 1 SPI 发送缓冲区空</p>
4 MODF	<p>主机模式错误标志</p> <p>MODF 在 SPI 配置为主机且从机选择输入变为低电平时置位，表示另一个 SPI 器件也配置为主机。<math>\overline{SS}</math> 引脚仅在 C1[MSTR]为 1、C2[MODFEN]为 1 且 C1[SSOE]为 0 时用作模式故障错误输入；其他情况下，MODF 绝不会置位。清零 MODF 的方法是在 MODF 为 1 时对其进行读操作，然后对 SPI 控制寄存器 1 (C1)进行写操作。</p> <p>0 无模式故障错误 1 检测到模式故障错误</p>
保留	<p>此字段为保留字段。</p> <p>此只读字段为保留字段且值始终为 0。</p>

**30.3.5 SPI 数据寄存器 (SPIx\_D)**

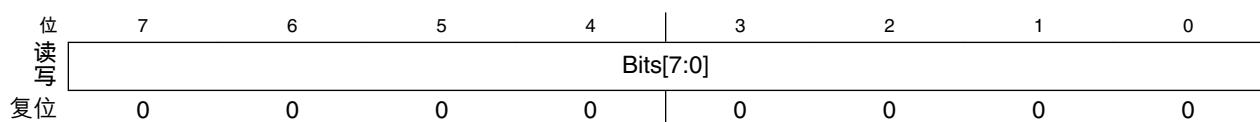
该寄存器一起构成 SPI 数据的输入和输出寄存器。写入这些寄存的结果是写入发送数据缓冲区，以便让数据排队并发送。

SPI 配置为主机时，发送数据缓冲区中排队的数据在前一发送完成后会被立即发送。

S 寄存器中的 SPTEF 位指示发送数据缓冲区准备接纳新数据的时间。必须在数据写入 SPI 数据寄存之前且 S[SPTEF]置位的情况下对 S 寄存器进行读操作，否则写操作会被忽略。发送 DMA 请求使能 (TXDMAE 为 1) 的情况下：如果 S[SPTEF]置位，DMA 无需先对 S 寄存器进行读操作便可自动对 SPI 数据寄存器进行写操作。

在 S[SPRF]置位后以及另一个传输完成前的任何时候，都可以从 SPI 数据寄存读取数据。如果在新传输结束前未能从接收数据缓冲区将数据读出，则会引起接收溢出状况，并且新传输的数据会丢失。新数据丢失的原因是接收缓冲区仍保持前一字符，未准备好接纳新数据。接收溢出状况没有相应的指示，因此用户必须确保先从接收缓冲区读取前一数据，然后再启动新的传输。

地址: 4007\_6000h 基准 + 5h 偏移 = 4007\_6005h



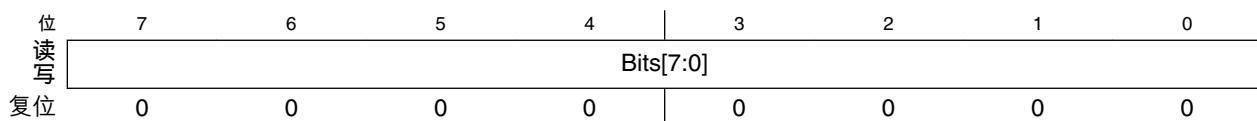
**SPI0\_D 字段描述**

字段	描述
Bits[7:0]	数据 ( 低字节 )

**30.3.6 SPI 匹配寄存器 (SPIx\_M)**

该寄存器都含有硬件比较值。当 SPI 接收数据缓冲区中接收到的值等于该硬件比较值时，S 寄存器中的 SPI 匹配标志(S[SPMF])置位。

地址: 4007\_6000h 基准 + 7h 偏移 = 4007\_6007h

**SPI0\_M 字段描述**

字段	描述
Bits[7:0]	硬件比较值 ( 低字节 )

**30.4 功能说明**

本节说明该模块的功能。

**30.4.1 综述**

使能 SPI 系统的方法是置位 SPI 控制寄存器 1 中的 SPI 使能(SPE)位。C1[SPE]置位时，四个相关的 SPI 端口引脚专门用于 SPI 功能：

- 从机选择 (SS)
- 串行时钟 (SPSCK)
- 主机输出/从机输入 (MOSI)
- 主机输入/从机输出 (MISO)

在 SPI 主机中发起 SPI 传输的方法是：先在 S[SPTEF] = 1 时对 SPI 状态寄存器 (SPIx\_S) 进行读操作，然后将数据写入发送数据缓冲区（写入 SPIxD）。传输完成时，接收到的数据移入接收数据缓冲区。SPIxD 寄存器用作 SPI 接收数据缓冲区（读操作）和 SPI 发送数据缓冲区（写操作）。

通过 SPI 控制寄存器 1 (SPIx\_C1) 中的时钟相位控制(CPHA)位和时钟极性控制(CPOL)位选择四种可能的时钟格式中的一种提供给 SPI 系统。CPOL 位只选择非反相或反相时钟。C1[CPHA] 用于支持两种截然不同的协议，其通过在奇数 SPSCK 边沿或偶数 SPSCK 边沿上进行数据采样而实现。

SPI 可配置为以主机或从机进行工作。SPI 控制寄存器 1 中的 MSTR 位置位时，选择主机模式；C1[MSTR] 被清零时，选择从机模式。

### 30.4.2 主机模式

SPI 在 C1[MSTR] 置位时以主机模式工作。只有主机 SPI 模块能发起发送。使发送开始的方法是：在 S[SPTEF] = 1 时先对 SPIx\_S 寄存器进行读操作，然后对主机 SPI 数据寄存器进行写操作。如果移位寄存器为空，字节会立即转移到移位寄存器。数据在串行时钟的控制下在 MOSI 引脚上移出。

- SPSCK
  - SPI 波特率寄存器中的波特率选择位 SPR3、SPR2、SPR1、SPR0 和波特率预选择位 SPPR2、SPPR1、SPPR0 共同控制波特率发生器并决定发送速度。SPSCK 引脚是 SPI 时钟输出。通过 SPSCK 引脚，主机的波特率发生器控制从机外设的移位寄存器。
- MOSI、MISO 引脚
  - 在主机模式下，串行数据输出引脚(MOSI)和串行数据输入引脚(MISO)的功能由 SPC0 和 BIDIROE 控制位确定。
- SS 引脚
  - 如果 C2[MODFEN] 和 C1[SSOE] 均置位，那么 SS 引脚配置为从机选择输出。SS 输出在每次发送期间变为低电平，在 SPI 处于空闲状态时变为高电平。如果 C2[MODFEN] 置位且 C1[SSOE] 被清零，则 SS 引脚配置为用于检测模式故障错误的输入。如果 SS 输入变为低电平，则表示存在模式故障错误（另一个主机试图驱动 MOSI 和 SPSCK 线路）。在该情况下，SPI 通过清零 C1[MSTR] 立即切换到从机模式，并且还禁用从机输出缓冲区 MISO（双向模式下则是 SISO）。结果，所有输出都禁用，SPSCK、MOSI 和 MISO 均为输入。如果在发生模式故障时有发送正在进行，那么该发送会中止并且 SPI 会强制进入空闲状态。该模式故障错误还会置位 SPI 状态寄存器 (SPIx\_S) 中的模式故障(MODF)标志。如果 S[MODF] 置位时，SPI 中断使能位(SPIE)

置位，则随后还会请求 SPI 中断序列。在主机刚开始时，SPI 数据寄存器进行写操作时，会有半个 SPSCK 周期的延迟。延迟之后，SPSCK 在主机中启动。传输操作的其余部分略有不同，具体取决于 SPI 控制寄存器 1 中 SPI 时钟相位位 CPHA 指定的时钟格式（参见 [SPI 时钟格式](#)）。

## 注

在主机模式下，更改 C1[CPOL]、C1[CPHA]、C1[SSOE]、C1[LSBFE]、C2[MODFEN]、C2[SPC0]、C2[BIDIROE]（C2[SPC0]置位时）、SPPR2-SPPR0 和 SPR3-SPR0 会中止正在进行的发送并强制使 SPI 进入空闲状态。远程从机无法检测出这一问题，因此主机必须确保远程从机被设置回闲置状态。

### 30.4.3 从机模式

SPI 在 SPI 控制寄存器 1 中的 MSTR 位被清零时以从机模式工作。

- SPSCK

在从机模式下，SPSCK 是来自主机的 SPI 时钟输入。

- MISO、MOSI 引脚

在从机模式下，串行数据输出引脚(MISO)和串行数据输入引脚(MOSI)的功能由 SPI 控制寄存器 2 中的 SPC0 位和 BIDIROE 位确定。

- SS 引脚

SS 引脚是从机选择输入。在数据发送开始之前，从机 SPI 的 SS 引脚必须为低电平。SS 在发送完成之前必须一直保持低电平。如果 SS 变为高电平，SPI 会强制进入空闲状态。

SS 输入还控制串行数据输出引脚。如果 SS 为高电平（未选中），则串行数据输出引脚为高阻抗。如果 SS 为低电平，则 SPI 数据寄存器中的首位会从串行数据输出引脚驱动输出。此外，如果未选中从机（SS 为高电平），那么就会忽略 SPSCK 输入并且 SPI 移位寄存器不会发生内部移位。

虽然 SPI 能够进行双工操作，但是部分 SPI 外围设备在从机模式中只能接收 SPI 数据。这些更简单的设备没有串行数据输出引脚。

## 注

使用具有双工能力的外设时，切勿同时使能如下的两个接收器：二者的串行输出驱动同一系统从机的串行数据输出线路。

只要不是一个以上的从机驱动系统从机的串行数据输出线路，多个从机从一个主机接收同一信息是可以的，不过主机不会从所有接收从机接收到返回信息。

如果 SPI 控制寄存器 1 中的 CPHA 位处于清零状态，那么 SPSCK 输入上的奇数边沿会使串行数据输入引脚处的数据进入锁存状态。偶数边沿使之前从串行数据输入引脚锁存的值移入 SPI 移位寄存器的 LSB 或 MSB，具体情况取决于 LSBFE 位。

如果 C1[CPHA]置位，那么 SPSCK 输入上的偶数边沿会使串行数据输入引脚上的数据进入锁存状态。奇数边沿使之前从串行数据输入引脚锁存的值移入 SPI 移位寄存器的 LSB 或 MSB，具体情况取决于 C1[LSBFE]。

如果 C1[CPHA]置位，第一个边沿用于将第一数据位移到串行数据输出引脚上。当 C1[CPHA]处于清零状态且 SS 输入为低电平（已选择从机）时，SPI 数据的首位从串行数据输出引脚驱动输出。移出第 8 位数据后，认为传输完成，接收数据转移到 SPI 数据寄存器。为了指示传输完成，SPI 状态寄存器中的 SPRF 标志置位。

## 注

在从机模式下，更改 C2[BIDIROE]（C2[SPC0]置位时）、C1[CPOL]、C1[CPHA]、C1[SSOE]、C1[LSBFE]、C2[MODFEN]和 C2[SPC0]位会破坏正在进行的发送，必须避免此类情况。

### 30.4.4 SPI 时钟格式

为了支持不同制造商的各种同步串行外设，SPI 系统的控制寄存器 1 有一个时钟极性(CPOL)位和一个时钟相位(CPHA)控制位，用于选择四种时钟格式中的一种进行数据传输。C1[CPOL]选择插入一个与时钟串联的反相器。C1[CPHA]选择时钟与数据之间的两种不同时钟相位关系中的一种。

下图所示为 CPHA = 1 时的时钟格式。图顶部显示的是供参考的 8 位时间，其中，位 1 开始于第一个 SPSCK 边沿，位 8 结束于第八个 SPSCK 边沿后的半个 SPSCK 周期。MSB 优先和 LSB 优先行表示 SPI 数据位的顺序，它取决于 LSBFE 的设置。图中显示了 SPSCK 极性的两种形态，但对于特定传输，只有一种波形适用，具体情况取决于 C1[CPOL]中的值。SAMPLE IN 波形施加于从机的 MOSI 输入或主机的 MISO 输入。MOSI 波形施加于主机的 MOSI 输出引脚，MISO 波形施加于从机的

MISO 输出。 $\overline{SS}$  OUT 波形施加于主机的从机选择输出（前提是 C2[MODFEN]和 C1[SSOE] = 1）。主机  $\overline{SS}$  输出会在传输开始前的半个 SPSCK 周期变为低电平，在传输的第八位时间结束时变回高电平。 $\overline{SS}$  IN 波形施加于从机的从机选择输入。

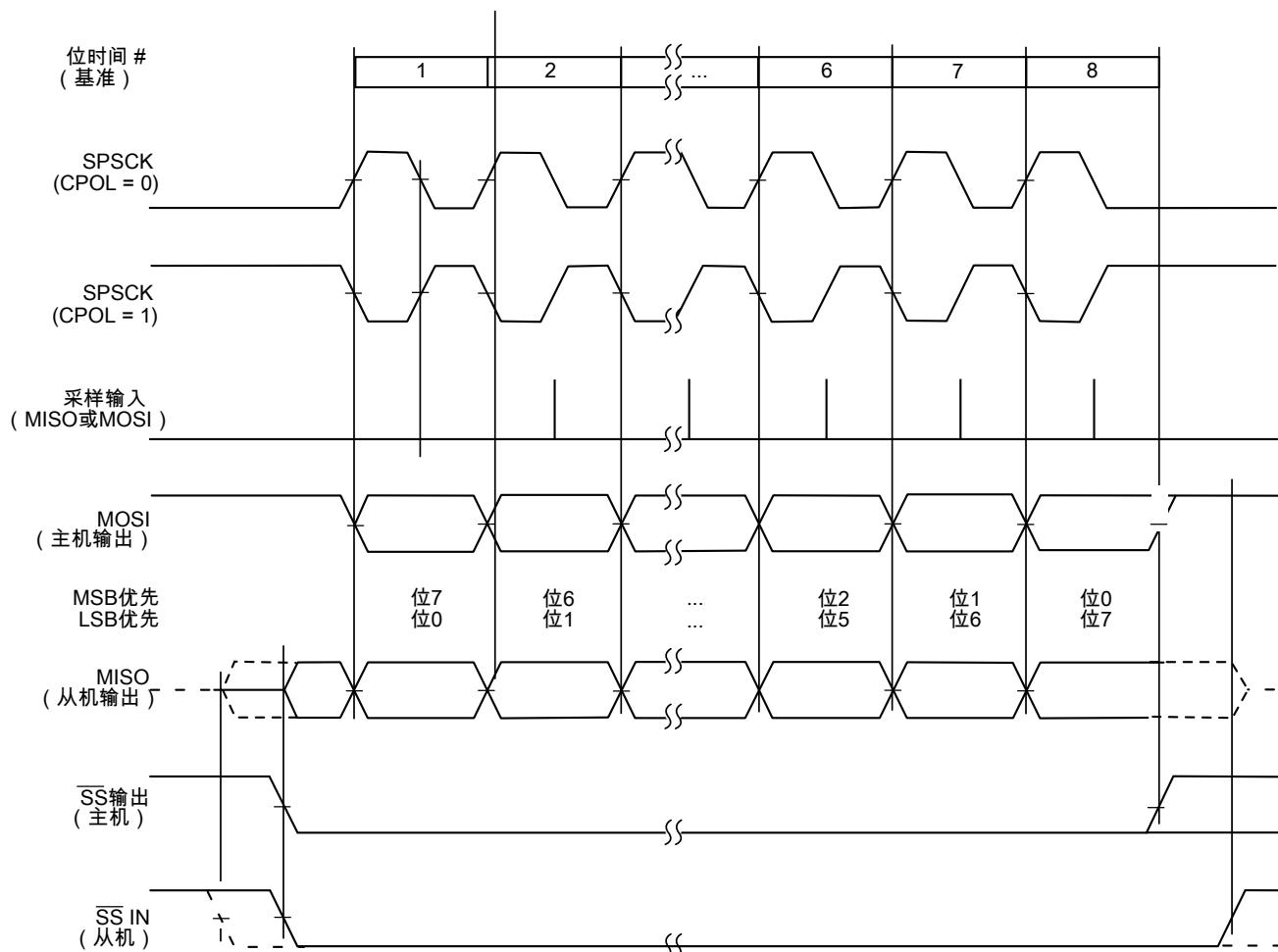


图 30-3. SPI 时钟格式(CPHA = 1)

当 C1[CPHA] = 1 时，从机在  $\overline{SS}$  变为低电平有效时开始驱动其 MISO 输出，但数据要等到第一个 SPSCK 边沿后才定义。第一个 SPSCK 边沿将数据的第一位从移位器移出到主机的 MOSI 输出和从机的 MISO 输出上。在下一个 SPSCK 边沿，主机和从机分别对 MISO 和 MOSI 输入上的数据位值进行采样。在第三个 SPSCK 边沿，SPI 移位器移动一个位位置，以便移入刚刚采样的位值，并将第二数据位值从移位器的另一端分别移出到主机的 MOSI 输出和从机的 MISO 输出。

当 C1[CPHA] = 1 时，从机的  $\overline{SS}$  输入在两次传输之间无需变为高电平无效状态。采用该时钟格式时，可能会发生背靠背发送，如下所述：

1. 一个发送正在进行。
2. 一个新数据字节在正在进行的发送完成之前被写入发送缓冲区。
3. 正在进行的发送完成后，新的已就绪数据会被立即发送。

在这两次连续发送之间，无需插入暂停； $\overline{SS}$  引脚保持低电平。

下图所示为 C1[CPHA] = 0 时的时钟格式。图顶部显示的是供参考的 8 位时间，其中，位 1 在从机被选中时 (SS 输入变为低电平) 开始，位 8 结束于最后一个 SPSCK 边沿。MSB 优先和 LSB 优先行显示 SPI 数据位的顺序，它取决于 LSBFE 的设置。图中显示了 SPSCK 极性的两种形态，但对于特定传输，只有一种波形适用，具体取决于 CPOL 中的值。SAMPLE IN 波形施加于从机的 MOSI 输入或主机的 MISO 输入。MOSI 波形施加于主机的 MOSI 输出引脚，MISO 波形施加于从机的 MISO 输出。SS OUT 波形施加于主机施加于的从机选择输出 (前提是 C2[MODFEN] 和 C1[SSOE] = 1)。主机 SS 输出在传输的第一位时间开始时变为有效的低电平，在传输的第八位时间结束后的半个 SPSCK 周期变回高电平。SS IN 波形施加于从机的从机选择输入。

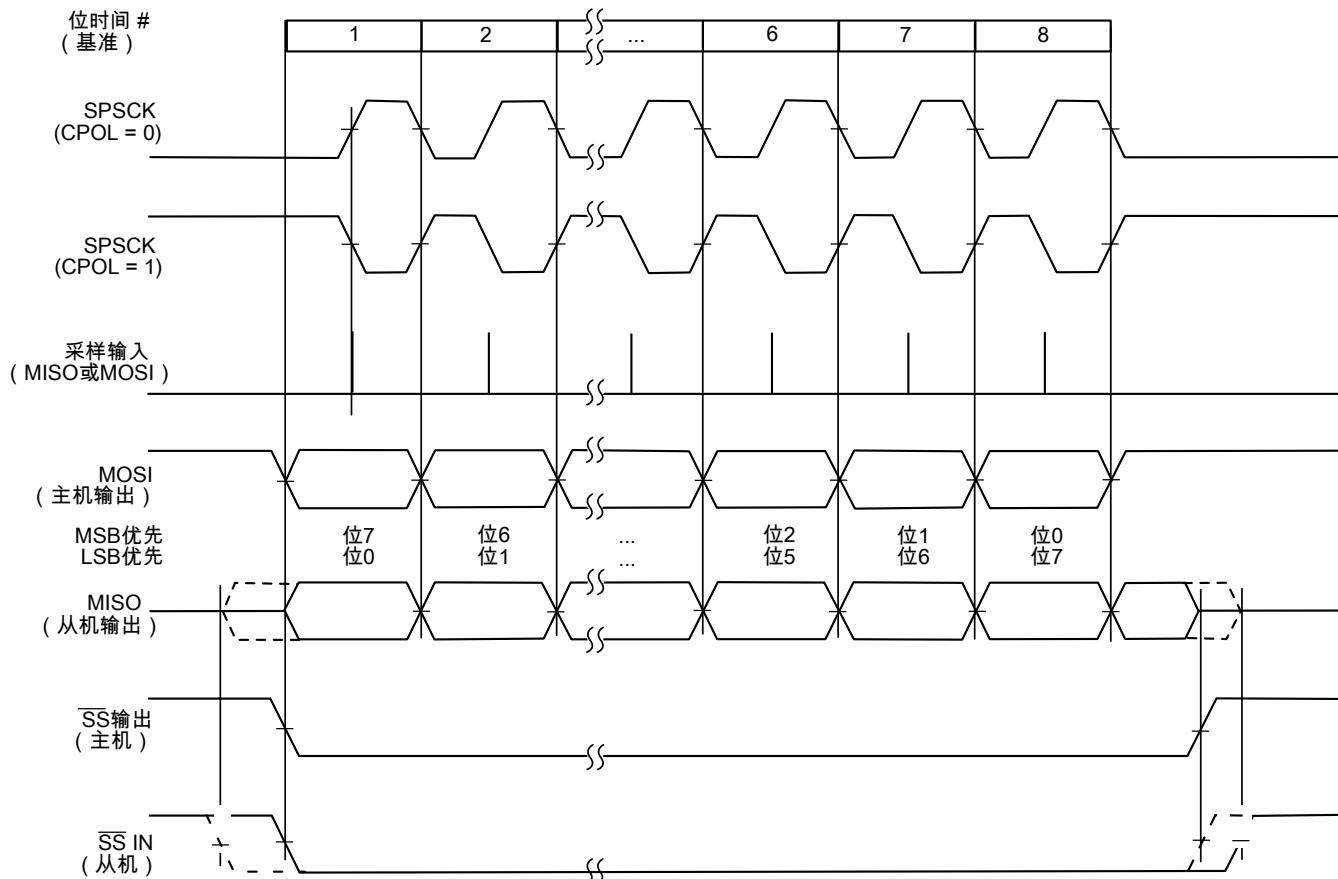


图 30-4. SPI 时钟格式(CPHA = 0)

当 C1[CPHA] = 0 时，从机在 SS 变为低电平有效时开始用第一数据位值 (MSB 或 LSB，具体取决于 LSBFE) 驱动其 MISO 输出。在第一个 SPSCK 边沿，主机和从机分别对 MISO 和 MOSI 输入上的数据位值进行采样。在第二个 SPSCK 边沿，SPI 移位器移动一个位位置，以便移入刚刚采样的位值，并将第二数据位值从移位器的另一端分别移出到主机的 MOSI 输出和从机的 MISO 输出。当 C1[CPHA] = 0 时，从机的 SS 输入在两次传输之间必须变为无效的高电平状态。

### 30.4.5 SPI 波特率生成

如下图所示，SPI 波特率发生器的时钟源是总线时钟。三个预分频位 (SPPR2:SPPR1:SPPR0) 选择预分频因子 1、2、3、4、5、6、7 或 8。三个速率选择位 (SPR3:SPR2:SPR1:SPR0) 对预分频器级的输出进行 2、4、8、16、32、64、128、256 或 512 分频，以获得内部 SPI 主机模式比特率时钟。

波特率发生器仅当 SPI 处于主机模式且发生串行传输时被激活。在其他情况下，会禁用分频器以降低  $I_{DD}$  电流。

波特率因子等式如下所示（不包括 SPI 波特率因子表中保留的组合）。

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \times 2^{(\text{SPR} + 1)}$$

可按照以下公式计算波特率：

$$\text{BaudRate} = \text{BusClock} / \text{BaudRateDivisor}$$

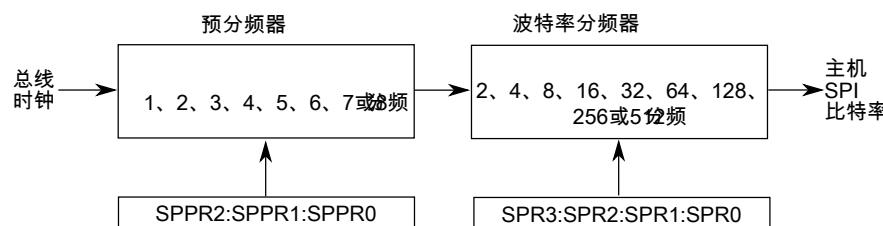


图 30-5. SPI 波特率生成

### 30.4.6 特殊功能

以下各节说明 SPI 模块的特殊功能。

#### 30.4.6.1 SS 输出

$\overline{\text{SS}}$  输出在发送期间自动将  $\overline{\text{SS}}$  引脚驱动为低电平以选择外部器件，在空闲期间将  $\overline{\text{SS}}$  引脚驱动为高电平以取消选择外部器件。当选择  $\overline{\text{SS}}$  输出时， $\overline{\text{SS}}$  输出引脚连接到外部器件的  $\overline{\text{SS}}$  输入引脚。

在 SPI 正常操作期间， $\overline{\text{SS}}$  输出仅在主机模式下可用；要使能该功能，请按照 C1[SSOE] 的说明执行 C1[SSOE] 和 C2[MODFEN] 操作。

使能  $\overline{\text{SS}}$  输出时，模式故障功能禁用。

**注**

在多主机系统中使用  $\overline{SS}$  输出功能时应注意，因为无法利用模式故障功能来检测主机之间的系统错误。

### 30.4.6.2 双向模式 (MOMI 或 SISO)

选择双向模式是在 SPI 控制寄存器 2 中的 SPC0 位置位时（参见下表）。在该模式下，SPI 仅使用一个串行数据引脚与一个或多个外部器件接合。C1[MSTR]决定使用哪个引脚。MOSI 引脚成为串行数据 I/O (MOMI)引脚（适用于主机模式），MISO 引脚成为串行数据 I/O (SISO)引脚（适用于从机模式）。SPI 不使用主机模式下的 MISO 引脚和从机模式下的 MOSI 引脚。

**表 30-1. 正常模式和双向模式**

SPE = 1 时	主机模式 MSTR = 1	从机模式 MSTR = 0
正常模式 <b>SPC0 = 0</b>		
双向模式 <b>SPC0 = 1</b>		

各串行 I/O 引脚的方向取决于 C2[BIDIROE]。如果该引脚配置为输出，则移位寄存器的串行数据从该引脚驱动输出。该引脚也是移位寄存器的串行输入。

SPSCK 对于主机模式是输出，对于从机模式是输入。

$\overline{SS}$  对于主机模式是输入或输出，但对于从机模式始终是输入。

双向模式不影响 SPSCK 功能和  $\overline{SS}$  功能。

**注**

处于双向主机模式且模式故障特性使能时，MISO 和 MOSI 这两个数据引脚都可供 SPI 占用，不过 MOSI 常用于双向模式下的发送且 SPI 不使用 MISO。如果发生模式故障，SPI 会自动切换至从机模式。在该情况下，MISO 变为由 SPI 占用且 SPI 不使用 MOSI。如果 MISO 引脚另作他用，则应考虑这种情形。

## 30.4.7 错误条件

SPI 模块有一个错误条件：模式故障错误。

### 30.4.7.1 模式故障错误

如果 SPI 配置为主机的同时  $\overline{SS}$  输入变为低电平，那么表示存在系统错误（有一个以上的主机可能在同时尝试驱动 MOSI 和 SPSCK 线路）。这种情况下正常工作中是不允许的，在 C2[MODFEN]置位的情况下，它会使 SPI 状态寄存器中的 MODF 位自动置位。

在 SPI 处于主机模式且 C2[MODFEN]被清零的特殊情况下，SPI 不使用  $\overline{SS}$  引脚。此时，模式故障错误功能禁用，MODF 保持被清零状态。如果 SPI 系统配置为从机，那么  $\overline{SS}$  引脚是专用输入引脚。模式故障错误在从机模式下不会发生。

发生模式故障错误时，SPI 会切换到从机模式，但从机输出缓冲区禁用的情况除外。因此，SPSCK、MISO 和 MOSI 引脚会强制变为高阻抗输入，以免与另一输出驱动器发生任何可能的冲突。正在进行的发送会中止并且 SPI 会强制进入空闲状态。

对于在主机模式下配置的 SPI 系统，如果在双向模式下发生模式故障错误，那么 MOMI（双向模式下的 MOSI）的输出使能会被清零（若之前已置位）。对于在从机模式下配置的 SPI 系统，双向模式下不会发生任何模式故障错误。

将模式错误标志自动清零的方法是：先对 SPI 状态寄存器（MODF 置位时）进行读操作，再对 SPI 控制寄存器 1 进行写操作。将模式错误标志清零后，SPI 会再次变为正常主机或从机。

## 30.4.8 低功耗模式选项

本节说明低功耗模式选项。

### 30.4.8.1 Run 模式下的 SPI

在 Run 模式下，当 SPI 控制寄存器 1 中的 SPI 系统使能(SPE)位处于清零状态时，SPI 系统处于低功耗禁用状态。SPI 寄存器仍处于可访问状态，但该模块内核的时钟被禁用。

### 30.4.8.2 Wait 模式下的 SPI

SPI 在 Wait 模式下的操作取决于 SPI 控制寄存器 2 中 SPISWAI 位的状态。

- 如果 C2[SPISWAI]处于清零状态，则当 CPU 处于 Wait 模式时，SPI 正常工作。
- 如果 C2[SPISWAI]置位，则当 CPU 处于 Wait 模式时，SPI 时钟发生器停止工作，SPI 模块进入降耗状态。
  - 如果 C2[SPISWAI]置位且 SPI 配置为主机，那么任何正在进行的发送和接收都会在进入 Wait 模式时停止。SPI 退出 Wait 模式时，发送和接收恢复。
  - 如果 C2[SPISWAI]置位且 SPI 配置为从机，那么任何正在进行的发送和接收都会继续进行（如果主机继续驱动 SPSCK）。这就使从机保持与主机和 SPSCK 的同步。

如果主机在从机处于 Wait 模式的同时发送数据，那么从机会继续发送与 Wait 模式开始时的工作模式一致的数据（也就是说，如果从机当前正在向主机发送 SPIx\_D，则它会继续发送该字节。否则，如果从机当前正在发送从主机接收到的最后数据字节，则它会继续发送主机字节中的数据）。

#### 注

从机处于 Wait 模式或 Stop 模式（外设总线时钟停止但内部逻辑状态得以保留）时，如果预期主机会提供数据，则必须小心处理。即使移位寄存器继续工作，但 SPI 的其余部分也会被关断（也就是说，SPRF 中断要等到退出 Stop 或 Wait 模式后才会生成）。此外，移位寄存器的数据要等到从机 SPI 退出 Wait 或 Stop 模式后才复制到 SPIx\_D 寄存器中。

只有在发送期间进入或退出 Wait 模式时，才会生成 SPRF 标志和 SPIx\_D 复制。如果从机在空闲模式下进入 Wait 模式且在空闲模式下退出 Wait 模式，那么 SPRF 和 SPIx\_D 复制都不会发生。

### 30.4.8.3 Stop 模式下的 SPI

Stop 模式（外设总线时钟停止）下的操作，但内部逻辑状态的保留取决于 SPI 系统。Stop 模式与 C2[SPISWAI]无关。一进入此类 Stop 模式，SPI 模块时钟就会被禁用（保持在高电平或低电平）。

- 如果 SPI 在 CPU 进入 Stop 模式时处于主机模式且正在交换数据，那么发送在 CPU 退出 Stop 模式之前会一直被冻结。退出 Stop 模式后，与外部 SPI 的数据交换正常进行。
- 在从机模式下，SPI 保持与主机同步。

在 Stop 模式（外设总线时钟停止且不保留内部逻辑状态）下，SPI 完全禁用。退出此类 Stop 模式后，所有寄存器都被复位到其默认值，并且必须重新初始化 SPI 模块。

### 30.4.9 复位

寄存器和信号的复位值在“存储器映射”和“寄存器说明”内容中均有介绍，其详细介绍了寄存器及其位字段。

- 如果复位后在从机模式下发生数据发送且未写入 SPIx\_D，则此次发送包含“无用资料”或是在复位前最后一次从主机接收的数据。。
- 复位后从 SPIx\_D 读取始终返回零。

### 30.4.10 中断

SPI 仅在使能时（SPIx\_C1 寄存器中的 SPE 位置位）发起中断请求。以下是关于 SPI 如何发出请求以及 MCU 应如何应答该请求的描述。中断矢量偏移和中断优先权和芯片有关。

与 SPI 系统相关的有四个标志位、三个中断屏蔽位和一个中断矢量。SPI 中断使能屏蔽(SPIE)可从 SPI 接收器完整标志(SPRF)和模式错误标志(MODF)处使能中断。SPI 发送中断使能屏蔽(SPTIE)可从 SPI 发送缓冲器空标志(SPTEF)处使能中断。SPI 匹配中断使能屏蔽位(SPIMIE)可从 SPI 匹配标志(SPMF)处使能中断。其中某个标志位置位且相关中断屏蔽位也置位时，会向 CPU 发送一个硬件中断请求。如果中断屏蔽位已清除，则软件可对相关标志位进行轮询，而非使用中断。SPI 中断服务程序(ISR)应检查标志位以确定引起中断的事件。服务程序在从 ISR 返回之前(通常接近 ISR 起始位置)还应当清除标志位。

#### 30.4.10.1 MODF

MODF 在主机检测到 SS 引脚出错时发生。主机 SPI 必须针对 MODF 特性进行配置(参见 C1[SSOE]位的说明)。一旦 MODF 置位，当前传输就会中止，SPIx\_C1 寄存器中的主机(MSTR)位复位到 0。

MODF 中断在状态寄存器的 MODF 标志中反映。清除该标志也会清除该中断。该中断在 MODF 标志置位期间保持有效状态。MODF 有一个自动清除流程，具体请参见“SPI 状态寄存器”章节中的说明。

### 30.4.10.2 SPRF

SPIF 在新数据已被接收并复制到 SPI 接收数据缓冲区时发生。

SPRF 置位后，对其进行相应操作之前不会进行清零。SPRF 有一套自动清零流程，这在“SPI 状态寄存器”一节有详细说明。如果下一次传输结束之前 SPRF 仍未进行相应操作（即在另一次传输过程中 SPRF 仍有效），则后续传输将被忽略而且不会有新数据复制到数据寄存器。

### 30.4.10.3 SPTEF

SPTEF 在 SPI 发送缓冲区准备好接受新数据时发生。

SPTEF 置位后，对其进行相应操作之前不会进行清零。SPTEF 有一套自动清零流程，这在 SPI 状态寄存器中有详细说明。

### 30.4.10.4 SPMF

SPMF 在接收数据缓冲区中的数据等于 SPI 匹配寄存器中的数据时发生。

### 30.4.10.5 低功耗模式下的异步中断

CPU 处于 Wait 模式或 Stop 模式且 SPI 模块接收发送时，SPI 模块可生成一个异步中断将 CPU 从低功耗模式中唤醒。该模块仅在以下条件全部得到满足时生成异步中断：

1. C1[SPIE]置 1。
2. CPU 处于 Wait 模式 (C2[SPISWAI]在该情况下必须为 1) 或处于 Stop 模式 (外部总线时钟停止但内部逻辑状态得以保留)。
3. SPI 模块处于从机模式。
4. 收到的发送结束。

当中断唤醒 CPU 且外设总线时钟再次处于工作状态时，此 SPI 模块会将接收到的数据从移位器中复制到数据寄存器中并生成标志信号。在唤醒阶段，如果从主机连续不断地发送，则会破坏首次接收到的数据。

## 30.5 初始化/应用信息

本节讨论如何初始化和使用 SPI 的例子。

### 30.5.1 初始化序列

在 SPI 模块可用于通信之前，必须执行如下的初始化程序：

1. 更新控制寄存器 1 (SPIx\_C1)以使能 SPI 并控制中断使能。该寄存器还可将 SPI 设置为主机或从机，确定时钟相位和极性，以及配置 SPI 主要选项。
2. 更新控制寄存器 2 (SPIx\_C2)以使能 SPI 匹配中断特性等其他 SPI 功能、主机模式故障功能和双向模式输出，以及控制和其他可选特性。
3. 更新波特率寄存器(SPIx\_BR)以设置 SPI 主机的预分频器和比特率因子。
4. 用要与接收数据寄存器作比较的值更新硬件匹配寄存器(SPIx\_M)，以便在硬件匹配中断使能时触发中断。
5. 在主机中，在 S[SPTEF] = 1 时对 SPIx\_S 进行读操作，然后写入发送数据寄存器 (SPIx\_D)以开始传输。

### 30.5.2 伪代码示例

在此示例中，SPI 模块设置为主机模式，且只使能硬件匹配中断。SPI 在以 2 分频总线时钟的最大波特率运行。时钟相位和极性针对高电平有效 SPI 时钟而设置，其中 SPSCK 上的第一条边沿在首个数据传输周期开始时出现。

<b>SPIx_C1=0x54(%01010100)</b>			
位 7	SPIE	= 0	禁用接收和模式故障中断
位 6	SPE	= 1	启用 SPI 系统
位 5	SPTIE	= 0	禁用 SPI 发送中断
位 4	MSTR	= 1	将 SPI 模块设置为 SPI 主机
位 3	CPOL	= 0	将 SPI 时钟配置为高电平有效
位 2	CPHA	= 1	SPSCK 上的第一条边沿在首个数据传输周期开始时出现
位 1	SSOE	= 0	使能模式故障时确定 SS 引脚功能
位 0	LSBFE	= 0	SPI 串行数据传输从最高有效位开始

<b>SPIx_C2 = 0x80(%10000000)</b>			
位 7	SPMIE	= 1	SPI 硬件匹配中断已使能
位 6		= 0	未执行
位 5		= 0	保留
位 4	MODFEN	= 0	禁用模式故障功能
位 3	BIDIROE	= 0	SPI 数据 I/O 引脚用作输入

下一页继续介绍此表...

