

User Guide

megawin

MG32x02z
用户手册

版本 2.2
日期 2019/12/20

目录

1. 文档用法	26
1.1. 文档规则	26
1.2. 方框图词汇表	26
1.3. 电源工作模式指示	26
1.4. 词汇表	27
2. 芯片与系统	28
2.1. 芯片概述	28
2.2. 适用芯片	29
2.2.1. 芯片配置概览	29
2.3. 芯片主体	30
2.4. CPU 核心	32
2.4.1. CPU 特性	32
2.4.2. ARM Cortex-M0 处理器	32
2.5. 存储器组织	33
2.5.1. CPU 内存映射	33
2.5.2. 外围内存边界	35
2.5.3. 启动模式	38
2.5.4. 内存代码锁定	38
2.6. 芯片调试	39
2.6.1. 芯片 DAP	39
2.6.2. SWD 接口	39
3. 系统电源	40
3.1. 简介	40
3.2. 芯片电源特性	40
3.3. 电源工作模式	41
3.4. 供电	42
3.5. 电源控制器块	43
3.6. 电源电压检测	43
3.6.1. POR/LVR 检测器	43
3.6.2. 掉电检测器	43
3.6.3. 电源电压检测阈值	44
3.7. 中断和复位	45
3.7.1. PW 中断标志	45
3.7.2. PW 复位事件	45
3.8. 寄存器保护和锁定	46
3.9. 处理器掉电	46
3.10. 掉电模式下设备电源使能控制	46
3.11. 唤醒控制	47
3.11.1. 唤醒事件源	47
3.11.2. 唤醒和中断	48
3.12. 内部设备控制	51
3.12.1. 内部电源 LDO	51
3.12.2. 内部参考电压	51
3.13. 芯片供电应用电路	51
4. 系统复位	52
4.1. 简介	52
4.2. 芯片复位特性	52
4.3. 复位源控制器	53

4.4. 芯片复位	55
4.4.1. 芯片复位等级	55
4.4.2. 上电复位和芯片复位时序	56
4.4.3. 外部复位时序	56
4.5. 复位事件	58
4.5.1. 复位事件源	58
4.5.2. 复位事件控制	58
4.6. 寄存器保护和锁定	60
4.7. 模块复位	61
4.7.1. 模块软件复位	61
4.7.2. GPIO 复位控制	62
4.8. 外部复位应用电路	63
5. 系统时钟	64
5.1. 简介	64
5.2. 芯片时钟特性	64
5.3. 时钟源控制器	65
5.4. 中断和复位	66
5.4.1. CSC 中断标志	66
5.4.2. CSC 时钟状态	66
5.4.3. CSC 复位事件	67
5.5. 寄存器保护和锁定	67
5.6. 系统时钟控制	67
5.6.1. 系统时钟源	67
5.6.2. 高速时钟和低速时钟	68
5.6.3. PLL 时钟	68
5.6.4. 内部系统时钟	69
5.7. 模块进程时钟控制	69
5.7.1. 模块进程时钟选择	69
5.7.2. 模块进程时钟使能	70
5.7.3. 掉电模式下设备时钟使能控制	72
5.8. 模块事件时钟	73
5.9. 内部时钟输出控制	73
5.10. 晶振电路	74
6. 系统普通控制	75
6.1. 简介	75
6.2. 特性	75
6.3. 中断和事件	76
6.4. 芯片制造 ID	76
6.5. 系统备份寄存器	76
7. 系统内存	77
7.1. 简介	77
7.2. 特性	77
7.2.1. 内嵌内存	77
7.2.2. 内存控制器	77
7.3. 内存控制器	78
7.4. 使能和时钟	79
7.5. 中断和事件	79
7.5.1. 内存控制器中断控制和复位	79
7.5.2. MEM 中断标志	79

7.6. 寄存器保护和锁定.....	80
7.7. 启动和片上内存.....	80
7.7.1. 内存启动模式.....	80
7.7.2. 片上闪存.....	80
7.7.3. 片上数据 RAM.....	82
7.8. 内存控制器功能.....	82
7.8.1. 闪存访问.....	82
7.8.2. 硬件设置字节闪存.....	85
7.8.3. 用于闪存的 ICP/ISP/IAP.....	86
7.8.4. 闪存访问限制.....	86
7.8.5. CPU 代码执行和暂停.....	87
7.8.6. 内存访问错误管理.....	87
7.9. 不同产品的闪存设置.....	88
8. 硬件设置.....	89
8.1. 简介.....	89
8.2. 硬件设置字节.....	89
8.3. CFG 设置寄存器.....	91
8.3.1. CFG 寄存器保护和锁定.....	91
8.3.2. 工厂 ADC 校准值.....	91
9. GPIO (通用 IO).....	92
9.1. 简介.....	92
9.2. 特性.....	92
9.3. 控制块.....	93
9.4. IO 模式.....	94
9.4.1. IO 模式控制块.....	94
9.4.2. IO 设置.....	95
9.5. IO 结构.....	96
9.5.1. 模拟 IO 和数字输入结构.....	96
9.5.2. 推挽输出结构.....	97
9.5.3. 开漏输出结构.....	98
9.5.4. 准双向 IO 结构.....	98
9.6. IO 端口访问.....	99
9.6.1. GPIO IO 控制.....	99
9.6.2. 设置和清除控制.....	100
9.6.3. 位方式 IO 控制.....	100
9.7. 功能复用选择.....	100
9.7.1. 功能复用选择控制.....	100
9.7.2. 特殊引脚的功能复用.....	101
9.7.3. GPIO AFS 锁定.....	102
10. 中断.....	103
10.1. 简介.....	103
10.2. 特性.....	103
10.3. 中断结构.....	104
10.3.1. 中断源.....	104
10.3.2. 异常类型.....	105
10.3.3. 异常处理程序.....	106
10.3.4. 中断优先级.....	106
10.3.5. 锁定 Cortex-M0.....	106
10.4. 中断控制块.....	107

10.5. 嵌套向量中断控制器	108
10.5.1. NVIC 功能	108
10.5.2. NVIC 异常控制	108
10.5.3. 电平和脉冲中断	109
10.5.4. Hard Fault 处理	110
10.6. 唤醒中断控制器	110
10.7. 外部中断控制器	111
10.7.1. EXIC 中断控制	111
10.7.2. 外部端口输入中断	112
11. GPL (通用逻辑)	114
11.1. 简介	114
11.2. 特性	114
11.3. 控制块	115
11.4. 时钟	116
11.4.1. GPL 时钟控制	116
11.5. GPL 功能控制	117
11.5.1. 字节顺序变换	117
11.5.2. 位顺序改变	118
11.5.3. 数据反相	118
11.5.4. 奇偶校验	118
11.5.5. 循环冗余校验计算	118
11.6. GPL DMA 操作	120
11.6.1. DMA 模块设置	120
11.6.2. GPL DMA 控制	120
12. DMA (直接内存访问)	121
12.1. 简介	121
12.2. 特性	121
12.3. 配置	121
12.3.1. 芯片配置	121
12.4. 控制块	122
12.5. IO 线	122
12.5.1. IO 信号	122
12.5.2. IO 设置	122
12.6. 使能和时钟	123
12.7. 中断和事件	123
12.7.1. DMA 中断控制和状态	123
12.7.2. DMA 中断标志	123
12.8. DMA 控制	125
12.8.1. DMA 源和目的地	125
12.8.2. DMA 通道仲裁	127
12.8.3. DMA 通道工作	127
12.8.4. DMA SRAM 使用	129
12.9. DMA 传输	130
12.9.1. DMA 传输设置和序列	130
12.9.2. DMA SLEEP 模式下工作	130
12.9.3. 外围 DMA RX 请求和承认	131
12.9.4. 外围 DMA TX 请求和承认	132
12.9.5. 外围 DMA 传输停止	132
12.9.6. 外围中断标志控制	133

12.10. DMA 外部请求触发输入	135
13. EMB (外置内存总线)	136
13.1. 简介	136
13.2. 特性	136
13.3. 配置	136
13.3.1. 芯片配置	136
13.4. 控制块	137
13.5. IO 线	137
13.5.1. IO 信号	137
13.5.2. IO 设置	138
13.6. 使能和时钟	138
13.6.1. EMB 全局使能	138
13.6.2. EMB 时钟控制	138
13.7. 中断和事件	139
13.7.1. EMB 中断控制和状态	139
13.7.2. EMB 中断标志	139
13.8. EMB IO 控制	140
13.8.1. EMB 时钟和指令信号控制	140
13.8.2. EMB 地址和数据信号控制	142
13.8.3. 用于外部设备的信号映射建议	142
13.9. EMB 内存控制	143
13.9.1. EMB 内存空间	143
13.9.2. AHB 到外部内存传输	144
13.9.3. EMB 写保护	144
13.9.4. EMB 时序控制	145
13.10. EMB 地址和数据接口模式	147
13.10.1. EMB 16 位 MA 和 16 位 MD	148
13.10.2. EMB 16 位 MA 和复用 16 位 MD	149
13.10.3. EMB 复用 16 位具有 2 个地址相位的 MAD	150
13.11. EMB 设备接口和时序	151
13.11.1. SRAM 接口和访问时序	151
13.11.2. NOR-Flash 接口和访问时序	153
13.11.3. NAND-Flash 接口和访问时序	155
13.11.4. LCD 接口和访问时序	157
13.12. EMB DMA 操作	159
13.12.1. DMA 模块设置	159
13.12.2. EMB DMA 控制	159
13.13. EMB 应用注释	159
13.13.1. EMB 信号引脚建议	159
14. APB 一般控制	161
14.1. 简介	161
14.2. 特性	161
14.3. 配置	161
14.3.1. 芯片配置	161
14.4. 中断和事件	162
14.4.1. APB 中断控制和状态	162
14.4.2. APB 中断标志	162
14.5. 定时器一般控制	163
14.5.1. 定时器同步使能控制	163

14.5.2. 定时器一般触发/时钟源选择	164
14.6. OBM 控制	165
14.6.1. OBM 输出通道	166
14.6.2. OBM 中止通道	167
14.6.3. OBM 工作模式	167
14.7. IR 控制	168
14.7.1. IR 控制接口	168
14.7.2. IR 发送调制	169
15. I2C (集成电路总线).....	170
15.1. 简介	170
15.2. 特性	170
15.3. 配置	171
15.3.1. 芯片配置	171
15.3.2. 模块功能	171
15.4. 控制块	172
15.5. IO 线	172
15.5.1. IO 信号	172
15.5.2. IO 设置	172
15.6. 使能和时钟	173
15.6.1. I2C 全局使能	173
15.6.2. I2C 时钟控制	173
15.7. 中断和事件	174
15.7.1. I2C 中断控制和状态	174
15.7.2. I2C 子范围中断	174
15.7.3. I2C 中断标志	175
15.7.4. I2C 控制标志	176
15.8. I2C 连接应用	177
15.9. I2C 基本控制	178
15.9.1. I2C 基本协议	178
15.9.2. I2C 控制模式	178
15.9.3. I2C 从机地址	178
15.9.4. I2C 数据寄存器和移动缓冲	178
15.9.5. I2C 访问指令	179
15.9.6. I2C START/STOP/数据改变	179
15.10. I2C 字节模式控制	180
15.10.1. I2C 字节模式收发	180
15.10.2. I2C 事件码	180
15.11. I2C 缓冲模式控制	185
15.11.1. I2C 缓冲模式数据缓冲	185
15.11.2. I2C 数据缓冲控制	186
15.11.3. I2C 缓冲模式访问指令	186
15.11.4. I2C 主机收发	186
15.11.5. I2C 从机收发	189
15.12. I2C 功能控制	191
15.12.1. I2C 信号驱动控制时序	191
15.12.2. I2C 从机 SCL 拉伸	192
15.12.3. STOP 模式唤醒	193
15.12.4. I2C 超时定时器控制	193
15.12.5. I2C NACK 控制	194

15.13. I2C 错误管理	195
15.13.1. I2C 总线错误事件	195
15.13.2. I2C 主机发送 NACK 检测	195
15.13.3. I2C SCL 拉伸禁用时从机数据溢出	196
15.14. I2C DMA 操作	197
15.14.1. DMA 模块设置	197
15.14.2. I2C DMA 控制	197
15.14.3. I2C 的 DMA 中断标志控制	197
16. UART (通用异步收发传输器)	198
16.1. 简介	198
16.2. 特性	198
16.3. 配置	199
16.3.1. 芯片配置	199
16.3.2. 模块功能	200
16.4. 控制块	201
16.5. IO 线	201
16.5.1. IO 信号	201
16.5.2. IO 设置	202
16.6. 使能和时钟	202
16.6.1. UART 全局使能	202
16.6.2. UART 时钟控制	202
16.6.3. UART PSC 超时信号输出	204
16.7. 中断和事件	205
16.7.1. UART 中断控制和状态	205
16.7.2. UART 子范围中断	206
16.7.3. UART 中断标志	206
16.8. UART 模块 IO 控制	208
16.8.1. UART IO 控制	208
16.8.2. UART IO 模式	209
16.8.3. UART 循环模式	209
16.9. UART 连接应用	210
16.9.1. UART UART/IrDA 模式连接	210
16.9.2. UART LIN 模式连接	210
16.9.3. UART 智能卡模式连接	211
16.9.4. UART SPI 主机模式连接	211
16.10. UART 格式	212
16.10.1. UART 数据格式设置	212
16.10.2. UART 校验位	213
16.10.3. UART Idle 线格式设置	213
16.10.4. UART 中止状态格式设置	214
16.11. UART 基本控制	215
16.11.1. UART 控制模式设置	215
16.11.2. UART 工作模式设置	215
16.11.3. UART 发送	215
16.11.4. UART 接收	217
16.12. UART 数据缓冲	218
16.12.1. UART 数据缓冲控制	218
16.12.2. UART 数据反相	219
16.12.3. UART 移动缓冲	219

16.12.4. UART 数据缓冲清除	219
16.13. UART 多处理器	220
16.13.1. UART 多处理器工作模式	220
16.13.2. UART 多处理器和静音模式	220
16.13.3. UART 多处理器地址设置	220
16.13.4. UART 多处理器从机地址发送	221
16.14. UART 中止状态发送	222
16.15. UART 波特率控制	223
16.15.1. UART 波特率定时器启动/关闭控制	223
16.15.2. UART 波特率校准	223
16.15.3. UART BR 超时信号输出	225
16.15.4. UART 波特率设置示例	225
16.16. UART 数据接收和采样	226
16.16.1. UART 数据采样	226
16.16.2. UART 接收噪音字符	226
16.16.3. UART 接收过采样	227
16.16.4. UART 接收位时间错误公差	227
16.17. UART TMO 超时控制	229
16.17.1. UART TMO 超时定时器启动/关闭控制	230
16.17.2. UART 保持时间和超时检测	230
16.17.3. UART 自动校准超时	231
16.17.4. UART 中止状态超时	232
16.17.5. UART TMO 超时信号输出	232
16.18. UART 错误管理	233
16.18.1. UART 中止和校验/帧错误检测	233
16.18.2. UART TX 错误检测和重传控制	234
16.18.3. UART RX 校验错误检测和重试控制（用于智能卡）	235
16.19. UART 静音模式控制	236
16.20. UART 同步模式	237
16.20.1. UART 同步模式时序	237
16.20.2. UART 同步模式 NSS 控制	238
16.21. UART 智能卡时钟输出	239
16.21.1. 智能卡时钟频率计算	239
16.21.2. UART 智能卡时钟输出设置示例	240
16.22. UART IrDA 控制	241
16.22.1. UART IrDA 时序	241
16.22.2. UART IrDA 数据采样	242
16.22.3. UART IrDA 占用标志	243
16.23. UART DE 控制	244
16.24. UART 硬件流控制	245
16.24.1. UART 硬件流控制连接	245
16.24.2. UART CTS 硬件流控制	245
16.24.3. UART RTS 硬件流控制	246
16.24.4. UART 软件流控制	247
16.25. UART 接收硬件停止和捕获控制	247
16.26. UART DMA 操作	248
16.26.1. DMA 模块设置	248
16.26.2. UART DMA 控制	248
16.26.3. UART 的 DMA 中断标志控制	249

17. SPI (串行外设接口).....	250
17.1. 简介	250
17.2. 特性	250
17.3. 配置	251
17.3.1. 芯片配置	251
17.3.2. 模块功能	251
17.4. 控制块	252
17.5. IO 线	252
17.5.1. IO 信号	252
17.5.2. IO 设置	253
17.6. 使能和时钟	253
17.6.1. SPI 全局使能	253
17.6.2. SPI 时钟控制	253
17.7. 中断和事件	253
17.7.1. SPI 中断控制和状态	253
17.7.2. SPI 中断标志	254
17.8. SPI 模式 IO 控制	256
17.8.1. SPI IO 控制	256
17.8.2. SPI IO 模式	257
17.8.3. SPI 循环模式	258
17.9. SPI 应用连接	259
17.9.1. 全双工通信	259
17.9.2. 单工通信	259
17.9.3. 半双工通信	260
17.9.4. 多从机通信	261
17.9.5. 双主机通信	262
17.10. SPI 基本控制	263
17.10.1. SPI 时钟	263
17.10.2. SPI 发送	264
17.10.3. SPI 接收	264
17.10.4. SPI 基本时序	265
17.10.5. SPI DTR 模式	267
17.10.6. SPI NSS 模式	267
17.11. SPI 数据缓冲	270
17.11.1. SPI 数据缓冲控制	270
17.11.2. SPI 数据帧	271
17.11.3. SPI 发送数据帧大小	271
17.11.4. SPI 数据缓冲清除	272
17.12. SPI 数据模式	273
17.12.1. SPI 数据模式 – SPI	274
17.12.2. SPI 数据模式 – SPI-1	274
17.12.3. SPI 数据模式 – SPI-2	275
17.12.4. SPI 数据模式 – SPI-2C	275
17.12.5. SPI 数据模式 – SPI-4	276
17.12.6. SPI 数据模式 – SPI-4C	276
17.12.7. SPI 数据模式 – SPI-8	277
17.13. SPI 串行 Flash 支持	278
17.14. SPI 接收溢出	279
17.15. SPI 发送错误	279

17.16. SPI 主机模式改变检测.....	280
17.17. SPI DMA 操作	281
17.17.1. DMA 模块设置.....	281
17.17.2. SPI DMA 控制	281
17.17.3. SPI 的 DMA 中断标志控制.....	281
18. 定时器(通用定时器).....	282
18.1. 简介	282
18.2. 特性	282
18.3. 配置	283
18.3.1. 芯片配置.....	283
18.3.2. 模块功能.....	283
18.4. 控制块	284
18.5. IO 线.....	287
18.5.1. IO 信号	287
18.5.2. IO 设置	287
18.6. 使能和时钟	288
18.6.1. 定时器使能	288
18.6.2. 定时器工作时钟控制.....	288
18.6.3. 定时器模块时钟和触发源.....	288
18.7. 中断和事件	290
18.7.1. 定时器更新事件.....	290
18.7.2. 定时器中断控制和状态	290
18.7.3. 定时器中断标志.....	291
18.8. 定时器操作	293
18.8.1. 定时器工作模式.....	293
18.8.2. 定时器和计数器控制.....	294
18.8.3. 定时器工作模式和控制验证	294
18.9. 定时器触发控制块.....	295
18.9.1. 定时器触发输入事件.....	295
18.9.2. 定时器触发输出事件.....	297
18.10. 定时器输入/输出通道.....	298
18.10.1. TM3x 定时器输入/输出通道块	298
18.10.2. TM2x 定时器输入/输出通道块	299
18.10.3. 定时器输入通道控制.....	299
18.10.4. 定时器时钟输出	300
18.11. 定时器捕获和比较块.....	302
18.11.1. 定时器通道模式.....	302
18.11.2. 软件输入捕获和输出比较产生	302
18.11.3. 定时器捕获和比较寄存器控制	302
18.12. 定时器输入捕获.....	303
18.12.1. 捕获数据和边沿选择.....	303
18.12.2. 捕获数据溢出	304
18.12.3. 捕获控制和状态	304
18.13. 定时器输出比较和 PWM.....	308
18.13.1. 比较重载寄存器	308
18.13.2. 比较输出状态	308
18.13.3. 16 位比较/PWM 模式.....	308
18.13.4. 双 8 位比较/PWM 模式	309
18.13.5. 输出比较/PWM 控制和状态	309

18.13.6. 定时器输出比较和 PWM 时序.....	310
18.13.7. PWM 死区控制	314
18.14. 定时器输出控制块	315
18.14.1. TM2x 定时器输出控制块.....	315
18.14.2. TM3x 定时器输出控制块.....	316
18.14.3. 定时器输出使能预载控制.....	317
18.15. 定时器中止控制块	318
18.15.1. 中止使能和事件源	318
18.15.2. 中止控制模式	318
18.16. QEI 控制块	320
18.16.1. QEI 控制模式.....	320
18.16.2. QEI 输入信号.....	321
18.16.3. QEI 控制模式 1,2.....	321
18.16.4. QEI 控制模式 3,4,5.....	322
18.17. 定时器 DMA 操作	324
18.17.1. DMA 模块设置.....	324
18.17.2. 定时器 DMA 控制	324
18.17.3. 定时器的 DMA 中断标志.....	324
19. IWDG (独立看门狗).....	325
19.1. 简介	325
19.2. 特性	325
19.3. 控制块	325
19.4. 使能和时钟	326
19.4.1. IWDG 全局使能.....	326
19.4.2. IWDG 时钟控制.....	326
19.5. 中断和事件	326
19.5.1. IWDG 中断控制和状态	326
19.5.2. IWDG 中断标志.....	326
19.6. 寄存器保护和锁定.....	327
19.7. IWDG 功能控制.....	327
19.7.1. IWDG 定时器控制	327
19.7.2. STOP 模式下工作	328
19.7.3. STOP 模式下唤醒	328
19.7.4. IWDG 事件时序.....	328
20. WWDG (窗口看门狗定时器)	329
20.1. 简介	329
20.2. 特性	329
20.3. 控制块	329
20.4. 使能和时钟	330
20.4.1. WWDG 全局使能	330
20.4.2. WWDG 时钟控制	330
20.5. 中断和事件	330
20.5.1. WWDG 中断控制和状态	330
20.5.2. WWDG 中断标志	330
20.6. WWDG 寄存器保护	331
20.7. WWDG 功能控制	331
20.7.1. WWDG 定时器控制.....	331
20.7.2. WWDG 模块复位控制	331
20.7.3. WWDG 窗口时序	332

21. RTC (实时时钟)	333
21.1. 简介	333
21.2. 特性	333
21.3. 控制块	333
21.4. IO 线	334
21.4.1. IO 信号	334
21.4.2. IO 设置	334
21.5. 使能和时钟	334
21.5.1. RTC 全局使能	334
21.5.2. RTC 时钟控制	334
21.6. 中断和事件	335
21.6.1. RTC 中断控制和状态	335
21.6.2. RTC 中断标志	335
21.7. 寄存器保护和锁定	335
21.8. RTC 功能控制	335
21.8.1. RTC 报警	335
21.8.2. RTC 时间戳	335
21.8.3. RTC 定时器捕获和重载	336
21.8.4. RTC 输出	336
21.8.5. STOP 模式工作	336
21.8.6. STOP 模式下唤醒	336
22. ADC (模数转换器)	337
22.1. 简介	337
22.2. 特性	337
22.3. 配置	338
22.3.1. 芯片配置	338
22.3.2. 模块功能	338
22.4. 控制块	339
22.5. IO 线	339
22.5.1. IO 信号	339
22.5.2. IO 设置	339
22.6. 电源和时钟	340
22.6.1. ADC 电源控制	340
22.6.2. ADC 时钟控制	340
22.7. 中断和事件	341
22.7.1. ADC 中断和复位事件	341
22.7.2. ADC 中断标志	341
22.8. ADC 操作	343
22.8.1. ADC 输入信号	343
22.8.2. 单端和差分模式	345
22.8.3. 输入动态电压范围和码范围	345
22.8.4. ADC 增益控制	345
22.8.5. ADC 转换	346
22.8.6. ADC 校准	347
22.8.7. SLEEP 模式下 ADC 工作	347
22.9. ADC 转换序列	348
22.9.1. ADC 转换设置和序列	348
22.9.2. ADC 转换暂停	349
22.9.3. ADC 转换过载	349

22.10. ADC 转换模式	349
22.10.1. 单次和持续转换	349
22.10.2. 通道扫描转换	351
22.10.3. 扫描循环转换	352
22.10.4. 输入通道改变	352
22.10.5. ADC 转换流	353
22.11. ADC 输出控制	354
22.11.1. 数字偏移调整器	354
22.11.2. 符号码转换器	355
22.11.3. 分辨率和数据对齐	356
22.11.4. ADC 数据寄存器	357
22.11.5. 电压窗口检测	357
22.11.6. 输出码限制	359
22.12. ADC 数据和	360
22.12.1. ADC 数据累加	360
22.12.2. ADC 数据和寄存器	361
22.12.3. ADC 数据和过载	361
22.12.4. ADC 数据和转换时序	362
22.13. ADC 等待和自动关闭	364
22.14. ADC DMA 操作	364
22.14.1. DMA 模块设置	364
22.14.2. ADC DMA 控制	364
22.14.3. ADC 的 DMA 中断标志控制	365
22.15. ADC 应用电路	366
23. CMP (模拟比较器)	367
23.1. 简介	367
23.2. 特性	367
23.3. 配置	367
23.3.1. 芯片配置	367
23.4. 控制块	368
23.5. IO 线	369
23.5.1. IO 信号	369
23.5.2. IO 设置	369
23.6. 电源和时钟	369
23.6.1. CMP 电源控制	369
23.6.2. CMP 时钟控制	369
23.7. 中断和事件	369
23.7.1. CMP 中断控制	369
23.7.2. CMP 中断标志	370
23.8. CMP 模拟比较器	371
23.8.1. 输入通道	371
23.8.2. 内部电压参考	371
23.8.3. 模拟比较器-0	371
23.8.4. 模拟比较器-1/2/3	371
23.8.5. 响应时间选择	372
23.8.6. 比较器输出	372
23.9. CMP 内部电压参考	373
23.9.1. IVREF 和 IVREF2	373
23.9.2. IVREF 输出电压	373

23.10. CMP 输入迟滞电压.....	374
23.11. CMP 输入网络.....	375
24. DAC (数模转换器)	377
24.1. 简介	377
24.2. 特性	377
24.3. 配置	377
24.3.1. 芯片配置	377
24.4. 控制块	378
24.5. IO 线	378
24.5.1. IO 信号	378
24.5.2. IO 设置	378
24.6. 电源和时钟	378
24.7. 中断和事件	379
24.7.1. DAC 中断控制	379
24.7.2. DAC 中断标志	379
24.8. DAC 输出电压	380
24.9. DAC 转换.....	381
24.9.1. 数据寄存器写	381
24.9.2. DAC 事件触发	382
24.9.3. DAC 数据分辨率和对齐	382
24.10. DAC DMA 操作	383
24.10.1. DMA 模块设置	383
24.10.2. DAC DMA 控制	383
24.10.3. DAC 的 DMA 中断标志控制	383
24.11. DAC 应用电路	383
25. 版本历史	384

图表

图 1-1. 方框图词汇表	26
图 1-2. 电源工作模式指示	26
图 2-1. 芯片配置表	29
图 2-2. 芯片主体 – MG32F02A132/072	30
图 2-3. 芯片主体 – MG32F02A032	31
图 2-4. ARM Cortex-M0 处理器	32
图 2-5. CPU 内存映射	33
图 2-6. CPU 调试连接块图	39
图 3-1. 供电框图	42
图 3-2. 电源控制器	43
图 3-3. 电源电压检测	44
图 3-4. 电源控制器中断和复位事件	45
图 3-5. 掉电模式下的设备电源使能控制	47
图 3-6. STOP 唤醒时序	49
图 3-7. 唤醒和中断控制 – MG32F02A132/072	49
图 3-8. 唤醒和中断控制 – MG32F02A032	50
图 3-9. 内部电源 LDO 控制	51
图 3-10. 供电电路	51
图 4-1. 复位源控制器 – MG32F02A132/072	53
图 4-2. 复位源控制器 – MG32F02A032	54
图 4-3. 芯片复位时序	56
图 4-4. 外部复位时序	57
图 4-5. 模块软件复位	61
图 4-6. GPIO 复位控制 – MG32F02A132/072	62
图 4-7. GPIO 复位控制 – MG32F02A032	63
图 4-8. 复位电路	63
图 5-1. 时钟源控制块	65
图 5-2. 时钟源控制器中断	66
图 5-3. PLL 控制块	68
图 5-4. 模块进程时钟选择	70
图 5-5. 模块进程时钟控制 – STOP 模式下不被支持	71
图 5-6. 模块进程时钟控制 – STOP 模式下支持运行	71
图 5-7. 晶振电路	74
图 6-1. 系统中断控制	76
图 7-1. 内存控制块	78
图 7-2. 内存中断/复位事件控制	79
图 7-3. 闪存地址映射	81
图 7-4. 设置字节闪存映射和寄存器导入	85
图 7-5. 硬件设置字节装载时序	85
图 7-6. ICP/ISP/IAP 闪存访问限制	86
图 7-7. 内存访问错误管理	88
图 7-8. 不同型号的闪存大小和设置	88
图 9-1. GPIO 控制块	93
图 9-2. IO 模式控制块	94
图 9-3. IO 控制结构图	96
图 9-4. IO 结构–模拟 IO 和数字输入	97
图 9-5. IO 结构–推挽	97
图 9-6. IO 结构–开漏	98

图 9-7. IO 结构—准双向.....	99
图 9-8. IO 端口访问块.....	99
图 9-9. 引脚功能复用选择块.....	100
图 9-10. 特殊引脚的功能复用选择.....	101
图 10-1. 中断功能块.....	107
图 10-2. NVIC 异常控制块.....	108
图 10-3. NVIC 中断控制.....	109
图 10-4. WIC 控制块.....	110
图 10-5. 外部中断控制器.....	111
图 10-6. 外部端口中断块.....	113
图 11-1. 通用逻辑.....	115
图 11-2. GPL 工作时钟控制.....	116
图 11-3. 字节顺序大端/小端规则和位顺序改变图表.....	117
图 11-4. 位顺序改变图表.....	118
图 11-5. CRC 控制块.....	119
图 11-6. CRC 多项式.....	119
图 11-7. GPL DMA 控制块.....	120
图 12-1. DMA 控制块.....	122
图 12-2. DMA 工作时钟控制.....	123
图 12-3. DMA 中断控制.....	124
图 12-4. DMA 源和目的地控制.....	125
图 12-5. DMA 通道选择优先级.....	127
图 12-6. DMA SRAM 使用建议.....	129
图 12-7. 外围 DMA RX 请求和承认控制时序.....	131
图 12-8. 外围 DMA TX 请求和承认控制时序.....	132
图 12-9. DMA 外部请求触发输入时序.....	135
图 13-1. 外部内存总线控制器.....	137
图 13-2. EMB 工作时钟控制.....	138
图 13-3. EMB 中断控制.....	139
图 13-4. EMB 模块 IO 控制 - MCLK, MWE, MOE.....	140
图 13-5. EMB 模块 IO 控制 - MCE, MALE, MALE2.....	141
图 13-6. EMB 时钟和控制信号极性.....	141
图 13-7. EMB 模块 IO 控制 - MBW0, MBW1, MAD.....	142
图 13-8. EMB 内存映射.....	143
图 13-9. EMB AHB 到外部内存传输.....	144
图 13-10. EMB 访问控制时序状态.....	146
图 13-11. EMB 地址/数据接口 - 16bit MA + 16bit MD.....	148
图 13-12. EMB 地址/数据时序 - 16bit MA + 16bit MD.....	148
图 13-13. EMB 地址/数据接口 - 16bit MA + 16bit MAD.....	149
图 13-14. EMB 地址/数据时序 - 16bit MA + 16bit MAD.....	149
图 13-15. EMB 地址/数据接口 - 复用 16 位具有 2 个地址相位的 MAD.....	150
图 13-16. EMB 地址/数据时序 - 复用 16 位具有 2 个地址相位的 MAD.....	150
图 13-17. EMB SRAM 接口.....	151
图 13-18. EMB SRAM 16 位数据时序.....	152
图 13-19. EMB NOR-Flash 接口.....	153
图 13-20. EMB NOR-Flash 16 位数据时序.....	154
图 13-21. EMB NAND-Flash 接口.....	155
图 13-22. EMB NAND-Flash 16 位数据时序.....	156
图 13-23. EMB LCD 接口.....	157

图 13-24. EMB LCD 寄存器 16 位总线访问时序	157
图 13-25. EMB LCD GRAM 16 位总线访问时序	158
图 14-1. APB 中断控制	162
图 14-2. 定时器同步使能控制	163
图 14-3. 定时器一般触发/时钟源选择	164
图 14-4. 输出信号中止和调制控制	165
图 14-5. IR 控制接口	168
图 15-1. I2C 主控制块 – I2C0/1	172
图 15-2. I2C 工作时钟控制	173
图 15-3. I2C 状态和中断控制 – I2C0/1	174
图 15-4. I2C 子范围中断控制 – I2C0/1	175
图 15-5. I2C 单主机连接通信	177
图 15-6. I2C 多主机和从机模式连接通信	177
图 15-7. I2C 基本协议	178
图 15-8. I2C START/STOP/数据改变时序	179
图 15-9. I2C 数据缓冲模式控制块 – I2C0/I2C1	185
图 15-10. I2C 访问指令位	186
图 15-11. I2C 主机发送操作序列	187
图 15-12. I2C 主机接收操作序列	188
图 15-13. I2C 从机发送操作序列	189
图 15-14. I2C 从机接收操作序列	190
图 15-15. I2C 信号驱动控制时序	191
图 15-16. I2C 从机 SCL 拉伸	192
图 15-17. I2C TMO 超时定时器控制	193
图 15-18. I2C 总线错误事件	195
图 15-19. I2C 主机发送 NACK 检测	195
图 15-20. I2C SCL 拉伸禁用时从机数据溢出	196
图 16-1. UART 主控制块 – URT0/1/2/3	201
图 16-2. UART 工作时钟控制	202
图 16-3. UART 时钟控制 – MG32F02A132/072	203
图 16-4. UART 时钟控制 – MG32F02A032	204
图 16-5. UART 状态和中断控制 – URT0/1/2/3	205
图 16-6. UART 子范围中断控制 – URT0/1/2/3	206
图 16-7. UART RX/TX IO 控制 – MG32F02A132/072	208
图 16-8. UART RX/TX IO 控制 – MG32F02A032	208
图 16-9. UART 其他 IO 控制	209
图 16-10. UART UART/IrDA 模式连接	210
图 16-11. UART LIN 模式连接	210
图 16-12. UART 智能卡模式连接	211
图 16-13. UART SPI 主机模式连接	211
图 16-14. UART 数据格式	212
图 16-15. UART Idle 线和中止状态格式	214
图 16-16. UART 数据收发	217
图 16-17. UART 数据模式控制 – URT0/1/2/3	218
图 16-18. UART 多处理器工作模式	220
图 16-19. UART 多处理器地址设置示例	221
图 16-20. UART 多处理器从机地址发送	221
图 16-21. UART 中止状态发送	222
图 16-22. UART 波特率定时器控制块	223

图 16-23. UART 自动波特率控制模式时序	224
图 16-24. UART 接收数据采样	226
图 16-25. UART 接收过采样多数点	227
图 16-26. UART 接收位时间错误公差.....	228
图 16-27. UART 接收位时间错误公差示例	228
图 16-28. UART TMO 超时定时器控制块	229
图 16-29. UART 保持时间和超时检测.....	230
图 16-30. UART 自动校准超时错误检测	231
图 16-31. UART 中止和校验/帧错误检测	233
图 16-32. UART TX 错误检测和重传控制	234
图 16-33. UART RX 校验错误检测和重试控制.....	235
图 16-34. UART 静音模式控制时序	236
图 16-35. UART 同步模式时序	237
图 16-36. UART 同步模式硬件 NSS 时序	238
图 16-37. UART 智能卡时钟输出.....	239
图 16-38. UART IrDA 时序 – MG32F02A132/072.....	241
图 16-39. UART IrDA 时序 – MG32F02A032	242
图 16-40. UART 占用和 IrDA 占用状态.....	243
图 16-41. UART 驱动使能时序	244
图 16-42. UART 硬件流控制连接.....	245
图 16-43. UART CTS 硬件流控制时序.....	245
图 16-44. UART RTS 硬件流控制和 RX 溢出时序	246
图 16-45. UART 软件流控制.....	247
图 16-46. UART 接收硬件和捕获停止事件	248
图 17-1. SPI 主控制块	252
图 17-2. SPI 工作时钟控制	253
图 17-3. SPI 状态和中断.....	254
图 17-4. SPI 数据 IO 控制.....	256
图 17-5. SPI 时钟/NSS IO 控制 – MG32F02A132/072	257
图 17-6. SPI 时钟/NSS IO 控制 – MG32F02A032	257
图 17-7. SPI 全双工通信.....	259
图 17-8. SPI 单工通信	259
图 17-9. SPI 半双工通信.....	260
图 17-10. 八数据线半双工通信	261
图 17-11. SPI 多从机通信	261
图 17-12. SPI 双主机通信.....	262
图 17-13. SPI 主机和从机调换通信	262
图 17-14. SPI 主机时钟输出	263
图 17-15. SPI 带 NSS 的基本时序	265
图 17-16. SPI 不带 NSS 的基本时序	266
图 17-17. SPI 主机 DTR 模式时序	267
图 17-18. SPI 主机模式硬件 NSS 时序	269
图 17-19. SPI 数据缓冲模式控制 – SPI0.....	270
图 17-20. SPI 发送数据帧大小控制	271
图 17-21. SPI 数据模式-分离的 I/O (SPI).....	274
图 17-22. SPI 数据模式- 1 双向数据线	274
图 17-23. SPI 数据模式– 2 分离数据的双向数据线.....	275
图 17-24. SPI 数据模式– 2 相同数据的双向数据线.....	275
图 17-25. SPI 数据模式– 4 分离数据的双向数据线.....	276

图 17-26. SPI 数据模式– 4 相同数据的双向数据线.....	276
图 17-27. SPI 数据模式– 8 分离数据的双向数据线.....	277
图 17-28. SPI 接收溢出	279
图 17-29. SPI 从机模式发送错误	279
图 17-30. SPI 主机模式改变检测	280
图 18-1. 定时器主块 ~ TM0x/TM1x.....	284
图 18-2. 定时器主块 ~ TM2x.....	285
图 18-3. 定时器主块 ~ TM3x.....	286
图 18-4. 定时器工作时钟控制	288
图 18-5. 定时器时钟和触发源	289
图 18-6. 定时器事件控制和定时器标志	290
图 18-7. 定时器状态和中断控制 – TM0x/1x	291
图 18-8. 定时器状态和中断控制 – TM2x/3x	291
图 18-9. 定时器工作模式控制 – 级联模式	293
图 18-10. 定时器工作模式控制 – 分离模式	293
图 18-11. 定时器工作模式控制 – 全计数模式	294
图 18-12. 定时器触发输入控制块.....	295
图 18-13. 定时器时钟和触发控制.....	295
图 18-14. 定时器触发输出控制块.....	297
图 18-15. 定时器输入/输出通道 ~ TM3x.....	298
图 18-16. 定时器输入/输出通道 ~ TM2x.....	299
图 18-17. 定时器时钟输出块.....	300
图 18-18. 定时器 CKO 时钟输出时序.....	301
图 18-19. 定时器输入捕获和输出比较块	302
图 18-20. 定时器输入捕获块.....	303
图 18-21. 全计数模式定时器输入捕获时序 – MG32F02A132/072.....	305
图 18-22. 全计数模式定时器输入捕获时序 – MG32F02A032	306
图 18-23. 级联和分离模式定时器输入捕获时序	307
图 18-24. 定时器 16 位输出比较/PWM 块.....	308
图 18-25. 定时器双 8 位比较/PWM 输出块	309
图 18-26. 定时器向上计数输出比较时序.....	310
图 18-27. 定时器向上计数边沿对齐 PWM 时序.....	311
图 18-28. 定时器向下计数边沿对齐 PWM 时序.....	312
图 18-29. 定时器中心对齐 PWM 时序	313
图 18-30. 定时器 PWM 死区控制时序.....	314
图 18-31. 定时器输出控制块 ~ TM2x	315
图 18-32. 定时器输出控制块 ~ TM3x	316
图 18-33. 定时器 OC 输出使能预载控制	317
图 18-34. 定时器中止输入源	318
图 18-35. 定时器中止事件控制时序	319
图 18-36. 定时器 QEI 控制块	320
图 18-37. 定时器 QEI 控制模式 1,2.....	321
图 18-38. 定时器 EXUD 控制模式 3,4,5	323
图 19-1. IWDG 控制块	325
图 19-2. IWDG 工作时钟控制.....	326
图 19-3. IWDG 事件时序	328
图 20-1. WWDG 系统窗口看门狗定时器	329
图 20-2. WWDG 工作时钟控制	330
图 20-3. WWDG 窗口时序.....	332

图 21-1. RTC 实时时钟控制.....	333
图 21-2. RTC 工作时钟控制.....	334
图 22-1. ADC 块图.....	339
图 22-2. ADC 工作时钟控制.....	340
图 22-3. ADC 输入时钟.....	340
图 22-4. ADC 中断控制.....	342
图 22-5. ADC 和模拟输入多路复用器.....	343
图 22-6. ADC 从 DAC 输出输入.....	344
图 22-7. ADC 动态电压范围.....	345
图 22-8. ADC PGA 控制.....	346
图 22-9. ADC 转换序列.....	347
图 22-10. ADC 转换启动控制.....	348
图 22-11. ADC 转换模式-单通道.....	350
图 22-12. ADC 转换模式-通道扫描.....	351
图 22-13. ADC 转换模式 - 循环扫描.....	352
图 22-14. ADC 转换流.....	353
图 22-15. ADC 输出控制块.....	354
图 22-16. ADC 电压窗口检测.....	358
图 22-17. ADC 输出码限制.....	359
图 22-18. ADC 数据和模式-单次或间断通道扫描.....	362
图 22-19. ADC 数据和模式-通道/循环扫描.....	363
图 22-20. ADC 等待和自动关闭模式时序.....	364
图 22-21. ADC 应用电路.....	366
图 23-1. CMP 主块.....	368
图 23-2. CMP 工作时钟控制.....	369
图 23-3. CMP 中断控制.....	370
图 23-4. CMP 模拟比较器 CMP0 块.....	371
图 23-5. CMP 模拟比较器 CMP1/2/3 块.....	372
图 23-6. CMP 比较器 IVREF 阶梯块.....	373
图 23-7. CMP 比较器迟滞电压.....	374
图 23-8. CMP 输入网络示例.....	375
图 24-1. DAC 主块.....	378
图 24-2. DAC 工作时钟控制.....	379
图 24-3. DAC 中断控制.....	379
图 24-4. DAC 输出电压.....	380
图 24-5. DAC 转换 - 输出通过写数据寄存器更新.....	381
图 24-6. DAC 转换 - 输出通过事件触发更新.....	382
图 24-7. DAC 应用电路.....	383

表单

表 1-1. 词汇表.....	27
表 2-1. CPU 内存地址映射	34
表 2-2. 外围内存边界地址 – MG32F02A132/072.....	35
表 2-3. 外围内存边界地址 – MG32F02A032.....	36
表 3-1. 电源工作模式.....	41
表 3-2. 电源模式选择.....	46
表 3-3. 内部设备电源控制.....	46
表 3-4. 唤醒事件源.....	48
表 4-1. 不同复位等级的复位控制功能.....	55
表 4-2. 复位事件源.....	58
表 4-3. 模块寄存器复位对比锁定功能.....	60
表 5-1. 模块可用时钟使能寄存器.....	72
表 5-2. 内部设备时钟控制.....	72
表 5-3. 模块可用输入事件时钟源表.....	73
表 5-4. C1 & C2 晶振电路参考电容	74
表 7-1. 内存启动方式选择.....	80
表 7-2. 闪存访问控制和键值	83
表 7-3. 闪存访问限制对比启动模式.....	86
表 7-4. 闪存访问下 CPU 暂停控制.....	87
表 9-1. IO 工作模式控制	95
表 9-2. 端口 C IO 模式默认设置.....	102
表 10-1. 中断源.....	104
表 10-2. CPU 锁定退出事件	106
表 10-3. Cortex-M0 上的 Hard Fault 处理事件.....	110
表 12-1. DMA 配置	121
表 12-2. DMA 通道源/目的地请求选择.....	126
表 12-3. DMA 通道工作类型支持.....	126
表 12-4. DMA 传输数量/起始地址和单位大小设置注释.....	128
表 12-5. DMA 功能的外围模块中断标志控制 – MG32F02A132/072	133
表 12-6. DMA 功能的外围模块中断标志控制 – MG32F02A032.....	134
表 13-1. EMB 配置	136
表 13-2. EMB 接口和设备引脚映射	142
表 13-3. EMB AHB 到外部内存传输支持	144
表 13-4. EMB 地址/数据接口模式设置.....	147
表 13-5. EMB 接口信号和建议引脚表.....	159
表 14-1. APB 配置	161
表 14-2. 定时器一般 ITR6/ITR7 信号表格.....	164
表 14-3. OBM 块输出通道信号表.....	166
表 14-4. OBM 块中止通道信号表.....	167
表 14-5. IR 时钟/数据信号表.....	169
表 15-1. I2C 配置.....	171
表 15-2. I2C 模块功能	171
表 15-3. I2C 事件码表	180
表 15-4. I2C 主机发送模式事件表.....	181
表 15-5. I2C 主机接收模式事件表.....	181
表 15-6. I2C 从机发送模式事件表.....	182
表 15-7. I2C 从机接收模式事件表.....	183
表 15-8. I2C 杂项事件表.....	184

表 15-9. I2C 信号输出驱动设定	192
表 15-10. I2C TMO 超时定时器启动/关闭控制	194
表 15-11. I2C DMA 功能中断标志控制	197
表 16-1. UART 配置	199
表 16-2. UART 模块功能	200
表 16-3. UART 数据格式设置	212
表 16-4. UART 校验位定义	213
表 16-5. UART 控制模式对比数据控制寄存器	215
表 16-6. UART 工作模式设置	215
表 16-7. UART 工作模式设置	216
表 16-8. UART 中止状态发送和检测控制	222
表 16-9. UART BR 波特率定时器启动/关闭控制	223
表 16-10. UART 通用波特率设置示例	225
表 16-11. UART 接收数据过采样和噪音检测	226
表 16-12. UART TMO 超时定时器开启/关闭控制	230
表 16-13. UART 校准超时状态	231
表 16-14. UART 校准超时时间	232
表 16-15. UART 多处理器地址匹配对比静音模式控制	236
表 16-16. UART 同步时钟模式表	237
表 16-17. UART 同步模式 NSS 时序表	238
表 16-18. UART 智能卡时钟输出设置示例	240
表 16-19. UART IrDA 接收数据过采样和采样模式	242
表 16-20. 多处理器地址不匹配时 UART 数据缓冲对比 RHF 状态	248
表 16-21. UART DMA 功能中断标志控制	249
表 17-1. SPI 配置	251
表 17-2. SPI 模块功能	251
表 17-3. SPI 时钟模式表	263
表 17-4. SPI NSS 控制表	267
表 17-5. SPI 主机 NSS 时序表	268
表 17-6. SPI 数据控制表	273
表 17-7. SPI Flash 类型表	278
表 17-8. SPI Flash 控制表	278
表 17-9. SPI DMA 功能中断标志控制	281
表 18-1. 定时器配置	283
表 18-2. 定时器模块功能	283
表 18-3. 定时器工作模式和控制验证	294
表 18-4. 定时器内部触发信号表 – MG32F02A132/072	296
表 18-5. 定时器内部触发信号表 – MG32F02A032	296
表 18-6. 定时器通道输入信号表	299
表 18-7. 定时器捕获和比较寄存器控制	303
表 18-8. 定时器输入捕获模式 – MG32F02A132/072	304
表 18-9. 定时器输入捕获模式 – MG32F02A032	304
表 18-10. 定时器输出比较/PWM 模式	309
表 18-11. 定时器输出中止或停止控制	318
表 18-12. 定时器 QEI 外部输入向上/向下控制模式	320
表 18-13. 定时器 QEI 编码器状态	322
表 18-14. 定时器 QEI 编码器状态转换	322
表 18-15. 定时器 DMA 功能中断标志控制	324
表 19-1. IWDG 寄存器写保护表	327

表 22-1. ADC 配置	338
表 22-2. ADC 模块功能	338
表 22-3. ADC 通道定义	344
表 22-4. ADC 转换模式控制	349
表 22-5. ADC 数字偏移操作	354
表 22-6. ADC 单端模式数据格式定义	355
表 22-7. ADC 差分模式数据格式定义	355
表 22-8. ADC 数据对齐定义	356
表 22-9. ADC 数据码示例	357
表 22-10. ADC 数据和以及转换模式控制	360
表 22-11. ADC 数据和标志对比转换模式	360
表 22-12. ADC 数据和值范围	361
表 22-13. ADC DMA 功能中断标志控制 – MG32F02A132/072	365
表 22-14. ADC DMA 功能中断标志控制 – MG32F02A032	365
表 23-1. CMP 配置	367
表 23-2. CMP 比较器 IVREF R-阶梯输出	373
表 24-1. DAC 配置	377
表 24-2. DAC 数据对齐定义	382
表 24-3. DAC DMA 功能中断标志控制	383

1. 文档用法









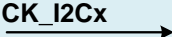

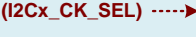





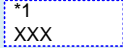




1.1. 文档规则

以下的文档范例用于指示本文档中的特殊含义。

- PA[15:0]** : 引脚名 以暗绿色表明
- DAC_TRG0** : GPIO 引脚复用 IO 名 以橙色表明
- I2Cx_EN** : 寄存器名 以暗红色表明
- CK_HS** : 内部信号 以蓝色表明
- [Notify] : “[Notify]”后面以斜体和下划线表明 “注释” 用以提醒用户

1.2. 方框图词汇表

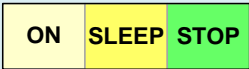
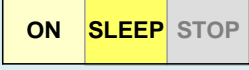
图 1-1. 方框图词汇表

	Chip Pin and Pin Name		Voltage, Current Reference Source
	AFS IO and IO Name		Analog, Digital Multiplex
	Module IO and IO Name		On/Off Switch - default Enable
	Module External Signal		On/Off Switch - default Disable
	Module Internal Signal		Digital Comparator
	Register Name and Control Indicated Line		Clock Divider
	Rising, Falling Edge Signals		Signal Inverter/Polarity Control
	Pulse, Level Signal		Signal Toggler
	<Note-1> and description		Flag set (blue) and clear (red)
	See <Note-1>, <Note-2> below block diagram for more information		Branch Item
<p><Note> : General descriptions to notify for this diagram. <Note-1> : Descriptions to notify for *1 in the diagram. <Note-2> : Descriptions to notify for *2 in the diagram.</p>			Step or Separated Item

1.3. 电源工作模式指示

以下图表展示了模块的电源工作模式。

图 1-2. 电源工作模式指示

Power Operation Modes		Indicator	
① ON	CPU is able running in full speed.		The module can be running in all power operation modes.
② SLEEP	CPU is stopped and all peripheral modules can be configurable to continue running.		The module can be running in ON and SLEEP modes only.
③ STOP	CPU and all peripheral modules are stopped except analog comparator, RTC and IWDG modules can be configurable to continue running.		

1.4. 词汇表

该表格提供了用于该文档的缩写词的缩写。

表 1-1. 词汇表

项目	描述
ON	ON 模式 ~ 芯片处于普通工作模式
SLEEP	SLEEP 模式 ~ 芯片处于睡眠模式, CPU 进入睡眠
STOP	STOP 模式 ~ 芯片处于停止模式, CPU 进入深度睡眠, AHB/APB 时钟停止
Byte	8 位数据宽度
Half-Word	16 位数据宽度
Word	32 位数据宽度
Module	用特定方法控制的外设模块
ADC	模数转换器
AHB	高级高性能总线
APB	高级外设总线
BOD	掉电检测
CAN	控制器局域网
CCP, IC/OC/PWM	输入捕获 / 输出比较 / 脉冲宽度调制
CMP / ACMP	模拟比较器
CRC	循环冗余校验
DAC	数模转换器
DAP	ARM Cortex-Mx 调试访问端口
DMA	直接内存访问
EXTCK	外部输入时钟
GPIO	通用输入/输出
I2C	集成电路总线
IAP	在应用编程 (为用户编程代码服务)
ICP	在电路编程 (为 SWD/JTAG 接口服务)
ISP	在系统编程 (为启动引导代码服务)
IHRCO	内部高速 RC 振荡器
ILRCO	内部低速 RC 振荡器
IrDA	红外数据组织
KBI	键盘中断
LDO	低压降线性稳压器
LIN	本地内部连接网络
MCD	时钟丢失检测器
NVIC	嵌套中断向量控制器
OB	硬件选项字节内存
OR	硬件选项设置寄存器
PLL	锁相环
SMBus	系统管理总线 (I2C 协议)
SPI	串行外设接口
SWD	串口线调试
WIC	唤醒中断控制器
XOSC	为外部晶振电路服务的内嵌晶体振荡器

2. 芯片与系统

2.1. 芯片概述

MG32x02z 是基于拥有嵌套中断向量控制器（NVIC）的高性能 ARM 32 位 Cortex™-M0 单片机。

MG32x02z 系列有最多 132K 字节的内置 Flash 存储器用于存储代码和数据、设置用于保存启动码和用于芯片配置的 64 字节闪存。整个 Flash 空间均可通过串行烧写模式（ICP，在电路编程），主存也可在 ISP 模式、SRAM（在 SRAM 启动）模式进行编程。ICP 和 ISP 让用户无需从产品中去下微控制器就可以下载新的代码；IAP 意味着应用程序正在运行时，微控制器能够在 Flash 中写入非易失数据。由于内建有电荷泵电路，因此不需要外部高电压进行编程。

MG32x02z 系列包含了 ARM 32 位 Cortex™-M0 的所有特性，包括 16K 字节的 SRAM、5 个 I/O 端口，32 条 4 级外部中断源控制器和 7 个 8/16 位定时器/计数器。此外，**MG32x02z** 系列还有 1 个系统滴嗒定时器、2 个看门狗定时器，3 个拥有 IC/OC/PWM 的高性能定时器、4 个通用定时器用于通用用途、1 个用于支持 32.768KHz 到 25MHz 的片上晶体振荡器、2 个高精度内部振荡器分别是 11.059/12MHz 的 IHRCO 和 32KHz 的 ILRCO、1 个 12 位 ADC、4 个可编程阈值比较器、1 个 10 位电流型 DAC。

此外，**MG32x02z** 系列支持多种灵活的通讯接口。它提供了包括 GPIO、I2C、SPI、KBI、UART、SmartCard、LIN 和 SWD（片上调试）的功能复用。它有最多 73 个 GPIO 引脚，并提供有：准双向、推挽输出、开漏输出、可选上拉的仅输入（Hi-z）。另外，它内建了内部去抖电路用于过滤恶劣信号的噪音。

1 个直接内存访问控制器（DMA）用于改善外设与内存、内存与内存的数据传输体验。通过 DMA，数据可以直接进行传输，并不消耗 CPU 时间。

1 个外部内存总线（EMB）用于访问外部 SRAM、NOR/NAND 闪存或 LCD 显示面板。它支持多种地址总线和数据总线复用模式，它还支持用于外部设备的同步或者异步的编程时序。

对于电源管理和复位控制，**MG32x02z** 系列内置 1 个包含了 1 个低压检测器（LVD），2 个掉电检测器（BOD0/BOD1），1 个上电复位（POR），1 个低压复位（LVR）的电源监视器。**MG32x02z** 系列含有多种掉电模式用于降低功耗：Sleep 模式和 Stop 模式。

在 Sleep 模式中，CPU 会被冻结，而外设和中断系统仍处于工作状态；在 Stop 模式中，RAM 和 SFR 的值会被保存，但是其他的功能将被停止，最重要的是，在 Sleep 模式下，芯片可以被很多中断或者复位源（POR/LVR/BOD0/BOD1）唤醒。

2.2. 适用芯片

该用户手册适用于以下芯片。

- MG32F02A132
- MG32F02A072
- MG32F02A032

2.2.1. 芯片配置概览

下表展示了芯片的模块和包含的功能。

图 2-1. 芯片配置表

功能/模块	芯片			注释
	MG32F02A132	MG32F02A072	MG32F02A032	
CPU 核心	ARM Cortex M0	ARM Cortex M0	ARM Cortex M0	带有 1 个 NVIC 和 1 个单周期 32 位乘法器
闪存 ROM	132KB	72KB	32KB	AP+IAP+ISP 的总存储空间
SRAM	16KB	8KB	4KB	2KB(用于 DMA), 14KB(用于软件) ~ 16KB 纯 SRAM
最大 CPU 频率	48MHz	48MHz	48MHz	
内部时钟源	ILRCO+IHRCO	ILRCO+IHRCO	ILRCO+IHRCO	IHRCO 可选 12MHz(默认) 或 11.059MHz
电压检测器	LVR+BOD0/1	LVR+BOD0/1	LVR+BOD0/1	
封装	LQFP80/64	LQFP64/48	LQFP48 QFN32 TSSOP20	
IO 数量	73/59	59/44	44/29/17	
最多外部中断	63	59	44	
定时器	16 位*2 + 32 位*5	16 位*2 + 32 位*5	16 位*2 + 32 位*3	支持 Full-Counter, Cascade, Separate 模式
IC/OC/PWM	8-CH	8-CH	4-CH	
互补 PWM	7-CH	7-CH	3-CH	
WDT	IWDT + WWDT	IWDT + WWDT	IWDT + WWDT	
RTC	yes	yes	yes	
ADC	12 位, 16-CH 400Ksps	12 位, 16-CH 400Ksps	12 位, 12-CH 800Ksps	内嵌 1 个带 PGA 的输入缓冲
模拟比较器	4	4	2	内嵌 2 个 R-阶梯参考电压
DAC	10-Bit, 1-CH	10-Bit, 1-CH	-	电流模式 DAC
UART 单元	4	4	2	支持 SPI 主机模式,多处理器,IrDa,LIN,ISO-7816 (智能卡),硬件流控制
SPI 单元	1	1	1	支持 1/2/4/8 数据线模式
I2C 单元	2	2	1	可选择 Byte/Buffer 模式
DMA 通道	3-CH	3-CH	1-CH	内存到内存, 内存到外围设备, 外围设备到内存, 外围设备到外围设备
DMA 内存源	SRAM, EMB	SRAM, EMB	SRAM, Flash	EMB :通过 EMB 接口访问外存设备
EMB	16 位	16 位	-	支持 SRAM,NOR/NAND 闪存,8088 LCD IF
CRC	CRC8+16+32	CRC8+16+32	CRC8+16+32	CRC8/CRC16/CCITT16/CRC32 固定多项式
OBI 单元	1	1	2	

2.3. 芯片主体

以下图表展示了芯片内部的设备块。

该嵌入式 ARM Cortex-M0 处理器, 具有 NVIC (嵌套中断向量控制器) 和 DAP (调试访问端口); AHB lite 总线用于 SRAM/Flash 闪存, 电源/复位/时钟系统控制器, GPIO 控制块和 GPL (通用逻辑); 使用 APB 总线的 UART/SPI/I2C 通讯控制器, 定时器包括通用定时器/ IWDG / WWDG / RTC 和模拟控制块 ADC /模拟比较器/ DAC; 模拟设备包括 POR (上电复位), BOD0/BOD1 (掉电检测器), ILRGO (内部低频 RC 振荡器)/IHRGO 内部高频 RC 振荡器/PLL。

图 2-2. 芯片主体 - MG32F02A132/072

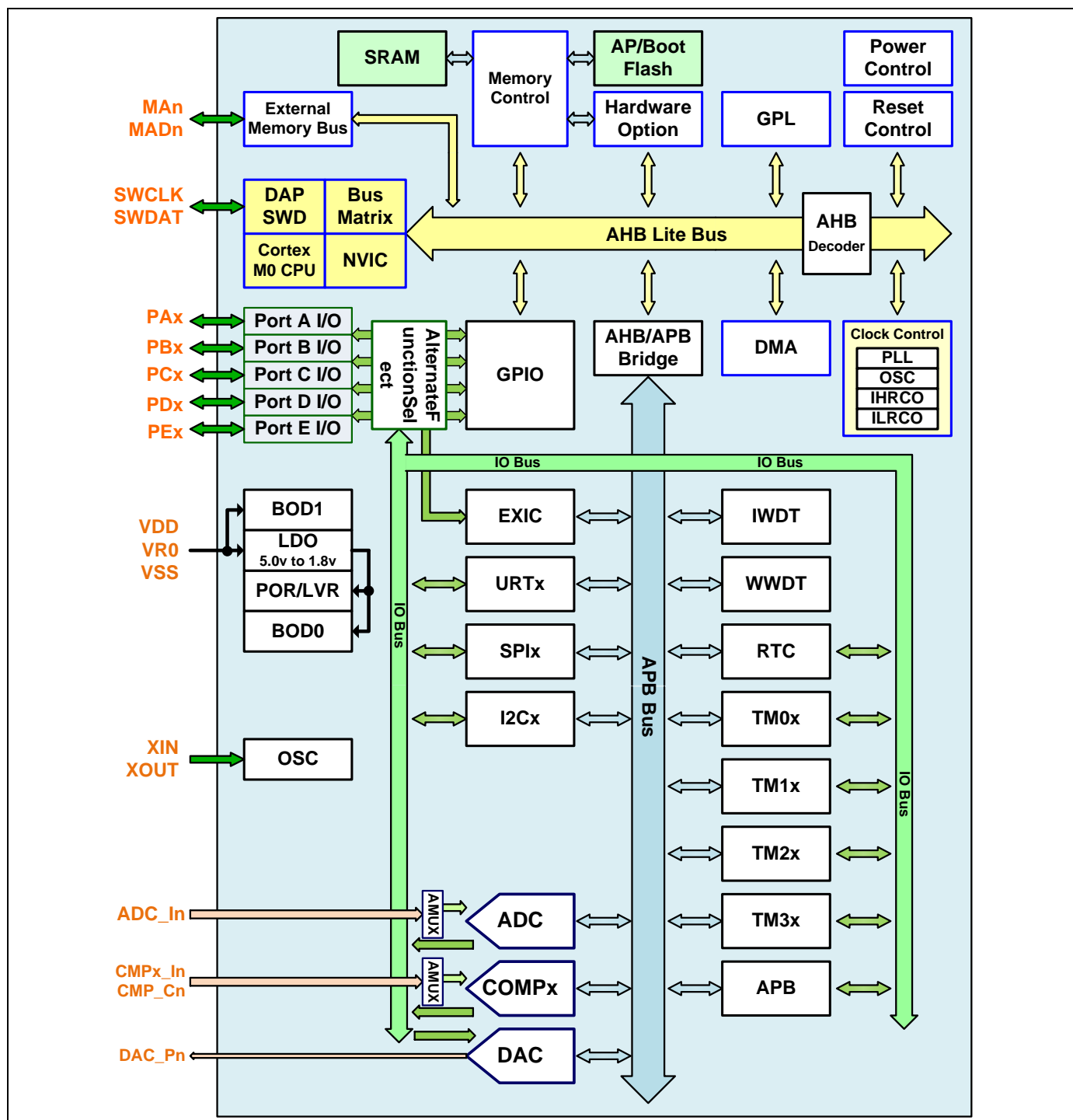
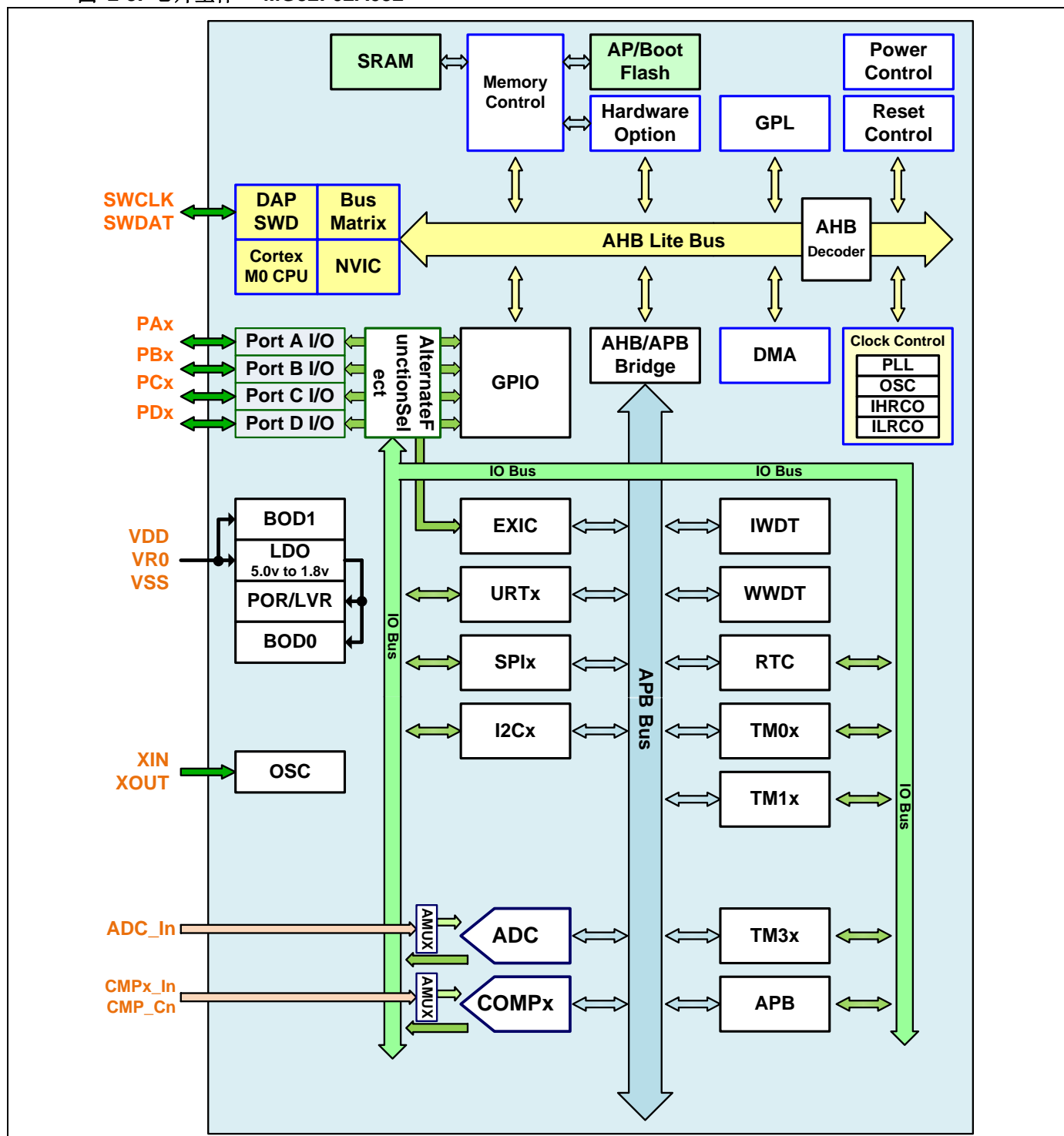


图 2-3. 芯片主体 – MG32F02A032



2.4. CPU 核心

该芯片内嵌了 Cortex™-M0 核心处理器。该处理器是可设置、多平台、32 位 RISC 处理器。它含有 1 个 AMBA AHB-Lite 接口和 1 个 NVIC 组件，还包含可设置 DAP 的硬件调试功能。

该芯片可执行 Thumb 指令，并可兼容于其他 Cortex-M 系列处理器，该系列支持两种模式：Thread 模式和 Handler 模式。Handler 模式会在异常发生时进入，一个异常的返回只能在 Handler 模式发出。Thread 模式在复位时进入，也可以在异常发生时进入。

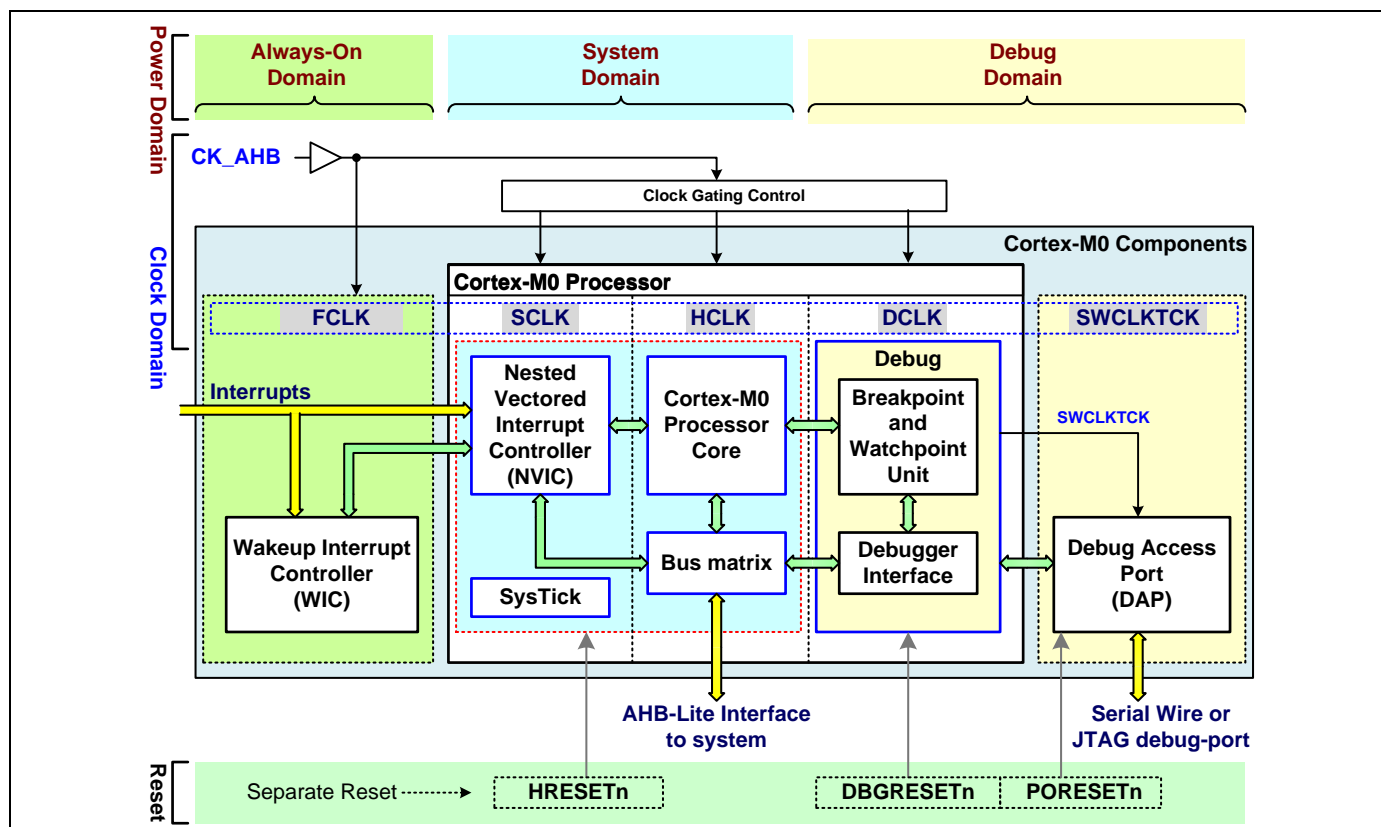
2.4.1. CPU 特性

- ARM 32 位 Cortex-M0 CPU
- 工作频率最高 48MHz
- 内置 1 个有 32 条 4 级优先级的外部中断输入的 NVIC
- 内置 1 个 24 位系统滴嗒定时器
- 内置 1 个单周期 32 位乘法器
- 内置 1 个含有 2 个监视点和 4 个断点的 SWD 串行线调试器
- ARMv6-M Thumb®指令集

2.4.2. ARM Cortex-M0 处理器

下面的图表展示了 ARM Cortex-M0 处理器的块。

图 2-4. ARM Cortex-M0 处理器



2.5. 存储器组织

芯片内建最大 **16K** 字节的 SRAM，有着最多 **132K** 字节的存储空间用于存储代码和数据、设置用于保存启动码和用于芯片配置的 **64** 字节闪存（**OB**）。另外，还有许多模块独立的硬件控制寄存器，并且位于 AHB/APB 设备的存储空间中。请根据芯片的数据手册以获取实际的内嵌 SRAM 和内存大小。

用户可以为自己的程序代码（AP）、系统编程代码（ISP）、在应用编程（IAP）配置整个存储器，用户可以控制这三个存储空间的大小。

2.5.1. CPU 内存映射

下面的图表展示了 CPU 总共 4G 字节地址空间的内存映射，CPU 内存空间被分成了每块 512M 字节的 8 块内存块，这些不能执行代码的块会被用“XN”标记。

图 2-5. CPU 内存映射

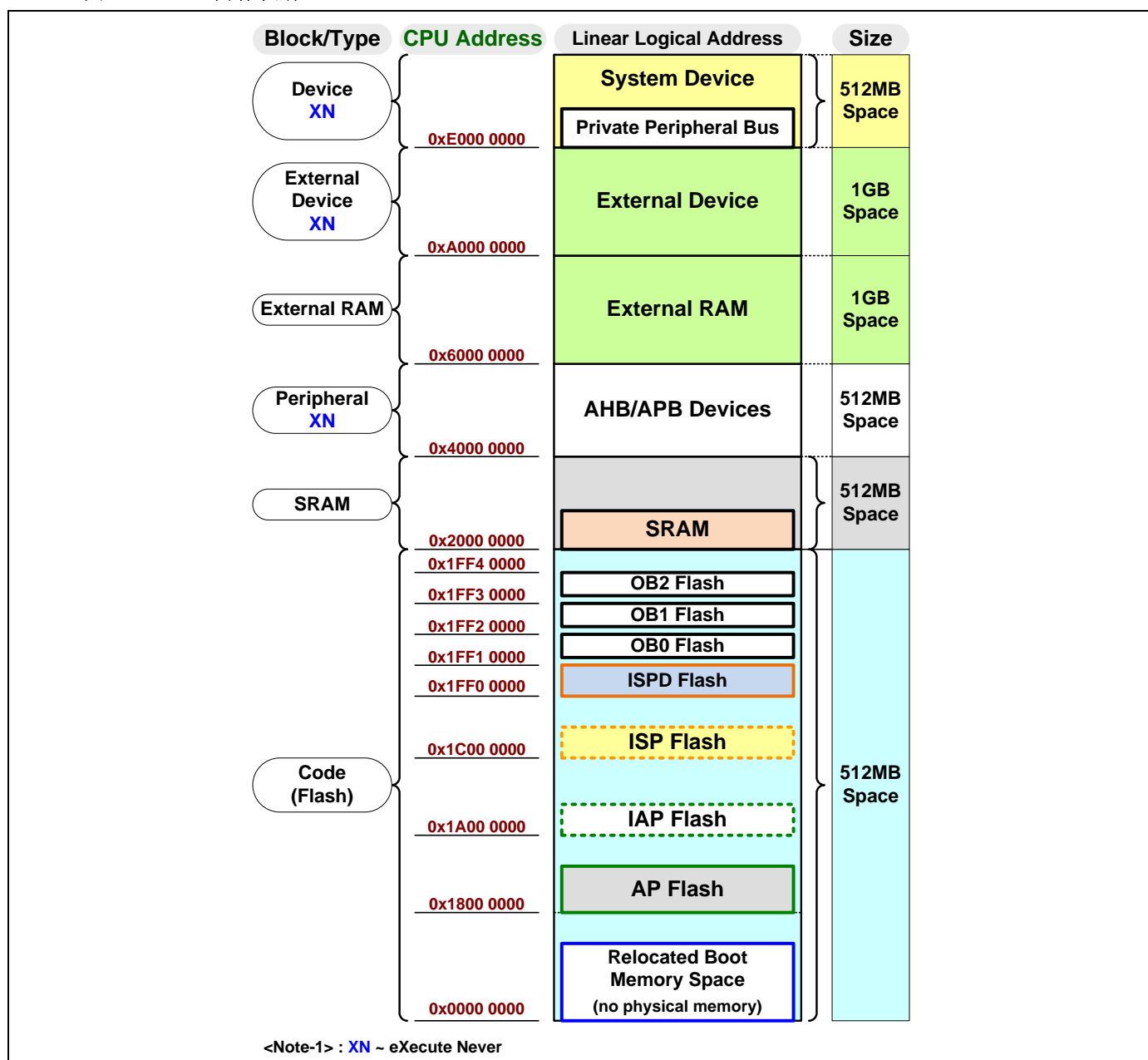


表 2-1. CPU 内存地址映射

块索引	块名称	XN	边界地址		大小	地址空间	注释
			起始地址	结束地址			
7	系统设备	XN	0xE010 0000	0xFFFF FFFF	511MB	VENDOR_SYS	
			0xE000 0000	0xE00F FFFF	1MB	专用外围总线(PPB)	M0 保留的 Cortex M0 内部外围设备
6	外部设备	XN	0xC000 0000	0xDFFF FFFF	512MB	保留	外部内存 (SRAM, Flash)
5	外部设备	XN	0xA000 0000	0xBFFF FFFF	512MB	保留	外部内存(SRAM, Flash)
4	外部 RAM		0x8000 0000	0x9FFF FFFF	512MB	保留	外部内存(SRAM, Flash)
3	外部 RAM		0x6000 0000	0x7FFF FFFF	512MB	保留	外部内存(SRAM, Flash)
2	外围设备	XN	0x4000 0000	0x5FFF FFFF	512MB	APB/AHB	APB/AHB 模块
1			0x2000 4000	0x3FFF FFFF	512MB	保留	
			0x2000 3800	0x2000 3FFF	2KB	SRAM	上部的 2K 字节用于 DMA
			0x2000 0000	0x2000 37FF	14KB		
0	代码		0x1FF4 0000	0x1FFF FFFF	768KB	保留	
			0x1FF3 0400	0x1FF3 FFFF	63KB	保留	
			0x1FF3 0040	0x1FF3 03FF	960B	OB Flash-2	
			0x1FF3 0000	0x1FF3 003F	64B		硬件设置字节-2 (64 字节)
			0x1FF2 0400	0x1FF2 FFFF	63KB	保留	
			0x1FF2 0050	0x1FF2 03FF	944B	OB Flash-1	
			0x1FF2 0040	0x1FF2 004F	16B		随机 ID (16 字节)
			0x1FF2 0000	0x1FF2 003F	64B		硬件设置字节-1 (64 字节)
			0x1FF1 0400	0x1FF1 FFFF	63KB	保留	
			0x1FF1 0040	0x1FF1 03FF	960B	OB Flash-0	
			0x1FF1 0000	0x1FF1 003F	64B		硬件设置字节-0 (64 字节)
			0x1FF0 0400	0x1FF0 FFFF	63KB	保留	
			0x1FF0 0000	0x1FF0 03FF	1KB	ISPD Flash	ISP 数据内存
			0x1C02 0000	0x1FEF FFFF	63MB	保留	
			0x1C00 0000	0x1C01 FFFF	128KB	ISP Flash	引导闪存 (可设置大小)
			0x1A02 0000	0x1BFF FFFF	32MB	保留	
			0x1A00 0000	0x1A01 FFFF	128KB	IAP Flash	数据闪存(可设置大小)
			0x1802 0000	0x19FF FFFF	32MB	保留	
			0x1800 0000	0x1801 FFFF	128KB	AP Flash	应用闪存 (可根据芯片配置设置大小)
			0x0002 0000	0x17FF FFFF	384MB	保留	
			0x0000 0000	0x0001 FFFF	128KB	重定向内存空间	中断向量 0x0000 00C0~0x0000 0000

XN: 不可执行, 1 块 = 512MB

重定向内存空间: 主存、引导闪存、SRAM 取决于 BOOT 配置

2.5.2. 外周内存边界

下方的表格展示了芯片的相应外围边界地址。

表 2-2. 外周内存边界地址 – MG32F02A132/072

地址类型	边界地址		大小	节 / 组 外围设备	模块	注释
	起始地址	结束地址				
APB	0x5F00 0100	0x5FFF FFFF	16MB	APB	保留	
	0x5F00 0000	0x5F00 00FF	256B		APB	APB 模块全局控制
	0x5E00 0000	0x5EFF FFFF	16MB	保留	保留	
	0x5D04 0100	0x5DFF FFFF	16MB	WDT/RTC	保留	
	0x5D04 0000	0x5D04 00FF	256B		RTC	实时时钟
	0x5D01 0100	0x5D03 FFFF	192KB		保留	
	0x5D01 0000	0x5D01 00FF	256B		WWDT	窗口看门狗定时器
	0x5D00 0100	0x5D00 FFFF	64KB		保留	
	0x5D00 0000	0x5D00 00FF	256B		IWDT	独立看门狗定时器
	0x5C08 0100	0x5CFF FFFF	15MB	CMP/DAC	保留	
	0x5C08 0000	0x5C08 00FF	256B		DAC	数模转换器
	0x5C00 0100	0x5C07 FFFF	512KB		保留	
	0x5C00 0000	0x5C00 00FF	256B		CMP	模拟比较器 0,1,2,3
	0x5B00 0100	0x5BFF FFFF	16MB	ADC	保留	
	0x5B00 0000	0x5B00 00FF	256B		ADC	模数转换器
	0x5700 0000	0x5AFF FFFF	64MB	保留	保留	
	0x5686 0100	0x56FF FFFF	8MB	TM2x/3x	保留	
	0x5686 0000	0x5686 00FF	256B		TM36	16+16 位定时器带有 4 IC/OC/PWM
	0x5606 0100	0x5685 FFFF	8MB		保留	
	0x5606 0000	0x5606 00FF	256B		TM26	16+16 位定时器带有 2 IC/OC/PWM
	0x5600 0100	0x5605 FFFF	384KB		保留	
	0x5600 0000	0x5600 00FF	256B		TM20	16+16 位定时器带有 2 IC/OC/PWM
	0x5586 0100	0x55FF FFFF	8MB	TM0x/1x	保留	
	0x5586 0000	0x5586 00FF	256B		TM16	通用 16+16 位定时器/计数器
	0x5580 0100	0x5585 FFFF	384KB		保留	
	0x5580 0000	0x5580 00FF	256B		TM10	通用 16+16 位定时器/计数器
	0x5501 0100	0x557F FFFF	8MB		保留	
	0x5501 0000	0x5501 00FF	256B		TM01	通用 8+8 位定时器/计数器
	0x5500 0100	0x5500 FFFF	64KB		保留	
	0x5500 0000	0x5500 00FF	256B		TM00	通用 8+8 位定时器/计数器
	0x5400 0000	0x54FF FFFF	16MB	保留	保留	
	0x5300 0100	0x53FF FFFF	16MB	SPI	保留	
	0x5300 0000	0x5300 00FF	256B		SPI0	带有数据缓冲的 SPI
	0x5203 0100	0x52FF FFFF	16MB	UART	保留	
	0x5203 0000	0x5203 00FF	256B		URT3	先进的 UART 总线控制器
	0x5202 0100	0x5202 FFFF	64KB		保留	
	0x5202 0000	0x5202 00FF	256B		URT2	先进的 UART 总线控制器
	0x5201 0100	0x5201 FFFF	64KB		保留	
	0x5201 0000	0x5201 00FF	256B		URT1	先进的 UART 总线控制器
	0x5200 0100	0x5200 FFFF	64KB		保留	
	0x5200 0000	0x5200 00FF	256B		URT0	先进的 UART 总线控制器
	0x5101 0100	0x51FF FFFF	16MB	I2C	保留	
	0x5101 0000	0x5101 00FF	256B		I2C1	I2C 总线控制器
	0x5100 0100	0x5100 FFFF	64KB		保留	

	0x5100 0000	0x5100 00FF	256B		I2C0	I2C 总线控制器
	0x5000 0100	0x50FF FFFF	16MB	EXT 中断	保留	
	0x5000 0000	0x5000 00FF	256B		EXIC	外部中断控制器
AHB	0x4FF0 0100	0x4FFF FFFF	1024KB	芯片	保留	
	0x4FF0 0000	0x4FF0 00FF	256B		CFG	硬件选项 (NVR0/1/2)
	0x4F00 0100	0x4FEF FFFF	15MB		保留	
	0x4F00 0000	0x4F00 00FF	256B		WRI	写入接口控制
	0x4E00 0000	0x4EFF FFFF	16MB	保留	保留	
	0x4D02 0100	0x4DFF FFFF	16MB	内存	保留	
	0x4D02 0000	0x4D02 00FF	256B		EMB	外部内存总线控制器
	0x4D00 0100	0x4D01 FFFF	128KB		保留	
	0x4D00 0000	0x4D00 00FF	256B		MEM	内部内存控制器
	0x4C03 0100	0x4CFF FFFF	16MB	系统	保留	
	0x4C03 0000	0x4C03 00FF	256B		SYS	系统和芯片控制
	0x4C02 0100	0x4C02 FFFF	64KB		保留	
	0x4C02 0000	0x4C02 00FF	256B		PW	电源管理控制器
	0x4C01 0100	0x4C01 FFFF	64KB		保留	
	0x4C01 0000	0x4C01 00FF	256B		CSC	时钟源控制器
	0x4C00 0100	0x4C00 FFFF	64KB		保留	
	0x4C00 0000	0x4C00 00FF	256B		RST	复位源控制器
	0x4BF0 0100	0x4BFF FFFF	1024KB	通用	保留	
	0x4BF0 0000	0x4BF0 00FF	256B		DMA	直接内存访问
	0x4B00 0100	0x4BEF FFFF	15MB		保留	
	0x4B00 0000	0x4B00 00FF	256B		GPL	通用逻辑
	0x4500 0000	0x4AFF FFFF	96MB	保留	保留	为后续设计保留
	0x4404 0100	0x44FF FFFF	16MB	IO 设置	保留	
	0x4404 0000	0x4404 00FF	256B		PE	
	0x4403 0100	0x4403 FFFF	64KB		保留	
	0x4403 0000	0x4403 00FF	256B		PD	
	0x4402 0100	0x4402 FFFF	64KB		保留	
	0x4402 0000	0x4402 00FF	256B		PC	
	0x4401 0100	0x4401 FFFF	64KB		保留	
	0x4401 0000	0x4401 00FF	256B		PB	
	0x4400 0100	0x4400 FFFF	64KB		保留	
	0x4400 0000	0x4400 00FF	256B		PA	
	0x4200 0000	0x43FF FFFF	32MB	保留		为后续设计保留
	0x4100 0200	0x41FF FFFF	16MB	保留		
	0x4100 0000	0x4100 01FF	512B	GPIO	IOP	IO 端口输入输出
	0x4000 0000	0x40FF FFFF	16MB	保留		为后续设计保留

表 2-3. 外围内存边界地址 – MG32F02A032

地址类型	边界地址		大小	节 / 组 外围设备	模块	注释
	起始地址	结束地址				
APB	0x5F00 0100	0x5FFF FFFF	16MB	APB	保留	
	0x5F00 0000	0x5F00 00FF	256B		APB	APB 模块全局控制
	0x5E00 0000	0x5EFF FFFF	16MB	保留	保留	
	0x5D04 0100	0x5DFF FFFF	16MB	WDT/RTC	保留	
	0x5D04 0000	0x5D04 00FF	256B		RTC	实时时钟
	0x5D01 0100	0x5D03 FFFF	192KB		保留	
	0x5D01 0000	0x5D01 00FF	256B		WWDI	窗口看门狗定时器

	0x5D00 0100	0x5D00 FFFF	64KB		保留	
	0x5D00 0000	0x5D00 00FF	256B		IWDT	独立看门狗定时器
	0x5C00 0100	0x5CFF FFFF	16MB	CMP/DAC	保留	
	0x5C00 0000	0x5C00 00FF	256B		CMP	模拟比较器 0,1
	0x5B00 0100	0x5BFF FFFF	16MB	ADC	保留	
	0x5B00 0000	0x5B00 00FF	256B		ADC	模数转换器
	0x5700 0000	0x5AFF FFFF	64MB	保留	保留	
	0x5686 0100	0x56FF FFFF	8MB	TM2x/3x	保留	
	0x5686 0000	0x5686 00FF	256B		TM36	16+16 位定时器带有 4 IC/OC/PWM
	0x5600 0000	0x5685 FFFF	8MB		保留	
	0x5586 0100	0x55FF FFFF	8MB	TM0x/1x	保留	
	0x5586 0000	0x5586 00FF	256B		TM16	通用 16+16 位定时器/计数器
	0x5580 0100	0x5585 FFFF	384KB		保留	
	0x5580 0000	0x5580 00FF	256B		TM10	通用 16+16 位定时器/计数器
	0x5501 0100	0x557F FFFF	8MB		保留	
	0x5501 0000	0x5501 00FF	256B		TM01	通用 8+8 位定时器/计数器
	0x5500 0100	0x5500 FFFF	64KB		保留	
	0x5500 0000	0x5500 00FF	256B		TM00	通用 8+8 位定时器/计数器
	0x5400 0000	0x54FF FFFF	16MB	保留	保留	
	0x5300 0100	0x53FF FFFF	16MB	SPI	保留	
	0x5300 0000	0x5300 00FF	256B		SPI0	带有数据缓冲的 SPI
	0x5201 0100	0x52FF FFFF	16MB	UART	保留	
	0x5201 0000	0x5201 00FF	256B		URT1	先进的 UART 总线控制器
	0x5200 0100	0x5200 FFFF	64KB		保留	
	0x5200 0000	0x5200 00FF	256B		URT0	先进的 UART 总线控制器
	0x5100 0100	0x51FF FFFF	16MB	I2C	保留	
	0x5100 0000	0x5100 00FF	256B		I2C0	I2C 总线控制器
	0x5000 0100	0x50FF FFFF	16MB	EXT 中断	保留	
	0x5000 0000	0x5000 00FF	256B		EXIC	外部中断控制器
AHB	0x4FF0 0100	0x4FFF FFFF	1024KB	芯片	保留	
	0x4FF0 0000	0x4FF0 00FF	256B		CFG	硬件选项 (NVR0/1/2)
	0x4F00 0100	0x4FEF FFFF	15MB		保留	
	0x4F00 0000	0x4F00 00FF	256B		WRI	写入接口控制
	0x4E00 0000	0x4EFF FFFF	16MB	保留	保留	
	0x4D00 0100	0x4DFF FFFF	16MB	内存	保留	
	0x4D00 0000	0x4D00 00FF	256B		MEM	内部内存控制器
	0x4C03 0100	0x4CFF FFFF	16MB	系统	保留	
	0x4C03 0000	0x4C03 00FF	256B		SYS	系统和芯片控制
	0x4C02 0100	0x4C02 FFFF	64KB		保留	
	0x4C02 0000	0x4C02 00FF	256B		PW	电源管理控制器
	0x4C01 0100	0x4C01 FFFF	64KB		保留	
	0x4C01 0000	0x4C01 00FF	256B		CSC	时钟源控制器
	0x4C00 0100	0x4C00 FFFF	64KB		保留	
	0x4C00 0000	0x4C00 00FF	256B		RST	复位源控制器
	0x4BF0 0100	0x4BFF FFFF	1024KB	通用	保留	
	0x4BF0 0000	0x4BF0 00FF	256B		DMA	直接内存访问
	0x4B00 0100	0x4BEF FFFF	15MB		保留	
	0x4B00 0000	0x4B00 00FF	256B		GPL	通用逻辑
	0x4500 0000	0x4AFF FFFF	96MB	保留	保留	为后续设计保留
	0x4403 0100	0x44FF FFFF	16MB	IO 设置	保留	
	0x4403 0000	0x4403 00FF	256B		PD	

	0x4402 0100	0x4402 FFFF	64KB		保留	
	0x4402 0000	0x4402 00FF	256B		PC	
	0x4401 0100	0x4401 FFFF	64KB		保留	
	0x4401 0000	0x4401 00FF	256B		PB	
	0x4400 0100	0x4400 FFFF	64KB		保留	
	0x4400 0000	0x4400 00FF	256B		PA	
	0x4200 0000	0x43FF FFFF	32MB	保留		为后续设计保留
	0x4100 0200	0x41FF FFFF	16MB	保留		
	0x4100 0000	0x4100 01FF	512B	GPIO	IOP	IO 端口输入输出
	0x4000 0000	0x40FF FFFF	16MB	保留		为后续设计保留

2.5.3. 启动模式

芯片启动时，硬件配置字节(**OB**)会被用来选择三种启动方式中的一种进行启动：

- 从用户应用程序（**AP**）闪存引导启动
- 从系统编程（**ISP**）引导启动
- 从内嵌 **SRAM** 引导启动

参照 “[内存启动模式](#)” 节中 **MEM_BOOT_MS** 寄存器的设置和 “[硬件选项字节](#)” 节中 **BOOT_MS** 硬件配置以获取更多信息。

2.5.4. 内存代码锁定

从笙泉工厂生产出来的芯片的内存代码是默认锁定的。用户可以通过笙泉 ICP 编程器解锁和编程内存。参照 “[硬件选项字节](#)” 节中的 **LOCK_DIS** 硬件配置以获取更多信息。

2.6. 芯片调试

该芯片内置含有完整的硬件调试方案的 Cortex™-M0 处理器，并拥有较多的硬件断点和监视点设置。这些配置可以通过 1 个双线串行调试端口（SWD）提供高透明的处理器、内存和外设情况。

请参照“Cortex™-M0 技术参考手册”以获得更多信息。

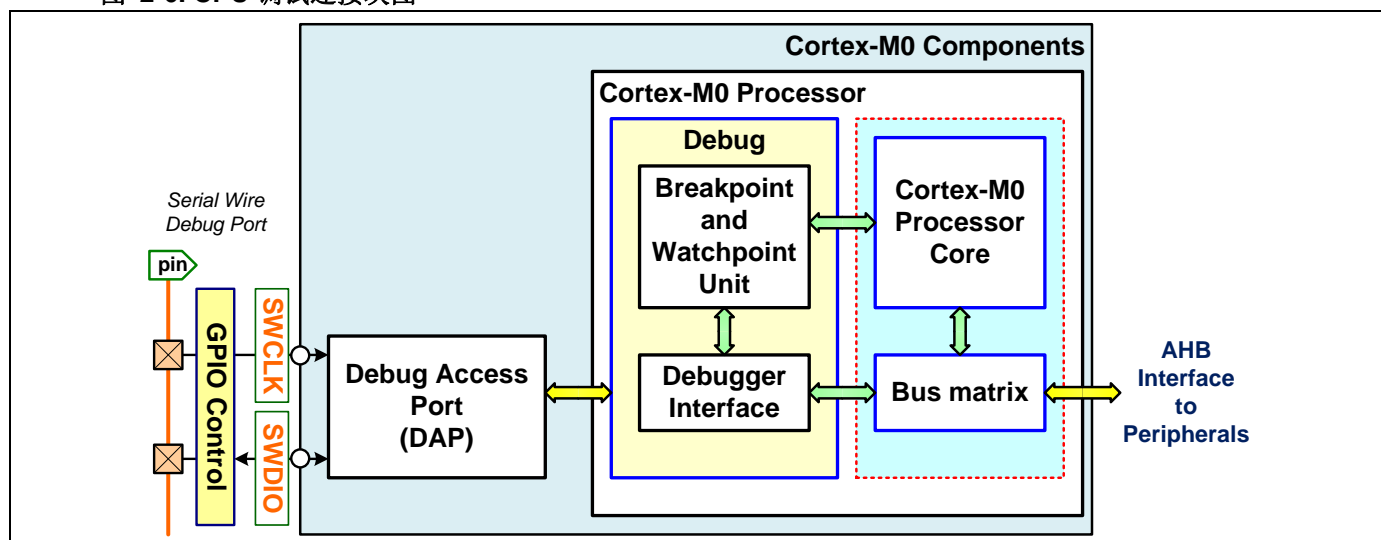
2.6.1. 芯片 DAP

该芯片包含了调试访问端口(DAP)，断点单元 (BPU)和数据监视点触发器(DWT)作为硬件调试方案。DAP 可以控制 SWD 接口。它通过 AHP 接口连接 Cortex™-M0 处理器的调试访问端口和总线矩阵到内部外设模块。

用户可以利用 SWD 接口通过给定的执行指令（断点）或者数据访问（监视点）停止核心。当芯片停止，用户可以监视或访问 CPU 内部状态，此外，用户可以监视或访问芯片任一使用的内存或寄存器。当用户完成监视并释放 DAP 时该芯片会恢复系统状态并继续处理器指令执行。

下面的图表展示了处理器调试连接块。

图 2-6. CPU 调试连接块图



2.6.2. SWD 接口

SWD 接口包含了 PC4 和 PC5 的 SWCLK 和 SWDIO 两根引脚。

- **SWCLK**

串行线时钟输入信号。

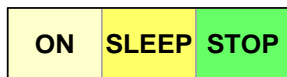
- **SWDIO**

串行线数据输入/输出双向信号。

SWD 的这两根引脚在上电复位或冷复位后被指定为带有内部上拉，用于芯片调试器的专用引脚，用户可以通过硬件设置寄存器 OB 的 SWD_PIN 来禁用 SWD 引脚并释放为 GPIO 或者其他功能复用引脚。参照“[硬件配置字节](#)”节和“[特殊引脚功能复用](#)”节以获取更多关于 SWD 引脚设置的信息。

3. 系统电源

3.1. 简介



The module can be running in all power operation modes.

该芯片的电源被设置成只需要通过 1 个电源输入，芯片还有 1 个内嵌 LDO 来为内部核心逻辑供电。该芯片支持 1 个电源管理器（PW）以管理上电复位电路、低压复位电路、掉电检测器（BOD0/1）、掉电控制和唤醒控制。

该芯片支持掉电模式：**SLEEP** 模式和 **STOP** 模式。这两种掉电模式可以降低芯片功耗并为芯片程序提供不同的节电计划。

3.2. 芯片电源特性

- 内置 1 个用于核心逻辑供电的 1.8V 输出调节器
- 内置 2 个掉电检测器
 - BOD0 固定检测 1.7V
 - BOD1 可选检测 4.2V/3.7V/2.4V/2.0V
- 内置 1 个带有掉电和唤醒控制的电源管理控制器
- 支持 3 种电源工作模式
 - On（普通）模式、**SLEEP**，**STOP** 掉电模式
- 支持通过多种源从 **SLEEP/STOP** 模式唤醒
 - 来自 GPIO、IWDG、RTC、模拟比较器、I2C、BOD 的唤醒源

3.3. 电源工作模式

电源控制器一共支持 3 种电源模式：**ON**, **SLEEP**, **STOP**。**PW_STATE** 寄存器提供了用于调试的当前工作模式状态。



- **ON 模式**
在 **ON** 模式，处理器可以全速运行，所有的外设均可以最大功率正常进行工作，同时，这些模块也可以为了降低功耗而独立进行启用和禁用。
- **SLEEP 模式**
在 **SLEEP** 模式，只有处理器会被冻结并进入 **SLEEP** 模式，所有的外设均可自行设置继续工作或者休眠。在该模式下，该芯片可以被相关的中断或者事件的发生唤醒。参照“[唤醒和中断](#)”节以获取更多信息。
- **STOP 模式**
STOP 模式提供了最低的功耗，与 **SLEEP** 模式不同的地方在于处理器会进入深度睡眠模式，且除了几个特殊模块和设备以外，所有的外设都会被禁用。这些特殊的模块和设备可以被设置继续在 **STOP** 模式中继续工作或者禁用，这些设备包括 **IWDT**, **RTC**, **CMP** 模块和 **LVR**, **BOD0**, **BOD1** 设备。内部电压调节器也会以低功耗模式中运行。
在该模式下，芯片可以被一些外部输入线（**GPIO**）和一些事件检测唤醒。参照“[唤醒和中断](#)”节以获取更多信息。

下面的表格展示了内部设备在不同的工作模式下的状态。

表 3-1. 电源工作模式

电源工作模式	普通		
	ON	SLEEP	STOP
内部设备			
电流功耗	全功耗	低功耗	超低功耗
ARM32 Cortex-M0	普通模式	睡眠模式	深度睡眠模式
核心 LDO	全/低 功耗 (*1)	全/低 功耗 (*1)	全/低 功耗 (*2)
CPU 时钟	运行	停止	停止
系统时钟/PLL/IHRCO	运行	运行	停止
XOSC/ILRCO	普通	普通	硬件设置
IO 引脚	普通	普通	普通
外围模块	普通	可设置	除了可设置 IWDT,RTC,CMP 之外均禁用
进入电源模式		执行 WFI/WFE 指令+ SLEEPDEEP=0	执行 WFI/WFE 指令+ SLEEPDEEP=1
唤醒事件		所有的中断或唤醒事件	LVR, BOD0/BOD1, CMP,RTC(通过 CK_LS,CK_UT),外部中断引脚 (只可电平检测), IWDT(CK_ILRCO), I2C(从机地址检测)

注释 *1: LDO 可通过设置寄存器 **PW_LDO_ON** 选择普通或低功耗模式。

注释 *2: LDO 可通过设置寄存器 **PW_LDO_STP** 选择普通或低功耗模式。

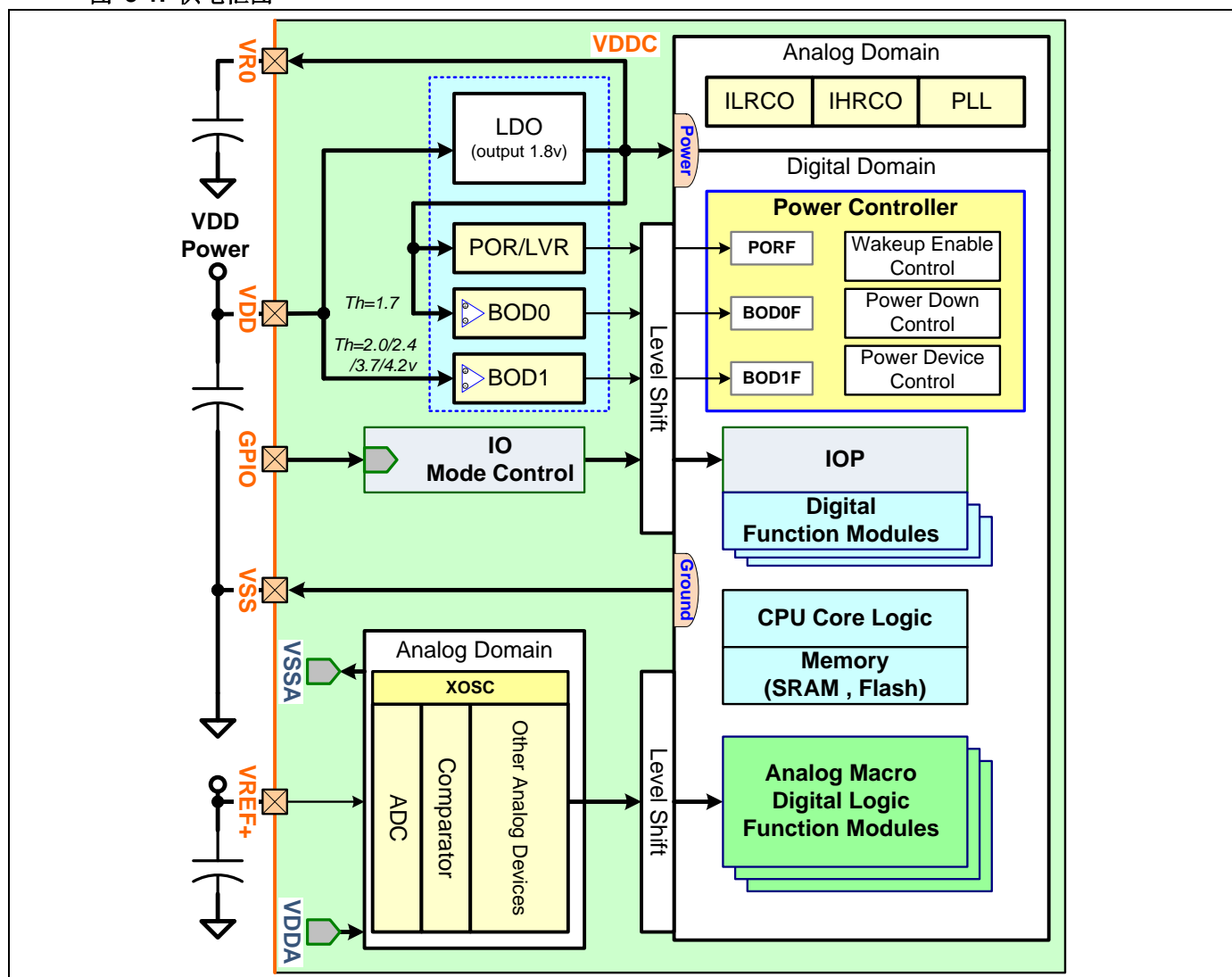
3.4. 供电

该芯片的电源只需要通过一个简单的 PCB 设计的 1 个电源输入即可。内嵌的 1 个内部低压差线性稳压器(LDO)可产生+1.8V 的电压 VDDC 来为核心逻辑进行供电。

VDD 引脚用于 IO 电源输入和内部 LDO 输入，**VSS** 引脚用于连接内部 LDO 的内部参考地、硬宏和数字逻辑的外部接地。**VRO** 引脚是 LDO 的输出，且为了保证正常工作，还需要连接旁路电容。**+VREF** 引脚是 ADC 的参考电压输入，在通常应用中，直接连接 **VDD** 即可。参照“[芯片电源应用电路](#)”节以获取更多信息。

下面的图表展示了芯片的电源供电。

图 3-1. 供电框图

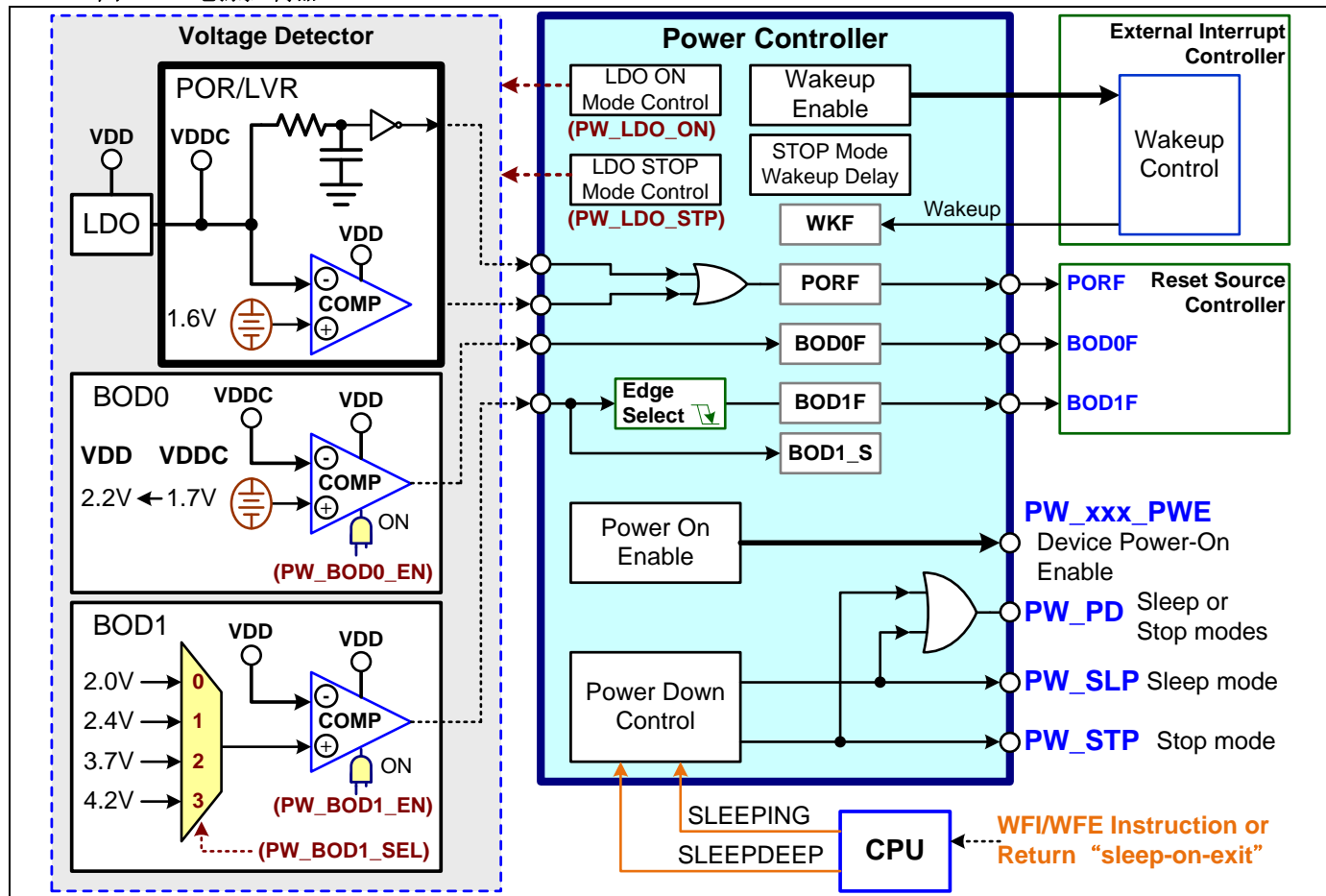


3.5. 电源控制块

该芯片支持 1 个电源控制器(PW)来管理上电复位(POR)电路、低压复位(LVR)电路、掉电检测器(BOD0/1)、掉电控制和唤醒控制。

下面的图表展示了电源控制块。

图 3-2. 电源控制器



3.6. 电源电压检测

3.6.1. POR/LVR 检测器

该芯片内嵌 1 个总是有效的上电复位 (POR) 和 1 个低压复位 (LVR) 电路。

当芯片上电，内部 RC 复位电路将会在电压阈值 0.7V (不同芯片会在 0.6~0.8V 之间浮动) 的 VDD 时被启动。在上电状态过程中，当 VDD 电源掉到 0.3V 以下时，RC 复位电路需被重启并重新上电。LVR 可以监视核心供电 VDDC 以保护核心逻辑工作。当核心电源掉到 1.60V 以下，LVR 将会被启动，直到核心电源重新上升到 1.65V。PORF 标志 (**PW_PORF**) 用于检测芯片的复位状态。

3.6.2. 掉电检测器

在该芯片中，有 2 个掉电检测器 (BOD0, BOD1) 用于监测芯片电源。用户可以独立地通过 **PW_BOD0_EN** 或 **PW_BOD1_EN** 寄存器使能 BOD0 或者 BOD1。

BOD0 提供固定检测 VDDC=1.7V 的电压，并用于监视核心电源，以保护内部闪存写操作。当核心电源电压低于 BOD0 电压阈值，表明闪存写入功能在这个状态下将会失败。当 BOD0 到达检测阈值电压时，标志 **PW_BOD0F** 会被置起。BOD0 通过软件设置提供了中断处理器或复位的功能。

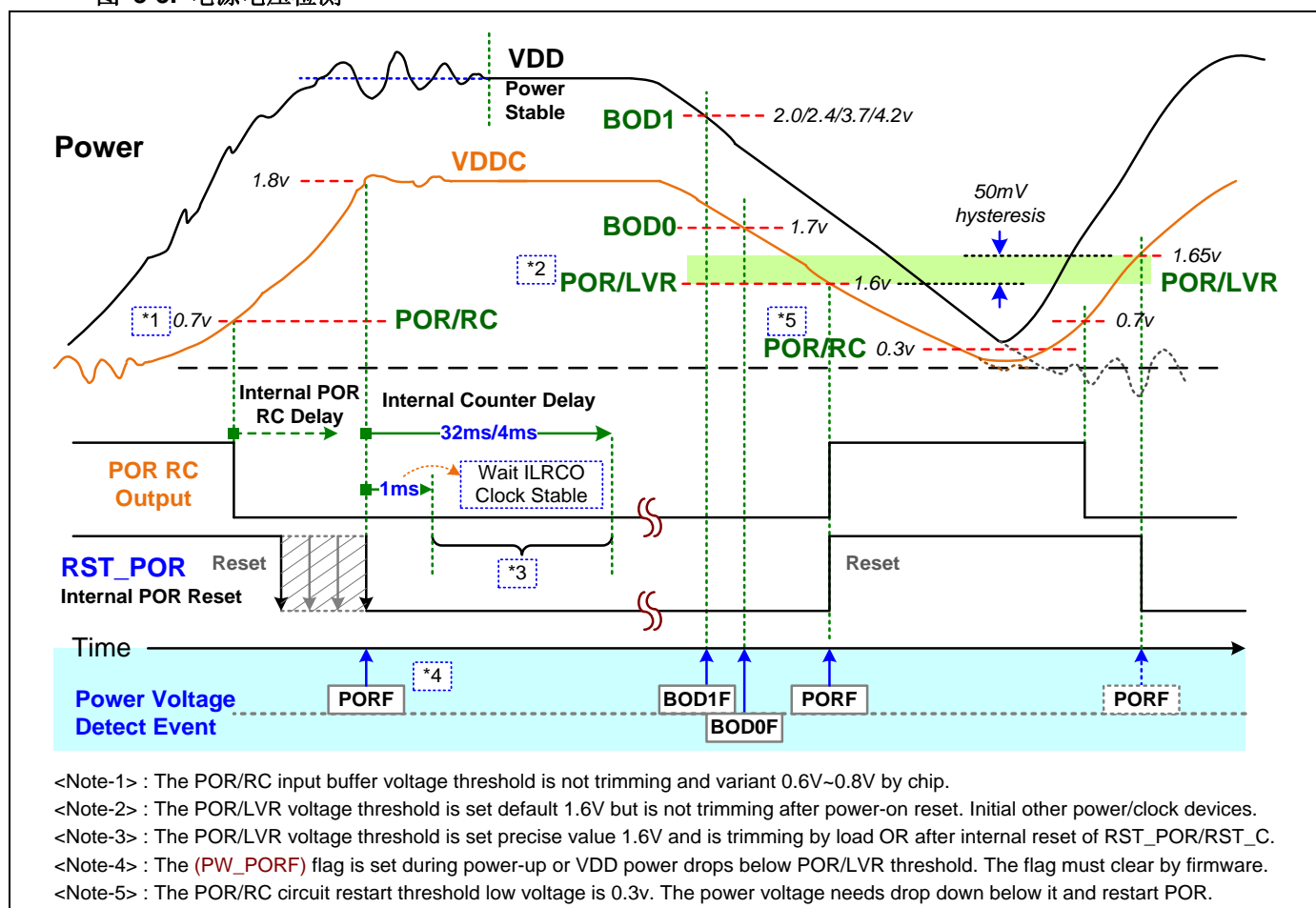
若 **SYS IEA** 在 SYS 模块中使能，且 **PW_BOD0 IE** 在 PW 模块中使能，BOD0F 标志置起事件会产生系统标志中断。它可在处理器处于 **ON** 模式或者 **SLEEP** 模式时中断处理器，若 **PW_WKSTP_BOD0** 被使能，该检测事件还可以将处理器从 **STOP** 模式中唤醒。

在该芯片中, 还有 1 个掉电检测器 (BOD1) 用于监测 VDD 电源, 控制方式与 BOD0 相同, 通过 **PW_BOD1_IE** 和 **PW_WKSTP_BOD1** 寄存器即可。BOD1 与 BOD0 不同的地方是 BOD1 通过设置寄存器 **PW_BOD1_SEL** 提供了可设置检测电压 $VDD=2.0/2.4/3.7/4.2V$ 。另一个不同点是用户可以通过设置寄存器 **PW_BOD1_TRGS** 选择 BOD1 的触发沿: 上升沿、下降沿、双边沿中断。此外还提供了 1 个状态位 **PW_BOD1_S** 作为 BOD1 比较器输出。

3.6.3. 电源电压检测阈值

下面图表展示了上电和掉电时的电源电压检测阈值。

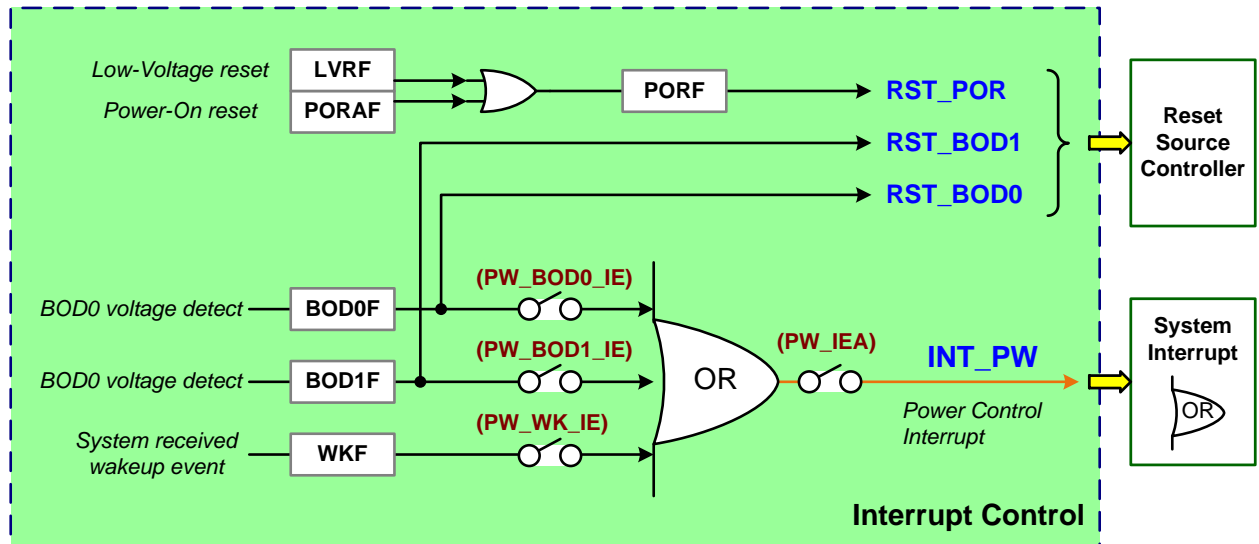
图 3-3. 电源电压检测



3.7. 中断和复位

在 PW 控制模块中有这些信号产生：**INT_PW**, **RST_POR**, **RST_BOD0**, **RST_BOD1**。**INT_PW** 发送到外部中断控制器（EXIC）中作为中断事件；**RST_POR**, **RST_BOD0** 和 **RST_BOD1** 发送到复位源控制器作为复位事件。

图 3-4. 电源控制器中断和复位事件



3.7.1. PW 中断标志

中断 **INT_PW** 在 BOD0, BOD1 的电压检测标志和唤醒事件标志是“或”事件。这会被作为其中一个 **INT_SYS** 系统中断事件输出。参照“系统普通控制”章中的“[中断和事件](#)”节以获取更多信息。

这些中断标志用于中断服务例程（ISR）的流控制。通常这些标志会被硬件置位并在相关的 ISR 服务工作完成时由软件清零。每个中断标志有 1 个中断使能位，用户可以选择使能或禁用。在这个模块中，有一个中断全局使能位 **PW_IEA** 来使能或禁用该模块所有的中断源。

通常这些中断标志会被硬件置位或被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和中断使能位的信息。

- **BOD0F**

BOD0 掉电检测中断标志 (**PW_BOD0F**)。相关的中断使能寄存器位是 **PW_BOD0_IE**。该标志会在电源电压低于 BOD0 阈值时被产生。

- **BOD1F**

BOD1 掉电检测中断标志(**PW_BOD1F**)。相关的中断使能寄存器位是 **PW_BOD1_IE**。

- **WKF**

系统接收到的唤醒事件标志 (**PW_WKF**)。相关的中断使能寄存器位是 **PW_WK_IE**。

3.7.2. PW 复位事件

RST_POR, **RST_BOD0** 和 **RST_BOD1** 信号发送到复位源控制器（RST）作为热复位事件或者冷复位事件。这些复位事件可通过设置 RST 模块的寄存器使能用于复位芯片。

参照系统复位章的描述以获取更多关于复位事件和控制的信息。

- **PORF**

上电复位状态标志(**PW_PORF**)。PORF 标志在上电或者 VDD 电源掉到 POR/LVR 阈值以下时会被置位。通常，这个标志可在上电后被固件清除并用于表明固件流是上电后的第一次循环。

- **BOD0F, BOD1F**

这两个标志是 BOD0、BOD1 掉电检测中断标志(**PW_BOD0F**、**PW_BOD1F**)。

3.8. 寄存器保护和锁定

电源控制器复位之后，除了 **PW_STA**、**PW_KEY** 这两个寄存器以外，所有的 **PW** 寄存器将会处于写保护状态。通过向 **PW_KEY** 寄存器写 **0xA217** 值即可解除寄存器保护，并可持续处理控制。相对地，写其他除 **0xA217** 以外的值将会使寄存器处于保护状态。读 **PW_KEY** 寄存器的值以获得寄存器的状态是被保护 (=1) 还是未保护 (=0)。

参照系统复位章的“[寄存器保护和锁定](#)”节的表格和描述以获取更多信息。

在硬件功能控制中比较特别的，**PW_WKSTP_IWDT** 是被 **IWDT_LOCK** 寄存器控制锁定的。另外，寄存器 **PW_WKSTP_RTC** 是被 **RTC_LOCK** 寄存器控制锁定的。参照 **IWDT** 和 **RTC** 章的“寄存器保护和锁定”节以获取更多信息。

3.9. 处理器掉电

为了使芯片进入掉电模式，固件必须执行 **WFI** 或 **WFE** 指令来使 **CPU** 强制进入睡眠或者深度睡眠模式。然后芯片就会进入 **SLEEP** 或 **STOP** 模式。用户可以通过设置 **CPU** 寄存器的 **SLEEPDEEP** 来设置处理器执行 **WFI** 或者 **WFE** 指令后进入的是哪种掉电模式。

下面表格展示了处理器和系统芯片的掉电模式关系。

表 3-2. 电源模式选择

芯片电源模式	CPU	CPU 寄存器	状态
		SLEEPDEEP	
ON	运行	x	正常
SLEEP	睡眠	0	所有模块均可通过设置继续工作
STOP	深度睡眠	1	除 IWDT , RTC , CMP 可设置继续运行外，所有的模块都会被禁用，电压调节器可被设置为普通或低功耗模式。

注释：执行 **WFI** 或 **WFE** 指令后进入 **SLEEP** 或 **STOP** 模式。

3.10. 掉电模式下设备电源使能控制

PW 控制器支持电源控制计划，用于内部设备在 **SLEEP** 或 **STOP** 模式的时候。用户可以在进入 **SLEEP** 或 **STOP** 模式之前规划内部每个独立设备是工作还是关闭。当 **CPU** 执行 **WFI** 或 **WFE** 指令进入睡眠或深度睡眠模式时，该芯片会进入 **SLEEP** 或 **STOP** 模式。

下面的表格展示了不同工作模式下内部设备的电源控制。

表 3-3. 内部设备电源控制

内部设备	ON	SLEEP	STOP
核心 LDO	on (普通或低功耗模式)	on (普通或低功耗模式)	on (普通或低功耗模式)
上电复位	on	on	PW 寄存器预先设置
BOD0 掉电检测器	寄存器设置	寄存器设置	PW 寄存器预先设置
BOD1 掉电检测器	寄存器设置	寄存器设置	PW 寄存器预先设置
ILRCO/XOSC	寄存器设置	寄存器设置	硬件设置 (*1)
IHRCO/PLL	寄存器设置	寄存器设置	off
模拟比较器	寄存器设置	PW 寄存器预先设置	PW 寄存器预先设置

注释 *1：硬件通过 **ILRCO** 或 **XOSC** 时钟自动设置开/关

- **POR**

用户可提前设置 **PW_STP_POR** 寄存器规划处于 **STOP** 模式时 **POR** 电源使能控制。

- **BOD0 / BOD1**

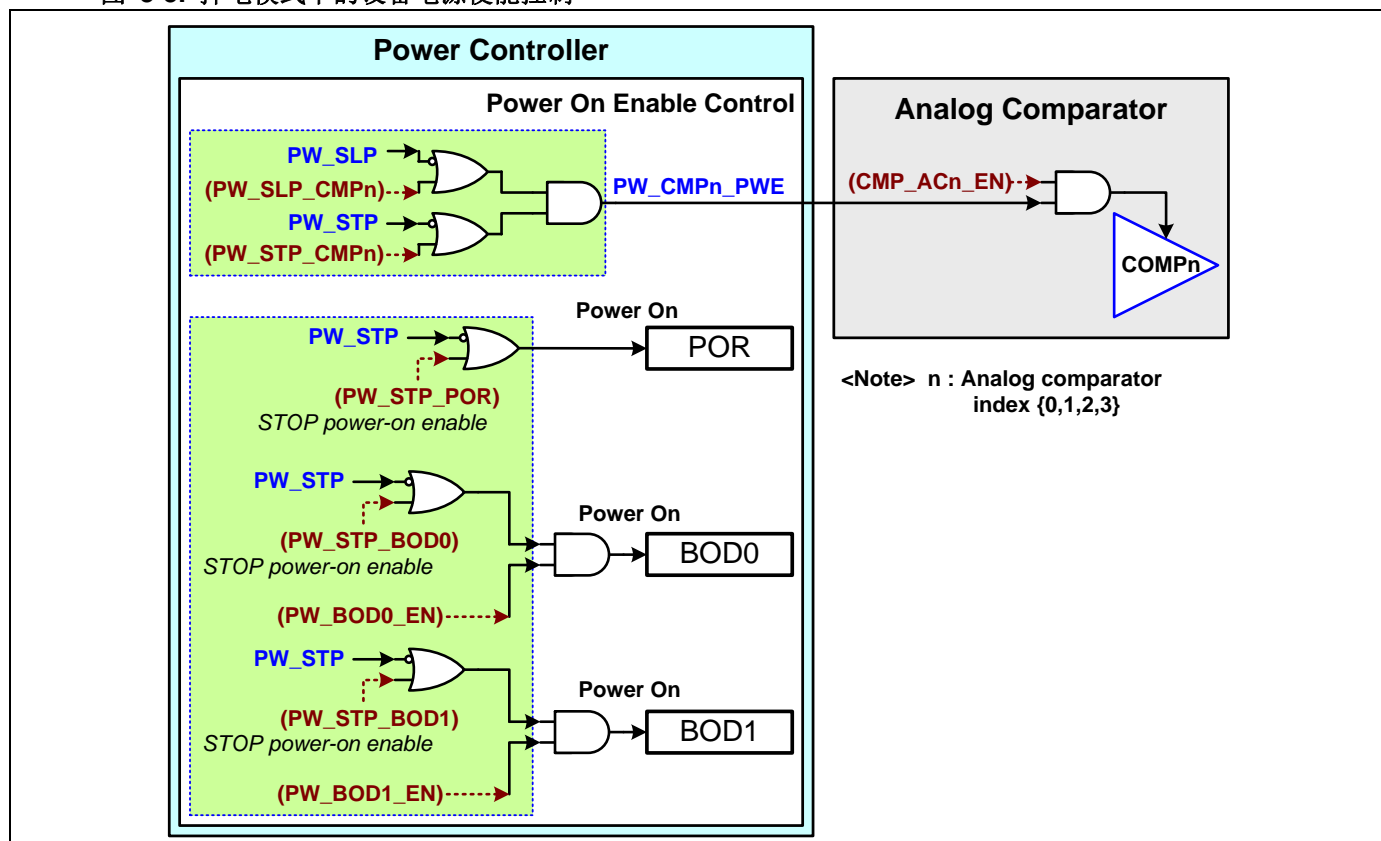
用户可提前设置 **PW_STP_BOD0** 和 **PW_STP_BOD1** 寄存器规划处于 **STOP** 模式时 **BOD0** 和 **BOD1** 电源使能控制。

- 模拟比较器

用户可提前设置 **PW_SLP_CMPn** 寄存器规划处于 **SLEEP** 模式时模拟比较器电源使能控制，用户还可以预先设置 **PW_STP_CMPn** 寄存器规划处于 **STOP** 模式时模拟比较器电源使能控制。(n = {0,1,2,3})

下面的图表展示了内部设备的电源使能控制。

图 3-5. 掉电模式下的设备电源使能控制



3.11. 唤醒控制

PW 控制器支持 **SLEEP** 和 **STOP** 模式下的可编程和灵活的唤醒源控制。用户需在进入 **SLEEP** 和 **STOP** 模式前规划唤醒源。参照中断章的“[RXEV 和 TXEV 控制](#)”节以获取更多关于 RXEV 唤醒功能的相关信息。

3.11.1. 唤醒事件源

SLEEP 模式下，只要相应的模块和 NVIC 使能了，所有的中断事件都可以作为唤醒事件，所有的 PA/PB/PC/PD 端口的 GPIO 引脚都可以设置为外部中断作为 **SLEEP** 的唤醒源。该芯片可以在 EXIC 模块设置里检测电平或边沿触发输入信号。

STOP 模式下，该芯片提供了 BOD0, BOD1, 模拟比较器, RTC, IWDG, SWD 调试和 GPIO 引脚触发作为唤醒源。

- **BOD0/BOD1**

用户可通过 BOD0/BOD1 的 **PW_WKSTP_BOD0** 和 **PW_WKSTP_BOD1** 寄存器设置，使能 BOD0/BOD1 通过电压检测把处理器从 **STOP** 模式唤醒。

- **SWD/调试**

用户可通过 SWD 的 **PW_WKSTP_DBG** 寄存器设置，使能 SWD 调试通信把处理器从 **STOP** 模式唤醒。

- **GPIO**

用户可通过 GPIO 的 EXIC 中断使能寄存器设置，使能通过仅电平触发检测的外部中断把处理器从 **STOP** 模式唤醒。

- **模拟比较器**

用户可通过模拟比较器的 **PW_WKSTP_CMPn** 寄存器设置，使能 CMPn 通过电压检测把处理器从 **STOP** 模式唤醒。(n = {0,1,2,3})

- **RTC**

用户可通过 **PW_WKSTP_RTC** 寄存器设置, 让处理器通过 RTC 模块事件从 **STOP** 模式唤醒。

- **IWDT**

用户可通过 **PW_WKSTP_IWDT** 寄存器设置, 让处理器通过 IWDT 模块事件从 **STOP** 模式唤醒。

- **I2C**

用户可通过 **PW_WKSTP_I2Cx** 寄存器设置, 使能通过 I2C 从机地址检测把处理器从 **STOP** 模式唤醒。(x = 模块标号)

[注释]: I2C 唤醒功能不支持于 MG32F02A132/072。

下面的表格展示了 **SLEEP** 和 **STOP** 模式下的唤醒事件源。

表 3-4. 唤醒事件源

内部设备	SLEEP		STOP		注释
	唤醒使能	唤醒事件	唤醒使能	唤醒事件	
BOD0	中断使能设定	使能的中断事件	PW 预先设定寄存器	BOD0 事件检测	
BOD1	中断使能设定	使能的中断事件	PW 预先设定寄存器	BOD1 事件检测	
SWD/调试	中断使能设定	使能的中断事件	PW 预先设定寄存器	SWD 通信检测	
外部 INT 引脚	中断使能设定	使能的中断事件 (检测边沿/电平)	中断使能设定	使能的中断事件 (只能检测电平)	
模拟比较器	中断使能设定	使能的中断事件	PW 预先设定寄存器	比较器输出状态变化检测	
RTC	中断使能设定	使能的中断事件	PW 预先设定寄存器	RTC 事件检测	CK_LS,CK_UT 在 STOP 模式下
IWDT	中断使能设定	使能的中断事件	PW 预先设定寄存器	IWDT 事件检测	CK_ILRCO 在 STOP 模式下
I2Cx	中断使能设定	使能的中断事件	PW 预先设定寄存器	仅从机地址检测	
其他模块	中断使能设定	使能的中断事件			不支持 STOP 模式唤醒

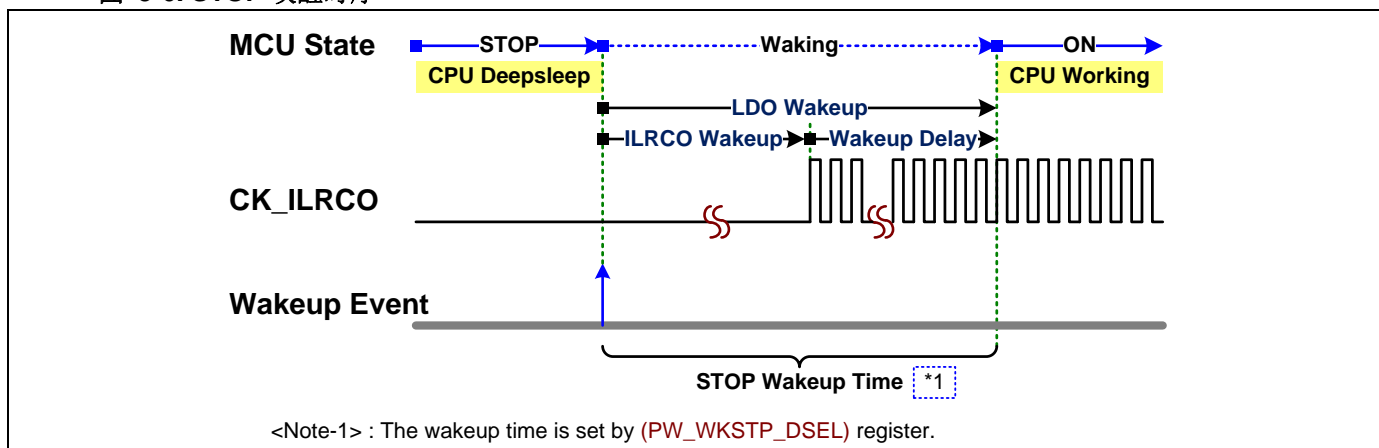
3.11.2. 唤醒和中断

在配置了唤醒源且固件执行 WFI 或 WFE 指令使芯片进入掉电模式后, 芯片将监视这些被使能了的唤醒源。PW 控制器将会控制这些信号, 除外部中断触发事件外, 外部中断触发事件可以通过或禁止信号发送到 EXIC 控制器进行唤醒控制。

EXIC 通过中断路径将唤醒信号与 NVIC 连接, 从而触发唤醒进程。当 WIC (唤醒中断控制器) 收到有效的唤醒信号, 它会重启时钟系统并唤醒 CPU。用户可以为了稳定时钟, 通过 **PW_WKSTP_DSEL** 寄存器从 **STOP** 模式唤醒选择唤醒延时时间 45~150us。

下面的图表展示了从 **STOP** 模式唤醒时序图。

图 3-6. STOP 唤醒时序



当芯片从掉电模式唤醒，WKF 标志将会被置起，而如果 **PW_WK_IE** 被使能，则用户会被触发的中断提醒，用户可以通过 **PW_WKMODE** 寄存器获得系统是从哪种掉电模式唤醒的信息(**SLEEP** 或 **STOP**)。

下面的图表展示了系统唤醒和中断控制的连接图。

图 3-7. 唤醒和中断控制 – MG32F02A132/072

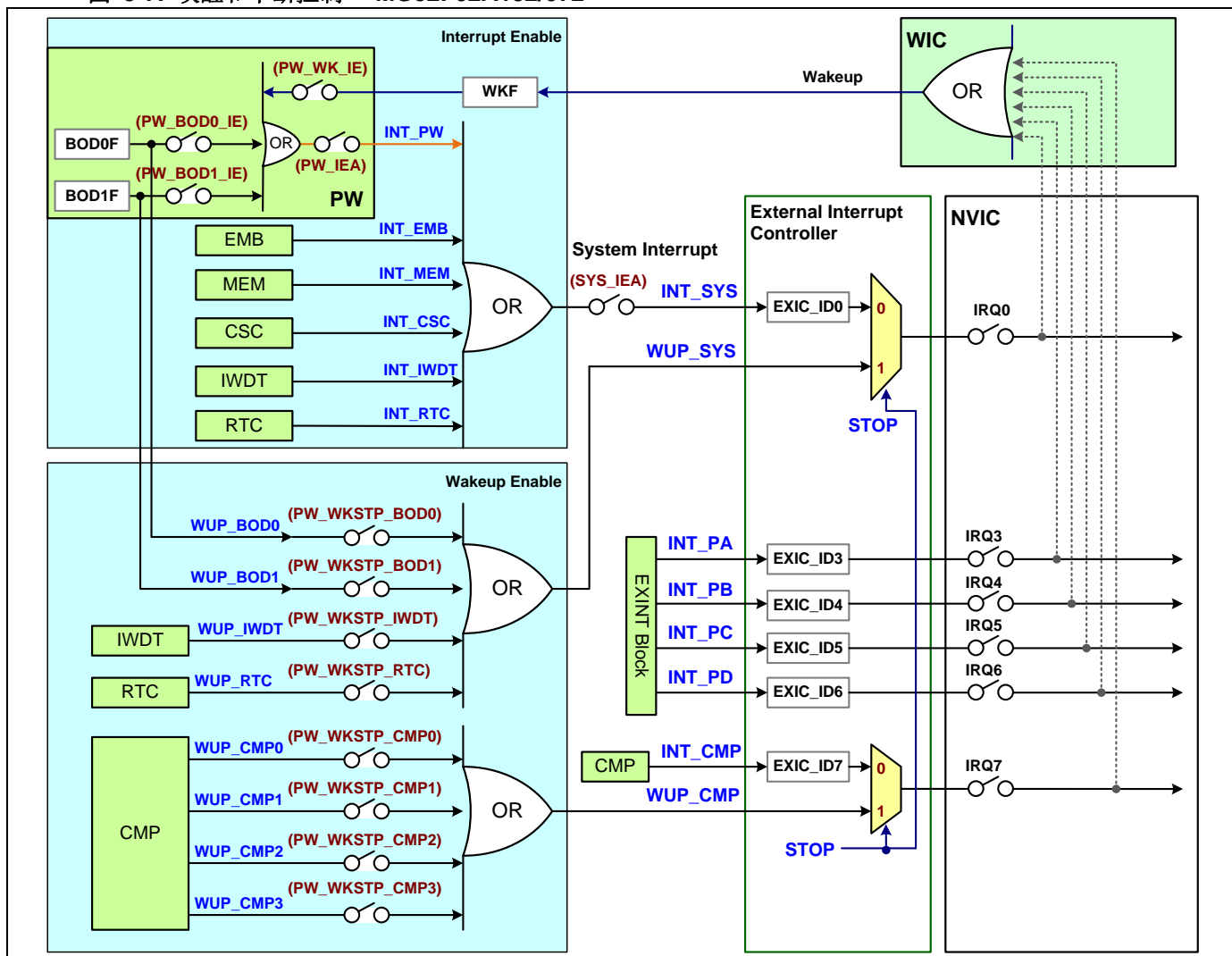
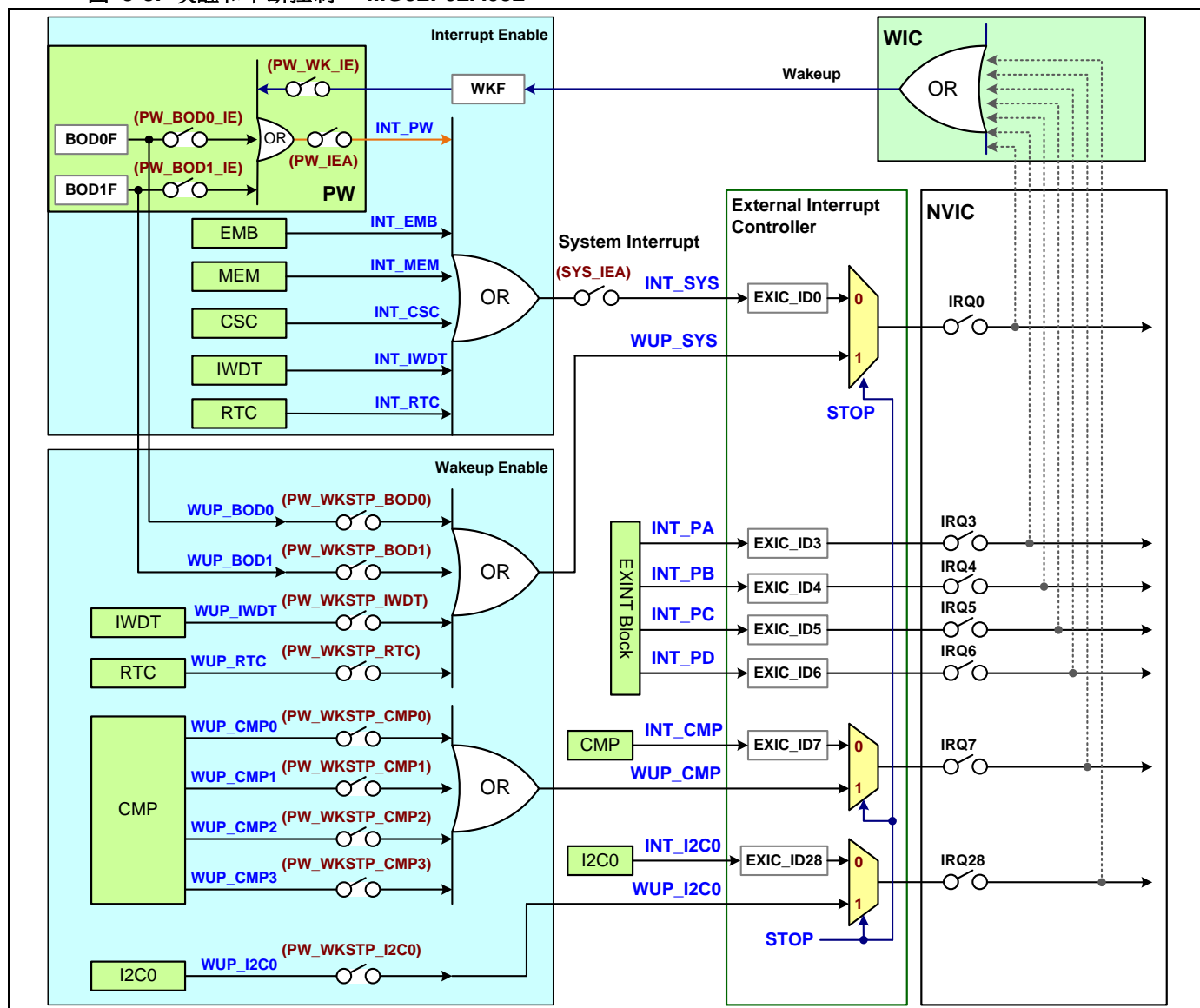


图 3-8. 唤醒和中断控制 – MG32F02A032



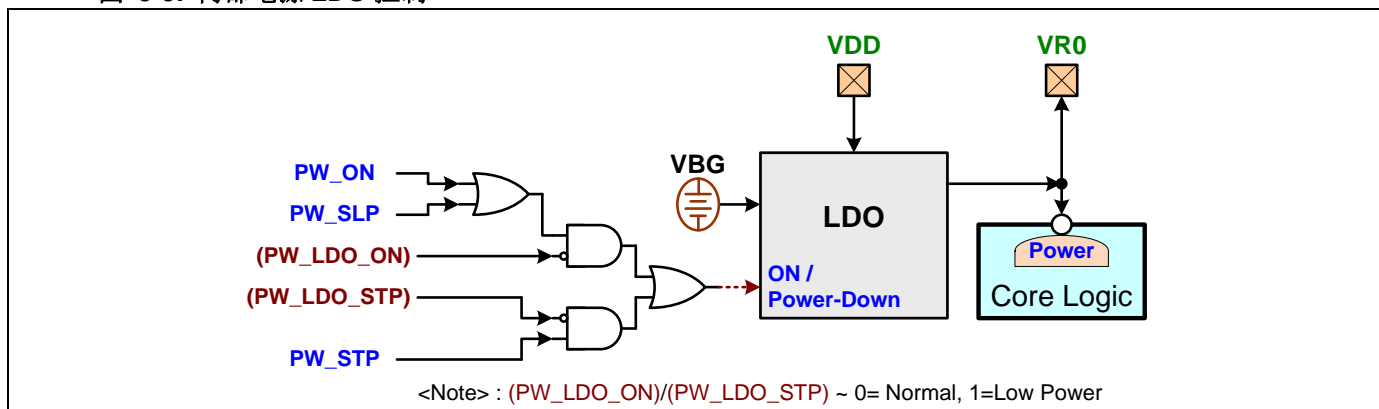
3.12. 内部设备控制

3.12.1. 内部电源 LDO

该芯片内嵌 1 个内部低压差线性稳压器（LDO）用于提供内部核心逻辑的电力。LDO 有两种工作模式，分别是正常模式和低功耗模式，用户可以通过设置 **PW_LDO_ON** 寄存器来选择 **ON** 模式和 **SLEEP** 模式，同时，用户也可以在进入 **STOP** 模式前预先设置 **PW_LDO_STP** 寄存器。参照“[掉电模式下设备电源使能控制](#)”节以获取更多关于其他内部设备的电源使能控制信息。

普通模式下，LDO 能最大提供 30mA 的核心电力，而低功耗模式，LDO 最大能提供 0.5mA 的核心电力，当芯片核心电力工作在低于 0.5mA 时，用户可以选择低功耗模式以省电。

图 3-9. 内部电源 LDO 控制



3.12.2. 内部参考电压

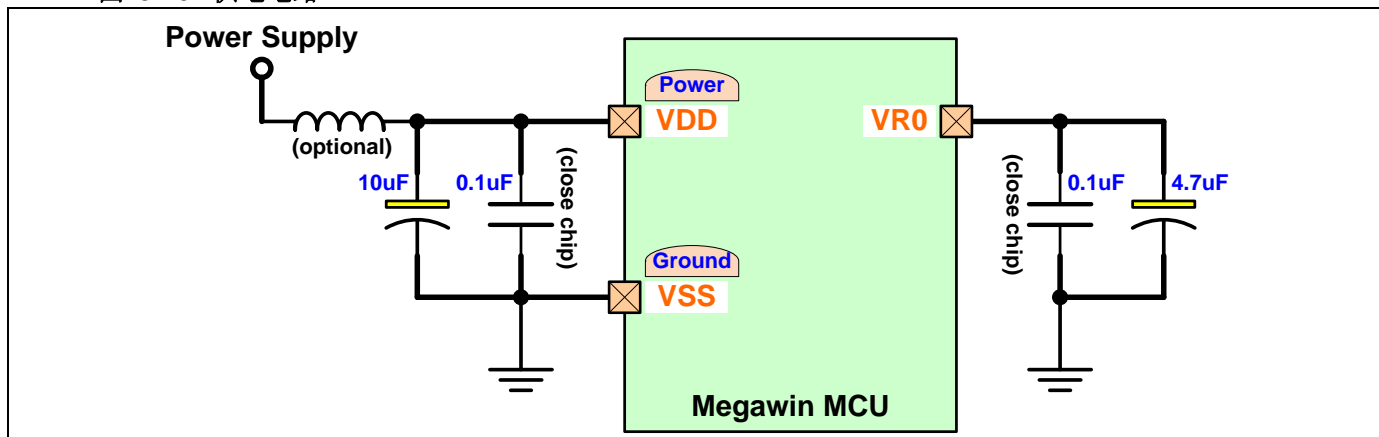
该芯片内建有内部参考电压（IVR）源用于 ADC 和模拟比较器模拟部分。为了使用 ADC 和模拟比较器，用户需要使能 **PW_IVR_EN** 寄存器。

3.13. 芯片供电应用电路

为了让芯片在 2.0V 到 5.5V 之间正常工作，需要在 **VDD2/VSS2** 电源引脚上增加一些外部去耦和旁路电容，如下图所示。此外相同的应用建议用在封装 LQFP80 **VDD2/VSS2** 电源引脚上。

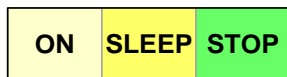
强烈建议在 VDD 电源放置 1 个 10uF 的去耦电容。再放一个 1 个 0.1uF 的电容，该电容需要放置得接近 **VDD** 引脚，该旁路电容能储存电力为 **VDD** 线路提供稳定的电压。

图 3-10. 供电电路



4. 系统复位

4.1. 简介



The module can be running in all power operation modes.

在复位的过程中，所有的寄存器都会被设置成初始值，程序也会从复位向量开始执行。该芯片包含了 1 个复位源控制器（RST）来管理多种复位源并产生热复位和冷复位信号到芯片系统和内部模块中。该控制器还为固件提供了复位事件标志，从而能对发生的复位源进行识别。

注释：标志 (**OB** =硬件设置字节内存)在该章描述时会被使用。

4.2. 芯片复位特性

- 内嵌 1 个 POR(上电复位)/LVR(低电压复位)电路
- 内置 1 个复位源控制器
 - 可设置芯片冷复位和热复位复位源
 - 内部模块使用的独立的软件复位控制
- 提供多种复位源
 - POR/LVR/BOD0/BOD1/外部复位引脚输入/软件强制复位
 - IWD/WWDT/ADC/比较器
 - IAR(非法地址错误复位)/闪存访问保护错误复位
 - 丢失时钟检测(MCD)复位

4.3. 复位源控制器

下面的图表展示了复位源控制块。

图 4-1. 复位源控制器 – MG32F02A132/072

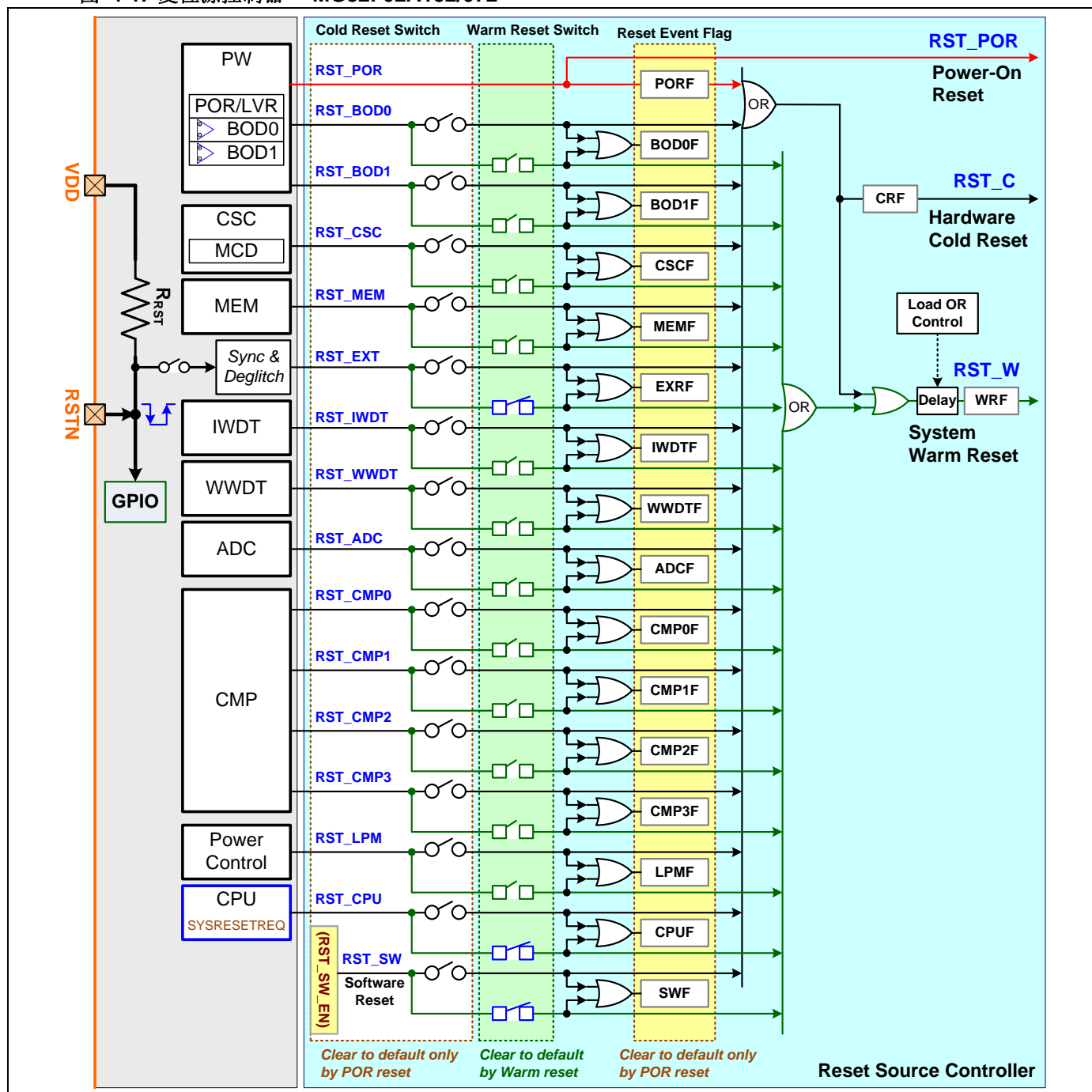
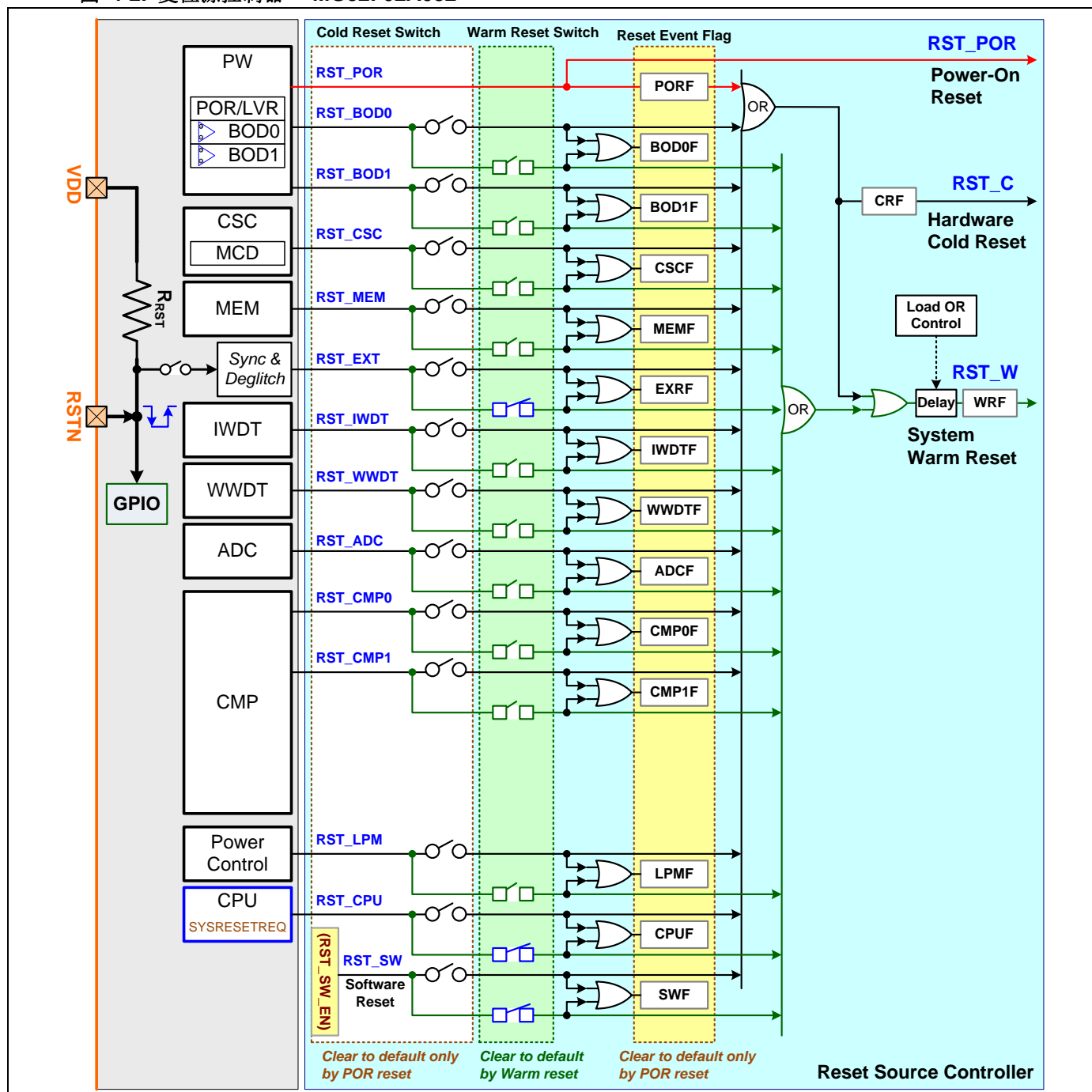


图 4-2. 复位源控制器 – MG32F02A032



4.4. 芯片复位

4.4.1. 芯片复位等级

该芯片提供 3 级复位等级：POR 复位、冷复位、热复位。POR 复位是最高优先级的复位，并且它是通过芯片硬件被产生的。冷复位是第二优先级的复位源，热复位则是最低优先级的复位源。

当 POR 复位发生时，它会导致芯片发生冷复位，而当冷复位发生时，它会导致芯片发生热复位。用户可以通过读取标志 CRF (**RST_CRF**) 和 WRF (**RST_WRF**) 以获取前一个复位的等级信息。参照图“复位源控制器”以获取详情。

下面的表格展示了不同复位等级下的复位控制功能。

表 4-1. 不同复位等级的复位控制功能

复位等级	芯片功能	
热复位 (中优先级)	(1) 复位芯片所有的块并从向量表的复位入口地址重新启动。 (2) 复位标志和冷复位使能位不会被清除。 (3) 一些设置字节内存(OB)数据读入硬件设置字节控制寄存器(OR)。 (4) 一些硬件控制寄存器数据读入相关模块控制寄存器，其他寄存器则复位至初始值。 (5) 保持可不被复位的可设置可被复位或不复位的 IO 引脚的 IO 设置。 (6) 锁定功能仍继续使用。	
	解锁状态	锁定状态
	清除热复位使能位 解锁功能被禁用	不清除热复位使能位 保持解锁功能
冷复位 (高优先级)	除以下不同外与“热复位”相同： (1) 所有设置字节 (OB) 数据读入硬件字节控制寄存器(OR)。 (2) 复位所有 IO 设置。 (3) 锁定/解锁功能禁用	
POR/LVR 复位 (最高优先级)	除以下不同外与“冷复位”相同： (1) 复位标志和冷/热复位使能位均被清除。	

● 上电复位

上电复位 (**POR**) 用于上电时内部复位芯片和 CPU。在 VDD 供电高于 POR 电压之前，该芯片都会保持复位状态而不开始工作。而且，一旦 VDD 供电低于 POR 阈值电压，该复位状态便会再次启动。在整个掉电周期内，为了保证上电复位，在重新开始供电之前，VDD 必须低于 POR 阈值电压。

POR 复位只会发送到 RST 控制器，而不会到其他模块中，它会清除 RST 控制器中的复位事件源的标志、冷复位源使能位和热复位源使能位。

参照系统电源章中的“[电源电压检测](#)”节以获取更多关于 POR 复位的信息。

● 冷复位

冷复位是第二优先级复位，当 POR 复位发生时，冷复位也会被产生，它会向比如 IWDT, WWDT ...等模块发送指令去执行深层模块复位。它还会导致所有的硬件配置 **OB** 重载，并禁用支持寄存器锁定功能的模块的寄存器锁定功能。

● 热复位

热复位是最低优先级的复位。热复位也会在冷复位发生时被产生。它发送给所有的模块以清除标志和硬件电路。它会导致一些硬件设置 **OB** 重载，并复位未锁定或者不支持锁定功能的模块的寄存器到默认值。如果 RST 控制器是未锁定的，它还会清除 RST 控制器的热复位源使能位。

4.4.2. 上电复位和芯片复位时序

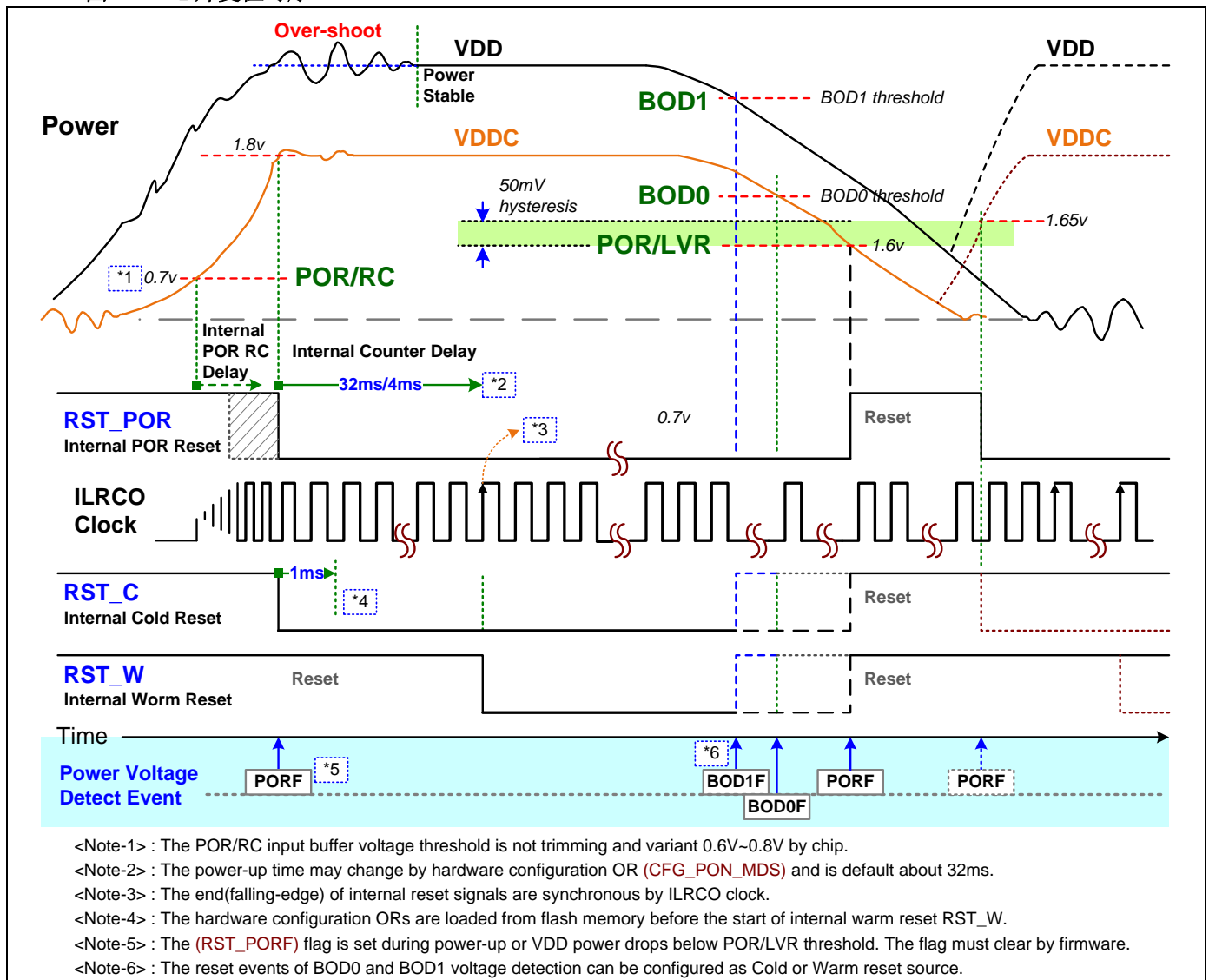
在上电过程中，内部 POR 电路会产生 POR 复位信号来启动 ILRCO 时钟电路和低压检测器。POR 复位根据内部 RC 的放电和电源电压检测器的检测电压控制并以不同的电源上电速率释放。而且标志 PORF (**RST_PORF**) 会在 POR 结束时被置起。

同时，POR 复位产生冷复位到其他内部设备中。这会强制将硬件设置 **OB** 的设置重载。为了等待 ILRCO 时钟稳定，该操作将会延迟 1ms 时间在冷复位后释放。

热复位将会复位 CPU 和所有的内部模块，它会在冷复位延时 4 或 32ms 后释放，该延时时间由硬件设置 **OB** 决定。

该芯片内建 BOD0/BOD1 和 POR/LVR 电压检测器监视芯片电源电压以保护芯片内存内容和保证芯片工作正常。参照“系统电源”章中“[电源电压检测](#)”节以获取更多信息。

图 4-3. 芯片复位时序



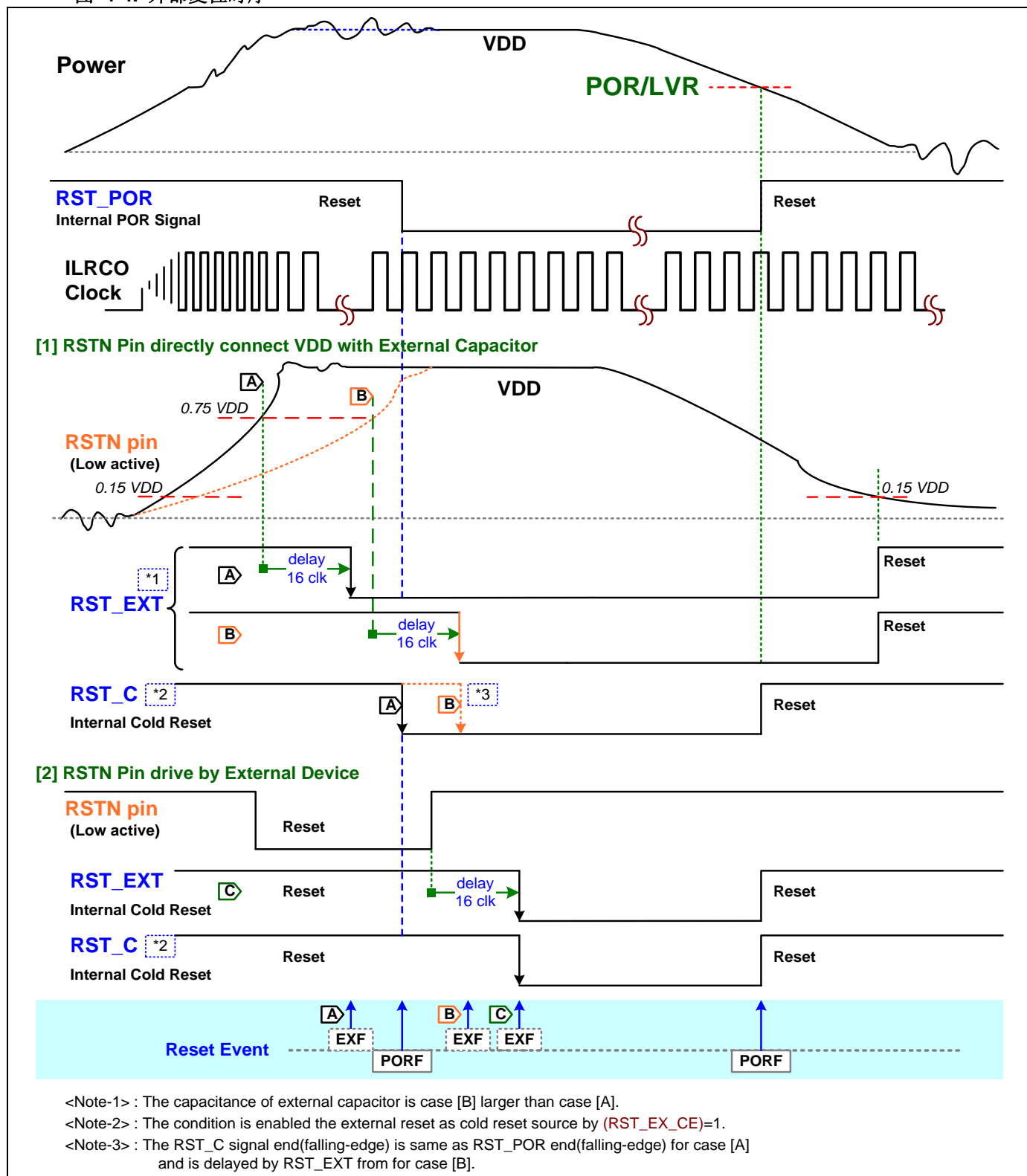
4.4.3. 外部复位时序

该芯片通过保持 **RSTN** 引脚低电平来提供 1 个外部硬件复位输入。**RSTN** 引脚通过硬件配置 **OB** 配置为外部复位引脚或者其他 (GPIO...) 等引脚。为了保证可靠地上电复位，来自 **RSTN** 引脚的硬件复位是必须的。

当 **RSTN** 引脚直接连接 **VDD** 引脚且输入外部复位信号的电压低于 0.15 VDD (**VDD** 引脚上的工作电压) 时，外部复位会被产生。当输入外部复位信号的电压高于 0.75 VDD 时，外部复位会在 16 个 ILRCO 时钟的延时后产生。EXF 标志 (**RST_EXF**) 会在外部复位后被置起。

参照“[外部复位应用电路](#)”节以获取更多信息。

图 4-4. 外部复位时序



4.5. 复位事件

4.5.1. 复位事件源

该芯片支持多种复位源，包括：上电复位、外部复位、软件复位、内存非法地址复位、内存访问错误复位、WDT 复位、MCD 复位、ADC 窗口监测复位、模拟比较器复位和掉电检测复位。用户可以读取复位事件标志识别复位发生时的复位来源。

通常，这些复位事件也可以在相关的模块中作为中断源并可设置为系统中断。参照相关的源模块章节以获取更多信息。

下面的表格展示了冷/热复位事件源和中断兼容性。

[注释]: CMP2/CMP3 复位功能不支持于 MG32F02A032。

表 4-2. 复位事件源

复位源	功能	复位标志	冷复位门	热复位门	兼容中断
POR	上电复位	支持	强制	强制	
BOD0	掉电复位	支持	支持	支持	支持
BOD1	掉电复位	支持	支持	支持	支持
CSC MCD Error	XOSC 时钟丢失检测复位	支持	支持	支持	支持
MEM IA Error	Flash/SRAM 非法地址错误复位	支持	支持	支持	支持
MEM RP Error	Flash/SRAM 读保护错误复位	支持	支持	支持	支持
MEM WP Error	Flash/SRAM 写保护错误复位	支持	支持	支持	支持
IWDT Reset	独立看门狗超时复位	支持	支持	支持	支持
WWDT Reset	窗口看门狗超时复位	支持	支持	支持	支持
ADC Reset	ADC 电压窗口监测复位	支持	支持	支持	支持
CMP0 Reset	模拟比较器电压检测复位	支持	支持	支持	支持
CMP1 Reset	模拟比较器电压检测复位	支持	支持	支持	支持
CMP2 Reset	模拟比较器电压检测复位	支持	支持	支持	支持
CMP3 Reset	模拟比较器电压检测复位	支持	支持	支持	支持
EXT Reset	外部输入复位	支持	支持	支持	
SW Reset	芯片软件复位	支持	支持	支持	
Low Power Mode Reset	进入 STOP 模式时以复位代替进入 STOP 模式	支持	支持	支持	

4.5.2. 复位事件控制

RST 控制器可以设置复位事件是作为冷还是热复位触发，当复位事件发生且相关冷复位或者热复位使能位被使能时，RST 控制器将会置起相关复位事件标志来识别发生复位的来源。

通常，这些复位标志会被硬件置起，被软件通过写 1 清除。参照寄存器描述以获取更多关于相关复位标志和复位使能位的信息。

● 上电复位

PORF (**RST_PORF**) 标志用于识别该复位事件源。上电复位(**POR**)会被内部硬件 RC 和电压检测电路产生，且没有软件使能控制位。

● 外部复位

该芯片提供来自 **RSTN** 引脚的外部硬件复位输入来保证可靠地上电复位。

EXF (**RST_EXF**) 标志用于识别该复位事件源。用户可以通过设置 **RST_EX_CE** 或 **RST_EX_WE** 寄存器使能该复位事件来触发冷或热复位。

● 软件复位

用户可以通过 **RST_SW_EN** 寄存器写 1 的软件复位功能触发芯片复位。

SWF (**RST_SWF**) 标志用于识别该复位事件源。用户可以通过设置 **RST_SW_CE** 或 **RST_SW_WE** 寄存器使能该复位事件来触发冷或热复位。

- 内存读/写保护错误和非法地址复位

当发生内存访问保护错误或者在内部 Flash 或 SRAM 的软件程序跑到了比如超过了编程内存限制的非法的地址时，就会触发复位。

MEMF (**RST_MEMF**) 标志用于识别该复位事件源。用户可以通过设置 **RST_MEM_CE** 或 **RST_MEM_WE** 寄存器使能该复位事件来触发冷或热复位。

- IWDT/WWDT 复位

当 IWDT 或 WWDT 被使能启动计数器，溢出标志会在 IWDT 或 WWDT 溢出时被置起。若 **RST_IWDT_WE** 或 **RST_WWDT_WE** 被使能，IWDT/WWDT 溢出将触发系统热复位；若 **RST_IWDT_CE** 或 **RST_WWDT_CE** 被使能，WDT 溢出将触发硬件冷复位。IWDTF (**RST_IWDTF**) 标志和 WWDTF (**RST_WWDTF**) 标志用于识别这两种复位事件源。

- MCD 复位

当 **XOSC** 时钟或外部输入时钟的时钟丢失检测到丢失事件时，会触发复位。

CSCF (**RST_CSCF**) 标志用于识别该复位事件源。用户可以通过设置 **RST_CSC_CE** 或 **RST_CSC_WE** 寄存器使能该复位事件来触发冷或热复位。

- ADC 电压窗口检测复位

当 ADC 电压窗口检测事件发生时，会触发复位。

ADCF (**RST_ADCF**) 标志用于识别该复位事件源。用户可以通过设置 **RST_ADC_CE** 或 **RST_ADC_WE** 寄存器使能该复位事件来触发冷或热复位。

- 模拟比较器复位

当任何模拟比较器比较事件发生时，会触发复位。

一共 4 个 CMPnF (**RST_CMPnF**) 标志用于识别独立的比较器复位事件源 CMPn (n= {0, 1, 2, 3})。用户可以通过设置 **RST_CMP_CE** 或 **RST_CMP_WE** 寄存器使能该复位事件来触发冷或热复位。

- 掉电复位

一共 2 个掉电检测器 (BOD0/BOD1) 用于监视芯片电源，若核心电源低于 BOD0 监视电压或 VDD 电源低于 BOD1 监视电压，会触发复位。

BOD0F (**RST_BOD0F**) 标志和 BOD1TF (**RST_BOD1F**) 标志用于识别这两种复位事件源。用户可以通过设置 **RST_BOD0_CE**, **RST_BOD0_WE** 和 **RST_BOD1_CE**, **RST_BOD1_WE** 寄存器使能该复位事件来触发冷或热复位。

- CPU 系统复位

当 CPU SYSRESETREQ 系统复位被软件置位，会触发复位，该位只能通过写 1 置起，写 0 无效。

CPUF (**RST_CPUF**) 标志用于识别该复位事件源。用户可以通过设置 **RST_CPU_CE** 或 **RST_CPU_WE** 寄存器使能该复位事件来触发冷或热复位。

- 低功耗模式复位

当芯片进入低功耗 STOP 模式且冷或热复位使能位 **RST_LPM_CE** 或 **RST_LPM_WE** 被设置，芯片会触发复位，而不会进入 STOP 模式。

LPMF (**RST_LPMF**) 标志用于识别该复位事件源。

4.6. 寄存器保护和锁定

除 **RST_STA**, **RST_KEY** 寄存器外，整个 RST 寄存器都处于写保护状态。向 **RST_KEY** 寄存器写值 **0xA217** 解除寄存器保护，且一直都可以保持控制。相对地，写除 **0xA217** 外的值将保护寄存器。读 **RST_KEY** 寄存器的值可得知寄存器处于保护 (=1) 或未保护 (=0) 状态。

在热复位之后还有提供寄存器锁定的功能。写值 **0x712A** 到 **RST_LOCK** 寄存器可以锁定除 **RST_STA**, **RST_KEY** 寄存器外的写权限。当寄存器被锁定，寄存器在被冷复位之前，通过系统热复位事件是不能改变寄存器的。写除 **0x712A** 外的值时没有意义的。读 **RST_LOCK** 寄存器的值可得知寄存器处于锁定 (=1) 或未锁定 (=0) 状态。

该锁定功能可在热复位后用于 RST、RTC、IWDT 模块。其他 CSC、CFG、PW、MEM 和 WWDT 仅支持写权限保护功能。

● 模块冷/热复位对比锁定功能

表 4-3. 模块寄存器复位对比锁定功能

模块	功能	POR 复位	冷复位	热复位		注释
				未锁定	锁定	
RST	清除状态标志	V				只支持从 POR 复位清除
	清除锁定寄存器	V	V			
	冷复位使能寄存器复位至初始值	V				只支持从 POR 复位清除
	热复位使能寄存器复位至初始值	V	V	V		
	复位解锁寄存器/保护寄存器	V	V	V		
CSC	清除状态标志	V	V	V		无锁定功能
	复位解锁寄存器/保护寄存器	V	V	V		
CFG	清除状态标志	V	V	V		无锁定功能
	复位解锁寄存器/保护寄存器	V	V	V		
PW	清除状态标志	V	V	V		无锁定功能
	复位解锁寄存器/保护寄存器	V	V	V		
MEM	清除状态标志	V	V	V		无锁定功能
	复位解锁寄存器/保护寄存器	V	V	V		
RTC	复位锁定寄存器/状态寄存器	V	V			
	复位解锁寄存器/保护寄存器	V	V	V		
IWDT	复位锁定寄存器/状态寄存器	V	V			
	复位解锁寄存器/保护寄存器	V	V	V		
WWDT	清除状态标志	V	V	V		无锁定功能
	复位解锁寄存器/保护寄存器	V	V	V		

V: 表明功能支持 空白: 表明功能不支持

4.7. 模块复位

每个 AHB 或 APB 控制模块，都可以收到来自系统热复位信号来复位模块控制标志、寄存器和逻辑电路。IWDT、WWDT、RTC、PW、CSC 和 MEM 都可以接收冷复位信号解锁被寄存器锁定的功能并复位模块。

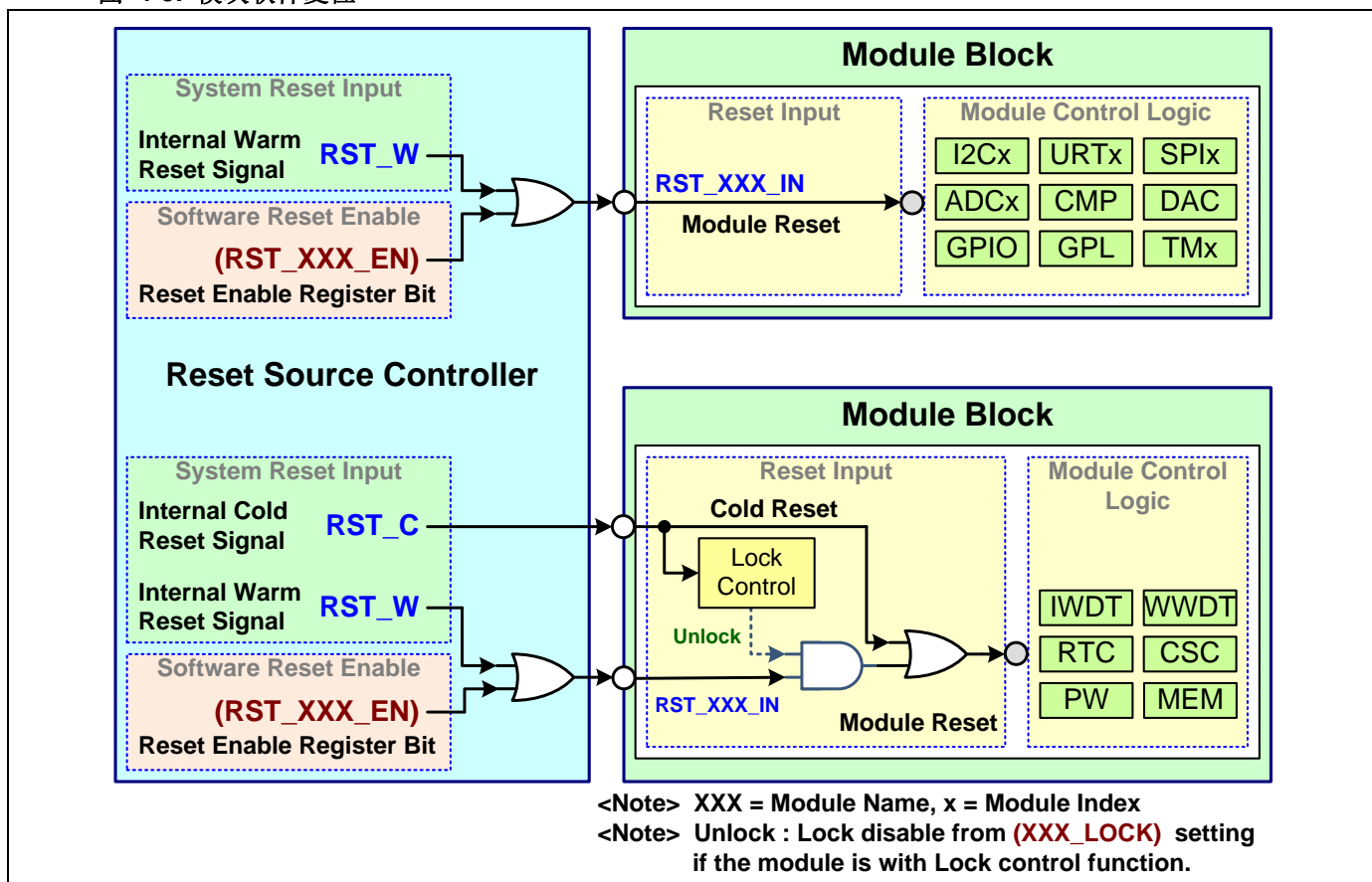
4.7.1. 模块软件复位

每个 AHB 和 APB 控制模块都有 1 个独立的软件复位使能位，用户可以直接设置 **RST_XXX_EN** 寄存器以独立复位相关的模块 (XXX：特定的模块名)。就像模块收到了热复位信号，但是只复位相关的模块。参照寄存器描述以获取更多关于模块软件复位的信息。

下面的图表展示了模块的软件复位使能控制。

[注释]: DAC 模块只支持于 MG32F02A132/072。

图 4-5. 模块软件复位



4.7.2. GPIO 复位控制

该芯片为某些 GPIO 引脚提供了可禁止系统热复位时复位 GPIO 输出数据位和 IO 模式的设置，这些 GPIO 引脚分别是：**PA[3:0]**, **PA[11:8]**, **PB[3:0]**, **PB[11:8]**, **PC[3:0]**, **PC[11:8]**, **PD[3:0]**, **PD[11:8]**, **PE[3:0]**和 **PE[9:8]**。

他们被分成每组 4 根引脚和 1 个独立控制寄存器位的 10 组，用户通过 **RST_PA_DIS0**, **RST_PA_DIS1**, **RST_PB_DIS0**, **RST_PB_DIS1**, **RST_PC_DIS0**, **RST_PC_DIS1**, **RST_PD_DIS0**, **RST_PD_DIS1**, **RST_PE_DIS0** 或 **RST_PE_DIS1** 寄存器为每组设置热复位禁止功能。

下面的图表展示了 GPIO 的数据输出位和 IO 模式的复位控制。

图 4-6. GPIO 复位控制 – MG32F02A132/072

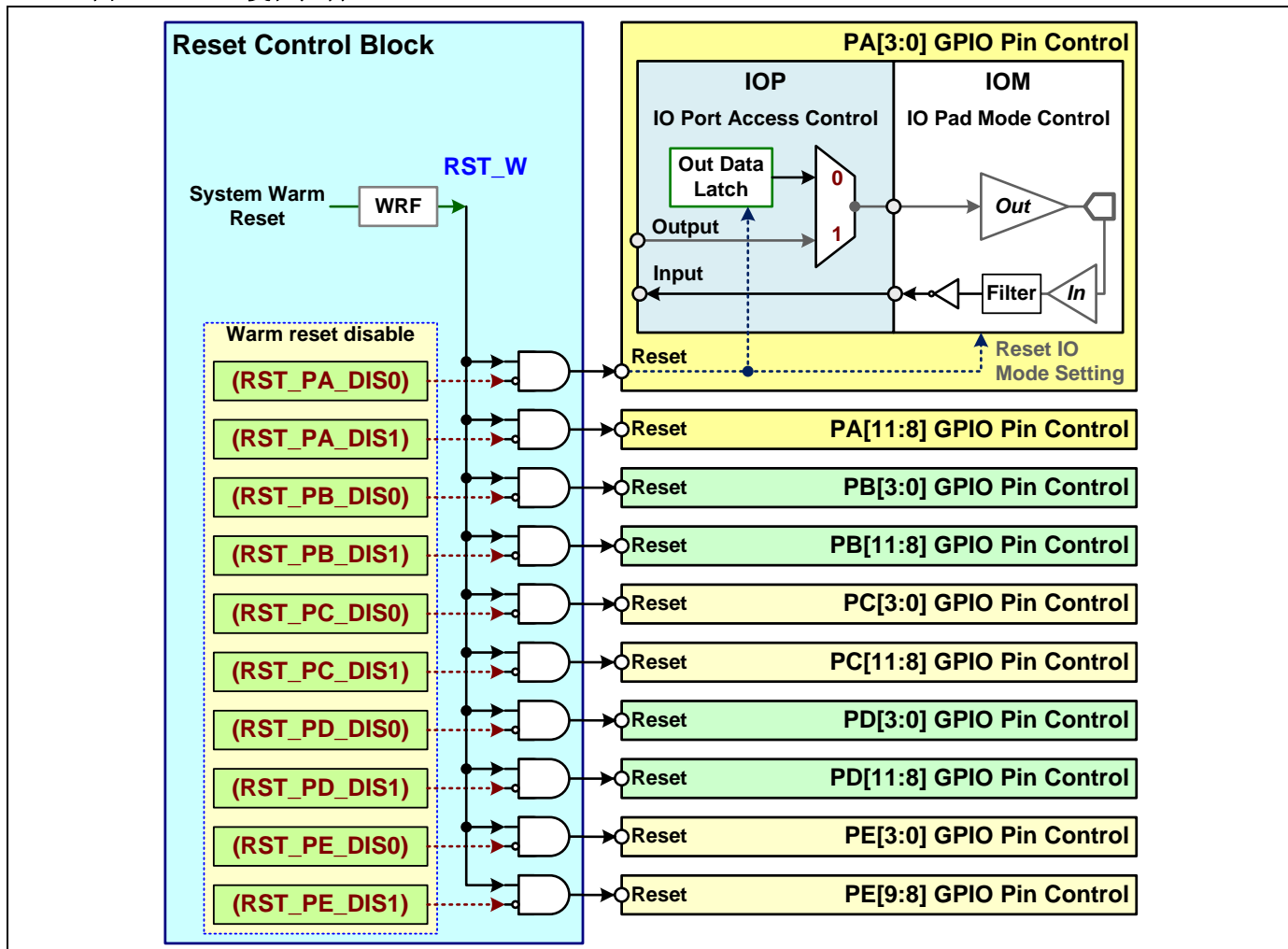
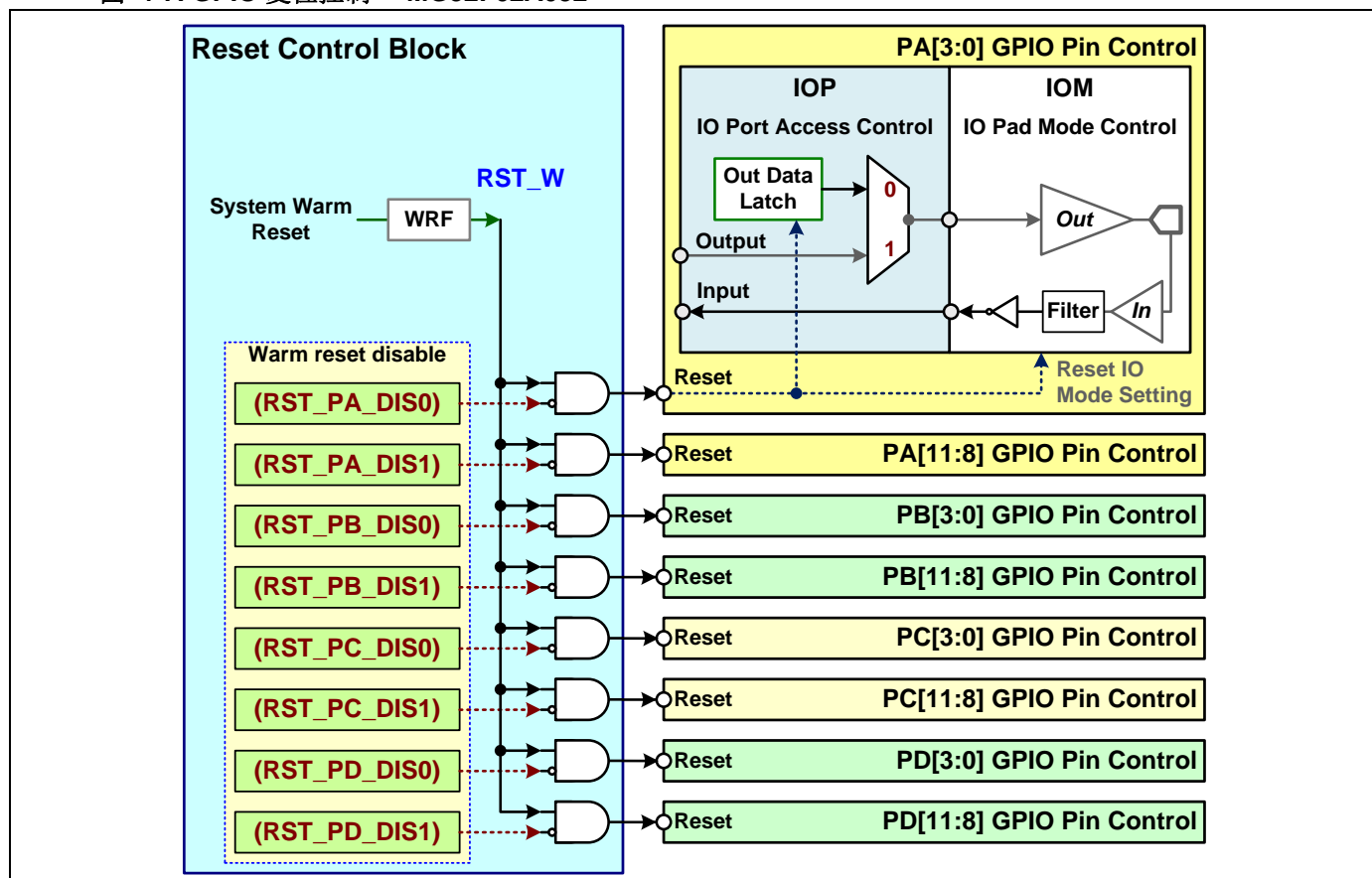


图 4-7. GPIO 复位控制 – MG32F02A032

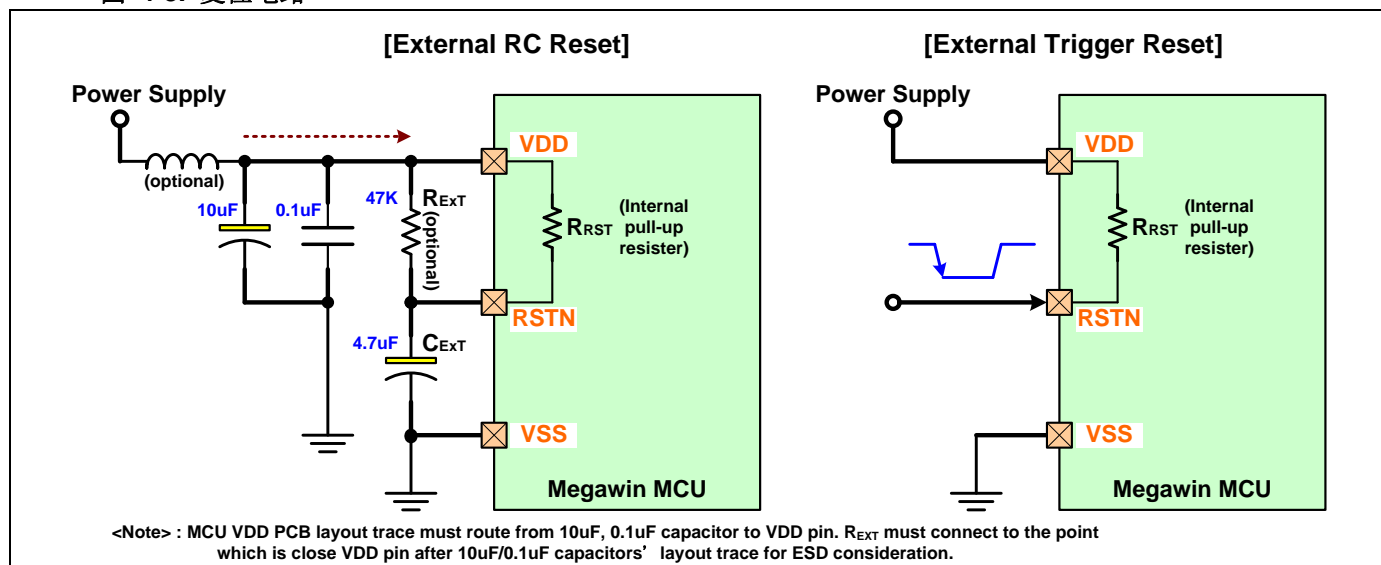


4.8. 外部复位应用电路

通常，上电复位都能在上电过程中成功产生。然而，为了更加保证 MCU 在上电过程中有个可靠的复位，外部复位是很有必要的。下面的图表展示了包含了 1 个连接到 **VSS** (地) 的 C_{EXT} 电容和一个连接到 **VDD** (供电) 的 R_{EXT} 电阻的外部复位电路。

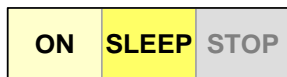
通常， R_{EXT} 是可选的，因为 **RSTN** 引脚包含内部上拉电阻(R_{RST})。该到 **VDD** 的内部电阻允许仅使用 C_{EXT} 到 **VSS** 进行上电复位。

图 4-8. 复位电路



5. 系统时钟

5.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片内置 1 个时钟源控制器 (CSC) 用于系统时钟源管理。在系统应用中, 有四种时钟源: 内部高频 RC 振荡器(**IHRCO**)、内部晶体振荡器(**XOSC**)、内部低频 RC 振荡器(**ILRCO**)、外部时钟输入(**EXTCK**)。

1 个内嵌的 **XOSC** 振荡器被用于外部 Xtal 电路。1 个内嵌 PLL 被用于时钟源倍频和 CPU 输出时钟和其它的外围模块。1 个内嵌的时钟丢失检测器(**MCD**)被用于监视外部 Xtal 或者外部时钟源的时钟。

注释: 标志 (**OB** = 硬件设置字节内存)在该章描述时会被使用。.

