

**WT32L064/032**  
**闪存类型**  
**32 位微控制器**

**规格书**

**Rev. 1.02**

**February 2022**

**版权声明**

本数据表受伟詮半导体公司版权保护。未经伟詮半导体公司明确书面许可，请勿复制、转换为任何其它格式，或发送/传输本文档的任何部分。

**免责声明**

进行更改的权利， 本文文件为用户提供技术信息。伟詮半导体公司保留对此处任何产品进行更改的权利，恕不另行通知。

### 目 录

目 录.....	i
图标列表.....	vii
表格列表.....	ix
<b>1 概述 .....</b>	<b>- 1 -</b>
1.1 主要特性.....	- 2 -
1.2 系统架构方块图 7.....	- 4 -
1.3 系统电源架构 .....	- 5 -
1.4 系统时钟树 .....	- 6 -
1.5 ARM® Cortex™-M0 Processor .....	- 7 -
1.5.1 Cortex M0 处理器功能 .....	- 7 -
1.5.2 嵌入式向量中断控制器 (Nested Vectored Interrupt Controller, NVIC) .....	- 8 -
1.5.3 系统节拍定时器 (SysTick) .....	- 10 -
<b>2 封装引脚规划 .....</b>	<b>- 11 -</b>
2.1 包装.....	- 11 -
2.1.1 LQFP-64.....	- 11 -
2.1.2 LQFP-48.....	- 12 -
2.1.3 QFN-32.....	- 13 -
2.2 引脚描述.....	- 14 -
2.3 复合功能 I/O 与优先权.....	- 18 -
<b>3 内存映像 .....</b>	<b>- 19 -</b>
3.1 AMBA 总线地址映像 .....	- 19 -
3.2 APB 内存空间 .....	- 19 -
<b>4 复位和时钟控制 .....</b>	<b>- 22 -</b>
4.1 主要概述.....	- 22 -
4.2 方块图 .....	- 22 -
4.3 RCC 缓存器描述 .....	- 24 -
4.4 功能描述.....	- 29 -
4.4.1 复位 (RESET).....	- 29 -
4.4.1.1 系统复位 .....	- 29 -
4.4.1.2 电源复位 .....	- 31 -
4.4.1.3 待机离开时复位 (Standby Exit Reset) .....	- 32 -
4.4.2 时钟 (Clock).....	- 32 -
4.4.2.1 PLL 时钟 (PLL Clock) .....	- 32 -
<b>5 电源管理单元 .....</b>	<b>- 33 -</b>
5.1 电源管理单元概述 .....	- 33 -
5.2 Power Domain Overview .....	- 33 -
5.3 电压稳压器 .....	- 34 -
5.4 动态核心电压 .....	- 34 -
5.5 省电模式.....	- 35 -
5.5.1 外部中断 (Extended Interrupt, EXTI) .....	- 36 -
5.5.2 节能模式下时钟的行为 .....	- 36 -
5.6 缓存器定义.....	- 37 -
5.7 状态机 (State Machine).....	- 41 -
5.7.1 执行模式 (Run mode) .....	- 41 -
5.7.2 睡眠模式 (Sleep Mode).....	- 42 -

5.7.3	停机模式 (Stop Mode) .....	- 45 -
5.7.4	待机模式 (Standby Mode).....	- 48 -
5.8	设定范例.....	- 51 -
5.9	操作模式 与 唤醒源 .....	- 51 -
6	eFlash 控制 .....	- 53 -
6.1	主要概述.....	- 53 -
6.2	方块图 .....	- 53 -
6.3	缓存器描述.....	- 54 -
R/W1C:	读取 与 写“1”清除 .....	- 58 -
6.4	功能描述.....	- 59 -
6.4.1	NVM 功能描述 .....	- 59 -
6.4.2	读取 NVM.....	- 59 -
6.4.3	写入与抹除 NVM.....	- 63 -
6.4.4	锁定与解锁操作 .....	- 63 -
6.4.5	状态缓存器.....	- 64 -
6.5	内存保护.....	- 65 -
6.5.1	读取保护 RDP (Read Out Protection) .....	- 65 -
6.5.2	资产保护 PcROP (Proprietary Code Read-Out Protection) .....	- 66 -
6.5.3	防止意外的写入或擦除操作 .....	- 67 -
6.5.4	读取 NVM .....	- 67 -
6.5.5	保护错误 (Protection Errors).....	- 68 -
6.6	NVM 中断.....	- 68 -
6.6.1	硬件错误 (Hard Fault) .....	- 68 -
6.7	内存接口管理 .....	- 69 -
6.7.1	操作优先权 (Priority) 和进程 (Evolution) .....	- 69 -
6.7.2	操作之程序 (Sequence of Operations) .....	- 69 -
6.7.3	读取时更改等待状态数 (Wait-States) .....	- 70 -
6.8	可选字节 (Option bytes).....	- 71 -
6.8.1	Option bytes 描述 .....	- 71 -
6.8.2	保护旗标加载不匹配 .....	- 71 -
6.8.3	模拟的 EEPROM .....	- 72 -
6.8.3.1	描述 .....	- 72 -
7	DMA .....	- 73 -
7.1	概要.....	- 73 -
7.2	功能描述.....	- 74 -
7.2.1	方块图.....	- 74 -
7.2.2	AHB 主机接口 .....	- 74 -
7.2.3	AHB 从机界面 .....	- 74 -
7.2.4	FIFO 缓冲区.....	- 74 -
7.2.5	DMA 核心.....	- 74 -
7.2.6	优先权仲裁.....	- 74 -
7.2.7	控制逻辑与缓存器组 .....	- 75 -
7.3	DMA Register Table .....	- 76 -
7.3.1	中断状态/清除、通道 Busy 的全局设置 .....	- 76 -
7.3.2	DMA 信道 x 来源之 AddrESS 缓存器.....	- 79 -
7.3.3	DMA 信道 x 目的地之地址缓存器 .....	- 79 -
7.3.4	DMA 信道 x 数据数量之缓存器 .....	- 80 -
7.3.5	DMA 信道 x 设定缓存器 .....	- 80 -
8	GPIO .....	- 87 -

8.1	主要概述.....	- 87 -
8.2	方块图 .....	- 87 -
8.3	缓存器描述 .....	- 87 -
8.4	功能描述.....	- 97 -
8.4.1	泛用型 I/O .....	- 97 -
8.4.2	泛用型 I/O .....	- 98 -
8.4.3	I/O port 控制缓存器.....	- 98 -
8.4.4	I/O port 数据缓存器.....	- 98 -
8.4.5	I/O data bitwise handling.....	- 98 -
8.4.6	I/O 复用 (Alternate) 功能之输入/输出.....	- 99 -
8.4.7	输入规划 .....	- 99 -
8.4.8	输出规划 .....	- 99 -
8.4.9	复用 (Alternate) 功能规划 .....	- 100 -
8.4.10	模拟功能规划 .....	- 101 -
8.4.11	使用 HXTAL 或 LXTAL 振荡器引脚作为 GPIOs.....	- 102 -
9	USB.....	- 103 -
9.1	主要概述.....	- 103 -
9.2	方块图 .....	- 103 -
9.3	缓存器描述 .....	- 104 -
9.4	功能描述.....	- 115 -
9.4.1	主要功能模块 .....	- 115 -
9.4.2	功能端点 .....	- 115 -
9.4.3	传输 FIFOs .....	- 116 -
9.4.3.1	传输 FIFOs 概述 .....	- 116 -
9.4.3.2	传输数据集管理 .....	- 116 -
9.4.4	接收 FIFOs .....	- 118 -
9.4.4.1	接收 FIFOs 概述 .....	- 118 -
9.4.4.2	接收数据集管理 .....	- 119 -
9.4.5	Setup Token 接收 FIFO 缓冲区处理.....	- 120 -
9.4.6	暂停与恢复 (Suspend and Resume) .....	- 120 -
9.4.7	FIFO 内存地址映像.....	- 122 -
10	时钟还原系统 .....	- 123 -
10.1	主要概述.....	- 123 -
10.2	方块图 .....	- 124 -
10.3	缓存器描述 .....	- 125 -
10.4	功能描述.....	- 127 -
10.4.1	同步输入 .....	- 127 -
10.4.2	频率错误量测 .....	- 127 -
10.4.3	频率误差评估和自动修补 .....	- 128 -
10.4.4	CRS 初始化和规划 .....	- 128 -
10.4.5	CRS 省电模式 (low-power modes) .....	- 129 -
10.4.6	CRS 中断 (interrupts).....	- 129 -
11	I <sup>2</sup> C.....	- 130 -
11.1	主要概述.....	- 130 -
11.2	方块图 .....	- 131 -
11.3	缓存器描述 .....	- 131 -
11.4	功能描述.....	- 137 -
11.4.1	模式选择 .....	- 137 -
11.4.1.1	通信流程 .....	- 137 -

11.4.2	I2C 初始化 .....	- 138 -
11.4.2.1	噪声滤波 .....	- 138 -
11.4.2.2	时序 .....	- 138 -
11.4.2.3	软件复位 .....	- 138 -
11.4.3	数据传输 .....	- 138 -
11.4.3.1	接收 .....	- 138 -
11.4.3.2	传送 .....	- 138 -
11.4.3.3	硬件传输管理 .....	- 138 -
11.4.4	I2C 从机模式 .....	- 139 -
11.4.5	I2C 主机模式 .....	- 140 -
11.4.5.1	通信初始化 .....	- 140 -
11.4.5.2	主机发射模式 .....	- 140 -
11.4.5.3	主机接受模式 .....	- 141 -
11.4.6	DMA 要求 .....	- 141 -
11.4.6.1	使用 DMA 传送 .....	- 141 -
11.4.6.2	使用 DMA 接收 .....	- 141 -
12	UART .....	- 142 -
12.1	主要概述 .....	- 142 -
12.2	方块图 .....	- 143 -
12.3	缓存器描述 .....	- 143 -
12.4	功能描述 .....	- 146 -
12.4.1	分数波特率 (baud rate) 产生 .....	- 146 -
12.4.2	接收机 .....	- 147 -
12.4.3	奇偶校验控制 .....	- 148 -
12.4.4	单线半双工通信 .....	- 148 -
13	SPI .....	- 149 -
13.1	主要概述 .....	- 149 -
13.2	方块图 .....	- 150 -
13.3	缓存器描述 .....	- 150 -
13.4	功能描述 .....	- 152 -
13.5	SPI Timing Diagram .....	- 155 -
14	I2S .....	- 157 -
14.1	主要概述 .....	- 157 -
14.2	方块图 .....	- 157 -
14.3	缓存器描述 .....	- 158 -
14.4	功能描述 .....	- 159 -
14.4.1	I2S Bus 的基础 .....	- 160 -
14.4.2	左对齐模式 .....	- 160 -
14.4.3	I2S Mode .....	- 161 -
14.4.4	I2S 时钟产生器 .....	- 161 -
15	实时计数器 (Real-Time Clock) .....	- 163 -
15.1	主要概述 .....	- 163 -
15.2	方块图 .....	- 164 -
15.3	缓存器描述 .....	- 165 -
15.4	功能描述 .....	- 167 -
15.4.1	可程序设定闹钟 (Programmable Alarm) .....	- 167 -
15.4.2	周期中断 (Periodic Interrupt) .....	- 167 -
16	独立看门狗定时器 .....	- 168 -

16.1	主要概述.....	- 168 -
16.2	方块图 .....	- 168 -
16.3	缓存器描述 .....	- 169 -
16.4	功能描述.....	- 170 -
17	窗口看门狗定时器 .....	- 171 -
17.1	主要概述.....	- 171 -
17.2	方块图 .....	- 171 -
17.3	缓存器描述 .....	- 171 -
17.4	功能描述.....	- 172 -
18	PWM .....	- 174 -
18.1	主要概述.....	- 174 -
18.2	缓存器描述 .....	- 175 -
18.3	功能描述.....	- 177 -
18.3.1	独立 PWM.....	- 177 -
19	定时器 .....	- 178 -
19.1	主要概述.....	- 178 -
19.2	方块图 .....	- 179 -
19.3	缓存器描述 .....	- 179 -
19.4	功能描述.....	- 185 -
19.4.1	多个捕获和匹配引脚 .....	- 185 -
19.4.2	中断 (Interrupts).....	- 185 -
19.4.3	DMA.....	- 185 -
19.4.4	计数控制 (Count Control).....	- 185 -
19.4.5	捕捉控制 (Capture Control) .....	- 186 -
19.4.6	配对控制 (Match Control) .....	- 186 -
20	ADC .....	- 187 -
20.1	主要概述.....	- 187 -
20.2	方块图 .....	- 187 -
20.3	缓存器描述 .....	- 188 -
20.4	功能描述.....	- 190 -
20.4.1	转换功能 .....	- 190 -
20.4.1.1	ADC_SLOW=0, 当 ADC clock $\leq$ APB clock .....	- 190 -
20.4.1.2	ADC_SLOW=1, 当 ADC clock $>$ APB clock .....	- 190 -
20.4.2	转换时序 .....	- 191 -
20.4.3	模拟看门狗的窗口.....	- 191 -
21	DAC.....	- 192 -
21.1	主要概述.....	- 192 -
21.2	方块图 .....	- 192 -
21.3	缓存器描述 .....	- 193 -
22	超低功耗比较器 .....	- 194 -
22.1	主要概述.....	- 194 -
23	CRC32.....	- 195 -
23.1	主要概述.....	- 195 -
23.2	缓存器描述 .....	- 195 -
23.3	功能描述.....	- 195 -
23.4	CRC32 示例代码:使用 WT32L064/032 .....	- 196 -
24	Boot ROM & IAP.....	- 199 -

24.1	说明 .....	- 199 -
24.2	进入 Boot Code IAP 的方式 .....	- 199 -
24.3	Boot 与 IAP 内存映像 .....	- 199 -
24.4	流程图 .....	- 201 -
25	电气特性 .....	- 202 -
25.1	极限参数 .....	- 202 -
25.2	DC 电气特性 .....	- 203 -
25.2.1	电源消耗 .....	- 203 -
25.2.2	Digital I/O 电气特性 .....	- 204 -
25.2.3	LVR 电气特性 .....	- 204 -
25.2.4	PLL 电气特性 .....	- 205 -
25.2.5	ADC 电气特性 .....	- 205 -
25.2.6	DAC 电气特性 .....	- 206 -
25.2.7	LDO 电气特性 .....	- 207 -
25.2.8	HSI 16 MHz .....	- 207 -
25.2.9	MSI 4.2 MHz .....	- 208 -
25.2.10	IRC 48MHz 电气特性 .....	- 209 -
25.2.11	IRC 37kHz 电气特性 .....	- 209 -
25.2.12	Crystal 32768 电气特性 .....	- 210 -
25.2.13	COMP 电气特性 .....	- 210 -
25.2.14	BOR/PVD 电气特性 .....	- 211 -
25.2.15	POR 电气特性 .....	- 212 -
25.2.16	POWER Supply 电气特性 .....	- 213 -
25.3	AC 电气特性 .....	- 214 -
25.4	Thermal 电气特性 .....	- 215 -
26	应用线路 .....	- 216 -
26.1	VDD33 power supply .....	- 216 -
26.2	VDDBat power supply .....	- 216 -
26.3	USB Vbus 5V power supply .....	- 216 -
26.4	VBAT & VDD33 dual power supply .....	- 217 -
27	产品命名规则 .....	- 218 -
28	订单讯息 .....	- 219 -
28.1	WT32L064 .....	- 219 -
28.1.1	Top Marking – LQFP64/48 .....	- 219 -
28.1.2	Top Marking – QFN32 .....	- 219 -
28.2	WT32L032 .....	- 220 -
28.2.1	Top Marking – LQFP64/48 .....	- 220 -
28.2.2	Top Marking – QFN32 .....	- 220 -
29	封装外观 .....	- 221 -
29.1	LQFP-64 外观图示 .....	- 221 -
29.2	LQFP-48 外观图示 .....	- 222 -
29.3	QFN-32 外观图示 .....	- 223 -
30	开发工具 .....	- 224 -
31	版本更改纪录 .....	- 225 -

图标列表

图 1	WT32L064/032 系统功能方块图.....	- 1 -
图 2	系统方块图 .....	- 4 -
图 3	系统电源架构.....	- 5 -
图 4	系统时钟树 .....	- 6 -
图 5	ARM® Cortex™-M0 处理器.....	- 7 -
图 6	Cortex-M0 NVIC 向量.....	- 8 -
图 7	Cortex M0 SysTick .....	- 10 -
图 8	RCC 模块复位图.....	- 22 -
图 9	RCC 模块时钟方块图 .....	- 23 -
图 10	PLL Diagram .....	- 32 -
图 11	电源域概观 .....	- 33 -
图 12	低功耗运行模式状态机 .....	- 41 -
图 13	睡眠模式状态机 .....	- 42 -
图 14	进入睡眠模式的时序图 .....	- 43 -
图 15	退出睡眠模式的时序图 .....	- 44 -
图 16	停机模式状态机 .....	- 45 -
图 17	进入停机模式的时序图 .....	- 46 -
图 18	退出停机模式的时序图 .....	- 47 -
图 19	待机模式状态机 .....	- 48 -
图 20	进入待机模式的时序图 .....	- 49 -
图 21	退出待机模式的时序图 .....	- 50 -
图 22	断电流程图 .....	- 51 -
图 23	eFlash 控制方块图.....	- 53 -
图 24	一个内部缓冲区的结构 .....	- 60 -
图 25	内存保护 .....	- 65 -
图 26	DMA 方块图.....	- 74 -
图 27	GPIO 方块图.....	- 87 -
图 28	I/O 端口位的基本结构 .....	- 97 -
图 29	输入规划 .....	- 99 -
图 30	输出规划 .....	- 100 -
图 31	复用功能规划 .....	- 101 -
图 32	模拟规划 .....	- 102 -
图 33	USB 方块图 .....	- 103 -
图 34	USB Endpoints FIFO.....	- 104 -
图 35	传输 FIFO 缓冲区概观 (以 8 字节 FIFO 为例) .....	- 116 -
图 36	接收 FIFO 概观 (以 8 字节 FIFO 为例).....	- 118 -
图 37	暂停 (Suspend)和恢复 (Resume)状态图 .....	- 121 -
图 38	CRS 方块图 .....	- 124 -
图 39	CRS Counter Behavior.....	- 127 -
图 40	I <sup>2</sup> C 方块图.....	- 131 -
图 41	I2C Bus Protocol .....	- 137 -
图 42	UART 方块图 .....	- 143 -
图 43	波特率发生器示例.....	- 147 -
图 44	SPI module 方块图 .....	- 150 -
图 45	I <sup>2</sup> S 方块图 .....	- 157 -
图 46	(a) I2S 主机模式 (b) I2S 从机模式.....	- 160 -
图 47	I2S 接口的基本计时图 .....	- 160 -
图 48	I2S 接口左对齐模式计时图 .....	- 161 -



图 49	I2S 接口的 I2S 模式计时图 .....	- 161 -
图 50	RTC 方块图 .....	- 164 -
图 51	IWDT 方块图 .....	- 168 -
图 52	WWDT 方块图 .....	- 171 -
图 53	Timing Diagram of WWDT .....	- 173 -
图 54	PWM 方块图 .....	- 174 -
图 55	PWM0~3 功能 .....	- 177 -
图 56	Timer Module 方块图 .....	- 179 -
图 57	ADC 方块图 .....	- 187 -
图 58	ADC_SLOW=0 转换顺序 .....	- 190 -
图 59	ADC_SLOW_1 的转换顺序 .....	- 190 -
图 60	ADC 转换时间 .....	- 191 -
图 61	ADC_SLOW=0 时序图 .....	- 191 -
图 62	模拟看门狗窗口 .....	- 191 -
图 63	DAC 方块图 .....	- 192 -
图 64	DAC 模拟方块图 .....	- 192 -
图 65	WT32L064/032 开发工具包 .....	- 224 -

### 表格列表

表格 1	WT32L064/032 中断和异常向量.....	- 9 -
表格 2	引脚计数表 .....	- 14 -
表格 3	引脚分配表 .....	- 15 -
表格 4	RCC 控制缓存器.....	- 24 -
表格 5	性能与 V <sub>CORE</sub> 范围.....	- 34 -
表格 6	节能模式摘要 .....	- 35 -
表格 7	EXTI Lines Connections.....	- 36 -
表格 8	PMU 控制缓存器.....	- 37 -
表格 9	操作模式上的外围.....	- 52 -
表格 10	eFlash 控制缓存器表 .....	- 54 -
表格 11	NVM 组织 .....	- 59 -
表格 12	等待状态数(wait-states).....	- 59 -
表格 13	预取-预缓冲管理 (Pre-fetch & Pre-buffer) .....	- 61 -
表格 14	缓冲区和推测读取的规划.....	- 62 -
表格 15	PcROP & wrprot 闪存保护范围 .....	- 66 -
表格 16	内存访问与模式 .....	- 67 -
表格 17	闪存中断要求.....	- 68 -
表格 18	API 菜单.....	- 71 -
表格 19	GPIO 控制缓存器.....	- 87 -
表格 20	USB 控制缓存器 .....	- 104 -
表格 21	写入字节计数缓存器 .....	- 117 -
表格 22	传输 FIFO 管理真值表 .....	- 118 -
表格 23	Status of the Receive FIFO Data Set.....	- 119 -
表格 24	接收 FIFO 管理的真值表.....	- 120 -
表格 25	FIFO 内存地址映像.....	- 122 -
表格 26	CRS 控制缓存器 .....	- 125 -
表格 27	低功耗模式对 CRS 的影响 .....	- 129 -
表格 28	CRS 中断控制位 .....	- 129 -
表格 29	I <sup>2</sup> C 控制缓存器.....	- 131 -
表格 30	UART 控制缓存器.....	- 143 -
表格 31	UART Baud Rate Table .....	- 146 -
表格 32	UART 噪声误差 .....	- 148 -
表格 33	同位(Parity)控制表.....	- 148 -
表格 34	SPI 控制缓存器 .....	- 150 -
表格 35	I <sup>2</sup> S 控制缓存器 .....	- 158 -
表格 36	I2S Sampling Rate .....	- 162 -
表格 37	I2S System Clock = 24MHz (assume f <sub>BCLK</sub> =64fs) .....	- 162 -
表格 38	I2S System Clock = 12MHz (assume f <sub>BCLK</sub> =64fs) .....	- 162 -
表格 39	RTC 控制缓存器 .....	- 165 -
表格 40	IWDT 控制缓存器 .....	- 169 -
表格 41	IWDT Min./Max. 超时周期 使用 37 kHz 时钟源 .....	- 170 -
表格 42	WWDT 控制缓存器 .....	- 171 -
表格 43	WWDT 最小值/最大超时值(Timeout), 24.00MHz .....	- 173 -
表格 44	PWM 控制缓存器.....	- 175 -
表格 45	Timer 控制缓存器 .....	- 179 -
表格 46	ADC 控制缓存器 .....	- 188 -
表格 47	DAC 控制缓存器.....	- 193 -
表格 48	CRC32 控制缓存器.....	- 195 -

### 1 概述

WT32L064/032 是一款采用 Arm 32 位 Cortex M0 中央处理器的超低功耗微控制器，具有 64KB/32KB 嵌入式闪存和 8KB SRAM。藉由自行研发的零等待(Zero Wait-State)的闪存(flash memory)存取技术，当操作在 32MHz，可达到接近 32 MIPS 的效能。

此外，藉由先进电路设计和极低漏电的半导体制程技术，其漏电电流在停机(Stop)模式和待机(Standby)模式下，分别可低至 0.6µA 和 0.3µA。

WT32L064/032 是高度整合的 MCU，包含丰富的外围，如 12 位 ADC、12 位 DAC、CRC32、GPIO、PWM、全速 USB、I2S、I2C、UART、SPI 等，特别适合于电池或能源采集供电的场合应用。

WT32L064/032 的一些应用范例包括真无线耳机(TWS)充电盒、双模无线电竞键盘/鼠标、工业流量计、各种传感应用，例如气体、温度、湿度和压力的侦测/显示设备以及传感集线器(Sensor Hub)。

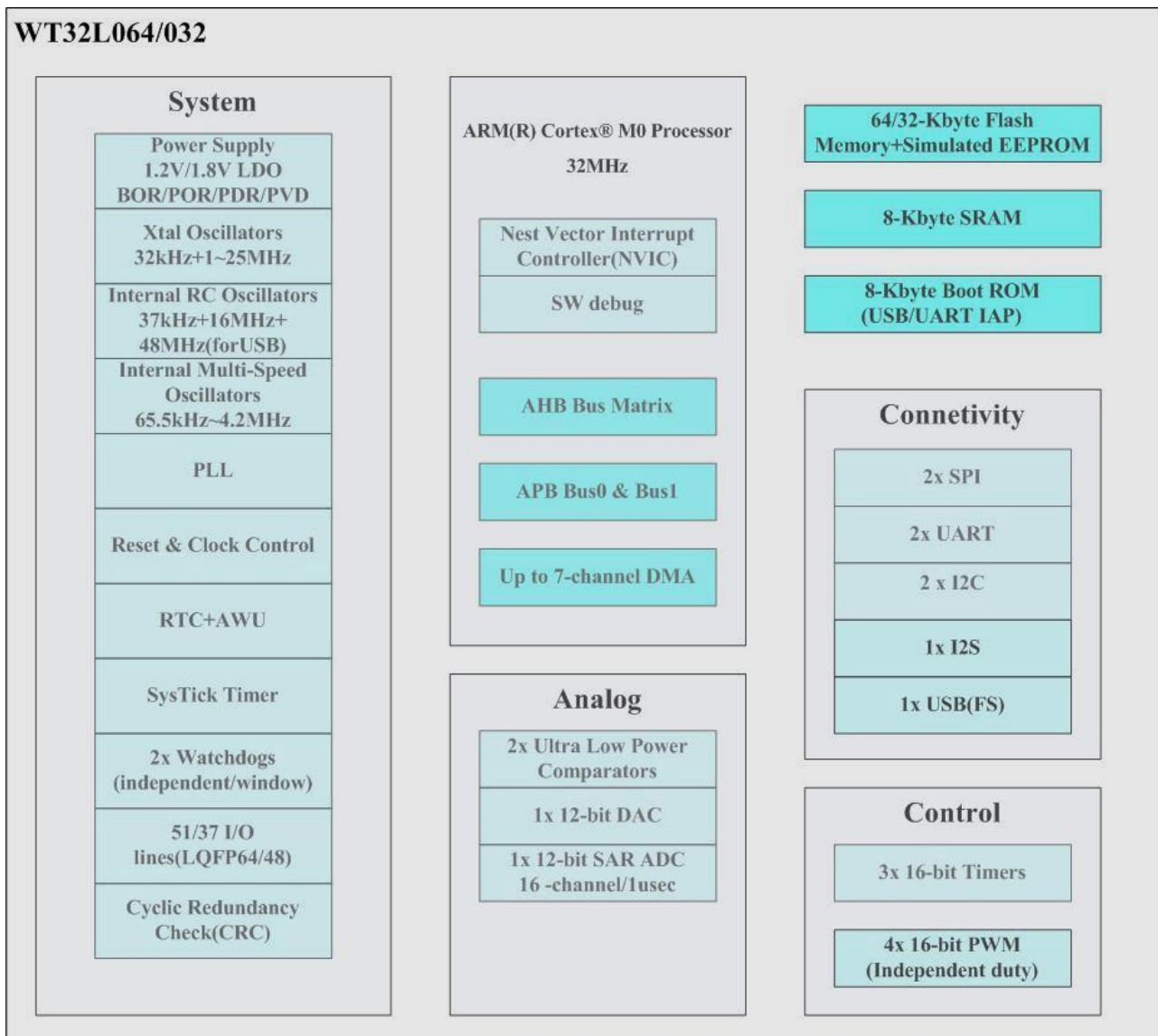


图 1 WT32L064/032 系统功能方块图

## 1.1 主要特性

- 超低功耗平台
    - 1.65V 至 3.6V 操作电压
      - ✓ 1.65V~1.8V @Max.4.2 MHz (Flash 仅能读)
      - ✓ 1.8V~2.2V @Max.4.2 MHz (Flash 可读可写)
      - ✓ 2.3V~3.6V @Max.32 MHz (Flash 可读可写)
    - -40° C 至 105° C 的操作温度
    - 0.3 μA 待机 (Standby)模式
    - 1.0 μA 打开 RTC 的待机 (Standby)模式
    - 0.6 μA 停机模式 (Stop Mode)
    - 1.3 μA 打开 RTC 的停机模式 (Stop Mode)
  - 核心: ARM® 32 位 Cortex®-M0 处理器
    - 操作频率 65.5 kHz 至 32 MHz
    - 0.88 DMIPS/MHz
  - 中断源
    - ARM Cortex-M0 内置嵌入式向量中断控制器 (NVIC)
    - 32 个外部中断输入, 每个输入具有四个优先级
    - 专用不可屏蔽中断 (NMI) 输入
    - 支持电平触发 (level-sensitive)和脉冲触发 (pulse-sensitive)中断线路
    - 唤醒中断控制器 (WIC), 支持睡眠模式
  - 系统节拍定时器
    - 24 位定时器时钟可固定到系统时钟的频率
  - 内部存储器
    - WT32L064: 高达 64 KB 闪存
    - WT32L032: 高达 32 KB 闪存
    - 8 KB SRAM 内存
  - 高达 51 个快速 I/O (大部份 I/O 可容忍 5V 输入)
  - 复位管理
    - 超低功耗具 5 个门坎可设定的 BOR (Brownout Reset)
    - 超低功耗 POR/PDR
    - 可程序化的电压侦测 (PVD)
  - 时钟源
    - 1 MHz ~ 25 MHz 外挂晶振
    - 32 kHz 外挂晶振支持实时时钟与万年历功能。(注意: RTC clock 必须先启动, 请参考 RCC 章节表格 4 描述)
    - 高速 16 MHz 内部 RC 振荡器出厂校正 (±1%)
-

- 低功耗 37 kHz 内部 RC 振荡器
- 低功耗多速 65 kHz 至 4.2 MHz 内部 RC 振荡器
- 具内部自动校准 48 MHz 内部 RC 振荡器给 USB 使用
- CPU 时钟的 PLL
- 开发支持
  - 支持串行线路调试
- 丰富的模拟外围
  - 12-bit SAR ADC **0.5 Ms/s** 最多 16 个通道 (可低至 1.8V 工作)
  - 具有输出缓冲器的 12 bit 通道 DAC (可低至 1.8V 工作)
  - 超低功耗比较器 (窗口模式和唤醒功能, 可低至 1.65V 工作)
- 7 通道 DMA 控制器, 支持 ADC、SPI、I<sup>2</sup>C、UART、定时器和 USB
- 外围通信接口
  - 1x USB 1.1 无需晶振
  - 2x UART
  - Up to 2x SPI 16 M bits /sec.
  - 2x I<sup>2</sup>C
  - 1x I<sup>2</sup>S up to 32-bit word size
- 6 个定时器: 3 组 16-bit, 支持最多 4 个通道、1 组 RTC 和 2 组 看门狗 (独立型与窗口型)
- 4 通道 PWM
- CRC 循环冗余计算单元
- LQFP-64 (7mmx7mm)、LQFP-48 (7mmx7mm)、QFN32 (5mmx5mm) 封装

1.2 系统架构方块图 7

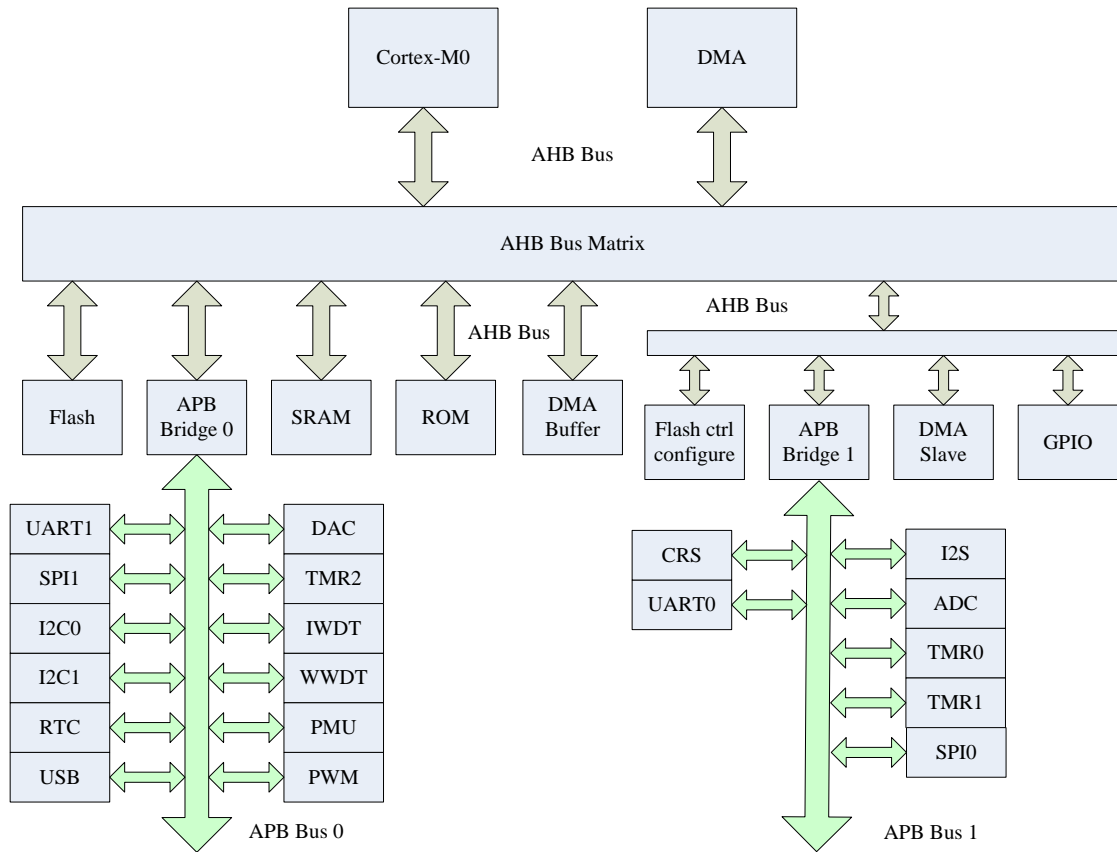
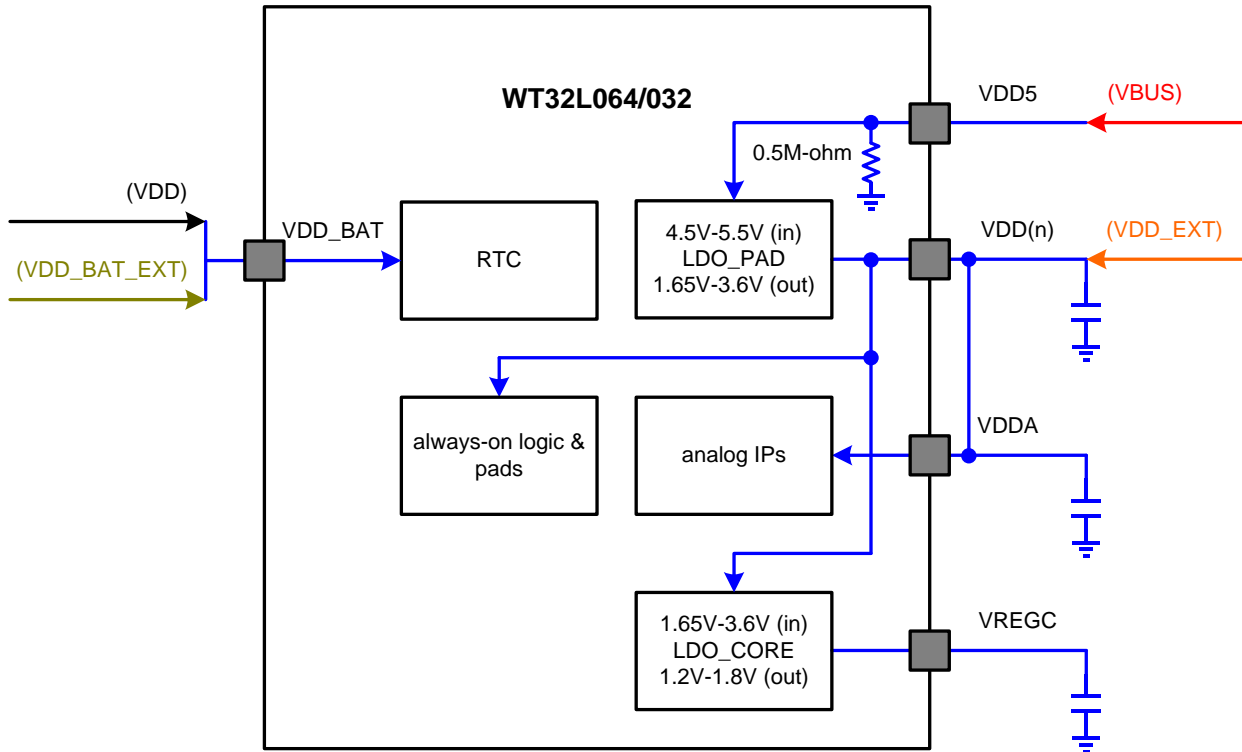