**SPIx\_C2 = 0x80(%10000000)**

位 2	= 0	保留
位 1	= 0	SPI 时钟在 Wait 模式下运行
位 0	= 0	数据输入和输出的引脚不复用

**SPIx\_BR = 0x00(%00000000)**

位 7	= 0	保留
位 6:4	= 000	将预分频器除数设置为 1
位 3:0	= 0000	将波特率除数设置为 2

**SPIx\_S = 0x00(%00000000)**

位 7	SPRF	= 0	标志在接收数据缓冲区满时置位
位 6	SPMF	= 0	标志在 SPIx_M = 接收数据缓冲区时置位
位 5	SPTEF	= 0	标志在发送数据缓冲区满时置位
位 4	MODF	= 0	主机模式的模式错误标志
位 3:0		= 0	保留

**SPIx\_M = 0xXX**

放置硬件匹配缓冲区的位 0 到位 7。

**SPIx\_D = 0xxx**

放置由发送缓冲区发送并由接收缓冲区接收的数据的位 0 到位 7。

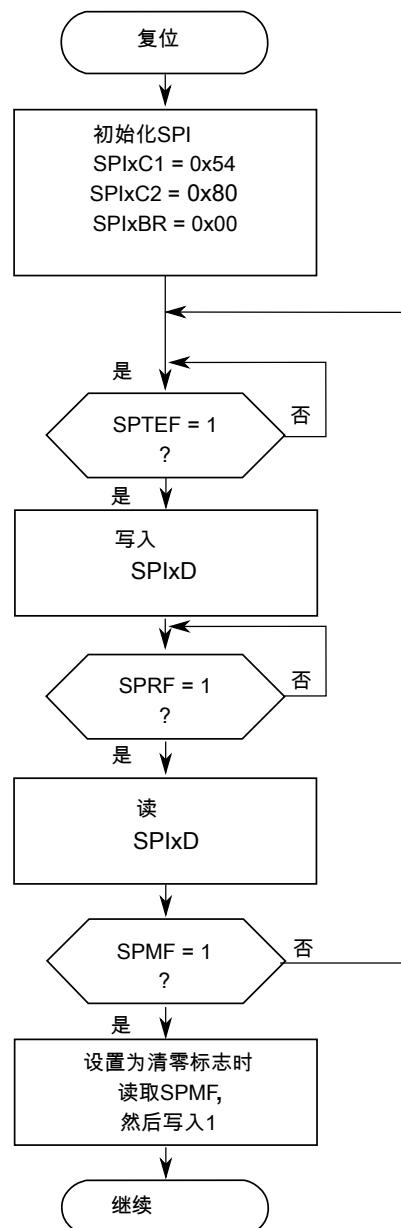


图 30-6. 时下 SPI 主机的初始化流程图示例

# 第 31 章 I2C 模块

## 31.1 简介

### 注

关于与具体芯片相关的本模块详细操作示例，请参见芯片配置信息。

I2C (I<sup>2</sup>C、I2C、或者 IIC) 模块可实现多个设备之间的通信。

该接口适用于总线速度低于 100 kbit/s 的应用场景。在降低总线负载状态下，I2C 设备能够在更高的波特率下运行，最高波特率为时钟/20。最长通信距离和可连接的设备数量受最高 400 pF 的总线电容限制。I2C 模块还符合版本 2 的系统管理总线 (SMBus) 规格。

### 31.1.1 特性

I2C 模块的特性如下：

- 与 I<sup>2</sup>C-总线规范兼容
- 多主机操作
- 可通过软件编程选择 64 种不同串行时钟频率的其中之一
- 可通过软件选择使能应答位
- 通过中断驱动的逐字节数据传输
- 仲裁丢失产生中断，并且从主机模式切换到从机模式
- 广播地址识别中断
- 开始和停止信号的生成和检测
- 可重复的开始信号生成和检测
- 应答位的生成和检测
- 可检测总线忙状态
- 可识别通用广播地址
- 可扩展 10 位地址

- 支持系统管理总线(*SMBus*)规范 (第 2 版)
- 可编程的输入去抖滤波器
- 从机地址匹配时的低功耗唤醒
- 支持从机地址范围

### 31.1.2 工作模式

各种低功耗模式下的 I2C 模块工作如下所示：

- Run 模式：这是基本工作模式。要在该模式下节能，则禁用某些外设。
- Wait 模式：该模块可在内核处于 Wait 模式时继续工作且可提供唤醒中断。
- Stop 模式：在 Stop 下，除非在 Stop 模式下地址匹配使能，否则该模块处于无效状态以降低功耗。STOP 指令不影响 I2C 模块的寄存器状态。

### 31.1.3 结构框图

下图是 I2C 模块的结构框图。

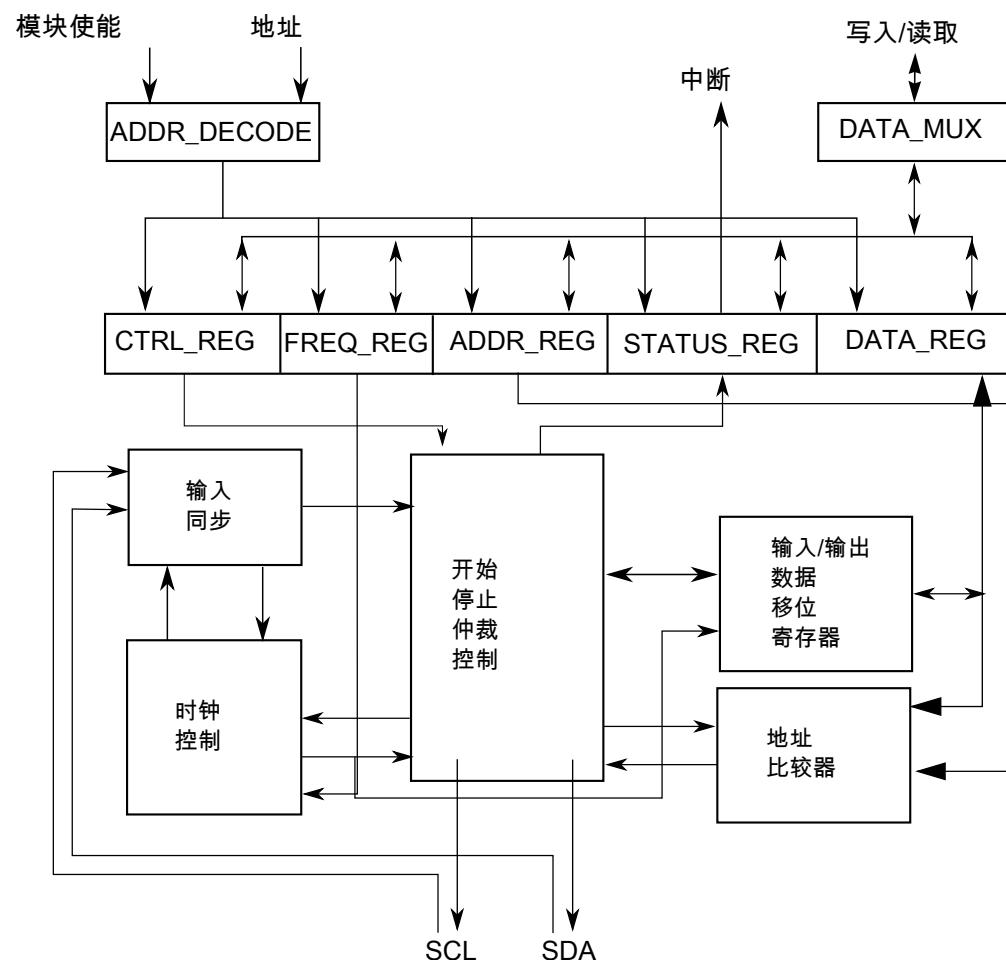


图 31-1. I2C 功能结构框图

## 31.2 I<sup>2</sup>C 信号说明

I<sup>2</sup>C 的信号属性如下表所示。

表 31-1. I<sup>2</sup>C 信号说明

信号	说明	I/O
SCL	I <sup>2</sup> C 系统的双向串行时钟线路。	I/O
SDA	I <sup>2</sup> C 系统的双向串行数据线路。	I/O

### 31.3 存储器映像/寄存器定义

本节详细介绍最终用户可访问的所有 I2C 寄存器。

#### I2C 存储器映射

绝对地址（十六进制）	寄存器名称	宽度（单位：位）	访问	复位值	小节/页
4006_6000	I2C 地址寄存器 1 (I2C0_A1)	8	R/W	00h	<a href="#">31.3.1/523</a>
4006_6001	I2C 分频器寄存器 (I2C0_F)	8	R/W	00h	<a href="#">31.3.2/523</a>
4006_6002	I2C 控制寄存器 1 (I2C0_C1)	8	R/W	00h	<a href="#">31.3.3/524</a>
4006_6003	I2C 状态寄存器 (I2C0_S)	8	R/W	80h	<a href="#">31.3.4/525</a>
4006_6004	I2C 数据 I/O 寄存器 (I2C0_D)	8	R/W	00h	<a href="#">31.3.5/527</a>
4006_6005	I2C 控制寄存器 2 (I2C0_C2)	8	R/W	00h	<a href="#">31.3.6/528</a>
4006_6006	I2C 可编程输入去抖滤波器寄存器 (I2C0_FLT)	8	R/W	00h	<a href="#">31.3.7/529</a>
4006_6007	I2C 范围地址寄存器 (I2C0_RA)	8	R/W	00h	<a href="#">31.3.8/530</a>
4006_6008	I2C SMBus 控制和状态寄存器 (I2C0_SMB)	8	R/W	00h	<a href="#">31.3.9/530</a>
4006_6009	I2C 地址寄存器 2 (I2C0_A2)	8	R/W	C2h	<a href="#">31.3.10/532</a>
4006_600A	I2C SCL 低位定时溢出寄存器高电平 (I2C0_SLTH)	8	R/W	00h	<a href="#">31.3.11/532</a>
4006_600B	I2C SCL 低位定时溢出寄存器低电平 (I2C0_SLTL)	8	R/W	00h	<a href="#">31.3.12/533</a>
4006_7000	I2C 地址寄存器 1 (I2C1_A1)	8	R/W	00h	<a href="#">31.3.1/523</a>
4006_7001	I2C 分频器寄存器 (I2C1_F)	8	R/W	00h	<a href="#">31.3.2/523</a>
4006_7002	I2C 控制寄存器 1 (I2C1_C1)	8	R/W	00h	<a href="#">31.3.3/524</a>
4006_7003	I2C 状态寄存器 (I2C1_S)	8	R/W	80h	<a href="#">31.3.4/525</a>
4006_7004	I2C 数据 I/O 寄存器 (I2C1_D)	8	R/W	00h	<a href="#">31.3.5/527</a>
4006_7005	I2C 控制寄存器 2 (I2C1_C2)	8	R/W	00h	<a href="#">31.3.6/528</a>
4006_7006	I2C 可编程输入去抖滤波器寄存器 (I2C1_FLT)	8	R/W	00h	<a href="#">31.3.7/529</a>
4006_7007	I2C 范围地址寄存器 (I2C1_RA)	8	R/W	00h	<a href="#">31.3.8/530</a>
4006_7008	I2C SMBus 控制和状态寄存器 (I2C1_SMB)	8	R/W	00h	<a href="#">31.3.9/530</a>
4006_7009	I2C 地址寄存器 2 (I2C1_A2)	8	R/W	C2h	<a href="#">31.3.10/532</a>
4006_700A	I2C SCL 低位定时溢出寄存器高电平 (I2C1_SLTH)	8	R/W	00h	<a href="#">31.3.11/532</a>
4006_700B	I2C SCL 低位定时溢出寄存器低电平 (I2C1_SLTL)	8	R/W	00h	<a href="#">31.3.12/533</a>

### 31.3.1 I2C 地址寄存器 1 (I2Cx\_A1)

该寄存器包含 I2C 模块使用的从机地址。

地址: 基址 基准 + 0h 偏移

位	7	6	5	4		3	2	1	0
读写	AD[7:1]								0
复位	0	0	0	0		0	0	0	0

I2Cx\_A1 字段描述

字段	描述
7–1 AD[7:1]	地址 包含编址为从机地址时由 I2C 模块使用的主要从机地址。该字段用于 7 位地址方案以及 10 位地址方案中的七个低位。
0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

### 31.3.2 I2C 分频器寄存器 (I2Cx\_F)

地址: 基址 基准 + 1h 偏移

位	7	6	5	4		3	2	1	0
读写	MULT					ICR			
复位	0	0	0	0		0	0	0	0

I2Cx\_F 字段描述

字段	描述
7–6 MULT	乘法器系数 定义乘法器系数(mul)。该系数与 SCL 分频器一起用于生成 I2C 波特率。 00 mul = 1 01 mul = 2 10 mul = 4 11 保留
ICR	时钟速率 对 I2C 模块进行预分频，以便选择比特率。此字段和 MULT 字段确定 I2C 波特率、SDA 保持时间、SCL 起始状态保持时间和 SCL 停止状态保持时间。有关每项 ICR 设置对应值的列表，请参见 <a href="#">I2C 分频器和保持值</a> 。 SCL 分频器乘以乘法器系数(mul)可确定 I2C 波特率。 $\text{I2C baud rate} = \text{I2C module clock speed (Hz)} / (\text{mul} \times \text{SCL divider})$ SDA 保持时间是从 SCL 下降沿 (I2C 时钟) 到 SDA 更改 (I2C 数据) 的延迟。

下一页继续介绍此表...

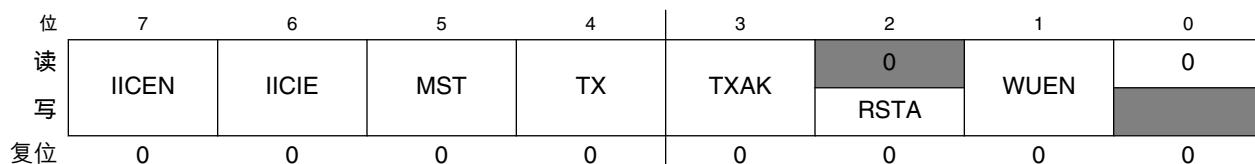
**I2Cx\_F 字段描述 (继续)**

字段	描述				
	<p>SDA hold time = I2C module clock period (s) × mul × SDA hold value</p> <p>SCL 起始状态保持时间是从 SCL 高电平 (开始条件) 时从 SDA 下降沿 (I2C 数据) 到 SCL 下降沿 (I2C 时钟) 的延迟。</p> <p>SCL start hold time = I2C module clock period (s) × mul × SCL start hold value</p> <p>SCL 停止状态保持时间是 SCL 高电平 (停止条件) 时从 SCL 上升沿 (I2C 时钟) 到 SDA 上升沿 (I2C 数据) 的延迟。</p> <p>SCL stop hold time = I2C module clock period (s) × mul × SCL stop hold value</p> <p>例如，假设 I2C 模块时钟速率为 8 MHz，下表展示了选择不同的 ICR 和 MULT 以实现 100 kbit/s 的 I2C 波特率时可能的保持时间值。</p>				

MULT	ICR	保持时间(μs)		
		SDA	SCL 开始状态	SCL 停止状态
2h	00h	3.500	3.000	5.500
1h	07h	2.500	4.000	5.250
1h	0Bh	2.250	4.000	5.250
0h	14h	2.125	4.250	5.125
0h	18h	1.125	4.750	5.125

**31.3.3 I2C 控制寄存器 1 (I2Cx\_C1)**

地址: 基址 基准 + 2h 偏移

**I2Cx\_C1 字段描述**

字段	描述
7 IICEN	I2C 使能 使能 I2C 模块工作。 0 禁用 1 使能
6 IICIE	I2C 中断使能 使能 I2C 中断请求。 0 禁用 1 使能

下一页继续介绍此表...

## I2Cx\_C1 字段描述 (继续)

字段	描述
5 MST	<p>主机模式选择</p> <p>MST 从 0 更改为 1 时 ,将在总线上生成 START 信号 ,并选择主机模式。此位从 1 更改为 0 时 ,将生成 STOP 信号 , 工作模式从主机更改为从机。</p> <p>0 从机模式 1 主机模式</p>
4 TX	<p>发送模式选择</p> <p>选择主机和从机传输的方向。在主机模式下 ,必须根据所需的传输类型将此位置位。因此 ,对地址周期而言 ,此位始终置位。编址为从机地址时 ,此位必须由软件根据状态寄存器中的 SRW 位置位。</p> <p>0 接收 1 发送</p>
3 TXAK	<p>发送应答使能</p> <p>指定在主机和从机接收器的数据应答周期过程中驱动到 SDA 的值。SMB[FACK]的值影响 NACK/ACK 的生成。</p> <p>注: 对 TXAK 采取写入操作之前 , SCL 会一直保持低电平。</p> <p>0 向接下来的接收字节 ( 如果 FACK 已清零 ) 或当前的接收字节 ( 如果 FACK 已置位 ) 上的总线发送了应答信号。 1 没有向接下来的接收数据字节 ( 如果 FACK 已清零 ) 或当前的接收数据字节 ( 如果 FACK 已置位 ) 上的总线发送应答信号。</p>
2 RSTA	<p>重复 START</p> <p>只要是当前主机 ,向此位写入 1 就会生成重复的 START 条件。此位将始终读取为 0。在错误的时间尝试重复将导致仲裁丢失。</p>
1 WUEN	<p>唤醒使能</p> <p>出现从机地址匹配时 ,I2C 模块可将 MCU 从低功耗模式唤醒 ,无需运行外设总线。</p> <p>0 正常工作。在低功耗模式下出现地址匹配时 ,不生成中断。 1 在低功耗模式下使能唤醒功能。</p>
0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

## 31.3.4 I2C 状态寄存器 (I2Cx\_S)

地址: 基址 基准 + 3h 偏移

位	7	6	5	4	3	2	1	0
读	TCF	IAAS	BUSY	ARBL	RAM	SRW	IICIF	RXAK
写				w1c			w1c	
复位	1	0	0	0	0	0	0	0

**I2Cx\_S 字段描述**

字段	描述
7 TCF	<p>传输完成标志</p> <p>对字节传输做出应答；完成一个字节的传输后，TCF 置位。此位只有在与 I2C 模块进行传输期间或传输完成后即刻有效。通过在接收模式下对 I2C 数据寄存器进行读取操作或在发送模式下对 I2C 数据寄存器进行写入操作，可以清零 TCF。</p> <p>0 传输进行中 1 传输完成</p>
6 IAAS	<p>作为从机被寻址</p> <p>在以下情况下将设置此位：</p> <ul style="list-style-type: none"> <li>• 调用地址与 A1 寄存器中的已编程主要从机地址匹配，或与 RA 寄存器中的范围地址匹配（必须设置为非零值，且在 I2C_C2[RMEN] = 1 的条件下）。</li> <li>• C2[GCAEN]置位，并且接收到通用调用。</li> <li>• SMB[SIICAEN]置位，并且调用地址与第二个已编程从机地址匹配。</li> <li>• ALERTEN 置位，并且接收到 SMBus 提醒响应地址</li> <li>• RMEN 置位，并且接收到的地址在 A1 和 RA 寄存器的值范围内。</li> </ul> <p>IAAS 在 ACK 位之前置位。CPU 必须检查 SRW 位并相应地设置 TX/RX。将任何值写入 C1 寄存器都会将此位清零。</p> <p>0 未被寻址 1 作为从机被寻址</p>
5 BUSY	<p>总线忙</p> <p>无论是从机还是主机模式，都指示总线的状态。检测到 START 信号时置位此位，检测到 STOP 信号时清零此位。</p> <p>0 总线空闲 1 总线忙</p>
4 ARBL	<p>仲裁丢失</p> <p>仲裁程序丢失时，此位由硬件置位。必须通过将 1 写入 ARBL 位由软件将此位清零。</p> <p>0 标准总线工作。 1 丢失仲裁。</p>
3 RAM	<p>范围地址匹配</p> <p>如果 I2C_C2[RMEN] = 1，以下任一条件都会导致此位置 1：</p> <ul style="list-style-type: none"> <li>• 已收到任何与 RA 寄存器中地址匹配的非零调用地址。</li> <li>• 调用地址在 A1 和 RA 寄存器的值范围内。</li> </ul> <p>向 C1 寄存器写入任何值都会将此位清 0。</p> <p>0 未被寻址 1 作为从机被寻址</p>
2 SRW	<p>从机读取/写入</p> <p>编址为从机地址时，SRW 指示发送到主机的调用地址的 R/W 命令位的值。</p> <p>0 从机接收，主机向从机写入 1 从机发送，主机对从机读取</p>

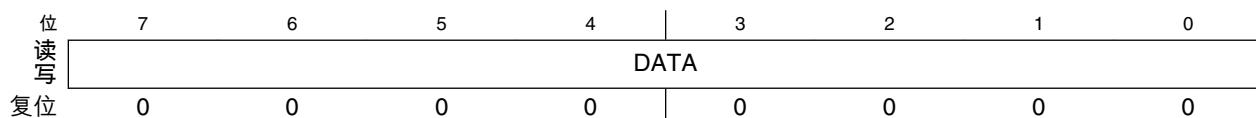
下一页继续介绍此表...

**I2Cx\_S 字段描述 (继续)**

字段	描述
1 IICIF	<p>中断标志</p> <p>有中断挂起时，此位将置位。必须通过软件将 1 写入该位来将其清零，比如在中断程序中。以下事件之一可置位此位：</p> <ul style="list-style-type: none"> <li>• FACK 为 0 的情况下，完成一个字节的传输，包括 ACK/NACK 位。在接收模式下置位此位后，通过将 0 或 1 写入 TXAK，在总线上发送 ACK 或 NACK。</li> <li>• FACK 为 1 的情况下，完成一个字节的传输，不包括 ACK/NACK 位。</li> <li>• 从机地址与调用地址的匹配包括主要从机地址、范围从机地址、提醒响应地址、第二从机地址或通用调用地址。</li> <li>• 仲裁丢失</li> <li>• 在 SMBus 模式下，除了 SCL 和 SDA 高位定时溢出以外的任何定时溢出</li> <li>• 如果输入干扰滤波器寄存器中的 SSIE 位值为 1，I2C 总线停止位或起始位检测</li> </ul> <p>注：要清零 I2C 总线停止状态或起始状态检测中断：在中断服务程序中，首先将 1 写入输入去抖滤波器寄存器中的 STOPF 或 STARTF 位以将其清零，然后再清零 IICIF 位。如果此顺序相反，则 IICIF 位的电平将再次变为有效值。</p> <p>0 没有中断挂起 1 中断挂起</p>
0 RXAK	<p>接收应答</p> <p>0 在总线上完成一个字节的数据发送后，接收到应答信号 1 未检测到应答信号</p>

**31.3.5 I2C 数据 I/O 寄存器 (I2Cx\_D)**

地址: 基址 基准 + 4h 偏移

**I2Cx\_D 字段描述**

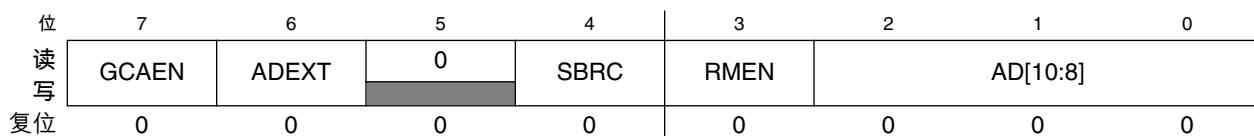
字段	描述
DATA	<p>数据</p> <p>在主机发送模式下，向该寄存器写入数据时，会启动数据传输。首先发送最高有效位。在主机接收模式下，从该寄存器读取数据开始接收下一个字节的数据。</p> <p>注：从主机接收模式中进行转换时，在读取数据寄存器之前先切换 I2C 模式，以防意外启动主机接收数据传输。</p> <p>在从机模式下，发生地址匹配后，可使用相同的功能。</p> <p>C1[TX]位必须正确反映主机和从机模式下所需的传输方向，只有这样才能开始发送。例如，假设 I2C 模块配置为主机发送，但需要主机接收，此时对数据寄存器进行读取操作就不会开始接收。</p> <p>I2C 模块配置为主机接收或从机接收模式时，对数据寄存器进行读取操作会返回收到的最后一个字节。数据寄存器不会反映 I2C 总线上发送的每一个字节，软件也不会通过回读来检验某个字节是否正确地写入到数据寄存器中。</p>

**I2Cx\_D 字段描述 (继续)**

字段	描述
	在主机发送模式下，MST（开始位）或 RSTA（重复的开始位）的电平变为有效值之后向数据寄存器写入的第一个字节的数据将用于地址传输，而且必须包含调用地址（在位 7-1 中），再连上所需的 R/W 位（在位置位 0 中）。

**31.3.6 I2C 控制寄存器 2 (I2Cx\_C2)**

地址: 基址 基准 + 5h 偏移

**I2Cx\_C2 字段描述**

字段	描述
7 GCAEN	通用调用地址使能 使能通用调用地址。 0 禁用 1 已使能
6 ADEXT	地址扩展 控制从机地址所用的位数。 0 7 位地址方案 1 10 位地址方案
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
4 SBRC	从机波特率控制 在最高频率条件下使能独立从机模式波特率，这将在非常快的 I2C 模式下在 SCL 中强制时钟拉伸。对从机而言，“非常快”的模式可以是这样的：例如，主机传输速率为 40 kbit/s，但从机可在低至 10 kbit/s 的速率下捕捉主机的数据。 0 从机波特率遵循主机波特率，可能发生时钟拉伸 1 从机波特率与主机波特率无关
3 RMEN	范围地址匹配使能 此位针对 A1 和 RA 寄存器值范围内的地址控制从机地址匹配。此位置位时，针对任何大于 A1 寄存器值且小于或等于 RA 寄存器值的地址发生从机地址匹配。 0 禁用范围模式。与 A1、RA 寄存器值表示范围内的地址不进行匹配。 1 使能范围模式。从机接收到与 A1 和 RA 寄存器值范围内的地址时，发生地址匹配。
AD[10:8]	从机地址 包含 10 位地址方案中从机地址的三个高位。只有在 ADEXT 位置位的情况下，该字段才有效。

### 31.3.7 I2C 可编程输入去抖滤波器寄存器 (I2Cx\_FLT)

地址: 基址 基准 + 6h 偏移

位	7	6	5	4	3	2	1	0
读	SHEN	STOPF	SSIE	STARTF			FLT	
写	w1c			w1c				
复位	0	0	0	0	0	0	0	0

I2Cx\_FLT 字段描述

字段	描述
7 SHEN	<p>停止状态保持使能</p> <p>置位此位以避免在发生任何数据发送或接收时进入 Stop 模式。</p> <p>以下方案说明此隔离功能：</p> <ol style="list-style-type: none"> <li>1. IIC 模块被配置做基本传输，并将 SHEN 位置 1。</li> <li>2. 传输开始。</li> <li>3. MCU 用信号通知 I2C 模块进入 Stop 模式。</li> <li>4. 当前正在传输的字节（包括地址和数据）完成其传输。</li> <li>5. I2C 从机和主机确认传输中的字节已完成其传输，并确认进入 Stop 模式的申请。</li> <li>6. 收到 I2C 模块对申请进入 Stop 模式的确认后，MCU 确定是否关闭 I2C 模块的时钟。</li> </ol> <p>MCU 通知 IIC 模块进入 Stop 模式时，如果 SHEN 位置 1 且 I2C 模块处于闲置或禁用状态，那么此模块将立即确认进入 Stop 模式。</p> <p>在 SHEN 被清 0 时进入 Stop 模式，则被暂停的所有发送或接收的数据处于未完成状态：要在 MCU 退出 Stop 模式后恢复所有的发送或接收，软件必须通过重发从机地址来重新初始化传输。</p> <p>如果 MCU 进入 Stop 模式之前，I2C 控制寄存器 1 的 IICIE 位已置 1，则在 MCU 从 Stop 模式中唤醒后，系统软件将收到 I2C 状态寄存器的 TCF 位触发的中断。</p> <p>0 停止迟滞禁用。MCU 进入 Stop 模式未受门控。 1 停止迟滞使能。</p>
6 STOPF	<p>I2C 总线停止状态检测标志</p> <p>检测到 I2C 总线处于停止状态时，硬件会置位此位。必须通过将 1 写入 STOPF 位来将其清零。</p> <p>0 I2C 总线上无停止状态 1 I2C 总线上检测到停止状态</p>
5 SSIE	<p>I2C 总线停止状态或起始状态中断使能</p> <p>此位用于使能 I2C 总线停止状态或起始状态检测的中断。</p> <p>注：要清除 I2C 总线停止状态或起始状态检测中断：在中断服务程序中，首先将 1 写入 STOPF 或 STARTF 位以将其清零，然后将状态寄存器中的 IICIF 位清零。如果此顺序相反，则 IICIF 位的电平将再次变为有效值。</p> <p>0 停止状态或起始状态检测中断禁用 1 停止状态或起始状态检测中断使能</p>
4 STARTF	<p>I2C 总线起始状态检测标志</p> <p>检测到 I2C 总线的起始状态后，硬件会置位此位。必须将 1 写入 STARTF 位来将其清零。</p>

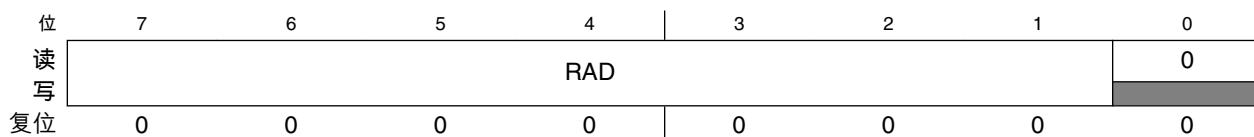
下一页继续介绍此表...

**I2Cx\_FLT 字段描述 (继续)**

字段	描述
	0 I2C 总线上无起始状态 1 I2C 总线上检测到起始状态
FLT	I2C 可编程滤波器系数  控制滤波器必须吸收的毛刺宽度(从 I2C 模块时钟周期的角度而言)。对于宽度小于或等于此宽度设置的任何毛刺，滤波器都不会允许该毛刺通过。  0h 无滤波器/旁路 1-Fh 对宽度最多为 n 个 I2C 模块时钟周期的毛刺进行滤波，其中 n=1-15d

**31.3.8 I2C 范围地址寄存器 (I2Cx\_RA)**

地址: 基址 基准 + 7h 偏移

**I2Cx\_RA 字段描述**

字段	描述
7-1 RAD	范围从机地址  该字段包含 I2C 模块使用的从机地址。此字段将用于 7 位地址方案中。如果 I2C_C2[RMEN]置 1，写入任何非零值都会使能此寄存器。可将此寄存器值视作范围匹配模式下的最大边界。
0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

**31.3.9 I2C SMBus 控制和状态寄存器 (I2Cx\_SMB)****注**

如果 SCL 和 SDA 信号保持高电平的时间超过高位定时溢出周期的时长，SHTF1 标志将置位。达到此阈值之前，系统正在检测这些信号保持多久的高电平时，主机假定总线空闲。然而，在总线状态为空闲的条件下，SHTF1 位会在总线发送过程中置 1。

**注**

因为总线速度太高，难以匹配 SMBus 协议，所以 TCKSEL 位置位后，再无需监控 SHTF1 位。

地址: 基址 基准 + 8h 偏移

位 读 写	7	6	5	4	3	2	1	0
复位	0	0	0	0	0	0	0	SHTF2IE
	w1c			w1c				

**I2Cx\_SMB 字段描述**

字段	描述
7 FACK	快速 NACK/ACK 使能  对于 SMBus 数据包错误检查，CPU 必须能够根据接收数据字节的结果发出 ACK 或 NACK。  0 在接收完数据后紧接着发送 ACK 或 NACK 1 接收数据字节后将 0 写入 TXAK 可生成 ACK。接收数据字节后将 1 写入 TXAK 可生成 NACK。
6 ALERTEN	SMBus 提醒响应地址使能  使能或禁用 SMBus 提醒响应地址匹配。  注： 主机对某个使用提醒响应地址的器件作出响应后，必须使用软件将器件的地址放在总线上。SMBus 规范中介绍了提醒协议。  0 禁用 SMBus 提醒响应地址匹配 1 使能 SMBus 提醒响应地址匹配
5 SIICAEN	第二 I2C 地址使能  使能或禁用 SMBus 器件默认地址。  0 禁用 I2C 地址寄存器 2 匹配 1 使能 I2C 地址寄存器 2 匹配
4 TCKSEL	定时溢出计数器时钟选择  选择定时溢出计数器的时钟源。  0 定时溢出计数器按 I2C 模块时钟/64 的频率计数 1 定时溢出计数器按 I2C 模块时钟的频率计数
3 SLTF	SCL 低位定时溢出标志  在 SLT 寄存器（包括 SLTH 和 SLTL 寄存器）加载非零值(LoValue)且发生 SCL 低位定时溢出的情况下，此位将置位。软件通过向此位写入逻辑 1 将此位清零。  注： SLT 寄存器的值为 0 时，将禁用低位定时溢出功能。  0 未发生低位定时溢出 1 发生了低位定时溢出
2 SHTF1	SCL 高位定时溢出标志 1  SCL 和 SDA 保持高电平的时间超过时钟 × LoValue / 512 时（表明总线空闲），该只读位将置位。此位自动清零。  0 未发生 SCL 高位和 SDA 高位定时溢出 1 发生了 SCL 高位和 SDA 高位定时溢出
1 SHTF2	SCL 高位定时溢出标志 2

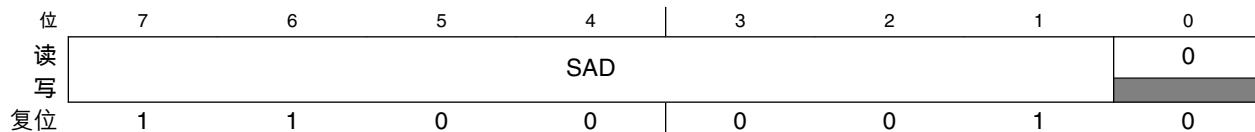
下一页继续介绍此表...

**I2Cx\_SMB 字段描述 (继续)**

字段	描述
	SCL 保持高位且 SDA 保持低位的时间超过时钟 $\times$ LoValue / 512 时, 此位将置位。软件通过向此位写入 1 将此位清零。 0 未发生 SCL 高位和 SDA 低位定时溢出 1 发生了 SCL 高位和 SDA 低位定时溢出
0 SHTF2IE	SHTF2 中断使能 使能 SCL 高位和 SDA 低位定时溢出中断。 0 禁用 SHTF2 中断 1 使能 SHTF2 中断

**31.3.10 I2C 地址寄存器 2 (I2Cx\_A2)**

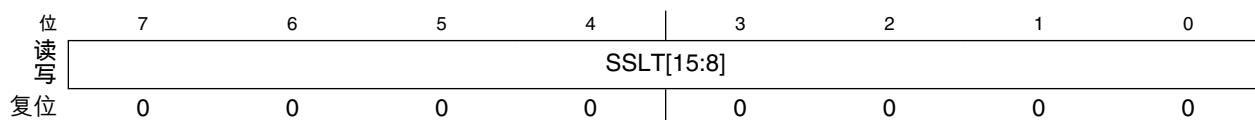
地址: 基址 基准 + 9h 偏移

**I2Cx\_A2 字段描述**

字段	描述
7-1 SAD	SMBus 地址 包含 SMBus 使用的从机地址。该字段用于器件默认地址或其他相关地址。
0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

**31.3.11 I2C SCL 低位定时溢出寄存器高电平 (I2Cx\_SLTH)**

地址: 基址 基准 + Ah 偏移

**I2Cx\_SLTH 字段描述**

字段	描述
SSLT[15:8]	SSLT[15:8] 确定 SCL 低位定时溢出周期的 SCL 低位定时溢出值的最高有效位。

### 31.3.12 I2C SCL 低位定时溢出寄存器低电平 (I2Cx\_SLTL)

地址: 基址 基准 + Bh 偏移

位 读写	7	6	5	4		3	2	1	0	
	SSLT[7:0]									
复位	0	0	0	0		0	0	0	0	

**I2Cx\_SLTL** 字段描述

字段	描述
SSLT[7:0]	SSLT[7:0] 确定 SCL 低位定时溢出周期的 SCL 低位定时溢出值的最低有效位。

## 31.4 功能说明

本章节全面说明 I2C 模块的功能。

### 31.4.1 I2C 协议

I2C 总线系统采用串行数据线路(SDA)和串行时钟线路(SCL)进行数据传输。

连接至 I2C 总线系统的所有器件都必须具有开漏或集电极开路输出。使用外部上拉电阻在这两条线路上执行逻辑和(AND)功能。电阻值取决于系统。

正常情况下，一个标准的通信实例由四部分组成：

1. 开始信号
2. 从机地址发送
3. 数据传输
4. STOP 信号

不得将停止信号与 CPU 停止指令混淆。下图对 I2C 总线系统通信进行了说明。

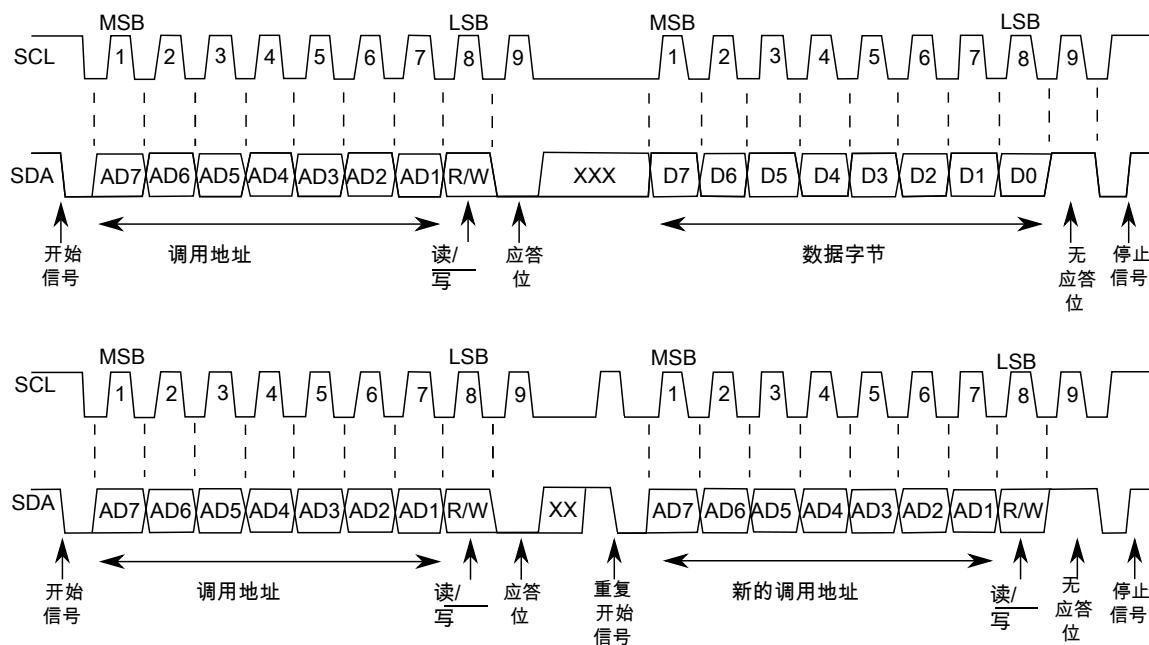


图 31-2. I2C 总线发送信号

### 31.4.1.1 START 信号

当无主机支配总线时 (SCL 和 SDA 均为高电平)，总线空闲。当总线空闲时，主机可通过发送 START 信号启动通信。START 信号定义为在 SCL 为高电平的同时 SDA 发生高电平到低电平转换。此信号表示新数据传输的开始 (每次数据传输可能包含多个字节的数据)，所有从机都会离开空闲状态。

### 31.4.1.2 从机地址发送

紧接着 START 信号之后，数据传输的第一个字节是主机发送的从机地址。该地址包括一个 7 位调用地址和紧接着的一个 R/W 位。R/W 位告知从机数据传输的方向。

- 1 = 读传输：从机向主机发送数据
- 0 = 写传输：主机向从机发送数据

只有地址与主机发送的调用地址一致的从机才会响应，发送一个应答位。从机通过在第 9 个时钟周期拉低 SDA 来发送应答位。

系统中任意两个从机的地址不能相同。如果 I2C 模块是主机，则它不得发送一个与其自己的从地址相同的地址。I2C 模块不能既是主机，同时又是从机。然而，若仲裁在一个地址周期中丢失，I2C 模块将变回从机模式，即使被另一个主机寻址，它也能正常工作。

### 31.4.1.3 数据传输

成功实现从机寻址时，便可按照调用主机发送的 R/W 位所指定的方向，开始逐字节的数据传输。

所有遵循一个地址周期的传输都被称为数据传输，哪怕是携带从机子地址信息的传输。

每个数据字节为 8 位长。数据只能在 SCL 为低电平时改变。在 SCL 为高电平时，数据必须保持稳定。对每个数据位，SCL 有一个时钟脉冲；按照 MSB 优先顺序传输。每个数据字节之后是第 9 位（应答位），它由接收器件在第 9 个时钟脉冲拉低 SDA 而发出。总之，一个完整的数据传输需要 9 个时钟脉冲。

如果从机接收器未在第 9 位应答主机，从机必须不改变 SDA，让其保持高电平。主机将未能应答情况解释为数据传输不成功。

一个数据字节发送完毕后，如果主机接收器未应答从机发送器，从机将把该情况解释为数据传输结束，从而释放 SDA 线。

无论是从机还是主机未能应答，数据传输都会中止，并且主机执行以下两个操作之一：

- 产生 STOP 信号，释放总线。
- 产生重复 START 信号，开始新的调用。

### 31.4.1.4 停止信号

主机可通过生成停止信号释放总线的方式终止通信。停止信号定义为 SCL 的电平变为有效值时，SDA 从低电平转为高电平。

即使从机已生成了应答信号，主机仍可生成停止信号，此时从机必须释放总线。

### 31.4.1.5 重复开始信号

主机可能会在生成开始信号后，随即生成调用命令，而不会先生成停止信号。此操作称为重复开始。主机使用重复开始的方式并在不同的模式（发送/接收模式）下与另一台从机或同一台从机进行通信，而无需释放总线。

### 31.4.1.6 仲裁程序

I2C 总线是真正的多主机总线，支持连接多个主机。

若有两个或更多主机同时试图控制总线，则通过一个时钟同步程序来确定总线时钟。总线时钟的低电平周期等于主机中最长的时钟低电平周期，高电平周期等于最短的高电平周期。

竞争主机的相对极性由数据仲裁程序决定。如果一个总线主机发送逻辑电平 1，而另一个主机发送逻辑电平 0，则前者出局。出局的主机立即切换到从机接收模式，并且停止驱动 SDA 输出。这种情况下，从主机到从机的模式转换不产生 STOP 条件。与此同时，硬件设置一个状态位以指示其出局。

### 31.4.1.7 时钟同步

由于线“与”逻辑是在 SCL 上执行，因此 SCL 上的高到低转换会影响总线上的所有器件。器件开始计数其低电平周期，某个器件的时钟变为低电平后，该器件将使 SCL 保持低电平，直至时钟达到高电平状态。然而，如果另一器件的时钟仍在低电平周期内，该器件时钟的低到高变化可能不会改变 SCL 的状态。因此，同步时钟 SCL 保持低电平的时间取决于低电平周期最长的器件。低电平周期较短的器件会在该时间内进入高电平等待状态，参见下图。当所有相关的器件都已计数完其低电平周期时，同步时钟 SCL 被释放并变为高电平。此后，器件时钟与 SCL 状态之间即无差异，所有器件开始计数其高电平周期。第一个计数完高电平周期的器件再次将 SCL 拉低。

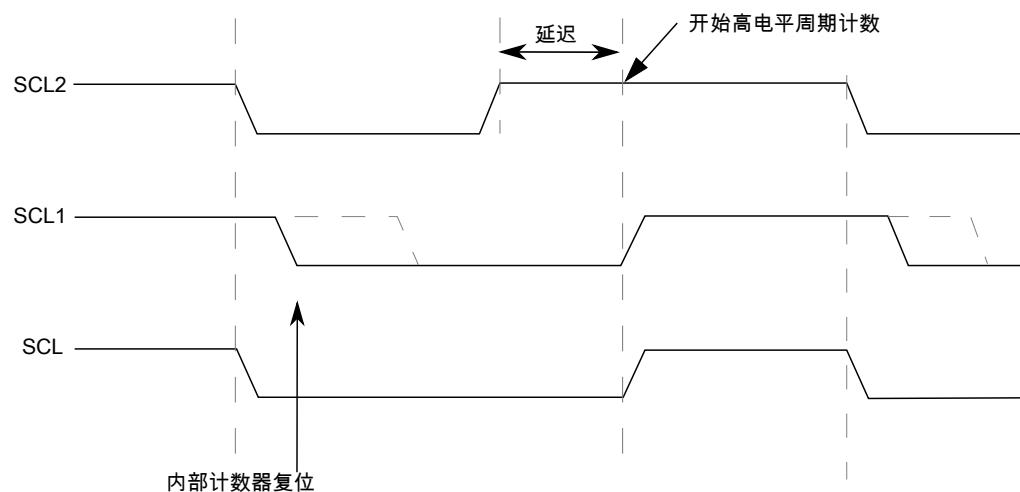


图 31-3. I2C 时钟同步

### 31.4.1.8 握手

时钟同步机制可用作数据传输中的握手。完成单字节传输（9位）之后，从机可能使 SCL 保持低电平。这种情况下，它会停止总线时钟，使主时钟强制进入等待状态，直至该从机释放 SCL。

### 31.4.1.9 时钟拉伸

从机可使用时钟同步机制降低传输比特率。主机将 SCL 驱动为低电平之后，从机可将 SCL 驱动为低电平，在经过所需的周期后将其释放。如果从机的 SCL 低电平周期大于主机的 SCL 低电平周期，则会拉伸产生的 SCL 总线信号的低电平周期。换言之，将增加 SCL 总线信号的低电平周期，使其与从机的 SCL 低电平周期相同。

### 31.4.1.10 I2C 分频器和保持值

#### 注

对于某些器件，在某些情况下，当 ICR 的值在 00h 到 0Fh 这一范围时，SCL 分频器的值可能存在±2 或±4 的差异。下表突出显示了这些可能存在差异的 SCL 分频器值。有关您的器件实际的 SCL 分频器值，请参见关于 I2C 模块的芯片特定详情。

表 31-2. I2C 分频器和保持值

ICR (十六进制)	SCL 分频器	SDA 保持值	SCL 保持(开始)值	SCL 保持(停止)值	ICR (十六进制)	SCL 分频器(时钟)	SDA 保持(时钟)	SCL 保持(开始)值	SCL 保持(停止)值
00	20	7	6	11	20	160	17	78	81
01	22	7	7	12	21	192	17	94	97
02	24	8	8	13	22	224	33	110	113
03	26	8	9	14	23	256	33	126	129
04	28	9	10	15	24	288	49	142	145
05	30	9	11	16	25	320	49	158	161
06	34	10	13	18	26	384	65	190	193
07	40	10	16	21	27	480	65	238	241
08	28	7	10	15	28	320	33	158	161
09	32	7	12	17	29	384	33	190	193
0A	36	9	14	19	2A	448	65	222	225
0B	40	9	16	21	2B	512	65	254	257
0C	44	11	18	23	2C	576	97	286	289
0D	48	11	20	25	2D	640	97	318	321

下一页继续介绍此表...