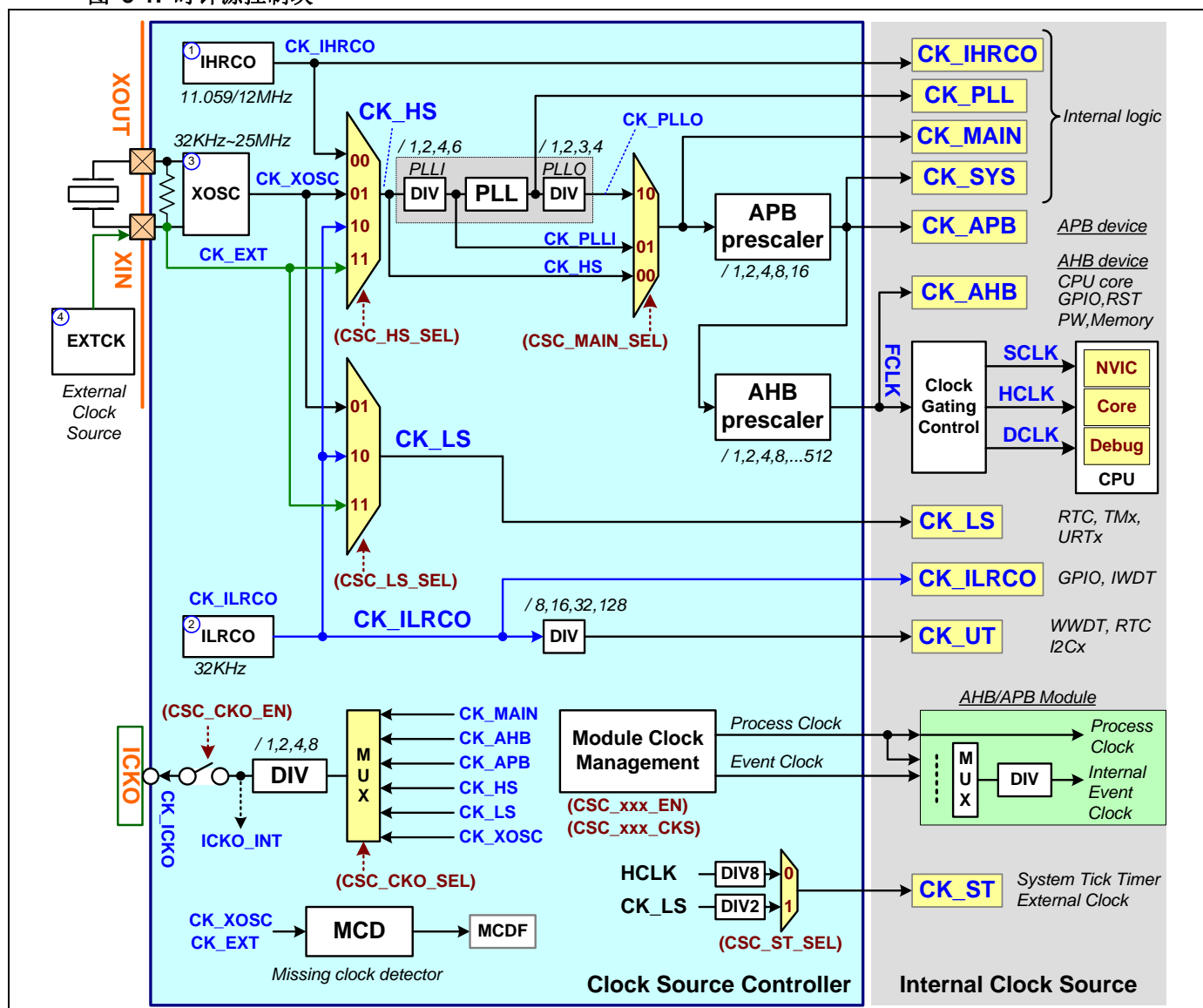
5.2. 芯片时钟特性

- 内嵌 32KHz 的 **ILRCO** (内部低频 RC 振荡器)
- 内嵌 **IHRCO** (内部高频 RC 振荡器)
 - 校准至 11.059 或 12MHz $\pm 1\%$ @ +25°C
- 内嵌最高 48MHz 系统时钟输出的 PLL
- 内嵌支持外部 32KHz 或者 4 到 25MHz Xtal 的带 **MCD** 的 **XOSC** 振荡器
- 支持最高 36MHz 的外部时钟输入
- 内置 1 个用于模块的时钟使能控制的时钟源控制器
- 支持内部 **XOSC** 振荡器和内部 **ILRCO**/**IHRCO** 时钟输出

5.3. 时钟源控制器

下面的图表展示了时钟源控制块。

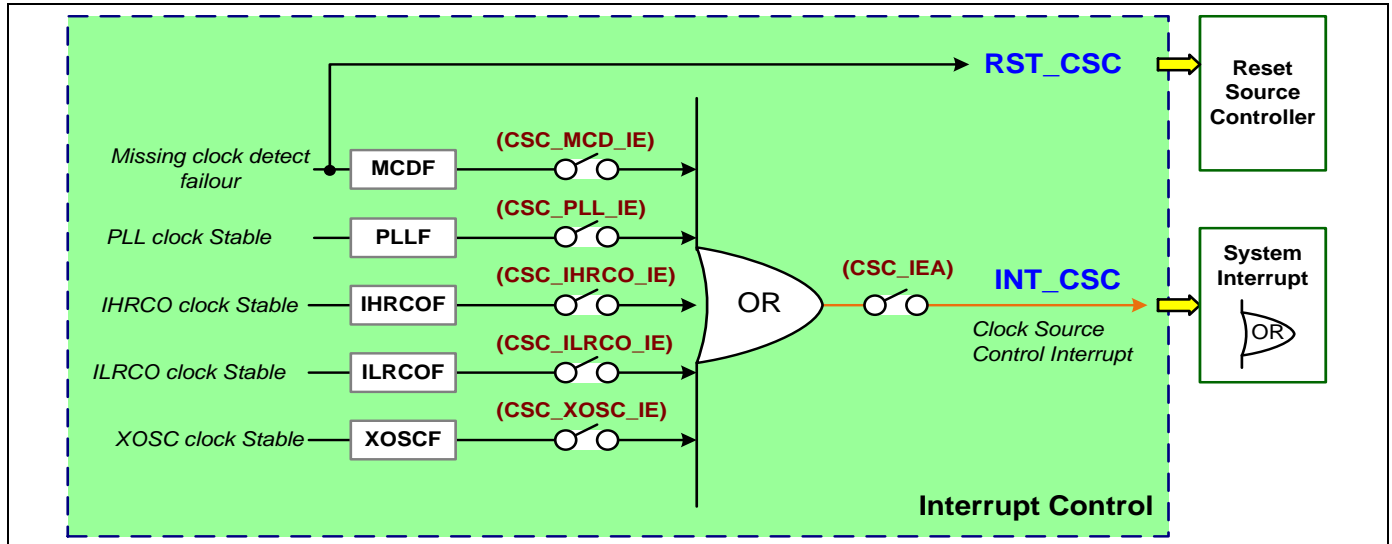
图 5-1. 时钟源控制块



5.4. 中断和复位

在 PW 控制模块中有两种信号会被产生：**INT_CSC**、**RST_CSC**。**INT_CSC** 发送至外部中断控制器（EXIC）作为中断事件；**RST_CSC** 发送至复位源控制器作为复位事件。

图 5-2. 时钟源控制器中断



5.4.1. CSC 中断标志

INT_CSC 是 **XOSC**、**IHRCO**、**ILRCO**、**PLL** 的时钟稳定标志和丢失时钟检测事件的“或”事件标志。这是其中一个作为 **INT_SYS** 系统中断的中断事件。参照系统一般控制章中“[中断和事件](#)”节以获取更多关于系统中断的信息。

这些中断标志是用在中断服务进程（ISR）流控制的。通常这些标志是被硬件置起，在服务工作完成相关 ISR 之后通过软件清除的。每个中断标志都有一个中断使能位，用户可以选择启用或禁用。中断全局使能位 **CSC_IEA** 的使能或者禁用用来控制该模块所有的中断源。

参照寄存器描述以获取更多关于相关中断标志和中断使能位的信息。

- **XOSCF**

XOSC 时钟稳定和就绪检测标志(**CSC_XOSCF**)。相关的中断使能寄存器位是 **CSC_XOSC_IE**。该标志会在 XOSC 振荡器输出时钟稳定时置起。

- **ILRCOF**

ILRCO 时钟稳定和就绪检测标志(**CSC_ILRCOF**)。相关的中断使能寄存器位是 **CSC_ILRCO_IE**。该标志会在 ILRCO RC 振荡器输出时钟稳定时置起。

- **IHRCOF**

IHRCO 时钟稳定和就绪检测标志(**CSC_IHRCOF**)。相关的中断使能寄存器位是 **CSC_IHRCO_IE**。该标志会在 IHRCO RC 振荡器输出时钟稳定时置起。

- **PLL F**

PLL 时钟稳定和就绪检测标志(**CSC_PLL F**)。相关的中断使能寄存器位是 **CSC_PLL_IE**。该标志会在 PLL 输出时钟稳定时置起。

- **MCDF**

XOSC 丢失时钟检测错误事件标志(**CSC_MCDF**)。相关的中断使能寄存器位是 **CSC_MCD_IE**。该标志会在 XOSC 振荡器或外部时钟输入出现时钟丢失时置起。

5.4.2. CSC 时钟状态

有一些只读状态标志代表时钟源状态或时钟切换状态。通常这些标志都是被硬件控制置起和清除的。

- **XOSC_STA**

XOSC 使能后时钟稳定和就绪状态(**CSC_XOSC_STA**)。该标志会在 XOSC 使能后 XOSC 振荡器输出时钟稳定时置起。该标志在 XOSC 振荡器被禁用时无效。

- **ILRCO_STA**

ILRCO 使能后时钟稳定和就绪实时状态(**CSC_ILRCO_STA**)。该标志会在 ILRCO 使能后 ILRCO RC 振荡器输出时钟稳定时置起。该标志在 ILRCO 振荡器被禁用时无效。

- **IHRCO_STA**

IHRCO 使能后时钟稳定和就绪实时状态(**CSC_IHRCO_STA**)。该标志会在 IHRCO 使能后 IHRCO RC 振荡器输出时钟稳定时置起。该标志在 IHRCO 振荡器被禁用时无效。

- **PLL_STA**

PLL 使能后时钟稳定和就绪实时状态(**CSC_PLL_STA**)。该标志会在 PLL 使能后内部 PLL 输出时钟稳定时置起。该标志在 PLL 振荡器被禁用时无效。

- **LS_STA**

输入低速时钟源选择多路复用器 (CK_LS) 切换状态(**CSC_LS_STA**)。它会为时钟源多路复用器报告当前切换的时钟源或正在切换的时钟源。

- **HS_STA**

输入高速时钟源选择多路复用器 (CK_HS) 切换状态(**CSC_HS_STA**)。它会为时钟源多路复用器报告当前切换的时钟源或正在切换的时钟源。

- **MAIN_STA**

系统主时钟源选择多路复用器 (CK_MAIN) 切换状态(**CSC_MAIN_STA**)。它会为时钟源多路复用器报告当前切换的时钟源或正在切换的时钟源。

5.4.3. CSC 复位事件

RST_CSC 信号发送到复位源控制器 (RST) 作为热复位事件或冷复位事件。该复位事件可通过设置 RST 的寄存器得以复位芯片。

- **MCD 检测事件**

当丢失时钟错误事件发生时, **RST_CSC** 信号将会被置起, CSC 会发出 MCDF 标志。

5.5. 寄存器保护和锁定

时钟源控制器复位之后, 除 **CSC_STA** 和 **CSC_KEY** 寄存器外所有的 CSC 寄存器都会被设置为写保护状态。向 **CSC_KEY** 寄存器写值 **0xA217** 可解除寄存器保护状态且一直都可以保持控制, 相对的, 向寄存器写除 **0xA217** 外的值会保护寄存器。读取 **CSC_KEY** 寄存器的值可得知寄存器处于保护状态 (=1) 还是未保护状态 (=0)。

参照系统复位章节的“[寄存器保护和锁定](#)”的表格和描述以获取更多信息。

在硬件功能控制中比较特别的, **CSC_IWDT_EN**, **CSC_SLP_IWDT** 和 **CSC_STP_IWDT** 寄存器是被 **IWDT_LOCK** 寄存器锁定。其他的, 寄存器 **CSC_RTC_EN**, **CSC_SLP_RTC** 和 **CSC_STP_RTC** 是被 **RTC_LOCK** 寄存器锁定。参照 IWDT 和 RTC 章中的“[寄存器保护和锁定](#)”节以获取更多信息。

5.6. 系统时钟控制

5.6.1. 系统时钟源

在系统应用中, 有四种时钟源: 内部高频 RC 振荡器(**IHRCO**)、内部晶振(**XOSC**)、内部低频 RC 振荡器(**ILRCO**)、外部时钟输入(**EXTCK**)。软件可以选择其一并立即进行切换, 但是软件在切换之前必须稳定时钟源。

- **IHRCO 和 ILRCO 时钟**

该芯片内嵌 1 个内部高频 RC 振荡器(**IHRCO**) 和 1 个内部低频 RC 振荡器(**ILRCO**)。通过硬件设置字节(**OB**) 设置, 芯片一般会以 **IHRCO** 12-MHz 或 **ILRCO** 32-KHz 启动。

内建的 **IHRCO** 通过软件设置 **CSC_IHRCO_SEL** 寄存器提供两种频率。一种是 12MHz, 另一种是 11.059MHz。 **IHRCO** 的这两种频率都是高精度的。用户可以通过 **CSC_IHRCO_EN** 寄存器启用或禁用 **IHRCO** 电路。

IHRCOF (**CSC_IHRCOF**)标志和 ILRCOF (**CSC_ILRCOF**)标志都是被硬件置起, 标志着 IHRCO 和 ILRCO 已稳定。

- **XOSC 时钟**

该芯片内置 1 个内部晶体振荡器(**XOSC**)用于外部 Xtal 电路。 **XOSC** 振荡器支持 32K-Hz 或 4 到 25-MHz 频率

的外部晶振。在 **XOSC** 模式下，用户需通过 GPIO AFS 寄存器配置 **XOSC** 的 **XIN** 和 **XOUT** 引脚作为外部晶振的输入和输出。同时，用户可以通过应用程序设置 IO AFS 设置把 **XOSC** 晶振引脚作为 GPIO 功能引脚。

当 **XIN** 和 **XOUT** 引脚被设置为外部晶振输入和输出时，内部 **XOSC** 振荡器会被硬件使能，外部晶振会开始震荡。**XOSCF** (**CSC_XOSCF**) 标志会被硬件置起，标志着外部晶振时钟源已稳定，可用软件进行切换时钟源，在切换晶体振荡器作为系统时钟源之前，软件必须轮询这个位。

用户可以通过 **CSC_XOSC_GN** 寄存器配置 **XOSC** 振荡器控制增益用于不同的外部晶振。

该芯片能通过设置硬件设置字节 (**OB**) 来启动使能 **XOSC** 晶振并设置 **XIN** 和 **XOUT** 引脚作为外部晶振输入和输出。

参照“[晶振电路](#)”节以获取更多信息。

● EXTCK 时钟

该芯片支持最高 36MHz 的外部时钟输入。在外部时钟输入模式 **EXTCK** 下，时钟源来自 **XIN** 引脚，用户需通过设置 IO AFS 设置来将引脚设置为外部时钟输入。

● 丢失时钟检测器

芯片内置 1 个丢失时钟检测器 (**MCD**) 用于监视内部带外部晶振的 **XOSC** 振荡器的时钟。**MCD** 默认是被使能的，而用户可以通过设置 **CSC_MCD_DIS** 寄存器禁用它。当被使能且用户通过时钟选择多路复用器中选的 **CK_HS** 或 **CK_LS** 的时钟源是 **XOSC** 时，**MCD** 会启动并监视时钟源状态，当检测到时钟丢失事件时，**CSC** 会自动切换到 **IHRCO** 给 **CK_HS**，切换 **ILRCO** 给 **CK_LS**，并置起 **MCDF** 标志。

[注释]: 在 MG32F02A132/072 中，丢失时钟检测功能只支持于 **CK_HS** 时钟选择多路复用器且不支持于 **CK_LS** 时钟选择多路复用器。

用户可以通过设置 **CSC_MCD_SEL** 寄存器来设置丢失时钟检测时间。

5.6.2. 高速时钟和低速时钟

高速时钟 **CK_HS** 是通过设置 **CSC_HS_SEL** 寄存器选择 **IHRCO**, **ILRCO**, **XOSC** 和 **EXTCK** 这 4 种时钟源之一来获得的。该高速时钟选择器由于硬件设置字节 (**OB**) 的设置，默认选择 **CK_IHRCO** 或 **CK_ILRCO**。**CK_HS** 可以发送至 PLL 作为参考时钟输入或通过设置主时钟选择器直接输出作为系统主时钟 **CK_MAIN**。

低速时钟 **CK_LS** 是通过设置 **CSC_LS_SEL** 寄存器选择 **ILRCO**, **XOSC** 和 **EXTCK** 这 3 种时钟源之一来获得的。该低速时钟选择器默认选择 **CK_ILRCO**。**CK_LS** 可以发送到 RTC 和 TMx 模块作为定时器时钟，用于低速工作。

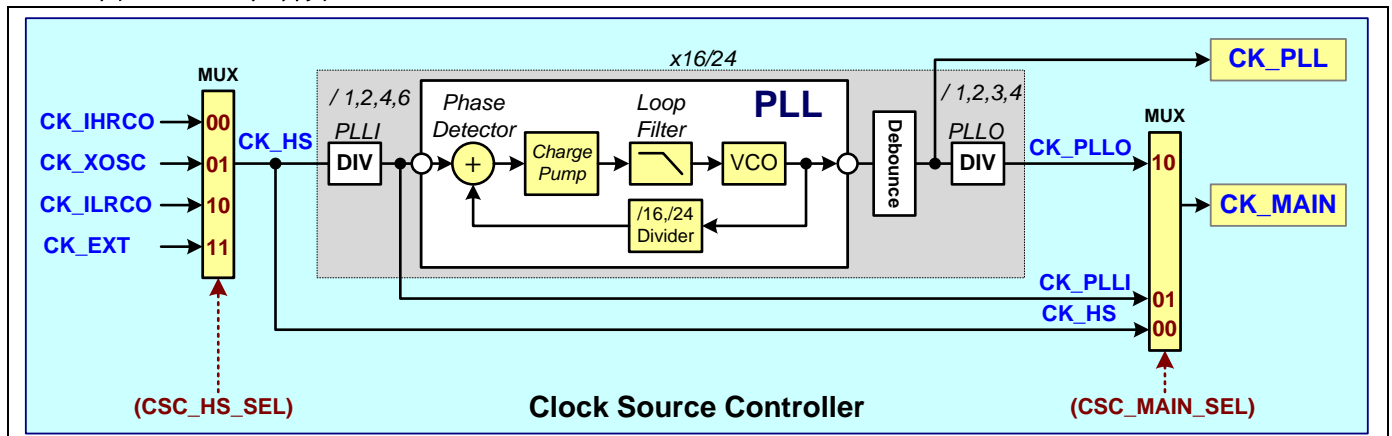
5.6.3. PLL 时钟

内建的 PLL 是通过 **CSC_PLL_EN** 寄存器使能用来倍频高速时钟源 **CK_HS** 的。**CK_HS** 的时钟频率会被 PLL 输入分频器分频，并输出作为 PLL 参考时钟输入 **CK_PLLI**。PLL 输入频率范围是 5~7 MHz，所以用户必须通过 **CSC_PLLI_DIV** 寄存器设定有效的分频值 1、2、4、6 将频率分频。

PLL 能将 PLL 输入时钟频率通过 **CSC_PLL_MUL** 寄存器进行 16 倍或 24 倍频，并输出时钟 **CK_PLL** 最高 96-MHz 或 144-MHz 的频率。由于芯片工作速度的限制，PLL 输出时钟可通过 **CSC_PLLO_DIV** 寄存器分 1、2、3、4 分频。输出时钟 **CK_PLLO** 可通过主时钟选择器选择作为其中一种系统主时钟 **CK_MAIN**。

下面的图表展示了 PLL 控制块。

图 5-3. PLL 控制块



5.6.4. 内部系统时钟

- 系统主时

系统主时 **CK_MAIN** 是通过 **CK_HS** 通过经过 PLL 时钟分频器、PLL 和系统主时选择器得到的。系统主时 **CK_MAIN** 可通过 **CSC_MAIN_SEL** 寄存器选择 **CK_HS**, **CK_PLLI** 和 **CK_PLLO** 三种时钟源。该时钟是用于内部控制逻辑和作为 AHB、APB 外设模块的时钟源的。

用户可以配置 PLL 输入分频器和输出分频控制位获得想要的系统主时时钟，系统主时默认是 **CK_HS** 并绕过 PLL 分频器的。

- AHB 时钟

CK_AHB 时钟提供 AHB 设备的时钟，并可通过 **CSC_AHB_DIV** 寄存器从 APB 时钟 **CK_APB** 进行 1、2、4、8、16、32、64、128、256、512 预分频。

- APB 时钟

CK_APB 时钟提供了 APB 设备的时钟，并可通过 **CSC_APB_DIV** 寄存器从系统主时时钟 **CK_MAIN** 进行 1、2、4、8、16 预分频。该时钟还可作为 **CK_SYS** 系统时钟用于内部逻辑。

- 单位时钟

CK_UT 提供了约 1ms 的单位时间用于 WWDT、RTD、I2C 等内部模块。它可以通过 **CSC_UT_DIV** 寄存器来将 **CK_ILRCO** 时钟经过单位时钟分频器进行 8、16、32、128 分频。

- 系统嘀嗒时钟

ARM CPU 内置 1 个 24 位向下计数系统嘀嗒定时器(**SysTick**)，若 CPU 寄存器 **SYST_CSR** 的 **CLKSOURCE** 控制位被设置，则该时钟 **CK_ST** 提供外部参考时钟输入用于系统嘀嗒定时器。通过设置 **CSC_ST_SEL** 寄存器，可以从 CPU 时钟 **HCLK** 进行 8 分频或 **CK_LS** 进行 2 分频中选择时钟源。

5.7. 模块进程时钟控制

CSC 模块能进行进程时钟使能设置和选择用于内部模块的进程时钟源。用户必须在配置模块正常使用之前选择模块进程时钟并使能。

5.7.1. 模块进程时钟选择

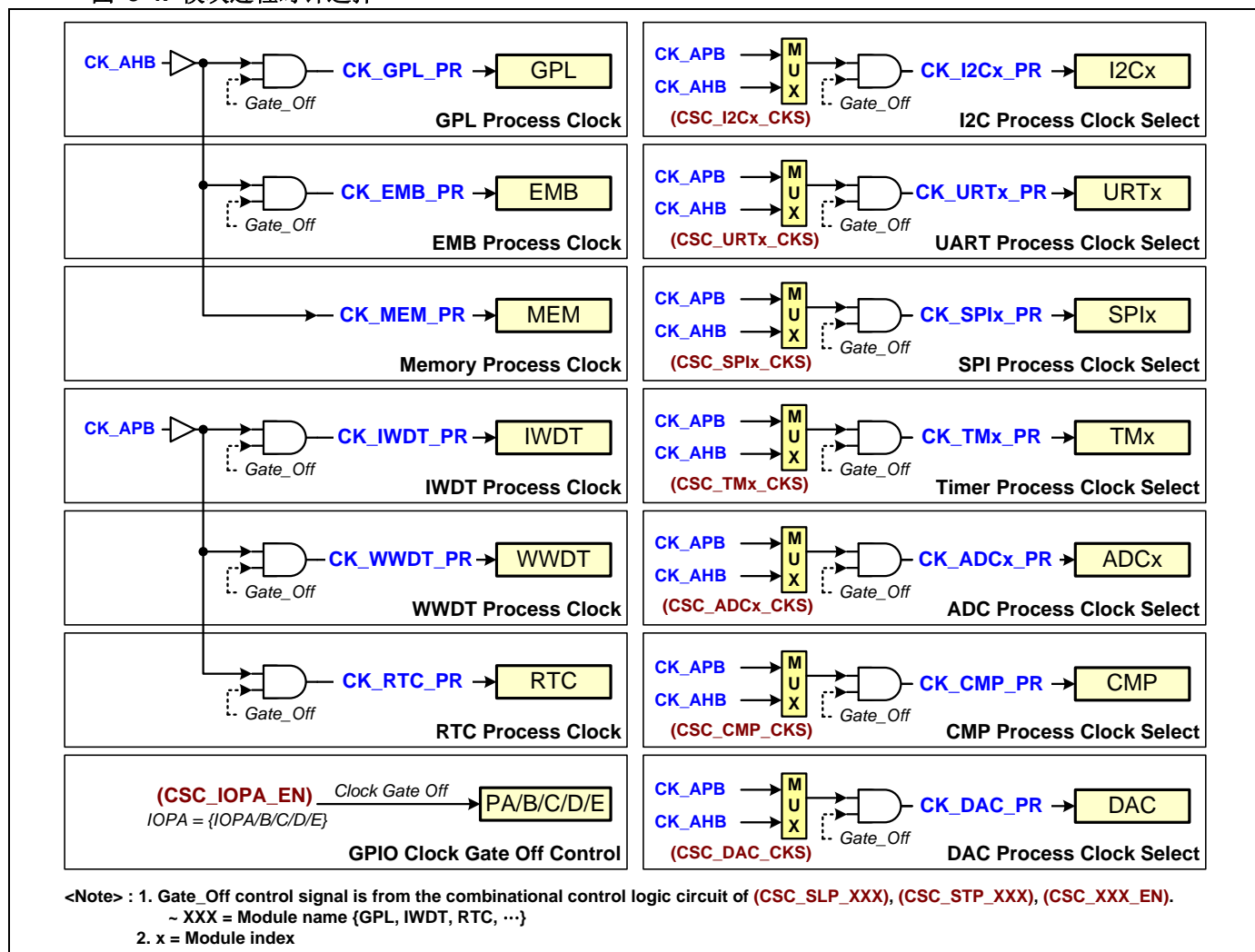
用户可以选择进程时钟源来自 AHB 时钟(**CK_AHB**)或 APB 时钟(**CK_APB**)。用户可以通过 **CSC_xxx_CKS** 寄存器为每个模块独立地选择模块进程时钟(yyy :内部模块名，如 IWDT, RTC,和 CMP ...)。

参照寄存器描述以获取更多关于模块进程时钟选择位的信息。

下面的图表展示了模块的进程时钟选择。

[注释]: DAC 模块只支持于 MG32F02A132/072。

图 5-4. 模块进程时钟选择



5.7.2. 模块进程时钟使能

用户可以通过 **CSC_XXX_EN** 寄存器为每个模块独立地使能模块进程时钟。

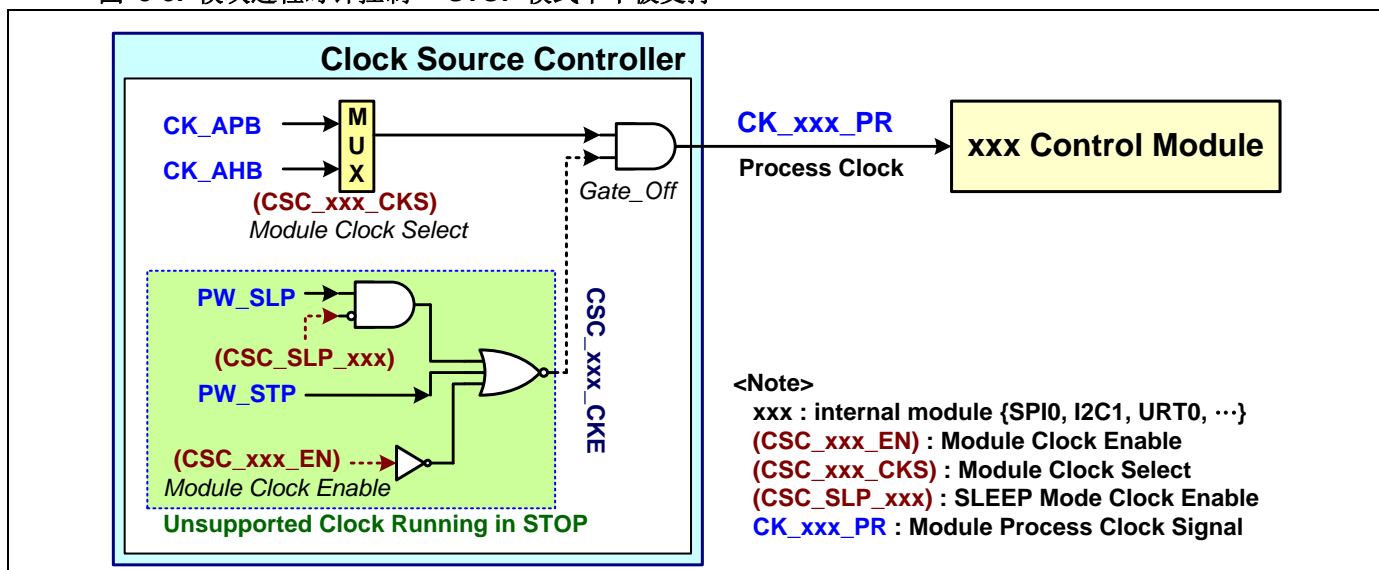
该芯片还支持用于 **SLEEP** 和 **STOP** 模式的时钟使能预设功能。用户可通过设置 **CSC_SLP_XXX** 和 **CSC_STP_XXX** 寄存器，在进入 **SLEEP** 和 **STOP** 模式之前规划进程时钟继续在 **SLEEP** 和 **STOP** 模式中运行的功能。只有 IWDT 和 RTC 支持在 **STOP** 模式下继续工作的时钟预设。（xxx: 内部模块名，如 IWDT, RTC, 和 CMP ...）。

参照寄存器描述以获取更多关于进程时钟使能和 **SLEEP / STOP** 模式时钟使能位的信息。

- **在 STOP 模式下未被支持的时钟**

下面的图表展示了不支持在 **STOP** 模式下运行的时钟的进程时钟使能控制。

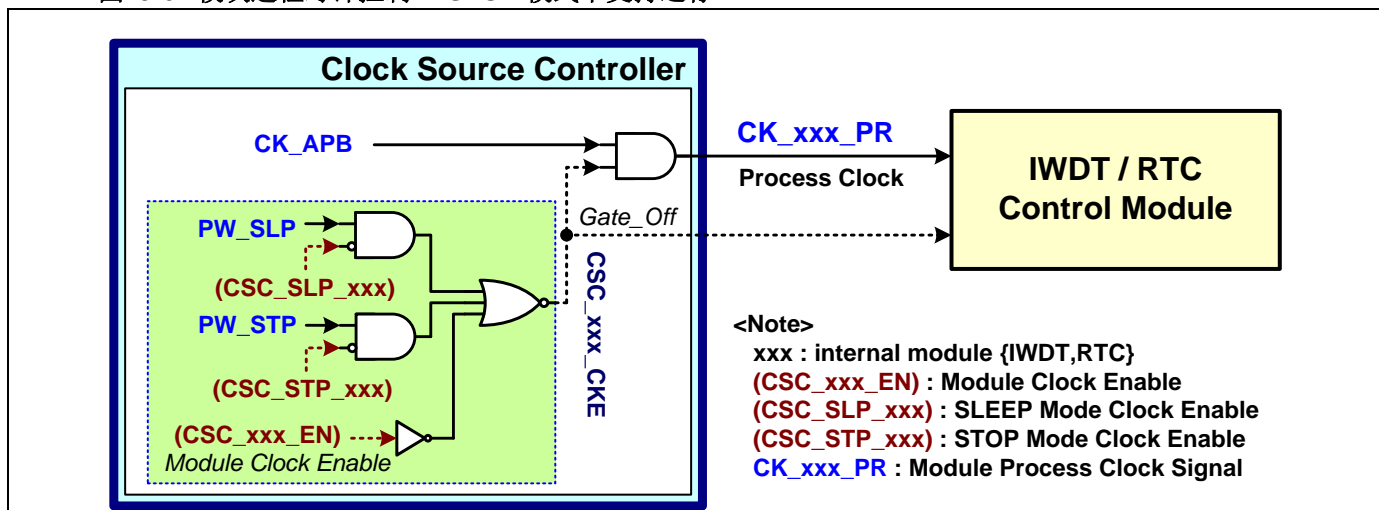
图 5-5. 模块进程时钟控制 – STOP 模式下不被支持



- STOP 模式下支持运行

下面的图表展示了 IWDT、RTC 两个支持在 STOP 模式下运行的进程时钟使能控制。

图 5-6. 模块进程时钟控制 – STOP 模式下支持运行



下面的表格展示了通过寄存器设定时钟使能控制后在 **ON/SLEEP/STOP** 模式下的工作情况。

[注释]: EMB, DAC, TM2x, I2C1 和 URT2/3 模块只支持于 MG32F02A132/072。

表 5-1. 模块可用时钟使能寄存器

模块名	寄存器		
	CSC_xxx_EN	CSC_SLP_xxx	CSC_STP_xxx
EMB	V	V	
Px	V		
GPL	V		
DMA	V		
ADC0	V	V	
CMP	V	V	
DAC	V	V	
IWDT	V	V	V
WWDT	V	V	
RTC	V	V	V
TM0x	V	V	
TM1x	V	V	
TM2x	V	V	
TM3x	V	V	
I2C0	V	V	
I2C1	V	V	
SPI0	V	V	
URT0/1/2/3	V	V	

注释

Px : GPIO 端口 = {PA,PB,PC,PD,PE}

5.7.3. 掉电模式下设备时钟使能控制

CSC 控制器支持 **SLEEP** 或 **STOP** 模式下的内部设备电源控制规划。用户可以在进入 **SLEEP** 或 **STOP** 模式之前为每个独立模块开启或关闭。

下面的表格展示了不同工作模式下内部设备的电源控制。

表 5-2. 内部设备时钟控制

内部设备	ON	SLEEP	STOP
ARM32 Cortex-M0	on	off	off
系统核心时钟 (AHB,APB)	on	on	off
SWD/调试	CSC 寄存器设置	CSC 寄存器预设置	CSC 寄存器预设置
外部 INT 引脚	CSC 寄存器设置	中断使能设置	中断使能设置
模拟比较器	CSC 寄存器设置	CSC 寄存器预设置	CSC 寄存器预设置
RTC	CSC 寄存器设置	CSC 寄存器预设置	CSC 寄存器预设置
IWDT	CSC 寄存器设置	CSC 寄存器预设置	CSC 寄存器预设置
I2Cx	CSC 寄存器设置	CSC 寄存器预设置	off (从机地址检测使用外部 SCL 时钟)
其他模块	CSC 寄存器设置	CSC 寄存器预设置	off

5.8. 模块事件时钟

外围模块的事件时钟是作为模块内部定时器、时序发生器和相关控制逻辑的时钟源。下面的表格展示了可用输入事件时钟源。

[注释]: EMB, DAC, TM2x, I2C1 和 URT2/3 模块只支持于 MG32F02A132/072。

表 5-3. 模块可用输入事件时钟源表

模块名	高速时钟		低速时钟			定时器输出作为时钟	
	CK_AHB	CK_APB	CK_LS	CK_ILRCO	CK_UT	TM00_TRGO	TM01_TRG1
SYS	V						
CFG	V						
PW	V						
RST	V						
MEM	V						
EMB	V						
IOP	V						
Px	V			V		V	
AFS	V						
GPL	V						
DMA		V					
APB		V					
EXIC		V					
ADC0	V	V				V	V
CMP	V	V					
DAC		V					
IWDT				V			
WWDT		V			V		
RTC		V	V		V		V
TM0x	V	V	V				
TM1x	V	V	V				
TM2x	V	V	V				
TM3x	V	V	V				
I2C0/1	V	V			TMO	V	
SPI0	V	V				V	
URT0/1/2/3	V	V	V			V	

注释 Px: GPIO 端口 = {PA,PB,PC,PD,PE}; 所选时钟仅用于过滤
 CK_LS: 来自时钟源{CK_XOSC, CK_ILRCO 或 CK_EXT}
 TMO: 使用 CK_UT 作为 I2C 超时定时器时钟源

5.9. 内部时钟输出控制

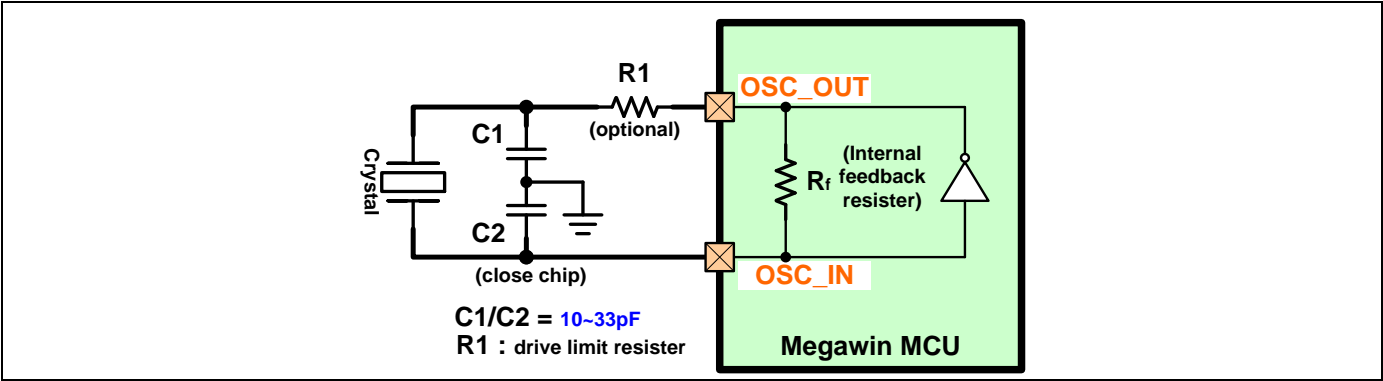
CSC 模块可输出 CK_MAIN, CK_AHB, CK_APB, CK_HS, CK_LS 和 CK_XOSC 内部时钟到外部引脚 ICKO。用户可以通过寄存器 CSC_CKO_EN 使能输出功能并通过寄存器 CSC_CKO_SEL 选择输出时钟源。

由于引脚 ICKO 的速度限制，用户可以将选择的时钟源通过 CSC_CKO_DIV 寄存器经过内部时钟分频器进行 1、2、4、8 分频。

5.10. 晶振电路

为了获得完好的震荡（最大 24MHz），电容 **C1** 和 **C2** 是必要的，如下图所示。通常，**C1** 和 **C2** 选择一样的值。

图 5-7. 晶振电路



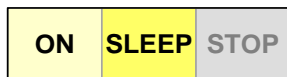
下表是在不同频率下 **C1** & **C2** 的建议值,参照 Xtal 制造规范中的电容负载值确定 **C1** & **C2** 的最终匹配电容器。

表 5-4. C1 & C2 晶振电路参考电容

晶振	C1, C2 电容值
12MHz ~ 25MHz	15pF (12~20pF)
4MHz ~ 12MHz	20pF (15~33pF)
32KHz	10pF (7~12pF)

6. 系统普通控制

6.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片内建 1 个系统控制模块（SYS）用于系统普通控制。包含了 1 个系统事件全局中断使能控制和芯片制造识别码。

6.2. 特性

- 用于系统中断源的系统中断全局使能控制
- 芯片制造识别码- 设备 ID, 产品 ID, 用户 ID, 模块配置

❖ 仅 MG32F02A032

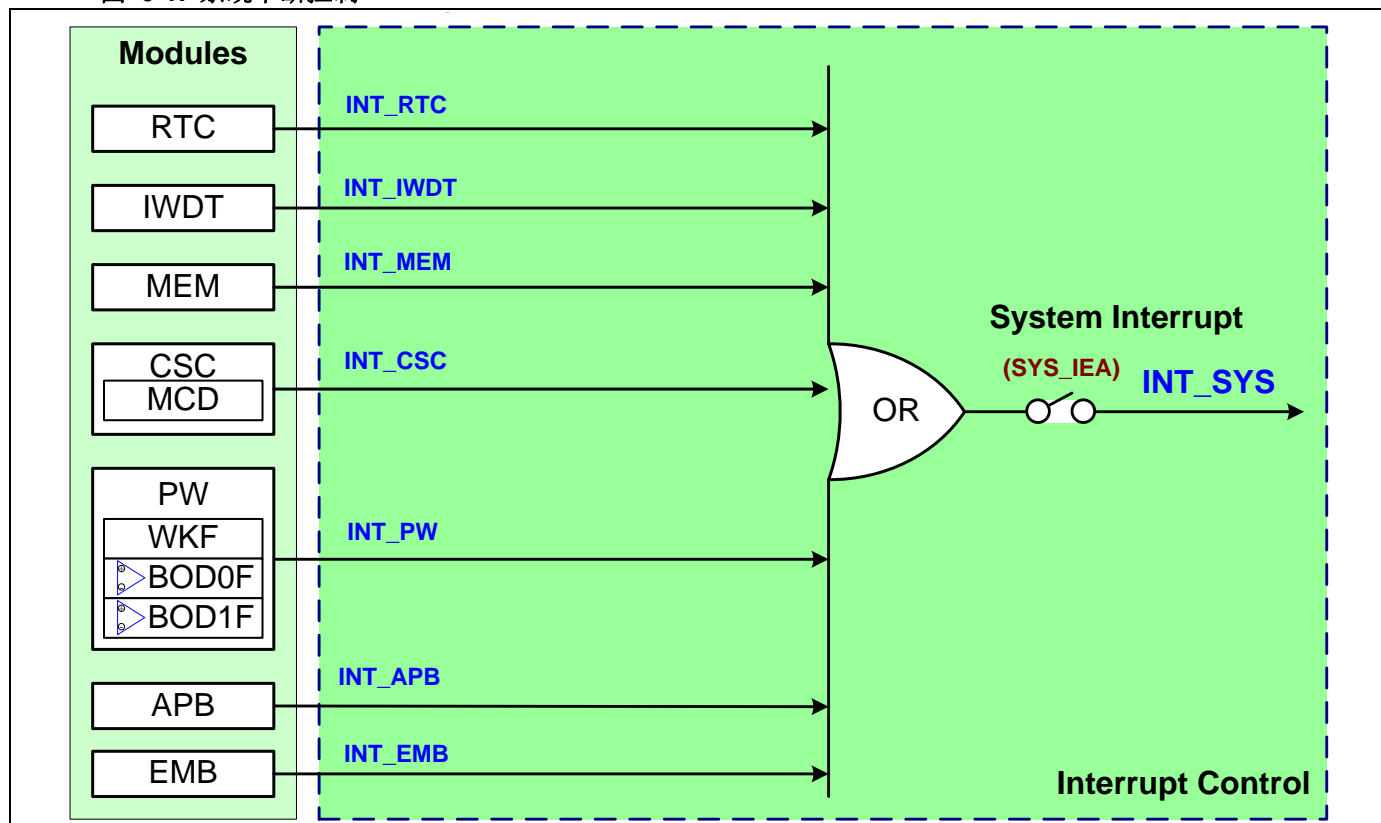
- 32 位不复位备份寄存器

6.3. 中断和事件

系统中断全局使能控制来自 RTC、MEM、CSC、PW、IWDT、APB、EMB 模块的中断事件。系统中断信号 **INT_SYS** 被发送至外部中断控制器(EXIC)作为中断事件。**SYS_IEA** 是系统中断全局使能寄存器。

[注释]: EMB 模块只支持于 MG32F02A132/072。

图 6-1. 系统中断控制



6.4. 芯片制造 ID

用户可以通过读取 **SYS_MID** 寄存器获取设备 ID, 产品 ID, 用户 ID, 模块配置。制造 ID 代码仅供笙泉内部使用。

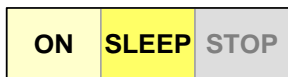
6.5. 系统备份寄存器

该 32 位寄存器用于应用程序固件, 且没有任何硬件控制。它可被读写, 但不会被 POR 或其他冷/热复位进行复位。

[注释]: 该寄存器不支持于 MG32F02A132/072。

7. 系统内存

7.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片把地址空间分割为程序和数据存储内存。程序与数据的逻辑分割允许内存以 32 位进行访问，从而使 CPU 能够快速存储和操作。该芯片支持 1 个内存控制器（MEM）来管理内部闪存和 SRAM 的访问工作。

7.2. 特性

7.2.1. 内嵌内存

❖ MG32F02A132

- 内嵌 132K 字节闪存用于应用代码
- 内嵌 16K 字节 SRAM
 - 支持 DMA 独占 2K 字节，另外 14K 字节用于程序，使数据访问表现更好

❖ MG32F02A072

- 内嵌 72K 字节闪存用于应用代码
- 内嵌 8K 字节 SRAM

❖ MG32F02A032

- 内嵌 32K 字节闪存用于应用代码
- 内嵌 4K 字节 SRAM

7.2.2. 内存控制器

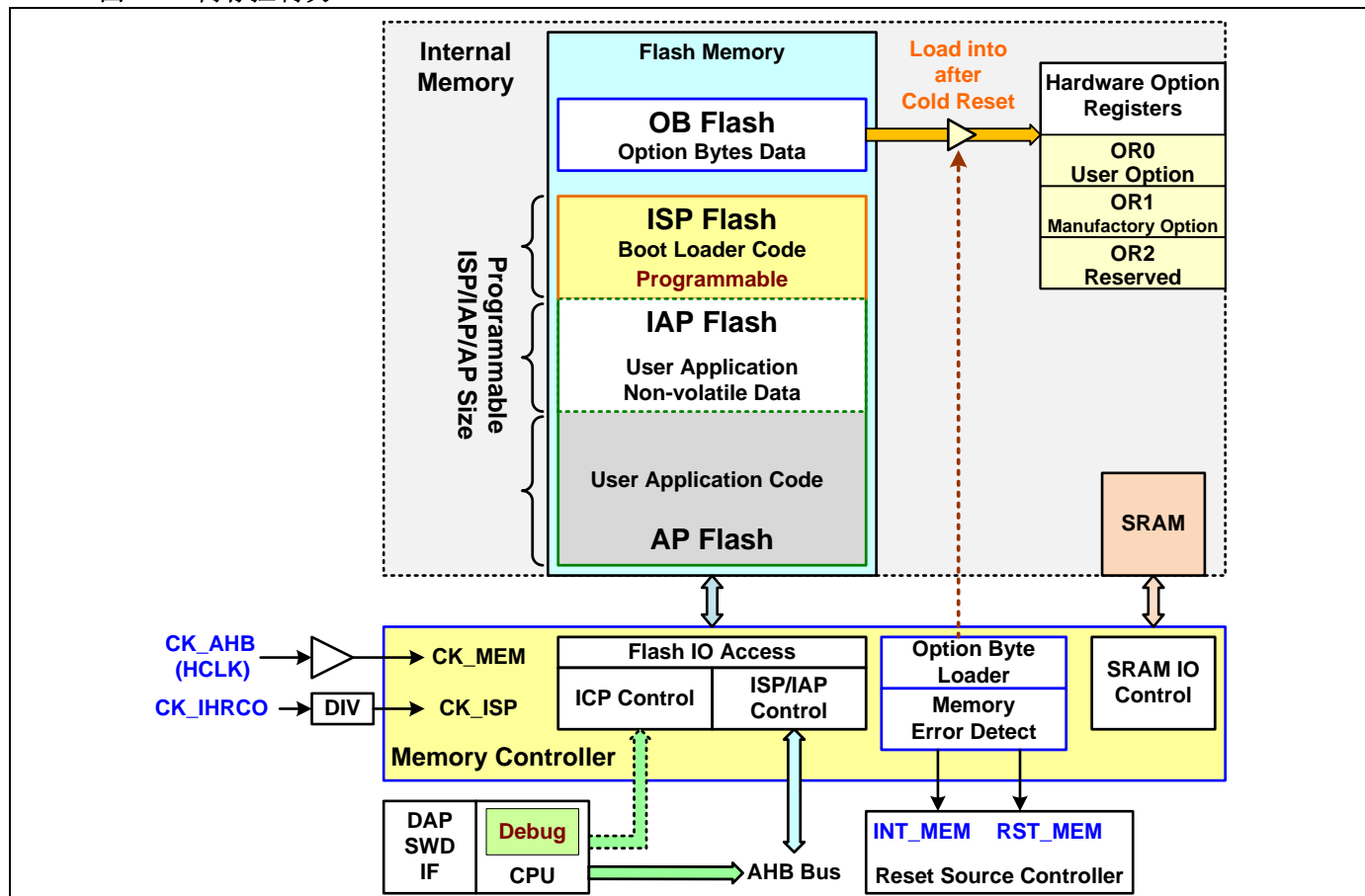
- 支持 ICP（在电路编程）通过 SWD 接口升级 ISP 引导码
- 支持使用 ISP（在系统编程）更新应用程序代码
 - 支持可设置 ISP 启动代码的 ISP 内存大小
 - 提供固定 1K 字节 ISPD 作为 ISP 专用数据空间
- 支持使用 IAP（在应用编程）更新应用程序代码
 - 支持可设置 IAP 启动代码的内存大小
- 支持以 1K 字节为单位进行页擦除

7.3. 内存控制器

内存控制器被支持访问片上内存和在 AHB 总线的 SRAM。内存控制器包括了用于内存访问的 **ICP** (在电路编程)/ **ISP** (在系统编程)/ **IAP** (在应用编程) 电路、用于硬件选项寄存器加载的选项字节加载器和具有访问外部程序存储器的能力的外部存储器总线 EMB 接口。

下面的图表展示了内存控制块。

图 7-1. 内存控制块



7.4. 使能和时钟

该模块的所有功能的全局使能位是 **MEM_EN**。当该位被禁用，所有的 **MEM** 功能均不能工作。

该模块进程时钟用于 **AHB** 和 **APB** 总线的接口逻辑控制和内存控制器。该时钟一直来源于 **CSC**（时钟源控制器）模块，且没有关闭控制寄存器。

7.5. 中断和事件

在 **MEM** 控制模块中有两种信号会被产生：**INT_MEM** 和 **RST_MEM**。**INT_MEM** 发送至外部中断控制器(**EXIC**)作为中断事件；**RST_MEM** 发送至复位源控制器作为复位事件。

7.5.1. 内存控制器中断控制和复位

● 中断事件

INT_MEM 信号发送至外部中断控制器(**EXIC**) 作为中断事件。这些中断标志是用于中断服务进程(**ISR**)流控制的。通常这些标志是被硬件置起，在服务工作完成相关 **ISR** 之后通过软件清除的。每个中断标志都有一个中断使能位，用户可以选择启用或禁用。中断全局使能位 **MEM_IEA** 的使能或者禁用用来控制该模块所有的中断源。

有一些只读状态位用于提供内部控制状态读取。**MEM_FBUSYF** 占用标志用于标识闪存访问占用状态；**MEM_IAPSEF** 标志用于标识 **IAP** 闪存空间大小设置错误。参照寄存器描述以获取更多与相关状态位有关信息。

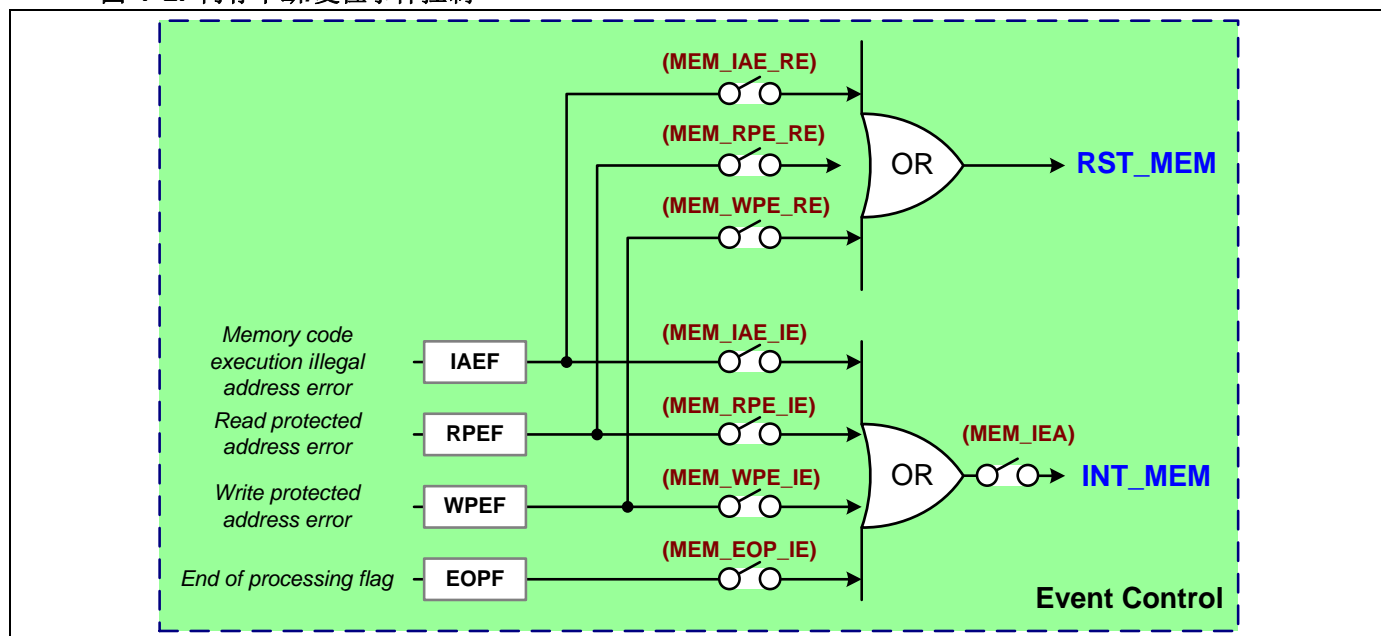
● 复位事件

RST_MEM 信号作为热复位或冷复位信号到复位源控制器(**RST**) 中。这些复位事件可以通过 **RST** 寄存器的设置使能复位芯片。

MEM 可以检测内存代码执行非法地址错误、闪存写保护错误、闪存读保护错误，这些复位事件都有自己的独立复位事件使能位：**MEM_IAE_RE**, **MEM_WPE_RE** 和 **MEM_RPE_RE**。

参照系统复位章的描述以获取更多关于复位事件和控制的信息。

图 7-2. 内存中断/复位事件控制



7.5.2. MEM 中断标志

通常这些中断标志都是被硬件置位，通过软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和中断使能位的信息。

● EOPF

闪存操作结束标志是(**MEM_EOPF**)。相关的中断使能寄存器位是 **MEM_EOP_IE**。该标志用于标识前一次闪存访问指令完成。

● IAEF

内存代码执行非法地址错误检测标志是(**MEM_IAEF**)。相关的中断使能寄存器位是 **MEM_IAE_IE**。

- **WPEF**

闪存写保护错误检测标志是 (**MEM_WPEF**)。相关的中断使能寄存器位是 **MEM_WPE_IE**。当对闪存进行写或擦除操作且操作指令设置、地址区域错误或 **IHRCO** 被禁用时，该标志会被置起。

- **RPEF**

闪存读保护错误检测标志是(**MEM_RPEF**)。相关的中断使能寄存器位是 **MEM_RPE_IE**。当对闪存进行读操作且操作指令设置错误时，该标志会被置起。

7.6. 寄存器保护和锁定

内存控制器复位之后，除 **MEM_STA** 和 **MEM_KEY** 寄存器外所有的 **MEM** 寄存器都会被设置为写保护状态。向 **MEM_KEY** 寄存器写值 **0xA217** 可解除寄存器保护状态且一直都可以保持控制，相对的，向寄存器写除 **0xA217** 外的值会保护寄存器。读取 **MEM_KEY** 寄存器的值可得知寄存器处于保护状态 (=1) 还是未保护状态 (=0)。

特殊的是，**MEM_ISP_WEN** 和 **MEM_ISP_REN** 寄存器是被 **MEM_KEY2** 寄存器保护的，向 **MEM_KEY2** 寄存器写值 **0xA217** 可解除寄存器保护状态且一直都可以保持控制，相对的，向寄存器写除 **0xA217** 外的值会保护寄存器。读取 **MEM_KEY2** 寄存器的值可得知寄存器处于保护状态 (=1) 还是未保护状态 (=0)。

参照系统复位章的“[寄存器保护和锁定](#)”节的表格和描述以获取更多信息。

7.7. 启动和片上内存

7.7.1. 内存启动模式

该芯片能选择从用户程序闪存（AP）、在系统编程（ISP）和内嵌 SRAM 中启动，芯片启动的过程中，硬件设置选项字节(**OB**)会被载入 **MEM_BOOT_MS** 寄存器以选择如下表中的其中一种方式启动。

表 7-1. 内存启动方式选择

启动模式	内存	寄存器
		BOOT_MS
AP	用户程序闪存	0
ISP	ISP 启动闪存	1
SRAM	内置 SRAM	2

<注释> 被选择的启动内存会被硬件重映射到 0x0000 0000。

7.7.2. 片上闪存

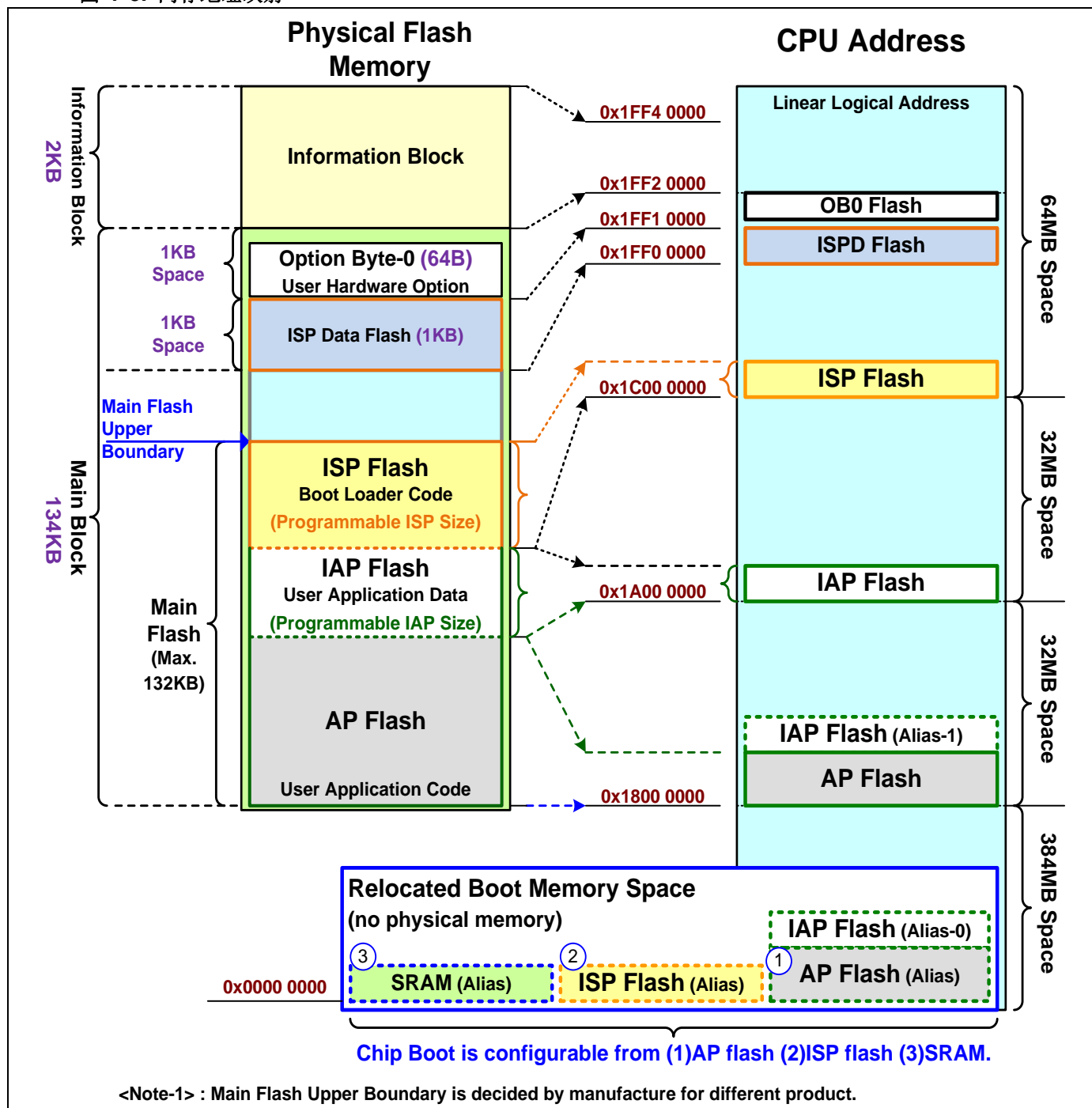
整个块包含了应用程序闪存(**AP** 和 **IAP**)、引导闪存(**ISP**)、用户硬件设置选项字节(**OB**)和 ISP 专用数据闪存(**ISPD**)。 **OB** 闪存和 **ISPD** 闪存会被固定在主块的高 2K 字节区域。该信息块存储了内部芯片和制造的信息。

一共有最多 **132K** 字节的片上内存空间可以用于用户编程，这些空间包含了 **AP**、**IAP** 和 **ISP**。 **ISPD** 内存被专用于 ISP 编程。

这些内存块可以被映射到独立的 **CPU** 地址区域。被重定向的启动内存空间（基地址 0x00000000）由启动设置决定是 **AP**、**ISP** 还是 **SRAM**。所以它实际上不是物理内存。

下面的图表展示了不同的闪存块闪存地址映射，物理闪存内嵌入芯片中。

图 7-3. 闪存地址映射



● 启动闪存

ISP 内存被用于存储 **ISP** (在系统编程)启动引导代码。当 MCU 在 **ISP** 中运行时, MCU 可以进行 **AP** 和 **IAP** 闪存修改以进行软件升级。

— ISP 内存

ISP 内存被定位在基地址 0x1C000000 上。**CFG_ISP_SIZE** 寄存器可以定义整个闪存空间和 **IAP** 闪存空间的大小。**CFG_ISP_SIZE** 寄存器的值需要通过硬件选项字节(**OB**)设定。

当芯片设定为从 **ISP** 启动, **ISP** 闪存会被重定向在基地址 0x00000000 上, 且内容会和物理内存基地址 0x1C000000 的内容一致。

— ISPD 内存

固定的 1K 字节 ISPD 闪存是专用于 **ISP** 功能的(**ISP** 模式启动)。而且, ISPD 可以通过设置 **MEM_ISPD_WEN** 和 **MEM_ISPD_REN** 寄存器用于从 **AP** 或从 **SRAM** 模式启动。**MEM_ISPD_WEN** 只能在 **ISP** 模式下修改;
MEM_ISPD_REN 只能在 **ISP** 模式下设置。在任何模式下可以清除或禁用。

- 程序闪存

编程闪存被分为 **AP** 闪存和 **IAP** 闪存。**AP** 闪存用于存储用户应用程序；**IAP** 闪存用于存储非易失应用程序数据。若 MCU 在 **AP** 区域中运行，软件只能修改 **IAP** 闪存以更新存储数据。

- AP 闪存

应用程序闪存是用于存储 CPU 执行代码的，**AP** 闪存被定位在基地址 0x18000000 上。

当芯片设置为从 AP 启动，**AP** 闪存会被重定向到基地址 0x00000000 上，而内容与物理闪存基地址 0x18000000 上的内容一致。

在复位后且芯片从 AP 启动后，CPU 会开始从复位中断向量地址(0x00000004)开始执行，该地址会是用户程序代码的起始地址。为了使用这些中断，中断服务地址（中断向量）必须被定位于 0x000000C0~0x00000000 上。

- IAP 闪存

IAP 闪存是定位在基地址 0x1A000000 上的。**IAP** 闪存大小可以通过 **MEM_IAP_SIZE** 寄存器定义。用户可以用 1K 字节为单位配置这个寄存器。值为 0 代表 IAP 闪存空间为 0K 字节，值为 1 代表 IAP 闪存空间为 1K 字节。该寄存器在 **MEM_IAP_AEN=0** 时是无法进行写操作的。**MEM_IAP_AEN** 是作为 **MEM_IAP_SIZE** 寄存器访问使能位的。

该芯片支持通过设置 **MEM_IAP_EXEC** 寄存器进行 IAP 闪存代码执行设置。

MEM_IAP_SIZE 寄存器的值可以在上电后或冷复位或软件编程时通过硬件设置字节(**OB**)内存进行配置。

- 设置字节闪存

片上设置字节空间最大能有 64 字节。

设置字节用于存储硬件配置，比如 **CFG_BOOT_MS** 用于保存系统冷复位启动内存选择，**CFG_HS_SEL** 用于保存 **CK_HS** 的默认源。

7.7.3. 片上数据 RAM

内嵌 SRAM 有最大 16K 字节，同时用户也可以选择 SRAM 中运行代码。SRAM 内存被定位于基地址 0x20000000 上。

[注释]: MG32F02A132 内嵌 16K 字节 SRAM。

MG32F02A072 内嵌 8K 字节 SRAM。

MG32F02A032 内嵌 4K 字节 SRAM。

当芯片设置为从 SRAM 启动，SRAM 会被重定向到基地址 0x00000000 上，而内容与物理闪存基地址 0x20000000 上的内容一致。

为了加强 DMA 的数据传输性能，SRAM 被设计分成 2 块：高 2K 字节和低 14K 字节。强烈建议用户将高 2K 字节用于 DMA，而低 14K 字节用于程序使用。

当然，用户可以设置 SRAM 的任何区域作为软件使用或 DMA 传输。这没有任何硬件限制。

7.8. 内存控制器功能

7.8.1. 闪存访问

该芯片含有最大 132K 字节的内嵌闪存空间用于代码和数据、设置用于保存启动码和用于芯片配置的 64 字节闪存。

内存控制器 (MEM) 支持通过 **MEM_MDS** 寄存器设置读取/编程 (写入) /擦除闪存。用户可以通过 CPU 读取指令直接从闪存读取数据而不需通过任何寄存器。对于编程模式，MEM 提供 32 位数据写入操作给内存做新数据的更新。对于擦除模式，对于“擦除”模式，擦除地址仅在低 10 位 CPU 地址=0 (X..X00 0000 000 B) 有效，并寻址 1K 字节对齐。

下面的表格展示了闪存访问控制指令和键值序列。

表 7-2. 闪存访问控制和键值

闪存	操作	模式	寄存器			
			MDS	操作使能	SKEY 1st	SKEY 2nd
AP 闪存	写	单	0x1	AP_WEN=1	0x46	0xB9
		多				0xBE
	擦除	单	0x2			0xB9
		多				0xBE
	读	全部	直接通过内存访问指令读取			
IAP 闪存	写	单	0x1	IAP_WEN=1	0x46	0xB9
		多				0xBE
	擦除	单	0x2			0xB9
		多				0xBE
	读	全部	直接通过内存访问指令读取			
ISPD 闪存	写	单	0x1	ISPD_WEN=1	0x46	0xB9
		多				0xBE
	擦除	单	0x2			0xB9
		多				0xBE
	读	全部	-			ISPD_REN=1
闪存	操作	模式	寄存器			
			MDS	操作使能	SKEY2 1st	SKEY2 2nd
ISP 闪存	写	单	0x5	ISP_WEN=1	0x9867	0xB955
		多				0xBEAA
	擦除	单	0x6			0xB955
		多				0xBEAA
	读	全部	-			ISP_REN=1

<注释-1> 全部: 单或多数据访问

<注释-2> x: 无保护键值需求; -: 不必要

<注释-3> -: 直接通过内存访问指令读取

<注释-4> 向(MEM_SKEY)写任意值以锁定多数据写入功能

<注释-5> SRAM 编程属于 AP 编程

<注释-6> 若(ISP_REN=0), ISP 闪存使能在芯片从 ISP 模式启动时可读取数据

当用户想要修改或删除片上闪存, 首先需要通过 MEM_KEY 寄存器解锁 MEM 寄存器保护并通过 MEM_EN 寄存器使能 MEM 控制器。其次, 用户需要通过 MEM_MDS 寄存器设置闪存访问模式并通过 MEM_AP_WEN, MEM_IAP_WEN, MEM_ISP_WEN 或 MEM_ISPD_WEN 寄存器使能闪存写操作使能位。然后, 用户还需要通过设置 MEM_SKEY 或 MEM_SKEY2 寄存器解锁闪存访问键值序列, 这样, 用户就可以对闪存进行修改和清除了。

访问键值序列 MEM_SKEY 是用于 AP/IAP/ISPD 闪存的; 访问键值序列 MEM_SKEY2 是用于 ISP/OB 闪存的。后面的小节介绍 AP/IAP/ISP/ISPD/OB 的修改或擦除流程。

● AP/IAP 闪存修改/擦除流程

(1) 解锁寄存器保护 ~ (MEM_KEY) = 0xA217

(2) 使能内存控制器 ~ (MEM_EN) = 1

(3) 设置闪存写使能 ~ (MEM_AP_WEN/MEM_IAP_WEN) = 1

设置闪存操作模式 ~ (MEM_MDS) (=1: 写入操作, =2: 擦除操作)

(4) 解锁闪存访问键值序列 ~ (MEM_SKEY) = 0x46

- (5) 单次写入操作或多次写入操作
 - (a) 单次写入使能 ~ (MEM_SKEY) = 0xB9
 - (b) 多次写入使能 ~ (MEM_SKEY) = 0xBE
- (6) 闪存操作 ~ 写入闪存或擦除闪存指令
(直接通过 CPU 往 CPU 地址写入数据触发闪存修改/擦除)
- (7) 若是多次写入则重复 (6) 过程
- (8) 锁定闪存访问 ~ (a) 单次写入 ~ 硬件在单次写入后自动上锁
(b) 多次写入 ~ (MEM_SKEY) = 0x64 (任意值)
- ISP 闪存修改/擦除流程
 - (1) 解锁寄存器保护 ~ (MEM_KEY) = 0xA217 , (MEM_KEY2) = 0xA217
 - (2) 使能内存控制器 ~ (MEM_EN) = 1
 - (3) 设置闪存写使能 ~ (MEM_ISP_WEN/MEM_ISPD_WEN) = 1
设置闪存操作模式 ~ (MEM_MDS) (=5: 写入操作 , =6: 擦除操作)
 - (4) 解锁闪存访问键值序列 ~ (MEM_SKEY2) = 0x9867
 - (5) 单次写入操作或多次写入操作
 - (a) 单次写入使能 ~ (MEM_SKEY2) = 0xB955
 - (b) 多次写入使能 ~ (MEM_SKEY2) = 0xBEAA
 - (6) 闪存操作 ~ 写入闪存或擦除闪存指令
(直接通过 CPU 往 CPU 地址写入数据触发闪存修改/擦除)
 - (7) 若是多次写入则重复 (6) 过程
 - (8) 锁定闪存访问 ~ (a) 单次写入 ~ 硬件在单次写入后自动上锁
(b) 多次写入 ~ (MEM_SKEY2) = 0x1234 (任意值)

对于闪存读操作，用户可以直接通过 CPU 读指令把数据从闪存中读出来。用户根据通过不同的 AHB 时钟频率 **CK_AHB** 设置 **MEM_FWAIT** 寄存器中的读访问等待状态控制。该寄存器选择 AHB 时钟周期到闪存访问时间的延时时间，如下表所示。

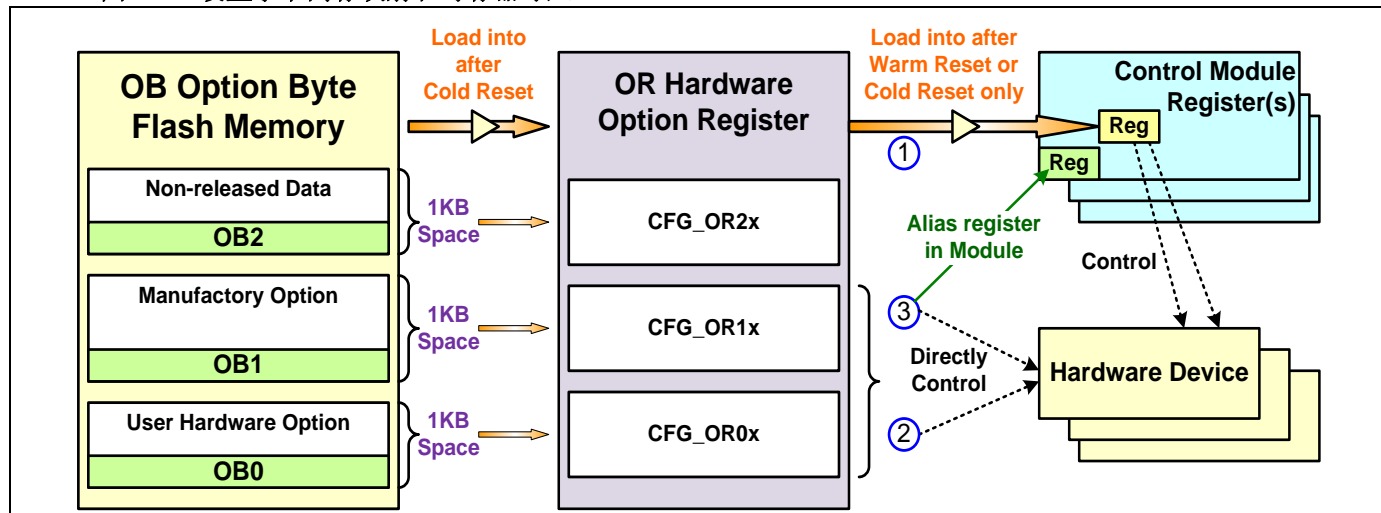
- 0：若 $CK_AHB < 25\text{ MHz}$ 则为 0 等待状态
- 1：若 $25\text{MHz} < CK_AHB < 50\text{ MHz}$ 则为 1 等待状态
- 2：若 $50\text{MHz} < CK_AHB < 75\text{ MHz}$ 则为 2 等待状态

7.8.2. 硬件设置字节闪存

片上能包含最大 64 字节的设置字节闪存，该闪存用于存储硬件设置的配置。

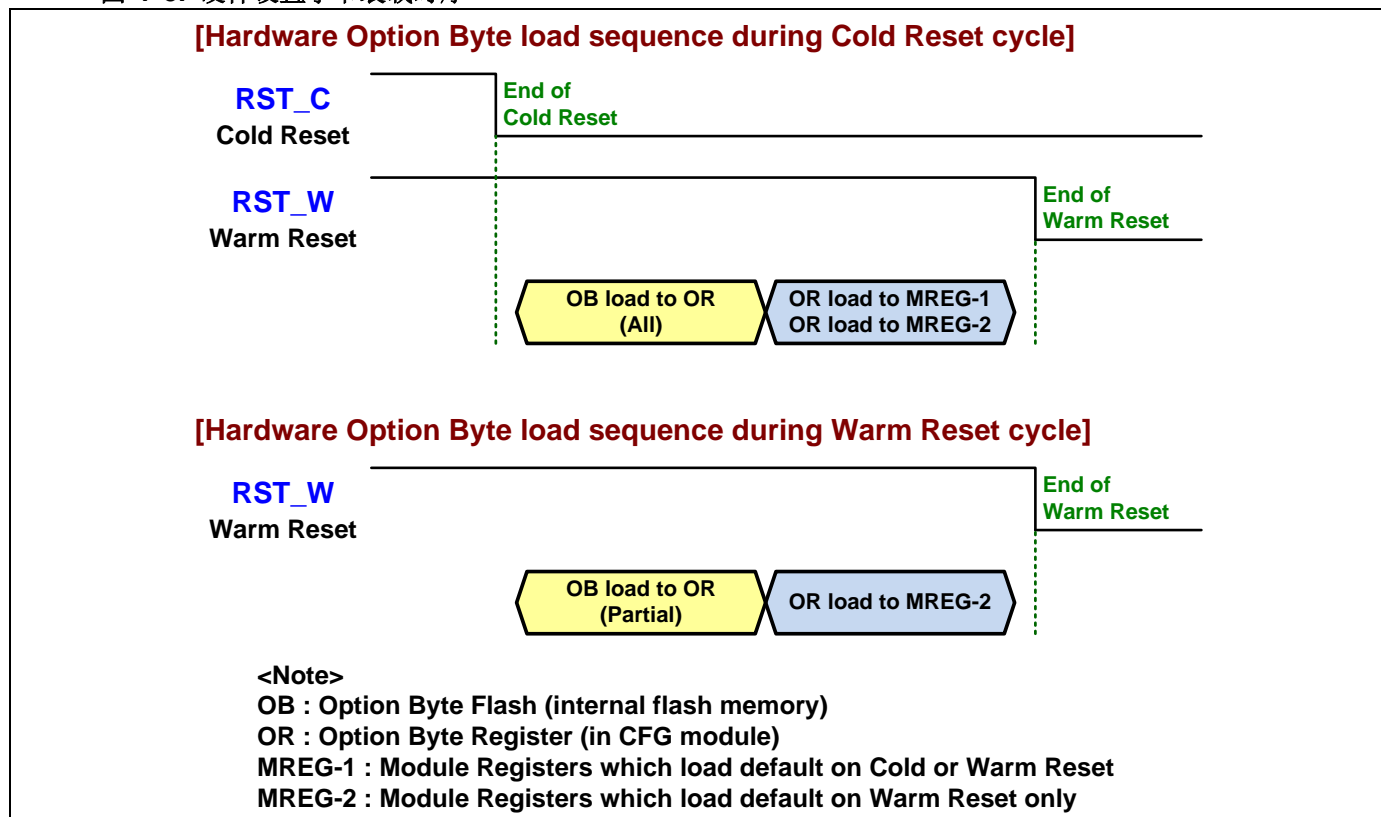
下面的图表展示了在热复位和冷复位过程中设置字节闪存(OB)导入芯片设置寄存器(OR)并载入模块寄存器的整个流程。

图 7-4. 设置字节闪存映射和寄存器导入



下面的图表展示了在热复位或冷复位过程中设置字节闪存(OB)的读取时序。

图 7-5. 硬件设置字节装载时序

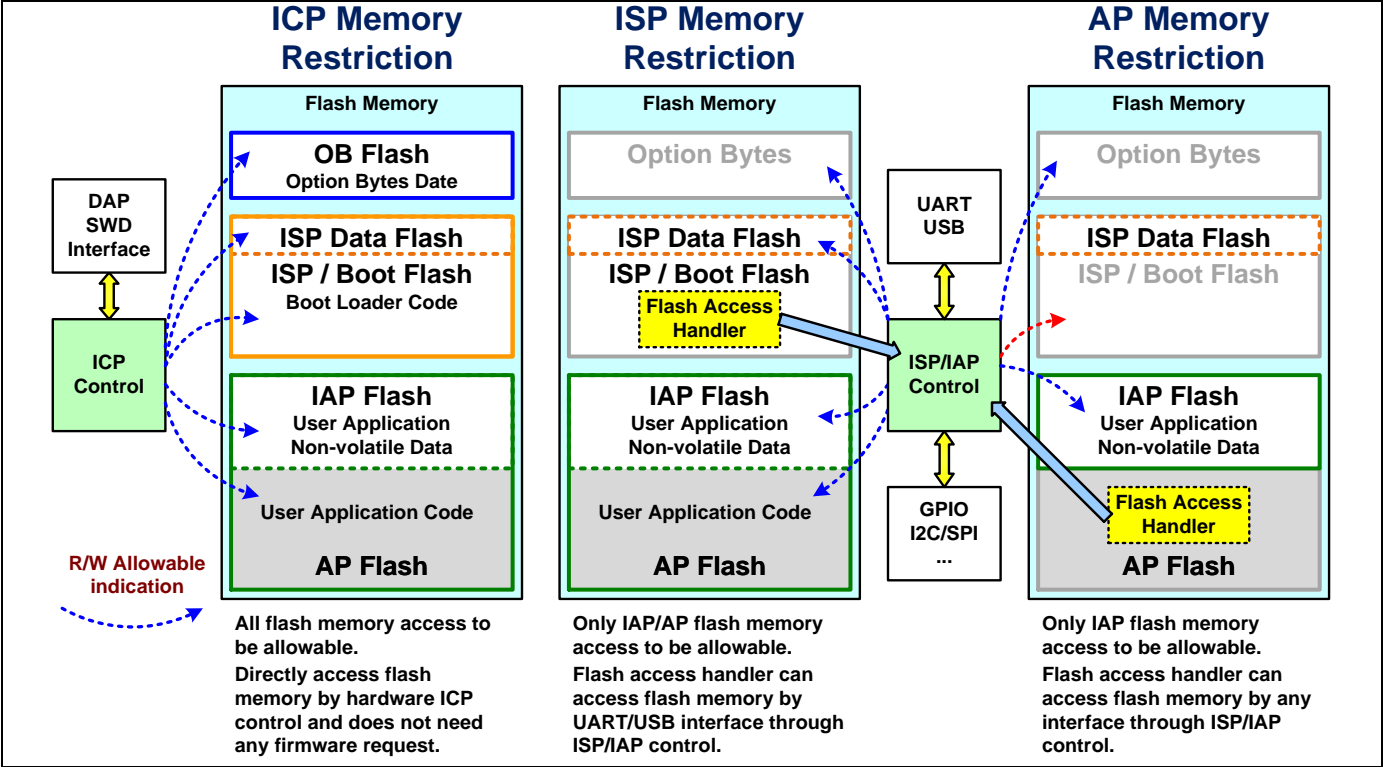


7.8.3. 用于闪存的 ICP/ISP/IAP

芯片提供 3 种闪存访问模式用于 ICP, ISP 和 IAP 应用。ICP 支持通过硬件 SWD 接口且不需要固件的情况下更新整个闪存的数据。此外, 用户可以使用 ISP 和 IAP 这另外两种模式上传新数据到闪存中, 并通过固件闪存访问处理程序获得闪存内容。

下面的图表展示了 ICP/ISP/IAP 闪存访问和限制。

图 7-6. ICP/ISP/IAP 闪存访问限制



7.8.4. 闪存访问限制

当芯片从 AP, ISP, SRAM 启动时, 都会由于用户的控制或硬件的关系, 有不同的闪存访问限制。用户可以通过设置 **MEM_AP_WEN** 或 **MEM_IAP_WEN** 寄存器保护 AP 或 IAP 闪存写权限。

用户可以通过设置 **MEM_ISP_REN** 或 **MEM_ISP_WEN** 寄存器保护 ISP 闪存写或读权限。而且用户可以通过设置 **MEM_ISPD_REN** 或 **MEM_ISPD_WEN** 寄存器保护 ISPD 闪存写或读权限。**MEM_ISP_REN**, **MEM_ISP_WEN** 和 **MEM_ISPD_REN** 寄存器只能在从 ISP 模式启动时被设置, 但是可以在任何启动模式下被清除或禁用。**MEM_ISPD_WEN** 寄存器只能在从 ISP 模式启动时被设置。

下面的表格展示了不同启动模式下闪存读/写限制。

表 7-3. 闪存访问限制对比启动模式

闪存	CPU 起始地址	启动来源			访问使能寄存器
		AP 闪存	ISP 闪存	SRAM	
重定向内存空间	0x0000 0000	Alias of AP 只读	Alias of ISP 只读	Alias of SRAM 只读	
Alias-0 of IAP	与引导码相邻	只读			
AP 闪存	0x1800 0000	只读	只读	只读	AP_WEN=0
			可读/可写	可读/可写	AP_WEN=1
Alias-1 of IAP	与 AP 闪存相邻	只读	只读	只读	IAP_WEN=0
			可读/可写	可读/可写	IAP_WEN=1

IAP 闪存	0x1A00 0000	只读	只读	只读	IAP_WEN=0
		可读/可写	可读/可写	可读/可写	IAP_WEN=1
ISP 闪存	0x1C00 0000	X	只读	X	ISP_WEN=0 ISP_REN=0
		只写		只写	ISP_WEN=1 ISP_REN=0
		只读		只读	ISP_WEN=0 ISP_REN=1
		可读/可写		可读/可写	ISP_WEN=1 ISP_REN=1
ISPD 闪存	0x1FF0 0000	X	只读	X	ISPD_WEN=0 ISPD_REN=0
			可读/可写		ISPD_WEN=1 ISPD_REN=0
		只读	只读	只读	ISPD_WEN=0 ISPD_REN=1
			可读/可写		ISPD_WEN=1 ISPD_REN=1
SRAM	0x2000 0000	可读/可写	可读/可写	可读/可写	

<注释-1> X：不可访问
<注释-2> ISPD_WEN/ISPD_REN 寄存器位只能在 ISP 模式下改变

7.8.5. CPU 代码执行和暂停

CPU 可以在 AP/ISP 或内嵌 SRAM 中运行代码，也可以通过设置 **MEM_IAP_EXEC** 寄存器让 CPU 运行 IAP 闪存上的代码。

当芯片从 AP 或 ISP 闪存启动时，**MEM_HOLD** 寄存器位用于使能或禁止在闪存写访问时的 CPU 暂停控制。默认是使能了暂停控制的，所以 CPU 会在闪存写访问结束之前都被暂停。当 CPU 在闪存中运行代码时，若执行闪存擦除指令时，CPU 必须暂停，当 CPU 在 SRAM 上运行时，在执行闪存擦除指令时 CPU 不需要暂停，且固件可以禁用 CPU 暂停控制。

表 7-4. 闪存访问下 CPU 暂停控制

启动模式	CPU 暂停	寄存器
		HOLD
AP/ISP	暂停	0
	正常	1
SRAM	正常	x (无影响)

<注释> 暂停 ~ 闪存访问结束前 CPU 将暂停。

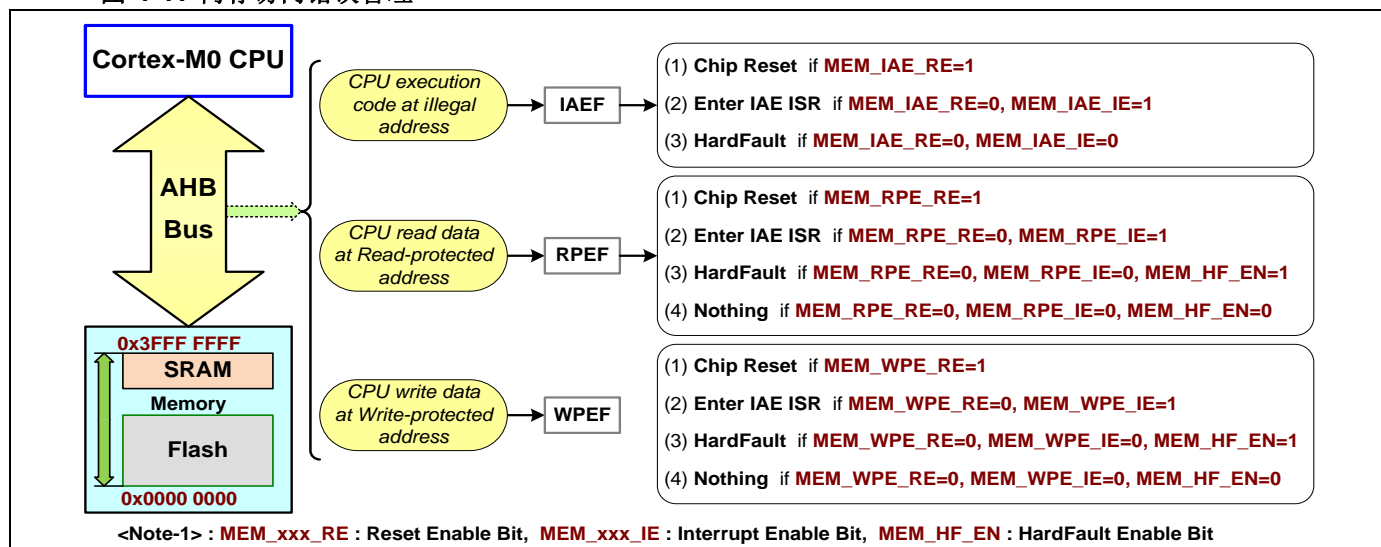
7.8.6. 内存访问错误管理

内存控制器内建 1 个错误管理控制块来管理 CPU 执行代码到非法地址、CPU 读取读保护状态的地址或 CPU 写入写保护状态的地址时产生的内存访问错误。

用户可以通过 **MEM_HF_EN** 寄存器使能在存读写保护错误时产生 CPU **HardFault** 异常。若该位使能，当内存数据读错误发生且 **MEM_RPE_IE** / **MEM_RPE_RE** 寄存器被禁用时，会产生 **HardFault**，而当内存写错误发生且 **MEM_WPE_IE** / **MEM_WPE_RE** 寄存器被禁用时，也会产生 **HardFault**。

下面的图表展示了内存访问错误管理。用户可以选择“芯片复位”、产生“中断”、产生“硬件错误”或者“什么都不做”。

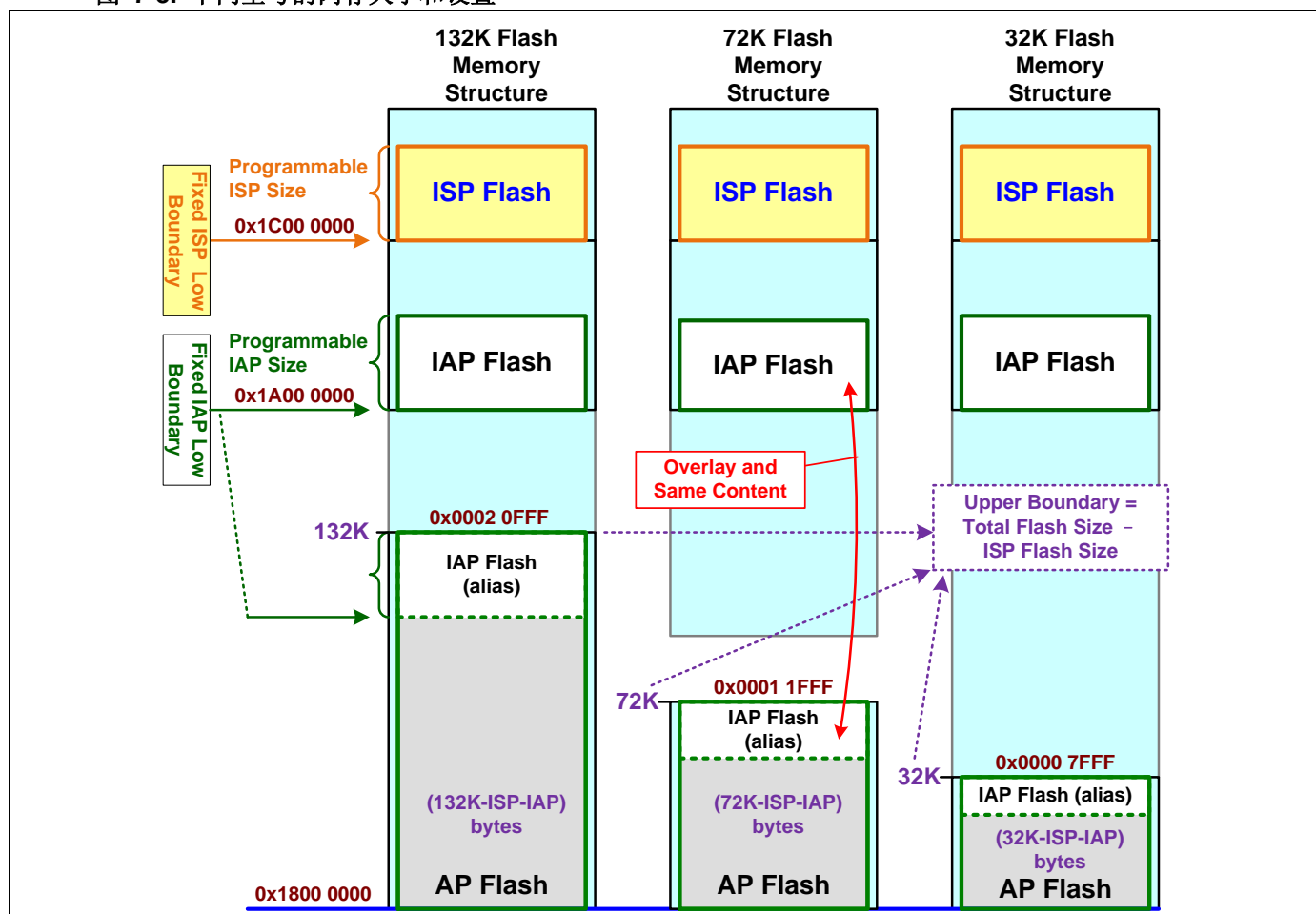
图 7-7. 内存访问错误管理



7.9. 不同产品的闪存设置

根据不同的型号，可以选择 32K、72K 或 132K 字节的闪存大小。下面的图表展示了不同型号的闪存大小和设置。

图 7-8. 不同型号的闪存大小和设置



8. 硬件设置

8.1. 简介

硬件设置决定了芯片的默认行为，且这些设置不会在掉电后丢失。一共有最多 64 字节的片上设置字节(**OB**)闪存用于设置硬件设置并被存储到内嵌闪存中。硬件设置字节只可以通过一般的编程器、“笙泉 ARM 烧录器 U1”、“笙泉 ARM ICE 适配器”来修改（ICE 适配器也支持 ICP 编程功能）。

整片擦除后，所有的硬件设置都会被清除到初始值，且没有 ISP 和 IAP 内存配置。在热复位或冷复位的过程中，设置字节(**OB**)闪存会被装载到芯片的设置寄存器(**OR**)进行硬件设置。

8.2. 硬件设置字节

用户可设置的硬件设置字节如下：：(☑ ~ 建议)

- **BOOT_MS**

系统冷复位启动内存选择且内存映射至地址 0x0000 0000。

☑: 应用程序闪存

☐: 引导闪存

☐: 内嵌 SRAM

- **LOCK_DIS**

☑: 使能。为安全起见，被存到烧写器或编程器的代码将被锁在 0xFFFFFFFF。

☐: 禁用。不锁定。

- **IAP_SIZE**

IAP_SIZE 专指用户定义的 IAP 内存大小。值 0 标志着 IAP 内存大小为 0K 字节，值 1 则是 1K 字节。

- **ISP_SIZE**

ISP_SIZE 专指装有引导码的 ISP 内存大小。值 0 标志着没有 ISP 内存，值 1~128 则是 1~128K 字节。

- **BOD0_WE**

BOD0 触发热复位使能。使能时，当电压阈值检测事件发生时，BOD0 会触发复位信号到 CPU

☐: 使能。BOD0 会触发复位事件到 CPU。

☐: 禁用。BOD0 不会触发复位事件到 CPU。

- **BOD1_WE**

BOD1 触发热复位使能。使能时，当电压阈值检测事件发生时，BOD1 会触发复位信号到 CPU

☐: 使能。BOD1 会触发复位事件到 CPU。

☐: 禁用。BOD1 不会触发复位事件到 CPU。

- **IWDT_EN**

☐: 使能。IWDT 会在上电后自动使能。

☐: 禁用。IWDT 不会在上电后自动使能。

- **IWDT_WP**

IWDT 寄存器写保护使能

☐: 使能。IWDT 寄存器会被写保护。

☐: 禁用。IWDT 寄存器可被软件写入。

- **IWDT_WE**

IWDT 复位产生使能设置

☐: 使能。使能后 IWDT 将能够因 IWDT 事件产生系统复位。

☐: 禁用。禁用后 IWDT 将不能因 IWDT 事件产生系统复位。

- **IWDT_SLP**

SLEEP 模式下 IWDT 计数控制

☐: 停止。停止计数并禁用 IWDT。

☐: 继续。继续计数并使能 IWDT。

- **IWDT_STP**

STOP 模式下 IWDT 计数控制

☐: 停止。停止计数并禁用 IWDT。

☐: 继续。继续计数并使能 IWDT。

- **IWDT_DIV**

IWDT 内部时钟输入分频选择，当 IWDT_EN 被使能时，这些位会被载入 IWDT 输入分频器控制寄存器。

- **EXRST_PIN**

上电后外部复位引脚控制

☒: 使能。RSTN 引脚默认作为外部复位引脚。

☐: 禁用。RSTN 引脚默认作为 GPIO 引脚。

- **SWD_PIN**

上电后 SWD 接口引脚控制

☒: 使能。SWCLK/SWDIO 引脚默认作为 SWD 接口引脚。

☐: 禁用。SWCLK/SWDIO 引脚默认作为 GPIO 引脚。

- **PC_IOM**

上电后端口 C 默认 IO 模式选择。通过这个设置，端口 C 上除 PC4/5/6/13/14 外的所有引脚都默认为 AIO 或 QB 模式。PC4/5/6 引脚的 IO 模式默认为 QB 模式，若 XOSC_EN 被使能，PC13/14 引脚会直接被芯片控制；若 XOSC_EN 被禁用，PC13/14 引脚则根据该寄存器设置控制。

☒: AIO。端口 C 引脚默认为模拟 IO 模式

☐: QB。端口 C 引脚默认为准双向输出模式

[注释]: **PC_IOM** 设置字节不支持于 MG32F02A132/072。

8.3. CFG 设置寄存器

设置寄存器(**OR**)用于芯片硬件设置控制。在热复位或冷复位过程中会从设置字节闪存中加载。参照系统内存章中“[硬件设置字节闪存](#)”节和 CFG 寄存器描述以获取更多信息。

8.3.1. CFG 寄存器保护和锁定

芯片复位之后，除 **CFG_KEY** 寄存器外所有的 CFG 寄存器都会被设置为写保护状态。向 **CFG_KEY** 寄存器写值 **0xA217** 可解除寄存器保护状态，相对的，向寄存器写除 **0xA217** 外的值会保护寄存器。读取 **CFG_KEY** 寄存器的值可得知寄存器处于保护状态（=1）还是未保护状态（=0）。

参照系统复位章的“[寄存器保护和锁定](#)”节的表格和描述以获取更多信息。

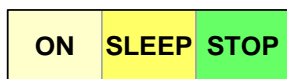
8.3.2. 工厂 ADC 校准值

ADC 参考电压的最高点，中间点和最低点可以被 ADC 校准调节。厂商在 **CFG_ADC0_REFT**，**CFG_ADC0_REFM** 和 **CFG_ADC0_REFB** 寄存器中提供了默认的校准值。这三个寄存器在芯片复位后默认从设置字节闪存中加载。用户也可以校准 ADC 后更新这些寄存器。

[注释]: **CFG_ADC0_REFT**, **CFG_ADC0_REFM** 和 **CFG_ADC0_REFB** 寄存器不支持于 MG32F02A032。

9. GPIO (通用 IO)

9.1. 简介



The module can be running in all power operation modes.

MG32x02z 系列有 **PA, PB, PC, PD** 和 **PE** I/O 端口, 并包含引脚: **PA[15:0], PB[15:0], PC[14:0], PD[15:0], PE[0:3][8:9][12:15]**。LQFP80 封装支持最多 73 个 GPIO 引脚。**PC4, PC5** 和 **PC6** 引脚是专有 **SWCLK, SWDIO** 和 **RSTN** 功能的引脚。若选择外部晶体振荡器作为系统时钟输入, **PC13** 和 **PC14** 引脚需要分别被设置为 **XIN** 和 **XOUT**。实际的 I/O 引脚数量取决于使用的封装类型。

芯片为每个 GPIO 端口内建了一些 IO 模式控制(**PA/PB/PC/PD/PE**)模块。这些模块用于 GPIO 的模式控制、功能复用选择、驱动力设置、输入反相选择、拉高使能、滤波设置和高速使能。另外, 内建的 IO 端口访问控制(**IOP**)模块被用于控制所有的 GPIO 端口的 GPIO 模式输入和输出状态。

注释: (**Px** = 模块{**PA, PB, PC, PD, PE**}, **n**= 输入/输出引脚标号, **OB** = 硬件设置选项字节闪存)会被用于该章的寄存器、信号和引脚/端口描述中。

9.2. 特性

- 支持的通用 IO 引脚
 - LQFP80 封装支持最多 73 根 GPIO 引脚
 - LQFP64 封装支持最多 59 根 GPIO 引脚
 - LQFP48 封装支持最多 44 根 GPIO 引脚
 - QFN32 封装支持最多 29 根 GPIO 引脚
 - TSSOP20 封装支持最多 17 根 GPIO 引脚
- 每根引脚独立可选 IO 模式
 - 推挽输出
 - 准双向
 - 开漏输出
 - 高阻抗输入
 - 模拟 IO
- 灵活的引脚复用功能
- 每根引脚独立支持可配置驱动力
 - 除 **RSTN, XIN, XOUT** 和支持 4 级驱动力选择的引脚外, 所有 GPIO 引脚可配置 2 级驱动力
- 每根引脚独立支持滤波
- 每根引脚独立支持输入反相
- 每根引脚独立支持拉高设置
- 复位后 GPIO 引脚状态和 IO 模式保持
- ❖ **MG32F02A132/072**
 - **PE[3:0]**支持 4 级驱动力选择
 - **PC[3:0], PC14, PD[3:0]** 支持高速选项
- ❖ **MG32F02A032**
 - **PB[3:0][9:8], PD[3:0][8:7]**支持 4 级驱动力选择
 - 除 **RSTN, XIN** 引脚外, 每根引脚独立支持高速选项

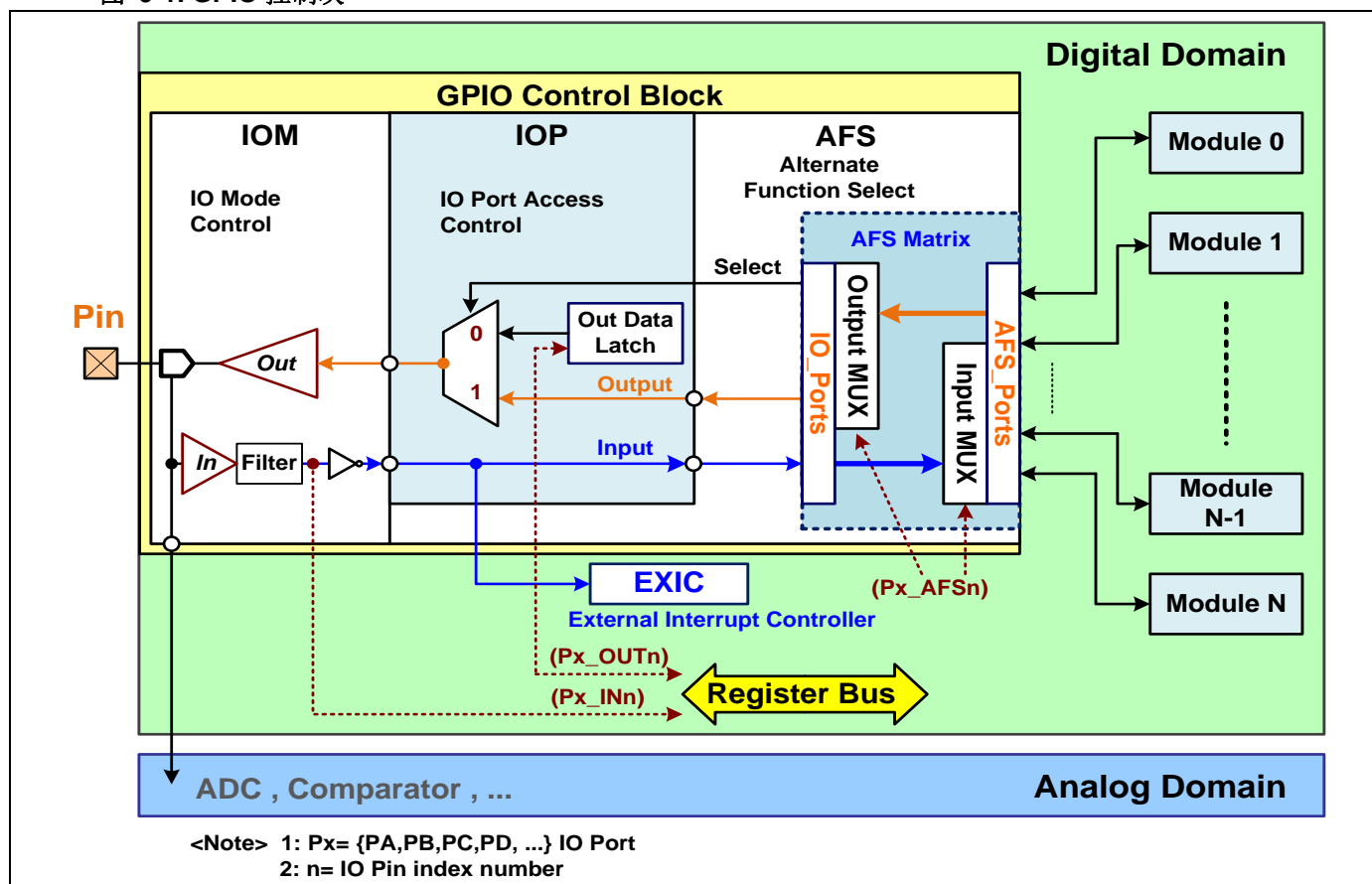
9.3. 控制块

GPIO 控制块包括 IOM、IOP 和 AFS（功能复用选择）块。IOM 块用于设置 IO 工作模式；IOP 块用于控制 GPIO 端口读写访问；AFS 块用于选择 IO 功能复用。每个 GPIO 引脚都可以通过自己独立的控制块和寄存器设置 IOM、IOP、AFS。

GPIO 输出数据位和一些 GPIO 引脚的 IO 模式设置可禁用被系统热复位进行复位。请参照系统复位章的“[GPIO 复位控制](#)”节以获取更多信息。

下面的图表展示了 GPIO 控制块。

图 9-1. GPIO 控制块



9.4. IO 模式

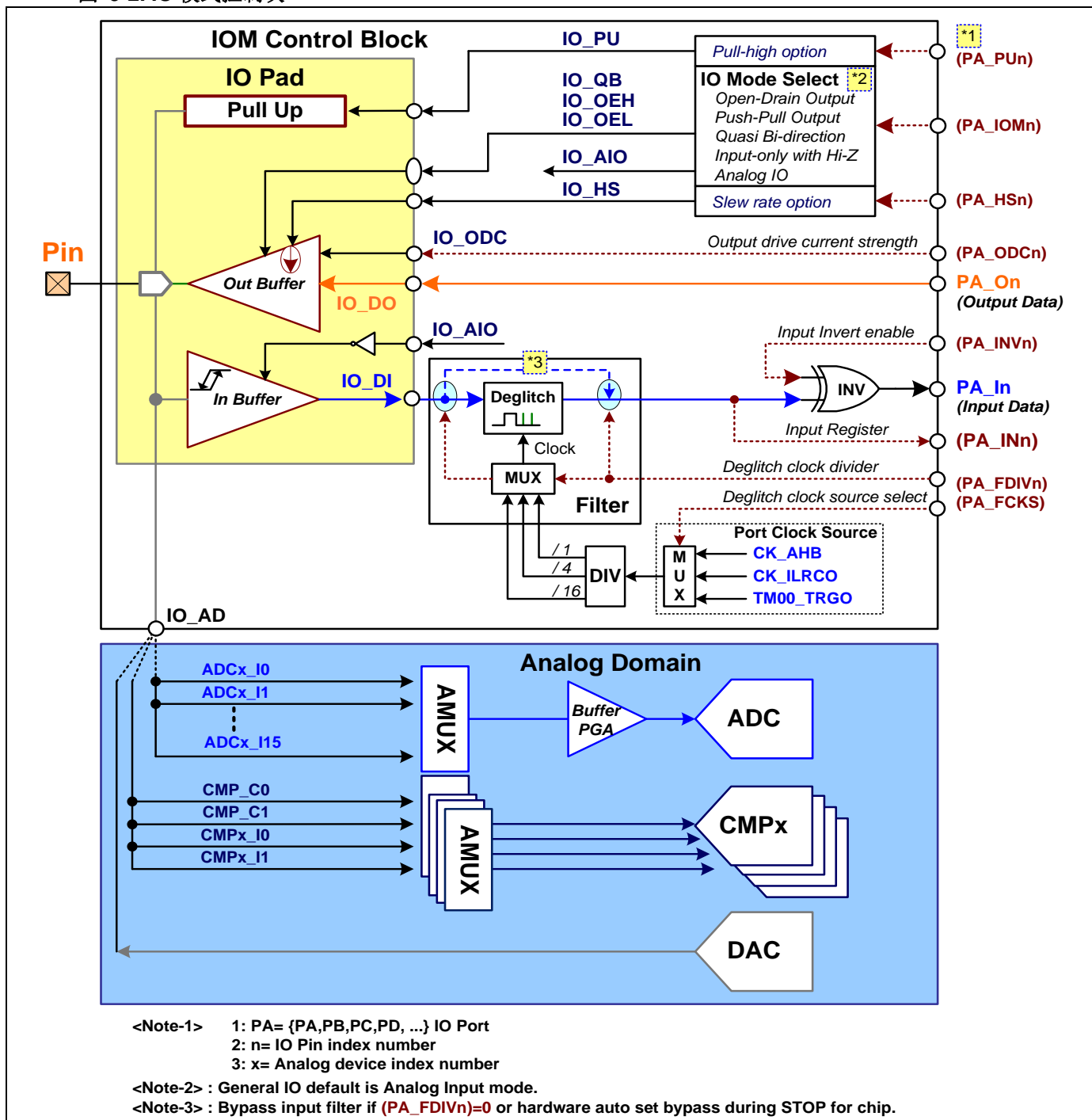
IO 工作模式为每个引脚独立地支持模拟 IO(AIO)、数字输入(DIN)、推挽输出(PPO)、开漏输出(ODO)、准双向(QB)功能。

9.4.1. IO 模式控制块

IO 模式控制块为每个引脚独立地支持可配置 IO 工作模式、高速输出选项、拉高选项、驱动力、IO 滤波和输入反相选择。

下面的图表展示了 IO 模式控制块。

图 9-2. IO 模式控制块



9.4.2. IO 设置

该芯片为 PA, PB, PC, PD 和 PE 的 IO 配置提供了独立的模块。参照引脚定义的描述以获得更多信息。

● IO 模式控制

所有的 GPIO 引脚可以被独立地配置 IO 工作模式。用户可以通过设置 **Px_IOMn** 寄存器为每个 GPIO 引脚提供模拟 IO、数字输入、推挽输出、开漏输出、准双向功能。

下面的表格展示了 IO 模式、拉高、高速控制的寄存器设置。

表 9-1. IO 工作模式控制

IO 模式	拉高	高速	IO 设置	
Px_IOMn	Px_PUn	Px_HSn		
0x0	-	-	模拟 IO	
0x1	0	0	开漏输出	
	0	1		高速
	1	0		拉高
	1	1		拉高+高速
0x2	0	0	推挽输出	
	0	1		高速
	1	0		拉高
	1	1		拉高+高速
0x3	0	-	数字输入	
	1	-		拉高
0x4	0	0	准双向输出 (拉高 1 个时钟)	
	0	1		高速
	1	0		拉高
	1	1		拉高+高速

注释：“Px” = {PA,PB,PC,PD,PE}, “n” = 引脚标号, “-” = 不影响

MG32F02A132/072 在上电时，PA, PB, PD 和 PE IO 模式为模拟 IO(AIO)，而 PC 的 IO 模式则是准双向输出(QB)。

特别的是，MG32F02A032 在上电时，除 **PC4/5/6/13/14** 外的端口 C(PC)引脚可通过硬件设置字节(**OB**)的 **CFG_PC_IOM** 寄存器设置为 AIO(默认)或 QB 模式。

请参照“[特殊引脚的功能复用](#)”节以获得更多关于 PC 特殊引脚上电默认设置的信息。

● 高速输出控制

对于 MG32F02A132/072, **PB[3:0]**, **PC[3:0]**和 **PC14** 引脚支持高速输出选项。对于 MG32F02A032, 除 **RSTN**, **XIN** 引脚外都支持高速选项，用户可以通过 **Px_HSn** 寄存器为这些 GPIO 独立地使能 IO 高速输出选项。

● 拉高控制

每个 GPIO 引脚都可以被独立地设置 IO 拉高设置。用户可以通过 **Px_PUn** 寄存器为这些 GPIO 独立地使能拉高设置。

● 输入反相控制

每个 GPIO 引脚都可以被独立地设置输入反相。用户可以通过 **Px_INVn** 寄存器为这些 GPIO 独立地使能输入反相。

● 输出驱动力控制

每个 GPIO 引脚都可以被独立地设置输出驱动力。一共有两种 IO 引脚带有可设置驱动力强度，一种是可设置两级驱动力，另一种则是可以设置 4 级驱动力。用户可以通过 **Px_ODCn** 寄存器为这些 GPIO 独立地使能输出驱动力设置。

● 输入滤波控制

每个 GPIO 引脚都可以被独立地设置输入数字滤波。该滤波器提供了可设置的滤波时钟源和分频器以适配输入信号的工作频率和可能的噪音。

用户可以通过设置 **Px_FCKS** 寄存器选择 AHB 时钟、8 分频的 AHB 时钟、ILRCO 时钟、TM00 触发输出或 UT（单位时间）时钟作为滤波时钟源并为 GPIO 端口的所有引脚进行配置。滤波时钟源可以为每个 GPIO 端口独立地设置。此外，用户还可以通过 **Px_FDIVn** 寄存器为每个 GPIO 引脚独立地设置滤波时钟分频值 1、4、16 分频或绕过分频器。

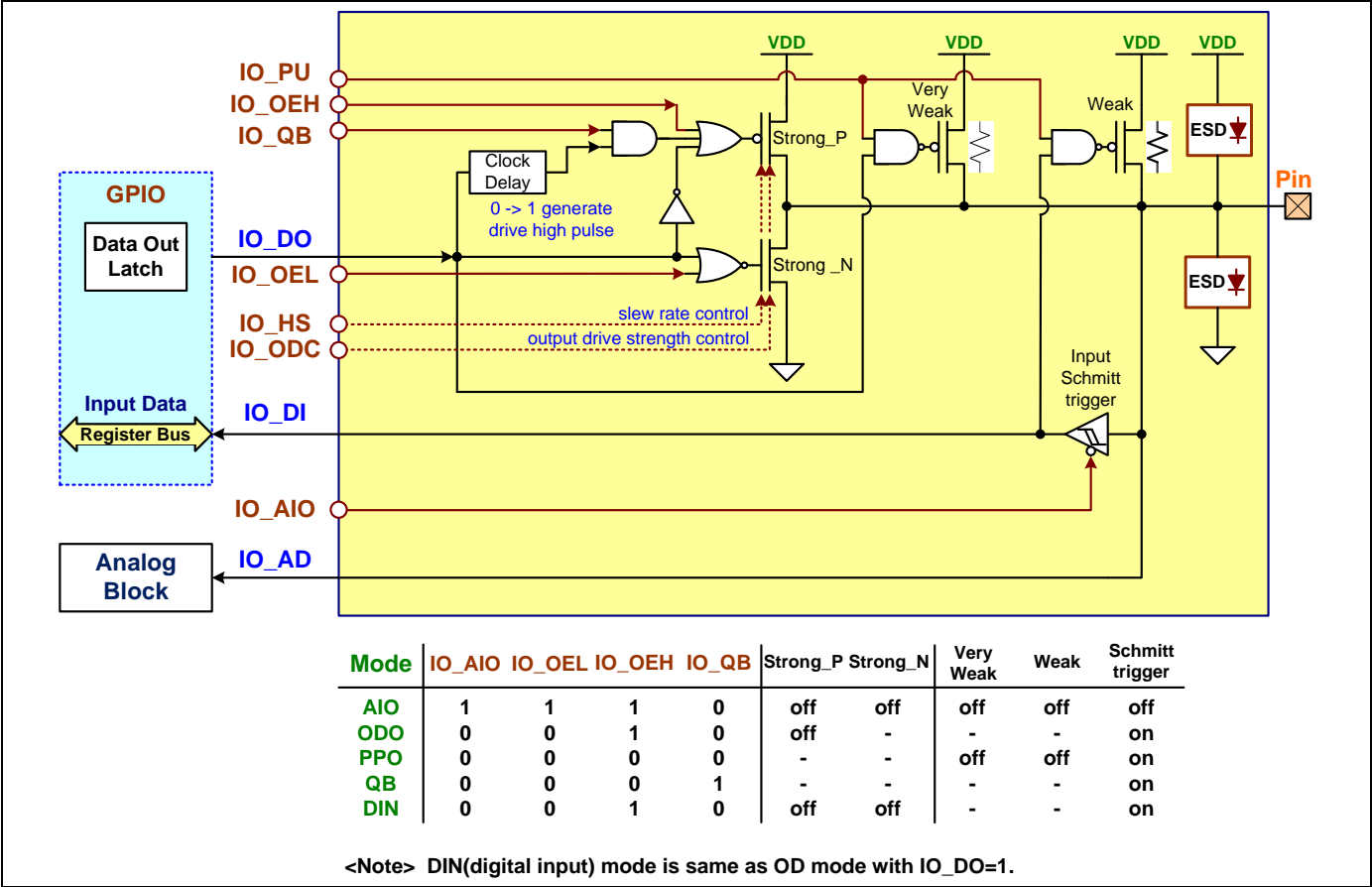
● 功能复用选择

每个 GPIO 引脚都可以独立地被使能输入反相。用户可以通过 **Px_AFSn** 寄存器为每个 GPIO 引脚独立地被使能输入反相。

9.5. IO 结构

I/O 用于支持模拟 IO、数字输入、推挽输出、开漏输出和准双向功能。
下面的图表展示了通用 IO 的控制结构块。

图 9-3. IO 控制结构图



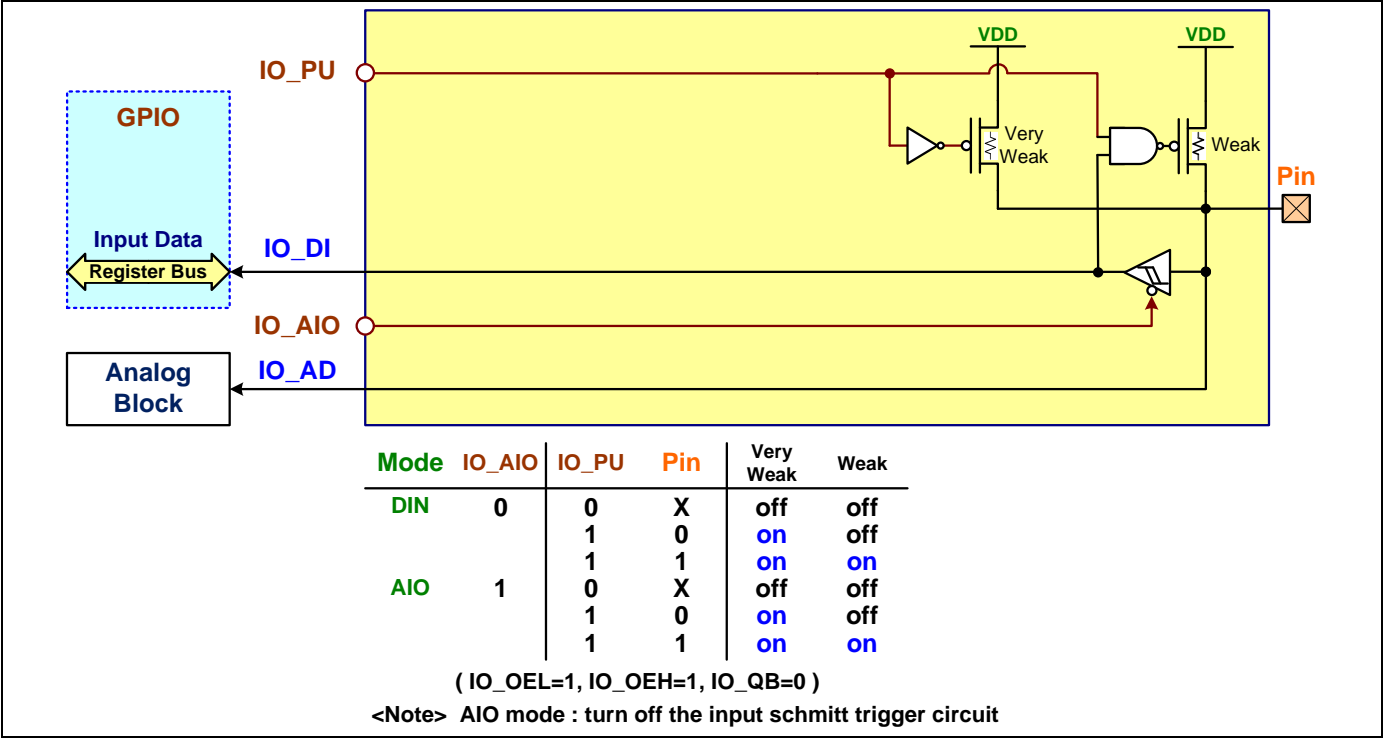
9.5.1. 模拟 IO 和数字输入结构

数字输入设定是带有施密特触发器缓冲的高阻抗输入。每个引脚都可以独立控制 1 个上拉电阻。该模拟输入直接把 IO 面板连接到了内部模拟设备。

模拟 IO 的设置是用于 ADC 输入、模拟比较器和 DAC 输出的。在应用中，模拟 IO 引脚上是不带上拉电阻的，当选择模拟 IO 模式(AIO)，施密特触发器缓冲会被强制禁用。

该模拟 IO 和数字输入端口设置如下图展示。

图 9-4. IO 结构–模拟 IO 和数字输入

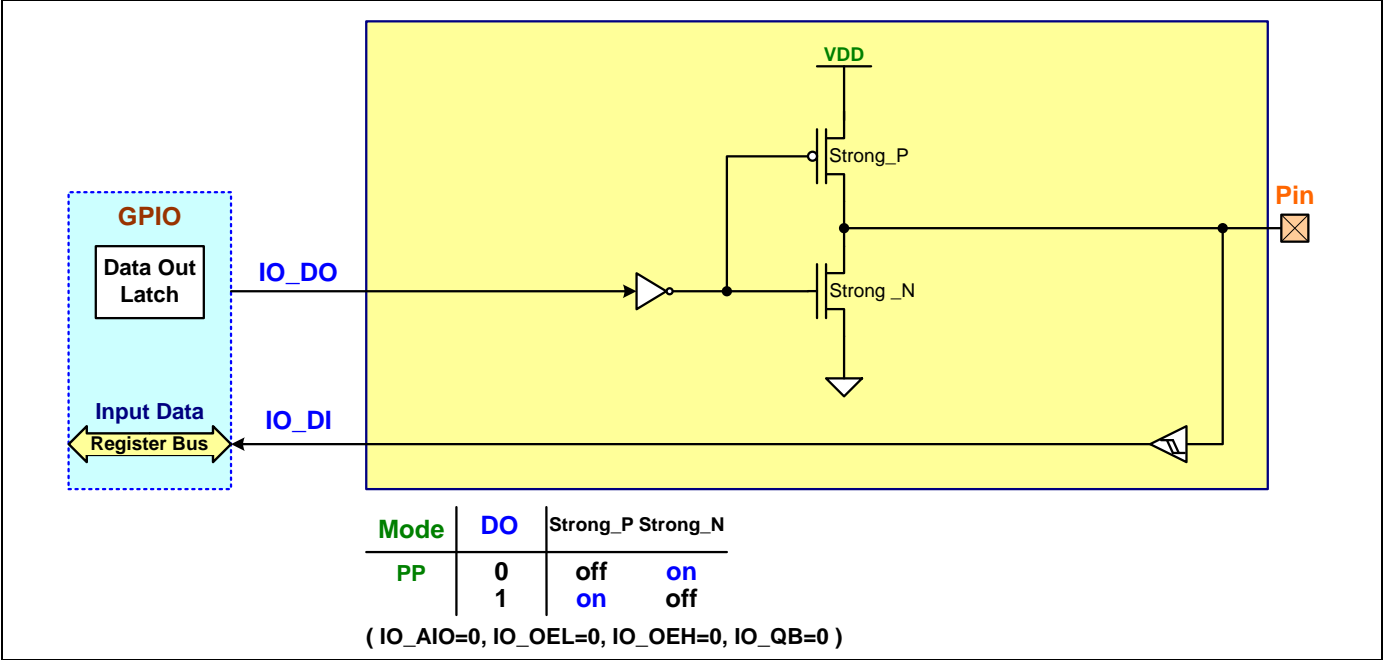


9.5.2. 推挽输出结构

推挽输出设置和开漏输出和准双向输出模式有相同的下拉结构，但是在端口寄存器包含了逻辑“1”时将提供持续性的强上拉。推挽模式一般在需要从端口输出较多的电流时使用。此外，端口引脚的输入路径设置和准双向模式相同。

下面的图表展示了推挽端口设置。

图 9-5. IO 结构–推挽

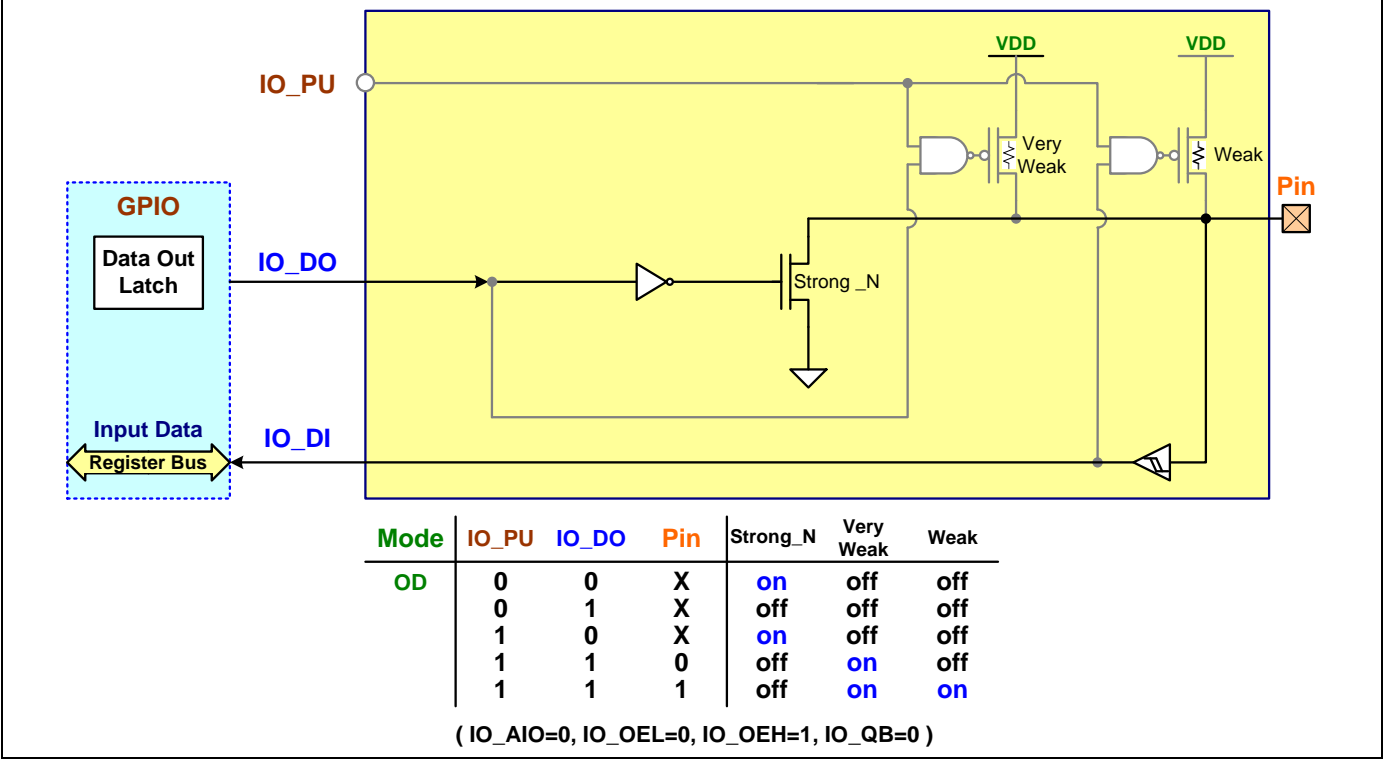


9.5.3. 开漏输出结构

开漏输出设置会在端口寄存器包含逻辑“0”时关闭所有上拉，并只驱动端口引脚的下拉晶体管。要使用该设置，端口引脚必须有外部上拉，一般来说是使用一个连接到VDD的电阻。该模式的拉低与准双向模式相同。此外，端口引脚的输入路径设置和准双向模式相同。

下面的图表展示了开漏端口设置。

图 9-6. IO 结构-开漏



9.5.4. 准双向 IO 结构

准双向端口可以用于作为输入和输出时不需要重新设置端口的情况。由于端口输出逻辑“高电平”时，它的弱驱动能力，导致外部设备能够拉低该引脚，使不需要重新设置端口成为可能。当引脚输出低电平时，它的强驱动力能拉低大电流。为了符合不同的需求，准双向输出端口中包含 3 个上拉晶体管。

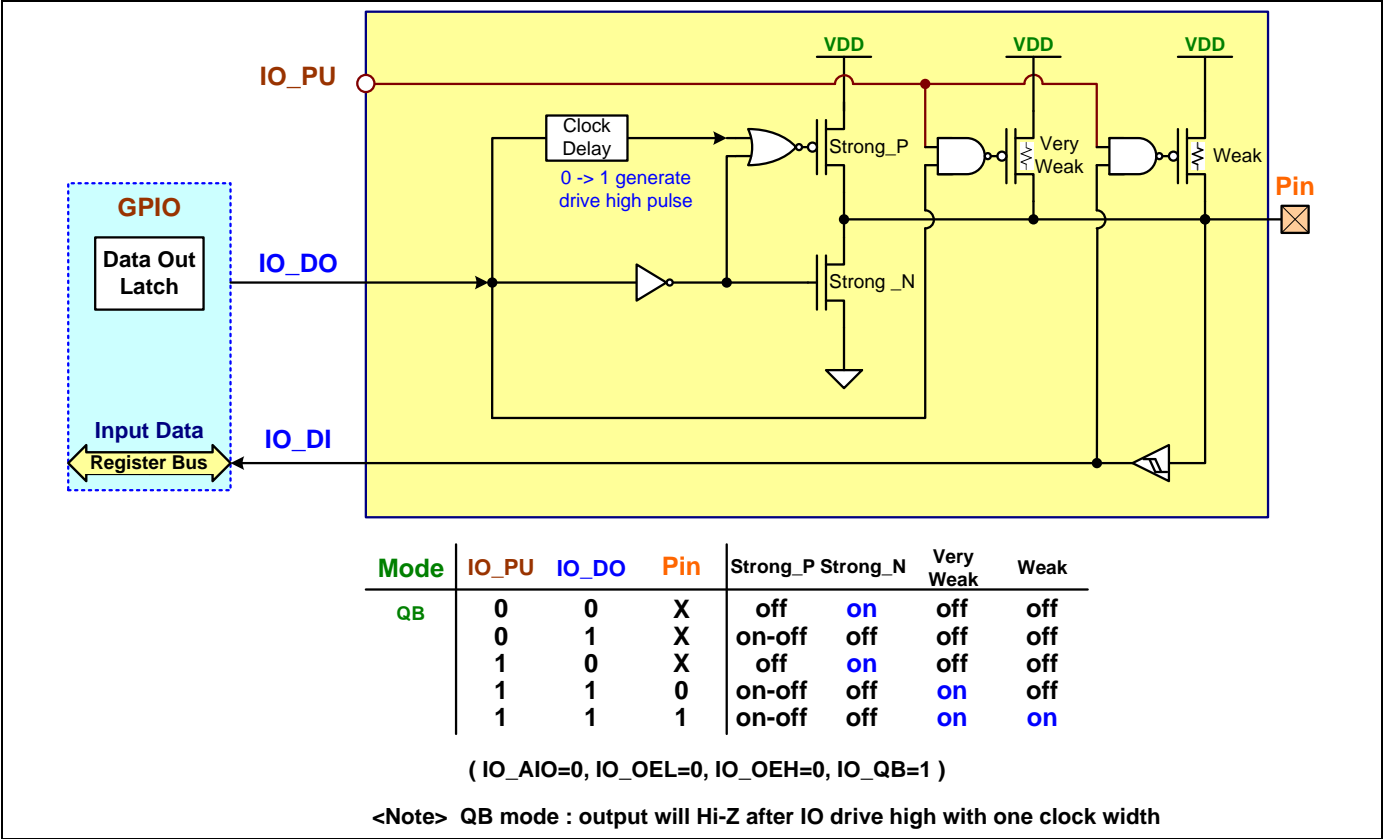
其中一个上拉，被称为“非常弱”的上拉，该上拉会在引脚包含逻辑“1”时启动。该上拉源的非常小的电流将拉高浮动不定的引脚。

第二个上拉，被称为“弱”上拉。该上拉会在该引脚的端口寄存器包含逻辑“1”时启动，并且该引脚本身也会处于逻辑“1”状态。该上拉为输出“1”的准双向引脚提供了初级的上拉源电流。若该引脚被外部设备拉低，该“弱”上拉关闭，“非常弱”上拉会保持启动。为了在这些情况下拉低引脚，外部设备必须吸收足够的电流以对弱上拉进行压制，并将端口引脚拉到低于其输入阈值电压。

第三个上拉被称为“强”上拉。该上拉用于在准双向端口引脚的端口寄存器(Px_OUTn)从逻辑“0”到“1”改变时的低电平到高电平的提速。当该情况发生时，强上拉将启动 1 个系统时钟(CK_SYS)的时间，快速地拉高引脚。

下面的图表展示了准双向端口设置。

图 9-7. IO 结构—准双向

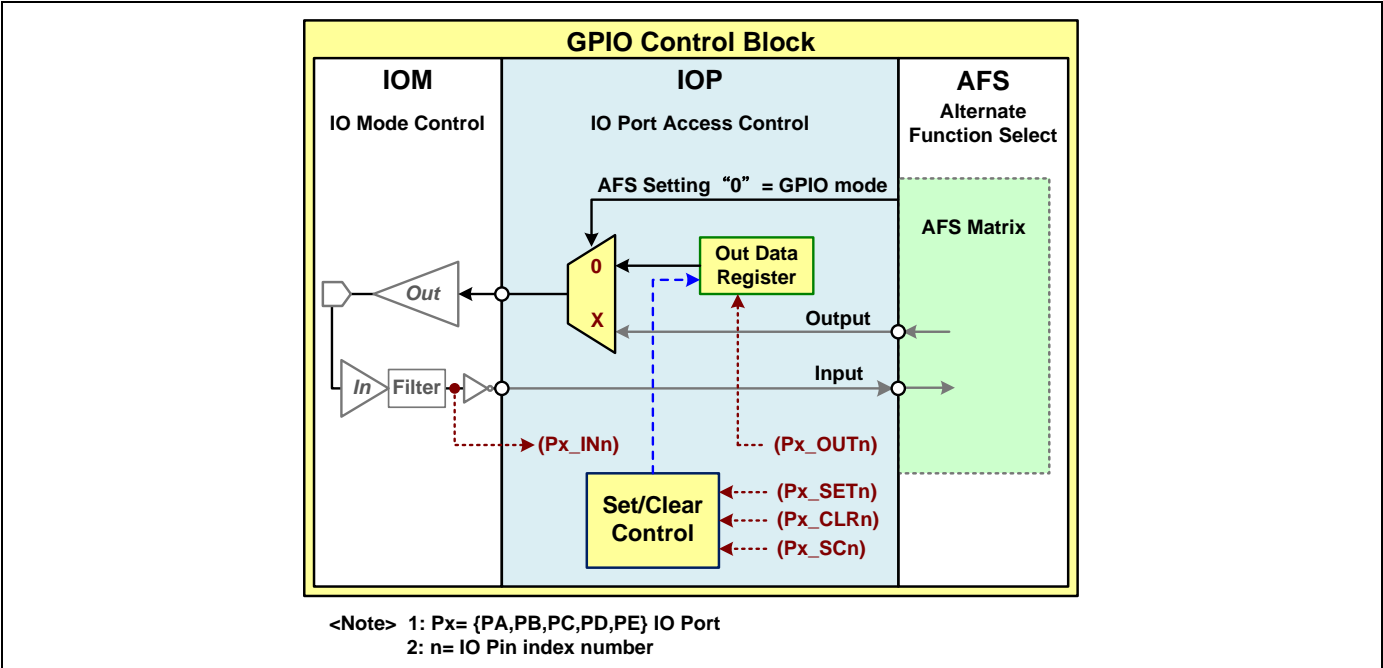


9.6. IO 端口访问

9.6.1. GPIO IO 控制

当任意 IO 引脚被通过 AFS 设置为 GPIO 时,用户可以直接设置逻辑输出或获取 IO 引脚的逻辑输入。每个 GPIO 引脚都有 1 个独立的数据输出寄存器位存储输出逻辑值。用户可以直接通过每个 GPIO 引脚的 **Px_OUTn** 寄存器位读写数据。此外,用户还可以直接通过读取 **Px_INn** 寄存器位的值获得 GPIO 引脚的逻辑状态。

图 9-8. IO 端口访问块



每个端口的 **Px_OUT** 寄存器和 **Px_IN** 寄存器是被设计用于通过 **Px_OUTn** 和 **Px_INn** 位读写每个 GPIO 端口的所有引脚的逻辑输出与输入。这让固件能简单地同时控制 GPIO 端口($n=\{0\sim15\}$)。

9.6.2. 设置和清除控制

为了固件控制，每个 GPIO 引脚都有 1 个 **Px_SETn** 控制位用于设置数据输出寄存器位，还有 1 个 **Px_CLRn** 控制位用于清除数据输出寄存器位。**Px_SC** 寄存器用于通过 **Px_SETn** 位和 **Px_CLRn** 位设置或清除所有引脚的数据输出寄存器位。

这些控制寄存器位被设计成仅写 1 时有效，写 0 无效。因此让固件能简单地同时设置或清除一个 GPIO 端口下的多个引脚。当 GPIO 引脚相关的 **Px_SETn** 位和 **Px_CLRn** 位被设置成 1，相关的数据位也会被设置成 1 ($n=\{0\sim15\}$)。

9.6.3. 位方式 IO 控制

该芯片为每个 GPIO 引脚提供 1 个 **Px_SCn** 寄存器控制位用于设置、清除数据输出寄存器位或读引脚状态。该寄存器位写 1 时设置数据位，写 0 清除数据，读该寄存器位则可获取 GPIO 引脚状态。

由于 **Px_SCn** 寄存器位需要占用 8 位内存空间，因此固件能够简单的通过 CPU 字节访问指令控制单个 GPIO 引脚。这有点像 8051 MCU 的位访问 IO 控制。

参照 **Px_SCRn** 寄存器定义以获取更多信息。

9.7. 功能复用选择

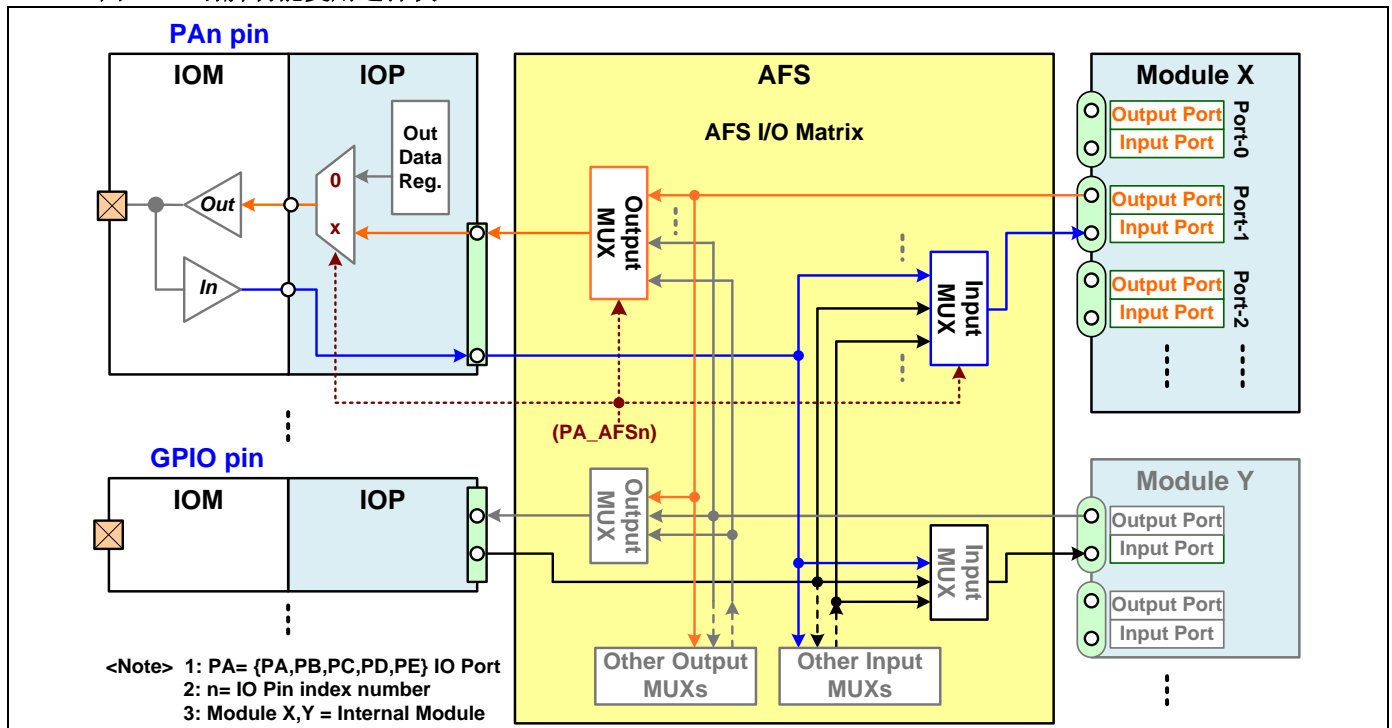
9.7.1. 功能复用选择控制

用户可以为每个 GPIO 引脚独立地通过设置 **Px_AFSn** 寄存器进行 AFS 选择 IO 的模块功能。参照引脚定义和寄存器定义描述以获取更多详细信息。

通常 AFS 默认把除 **XIN/XOUT**, **SWCLK/SWDIO** 和 **RSTN** 功能引脚外的所有的 GPIO 引脚设定为 GPIO 模式。这些引脚可被硬件设置 **OB** 改变。参照“[特殊引脚的功能复用](#)”节以获取更多信息。

下面的图表展示了引脚功能复用选择块。

图 9-9. 引脚功能复用选择块



● 引脚 AFS 表

参照芯片数据手册的“引脚功能复用”表以获取更多信息。

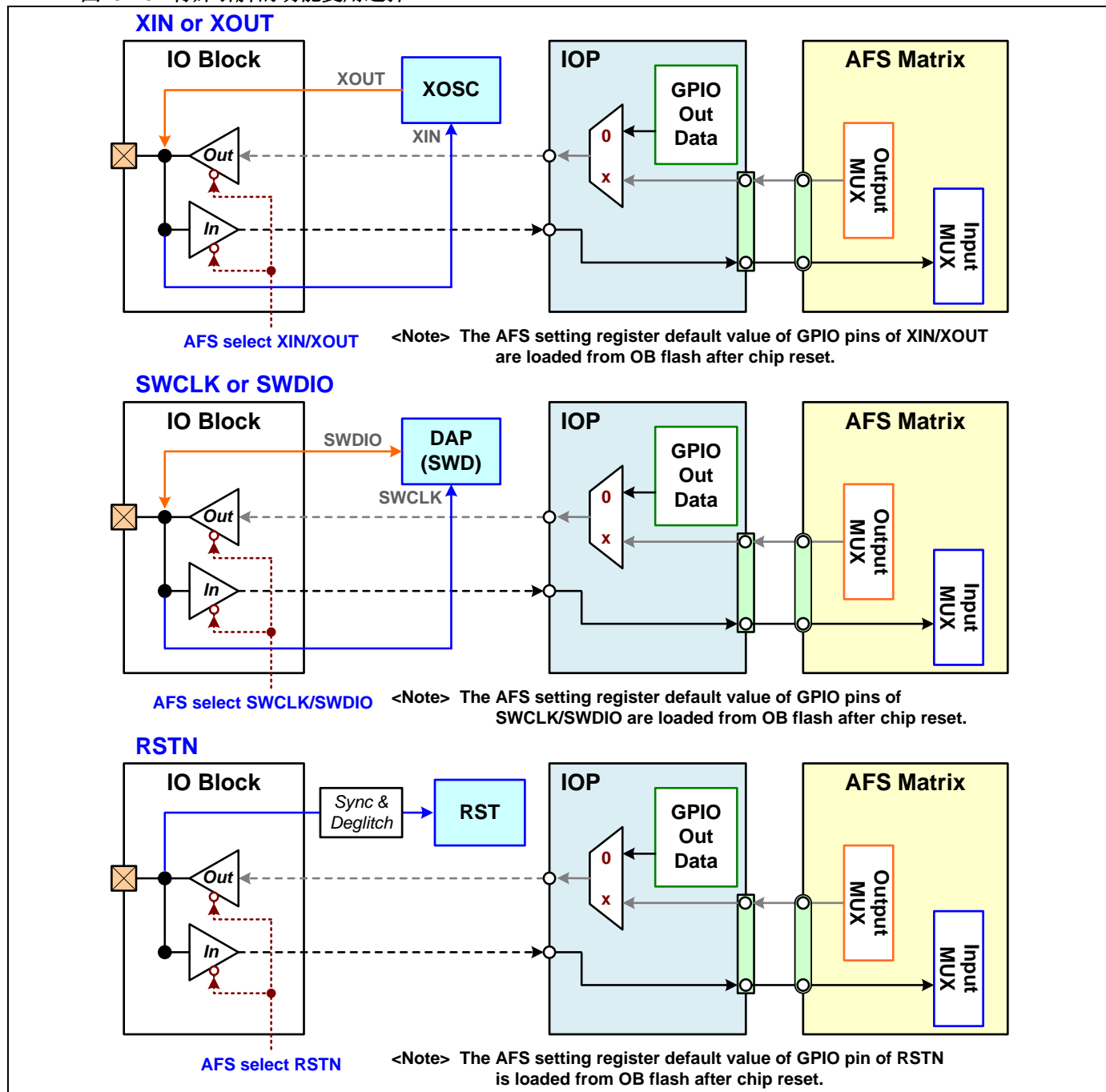
9.7.2. 特殊引脚的功能复用

通常 AFS 默认把除 **XIN/XOUT**, **SWCLK/SWDIO** 和 **RSTN** 功能引脚外的所有的 GPIO 引脚设定为 GPIO 模式。**XIN/XOUT**, **SWCLK/SWDIO** 和 **RSTN** 功能引脚可在芯片复位（冷复位）后被硬件设置 **OB** 闪存使能，而且这些硬件设置 **OB** 值是通过被载入 GPIO AFS 设置的方式更新这些引脚功能的。所以用户可以在上电完成后通过修改 GPIO AFS 设置修改这些引脚。

当 **XIN/XOUT**, **SWCLK/SWDIO** 或 **RSTN** 引脚功能被使能，相关引脚的 IO 模式寄存器设置和拉高寄存器设置会失效，由硬件直接进行控制。

下面的图表展示了 **XIN/XOUT**, **SWCLK/SWDIO** 和 **RSTN** 功能引脚的功能复用选择块。

图 9-10. 特殊引脚的功能复用选择



● 外部晶振引脚

该芯片为外部晶振电路内建了 1 个内部晶体振荡器(**XOSC**)。该芯片能通过硬件设置字节(**OB**)使能并通过晶振 **XOSC** 启动, 并设置 **XIN** 和 **XOUT** 引脚作为外部晶振输入和输出。此外, 用户也可以通过 GPIO AFS 寄存器重新设置 **XIN** 和 **XOUT** 引脚的 AFS 设置。

当 **XIN** 和 **XOUT** 引脚被设置为外部晶振输入和输出时, 内部 **XOSC** 振荡器会被硬件使能, 外部晶振会开始震荡。当 **XOUT** 引脚不被设置为外部晶振输出时, 内部晶体振荡器(**XOSC**)会被芯片关闭。只有 **XIN** 引脚能被设置为外部晶振输入且能输入比如 OSC 的外部时钟源。参照[系统时钟](#)章以获取更多信息。

● SWD 引脚

该芯片为系统调试内建 1 个调试访问端口(**DAP**)和调试控制电路。该芯片能通过硬件设置字节(**OB**)使能 **SWCLK** 和 **SWDIO** 引脚用于连接比如 ICE 外部调试设备。此外, 用户也可以通过 GPIO AFS 寄存器重新设置 **SWCLK** 和 **SWDIO** 引脚的 AFS 设置。

● 外部复位引脚

该芯片从 **RSTN** 引脚提供了 1 个外部硬件复位输入以保证可靠的上电复位。参照[系统复位](#)章以获取更多信息。

该芯片能通过硬件设置字节(**OB**)使能 **RSTN** 引脚作为外部复位引脚或其他(GPIO ...)。此外, 用户也可以通过 GPIO AFS 寄存器重新设置 **RSTN** 引脚的 AFS 设置。除非其他 GPIO 引脚已被使用, 强烈建议保持 **RSTN** 引脚作为外部复位引脚的功能。

上电时的 **PC4/5/6** 引脚 IO 模式一直为 QB 模式。若 **CFG_XOSC_EN** 被使能, 上电时的 **PC13/14** 引脚 IO 模式是直接被芯片控制的; 若 **CFG_XOSC_EN** 被禁用, 上电时的 **PC13/14** 引脚 IO 模式则是被该寄存器控制的。直接被芯片控制的。

下面的表格展示了端口 C 上电默认 IO 模式。

[注释]: 在 MG32F02A132/072 中, **CFG_PC_IOM** 不被支持, 且端口 C 上电 IO 模式恒为 AIO 模式。

表 9-2. 端口 C IO 模式默认设置

端口 C 引脚	硬件 OR 设置				IO 寄存器上电默认值		
	CFG_XOSC_EN	CFG_EXRST_PIN	CFG_SWD_PIN	CFG_PC_IOM	Px_IOMn	Px_PUn	Px_AFSn
PC[14:13]	0	x	x	0	0x0 (AIO)	0	GPIO
				1	0x4 (QB)	1	GPIO
	1	x	x	x	0x0 (AIO)	0	XIN, XOUT
PC6	x	0	x	x	0x4 (QB)	1	RSTN
		1					GPIO
PC[5:4]	x	x	0	x	0x4 (QB)	1	SWDIO SWCLK
			1				GPIO
PC[12:7,3:0]	x	x	x	0	0x0 (AIO)	0	GPIO
				1	0x4 (QB)	0	GPIO

注释 : x : 不影响, AIO : 模拟 IO, QB : 准双向输出

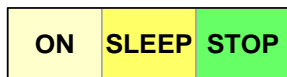
9.7.3. GPIO AFS 锁定

该芯片只支持针对 **SWCLK/SWDIO** 和 **RSTN** 引脚的 GPIO AFS 设置锁定。用户必须同时在 **PC_AFSn** 寄存器设置 AFS 并设置 **PC_LCKn** 寄存器位为 1 才行(n = {4, 5, 6})。当 **PC_LCKn** 寄存器位为 0 时, **PC_AFSn** 寄存器位的写访问无效。该芯片会在寄存器写操作后自动清除 **PC_LCKn** 寄存器位。

参照 **PC_CR4**, **PC_CR5** 和 **PC_CR6** 寄存器描述以获取更多信息。

10. 中断

10.1. 简介



The module can be running in all power operation modes.

复位后，CPU 开始从复位中断向量(0x00000004)地址开始执行，这个地址会是用户应用程序的起点。为了使用这些中断，中断服务地址（中断向量）需被定位在地址 0x000000BF~0x00000000。

该 ARM cortex M0 CPU 内嵌了 1 个含有 32 个 4 级优先级外部中断的 **NVIC**（嵌套向量中断控制器），此外，还内置了 1 个与 **NVIC** 连接的 EXIC（外部中断控制器）模块。

注释: (Px = 模块{PA, PB, PC, PD, PE}, n= 输入/输出引脚标号, **OB** = 硬件设置选项字节闪存)会被用于该章的寄存器、信号和引脚/端口描述中。

10.2. 特性

- 内置 1 个含有 32 个 4 级优先级外部中断的 **NVIC**
- 内置 1 个与 **NVIC** 连接的 **EXIC**（外部中断控制器）模块
 - 独立的高电平/低电平和上升沿/下降沿触发选项
- 内置 1 个用于唤醒事件控制的 **WIC**（唤醒中断控制器）
- 所有的 **GPIO** 引脚均可被设置为中断源或按键输入
 - 中断功能端口支持“或”逻辑
 - **KBI** 功能端口支持“与”逻辑
- 支持 **CPU NMI/RXEV/TXEV** 功能的外部引脚
 - 可设置 **CPU NMI** 输入功能的引脚
 - 可设置 **CPU RXEV** 输入功能的引脚
 - 可设置 **CPU TXEV** 输出功能的引脚

10.3. 中断结构

每个中断在程序存储器中会被分配一个固定的位置。中断会导致 CPU 转跳至那个位置，在那里执行服务程序。比如 NMI 中断，会被分配到 0x00000008 地址，当 NMI 被使用时，它的服务程序就必须在 0x00000008 地址开始执行。

中断服务地址被以 4 字节分隔：用于复位中断的 0x00000004、用于 **NMI** 的 0x00000008、用于 **Hard-Fault** 的 0x0000000C、用于 **SVC** 的 0x0000002C、用于 **PendSV** 的 0x00000038、用于 **SysTick** 的 0x0000003C 等。

NVIC 有 7 种异常类型：**Reset**、**NMI**、**HardFault**、**SVC**、**PendSV**、**SysTick** 和中断(IRQ)。NVIC 支持 32 个外部中断输入。中断是由外围设备发出信号或由软件请求生成的异常。4 级优先级中断结构在处理这些中断源方面具有很大的灵活性。

10.3.1. 中断源

“挂起位”是一个若通过设置“设置使能位”，意味着将有中断产生的中断标志。“挂起位”可以被软件设置或清除，结果与硬件设置或清除的结果相同。也就是说，中断可以被软件生成，挂起的中断也可以被取消。“优先级位”决定了各个中断的优先级。“级别内优先级”用于解决同优先级的同时请求。“向量地址”是在程序存储器内的中断服务程序的入口。

下面的表格展示了所有中断源。

[注释]: EMB, DAC, TM2x, I2C1 和 URT2/3 模块仅支持于 MG32F02A132/072。

表 10-1. 中断源

NVIC			主要模块	寄存器			优先级	向量地址
异常 No.	IRQ No.	中断名		设置/清除使能	挂起位	优先级位		
0	-	Initial					-	0x00000000
1	-	Reset					-3	0x00000004
2	-14	NMI			0xE000ED04[31]		-2	0x00000008
3	-13	HardFault					-1	0x0000000C
4~10	-	保留					-	
11	-5	SVC				0xE000ED1C [31:24]	可设置	0x0000002C
12~13	-	保留					-	
14	-2	PendSV			0xE000ED04[28] 0xE000ED04[27]	0xE000ED20 [23:16]	可设置	0x00000038
15	-1	SysTick		0xE000E010 [1]	0xE000ED04[26] 0xE000ED04[25]	0xE000ED20 [31:24]	可设置	0x0000003C
16	0	WWDT	WWDT	0xE000E100 0xE000E180	0xE000E200 0xE000E280	0xE000E400	可设置	0x00000040
17	1	SYS	SYS				可设置	0x00000044
18	2	保留					可设置	0x00000048
19	3	EXINT0	EXIC				可设置	0x0000004C
20	4	EXINT1	EXIC			0xE000E404	可设置	0x00000050
21	5	EXINT2	EXIC				可设置	0x00000054
22	6	EXINT3	EXIC				可设置	0x00000058
23	7	COMP	CMP				可设置	0x0000005C
24	8	DMA	DMA			0xE000E408	可设置	0x00000060
25	9	保留					可设置	0x00000064
26	10	ADC	ADC0				可设置	0x00000068
27	11	DAC	DAC				可设置	0x0000006C
28	12	TM0x	TM00			0xE000E40C	可设置	0x00000070
29	13	TM10	TM10				可设置	0x00000074
30	14	TM1x	TM16				可设置	0x00000078

31	15	TM20	TM20				可设置	0x0000007C
32	16	TM2x	TM26			0XE000E410	可设置	0x00000080
33	17	TM3x	TM36				可设置	0x00000084
34	18	保留					可设置	0x00000088
35	19	保留					可设置	0x0000008C
36	20	URT0	URT0			0XE000E414	可设置	0x00000090
37	21	URT123	URT1				可设置	0x00000094
38	22	保留					可设置	0x00000098
39	23	保留					可设置	0x0000009C
40	24	SPI0	SPI0			0XE000E418	可设置	0x000000A0
41	25	保留					可设置	0x000000A4
42	26	保留					可设置	0x000000A8
43	27	保留					可设置	0x000000AC
44	28	I2C0	I2C0			0XE000E41C	可设置	0x000000B0
45	29	I2Cx	I2C1				可设置	0x000000B4
46	30	保留					可设置	0x000000B8
47	31	保留					可设置	0x000000BC
可设置: 可设置优先级 0~3								

10.3.2. 异常类型

一共有 7 种异常类型: *Reset*、*NMI*、*HardFault*、*SVC**all*、*PendSV*、*SysTick* 和中断(IRQ)。

- Reset**

复位是在上电或热复位时被调用的。该异常模组将复位作为一种特殊形式的异常。当复位发生时，处理器在任何操作的指令点都会停止。当复位结束，处理器会从向量表的复位入口重新启动。此时的工作模式将是线程模式。
- NMI**

NMI 可被软件通过外围设备发出信号或由软件触发生成。

该异常是除复位以外最高优先级的异常。该异常会被永久使能，优先级保持在-2。NMIs 不可以:

 - 被其它异常屏蔽或阻止启动。
 - 被其它除复位外异常抢占。
- HardFault**

HardFault 是正常工作中或异常处理时发生了错误产生的异常。*HardFault* 固定的优先级是-1，代表着它们拥有在可配置优先级的异常中最高的优先级。
- SVC***all*

监督者响应(SVC)异常是通过 SVC 指令触发的。在系统环境下，应用程序能使用 SVC 指令访问 OS 内核功能和设备驱动。
- PendSV**

PendSV 是用于系统级设备的中断驱动请求。在系统环境下，在没有其它异常发生时，使用 *PendSV* 进行上下文切换。
- SysTick**

当设备使用了 *SysTick* 定时器，*SysTick* 异常会在系统定时器计数到零时产生。

软件也可以产生 *SysTick* 异常。在系统环境下，设备可以使用该异常作为系统时基。
- 中断 (IRQ)**

中断，或 IRQ，是一种可被外围设备信号或软件请求产生的异常。所有的中断都是异步执行指令，在系统中，外设使用中断与处理器进行通信。

10.3.3. 异常处理程序

处理器通过以下方式处理异常：

- **ISRs**
IRQ 中断异常通过 ISRs 处理。
- **错误处理程序**
HardFault 异常是唯一一个通过错误处理程序处理的异常。
- **系统处理程序**
NMI, *PendSV*, *SVCALL*, *SysTick* 和 *HardFault* 都通过系统处理程序处理。

10.3.4. 中断优先级

用于中断的优先级方案中，有 4 个中断优先级。CPU 寄存器，**IPR0-7**, **SHPR2** 和 **SHPR3** 优先级位决定了各个中断的优先级。

中断优先级寄存器为每个中断提供 8 位优先级空间，为每个寄存器保留 4 个优先级空间。处理器只执行每块空间的位[7:6]，位[5:0]会被读作 0 并忽略写。

高级优先级中断不会被低级优先级中断请求中断。如果同时接收到两个不同优先级的中断请求，则执行优先级较高的请求。当同时接收到两个相同优先级的中断请求，则根据内部轮询序列执行服务程序。“中断源”表格展示了同优先级下的内部轮询序列和中断向量地址，异常数字越低，优先级越高。

10.3.5. 锁定 Cortex-M0

如果执行 NMI 或 HardFault 处理程序时发生错误，或者系统在使用 MSP 在异常返回取消 PSR 的跟踪时产生总线错误，则处理器进入锁定状态。当处理器处于锁定状态时，它不执行任何指令。直到出现以下情况之一，处理器会保持锁定状态：

表 10-2. CPU 锁定退出事件

标号	锁定退出事件
1	复位了
2	被调试器中止
3	NMI 发生，且此时锁定是在 HardFault 处理程序中

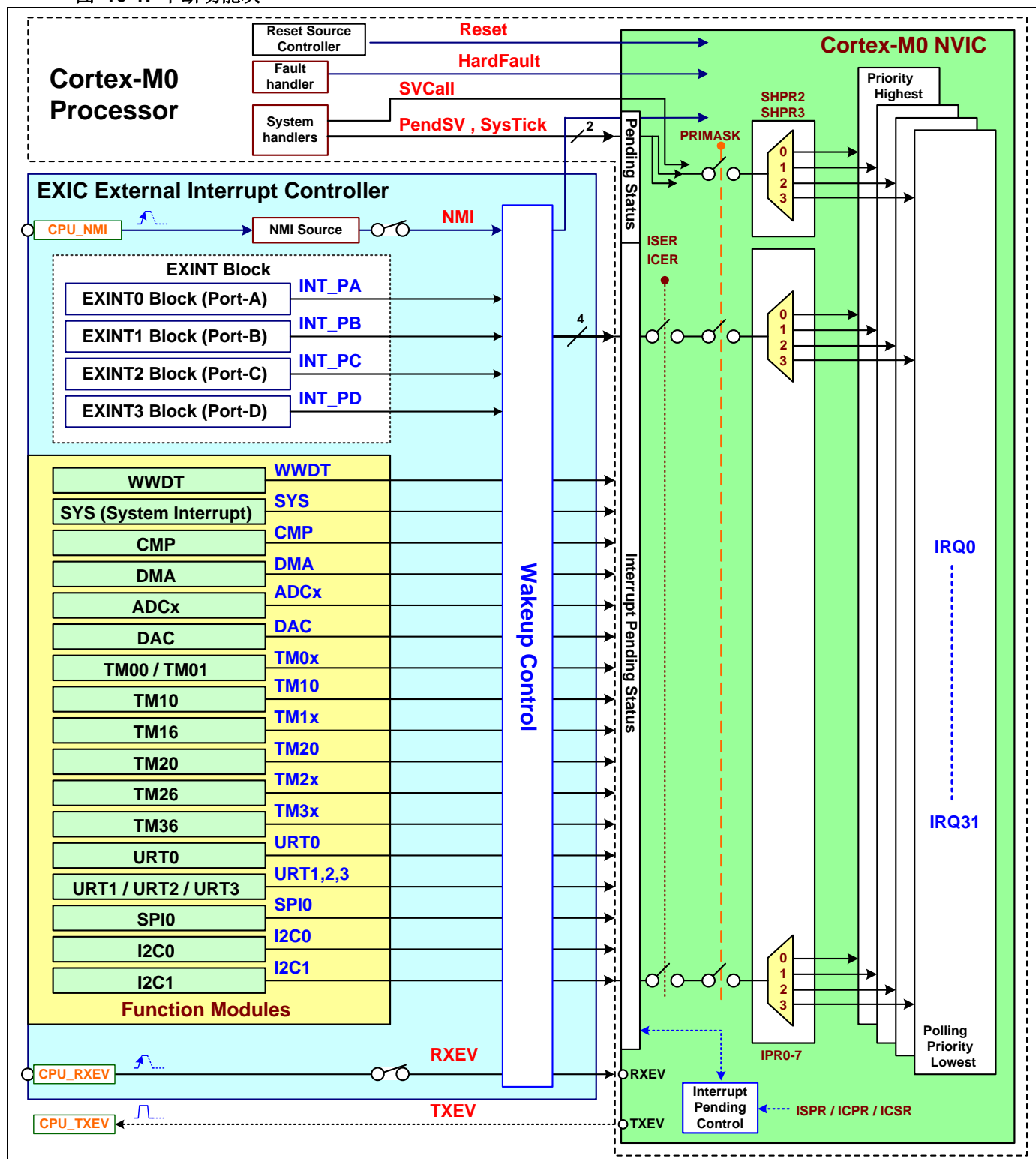
注释-1 如果在 NMI 处理程序中发生锁定状态，则后续的 NMI 不会导致处理器离开锁定状态。

10.4. 中断控制块

中断控制块包含了 Cortex-M0 NVIC 和 EXIC 控制器。下面的图表展示了中断控制块。

[注释]: EMB, DAC, TM2x, I2C1 和 URT2/3 模块仅支持于 MG32F02A132/072。

图 10-1. 中断功能块



10.5. 嵌套向量中断控制器

10.5.1. NVIC 功能

Cortex-M0 处理器集成了 1 个可配置的嵌套中断向量控制器 (NVIC)，它支持低延迟中断处理，并且包括非屏蔽中断(NMI)。NVIC 提供了 1 个零抖动中断选项和 4 个中断优先级选择。

中断处理程序不需要任何汇编程序代码，或 ISR 中删除任何代码开销。尾链优化也显著地降低了从一个 ISR 切换到另一个 ISR 时的开销。

为了优化低功耗设计，NVIC 集成了 sleep 模式。Sleep 模式包含可选的 deep sleep 模式从而使整个设备能快速进入掉电模式。

- NVIC 支持:

- 一系列定义好顺序 1-32 的中断。
- 以每步 64 的大小为每个中断配置 0-192 级优先级。
较高级别对应于较低优先级，因此级别 0 是最高中断优先级。
- 电平和脉冲检测作为中断信号。
- 中断尾链和延迟触发。
- 外部 NMI。

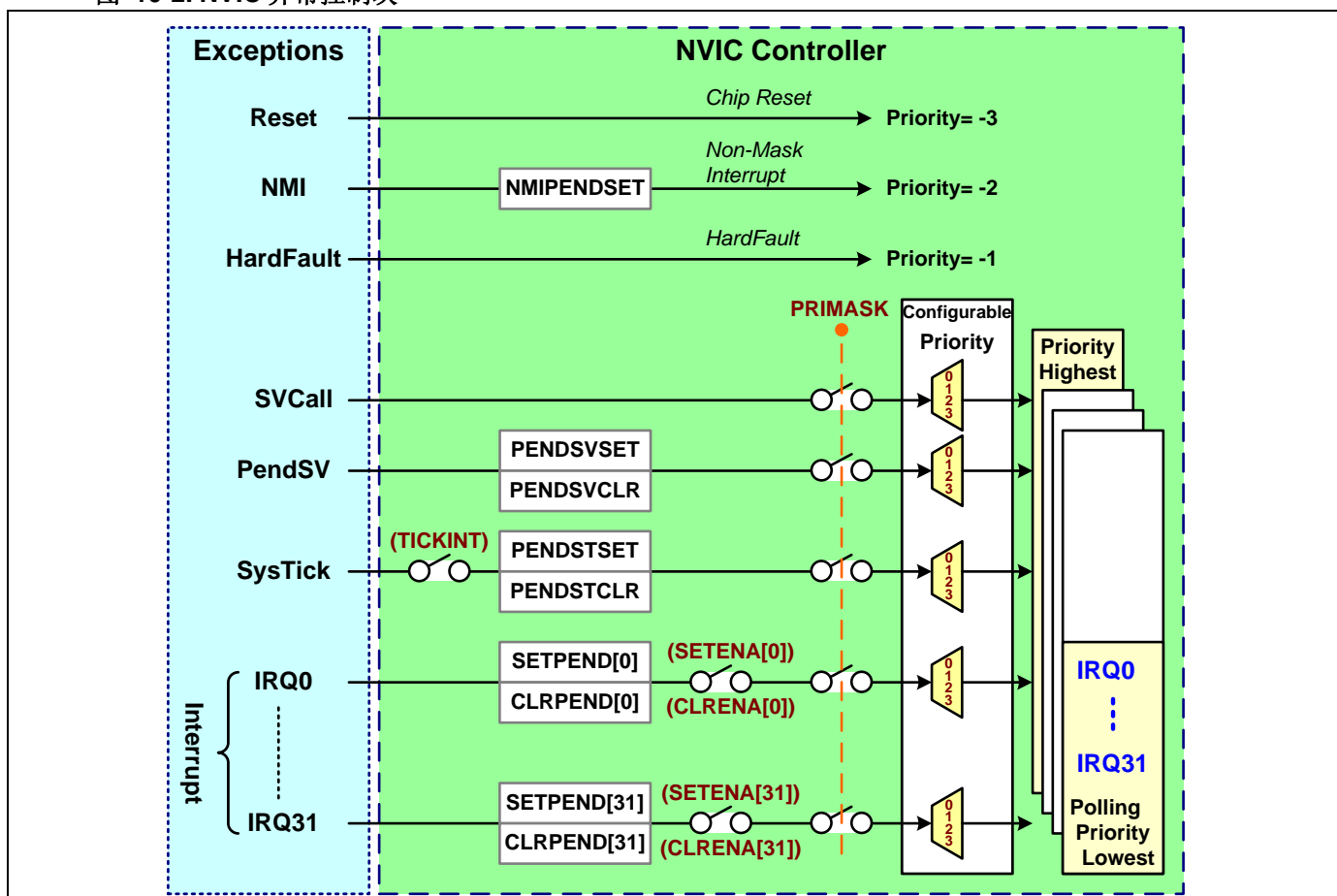
10.5.2. NVIC 异常控制

NVIC 支持 7 种异常类型：**Reset**、**NMI**、**HardFault**、**SVCall**、**PendSV**、**SysTick** 和中断(IRQ)。参照 10.3.2 节以获取更多信息。

- 异常控制

下面的图表展示了 NVIC 的异常控制块。当 **SysTick** 下溢且 **TICKINT** 位为 0 时，**SysTick** 挂起位将不会被激活，也不会置起异常请求。

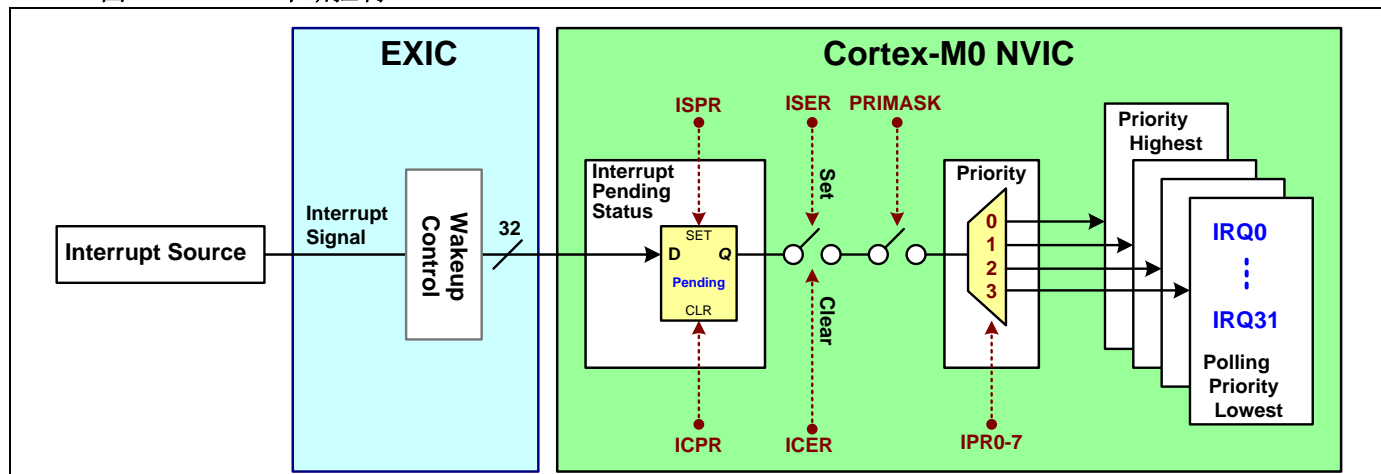
图 10-2. NVIC 异常控制块



● 中断使能和挂起

NVIC 为每个中断异常独立地提供了 CPU 寄存器的 **ISER** 和 **ICER** 来设置和清除中断使能位。此外，还为每个中断异常独立地提供了 CPU 寄存器的 **ISPR** 和 **ICPR** 来设置和清除中断挂起位。若挂起中的中断被使能，NVIC 会根据它的优先级触发中断。若中断未被使能，产生的中断信号把中断状态改变成挂起状态，但 NVIC 不会启动中断。

图 10-3. NVIC 中断控制



● 中断块状态

在每个系统时钟周期中都会对每个中断标志进行采样。若其中一个标志被置起，中断系统会产生硬件异常，只要它不受以下任何条件阻塞的话会转跳至正确的服务例程中：

- 相关中断使能位未被设置。
- 另一同优先级或更高优先级中断正在进行。

[注释]: 相关中断使能位包括中断事件使能位(*xxx_zzz_IE*), 中断源模块中断全局使能位(*xxx IEA*)和 NVIC 中断使能位。

● 中断优先级

NVIC 为提供了为每个中断提供 8 位优先级区域和保留 4 块优先级区域的中断优先级寄存器 **IPR0 ~ IPR7**。这意味着这些寄存器是被定义和对应相应中断的，这些寄存器只可进行字访问。

若中断未被使能，NVIC 不会启动中断，优先级也没有任何意义。

10.5.3. 电平和脉冲中断

NVIC 支持电平和脉冲中断。脉冲中断也可以被称为边沿触发中断。

电平中断会被保持置起直到外设释放中断信号。这通常因为 **ISR** 访问外设，导致清除中断请求时发生。脉冲中断是在处理器时钟的上升沿同步采样的中断信号。为了保证 NVIC 检测该中断，外设必须置起该中断信号至少 1 个时钟周期，在这期间，NVIC 会检测到这个脉冲并锁存该中断。

当处理器进入 **ISR** 时，它会自动从中断移除挂起状态，参照“中断硬件和软件控制”。对于电平中断，如果在处理器从 **ISR** 返回之前信号没有被清除，则中断再次变为挂起，并且处理器必须再次执行其 **ISR**。这意味着外设可以保持置起中断信号，直到不再需要使用为止。

● 中断硬件和软件控制

NVIC 锁存所有中断。由于以下原因之一，外围中断会变为挂起：

- NVIC 检测到中断信号启动，但是相应中断没启动。
- NVIC 检测中断信号的上升沿。
- 软件写相应的中断设置挂起寄存器位。

10.5.4. Hard Fault 处理

Hard faults 是异常的一种，HardFault 异常的所有错误都会被处理或着在 NMI 或 HardFault 中的处理程序中发生时导致锁定。只有 Reset 和 NMI 才能抢占固定优先级的 Hard fault 处理程序。HardFault 可以抢占除 Reset, NMI, 或其它 HardFault 外的异常。

下面的表格展示了 Cortex-M0 上的 hard fault 处理事件。

表 10-3. Cortex-M0 上的 Hard Fault 处理事件

标号	错误
1	SVC 执行指令的优先级大于等于 SVCall
2	没有调试器的情况下执行 BKPT 指令
3	在读写过程中发生系统生成的总线错误
4	从 XN 内存地址发出的执行指令
5	从系统产生总线故障的位置发出的执行指令
6	向量提取时发生系统生成的总线错误
7	执行未定义的指令
8	T 位清零后不在 Thumb 状态时执行指令
9	试图加载或存储到未对齐的地址

注释-1 HardFault 异常的所有错误都会被处理或着在 NMI 或 HardFault 中的处理程序中发生时导致锁定。

注释-2 HardFault 可以抢占除 Reset, NMI, 或其它 HardFault 外的异常

注释-3 当 HardFault 发生时，HardFault 异常服务例程通常会复位系统。

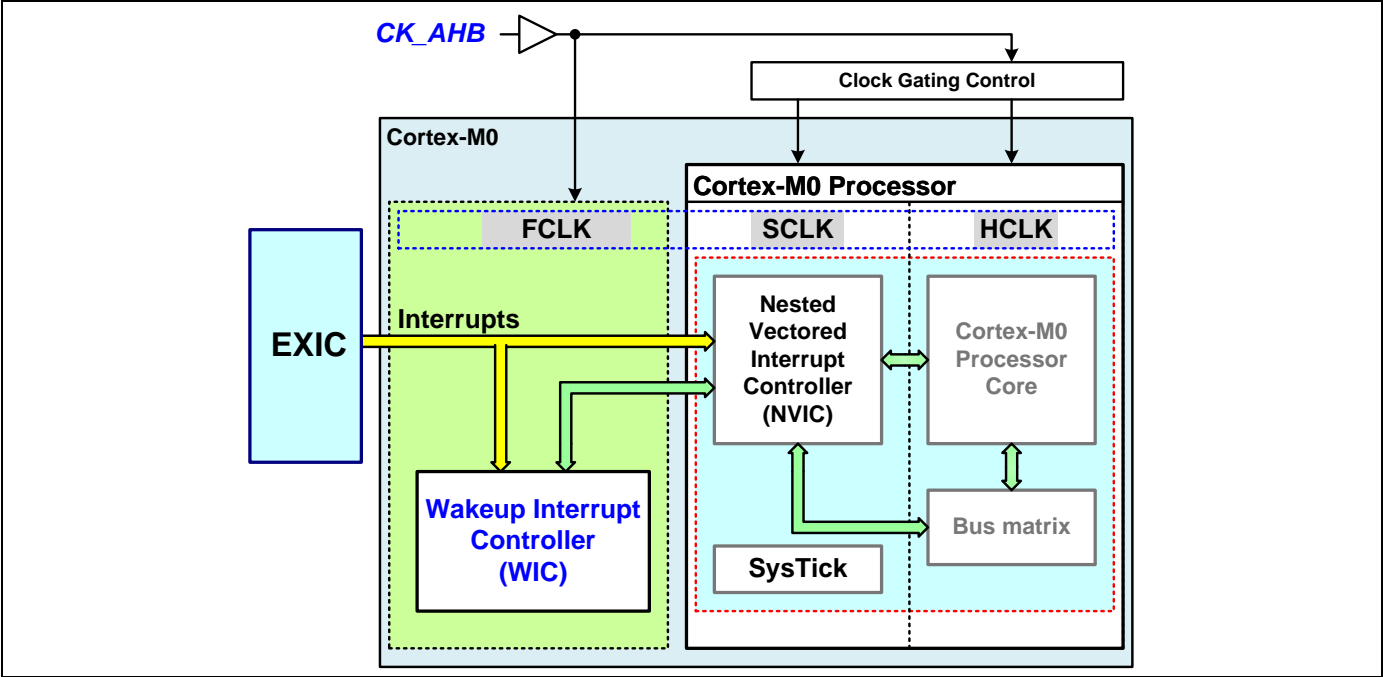
10.6. 唤醒中断控制器

该芯片包含了 1 个能检测来自 EXIC 的中断和唤醒事件并将处于深度睡眠模式的处理器唤醒的唤醒中断控制器（WIC）。只需要将 CPU 的 SCR 寄存器内的 DEEPSLEEP 位置 1，就可以使能 WIC。WIC 是不可编程的，也不含有任何寄存器或者用户接口，它是完全通过硬件信号工作的。

当 WIC 被启用并且处理器进入深度睡眠模式时，PW 控制器将关闭 CPU。这会有停止 SysTick 计时器的副作用。当 WIC 接收到中断时，它需要多个时钟周期来唤醒处理器并在它处理中断之前恢复其状态，这意味着在深睡眠模式下中断延迟时间会增加。

下面的图表展示了 WIC 控制块。

图 10-4. WIC 控制块

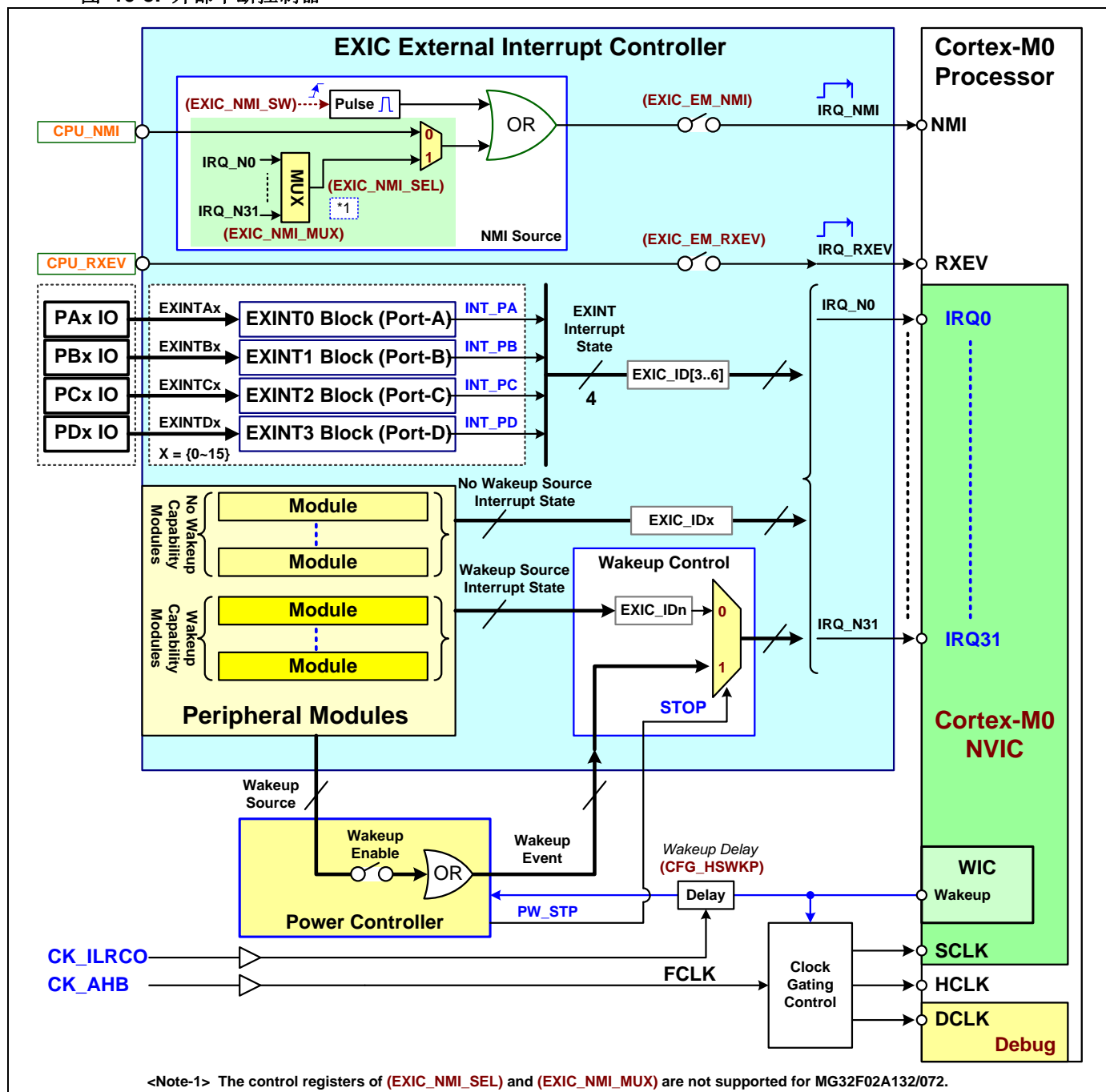


10.7. 外部中断控制器

外部中断控制器（EXIC）含有 4 个外部端口中断块（EXINT）来管理外部引脚输入中断事件，还有 1 个唤醒控制块来控制 NMI、RXEV 事件和唤醒事件。EXIC 还作为内部模块和 NVIC 之间的接口控制器，用于中断和唤醒事件管理。

下面的图表展示了 EXIC 控制块。

图 10-5. 外部中断控制器



10.7.1. EXIC 中断控制

所有产生中断的位都可以被软件设置或清除，与硬件设置或清除结果一致。也就是说，中断可以被软件产生，挂起的中断也可以被软件取消。

● 中断源识别

当中断事件发生时，用户可以读 **EXIC_ID0 ~ EXIC_ID31** 寄存器检查是哪个中断源并通过固件执行相关的事件。每个中断源都有自己独立的识别位。

参照 **EXIC_SRC0 ~ EXIC_SRC7** 寄存器描述以获取详细信息。

● NMI 控制

该信息支持 1 个 **NMI** 引脚输入并发送不可屏蔽中断(**NMI**)信号至 NVIC。**EXIC_EM_NMI** 屏蔽控制寄存器位被用于屏蔽 **NMI** 信号，软件 NMI 触发器位 **EXIC_NMI_SW** 用于测试触发 **NMI**。用户可通过设置 **EXIC_NMI_SEL** 寄存器选择 NMI 信号来自外部 **CPU_NMI** 或内部中断事件 **IRQ_Nn**。当选择内部中断时间，用户也可通过设置 **EXIC_NMI_MUX** 寄存器选择 **IRQ_Nn** 信号。

[注释]: NMI MUXs (**EXIC_NMI_MUX**, **EXIC_NMI_SEL** 寄存器)不支持于 MG32F02A132/072。外部 **CPU_NMI** 信号直接连接至内部 OR 门。

● RXEV 和 TXEV 控制

该芯片支持通过 GPIO AFS 设置 **RXEV** 引脚输入和 **TXEV** 引脚输出。当 CPU 执行 WFE 指令进入睡眠或深度睡眠且 **RXEV** 输入信号已启动时，NVIC 检测到 **RXEV** 信号上升沿时，芯片会被唤醒。**EXIC_EM_RXEV** 屏蔽寄存器位用于屏蔽 **RXEV** 信号。

当 CPU 执行指令“SEV”时，芯片会发送启动脉冲到经过 GPIO AFS 设定的 **TXEV** 输出引脚。

● 唤醒控制

EXIC 内建唤醒控制块用于接收中断源中断事件和来自 PW 控制器的唤醒事件。当芯片工作在 **ON** 模式时，唤醒控制块会把中断事件传递到 NVIC 做中断功能。当芯片工作在 **SLEEP** 模式时，唤醒控制块也会把中断事件传递到 NVIC 做 **SLEEP** 模式唤醒功能。这时，中断事件被作为唤醒事件。

当芯片工作在 **STOP** 模式时，唤醒控制块会向唤醒中断控制器(WIC)发送唤醒事件做 **STOP** 模式的唤醒功能。

参照系统电源章中“[唤醒控制](#)”节以获取更多关于唤醒功能的信息。

10.7.2. 外部端口输入中断

EXIC 控制器中一共有 4 个外部端口中断块(**EXINT**) 用于管理 GPIO 端口 A, B, C 和 D 的外部引脚输入中断事件。每个外部端口中断块都支持将一个 GPIO 端口的所有管脚配置为中断或唤醒事件源和键盘输入。它为每个引脚独立地提供的高/低电平和上升/下降沿触发选择。

EXINT 支持端口“或”逻辑用于中断，还支持端口“与”逻辑用于键盘输入 (KBI)。每个 GPIO 端口都有一个 **EXIC_Px_IEA** 中断全局使能位。

● 外部中断标志

每个 GPIO 引脚都有自己独立的中断挂起标志 **EXIC_Pxn_PF**。这个标志用于识别该事件是通过设置的电平触发还是边沿触发。该标志通过硬件置起，软件写 1 清除。

● 外部中断边沿和电平选择

每个 GPIO 引脚都有自己独立的 **EXIC_Pxn_TRGS** 寄存器设置高低电平检测或上升下降沿检测。当寄存器设置为 0 时，会禁用 **EXIC_Pxn_PF** 外部中断挂起标志更新。

使用输入信号反相寄存器位 **Px_INVn** 来选择高低电平或上下边沿触发。当 **Px_INVn=0**，**EXIC_Pxn_TRGS=0x01** 是低电平触发，**EXIC_Pxn_TRGS=0x02** 是下降沿触发。在 **STOP** 模式，即使设置的是上升沿还是下降沿触发该功能被强制通过硬件设置为电平检测模式。(x={A, B, C, D}; n={0~15})

● 外部中断屏蔽

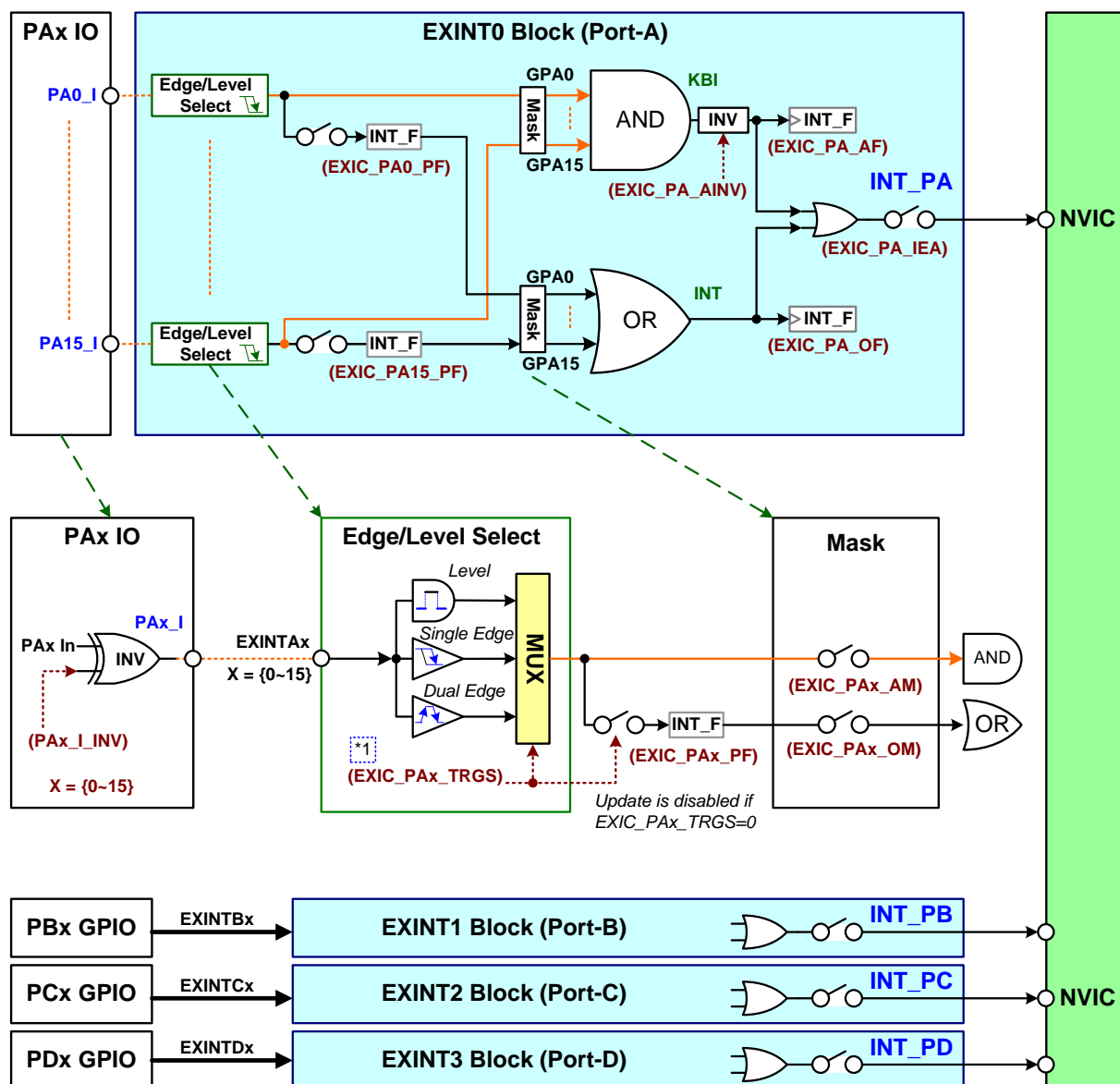
在中断功能中，在同一个端口中支持端口“或”逻辑，来“或”所有同 GPIO 端口的中断输入信号。当一个外部中断被检测到，中断挂起标志 **EXIC_Pxn_PF** 会被产生并在中断全局使能位 **EXIC_Px_IEA** 使能时置起中断。有 1 个外部中断“或”路径中断标志 **EXIC_Px_OF** 用于识别是哪个 GPIO 端口的引脚被检测。用户可以通过设置 **EXIC_Pxn_OM** 寄存器为每个 GPIO 引脚独立地屏蔽中断输入。

● 外部 KBI 屏蔽

在 KBI 功能中，在同一个端口中支持端口“与”逻辑，来“与”所有同 GPIO 端口的中断输入信号。用户可以通过设置 **EXIC_Pxn_AM** 寄存器为每个 GPIO 引脚独立地屏蔽中断输入。当所有未屏蔽外部 KBI 中断输入被检测时，“与”路径检测标志 **EXIC_Px_AF** 会被产生并在中断全局使能位 **EXIC_Px_IEA** 使能时置起中断反相控制位 **EXIC_Px_AINV** 用于反相 KBI 功能的“与”逻辑输出。

下面的图表展示了外部端口中断块。

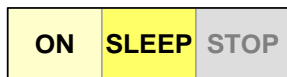
图 10-6. 外部端口中断块



<Note-1> : Set level-trigger if (EXIC_PAx_TRGS)=1 or hardware auto set level-trigger during STOP for chip.