图 2 系统方块图

1.3 系统电源架构



Notes:

1. blue circuits are fixed
2. only one of VBUS or VDD\_EXT is supplied, the other is floating
3. only one of VDD or VDD\_BAT\_EXT is supplied for VDD\_BAT, the other is floating

图 3 系统电源架构

1.4 系统时钟树

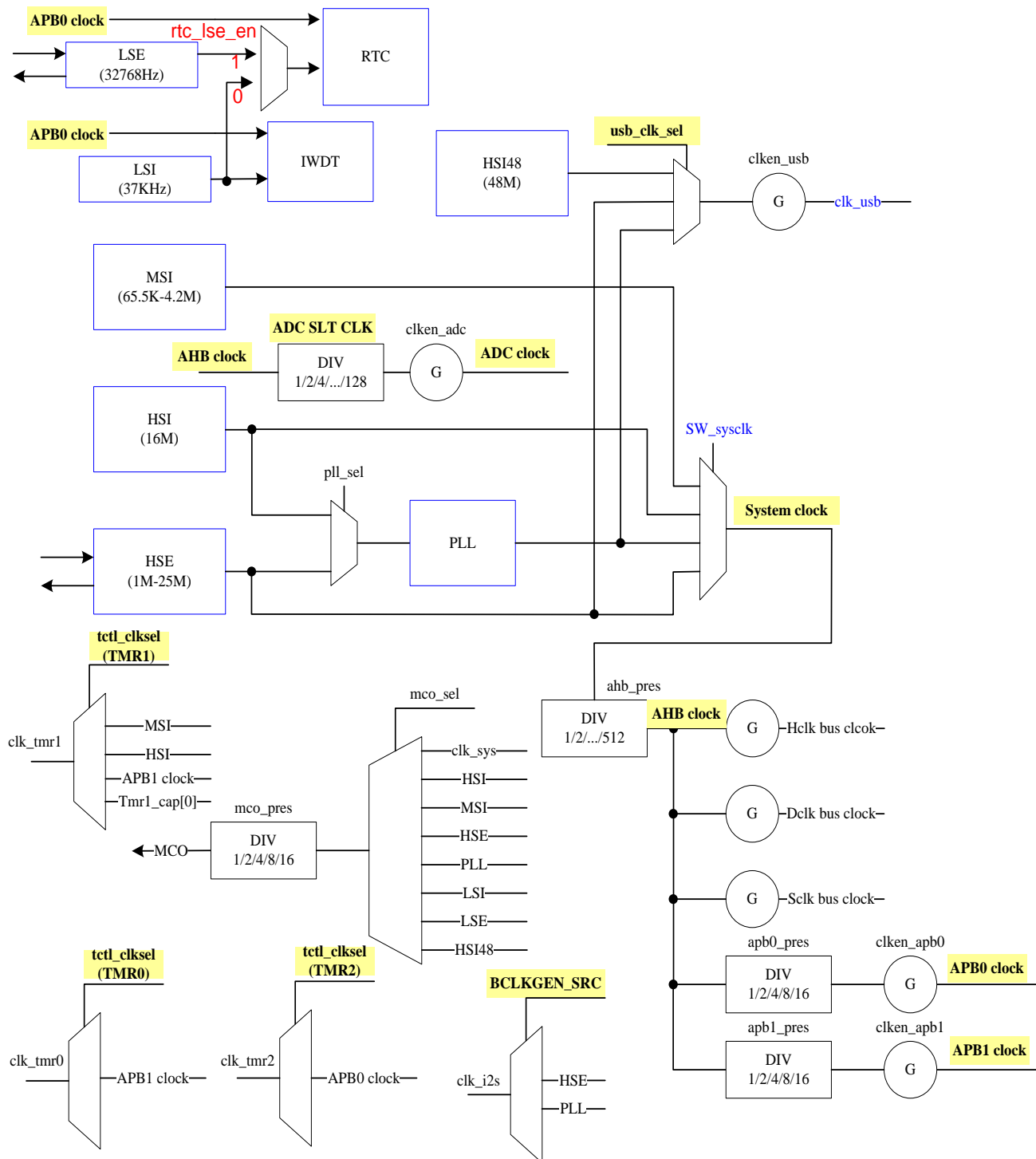


图 4 系统时钟树



### 1.5 ARM® Cortex™-M0 Processor

Cortex™-M0 处理器是一个可规划的,多级的 32 位 RISC 处理器。它有一个 AMBA AHB-Lite 接口, 并包括一个 NVIC 组件。它还具有可选的硬件调试功能。处理器可以执行 Thumb 代码, 并且与其它 Cortex-M 家族处理器兼容。

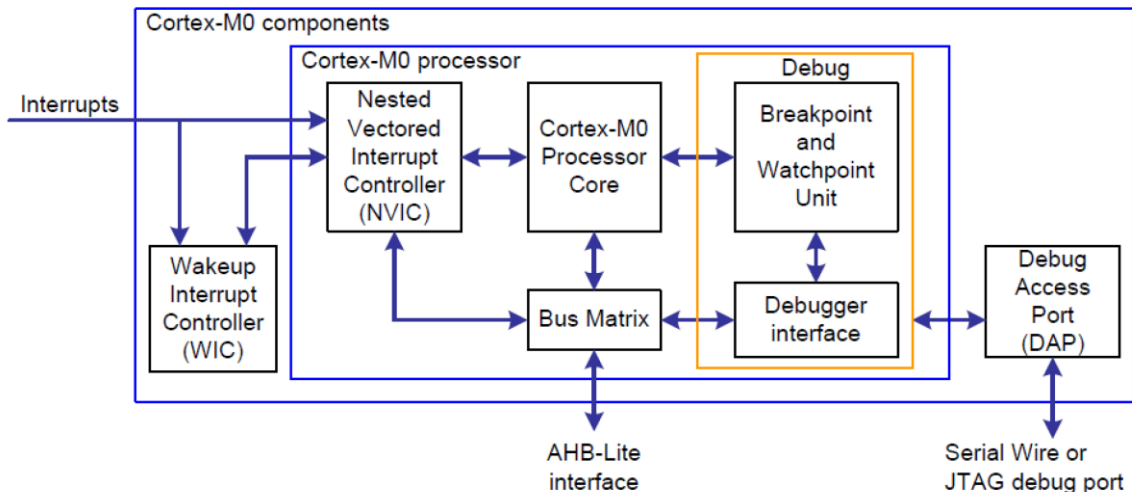


图 5 ARM® Cortex™-M0 处理器

#### 1.5.1 Cortex M0 处理器功能

- Cortex-M0 处理器包括:
  - ✓ Cortex-M0 处理器内核
  - ✓ 嵌入式向量中断控制器 (NVIC)
  - ✓ 系统节拍定时器
- ARMv6-M Thumb 指令集
- NVIC: 32个外部中断输入
- 调试: 4 个 HD 断点, 2 个观察点
- 总线接口: 32 位 AMBA-3 AHB-Lite 系统接口
- ARMv6-M (这是ARMv7-M的子集, 向上相容)
  - ✓ 它仅支持 Thumb 指令集
  - ✓ 无需互通代码 (Interworking code)
  - ✓ 总共有56个指令, 6个是 32位长度, 其它是16 位长度
  - ✓ 32位指令有: BL、DMB、DSB、ISB、MRS、MSR
- 支持字节 (8位)、半字half-word (16位) 和字组 (32位)数据类型, 每种数据类型都必须使用自然对齐方式进行访问
- 内置定时器和中断控制器
  - ✓ SysTick、NVIC



表格 1 WT32L064/032 中断和异常向量

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	保留	0x0000 0000
-	-3	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	不可屏蔽中断。RCC 时钟安全系统 (CSS) 是连结的 NMI 向量。	0x0000 0008
-	-1	fixed	HardFault	All class of fault	0x0000 000C
-	3	settable	SVCall	通过 SWI 指令进行系统服务呼叫	0x0000 002C
-	5	settable	PendSV	系统服务的可预订请求	0x0000 0038
-	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	UART0	UART0 全局中断	0x0000 0040
1	8	settable	UART1	UART1 全局中断	0x0000 0044
2			保留		0x0000 0048
3	10	settable	I2C0	I2C0 全局中断	0x0000 004C
4	11	settable	I2C1	I2C1 全局中断	0x0000 0050
5	12	settable	PWM	PWM 全局中断	0x0000 0054
6	13	settable	SPI0	SPI0 全局中断	0x0000 0058
7	14	settable	SPI1	SPI1 全局中断	0x0000 005C
8	15	settable	WWDT	WWDT 全局中断	0x0000 0060
9	16	settable	TMR0	TMR0 全局中断	0x0000 0064
10	17	settable	TMR1	TMR1 全局中断	0x0000 0068
11	18	settable	I2S_RX	I2S RX 全局中断	0x0000 006C
12	19	settable	I2S_TX	I2S TX 全局中断	0x0000 0070
13	20	settable	USB0	USB0 全局中断	0x0000 0074
14	21	settable	USB1	USB1 全局中断	0x0000 0078
15	22	settable	ADC	ADC 全局中断	0x0000 007C
16	23	settable	RTC	RTC 全局中断	0x0000 0080
17			保留		0x0000 0084
18	25	settable	CRS	CRS 全局中断	0x0000 0088
19	26	settable	TMR2	TMR2 全局中断	0x0000 008C
20	27	settable	PVD	PVD 全局中断	0x0000 0090
21	28	settable	CMP0	CMP0 全局中断	0x0000 0094
22	29	settable	CMP1	CMP1 全局中断	0x0000 0098
23	30	settable	eFlash_CTRL	eFlash controller 全局中断	0x0000 009C
24	31	settable	DMA0	DMA channel 0 interrupt	0x0000 00A0
25	32	settable	DMA1	DMA channel 1 interrupt	0x0000 00A4

Position	Priority	Type of priority	Acronym	Description	Address
26	33	settable	DMA2	DMA channel 2 interrupt	0x0000 00A8
27	34	settable	DMA3	DMA channel 3 interrupt	0x0000 00AC
28	35	settable	DMA4	DMA channel 4 interrupt	0x0000 00B0
29	36	settable	DMA5	DMA channel 5 interrupt	0x0000 00B4
30	37	settable	DMA6	DMA channel 6 interrupt	0x0000 00B8
31	38	settable	GPIO	GPIO 全局中断	0x0000 00BC

### 1.5.3 系统节拍定时器 (SysTick)

Cortex-M0 包括一个集成的系统节拍定时器 SysTick 提供了一个简单的 24 位写入(clear-on-write)、递减、复归 (wrap-on-zero)计数器，具有灵活的控制机制。计数器可用作实时操作系统(RTOS)的节拍定时器或简单计数器。

- SysTick: 24位写入时清除、递减、零包装计数器
- 用作实时操作系统 (RTOS) 节拍定时器或简单计数器
- 启动后，从 SysTick 当前值缓存器 (SYST\_CVR) 倒数到零，然后重载 SysTick 重载值缓存器 (SYST\_RVR)，然后继续递减
- 当倒数为零时，产生中断旗标，COUNTFLAG=1，否则COUNTFLAG=0
- 计数FLAG=0，在读取时。SYST\_CVR 值在复位时为“未知”
- SYST\_RVR=0，定时器=0 (关闭定时器，即使启动定时器)

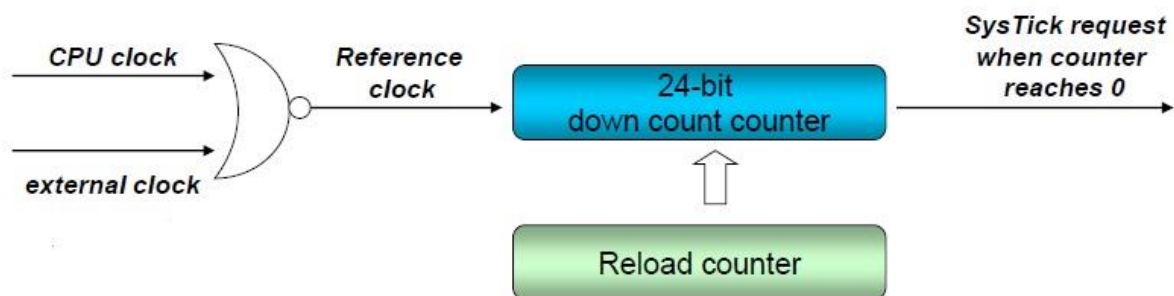
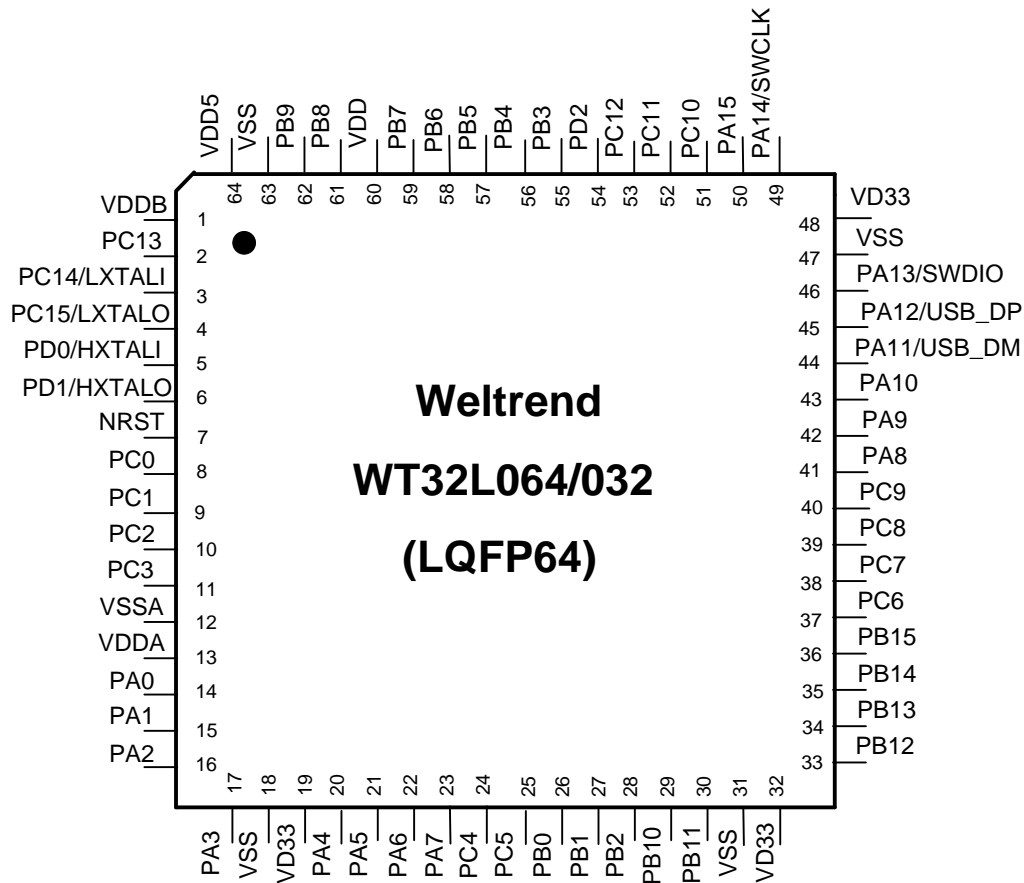


图 7 Cortex M0 SysTick

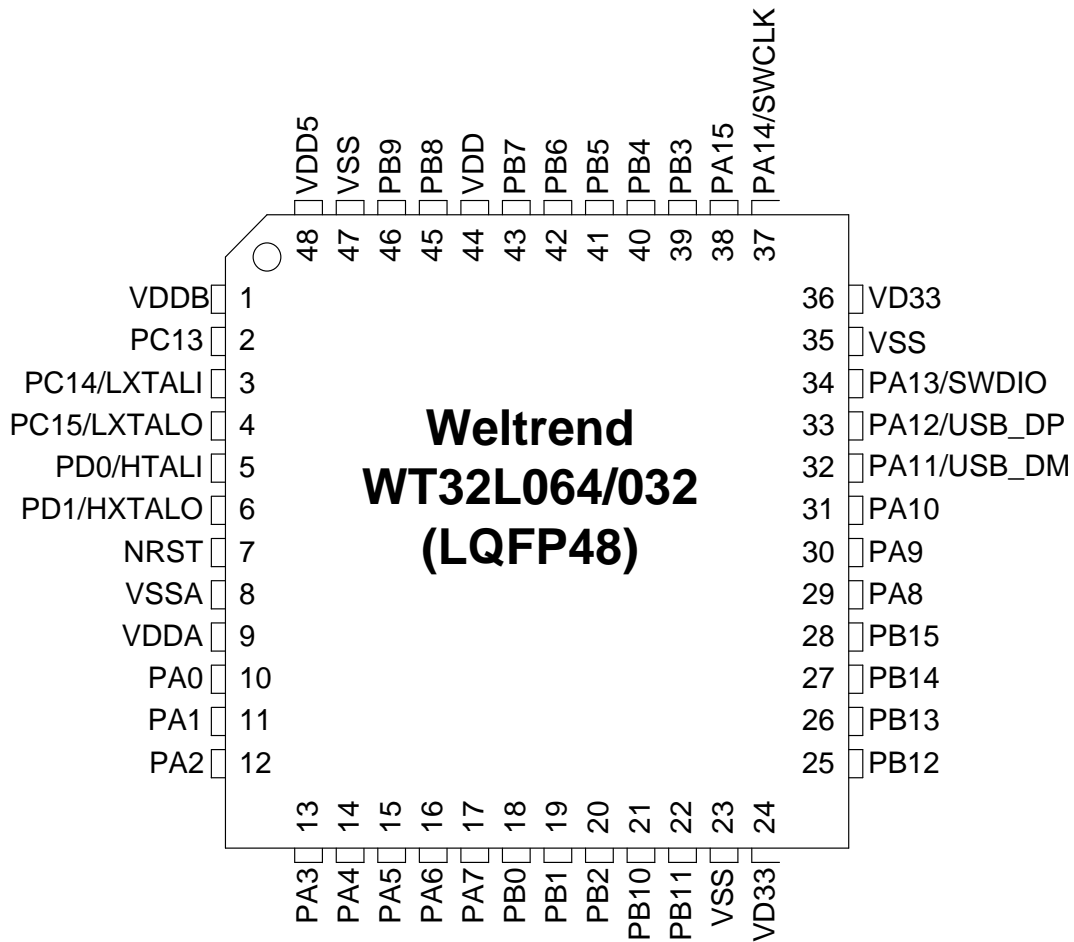
## 2 封装引脚规划

### 2.1 包装

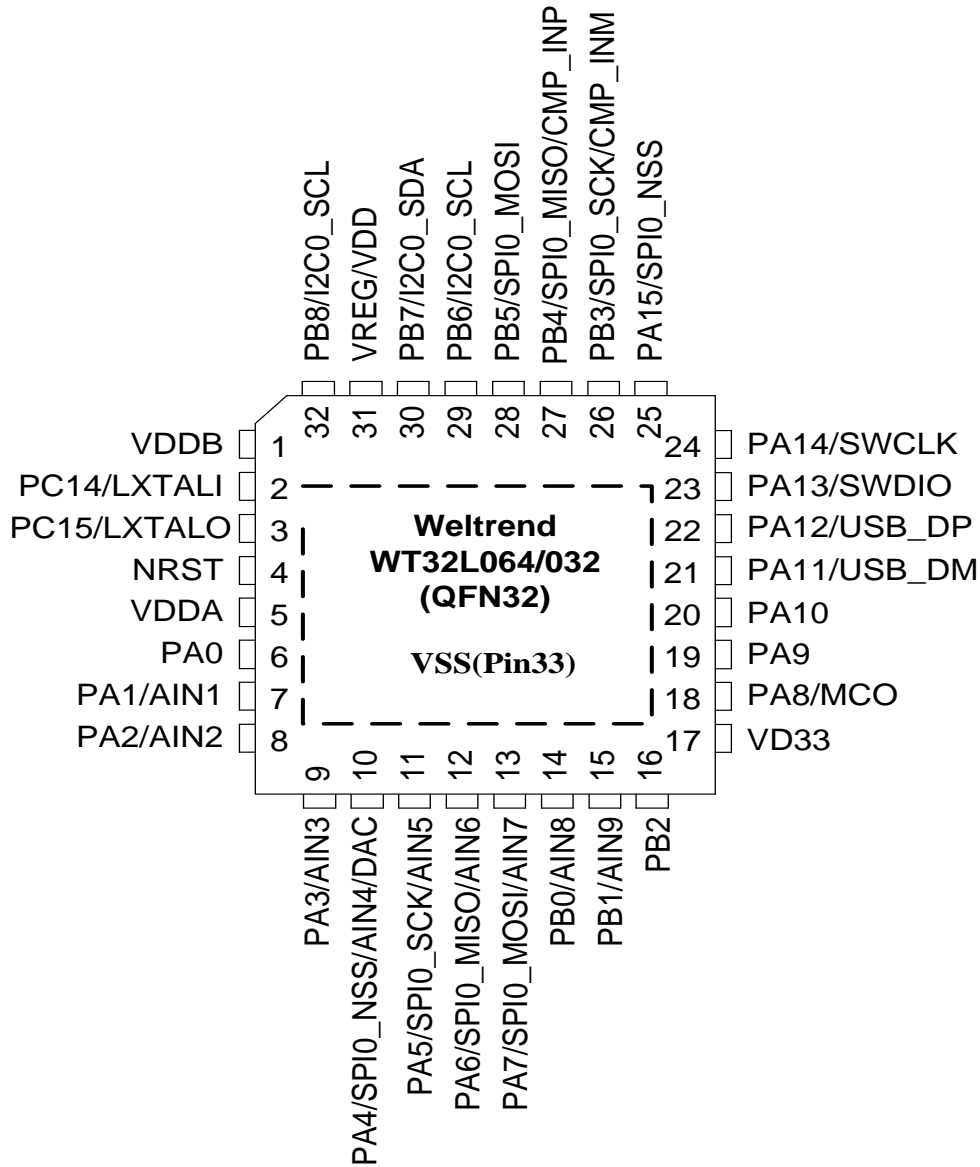
#### 2.1.1 LQFP-64



2.1.2 LQFP-48



2.1.3 QFN-32



**2.2 引脚描述**

表格 2 引脚计数表

Function	LQFP-64	LQFP-48	QFN-32
XTAL	2+2	2+2	1+1
ADC	16	10	10
DAC	1	1	1
comparator	2+2	2+2	2+2
USB	1+1	1+1	1+1
analog power	1+1	1+1	1+0
digital power	3+3	3+3	1+0
VBUS power	1+1	1+1	1+1
LDO_CORE	1	1	1
GPIO	51	37	27
serial wire	2	2	2
MCO (X1)	1 (with remap)	1 (with remap)	1 (with remap)
timer CH (X3)	12 (with remap)	12 (with remap)	12
timer ETR (X1)	1 (with remap)	1 (with remap)	1 (with remap)
UART (X2)	2+2 (with remap)	2+2 (with remap)	2+2 (with remap)
SPI (X2)	2+2+2+2 (with remap)	2+2+2+2 (with remap)	2+2+2+2 (with remap)
I2C (X2)	2+2+2 (with remap)	2+2+2 (with remap)	1+1+1
I2S (X1)	1+1+1+1 (with remap)	-	-
PWM (X4)	4 (with remap)	4 (with remap)	4 (with remap)
WKUP	2	2	1
PVD_IN	1	1	1



表格 3 引脚分配表

LQFP 64	LQFP 48	QFN 32	Name	Type	Structure Type	Description
1	1	1	VDDDB	P	—	battery power pin (RTC)
2	2		PC13	I/O	—	WKUP1
3	3	2	PC14	I/O	A	LXTALI
4	4	3	PC15	I/O	A	LXTALO
5	5		PD0	I/O	B	HXTALI
6	6		PD1	I/O	B	HXTALO
7	7	4	NRST	I	C	active low external reset
8			PC0	I/O	A1	AIN10 / I2S_DIN
9			PC1	I/O	A1	AIN11 / I2S_DOUT
10			PC2	I/O	A1	AIN12 / I2S_BCLK
11			PC3	I/O	A1	AIN13 / I2S_LRCLK / PWM01A
12	8	33	VSSA	P	—	analog ground pin
13	9	5	VDDA	P	—	analog power pin
14	10	6	PA0	I/O	A1	AIN0 / TMR1_CH0_ETR / WKUP0 / PWM02A/COMP0_INM
15	11	7	PA1	I/O	A1	AIN1 / TMR1_CH1 / PWM03A/COMP0_INP
16	12	8	PA2	I/O	A1	AIN2 / UART1_TX / TMR1_CH2
17	13	9	PA3	I/O	A1	AIN3 / UART1_RX / TMR1_CH3
18		33	VSS	P	—	ground pin
19			VD33	P	—	power pin
20	14	10	PA4	I/O	B	AIN4 / SPI0_NSS / DAC
21	15	11	PA5	I/O	A1	AIN5 / SPI0_SCK
22	16	12	PA6	I/O	A1	AIN6 / SPI0_MISO / TMR2_CH0
23	17	13	PA7	I/O	A1	AIN7 / SPI0_MOSI / TMR2_CH1
24			PC4	I/O	A1	AIN14
25			PC5	I/O	A1	AIN15
26	18	14	PB0	I/O	B	AIN8 / TMR2_CH2/EXT_REF
27	19	15	PB1	I/O	A1	AIN9 / TMR2_CH3
28	20	16	PB2	I/O	A	MCO / PWM00A
29	21		PB10	I/O	A	I2C1_SCL / TMR1_CH2
30	22		PB11	I/O	A	I2C1_SDA / TMR1_CH3
31	23	33	VSS	P	—	ground pin
32	24	17	VD33	P	—	power pin
33	25		PB12	I/O	A	SPI1_NSS / I2C1_SMBA
34	26		PB13	I/O	A	SPI1_SCK
35	27		PB14	I/O	A	SPI1_MISO
36	28		PB15	I/O	A	SPI1_MOSI
37			PC6	I/O	A	TMR2_CH0 / I2S_DIN

LQFP 64	LQFP 48	QFN 32	Name	Type	Structure Type	Description
38			PC7	I/O	A	TMR2_CH1 / I2S_DOUT
39			PC8	I/O	A	TMR2_CH2 / I2S_BCLK
40			PC9	I/O	A	TMR2_CH3 / I2S_LRCLK
41	29	18	PA8	I/O	A	MCO / TMR0_CH0 / PWM01B
42	30	19	PA9	I/O	A	UART0_TX / TMR0_CH1 / PWM02B
43	31	20	PA10	I/O	A	UART0_RX / TMR0_CH2 / PWM03B
44	32	21	PA11	I/O	B	USB_DM / TMR0_CH3
45	33	22	PA12	I/O	B	USB_DP
46	34	23	PA13	I/O	A	SWDIO
47	35	33	VSS	P	—	ground pin
48	36		VD33	P	—	power pin
49	37	24	PA14	I/O	A	SWCLK
50	38	25	PA15	I/O	A	TMR1_CH0_ETR / SPI0_NSS
51			PC10	I/O	A	TMR0_CH0 / I2C1_SCL
52			PC11	I/O	A	TMR0_CH1 / I2C1_SDA
53			PC12	I/O	A	TMR0_CH2
54			PD2	I/O	A	TMR0_CH3
55	39	26	PB3	I/O	A1	TMR1_CH1 / SPI0_SCK / COMP1_INM
56	40	27	PB4	I/O	A1	SPI0_MISO / COMP1_INP
57	41	28	PB5	I/O	A	I2C0_SMBA / SPI0_MOSI
58	42	29	PB6	I/O	A	I2C0_SCL / UART0_TX / PWM00B
59	43	30	PB7	I/O	A1	I2C0_SDA / UART0_RX / PVD_IN
60	44	31	VDD	P	—	regulator core power output, connect to bypass capacitor.
61	45	32	PB8	I/O	A	I2C0_SCL / UART1_TX
62	46		PB9	I/O	A	I2C0_SDA / UART1_RX
63	47	33	VSS	P	—	ground pin
64	48		VDD5	P	—	VBUS power pin

GPIO Structure Type:

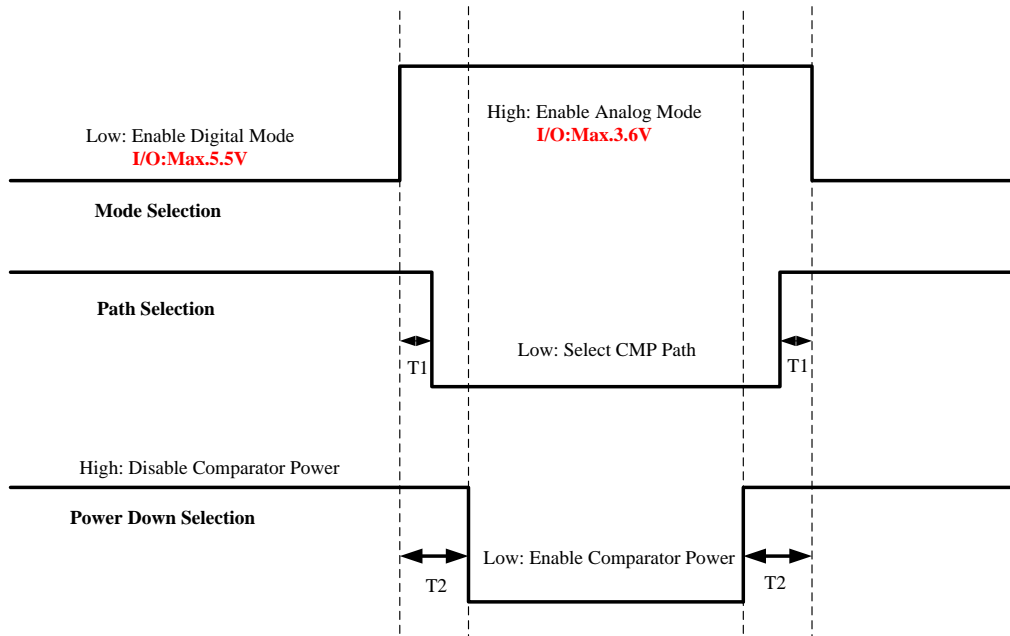
**A:** 5V Tolerant I/O

**B:** STD 3.3V I/O

**C:** 双向复位用的嵌入式弱上拉电阻

**A1:** 5V 容忍 (Tolerant) I/O: 仅在数字模式下, 不包括模拟模式

- ✓ 大部份的 **A1** Type IO 在复位期间默认值是在模拟模式(3.3V), 必须避免连接 5V 应用, 若有需要, 选择 **A** Type
- ✓ 如果在 **A1** Type 中使用, 控制时序必须遵循下面的比较器范例。  
(避免数字模式 5V 输入导致 3.3V 模拟模式的硬件造成损坏)



**2.3 复合功能 I/O 与优先权**

Name	AF_SEL 0	AF_SEL 1	AF_SEL 2	AF_SEL 3	AF_SEL 4	AF_SEL 5	Analog
PA0					TMR1_CH0_ETR	PWM02A	WKUP0 / AIN0/ COM0_INM
PA1					TMR1_CH1	PWM03A	AIN1/COM0_INP
PA2			UART1_TX		TMR1_CH2		AIN2
PA3			UART1_RX		TMR1_CH3		AIN3
PA4				SPI0_NSS			AIN4 / DAC
PA5				SPI0_SCK			AIN5
PA6				SPI0_MISO	TMR2_CH0		AIN6
PA7				SPI0_MOSI	TMR2_CH1		AIN7
PA8		MCO			TMR0_CH0	PWM01B	
PA9			UART0_TX		TMR0_CH1	PWM02B	
PA10			UART0_RX		TMR0_CH2	PWM03B	
PA11					TMR0_CH3		USB_DM
PA12							USB_DP
PA13	SWDIO						
PA14	SWCLK						
PA15				(SPI0_NSS)	(TMR1_CH0_ETR)		
PB0					TMR2_CH2		AIN8/EXT_REF
PB1					TMR2_CH3		AIN9
PB2		MCO				PWM00A	
PB3				(SPI0_SCK)	(TMR1_CH1)		COMP1_INM
PB4				(SPI0_MISO)			COMP1_INP
PB5		I2C0_SMBA		(SPI0_MOSI)			
PB6		I2C0_SCL	(UART0_TX)			PWM00B	
PB7		I2C0_SDA	(UART0_RX)				PVD_IN
PB8		(I2C0_SCL)	(UART1_TX)				
PB9		(I2C0_SDA)	(UART1_RX)				
PB10		I2C1_SCL			(TMR1_CH2)		
PB11		I2C1_SDA			(TMR1_CH3)		
PB12		I2C1_SMBA		SPI1_NSS			
PB13				SPI1_SCK			
PB14				SPI1_MISO			
PB15				SPI1_MOSI			
PC0		I2S_DIN					AIN10
PC1		I2S_DOUT					AIN11
PC2		I2S_BCLK					AIN12
PC3		I2S_LRCLK				PWM01A	AIN13
PC4							AIN14
PC5							AIN15
PC6		(I2S_DIN)			(TMR2_CH0)		
PC7		(I2S_DOUT)			(TMR2_CH1)		
PC8		(I2S_BCLK)			(TMR2_CH2)		
PC9		(I2S_LRCLK)			(TMR2_CH3)		
PC10		(I2C1_SCL)			(TMR0_CH0)		
PC11		(I2C1_SDA)			(TMR0_CH1)		
PC12					(TMR0_CH2)		
PC13							WKUP1
PC14							LXTALI
PC15							LXTALO
PD0							HXTALI
PD1							HXTALO
PD2					(TMR0_CH3)		
PD3							
PD4							

注: ()表示该功能输入的优先级较低

### 3 内存映像

#### 3.1 AMBA 总线地址映像

Index	Function	Description
0x0000_0000~0x1FEF_FFFF	保留	
0x1000_0000~0x1000_FFFF	embedded flash	64KB
0x1FF0_0000~0x1FF0_07FF	保留	
0x1FF0_0800~0x1FF0_FFFF	保留	
0x1FF1_0000~0x1FF1_1FFF	boot ROM	8KB
0x1FF1_2000~0x1FFF_FFFF	保留	
0x2000_0000~0x2000_1FFF	SRAM	8KB
0x2000_5000~0x2FFF_FFFF	保留	
0x3000_0000~0x3000_0FFF	DMA buffer	4KB
0x3000_1000~0x3FFF_FFFF	保留	
0x4000_0000~0x4003_FFFF	APB0 memory space	256KB
0x4004_0000~0x4007_FFFF	APB1 memory space	256KB
0x4008_0000~0x400B_FFFF	AHB memory space	256KB
0x400C_0000~0xDFFF_FFFF	保留	
0xE000_0000~0xE00F_FFFF	保留	
0xE010_0000~0xFFFF_FFFF	保留	
0xF000_0000~0xF000_003FF	System ROM Table	
0xF000_0400~0xFFFF_FFFF	保留	