表 31-2. I2C 分频器和保持值 (继续)

ICR (十六进制)	SCL 分频器	SDA 保持值	SCL 保持(开始)值	SCL 保持(停止)值	ICR (十六进制)	SCL 分频器(时钟)	SDA 保持(时钟)	SCL 保持(开始)值	SCL 保持(停止)值
0E	56	13	24	29	2E	768	129	382	385
0F	68	13	30	35	2F	960	129	478	481
10	48	9	18	25	30	640	65	318	321
11	56	9	22	29	31	768	65	382	385
12	64	13	26	33	32	896	129	446	449
13	72	13	30	37	33	1024	129	510	513
14	80	17	34	41	34	1152	193	574	577
15	88	17	38	45	35	1280	193	638	641
16	104	21	46	53	36	1536	257	766	769
17	128	21	58	65	37	1920	257	958	961
18	80	9	38	41	38	1280	129	638	641
19	96	9	46	49	39	1536	129	766	769
1A	112	17	54	57	3A	1792	257	894	897
1B	128	17	62	65	3B	2048	257	1022	1025
1C	144	25	70	73	3C	2304	385	1150	1153
1D	160	25	78	81	3D	2560	385	1278	1281
1E	192	33	94	97	3E	3072	513	1534	1537
1F	240	33	118	121	3F	3840	513	1918	1921

## 31.4.2 10 位地址

在 10 位编址中，将对第一个地址字节的前 5 个位使用 0x11110。在包含 10 位编址的传输中，可以有各种组合形式的读/写格式。

### 31.4.2.1 主发送器对从接收器进行寻址

传输方向未更改。当 START 条件后跟 10 位地址时，各从机会将从地址 (11110XX) 第一个字节的前 7 位与其自己的地址作比较并测试第八位 ( $R/W$  方向位) 是否为 0。可能会有一个以上的器件出现匹配情况并生成应答(A1)。每个出现匹配情况的从机都会将从地址第二个字节的 8 位与其自己的地址作比较，但仅有一个从机出现匹配情况并生成应答(A2)。匹配的从机仍由主机寻址，直至它收到 STOP 条件(P)或后跟不同从地址的重复 START 条件(Sr)。

表 31-3. 主发送器对具有 10 位地址的从接收器进行寻址

S	从机地址 前 7 位 11110 + AD10 + AD9	R/W 0	A1	从机地址 第二个字 节 AD[8:1]	A2	数据	A	...	数据	A/A	P
---	---	-------	----	---------------------------	----	----	---	-----	----	-----	---

主发送器发送 10 位地址的第一个字节后，从接收器会发现 I2C 中断。对于此中断，用户软件必须确保忽略数据寄存器的内容，不将其视为有效数据。

### 31.4.2.2 主机-发送器对从机-接收器进行寻址

第二个 R/W 位之后，传输方向改变。截止应答位 A2 (包括该位) 的程序与针对主机发送器寻址从机接收器所述的程序相同。重复 START 条件(Sr)之后，匹配的从机记得它曾被寻址过。该从机随后检查 Sr 之后的从机地址的第一个字节的前 7 位是否与 START 条件(S)之后的情形相同，并测试第 8 位(R/W)是否为 1。如果一致，从机就会认为它已被寻址为发送器，并产生应答 A3。该从机发送器保持定址状态，直至收到 STOP 条件(P)或后跟其他从机地址的重复 START 条件(Sr)。

重复 START 条件(Sr)之后，所有其他从机也会将从机地址的第一个字节的前 7 位与其自己的地址相比较，并测试第 8 位(R/W)。然而，它们中的任何一个器件都不会被寻址，因为 R/W = 1 (对于 10 位器件)，或者 11110XX 从机地址 (对于 7 位器件) 不匹配。

表 31-4. 主机接收器用 10 位地址寻址从机发送器

S	从机地址 前 7 位 11110 + AD10 + AD9	R/W 0	A1	从机地址 第二个字 节 AD[8:1]	A2	Sr	从机地址 前 7 位 11110 + AD10 + AD9	R/W 1	A3	数据	A	...	数据	A	P
---	---	----------	----	------------------------------	----	----	---	----------	----	----	---	-----	----	---	---

主机接收器发送 10 位地址的第一个字节之后，从机发送器收到一个 I2C 中断。对于此中断，用户软件必须确保忽略数据寄存器的内容，不将其当作有效数据对待。

### 31.4.3 地址匹配

对于所有接收到的地址，都能够以 7 位或 10 位地址格式请求。

- 地址寄存器 1 (包含 I2C 第一从机地址) 中的 AD[7:1]始终参与地址匹配过程。它提供一个 7 位地址。
- 如果 ADEXT 位置位，控制寄存器 2 中的 AD[10:8]将参与地址匹配过程。它将 I2C 第一从机地址扩展为一个 10 位地址。

影响地址匹配的其他条件包括：

- 如果 GCAEN 位置位，通用广播地址将参与地址匹配过程。
- 如果 ALERTEN 位置位，警告地址将参与地址匹配过程。
- 如果 SIICAEN 位置位，地址寄存器 2 将参与地址匹配过程。
- 如果 RMEN 位置位，当范围地址寄存器编程为非零值时，地址寄存器 1（不包含）和范围地址寄存器（包含）的值范围内的任何地址都将参与地址匹配过程。范围地址寄存器的值必须编程为大于地址寄存器 1 的值。

I2C 模块响应这些地址中的任一地址时，将充当从机接收器，并且 IAAS 位将在地址周期结束后置位。软件必须在第一个字节传输后读取数据寄存器，以确定地址已匹配。

### 31.4.4 系统管理总线规范

SMBus 提供系统和电源管理相关任务的控制总线。系统可使用 SMBus 向设备发送报文和从设备接收报文，而不是跳变单个控制线路。

取消单个控制线路可减少引脚数量。接受报文可确保未来的可扩展性。使用系统管理总线，设备可实现以下功能：提供制造商信息、告知系统其型号/设备编号、保存挂起事件的状态、报告不同类型的错误、接受控制参数、返回其状态。

#### 31.4.4.1 定时溢出

根据  $T_{TIMEOUT,MIN}$  参数，允许主机或者从机结束故障器件无限的拉低电平或者主机无限尝试抢占总线。从机检测到任意单个时钟保持在低电平的时间长于  $T_{TIMEOUT,MIN}$  时，必须释放总线（停止驱动总线并使 SCL 和 SDA 悬空为高电平）。已检测到此条件的器件必须复位其通信并且能在  $TIMEOUT,MAX$  的时间范围内接收新的 START 条件。

SMBus 定义 35 ms 的时钟低电平定时溢出  $T_{TIMEOUT}$ ，将  $T_{LOW:SEXT}$  指定为从机的累积时钟低电平延长时间，将  $T_{LOW:MEXT}$  指定为主机的累积时钟低电平延长时间。

##### 31.4.4.1.1 SCL 低电平定时溢出

如果 SCL 线路通过总线上的从机保持为低电平，则无法开展进一步通信。此外，主机无法将 SCL 线路强制变为高电平以纠正该错误状况。为了解决这个问题，SMBus 协议要求参与传输的器件必须检测时钟周期保持低电平的时间是否超过某一定时溢出值。检测到定时溢出状况的器件必须复位通信。当 I2C 模块是有效主机时，若它检测到 SMBCLK 低电平时间已超过  $T_{TIMEOUT,MIN}$  的值，它必须在传输过程的当前数据字节内或之后产生停止条件。当 I2C 模块是从机时，若它检测到  $T_{TIMEOUT,MIN}$  状况，它将复位通信，然后便能接收新的起始(START)条件。

### 31.4.4.1.2 SCL 高电平定时溢出

I2C 模块确定 SMBCLK 和 SMBDAT 信号已处于高电平状态至少  $T_{HIGH:MAX}$  时间后，它假设总线处于闲置状态。

当总线上出现 START 条件，并且 STOP 条件未出现前，高电平定时溢出会出现。满足下列任一条件时，任何检测到该情况的主机都可假设总线处于空闲状态。

- SHTF1 上升。
- BUSY 位处于高电平状态，SHTF1 也处于高电平状态。

当有一段时间 SMBDAT 信号处于低电平状态且 SMBCLK 信号处于高电平状态时，会发生另一种定时溢出。必须在软件中定义该时间周期。SHTF2 在已达到时间限制的情况下作为标志。该标志也是一个中断资源，因此它会触发 IICIF。

### 31.4.4.1.3 CSMBCLK TIMEOUT MEXT 和 CSMBCLK TIMEOUT SEXT

下图说明定时溢出间隔  $T_{LOW:SEXT}$  和  $T_{LOW:MEXT}$  的定义。在主机模式下，在一个字节之内，I2C 模块时钟周期的累计延长时间不得超过  $T_{LOW:MEXT}$ ；各字节定义为 START 至 ACK、ACK 至 ACK 或 ACK 至 STOP。发生 CSMBCLK TIMEOUT MEXT 时，SMBus MEXT 上升，同时会触发 SLTF。

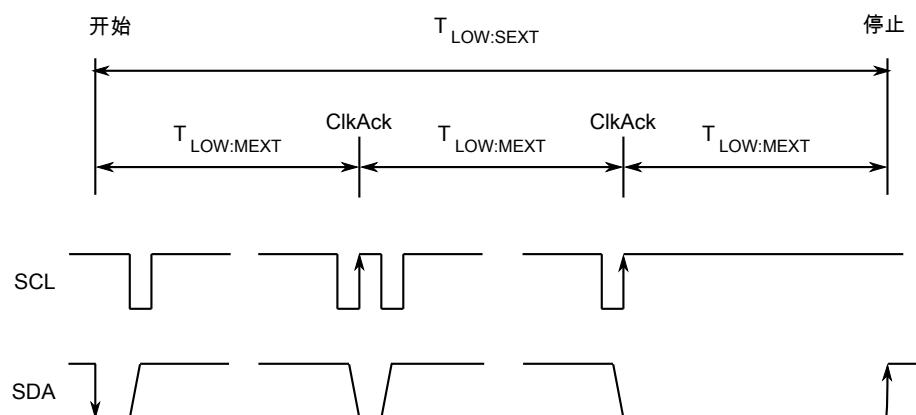


图 31-4. 定时溢出测量间隔

主机可中止正在进行的任何违反  $T_{LOW:SEXT}$  或  $T_{TIMEOUT,MIN}$  要求的从机的处理。要中止处理，主机应在当前字节传输结束时发送 STOP 条件。作为从机，在从初始 START 到 STOP 的任何报文传输期间，I2C 模块时钟周期的累计延长时间不得超过  $T_{LOW:SEXT}$ 。发生 CSMBCLK TIMEOUT SEXT 时，SEXT 上升，同时会触发 SLTF。

#### 注

CSMBCLK TIMEOUT SEXT 和 CSMBCLK TIMEOUT MEXT 是在第二步中实现的可选功能。

### 31.4.4.2 FAST ACK 和 NACK

为了提高可靠性和通信的稳定性，SMBus 设备是否实现数据包差错校验(PEC)是可选的，但对于参与地址解析协议(ARP)过程的器件，PEC 则是必需的，并且仅在该过程中需要。PEC 是一个基于所有报文字节的 CRC-8 错误校验字节。PEC 由提供最后一个数据字节的器件追加到报文中。如果 PEC 存在但不正确，接收器将发送 NACK，否则发送 ACK。为了通过软件计算 CRC-8，在接收到第 8 个 SCL (第 8 位) 之后，如果该字节是数据字节，此模块将使 SCL 线保持低电平。这样，软件就能确定应当向总线发送 ACK 还是 NACK，即置位还是清除 TXAK 位 (如果快速 ACK/NACK 使能位 FACK 已使能)。

SMBus 要求设备始终应答其自己的地址，作为检测总线上可移除设备 (如电池或插接站) 存在与否的机制。除了指示从设备繁忙条件以外，SMBus 还使用 NACK 机制来表示接收到无效命令或数据。由于这种条件可能出现在最后一个字节时，因此 SMBus 设备必须有能力在传输每个字节之后及完成处理之前产生不应答信号。因为 SMBus 不提供其他重新发送命令，所以这一要求非常重要。使用 NACK 信令中的这一差别对 SMBus 端口的具体实现有影响，尤其是对于 SMBus 主机和 SBS 器件等处理关键系统数据的设备。

#### 注

在主机接收从机发送模式的最后一个字节中，主机必须向总线发送一个 NACK，因此在发送最后一个字节之前必须关闭 FACK。

### 31.4.5 复位

复位后，I2C 模块禁用。I2C 模块无法引发内核复位。

### 31.4.6 中断

假定 IICIE 位置位，当发生此处表格中的任一事件时，I2C 模块都会生成中断。

中断由 (I2C 状态寄存器的) IICIF 位驱动，通过 (I2C 控制寄存器 1 的) IICIE 位屏蔽。在中断程序中，必须通过向 IICIF 位写入 1 的方式由软件清零 IICIF 位。SMBus 定时溢出中断由 SLTF 驱动并通过 IICIE 位屏蔽。在中断程序中，必须通过向 SLTF 位写入 1 的方式由软件清零 SLTF 位。可通过读状态寄存器进行操作确定中断类型。

#### 注

在主机接收模式下，最后一个字节传输之前，FACK 位必须设为零。

表 31-5. 中断汇总

中断源	状态	标志	本地使能
完成 1 个字节的传输	TCF	IICIF	IICIE
已接收的调用地址与主/从机地址匹配	IAAS	IICIF	IICIE
仲裁丢失	ARBL	IICIF	IICIE
I <sup>2</sup> C 总线停止检测	STOPF	IICIF	IICIE & SSIE
I <sup>2</sup> C 总线起始位检测	STARTF	IICIF	IICIE & SSIE
SMBus SCL 低电平定时溢出	SLTF	IICIF	IICIE
SMBus SCL 高电平 SDA 低电平定时溢出	SHTF2	IICIF	IICIE & SHTF2IE
从 Stop 或 Wait 模式中唤醒	IAAS	IICIF	IICIE 和 WUEN

### 31.4.6.1 字节传输中断

传输完成标志(TCF)位在第九个时钟的下降沿处置位，以表示字节和应答传输完成。启用 FACK 后，TCF 在第八个时钟的下降沿处置位，以表示字节完成。

### 31.4.6.2 地址检测中断

当调用地址与所设置的从机地址 (I2C 地址寄存器) 一致时，或者当 GCAEN 位置位且收到广播时，状态寄存器的 IAAS 位置位。如果 IICIE 位置位，则 CPU 中断。CPU 必须检查 SRW 位，并相应地设置其 Tx 模式。

### 31.4.6.3 停止检测中断

当在 I<sup>2</sup>C 总线上检测到停止状态时，SSIE 位置 1。如果 IICIE 和 SSIE 位均置 1，则会中断 CPU。

### 31.4.6.4 退出低功耗/Stop 模式

在低功耗模式 (Wait 和 Stop) 下，从机接收输入检测电路和地址匹配功能仍然有效。如果中断未被屏蔽，与从机地址或广播地址匹配的异步输入将使 CPU 离开低功耗/Stop 模式。因此，TCF 和 IAAS 均可触发该中断。

### 31.4.6.5 仲裁丢失中断

I2C 模块不能既是主器件，同时又是从器件。如果两个或两个以上主机要同时控制总线，则进行竞争的主机的相对极性由数据仲裁程序决定。I2C 模块在其丢失数据仲裁进程且状态寄存器的 ARBL 位置位时会使仲裁-丢失中断的电平变为有效。

仲裁在下列情形时丢失：

1. 在地址或数据发送周期内，主机驱动为高电平时以低电平采样 SDA。
2. 在应答位的数据接收周期内，主机驱动为高电平时以低电平采样 SDA。
3. 总线繁忙时尝试采用 START 周期。
4. 从机模式请求重复的 START 周期。
5. 主机未请求该周期时检测到 STOP 条件。

必须通过将 1 写入 ARBL 位由软件将此位清零。

### 31.4.6.6 SMBus 中的定时溢出中断

当 IICIE 位置位时，一旦检测到上述任一定时溢出条件，I2C 模块就会将定时溢出中断的电平变为有效值(输出 SLTF 和 SHTF2)，但有一个例外。SCL 高电平和 SDA 高电平 TIMEOUT 机制不得用来影响定时溢出中断输出，因为该定时溢出指示总线空闲条件。当与 SCL 高电平和 SDA 高电平 TIMEOUT 匹配时，SHTF1 上升，并自动下降以指示总线状态。SHTF2 的定时溢出周期与 SHTF1 相同，比 SLTF 要短，因此增加了另一个控制位 SHTF2IE 来使能或禁用它。

### 31.4.7 可编程输入去抖滤波器

I2C 去抖滤波器已添加到传统 I2C 模块的外部，而非 I2C 封装内。该过滤器可吸附 I2C 时钟和 I2C 模块数据线路上的毛刺。

可根据（半）I2C 模块时钟周期数指定待吸收毛刺的宽度。提供单个可编程输入去抖滤波器控制寄存器。实际上，I2C 模块通常会忽略数据线路上的任意下降-上升-下降或上升-下降-上升转换（在该寄存器编程的时钟周期数范围内）。程序员必须指定毛刺的大小（根据 I2C 模块时钟周期），以便过滤器吸收毛刺，并且阻止毛刺通过。

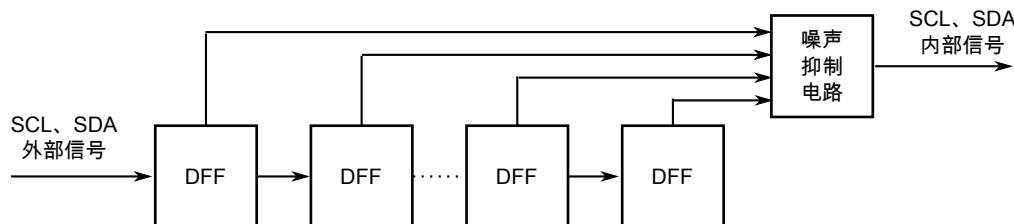


图 31-5. 可编程输入去抖滤波器示意图

### 31.4.8 地址匹配唤醒

当 I2C 模块处于从机接收模式时，如果发生主要、范围、或通用调用地址匹配，则 MCU 从低功耗模式（此时所有外设总线均处于非运行状态）唤醒。

置位地址匹配 IAAS 位后，地址匹配结束时，发送一条中断指令以唤醒内核。

#### 注

唤醒过程中，如果某个外部主机继续向从机发送数据，则 Stop 模式下的波特率必须低于 50 kbit/s。为避免在 Stop 模式下出现速度较低的波特率，主机可在固件中增加一个短时延迟，直至唤醒过程完成，然后再发送数据。

#### 注

SMBus 模式不支持地址匹配引起的唤醒。

## 31.5 初始化/应用信息

### 模块初始化（从机）

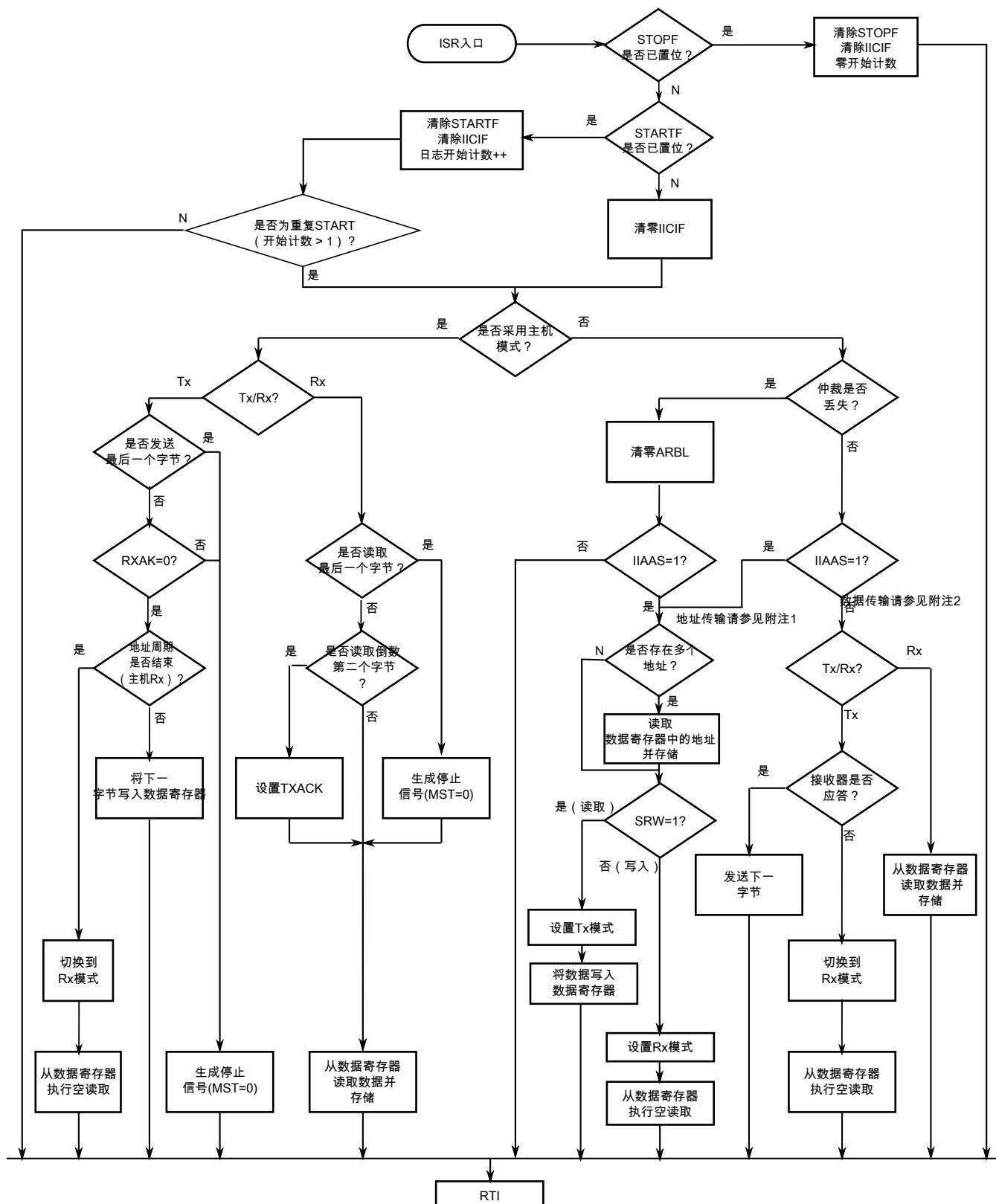
1. 写入：控制寄存器 2
  - 使能或禁用通用调用命令
  - 选择 10 位或 7 位编址模式

2. 写入：地址寄存器 1，以置位从机地址
3. 写入：控制寄存器 1，以使能 I2C 模块和中断
4. 初始化发送数据的 RMA 变量 ( $IICEN = 1$  和  $IICIE = 1$ )
5. 初始化用于达到下图所示程序的 RAM 变量

### 模块初始化（主机）

1. 写入：分频器寄存器，以置位 I2C 波特率（参见 [ICR](#) 说明中的示例）
2. 写入：控制寄存器 1，以使能 I2C 模块和中断
3. 初始化发送数据的 RMA 变量 ( $IICEN = 1$  和  $IICIE = 1$ )
4. 初始化用于实现下图所示程序的 RAM 变量
5. 写入：控制寄存器 1，以使能 TX
6. 写入：控制寄存器 1，以使能 MST（主机模式）
7. 写入：数据寄存器和目标从机地址（该字节的 LSB 确定是主机接收还是发送通信）

下图程序显示了主机和从机 I2C 操作。对于从机操作，包含正确地址的 I2C 传入报文开始 I2C 通信。对于主机操作，通信初始化必须通过写入数据寄存器的方式完成。可在 [AN4342: ColdFire+和 Kinetis I2C 的使用](#) 中查看执行此处所述许多步骤的 I2C 驱动器示例。



注释：

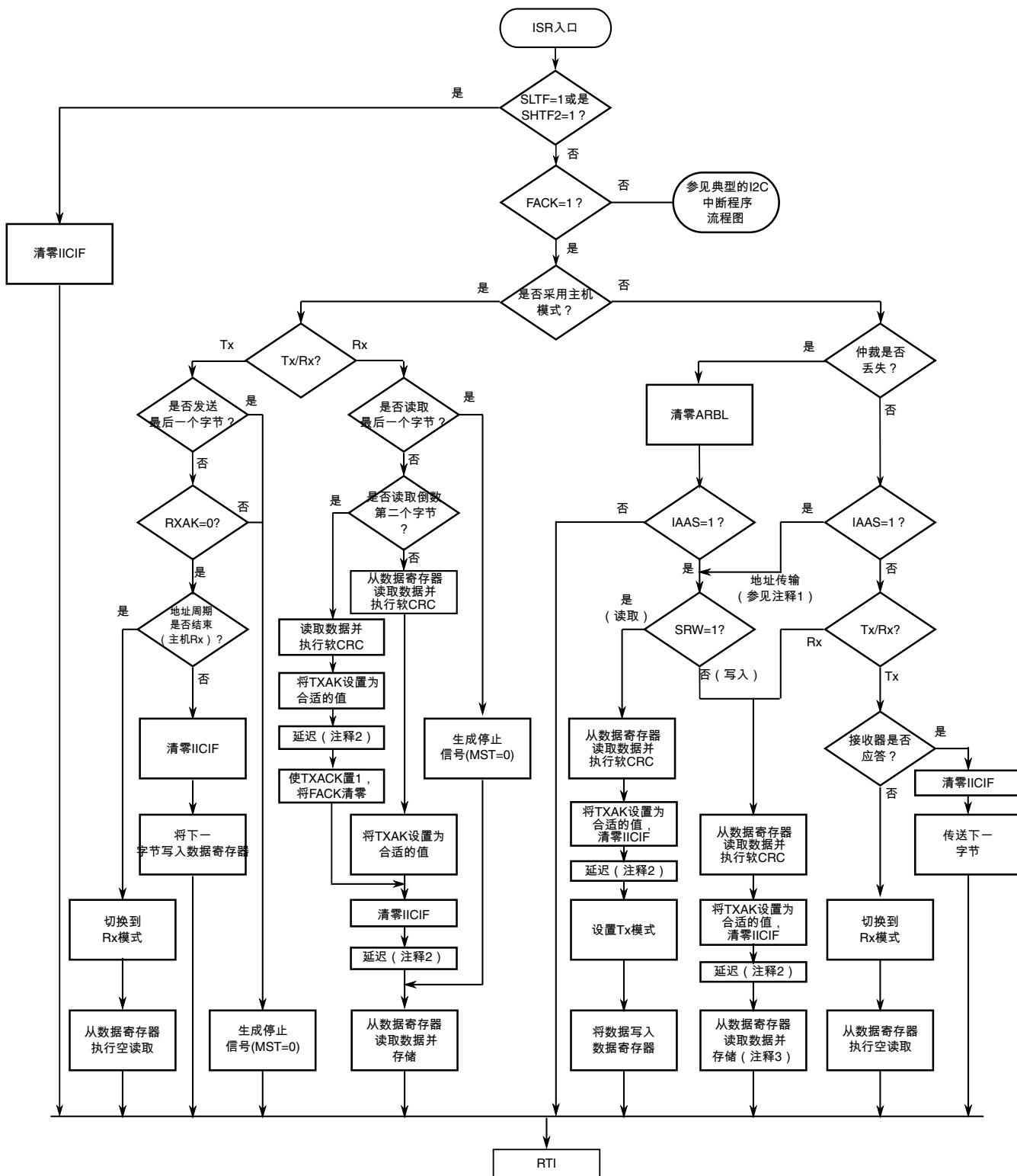
1. 如果已使能通用调用，则检查以确定接收到的地址是否是通用调用地址(0x00)。

如果接收到的地址是通用调用地址，则通用调用必须由用户软件处理。

2. 如果10位寻址进行的是从机寻址，则在扩展地址的首个字节后，该从机会遭遇中断。

对此中断，必须确保忽略数据寄存器的内容，且不将其当作有效数据传输。

图 31-6. 典型 I2C 中断程序



注释：

- 如果已使能通用调用或SIICAEN，则进行检查，以确定接收到的地址是通用调用地址(0x00)还是SMBus器件默认地址。无论是哪一种，都必须由用户软件来处理。
- 在接收模式下，在第一个和第二个数据读取之前，可能需要一个位时间的延迟，以等待第9个SCL周期可能的最长时间周期（最差的情况下）。
- 这是一个空读取，是为了复位SMBus接收器状态机。

图 31-7. 典型 I2C SMBus 中断程序

# 第 32 章

## MSCAN

### 32.1 简介

MSCAN 是一款实施 Bosch 规范（1991 年 9 月）所定义的 CAN 2.0A/B 协议的通信控制器。为方便用户全面理解 MSCAN 规范，建议首先阅读 Bosch 规范，以便熟悉本文件所含的术语和概念。

CAN 协议尽管并非专为汽车应用而设计，但它满足车辆串行数据总线的特定要求：实时处理、在车辆的 EMI 环境中可靠工作、高性价比以及带宽要求。

MSCAN 采用先进的缓冲区排列来实现可预测的实时特性并简化应用软件。

#### 32.1.1 术语表

表 32-1. 术语

术语	说明
ACK	对 CAN 报文的应答
CAN	控制器局域网
CRC	循环冗余校验码
EOF	帧结束
FIFO	先进先出存储器
IFS	帧间序列
SOF	帧开始
CPU 总线	CPU 相关的读/写数据总线
CAN 总线	CAN 协议相关的串行总线
振荡器时钟	外部振荡器直接提供的时钟
总线时钟	CPU 总线相关的时钟
CAN 时钟	CAN 协议相关的时钟

### 32.1.2 结构框图

下图是 MSCAN 的结构框图

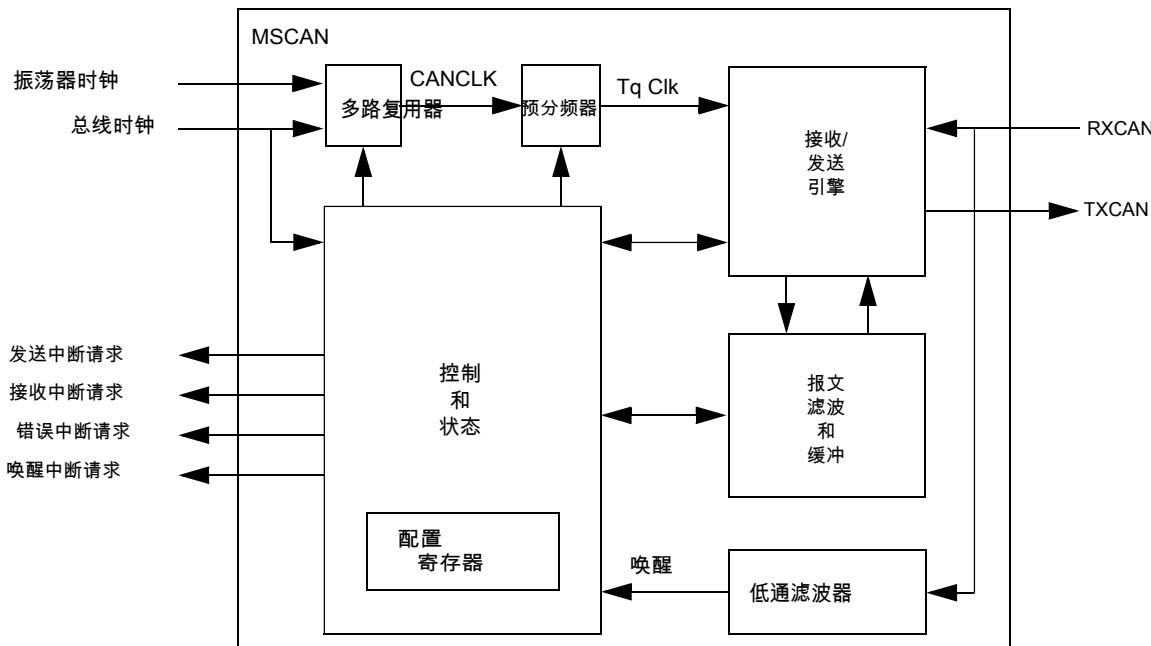


图 32-1. MSCAN 结构框图

### 32.1.3 特性

MSCAN 具有如下基本特性：

- 实施了 CAN 协议 - 2.0A/B 版
  - 标准和扩展数据帧
  - 0 到 8 字节的数据长度
  - 最高 1 Mbps 的可编程比特率 <sup>1</sup>
  - 支持远程帧
- 采用 FIFO 存储方案的五个接收缓冲区
- 3 个发送缓冲区，采用“本地优先级”概念进行内部排序
- 灵活的可屏蔽标识符滤波器支持两个全尺寸（32 位）扩展的标识符滤波器，或者四个 16 位滤波器，或者八个 8 位滤波器
- 带集成式低通滤波器的可编程唤醒功能
- 可编程回环模式支持自检操作
- 用于监控 CAN 总线的可编程仅监听模式
- 可编程总线关闭恢复功能
- 所有的 CAN 接收器和发送器错误状态（警告、错误被动状态、总线关闭）都有独立的信号和中断功能

1. 取决于实际位定时和 PLL 的时钟抖动

- 可编程 MSCAN 时钟源，总线时钟或振荡器时钟
- 内部定时器用于已接收和已发送报文的时间标志
- 三种低功耗模式：睡眠、关机和 MSCAN 使能
- 配置寄存器的全局初始化

## 32.1.4 工作模式

### 32.1.4.1 系统正常工作模式

MSCAN 模块在所有系统正常工作模式下都按照本规范所述方式工作。某些寄存器存在写操作限制。

### 32.1.4.2 系统特殊工作模式

MSCAN 模块在所有系统特殊工作模式下都按照本规范所述方式工作。为了在特殊模式下进行测试，解除了正常模式下在特定寄存器上存在的写操作限制。

### 32.1.4.3 仿真模式

在所有仿真模式下，MSCAN 模块就像在本规范所述系统正常工作模式下一样工作。

### 32.1.4.4 仅监听模式

在可选的 CAN 总线监控模式（仅监听）下，CAN 节点能够接收有效数据帧和有效远程帧，但它在 CAN 总线上只发送“隐性”位。此外，它不能启动发送。

如果要求 MAC 子层发送一个“显性”位（ACK 位、过载标志或主动错误标记），那么在内部会按新路线重新发送该位，以便 MAC 子层监控该“显性”位，不过 CAN 总线在外部可能仍然处于隐性状态。

### 32.1.4.5 MSCAN 初始化模式

MSCAN 在使能(CANE=1)时进入初始化模式。

在工作期间进入初始化模式时，任何正在进行的发送或接收都会立即中止，与 CAN 总线的同步丢失，从而可能导致违反 CAN 协议的情况发生。为防止因违规而使 CAN 总线系统遭受严重影响，MSCAN 会立即驱动 TXCAN，使其进入隐性状态。

**注**

进入初始化模式时，用户有责任确保 MSCAN 不处于激活状态。建议流程是先让 MSCAN 进入睡眠模式 ( $SLPRQ = 1$  且  $SLPAK = 1$ )，然后再在 CANCTL0 寄存器中置位 INITRQ 位。否则，中止正在传输的报文可能会导致出错并影响其他 CAN 总线器件。

初始化模式下，MSCAN 停止工作。不过，接口寄存器仍然处于可访问状态。该模式用于将 CANCTL0、CANRFLG、CANRIER、CANTFLG、CANTIER、CANTARQ、CANTAACK 和 CANTBSEL 寄存器复位到默认值。此外，MSCAN 支持对 CANBTR0、CANBTR1 位定时寄存器、CANIDAC 以及 CANIDAR、CANIDMR 报文滤波器的配置。

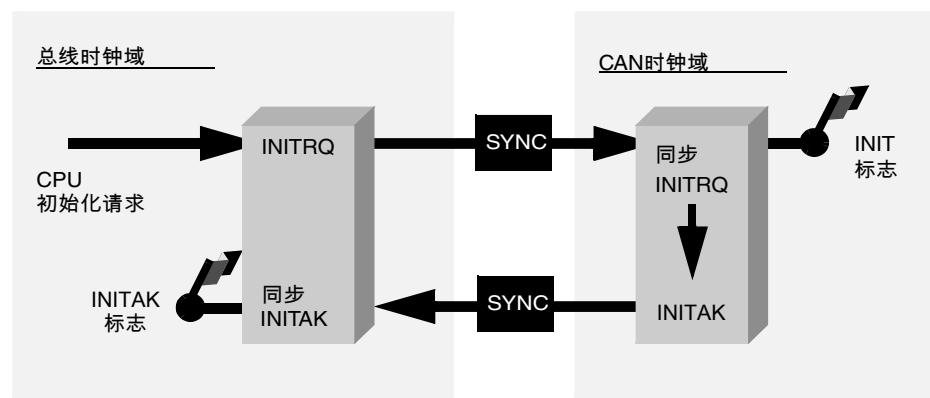


图 32-2. 初始化请求/应答周期

由于 MSCAN 内存在独立时钟域，必须采用特殊的握手机制使 INITRQ 与所有域都同步。该握手会引起额外的同步延迟（参见图 32-2）。

如果 CAN 总线上没有进行任何报文传输，那么最短延迟将是再增加 2 个总线时钟和 3 个 CAN 时钟。当 MSCAN 的所有部分都处于初始化模式时，INITAK 标志置位。应用软件必须将 INITAK 用作请求(INITRQ)进入初始化模式的握手指示。

**注**

在初始化模式 ( $INITRQ = 1$  且  $INITAK = 1$ ) 有效之前，CPU 无法清除 INITRQ。

## 32.2 外部信号说明

MSCAN 使用两个外部引脚。

**注**

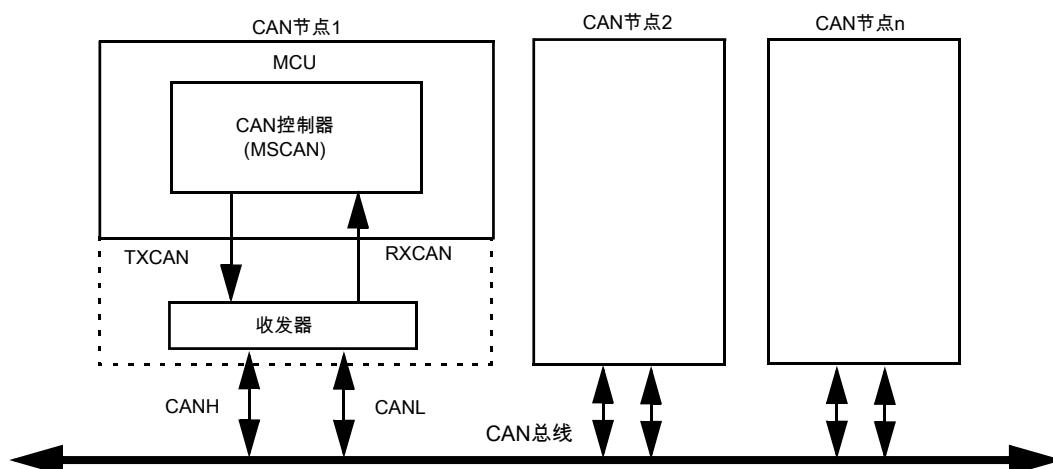
在集成了 CAN 物理接口（收发器）的 MCU 上，MSCAN 接口内部连接到收发器接口。在这类情况下，信号 TXCAN 和 RXCAN 在外部是否可用是可选的。

**表 32-2. MSCAN 信号说明**

信号	说明	I/O	功能
RXCAN	CAN 接收器输入引脚	I	-
TXCAN	CAN 发送器输出引脚	O	TXCAN 输出引脚代表 CAN 总线的逻辑电平： • 0 = 显性状态 • 1 = 隐性状态

### 32.2.1 CAN 系统

采用 MSCAN 的典型 CAN 系统如下图所示。各 CAN 站通过收发器设备物理连接到 CAN 总线线路。收发器能够驱动 CAN 总线所需的大电流，并具有电流保护功能，可免受有缺陷的 CAN 或 CAN 站影响。



**图 32-3. CAN 系统**

## 32.3 存储器映像和寄存器定义

### 32.3.1 程序员报文存储模型

每个接收和发送报文缓冲区在存储器映像中分配 16 个字节，其中包含一个 13 字节的数据结构。

另外还为发送缓冲区定义了一个发送缓冲区优先级寄存器(TBPR)。在该存储器映像的最后两个字节中，MSCAN 存储一个特殊的 16 位时间标志，它是在成功发送或接收报文后从一个内部定时器采样得到的。该特性仅可用于发送和接收缓冲区，并且要求 CANCTL0[TIME]置位。

时间标志寄存器的写操作由 MSCAN 执行。CPU 只能对这些寄存器进行读操作。

下表显示了扩展标识符和标准标识符的接收和发送缓冲区数据结构所映射到的寄存器。

**表 32-3. 报文缓冲区结构**

偏移地址	扩展标识符	标准标识符
0x0020	REIDR0	RSIDR0
0x0021	REIDR1	RSIDR1
0x0022	REIDR2	
0x0023	REIDR3	
0x0024	REDSR0	
0x0025	REDSR1	
0x0026	REDSR2	
0x0027	REDSR3	
0x0028	REDSR4	
0x0029	REDSR5	
0x002A	REDSR6	
0x002B	REDSR7	
0x002C	RDLR	
0x0030	TEIDR0	TSIDR0
0x0031	TEIDR1	TSIDR1
0x0032	TEIDR2	
0x0033	TEIDR3	
0x0034	TEDSR0	
0x0035	TEDSR1	
0x0036	TEDSR2	
0x0037	TEDSR3	
0x0038	TEDSR4	
0x0039	TEDSR5	
0x003A	TEDSR6	
0x003B	TEDSR7	
0x003C	TDLR	

## 注

读操作：

- 对于发送缓冲区，当 CANTFLG[TXE]标志置位且在 CANTBSEL 中选中对应的发送缓冲区时，可以随时执行。
- 对于接收缓冲区，仅当 CANRFLG[RXF]标志置位时才能执行。

写操作：

- 对于发送缓冲区，当 CANTFLG[TXE]标志置位且在 CANTBSEL 中选中对应的发送缓冲区时，可以随时执行。
- 对于接收缓冲区，未实施。

复位：因基于 RAM 实施而未定义

### MSCAN 存储器映射

绝对地址（十六进制）	寄存器名称	宽度（单位：位）	访问	复位值	小节/页
4002_4000	MSCAN 控制寄存器 0 (MSCAN_CANCTL0)	8	R/W	01h	<a href="#">32.3.2/558</a>
4002_4001	MSCAN 控制寄存器 1 (MSCAN_CANCTL1)	8	R/W	11h	<a href="#">32.3.3/560</a>
4002_4002	MSCAN 总线定时寄存器 0 (MSCAN_CANBTR0)	8	R/W	00h	<a href="#">32.3.4/561</a>
4002_4003	MSCAN 总线定时寄存器 1 (MSCAN_CANBTR1)	8	R/W	00h	<a href="#">32.3.5/562</a>
4002_4004	MSCAN 接收器标志寄存器 (MSCAN_CANRFLG)	8	R/W	00h	<a href="#">32.3.6/563</a>
4002_4005	MSCAN 接收器中断使能寄存器 (MSCAN_CANRIER)	8	R/W	00h	<a href="#">32.3.7/565</a>
4002_4006	MSCAN 发送器标志寄存器 (MSCAN_CANTFLG)	8	R/W	07h	<a href="#">32.3.8/566</a>
4002_4007	MSCAN 发送器中断使能寄存器 (MSCAN_CANTIER)	8	R/W	00h	<a href="#">32.3.9/567</a>
4002_4008	MSCAN 发送器报文中止请求寄存器 (MSCAN_CANTARQ)	8	R/W	00h	<a href="#">32.3.10/568</a>
4002_4009	MSCAN 发送器报文中止应答寄存器 (MSCAN_CANTAAK)	8	R	00h	<a href="#">32.3.11/568</a>
4002_400A	MSCAN 发送缓冲区选择寄存器 (MSCAN_CANTBSEL)	8	R/W	00h	<a href="#">32.3.12/569</a>
4002_400B	MSCAN 标识符验收控制寄存器 (MSCAN_CANIDAC)	8	R/W	00h	<a href="#">32.3.13/570</a>
4002_400D	MSCAN 其他寄存器 (MSCAN_CANMISC)	8	R/W	00h	<a href="#">32.3.14/571</a>
4002_400E	MSCAN 接收错误计数器 (MSCAN_CANRXERR)	8	R	00h	<a href="#">32.3.15/572</a>
4002_400F	MSCAN 发送错误计数器 (MSCAN_CANTXERR)	8	R	00h	<a href="#">32.3.16/572</a>
4002_4010	第一群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDAR0)	8	R/W	00h	<a href="#">32.3.17/573</a>
4002_4011	第一群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDAR1)	8	R/W	00h	<a href="#">32.3.17/573</a>
4002_4012	第一群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDAR2)	8	R/W	00h	<a href="#">32.3.17/573</a>

下一页继续介绍此表...

## MSCAN 存储器映射 (继续)

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4002_4013	第一群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDAR3)	8	R/W	00h	32.3.17/ 573
4002_4014	第一群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMR0)	8	R/W	00h	32.3.18/ 574
4002_4015	第一群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMR1)	8	R/W	00h	32.3.18/ 574
4002_4016	第一群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMR2)	8	R/W	00h	32.3.18/ 574
4002_4017	第一群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMR3)	8	R/W	00h	32.3.18/ 574
4002_4018	第二群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDAR4)	8	R/W	00h	32.3.19/ 574
4002_4019	第二群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDAR5)	8	R/W	00h	32.3.19/ 574
4002_401A	第二群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDAR6)	8	R/W	00h	32.3.19/ 574
4002_401B	第二群组中的 MSCAN 标识符验收寄存器 n (MSCAN_CANIDAR7)	8	R/W	00h	32.3.19/ 574
4002_401C	第二群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMR4)	8	R/W	00h	32.3.20/ 575
4002_401D	第二群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMR5)	8	R/W	00h	32.3.20/ 575
4002_401E	第二群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMR6)	8	R/W	00h	32.3.20/ 575
4002_401F	第二群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN_CANIDMR7)	8	R/W	00h	32.3.20/ 575
4002_4020	接收扩展标识符寄存器 0 (MSCAN_REIDR0)	8	R/W	未定义	32.3.21/ 576
4002_4020	接收标准标识符寄存器 0 (MSCAN_RSIDR0)	8	R/W	未定义	32.3.22/ 576
4002_4021	接收扩展标识符寄存器 1 (MSCAN_REIDR1)	8	R/W	未定义	32.3.23/ 577
4002_4021	接收标准标识符寄存器 1 (MSCAN_RSIDR1)	8	R/W	未定义	32.3.24/ 578
4002_4022	接收扩展标识符寄存器 2 (MSCAN_REIDR2)	8	R/W	未定义	32.3.25/ 579
4002_4023	接收扩展标识符寄存器 3 (MSCAN_REIDR3)	8	R/W	未定义	32.3.26/ 579
4002_4024	接收扩展数据段寄存器 N (MSCAN_REDSTR0)	8	R/W	未定义	32.3.27/ 580
4002_4025	接收扩展数据段寄存器 N (MSCAN_REDSTR1)	8	R/W	未定义	32.3.27/ 580
4002_4026	接收扩展数据段寄存器 N (MSCAN_REDSTR2)	8	R/W	未定义	32.3.27/ 580

下一页继续介绍此表...