11. GPL (通用逻辑)

11.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片内置 1 个通用逻辑（GPL）模块，它提供了数据顺序调换、数据反相、奇偶校验和 CRC 的多种功能。

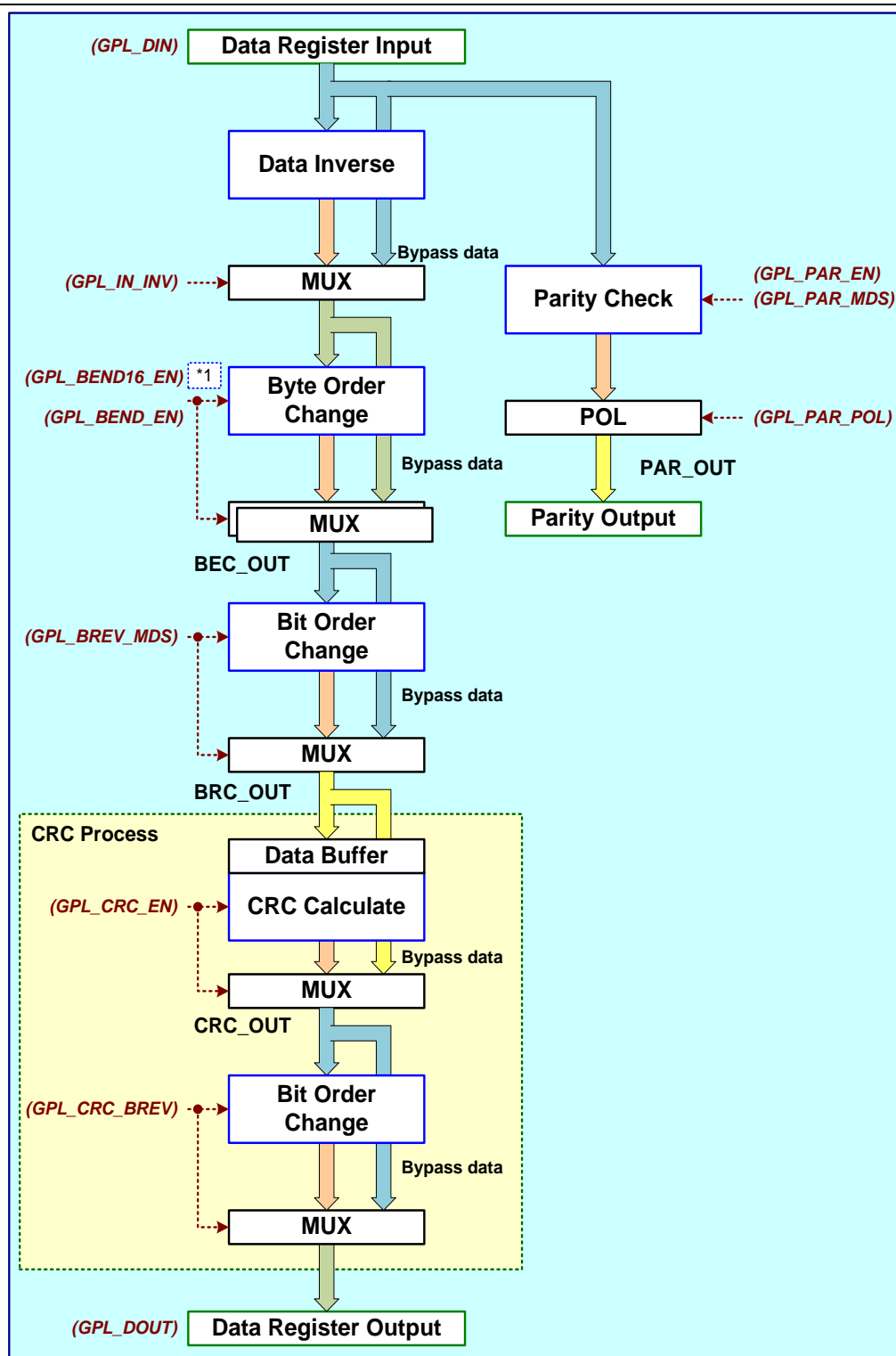
11.2. 特性

- 支持数据反相、位顺序调换、字节顺序调换和奇偶校验
 - 支持 8/16/32 位的数据位调换
 - 支持数据字节顺序进行大端规则和小端规则的调换
 - 支持 8/16/32 位奇偶校验
- 支持 CRC (循环冗余码校验) 计算
 - 可编程 CRC 初始值
 - CRC 输出位顺序调换
 - CRC 计算完成时间：32/16/8 位数据为 4/2/1 个 AHB 时钟周期
- 具有固定公共多项式的 CRC
 - CRC8 多项式 0x07
 - CRC16 多项式 0x8005
 - CCITT16 多项式 0x1021
 - CRC32(IEEE 802.3)多项式 0x4C11DB7
- 可使用 DMA 缓冲输入数据

11.3. 控制块

下面的图表展示了 GPL 控制块。

图 11-1. 通用逻辑



<Note-1> The control register of *(GPL_BEND16_EN)* is not supported for MG32F02A132/072.

11.4. 时钟

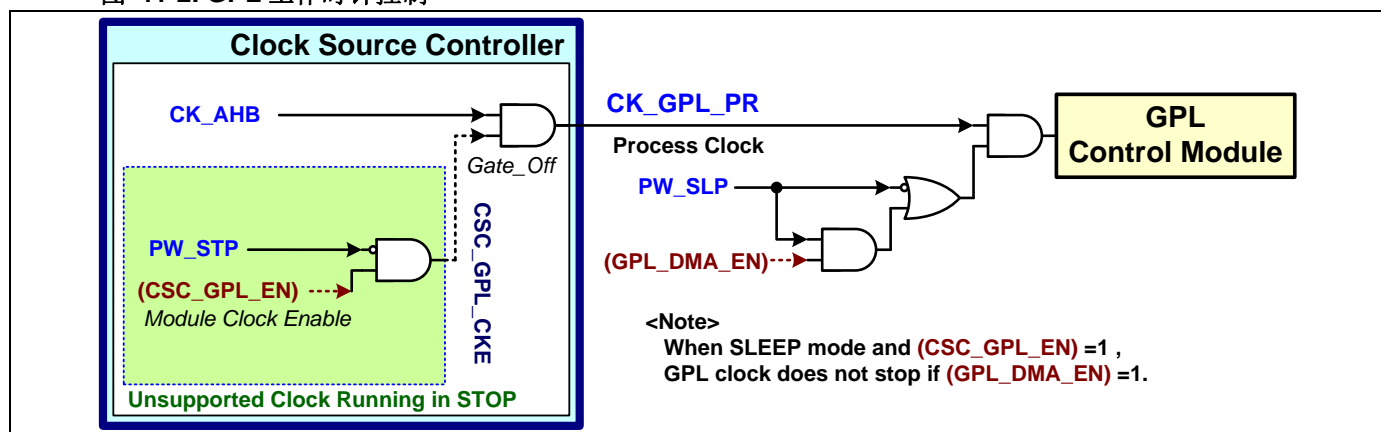
11.4.1. GPL 时钟控制

- 模块工作时钟

模块工作时钟 **CK_GPL_PR** 用于 AHB 总线和模块的接口控制逻辑。该时钟来源于 CSC（时钟源控制器）模块。它可以被 **CSC_GPL_EN** 寄存器使能。

在 **ON** 模式下，GPL 工作时钟只能在 **CSC_GPL_EN** 寄存器使能时工作。在 **SLEEP** 模式下，GPL 工作时钟只能在 **CSC_GPL_EN** 和 **GPL_DMA_EN** 寄存器同时使能时工作。通常 **GPL_DMA_EN** 寄存器使能是用于 CRC 数据通过 DMA 传输的，工作时钟不能停下来。参照系统时钟章以获取更多信息。在 **STOP** 模式下，GPL 工作时钟是保持关闭的。

图 11-2. GPL 工作时钟控制



11.5. GPL 功能控制

GPL_DIN 输入数据寄存器和 **GPL_DOUT** 输出数据寄存器用于 GPL 工作。写操作中，**GPL_DIN** 寄存器用于写新的计算数据；读操作中，**GPL_DOUT** 寄存器用于读上一次 CRC 的计算结果。**GPL_DOUT** 寄存器用于存储除奇偶校验以外的计算操作结果。

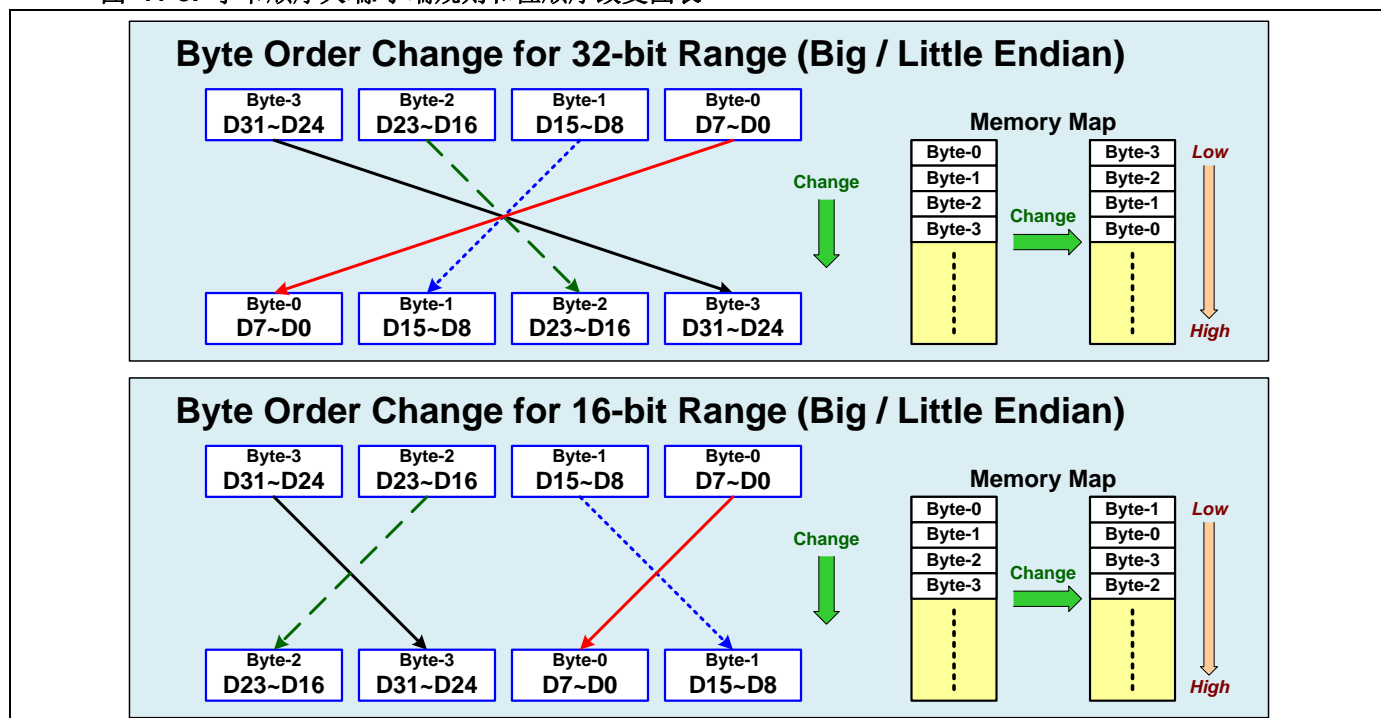
11.5.1. 字节顺序变换

GPL 可以以大端规则或小端规则的方式改变 32 位或 16 位数据的字节顺序。该操作是通过 **GPL_BEND_EN** 寄存器使能的。

[注释]: **GPL_BEND16 EN** 寄存器不支持于 MG32F02A132/072。

下面的图表展示了 GPL 字节顺序改变和位顺序改变控制块。

图 11-3. 字节顺序大端/小端规则和位顺序改变图表

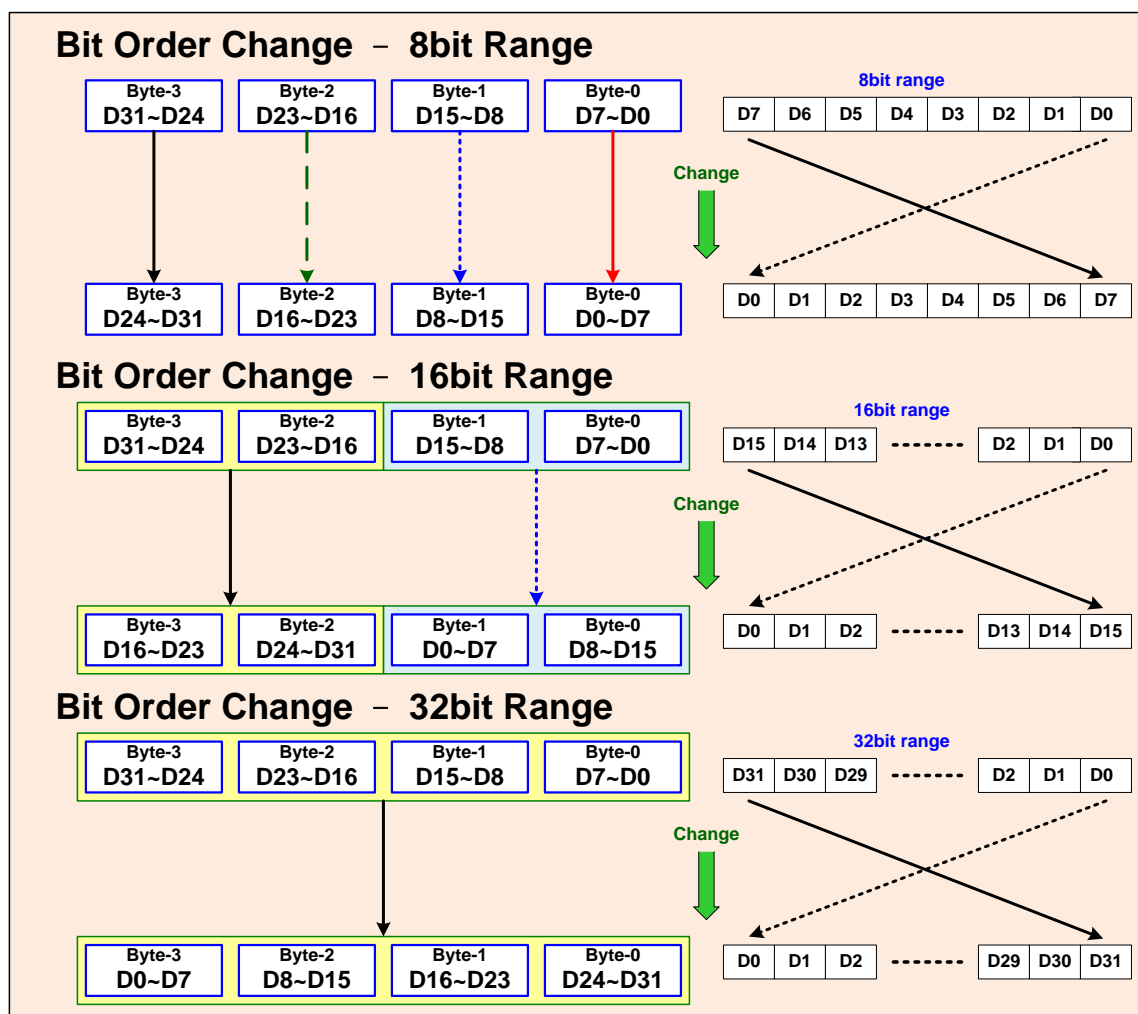


11.5.2. 位顺序改变

GPL 可以改变 8/16/32 位数据的位顺序。该操作可以通过 **GPL_BREV_MDS** 寄存器选择 8/16/32 位数据。

下面的图表展示了 GPL 位顺序改变控制块。

图 11-4. 位顺序改变图表



11.5.3. 数据反相

GPL 可以通过 **GPL_IN_INV** 寄存器反相输入数据值。

11.5.4. 奇偶校验

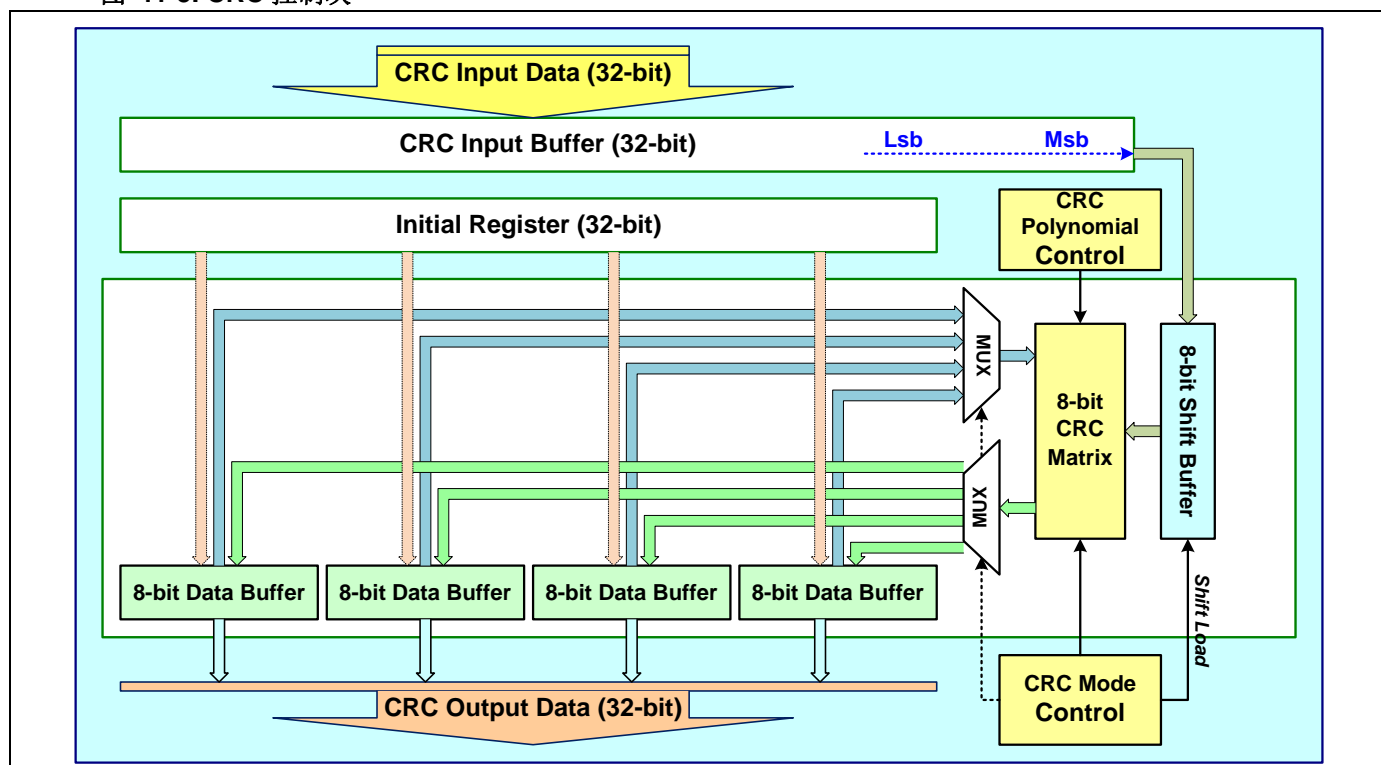
GPL 可以根据 **GPL_PAR_POL** 寄存器中通过奇数或偶数设置奇偶校验极性计算输入数据奇偶校验值。它可以同时输出 8、16、32 位输入数据的计算结果到 **GPL_PAR8_OUT**, **GPL_PAR16_OUT** 和 **GPL_PAR32_OUT** 寄存器中。

11.5.5. 循环冗余校验计算

CRC (循环冗余校验) 块用于获得 8/16/32 位 CRC 数据并通过 CRC 多项式控制块计算数据缓冲的结果。CRC 块可以持续保持处理 CRC 代码，并将最后的结果存储到数据缓冲区中。

下面的图表展示了 CRC 控制块。

图 11-5. CRC 控制块



CRC 通过设置 **GPL_CRC_EN** 寄存器使能，并通过设置 **GPL_CRC_MDS** 寄存器选择 CRC 多项式模式。CRC 控制块支持以下公共固定多项式。

图 11-6. CRC 多项式

CRC-CCITT	$X^{16} + X^{12} + X^5 + 1$
CRC-8	$X^8 + X^2 + X + 1$
CRC-16	$X^{16} + X^{15} + X^2 + 1$
CRC-32	$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

CRC 能通过设置 **GPL_CRC_DSIZE** 寄存器选择 CRC 的输入数据大小：8/16/32 位。此外，用户可以通过设置 **GPL_CRC_INIT** 寄存器设置 CRC 计算初始值。

位顺序改变块用于改变 CRC 输出数据的位顺序。该控制块可以通过设置 **GPL_CRC_BREV** 寄存器选择数据大小：8/16/32 位。

11.6. GPL DMA 操作

11.6.1. DMA 模块设置

当芯片支持 DMA（直接内存访问）控制器时，用户可以在 DMA 数据传输前，在 DMA 模块中设置关于来源/目的设备、通道请求仲裁等 DMA 设置。DMA 来源和目的可以是内存或外设。

参照 DMA 章以获取更多关于 DMA 模组设置的细节。

11.6.2. GPL DMA 控制

DMA 设置完成后，用户需设置 GPL 模块的 DMA 使能位 **GPL_DMA_EN**。

最后，需要相关的通道请求起始位 **DMA_CHn_REQ** 用于设置 DMA 传输启动($n = \text{DMA 通道标号}$)。然后传输源和目的设备会置起 RX/TX 请求信号到 DMA 控制器，DMA 控制器便会置起接收信号到请求源/目的设备。此时，该数据传输连接是用于 DMA 传输的。

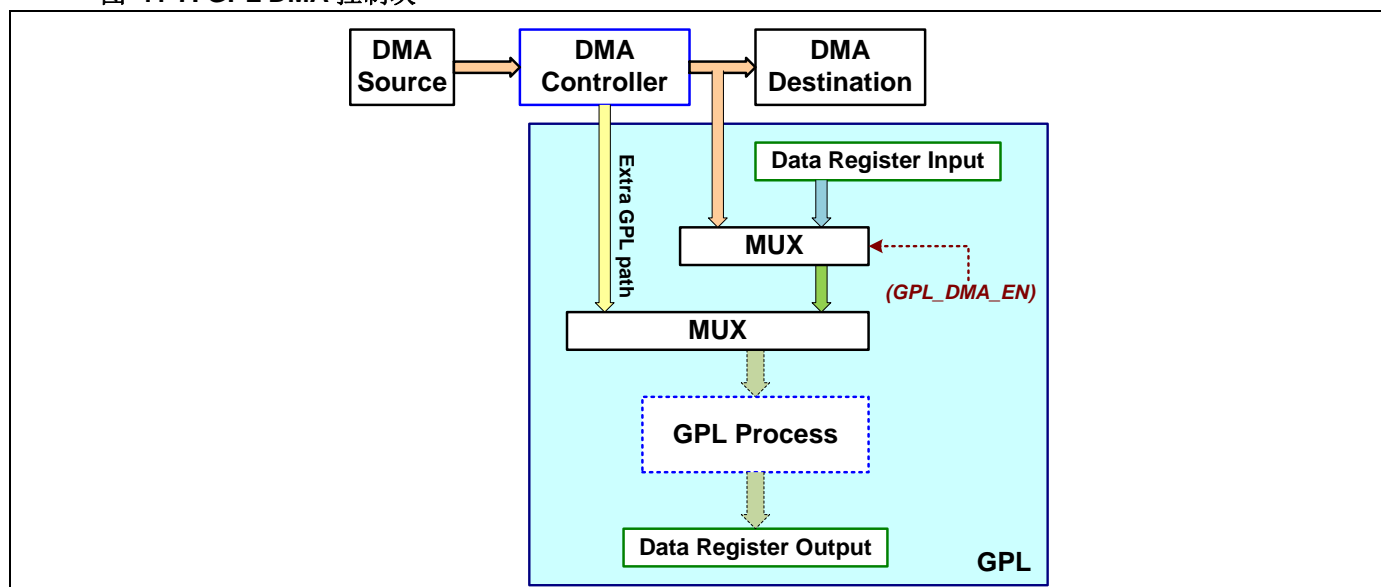
寄存器位 **GPL_DMA_EN** 用于使能来自 DMA 源到 GPL 模块进行 GPL 操作的数据传输。当 **GPL_DMA_EN** 位被使能，输入数据路径会被切换到 DMA 目的地数据路径，取代 GPL 寄存器输入。DMA 操作完成后，硬件将会自动禁用 **GPL_DMA_EN** 位。

当 DMA 源和 DMA 目的地正在进行 DMA 操作，DMA 支持额外的 GPL 路径把目的地的数据直接发送到 GPL，做 GPL 操作。该操作是 GPL 硬件自动控制的，且不需要设置使能 **GPL_DMA_EN** 位。用户可以通过设置 **DMA_GPL_CHS** 寄存器为额外 GPL 数据路径选择 DMA 通道源。

当 DMA 源为内嵌 FLASH 且 DMA 目的地为 GPL，用户可通过 **DMA_FGBUS_SEL** 寄存器设置 DMA 发送总线宽度 8 位或 32 位。当 GPL DMA 被使能，且 **DMA_FGBUS_SEL=0** 时，**GPL_CRC_DSIZE** 寄存器被硬件固定为 8 位；当 GPL DMA 被使能，且 **DMA_FGBUS_SEL=1** 时，**GPL_CRC_DSIZE** 寄存器被硬件固定为 32 位。

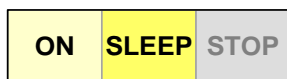
[注释]: 在 MG32F02A132/072 中, **DMA_FGBUS_SEL** 寄存器不被支持且恒为 8 位 DMA 发送总线宽度。

图 11-7. GPL DMA 控制块



12. DMA (直接内存访问)

12.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片内置 1 个直接内存访问控制器 (DMA) 用于加强外设-内存、内存-内存、外设-外设的数据传输。数据可以在不使用 CPU 资源的情况下快速的通过 DMA 传输。

注释: (n= DMA 通道标号) 会被用于该章的寄存器、信号和引脚/端口描述中。[EX]: **DMA_CHn_EN** ~ n 表明通道标号。

12.2. 特性

- DMA 传输类型
 - 内存-内存
 - 外设-内存
 - 内存-外设
 - 外设-外设
- 传输数据量可设置最高 65535 字节
- 单次数据宽度 1,2,4 字节
- 支持循环发送模式和自动重载起始地址控制
- 为引脚触发请求提供 single/block/demand 模式

❖ MG32F02A132/072

- 3 个独立的可配置通道专用硬件相应 DMA 请求
 - 访问内存、APB 和 AHB 外设作为源和目的地
 - 支持 SRAM/EMB 访问存储空间作为内存源和目的地
 - 外设包含 ADC0、DAC、I2Cx、URTx、SPIx、TM36 和 GPL 模块
- 内置 2 种优先级控制用于通道请求
 - 轮询请求
 - 软件设置优先级

❖ MG32F02A032

- 1 个独立的可配置通道专用硬件相应 DMA 请求
 - 访问内存、APB 和 AHB 外设作为源和目的地
 - 存储器包含 SRAM、EMB 访问存储空间
 - 外设包含 ADC0、I2Cx、URTx、SPIx、TM36 和 GPL 模块

12.3. 配置

12.3.1. 芯片配置

下面的表格展示了芯片 DMA 通道配置。

表 12-1. DMA 配置

芯片	DMA 模块		
	通道 0	通道 1	通道 2
MG32F02A132	V	V	V
MG32F02A072	V	V	V
MG32F02A032	V		

注释 V: 包含

12.4. 控制块

DMA 控制器（DMA）用于 AHB 外设、APB 外设、SRAM 和外部存储器这些源和目的地之间传输数据。DMA 中断块用于 DMA 事件的检测和服务。同时，DMA 有仲裁的功能，用于处理 DMA 请求的优先级。

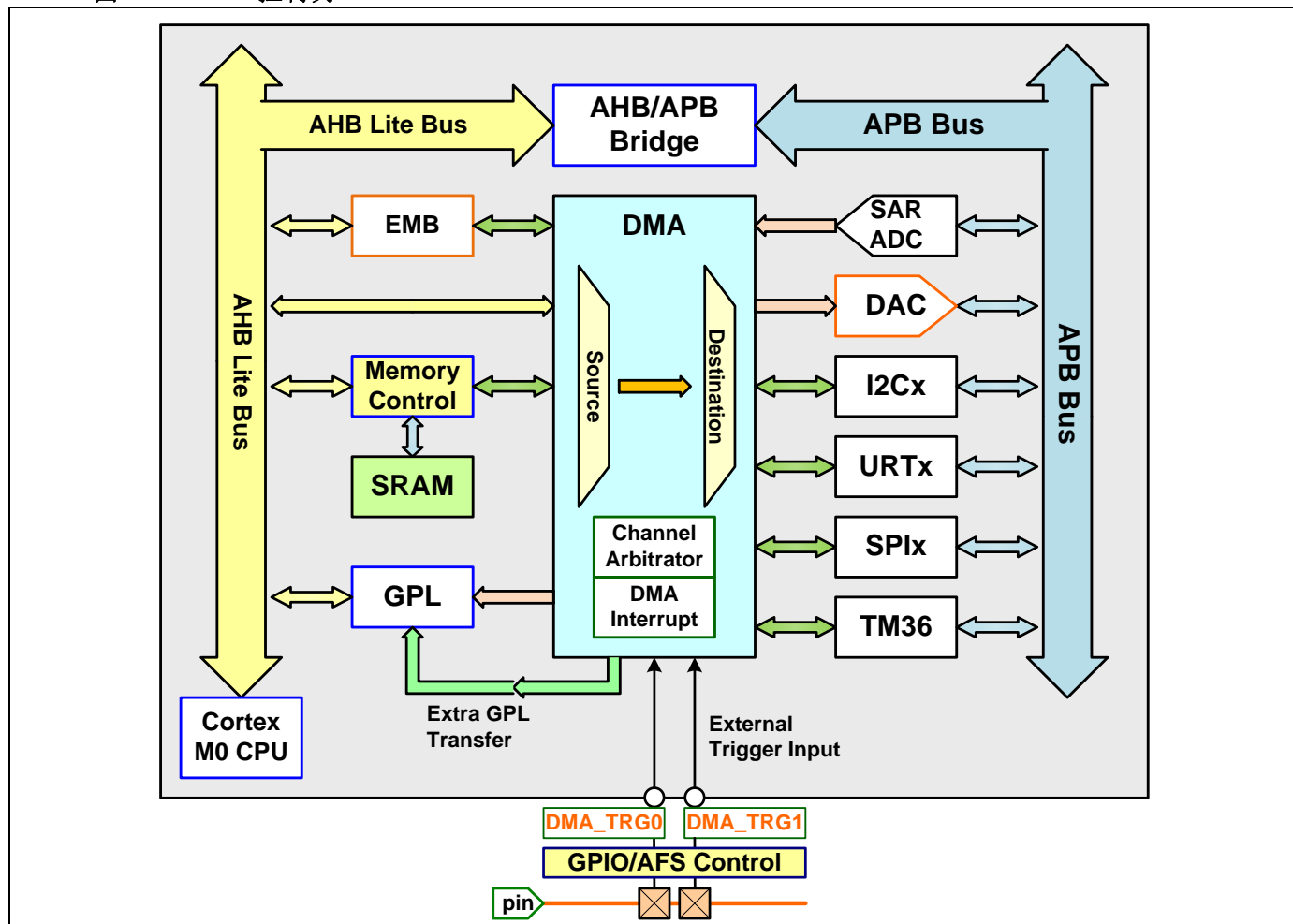
DMA_TRG0 和 **DMA_TRG1** 这两个外部引脚能够作为 DMA 数据传输的触发信号输入。

[注释]: MG32F02A032 不支持 **DMA_TRG1**。

下面的图表展示了 DMA 的连接图。

[注释]: DAC, EMB 模块仅支持于 MG32F02A132/072。

图 12-1. DMA 控制块



12.5. IO 线

12.5.1. IO 信号

● DMA_TRGn

这是 DMA 外部触发信号输入，并用于触发 single/block/demand 模式下 DMA 数据传输。

12.5.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。参照 GPIO 章中“[IO 模式](#)”节以获取更多关于 IO 模式设置的信息。

每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“[功能复用选择](#)”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“引脚功能复用表”以获取更多信息。

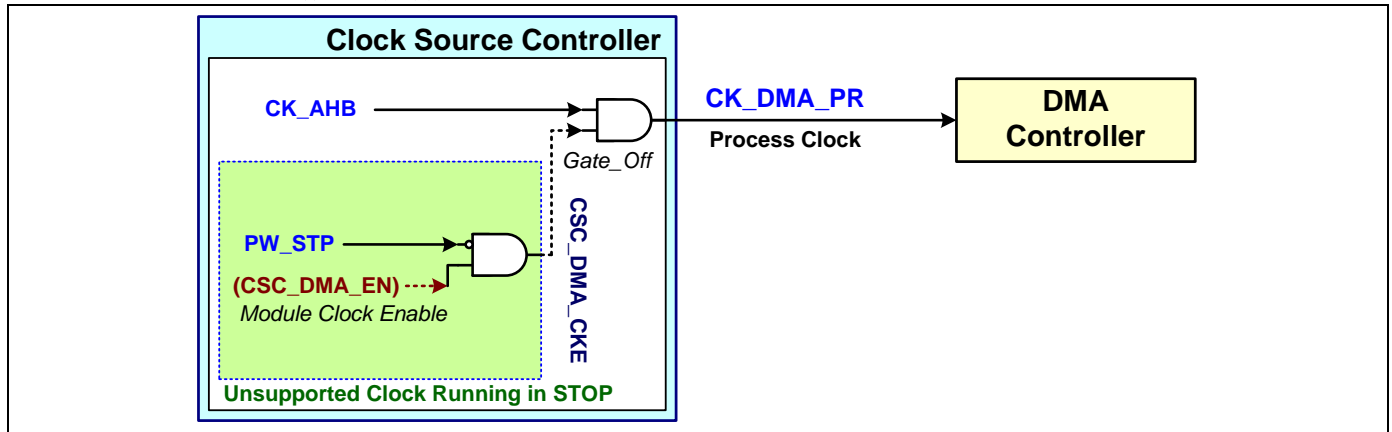
12.6. 使能和时钟

该模块的功能全局使能位是 **DMA_EN**。当该位被禁用时，所有的 DMA 功能将无法工作。

该模块工作时钟 **CK_DMA_PR** 是用于 AHB 总线和模块的接口逻辑控制。该时钟来源于 CSC (时钟源控制器) 模块。该时钟可通过 **CSC_DMA_EN** 寄存器使能。

ON 和 **SLEEP** 模式下，工作时钟只会在 **CSC_DMA_EN** 寄存器被使能时运行。**STOP** 模式下，工作时钟会被禁用，参照系统时钟章以获取更多信息。

图 12-2. DMA 工作时钟控制



12.7. 中断和事件

DMA 模块能产生 **INT_DMA**。**INT_DMA** 发送到外部中断控制器 (EXIC) 作中断事件。

12.7.1. DMA 中断控制和状态

中断标识是用于中断服务程序 (ISR) 流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **DMA_IEA** 用于使能和禁用该模块的所有中断源。

只读的全局中断标志状态 **DMA_CHn_GIF** 是用于各个 DMA 通道的。该状态位被置起是标志着在该通道中检测到中断事件。参照相关状态位寄存器描述以获取更多信息。

12.7.2. DMA 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

● TCF / TC2F

DMA 传输完成标志是 (**DMA_CHn_TCF**)。相关中断使能寄存器位是 **DMA_CHn_CIE**。每个通道都有 1 个独立标志和 1 个独立中断使能位。当 DMA 数据从源到目的地的传输完成时，会置起该标志。TC2F 标志 (**DMA_CHn_TC2F**) 与 TCF 标志完全一样，也可以被固件使用。每个通道的 TCF 标志会被在 **DMA_STA** 寄存器中分组。TC2F 标志与 THF, ERRF 标志分为一组，**DMA_CHnA** 寄存器中的相关中断使能位都是每个通道独立设置的。

● THF / TH2F

DMA 传输一半标志是 (**DMA_CHn_THF**)。相关中断使能寄存器位是 **DMA_CHn_HIE**。每个通道都有 1 个独立标志和 1 个独立中断使能位。当 DMA 数据从源到目的地的传输完成一半时，会置起该标志，置起该标志时，用户可以预备下次 DMA 传输的数据。TH2F 标志 (**DMA_CHn_TH2F**) 与 THF 标志完全一样，也可以被固件使用。每个通道的 THF 标志会被在 **DMA_STA** 寄存器中分组。TH2F 标志与 TCF, ERRF 标志分为一组，**DMA_CHnA** 寄存器中的相关中断使能位都是每个通道独立设置的。

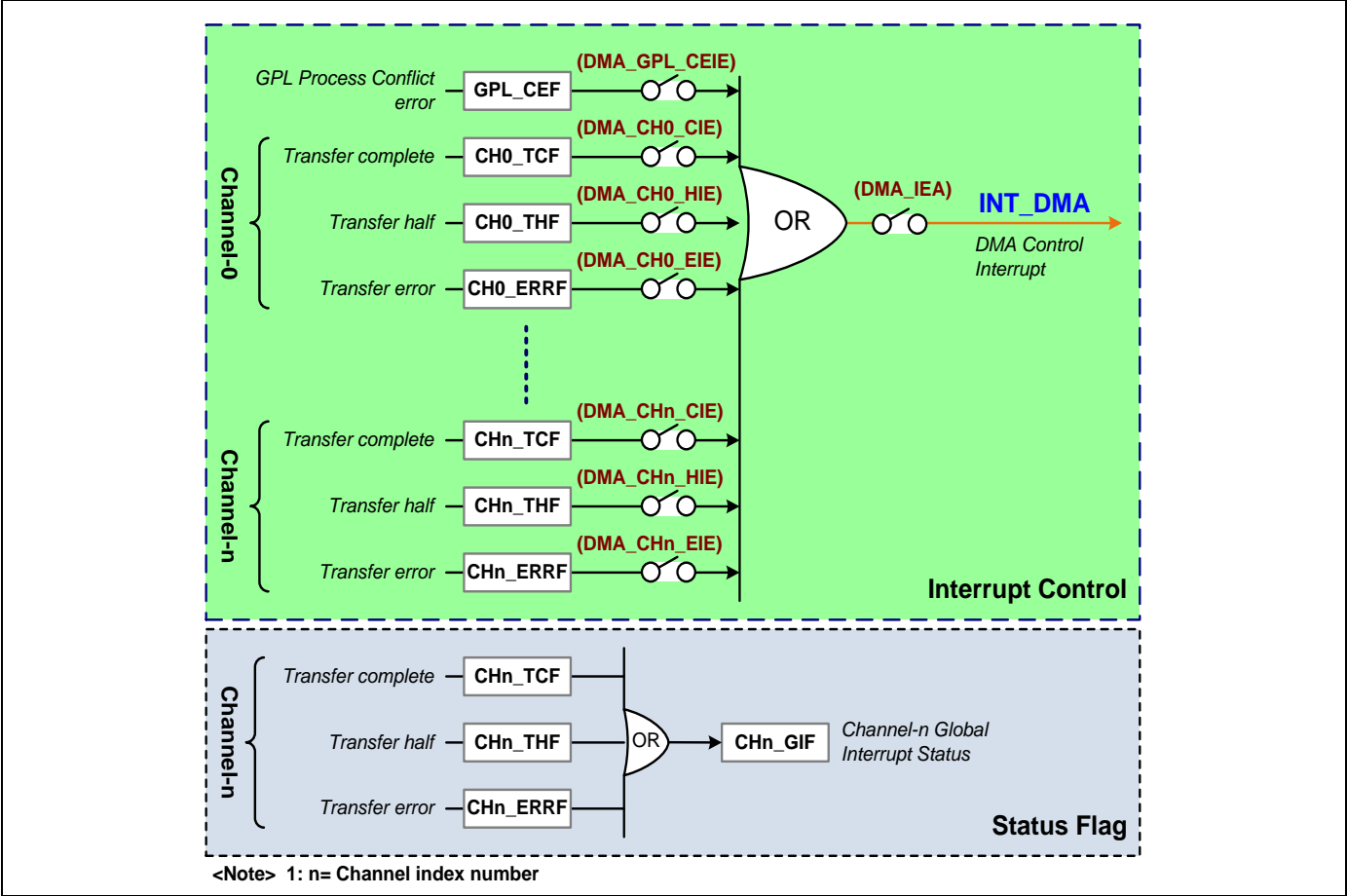
● ERRF / ERR2F

DMA 传输错误标志是 (**DMA_CHn_ERRF**)。相关中断使能寄存器位是 **DMA_CHn_EIE**。每个通道都有 1 个独立标志和 1 个独立中断使能位。当 DMA 数据从源到目的地的传输完成 (**DMA_CHn_CNT = 0**) 且外设仍执行 DMA 请求，从而发生通道传输错误时，会置起该标志。

ERR2F 标志 (**DMA_CHn_ERR2F**) 与 ERRF 标志完全一样，也可以被固件使用。每个通道的 ERRF 标志会被在 **DMA_STA** 寄存器中分组。ERR2F 标志与 TCF, THF 标志分为一组，**DMA_CHnA** 寄存器中的相关中断使能

位都是每个通道独立设置的。

图 12-3. DMA 中断控制



[注释]: MG32F02A032 仅支持通道-0。

12.8. DMA 控制

12.8.1. DMA 源和目的地

DMA 控制器有最多 3 个独立通道，每个通道可以独立决定源到目的地的数据传输管理。源和目的地可以是内存或外设。每个通道都有 1 个 DMA 复用源和 1 个 DMA 目的地复用源，并独立决定 DMA 源和目的地。

内存包含内嵌 FLASH、SRAM、EMB 访问内存空间。外设包含 ADC0, DAC, I2Cx, URTx, SPIx, TM36 和 GPL 模块。ADC 模块只能作为 DMA 源。DAC 和 GPL 只能作为 DMA 目的地。

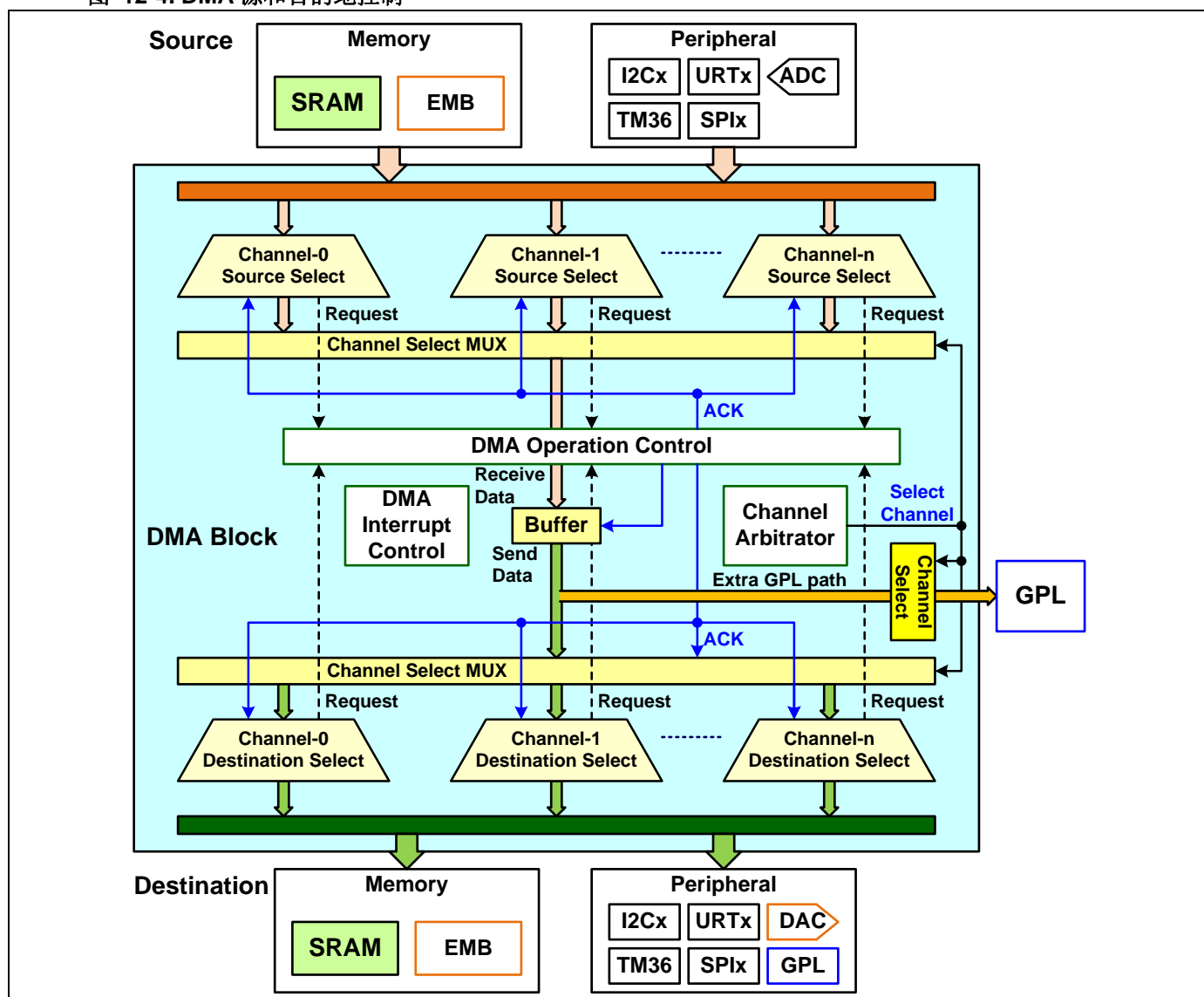
[注释]: 在 MG32F02A132/072 中，内嵌 FLASH 不支持作为 DMA 源。

特殊的是，有 1 个能复制通过设置的通道中需要传输的数据发送到 GPL 执行 GPL 工作（如 CRC）的额外的 GPL 路径。

下面的图表展示了源和目的地之间的 DMA 数据传输连接。

[注释]: DAC, EMB 模块仅支持于 MG32F02A132/072。

图 12-4. DMA 源和目的地控制



● DMA 通道源和目的地选择

每个 DMA 通道的源和目的地选择表都是一样的，用户可以设置 **DMA_CHn_SRC** 和 **DMA_CHn_DET** 寄存器设置 DMA 源和目的地。

后缀“RX”指的是源数据是来自模块接收接口；后缀“TX”指的是目的地数据会被发送到模块的发送接口。此外，后缀“Read”指的是该源数据是通过读获取的；后缀“Write”指的是该目的地数据会被用于写。“ADC0_IN”指的是该源数据是从 ADC 转换输出中输入的。

“DAC_OUT”指的是目的地数据会被输出到 DAC 转换输出的输入。TM36 可发送从定时器输入捕获的数据的 DMA 源数据和接收目的地数据到 3 个中的 1 个输出比较重载寄存器。

下面的表格展示了 DMA 通道源和目的地选择表。

表 12-2. DMA 通道源/目的地请求选择

源请求选择				目的地请求选择			
寄存器	MG32F02A132 MG32F02A072	MG32F02A032	类型	寄存器	MG32F02A132 MG32F02A072	MG32F02A032	类型
CHn_SRC				CHn_DET			
0	MEM_Read	MEM_Read	内存	0	MEM_Write	MEM_Write	内存
1	ADC0_IN	ADC0_IN	外设	1	DAC_OUT	-	外设
2	I2C0_RX	I2C0_RX		2	I2C0_TX	I2C0_TX	
3	I2C1_RX	-		3	I2C1_TX	-	
4	URT0_RX	URT0_RX		4	URT0_TX	URT0_TX	
5	URT1_RX	URT1_RX		5	URT1_TX	URT1_TX	
6	URT2_RX	-		6	URT2_TX	-	
7	URT3_RX	-		7	URT3_TX	-	
8	SPI0_RX	SPI0_RX		8	SPI0_TX	SPI0_TX	
9	-	-		9	-	-	
10	-	-		10	-	-	
11	-	-		11	GPL_Write	GPL_Write	
12	-	-		12	TM36_CC0B	TM36_CC0B	
13	-	-		13	TM36_CC1B	TM36_CC1B	
14	-	-		14	TM36_CC2B	TM36_CC2B	
15	TM36_IC3	TM36_IC3		15	-	-	

注释-1：“-” = 保留，CHn：n = 通道标号

注释-2：只有通道-0 支持内存到内存传输

● DMA 通道工作类型支持

DMA 控制器支持内存到内存、外设到内存、内存到外设、外设到外设的工作类型。用户可以通过直接设置通道源和目的地设置通道工作类型。只有 DMA 通道-0 支持内存到内存传输。

下面的表格展示了各个通道支持的 DMA 通道工作类型。

表 12-3. DMA 通道工作类型支持

DMA 工作类型		通道 0	通道 1	通道 2
源	目的地			
内存	内存	V	X	X
内存	外设	V	V	V
外设	内存	V	V	V
外设	外设	V	V	V

注释-1：“V” = 支持，“X” = 不支持

● DMA ADC 发送

当 DMA 源是 ADC 数据输出时，用户可通过 **ADCx_DMA_DSIZE** 寄存器设置 DMA 发送数据包大小为 16 位

或 32 位。参照“[ADC DMA 控制](#)”节以获取更多信息。

[注释]: 在 MG32F02A132/072 中 **ADCx DMA DSIZE** 寄存器不被支持且恒为 32 位数据包大小。

● DMA Flash 到 GPL

当 DMA 源是内嵌 FLASH 且 DMA 目的地是 GPL 时，用户可通过 **DMA_FGBUS_SEL** 寄存器设置 DMA 发送总线数据宽度为 8 位或 32 位。当选择 1 字节，则每次发送数据周期的字节数为 1 字节，当选择 4 字节时，每次发送数据周期的字节数为 4 字节。用户可设置仅 4 字节用于内存到 GPL DMA 数据发送。它需要为其他 DMA 数据传输条件设置 1 个字节。请参照“[GPL DMA 控制](#)”节以获取更多信息。

[注释]: 在 MG32F02A132/072 中 **DMA_FGBUS_SEL** 寄存器不被支持且恒为 8 位发送总线宽度大小。

● DMA 额外 GPL 数据路径

芯片中有 1 个能通过设置的通道复制需要传输的数据并发送到 GPL 执行 GPL 工作（如 CRC）的额外的 GPL 路径。用户可以通过使能 **DMA_GPL_CHS** 寄存器来使能 GPL 功能和选择 DMA 通道。被选择的通道将会处理从请求源到目的地的 DMA 操作。

12.8.2. DMA 通道仲裁

DMA 控制器内置 1 个通道仲裁。它在通道请求中提供了 2 种优先级控制，且用户可以通过 **DMA_PRI_MDS** 寄存器设置。一种仲裁方式是“轮询”，另一种则是软件设置“优先级”。

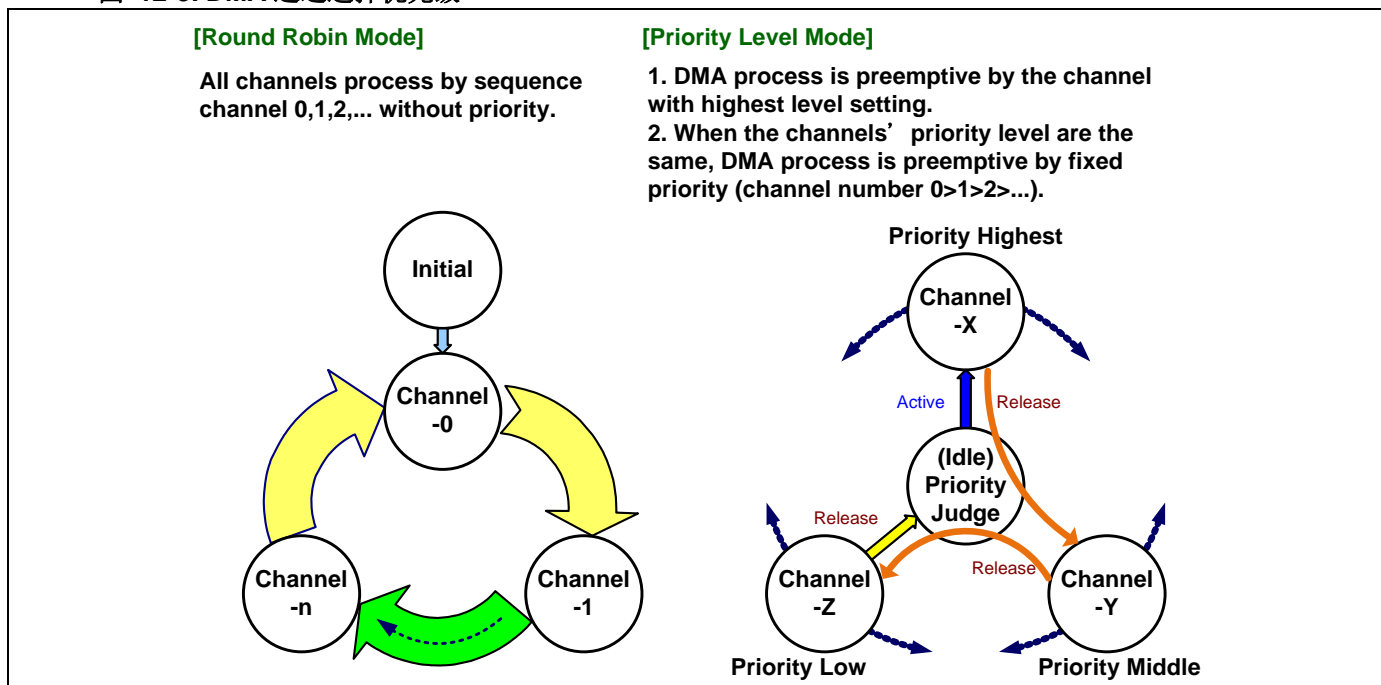
[注释]: MG32F02A032 由于只支持 1 个 DMA 通道，因此不含有通道仲裁。

当选择“轮询”时，第一个工作通道是通道 0，通道 0 完成一次数据传输时，下一个工作通道则会被改为通道 1，后面通道又会变成通道 2，然后回到通道 0，如此循环。

当选择“优先级”模式，做法就会与中断优先级控制方式非常相似，优先级可以通过 **DMA_CHn_PLS** 寄存器为每个通道独立地设置。

下面的图表展示了两种通道优先级控制。

图 12-5. DMA 通道选择优先级



12.8.3. DMA 通道工作

● DMA 通道使能

每个 DMA 通道都有 **DMA_CHn_EN** 通道使能位用于工作控制。当该位被禁用时，该 DMA 通道操作将无法进行，并进入通道复位状态。每个通道的伪控制位 **DMA_CHn_ENB** 都有自己的完全一样的寄存器位 **DMA_CHn_EN**。所有的伪控制位都被包含进了 **DMA_CR0** 寄存器，方便固件调用。

● DMA 通道传输单位大小

每个 DMA 通道都可以通过 **DMA_CHn_BSIZE** 寄存器设置传输单位数据大小。单位数据大小是在传输周期中不能被通道仲裁中断的顺序数据字节数。

单位数据大小可以根据外设缓冲大小或 EMB 总线宽度设置为 1/2/4 字节。参照表“DMA 传输数量/起始地址和单位大小设置注释”以获得更多信息。比如，当 EMB 总线宽度为 16 位时单位大小不可设置为 1 字节。

● DMA 通道传输总字节数

每个 DMA 通道都可以通过 **DMA_CHn_NUM** 寄存器设置每次 DMA 数据传输工作的总传输字节数。值 0 代表没有数据可进行传输，0xFFFF 则是传输 65535 字节数据。该寄存器的值必须与 **DMA_CHn_BSIZE** 寄存器设置的大小成整数倍。

DMA 控制器为每个独立通道提供了总计数器 **DMA_CHn_CNT** 用于读取剩下剩余需要传输的数据量。在每次 DMA 单位传输之后，这个寄存器被减少 **DMA_CHn_BSIZE** 寄存器中的单位大小。

● DMA 通道源和目的地起始地址

当通道工作在内存到内存、外设到内存、内存到外设、外设到外设的工作类型时，每个 DMA 通道都可以在每次 DMA 数据传输时，通过 **DMA_CHn_SSA** 和 **DMA_CHn_DSA** 寄存器设置源或目的地内存起始地址。当 DMA 源或目的地为外设时，相关的源或目的地内存起始地址是无效的。

参照表格“DMA 传输数量/起始地址和单位大小设置注释”，若单位大小为 4 字节时，源和目的地内存起始地址必须是字对齐，相同的，若单位大小为 2 字节，它们必须是半字对齐。

DMA 控制器为每个独立通道提供了 **DMA_CHn_SCA** 和 **DMA_CHn_DCA** 内存地址寄存器用于读取内存传输的实时地址。该地址工作范围被限制在 64k 对齐的地址空间。当地址在超过这个区域工作，地址会滚到 64k 对齐的地址空间的 0x0000。

被传输的源和目的地的地址可以通过设置 **DMA_CHn_SINC** 和 **DMA_CHn_DINC** 寄存器设置每次单位大小传输完成后自动增加，或不变。

当使能了地址自动增加，用户可以通过设置 **DMA_CHn_ADSEL** 寄存器选择“普通”或“SKIP3”两种增加模式。当选择“普通”，地址会自动加 1，选择“SKIP3”，则 LSB 的 2 位地址会从 0 增加到 1,1 到 2,2 到 0，而跳过地址 3。

表 12-4. DMA 传输数量/起始地址和单位大小设置注释

单位大小 (字节)	传输数量	内存起始地址		外设	EMB
	CHn_NUM	CHn_SSA	CHn_DSA	缓冲大小(字节)	总线宽度(位)
CHn_BSIZE	整数倍	地址对齐	地址对齐	(建议)	(建议)
1	1	字节	字节	1,2,3,4	8
2	2	半字	半字	2,4	8,16
4	4	字	字	4	8,16

注释-1：CHn：n=通道标号

● DMA 通道同步工作

每个 DMA 通道都可以通过设置 **DMA_CHn_SSYNC** 和 **DMA_CHn_DSYNC** 寄存器使能 DMA 源到 DMA 控制器和 DMA 控制器到 DMA 目的地的同步工作。当源工作时钟频率与 DMA 工作时钟频率相同，建议使能该位以增强 DMA 性能，目的地也是一样。

● DMA 通道循环模式

每个 DMA 通道都可以通过设置 **DMA_CHn_LOOP** 寄存器使能 DMA 数据传输循环模式。当使能时，上一次 DMA 数据传输完成时传输数据会自动重载为初始值，DMA 请求会被继续。该状态下，DMA 寄存器 **DMA_CHn_CNT** 中的数据计数器会自动通过 **DMA_CHn_NUM** 寄存器重载。禁用时，上一次 DMA 传输完成后，不会再有 DMA 请求被进行。

用于固件流控制，用户可通过设置 **DMA_CHn_LAST** 寄存器，在最后一次循环 DMA 传输完成之前的上一次 TCF 置起时在 DMA TCF 标志的终端服务程序中设置最后循环指令。然后 DMA 控制器会禁用循环功能并在最后数据传输完成并置起 TCF 标志时停止 DMA 传输，为了下一次的使用，用户需清除 **DMA_CHn_LAST** 寄存器位。

[注释]: 在 MG32F02A132/072 中, **DMA CHn LAST** 寄存器控制位被替换为 **SYS CHn LAST**。

DMA 循环模式可用于类似 ADC 通道扫描操作的顺序数据传输和循环数据传输操作。ADC 通道扫描操作下，DMA 可传输通道 0 到 15 的 ADC 转换数据到 SRAM。然后，下一循环的通道 0 的 ADC 数据会写到与上次通道 0 相同的写入地址上，其他通道也是这样。

12.8.4. DMA SRAM 使用

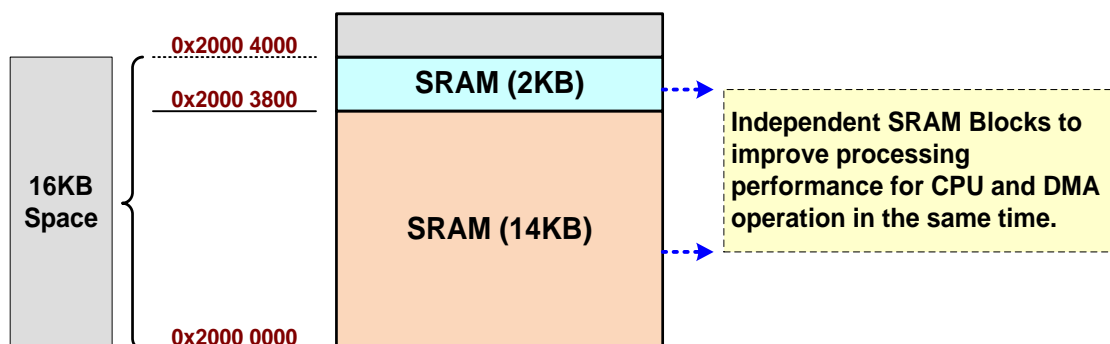
当 CPU 和 DMA 控制器同时访问了内部 SRAM 的同一个内存地址，CPU 会比 DMA 控制器拥有更高的访问优先级。当 CPU 获得 SRAM 内存访问控制，DMA 控制器会被停止，直到 CPU 释放 SRAM 内存控制。DMA 控制器可访问 SRAM 做数据传输，直到被下次发生 CPU 访问指令时中断。

为了增强 DMA 数据传输性能，SRAM 地址被设计成分开的两个块：上 2K 字节，下 14K 字节。强烈建议用户保留上 2K 字节用于 DMA 传输，下 14K 字节为软件服务。

[注释]: 分离两块 SRAM 块的设计不支持于 MG32F02A072/032。

当然，用户可以自行设置 SRAM 的任何内存空间作为软件使用或 DMA 传输，这没有任何硬件限制。

图 12-6. DMA SRAM 使用建议



[Upper 2KB]

1. Strongly suggest to use upper 2KB SRAM as DMA transfer 1st data buffer.
2. Use for firmware parameter and data buffer if does not request DMA.

[Lower 14KB]

1. Suggest to use lower 14KB SRAM for firmware parameter and data buffer.
2. If it is necessary that use partial of lower 14KB SRAM as DMA transfer 2nd data buffer. Additionally the DMA transfer performance is reduced by CPU operation for this SRAM.

12.9. DMA 传输

12.9.1. DMA 传输设置和序列

- 准备 DMA 传输

在启动 DMA 传输之前，用户需：

- 若通道工作类型为外设到内存、内存到外设或外设到外设时，需要配置工作的外设模块
- 通过寄存器 **CSC_DMA_EN** 设置使能 DMA 工作时钟
- 通过 **DMA_EN** 位启动 DMA 硬件
- 若芯片支持 **DMA_PRI_MDS** 寄存器，则选择通道优先级“轮询”或“优先级”

若 DMA 源或目的地为外设或 EMB 模块，外设模块必须设置完成。此外，DMA 控制器和工作通道必须设置“DMA 控制”节中相应的寄存器。用户还需设置模块的 DMA 使能位（类似 **XXX_DMA_RXEN**, **XXX_DMA_TXEN** 或 **XXX_DMA_EN**），为了启动 DMA 传输，相关的通道请求起始位 **DMA_CHn_REQ** 是必须的。**DMA_CHn_REQ** 位会在 DMA 数据传输完成后自动被硬件清除。（XXX = 外设模块名）

- DMA 通道选择

参照“[DMA 源和目的地](#)”节以获取更多关于 DMA 通道的信息。

- 通过 **DMA_CHn_EN** 位使能 DMA 发送通道
- 通过 **DMA_CHn_CIE** 和 **DMA_IEA** 位设置 DMA 发送完成中断使能
- 通过 **DMA_CHn_SRC** 和 **DMA_CHn_DET** 寄存器选择 DMA 通道源和目的地
- 通过 **DMA_CHn_BSIZE** 寄存器设置 DMA 单次发送字节量
- 通过 **DMA_CHn_NUM** 位设置 DMA 发送数据计数初始数
- 通过 **DMA_CHn_SSA** 和 **DMA_CHn_DSA** 位设置 DMA 源和/或目的地起始地址
- 若需要 DMA 通道循环模式则通过 **DMA_CHn_LOOP** 位使能

- DMA 传输启动

当 DMA 控制器和工作外设设置完成，用户需通过 **DMA_CHn_REQ** 位设置相关通道请求起始位，以启动 DMA 传输。**DMA_CHn_REQ** 位会在 DMA 数据传输完成后自动被硬件清除。

数据传输事件发生后，传输源和目的地设备会置起 RX/TX 请求信号到 DMA 控制器中。通过通道仲裁控制，DMA 控制器会根据优先级服务请求并置起承认信号到源/目的地设备。同时，会建立起 DMA 数据传输链接。请求源/目的地设备会在收到 DMA 控制器的承认信号后释放请求并开始传输数据。当所有传输数据传输完成，**DMA_CHn_CNT** 中的 DMA 传输数据计数器会等于 0 且 DMA 控制器会激活 TCF 标志。

12.9.2. DMA SLEEP 模式下工作

通常，DMA 传输是被设置和工作在 **ON** 模式中。若寄存器 **CSC_DMA_EN** 在芯片进入 **SLEEP** 模式之前被使能，DMA 控制器可支持 DMA 传输在 **SLEEP** 模式中继续保持工作类型为内存到内存的工作。

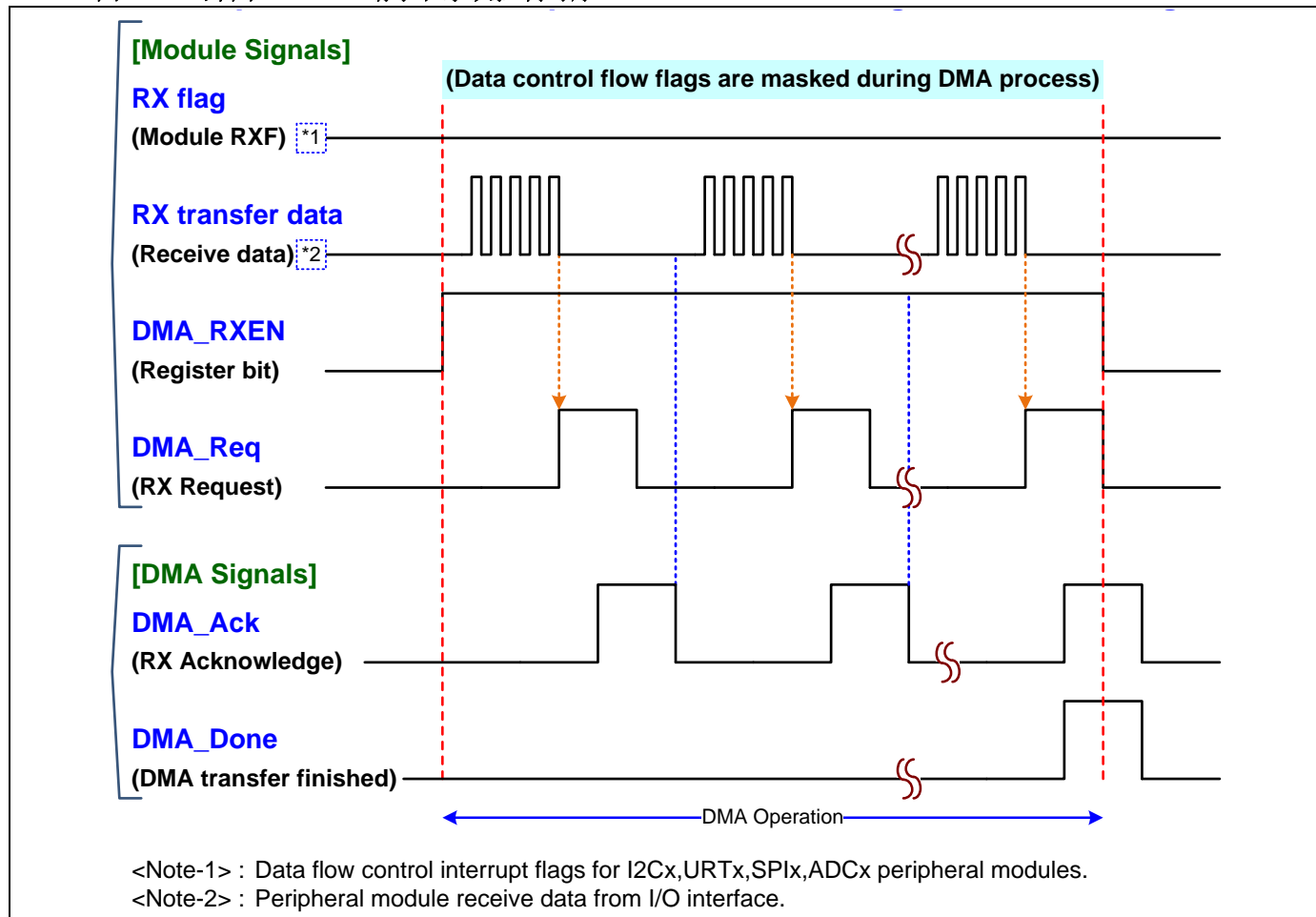
对于通道工作类型为外设到内存、内存到外设、外设到外设的时候，用户也可在芯片进入 **SLEEP** 模式之前，将工作模块的时钟通过 **CSC_SLP_xxx** 寄存器设置为在 **SLEEP** 模式下继续工作，那么 DMA 传输工作则可继续工作。当芯片进入 **SLEEP** 模式，DMA 传输会继续传输数据，直到 DMA 传输完成。

12.9.3. 外围 DMA RX 请求和承认

DMA 控制器和外围模块已完成 DMA 数据传输设置后，当外设模块收到来自外部设备通过通信接口得到的信号时，外围模块会置起 DMA 请求到 DMA 控制器，通过 DMA 控制器的操作，它会发送 DMA 承认信号到外围模块以提醒外围模块 DMA 请求已接收，DMA 请求会被外设模块释放，然后 DMA 数据传输会通过 DMA 控制器从外围模块开始向目的地传输。反复地进行上述操作直到传输完成，最后，DMA 控制器发送 DMA 完成信号到外围模块以终止 DMA 数据传输。

在 DMA 传输周期中，DMA 源设备的接收数据标志 **XXX_RXF** 会被硬件屏蔽。

图 12-7. 外围 DMA RX 请求和承认控制时序

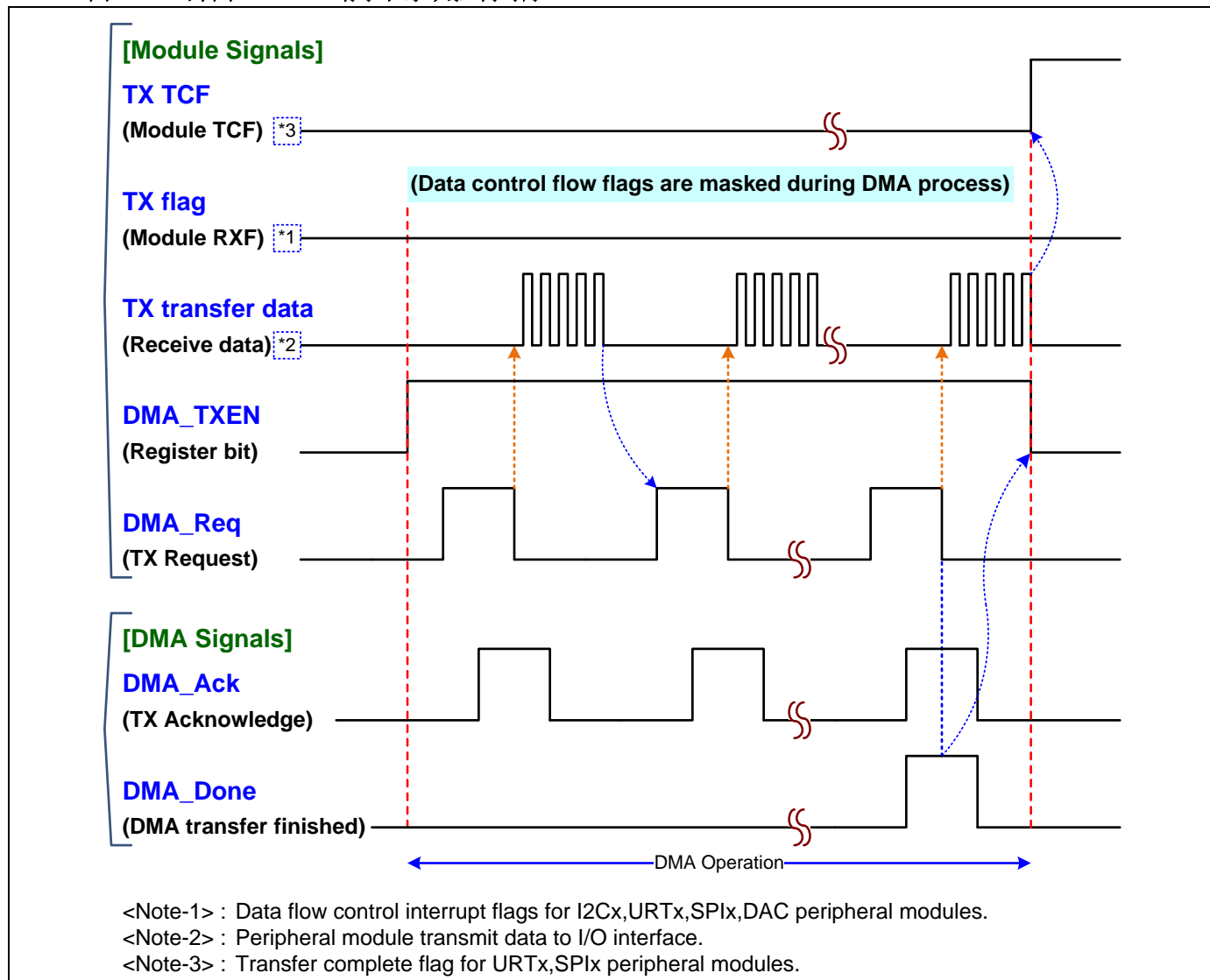


12.9.4. 外围 DMA TX 请求和承认

与“外围 DMA RX 请求和承认”相同，DMA 数据传输也用于外围模块的 TX 操作上。当 DMA 控制器最后一次完成传输数据到外围模块时，DMA 控制器发送 DMA 完成信号到外围模块以终止 DMA 数据传输，外设会通过通信接口发送最后的数据到外部设备。

在 DMA 传输周期中，DMA 目的地设备的发送数据标志 **XXX_TXF** 会被硬件屏蔽。通常，传输完成标志 **XXX_TCF** 会在 DMA 传输完成后被置起。

图 12-8. 外围 DMA TX 请求和承认控制时序



12.9.5. 外围 DMA 传输停止

对于 DMA 数据流控制或一些意外状态，固件可以直接通过 **DMA_CHn_HOLD** 寄存器为每个通道独立地强制停止 DMA 传输。

当 1 个通道停止功能被使能时，若相关 DMA 通道数据传输正在进行，DMA 控制器会传输完成当前正在传输的单位大小数据。然后 DMA 通道数据传输工作就会被停止，通道操作将会被改到下一个由仲裁控制决定优先级的通道。该 DMA 通道会保持停止 DMA 传输直到 **DMA CHn HOLD** 控制位被固件禁用。

12.9.6. 外围中断标志控制

DMA 操作过程中一共有 3 种外围模块中断标志控制。

- 操作过程中屏蔽标志
 - 一般用于数据流控制中断标志
- DMA 在标志置起后禁用
 - 一般用于错误检测中断标志
- 一般控制
 - 标志独立于 DMA 操作中

下面的表格展示了 DMA 功能的外围模块中断标志控制。

表 12-5. DMA 功能的外围模块中断标志控制 – MG32F02A132/072

操作	DMA 操作过程中屏蔽标志	DMA 在标志置起后禁用 (*1)	一般控制	
外围设备	(数据流标志)	(错误/检测标志)	(其他标志)	
I2Cx	I2Cx_EVENTF I2Cx_BUFF	I2Cx_ERRF(*1) I2Cx_STPSTRF(*1)	I2Cx_TMOUTF	I2Cx_RXF I2Cx_TXF I2Cx_RSTRF I2Cx_STOPF I2Cx_CNTF I2Cx_ERRCF I2Cx_SADRF I2Cx_TSCF I2Cx_ROVRF I2Cx_TOVRF I2Cx_NACKF I2Cx_ALOSF I2Cx_BERRF
URTx	URTx_TCF (*2) URTx_RXF URTx_TXF	URTx_ERRF URT0_BKF(*3) URT0_IDLF(*4) URT0_CTSF(*5)	URTx_UGF URTx_LSF	URT0_SADRF URT0_BRTF URT0_TMOF URT0_CALCF URT0_PEF URT0_FEF URT0_NCEF URT0_ROVRF URT0_TXEF URT0_RXTMOF URT0_IDTMOF URT0_BKTMOF URT0_CALTMOF
SPIx	SPIx_TCF (*2) SPIx_RXF SPIx_TXF	SPIx_MODF SPIx_WEF SPIx_ROVRF SPIx_TUDRF	SPIx_IDLF	
ADCx	ADC0_ESMPF ADC0_E1CNVF ADC0_ESCNVF(*2)	ADC0_OVRF ADC0_WDLF ADC0_WDIF ADC0_WDHF	ADC0_SUMOF ADC0_SUMCF ADC0_SUMOVRF	
DAC	DAC_RDY0F	DAC_UDR0F		
TM36	TM36_CF0A/0B TM36_CF1A/1B TM36_CF2A/2B TM36_CF3A/3B	TM36_TOF TM36_TUF	TM36_EXF TM36_TOF2 TM36_TUF2 TM36_DIRCF	TM36_IDXF TM36_QPEF TM36_BKF
EMB		EMB_WPEF(*1) EMB_BWEF(*1)		

注释-1：当标志被置起时，若相关中断使能标志未被使能，它将不会强制禁用外围的 DMA，特别的是，即使相关中断使能标志未被使能，I2Cx_ERRF 或 I2Cx_STPSTRF 仍能使 DMA 被禁用。

注释-2：该标志会在 DMA TX 完成后被置起。

注释-3：当标志被置起，若相关中断使能标志被使能，它将会强制禁用外围的 URTx DMA RX。同时，通过设置 URTx_DDTX_EN 寄存器，会决定 URTx DMA TX 是否被禁用。

注释-4：当标志被置起，若相关中断使能标志被使能，它将会强制禁用外围的 URTx DMA RX，但是 URTx DMA TX 不会。

注释-5：当标志被置起，若相关中断使能标志被使能，它将会强制禁用外围的 URTx DMA TX，但是 URTx DMA RX 不会。通常，用户设置 CTS 硬件自动控制模式(URT_x_CTS_EN=1)将不会使能 URT_x_CTS_IE，URT_x 将会停止传输，并在检测到 CTS 启动时不会禁用 URT_x DMA TX。

表 12-6. DMA 功能的外围模块中断标志控制 – MG32F02A032

操作	DMA 操作过程中 屏蔽标志	DMA 在标志置起 后禁用 (*1)	一般控制	
外围设备	(数据流标志)	(错误/检测标志)	(其他标志)	
I2Cx	I2Cx_EVENTF I2Cx_BUFF	I2Cx_ERRF(*1) I2Cx_STPSTRF(*1)	I2Cx_TMOUTF I2Cx_WUPF	I2Cx_RXF I2Cx_TXF I2Cx_RSTRF I2Cx_STOPF I2Cx_CNTF I2Cx_ERRCF I2Cx_SADRF I2Cx_TSCF I2Cx_ROVRF I2Cx_TOVRF I2Cx_NACKF I2Cx_ALOSF I2Cx_BERRF
URT _x	URT _x _TCF (*2) URT _x _RXF URT _x _TXF	URT _x _ERRF URT0_BKF(*3) URT0_IDLF(*4) URT0_CTSF(*5)	URT _x _UGF URT _x _LSF	URT0_SADRF URT0_BRTF URT0_TMOF URT0_CALCF URT0_PEF URT0_FEF URT0_NCEF URT0_ROVRF URT0_TXEF URT0_RXTMOF URT0_IDTMOF URT0_BKTMOF URT0_CALTMOF
SPI _x	SPI _x _TCF (*2) SPI _x _RXF SPI _x _TXF	SPI _x _MODF SPI _x _WEF SPI _x _ROVRF SPI _x _TUDRF	SPI _x _IDLF	
ADC _x	ADC0_ESMPF ADC0_E1CNVF(*6) ADC0_ESCNVF(*2)	ADC0_OVRF ADC0_WDLF ADC0_WDIF ADC0_WDHF	ADC0_SUMOF ADC0_SUMCF ADC0_SUMOVF	
TM36	TM36_CF0A/0B TM36_CF1A/1B TM36_CF2A/2B TM36_CF3A/3B	TM36_TOF TM36_TUF	TM36_EXF TM36_TOF2 TM36_TUF2 TM36_DIRCF	TM36_IDXF TM36_QPEF TM36_BKF

注释-1：当标志被置起时，若相关中断使能标志未被使能，它将不会强制禁用外围的 DMA，特别的是，即使相关中断使能标志未被使能，I2Cx_ERRF 或 I2Cx_STPSTRF 仍能使 DMA 被禁用。

注释-2：该标志会在 DMA TX 完成后被置起。

注释-3：当标志被置起，若相关中断使能标志被使能，它将会强制禁用外围的 URT_x DMA RX。同时，通过设置 URT_x_DDTX_EN 寄存器，会决定 URT_x DMA TX 是否被禁用。

注释-4：当标志被置起，若相关中断使能标志被使能，它将会强制禁用外围的 URT_x DMA RX，但是 URT_x DMA TX 不会。

注释-5：当标志被置起，若相关中断使能标志被使能，它将会强制禁用外围的 URT_x DMA TX，但是 URT_x DMA RX 不会。通常，用户设置 CTS 硬件自动控制模式(URT_x_CTS_EN=1)将不会使能 URT_x_CTS_IE，URT_x 将会停止传输，并在检测到 CTS 启动时不会禁用 URT_x DMA TX。

注释-6：当 ADC0_DMA_MDS 被禁用，ADC0_E1CNVF 标志会在 DMA 工作时被屏蔽，当 ADC0_DMA_MDS 是“Keep”，ADC0_E1CNVF 标志则会在 DMA 工作时正常，而不被屏蔽

12.10. DMA 外部请求触发输入

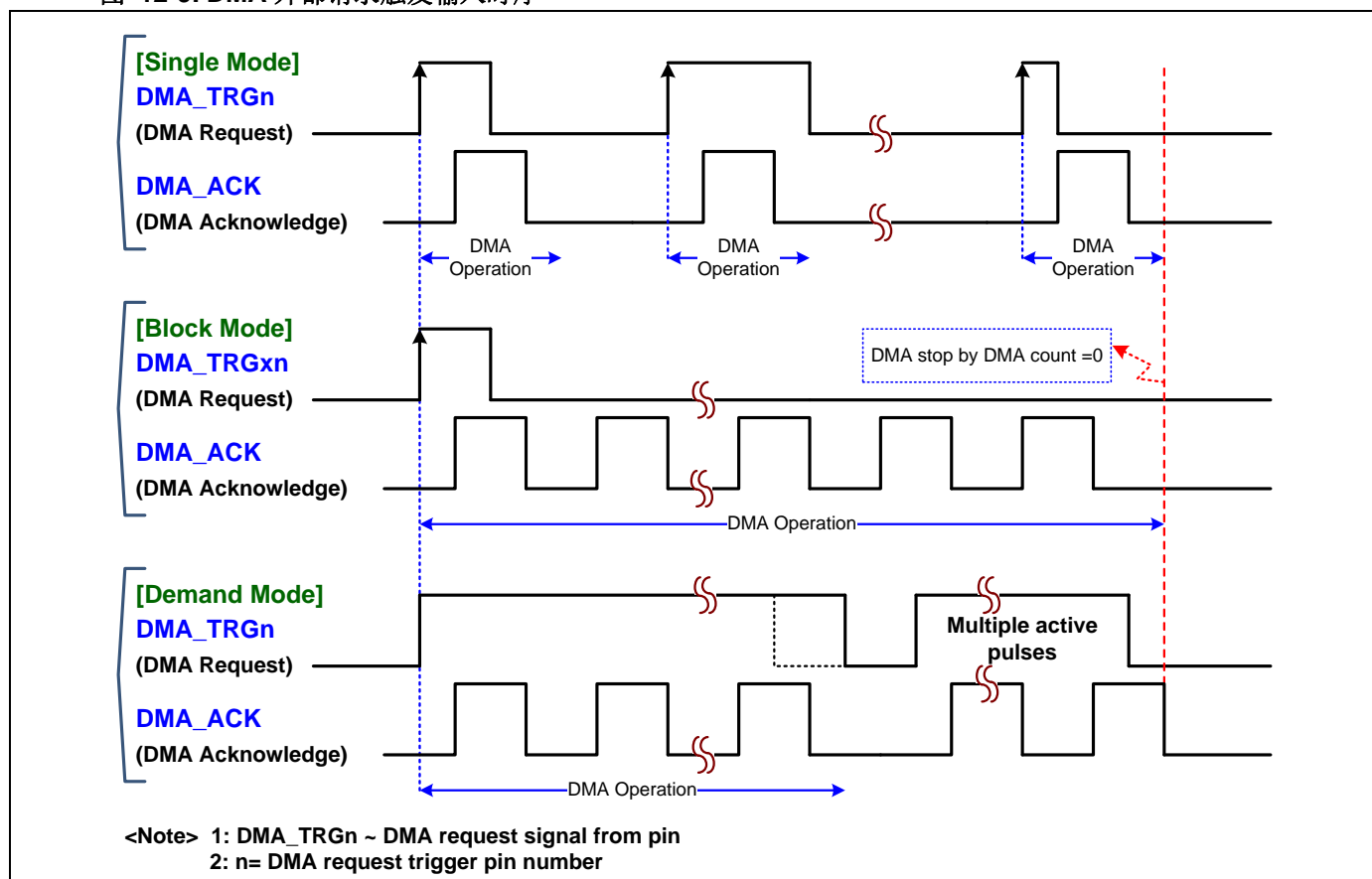
DMA 支持 2 个 **DMA_TRG0** 和 **DMA_TRG1** 外部引脚作为 DMA 请求触发信号做 DMA 数据传输。一共有 3 种控制模式用于外部请求触发输入。用户可通过设置 **DMA_CH0_XMDS** 寄存器为每个通道独立地使能和设置 single/block/demand 模式。

注释: Demand 模式不支持于外设到外设 DMA 传输功能。

当使能了 DMA 外部触发模式，每个 DMA 通道可通过设置每个独立通道的 **DMA_CHn_XPIN** 寄存器，设置每个通道的触发信号输入来自 **DMA_TRG0** 或 **DMA_TRG1** 引脚。

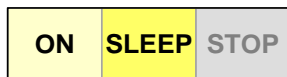
下面的图表展示了 single/block/demand 模式的控制时序。

图 12-9. DMA 外部请求触发输入时序



13. EMB (外置内存总线)

13.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片内置 1 个外置内存总线（EMB）控制器用来访问 SRAM、NOR/NAND 闪存、8080 接口 LCD 外部设备。EMB 控制器支持地址总线 and 数据总线多路复用模式，此外，它提供两个地址锁存使能信号用于支持地址和数据周期的多重控制。

13.2. 特性

- 支持 SRAM, NAND/NOR 闪存, LCD 接口
- 支持同步或异步时序模式控制
- 支持 16 位数据宽度
- 支持多种类型的地址和数据多路复用模式
- 提供可选的 16/24/30 位地址模式
 - 16 位数据宽度的内存空间 2G/32M/128K 字节
- 支持以 16 位数据宽度模式进行字节写操作
- 可配置时钟周期用于地址锁存时钟和数据访问时钟
- 可用 DMA 进行数据发送和接收
 - 对于 DMA 来说，外部内存空间与内部内存一样
- 允许在外部 SRAM 运行 CPU 代码

13.3. 配置

13.3.1. 芯片配置

下面的表格展示了芯片的 EMB 通道配置。

表 13-1. EMB 配置

芯片	封装	EMB 模块		
		16bit-MA/16bit-MD	16bit-MA/16bit-MAD	16bit-MAD
MG32F02A132AD80	LQFP80	V	V	V
MG32F02A132AD64	LQFP64	V	V	V
MG32F02A072AD64	LQFP64	V	V	V
MG32F02A072AD48	LQFP48			
MG32F02A032	All			

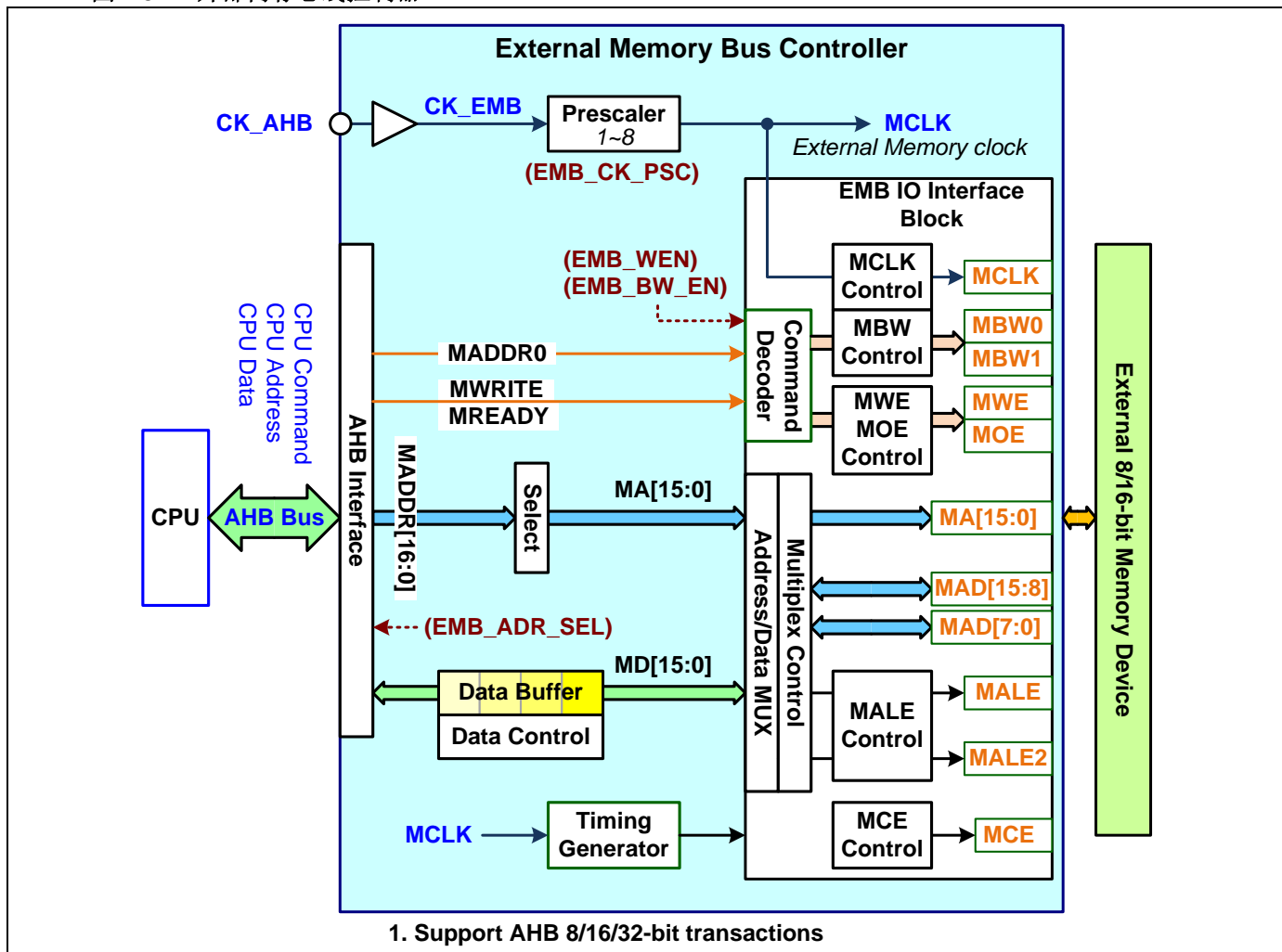
注释
 V：包含
 MA：内存地址，MD：内存数据，
 MAD：内存地址和数据复用

13.4. 控制块

EMB 控制器包含 AHB 逻辑接口和外部内存总线 IO 接口块、16 位数据缓冲和 1 个总线时钟发生器。EMB 的 IO 接口块可控制地址、数据和指令信号用于外部设备。总线时序发生器输入 MCLK 时钟，并输出用于 EMB IO 接口块的可设置的时序。

下面的图表展示了 EMB 控制块。

图 13-1. 外部内存总线控制器



13.5. IO 线

13.5.1. IO 信号

- **MCLK**

外部内存总线的时钟信号，用于外部同步 SRAM、NOR Flash 或其他内存设备。

- **MCE**

芯片使能或片选信号，并用于作为外部内存总线的输出。

- **MOE**

输出使能(OE)或读选通(RD)信号，并用于作为外部内存总线的输出。

- **MWE**

写使能(WE)或写选通(WR)信号，并用于作为外部内存总线的输出。

- **MALE**

地址锁存使能(ALE)或数据/指令选择(DC)信号，并用于作为外部内存总线的输出。

- **MALE2**

2nd 地址锁存使能(ALE2)或指令锁存(CLE)信号，并用于作为外部内存总线的输出。

- **MBW0**

字节写使能 0 或地址信号的最低有效位，并用于作为外部内存总线的输出。

- **MBW1**

字节写使能 1 信号，并用于作为外部内存总线的输出。

- **MA[0..15]**

内存地址输出信号，并用于作为外部内存总线的输出。

- **MAD[0..15]**

数据双向信号用于外部内存总线。此外，它们还可以用作地址和数据的复用信号，可以使用 MALE/MALE2 控制信号来区分地址和数据。

13.5.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。参照用户手册 GPIO 章中“[IO 模式](#)”节以获取更多关于 IO 模式设置的信息。

每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“[功能复用选择](#)”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“[引脚功能复用表](#)”以获取更多信息。

13.6. 使能和时钟

13.6.1. EMB 全局使能

EMB 模块的所有功能全局使能位是 **EMB_EN**。当该位被禁用时，所有的 EMB 功能将无法工作。

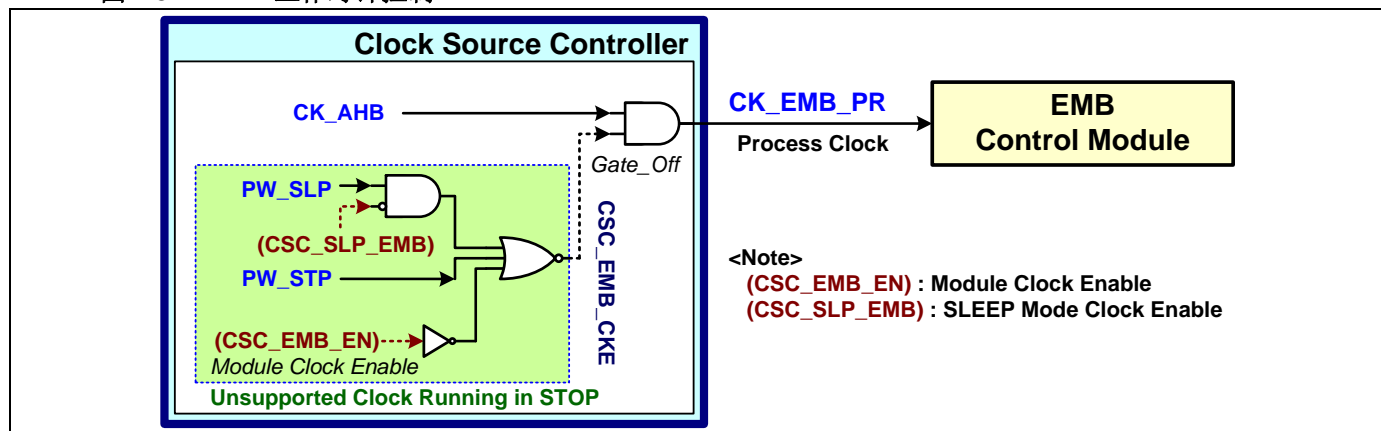
13.6.2. EMB 时钟控制

- **模块工作时钟**

该模块工作时钟 **CK_EMB_PR** 是用于 AHB 总线和模块的接口逻辑控制。该时钟来源于 CSC (时钟源控制器) 模块。该时钟可通过 **CSC_EMB_EN** 寄存器使能。

ON 下，工作时钟只会在 **CSC_EMB_EN** 寄存器被使能时运行；用户可以在进入 **SLEEP** 模式之前通过设置 **CSC_SLP_EMB** 寄存器规划在 **SLEEP** 模式下是否让 EMB 继续运行；**STOP** 模式下，工作时钟会被禁用，参照系统时钟章以获取更多信息。

图 13-2. EMB 工作时钟控制



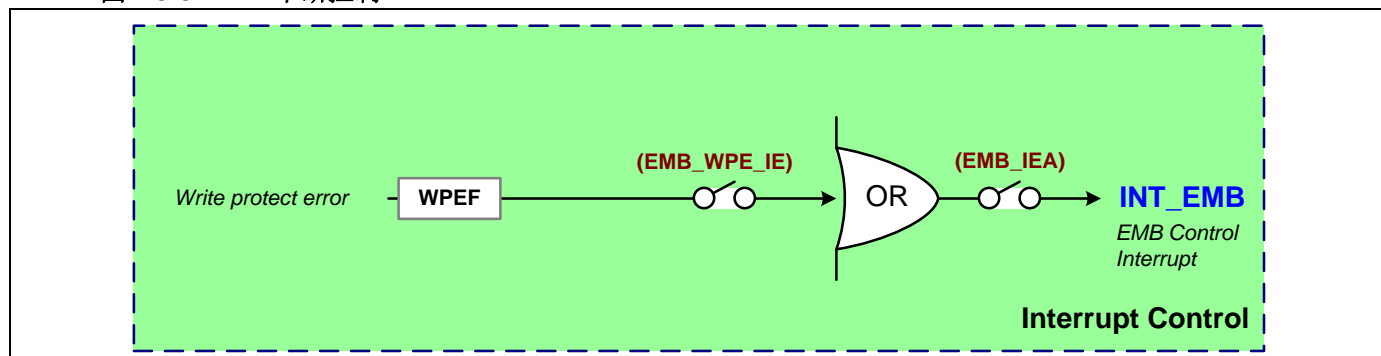
- **模块内部时钟**

EMB 内部时钟来源于 AHB 时钟 **CK_AHB**。该时钟频率可通过时钟分频器进行 1~8 分频作为外部内存总线输出时钟 **MCLK**。**MCLK** 时钟也可作为 EMB 时序发生器的时基时钟。用户可通过 **EMB_CK_PSC** 寄存器设置时钟分频。

13.7. 中断和事件

EMB 模块中有 1 种信号 **INT_EMB**。**INT_EMB** 发送到外部中断控制器（EXIC）作为中断事件。

图 13-3. EMB 中断控制



13.7.1. EMB 中断控制和状态

中断标识是用于中断服务程序（ISR）流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **EMB_IEA** 用于使能和禁用该模块的所有中断源。

占用标志 **EMB_BUSYF** 表明数据传输繁忙状态。参照相关状态位寄存器描述以获取更多信息。

13.7.2. EMB 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

- **WPEF**

EMB 总线写保护错误检测标志是 (**EMB_WPEF**)。相关中断使能寄存器位是 **EMB_WPE_IE**。当 EMB 写操作被禁用(**EMB_WEN=0**)且进行了 EMB 写操作时，会置起该标志。

13.8. EMB IO 控制

EMB 提供了 16 位数据信号—**MAD[15:0]**，最大 16 线地址信号—**MA[15:0]**，片选信号—**MCE**，数据输出使能信号—**MOE**，数据写使能信号—**MWE**，2 个地址锁存使能信号—**MALE**/**MALE2**，字节写使能信号—**MBW0**/**MBW1**和输出时钟信号—**MCLK**。**MAD[15:0]**信号可以输出地址或数据信号，用于地址和数据复用模式。

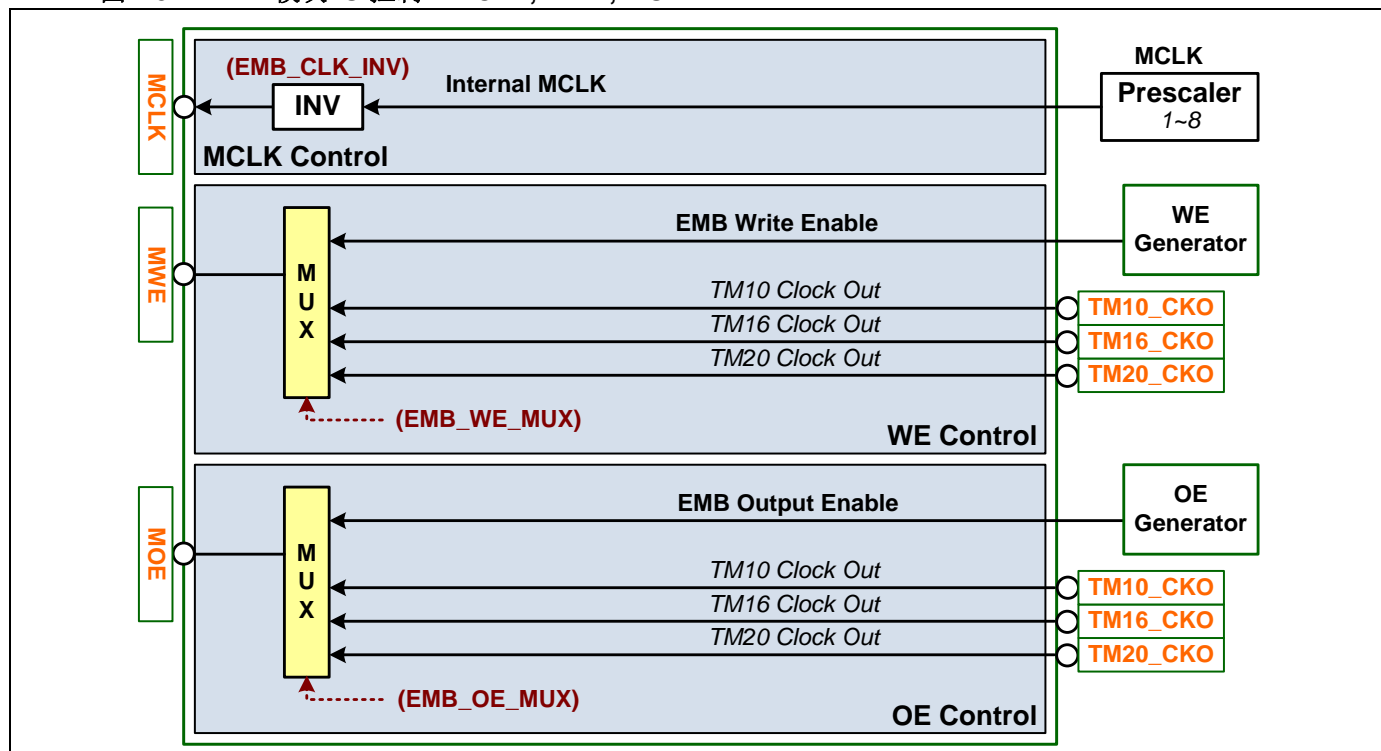
EMB 为了让用户能更简单的设计，支持 IO 信号反相，切换和源选择。EMB 接口 IO 可强制用软件控制模式进行**MCE**，**MOE**，**MWE**，**MBW0**，**MBW1**，**MALE**，**MALE2**的输出。

13.8.1. EMB 时钟和指令信号控制

MCLK时钟可通过设置**EMB_CLK_INV**寄存器进行反相。在应用中，用户可通过设置**EMB_WE_MUX**寄存器选择**MWE**输出信号源来自 WE 发生器或定时器 TM10/16/20 时钟输出。同时，用户还可以通过设置**EMB_OE_MUX**寄存器选择**MOE**输出信号源来自 WE 发生器或定时器 TM10/16/20 时钟输出。所有的**TMx_CKO**信号都来自于 TMx 定时器模块。

下面的表格展示了 EMB 模块**MCLK**，**MWE**和**MOE**的 IO 信号控制。

图 13-4. EMB 模块 IO 控制 - MCLK, MWE, MOE

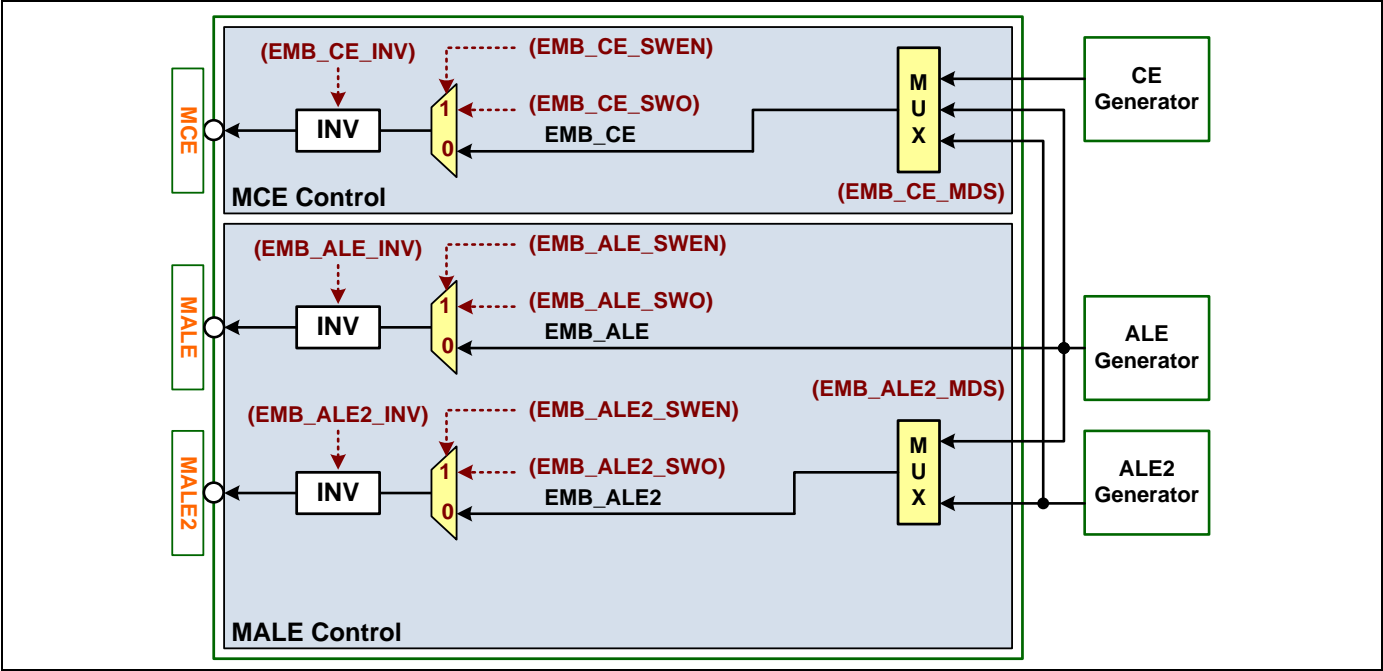


MCE，**MALE**和**MALE2**输出可通过设置**EMB_CE_INV**，**EMB_CLK_ALE**和**EMB_ALE2_INV**寄存器进行反相。**EMB_CE_SWEN**寄存器用于使能软件控制**EMB_MCE**输出。当被使能时，用户可通过设置**EMB_CE_SWO**寄存器直接控制输出。对于**MALE**和**MALE2**输出，用户可通过设置**EMB_ALE_SWEN**和**EMB_ALE2_SWEN**寄存器使能软件控制。当被使能时，用户可通过设置**EMB_ALE_SWO**和**EMB_ALE2_SWO**寄存器直接控制输出。

对于 PCB 线路或外部设备要求，用户可通过设置**EMB_CE_MDS**寄存器，选择**MCE**输出信号源来自 CE 发生器、ALE 发生器或 ALE2 发生器。用户也可以选择通过设置**EMB_ALE2_MDS**寄存器，选择**MALE2**输出信号源来自 ALE 发生器或 ALE2 发生器。

下面的图表展示了 EMB 模块的**MCE**，**MALE**和**MALE2** IO 信号控制。

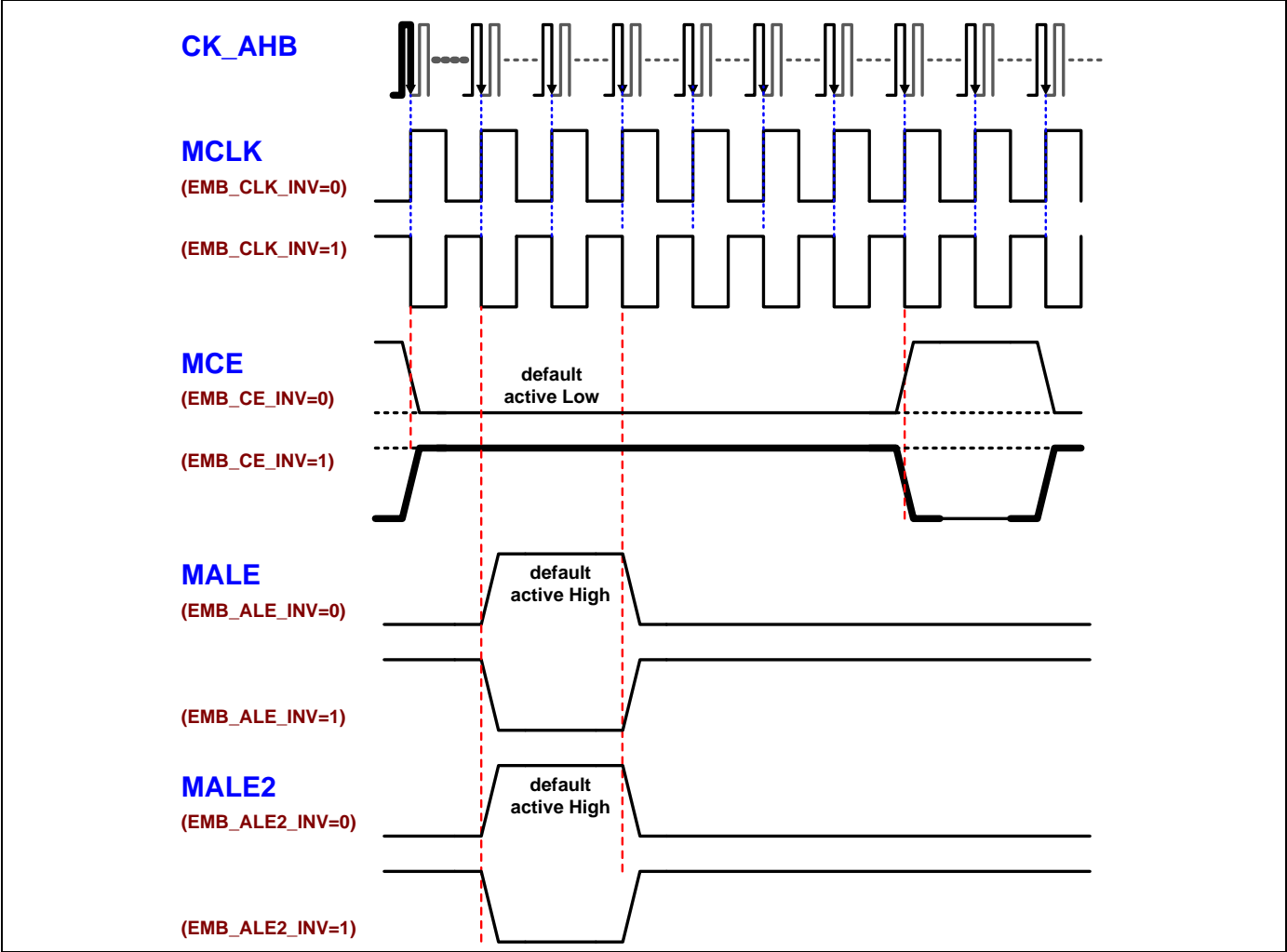
图 13-5. EMB 模块 IO 控制 - MCE, MALE, MALE2



EMB 支持通过设置 **EMB_CLK_INV**, **EMB_CE_INV**, **EMB_ALE_INV** 和 **EMB_ALE2_INV** 寄存器使 **MCLK**, **MCE**, **MALE** 和 **MALE2** 反相。

下面的图表展示了可设置极性的 EMB 时钟和控制信号。

图 13-6. EMB 时钟和控制信号极性



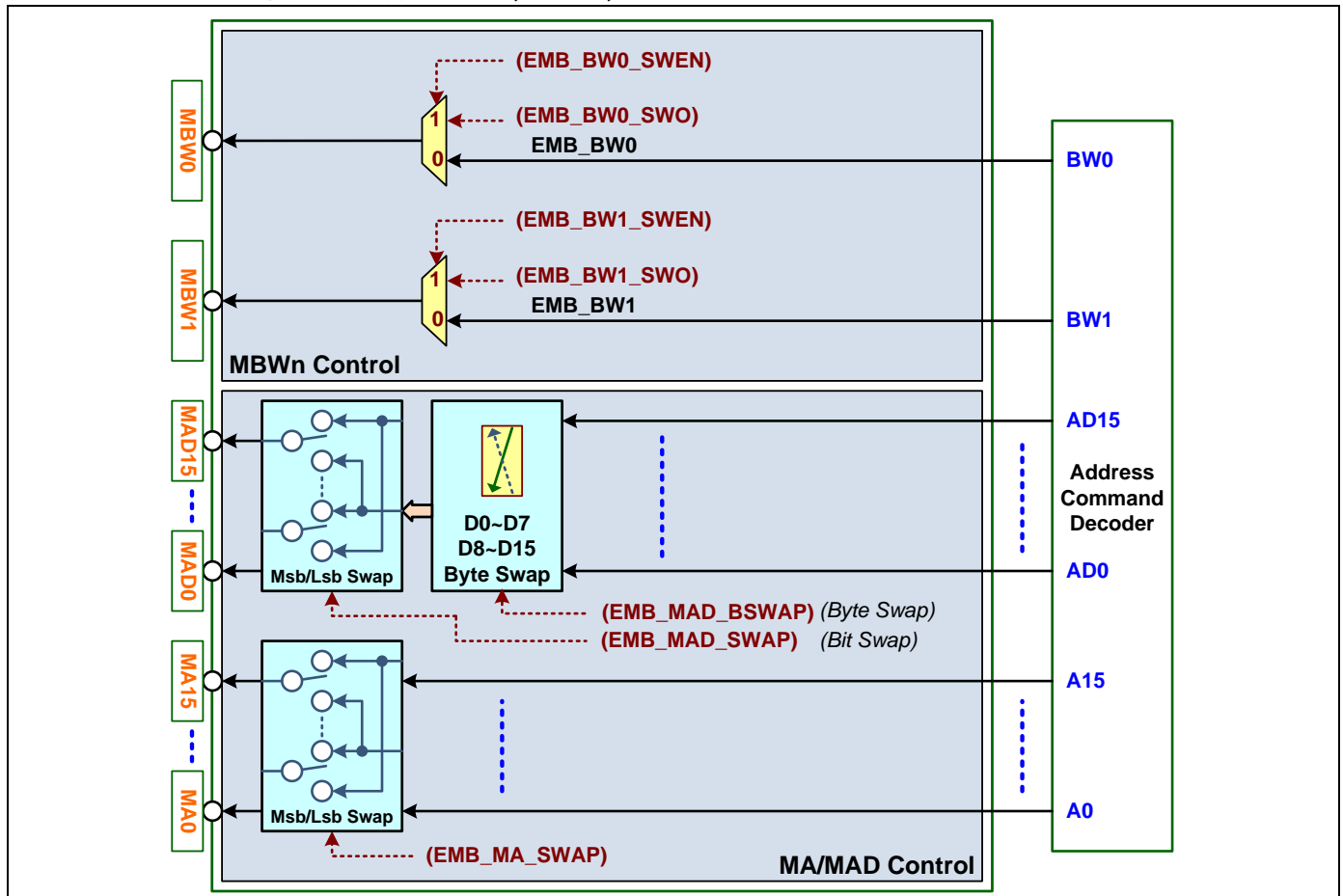
13.8.2. EMB 地址和数据信号控制

用于 **MBW0** 和 **MBW1** 输出，用户可通过设置 **EMB_BW0_SWEN** 和 **EMB_BW1_SWEN** 寄存器使能软件控制。当被使能时，用户可通过设置 **EMB_BW0_SWO** 和 **EMB_BW1_SWO** 寄存器直接控制输出。

EMB_MA_SWP 寄存器是用于使能 **MA[15:0]** 信号的 Msb/Lsb 信号切换的；**EMB_MAD_SWP** 寄存器是用于使能 **MAD[15:0]** 信号的 Msb/Lsb 信号切换的；**EMB_MAD_BSWP** 寄存器是用于使能 **MAD[15:8]** 和 **MAD[7:0]** 信号的 Msb/Lsb 信号切换的。

下面的图表展示了 EMB 模块 **MBW0**, **MBW1** 和 **MAD** IO 信号控制。

图 13-7. EMB 模块 IO 控制 - MBW0, MBW1, MAD



13.8.3. 用于外部设备的信号映射建议

下面的表格展示了 EMB 接口和外部设备的信号映射建议。

表 13-2. EMB 接口和设备引脚映射

设备	SRAM	NOR	NAND	LCD RAM
信号	16 位数据	16 位数据	16 位数据	16 位数据
MCLK	CLK	CLK (*3)	-	-
MA[15:0]	A[15:0]	A[15:0]	-	-
MAD[15:0]	DQ[15:0]	DQ[15:0]	DQ[15:0]	DB[15:0]
MCE	CE#	CE#	CE#	/CS (*1)
MWE	WE#	WE#	WE#	/WR
MOE	OE#	OE#	RE#	/RD

MALE	ADSP#	ADV# (*3)	ALE (*1)	RS (*1)
MALE2	-	RST# (*1)	CLE (*1)	RESET (*1)
MBW0	BW0	BYTE#	-	-
MBW1	BW1	WP# (*1)	WP# (*1)	-
GPIO (*2)	-	RY/BY# (*4)	RY/BY#	-

*1 :输出控制直接被软件模式寄存器设置

*2 : 任何未被使用的 GPIO 引脚（输入）

*3 : 只可用于同步 NOR

*4 : 只可用于异步 NOR

13.9. EMB 内存控制

13.9.1. EMB 内存空间

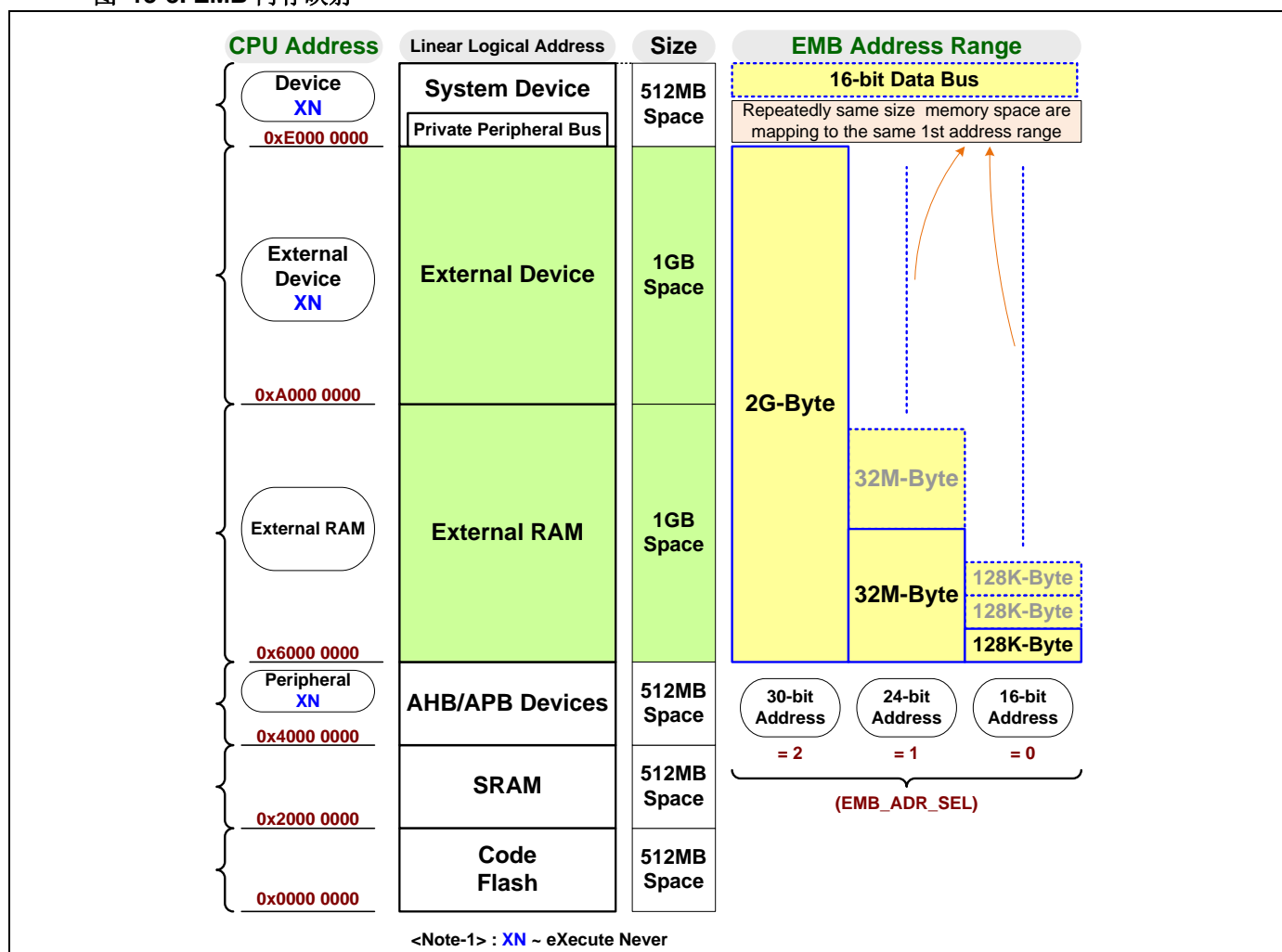
EMB 最大内存空间为 2G 字节。EMB 内存地址位于 CPU 基地址 0x6000 0000。

EMB 支持 16 位总线 and 可选择 16/24/30 位地址模式。用户可通过 **EMB_ADR_SEL** 寄存器设置地址范围。

外部设备的最大存储空间大小是 16 位数据宽度的 2G/32M/128K 字节或 8 位数据宽度的 1G/16M/64K 字节。除了配置 16 位数据宽度和 30 位数据地址模式而耗费所有 2G 字节空间外，其他的都用 EMB 2G 字节空间中的 1st EMB 内存空间的相同内容填充到伪存储器空间。

下面的图表展示了外部设备的 CPU 内存映射。

图 13-8. EMB 内存映射



13.9.2. AHB 到外部内存传输

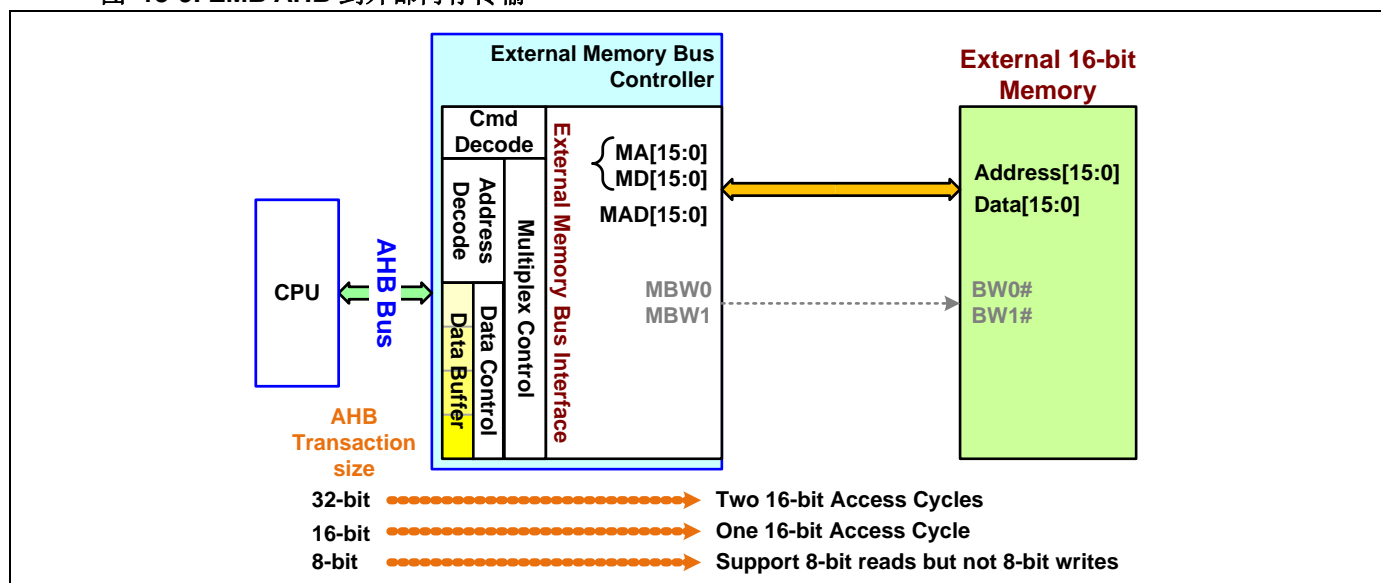
AHB 总线支持 8/16/32 位数据宽度，EMB 支持 16 位数据宽度。当 CPU 执行指令且访问内存空间为 EMB 空间时，EMB 会根据用户设置自动操作 CPU 与外部内存/设备的数据传输。若 EMB 被有效地配置初始化，用户可把外部内存或设备当做内部内存。

在 EMB 数据传输前，用户需初始化 EMB 数据总线宽度、EMB 地址模式、EMB 地址和数据复用模式、EMB 写保护和 EMB 时序设置。

EMB 写访问错误是必须的，用于避免 CPU 执行 8 位数据写操作。

下面的表格展示了 AHB 总线和外部设备之间的数据传输。

图 13-9. EMB AHB 到外部内存传输



下面的表格展示了 AHB 总线和外部设备的数据传输兼容性。EMB 支持访问外部内存，如 SRAM, 16 位数据总线的 NOR/NAND-flash。SRAM 和 NOR-flash 支持同步和异步时序，但是 NAND-flash 只支持异步时序。

表 13-3. EMB AHB 到外部内存传输支持

设备	模式	内存数据大小	AHB 数据大小	访问读写 RW	运行访问	访问周期	注释
SRAM	同步/异步	16	32	R/W	Yes	2	
			16	R/W	Yes	1	
			8	R	Yes	1	
				W	No	1	不支持字节写功能
NOR	同步/异步	16	32	R/W	Yes	2	
			16	R/W	Yes	1	
			8	R	Yes	1	
				W	No	1	不支持 BYTE#功能
NAND	异步	16	32	R/W	Yes	2	
			16	R/W	Yes	1	
			8	R	Yes	1	
				W	No	1	不支持字节写功能

13.9.3. EMB 写保护

EMB 为外部设备或内存支持写保护功能。用户可通过设置 **EMB_WEN** 寄存器使能或禁用写操作。当 EMB 写操作被禁用(**EMB_WEN**=0)，且执行了 EMB 读操作，写保护错误检测标志 WREF (**EMB_WPEF**)会被置起。

13.9.4. EMB 时序控制

EMB 内建时序发生器用于产生 EMB 输出信号。它提供了多种时序状态，并为外部设备的灵活设计提供了可设置时序周期。

用户可通过设置 **EMB_ALES**, **EMB_ALEW**, **EMB_ALEH**, **EMB_ACCS**, **EMB_ACCW**, **EMB_ACCH** 寄存器设置 MCLK 时钟的时钟周期 t_{ALES} , t_{ALEW} , t_{ALEH} , t_{ACCS} , t_{ACCW} , t_{ACCH} 。时钟周期 t_{ACCH} 可被 **EMB_ACCS** 寄存器设置，用于写操作，且会被硬件或读操作强制变成 0。时钟周期 t_{IDLE} 会被 2 个 16 位 EMB 数据访问组成的 32 位数据 CPU 指令强制变成 0。由于 16 位或 32 位的数据 CPU 指令，在最后一次 16 位 EMB 数据访问之后，它被强制为 1 个 MCLK 时钟时间。

EMB 可支持同步和异步传输的外部设备。用户需通过 **EMB_SYNC_EN** 寄存器为同步类型设备设置同步时序。

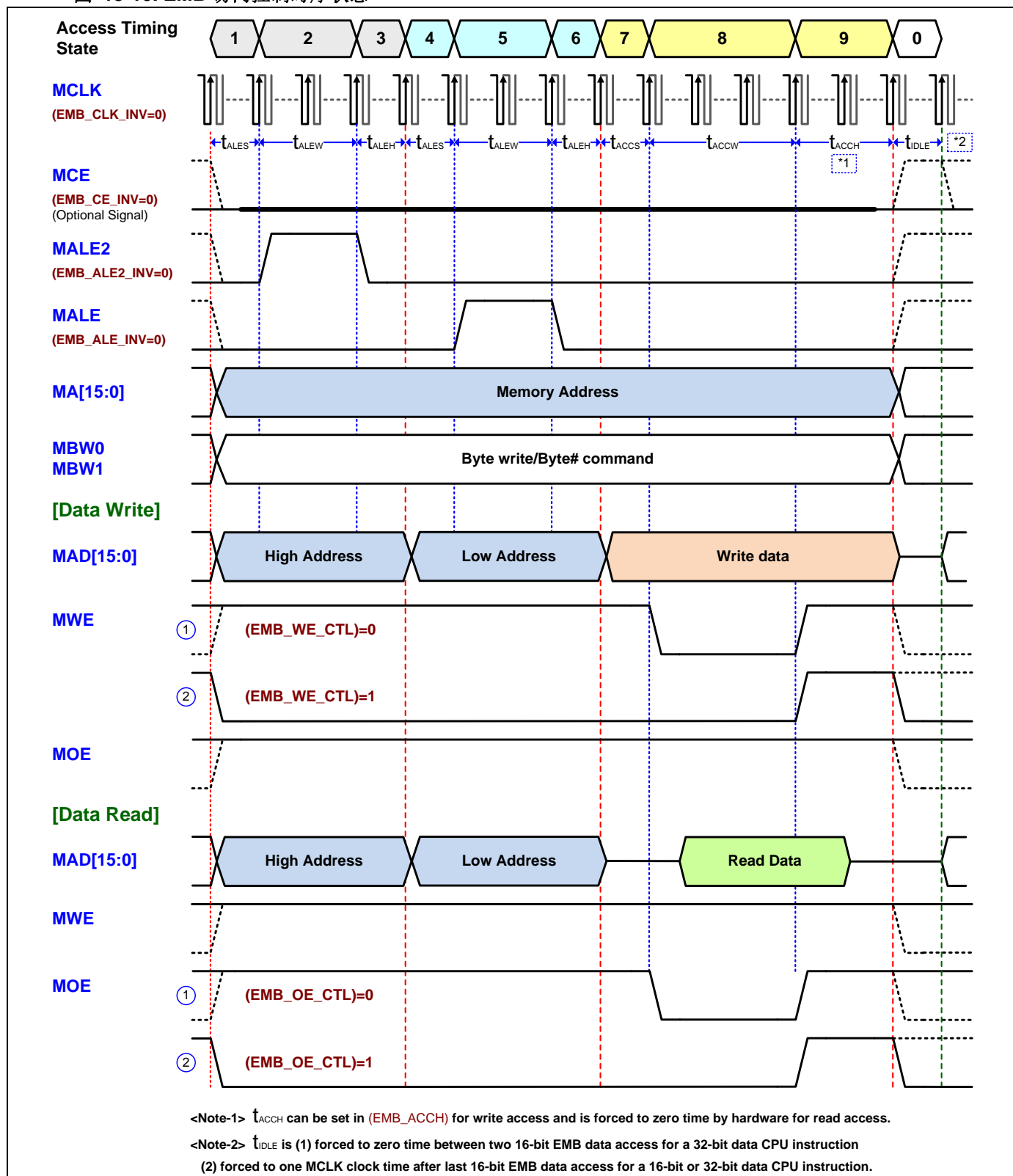
MWE 和 **MOE** 输出时序可设置“切换”和“低”2 种控制模式。当设置为“切换”模式时，信号将会在数据访问周期（下图中状态 8）被激活；当设置为“低”模式时，信号将会在开始时（下图中状态 1）被激活。

下面的图表展示了 EMB 访问控制时序状态。状态 0 是空闲状态；状态 1、4、7 是设置时序状态；状态 2、5 是脉冲时序状态；状态 8 是访问数据时序状态；状态 3、6、9 是停止时序状态。

用户可通过 **EMB_BUS_MDS** 和 **EMB_ADR_TWO** 寄存器设置多路复用地址数量。参照“[EMB 地址和数据接口模式](#)”节以获取更多信息。当选择 1 个多路复用地址，状态 1、2、3 会消失，当选择不使用多路地址复用，状态 1、2、3、4、5、6 都将消失。

下面的图表展示了 EMB 访问控制时序状态。

图 13-10. EMB 访问控制时序状态



13.10. EMB 地址和数据接口模式

EMB 支持 3 种地址和数据接口模式。他们包含 2 种地址和数据复用模式。这些模式包括了“16bit-MA + 16bit-MD” “16bit-MA + 16bit-MAD” 和 “16bit-MAD with 2 Adr”。(Adr :地址相位)

用户需通过 **EMB_ADR_TWO** 和 **EMB_BUS_MDS** 寄存器选择一种模式，**EMB_BUS_MDS** 寄存器是用于选择 EMB 总线地址和数据为“复用”还是“分离”的。

EMB_ADR_TWO 寄存器是用于使能 2 种多路地址复用时序模式的。被禁用时，若 **EMB_BUS_MDS** 被设置为“分离”，则没有多路复用地址；**EMB_BUS_MDS** 被设置为“复用”时，则有 1 个多路复用地址；被使能时，则有 2 个多路复用地址，且 **EMB_BUS_MDS** 必须设置为“多路复用”。

EMB_ADR_SEL 寄存器用于选择 16、24、30 位地址范围。

下面的表格展示了 EMB 地址/数据接口模式寄存器设定。

表 13-4. EMB 地址/数据接口模式设置

接口模式	EMB 设置寄存器		
	ADR_SEL	ADR_TWO	BUS_MDS
16bit-MA + 16bit-MD	0 16-bit Adr	0 0 phase Adr	1 separated
16bit-MA(HAdr) + 16bit-MAD(LAdr+Data)	0,1,2 16,24,30-bit Adr	0 1 phase Adr	0 multiplexed
16bit-MAD with 2 Adr (HAdr+LAdr+Data)	0,1,2 24,30-bit Adr	1 2 phase Adr	0 multiplexed

MA：内存地址，MD：内存数据，MAD：内存地址和数据复用
HAdr：高位地址，LAdr：低位地址

13.10.1. EMB 16 位 MA 和 16 位 MD

该模式被设置为分离的 16 位地址和 16 位数据。地址模式只有 16 位地址空间。没有时序状态 0、1、2、3、4、5、6。

图 13-11. EMB 地址/数据接口 – 16bit MA + 16bit MD

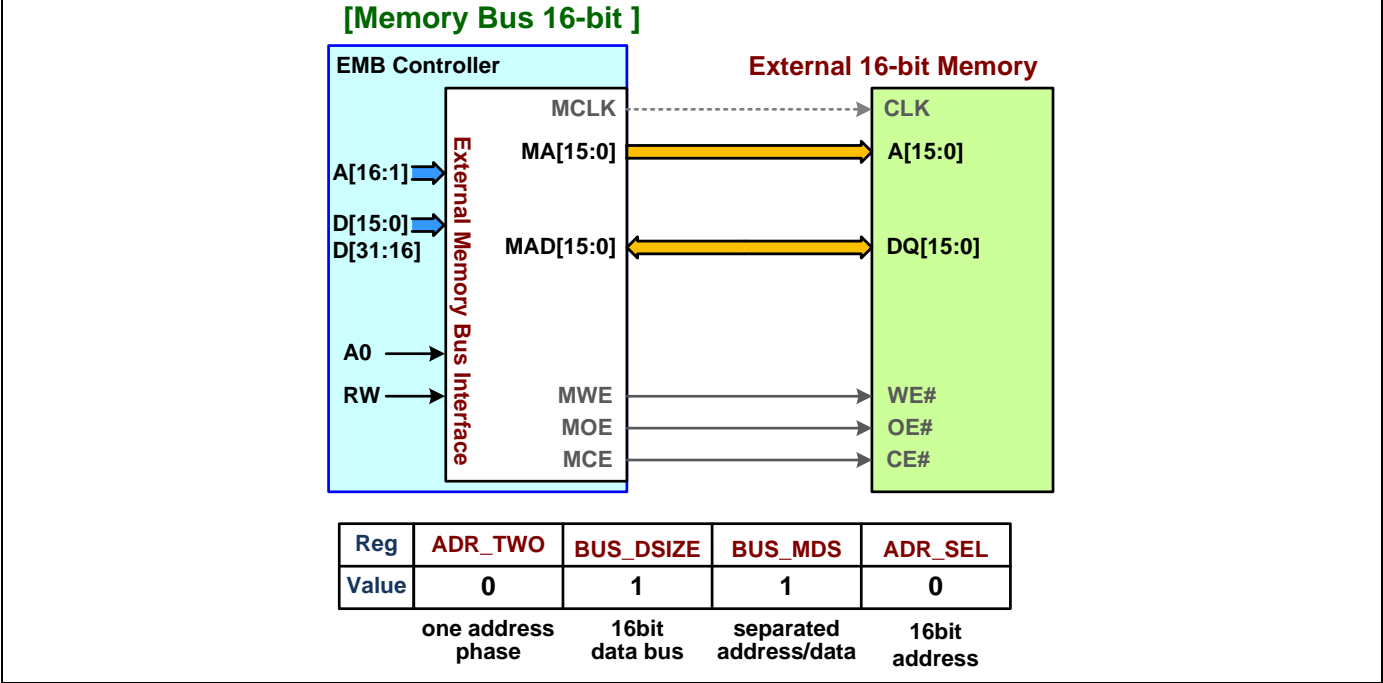
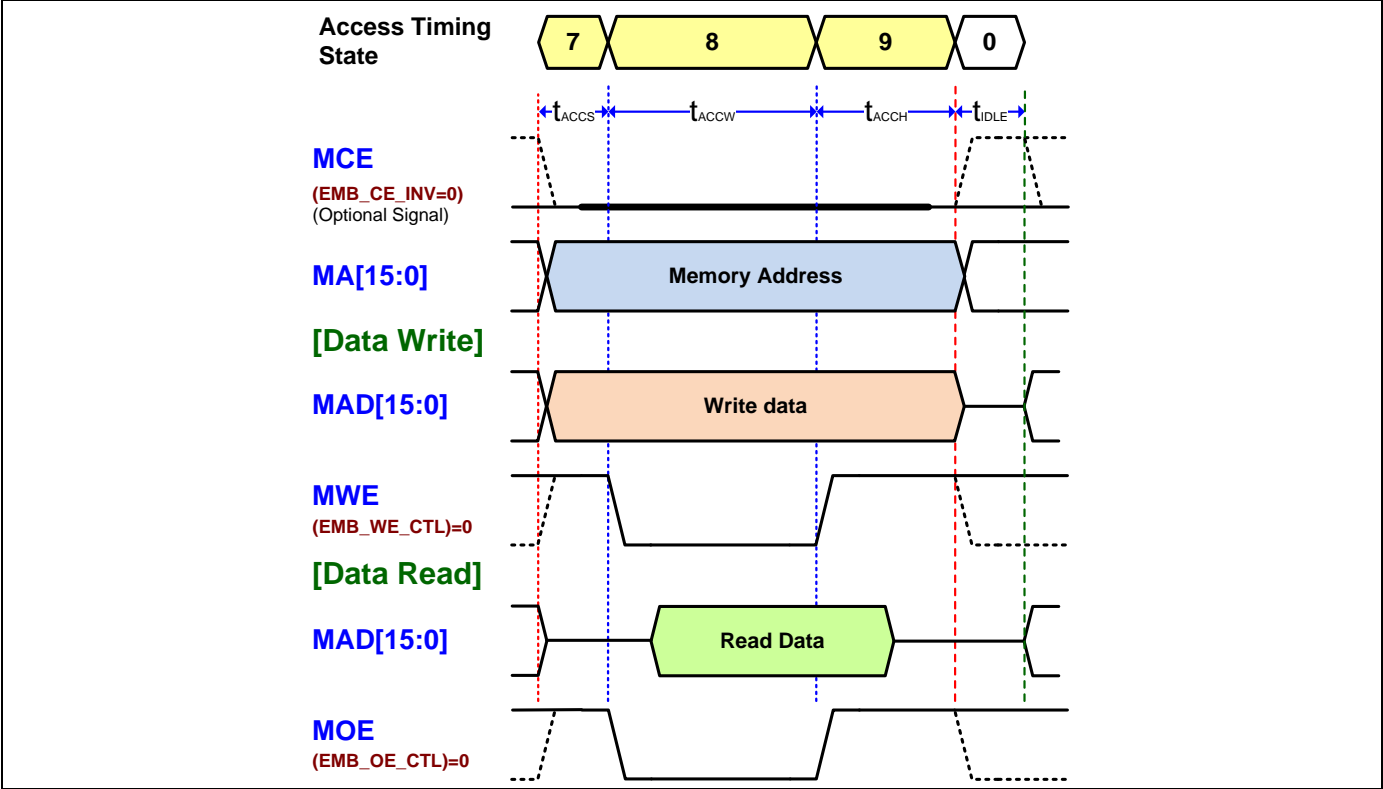


图 13-12. EMB 地址/数据时序 – 16bit MA + 16bit MD



13.10.2. EMB 16 位 MA 和复用 16 位 MD

该模式被设置为独立地 16 位地址和复用的带 1 个地址锁存信号 **MALE** 的 16 位地址/数据。该地址模式可被设置为 16、24、30 位地址空间。没有时序状态 0、1、2、3。

图 13-13. EMB 地址/数据接口 – 16bit MA + 16bit MAD

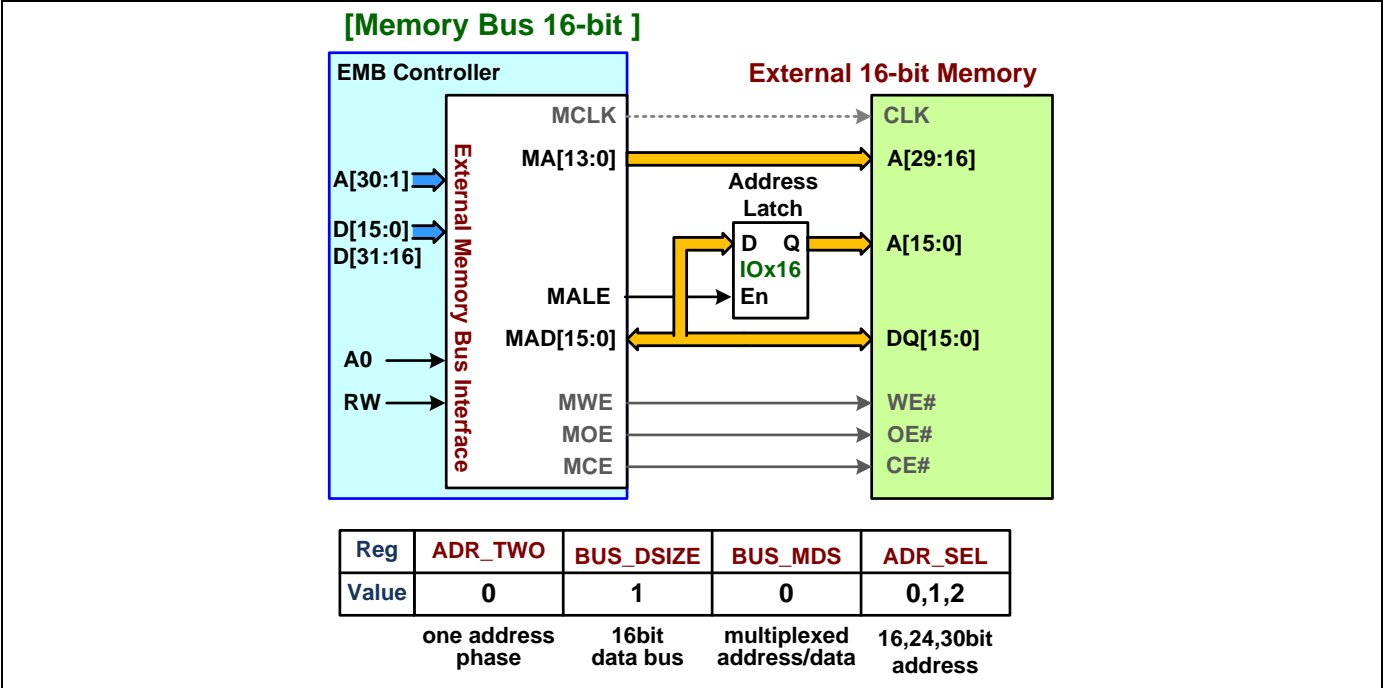
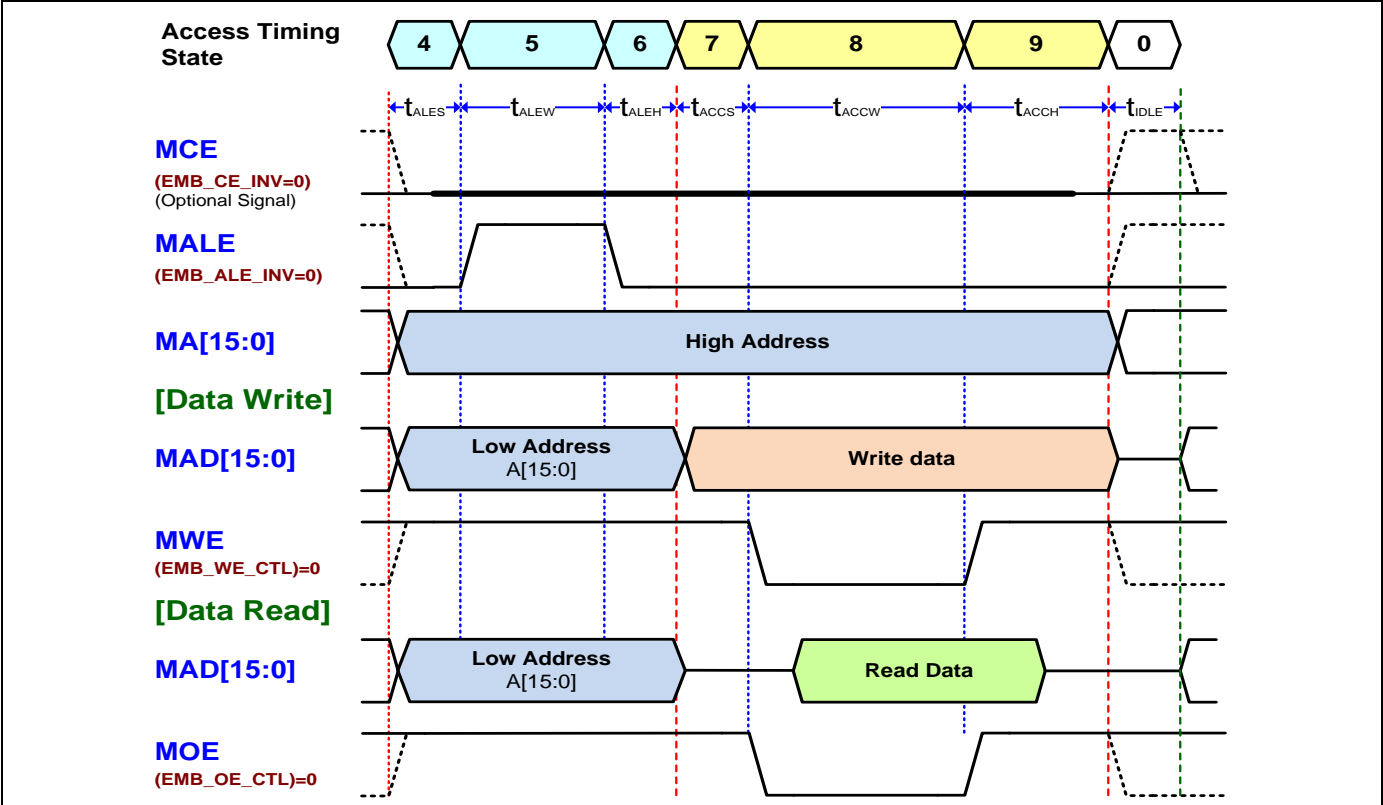


图 13-14. EMB 地址/数据时序 – 16bit MA + 16bit MAD



13.10.3. EMB 复用 16 位具有 2 个地址相位的 MAD

该模式被设置为复用的带 2 个地址锁存信号 **MALE** 和 **MALE2** 的 16 位地址/数据。该地址模式可被设置为 16、24、30 位地址空间。

图 13-15. EMB 地址/数据接口 - 复用 16 位具有 2 个地址相位的 MAD

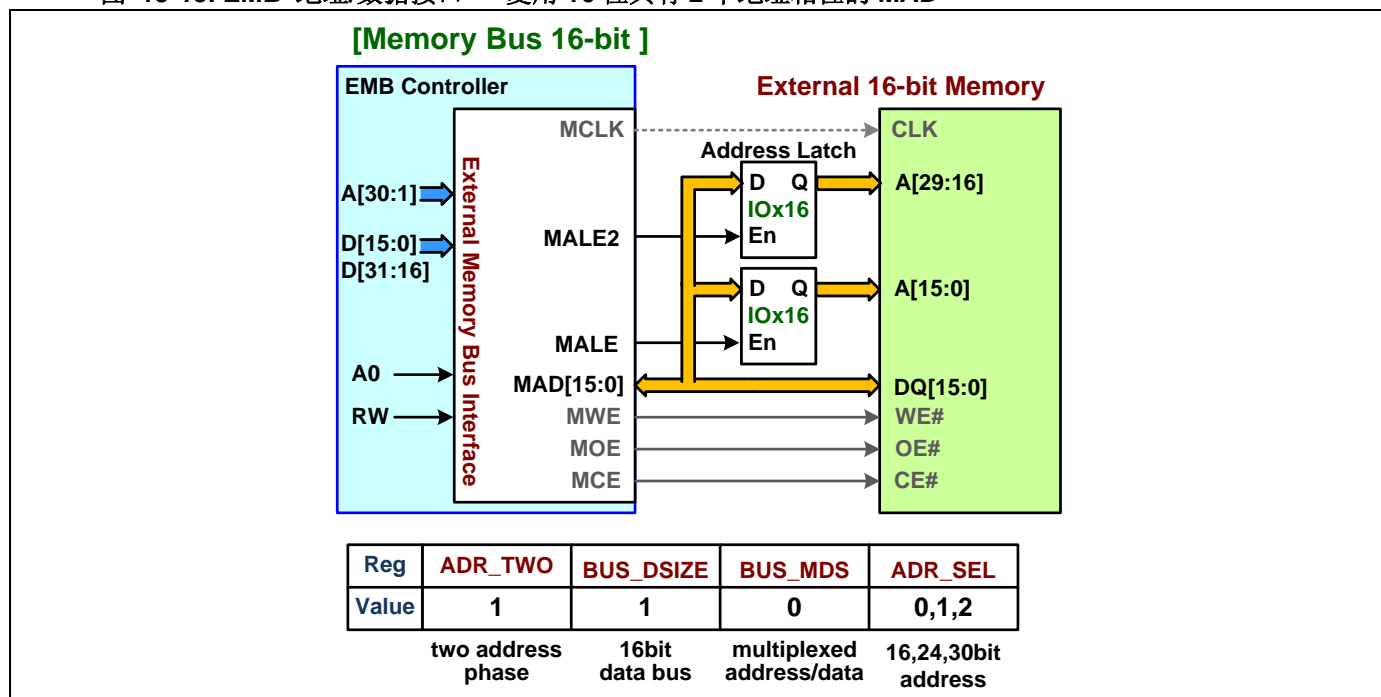
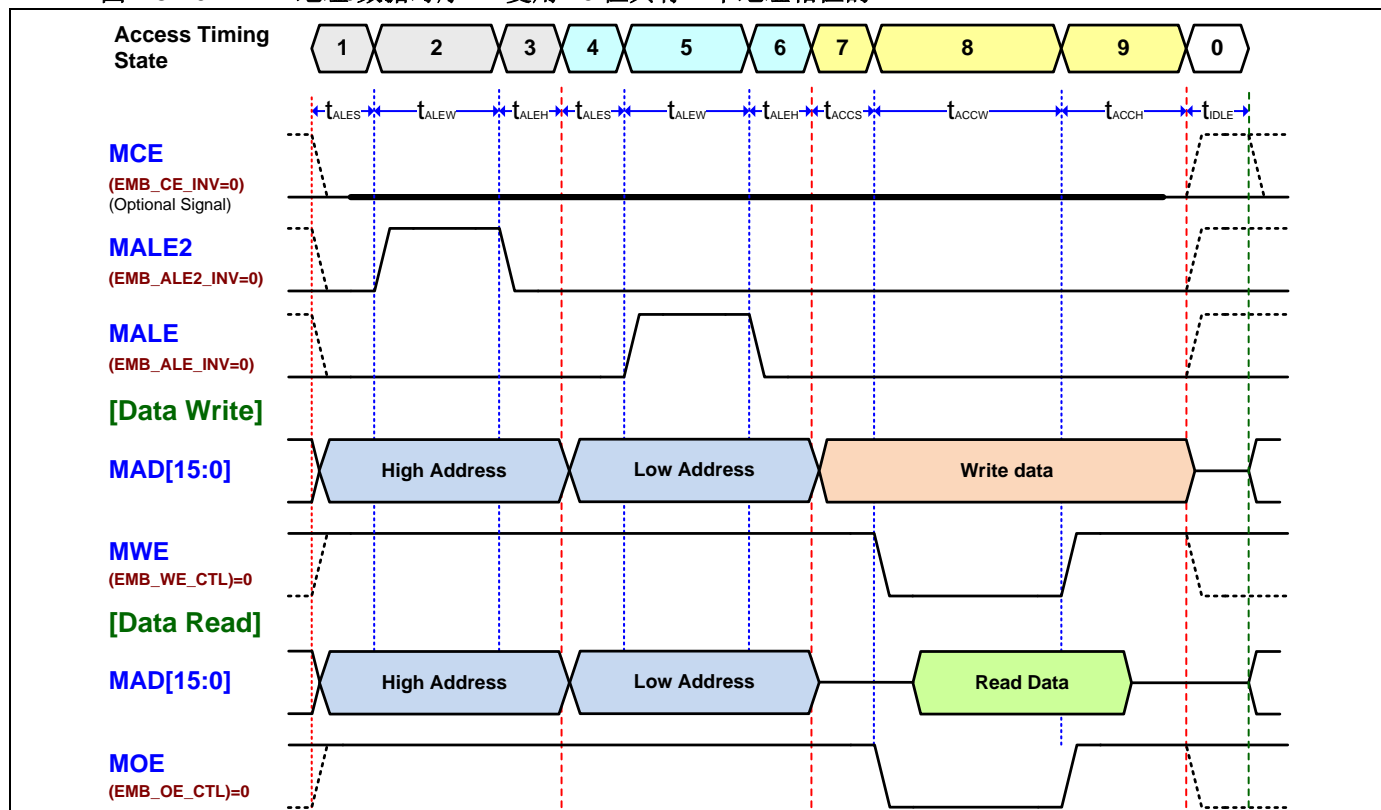


图 13-16. EMB 地址/数据时序 - 复用 16 位具有 2 个地址相位的 MAD



13.11. EMB 设备接口和时序

EMB 控制器支持访问 SRAM、NOR/NAND 闪存、8080 接口 LCD 外部设备。用户需通过软件根据外部设备特性初始化 EMB 数据总线宽度、EMB 地址模式、EMB 地址和数据复用模式、EMB 写保护和 EMB 时序设置。参照“[EMB 内存控制](#)”和“[EMB 地址和数据接口模式](#)”节以获取更多关于设置的信息。

用户需通过外部设备的引脚特性规划 EMB IO 信号映射外部设备的 IO 信号。参照“[外部设备的信号映射建议](#)”以获取更多关于 EMB 信号到外部设备引脚建议的信息。

用户还需要设置芯片 GPIO AFS 设置用于 EMB 控制。参照“[EMB 信号引脚建议](#)”节以获取关于 GPIO AFS 引脚建议信息。

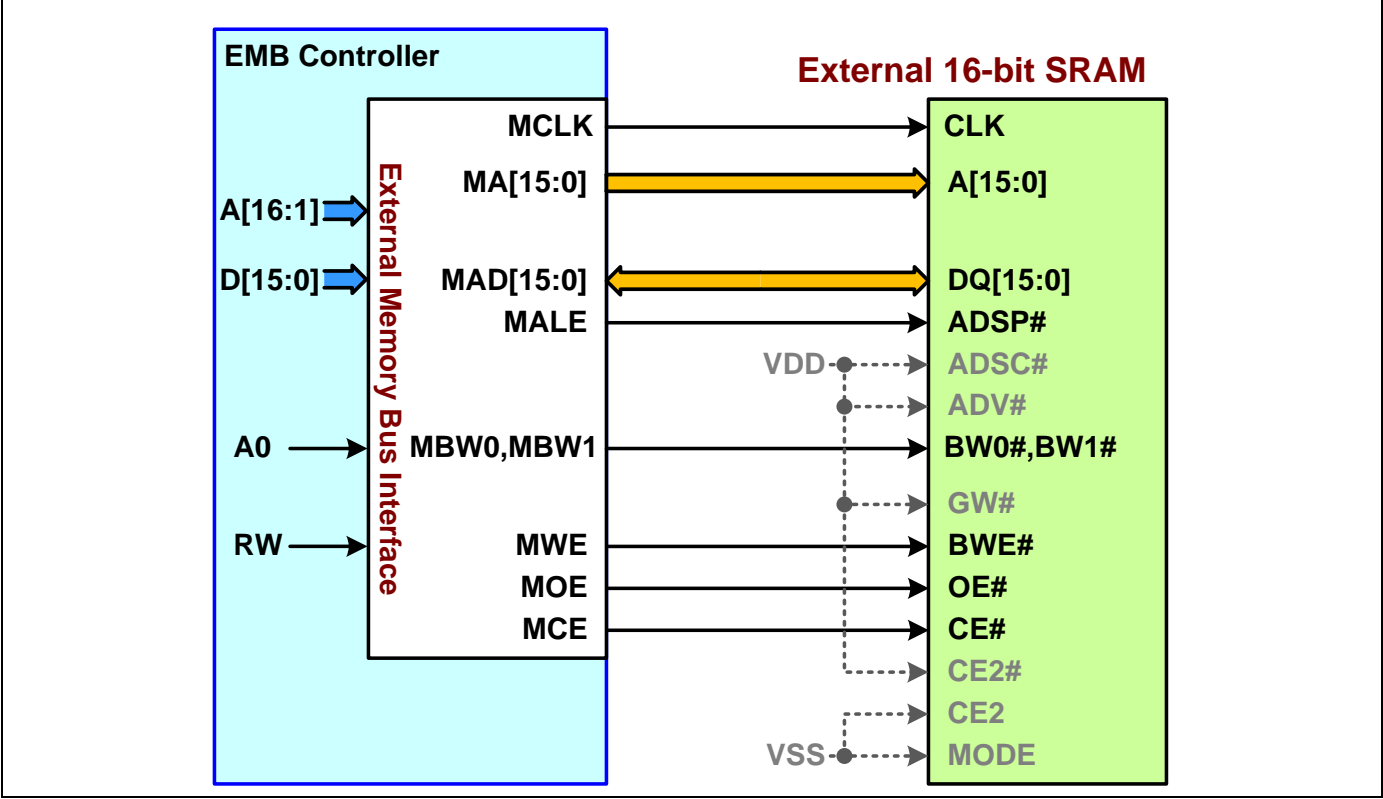
13.11.1. SRAM 接口和访问时序

● SRAM 接口

通常，SRAM 数据总线是 16、32 位或更大，EMB 可支持同步和异步 SRAM，但是只支持 16 位数据总线。当连接到异步 SRAM，MCLK 在数据传输中不是必要的。

下面的图表展示了 SRAM 接口的连接建议。

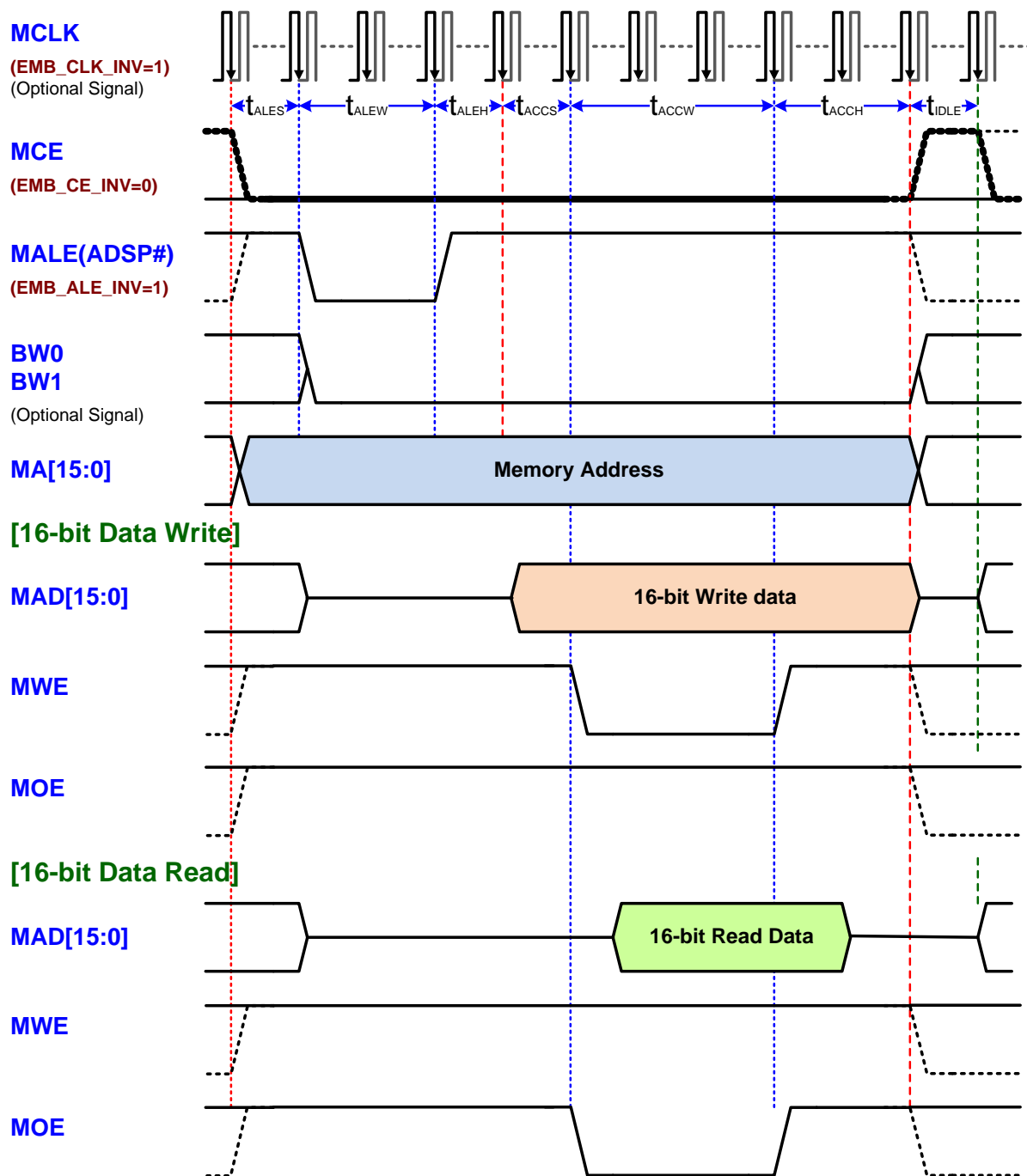
图 13-17. EMB SRAM 接口



- SRAM 16 位数据访问时序

下面的图表展示了 16 位 SRAM 接口的 16 位数据访问时序。

图 13-18. EMB SRAM 16 位数据时序



<Note-1> : MCLK is optional signal for synchronous SRAM.

<Note-2> : BW0/BW1 are optional signals for 16-bit data bus SRAM with byte write capability.

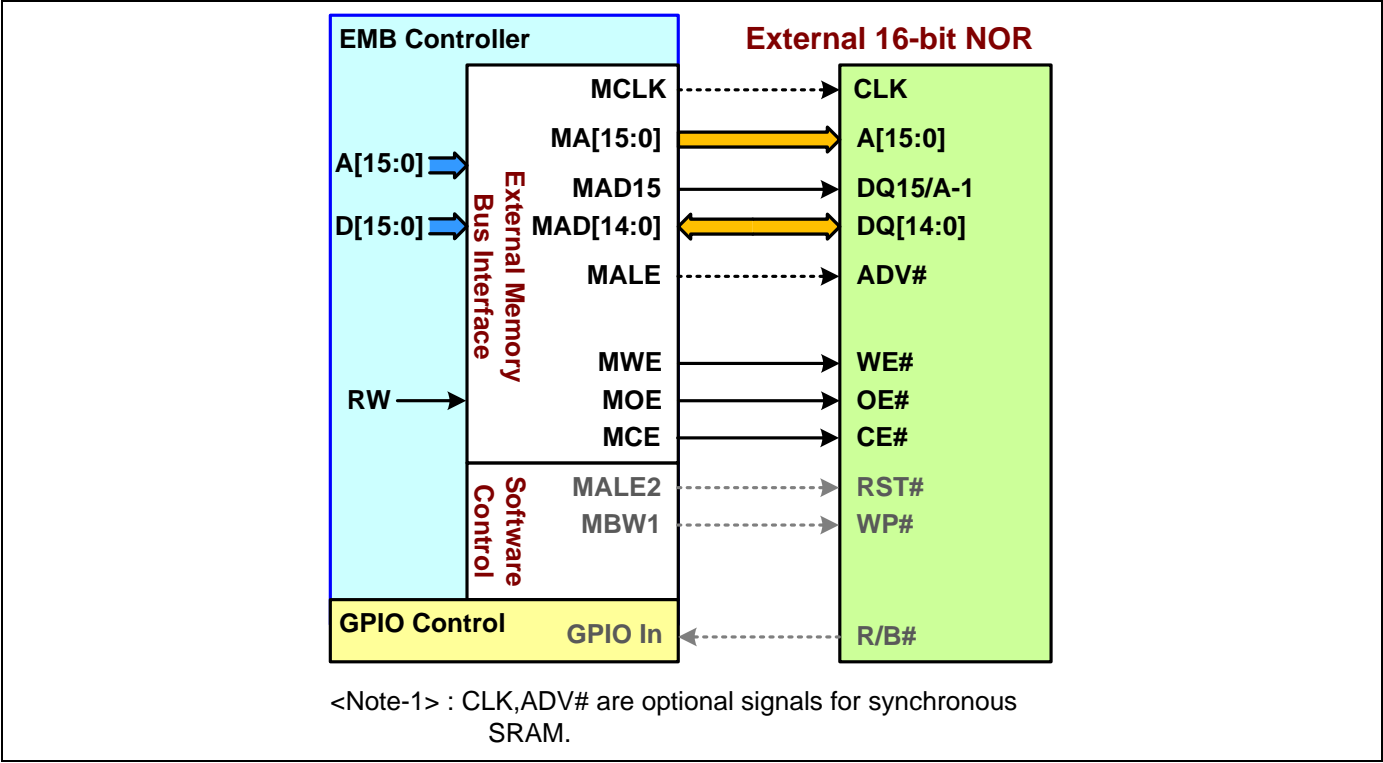
13.11.2. NOR-Flash 接口和访问时序

● NOR-Flash 接口

EMB 支持同步和异步 NOR-flash。此外,EMB 支持 16 位数据总线用于 NOR-flash。当连接到异步时 NOR-flash,在数据传输中, **MCLK** 不是必要的。

下面的图表展示了 NOR-flash 接口的连接建议。

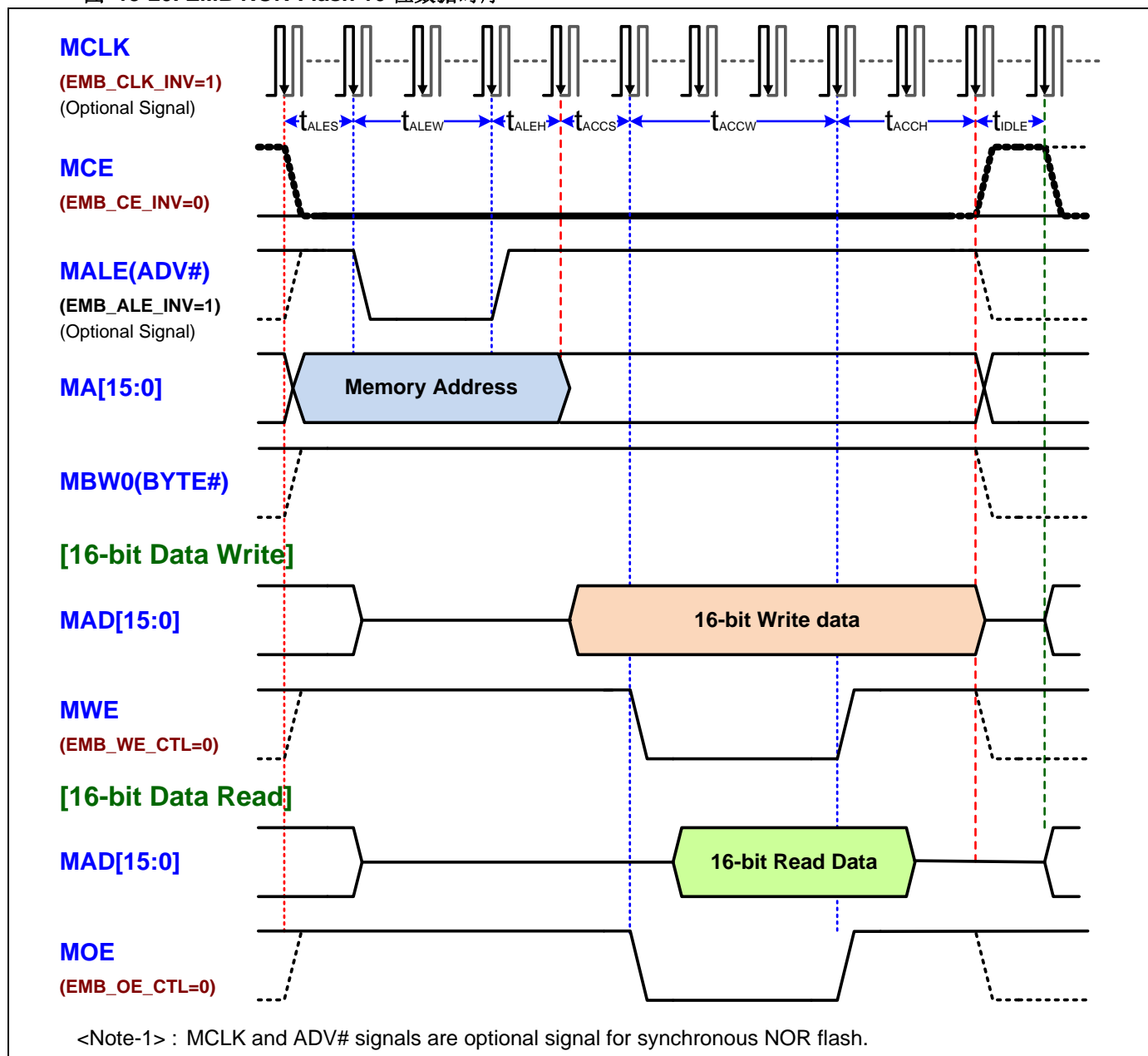
图 13-19. EMB NOR-Flash 接口



- NOR-Flash 16 位数据访问时序

下面的图表展示了 16 位 NOR-flash 的 16 位数据访问时序。

图 13-20. EMB NOR-Flash 16 位数据时序



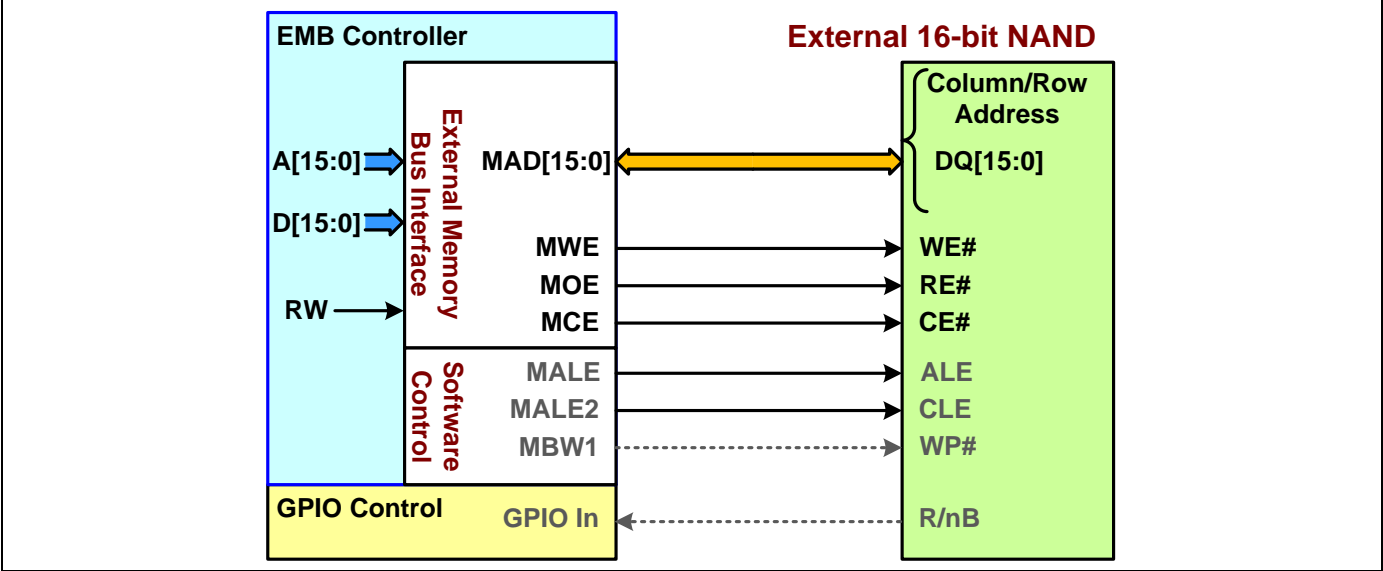
13.11.3. NAND-Flash 接口和访问时序

● NAND-Flash 接口

通常，NAND-flash 只支持异步数据传输，EMB 支持 16 位数据总线用于 NAND-flash。

下面的图表展示了 NAND -flash 接口的连接建议。

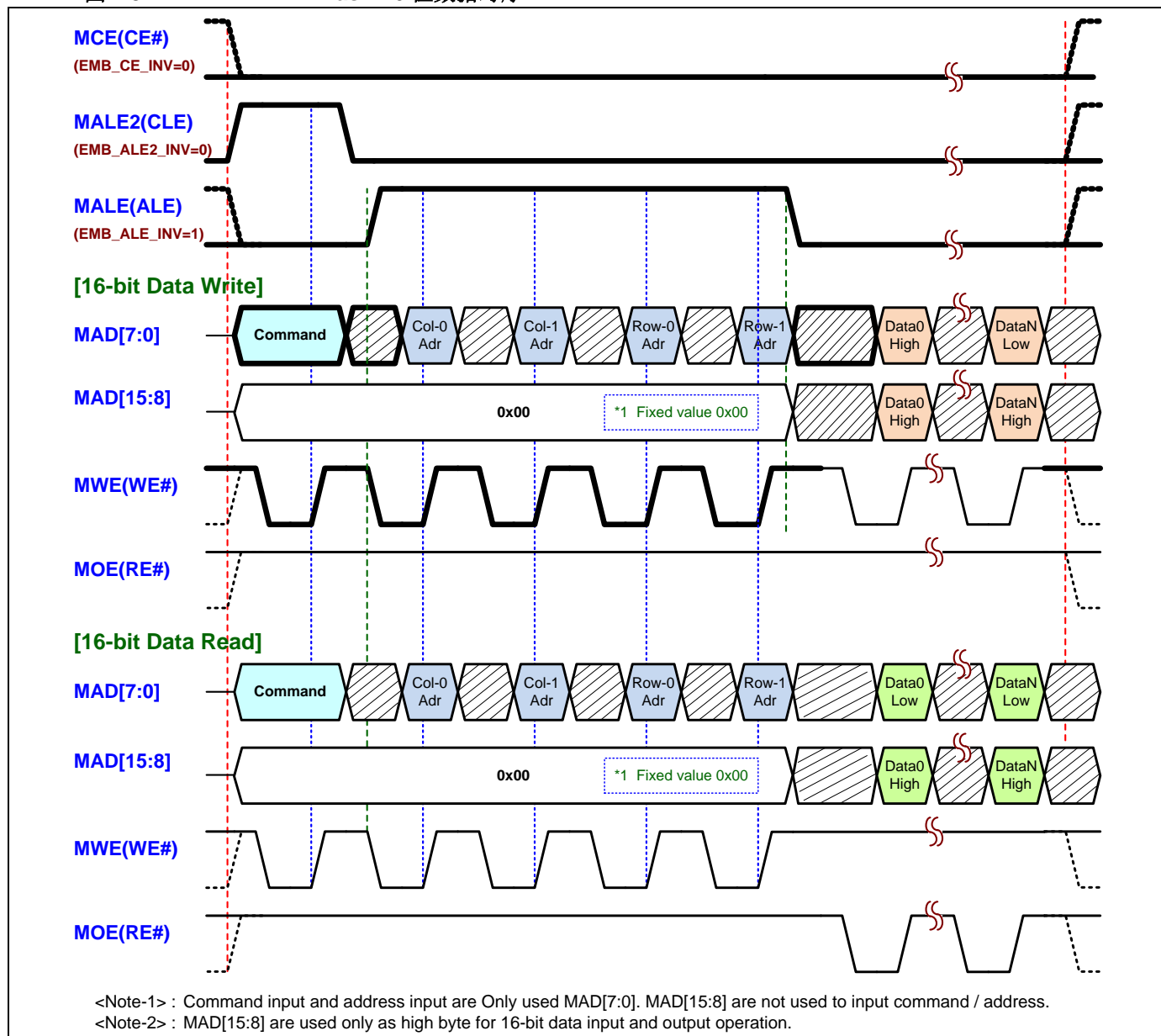
图 13-21. EMB NAND-Flash 接口



● NAND-Flash 数据访问时序

下面的图表展示了 NAND -flash 的数据访问时序。

图 13-22. EMB NAND-Flash 16 位数据时序

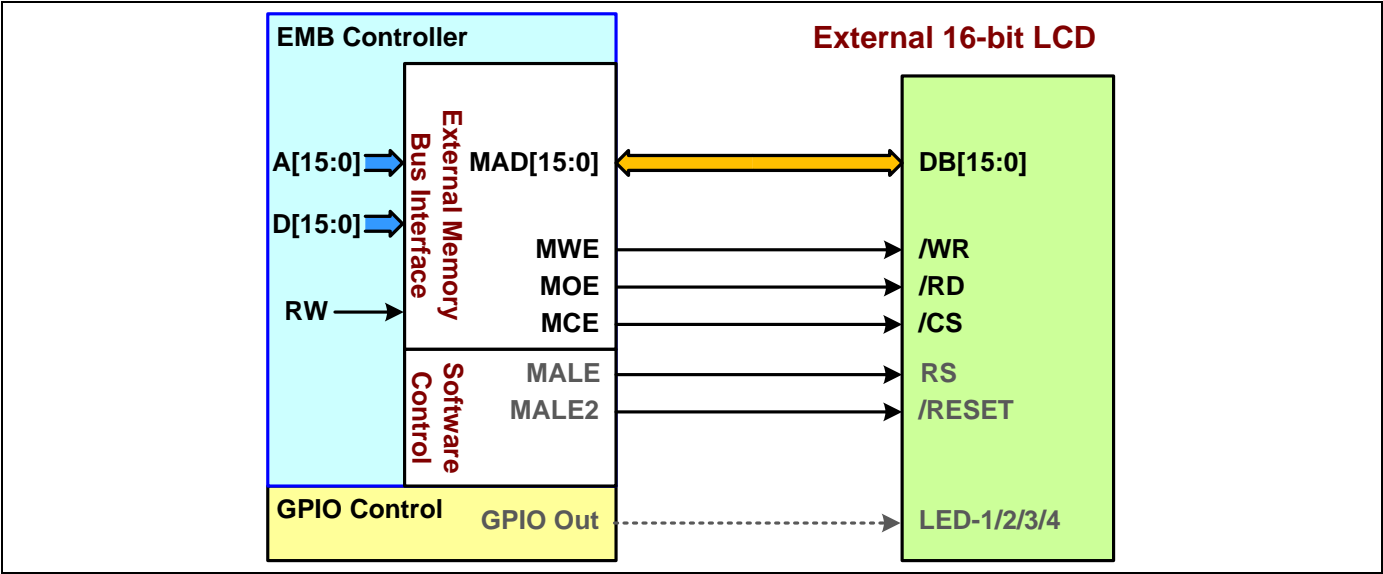


13.11.4. LCD 接口和访问时序

● LCD 接口

EMB 为 8080 LCD 接口支持 16 位数据总线。下面的图表展示了 LCD 接口的连接建议。

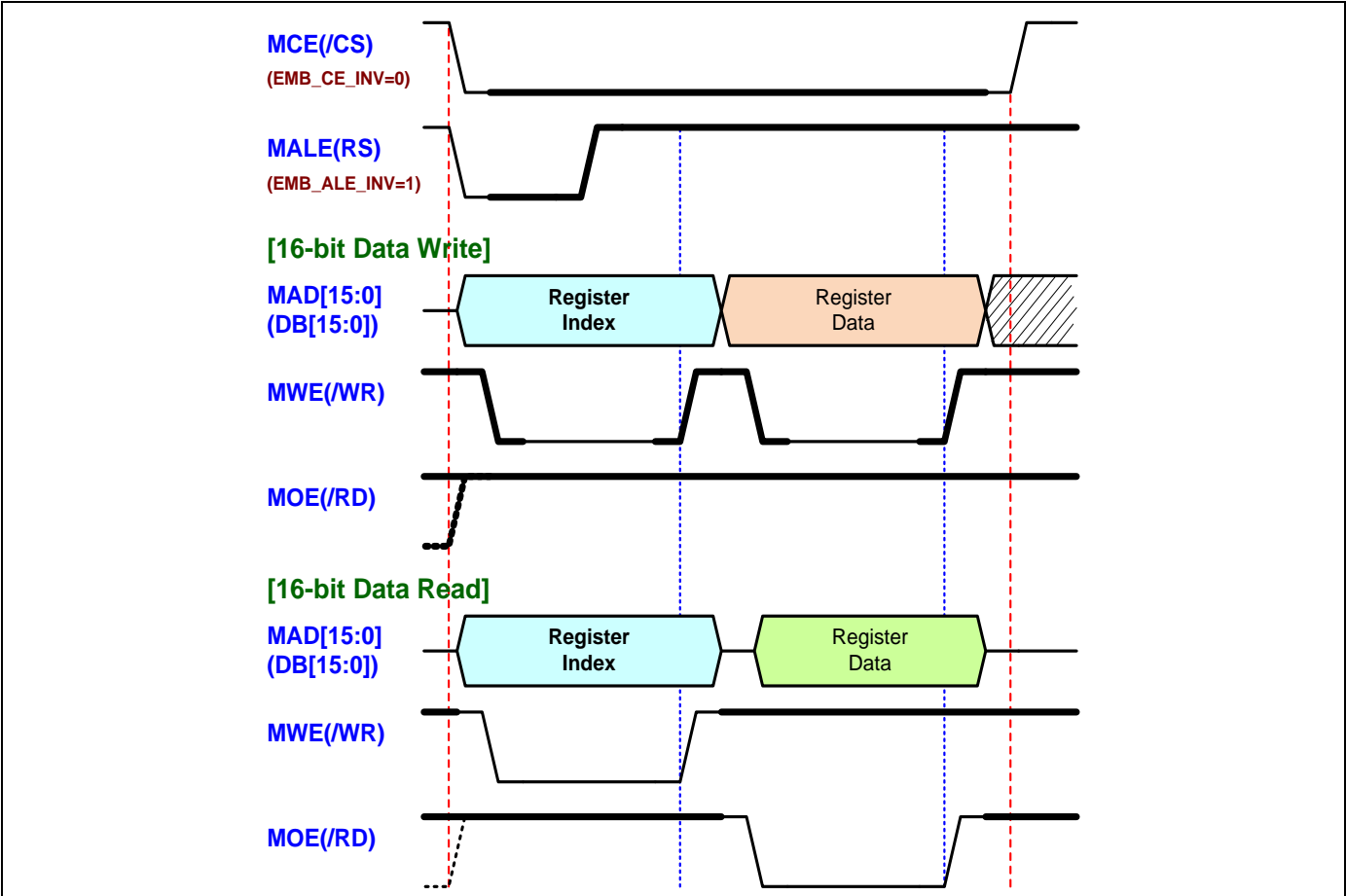
图 13-23. EMB LCD 接口



● LCD 16 位寄存器访问时序

下面的图表展示了 LCD 接口的 16 位寄存器访问时序。

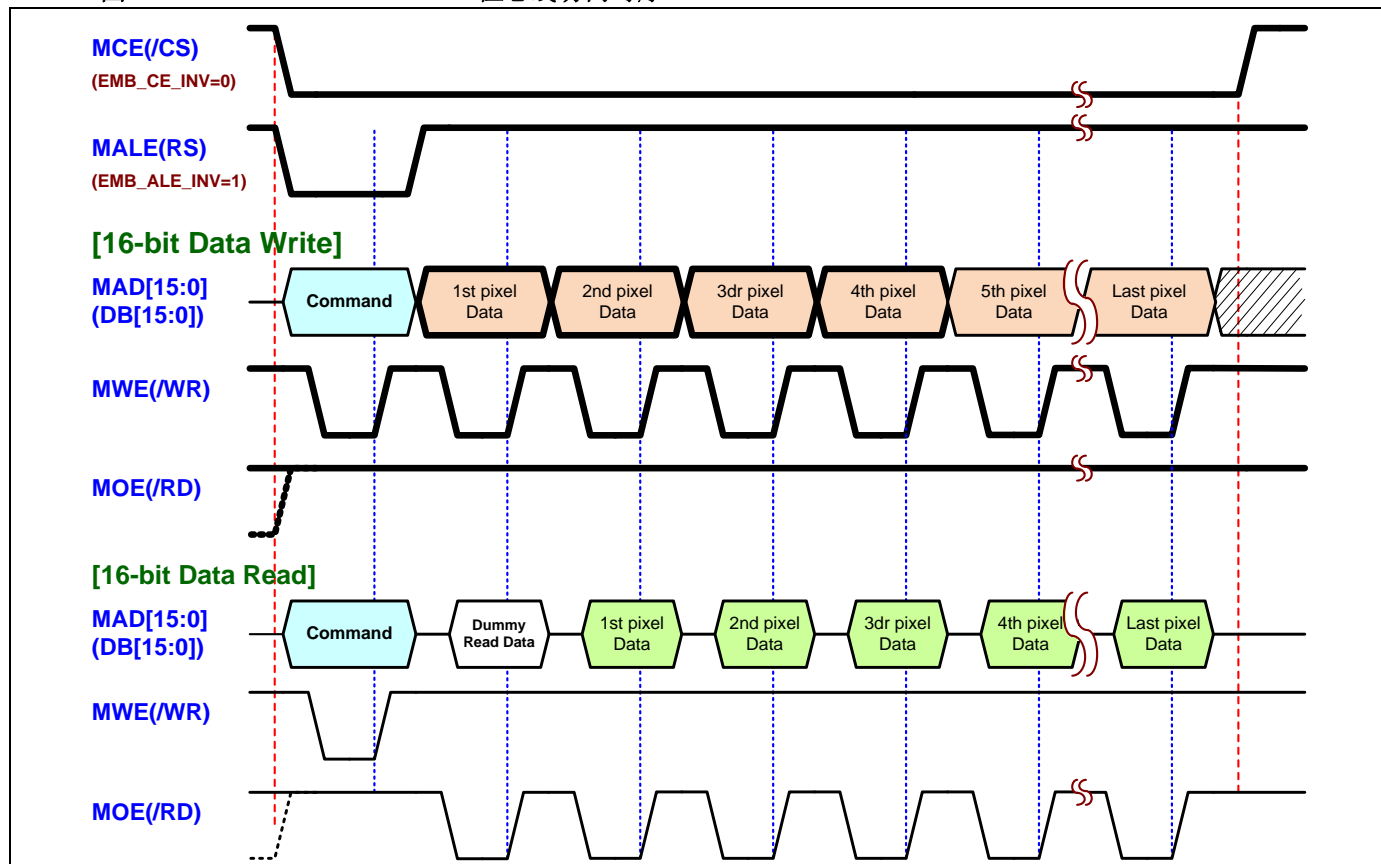
图 13-24. EMB LCD 寄存器 16 位总线访问时序



- LCD GRAM 访问时序

下面的图表展示了 LCD 接口的 GRAM 访问时序。

图 13-25. EMB LCD GRAM 16 位总线访问时序



13.12. EMB DMA 操作

13.12.1. DMA 模块设置

当芯片支持 DMA（直接内存访问）控制器时，用户可以在 DMA 数据传输前，在 DMA 模块中设置关于来源/目的设备、通道请求仲裁等 DMA 设置。DMA 来源和目的可以是内存或外设。

参照 DMA 章以获取更多关于 DMA 模组设置的细节。

13.12.2. EMB DMA 控制

DMA 设置完成后，用户需设置 EMB 模块的 DMA 使能位 **EMB_DMA_EN**。

最后，相关的通道请求起始位 **DMA_CHn_REQ** 是必须的，用于设置 DMA 传输启动(n = DMA 通道标号)。然后传输源和目的设备会置起 RX/TX 请求信号到 DMA 控制器，DMA 控制器便会置起接收信号到请求源/目的设备。此时，该数据传输连接是用于 DMA 传输的。

13.13. EMB 应用注释

13.13.1. EMB 信号引脚建议

下面的表格展示了 EMB 映射外部设备信号的建议 AFS 引脚规划。这对 PCB 设计应用电路有降低难度的作用。

EMB 外部设备包含 SRAM、NOR/NAND 闪存、LCD 模块和 LCD RAM 设备。LCD RAM(SPI 线-OR)是将单片机芯片和 LCD RAM 设备结合在一起的一种特殊应用。

PB[3:0]引脚被保留下来用于设置 I2C 和 UART；**PC[5:4]**引脚被保留下来用于设置 SWD 调试信号；**PC6** 引脚被保留下来用于 MCU 芯片外部复位；**PC[14:13]**引脚被保留下来用于设置 MCU 外部 Xtal 引脚。当然这些引脚都可以被用户作为 GPIO 引脚。

表 13-5. EMB 接口信号和建议引脚表

MCU 引脚	SRAM		NOR		NAND		LCD Module		LCD RAM	
	(EMB_MAD_SWAP) =禁用		(EMB_MAD_SWAP) =禁用		(EMB_MAD_SWAP) =禁用		(EMB_MAD_SWAP) =使能		(EMB_MAD_SWAP) =使能	
	(EMB_MAD_BSWAP) =禁用		(EMB_MAD_BSWAP) =禁用		(EMB_MAD_BSWAP) =禁用		(EMB_MAD_BSWAP) =使能		(EMB_MAD_BSWAP) =使能	
	EMB 信号	SRAM 引脚	EMB 信号	NOR 引脚	EMB 信号	NAND 引脚	EMB 信号	LCD RAM 引脚	EMB 信号	LCD RAM 引脚
PA[15:0]	MA[15:0]	A[15:0]	MA[15:0]	A[15:0]						
PB0	GPIO / I2C1_SCL									
PB1	GPIO / I2C1_SDA									
PB2	URT2_TX									
PB3	URT2_RX									
PE0										BL_LED1 (*2)
PE1										BL_LED2 (*2)
PE2										BL_LED3 (*2)
PE3	MALE2	ADSP# (*4)	MALE2	ADV# (*4)						BL_LED4 (*2)
PB4	MALE	ADSP#	MALE	ADV#	MALE	ALE (*1)	MALE	RESET (*1)	MAD8	DB7
PB5	MOE	OE#	MOE	OE#	MOE	RE#			MAD9	DB6
PB6	MWE	WE#	MWE	WE#	MWE	WE#			MAD10	DB5
PB7	MCE	CE#	MCE	CE#	MCE	CE#	MCE	/CS (*1)	MALE2	RESET (*1)
PB8	MAD0	DQ0	MAD0	DQ0	MAD0	DQ0	MAD0	DB15	MAD0	DB15
PB9	MAD1	DQ1	MAD8	DQ8	MAD8	DQ8	MAD1	DB14	MAD1	DB14
PB10	MAD2	DQ2	MAD1	DQ1	MAD1	DQ1	MAD2	DB13	MAD2	DB13
PB11	MAD3	DQ3	MAD9	DQ9	MAD9	DQ9	MAD3	DB12	MAD3	DB12
PB12	MAD4	DQ4	MAD2	DQ2	MAD2	DQ2	MAD4	DB11	MAD4	DB11
PB13	MAD5	DQ5	MAD10	DQ10	MAD10	DQ10	MAD5	DB10	MAD5	DB10

PB14	MAD6	DQ6	MAD3	DQ3	MAD3	DQ3	MAD6	DB9	MAD6	DB9
PB15	MAD7	DQ7	MAD11	DQ11	MAD11	DQ11	MAD7	DB8	MAD7	DB8
PE8									MAD11	DB4
PE9							MOE	/RD	MOE	/RD
PC0							MWE	/WR	MWE	/WR
PC1	MAD8	DQ8	MAD4	DQ4	MAD4	DQ4	MAD8	DB7		
PC2	MAD9	DQ9	MAD12	DQ12	MAD12	DQ12	MAD9	DB6		
PC3	MAD10	DQ10	MAD5	DQ5	MAD5	DQ5	MAD10	DB5		
PC4	SWCLK									
PC5	SWDIO									
PC6	GPIO/RSTN						MBW1	RS (*1)	MALE	RS (*1)
PC7									MCE	/CS (*1)
PC8	MAD11	DQ11	MAD13	DQ13	MAD13	DQ13	MAD11	DB4		
PC9	MAD12	DQ12	MAD6	DQ6	MAD6	DQ6	MAD12	DB3	MAD12	DB3
PC10	MAD13	DQ13	MAD14	DQ14	MAD14	DQ14	MAD13	DB2	MAD13	DB2
PC11	MAD14	DQ14	MAD7	DQ7	MAD7	DQ7	MAD14	DB1	MAD14	DB1
PC12	MAD15	DQ15	MAD15	DQ15/A-1	MAD15	DQ15/A-1	MAD15	DB0	MAD15	DB0
PC13	XIN									
PC14	XOUT									
PE12	MBW0	BW0	MBW0	BYTE#						
PE13	MBW1	BW1	MBW1	WP# (*1)	MBW1	WP# (*1)				
PE14			MALE2	RST# (*1)	MALE2	CLE (*1)				
PE15				RY/BY# (*3)		RY/BY# (*3)				
PD0	MCLK	CLK	MCLK	CLK						

*1: 软件模式下寄存器设置直接输出控制

*2: 从 GPIO 输出引脚输出

*3: 从 GPIO 输入引脚输入

*4: 设置引脚用于 ADSP#或 ADV#

14. APB 一般控制

14.1. 简介

ON	SLEEP	STOP
----	-------	------

The module can be running in ON and SLEEP modes only.