#### 3.2 APB 内存空间

Range	Index	Function	Description
APB0	0x4000_0000~0x4000_0FFF		
	0x4000_1000~0x4000_1FFF		
	0x4000_2000~0x4000_2FFF	-	
	0x4000_3000~0x4000_3FFF	-	
	0x4000_4000~0x4000_4FFF	UART1	
	0x4000_5000~0x4000_5FFF	-	
	0x4000_6000~0x4000_6FFF	SPI1	
	0x4000_7000~0x4000_7FFF	-	
	0x4000_8000~0x4000_8FFF	I2C0	
	0x4000_9000~0x4000_9FFF	I2C1	
	0x4000_A000~0x4000_AFFF	-	
	0x4000_B000~0x4000_BFFF	-	
	0x4000_C000~0x4000_CFFF		
	0x4000_D000~0x4000_DFFF	-	
	0x4000_E000~0x4000_EFFF	USB	
	0x4000_F000~0x4000_FFFF	-	
	0x4001_0000~0x4001_0FFF	DAC	
	0x4001_1000~0x4001_1FFF	-	
	0x4001_2000~0x4001_2FFF		
	0x4001_3000~0x4001_3FFF	-	
0x4001_4000~0x4001_4FFF	TMR2		

Range	Index	Function	Description
	0x4001_5000~0x4001_5FFF	-	
	0x4001_6000~0x4001_6FFF	-	
	0x4001_7000~0x4001_7FFF	-	
	0x4001_8000~0x4001_8FFF	IWDT	
	0x4001_9000~0x4001_9FFF	WWDT	
	0x4001_A000~0x4001_AFFF	PMU	power management unit
	0x4001_B000~0x4001_BFFF	RTC	
	0x4001_C000~0x4001_CFFF	PWM	
	0x4001_D000~0x4001_DFFF	-	
	0x4001_E000~0x4001_EFFF	-	
	0x4001_F000~0x4001_FFFF	-	
APB1	0x4004_0000~0x4004_0FFF	-	
	0x4004_1000~0x4004_1FFF	-	
	0x4004_2000~0x4004_2FFF	-	
	0x4004_3000~0x4004_3FFF	-	
	0x4004_4000~0x4004_4FFF	UART0	
	0x4004_6000~0x4004_6FFF	SPI0	
	0x4004_7000~0x4004_7FFF	-	
	0x4004_8000~0x4004_8FFF	-	
	0x4004_9000~0x4004_9FFF	-	
	0x4004_A000~0x4004_AFFF	-	
	0x4004_B000~0x4004_BFFF	-	
	0x4004_C000~0x4004_CFFF	I2S	
	0x4004_D000~0x4004_DFFF	-	
	0x4004_E000~0x4004_EFFF	-	
	0x4004_F000~0x4004_FFFF	-	
	0x4005_0000~0x4005_0FFF	ADC	
	0x4005_1000~0x4005_1FFF	-	
	0x4005_2000~0x4005_2FFF	-	
	0x4005_3000~0x4005_3FFF	-	
	0x4005_4000~0x4005_4FFF	TMR0	
	0x4005_5000~0x4005_5FFF	TMR1	
	0x4005_6000~0x4005_6FFF	-	
	0x4005_7000~0x4005_7FFF	-	
	0x4005_9000~0x4005_9FFF	CRS	
	0x4005_A000~0x4005_AFFF	-	
	0x4005_B000~0x4005_BFFF	-	
	0x4005_C000~0x4005_CFFF	-	
	0x4005_D000~0x4005_DFFF	-	
	0x4005_E000~0x4005_EFFF	-	
	0x4005_F000~0x4005_FFFF	-	
AHB	0x4008_0000~0x4008_0FFF	FLASH	flash control
	0x4008_1000~0x4008_1FFF	-	
	0x4008_2000~0x4008_2FFF	DMA	
	0x4008_3000~0x4008_3FFF	-	
	0x4008_4000~0x4008_4FFF	RCC	
	0x4008_5000~0x4008_5FFF	-	

Range	Index	Function	Description
	0x4008_6000~0x4008_6FFF	-	
	0x4008_7000~0x4008_7FFF	-	
	0x4008_9000~0x4008_9FFF	-	
	0x4008_A000~0x4008_AFFF	CRC32	
	0x4008_B000~0x4008_BFFF	-	
	0x4008_C000~0x4008_CFFF	GPIO	
	0x4008_D000~0x4008_DFFF	-	
	0x4008_E000~0x4008_EFFF	-	
	0x4008_F000~0x4008_FFFF	-	
	0x5001_F000~0x5001_FFFF	system control	

## 4 复位和时钟控制

### 4.1 主要概述

复位和时钟控制模块(RCC)如下 2 张方块图所示，用于提供系统各单元的复位和时钟。

- 复位输出
  - 提供 Flash 复位
  - 提供全局复位
- 时钟输出:
  - 提供 AHB, APB0 & APB1 时钟
  - 提供 ADC 时钟
  - 提供 USB 时钟
  - 微控制器的时钟输出 (MCO)

### 4.2 方块图

Reset Diagram

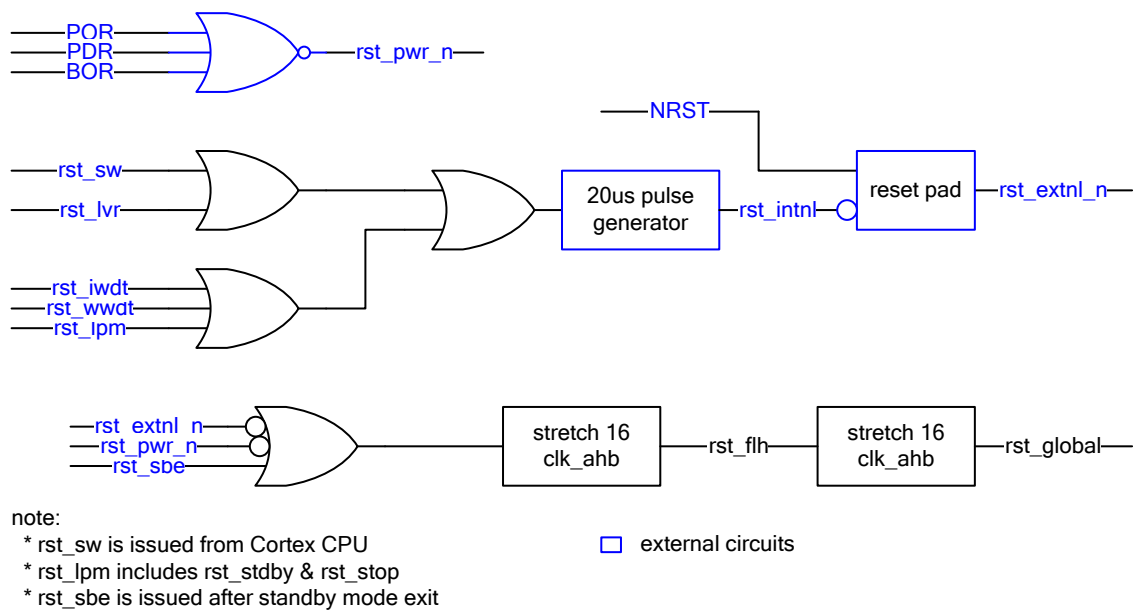


图 8 RCC 模块复位图



**Clock Diagram**

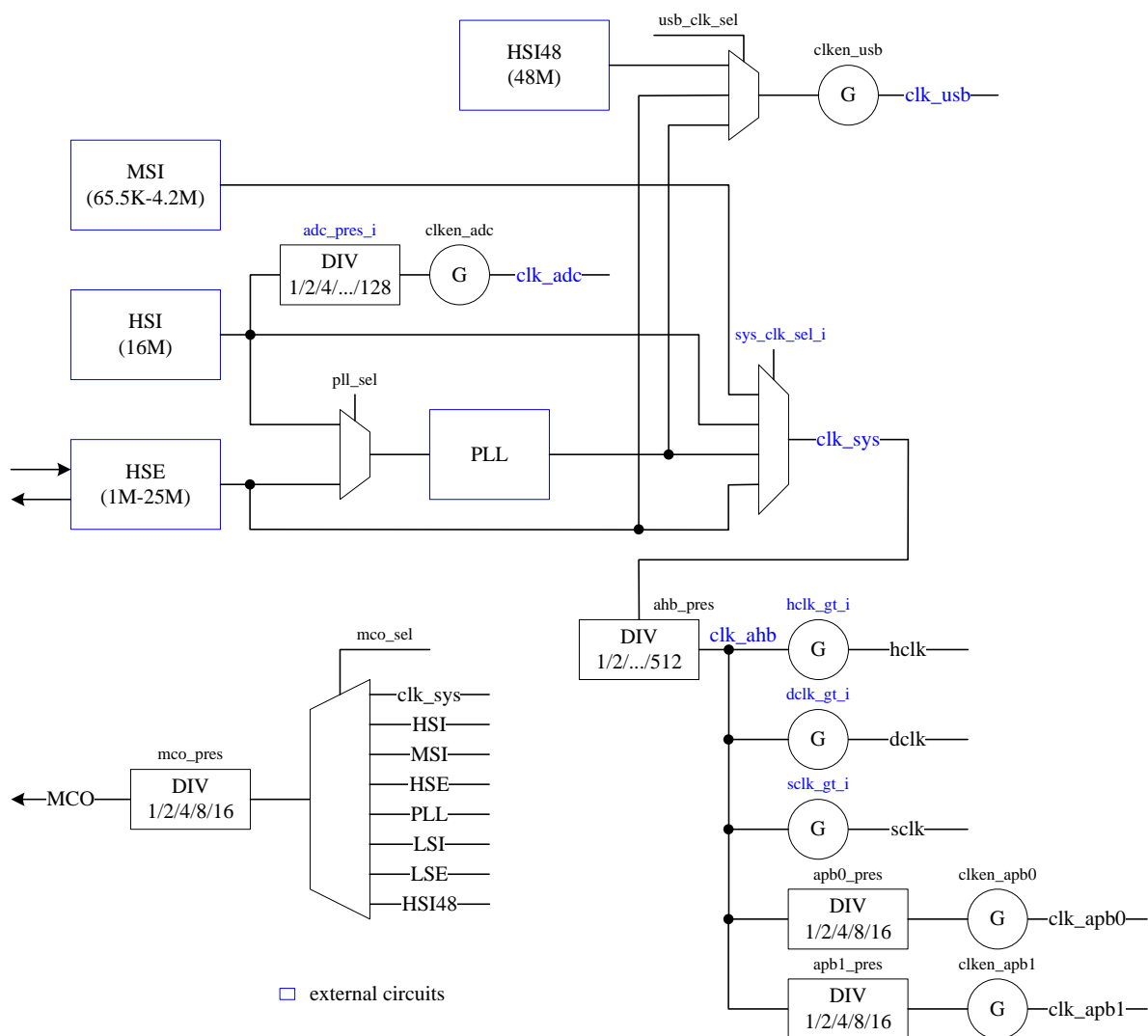


图 9 RCC 模块时钟方块图

**4.3 RCC 缓存器描述**

表格 4 RCC 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>RCC_CCR: Clock control settings</b>					
00h	28:24	R/W	03h	pll_prediv	输入参考时钟 预分频器 $F_{int} = F_{in} / (pll\_prediv + 1)$
	17:16	R/W	0h	pll_postdiv	输出时钟分频器 $F_{out} = F_{vco} / (2^{**} (pll\_postdiv + 1))$
	14:8	R/W	0fh	pll_mul	feedback clock multiplier $F_{fb} = F_{vco} / (pll\_mul + 1)$
	0	R/W	0	pll_sel	PLL 时钟源选择: 0: 选择 HSI 振荡器时钟作为 PLL 输入时钟 1: HSE 振荡器时钟选择为 PLL 输入时钟

<b>RCC_CSCR: Source control settings</b>					
04h	9:8	R/W	0	usb_clk_sel	USB 时钟选择: 0: HSI48 1: PLL 2: HSE
	1	R/W	1	clken_apb1	APB1 时钟启动
	0	R/W	1	clken_apb0	APB0 时钟启动

<b>RCC_CCFGR: Source control settings</b>					
08h	30:28	R/W	0	mco_pres	MCO 预分频: $MCO\_PAD = MCO / 2^{**} mco\_pres$ valid range: 0 – 4
	27:24	R/W	0	mco_sel	microcontroller 时钟输出选择: 0000: MCO 输出关闭, MCO 无时钟 0001: SYSCLK 时钟 0010: HSI oscillator 时钟 0011: MSI oscillator 时钟 0100: HSE oscillator 时钟 0101: PLL 时钟 0110: LSI oscillator 时钟 0111: LSE oscillator 时钟 1000: HSI48 oscillator 时钟
	13:11	R/W	0	apb0_pres	控制 APB0 低速的时钟除数 0: HCLK not divided 1: HCLK divided by 2 2: HCLK divided by 4 3: HCLK divided by 8 4: HCLK divided by 16
	10:8	R/W	0	apb1_pres	控制 APB1 时钟除数 时钟如下: 0: HCLK not divided 1: HCLK divided by 2 2: HCLK divided by 4

RCC_CCFGR: Source control settings					
					3: HCLK divided by 8 4: HCLK divided by 16
	7:4	R/W	0	ahb_pres	控制 AHB 时钟除数 时钟如下: 0: SYSCLK not divided 1: SYSCLK divided by 2 2: SYSCLK divided by 4 3: SYSCLK divided by 8 4: SYSCLK divided by 16 5: SYSCLK divided by 32 6: SYSCLK divided by 64 7: SYSCLK divided by 128 8: SYSCLK divided by 256 9: SYSCLK divided by 512

RCC_RAPB0R: Module Resets for APB0					
10h	28	R/W	0	rst_pwm	PWM 复位: 0: 无效 1: PWM 复位
	27	R/W	0	rst_rtc	RTC 复位, 当全局复位仍持续执行
	25	R/W	0	rst_wwdt	WWDT 复位
	24	R/W	0	rst_iwdt	IWDT 复位
	20	R/W	0	rst_tmr2	timer-2 复位
	16	R/W	0	rst_dac	DAC 复位
	14	R/W	0	rst_usb	USB 复位
	9	R/W	0	rst_i2c1	I2C1 复位
	8	R/W	0	rst_i2c0	I2C0 复位
	6	R/W	0	rst_spi1	SPI1 复位
	4	R/W	0	rst_uart1	UART1 复位
	1	-	-	-	保留
	0	-	-	-	保留

RCC_RAPB1R: Module Resets for APB1					
14h	24	-	-	-	保留
	21	R/W	0	rst_tmr1	timer-1 复位
	20	R/W	0	rst_tmr0	timer-0 复位
	16	R/W	0	rst_adc	ADC 复位
	12	R/W	0	rst_i2s	I2S 复位
	6	R/W	0	rst_spi0	SPI0 复位
	5	-	-	-	保留
	4	R/W	0	rst_uart0	UART0 复位

RCC_RAHBR: Module Resets for AHB					
18h	12	R/W	0	rst_gpio	GPIO 复位
	10	R/W	0	rst_crc32	CRC32 复位
	8	-	-	-	保留
	2	R/W	0	rst_dma	DMA 复位

RCC_CEAPB0R: Module clock enables for APB0					
20h	28	R/W	0	clken_pwm	PWM 时钟启动: 0: PWM 时钟禁能 1: PWM 时钟启动
	27	R/W	0	clken_rtc	RTC 时钟启动
	26	R/W	1	clken_pmu	PMU 时钟启动
	25	R/W	0	clken_wwdt	WWDT 时钟启动
	24	R/W	0	clken_iwdt	IWDT 时钟启动
	20	R/W	0	clken_tmr2	timer-2 时钟启动
	16	R/W	0	clken_dac	DAC 时钟启动
	14	R/W	0	clken_usb	USB 时钟启动
	12	R/W	0	clken_intc	interrupt controller 时钟启动
	9	R/W	0	clken_i2c1	I2C1 时钟启动
	8	R/W	0	clken_i2c0	I2C0 时钟启动
	6	R/W	0	clken_spi1	SPI1 时钟启动
	4	R/W	0	clken_uart1	UART1 时钟启动
	1	-	-	-	保留
0	-	-	-	保留	

RCC_CEAPB1R: Module clock enables for APB1					
24h	25	R/W	0	Clken_crs	CRS 时钟启动 0: CRS 时钟禁能 1: CRS 时钟启动
	24	-	-	-	保留
	21	R/W	0	clken_tmr1	timer-1 时钟启动
	20	R/W	0	clken_tmr0	timer-0 时钟启动
	16	R/W	0	clken_adc	ADC 时钟启动
	12	R/W	0	clken_i2s	I2S 时钟启动
	6	R/W	0	clken_spi0	SPI0 时钟启动
	5	-	-	-	保留
	4	R/W	0	clken_uart0	UART0 时钟启动

RCC_CEAHBR: Module clock enables for AHB					
28h	15	R/W	0	clken_sys	SYS 时钟启动: 0: SYS 时钟禁能 1: SYS 时钟启动
	12	R/W	0	clken_gpio	GPIO 时钟启动
	10	R/W	0	clken_crc32	CRC32 时钟启动
	8	-	-	-	保留

RCC_CEAHBR: Module clock enables for AHB					
	2	R/W	0	clken_dma	DMA 时钟启动
	0	R/W	1	clken_flh_prog	flash programmer 时钟启动

RCC_LCEAPB0R: Low power module clock enables for APB0					
30h	28	R/W	0	lpclken_pwm	PWM 低功耗模式时时钟启动: 0: PWM 低功耗模式时时钟禁能 1: PWM 低功耗模式时时钟启动
	27	R/W	0	lpclken_rtc	RTC 低功耗模式时时钟启动 0: RTC 低功耗模式时时钟禁能 1: RTC 低功耗模式时时钟启动
	26	R/W	1	lpclken_pmu	PMU 低功耗模式时时钟启动
	25	R/W	0	lpclken_wwdt	WWDT 低功耗模式时时钟启动
	24	R/W	0	lpclken_iwdt	IWDT 低功耗模式时时钟启动
	20	R/W	0	lpclken_tmr2	timer-2 低功耗模式时钟启动
	16	R/W	0	lpclken_dac	DAC 低功耗模式时时钟启动
	14	R/W	0	lpclken_usb	USB 低功耗模式时时钟启动
	12	R/W	0	lpclken_intc	interrupt controller 低功耗模式时时钟启动
	9	R/W	0	lpclken_i2c1	I2C1 低功耗模式时时钟启动
	8	R/W	0	lpclken_i2c0	I2C0 低功耗模式时时钟启动
	6	R/W	0	lpclken_spi1	SPI1 低功耗模式时时钟启动
	4	R/W	0	lpclken_uart1	UART1 低功耗模式时时钟启动
	1	-	-	-	保留
0	-	-	-	保留	

RCC_LCEAPB1R: Low power module clock enables for APB1					
34h	25	R/W	0	lpclken_crs	CRS 低功耗模式时时钟启动: 0: CRS 低功耗模式时时钟禁能 1: CRS 低功耗模式时时钟启动
	24	-	-	-	保留
	21	R/W	0	lpclken_tmr1	Timer-1 低功耗模式时时钟启动
	20	R/W	0	lpclken_tmr0	Timer-0 低功耗模式时时钟启动
	16	R/W	0	lpclken_adc	ADC 低功耗模式时时钟启动
	12	R/W	0	lpclken_i2s	I2S 低功耗模式时时钟启动
	6	R/W	0	lpclken_spi0	SPI0 低功耗模式时时钟启动
	5	-	-	-	保留
	4	R/W	0	lpclken_uart0	UART0 低功耗模式时时钟启动

RCC_LCEAHBR: Low power module clock enables for AHB					
38h	15	R/W	0	lpclken_sys	SYS 低功耗模式时时钟启动: 0: SYS 低功耗模式时时钟禁能 1: SYS 低功耗模式时时钟启动
	12	R/W	0	lpclken_gpio	GPIO 低功耗模式时时钟启动
	10	R/W	0	lpclken_crc32	CRC32 低功耗模式时时钟启动
	8	-	-	-	保留
	2	R/W	0	lpclken_dma	DMA 低功耗模式时时钟启动

RCC_LCEAHBR: Low power module clock enables for AHB					
	0	R/W	1	lpclken_flh_prog	flash programmer 低功耗模式时钟启动

RCC_CSR: Reset flag					
40h	31	R/W1c	0	rstf_lpm	低功率管理复位旗标
	30	R/W1c	0	rstf_wwdt	WWDT 复位旗标
	29	R/W1c	0	rstf_iwdt	IWDT 复位旗标
	28	R/W1c	0	rstf_lockup	lockup 复位旗标
	27	R/W1c	1	rstf_pwr	PWR (POR/PDR/BOR) 复位旗标
	26	R/W1c	0	rstf_lvr	LVR 复位旗标
	25	R/W1c	0	rstf_sbe	standby-exit 复位旗标
	24	R/W1c	0	rstf_sys	系统复位旗标 (软件复位旗标)
	23	R/W1c	0	rstf_ext	external 复位旗标

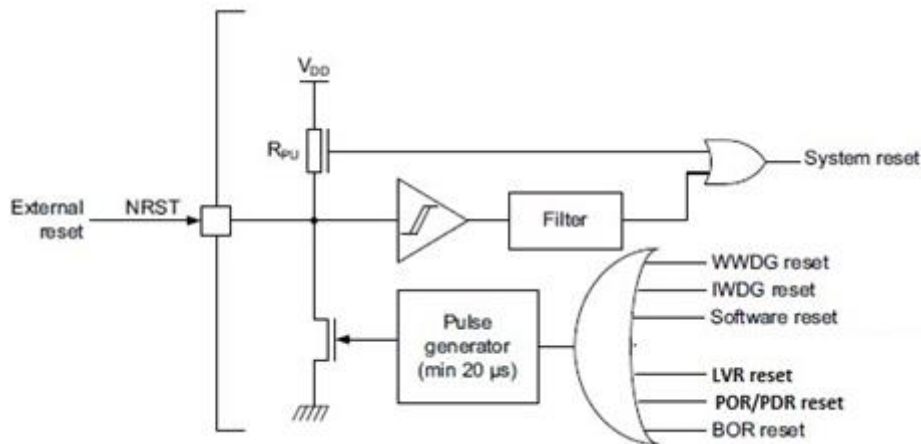
R/W1C: 读取 与 写“1”清除

注意: 使用 RTC 功能前必须先启动 RTC clock, 因为没有开启就无 APB0 时钟会引起系统当机。

## 4.4 功能描述

### 4.4.1 复位 (RESET)

RCC 模块中包括三种类型的复位，定义为系统复位、电源复位和退出待机的复位。



#### 4.4.1.1 系统复位

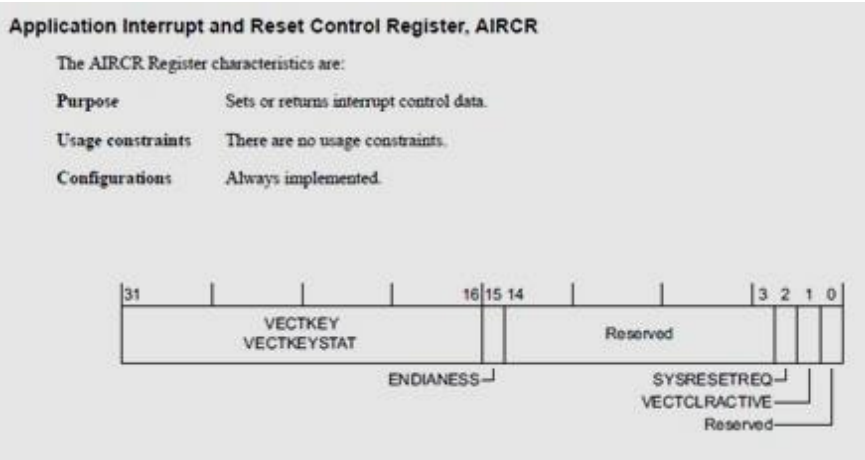
当发生以下事件之一时，将产生系统复位：

1. RST\_N 引脚上有低电平产生 (外部复位)
2. 窗口看门狗计数终止的情况 (WWDG 复位)
3. 独立看门狗计数终止的情况 (IWDG 复位)
4. 软件复位 (SW 复位)
5. LVR 复位

这些来源作用于 NRST 引脚，该引脚在复位过程始终保持在低电位。RESET 服务例程向量固定在内存映像的地址 0x0000\_0004。提供给组件的系统复位信号输出在 NRST 引脚上。脉冲发生器保证每个内部复位源的最小复位脉冲持续时间为 20μs。在外部复位的情况下，当 NRST 引脚应用为低时，将产生复位脉冲。

#### 软件复位 (Software Reset)

必须将 Cortex®-M0 应用程序中断和复位控制寄存器中的 SYSRESETREQ 位设置为可强制设备上的软件复位。有关详细信息，请参阅 Cortex®-M0 技术参考手册。



Bits	Type	Name	Function
[2]	WO	SYSRESETREQ	<p>System Reset Request:</p> <p>0 do not request a reset.</p> <p>1 request reset.</p> <p>Writing 1 to this bit asserts a signal to request a reset by the external system. The system components that are reset by this request are IMPLEMENTATION DEFINED. A Local reset is required as part of a system reset request.</p> <p>A Local reset clears this bit to 0.</p>

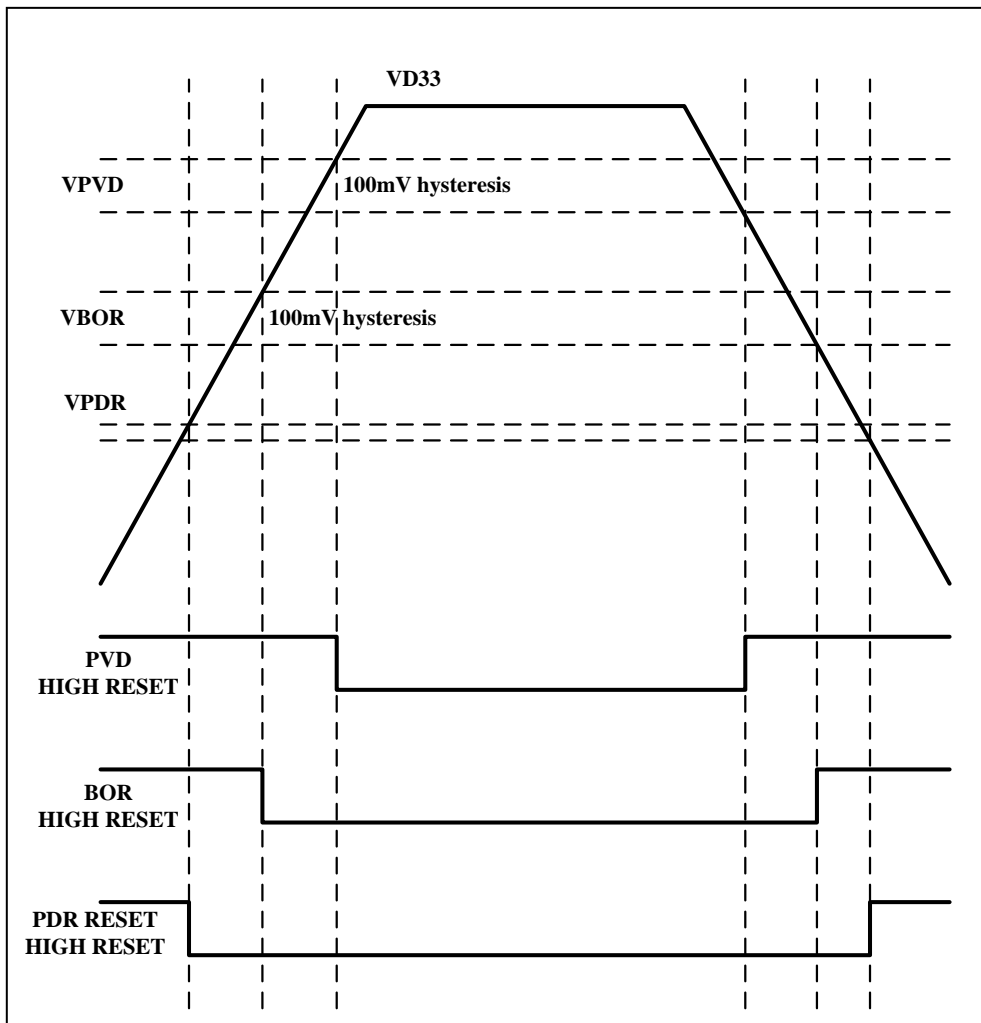


**4.4.1.2 电源复位**

当发生以下事件之一时，将产生电源复位:

1. 电源打开/断电复位 (POR/PDR 复位)
2. BOR 复位

电源复位将所有缓存器设置为其复位值。



1. PVD 由软件激活或关闭。
2. BOR 可运行从 1.8V 到 3.6V。
3. PDR 复位级别约为 1.0V。
4. 当 **VD33** 从 1.65V 工作到 3.6V 时没有 BOR;当 **VD33** 低于 PDR 电平时，将启动复位。
5. BOR 复位电平设置: 请参阅第 16.6 PMU 章节。
6. PVD 设置: 请参阅第 16.6 PMU 章节。

#### 4.4.1.3 待机离开时复位 (Standby Exit Reset)

当系统从待机模式恢复时，将产生退出待机(Standby Exit)复位。

#### 4.4.2 时钟 (Clock)

四个不同的时钟源振荡器可用于驱动系统的时钟 (SYSCLK):

- HSI 内部高速振荡器 (High-speed internal, oscillator clock)
- HSE 外部高速振荡器 (High-speed external, oscillator clock)
- PLL 锁相回路振荡器 (PLL clock)
- MSI 内部多种速度振荡器 (Multispeed internal, oscillator clock)

从复位启动、从停止唤醒或待机(Standby)低功耗模式唤醒后，使用 MSI 作为系统时钟源。

此时 IC 仍具有以下两个辅助时钟源:

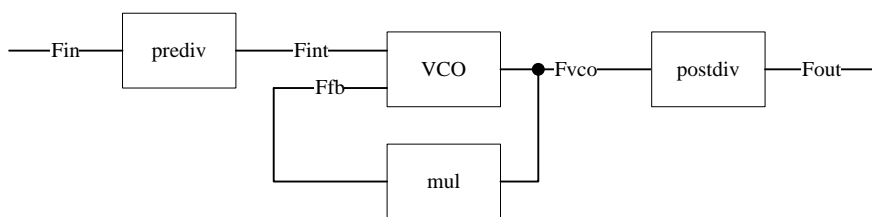
- 37 kHz 低速内部 RC (LSI RC)，驱动独立看门狗。
- 32.768 kHz 低速外部晶体 (LSE Xtal)，可驱动系统时钟 (RTCCLK)

##### 4.4.2.1 PLL 时钟 (PLL Clock)

内部 PLL 可以由 HSI RC 或 HSE 晶体当时钟源。它用于驱动系统的时钟，并为 USB 外围产生 48 MHz 时钟。PLL 输入时钟频率必须介于 2 MHz 和 24 MHz 之间。通过使用嵌入在 PLL 中的输入除法、乘法因子和输出除法获得所需的频率。

在启动 PLL 之前，必须执行 PLL 规划 (选择源时钟、输出除法、乘法因子和输出除法因子)。启动 PLL 后，无法更改这些参数。

**PLL Diagram**



**图 10 PLL Diagram**

**注意:**

1.  $8\text{MHz} \leq F_{vco} \leq 192\text{MHz}$ , if  $V_{DDA} \geq 1.8\text{V}$ ;  $8\text{MHz} \leq F_{vco} \leq 96\text{MHz}$ , if  $1.65\text{V} \leq V_{DDA} < 1.8\text{V}$ ;
2.  $F_{out} = F_{in} * (\text{pll\_mul} + 1) / (\text{pll\_prediv} + 1) / (2 ** (\text{pll\_postdiv} + 1))$
3.  $F_{vco} = F_{in} * (\text{pll\_mul} + 1) / (\text{pll\_prediv} + 1)$ .