## MSCAN 存储器映射 (继续)

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4002_4027	接收扩展数据段寄存器 N (MSCAN_REDLSR3)	8	R/W	未定义	32.3.27/ 580
4002_4028	接收扩展数据段寄存器 N (MSCAN_REDLSR4)	8	R/W	未定义	32.3.27/ 580
4002_4029	接收扩展数据段寄存器 N (MSCAN_REDLSR5)	8	R/W	未定义	32.3.27/ 580
4002_402A	接收扩展数据段寄存器 N (MSCAN_REDLSR6)	8	R/W	未定义	32.3.27/ 580
4002_402B	接收扩展数据段寄存器 N (MSCAN_REDLSR7)	8	R/W	未定义	32.3.27/ 580
4002_402C	接收数据长度寄存器 (MSCAN_RDLR)	8	R/W	未定义	32.3.28/ 580
4002_402E	接收时间标志寄存器高位 (MSCAN_RTSRH)	8	R	未定义	32.3.29/ 581
4002_402F	接收时间标志寄存器低位 (MSCAN_RTSRL)	8	R	未定义	32.3.30/ 581
4002_4030	发送扩展标识符寄存器 0 (MSCAN_TEIDR0)	8	R/W	未定义	32.3.31/ 582
4002_4030	发送标准标识符寄存器 0 (MSCAN_TSIDR0)	8	R/W	未定义	32.3.32/ 583
4002_4031	发送扩展标识符寄存器 1 (MSCAN_TEIDR1)	8	R/W	未定义	32.3.33/ 583
4002_4031	发送标准标识符寄存器 1 (MSCAN_TSIDR1)	8	R/W	未定义	32.3.34/ 584
4002_4032	发送扩展标识符寄存器 2 (MSCAN_TEIDR2)	8	R/W	未定义	32.3.35/ 585
4002_4033	发送扩展标识符寄存器 3 (MSCAN_TEIDR3)	8	R/W	未定义	32.3.36/ 585
4002_4034	发送扩展数据段寄存器 N (MSCAN_TEDSR0)	8	R/W	未定义	32.3.37/ 586
4002_4035	发送扩展数据段寄存器 N (MSCAN_TEDSR1)	8	R/W	未定义	32.3.37/ 586
4002_4036	发送扩展数据段寄存器 N (MSCAN_TEDSR2)	8	R/W	未定义	32.3.37/ 586
4002_4037	发送扩展数据段寄存器 N (MSCAN_TEDSR3)	8	R/W	未定义	32.3.37/ 586
4002_4038	发送扩展数据段寄存器 N (MSCAN_TEDSR4)	8	R/W	未定义	32.3.37/ 586
4002_4039	发送扩展数据段寄存器 N (MSCAN_TEDSR5)	8	R/W	未定义	32.3.37/ 586
4002_403A	发送扩展数据段寄存器 N (MSCAN_TEDSR6)	8	R/W	未定义	32.3.37/ 586
4002_403B	发送扩展数据段寄存器 N (MSCAN_TEDSR7)	8	R/W	未定义	32.3.37/ 586

下一页继续介绍此表...

## MSCAN 存储器映射 (继续)

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
4002_403C	发送数据长度寄存器 (MSCAN_TDLR)	8	R/W	未定义	32.3.38/ 586
4002_403D	发送缓冲区优先级寄存器 (MSCAN_TBPR)	8	R/W	未定义	32.3.39/ 587
4002_403E	发送时间标志寄存器高位 (MSCAN_TTSRH)	8	R	未定义	32.3.40/ 588
4002_403F	发送时间标志寄存器低位 (MSCAN_TTSRL)	8	R	未定义	32.3.41/ 588

### 32.3.2 MSCAN 控制寄存器 0 (MSCAN\_CANCTL0)

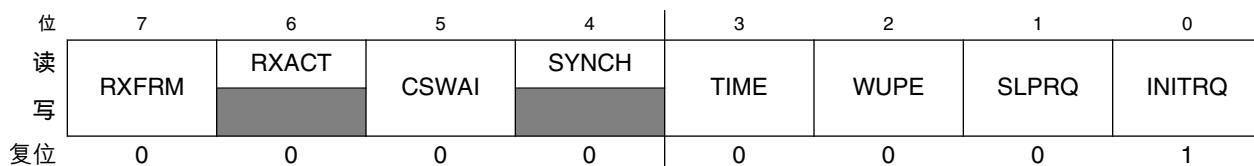
CANCTL0 寄存器提供 MSCAN 模块的各种控制位。

#### 注

写操作：退出初始化模式后随时可执行；例外情况是只读的 RXACT 和 SYNCH、RXFRM（仅由该模块设置）以及 INITRQ（在初始化模式下也可写入）

初始化模式有效 (INITRQ = 1 且 INITAK = 1) 时，CANCTL0 寄存器除 WUPE、INITRQ 和 SLPRQ 以外均保持在复位状态。只要退出初始化模式 (INITRQ = 0 且 INITAK = 0)，就能再次对该寄存器进行写操作。

地址: 4002\_4000h 基准 + 0h 偏移 = 4002\_4000h



#### MSCAN\_CANCTL0 字段描述

字段	描述
7 RXFRM	<p>接收帧标志</p> <p>该位只能读取和清零。该位在接收器正确接收到有效报文时置位，与滤波器配置无关。置位后，它将保持置位状态，直到被软件清零或复位。写入 1 即可清零。写入 0 会被忽略。该位在回环模式下无效。</p> <p>0 上次清零该标志以来，未接收到任何有效报文。 1 上次清零该标志以来，接收到一条有效报文。</p>
6 RXACT	<p>接收器活动状态</p> <p>该只读标志指示 MSCAN 正在接收报文。该标志由接收器前端控制。该位在回环模式下无效。</p>

下一页继续介绍此表...

## MSCAN\_CANCTL0 字段描述 (继续)

字段	描述
	<p>注：有关发送器和接收器状态的详细定义，请参见 Bosch CAN 2.0A/B 规范。</p> <p>0 MSCAN 正在发送或空闲。 1 MSCAN 正在接收报文，包括仲裁丢失时。</p>
5 CSWAI	<p>CAN 在 Wait 模式下停止 使能该位可实现 Wait 模式下的更低功耗，方法是禁用 MSCAN 模块在 CPU 总线接口处的所有时钟。</p> <p>注：为防止意外违反 CAN 协议，在 CPU 进入 Wait (CSWAI = 1) 或 Stop 模式时，会立即强制使 TXCAN 进入隐性状态</p> <p>0 该模块在 Wait 模式期间不受影响。 1 该模块的时钟在 Wait 模式期间停止。</p>
4 SYNCH	<p>同步状态 该只读标志指示 MSCAN 是否与 CAN 总线同步，并且能参与通信过程。它由 MSCAN 置位和清零。</p> <p>0 MSCAN 未与 CAN 总线同步。 1 MSCAN 已与 CAN 总线同步。</p>
3 TIME	<p>定时器使能 该位激活一个根据位时钟速率进行计时的内部 16 位宽自由运行定时器。如果该定时器使能，那么有效 TX/RX 缓冲区中每条已发送/已接收的报文都会分配到一个 16 位时间标志。就在 CAN 总线上有效报文的 EOF 之后，时间标志会立即被写入相应缓冲区的最高字节 (0x000E, 0x000F)。在回环模式下，不会生成任何接收时间标志。内部定时器在禁用时复位（所有位置 0）。该位在初始化模式下保持置 0。</p> <p>0 禁用内部 MSCAN 定时器。 1 使能内部 MSCAN 定时器。</p>
2 WUPE	<p>唤醒使能 在检测到 CAN 上有流量时，该配置位使 MSCAN 从睡眠模式或掉电模式（从睡眠模式进入）重启。为使所选功能生效，必须在进入睡眠模式之前对该位进行配置。</p> <p>注：如果需要从停止或等待状态恢复的机制，CPU 必须确保 WUPE 寄存器和 WUPIE 唤醒中断使能寄存器已使能。</p> <p>0 唤醒禁用 - MSCAN 忽略 CAN 上的通讯。 1 唤醒使能 - MSCAN 能够重启。</p>
1 SLPRQ	<p>睡眠模式请求 该位请求 MSCAN 进入睡眠模式，它是一种内部省电模式。睡眠模式请求在 CAN 总线空闲，即该模块未在接收报文且所有发送缓冲区皆空时予以处理。该模块通过设置 SLPAK = 1 来指示已进入睡眠模式。SLPRQ 在 WUPIF 标志置位时无法置位。直到 SLPRQ 被 CPU 清零之前，睡眠模式将一直有效。或者根据 WUPE 的置位情况，MSCAN 检测到 CAN 总线上有活动并将 SLPRQ 清零为止。</p> <p>注：CPU 在 MSCAN 进入睡眠模式 (SLPRQ = 1 且 SLPAK = 1) 之前无法将 SLPRQ 清零。</p> <p>0 运行 - MSCAN 正常工作。 1 睡眠模式请求 - MSCAN 在 CAN 总线空闲时进入睡眠模式。</p>
0 INITRQ	<p>初始化模式请求 当该位由 CPU 置位时，MSCAN 跳转到初始化模式。正在进行中的任何发送或接收活动都会中止并且与 CAN 总线的同步丢失。该模块通过设置 INITAK = 1 来指示已进入初始化模式。</p>

下一页继续介绍此表...

### MSCAN\_CANCTL0 字段描述 (继续)

字段	描述
	<p>以下寄存器进入其硬复位状态并恢复其默认值 : CANCTL0( 不包括 WUPE、INITRQ 和 SLPRQ )、CANRFLG ( TSTAT1 和 TSTAT0 不受初始化模式影响 )、CANRIER ( RSTAT1 和 RSTAT0 不受初始化模式影响 )、CANTFLG、CANTIER、CANTARQ、CANTAACK 和 CANTBSEL。寄存器 CANCTL1、CANBTR0、CANBTR1、CANIDAC、CANIDAR0-7 和 CANIDMR0-7 在 MSCAN 处于初始化模式 ( INITRQ = 1 且 INITAK = 1 ) 时只能由 CPU 对其进行写操作。错误计数器的值不受初始化模式的影响。</p> <p>当该位被 CPU 清零时，MSCAN 重启，然后尝试与 CAN 总线同步。如果 MSCAN 未处于总线关闭状态，它将在 CAN 总线上出现 11 个连续隐性位之后同步；如果 MSCAN 处于总线关闭状态，它将继续等待直到 11 个连续隐性位出现 128 次。</p> <p>要写入 CANCTL0、CANRFLG、CANRIER、CANTFLG 或 CANTIER 中的其它位，必须在退出初始化模式 ( INITRQ = 0 且 INITAK = 0 ) 之后执行。</p> <p>注：CPU 在 MSCAN 进入初始化模式 ( INITRQ = 1 且 INITAK = 1 ) 之前无法将 INITRQ 清零。</p> <p>为防止意外违反 CAN 协议，在 CPU 请求初始化模式时，会立即强制使 TXCAN 进入隐性状态。因此，建议流程是先让 MSCAN 进入睡眠模式 ( SLPRQ = 1 且 SLPAK = 1 )，再请求初始化模式。</p> <p>0 正常工作。 1 MSCAN 处于初始化模式。</p>

### 32.3.3 MSCAN 控制寄存器 1 (MSCAN\_CANCTL1)

CANCTL1 寄存器提供 MSCAN 模块的各种控制位和握手状态信息。

#### 注

写操作：除 CANE 在正常模式下只能写入一次外，初始化模式下 ( INITRQ=1 且 INITAK=1 ) 随时可执行；处于初始化模式时的特殊系统工作模式也可随时执行。

地址: 4002\_4000h 基准 + 1h 偏移 = 4002\_4001h

位 读 写	7	6	5	4	3	2	1	0
	CANE	CLKSRC	LOOPB	LISTEN	BORM	WUPM	SLPAK	INITAK
复位	0	0	0	1	0	0	0	1

### MSCAN\_CANCTL1 字段描述

字段	描述
7 CANE	MSCAN 使能 0 MSCAN 禁用。 1 MSCAN 模块使能。
6 CLKSRC	MSCAN 时钟源 该位定义 MSCAN 模块的时钟源 ( 仅适用于具有时钟产生模块的系统 )。 0 MSCAN 时钟源是振荡器时钟。 1 MSCAN 时钟源是总线时钟。

下一页继续介绍此表...

**MSCAN\_CANCTL1 字段描述 (继续)**

字段	描述
5 LOOPB	<p>回环自测模式</p> <p>该位置位时，MSCAN 执行可用于自测操作的内部回环。发送器的位流输出在内部反馈到接收器。RXCAN 输入被忽略，TXCAN 输出变为隐性状态（逻辑 1）。MSCAN 如同发送时一样正常工作，并将其自己发送的报文视为从远程节点接收到的报文。在该状态下，MSCAN 会忽略 CAN 帧应答字段中 ACK 槽期间传送的位，从而确保正确接收到自己的报文。发送中断和接收中断均会生成。</p> <p>0 回环自测禁用。 1 回环自测使能。</p>
4 LISTEN	<p>仅监听模式</p> <p>该位将 MSCAN 配置为 CAN 总线监控器。LISTEN 置位时，它会接收所有具有匹配 ID 的有效 CAN 报文，但不会发出应答或错误帧。此外，错误计数器冻结。仅监听模式支持需要“热插拔”或吞吐量分析的应用。MSCAN 在仅监听模式有效时无法发送任何报文。</p> <p>0 正常工作。 1 仅监听模式激活。</p>
3 BORM	<p>总线关闭恢复模式</p> <p>该位配置 MSCAN 的总线关闭状态恢复模式。</p> <p>0 总线关闭自动恢复（参见 Bosch CAN 2.0A/B 协议规范）。 1 用户请求时总线关闭恢复。</p>
2 WUPM	<p>唤醒模式</p> <p>如果 CANCTL0 中的 WUPE 使能，该位将决定是否采用集成式低通滤波器来防止 MSCAN 受杂散唤醒影响。</p> <p>0 MSCAN 在 CAN 总线上出现任何显性电平时唤醒。 1 MSCAN 仅当 CAN 总线上出现长度为 <math>T_{wup}</math> 的显性脉冲时才会唤醒。</p>
1 SLPAK	<p>睡眠模式应答</p> <p>该标志指示 MSCAN 模块是否已进入睡眠模式。它用作 SLPRQ 睡眠模式请求的握手标志。睡眠模式在 <math>SLPRQ = 1</math> 且 <math>SLPAK = 1</math> 时有效。根据 WUPE 的置位情况，如果 MSCAN 在睡眠模式下检测到 CAN 总线上有活动，它会将该标志清零。</p> <p>0 运行 - MSCAN 正常工作。 1 睡眠模式有效 - MSCAN 已进入睡眠模式。</p>
0 INITAK	<p>初始化模式应答</p> <p>该标志指示 MSCAN 模块是否处于初始化模式。它用作 INITRQ 初始化模式请求的握手标志。初始化模式在 <math>INITRQ = 1</math> 且 <math>INITAK = 1</math> 时有效。寄存器 CANCTL1、CANBTR0、CANBTR1、CANIDAC、CANIDAR0-CANIDAR7 和 CANIDMR0-CANIDMR7 在 MSCAN 处于初始化模式时只能由 CPU 对其进行写操作。</p> <p>0 运行 - MSCAN 正常工作。 1 初始化模式有效 - MSCAN 已进入初始化模式。</p>

**32.3.4 MSCAN 总线定时寄存器 0 (MSCAN\_CANBTR0)**

CANBTR0 寄存器配置 MSCAN 模块的各种 CAN 总线时序参数。

**注**

写操作：初始化模式（INITRQ = 1 且 INITAK = 1）下随时可执行。

地址: 4002\_4000h 基准 + 2h 偏移 = 4002\_4002h

位 读写	7	6	5	4	3	2	1	0
复位	0	0	0	0	0	0	0	0
SJW					BRP			

**MSCAN\_CANBTR0 字段描述**

字段	描述												
7–6 SJW	<p>同步跳转宽度</p> <p>同步跳转宽度定义的是，为了实现与 CAN 总线上数据转换的重新同步，一位中最多可被缩短或延长的时间量子(Tq)时钟周期的数目。</p> <table> <tr><td>00</td><td>1 个 Tq 时钟周期。</td></tr> <tr><td>01</td><td>2 个 Tq 时钟周期。</td></tr> <tr><td>10</td><td>3 个 Tq 时钟周期。</td></tr> <tr><td>11</td><td>4 个 Tq 时钟周期。</td></tr> </table>	00	1 个 Tq 时钟周期。	01	2 个 Tq 时钟周期。	10	3 个 Tq 时钟周期。	11	4 个 Tq 时钟周期。				
00	1 个 Tq 时钟周期。												
01	2 个 Tq 时钟周期。												
10	3 个 Tq 时钟周期。												
11	4 个 Tq 时钟周期。												
BRP	<p>波特率预分频器</p> <p>这些位确定用于建立位定时的时间量子(Tq)时钟。</p> <table> <tr><td>000000</td><td>1</td></tr> <tr><td>000001</td><td>2</td></tr> <tr><td>000010</td><td>.....</td></tr> <tr><td>000011</td><td>.....</td></tr> <tr><td>111110</td><td>63</td></tr> <tr><td>111111</td><td>64</td></tr> </table>	000000	1	000001	2	000010	.....	000011	.....	111110	63	111111	64
000000	1												
000001	2												
000010	.....												
000011	.....												
111110	63												
111111	64												

**32.3.5 MSCAN 总线定时寄存器 1 (MSCAN\_CANBTR1)**

CANBTR1 寄存器配置 MSCAN 模块的各种 CAN 总线时序参数。

**注**

写操作：初始化模式（INITRQ = 1 且 INITAK = 1）下随时可执行。

地址: 4002\_4000h 基准 + 3h 偏移 = 4002\_4003h

位 读写	7	6	5	4	3	2	1	0
复位	0	0	0	0	0	0	0	0
SAMP			TSEG2			TSEG1		

**MSCAN\_CANBTR1 字段描述**

字段	描述																
7 SAMP	<p>采样</p> <p>该位确定每位时间采集的 CAN 总线样本数。</p> <p>如果 SAMP = 0 , 那么作为结果的位值等于位于采样点的单个位的值。如果 SAMP = 1 , 那么作为结果的位值的确定方法是对总共三个样本采用多数决定原则。比特率较高时 , 建议每位时间仅采集一个样本(SAMP = 0)。</p> <p>0 每位一个样本。 1 每位三个样本。在该情况下 , PHASE_SEG1 必须至少为 2 个时间量子(Tq)。</p>																
6-4 TSEG2	<p>时间段 2</p> <p>位时间内的时段确定每位时间的时钟周期数和采样点的位置。</p> <table> <tr><td>000</td><td>1 个 Tq 时钟周期 ( 无效 )</td></tr> <tr><td>001</td><td>2 个 Tq 时钟周期</td></tr> <tr><td>010</td><td>3 个 Tq 时钟周期</td></tr> <tr><td>011</td><td>4 个 Tq 时钟周期</td></tr> <tr><td>100</td><td>5 个 Tq 时钟周期</td></tr> <tr><td>101</td><td>6 个 Tq 时钟周期</td></tr> <tr><td>110</td><td>7 个 Tq 时钟周期</td></tr> <tr><td>111</td><td>8 个 Tq 时钟周期</td></tr> </table>	000	1 个 Tq 时钟周期 ( 无效 )	001	2 个 Tq 时钟周期	010	3 个 Tq 时钟周期	011	4 个 Tq 时钟周期	100	5 个 Tq 时钟周期	101	6 个 Tq 时钟周期	110	7 个 Tq 时钟周期	111	8 个 Tq 时钟周期
000	1 个 Tq 时钟周期 ( 无效 )																
001	2 个 Tq 时钟周期																
010	3 个 Tq 时钟周期																
011	4 个 Tq 时钟周期																
100	5 个 Tq 时钟周期																
101	6 个 Tq 时钟周期																
110	7 个 Tq 时钟周期																
111	8 个 Tq 时钟周期																
TSEG1	<p>时间段 1</p> <p>位时间内的时段确定每位时间的时钟周期数和采样点的位置。</p> <p>位时间由振荡器频率、波特率预分频器和每位的时间量子(Tq)时钟周期数决定。</p> <p>位时间 = <math>(1 + \text{时段 1} + \text{时段 2}) * (\text{预分频器值}) / f_{\text{CANCLK}}</math></p> <table> <tr><td>0000</td><td>1 个 Tq 时钟周期 ( 无效 )</td></tr> <tr><td>0001</td><td>2 个 Tq 时钟周期 ( 无效 )</td></tr> <tr><td>0010</td><td>3 个 Tq 时钟周期 ( 无效 )</td></tr> <tr><td>0011</td><td>4 个 Tq 时钟周期</td></tr> <tr><td>.....</td><td>.....</td></tr> <tr><td>1110</td><td>15 个 Tq 时钟周期</td></tr> <tr><td>1111</td><td>16 个 Tq 时钟周期</td></tr> </table>	0000	1 个 Tq 时钟周期 ( 无效 )	0001	2 个 Tq 时钟周期 ( 无效 )	0010	3 个 Tq 时钟周期 ( 无效 )	0011	4 个 Tq 时钟周期	.....	.....	1110	15 个 Tq 时钟周期	1111	16 个 Tq 时钟周期		
0000	1 个 Tq 时钟周期 ( 无效 )																
0001	2 个 Tq 时钟周期 ( 无效 )																
0010	3 个 Tq 时钟周期 ( 无效 )																
0011	4 个 Tq 时钟周期																
.....	.....																
1110	15 个 Tq 时钟周期																
1111	16 个 Tq 时钟周期																

**32.3.6 MSCAN 接收器标志寄存器 (MSCAN\_CANRFLG)**

当引起标志置位的条件不再有效时，只能通过软件将标志清零（将 1 写入对应的位位置）。每个标志在 CANIER 寄存器中都有相应的中断使能位。

**注**

写操作：除了只读标志 RSTAT[1:0] 和 TSTAT[1:0] 外，不在初始化模式时随时可执行；写入 1 会将标志清零，写入 0 会被忽略。

CANRFLG 寄存器在初始化模式有效 (INITRQ = 1 且 INITAK = 1) 时保持在复位状态。只要该寄存器退出初始化模式 (INITRQ = 0 且 INITAK = 0)，就能再次对其进行写操作。

地址: 4002\_4000h 基准 + 4h 偏移 = 4002\_4004h

位 读 写	7	6	5	4	3	2	1	0
	WUPIF	CSCIF	RSTAT		TSTAT		OVRIF	RXF
复位	0	0	0	0	0	0	0	0

### MSCAN\_CANRFLG 字段描述

字段	描述
7 WUPIF	<p>唤醒中断标志</p> <p>如果 MSCAN 在处于睡眠模式且 CANTCTL0[WUPE] = 1 的同时检测到 CAN 总线上有活动，那么该模块将置位 WUPIF。如果未被屏蔽，那么唤醒中断在该标志置位的同时挂起。</p> <p>0 睡眠模式下未观察到任何唤醒活动。 1 MSCAN 检测到 CAN 总线上有活动并请求唤醒。</p>
6 CSCIF	<p>CAN 状态变更中断标志</p> <p>当 MSCAN 因发送错误计数器(TEC)和接收错误计数器(REC)的实际值而改变其当前 CAN 总线状态时，该标志置位。另有一个 4 位 (RSTAT[1:0]、TSTAT[1:0] ) 状态寄存器 (TEC 和 REC 各有对应的部分) 会将 CAN 总线的实际状态告知系统。如果未被屏蔽，那么错误中断在该标志置位的同时挂起。CSCIF 提供一个阻塞中断。它保证仅当无任何 CAN 状态变更中断挂起时，才会更新接收器/发送器状态位(RSTAT/TSTAT)。如果 TEC/REC 在 CSCIF 的电平变为有效值后改变其当前值，导致 RSTAT/TSTAT 位又发生状态变化，那么这些位将保持其状态，直到当前 CSCIF 中断再次被清除为止。</p> <p>0 CAN 总线状态自上次中断以来无任何变化。 1 MSCAN 改变了 CAN 总线当前状态。</p>
5–4 RSTAT	<p>接收器状态</p> <p>错误计数器的值控制 MSCAN 的 CAN 总线实际状态。只要状态变更中断标志(CSCIF)置位，这些位就会指示 MSCAN 与相应接收器相关的 CAN 总线状态。</p> <p>注：该字段不受初始化模式的影响。</p> <p>00 RxOK : 0≤接收错误计数器&lt;96 01 RxWRN : 96≤接收错误计数器&lt;128 10 RxERR : 128≤接收错误计数器 11 总线关闭 : 256≤发送错误计数器 (有关最重要的 CAN 总线状态 (“总线关闭”) 的冗余信息。仅当发送错误计数器的错误数超过 255 时，才会发生这种情况。总线关闭影响接收器状态。只要发送器离开其总线关闭状态，接收器状态就同样会跳转到 RxOK。另请参见该寄存器中的 TSTAT[1:0] 编码。)</p>
3–2 TSTAT	<p>发送器状态</p> <p>错误计数器的值控制 MSCAN 的 CAN 总线实际状态。只要状态变更中断标志(CSCIF)置位，这些位就会指示 MSCAN 与相应发送器相关的 CAN 总线状态。</p> <p>注：该字段不受初始化模式的影响。</p> <p>00 TxOK : 0≤发送错误计数器&lt;96</p>

下一页继续介绍此表...

**MSCAN\_CANRFLG 字段描述 (继续)**

字段	描述
	01 TxWRN : 96≤发送错误计数器<128 10 TxERR : 128≤发送错误计数器<256 11 总线关闭 : 256≤发送错误计数器
1 OVRIF	<b>溢出中断标志</b> 该标志在发生数据溢出状况时置位。如果未被屏蔽，那么错误中断在该标志置位的同时挂起。 0 无数据溢出状况。 1 检测到数据溢出。
0 RXF	<b>接收缓冲区满标志</b> 有新报文移入接收器 FIFO 时，RXF 由 MSCAN 置位。该标志指示移入的缓冲区是否加载了正确接收的报文( 标识符匹配、循环冗余校验码(CRC)匹配且未检测到其它错误 )。CPU 从接收器 FIFO 的 RxFG 缓冲区读取该报文之后，必须将 RXF 标志清零以释放缓冲区。已置位的 RXF 标志会禁止将下一个 FIFO 条目移入前台缓冲区(RxFG)。如果未被屏蔽，那么接收中断在该标志置位的同时挂起。 <b>注：</b> 为确保数据完整性，在将 RXF 标志清零的同时请勿对接收缓冲区寄存器进行读操作。对于具有双 CPU 的 MCU，在将 RXF 标志清零的同时读取接收缓冲区寄存器可能会导致 CPU 故障。 0 RxFG 中无任何新报文可用。 1 接收器 FIFO 非空。RxFG 中有新报文可用。

**32.3.7 MSCAN 接收器中断使能寄存器 (MSCAN\_CANRIER)**

该寄存器包含 CANRFLG 寄存器中所述中断标志的中断使能位。

**注**

CANRIER 寄存器在初始化模式有效 (INITRQ = 1 且 INITAK = 1) 时保持在复位状态。不处于初始化模式 (INITRQ=0 且 INITAK=0) 时，可对该寄存器进行写操作。

RSTATE[1:0]、TSTATE[1:0]位不受初始化模式的影响。

地址: 4002\_4000h 基准 + 5h 偏移 = 4002\_4005h

位 读写	7	6	5	4	3	2	1	0
	WUPIE	CSCIE	RSTATE		TSTATE		OVRIE	RXFIE
复位	0	0	0	0	0	0	0	0

**MSCAN\_CANRIER 字段描述**

字段	描述
7 WUPIE	<b>唤醒中断使能</b> 如果需要从停止或等待状态恢复的机制，WUPIE 和 WUPE 均必须使能。

下一页继续介绍此表...

**MSCAN\_CANRIER 字段描述 (继续)**

字段	描述
	0 该事件不生成任何中断请求。 1 唤醒事件引起唤醒中断请求。
6 CSCIE	CAN 状态变更中断使能 0 该事件不生成任何中断请求。 1 CAN 状态变更事件引起错误中断请求。
5-4 RSTATE	接收器状态变更使能 <p>这些 RSTAT 使能位控制接收器状态变更引起 CSCIF 中断的灵敏度水平。无论选择何种灵敏度水平 ,RSTAT 标志都会持续指示接收器实际状态，并且仅在没有 CSCIF 中断挂起时更新。</p> <ul style="list-style-type: none"> <li>00 不生成由接收器状态变更而引起的任何 CSCIF 中断。</li> <li>01 仅当接收器进入或退出“总线关闭”状态时生成 CSCIF 中断。放弃其它接收器状态变更以便生成 CSCIF 中断。</li> <li>10 仅当接收器进入或退出“RxErr”或“总线关闭”状态时生成 CSCIF 中断 <sup>1</sup> 状态。放弃其它接收器状态变更以便生成 CSCIF 中断。</li> <li>11 对所有状态变更都生成 CSCIF 中断。</li> </ul>
3-2 TSTATE	发送器状态变更使能 <p>这些 TSTAT 使能位控制发送器状态变更引起 CSCIF 中断的灵敏度水平。无论选择何种灵敏度水平 ,TSTAT 标志都会持续指示发送器实际状态，并且仅在没有 CSCIF 中断挂起时更新。</p> <ul style="list-style-type: none"> <li>00 不生成由发送器状态变更引起的任何 CSCIF 中断。</li> <li>01 仅当发送器进入或退出“总线关闭”状态时生成 CSCIF 中断。放弃其它发送器状态变更以便生成 CSCIF 中断。</li> <li>10 仅当发送器进入或退出“TxErr”或“总线关闭”状态时生成 CSCIF 中断。放弃其它发送器状态变更以便生成 CSCIF 中断。</li> <li>11 对所有状态变更都生成 CSCIF 中断。</li> </ul>
1 OVRIE	溢出中断使能 0 该事件不生成任何中断请求。 1 溢出事件引起错误中断请求。
0 RXFIE	接收器满中断使能 0 该事件不生成任何中断请求。 1 接收缓冲区满 ( 报文接收成功 ) 事件引起接收器中断请求。

1. CAN 标准仅针对发送器定义了总线关闭状态 ( 参见 Bosch CAN 2.0A/B 协议规范 )。由于发送器从总线关闭到 TxOK 的唯一可能的状态变更也会强制使接收器从当前状态跳转到 RxOK ,因此 RXSTAT[1:0]标志的编码针对接收器状态定义了又一个总线关闭

**32.3.8 MSCAN 发送器标志寄存器 (MSCAN\_CANTFLG)**

发送缓冲区空标志在 CANTIER 寄存器中都有相应的中断使能位。

**注**

写操作：不处于初始化模式时随时可执行；写入 1 会将标志清零，写入 0 会被忽略。

CANTFLG 寄存器在初始化模式有效 (INITRQ = 1 且 INITAK = 1) 时保持在复位状态。该寄存器不处于初始化模式 (INITRQ = 0 且 INITAK = 0) 时，可对其进行写操作。

地址: 4002\_4000h 基准 + 6h 偏移 = 4002\_4006h

位	7	6	5	4		3	2	1	0
读									
写									
复位	0	0	0	0		0	1	1	1

### MSCAN\_CANTFLG 字段描述

字段	描述
7–3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
TXE	发送器缓冲区空  该标志指示相关的发送报文缓冲区为空，因此无发送安排。当有报文移入发送缓冲区且应予以发送时，CPU 必须将该标志清零。MSCAN 在报文成功传送后置位该标志。该标志在发送请求因挂起的中止请求而成功中止时也会由 MSCAN 置位。如果未被屏蔽，那么发送中断在该标志置位的同时挂起。  将 TXEx 标志清零也会将对应的 ABTAKx 清零。TXEx 标志置位时，会将对应的 ABTRQx 位清零。  监听模式有效时，无法将 TXEx 标志清零且不会开始发送。  如果将对应的 TXEx 位清零(TXEx = 0)且安排缓冲区发送报文，那么对发送缓冲区的读写访问将受阻。  0 相关的报文缓冲区已满 (载入了要发送的报文)。 1 相关的报文缓冲区为空 (无发送安排)。

### 32.3.9 MSCAN 发送器中断使能寄存器 (MSCAN\_CANTIER)

该寄存器包含发送缓冲区空中断标志的中断使能位。

#### 注

CANTIER 寄存器在初始化模式有效 (INITRQ = 1 且 INITAK = 1) 时保持在复位状态。该寄存器不处于初始化模式 (INITRQ = 0 且 INITAK = 0) 时，可对其进行写操作。

不处于初始化模式时，随时可写入。

地址: 4002\_4000h 基准 + 7h 偏移 = 4002\_4007h

位	7	6	5	4		3	2	1	0
读									
写									
复位	0	0	0	0		0	0	0	0

**MSCAN\_CANTIER 字段描述**

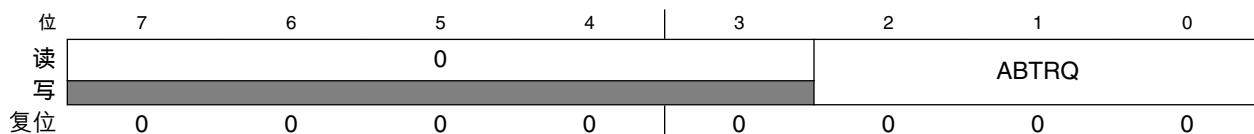
字段	描述
7–3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
TXEIE	发送器空中断使能  0 该事件不生成任何中断请求。 1 发送器空 (发送缓冲区可用于发送) 事件引起发送器空中断请求。

**32.3.10 MSCAN 发送器报文中止请求寄存器 (MSCAN\_CANTARQ)**  
CANTARQ 寄存器用于对排队的报文提出中止请求。**注**

写操作：不处于初始化模式时，随时可写入。

CANTARQ 寄存器在初始化模式有效 (INITRQ = 1 且 INITAK = 1) 时保持在复位状态。该寄存器不处于初始化模式 (INITRQ = 0 且 INITAK = 0) 时，可对其进行写操作。

地址: 4002\_4000h 基准 + 8h 偏移 = 4002\_4008h

**MSCAN\_CANTARQ 字段描述**

字段	描述
7–3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
ABTRQ	中止请求  CPU 置位 ABTRQx 位以请求中止一个已安排的报文缓冲区(TXEx = 0)。如果该报文尚未开始发送，或者发送不成功 (仲裁丢失或错误)，MSCAN 就会批准该请求。报文中止时，相关的 TXE 和中止应答标志 ABTAK 置位，并且会生成发送中断 (若已使能)。CPU 无法复位 ABTRQx。只要相关的 TXE 标志置位，ABTRQx 就会复位。  0 无中止请求。 1 中止请求挂起。

**32.3.11 MSCAN 发送器报文中止应答寄存器 (MSCAN\_CANTAAK)**  
CANTAAK 寄存器指示成功中止排队报文(如果请求由 CANTARQ 寄存器中的相应位发出)。

**注**

CANTAAK 寄存器在初始化模式有效 (INITRQ = 1 且 INITAK = 1) 时保持在复位状态。

地址: 4002\_4000h 基准 + 9h 偏移 = 4002\_4009h

位	7	6	5	4		3	2	1	0
读	0					ABTAK			
写									
复位	0	0	0	0		0	0	0	0

**MSCAN\_CANTAAK 字段描述**

字段	描述
7–3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
ABTAK	中止应答  该标志确认某条报文因挂起的来自 CPU 的中止请求而中止。特定报文缓冲区的空标志置位后，应用软件可利用该标志来确定该报文是被成功中止还是已经传送出去。只要将对应的 TXE 标志清零，就会将 ABTAKx 标志清零。  0 报文未被中止。 1 报文已被中止。

**32.3.12 MSCAN 发送缓冲区选择寄存器 (MSCAN\_CANTBSEL)**

CANTBSEL 寄存器用于选择实际的发送报文缓冲区，然后便可在 CANTXFG 寄存器空间中访问它。

**注**

读操作：找到置 1 的位序最低位，所有其它位将被读作 0。

写操作：不处于初始化模式时，随时可写入。

CANTBSEL 寄存器在初始化模式有效 (INITRQ = 1 且 INITAK = 1) 时保持在复位状态。该寄存器不处于初始化模式 (INITRQ = 0 且 INITAK = 0) 时，可对其进行写操作。

下面是有关 CANTBSEL 寄存器使用的一个简短编程示例：

为了获得下一个可用的发送缓冲区，应用软件必须对 CANTFLG 寄存器进行读操作，并将该值写回到 CANTBSEL 寄存器中。本例中，Tx 缓冲区 TX1 和 TX2 可用。因此，从 CANTFLG 读取的值为 0b0000\_0110。将该值写回到 CANTBSEL 时，CANTXFG 选择 Tx 缓冲区 TX1，因为置 1 的位序最低位在位置 1 处。从 CANTBSEL 读回该值的结果为 0b0000\_0010，因为仅显示置 1 的位序最低位位置。该机制可简化应用软件选择下一个可用 Tx 缓冲区的过程。

如果取消选择所有发送报文缓冲区，则不允许对 CANTXFG 寄存器进行任何访问。

地址: 4002\_4000h 基准 + Ah 偏移 = 4002\_400Ah

位	7	6	5	4		3	2	1	0
读									
写									
复位	0	0	0	0		0	0	0	0

#### MSCAN\_CANTBSEL 字段描述

字段	描述
7-3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
TX	发送缓冲区选择  位序最低位将相应的发送缓冲区置于 CANTXFG 寄存器空间（例如，TX1 = 1 且 TX0 = 1 选择发送缓冲区 TX0；TX1 = 1 且 TX0 = 0 选择发送缓冲区 TX1）。如果对应的 TXEx 位被清零且安排缓冲区发送报文，那么对所选发送缓冲区的读写访问将受阻。  0 取消选择相关的报文缓冲区。 1 如果是位序最低位，则选择相关的报文缓冲区。

### 32.3.13 MSCAN 标识符验收控制寄存器 (MSCAN\_CANIDAC)

CANIDAC 寄存器用于标识符验收控制，如下所述。

#### 注

写操作：初始化模式 (INITRQ = 1 且 INITAK = 1) 下随时可执行，只读位 IDHITx 除外。

地址: 4002\_4000h 基准 + Bh 偏移 = 4002\_400Bh

位	7	6	5	4		3	2	1	0
读									
写									
复位	0	0	0	0		0	0	0	0

#### MSCAN\_CANIDAC 字段描述

字段	描述
7-6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5-4 IDAM	标识符验收模式  CPU 置位这些标志以定义标识符验收滤波器结构。在滤波器关闭模式下，不接受任何报文，前台缓冲区因此也不会重新加载。  00 两个 32 位验收滤波器。 01 四个 16 位验收滤波器。

下一页继续介绍此表...