该芯片内置 1 个 APB (APB 总线一般控制)模块用于 APB 设备的一般控制。

注释: (x = 模块标号, n= OBM 设置标号, m= OBM 通道标号)会被用于该章的寄存器、信号和引脚/端口描述中。

14.2. 特性

- 定时器同步使能全局控制用于 TMx 定时器模块
- 定时器内部触发/时钟源选择用于 TMx 定时器
- 红外遥控输出

❖ MG32F02A132/072

- OBM(输出信号中止和调制)控制
 - 支持单组 OBM 输出信号中止和调制控制

❖ MG32F02A032

- OBM(输出信号中止和调制)控制
 - 支持两组 OBM 输出信号中止和调制控制

14.3. 配置

14.3.1. 芯片配置

下面的表格展示了该芯片的 APB 功能配置。

表 14-1. APB 配置

芯片	APB 模块			
	OBM0	OBM1	定时器全局控制	红外调制输出
MG32F02A132		V	V	V
MG32F02A072		V	V	V
MG32F02A032	V	V	V	V

注释 V: 包含

14.4. 中断和事件

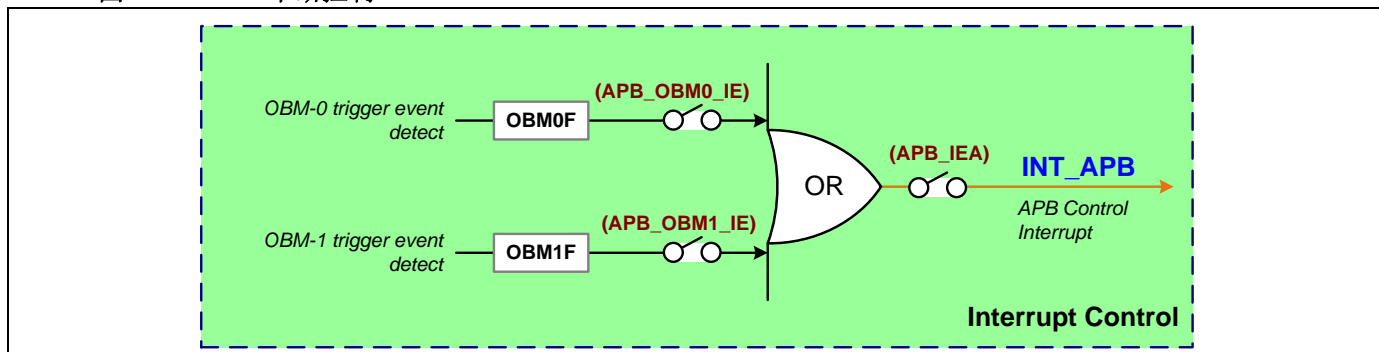
APB 模块中有 1 种信号 **INT_APB**。**INT_APB** 发送到外部中断控制器（EXIC）作为中断事件。

14.4.1. APB 中断控制和状态

中断标识是用于中断服务程序（ISR）流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **APB_IEA** 用于使能和禁用该模块的所有中断源。

有一些只读状态位用于提供内部控制状态。参照相关寄存器状态位的描述以获取更多信息。

图 14-1. APB 中断控制



14.4.2. APB 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

[注释]: OBM0F 和 **APB_OBM0_IE** 寄存器不支持于 MG32F02A132/072。

- **OBM0F / OBM1F**

OBM-0/1 中止触发事件检测标志(**APB_OBM0F**, **APB_OBM1F**)。相关的寄存器位是 **APB_OBM0_IE** 和 **APB_OBM1_IE**。

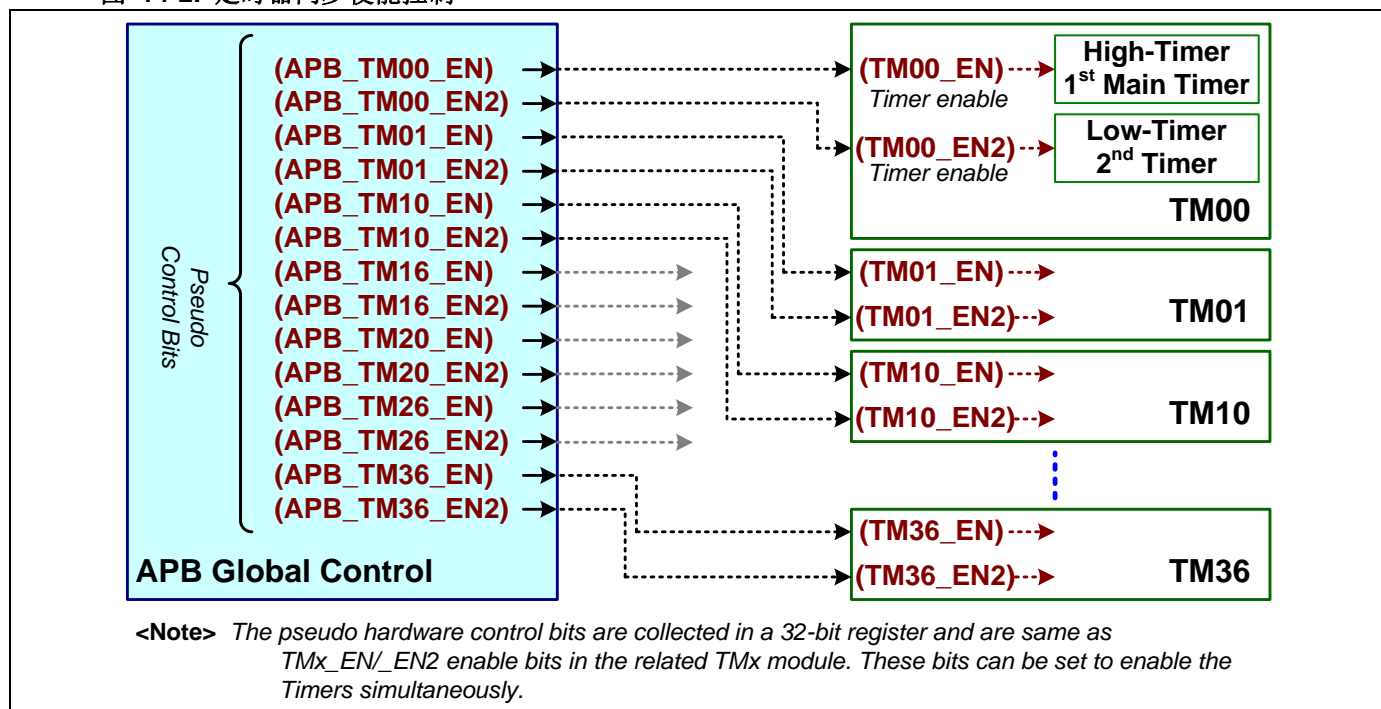
14.5. 定时器一般控制

14.5.1. 定时器同步使能控制

定时器内建 TMx 模块支持 3 种定时器工作模式：(1) Cascade 模式(2) Separate 模式(3) Full-Counter 模式。所以 1 个 TMx 模块的定时器可为 1 个全计数器或分离成 2 个计数器。每个 TMx 模块都有 2 个定时器使能控制位 **TMx_EN** 和 **TMx_EN2**。**TMx_EN** 位用于使能这些工作模式的全计数器，高位计数器或主计数器。**TMx_EN2** 位用于使能低位定时器或 2nd 定时器用于 Cascade 或 Separate 模式。

APB 模块里，**APB_TMx_EN** 和 **APB_TMx_EN2** 的控制位与定时器使能 **TMx_EN** 和 **TMx_EN2** 的控制位相同。这些控制寄存器位被设计在 1 个 32 位寄存器里。然而，所有 TMx 模块的所有的全计数器或分离计数器都可以通过固件设置 **APB_TMx_EN** 或 **APB_TMx_EN2** 寄存器同步设置使能或禁用。(x = 定时器模块标号)

图 14-2. 定时器同步使能控制



[注释]: **APB_TM20_EN, APB_TM20_EN2, APB_TM26_EN 和 APB_TM26_EN2** 不支持于 MG32F02A032。

14.5.2. 定时器一般触发/时钟源选择

ITR6 和 **ITR7** 是用于所有定时器模块的触发事件的一般信号或时钟信号。用户可通过 **APB_ITR6_MUX** 和 **APB_ITR7_MUX** 寄存器选择触发源信号。**APB_ITR6** 和 **APB_ITR7** 信号可被其他 TMx 定时器, URTx, ADC0, RTC 模块和 EXIC 全局中断事件选择。如下表格所示。

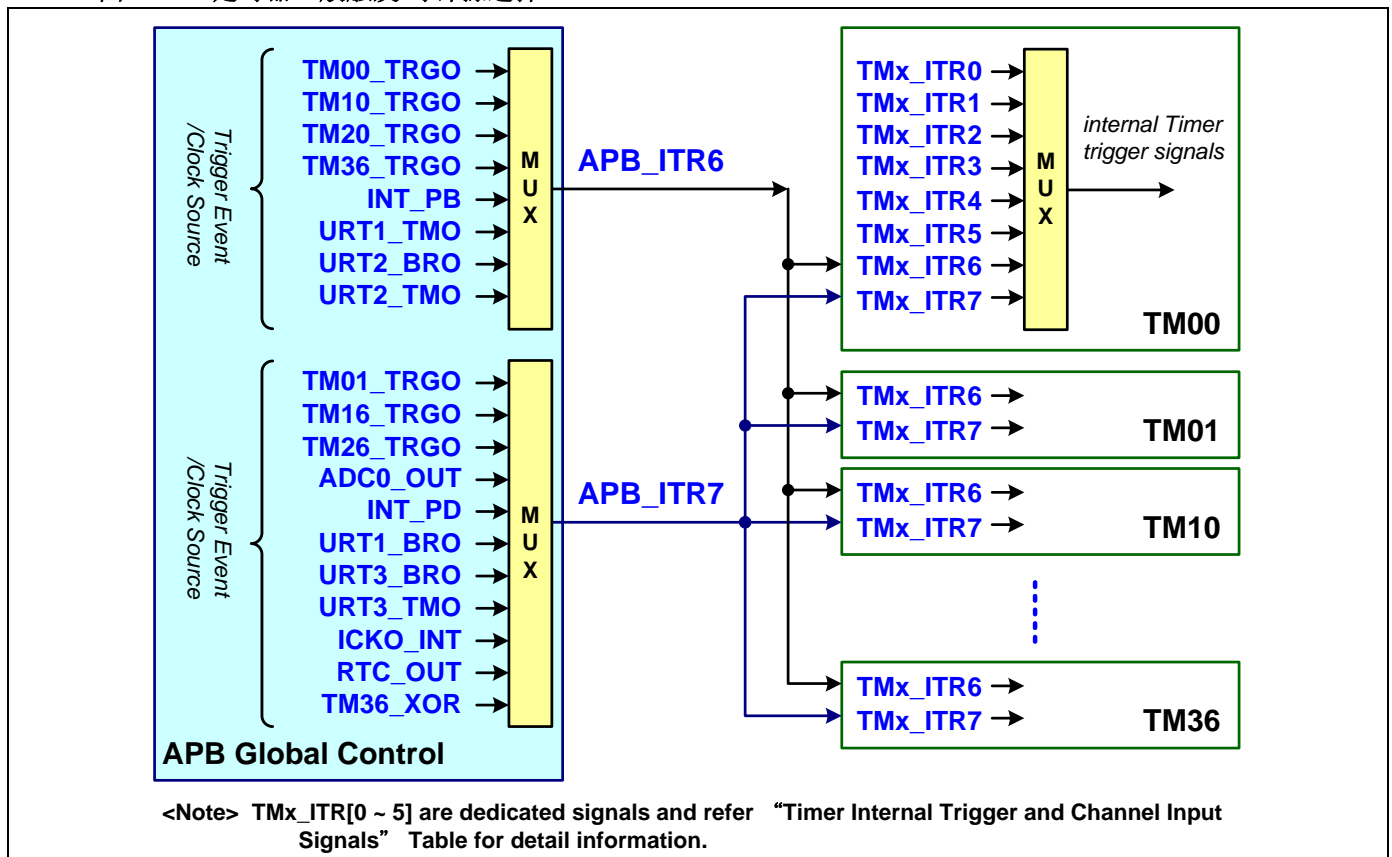
表 14-2. 定时器一般 ITR6/ITR7 信号表格

芯片	MG32F02A132/072		MG32F02A032	
ITRx	ITR6	ITR7	ITR6	ITR7
	APB 寄存器		APB 寄存器	
触发源	ITR6_MUX	ITR7_MUX	ITR6_MUX	ITR7_MUX
TRG0 信号	TM00_TRGO	TM01_TRGO	TM00_TRGO	TM01_TRGO
TRG1 信号	TM10_TRGO	TM16_TRGO	TM10_TRGO	TM16_TRGO
TRG2 信号	TM20_TRGO	TM26_TRGO	-	-
TRG3 信号	TM36_TRGO	ADC0_OUT	TM36_TRGO	ADC0_OUT
TRG4 信号	INT_PB	INT_PD	INT_PB	INT_PD
TRG5 信号	URT1_TMO	URT1_BRO	URT1_TMO	URT1_BRO
TRG6 信号	URT2_BRO	URT3_BRO	-	-
TRG7 信号	URT2_TMO	URT3_TMO	-	-
TRG8 信号	-	ICKO_INT	-	ICKO_INT
TRG9 信号	-	RTC_OUT	-	RTC_OUT
TRG10 信号	-	TM36_XOR	-	TM36_XOR

下面的图表展示了定时器一般触发/时钟源选择。

[注释]: TM20_TRGO, TM26_TRGO, URT2_BRO, URT3_BRO, URT2_TMO 和 URT3_TMO 不支持于 MG32F02A032。

图 14-3. 定时器一般触发/时钟源选择



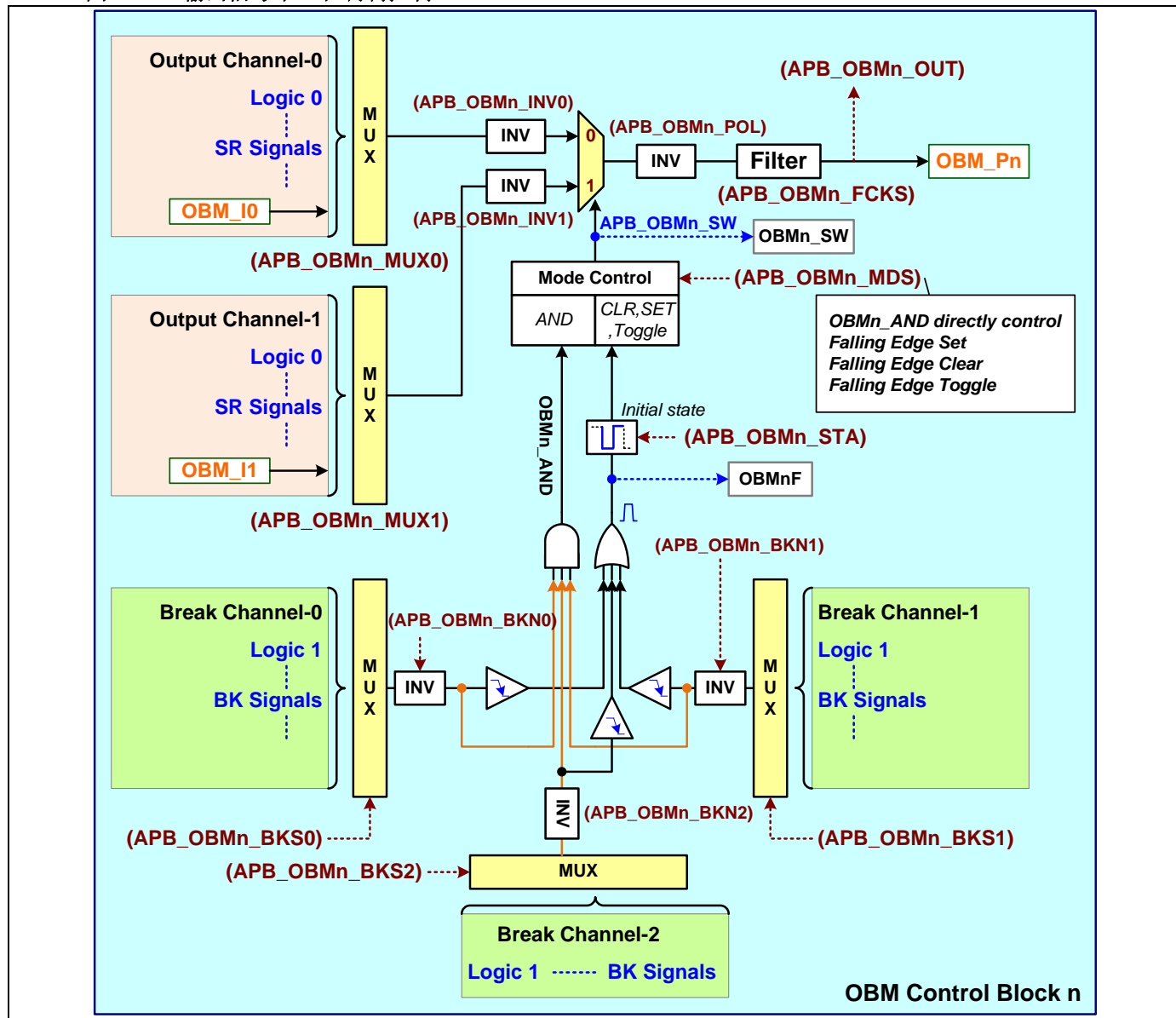
14.6. OBM 控制

APB 模块含有 2 组完全相同的输出信号中止和调制 (OBM) 块。OBM 块是用于中止 **TMx_CKO**, **APB_ITR6**, 和 **APB_ITR7** ... 的其中一个输出信号或如下图表所示作为信号调制。

每组 OBM 中有 2 个 OBM 输出源通道, 3 个 OBM 中止源通道和 1 个 OBM 工作模式控制块。

[注释]: MG32F02A132/072 只支持 1 组 OBM 块 – OBM1。

图 14-4. 输出信号中止和调制控制



14.6.1. OBM 输出通道

在 **APB_OBMn_MUXm** 寄存器中可以选择两种 OBM 输出源通道，可以选择输出源为“中止”或“调制”。每个输出通道都有 1 个信号反相控制位 **APB_OBMn_INVm**。(n = OBM 组号; m=OBM 通道标号)

对于 OBM 输出路径，控制位 **APB_OBMn_POL** 用于决定输出信号极性。OBM 内建 1 个数字滤波器用于输出信号。用户可通过 **APB_OBMn_FCKS** 寄存器选择 APB 时钟、APB 时钟 8 分频或定时器 TM00 触发输出 (**TM00_TRGO**)作为滤波器的时钟源。

状态位 **APB_OBMn_OUT** 反映 **OBMn** 输出信号的实时状态。

下面的表格展示了通过寄存器设置对这两个输出通道 MUX 的源信号选择。

表 14-3. OBM 块输出通道信号表

芯片	MG32F02A132/072		MG32F02A032	
通道	通道-0	通道-1	通道-0	通道-1
	APB 寄存器		APB 寄存器	
OBM 输出源	OBMn_MUX0	OBMn_MUX1	OBMn_MUX0	OBMn_MUX1
SR0 信号	0	0	0	0
SR1 信号	INT_PA	INT_PB	INT_PA	INT_PB
SR2 信号	INT_PC	INT_PD	INT_PC	INT_PD
SR3 信号	-	-	-	-
SR4 信号	TM00_CKO	TM01_CKO	TM00_CKO	TM01_CKO
SR5 信号	TM10_CKO	TM16_CKO	TM10_CKO	TM16_CKO
SR6 信号	TM20_CKO	TM26_CKO	-	-
SR7 信号	TM36_CKO	-	TM36_CKO	-
SR8 信号	TM20_OC00	TM20_OC01	-	-
SR9 信号	TM36_OC00	TM36_OC01	TM36_OC00	TM36_OC01
SR10 信号	TM36_OC2	TM36_OC3	TM36_OC2	TM36_OC3
SR11 信号	-	-	-	-
SR12 信号	OBM_I0	OBM_I1	OBM_I0	OBM_I1
SR13 信号	ITR6	ITR7	ITR6	ITR7
SR14 信号	-	ICKO_INT	-	ICKO_INT
SR15 信号	-	-	-	-

注释-1：“-” = 保留，“0” = 逻辑-0

14.6.2. OBM 中止通道

一共有 3 个 OBM 中止源通道可以通过 OBM 工作模式控制块选择信号来产生中止或调制信号 **APB_OBMn_SW**。

用户可在 **APB_OBMn_BKSm** 寄存器为每个 OBM 中止通道选择中止信号源。信号反相控制位 **APB_OBMn_BKNm** 用于每个中止通道的反相。(n = OBM 组号; m=OBM 通道标号)

状态位 **APB_OBMn_SW** 反映 OBMn 中止切换信号的实时状态。

下面的表格展示了通过寄存器设置对这三个输出通道 MUX 的源信号选择。

表 14-4. OBM 块中止通道信号表

芯片	MG32F02A132/072			MG32F02A032		
通道	通道-0	通道-1	通道-2	通道-0	通道-1	通道-2
	APB 寄存器			APB 寄存器		
OBM 中止源	OBMn_BKS0	OBMn_BKS1	OBMn_BKS2	OBMn_BKS0	OBMn_BKS1	OBMn_BKS2
BK0 信号	1	1	1	1	1	1
BK1 信号	-	-	-	INT_PA	INT_PB	-
BK2 信号	-	-	-	INT_PC	INT_PD	-
BK3 信号	ADC0_OUT	-	SPI0_MOSI	ADC0_OUT	INT_BOD1	SPI0_MOSI
BK4 信号	TM00_TRGO	TM01_TRGO	-	TM00_TRGO	TM01_TRGO	-
BK5 信号	TM10_TRGO	-	-	TM10_TRGO	TM16_TRGO	-
BK6 信号	-	TM26_TRGO	TM36_TRGO	-	-	TM36_TRGO
BK7 信号	-	-	-	-	-	-
BK8 信号	TM20_OC00	TM20_OC10	-	-	-	-
BK9 信号	TM36_OC2	TM36_OC3	-	TM36_OC2	TM36_OC3	-
BK10 信号	CMP0_OUT	CMP1_OUT	CMP2_OUT	CMP0_OUT	CMP1_OUT	-
BK11 信号	-	-	CMP3_OUT	-	-	-
BK12 信号	URT0_TX	URT1_TX	URT2_BRO	URT0_TX	URT1_TX	-
BK13 信号	URT2_TX	URT3_TX	URT2_TMO	-	-	-
BK14 信号	URT0_RX	URT1_RX	URT3_BRO	URT0_RX	URT1_RX	-
BK15 信号	URT2_RX	URT3_RX	URT3_TMO	-	-	-

注释-1：“-” = 保留，“1” = 逻辑-1，SPI0_MOSI = SPI 输出信号

14.6.3. OBM 工作模式

OBM 工作模式块支持 AND, CLR/SET/TOGGLE 工作模式。用户可通过 **APB_OBMn_MDS** 寄存器选择 OBM 工作模式控制中止或调制信号 **APB_OBMn_SW**。

当选择 AND，**APB_OBMn_SW** 信号直接被所有的中止通道输出的 AND 信号控制。当选择 CLR/SET/TOGGLE，**APB_OBM0_SW** 信号会被 STA(**APB_OBMn_STA**)位控制，且可被固件更新。**APB_OBMn_STA** 位用于设置 OBMn 中止切换信号初始状态。只有同时向 **APB_OBM0_LCK** 写 1，该位才能有效地被写入。

14.7. IR 控制

APB 模块包含 1 个 IR (远程红外)调制块。该模块用于红外控制传输的信号调制。

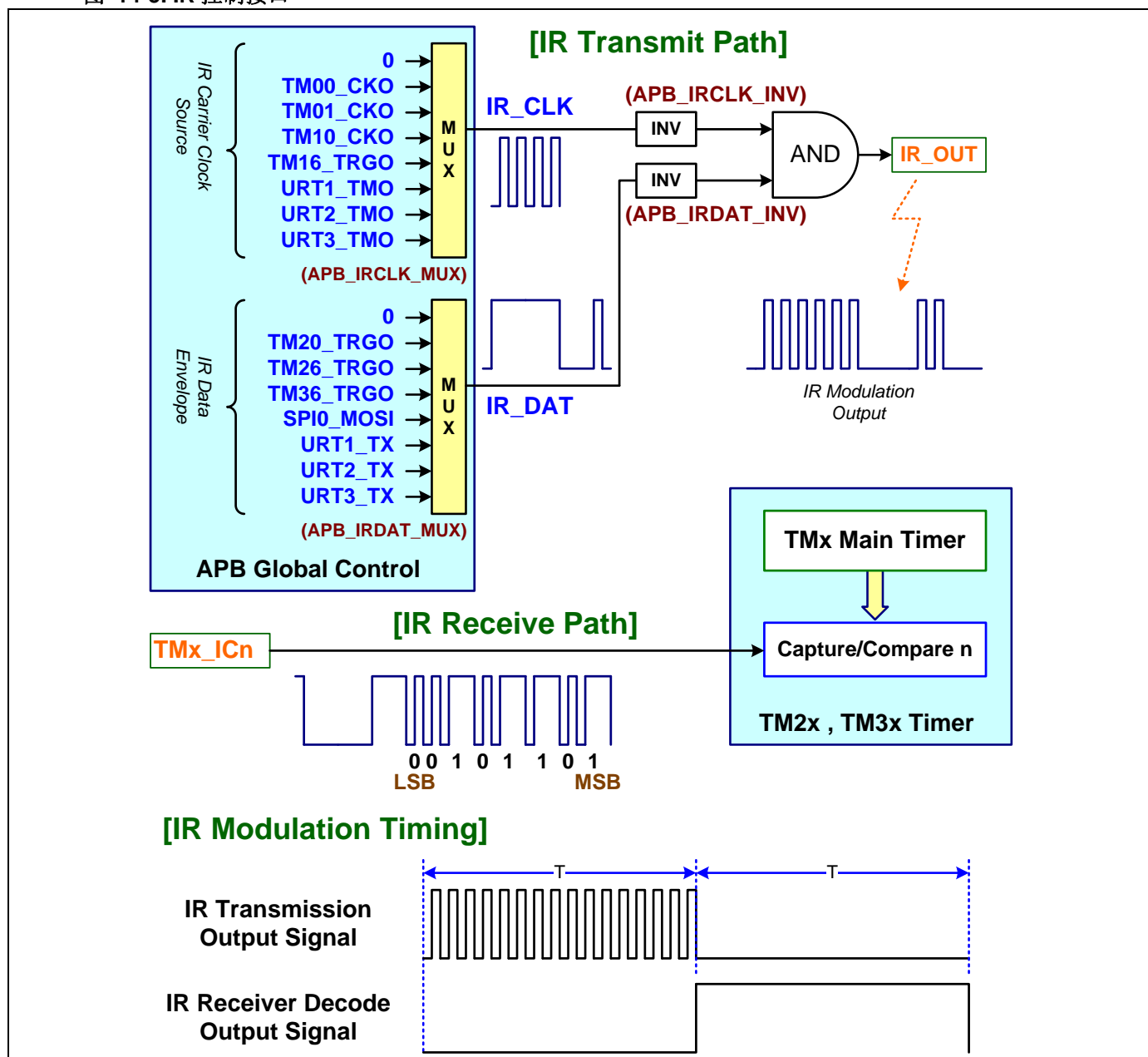
14.7.1. IR 控制接口

用于 IR 传输, IR 调制块可输出 IR 调制信号到 **IR_OUT** 输出。用于 IR 接收, 用户可输入来自 **TMx_ICn** 的 IR 数字信号到定时器模块, 并使用定时器捕获功能记录 IR 信息, 并为固件所用。

下面的图表展示了 IR 发送和接收控制接口。

[注释]: TM20_TRGO, TM26_TRGO, URT2_TX, URT3_TX, URT2_TMO 和 URT3_TMO 不支持于 MG32F02A032。

图 14-5. IR 控制接口



14.7.2. IR 发送调制

用户可通过 **APB_IRCLK_MUX** 寄存器选择 IR 发送调制时钟信号，也可通过 **APB_IRDAT_MUX** 寄存器设置调制数据信号。有 2 个反相位 **APB_IRCLK_INV** 和 **APB_IRDAT_INV** 分别用于时钟信号和数据信号的反相。

下面的表格展示了通过寄存器设置 IR 时钟/数据信号的源信号选择。

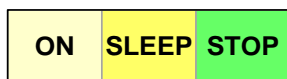
表 14-5. IR 时钟/数据信号表

芯片	MG32F02A132/072		MG32F02A032	
通道	载波时钟	数据包	载波时钟	数据包
	APB 寄存器		APB 寄存器	
时钟/数据源	IRCLK_MUX	IRDAT_MUX	IRCLK_MUX	IRDAT_MUX
CLK0/DAT0 信号	0	0	0	0
CLK1/DAT1 信号	TM00_CKO	TM20_TRGO	TM00_CKO	-
CLK2/DAT2 信号	TM01_CKO	TM26_TRGO	TM01_CKO	-
CLK3/DAT3 信号	TM10_CKO	TM36_TRGO	TM10_CKO	TM36_TRGO
CLK4/DAT4 信号	TM16_TRGO	SPI0_MOSI	TM16_TRGO	SPI0_MOSI
CLK5/DAT5 信号	URT1_TMO	URT1_TX	URT1_TMO	URT1_TX
CLK6/DAT6 信号	URT2_TMO	URT2_TX	-	-
CLK7/DAT7 信号	URT3_TMO	URT3_TX	-	-

注释-1：“0” = 逻辑-0，SPI0_MOSI = SPI 输出信号

15. I2C (集成电路总线)

15.1. 简介



The module can be running in all power operation modes.

I2C 接口是双线双向串行总线。它非常适合于典型的微控制器应用。I2C 协议允许系统设计者仅使用 2 个双向总线线路，一个用于时钟（SCL）和一个用于数据（SDA）来互连 128 个不同的设备。I2C 总线提供对 SDA，SCL 生成和同步、仲裁逻辑和 START/STOP 控制和生成。实现此总线所需的唯一外部硬件是在每条 I2C 总线上的 1 个上拉电阻。连接到总线的所有设备都有单独的地址，并且在 I2C 协议中固有解决总线争用的机制。

I2C 模块内建阴影缓冲区和数据寄存器，以提高发送和接收通信性能。

注释：（x = 模块标号）会被用于该章的寄存器、信号和引脚/端口描述中。[EX]: I2Cx_EN ~ x 表示模块标号 0,1。

15.2. 特性

- I2C 模块一般功能
 - 支持主机和从机模式
 - 支持可设置时钟速率控制
 - 支持可设置的主机模式高/低周期控制
 - 支持从机模式的时钟拉伸模式
 - 内建拉高预驱动控制电路
 - 生成和检测 7 位地址
 - 支持通用响应功能
 - 支持多主机处理
 - 支持使用两组地址的多从机地址解码
 - 支持总线错误、非法的 No-ACK、数据溢出和多主机仲裁错误检测
 - 支持字节模式和缓冲模式流控制
 - 支持字节模式总线事件码用于固件控制
 - 支持缓冲模式 4 字节数据缓冲和 32 位数据寄存器用于高速通讯
 - 可使用 DMA 进行数据接收和发送
 - 支持 SMBus 超时检测
 - 支持缓冲模式监视功能

❖ MG32F02A132/072

- 提供 2 个完全相同的 I2C 模块：I2C0 , I2C1

❖ MG32F02A032

- 提供 1 个 I2C 模块：I2C0
- 支持从机地址硬件检测从 STOP 模式唤醒

15.3. 配置

15.3.1. 芯片配置

下面的表格展示了芯片 I2C 配置。

表 15-1. I2C 配置

芯片	I2C 模块	
	I2C0	I2C1
MG32F02A132	V	V
MG32F02A072	V	V
MG32F02A032	V	

注释 V:包含

15.3.2. 模块功能

下面的表格展示了 I2C 模块包含的功能。

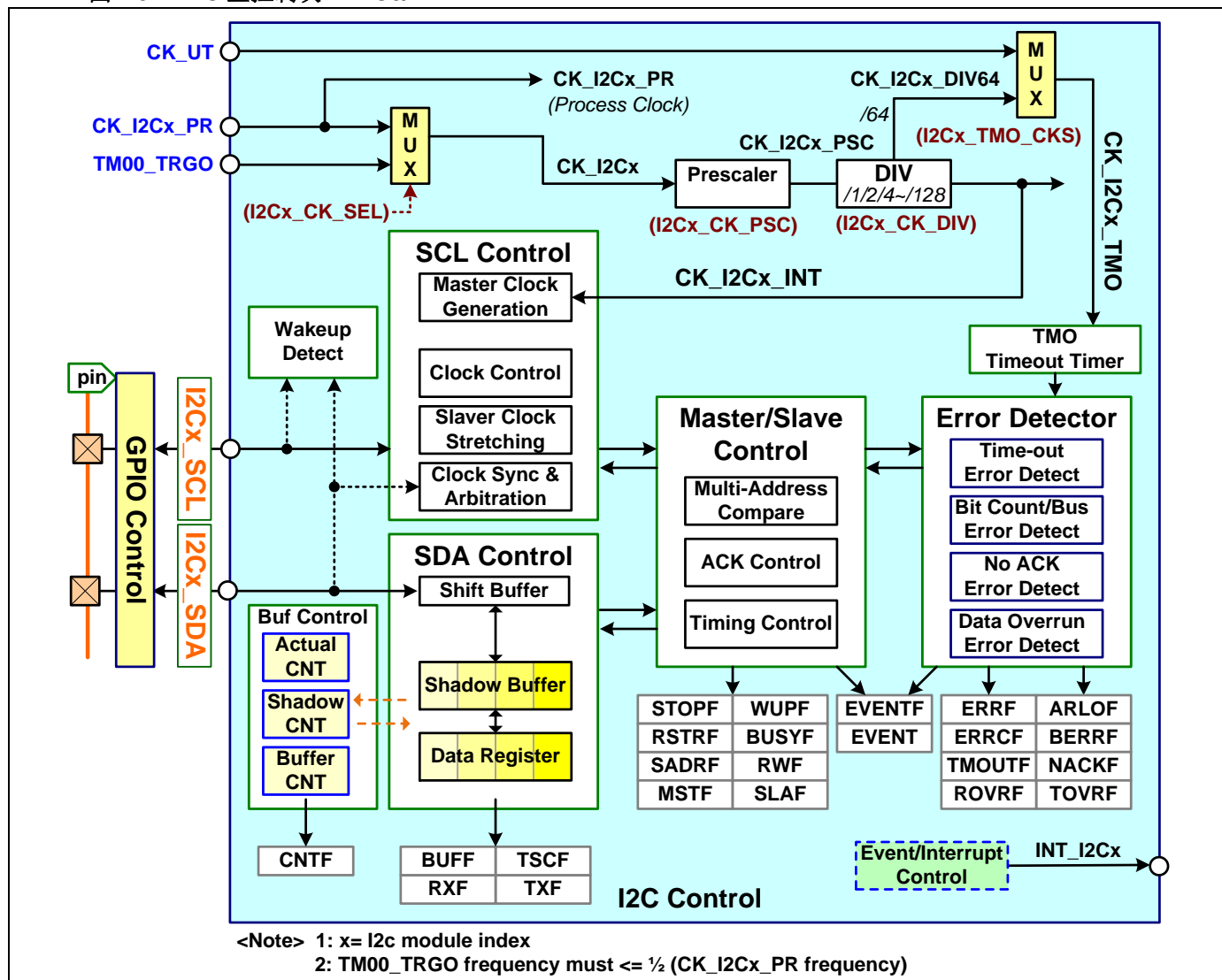
表 15-2. I2C 模块功能

模块功能	MG32F02A132/072		MG32F02A032	注释
	I2C0	I2C1	I2C0	
主机模式	yes	yes	yes	
从机模式	yes	yes	yes	
从机监视模式	yes	yes	yes	只可用于缓冲模式；不可驱动 SCL/SDA(SCL 拉伸禁用)，no-ACK(ACK 禁用)，从机地址/数据（只可 RX）记录，中断 - SADR/STOPF/RSTRF/ERRF
多主机	yes	yes	yes	只可用于缓冲模式
通用响应	yes	yes	yes	指令支持- 复位、写可设置的从机地址、主机地址解码
多从机地址	2 组	2 组	2 组	
数据字节模式	yes	yes	yes	8 位转移缓冲+8 位数据寄存器；软件流控制模式
数据缓冲模式	yes	yes	yes	8 位转移缓冲+32 位阴影缓冲+32 位数据寄存器
数据寄存器	4 字节	4 字节	4 字节	软件用的数据寄存器
阴影缓冲	4 字节	4 字节	4 字节	内部数据控制缓冲
Standard/Fast 模式	yes	yes	yes	
Fast mode plus 模式 (1M/s)	yes	yes	yes	支持用于缓冲模式和 DMA 模式
内建高预驱动	yes	yes	yes	硬件支持预驱动 SCL 和 SDA
SCL 拉伸	可设置	可设置	可设置	ACK 周期 SCL 拉伸；硬件高电平检测
地址检测唤醒	-	-	yes	从机地址检测并从 STOP 模式唤醒
可设置 SCL 高/低电平时间	yes	yes	yes	
SCL/SDA 输入滤波	IO 控制	IO 控制	IO 控制	
SCL/SDA 输入施密特触发器	IO 控制	IO 控制	IO 控制	
超时检测	yes	yes	yes	SMBus 超时，检测 SCL 低电平或 SCL/SDA 都为高电平超时
仲裁丢失检测	yes	yes	yes	用于多主机模式
总线错误检测	yes	yes	yes	合法‘启动’或‘停止’前位计数匹配错误；数据在 SCL 高电平时改变
非法 NACK 检测	yes	yes	yes	
数据溢出检测	yes	yes	yes	用于 SCL 时钟拉伸禁用时
DMA 请求	yes	yes	yes	

15.4. 控制块

下面的图表展示了 I2C 控制块。

图 15-1. I2C 主控制块 – I2C0/1



15.5. IO 线

15.5.1. IO 信号

- **I2Cx_SCL**

I2C 时钟 SCL 信号，用于 I2C 主机模式输出或 I2C 从机模式输入。

- **I2Cx_SDA**

I2C 数据 SDA 信号，用于 I2C 主机模式和 I2C 从机模式的双向 IO。

15.5.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。参照用户手册 GPIO 章中“[IO 模式](#)”节以获取更多关于 IO 模式设置的信息。

每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“[功能复用选择](#)”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“[引脚功能复用表](#)”以获取更多信息。

15.6. 使能和时钟

15.6.1. I2C 全局使能

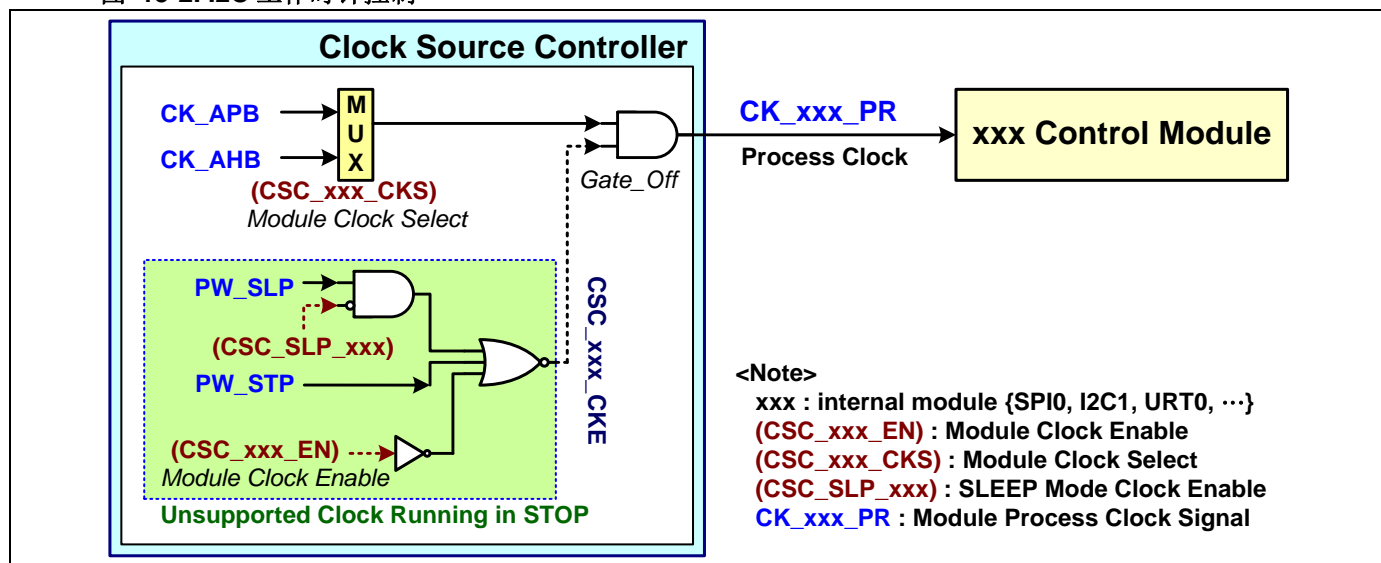
I2C 模块的所有功能全局使能位是 **I2Cx_EN**。当该位被禁用时，所有的 I2C 功能将无法工作。

15.6.2. I2C 时钟控制

● 模块工作时钟

该模块工作时钟 **CK_I2Cx_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 CSC（时钟源控制器）模块。该时钟可通过 **CSC_I2Cx_EN** 寄存器使能并通过 **CSC_I2Cx_CKS** 寄存器选择时钟源来自 APB 或 AHB。用户可以在进入 **SLEEP** 模式之前通过设置 **CSC_SLP_I2Cx** 寄存器规划在 **SLEEP** 模式下是否让 I2C 时钟继续运行。参照系统时钟章以获取更多信息。

图 15-2. I2C 工作时钟控制



● 模块内部时钟

I2C 模块可输出内部时钟 **CK_I2Cx_INT** 作为数据接收信号的采样时钟和发送数据信号的时钟源。用户可通过 **I2Cx_CK_SEL** 寄存器选择时钟源是来自模块工作时钟 **CK_I2Cx_PR** 还是定时器触发输出信号 **TM00_TRGO**。此外，模块提供了 1 个时钟预分频器和 1 个时钟分频器用于产生内部时钟 **CK_I2Cx_INT**。时钟预分频器可通过 **I2Cx_CK_PSC** 寄存器对输入时钟进行 1~8 分频，且寄存器值必须 >0；时钟分频器可通过 **I2Cx_CK_DIV** 寄存器将输入时钟进行 1/2/4/8/~1/128 分频。

[注释]: 输入时钟分频器值 **I2Cx_CK_PSC** 在 MG32F02A132/072 中是 2~8；而 MG32F02A032 则是 2~16。

[注释]: 通常内部时钟频率需比模块工作时钟慢至少 1/2。

TMO 定时器时钟源可通过 **I2Cx_TMO_CKS** 寄存器选择 **CK_UT** 或 **CK_I2Cx_DIV64** 时钟。时钟 **CK_I2Cx_DIV64** 是被固定为 **CK_I2Cx_PSC** 时钟信号的 64 分频的。

15.7. 中断和事件

I2C 模块中有 2 种信号 **INT_I2Cx**, **WUP_I2Cx**。**INT_I2Cx** 发送到外部中断控制器（EXIC）作为中断事件；**WUP_I2Cx** 发送到外部中断控制器（EXIC）作为唤醒事件。

15.7.1. I2C 中断控制和状态

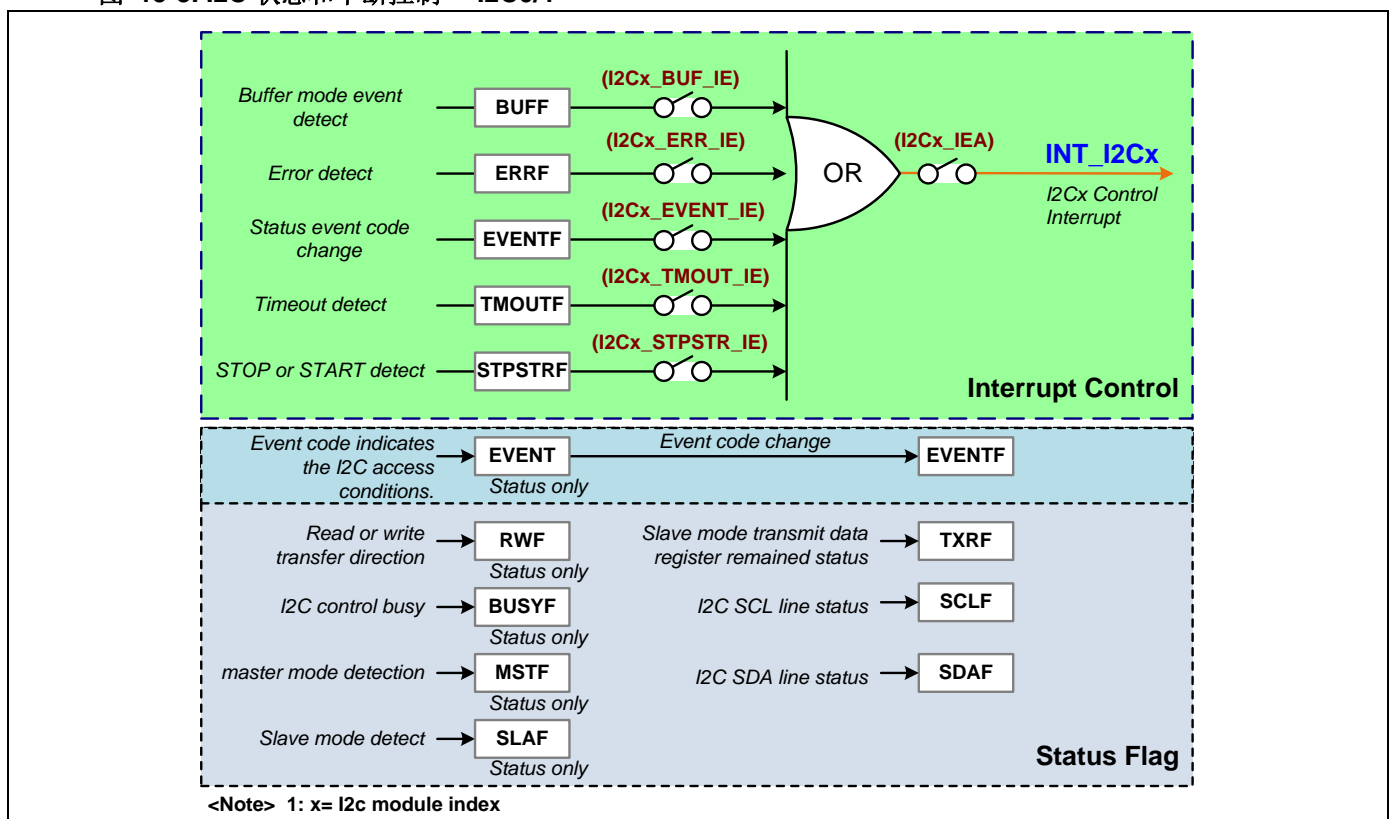
中断标识是用于中断服务程序（ISR）流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **I2Cx_IEA** 用于使能和禁用该模块的所有中断源。

该模块中有一些只读的状态位，用于提供内部控制状态，其中一个标志是占用标志 **I2Cx_BUSYF**，表明数据传输繁忙状态。参照相关状态位寄存器描述以获取更多信息。

在 STOP 模式中，I2C 控制模块可检测 **I2Cx_SADR** 或 **I2Cx_SADR2** 寄存器中设置的 I2C 从机地址。若 **I2Cx_WUP_IE** 和 **I2Cx_IEA** 寄存器被使能，当拥有的 I2C 从机地址被检测到时，芯片会被唤醒。

[注释]: STOP 模式唤醒功能不支持于 MG32F02A132/072。

图 15-3. I2C 状态和中断控制 – I2C0/1



[注释]: **I2Cx_STPSTR IE** 不支持于 MG32F02A132/072。

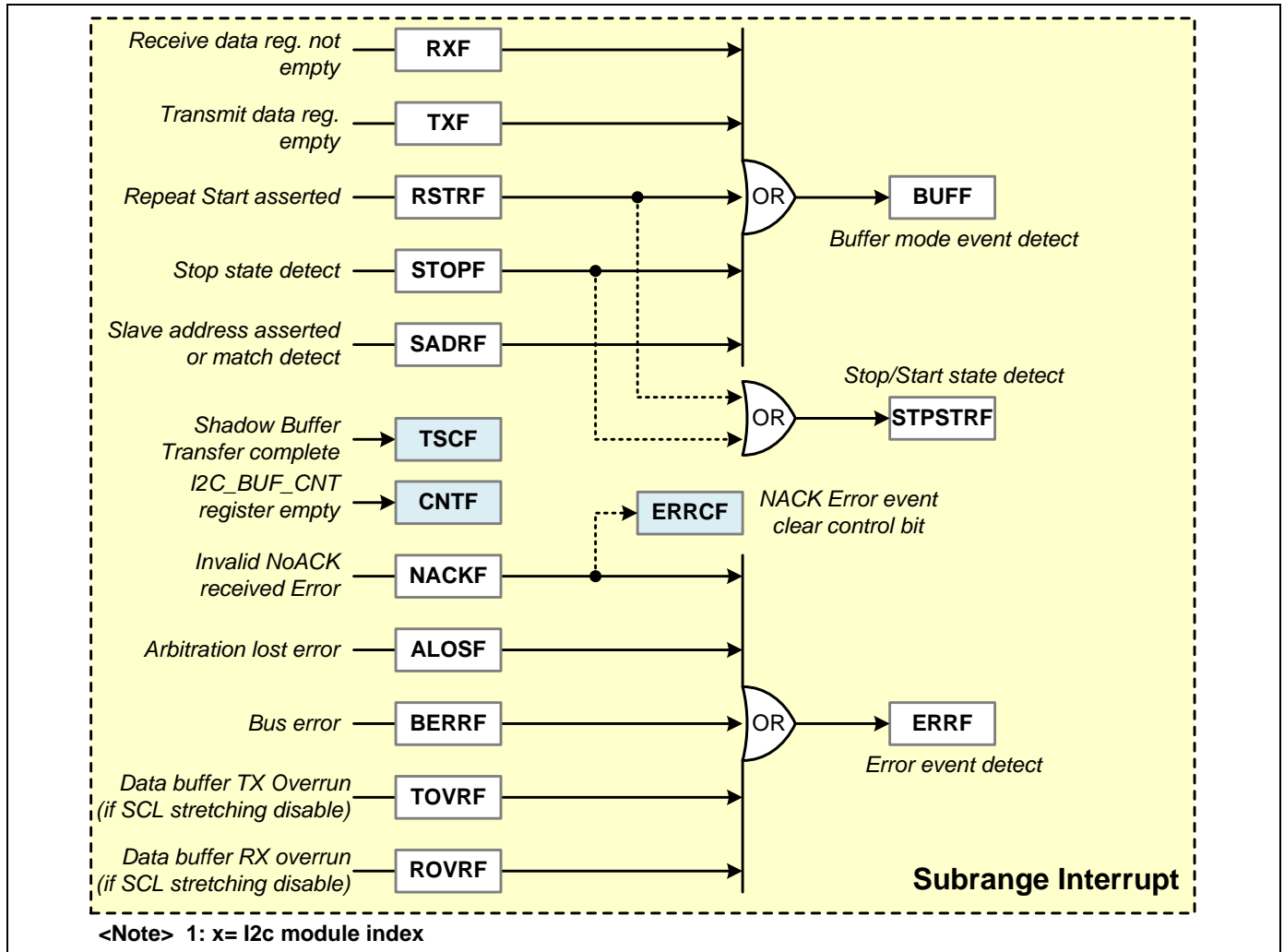
15.7.2. I2C 子范围中断

在中断标志中有一些子范围中断标志 **I2Cx_BUFF**, **I2Cx_ERRF** 和 **I2Cx_STPSTRF**。这些子范围中断标志可被硬件置起，并引起置起更高级的 **I2Cx_BUFF**, **I2Cx_ERRF** 和 **I2Cx_STPSTRF** 标志以产生中断。用户可使用这些高级的标志用于中断流控制，且没有任何硬件事件控制。

I2Cx_TSCF, **I2Cx_CNTF** 和 **I2Cx_ERRCF** 标志只能通过硬件置起，用于固件控制，他们不作为中断事件。

下面的图表展示了 I2C 子范围中断控制块。

图 15-4. I2C 子范围中断控制 – I2C0/1



15.7.3. I2C 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

● EVENTF

I2C 状态事件中断标志是(**I2Cx_EVENTF**)，相关的中断使能寄存器位是 **I2Cx_EVENT_IE**。该标志用于 I2C 字节模式中中断控制。每当事件码寄存器的 **I2Cx_EVENT** 被硬件更新时，该标志被置起。与 **I2Cx_EVENTF** 标志一模一样的标志 **I2Cx_EVENTF2** 被和 I2C_EVENT (事件码值)一起设计在相同的 32 位寄存器中为固件方便使用。

● BUFF

I2C 缓冲模式事件标志是(**I2Cx_BUFF**)，相关的中断使能寄存器位是 **I2Cx_BUFF_IE**。该标志用于 I2C 缓冲模式中中断控制。若 **I2Cx_BUFF_IE** 位被使能，每当 RXF, TXF, RSTRF, STOPF 或 SADRf 中的任何标志被置起，该标志会置起并产生中断事件。

— RXF

I2C 接收数据寄存器不为空标志是(**I2Cx_RXF**)，这是 BUFF 中断标志的范围中断标志。当阴影缓冲接收到数据，硬件会复制阴影缓冲数据到数据寄存器 **I2Cx_DAT**，并置起该标志。当 **I2Cx_DAT** 被读或该位被软件写 1 时清除。

注释: 当 SWD 调试时，I2Cx_DAT 被读取，RXF 标志也不会被清除。

— TXF

I2C 发送数据寄存器为空标志是(**I2Cx_TXF**)，这是 BUFF 中断标志的范围中断标志。当发送过程中阴影缓冲为空时，数据寄存器 **I2Cx_DAT** 会复制到阴影缓冲并置起该标志。当 **I2Cx_DAT** 被写或该位被软件写 1 时清除。

— SADRf

I2C 从机地址匹配标志是(**I2Cx_SADRf**)，这是 BUFF 中断标志的范围中断标志。该标志也会在主机模式下在发送模式下从机地址不匹配或接收模式下从机地址置起时被置起。当从 **STOP** 模式通过检测匹配从机地址唤醒时，

用户需清除该位以禁用时钟拉伸并释放外部主机的时钟信号。

[注释: **I2Cx_STPSTR IE** 不支持于 MG32F02A132/072。]

— RSTRF

I2C 重复启动置起标志(**I2Cx_RSTRF**)，这是 BUFF 中断标志的范围中断标志。当硬件检测到 I2C 重复启动状态，该位被置起。

— STOPF

I2C 停止检测标志是 (**I2Cx_STOPF**)，这是 BUFF 中断标志的范围中断标志。当硬件检测到 I2C 停止状态，该位被置起。

● ERRF

I2C 错误中断标志是(**I2Cx_ERRF**)，相关的中断使能寄存器位是 **I2Cx_ERR_IE**。它表明任何非法的 N-ACK、总线仲裁丢失、总线错误或数据溢出错误的发生。若 **I2Cx_ERR_IE** 位被使能，每当 RXOVRF, TXOVRF, BERRF, ALOSF 或 NACKF 中的任何标志被置起，该标志会置起并产生中断事件。

— ROVRF

I2C 接收溢出标志 (**I2Cx_ROVRF**)，这是 ERRF 中断标志的范围中断标志。当接收溢出时，硬件将停止接收下一个数据到数据阴影缓冲中，直到清除此标志。

— TOVRF

I2C 发送下溢标志(**I2Cx_TXOVRF**)，这是 ERRF 中断标志的范围中断标志。当发送下溢时，硬件将发生 0xFF 作为下一个数据到数据阴影缓冲中，直到数据缓冲不为空。

— NACKF

I2C “不承认”接收错误标志(**I2Cx_NACKF**)，这是 ERRF 中断标志的范围中断标志。

— ALOSF

I2C 总线仲裁丢失错误标志(**I2Cx_ALOSF**)，这是 ERRF 中断标志的范围中断标志。

— BERRF

I2C 非法停止/启动状态的总线错误标志(**I2Cx_BERRF**)，这是 ERRF 中断标志的范围中断标志。

● TMOUTF

I2C 总线超时标志(**I2Cx_TMOUTF**)，相关的中断使能寄存器位是 **I2Cx_TMOUT_IE**。

● WUPF

STOP 模式通过 I2C 事件检测唤醒标志(**I2Cx_WUPF**)，相关的中断使能寄存器位是 **I2Cx_WUP_IE**。当 **STOP** 模式下硬件检测到从机地址匹配到 **I2Cx_SADR** (**I2Cx_SADR_EN=1**)或 **I2Cx_SADR2** (**I2Cx_SADR2_EN=1**)寄存器的设置时置起。

● STPSTRF

I2C 停止或启动检测标志(**I2Cx_STPSTRF**)，相关的中断使能寄存器位是 **I2Cx_STPSTR_IE**。若 **I2Cx_STPIR_IE** 位被使能，每当 RSTRF 或 STOPF 中的任何标志被置起，该标志会置起并产生中断事件。

15.7.4. I2C 控制标志

I2Cx_TSCF, **I2Cx_CNTF** 和 **I2Cx_ERRCF** 标志只能通过芯片置起，用于固件控制，他们不作为中断事件。

● TSCF

I2C 阴影缓冲发送完成标志 (**I2Cx_TSCF**)，该标志被硬件置起，被软件写 1 清除。该标志用于内部硬件控制。

● CNTF

I2C 缓冲计数寄存器 **I2Cx_BUF_CNT** 空状态(**I2Cx_CNTF**)，该标志被硬件置起，被软件写 1 清除。

● ERRCF

I2C 主机模式 NACK 错误标志和状态控制位(**I2Cx_ERRCF**)，该标志被硬件置起，被软件写 1 或在启动/停止状态时自动清除。该位会在从机地址周期或数据接收周期发生 NACK 时置起。

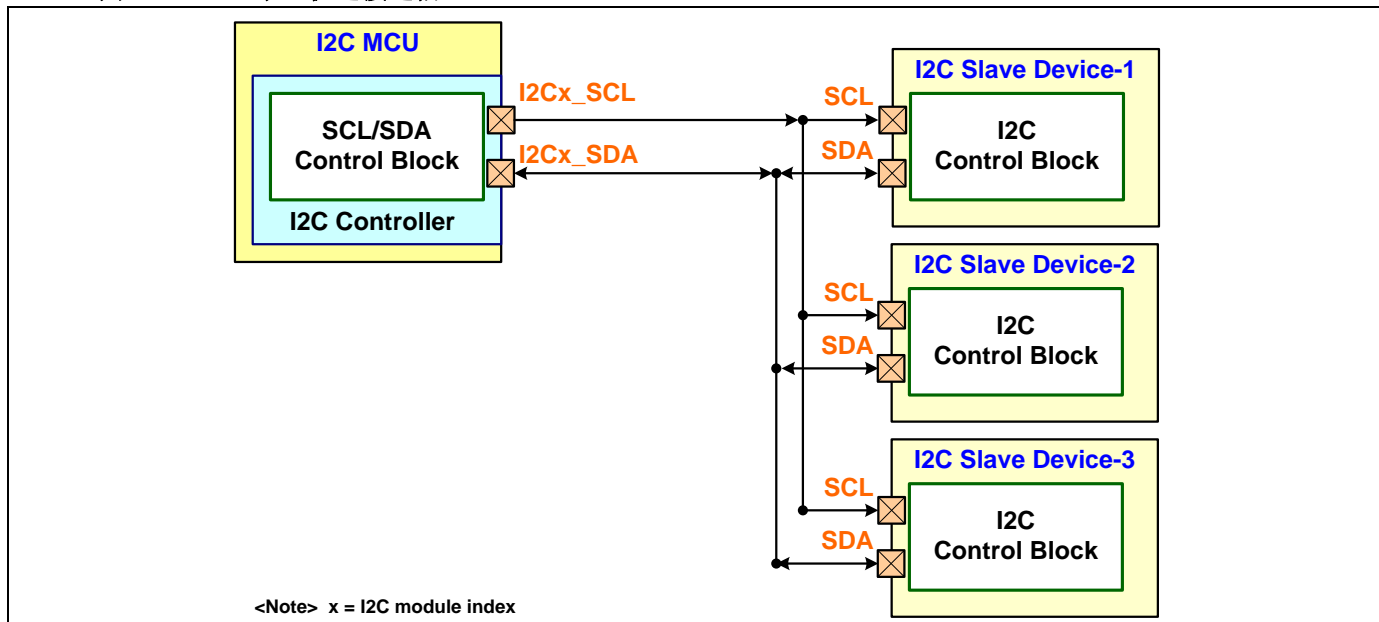
15.8. I2C 连接应用

通常用户需要设置 I2C SCL 和 SDA 信号使用的 IO 引脚的 IO 模式为开漏输出。

下面的图表展示了单主机、多主机和从机模式的连接图。在 I2C 模块中有 2 个只读状态位用于直接反映 I2C SCL 和 SDA 的线路状态。用户可通过 **I2Cx_SCLF** 和 **I2Cx_SDLF** 寄存器读取这两个状态位。

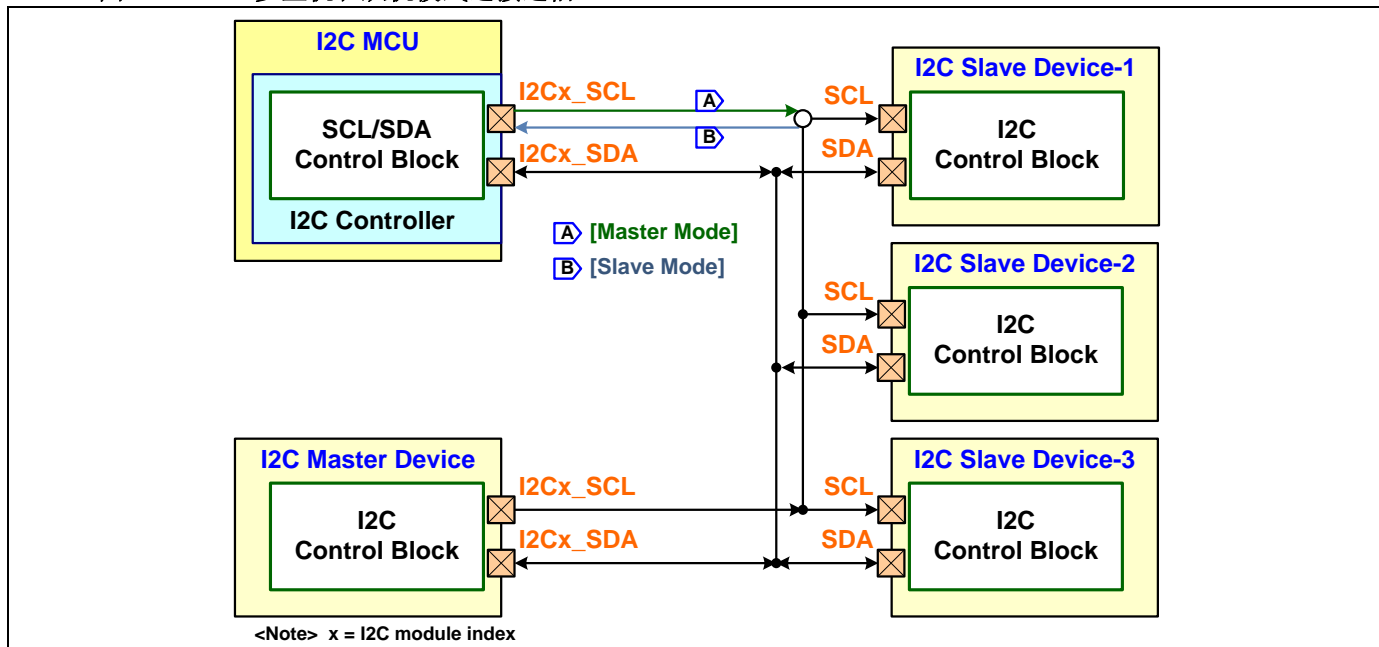
● I2C 单主机连接通信

图 15-5. I2C 单主机连接通信



● I2C 多主机和从机模式连接通信

图 15-6. I2C 多主机和从机模式连接通信



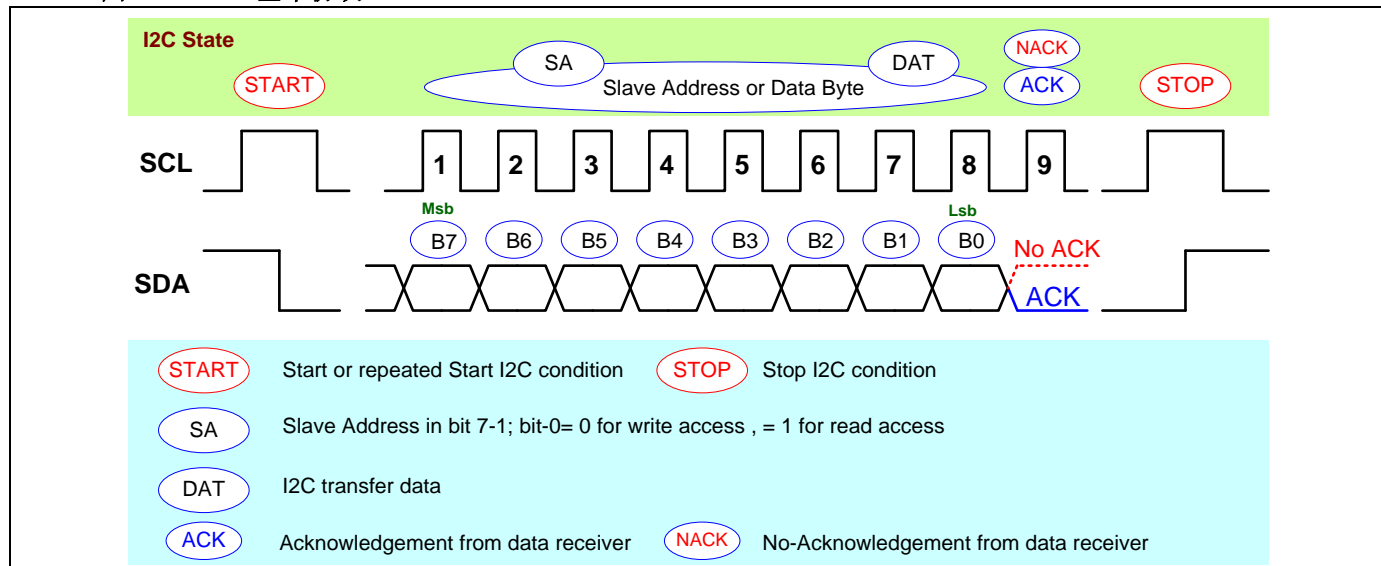
15.9. I2C 基本控制

15.9.1. I2C 基本协议

I2C 控制器的基本功能基于标准 I2C 通信协议设计。参照“I2C 总线规格和用户手册”以获取更多关于 I2C 协议的信息。

下面的图表展示了 I2C 基本协议。

图 15-7. I2C 基本协议



15.9.2. I2C 控制模式

I2C 支持通过 **I2Cx_MDS** 寄存器设置单主机、多主机、从机和监视工作模式。多主机模式仅支持字节模式，监视模式有点类似从机模式，但是不会置起 ACK，这只支持于缓冲模式。

I2C 支持通过 **I2Cx_BUF_EN** 寄存器设置可选的字节和缓冲数据控制模式。当选择字节模式，固件可简单的使用事件标志 **EVENTF** (**I2Cx_EVENTF**) 和事件码寄存器的 **I2Cx_EVENT** 来进行数据发送流控制；当使能缓冲模式，1 个阴影缓冲会用于加速数据流控制。RXF 和 TXF 标志用于标识数据寄存器接收不为空或发送为空。

15.9.3. I2C 从机地址

I2C 控制器支持 2 组从机地址用于从机模式地址检测。用户可通过 **I2Cx_SADR** 和 **I2Cx_SADR2** 寄存器设置这两个地址，并通过 **I2Cx_SADR_EN** 和 **I2Cx_SADR_EN2** 寄存器为它们独立使能。对于 **I2Cx_SADR** 寄存器里的 I2C 从机地址，屏蔽寄存器 **I2Cx_SA_MSK** 被用于为 I2C 从机模式提供设置和屏蔽地址位，寄存器零位使得相关地址位在比较时被认为是“不影响”，该屏蔽寄存器在寄存器中无效，此外，I2C 控制器还支持检测 I2C 通用响应地址 **0x00**，并可独立的通过 **I2Cx_GC_EN** 寄存器使能。

[注释]: **I2Cx_SA_MSK** 寄存器不支持于 MG32F02A132/072。

用户可通过 **I2Cx_SA_CODE** 寄存器在 I2C 从机模式下获取检测的从机地址码。在从机模式工作下，I2C 控制器会始终把从机地址码抓入该寄存器。

15.9.4. I2C 数据寄存器和移动缓冲

数据寄存器 **I2Cx_DAT** 被设计为 8/16/32 位数据访问。当使能缓冲模式时，读该数据寄存器会清除 RXF 标志，而写该数据寄存器将清除 TXF 标志。

I2C 控制器包含 1 个 8 位移动缓冲用于数据收发。寄存器 **I2Cx_SBUF** 为可读，并可用于获取移动缓冲中收发数据的实时数据位。

15.9.5. I2C 访问指令

I2C 数据收发中，一共有 3 个指令寄存器位 STA/STO/AA 用于控制 I2C 启动、停止和承认状态。这些位处于 **I2Cx_STA**、**I2Cx_STO** 和 **I2Cx_AA** 寄存器中。特别的是，当保护位 **I2Cx_STA_LCK** 被同时写 1 时，寄存器位 **I2Cx_STA** 为只写权限此外，**I2Cx_STO** 和 **I2Cx_AA** 寄存器位也有相同的保护功能和独立地寄存器位 **I2Cx_STO_LCK**、**I2Cx_AA_LCK**。

● STA - I2C START 使能位

当 STA 位被置起进入主机模式，I2C 硬件会检测该串行总线状态并在该总线空闲时产生 START 状态。若总线不为空闲状态，I2C 会等待 STOP 状态并在稍微的延时后产生 START 状态。若 I2C 已处于主机模式且有 1 个或多个字节已被发送或接收时 STA 被置起，I2C 会发送 1 个重复 START 状态。STA 可在任何时间被置起。STA 也可在 I2C 为从机地址时被置起。当 STA 被复位，START 状态和重复 START 状态将不会被产生。

● STO - I2C STOP 使能位

当 I2C 进入主机模式且 STO 位被置起时，STOP 状态会被发送到串行总线上。当 STOP 状态在总线上被检测到时，I2C 硬件会清除 STO 标志。从机模式下，STO 标志可被置起去把总线从错误状态恢复，这种情况下，STOP 状态将不会被发送到总线上。然而，I2C 硬件表现的好像已经收到了 STOP 状态并且换到了定义的未寻址从机接收模式。STO 标志会自动被硬件清除。若 STA 和 STO 位都被置起，且 I2C 处于主机模式（从机模式下，I2C 产生不是发送的内部 STOP 状态）STOP 状态会被发送到总线上，然后发送 START 状态。

● AA - I2C 置起承认使能位

若 AA 位被置 1, 当以下情况发生时，SCL 线上的 ACK 时钟脉冲中会返回 1 个 ACK:

- (1) 拥有的从机地址已接收
- (2) 当 I2C 处于主机/接收模式时收到数据字节
- (3) 当 I2C 处于从机/接收模式时收到数据字节

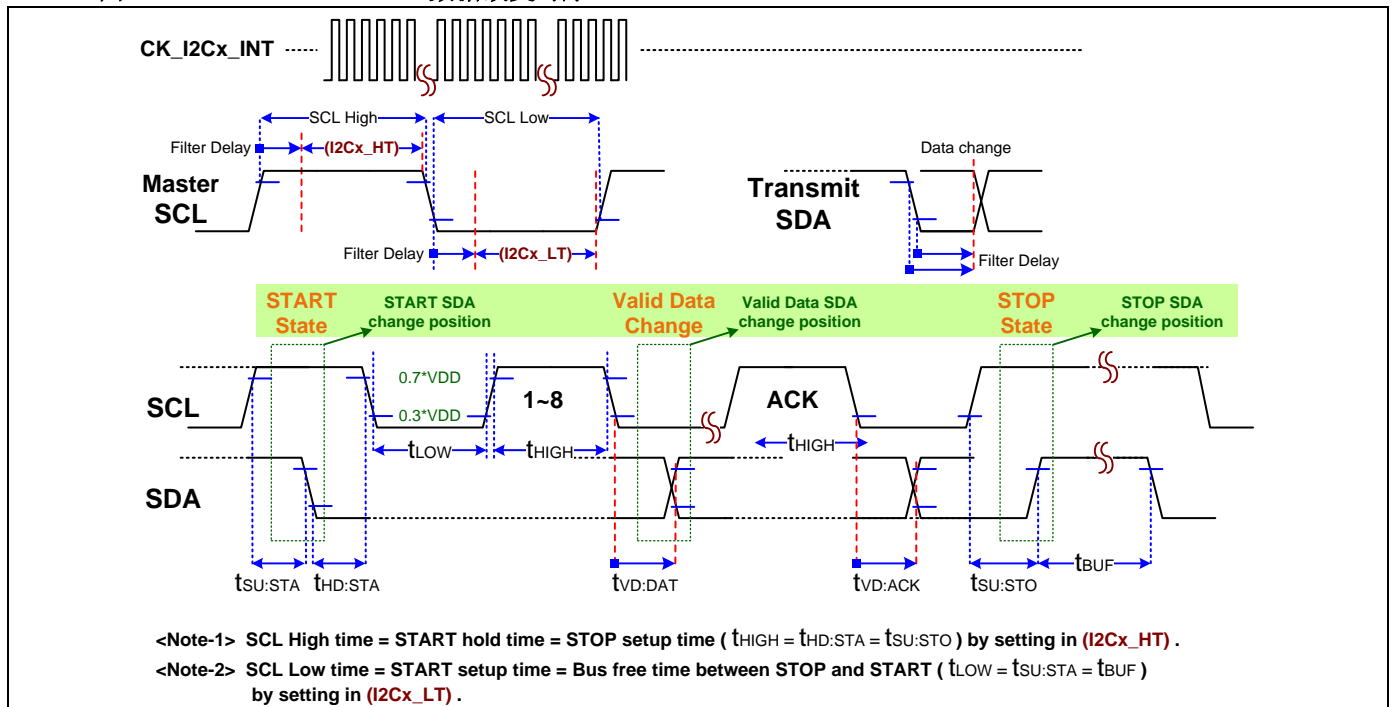
若 AA 位被置 0, 当以下情况发生时，SCL 线上的 ACK 时钟脉冲中会返回 1 个 NACK:

- (1) 当 I2C 处于主机/接收模式时收到数据
- (2) 当 I2C 处于从机/接收模式时收到数据字节

15.9.6. I2C START/STOP/数据改变

I2Cx_HT 和 **I2Cx_LT** 寄存器用于设置 I2C 主机模式时序。用户可通过 **I2Cx_HT** 寄存器设置 SCL 高电平时间，通过 **I2Cx_LT** 寄存器设置 SCL 低电平时间。

图 15-8. I2C START/STOP/数据改变时序



15.10. I2C 字节模式控制

用户可简单地使用事件标志 **EVENTF (I2Cx_EVENTF)** 和事件码寄存器 **I2Cx_EVENT** 用于数据发送流控制。

15.10.1. I2C 字节模式收发

EVENTF 标志会在每次硬件检测到新的 I2C 状态或错误事件时置起。此时，事件码寄存器的 **I2Cx_EVENT** 也会被更新，若中断使能位 **I2Cx_EVENT_IE** 被使能，I2C 控制器还会产生中断事件。用户可通过读 **I2Cx_EVENT** 寄存器以获取事件码得知检测到的 I2C 事件。

用户可在固件中使用中断事件触发实现 I2C 通信，并通过读 I2C 事件码以获取即时的 I2C 状态。用户可在 **I2Cx_STA**, **I2Cx_STO** 和 **I2Cx_AA** 寄存器中设置 I2C 访问指令 **STA/STO/AA**，用于 I2C 通信控制。

参照“[I2C 访问指令](#)”节以获取更多关于 I2C 访问指令的信息。

15.10.2. I2C 事件码

下面的表格展示了 I2C 事件码的摘要、主机和从机模式。

● I2C 事件码摘要

表 15-3. I2C 事件码表

事件码	硬件和总线状态	主机/从机模式			
		主机发送	主机接收	从机接收	从机发送
0x00	总线错误（由于非法的启动或停止状态）	V	V	V	V
0x08	START 状态发送	V	V		
0x10	重复 START 发送	V	V		
0x18	SLA+W 发送和 ACK 接收	V			
0x20	SLA+W 发送和 NACK 接收	V			
0x28	DAT 发送和 ACK 接收	V			
0x30	DAT 发送和 NACK 接收	V			
0x38	SLA+W 或 DAT 仲裁丢失	V			
	SLA+R 或 DAT 仲裁丢失		V		
	NACK 位仲裁丢失		V		
0x40	SLA+R 发送和 ACK 接收		V		
0x48	SLA+R 发送和 NACK 接收		V		
0x50	DAT 接收和 ACK 接收		V		
0x58	DAT 接收和 NACK 接收		V		
0x60	拥有的 SLA+W 接收和 ACK 返回			V	
0x68	拥有的 SLA+W 接收，仲裁丢失和 ACK 返回			V	
0x70	通用响应地址接收和 ACK 返回			V	
0x78	通用响应地址接收和仲裁丢失			V	
0x80	之前寻址于自己的 SLA，DAT 接收和 ACK 返回			V	
0x88	之前寻址于自己的 SLA，DAT 接收和 NoACK 返回			V	
0x90	DAT 接收和 ACK 返回(通用响应)			V	
0x98	DAT 接收和 NoACK 返回(通用响应)			V	
0xA0	收到 STOP 或重复 START			V	
0xA8	拥有的 SLA+R 接收和 ACK 返回				V
0xB0	拥有的 SLA+R 接收，仲裁丢失和 ACK 返回				V
0xB8	DAT 发送和 ACK 接收				V
0xC0	DAT 发送和 NoACK 接收				V
0xC8	上一次 DAT 发送和 ACK 接收				V
0xF8	STOP 或总线被释放；没有可用的相关状态信息 (EVENTF = 0 且无中断发生)	V	V	V	V

DAT = I2Cx_DAT 数据寄存器 (x:模块标号)
SLA+W/R = 从机地址+写/读指令位

● I2C 主机发送模式事件表

表 15-4. I2C 主机发送模式事件表

事件 码	硬件和总线状态	到/来自 I2Cx_DAT	STA	STO	AA	硬件下一步动作
0x08	START 状态已被发送	载入 SLA+W	X	0	X	SLA+W 会被发送; ACK 位会被接收
0x10	重复 START 状态已被发送	载入 SLA+W	X	0	X	SLA+W 会被发送; ACK 位会被接收
		载入 SLA+R; 清除 STA	X	0	X	SLA+W 会被发送; I2C 块会被切换到主机接收模式
0x18	SLA+W 已发送; ACK 已接收	载入数据字节	0	0	X	数据字节会被发送; ACK 位会被接收
		无 DAT 动作	1	0	X	重复 START 会被发送
		无 DAT 动作	0	1	X	STOP 状态会被发送; STO 标志会被复位
		无 DAT 动作	1	1	X	STOP 状态会跟着 START 状态发送, STO 标志会被复位
0x20	SLA+W 已发送;NOT ACK 已接收	载入数据字节	0	0	X	数据字节会被发送; ACK 位会被接收
		无 DAT 动作	1	0	X	重复 START 会被发送
		无 DAT 动作	0	1	X	STOP 状态会被发送; STO 标志会被复位
		无 DAT 动作	1	1	X	STOP 状态会跟着 START 状态发送, STO 标志会被复位
0x28	DAT 中数据字节已发送; ACK 已接收	载入数据字节	0	0	X	数据字节会被发送; ACK 位会被接收
		无 DAT 动作	1	0	X	重复 START 会被发送
		无 DAT 动作	0	1	X	STOP 状态会被发送; STO 标志会被复位
		无 DAT 动作	1	1	X	STOP 状态会跟着 START 状态发送, STO 标志会被复位
0x30	DAT 中数据字节已发送; NOT ACK 已接收	载入数据字节	0	0	X	数据字节会被发送; ACK 位会被接收
		无 DAT 动作	1	0	X	重复 START 会被发送
		无 DAT 动作	0	1	X	STOP 状态会被发送; STO 标志会被复位
		无 DAT 动作	1	1	X	STOP 状态会跟着 START 状态发送, STO 标志会被复位
0x38	SLA+R/W 或数据字节仲裁丢失	无 DAT 动作	0	0	X	I2C 总线会被释放; 未寻址从机会被进入
		无 DAT 动作	1	0	X	当总线空闲时 START 状态会被发送
DAT = I2Cx_DAT 数据寄存器 (x:模块标号)						
SLA+W/R =从机地址+写/读指令位						

● I2C 主机接收模式事件表

表 15-5. I2C 主机接收模式事件表

事件码	硬件和总线状态	到/来自 I2Cx_DAT	STA	STO	AA	硬件下一步动作
0x08	START 状态已被发送	载入 SLA+R	X	0	X	SLA+R 会被发送; ACK 会被接收
0x10	重复 START 状态已被发送	载入 SLA+R	X	0	X	同上
		载入 SLA+W	X	0	X	SLA+W 会被发送; I2C 块会被切换到主机/TRX 模式
0x38	NOT ACK 位仲裁丢失	无 DAT 动作	0	0	X	I2C 总线会被释放; I2C 块会被切换到从机模式
		无 DAT 动作	1	0	X	当总线空闲时 START 状态会被发送
0x40	SLA+R 已发送; ACK 已接收	无 DAT 动作	0	0	0	数据字节会被接收; NOT ACK 位会被返回
		无 DAT 动作	0	0	1	数据字节会被接收; ACK 位会被返回
0x48	SLA+R 已发送; NOT ACK 已接收	无 DAT 动作	1	0	X	重复 START 会被发送
		无 DAT 动作	0	1	X	STOP 状态会被发送; STO 标志会被复位

		无 DAT 动作	1	1	X	STOP 状态会跟着 START 状态发送, STO 标志会被复位
0x50	DAT 中数据字节已接收; ACK 已返回	读数据字节	0	0	0	数据字节会被接收; NOT ACK 位会被返回
		读数据字节	0	0	1	数据字节会被接收; ACK 位会被返回
0x58	DAT 中数据字节已接收; NOT ACK 已返回	读数据字节	1	0	X	重复 START 会被发送
		读数据字节	0	1	X	STOP 状态会被发送; STO 标志会被复位
		读数据字节	1	1	X	STOP 状态会跟着启动状态发送, STO 标志会被复位

● I2C 从机发送模式事件表

表 15-6. I2C 从机发送模式事件表

事件码	硬件和总线状态	到/来自 I2Cx_DAT	STA	STO	AA	硬件下一步动作
0xA8	拥有的 SLA+R 已接收; ACK 已返回	载入数据字节	X	0	0	最后的数据字节会被发送且 ACK 位会被接收
		载入数据字节	X	0	1	数据字节会被发送; ACK 位会被接收
0xB0	SLA+R/W 为主机的仲裁丢失; 拥有的 SLA+R 已接收; ACK 已返回	载入数据字节	X	0	0	最后的数据字节会被发送且 ACK 位会被接收
		载入数据字节	X	0	1	数据字节会被发送; ACK 位会被接收
0xB8	DAT 中数据字节已发送; ACK 已接收	载入数据字节	X	0	0	最后的数据字节会被发送且 ACK 位会被接收
		载入数据字节	X	0	1	数据字节会被发送; ACK 位会被接收
0xC0	DAT 中数据字节已发送; NOT ACK 已接收	无 DAT 动作	0	0	01	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址
		无 DAT 动作	0	0	1	切换到未寻址从机模式; 识别自己的 SLA 模式; 通用响应地址会在 ADR[0] = 逻辑 1 时被识别
		无 DAT 动作	1	0	0	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址; 当总线空闲时 START 状态会被发送
		无 DAT 动作	1	0	1	切换到未寻址从机模式; 识别自己的 SLA 模式; 通用响应地址会在 ADR[0] = 逻辑 1 时被识别; 当总线空闲时 START 状态会被发送
0xC8	DAT 中上一个数据字节已发送(AA = 0); ACK 已接收	无 DAT 动作	0	0	0	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址
		无 DAT 动作	0	0	1	切换到未寻址从机模式; 识别自己的 SLA 模式; 通用响应地址会在 ADR[0] = 逻辑 1 时被识别
		无 DAT 动作	1	0	0	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址; 当总线空闲时 START 状态会被发送
		无 DAT 动作	1	0	1	切换到未寻址从机模式; 识别自己的 SLA 模式; 通用响应地址会在 ADR[0] = 逻辑 1 时被识别; 当总线空闲时 START 状态会被发送

● I2C 从机接收模式事件表

表 15-7. I2C 从机接收模式事件表

事件 码	硬件和总线状态	到/来自 I2Cx_DAT	STA	STO	AA	硬件下一步动作
0x60	拥有的 SLA+W 已接收; ACK 已返回	无 DAT 动作	X	0	0	数据字节会被接收; NOT ACK 位会被返回
		无 DAT 动作	X	0	1	数据字节会被接收; ACK 位会被返回
0x68	SLA+R/W 为主机的仲裁丢失; 拥有的 SLA+W 已接收; ACK 已返回	无 DAT 动作	X	0	0	数据字节会被接收; NOT ACK 位会被返回
		无 DAT 动作	X	0	1	数据字节会被接收; ACK 位会被返回
0x70	通用响应地址 (0x00) 已被接收; ACK 已返回	无 DAT 动作	X	0	0	数据字节会被接收; NOT ACK 位会被返回
		无 DAT 动作	X	0	1	数据字节会被接收; ACK 位会被返回
0x78	SLA+R/W 为主机的仲裁丢失; 通用响应地址 (0x00) 已被接收; ACK 已返回	无 DAT 动作	X	0	0	数据字节会被接收; NOT ACK 位会被返回
		无 DAT 动作	X	0	1	数据字节会被接收; ACK 位会被返回
0x80	之前寻址于自己的 SLA 地址; DATA 已接收; ACK 已返回	读数据字节	X	0	0	数据字节会被接收; NOT ACK 位会被返回
		读数据字节	X	0	1	数据字节会被接收; ACK 位会被返回
0x88	之前寻址于自己的 SLA 地址; DATA 已接收; NOT ACK 已返回	读数据字节	0	0	0	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址;
		读数据字节	0	0	1	切换到未寻址从机模式; 识别自己的 SLA; 通用响应地址会在 ADR[0] = 逻辑 1 时被识别
		读数据字节	1	0	0	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址; 当总线空闲时 START 状态会被发送
		读数据字节	1	0	1	切换到未寻址从机模式; 识别自己的 SLA; 通用响应地址会在 ADR[0] = 逻辑 1 时被识别; 当总线空闲时 START 状态会被发送
0x90	之前寻址于通用响应地址; DATA 已接收; ACK 已返回	读数据字节	X	0	0	数据字节会被接收; NOT ACK 位会被返回
		读数据字节	X	0	1	数据字节会被接收; ACK 位会被返回
0x98	之前寻址于通用响应地址; DATA 已接收; NOT ACK 已返回	读数据字节	0	0	0	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址;
		读数据字节	0	0	1	切换到未寻址从机模式; 识别自己的 SLA; 通用响应地址会在 ADR[0] = 逻辑 1 时被识别
		读数据字节	1	0	0	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址; 当总线空闲时 START 状态会被发送
		读数据字节	1	0	1	切换到未寻址从机模式; 识别自己的 SLA; 通用响应地址会在 ADR[0] = 逻辑 1 时被识别; 当总线空闲时 START 状态会被发送
0xA0	当仍寻址于从机接收或从机发送模式时 STOP 状态或重复 START 状态已接收	无 DAT 动作	0	0	0	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址;
		无 DAT 动作	0	0	1	切换到未寻址从机模式; 识别自己的 SLA; 通用响应地址会在 ADR[0] = 逻辑 1 时被识别
		无 DAT 动作	1	0	0	切换到未寻址从机模式; 不识别自己的 SLA 或通用响应地址; 当总线空闲时 START 状态会被发送

		无 DAT 动作	1	0	1	切换到未寻址从机模式；识别自己的 SLA；通用响应地址会在 ADR[0] = 逻辑 1 时被识别；当总线空闲时 START 状态会被发送
--	--	----------	---	---	---	--

● I2C 杂项事件表

表 15-8. I2C 杂项事件表

事件码	硬件和总线状态	到/来自 I2Cx_DAT	STA	STO	AA	硬件下一步动作
0xF8	没有可用的相关状态信息；总线被释放；EVENTF = 0 且没有中断产生	无 DAT 动作				等待或操作当前的传输
0x00	主机或选择的从机模式下总线错误（由于非法的 START 或 STOP 状态）；当干扰导致 I2C 块进入未定义的状态时，状态 0x00 也可能发生。	无 DAT 动作	0	1	x	在主机模式或被寻址的从机模式中仅影响内部硬件；在所有的情况下，总线会被释放且 I2C 会被切换到未寻址从机模式，STO 会被复位

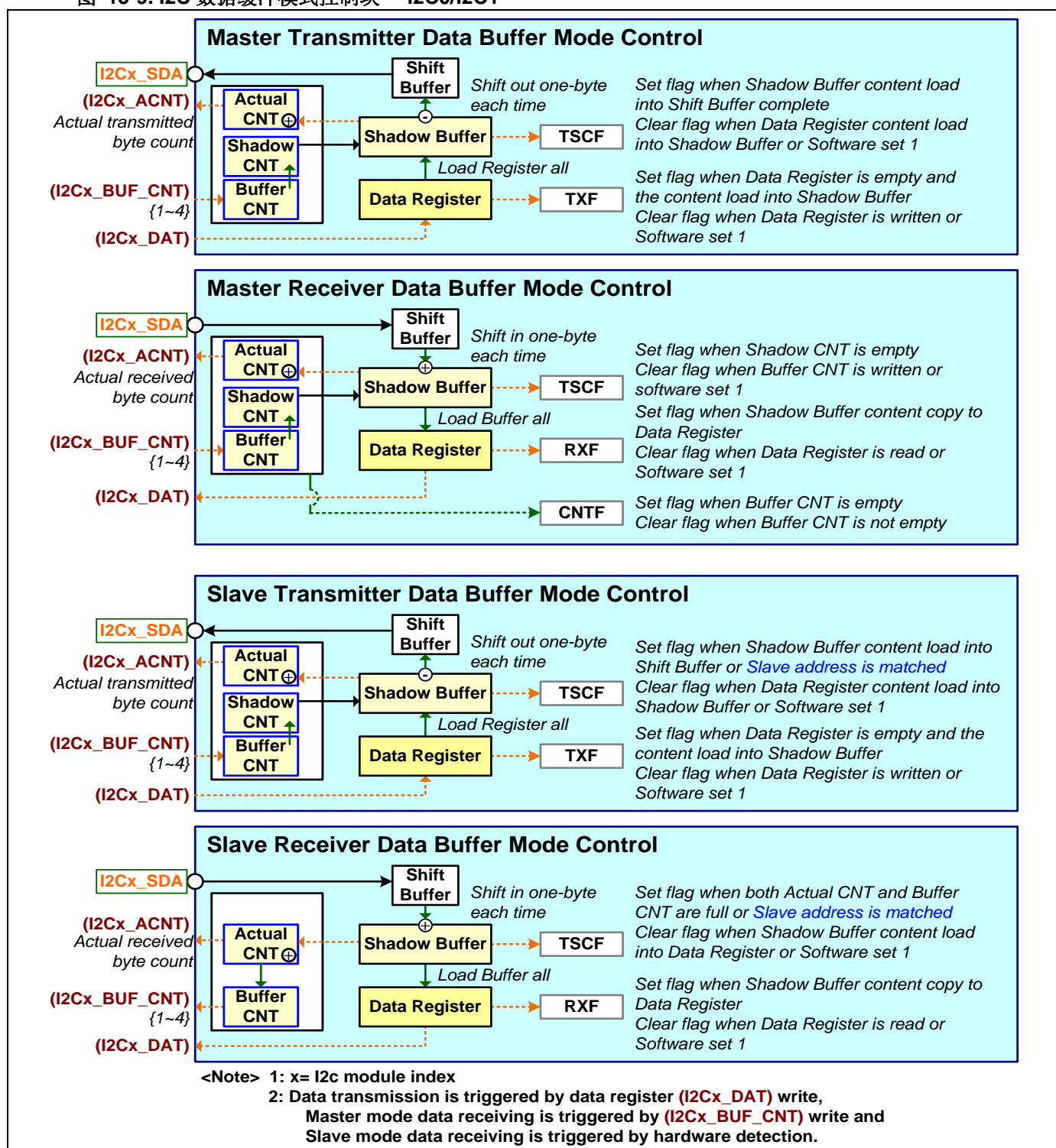
15.11. I2C 缓冲模式控制

用户可简单地使用 TXF (I2Cx_TXF)和 RXF (I2Cx_RXF)事件标志用于数据传输流控制。

15.11.1. I2C 缓冲模式数据缓冲

该模块包含 1 个 8 位移动缓冲、1 个 32 位阴影缓冲和 1 个 32 位数据寄存器用于数据缓冲模式的数据流控制。下面的图表展示了 I2C 数据缓冲模式控制块。

图 15-9. I2C 数据缓冲模式控制块 – I2C0/I2C1



15.11.2. I2C 数据缓冲控制

每次阴影缓冲内容被复制到(I2Cx_DAT)数据寄存器中时, RXF(I2Cx_RXF)标志都会被置起。此外, 若 I2Cx_BUFF_IE 寄存器已使能, 相关的中断将会被置起。

每次数据寄存器内容被复制到阴影缓冲中时, TXF(I2Cx_TXF)标志都会被置起。此外, 若 I2Cx_BUFF_IE 寄存器已使能, 相关的中断将会被置起。每次阴影缓冲内容被复制到移动缓冲中时, TSCF(I2Cx_TSCF)标志都会被置起。

I2C 阴影缓冲发送完成标志是 TSCF (I2Cx_TSCF)。通常这只用于内部硬件控制。

对于主机发送模式, 该标志会在阴影缓冲的内容载入移动缓冲完成时被置起, 当数据寄存器内容载入阴影缓冲或软件写 1 时清除。对于主机接收模式, 该标志会在缓冲 CNT (I2Cx_BUF_CNT 的阴影缓冲计数寄存器) 为空时置起, 在缓冲 CNT 被写入或软件写 1 时清除。

对于从机发送模式, 该标志会在阴影缓冲的内容载入移动缓冲完成或从机地址匹配时被置起, 当数据寄存器内容载入阴影缓冲或软件写 1 时清除。对于从机接收模式, 该标志会在 ACNT (I2Cx_ACNT 的实际计数寄存器) 和缓冲 CNT 满或从机地址匹配时置起, 在阴影缓冲内容载入数据寄存器或软件写 1 时清除。

I2Cx_BUF_CNT 寄存器用于作为发送或接收数据字节数阈值。当发送或接收的数据达到阈值且中断使能位 I2Cx_BUFF_IE 使能时, 会发生中断。当写该寄存器时, 硬件将自动清除 I2Cx_CNTF。

I2Cx_ACNT 寄存器可被读取以获取发送或接收数据的实际字节数值。当通过上次数据完成收发或错误状态时, 实际发送或接收的数据字节号被记录在这个寄存器中。该计数值不计算也不包括 NACK 错误字节。对于其他状态, 该寄存器值是无意义的。

15.11.3. I2C 缓冲模式访问指令

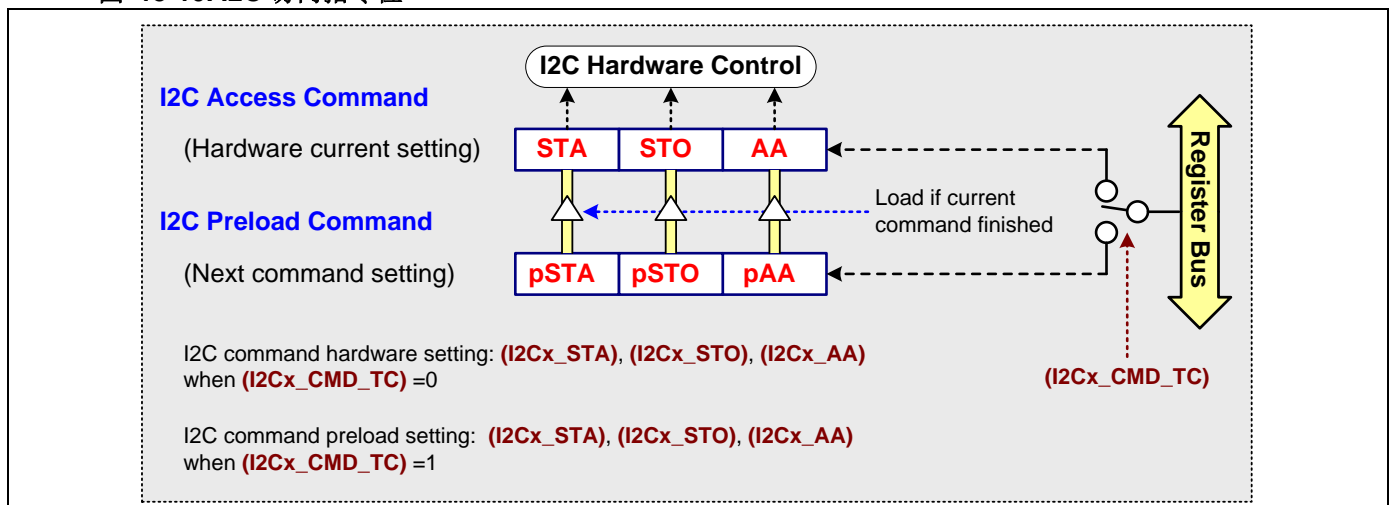
与字节模式相同, 相同的 3 个指令寄存器位 STA/STO/AA 用于控制 I2C 启动、停止和承认状态。这些位处于 I2Cx_STA, I2Cx_STO 和 I2Cx_AA 寄存器中。参照 “I2C 访问指令” 节以获取更多关于 I2C 访问指令的信息。

额外的 3 个指令寄存器位 pSTA/pSTO/pAA 作为预载指令用于设置下一个 I2C 指令。这些位可直接在 I2Cx_STA, I2Cx_STO 和 I2Cx_AA 寄存器中初始化。这些位同样通过被 I2Cx_CMD_TC 寄存器控制处于 I2Cx_STA, I2Cx_STO 和 I2Cx_AA 寄存器中。当 I2Cx_CMD_TC 寄存器被设置为 0 时, 写这 3 个寄存器位会初始化 I2C 访问指令用于硬件当前设置; 当 I2Cx_CMD_TC 寄存器被设置为 1 时, 写这 3 个寄存器位会初始化 I2C 预载指令用于下一个指令设置。

[注释]: I2Cx PSTA, I2Cx PSTO 和 I2Cx PAA 寄存器不支持于 MG32F02A132/072。

尽管指令位 STA/STO/AA 和 pSTA/pSTO/pAA 都是通过相同的寄存器位 I2Cx_STA, I2Cx_STO 和 I2Cx_AA 进行写操作的, 但是读这 3 个寄存器位只能得到 STA/STO/AA。

图 15-10. I2C 访问指令位



15.11.4. I2C 主机收发

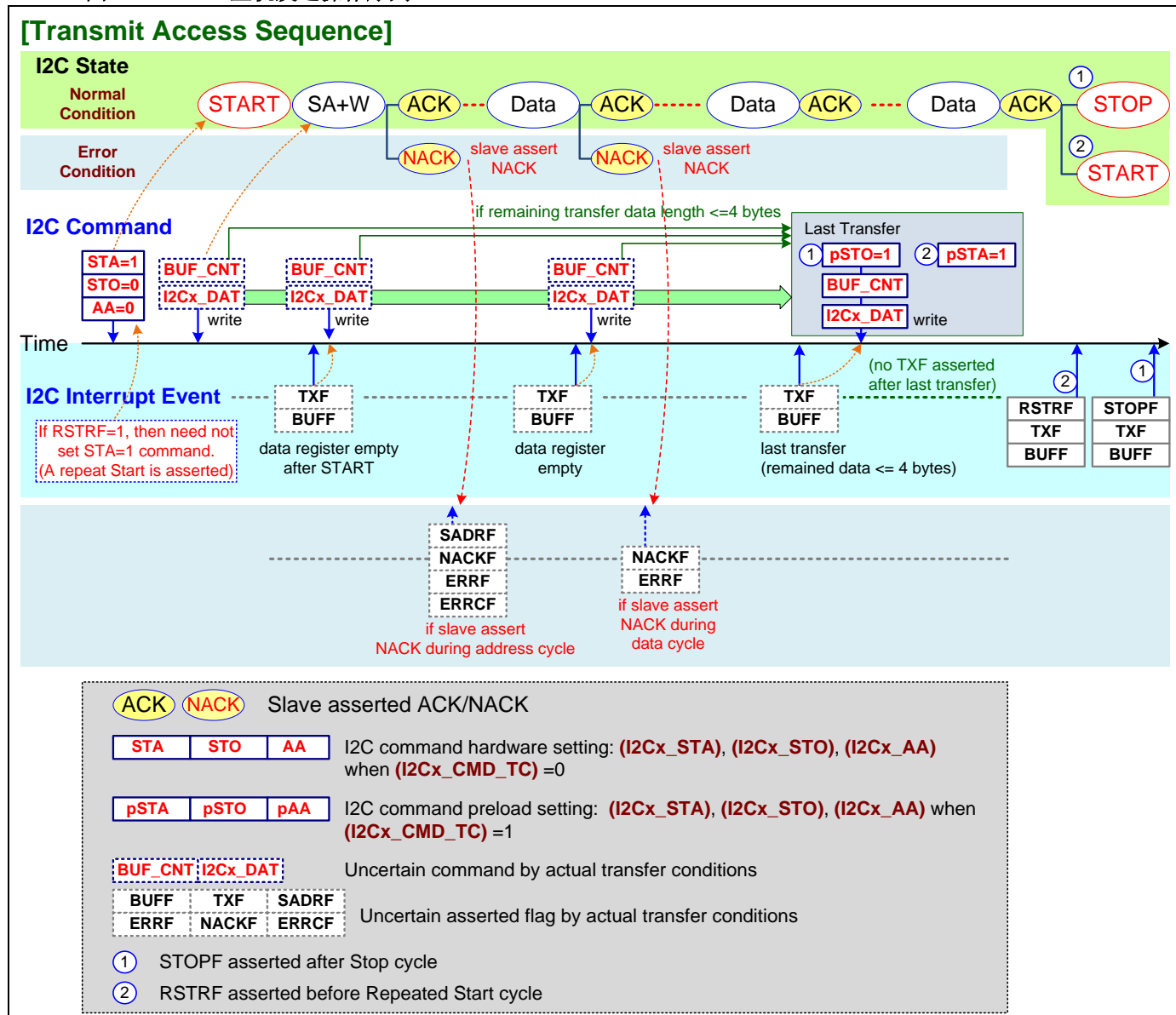
● I2C 主机发送操作序列

用户可设置 STA/STO/AA 指令以启动 I2C 主机发送操作。根据中断 TXF/BUFF, 用户把发送数据装入 I2C 数

据寄存器直到最后的数据发送。用户可在 I2C 最后一次数据寄存器写作 I2C 数据发送时初始化预载指令以准备停止 I2C 主机发送。

下面的图表展示了主机模式发送操作序列。

图 15-11. I2C 主机发送操作序列

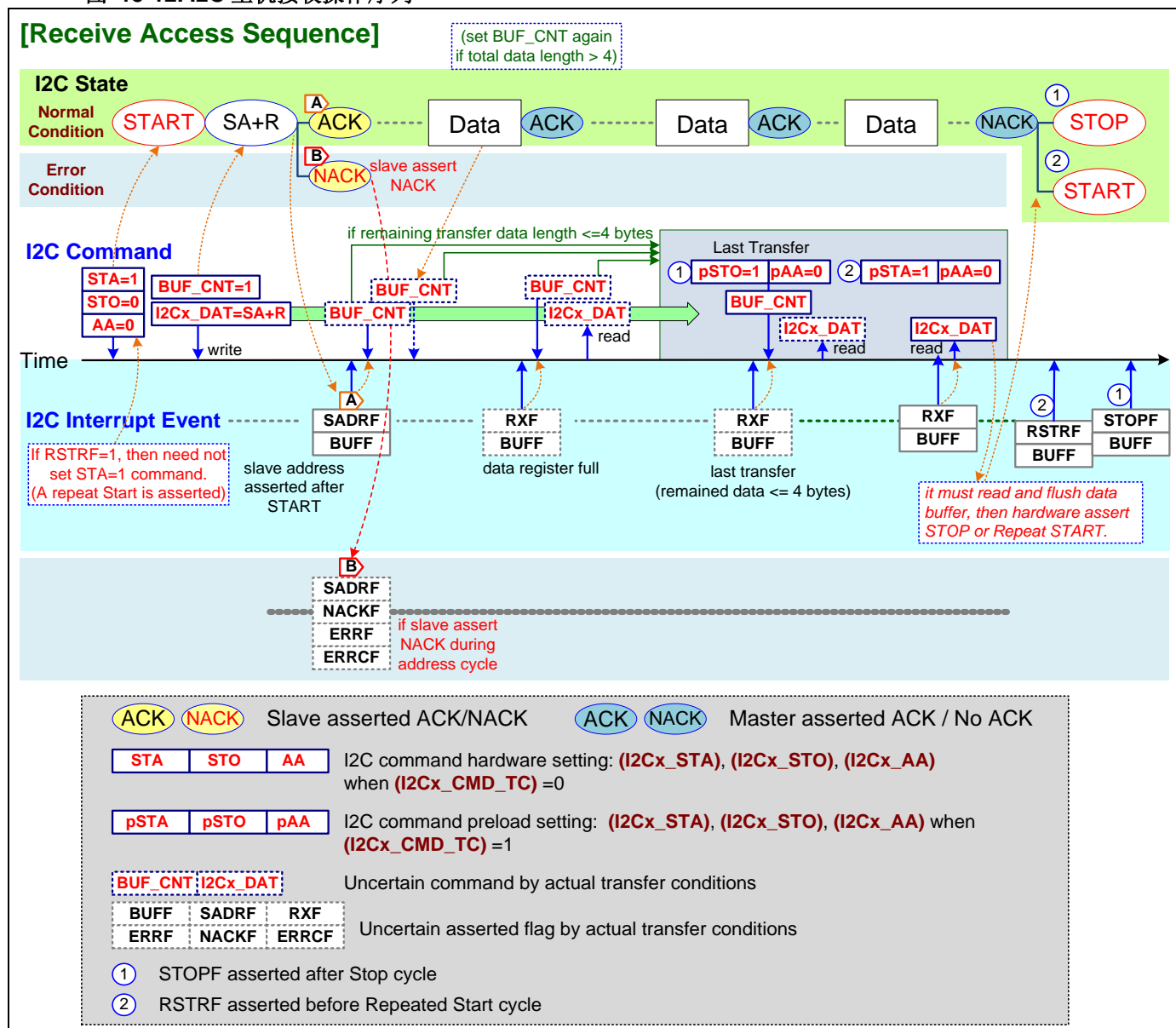


● I2C 主机接收操作序列

用户可设置 STA/STO/AA 指令以启动 I2C 主机接收操作。根据中断 RXF/BUFF，用户通过 I2C 数据寄存器获取接收数据并通过设置 AA 指令返回 ACK，直到最后的数据接收完成。用户可在 I2C 最后一次数据寄存器读作 I2C 数据接收时初始化预载指令以准备停止 I2C 主机接收。

下面的图表展示了主机模式接收操作序列。

图 15-12. I2C 主机接收操作序列



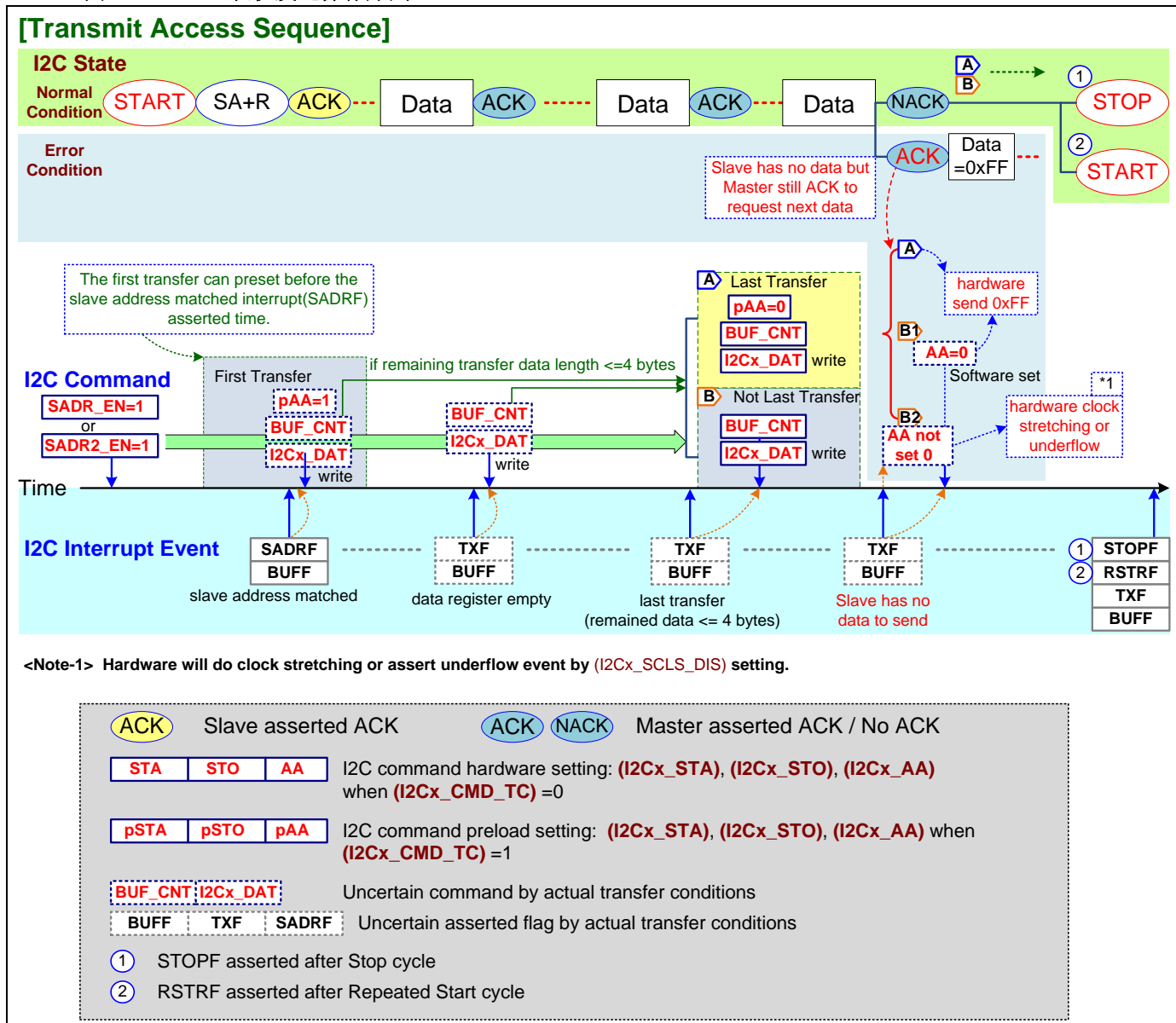
15.11.5. I2C 从机收发

● I2C 从机发送操作序列

当 I2C 控制器检测到匹配的从机地址且硬件置起 SDRF 标志，用户可初始化固件设置，并事先写发送 I2C 数据到 I2C 数据寄存器以准备进行 I2C 从机发送操作。根据中断 TXF/BUFF，用户把发送数据装入 I2C 数据寄存器直到最后的数据发送。若固件可通过预定义的数据流协议识别最后一个 I2C 数据发送，用户可在 I2C 最后一次数据寄存器写作 I2C 数据发送时初始化预载指令 pAA 以准备停止 I2C 主机发送。

下面的图表展示了从机模式发送操作序列。

图 15-13. I2C 从机发送操作序列

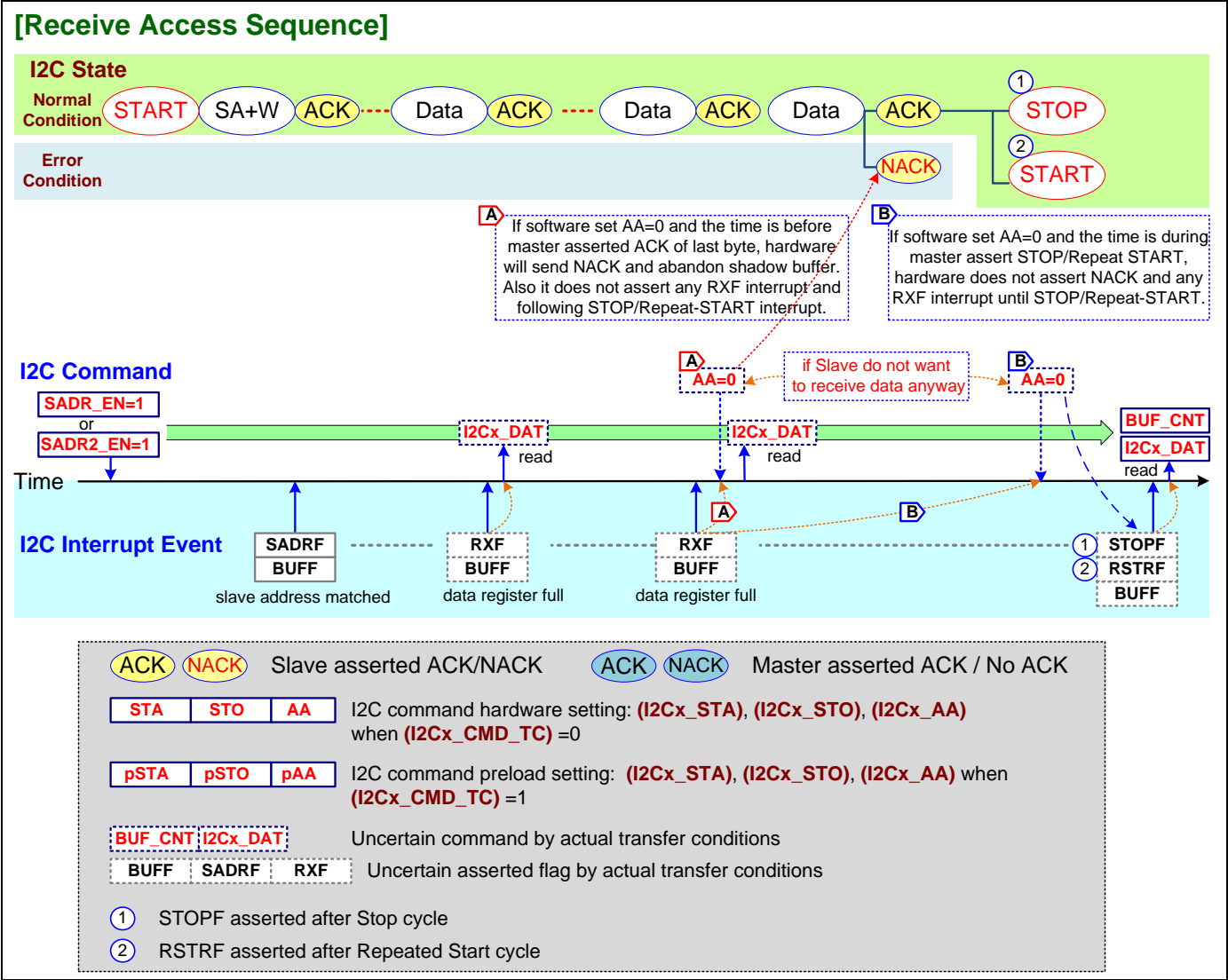


● I2C 从机接收操作序列

当 I2C 控制器被检测到匹配的从机地址且硬件置起 SADRf 标志，用户可初始化固件设置以进行 I2C 从机接收操作。根据中断 RXF/BUFF，用户从 I2C 数据寄存器获取接收的数据并通过设置 AA 指令返回 ACK 直到最后的数据接收完成。

下面的图表展示了从机模式接收操作序列。

图 15-14. I2C 从机接收操作序列



15.12. I2C 功能控制

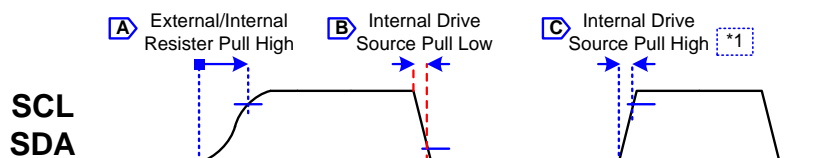
15.12.1. I2C 信号驱动控制时序

I2C 控制器支持驱动 SCL 和 SDA 信号为高电平。用户可通过设置 **I2Cx_PDRV_SEL** 寄存器选择预驱动时间为 **CK_I2Cx** 时钟的 0T（禁用预驱动），1T, 2T, 和 3T。

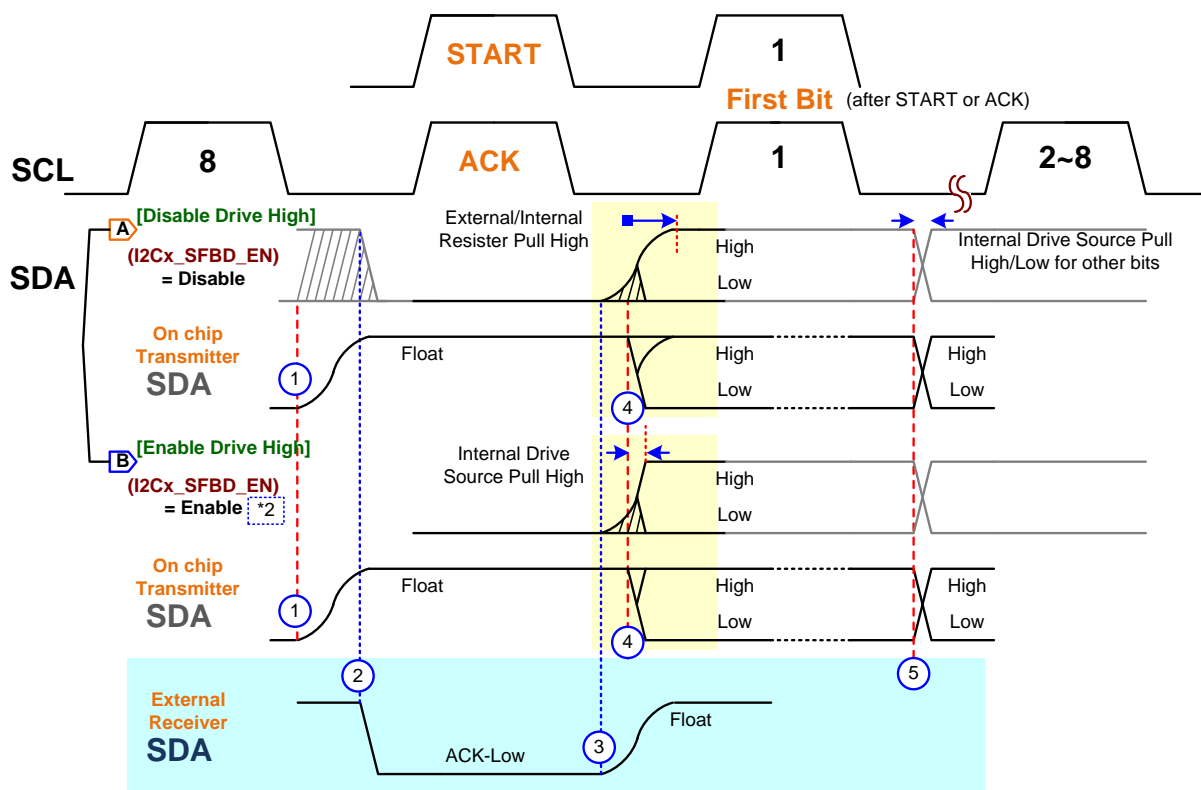
I2Cx_SFBD_EN 寄存器用于使能数据发送时 SDA 首位预驱动。当 **I2Cx_PDRV_SEL=0** 时，该位无效且被禁用。

图 15-15. I2C 信号驱动控制时序

[Signal Pull Types]



[SDA First Bit Drive High Option for transmitter]



- ① Transmitter release bus for Receiver asserted ACK
- ② Receiver drive low to assert ACK
- ③ Receiver release bus for next byte transfer
- ④ Resistor pull-high or Drive source pull-high/low optional for first bit
- ⑤ Drive source pull-high/low for bit 2~8

<Note-1> When (I2Cx_PDRV_SEL)=0T, chip will pull high by internal resistor for both SCL and SDA signals.

When (I2Cx_PDRV_SEL)≠0T, chip will pre-drive high for both SCL and SDA signals.

<Note-2> To set (I2Cx_SFBD_EN)=Enable is no effect if (I2Cx_PDRV_SEL)=0T and force to disable this function.

下面的表格展示了 I2C SCL 和 SDA 信号的输出驱动设定。

表 15-9. I2C 信号输出驱动设定

I2C 信号	I2Cx 寄存器		地址/数据位	
	PDRV_SEL	SFBD_EN	主机模式	从机模式
SCL	0	x	ODO	
	1,2,3	x	PDRV	ODO
I2C 信号	I2Cx 寄存器		1 st 数据位	地址位 数据 2~8 th 位
	PDRV_SEL	SFBD_EN	主/从	主/从
SDA	0	x	ODO	
	1,2,3	0	ODO	PDRV
		1	PDRV	

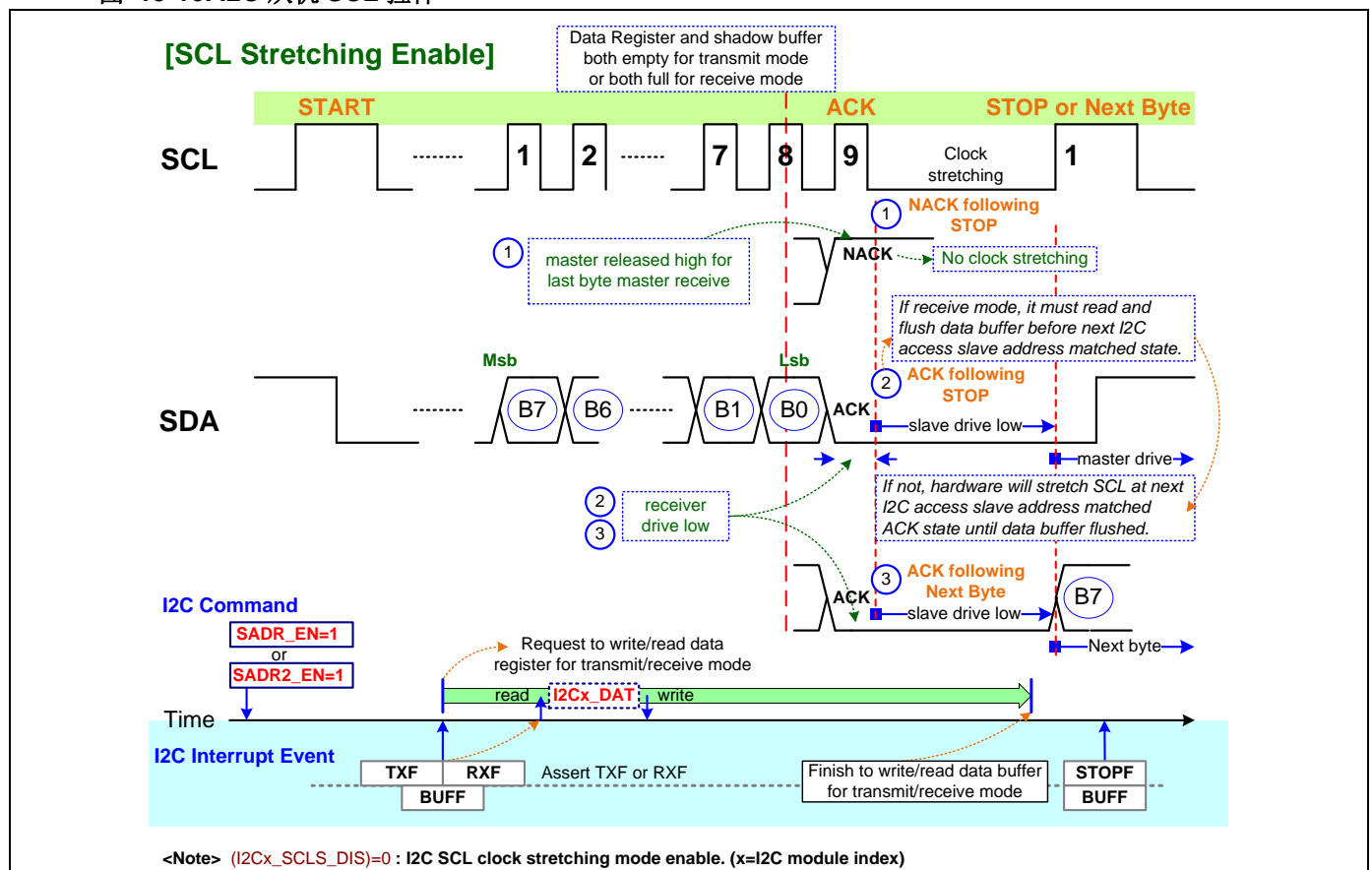
注释：I2C IO 引脚设置为开漏输出；x：不影响

ODO：开漏输出；PDRV：预驱动 1~3 时钟的高电平输出

15.12.2. I2C 从机 SCL 拉伸

I2C 支持从机模式中 SCL 时钟信号拉伸低电平功能。该功能通过 **I2Cx_SCLS_DIS** 寄存器使能或禁用，该功能默认是使能的。当使能时，该芯片会在 I2C 数据接收缓冲已满或发送缓冲为空时拉伸 SCL 信号到低电平。

图 15-16. I2C 从机 SCL 拉伸



15.12.3. STOP 模式唤醒

I2C 支持通过定时器上溢、定时器输入周期时钟和报警比较输出事件把芯片从 **STOP** 模式唤醒。他们独立唤醒使能位 **I2C_TF_WPEN**, **I2C_PC_WPEN** 和 **I2C_ALM_WPEN** 用于这三种唤醒事件。

当芯片进入 **STOP** 模式且任意一个 I2C 唤醒事件发生时, I2C 会发送唤醒事件到电源控制器(PW)作为系统唤醒事件。若相关控制寄存器已使能, 唤醒事件会使芯片唤醒。

在 **STOP** 模式中, I2C 控制模块可检测 **I2Cx_SADR** 或 **I2Cx_SADR2** 寄存器中设置的 I2C 从机地址。若 **I2Cx_WUP_IE** 和 **I2Cx_IEA** 寄存器被使能, 当拥有的 I2C 从机地址被检测到时, 芯片会被唤醒。

参照系统电源和中断章以获取更多关于唤醒事件和控制的相关信息。

[注释]: **STOP 模式唤醒功能不支持于 MG32F02A132/072。**

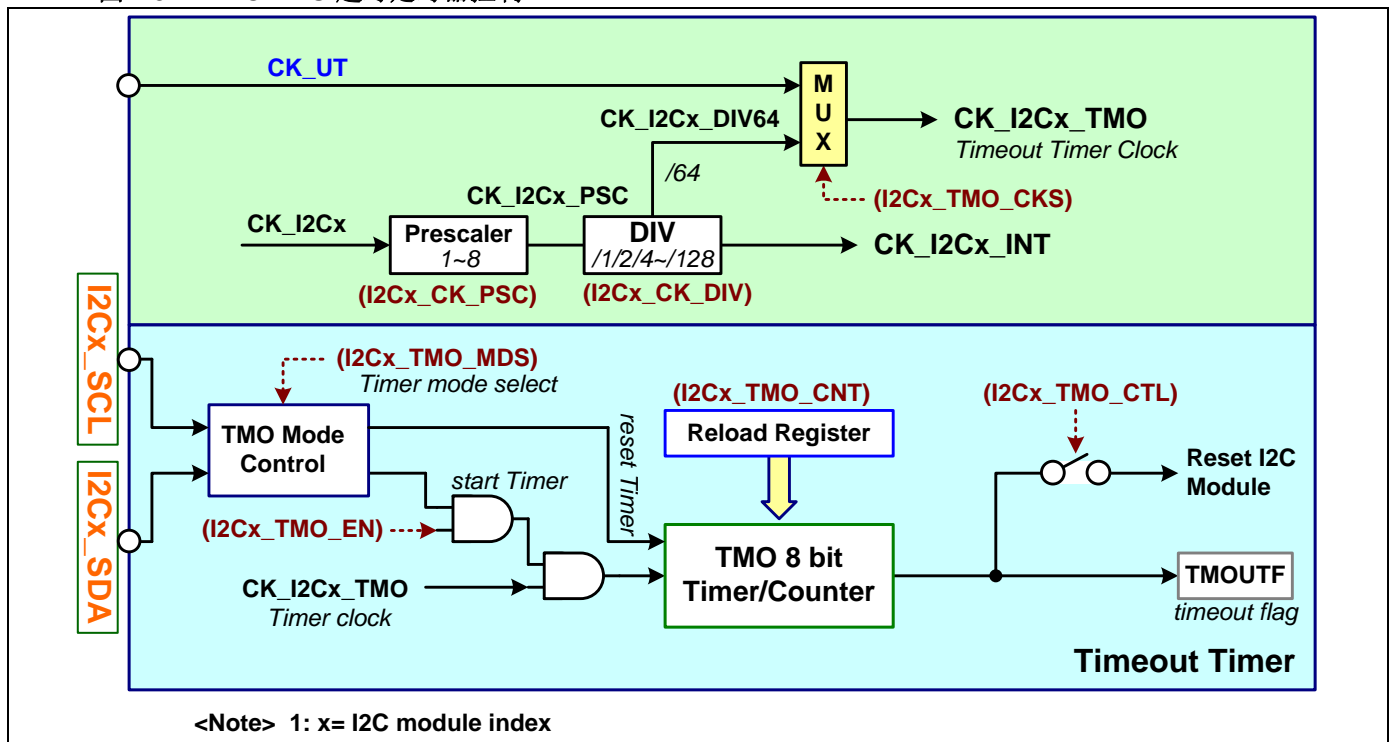
15.12.4. I2C 超时定时器控制

该模块提供 1 个 8 位超时定时器 (TMO) 用于 I2C 访问超时控制。它可以通过 **I2Cx_TMO_MDS** 寄存器设置并通过 **I2Cx_TMO_EN** 寄存器使能为 I2C 超时定时器或通用定时器。当 TMO 作为 I2C 超时定时器时, 用户可通过设置 **I2Cx_TMO_MDS** 寄存器选择检测和计数 SCL 低电平或总线空闲状态 (SCL 和 SDA 保持高电平) 的时间。

TMO 输入定时器时钟源可通过 **I2Cx_TMO_CKS** 寄存器设置选择 **CK_UT** 或 **CK_I2Cx_DIV64** 时钟源。**CK_I2Cx_DIV64** 为固定的 **CK_I2Cx_PSC** 时钟信号的 64 分频。

TMO 提供了 **I2Cx_TMO_CNT** 寄存器用于设置 TMO 超时时间。当 TMO 超时发生时, 用户可通过 **I2Cx_TMO_CTL** 寄存器使能复位 I2C 控制器。

图 15-17. I2C TMO 超时定时器控制



当 TMO 定时器被设置为通用定时器, 该定时器通过 **I2Cx_EN** 寄存器的使能功能是无效的。

[注释]: 在 MG32F02A132/072 中, 当 TMO 定时器作为“通用定时器”时, **I2Cx_EN** 必须被使能。

下面的表格展示了 I2C TMO 超时定时器启动/关闭控制。

表 15-10. I2C TMO 超时定时器启动/关闭控制

TMO 定时器功能	I2Cx 寄存器			定时器启动/关闭
	TMO_MDS	EN	TMO_EN	
I2C 超时定时器	0x	0	x	定时器关闭
		1	0	定时器关闭
			1	定时器启动
通用定时器	10	x	0	定时器关闭
			1	定时器启动

x：不影响

15.12.5. I2C NACK 控制

通常，除主机发送模式的最后一个字节外，I2C 设备必须在接收到 8 位数据或从机地址后置起 ACK 位。当检测到异常 NACK，芯片会置起 NACKF 标志，若中断使能位(I2Cx_NACKF)被使能，还会置起相关中断。请参照“[I2C 中断标志](#)”和“[I2C 主机发送 NACK 检测](#)”节以获取更多关于 NACK 状态的信息。

对于主机发送模式应用，用户通过设置 I2Cx_NACK_EN 寄存器可在接收到缓冲模式的 NACK 位时忽略接收 NACK 和使能 I2C 来持续的发送下一个数据。

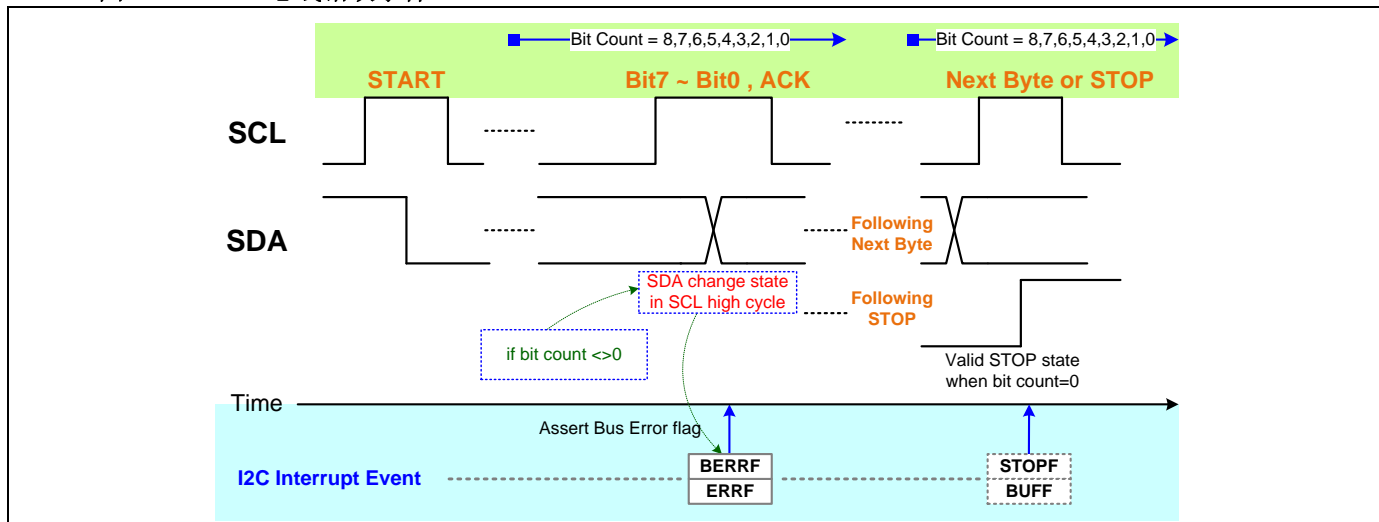
[注释]: I2Cx_NACK_EN 寄存器不支持于 MG32F02A132/072。

15.13. I2C 错误管理

15.13.1. I2C 总线错误事件

当 I2C 控制器在数据位发送周期检测到 START 或 STOP 状态但其中 1 个数据位还没被发送完成时(位计数 $\neq 0$)，BERRF 标志(**I2Cx_BERRF**)会被置起。

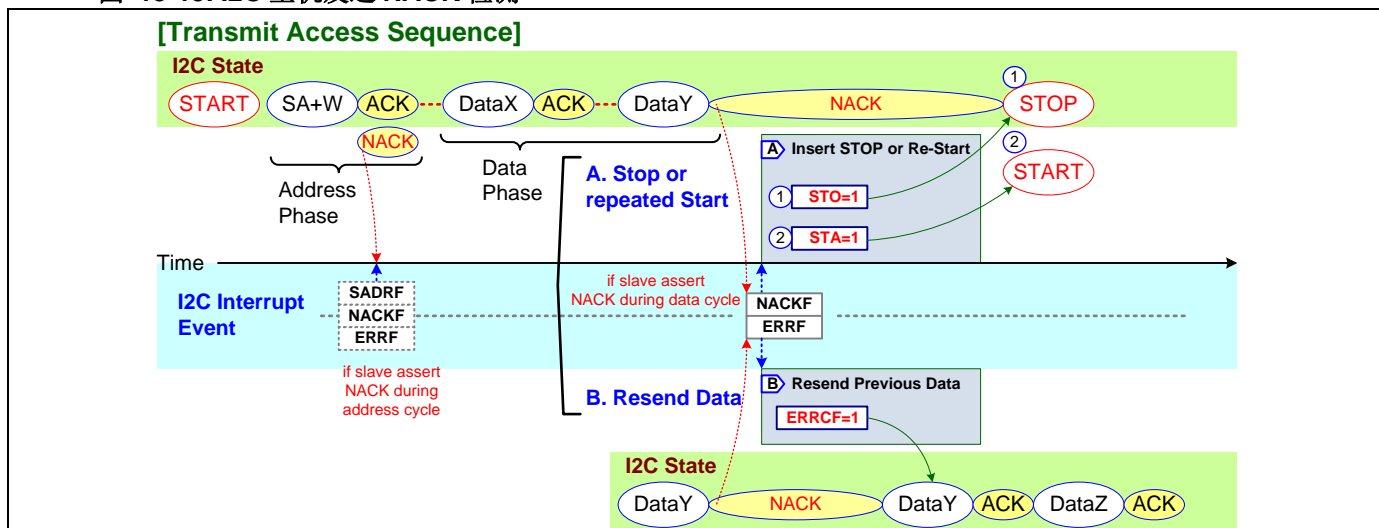
图 15-18. I2C 总线错误事件



15.13.2. I2C 主机发送 NACK 检测

对于 I2C 主机发送模式，I2C 接收到 NACK 状态，NACK 标志(**I2Cx_NACKF**)会被置起；当从机地址周期中 NACK 被检测到，SADRF 标志(**I2Cx_SADRF**)会同时被置起；当数据字节周期中 NACK 被检测到，用户可直接设置 **I2Cx_STA** 或 **I2Cx_STO** 寄存器位，置起重复 START 或 STOP 状态，或者用户可设置 **I2Cx_ERRCF** 寄存器为 1 以重新发送上一个数据字节。

图 15-19. I2C 主机发送 NACK 检测

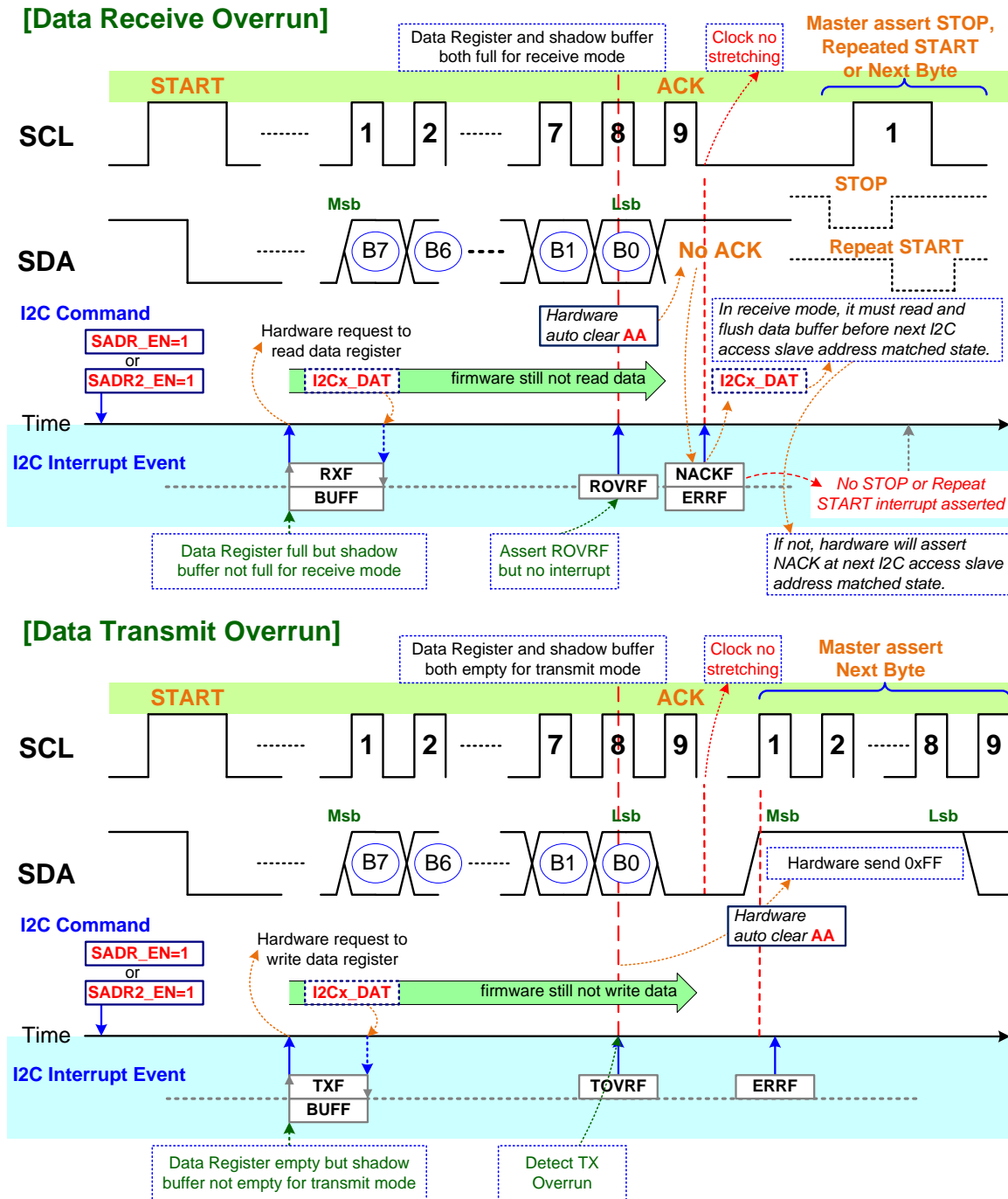


I2Cx_TXRF 是只读状态位，用于标识 I2C 发送数据寄存器保持状态。当 I2C 通信总线发生 NACK 错误且 **I2Cx_NACKF** 已置起，该位是用于标识数据寄存器内容是否还有数据要发送。此状态下，I2C 主机会置起 STOP。用户可通过读 **I2Cx_ACNT** 寄存器计算正确的总发送数据值。**I2Cx_TXRF** 位会在从机地址匹配状态清除并在最后字节 NACK 状态后更新。

15.13.3. I2C SCL 拉伸禁用时从机数据溢出

当从机模式下，阴影缓冲满且 RX 数据寄存器(I2Cx_DAT)不为空时，若移动缓冲接收已满，ROVRF 标志(I2Cx_ROVRF)会被置起。当从机模式下，阴影缓冲和 TX 数据寄存器 (I2Cx_DAT) 为空时，若移动缓冲移动已完成且外部 I2C 主机仍请求数据，TOVRF 标志(I2Cx_TOVRF) 会被置起。

图 15-20. I2C SCL 拉伸禁用时从机数据溢出



<Note> (I2Cx_SCLS_DIS)=1 : I2C SCL clock stretching mode disable. (x=I2C module index)

15.14. I2C DMA 操作

I2C 支持字节模式下的数据 DMA 收发。

15.14.1. DMA 模块设置

当芯片支持 DMA（直接内存访问）控制器时，用户可以在 DMA 数据传输前，在 DMA 模块中设置关于来源/目的设备、通道请求仲裁等 DMA 设置。DMA 来源和目的可以是内存或外设。

参照 DMA 章以获取更多关于 DMA 模组设置的细节。

15.14.2. I2C DMA 控制

DMA 设置完成后，用户需设置 I2C 模块的 DMA 使能位 **I2Cx_DMA_RXEN** 和 **I2Cx_DMA_TXEN**。

最后，相关的通道请求起始位 **DMA_CHn_REQ** 是必须的，用于设置 DMA 传输启动(n = DMA 通道标号)。然后传输源和目的设备会置起 RX/TX 请求信号到 DMA 控制器，DMA 控制器便会置起接收信号到请求源/目的设备。此时，该数据传输连接是用于 DMA 传输的。

- **I2C RX 到 DMA**

I2Cx_DMA_RXEN 寄存器位是用于使能从 I2C 接收的数据到 DMA 目的地的 DMA 发送。

DMA 发送周期中，接收数据标志 **I2Cx_RXF** 是被硬件屏蔽的。

- **I2C TX 来自 DMA**

I2Cx_DMA_TXEN 寄存器位是用于使能 DMA 数据寄存器的源到 I2C 发送输出的 DMA 发送。

在 DMA 发送周期中，发送标志 **I2Cx_TXF** 是被硬件屏蔽的。通常，发送完成标志 **I2Cx_TSCF** 会在 DMA 发送完成后被置起。

当硬件检测到其中 1 种错误或特殊情况，硬件会清除 **I2Cx_DMA_TXEN** 和 **I2Cx_DMA_RXEN** 位。

15.14.3. I2C 的 DMA 中断标志控制

DMA 工作周期中，模块的中断标志会控制和执行 3 种类型，如下表格。其中一方在 DMA 工作过程中被屏蔽，另一方会在标志被置起后禁用 DMA 功能。此时，硬件会禁用 **I2Cx_DMA_TXEN** 和 **I2Cx_DMA_RXEN** 位。其他操作与 DMA 操作中不执行的操作相同。

表 15-11. I2C DMA 功能中断标志控制

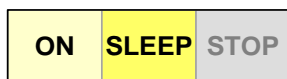
行为	DMA 操作中屏蔽标志	标志置起后 DMA 被禁用(*1)	一般控制	
外设	(数据溢出标志)	(错误/检测标志)	(其他标志)	
I2Cx	I2Cx_EVENTF I2Cx_BUFF	I2Cx_ERRF(*1) I2Cx_STPSTRF(*1)	I2Cx_TMOUTF I2Cx_WUPF	I2Cx_RXF I2Cx_TXF I2Cx_RSTRF I2Cx_STOPF I2Cx_CNTF I2Cx_ERRCF I2Cx_SADRF I2Cx_TSCF I2Cx_ROVRF I2Cx_TOVRF I2Cx_NACKF I2Cx_ALOSF I2Cx_BERRF

注释-1：当标志被置起，若相关中断使能位没被使能，它不会强行禁用外设 DMA。特别的是，不管你有没有使能相关中断，I2Cx_ERRF 或 I2Cx_STPSTRF 标志会强制禁用 DMA。

[注释]: **I2Cx_WUPF** 标志不支持于 MG32F02A132/072。

16. UART (通用异步收发传输器)

16.1. 简介



The module can be running in ON and SLEEP modes only.

UART 模块支持全双工传输，意味着它可以同时发送和接收。该模块内置阴影缓冲器和数据寄存器独立地用于发送和接收中，以提高发送和接收通信性能。在从寄存器中读取先前接收的字节之前，可以开始接收第二字节。然而，如果在第二字节的接收完成时仍未读取第一字节，则其中一个字节将丢失。

该模块可以多种方式工作：异步通信、同步通信、SPI 主机、**SmartCard**、**LIN**、多处理器模式。异步通信作为全双工通用异步接收机和发射机(UART)工作，它可以不同波特率同时发送和接收。

注释：(x = 模块标号) 会被用于该章的寄存器、信号和引脚/端口描述中。[EX]: **URTx_EN** ~ x 表示模块标号 0,1, 2,3。

16.2. 特性

- **UART 模块一般功能**
 - 通过可编程过采样率提供精确的 UART 波特率控制
 - 可编程数据字长- 7 或 8 位
 - 可选择 MSB 或 LSB 数据顺序
 - 硬件奇偶校验与奇偶校验生成
 - 可编程 4~32 过采样率
 - 可选择 1 或 3 点用于数据多数投票
 - 分离的使能位用于发送和接收
 - 可交换 TX/RX 引脚配置
 - 发送与接收信号独立的极性控制
 - 支持 Idle/RX/Break/Calibration 的超时定时器超时检测
 - 支持多种事件硬件自动检测进入或退出 UART 静音模式
 - 支持 UART,同步,SPI 主机,智能卡,LIN,多处理器模式
 - 支持 4 字节数据缓冲和 32 位数据寄存器用于高速数据通信
 - 支持使用 DMA 收发数据
 - 接收波特率最高 6 Mbit/s
 - 支持自动波特率检测和校准
 - 支持多处理机通信用于主从机模式 -Idle 线，地址位
 - 支持低速类 UART 的 IrDA 帧格式
 - 支持仅通过 CTS/RTS 信号进行收发硬件流控制
 - 提供驱动使能信号以启动传输用于单线通信
 - 支持 RX 超时检测用于字符时效检测
 - 智能卡应用中支持传输错误的硬件检测与自动重传控制
 - 智能卡应用中支持接收奇偶错误硬件检测和自动重试控制

❖ **MG32F02A132/072**

- 提供四个相同的 UART 模块: URT0,URT1,URT2,URT3
- 可配置的停止位- 1 或 2 位停止位

❖ **MG32F02A032**

- 提供两个相同的 UART 模块: URT0,URT1
- 可配置的停止位- 0.5, 1, 1.5 或 2 位停止,

16.3. 配置

16.3.1. 芯片配置

下面的表格展示了芯片的 UART 模块。

表 16-1. UART 配置

芯片	UART 模块			
	URT0	URT1	URT2	URT3
MG32F02A132	V	V	V	V
MG32F02A072	V	V	V	V
MG32F02A032	V	V		

注释 V：包含

16.3.2. 模块功能

下面的表格展示了 UART 模块包含的功能。

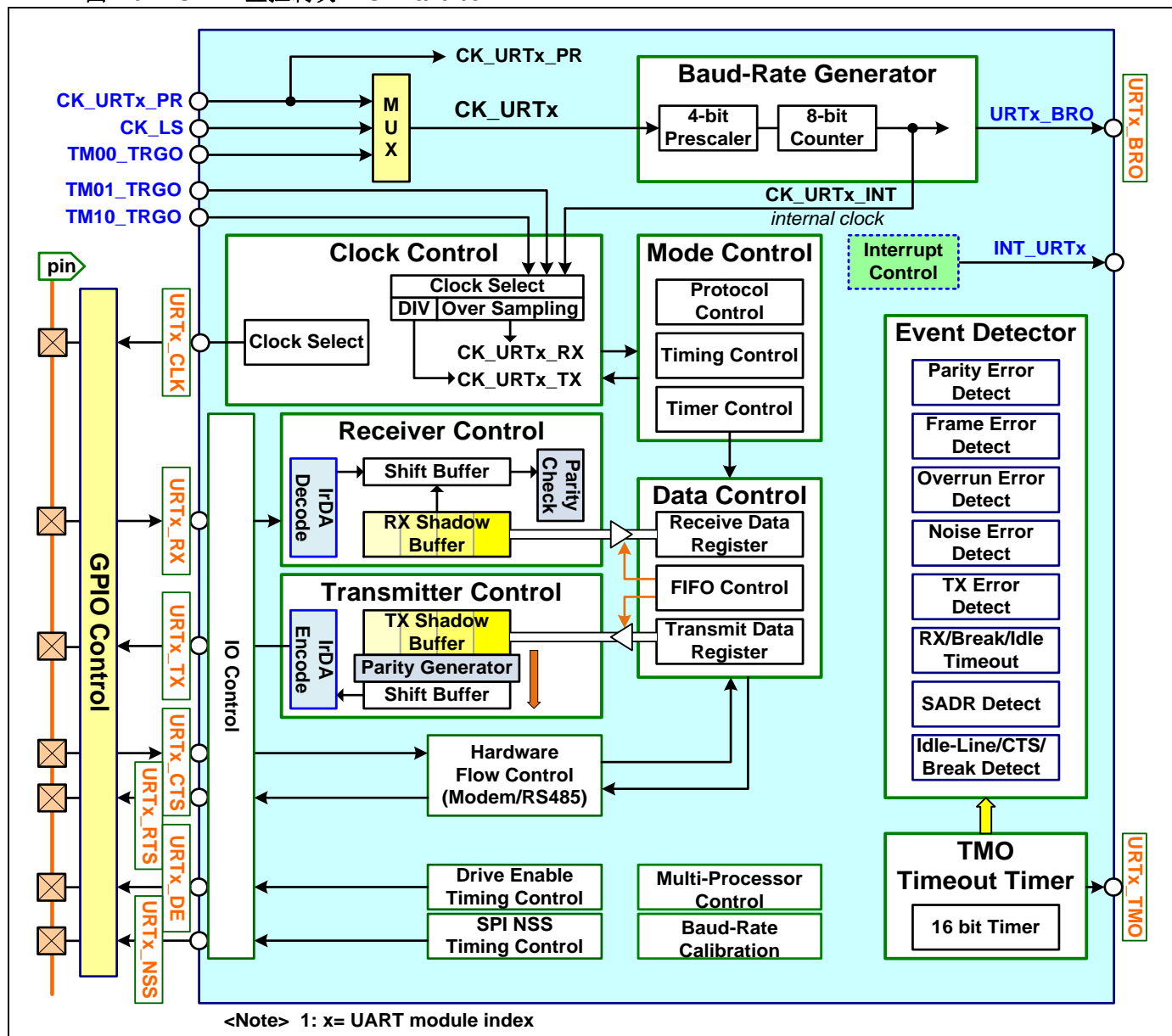
表 16-2. UART 模块功能

模块功能	芯片	MG32F02A132 MG32F02A072	MG32F02A032	注释
		URT0/1/2/3	URT0/1	
UART –异步		yes	yes	
同步-主机 2 数据线		yes	yes	SPI 主机模式
同步-主机 1 数据线		yes	yes	半双工通信
智能卡-ISO7816-3		yes	yes	
LIN		yes	yes	
多处理器-地址位		yes	yes	
多处理器-Idle 线		yes	yes	
IrDA –类 UART		yes	yes	低速类 UART 帧格式 IrDA(SIR 普通模式)
硬件流控制		yes	yes	只支持 CTS/RTS
外部时钟引脚		1	1	
定时器 BRO,TMO 引脚		2	2	
阴影缓冲		4 字节	4 字节	
7 位数据选择		yes	yes	
TX 校验位生成		yes	yes	硬件自动从数据位产生校验位
Msb/Lsb 发送选择		yes	yes	
可设置停止位		1, 2	0.5, 1, 1.5, 2	可设置停止位长度
自动波特率校准		yes	yes	自动波特率检测和校准
自动进入/退出静音模式		yes	yes	若未发生地址匹配进入静音模式
中止状态检测		yes	yes	
Idle 线检测		yes	yes	
可设置过采样数		8~32	8~32	
可设置时钟相位/极性		yes	yes	用于同步模式
通用定时器控制		yes	yes	波特率定时器和超时定时器作为通用定时器
驱动使能		yes	yes	用于外部发送者的发送模式的驱动使能信号
RX 校验错误检测		yes	yes	接收数据字节进行检查校验
帧错误检测		yes	yes	
数据溢出检测		yes	yes	接收缓冲超过阈值；发送缓冲空
TX 错误检测		yes	yes	智能卡/LIN
噪音特征检测		yes	yes	用于噪音特征的跳过或不跳过
Idle 超时检测		yes	yes	用于智能卡应用
RX 超时检测		yes	yes	字符时效检测
Break 超时检测		yes	yes	用于 LIN 应用
校准超时检测		yes	yes	用于 LIN 应用
DMA 支持		yes	yes	

16.4. 控制块

下面的图表展示了 UART 控制块。

图 16-1. UART 主控制块 – URT0/1/2/3



16.5. IO 线

16.5.1. IO 信号

- URT_x_CLK**
 UART 时钟信号，并作为 UART 同步模式或 SPI 主机模式输出。
- URT_x_RX**
 UART 接收 RX 数据信号或 SPI 主机模式的 SPI MISO 输入信号。
- URT_x_TX**
 UART 发送 TX 数据信号或 SPI 主机模式的 SPI MOSI 输出信号。
- URT_x_NSS**
 SPI 从机选择或片选输出信号用于 SPI 主机模式。

- **URTx_CTS**
UART 清除发送向内的流控制输入信号。
- **URTx_RTS**
UART 请求发送向外的流控制输入信号。
- **URTx_DE**
UART 数据使能输出信号用于外部驱动设备。
- **URTx_BRO**
来自 UART 波特率定时器溢出信号的 UART 输出信号。
- **URTx_TMO**
来自 UART 超时定时器溢出信号的 UART 输出信号。

16.5.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。参照用户手册 GPIO 章中“[IO 模式](#)”节以获取更多关于 IO 模式设置的信息。

每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“[功能复用选择](#)”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“[引脚功能复用表](#)”以获取更多信息。

16.6. 使能和时钟

16.6.1. UART 全局使能

UART 模块的所有功能全局使能位是 **URTx_EN**。当该位被禁用时，所有的 UART 功能将无法工作。

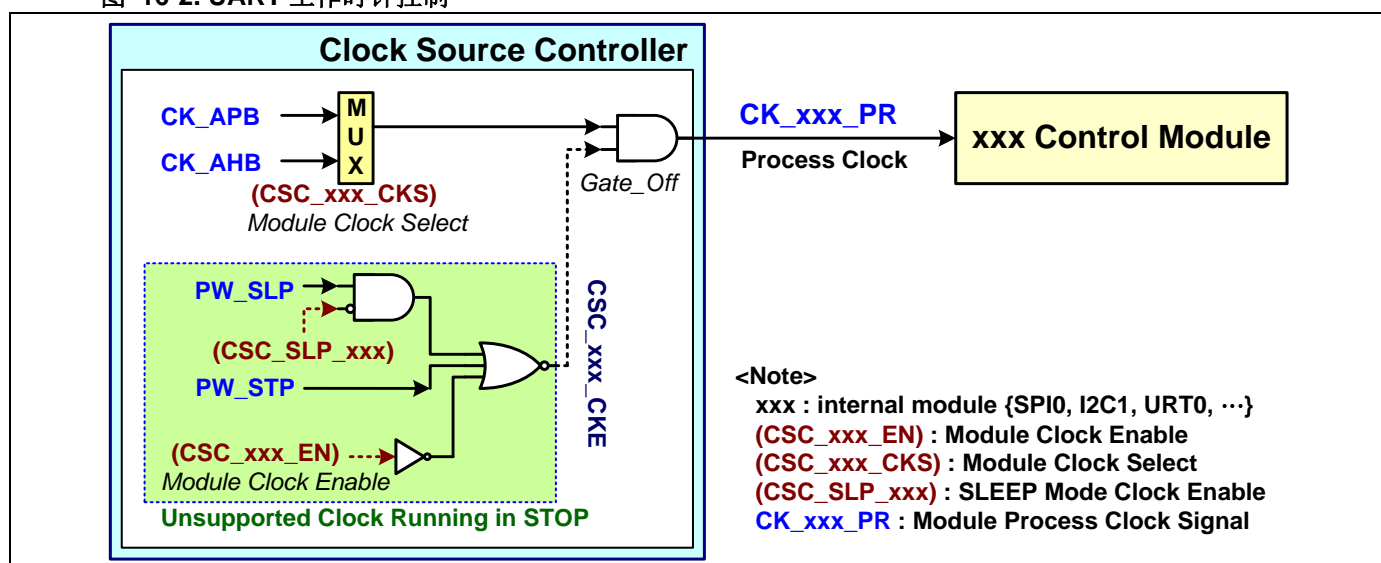
16.6.2. UART 时钟控制

● 模块工作时钟

该模块工作时钟 **CK_URTx_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 CSC(时钟源控制器)模块。该时钟可通过 **CSC_URTx_EN** 寄存器使能并通过 **CSC_URTx_CKS** 寄存器选择时钟源来自 APB 或 AHB 时钟。

ON 下，工作时钟只会在 **CSC_URTx_EN** 寄存器被使能时运行；用户可以通过设置 **CSC_SLP_URTx** 寄存器规划在 **SLEEP** 模式下是否让 UART 继续运行；**STOP** 模式下，工作时钟会被禁用，参照系统时钟章以获取更多信息。

图 16-2. UART 工作时钟控制



● 模块内部时钟

用户可通过 **URTx_CK_SEL** 寄存器设置选择模块工作时钟源 **CK_URTx_PR**，内部 ILRCO 时钟 **CK_ILRCO** 或定时器触发输出信号 **TM00_TRGO**。

UART 模块内建 1 个波特率发生器以输出内部时钟 **CK_URTx_INT** 作为采样时钟用于接收数据信号和发送数据信号的时钟源。此外，它可输出 **CK_URTx_SC** 时钟用于 **SmartCard** 应用。用户可通过 **URTx_PSR** 和 **URTx_RLR** 寄存器设置波特率发生器。

[注释]: 通常内部时钟频率需比模块工作时钟慢至少 1/2。

对于异步通信，数据收发可用不同的波特率用于 **URTx_RX** 和 **URTx_TX** 信号。**URTx_TX_CKS** 和 **URTx_RX_CKS** 寄存器独立使用以选择时钟源作输入信号 **URTx_RX** 采样和输出信号 **URTx_TX** 的生成。

对于同步通信，**URTx_CLK** 可作为时钟输出，该时钟可通过 **URTx_CLK_EN** 寄存器使能。用户可设置 **URTx_CLK_CKS** 寄存器选择 **CK_URTx_SC** 用于 **SmartCard** 或 **CK_URTx_OUT** 用于其他同步通信应用。

[注释]: **URTx_CLK** 输入作为时钟输入不支持于 MG32F02A132/072。

输出信号 **URTx_BRO** 可来自波特率发生器定时器溢出信号并作为时钟输出信号。

下面的图表展示了 UART 时钟控制块。

图 16-3. UART 时钟控制 – MG32F02A132/072

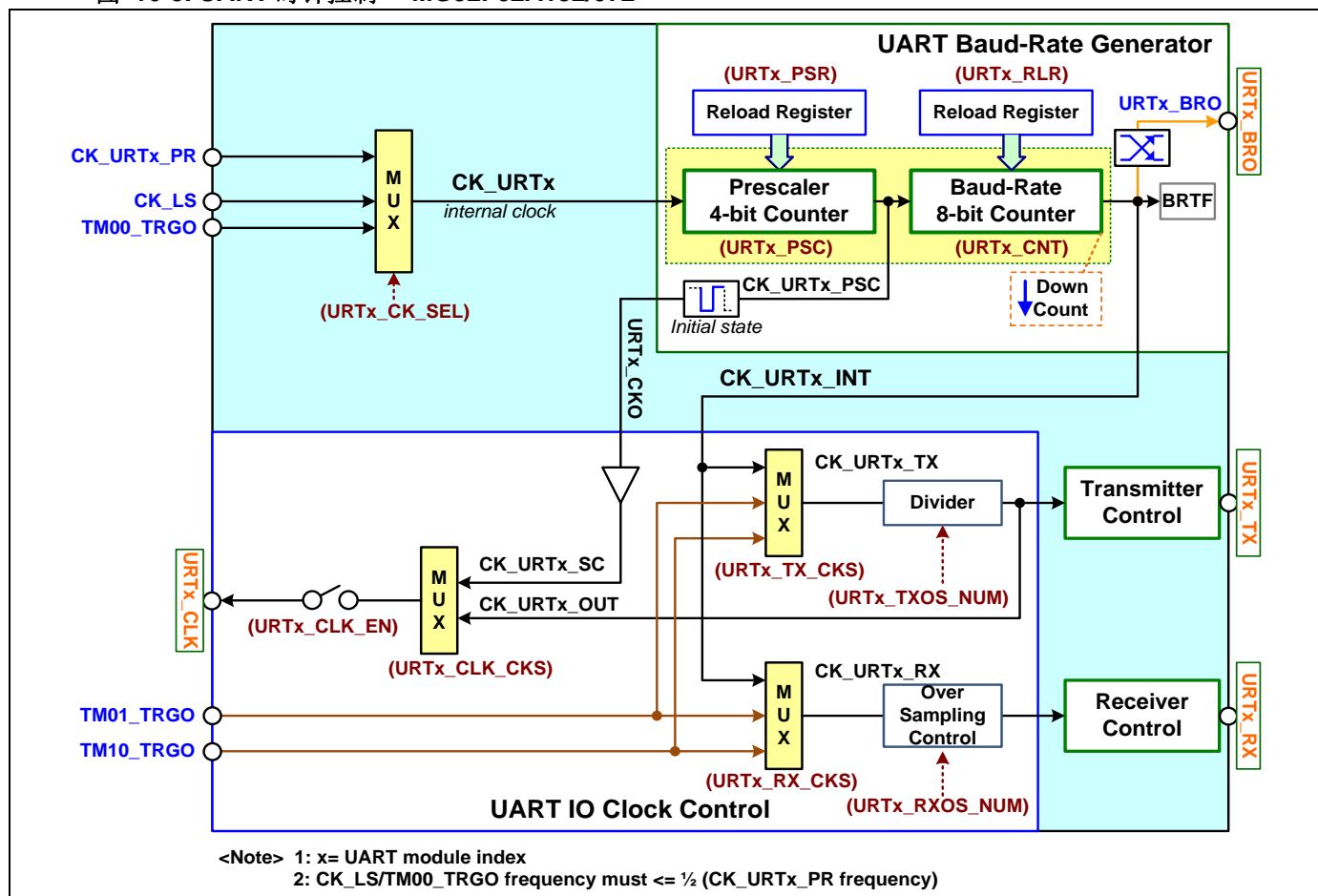
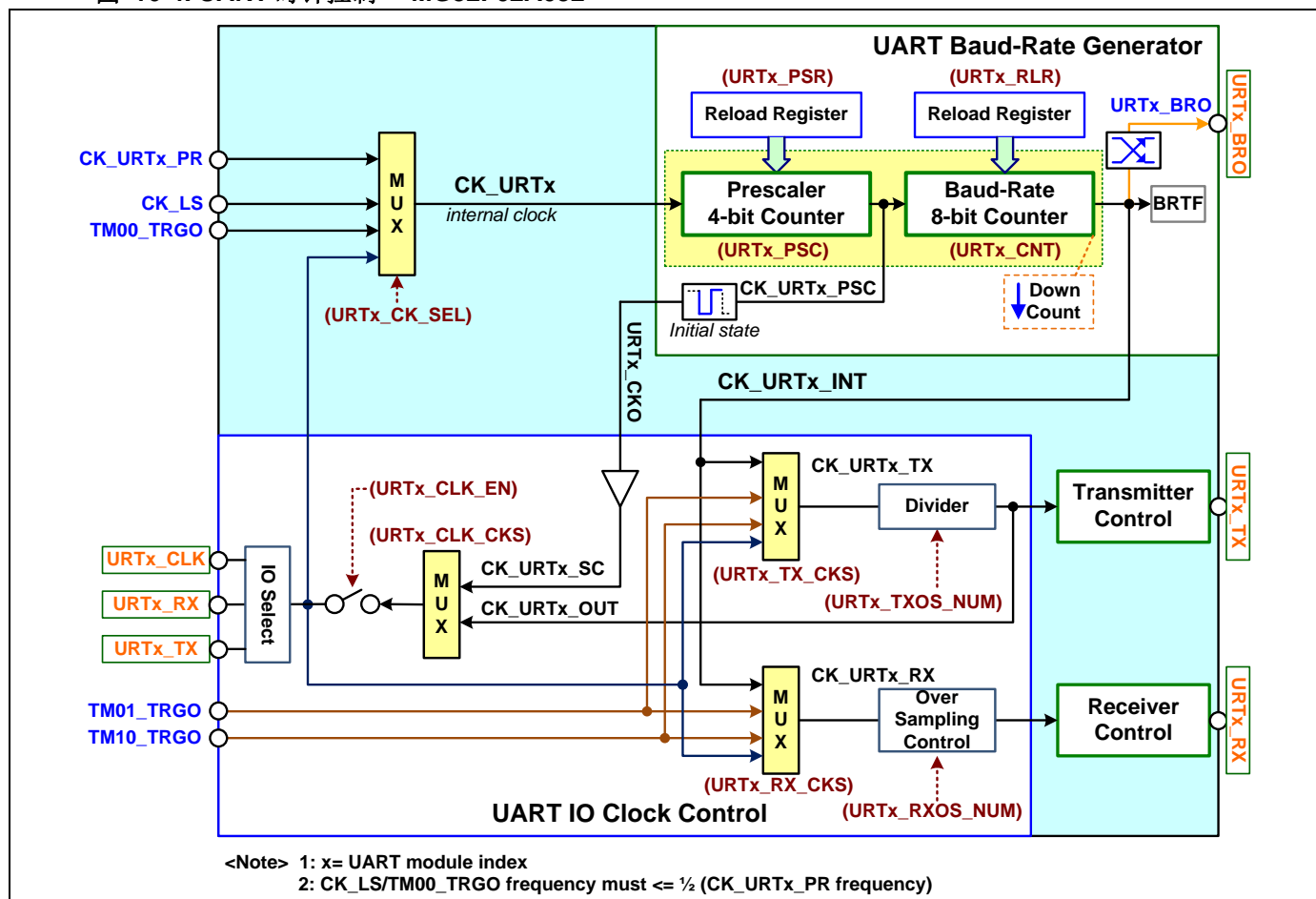


图 16-4. UART 时钟控制 – MG32F02A032



URTx_CLK 可通过 **URTx_CK_SEL** 寄存器设置当波特率发送器作为通用定时器时作为波特率发送器时钟输入。它还可以通过 **URTx_TX_CKS** 或 **URTx_RX_CKS** 寄存器输入作为 TX 分频器或 RX 过采样控制输入时钟。

[注释]: **URTx_CLK** 输入作为时钟输入不支持于 MG32F02A132/072。

16.6.3. UART PSC 超时信号输出

PSC 预分频器定时器可输出超时信号作为 **CK_URTx_PSC** 时钟信号到外部端口 **URTx_CLK**。用户可通过 **URTx_CKO_STA** 寄存器设置超时信号的初始值。特别是，只有当 **URTx_CKO_LCK** 寄存器被同时写 1 时，该初始状态寄存器才能被写入。

16.7. 中断和事件

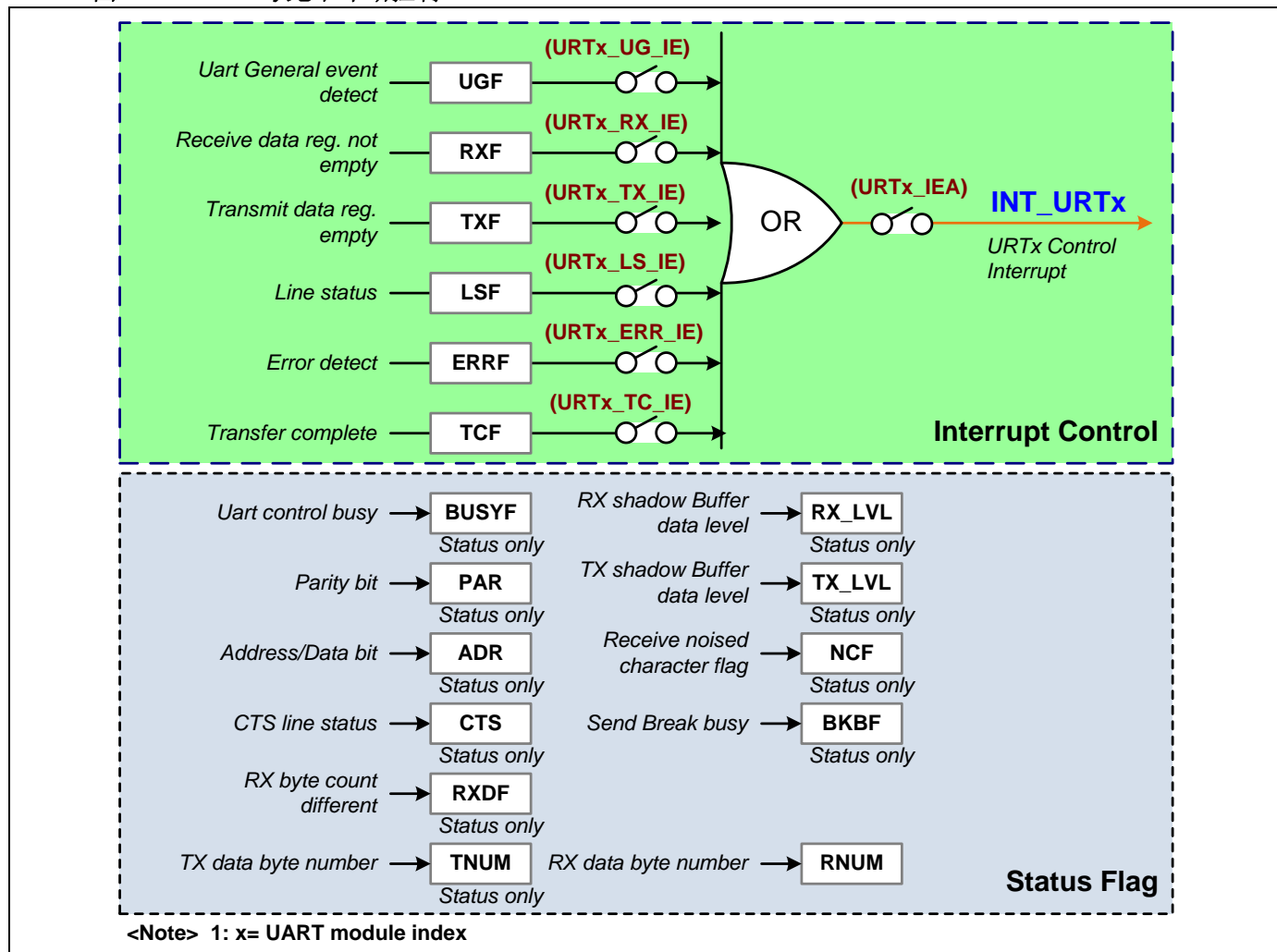
UART 模块中有 2 种信号 **INT_URTx**。**INT_URTx** 发送到外部中断控制器（EXIC）作为中断事件。

16.7.1. UART 中断控制和状态

中断标识是用于中断服务程序（ISR）流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **URTx_IEA** 用于使能和禁用该模块的所有中断源。

该模块中有一些只读的状态位，用于提供内部控制状态，其中的占用标志 **URTx_BUSYF** 表明数据传输繁忙状态。当检测到合法的起始位，该位会被置起并在停止位后清除。参照相关状态位寄存器描述以获取更多信息。