## 5 电源管理单元

### 5.1 电源管理单元概述

电源管理单元 (PMU) 控制核心和外围组件的所有电源, 还控制 MSI 的系统时钟和频率范围的来源。该器件集成了零功率上电复位 (zeropower power-on reset, POR)、断电复位 (power-down reset, PDR), 并配有欠电压复位 (brown out reset, BOR) 电路。(请参阅 BOR) 该器件具有嵌入式可程序设定电压检测器 (PVD), 可监控  $V_{DD}/V_{DDA}$  电源并将其与  $V_{PVD}$  门坎值进行比较。软件可在 1.85V 和 3.05 V 之间选择 七种不同的 PVD 级别, 步进级距为 200 mV。(请参阅 PVD)

- 三种不同的稳压器模式:
  - Main (MR)
  - Low-power (LPR)
  - Power-down
- 动态核心电压:
  - 1.8V
  - 1.2V
- 五种省电模式:
  - Low-power run mode (LPR)
  - Sleep mode (MR)
  - Stop mode (MR)
  - Low-power stop mode (LPR)
  - Standby mode (Power-down)

### 5.2 Power Domain Overview

该器件可分为两个模块, 由两个电压提供。核心电压可在待机(Standby)模式下关闭, 如图 11 所示。

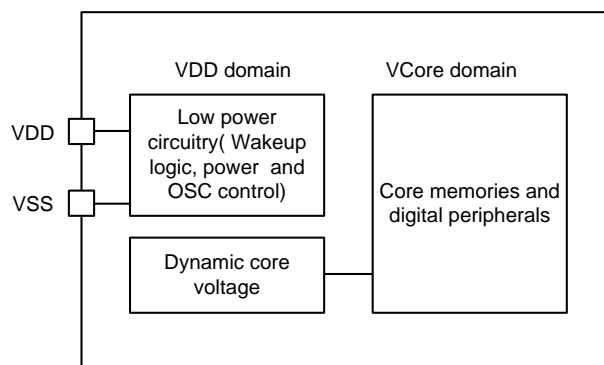


图 11 电源域概观

### 5.3 电压稳压器

嵌入式线性稳压器提供除待机(Standby)电路外的所有数字电路。稳压器输出电压 ( $V_{CORE}$ ) 可以通过软件程序设定到 1.2V 或 1.8 V 内的二个不同范围 (参见动态核心电压章节)。

复位后一直启动电压稳压器(regulator)。它适用于三种不同的模式: 主控(MR)、低功耗(LPR) 和断电模式, 具体取决于应用模式。

**Run mode**, 稳压器(Regulator)是主控模式(MR), 为  $V_{CORE}$  域 (内核、存储器和数字外围设备)提供全功率。

**Low-power run mode**, 稳压器处于低功耗模式 (LPR), 为  $V_{CORE}$  域提供低功耗, 从而保留缓存器和内部 SRAM 的内容。

**Sleep mode**, 稳压器是主控模式(MR), 为  $V_{CORE}$  域提供全功率, 保留缓存器和内部 SRAM 的内容。关闭系统时钟, 但为 NVIC 保留外围时钟。

**Stop mode**, 稳压器是主控模式(MR), 为  $V_{CORE}$  域提供全功率, 保留缓存器和内部 SRAM 的内容。所有内部时钟都关闭电源, 只能通过扩展中断 (EXTI) 唤醒。(请参阅 EXTI 部分)

**Low-power stop mode**, 稳压器处于低功耗模式 (LPR), 向  $V_{CORE}$  域提供低功耗, 从而保留缓存器和内部 SRAM 的内容。所有内部时钟都关闭电源, 只能通过扩展中断 (EXTI) 唤醒。(请参阅 EXTI 部分)。

**Standby mode**, 稳压器已关闭电源。缓存器和 SRAM 的内容丢失, 待机电路除外。

### 5.4 动态核心电压

动态电压缩放是一种电源管理技术, 根据具体情况增加或降低用于数字外围 ( $V_{CORE}$ ) 的电压。

#### Range 1

范围 1 是着重“CPU 性能(performance)”电压范围。只要 VDD 输入电压高于 2.3V。闪存程序, 稳压器输出 1.8 V 电压, 即可在此范围内执行擦除操作。

#### Range 2

范围 2 是着重“功率性能”电压范围。稳压器输出 1.2 V 电压, VDD 没有任何限制, 但系统时钟不能超过 4.2MHz。

有关每个系列的性能的信息, 请参阅表格 5。

表格 5 性能与  $V_{CORE}$  范围

CPU performance	Power performance	Vcore Range	Typical Value (V)	Max frequency (MHz)	VDD range
High	Low	1	1.8	System Clock <sub>(MAX)</sub>	2.3V-VDD <sub>MAX</sub>
Low	High	2	1.2	4.2	VDD <sub>MIN</sub> -VDD <sub>MAX</sub>

→ 当 VD33 电压低于 2.2V, 系统会自动将  $V_{CORE}$  power 由 1.8V 切换到 1.2V, 因此用户必须使用 PVD (可程序电压侦测器) 监测 VD33, 一旦低于 2.2V 之前就靠软件将时钟源切换到较低频的 MSI (4.2 MHz)。

→ 当 VD33 以高速时钟工作并交替快速开启/关闭时, 用户必须开启 BOR 以防止  $VD33 \leq 2.2V$  引起 MCU 故障。

## 5.5 省电模式

默认情况下，微控制器在系统或上电复位后处于运行模式。在运行模式下，CPU 由 HCLK 计时，程序代码执行。当 CPU 不需要保持运行时（例如在等待外来事件时）可以使用几种低功耗模式来节省电源。用户需要选择在低功耗、性能、启动时间短和可用唤醒源之间提供最佳折衷模式。

设备具有五种省电模式：

- **Low-power run mode:** 稳压器在低功耗模式下，时钟频率有限，运行的外围设备数量有限。
- **Sleep mode:** Cortex<sup>®</sup>-M0 内核停止，外围设备继续运行。
- **Stop mode:** 所有时钟都停止，稳压器运行。
- **Low-power stop mode:** 所有时钟都停止，稳压器处于低功耗模式。
- **Standby mode:** V<sub>CORE</sub> 域已关闭电源。

此外，运行模式下的功耗可以通过以下方法之一降低：

- 减慢系统时钟的速度
- 当未使用到 APBx 和 AHBx 外围设备时，可以关闭时钟。

表格 6 节能模式摘要

Mode name	Entry	Wakeup	Effect on V <sub>CORE</sub> domain clocks	Effect on V <sub>DD</sub> Domain clocks	Voltage regulator
<b>Low-power run</b>	bldo_pd and LPRUN bits + Clock setting	稳压器(regulator)强制使用主稳压器 (1.8V)	None	None	低功耗模式下 (low-power mode)
<b>Sleep mode</b>	WFI or Return from ISR	Any interrupt	关闭 CPU CLK, 对其它时钟或模拟时钟源没有影响	None	ON
	WFE	Wakeup event			
<b>Stop mode</b>	SLEEPDEEP bit + WFI or Return from ISR or WFE	任何扩展中断 (EXTI) 和 IWDT 复位	所有 V <sub>CORE</sub> 域时钟关闭	HSI16(1), HSE and MSI oscillators OFF	ON
<b>Low-power stop mode</b>	lpsdsr bits + SLEEPDEEP bit + WFI or Return from ISR or WFE				低功耗模式下 (low-power mode)
<b>Standby mode</b>	pdds bits + SLEEPDEEP bit + WFI or Return from ISR or WFE				WKUP 引脚高电平, RTC 报警, RTC 周期定时器, 外部复位输入, NRST 引脚, IWDT 复位

### 5.5.1 外部中断 (Extended Interrupt, EXTI)

EXTI 允许管理超过 20 条事件线路，这些事件线路可以从停机模式唤醒设备。

于停机模式下使用 I/O 单向边缘触发来进行唤醒时，需留意以下两点以避免异常，或可改用双向边缘触发或准位触发：

- 1· 下缘触发：若 I/O 处于低位准时进入停机模式，则此 I/O 无法再产生下降缘而无法唤醒。
- 2· 上缘触发：若 I/O 处于高位准时进入停机模式，则此 I/O 无法再产生上升缘而无法唤醒。

表格 7 EXTI Lines Connections

EXTI line	Line source
每个 port 的 0-15	GPIO
16	PVD
17	RTC alarm
18	RTC periodic timer
19	COMP output

### 5.5.2 节能模式下时钟的行为

APB 外围时钟可以通过软件关闭。

#### 睡眠模式 (Sleep modes)

CPU 时钟在睡眠模式下停止。内存接口时钟 (闪存和 RAM 接口) 和所有外围设备时钟可以在睡眠期间由软件停止。当连接外围的所有时钟时，在睡眠模式下，硬件会关闭 AHB 到 APB 网桥时钟。

#### 停止和待机模式 (Stop and Standby modes)

系统时钟和所有高速时钟在停机和待机模式下停止：

- PLL 功能被关闭
- 内部 RC 16 MHz (HSI16) 振荡器被关闭，除非在停机模式下设置 hsi16keron 位
- 外部 1-24 MHz (HSE) 振荡器被关闭
- 内部 65 kHz – 4.2 MHz (MSI) 振荡器被关闭

当通过中断 (停机模式) 退出此模式时，内部 MSI 或 HSI16 可以通过 PMU 单元控制位 (Stopwuck) 选择为系统时钟。对于两个振荡器，其各自的范围调适值保持在停机模式退出。

当通过复位 (待机模式) 退出此模式时，内部 MSI 振荡器将选择为系统时钟。范围调适值复位为预设的 2.1 MHz。

如果闪存程序操作或对 APB 域的访问正在进行，则延迟停机模式或待机模式的执行，直到闪存或 APB 访问完成。

进入停机模式 (STOP Mode) 前，须关闭所有 TIMER 与 SYSTICK，待唤醒后再重新开启，以避免唤醒异常。

**5.6 缓存器定义**
**表格 8 PMU 控制缓存器**

Index	Bit	R/W	Default	Name	Description
<b>PMU_EN: PMU Enable</b>					
00h	[0]	R/W	1	pmu_enable	0: 关闭 PMU 以节省更多电源 1: 启动 PMU
<b>PMU_PVD_IE: Interrupt signals</b>					
04h	[1]	R/W	1	pvd_int_en	启动 PVD interrupt 0: 禁能 1: 启动
	[0]	R	0	pvd_int	PVD interrupt flag
<b>PMU_RGLTR_CR: Regulator control signals</b>					
08h	[8]	R/W	1	ch1p2v_n	核心电源选择。 0: 1.2V 1: 1.8V
	[5]	R/W	0	lpsdsr	进入睡眠模式或停机模式时关闭 BLDO
	[4]	R/W	0	pdds	当 MCU 申请深度睡眠时, 状态选择 0: 停机模式 ( <b>Stop Mode</b> ) 1: 待机模式 ( <b>Standby Mode</b> )
	[1]	R/W	0	sldo_pd	低功耗稳压器 (regulator) 断电 0: 开启 1: 断电
	[0]	R/W	0	bldo_pd	主要稳压器 (regulator) 断电 0: 开启 1: 断电
<b>PMU_VD_CR: Voltage detector control signals</b>					
0Ch	[10:8]	R/W	000	bor_sel	BOR level selection [000]: setting VBOR0=1.7V [001]: setting VBOR1=1.93V [010]: setting VBOR2=2.3V [011]: setting VBOR3=2.55V [100]: setting VBOR4=2.8V
	[6:4]	R/W	000	pls_sel	PVD 电平选择 [000]: 设置 VPVD0=1.85V [001]: 设置 VPVD1=2.04V [010]: 设置 VPVD2=2.24V [011]: 设置 VPVD3=2.4 V [100]: 设置 VPVD4=2.64V [101]: 设置 VPVD5=2.83V [110]: 设置 VPVD6=3.05V [111]: 设置 VPVD7: 外部输入模拟电压 (内部比较与 VBGC)
	[3:2]	R/W	11	cmp_hs	选择 CMP 的操作模式 High Speed Mode: 1 Low Speed Mode: 0 cmp_hs[1]: CMP2

Index	Bit	R/W	Default	Name	Description
					cmp_hs[0]: CMP1
	[1]	R/W	1	bor_pd	BOR 电源控制 0: 开启 1: 关闭
	[0]	R/W	1	pvd_pd	PVD 电源控制 0: 开启 1: 关闭
<b>PMU_AN_PD: Analog power control</b>					
14h (pmu_an_pd)	[11]	R/W	1	dac_pd	DAC 电源控制 0: 开启 1: 关闭
	[10]	R/W	1	r2r_pd	Rail-to-Rail 电源控制 0: 开启 1: 关闭
	[9]	R/W	1	adc_pd	ADC 电源控制 0: 开启 1: 关闭
	[8:7]	R/W	1	cmp_pd	CMP1 & CMP2 电源控制 [7]: CMP1, [8]: CMP2 0: 开启 1: 关闭
	[6]	R/W	0	flash_keep_off	退出停机模式时, Flash 保持关闭电源
	[5]	R/W	0	run_pd_fls	退出睡眠模式以运行模式时, Flash 关闭
	[4]	R/W	0	sleep_pd_fls	进入睡眠模式时, Flash 关闭
	[1]	R/W	0	rom_pd	ROM 电源控制 0: 开启 1: 关闭
	[0]	R/W	0	flash_pd	Flash 电源控制 0: 开启 1: 关闭
<b>PMU_CLK_PD: Clock power signals</b>					
18h	[5]	R/W	1	usb48m_pd	USB 48M 电源控制 0: 开启 1: 关闭
	[4]	R/W	1	hse_pd	HSE 电源控制 0: 开启 1: 关闭
	[3]	R/W	1	pll_pd	PLL 电源控制 0: 开启 1: 关闭
	[2]	R/W	1	hsi_pd	HSI 电源控制 0: 开启 1: 关闭
	[1]	R/W	1	lsi_pd	LSI 电源控制 0: 开启 1: 关闭
	[0]	R/W	0	msi_pd	MSI 电源控制



Index	Bit	R/W	Default	Name	Description
					0: 开启 1: 关闭
<b>PMU_CLK_CR: Clock control signals</b>					
1Ch	[11]	R/W	0	dclk_gt	1: Disable debug mode DCLK for power saving 0: Enable debug mode DCLK, no power saving
	[9]	R/W	0	stopwuck	退出停机模式 (STOP Mode) 时, 使用时钟源 1: Turn on HSI 0: Turn on MSI
	[8]	R/W	0	hsi16keron	进入停机模式时保持 HSI 电源打开
	[6:4]	R/W	101	msi_range	频率输出选择引脚 3'b000: range0 65.5K 3'b001: range1 131K 3'b010: range2 262K 3'b011: range3 524K 3'b100: range4 1.05M 3'b101: range5 2.1M 3'b110: range6 4.2M 3'b111: range6 4.2M
	[1:0]	R/W	00	sw_sysclk	选择系统频率 00: MSI 01: HSI16 10: HSE 11: PLL
<b>PMU_WKUP_CR: Wakeup control signals</b>					
20h	[3]	R/W	0	rst_stop	进入停机模式时产生复位
	[2]	R/W	0	rst_stdby	进入待机模式时产生复位
	[1]	R/W	0	wukp1_en	Wakeup pin 1 启动
	[0]	R/W	0	wukp0_en	Wakeup pin 0 启动
<b>PMU_CSR: Status Flag</b>					
24h	[4]	R/W	0	boot_sel	Boot select
	[1]	R/W1C	0	stop_flag	停止旗标, 如果从停止唤醒, 写入 1 以清除
	[0]	R/W1C	0	stby_flag	待机旗标, 如果从待机唤醒, 请写入 1 以清除
<b>PMU_ATPD_STOP: Auto power down at Stop</b>					
28h	[8]	R/W	0	stop_lvr22_pd	设定 lvr22_pd 当 PMU 进入停机模式
	[7]	R/W	0	stop_lvr18_pd	设定 lvr18_pd 当 PMU 进入停机模式
	[4]	R/W	1	stop_r2r_pd	设定 r2r_pd 当 PMU 进入停机模式
	[3]	R/W	1	stop_dac_pd	设定 dac_pd 当 PMU 进入停机模式
	[2]	R/W	1	stop_adc_pd	设定 adc_pd 当 PMU 进入停机模式
	[1]	R/W	1	stop_cmp_pd	设定 cmp_pd 当 PMU 进入停机模式
	[0]	R/W	1	stop_lsi_pd	设定 lsi_pd 当 PMU 进入停机模式
<b>PMU_ATPD_STBY: Auto power down at Standby</b>					
2Ch	[11]	R/W	0	stby_pdvbg_dig	设定 pdvbg_dig 当 PMU 进入待机模式
	[10]	R/W	0	stby_bor_pd	设定 bor_pd 当 PMU 进入待机模式
	[9]	R/W	0	stby_pvd_pd	设定 pvd_pd 当 PMU 进入待机模式
	[8]	R/W	0	stby_lvr22_pd	设定 lvr22_pd 当 PMU 进入待机模式
	[7]	R/W	0	stby_lvr18_pd	设定 lvr18_pd 当 PMU 进入待机模式

Index	Bit	R/W	Default	Name	Description
	[6]	R/W	0	stby_pdr_v18_pd	设定 pdr_v18_pd 当 PMU 进入待机模式
	[5]	R/W	0	stby_pdr_vda_pd	设定 pdr_vda_pd 当 PMU 进入待机模式
	[4]	R/W	1	stby_r2r_pd	设定 r2r_pd 当 PMU 进入待机模式
	[3]	R/W	1	stby_dac_pd	设定 dac_pd 当 PMU 进入待机模式
	[2]	R/W	1	stby_adc_pd	设定 adc_pd 当 PMU 进入待机模式
	[1]	R/W	1	stby_cmp_pd	设定 cmp_pd 当 PMU 进入待机模式
	[0]	R/W	1	stby_lsi_pd	设定 lsi_pd 当 PMU 进入待机模式

R/W1C: 读取 与 写“1”清除

## 5.7 状态机 (State Machine)

### 5.7.1 执行模式 (Run mode)

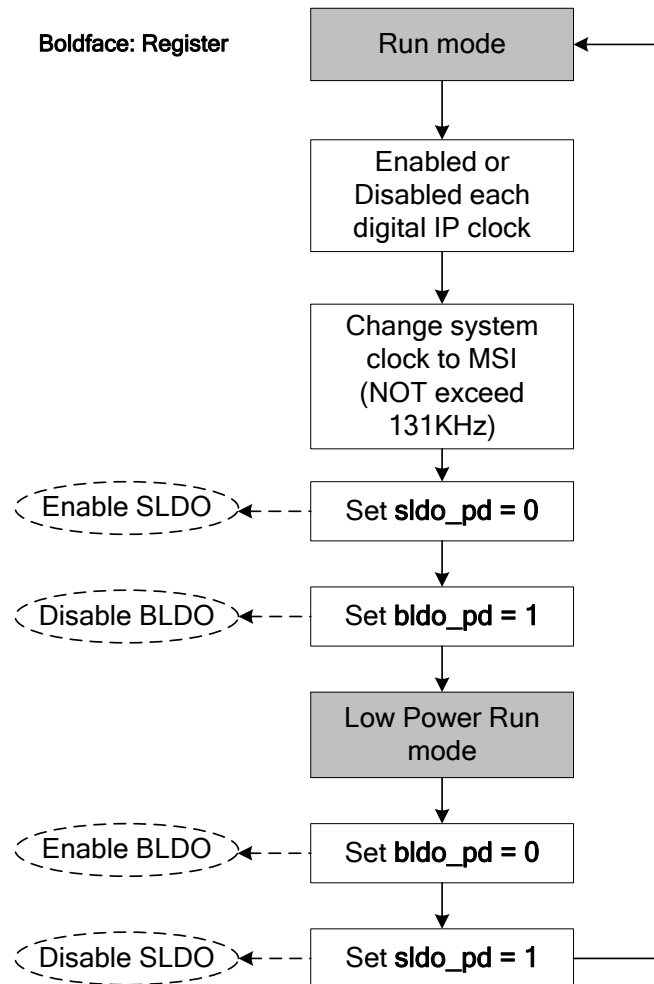


图 12 低功耗运行模式状态机

5.7.2 睡眠模式 (Sleep Mode)

**Boldface: Register**

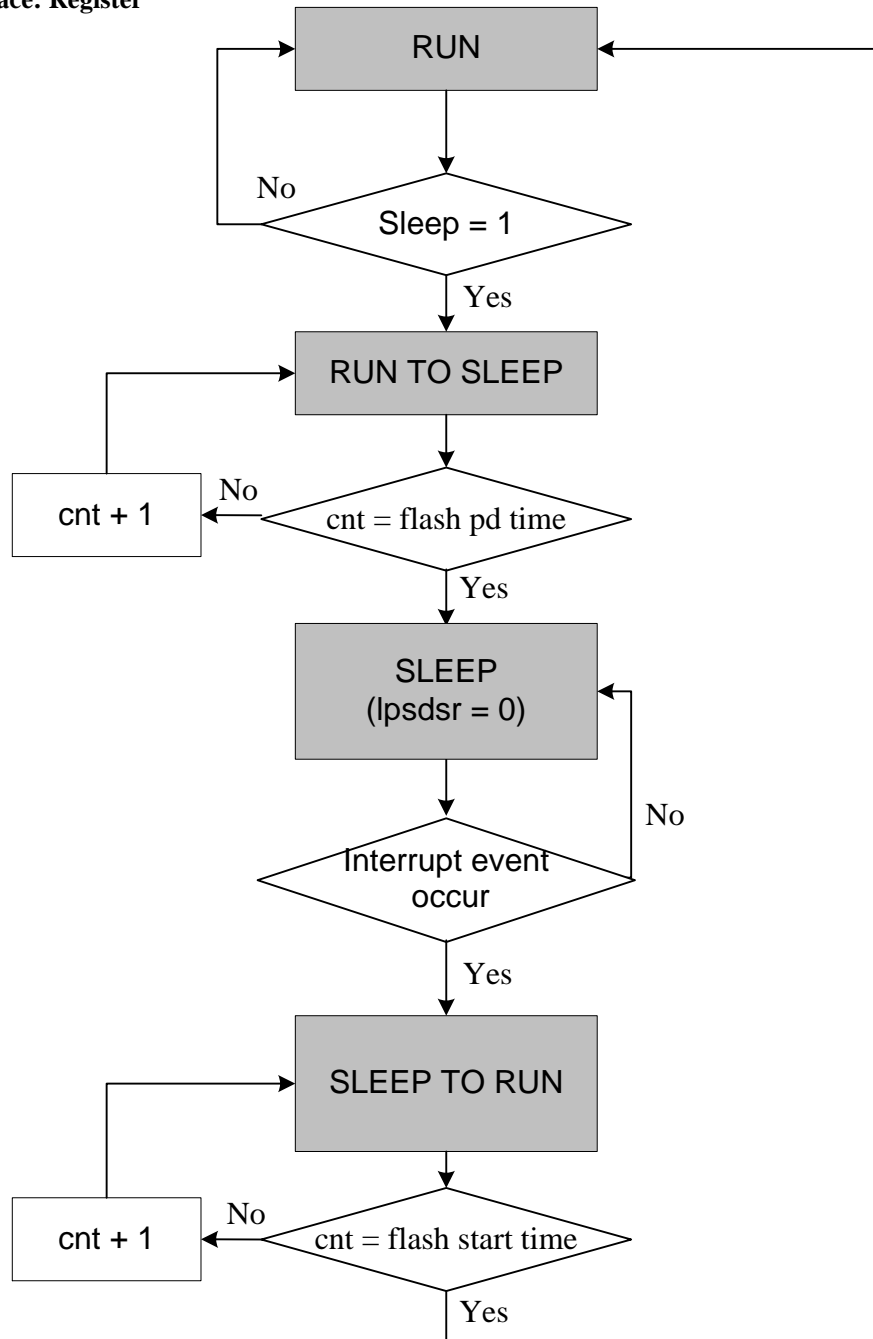


图 13 睡眠模式状态机

### 睡眠模式

CPU 时钟在睡眠模式下停止。内存接口时钟 (闪存和 RAM 接口) 和所有外围设备时钟可以在睡眠期间由软件停止。在睡眠模式下, 当连接到这些时钟的所有外围设备时钟被关闭时, 硬件会关闭 AHB 到 APB 网桥时钟。

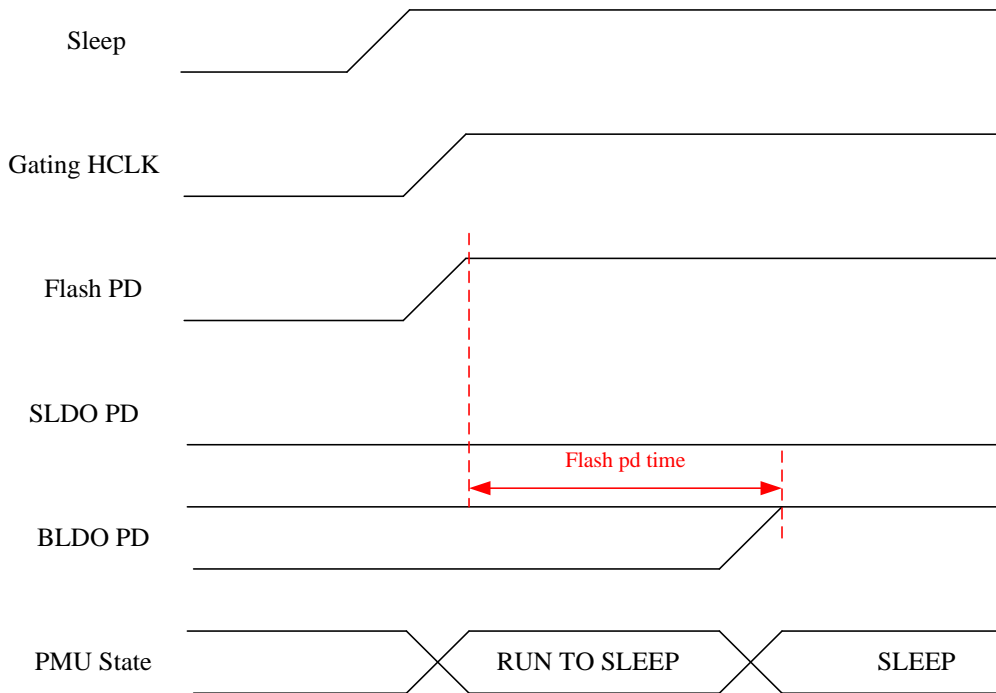


图 14 进入睡眠模式的时序图

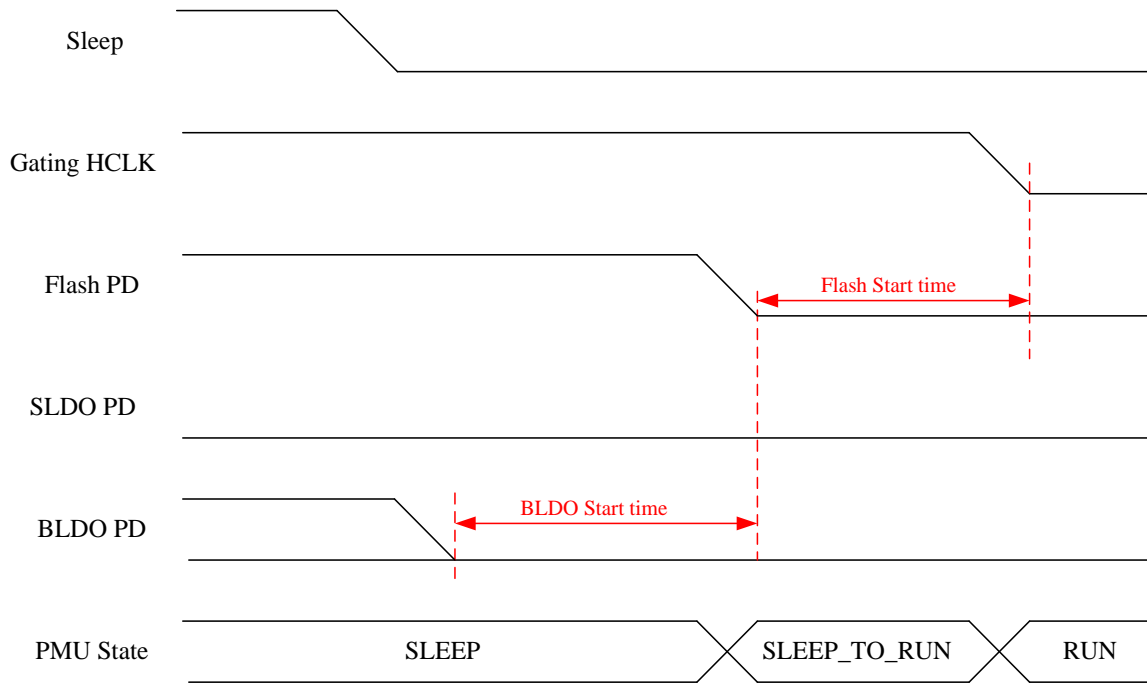


图 15 退出睡眠模式的时序图

5.7.3 停机模式 (Stop Mode)

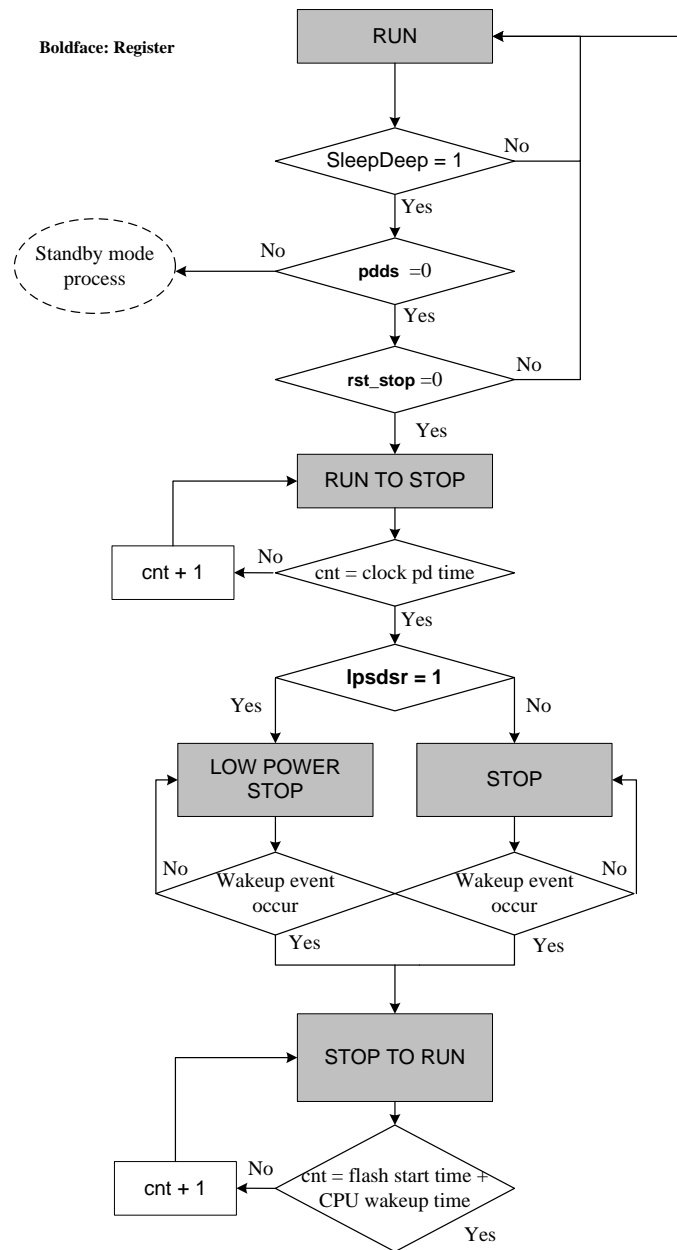


图 16 停机模式状态机

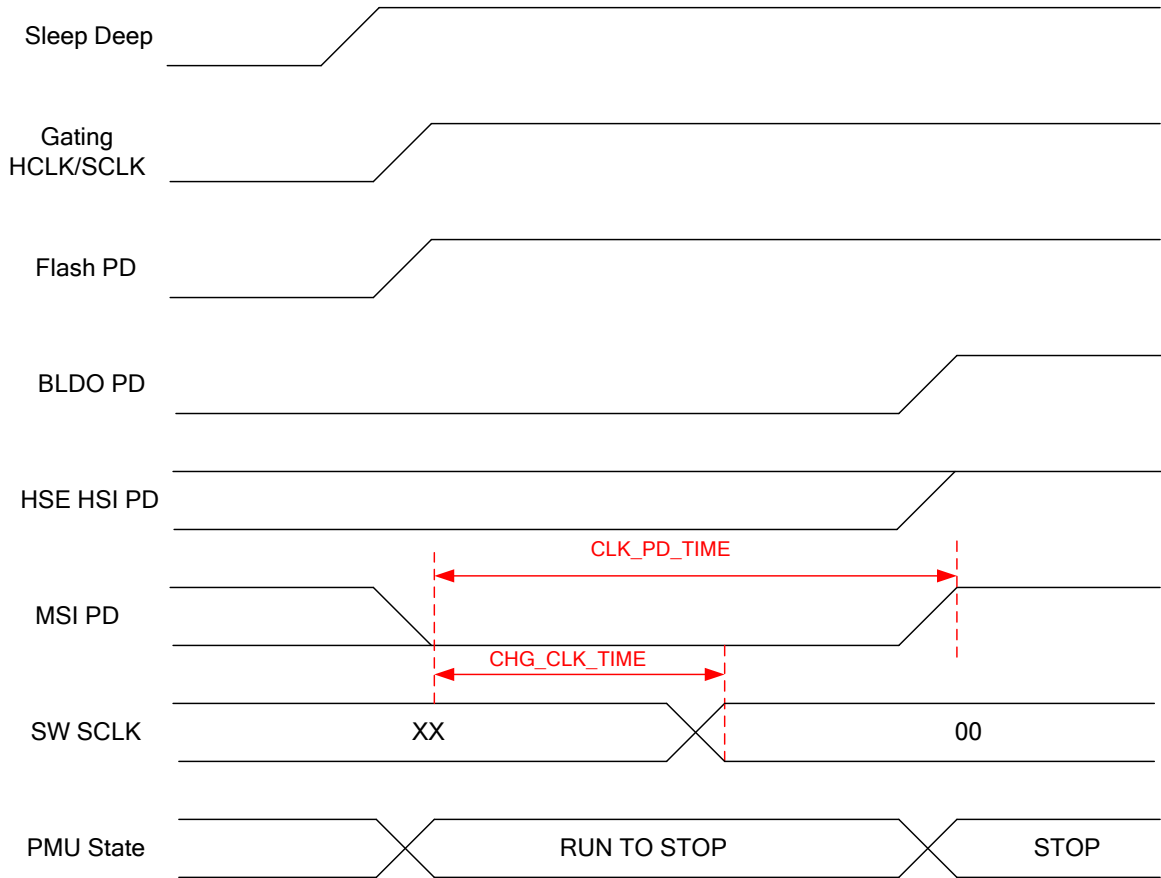


图 17 进入停机模式的时序图



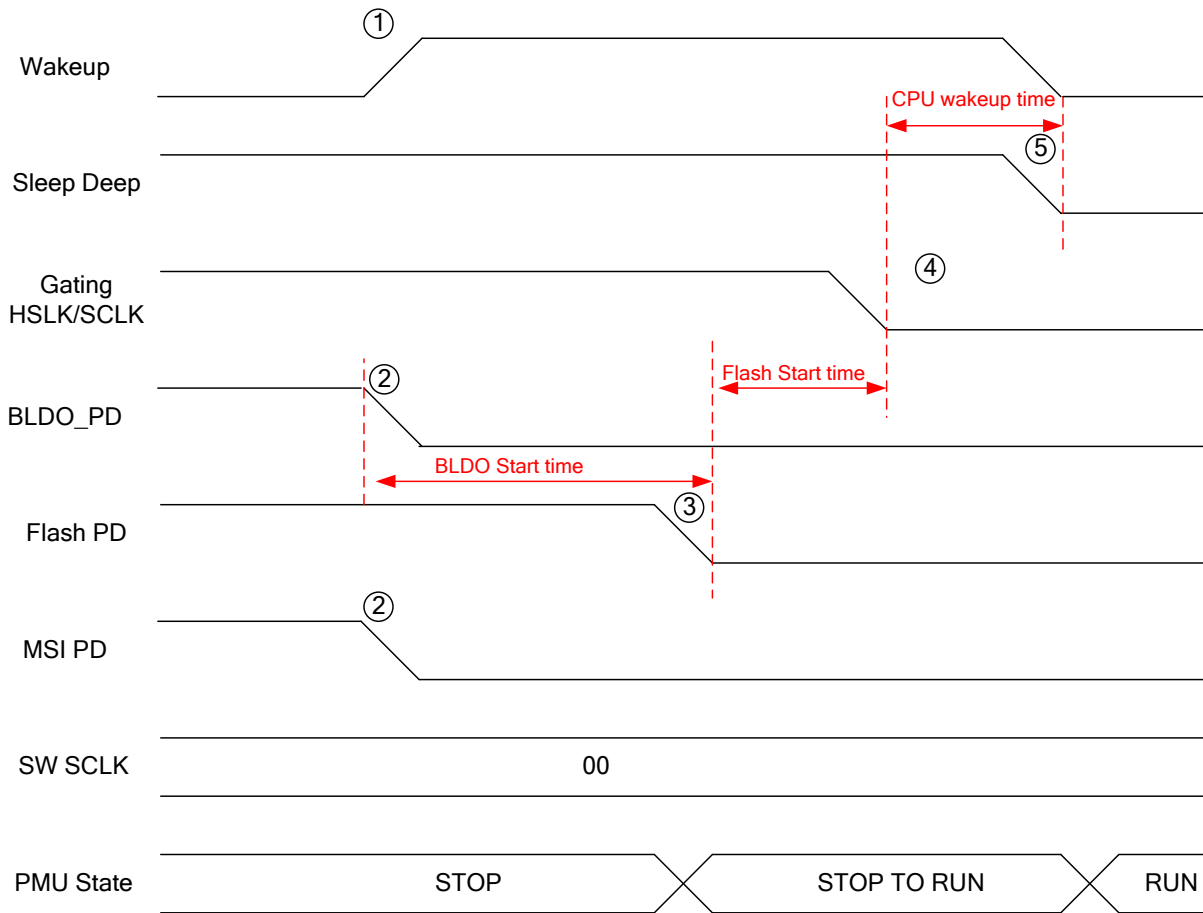


图 18 退出停机模式的时序图

5.7.4 待机模式 (Standby Mode)