**MSCAN\_CANIDAC 字段描述 (继续)**

字段	描述
	10 八个 8 位验收滤波器。 11 滤波器关闭。
3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
IDHIT	标识符验收命中指示器  MSCAN 置位这些标志以指示标识符验收命中情况。IDHIT 指示器始终与前台缓冲区(RxFG)中的报文相关。当报文移入接收器 FIFO 的前台缓冲区时，指示器也会更新。  000 滤波器 0 命中。 001 滤波器 1 命中。 010 滤波器 2 命中。 011 滤波器 3 命中。 100 滤波器 4 命中。 101 滤波器 5 命中。 110 滤波器 6 命中。 111 滤波器 7 命中。

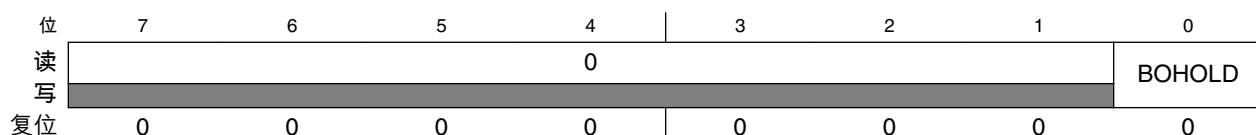
**32.3.14 MSCAN 其他寄存器 (MSCAN\_CANMISC)**

该寄存器提供附加特性。

**注**

写操作：随时；写入“1”会将标志清零，写入“0”会被忽略。

地址: 4002\_4000h 基准 + Dh 偏移 = 4002\_400Dh

**MSCAN\_CANMISC 字段描述**

字段	描述
7-1 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
0 BOHOLD	总线关闭状态保持到用户请求为止  如果 MSCAN 控制寄存器 1 (CANCTL1)的 BORM 置位，那么该位指示该模块是否已进入总线关闭状态。将该位清零会发出从总线关闭状态恢复的请求。  0 模块不处于总线关闭状态，或者用户在总线关闭状态下已请求恢复。 1 模块处于总线关闭状态，且会保持到用户请求为止。

### 32.3.15 MSCAN 接收错误计数器 (MSCAN\_CANRXERR)

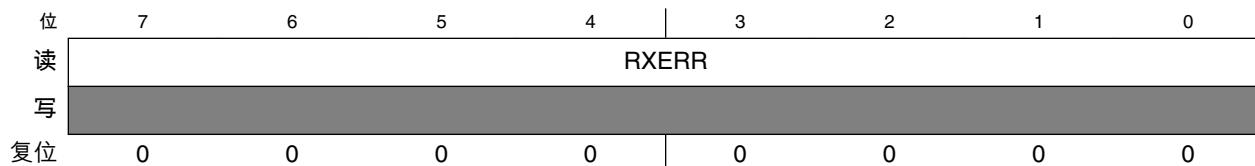
该寄存器反映 MSCAN 接收错误计数器的状态。

#### 注

读操作：只能在睡眠模式 (SLPRQ = 1 且 SLPAK = 1) 或初始化模式 (INITRQ = 1 且 INITAK = 1) 下进行。

在非睡眠或初始化模式下对该寄存器进行读操作可能会返回不正确的值。对于具有双 CPU 的 MCU, 这可能导致 CPU 发生故障。

地址: 4002\_4000h 基准 + Eh 偏移 = 4002\_400Eh



#### MSCAN\_CANRXERR 字段描述

字段	描述
RXERR	接收错误计数器 该字段只能在睡眠模式 (SLPRQ = 1 且 SLPAK = 1) 或初始化模式 (INITRQ = 1 且 INITAK = 1) 下读取。

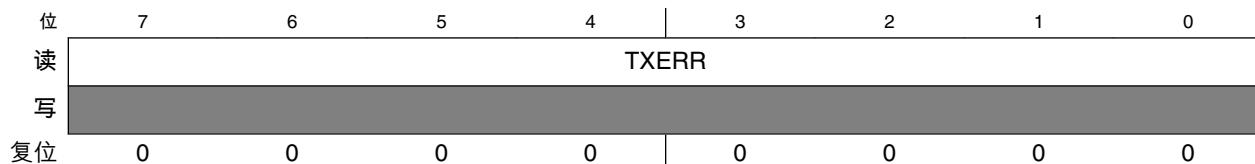
### 32.3.16 MSCAN 发送错误计数器 (MSCAN\_CANTXERR)

该寄存器反映 MSCAN 发送错误计数器的状态。

#### 注

在非睡眠或初始化模式下对该寄存器进行读操作可能会返回不正确的值。对于具有双 CPU 的 MCU, 这可能导致 CPU 发生故障。

地址: 4002\_4000h 基准 + Fh 偏移 = 4002\_400Fh



#### MSCAN\_CANTXERR 字段描述

字段	描述
TXERR	发送错误计数器

**MSCAN\_CANTXERR 字段描述 (继续)**

字段	描述
	该字段只能在睡眠模式 ( SLPRQ = 1 且 SLPAK = 1 ) 或初始化模式 ( INITRQ = 1 且 INITAK = 1 ) 下读取。

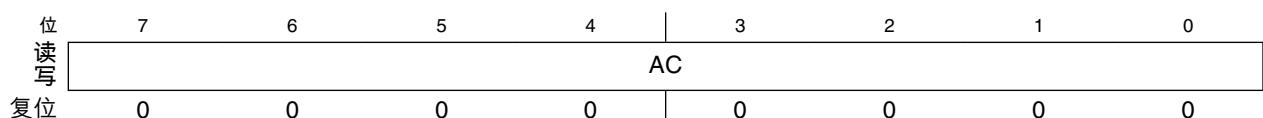
### 32.3.17 第一群组中的 MSCAN 标识符验收寄存器 n (MSCAN\_CANIDARn)

接收时，每条报文都被写入后台接收缓冲区。CPU 被告知对报文进行读取的唯一条件是，该报文符合标识符验收和标识符屏蔽寄存器中的标准 (已接受)；不符合标准的报文会被下一条报文覆盖 (已丢弃)。

MSCAN 的验收寄存器以逐位方式应用于 REIDR0-REIDR3 寄存器来检查传入报文的扩展标识符，并应用于 RSIDR0-RSIDR1 寄存器来检查标准标识符。

对于扩展标识符，应用所有四个验收和屏蔽寄存器。对于标准标识符，仅应用前两个 (CANIDAR0/1、CANIDMR0/1) 寄存器。

Address: 4002\_4000h base + 10h offset + (1d × i), where i=0d to 3d

**MSCAN\_CANIDARn 字段描述**

字段	描述
AC	验收代码位 AC[7:0]包含用户定义的位序列，它与接收报文缓冲区的相关标识符寄存器 (RSIDRn 或 REIDRn) 的对应位作比较。该比较的结果随后用对应的标识符屏蔽寄存器进行屏蔽。

### 32.3.18 第一群组中的 MSCAN 标识符屏蔽寄存器 n (MSCAN\_CANIDMRn)

标识符屏蔽寄存器指定标识符验收寄存器中哪些对应的位与验收滤波相关。要在 32 位滤波器模式下接收标准标识符，须将屏蔽寄存器 CANIDMR1 和 CANIDMR5 的最后三位(AM[2:0])编程为“无关位”。要在 16 位滤波器模式下接收标准标识符，须将屏蔽寄存器 CANIDMR1、CANIDMR3、CANIDMR5 和 CANIDMR7 的最后三位(AM[2:0])编程为“无关位”。

Address: 4002\_4000h base + 14h offset + (1d × i), where i=0d to 3d

位 读写	7	6	5	4		3	2	1	0
复位	0	0	0	0	AM	0	0	0	0

#### MSCAN\_CANIDMRn 字段描述

字段	描述
AM	<p>验收屏蔽位</p> <p>该寄存器中的特定位被清零时，表示标识符验收寄存器中的对应位必须与其标识符位相同，才能检测到匹配。如果所有此类位都匹配，则接受报文。如果某位置位，这表示标识符验收寄存器中对应位的状态不影响报文的接受与否。</p> <p>0 匹配对应的验收代码寄存器和标识符位。 1 忽略对应的验收代码寄存器位。</p>

### 32.3.19 第二群组中的 MSCAN 标识符验收寄存器 n (MSCAN\_CANIDARn)

接收时，每条报文都被写入后台接收缓冲区。CPU 被告知对报文进行读取的唯一条件是，该报文符合标识符验收和标识符屏蔽寄存器中的标准（已接受）；不符合标准的报文会被下一条报文覆盖（已丢弃）。

MSCAN 的验收寄存器以逐位方式应用于 REIDR0-REIDR3 寄存器来检查传入报文的扩展标识符，并应用于 RSIDR0-RSIDR1 寄存器来检查标准标识符。

对于扩展标识符，应用所有四个验收和屏蔽寄存器。对于标准标识符，仅应用前两个 (CANIDAR4/5、CANIDMR4/5) 寄存器。

Address: 4002\_4000h base + 18h offset + (1d × i), where i=0d to 3d

位 读写	7	6	5	4		3	2	1	0
复位	0	0	0	0	AC	0	0	0	0

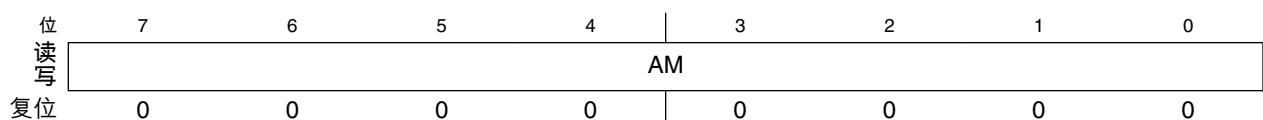
**MSCAN\_CANIDARn 字段描述**

字段	描述
AC	验收代码位 AC[7:0]包含用户定义的位序列，它与接收报文缓冲区的相关标识符寄存器（RSIDRn 或 REIDRn）的对应位作比较。该比较的结果随后用对应的标识符屏蔽寄存器进行屏蔽。

**32.3.20 第二群组中的 MSCAN 标识符屏蔽寄存器 n  
(MSCAN\_CANIDMRn)**

标识符屏蔽寄存器指定标识符验收寄存器中哪些对应的位与验收滤波相关。要在 32 位滤波器模式下接收标准标识符，须将屏蔽寄存器 CANIDMR1 和 CANIDMR5 的最后三位(AM[2:0])编程为“无关位”。要在 16 位滤波器模式下接收标准标识符，须将屏蔽寄存器 CANIDMR1、CANIDMR3、CANIDMR5 和 CANIDMR7 的最后三位(AM[2:0])编程为“无关位”。

Address: 4002\_4000h base + 1Ch offset + (1d × i), where i=0d to 3d

**MSCAN\_CANIDMRn 字段描述**

字段	描述
AM	验收屏蔽位 该寄存器中的特定位被清零时，表示标识符验收寄存器中的对应位必须与其标识符位相同，才能检测到匹配。如果所有此类位都匹配，则接受报文。如果某位置位，这表示标识符验收寄存器中对应位的状态不影响报文的接受与否。 0 匹配对应的验收代码寄存器和标识符位。 1 忽略对应的验收代码寄存器位。

### 32.3.21 接收扩展标识符寄存器 0 (MSCAN\_REIDR0)

扩展格式标识符的寄存器总共有 32 位：REID[28:0]、RSRR、RIDE 和 RRTR。

地址: 4002\_4000h 基准 + 20h 偏移 = 4002\_4020h

位 读写	7	6	5	4		3	2	1	0
	REID28_REID21								
复位	x*	x*	x*	x*		x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

#### MSCAN\_REIDR0 字段描述

字段	描述
REID28_REID21	扩展格式标识符  扩展格式标识符由 29 位(REID[28:0])组成。REID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。

### 32.3.22 接收标准标识符寄存器 0 (MSCAN\_RSIDR0)

标准格式标识符的寄存器总共有 13 位：RID[10:0]、RRTR 和 RSIDE。

地址: 4002\_4000h 基准 + 20h 偏移 = 4002\_4020h

位 读写	7	6	5	4		3	2	1	0
	RSID10_RSID3								
复位	x*	x*	x*	x*		x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

#### MSCAN\_RSIDR0 字段描述

字段	描述
RSID10_RSID3	标准格式标识符  标准格式标识符由 11 位(RSID[10:0])组成。RSID10 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。

### 32.3.23 接收扩展标识符寄存器 1 (MSCAN\_REIDR1)

扩展格式标识符的寄存器总共有 32 位：REID[28:0]、RSRR、REIDE 和 RRTR。

地址: 4002\_4000h 基准 + 21h 偏移 = 4002\_4021h

位 读写	7	6	5	4		3	2	1	0
	REID20_REID18			RSRR		REIDE	REID17_REID15		
复位	x*	x*	x*	x*		x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

#### MSCAN\_REIDR1 字段描述

字段	描述
7-5 REID20_REID18	扩展格式标识符 20-18  扩展格式标识符由 29 位(REID[28:0])组成。EID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。
4 RSRR	替代远程请求  该固定隐性位只能用于扩展格式。对于发送缓冲区，用户必须将其置 1；对于接收缓冲区，将其如同在 CAN 总线上接收到一样存储。
3 REIDE	ID 扩展  该标志指示该缓冲区应用的是扩展格式标识符还是标准格式标识符。对于接收缓冲区，该标志按接收值设置，并告知 CPU 如何处理缓冲区标识符寄存器。对于发送缓冲区，该标志告知 MSCAN 应传送何种类型的标识符。  0 标准格式 (11 位)。 1 扩展格式 (29 位)。
REID17_REID15	扩展格式标识符 17-15  扩展格式标识符由 29 位(REID[28:0])组成。RID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。

### 32.3.24 接收标准标识符寄存器 1 (MSCAN\_RSIDR1)

标准格式标识符的寄存器总共有 13 位：RSID[10:0]、RRTR 和 REIDE。

地址: 4002\_4000h 基准 + 21h 偏移 = 4002\_4021h

位 读 写	7	6	5	4	3	2	1	0
				RSRTR	RSIDE	Reserved	Reserved	
复位	x*	x*	x*	x*	x*	x*	x*	x*

\* 注：

- x = 复位时未定义。

#### MSCAN\_RSIDR1 字段描述

字段	描述
7–5 RSID2_RSID0	标准格式标识符 2-0  标准格式标识符由 11 位(RID[10:0])组成。RID10 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。
4 RSRTR	远程发送请求  该标志反映 CAN 帧中远程发送请求位的状态。对于接收缓冲区，它指示接收帧的状态，支持通过软件发送回应帧。对于发送缓冲区，该标志定义要传送的 RRTR 位的置位。  0 数据帧。 1 远程帧。
3 RSIDE	ID 扩展  该标志指示该缓冲区应用的是扩展格式标识符还是标准格式标识符。对于接收缓冲区，该标志按接收值设置，并告知 CPU 如何处理缓冲区标识符寄存器。对于发送缓冲区，该标志告知 MSCAN 应传送何种类型的标识符。  0 标准格式 (11 位)。 1 扩展格式 (29 位)。
Reserved	此字段为保留字段。

### 32.3.25 接收扩展标识符寄存器 2 (MSCAN\_REIDR2)

扩展格式标识符的寄存器总共有 32 位：REID[28:0]、RSRR、REIDE 和 RRTR。

地址: 4002\_4000h 基准 + 22h 偏移 = 4002\_4022h

位 读写	7	6	5	4		3	2	1	0
					REID14_REID7				
复位	x*	x*	x*	x*		x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

#### MSCAN\_REIDR2 字段描述

字段	描述
REID14_REID7	扩展格式标识符 14-7  扩展格式标识符由 29 位(REID[28:0])组成。REID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。

### 32.3.26 接收扩展标识符寄存器 3 (MSCAN\_REIDR3)

扩展格式标识符的寄存器总共有 32 位：REID[28:0]、RSRR、REIDE 和 RRTR。

地址: 4002\_4000h 基准 + 23h 偏移 = 4002\_4023h

位 读写	7	6	5	4		3	2	1	0
					REID6_REID0				RERTR
复位	x*	x*	x*	x*		x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

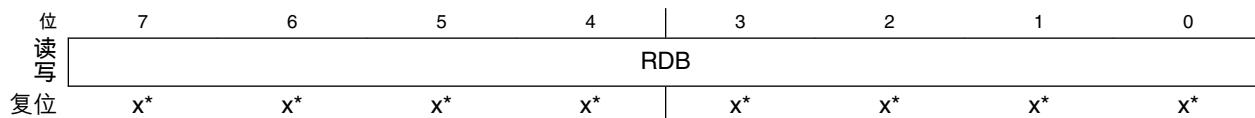
#### MSCAN\_REIDR3 字段描述

字段	描述
7-1 REID6_REID0	扩展格式标识符 6-0  扩展格式标识符由 29 位(REID[28:0])组成。REID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。
0 RERTR	远程发送请求  该标志反映 CAN 帧中远程发送请求位的状态。对于接收缓冲区，它指示接收帧的状态，支持通过软件发送回应帧。对于发送缓冲区，该标志定义要传送的 RTR 位的置位。  0 数据帧。 1 远程帧。

### 32.3.27 接收扩展数据段寄存器 N (MSCAN\_REDSR $n$ )

8 个数据段寄存器（各有位 RDB[7:0]）包含要接收的数据。要接收的字节数由对应 RDLR 寄存器中的数据长度码决定。

Address: 4002\_4000h base + 24h offset + (1d × i), where i=0d to 7d



\* 注:

- x = 复位时未定义。

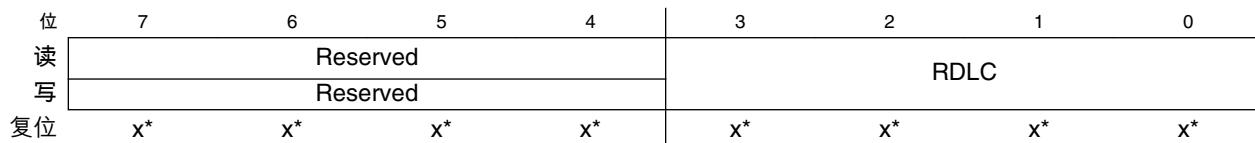
**MSCAN\_REDSR $n$  字段描述**

字段	描述
RDB	数据位 要接收的数据

### 32.3.28 接收数据长度寄存器 (MSCAN\_RDLR)

该寄存器保存 CAN 帧的数据长度字段。

地址: 4002\_4000h 基准 + 2Ch 偏移 = 4002\_402Ch



\* 注:

- x = 复位时未定义。

**MSCAN\_RDLR 字段描述**

字段	描述
7–4 Reserved	此字段为保留字段。
RDLC	数据长度码位  数据长度码包含相应报文的字节数（数据字节计数）。发送远程帧时，数据长度码按编程值发送，而发送的数据字节数始终是 0。对于一个数据帧，数据字节数范围是 0 到 8。  0000 0 0001 1

下一页继续介绍此表...

**MSCAN\_RDLR 字段描述 (继续)**

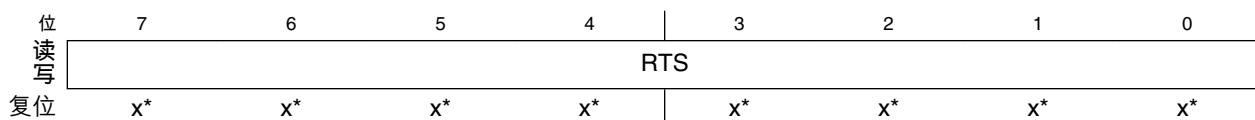
字段	描述
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
其他	保留

**32.3.29 接收时间标志寄存器高位 (MSCAN\_RTSRH)**

如果 TIME 位使能，则在 CAN 总线上出现有效报文的 EOF 之后，MSCAN 会立即写入有效接收缓冲区中的相应寄存器。

用作时间标志的定时器值来自一个自由运行的 CAN 内部位时钟。MSCAN 不会指示定时器溢出。该定时器在初始化模式期间复位(所有位都置 0)。CPU 只能对时间标志寄存器进行读操作。

地址: 4002\_4000h 基准 + 2Eh 偏移 = 4002\_402Eh



\* 注:

- x = 复位时未定义。

**MSCAN\_RTSRH 字段描述**

字段	描述
RTS	时间标志 时间标志 7 至 0。

**32.3.30 接收时间标志寄存器低位 (MSCAN\_RTSRL)**

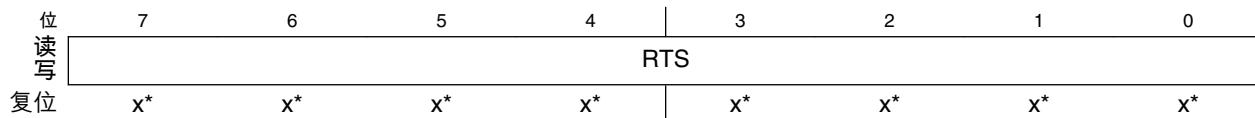
如果 TIME 位使能，则在 CAN 总线上出现有效报文的 EOF 之后，MSCAN 会立即写入有效接收缓冲区中的相应寄存器。

用作时间标志的定时器值来自一个自由运行的 CAN 内部位时钟。MSCAN 不会指示定时器溢出。该定时器在初始化模式期间复位(所有位都置 0)。CPU 只能对时间标志寄存器进行读操作。

**注**

当 TXEx 标志置位且在 CANTBSEL 中选中对应的发送缓冲区时，可随时对该寄存器进行读写操作。

地址: 4002\_4000h 基准 + 2Fh 偏移 = 4002\_402Fh



\* 注:

- x = 复位时未定义。

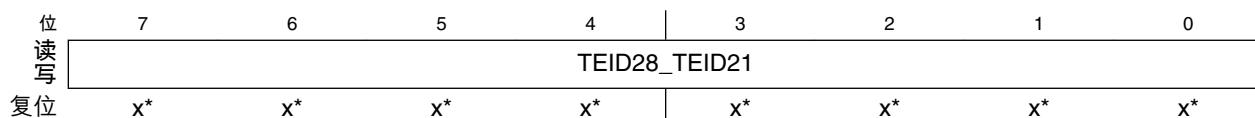
**MSCAN\_RTSRL 字段描述**

字段	描述
RTS	时间标志 时间标志 15 至 8。

**32.3.31 发送扩展标识符寄存器 0 (MSCAN\_TEIDR0)**

扩展格式标识符的寄存器总共有 32 位: TEID[28:0]、TSRR、TIDE 和 TRTR。

地址: 4002\_4000h 基准 + 30h 偏移 = 4002\_4030h



\* 注:

- x = 复位时未定义。

**MSCAN\_TEIDR0 字段描述**

字段	描述
TEID28_TEID21	扩展格式标识符 扩展格式标识符由 29 位(TEID[28:0])组成。TEID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。

### 32.3.32 发送标准标识符寄存器 0 (MSCAN\_TSIDR0)

标准格式标识符的寄存器总共有 13 位： TID[10:0]、 TRTR 和 TSIDE。

地址: 4002\_4000h 基准 + 30h 偏移 = 4002\_4030h

位 读写	7	6	5	4		3	2	1	0
					TSID10_TSID3				
复位	x*	x*	x*	x*		x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

#### MSCAN\_TSIDR0 字段描述

字段	描述
TSID10_TSID3	标准格式标识符  标准格式标识符由 11 位(TSID[10:0])组成。TSID10 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。

### 32.3.33 发送扩展标识符寄存器 1 (MSCAN\_TEIDR1)

扩展格式标识符的寄存器总共有 32 位： TEID[28:0]、 TSRR、 TEIDE 和 TRTR。

地址: 4002\_4000h 基准 + 31h 偏移 = 4002\_4031h

位 读写	7	6	5	4		3	2	1	0
					TEID20_TEID18	TSRR	TEIDE		TEID17_TEID15
复位	x*	x*	x*	x*		x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

#### MSCAN\_TEIDR1 字段描述

字段	描述
7–5 TEID20_TEID18	扩展格式标识符 20-18  扩展格式标识符由 29 位(TEID[28:0])组成。TEID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。
4 TSRR	替代远程请求  该固定隐性位只能用于扩展格式。对于发送缓冲区，用户必须将其置 1；对于接收缓冲区，将其如同在 CAN 总线上接收到一样存储。
3 TEIDE	ID 扩展

下一页继续介绍此表...

**MSCAN\_TEIDR1 字段描述 (继续)**

字段	描述
	该标志指示该缓冲区应用的是扩展格式标识符还是标准格式标识符。对于接收缓冲区，该标志按接收值设置，并告知 CPU 如何处理缓冲区标识符寄存器。对于发送缓冲区，该标志告知 MSCAN 应传送何种类型的标识符。 0 标准格式 (11 位)。 1 扩展格式 (29 位)。
TEID17_TEID15	扩展格式标识符 17-15  扩展格式标识符由 29 位(TEID[28:0])组成。TID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。

**32.3.34 发送标准标识符寄存器 1 (MSCAN\_TSIDR1)**

标准格式标识符的寄存器总共有 13 位：TEID[10:0]、TRTR 和 TEIDE。

地址: 4002\_4000h 基准 + 31h 偏移 = 4002\_4031h

位	7	6	5	4	3	2	1	0
读写		TSID2_TSID0		TSRTR	TSIDE	Reserved	Reserved	
复位	x*	x*	x*	x*	x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

**MSCAN\_TSIDR1 字段描述**

字段	描述
7-5 TSID2_TSID0	标准格式标识符 2-0  标准格式标识符由 11 位(TID[10:0])组成。ID10 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。
4 TSRTR	远程发送请求  该标志反映 CAN 帧中远程发送请求位的状态。对于接收缓冲区，它指示接收帧的状态，支持通过软件发送回应帧。对于发送缓冲区，该标志定义要传送的 TRTR 位的置位。  0 数据帧。 1 远程帧。
3 TSIDE	ID 扩展  该标志指示该缓冲区应用的是扩展格式标识符还是标准格式标识符。对于接收缓冲区，该标志按接收值设置，并告知 CPU 如何处理缓冲区标识符寄存器。对于发送缓冲区，该标志告知 MSCAN 应传送何种类型的标识符。  0 标准格式 (11 位)。 1 扩展格式 (29 位)。
Reserved	此字段为保留字段。

### 32.3.35 发送扩展标识符寄存器 2 (MSCAN\_TEIDR2)

扩展格式标识符的寄存器总共有 32 位：TEID[28:0]、TSRR、TEIDE 和 TRTR。

地址: 4002\_4000h 基准 + 32h 偏移 = 4002\_4032h

位 读写	7	6	5	4		3	2	1	0
					TEID14_TEID7				
复位	x*	x*	x*	x*		x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

#### MSCAN\_TEIDR2 字段描述

字段	描述
TEID14_TEID7	扩展格式标识符 14-7  扩展格式标识符由 29 位(TEID[28:0])组成。TEID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。

### 32.3.36 发送扩展标识符寄存器 3 (MSCAN\_TEIDR3)

扩展格式标识符的寄存器总共有 32 位：TEID[28:0]、TSRR、TEIDE 和 TRTR。

地址: 4002\_4000h 基准 + 33h 偏移 = 4002\_4033h

位 读写	7	6	5	4		3	2	1	0
					TEID6_TEID0				TERTR
复位	x*	x*	x*	x*		x*	x*	x*	x*

\* 注:

- x = 复位时未定义。

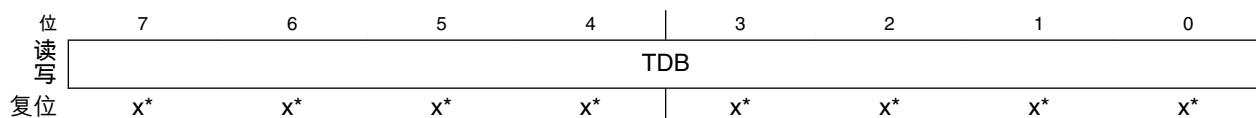
#### MSCAN\_TEIDR3 字段描述

字段	描述
7-1 TEID6_TEID0	扩展格式标识符 6-0  扩展格式标识符由 29 位(TEID[28:0])组成。TEID28 是最高有效位，在仲裁程序中，它最先通过 CAN 总线发送。二进制数最小的标识符，其优先级最高。
0 TERTR	远程发送请求  该标志反映 CAN 帧中远程发送请求位的状态。对于接收缓冲区，它指示接收帧的状态，支持通过软件发送回应帧。对于发送缓冲区，该标志定义要传送的 TRTR 位的置位。  0 数据帧。 1 远程帧。

### 32.3.37 发送扩展数据段寄存器 N (MSCAN\_TEDSRn)

8 个数据段寄存器（各有位 TDB[7:0]）包含要发送的数据。要发送的字节数由对应 TDLR 寄存器中的数据长度码决定。

Address: 4002\_4000h base + 34h offset + (1d × i), where i=0d to 7d



\* 注:

- x = 复位时未定义。

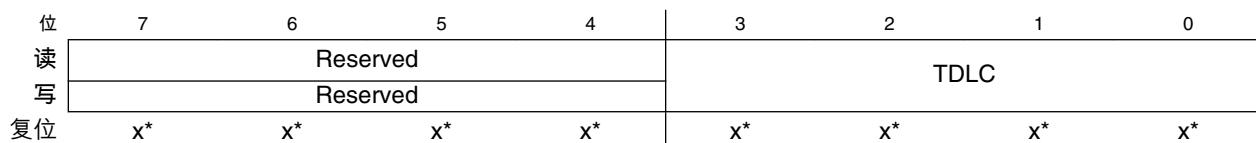
**MSCAN\_TEDSRn 字段描述**

字段	描述
TDB	数据位 要接收的数据

### 32.3.38 发送数据长度寄存器 (MSCAN\_TDLR)

该寄存器保存 CAN 帧的数据长度字段。

地址: 4002\_4000h 基准 + 3Ch 偏移 = 4002\_403Ch



\* 注:

- x = 复位时未定义。

**MSCAN\_TDLR 字段描述**

字段	描述
7–4 Reserved	此字段为保留字段。
TDLC	数据长度码位  数据长度码包含相应报文的字节数（数据字节计数）。发送远程帧时，数据长度码按编程值发送，而发送的数据字节数始终是 0。对于一个数据帧，数据字节数范围是 0 到 8。  0000 0 0001 1

下一页继续介绍此表...

**MSCAN\_TDRL 字段描述 (继续)**

字段	描述
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
其他	保留

**32.3.39 发送缓冲区优先级寄存器 (MSCAN\_TBPR)**

该寄存器定义相关报文缓冲区的本地优先级。本地优先级用于 MSCAN 的内部优先排序处理，最小二进制数的优先级最高。MSCAN 实施了如下的内部优先排序机制：

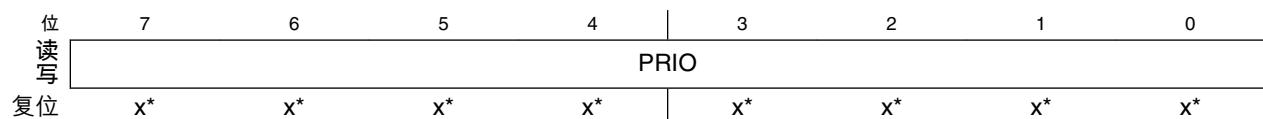
- 在传送 SOF (帧开始) 的前一刻，所有已清除 TXEx 标志的发送缓冲区都要参与优先排序。
- 本地优先级字段最低的发送缓冲区将获得优先安排。

若多个缓冲区具有相同的最低优先级字段，则索引号较低的报文缓冲区胜出。

**注**

当 TXEx 标志置位且在 CANTBSEL 中选中对应的发送缓冲区时，可随时对该寄存器进行读写操作。

地址: 4002\_4000h 基准 + 3Dh 偏移 = 4002\_403Dh



\* 注:

- x = 复位时未定义。

**MSCAN\_TBPR 字段描述**

字段	描述
PRIO	优先级 相关报文缓冲区的本地优先级。

### 32.3.40 发送时间标志寄存器高位 (MSCAN\_TTSRH)

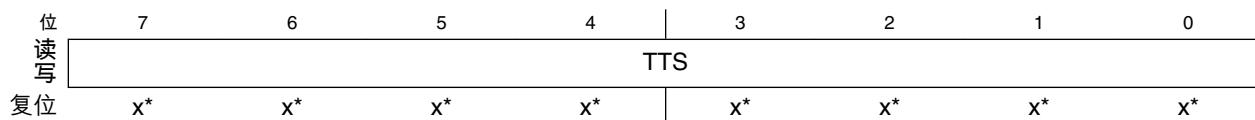
如果 TIME 位使能，则在 CAN 总线上出现有效报文的 EOF 之后，MSCAN 会立即将时间标志写入有效发送缓冲区中的相应寄存器。对于发送，CPU 只能在相应的发送缓冲区空标志置位后读取时间标志。

用作时间标志的定时器值来自一个自由运行的 CAN 内部位时钟。MSCAN 不会指示定时器溢出。该定时器在初始化模式期间复位(所有位都置 0)。CPU 只能对时间标志寄存器进行读操作。

#### 注

对于发送缓冲区，当 TXEx 标志置位且在 CANTBSEL 中选中对应的发送缓冲区时，可以随时对该寄存器进行读操作。

地址: 4002\_4000h 基准 + 3Eh 偏移 = 4002\_403Eh



\* 注:

- x = 复位时未定义。

#### MSCAN\_TTSRH 字段描述

字段	描述
TTS	时间标志 时间标志 7 至 0。

### 32.3.41 发送时间标志寄存器低位 (MSCAN\_TTSRL)

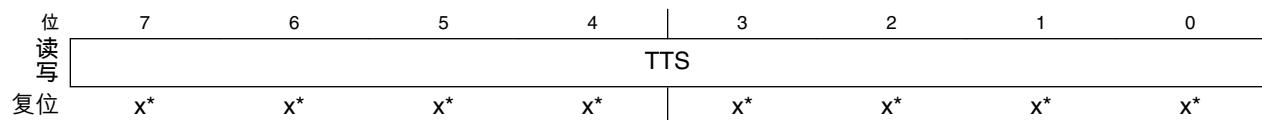
如果 TIME 位使能，则在 CAN 总线上出现有效报文的 EOF 之后，MSCAN 会立即将时间标志写入有效发送缓冲区中的相应寄存器。对于发送，CPU 只能在相应的发送缓冲区空标志置位后读取时间标志。

用作时间标志的定时器值来自一个自由运行的 CAN 内部位时钟。MSCAN 不会指示定时器溢出。该定时器在初始化模式期间复位(所有位都置 0)。CPU 只能对时间标志寄存器进行读操作。

#### 注

当 TXEx 标志置位且在 CANTBSEL 中选中对应的发送缓冲区时，可随时对该寄存器进行读写操作。

地址: 4002\_4000h 基准 + 3Fh 偏移 = 4002\_403Fh



\* 注:

- x = 复位时未定义。

### MSCAN\_TTSRL 字段描述

字段	描述
TTS	时间标志 时间标志 15 至 8。

## 32.4 功能说明

### 32.4.1 报文存储

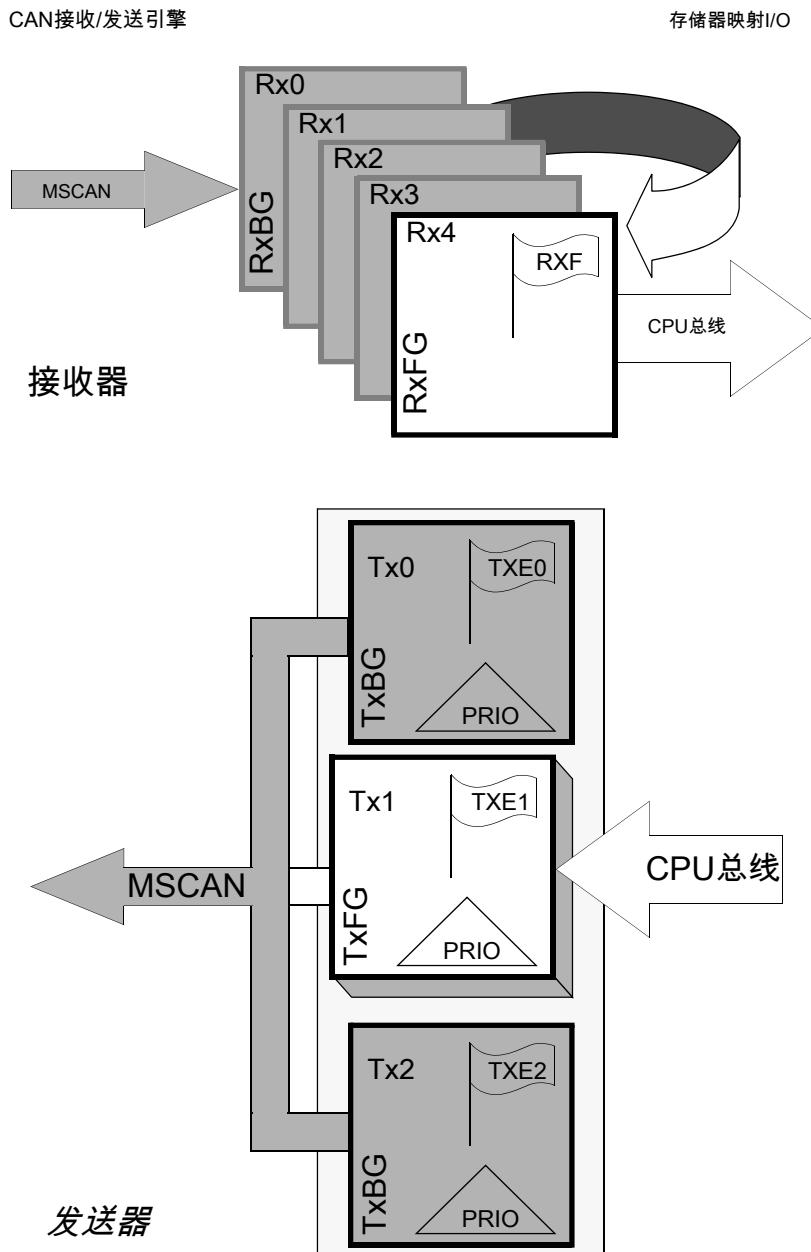


图 32-4. 报文缓冲区组织的用户模型

MSCAN 可简化复杂的报文存储系统，该系统满足各种网络应用的要求。

### 32.4.2 报文发送后台

现代应用层软件建立在两个基本假设之上：

- 任何 CAN 节点都能在不释放两条报文之间 CAN 总线的情况下发出一连串预定报文。这些节点在发出先前的报文后会立即为该 CAN 总线仲裁，并且仅在仲裁丢失的情况下释放 CAN 总线。
- 任何 CAN 节点中的内部报文队列都是如此组织的，即如果有一条以上报文准备要发送，则首先发送优先级最高的报文。

以上项目中描述的行为不能在单一发送缓冲区实现。在发出先前的报文后，必须立即重新加载该缓冲区。该加载过程持续时间有限，并且必须在内置帧序列(IFS)中完成，这样才能发送一连串不中断的报文。即使这对受限的 CAN 总线速度可行，也需要 CPU 以短延迟对发送中断作出反应。

双缓冲区方案使发送缓冲区的重新加载与报文的实际发送脱钩，从而降低 CPU 的反应性要求。如果在完成报文发送的同时 CPU 重新加载第二个缓冲区，则可能会引发问题。之后由于没有准备好要发送报文的缓冲区，CAN 总线将被释放。

无论如何，要满足上述第一个要求至少需要三个发送缓冲区。MSCAN 有三个发送缓冲区。

第二个要求需要某种内部优先次序，该次序由 MSCAN 根据“本地优先级”概念进行实施。

### 32.4.3 发送结构

MSCAN 三重发送缓冲区方案通过允许提前设置多条报文来优化实时性能。三个缓冲区的安排如图 32-4 所示。

三个缓冲器都有一个 13 字节数据结构，类似于接收缓冲器的大致结构。其他发送缓冲器优先寄存器(TBPR)包含一个 8 位的本地优先字段(PRIO)。如果需要，剩余的两个字节用于报文的时间标记。

要发送报文，CPU 必须要识别一个由一组发送缓冲区空(TXEx)标志指示的可用发送缓冲器。如果发送缓冲器可用，那么 CPU 必须通过对 CANTBSEL 寄存器进行写操作作为该缓冲器设置一个指针。这样对应的缓冲器在 CANTXFG 地址空间中就是可访问的。与 CANTBSEL 寄存器相关的算法特性可简化发送缓冲器选择。此外，因为只有一个地址区适用于发送过程，并且所需的地址空间也得到最大限度降低，所以该方案还可简化处理程序软件。

CPU 随后将标识符、控制位和数据内容存储到其中的某个发送缓冲器中。最后，通过清零相关 TXE 标志，缓冲器被标志为准备用于发送。

MSCAN 之后将报文调度为用于发送，并通过设置相关 TXE 标志，发出缓冲器成功发送的信号。当 TXEx 置位并可用于驱动应用软件去重载缓冲器时，一个发送中断被产生<sup>3</sup> TXEx 置位且可用于驱动应用软件以重新加载缓冲器时。

CAN 总线变为可用于仲裁时，如果将一个以上的缓冲器调度用于发送，那么 MSCAN 会使用这三个缓冲器的本地优先级设置来决定优先次序。为此，每个发送缓冲器都有一个 8 位本地优先级字段(PRIO)。应用软件在该报文被设置时对该字段进行编程。本地优先级反映的是该特定报文相对于正从该节点发送的报文组的优先级。PRIO 字段的最低二进制值被定义为最高优先级。每当 MSCAN 为 CAN 总线进行仲裁时，都会发生内部调度流程。在发生发送错误后，也是如此。

高优先级报文由应用软件调度时，可能需要在其中某个发送缓冲器中中止较低优先级报文。由于无法中止处于发送过程中的报文，因此用户必须通过设置相应的中止请求位(ABTRQ)来请求中止。MSCAN 随后同意该请求，可能的方法为：

1. 在 CANTAAK 寄存器中设置相应的中止应答标志(ABTAK)。
2. 设置关联的 TXE 标志以释放缓冲器。
3. 生成发送中断。发送中断发送处理程序软件可根据 ABTAK 标志的设置确定报文是被中止(ABTAK = 1)还是被发送(ABTAK = 0)。

### 32.4.4 接收结构

接收的报文存储在五级输入 FIFO。五个报文缓冲区被交替地映射入单个存储器区域。后台接收缓冲区(RxBG)仅与 MSCAN 关联，但前台接收缓冲区(RxFG)可由 CPU 寻址。因为仅有一个地址区域适用于接收流程，所以该方案可简化处理程序软件。

所有接收缓冲区均有 15 字节大小可存储 CAN 控制位、标识符（标准或扩展）、数据内容和时间标志（如使能）。

接收器完整标志(RXF)用信号通知前台接收缓冲区的状态。该缓冲区包含相匹配标识符的正确接收报文时，该标志置位。

接收时，对每一报文进行检查以查看其是否通过滤波器，并将其同时写入活动的 RxBG。成功接收有效报文之后，MSCAN 将 RxBG 的内容移入接收器 FIFO，使 RXF 标志置位，并生成接收中断<sup>4</sup> 至 CPU。用户接收处理程序必须从 RxFG 读取接收的报文，然后复位 RXF 标志以应答中断和释放前台缓冲器。可紧跟在 CAN 帧的 IFS 字段后的新报文被接收到下一可用的 RxBG。如果 MSCAN 在其 RxBG(错误的标识符、发送错误等) 中接收无效的报文，该缓冲区的实际内容将由下一报文重写。该缓冲区随后将不会移位进入 FIFO。

- 
3. 发送中断仅在未屏蔽时发生。轮询方案在 TXEx 上也适用。
  4. 接收中断仅在未屏蔽的情况下发生。还可将轮询方案应用到 RXF 上。

MSCAN 模块处于发送状态时，MSCAN 接收其自身发送到后台接收缓冲区 RxBG 的报文，但不会移入接收器 FIFO，生成接收中断或在 CAN 总线上应答其自身报文。此规则的异常情况是：MSCAN 处于回环模式时如同处理所有其他传入报文一样处理其自身报文。MSCAN 在丢失仲裁的情况下接收其自身发送的报文。如果丢失仲裁，则必须准备将 MSCAN 变为接收器。

当所有位于 FIFO 的接收报文缓冲区由带有接受标识符的正确接收报文填充，且另一报文从使用带有接受标识符的 CAN 总线正确接收时，会发生溢出情况。如果使能，会放弃后一则报文并生成带溢出指示的错误中断。在填满接收器 FIFO 时，MSCAN 仍然能够发送报文，但会放弃所有传入的报文。只要位于 FIFO 中的接收缓冲区再次可用，就将接受新的有效报文。

### 32.4.5 标识符验收滤波器

MSCAN 标识符验收寄存器定义标准或扩展标识符 (ID[10:0] 或 ID[28:0]) 的可接受模式。任何这些位均可在 MSCAN 标识符屏蔽寄存器中标记为“无关”位。

滤波器命中通过设置接收缓冲器满标志位和三个 CANIDAC 寄存器位去指示应用程序。这些标识符命中标志位(IDHIT[2:0])清楚地识别导致验收的滤波器区段。它们简化了应用程序的任务以便识别接收器中断的原因。如果发生一个以上的命中(两个或多个滤波器匹配)，则较低的命中具有优先级。

已引入一个极为灵活的可编程通用标识符验收滤波器以减轻 CPU 中断负荷。滤波器经编程后可在四个不同的模式中工作：

- 两个标识符验收滤波器，每个都将用于：
  - 扩展标识符的完整 29 位和 CAN 2.0B 帧的下列位：
    - 远程发送请求(RTR)
    - 标识符扩展(IDE)
    - 替代远程请求(SRR)
  - 标准标识符的 11 位以及 CAN 2.0A/B 报文的 RTR 位和 IDE 位。该模式针对具有完整长度且兼容 CAN 2.0B 的扩展标识符实施两个滤波器。尽管该模式可用于标准标识符，但仍建议使用四个或八个标识符验收滤波器。

[图 32-5](#) 显示首个 32 位滤波器组(CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3)如何产生滤波器 0 命中。类似地，第二个滤波器组(CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7)产生滤波器 1 命中。

- 四个标识符验收滤波器，每个都将用于：
  - 扩展标识符的 14 个最高有效位以及 CAN 2.0B 报文的 SRR 位和 IDE 位。
  - 标准标识符的 11 位以及 CAN 2.0A/B 报文的 RTR 位和 IDE 位。[图 32-6](#) 显示首个 32 位滤波器组(CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3)如何产生过滤器 0 和 1 命中。类似地，第二个滤波器组(CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7)产生滤波器 2 和 3 命中。

- 八个标识符验收滤波器，每个都将用于标识符的前 8 位。该模式实施八个独立滤波器，用于兼容 CAN 2.0A/B 的标准标识符或兼容 CAN 2.0B 的扩展标识符的前 8 位。[图 32-7](#) 显示首个 32 位滤波器组(CANIDAR0-CANIDAR3, CANIDMR0-CANIDMR3)如何产生滤波器 0 到 3 的命中。类似地，第二个滤波器组(CANIDAR4-CANIDAR7, CANIDMR4-CANIDMR7)产生滤波器 4 到 7 的命中。
- 封闭式滤波器。无 CAN 报文复制到前台缓冲区 RxFG，且 RXF 标志从不置位。