图 16-5. UART 状态和中断控制 – URT0/1/2/3

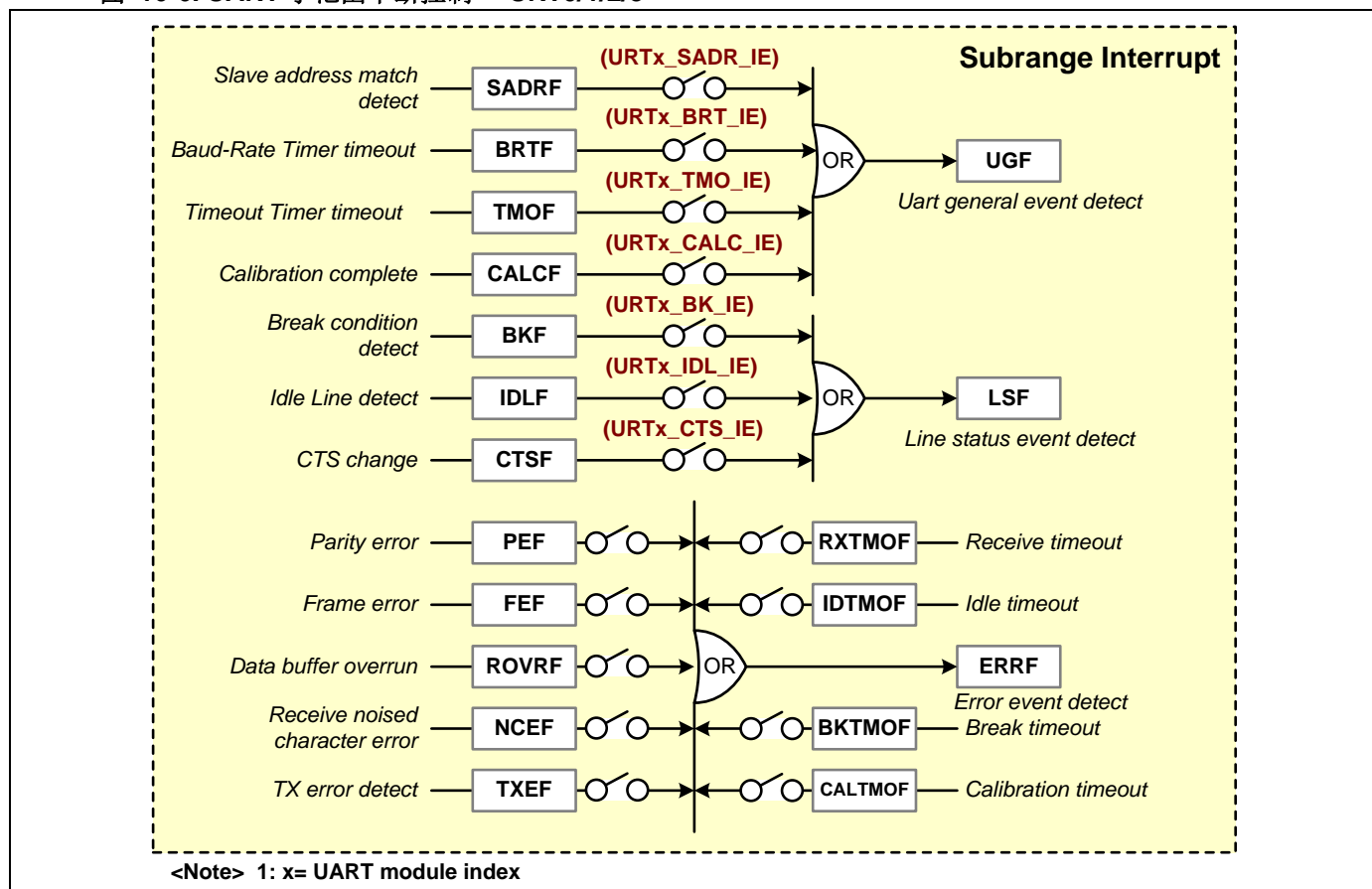


16.7.2. UART 子范围中断

在中断标志中有一些子范围中断标志 **URT_x_UGF**, **URT_x_LSF** 和 **URT_x_ERRF**。

下面的图表展示了 UART 子范围中断控制块。

图 16-6. UART 子范围中断控制 – URT0/1/2/3



16.7.3. UART 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

● UGF

UART 通用事件标志是(**URT_x_UGF**)，相关的中断使能寄存器位是 **URT_x_UG_IE**。该标志表明当该标志被置起时，**URT_x_SADRf**, **URT_x_BRTf**, **URT_x_TMOF** 或 **URT_x_CALCF** 其中的标志被置起了。

● TCF

UART 发送完成标志是(**URT_x_TCF**)。当阴影缓冲和数据寄存器都是空的且移动缓冲以完成移出，则置起该标志。相关的中断使能寄存器位是 **URT_x_TC_IE**。

● ERRF

UART 错误中断标志(**URT_x_ERRF**)，相关的中断使能寄存器位是 **URT_x_ERR_IE**。它表明了发生校验错误、帧错误、溢出错误、接收超时或噪声错误。

● LSF

UART 通用事件标志是(**URT_x_LSF**)，相关的中断使能寄存器位是 **URT_x_LS_IE**。它表明了任何一种中止状态、空闲、CTS 检测的线路状态标志。

● RXF

UART 接收数据寄存器不为空标志是(**URT_x_RXF**)，相关的中断使能寄存器位是 **URT_x_RX_IE**。当接收数据缓冲等级 **URT_x_RX_LVL** 大于等于阴影缓冲阈值 **URT_x_RX_TH** 设置，该位会被置起，数据缓冲内容被复制到数据寄存器 **URT_x_RDAT** 中。当 **URT_x_RDAT** 被读或该位被软件写 1 时清除。

注释: 当 SWD 调试时, URT_x RDAT 被读取, RXF 标志也不会被清除。

- **TXF**

UART 发送数据寄存器为空标志是(**URTx_TXF**)，相关的中断使能寄存器位是 **URTx_TX_IE**。当发送过程中阴影缓冲为空时，数据寄存器会复制到阴影缓冲并置起该标志。当 **URTx_TDAT** 被写或该位被软件写 1 时清除。

- **SADRF**

UART 从机地址匹配标志是(**URTx_SADRF**)。该标志用于多处理器模式。相关的中断使能寄存器位是 **URTx_SADR_IE**。

- **BRTF**

UART 波特率发生器定时器超时标志是(**URTx_BRTF**)，相关的中断使能寄存器位是 **URTx_BRT_IE**。

- **TMOF**

UART 超时定时器超时标志是(**URTx_TMOF**)，相关的中断使能寄存器位是 **URTx_TMO_IE**。

- **CALCF**

UART 自动波特率校准完成标志是(**URTx_CALCF**)，相关的中断使能寄存器位是 **URTx_CALC_IE**。

- **BKF**

UART 中止状态检测标志是(**URTx_BKF**)，相关的中断使能寄存器位是 **URTx_BK_IE**。参照“[UART 中止和校验/帧错误检测](#)”节以获取更多信息。

- **IDLF**

UART 空闲线路检测标志是(**URTx_IDLF**)，相关的中断使能寄存器位是 **URTx_IDL_IE**。

- **CTSF**

UART CTS 改变检测中断标志是(**URTx_CTSF**)，相关的中断使能寄存器位是 **URTx_CTS_IE**。

- **PEF**

UART 奇偶性错误标志是(**URTx_PEF**)，相关的中断使能寄存器位是 **URTx_PE_IE**。当运行于多处理器模式，校验值会包括地址位。参照“[UART 中止和校验/帧错误检测](#)”节以获取更多信息。

- **FEF**

UART 帧错误标志是(**URTx_FEF**)，相关的中断使能寄存器位是 **URTx_FE_IE**。参照“[UART 中止和校验/帧错误检测](#)”节以获取更多信息。

- **NCEF**

UART 接收噪音特征错误标志是(**URTx_NCEF**)，相关的中断使能寄存器位是 **URTx_NCE_IE**。

- **ROVRF**

UART 接收溢出标志是(**URTx_ROVRF**)，相关的中断使能寄存器位是 **URTx_ROVR_IE**。当发生接收溢出，硬件会停止接收下一个数据进入数据阴影缓冲直到该标志被清除。该标志表明下列两种情况：(1) 当 RX 阴影缓冲超过 RX 阈值且数据寄存器没有读出，若移动缓冲装入下一个数据，该标志会被置起。(2) 当校验错误、帧错误、中止检测或从机地址检测发送并导致 RX 阴影缓冲输入停止，若此时移动缓冲装入下一个数据，该标志会被置起。

- **TXEF**

UART TX 错误检测标志是(**URTx_TXEF**)，相关的中断使能寄存器位是 **URTx_TXE_IE**。参照“[UART TX 错误检测和重传控制](#)”节以获取更多信息。

- **RXTMOF**

UART 接收超时标志是(**URTx_RXTMOF**)，相关的中断使能寄存器位是 **URTx_RXTMO_IE**。参照“[UART TMO 超时控制](#)”节以获取更多信息。

- **IDTMOF**

UART 空闲状态超时标志是(**URTx_IDTMOF**)，相关的中断使能寄存器位是 **URTx_IDTMO_IE**。参照“[UART TMO 超时控制](#)”节以获取更多信息。

- **BKTMOF**

UART 中止接收超时标志是 (**URTx_BKTMOF**)，相关的中断使能寄存器位是 **URTx_BKTMO_IE**。参照“[UART TMO 超时控制](#)”节以获取更多信息。

- **CALTMOF**

UART 自动波特率校准同步区接收超时标志是(**URTx_CALTMOF**)，相关的中断使能寄存器位是 **URTx_CALTMO_IE**。参照“[UART TMO 超时控制](#)”节以获取更多信息。

16.8. UART 模块 IO 控制

该模块提供了 2 个数据信号—**URTx_RX**和**URTx_TX**，1 个时钟信号—**URTx_CLK**，2 个硬件流控制信号—**URTx_CTS**和**URTx_RTS**，1 个数据使能信号—**URTx_DE**和 1 个从机选择输出信号—**URTx_NSS**。

16.8.1. UART IO 控制

所有的 **URTx_ZZZ_INV** (ZZZ: 寄存器串) 寄存器用于反相相关信号。**URTx_IO_SWP** 寄存器用于使能调换 **URTx_RX**和**URTx_TX**信号。

图 16-7. UART RX/TX IO 控制 – MG32F02A132/072

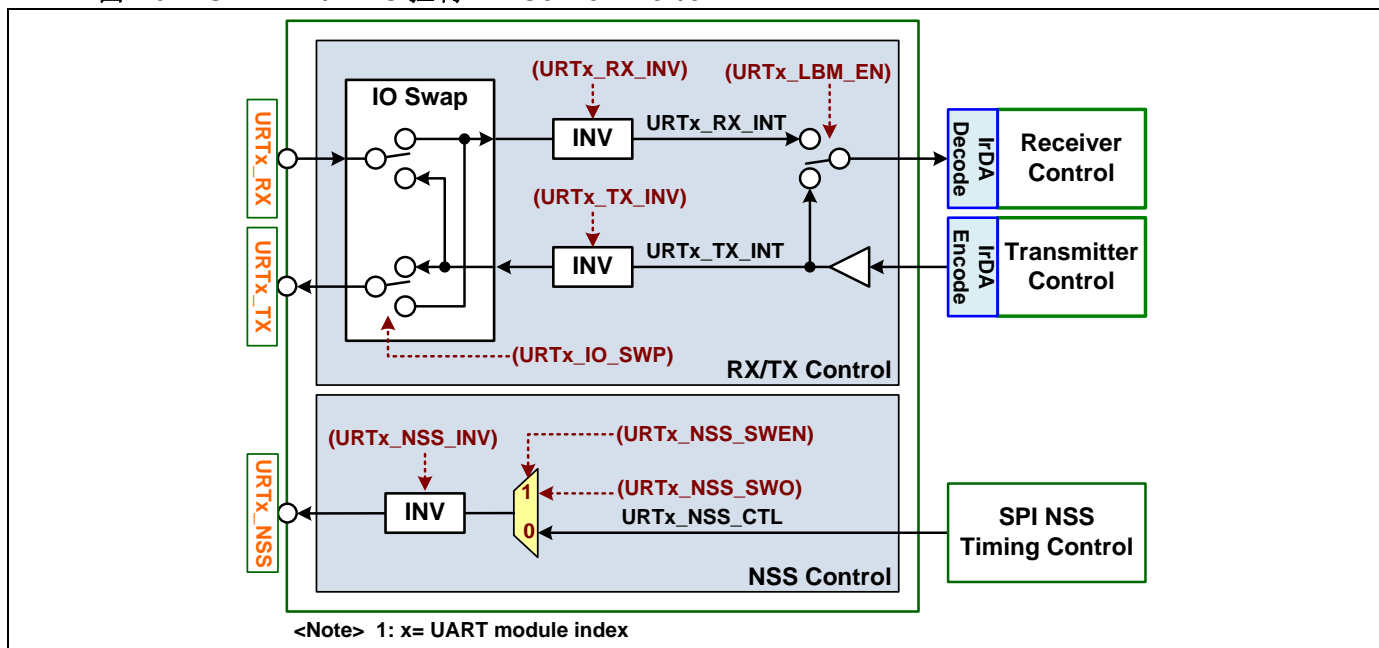
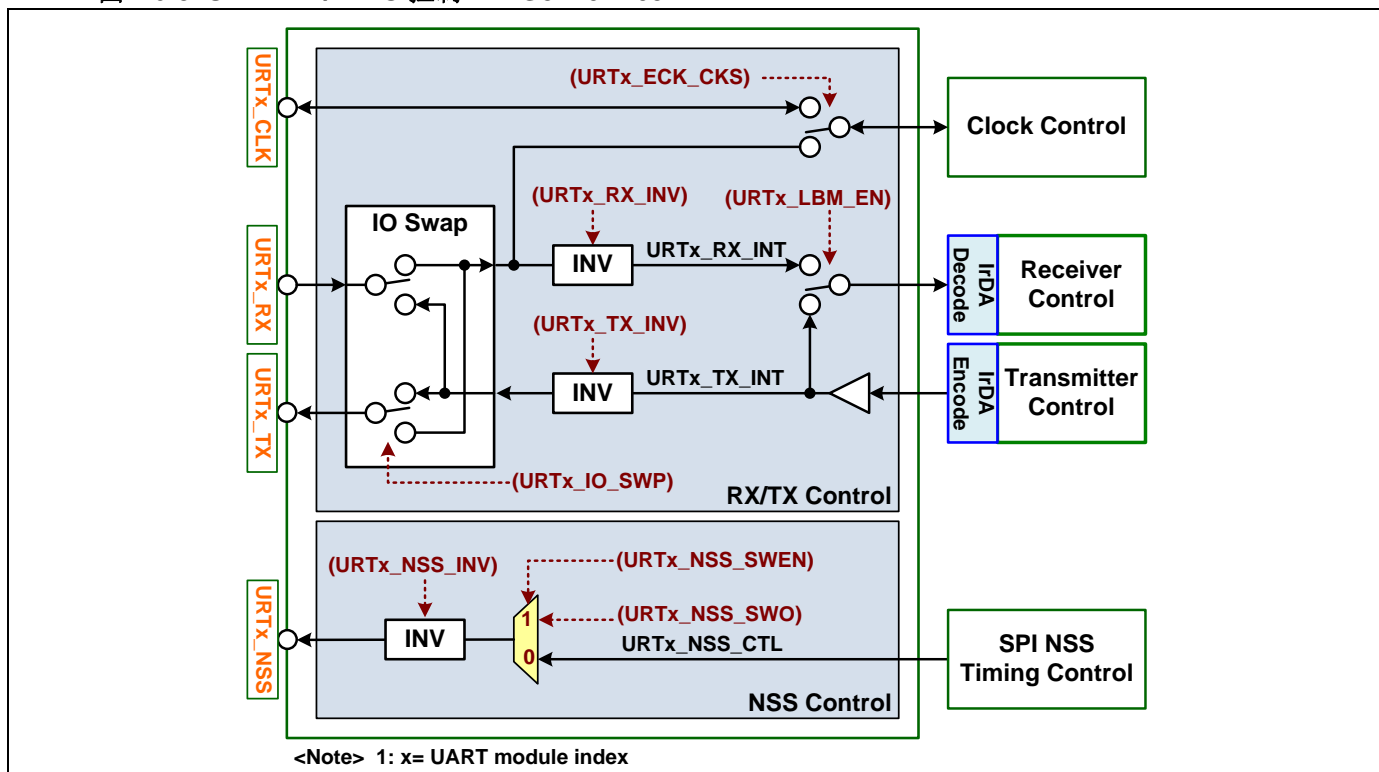


图 16-8. UART RX/TX IO 控制 – MG32F02A032



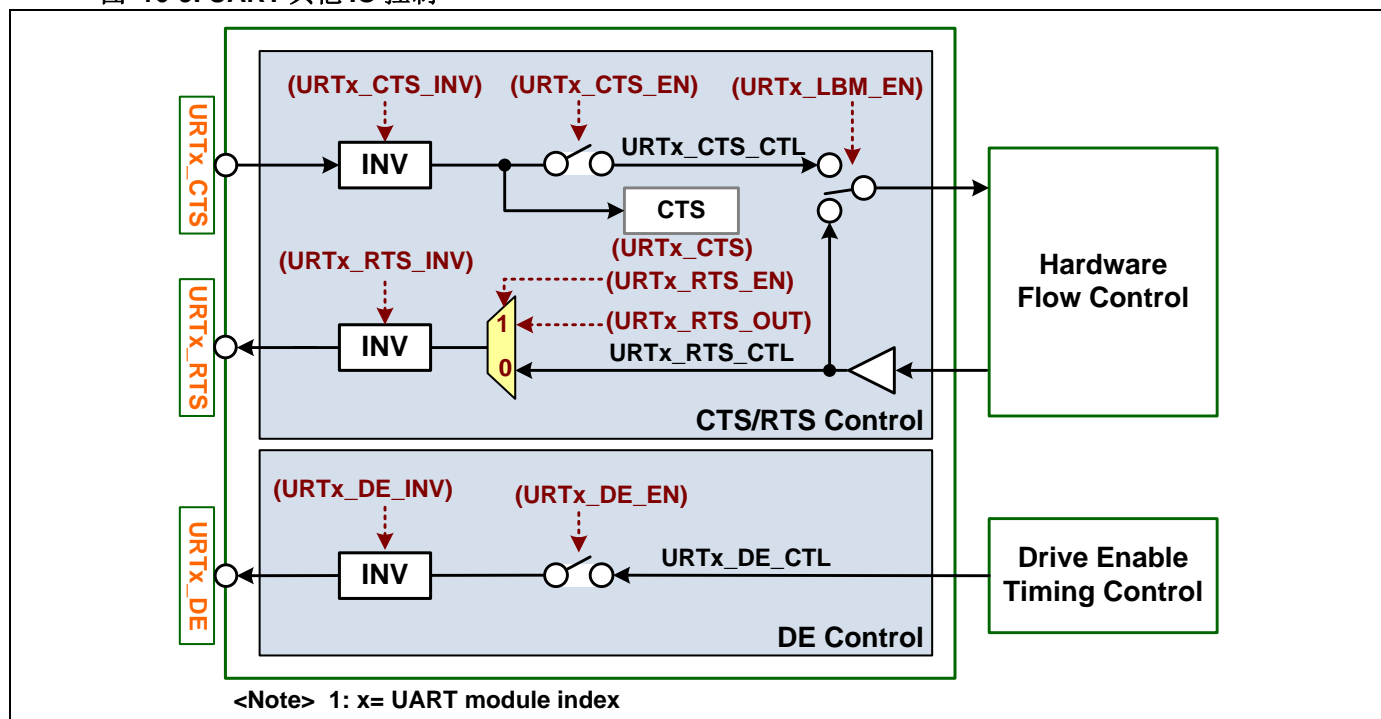
URT_x_NSS_SWEN 寄存器用于通过软件控制使能 **URT_x_NSS** 信号。当使能时，用户可通过 **URT_x_NSS_SWO** 寄存器设置直接控制输出。

URT_x_ECK_CKS 寄存器用于选择 UART 同步模式的时钟 IO 端口。通过设置 **URT_x_IO_SWP** 寄存器，该时钟可连接到 **URT_x_CLK** IO 功能或选择 **URT_x_RX** 和 **URT_x_TX** IO 功能。

[注释]: **URT_x_ECK_CKS** 寄存器不支持于 MG32F02A132/072。

URT_x_CTS_EN, **URT_x_RTS_EN** 和 **URT_x_DE_EN** 寄存器用于使能 **URT_x_CTS**, **URT_x_RTS** 和 **URT_x_DE** 信号的 IO 功能。参考“UART 硬件流控制”节以获取更多 CTS/RTS 控制的信息。

图 16-9. UART 其他 IO 控制



16.8.2. UART IO 模式

通常，用户可为 UART 输出信号设置使用的 IO 引脚的 IO 模式为推挽输出或开漏输出。此外，用户可为 UART 输入信号设置使用的 IO 引脚的 IO 模式为数字输入或开漏输出，对于 UART 双向信号，使用的 IO 引脚的 IO 模式为开漏输出。

16.8.3. UART 循环模式

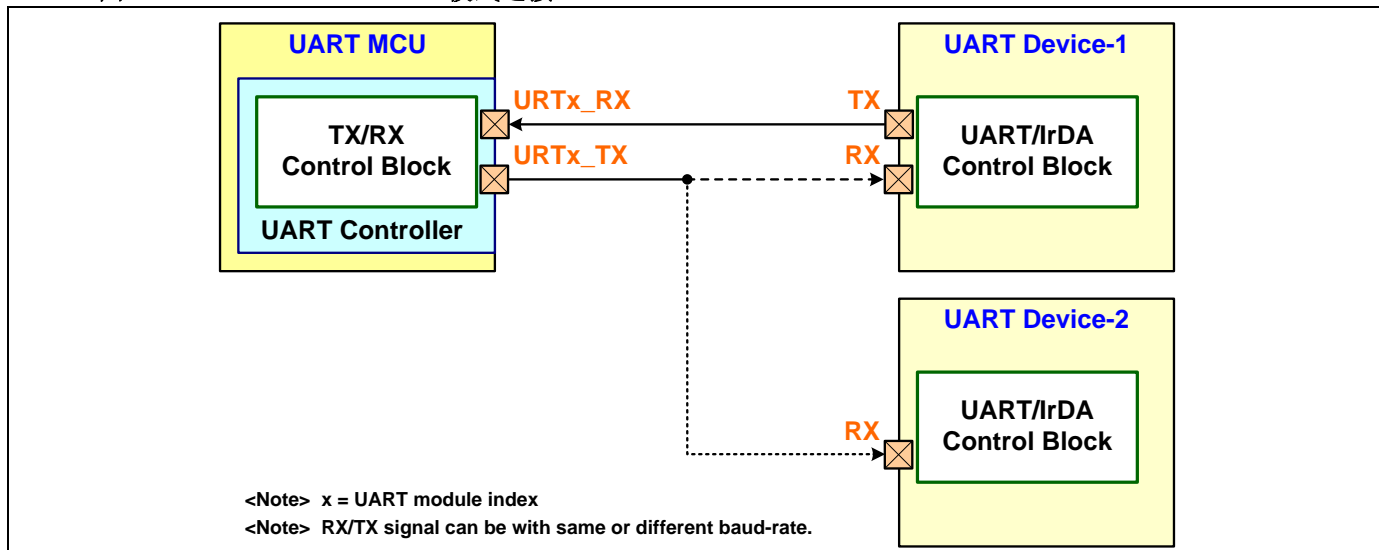
UART 模块内建 1 个循环模式通过 **URT_x_LBM_EN** 寄存器设置用于内部自测试。当使能时，接收到的输入取自发送输出，并替代 RX 输入和 CTS 输入。

16.9. UART 连接应用

下面的图表展示了 UART 连接，用于 UART/IrDA 通信、LIN 通信、智能卡通信和 SPI 主机通信。

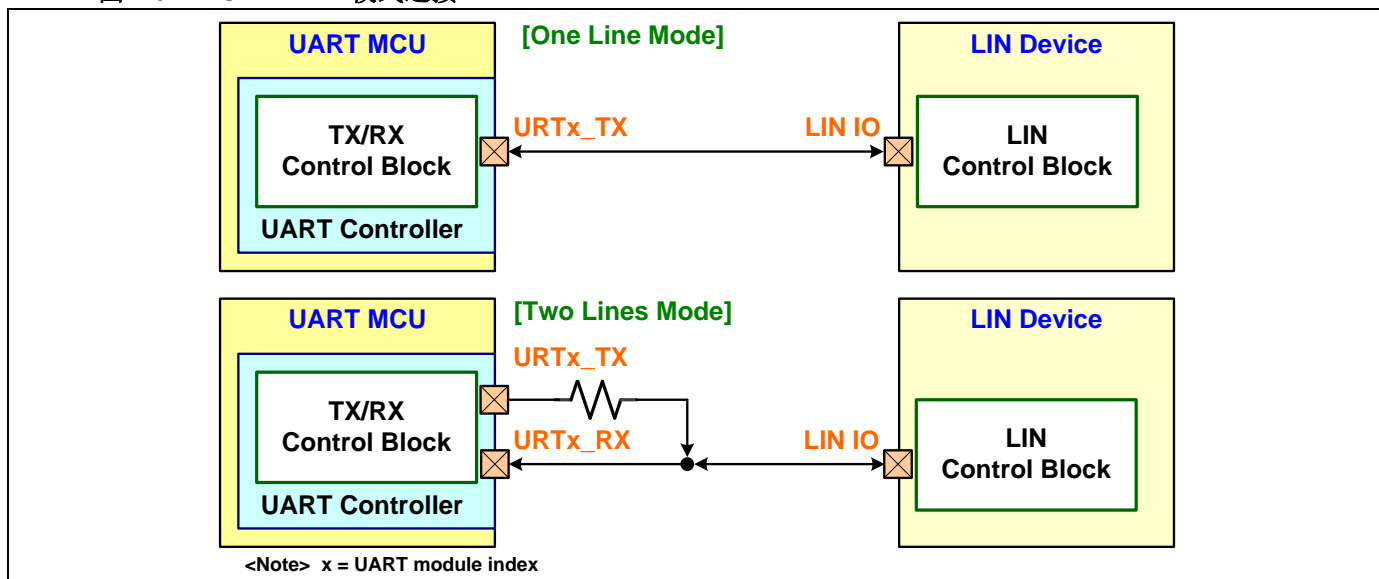
16.9.1. UART UART/IrDA 模式连接

图 16-10. UART UART/IrDA 模式连接



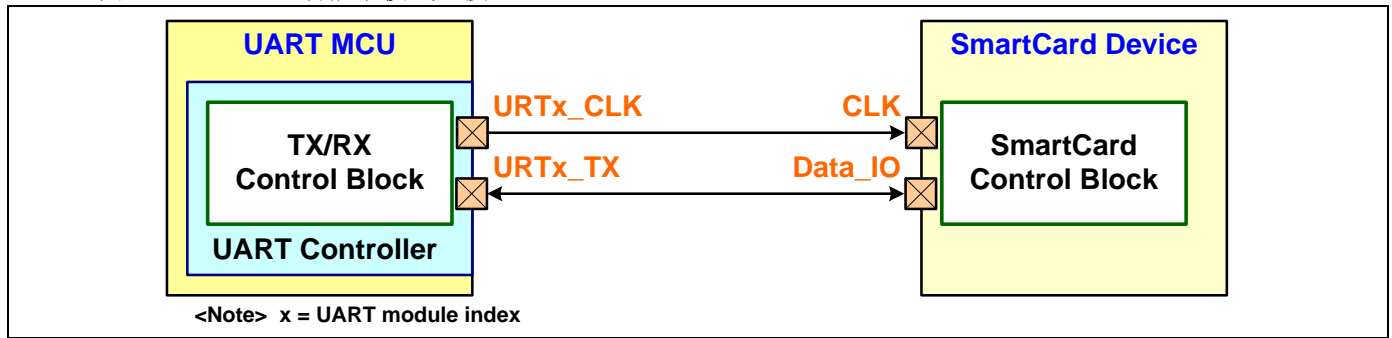
16.9.2. UART LIN 模式连接

图 16-11. UART LIN 模式连接



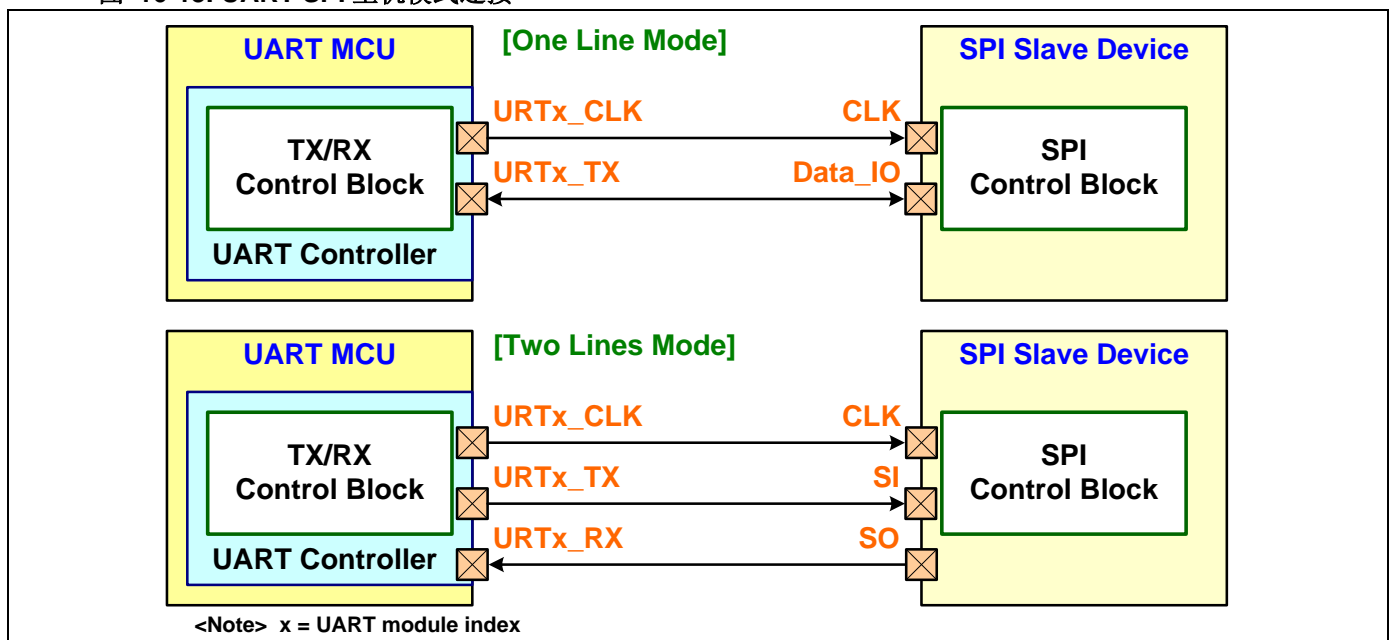
16.9.3. UART 智能卡模式连接

图 16-12. UART 智能卡模式连接



16.9.4. UART SPI 主机模式连接

图 16-13. UART SPI 主机模式连接



16.10. UART 格式

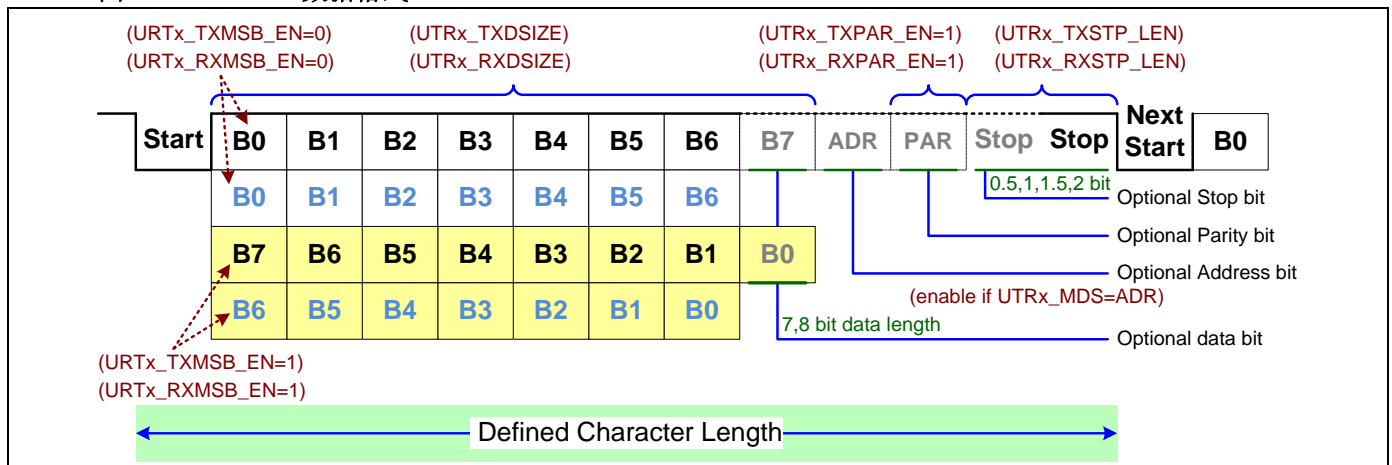
16.10.1. UART 数据格式设置

UART 字符被定义为 UART 传输的数据单元。通常，该字符会包含 1 个起始位，8 或 7 个数据位和 1 个停止位。此外，它还可以引入 1 个校验位(PAR)和 1 个地址(ADR)位用于多处理器模式。

相关的接收和发送控制寄存器是独立设定的。数据位可通过 **URT_x_TXDSIZE** 和 **URT_x_RXDSIZE** 寄存器设定为 8 或 7 位；停止位可通过 **URT_x_TXSTP** 和 **URT_x_RXSTP** 寄存器设定位长；校验位可通过 **URT_x_TXPAR_EN** 和 **URT_x_RXPAR_EN** 寄存器使能。这两个寄存器不可用于 SYNC 模式。当 UART 通过 **URT_x_MDS** 寄存器设置为“ADR” (地址位模式用于多处理器) 时，地址位会自动被硬件引入。

用户可通过 **URT_x_TXMSB_EN** 和 **URT_x_RXMSB_EN** 寄存器设置 LSB 或 MSB 方式传输数据。

图 16-14. UART 数据格式



下面的表格展示了不同数据格式的寄存器设置。

表 16-3. UART 数据格式设置

URT _x 寄存器					UART 数据格式												
TXDSIZE RXDSIZE	TXMSB_EN RXMSB_EN	MDS	TXPAR_EN RXPAR_EN	TXSTP_LEN RXSTP_LEN													
0 (8-位)	0	0~6	0	1 (1-位)	S	B0	B1	B2	B3	B4	B5	B6	B7	P			
0 (8-位)	0	0~6	0	3 (2-位)	S	B0	B1	B2	B3	B4	B5	B6	B7	P	P		
0 (8-位)	0	0~6	1	1 (1-位)	S	B0	B1	B2	B3	B4	B5	B6	B7	PAR	P		
0 (8-位)	0	0~6	1	3 (2-位)	S	B0	B1	B2	B3	B4	B5	B6	B7	PAR	P	P	
0 (8-位)	0	7	0	1 (1-位)	S	B0	B1	B2	B3	B4	B5	B6	B7	ADR	P		
0 (8-位)	0	7	0	3 (2-位)	S	B0	B1	B2	B3	B4	B5	B6	B7	ADR	P	P	
0 (8-位)	0	7	1	1 (1-位)	S	B0	B1	B2	B3	B4	B5	B6	B7	ADR	PAR	P	
0 (8-位)	0	7	1	3 (2-位)	S	B0	B1	B2	B3	B4	B5	B6	B7	ADR	PAR	P	P
0 (8-位)	1	0~6	0	1 (1-位)	S	B7	B6	B5	B4	B3	B2	B1	B0	P			
0 (8-位)	1	0~6	0	3 (2-位)	S	B7	B6	B5	B4	B3	B2	B1	B0	P	P		
0 (8-位)	1	0~6	1	1 (1-位)	S	B7	B6	B5	B4	B3	B2	B1	B0	PAR	P		
0 (8-位)	1	0~6	1	3 (2-位)	S	B7	B6	B5	B4	B3	B2	B1	B0	PAR	P	P	
0 (8-位)	1	7	0	1 (1-位)	S	B7	B6	B5	B4	B3	B2	B1	B0	ADR	P		
0 (8-位)	1	7	0	3 (2-位)	S	B7	B6	B5	B4	B3	B2	B1	B0	ADR	P	P	
0 (8-位)	1	7	1	1 (1-位)	S	B7	B6	B5	B4	B3	B2	B1	B0	ADR	PAR	P	
0 (8-位)	1	7	1	3 (2-位)	S	B7	B6	B5	B4	B3	B2	B1	B0	ADR	PAR	P	P
1 (7-位)	0	0~6	0	1 (1-位)	S	B0	B1	B2	B3	B4	B5	B6	P				
1 (7-位)	0	0~6	0	3 (2-位)	S	B0	B1	B2	B3	B4	B5	B6	P	P			
1 (7-位)	0	0~6	1	1 (1-位)	S	B0	B1	B2	B3	B4	B5	B6	PAR	P			
1 (7-位)	0	0~6	1	3 (2-位)	S	B0	B1	B2	B3	B4	B5	B6	PAR	P	P		

1 (7-位)	0	7	0	1 (1-位)	S	B0	B1	B2	B3	B4	B5	B6	ADR	P		
1 (7-位)	0	7	0	3 (2-位)	S	B0	B1	B2	B3	B4	B5	B6	ADR	P	P	
1 (7-位)	0	7	1	1 (1-位)	S	B0	B1	B2	B3	B4	B5	B6	ADR	PAR	P	
1 (7-位)	0	7	1	3 (2-位)	S	B0	B1	B2	B3	B4	B5	B6	ADR	PAR	P	P
1 (7-位)	1	0~6	0	1 (1-位)	S	B6	B5	B4	B3	B2	B1	B0	P			
1 (7-位)	1	0~6	0	3 (2-位)	S	B6	B5	B4	B3	B2	B1	B0	P	P		
1 (7-位)	1	0~6	1	1 (1-位)	S	B6	B5	B4	B3	B2	B1	B0	PAR	P		
1 (7-位)	1	0~6	1	3 (2-位)	S	B6	B5	B4	B3	B2	B1	B0	PAR	P	P	
1 (7-位)	1	7	0	1 (1-位)	S	B6	B5	B4	B3	B2	B1	B0	ADR	P		
1 (7-位)	1	7	0	3 (2-位)	S	B6	B5	B4	B3	B2	B1	B0	ADR	P	P	
1 (7-位)	1	7	1	1 (1-位)	S	B6	B5	B4	B3	B2	B1	B0	ADR	PAR	P	
1 (7-位)	1	7	1	3 (2-位)	S	B6	B5	B4	B3	B2	B1	B0	ADR	PAR	P	P
B0~B7 = 数据位 0~7 , S = 起始位 , P = 停止位																
ADR : 地址位(1=从机地址 , 0=数据) , PAR :校验位 (包含 B0~B7 和 ADR)																

16.10.2. UART 校验位

用户可通过 **URT_x_TXPAR_POL**, **URT_RXPAR_POL**, **URT_x_TXPAR_STK** 和 **URT_RXPAR_STK** 寄存器设置校验位功能。下面的表格展示了校验位的寄存器设置。

表 16-4. UART 校验位定义

校验功能	URT _x 寄存器		校验位值
	TXPAR_POL RXPAR_POL	TXPAR_STK RXPAR_STK	
Even	0	0	PAR = B0⊕B1⊕B2⊕B3⊕B4⊕B5⊕B6⊕B7⊕ADR
Odd	1	0	PAR = Not (B0⊕B1⊕B2⊕B3⊕B4⊕B5⊕B6⊕B7⊕ADR)
Fixed 0	0	1	PAR = 0
Fixed 1	1	1	PAR = 1
PAR : 校验位 (包含 B0~B7 和 ADR)			

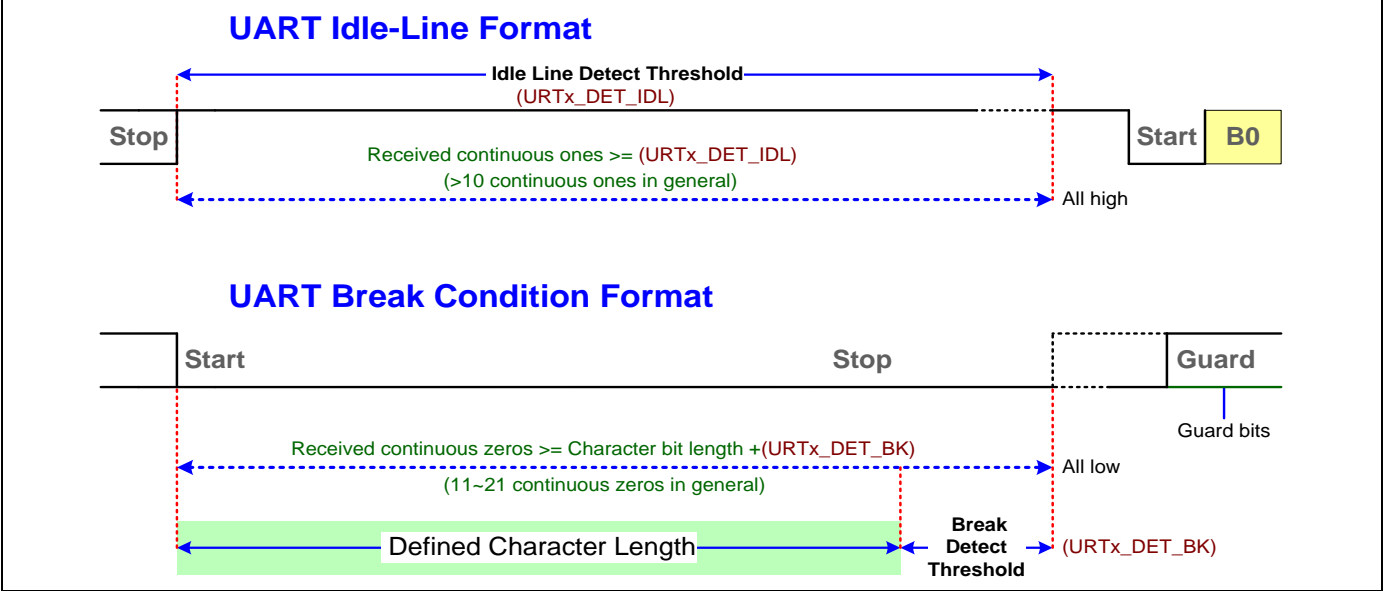
16.10.3. UART Idle 线格式设置

UART 模块可检测 Idle 线, 且用户可通过 **URT_x_DET_IDL** 寄存器设置定义从最后 1 个停止位开始的 Idle 线的位时间。

16.10.4. UART 中止状态格式设置

UART 模块可检测中止状态，且用户可通过 **URT_x_DET_BK** 寄存器设置把中止状态的位时间定义为 1 个字符长度加上额外的位时间。

图 16-15. UART Idle 线和中止状态格式



16.11. UART 基本控制

16.11.1. UART 控制模式设置

UART 模块可设置 **UART** (异步模式), **SYNC** (同步模式), **IDLE** (多处理器空闲模式)和 **ADR** (多处理器地址位模式)模式。用户可通过 **URT_x_MDS** 寄存器设置这些控制模式。

下面的表格展示了不同控制模式下的数据控制寄存器。

表 16-5. UART 控制模式对比数据控制寄存器

UART 控制模式	URT _x 数据控制寄存器					
	TXMSB_EN RXMSB_EN	TXDSIZE TXSTP_LEN	RXDSIZE RXSTP_LEN	TXPAR_EN TXPAR_POL TXPAR_STK	RXPAR_EN RXPAR_POL RXPAR_STK	TXOS_NUM RXOS_NUM
UART (异步 1/2 线)	v	v	v	v	v	3~31 (*1)
SYNC (同步 1/2 线)	v	B				3~31 (*1)
IDLE (多处理器空闲线)	v	v	v	v	v	3~31 (*1)
ADR (多处理器地址位)	v	v	v	v	v	3~31 (*1)

v: 独立 TX 或 RX 控制下可用, B: TX/RX 控制下都可用, 空白表示控制无效

*1: 值 3~31 表明实际过采样数值 4~32。

16.11.2. UART 工作模式设置

下面的表格展示了 UART 工作模式的寄存器设置。用户可根据应用设置寄存器选择工作模式。

表 16-6. UART 工作模式设置

UART 工作模式	URT _x 设置寄存器									
	MDS	DAT_LINE	HDX_EN	CLK_EN	CLK_CKS	TXE_MDS	RXE_MDS	TXPAR_EN RXPAR_EN	IR_EN	RTS_EN CTS_EN
UART (异步模式)	0	2	0	0	x	0	0	0/1	0	0
IrDA (红外数据协会)	0	2	0	0	x	0	0	0/1	1	0
HFC (硬件流控制)	0	2	0	0	x	0	0	0/1	0	1
LIN - 1 线 (本地内部连接网络)	0	1	1	0	x	2	0	0	0	0
LIN - 2 线 (本地内部连接网络)	0	2	1	0	x	2	0	0	0	0
SC (智能卡)	0	1	1	1	1	1	1	1	0	0
SYNC (同步 1 线)	1	1	0	1	0	0	0	0	0	0
SPI (同步 2 线)	1	2	0	1	0	0	0	0	0	0
IDLE (多处理器 Idle 线)	2	2	0	0	x	0	0	0/1	0	0
ADR (多处理器地址位)	3	2	0	0	x	0	0	0/1	0	0

x: 不影响, 0/1: 可为 0 或 1

16.11.3. UART 发送

当检测到 TXF(**URT_x_TXF**)标志时, 用户可写发送数据到 TX 数据寄存器(**URT_x_TDAT**), 芯片会自动清除 TXF 标志。然后, 当再次检测到 TXF 标志时, 用户可写下一次的数据同样到 TX 数据寄存器(**URT_x_TDAT**)中, 重复该操作直到数据发送完成。

当 UART 运行于单线双向模式，TCF 标志(**URTx_TCF**)可表示之前的数据发送已完成。用户可通过 **URTx_RX_EN** 寄存器改变数据发送方向为接收。

用户可读 **URTx_TNUM** 寄存器以获取实时数据寄存器剩余的数据字节数量。这通常用于计算当错误发生时最后发送数据长度。

用户可通过 **URTx_TXSTP_LEN** 寄存器为 UART 数据发送设置停止位宽度。实际的停止位宽度会被 **URTx_TXOS_NUM** 寄存器中的发送过采样设置数影响。

下面的表格展示了 UART TX 停止位设置。

表 16-7. UART 工作模式设置

停止位需求	URTx 寄存器		实际停止位宽度	芯片支持		注释
	TXSTP_LEN	TXOS_NUM		MG32F02A132 MG32F02A072	MG32F02A032	
0.5-位	0	3,5,7, ~ , 31	0.5 位		V	TX 过采样数的偶数
		4,6,8, ~ , 30	0.5 位 + 1 CK_URTx_TX 时钟时间			TX 过采样数的奇数
1-位	1	3 ~ 31	1 位	V	V	
1.5-位	2	3,5,7, ~ , 31	1.5 位		V	TX 过采样数的偶数
		4,6,8, ~ , 30	1.5 位 + 1 CK_URTx_TX 时钟时间			TX 过采样数的奇数
2-位	3	3 ~ 31	2 位	V	V	

注释：TX 过采样数 = TXOS_NUM+1

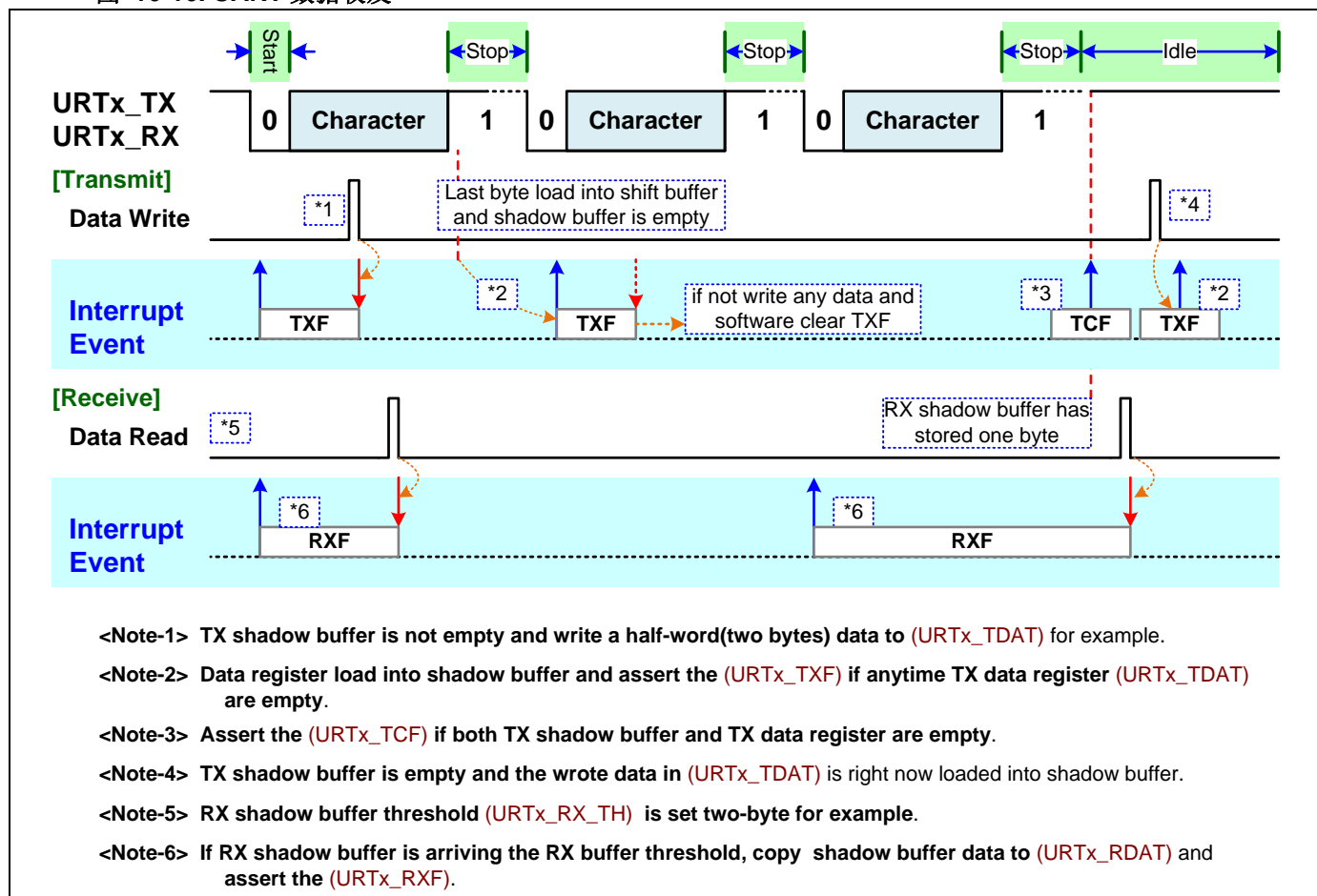
16.11.4. UART 接收

与数据发送相同，当检测到 RXF(**URTx_RXF**)标志时，用户可从 RX 数据寄存器(**URTx_RDAT**)读取接收到的数据，芯片会自动清除 RXF 标志。同样的，当再次检测到 RXF 标志时，用户可从 RX 数据寄存器(**URTx_RDAT**)读取接收到的数据，重复该操作直到数据接收完成。

URTx_RX_TH 寄存器用于设置阴影缓冲的数据字节阈值。当阴影缓冲的数据字节数等于阈值时，硬件会把阴影缓冲的内容复制到数据寄存器。

用于末尾数据控制，用户可检查 RXDF 标志(**URTx_RXDF**)和 **URTx_RNUM** 寄存器。RXDF 标志表明接收到的数据字节数量与前一次接收到的数据字节数不同。当数据阴影缓冲最后一次传输到 **URTx_RDAT** 寄存器时 **URTx_RNUM** 寄存器标识接收数据字节数量。固件可为接收数据字节数比较而写初始值。

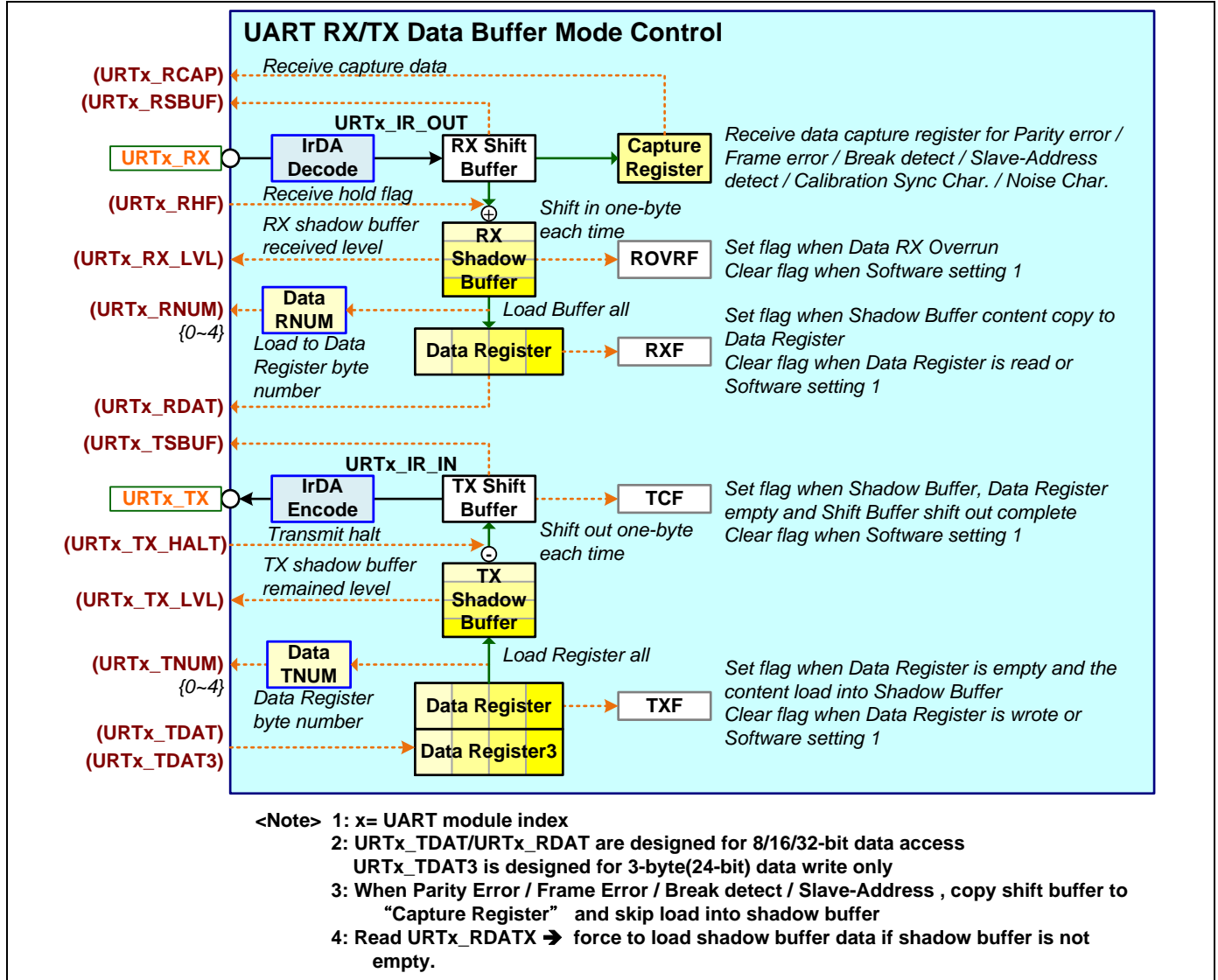
图 16-16. UART 数据收发



16.12. UART 数据缓冲

UART 模块包含 2 个 8 位移动缓冲, 2 个 32 位阴影缓冲和 2 个 32 位数据寄存器用于数据流控制和减少 CPU 开销。用户可直接使用 TXF (**URT_x_TXF**)和(**URT_x_RXF**)事件标志进行数据流控制。下面的图表展示了 UART 数据缓冲模式控制块。

图 16-17. UART 数据模式控制 – URT0/1/2/3



16.12.1. UART 数据缓冲控制

RXF 标志(**URT_x_RXF**)会在每次 RX 阴影缓冲内容复制到 RX 数据寄存器中时置起。此外若 **URT_x_RX_IE** 寄存器被使能, 相关的中断会被置起。RHF 标志(**URT_x_RHF**)是被一些数据停止触发事件产生并用于停止将接收到的数据存入阴影缓冲的。参照“UART 接收硬件停止和捕获控制”节以获取更多信息。

TXF 标志(**URT_x_TXF**)会在每次 TX 数据寄存器内容复制到 TX 阴影缓冲时置起。此外若 **URT_x_TX_IE** 寄存器被使能, 相关的中断会被置起。当阴影缓冲和 TX 数据寄存器为空, 若移动缓冲完成移出, TCF(**URT_x_TCF**)标志会被置起。用户可设置 **URT_x_TX_HALT** 寄存器暂停数据传输作为软件数据流控制。

数据寄存器 **URT_x_TDAT** 和 **URT_x_RDAT** 被设计为 8/16/32 位数据访问, 且寄存器 **URT_x_TDAT3** 被设计为只可进行 3 字节 (24 位) 写操作。当用户写任意一个 8/16/32 位数据到 **URT_x_TDAT3** 寄存器中, 芯片都会当做 3 字节 (24 位) 数据写入。

16.12.2. UART 数据反相

用户可通过 **URT_x_RDAT_INV** 或 **URT_x_TDAT_INV** 寄存器使能反相收发的数据位,将数据位从 0 到 1 或从 1 到 0。当使能该数据反相功能时,收发的数据为会被反相,但是起始、停止、地址和校验位不会被反相。

16.12.3. UART 移动缓冲

UART 模块包含 2 个独立的 8 位移动缓冲用于数据收发。**URT_x_RSBUF** 和 **URT_x_TSBUF** 寄存器可被读以获取收发移动缓冲的实时数据位。此外用户可读 **URT_x_ADR** 和 **URT_x_PAR** 寄存器位以获取收发移动缓冲的实时地址和校验位。

16.12.4. UART 数据缓冲清除

用于固件数据控制,用户可通过 **URT_x_RDAT_CLR** 或 **URT_x_TDAT_CLR** 寄存器设置强制清除收发数据缓冲。这两个寄存器位是被软件设置,被硬件清除的。

当使能了 **URT_x_RDAT_CLR**,接收数据寄存器和阴影缓冲会被清除,此外 **URT_x_RXF** 标志和 **URT_x_RX_LVL** 会被清除。当使能了 **URT_x_TDAT_CLR**,发送数据寄存器和阴影缓冲会被清除,此外 **URT_x_TX_LVL** 会被清除, **URT_x_TXF** 标志会被置起。

16.13. UART 多处理器

16.13.1. UART 多处理器工作模式

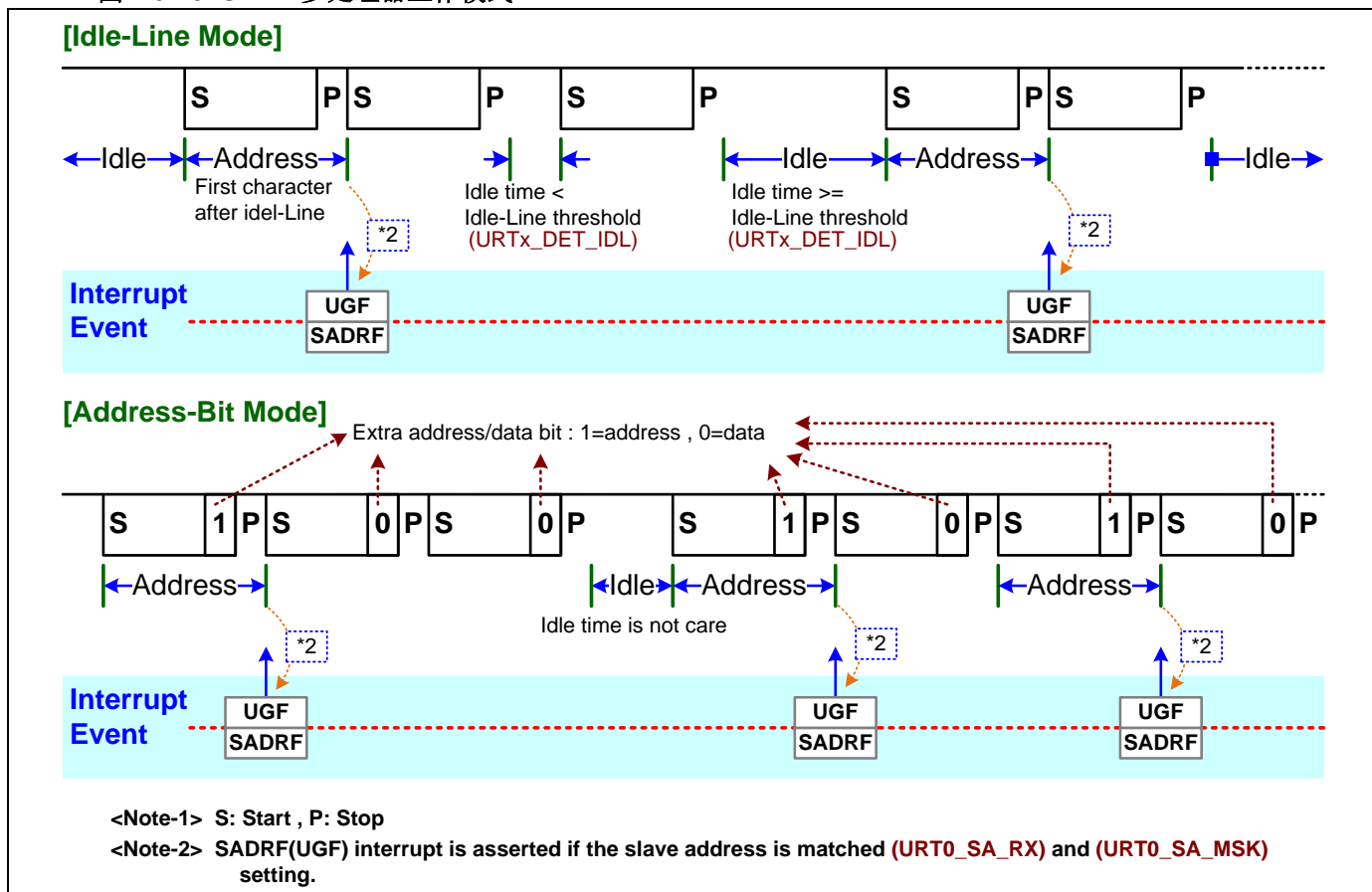
UART 模块通过 **URT_x_MDS** 寄存器设置包含 Idle 线或地址位模式用于多处理器工作模式。

当 Idle 线模式接收到地址字符或在地址位模式接收到地址位时，若相关中断使能寄存器位 **URTx_SADR_IE** 被使能，**SADRF** 标志(**URTx_SADRF**)会被置起并引发中断。

从机地址是通过 **URT_x SA RX** 和 **URT_x SA MSK** 寄存器定义的。

下面的图表展示了 UART 多处理器工作模式的时序。

图 16-18. UART 多处理器工作模式



16.13.2. UART 多处理器和静音模式

当 **UART** 运行于多处理器模式，用户可通过寄存器设置当接收到从机地址不匹配时进入静音模式并在接收到匹配的从机地址时退出静音模式。参照“**UART 静音模式控制**”的描述以获取更多关于静音模式的信息。

16.13.3. UART 多处理器地址设置

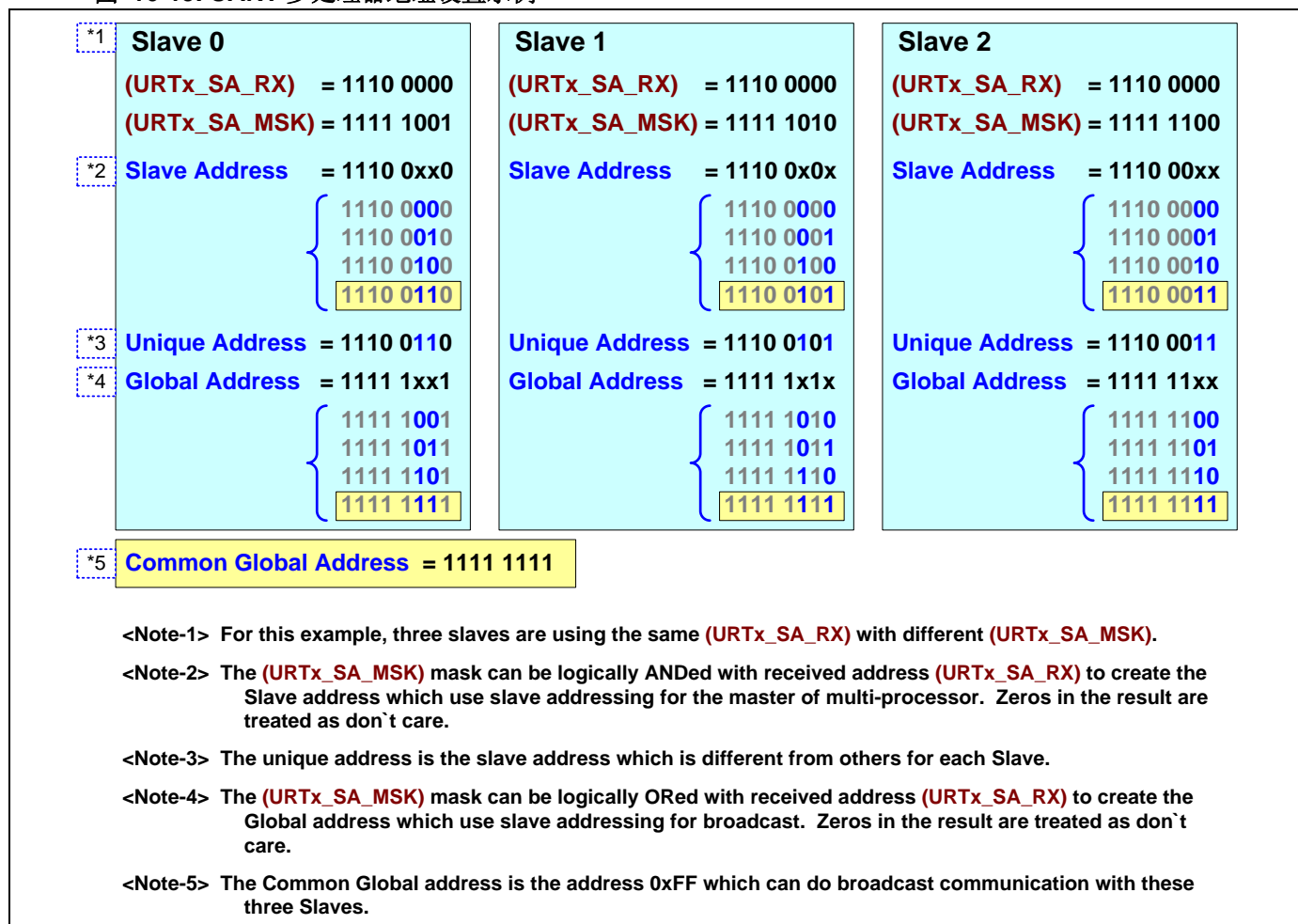
用户可通过 **URT_x SA RX** 寄存器设置接收地址，并可通过 **URT_x SA MSK** 寄存器设置地址掩码。

下面的图表展示了 UART 多处理器从机地址设置的示例。

在该示例中，使用了 3 个从机，这 3 个从机使用了相同的从机地址(**URT_x_SA_RX**)，但是使用不同的地址掩码(**URT_x_SA_MSK**)。地址掩码寄存器 **URT_x_SA_MSK** 可与接收地址寄存器 **URT_x_SA_RX** 进行逻辑“与”操作产生最终的**从机地址**用于多处理器的主机进行从机寻址。结果 0 则被视为不用在意。

从机地址是独一无二的，与其他从机都是不同的。地址掩码寄存器 **URTx_SA_MSK** 可和接收地址寄存器 **URTx_SA_RX** 进行逻辑“或”操作产生最终的**全局地址**用于广播从机寻址。一般全局地址为 **0xFF**，可与这 3 个从机进行广播通信。多处理器全局从机地址可在 **URTx_GSA_EN** 寄存器中使能。

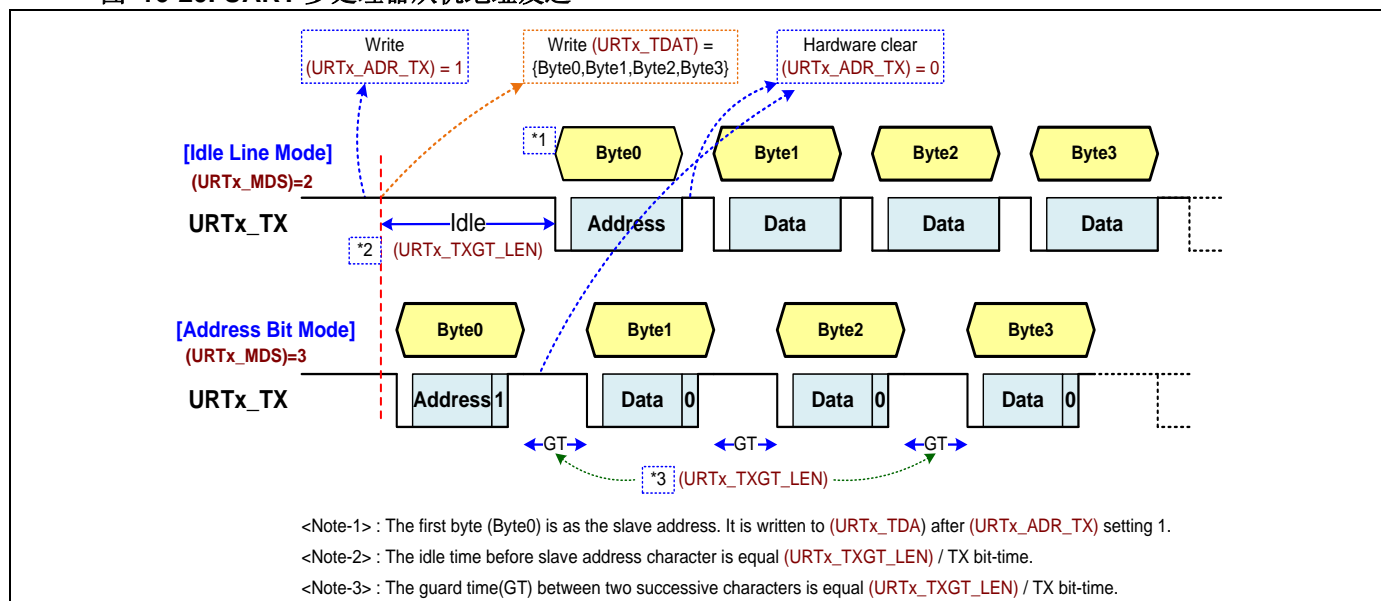
图 16-19. UART 多处理器地址设置示例



16.13.4. UART 多处理器从机地址发送

用户可通过设置 **URTx_ADR_TX** 寄存器为 1 设置发送地址字符用于 Idle 线模式或发送地址位用于地址位模式。对于 Idle 线模式，用户可通过 **URTx_TXGT_LEN** 寄存器设置 Idle 线时间。

图 16-20. UART 多处理器从机地址发送

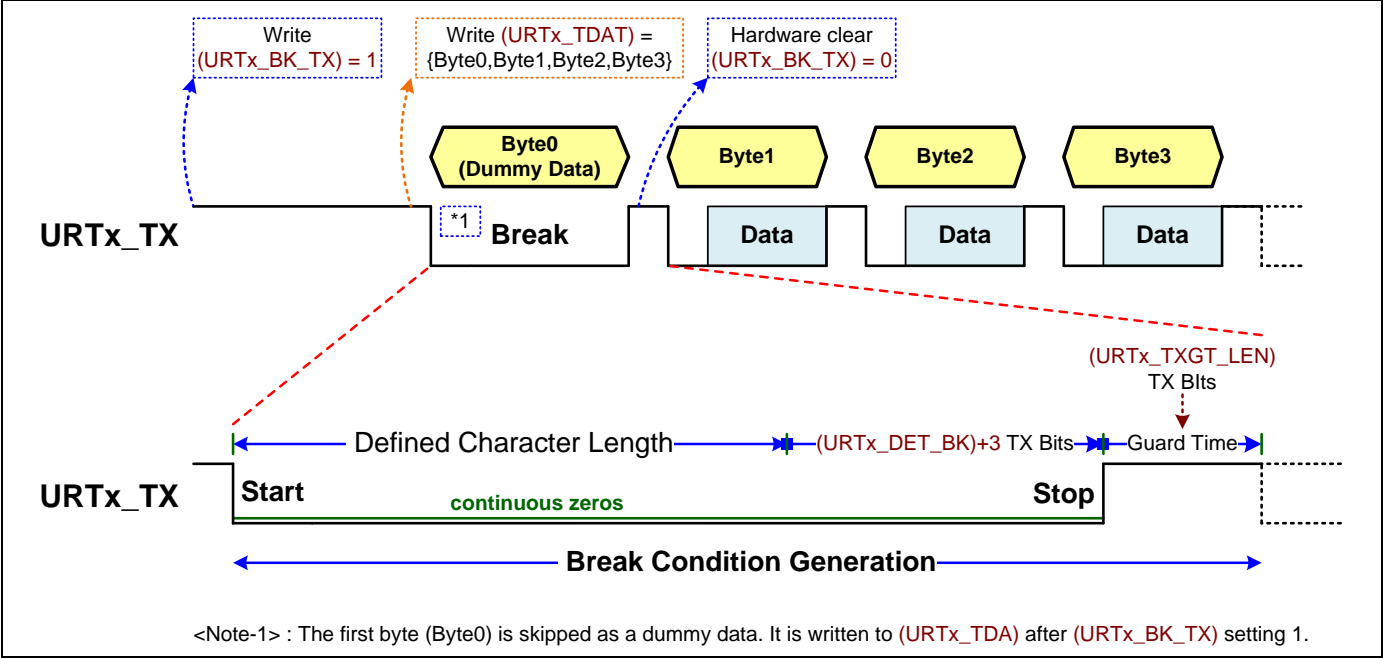


16.14. UART 中止状态发送

用户可通过 **URT_x_BK_TX** 寄存器设置为 1 发送中止状态字符。标志 **URT_x_BKBF** 为可读的，用于标识中止发送操作的占用状态。当该位为“1”，表明中止发送操作还没结束。为“0”则是已结束。

中止状态字符时间等于字符位时间加（**URT_x_DET_BK** 寄存器设置值+3）。**URT_x_TXGT_LEN** 可用于设置两个发送字符之间的保持时间。

图 16-21. UART 中止状态发送



下面的表格展示了 UART 中止状态发送和检测控制设置表。

表 16-8. UART 中止状态发送和检测控制

控制模式	URT _x 寄存器			中止发送长度/ BKF 标志置起位置
	DET_BK	TXSTP_LEN	RXSTP_LEN	
发送中止状态(*1)	0	0,1,2,3	-	中止长度= 14 bits
	1	0,1,2,3	-	中止长度= 16 bits
检测中止状态(*2)	0	-	0,1,2	BKF 在第 10.5 位置起
			3	BKF 在第 11.5 位置起
	1	-	0,1,2	BKF 在第 12.5 位置起
			3	BKF 在第 13.5 位置起

*1：设置 TXDSIZE=8 位，TXPAR_EN=禁用，MDS= {0~2}

*2：设置 RXDSIZE=8 位，RXPAR_EN=禁用，MDS= {0~2}

16.15. UART 波特率控制

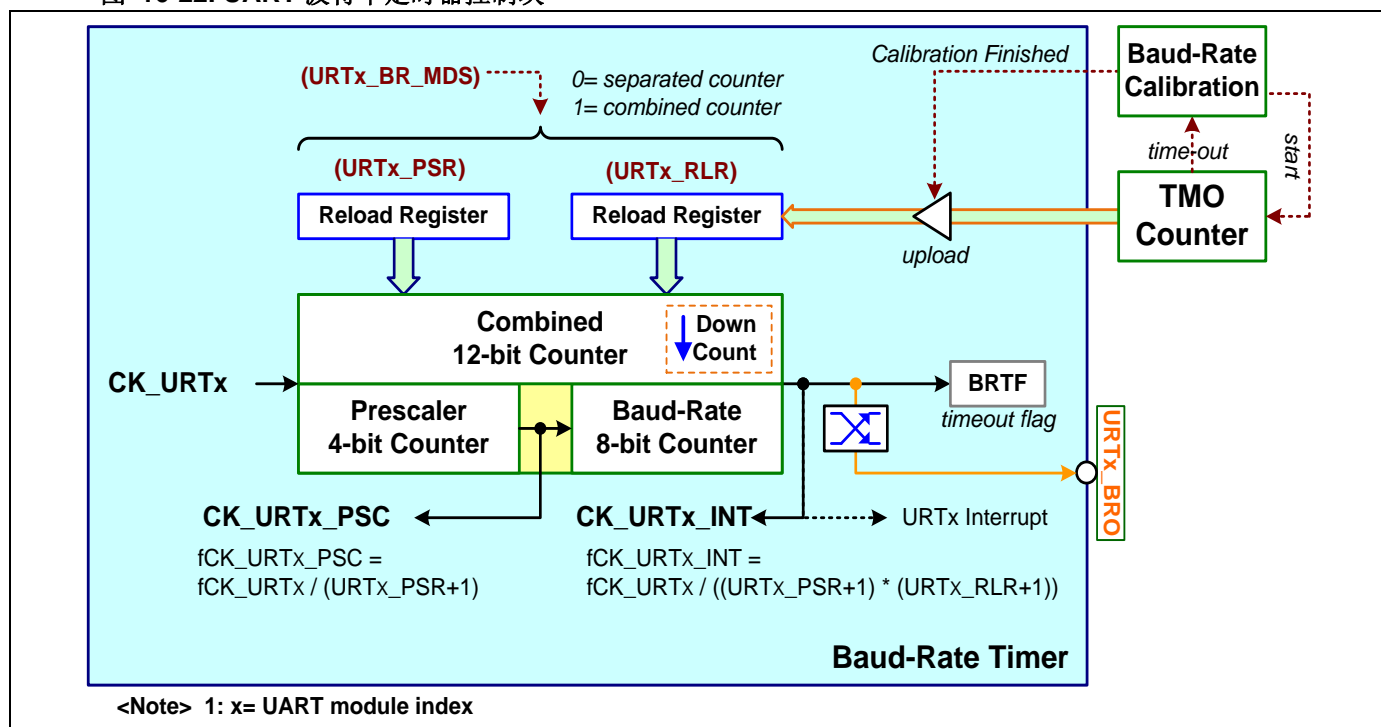
波特率定时器(BR)可通过 **URTx_BR_MDS** 寄存器被设置分离计数模式或混合计数模式作为波特率发生器或通用定时器，并通过 **URTx_BR_EN** 寄存器使能。

波特率定时器发生器可输出内部时钟用于 UART 通信波特率控制。用户可通过 **URTx_PSR** 和 **URTx_RLR** 寄存器设置波特率发生器。

此外，用户可通过读 **URTx BR PSC** 和 **URTx BR CNT** 寄存器获取实时的 BR 定时器的计数器值。

下面的图表展示了 UART 波特率定时器控制块。

图 16-22. UART 波特率定时器控制块



16.15.1. UART 波特率定时器启动/关闭控制

当 BR 定时器被设置为通用定时器时，通过 **URT_x EN** 寄存器无法使能该定时器。

下面的表格展示了 UART 波特率定时器的启动/关闭控制。

表 16-9. UART BR 波特率定时器启动/关闭控制

BR 定时器功能	计数器模式	URTx 寄存器			定时器开启/关闭
		BR_MDS	EN	BR_EN	
URTx 波特率发生器 通用定时器 BR 定时器功能	分离	0	0	x	定时器关闭
			1	0	定时器关闭
				1	定时器开启
	分离	1	0	x	定时器关闭
			1	0	定时器关闭
				1	定时器开启
	混合	1	x	0	定时器关闭
				1	定时器开启

x: 不影响

16.15.2. UART 波特率校准

UART 模块可接收中止状态时置起 BKF 标志(**URT_x_BKF**),并在相关中断使能寄存器位 **URT_x_BK_IE** 被使能时产生中断。用户可通过设置 **URT_x_CAL_EN** 寄存器使能波特率校准,当校准完成时,芯片会置起 **CALCF** 标志(**URT_x_CALCF**)。UART 波特率校准可支持分离和混合模式下的波特率定时器。

[注释]: 在 MG32F02A132/072 中, UART 波特率校准仅支持分离模式的波特率定时器。

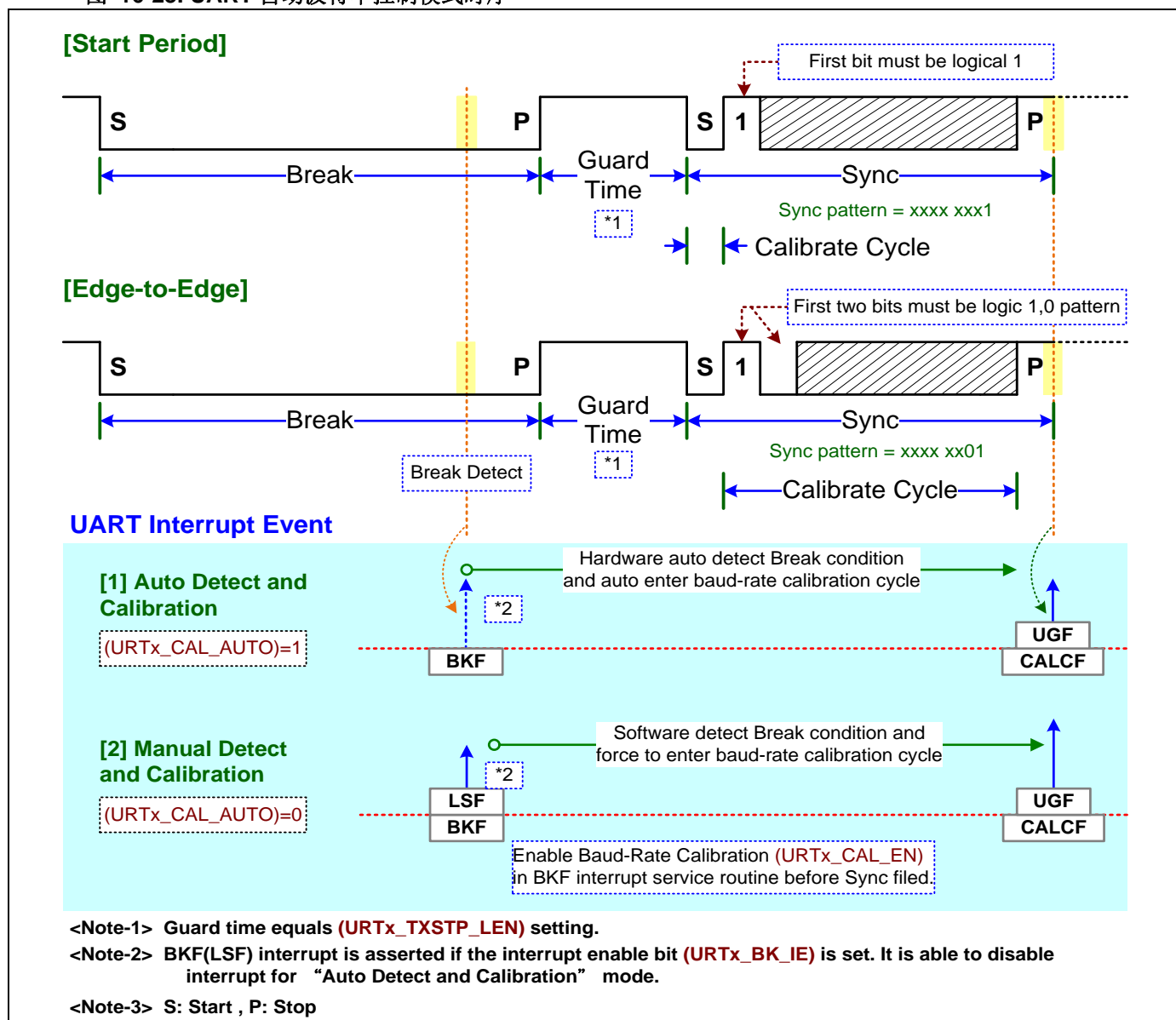
校准完成后, 通常芯片会更新波特率定时器校准重载值到 **URT_x_PSR** 和 **URT_x_RLR** 寄存器中。若校准因波特率定时器上溢或下溢而失败时, 芯片则不会更新波特率定时器重载值。用户可在校准完成后通过 **URT_x_CALOVF** 和 **URT_x_CALUDF** 寄存器标志获取校准上溢或下溢状态。

[注释]: **URT_x_CALOVF** 和 **URT_x_CALUDF** 寄存器不支持于 MG32F02A132/072。

此外, 用户可设置 **URT_x_CAL_AUTO** 寄存器以使能硬件自动波特率校准。当自动波特率校准功能被使能并接收了中止状态, 芯片会在同步字符接收周期中自动进入波特率校准操作。

UART 通过 **URT_x_CAL_MDS** 寄存器设置支持 2 种校准模式: 启动模式和边沿模式。启动模式是通过仅测量起始位来校准波特率的, 且同步字符必须为 $(xxxx\ xxx1)_2$; 边沿模式则是通过测量起始下降沿到下一个下降沿来校准波特率的, 且同步字符必须为 $(xxxx\ xx01)_2$ 。(x = 0 或 1)

图 16-23. UART 自动波特率控制模式时序



16.15.3. UART BR 超时信号输出

BR 定时器可输出超时信号作时钟信号到外部端口 **URTx_BRO** 或内部其他模块。用户可通过 **URTx_BRO_STA** 寄存器设置超时信号的初始状态。特别的是，该初始值寄存器只有在 **URTx_BRO_LCK** 寄存器同时写“1”时才能被写入。

16.15.4. UART 波特率设置示例

下面的表格展示了 UART 通用波特率设置示例。

表 16-10. UART 通用波特率设置示例

UART 波特率		CK_URTx 频率 (MHz)	URTx 寄存器			实际 波特率
波特率 (*2)	错误 (%)		PSR	RLR	OS_NUM (*1)	
1200	0.00	48.000	9	249	15	1200
2400	0.00	48.000	9	124	15	2400
4800	0.00	48.000	4	124	15	4800
9600	0.00	48.000	4	99	9	9600
14400	-0.01	48.000	2	100	10	14401.44014
19200	0.00	48.000	4	49	9	19200
28800	-0.04	48.000	6	16	13	28811.52461
38400	0.00	48.000	4	24	9	38400
57600	-0.04	48.000	6	6	16	57623.04922
115200	-0.16	48.000	1	12	15	115384.6154
230400	-0.16	48.000	1	7	12	230769.2308
460800	-0.16	48.000	1	3	12	461538.4615
576000	0.79	48.000	1	2	13	571428.5714
921600	-0.16	48.000	1	1	12	923076.9231
1152000	0.79	48.000	2	0	13	1142857.143
2000000	0.00	48.000	1	0	11	2000000
3000000	0.00	48.000	1	0	7	3000000
4000000	0.00	48.000	1	0	5	4000000
6000000	0.00	48.000	1	0	3	6000000
1200	-0.01	8.000	2	201	10	1200.120012
2400	-0.01	8.000	2	100	10	2400.240024
4800	-0.04	8.000	6	16	13	4801.920768
9600	-0.04	8.000	6	6	16	9603.841537
14400	0.08	8.000	0	138	3	14388.48921
19200	-0.16	8.000	1	12	15	19230.76923
28800	0.44	8.000	2	2	30	28673.83513
38400	-0.16	8.000	0	12	15	38461.53846
57600	-0.64	8.000	1	2	22	57971.01449
115200	-0.64	8.000	0	2	22	115942.029
230400	0.79	8.000	0	4	6	228571.4286
576000	0.79	8.000	0	0	13	571428.5714
1152000	0.79	8.000	0	0	6	1142857.143

*1 : OS_NUM = URTx_RXOS_NUM 或 URTx_TXOS_NUM 寄存器值

*2 : 波特率 = $f(\text{CK_URTx}) / (\text{PSR} + 1) / (\text{RLR} + 1) / (\text{OS_NUM} + 1)$

16.16. UART 数据接收和采样

UART 接收者数据采样模式可通过 **URT_x_OS_MDS** 寄存器设置选择 3 样本或 1 样本的过采样多数投票。过采样数可通过 **URT_x_RXOS_NUM** 寄存器进行设置。

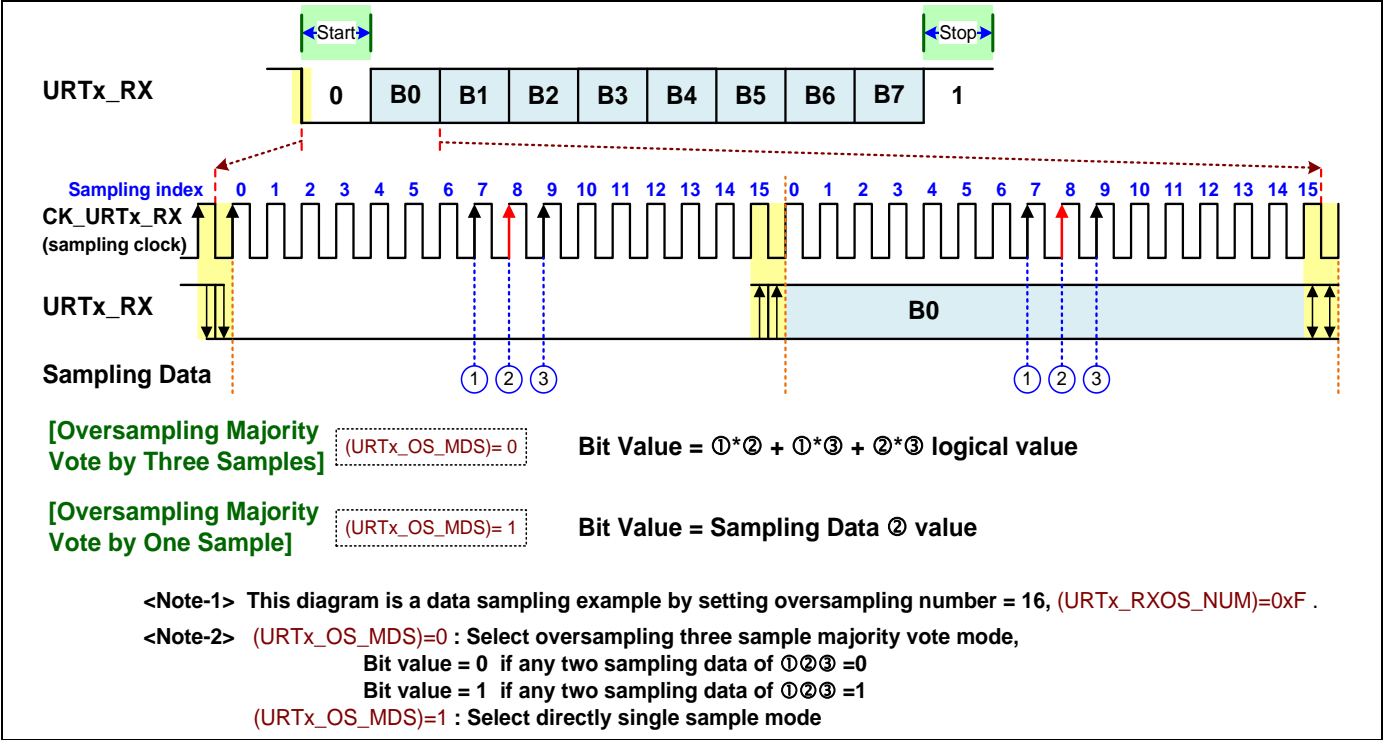
16.16.1. UART 数据采样

用户可通过 **URT_x_RXOS_NUM** 寄存器选择有效的过采样数用于数据采样。

URT_x_OS_MDS 寄存器中默认的过采样多数投票为 3 样本。UART 模块从这 3 个采样值决定接收位的值。此外，用户通过 **URT_x_OS_MDS** 寄存器可选择过采样多数投票为 1 样本，但是这样会导致传输抗噪音能力较差。通常该设置是用于提升时钟速率和通信速度的。

下面的图表展示了 UART 过采样数为 16 的接收数据采样示例。

图 16-24. UART 接收数据采样



下面的表格展示了 3 样本模式的接收位值和噪音位状态标志 **NCF** (**URT_x_NCF**)。若 2 个样本都是“0”的话，接收位为“0”；若 2 个样本都是“1”的话，接收位为“1”。

表 16-11. UART 接收数据过采样和噪音检测

采样序列值	接收位		NCF 状态	
000	0	任意两个 =0	0	所有样本 =0
001	0	任意两个 =0	1	
010	0	任意两个 =0	1	
011	1	任意两个 =0	1	
100	0	任意两个 =0	1	
101	1	任意两个 =0	1	
110	1	任意两个 =0	1	
111	1	任意两个 =0	0	所有样本=1

16.16.2. UART 接收噪音字符

通常，UART 数据接收信号 **URT_x_RX** 可被内部滤波器滤波并通过过采样多数投票获取有效的数据。UART 模

块可检测噪音字符位并置起 NCF 标志。这对用户通过固件或硬件提升 UART 通信的精确性提供了信息。

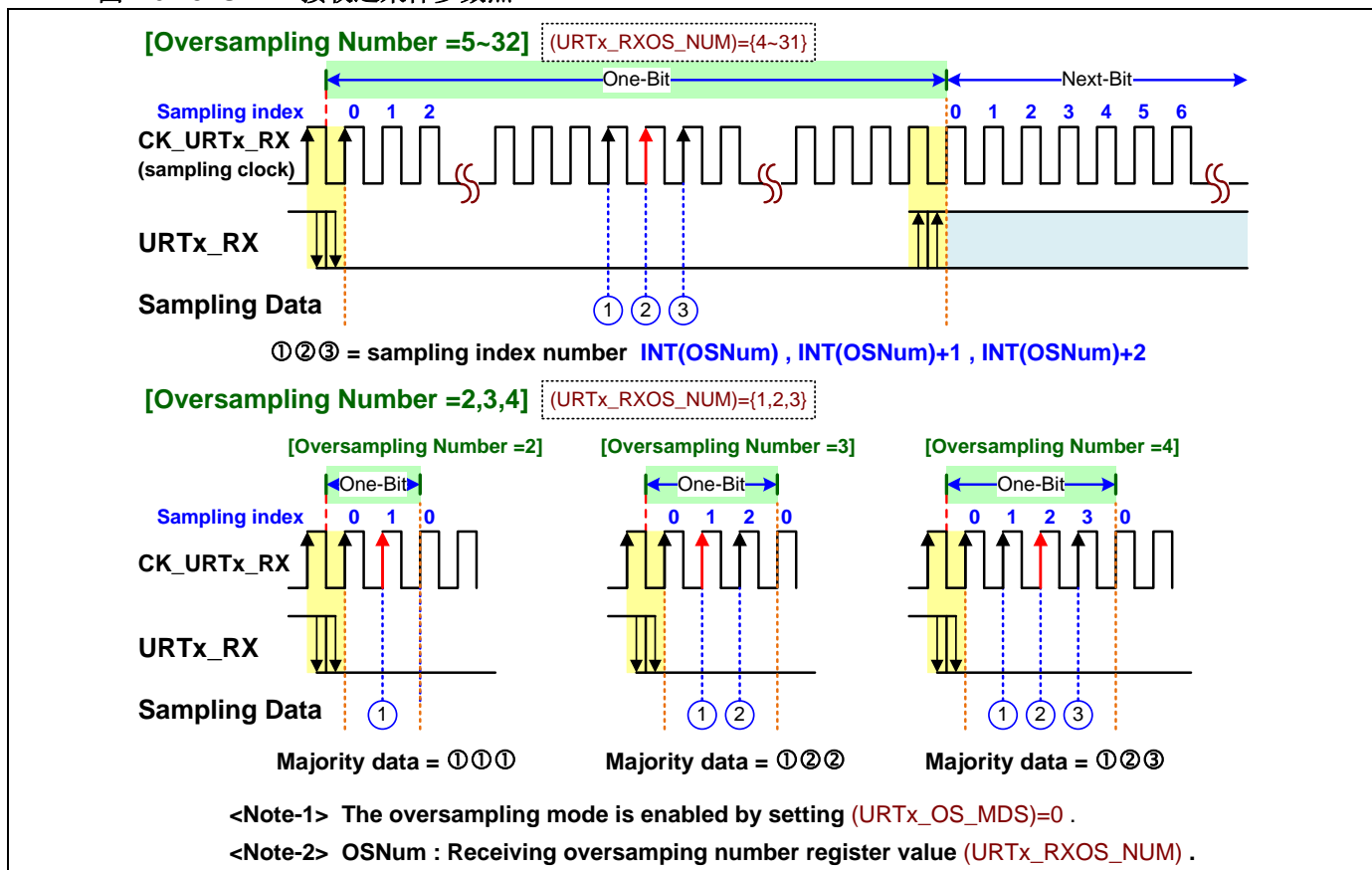
如表格“UART 接收数据过采样和噪音检测”所示，NCF 标志会在这 3 个样本均不为 0 或 1 时置起。

16.16.3. UART 接收过采样

用户可在 **URT_x_RXOS_NUM** 寄存器中选择有效的过采样数。当用户选择 3 样本作过采样多数投票时，建议选择有效值位 8~32（寄存器值 7~31）。当过采样数为 5~32，芯片会采用 3 个样本的中间数做数据值多数投票。当过采样数设置为 2~4，则会选择固定的样本，如下图表所示。

下面的图表展示了 UART 接收过采样多数点用于不同的过采样数。

图 16-25. UART 接收过采样多数点



16.16.4. UART 接收位时间错误公差

有一个简单的办法计算最快和最慢的位时间范围，以得到正确的通信。

$$\text{Valid Bit-Time : } T_Q < T_{\text{vaild}} < T_S$$

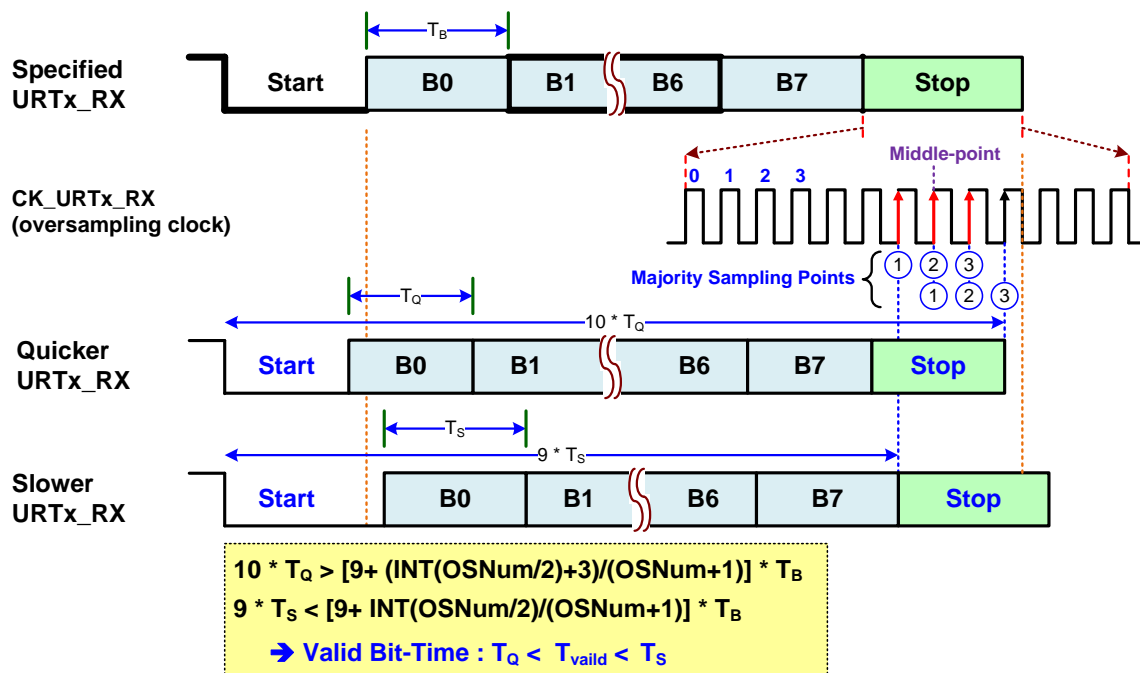
T_B 是用于数据接收的指定位时间； T_Q 和 T_S 是数据接收的最快和最慢速度的位时间；OSNum 是在 **URT_x_RXOS_NUM** 寄存器中设置的过采样数。下面的公式用于计算有效的数据接收时间范围。

$$10 * T_Q > \frac{9 + (\text{INT}(\text{OSNum}/2) + 3)}{(\text{OSNum} + 1)} * T_B$$

$$9 * T_S < \frac{9 + \text{INT}(\text{OSNum}/2)}{(\text{OSNum} + 1)} * T_B$$

下面的图表展示了 UART 接收位时间错误公差。

图 16-26. UART 接收位时间错误公差

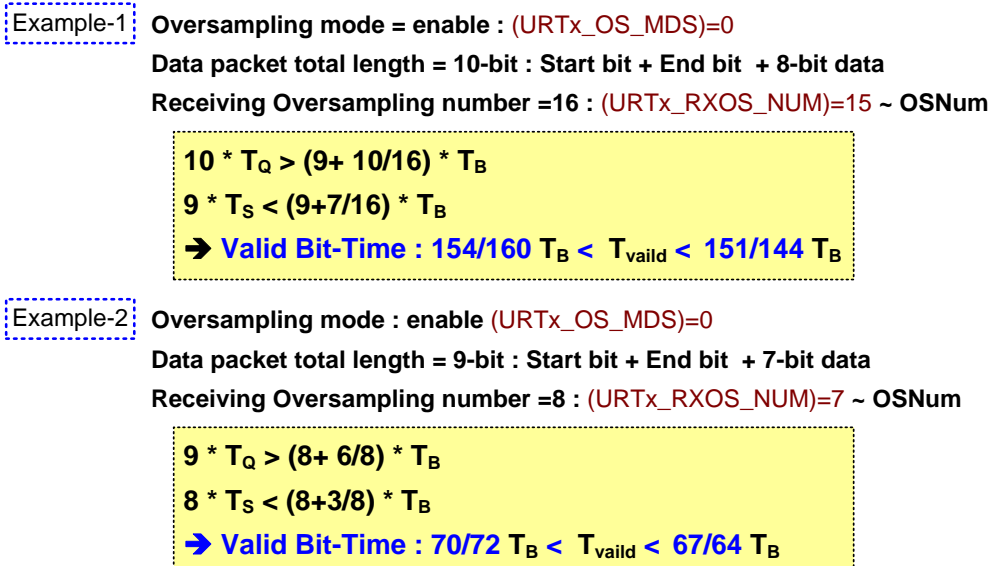


<Note-1> The oversampling mode is enabled by setting (URTx_OS_MDS)=0 and data packet length =10-bit .

<Note-2> OSNum : Receiving oversampling number register value (URTx_RXOS_NUM) .

下面的图表展示了上图的 UART 接收位时间错误公差计算示例。

图 16-27. UART 接收位时间错误公差示例



16.17. UART TMO 超时控制

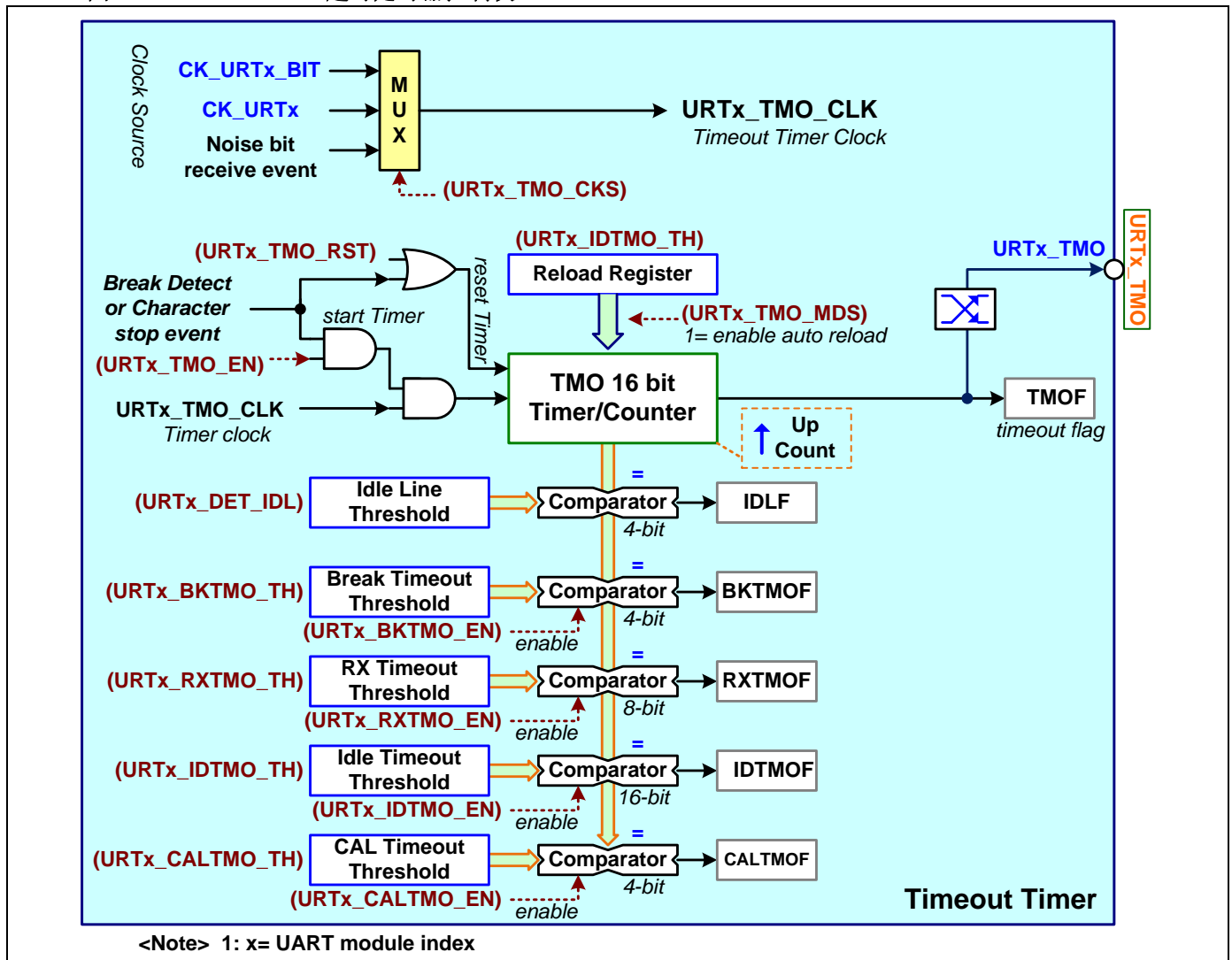
该模块提供了 1 个 16 位超时定时器(TMO) 用于 UART 访问超时控制。TMO 定时器可通过 **URTx_TMO_MDS** 寄存器设置为 UART 超时定时器或通用定时器，并通过 **URTx_TMO_EN** 寄存器使能。当 TMO 定时器被设置为通用定时器时，**URTx_IDTMO_TH** 寄存器用于作为定时器的重载寄存器。

TMO 定时器可用于检测 Idle 线状态、中止超时、RX 超时、Idle 超时和波特率校准超时。检测中止超时、RX 超时、Idle 线状态和波特率校准超时有它们独立的使能寄存器位 **URTx_BKTMO_EN**, **URTx_RXTMO_EN**, **URTx_IDTMO_EN** 和 **URTx_CALTMO_EN**。

用户可设置 **URTx_TMO_CKS** 寄存器选择 TMO 定时器时钟源。特别的，定时器可根据接收到的噪音位输入计算通信噪音状态。参照“UART 接收数据过采样和噪音检测”表以获取关于噪音位定义的信息。

超时定时器复位使能位 **URTx_TMO_RST** 用于强制复位 TMO 定时器。此外，用户还可通过读 **URTx_TMO_CNT** 寄存器获取 TMO 定时器的实时计数器值。

图 16-28. UART TMO 超时定时器控制块



16.17.1. UART TMO 超时定时器启动/关闭控制

当 TMO 定时器被设置为通用定时器，不能通过设置 **URT_x_EN** 寄存器使能该定时器。
[注释]:在 MG32F02A132/072 中，将 TMO 定时器作为“通用定时器”时必须使能 **URT_x_EN**。

下面的表格展示了 UART TMO 超时定时器开启/关闭控制。

表 16-12. UART TMO 超时定时器开启/关闭控制

TMO 定时器功能	URT _x 寄存器			定时器开启/关闭
	TMO_MDS	EN	TMO_EN	
URT _x 超时定时器	0	0	x	定时器关闭
		1	0	定时器关闭
			1	定时器开启
通用定时器	1	x	0	定时器关闭
			1	定时器开启

x：不影响

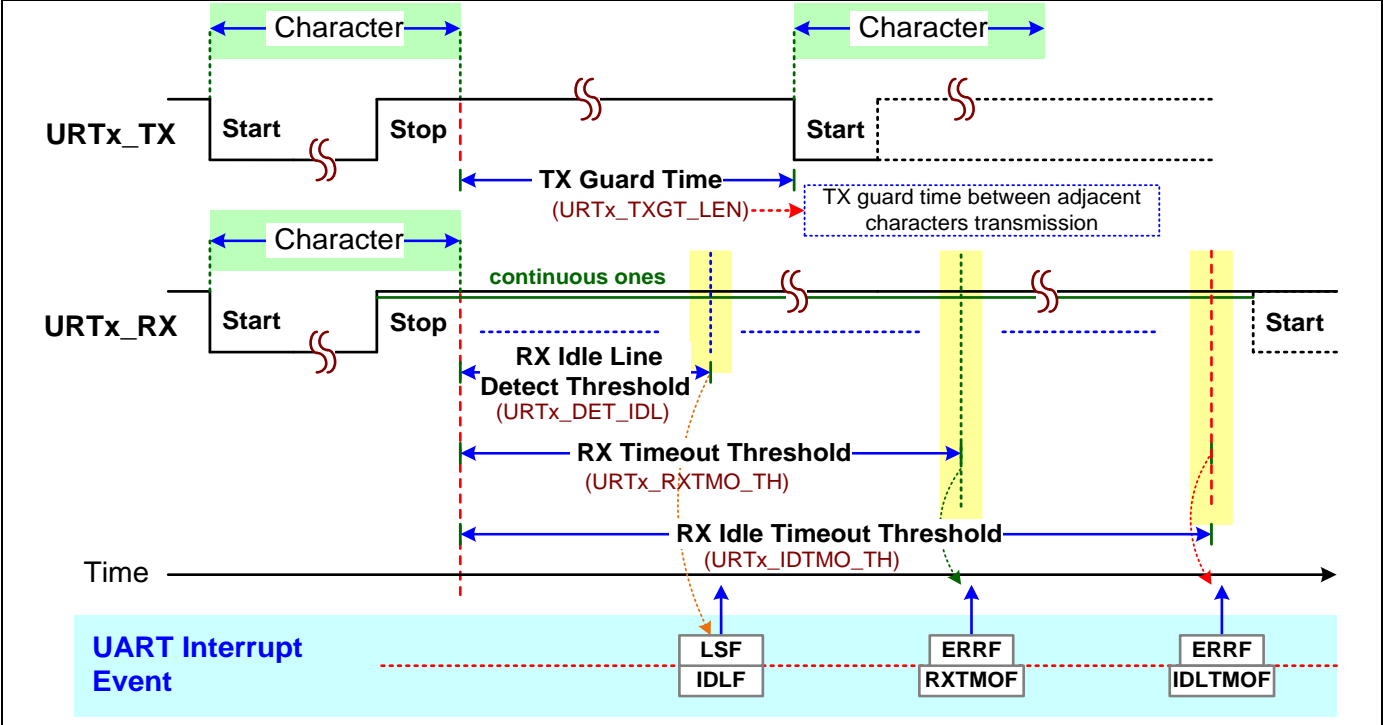
16.17.2. UART 保持时间和超时检测

用于数据传输，**URT_x_TXGT_LEN** 可用于设置两个传输字符之间的保持时间。

对于数据接收，用户可通过 **URT_x_DET_IDL** 寄存器定义从上一个停止位开始的 Idle 线检测阈值的位时间。用户可通过 **URT_x_IDL_MDS** 寄存器选择 Idle 线检测管理模式。当选择“**No**”且检测到 Idle 线时，芯片会载入。当选择“**Load**”且检测到 Idle 线时，若阴影缓冲不为空，即使没有超过接收阈值 **URT₀_RX_TH**，芯片也会把阴影缓冲载入 **URT_x_RDAT** 数据寄存器。

URT_x_IDTMO_TH 寄存器用于定义数据发送超时阈值用于 **智能卡** 通信中。

图 16-29. UART 保持时间和超时检测

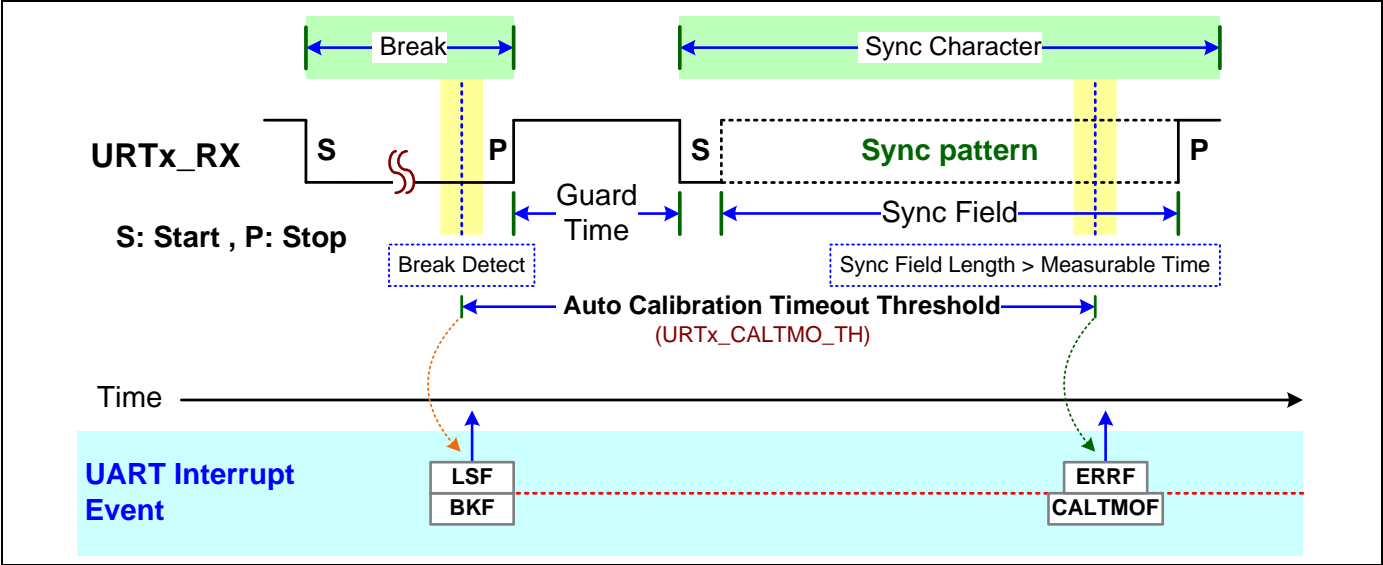


URT_x_RXTMO_TH 寄存器用于定义 RX 缓冲数据时效超时阈值。这对用户进行控制接收数据流有所帮助。当阴影缓冲接收的数据数量未达到 RX 阈值且过了很长的时间，若用户设置了 **URT_x_RXTMO_TH** 寄存器，芯片可检测该状态并置起 RXTMOF 标志(**URT_x_RXTMOF**)。此时，若 RX 数据寄存器为空，阴影缓冲数据会强制载入 RX 数据寄存器(**URT_x_RDAT**)并置起 RXF 标志(**URT_x_RXF**)。用户可通过 **URT_x_RNUM** 寄存器获取数据字节数。

16.17.3. UART 自动校准超时

当 UART 模块进行波特率校准操作时，**URT_x_CALTMO_TH** 寄存器可定义校准超时阈值用于不可预知的校准错误状况。

图 16-30. UART 自动校准超时错误检测



下面的表格展示了波特率定时器分离模式和混合模式的 UART 波特率校准超时状态。

表 16-13. UART 校准超时状态

波特率发生器模式	寄存器	芯片支持		校准超时状态
BR_MDS	CALTMO_TH	MG32F02A132 MG32F02A072	MG32F02A032	
0 (分离模式)	0	V	V	RLR 上溢 0xFF
	1 ~ 15			RLR 上溢(CALTMO_TH * 0x10)
1 (混合模式)	0		V	[PSR : RLR]上溢 0xFFFF
	1 ~ 15			[PSR : RLR] 上溢(CALTMO_TH * 0x100)

注释：PSR/RLR/CALTMO_TH ~ URT_x 寄存器

下面的表格展示了波特率定时器分离模式和混合模式的 UART 波特率校准超时时间。

表 16-14. UART 校准超时时间

波特率发生器模式	校准寄存器		芯片支持		校准超时时间
BR_MDS	CAL_MDS	CALTMO_TH	MG32F02A132 MG32F02A072	MG32F02A032	
0 (分离模式)	0 (Start 模式)	0	V	V	(PSR * RXOS_NUM) * 0xFF
		1 ~ 15			(PSR * RXOS_NUM) * (CALTMO_TH * 0x10)
	1 (Edge 模式)	0			(PSR * RXOS_NUM) * 0xFF * 2
		1 ~ 15			(PSR * RXOS_NUM) * (CALTMO_TH * 0x10) * 2
1 (混合模式)	0 (Start 模式)	0	V	V	RXOS_NUM * 0xFFFF
		1 ~ 15			RXOS_NUM * (CALTMO_TH * 0x100)
	1 (Edge 模式)	0			RXOS_NUM * 0xFFFF * 2
		1 ~ 15			RXOS_NUM * (CALTMO_TH * 0x100) * 2

注释：PSR/RXOS_NUM/CALTMO_TH ~ URTx 寄存器

16.17.4. UART 中止状态超时

URTxBKTMO_TH 寄存器定义中止状态超时阈值用于时间过长的中止错误状态。参照“UART 中止和校验/帧错误检测”节的描述以获取更多信息。

16.17.5. UART TMO 超时信号输出

TMO 超时定时器可输出超时信号作为时钟信号到外部端口 URTxTMO 或内部模块中。用户可通过 URTxTMO_STA 寄存器设置超时信号的初始状态。特别的是，该初始值寄存器只有在 URTxTMO_LCK 寄存器同时写“1”时才能被写入。

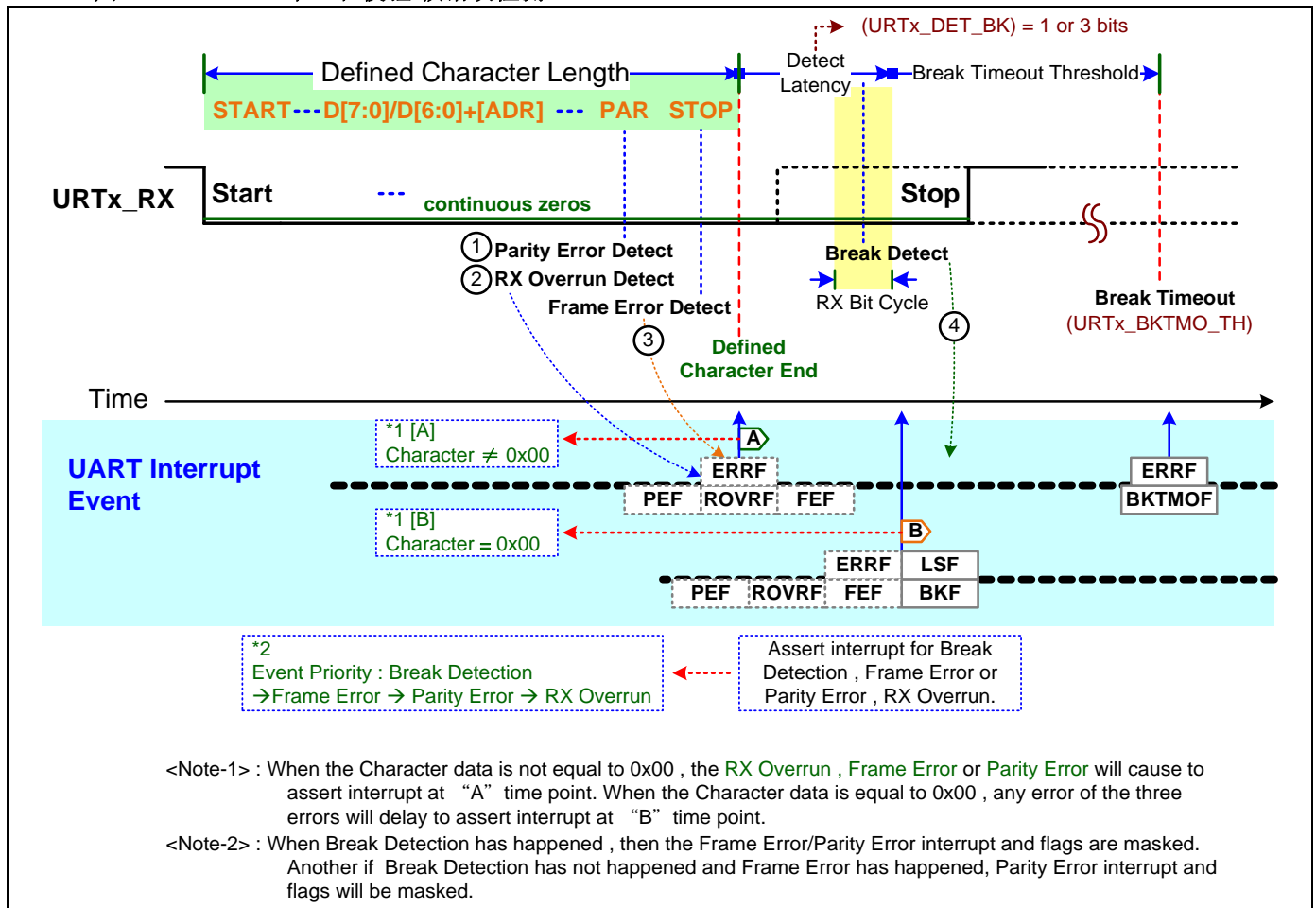
16.18. UART 错误管理

UART 模块可检测和管理一些通用通信错误。超时定时器(TMO)用于检测中止状态、波特率校准、阴影缓冲时效和 Idle 状态的超时错误。参照“UART TMO 超时控制”节以获取更多相关描述。

16.18.1. UART 中止和校验/帧错误检测

当 UART 模块在 PAR 位检测到校验错误或数据接收溢出和 STOP 位检测到帧错误时，相关的 PEF, ROVRF 和 FEF 事件标志会在 STOP 位后被延迟启动。若接收字符不等于 0x00 时，这些标志会在 STOP 位末尾置起；若接收字符等于 0x00 时，这些标志会在中止状态的被检测时间置起。该状态下，若中止状态被检测到时，帧错误和校验错误的中断和标志会被屏蔽。若中止状态没发生且没检测到帧错误，校验错误中断和标志会被屏蔽。

图 16-31. UART 中止和校验/帧错误检测



16.18.2. UART TX 错误检测和重传控制

UART 模块可检测智能卡通信中 STOP 位拉低的发送 TX 错误或 Lin 通信中发送数据读回检查, 这些功能都是通过 **URT_x TXE MDS** 寄存器设置实现。当检测到 TX 错误, TXEF 标志(**URT_x TXEF**)会被置起。

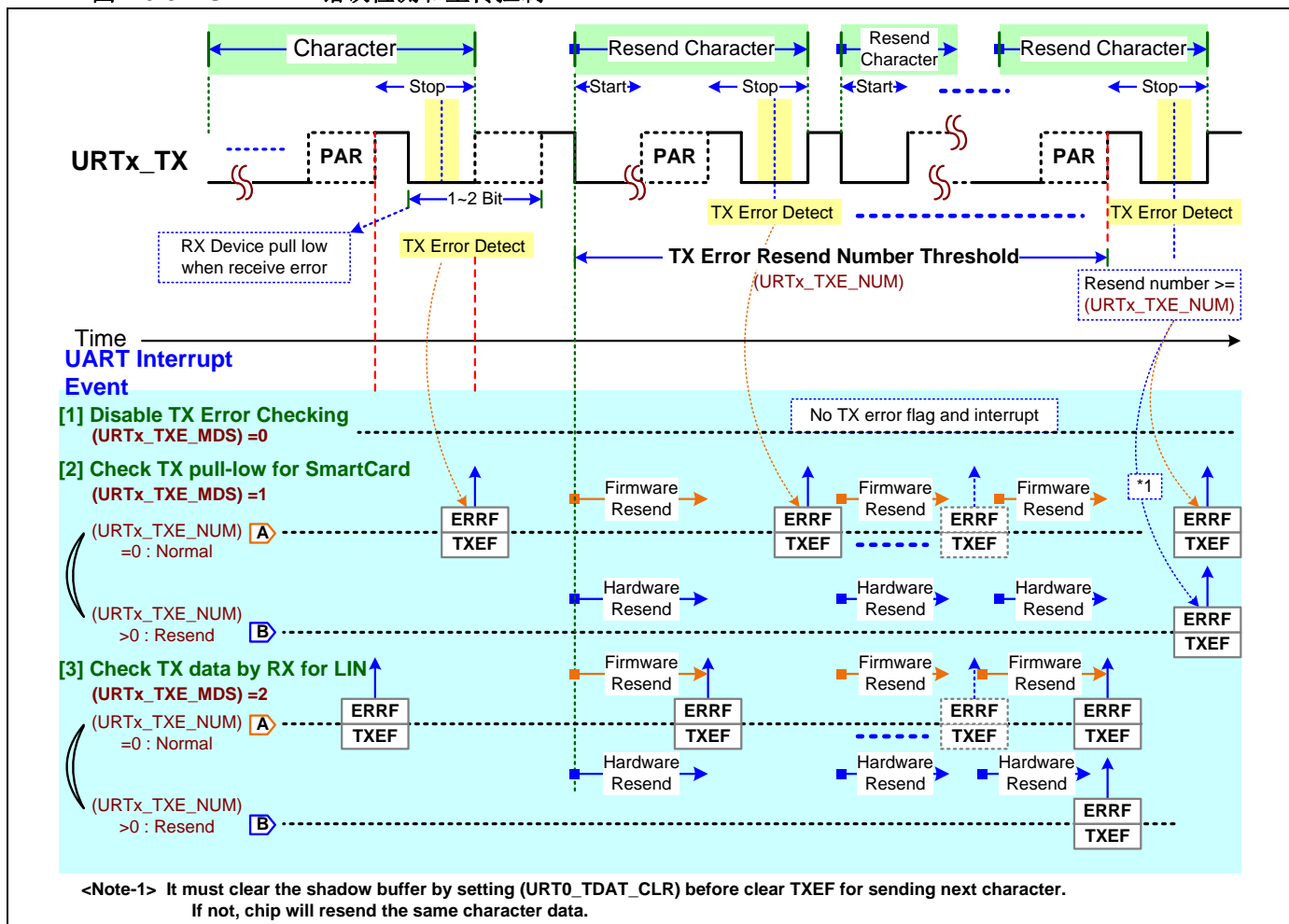
对于 **智能卡** 通信，用户可设置 **URTx_TXE_MDS** 寄存器到“CHK_Low”，且芯片可通过在停止位被外部接收者拉低检测 **URTx_TX** 信号，以表明通信错误。

对于 **LIN** 通信，用户可通过 **URTx_TXE_MDS** 寄存器设置为“CHK_TX”，使芯片可从内部接收路径读回发送的数据并检查数据是否因为应用软件或其它原因出错。

用户可通过设置 **URTx_TXE_MDS** 寄存器到“CHK_Low”或“CHK_TX”使能自动重传功能并通过 **URTx_TXE_NUM** 寄存器设置当检测到 TX 错误时的最后数据的重传数。当自动重传功能已使能且 **URTx_TXE_NUM** 的值>0 时, TXEF 标志会被屏蔽直到仍在最后一次重传时仍发生错误。

[注释]: 当使能了TX 错误检测功能时 **URT_x TX** 引脚需设置为开漏模式。

图 16-32. UART TX 错误检测和重传控制



16.18.3. UART RX 校验错误检测和重试控制（用于智能卡）

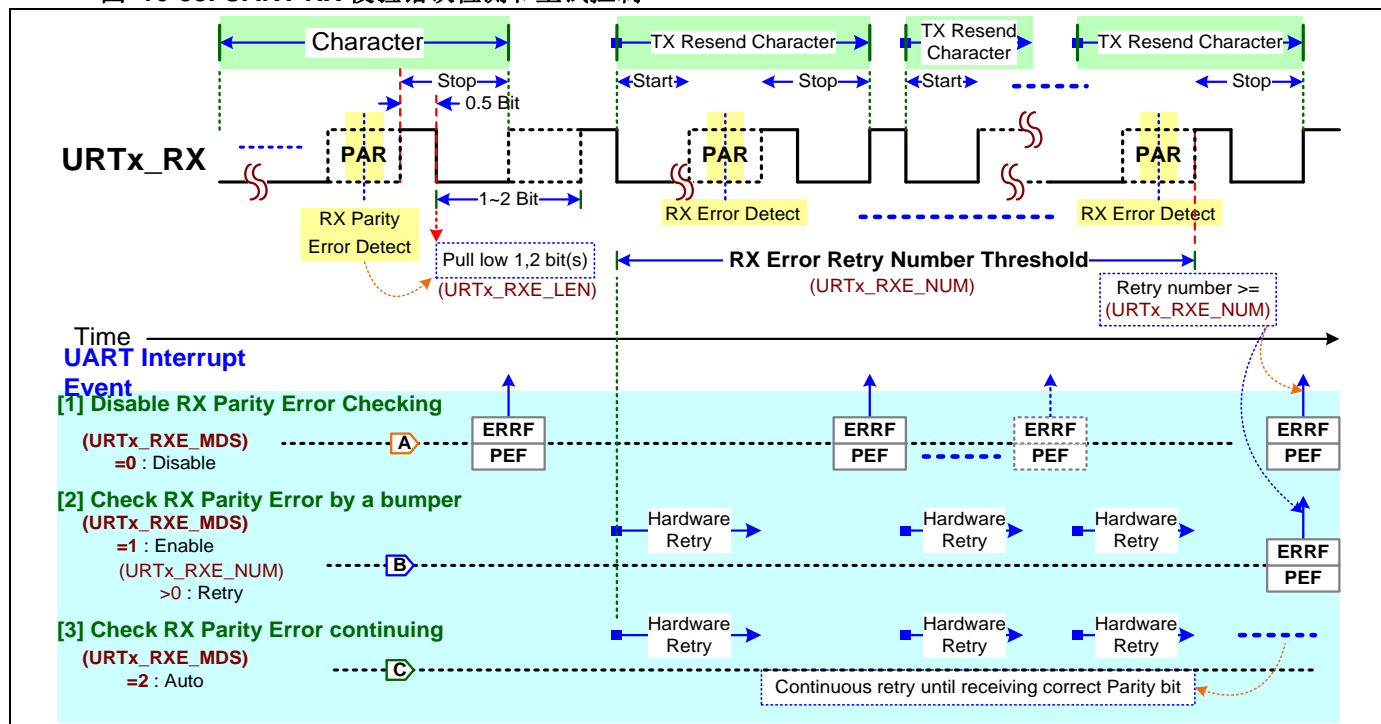
UART 模块可通过 **URT_xRXE_MDS** 寄存器使能接收 RX PAR 位校验错误和选择检测模式。当检测到 RX 校验错误时，PEF 标志(**URT_PEF**) 会被置起。PAR 位校验错误功能只支持停止位 = 1 位。

用户可通过 **URT_xRXE_MDS** 寄存器使能自动重试功能为“使能”或“自动”模式，还可通过 **URT_xRXE_MDS** 寄存器设置当检测到 RX 校验错误时最后数据的重试数。当自动重试功能被使能，当检测到 RX 校验错误时，芯片会在停止位周期拉低 **URT_xRX** 输入信号。若 **URT_xRXE_MDS** 寄存器被设置为“自动”模式，芯片会一直做重试操作，PEF 标志会一直被屏蔽。此时，芯片将不会置起中断和重试接收新数据；若 **URT_xRXE_MDS** 寄存器被设置为“使能”模式且 **URT_xRXE_NUM** 的值 > 0 时，芯片会执行重试操作，TXEF 标志会被屏蔽，直到在最后一次重试数据仍出错。

*[注释]: 当使能了 TX 错误检测功能时 **URT_xRX** 引脚需设置为开漏模式。*

用户可通过 **URT_xRXE_LEN** 寄存器设置当 RX 校验错误发生时的拉低 RX 线位时间长度。

图 16-33. UART RX 校验错误检测和重试控制



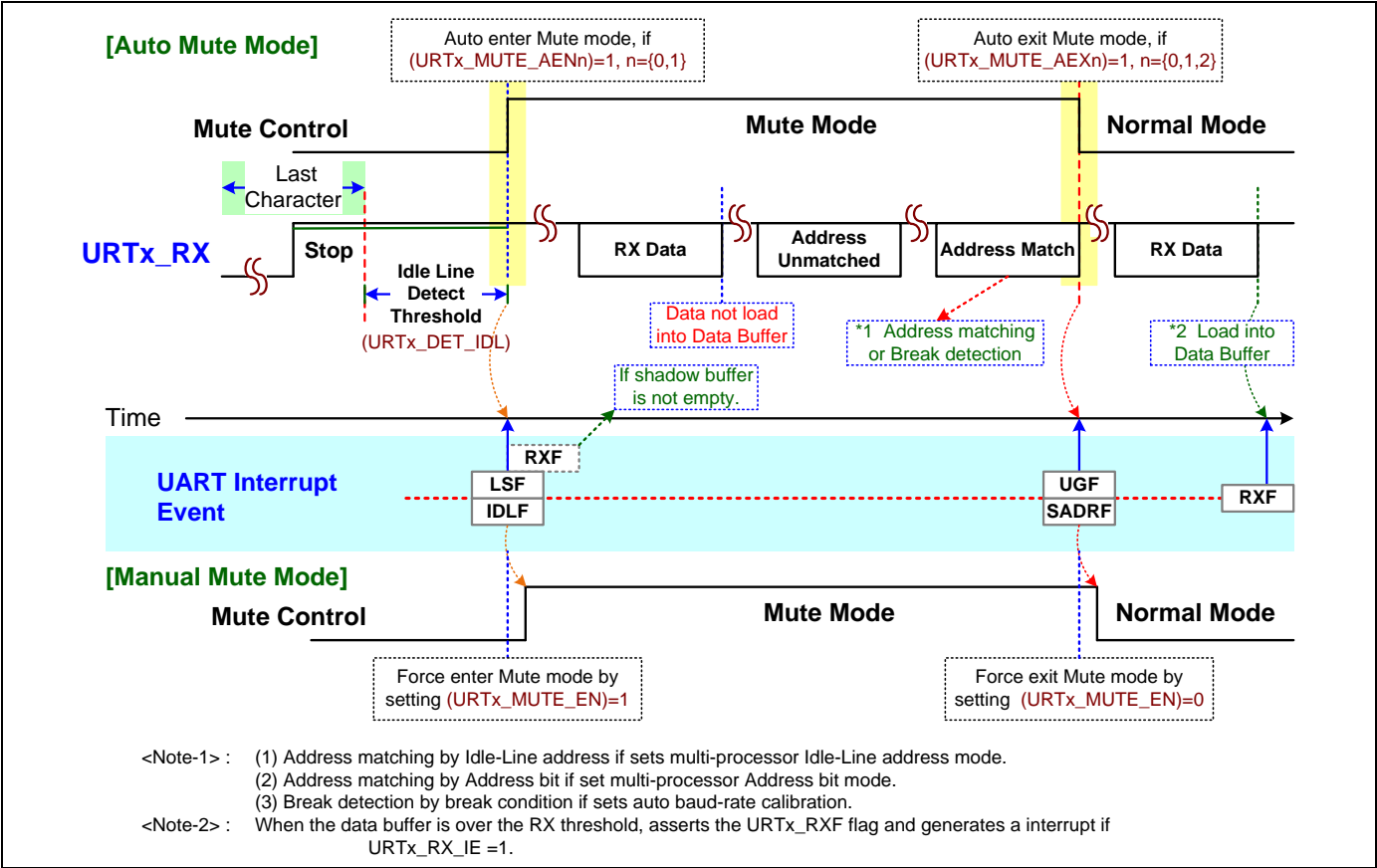
16.19. UART 静音模式控制

UART 模块支持静音模式以禁用接收数据字符，但是移动缓冲仍会工作，作为状态检测。当 UART 进入静音模式，RX 阴影缓冲将不会再从移动缓冲中载入数据。静音模式在多处理器通信中能起到作用。

静音模式可通过 **URT_xMUTE_EN** 寄存器设置直接强行进入或退出。因此用户可手动控制静音模式的进入和退出。

用户可通过 **URT_xMUTE_AEN_n** 寄存器设置多处理器从机地址不匹配状态和 Idle 线检测自动进入或退出静音模式的触发事件。此外用户还可通过 **URT_xMUTE_AEN_n** 寄存器设置多处理器从机地址匹配状态、中止状态检测和 Idle 线检测自动进入或退出静音模式的触发事件。(n=寄存器标号)

图 16-34. UART 静音模式控制时序



下面的表格展示了 UART 多处理器地址匹配或不匹配状态的静音模式控制。**URT_xMUTE_EN** 寄存器会根据这些状态被硬件设置。

表 16-15. UART 多处理器地址匹配比对静音模式控制

地址匹配状态	寄存器设置		受影响的寄存器
	MUTE_AEN0	MUTE_AEX0	MUTE_EN
地址匹配	0	x	0
	1	0	1
		1	0
地址不匹配	0	x	0
	1	x	1

注释：若 MUTE_EN=1，RX 移动寄存器数据将不会被载入 RX 阴影缓冲，

16.20. UART 同步模式

16.20.1. UART 同步模式时序

UART 模块可通过 **URT_x_MDS** 寄存器设置为“SYNC”支持同步模式时序。对于同步模式，用户可通过 **URT_x_CPOL** 寄存器设置时钟模式，通过 **URT_x_CPHA** 寄存器设置 **URT_x_CLK** 信号。此外，用户可通过 **URT_x_SWEN** 和 **URT_x_SWO** 寄存器直接控制 **URT_x_NSS** 输出信号用于 SPI 主机通信。用户可通过 **URT_x_TX_CKS** 和 **URT_x_TXOS_NUM** 寄存器设置同步模式时钟源和频率。参照“[UART 时钟控制](#)”节以获取更多关于 SPI 时钟的信息。

MG32F02A132/072 的同步模式的起始位是低电平，MG32F02A032 则是高电平。

图 16-35. UART 同步模式时序

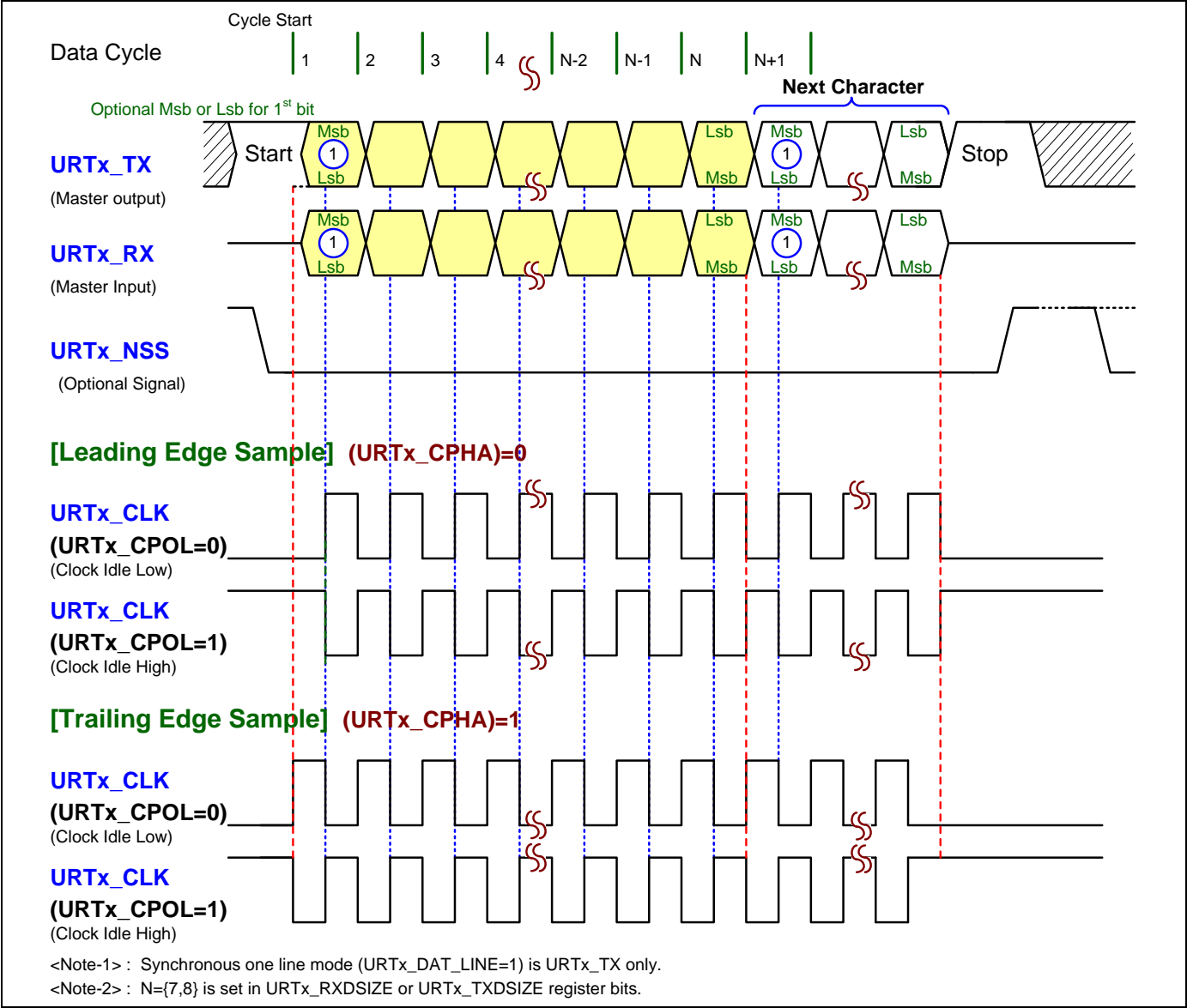


表 16-16. UART 同步时钟模式表

时钟模式	URT _x 寄存器		功能描述
	URT _x _CPOL	URT _x _CPHA	
0	0	0	UART 时钟低电平为闲置状态，数据采样为前沿。
1	0	1	UART 时钟低电平为闲置状态，数据采样为后沿。
2	1	0	UART 时钟高电平为闲置状态，数据采样为前沿。
3	1	1	UART 时钟高电平为闲置状态，数据采样为后沿。

16.20.2. UART 同步模式 NSS 控制

UART 同步模式支持 SPI 通信中的带 NSS 信号的 SPI 主机模式。用户可通过 **URT_x_NSS_SWEN** 寄存器设置硬件控制 NSS 控制模式的 **URT_x_NSS** 信号或软件直接控制。**URT_x_NSS_INV** 寄存器用于反相 **URT_x_NSS** 输出信号。

- 同步模式软件 NSS 控制

当通过 **URT_x_NSS_SWEN** 寄存器设置使能软件 NSS 控制，用户可直接通过 **URT_x_NSS_SWO** 寄存器控制 **URT_x_NSS** 信号输出电平以支持带 NSS 信号的 SPI 通信。

- 同步模式硬件 NSS 控制

当通过 **URT_x_NSS_SWEN** 寄存器设置使能硬件 NSS 控制，芯片会如下面同步模式 NSS 图表自动生成 **URT_x_NSS** 信号。

图 16-36. UART 同步模式硬件 NSS 时序

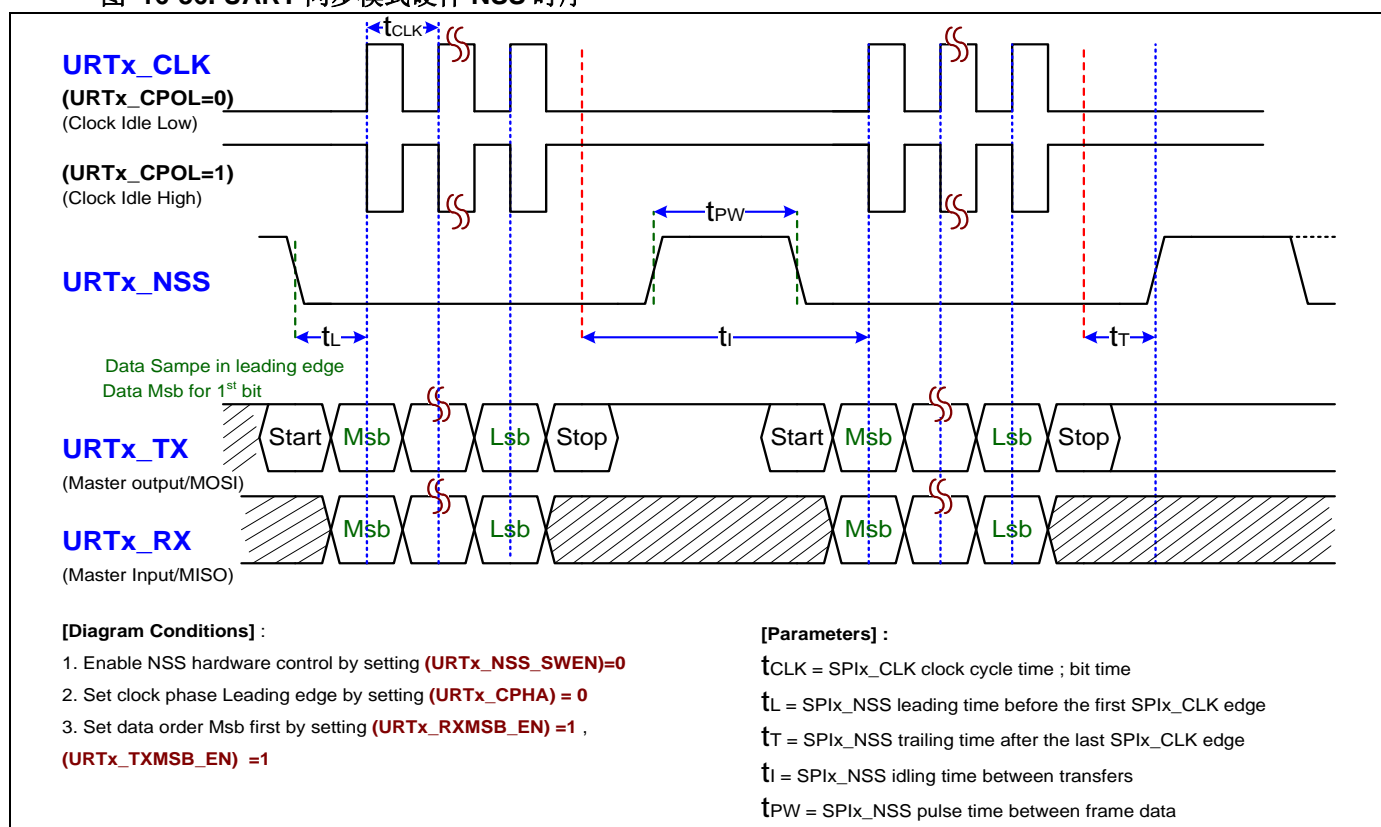


表 16-17. UART 同步模式 NSS 时序表

时钟模式	NSS 时序				单位
	t_L	t_T	t_I	t_{PW}	
0 (CPOL=0, CPHA=0)	1	0/0.5 (*1)	1.5/2/3 (*3)	0.5/1/1.5 (*4)	t_{CLK}
1 (CPOL=0, CPHA=1)	0.5	0.5/1 (*2)	1.5/2/3 (*3)	0.5/1/1.5 (*4)	t_{CLK}
2 (CPOL=1, CPHA=0)	1	0/0.5 (*1)	1.5/2/3 (*3)	0.5/1/1.5 (*4)	t_{CLK}
3 (CPOL=1, CPHA=1)	0.5	0.5/1 (*2)	1.5/2/3 (*3)	0.5/1/1.5 (*4)	t_{CLK}

t_{CLK} : UART 位时间

t_L : 第一个 URT_x_CLK 沿 (位时间) 前的 URT_x_NSS 前沿时间

t_T : 最后一个 URT_x_CLK 沿 (位时间) 后的 URT_x_NSS 后沿时间

t_I : 传输之间 (位时间) 的 URT_x_NSS 空闲时间 $t_I = 0$ if SPI0_NSS_PEN=0

t_{PW} : URT_x SPI 模式空闲期间硬件 NSS 的脉宽 (位时间)

注释 : *1 : 0 位时间若 Stop bit=0.5bit, 0.5 位时间若 Stop bit=1/1.5/2bit

*2 : 0.5 位时间若 Stop bit=0.5bit, 1 位时间若 Stop bit=1/1.5/2bit

*3 : 1.5 位时间若 Stop bit=0.5bit, 2 位时间若 Stop bit=1/1.5bit, 3 位时间若 Stop bit=2bit

*4 : 0.5 位时间若 Stop bit=0.5/1bit, 1 位时间若 Stop bit=1.5bit, 1.5 位时间若 Stop bit=2bit

16.21. UART 智能卡时钟输出

波特率发生器可通过 **CK_URTx_PSC** 寄存器设置输出 **智能卡** 时钟。用户可通过 **URTx_PSR**, **URTx_RLR** 和 **URTx_TXOS_NUM** 寄存器设置波特率发生器用于 **智能卡** 时钟输出。

16.21.1. 智能卡时钟频率计算

根据 **ISO/IEC 7816-3** 标准，以下列出了相关缩略术语的描述：

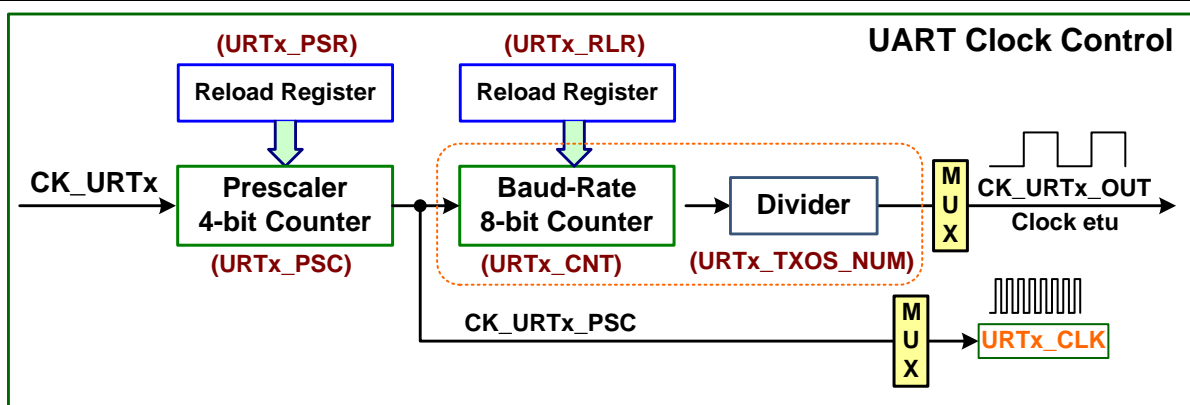
- etu** : 基本时间单位(位时间)
- F** : 时钟速率转换整数
- D** : 波特率调整整数
- f** : 通过接口设备提供到卡的时钟信号频率值

在 **智能卡** 通信中，该模块可接收元素“F”，“D”，“f”用于外部智能卡。用户可计算这些元素并设置有效值到 **URTx_PSR**, **URTx_RLR** 和 **URTx_TXOS_NUM** 寄存器中。

下面的图表展示了 UART 智能卡时钟输出的计算。

[注释]: 元素 “D” 的值被固定为 1。

图 16-37. UART 智能卡时钟输出



$$etu = F / (D * f)$$

$$f(CK_URTx) = (URTx_PSR + 1) * f(CK_URTx_PSC)$$

$$\begin{aligned} f(etu) &= (D * f) / F = f(CK_URTx_BIT) \\ &= f(CK_URTx) / ((URTx_PSR + 1) * (URTx_RLR + 1) * (URTx_TXOS_NUM + 1)) \\ &= (f(CK_URTx_PSC)) / ((URTx_RLR + 1) * (URTx_TXOS_NUM + 1)) \end{aligned}$$

<Note>

$$F = (URTx_RLR + 1) * (URTx_TXOS_NUM + 1)$$

D = fixed to 1

$$f = f(CK_URTx_PSC) = f(CK_URTx) / (URTx_PSR + 1)$$

16.21.2. UART 智能卡时钟输出设置示例

下面的表格展示了 UART 智能卡输出设置的示例。

表 16-18. UART 智能卡时钟输出设置示例

智能卡 etu		CK_URTx_SC	URTx 寄存器			智能卡参数		
etu (*1) (us)	f _(etu) (KHz)	f _(sc) (MHz)	PSR (*2)	RLR	TXOS_NUM RXOS_NUM	F	f _(max) (MHz)	D
93.00	10.75	4.000	11	11	30	372	4	1
77.50	12.90	4.800	9	11	30	372	5	1
93.00	10.75	6.000	7	17	30	558	6	1
93.00	10.75	8.000	5	92	7	744	8	1
93.00	10.75	12.000	3	35	30	1116	12	1
93.00	10.75	16.000	2	185	7	1488	16	1
116.25	8.60	16.000	2	59	30	1860	20	1
106.67	9.38	4.800	9	63	7	512	5	1
112.00	8.93	6.857	6	95	7	768	7.5	1
106.67	9.38	9.600	4	127	7	1024	10	1
128.00	7.81	12.000	3	191	7	1536	15	1
128.00	7.81	16.000	2	127	15	2048	20	1

*1 : $etu = F / (D * f_{(sc)})$

*2 : $CK_URTx_SC = CK_URTx / (PSR + 1)$ ~ $CK_URTx = 48.0\text{ MHz}$

16.22. UART IrDA 控制

UART 模块内建 1 个 IrDA 编码器和 IrDA 解码器在数据接口中用于 IrDA 通信，并可通过 **URT_x_IR_EN** 寄存器使能。当使能 IrDA 编码器和解码器时，过采样模式必须通过 **URT_x_RXOS_MDS** 寄存器设置为“3”。

16.22.1. UART IrDA 时序

对于 IrDA 数据传输，用户可通过 **URT_x_IR_PW** 寄存器设置 IrDA 数据脉冲宽度。

$$\text{IrDA 脉冲宽度} = (\text{URT}_{\text{x}}\text{_IR_PW}+1) * T_{\text{CK_URT}_{\text{x}}\text{_TX}}$$

对于有效的 IrDA 通信，数据脉冲宽度需要等于 3/16 到 4/16 的 1 个位时间的范围。比如说，当寄存器 **URT_x_TXOS_NUM** 值为 15 时，寄存器 **URT_x_IR_PW** 的值可为 2 或 3。(过采样数=16)

对于 IrDA 数据接收，用户可通过 **URT_x_IR_MDS** 寄存器选择接收位采样模式，当选择“Normal”，IrDA 采样序列值需等于 000，然后输出位值 0，其他输出 1；当选择“Wide”，IrDA 采样序列值需等于 000,001,010,100，然后输出位值为 0，其他的输出 1。

[注释]: **URT_x_IR_MDS** 寄存器不支持于 MG32F02A132/072。

图 16-38. UART IrDA 时序 – MG32F02A132/072

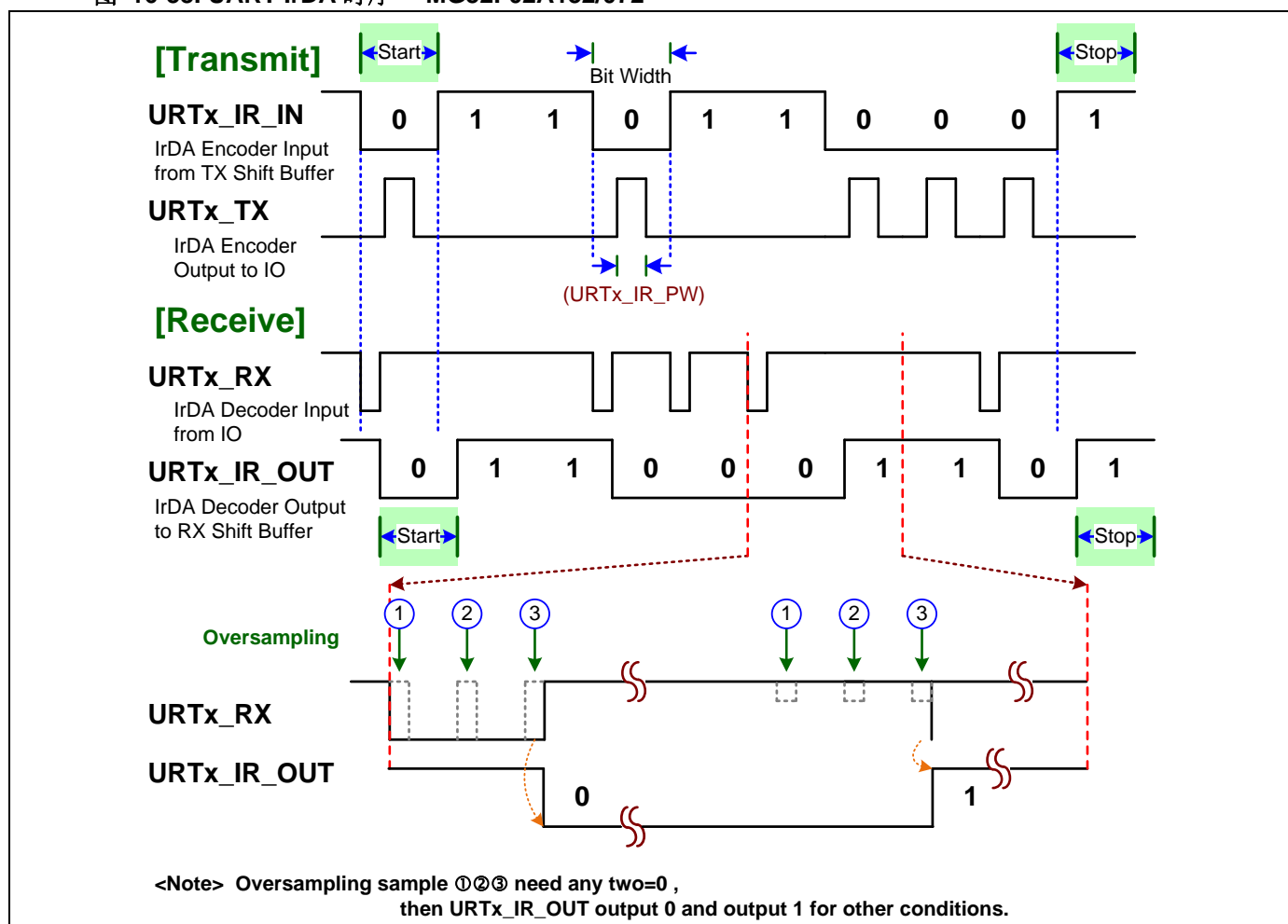
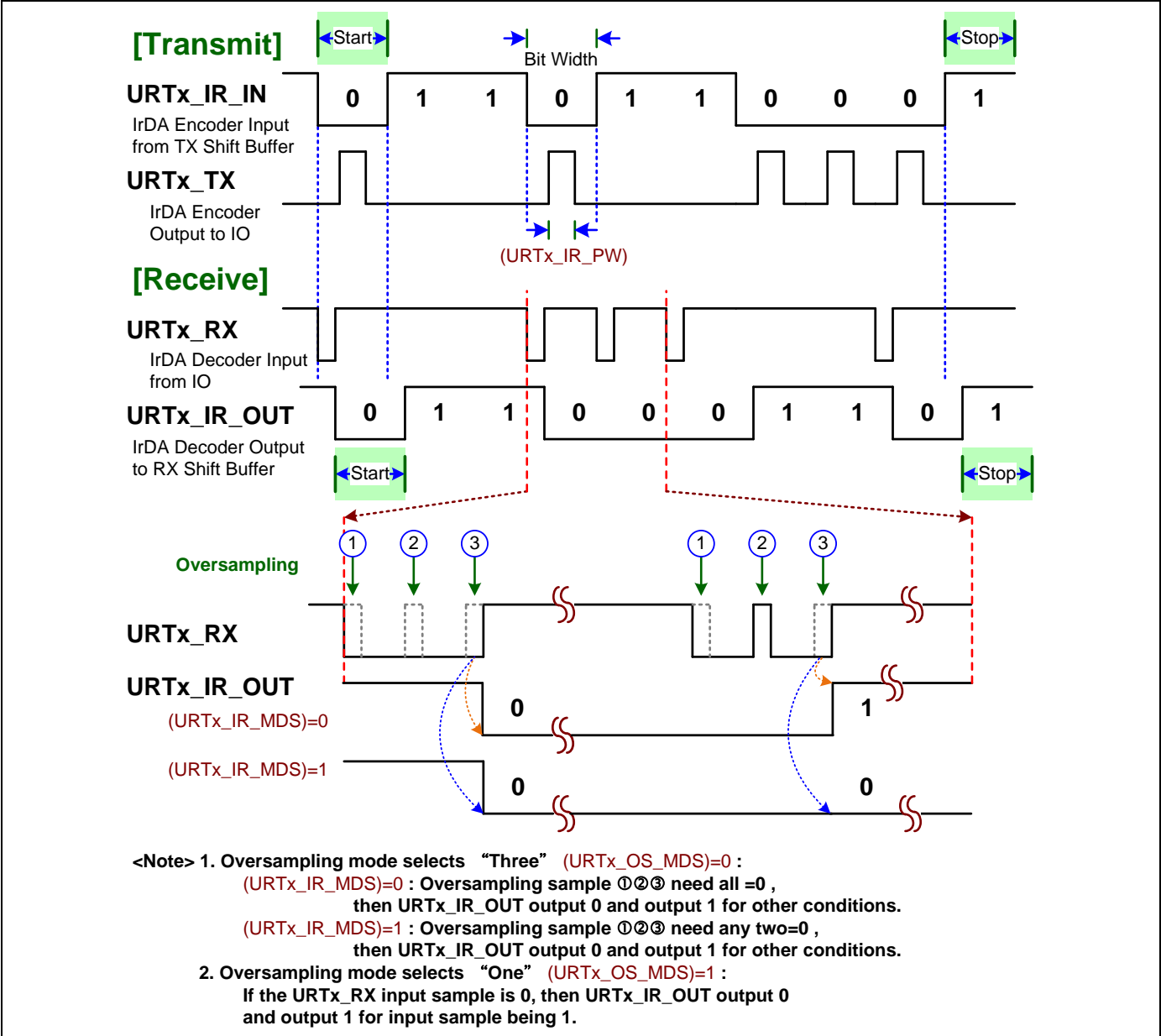


图 16-39. UART IrDA 时序 – MG32F02A032



16.22.2. UART IrDA 数据采样

下面的表格展示了“Normal”和“Wide”模式下接收位的值用于 IrDA 通信。

[注释]: MG32F02A132/072 不支持 “Wide”模式。

表 16-19. UART IrDA 接收数据过采样和采样模式

采样序列值	寄存器	接收位	注释
	IR_MDS		
000	0 (Normal)	0	全部 3 个样本=0
001		1	
010		1	
011		1	
100		1	
101		1	
110		1	
111		1	
000	1 (Wide)	0	任何 2 个样本 =0
001		0	任何 2 个样本 =0
010		0	任何 2 个样本 =0

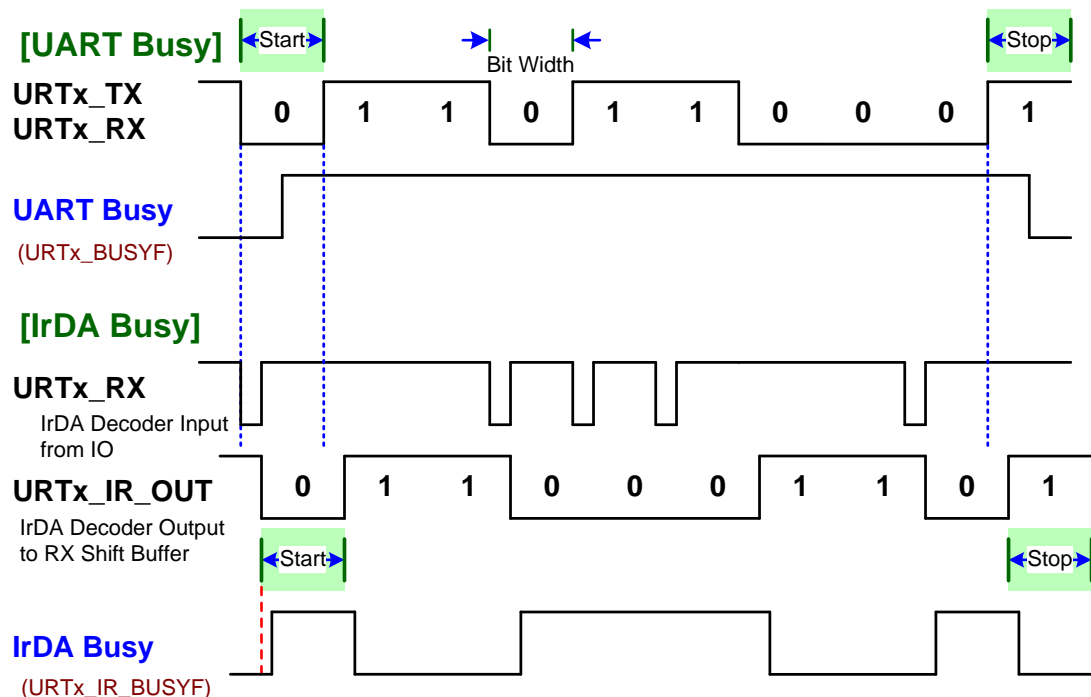
011		1	任何 2 个样本 =1
100		0	任何 2 个样本 =0
101		1	任何 2 个样本 =1
110		1	任何 2 个样本 =1
111		1	任何 2 个样本 =1

16.22.3. UART IrDA 占用标志

UART 模块提供 UART 占用标志(**URTx_BUSYF**)和 IrDA 占用标志(**URTx_IR_BUSYF**) 用于软件监视数据传输状态。

下面的图表展示了 UART 占用标志和 UART TX/RX 信号的关系, 也展示了 IrDA 占用标志和 UART RX 信号的关系。

图 16-40. UART 占用和 IrDA 占用状态



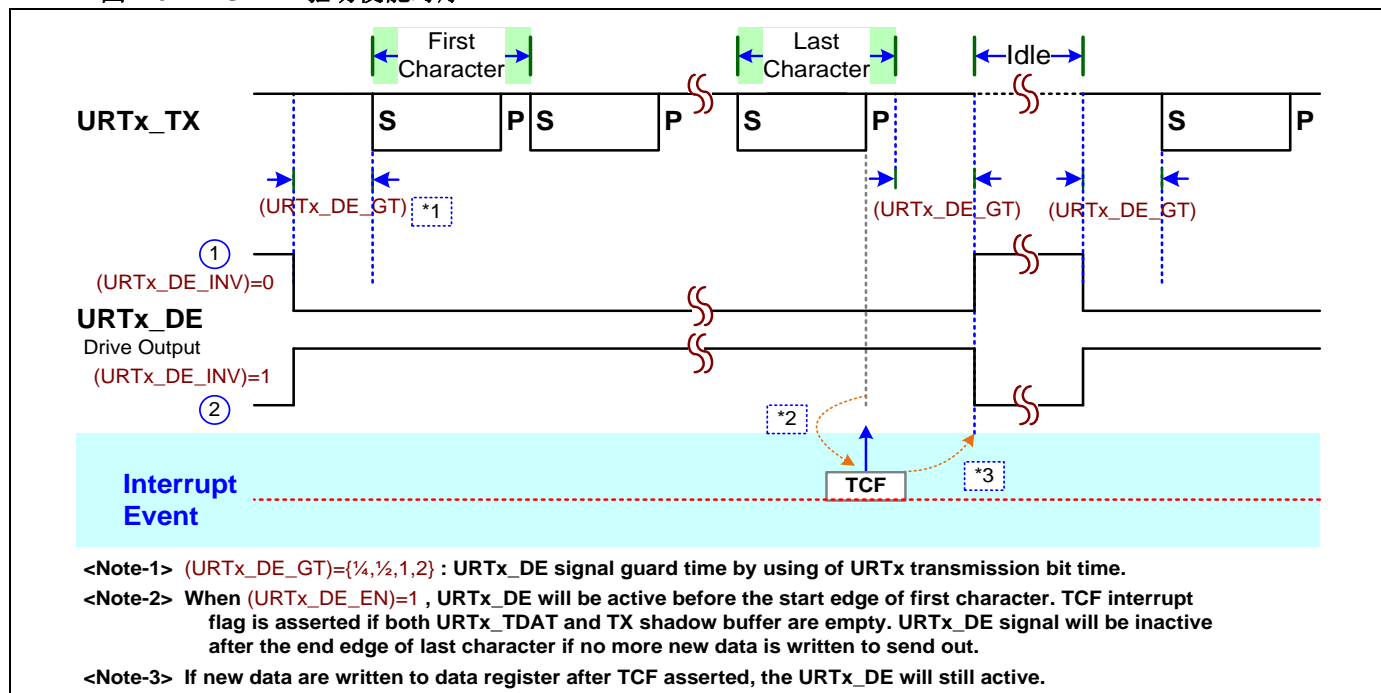
- <Note> 1. Oversampling mode selects "Three" (**URTx_OS_MDS**)=0 :
If the URTx_RX input oversampling samples ①②③ are any one =0, then (**URTx_IR_BUSYF**) output 1 and output 0 for other conditions.
2. Oversampling mode selects "One" (**URTx_OS_MDS**)=1 :
If the URTx_RX input sample is 0, then (**URTx_IR_BUSYF**) output 1 and output 0 for input sample being 1.