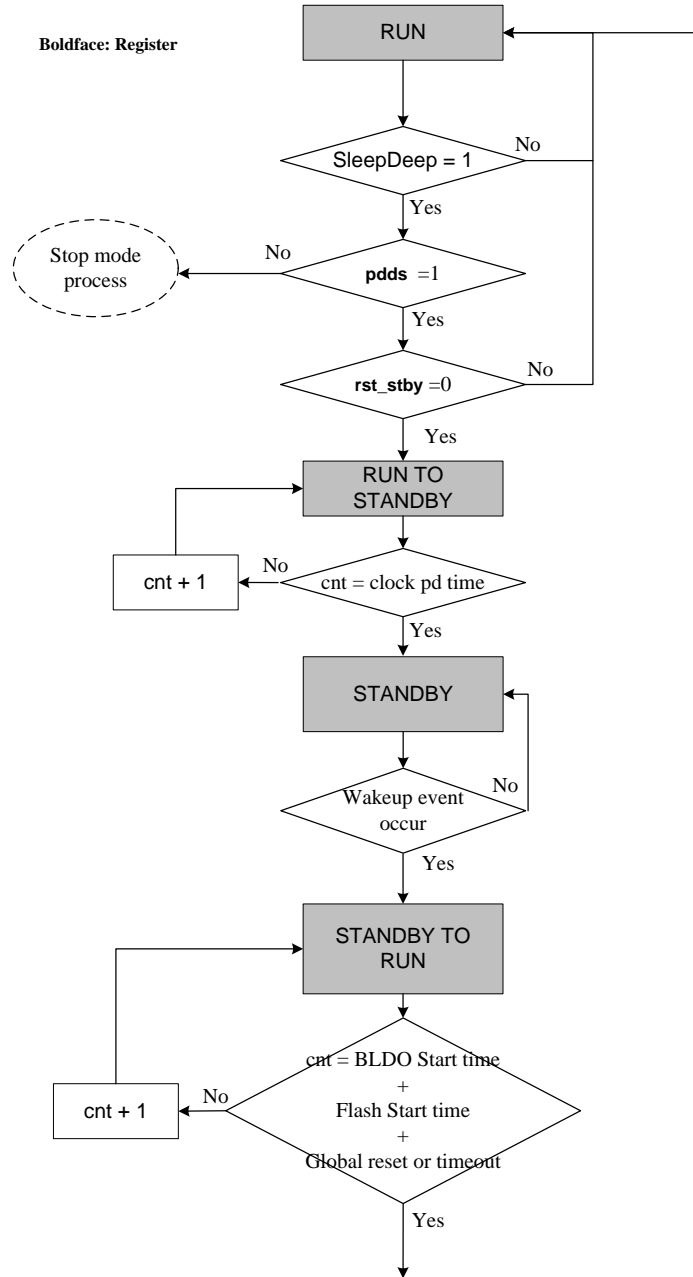


图 19 待机模式状态机

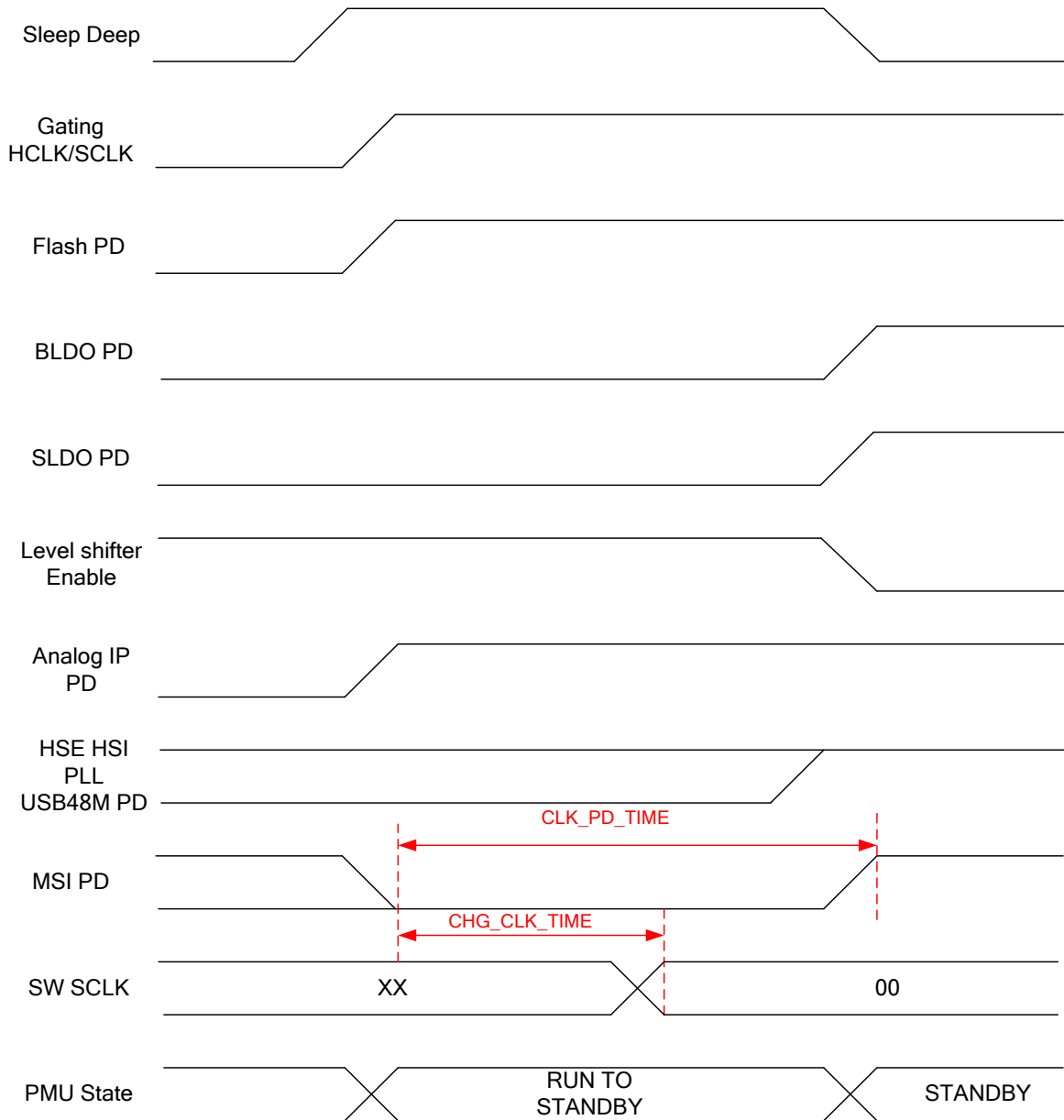


图 20 进入待机模式的时序图

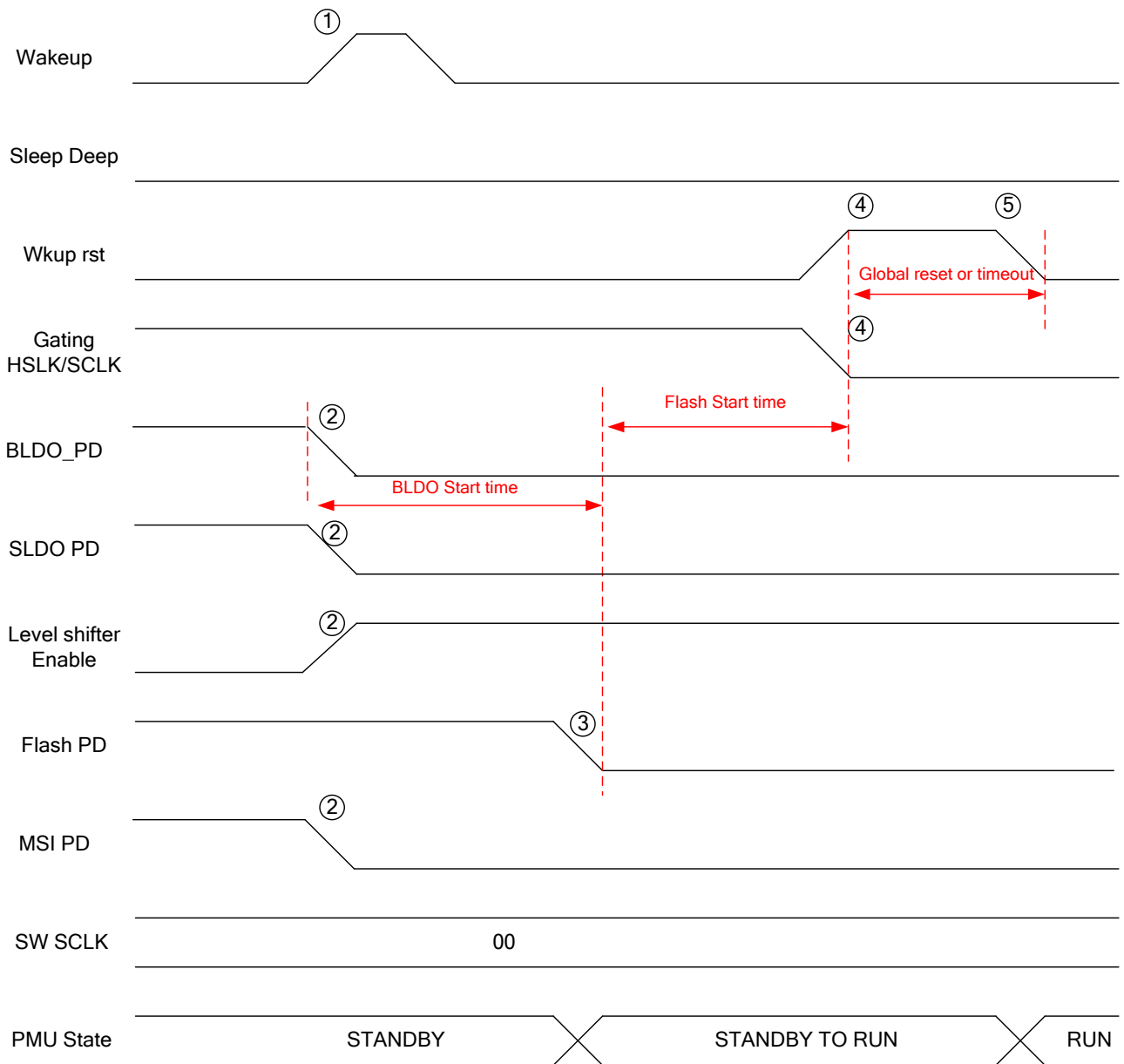


图 21 退出待机模式的时序图

### 5.8 设定范例

将 GPIO PA[0] 设置为唤醒引脚，PA[3:1] 表示 PMU 状态。用户可以在任何电源模式下测量电流（流程图的红色方块）。

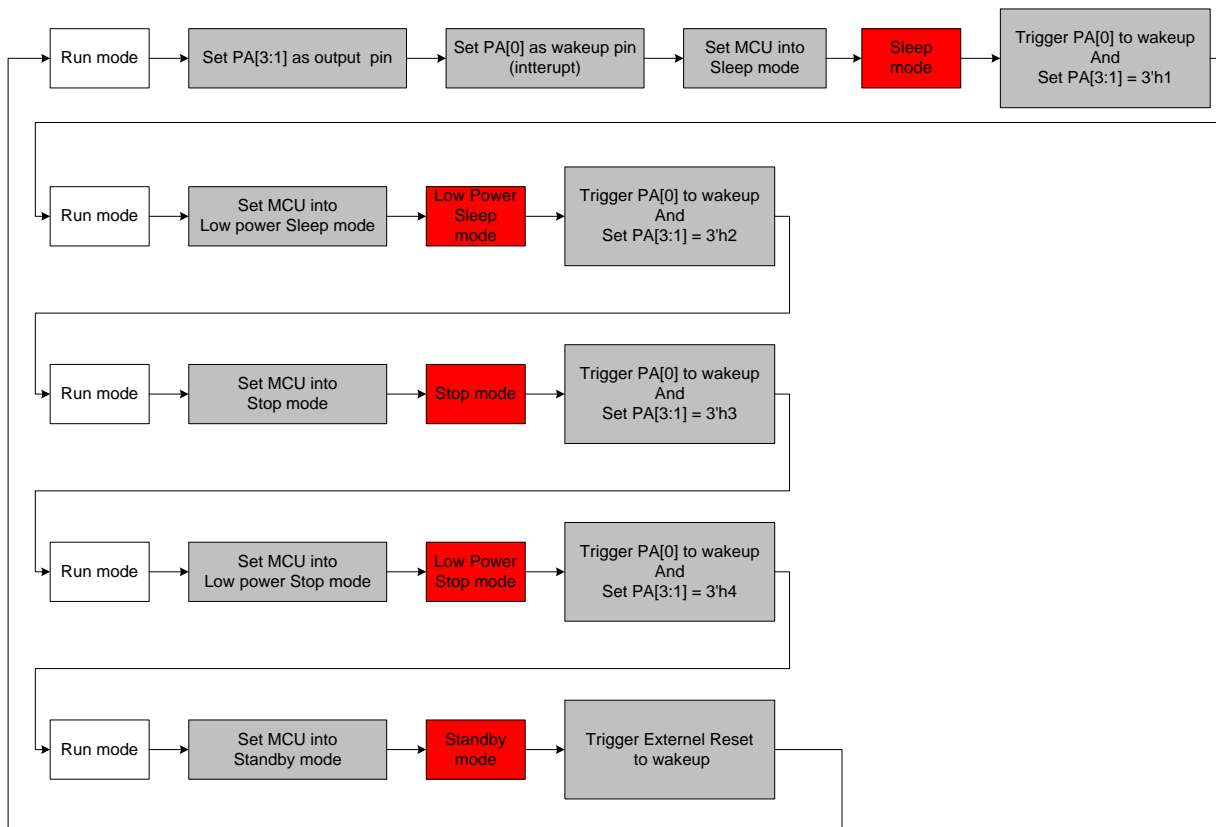


图 22 断电流程图

### 5.9 操作模式与唤醒源

操作模式:

Operating Mode	BLDO	SLDO	Low Freq. OSC.	High Freq. OSC	PLL	CPU	Peripherals	SRAM Data Retention
NORMAL	On	On	On	On	NA	On	NA	V
Low-Power RUN	Off	On	On	Off	Off	On	NA	V
SLEEP	On	On	On	On	Off	Off	NA	V
STOP	On	On	Off	On/Off(*1)	Off	Off	NA	V
Low-Power STOP	Off	On	Off	Off	Off	Off	Off	V
STANDBY	Off	Off	Off	Off	Off	Off	Off	X

\*1: 如果 BLDO 打开，让 HSI 保持外围设备工作

\*2: 如果 BLDO 关闭，高频时钟恐无法正常运作，必须靠软件关闭

唤醒源:

Power Down Mode	2 lines Wake up I/O PA0 & PC13	GPIO Wake up	COMP Wake up	DAC Wake up	ADC Wake up	RTC Wake up	IWDT Wake up	USB Resume
SLEEP	V(INT)	V(INT)	V(INT)	V(INT)	V(INT)	V(INT)	V (RESET)	V V(INT)
LOW POWER STOP	V(INT)	V(INT)	V(INT)	NA	NA	V(INT)	V (RESET)	V V(INT)
STANDBY	V (high level trigger) (RESET)	NA	NA	NA	NA	V (RESET)	V (RESET)	

功能概述

表格 9 操作模式上的外围

Ips	Run / Active	Sleep [wk up]	Low-power run	Low-power sleep[w k]	Stop		Low- power stop		Standby	
					Wakeup capability	Wakeup capability	Wakeup capability	Wakeup capability		
USB	0	0	--	--	--	0	--	0	--	--
DAC	0	0	0	0	0	--	0	--	--	--
ADC	0	0	--	--	--	--	--	--	--	--
CMP	0	0	0	0	0	0	0	0	--	--
WWDT	0	0	0	0	--	--	--	--	--	--
IWDT	0	0	0	0	0	0	0	0	0	0
SysTick Timer	0	0	0	0	--	--	--	--	--	--
PVD	0	0	0	0	0	0	0	0	--	--
GPIOs	0	0	0	0	0	0	0	0	--	--
RTC	0	0	0	0	0	0	0	0	0	0
2 lines Wake up I/O PA0 & PC13 (high level trigger)	0	0	0	0	0	0	0	0	0	0
Wakeup time to Run mode	0μs	~7.2μs	0μs	~7.2μs	~42μs		~42μs		~610μs	
Consumption VDD=3.0 V (Typ)	~8.9mA (32 MHz)	~330μA (MSI 4.2 MHz)	~202μA (MSI 1 MHz)	~265μA (MSI 4.2 MHz)	33μA (TYP.) (No RTC)		<b>0.6μA (TYP.)</b> (No RTC)		0.3 μA (TYP.) (No RTC)	
					33.7μA (TYP.) (with RTC)		<b>1.3μA (TYP.)</b> (with RTC)		0.9 μA (TYP.) (with RTC)	

“O” = Option 可选软件可以启动/关闭。 “TBD” = 尚未量测。

## 6 eFlash 控制

### 6.1 主要概述

NVM (Non-Volatile-Memory, 嵌入式非挥发闪存)为 64-bit 内存单元所组成, 可用于存储代码、数据、开机程序(Boot code)或可选字节(Option bytes)。内存数组被分为多个页面(Page), 每个页面(Page)由 128x64 位 (或 1024 bytes)组成, 且每个扇区(Sector)是由闪存程序代码区域中的 4K bytes 和 Option Bytes 区域 (原厂参数值)所组成。

- 读取接口支持字组(word)、半字组(half-word)或字节(byte)之读取
- 闪存(Flash Memory)之写入操作以字组(WORD)为单位
- 可选字节(Option bytes)区域之写入操作乃以字组(WORD)为单位
- 擦除(Erase)操作乃以页面(Page)为单位 (闪存和 Option bytes 皆是)
- 具有可选字节(Option bytes)加载器 (init\_load)
- 支持大量擦除(Mass Erase)操作
- 支持读或写入保护
- 支持 PCROP (Proprietary Code Read-Out Protecton, 专有代码读取保护)

### 6.2 方块图

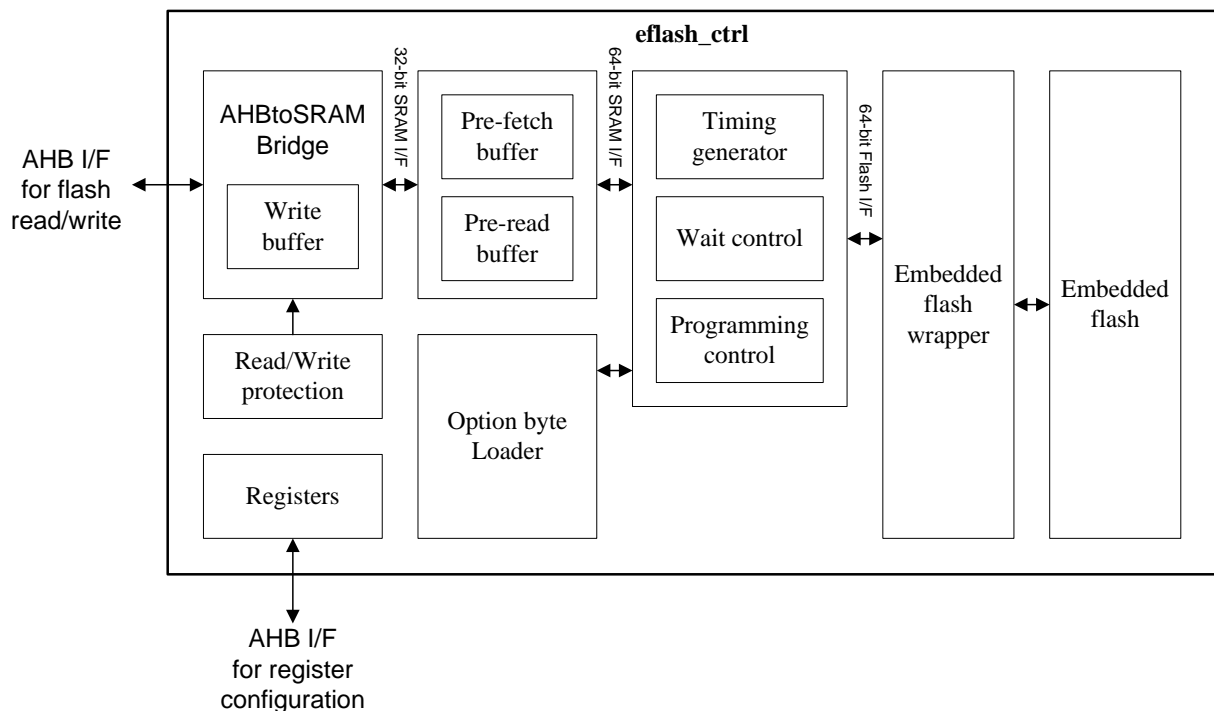


图 23 eFlash 控制方块图

**6.3 缓存器描述**

表格 10 eFlash 控制缓存器表

Index	Bit	R/W	Default	Name	Description
<b>FLASH_ACR: Flash access control</b>					
00h	31:7	-	-	-	保留
	6	R/W	0	prften	预取(Prefetch)启用设置: 0: 预取已禁用。 1: 预取已启用。当内存接口储存完最后提取的地址时,接着会尝试读取下一个地址(在没有其其它读取或写入操作正在进行时)。 注意: 每当 <b>disab_buf</b> 位 (在此缓存器中)设置为 1 时,它都会自动复位。要了解预取的工作原理,请参阅“提取和预取”部分。
	5	R/W	0	pre_read	预读(Pre-read)启用设置: 0: 预读已禁用 1: 预读已启用。当内存接口储存完最后一个数据读取的地址,接着会尝试读取下一个地址的数据(在没有其它读取或写入或预取操作正在进行时)。 注意: 每当 <b>disab_buf</b> 位 (在此缓存器中)设置为 1 时,它都会自动复位。
	4	R/W	1	disab_buf	快取(cache)缓冲区禁用设置(用于读取期间)。 0: 启用缓冲区 1: 禁用缓冲区。每次需要一个 <b>NVM</b> 值时,都会执行一个新的内存读取程序。这意味着,即使对于已读取过的地址(例如前一笔读取的地址),每次读取亦都需再次访问 <b>NVM</b> 。 注意: 复位此位时, <b>prften</b> 和 <b>pre_read</b> 位也会自动复位。
	3:2	-	-	-	保留
	1:0	R/W	0	latency	<b>NVM</b> 读取等待设置(0~3 Wait State)。 0: 0 Wait State(由 <b>NVM</b> 读取一个字组之等待时间)。 1: 1 Wait State(由 <b>NVM</b> 读取一个字组之等待时间) 2: 2 Wait State(由 <b>NVM</b> 读取一个字组之等待时间) 3: 3 Wait State(由 <b>NVM</b> 读取一个字组之等待时间) 用户必须依核心工作频率和操作模式(电源)来写入正确的值,请参照表 12 以取得正确的设定值, <b>MCU</b> 并不会自动检查设置是否正确。 若要增加时钟频率,用户必须先更改此值,然后再增加频率;若要降低时钟频率,用户必须先降低频率,然后再更改此值。
<b>FLASH_PECR: Flash program erase control</b>					
04h	31:18	-	-	-	保留
	17	R/W	0	errie	错误中断(Error interrupt) 启动设置: 0: 错误中断(Error interrupt) 禁能。 1: 错误中断(Error interrupt) 启动。 注意: 仅当 <b>pelock</b> 为 0 时,才能修改此位;而当



Index	Bit	R/W	Default	Name	Description
					此位启动时去设定 <code>pelock</code> 为锁定状态并不会复位此位, 此时中断保持于启用状态。
	16	R/W	0	<code>eopie</code>	写入完成中断(End of programming interrupt)启动设置: 0: 写入完成中断禁能. 1: 写入完成中断启动. 注意:仅当 <code>pelock</code> 为 0 时, 才能修改此位; 而当此位启动时去设定 <code>pelock</code> 为锁定状态并不会复位此位, 此时中断保持于启用状态。
	15:12	-	-	-	保留
	11	-	-	-	保留
	10	R/W	0	<code>mass</code>	大量擦除请求设置 0: 未请求大量擦除操作 1: 请求大量擦除操作 注意: 当 <code>pelock</code> 为 0 时, 才能修改此位; 当设置 <code>pelock</code> 时, 此位将会复位。
	9	-	-	-	保留
	8	R/W	0	<code>erase</code>	擦除请求设置(Page Erase) 0: 未请求擦除操作。 1: 请求擦除操作。 注意: 当 <code>pelock</code> 为 0 时, 才能修改此位; 当设置 <code>pelock</code> 时, 此位将会复位。
	7:3	-	-	-	保留
	2	-	-	-	保留
	1	R/W	1	<code>prglock</code>	闪存之程序代码区域(Program memory)锁定设置: 0: 解锁程序代码区域中的写入和擦除操作。 1: 锁定程序代码区域中的写入和擦除操作。 此位阻止闪存之程序代码区域之写入或擦除操作。它只能写入 1 来重新锁定, 若要将其复位为 0, 则需要使用 <code>flash_prgkeyr[31:0]</code> 缓存器来进行正确的解锁顺序 (请参阅“解锁闪存程序代码区域”), 当 <code>pelock</code> 位为 0 时, 解锁的钥匙为: - First key: 8C9D_AEBFh - Second key: 1314_1516h 注意: 当设置 <code>pelock</code> 为锁定时, 此位亦会跟着被设置为锁定。
	0	R/W	1	<code>pelock</code>	<code>flash_peocr[31:0]</code> 缓存器锁定设置(即此位所在之缓存器): 0: 解锁缓存器 <code>flash_peocr[31:0]</code> , 它可以被修改, 其它位解锁。启用数据写入或擦除操作。 1: 锁定缓存器 <code>flash_peocr[31:0]</code> , 此时无法启动写入或擦除操作。 此位锁定 <code>flash_peocr[31:0]</code> 缓存器, 它只能写入 1 来重新锁定, 若要将其复位为 0, 则需要使用 <code>flash_pekeyr[31:0]</code> 缓存器来进行正确的解锁顺序 (请参阅“解锁 FLASH_PECR 缓存器”)。解锁的钥匙为: - First key: 89AB_CDEF - Second key: 0203_0405

Index	Bit	R/W	Default	Name	Description
<b>Reserved</b>					
08h	31:0	-	-	-	保留
<b>FLASH_PEKEYR: Flash program erase key</b>					
0ch	31:0	W	-	flash_pekeyr	flash_pecr[31:0]解锁缓存器(唯写) 使用两个写入操作程序来进行解锁 (第一个操作为 89AB_CDEFh, 第二个操作为 0203_0405h), 写入大小为字组, 可以解锁 flash_pecr[31:0] 缓存器。详细信息, 请参阅“解锁 FLASH_PECR 缓存器”。
<b>FLASH_PRGKEYR: Flash program memory key</b>					
10h	31:0	W	-	flash_prgkeyr	闪存之程序代码区域解锁缓存器(唯写) 使用两个写入操作程序来进行解锁 (第一个操作为 8C9D_AEBFh, 第二个操作为 1314_1516h), 写入大小为字组, 可以解锁闪存之程序代码区域。 注意: 仅当 pelock 已解锁时才能执行此程序来进行解锁, 详细信息请参阅“解锁闪存程序代码区域”。
<b>FLASH_SR: Flash status</b>					
18h	31:14	-	-	-	保留
	13	R/W1 c	0	rderr	读取保护错误旗标 0: 未发生读取保护错误。 1: 发生读取保护错误。 当用户尝试读取受 PCROP 保护的区域时, 硬件会设置此位, 通过写入 1 来进行清除。
	12	-	-	-	保留
	11	R/W1 c	0	optverr	Option bytes 区域有效内容错误旗标 0: 在 Option bytes 加载期间未发生错误。 1: 在 Option bytes 加载期间发生一个或多个错误 当 Options bytes 加载期间, 一个或多个配置不匹配时, 硬件会设置此位, 这意味着加载的配置可能与用户在内存中写入的配置不同。通过写入 1 来进行清除。 注意: 如果在加载保护相关位时发生错误 (wprmod、rdprot[7:0]、wrprot[x]), 则 Flash 程序内存中的源代码可能无法正确执行。
	10:9	-	-	-	保留
	8	R/W1 c	0	wrperr	写入保护错误旗标 0: 未发生写入保护错误。 1: 发生写入保护错误。 当程序所要写入或擦除的地址为写入保护时, 硬件会设置此位, 通过写入 1 来进行清除。
	7:4	-	-	-	保留
	4	R	0	lve	低电压读取状态旗标 0: NVM 处于正常电压读取模式 1: NVM 处于低电压读取模式
	3	-	-	-	保留
	2	R	1	endhv	高电压结束状态旗标 0: NVM 正处于高电压状态以执行写入或擦除操作。 1: NVM 未处于高电压状态, 即此时没有写入或擦除

Index	Bit	R/W	Default	Name	Description
					操作正在进行。 注意: 此位由硬件进行设置与复位。
	1	R/W1 c	0	eop	写入结束 (EOP) 状态旗标 0: 未发生写入结束(EOP) 事件 1: 发生写入结束( EOP) 事件, 此时若设置了 eopie 位, 则会产生中断。 当完成一未被中止之写入或擦除操作时, 硬件会将此位设置为 1, 软件通过写入 1 来进行清除。
	0	R	0	bsy	内存接口忙碌旗标 0: 未进行写入或擦除操作。 1: 写入或擦除操作正在进行中。
<b>Reserved</b>					
1ch	31:0			-	保留
<b>FLASH_INIT0: Analog initial load</b>					
20h	31:0			-	保留
<b>FLASH_INIT1: Analog initial load</b>					
24h	31:0			-	保留
<b>FLASH_OPTR: Boot ROM option</b>					
28h	31:30	R/W	-	nboot	自举模式选择 0: boot from ROM(remap 0x00000000 to ROM) 1: boot from Flash(remap 0x00000000 to Flash) 2: boot from system RAM (remap 0x00000000 to system RAM) 此位用于选择设备之自举模式, 如果在 Option bytes 加载期间此值发生配置不匹配, 则硬件自动加载 0 待机复位后, 如果在 Option bytes 加载期间此值匹配正确, 则此位会自动被设置。
	29:9	-	-	-	保留
	8	R	-	wprmod	写入或读取保护模式 (闪存程序代码区域) 0: PCROP=0, 此时 flash_wrprot[x] 用于扇区 (Sector) 写入保护之设置。 1: PCROP=1, 此时 flash_wrprot[x] 用于扇区 (Sector) 读取保护之设置。(同时也无法写入) 如果在 Option bytes 加载期间发生此值配置不匹配, 则硬件自动加载 1
	7:0	R/W	-	rdprot	读取保护等级 Aah: level 0 (可读) 其它: level 1 (防读) 此值记录读取保护等级, 于 Option bytes 加载期间自动加载。如果在 Option bytes 加载期间发生此值配置不匹配, 则硬件自动加载 0x00。 注意: 软件只能从未保护改写为保护, 无法在已保护下改写为未保护

FLASH_WRPROT0: Write protection					
2ch					扇区(sector) 0~31 保护设置
	31:0	R/W	0	flash_wrprot[31:0]	<p>1. 当 wprmod[0] = 0: 在闪存内容的缓存器中, 这些位为闪存的写入保护设定 (每个位保护 4KB 字节的扇区:第一个位保护第一个扇区, 第二个位保护第二个扇区等)。在这种情况下, 1 = 扇区受写入保护, 0 = 无保护。这些位以及其相对应的 Option bytes 的内容可以随意进行设置或复位, 并无任何限制。</p> <p>2. 当 wprmod[0] = 1: 则这些位用于读取保护设定 (请参阅“数据读取与预读”), 开启读取保护的同时亦无法进行写入, 而读取保护并不会防止程序代码提取。在这种情况下, 1 = 无保护, 0 = 扇区读取保护。这些位只能增加保护(可将某位由 1 设置为 0, 但不能将原本为 0 的部分设置为 1)</p> <p>大量擦除(mass erase)时将复位 wprmod 位, 但不会复位此缓存器的内容, 所以大量擦除后, 使用者必须写入具有零的相对 Option bytes 才能完全删除写入保护。</p> <p>如果在 Option bytes 加载期间此配置发生不匹配的状况, 并且 FLASH_OPTR 缓存器中的 wprmod=1, 则将会加载 0000h 至此缓存器; 反之若 wprmod=0, 则将会加载 FFFFh 至此缓存器。</p> <p>如果在 FLASH_OPTR 缓存器中加载 wprmod 时出现不匹配的状况 (因此加载了 1), 则将加载 0000h 至缓存器。</p>
FLASH_WRPROT1: Write protection					
30h	31:0	R/W	0	flash_wrprot[63:32]	扇区(sector) 32~63 保护设置, 详细内容请参考上方 flash_wrprot[31:0]之说明
<b>Reserved</b>					
40h	31:0				保留
<b>Reserved</b>					
44h	31:0				保留
<b>Reserved</b>					
48h	31:0				保留
<b>Reserved</b>					
4ch	31:0				保留
<b>Reserved</b>					
50h	31:0				保留

**R/W1C:** 读取 与 写“1”清除

## 6.4 功能描述

### 6.4.1 NVM 功能描述

NVM (Non-Volatile-Memory, 嵌入式非挥发闪存)为 64-bit 内存单元所组成, 可用于存储代码、数据、开机程序 (Boot code)或可选字节 (Option bytes)。内存数组被分为多个页面 (Page), 每个页面 (Page)由 128x64 位 (或 1024 bytes)组成; 每个扇区 (Sector)乃由闪存程序内存中的 4K bytes 和 Option bytes 区域 (原厂参数值)所组成。表格 11 显示了 NVM 组织, 包括主区块中的 64 页面。

表格 11 NVM 组织

NVM	NVM address	Page	Sector	Description
Main memory block	1000_0000h – 1000_03FFh	0	0	Flash program memory
	1000_0400h – 1000_07FFh	1		
	1000_0800h – 1000_0BFFh	2		
	1000_0C00h – 1000_0FFFh	3		
	...	...	15	
	1000_F000h – 1000_F3FFh	60		
	1000_F400h – 1000_F7FFh	61		
	1000_F800h – 1000_FBFFh	62		
	1000_FC00h – 1000_FFFFh	63		

### 6.4.2 读取 NVM

➤ 读取保护

根据时钟频率, 读取 NVM 可能需要 0~1 Wait-State, 使用者必须设置正确的 Wait State 数量 (FLASH\_ACR 寄存器中的 latency [1:0]位), 因对于 Wait State 所设置的的数量并不会自动验证所设置的值是否正确。错误的 Wait State 数量的可能会造成读取值的错误 (高频时钟下设置为 0 个 wait states)或拉长执行代码的时间 (低频时钟下设置为 3 个 wait-state)。

程序可以按字组 (WORD=4 字节)、半字组 (half-word=2 字节)或字节来读取 NVM, 但在写入或擦除操作期间则会无法读取。如果写入或擦除操作正在进行, 则读取将处于等待 (Wait-State)状态, 直到写入或擦除操作完成, 此时硬件将会延迟读取操作之要求, 除非此时所读取的地址为读取保护, 在这种情况下, 错误会通过硬件故障 (hard-fault)或内存接口旗标发送给主控器, 此时不会拖延与等待。表格 12 显示了在 NVM 中读取字组所需的延迟 wait-state。

表格 12 等待状态数 (wait-states)

range	Wait state required
≤ 16MHz	0
> 16MHz, ≤ 32MHz	1

➤ 更改 CPU 频率

复位后, 预计所使用的时钟是 MSI (2.1 MHz), 闪存缓存器中会配置为 0 Wait-State, 此时必须遵守以下软件程序, 以调整 CPU 时钟频率以及访问 NVM 所需的 Wait State 数。CPU 时钟或 Wait State 配置之更改可能需要一些时间才能生效, 可利用检查 AHB 预除器 (pre-scaler) 之因子 (factor) 和时钟源状态值来确认 CPU 时钟频率是否即与所设置的频率相同, 亦可利用读取 FLASH\_ACR 缓存器来确保程序有正确设计 Wait State 数量。

■ 增加 CPU 频率

1. 将 FLASH\_ACR 缓存器中的延迟 latency[1:0] 位设定为 1 个 Wait-State (如有必要)。
2. 通过读取 FLASH\_ACR 缓存器, 检查是否 Wait State 数量已为新的设定值。当 Wait State 的数量发生更改时, 内存接口将更改对 NVM 进行读取存取的方式, 若此时读取操作正在进行时, 则无法修改 Wait State 的数量, 因此内存接口将会进行等待, 直到 NVM 上没有未完成之读取。

当主控器读回 FLASH\_ACR 缓存器的内容, 此时读取动作及要求之主控器将会停止, 直到 Wait State 的数量真正更改完之。如果用户不读回缓存器, 则之后对 NVM 的访问可能会以 0 Wait State 的时间完成, 就算是时钟频率增加, 此时去计算该值也会是错误的。

3. 在复位和时钟控制器 (RCC)中修改 CPU 时钟源或者 AHB 时钟预除器(prescaler)。
4. 通过分别读取复位与时钟控制器 (RCC)中的时钟源状态或者 AHB 预除器(prescaler)值, 检查是否已是新的 CPU 时钟源或新的 CPU 时钟预除器(prescaler)值。此检查很重要, 因为某些时钟可能需要一些时间才能完成设置与使用。

■ 降低 CPU 频率

1. 在复位和时钟控制器 (RCC)中修改 CPU 时钟源或 AHB 时钟预除器(prescaler)。
2. 通过分别读取复位和时钟控制器 (RCC)中的时钟源状态或者 AHB 预除器(prescaler)的值, 检查是否已是新的 CPU 时钟源或者新的 CPU 时钟预除器(prescaler)值。
3. 将 FLASH\_ACR 缓存器中的延迟 latency[1:0] 位设定为 0 Wait State (如果需要)。
4. 通过读取 FLASH\_ACR 缓存器, 检查是否已是新的 Wait-State 数。由于上一段解释的种种因素, 有必要重新读取检查该缓存器。

➤ 数据缓冲 (Data buffering)

在 NVM 中, 有六个缓冲区可影响读取操作 (用于提取和数据)的性能, 且在某些情况下有助于降低功耗。缓冲区的结构如图 24 所示, 每个缓冲区存储 3 种不同类型的信息:地址、数据和历史记录。在读取操作中, 如果在缓冲区中找到要读取之地址, 则内存接口可以在不访问 NVM 的情况下返回数据。缓冲区中的数据为 32 位宽 (即使主服务器只读取 8 bit 或 16 bit), 因此无论该值在以前的读取中使用的大小如何皆可以直接传回值。历史记录用于了解缓冲区的内容是否有效, 并删除较旧的缓冲区 (以存放新值)。



图 24 一个内部缓冲区的结构

缓冲区用于存储正常读取操作期间从 NVM 所接收的值，以及用于存放推测读取的值。禁用推测读取下将使得仅有主控器所要求的数据会存储在缓冲区中 (预设为启用)。如果要读的值已在缓冲区中，则可以增加性能，因为该值已在缓冲区中，此时则无需 Wait-State；同时随着内存中的读取次数减少且内存中的所有组合路径都稳定，因此降低了功耗。

缓冲区被划分为多个群组来管理不同的任务，每个群组中的缓冲区数量可以从用户选择的配置而更改 (参见表格 13)。使用的缓冲区总数始终为 6 (已启用下)，历史记录始终以群组来进行管理，内存接口始终于所有缓冲区之中搜寻特定地址是否已经存在，而非检查缓冲区群组或读取是提取指令或数据。

在复位时或进行数个地址更新之写入/擦除操作后，所有缓冲区都会设为空值，历史记录设置为 EMPTY；而在编程使用字组、半字组或字节的程序后，则仅清除具有相关地址的缓冲区。

**表格 13 预取-预缓冲管理 (Pre-fetch & Pre-buffer)**

disab_buf	prften	pre_read	Buffers for fetch			Buffers for data	
			Buffers for jumps	Buffers for prefetch	Buffers for last value	Buffers for pre-read	Buffers for last value
1	-	-	0	0	0	0	0
0	0	0	3	0	1	0	2
0	1	0	2	1	1	0	2
0	0	1	3	0	1	1	1
0	1	1	2	1	1	1	1

如果缓冲区中的值不为空，则历史记录将显示读取或写入该值所花费的时间。历史记录乃由一个从最新到最旧值的列表所组成，在给定的瞬间，群组中只有一个缓冲区可以具有特定的历史记录值 (空值除外)。当将缓冲区移动到最新位置时，群组中的所有其它缓冲区将同时移动一步，从而维持顺序。当缓冲区被读取 (主控器要求读取缓冲区的内容)或被写入 (从 NVM 中取得新的值)时，历史记录将更改为最新位置。当内存需要新地址时，内存接口始终覆写至正确群组中之最旧的一个缓冲区 (若有空的缓冲区则直接写入空的缓冲区)。

位于闪存缓存器(FLASH\_ACR)中有三个设定值可用于管理缓冲区：

■ **disab\_buf**

设置此位将禁用所有缓冲区。当此位为 1 时，无法启用预取或预读操作，例如，如果主控器请求同一地址两次，则在 NVM 中产生两次读取。

■ **prften**

将此位设置为 1 (需先将 disab\_buf 设置为 0)可启用预取。当内存接口没有任何操作正在进行时，将读取最后获取的地址后的地址并将其存储在缓冲区中。

■ **pre\_read**

将此位设置为 1 (需先将 disab\_buf 设置为 0)可启用预读。当内存接口没有任何操作进行中或预取执行时，将读取最后一个数据地址后的地址并将其存储在缓冲区中..