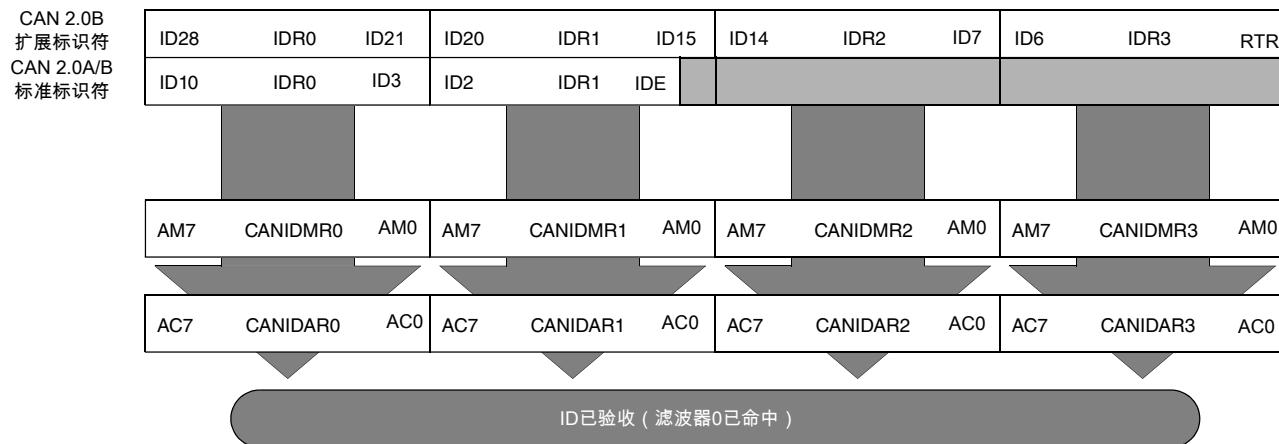


图 32-5. 32 位可屏蔽标识符验收滤波器

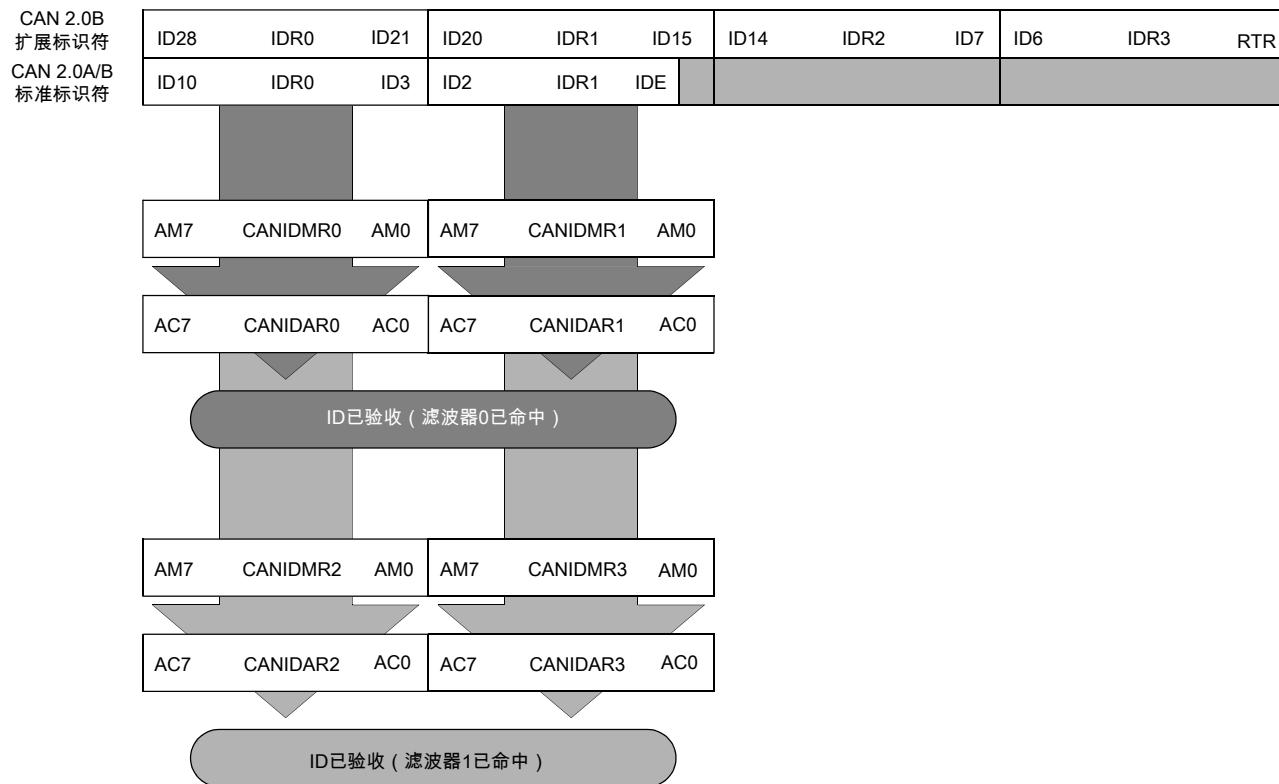


图 32-6. 16 位可屏蔽标识符验收滤波器

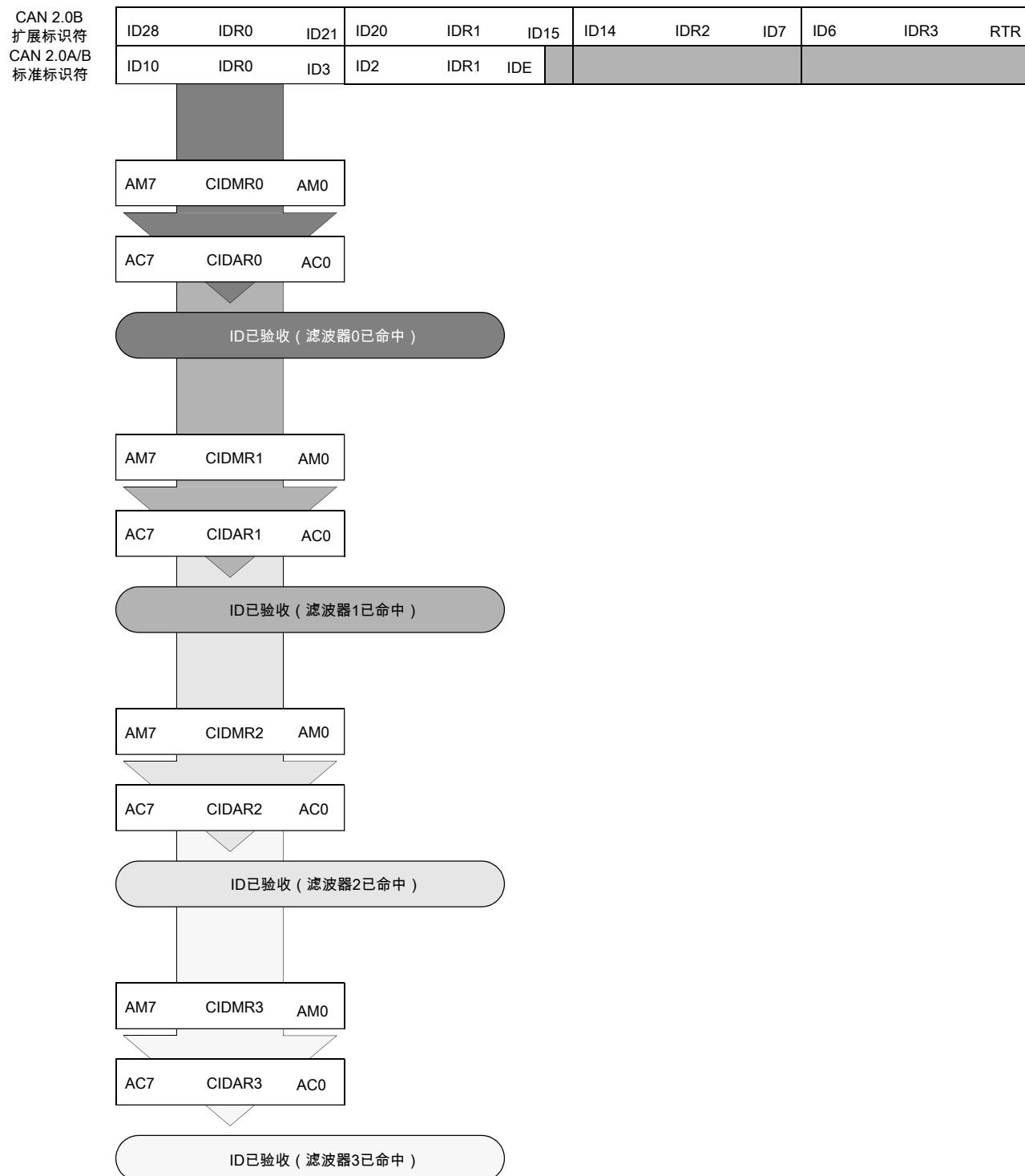


图 32-7. 8 位可屏蔽标识符验收滤波器

### 32.4.5.1 协议违规保护

MSCAN 可防止用户在进行错误编程时意外违反 CAN 协议。保护逻辑包含下列特性：

- 不能对接受和发送错误计数器进行写操作或其他方面的处理。
- MSCAN 在线时不能对控制 MSCAN 配置的所有寄存器进行改动。MSCAN 必须处于初始化模式。CANCTL0/CANCTL1 寄存器对应的 INITRQ/INITAK 握手位用作锁定以保护下列寄存器：
  - MSCAN 控制 1 寄存器(CANCTL1)
  - MSCAN 总线定时寄存器 0 和 1 (CANBTR0, CANBTR1)
  - MSCAN 标识符验收控制寄存器(CANIDAC)
  - MSCAN 标识符验收寄存器(CANIDAR0-CANIDAR7)
  - MSCAN 标识符屏蔽寄存器(CANIDMR0-CANIDMR7)
- MSCAN 转入掉电模式或初始化模式时，TXCAN 会被立即强制进入隐性状态。
- MSCAN 使能位(CANE)在正常系统工作模式下只能写入一次，可进一步防止对 MSCAN 的无意禁用。

### 32.4.5.2 时钟系统

下图显示 MSCAN 时钟生成电流的结构。

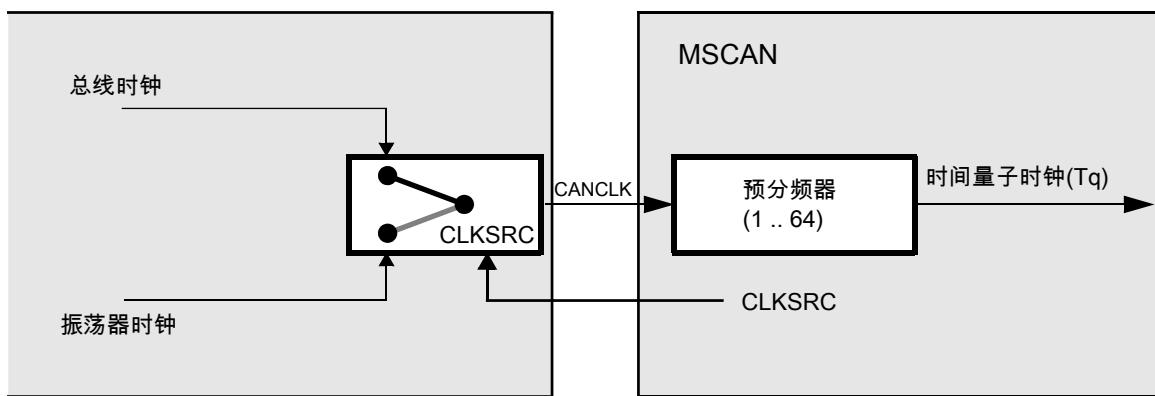


图 32-8. MSCAN 计时方案

CANCTL1 寄存器中的时钟源位(CLKSRC)定义内部 CANCLK 是连接到晶振（振荡器时钟）还是总线时钟。

时钟源必须选择有严格公差要求的振荡器（最大 0.4%）以满足 CAN 协议。此外，对于高 CAN 总线率(1 Mbps)，要求时钟的占空比在 45% 到 55% 之间。

如果总线时钟从 PLL 生成，则出于抖动方面的考虑，建议选择振荡器时钟而非总线时钟，尤其在处于更快的 CAN 总线速率时。

对于无时钟和复位生成器(CRG)的微控制器，从晶振（振荡器时钟）驱动 CANCLK。

可编程的预分频器从 CANCLK 生成时间量子( $T_q$ )时钟。时间量子是 MSCAN 处理的原子单位时间。

$$\Gamma_q = f_{CANCLK} / (\text{预分频器值})$$

位时间可细分为三个区段，如 Bosch CAN 2.0A/B 规范所述。(参见图 32-9)：

- SYNC\_SEG：该区段有一个固定长度的时间量子。在该部分内预期发生信号边沿。
- 时间段 1：该区段包括 PROP\_SEG 和 CAN 标准的 PHASE\_SEG1。通过将参数 TSEG1 设置为由 4 到 16 个时间量子组成可对该区段进行编程。
- 时间段 2：该区段代表 CAN 标准的 PHASE\_SEG2。通过将参数 TSEG2 设置为 2 到 8 个时间量子长可对该区段进行编程。

$$\text{比特率} = f_{\Gamma_q} / (\text{时间量子数})$$

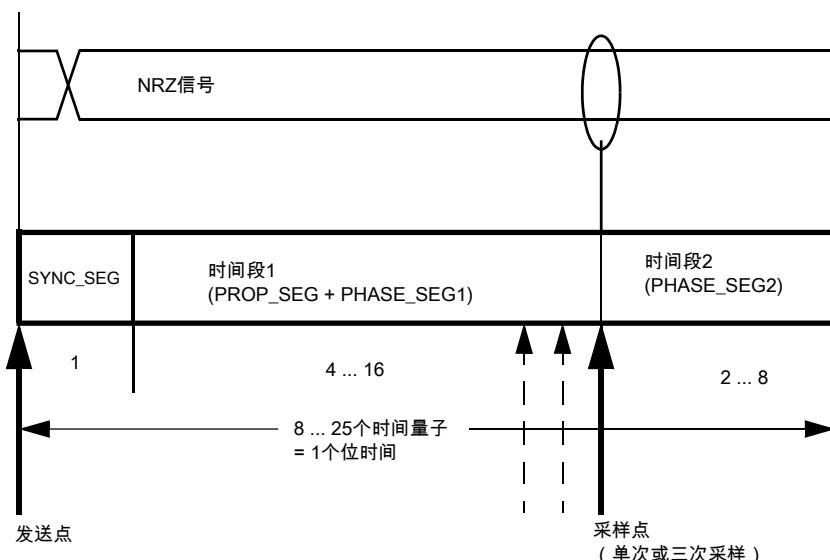


图 32-9. 位时间内的区段

表 32-4. 时间段句法

句法	说明
SYNC_SEG	系统期望在该周期内在 CAN 总线上发生转换。
发送点	处于发送模式的节点于此时将新值传输到 CAN 总线。
采样点	处于接收模式的节点于此时对 CAN 总线采样。如果选择每位三样本选项，那么该点标志第三个样本的位置。

通过设置 SJW 参数可将同步跳转宽度(有关详情，请参见 Bosch CAN 2.0A/B 规范)编程为位于 1 到 4 个时间量子范围内。

设定 SYNC\_SEG、TSEG1、TSEG2 和 SJW 参数的方法是对 MSCAN 总线定时寄存器(CANBTR0, CANBTR1)进行编程。

表 32-5 概述了符合 Bosch CAN 2.0A/B 规范的段设置以及相关参数值。

表 32-5. 符合 Bosch CAN 2.0A/B 规范的位时间段设置

时间段 1	TSEG1	时间段 2	TSEG2	同步跳转宽度	SJW
5 .. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

### 32.4.6 低功耗选项

如果 MSCAN 禁用 (CANE = 0)，则会停止 MSCAN 时钟以便节能。

如果 MSCAN 使能 (CANE = 1)，则与正常模式相比，MSCAN 有另外两个可减少功耗的模式：睡眠模式和降耗模式。在睡眠模式中，降低功耗的方法是停止所有时钟（那些从 CPU 侧访问寄存器的除外）。在降耗模式中，所有时钟都停止且无功耗。

表 32-6 总结了 MSCAN 和 CPU 模式的组合。根据 CSWAI 和 SLPRQ/SLPAK 位上给定的设置会进入特定的模式组合。

表 32-6. CPU 与 MSCAN 工作模式

CPU 模式	MSCAN 模式			
	正常	降低功耗		
		睡眠	降耗	禁用(CANE=0)
RUN	<ul style="list-style-type: none"> <li>CSWAI = X<sup>1</sup></li> <li>SLPRQ = 0</li> <li>SLPAK = 0</li> </ul>	<ul style="list-style-type: none"> <li>CSWAI = X</li> <li>SLPRQ = 1</li> <li>SLPAK = 1</li> </ul>		<ul style="list-style-type: none"> <li>CSWAI = X</li> <li>SLPRQ = X</li> <li>SLPAK = X</li> </ul>
等待	<ul style="list-style-type: none"> <li>CSWAI = 0</li> <li>SLPRQ = 0</li> <li>SLPAK = 0</li> </ul>	<ul style="list-style-type: none"> <li>CSWAI = 0</li> <li>SLPRQ = 1</li> <li>SLPAK = 1</li> </ul>	<ul style="list-style-type: none"> <li>CSWAI = 1</li> <li>SLPRQ = X</li> <li>SLPAK = X</li> </ul>	<ul style="list-style-type: none"> <li>CSWAI = X</li> <li>SLPRQ = X</li> <li>SLPAK = X</li> </ul>
停止			<ul style="list-style-type: none"> <li>CSWAI = X</li> <li>SLPRQ = X</li> <li>SLPAK = X</li> </ul>	<ul style="list-style-type: none"> <li>CSWAI = X</li> <li>SLPRQ = X</li> <li>SLPAK = X</li> </ul>

1. X 表示无关紧要。

### 32.4.6.1 Run 模式下的操作

如表 32-6 所示，CPU 处于 Run 模式时，仅 MSCAN 睡眠模式可用作低功耗选项。

### 32.4.6.2 Wait 模式下的操作

WAI 指令将 MCU 置于低功耗待机模式。如果 CSWAI 位置位，那么由于 CPU 时钟停止，因而可在降耗模式下节省额外电能。离开该降耗模式后，MSCAN 重启并再次进入正常模式。

CPU 处于 Wait 模式时，MSCAN 可在正常模式下工作并生成中断（通过 Background Debug 模式可访问寄存器）。

### 32.4.6.3 Stop 模式下的操作

STOP 指令将 MCU 置于低功耗待机模式。在 Stop 模式下，MSCAN 都被置于降耗模式，而无论 SLPRQ/SLPAK 位和 CSWAI 位的值如何（表 32-6）。

### 32.4.6.4 MSCAN 正常模式

这是非节能模式。使能 MSCAN 会使禁用模式中的模块进入正常模式。在该模式下，模块可处于初始化模式或退出初始化模式。请参见 [MSCAN 初始化模式](#)。

### 32.4.6.5 MSCAN 睡眠模式

CPU 可通过在 CANCTL0 寄存器中使 SLPRQ 位的电平变为有效值而请求 MSCAN 进入此低功耗模式。MSCAN 进入睡眠模式的时间取决于固定同步延迟和其当前的活动：

- 如果有一个或多个报文缓冲器经调度后用于发送( $\text{TXEx} = 0$ )，则 MSCAN 将继续发送，直到所有发送报文缓冲器为空 ( $\text{TXEx} = 1$ ，成功发送或中止)，然后进入睡眠模式。
- 如果 MSCAN 正处于接收状态，其将继续接收并当 CAN 总线下一次进入空闲状态时进入睡眠模式。
- 如果 MSCAN 既不发送也不接收，就会立即进入睡眠模式。

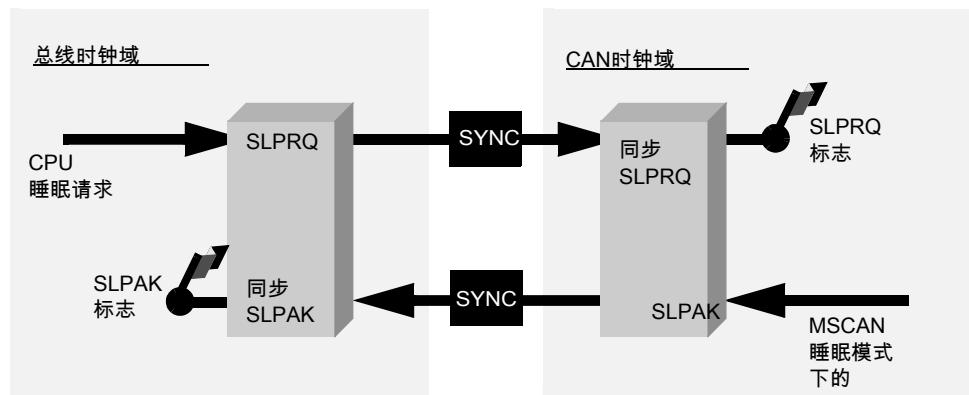


图 32-10. 睡眠请求/应答周期

### 注

应用软件必须避免设置发送 (通过清零一个或多个 TXEx 标志) 并立即请求睡眠模式 (通过设置 SLPRQ)。MSCAN 开始发送还是直接进入睡眠模式取决于操作的确切顺序。

如果睡眠模式处于有效状态，则 SLPRQ 位和 SLPAK 位置位 (图 32-10)。应用软件必须将 SLPAK 用作请求(SLPRQ)进入睡眠模式的握手指示。

处于睡眠模式 ( $SLPRQ = 1$  且  $SLPAK = 1$ ) 时，MSCAN 停止其内部时钟。但是，允许从 CPU 一侧访问寄存器的时钟继续运行。

如果 MSCAN 处于总线关闭状态，则其会因时钟停止而停止对 11 个连续隐性位 128 次出现进行计数。TXCAN 保持隐性状态。如果  $RXF = 1$ ，则可读取报文和清零 RXF。将新报文移位到接收器 FIFO (RxFG) 的前台缓冲区中在睡眠模式下不发生。

可以访问发送缓冲器和清零关联的 TXE 标志位。在睡眠模式下不发生报文中止。

如果未使 CANCTL0 中 WUPE 位的电平变为有效，则 MSCAN 将屏蔽其在 CAN 上检测的任何活动。RXCAN 因此在内部被保持在隐性状态。这将 MSCAN 锁定在睡眠模式中。WUPE 在进入睡眠模式生效之前必须置位。

MSCAN 能够离开睡眠模式 (唤醒) 的唯一条件是：

- CAN 总线活动发生并且  $WUPE=1$  或
- CPU 清零 SLPRQ 位

### 注

CPU 在睡眠模式 ( $SLPRQ = 1$  和  $SLPAK=1$ ) 处于有效状态之前无法清零 SLPRQ 位。

唤醒之后，MSCAN 等待 11 个连续的隐性位，以便与 CAN 总线同步。因此，如果 CAN 帧唤醒 MSCAN，则不接收此帧。

接收报文缓冲器 (RxFG 和 RxBG) 包含其在进入睡眠模式之前是否被接收的报文。一旦唤醒就会执行所有挂起操作；RxBG 复制到 RxFG、报文中止和报文发送。如果在退出睡眠模式之后 MSCAN 保持在总线关闭状态，其会继续对 11 个连续隐性位 128 次出现进行计数。

### 32.4.6.6 MSCAN 降耗模式

MSCAN 处于降耗模式([表 32-6](#))的条件是：

- CPU 处于 Stop 模式或
- CPU 处于 Wait 模式且 CSWAI 位置位

进入降耗模式时，MSCAN 会立即停止所有正在进行的发送和接收，从而而可能导致违反 CAN 协议。为防止 CAN 总线系统发生违反上述规则的致命后果，MSCAN 立即驱动 TXCAN，使其进入隐性状态。

#### 注

用户负责确保在进入降耗模式时 MSCAN 不处于有效状态。建议的步骤是在执行 STOP 或 WAI 指令前使 MSCAN 进入睡眠模式（如果 CSWAI 置位）。否则，中止正在进行的报文可能会导致错误情况并影响其他 CAN 总线器件。

在降耗模式下，会停止所有时钟且不访问寄存器。如果 MSCAN 在降耗模式变为有效状态前不处于睡眠模式，则该模块在上电后执行内部恢复周期。这在该模块再次进入正常模式之前会导致某个固定延迟。

### 32.4.6.7 禁用模式

MSCAN 在退出复位后处于禁用模式(CANE=0)。所有模块时钟都停止以便省电，但仍可根据规定访问寄存器映像。

### 32.4.6.8 可编程唤醒功能

只要检测到 CAN 总线活动，便可将 MSCAN 编程为从睡眠或降耗模式中唤醒（参见 MSCAN 控制寄存器 0 (CANCTL0)中的控制位 WUPE。）对现有 CAN 总线动作的敏感度是可以修改的，方法是：将低通过滤波器功能应用到 RXCAN 输入线路。

该特性可用于防止 MSCAN 因 CAN 总线线路上的短时毛刺而唤醒。这类毛刺可能由噪音环境中的电磁干扰等情况而引起。

### 32.4.7 复位初始化

每一独立位的复位状态列示在详细介绍所有寄存器及其位字段的[存储器映像和寄存器定义](#)中。

### 32.4.8 中断

本节介绍由 MSCAN 引起的所有中断。它记录使能位和生成的标志。每个中断都分别列示和介绍。

#### 32.4.8.1 中断操作说明

MSCAN 支持四个中断向量（参见 [表 32-7](#)），其中任何一个都可被单独屏蔽。

请参阅设备概述一节以确定专用的中断向量地址。

表 32-7. 中断向量

中断源	CCR 掩码	本地使能
唤醒中断(WUPIF)	1 位	CANRIER (WUPIE)
错误中断中断(CSCIF, OVRIF)	1 位	CANRIER (CSCIE, OVRIE)
接收中断(RXF)	1 位	CANRIER (RXFIE)
发送中断(TXE[2:0])	1 位	CANTIER (TXEIE[2:0])

#### 32.4.8.2 发送中断

三个发送缓冲器中至少有一个为空（未经调度）并且经加载后可将报文调度用于发送。空报文缓冲器的 TXEx 标志置位。

#### 32.4.8.3 接收中断

成功接收一则报文并将其移位到接收器 FIFO 的前台缓冲区(RxFG)中。接收 EOF 符号后立即生成该中断。RXF 标志置位。如果接收器 FIFO 中有多则报文，则只要下一则报文移位到前台缓冲区，RXF 标志就为置位。

### 32.4.8.4 唤醒中断

如果在 MSCAN 睡眠或电源中断模式期间在 CAN 总线上发生活动，则会生成唤醒中断。

#### 注

该中断发生的条件仅限于以下三种：MSCAN 在进入降耗模式之前处于睡眠模式 ( $SLPRQ = 1$  且  $SLPAK = 1$ )、唤醒选项使能( $WUPE = 1$ )以及唤醒中断使能( $WUPIE = 1$ )。

### 32.4.8.5 错误中断

如果发生接收器 FIFO 溢出、错误、警告或者总线关闭情况，则会生成错误中断。MSCAN 接收器标志寄存器(CANRFLG)指示下列情况之一：

- 溢出—[接收结构](#) 中所述的接收器 FIFO 溢出情况。
- CAN 状态变更—发送和接收错误计时器的实际值控制 MSCAN 的 CAN 总线状态。只要错误计数器直接进入临界范围(发送/接收警告、发送/接收错误、总线关闭)，MSCAN 就会标记错误情况。引起错误情况的状态变更由 TSTAT 标志和 RSTAT 标志指示。

### 32.4.8.6 中断应答

中断与 MSCAN 接收器标志位寄存器(CANRFLG)或 MSCAN 发送器标志位寄存器(CANTFLG)中的一个或多个状态标志位直接关联。只要其中一个相应标志置位，中断就会挂起。必须在中断处理程序中复位 CANRFLG 和 CANTFLG 中的标志以使能下一次中断信号。复位这些标志的方法是将 1 写入相应的位位置。如果相关条件生效，则无法清零该标志。

#### 注

必须保证 CPU 只清零引起当前中断的位。因此，位操作指令(BSET)不得用于清零中断标志。这些指令可能会意外清零在进入当前中断服务程序后置位的中断标志。

### 32.4.9 初始化/应用信息

#### 32.4.9.1 MSCAN 初始化

初始启动离开复位状态的 MSCAN 模块的步骤如下所示：

1. 使 CANE 的电平变为有效
2. 对初始化模式下的配置寄存器进行写操作

### 3. 清零 INITRQ 以退出初始化模式

如果要更改仅在初始化模式下可写入的寄存器配置：

1. 通过在 CAN 总线变为空闲状态后设置 SLPRQ 并等待 SLPACK 的电平变为有效使该模块进入睡眠模式。
2. 进入初始化模式：使 INITRQ 的电平变为有效并等待 INITAK
3. 对初始化模式下的配置寄存器进行写操作
4. 清零 INITRQ 以退出初始化模式并继续

#### 32.4.9.2 总线关闭恢复

总线关闭恢复可由用户配置。可自动或者根据用户请求离开总线关闭状态。

由于向后兼容性原因，MSCAN 在复位后默认为自动恢复。在这种情况下，在 CAN 总线上出现 128 次 11 个连续隐性位后，MSCAN 将再次进入错误激活状态（有关详情，请参见 Bosch CAN 2.0 A/B 规范）。

如果针对用户请求配置 MSCAN (BORM 在 MSCAN 控制寄存器 1 (CANCTL1) 中置位)，那么从总线关闭的恢复在两个独立事件变为有效后启动：

- 监测到 CAN 总线上出现 128 次 11 个连续隐性位
- MSCAN 其他寄存器(CANMISC)中的 BOHOLD 已被用户清除

这两个事件可以任意顺序发生。

# 第 33 章 通用异步收发器(UART)

## 33.1 简介

### 33.1.1 特性

UART 模块的特性包括：

- 全双工标准不归零(NRZ)格式
- 具有单独使能的双缓冲发送器和接收器
- 可编程波特率 (13 位模数分频器)
- 中断驱动或轮询操作：
  - 发送数据寄存器为空且发送完成
  - 接收数据寄存器已满
  - 接收溢出、奇偶错误、成帧错误和噪声错误
  - 空闲接收器检测
  - 接收引脚上的活动边沿
  - 支持 LIN 的分割符检测
- 硬件奇偶生成和校验
- 可编程 8 位或 9 位字符长度
- 可编程 1 位或 2 位停止位
- 由闲置线路或地址标志唤醒接收器
- 可选 13 位分隔字符生成/11 位分隔字符检测
- 可选择发送器输出极性

### 33.1.2 工作模式

有关这些模式下的相关 UART 操作，请参见章节[功能说明](#)：

- 8 位和 9 位数据模式
- Stop 模式操作

- 循环模式
- 单线模式

### 33.1.3 结构框图

下图显示 UART 的发送器部分。

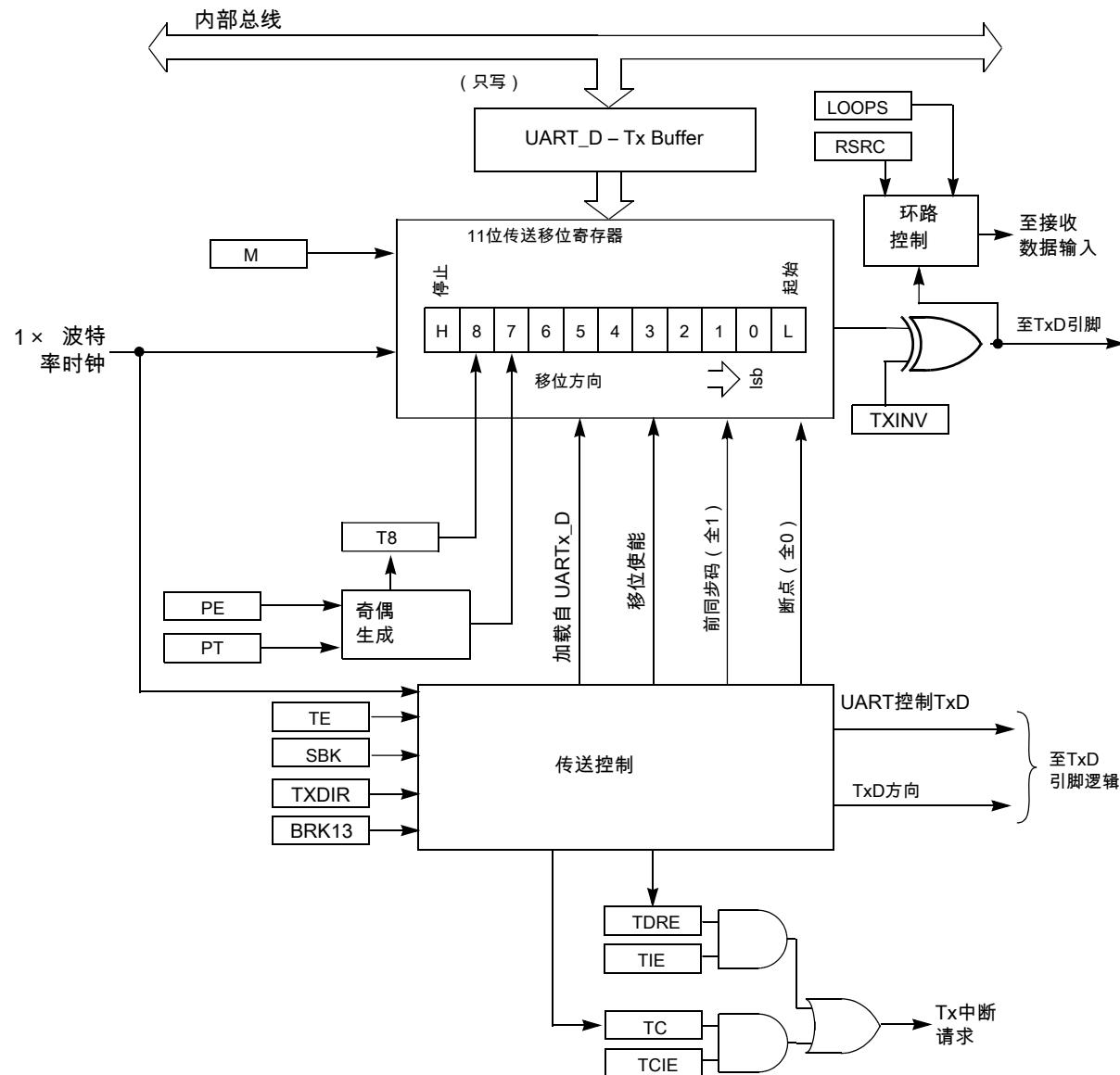


图 33-1. UART 发送器结构框图

下图显示 UART 的接收器部分。

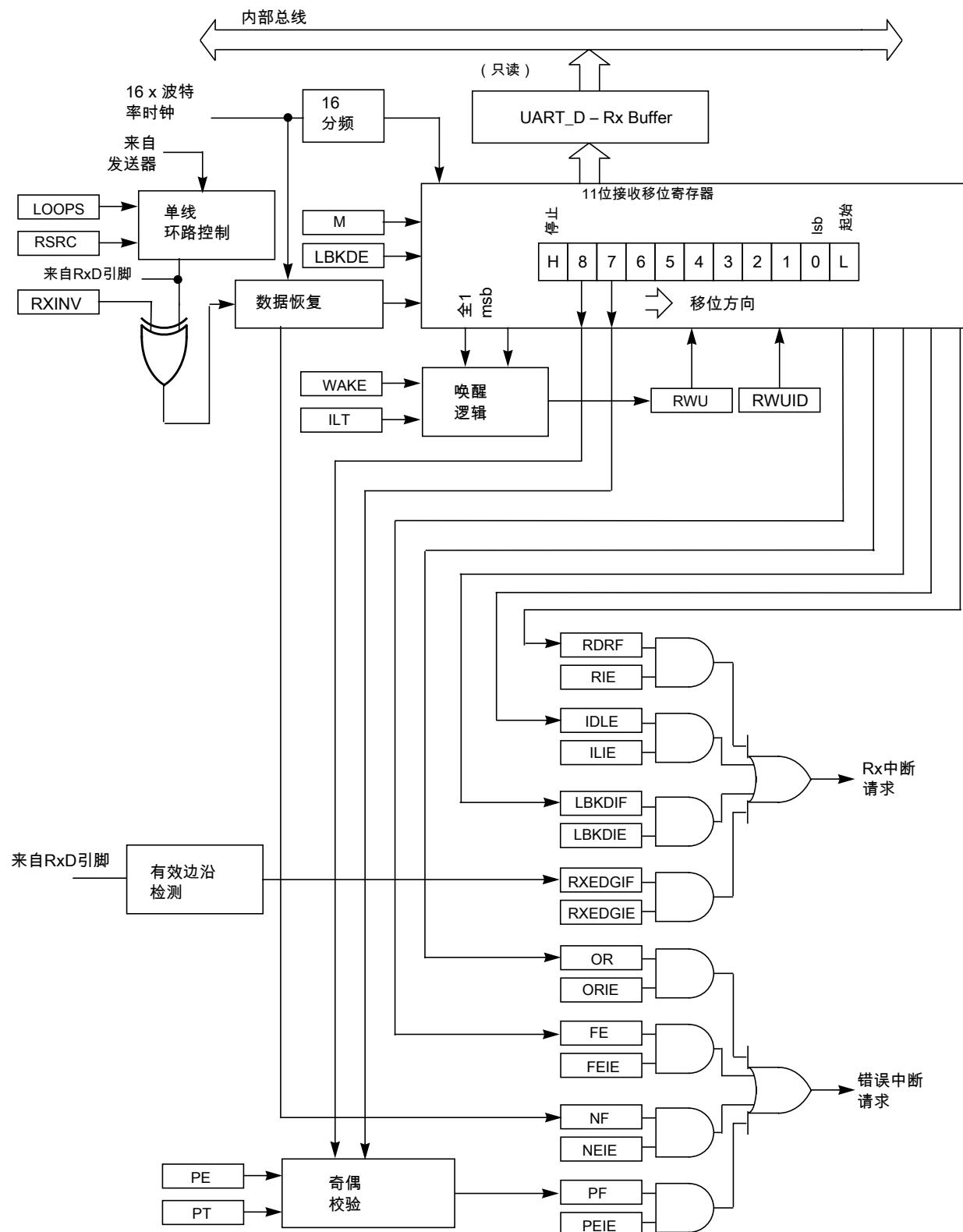


图 33-2. UART 接收器结构框图

## 33.2 UART 信号说明

UART 信号如此表所示。

表 33-1. UART 信号说明

信号	说明	I/O
RxD	接收数据	I
TxD	发送数据	I/O

### 33.2.1 详细信号说明

UART 的详细信号说明如下表所示。

表 33-2. UART—详细信号说明

信号	I/O	说明	
RxD	I	接收数据。接收器的串行数据输入。	
		状态含义	RxD 是被解析为 1 还是 0 取决于位编码方法以及其他配置设置。
		时序	以模块时钟除以波特率所得频率进行采样。
TxD	I/O	发送数据。发送器的串行数据输出。	
		状态含义	TxD 是被解析为 1 还是 0 取决于位编码方法以及其他配置设置。
		时序	根据位编码方法以及其他配置设置，在开始时或位时间内驱动。否则，发送与接收时序无关。

## 33.3 寄存器定义

UART 拥有 8 位寄存器以控制波特率，选择 UART 选项，报告 UART 状态，以及发送/接收数据。

有关所有 UART 寄存器，请参见本文档存储器章节中的直接页面寄存器摘要和寄存器的绝对地址分配。该部分仅按其名称参考寄存器和控制位。Freescale 提供的等同或头文件用于将这些名称转换为正确的绝对地址。

## UART 存储器映射

绝对地址(十六进制)	寄存器名称	宽度(单位:位)	访问	复位值	小节/页
4006_A000	UART 波特率寄存器 : 高位 (UART0_BDH)	8	R/W	00h	<a href="#">33.3.1/609</a>
4006_A001	UART 波特率寄存器 : 低位 (UART0_BDL)	8	R/W	04h	<a href="#">33.3.2/610</a>
4006_A002	UART 控制寄存器 1 (UART0_C1)	8	R/W	00h	<a href="#">33.3.3/611</a>
4006_A003	UART 控制寄存器 2 (UART0_C2)	8	R/W	00h	<a href="#">33.3.4/612</a>
4006_A004	UART 状态寄存器 1 (UART0_S1)	8	R	C0h	<a href="#">33.3.5/613</a>
4006_A005	UART 状态寄存器 2 (UART0_S2)	8	R/W	00h	<a href="#">33.3.6/615</a>
4006_A006	UART 控制寄存器 3 (UART0_C3)	8	R/W	00h	<a href="#">33.3.7/617</a>
4006_A007	UART 数据寄存器 (UART0_D)	8	R/W	00h	<a href="#">33.3.8/618</a>
4006_B000	UART 波特率寄存器 : 高位 (UART1_BDH)	8	R/W	00h	<a href="#">33.3.1/609</a>
4006_B001	UART 波特率寄存器 : 低位 (UART1_BDL)	8	R/W	04h	<a href="#">33.3.2/610</a>
4006_B002	UART 控制寄存器 1 (UART1_C1)	8	R/W	00h	<a href="#">33.3.3/611</a>
4006_B003	UART 控制寄存器 2 (UART1_C2)	8	R/W	00h	<a href="#">33.3.4/612</a>
4006_B004	UART 状态寄存器 1 (UART1_S1)	8	R	C0h	<a href="#">33.3.5/613</a>
4006_B005	UART 状态寄存器 2 (UART1_S2)	8	R/W	00h	<a href="#">33.3.6/615</a>
4006_B006	UART 控制寄存器 3 (UART1_C3)	8	R/W	00h	<a href="#">33.3.7/617</a>
4006_B007	UART 数据寄存器 (UART1_D)	8	R/W	00h	<a href="#">33.3.8/618</a>
4006_C000	UART 波特率寄存器 : 高位 (UART2_BDH)	8	R/W	00h	<a href="#">33.3.1/609</a>
4006_C001	UART 波特率寄存器 : 低位 (UART2_BDL)	8	R/W	04h	<a href="#">33.3.2/610</a>
4006_C002	UART 控制寄存器 1 (UART2_C1)	8	R/W	00h	<a href="#">33.3.3/611</a>
4006_C003	UART 控制寄存器 2 (UART2_C2)	8	R/W	00h	<a href="#">33.3.4/612</a>
4006_C004	UART 状态寄存器 1 (UART2_S1)	8	R	C0h	<a href="#">33.3.5/613</a>
4006_C005	UART 状态寄存器 2 (UART2_S2)	8	R/W	00h	<a href="#">33.3.6/615</a>
4006_C006	UART 控制寄存器 3 (UART2_C3)	8	R/W	00h	<a href="#">33.3.7/617</a>
4006_C007	UART 数据寄存器 (UART2_D)	8	R/W	00h	<a href="#">33.3.8/618</a>

### 33.3.1 UART 波特率寄存器: 高位 (UARTx\_BDH)

该寄存器与 UART\_BDL 一起控制 UART 波特率生成的预分频因子。要更新 13 位波特率设置 SBR[12:0], 首先应写入 UART\_BDH 以缓存新值的高半部分, 然后写入 UART\_BDL。UART\_BDH 中的工作值要等到写入 UART\_BDL 之后才改变。

地址: 基址 基准 + 0h 偏移

位 读写	7	6	5	4		3	2	1	0
	LBKDI	RXEDGIE	SBNS			SBR			
复位	0	0	0	0		0	0	0	0

**UARTx\_BDH 字段描述**

字段	描述
7 LBKDI	LIN 断点检测中断使能 ( 用于 LBKDI )  0 来自 UART_S2[LBKDI] 的硬件中断禁用 ( 使用轮询 )。 1 UART_S2[LBKDI] 标志为 1 时 , 请求硬件中断。
6 RXEDGIE	RxD 输入有效边沿中断使能 ( 用于 RXEDGIE )  0 来自 UART_S2[RXEDGIE] 的硬件中断禁用 ( 使用轮询 )。 1 UART_S2[RXEDGIE] 标志为 1 时 , 请求硬件中断。
5 SBNS	停止位数选择  SBNS 决定数据字符是一个停止位还是两个停止位。  0 一个停止位。 1 两个停止位。
SBR	波特率模数分频因子。  SBR[12:0] 中的 13 位统称为 BR , 用于设置 UART 波特率发生器的模数分频比。清除 BR 时 , UART 波特率发生器禁用以降低供电电流。BR 为 1 - 8191 时 , UART 波特率等于 BUSCLK/(16×BR)。

**33.3.2 UART 波特率寄存器: 低位 (UARTx\_BDL)**

该寄存器与 UART\_BDH 一起控制 UART 波特率生成的预分频因子。要更新 13 位波特率设置[SBR12:SBR0]，首先应写入 UART\_BDH 以缓存新值的高半部分，然后写入 UART\_BDL。UART\_BDH 中的工作值要等到写入 UART\_BDL 之后才改变。

UART\_BDL 复位为非零值，因此在复位后，波特率发生器保持禁用到首次使能接收器或发送器时为止，也就是向 UART\_C2[RE] 或 UART\_C2[TE] 位写入 1。

地址: 基址 基准 + 1h 偏移

位 读写	7	6	5	4		3	2	1	0
SBR									
复位	0	0	0	0		0	1	0	0

**UARTx\_BDL 字段描述**

字段	描述
SBR	波特率模数分频因子  SBR[12:0] 中的这 13 位统称为 BR , 用于设置 UART 波特率发生器的模数分频比。清除 BR 时 , UART 波特率发生器禁用以降低供电电流。BR 为 1 - 8191 时 , UART 波特率等于 BUSCLK/(16×BR)。

### 33.3.3 UART 控制寄存器 1 (UARTx\_C1)

该读/写寄存器控制 UART 系统的各种可选特性。

地址: 基址 基准 + 2h 偏移

位 读写	7 LOOPS	6 UARTSWAI	5 RSRC	4 M	3 WAKE	2 ILT	1 PE	0 PT
复位	0	0	0	0	0	0	0	0

UARTx\_C1 字段描述

字段	描述
7 LOOPS	环路模式选择 选择环路模式或正常 2 引脚全双工模式。LOOPS 置位时，发送器输出内部连接到接收器输入。 0 正常工作 - RxD 和 TxD 使用不同的引脚。 1 环路模式或单线模式，发送器输出内部连接到接收器输入。( 参见 RSRC 位。) UART 不使用 RxD 引脚。
6 UARTSWAI	UART 在等待模式下停止 0 UART 时钟在等待模式下继续运行，UART 因此可用作唤醒 CPU 的中断源。 1 CPU 处于等待模式的同时 UART 时钟冻结。
5 RSRC	接收器源选择 该字段无意义或无效，除非 LOOPS 设置为 1。当 LOOPS 置位时，接收器输入内部连接到 TxD 引脚，RSRC 决定此连接是否同时连接到发送器输出。 0 如果 LOOPS 置位且 RSRC 被清除，则选择内部回环模式，UART 不使用 RxD 引脚。 1 单线 UART 模式，TxD 引脚连接到发送器输出和接收器输入。
4 M	9 位或 8 位模式选择 该字段配置 UART 以 9 位或 8 位数据模式工作。 0 正常 - 起始 + 8 数据位 ( LSB 优先 ) + 停止。 1 接收器和发送器使用 9 位数据字符：起始 + 8 数据位 ( LSB 优先 ) + 第 9 数据位 + 停止。
3 WAKE	接收器唤醒方法选择 该字段选择接收器唤醒方法。 0 空闲线路唤醒。 1 地址标记唤醒。
2 ILT	空闲线路类型选择 该字段置 1 可确保字符结束时的停止位和逻辑 1 位不计数到空闲线路检测逻辑所需的 10 或 11 位逻辑高电平时间。 0 空闲字符位计数开始于起始位之后。 1 空闲字符位计数开始于停止位之后。

下一页继续介绍此表...