16.23. UART DE 控制

UART 模块提供了 1 个数据使能信号 **URTx_DE** 并通过寄存器 **URTx_DE_EN** 使能。该信号用于标识数据发送周期并可输出到外部信号驱动设备。外部信号驱动设备可接收 UART TX 信号并通过信号增强缓冲器将其驱动到 UART 接收机目标以进行远程通信。

用户可通过 **URTx_DE_GT** 寄存器设置起始位之前和停止位之后的保持时间。**URTx_DE_INV** 寄存器用于反相 **URTx_DE** 输出信号。

图 16-41. UART 驱动使能时序



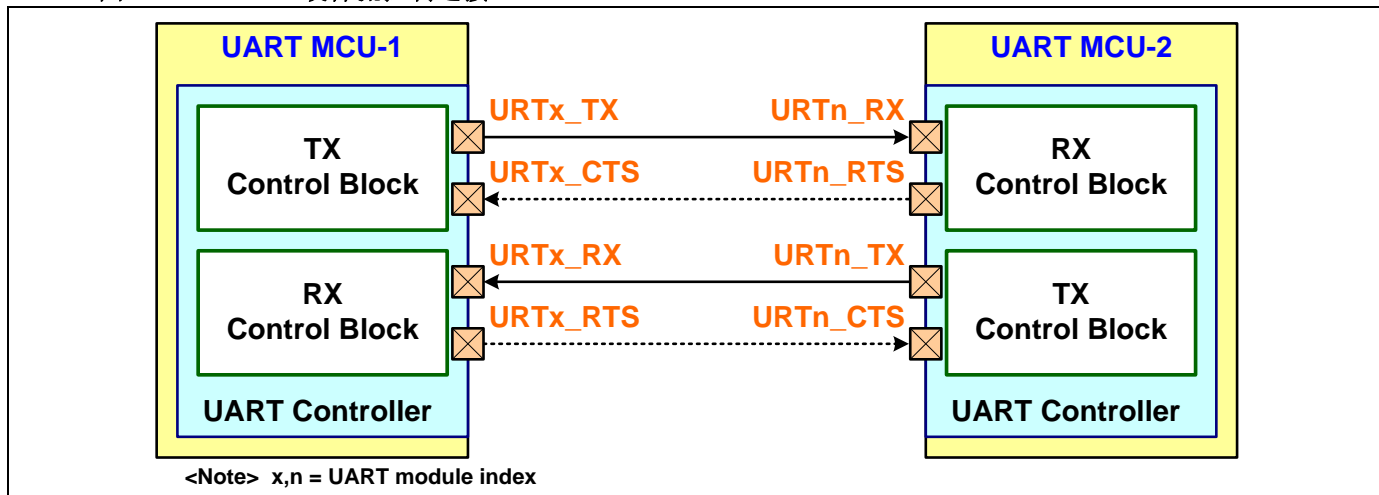
16.24. UART 硬件流控制

UART 支持硬件流控制功能用于数据传输，并提供了 **URTx_CTS**(清除发送)和 **URTx_RTS**(请求发送) 2 种控制信号用于硬件流控制。

16.24.1. UART 硬件流控制连接

下面的图表展示了 UART 硬件流控制连接。

图 16-42. UART 硬件流控制连接



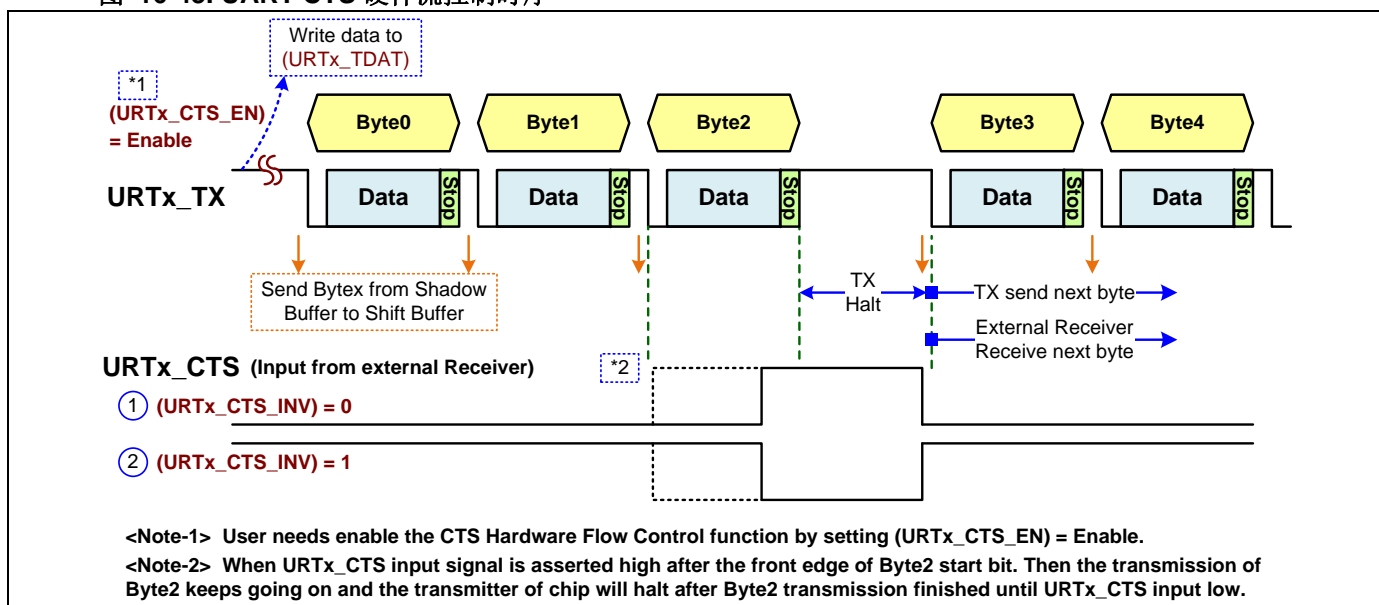
16.24.2. UART CTS 硬件流控制

CTS 信号是输入信号，置起时表明外部 UART 接收机不能再接收数据。UART 模块必须停止数据传输直到 CTS 信号被清除。

用户可通过 **URTx_CTS_EN** 寄存器使能硬件 CTS 输入功能。**URTx_CTS_INV** 寄存器用于反相 **URTx_CTS** 输出信号。

用户可通过读 **URTx_CTS** 寄存器位获得 CTS 线的实时状态。

图 16-43. UART CTS 硬件流控制时序



16.24.3. UART RTS 硬件流控制

在数据接收的过程中，若 UART 模块所有的数据缓冲已满时，RX 会发生溢出，ROVRF (**URT_x_ROVRF**)标志也会被置起，用户需要管理该状态。

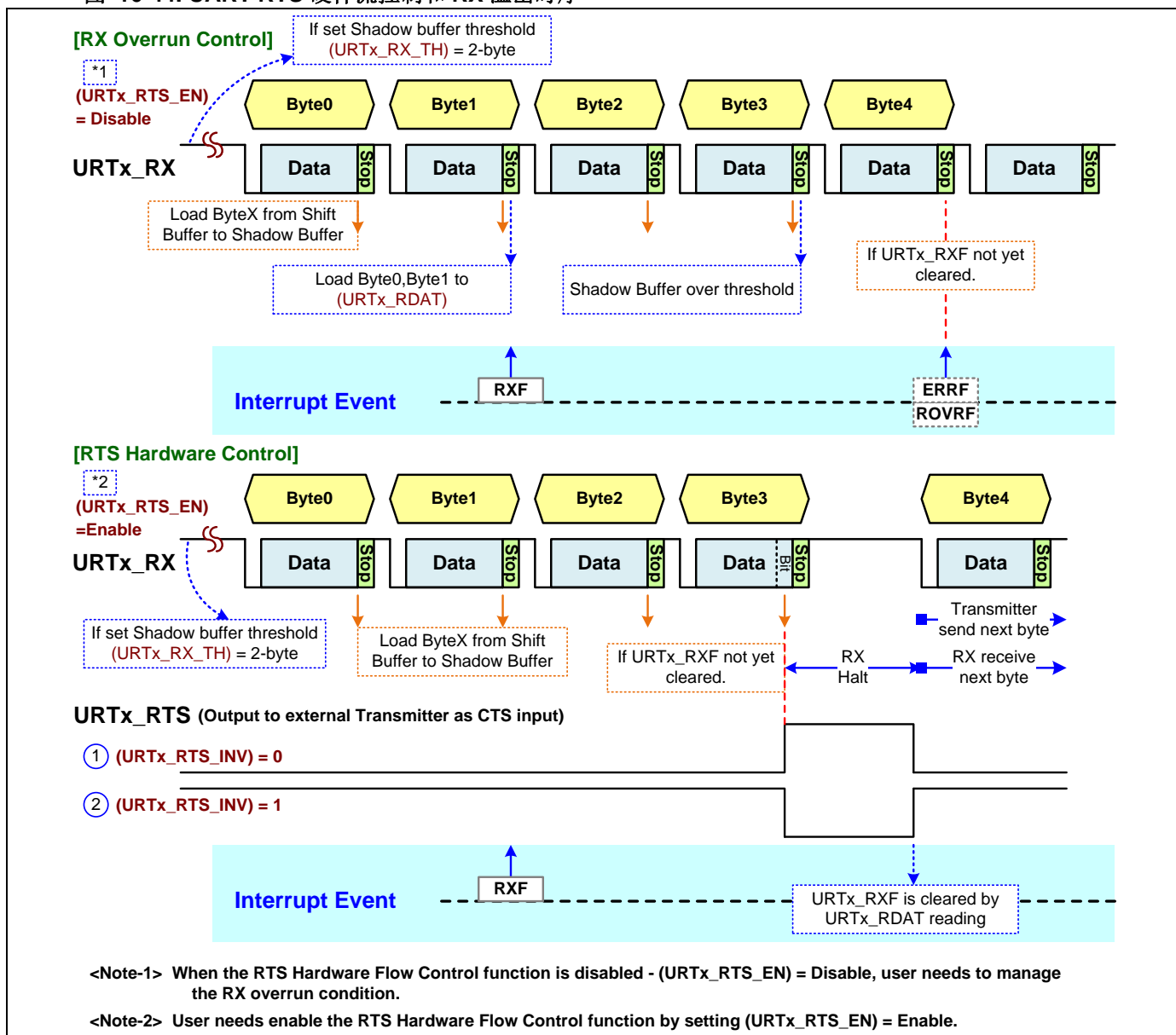
当外部发送者支持 CTS 数据流控制，用户可使用 RTS 硬件流控制功能自动硬件控制以避免 RX 数据溢出。

当所有的 UART 模块数据缓冲已满，它可输出 RTS 信号并表明外部 UART 发送者不再发送数据。外部 UART 发送者必须停止数据传输直到 UART 模块数据缓冲不为满，且 RTS 信号被清除。

用户可通过 **URT_x_RTS_EN** 寄存器使能硬件 RTS 输出功能。**URT_x_RTS_INV** 寄存器用于反相 **URT_x_RTS** 输出信号。

当通过 **URT_x_RTS_EN** 寄存器禁用了硬件 RTS 输出功能，用户可直接通过 **URT_x_RTS_OUT** 寄存器进行软件控制 **URT_x_RTS** 信号。

图 16-44. UART RTS 硬件流控制和 RX 溢出时序

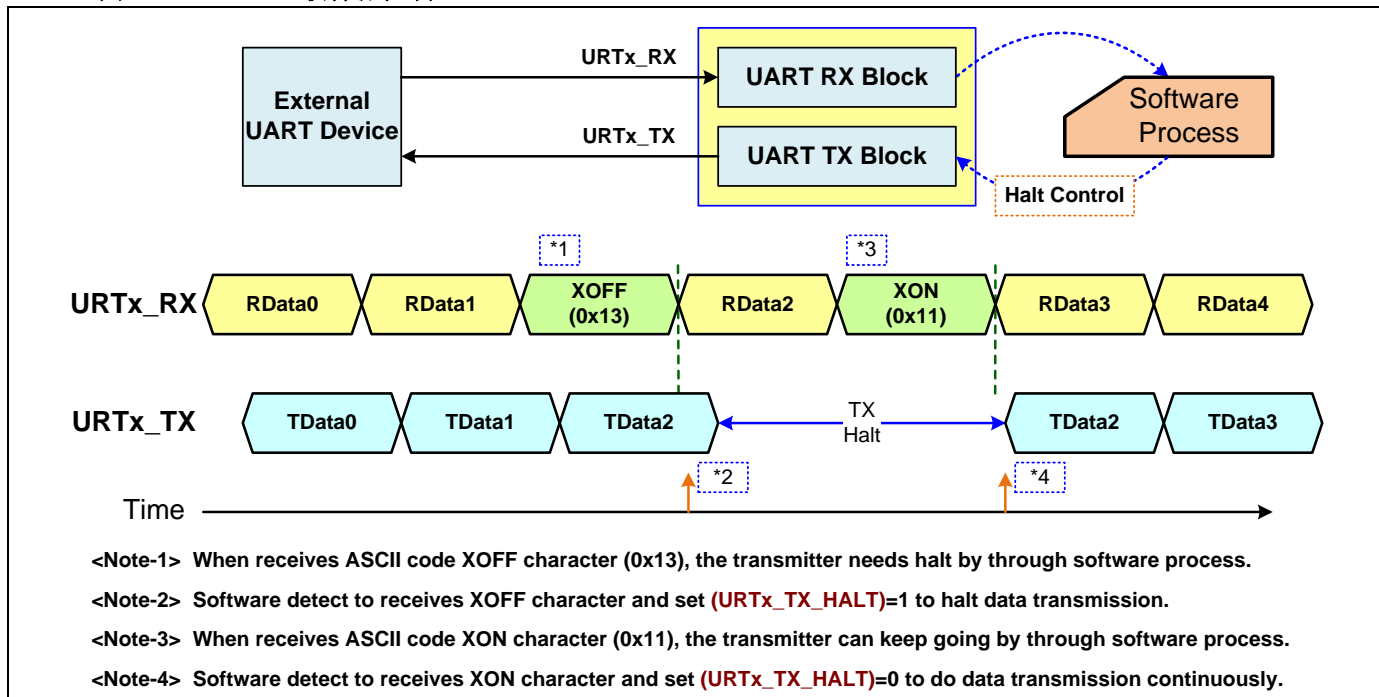


16.24.4. UART 软件流控制

UART 模块通过软件控制寄存器 **URT_x_TX_HALT** 做 UART 数据传输软件流控制用于 UART ASCII 码模式。当接收到 **XOFF** 字符(0x13)时, 用户可设置 **URT_x_TX_HALT** 寄存器停止数据发送; 接收到 **XON** 字符(0x11)时, 用户可清除 **URT_x_TX_HALT** 继续数据发送。

对于数据接收, 若 UART 模块的所有的接收数据缓冲已满, 用户可发送 **XOFF** 字符(0x13)提醒外部 UART 发送者停止数据发送, TX 溢出将会发生。此外用户可以发送 **XON** 字符(0x11)提醒外部发送者 TX 溢出状态已解除。

图 16-45. UART 软件流控制



16.25. UART 接收硬件停止和捕获控制

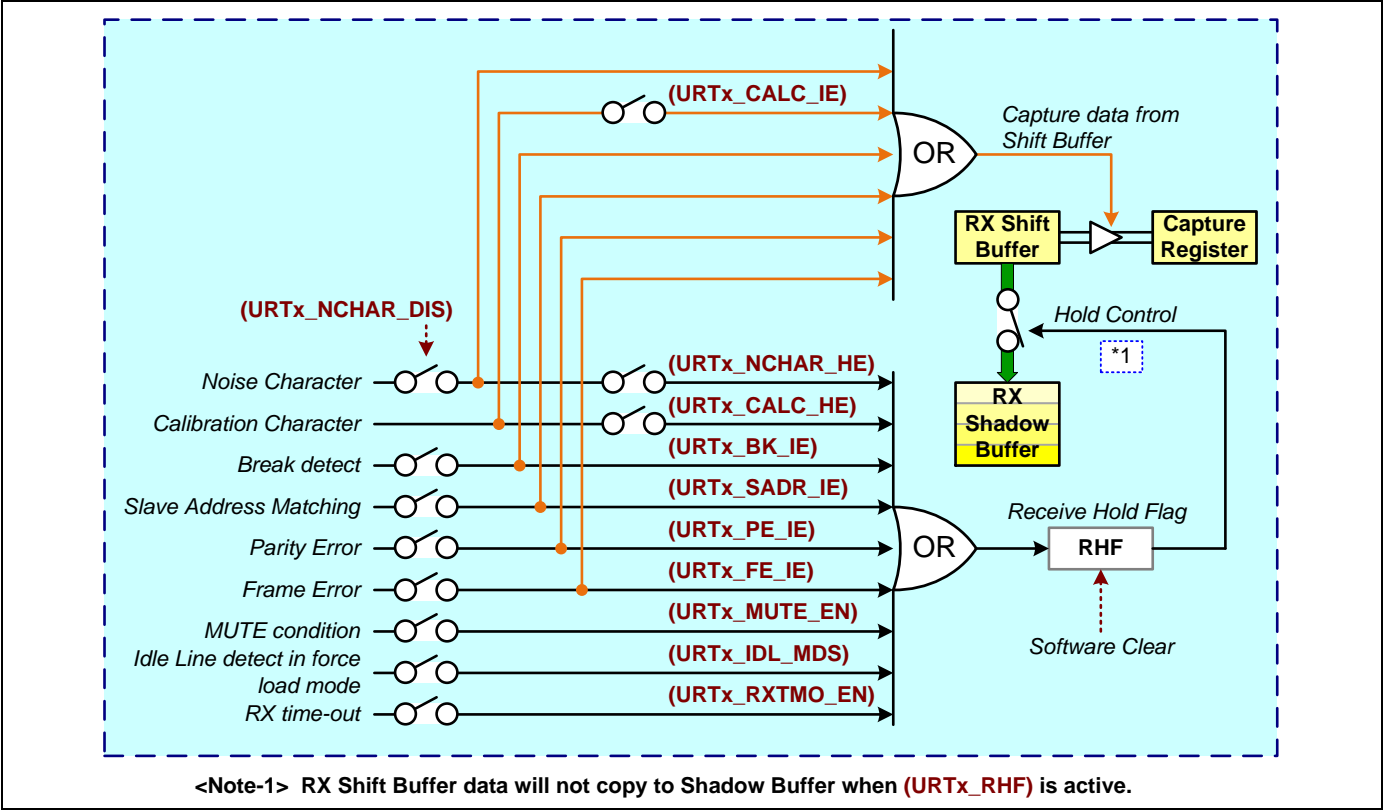
用于 UART 数据接收, UART 模块提供 1 个捕获寄存器并支持数据停止功能用于管理错误或意外状态。

在下面的图表中, 所有的 UART 事件都有一个相同的控制位来使能捕获数据到捕获寄存器和停止数据到阴影缓冲中。当然这些中断使能位也控制中断事件使能功能。特别是, 波特率校准完成和噪声发生事件的控制位是分开的。**URT_x_CALC_IE** 和 **URT_x_NCHAR_DIS** 寄存器用于捕获数据功能; **URT_x_CALC_HE** 和 **URT_x_NCHAR_HE** 寄存器用于数据停止功能。

捕获寄存器是用于捕获波特率校准完成、噪声事件发生、中止检测、从机地址匹配、校验错误和帧错误状态时, 接收数据(**URT_x_RCAP_DAT**), 停止位(**URT_x_RCAP_STP**), 校验位(**URT_x_RCAP_PAR**)和地址位(**URT_x_RCAP_ADR**)。这些状态可通过设置相关寄存器使能触发捕获功能, 如下图表所示。

数据停止功能用于在波特率校准完成、噪声事件发生、中止检测、从机地址匹配、校验错误、帧错误、静音进入完成、强制载入模式中 **Idle** 线检测和 **RX** 超时的情况下, 停止在 **RX** 移动缓冲中的接收数据载入阴影缓冲中。这些状态可通过设置相关寄存器使能停止功能, 如下图表所示。当任何数据停止触发事件发生且相关停止使能控制使能时, **RHF (URT_x_RHF)** 标志会置起, 接收数据会被停止载入阴影缓冲直到触发事件被释放。

图 16-46. UART 接收硬件和捕获停止事件



下面的表格展示了当多处理器地址不匹配时阴影缓冲和 RX 数据寄存器的 RHF 状态。RHF 会被硬件设置用于中止控制。请参照“[UART 多处理器](#)”和“[UART 静音模式控制](#)”节以获取更多关于多处理器模式的信息。

表 16-20. 多处理器地址不匹配时 UART 数据缓冲对比 RHF 状态

阴影缓冲(RX_LVL) & RDAT 寄存器(RNUM)	寄存器	RHF 标志状态	注释
	MUTE_AEN0		
都为满或空	x	0	阴影缓冲满是被 RX_TH 定义的
阴影缓冲有数据但是不满	0	0	RX_LVL & RNUM 不为 0 且 RX_LVL 不等于 RX_TH
	1	1	

16.26. UART DMA 操作

16.26.1. DMA 模块设置

当芯片支持 DMA（直接内存访问）控制器时，用户可以在 DMA 数据传输前，在 DMA 模块中设置关于来源/目的设备、通道请求仲裁等 DMA 设置。DMA 来源和目的可以是内存或外设。

参照 DMA 章以获取更多关于 DMA 模组设置的细节。

16.26.2. UART DMA 控制

DMA 设置完成后，用户需设置 UART 模块的 DMA 使能位 **URTx_DMA_RXEN** 和 **URTx_DMA_TXEN**。

最后，相关的通道请求起始位 **DMA_CHn_REQ** 是必须的，用于设置 DMA 传输启动(n = DMA 通道标号)。然后传输源和目的设备会置起 RX/TX 请求信号到 DMA 控制器，DMA 控制器便会置起接收信号到请求源/目的设备。此时，该数据传输连接是用于 DMA 传输的。

● UART RX 到 DMA

URT_x_DMA_RXEN 寄存器位是用于使能来自 UART 接收的数据通过 DMA 发送到 DMA 目的地。
DMA 发送周期中，接收数据标志 **URT_x_RXF** 是被硬件屏蔽的。

● UART TX 通过 DMA

URT_x_DMA_TXEN 寄存器位是用于使能 DMA 数据寄存器的源到 UART 发送输出的 DMA 发送。

在 DMA 发送周期中，发送标志 **URT_x_TXF** 是被硬件屏蔽的。通常，发送完成标志 **URT_x_TCF** 会在 DMA 发送完成后被置起。

当硬件检测到其中 1 种错误或特殊情况，用户可设置 **URT_x_DDTX_EN** 寄存器禁用 DMA TX 功能，同时，硬件会清除 **URT_x_DMA_TXEN** 和 **URT_x_DMA_RXEN** 位。

16.26.3. UART 的 DMA 中断标志控制

DMA 工作周期中，模块的中断标志会控制和执行 3 种类型，如下表格。其中一方在 DMA 工作过程中被屏蔽，另一方会在标志被置起后禁用 DMA 功能。此时，硬件会禁用 **URT_x_DMA_TXEN** 和 **URT_x_DMA_RXEN** 位。其他操作与 DMA 操作中不执行的操作相同。

表 16-21. UART DMA 功能中断标志控制

行为	DMA 操作中屏蔽标志	标志置起后 DMA 被禁用 (*1)	一般控制	
外设	(数据溢出标志)	(错误/检测标志)	(其他标志)	
URT _x	URT _x _TCF (*2) URT _x _RXF URT _x _TXF	URT _x _ERRF URT ₀ _BKF(*3) URT ₀ _IDLF(*4) URT ₀ _CTS(*5)	URT _x _UGF URT _x _LSF	URT ₀ _SADRF URT ₀ _BRTF URT ₀ _TMOF URT ₀ _CALCF URT ₀ _PEF URT ₀ _FEF URT ₀ _NCEF URT ₀ _ROVRF URT ₀ _TXEF URT ₀ _RXTMOF URT ₀ _IDTMOF URT ₀ _BKTMOF URT ₀ _CALTMOF

注释-1：当标志被置起，若相关中断使能位没被使能，它不会强行禁用外设 DMA。

注释-2：URT_x_TCF 会在 DMA TX 完成后置起。

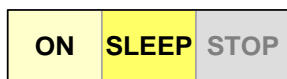
注释-3：当标志被置起，若相关中断使能位被使能，会强制禁用 URT_x DMA RX，同时，根据 URT_x_DDTX_EN 寄存器可设置是否禁用 URT_x DMA TX。

注释-4：当标志被置起，若相关中断使能位被使能，URT_x DMA RX 会被禁用，但是 URT_x DMA TX 不会。

注释-5：当标志被置起，若相关中断使能位被使能，URT_x DMA TX 会被禁用，但是 URT_x DMA RX 不会。通常用户会设置 CTS 硬件自动控制模式(URT_x_CTS_EN=1)，且不会使能 URT_x_CTS_IE，当检测到 CTS 置起时，URT_x 会停止发送且不会禁用 URT_x DMA TX。

17. SPI (串行外设接口)

17.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片提供了高速串行外设接口（SPI）。SPI 是一种全双工、高速、同步通信总线，具有主机模式和从机模式两种工作模式。

SPI 模块内置阴影缓冲器和数据寄存器用于独立地发送和接收，以提高发送和接收通信性能。

注释：（x = 模块标号）会被用于该章的寄存器、信号和引脚/端口描述中。[EX]: **SPIx_EN** ~ x 表示模块标号。

17.2. 特性

- 支持 1 个 SPI 模块 – SPI0
- 支持主机和从机模式
 - 支持全双工、半双工或单工通信方式
 - 支持无 NSS（从机选择信号）通信方式
- 可选择 4~32 位帧大小
 - 支持 4 字节数据缓冲器和 32 位数据寄存器用于高速数据通信
- 可用 DMA 收发数据
- 支持多主机处理
- 可选择时钟极性和相位
- 可选择 MSB 或 LSB 数据顺序
- 用于主机的 NSS 线软硬件管理
- 可设置数据传输模式
 - 标准 SPI 模式（分离的发送和接收线）
 - 具有双向数据传输的单线 SPI 模式
 - 具有双向数据传输的双线 SPI 模式
 - 具有双向数据传输的四线 SPI 模式
 - 支持数据复制模式用于双线/四线 SPI 模式
- 数据发送/接收溢出检测
- 支持硬件主机模式故障检测和自动从机模式改变

❖ MG32F02A132/072

- 支持可设置时钟速率控制
 - 最大支持时钟速率：主机 12MHz，从机 6MHz
- 可设置数据传输拓展模式
 - 具有双向数据传输的八线 SPI 模式

❖ MG32F02A032

- 支持可设置时钟速率控制
 - 最大支持时钟速率：主机 24MHz，从机 16MHz

17.3. 配置

17.3.1. 芯片配置

下面的表格展示了芯片 SPI 配置。

表 17-1. SPI 配置

芯片	SPI 模块
	SPI0
MG32F02A132	V
MG32F02A072	V
MG32F02A032	V

注释 V:包含

17.3.2. 模块功能

下面的表格展示了 SPI 模块包含的功能。

表 17-2. SPI 模块功能

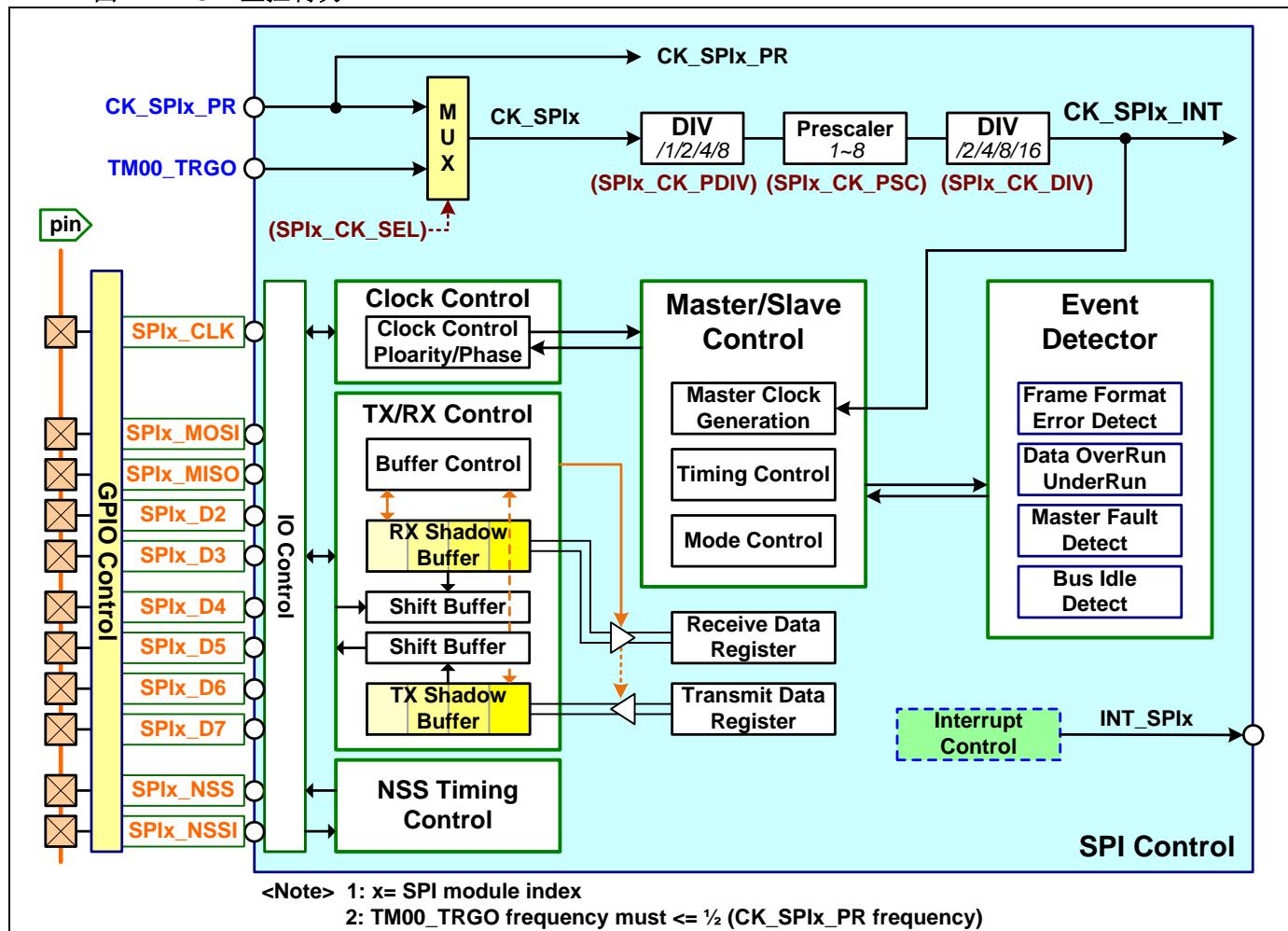
模块功能	芯片	MG32F02A132 MG32F02A072	MG32F02A032	注释
		SPI0	SPI0	
SPI 标准模式		yes	yes	2 数据线 全双工通信
SPI 1 线模式		yes	yes	1 数据线 半双工通信
SPI 2 线模式		yes	yes	2 数据线 半双工通信
SPI 4 线模式		yes	yes	4 数据线 半双工通信
SPI 8 线模式		yes	-	8 数据线 半双工通信
数据线复制模式		yes	yes	所有的数据线都是相同的数据用于 2/4 数据线模式
DTR 模式		-	yes	双传输速率模式
CLK 引脚		1	1	时钟信号
数据引脚		8	4	MOSI(D0),MSIO(D1),D[2..7]信号用于全/半双工
NSS 引脚		1	1	NSS 线软硬件管理用于主机模式
NSSI 引脚		1	1	额外的 NSS 输入用于多主机应用
主机模式		yes	yes	
从机模式		yes	yes	
阴影缓冲		4 字节	4 字节	RX/TX 32 位
Msb/Lsb 发送选项		yes	yes	
可设置数据位大小		4~32 位	4~32 位	4~32 位
可设置时钟相位和极性		yes	yes	
硬件 NSS 控制		yes	yes	主从机硬件控制
NSS 脉冲模式		yes	yes	可设置两个顺序数据帧之间的电平
可设置 NSS 电平宽度		-	yes	
模式错误检测		yes	yes	主机模式失败/改变检测
总线空闲检测		yes	yes	从机模式总线空闲检测
接收溢出检测		yes	yes	
发送下溢检测		yes	yes	从机模式数据发送下溢
发送写错误检测		yes	yes	从机模式 NSS 无效中止; 位计数错误
DMA 支持		yes	yes	

17.4. 控制块

下面的图表展示了 SPI 控制块。

[注释]: **SPIx_D[7:4]** 不支持于 MG32F02A032。

图 17-1. SPI 主控制块



17.5. IO 线

17.5.1. IO 信号

- **SPIx_CLK**

SPI 时钟信号，用于 SPI 主机模式输出或 SPI 从机模式输入。

- **SPIx_MOSI**

SPI 数据信号，当运行于 SPI 标准双线全双工模式时作为 SPI 主机模式输出或 SPI 从机模式输入。当运行于 SPI 半双工通信模式，该信号作为 SPI 数据线 0(SPIx_D0)，并作为双向 IO 用于 SPI 主从机模式。

- **SPIx_MISO**

SPI 数据信号，当运行于 SPI 标准双线全双工模式时作为 SPI 主机模式输入或 SPI 从机模式输出。当运行于 SPI 半双工通信模式，该信号作为 SPI 数据线 1(SPIx_D1)，并作为双向 IO 用于 SPI 主从机模式。

- **SPIx_D[2..7]**

这些信号是 SPI 数据信号。这些信号作为 SPI 数据线 2~7 并作为双向 IO 用于 SPI 主从机模式。

[注释]: **SPIx_D[4..7]** 不支持于 MG32F02A032。

- **SPIx_NSS**

SPI 从机选择或片选信号，该信号作为 SPI 主机模式的输出或 SPI 从机模式的输入。

- SPIx_NSSI

SPI 从机选择或片选输入信号，该信号用于 SPI 从机模式的或多主机应用。

17.5.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。参照用户手册 GPIO 章中“[IO 模式](#)”节以获取更多关于 IO 模式设置的信息。

每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“[功能复用选择](#)”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“[引脚功能复用表](#)”以获取更多信息。

17.6. 使能和时钟

17.6.1. SPI 全局使能

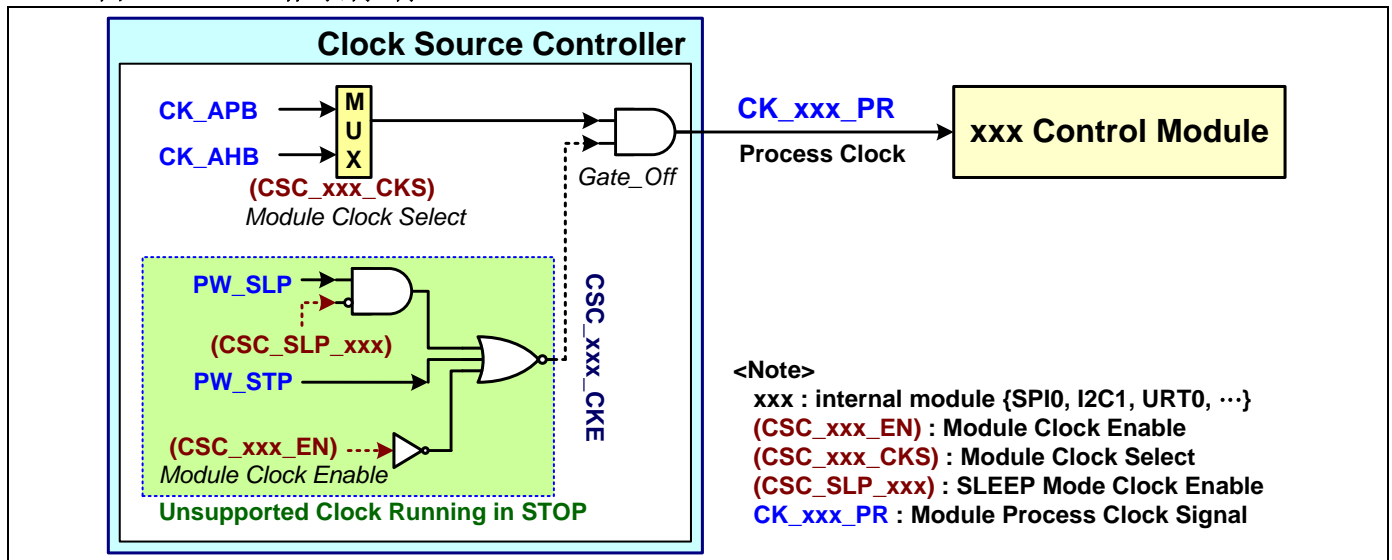
SPI 模块的所有功能全局使能位是 **SPIx_EN**。当该位被禁用时，所有的 SPI 功能将无法工作。

17.6.2. SPI 时钟控制

- 工作时钟

该模块工作时钟 **CK_SPIx_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 CSC（时钟源控制器）模块。该时钟可通过 **CSC_SPIx_EN** 寄存器使能并通过 **CSC_SPIx_CKS** 寄存器选择时钟源来自 APB 或 AHB。用户可以在芯片进入 **SLEEP** 模式之前通过设置 **CSC_SLP_SPIx** 寄存器规划在 **SLEEP** 模式下是否让 SPI 时钟继续运行。参照系统时钟章以获取更多信息。

图 17-2. SPI 工作时钟控制



- 模块内部时钟

SPI 模块可输出内部时钟 **CK_SPIx_INT** 作为数据接收信号的采样时钟和发送数据信号的时钟源。用户可通过 **SPIx_CK_SEL** 寄存器选择时钟源是来自模块工作时钟 **CK_SPIx_PR** 还是定时器触发输出信号 **TM00_TRGO**。此外，模块提供了 1 个时钟预分频器和 1 个时钟分频器和 1 个后分频器用于产生内部时钟 **CK_SPIx_INT**。相关的寄存器为 **SPIx_CK_PDIV**、**SPIx_CK_PSC** 和 **SPIx_CK_DIV**。

[注释]: 通常内部时钟频率需比模块工作时钟慢至少 1/2。

17.7. 中断和事件

17.7.1. SPI 中断控制和状态

I2C 模块中有 1 种信号 **INT_SPIx**。**INT_SPIx** 发送到外部中断控制器（EXIC）作为中断事件。

中断标识是用于中断服务程序（ISR）流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成

时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **SPIx_IEA** 用于使能和禁用该模块的所有中断源。

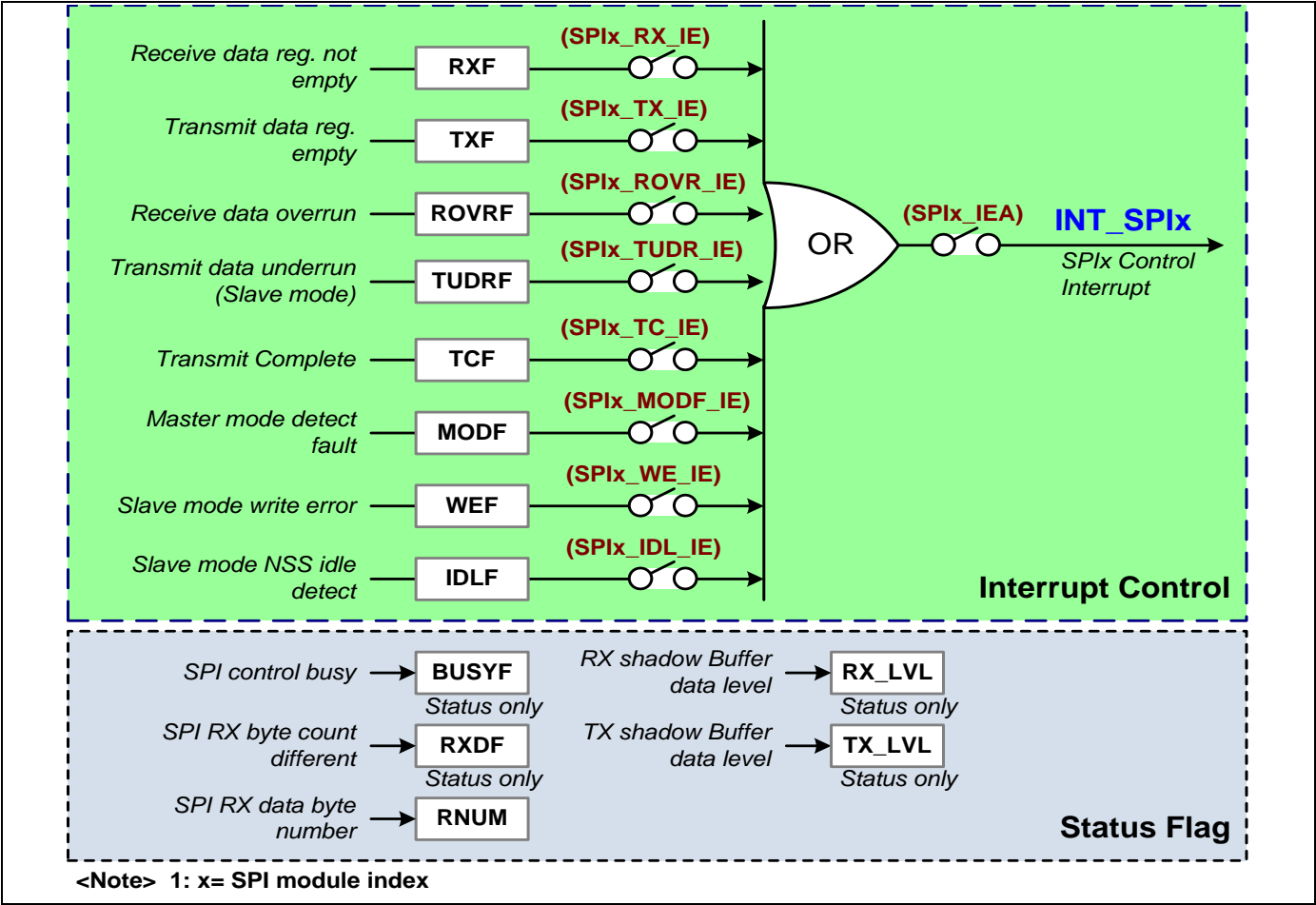
该模块中有一些只读的状态位，用于提供内部控制状态，其中一个占用标志 **SPIx_BUSYF** 表明数据传输繁忙状态。参照相关状态位寄存器描述以获取更多信息。

17.7.2. SPI 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

下面的图表展示了 SPI 状态和中断控制块。

图 17-3. SPI 状态和中断



- IDLF

SPI 从机模式 NSS 空闲检测标志是(**SPIx_IDLF**)，相关的中断使能寄存器位是 **SPIx_IDL_IE**。

- TCF

SPI 发送完成标志是(**SPIx_TCF**)。当阴影缓冲和数据寄存器都为空且移动缓冲移动完成，则置起该标志。相关的中断使能寄存器位是 **SPIx_TC_IE**。

- RXF

SPI 接收数据寄存器不为空标志是(**SPIx_RXF**)，相关的中断使能寄存器位是 **SPIx_RX_IE**。当接收数据缓冲等级 **SPIx_RX_LVL** 大于等于阴影缓冲阈值 **SPIx_RX_TH** 设置，该标志置起，数据缓冲会复制到数据寄存器 **SPIx_RDAT** 中。当 **SPIx_RDAT** 被读或该位被软件写 1 时清除。

注释: 在 MG32F02A132/072 中，当通过 SWD 调试读取 SPIx_RDAT 时，RXF 标志会被清除。

注释: 在 MG32F02A032 中，当通过 SWD 调试读取 SPIx_RDAT 时，RXF 标志不会被清除。

- TXF

SPI 发送数据寄存器为空标志是(**SPIx_TXF**)，相关的中断使能寄存器位是 **SPIx_TX_IE**。当发送阴影缓冲为空时，数据寄存器的内容 **SPIx_TDAT** 会复制到阴影缓冲并置起该标志。当 **SPIx_TDAT** 被写或该位被软件写 1 时清除。

- MODF

SPI 模式检测错误标志是(**SPIx_MODF**)。对于主机模式，该标志会在 NSS 信号被外部设备拉低时置起。相关的中断使能寄存器位是 **SPIx_MODF_IE**。参照“[SPI 主机模式改变检测](#)”节以获取更多信息。

- WEF

SPI 从机模式写错误标志是(**SPIx_WEF**)。在数据发送完成之前，主机通过拉高 NSS 信号停止主机读取时会置起该错误。数据发送的位大小是在 **SPIx_DSIZE** 寄存器中定义的。相关的中断使能寄存器位是 **SPIx_WE_IE**。

- ROVRF

SPI 接收溢出标志是(**SPIx_ROVRF**)，相关的中断使能寄存器位是 **SPIx_ROVR_IE**。

- TUDRF

SPI 从机模式发送下溢标志是(**SPIx_TUDRF**)，相关的中断使能寄存器位是 **SPIx_TUDRF_IE**。

17.8. SPI 模式 IO 控制

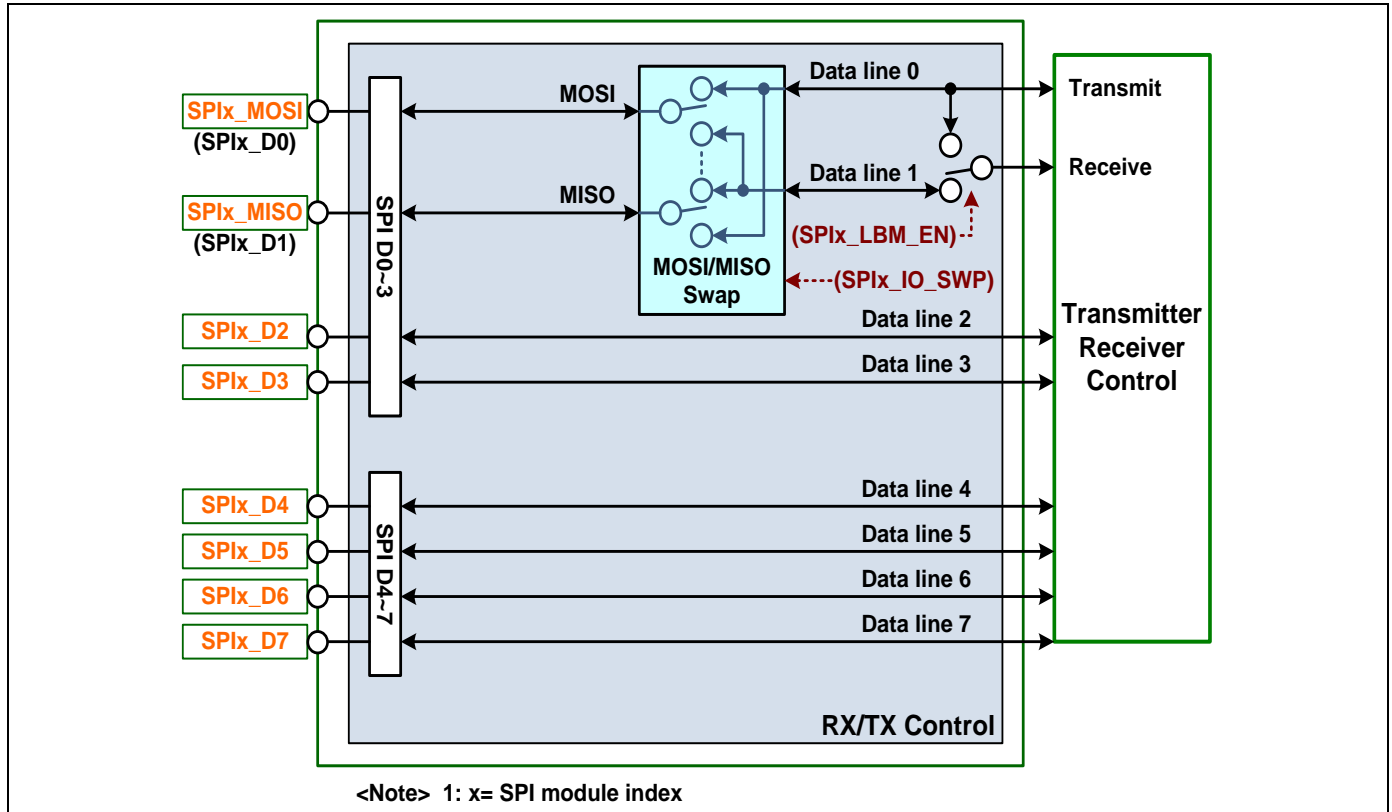
该模块提供最多 8 个数据信号—**SPIx_D[7:0]**，1 个时钟信号 —**SPIx_CLK**，1 个从机选择输入/输出信号—**SPIx_NSS** 和 1 个额外从机选择输入信号—**SPIx_NSSI**。**SPIx_D0** 和 **SPIx_D1** 信号也被命名为 **SPIx_MOSI** 和 **SPIx_MISO** 作为 SPI 标准通信的 SPI 主机输出/从机输入数据信号和 SPI 主机输入/从机输出数据信号。

[注释]: **SPIx_D[7:4]** 不支持于 MG32F02A032。

17.8.1. SPI IO 控制

SPIx_IO_SWP 寄存器用于使能 **SPIx_MOSI** 和 **SPIx_MISO** 信号的调换。

图 17-4. SPI 数据 IO 控制



在实际应用中，用户可通过设置 **SPIx_CKO_MUX** 寄存器选择 **SPIx_CLK** 时钟输出源。**MWE** 和 **MOE** 信号来源于 EMB（外部内存总线）模块。所有的 **TMx_CKO** 信号都来源于 TMx 定时器模块。

SPIx_NSSO_INV 和 **SPIx_NSSI_INV** 寄存器用于反相相关的信号。用户可通过 **SPIx_NSSI_SEL** 寄存器设置选择 NSS 信号输入来源于 **SPIx_NSS** 或 **SPIx_NSSI**。

SPIx_NSS_SWEN 寄存器用于软件控制使能 **SPIx_NSS** 和 **SPIx_NSSI** 输出信号。当使能时，用户可通过 **SPIx_NSS_SWO** 寄存器设置直接进行 **SPIx_NSS** 控制输出。**SPIx_NSSI_SWEN** 寄存器则用于软件控制使能 **SPIx_NSS** 或 **SPIx_NSSI** 输入信号。当使能时，用户可通过读 **SPIx_NSSI_SWO** 寄存器直接获取 **SPIx_NSS** 或 **SPIx_NSSI** 输入状态。

[注释]: **SPIx_NSSI_SWEN** 不支持于 MG32F02A132/072。当 **SPIx_NSS_SWEN** 寄存器在 MG32F02A132/072 中被使能，用户可通过 **SPIx_NSS_SWO** 寄存器设置直接进行 **SPIx_NSS** 控制输出，并可通过读 **SPIx_NSS_SWI** 寄存器直接获取 **SPIx_NSS** 或 **SPIx_NSSI** 输入状态。

图 17-5. SPI 时钟/NSS IO 控制 – MG32F02A132/072

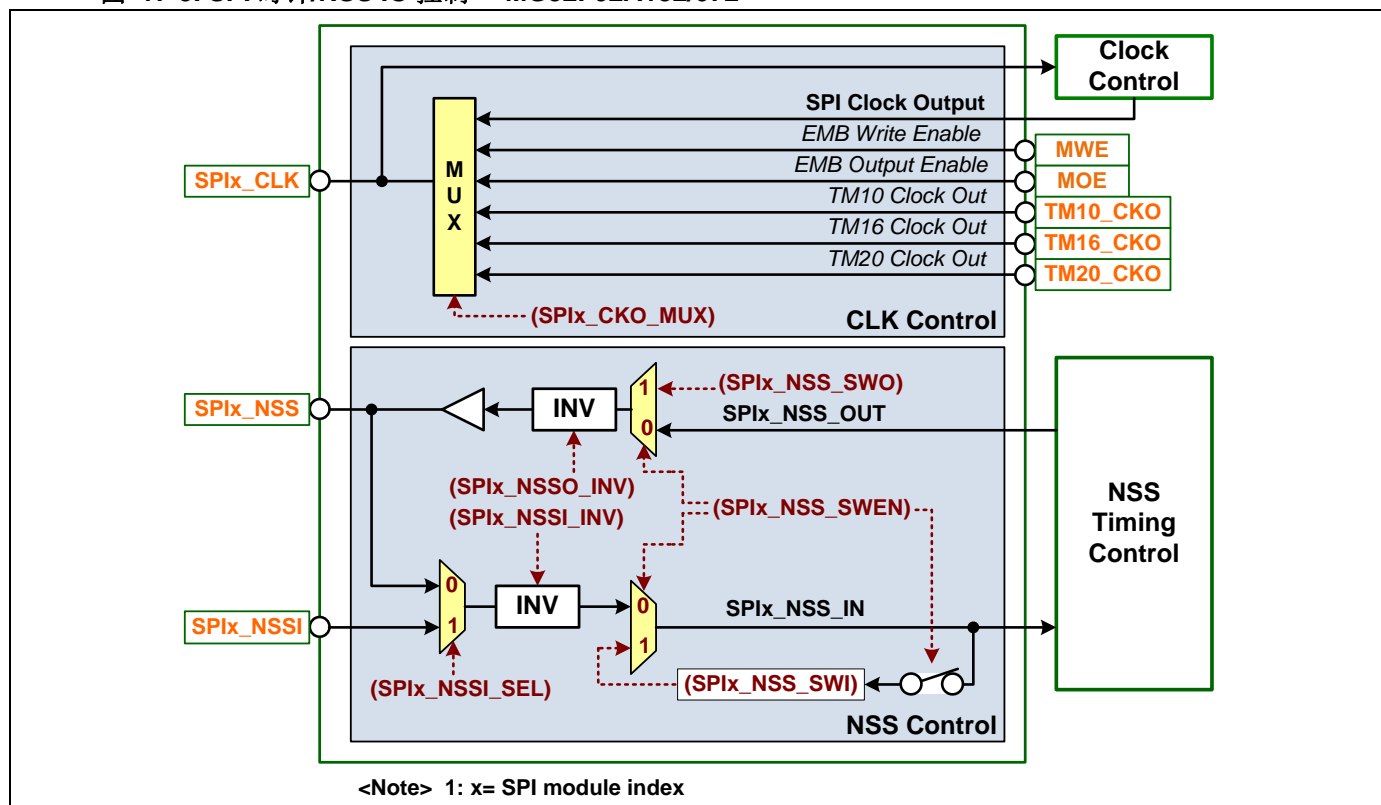
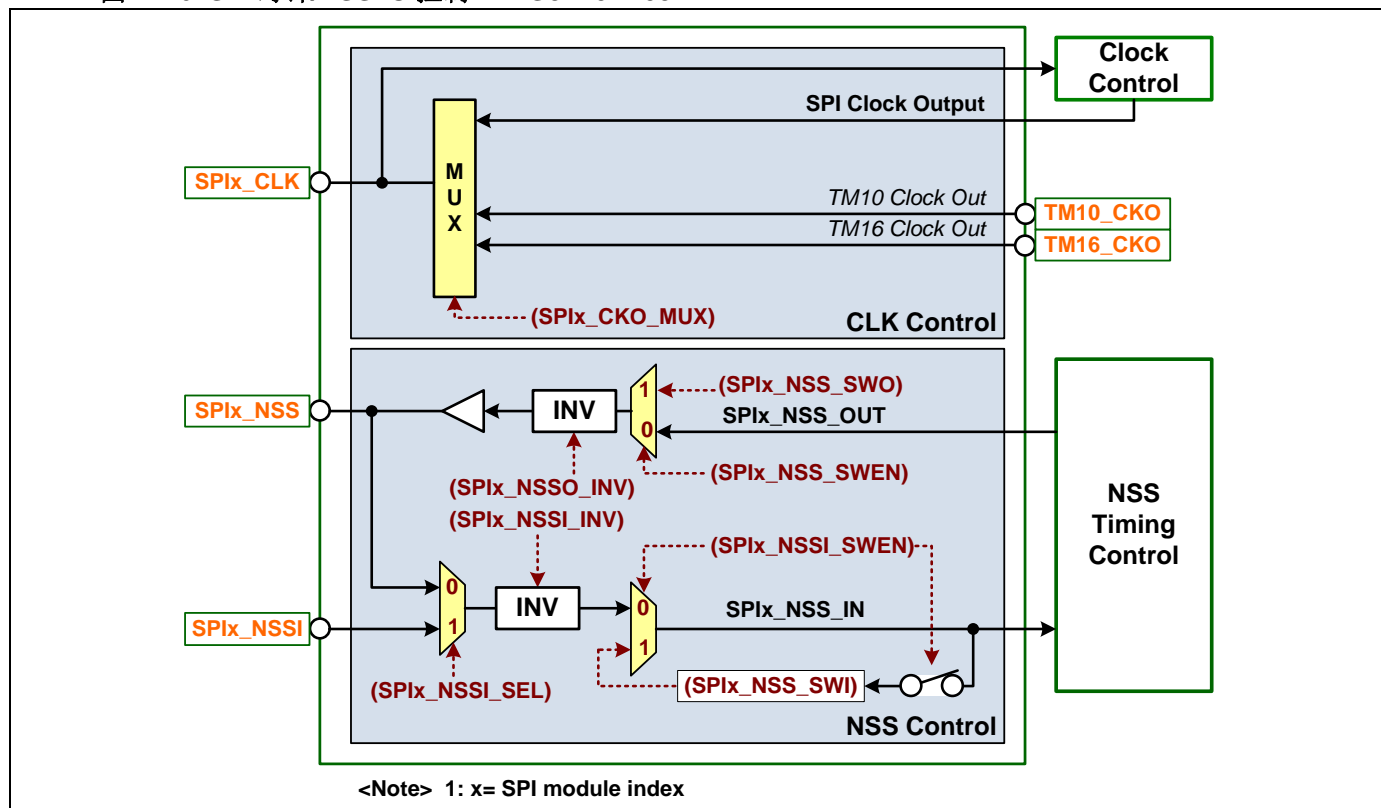


图 17-6. SPI 时钟/NSS IO 控制 – MG32F02A032



17.8.2. SPI IO 模式

通常，用户可设置使用的 GPIO 引脚为推挽或开漏输出用于 SPI 数据信号，此外，用户可设置使用的 IO 引脚模式为数字输入或开漏输出作为 SPI 输入信号。对于 SPI 双向信号，使用的 IO 引脚模式需为推挽或开漏输出，当选择推挽模式，SPI 数据信号可在输出时推挽，输入或空闲状态时为开漏模式；当选择开漏模式，SPI 数据信号将

在空闲、输入、输出状态下保持开漏模式。

17.8.3. SPI 循环模式

SPI 模块通过 **SPIx_LBM_EN** 寄存器设置内建 1 个循环模式用于内部自测试。当使能时，接收到的输入信号取自发送输出，并替代输入引脚的接收输入。SPI 循环模式只支持于标准 SPI 模式。

17.9. SPI 应用连接

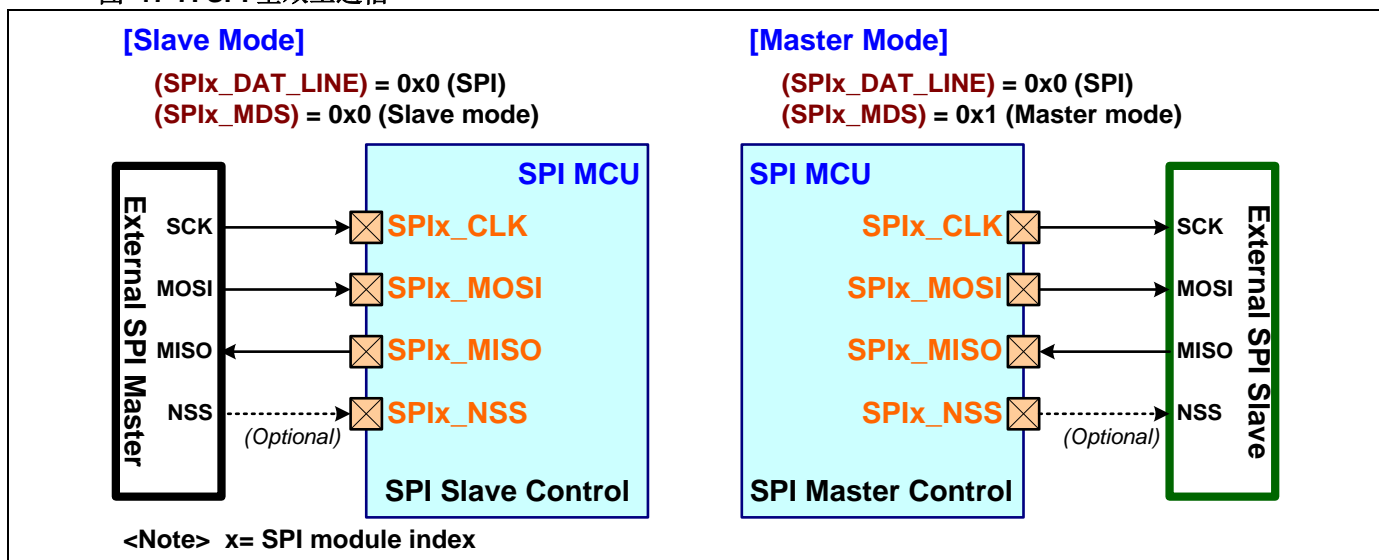
下面的图表展示了 SPI 全双工、单工、半双工、多从机、双主机和主机与从机调换通信的应用连接。

以下描述中，**SCK** 是 SPI 时钟信号；**NSS** 是 SPI 从机选择信号。

17.9.1. 全双工通信

一共有 4 个连接的信号：**SCK**, **MOSI**（主出从入）, **MISO**（主入从出）和可选的 **NSS**。 **MOSI** 数据信号是由主到从的单向信号， **MISO** 数据信号是由从到主的单向信号。该连接可做全双工通信。

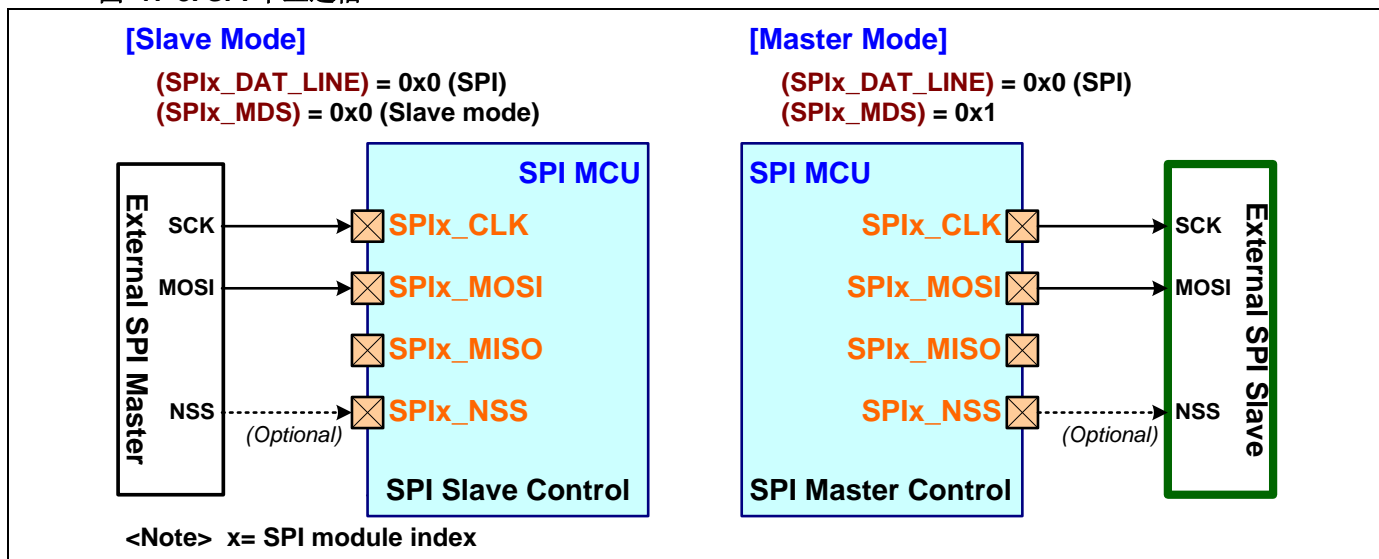
图 17-7. SPI 全双工通信



17.9.2. 单工通信

一共有 3 个连接的信号：**SCK**, **MOSI**（主出从入）和可选的 **NSS**。 **MOSI** 数据信号是由主到从的单向信号，该连接只可做单工通信。

图 17-8. SPI 单工通信



17.9.3. 半双工通信

以下连接方式可做半双工通信。

- 单数据线

一共有 3 个连接的信号：**SCK,MOMI**（主出主入），或 **SOSI**（从出从入），可选的 **NSS**。**MOMI** 或 **SOSI** 数据信号是主从机之间的双向信号。

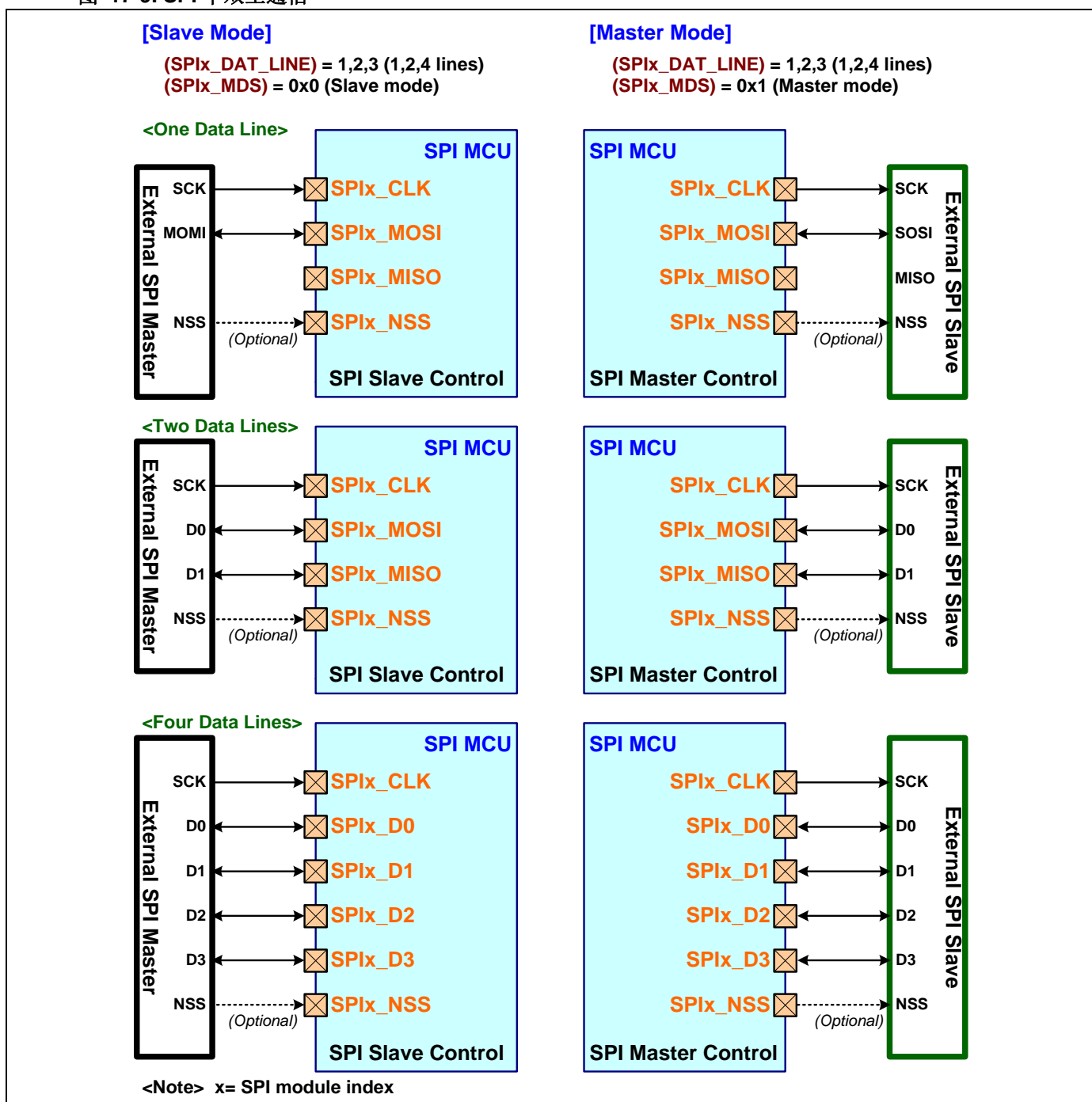
- 双数据线

一共有 4 个连接的信号：**SCK,D[1:0]**（数据信号），可选的 **NSS**。**D[1:0]** 数据信号是主从机之间的双向信号。

- 四数据线

一共有 6 个连接的信号：**SCK,D[3:0]**（数据信号），可选的 **NSS**。**D[3:0]** 数据信号是主从机之间的双向信号。

图 17-9. SPI 半双工通信

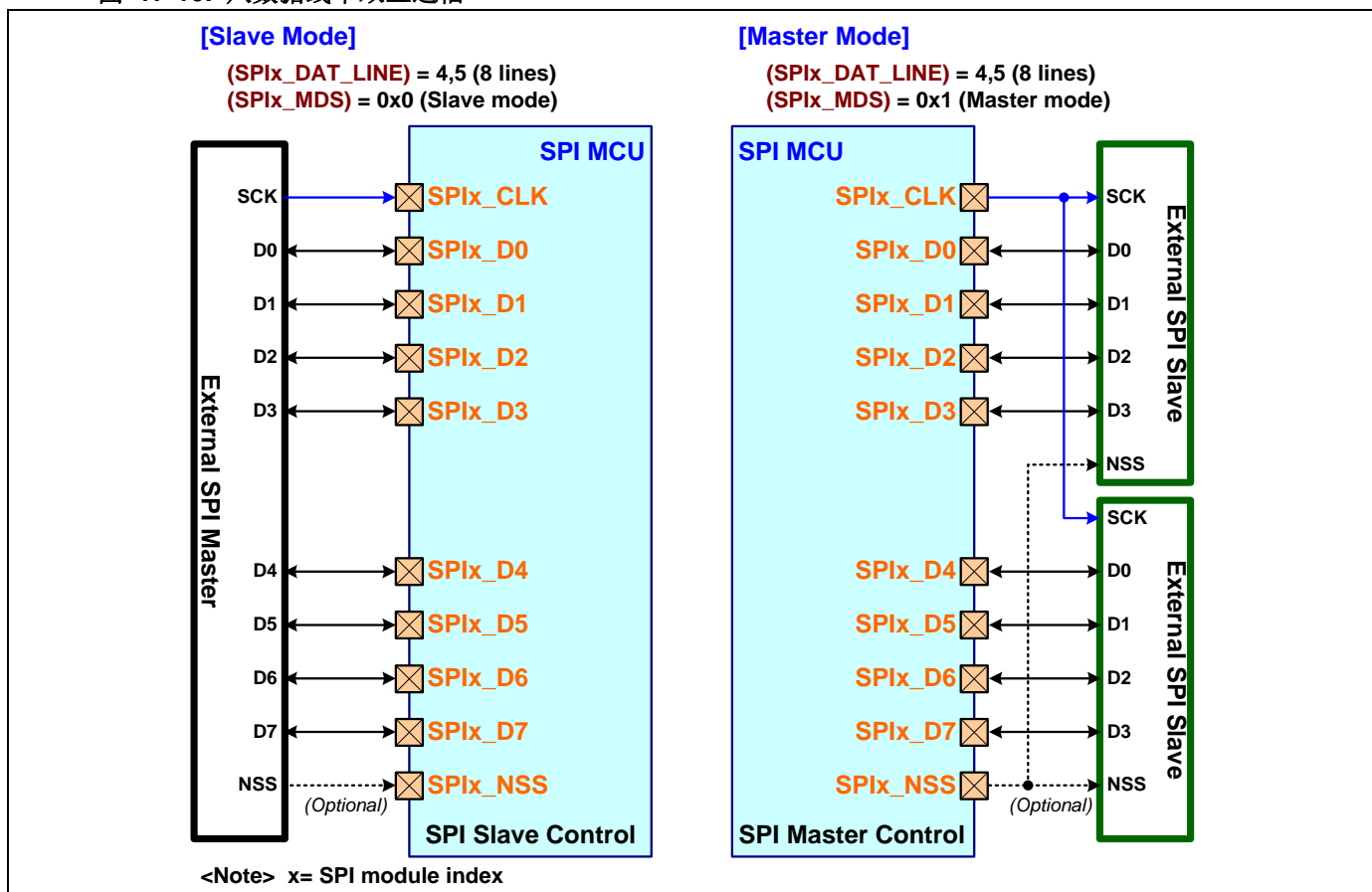


- 八数据线

一共有 10 个连接的信号：**SCK,D[7:0]**（数据信号），可选的 **NSS**。**D[7:0]** 数据信号是主从机之间的双向信号。

[注释]:八线数据线模式不支持于 MG32F02A032。

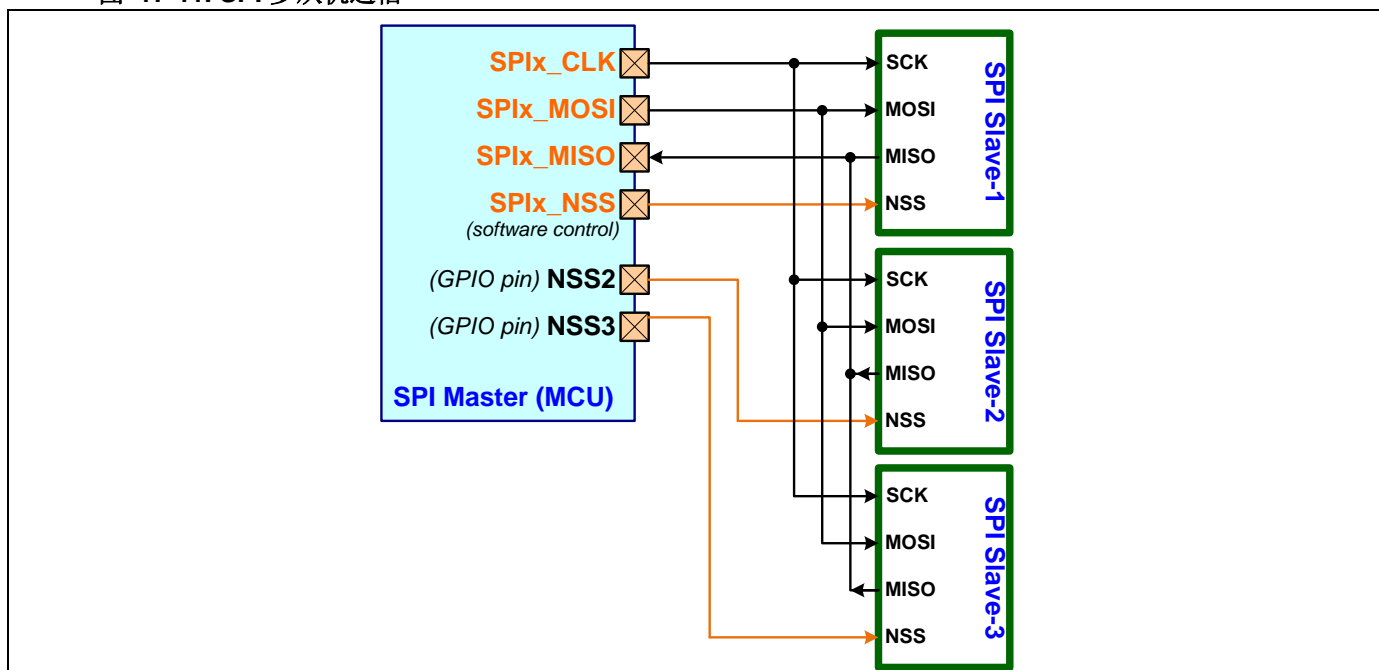
图 17-10. 八数据线半双工通信



17.9.4. 多从机通信

该连接为 1 个 SPI 主机和多 SPI 从机的连接方式。根据从机的数量，用户需增加额外的 NSS 信号线(NSS2, NSS3, ...)用于从机选择控制。

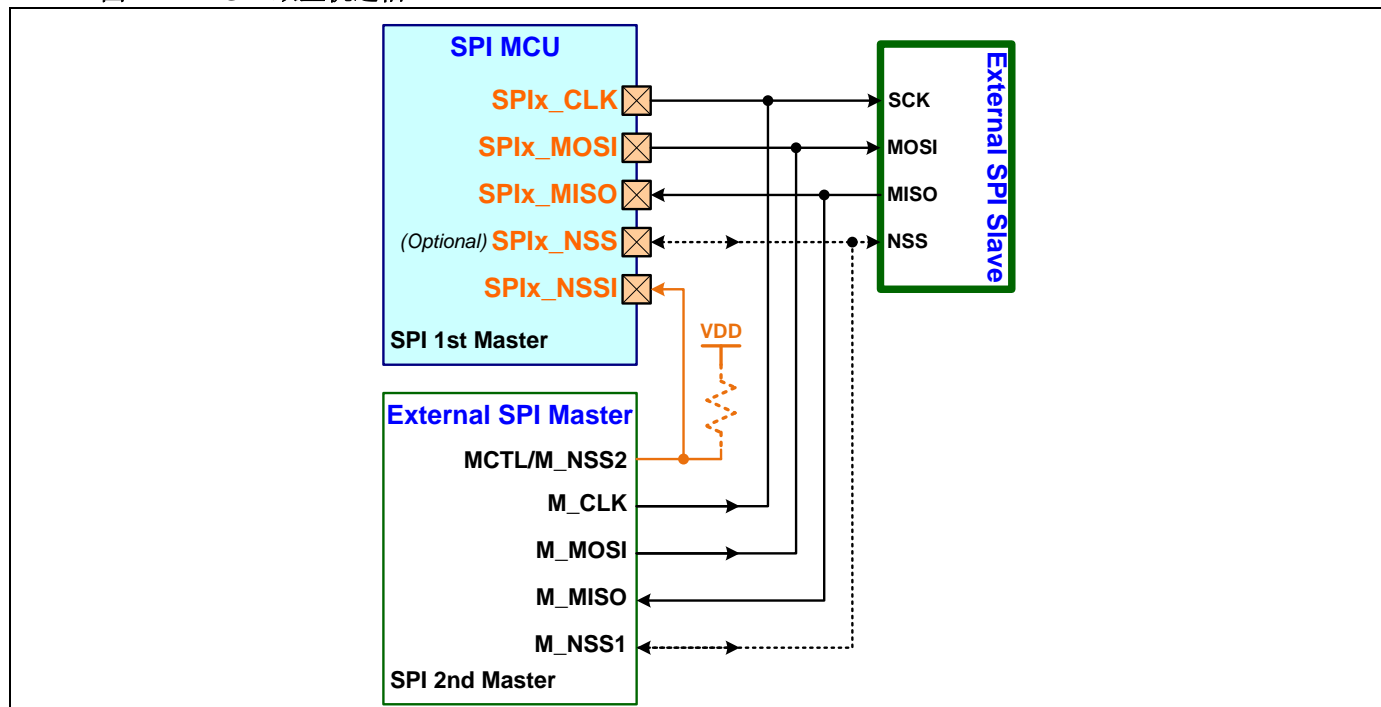
图 17-11. SPI 多从机通信



17.9.5. 双主机通信

该连接为 2 个 SPI 主机和 1 个 SPI 从机的连接方式。1 个额外的 **NSSI** 信号用于通过 2nd 主机提醒 1st 主机释放 SPI 总线。当 **NSSI** 置起，1st 主机需禁用和释放 SPI 总线，然后从机就会空闲下来且可被 2nd 主机控制。当 2nd 主机结束 SPI 传输，它可通过解除 **NSSI** 释放总线，然后 1st 主机可重新得到 SPI 总线。

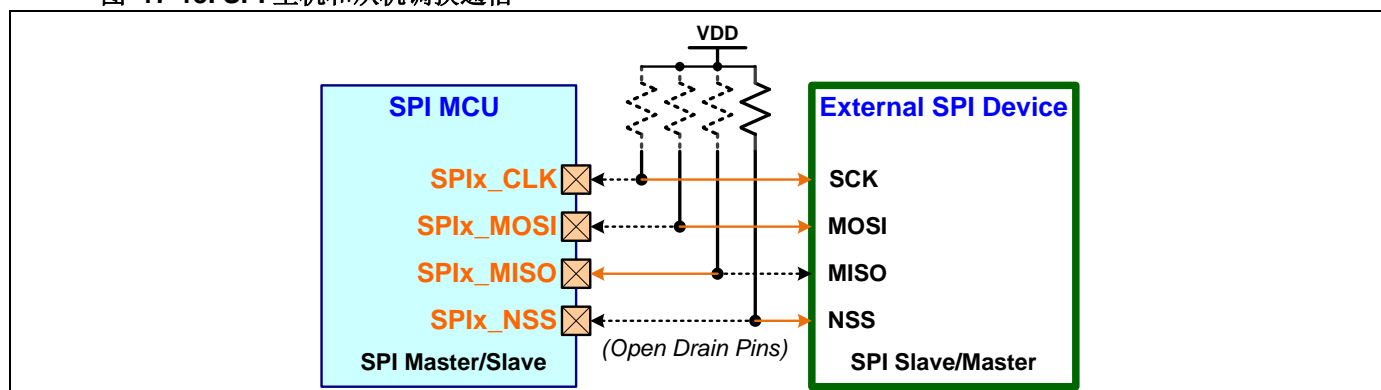
图 17-12. SPI 双主机通信



- 主机和从机调换通信

该连接为两个可作为主机或从机的 SPI 设备。当 1 个为主机时，另一个一定从机，在 **NSS** 信号在未使能状态下，他们可通过使能和拉动 **NSS** 信号进行互相通知，来调换主从机的角色。参照“[SPI 主机模式改变检测](#)”节以获取更多信息。

图 17-13. SPI 主机和从机调换通信



17.10. SPI 基本控制

17.10.1. SPI 时钟

SPI 模块可通过 **SPIx_CPOL** 和 **SPIx_CPHA** 寄存器设置时钟模式作为 **SPIx_CLK** 信号。下面的表格展示了时钟空闲状态和数据采样时钟沿的 SPI 时钟设定。

表 17-3. SPI 时钟模式表

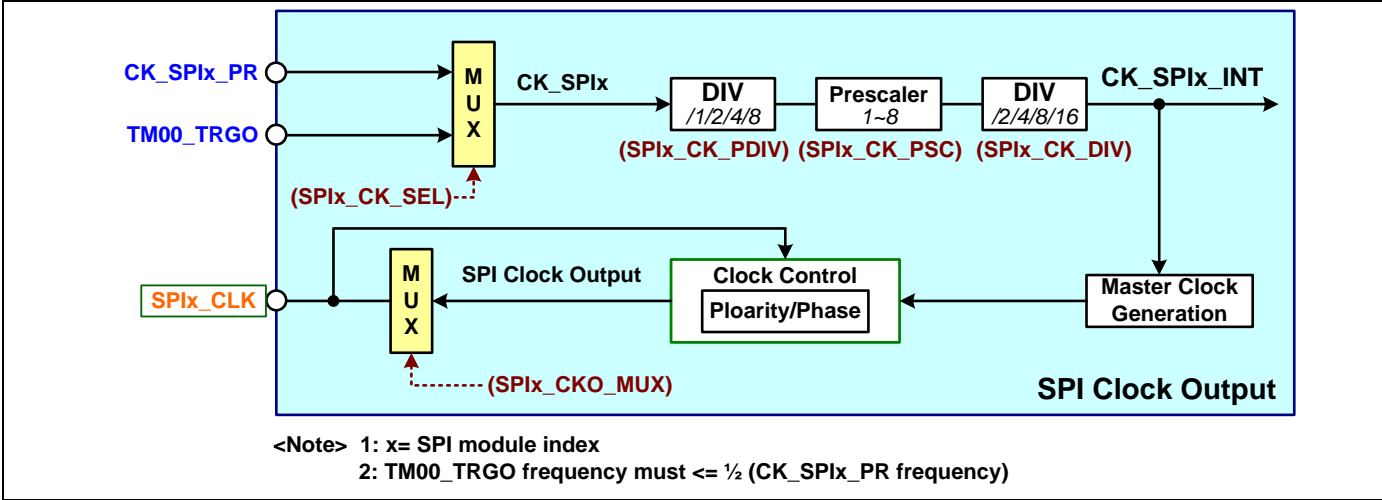
时钟模式	SPI 寄存器		功能描述
	SPIx_CPOL	SPIx_CPHA	
0	0	0	SPI 时钟低电平空闲状态，数据在前边沿采样
1	0	1	SPI 时钟低电平空闲状态，数据在后边沿采样
2	1	0	SPI 时钟高电平空闲状态，数据在前边沿采样
3	1	1	SPI 时钟高电平空闲状态，数据在后边沿采样

● SPI 时钟设定

SPI 时钟通过 **SPIx_CK_PDIV**, **SPIx_CK_PSC** 和 **SPIx_CK_DIV** 寄存器对 CK_APB, CK_AHB 或 TM00_TRGO 模块输入时钟进行分频。根据设计，主机模式中，SPI 时钟可达模块输入时钟频率的 1/2，从机模式中可达模块输入时钟频率的 1/4。用户可通过 **SPIx_HS_EN** 寄存器为 SPI 从机模式使能高速功能并提高 SPI 频率到模块输入时钟的 1/3。对于 MG32F02A132/072，最大 SPI 主机时钟频率为 12MHz，从机为 6MHz。对于 MG32F02A032，最大 SPI 主机时钟频率为 24MHz，从机为 16MHz。

[注释]: **SPIx_HS_EN** 寄存器不支持于 MG32F02A132/072。

图 17-14. SPI 主机时钟输出



● SPI 时钟切换

通常，SPI 时钟都是由 SPI 主机模式硬件控制产生和切换 SPI 时钟。对于应用请求，用户可通过 **SPIx_CKO_TOG** 寄存器直接切换 SPI 时钟输出信号 **SPIx_CLK**。当被使能时，**SPIx_CLK** 信号会被从低电平切换到高电平或高电平切换到低电平。该寄存器是被软件设置，并被硬件清除的。

[注释]: **SPIx_CKO_TOG** 寄存器不支持于 MG32F02A132/072。

17.10.2. SPI 发送

当检测到 TXF(**SPIx_TXF**)标志时, 用户可写发送数据到 TX 数据寄存器(**SPIx_TDAT**), 芯片会自动清除 TXF 标志。然后, 当再次检测到 TXF 标志时, 用户可写下一次的数据同样到 TX 数据寄存器(**SPIx_TDAT**)中, 重复该操作直到数据发送完成。

当 SPI 运行于半双工双向通信模式时, TCF 标志(**SPIx_TCF**)代表上一次数据传输已完成。用户可通过 **SPIx_RX_EN** 寄存器改变数据传输方向为接收, 也可通过 **SPIx_TX_DIS** 寄存器设置改变数据线为 Hi-Z 或 GPIO 数据锁存状态。

当 SPI 主机标准全双工通信模式时, 用户可通过 **SPIx_DOUT_MDS** 寄存器选择空闲状态的数据输出的三态或驱动模式。主机模式时, 在空闲状态下禁用且数据发送时, **SPIx_MOSI** 会三态输出。主机模式时, 在空闲状态下使能且数据发送时, **SPIx_MOSI** 会驱动输出。

[注释]: **SPIx_DOUT_MDS** 寄存器不支持于 MG32F02A132/072。

根据设计, SPI 数据必须在 SPI 从机数据发送模式的帧数据第一个采样时钟边缘的前一个时钟边缘之前更新到 TX 移动缓冲区。用于使能 SPI 从机模式发送数据的数据更新控制设置直接更新于 **SPIx_TXUPD_EN** 寄存器。用户可使能该寄存器且 SPI 数据可在帧数据的第一个采样时钟边沿之前被延迟更新到 TX 移动缓冲区。

[注释]: **SPIx_TXUPD_EN** 寄存器不支持于 MG32F02A132/072。

17.10.3. SPI 接收

与数据发送相同, 当检测到 RXF (**SPIx_RXF**)标志时, 用户可从 RX 数据寄存器(**SPIx_RDAT**)读取接收到的数据, 芯片会自动清除 RXF 标志。同样的, 当再次检测到 RXF 标志时, 用户可读取接收到的数据, 重复该操作直到数据发送完成。

SPIx_RX_TH 寄存器用于设置阴影缓冲的数据字节阈值。当阴影缓冲的数据字节数等于阈值时, 硬件会把阴影缓冲的内容复制到数据寄存器。

[注释]: 对于 MG32F02A132/072, 当 SPI 是主机模式时, **SPIx_RX_TH** 寄存器不可被改变值。

对于末尾数据控制, 用户可检查 RXDF 标志(**SPIx_RXDF**)和 **SPIx_RNUM** 寄存器。RXDF 标志代表接收数据字节数于前一个接收数据字节数不同。**SPIx_RNUM** 寄存器代表当数据阴影缓冲最后发送给 **SPIx_RDAT** 寄存器的接收数据字节数。固件可写初始值作为接收字节数比较。

当接收阴影缓冲在 SPI 从机模式最后一次数据接收后仍有数据, 用户可通过 **SPIx_RSB_TRG** 寄存器设置触发更新阴影缓冲的时效数据到 **SPIx_RDAT** 寄存器中。

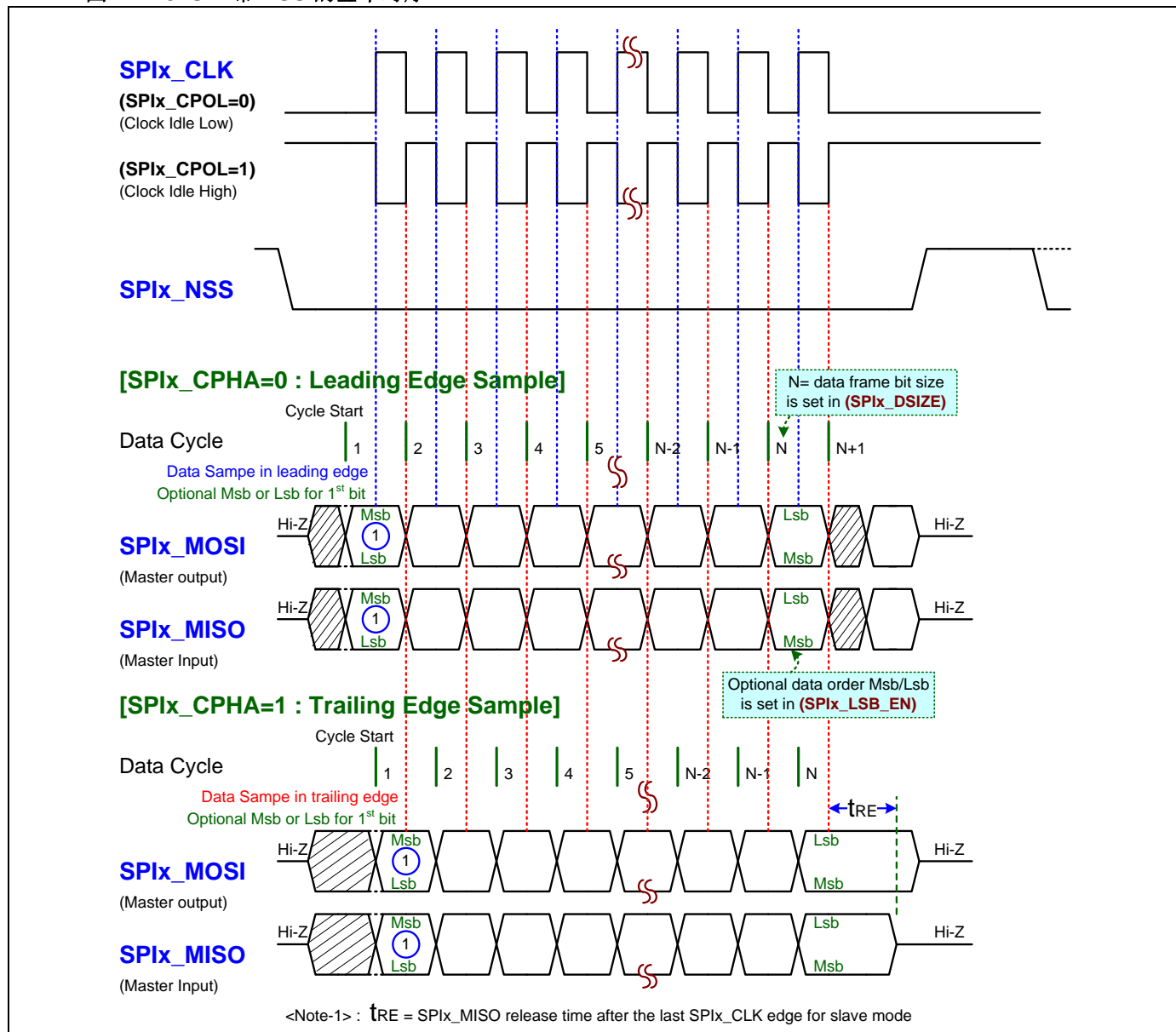
[注释]: **SPIx_RSB_TRG** 寄存器不支持于 MG32F02A132/072。

17.10.4. SPI 基本时序

● SPI 带 NSS 的控制

下面的图表展示了 SPI 带 NSS 信号的时钟模式 0~3 基本时序。数据信号在 **NSS** 未启动周期内 **MOSI** 和 **MISO** 是 Hi-Z 状态代表输出状态。

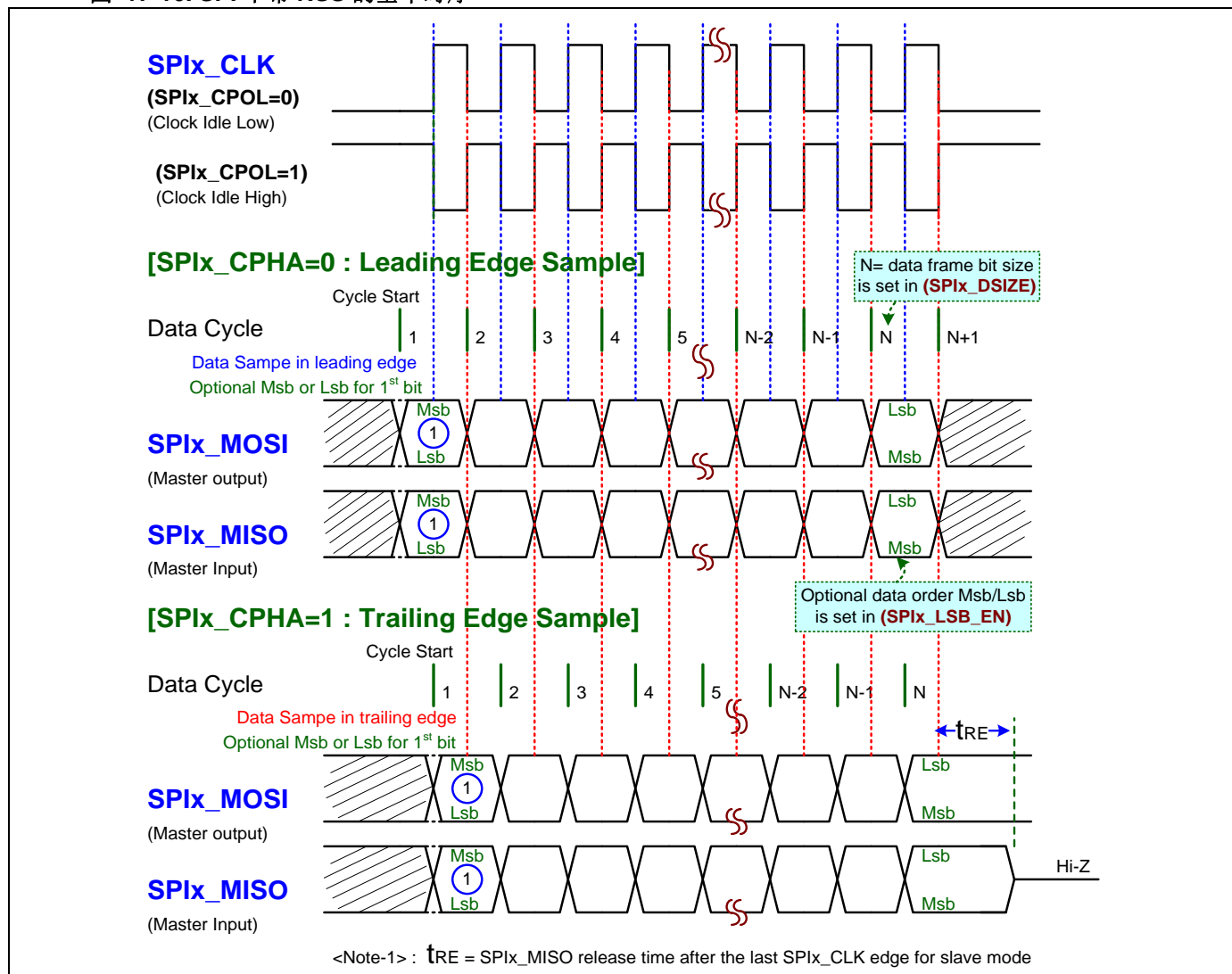
图 17-15. SPI 带 NSS 的基本时序



- SPI 不带 NSS 的控制

下面的图表展示了 SPI 不带 NSS 信号的时钟模式 0~3 基本时序。当不使用 NSS 信号, 数据信号 **MOSI** 和 **MISO** 在 NSS 未启动周期内为未知状态或 Hi-Z 状态。

图 17-16. SPI 不带 NSS 的基本时序

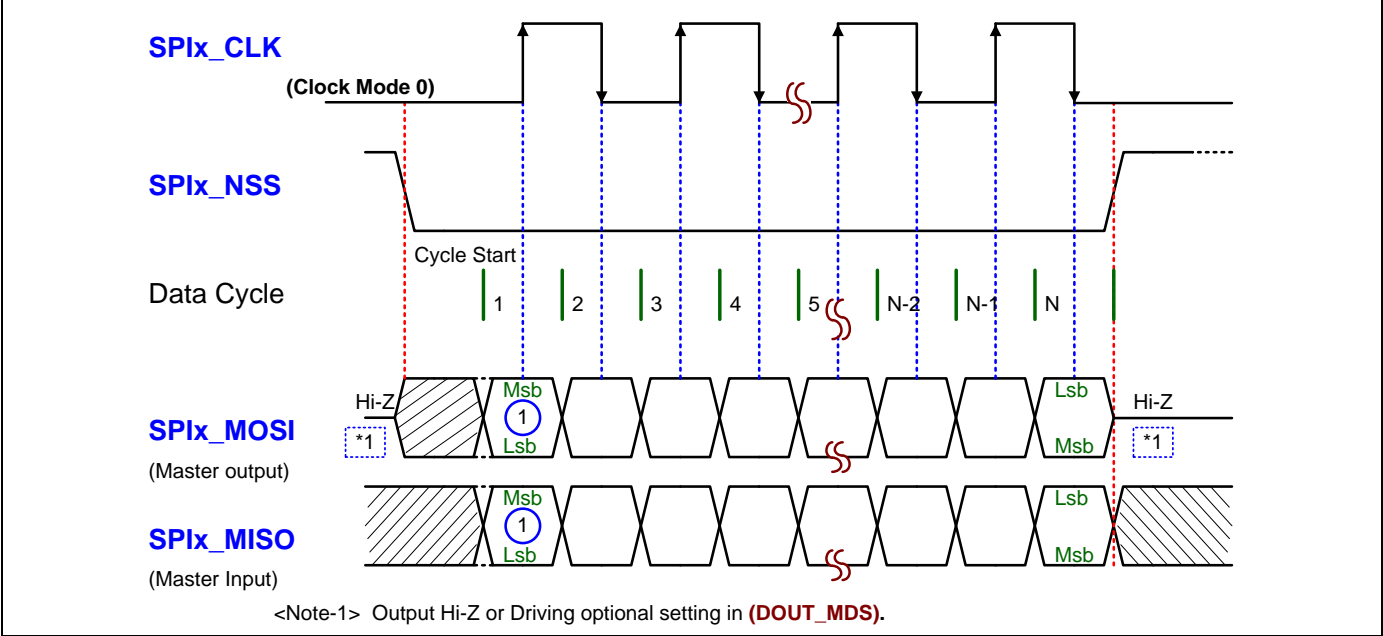


17.10.5. SPI DTR 模式

SPI 模块仅在 SPI 主机时钟模式-0 中通过使能 **SPIx_DTR_EN** 寄存器支持双传输速率（DTR）模式。当使能时，SPI 数据会在 SPI 时钟的上升和下降沿都发送数据。

[注释]: **SPIx_DTR_EN** 不支持于 MG32F02A132/072。

图 17-17. SPI 主机 DTR 模式时序



17.10.6. SPI NSS 模式

用户可通过设置 **SPIx_MDS**, **SPIx_NSSO_EN**, **SPIx_NSSI_EN** 和 **SPIx_NSSI_SEL** 寄存器设置 NSS 模式的 **SPIx_NSS** 或 **SPIx_NSSI** 信号。

SPI 模块可通过 **SPIx_NSSO_EN** 或 **SPIx_NSSI_EN** 寄存器支持是否使用 **NSS** 信号进行 SPI 通讯。此外，用户还可使用软件寄存器控制 **NSS** 信号。**SPIx_NSS_SWEN** 寄存器是用于通过软件控制使能 **SPIx_NSS** 和 **SPIx_NSSI** 信号的。

● SPI NSS 控制设置

下面的表格展示了 SPI NSS 模式控制的寄存器设置。

表 17-4. SPI NSS 控制表

NSS 模式	SPI 寄存器				功能描述
	MDS	NSSO_EN	NSSI_EN	NSSI_SEL	
主机	1	0	0	x	主机不带 NSS 输出
主机+ NSS	1	1	0	x	主机带 NSS 输出
从机	0	x	0	x	从机不带 NSS 输入
从机+ NSS	0	x	1	0	从机带来自 SPIx_NSS 引脚的 NSS 输入
				1	从机带来自 SPIx_NSSI 引脚的 NSS 输入
主机+ MODF	1-> 0	0	1	0	主机不带 NSS 输出，带 MODF 功能(使用 SPIx_NSS 引脚)，改变从机为 NSS 输入
				1	主机不带 NSS 输出，带 MODF 功能(使用 SPIx_NSSI 引脚)，改变从机为 NSS 输入
主机+ NSS + MODF	1-> 0	1	1	1	主机带 NSS 输出，带 MODF 功能(使用 SPIx_NSSI 引脚)，改变从机为 NSS 输入 (主机 NSS 输出到 SPIx_NSS 引脚，NSS(MODF) 从 SPIx_NSSI 引脚输入)
				0	不支持

x：不影响

1->0：芯片检测 NSS 或 NSSI 信号拉低，然后改变为从机模式或禁用 SPI

● SPI 主机硬件 NSS 控制

当通过 **SPIx_NSS_SWEN** 寄存器设置使用硬件控制 NSS 功能，用户可设置 **SPI0_NSS_PEN** 寄存器使能在两个帧数据传输之间产生 **NSS** 电平。

[注释]: 时钟模式不支持模式 1, 3 当 **SPIx_NSS_PEN** 被禁用时于 MG32F02A132/072。

下面的表格展示了 SPI NSS 时序表。

表 17-5. SPI 主机 NSS 时序表

	NSS Timing								
芯片	MG32F02A132/072 (*1)				MG32F02A032/A128/U128/A064/U064				单位
时钟模式	t _L	t _T	t _I	t _{pw}	t _L	t _T	t _I	t _{pw}	
0	1.5	0/1 (*2)	2.5/4.5 (*3)	2	1	0	2 or 3 (*4)	1 or 2 (*5)	t _{CLK}
1	1.5	1	4.5	2	0.5	0	2 or 3 (*4)	1 or 2 (*5)	t _{CLK}
2	1.5	0/1 (*2)	2.5/4.5 (*3)	2	1	0	2 or 3 (*4)	1 or 2 (*5)	t _{CLK}
3	1.5	1	4.5	2	0.5	0	2 or 3 (*4)	1 or 2 (*5)	t _{CLK}
<div>t_{CLK} : SPI 位时间</div> <div>t_L : SPIx_NSS 在第一个 SPIx_CLK 边沿的前面的时间（位时间）</div> <div>t_T : SPIx_NSS 在最后一个 SPIx_CLK 边沿的后面的时间（位时间）</div> <div>t_I : SPIx_NSS 传输时的空闲时间（位时间）</div> <div>t_{pw} : SPIx 空闲期间硬件 NSS 的脉宽（位时间）</div>									

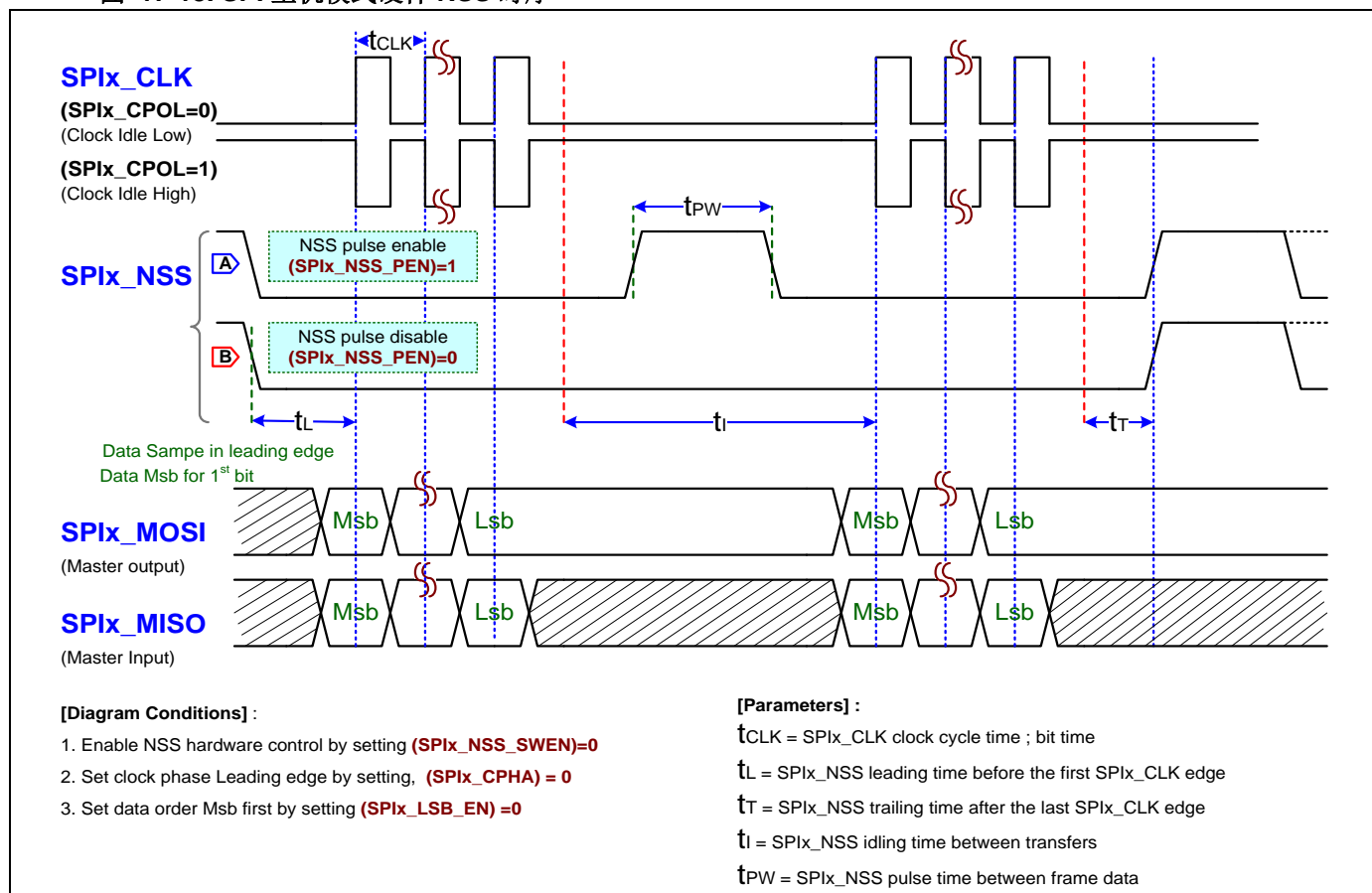
Note *1 : MG32F02A132/072 时钟模式不支持模式 1, 3 当 SPIx_NSS_PEN =0
*2 : tT = 0 若 SPIx_NSS_PEN=0; = 1 若 SPIx_NSS_PEN=1
*3 : tI = 2.5 若 SPIx_NSS_PEN=0; = 4.5 若 SPIx_NSS_PEN=1
*4 : tI = 0 若 SPIx_NSS_PEN=0; = 2 或 3 根据设置 SPIx_NSS_IDT 设置若 SPIx_NSS_PEN=1
*5 : tpw = 0 若 SPIx_NSS_PEN=0; = 1 或 2 根据设置 SPIx_NSS_IDT 设置若 SPIx_NSS_PEN=1

当 **SPI0_NSS_PEN** 被使能，用户可通过 **SPIx_NSS_IDT** 寄存器设置 **NSS** 电平宽度。

[注释]: **SPIx NSS IDT** 寄存器不支持于 MG32F02A132/072。

下面的图表展示了 SPI 主机模式的硬件 NSS 时序图。

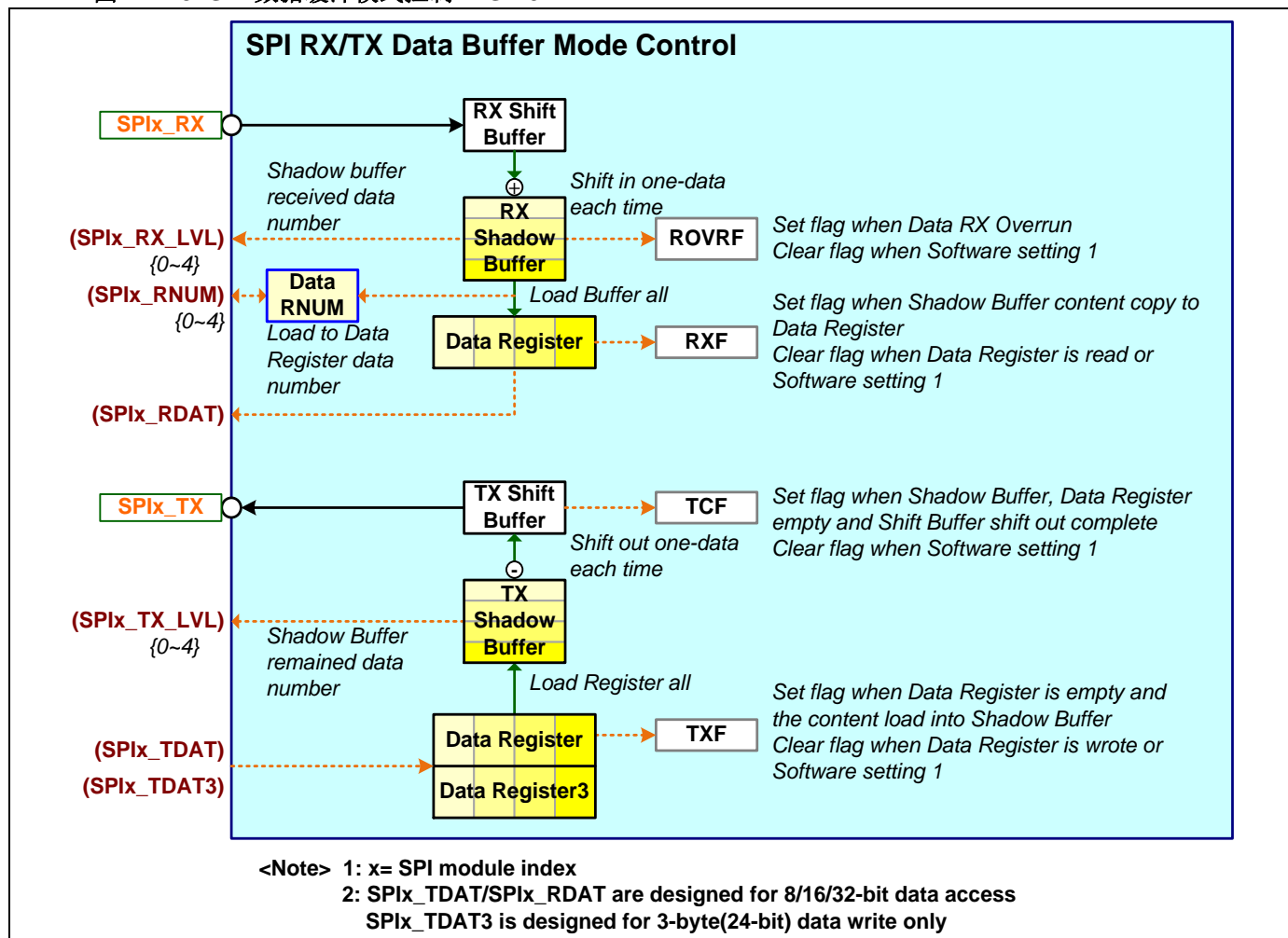
图 17-18. SPI 主机模式硬件 NSS 时序



17.11. SPI 数据缓冲

该模块包含 2 个 32 位移动缓冲、2 个 32 位阴影缓冲和 2 个 32 位数据寄存器用于数据流控制并减少 CPU 开销。用户可直接使用 TXF (**SPIx_TXF**)和 RXF (**SPIx_RXF**)事件标志做数据传输流控制。下面的图表展示了 SPI 数据缓冲模式控制块

图 17-19. SPI 数据缓冲模式控制 – SPI0



17.11.1. SPI 数据缓冲控制

RXF 标志(**SPIx_RXF**)会在每次 RX 阴影缓冲内容被复制到 RX 数据寄存器中时置起，此外，若 **SPIx_RX_IE** 寄存器已使能，相关的中断将会被置起。

TXF 标志(**SPIx_TXF**)会在每次 TX 数据寄存器内容被复制到 TX 阴影缓冲中时置起，此外，若 **SPIx_TX_IE** 寄存器已使能，相关的中断将会被置起。当阴影缓冲和 TX 数据寄存器为空时，若移动缓冲完成搬出，TCF 标志(**SPIx_TCF**)会被置起。

数据寄存器 **SPIx_TDAT** 和 **SPIx_RDAT** 被设计为 8/16/32 位数据访问且寄存器 **SPIx_TDAT3** 被设计为只可进行 3 字节 (24 位) 写操作。当用户写任意一个 8/16/32 位数据到 **SPIx_TDAT3** 寄存器中，芯片会当做 3 字节 (24 位) 数据写入。

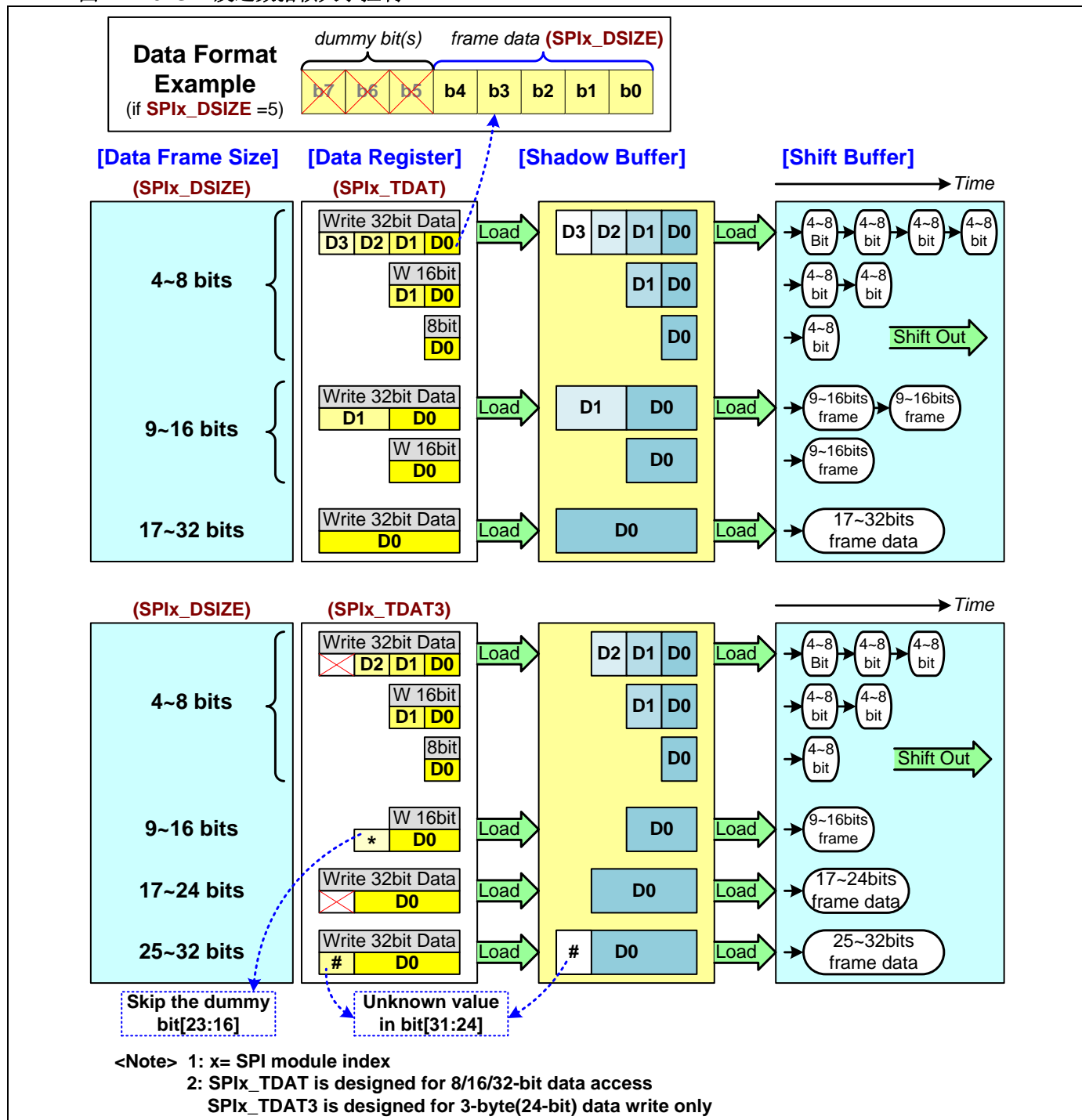
17.11.2. SPI 数据帧

用户可通过 **SPIx_DSIZE** 寄存器设置数据帧位大小为 4 位到 32 位。此外，用户可设置 **SPIx_LSB_EN** 寄存器以设置帧数据顺序是 MSB 或 LSB。

17.11.3. SPI 发送数据帧大小

下面的图表展示了不同帧大小和写数据位下，用于 SPI 发送数据控制块的 **SPIx_TDAT** 和 **SPIx_TDAT3** 寄存器。

图 17-20. SPI 发送数据帧大小控制



- 写 8 位数据

当写 8 位数据到 **SPIx_TDAT** 或 **SPIx_TDAT3** 数据寄存器:

- 帧大小 4~8 位

该芯片会复制数据到阴影缓冲并发送 1 次帧数据大小的数据到移动缓冲做数据传输。

— 帧大小 9~32 位

该芯片会与垃圾值数据位[31:8]一起复制到阴影缓冲并发送 1 次帧数据大小的数据到移动缓冲做数据传输。

● 写 16 位数据

当写 16 位数据到 **SPIx_TDAT** 或 **SPIx_TDAT3** 数据寄存器:

— 帧大小 4~8 位

该芯片会复制数据到阴影缓冲并拆分数据发送 2 次帧数据大小的数据到移动缓冲做数据传输。

— 帧大小 9~16 位

该芯片会复制数据到阴影缓冲并发送 1 次帧数据大小的数据到移动缓冲做数据传输。

— 帧大小 17~32 位

该芯片会与垃圾值数据位[31:16]一起复制到阴影缓冲并发送 1 次帧数据大小的数据到移动缓冲做数据传输。

● 写 32 位数据

(1) 当写 32 位数据到 **SPIx_TDAT** 数据寄存器:

— 帧大小 4~8 位

该芯片会复制数据到阴影缓冲并拆分数据发送 4 次帧数据大小的数据到移动缓冲做数据传输。

— 帧大小 9~16 位

该芯片会复制数据到阴影缓冲并拆分数据发送 2 次帧数据大小的数据到移动缓冲做数据传输。

— 帧大小 17~32 位

该芯片会复制数据到阴影缓冲并发送 1 次帧数据大小的数据到移动缓冲做数据传输。

(2) 当写 32 位数据到 **SPIx_TDAT3** 数据寄存器:

— 帧大小 4~8 位

该芯片会复制数据到阴影缓冲并拆分数据发送 3 次帧数据大小的数据到移动缓冲做数据传输。

— 帧大小 9~24 位

该芯片会复制数据到阴影缓冲并拆分数据发送 1 次帧数据大小的数据到移动缓冲做数据传输。

— 帧大小 25~32 位

该芯片会与垃圾值数据位[31:24]一起复制到阴影缓冲并发送 1 次帧数据大小的数据到移动缓冲做数据传输。

17.11.4. SPI 数据缓冲清除

用于固件数据控制,用户可通过 **SPIx_RDAT_CLR** 或 **SPIx_TDAT_CLR** 寄存器强制清除接收或发送的数据缓冲。这两个寄存器位都是通过软件设置,被硬件清除的。

当使能了 **SPIx_RDAT_CLR** 寄存器位,接收数据寄存器和阴影缓冲会被清除,同时标志 **SPIx_RXF** 和 **SPIx_RX_LVL** 会被清除;当使能了 **SPIx_TDAT_CLR** 寄存器位,发送数据寄存器和阴影缓冲会被清除,同时标志 **SPIx_TXF** 和 **SPIx_TX_LVL** 会被清除。

17.12. SPI 数据模式

SPI 模块提供多种灵活的数据模式：标准 SPI, 1-线 SPI, 2-线 SPI, 4-线 SPI, 8-线 SPI。参照“[SPI 应用连接](#)”节以获取更多信息。2-线 SPI 仅支持主机模式。

用户可通过 **SPIx_DAT_LINE**, **SPIx_BDIR_OE**, **SPIx_MDS** 和 **SPIx_COPY_EN** 寄存器设置数据模式。对于带 NSS 的 SPI 从机模式, 用户可通过 **SPIx_ADPX_EN** 寄存器使能在数据传输之前硬件自动改为标准全双工模式。当 **SPIx_NSSI_EN** 被禁用时, 该功能是无效的。当该功能被使能且 NSS 输入被从未启动改为启动时, **SPIx_DAT_LINE** 会被自动强制改为 0 并改为全双工标准 SPI 模式。

[注释]: **I2Cx_STPSTR_IE** 寄存器不支持于 MG32F02A132/072。

下面的表格展示了 SPI 数据模式控制的寄存器设置和 IO 引脚使用。

[注释]: SPI-8 模式不支持于 MG32F02A032。

表 17-6. SPI 数据控制表

数据模式	DPX/HPX	SPIx 寄存器				I/O 引脚				功能描述
		DAT_LINE	BDIR_OE	MDS	COPY_EN	MOSI/D0	MISO/D1	D2,D3	D4~D7	
SPI	DPX	0	x	0	x	Input	Output			2 数据线全双工通信
		0	x	1	x	Output	Input			
SPI-1 (1-线)	HPX In	1	0	x	x	Input				1 数据线半双工输入
	HPX Out	1	1	x	x	Output				1 数据线半双工输出
SPI-2 (2-线)	HPX In	2	0	x	0	Input	Input			2 分离数据线输入
	HPX Out	2	1	x	0	Output	Output			2 分离数据线输出
SPI-2C (2-线)	HPX In	2	0	x	1	Input	Input			2 分离数据线输入
	HPX Out	2	1	x	1	Output	Output			2 复制数据线输出
SPI-4 (4-线)	HPX In	3	0	x	0	Input	Input	Input		4 分离数据线输入
	HPX Out	3	1	x	0	Output	Output	Output		4 分离数据线输出
SPI-4C (4-线)	HPX In	3	0	x	1	Input	Input	Input		4 分离数据线输入
	HPX Out	3	1	x	1	Output	Output	Output		4 复制数据线输出
SPI-8 (8-线)	HPX In	5	0	x	x	Input	Input	Input	Input	8 分离数据线输入
	HPX Out	5	1	x	x	Output	Output	Output	Output	8 分离数据线输出

注释 : (1) x = 0 或 1

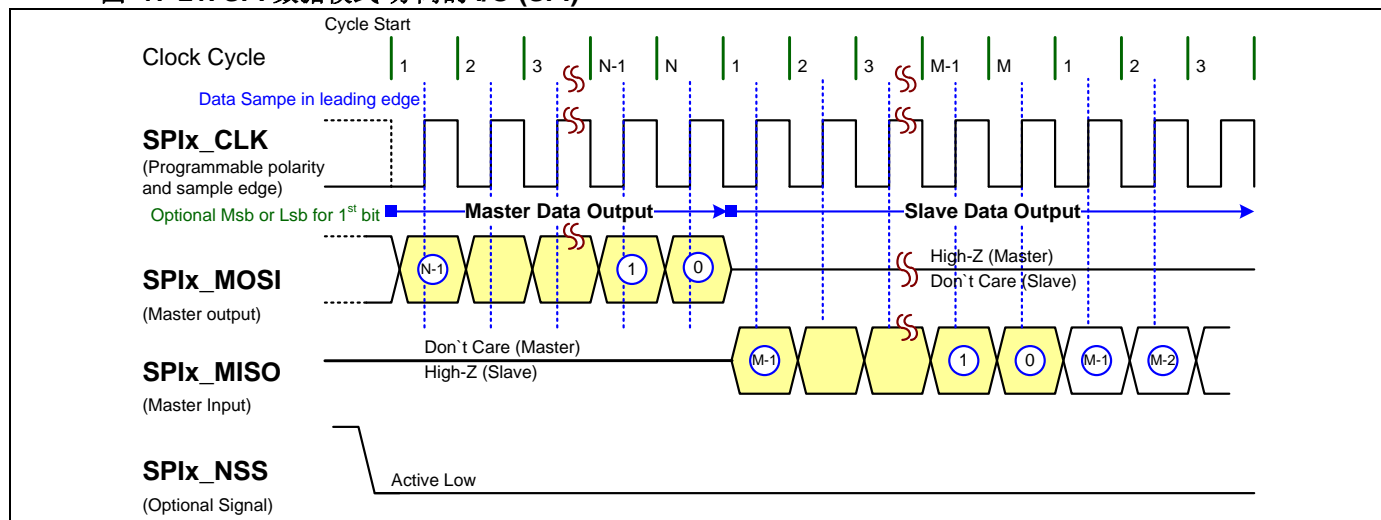
(2) DPX = 全双工通信

(3) HPX = 半双工通信

17.12.1. SPI 数据模式 – SPI

该数据模式可使用全双工通信传输数据。有 2 个分离的数据信号 **SPIx_MOSI** (**SPIx_D0**) 和 **SPIx_MISO** (**SPIx_D1**) 用于 SPI 收发。

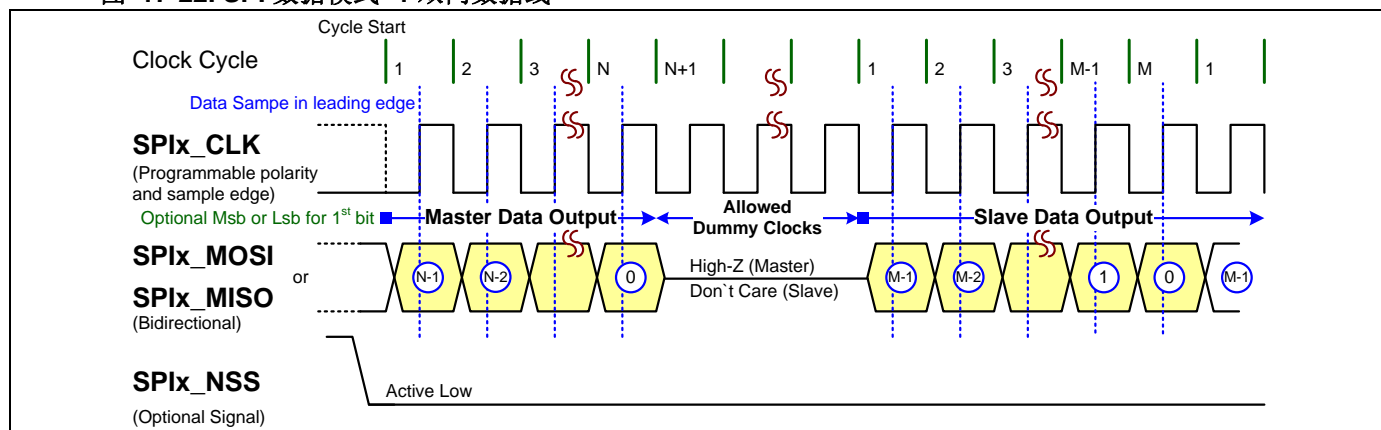
图 17-21. SPI 数据模式-分离的 I/O (SPI)



17.12.2. SPI 数据模式 – SPI-1

该数据模式只可使用半双工通信传输数据。只有 1 个双向的数据信号 **SPIx_MOSI** (**SPIx_D0**) 或 **SPIx_MISO** (**SPIx_D1**) 用于 SPI 收发。

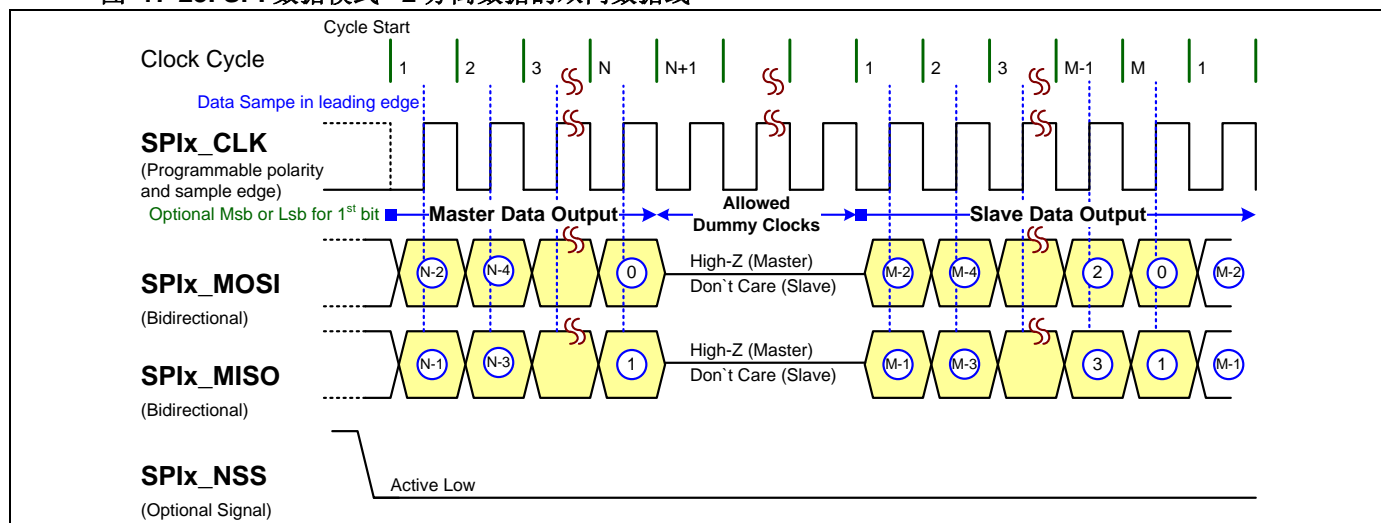
图 17-22. SPI 数据模式- 1 双向数据线



17.12.3. SPI 数据模式 – SPI-2

该数据模式只可使用半双工通信传输数据。有 2 个双向的数据信号 **SPIx_MOSI** (**SPIx_D0**) 和 **SPIx_MISO** (**SPIx_D1**) 用于 SPI 收发。

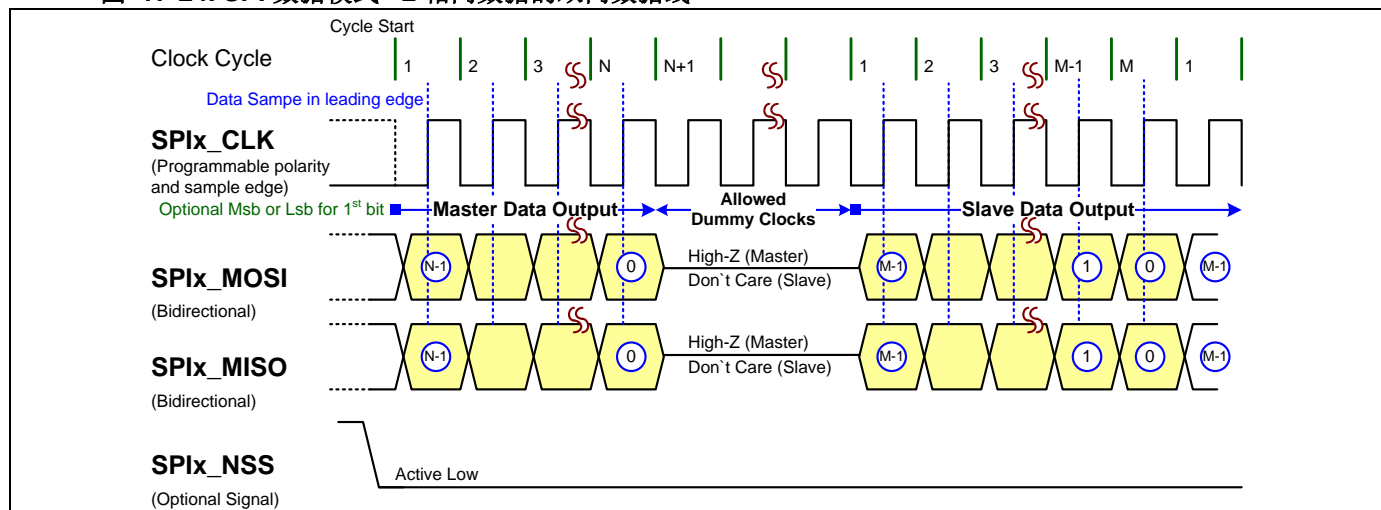
图 17-23. SPI 数据模式– 2 分离数据的双向数据线



17.12.4. SPI 数据模式 – SPI-2C

该数据模式只可使用半双工通信传输数据。有 2 个数据相同的双向的数据信号 **SPIx_MOSI** (**SPIx_D0**) 和 **SPIx_MISO** (**SPIx_D1**) 用于 SPI 收发。

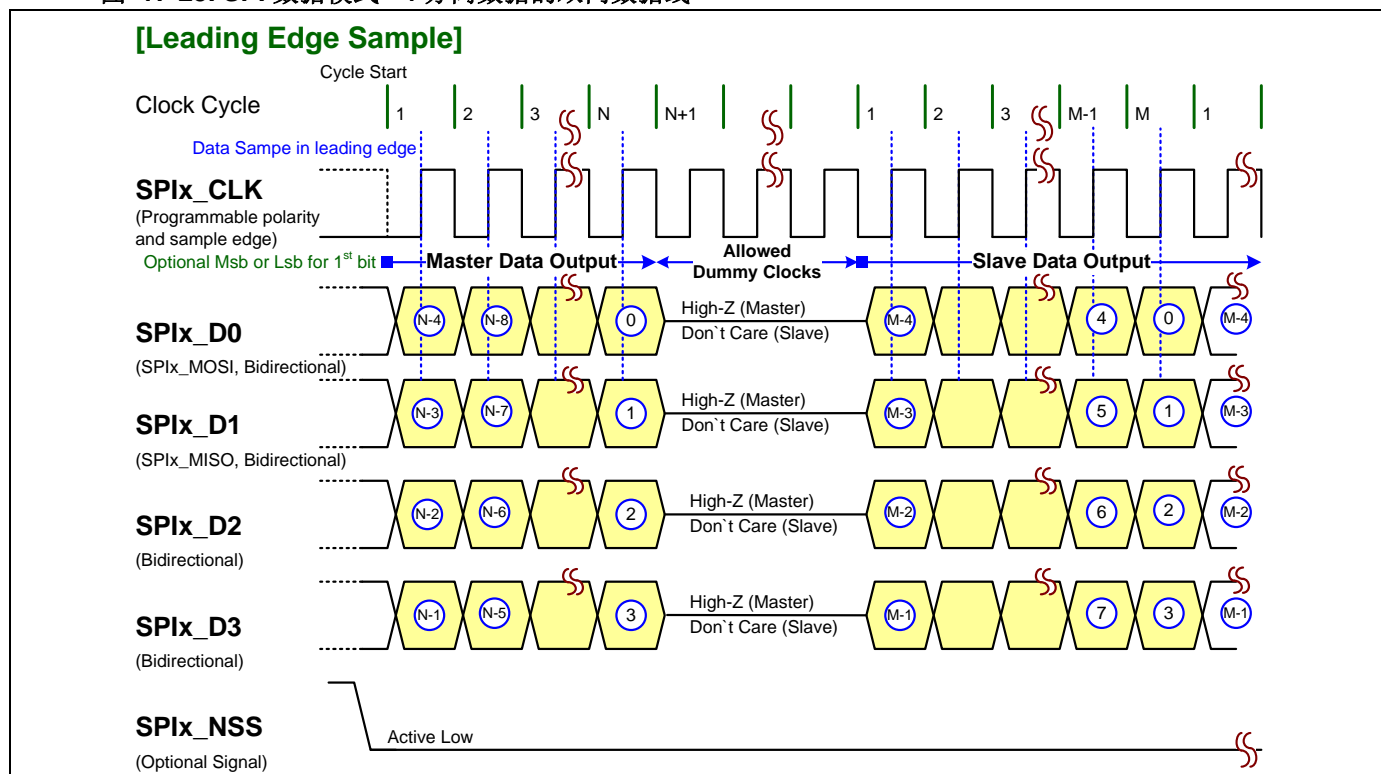
图 17-24. SPI 数据模式– 2 相同数据的双向数据线



17.12.5. SPI 数据模式 – SPI-4

该数据模式只可使用半双工通信传输数据。有 4 个双向的数据信号 **SPIx_D[3:0]** 用于 SPI 收发。

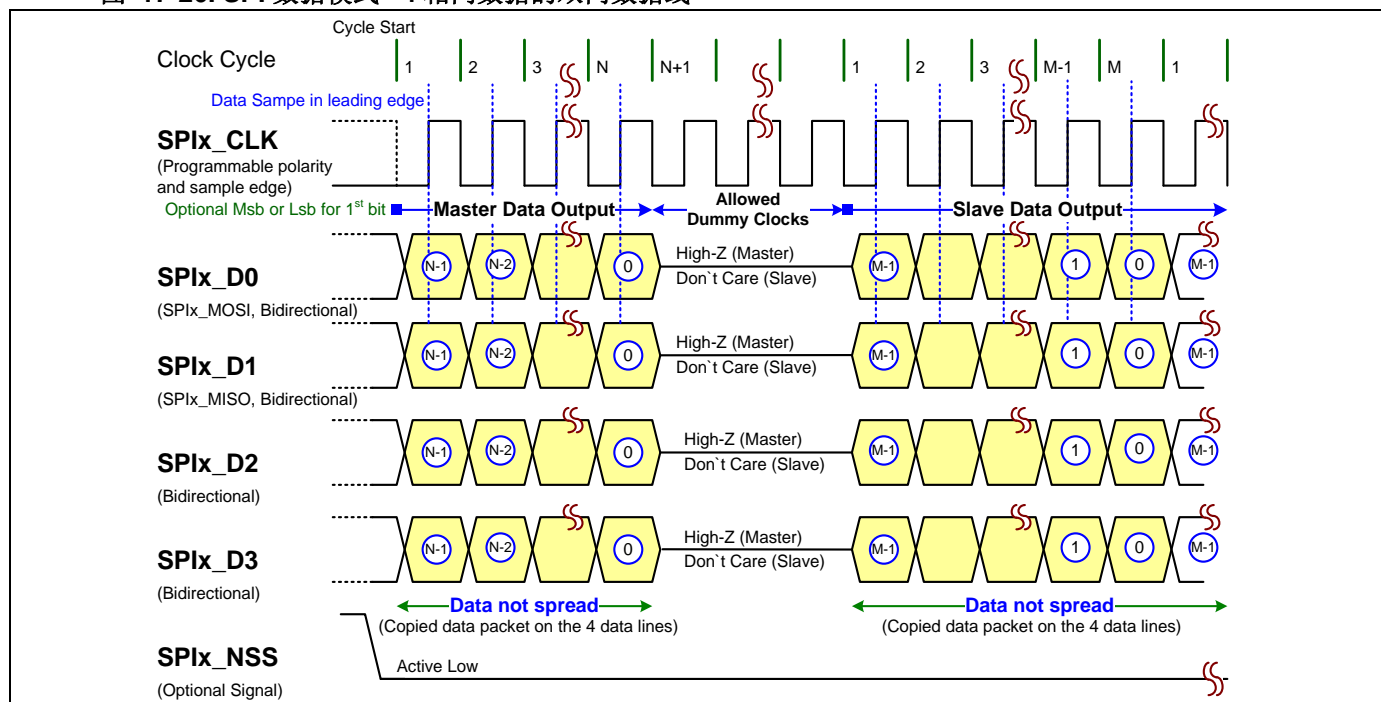
图 17-25. SPI 数据模式– 4 分离数据的双向数据线



17.12.6. SPI 数据模式 – SPI-4C

该数据模式只可使用半双工通信传输数据。有 4 个双向的相同数据的数据信号 **SPIx_D[3:0]** 用于 SPI 收发。

图 17-26. SPI 数据模式– 4 相同数据的双向数据线

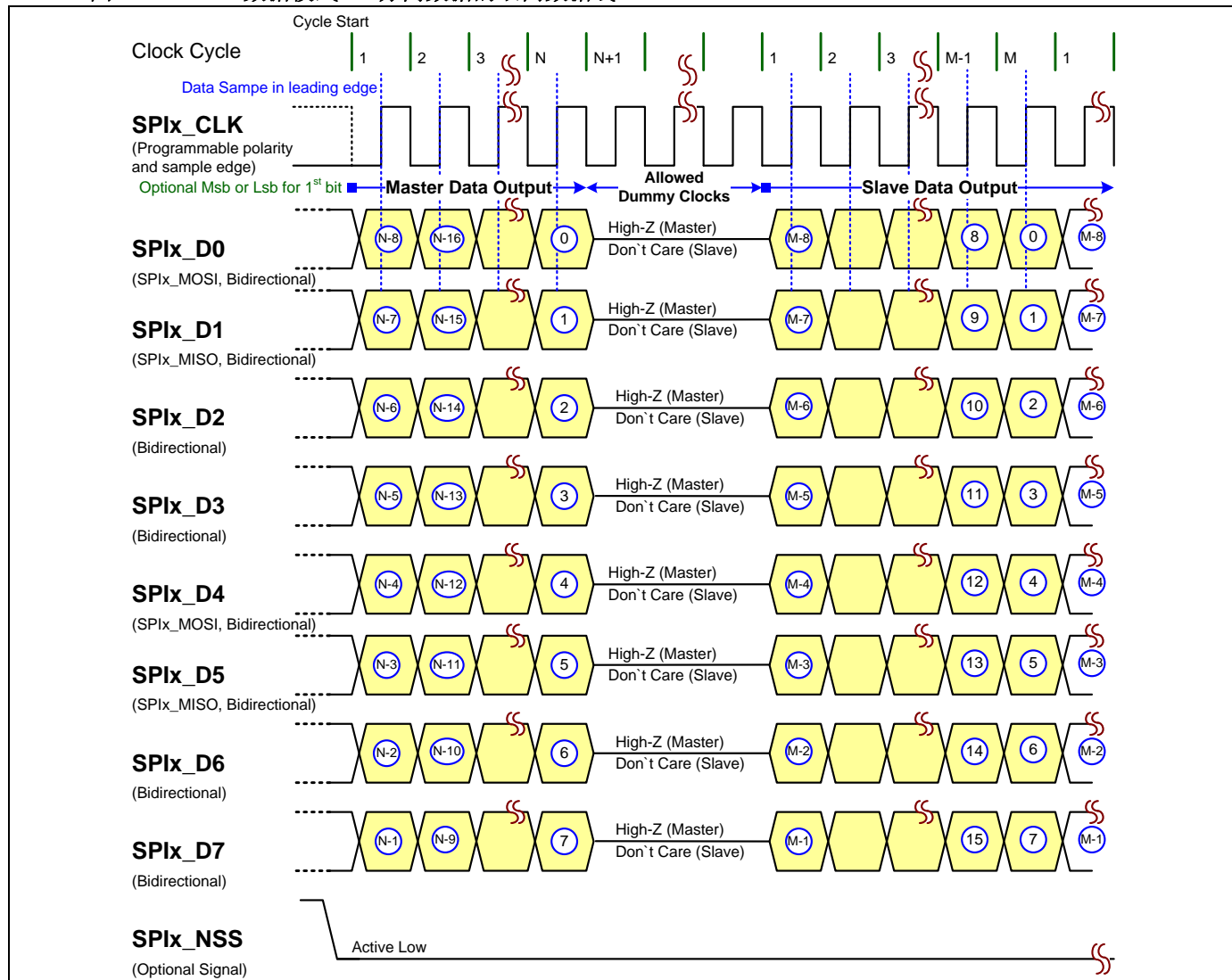


17.12.7. SPI 数据模式 – SPI-8

该数据模式只可使用半双工通信传输数据。有 8 个双向的数据信号 **SPIx_D[7:0]** 用于 SPI 收发。

[注释]: SPI-8 模式不支持于 MG32F02A032。

图 17-27. SPI 数据模式– 8 分离数据的双向数据线



17.13. SPI 串行 Flash 支持

用户可使用该 SPI 模块与外部 SPI 串行 Flash 进行通信。下面的表格展示了通用的 SPI flash 类型。

[注释]: SP8 模式不支持于 MG32F02A032。

表 17-7. SPI Flash 类型表

Flash 类型	指令地址	数据		功能描述
	访问线	访问线	DPX/HPX	
SPI (Standard)	1	1(In) / 1(Out)	DPX	2 数据线全双工通信
SPI_D (Dual Output)	1	2	HPX	从机 1 数据线输入和 2 数据线输出 主机 1 数据线输出和 2 数据线输入
SF (SPI-1)	1	1	HPX	1 数据线半双工通信
SF2 (SPI+DPI)	1	2	HPX	发送指令为 1 数据线, 发送数据为 2 数据线 (数据只能选择分离)
SP2 (DPI)	2	2	HPX	发送指令和数据都为 2 数据线
SF4 (SPI+QPI)	1	4	HPX	发送指令为 1 数据线, 发送数据为 4 数据线 (数据只能选择分离)
SP4 (SPI+QPI)	1 or 4	4	HPX	发送指令/地址为 1 或 4 线, 发送数据为 4 数据线 (数据只能选择分离)
SP8 (SPI+OPI)	1 or 8	8	HPX	发送指令/地址为 1 或 8 线, 发送数据为 8 数据线

注释 : DPX/HPX = 全/半双工通信

用户可设置 **SPIx_DAT_LINE**, **SPIx_BDIR_OE** 和 **SPIx_MDS** 寄存器用于不同的 flash 控制模式。参照“[SPI 数据模式](#)”节以获取更多信息。

下面的表格展示了 SPI flash 的寄存器设置和引脚 IO。

[注释]: OP 模式不支持于 MG32F02A032。

表 17-8. SPI Flash 控制表

Flash 模式	DPX/HPX	SPI 寄存器			I/O 引脚				功能描述
		DAT_LINE	BDIR_OE	MDS	MOSI/D0	MISO/D1	D2,D3	D4~D7	
SPI	DPX	0	x	0	Input	Output			2 数据线全双工通信
		0	x	1	Output	Input			
SPI-1	HPX	1	0	x	Input				1 数据线半双工通信
		1	1	x	Output				
DPI	HPX	2	0	x	Input	Input			发送指令和数据都为 2 数据线
		2	1	x	Output	Output			
QPI	HPX	3	0	x	Input	Input	Input		发送指令和数据都为 4 数据线 (数据只能选择分离)
		3	1	x	Output	Output	Output		
OPI	HPX	5	0	x	Input	Input	Input	Input	发送指令和数据都为 8 数据线
		5	1	x	Output	Output	Output	Output	

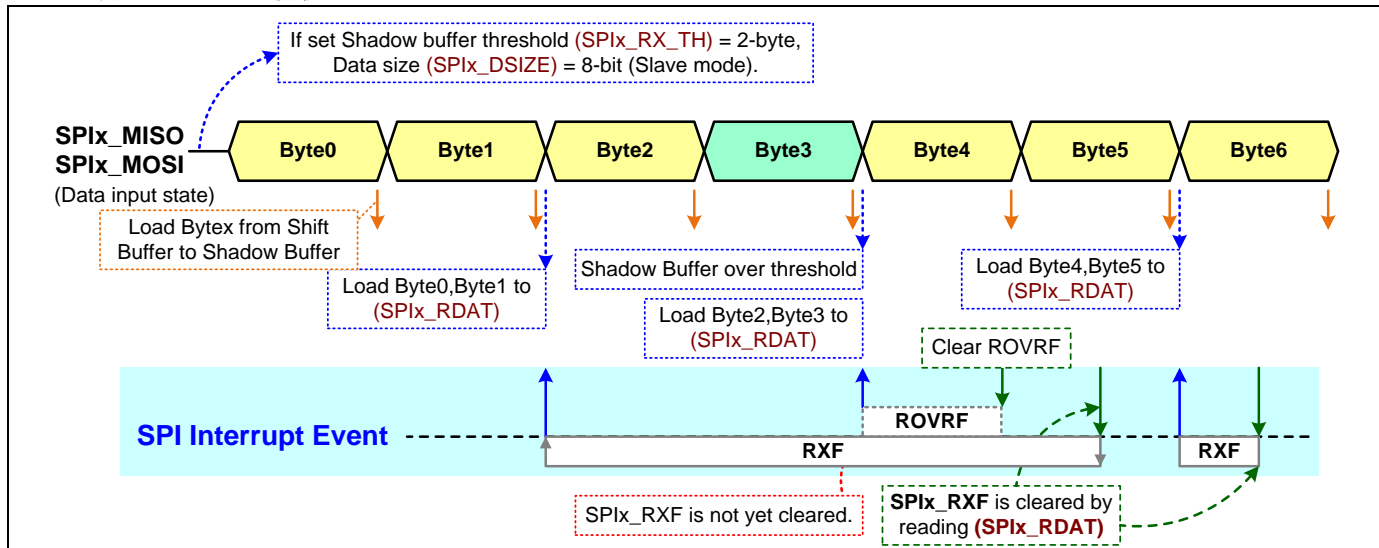
注释: (1) x = 0 或 1

(2) DPX/HPX = 全/半双工通信

17.14. SPI 接收溢出

从机模式下，当阴影缓冲满且 RX 数据寄存器(**SPIx_RDAT**)不为空时，ROVRF 标志(**SPIx_ROVRF**)会被置起。若移动缓冲仍在接收 1 帧数据且 RXF 标志还没被软件清除，下一个帧数据会被输入并覆盖移动缓冲的内容，如下图表所示字节-4 和字节-5 的状态。

图 17-28. SPI 接收溢出

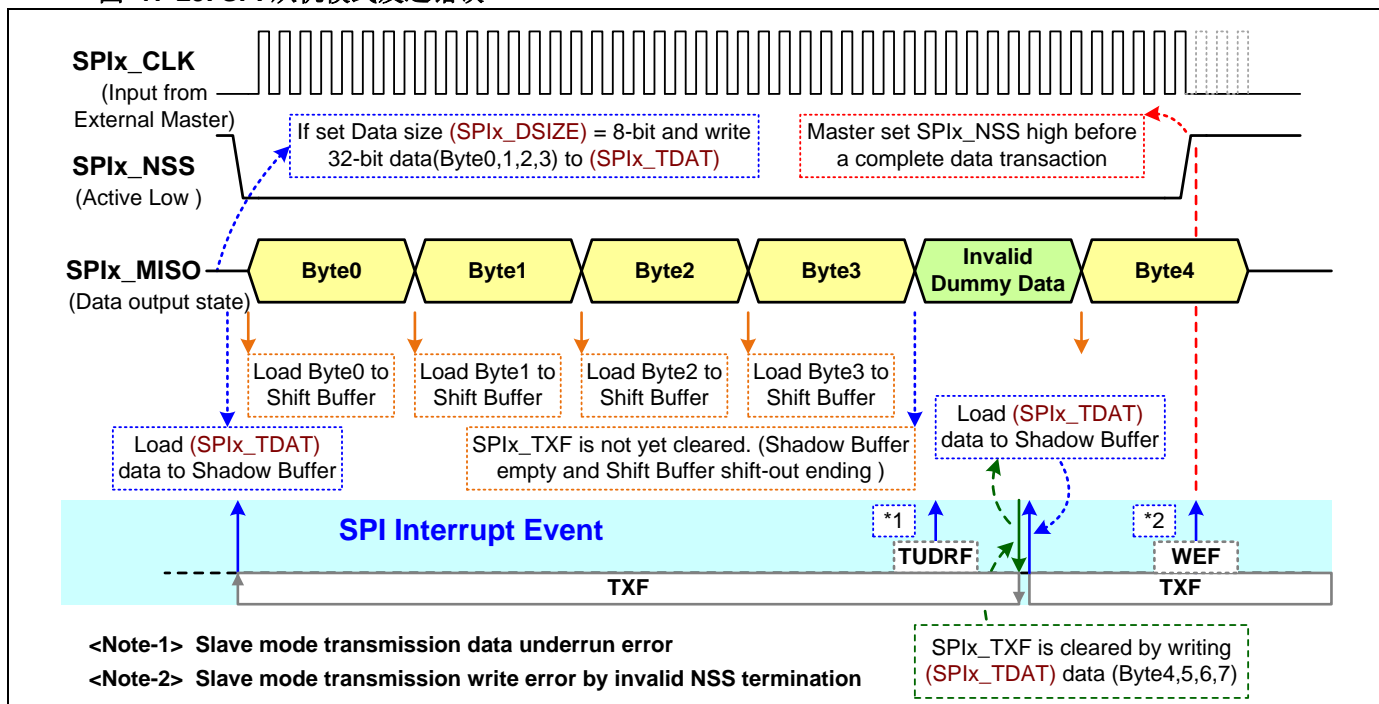


17.15. SPI 发送错误

从机模式下，当阴影缓冲和 TX 数据寄存器都为空时，若移动缓冲完成移出且外部 SPI 主机仍在请求数据时 TUDRF 标志(**SPIx_TUDRF**)会被置起。

当 NSS 信号(**SPIx_NSS**)在与外部主机通信过程中，数据帧未完成传输时 NSS 未启动，写错误 WEF 标志(**SPIx_WEF**)会被置起。

图 17-29. SPI 从机模式发送错误



17.16. SPI 主机模式改变检测

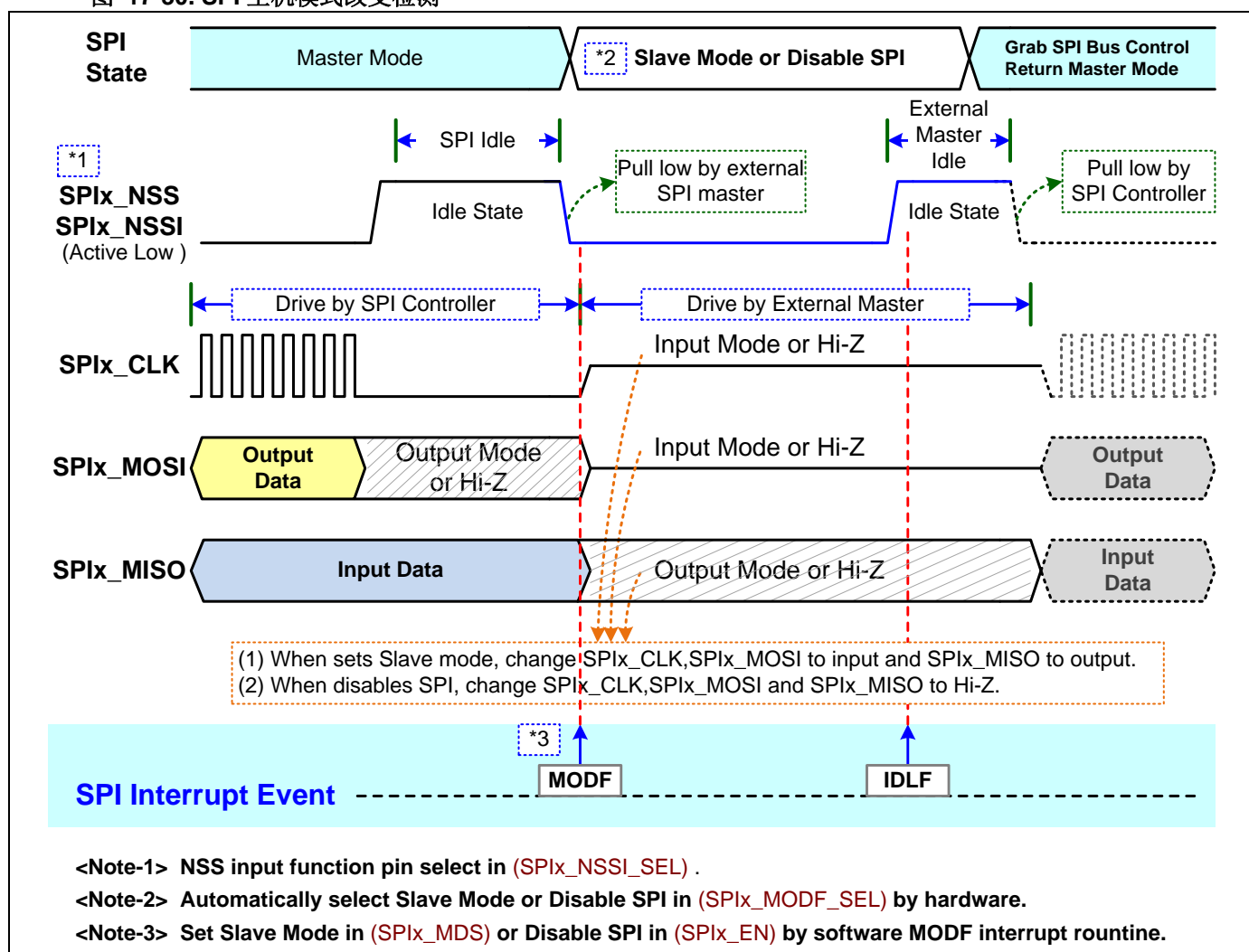
对于多主机通信, SPI 模块可检测 **SPIx_NSS** 或 **SPIx_NSSI** 信号的电平。当 SPI 模块运行于主机模式且在空闲状态下被外部 SPI 主机禁用, 模式错误标志 **MODF (SPIx_MODF)** 会被置起, SPI 模块会根据 **SPIx_MODF_SEL** 寄存器设置禁用或切换到从机模式。

然后 SPI 模块可检测 **SPIx_NSS** 或 **SPIx_NSSI** 信号是否进入闲置状态。当检测到进入闲置状态, **IDLF** 标志 (**SPIx_IDLF**) 会被置起, 因此用户可抓住 SPI 总线并重新设置主机模式。另一个硬件状态位用于通过 **SPIx_IDL_STA** 寄存器在从机带 NSS 模式报告 SPI 总线空闲状态。

[注释]: **SPIx_IDL_STA** 寄存器不支持于 MG32F02A132/072。

参照 “[SPI 双主机通信](#)” 节以获取更多信息。

图 17-30. SPI 主机模式改变检测



17.17. SPI DMA 操作

17.17.1. DMA 模块设置

当芯片支持 DMA（直接内存访问）控制器时，用户可以在 DMA 数据传输前，在 DMA 模块中设置关于来源/目的设备、通道请求仲裁等 DMA 设置。DMA 来源和目的可以是内存或外设。

参照 DMA 章以获取更多关于 DMA 模组设置的细节。

17.17.2. SPI DMA 控制

DMA 设置完成后，用户需设置 SPI 模块的 DMA 使能位 **SPIx_DMA_RXEN** 和 **SPIx_DMA_TXEN**。

最后，相关的通道请求起始位 **DMA_CHn_REQ** 是必须的，用于设置 DMA 传输启动(n = DMA 通道标号)。然后传输源和目的设备会置起 RX/TX 请求信号到 DMA 控制器，DMA 控制器便会置起接收信号到请求源/目的设备。此时，该数据传输连接是用于 DMA 传输的。

- **SPI RX 到 DMA**
SPIx_DMA_RXEN 寄存器位是用于使能从 SPI 输入到 DMA 目的地的 DMA 数据发送。
在 DMA 发送周期中，接收标志 **SPIx_RXF** 是被硬件屏蔽的。
- **SPI TX 来自 DMA**
SPIx_DMA_TXEN 寄存器位是用于使能 DMA 数据从 DMA 源到 SPI 发送输出的数据发送。
在 DMA 发送周期中，发送标志 **SPIx_TXF** 是被硬件屏蔽的。通常，发送完成标志 **SPIx_TCF** 会在 DMA 发送完成后被置起。

17.17.3. SPI 的 DMA 中断标志控制

DMA 工作周期中，模块的中断标志会控制和执行 3 种类型，如下表格。其中一方在 DMA 工作过程中被屏蔽，另一方会在标志被置起后禁用 DMA 功能。此时，硬件会清空 **SPIx_DMA_TXEN** 和 **SPIx_DMA_RXEN** 位。其他操作与 DMA 操作中不执行的操作相同。

表 17-9. SPI DMA 功能中断标志控制

行为	DMA 操作中屏蔽标志	标志置起后 DMA 被禁用(*1)	一般控制
外设	(数据溢出标志)	(错误/检测标志)	(其他标志)
SPIx	SPIx_TCF (*2) SPIx_RXF SPIx_TXF	SPIx_MODF SPIx_WEF SPIx_ROVRF SPIx_TUDRF	SPIx_IDLF

注释-1：当标志被置起，若相关中断使能位没被使能，它不会强行禁用外设 DMA。

注释-2：SPIx_TCF 会在 DMA TX 完成后被置起。

18. 定时器(通用定时器)

18.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片有最多 7 个定时器/计数器模块：TM00、TM01、TM10、TM16、TM20、TM26 和 TM36。他们全部都可以被设置为定时器或事件计数器。

TM0x 有 1 个 8 位预分频器的 8 位定时器/计数器；TM1x 有 1 个 16 位预分频器的 16 位定时器/计数器；TM2x 有 1 个 16 位预分频器和内嵌 2 个输入捕获/输出比较通道的 16 位定时器/计数器；TM36 有 1 个 16 位预分频器和内嵌 4 个输入捕获/输出比较通道的 16 位定时器/计数器。

注释：（x = 模块标号；n = 输入输出通道号；m = I/O 线号）会被用于该章的寄存器、信号和引脚/端口描述中。

[EX]: **TMx_EN**, **TMx_OCn_OEm** ~ x 表示模块标号{00,01,10,16,20,26,36}; n = 输入输出通道号{0~3}; m = I/O 线号{0,1,2}。

18.2. 特性

- 定时器模块一般功能
 - 可选择 **Full-counter, Cascade, Separate** 模式
 - 多个内部和外部信号作为定时器时钟源或触发源
 - 将内部定时器事件输出到引脚或其他模块作为输入触发事件
 - 支持用于触发源功能的定时器复位、触发启动和时钟门控
 - 定时器溢出作为时钟输出到外部引脚输出
 - 可编程计数器 **auto-stop** 模式
 - 主要计数器向上/向下控制（仅 TM16/TM26/TM36）
 - 第二计数器支持向上/向下计数控制（**Cascade /Separate** 模式）
- 提供 TM36 定时器模块
 - 32 位定时器/计数器：有 16 位预分频器的 16 位定时器
 - 4 个 CCP (输入捕获/输出比较/PWM)通道
 - 3 个具有 OCN (互补输出比较)的 CCP 通道
 - 具有中心对齐/边沿对齐/可设置死区控制功能的 PWM
 - 支持 OC 比较器分割为两个独立的比较器模式
 - 中止输入控制
 - 支持 QEI(正交编码接口)
 - 1 个 IC 和 3 个 OC 可使用 DMA
 - 外部输入定时器向上/向下计数控制
- 提供 TM2x 定时器模块
 - 32 位定时器/计数器：有 16 位预分频器的 16 位定时器
 - 2 个 CCP (输入捕获/输出比较/PWM)通道
 - 2 个具有 OCN (互补输出比较)的 CCP 通道
 - 支持 QEI(正交编码接口) (仅 TM26)
 - 具有边缘对齐的 PWM 功能
 - 支持 OC 比较器分割为两个独立的比较器模式
- 提供 TM1x 定时器模块
 - 32 位定时器/计数器：有 16 位预分频器的 16 位定时器
- 提供 TM0x 定时器模块
 - 16 位定时器/计数器：有 8 位预分频器的 8 位定时器

❖ MG32F02A132/072

- 提供 7 个定时器/计数器：TM00,TM01,TM10,TM16,TM20,TM26,TM36

❖ MG32F02A032

- 提供 5 个定时器/计数器：TM00,TM01,TM10,TM16,TM36

18.3. 配置

18.3.1. 芯片配置

下面的表格展示了芯片的定时器模块配置。

表 18-1. 定时器配置

芯片	定时器模块						
	TM00	TM01	TM10	TM16	TM20	TM26	TM36
MG32F02A132	V	V	V	V	V	V	V
MG32F02A072	V	V	V	V	V	V	V
MG32F02A032	V	V	V	V			V

注释 V: 包含

18.3.2. 模块功能

下面的表格展示了定时器模块包含的功能。

表 18-2. 定时器模块功能

定时器模块	TM00	TM01	TM10	TM16	TM20	TM26	TM36
定时器/计数器总位数	16	16	32	32	32	32	32
定时器级联模式	yes	yes	yes	yes	yes	yes	yes
定时器分离模式	yes	yes	yes	yes	yes	yes	yes
定时器全计数模式	yes	yes	yes	yes	yes	yes	yes
独立通道					2	2	4
内部 TRGI 线	8	8	8	8	8	8	8
外部 TRGI 线	1	1	1	1	1	1	1
输出 TRGO 线	1	1	1	1	1	1	1
输出 CKO 线	1	1	1	1	1	1	1
输入捕获 IC 线					2	2	4
输出 OC 线					2	2	4
输出 OCN 线					2	2	3
输出 OCH 线					2	2	4
输入中止线							1
PWM 一分二					yes	yes	yes
PWM 边沿对齐					yes	yes	yes
PWM 中心对齐							yes
死区发生器							yes
1st 定时器向上向下计数	U	U	U	U/D	U	U/D	U/D
2nd 定时器向上向下计数	U/D	U/D	U/D	U/D	U/D	U/D	U/D
定时器自动停止	yes	yes	yes	yes	yes	yes	yes
QEI 定时器 U/D 控制				—		yes	yes
3 输入 XOR 到 CH-0							yes
DMA 支持							yes

注释 1. 定时器级联模式 ~ 16 位计数器+16 位预分频器或 8 位计数器+8 位预分频器

2. 定时器分离模式 ~ 两个 16 位计数器或 8 位计数器

3. 定时器全计数模式 ~ 32 位计数器或 16 位计数器

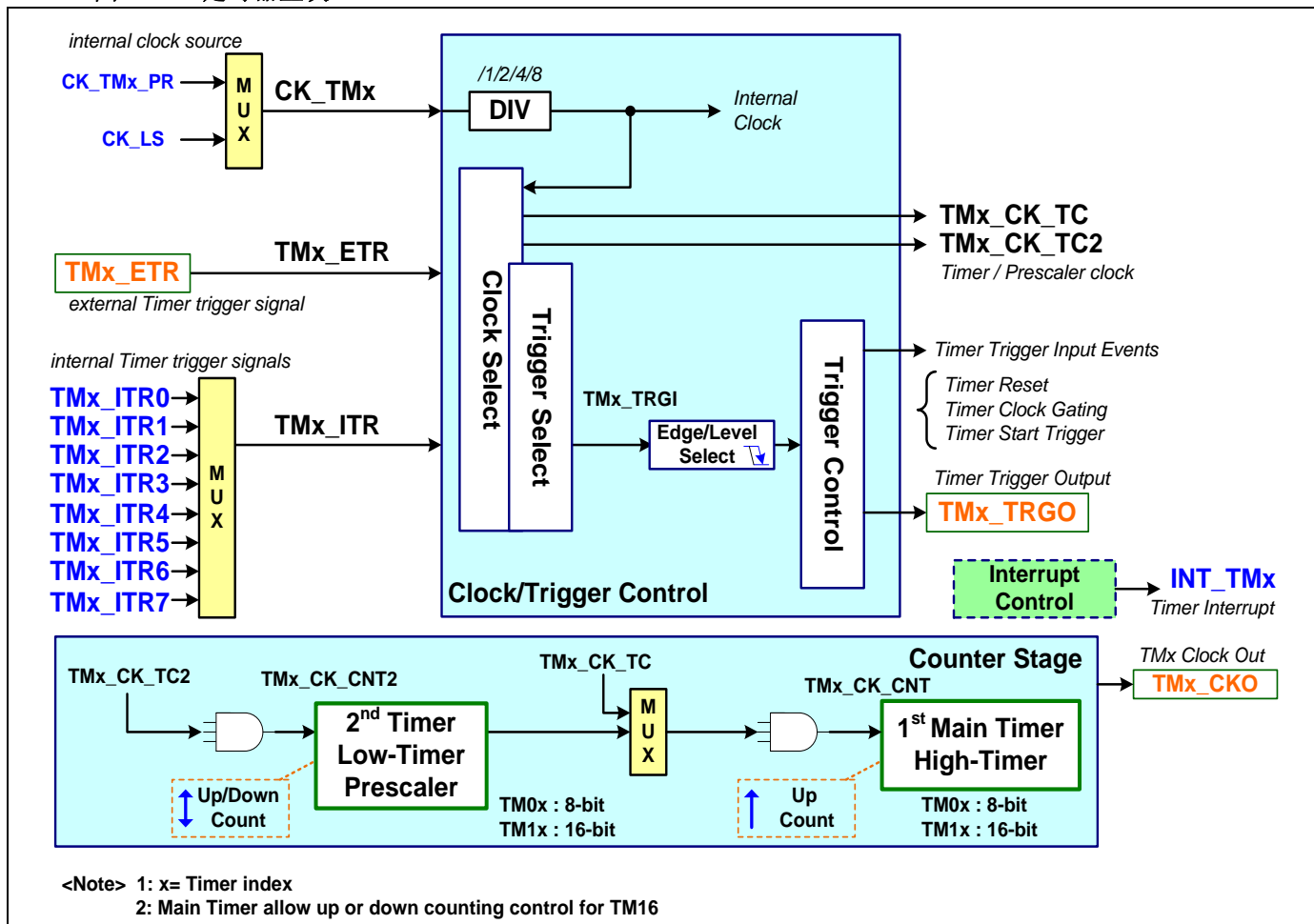
18.4. 控制块

TMx 模块包含 1 个时钟/触发控制块、1 个计数器块、1 个捕获/比较控制块和通道 I/O 控制（仅 TM2x、TM3x）的输入/输出块和中止控制块（仅 TM36）。

下面的图表展示了 TM0x/TM1x, TM2x, TM3x 模块的定时器控制块。

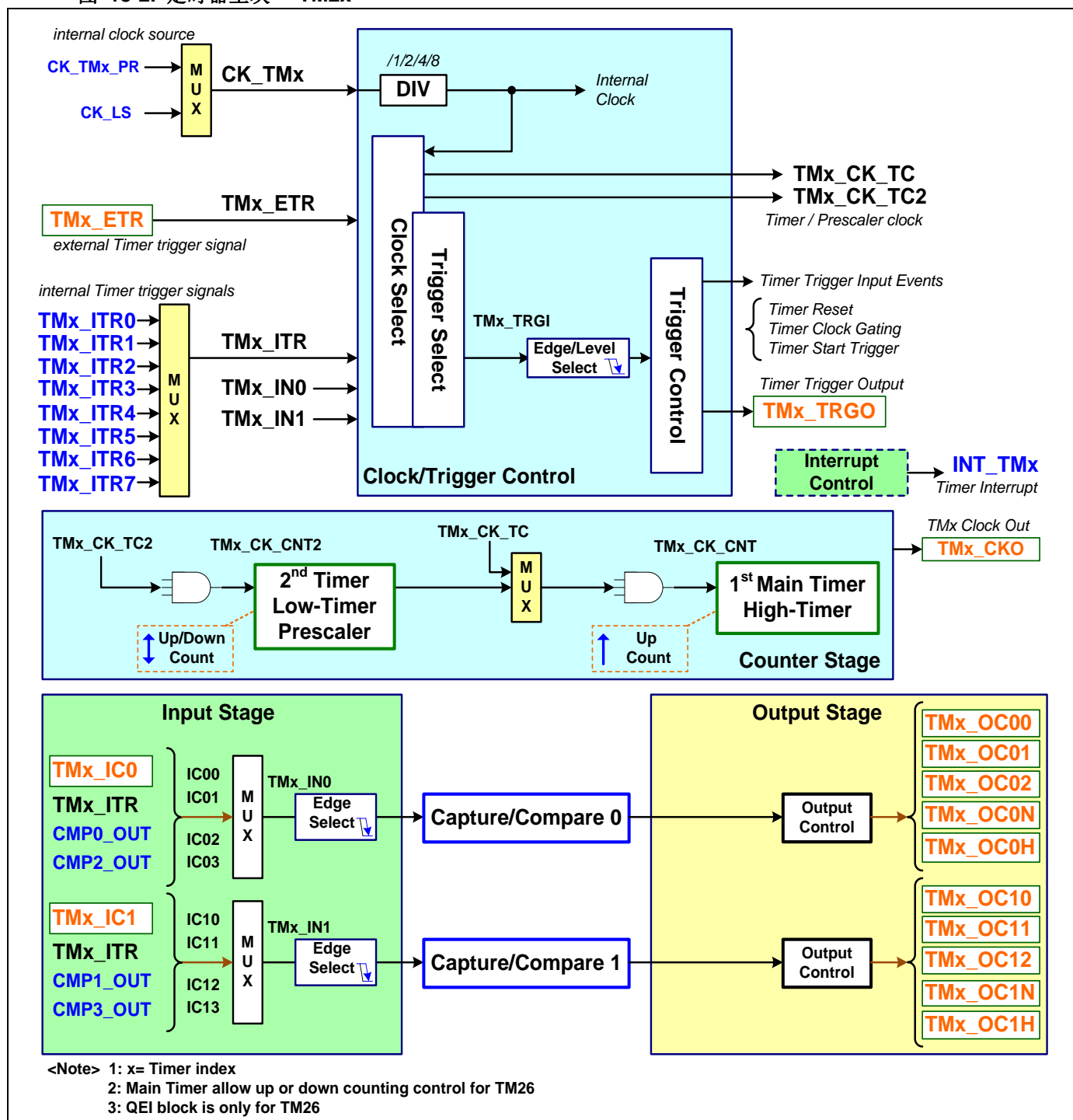
● TM0x/1x 定时器主块图表

图 18-1. 定时器主块 ~ TM0x/TM1x



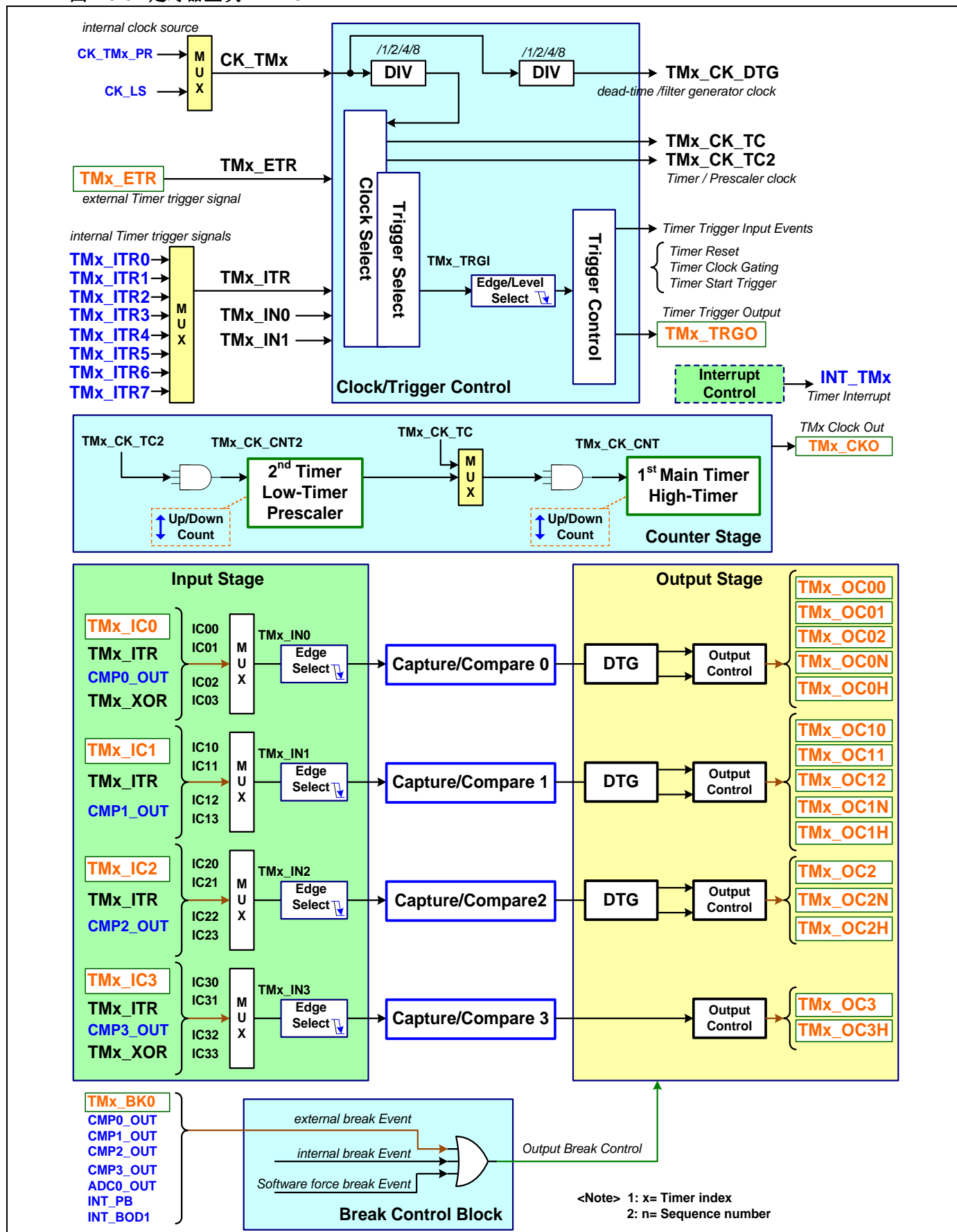
● TM2x 定时器主块图表

图 18-2. 定时器主块 ~ TM2x



- TM3x 定时器主块图表

图 18-3. 定时器主块 ~ TM3x



18.5. IO 线

18.5.1. IO 信号

- **TMx_CKO**

定时器上溢输出信号，并作为时钟输出。

- **TMx_TRGO**

从定时器内部事件或信号选择的可选输出信号，作为触发信号输出到引脚或其他内部模块。

- **TMx_ETR**

外部触发或时钟输入信号。它可被设置并作为定时器时钟使用。此外它还可被设置为定时器复位、定时器开始计数或定时器时钟门控的触发输入。

- **TMx_ICn**

通道 n 的定时器输入捕获信号。

- **TMx_OCn**

通道 n 的定时器比较或 PWM 输出信号。

- **TMx_OCnN**

通道 n 的 **TMx_OCn** 的互补输出信号。

- **TMx_OCnH**

高位通道的通道 n 的定时器比较或 PWM 输出信号。该信号仅用于“2 个 8 位比较/PWM”模式。

- **TMx_BK0**

通过寄存器设置用于单独停止或暂停 **TMx_OCn**, **TMx_OCnN** 和 **TMx_OCnH** 信号的定时器中止输入信号。

18.5.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。参照用户手册 GPIO 章中“[IO 模式](#)”节以获取更多关于 IO 模式设置的信息。

每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“[功能复用选择](#)”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“[引脚功能复用表](#)”以获取更多信息。

18.6. 使能和时钟

18.6.1. 定时器使能

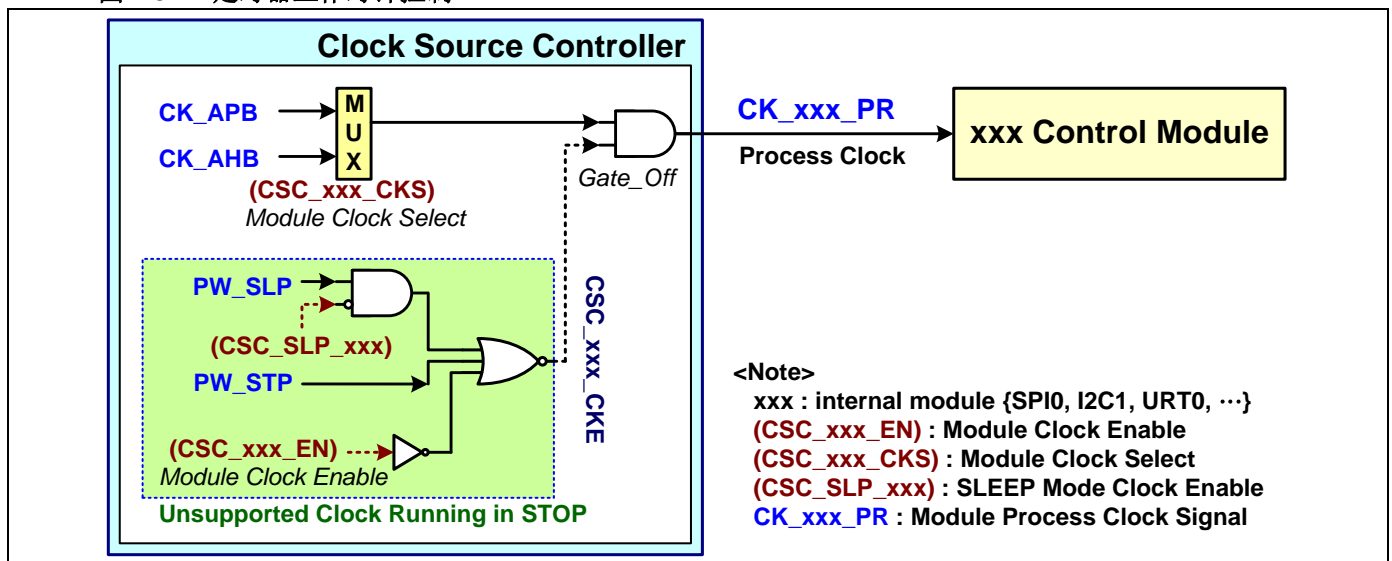
每个 TMx 模块都有 2 个使能控制位 **TMx_EN** 和 **TMx_EN2**。该定时器内建 TMx 模块，支持 3 种工作模式：(1) 级联模式 (2) 分离模式 (3) 全计数模式。因此 TMx 定时器可为 1 个全计数定时器或分离成两个定时器。

TMx_EN 位用于使能全计数器、高位定时器或主定时器用于级联模式、分离模式或全计数模式。**TMx_EN2** 位用于使能低位定时器或 2nd 定时器用于级联模式、分离模式。

18.6.2. 定时器工作时钟控制

该模块工作时钟 **CK_TMx_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 CSC (时钟源控制器) 模块。该时钟可通过 **CSC_TMx_EN** 寄存器使能并通过 **CSC_TMx_CKS** 寄存器选择时钟源来自 APB 或 AHB。用户可以在芯片进入 **SLEEP** 模式之前通过设置 **CSC_SLP_TMx** 寄存器规划在 **SLEEP** 模式下是否让模块时钟继续运行。参照系统时钟章以获取更多信息。

图 18-4. 定时器工作时钟控制



18.6.3. 定时器模块时钟和触发源

● 模块内部时钟

用户可通过 **TMx_CKI_SEL** 寄存器选择时钟源来自模块工作时钟 **CK_TMx_PR** 还是内部低速时钟 **CK_LS**。此外，该模块还通过 **TMx_CKI_DIV** 寄存器设置提供一个时钟分频器用于生成内部时钟 **CK_TMx_INT**。该时钟分频器可将时钟频率分频为 1/2/4/8。

[注释]: 当 TMx 模块工作在 QE1 功能时, **TMx_CKI_DIV** 寄存器只能被设置为 0 且分频值为 1。

CK_TMx_INT 信号输出为 DTG 的内部时钟源或其中 1 个 TMx 定时器时钟源。此外，还有 1 个时钟分频器可通过 **TMx_DTG_DIV** 寄存器进行 1/2/4/8 分频产生 DTG 输入时钟 **TMx_CK_DTG**。

[注释]: 通常内部时钟频率需比模块工作时钟慢至少 1/2。

● 内部和外部定时器触发信号

内部定时器触发信号 **TMx_ITR[7:0]** 可通过 **TMx_ITR_MUX** 寄存器进行选择以输出 **TMx_ITR** 信号。**TMx_ITR6** 和 **TMx_ITR7** 触发信号是用于所有定时器模块的全局信号。它们可被选择为来自内部定时器模块或其他外设模块的触发信号。参照用户手册的“APB 一般控制”章以获取更多信息。

1 个外部定时器触发信号 **TMx_ETR** 来自 **TMx_ETR** 输入。**TMx_ETR** 输入可通过 GPIO AFS 进行设置在哪个 GPIO 引脚。

● 定时器输入触发信号和外部时钟源

TMx_ITR 信号与外部定时器触发信号 **TMx_ETR** 结合；定时器外部通道输入信号 **TMx_IN0/TMx_IN1** 作为定时器输入触发源或定时器外部时钟源。用户可通过 **TMx_CKE_SEL** 寄存器选择定时器外部时钟源到

18.7. 中断和事件

18.7.1. 定时器更新事件

TMx 定时器可分为主定时器和 2nd 定时器。参照“[定时器工作模式](#)”的描述以获取更多关于定时器工作模式的信息。主定时器可输出上溢/下溢信号 TOF/TUF，向上/向下计数方向标志 DIRF，计数方向改变标志 DIRCF 标志。2nd 定时器可输出上溢/下溢信号 TOF2/TUF2。

TMx 模块可产生 2 种更新事件标志 **TMx_UEV** 和 **TMx_UEV2** 到 **TMx_TRGO** 输出。**TMx_UEV** 信号可被用于触发 TM2x/TM3x 模块的比较预载寄存器的内容加载到比较阴影寄存器中。用户可通过设置 **TMx_UEV_DIS** 寄存器使能或禁用更新事件功能。更新事件信号源可来自主定时器上溢/下溢事件、软件更新事件发生位 (**TMx_USW_EN**) 或外部触发输入 **TMx_TRGI**。用户可通过 **TMx_UEV_SEL** 寄存器选择输出电平来自于主定时器上溢 TOF 事件、下溢 TUF 事件或所有事件。

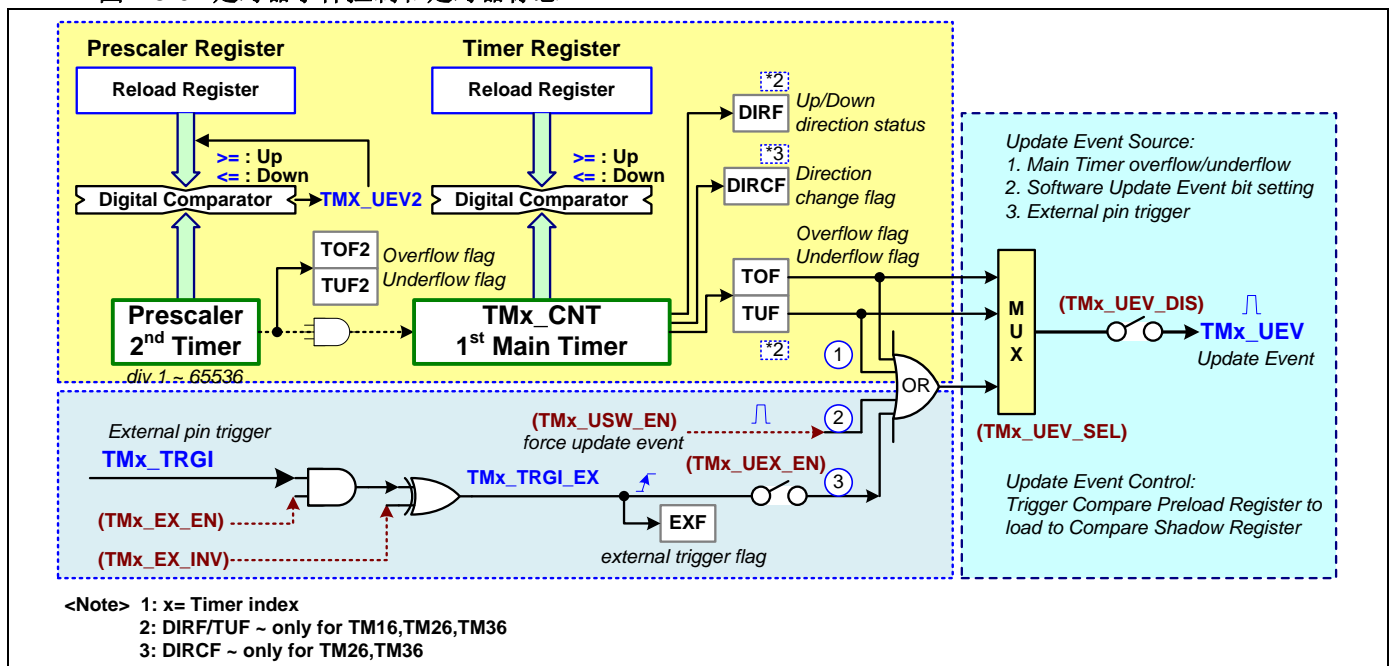
TMx_UEV2 信号会在 2nd 定时器上溢或下溢事件发生时产生。这也会用于复位计数器用于向上计数或重载定时器重载寄存器用于向下计数。

当用户设置 **TMx_USW_EN** 寄存器位时，在“更新事件操作”完成后，该位会自动被硬件自动清除。

对于外部触发输入 **TMx_TRGI**，有 1 个外部触发事件使能位 **TMx_EX_EN** 用于使能信号输入 **TMx_TRGI**。用户可设置 **TMx_EX_INV** 寄存器反相 **TMx_TRGI** 信号。当芯片检测到 **TMx_TRGI_EX** 信号的上升沿，若相关中断使能位被使能，EXF 标志会被置起。该情况下，若定时器外部触发更新事件使能位 **TMx_UEX_EN** 已使能，该外部引脚触发事件会强行做“更新事件操作”。

下面的图表展示了定时器事件控制和定时器标志。

图 18-6. 定时器事件控制和定时器标志



18.7.2. 定时器中断控制和状态

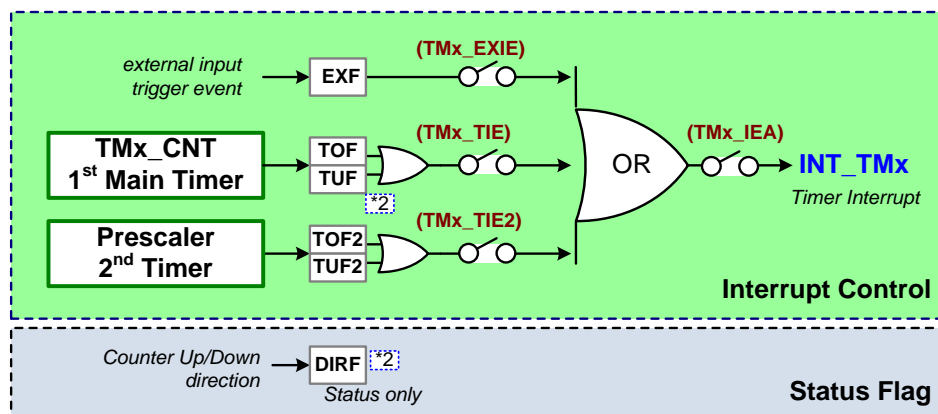
TMx 模块中有 1 种信号 **INT_TMx**。INT_TMx 发送到外部中断控制器 (EXIC) 作为中断事件。

中断标识是用于中断服务程序 (ISR) 流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **TMx_IEA** 用于使能和禁用该模块的所有中断源。

该模块中有一些只读的状态位，用于提供内部控制状态，DIRF 标志 (**TMx_DIRF**) 表明主定时器向上/向下计数方向。参照相关状态位寄存器描述以获取更多信息。

下面的图表展示了 TM0x 和 TM1x 的定时器状态和中断控制。

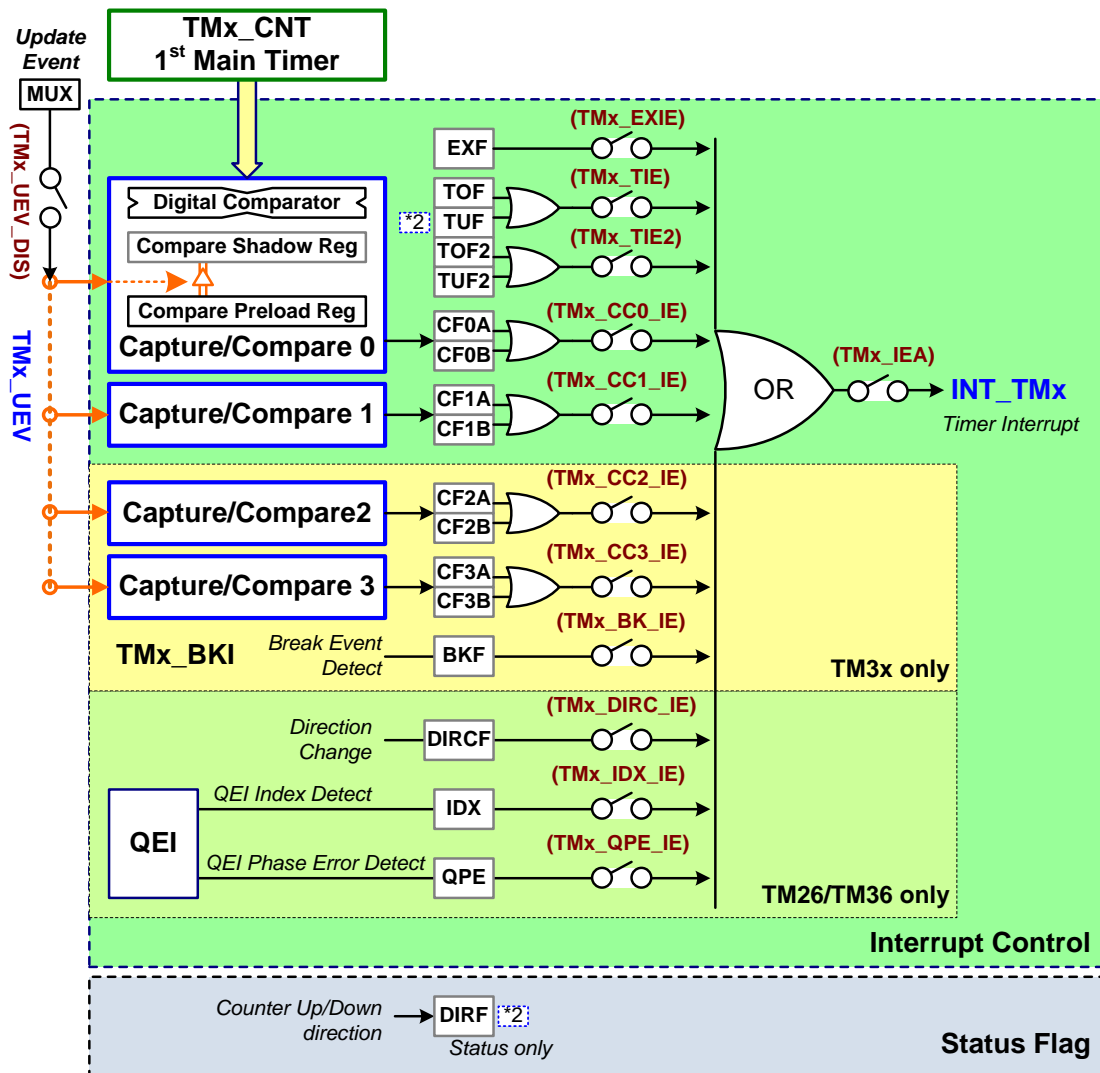
图 18-7. 定时器状态和中断控制 – TM0x/1x



<Note> 1: x= Timer index
2: DIRF/TUF ~ only for TM16

下面的图表展示了 TM2x 和 TM3x 的定时器中断控制和比较寄存器更新控制。

图 18-8. 定时器状态和中断控制 – TM2x/3x



<Note> 1: x= Timer index
2: DIRF/TUF ~ only for TM26,TM36

18.7.3. 定时器中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的

信息。

- **BKF**

定时器中止输入标志(**TMx_BKF**)，相关的中断使能寄存器位是 **TMx_BKIE**。

- **EXF**

定时器外部触发标志(**TMx_EXF**)，相关的中断使能寄存器位是 **TMx_EXIE**。

- **TOF / TUF**

主定时器上溢/下溢标志(**TMx_TOF** / **TMx_TUF**)，这两个标志相关的中断使能寄存器位是 **TMx_TIE**。

- **TOF2 / TUF2**

2nd 定时器上溢/下溢标志(**TMx_TOF2** / **TMx_TUF2**)，这两个标志相关的中断使能寄存器位是 **TMx_TIE2**。

- **CFnA / CFnB**

定时器输入捕获和输出比较事件标志(**TMx_CFnA** / **TMx_CFnB**)，这两个标志相关的中断使能寄存器位是 **TMx_CCnE**。(n = 定时器 IC/OC/PWM 通道标号)

- **DIRCF**

主定时器向上/向下计数方向改变标志(**TMx_DIRCF**)，相关的中断使能寄存器位是 **TMx_DIRC_IE**。当主定时器向上/向下计数方向被外部 **QEI** 信号改变时，该标志会被置起，若中断使能位被使能，该芯片还会产生中断。

- **IDXF**

主定时器 **QEI** 外部标号信号输入启动检测和内部定时器复位标志(**TMx_IDXF**)，相关的中断使能寄存器位是 **TMx_IDX_IE**。当主定时器 **QEI** 外部标号信号输入检测到启动时，该标志会被置起，主定时器会复位。此外，该芯片会在中断使能位使能时发生中断。

- **QPEF**

主定时器 **QEI** 相位状态改变错误检测标志(**TMx_QPEF**)，相关的中断使能寄存器位是 **TMx_QPE_IE**。当检测到主定时器 **QEI** 输入相位状态改变错误时，该信号会被置起，且当中断使能位被使能时发生中断。

18.8. 定时器操作

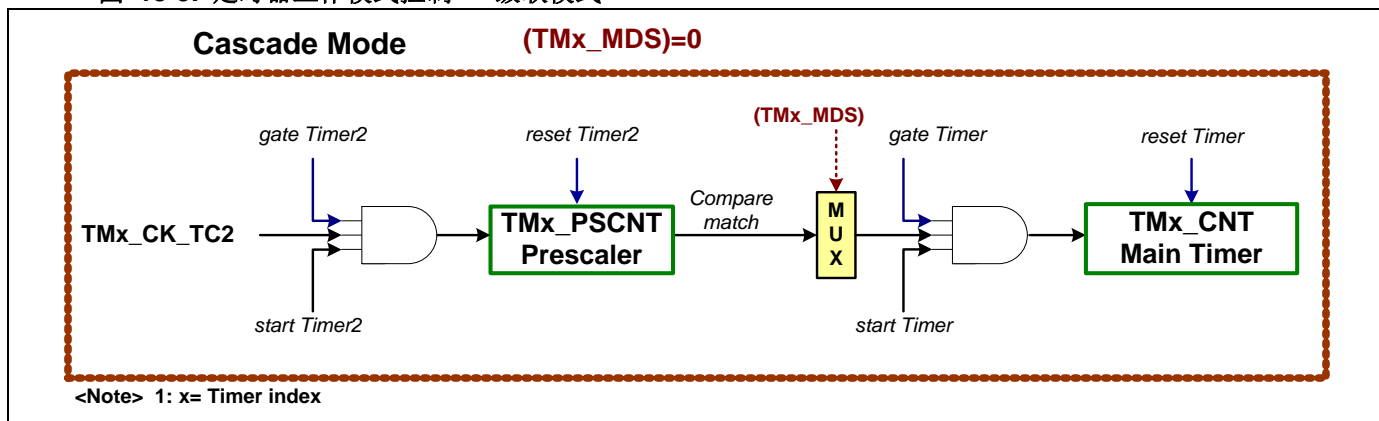
18.8.1. 定时器工作模式

TMx 支持 3 种工作模式：(1) 级联模式 (2) 分离模式 (3) 全计数模式。用户可设置 **TMx_MDS** 寄存器选择这些定时器工作模式。下面的图表展示了这 3 种定时器模式的控制块。

- 定时器工作模式 – 级联模式

该模块包含 1 个含有 1 个预分频器的主定时器。时钟源 **TMx_CK_TC2** 可被预分频器分频，并作为主定时器/计数器的输入时钟。

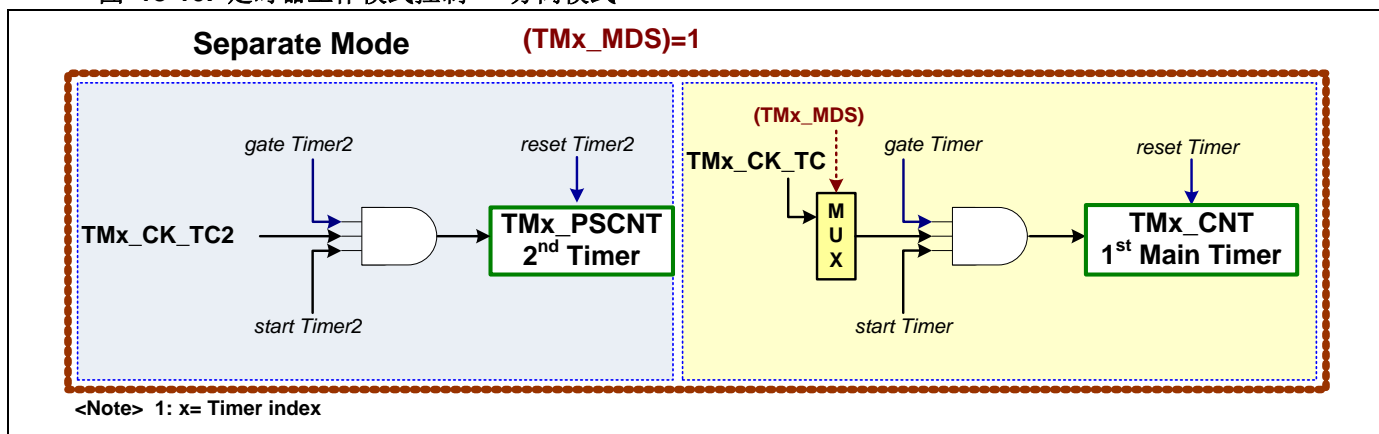
图 18-9. 定时器工作模式控制 – 级联模式



- 定时器工作模式 – 分离模式

主定时器和预分频器分别作为 1st 定时器和 2nd 定时器的独立定时器。时钟源 **TMx_CK_TC** 用于 1st 定时器；时钟源 **TMx_CK_TC2** 用于 2nd 定时器。

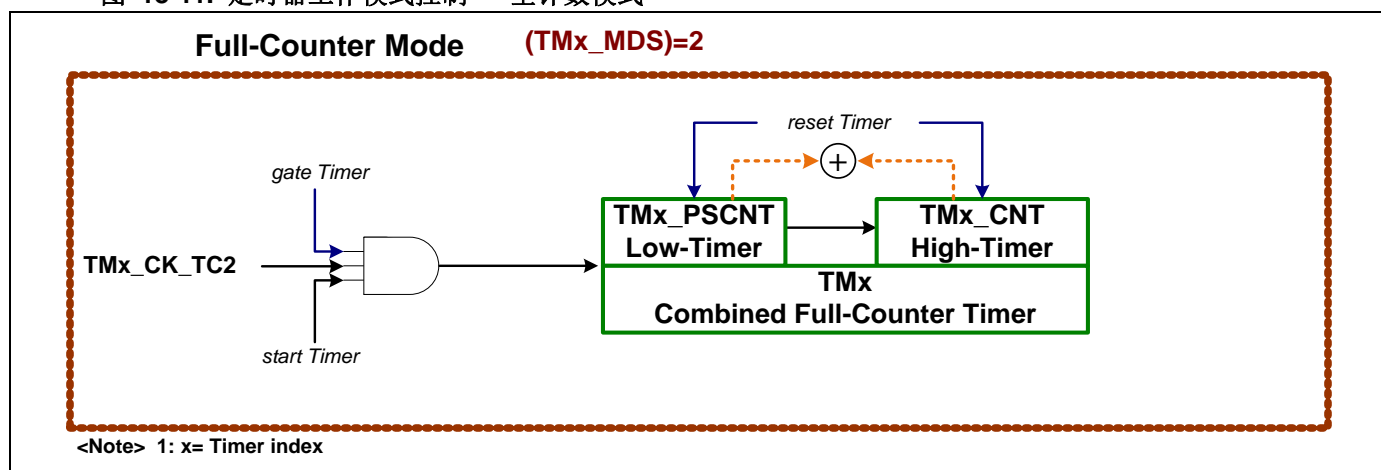
图 18-10. 定时器工作模式控制 – 分离模式



- 定时器工作模式 – 全计数模式

主定时器和预分频器/2nd 定时器可融合成 1 个完整定时器或计数器。时钟源 **TMx_CK_TC2** 是作为该融合的全计数定时器的输入时钟。

图 18-11. 定时器工作模式控制 – 全计数模式



18.8.2. 定时器和计数器控制

定时器模块内建 1 个主定时器和 1 个时钟预分频器/2nd 定时器用于这 3 种工作模式。参照前面的章节获取关于工作模式的信息。

主定时器包含 16 位计数器用于 TM1x/2x/3x 模块、8 位计数器用于 TM0x 模块。此外，时钟预分频器/2nd 定时器包含 16 位计数器用于 TM1x/2x/3x 模块、8 位计数器用于 TM0x 模块。计数器值寄存器 **TMx_CNT** 和 1 个定时器重载寄存器 **TMx_ARR** 用于主定时器。此外，计数器值寄存器 **TMx_PSCNT** 和 1 个定时器重载寄存器 **TMx_PSARR** 用于时钟预分频器/2nd 定时器。根据设计，有一个 **TMx_CNTA** 寄存器，它是 **TMx_CNT** 的别名，并在 32 位寄存器中与 **TMx_PSCNT** 结合，用于完全计数器模式。

用户可在 **TMx_DIR** 寄存器里设置主定时器计数方向，在 **TMx_DIR2** 寄存器里设置 2nd 定时器计数方向。

[注释]: 主定时器计数方向控制只在 TM16, TM26 和 TM36 模块中支持。

[注释]: 在 MG32F02A132/072 中, TM36 模块中不支持 **TMx_DIR2** 2nd 定时器计数方向控制是通过 **TMx_DIR** 支持的。

● 自动重载寄存器值限制

主定时器/计数器的自动重载寄存器 **TMx_ARR** 设置范围可被设置 0~0xFF 用于 8 位定时器、0~0xFFFF 用于 16 位定时器。但一些特殊情况下范围有所限制。

当所有的通道被设置为双 8 位 OC/PWM 模式时，自动重载寄存器值会被限制为 0x00zz (zz={0x00~0xFF})。当一些通道被设置为双 8 位 OC/PWM 模式，一些通道被设置为 16 位 OC/PWM 模式时，自动重载寄存器值会被限制为 0xzzFF (zz={0x00~0xFF})。

参照“[定时器输出比较和 PWM](#)”节以获取更多信息。

18.8.3. 定时器工作模式和控制验证

下面的表格展示了三种工作模式下，QE1 功能下的控制验证与主定时器和 2nd 定时器的控制行为。

表 18-3. 定时器工作模式和控制验证

模式	设置	定时器控制	主定时器控制				2nd 定时器控制				QE1 控制
	MDS		EN	DIR	RST	GT	EN2	DIR2	RST2	GT2	
级联	0	独立	V	V	V	V	V	V	V	V	全部定时器
分离	1	独立	V	V	V	V	V	V	V	V	主定时器
全计数	2	仅主定时器	V	V	V	V	X	X	X	X	全部定时器

仅主定时器：通过主定时器寄存器同时控制主定时器和 2nd 定时器。

V：有效控制, X：无效控制

QE1：定时器外部输入向上/向下控制

EN 和 EN2 代表定时器使能/起始控制 (**TMx_EN** 和 **TMx_EN2** 寄存器)；DIR 和 DIR2 代表定时器计数方向控制(**TMx_DIR** 和 **TMx_DIR2** 寄存器)；RST 和 RST2 代表触发输入定时器复位控制；GT 和 GT2 代表触发输入定时器时钟门控。

18.9. 定时器触发控制块

该触发控制块有两种功能：控制定时器触发输入事件、控制定时器触发输出事件。

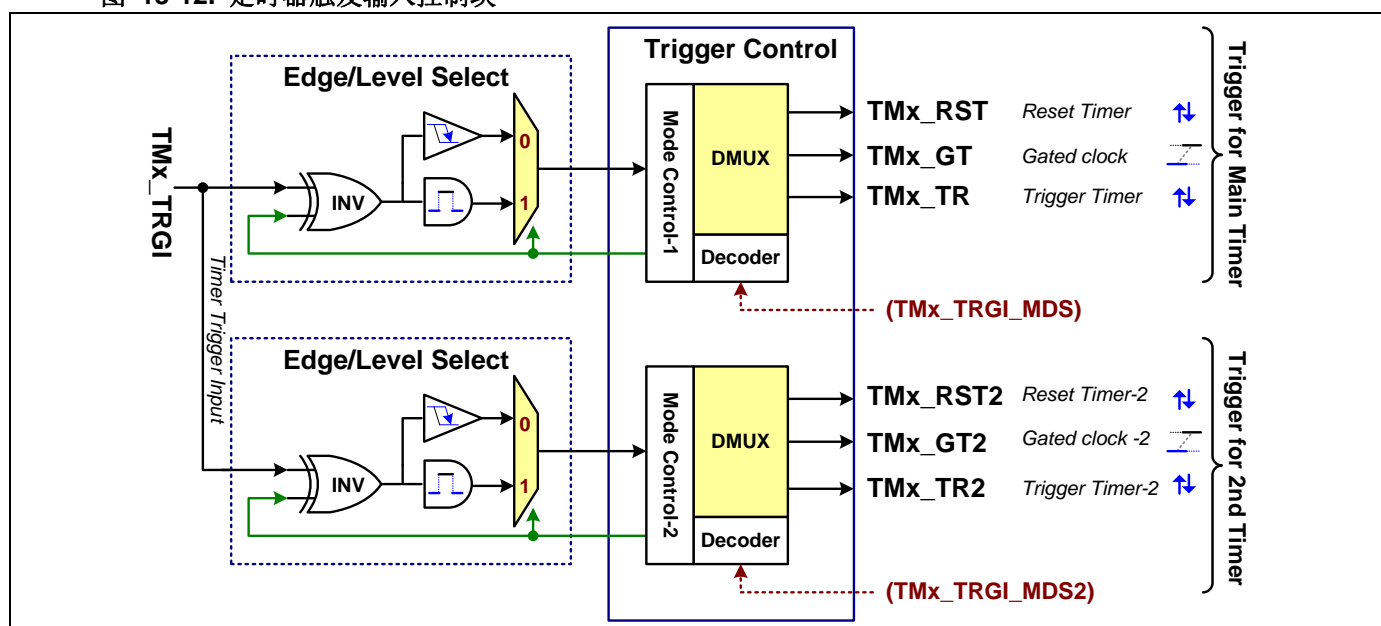
18.9.1. 定时器触发输入事件

定时器触发输入事件可做“复位定时器”、“门控时钟”和“定时器启动触发”。用户可分别设置 **TMx_TRGI_MDS** 和 **TMx_TRGI_MDS2** 寄存器为主定时器和 2nd 定时器选择触发输入事件。定时器触发输入事件的输入源可通过 **TMx_TRG_MUX** 寄存器选择外部触发信号 **TMx_ETR**、内部触发信号 **TMx_ITR[7:0]** 或外部通道输入信号 **TMx_IN0/TMx_IN1**。