➤ **提取与预取 (Fetch and prefetch)**

内存接口提取是指从 NVM 读取数据并执行已读取之数据。内存接口不会检查执行读取操作的主控器及其所读取之位置，它仅验证读取操作是否完成且执行已读取之数据。这意味着可以执行提取于：

- 所有区域
- 任意大小(16 或 32 位)

内存接口存储在缓冲区中的内容:

- 跳转的地址。因此，在循环中，只需于第一次访问时进行 NVM 读取，后续操作则因为跳转地址已经位于缓冲区中已不再需要进行 NVM 读取。
- 最后一个读取地址。所以当执行提取 16 位数据时，其它 16 位数据业已位在缓冲区中。

为了管理提取，内存接口使用 4 个缓冲区:复位时 (`disab_buf[0] = 0`, `prften[0] = 0`, `pre_read[0] = 0`)，3 个缓冲区用于管理跳转(jumps)，1 个缓冲区用于存储获取的最后一笔数据(last value)。使用此配置，4 个用于提取的缓冲区被组织成 2 个群组，具有独自的历史记录:循环群组和获取最后一笔数据的群组。

将 `prften` 位设置为 1 可启用预取(prefetch)。预取是 NVM 中的推测读取，在主控器不请求读取时执行，并且从获取的最后一个地址处再额外读取一个字组的数据(4 个字节)，此读取具有较低的优先级，如果主控器请求读取 (数据或提取)至与预取地址不同的地址，则此读取将中止。启用预取时，循环群组中的一个缓冲区将移动到一个新群组 (仅使用一个缓冲区)，用以储存预取值，此时: 2 个缓冲区继续存储跳转(jumps)，1 个缓冲区用于预取(prefetch)，1 个缓冲区用于最后一笔数据(last value)。

内存接口可以仅预取一个地址，所以当未进行提取且预取已完成下该功能会暂时停用。预取完成后，如果主控器请求预取的值，则预取缓冲区中的内容将被复制到最后一个值缓冲区且新的预取动作会启用。相反，如果主控器请求不同的地址，预取缓冲的内容丢失，此时将进行 NVM 读取 (如果需要)，并在完成后，一新的预取动作开始预取 NVM 读取之新地址再加 4 之地址。

预取仅能增加效能于 1 个 wait state 下之读取以及大多数为连续线性的程序代码:使用者必须评估利弊并依状况来决定是否要启用预取。预取会造成耗电增加，因为会有较多 NVM 读取操作 (但并非所有的预取数据都会被使用到)。

#### ➤ 数据读取与预读 (pre-read)

从内存接口读取的数据，对应于不是提取的任何读取操作，主控器读取操作之常数和参数皆为数据。复位时，(`disab_buf[0] = 0`, `prften[0] = 0`, `pre_read[0] = 0`)，内存接口使用在 2 个缓冲区所组成的一个群组来存储最后两个值作为数据读取。

在某些特殊情况下，启用预读 (`pre_read[0] = 1`，带 `disab_buf[0] = 0`)可能很有用。预读(pre-read)的工作方式与预取完全一样:它是最后一个数据地址的推测性读取，每次递增 4 个字节 (一个字组)。使用此配置，一个数据缓冲区将移动到新群组以存储预读值，而第二个缓冲区继续存储读取的最后一个值。对于预取，如果主控器请求预读值，则在最后一个读取值中复制预读值;如果主控器请求其它地址，则该值将丢失。

预读(pre-read)的优先级低于正常读取或预取操作:这意味着只有在没有其它类型的读取正在进行时才会启动预读。请注意，在错误情况下使用的预读可能是有害的:在代码中，数据读取不是线性完成的，减少用于最后读取值的缓冲区数 (从 2 到 1)可能会增加对 NVM 的访问次数 (以及访问 NVM 的时间)。此外，这还会导致预取延迟。表格 14 是可能之配置摘要。

表格 14 缓冲区和推测读取的规划

disab_buf	prften	pre_read	Description
1	X	X	Buffers 禁能
0	0	0	启动缓冲区:不进行推测读取
0	1	0	启动预取:启动提取时的推测读取
0	0	1	启动预读:对启动的数据进行推测读取
0	1	1	启动预取和预读:提取的推测读取和启动的数据



### 6.4.3 写入与抹除 NVM

有许多方法可以更改 NVM 之内容。内存接口有助于减少发生 NVM 内容被意外更改的可能性，并通过硬件实现在不同内存区域中进行擦除或写入所需的所有程序。

➤ **写入与抹除协议 (Write/erase protocol)**

要在移除保护时写入或擦除内存内容，用户需要：

1. 将操作配置为执行(写入或抹除)。
2. 向内存接口发送正确数量的数据，在 NVM 中写入一个或多个地址。
3. 等待操作完成。

在等待期间，使用者可以准备下一个操作 (在非常特殊的情况下除外)，编写新配置并开始为下一个写入或擦除操作写入数据。

➤ **单写入操作方式 (Single programming operation)**

使用此协议，软件必须将值写入 NVM 的未受保护的地址中。当内存接口收到此写入请求时，它会在检查保护和上一个值时，将主时钟脉冲停止，并在 NVM 内锁定新值。然后，软件可以开始配置下一个操作。当闪存缓存器 FLASH\_SR 的 eop 位被设置为 1 时 (如果操作开始时为 0)代表操作将完成。

### 6.4.4 锁定与解锁操作

在执行写入或擦除操作之前，必须进行解锁操作，以便用户可以写入闪存缓存器、程序内存和 Option bytes 区域。要执行写入或擦除操作，请解锁 FLASH\_PECR 缓存器的 pelock 位，当此位解锁 (其值为 0)时，即可修改同一缓存器的其它位。

要写入或擦除闪存程序代码区域，请解锁闪存缓存器的 prglock 位，但仅有在 pelock 为 0 时，才能解锁该位。

#### 解锁 FLASH\_PECR 缓存器

复位后，由于在 FLASH\_PECR 缓存器中设置了 pelock 位，因此 FLASH\_PECR 缓存器无法进行写入操作，相同的解锁程序同时取消此两者之保护。以下程序用于解锁 FLASH\_PECR 缓存器：

1. 写入第 1 个 KEY 值(PEKEY1 = 0x89ABCDEF)至缓存器 FLASH\_PEKEYR
2. 写入第 2 个 KEY 值(PEKEY2 = 0x02030405)至缓存器 FLASH\_PEKEYR

任何错误的 KEY 顺序列都会产生硬件错误(hard fault)，即使主控器尝试在两个 KEY 顺序之间编写另一个缓存器，或者使用错误的 KEY 值，亦为如此；但读取操作并不会产生错误，也不会中断程序。在以下四种情况下，将会返回硬件错误(hard fault)：

- 在第一次写入 KEY 后，且输入的 PEKEY1 值错误。
- PEKEY1 输入正确，但在第二次写入的 PEKEY2 的值不匹配。
- 尝试将第三个值写入 flash\_pekeyr (请注意:仿真中也是如此)。
- 尝试在 PEKEY1 和 PEKEY2 之间进行另一内存接口缓存器之写入操作。

当正确执行后，解锁程序将清除 FLASH\_PECR 缓存器中的 pelock 位，若要再次锁定 flash\_pecr，软件只需将 flash\_pecr 中的 pelock 位设置为 1 即可。重新锁定时，pelock 位需要一个新解锁程序才能返回到 0。

➤ **解锁闪存之程序代码区域**

另一额外的保护用来防止写入或擦除闪存之程序代码区域。复位后，闪存之程序代码区域无法进行写入操作：在 FLASH\_PECR 缓存器中设置了 prglock 位，通过清除 prglock 位可授予对闪存之程序代码区域的写入访问权限。以下程列用于解锁闪存之程序代码区域：

1. 解锁 FLASH\_PECR 缓存器 (请参阅上一章节“解锁 FLASH\_PECR 缓存器”)
2. 写入第 1 个 KEY 值( PRGKEY1 = 0x8C9DAEBF)至缓存器 FLASH\_PRGKEYR
3. 写入第 2 个 KEY 值( PRGKEY2 = 0x13141516)至缓存器 FLASH\_PRGKEYR

如果在 pelock 的值为 1 时写入 KEY 值，并不会产生错误，并且 prglock 的值将会依然为 1，此时若要解锁则必须在使 pelock = 0 之下重新执行解锁程。在以下四种情况下，将会返回硬件错误(hard fault)：

- 在第一次写入 KEY 后，且输入的 PRGKEY1 值错误。
- PRGKEY1 输入正确，但在第二次写入的 PRGKEY2 的值不匹配。
- 尝试将第三个值写入 flash\_prgkeyr (请注意:仿真中也是如此)。
- 尝试在 PRGKEY1 和 PRGKEY2 之间进行另一内存接口缓存器之写入操作。

当正确执行时，解锁程序将清除 prglock 位，并且闪存之程序代码区域可进行写入操作，若要再次锁定闪存程序代码区域，软件只需将 FLASH\_PECR 缓存器中的 prglock 位设置为 1 即可。重新锁定时，prglock 位需要一个新的解锁程序才能返回到 0。如果 pelock 的值返回到 1 (锁定)，则 prglock 也会自动跟着锁定。

#### 6.4.5 状态缓存器

FLASH\_SR 状态缓存器提供有关内存接口或 NVM 状态 (操作是否正在进行)和发生的错误信息。

##### ➤ bsy

此旗标由硬件设置和复位。每次内存接口执行写入或擦除操作时，它都会被设置为 1 以通知此时无法执行任何其它操作。如果此时请求新操作，则可能发生以下不同的行为：

- 等待读取、或等待写入或擦除:如果软件在执行写入或擦除操作时再请求一写入操作 (hvoff = 0)，则内存接口将暂停主控器，并在前一写入或擦除操作完成后立刻执行被暂停的工作。
- rderr 错误:  
如果软件在执行写入或擦除操作时再请求一读取操作 (hvoff = 0)，但欲读取的地址受到保护，则内存接口将举起 rderr 旗标并继续等待前一写入或擦除操作的结束。

##### ➤ eop

此旗标由硬件设置，并由软件进行复位。软件可透过将其写 1 来将其复位，当写入或擦除操作完成且内存接口可以处理其它操作 (或开始一个被暂停的操作)时，此位将会被设置为 1。

在开始新的写入或擦除操作之前清除此位可以帮助知道实际的操作何时完成。当在进行大量擦除动作时等待此位旗标被设置 1 乃是非常重要的事，如此方能得知何时方可进行下一个新的操作。

##### ➤ hvoff

此旗标由硬件设置和复位，它是一个来自于 NVM 内存接口之信息：它通知此时高压稳压器(High-Voltage Regulators)是打开 (= 0)或关闭 (= 1)。

## 6.5 内存保护

用户可以保护部分的 NVM (闪存程序代码区域和 Option bytes 区域), 以免受意外的写入和黑客代码攻击 (意外之代码读取), 为此导入了三种类型的保护。

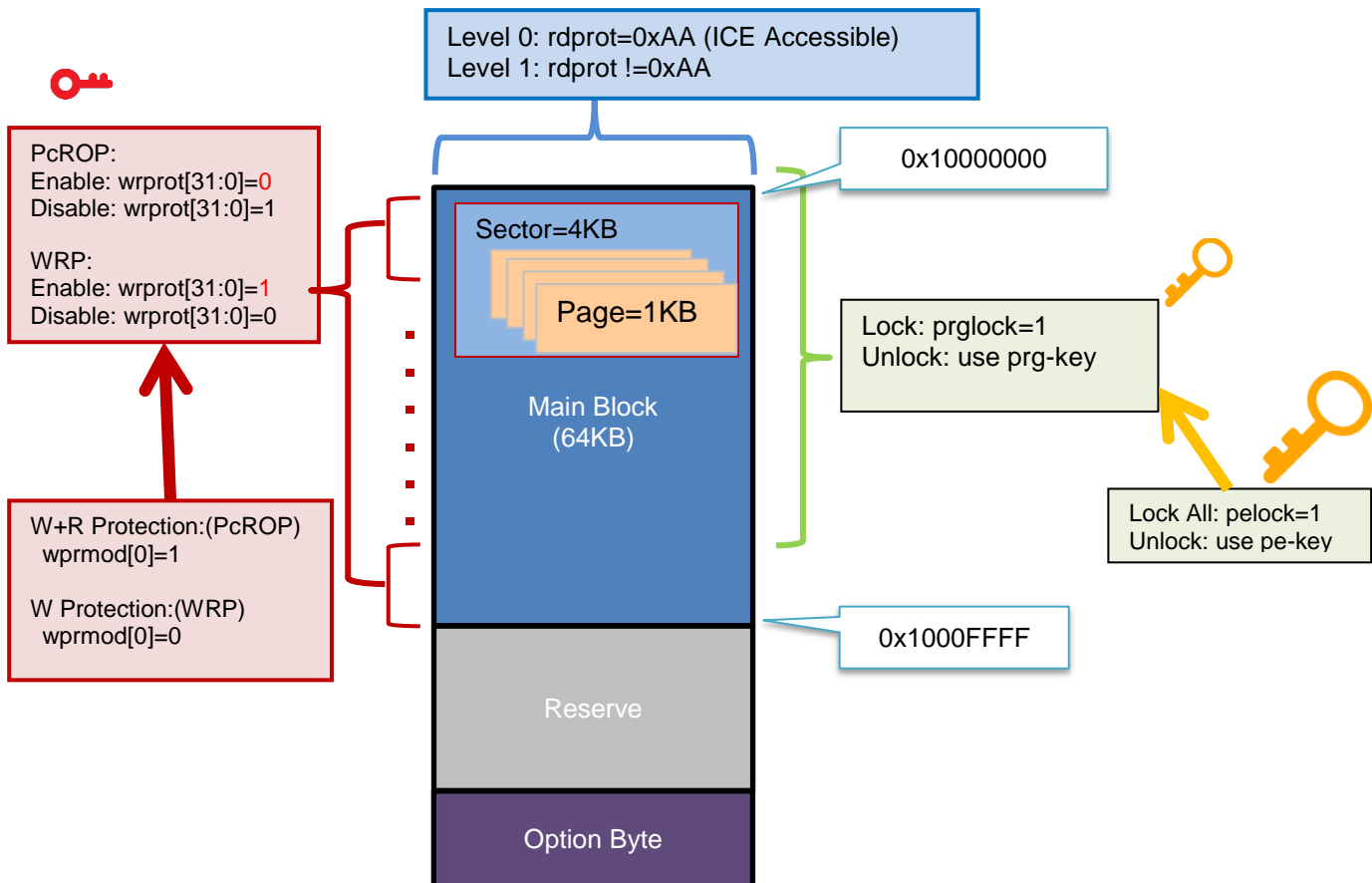


图 25 内存保护

### 6.5.1 读取保护 RDP (Read Out Protection)

这种类型的保护旨在防止 NVM 内容之意外读取 (黑客攻击)。此保护由 FLASH\_OTPR 寄存器中的 rdprot[7:0] 位来管理, 该值在开机(boot)期间从 Option bytes 区域加载, 并复制到寄存器中。

定义了两个保护层级:

- Level 0: 无保护

当 rdprot[7:0] 设置为 0xAA 时, 此时即 level 0。启用此级别时, 如果没有启用其它保护, 则可以在快闪程序内存和 Option bytes 区域中无限制地进行读取和写入。

- Level 1: 闪存读取保护

当 rdprot[7:0] 设置为除 0xAA 以外的任何值时, 此时即 level 1。当 Option bytes 被擦除或是 rdprot[7:0] 字段中的值发生不匹配时, 此时默认之保护等级即为此保护层级。

当设置为 Level 1 等级, 此时程序代码区域为防读保护。

启用此级别后:

- 当连接了调试器 (single-wire), 亦无法执行对闪存程序代码区域之访问 (程序代码与数据之读取与写入), 但对其它区域则没有限制: 可以读取、写入或擦除 Option bytes 区域。
- 当完成启动于闪存之程序代码区域中时, 所有操作能进行。
- 大量擦除将会删除闪存之程序代码区域, 并将保护层级更改为 Level 0, 删除 Option bytes 之第三个字组然后进行重写以启用 Level 0 并禁用 PCROP (WPRMOD = 0)。

### 6.5.2 资产保护 PcROP (Proprietary Code Read-Out Protection)

闪存之程序代码区域可以防止被黑客代码读取: 读取的数据会被阻止 (不含程序代码提取)。受保护的代码不得访问受保护区域中的数据, 包括文字池(literal pool)。闪存之程序代码区域可以防止黑客代码进行读取: 这会阻止读取资料 (不含程序代码提取), 假设本机代码是根据 PCROP 选项编译的。此模式透过设置 FLASH\_OPTR 寄存器中的 wprmod 位为 1 来启动。

保护设置以是扇区(sector)为单位(1 sector=4 pages=4KB), 为了保护扇区, 需将 wrprot[63:0] 对应的位设置为 0: 设置 0 表示读取和写入保护; 1 则表示没有保护。表格 15 显示了 wrprot[15:0]配置位与 64KB 闪存 16 扇区地址之间的链接。

在受保护的扇区中作为数据执行的任何读取操作 (请参阅“读取数据和预读部分”)都将在 FLASH\_SR 寄存器中触发 rderr 旗标。任何受读取保护的扇区也受写入保护, 对这些扇区任何的写入访问都将触发 FLASH\_SR 寄存器中的 wrperr 旗标。

**表格 15 PcROP & wrprot 闪存保护范围**

Bit	Start address	End address	Bit	Start address	End address
0	1000_0000h	1000_0FFFh	16	1001_0000h (保留)	1001_0FFFh (保留)
1	1000_1000h	1000_1FFFh	17	1001_1000h (保留)	1001_1FFFh (保留)
2	1000_2000h	1000_2FFFh	18		
3	1000_3000h	1000_3FFFh	19		
4	1000_4000h	1000_4FFFh	20		
5	1000_5000h	1000_5FFFh	21		
6	1000_6000h	1000_6FFFh	22		
7	1000_7000h	1000_7FFFh	23		
8	1000_8000h	1000_8FFFh	24		
9	1000_9000h	1000_9FFFh	25		
10	1000_A000h	1000_AFFFh	26		
11	1000_B000h	1000_BFFFh	27		
12	1000_C000h	1000_CFFFh	28		
13	1000_D000h	1000_DFFFh	29		
14	1000_E000h	1000_EFFFh	...	...	...
15	1000_F000h	1000_FFFFh	63	1003_F000h (保留)	1003_FFFFh (保留)

当 wprmod=1 (启用 PCROP)时, 无法删除对扇区的保护: 可以于 wrprot[63:0]设置新的零 (保护新扇区), 但不能设置新的 1 (从扇区中删除保护), 无论保护等级 (rdprot 的配置)为何皆为如此。当 wprmod 处于活动状态时, 如果使用者尝试复位 wprmod 或从扇区中删除保护, 此时程序会持续执行, 但 wprmod 位或受保护的扇区保持不变。

要从扇区(sector)中移除保护的唯一方法是请求大量擦除 (即 mass erase, 会将保护等级更改为 Level 0 并禁用 PcROP): 禁用 PCROP 时, 即可自由更改扇区(sector)保护。

### 6.5.3 防止意外的写入或擦除操作

内存接口支持两种方法来防止意外的写入或擦除操作，这些操作可对所有闪存之区域有效，或者仅限于闪存之程序代码区域之特定扇区(sectors)。

如“解锁或锁定操作”部分所述，使用者可以：

- 仅当 `pelock = 0` 和 `optlock = 0` 时，才能写入或擦除到 Option bytes 区域。
- 仅当 `pelock = 0` 和 `prglock = 0` 时，才能写入或擦除到闪存之程序代码区域。

要了解设置 `pelock`、`prglock` 和 `optlock` 的程序，请参阅“解锁 FLASH\_PECR 缓存器”、“解锁闪存程序代码区域”。

在闪存之程序代码区域中，可以使用扇区 (sector) 为单位来添加另一个写入保护。禁用 PCROP 时 (`wprmod[0] = 0`)，使用 `wrprot[15:0]` 位来启用扇区 (sector) 上的写入保护。与 PcROP 相比，极性是相对的：为了保护扇区 (sector)，必须将相对的位设置为 1；要删除保护，则必须要设置为 0。表格 15 之内容对于写入保护亦为有效。如上所述，启用 PCROP 时，防止读取的扇区 (sector) 也受到写入与擦除保护，可防止写入或擦除之操作。不管是在 Level 0 或 Level 1 中皆可以随时更改扇区 (sector) 上的写入保护。表格 16 中简述各种保护状态。

**表格 16 内存访问与模式**

Flash program memory sectors	Mode	
	User mode (including In Application Programming)	Debug mode
Flash program memory (RDP Level Setting)	R/W (Level 1 or 0)	Level 0 (rdprot[7:0]=0XAA)
		Level 1 (rdprot[7:0]≠0XAA)
Flash program memory (prglock[0] = 1)	R	Protected (no access)
Flash program memory (prglock[0] = 0)	R/W	Protected (no access)
Flash program memory in WRP pages	R	Protected (no access)
Flash program memory in PcROP pages	Fetch	Protected (no access)

### 6.5.4 读取 NVM

以下为所有前面提到的保护更改规则之摘要：

- 在 Level 1 时进行大量擦除(mass erase)后，保护等级会减小到 Level 0。
- PCROP 可在请求进行大量擦除(mass erase)时删除。
- 启用 PCROP 时，可以添加受保护的扇区 sector (写入 0)，但不能移除保护。当不匹配发生时会影响所有受读和写入保护的扇区 (如果启用了 PCROP)。

### 6.5.5 保护错误 (Protection Errors)

➤ 写入保护错误旗标(wrperr)

如果对闪存之程序代码区域中受写入保护的页面进行擦除与写入操作，则硬件在 FLASH\_SR 缓存器中设置写入保护错误旗标 (wrperr)。因此，当软件尝试：

- 写入至受写入保护之页面
- 写入至工厂预设之 Option bytes.
- 在 PEKEY、PRGKEY 未解锁下，请求写入闪存之程序代码区域或 Option bytes 区域。

写入保护错误发生后中止写入或擦除操作，并可以产生中断 (当 FLASH\_PECR 缓存器中的 errie= 1 时)。若要复位此旗标，软件需要将其写入 1。

➤ 读取错误旗标 (rderr)

如果软件尝试读取受 PCROP 保护的扇区(sector)，则硬件在 FLASH\_SR 缓存器中设置 rderr 旗标，此时在总线上接收的数据为 0。如果启用了错误中断 (FLASH\_PECR 缓存器中的 errie = 1)，则会产生一个中断。若要重置此旗标，软件需要将其写入 1。

## 6.6 NVM 中断

将 FLASH\_PECR 缓存器中之刻录结尾(End of programming)的中断启用位 (eopie)设置为 1 即会在擦除或刻录执行完成且正确结束时产生中断，此时 FLASH\_SR 缓存器中的程序结束位旗标 (eop)会被设置为 1，软件需写入 1 来将其复位。

在 FLASH\_PECR 缓存器中设置错误中断启用位 (errie)即可在刻录或擦除操作期间发生错误时产生中断。在这种情况下，在 FLASH\_SR 缓存器中会有一个甚至多个错误旗标被设置起来：

- rderr (PCROP 读取保护错误旗标)
- wrperr (写入保护错误旗标)
- optverr (Option bytes 合法性错误旗标)

要复位错误旗标，软件需要将正确之旗标位写入 1。

表格 17 闪存中断要求

Interrupt event	Event flag	Enable control bit
End of operation	eop	eopie[0]
Error	rderr[0] wrperr[0] optverr[0]	errie[0]

### 6.6.1 硬件错误 (Hard Fault)

以下状况会产生一个硬件错误：

- 如果在内存总线上尝试读取任何已设置为读取保护之区域时。
- 如果在缓存器总线上将不正确的值写入 FLASH\_PEKEYR、FLASH\_PRGKEYR 时。

## 6.7 内存接口管理

本节的目的是阐明当请求一个操作而另一个操作正在进行时会发生什么情况：即不同的操作同时进行下之内存接口管理方式。

### 6.7.1 操作优先权 (Priority) 和进程 (Evolution)

有三种类型的操作，并且每种操作都有不同的流程：

#### ➤ 读取(Read)

- 如果没有任何操作正在进行，并且欲读取的地址未受保护，则读取将会直接执行而不会有延迟，并且使用实际配置执行读取。
- 如果欲读取的地址受到保护，则此操作会被滤掉 (读取请求永远不会发送到内存)，并会引发错误。
- 如果欲读取之地址不受保护，但内存接口繁忙而无法执行操作，则操作将被置于保留状态，会尽快被执行。

#### ➤ 写入(Write) / 擦除(erase)

- 如果没有任何操作正在进行，并且欲写入的地址未受保护，则写入或擦除将立即开始；在总线和主控器被阻止操作几个系统时钟的时间之后，内存接口接着将会释放总线和主控器的操作。
- 如果地址受到保护，则过滤掉此写入或擦除操作 (写入或擦除之请求将永远不会发送到内存)，并会引发错误。
- 如果地址未受保护，但未满足一个或多个条件，则操作将中止 (中止需要更多时间执行，因为 NVM 需要返回到默认配置)，并会引发错误。
- 如果欲写入或擦除的地址未受保护且所有规则都符合，但内存接口处于繁忙状态，则操作将被置于保留状态，会尽快被执行。

### 6.7.2 操作之程序 (Sequence of Operations)

#### ➤ 在写入进行时读取数据

如果主服务器在写入操作进行时请求读取数据 (请参阅“数据读取与预读”部分)，则有三种不同的情况：

- 如果读取地址于受保护区域中，则触发 **rderr** 旗标并继续进行目前之写入操作。
- 读取将置于保留状态，并在写入操作完成后执行读取。需要强调的是，在读取等待执行的所有时间中，主控器将被阻止运作，在写入和读取操作完成之前，无法执行任何其它操作。

#### ➤ 在写入进行时提取程序代码

如果主控器在写入时获取指令，则情况类似于读取数据 (请参阅“写入进行时读取数据”的前两个条件)。

#### ➤ 在另一个写入进行时要求写入操作

如果主控器请求写入操作，而另一个写入操作正在进行，则有以下不同的情况：

- 如果新写入地址位于受保护区域中，则触发 **wrperr** 旗标，此时继续进行当前的写入并删除新写入之要求。
- 如果新的写入操作不在受保护区域中，则新的写入将置于保留状态，并于当下进行的写入操作完成后执行。必须强调，请求新写入时，主控器将被阻止运作，直到当下进行之写入操作完成，且新写入已完成内部地址和数据之储存为止。

- 当大量擦除正在进行时，则会禁止新的写入请求：在大量擦除的所有步骤中，资料不会在内部存储，并且新数据可以更改存储为保护的，从而添加不需要的保护。
- 此时可以更改配置以准备新的写入操作。

### 6.7.3 读取时更改等待状态数 (Wait-States)

要更改 Wait-State 的数量，必须写入 FLASH\_ACR 寄存器，而寄存器的读/写与内存读/写使用不同的接口。读取内存接口时无法更改 Wait-State 的数量，如果要求传送到寄存器接口，则无法停止内存接口。

因此，当主控器正在读取内存，而另一个主控器更改 Wait-State 数量时，寄存器接口将被锁定，直到更改生效（直到读取停止）。要停止进行更改 Wait State 数量之主控器，请务必重新读取 FLASH\_ACR 寄存器的内容，因无法知道更改 Wait-State 数所需的时钟周期数，这取决于客户代码。



## 6.8 可选字节 (Option bytes)

在 NVM 上，保留了一个区域以存储一组用于配置产品的 Option Bytes。某些 Option Bytes 在工厂中写入，而其它 Option Bytes 则可以由终端用户来进行配置。

由终端用户管理的配置存储于 Option bytes 区域中，必须留意开机(boot)程序之执行，此开机程序发生在开机复位后，Option bytes 会在开机(boot)期间自动加载，用于设置 flash\_optr 和 FLASH\_WRPROT 缓存器的内容。

### 6.8.1 Option bytes 描述

许多重要的规划设定是放在可选字节 (Option bytes)的区域，藉由应用接口库 (API)让用户读取与修改，详细请参考表格 18.