### UARTx\_C1 字段描述 (继续)

字段	描述
1 PE	<p>奇偶校验使能</p> <p>使能硬件奇偶生成和校验。使能奇偶校验时，数据字符的最高有效位（第 8 或第 9 数据位）被视为奇偶校验位。</p> <p>0 无硬件奇偶生成或校验。 1 使能奇偶校验。</p>
0 PT	<p>奇偶校验类型</p> <p>奇偶校验使能时(<math>PE = 1</math>)，该字段选择偶数或奇数校验。奇数校验是指数据字符中 1 的总数（包括奇偶校验位）为奇数。偶数校验是指数据字符中 1 的总数（包括奇偶校验位）为偶数。</p> <p>0 偶数校验。 1 奇数校验。</p>

### 33.3.4 UART 控制寄存器 2 (UARTx\_C2)

可随时对该寄存器进行读写操作。

地址: 基址 基准 + 3h 偏移

位 读写	7	6	5	4	3	2	1	0
	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
复位	0	0	0	0	0	0	0	0

### UARTx\_C2 字段描述

字段	描述
7 TIE	<p>用于 TDRE 的发送中断使能</p> <p>0 来自 TDRE 的硬件中断禁用；使用轮询。 1 TDRE 标志为 1 时，请求硬件中断。</p>
6 TCIE	<p>用于 TC 的发送完成中断使能</p> <p>0 来自 TC 的硬件中断禁用；使用轮询。 1 TC 标志为 1 时，请求硬件中断。</p>
5 RIE	<p>用于 RDRF 的接收器中断使能</p> <p>0 来自 S1[RDRF]的硬件中断禁用；使用轮询。 1 S1[RDRF]标志为 1 时，请求硬件中断。</p>
4 ILIE	<p>用于 IDLE 的空闲线路中断使能</p> <p>0 来自 S1[IDLE]的硬件中断禁用；使用轮询。 1 S1[IDLE]标志为 1 时，请求硬件中断。</p>
3 TE	<p>发送器使能</p> <p>要使用 UART 发送器，TE 必须为 1。TE 置位时，UART 将 TxD 引脚强制用作 UART 系统的输出。</p>

下一页继续介绍此表...

## UARTx\_C2 字段描述 (继续)

字段	描述
	<p>当 UART 配置为单线工作(LOOPS = RSRC = 1)时 , TXDIR 控制单一 UART 通信线路 ( TxD 引脚 ) 上的流量方向。</p> <p>在发送进行的同时 , TE 也可将某个空闲字符加入队列 , 方法是将 TE 先清零再置位。</p> <p>将 0 写入 TE 时 , 发送器一直保持对端口 TxD 引脚的控制 , 直到数据、排队空闲或断点字符的发送完成后 , 才允许该引脚恢复为通用 I/O 引脚。</p> <p>0 发送器关闭。 1 发送器开启。</p>
2 RE	<p>接收器使能</p> <p>UART 接收器关闭时 , RxD 引脚恢复为通用端口 I/O 引脚。如果 C1[LOOPS]置位 , 即使 RE 置位 , RxD 引脚也会恢复为通用 I/O 引脚。</p> <p>0 接收器关闭。 1 接收器开启。</p>
1 RWU	<p>接收器唤醒控制</p> <p>可将 1 写入该字段以将 UART 接收器置于待机状态 , 令其等待硬件自动检测选定的唤醒条件。唤醒条件是报文之间有空闲线路、WAKE = 0、空闲线路唤醒 , 或者字符的最高有效位为逻辑 1、WAKE = 1、地址标记唤醒。应用软件置位 RWU , 选定的硬件条件一般会自动将 RWU 清零。</p> <p>0 UART 接收器正常工作。 1 UART 接收器处于待机状态 , 等待唤醒条件。</p>
0 SBK	<p>发送断点</p> <p>先将 1 写入 SBK 然后再将 0 写入 , 可让一个断点字符在发送数据流中排队。只要 SBK 置位 , 就可将其他断点字符加入队列 , 比如逻辑 0 的 10、11、12 或 13、14、15 ( 若 BRK13 = 1 ) 位时间。根据 SBK 置位和清零相对于当前发送信息的时序 , 可在软件将 SBK 清零前 , 将第二个断点字符加入队列。</p> <p>0 发送器正常工作。 1 将断点字符加入发送队列。</p>

## 33.3.5 UART 状态寄存器 1 (UARTx\_S1)

该寄存器有 8 个只读状态标志。写操作无效。通过特殊软件序列 ( 无需对该寄存器进行写操作 ) 来清除这些状态标志。

地址: 基址 基准 + 4h 偏移

位	7	6	5	4	3	2	1	0
读	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
写								
复位	1	1	0	0	0	0	0	0

**UARTx\_S1 字段描述**

字段	描述
7 TDRE	<p>发送数据寄存器空标志</p> <p>离开复位状态时，以及当发送数据值从发送数据缓冲区转移到发送移位器，使缓冲区有空间加载新字符时，TDRE 置位。要将 TDRE 清零，应在 TDRE 置位的情况下对 UART_S1 进行读操作，然后写入 UART 数据寄存器(UART_D)。</p> <p>0 发送数据寄存器(缓冲区)满。 1 发送数据寄存器(缓冲区)空。</p>
6 TC	<p>发送完成标志</p> <p>离开复位状态时，以及当 TDRE 置位且无数据、前同步码或断点字符在发送时，TC 置位。</p> <p>在 TC 置位的情况下对 UART_S1 进行读操作，然后执行以下操作之一，会自动将 TC 清零：</p> <ul style="list-style-type: none"> <li>• 写入 UART 数据寄存器(UART_D)以发送新数据</li> <li>• 将一个前同步码加入队列，方法是使 TE 从 0 变为 1</li> <li>• 将一个断点字符加入队列，方法是将 1 写入 UART_C2[SBK]</li> </ul> <p>0 发送器有效(发送数据、前同步码或断点)。 1 发送器空闲(发送活动完成)。</p>
5 RDRF	<p>接收数据寄存器满标志</p> <p>当一个字符从接收移位器转移到接收数据寄存器(UART_D)时，RDRF 置位。要将 RDRF 清零，应在 RDRF 置位的情况下对 UART_S1 进行读操作，然后对 UART 数据寄存器(UART_D)进行读操作。</p> <p>0 接收数据寄存器空。 1 接收数据寄存器满。</p>
4 IDLE	<p>空闲线路标志</p> <p>一段活动时间后，当 UART 接收线路变为空闲且持续整个字符时间时，IDLE 置位。当 C1[ILT]被清零时，接收器在起始位之后开始计数空闲位时间。如果接收字符为全 1，那么这些位时间和停止位时间将计数到接收器检测空闲线路所需的全字符逻辑高电平时间(10 或 11 位时间，具体取决于 M 控制位)。当 ILT 置位时，接收器在停止位之后开始计数空闲位时间。前一字符结束时的停止位和任何逻辑高电平时间不计数到接收器检测空闲线路所需的全字符逻辑高电平时间。</p> <p>要将 IDLE 清零，应在 IDLE 置位的情况下对 UART_S1 进行读操作，然后对 UART 数据寄存器(UART_D)进行读操作。将 IDLE 清零后，要再次将其置位，必须等到接收到新字符且 RDRF 置位之后。即使接收线路长时间空闲，IDLE 也只会置位一次。</p> <p>0 未检测到空闲线路。 1 检测到空闲线路。</p>
3 OR	<p>接收器溢出标志</p> <p>如果已经准备好将一个新的串行字符转移到接收数据寄存器(缓冲区)，但尚未从 UART_D 读取之前接收的字符，则 OR 置位。这种情况下，新字符和所有相关的错误信息都会丢失，因为没有空间可将其转移到 UART_D。要将 OR 清零，应在 OR 置位的情况下对 UART_S1 进行读操作，然后对 UART 数据寄存器(UART_D)进行读操作。</p> <p>0 未溢出。 1 接收溢出(新 UART 数据丢失)。</p>
2 NF	<p>噪声标志</p> <p>接收器采用的先进采样技术会在起始位期间采集 7 个样本，在各数据位和停止位期间采集 3 个样本。在帧中的任何位时间内，如果任一样本与其余样本不一致，则对于该字符，在 RDRF 置位的同时，标志 NF 也置位。要将 NF 清零，应先对 UART_S1 进行读操作，再对 UART 数据寄存器(UART_D)进行读操作。</p>

下一页继续介绍此表...

**UARTx\_S1 字段描述 (继续)**

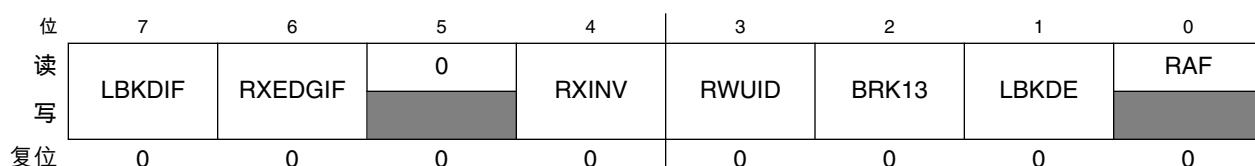
字段	描述
	0 未检测到噪声。 1 在 UART_D 的接收字符中检测到噪声。
1 FE	帧传输错误标志  当接收器在预期出现停止位的地方检测到逻辑 0 时，FE 与 RDRF 同时置位。这表示接收器与字符帧未正确对齐。要将 FE 清零，应在 FE 置位的情况下对 UART_S1 进行读操作，然后对 UART 数据寄存器(UART_D)进行读操作。  0 未检测到帧传输错误。这并不保证帧传输正确。 1 帧传输错误
0 PF	奇偶校验错误标志  当奇偶校验使能且接收字符中的奇偶位与预期奇偶值不一致时，PF 与 RDRF 同时置位。要将 PF 清零，应先对 UART_S1 进行读操作，再对 UART 数据寄存器(UART_D)进行读操作。  0 无奇偶校验错误。 1 奇偶校验错误。

**33.3.6 UART 状态寄存器 2 (UARTx\_S2)**

该寄存器包含一个只读状态标志。

当 LIN 系统使用内部振荡器时，需要将断点检测阈值提高一个位时间。在 LIN 允许的最差时序情况下，如果从机的运行速度比主机快 14%，0x00 数据字符可能表现为 10.26 位时间长。这会触发设计用于检测 10 位断点符号的正常断点检测电路。当 LBKDE 位置位时，帧传输错误被抑制，断点检测阈值从 10 位变为 11 位，防止误将 0x00 数据字符检测为 LIN 断点符号。

地址: 基址 基准 + 5h 偏移

**UARTx\_S2 字段描述**

字段	描述
7 LBKDIF	LIN 断点检测中断标志  当 LIN 断点检测电路使能且检测到 LIN 断点字符时，LBKDIF 置位。将 LBKDIF 清零的方法是将 1 写入该位。  0 未检测到 LIN 断点字符。 1 已检测到 LIN 断点字符。
6 RXEDGIF	RxD 引脚有效边沿中断标志

下一页继续介绍此表...

## UARTx\_S2 字段描述 (继续)

字段	描述
	当 RxD 引脚上出现一个有效边沿 ( RXINV = 0 时为下降沿 , RXINV = 1 时为上升沿 ) 时 , RXEDGIF 置位。将 RXEDGIF 清零的方法是将 1 写入该位。 0 接收引脚上未出现有效边沿。 1 接收引脚上已出现有效边沿。
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
4 RXINV	接收数据反转 置位该字段会反转接收数据输入的极性。 注：置位 RXINV 会反转所有字符的 RxD 输入：数据位、起始和停止位、断点及空闲。 0 接收数据不反转。 1 接收数据反转。
3 RWUID	接收唤醒空闲检测 RWUID 控制唤醒接收器的空闲字符是否置位 S1[IDLE]位。 0 在接收待机状态(RWU = 1)期间，检测到空闲字符时，S1[IDLE]不置位。 1 在接收待机状态(RWU = 1)期间，检测到空闲字符时，S1[IDLE]置位。
2 BRK13	断点字符生成长度 BRK13 选择较长的发送断点字符长度。帧传输错误检测不受该字段状态的影响。 0 断点字符以 10 位时间 ( M = 0、SBNS = 0 )、11 位时间 ( M = 1、SBNS = 0 后 M = 0、SBNS = 1 ) 或 12 位时间 ( M = 1、SBNS = 1 ) 的长度发送。 1 断点字符以 13 位时间 ( M = 0、SBNS = 0 )、14 位时间 ( M = 1、SBNS = 0 后 M = 0、SBNS = 1 ) 或 15 位时间 ( M = 1、SBNS = 1 ) 的长度发送。
1 LBKDE	LIN 断点检测使能 LBKDE 使能断点检测。LBKDE 置位时，禁止 S1[FE] 和 S1[RDRF] 标志置位。 0 断点检测禁用。 1 断点检测使能，断点字符以 11 位时间( C1[M] = 0、BDH[SBNS] = 0 )、12 位时间( C1[M] = 1、BDH[SBNS] = 0 或 C1[M] = 0、BDH[SBNS] = 1 ) 或 13 位时间 ( C1[M] = 1、BDH[SBNS] = 1 ) 长度进行检测。
0 RAF	接收器有效标志 当 UART 接收器检测到一个有效起始位的开始时，RAF 置位；当接收器检测到空闲线路时，RAF 被自动清零。该状态标志在 MCU 受令进入 Stop 模式之前，可用于检查是否正在接收 UART 字符。 0 UART 接收器空闲，等待起始位。 1 UART 接收器有效 ( RxD 输入非空闲 )。

### 33.3.7 UART 控制寄存器 3 (UARTx\_C3)

地址: 基址 基准 + 6h 偏移

位	7	6	5	4	3	2	1	0
读	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
写								
复位	0	0	0	0	0	0	0	0

#### UARTx\_C3 字段描述

字段	描述
7 R8	接收器的第 9 数据位  当 UART 针对 9 位数据(C1[M] = 1)进行配置时，可将 R8 视作 UART_D 寄存器中缓存数据 MSB 左边的第 9 接收数据位。读取 9 位数据时，先对 R8 进行读操作，再对 UART_D 进行读操作，因为对 UART_D 进行读操作会完成标志自动清除序列，R8 和 UART_D 因此可能会被新数据覆盖。
6 T8	发送器的第 9 数据位  当 UART 针对 9 位数据(C1[M] = 1)进行配置时，可将 T8 视作 UART_D 寄存器中数据 MSB 左边的第 9 发送数据位。写入 9 位数据时，在写入 UART_D 之后，9 位值整体转移到 UART 移位寄存器。因此，应先写入 T8( 如果它与先前的值不同 )，再写入 UART_D。如果新值中的 T8 保持不变，例如用于传号或空号校验，则无需在每次写入 UART_D 时写入 T8。
5 TXDIR	单线模式下的 TxD 引脚方向  当 UART 配置为单线半双工工作模式 ( LOOPS = RSRC = 1 ) 时，该字段决定 TxD 引脚的数据方向。  0 TxD 引脚在单线模式下为输入。 1 TxD 引脚在单线模式下为输出。
4 TXINV	发送数据反转  置位该字段会反转发送数据输出的极性。  注：置位 TXINV 会反转所有字符的 TxD 输出：数据位、起始和停止位、断点及空闲。  0 发送数据不反转。 1 发送数据反转。
3 ORIE	溢出中断使能  使能溢出标志(OR)以生成硬件中断请求。  0 OR 中断禁用；使用轮询。 1 OR 置位时，请求硬件中断。
2 NEIE	噪声错误中断使能  使能噪声标志(NF)以生成硬件中断请求。  0 NF 中断禁用；使用轮询。 1 NF 置位时，请求硬件中断。
1 FEIE	帧传输错误中断使能  使能帧传输错误标志(FE)以生成硬件中断请求。

下一页继续介绍此表...

**UARTx\_C3 字段描述 (继续)**

字段	描述
	0 FE 中断禁用；使用轮询。 1 FE 置位时，请求硬件中断。
0 PEIE	奇偶校验错误中断使能  使能奇偶校验错误标志(PF)以生成硬件中断请求。  0 PF 中断禁用；使用轮询。 1 PF 置位时，请求硬件中断。

**33.3.8 UART 数据寄存器 (UARTx\_D)**

此寄存器实际上是两个独立的寄存器。读操作返回只读接收数据缓冲区的内容，写操作将内容写入只写发送数据缓冲区。UART 状态标志的标志自动清除机制还涉及到对该寄存器的读写操作。

地址: 基址 基准 + 7h 偏移

位	7	6	5	4	3	2	1	0
读写	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	ROT0
复位	0	0	0	0	0	0	0	0

**UARTx\_D 字段描述**

字段	描述
7 R7T7	读取接收数据缓冲区 7 或写入发送数据缓冲区 7。
6 R6T6	读取接收数据缓冲区 6 或写入发送数据缓冲区 6。
5 R5T5	读取接收数据缓冲区 5 或写入发送数据缓冲区 5。
4 R4T4	读取接收数据缓冲区 4 或写入发送数据缓冲区 4。
3 R3T3	读取接收数据缓冲区 3 或写入发送数据缓冲区 3。
2 R2T2	读取接收数据缓冲区 2 或写入发送数据缓冲区 2。
1 R1T1	读取接收数据缓冲区 1 或写入发送数据缓冲区 1。
0 ROT0	读取接收数据缓冲区 0 或写入发送数据缓冲区 0。

## 33.4 功能说明

UART 允许在 MCU 和远程设备（包括其他 MCU）之间进行全双工、异步、NRZ串行通信。

UART 包含波特率发生器、发送器和接收器模块。尽管发送器和接收器使用相同的波特率发生器，但它们是独立运行的。在正常操作中，MCU 监控 UART 的状态，写入要发送的数据，并处理接收到的数据。以下介绍 UART 的各个模块。

### 33.4.1 波特率生成

如下图所示，UART 波特率发生器的时钟源是总线速率时钟。

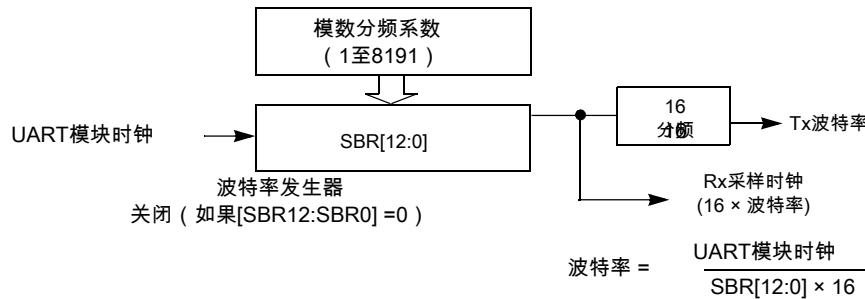


图 33-3. UART 波特率生成

UART 通信通常要求发送器和接收器从独立的时钟源获得相同的波特率。此波特频率上允许的公差取决于接收器与起始位的前沿的同步方式以及执行位采样的方式。

MCU 在每次高电平跃迁到低电平时与位边界重新同步。在最糟糕的情况下，完整的 10、11 或 12 位时间字符帧中没有此类传输，因此整个字符时间内将累积起任何波特率的不匹配情况。对于总线频率由晶振驱动的 FreescaleUART 系统，允许的波特率不匹配在 8 位数据格式中约为 $\pm 4.5\%$ ，9 位数据格式中约为 $\pm 4\%$ 。尽管波特率模数分频器设置并不始终产生与标准波特率完全匹配的波特率，但通常可以将误差限制在几个百分点内，可视为可靠的通信。

### 33.4.2 发送器功能说明

本节说明 UART 发送器的总体结构框图，以及发送断点和空闲字符的特殊功能。

发送器输出(TxD)空闲状态默认处于逻辑高电平，UART\_C3[TXINV]在复位后被清零。发送器输出通过UART\_C3[TXINV]置位反相。发送器通过UART\_C2中的TE位置位使能。这将把作为闲置状态的一个全角字符帧的报头字符排入队列。然后，发送器继续空闲，直到发送数据缓冲区中有数据可用。程序通过对UART数据寄存器(UART\_D)进行写操作而将数据存储到发送数据缓冲区。

UART发送器的核心部分是发送移位寄存器，它有10、11或12位，具体取决于UART\_C1[M]控制位和UART\_BDH[SBNS]位的设置。本节其余部分假设

UART\_C1[M]清零，UART\_BDH[SBNS]也清零，选择常规8位数据模式。在8位数据模式下，移位寄存器占用一个起始位、八个数据位和一个停止位。当发送移位寄存器可载入新UART字符时，在发送数据寄存器中等待的值会被传输到移位寄存器，与波特率时钟同步，并且发送数据寄存器空(UART\_S1[TDRE])状态标志置位，指示可将另一个字符写入UART\_D处的发送数据缓冲区。

### 注

务必先对UART\_S1进行读操作，再对UART\_D进行写操作，以便能够发送数据。

TxD引脚传出一个停止位后，如果发送数据缓冲区中没有新字符在等待，那么发送器会置位发送完成标志并进入空闲模式，且TxD为高电平，等待发送更多字符。

向UART\_C2[TE]写入0不会立即将此引脚释放为通用I/O引脚。必须首先完成所有正在进行的发送任务。这包括正在发送的数据字符、排队的空闲字符和断点字符。

#### 33.4.2.1 发送分隔和已排队的闲置

UART\_C2[SBK]发送断点字符，其原始用途是引起旧式电传打字接收机的注意。分隔字符是逻辑0的全角字符时间，包括起始位和停止位在内共10位时间。可通过设置UART\_S2[BRK13]启用更长的13位时间的分隔字符。通常情况下，程序会等待UART\_S1[TDRE]置位以指示报文的最后一个字符已被移入发送移位器，然后向UART\_C2[SBK]写入1，再写入0。一旦移位器可用，此操作就将要发送的分隔字符排入队列。当排入队列的分隔字符移入移位器，与波特率时钟进行同步时，如果UART\_C2[SBK]保持为1，则将把另一个分割字符排入队列。如果接收设备是另一个Freescale Semiconductor UART，在所有8数据位中分隔字符都将作为0被接受，并且生成帧错误(UART\_S1[FE]=1)。

使用空闲线路唤醒时，报文之间需要具有完整字符时间的空闲（逻辑1）以唤醒任何处于睡眠状态的接收器。通常，程序将等待UART\_S1[TDRE]变为设置状态，以指示报文的最后一个字符已移至发送移位器，向UART\_C2[TE]位写入0，然后写入1。一旦移位器可用，此操作就将要发送的闲置字符排入队列。清除UART\_C2[TE]时，只要移位器中的字符还未完成，发送器绝不会真正释放对UART引脚的控制。如果存在这样一种可能性：清除UART\_C2[TE]的同时移位器正在完成发送，那么

设置通用 I/O 控制，这样与 TxD 共用的引脚就是一个驱动逻辑 1 的输出。如此一来，即使在向 UART\_C2[TE]先写入 0 再写入 1 期间 UART 失去对端口引脚的控制，TxD 线路看起来也是正常的空闲线路。

断点字符的长度受 UART\_S2[BRK13]和 UART\_C1[M]影响，如下表所示。

表 33-3. 分隔字符长度

BRK13	M	SBNS	断点字符长度
0	0	0	10 位时间
0	0	1	11 位时间
0	1	0	11 位时间
0	1	1	12 位时间
1	0	0	13 位时间
1	0	1	14 位时间
1	1	0	14 位时间
1	1	1	15 位时间

### 33.4.3 接收器功能说明

在本节中，以接收器功能框图为指南来说明接收器的整体功能。

然后将更详细地说明用来重新构建接收器数据的数据采样技术。最后将介绍接收器唤醒功能的两种方式。

可通过置位 UART\_S2[RXINV]反转接收器输入。可通过置位 UART\_C2[RE]位启用接收器。字符帧由一个逻辑 0 起始位、八个（或九个）数据位（LSB 优先）以及一个（或二个）逻辑 1 停止位组成。有关 9 位数据模式的信息，请参见 [8 位和 9 位数据模式](#)。有关此讨论的剩余部分，假设为正常的 8 位数据模式配置了 UART。

将停止位接收到接收移位器中后，如果接收数据寄存器未满，数据字符将传输到此接收数据寄存器，并设置接收数据寄存器已满 (UART\_S1[RDRF]) 状态标志。如果 UART\_S1[RDRF]置位，说明此接收数据寄存器（缓冲器）已满，溢出(OR)状态标志会置位且新数据丢失。由于 UART 接收器采用双缓冲区，因此程序在 UART\_S1[RDRF]置位后，以及在读取接收数据缓冲区中的数据之前，有一个完整字符的时间，以免接收器溢出。

当某个程序检测到接收数据寄存器已满 (UART\_S1[RDRF] = 1) 时，它通过读取 UART\_D 从此接收数据寄存器中获取数据。通过管理接收数据的用户程序的流程中通常满足的两步序列自动清除 UART\_S1[RDRF]标志。有关标志清除的更多详情，请参见 [中断和状态标志](#)。

### 33.4.3.1 数据采样技术

UART 接收器使用 16 倍波特率时钟进行采样。过采样比是固定值 16。接收器启动后，以 16 倍的波特率采集逻辑电平样本，寻找 RxD 串行数据输入引脚上的下降沿。下降沿被定义为三个连续的逻辑 1 样本后的逻辑 0 样本。16 倍波特率时钟将位时间分割为 16 段，标记为 UART\_D[RT1]至 UART\_D[RT16]。定位到一个下降沿后，将在 UART\_D[RT3]、UART\_D[RT5]和 UART\_D[RT7]上再采三个样本，以确保这是真实的起始位，而非仅仅是噪声。如果这三个样本至少有两个为 0，接收器将视为已经与接收字符同步。

然后此接收器将在 UART\_D[RT8]、UART\_D[RT9]和 UART\_D[RT10]对各个位时间（包括起始位和停止位）进行采样，以确定该位的逻辑电平。此逻辑电平被认为是相关的位时间内所采的大部分样本的逻辑电平。如果是开始位，如果 UART\_D[RT3]、UART\_D[RT5]和 UART\_D[RT7]中至少两个样本为 0，则此位将被假定为 0，即便 UART\_D[RT8]、UART\_D[RT9]和 UART\_D[RT10]中的一个或所有样本为 1。如果在某个字符帧中，任何位时间（包括起始位和停止位）中的任何样本未与此位的逻辑电平保持一致，则当接收的字符传输到接收数据缓冲区时，噪声标志 (UART\_S1[NF]) 置位。

下降沿检测逻辑继续寻找下降沿。检测到一个下降沿时，采样时钟便与位时间重新同步。在有噪声或波特率不匹配的情况下，这可以提高接收器的可靠性。它不能改善最差情况分析，因为某些字符在字符帧中的任何地方都没有多余的下降沿。

发生帧传输错误时，如果已接收到的字符不是断点字符，那么寻找下降沿的采样逻辑将用三个逻辑 1 样本填充，这样就几乎能立即检测到新起始位。

发生成帧错误时，将禁止接收器接收任何新字符，直至清除成帧错误标志。接收移位寄存器继续正常工作，但如果 UART\_S1[FE]仍保持置位状态，则完整的字符无法传输到接收数据缓冲区。

### 33.4.3.2 接收器唤醒操作

接收器唤醒是一种硬件机制，使 UART 接收器能忽略用于其他 UART 接收器的报文中的字符。在此类系统中，各接收器都会评估每条报文的首字符，一旦确定其用于其他接收器，就会将逻辑 1 写入接收器唤醒控制字段(UART\_C2[RWU])。当 UART\_C2[RWU] 置位时，禁止与接收器相关的状态标志 (UART\_S2[RWUID] 置位时空闲位 IDLE 除外) 置位，从而消除软件处理不重要报文字符的开销。报文结束时，或下一报文开始时，所有接收器都自动将 UART\_C2[RWU] 强制为 0，因此所有接收器都会及时醒来，以便检查下一报文的首字符。

### 33.4.3.2.1 空闲线路唤醒

清除唤醒后，将为接收器配置闲置线路唤醒。在此模式中，当接收器检测到闲置线路级别的全角字符时间时，将自动清除 UART\_C2[RWU]。UART\_C1[M]控制字段选择 8 位或 9 位数据模式，UART\_BDH[SBNS]选择 1 个或 2 个停止位，从而确定需要多少位时间的空闲才能构成完整字符时间：10 位、11 位或 12 位时间（包括起始位和停止位）。

当 UARTI\_C2[RWU] 为 1 且 UART\_S2[RWUID] 为 0 时，唤醒接收器的空闲条件不会置位 UART\_S1[IDLE]。接收器唤醒后，等待下一报文的第一个数据字符，由后者置位 UART\_S1[RDRF] 并生成中断（若使能）。当 UART\_S2[RWUID] 为 1 时，任何空闲条件都会置位 UART\_S1[IDLE] 标志并生成中断（若使能），无论 UART\_C2[RWU] 是 0 还是 1。

闲置线路类型(UART\_C1[ILT])控制位可选择两种方式检测闲置线路。

UART\_C1[ILT] 清零后，闲置位计数器从起始位之后开始计数，以便字符末尾的停止位和任何逻辑 1 都计入闲置的全角字符时间。UART\_C1[ILT] 置位后，闲置位计数器在停止位时间之后才开始计数，因此闲置检测不受上一条报文的最后一个字符中的数据影响。

### 33.4.3.2.2 地址标记唤醒

设置唤醒后，会针对地址标记唤醒配置接收器。在该模式中，如果 UART\_BDH[SBNS] = 1，则当接收器检测到已接收字符的最高有效位（UART\_C1[M] 清零时为第八位，UART 置位时为第九位）中有一到两个逻辑 1 时，将自动清除 UART\_C2[RWU]。

地址标记唤醒允许报文包含闲置字符，但需要保留 MSB 以用于地址帧中。如果 UART\_BDH[SBNS] = 1，则地址帧 MSB 中的这一到两个逻辑 1 会在接收到停止位之前清除 UART\_C2[RWU] 位并使 UART\_S1[RDRF] 标志置位。在这种情况下，即使在大部分字符时间里接收器都处于休眠状态，MSB 置位的字符被接收。

## 33.4.4 中断和状态标志

UART 系统具有一个中断向量，它有三种不同的中断类型。一个中断向量类型在 UART\_S1[TDRE] 和 UART\_S1[TC] 事件上与发送器相关联。另一个中断类型与接收器相关，用于 RDRF、IDLE、RXEDGIF 和 LBKDIF 事件。第三个类型用于 OR、NF、FE 和 PF 错误条件。这十个中断源各自都可以通过本地中断使能掩码加以屏蔽。当清除本地掩码以禁用硬件中断请求的生成时，可由软件轮询标志。

UART 发送器有两个状态标志，可选择性地生成硬件中断请求。发送数据寄存器空 (UART\_S1[TDRE]) 说明发送数据缓冲区中有空间将另一个发送字符写入 UART\_D。如果发送中断启用 (UART\_C2[TIE]) 位置位，则 UART\_S1[TDRE] 置位时

请求硬件中断。发送完成(UART\_S1[TC])说明发送器已完成所有数据、前同步码和断点字符的发送，现处于空闲状态，且 TxD 处于无效电平。此标志通常用于带调制解调器的系统，用于确定安全关闭此调制解调器的时间。如果发送完成中断启用(UART\_C2[TCIE])位置位，则 UART\_S1[TC]置位时请求硬件中断。如果相应的UART\_C2[TIE]或 UART\_C2[TCIE]本地中断掩码清零，则可使用软件轮询而非硬件中断以监控 UART\_S1[TDRE]和 UART\_S1[TC]状态标志。

当某个程序检测到接收数据寄存器已满(UART\_S1[RDRF] = 1)时，它通过读取 UART\_D 从此接收数据寄存器中获取数据。当 UART 置位时，可以通过读取 UART\_S1 然后读取 UART\_D 来清除 UART\_S1[RDRF]标志。

使用轮询时，用户程序的正常过程使用的就是此序列。如果使用硬件中断，则必须在中断服务程序(ISR)中读取 UART\_S1。通常都是在 ISR 中完成此步骤，以检查接收错误，因此可自动满足此序列。

IDLE 状态标志包括一种逻辑，可避免在 RxD 线路保持闲置状态更长的时间时重复设置此标志。当 UART\_S1[IDLE]置位时，可以通过读取 UART\_S1 然后读取 UART\_D 来清除 IDLE。UART\_S1[IDLE]清零后，在接收器接收到至少一个新字符并已使 UART\_S1[RDRF]置位前，都不可重新置位。

如果接收到的字符中检测到导致 UART\_S1[RDRF]置位的相关错误，则在 UART\_S1[RDRF]置位的同时使错误标志(即噪声标志(UART\_S1[NF])、成帧错误(UART\_S1[FE])和奇偶错误标志(UART\_S1[PF]))置位。溢出情况下这些标志不置位。

当准备将新字符从接收移位器传输到接收数据缓冲区时，如果 UART\_S1[RDRF]置位，则溢出的(UART\_S1[OR])标志置位而非数据，同时丢失任何相关 NF、FE 或 PF 情况。

RxD 串行数据输入引脚上的活动边沿随时都可导致 UART\_S2[RXEDGIF]标志置位。可通过向其中写入 1 清除 UART\_S2[RXEDGIF]标志。此功能需要启用接收器(UART\_C2[RE] = 1)。

### 33.4.5 波特率公差

发送器件工作时的波特率可能低于或高于接收器工作时的波特率。

累积的位时间偏差可导致三个停止位数据样本(RT8、RT9 和 RT10)之一超出实际停止位之外。如果 RT8、RT9 和 RT10 样本的逻辑值不全相同，就会产生噪声误差。如果接收器时钟发生偏差，即大部分 RT8、RT9 和 RT10 停止位样本值为逻辑零，则将产生成帧错误。

当接收器对传入的帧进行采样时，它在此帧的任何有效上升沿上重新与 RT 时钟进行同步。在帧中进行重新同步将纠正发送位时间与接收位时间之间的偏差。

### 33.4.5.1 慢速数据公差

图 33-4 显示了慢速接收帧最多可以有多大的对齐误差而不引起噪声错误或帧传输错误。慢速停止位开始于 RT8，而非 RT1，但会及时到达，以便在 RT8、RT9 和 RT10 进行停止位数据采样。

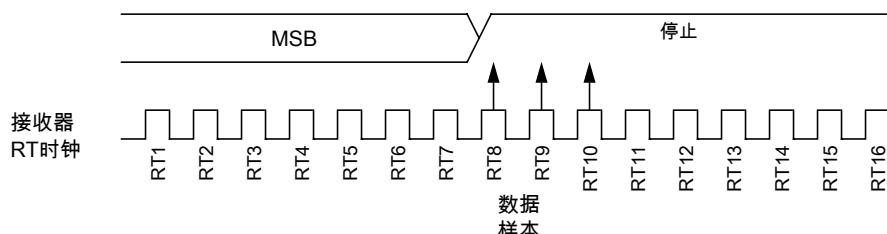


图 33-4. 慢速数据

对于 8 位数据和 1 停止位的字符，停止位的数据采样需要接收器 9 位时间  $\times 16$  RT 周期 + 10 RT 周期 = 154 RT 周期。

对于图 33-4 所示的未对齐字符，当发送器计数到 9 位时间  $\times 16$  RT 周期 + 3 RT 周期 = 147 RT 周期时，接收器计数到 154 RT 周期。

在无误差的情况下，慢速 8 位数据和 1 停止位的字符的接收器计数与发送器计数之间的最大百分比差异为：

$$((154 - 147) / 154) \times 100 = 4.54\%$$

对于 9 位数据或 2 停止位的字符，停止位的数据采样需要接收器 10 位时间  $\times 16$  RT 周期 + 10 RT 周期 = 170 RT 周期。

对于图 33-4 所示的未对齐字符，当发送器计数到 10 位时间  $\times 16$  RT 周期 + 3 RT 周期 = 163 RT 周期时，接收器计数到 170 RT 周期。

在无误差的情况下，慢速 9 位或 2 停止位的字符的接收器计数与发送器计数之间的最大百分比差异为：

$$((170 - 163) / 170) \times 100 = 4.12\%$$

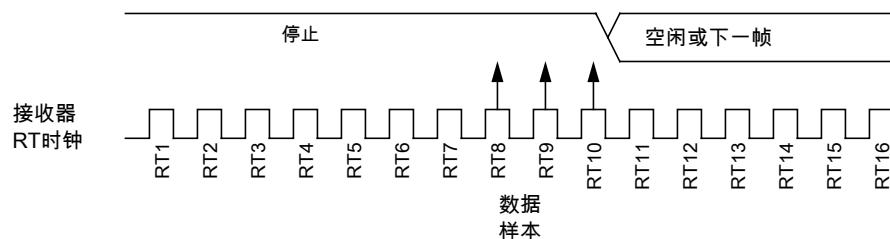
对于 9 位数据和 2 停止位的字符，停止位的数据采样需要接收器 11 位时间  $\times 16$  RT 周期 + 10 RT 周期 = 186 RT 周期。

对于图 33-4 所示的未对齐字符，当发送器计数到 11 位时间  $\times 16$  RT 周期 + 3 RT 周期 = 179 RT 周期时，接收器计数到 186 RT 周期。

在无误差的情况下，慢速 9 位和 2 停止位的字符的接收器计数与发送器计数之间的最大百分比差异为：  $((186 - 179) / 186) \times 100 = 3.76\%$

### 33.4.5.2 快速数据公差

图 33-5 显示了快速接收帧允许的最大对齐误差。快速停止位结束于 RT10，而非 RT16，不过仍然在 RT8、RT9 和 RT10 进行采样。



**图 33-5. 快速数据**

对于 8 位数据和 1 停止位的字符，停止位的数据采样需要接收器 9 位时间  $x 16$  RT 周期 + 10 RT 周期 = 154 RT 周期。

对于图 33-5 所示的未对齐字符，当发送器件计数到 10 位时间  $x 16$  RT 周期 = 160 RT 周期时，接收器计数到 154 RT 周期。

一个快速 8 位和 1 停止位的无错误字符的接收器计数与发送器计数之间的最大百分比差异为：

$$((154 - 160) / 154) \times 100 = 3.90\%$$

对于 9 位数据或 2 停止位的字符，停止位的数据采样需要接收器 10 位时间  $x 16$  RT 周期 + 10 RT 周期 = 170 RT 周期。

对于所示的未对齐字符，当发送器件计数到 11 位时间  $x 16$  RT 周期 = 176 RT 周期时，接收器计数到 170 RT 周期。

一个快速 9 位或 2 停止位的无错误字符的接收器计数与发送器计数之间的最大百分比差异为：

$$((170 - 176) / 170) \times 100 = 3.53\%$$

对于 9 位数据和 2 停止位的字符，停止位的数据采样需要接收器 11 位时间  $x 16$  RT 周期 + 10 RT 周期 = 186 RT 周期。

对于所示的未对齐字符，当发送器件计数到 12 位时间  $x 16$  RT 周期 = 193 RT 周期时，接收器计数到 186 RT 周期。

一个快速 9 位和 2 停止位的无错误字符的接收器计数与发送器计数之间的最大百分比差异为：

$$((186 - 193) / 186) \times 100 = 3.23\%$$

### 33.4.6 其他 UART 功能

以下各节介绍其他 UART 功能。

#### 33.4.6.1 8 位和 9 位数据模式

通过使 `UART_C1[M]` 置位，可将 UART 系统、发送器和接收器配置为以 9 位数据模式工作。在 9 位模式下，UART 数据寄存器的最高有效位左边有一个第九数据位。对于发送数据缓冲区，此位存储在 `UART_C3` 中的 `T8` 中；对于接收器，此第九位保留在 `UART_C3[R8]` 中。

为了一致地写入发送数据缓冲区，应先写入 `UART_C3[T8]`，再写入 `UART_D`。

如果要作为新字符的第九位传输的位值与上一个字符相同，则无需再次写入 `UART_C3[T8]`。当数据从发送数据缓冲区传输到发送移位器时，将在数据从 `UART_D` 传输到移位器的同时复制 `UART_C3[T8]` 中的值。

9 位数据模式通常配合奇偶校验使用，以支持 8 位数据并将第九位用于奇偶校验，或者配合地址标记唤醒使用，第九数据位用作唤醒位。在自定义协议中，第九位也可用作软件控制标记。

#### 33.4.6.2 Stop 模式操作

在所有 Stop 模式中，UART 模块的时钟都会暂停。

在 Stop 模式下，UART 模块寄存器不受影响。

接收输入活动边沿检测电路在 Stop 模式中保持活动状态，但在模式下不保持活动状态。如果中断未屏蔽(`UART_BDH[RXEDGIE]` = 1)，接收输入上的活动边沿将使 CPU 退出 Stop 模式。

由于时钟已停止，UART 模块在退出 Stop 模式（仅在 Stop 模式中）时恢复操作。当 UART 模块正在发送或接收字符（包括前同步码、断点和普通数据）时，软件必须确保它不进入 Stop 模式，也就是说，要进入 Stop 模式，必须满足以下全部条件：`UART_S1[TC]` = 1、`UART_S1[TDRE]` = 1 和 `UART_S2[RAF]`。

### 33.4.6.3 循环模式

UART\_C1[LOOPS]置位后，相同寄存器中的UART\_C1[RSRC]位将在循环模式(UART\_C1[RSRC] = 0)或单线模式(UART\_C1[RSRC] = 1)中进行选择。循环模式有时用于检查软件（与外部系统中的连接无关）以帮助隔离系统问题。在此模式下，从发送器到接收器的内部回环连接会导致接收器接收由发送器发出的字符。

### 33.4.6.4 单线操作

UART\_C1[LOOPS]置位后，UART\_C1[RSRC]将在循环模式(UART\_C1[RSRC] = 0)或单线模式(UART\_C1[RSRC] = 1)中进行选择。单线模式实施半双工串行连接。接收器内部连接到发送器输出和 TxD 引脚。RxD 引脚不使用，并恢复成通用端口 I/O 引脚。

在单线模式中，UART\_C3[TXDIR]位控制 TxD 引脚上串行数据的方向。UART\_C3[TXDIR]清零后，TxD 引脚成为 UART 接收器的输入，且发送器暂时与 TxD 引脚断开，使外部设备可向此接收器发送串行数据。UART\_C3[TXDIR]置位后，TxD 引脚是由此发送器驱动的输出。在单线模式中，发送器输出从内部连接到接收器输入，且 RxD 未由 UART 使用，因此它恢复为通用端口 I/O 引脚。

# 第 34 章 通用输入/输出 (GPIO)

## 34.1 简介

### 注

关于与具体芯片相关的本模块详细操作示例, 请参见芯片配置信息。

通用输入和输出(GPIO)模块可通过外设总线访问, 还能通过零等待状态接口(IOPORT)与处理器内核通信, 实现最高的引脚性能。GPIO 寄存器支持 8 位、16 位或 32 位访问。

当引脚配置为用于 GPIO 功能时, GPIO 数据方向和输出数据寄存器控制每个引脚的方向和输出数据。假设引脚相应的端口控制和中断模块已使能, 则当引脚配置为任意数字功能时, GPIO 输入数据寄存器显示每个引脚上的逻辑值。

通过为每个端口输出数据寄存器增加针对只写寄存器的置位、清零和跳变操作, 使通用输出模块支持高效的位操作。

### 34.1.1 特性

- GPIO 模块的特性包括:
  - 端口数据输入寄存器适用于所有数字引脚多路复用模式
  - 端口数据输出寄存器带对应的置位/清零/跳变寄存器
  - 端口数据方向寄存器
  - 通过 IOPORT 实现对 GPIO 寄存器的零等待状态访问

### 注

GPIO 模块的时钟源为系统时钟。

## 34.1.2 工作模式

下表介绍了 GPIO 模块的不同工作模式及其在这些模式下的行为。

表 34-1. 工作模式

工作模式	说明
运行	GPIO 模块正常运行。
等待	GPIO 模块正常运行。
停止	GPIO 模块被禁用。
调试	GPIO 模块正常运行。

## 34.1.3 GPIO 信号说明

表 34-2. GPIO 信号说明

GPIO 信号说明	说明	I/O
PTA7–PTA0	通用输入/输出	I/O
PTB7–PTB0	通用输入/输出	I/O
PTC7–PTC0	通用输入/输出	I/O
PTD7–PTD0	通用输入/输出	I/O
PTE7–PTE0	通用输入/输出	I/O
PTF7–PTF0	通用输入/输出	I/O
PTG7–PTG0	通用输入/输出	I/O
PTH7–PTH0	通用输入/输出	I/O
PTI7–PTI0 <sup>1</sup>	通用输入/输出	I/O

1. 该器件仅提供 PTI6–PTI0。

### 注

并非所有端口的所有引脚都绑定到所有封装上。有关设备上的可用 GPIO 端口数量信息，请参见信号多路复用一章。

### 34.1.3.1 详细信号说明

表 34-3. GPIO 接口详细信号说明

信号	I/O	说明	
PTA7–PTA0	I/O	通用输入/输出	
PTB7–PTB0		状态含义	电平有效：该引脚为逻辑 1。 电平无效：该引脚为逻辑 0。
PTC7–PTC0			
PTD7–PTD0			
PTE7–PTE0			

下一页继续介绍此表...