“复位定时器”用于复位向上计数的计数器或重载向下计数的定时器；“门控时钟”用于当检测到选择的输入信号电平时，通过关闭门控定时器时钟停止计数器计数；“定时器启动触发”用于检测到选择的输入信号边沿时，当定时器被使能时启动计数器计数。

下面的图表展示了定时器触发输入控制块。

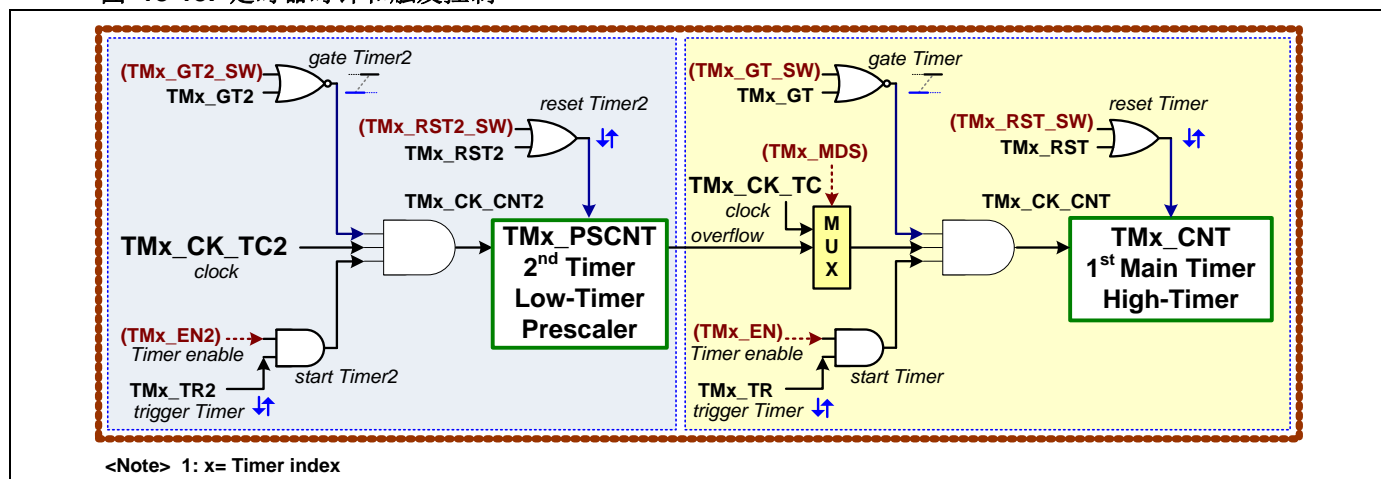
图 18-12. 定时器触发输入控制块



- 定时器内部触发软件控制

该模块通过用于主定时器的 **TMx_RST_SW**, **TMx_GT_SW** 寄存器和 2nd 定时器的 **TMx_RST2_SW**, **TMx_GT2_SW** 寄存器支持对复位定时器和门控时钟的软件控制功能。

图 18-13. 定时器时钟和触发控制



- 定时器内部触发输入信号

下面的表格展示了每个定时器模块的内部触发信号。特殊的是，**ITR6** 和 **ITR7** 是所有定时器模块的通用信号且通过设置 **APB_ITR6_MUX** 和 **APB_ITR7_MUX** 寄存器来选择触发源信号。参照 APB 一般控制章以获取更多信息。

[注释]: TM20/TM26 不支持于 MG32F02A032。

表 18-4. 定时器内部触发信号表 – MG32F02A132/072

定时器模块	TM00	TM01	TM10	TM16	TM20	TM26	TM36
ITR0 信号	TM10_TRGO	TM16_TRGO	TM00_TRGO	TM01_TRGO	TM00_TRGO	TM01_TRGO	TM10_TRGO
ITR1 信号	TM20_TRGO	TM26_TRGO	TM20_TRGO	TM26_TRGO	TM10_TRGO	TM16_TRGO	—
ITR2 信号	CMP0_OUT	CMP1_OUT	CMP0_OUT	CMP1_OUT	CMP0_OUT	CMP1_OUT	CMP0_OUT
ITR3 信号	CMP2_OUT	CMP3_OUT	CMP2_OUT	CMP3_OUT	CMP2_OUT	CMP3_OUT	CMP1_OUT
ITR4 信号	INT_PA	INT_PC	INT_PA	INT_PC	INT_PA	INT_PC	INT_PA
ITR5 信号	INT_PB	INT_PD	INT_PB	INT_PD	INT_PB	INT_PD	INT_PC
ITR6 信号	ITR6	ITR6	ITR6	ITR6	ITR6	ITR6	ITR6
ITR7 信号	ITR7	ITR7	ITR7	ITR7	ITR7	ITR7	ITR7

注释-1 ITR6 = {TM00_TRGO, TM10_TRGO, TM20_TRGO, TM36_TRGO, INT_PB, URT1_TMO, URT2_BRO, URT2_TMO}

ITR7 = {TM01_TRGO, TM16_TRGO, TM26_TRGO, ADC0_OUT, INT_PD, URT1_BRO, URT3_BRO, URT3_TMO, ICKO_INT, RTC_OUT, TM36_XOR}

表 18-5. 定时器内部触发信号表 – MG32F02A032

定时器模块	TM00	TM01	TM10	TM16	TM36
ITR0 信号	TM10_TRGO	TM16_TRGO	TM00_TRGO	TM01_TRGO	TM10_TRGO
ITR1 信号	-	-	-	-	-
ITR2 信号	CMP0_OUT	CMP1_OUT	CMP0_OUT	CMP1_OUT	CMP0_OUT
ITR3 信号	-	-	-	-	CMP1_OUT
ITR4 信号	INT_PA	INT_PC	INT_PA	INT_PC	INT_PA
ITR5 信号	INT_PB	INT_PD	INT_PB	INT_PD	INT_PC
ITR6 信号	ITR6	ITR6	ITR6	ITR6	ITR6
ITR7 信号	ITR7	ITR7	ITR7	ITR7	ITR7

注释-1 ITR6 = {TM00_TRGO, TM10_TRGO, TM36_TRGO, INT_PB, URT1_TMO}

ITR7 = {TM01_TRGO, TM16_TRGO, ADC0_OUT, INT_PD, URT1_BRO, ICKO_INT, RTC_OUT, TM36_XOR}

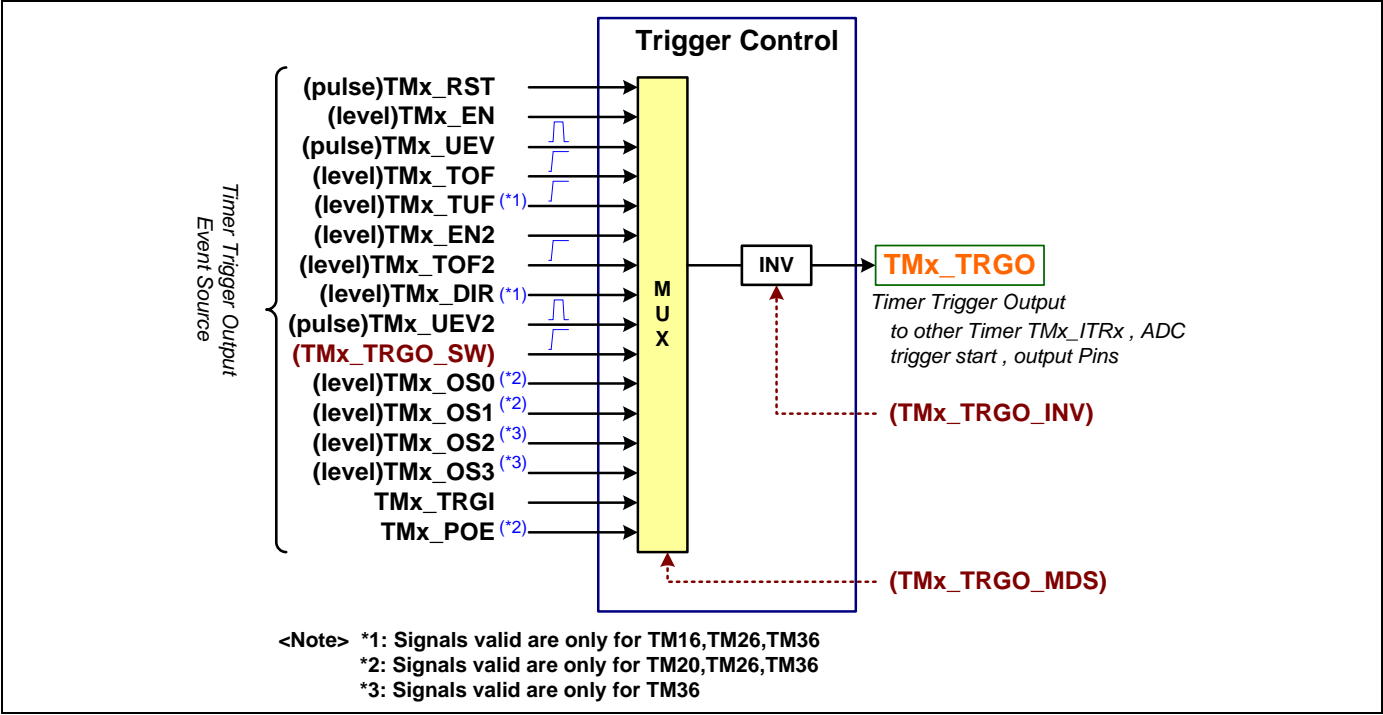
18.9.2. 定时器触发输出事件

定时器触发输出事件的来源可以来自该定时器模块的许多内部事件或信号，并输出到 **TMx_TRGO** 输出。此外，用户可使用软件寄存器 **TMx_TRGO_SW** 直接设置触发输出。这些输出事件源可通过 **TMx_TRGO_MDS** 寄存器选择，并可通过 **TMx_TRGO_INV** 寄存器反相输出信号。

当 **TMx_TRGO_MDS** 选择 **TMx_UEV** 信号作输出信号，用户可通过 **TMx_UEV_SEL** 寄存器选择主定时器上溢 TOF 事件和/或下溢 TUF 事件作输出电平功能。

下面的图表展示了定时器触发输出控制块。

图 18-14. 定时器触发输出控制块



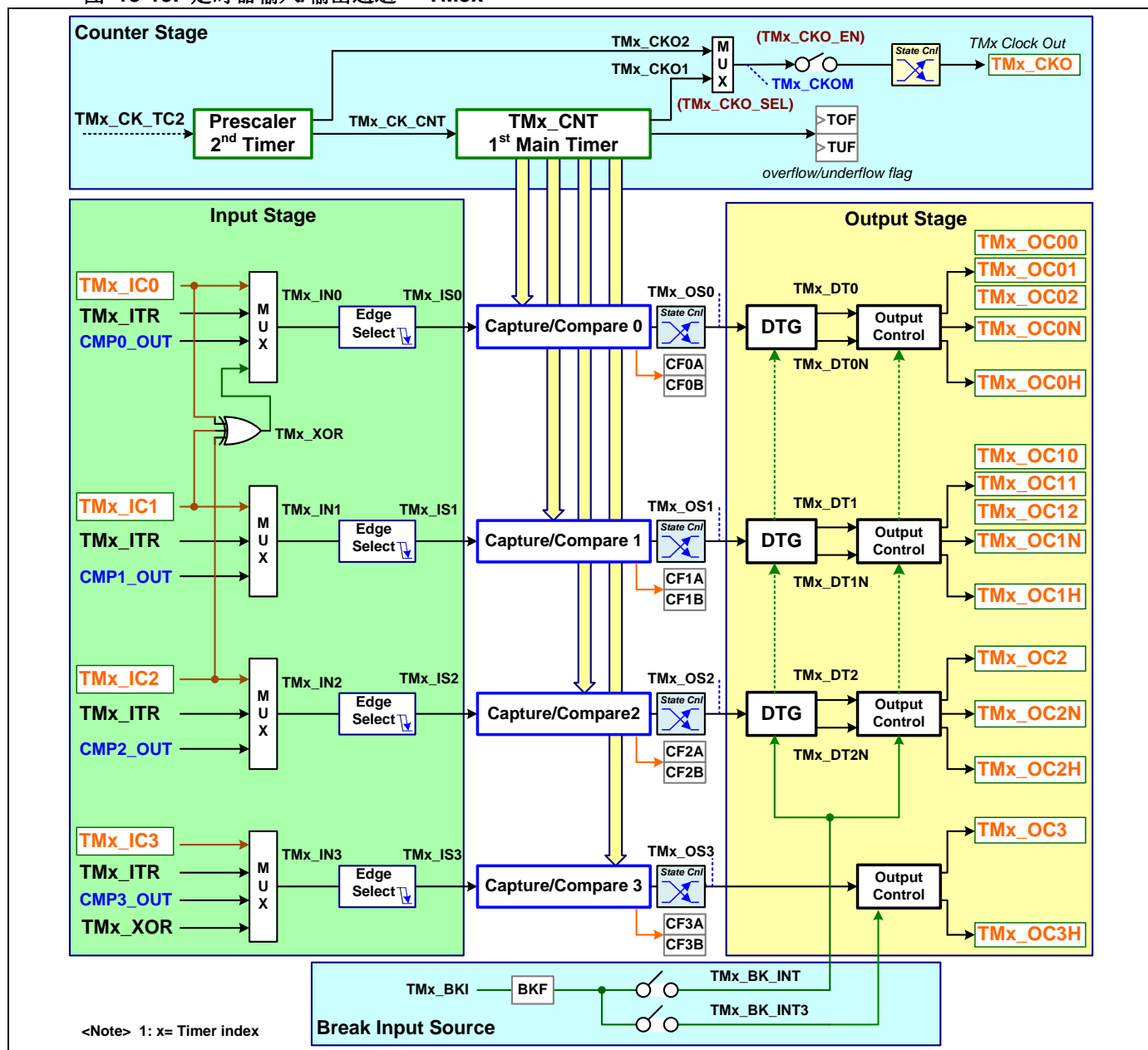
18.10. 定时器输入/输出通道

由于 TM0x 和 TM1x 模块不支持输入捕获 (IC) 和输出比较 (OC) 功能, 因此它们没有通道输入选择功能。TM2x 模块有 2 个 IC/OC/PWM 通道, TM36 模块有 4 个 IC/OC/PWM 通道。

18.10.1. TM3x 定时器输入/输出通道块

TM3x 模块有 4 个 IC/OC/PWM 通道。下图展示了定时器 TM3x 的输入/输出通道。

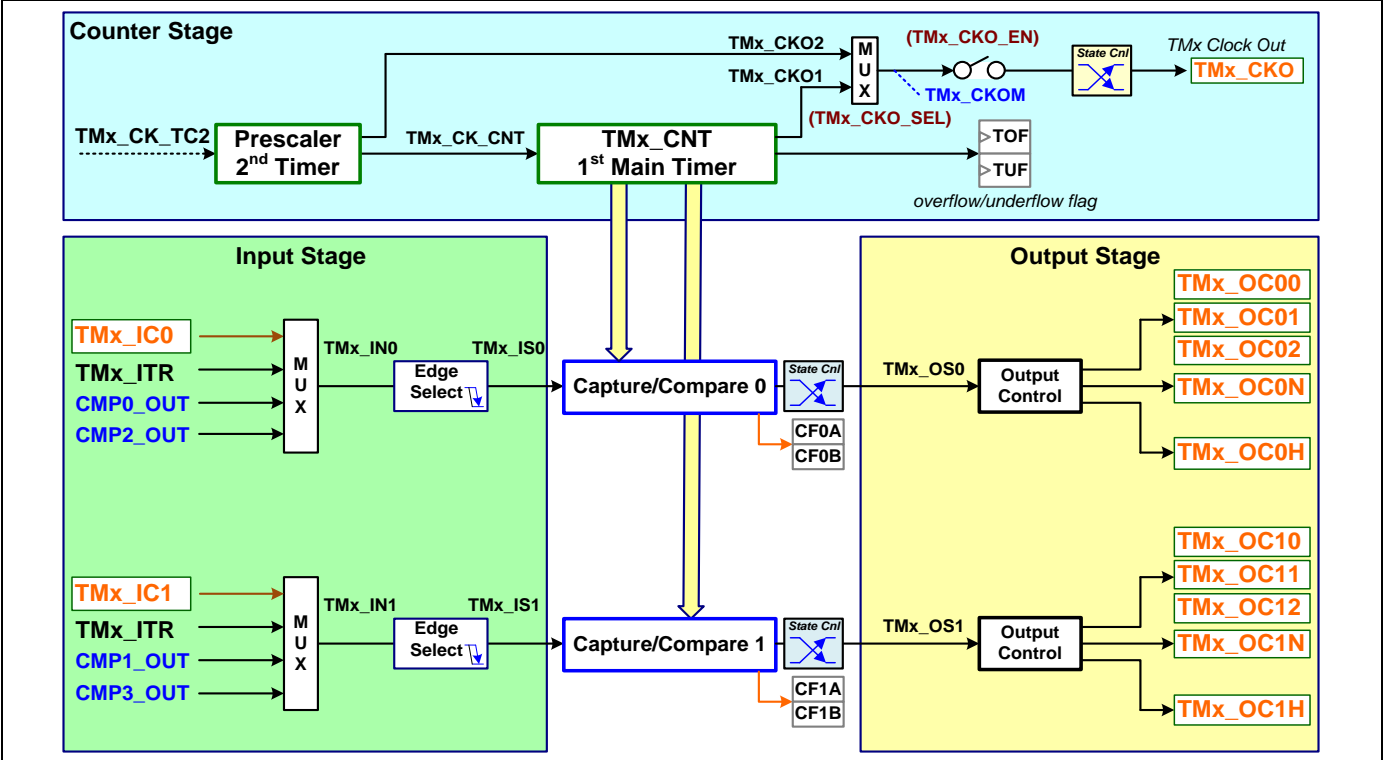
图 18-15. 定时器输入/输出通道 ~ TM3x



18.10.2. TM2x 定时器输入/输出通道块

TM2x 模块有 2 个 IC/OC/PWM 通道。下面的图表展示了定时器 TM2x 的输入输出通道块。

图 18-16. 定时器输入/输出通道 ~ TM2x



18.10.3. 定时器输入通道控制

输入通道信号源可通过 **TMx_ICn_MUX** 寄存器选择(x = 模块标号, n= 通道标号)。

每个通道都有 4 个输入线。**TMx_ICn** 信号直接从引脚输入, **TMx_ITR** 信号从内部“触发/时钟控制块”输入。用户可通过 **TMx_ICn_TRGS** 寄存器选择输入信号触发沿用于输入捕获功能。

通道 0 和通道 1 有 5 个输出线: **TMx_OCn0**, **TMx_OCn1**, **TMx_OCn2**, **TMx_OCnN**, **TMx_OCnH**; 通道 2 有 3 个输出线: **TMx_OC2**, **TMx_OC2N**, **TMx_OC2H**; 通道 3 有 2 个输出线: **TMx_OC3**, **TMx_OC3H**。

TMx_OCnN 信号是 **TMx_OCn** 信号的互补信号。**TMx_OCnH** 信号是若通道通过 **TMx_CCn_MDS** 寄存器设置为双 8 位比较或 PWM 模式时的比较-H 输出信号。

下面的表格展示了每个定时器模块的通道输入信号。

[注释]: TM20/TM26 不支持于 MG32F02A032。

表 18-6. 定时器通道输入信号表

定时器模块		TM00	TM01	TM10	TM16	TM20	TM26	TM36
IN0 信号	IC00					TM20_IC0	TM26_IC0	TM36_IC0
	IC01					TM20_ITR	TM26_ITR	TM36_ITR
	IC02	X	X	X	X	CMP0_OUT	CMP0_OUT	CMP0_OUT
	IC03					CMP2_OUT	CMP2_OUT	TM36_XOR
IN1 信号	IC10					TM20_IC1	TM26_IC1	TM36_IC1
	IC11					TM20_ITR	TM26_ITR	TM36_ITR
	IC12	X	X	X	X	CMP1_OUT	CMP1_OUT	CMP1_OUT
	IC13					CMP3_OUT	CMP3_OUT	Reserved
IN2 信号	IC20	X	X	X	X	X	X	TM36_IC2
	IC21							TM36_ITR

	IC22							CMP2_OUT
	IC23							Reserved
IN3 信号	IC30							TM36_IC3
	IC31							TM36_ITR
	IC32	X	X	X	X	X	X	CMP3_OUT
	IC33							TM36_XOR

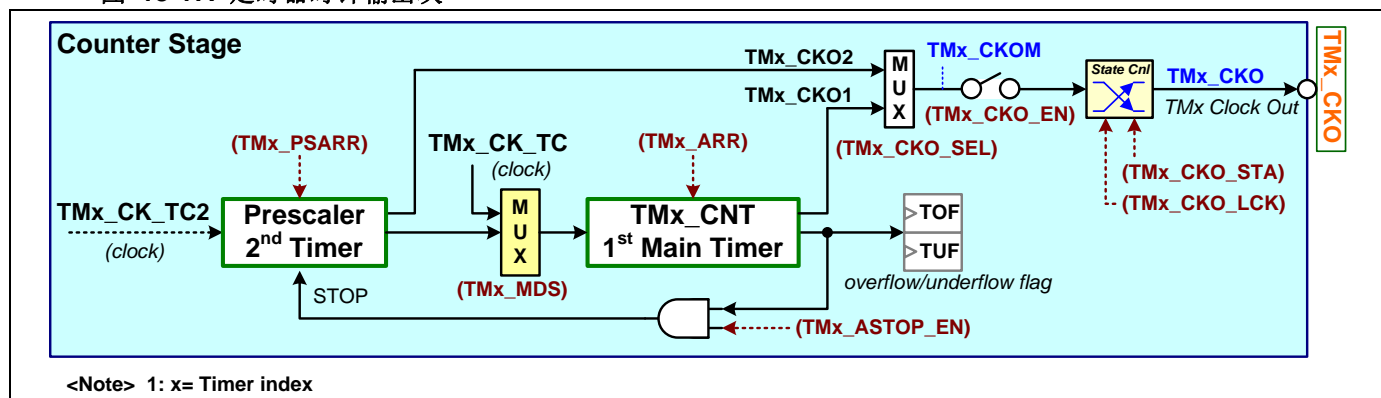
18.10.4. 定时器时钟输出

所有的定时器模块都可通过 **TMx_CKO_EN** 寄存器使能提供 **TMx_CKO** 信号作为时钟输出到外部端口 **TMx_CKO**。时钟输出信号可通过 **TMx_CKO_SEL** 寄存器设置从主定时器或 2nd 定时器的定时器上溢/下溢事件中产生。

用户可通过 **TMx_CKO_STA** 寄存器设置时钟输出信号的初始状态。特别的是，该初始状态会被写保护寄存器位 **TMx_CKO_LCK** 保护，只有在写保护位写 1 时才能被写入。

下面的图表展示了定时器 CKO 时钟输出块。

图 18-17. 定时器时钟输出块



● 主定时器 CKO 输出

用户可通过 **TMx_CKO_SEL** 寄存器选择定时器来自主定时器的时钟输出。当定时器工作模式被设置为分离模式，使用 **TMx_CK_TC** 作为定时器输入时钟。时钟周期是被主定时器的自动重载寄存器 **TMx_ARR** 设置的。当定时器工作模式被设置为全计数模式或级联模式，使用 **TMx_CK_TC2** 作为定时器输入时钟。时钟周期是被 **TMx_ARR** 和 **TMx_PSARR** 设置的。

● 2nd 定时器 CKO 输出

用户可通过 **TMx_CKO_SEL** 寄存器选择定时器来自主定时器的时钟输出。该情况下，使用 **TMx_CK_TC2** 作为定时器输入时钟，定时器工作模式必须设置为分离模式。时钟周期是被 2nd 定时器的自动重载寄存器 **TMx_PSARR** 设置的。

● 时钟输出自动停止模式

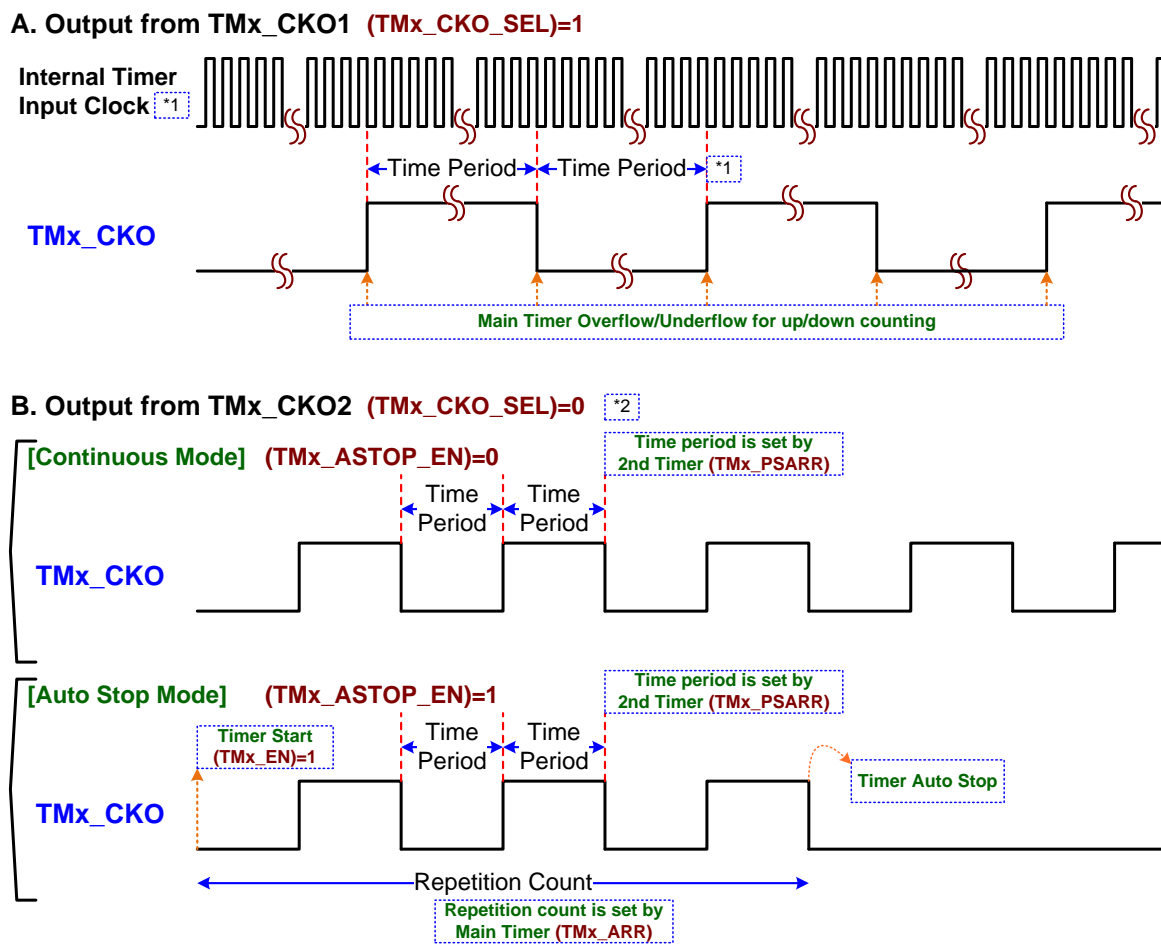
当选择从 2nd 定时器输出时钟时，定时器模块支持定时器输出自动停止模式。**TMx_ASTOP_EN** 寄存器用于使能定时器时钟输出自动停止模式。当使能时，主定时器会作为一个重复定时器。2nd 定时器会在主定时器计数上溢/下溢后自动停止并停止 **TMx_CKO** 信号。

用户可通过 **TMx_ACLEAR_EN** 寄存器使能强制定时器上溢或下溢标志自动清除用于时钟输出自动停止模式。若 **TMx_ASTOP_EN** 被禁用，该位是无效的。当被使能时，定时器会在定时器计数上溢或下溢时自动清除 **TMx_TOF** 或 **TMx_TUF** 标志。对于定时器被外部信号触发启动的应用，时钟输出会进入自动停止模式用于下一次定时器触发启动。

[注释]: **TMx_ACLEAR_EN** 不支持于 MG32F02A132/072。

下面的图表展示了定时器 CKO 时钟输出时序。

图 18-18. 定时器 CKO 时钟输出时序



- <Note-1> (1) When select [Separate Mode] , use **TMx_CK_TC** as timer input clock. The time period is set by Main Timer (**TMx_ARR**).
- (2)When select [Full-Counter Mode] or [Cascade Mode] , use **TMx_CK_TC2** as timer input clock. The time period is set by both (**TMx_ARR**) and (**TMx_PSARR**).
- <Note-2> Use **TMx_CK_TC2** as timer input clock. The time period set by 2nd Timer (**TMx_PSARR**).

18.11. 定时器捕获和比较块

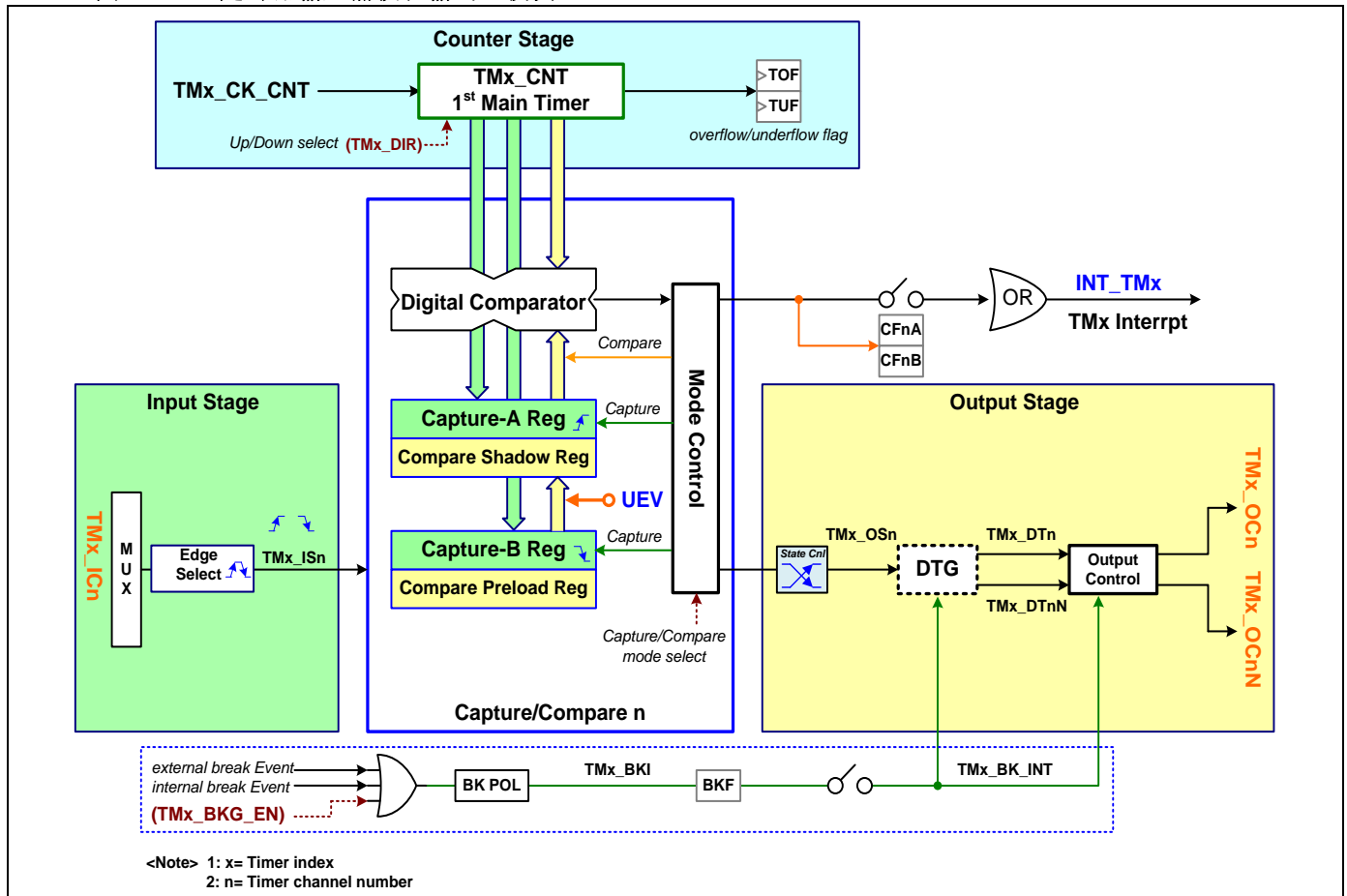
只有 TM2x 和 TM3x 模块支持输入捕获(IC)和输出比较(OC)功能。TM0x 和 TM1x 不支持。

18.11.1. 定时器通道模式

用户可通过 **TMx_CCn_MDS** 寄存器设置独立地设置定时器 IC/OC/PWM 的通道作为输入捕获、输出比较或 PWM 模式。

下面的图表展示了输入捕获和输出比较块。只有 TM36 模块支持 DTG（死区发生器）和中止控制。

图 18-19. 定时器输入捕获和输出比较块



18.11.2. 软件输入捕获和输出比较产生

该模块支持通过软件控制产生 IC/OC/PWM 事件。每个定时器通道有两个软件使能位 **TMx_CCnA_SEN** 和 **TMx_CCnB_SEN** 用于产生 IC/OC/PWM 事件。

对于输入捕获模式，**TMx_CCnA_SEN** 寄存器用于触发产生上升沿捕获事件；**TMx_CCnB_SEN** 寄存器用于触发产生下降沿捕获事件。对于输出比较或 PWM 模式，**TMx_CCnA_SEN** 寄存器仅用于置起 **TMx_CFnA** 标志；**TMx_CCnB_SEN** 寄存器仅用于置起 **TMx_CFnB** 标志。当用户设置软件控制位 **TMx_CCnA_SEN** 或 **TMx_CCnB_SEN**，它会自动在 IC 或 OC 事件发生后被硬件清除。

18.11.3. 定时器捕获和比较寄存器控制

对于每个输入/输出通道，有 **TMx_CC0A** 和 **TMx_CC0B** 两个定时器捕捉和比较寄存器。这些寄存器用于存储输入捕获模式的定时器捕获值或输出比较模式或 PWM 模式的定时器比较阈值。

下面的表格展示了定时器捕获和比较功能的寄存器控制功能。参照“[定时器输入捕获](#)”和“[定时器输出比较和 PWM](#)”以获取更多信息。

表 18-7. 定时器捕获和比较寄存器控制

捕获/比较		输入捕获				输出比较/PWM	
工作模式		级联/分离模式		全计数模式		级联/分离/全计数模式	
功能		双沿捕获	单沿捕获	双沿捕获	单沿捕获	单 16 位比较输出	双 8 位比较输出
TMx 寄存器							
CCnA	H 字节	上升沿捕获 16 位数据	第一捕获 16 位数据	上升和下降沿捕获 32 位数据	上升或下降沿捕获 32 位数据	比较阴影寄存器	比较-H 路径阴影寄存器
	L-字节						比较-L 路径阴影寄存器
CCnB	H 字节	下降沿捕获 16 位数据	第二捕获 16 位数据			比较预载寄存器	比较-H 路径预载寄存器
	L-字节						比较-L 路径预载寄存器

注释-1 CCnA, CCnB：定时器捕获和比较寄存器，n= {0,1,2,3}

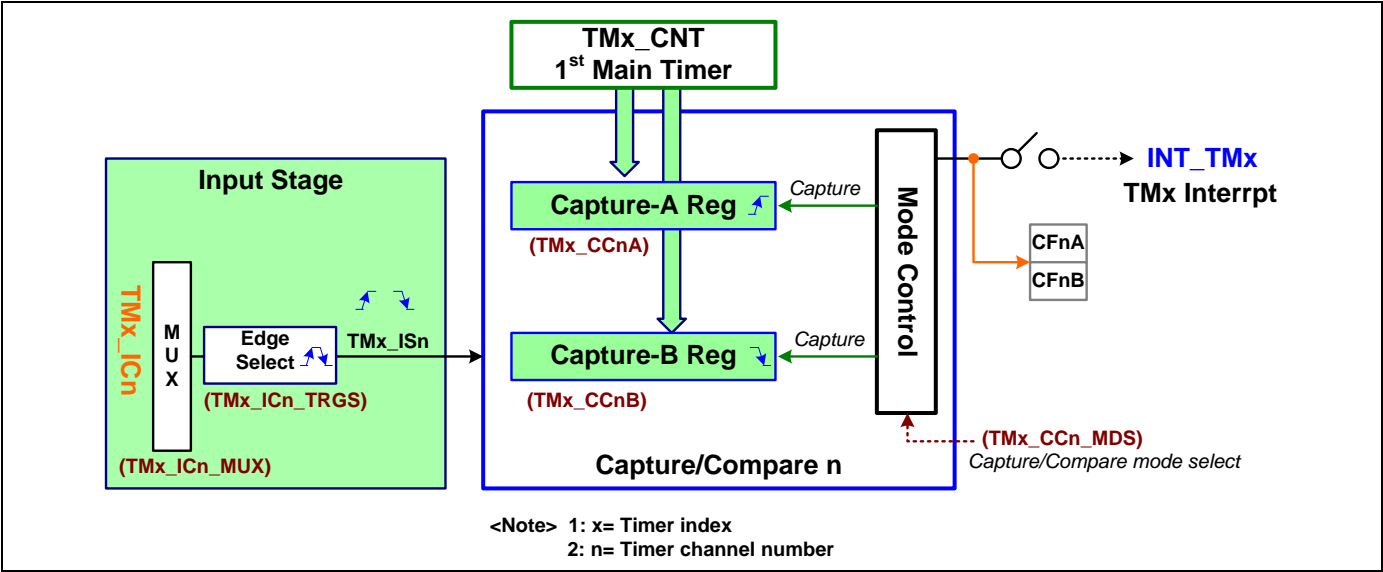
18.12. 定时器输入捕获

18.12.1. 捕获数据和边沿选择

若通道被设置为输入捕获模式，TMx_CCnA 和 TMx_CCnB 寄存器用于当输入捕获信号触发事件发生时捕获计数值。

下面的图表展示了输入捕获块。

图 18-20. 定时器输入捕获块



寄存器 TMx_ICn_TRGS 用于选择输入捕获信号的触发沿。

● 单沿

当在 TMx_ICn_TRGS 寄存器中选择单沿时，在定时器分离和级联模式中，TMx_CCnA 寄存器用于存储第一次捕获的数据，TMx_CCnB 寄存器用于存储第二次捕获的数据。对于全计数模式，TMx_CCnA 寄存器用于存储 Msb 16 位捕获数据，TMx_CCnB 寄存器用于存储 Lsb 16 位捕获数据。

TMx_CFnA 和 TMx_CFnB 位用作输入捕获事件主标志和子标志。

● 双沿

当在 TMx_ICn_TRGS 寄存器中选择双沿时，TMx_CCnA 寄存器用于存储上升沿捕获的数据，TMx_CCnB 寄存器用于存储下降沿捕获的数据。

TMx_CFnA 和 TMx_CFnB 位用作输入捕获上升沿事件标志和下降沿标志。

18.12.2. 捕获数据溢出

对于输入捕获模式，在单沿模式下，若 **TMx_CCnA** 和 **TMx_CCnB** 寄存器都有捕获数据且还未被固件移出，而下一个捕获事件发生时，捕获数据缓冲会溢出；在双沿模式下，若 **TMx_CCnA** 或 **TMx_CCnB** 寄存器有捕获数据且还未被固件移出，而下一个上升或下降捕获事件发生时，也会溢出。用户可通过 **TMx_OVRn_MDS** 寄存器为每个定时器通道选择捕获数据缓冲溢出模式。当选择“覆盖”且捕获数据缓冲溢出，**TMx_CCnA** 或 **TMx_CCnB** 寄存器会被新数据更新。当选择“保持”且捕获数据缓冲溢出，**TMx_CCnA** 或 **TMx_CCnB** 寄存器不会被新数据更新，会保存住旧的捕获数据。

18.12.3. 捕获控制和状态

下面的表格展示了定时器输入捕获模式的相关控制寄存器和状态寄存器。

表 18-8. 定时器输入捕获模式 – MG32F02A132/072

模式	设置	边沿选择	设置	捕获寄存器		状态寄存器		捕获缓冲
	MDS		ICn_TRGS	CCnA	CCnB	CFnA	CFnB	
级联	0	上升沿	1	1st CNT	2nd CNT	1 st 捕获	2 nd 捕获	Yes
		下降沿	2	1st CNT	2nd CNT	1 st 捕获	2 nd 捕获	Yes
		双沿	3	上升沿 CNT	下降沿 CNT	上升沿	下降沿	No
分离	1	上升沿	1	1st CNT	2nd CNT	1 st 捕获	2 nd 捕获	Yes
		下降沿	2	1st CNT	2nd CNT	1 st 捕获	2 nd 捕获	Yes
		双沿	3	上升沿 CNT	下降沿 CNT	上升沿	下降沿	No
全计数	2	上升沿	1	CNT	PSCNT	1 st 捕获	2 nd 捕获	No
		下降沿	2	CNT	PSCNT	1 st 捕获	2 nd 捕获	No
		双沿	3	CNT	PSCNT	上升沿	下降沿	No

CNT：主定时器计数值，PSCNT：预分频器/2nd 定时器计数器值

捕获缓冲：CCnA 作为第一个边沿的捕获缓冲；CCnB 作为下一个边沿的捕获缓冲。
硬件会在 CCnA 完成读取后把 CCnB 复制到 CCnA。(CCnA, CCnB: n = 通道标号)

表 18-9. 定时器输入捕获模式 – MG32F02A032

模式	设置	边沿选择	设置	捕获寄存器		状态寄存器		捕获缓冲
	MDS		ICn_TRGS	CCnA	CCnB	CFnA	CFnB	
级联	0	上升沿	1	1st CNT	2nd CNT	1 st 捕获	2 nd 捕获	Yes
		下降沿	2	1st CNT	2nd CNT	1 st 捕获	2 nd 捕获	Yes
		双沿	3	上升沿 CNT	下降沿 CNT	上升沿	下降沿	No
分离	1	上升沿	1	1st CNT	2nd CNT	1 st 捕获	2 nd 捕获	Yes
		下降沿	2	1st CNT	2nd CNT	1 st 捕获	2 nd 捕获	Yes
		双沿	3	上升沿 CNT	下降沿 CNT	上升沿	下降沿	No
全计数	2	上升沿	1	CNT	PSCNT	1 st 捕获	2 nd 捕获	No
		下降沿	2	CNT	PSCNT	1 st 捕获	2 nd 捕获	No
		双沿	3	CNT	PSCNT	上升沿	下降沿	No

CNT：主定时器计数值，PSCNT：预分频器/2nd 定时器计数器值

捕获缓冲：CCnA 作为第一个边沿的捕获缓冲；CCnB 作为下一个边沿的捕获缓冲。
硬件会在 CCnA 完成读取后把 CCnB 复制到 CCnA。(CCnA, CCnB: n = 通道标号)

- 全计数模式

主定时器和预分频器/ 2^{nd} 定时器计数器可融合成完整定时器或计数器用于全计数模式。

下面的图表展示了全计数模式下，向上计数，**TMx_ARR = 10** 时的定时器输入捕获时序。

图 18-21. 全计数模式定时器输入捕获时序 – MG32F02A132/072

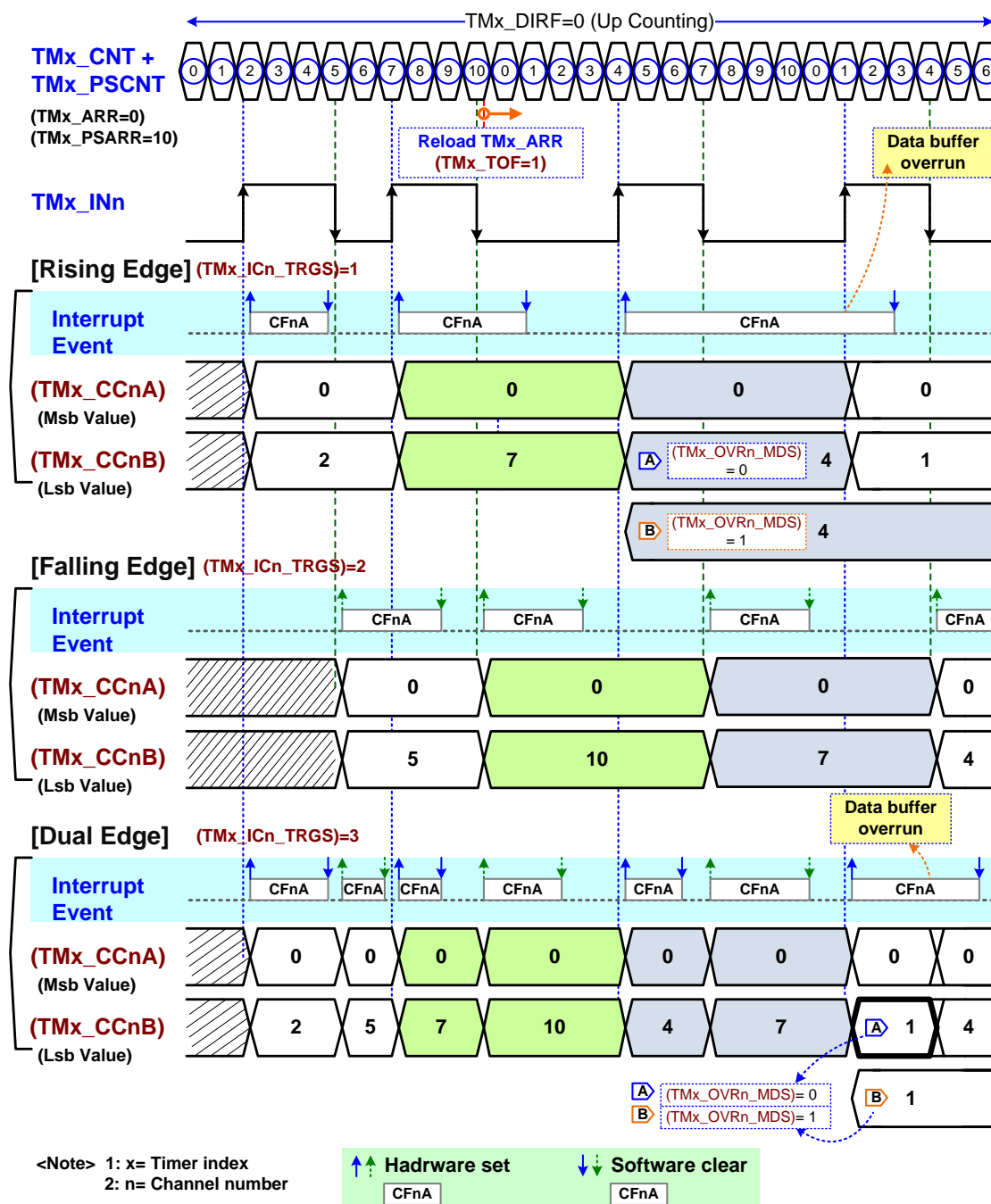
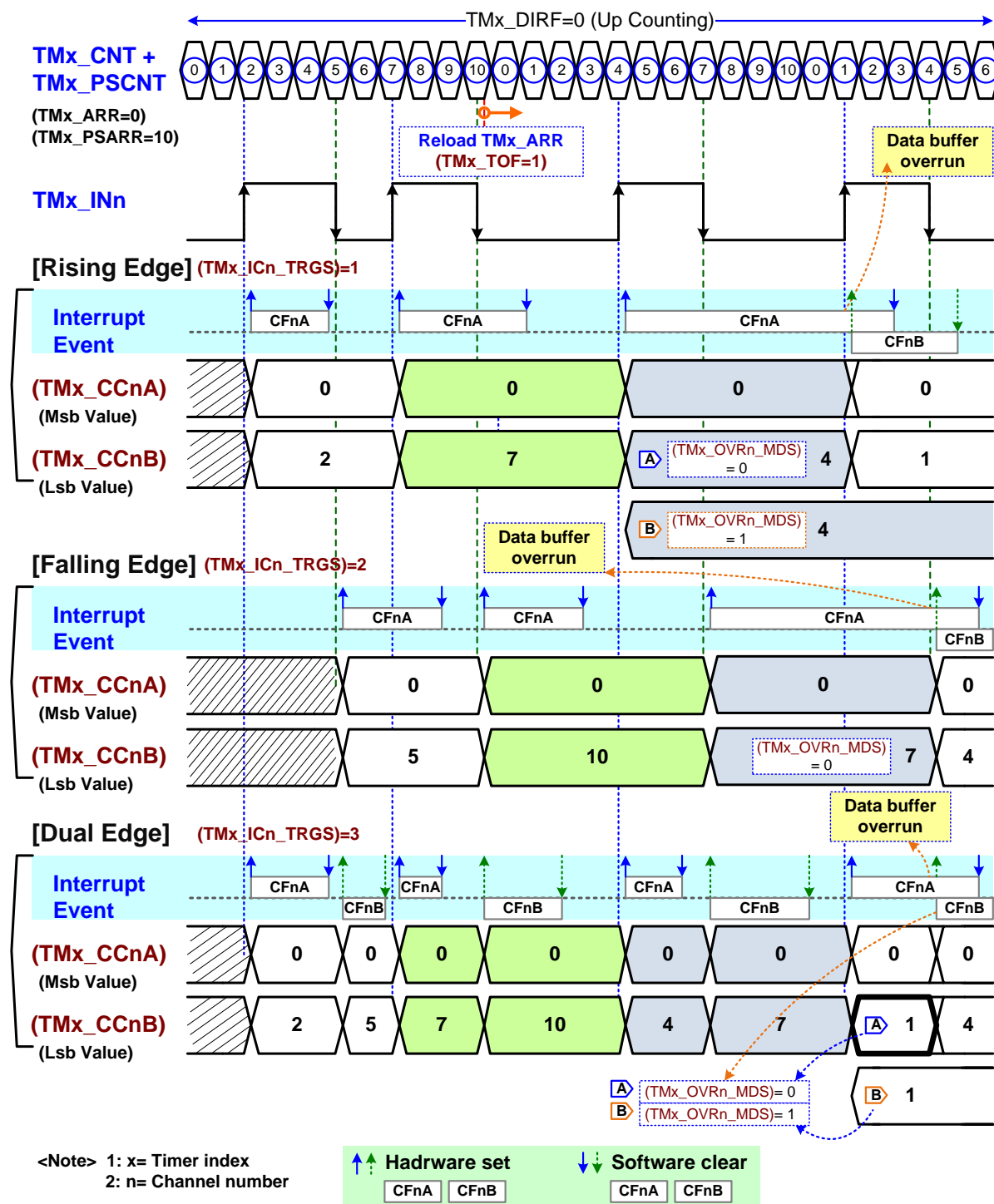


图 18-22. 全计数模式定时器输入捕获时序 - MG32F02A032

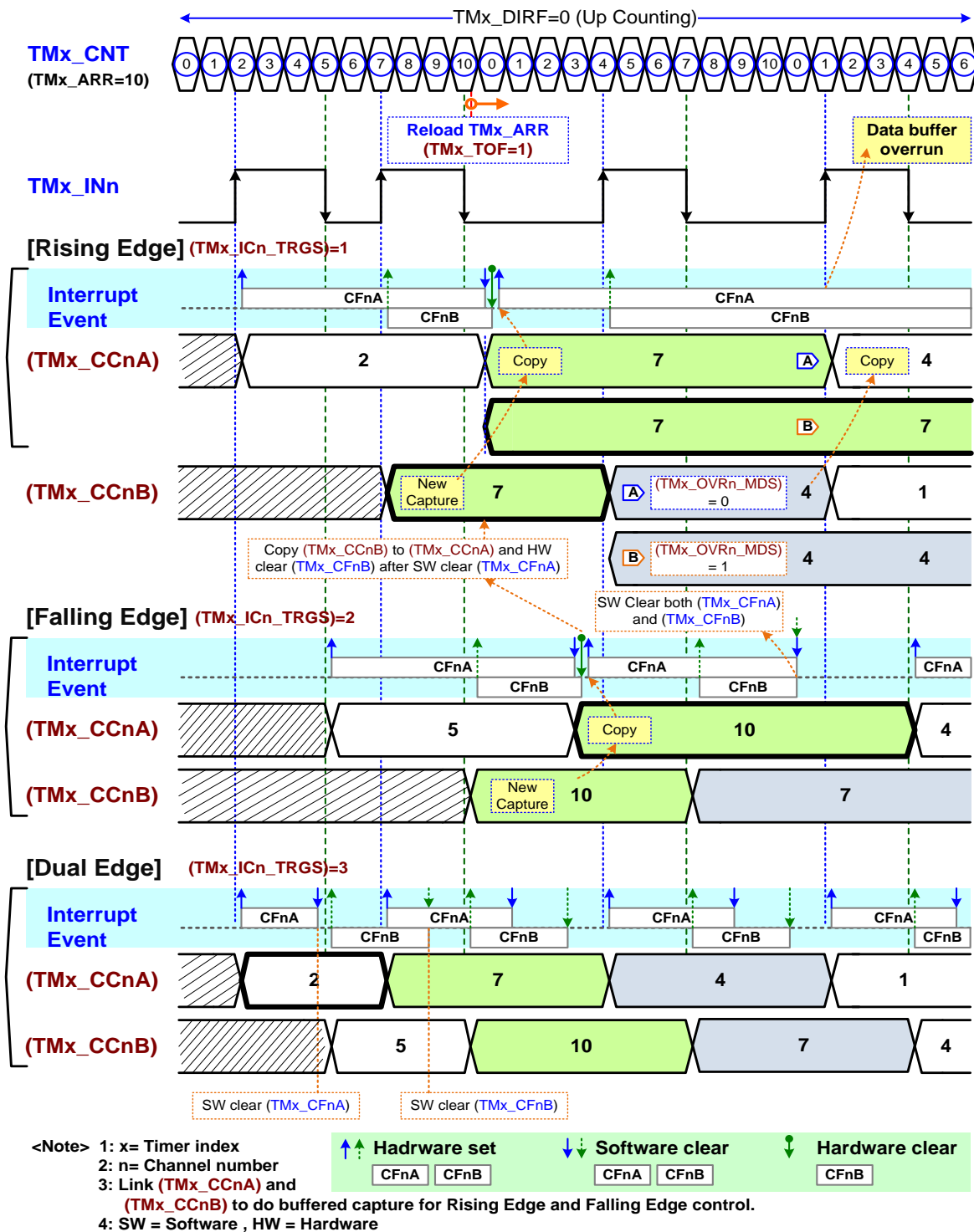


- 级联和分离模式

该模块包含 1 个有预分频器的主定时器用于级联模式；主定时器和预分频器作为独立定时器 1st 定时器和 2nd 定时器用于定时器分离模式。

下面的图表展示了级联和分离模式下，向上计数，**TMx_ARR = 10** 时定时器输入捕获时序。

图 18-23. 级联和分离模式定时器输入捕获时序



18.13. 定时器输出比较和 PWM

18.13.1. 比较重载寄存器

若通道被设置为输出比较/PWM 模式，**TMx_CCnA** 寄存器会被用作比较预载寄存器用于软件设置；**TMx_CCnB** 则用作定时器输出比较的比较阴影寄存器。**TMx_CCnB** 寄存器的值会在被写时复制到 **TMx_CCnA** 寄存器中。

在应用中，有 1 个定时器输出比较重载功能锁定使能位 **TMx_OC_LCK** 用于所有的通道。当该位使能且定时器更新事件发生，会锁定比较预载寄存器 **TMx_CCnB** 重载到比较阴影缓冲寄存器 **TMx_CCnA** 中。直到该位被禁用，比较预载寄存器的值会在下次定时器更新时间发生时更新比较阴影缓冲寄存器。

当通道被设置为双 8 位比较/PWM 模式，**TMx_CCnA** 寄存器会被分成低 8 位比较阴影寄存器用于比较-L，高 8 位比较阴影寄存器用于比较-H；**TMx_CCnB** 寄存器会被分成低 8 位比较预载寄存器用于比较-L，高 8 位比较预载寄存器用于比较-H。

当通道选择 PWM 模式，寄存器 **TMx_PWM_MDS** 用于选择 PWM 左右或中心对齐模式。

在中心对齐模式下，当 **TMx_CCnA** 和 **TMx_CCnB** 的值等于 **TMx_ARR** 或 0x0000 时，输出高和低电平的时间宽度为 0x10000 时钟宽度。

18.13.2. 比较输出状态

TMx 定时器模块可以通过设置 **TMx_OSn_STA** 寄存器设置定时器比较输出 **OSn** 的初始状态以进行 16 位比较/PWM 模式。此外，用户还可以通过设置 **TMx_OSn_STA** 寄存器设置定时器比较-L 输出 **OSn** 的初始状态，通过设置 **TMx_OSnH_STA** 寄存器来设置定时器比较-H 输出 **OSnH** 的初始状态用于双 8 位比较/PWM 模式。

这些初始状态寄存器有独立的写保护寄存器位 **TMx_OSn_LCK** 或 **TMx_OSnH_LCK**。它们只有在相关写保护位被同时写 1 时可被写入。

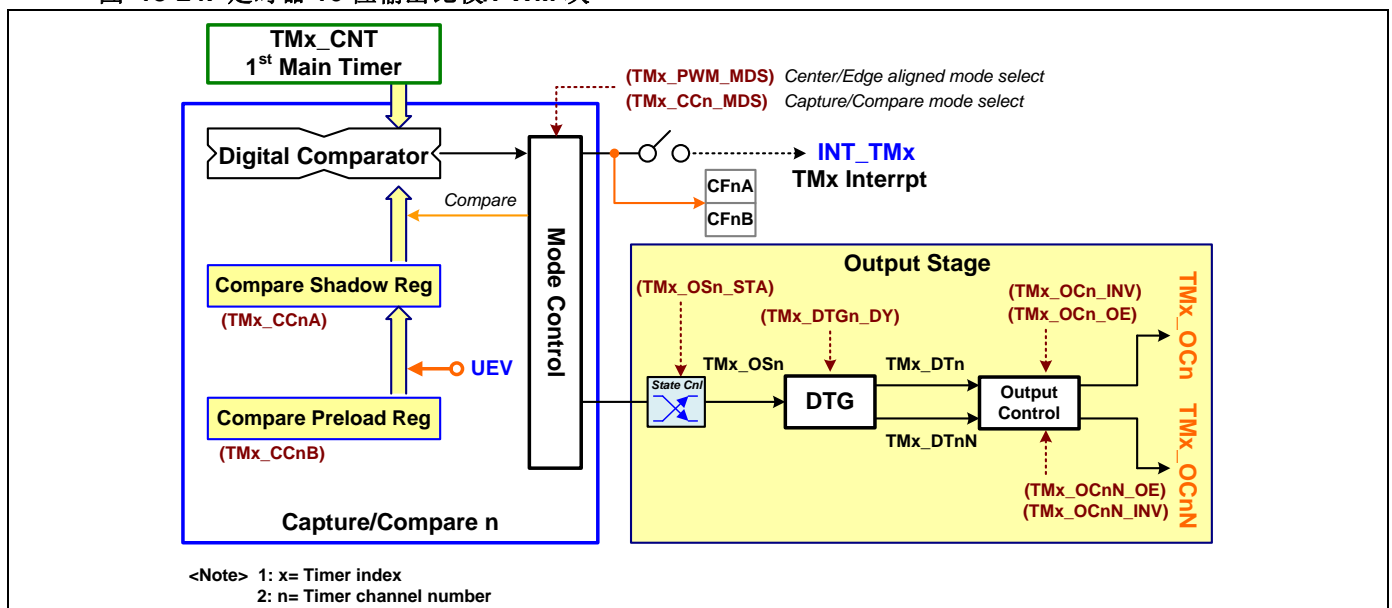
18.13.3. 16 位比较/PWM 模式

TMx_CFnA 位用作输出比较事件标志用于 16 位比较器模式；在中心对齐 PWM 模式用作向上计数 PWM 比较标志。

TMx_CFnB 位不用于 16 位比较器模式；但在中心对齐 PWM 模式用作向下计数 PWM 比较标志。

下面的图表展示了 16 位比较/PWM 输出块。

图 18-24. 定时器 16 位输出比较/PWM 块

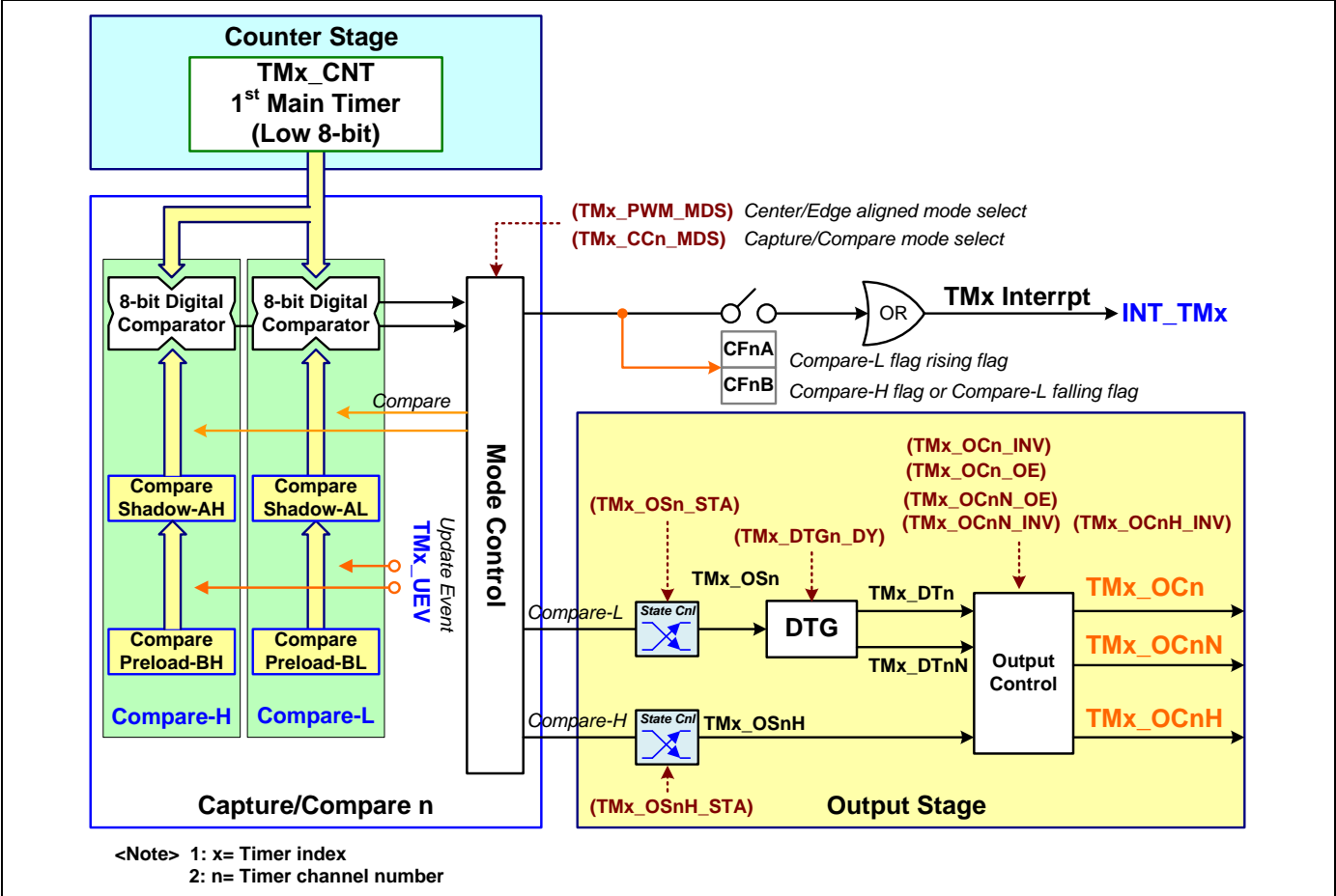


18.13.4. 双 8 位比较/PWM 模式

当比较-L 为 PWM 模式且为中心对齐模式，**TMx_CFnA** 位用作向上计数 PWM 比较-L 事件标志；**TMx_CFnB** 位用作向下计数 PWM 比较-H 事件标志。其他情况下，在 8 位比较器模式下，**TMx_CFnA** 位用作输出比较-L 事件标志；**TMx_CFnB** 位用作比较-H 事件标志。

下面的图表展示了双 8 位比较/PWM 输出块。

图 18-25. 定时器双 8 位比较/PWM 输出块



18.13.5. 输出比较/PWM 控制和状态

下面的表格展示了定时器输出比较/PWM 模式的相关控制寄存器和状态寄存器。

表 18-10. 定时器输出比较/PWM 模式

模式	设置	PWM 模式	设置	比较寄存器				状态寄存器	
	CCn_MDS		PWM_MDS	CCnA High	CCnA Low	CCnB High	CCnB Low	CFnA	CFnB
16 位比较	2	x	x	比较 16 位值		预载比较 16 位值		比较	x
16 位 PWM	4	边沿对齐	0					比较	x
		中心对齐	1					向上计数	向下计数
双 8 位比较	3	x	x	比较-H 8 位值	比较-L 8 位值	预载比较-H 8 位值	预载比较-L 8 位值	比较-L	比较-H
双 8 位 PWM	5	边沿对齐	0					比较-L	比较-H
		中心对齐	1					比较-L 向上计数	比较-L 向下计数

CCnA, CCnB : n =通道标号

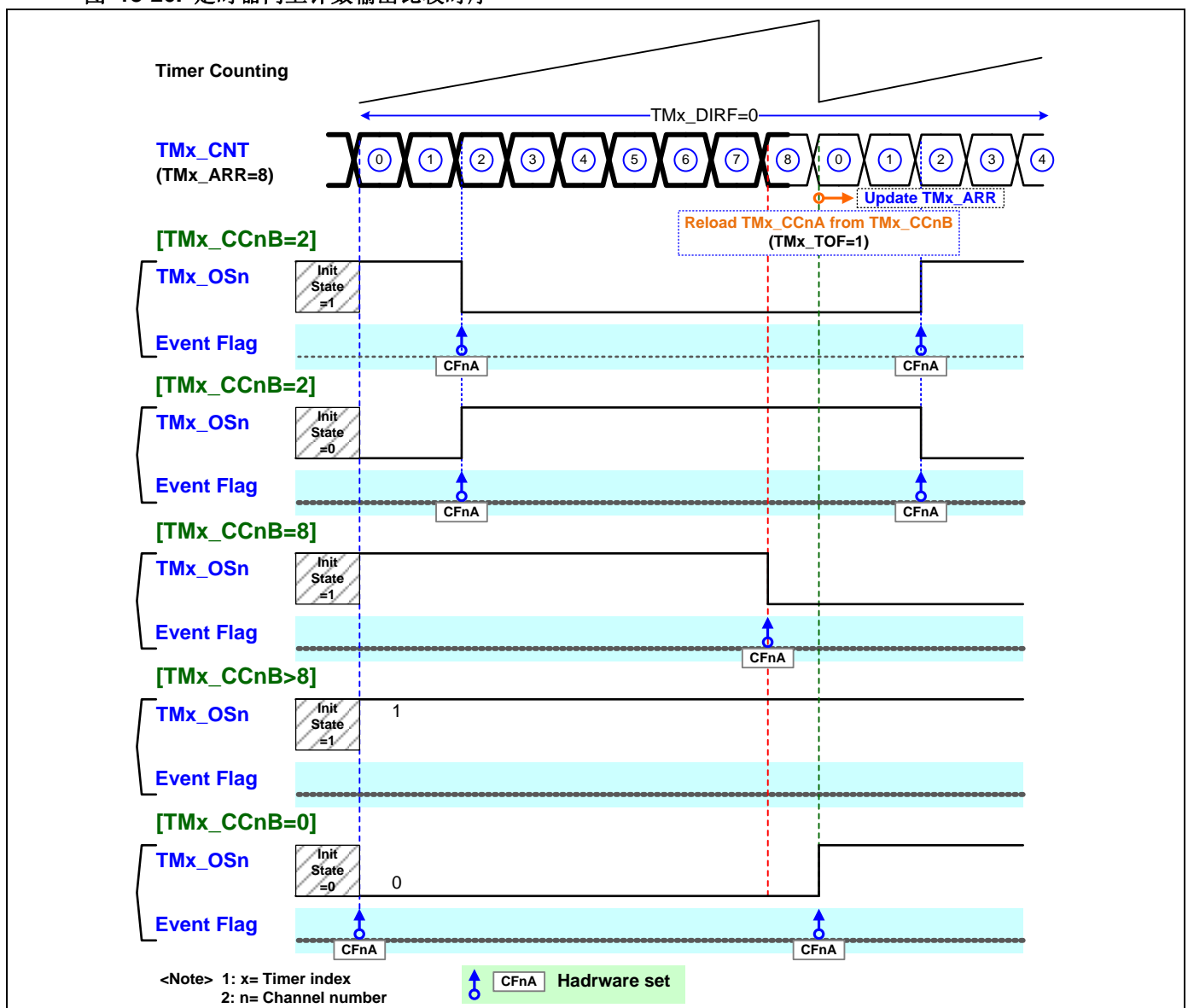
18.13.6. 定时器输出比较和 PWM 时序

❖ 定时器向上计数输出比较时序

下面的图表展示了定时器向上计数，**TMx_ARR** = 8 时的输出比较时序。TOF 标志在 **TMx_CNT** 计数器值从 8 到 0 和定时器上溢时置起。

- 比较阈值 **TMx_CCnB = 2**
TMx_OSn 信号会在 **TMx_CNT** 定时器值从 1 到 2 时切换电平，此外，CFnA 标志会被置起。
- 比较阈值 **TMx_CCnB = 8**
TMx_OSn 信号会在 **TMx_CNT** 定时器值从 7 到 8 时切换电平，此外，CFnA 标志会被置起。
- 比较阈值 **TMx_CCnB > 8**
TMx_OSn 信号会一直保持初始电平，此外，CFnA 标志将不会被置起。
- 比较阈值 **TMx_CCnB = 0**
TMx_OSn 信号会在 **TMx_CNT** 定时器值从 8 到 0 时切换电平，此外，CFnA 标志会被置起。

图 18-26. 定时器向上计数输出比较时序



❖ 定时器向上计数边沿对齐 PWM 时序

下面的图表展示了定时器向上计数， $TMx_ARR = 8$ 时，边沿对齐 PWM 时序。

- 比较阈值 $TMx_CCnB = 2$

TMx_OSn 信号会在 TMx_CNT 定时器值从 1 到 2 时切换电平，此外，CFnA 标志会被置起。然后 TMx_OSn 信号会在定时器上溢时切换回初始电平并置起 TOF 标志。

- 比较阈值 $TMx_CCnB = 8$

TMx_OSn 信号会在 TMx_CNT 定时器值从 7 到 8 时切换电平，此外，CFnA 标志会被置起。然后 TMx_OSn 信号会在定时器上溢时切换回初始电平并置起 TOF 标志。

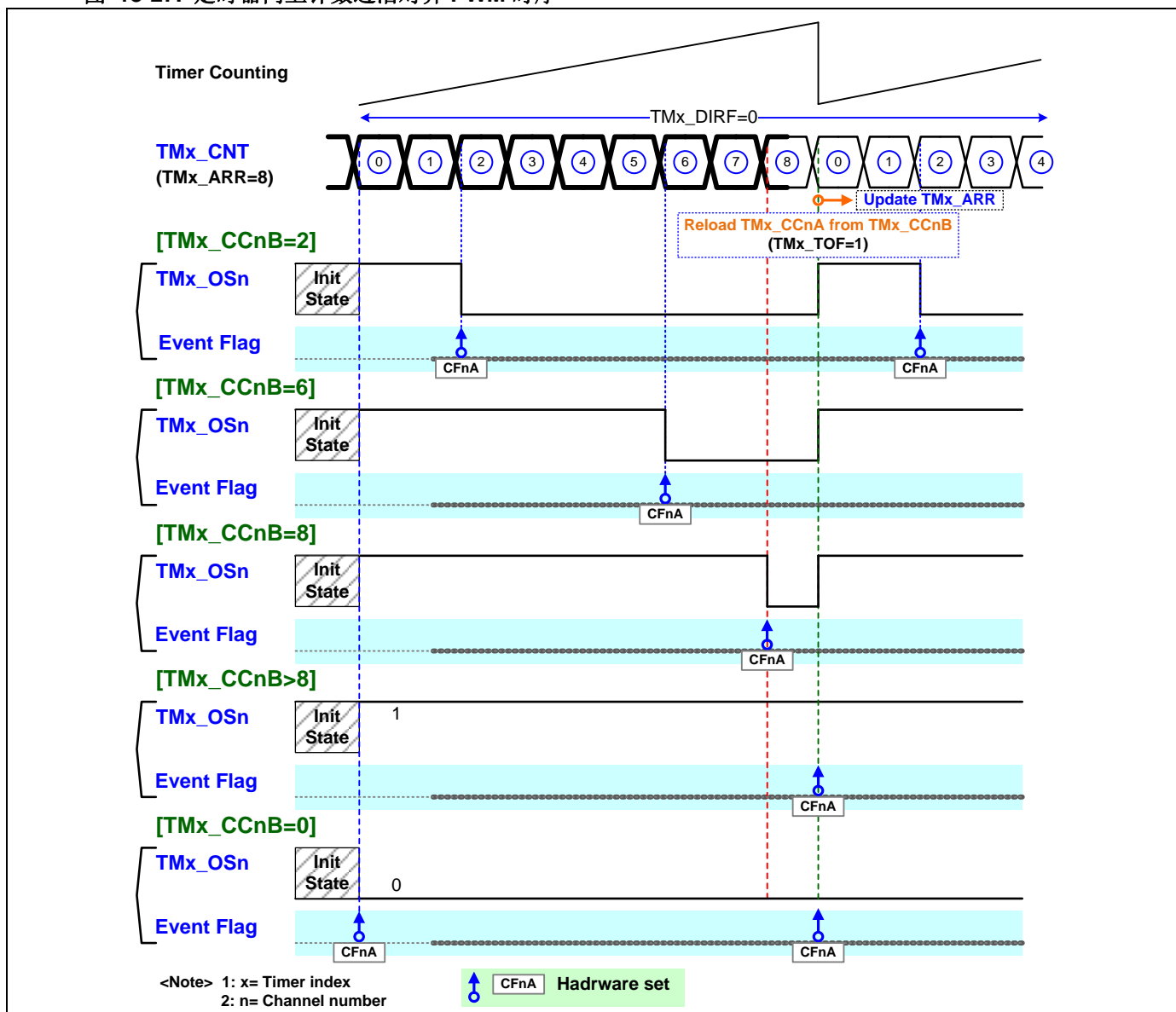
- 比较阈值 $TMx_CCnB > 8$

TMx_OSn 信号会一直保持高电平，CFnA 和 TOF 标志会在 TMx_CNT 计数器值从 8 到 0 时置起。

- 比较阈值 $TMx_CCnB = 0$

TMx_OSn 信号会一直保持低电平，CFnA 和 TOF 标志会在 TMx_CNT 计数器值从 8 到 0 时置起。CFnA 还会在 TMx_CNT 计数器启动后第一次等于 0 时置起。

图 18-27. 定时器向上计数边沿对齐 PWM 时序



❖ 定时器向下计数边沿对齐 PWM 时序

下面的图表展示了定时器向下计数， $TMx_ARR = 8$ 时，边沿对齐 PWM 时序。

- 比较阈值 $TMx_CCnB = 2$

TMx_OSn 信号会在 TMx_CNT 定时器值从 2 到 1 时切换电平，此外， $CFnA$ 标志会被置起。然后 TMx_OSn 信号会在定时器下溢时切换回初始电平并置起 TUF 标志。

- 比较阈值 $TMx_CCnB = 8$

TMx_OSn 信号会在 TMx_CNT 定时器值从 8 到 7 时切换电平，此外， $CFnA$ 标志会被置起。然后 TMx_OSn 信号会在定时器下溢时切换回初始电平并置起 TUF 标志。

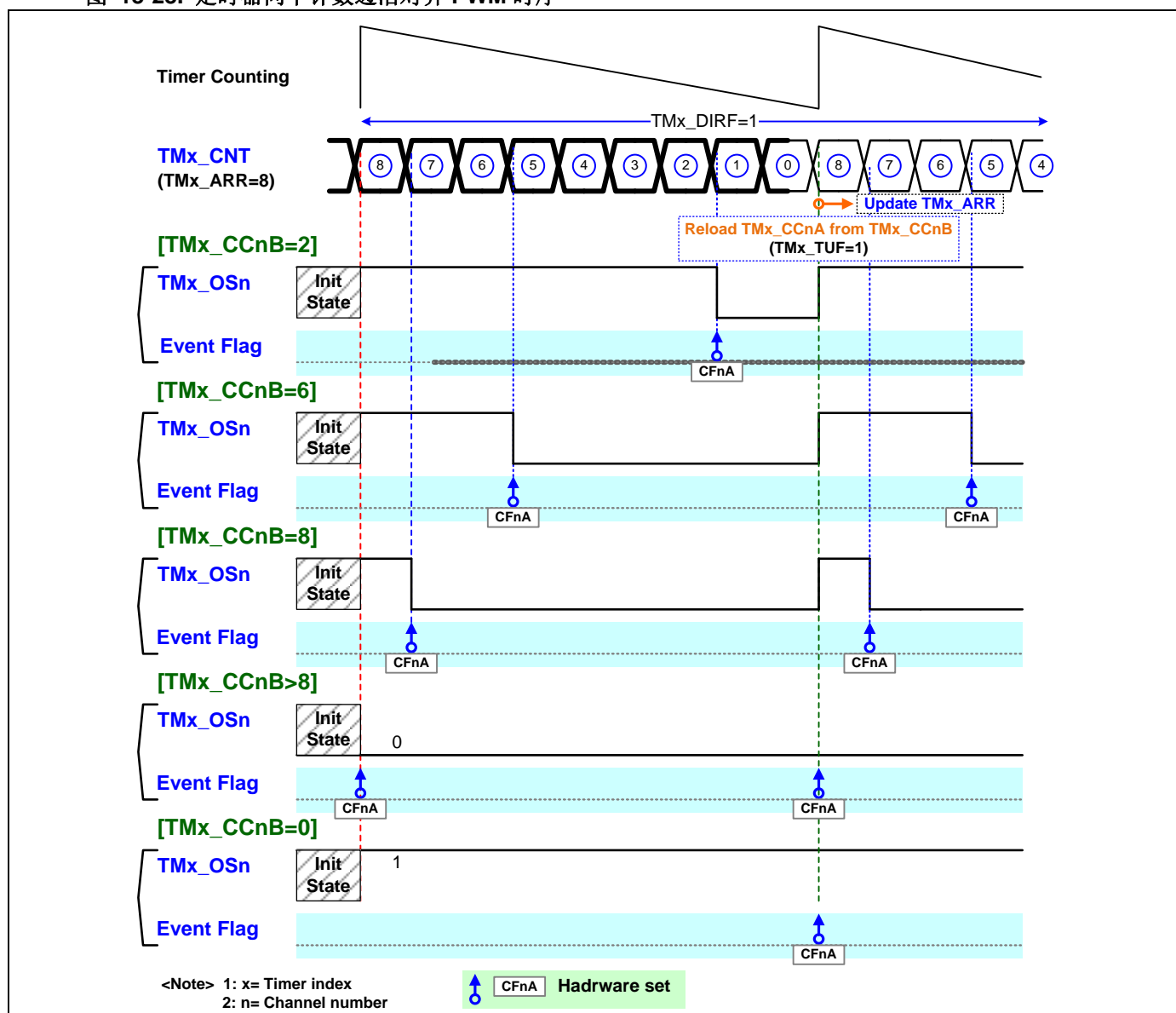
- 比较阈值 $TMx_CCnB > 8$

TMx_OSn 信号会一直保持低电平， $CFnA$ 和 TUF 标志会在 TMx_CNT 计数器值从 0 到 8 时置起。 $CFnA$ 还会在 TMx_CNT 计数器启动后第一次等于 8 时置起。

- 比较阈值 $TMx_CCnB = 0$

TMx_OSn 信号会一直保持高电平， $CFnA$ 和 TUF 标志会在 TMx_CNT 计数器值从 0 到 8 时置起。

图 18-28. 定时器向下计数边沿对齐 PWM 时序



❖ 定时器中心对齐 PWM 时序

下面的图表展示了定时器 $TMx_ARR = 6$ 时，中心对齐 PWM 时序。TOF 标志在 TMx_CNT 计数器值从 5 到 6 和定时器上溢时置起；TUF 标志在 TMx_CNT 计数器值从 1 到 0 和定时器下溢时置起。

● 比较阈值 $TMx_CCnB = 2$

TMx_OSn 信号会在 TMx_CNT 定时器值从 1 到 2 时切换电平，此外，CFnA 标志会被置起。然后 TMx_OSn 信号会在 TMx_CNT 定时器值从 3 到 2 时切换回初始电平并置起 CFnB 标志。

● 比较阈值 $TMx_CCnB = 6$

TMx_OSn 信号会一直保持高电平，CFnB 会在 TMx_CNT 定时器值从 5 到 6 时置起，CFnA 标志将不会被置起。

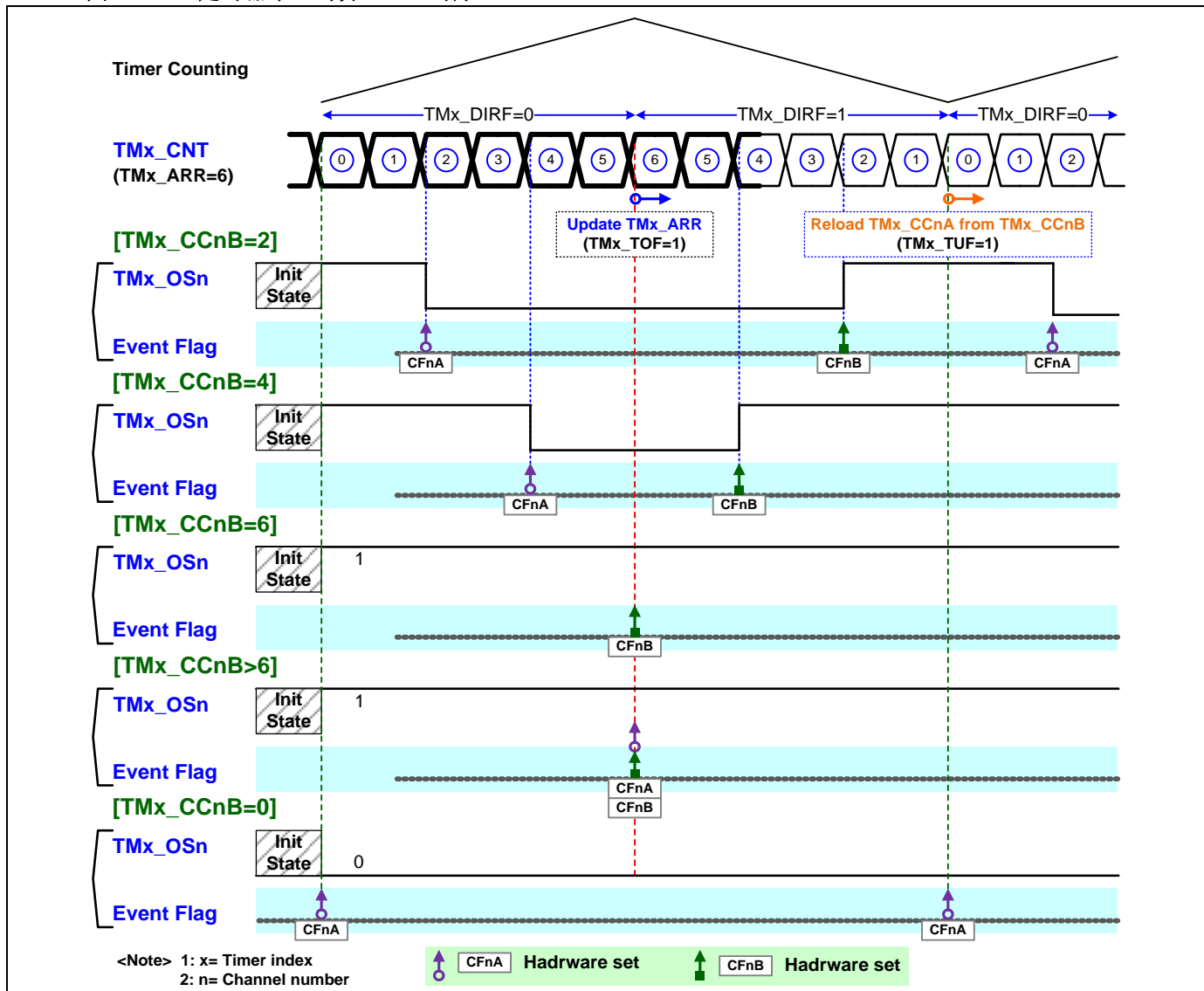
● 比较阈值 $TMx_CCnB > 6$

TMx_OSn 信号会一直保持高电平，CFnA 和 CFnB 会在 TMx_CNT 定时器值从 5 到 6 时置起。

● 比较阈值 $TMx_CCnB = 0$

TMx_OSn 信号会一直保持低电平，CFnA 会在 TMx_CNT 定时器值从 1 到 0 时置起，CFnA 还会在 TMx_CNT 计数器启动后第一次等于 0 时置起。

图 18-29. 定时器中心对齐 PWM 时序



❖ 定时器输出自动停止模式

该定时器模块在比较和 PWM 模式通过 **TMx_ASTOP_EN** 寄存器设置支持定时器输出自动停止模式，当被使能时，该定时器输出会在定时器计数上溢或下溢时停止定时器输出。

当定时器自动停止模式被使能后，用户可通过设置 **TMx_ACLEAR_EN** 寄存器使能在自动停止模式自动清除定时器上溢 **TMx_TOF** 标志或下溢 **TMx_TUF** 标志。对于定时器被外部信号触发的应用，定时器比较和 PWM 输出将会在下一次定时器触发启动时进入自动停止模式。

[注释]: **TMx_ACLEAR_EN** 不支持于 MG32F02A132/072。

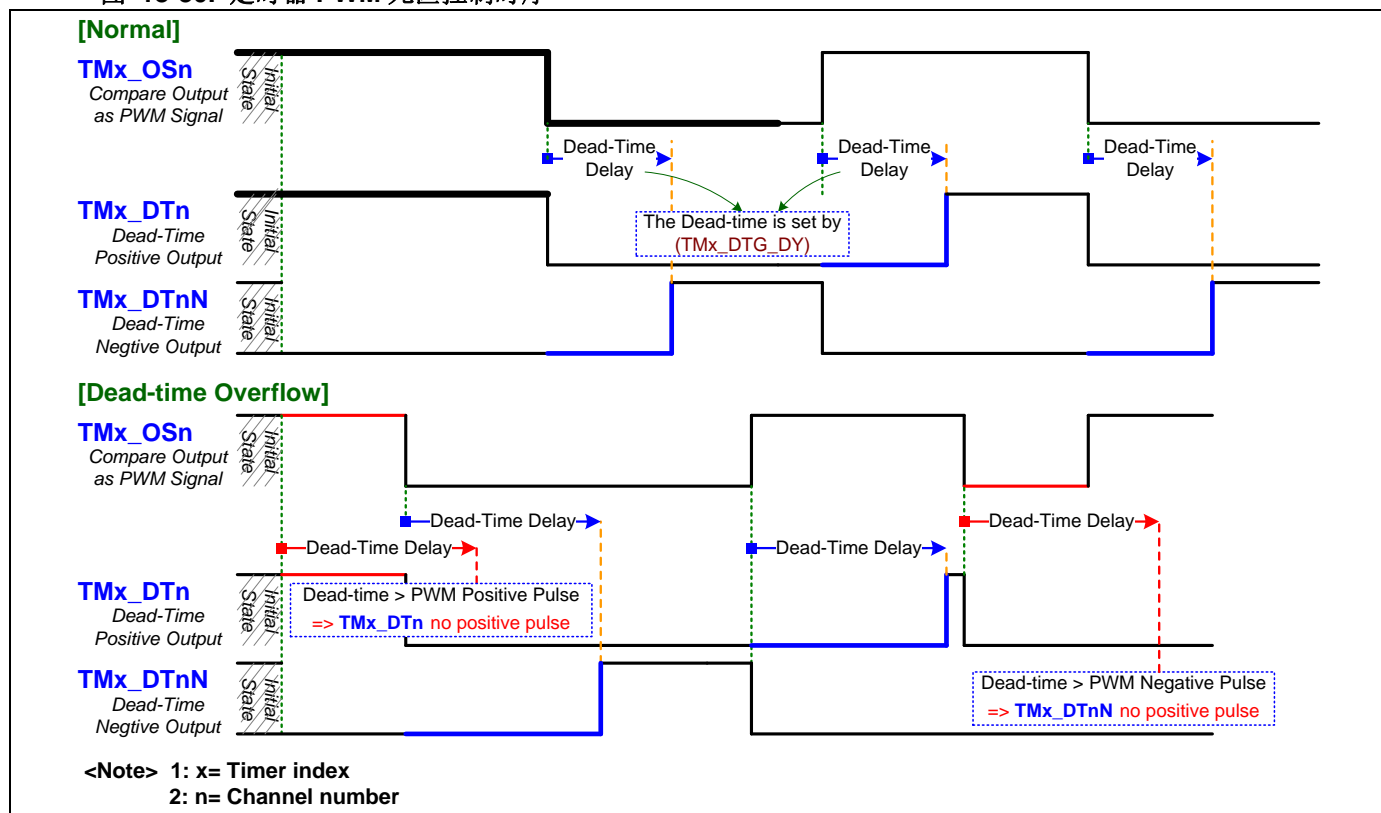
18.13.7. PWM 死区控制

死区发生器 (DTG) 只支持于 TM36。用户可通过 **TMx_CCn_MDS** 寄存器使用 DTG 功能和设置定时器通道为 16 位 PWM 模式或双 8 位 PWM 模式。

寄存器 **TMx_DTG_DY** 用于设置所有通道的 CK_DTG 时基单元内的 **TMx_DTn** 和 **TMx_DTnN** 信号之间的死区延时。**TMx_DTn** 和 **TMx_DTnN** 是来自 DTG 输出的内部信号。

下面的图表展示了定时器 PWM 死区控制时序。

图 18-30. 定时器 PWM 死区控制时序



18.14. 定时器输出控制块

最多有四个输出通道，每个通道有三种类型的 **OCn**, **OCnH** 和 **OCnN** 输出信号。通道-0,1 有 3 个独立输出线 **OCnm**。(n = 输出通道标号, m = 输出线标号{0,1,2})

输出控制块可通过 **TMx_OCn_INV**, **TMx_OCnH_INV** 和 **TMx_OCnN_INV** 寄存器设置反相 **OCn**, **OCnH** 和 **OCnN** 输出信号。用户可以通过设置输出通道 0,1 的 **TMx_OCn_OEm** 独立寄存器和输出通道 2,3 的 **TMx_OCn_OE** 独立寄存器, 将这些 **OCn** 输出设置为固定状态。此外, 此外, 用户还可以通过设置输出通道 0,1,2 的 **TMx_OCnN_OE** 的独立寄存器, 将这些 **OCnN** 输出设置为固定状态。(n = 输出通道标号, m = 输出线标号 {0,1,2})

独立的状态初始寄存器位 **TMx_STPn_STA** 用于 **OCn**, 输出通道-0/1/2/3, **TMx_STPnN_STA** 用于 **OCnN** 输出通道-0/1/2。

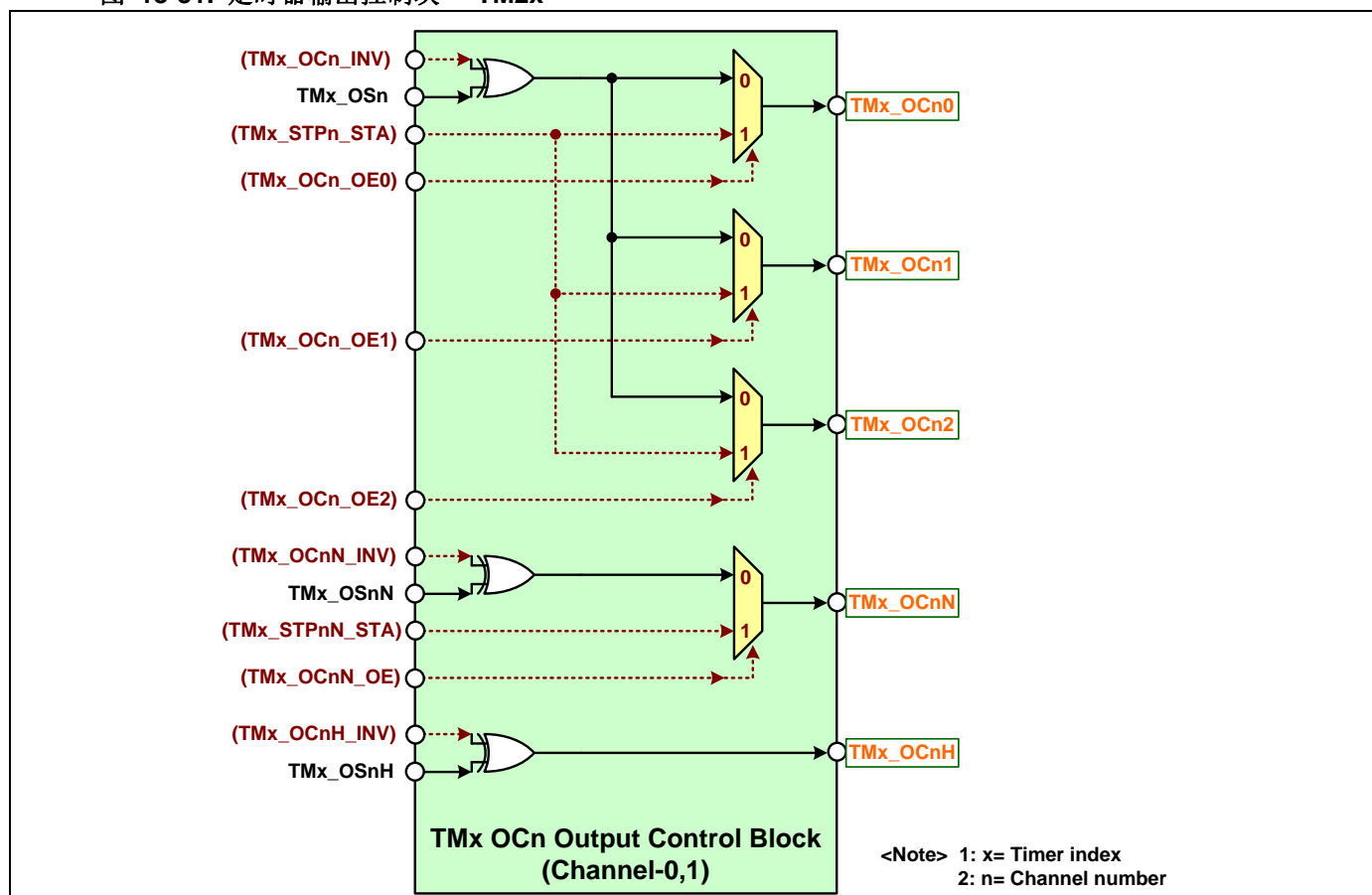
18.14.1. TM2x 定时器输出控制块

通道-0 和通道-1 的输出控制块相同, 一共有 5 个输出线 **TMx_OCn0**, **TMx_OCn1**, **TMx_OCn2**, **TMx_OCnN** 和 **TMx_OCnH**。

TMx_OSn, **TMx_OSnN** 和 **TMx_OSnH** 信号是定时器比较输出。**TMx_OCn0**, **TMx_OCn1** 和 **TMx_OCn2** 输出共用相同通用信号源 **TMx_OSn**。**TMx_OCnN** 和 **TMx_OCnH** 输出有独立的信号源 **TMx_OSnN** 和 **TMx_OSnH**。

下面的图表展示了 TM2x 的定时器输出控制块。

图 18-31. 定时器输出控制块 ~ TM2x



18.14.2. TM3x 定时器输出控制块

通道-0 和通道-1 的输出控制块相同；通道-2 有 3 个输出线 **TMx_OC2**、**TMx_OC2N** 和 **TMx_OC2H**。通道-3 有 2 个输出线 **TMx_OC3** 和 **TMx_OC3H**。

TMx_DTn 和 **TMx_DTnN** 信号是 DTG（死区发生器）的输出；**TMx_OSnH** 信号是定时器比较输出。

对于通道-0 和通道-1，**TMx_OCn0**、**TMx_OCn1** 和 **TMx_OCn2** 输出共用相同通用信号源 **TMx_DTn**。
TMx_OCnN 和 **TMx_OCnH** 输出有自己的信号源 **TMx_DTnN** 和 **TMx_OSnH**。对于通道-2，**TMx_OC2**、**TMx_OC2N** 和 **TMx_OC2H** 输出有自己的信号源 **TMx_DTn**、**TMx_DTnN** 和 **TMx_OSnH**。对于通道-3，**TMx_OC3** 和 **TMx_OC3H** 输出有自己的信号源 **TMx_DTn** 和 **TMx_OSnH**。

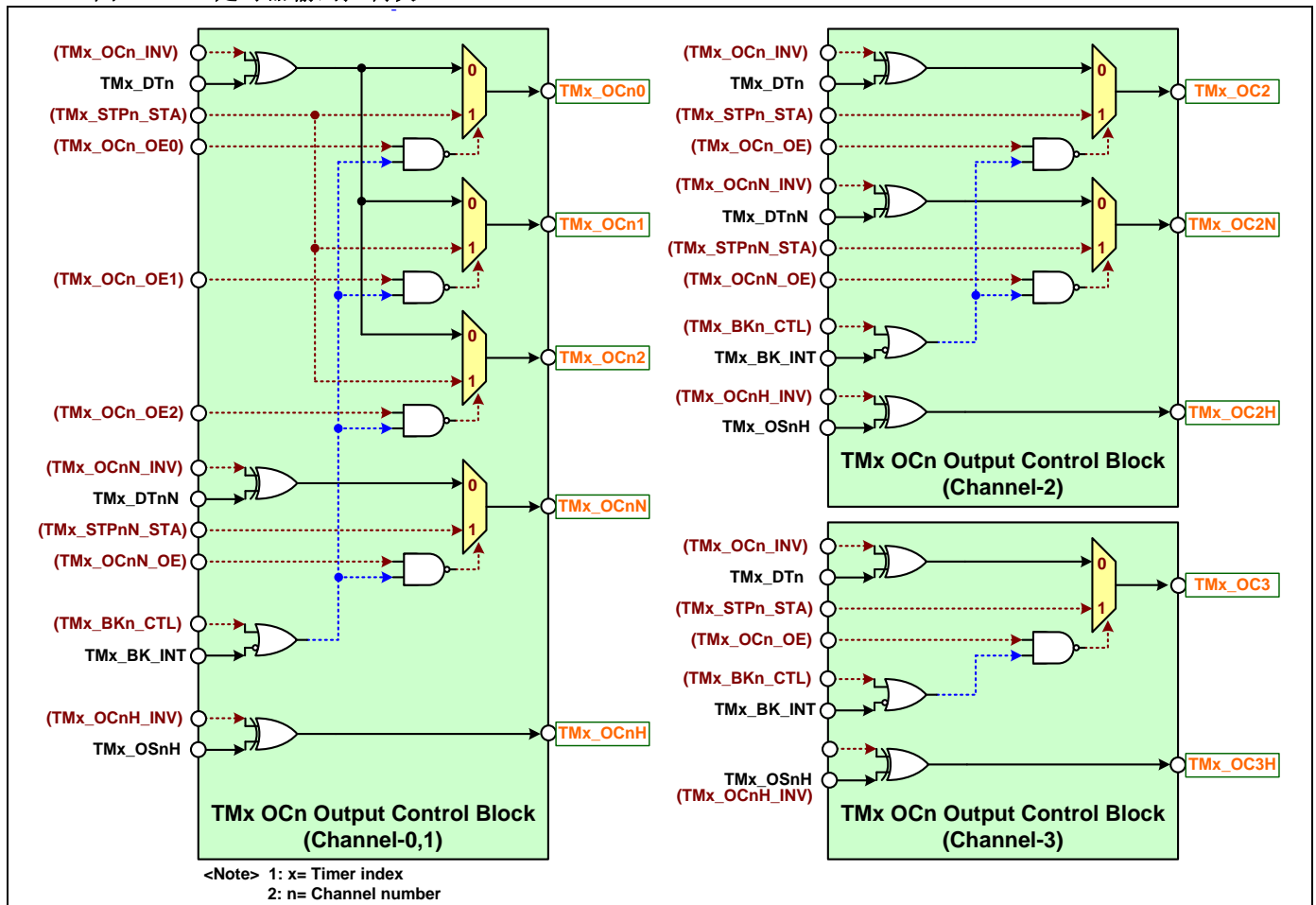
TMx_BK_INT 是定时器内部中止信号。用户可设置 **TMx_BKn_CTL** 寄存器设置当中止事件发送时，切换的行为是停止或暂停输出状态。参照“[中止控制块](#)”节以获取更多关于中止控制的信息。

在应用中，用户可通过 **TMx_ODLY_SEL** 寄存器选择输出延迟模式以延迟输出信号。通道-0 所有的输出信号根据该寄存器设置为不延迟。当选择“0 步”，通道-1、2、3 输出信号为正常无延迟输出；当选择“1 步”，通道-1、2、3 输出信号将分别延迟 1、2、3 步单位延时时间。该通道输出信号包含所有的 **TMx_OCn0**、**TMx_OCn1**、**TMx_OCn2**、**TMx_OCnN** 和 **TMx_OCnH**。1 步延时时间为约 3~5ns。

[注释: TMx_ODLY_SEL 不支持于 MG32F02A132/072。]

下面的图表展示了 TM3x 的定时器输出控制块。

图 18-32. 定时器输出控制块 ~ TM3x



18.14.3. 定时器输出使能预载控制

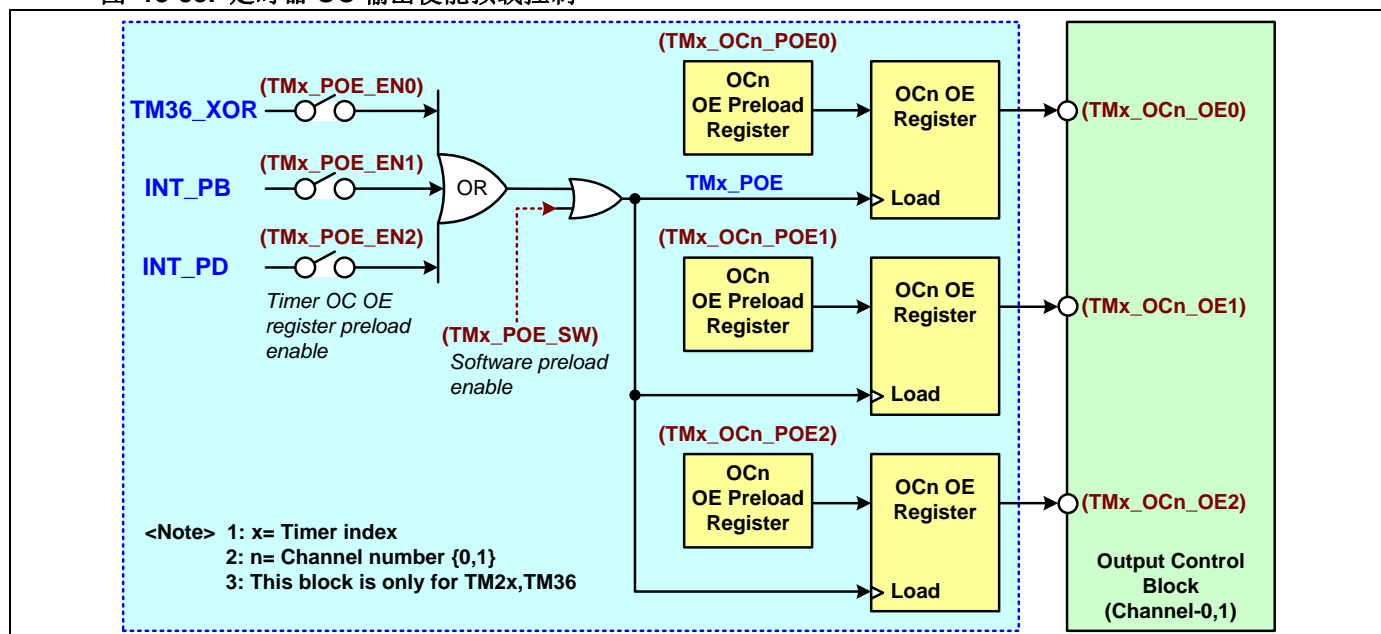
该模块提供输出预载寄存器 **TMx_OCn_POEm** 用于存储准备输出的控制状态，并在预载事件发生时预载到输出使能寄存器 **TMx_OCn_OEm** 中。**TMx_OCn_OEm** 寄存器是用于使能通道 0,1 的输出线 **TMx_OCn0**, **TMx_OCn1** 和 **TMx_OCn2** 输出。

独立状态初始寄存器位 **TMx_STPn_STA** 用于 **OCn** 输出通道-0/1/2/3; **TMx_STPnN_STA** 用于 **OCnN** 输出通道-0/1/2。

预载事件可来自于 TM36 XOR 输出、GPIO B/D 口内部 EXIC 全局中断事件或软件控制寄存器 **TMx_POE_SW** 设置。用户可通过设置独立寄存器 **TMx_POE_ENz** 使能预载事件源。(z = 预载事件源输入线标号{0,1,2})

下面的图表展示了定时器 OC 输出使能预载控制。

图 18-33. 定时器 OC 输出使能预载控制



18.15. 定时器中止控制块

中止控制块只支持于 TM36。该模块可从内部事件、外部事件或软件强制通过寄存器设置事件输入中止事件来中止定时器输出信号。

18.15.1. 中止使能和事件源

TMx_BK_EN 寄存器用于使能通道-0,1,2 的中止控制；独立的 TMx_BK_EN3 寄存器用于独立使能通道-3 的中止控制。

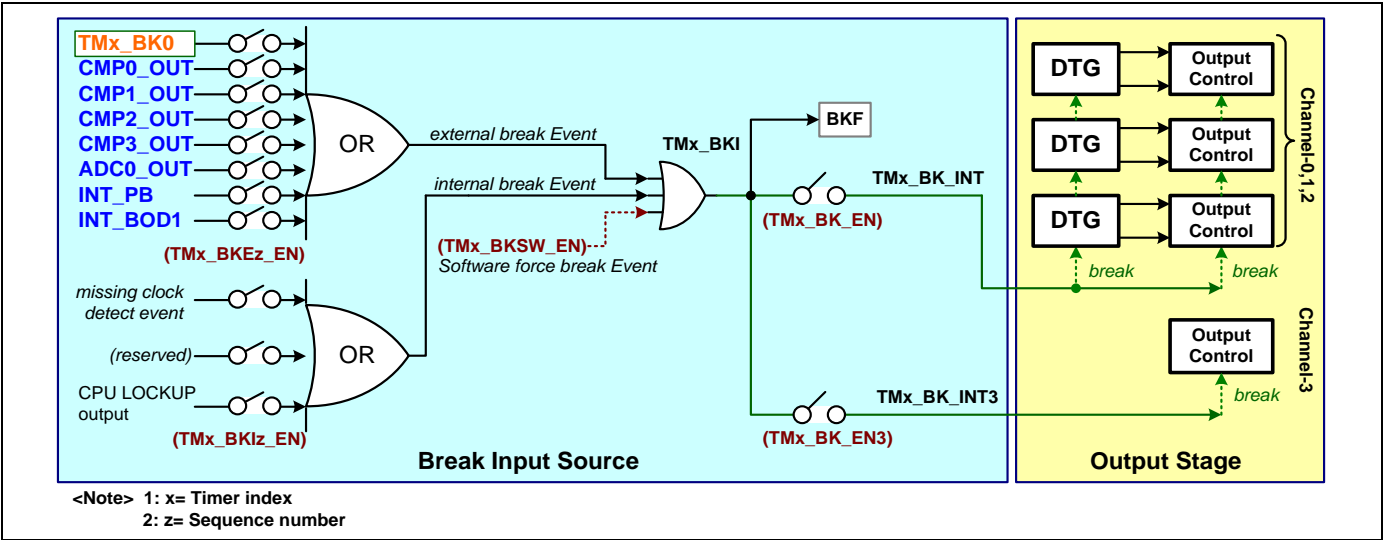
内部中止事件源包含丢失时钟检测事件和 CPU LOCKUP 输出事件；外部中止事件源包含外部引脚输入 TMx_BK0, 模拟比较器输出, ADC 电压窗口检测输出, GPIO B 端口内部 EXIC 全局中断事件和 BOD1 中断事件。此外，该模块还包含软件控制寄存器 TMx_POE_SW 强制中止事件。

用户可通过 TMx_BKI_ENz 寄存器设置使能内部中止事件源，通过 TMx_BKE_ENz 寄存器设置外部中止事件源，如下如表所示。(z =源标号)

当中止功能被使能且其中一个中止事件发生时，中止控制信号 TMx_BKI 会发送到 DTG 和输出控制块中停止输出信号。

下面的图表展示了定时器中止输入源块。

图 18-34. 定时器中止输入源



18.15.2. 中止控制模式

用户可通过 TMx_BK_MDS 寄存器为所有的通道选择中止功能模式：循环或锁止模式。

此外，用户可设置 TMx_BKn_CTL 寄存器设置当中止事件发生时，切换的行为是停止或暂停输出状态。参照“[定时器输出控制块](#)”节以获取更多信息。独立状态初始寄存器位 TMx_STPn_STA 用于 OCn 输出通道-0/1/2/3；TMx_STPnN_STA 用于 OCnN 输出通道-0/1/2。

下面的表格展示了定时器输出中止或停止控制的相关控制寄存器。TMx_OCn_OEm 寄存器用于使能输出线。

表 18-11. 定时器输出中止或停止控制

TMx 寄存器		中止信号 TMx_BK_INT	TMx_OCn / TMx_OCnN 输出
OCnN_OEm	BKn_CTL		
禁用	X	X	STPnN_STA 设置
使能	X	不启动	定时器比较或 PWM 输出状态
使能	暂停	启动	定时器比较或 PWM 输出最后的状态
使能	停止	启动	STPnN_STA 设置

TMx: x = 定时器模块标号, n= 通道标号, m= 线标号
OCnN: N = 代表互补通道

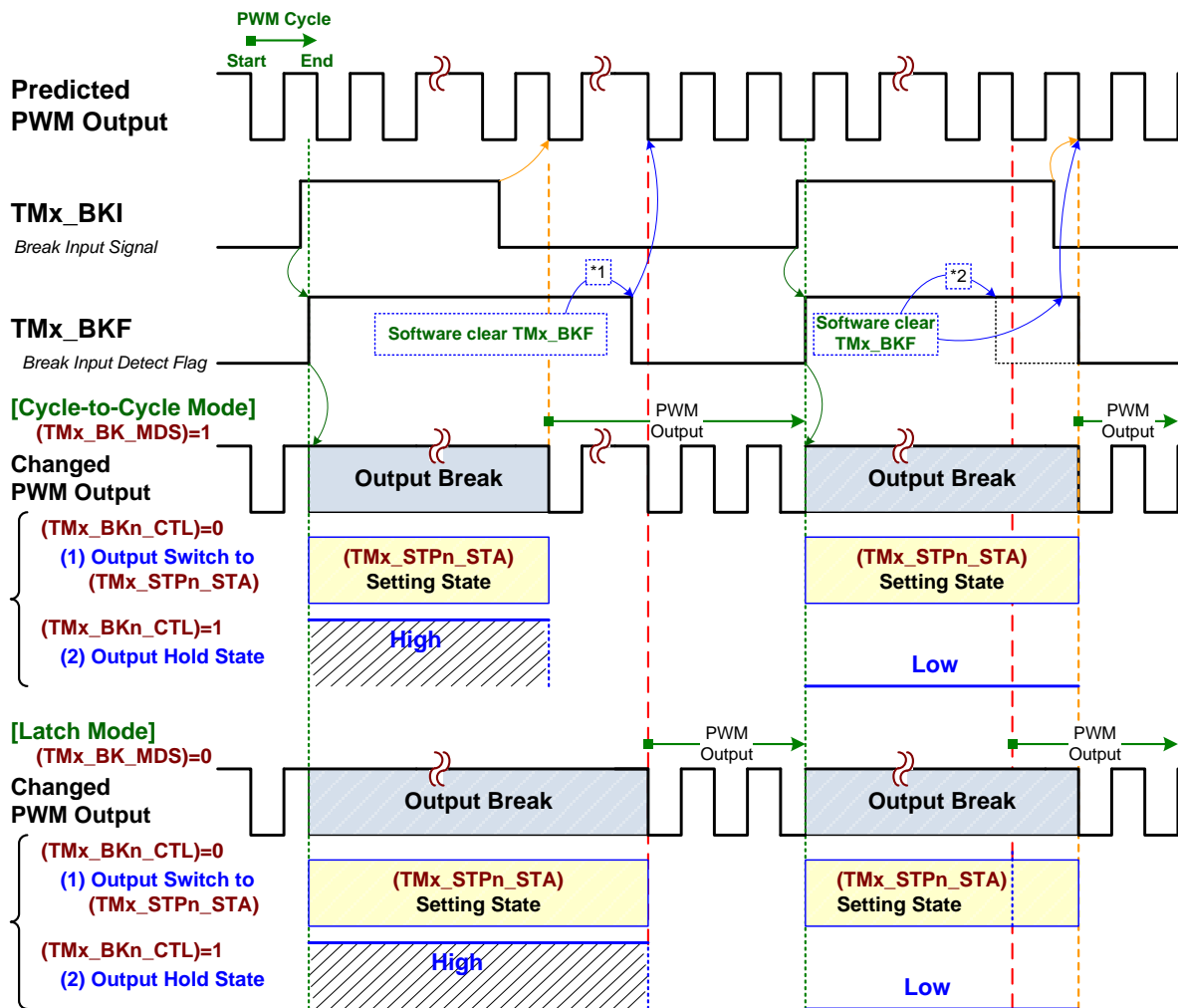
● 中止事件控制时序

当选择循环模式，比较或 PWM 输出会在中止控制信号 **TMx_BKI** 启动周期中停止。输出会在 **TMx_BKI** 信号未启动时，下一次比较或 PWM 输出的完整周期的起始被恢复和开始。当检测到中止控制信号启动时，**BKF** 标志也启动，并由固件清除。

当选择锁止模式，比较或 PWM 输出会在芯片检测到中止控制信号置起后停止并置起 **BKF** 标志。直到 **BKF** 标志被固件清除，会结束输出中止。输出会在下一次比较或 PWM 输出的完整周期的起始被恢复和开始。

下面的图表展示了定时器中止事件控制时序。

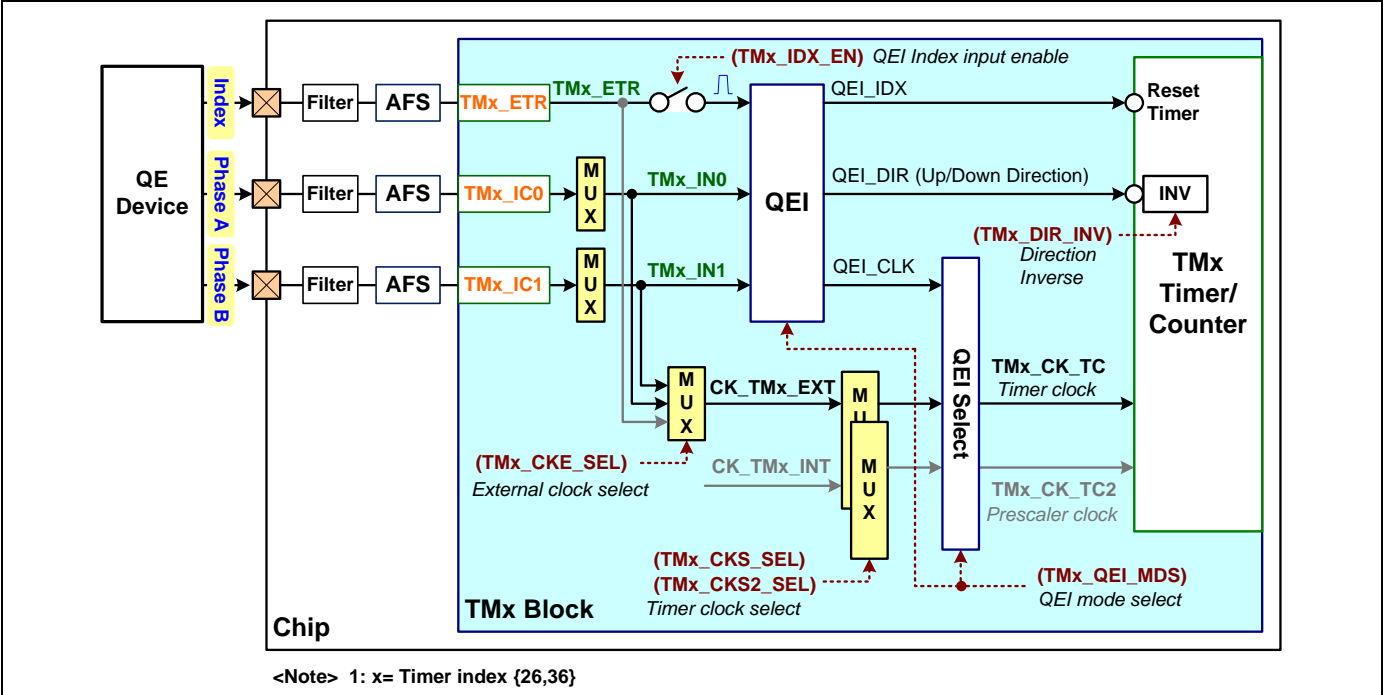
图 18-35. 定时器中止事件控制时序



18.16. QEI 控制块

QEI (正交编码接口)控制块只支持于 TM26 和 TM36 模块。下面的图表展示了定时器 QEI 控制块。

图 18-36. 定时器 QEI 控制块



18.16.1. QEI 控制模式

QEI 块提供 5 种控制模式，用户可以通过设置 **TMx_QEI_MDS** 寄存器来启用 QEI 控制并配置 QEI 控制模式。QEI 功能会在 **TMx_QEI_MDS** 寄存器设置为 0 时禁用。下面的表格展示了定时器 QEI 外部输入向上/向下控制模式的外部输入信号类型。

[注释]: QEI 模式 3,4 不支持于 MG32F02A132/072。

表 18-12. 定时器 QEI 外部输入向上/向下控制模式

QEI 模式	TMx_IN0	TMx_IN1	定时器计数
TMx_IN0 正极 (TMx_QEI_MDS)=1	高电平	X	持续向上计数
	低电平	X	持续向下计数
TMx_IN0 负极 (TMx_QEI_MDS)=2	高电平	X	持续向下计数
	低电平	X	持续向上计数
TMx_IN0 触发 (TMx_QEI_MDS)=3 (*1)	上升沿	高电平	向下计数一次
	上升沿	低电平	向上计数一次
	下降沿	高电平	向上计数一次
	下降沿	低电平	向下计数一次
TMx_IN1 触发 (TMx_QEI_MDS)=4 (*1)	高电平	上升沿	向上计数一次
	低电平	上升沿	向下计数一次
	高电平	下降沿	向下计数一次
	低电平	下降沿	向上计数一次
双沿 (TMx_QEI_MDS)=5	上升沿	高电平	向下计数一次
	上升沿	低电平	向上计数一次
	下降沿	高电平	向上计数一次
	下降沿	低电平	向下计数一次
	高电平	上升沿	向上计数一次
	低电平	上升沿	向下计数一次
	高电平	下降沿	向下计数一次
	低电平	下降沿	向上计数一次

TMx : x = 定时器模块标号

*1：不支持于 MG32F02A132/072

18.16.2. QEI 输入信号

QEI 块可输入 2 个外部信号 **TMx_IC0**和**TMx_IC1**以控制主定时器向上/向下计数；1 个可选的外部信号 **TMx_ETR**用于复位定时器。用户需通过 **TMx_IC0_MUX** 和 **TMx_IC1_MUX** 寄存器选择定时器通道-0,1 来自 **TMx_IC0**和**TMx_IC1**的输入信号源。

TMx_IDX_EN 寄存器位用于使能输入 QEI **Index** 信号。该信号会被 QEI 块检测，并输出 **QEI_IDX** 信号以复位主定时器。当 QEI 控制块被使能且检测到外部 **Index** 信号置起高电平，定时器将在向上计数期间复位，或在向下计数期间重载自动重载值，直到外部 **Index** 信号处于非活动状态。

QEI 块支持 2 种输入信号，一个是可输入 1 个时钟信号和 1 个向上/向下计数方向信号。用户可设置 QEI 控制模式 0,1 用于该输入信号类型。

第二个是可输入两相 A,B 信号。用户可设置 QEI 控制模式 3、4、5 用于该输入信号类型。QEI 块需转换成 1 个时钟信号(**QEI_CLK**)和 1 个向上/向下计数方向信号(**QEI_DIR**)。另外，方向控制位用于通过设置 **TMx_DIR_INV** 寄存器来反相实际输入信号的计时器计数方向。

18.16.3. QEI 控制模式 1,2

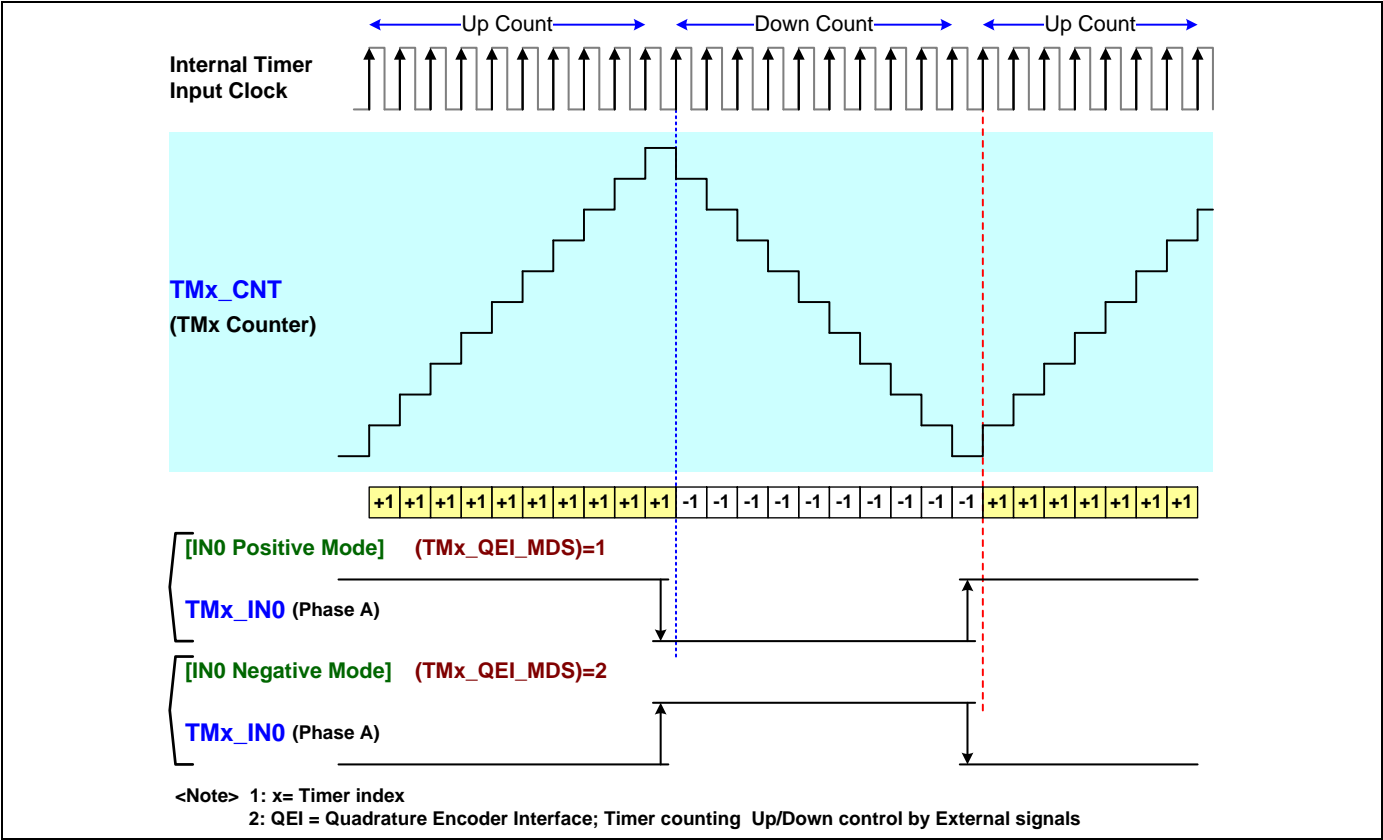
QEI 控制块可输入来自的时钟信号 **TMx_IC1**和 1 个来自的向上/向下方向信号 **TMx_IC0**。用户需通过 **TMx_QEI_MDS** 寄存器设置 QEI 模式 0,1。

若定时器时钟来源于外部 **TMx_IC1**，用户需通过 **TMx_CKE_SEL** 寄存器选择来自 **TMx_IN1** 的时钟源和通过 **TMx_CKS_SEL** 寄存器选择 **CK_TMx_EXT** 信号。当然，定时器时钟可通过用户程序选择定时器时钟来自内部时钟源。定时器向上或向下计数是直接通过输入信号 **TMx_IN0** 的电平控制的。

QEI 控制模式-0 和模式-1 之间的区别仅在于输入 **TMx_IN0** 信号的定时器计数方向极性。

下面的图表展示了 QEI 控制模式 1,2 的时序。

图 18-37. 定时器 QEI 控制模式 1,2



18.16.4. QEI 控制模式 3,4,5

QEI 控制块可输入来自 **TMx_IC0**和**TMx_IC1**的两相 A,B 信号。用户需通过 **TMx_QEI_MDS** 寄存器设置 QEI 控制模式 3、4、5。

[注释]: QEI 模式 3,4 不支持于 MG32F02A132/072。

QEI 控制模式 3 和 4 之间的不同点只有用于互换对于 **TMx_IN0** 和 **TMx_IN1** 输入信号的**相位 A** 和**相位 B** 功能定义。QEI 控制模式 3、4 和 5 之间的不同点在于模式 3、4 下，定时器只在**相位 A** 或**相位 B** 的边沿改变时计数，而模式 5 下则是在**相位 A** 和**相位 B** 的边沿改变时计数。

QEI 块接收来自 **TMx_IN0** 和 **TMx_IN1** 信号的**相位 A,B** 信号。它转换这两个信号成一个时钟信号(**QEI_CLK**)作为定时器时钟和一个向上向下方向控制信号(**QEI_DIR**)。定时器向上或向下计数是被 **TMx_IN0** 和 **TMx_IN1** 输入信号的电平和边沿控制的。

● QEI 编码器状态和转换

QEI 状态是通过**相位 A** 和**相位 B** 两个输入信号编码的，用于 QEI 块控制。

下面的表格展示了 QEI 编码状态。

表 18-13. 定时器 QEI 编码器状态

相位 A	相位 B	状态
1	0	1
1	1	2
0	1	3
0	0	4

下面的表格展示了定时器向上/向下计数方向控制的 QEI 编码器状态转换。

表 18-14. 定时器 QEI 编码器状态转换

从状态	到状态	方向
1	2	正极
2	3	
3	4	
4	1	
4	3	负极
3	2	
2	1	
1	4	

● QEI Index 控制

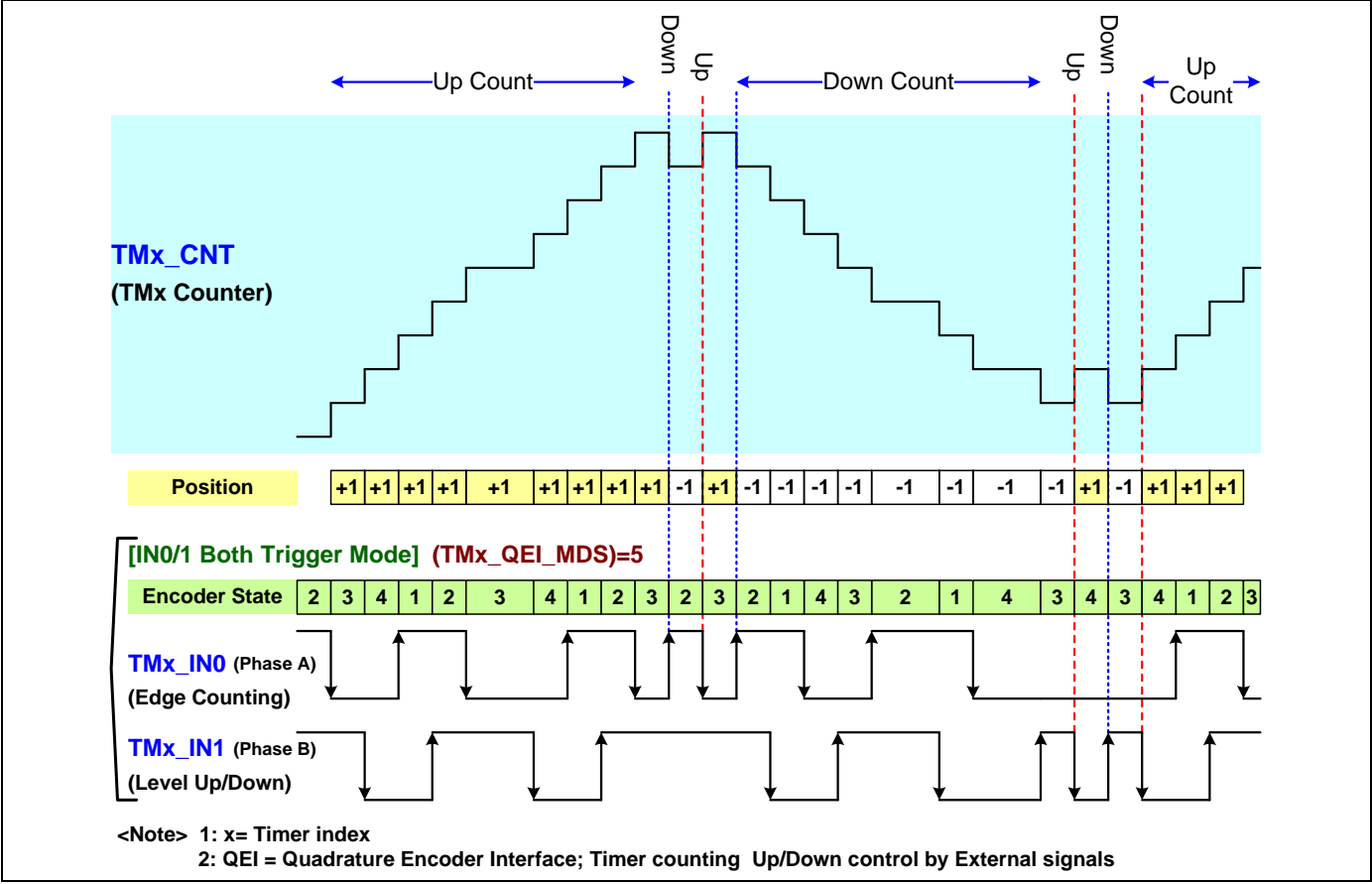
TMx_IDX_EN 寄存器用于使能 QEI 外部来自 **TMx_ETR**的 **Index** 输入信号。当外部 QE 设备输出信号只有一个时钟信号和一个向上/向下计数方向信号，用户需通过 **TMx_IDX_EN** 寄存器设置禁用 **Index** 输入功能。因此，QEI 块可直接使用这两个信号控制定时器向上/向下计数。

当外部 QE 时钟输出为两个相位信号和 1 个 **Index** 信号，用户可使能 **TMx_IDX_EN** 寄存器，QEI 块因此可检测 **Index** 信号以复位定时器。

TMx_IDX_MDS 寄存器用于通过 QEI 外部**相位**信号输入选择 QEI 编码器状态到复位定时器的转换状态。

下面的图表展示了 QEI 控制模式 3、4、5 时序。

图 18-38. 定时器 EXUD 控制模式 3,4,5



18.17. 定时器 DMA 操作

只有 TM36 模块的一条输入捕获通道和三条输出比较通道支持使用 DMA。定时器输入捕获数据可通过 DMA 控制器存入内存或其他外设。此外，定时器输出比较值可通过 DMA 控制器从内存或其他外设重载。

18.17.1. DMA 模块设置

当芯片支持 DMA（直接内存访问）控制器时，用户可以在 DMA 数据传输前，在 DMA 模块中设置关于来源/目的设备、通道请求仲裁等 DMA 设置。DMA 来源和目的可以是内存或外设。

参照 DMA 章以获取更多关于 DMA 模组设置的细节。

18.17.2. 定时器 DMA 控制

DMA 设置完成后，用户需设置定时器模块的 DMA 使能位 **TMx_DMA_CCnE**。(n=定时器通道标号)

最后，相关的通道请求起始位 **DMA_CHn_REQ** 是必须的，用于设置 DMA 传输启动(n = DMA 通道标号)。然后传输源和目的设备会置起 RX/TX 请求信号到 DMA 控制器，DMA 控制器便会置起接收信号到请求源/目的设备。此时，该数据传输连接是用于 DMA 传输的。

● 定时器 IC 到 DMA

TMx_DMA_CC3E 寄存器位用于使能定时器输入通道-3 数据利用 DMA 从定时器输入捕获到 DMA 目的地。

[注释]: DMA 定时器输入捕获模式不支持全计数模式。

在 DMA 传输周期中，IC/OC/PWM 标志 **TMx_CF3A** 和 **TMx_CF3B** 是被硬件屏蔽的。

● 定时器 OC 来自 DMA

TMx_DMA_CC0E, **TMx_DMA_CC1E** 和 **TMx_DMA_CC2E** 寄存器位用于独立使能来自 DMA 源到定时器输出通道-0/1/2 的输出比较重载寄存器的 DMA 数据发送。

在 DMA 传输周期中，IC/OC/PWM 标志 **TMx_CFnA** 和 **TMx_CFnB** 是被硬件屏蔽的。(n = 定时器通道标号 0,1,2)

对于定时器输出比较 DMA 功能，用户可通过 **TMx_DMA_OMDS** 寄存器选择定时器输出 DMA 请求模式，当选择 ITR，DMA 请求会在 TOF 或 ITR 输入信号启动时置起；当选择 TOF 时，DMA 请求只会在 TOF 启动时置起；被置起的 DMA 请求会为使能了 DMA 功能(**TMx_DMA_CCnE**)的通道更新 **TMx_CCnB** 寄存器(输出比较重载寄存器)。(n=定时器通道标号 0,1,2)

[注释]: 每次 DMA 更新 **TMx_CCnB** 寄存器时，TMx 会复制 **TMx_CCnB** 寄存器的内容到 **TMx_CCnA** 比较寄存器中，因此在做 DMA 数据传输之前，**TMx_CCnB** 寄存器一定要有第一个比较数据。

18.17.3. 定时器的 DMA 中断标志

DMA 工作周期中，模块的中断标志会控制和执行 3 种类型，如下表格。其中一方在 DMA 工作过程中被屏蔽，另一方会在标志被置起后禁用 DMA 功能。此时，硬件会禁用 **TMx_DMA_CCnE** 位。其他操作与 DMA 操作中不执行的操作相同。

表 18-15. 定时器 DMA 功能中断标志控制

行为	DMA 操作中屏蔽标志	标志置起后 DMA 被禁用(*1)	一般控制
外设	(数据溢出标志)	(错误/检测标志)	(其他标志)
TM36	TM36_CF0A/0B TM36_CF1A/1B TM36_CF2A/2B TM36_CF3A/3B	TM36_TOF TM36_TUF	TM36_EXF TM36_TOF2 TM36_TUF2 TM36_DIRCF TM36_IDXF TM36_QPEF TM36_BKF

注释-1：当标志被置起，若相关中断使能位没被使能，它不会强行禁用外设 DMA。

19. IWDT (独立看门狗)

19.1. 简介



该芯片有 1 个独立看门狗定时器 (IWDG) 作为恢复手段用于 CPU 可能受到软件翻转的情况。当计数器到达给定的超时值时它会触发系统复位。

19.2. 特性

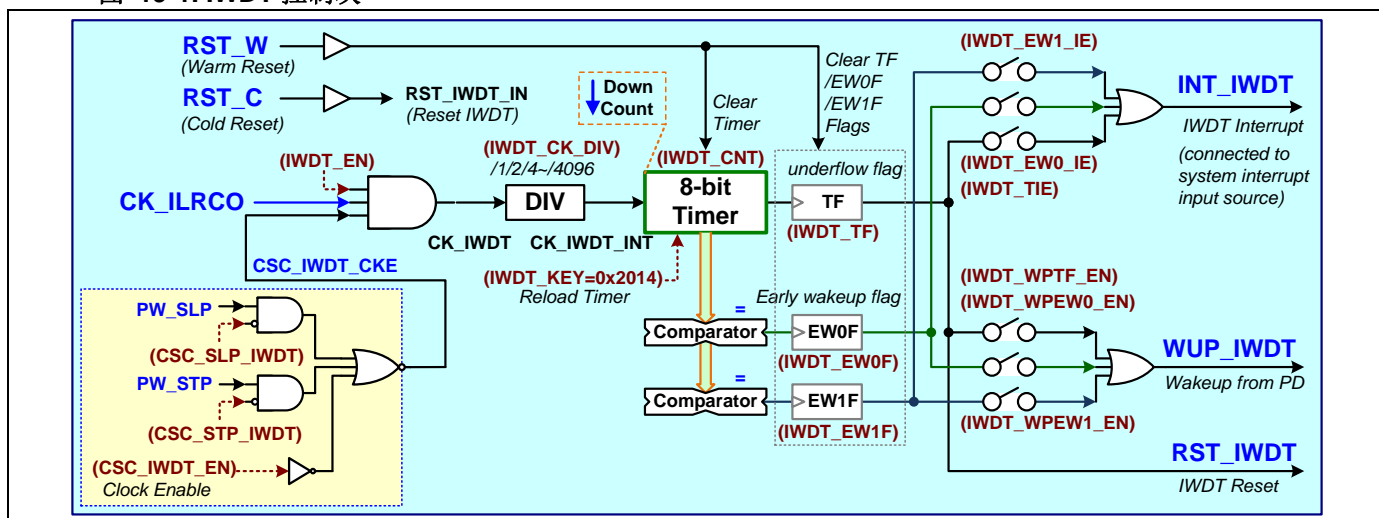
- 有 12 位预分配器的由自身 CK_ILRCO 作为时钟源的 8 位向下计数器
- 兼容工作在 SLEEP 和 STOP 模式
- 当计数器下溢时可选择复位或中断
- 通过中断支持两个早唤醒比较器
- 支持寄存器值保护和复位锁定功能

19.3. 控制块

IWDT 看门狗定时器由 1 个 12 位时钟分频器、1 个 8 位定时器和 2 个早期唤醒数字比较器组成。

下面的图表展示了 IWDT 控制块。

图 19-1. IWDt 控制块



19.4. 使能和时钟

19.4.1. IWDT 全局使能

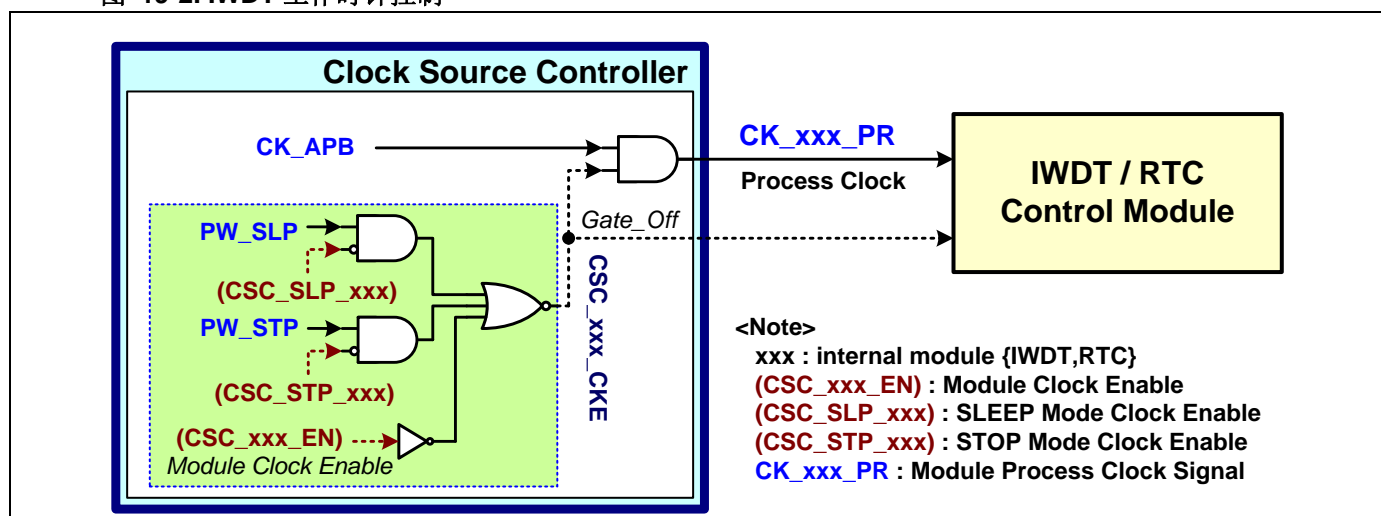
IWDT 模块的所有功能全局使能位是 **IWDT_EN**。当该位被禁用时，所有的 IWDT 功能将无法工作。

19.4.2. IWDT 时钟控制

● 模块工作时钟

该模块工作时钟 **CK_IWDT_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 CSC(时钟源控制器) 模块。该时钟可通过 **CSC_IWDT_EN** 寄存器使能。用户可以在芯片进入 **SLEEP** 模式之前通过设置 **CSC_SLP_IWDT** 或 **CSC_STP_IWDT** 寄存器规划在 **SLEEP** 或 **STOP** 模式下是否让 IWDT 时钟继续运行。参照系统时钟章以获取更多信息。

图 19-2. IWDT 工作时钟控制



● 模块内部时钟

IWDT 定时器时钟源来自于内部 ILRCO 时钟 **CK_ILRCO**。当模块全局使能位 **IWDT_EN** 被禁用时，内部定时器时钟会被关闭。

内部定时器时钟也像模块工作时钟一样被 CSC 的 **CSC_IWDT_EN**, **CSC_SLP_IWDT** 和 **CSC_STP_IWDT** 寄存器控制。

1 个 12 位时钟分频器用于 IWDT 输入时钟。用户可通过 **IWDT_CK_DIV** 寄存器设置该分频值为 1/2/4/8/~4096。

19.5. 中断和事件

IWDT 模块中有 1 种信号 **INT_IWDT**。**INT_IWDT** 发送到外部中断控制器 (EXIC) 作为中断事件。

19.5.1. IWDT 中断控制和状态

中断标识是用于中断服务程序 (ISR) 流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。

19.5.2. IWDT 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

● TF

IWDT 定时器超时中断标志是 (**IWDT_TF**)，相关的中断使能寄存器位是 **IWDT_TIE**。

● EW0F

IWDT 早期唤醒-0 中断标志是 (**IWDT_EW0F**)，相关的中断使能寄存器位是 **IWDT_EW0_IE**。该标志会在寄存器达到 **0x20** 时置起。

● EW1F

IWDT 早期唤醒-1 中断标志是 (**IWDT_EW1F**)，相关的中断使能寄存器位是 **IWDT_EW1_IE**。该标志会在寄存

器达到 0x40 时置起。

19.6. 寄存器保护和锁定

IWDT 复位之后，除了 IWDT_STA、IWDT_KEY 这两个寄存器以外，所有的 IWDT 寄存器将会处于写保护状态。通过向 IWDT_KEY 寄存器写 0xA217 值即可解除寄存器保护，并时刻处理控制。相对地，写其他除 0xA217 以外的值将会使寄存器处于保护状态。读 IWDT_KEY 寄存器的值以获得寄存器的状态是被保护 (=1)还是未保护 (=0)。

它还提供热复位后的寄存器锁定功能。向 IWDT_LOCK 寄存器写值 0x712A 可锁定除 IWDT_STA 和 IWDT_KEY 寄存器外的写操作。当寄存器被锁定，寄存器在被硬件冷复位之前寄存器的值都无法被系统热复位改变。写除 0x712A 外的值是无效果的。读 IWDT_LOCK 寄存器的值以获得寄存器的状态是被锁定 (=1)还是未锁定 (=0)。

比较特殊的硬件功能控制，CSC_IWDT_EN, CSC_SLP_IWDT, CSC_STP_IWDT 和 PW_WKSTP_IWDT 是被 IWDT_LOCK 寄存器控制锁定的，当 IWDT_LOCK 寄存器设置锁定时，这些寄存器不会被热复位复位。

参照系统复位章的“寄存器保护和锁定”节的表格和描述以获取更多信息。

硬件选项定义 IWDT_CFG_IWDT_WP 寄存器写保护默认值，这些默认值不会在断电后丢失。该寄存器值在系统复位后从选项字节(OB)闪存中载入。参照硬件选项章以获取更多信息。

下面的表格展示了 IWDT 寄存器不同设定下的写保护。

表 19-1. IWDT 寄存器写保护表

寄存器设置			IWDT 寄存器				
			MG32F02A132/072		MG32F02A032		
CFG_IWDT_WP	IWDT_LOCK	IWDT_KEY	IWDT_KEY IWDT_STA	其他	IWDT_KEY IWDT_STA	IWDT_EW1_IE IWDT_EW0_IE IWDT_TIE IWDT_EW1_WPEN IWDT_EW0_WPEN IWDT_TF_WPEN	其他
使能	锁定	保护	-	-	V	-	-
		未保护				-	
	未锁定	保护				-	
		未保护				V	
禁用	锁定	保护	V	-	V	-	-
		未保护		-		-	-
	未锁定	保护		-		-	-
		未保护		V		V	V

注释 V：允许写操作， -：锁定写操作

19.7. IWDT 功能控制

19.7.1. IWDT 定时器控制

IWDT 看门狗定时器由 1 个 12 位预分频器和 1 个 8 位定时器组成。该定时器通过 IWDT_EN 寄存器设置使能，当看门狗定时器被使能时，软件必须保持在定时器超时之前通过写 0x2014 到 IWDT_KEY 寄存器中复位定时器。看门狗定时器是向下计数定时器，且用户可通过读 IWDT_CNT 寄存器以获取实时计数器值。当看门狗定时器复位，定时器会重载 0xFF 值重新计数。

若芯片因任何错误失去控制，意味着 CPU 可能无法正常运行程序，然后固件会错过复位定时器（写 0x2014 到 IWDT_KEY 寄存器）从而使定时器超时，看门狗定时器超时使 IWDT 产生复位事件 RST_IWDT 并发送到复位源控制器（RST）中作为热复位或冷复位事件。

该复位事件可通过 RST 寄存器使能复位芯片。最终，若相关复位控制位被使能，该复位事件会使 CPU 重启。参照系统复位章以获取更多关于复位事件和控制的信息。

IWDT 能记录硬件设置字节（OB）中关于 IWDT on/off、输入时钟分频器值、IWDT 寄存器写保护相关的默认的初始值和 IWDT_EN, IWDT_CK_DIV, IWDT_LOCK 寄存器的关系。这些寄存器可通过硬件设置字节(OB)的设置在上电或冷复位周期中被自动初始化。

19.7.2. STOP 模式下工作

若用户在进入 **SLEEP** 或 **STOP** 模式之前设置 **CSC_IWDT_EN** 和 **CSC_STP_IWDT** 寄存器，IWDT 就可在 **STOP** 模式下运行。当 IWDT 工作于 **STOP** 模式下，APB 时钟会停止，且该模块会异步控制所有逻辑。

19.7.3. STOP 模式下唤醒

IWDT 支持通过看门狗定时器下溢事件、早期唤醒-0 检测(**IWDT_EW0F** 标志置起)和早期唤醒-1 检测事件(**IWDT_EW1F** 标志置起)把芯片从 **STOP** 模式唤醒。独立唤醒使能位 **IWDT_TF_WPEN**, **IWDT_EW0_WPEN** 和 **IWDT_EW1_WPEN** 用于上述唤醒事件。

当芯片进入 **STOP** 模式且发生任何一个 IWDT 唤醒事件时，IWDT 将唤醒事件发送到电源控制器（PW）以作为系统唤醒事件。若相关控制寄存器被使能，该唤醒事件会使芯片唤醒。

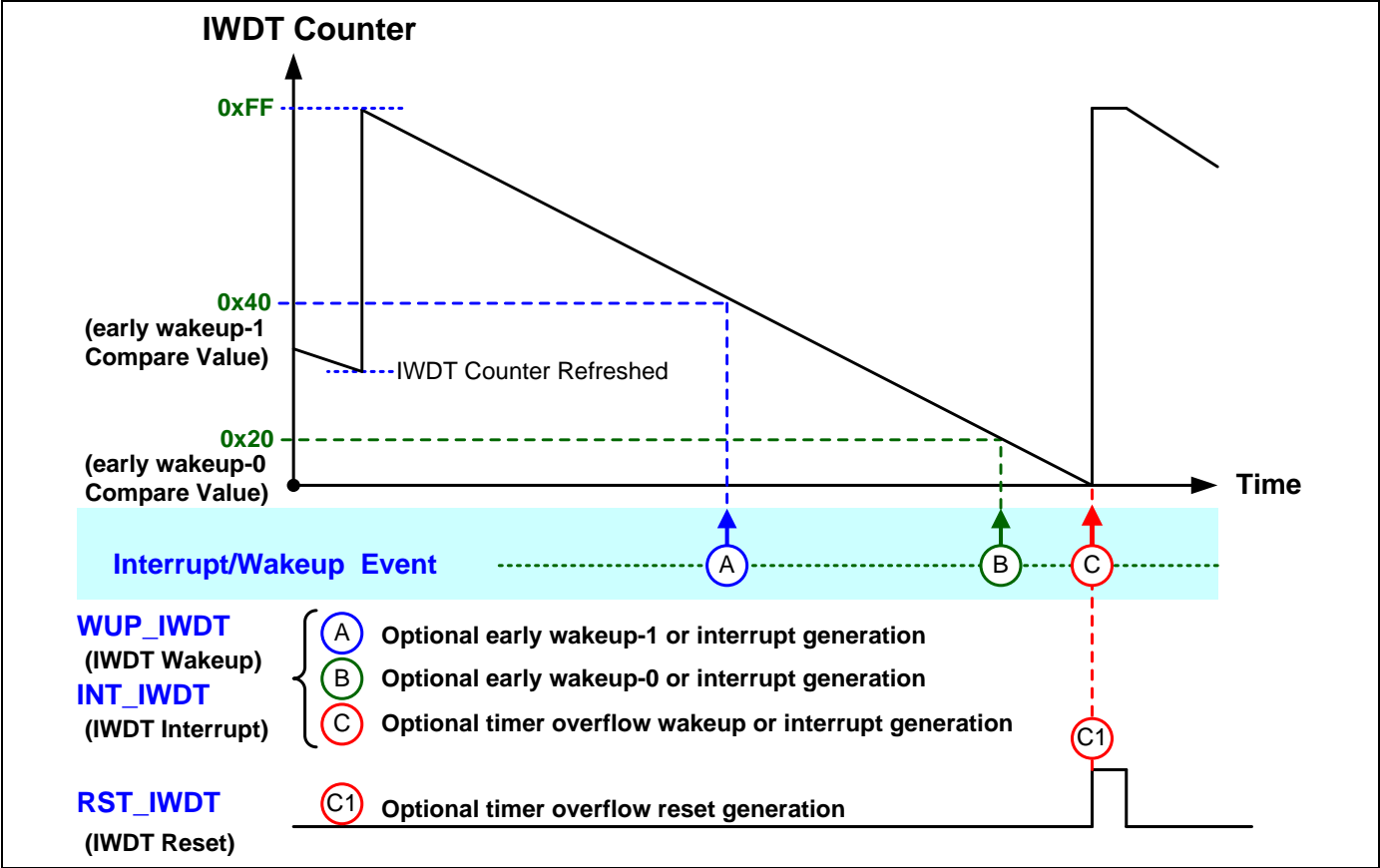
参照系统电源和中断章中描述以获取更多关于唤醒事件和控制的信息。

19.7.4. IWDT 事件时序

IWDT 看门狗定时器是向下计数定时器且定时器保持从 0xFF 值开始计数。IWDT 可输出 **IWDT_EW0F** 和 **IWDT_EW1F** 标志用于计数器值为 0x20,0x40 时，IWDT 定时器计数匹配事件。用户可在这两个 ISR 里加入管理代码以在 IWDT 定时器下溢之前修复可能发生的问题，此外，IWDT 还可以输出 **IWDT_TF** 作为定时器超时标志。

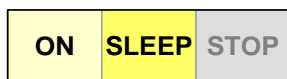
下面的图表展示了 IWDT 事件时序。

图 19-3. IWDT 事件时序



20. WWDT (窗口看门狗定时器)

20.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片内建 1 个系统窗口看门狗（WWDT）用于检测导致应用程序异常的软件错误发生。在计数器达到给定的超时值时看门狗电路将产生 1 个系统复位。

WWDT 有一个可配置的时间窗口，可用来检测异常发生晚或早的应用行为。

20.2. 特性

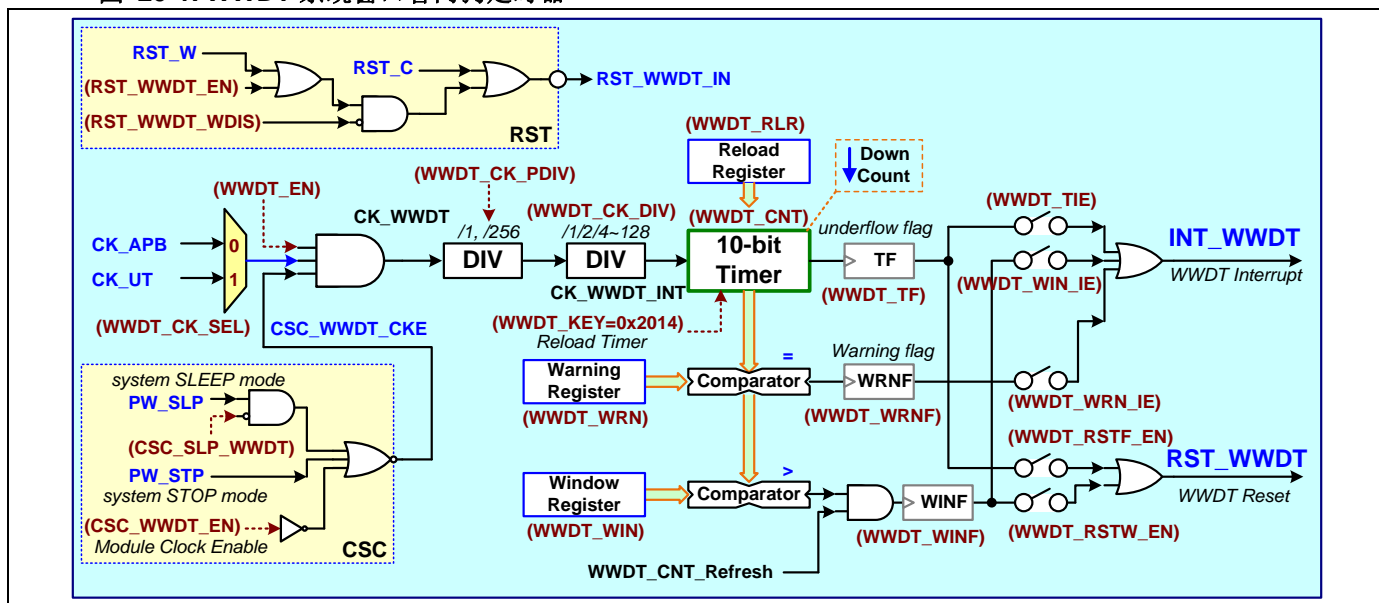
- 有 1 或 256 分频器、1/2/4~128 分频器的 10 位计数器
- 可配置的时间窗口用来检测异常发生晚或早的应用行为
- 计数器下溢或窗口外重载时可选择复位或中断
- 支持警报中断
- 支持寄存器键值保护和复位锁定功能

20.3. 控制块

WWDT 看门狗定时器由 1 个/1 或/256 的时钟预分频器、1 个 7 位时钟分频器和 1 个 10 位定时器、1 个警告数字比较器和 1 个窗口数字比较器组成。

下面的图表展示了 WWDT 控制块。

图 20-1. WWDT 系统窗口看门狗定时器



20.4. 使能和时钟

20.4.1. WWDT 全局使能

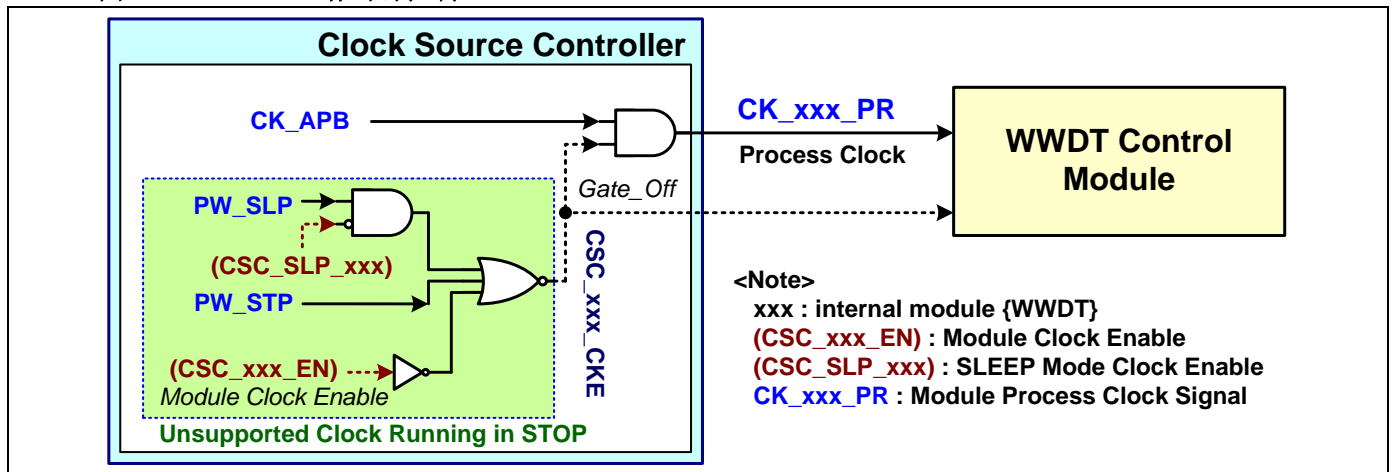
WWDT 模块的所有功能全局使能位是 **WWDT_EN**。当该位被禁用时，所有的 WWDT 功能将无法工作。

20.4.2. WWDT 时钟控制

● 模块工作时钟

该模块工作时钟 **CK_WWDT_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 **CSC**（时钟源控制器）模块。该时钟可通过 **CSC_WWDT_EN** 寄存器使能。用户可以在芯片进入 **SLEEP** 模式之前通过设置 **CSC_SLP_WWDT** 寄存器规划在 **SLEEP** 模式下是否让 WWDT 时钟继续运行。参照系统时钟章以获取更多信息。

图 20-2. WWDT 工作时钟控制



● 模块内部时钟

用户可通过 **WWDT_CK_SEL** 寄存器选择内部定时器时钟源来自于内部 APB 时钟 **CK_APB** 或内部单元时钟 **CK_UT**。当模块全局使能位 **WWDT_EN** 被禁用时，内部定时器时钟会被关闭。

内部定时器时钟也像模块工作时钟一样被 CSC 的 **CSC_WWDT_EN** 和 **CSC_SLP_WWDT** 寄存器控制。

1 个时钟预分频器和 1 个时钟分频器用于 WWDT 输入时钟。用户可通过 **WWDT_CK_PDIV** 寄存器设置预分频值为 1 或 256；通过 **WWDT_CK_DIV** 寄存器设置预分频值为 1/2/4/8~128。

[注释]: 通常内部时钟频率需比模块工作时钟慢至少 1/2。

20.5. 中断和事件

WWDT 模块中有 1 种信号 **INT_WWDT**。**INT_WWDT** 发送到外部中断控制器 (EXIC) 作为中断事件。

20.5.1. WWDT 中断控制和状态

中断标识是用于中断服务程序 (ISR) 流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。

20.5.2. WWDT 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

● TF

WWDT 定时器超时中断标志是 (**WWDT_TF**)，相关的中断使能寄存器位是 **WWDT_TIE**。

● WINF

WWDT 计数器刷新和值超出窗口比较阈值状态标志是 (**WWDT_WINF**)，相关的中断使能寄存器位是 **WWDT_WIN_IE**。当固件通过写 **0x2014** 到 **WWDT_KEY** 以重载定时器且计数器值超过 **WWDT_WIN** 阈值时该标志会被置起。

- WRNF

WWDT 计数器警告标志是(WWDT_WRNF)，相关的中断使能寄存器位是 WWDT_WRN_IE。当 WWDT 计数器达到 WWDT_WRN 值时该标志置起。

20.6. WWDT 寄存器保护

WWDT 复位之后，除了 WWDT_STA、WWDT_KEY 这两个寄存器以外，所有的 WWDT 寄存器将会处于写保护状态。通过向 WWDT_KEY 寄存器写 0xA217 值即可解除寄存器保护，并时刻处理控制。相对地，写其他除 0xA217 以外的值将会使寄存器处于保护状态。读 WWDT_KEY 寄存器的值以获得寄存器的状态是被保护 (=1) 还是未保护 (=0)。

参照“[寄存器保护和锁定](#)”节的表格和描述以获取更多信息。

20.7. WWDT 功能控制

20.7.1. WWDT 定时器控制

IWDT 看门狗定时器由 1 个/1 或/256 的预分频器、1 个 7 位时钟分频器和 1 个 10 位定时器组成。该定时器通过 WWDT_EN 寄存器设置使能，当看门狗定时器被使能时，软件必须保持在定时器超时之前通过写 0x2014 到 WWDT_KEY 寄存器中复位定时器。看门狗定时器是向下计数定时器，且用户可通过读 WWDT_CNT 寄存器以获取实时计数器值。当看门狗定时器复位，定时器会重载 WWDT_RLR 寄存器值重新计数。

若固件失去控制，意味着 CPU 可能无法正常运行程序，然后固件会错过复位定时器(写 0x2014 到 WWDT_KEY 寄存器)从而使定时器超时，看门狗定时器超时使 WWDT 产生复位事件 RST_WWDT 并发送到复位源控制器 (RST) 中作为热复位或冷复位事件。当通过固件通过写 0x2014 到 WWDT_KEY(复位定时器)以重载定时器且同时计数器值超过 WWDT_WIN 阈值时，仍会使 WWDT 产生复位事件。上述复位事件有自己独立地复位事件使能位 WWDT_RSTF_EN 和 WWDT_RSTW_EN。

该复位事件可通过 RST 寄存器使能复位芯片。最终，若相关复位控制位被使能，该复位事件会使 CPU 重启。参照“系统复位”章以获取更多关于复位事件和控制的信息。

20.7.2. WWDT 模块复位控制

WWDT 模块可被冷复位复位，当 WWDT 热复位禁用寄存器位 RST_WWDT_DIS 被设置为 0 时也可以被热复位复位；当 WWDT 寄存器位 RST_WWDT_DIS 被设置为 1 时，WWDT 模块将不能被热复位复位。在这种情况下，在接下来的热复位期间，WWDT 仍可以通过其他复位事件做一次看门狗功能，随后，若 WWDT 计数下溢，WWDT 会被复位。

1 个复位使能寄存器 RST_WWDT_EN 用于软件控制强制复位 WWDT 模块。

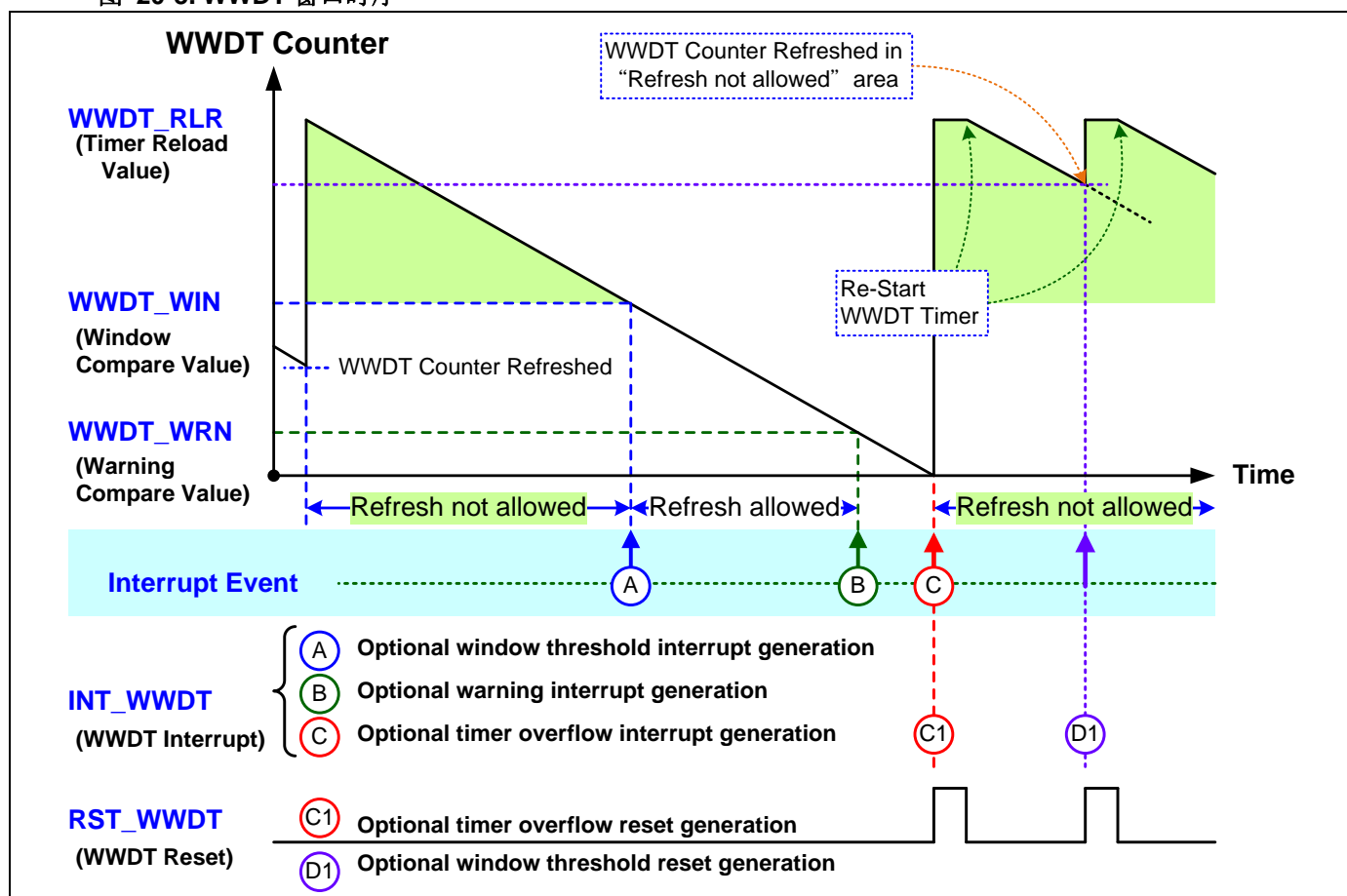
20.7.3. WWDT 窗口时序

WWDT 看门狗定时器是向下计数定时器，且定时器会从重载 **WWDT_RLR** 寄存器值开始计数。

WWDT 可输出 **WWDT_WINF** 和 **WWDT_WRNF** 标志用于窗口比较和可设置比较计数器值警告的 WWDT 定时器计数事件。**WWDT_WIN** 和 **WWDT_WRN** 寄存器用于设置比较器值用于置起 **WWDT_WINF** 和 **WWDT_WRN** 标志。此外，WWDT 可输出 **WWDT_TF** 作为 WWDT 定时器超时标志。

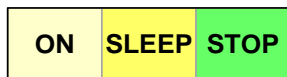
下面的图表展示了 WWDT 窗口时序。

图 20-3. WWDT 窗口时序



21. RTC (实时时钟)

21.1. 简介



The module can be running in all power operation modes.

实时时钟是 1 个独立的 32 位定时器，RTC 提供一个带有可编程报警中断的时钟。用户可以通过软件可编程的报警秒、分钟、小时、日和日期作为日历。

RTC 提供 1 个唤醒标志来用中断方式从掉电模式执行自动唤醒。

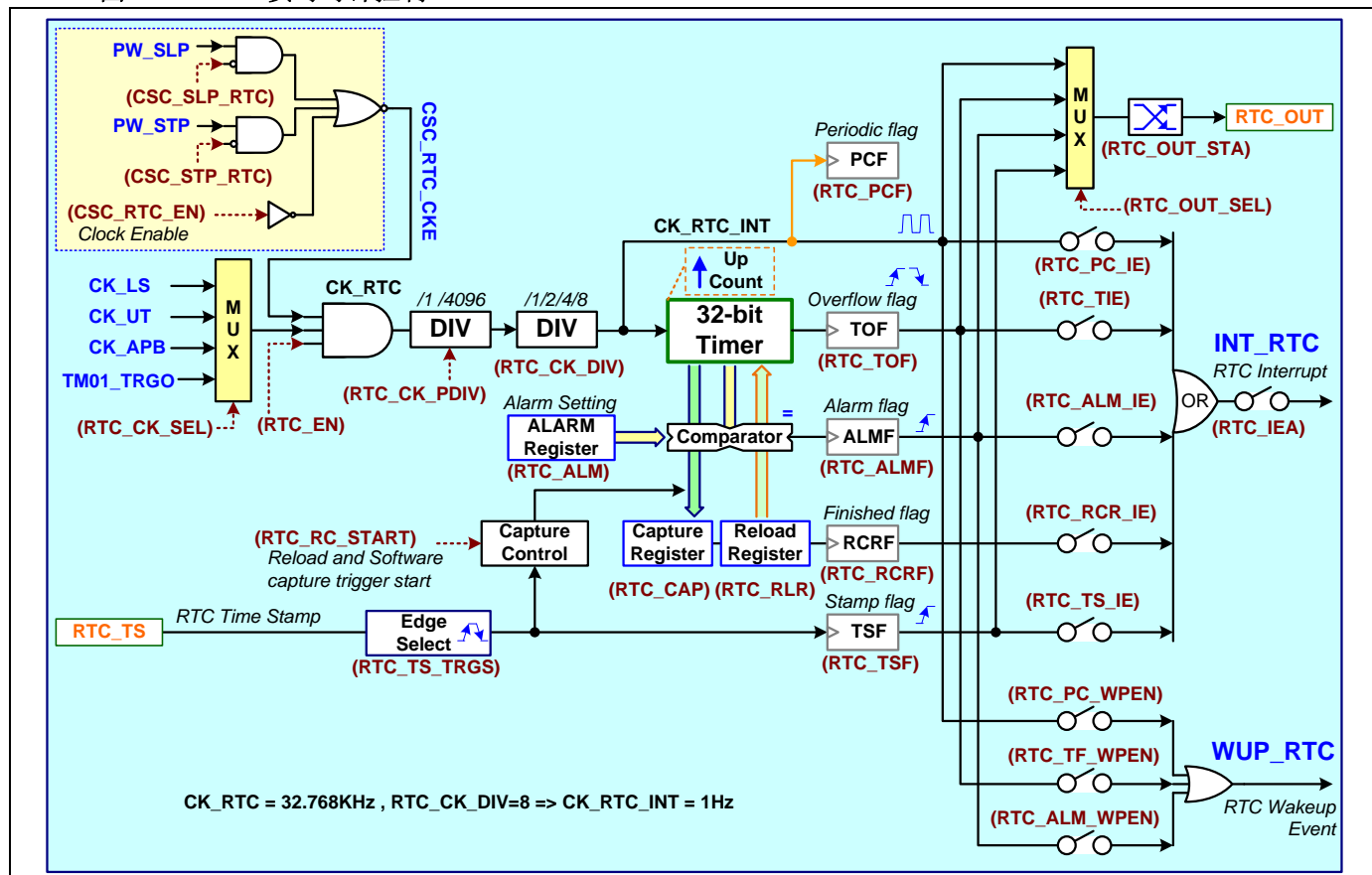
21.2. 特性

- 内置可选时钟源的 32 位计数器
- 支持报警和时间戳功能
 - 支持 32 位可编程比较寄存器用于报警功能
- 支持从 STOP 模式唤醒
- 支持定时时钟滴答中断或唤醒
- 支持寄存器键值保护和复位锁定功能

21.3. 控制块

RTC 由 1 个时钟预分频器和 1 个时钟分频器、1 个 32 位定时器、报警数字比较器和时间戳控制逻辑组成。下面的图表展示了 RTC 控制块。

图 21-1. RTC 实时时钟控制



21.4. IO 线

21.4.1. IO 信号

- **RTC_TS**

RTC 时间戳输入信号。当 RTC 接收到时间戳信号，TSF 标志会被置起，RTC 定时器值会被捕获到捕获寄存器。

- **RTC_OUT**

它是从 RTC 内部事件或信号中选择的选择性输出信号，并输出到引脚或其他内部模块。

21.4.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。参照用户手册 GPIO 章中“[IO 模式](#)”节以获取更多关于 IO 模式设置的信息。

每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“[功能复用选择](#)”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“[引脚功能复用表](#)”以获取更多信息。

21.5. 使能和时钟

21.5.1. RTC 全局使能

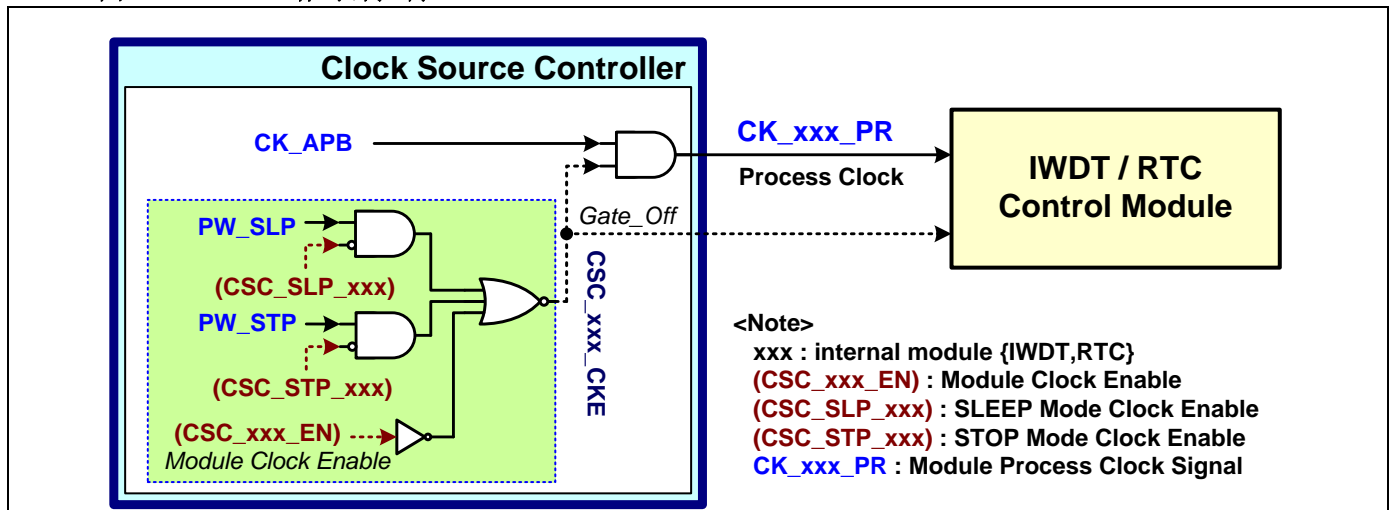
RTC 模块的所有功能全局使能位是 **RTC_EN**。当该位被禁用时，所有的 RTC 功能将无法工作。

21.5.2. RTC 时钟控制

- **模块工作时钟**

该模块工作时钟 **CK_RTC_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 CSC（时钟源控制器）模块。该时钟可通过 **CSC_RTC_EN** 寄存器使能。用户可以在芯片进入 **SLEEP** 模式或 **STOP** 模式之前通过设置 **CSC_SLP_RTC** 或 **CSC_STP_RTC** 寄存器规划在 **SLEEP** 或 **STOP** 模式下是否让 RTC 时钟继续运行。参照系统时钟章以获取更多信息。

图 21-2. RTC 工作时钟控制



- **模块内部时钟**

用户可通过 **RTC_CK_SEL** 寄存器选择内部定时器时钟源来自于内部 ILRCO 时钟 **CK_ILRCO**、内部单元时钟 **CK_UT**、内部 APB 时钟 **CK_APB** 或定时器触发输出信号 **TM01_TRGO**。当模块全局使能位 **RTC_EN** 被禁用时，内部定时器时钟会被关闭。

内部定时器时钟也像模块工作时钟一样被 CSC 的 **CSC_RTC_EN**、**CSC_SLP_RTC** 和 **CSC_STP_RTC** 寄存器控制。

1 个时钟预分频器和 1 个时钟分频器用于 RTC 输入时钟。用户可通过 **RTC_CK_PDIV** 寄存器设置预分频值为 1 或 4096；通过 **RTC_CK_DIV** 寄存器设置分频值为 1/2/4/8。

[注释]: 通常内部时钟频率需比模块工作时钟慢至少 1/2。

21.6. 中断和事件

RTC 模块中有 1 种信号 **INT_RTC**。**INT_RTC** 发送到外部中断控制器 (EXIC) 作为中断事件。

21.6.1. RTC 中断控制和状态

中断标识是用于中断服务程序 (ISR) 流控制的。通常, 这些中断标志被硬件置起, 在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位, 用户可以选择使能或禁用。中断全局使能位 **RTC_IEA** 用于使能和禁用该模块的所有中断源。

21.6.2. RTC 中断标志

通常, 这些中断标志被硬件置起, 被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

- **ALMF**

RTC 报警匹配中断标志是(**RTC_ALMF**), 相关的中断使能寄存器位是 **RTC_ALM_IE**。

- **PCF**

RTC 循环中断标志是(**RTC_PCF**), 相关的中断使能寄存器位是 **RTC_PC_IE**。通常用户可配置定时器输入分频器来设置定时器输入时钟为 1Hz, 然后用户就可通过循环中断触发每秒执行一次循环工作。

- **TSF**

RTC 时间戳中断标志是(**RTC_TSF**), 相关的中断使能寄存器位是 **RTC_TS_IE**。它表明检测到了一次时间戳事件。

- **TOF**

RTC 定时器计数溢出中断标志是(**RTC_TOF**), 相关的中断使能寄存器位是 **RTC_TIE**。

- **RCRF**

RTC 重载或捕获标志是(**RTC_RCRF**)。该标志会在 **RTC_RLR** 寄存器完成重载、**RTC_CAP** 寄存器完成软件捕获或 **RTC_ALM** 寄存器值更新允许标志时置起。相关的中断使能寄存器位是 **RTC_RCR_IE**。

21.7. 寄存器保护和锁定

RTC 复位之后, 除了 **RTC_STA**、**RTC_KEY** 这两个寄存器以外, 所有的 RTC 寄存器将会处于写保护状态。通过向 **RTC_KEY** 寄存器写 **0xA217** 值即可解除寄存器保护, 并时刻处理控制。相对地, 写其他除 **0xA217** 以外的值将会使寄存器处于保护状态。读 **RTC_KEY** 寄存器的值以获得寄存器的状态是被保护 (=1) 还是未保护 (=0)。

它还提供了热复位后的寄存器锁定功能。向 **RTC_LOCK** 寄存器写值 **0x712A** 可锁定除 **RTC_STA**、**RTC_KEY** 寄存器外的写操作。当寄存器被锁定, 直到硬件冷复位, 寄存器不能通过系统热复位事件改变。写 **0x712A** 以外的值是无意义的, 读 **RTC_LOCK** 寄存器的值可得知寄存器是被锁定 (=1) 还是未锁定 (=0)。

比较特殊的硬件功能控制, **CSC_RTC_EN**、**CSC_SLP_RTC**、**CSC_STP_RTC** 和 **PW_WKSTP_RTC** 是被 **RTC_LOCK** 寄存器控制锁定的。当 **RTC_LOCK** 寄存器被设置锁定, 这些寄存器不能通过热复位复位。

参照系统复位章的“[寄存器保护和锁定](#)”节的表格和描述以获取更多信息。

21.8. RTC 功能控制

21.8.1. RTC 报警

RTC 通过 **RTC_ALM_EN** 位支持和使能报警功能。**RTC_ALM** 寄存器设置 RTC 报警比较值, 该寄存器在 **RTC_ALM_EN** 被禁用时可更新值。当 **RTC_ALM_EN** 被禁用, 引脚会置起 **RTC_RCRF** 标志以提醒软件, 然后软件就可更新 **RTC_ALM** 寄存器值。

当 RTC 定时器值等于 **RTC_ALM** 值时, RTC 报警标志 (**RTC_ALMF**) 会被置起, 若 **RTC_ALM_IE**=1 时还会产生中断。

21.8.2. RTC 时间戳

RTC 通过外部输入支持时间戳功能。用户可通过 **RTC_TS_TRGS** 寄存器选择上升、下降或双沿输入触发。

当匹配到外部输入信号, RTC 时间戳标志(**RTC_TSF**)会被置起, 若 **RTC_TS_IE**=1 时还会产生中断。

21.8.3. RTC 定时器捕获和重载

RTC 可捕获 32 位定时器值到 **RTC_CAP** 寄存器，或从 **RTC_RLR** 寄存器重载值到 32 位定时器中。通过使用 **RTC_RLR** 和 **RTC_CAP** 寄存器，一共有 4 种控制模式：直接捕获、延时捕获、强制重载、自动重载。

RTC 通过 **RTC_RCR_MDS** 位选择重载或捕获控制模式。若选择直接捕获或延时捕获模式，RTC 计数器值会在软件捕获事件(**RTC_RC_START=1**)或硬件时间戳事件发生时捕获进 **RTC_CAP** 寄存器中；若选择强制重载模式，RTC 定时器计数器会在 **RTC_RLR** 被写入时被 **RTC_RLR** 寄存器值更新；当选择自动重载模式，RTC 定时器计数器会在 RTC 定时器上溢时被 **RTC_RLR** 寄存器值更新。

当 **RTC_RLR** 寄存器重载完成，**RTC_CAP** 寄存器软件捕获完成或 **RTC_ALM** 寄存器值允许更新时，**RTC_RCRF** 标志会被置起，若 **RTC_RCR_IE=1** 时还会产生中断。

21.8.4. RTC 输出

RTC_OUT 输出可输出 RTC 内部信号到内部模块或外部引脚。**RTC_OUT** 可输出四种信号，**RTC_OUT_SEL** 寄存器可选择这四种信号，这四种信号分别是：定时器溢出信号切换输出、时间戳触发事件、定时器输入循环时钟信号 **CK_RTC_INT** 和报警比较输出事件。

[注释]:在 MG32F02A132/072 中，当 **RTC_OUT_SEL** 寄存器被设置为 1 且选择循环时钟信号时，**RTC_CK_PDIV** 和 **RTC_CK_DIV** 寄存器只能被设置为 0（被 1 分频）。

用户可通过 **RTCx_OUT_STA** 寄存器设置输出信号的初始状态。特殊的是，初始状态寄存器只有在 **RTCx_OUT_LCK** 寄存器被写 1 时被写入。

21.8.5. STOP 模式工作

若芯片在进入 **STOP** 模式之前设置 **CSC_RTC_EN** 和 **CSC_STP_RTC** 寄存器，RTC 就可在 **STOP** 模式下运行。当 RTC 工作于 **STOP** 模式下，APB 时钟会停止，且该模块会异步控制所有逻辑。

21.8.6. STOP 模式下唤醒

RTC 支持通过定时器上溢、定时器输入循环时钟和报警比较输出事件中把芯片从 **STOP** 模式中唤醒。一共有 3 种独立地唤醒使能位 **RTC_TF_WPEN**, **RTC_PC_WPEN** 和 **RTC_ALM_WPEN** 用于上述 3 种唤醒事件。

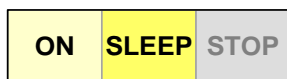
当芯片进入 **STOP** 模式且发生任何一个 RTC 唤醒事件时，RTC 将唤醒事件发送到电源控制器（PW）以作为系统唤醒事件。若相关控制寄存器被使能，该唤醒事件会使芯片唤醒。

[注释]:在 MG32F02A132/072 中，RTC 时钟源为外部 XTAL（CK_XOSC）时不支持从 STOP 模式下唤醒。

参照系统电源和中断章描述以获取更多关于唤醒事件和控制的信息。

22. ADC (模数转换器)

22.1. 简介



The module can be running in ON and SLEEP modes only.