表格 18 API 菜单

API Functions	Description	Example Code
FLASH_OB_LevelUpdate	Update the configuration of RDP (Read Out Protection) level in the Option Bytes area.	FLASH_OB_LEVEL
FLASH_OB_PcropUpdate	Update the configuration of PcROP (Read Out Protection) in the Option Bytes area.	FLASH_OB_WRITE_PROTECTION & FLASH_OB_READ_PROTECTION
FLASH_OB_EepromWrite	Write data to the EEPROM data in the Option Bytes area (max 512 bytes).	FLASH_OB_EEPROM
FLASH_OB_EepromRead	Read data to the EEPROM data in the Option Bytes area (max 512 bytes).	FLASH_OB_EEPROM

终端使用者在使用它们之前必须在 IDE 中选择软件套件“FLASHEXT”。该软件组件的类别为 Standard-Peripheral-Driver。它包含两个文件：“WT32L064/032\_flashext.h”和“WT32L064/032\_flashext.lib”，可以在 CMSIS PACK 安装路径下找到它们。

有关如何使用这些功能的信息，请参阅 CMSIS PACK 中的相关示例代码。

### 6.8.2 保护旗标加载不匹配

当 Option bytes 加载期间发生不匹配时，内存接口会将默认值设置到缓存器中。在 Option bytes 区域中，有三种保护信息：

➤ rdprot[7:0]

此配置设置保护等级(Level)，如下一节所述，更改此级别将更改访问 NVM 和产品的可能性。此默认值为 Level 1，虽可以从 Level 1 返回到 Level 0，但闪存程序内存的所有内容都将被删除 (因需进行大量擦除 mass erase)。

➤ wprmod

此旗标独立于 rdprot[7:0]，如果闪存程序代码区域受到保护，则不进行读或写。当此旗标为 1 (读取保护)时，复位它的唯一方法是请求大量擦除(mass erase)。此默认值为 1 (读取保护)，当此位上电加载发生不匹配时，同时还会将 wrprot[63:0] 设置为默认值。

➤ wrprot[63:0]

此配置用来设置闪存程序代码区域的哪些页面受防读或写保护。如果已禁用读取保护 (wprmod = 0)，则必须在正确对应之位中设置 1 以特定保护扇区(sector)；如果启用了读取保护 (wprmod = 1)，则 0 必须位于正确对应的位中才能保护时定扇区(sector)。如果在开机(boot)期间 wprmod 出现加载不匹配的状况，则此配置将被加载为 0，以便保护闪存程序代码区域的所有扇区(sector)不被读取。如果 wprmod 已正确读取，但 wrprot[63:0]发生不匹配的状况，则当 wprmod = 1 时，则此缓存器将加载为 0，如果此时 wprmod = 0 时，缓存器将加载为 1。

因此，保护不匹配会对程序代码的正常执行（如果它位于闪存程序代码区域中）产生严重影响：当存在读取保护时，只能进行提取。在闪存程序代码区域中，某些值在代码执行期间作为数据（例如常数）读取；保护所有扇区(secto)不被读取会阻止从闪存程序代码区域执行应用程序代码。

### 6.8.3 模拟的 EEPROM

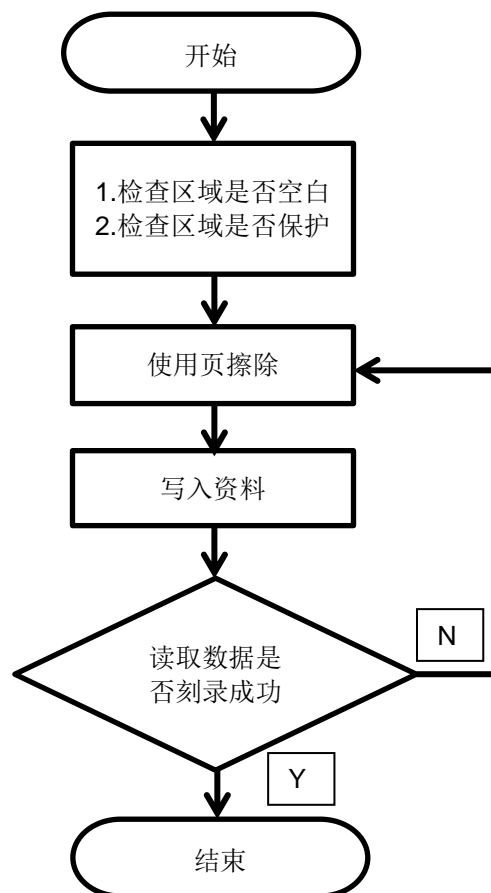
#### 6.8.3.1 描述

WT32L064/032 可以使用闪存 Flash 空间来模拟 E<sup>2</sup>PROM。

@ VDD33=1.8V~3.6V (Note: 1.65V~1.8V Flash 仅能读)

操作流程如下：

1. 选定空白 Flash 无程序占用的区域
2. 确认该区域无写保护或防读保护（参阅 6.5 章节说明）
3. 执行 Page Erase 清除该区域
4. 使用 Byte、Half-Word 或 Word 数据长度写入
5. 可使用 Byte、Half-Word 或 Word 数据长度执行读取与校验
6. 可透过外围如 UART、I<sup>2</sup>C 搭配收发仿真 EEPROM 数据



## 7 DMA

### 7.1 概要

- 符合 AMBA v2.0
  - 用于 DMA 控制器规划的 AHB 从机接口
  - 用于数据传输的 AHB 主机接口
  - 传输类型 – 单模式
  - 32-bits (word), 16-bits (half-word), 8-bits (byte) 等宽字符转换
  
- 7 个可规划的 DMA 通道
  - 支持内存到外围传输
  - 支持外围到内存的传输
  - 支持外围到外围传输
  
- 支持的外围硬件 (注意: 外围硬件的系统时钟 APB0 或 APB1 仅支持 AHB/1, AHB/2, 不支援 AHB/4, AHB/8, AHB/16)
  - 15 组的请求/确认硬件握手
  - UARTs (Tx/Rx)
  - Timers
  - IICs (Tx/Rx)
  - ADC
  - SPI (Tx/Rx)
  - USB
  - I2S (Tx/Rx)
  
- 仲裁 线路
  - Round-robin 仲裁.
  - 可规划 4 级优先级
  
- 循环模式

## 7.2 功能描述

直接内存访问控制器(DMA)增强了系统性能，减少了处理器中断产生。有 7 个可规划信道，用于内存到外围、外围到内存以及外围到外围传输。每个信道都连接到专用硬件握手信号。

### 7.2.1 方块图

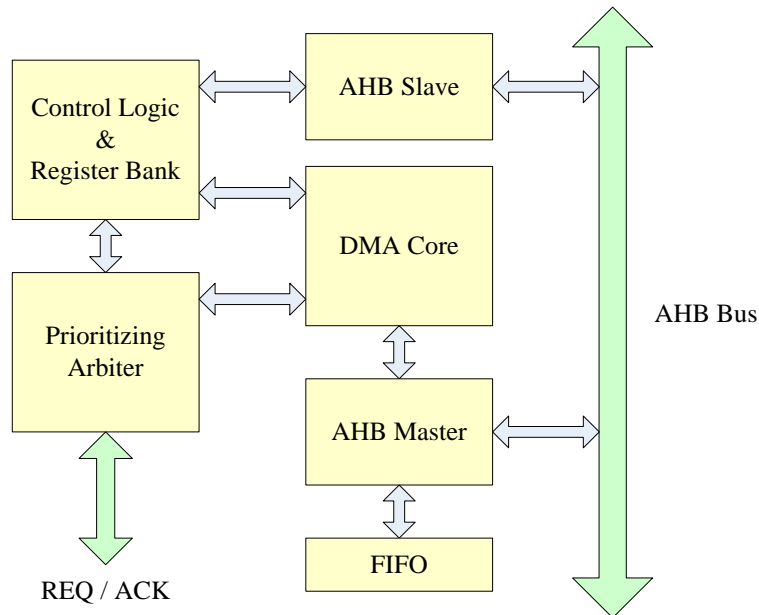


图 26 DMA 方块图

### 7.2.2 AHB 主机接口

系统可利用此 AHB 主机接口传输 AHB 总线上的数据。

### 7.2.3 AHB 从机界面

系统可以规划 DMA 控制器或通过此 AHB 从机接口访问 AHB 总线上的设备。

### 7.2.4 FIFO 缓冲区

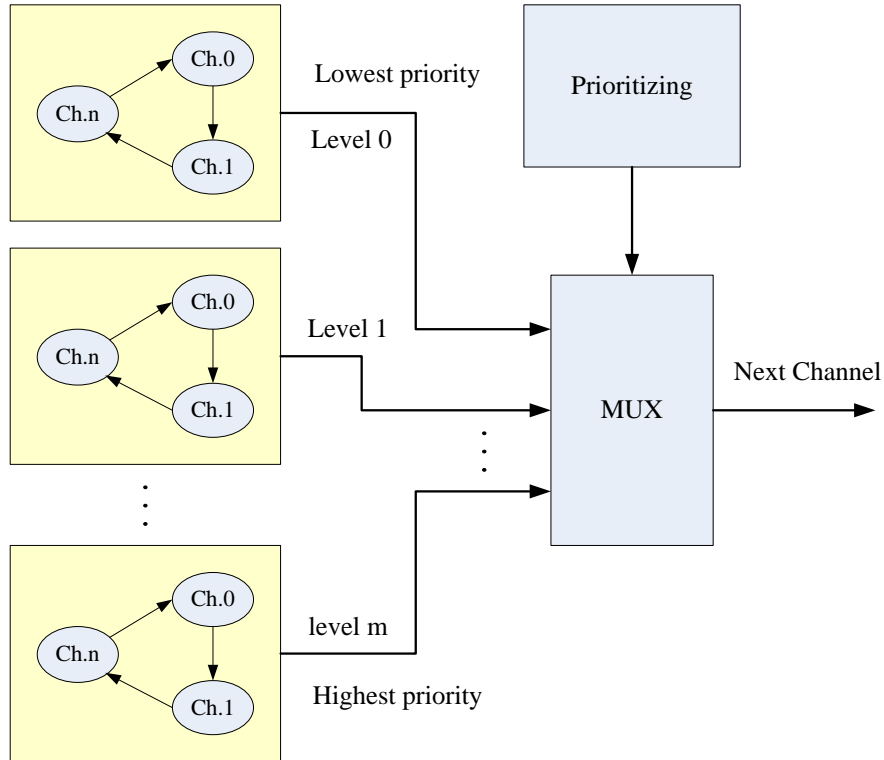
FIFO 缓冲区在源和目标之间提供数据传输缓冲区。

### 7.2.5 DMA 核心

DMA 数据传输引擎。

### 7.2.6 优先权仲裁

处理硬件握手信号以启动 DMA 传输并规划最多 7 个通道。他们可以将循环仲裁方案分为 4 个优先级。



### 7.2.7 控制逻辑与缓存器组

AHB 从机接口缓存器集合，并为 DMA 传输产生一些控制逻辑。

### 7.3 DMA Register Table

#### 7.3.1 中断状态/清除、通道 Busy 的全局设置

Base Address: 0x0030\_0000h

Index	Default	R/W	Bit	Name	Description
<b>DMA_BUSY: DMA busy</b>					
04			31:7		保留
	0h	RO	6	dma_busy6	1: 通道 6 忙碌 0: 通道 6 可用
	0h	RO	5	dma_busy5	1: 通道 5 忙碌 0: 通道 5 可用
	0h	RO	4	dma_busy4	1: 通道 4 忙碌 0: 通道 4 可用
	0h	RO	3	dma_busy3	1: 通道 3 忙碌 0: 通道 3 可用
	0h	RO	2	dma_busy2	1: 通道 2 忙碌 0: 通道 2 可用
	0h	RO	1	dma_busy1	1: 通道 1 忙碌 0: 通道 1 可用
	0h	RO	0	dma_busy0	1: 通道 0 忙碌 0: 通道 0 可用
<b>DMA_ISR: Interrupt status</b>					
08			31:28		保留
	0	RO	27	err_int6	由硬件设置，由软件清除。 0: 无传输错误事件。 1: 传输错误。
	0	RO	26	half_int6	0: 无 half_int 事件于通道 6 1: 有 half_int 事件发生于通道 6
	0	RO	25	total_int6	0: 无 total_int 事件于通道 6 1: 有 total_int 事件发生于通道 6
	0	RO	24	dma_int6	0: 无 half_int 或 total_int 事件于通道 6 1: 有 err_int, half_int 或 total_int 事件发生于通道 6
		RO	23	err_int5	由硬件设置，由软件清除。 0: 无传输错误事件。 1: 传输错误。
	0	RO	22	half_int5	0: 无 half_int 事件于通道 5 1: 有 half_int 事件发生于通道 5
	0	RO	21	total_int5	0: 无 total_int 事件于通道 5 1: 有 total_int 事件发生于通道 5
	0	RO	20	dma_int5	0: 无 half_int 或 total_int 事件于通道 5 1: 有 err_int, half_int 或 total_int 事件发生于通道 5
		RO	19	err_int4	由硬件设置，由软件清除。 0: 无传输错误事件。 1: 传输错误。
	0	RO	18	half_int4	0: 无 half_int 事件于通道 4 1: 有 half_int 事件发生于通道 4
	0	RO	17	total_int4	0: 无 total_int 事件于通道 4 1: 有 total_int 事件发生于通道 4

Index	Default	R/W	Bit	Name	Description
	0	RO	16	dma_int4	0: 无 half_int 或 total_int 事件于通道 4 1: 有 err_int, half_int 或 total_int 事件发生于通道 4
		RO	15	err_int3	由硬件设置, 由软件清除。 0: 无传输错误事件。 1: 传输错误。
	0	RO	14	half_int3	0: 无 half_int 事件于通道 3 1: 有 half_int 事件发生于通道 3
	0	RO	13	total_int3	0: 无 total_int 事件于通道 3 1: 有 total_int 事件发生于通道 3
	0	RO	12	dma_int3	0: 无 half_int 或 total_int 事件于通道 3 1: 有 err_int, half_int 或 total_int 事件发生于通道 3
		RO	11	err_int2	由硬件设置, 由软件清除。 0: 无传输错误事件。 1: 传输错误。
	0	RO	10	half_int2	0: 无 half_int 事件于通道 2 1: 有 half_int 事件发生于通道 2
	0	RO	9	total_int2	0: 无 total_int 事件于通道 2 1: 有 total_int 事件发生于通道 2
	0	RO	8	dma_int2	0: 无 half_int 或 total_int 事件于通道 2 1: 有 err_int, half_int 或 total_int 事件发生于通道 2
		RO	7	err_int1	由硬件设置, 由软件清除。 0: 无传输错误事件。 1: 传输错误。
	0	RO	6	half_int1	0: 无 half_int 事件于通道 1 1: 有 half_int 事件发生于通道 1
	0	RO	5	total_int1	0: 无 total_int 事件于通道 1 1: 有 total_int 事件发生于通道 1
	0	RO	4	dma_int1	0: 无 half_int 或 total_int 事件于通道 1 1: 有 err_int, half_int 或 total_int 事件发生于通道 1
		RO	3	err_int0	由硬件设置, 由软件清除。 0: 无传输错误事件。 1: 传输错误。
	0	RO	2	half_int0	0: 无 half_int 事件于通道 0 1: 有 half_int 事件发生于通道 0
	0	RO	1	total_int0	0: 无 total_int 事件于通道 0 1: 有 total_int 事件发生于通道 0
	0	RO	0	dma_int0	0: 无 half_int 或 total_int 事件于通道 0 1: 有 err_int, half_int 或 total_int 事件发生于通道 0
<b>DMA_CLR_F: Clear flags</b>					
			31:28		保留
		WO	27	Clr_err_int6	0: 无效 1: 清除 err_int6 旗标
0C	0	WO	26	clr_half_int6	0: 无效 1: 清除 half_int6 旗标
	0	WO	25	clr_total_int6	0: 无效 1: 清除 total_int6 旗标
	0	WO	24	clr_int6	0: 无效 1: 清除s dma_int6, err_int6, total_int6 and half_int6 旗标s
		WO	23	Clr_err_int5	0: 无效 1: 清除 err_int5 旗标

Index	Default	R/W	Bit	Name	Description
	0	WO	22	clr_half_int5	0: 无效 1: 清除 half_int5 旗标
	0	WO	21	clr_total_int5	0: 无效 1: 清除 total_int5 旗标
	0	WO	20	clr_int5	0: 无效 1: 清除s dma_int5, err_int5, total_int5 and half_int5 旗标s
		WO	19	Clr_err_int4	0: 无效 1: 清除 err_int4 旗标
	0	WO	18	clr_half_int4	0: 无效 1: 清除 half_int4 旗标
	0	WO	17	clr_total_int4	0: 无效 1: 清除 total_int4 旗标
	0	WO	16	clr_int4	0: 无效 1: 清除s dma_int4, err_int4, total_int4 and half_int4 旗标s
		WO	15	Clr_err_int3	0: 无效 1: 清除 err_int3 旗标
	0	WO	14	clr_half_int3	0: 无效 1: 清除 half_int3 旗标
	0	WO	13	clr_total_int3	0: 无效 1: 清除 total_int3 旗标
	0	WO	12	clr_int3	0: 无效 1: 清除s dma_int3, err_int3, total_int3 and half_int3 旗标s
		WO	11	Clr_err_int2	0: 无效 1: 清除 err_int2 旗标
	0	WO	10	clr_half_int2	0: 无效 1: 清除 half_int2 旗标
	0	WO	9	clr_total_int2	0: 无效 1: 清除 total_int2 旗标
	0	WO	8	clr_int2	0: 无效 1: 清除s dma_int2, err_int2, total_int2 and half_int2 旗标s
		WO	7	Clr_err_int1	0: 无效 1: 清除 err_int1 旗标
	0	WO	6	clr_half_int1	0: 无效 1: 清除 half_int1 旗标
	0	WO	5	clr_total_int1	0: 无效 1: 清除 total_int1 旗标
	0	WO	4	clr_int1	0: 无效 1: 清除s dma_int1, err_int1, total_int1 and half_int1 旗标s
		WO	3	Clr_err_int0	0: 无效 1: 清除 err_int0 旗标
	0	WO	2	clr_half_int0	0: 无效 1: 清除 half_int0 旗标
	0	WO	1	clr_total_int0	0: 无效 1: 清除 total_int0 旗标
	0	WO	0	clr_int0	0: 无效 1: 清除s dma_int0, err_int0, total_int0 and half_int0 旗标s



### 7.3.2 DMA 信道 x 来源之 AddrESS 缓存器

X = 0 ~ 6, where x is channel number

Channel X Address index:  $0x10 + 0x10 * (\text{channel number } X)$

Index	Default	R/W	Bit	Name	Description
<b>DMA_SR_ADDR0: Source address of channel 0</b>					
10	0	R/W	31:0	dma_saddr0	通道 0 source address
<b>DMA_SR_ADDR1: Source address of channel 1</b>					
20	0	R/W	31:0	dma_saddr1	通道 1 source address
<b>DMA_SR_ADDR2: Source address of channel 2</b>					
30	0	R/W	31:0	dma_saddr2	通道 2 source address
<b>DMA_SR_ADDR3: Source address of channel 3</b>					
40	0	R/W	31:0	dma_saddr3	通道 3 source address
<b>DMA_SR_ADDR4: Source address of channel 4</b>					
50	0	R/W	31:0	dma_saddr4	通道 4 source address
<b>DMA_SR_ADDR5: Source address of channel 5</b>					
60	0	R/W	31:0	dma_saddr5	通道 5 source address
<b>DMA_SR_ADDR6: Source address of channel 6</b>					
70	0	R/W	31:0	dma_saddr6	通道 6 source address

### 7.3.3 DMA 信道 x 目的地之地址缓存器

X = 0 ~ 6, where x is channel number

Channel X Address index:  $0x14 + 0x10 * (\text{channel number } X)$

Index	Default	R/W	Bit	Name	Description
<b>DMA_DT_ADDR0: Destination address of channel 0</b>					
14	0	R/W	31:0	dma_daddr0	通道 0 destination address
<b>DMA_DT_ADDR1: Destination address of channel 1</b>					
24	0	R/W	31:0	dma_daddr1	通道 1 destination address
<b>DMA_DT_ADDR2: Destination address of channel 2</b>					
34	0	R/W	31:0	dma_daddr2	通道 2 destination address
<b>DMA_DT_ADDR3: Destination address of channel 3</b>					
44	0	R/W	31:0	dma_daddr3	通道 3 destination address
<b>DMA_DT_ADDR4: Destination address of channel 4</b>					
54	0	R/W	31:0	dma_daddr4	通道 4 destination address
<b>DMA_DT_ADDR5: Destination address of channel 5</b>					
64	0	R/W	31:0	dma_daddr5	通道 5 destination address
<b>DMA_DT_ADDR6: Destination address of channel 6</b>					
74	0	R/W	31:0	dma_daddr6	通道 6 destination address

**7.3.4 DMA 信道 x 数据数量之缓存器**

X = 0 ~ 6, where x is channel number

Channel X Address index: 0x18 + 0x10 \* (channel number X)

Index	Default	R/W	Bit	Name	Description
<b>DMA_LENGTH0: Total data length of channel 0</b>					
18			31:9		保留
	0	R/W	8:0	dma_lenth0	通道 0 总大小传输长度: 1 ~ 511 0: DMA transfer stop
<b>DMA_LENGTH1: Total data length of channel 1</b>					
28			31:9		保留
	0	R/W	8:0	dma_lenth1	通道 1 总大小传输长度: 1 ~ 511 0: DMA transfer stop
<b>DMA_LENGTH2: Total data length of channel 2</b>					
38			31:9		保留
	0	R/W	8:0	dma_lenth2	通道 2 总大小传输长度: 1 ~ 511 0: DMA transfer stop
<b>DMA_LENGTH3: Total data length of channel 3</b>					
48			31:9		保留
	0	R/W	8:0	dma_lenth3	通道 3 总大小传输长度: 1 ~ 511 0: DMA transfer stop
<b>DMA_LENGTH4: Total data length of channel 4</b>					
58			31:9		保留
	0	R/W	8:0	dma_lenth4	通道 4 总大小传输长度: 1 ~ 511 0: DMA transfer stop
<b>DMA_LENGTH5: Total data length of channel 5</b>					
68			31:9		保留
	0	R/W	8:0	dma_lenth5	通道 5 总大小传输长度: 1 ~ 511 0: DMA transfer stop
<b>DMA_LENGTH6: Total data length of channel 6</b>					
78			31:9		保留
	0	R/W	8:0	dma_lenth6	通道 6 总大小传输长度: 1 ~ 511 0: DMA transfer stop

**7.3.5 DMA 信道 x 设定缓存器**

X = 0 ~ 6 where x is channel number

Channel X Address index: 0x1C + 0x10 \* (channel number X)

Index	Default	R/W	Bit	Name	Description
<b>DMA_CFG0: Configuration of channel 0</b>					
1C			31:16		保留
	0	RW	15:14	Src_width0	信道 0 来源发送资料宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
	0	RW	13:12	Dest_width0	信道 0 目的发送数据宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
			11		保留

Index	Default	R/W	Bit	Name	Description
	0	R/W	10	circ_mode0	信道 0 循环缓冲模式 1: 循环缓冲模式 启动 0: 循环缓冲模式 禁能
	0	R/W	9:8	pri_ch0	信道 0 优先级等级: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (Default)
	0	R/W	7:6	src_adr0_ctl	信道 0 来源地址控制: 3: 保留 2: Fixed source address 1: 保留 0: 递增来源地址 (Default)
	0	R/W	5:4	dst_adr0_ctl	信道 0 目的地址控制 3: 保留 2: 固定目标地址(destination address) 1: 保留 0: 增加目标地址 (默认值)
	0	R/W	3	en_err_int0	通道 0 错误中断启动 1: 错误中断启动 0: 错误中断关闭
	0	R/W	2	en_half_int0	通道 0 发送一半中断启动 1: 发送一半中断启动 0: 发送一半中断关闭
	0	R/W	1	en_total_int0	通道 0 发送全部中断启动: 1: 发送全部中断启动 0: 发送全部中断关闭
	0	R/W	0	dma_chen0	通道 0, 写1 致能.
<b>DMA_CFG1: Configuration of channel 1</b>					
2C			31:16		保留
	0	R/W	15:14	Src_width1	信道 1 来源发送资料宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
	0	R/W	13:12	Dest_width1	信道 1 目的发送数据宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
			11		保留
	0	R/W	10	circ_mode1	信道 1 循环缓冲模式 1: 循环缓冲模式启动 0: 循环缓冲模式禁能
	0	R/W	9:8	pri_ch1	信道 1 优先级等级: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (默认值)

Index	Default	R/W	Bit	Name	Description
	0	R/W	7:6	src_adr1_ctl	信道 1 来源地址控制: 3: 保留 2: Fixed source address 1: 保留 0: 递增来源地址 (默认值)
	0	R/W	5:4	dst_adr1_ctl	信道 1 目的地址控制: 3: 保留 2: Fixed destination address 1: 保留 0: 递增目标地址 (默认值)
	0	R/W	3	en_err_int1	通道 1 错误中断启动 1: 错误中断启动 0: 错误中断关闭
	0	R/W	2	en_half_int1	通道 1 发送一半中断启动 1: 发送一半中断启动 0: 发送一半中断关闭
	0	R/W	1	en_total_int1	通道 1 发送全部中断启动: 1: 发送全部中断启动 0: 发送全部中断关闭
	0	R/W	0	dma_chen1	通道 1 写“1” 致能
<b>DMA_CFG2: Configuration of channel 2</b>					
3C			31:16		保留
	0	RW	15:14	Src_width2	信道 2 来源发送资料宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
	0	RW	13:12	Dest_width2	信道 2 目的发送数据宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
			11		保留
	0	RW	10	circ_mode2	信道 2 循环缓冲模式 1: 循环缓冲模式 启动 0: 循环缓冲模式 禁能
	0	R/W	9:8	pri_ch2	信道 2 优先权等级: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (默认值)
	0	R/W	7:6	src_adr2_ctl	信道 2 来源地址控制: 3: 保留 2: Fixed source address 1: 保留 0: 递增来源地址 (默认值)
	0	R/W	5:4	dst_adr2_ctl	信道 2 目的地址控制: 3: 保留 2: Fixed destination address 1: 保留 0: 递增目标地址 (默认值)

Index	Default	R/W	Bit	Name	Description
	0	R/W	3	en_err_int2	通道 2 错误中断启动 1: 错误中断启动 0: 错误中断关闭
	0	R/W	2	en_half_int2	通道 2 发送一半中断启动 1: 发送一半中断启动 0: 发送一半中断关闭
	0	R/W	1	en_total_int2	通道 2 发送全部中断启动: 1: 发送全部中断启动 0: 发送全部中断关闭
	0	R/W	0	dma_chen2	通道 2 写“1” 致能
<b>DMA_CFG3: Configuration of channel 3</b>					
			31:16		保留
	0	RW	15:14	Src_width3	信道 3 来源发送资料宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
	0	RW	13:12	Dest_width3	信道 3 目的发送数据宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
			11		保留
	0	RW	10	circ_mode3	信道 3 循环缓冲模式 1: 循环缓冲模式启动 0: 循环缓冲模式禁能
	0	R/W	9:8	pri_ch3	信道 3 优先权等级: 3: Highest priority. 2: High priority. 1: Medium priority. 0: Low priority (默认值).
4C	0	R/W	7:6	src_adr3_ctl	信道 3 来源地址控制: 3: 保留 2: Fixed source address 1: 保留 0: 递增来源地址 (默认值)
	0	R/W	5:4	dst_adr3_ctl	信道 3 目的地址控制: 3: 保留 2: Fixed destination address 1: 保留 0: 递增目标地址 (默认值)
	0	R/W	3	en_err_int3	通道 3 错误中断启动 1: 错误中断启动 0: 错误中断关闭
	0	R/W	2	en_half_int3	通道 3 发送一半中断启动 1: 发送一半中断启动 0: 发送一半中断关闭
	0	R/W	1	en_total_int3	通道 3 发送全部中断启动: 1: 发送全部中断启动 0: 发送全部中断关闭
	0	R/W	0	dma_chen3	通道 3 写“1” 致能
<b>DMA_CFG4: Configuration of channel 4</b>					

Index	Default	R/W	Bit	Name	Description
5C			31:16		保留
	0	RW	15:14	Src_width4	信道 4 来源发送资料宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
	0	RW	13:12	Dest_width4	信道 4 目的发送数据宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
			11		保留
	0	RW	10	circ_mode4	信道 4 循环缓冲模式 1: 循环缓冲模式启动 0: 循环缓冲模式禁能
	0	R/W	9:8	pri_ch4	信道 4 优先权等级: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (默认值)
	0	R/W	7:6	src_adr4_ctl	信道 4 来源地址控制: 3: 保留 2: Fixed source address 1: 保留 0: 递增来源地址 (默认值)
	0	R/W	5:4	dst_adr4_ctl	信道 4 目的地址控制: 3: 保留 2: Fixed destination address 1: 保留 0: 递增目标地址 (默认值)
	0	R/W	3	en_err_int4	通道 4 错误中断启动 1: 错误中断启动 0: 错误中断关闭
	0	R/W	2	en_half_int4	通道 4 发送一半中断启动 1: 发送一半中断启动 0: 发送一半中断关闭
	0	R/W	1	en_total_int4	通道 4 发送全部中断启动: 1: 发送全部中断启动 0: 发送全部中断关闭
0	R/W	0	dma_chen4	通道 4 写"1" 致能	
<b>DMA_CFG5: Configuration of channel 5</b>					
6C			31:16		保留
	0	RW	15:14	Src_width5	信道 5 来源发送资料宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
	0	RW	13:12	Dest_width5	信道 5 目的发送数据宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
			11		保留

Index	Default	R/W	Bit	Name	Description
	0	R/W	10	circ_mode5	信道 5 循环缓冲模式 1: 循环缓冲模式启动 0: 循环缓冲模式禁能
	0	R/W	9:8	pri_ch5	信道 5 优先级等级: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (默认值)
	0	R/W	7:6	src_adr5_ctl	信道 5 来源地址控制: 3: 保留 2: Fixed source address 1: 保留 0: 递增来源地址 (默认值)
	0	R/W	5:4	dst_adr5_ctl	信道 5 目的地址控制: 3: 保留 2: Fixed destination address 1: 保留 0: 递增目标地址 (默认值)
	0	R/W	3	en_err_int5	通道 5 错误中断启动 1: 错误中断启动 0: 错误中断关闭
	0	R/W	2	en_half_int5	通道 5 发送一半中断启动 1: 发送一半中断启动 0: 发送一半中断关闭
	0	R/W	1	en_total_int5	通道 5 发送全部中断启动: 1: 发送全部中断启动 0: 发送全部中断关闭
	0	R/W	0	dma_chen5	通道 5 写“1” 致能
<b>DMA_CFG6: Configuration of channel 6</b>					
7C			31:16		保留
	0	R/W	15:14	Src_width6	信道 6 来源发送资料宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
	0	R/W	13:12	Dest_width6	信道 6 目的发送数据宽度 00: 8bits 01: 16bits 10: 32bits 11: 保留
			11		保留
	0	R/W	10	circ_mode6	信道 6 循环缓冲模式 1: 循环缓冲模式启动 0: 循环缓冲模式禁能
	0	R/W	9:8	pri_ch6	信道 6 优先级等级: 3: Highest priority 2: High priority 1: Medium priority 0: Low priority (默认值)

Index	Default	R/W	Bit	Name	Description
0		R/W	7:6	src_adr6_ctl	信道 6 来源地址控制: 3: 保留 2: Fixed source address 1: 保留 0: 递增来源地址 (默认值)
0		R/W	5:4	dst_adr6_ctl	信道 6 目的地址控制: 3: 保留 2: Fixed destination address 1: 保留 0: 递增目标地址 (默认值)
0		R/W	3	en_err_int6	通道 6 错误中断启动 1: 错误中断启动 0: 错误中断关闭
0		R/W	2	en_half_int6	通道 6 发送一半中断启动 1: 发送一半中断启动 0: 发送一半中断关闭
0		R/W	1	en_total_int6	通道 6 发送全部中断启动: 1: 发送全部中断启动 0: 发送全部中断关闭
0		R/W	0	dma_chen6	通道 6 写“1” 致能



## 8 GPIO

### 8.1 主要概述

每个通用 I/O 埠(PA、PB、PC、PD)有三个配置寄存器 (gpio\_px\_mode、gpio\_px\_ot 和 gpio\_px\_pupd)、两个数据寄存器(gpio\_px\_out 和 gpio\_px\_in)和设置/复位/切换寄存器 (gpio\_px\_br、gpio\_px\_bs 和 gpio\_px\_bt)。此外, 所有 GPIO 都有复用 (alternate)功能选择寄存器 (gpio\_px\_af)。

- AHB I/F, 支持单次大量的 WORD 存取
- 输出状态: 推挽式(push-pull)或开汲极(open-drain) + 上拉(pull-up)/下拉(pull-down)
- 输出数据寄存器 (gpio\_px\_out)或外围设备 (备用功能输出)输出数据
- 输入状态: 浮动、上拉或下拉、模拟
- 输入数据寄存器 (gpio\_px\_in)或外围设备 (备用函数输入)的数据
- 模拟功能
- 复用(alternate)功能选择寄存器
- 高度灵活的引脚, 允许使用 I/O 引脚作为 GPIO 或作为多种外围功能之一

### 8.2 方块图

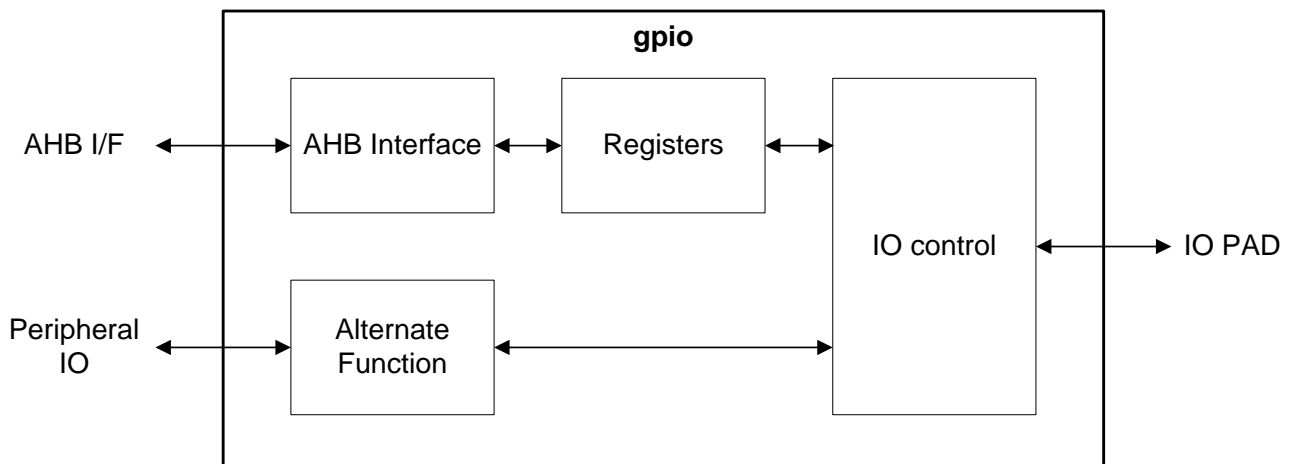


图 27 GPIO 方块图

### 8.3 寄存器描述

表格 19 GPIO 控制寄存器

Index	Bit	R/W	Default	Name	Description
<b>GPIOA_MODE: Pad mode register</b>					
000h	31:0	R/W	ebff_0fffh	gpio_pa_mode	PA 0~15 的模式寄存器。 这些位由软件编写以配置 I/O 模式。 [2y+1:2y]: X 端口的模式设定。X: 哪一个 I/O 埠, 范围 A~D 埠。Y: 哪一根 I/O 范围 0~31 2'b00: 输入模式

Index	Bit	R/W	Default	Name	Description
					2'b01: 普通输出模式 2'b10: 复用功能模式 2'b11: 模拟功能模式 注意: 预设 PA[13], PA[14] 是复用功能. PA[6], PA[7] 是 输入模式.
<b>Reserved</b>					
004h	31:0	-	-	-	保留
<b>GPIOA_PUPD: Pad pull-up/pull-down register</b>					
008h	31:0	R/W	2400_5000 h	gpio_pa_pupd	PA 上拉或下拉电阻。 这些位由软件编写，用于规划 I/O 上拉或下拉 [2y+1:2y]: 上拉或下拉于埠 X. 2'b00: 无上拉，下拉 2'b01: 上拉 2'b10: 下拉 2'b11: 保留 注意:预设情况下，PA[13] 是上拉;PA[14] 是下拉。 PA[6] 和 PA[7] 是上拉。
<b>Reserved</b>					
00ch	31:0	-	-	-	保留
<b>GPIOA_DO: Pad output data</b>					
010h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pa_out	PA 输出数据
<b>GPIOA_DI: Pad input data</b>					
014h	31:16	-	-	-	保留
	15:0	R	-	gpio_pa_in	PA 输入数据
<b>GPIOA_OTYPE: Pad output type</b>					
018h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pa_ot	PA 输出类型。 0: 推挽式 (Push-pull). 1: 开极极 (Open-drain) 注意:在开极极类型中，如果将 gpio_pa_pupd 致能 为上拉，则逻辑 1 由内部上拉电阻驱动引脚;否则， 逻辑 1 应由外部上拉电阻驱动。
<b>GPIOA_DS: Pad driving strength</b>					
01ch	15:0	W	-	gpio_pa_ds	PA 驱动能力 0: 低 (默认值). 1: 高
<b>GPIOA_BT_RST: Pad bit reset</b>					
020h	31:16	-	-	-	保留
	15:0	W	-	gpio_pa_br	PA 位重置 0: 无效 1: 将 gpio_pa_out 对应位重置为 “0”
<b>GPIOA_BT_SET: Pad bit set</b>					
024h	31:16	-	-	-	保留
	15:0	W	-	gpio_pa_bs	PA 位设置 0: 无效 1: 将 gpio_pa_out 对应位设置为 “1”
<b>GPIOA_BT_TGLE: Pad bit toggle</b>					
028h	31:16	-	-	-	保留