表 34-3. GPIO 接口详细信号说明 (继续)

信号	I/O	说明	
PTF7–PTF0 PTG7–PTG0 PTH7–PTH0 PTI7–PTI0 <sup>1</sup>		时序	有效电平: 输出时, 该信号发生在系统时钟的上升沿。就输入而言, 它可能在任何时刻发生, 且输入信号可能和系统时钟不同步。 无效电平: 输出时, 该信号发生在系统时钟的上升沿。就输入而言, 可能在任何时刻发生, 且输入信号可能和系统时钟不同步。

1. 该器件仅提供 PTI6–PTI0。

### 注

并非每个端口内的所有引脚都会在设备上实施。有关器件的可用 GPIO 端口数量, 请参见信号多路复用的相关章节。

## 34.2 存储器映像和寄存器定义

GPIO 模块在 AIPS-Lite 外设联接器上有两个地址插槽, 能够保持 Freescale 产品之间的软件兼容性。所有 GPIO 寄存器均可通过基地址 0x400F\_F000 或 0x4000\_F000 进行访问。建议将 0x400F\_F000 用作 GPIO 模块的基地址, 本章所述的寄存器存储器映像同样基于基地址 0x400F\_F000。

在有效存储器映像外对 GPIO 存储器空间进行任何读写访问都将导致总线错误。

### 34.2.1 GPIO/FGPIO 寄存器位分配

在该器件中, 所有 8 位端口引脚都映射到 32 位 GPIO/FGPIO 寄存器, 如表中所示。

表 34-4. GPIOA/FGPIOA 寄存器位分配

寄存器位	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
端口引脚	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0

表 34-5. GPIOB/FGPIOB 寄存器位分配

端口引脚	寄存器位	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9 8	8 7	7 6	6 5	4 3	3 2	2 1	1 0
PTH7	PTH6	PTH5	PTH4	PTH3	PTH2	PTH1	PTH0	PTG7	PTG6	PTG5	PTG4	PTG3	PTG2	PTG1	PTG0	PTF7	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0

表 34-6. GPIOC/FGPIOC 寄存器位分配

端口引脚	寄存器位	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9 8	8 7	7 6	6 5	4 3	3 2	2 1	1 0
保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	PTI6	PTI5	PTI4	PTI3	PTI2	PTI1	PTI0	

### GPIO 存储器映射

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
400F_F000	端口数据输出寄存器 (GPIOA_PDOR)	32	R/W	0000_0000h	<a href="#">34.2.2/633</a>
400F_F004	端口置位输出寄存器 (GPIOA_PSOR)	32	W (始终 读 0 )	0000_0000h	<a href="#">34.2.3/633</a>
400F_F008	端口清零输出寄存器 (GPIOA_PCOR)	32	W (始终 读 0 )	0000_0000h	<a href="#">34.2.4/634</a>
400F_F00C	端口跳变输出寄存器 (GPIOA_PTOR)	32	W (始终 读 0 )	0000_0000h	<a href="#">34.2.5/634</a>
400F_F010	端口数据输入寄存器 (GPIOA_PDIR)	32	R	0000_0000h	<a href="#">34.2.6/635</a>
400F_F014	端口数据方向寄存器 (GPIOA_PDDR)	32	R/W	0000_0000h	<a href="#">34.2.7/635</a>
400F_F018	端口输入禁用寄存器 (GPIOA_PIDR)	32	R/W	FFFF_FFFFh	<a href="#">34.2.8/636</a>
400F_F040	端口数据输出寄存器 (GPIOB_PDOR)	32	R/W	0000_0000h	<a href="#">34.2.2/633</a>
400F_F044	端口置位输出寄存器 (GPIOB_PSOR)	32	W (始终 读 0 )	0000_0000h	<a href="#">34.2.3/633</a>
400F_F048	端口清零输出寄存器 (GPIOB_PCOR)	32	W (始终 读 0 )	0000_0000h	<a href="#">34.2.4/634</a>
400F_F04C	端口跳变输出寄存器 (GPIOB_PTOR)	32	W (始终 读 0 )	0000_0000h	<a href="#">34.2.5/634</a>
400F_F050	端口数据输入寄存器 (GPIOB_PDIR)	32	R	0000_0000h	<a href="#">34.2.6/635</a>
400F_F054	端口数据方向寄存器 (GPIOB_PDDR)	32	R/W	0000_0000h	<a href="#">34.2.7/635</a>
400F_F058	端口输入禁用寄存器 (GPIOB_PIDR)	32	R/W	FFFF_FFFFh	<a href="#">34.2.8/636</a>
400F_F080	端口数据输出寄存器 (GPIOC_PDOR)	32	R/W	0000_0000h	<a href="#">34.2.2/633</a>
400F_F084	端口置位输出寄存器 (GPIOC_PSOR)	32	W (始终 读 0 )	0000_0000h	<a href="#">34.2.3/633</a>

下一页继续介绍此表...

## GPIO 存储器映射 (继续)

绝对地址（十六进制）	寄存器名称	宽度（单位：位）	访问	复位值	小节/页
400F_F088	端口清零输出寄存器 (GPIOC_PCOR)	32	W (始终读0)	0000_0000h	34.2.4/634
400F_F08C	端口跳变输出寄存器 (GPIOC_PTOR)	32	W (始终读0)	0000_0000h	34.2.5/634
400F_F090	端口数据输入寄存器 (GPIOC_PDIR)	32	R	0000_0000h	34.2.6/635
400F_F094	端口数据方向寄存器 (GPIOC_PDDR)	32	R/W	0000_0000h	34.2.7/635
400F_F098	端口输入禁用寄存器 (GPIOC_PIDR)	32	R/W	FFFF_FFFFh	34.2.8/636

### 34.2.2 端口数据输出寄存器 (GPIOx PDOR)

该寄存器配置在各通用输出引脚上驱动的逻辑电平。

注

请勿修改选定封装中未绑定引脚的配置寄存器。封装中未绑定的引脚将默认处于禁用状态，以实现最低功耗。

地址：基址 基准 + 0h 偏移

## GPIOx PDOR 字段描述

字段	描述
PDO	<p>端口数据输出</p> <p>读取未绑定引脚的寄存器位将返回无效值。</p> <p>0 假设引脚配置为通用输出，则在引脚上驱动逻辑电平 0。 1 假设引脚配置为通用输出，则在引脚上驱动逻辑电平 1。</p>

### 34.2.3 端口置位输出寄存器 (GPIOx PSOR)

该寄存器可以将 PDOR 字段置位。

地址·基址 基准 + 4h 偏移

**GPIOx\_PSOR 字段描述**

字段	描述
PTSO	<p>端口置位输出</p> <p>如下所示对该寄存器进行写操作将更新 PDOR 中对应位的内容：</p> <p>0 PDORn 中的对应位不变。</p> <p>1 PDORn 中的对应位置为逻辑 1。</p>

**34.2.4 端口清零输出寄存器 (GPIOx\_PCOR)**

该寄存器可以将 PDOR 字段清零。

地址: 基址 基准 + 8h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R																	0																	
W																		PTCO																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**GPIOx\_PCOR 字段描述**

字段	描述
PTCO	<p>端口清零输出</p> <p>如下所示对该寄存器进行写操作将更新端口数据输出寄存器(PDOR)中对应位的内容：</p> <p>0 PDORn 中的对应位不变。</p> <p>1 PDORn 中的对应位置为逻辑 0。</p>

**34.2.5 端口跳变输出寄存器 (GPIOx\_PTOR)**

地址: 基址 基准 + Ch 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R																	0																	
W																		PTTO																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**GPIOx\_PTOR 字段描述**

字段	描述
PTTO	<p>端口跳变输出</p> <p>如下所示对该寄存器进行写操作将更新 PDOR 中对应位的内容：</p> <p>0 PDORn 中的对应位不变。</p> <p>1 PDORn 中的对应位置为当前逻辑状态的反相电平。</p>

### 34.2.6 端口数据输入寄存器 (GPIOx\_PDIR)

#### 注

请勿修改选定封装中未绑定引脚的配置寄存器。封装中未绑定的引脚将默认处于禁用状态，以实现最低功耗。

地址: 基址 基准 + 10h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	PDI															
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

#### GPIOx\_PDIR 字段描述

字段	描述
PDI	<p>端口数据输入</p> <p>读取器件的未绑定引脚将返回 0。读取未配置为数字功能的引脚将返回 0。如果禁用端口控制和中断模块，那么 PDIR 中的对应位不更新。</p> <p>0 引脚逻辑电平为逻辑 0，或者未配置为数字功能。 1 引脚逻辑电平为逻辑 1。</p>

### 34.2.7 端口数据方向寄存器 (GPIOx\_PDDR)

PDDR 将各个端口引脚配置为输入或输出。

地址: 基址 基准 + 14h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	PDD															
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

#### GPIOx\_PDDR 字段描述

字段	描述
PDD	<p>端口数据方向</p> <p>将各个端口引脚配置为输入或输出。</p> <p>0 引脚配置为通用输入，用于 GPIO 功能。若在 GPIOx_PIDR 寄存器中禁用端口输入，则引脚将为高阻态。 1 引脚配置为通用输出，用于 GPIO 功能。</p>

### 34.2.8 端口输入禁用寄存器 (GPIOx\_P IDR)

地址: 基址 基准 + 18h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																PID																
W																																

复位 1

#### GPIOx\_P IDR 字段描述

字段	描述
PID	端口输入禁用 0 假设引脚配置为数字功能，则该引脚配置为通用输入。 1 引脚未配置为通用输入。读取对应的 PDIR 字段将返回 0。

### 34.3 FGPIO 存储器映像和寄存器定义

GPIO 寄存器也可另外命名，并指向 Cortex-M0+上地址 0xF800\_0000 处的 IOPORT 接口。

通过 IOPORT 接口执行的访问与任何指令的获取保持同步，因此可在一周期内完成。该另外命名的快速 GPIO 存储器映像称为 FGPIO。

对超出有效存储器映像范围的 FGPIO 存储器空间进行任何读写访问都会导致总线错误。所有对寄存器的访问均为零等待，但对错误地址的访问会有一个等待状态。

#### 34.3.1 GPIO/FGPIO 寄存器位分配

在该器件中，所有 8 位端口引脚都映射到 32 位 GPIO/FGPIO 寄存器，如表中所示。

表 34-7. GPIOA/FGPIOA 寄存器位分配

端口引脚	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
寄存器位	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0

表 34-8. GPIOB/FGPIOB 寄存器位分配

端口引脚	寄存器位	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9 8	8 7	7 6	6 5	5 4	4 3	3 2	2 1	1 0
PTH7	PTH6	PTH5	PTH4	PTH3	PTH2	PTH1	PTH0	PTG7	PTG6	PTG5	PTG4	PTG3	PTG2	PTG1	PTG0	PTF7	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0	

表 34-9. GPIOC/FGPIOC 寄存器位分配

端口引脚	寄存器位	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9 8	8 7	7 6	6 5	5 4	4 3	3 2	2 1	1 0
保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	保留位	PTI6	PTI5	PTI4	PTI3	PTI2	PTI1	PTI0			

**FGPIO 存储器映射**

绝对地址 (十六进制)	寄存器名称	宽度 (单位: 位)	访问	复位值	小节/页
F800_0000	端口数据输出寄存器 (FGPIOA_PDOR)	32	R/W	0000_0000h	<a href="#">34.3.2/638</a>
F800_0004	端口置位输出寄存器 (FGPIOA_PSOR)	32	W (始终读0)	0000_0000h	<a href="#">34.3.3/638</a>
F800_0008	端口清零输出寄存器 (FGPIOA_PCOR)	32	W (始终读0)	0000_0000h	<a href="#">34.3.4/639</a>
F800_000C	端口跳变输出寄存器 (FGPIOA_PTOR)	32	W (始终读0)	0000_0000h	<a href="#">34.3.5/639</a>
F800_0010	端口数据输入寄存器 (FGPIOA_PDIR)	32	R	0000_0000h	<a href="#">34.3.6/640</a>
F800_0014	端口数据方向寄存器 (FGPIOA_PDDR)	32	R/W	0000_0000h	<a href="#">34.3.7/640</a>
F800_0018	端口输入禁用寄存器 (FGPIOA_PIDR)	32	R/W	FFFF_FFFFh	<a href="#">34.3.8/640</a>
F800_0040	端口数据输出寄存器 (FGPIOB_PDOR)	32	R/W	0000_0000h	<a href="#">34.3.2/638</a>
F800_0044	端口置位输出寄存器 (FGPIOB_PSOR)	32	W (始终读0)	0000_0000h	<a href="#">34.3.3/638</a>
F800_0048	端口清零输出寄存器 (FGPIOB_PCOR)	32	W (始终读0)	0000_0000h	<a href="#">34.3.4/639</a>
F800_004C	端口跳变输出寄存器 (FGPIOB_PTOR)	32	W (始终读0)	0000_0000h	<a href="#">34.3.5/639</a>
F800_0050	端口数据输入寄存器 (FGPIOB_PDIR)	32	R	0000_0000h	<a href="#">34.3.6/640</a>
F800_0054	端口数据方向寄存器 (FGPIOB_PDDR)	32	R/W	0000_0000h	<a href="#">34.3.7/640</a>
F800_0058	端口输入禁用寄存器 (FGPIOB_PIDR)	32	R/W	FFFF_FFFFh	<a href="#">34.3.8/640</a>
F800_0080	端口数据输出寄存器 (FGPIOC_PDOR)	32	R/W	0000_0000h	<a href="#">34.3.2/638</a>
F800_0084	端口置位输出寄存器 (FGPIOC_PSOR)	32	W (始终读0)	0000_0000h	<a href="#">34.3.3/638</a>

下一页继续介绍此表...

## GPIO 存储器映射 (继续)

绝对地址(十六进制)	寄存器名称	宽度(单位:位)	访问	复位值	小节/页
F800_0088	端口清零输出寄存器 (FGPIOC_PCOR)	32	W (始终读0)	0000_0000h	<a href="#">34.3.4/639</a>
F800_008C	端口跳变输出寄存器 (FGPIOC_PTOR)	32	W (始终读0)	0000_0000h	<a href="#">34.3.5/639</a>
F800_0090	端口数据输入寄存器 (FGPIOC_PDIR)	32	R	0000_0000h	<a href="#">34.3.6/640</a>
F800_0094	端口数据方向寄存器 (FGPIOC_PDDR)	32	R/W	0000_0000h	<a href="#">34.3.7/640</a>
F800_0098	端口输入禁用寄存器 (FGPIOC_PIDR)	32	R/W	FFFF_FFFFh	<a href="#">34.3.8/640</a>

### 34.3.2 端口数据输出寄存器 (GPIOx PDOR)

该寄存器配置在各通用输出引脚上驱动的逻辑电平。

地址：基址 基准 + 0h 偏移

## FGPIOx PDOR 字段描述

字段	描述
PDO	<p>端口数据输出</p> <p>特定器件的未实施引脚读取为零。</p> <p>0 假设引脚配置为通用输出，则在引脚上驱动逻辑电平 0。</p> <p>1 假设引脚配置为通用输出，则在引脚上驱动逻辑电平 1。</p>

### 34.3.3 端口置位输出寄存器 (GPIOx PSOR)

该寄存器可以将 PDOR 字段置位。

地址·基址 基准 + 4h 偏移

## FGPIOx PSOR 字段描述

字段	描述
PTSO	端口置位输出

**FGPIOx\_PSOR 字段描述 (继续)**

字段	描述
	<p>如下所示对该寄存器进行写操作将更新 PDOR 中对应位的内容：</p> <p>0 PDORn 中的对应位不变。 1 PDORn 中的对应位置为逻辑 1。</p>

**34.3.4 端口清零输出寄存器 (FGPIOx\_PCOR)**

该寄存器可以将 PDOR 字段清零。

地址: 基址 基准 + 8h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R																	0																	
W																		PTCO																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**FGPIOx\_PCOR 字段描述**

字段	描述
PTCO	<p>端口清零输出</p> <p>如下所示对该寄存器进行写操作将更新端口数据输出寄存器(PDOR)中对应位的内容：</p> <p>0 PDORn 中的对应位不变。 1 PDORn 中的对应位置为逻辑 0。</p>

**34.3.5 端口跳变输出寄存器 (FGPIOx\_PTOR)**

地址: 基址 基准 + Ch 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R																	0																	
W																		PTTO																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**FGPIOx\_PTOR 字段描述**

字段	描述
PTTO	<p>端口跳变输出</p> <p>如下所示对该寄存器进行写操作将更新 PDOR 中对应位的内容：</p> <p>0 PDORn 中的对应位不变。 1 PDORn 中的对应位置为当前逻辑状态的反相电平。</p>

### 34.3.6 端口数据输入寄存器 (FGPIOx\_PDIR)

地址: 基址 基准 + 10h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																PDI																
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

FGPIOx\_PDIR 字段描述

字段	描述
PDI	<p>端口数据输入</p> <p>读取器件的未绑定引脚将返回 0。读取未配置为数字功能的引脚将返回 0。如果禁用端口控制和中断模块，那么 PDIR 中的对应位不更新。</p> <p>0 引脚逻辑电平为逻辑 0，或者未配置为数字功能。 1 引脚逻辑电平为逻辑 1。</p>

### 34.3.7 端口数据方向寄存器 (FGPIOx\_PDDR)

PDDR 将各个端口引脚配置为输入或输出。

地址: 基址 基准 + 14h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																PDD																
W																																

FGPIOx\_PDDR 字段描述

字段	描述
PDD	<p>端口数据方向</p> <p>将各个端口引脚配置为输入或输出。</p> <p>0 引脚配置为通用输入，用于 GPIO 功能。若在 FPIOx_PIDR 寄存器中禁用端口输入，则引脚将为高阻态。 1 引脚配置为通用输出，用于 GPIO 功能。</p>

### 34.3.8 端口输入禁用寄存器 (FGPIOx\_PIDR)

地址: 基址 基准 + 18h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																PID																
W																																

**FGPIOx\_PIDR 字段描述**

字段	描述
PID	端口输入禁用 0 假设引脚配置为数字功能，则该引脚配置为通用输入。 1 引脚未配置为通用输入。读取对应的 PDIR 字段将返回 0。

## 34.4 功能说明

### 34.4.1 通用输入

当引脚配置为数字功能，且对应的端口控制和中断模块已使能时，可通过端口数据输入寄存器得到该引脚的逻辑状态。

### 34.4.2 通用输出

当引脚配置为 GPIO 功能时，可通过端口数据输出寄存器和端口数据方向寄存器来控制每个引脚的逻辑状态。下表介绍了如何把引脚配置为输入/输出。

条件	结果
引脚已配置为 GPIO，并且已将对应的端口数据方向寄存器位清零。	引脚配置为输入。
引脚已配置为 GPIO，并且已将对应的端口数据方向寄存器位置位。	引脚配置为输出，并且引脚的逻辑状态与对应的端口数据输出寄存器相同。

为了让用户可以对通用输出模块实现高效的位操作，我们提供了端口置位、端口清零、以及端口跳变寄存器，以便通过单次寄存器的写操作来置位、清零或切换一个端口内的一个或多个输出。

无需启用对应的端口控制和中断模块来更新端口数据方向寄存器和端口数据输出寄存器（包括置位/清零/跳变寄存器）的状态。

### 34.4.3 IOPORT

GPIO 寄存器也可另外命名，并指向 Cortex-M0+上地址 0xF800\_0000 处的 IOPORT 接口。通过 IOPORT 接口执行的访问与任何指令的获取保持同步，因此可在一周期内完成。



# 第 35 章 键盘中断(KBI)

## 35.1 简介

### 35.1.1 特性

KBI 特性包括：

- 最多 32 个具有单个引脚使能位的键盘中断引脚
- 各键盘中断引脚可编程为：
  - 仅下降沿触发
  - 仅上升沿触发
  - 下降沿加低电平触发
  - 上升沿加高电平触发
- 一个软件使能的键盘中断
- 退出低功耗模式

### 35.1.2 工作模式

本节定义以下模式中的 KBI 操作：

- Wait 模式
- Stop 模式
- Background Debug 模式

### 35.1.2.1 Wait 模式下的 KBI

执行等待指令，将 MCU 置于 Wait 模式。如果需要，则在执行等待指令之前应先使能 KBI 中断(KBI\_SC[KBIE] = 1)，使 KBI 在 MCU 处于 Wait 模式时能够继续工作。如果已使能 KBI 中断(KBI\_SC[KBIE] = 1)，则已使能的 KBI 引脚(KBI\_PE[KBIPEn] = 1)可被用于使 MCU 退出 Wait 模式。

### 35.1.2.2 Stop 模式下的 KBI

执行停止指令可将 MCU 置于 Stop 模式（当已选定 Stop 时），在此模式中 KBI 可异步操作。如果这是所需行为，则在执行停止指令前必须先使能 KBI 中断(KBI\_SC[KBIE] = 1)，使 KBI 在 MCU 处于 Stop 模式时能够继续工作。如果已使能 KBI 中断(KBI\_SC[KBIE] = 1)，则已使能的 KBI 引脚(KBI\_PE[KBIPEn] = 1)可被用于使 MCU 退出 Stop 模式。

### 35.1.3 结构框图

键盘中断模块的结构框图如下所示。

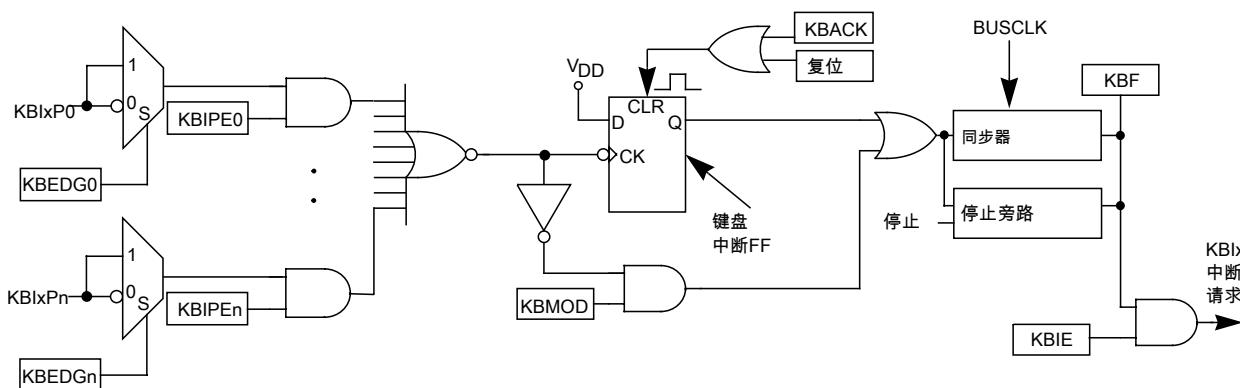


图 35-1. KBI 结构框图

## 35.2 外部信号说明

KBI 输入引脚可用于检测下降沿或同时检测下降沿和低电平中断请求。KBI 输入引脚还可用于检测上升沿，或同时检测上升沿加高电平中断请求。

KBI 的信号属性如下表所示：

表 35-1. 外部信号说明

信号	功能	I/O
KBIxPn	键盘中断引脚	I

### 35.3 寄存器定义

KBI 包括以下寄存器：

- 单个引脚状态和控制寄存器，KBIx\_SC
- 单个引脚使能寄存器，KBIx\_PE
- 单个边沿选择寄存器，KBIx\_ES
- 单个源引脚寄存器，KBIx\_SP

有关所有 KBI 寄存器的绝对地址分配信息，请参见“存储器”章节中的直接页面寄存器汇总。该部分仅按其名称参考寄存器和控制位。

某些 MCU 可能具有不止一个 KBI，因此，寄存器名称中包括用于标识所参考的 KBI 的占位字符。

### 35.4 存储器映像和寄存器

#### KBI 存储器映射

绝对地址 (十 六进制)	寄存器名称	宽度 (单 位: 位)	访问	复位值	小节/页
4007_9000	KBI 引脚使能寄存器 (KBI0_PE)	32	R/W	0000_0000h	<a href="#">35.4.1/646</a>
4007_9004	KBI 边沿选择寄存器 (KBI0_ES)	32	R/W	0000_0000h	<a href="#">35.4.2/646</a>
4007_9008	KBI 状态和控制寄存器 (KBI0_SC)	32	R/W	0000_0000h	<a href="#">35.4.3/647</a>
4007_900C	KBI 源引脚寄存器 (KBI0_SP)	32	R	0000_0000h	<a href="#">35.4.4/648</a>
4007_A000	KBI 引脚使能寄存器 (KBI1_PE)	32	R/W	0000_0000h	<a href="#">35.4.1/646</a>
4007_A004	KBI 边沿选择寄存器 (KBI1_ES)	32	R/W	0000_0000h	<a href="#">35.4.2/646</a>
4007_A008	KBI 状态和控制寄存器 (KBI1_SC)	32	R/W	0000_0000h	<a href="#">35.4.3/647</a>
4007_A00C	KBI 源引脚寄存器 (KBI1_SP)	32	R	0000_0000h	<a href="#">35.4.4/648</a>

### 35.4.1 KBI 引脚使能寄存器 (KBIx\_PE)

KBI\_PE 包含引脚使能控制位。

地址: 基址 基准 + 0h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R																																	
W																																	

复位 0

**KBIX\_PE 字段描述**

字段	描述
KBIPEn	<p>KBI 引脚使能</p> <p>每个 KBIPEn 位使能对应 KBI 中断引脚。</p> <p>0 引脚未使能为 KBI 中断。</p> <p>1 引脚已使能为 KBI 中断。</p>

### 35.4.2 KBI 边沿选择寄存器 (KBIX\_ES)

KBI\_ES 包含边沿选择控制位。

地址: 基址 基准 + 4h 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R																																		
W																																		

复位 0

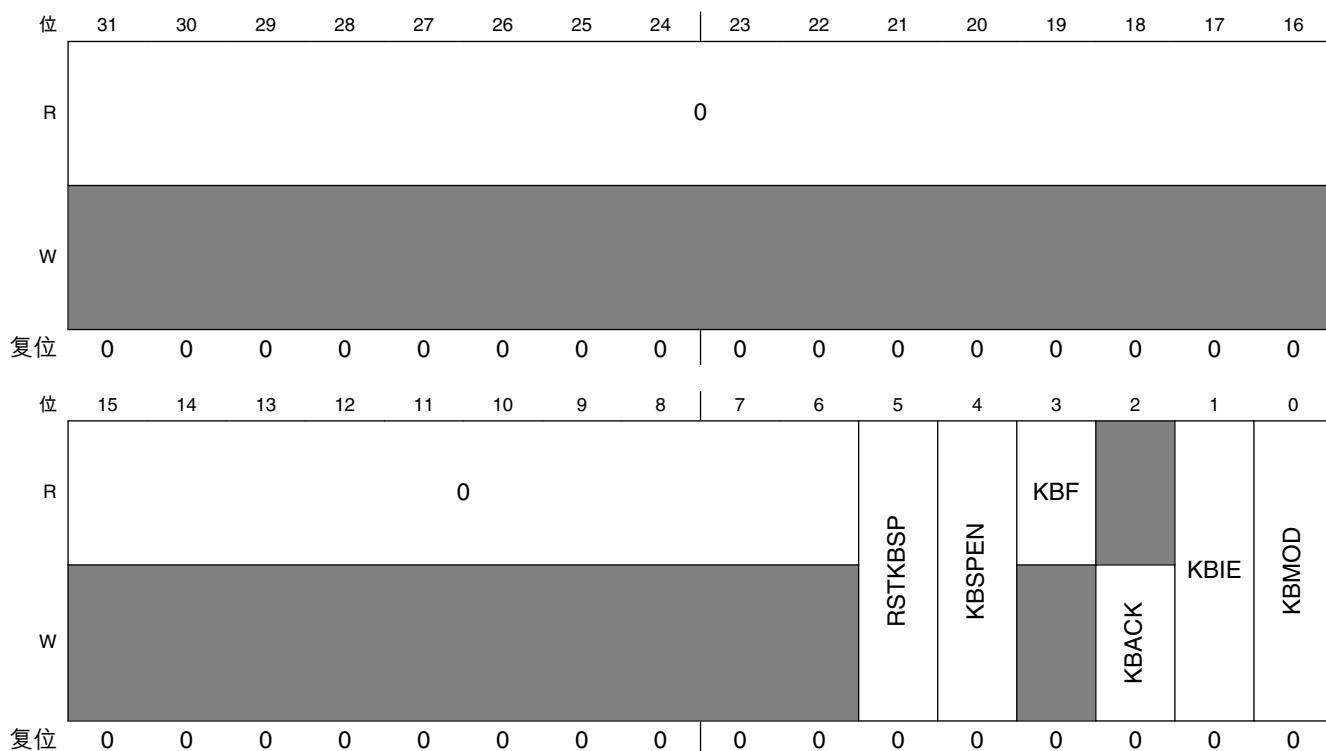
**KBIX\_ES 字段描述**

字段	描述
KBEDGn	<p>KBI 边沿选择</p> <p>每个 KBEDGn 位选择对应引脚的下降沿/低电平或上升沿/高电平功能。</p> <p>0 下降沿/低电平。</p> <p>1 上升沿/高电平。</p>

### 35.4.3 KBI 状态和控制寄存器 (KB<sub>Ix</sub>\_SC)

KBI\_SC 包含用于配置 KBI 的状态标志和控制位。

地址: 基址 基准 + 8h 偏移



KB<sub>Ix</sub>\_SC 字段描述

字段	描述
31–6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5 RSTKBSP	复位 KBI_SP 寄存器 在 RSTKBSP 中写入 1 即可清零 KB <sub>Ix</sub> SP 寄存器。此位始终读为 0。
4 KBSPEN	实际 KBI_SP 寄存器使能 0 要读取的键盘源引脚的实时值 1 当中断标志出现时要读取的 KB <sub>Ix</sub> SP 寄存器中的锁存值
3 KBF	KBI 中断标志 表示检测到 KBI 中断请求的时间。写操作对 KBF 无效。 0 未检测到的 KBI 中断请求。 1 检测到的 KBI 中断请求。
2 KBACK	KBI 应答 在 KBACK 中写入 1 是标志清零机制的一部分。

下一页继续介绍此表...

**KBIx\_SC 字段描述 (继续)**

字段	描述
1 KBIE	KBI 中断使能  确定 KBI 中断是否已使能。  0 KBI 中断未使能。 1 KBI 中断已使能。
0 KBMOD	KBI 检测模式  KBMOD ( 与 ES[KBEDG]一起 ) 控制 KBI 中断引脚的检测模式。  0 键盘仅检测边沿。 1 键盘检测边沿和电平。

**35.4.4 KBI 源引脚寄存器 (KBIx\_SP)**

KBIx 源引脚寄存器表示 KBI 的源引脚。其复位值来自系统复位。

地址: 基址 基准 + Ch 偏移

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	SP															
W																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**KBIx\_SP 字段描述**

字段	描述
SP	KBI 源引脚  该字段为只读，它表示定义为键盘中断的活动引脚是否被按下，当有效按下对应键盘引脚时，相应的位将置 1，仅系统复位或在 RSTKBSP 位中写入 1 才能清零此寄存器。

**35.5 功能说明**

该片上外设模块被称为键盘中断模块，因为其最初设计用于简化键盘开关行列式矩阵的连接和使用。但是，这些输入可以作为有用的额外的外部中断的输入，成为一个很好的将 MCU 从停止或等待低功耗模式唤醒的外部方式。

KBI 模型最多允许 8 个引脚用作为额外的中断源。在 KBIx\_PE[KBIPEn]位中单独写入将使能或禁用各 KBI 引脚。各 KBI 引脚可在 KBIx\_SC[KBMOD]位上配置为基于边沿触发或基于边沿和电平触发。边沿触发可通过软件编程为下降沿出发或上升沿触发；电平可配置为低电平触发或高电平触发。边沿极性或边沿和电平敏感度可使用 KBIx\_ES[KBEDGn]位选择。

### 35.5.1 边沿触发

同步逻辑用于检测边沿。当已使能的键盘中断( $KBIx\_PE[KBIPEn]=1$ )输入信号在一个总线周期中被采集为逻辑 1(无效电平)并且在下一个周期中被采集为逻辑 0(有效电平), 则会检测到下降沿。当该输入信号在一个总线周期中被采集为逻辑 0(无效电平)并且在下一个周期中被采集为逻辑 1(有效电平), 则会检测到上升沿。

检测到第一个边沿之前, 所有已使能的键盘中断输入信号必须处于无效逻辑电平。检测到任何边沿后, 在检测任何新边沿前, 所有已启用的键盘中断输入信号必须返回到无效电平。

已使能 KBI 引脚上的有效边沿将置位  $KBIx\_SC[KBF]$ 。如果  $KBIx\_SC[KBIE]$  已置位, 则 MPU 中会出现中断请求。在  $KBIx\_SC[KBACK]$  中写入 1 将会清零  $KBIx\_SC[KBF]$ 。

### 35.5.2 边沿和电平触发

已使能 KBI 上的有效边沿或电平将置位  $KBIx\_SC[KBF]$ 。如果  $KBIx\_SC[KBIE]$  已置位, 则 MCU 中会出现中断请求。如果所有已使能的键盘输入处于其无效电平, 则在  $KBIx\_SC[KBACK]$  中写入 1 将会清零  $KBIx\_SC[KBF]$ 。当尝试通过在  $KBIx\_SC[KBACK]$  中写入 1 来清零  $KBIx\_SC[KBF]$  时, 如果任何已使能的 KBI 引脚电平变为有效值, 则  $KBIx\_SC[KBF]$  将保持置位。

### 35.5.3 KBI 上拉电阻

各 KBI 引脚(如已由  $KBIx\_PE$  使能)可通过关联 I/O 端口的拉使能寄存器进行配置, 请参见“[端口控制\(PORT\)](#)”部分, 以使用:

- 使用内部上拉电阻
- 无电阻

如果内部上拉电阻已被使能的 KBI 引脚使能, 则相关 I/O 端口的拉选择寄存器(参见“[I/O 端口](#)”一章)可用于选择内部上拉电阻。

### 35.5.4 KBI 初始化

首次使能键盘中断时, 可能会获得错误的键盘中断标志。要在键盘初始化期间防止虚假的中断请求, 用户应执行以下操作:

1. 通过清除 KBIx\_SC[KBIE]屏蔽键盘中断。
2. 通过设置适当的 KBIx\_ES[KBEDGn]位置位而使能 KBI 极性。
3. 使用内部上拉电阻之前，先配置 PORT\_PUE0 和 PORT\_PUE1 中的关联位。
4. 通过设置适当的 KBIx\_PE[KBIPEn]位置位而使能 KBI 引脚。
5. 在 KBIx\_SC[KBACK]中进行写操作将清除任何错误中断。
6. 在 KBIx\_SC[RSTKBSP]中进行写操作将清除 KBIx\_SP 中的任何错误。
7. 使 KBIx\_SC[KBIE]置位以使能中断。

# 第 36 章

## 第 2 版的版本记录

表 36-1. 通篇改动

- 更新了支持的器件。

表 36-2. “关于本文”一章的改动

无实质性的内容改动。

表 36-3. “简介”一章的改动

- 文字性改动。
- [存储器和存储器接口](#) 更新以显示 64/128 KB Flash 存储器。
- 更新了[可订购部件编号](#) 部分。

表 36-4. “芯片配置”一章的改动

- 增加了[备用非易失性 IRC 用户微调说明](#) 部分。
- 下列部分更新以显示所有支持的器件：
  - 更新了[Flash 存储器大小](#) 部分。
  - 更新了[SRAM 大小](#) 部分
  - 更新了[ADC 实例化信息](#) 部分。

表 36-5. “存储器映射”一章的改动

- 更新了[外设 ID 寄存器 \(ROM\\_PERIPHID \$n\$ \)](#) 和[组件 ID 寄存器 \(ROM\\_COMPID \$n\$ \)](#) 寄存器的注释。

表 36-6. “时钟分布”一章的改动

- [FTM 和 PWT 计时](#) 部分中更新了注释。

表 36-7. “复位和引导”一章的改动

无实质性的内容改动。

**表 36-8. “电源管理”一章的改动**

无实质性的内容改动。

**表 36-9. “加密”一章的改动**

- Flash 加密 部分中更新了注释。

**表 36-10. “调试”一章的改动**

无实质性的内容改动。

**表 36-11. “信号多路复用和信号说明”一章的改动**

无实质性的内容改动。

**表 36-12. “端口控制（PORT）”一章的改动**

无实质性的内容改动。

**表 36-13. “系统集成模块”一章的改动**

无实质性的内容改动。

**表 36-14. “电源管理控制器”一章的改动**

无实质性的内容改动。

**表 36-15. “杂项控制模块”一章的改动**

无实质性的内容改动。

**表 36-16. “外设桥”一章的改动**

无实质性的内容改动。

**表 36-17. “WDOG 定时器”一章的改动**

- 更新了特性 部分。

**表 36-18. “位操作引擎”一章的改动**

无实质性的内容改动。

**表 36-19. “Flash 存储器模块”一章的改动**

- 更新了 Flash 存储器特性 部分中的特性列表。
- 更新了 Flash 存储器映射 部分
- 更新了系统复位之后的 Flash 初始化 部分
- 增加了图图 18-3。
- Flash 命令汇总 中增加了文本“这还会触发非法访问异常情况”
- 更新了寄存器 FTMRE\_FSEC 中位“SEC”的位说明

**表 36-20. “Flash 存储器控制器”一章的改动**

无实质性的内容改动。

**表 36-21. “内部时钟源”一章的改动**

- 更新了特性
- 更新了寄存器 ICS\_C1 中位“RDIV”的位说明
- 更新了寄存器 ICS\_C3 中位“SCTRIM”的位说明
- 更新了寄存器 ICS\_C4 中位“SCFTRIM”的位说明
- 寄存器 ICS\_S 中的 LOCK 位从 TRIM[7:0]变更为 SCTRIM[7:]
- 更新了下列部分中的代码段：
  - 初始化 FEI 模式
  - 初始化 FBI 模式
  - 初始化 FEE 模式
  - 初始化 FBE 模式

**表 36-22. “振荡器”一章的改动**

- 更新了寄存器 OSC\_CR 中位“RANGE”的位说明

**表 36-23. “循环冗余校验”一章的改动**

- 更新了 CRC 初始化/重新初始化 部分。

**表 36-24. “中断”一章的改动**

无实质性的内容改动。

**表 36-25. “模数转换器”一章的改动**

- 更新了 FIFO 操作 部分。
- 图 24-3 中作了小幅度的文字性更新

**表 36-26. “模拟比较器”一章的改动**

无实质性的内容改动。

表 36-27. “FlexTimer 模块”一章的改动

- 更新了 [FlexTimer 的基本理念](#) 部分
- FTMx\_CnSC 寄存器下的表“模式、边沿和电平选择”作了小幅的文字性更新。
- [FlexTimer 的基本理念](#) 部分作了小幅的文字性更新。
- 更新了 [向上-向下计数](#) 部分中的注释。
- 删除了 [输入捕捉模式](#) 部分中的注释。
- 删除了 [输出比较模式](#) 部分中的注释。
- 更新了 [边沿对齐 PWM \(EPWM\)模式](#) 部分中的注释。
- 删除了 [中心对齐 PWM \(CPWM\)模式](#) 部分中的注释。
- 移除了 [组合模式](#) 部分中的列表项“FTMEN = 1”
- 更新了 [互补模式](#) 部分
- 删除了 [PWM 同步](#) 部分中的注释
- 移除了 [反相](#) 部分中的列表项“FTMEN = 1”、“COMBINE = 1”和“CPWMS = 0”，并更新了注释
- 删除了 [软件输出控制](#) 部分中的注释
- 更新了 [死区插入](#) 部分中的注释
- 故障控制 部分中的“如果(FTMEN = 1)且(FAULTM[1:0] ≠ 0:0)，则故障控制使能”改动为“如果(FAULTM[1:0] ≠ 0:0)，则故障控制使能”。
- 更新了 [初始化](#) 部分中的注释
- 删除了 [通道触发器输出](#) 部分中的注释
- 删除了 [初始化触发](#) 部分中的注释
- 从 [双沿捕捉模式](#) 部分中的“如果 FTMEN = 1 且 DECAPEN = 1，则选择双沿捕获模式”移除了“FTMEN=1”。
- 从 [一次性捕捉模式](#) 和 [连续捕捉模式](#) 部分中的“当(FTMEN = 1)、(DECAPEN = 1)且(MS(n)A = 0)时，选择单次捕获模式”移除了“FTMEN=1”。
- [Debug 模式](#) 部分中增加了注释。
- 删除了 [中间加载](#) 部分中的注释
- [全局时基\(GTB\)](#) 部分作了小幅的文字性更新
- 增加了 [初始化流程](#) 部分
- 移除了 [STATUS](#) 寄存器的注释。
- 更新了 [MODE](#) 寄存器中 FTMEN 的位字段说明。
- 更新了 [COMBINE](#) 寄存器的位字段说明。
- 删除了 [FlexTimer 的基本理念](#) 部分中的注释。

表 36-28. “脉宽定时器”一章的改动

- [功能框图](#) 中作了小幅更新
- [功能框图](#) 部分中增加了注释
- 更新了 [PWT\\_R1](#) 寄存器中 PCLKS 位的位字段说明。
- 更新了 [图 27-2](#)

表 36-29. “周期性中断定时器”一章的改动

- [简介](#) 部分中作了小幅的文字性更新。
- 更新了 [链接定时器配置示例](#) 部分。

表 36-30. “实时计数器”一章的改动

无实质性的内容改动。

表 36-31. “串行外设接口”一章的改动

无实质性的内容改动。

**表 36-32. “I2C”一章的改动**

- [特性](#) 部分中作了小幅的文字性更新。

**表 36-33. “MSCAN”一章的改动**

- 更新了“MSCAN 发送缓冲区选择寄存器”部分。
- [发送结构](#) 部分增加图“报文缓冲区组织用户模型”的交叉参考

**表 36-34. “UART”一章的改动**

- [快速数据公差](#) 部分中作了小幅的文字性更新

**表 36-35. “通用输入/输出 (GPIO)”一章的改动**

- 更新了[特性](#) 部分。
- [详细信号说明](#) 部分中增加了注释。

**表 36-36. “键盘中断”一章的改动**

无实质性的内容改动。



**How to Reach Us:**

**Home Page:**  
[freescale.com](http://freescale.com)

**Web Support:**  
[freescale.com/support](http://freescale.com/support)

本文档中的信息仅供系统和软件实施方使用 Freescale 产品。本文并未明示或暗示授予利用本文档信息进行设计或者加工集成电路的版权许可。

Freescale 保留对本文档中所述任何产品进行更改的权利，恕不另行通知。

Freescale 对其产品在任何特定用途方面的适用性不做任何担保、表示或保证，也不承担因为应用或使用产品或电路所产生的任何责任，明确拒绝承担包括但不限于因果性或附带损害在内的所有责任。Freescale 的数据表和/或规格中所提供的“典型”参数在不同应用中可能并且确实不同，实际性能会随时间而有所变化。所有操作参数，包括“典型值”在内，在每个客户应用中必须经由技术专家进行验证。Freescale 未转让与其专利权及其他权利相关的许可。Freescale 根据标准销售条款和条件出售产品，该标准条款和条件可参考如下网址：[freescale.com/SalesTermsandConditions](http://freescale.com/SalesTermsandConditions)。

Freescale、Freescale 标志以及 Kinetis 是 Freescale Semiconductor, Inc. 的商标，已在美国专利商标局注册的商标。所有其他产品或服务名称都是其各自所有者的财产。ARM 和 Cortex-M0+ 是 ARM 有限公司的注册商标。

©2014 Freescale Semiconductor, Inc.

Document Number S9KEA128Z80M48SF0RM  
Revision 2, July 2014