该芯片内嵌 1 个含有 12 位逐步逼近式 ADC（模拟转数字转换器），1 个可增益 1~4 的 PGA（可编程增益放大器）和用于输出码控制的数字逻辑的 ADC0 模块。它支持可配置的包含 16 条外部和 4 条内部源的多路复用通道。模数转换可在单次、持续、单循环扫描或持续循环扫描模数下进行。

注释：(x = 模块, n, p= 输入通道标号)会被用于该章的寄存器、信号和引脚/端口描述中。[EX]: **ADCx_CONV_MDS**, **ADCx_SUMn** ~ x 代表模块标号；n 代表通道标号。

22.2. 特性

- 可设置分辨率：12/10/8 位
- 可设置采样时间
- 支持通过外部引脚、内部事件和软件位自动采样和触发
- 输出码数据对齐左对齐/右对齐
- 带旁路选项的内置输入缓冲
- 可编程增益：1~4
- 在采样结束、转换结束、扫描转换结束后产生中断
- 支持电压窗口监测和输出码限制
 - 2 级可编程窗口阈值
- 内置 3 通道独立硬件累加器用于 ADC 输出码
- 支持单次/通道扫描/环路扫描
- ADC 可使用 DMA 传输
- 支持 Wait 模式
- 防止低频率下 ADC 溢出

❖ MG32F02A132/072

- 12 位 400Ksps 的 SAR ADC
- 提供 16 条外部通道和 4 条内部通道输入
 - 内部通道源：VBG, VSSA, DAC 输出, ADC 参考电压
- 支持自校准以减少转换误差
- 支持 Auto-off 模式
 - 除启动转换的过程外 ADC 会自动关闭

❖ MG32F02A032

- 12 位 800Ksps 的 SAR ADC
- 提供 12 条外部通道和 4 条内部通道输入
 - 内部通道源：VBG, VSSA, LDO VR0 输出, ADC 参考电压

22.3. 配置

22.3.1. 芯片配置

下面的表格展示了芯片 ADC 模拟输入通道配置。

表 22-1. ADC 配置

芯片	封装	ADC 模块模拟输入			
		ADC_I[0..3]	ADC_I[4..7]	ADC_I[8..11]	ADC_I[12..15]
MG32F02A132AD80	LQFP80	V	V	V	V
MG32F02A132AD64	LQFP64	V	V	V	V
MG32F02A072AD64	LQFP64	V	V	V	V
MG32F02A072AD48	LQFP48	V		V	V
MG32F02A032AD48	LQFP48	V		V	V
MG32F02A032AY32	QFN32	V		V	
MG32F02A032AT20	TSSOP20	ADC_I8 , ADC_I10			

注释 V :包含

22.3.2. 模块功能

下面的表格展示了 ADC 模块包含的功能。

表 22-2. ADC 模块功能

芯片	MG32F02A132 MG32F02A072	MG32F02A032	注释
	ADC0	ADC0	
模块功能			
ADC 位分辨率	12 位	12 位	
ADC 最大转换率	400Ksps	800Ksps	
外部输入通道	16	12	
内部输入通道	4	4	
差分输入模式	yes	-	默认单端输入模式
LDO VR0 输出作为 ADC 输入	-	yes	内部核心逻辑 LDO 输出
DAC 输出作为 ADC 输入	yes	-	内部 DAC 输出
PGA 带增益	1~4	1~4	输入缓冲带增益级别 1~4
可设置采样时间	0~255	0~255	采样时钟时间
通道扫描转换	yes	yes	
硬件累加	3	3	
电压窗口检测	yes	yes	带高/低阈值的检测码
输出码限制	yes	yes	抓住输出码或跳过输出码
码左/右对齐	yes	yes	输出码数据对齐
符号码转换	yes	-	ADC 输出无符号码
Wait 模式	yes	yes	
Auto-Off 模式	yes	-	
硬件校准	yes	-	校准增益/内部参考
DMA 请求	yes	yes	
DMA 传输数据包	32 位	32 或 16 位	16 位 ADC 码; 16 位通道数据

22.4. 控制块

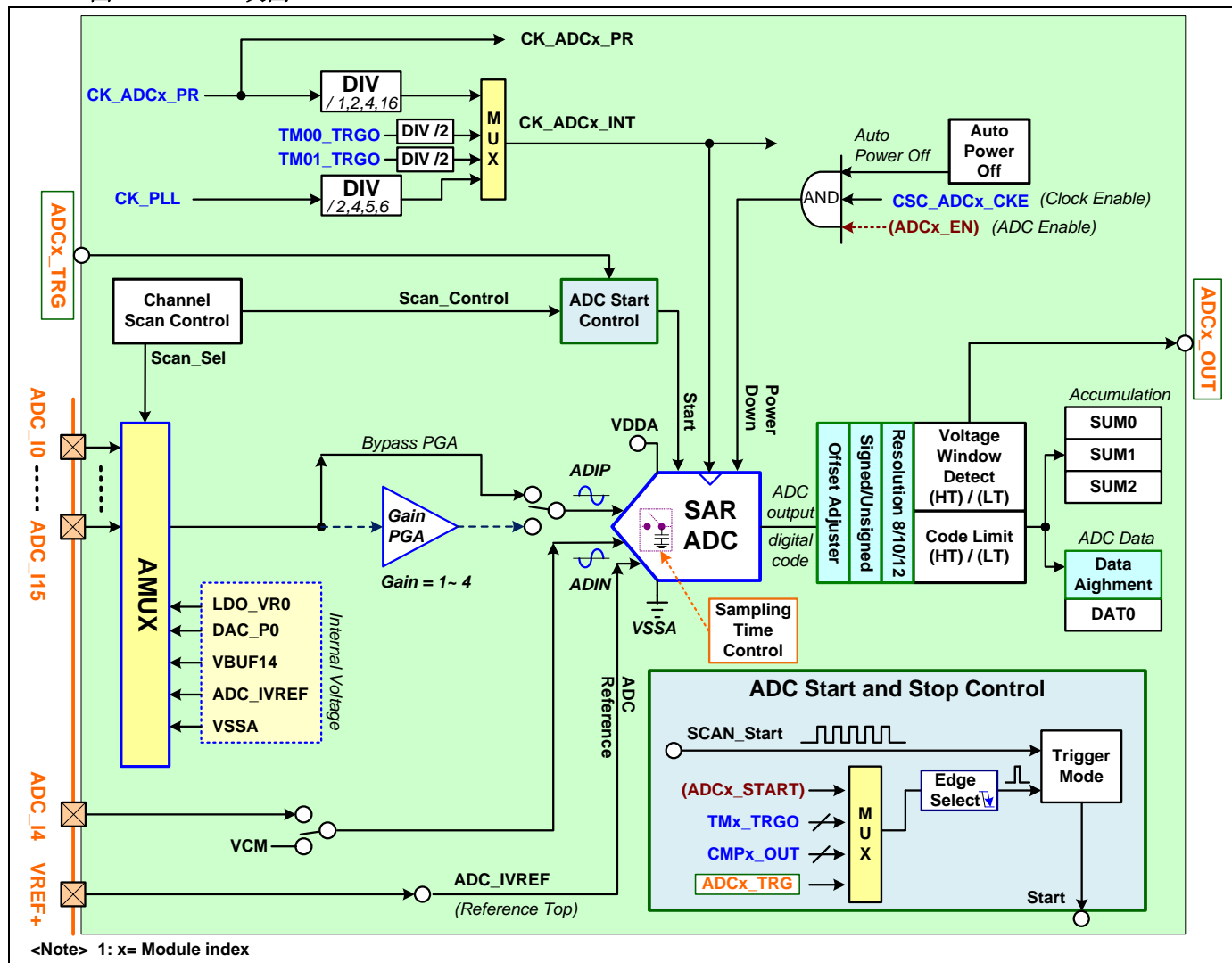
ADC 控制块由 1 个含有 16 个输入通道的模拟多路复用器 (AMUX)、1 个 400Ksps/12 位 SAR (逐步逼近寄存器) ADC、参考电压电路、ADC 转换触发启动控制块和改变扫描控制块组成。

[注释]: 在 MG32F02A132/072 中, 内部电压 LDO_VR0 不支持作为内部通道。

[注释]: 在 MG32F02A032 中, 内部电压 DAC_P0 不支持作为内部通道。

下面的表格展示了 ADC 控制块。

图 22-1. ADC 块图



22.5. IO 线

22.5.1. IO 信号

- **ADC_I[0..15]**
ADC 模拟信号输入通道 0~15。
- **ADCx_TRG**
外部触发起始输入信号用于 ADC 数据转换。
- **ADCx_OUT**
ADC 状态输出电压窗口比较结果信号。

22.5.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。使用的模拟输入 **ADC_I[0..15]** 引脚必须设置 IO 模式为

AIO 模式。参照用户手册 GPIO 章中“IO 模式”节以获取更多关于 IO 模式设置的信息。

每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“功能复用选择”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“引脚功能复用表”以获取更多信息。

22.6. 电源和时钟

22.6.1. ADC 电源控制

ADCx 模拟宏的工作电压 VDDA 来源于 IO 电源 **VDD** 引脚。ADC 模拟宏只能在 **ADCx_EN** 位设置为逻辑 1 时使能，当该位被设置为逻辑 0 时，ADC 模拟宏会被关闭。

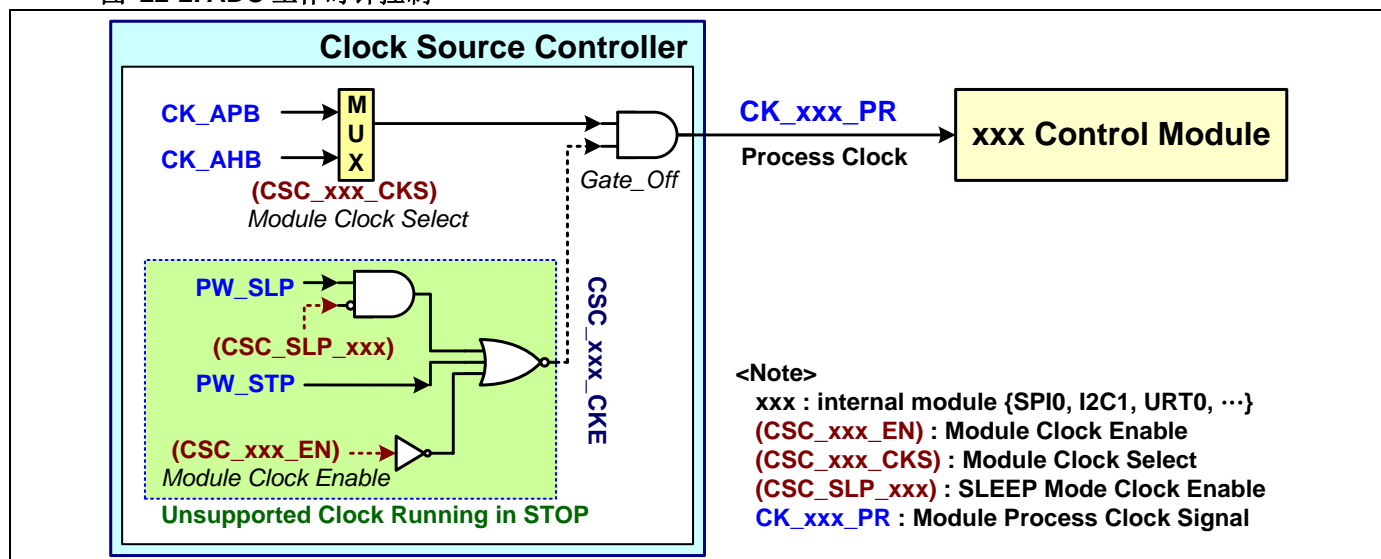
用户可在芯片进入 **SLEEP** 模式之前设置 **ADCx_EN** 为逻辑 1 或 0，并强制让 CPU 进入睡眠。然后芯片会进入 **SLEEP** 模式，ADC 模拟宏也会根据 **ADCx_EN** 寄存器的设置开启或关闭。ADC 模块不支持运行于 **STOP** 模式。参照系统电源章以获取更多信息。(n={0,1,2,3})

22.6.2. ADC 时钟控制

● 模块工作时钟

该模块工作时钟 **CK_ADCx_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 CSC(时钟源控制器)模块。该时钟可通过 **CSC_ADCx_EN** 寄存器使能并通过 **CSC_ADCx_CKS** 寄存器选择时钟源来自 APB 或 AHB。用户可以在芯片进入 **SLEEP** 模式之前通过设置 **CSC_SLP_ADCx** 寄存器规划在 **SLEEP** 模式下是否让 ADC 时钟继续运行。参照系统时钟章以获取更多信息。

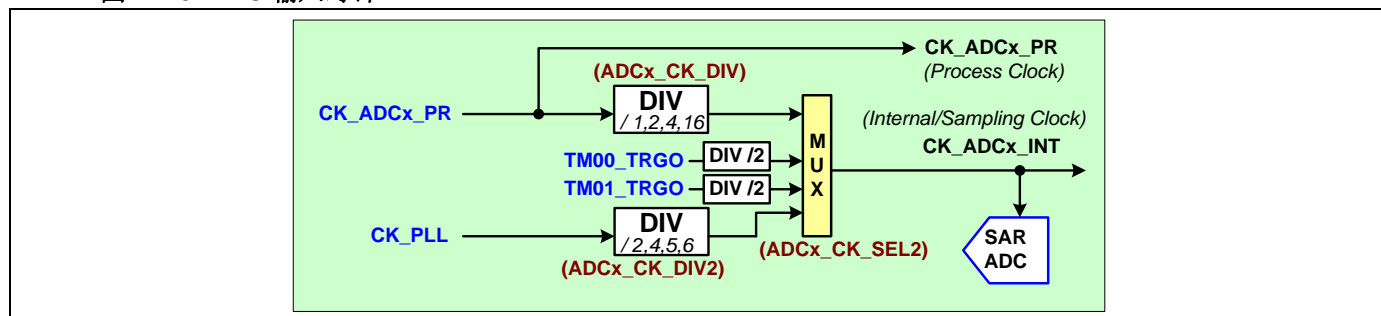
图 22-2. ADC 工作时钟控制



● 模块内部时钟

下面的图表展示了 ADC 输入时钟。

图 22-3. ADC 输入时钟



ADC 最快转换速度为 400Ksps。ADC 内部时钟或转换采样时钟可通过 **ADCx_CK_SEL2** 寄存器从模块工作时钟(AHB 时钟 **CK_AHB**, APB 时钟 **CK_APB**)、定时器触发输出信号(**TM00_TRGO** 或 **TM01_TRGO**)或 PLL 输

出时钟 **CK_PLL** 中选择。ADC 转换时钟不能超过 12 MHz。

该模块工作时钟可通过 **ADCx_CK_DIV** 寄存器设置被 1/2/4/16 分频作为内部时钟；**CK_PLL** 时钟可通过 **ADCx_CK_DIV2** 寄存器设置被 2/4/5/6 分频作为内部时钟。

[注释]: 通常内部时钟频率需比模块工作时钟慢至少 1/2。

22.7. 中断和事件

ADC 模块中有 2 种信号 **INT_ADC**, **RST_ADC**。**INT_ADC** 发送到外部中断控制器 (EXIC) 作为中断事件；**RST_ADC** 发送到复位源控制器作为复位事件。

22.7.1. ADC 中断和复位事件

- 中断事件

INT_ADC 信号发送到外部中断控制器 (EXIC) 作为中断事件。这些中断标志用于中断服务程序 (ISR) 流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **ADCx_IEA** 用于使能和禁用该模块的所有中断源。

- 复位事件

RST_ADC 信号发送到复位源控制器作为热复位或冷复位事件。这些复位事件可通过设置 RST 寄存器使能复位芯片。

参照 RST 章的描述以获取更多关于复位事件和控制的信息。

22.7.2. ADC 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

- **ESMPF**

ADC 采样结束标志是(**ADCx_ESMPF**)，该标志在采样结束时被置起，相关的中断使能寄存器位是 **ADCx_ESMP_IE**。

- **E1CNVF**

ADC 单次转换结束标志是(**ADCx_E1CNVF**)，该标志在通道转换结束时置起，新的数据结果可在 **ADCx_DAT0** 寄存器中使用。当清除该标志，也会清除 **ADCx_DAT0** 标志并准备接受下一个数据。相关的中断使能寄存器位是 **ADCx_ESCNV_IE**。

- **ESCNVF**

ADC 通道扫描转换结束标志是(**ADCx_ESCNVF**)，该标志在序列通道扫描转换结束时置起，相关的中断使能寄存器位是 **ADCx_ESCNV_IE**。

- **OVRF**

ADC 转换溢出事件标志是(**ADCx_OVRF**)，当清除该位，也会清除 **ADCx_DAT0** 寄存器的 **ADCx_DAT0_OVRF** 标志和 **OVRF** 标志。相关的中断使能寄存器位是 **ADCx_OVR_IE**。

- **WDLF / WDIF / WDHf**

ADC 电压窗口监测外部低、内部和外部高事件标志是 (**ADCx_WDLF**, **ADCx_WDIF** 和 **ADCx_WDHf**)，相关的中断使能寄存器位分别是 **ADCx_WDL_IE**, **ADCx_WDI_IE** 和 **ADCx_WDH_IE**。

- **SUMOF**

ADC 数据和-0,1,2 累加上溢或下溢标志是(**ADCx_SUMOF**)，当清除该位，也会清除全部 **ADCx_SUMn_OF** 和 **ADCx_SUMn_UF** 标志(n=0~2)。相关的中断使能寄存器位是 **ADCx_SUMO_IE**。

[注释]: **ADCx_SUMn_UF** 不支持于 MG32F02A032。

- **SUMCF**

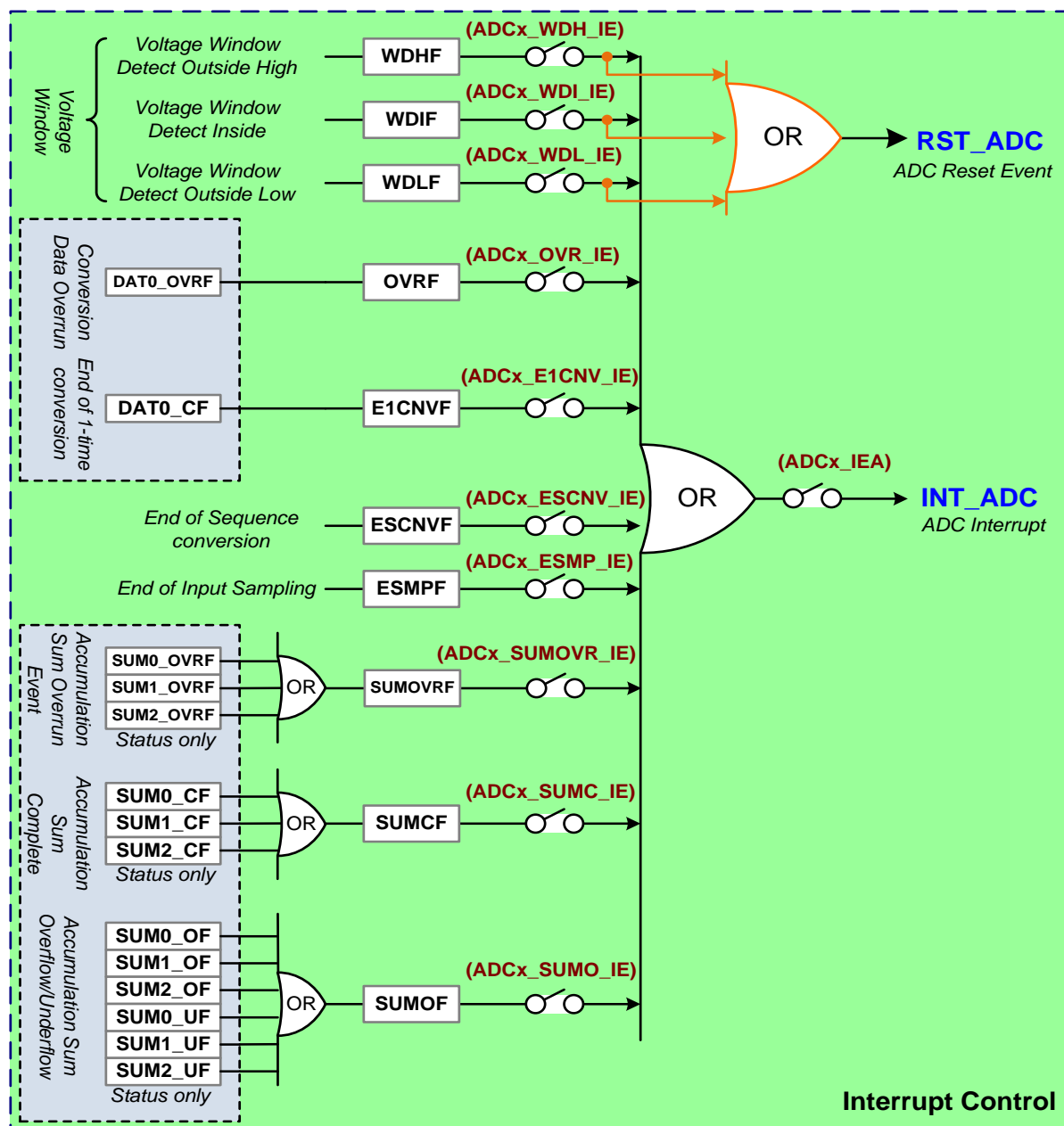
ADC 数据和-0,1,2 累加完成标志是(**ADCx_SUMCF**)，当清除该位，也会清除全部 **ADCx_SUMn_CF** 标志(n=0~2)。相关的中断使能寄存器位是 **ADCx_SUMC_IE**。

- **SUMOVRF**

ADC 数据和-0,1,2 寄存器过载标志是(**ADCx_SUMOVRF**)，当清除该位，也会清除全部 **ADCx_SUMn_OVRF** 标志(n=0~2)。相关的中断使能寄存器位是 **ADCx_SUMOVR_IE**。

下面的图表展示了 ADC 中断控制块。

图 22-4. ADC 中断控制



<Note> 1: x= Module index

22.8. ADC 操作

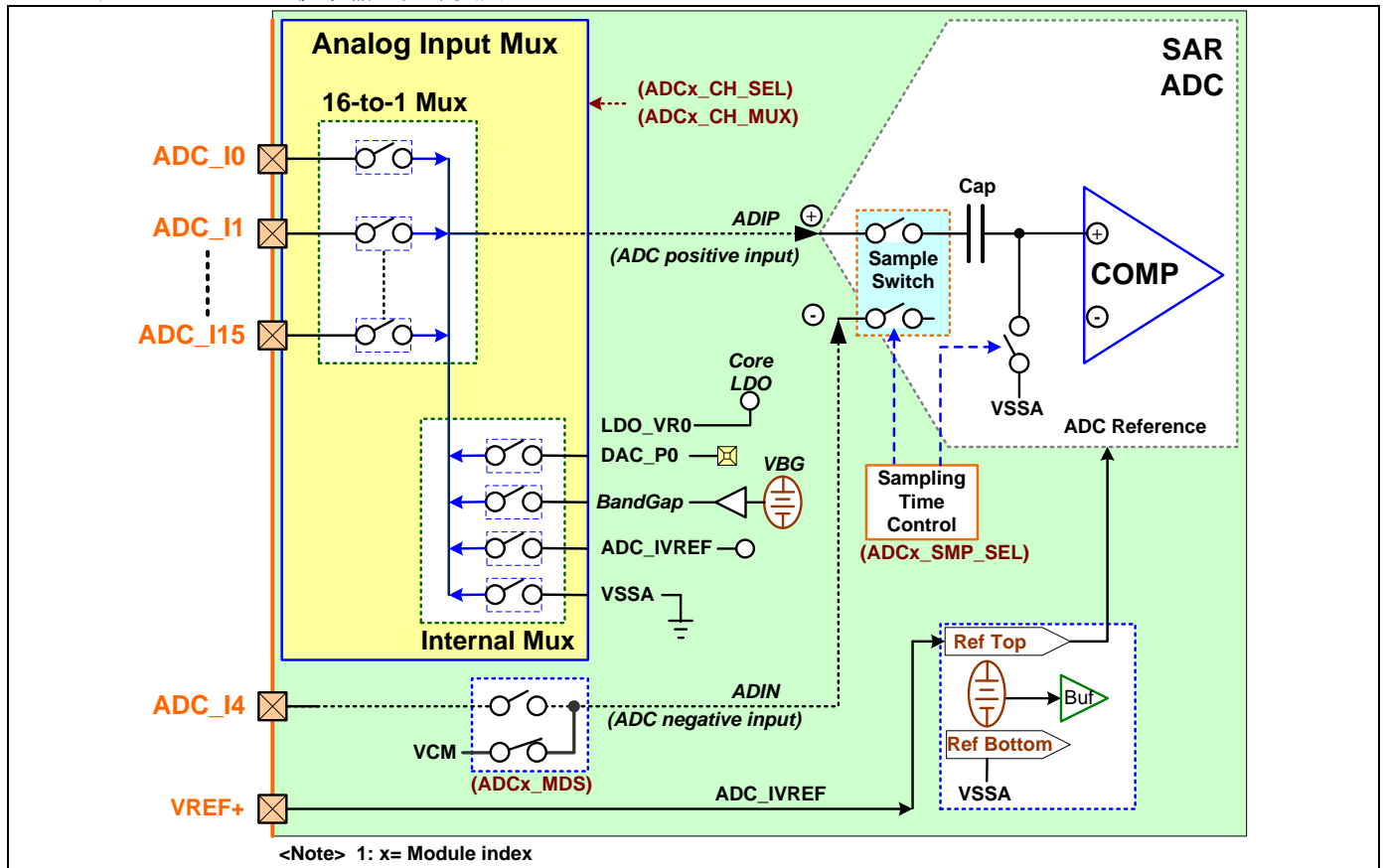
22.8.1. ADC 输入信号

模拟多路复用器（AMUX）用于选择 ADC 的输入，在单端模式下所有的输入引脚都可被测量。AMUX 可被 **ADCx_CH_SEL** 和 **ADCx_CH_MUX** 寄存器设置。被选择的引脚会相对于 VSSA（内部地）进行测量。

[注释]: 在 MG32F02A132/072 中, 内部电压 LDO_VR0 不支持作为内部通道。

[注释]: 在 MG32F02A032 中, 内部电压 DAC_P0 不支持作为内部通道。

图 22-5. ADC 和模拟输入多路复用器



- **ADC 功能的 I/O 引脚**

用于 A/D 转换器的模拟输入引脚还具有用于数字输入和输出功能。为了提供合适的模拟性能，使用 ADC 的引脚需要禁用数字输出—将端口引脚置仅输入模式即可。此外，当模拟信号被用于 **ADC_I[15:0]** 引脚且不需要将此引脚作为数字输入时，软件可以在 **PA_IOMn** ($n=\{0\sim15\}$) 寄存器中将相应的引脚设置成 AIO 模式来关闭数字输入缓冲区来降低功耗。

- **ADC 从 ADC_IVREF 和 VSSA 输入**

ADC_IVREF 是从 **+VREF** 引脚输入的 ADC 内部参考电压；**VSSA** 是 ADC 宏的内部地。通常，用户可输入这两个电压来校准 ADC 失调错误和增益错误。

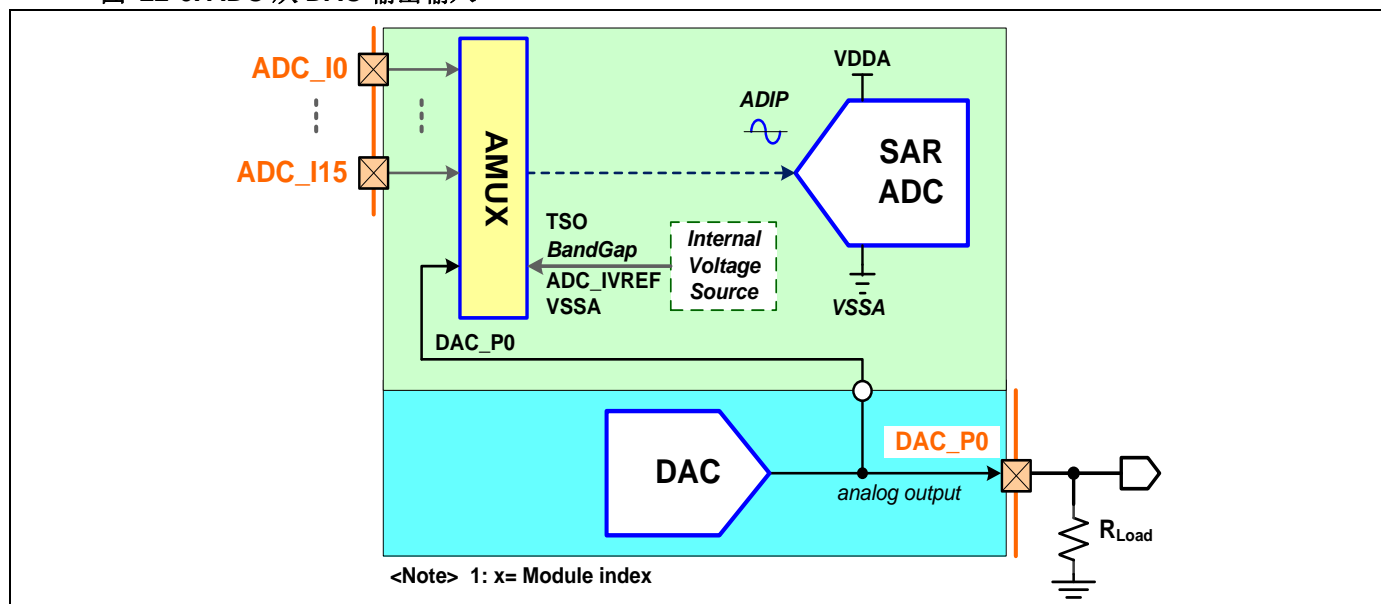
- **ADC 从 DAC 输入**

用户可设置 **ADCx_CH_MUX** 寄存器选择来源于内部 DAC 输出的模拟输入。使用 ADC 来测量 DAC 输出是很有用的办法。由于 DAC 输出为电流模式，需要外加 1 个负载电阻 R_{Load} 来把电流输出转换成电压。

[注释]: 在 MG32F02A032 中, 内部电压 DAC_P0 不支持作为 ADC 输入源。

下面的图表展示了从 DAC 输出到 ADC 输入的内部连接。

图 22-6. ADC 从 DAC 输出输入



● ADC 输入通道定义

下面的表格展示了用于 ADC 通道定义的寄存器设置。

[注释]: 在 MG32F02A132/072 中, 内部通道-8 是未定义且不支持 LDO VR0 输入。

[注释]: 在 MG32F02A032 中, 内部通道-2 是未定义且不支持 DAC_P0 输入。

表 22-3. ADC 通道定义

外部/内部通道选择			外部/内部通道定义	芯片支持	
CH_SEL	CH_MUX	Number	CH_SEL = EXT (External)	MG32F02A132 MG32F02A072	MG32F02A032
外部	0x0	0	外部通道 0 (ADC_I0 引脚输入)	V	V
	0x1	1	外部通道 1 (ADC_I1 引脚输入)	V	V
	0x2	2	外部通道 2 (ADC_I2 引脚输入)	V	V
	0x3	3	外部通道 3 (ADC_I3 引脚输入)	V	V
	0x4	4	外部通道 4 (ADC_I4 引脚输入)	V	
	0x5	5	外部通道 5 (ADC_I5 引脚输入)	V	
	0x6	6	外部通道 6 (ADC_I6 引脚输入)	V	
	0x7	7	外部通道 7 (ADC_I7 引脚输入)	V	
	0x8	8	外部通道 8 (ADC_I8 引脚输入)	V	V
	0x9	9	外部通道 9 (ADC_I9 引脚输入)	V	V
	0xA	10	外部通道 10 (ADC_I10 引脚输入)	V	V
	0xB	11	外部通道 11 (ADC_I11 引脚输入)	V	V
	0xC	12	外部通道 12 (ADC_I12 引脚输入)	V	V
	0xD	13	外部通道 13 (ADC_I13 引脚输入)	V	V
	0xE	14	外部通道 14 (ADC_I14 引脚输入)	V	V
	0xF	15	外部通道 15 (ADC_I15 引脚输入)	V	V
内部	0x0	0	内部通道 0 (VSSA 电压)	V	V
	0x1	1	内部通道 1 (IVREF 电压源)	V	V
	0x2	2	内部通道 2 (DAC_P0 DAC 电压输出)	V	
	0x3	3	内部通道 3 (VBG 电压源)	V	V
	0x4	4	为内部使用保留		
	0x5	5	未定义通道 (输入多路复用输出 Hi-Z)		
	0x6	6	未定义通道 (输入多路复用输出 Hi-Z)		
	0x7	7	未定义通道 (输入多路复用输出 Hi-Z)		
	0x8	8	内部通道 8 (LDO VR0 电压源)		V
	0x9	9	未定义通道 (输入多路复用输出 Hi-Z)		
	0xA	10	未定义通道 (输入多路复用输出 Hi-Z)		

	0xB	11	未定义通道(输入多路复用输出 Hi-Z)		
	0xC	12	未定义通道(输入多路复用输出 Hi-Z)		
	0xD	13	未定义通道(输入多路复用输出 Hi-Z)		
	0xE	14	未定义通道(输入多路复用输出 Hi-Z)		
	0xF	15	为内部使用保留		

CH_MUX : (ADCx_CH_MUX) ~通道多路复用选择

CH_SEL : (ADCx_CH_SEL) ~外部或内部通道选择

22.8.2. 单端和差分模式

ADC 支持单端和差分两种工作模式，用户可以自行根据应用为 ADC 的工作模式通过 **ADCx_MDS** 寄存器选择单端或差分模式。当选择差分模式时，**ADCI_4** 引脚作为负极输入，而正极输入可以选择其他输入引脚。当选择单端模式时，负极引脚会被短路到共模电压(**VCM**)，同时，正极输入可以从任何输入引脚进行输入。

ADC 在单端或差分模式下可以将 ADC 输出转换为无符号或有符号码。参照“[有符号码转换器](#)”节以获取更多信息。

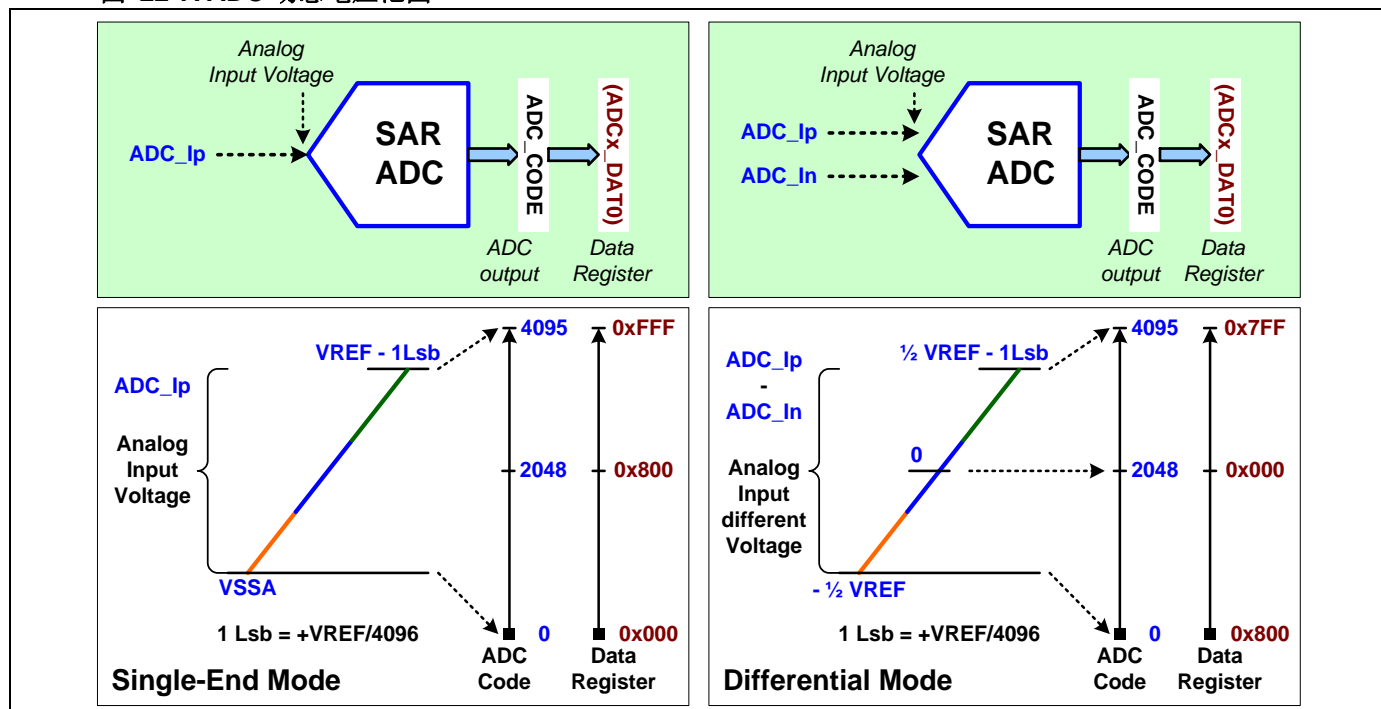
[注释]: 差分模式不支持于 MG32F02A032。

22.8.3. 输入动态电压范围和码范围

单端模式下，ADC 输入动态电压范围为从 **VSSA**（非常接近 0 伏）到 **VREF(+VREF 引脚电压)**；差分模式下则是从 $-\frac{1}{2}VREF$ 到 $+\frac{1}{2}VREF$ 。

下面的图表展示了单端模式和差分模式的动态电压范围。

图 22-7. ADC 动态电压范围

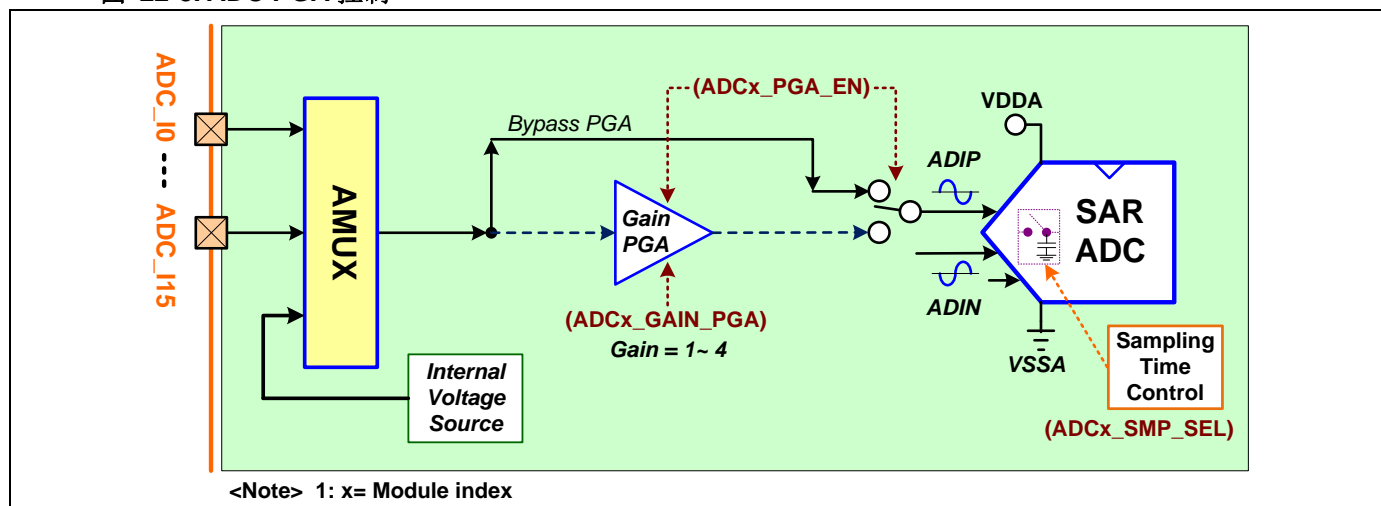


22.8.4. ADC 增益控制

● PGA 增益

ADC 模块内建 1 个 PGA（可设置增益放大器）以加强 ADC 输入质量并放大输入电压范围。PGA 增益在测量低电平信号时有所帮助。提高 PGA 增益，可减低输入参考噪音。用户可通过 **ADCx_PGA_EN** 寄存器使能 PGA，通过 **ADCx_GAIN_PGA** 寄存器设置增益值 1 到 4。

图 22-8. ADC PGA 控制



PGA 增益值根据公式设置:

$$\text{PGA Gain} = \frac{\text{ADCx_GAIN_PGA} * 3}{63} + 1$$

● PGA 失调校准

ADCx_OFFT_PGA 寄存器用于调整和校准 PGA 输入失调。用户可在 **ADCx_CAL_POFFT** 寄存器中使能 PGA 失调校准。当使能时，用户可设置 0x20 值到该寄存器中，还可从 **ADCx_POF** 寄存器中获得 PGA 失调校准状态位。持续增加或减少该值直到 PGA 失调校准状态位改变，设置最后的设置值到 **ADCx_OFFT_PGA** 寄存器中，PGA 输入失调校准即完成。

● PGA 偏置控制

ADCx_BUF_BIAS 寄存器用于 PGA 偏置电流设置。该寄存器默认设置为 0。根据用户应用，用户可设置为 1 来增加 PGA 的偏置电流和输入带宽，因此，可以提高 PGA 的运算速度，以提高输入 ADC 信号的频率。

22.8.5. ADC 转换

ADC 总的转换时间包括模拟信号采样时间和模拟信号转数字码的时间。ADC 总的转换时间需要最少 30 个 ADC 采样时钟。

● ADC 采样时间

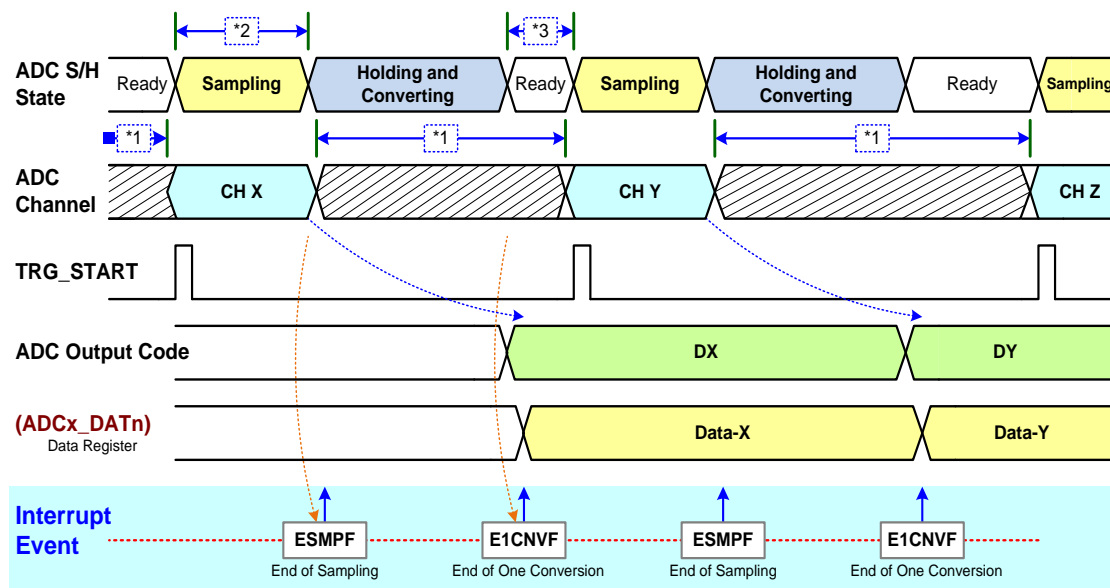
关于输入信号质量和转换速率的问题，用户可在 **ADCx_SMP_SEL** 寄存器中调整 ADC 采样时间。通常，若转换速率和信号带宽在实际应用中是合理有效的，增加 ADC 采样时间可获得更稳定的电压和更好的 ADC 表现。

用户可通过 **ADCx_SMP_SEL** 寄存器设置 ADC 采样时间 0~255。这也会增加总转换时间。

● ADC 转换时序

用户可根据模拟输入信号的频率选择合适的转换速度。当 ADC 最大输入时钟是 12MHz，最高转换速率为 400Ksps；当 ADC 最大输入时钟是 24MHz，最高转换速率为 800Ksps。

图 22-9. ADC 转换序列



<Note-1> Software set ADC channel by setting (ADCx_CH_MUX).

<Note-2> Programmable sampling time by setting (ADCx_SMP_SEL) register .
sampling cycle = (ADCx_SMP_SEL) +4 ~ ADC sampling clock time

<Note-3> ADC ready and wait to change channel for next conversion trigger.

<Note-4> {X,Y,...,Z} indicate the enabled channels those are following CH(channel) and D(Data) .

22.8.6. ADC 校准

由于用户应用和环境、工作电压和温度都会影响 ADC 单元，因此 ADC 参考电压最高点、中点和最低点可被 ADC 校准进行调整。这 3 个参数可在 **CFG_ADCx_REFT**, **CFG_ADCx_REFM** 和 **CFG_ADCx_REFB** 寄存器中设置。这 3 个寄存器在芯片复位后会载入来自设置字节内存中的初始值。特别的是，**ADCx_REFT**, **ADCx_REFM** 和 **ADCx_REFB** 寄存器为只读。

通常用户可通过输入 **VSSA** 和 **ADC_IVREF** 电压,持续设置这些寄存器以校准 ADC 失调并校准增益错误，直到 ADC 输出码为有效值用于用户应用。

ADC 提供自动归零功能以在 ADC 输入参考地电平时强行将 ADC 输出码置零。用户可通过 **ADCx_CAL_AZEN** 寄存器使能该功能。

[注释]: ADC 校准功能不支持于 MG32F02A032。

22.8.7. SLEEP 模式下 ADC 工作

若 ADC 在 **SLEEP** 模式中被启动，它会消耗一点电力。在进入 **SLEEP** 模式之前用户可通过关闭 ADC 硬件 (**ADCx_EN=0**)以降低功耗。

若软件触发 ADC 在 **SLEEP** 模式下工作，ADC 会完成转换并置起 ADC 中断标志。当 ADC 中断使能 (**ADCx_E1CNV_IE** 或 **ADCx_ESCNV_IE**)被置起，ADC 中断会把 CPU 从 **SLEEP** 模式唤醒。

22.9. ADC 转换序列

22.9.1. ADC 转换设置和序列

- 准备 ADC 转换

使用 ADC 之前，用户需：

- 通过 **ADCx_EN** 位启动 ADC 硬件。
- 通过 **ADCx_RES_SEL** 位设置 ADC 分辨率。
- 通过 **ADCx_CK_SEL** 和 **ADCx_CK_DIV** 位设置 ADC 输入时钟。
- 通过 **ADCx_CH_SEL** 和 **ADCx_CH_MUX** 位选择模拟输入通道。
- 通过 **PA_IOMn** 寄存器设置选择的输入为仅模拟输入模式。(n={0~15})
- 通过 **ADCx_ALIGN_SEL** 位设置 ADC 结果对齐。

- ADC 输入通道选择

用户可设置 **ADCx_CH_SEL** 和 **ADCx_CH_MUX** 寄存器选择 ADC 输入通道源。

参照“[ADC 输入通道](#)”节以获取更多信息。

- ADC 转换模式

ADC 支持通过外部引脚、内部事件和软件位自动采样和触发。用户可设置 **ADCx_CONV_MDS** 和 **ADCx_TRG_CONT** 寄存器设置 ADC 转换模式。参照“[ADC 转换模式](#)”以获取更多信息。

- ADC 转换开始

当完成 ADC 转换设置，用户需设置 **ADCx_START_SEL** 寄存器设置 ADC 触发源并启动 ADC 转换。ADC 启动触发源包括（1）软件控制寄存器位 **ADCx_START**（2）外部触发引脚 **ADC0_TRG**（3）内部模块触发事件 **TM00_TRGO**, **CMP0_OUT**, **CMP1_OUT**, **TM01_TRGO**, **TM20_TRGO** 和 **TM36_TRGO**。

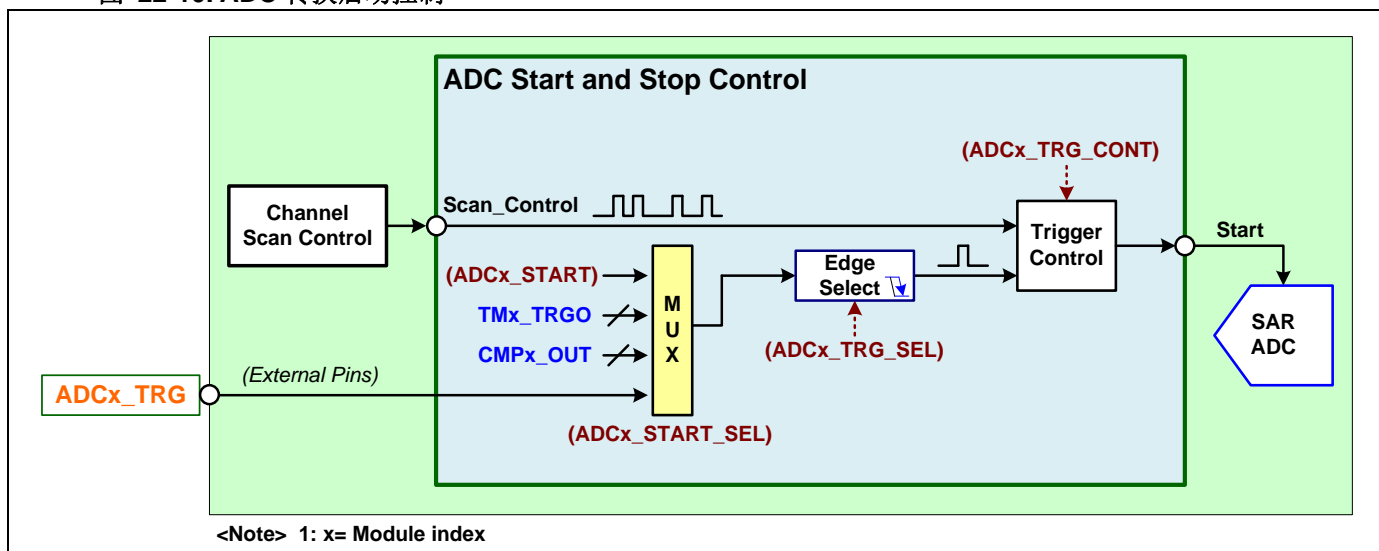
[注释]: 在 MG32F02A032 中, **TM20_TRGO** 不支持作为 ADC 启动触发信号。

用户可通过软件设置 **ADCx_START** 寄存器、定时器输出内部信号 **TMx_TRGO** 或模拟比较器结果输出内部信号 **CMPx_OUT** 触发 ADC 转换。

当选择软件寄存器触发, **ADCx_START** 寄存器会被通过软件设置启动 ADC 转换, 并在 ADC 转换完成后被硬件清除。

当选择内部触发信号触发, **ADCx_TRG_SEL** 寄存器用于设置上升沿、下降沿或双沿触发沿。

图 22-10. ADC 转换启动控制



22.9.2. ADC 转换暂停

在 ADC 转换的过程中,用户可设置 **ADCx_HOLD** 寄存器暂停 ADC 转换和清除 **ADCx_HOLD** 寄存器继续 ADC 转换。

22.9.3. ADC 转换过载

当 ADC 转换完成且前一个 ADC 数据还未被软件取出, ADC 转换数据会过载。用户可在发生过载前通过设置 **ADCx_OVR_MDS** 寄存器设置覆盖还是保留旧 ADC 数据。
此外用户可使能硬件“ADC 等待”功能避免转换过载。参照“[ADC 等待和自动关闭](#)”节以获取更多信息。

22.10. ADC 转换模式

ADC 支持单次、持续、单扫描、持续扫描、循环扫描 5 种转换模式。用户可设置 **ADCx_CONV_MDS** 和 **ADCx_TRG_CONT** 寄存器设置 ADC 转换模式。

表 22-4. ADC 转换模式控制

ADC 转换模式	ADC 寄存器		转换功能
	CONV_MDS	TRG_CONT	
单次模式	单次	禁用	单次启动触发后通过单个通道转换 1 个 ADC 数据后停止转换。
持续模式	单次	使能	单次启动触发后通过单个通道转换 1 个 ADC 数据后硬件自动开始下一次 ADC 转换而不停止。
单扫描模式	扫描	禁用	单次启动触发后通过单个选择的通道转换 1 个 ADC 数据后停止转换。下一次启动触发时, 则会自动跳转到下一个选择的通道进行 ADC 转换, 直到完成所有选择的通道。
持续扫描模式	扫描	使能	单次启动触发后通过多个选择的通道转换多个 ADC 数据, 在扫描完成所有选择的通道后停止转换。
循环扫描模式	循环	X	单次启动触发后, 硬件会对选择的通道一个一个进行 ADC 转换, 直到所有选择的通道完成, 然后硬件会自动开始下一次通道扫描, 而不会停止。

[注释]: 在 MG32F02A032 中, TM20_TRGO 不支持作为 ADC 启动触发信号。

22.10.1. 单次和持续转换

● 启动单次转换

用户需在 **ADCx_CONV_MDS** 和 **ADCx_TRG_CONT** 寄存器中选择单次转换。
现在, 用户可设置 **ADCx_START** 位开启模数转换。转换时间是被 **ADCx_CK_SEL** 和 **ADCx_CK_DIV** 位决定的。一旦转换完成, 硬件会自动清除 **ADCx_E1CNVF** 位, 置起中断标志 **ADCx_E1CNVF** 并同时载入 12 位转换结果进入 **ADCx_DAT0**(根据 **ADCx_ALIGN_SEL** 位)寄存器中。
如上描述, 中断标志当被 **ADCx_E1CNVF** 硬件设置, 表明转换完成。
有两种办法检查转换是否完成: (1) 始终软件轮询中断标志 **ADCx_E1CNVF**; (2) 通过设置位使能 ADC 中断, 然后 CPU 会在转换完成时跳入中断服务例程中。不管是(1)还是(2), **ADCx_E1CNVF** 标志需在下一转换开始之前用软件清除。

- 启动持续转换

用户需在 **ADCx_CONV_MDS** 和 **ADCx_TRG_CONT** 寄存器中选择持续转换。若用户通过 **ADCx_START_SEL** 寄存器选择自由运行或其他除手动模式设置软件位 **ADCx_START** 启动外的其他触发源，ADC 会持续进行转换直到 **ADCx_EN** 被清除或设置 ADC 为手动模式。

- 转换结果

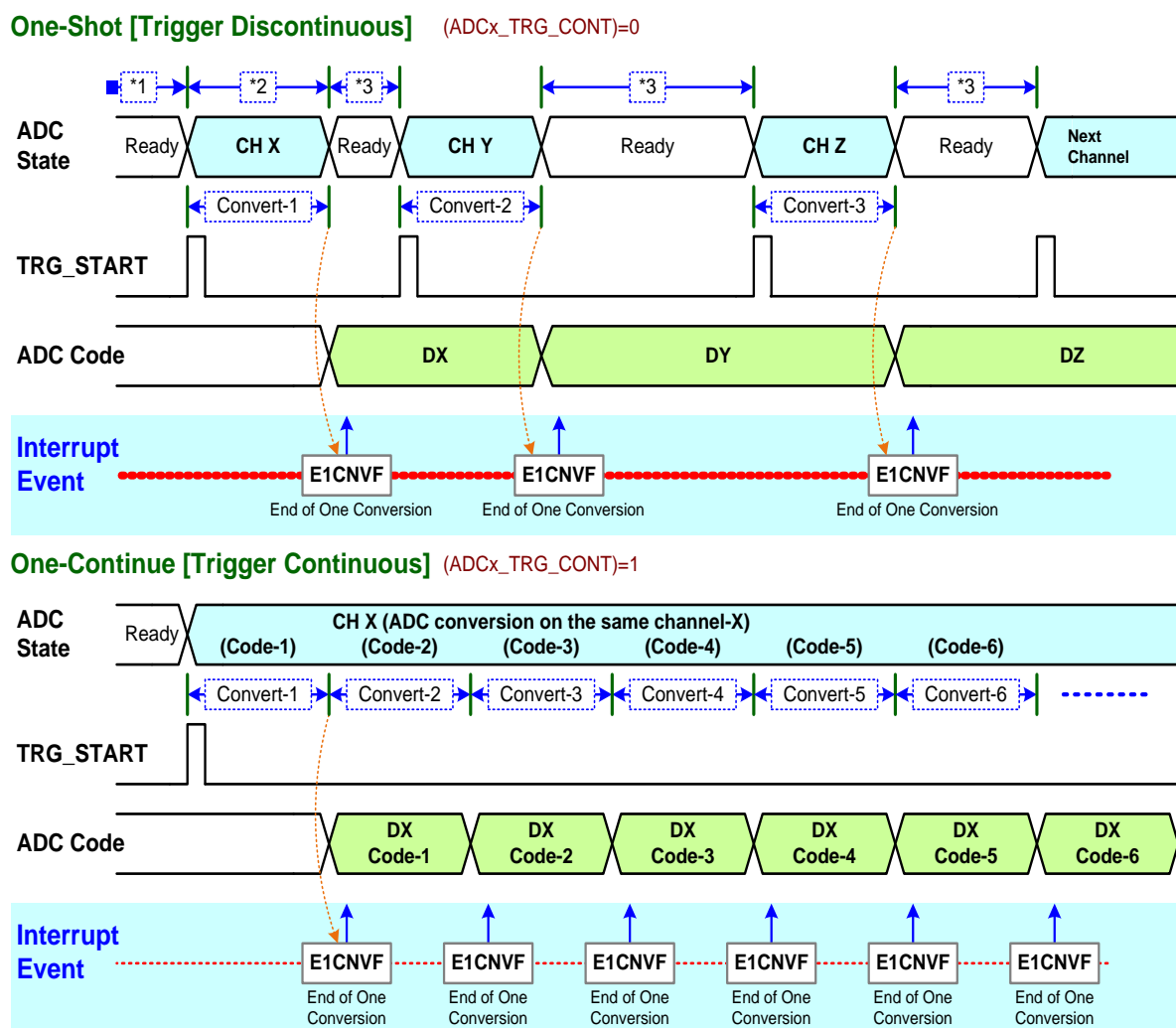
ADC 的转换结果会以左对齐或右对齐存入 16 位数据寄存器 **ADCx_DAT0** 中。

转换结束且 **ADCx_E1CNVF** 被置高后，ADC 转换完成并输出 ADC 码。对于单次转换，ADC 输出码为：

$$\text{ADC Code} = \frac{\text{VIN} * 4096}{\text{VREF}}$$

ADC 输出码会被 ADC 输出控制块调整并存入 **ADCx_DAT0** 寄存器中。参照“[ADC 输出控制](#)”节以获取更多信息。

图 22-11. ADC 转换模式-单通道



<Note-1> Software set ADC channel by setting (**ADCx_CH_MUX**) .

<Note-2> Input signal sample and hold for ADC conversion.

<Note-3> ADC ready and wait to change channel for next conversion trigger.

<Note-4> {X,Y,...,Z} indicate the enabled channels those are following CH(channel) and D(Data) .

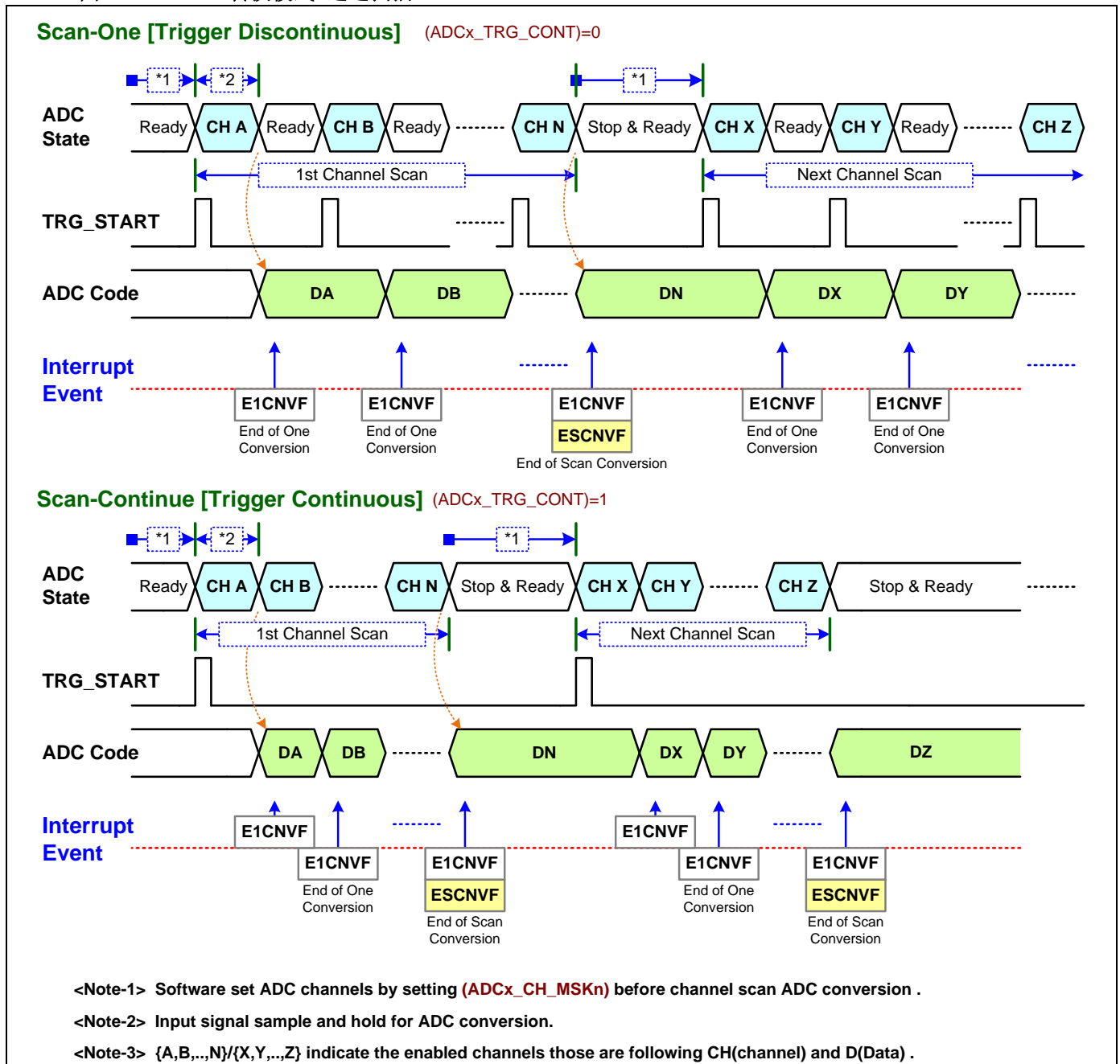
22.10.2. 通道扫描转换

用户可在 **ADCx_CONV_MDS** 和 **ADCx_TRG_CONT** 寄存器中选择单扫描模式或持续扫描转换模式。
ADCx_CH_MSK 寄存器设置 ADC 通道选择屏蔽用于序列通道扫描。当选择屏蔽，相关通道会在通道扫描循环时被屏蔽并禁用。

此外用户可设置 **ADCx_START** 位启动模数转换。在每个通道的单次转换完成后 **ADCx_E1CNVF** 标志会被置起，转换结果会在 **ADCx_DAT0** 寄存器中被找到，**ADCx_DAT_CH** 代表启动了的通道。

在每个使能的通道的单循环序列扫描完成后，ADC 转换会停止，**ADCx_ESCNVF** 标志被置起，用户可轮询该标志或如果中断使能位 **ADCx_ESCNV_IE** 为 1，通过中断触发运行固件服务。

图 22-12. ADC 转换模式-通道扫描



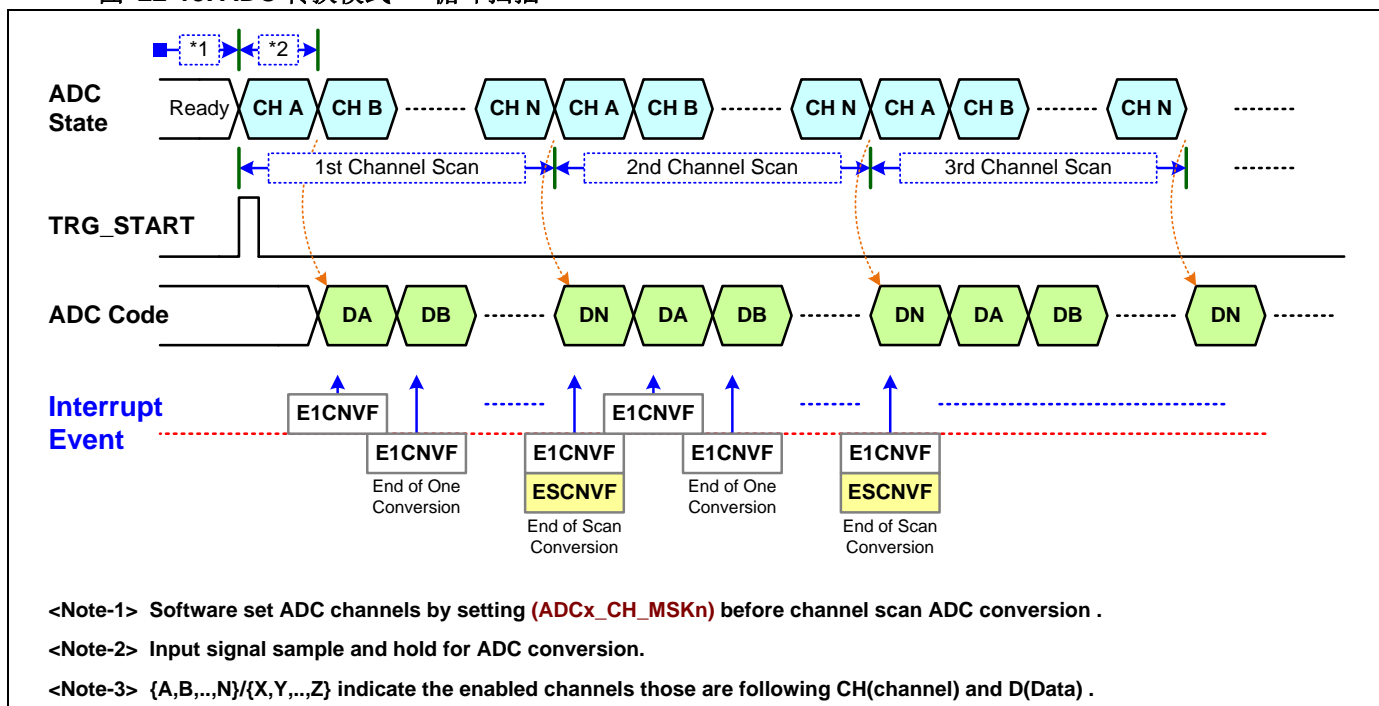
22.10.3. 扫描循环转换

用户可在 **ADCx_CONV_MDS** 寄存器中选择持续循环扫描转换。与单循环扫描转换相同，用户需设置 **ADCx_CH_MSK** 寄存器。

一样的，用户可设置 **ADCx_START** 位启动模数转换并进行后面的程序。单循环序列扫描完成后，ADC 转换不会停止，并根据定义的第一个通道开始继续转换数据，直到 **ADCx_START=0** 或 **ADCx_START_SEL** 寄存器中设置的其他停止触发启动信号源发生。

下面的图表展示了 ADC 序列转换硬件流控制。

图 22-13. ADC 转换模式 – 循环扫描



22.10.4. 输入通道改变

当选择单次模式，用户可在 ADC 数据转换完成时或 ADC 数据软件采样结束时改变输入通道复用器。

用户可在 **ADCx_ESCNV_IE** 寄存器中使能单次转换结束的中断。该芯片会在通道上的每次转换完成时置起标志 **E1CNVF (ADCx_E1CNVF)**，此时用户可改变输入通道。此外用户可在 **ADCx_ESMP_IE** 寄存器中使能 ADC 采样结束标志。该芯片会在采样结束时置起标志 **ESMPF (ADCx_ESMPF)**，此时用户可改变输入通道。

当选择通道扫描或循环扫描模式，硬件会自动改变输入通道到下一个输入通道。通过设置 **ADCx_CH_CHG** 寄存器，ADC 会在一次 ADC 数据转换完成后或 ADC 数据采样完成后改变输入通道多路复用器。

[注释]: **ADCx_CH_CHG** 寄存器不支持于 MG32F02A132/072。

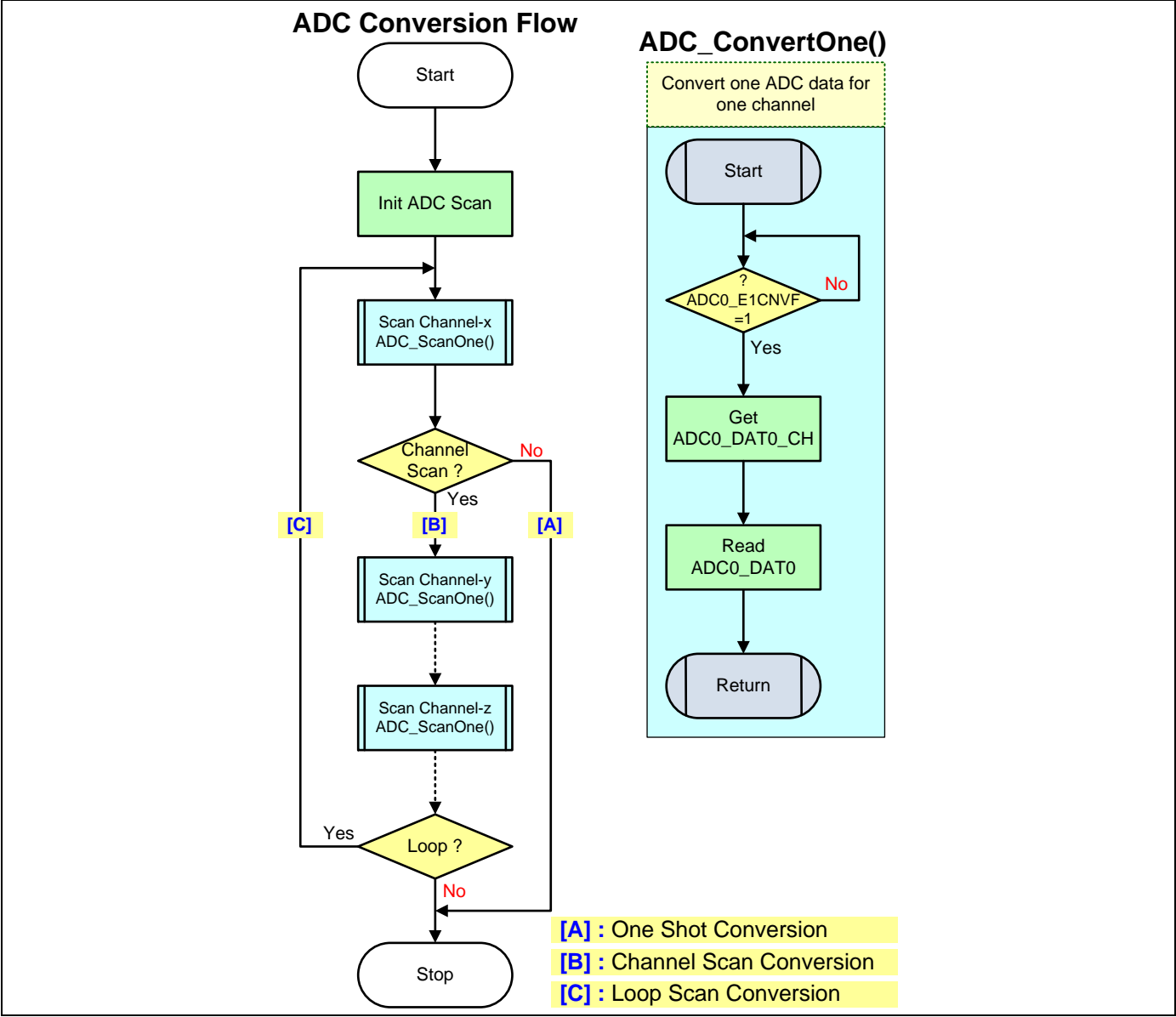
下一个输入通道标号是被序列通道号控制，序列通道号通过 **ADCx_CH_MSK** 寄存器定义。比如，通道 2~8 和 11~13 被屏蔽，那么通道扫描会按顺序扫描通道 0,1,9,10,14,15。

22.10.5. ADC 转换流

下面的图表展示了 ADC 转换流。[A]流代表单次 ADC 转换； [B]流代表通道扫描 ADC 转换； [C]流代表循环扫描 ADC 转换。

ADC_ConverOne()流程图为一次 ADC 转换数据控制的软件流。

图 22-14. ADC 转换流



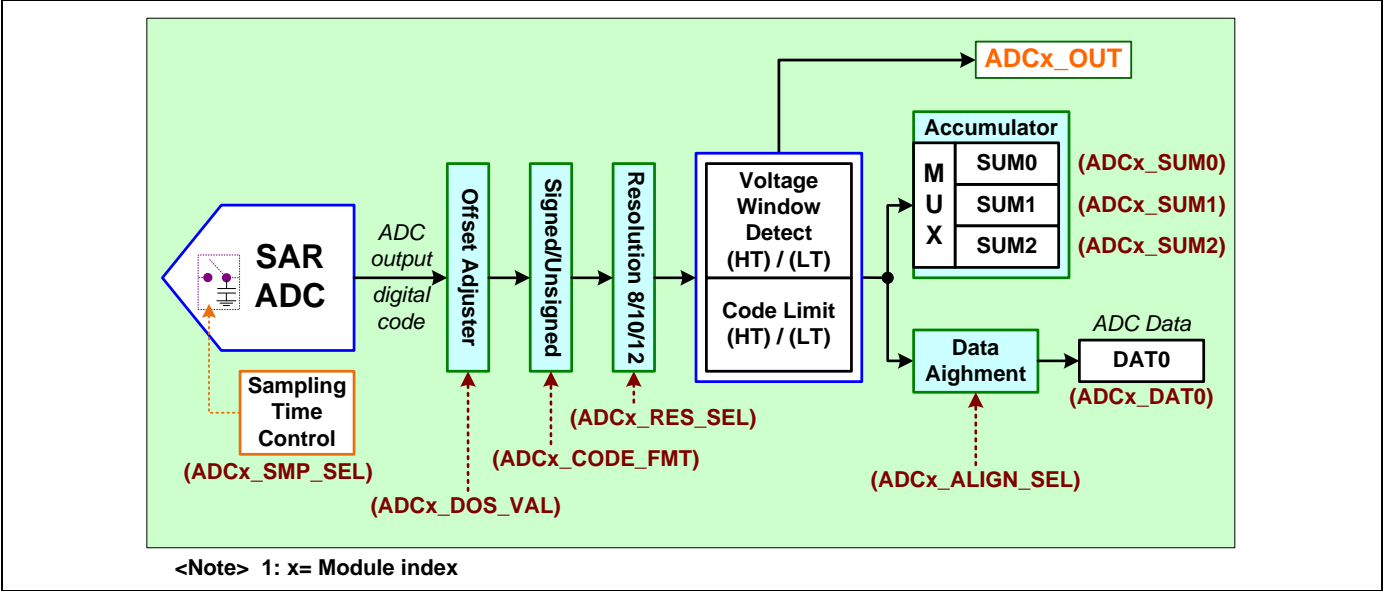
22.11. ADC 输出控制

当一次 ADC 转换完成，ADC 初始码会被产生并发送到包含了数字偏移调整器、符号码转换器、数字分辨率调整器、电压窗口检测器、码限制器和数据对齐调整器的 ADC 输出控制块中。

ADC 输出码会被 ADC 输出控制块调整并把转换结果数据存到 **ADCx_DAT0** 寄存器中。参照“[ADC 数据寄存器](#)”节以获取更多信息。

内嵌额外的累加器用于累加可设置数据数量的 ADC 数据并记录和到总和寄存器中。参照“[数据利](#)”节以获取更多信息。

图 22-15. ADC 输出控制块



22.11.1. 数字偏移调整器

ADC 内建 1 个数字码偏移调整器，用户可通过设置 **ADCx_DOS_VAL** 寄存器设置 2s 补码值用于代码偏移调整。因此 ADC 输出码可被设置值调整并输出到符号码转换器中。

表 22-5. ADC 数字偏移操作

数字偏移 操作输出	寄存器				
	ADCx_DOS_VAL				
ADC Code+15	0	1	1	1	1
ADC Code+14	0	1	1	1	0
....				
ADC Code+2	0	0	0	1	0
ADC Code+1	0	0	0	0	1
ADC Code+0	0	0	0	0	0
ADC Code-1	1	1	1	1	1
ADC Code-2	1	1	1	1	0
....				
ADC Code-15	1	0	0	0	1
ADC Code-16	1	0	0	0	0

ADCx_DOS_VAL Msb = 符号位

22.11.2. 符号码转换器

在应用中, ADC 码可通过 **ADCx_CODE_FMT** 寄存器设置调整为无符号码用于单端模式或有符号码用于差分模式。

[注释]: 差分模式和 **ADCx_CODE_FMT** 寄存器不支持于 MG32F02A032。

- 单端模式的码定义

下面的表格展示了单端模式的有符号码和无符号码格式的 ADC 码。

表 22-6. ADC 单端模式数据格式定义

格式	ADC 输入 电平 (LSB)	ADC 有符号/无符号输出码											
		B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
无符号	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	1
											
	2047	0	1	1	1	1	1	1	1	1	1	1	1
	2048	1	0	0	0	0	0	0	0	0	0	0	0
											
	4094	1	1	1	1	1	1	1	1	1	1	1	0
	4095	1	1	1	1	1	1	1	1	1	1	1	1

1 LSB = VREF 电压 / 4096

- 差分模式的码定义

下面的表格展示了差分模式的有符号码和无符号码格式的 ADC 码。

表 22-7. ADC 差分模式数据格式定义

格式	ADC 输入 电平 (LSB)	ADC 有符号/无符号输出码											
		B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
2's 补码 (有符号)	-2048	1	0	0	0	0	0	0	0	0	0	0	0
	-2047	1	0	0	0	0	0	0	0	0	0	0	1
											
	-1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	1
											
	2046	0	1	1	1	1	1	1	1	1	1	1	0
	2047	0	1	1	1	1	1	1	1	1	1	1	1

1 LSB = VREF 电压 / 4096

B11 = 2's 补码模式的符号位

22.11.3. 分辨率和数据对齐

用户可设置 **ADCx_RES_SEL** 寄存器选择 ADC 输出码分辨率。此外，用户可设置 **ADCx_ALIGN_SEL** 寄存器选择 ADC 数据寄存器中左对齐或右对齐码格式。

下面的表格展示了数据寄存器 **ADCx_DATn** 的数据对齐格式。

表 22-8. ADC 数据对齐定义

对齐	ADCx_DATn (ADC 数据寄存器)															
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
12 位-右	Z	Z	Z	Z	S ADC 输出码[11:0]											
10 位-右	Z	Z	Z	Z	Z	Z	S ADC 输出码[9:0]									
8 位-右	Z	Z	Z	Z	Z	Z	Z	Z	S AD 输出码[7:0]							
12 位-左	S ADC 输出码[11:0]												x	x	x	x
10 位-左	S ADC 输出码[9:0]										x	x	x	x	x	x
8 位-左	S ADC 输出码[7:0]								x	x	x	x	x	x	x	x

ADC 输出码 : ADC 通过数字偏移和有符号/无符号调整器的输出码输出

x：未知

S：有符号（2's 补码）模式的符号位或无符号模式的 Msb

Z：有符号（2's 补码）模式的符号位或无符号模式的 0

22.11.4. ADC 数据寄存器

当 ADC 转换完成，ADC 输出码会被 ADC 输出控制块调整并输出转换结果到 **ADCx_DAT0** 寄存器中。

只读寄存器 **ADCx_DAT0_CH** 用于表明此时 ADC 数据输入通道标号；完成标志在 **ADCx_DAT0_CF** 寄存器位中；溢出标志在 **ADCx_DAT0_OVRF** 寄存器位中用于当前 ADC 数据。

此外，还有 3 个额外的电压窗口检测标志 **ADCx_DAT0_WDLF**, **ADCx_DAT0_WDIF** 和 **ADCx_DAT0_WDLF** 用于当前 ADC 数据。

下面的表格展示了 ADC 12 位分辨率输出的 ADC 码范围示例。

[注释]: ADC 有符号码不支持于 MG32F02A032。

表 22-9. ADC 数据码示例

对齐	ADCx_DATn (ADC 数据寄存器)																值
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	
12 位-右(无符号)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	2047
	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2048
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4095
12 位-右(有符号)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	2047
	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	-2048
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1
12 位-左(无符号)	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	0
	0	1	1	1	1	1	1	1	1	1	1	1	x	x	x	x	2047
	1	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	2048
	1	1	1	1	1	1	1	1	1	1	1	1	x	x	x	x	4095
12 位-左(有符号)	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	0
	0	1	1	1	1	1	1	1	1	1	1	1	x	x	x	x	2047
	1	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	-2048
	1	1	1	1	1	1	1	1	1	1	1	1	x	x	x	x	-1

ADC 输出码：ADC 通过数字偏移和有符号/无符号调整器的输出码输出

x：未知

22.11.5. 电压窗口检测

ADC 可通过阈值窗口比较输入电压。用户可在 **ADCx_WIND_EN** 寄存器中使能电压窗口检测功能，并通过 **ADCx_WIND_MDS** 寄存器选择“单独”或“全部”模式。当选择“单独”模式，在 **ADCx_CH_MUX** 寄存器中选择的通道数据会被电压窗口检测器检测；当选择“全部”模式，所有选择的通道数据会被电压窗口检测器检测。

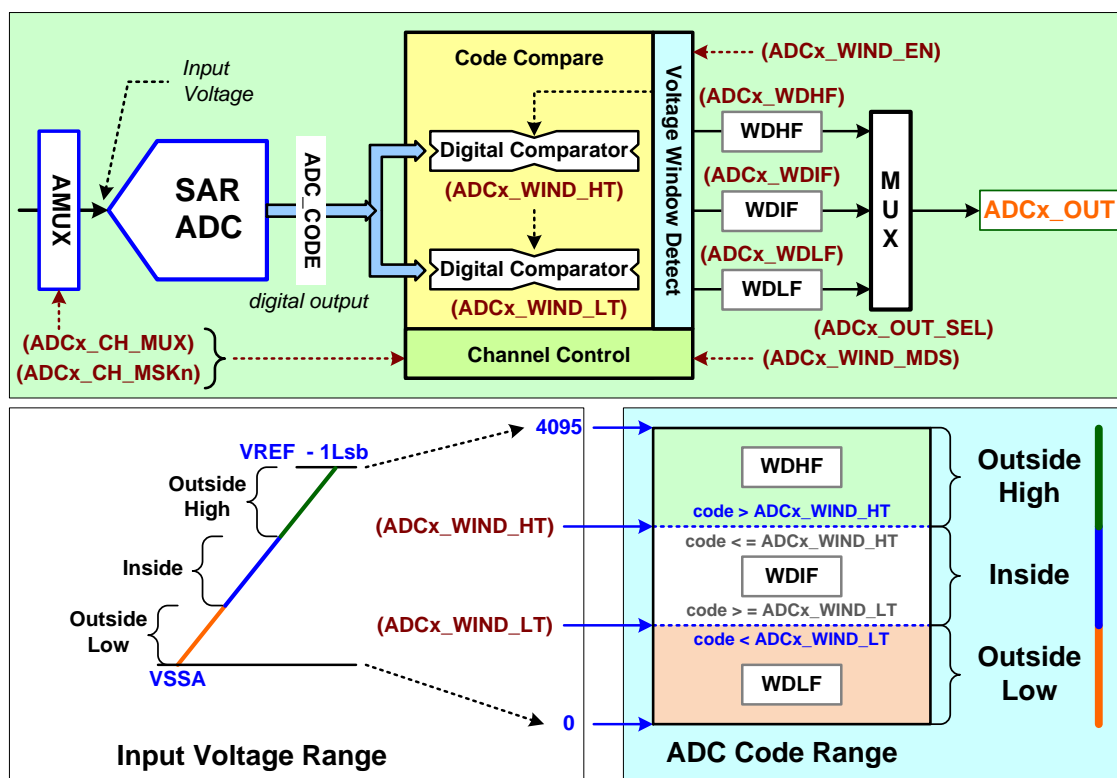
阈值窗口被 **ADCx_WIND_LT** 和 **ADCx_WIND_HT** 寄存器定义并分成 3 块区域。比较结果会被反映在状态标志 **WDLF**, **WDIF**, **WDHF** (**ADCx_WDLF**, **ADCx_WDIF**, **ADCx_WDHF**) 上，并指示外部低、内部、外部高区域。当被比较的 ADC 码大于等于 **ADCx_WIND_HT** 寄存器的阈值，**WDHF** 标志会被置起；当被比较的 ADC 码小于等于 **ADCx_WIND_LT** 寄存器的阈值，**WDLF** 标志会被置起；其他 ADC 码会导致置起 **WDIF** 标志。

它还可以像模拟看门狗，产生复位事件到复位源控制器。模拟看门狗的特性允许应用检测输入电压在用户定义的阈值以上或以下。参照“ADC 中断控制”以获取更多信息。

输出选择寄存器 **ADCx_OUT_SEL** 用于选择窗口检测状态（外部低、内部、外部高）或内部数据准备信号 (**ADCx_RDY**) 从 **ADCx_OUT** 端口输出。

下面的图表展示了 ADC 电压窗口检测块。

图 22-16. ADC 电压窗口检测



<Note> 1: x= Module index

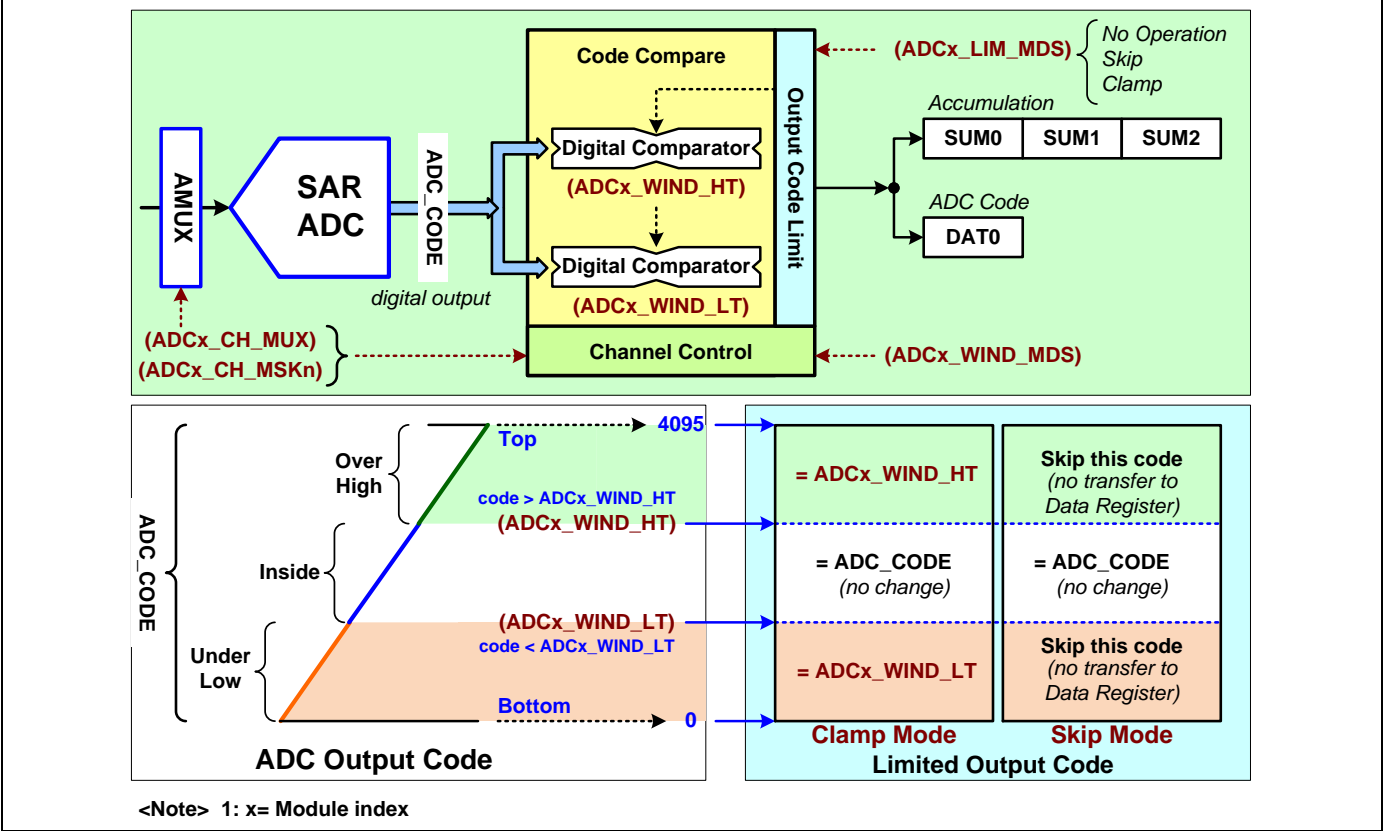
22.11.6. 输出码限制

ADC 输出码可被码限制区比较。用户可在 **ADCx_WIND_EN** 寄存器中使能电压窗口检测功能，并通过 **ADCx_WIND_MDS** 寄存器选择“单独”或“全部”模式。当选择“单独”模式，在 **ADCx_CH_MUX** 寄存器中选择的通道数据会被电压窗口检测器检测；当选择“全部”模式，所有可选的通道数据会被电压窗口检测器检测。

码限制区也被 **ADCx_WIND_LT** 和 **ADCx_WIND_HT** 寄存器定义。输出码限制功能可通过 **ADCx_LIM_MDS** 寄存器设置为(1)不操作 (2)跳过 (3)钳制。

下面的图表展示了 ADC 输出码限制块。

图 22-17. ADC 输出码限制



22.12. ADC 数据和

22.12.1. ADC 数据累加

ADC 内建硬件累加器用于 ADC 输出码。累加器用于累加可设置数据数量的 ADC 数据并记录和到总和寄存器中。用户可在 **ADCx_SUM_NUM** 寄存器中设置累加的 ADC 数据数量。ADC 支持 3 个数据和寄存器 **ADCx_SUMn** ($n = \{0, 1, 2\}$)。用户可从这些寄存器获取累加和。

用户可通过 **ADCx_SUM_MDS** 寄存器选择“单独”或“全部”累加模式。

当选择 ADC 单次转换模式的“单独”模式时，**ADCx_SUM0_MUX** 选择的通道数据会被累加入 **ADCx_SUM0**；当选择“全部”模式，所有的可选择通道数据会一个接一个的累加入 **ADCx_SUM0** 中；当选择 ADC 通道扫描转换模式的“单独”模式时，**ADCx_SUM1_MUX/ADCx_SUM2_MUX** 选择的通道数据会被分别累加入 **ADCx_SUM1/ADCx_SUM2**。这些寄存器可被用户预设初始值，并累加 ADC 转换数据和。

下面的表格展示了 ADC 数据和以及转换模式控制。

表 22-10. ADC 数据和以及转换模式控制

转换模式		分离和的功能	所有数据和的功能
CONV_MDS	TRG_CONT	SUM_MDS = Single	SUM_MDS = All
单次	禁用	只有在每次 ADC 转换完成后，才会将 ADC ADCx_SUM0_MUX 选择通道的 ADC 输出数据累积到 ADCx_SUM0_DAT 中。此外，每次 ADC 转换完成后，数据和计数器都会增加 1，直到计数器值等于 ADCx_SUM_NUM 。	只有在每次 ADC 转换完成后，才会将 ADC 所有的可选择通道的 ADC 输出数据一个接一个累积到 ADCx_SUM0_DAT 中。此外，每次 ADC 转换完成后，数据和计数器会增加 1，直到计数器值等于 ADCx_SUM_NUM 。
单次	使能		
扫描	禁用	只有在每次 ADC 转换完成后，才会将 ADC 分离的 ADCx_SUMn_MUX 选择通道的 ADC 输出数据累积到 ADCx_SUM0, ADCx_SUM1, ADCx_SUM2 中。此外，每次 ADC 转换完成后，数据和计数器会增加 1，直到计数器值等于 ADCx_SUM_NUM 。	只有在每次 ADC 转换完成后，才会将 ADC 所有的可选择通道的 ADC 输出数据一个接一个累积到 ADCx_SUM0_DAT 中。此外，每次通道扫描完成后，数据和计数器会增加 1，直到计数器值等于 ADCx_SUM_NUM 。
扫描	使能		
循环	X	只有在每次 ADC 转换完成后，才会将 ADC 分离的 ADCx_SUMn_MUX 选择通道的 ADC 输出数据累积到 ADCx_SUM0, ADCx_SUM1, ADCx_SUM2 中。此外，每次 ADC 通道扫描完成后，数据和计数器会增加 1，直到计数器值等于 ADCx_SUM_NUM 。	

ADCx_SUMn_DAT : ADC 输出码的数据和 ($x = \text{ADC 模块标号}, n = \text{数据和寄存器标号}$)

ADCx_SUMn_MUX : 选择的 ADC 通道复用器用于数据和 **ADCx_SUMn_DAT**

ADCx_SUM_NUM : **ADCx_SUMn_DAT** 总数据数量

下面的表格展示了 ADC 数据和标志对比转换模式。

表 22-11. ADC 数据和标志对比转换模式

模式	算数和的完成标志	算数和过载的标志	算数和溢出的标志
CONV_MDS	SUMCF / SUMn_CF	SUMOVRF / SUMn_OVRF	SUMOF / SUMn_OF
单次	若相关通道-n 已完成累加，置起 ADCx_SUMCF 和 ADCx_SUMn_CF	若相关通道-n 累加过载，置起 ADCx_SUMOVRF 和 ADCx_SUMn_OVRF	若相关通道-n 累加溢出，置起 ADCx_SUMOF 和 ADCx_SUMn_OF
扫描	若相关通道已完成累加，置起 ADCx_SUMCF 和所有的 ADCx_SUMn_CF	若相关通道累加过载，置起 ADCx_SUMOVRF 和所有的 ADCx_SUMn_OVRF	若相关通道累加溢出，置起 ADCx_SUMOF 和所有的 ADCx_SUMn_OF
循环			

22.12.2. ADC 数据和寄存器

累加器累加 ADC 数据并将累加和存入 **ADCx_SUMn** 寄存器中。(n = {0, 1, 2})

对于每个数据和寄存器，在 **ADCx_SUMn_CF** 寄存器位中有 1 个完成标志；在 **ADCx_SUMn_OVRF** 寄存器位中有 1 个过载标志。此外，有 2 个累加上溢标志和下溢标志在 **ADCx_SUMn_OF** 和 **ADCx_SUMn_UF** 寄存器位中。

下面的表格展示了 ADC 单端模式的无符号格式数据和的值范围，以及 ADC 差分模式的有符号格式数据和的值范围。

[注释]: ADC 有符号码不支持于 MG32F02A032。

表 22-12. ADC 数据和值范围

格式	ADCx_SUMn (ADC 累加和寄存器)																值
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	
无符号	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	32767
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32768
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	65535
有符号	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	32767
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-32768
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1

无符号：用于单端模式；有符号：用于差分模式

22.12.3. ADC 数据和过载

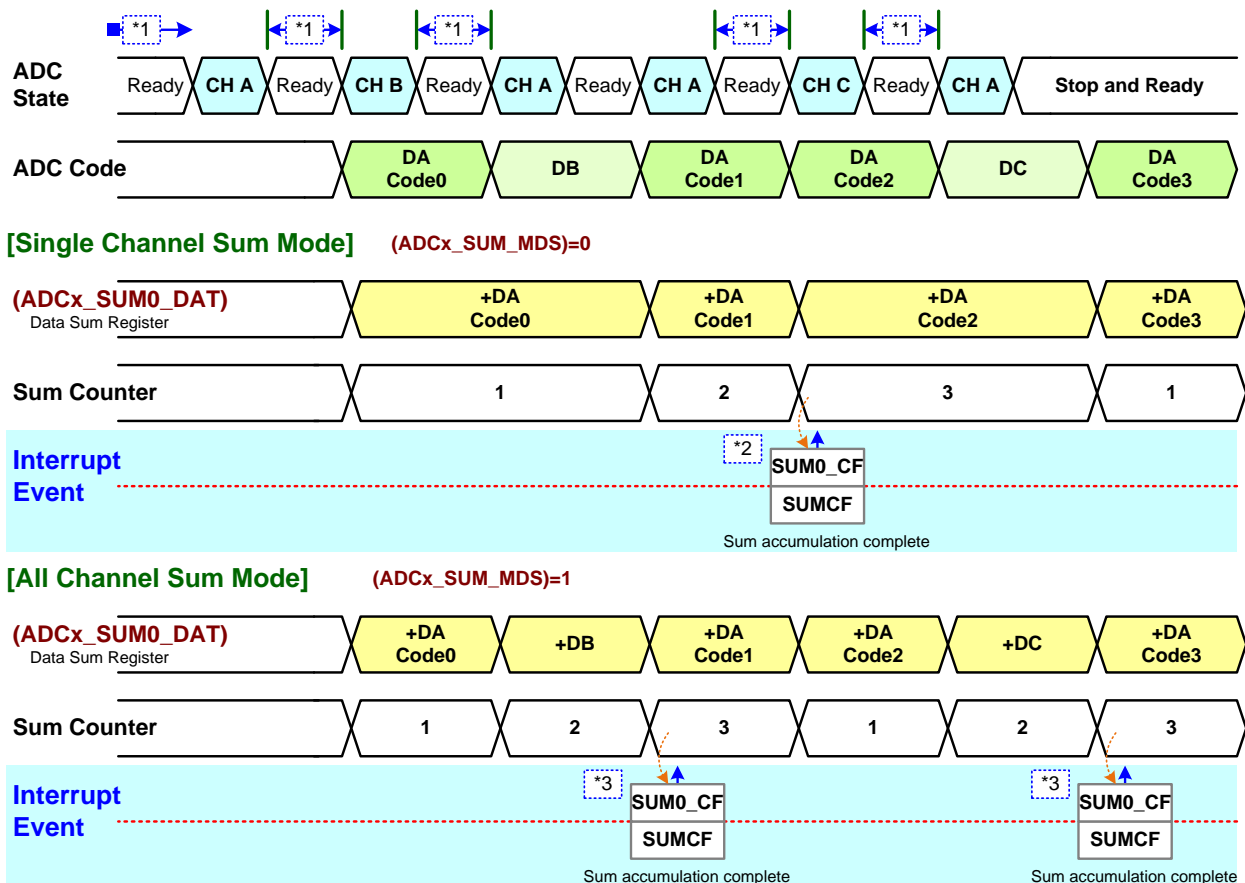
当 ADC 累加和完成且前一个累加和数据还未被软件移出，会导致 ADC 数据和寄存器过载，用户可在发生过载前通过设置 **ADCx_SOVR_MDS** 寄存器设置覆盖旧累加数据还是保留旧累加数据。

22.12.4. ADC 数据和转换时序

- ADC 数据和模式-单次或间断通道扫描

下面的图表展示了 ADC 数据和模式的“单次”或“间断通道扫描”的控制时序。

图 22-18. ADC 数据和模式-单次或间断通道扫描



<Note-1> Software set ADC channel by setting ($ADCx_CH_MUX$) .

<Note-2> The condition is ($ADCx_SUM_NUM$)=3 and ($ADCx_SUM0_MUX$)=A for example.

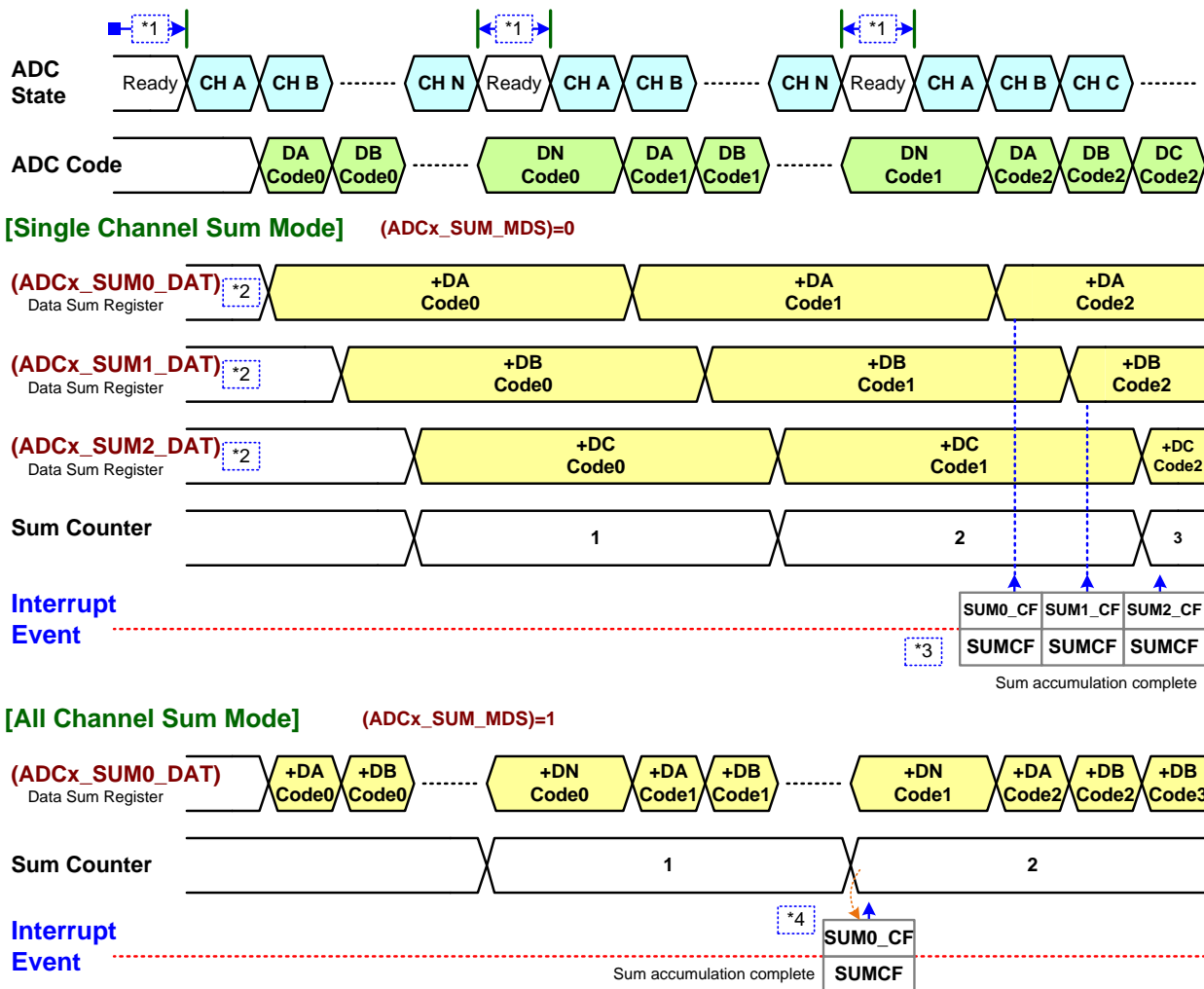
<Note-3> The condition is ($ADCx_SUM_NUM$)=3 and ($ADCx_SUM0_MUX$)=do not care for example.

<Note-4> {A,B,C...} indicate the enabled channels those are following CH(channel) and D(Data) .

- ADC 数据和模式-通道/循环扫描

下面的图表展示了 ADC 数据和模式的“通道扫描”或“循环扫描”的控制时序。

图 22-19. ADC 数据和模式-通道/循环扫描



<Note-1> Software set ADC channels by setting (ADCx_CH_MSKn) before channel scan ADC conversion .

<Note-2> (ADCx_SUM0_MUX)=A , (ADCx_SUM1_MUX)=B , (ADCx_SUM2_MUX)=C for example.

<Note-3> The condition is (ADCx_SUM_NUM)=3 for example.

<Note-4> The condition is (ADCx_SUM_NUM)=2 and (ADCx_SUM0_MUX)=do not care for example.

<Note-5> {A,B,C...,N} indicate the enabled channels those are following CH(channel) and D(Data) .

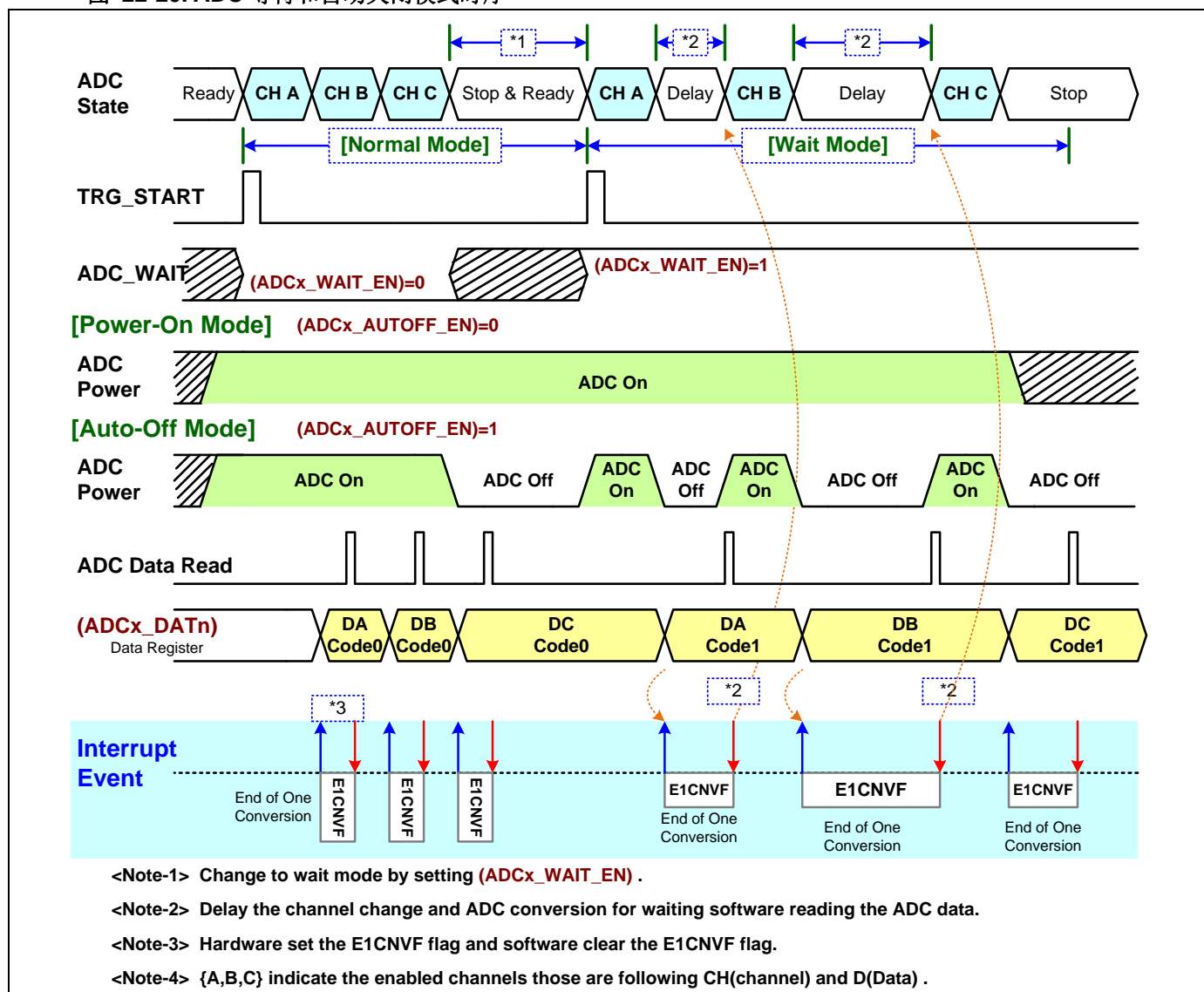
22.13. ADC 等待和自动关闭

ADC 通过设置 **ADCx_WAIT_EN** 寄存器支持和使能等待模式功能。通常，该功能可避免 ADC 在低频率的 ADC 采样时钟下过载。

此外 ADC 通过设置 **ADCx_AUTOFF_EN** 寄存器支持和使能自动关闭模式功能，该功能可在除正在进行转换外的时候强制 ADC 进入关闭模式。ADC 会在检测到下一次转换触发启动时自动上电并恢复对输入信号的转换。该模式提供了 ADC 转换应用中，长时间闲置周期里节省芯片耗电的功能。

[注释]: **ADCx_AUTOFF_EN** 不支持于 MG32F02A032。

图 22-20. ADC 等待和自动关闭模式时序



22.14. ADC DMA 操作

22.14.1. DMA 模块设置

当芯片支持 DMA（直接内存访问）控制器时，用户可以在 DMA 数据传输前，在 DMA 模块中设置关于来源/目的设备、通道请求仲裁等 DMA 设置。DMA 来源和目的可以是内存或外设。

参照 DMA 章以获取更多关于 DMA 模组设置的细节。

22.14.2. ADC DMA 控制

DMA 设置完成后，用户需设置 ADC 模块的 DMA 使能位 **ADCx_DMA_EN**。

最后，相关的通道请求起始位 **DMA_CHn_REQ** 是必须的，用于设置 DMA 传输启动(n = DMA 通道标号)。然

后传输源和目的设备会置起请求信号到 DMA 控制器，DMA 控制器便会置起接收信号到请求源/目的设备。此时，该数据传输连接是用于 DMA 传输的。

- **ADC 输入到 DMA**

ADCx_DMA_EN 寄存器位用于使能来自 ADC 转换的输入通过 DMA 发送到 DMA 目的地。

DMA 发送周期中，数据就绪标志标志 **ADCx_ESMPF**, **ADCx_E1CNVF** 和 **ADCx_ESCNVF** 是被硬件屏蔽的。

ADC DMA 传输数据包大小可通过 **ADCx_DAT0** 寄存器设置为 16 位或 32 位。当选择 16 位时，芯片会用 DMA 传输 **ADCx_DAT0** 的位[15:0]，当选择 32 位，芯片则会传输 **ADCx_DAT0** 的全部 32 位。

[注释]: 在 MG32F02A132/072 中，**ADCx_DMA_DSIZE** 不被支持且恒为 32 位数据包大小。

- **ADC 采样时钟限制**

当 ADC DMA 功能被使能，ADC 采样时钟频率必须小于等于 1/4 的 AHB 时钟频率用于正常的 DMA 传输。

22.14.3. ADC 的 DMA 中断标志控制

DMA 工作周期中，模块的中断标志会控制和执行 3 种类型，如下表格。

其中一方在 DMA 工作过程中被屏蔽(ESMPF, E1CNVF 和 ESCNVF 标志)，另一方会在标志(OVRF, WDLF, WDIF 和 WDHF 标志)被置起后禁用 DMA 功能。此时，硬件会清除 **ADCx_DMA_EN** 位。其他操作与 DMA 操作中不执行的操作相同。

用户可通过 **ADCx_DMA_MDS** 寄存器设置当 DMA 工作时 E1CNVF 标志的置起模式。当选择“禁用”时，E1CNVF 会在 ADC 转换结束时被屏蔽；当选择“保持”时，E1CNVF 会在 ADC 转换结束时被置起。该功能只在单次或单扫描 ADC 转换模式中被支持。

[注释]: **ADCx_DMA_MDS** 不支持于 MG32F02A132/072。

表 22-13. ADC DMA 功能中断标志控制 – MG32F02A132/072

行为	DMA 操作中屏蔽标志	标志置起后 DMA 被禁用(*1)	一般控制
外设	(数据溢出标志)	(错误/检测标志)	(其他标志)
ADCx	ADC0_ESMPF ADC0_E1CNVF ADC0_ESCNVF(*2)	ADC0_OVRF ADC0_WDLF ADC0_WDIF ADC0_WDHF	ADC0_SUMOF ADC0_SUMCF ADC0_SUMOVRF

注释-1：当标志被置起，若相关中断使能位没被使能，它不会强行禁用外设 DMA。

注释-2：该标志会在 DMA TX 完成后被置起。

表 22-14. ADC DMA 功能中断标志控制 – MG32F02A032

行为	DMA 操作中屏蔽标志	标志置起后 DMA 被禁用(*1)	一般控制
外设	(数据溢出标志)	(错误/检测标志)	(其他标志)
ADCx	ADC0_ESMPF ADC0_E1CNVF(*3) ADC0_ESCNVF(*2)	ADC0_OVRF ADC0_WDLF ADC0_WDIF ADC0_WDHF	ADC0_SUMOF ADC0_SUMCF ADC0_SUMOVRF

注释-1：当标志被置起，若相关中断使能位没被使能，它不会强行禁用外设 DMA。

注释-2：该标志会在 DMA TX 完成后被置起。

当被禁用时，标志会在 DMA 工作过程中被屏蔽，当设置为“保持”，E1CNVF 不会在 DMA

注释-3：工作过程中被屏蔽。

22.15. ADC 应用电路

ADC 参考电压源可来源于

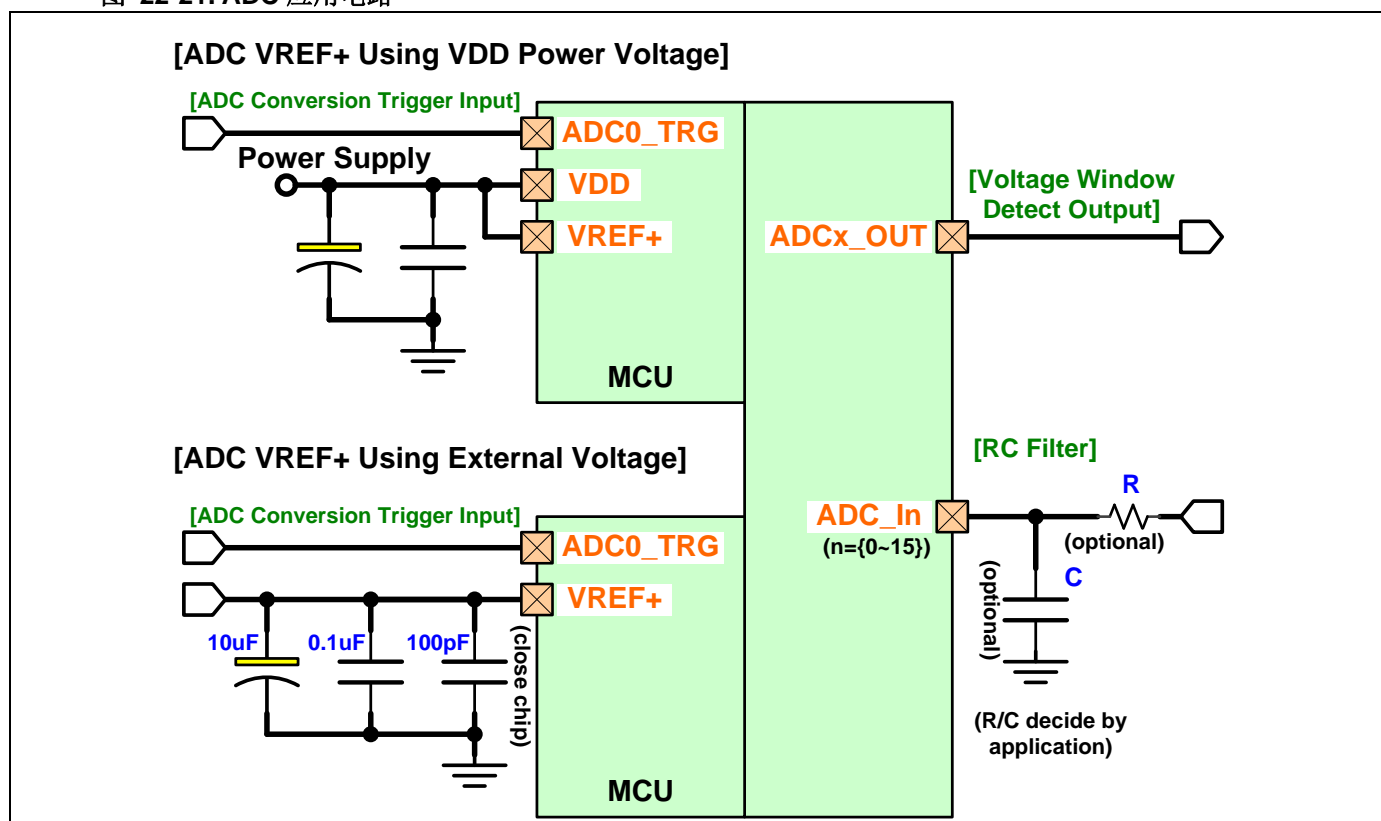
- (1) 通过把+VREF 引脚直接连到 VDD 引脚上使用 VDD 供电
- (2) 外部静态参考电压源

当使用 VDD 电源作为 ADC 的参考电压时，它必须将+VREF 引脚连接到电源电容器滤波后的连接点。当使用外部参考电压源作为 ADC 参考电压时，它必须添加一些去耦和旁路电容器，如下图所示。

[注释]: 在 MG32F02A132/072 中，当在+VREF 引脚使用外部参考电压作为 ADC 参考电压时，该电压必须 $\geq 3.0V$ 。

可选择的 [ADCx_TRG] 引脚用于输入触发 ADC 输入转换信号和可选择的 [ADCx_OUT] 引脚用于输出内部 ADC 窗口检测状态。

图 22-21. ADC 应用电路



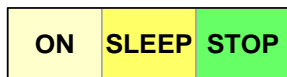
● 模拟输入滤波器

采用模拟输入滤波器作为抗混叠滤波器，消除了 ADC 采样过程中的混叠效应。当输入信号中存在不需要的信号或噪声时，会发生混叠，这些信号或噪声的频率高于 ADC 采样频率的一半（尼奎斯特频率）。这些不需要的信号会混入原始转换信号。这些噪音很难使用后端数字滤波器滤除。因此强烈建议，为了最好的 ADC 转换表现，用户需加外部抗混叠滤波器去除不需要的信号。

一般采用一阶电阻电容低通滤波器来滤除高频噪声，减少混叠效应。低通滤波器的频带宽度可以设计为 ADC 采样频率的一半或更小。

23. CMP (模拟比较器)

23.1. 简介



The module can be running in all power operation modes.

该芯片内嵌了 1 个含有 4 个有灵活的输入多路复用器的通用模拟比较器、2 个 R-阶梯内部参考电压和为每个模拟比较器独立配备的数字同步滤波器的 CMP 模块。这些模拟比较器可被配置为 4 个独立比较器或 1 个组合窗口比较器。这个模块提供了比较器输出结果状态位和上升沿和下降沿改变时的中断标志。此外，输出结果可以被输出到外部引脚或内部其他模块作为触发事件。

注释：标志(n=模拟比较器宏标号) 会被用于该章的寄存器、信号和引脚/端口描述中。

23.2. 特性

- 可设置 64 阶梯阈值的内部参考电压
- 为所有的+/-输入路径选择提供灵活的 6 个通道输入
- 为了最佳电流消耗提供可编程响应时间
- 使用 2 个比较器组合成窗口比较器
- 可选择的比较输出极性
- 支持从 SLEEP 和 STOP 唤醒
- 比较输出到 I/O、中断或作为内部模块触发事件
 - 定时器内部触发、捕获事件或中止事件
- 支持模拟看门狗作为复位源

❖ MG32F02A132/072

- 提供 4 个快速轨对轨比较器
- 为所有的比较器提供总共 10 个外部通道输入

❖ MG32F02A032

- 提供 2 个快速轨对轨比较器
- 为所有的比较器提供总共 6 个外部通道输入

23.3. 配置

23.3.1. 芯片配置

下面的表格展示了芯片 CMP 模拟比较器配置。

表 23-1. CMP 配置

芯片	CMP 模块模拟比较器			
	模拟 COMP0	模拟 COMP1	模拟 COMP2	模拟 COMP3
MG32F02A132	V	V	V	V
MG32F02A072	V	V	V	V
MG32F02A032	V	V		

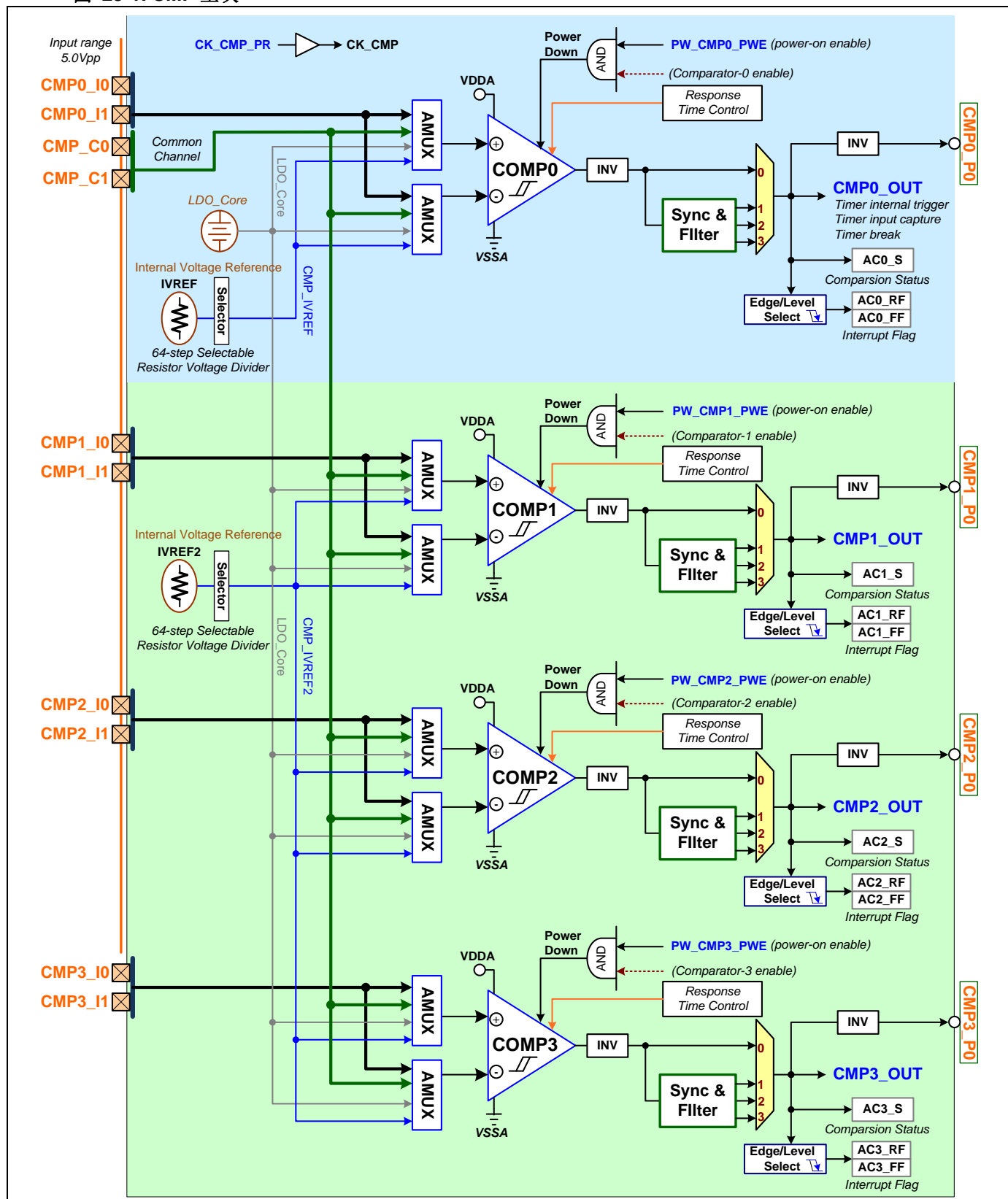
注释 V：包含

23.4. 控制块

CMP 模块包含最多 4 个相同设计的通用模拟比较器 CMP0~3 和 2 个 R-阶梯内部参考电压 $IVREF/IVREF2$ 。每一个都配有独立的输入多路复用器、数字同步滤波器和数字输出电路。 $IVREF$ 只用于 CMP0 而 $IVREF2$ 则是 CMP1~3 共享。

下面的图表展示了模拟比较器控制块。

图 23-1. CMP 主块



23.5. IO 线

23.5.1. IO 信号

- **CMP_Cn**
模拟比较器的通用通道-n 的模拟输入信号，可用作所有模拟比较器的正极或负极输入。
- **CMPn_I[0..1]**
模拟比较器 **COMPn** 的模拟输入信号用于正负极输入。
- **CMPn_P0**
模拟比较器 **COMPn** 的比较数据结果输出。

23.5.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。使用的模拟输入 **CMP_Cn** 和 **CMPn_I[0..1]** 引脚必须设置 IO 模式为 AIO 模式。参照用户手册 GPIO 章中“[IO 模式](#)”节以获取更多关于 IO 模式设置的信息。

每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“[功能复用选择](#)”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“[引脚功能复用表](#)”以获取更多信息。

23.6. 电源和时钟

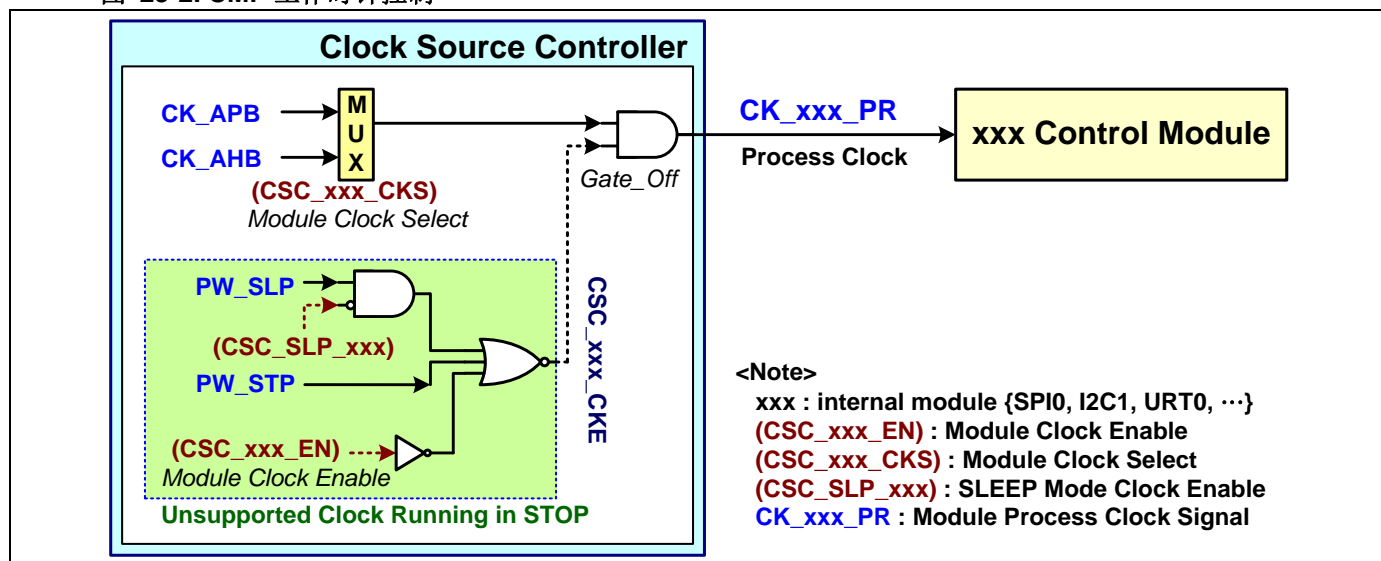
23.6.1. CMP 电源控制

CMP 模拟宏的工作电源 VDDA 来源于 IO 电源 **VDD** 引脚。4 个模拟比较器有自己独立的启动使能位 **CMP_ACn_EN**。每个 CMP 模拟比较器只有在 **CMP_ACn_EN** 位被设置为逻辑 1 时使能。每个模拟比较器都可以为了省电而独立的禁用和关闭。用户可通过 **PW_SLP_CMPn** 或 **PW_STP_CMPn** 寄存器在芯片进入 **SLEEP** 或 **STOP** 模式之前独立设置每个模拟比较器是否继续工作。参照系统电源章以获取更多信息。(n={0,1,2,3})

23.6.2. CMP 时钟控制

该模块工作时钟 **CK_CMP_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 CSC（时钟源控制器）模块。该时钟可通过 **CSC_CMP_EN** 寄存器使能并通过 **CSC_CMP_CKS** 寄存器选择时钟源来自 APB 或 AHB。用户可以在芯片进入 **SLEEP** 模式之前通过设置 **CSC_SLP_CMP** 寄存器规划在 **SLEEP** 模式下是否让 CMP 时钟继续运行。参照系统时钟章以获取更多信息。

图 23-2. CMP 工作时钟控制



23.7. 中断和事件

23.7.1. CMP 中断控制

模拟比较器控制模块中有 3 种信号 **INT_CMP**, **RST_CMPn**, **WUP_CMPn**。

- 中断事件

INT_CMP 发送到外部中断控制器（EXIC）作为中断事件。(n={0,1,2,3})

中断标识是用于中断服务程序（ISR）流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **CMP_IEA** 用于使能和禁用该模块的所有中断源。

- 复位事件

RST_CMPn 发送到复位源控制器（RST）作为热复位或冷复位事件(n={0,1,2,3})。这些复位事件可通过 RST 寄存器设置使能复位芯片。

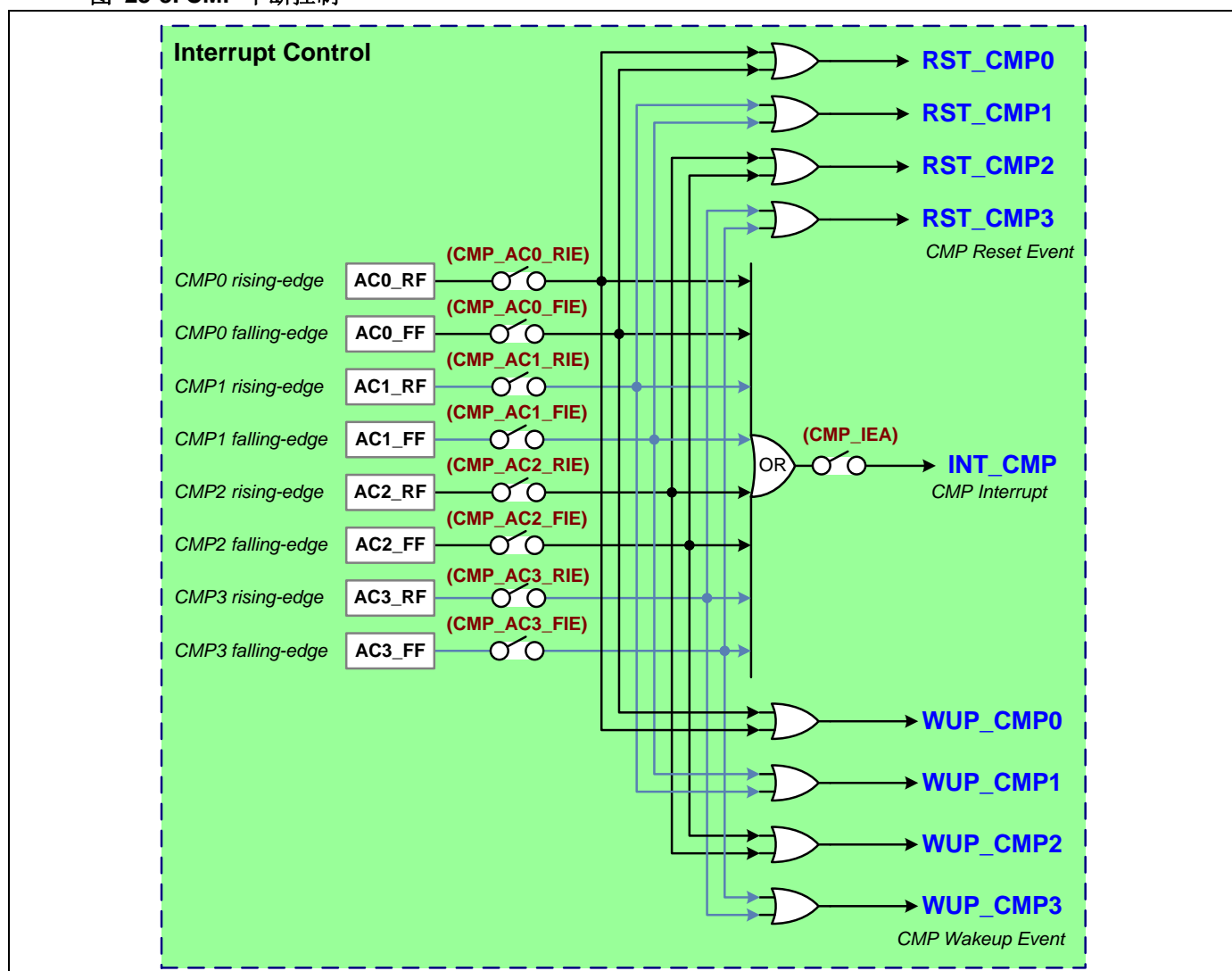
参照系统复位章描述以获取更多关于复位事件和控制的信息。

- 唤醒事件

WUP_CMPn 信号发送到电源控制器(PW)做系统唤醒事件(n={0,1,2,3})。这些唤醒事件可通过设置 PW 寄存器在 SLEEP 或 STOP 模式下唤醒芯片。电源控制器管理这些信号并发送唤醒信号到 EXIC 和 NVIC 做芯片唤醒控制。

参照系统电源和中断章描述以获取更多关于唤醒事件和控制的信息。

图 23-3. CMP 中断控制



23.7.2. CMP 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

- ACn_RF

模拟比较器 **CMPn** 上升沿中断标志是(**CMP_ACn_RF**)，相关的中断使能寄存器位是 **CMP_ACn_RIE**。

- **ACn_FF**

模拟比较器 **CMPn** 下降沿中断标志是(**CMP_ACn_FF**)，相关的中断使能寄存器位是 **CMP_ACn_FIE**。

23.8. CMP 模拟比较器

23.8.1. 输入通道

- **模拟输入多路复用**

模拟多路复用器(AMUX)选择到每个模拟比较器的 **CMPn_I0,CMPn_I1** 输入，和到模拟比较器 **CMP0/1/2/3** 的每个 **CMP_C0,CMP_C1** 输入。它允许任何到 **CMP0/1/2/3** 的输入引脚在正极输入和负极输入之间进行比较。AMUX 可通过 **CMP_ACn_PMUX, CMP_ACn_NMUX** 寄存器进行设置。(n={0,1,2,3})

- **用于比较器功能的 I/O 引脚**

用于比较器的模拟输入引脚也有它的 I/O 端口数字输入和输出功能。为了提供适当的模拟性能，被使用的引脚需要禁用数字输出，将端口引脚置仅输入模式即可。此外，当模拟信号已作用于模拟输入引脚且数字输入引脚不需要被使用时，软件可以通过 **PX_IOMn** (X={A,B},n={0~15})寄存器将相应的引脚设置成 AIO 模式来降低数字输入缓冲区的功耗。

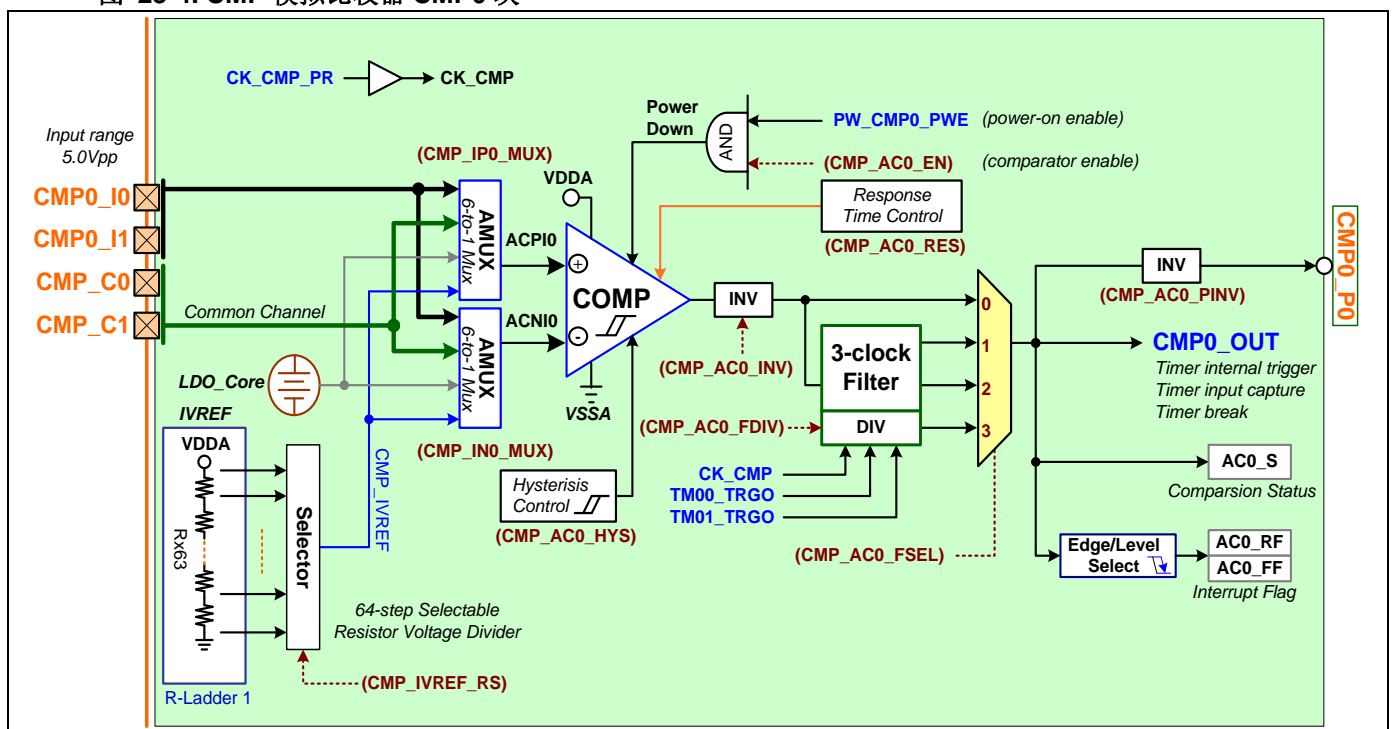
23.8.2. 内部电压参考

模拟比较器内置 2 个 64 阶梯 R-梯形内部参考电压 - **IVREF** 和 **IVREF2**。他们可以作为其中 1 个模拟比较器输入，并与其他外部源的输入进行比较。参照“CMP 内部电压参考”节以获取更多信息。

23.8.3. 模拟比较器-0

内建内部电压参考 - **IVREF**可作为比较器输入，但也只能用于 **CMP0**。下面的图表展示了模拟比较器-0 控制块。

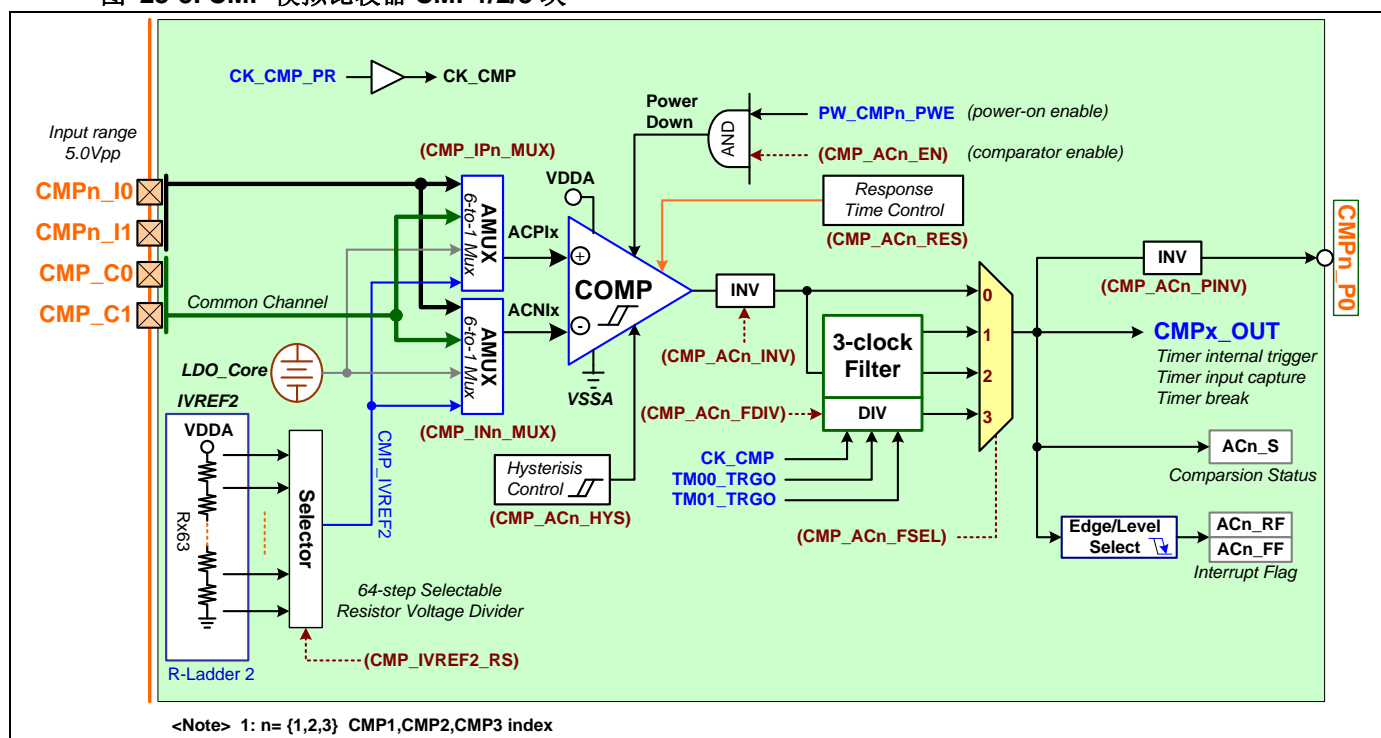
图 23-4. CMP 模拟比较器 CMP0 块



23.8.4. 模拟比较器-1/2/3

模拟比较器 **CMP1/2/3** 结构与 **CMP0** 相同。内建电压参考- **IVREF2** 被 **CMP1/2/3** 共用。下面的图表展示了模拟比较器-1/2/3 控制块。

图 23-5. CMP 模拟比较器 CMP1/2/3 块



23.8.5. 响应时间选择

模拟比较器根据不同应用可通过 **CMP_ACn_RES** 寄存器选择不同的响应时间。它还可以控制比较器的工作电流，以获得最佳的电流消耗。(n={0,1,2,3})

23.8.6. 比较器输出

● 比较器输出结果

模拟比较器也可以通过 **CMP_ACn_INV** 寄存器设置调整输出极性。输出结果会被存在 **CMP_ACn_S** 中。每个比较器的 **CMPn_OUT** 信号可输出到其他内部模块中。比如它可发送到内部定时器模块做定时器触发输入、定时器捕获输入或定时器中止输入。

当输出改变，相关标志 **CMP_ACn_RF** 或 **CMP_ACn_FF** 会在 0-1 上升改变或 1-0 下降改变时置起。(n={0,1,2,3})

● 比较器输出滤波器

模拟比较器输出可通过 **CMP_ACn_FSEL** 寄存器独立选择使用数字同步滤波器或旁路滤波器。同时该寄存器可使能 3-时钟同步滤波器并选择滤波器时钟源来自于模块内部时钟 **CK_CMP**、定时器触发输出信号 **TM00_TRGO** 和 **TM01_TRGO**。当使能了同步滤波器，用户可在 **CMP_ACn_FDIV** 寄存器中选择同步滤波器的时钟分频器。(n={0,1,2,3})

● 输出引脚控制

输出引脚 **CMPn_P0** 用于输出各个模拟比较器的结果到外部引脚。额外的反相控制位可在输出引脚 **CMPn_P0** 的 **CMP_ACn_PINV** 上进行独立的选择。(n={0,1,2,3})

23.9. CMP 内部电压参考

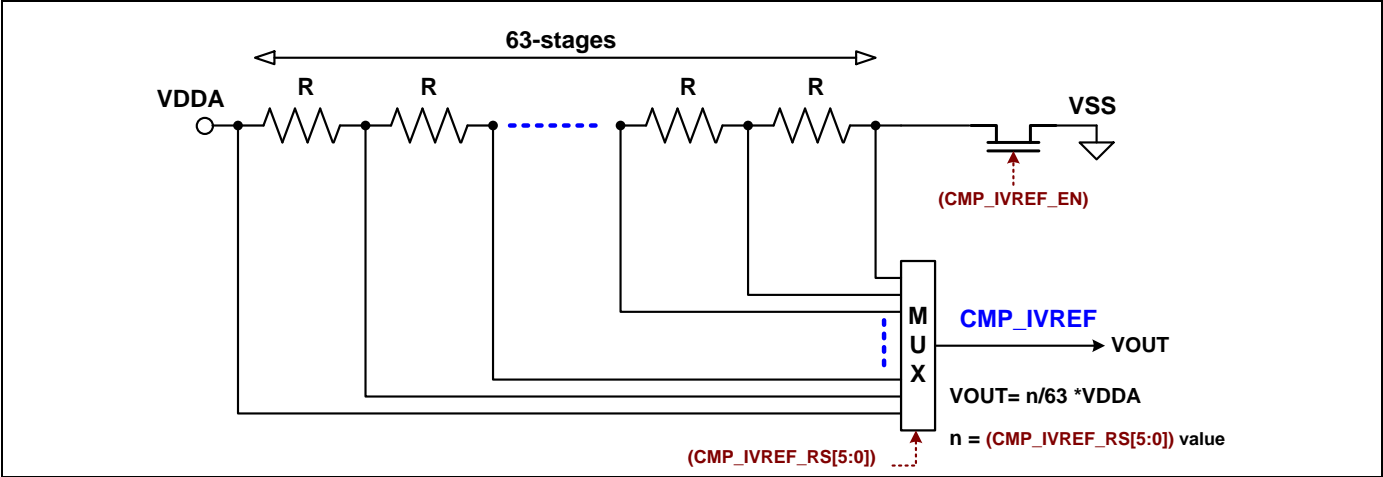
23.9.1. IVREF 和 IVREF2

模拟比较器内置 2 个相同结构的 64 阶梯 R-阶梯线性内部参考电压 - *IVREF* 和 *IVREF2*。*IVREF* 只用于 CMP0 而 *IVREF2* 与 CMP1~3 共享。他们可以作为其中 1 个模拟比较器输入，并与其他外部源的输入进行比较。

寄存器位 **CMP_IVREF_EN** 用于使能 *IVREF*，此外寄存器位 **CMP_IVREF2_EN** 用于使能 *IVREF2*。用户可禁用不使用的内部电压参考以省电。

下面的图表展示了内部电压参考 R-阶梯块。

图 23-6. CMP 比较器 IVREF 阶梯块



23.9.2. IVREF 输出电压

用户可通过下面的公式计算输出电压 *IVREF* 或 *IVREF2*:

$$\text{IVREF_OUT (volt)} = \frac{\text{VDDA} * \text{CMP_IVREF_RS}}{63}$$

下面的表格展示了通过 **CMP_IVREF_RS** 或 **CMP_IVREF2_RS** 寄存器设置的 R-阶梯 *IVREF* 或 *IVREF2* 输出。

表 23-2. CMP 比较器 IVREF R-阶梯输出

CMP 寄存器				IVREF 输出			
IVREFx_RS		VDDA		IVREFx_RS		VDDA	
		VDDA=+5V	VDDA=+3.3V			VDDA=+5V	VDDA=+3.3V
0	0/63 VDDA	0.000 V	0.000 V	32	32/63 VDDA	2.540 V	1.676 V
1	1/63 VDDA	0.079 V	0.052 V	33	33/63 VDDA	2.619 V	1.729 V
2	2/63 VDDA	0.159 V	0.105 V	34	34/63 VDDA	2.698 V	1.781 V
3	3/63 VDDA	0.238 V	0.157 V	35	35/63 VDDA	2.778 V	1.833 V
4	4/63 VDDA	0.317 V	0.210 V	36	36/63 VDDA	2.857 V	1.886 V
5	5/63 VDDA	0.397 V	0.262 V	37	37/63 VDDA	2.937 V	1.938 V
6	6/63 VDDA	0.476 V	0.314 V	38	38/63 VDDA	3.016 V	1.990 V
7	7/63 VDDA	0.556 V	0.367 V	39	39/63 VDDA	3.095 V	2.043 V
8	8/63 VDDA	0.635 V	0.419 V	40	40/63 VDDA	3.175 V	2.095 V
9	9/63 VDDA	0.714 V	0.471 V	41	41/63 VDDA	3.254 V	2.148 V
10	10/63 VDDA	0.794 V	0.524 V	42	42/63 VDDA	3.333 V	2.200 V
11	11/63 VDDA	0.873 V	0.576 V	43	43/63 VDDA	3.413 V	2.252 V
12	12/63 VDDA	0.952 V	0.629 V	44	44/63 VDDA	3.492 V	2.305 V
13	13/63 VDDA	1.032 V	0.681 V	45	45/63 VDDA	3.571 V	2.357 V
14	14/63 VDDA	1.111 V	0.733 V	46	46/63 VDDA	3.651 V	2.410 V
15	15/63 VDDA	1.190 V	0.786 V	47	47/63 VDDA	3.730 V	2.462 V

16	16/63 VDDA	1.270 V	0.838 V	48	48/63 VDDA	3.810 V	2.514 V
17	17/63 VDDA	1.349 V	0.890 V	49	49/63 VDDA	3.889 V	2.567 V
18	18/63 VDDA	1.429 V	0.943 V	50	50/63 VDDA	3.968 V	2.619 V
19	19/63 VDDA	1.508 V	0.995 V	51	51/63 VDDA	4.048 V	2.671 V
20	20/63 VDDA	1.587 V	1.048 V	52	52/63 VDDA	4.127 V	2.724 V
21	21/63 VDDA	1.667 V	1.100 V	53	53/63 VDDA	4.206 V	2.776 V
22	22/63 VDDA	1.746 V	1.152 V	54	54/63 VDDA	4.286 V	2.829 V
23	23/63 VDDA	1.825 V	1.205 V	55	55/63 VDDA	4.365 V	2.881 V
24	24/63 VDDA	1.905 V	1.257 V	56	56/63 VDDA	4.444 V	2.933 V
25	25/63 VDDA	1.984 V	1.310 V	57	57/63 VDDA	4.524 V	2.986 V
26	26/63 VDDA	2.063 V	1.362 V	58	58/63 VDDA	4.603 V	3.038 V
27	27/63 VDDA	2.143 V	1.414 V	59	59/63 VDDA	4.683 V	3.090 V
28	28/63 VDDA	2.222 V	1.467 V	60	60/63 VDDA	4.762 V	3.143 V
29	29/63 VDDA	2.302 V	1.519 V	61	61/63 VDDA	4.841 V	3.195 V
30	30/63 VDDA	2.381 V	1.571 V	62	62/63 VDDA	4.921 V	3.248 V
31	31/63 VDDA	2.460 V	1.624 V	63	63/63 VDDA	5.000 V	3.300 V

IVREFx_RS = CMP_IVREF_RS 或 CMP_IVREF2_RS 寄存器位

23.10. CMP 输入迟滞电压

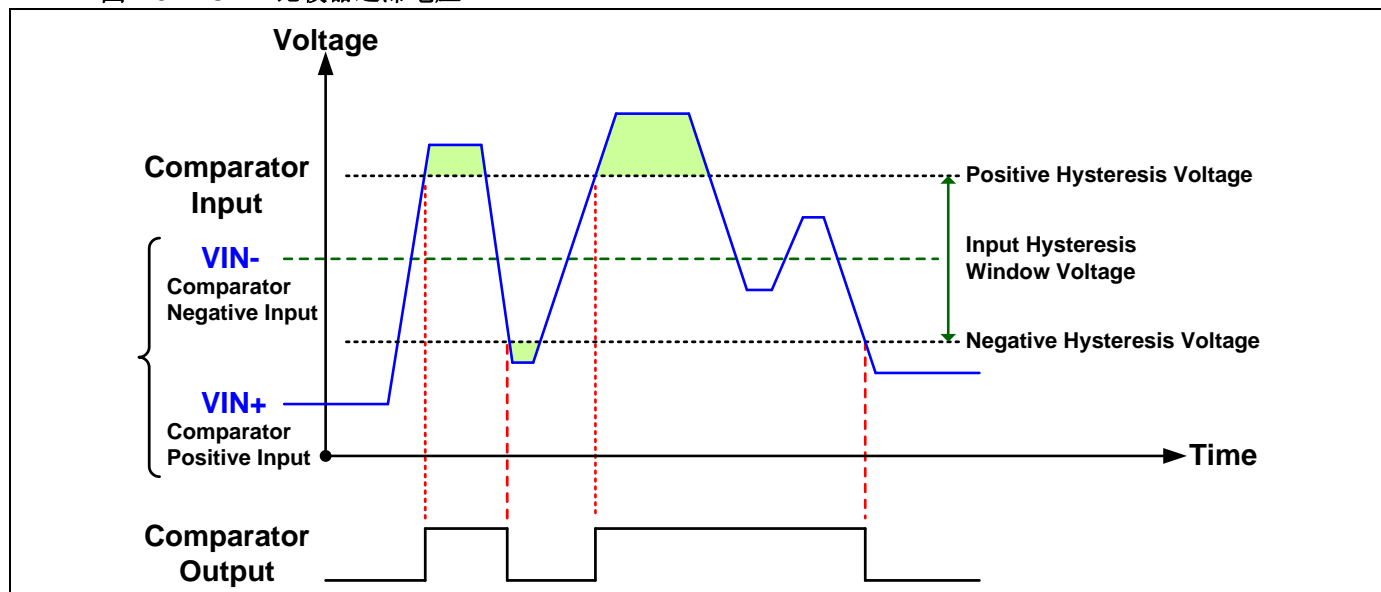
当正极输入相对于负极输入电压在迟滞电压的范围内时，模拟比较器输出无变化，并保持先前的结果。

用户可通过 **CMP_ACn_HYS** 寄存器设置输入迟滞窗口电压。当选择“不”时，迟滞电压范围为 0mV；当选择“低”时，迟滞电压范围会被设置为低延迟。请参照相关芯片数据手册获取关于迟滞电压范围的信息。

[注释]: 在 MG32F02A132/072 中, **CMPx_ACn_HYS** 寄存器不被支持且模拟比较器恒有输入迟滞。

下面的图表展示了模拟比较器输入的迟滞电压。

图 23-7. CMP 比较器迟滞电压



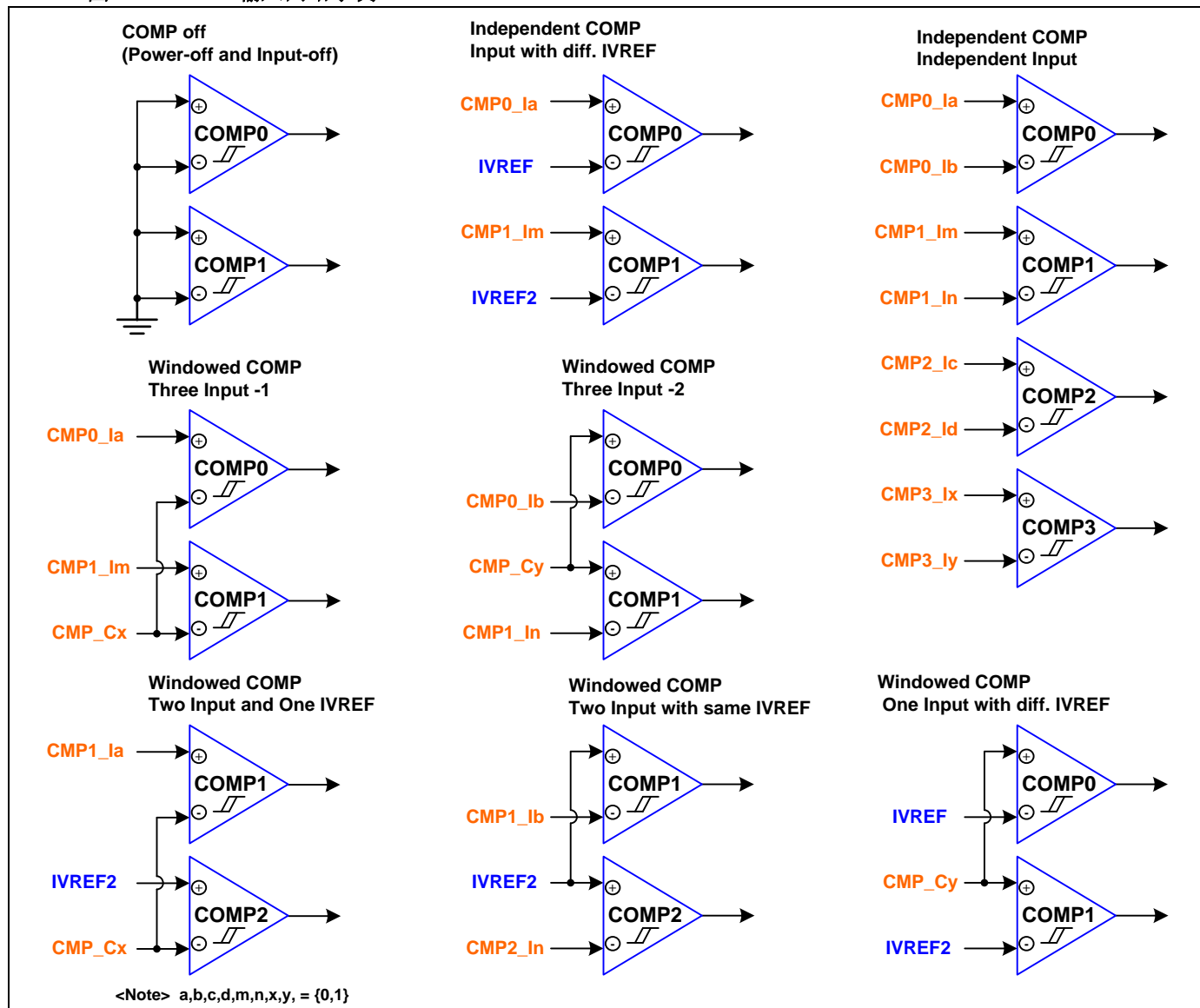
23.11. CMP 输入网络

用户可灵活地连接这些模拟比较器输入用于用户应用。

注释：标志(COMP x = 模拟比较器标号 x)用于以下描述。

下面的图表展示了模拟比较器不同应用的输入网络连接。

图 23-8. CMP 输入网络示例



[注释]: COMP2/COMP3 不支持于 MG32F02A032。

- **COMP 关闭**

短接所有正极输入和负极输入。模拟比较器将会以断电状态停止。

- **不同 IVREF 的独立 COMP 输入**

将 **IVREF** 电压连接到 COMP0 负极输入作为参考电压与外部输入电压 **CMP0_Ia** 进行比较；将 **IVREF2** 电压连接到 COMP1 负极输入作为参考电压与外部输入电压 **CMP0_Im** 进行比较。

- **独立 COMP 独立输入**

通常，所有的 COMP 可独立使用。所有 COMP 的正负极输入都与不同的外部电压源相连。

- **窗口 COMP 三输入-1**

把相同的 **CMP_Cx** 输入电压源连接到 COMP0 和 COMP1 的负极输入，把不同的 **CMP0_Ia** 和 **CMP1_Im** 输入电压源连接到 COMP0 和 COMP1 的正极输入，用户可在两个正极输入上输入两个电平电压，像 2 位 ADC 一样

来比较相同的电压。

- **窗口 COMP 三输入-2**

与窗口 COMP 三输入-1 相同，不同点在于公共输入电压 **CMP_Cy** 连接到 COMP0 和 COMP1 的正极输入上。

- **窗口 COMP 二输入和一个 IVREF**

与窗口 COMP 三输入-1 相同，不同点在于其中一个正极输入电压改成 **IVREF2** 电压。

- **窗口 COMP 二输入和相同 IVREF**

与窗口 COMP 三输入-2 相同，不同点在于公共电压改成 **IVREF2** 电压。

- **窗口 COMP 一输入和不同 IVREF**

与窗口 COMP 三输入-2 相同，不同点在于 COMP0 和 COMP1 正极输入电压改成 **IVREF** 和 **IVREF2** 电压。用户可设置 **IVREF** 和 **IVREF2** 为不同的电压电平来像 2 位 ADC 一样比较相同电压。

24. DAC (数模转换器)

24.1. 简介

ON

SLEEP

STOP

The module can be running in ON and SLEEP modes only.

该芯片内嵌 1 个含有 10 位电流模式 DAC（数模转换器）和用于输入码控制的数字逻辑的 DAC 模块。数字转换模拟的转换可通过写入数据寄存器、事件（外部引脚输入或内部事件）执行和触发启动。DAC 可以在转换速率最高 100 KHz 时输出满量程电流最大 2mA。

24.2. 特性

- 1 个 10 位电流 DAC
 - 最高转换速率为 100KHz
 - 模拟输出到 ADC 内部通道
- 通过寄存器写入、外部引脚和内部事件开始触发转换
- 可编程满量程输出电流
 - 0.5/1/2 mA
- 输入数据左对齐/右对齐的数据对齐调整
 - 可配置代码宽度：10/8 位
- 可用 DMA 缓冲输出数据

24.3. 配置

24.3.1. 芯片配置

下面的表格展示了芯片包含的 DAC 模拟输出。

表 24-1. DAC 配置

芯片	DAC 输出
	DAC0
MG32F02A132	V
MG32F02A072	V
MG32F02A032	

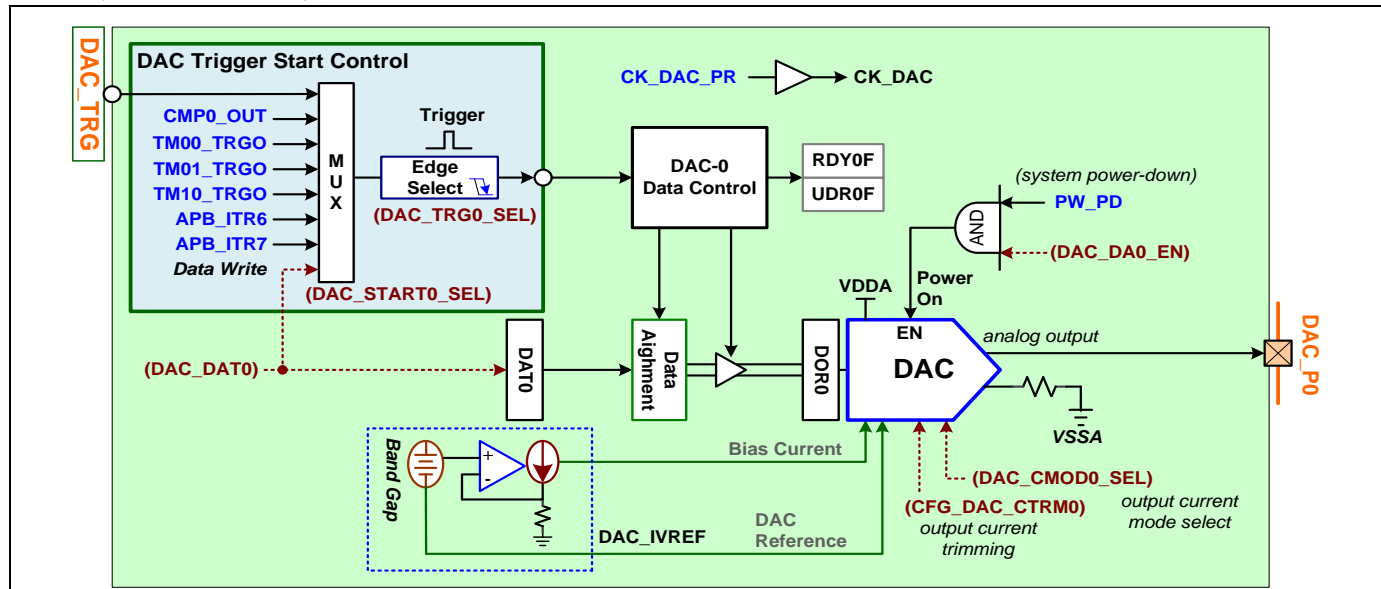
注释 V：包含

24.4. 控制块

DAC 控制块由 1 个 100Ksps/10 位电流模式 DAC、参考电压电路、1 个 DAC 数据码寄存器、1 个 DAC 转换输出寄存器（DOR0）和 DAC 转换触发启动控制块组成。

下面的图表展示了 DAC 控制块。

图 24-1. DAC 主块



24.5. IO 线

24.5.1. IO 信号

- **DAC_P0**
DAC 通道 0 的模拟输出。
- **DAC_TRG**
DAC 数据转换的外部触发输入信号。

24.5.2. IO 设置

用户必须通过设置相关的 IO 引脚来使用该模块的 IO 线。用户可以为每个引脚独立设置 IO 工作模式、高速输出选项、拉高选项、输出推力、IO 滤波和输入反相选择。使用的模拟输入 **DACP_0** 引脚必须设置 IO 模式为 **AIO 模式**。参照用户手册 GPIO 章中“**IO 模式**”节以获取更多关于 IO 模式设置的信息。

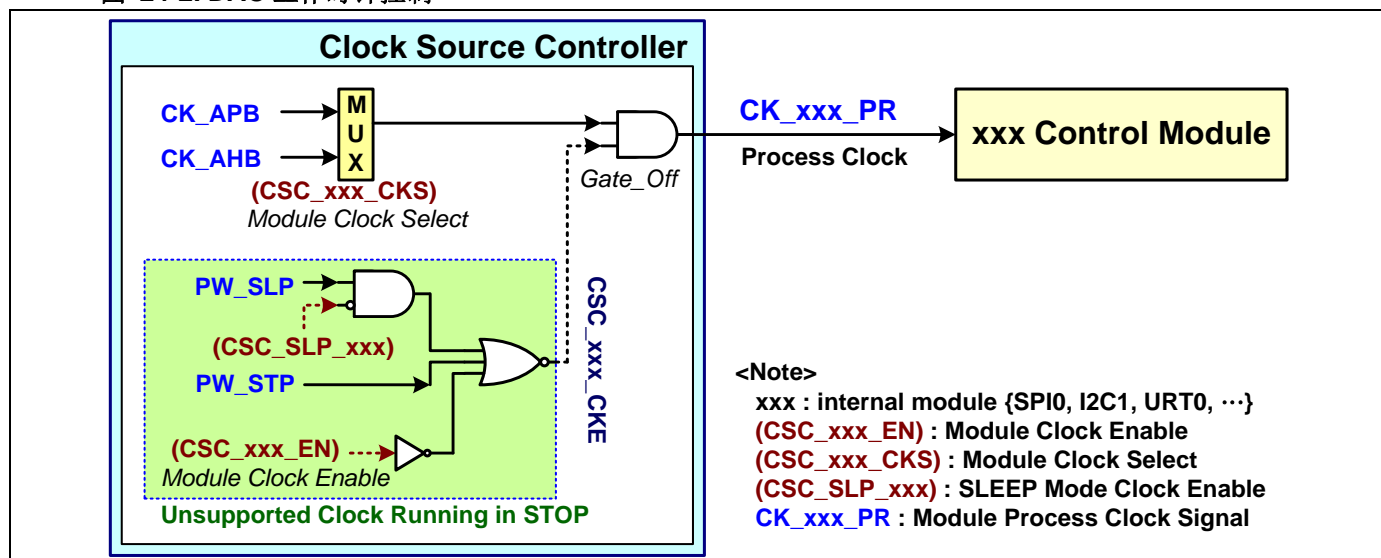
每个 IO 信号都被通过一些 IO 引脚的 IO AFS 设置进行映射和选择。参照用户手册 GPIO 章中“**功能复用选择**”节以获取更多关于 IO AFS 设置信息，参照芯片数据手册的引脚描述章中“**引脚功能复用表**”以获取更多信息。

24.6. 电源和时钟

DAC 模拟宏的工作电源 VDDA 来源于 IO 电源 **VDD** 引脚。DAC 模拟宏只有在 **DAC_DAC0_EN** 位被设置为逻辑 1 时使能，在设置为逻辑 0 时关闭。

该模块工作时钟 **CK_DAC_PR** 是用于 APB 总线和模块的接口逻辑控制。该时钟来源于 CSC（时钟源控制器）模块。该时钟可通过 **CSC_DAC_EN** 寄存器使能并通过 **CSC_DAC_CKS** 寄存器选择时钟源来自 APB 或 AHB。用户可以在芯片进入 **SLEEP** 模式之前通过设置 **CSC_SLP_DAC** 寄存器规划在 **SLEEP** 模式下是否让 DAC 时钟继续运行。参照系统时钟章以获取更多信息。

图 24-2. DAC 工作时钟控制



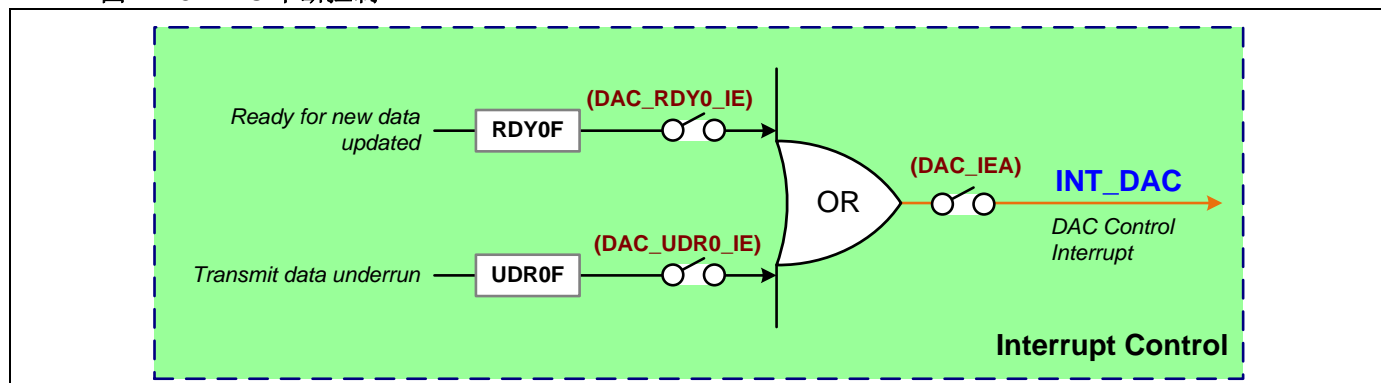
24.7. 中断和事件

24.7.1. DAC 中断控制

DAC 模块中有 1 种信号 **INT_DAC**。**INT_DAC** 发送到外部中断控制器 (EXIC) 作为中断事件。

中断标识是用于中断服务程序 (ISR) 流控制的。通常，这些中断标志被硬件置起，在相关 ISR 服务工作完成时被软件清除。每个中断标志都有 1 个中断使能位，用户可以选择使能或禁用。中断全局使能位 **DAC_IEA** 用于使能和禁用该模块的所有中断源。

图 24-3. DAC 中断控制



24.7.2. DAC 中断标志

通常，这些中断标志被硬件置起，被软件写 1 清除。参照寄存器描述以获取更多关于相关中断标志和使能位的信息。

- **UDR0F**

DAC-0 转换欠运行标志是 **(DAC_UDR0F)**，相关的中断使能寄存器位是 **DAC_UDR0_IE**。

- **RDY0F**

DAC-0 准备上传新数据到数据寄存器就绪标志是 **(DAC_RDY0F)**，相关的中断使能寄存器位是 **DAC_RDY0_IE**。

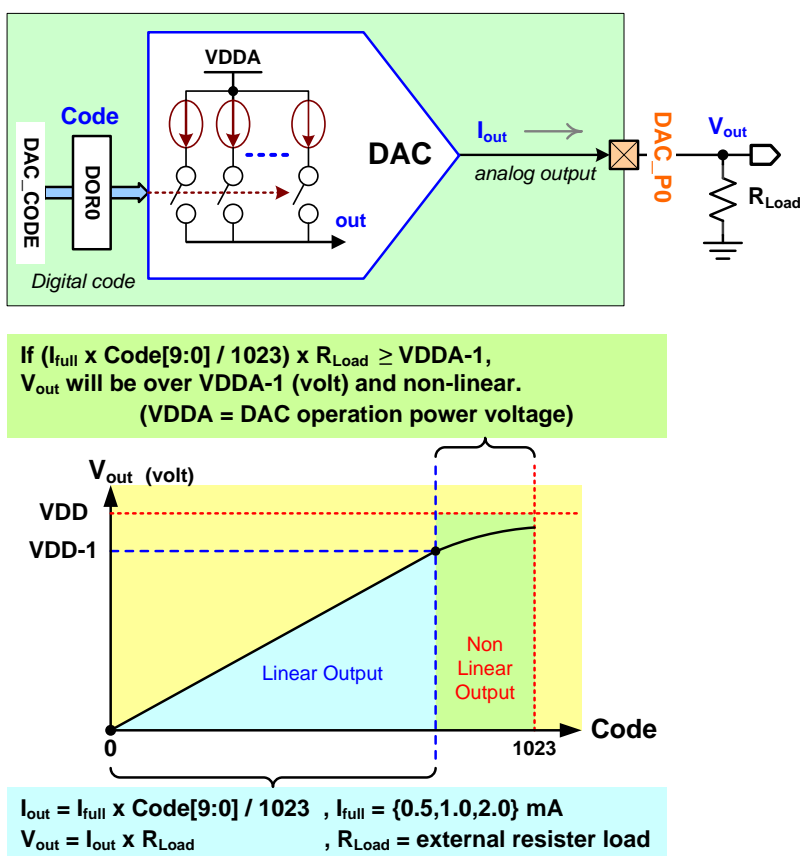
24.8. DAC 输出电压

DAC 输出可通过 **DAC_CMODO_SEL** 寄存器设置满量程输出电流为 0.5、1 或 2 mA。输出电流 I_{out} 通过以下图表公式计算。

由于 DAC 为电流模式，因此 DAC 输出一般需要连接外部电阻负载转换成电压输出。输出电压 V_{out} 通过以下图表公式计算。当输出电压低于边界电压“DAC 工作电源电压 - 1V”时，DAC 输出线性如下图中的公式所示。
当输出电压超过边界电压“DAC 工作电源电压 - 1V”时，输出为非线性。

下面的图表展示了 DAC 输出块和输出电压曲线。

图 24-4. DAC 输出电压



24.9. DAC 转换

DAC 转换可通过写入数据寄存器或事件来执行和触发启动。用户可通过设置 **DAC_START0_SEL** 寄存器设置启动触发控制源。对于事件触发，用户可通过 **DAC_TRG0_SEL** 寄存器选择外部引脚输入或内部事件并选择触发信号沿为上升沿、下降沿还是双沿。

DAC 输入码来源于 DAC 数据寄存器 **DAC_DAT0**，并发送到转换输出寄存器 **DAC_DOR0** 做输出转换。

DAC 最大转换率为 100KHz，因此 DAC 码更新间隔必须大于 10us。

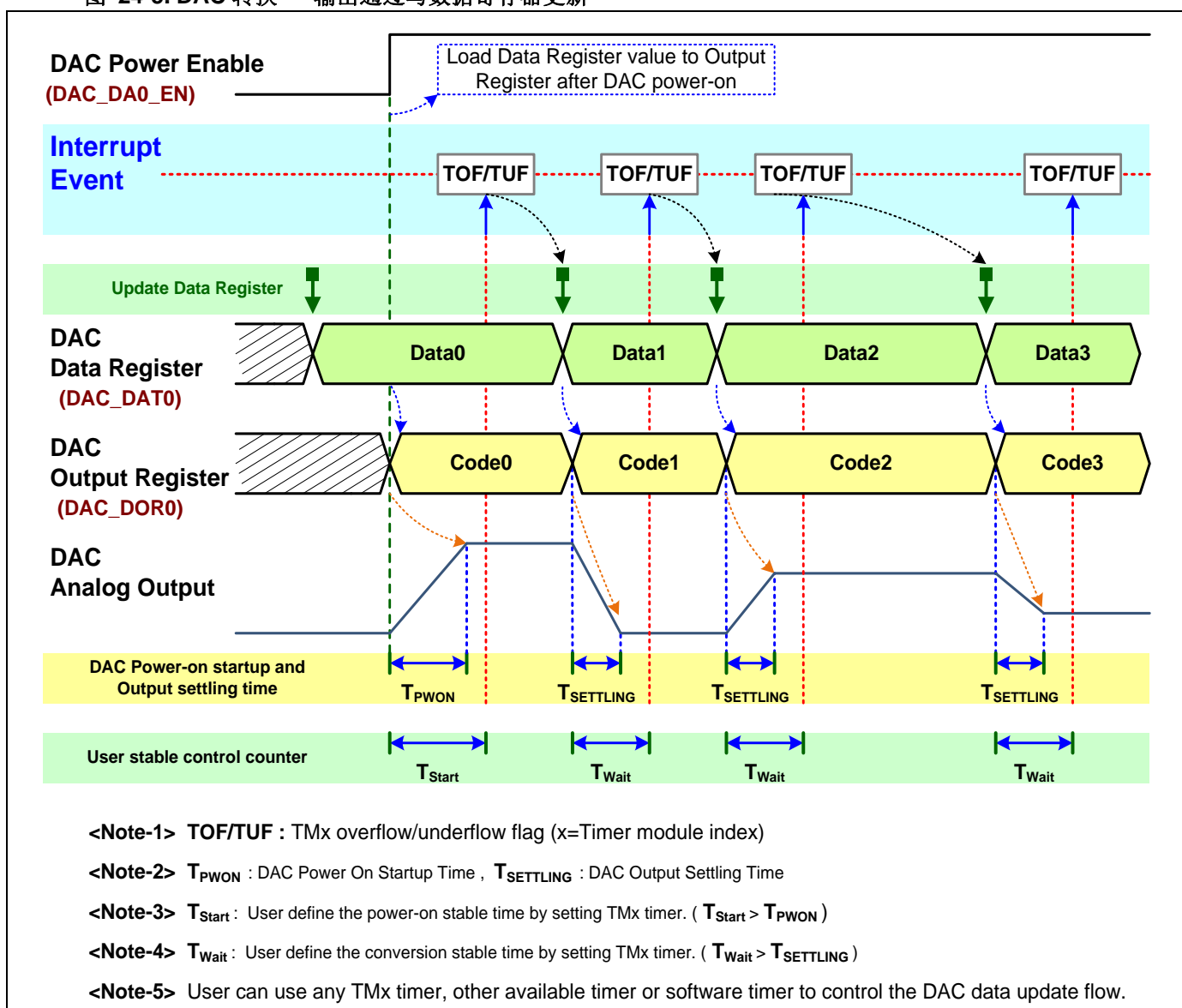
24.9.1. 数据寄存器写

DAC 转换可通过每次直接写入数据寄存器来执行和触发启动。DAC 会在数据寄存器写之后直接复制数据寄存器到 DOR。DAC 会在 DAC 启动后把数据寄存器自动复制到 DOR。

用户可使用内部定时器的定时器上溢或下溢中断控制 DAC 数据码更新间隔。

下面的图表展示了通过 TMx 定时器上溢或下溢内部事件触发启动示例。

图 24-5. DAC 转换 – 输出通过写数据寄存器更新

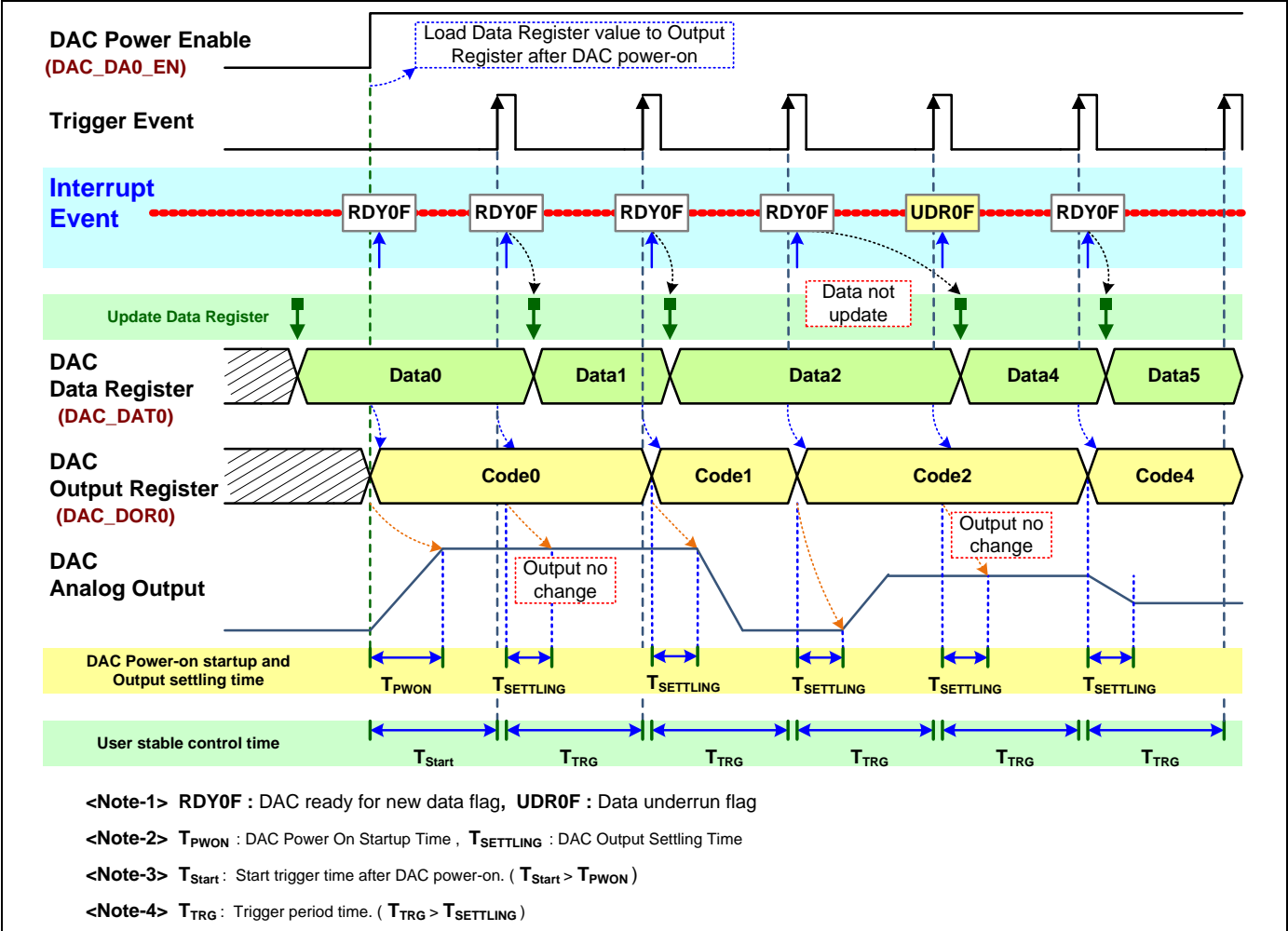


24.9.2. DAC 事件触发

DAC 转换可通过外部引脚输入 (**DAC_TRG0** 引脚) 或内部事件来执行和触发启动。**DAC_RDY0F** 会在选择的触发事件边沿置起, 然后 DAC 会复制数据寄存器的数据到 DOR, 此外, DAC 会在 DAC 启动后自动复制数据寄存器数据到 DOR。

DAC_UDR0F 会在下一个触发边沿发生, 但是数据寄存器还没更新时置起。

图 24-6. DAC 转换 – 输出通过事件触发更新



24.9.3. DAC 数据分辨率和对齐

用户可在 **DAC_RES0_SEL** 寄存器中选择数据寄存器 **DAC_DAT0** 的数据宽度为 8 位还是 10 位。此外用户还可在 **DAC_ALIGN0_SEL** 寄存器中进行输入码左/右对齐调整。

下面的表格展示了 DAC 码数据对齐格式定义和代码宽度。

表 24-2. DAC 数据对齐定义

对齐	DAC_DATn (DAC 数据寄存器 , n= {0,1})															
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
10 位-右	x	x	x	x	x	x	M DAC 输出码[9:0]									
8 位-右	x	x	x	x	x	x	x	x	M DAC 输出码[7:0]							
10 位-左	M DAC 输出码[9:0]										x	x	x	x	x	x
8 位-左	M DAC 输出码[7:0]								x	x	x	x	x	x	x	x

DAC 输出码 : 数字转模拟输出码

M : MSb 位, x : 不影响

24.10. DAC DMA 操作

24.10.1. DMA 模块设置

当芯片支持 DMA（直接内存访问）控制器时，用户可以在 DMA 数据传输前，在 DMA 模块中设置关于来源/目的设备、通道请求仲裁等 DMA 设置。DMA 来源和目的可以是内存或外设。

参照 DMA 章以获取更多关于 DMA 模组设置的细节。

24.10.2. DAC DMA 控制

DMA 设置完成后，用户需设置 DAC 模块的 DMA 使能位 **DAC_DMA_EN**。

最后，相关的通道请求起始位 **DMA_CHn_REQ** 是必须的，用于设置 DMA 传输启动(n = DMA 通道标号)。然后传输源和目的设备会置起 RX/TX 请求信号到 DMA 控制器，DMA 控制器便会置起接收信号到请求源/目的设备。此时，该数据传输连接是用于 DMA 传输的。

- **DAC 输出来自 DMA**
DAC_DMA_EN 寄存器位用于使能 DMA 从 DMA 源到 DAC 的传输输出。
DMA 发送周期中，数据就绪标志 **DAC_RDY0F** 是被硬件屏蔽的。

24.10.3. DAC 的 DMA 中断标志控制

DMA 工作周期中，模块的中断标志会控制和执行 3 种类型，如下表格。

其中一方在 DMA 工作过程中被屏蔽(RDY0F 标志)，另一方会在标志(UDR0F 标志)被置起后禁用 DMA 功能。此时，硬件会清除 **DAC_DMA_EN** 位。其他操作与 DMA 操作中不执行的操作相同。

表 24-3. DAC DMA 功能中断标志控制

行为	DMA 操作中屏蔽标志	标志置起后 DMA 被禁用(*1)	一般控制
外设	(数据溢出标志)	(错误/检测标志)	(其他标志)
DAC	DAC_RDY0F(*2)	DAC_UDR0F	

注释-1：当标志被置起，若相关中断使能位没被使能，它不会强行禁用外设 DMA。

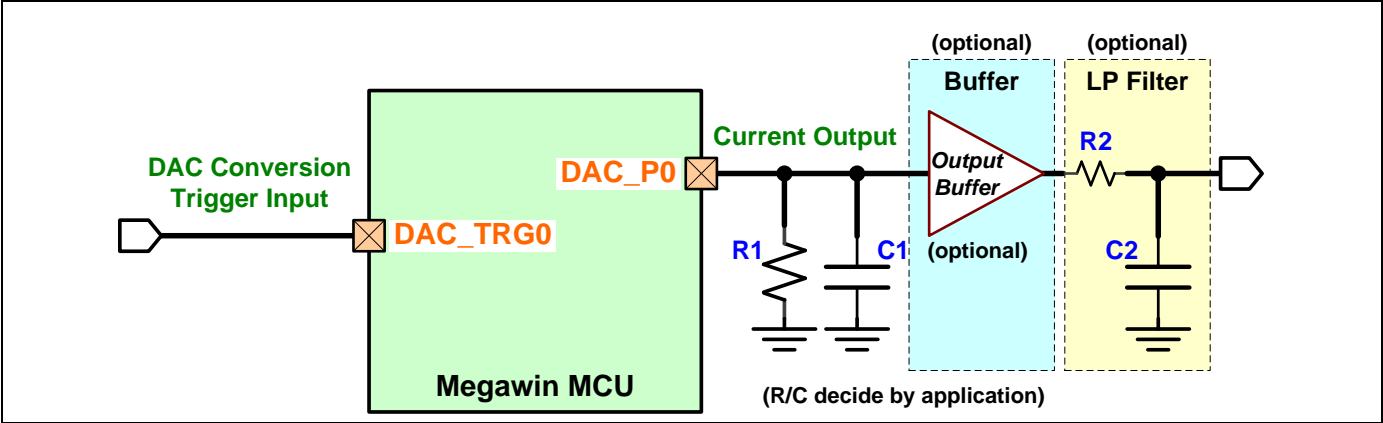
注释-2：DAC_RDY0F 会在 DMA TX 完成后被置起。

24.11. DAC 应用电路

DAC 需要外部 R-负载(**R1**)转换电流输出为电压输出。强烈建议使用外部低通滤波器(**R2/C2**)以获得更好的输出表现。

可选的 **DAC_TRG0** 引脚可输入 DAC 输出转换的触发信号。

图 24-7. DAC 应用电路



25. 版本历史

版本 2.2 (2019_1220)		章节
1	增加 VDD2/VSS2 电源引脚的描述	3.13
2	更新描述“笙泉 ARM 烧录器 U1”为“笙泉 ARM 烧录器”	8.1
3	在 GPIO 章的特性中增加 LQFP48, QFN32 和 TSSOP20 封装所支持最多的 GPIO 引脚数	9.2
4	更新图表“图 12-3. DMA 中断控制”	12.7.2
5	更新图表“图 16-7. UART RX/TX IO 控制 - MG32F02A132/072”, “图 16-8. UART RX/TX IO 控制 - MG32F02A032”, “图 16-9. UART 其他 IO 控制”及相关描述	16.8.1
6	在“16.20.2. UART 同步模式 NSS 控制”节中更新图表“图 16-36. UART 同步模式硬件 NSS 时序”和“表 17-5. SPI NSS 时序表”	16.20.2
7	在“17.10.6. SPI NSS 模式”节中增加“[注释]: 时钟模式不支持模式 1,3 当 SPIx_NSS_PEN 被禁用时于 MG32F02A132/072。”	17.10.6
8	在“17.10.6. SPI NSS 模式”节中更新图表“表 17-5. SPI NSS 时序表”和“图 17-18. SPI 主机模式硬件 NSS 时序”	17.10.6
9	更新图表“图 23-4. CMP 模拟比较器 CMP0 块”, “图 23-5. CMP 模拟比较器 CMP1/2/3 块”	23.8.3 23.8.4
版本 2.1 (2019_0805)		章节
1	在芯片和系统章更新图表“图 2-3. 芯片主块 - MG32F02A032”	2.3
2	在硬件选项章的特性中去除 BOD1_TH, HS_SEL, XOSC_EN 和 XOSC_GN 选项	8.2
3	在 GPIO 章更新“特性”节	9.2
4	在 DMA 章更新“特性”节	12.2
5	在 APB 一般控制章中修正“[注释]: MG32F02A032 只支持 1 组 OBM 块 - OBM1。”为“[注释]: MG32F02A132/072 只支持 1 组 OBM 块 - OBM1。”	14.6
6	在 I2C 章修正词“I2Cx_RDAT”和“I2Cx_TDAT”为“I2Cx_DAT”	15.7.3
7	在 I2C 章更新表格“表 15 9. I2C 信号输出驱动设置”	15.12.1
8	在定时器章修改 MG32F02A032 描述“7 个定时器/计数器”为“5 个定时器/计数器”	18.2
9	在定时器章更新表格“表 18 2. 定时器模块功能”	18.3.2
10	在 ADC 章更新节“ADC 转换设置和序列”	22.9.1
11	在 ADC 章的“简介”“ADC 转换模式”节中修改 ADC 转换模式的名称	22.1 22.10
12	在 ADC 章重命名“循环扫描转换”节为“扫描循环转换”	22.10.3
13	在 ADC 章更新节“ADC DMA 的中断标志控制”	22.14.3
版本 2.0 (2019_0514)		章节
1	增加和修改关于新加入的 MG32F02A32 芯片内容	
2	在芯片和系统章增加节“芯片配置概览”	2.2.1
3	在系统时钟章增加节“CSC 时钟状态”	5.4.2
4	在系统一般控制章增加节“系统备份寄存器”	6.5
5	去除“表 9-2. 引脚功能复用选择表”，用户可在芯片数据手册里找到“引脚功能复用选择表”	9.7.1
6	在 DMA 章增加节“DMA 传输设置和序列”、“SLEEP 模式下 DMA 传输”	12.9.1 12.9.2

7	在 I2C 章增加节“STOP 模式唤醒”、“I2C NACK 控制”	15.12.3 15.12.5
8	在 UART 章重排序“16.8. UART 模块 IO 控制”中的节或语句顺序	16.8
9	在 UART 章融合和重排序“16.12. UART 工作模式”和“16.13. UART 收发”为“16.11. UART 功能控制”	16.11
10	在 UART 章改变“UART 数据缓冲”的节顺序	16.12
11	在 UART 章改变“UART 波特率控制”、“UART 数据接收和采样”和“UART TMO 超时控制”的节顺序	16.15 16.16 16.17
12	在 UART 章增加节“UART IO 模式”、“UART 控制模式设置”、“UART 同步模式时序”和“UART 同步模式 NSS 控制”	16.8.2 16.11.1 16.20.1 16.20.2
13	在 SPI 章融合和重排序“17.10. SPI 收发”和“17.11. SPI 功能时序”为“17.11. SPI 功能控制”	17.10
14	在 SPI 章重排序“17.7.2. SPI 中断标志”中的节或语句顺序	17.7.2
15	在 SPI 章增加节“SPI DTR 模式”	17.10.5
16	在 SPI 章改变“SPI 数据缓冲”的节顺序	17.11
版本 1.8 (2019_0124)		章节
1	更改“第二计数器支持向上/向下计数控制 (Separate 模式)”成“第二计数器支持向上/向下计数控制 (Cascade /Separate 模式)”和 去除描述“外部输入定时器向上/向下计数控制(仅 TM16)”	18.2
2	更新“图 18 35. 定时器 QEI 控制块”	18.16