Index	Bit	R/W	Default	Name	Description
	15:0	W	-	gpio_pa_bt	PA 位变化 (toggle) 0: 无效 1: 切换相对反应的位变化 gpio_pa_out “0->1” 或 “1->0”
<b>GPIOA_AF0: Pad alt function</b>					
040h	31:0	R/W	0	gpio_pa_af0	PA Port 0~7 的复用功能 这些位由软件编写，用于规划复用功能 I/Os 0: 复用功能 0 (AF0) 1: 复用功能 1 (AF1) 2: 复用功能 2 (AF2) 3: 复用功能 3 (AF3) 4: 复用功能 4 (AF4) 5: 复用功能 5 (AF5) 6~15: 保留
<b>GPIOA_AF1: Pad alt function</b>					
044h	31:0	R/W	0	gpio_pa_af1	PA 埠 8~15 的复用功能。 这些位由软件编写，用于规划复用功能 I/O。 [4y+3:4y]: 端口 x 的模式规划。 0: 复用功能 0 (AF0) 1: 复用功能 1 (AF1) 2: 复用功能 2 (AF2) 3: 复用功能 3 (AF3) 4: 复用功能 4 (AF4) 5: 复用功能 5 (AF5) 6~15: 保留
<b>Reserved</b>					
048h	31:0	-	-	-	保留
<b>Reserved</b>					
04ch	31:0	-	-	-	保留
<b>GPIOA_IE: Pad interrupt enable</b>					
060h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pa_ie	PA 中断功能 0: 禁能 1: 致能
<b>GPIOA_ISS: Pad interrupt sense selection</b>					
064h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pa_iss	PA 中断输入选择。 0: 边缘(Edge)触发。 1: 电位(Level)触发。
<b>GPIOA_BET: Pad interrupt both edges trigger</b>					
068h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pa_bet	PA 两个边缘触发中断。 0: 中断条件由 gpio_pa_trg 控制。 1: 致能上升缘和下降缘触发。 注意: 当电位(Level)触发时，将忽略此缓存器。
<b>GPIOA_TRG: Pad interrupt trigger event select</b>					
06ch	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pa_trg	PA 触发事件选择。 0: 下降缘或低电位触发。

Index	Bit	R/W	Default	Name	Description
					1: 上升缘或高电位触发。
<b>GPIOA_IF: Pad interrupt raw flag</b>					
070h	31:16	-	-	-	保留
	15:0	R/W1c	0	gpio_pa_if	PA 中断原始旗标。 此缓存器指示 GPIO 原始中断的状态。如果启动 gpio_pa_ie 缓存器中的相应位, 则 GPIO 中断将发送到中断控制器。 0: 无中断 1: 有中断发生
<b>GPIOB_MODE: Pad mode register</b>					
100h	31:0	R/W	ffff_ffffh	gpio_pb_mode	PB 端口 0~15 的模式缓存器。 这些位由韧体编写, 用来规划 I/O 模式。 [2y+1:2y]: X 端口的模式设定 2'b00: 输入模式 2'b01: 普通输出模式 2'b10: 复用功能模式 2'b11: 模拟 mode
<b>Reserved</b>					
104h	31:0	-	-	-	保留
<b>GPIOB_PUPD: Pad pull-up/pull-down register</b>					
108h	31:0	R/W	0	gpio_pb_pupd	PB 上拉或下拉电阻 这些位由软件编写, 用于规划 I/O 上拉或下拉 [2y+1:2y]: 上拉或下拉于埠 X。 2'b00: No 上拉, 下拉 2'b01: 上拉 2'b10: 下拉 2'b11: 保留
<b>Reserved</b>					
10ch	31:0	-	-	-	保留
<b>GPIOB_DO: Pad output data</b>					
110h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pb_out	PB 输出数据
<b>GPIOB_DI: Pad input data</b>					
114h	31:16	-	-	-	保留
	15:0	R	-	gpio_pb_in	PB 输入数据
<b>GPIOB_OTYPE: Pad output type</b>					
118h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pb_ot	PB 输出类型 0: 推挽式 (Push-pull) 1: 开汲极 (Open-drain) 注意: 在开汲极类型中, 如果将 gpio_pb_pupd 启用为上拉, 则逻辑 1 由内部上拉电阻驱动引脚; 否则, 逻辑 1 应由外部上拉电阻驱动。
<b>GPIOB_DS: Pad driving strength</b>					
11ch	15:0	W	-	gpio_pb_ds	PB 驱动能力 0: 低 1: 高
<b>GPIOB_BT_RST: Pad bit reset</b>					

Index	Bit	R/W	Default	Name	Description
120h	31:16	-	-	-	保留
	15:0	W	-	gpio_pb_br	PB 位重置 0: 无效。 1: 将 gpio_pb_out 对应位重置为"0"。
<b>GPIOB_BT_SET: Pad bit set</b>					
124h	31:16	-	-	-	保留
	15:0	W	-	gpio_pb_bs	PB 位设置 0: 无效 1: 将 gpio_pb_out 对应位设置为"1"
<b>GPIOB_BT_TGLE: Pad bit toggle</b>					
128h	31:16	-	-	-	保留
	15:0	W	-	gpio_pb_bt	PB 位变化( toggle) 0: 无效 1: 切换相对反应的位变化 gpio_pb_out "0->1" 或"1->0"
<b>GPIOB_AF0: Pad alt function</b>					
140h	31:0	R/W	0	gpio_pb_af0	PB 埠 0~7 的复用功能 这些位由软件编写, 用来规划复用功能 I/Os [4y+3:4y]: X 端口的模式设定. 0: 复用功能 0 (AF0) 1: 复用功能 1 (AF1) 2: 复用功能 2 (AF2) 3: 复用功能 3 (AF3) 4: 复用功能 4 (AF4) 5: 复用功能 5 (AF5) 6~15: 保留
<b>GPIOB_AF1: Pad alt function</b>					
144h	31:0	R/W	0	gpio_pb_af1	PB 埠 8~15 的复用功能。 这些位由软件编写, 用于规划复用功能 I/O。 [4y+3:4y]:端口 x 的模式规划。 0: 复用功能 0 (AF0) 1: 复用功能 1 (AF1) 2: 复用功能 2 (AF2) 3: 复用功能 3 (AF3) 4: 复用功能 4 (AF4) 5: 复用功能 5 (AF5) 6~15: 保留
<b>Reserved</b>					
148h	31:0	-	-	-	保留
<b>Reserved</b>					
14ch	31:0	-	-	-	保留
<b>GPIOB_IE: Pad interrupt enable</b>					
160h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pb_ie	PB 中断功能 0: 禁能 1: 致能
<b>GPIOB_ISS: Pad interrupt sense selection</b>					
164h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pb_iss	PB 中断输入选择。

Index	Bit	R/W	Default	Name	Description
					0: 边缘(Edge)触发。 1: 电位(Level)触发。
<b>GPIOB_BET: Pad interrupt both edges trigger</b>					
168h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pb_bet	PB 两个边缘触发中断。 0: 中断条件由 gpio_pb_trg 控制。 1: 致能上升缘和下降缘触发。 注意: 当电位(Level)触发时, 将忽略此缓存器。
<b>GPIOB_TRG: Pad interrupt trigger event select</b>					
16ch	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pb_trg	PB 触发器事件选择。 0: 下降缘或低电平触发。 1: 上升缘或高电平触发。
<b>GPIOB_IF: Pad interrupt raw flag</b>					
170h	31:16	-	-	-	保留
	15:0	R/W c	0	gpio_pb_if	PB 中断原始旗标。 此缓存器指示 GPIO 原始中断的状态。如果致能 gpio_pb_ie 缓存器中的相应位, 则 GPIO 中断将发送到中断控制器。 0: 无中断。 1: 有中断发生。
<b>GPIOC_MODE: Pad mode register</b>					
200h	31:0	R/W	ffff_ffffh	gpio_pc_mode	PC 端口 0~15 的模式缓存器。 这些位由韧体编写, 用来规划 I/O 模式。 [2y+1:2y]: X 端口的模式设定。 2'b00: 输入模式 2'b01: 普通输出模式 2'b10: 复用功能模式 2'b11: 模拟 mode
<b>Reserved</b>					
204h	31:0	-	-	-	保留
<b>GPIOC_PUPD: Pad pull-up/pull-down register</b>					
208h	31:0	R/W	0	gpio_pc_pupd	PC 上拉或下拉 电阻 这些位由软件编写, 用于规划 I/O 上拉 或下拉 [2y+1:2y]: 上拉或下拉于埠 X。 2'b00: No 上拉, 下拉 2'b01: 上拉 2'b10: 下拉 2'b11: 保留
<b>Reserved</b>					
20ch	31:0	-	-	-	保留
<b>GPIOC_DO: Pad output data</b>					
210h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pc_out	PC 输出数据
<b>GPIOC_DI: Pad input data</b>					
214h	31:16	-	-	-	保留
	15:0	R	-	gpio_pc_in	PC 输入数据
<b>GPIOC_OTYPE: Pad output type</b>					
218h	31:16	-	-	-	保留

Index	Bit	R/W	Default	Name	Description
	15:0	R/W	0	gpio_pc_ot	PC 输出类型 0: 推挽式 (Push-pull) 1: 开汲极 (Open-drain) 注意: 在 开汲极 类型, 如果将 gpio_pc_pupd 致能为上拉, 则逻辑 1 由内部上拉电阻驱动引脚; 否则, 逻辑 1 应由外部上拉电阻驱动。
<b>GPIOC_DS: Pad driving strength</b>					
21ch	15:0	W	-	gpio_pc_ds	PC 驱动能力 0: 低 1: 高
<b>GPIOC_BT_RST: Pad bit reset</b>					
220h	31:16	-	-	-	保留
	15:0	W	-	gpio_pc_br	PC 位重置 0: 无效 1: 将 gpio_pc_out 对应位重置为"0"
<b>GPIOC_BT_SET: Pad bit set</b>					
224h	31:16	-	-	-	保留
	15:0	W	-	gpio_pc_bs	PC 位设置 0: 无效 1: 将 gpio_pc_out 对应位设置为"1"
<b>GPIOC_BT_TGLE: Pad bit toggle</b>					
228h	31:16	-	-	-	保留
	15:0	W	-	gpio_pc_bt	PC 位变化 (toggle) 0: 无效 1: 切换相对反应的位变化 gpio_pc_out "0->1" 或"1->0"
<b>GPIOC_AF0: Pad alt function</b>					
240h	31:0	R/W	0	gpio_pc_af0	PC 埠 0~7 的复用功能。 这些位由软件编写, 用于规划复用功能 I/O。 [4y+3:4y]: 端口 x 的模式规划。 0: 复用功能 0 (AF0) 1: 复用功能 1 (AF1) 2: 复用功能 2 (AF2) 3: 复用功能 3 (AF3) 4: 复用功能 4 (AF4) 5: 复用功能 5 (AF5) 6~15: 保留
<b>GPIOC_AF1: Pad alt function</b>					
244h	31:0	R/W	0	gpio_pc_af1	PC 埠 8~15 的复用功能。 这些位由软件编写, 用于规划复用功能 I/O。 [4y+3:4y]: 端口 x 的模式规划。 0: 复用功能 0 1: 复用功能 1 2: 复用功能 2 3: 复用功能 3 4: 复用功能 4 5: 复用功能 5 6~15: 保留
<b>Reserved</b>					

Index	Bit	R/W	Default	Name	Description
248h	31:0	-	-	-	保留
<b>Reserved</b>					
24ch	31:0	-	-	-	保留
<b>GPIOC_IE: Pad interrupt enable</b>					
260h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pc_ie	PC 中断功能 0: 禁能 1: 致能
<b>GPIOC_ISS: Pad interrupt sense selection</b>					
264h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pc_iss	PC 中断输入选择。 0: 边缘 (Edge) 触发。 1: 电位 (Level) 触发。
<b>GPIOC_BET: Pad interrupt both edges trigger</b>					
268h	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pc_bet	PC 两个边缘触发中断。 0: 中断条件由 gpio_pc_trg 控制。 1: 致能上升缘和下降缘触发。 注意: 当电位 (Level) 触发时, 将忽略此缓存器。
<b>GPIOC_TRG: Pad interrupt trigger event select</b>					
26ch	31:16	-	-	-	保留
	15:0	R/W	0	gpio_pc_trg	PC 触发事件选择。 0: 下降缘或低电位触发。 1: 上升缘或高电位触发。
<b>GPIOC_IF: Pad interrupt raw flag</b>					
270h	31:16	-	-	-	保留
	15:0	R/W1 c	0	gpio_pc_if	PC 中断原始旗标。 此缓存器指示 GPIO 原始中断的状态。如果致能 gpio_pd_ie 缓存器中的相应位, 则 GPIO 中断将发送到中断控制器。 0: 无中断。 1: 有中断发生。
<b>GPIOD_MODE: Pad mode register</b>					
300h	31:10	-	-	-	保留
	9:0	R/W	3ffh	gpio_pd_mode	PD 0~2 的模式缓存器。 这些位由软件编写以配置 I/O 模式。 [2y+1:2y]: X 端口的模式设定。 2'b00: 输入模式 2'b01: 普通输出模式 2'b10: 复用功能模式 2'b11: 模拟功能模式
<b>Reserved</b>					
304h	31:0	-	-	-	保留
<b>GPIOD_PUPD: Pad pull-up/pull-down register</b>					
308h	31:10	-	-	-	保留
	9:0	R/W	0	gpio_pd_pupd	PD 上拉或下拉电阻 这些位由软件编写, 用于规划 I/O 上拉 或 下拉 [2y+1:2y]: 上拉或下拉于埠 X。



Index	Bit	R/W	Default	Name	Description
					2'b00: No 上拉, 下拉 2'b01: 上拉 2'b10: 下拉 (PD1 无下拉功能) 2'b11: 保留
<b>Reserved</b>					
30ch	31:0	-	-	-	保留
<b>GPIOD_DO: Pad output data</b>					
310h	31:5	-	-	-	保留
	4:0	R/W	0	gpio_pd_out	PD 输出数据
<b>GPIOD_DI: Pad input data</b>					
314h	31:5	-	-	-	保留
	4:0	R	-	gpio_pd_in	PD 输入数据
<b>GPIOD_OTYPE: Pad output type</b>					
318h	31:5	-	-	-	保留
	4:0	R/W	0	gpio_pd_ot	PD 输出类型. 0: 推挽式. (Push-pull) 1: 开汲极. (Open-drain) 注意: 在开汲极类型中, 如果将 gpio_pd_pupd 致能为上拉, 则逻辑 1 由内部上拉电阻驱动引脚; 否则, 逻辑 1 应由外部上拉电阻驱动。
<b>GPIOD_DS: Pad driving strength</b>					
31ch	15:0	W	-	gpio_pd_ds	PD 驱动能力 0: 低 1: 高
<b>GPIOD_BT_RST: Pad bit reset</b>					
320h	31:5	-	-	-	保留
	4:0	W	-	gpio_pd_br	PD 位重置 0: 无效 1: 将 gpio_pd_out 对应位重置为"0"
<b>GPIOD_BT_SET: Pad bit set</b>					
324h	31:5	-	-	-	保留
	4:0	W	-	gpio_pd_bs	PD 位设置 0: 无效 1: 将 gpio_pd_out 对应位设置为"1"
<b>GPIOD_BT_TGLE: Pad bit toggle</b>					
328h	31:5	-	-	-	保留
	4:0	W	-	gpio_pd_bt	PD 位变化 (toggle) 0: 无效 1: 切换相对反应的位变化 gpio_pd_out "0->1" 或"1->0"
<b>GPIOD_AF0: Pad alt function</b>					
340h	31:20				PD 埠 0~4 的 复用功能。 这些位由软件编写, 用于规划复用功能 I/Os。 [4y+3:4y]: X 端口的模式设定。
	19:0	R/W	0	gpio_pd_af0	0: 复用功能 0 (AF0) 1: 复用功能 1 (AF1) 2: 复用功能 2 (AF2) 3: 复用功能 3 (AF3)

Index	Bit	R/W	Default	Name	Description
					4: 复用功能 4 (AF4) 5: 复用功能 5 (AF5) 6~15: 保留
<b>Reserved</b>					
344h	31:0	-	-	-	保留
<b>Reserved</b>					
348h	31:0	-	-	-	保留
<b>Reserved</b>					
34ch	31:0	-	-	-	保留
<b>GPIOD_IE: Pad interrupt enable</b>					
360h	31:5	-	-	-	保留
	4:0	R/W	0	gpio_pd_ie	PD 中断功能。 0: 禁能 1: 致能
<b>GPIOD_ISS: Pad interrupt sense selection</b>					
364h	31:5	-	-	-	保留
	4:0	R/W	0	gpio_pd_iss	PD 中断输入选择。 0: 边缘(Edge)触发。 1: 电位(Level)触发。
<b>GPIOD_BET: Pad interrupt both edges trigger</b>					
368h	31:5	-	-	-	保留
	4:0	R/W	0	gpio_pd_bet	PD 两个边缘触发中断。 0: 中断条件由 gpio_pd_trg 控制。 1: 致能上升缘和下降缘触发。 注意: 当电位(Level)触发时, 将忽略此缓存器。
<b>GPIOD_TRG: Pad interrupt trigger event select</b>					
36ch	31:5	-	-	-	保留
	4:0	R/W	0	gpio_pd_trg	PD 触发事件选择。 0: 下降缘或低电位触发。 1: 上升缘或高电位触发。
<b>GPIOD_IF: Pad interrupt raw flag</b>					
370h	31:5	-	-	-	保留
	4:0	R/W1 c	0	gpio_pd_if	PD 中断原始旗标。 此缓存器指示 GPIO 原始中断的状态。如果致能 gpio_pd_ie 缓存器中的相应位, 则 GPIO 中断将发送到中断控制器。 0: 无中断。 1: 有中断发生。

R/W1C: 读取 与 写“1”清除

## 8.4 功能描述

根据每个 I/O 的硬件特性，通用 I/O (GPIO) 的每个口可以通过软件以多种模式单独配置：

- 输入悬空 (Input floating)
- 输入带上拉电阻 (Input pull-up)
- 输入带下拉电阻 (Input pull-down)
- 模拟
- 输出开极汲，具有 pull-up / pull-down
- 输出推拉式，具有 pull-up / pull-down
- 复用功能，具有 pull-up / pull-down 功能

每个 I/O 端口都是可自由程序设定的；但是，I/O 端口缓存器必须作为 32 位字进行存取。Gpio\_px\_br、gpio\_px\_bs 和 gpio\_px\_bt 缓存器的目的是允许对任何 gpio\_px\_out 缓存器的原子进行读取/修改存取。这样，读取和修改存取之间不存在发生 IRQ 的风险。图 28 显示了标准 I/O 口的基本结构。

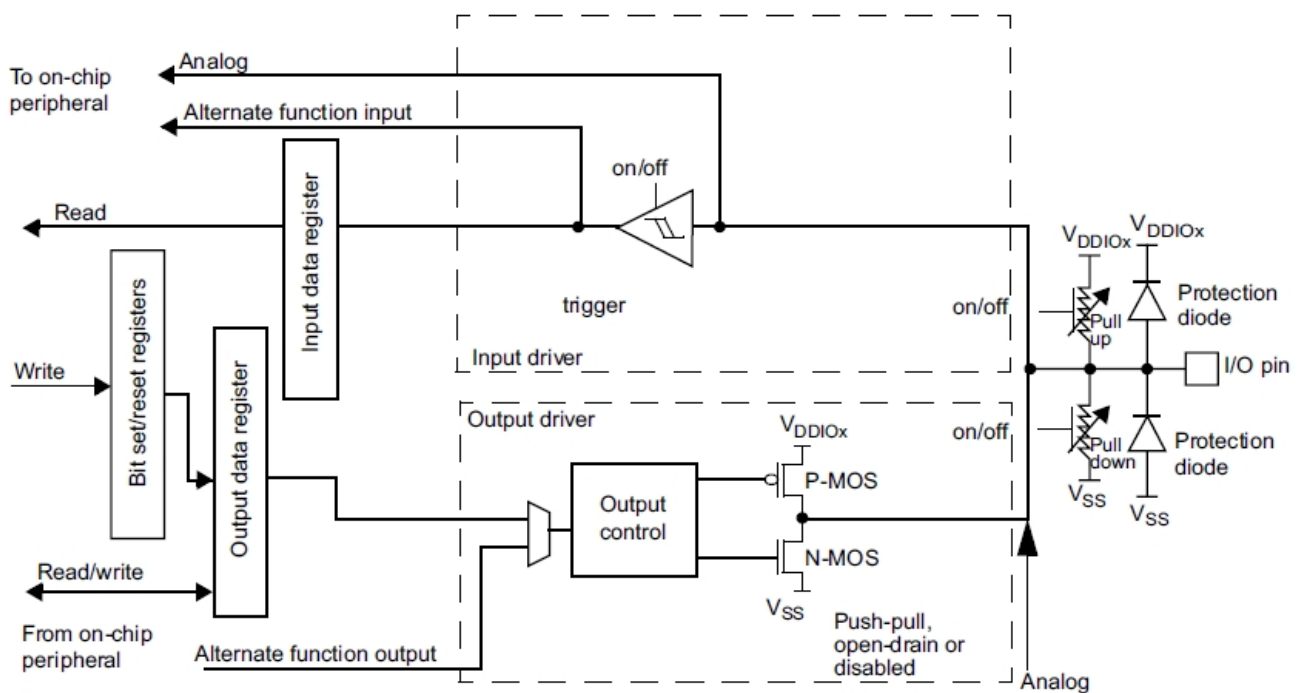


图 28 I/O 端口位的基本结构

### 8.4.1 泛用型 I/O

在复位期间和复位后，复用功能未启动，并且大多数 I/O 端口在模拟模式下规划。

复位后，调试引脚处于 AF 模式上拉或下拉：

- PA14: SWCLK 使用下拉 (pull-down)
- PA13: SWDIO 使用上拉 (pull-up)

当引脚规划为输出时，写入输出数据缓存器 (gpio\_px\_out) 的值将输出到 I/O 引脚上。可以在推拉模式或开极模式中使用输出磁盘驱动器 (仅驱动低电平，高电平为 HI-Z)。输入数据缓存器 (gpio\_px\_in) 捕获每个 AHB 时钟周期的 I/O 引脚上存在的数据。

### 8.4.2 泛用型 I/O

设备 I/O 引脚通过多任务器连接到板载外围/模块，该多任务器一次只允许一个外围设备复用功能 (Alternate Function, AF) 连接到 I/O 引脚。这样，同一 I/O 引脚上可用的外围设备之间不会发生冲突。每个 I/O 引脚都有一个多任务器，具有多达 16 个复用功能输入 (AF0 到 AF15)，可利用 gpio\_px\_af0 和 gpio\_px\_af1 缓存器进行规划：

- 复位后，多任务器选择是复用功能 0 (AF0)。I/O 通过规划 gpio\_px\_mode 缓存器以使用复用功能模式。
- 章节 2.3 “复用功能 I/O 与优先权”中详细介绍了每个引脚的特定复用函数分配。

除了这种灵活的 I/O 多任务架构外，每个外围都有映像到不同 I/O 引脚的复用功能，以优化较小封装中可用的外围设备数量。要在给定规划中使用 I/O，使用者必须按照以下步骤操作：

- 调试功能：每个设备复位后，这些引脚被指定为复用功能引脚，由调试器主机立即使用
- GPIO：在 gpio\_px\_mode 缓存器中将所需的 I/O 规划为输出、输入或模拟。
- 外围复用功能：
  - 将 I/O 连接到一个 gpio\_px\_af0 或 gpio\_px\_af1 缓存器中的所需 Afx。
  - 分别通过 gpio\_px\_ot、gpio\_px\_pu 和 gpio\_pa\_pd 缓存器选择输出类型、上拉或下拉。
  - 将所需的 I/O 规划为 gpio\_px\_mode 缓存器中的复用功能。

### 8.4.3 I/O port 控制缓存器

每个 GPIO 口都有三个内存映像控制缓存器 (gpio\_px\_mode、gpio\_px\_ot、gpio\_px\_pupd) 来配置 I/O。gpio\_px\_mode 缓存器用于选择 I/O 模式 (输入、输出、AF 模式、模拟)。Gpio\_px\_ot 缓存器用于选择输出类型 (推拉或开极)。无论此时 I/O 方向如何，gpio\_px\_pupd 缓存器用于选择向上/向下拉。

### 8.4.4 I/O port 数据缓存器

每个 GPIO 都有两个内存映像的数据缓存器：输入和输出数据缓存器 (gpio\_px\_in 和 gpio\_px\_out)。Gpio\_px\_out 存储要输出的数据，它是可读/写的。通过 I/O 输入的数据存储在输入数据缓存器 (gpio\_px\_in) 中，这是一个只读缓存器 (read-only register)。

### 8.4.5 I/O data bitwise handling

位的重置/设置/切换缓存器 (gpio\_px\_br、gpio\_px\_bs、gpio\_px\_bt)，允许应用程序重置/设置/切换输出数据缓存器 (gpio\_px\_out) 中的每个单独的位。

对于 gpio\_px\_out[i] 中的每个位，在 gpio\_px\_br[i]、gpio\_px\_bs[i] 和 gpio\_px\_bt[i] 中的对应位。当写入 1 时对 gpio\_px\_br[i] 重置相应的 gpio\_px\_out[i] 位。当写入 1 时对 gpio\_px\_bs[i] 设置 gpio\_px\_out[i] 相应的位。当写入 1 时对 gpio\_px\_bt[i] 切换 gpio\_px\_out[i] 对应位。顺便说一下，在 gpio\_px\_br / gpio\_px\_bs 与 gpio\_px\_bt 中写入任何位为 0，对 gpio\_px\_out 中的相应位没有任何影响。

使用 gpio\_px\_br 与 gpio\_px\_bs 与 gpio\_px\_bt 缓存器来更改 gpio\_px\_out 中各个位的值是一种“一次性 (one-shot)”效果，它不锁定 gpio\_px\_out 位。可以随时直接存取 gpio\_px\_out 位。Gpio\_px\_out 缓存器提供了一种执行原子 (atomic) 位处理的方法。

### 8.4.6 I/O 复用 (Alternate) 功能之输入/输出

提供复用功能缓存器，以选择可用于每个 I/O 的复用函数输入/输出之一。通过这些缓存器，用户可以根据应用程序的要求将复用功能连接到一些其它引脚。

这意味着使用 `gpio_px_af0` 和 `gpio_pa_af1` 复用功能缓存器在每个 GPIO 上多任务许多可能的外围函数。因此，应用程序可以为每个 I/O 选择任意一个可能的函数。AF 模式选择信号是复用函数输入和复用功能输出的通用信号，为给定 I/O 的复用函数输入/输出选择单个通道。

要了解每个 GPIO 口上多任务的功能，请参阅章节 2.3“复用功能 I/O 与优先权”。

### 8.4.7 输入规划

当 I/O 口程序设定为输入时：

- 输出缓冲区已关闭
- 施密特触发输入被启动
- 上拉和下拉电阻器根据 `gpio_px_pupd` 缓存器中的值启动
- 每个 AHB 时钟周期，I/O 引脚上的数据都会采样到输入数据缓存器中
- 对输入数据缓存器的读取存取提供 I/O 状态

图 29 显示了 I/O 口的输入规划。

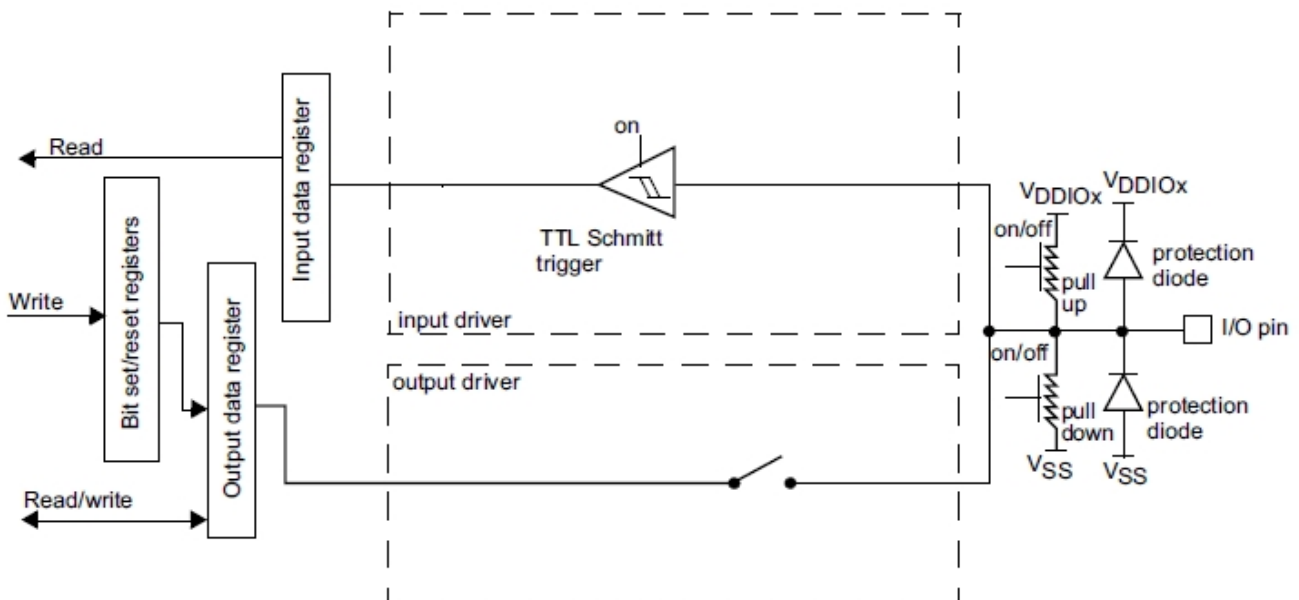


图 29 输入规划

### 8.4.8 输出规划

当 I/O 口程序设定为输出时：

- 输出缓冲区已开启

- 设定开汲极(Open-drain)模式: 输出缓存器中的“0”会启动下部 N-MOS, 而输出缓存器中的“1” 会使该口进入 Hi-Z 的状态 (P-MOS 永远不会启动)
- 推拉(Push-pull)模式: 输出缓存器中的“0”会启动 N-MOS, 而输出缓存器中的“1”会启动 P-MOS
- 施密特触发于输入时被启动
- 下拉和下拉电阻器根据 gpio\_px\_pupd 缓存器中的数值去启动
- 每个 AHB 时钟周期, I/O 引脚上的数据都会采样到输入数据缓存器中
- 对输入数据缓存器进行读取以存取 I/O 状态
- 对输出数据缓存器进行读取以存取最后写入的数值

图 30 显示了 I/O 端口位的输出规划。

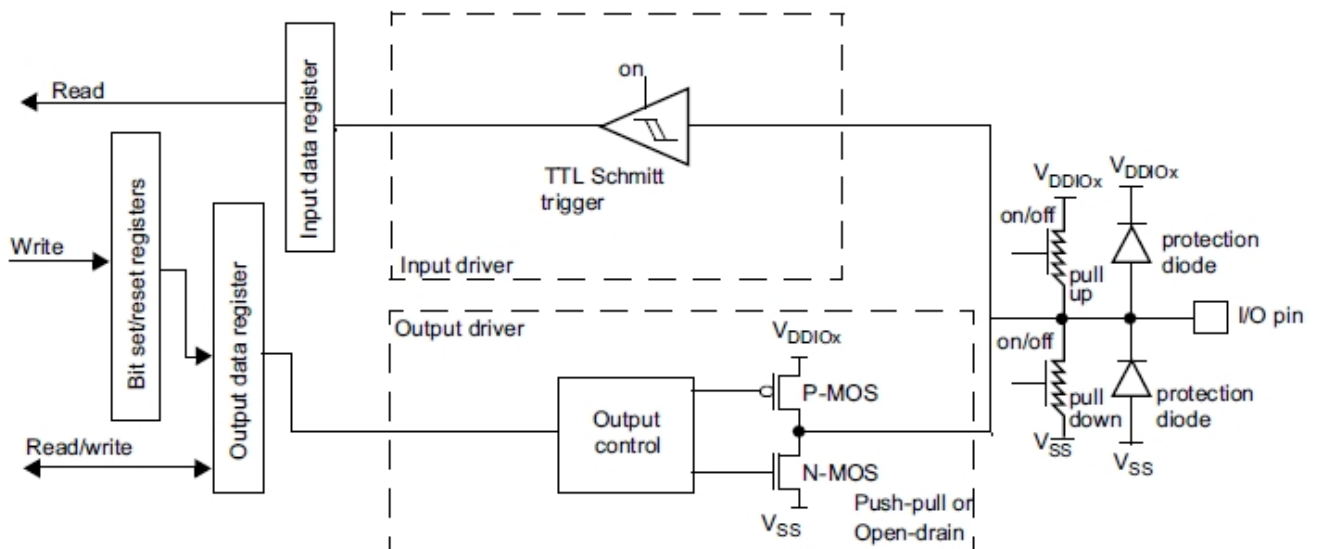


图 30 输出规划

#### 8.4.9 复用 (Alternate) 功能规划

当 I/O 口被程序设定为复用功能时:

- 输出缓冲器皆可在开汲极或推拉模式下规划
- 输出缓冲器由来自外围的信号驱动 (发送器的启动和数据)
- 施密特触发于输入时会被启动
- 弱上拉和下拉电阻器的启动取决于 gpio\_px\_pupd 缓存器中的数值
- 每个 AHB 时钟周期, I/O 引脚上的数据都会被采样到输入数据缓存器中
- 对输入数据缓存器的读取存以取获取 I/O 状态

图 31 显示了 I/O 端口位的复用功能规划。

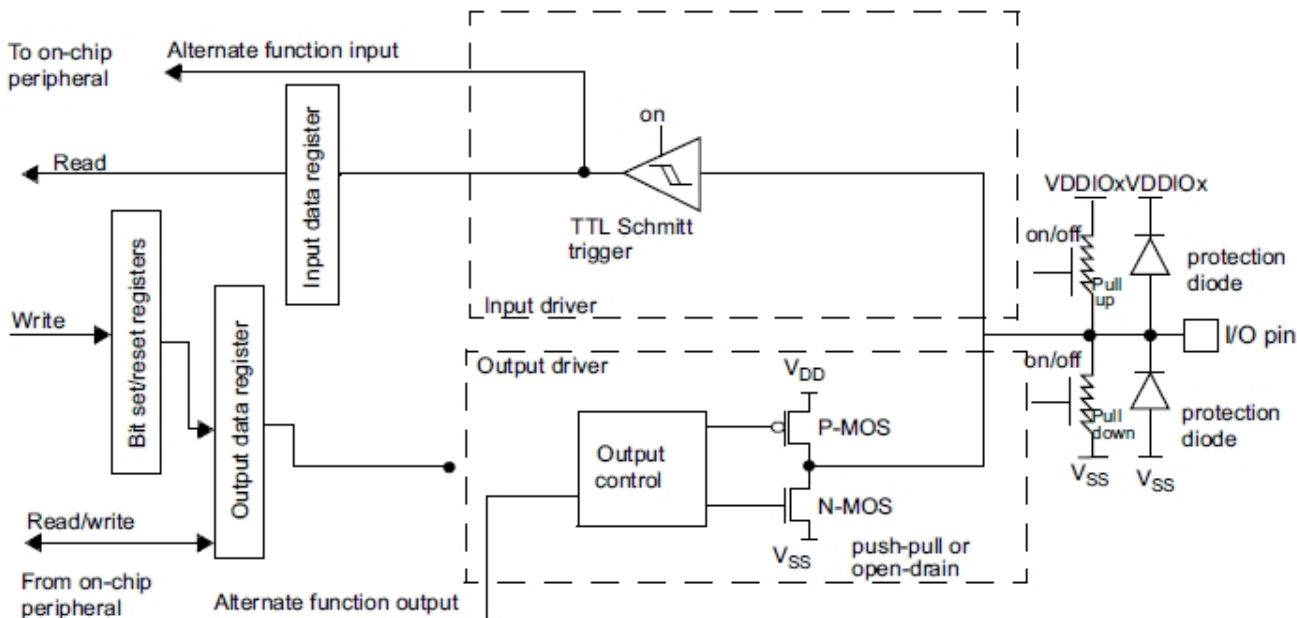


图 31 复用功能规划

#### 8.4.10 模拟功能规划

当 I/O 口程序设定为模拟规划时:

- 输出缓冲区已被关闭
- 施密特触发的输入功能已停用，使 I/O 引脚的每个模拟状态可避免功率消耗。施密特触发器的输出强制设为常数值“0”。
- 硬件关闭弱上拉和下拉电阻
- 对输入数据缓存器的读取存取只能获取“0”

图 32 显示了 I/O 端口位的高阻抗模拟输入规划。

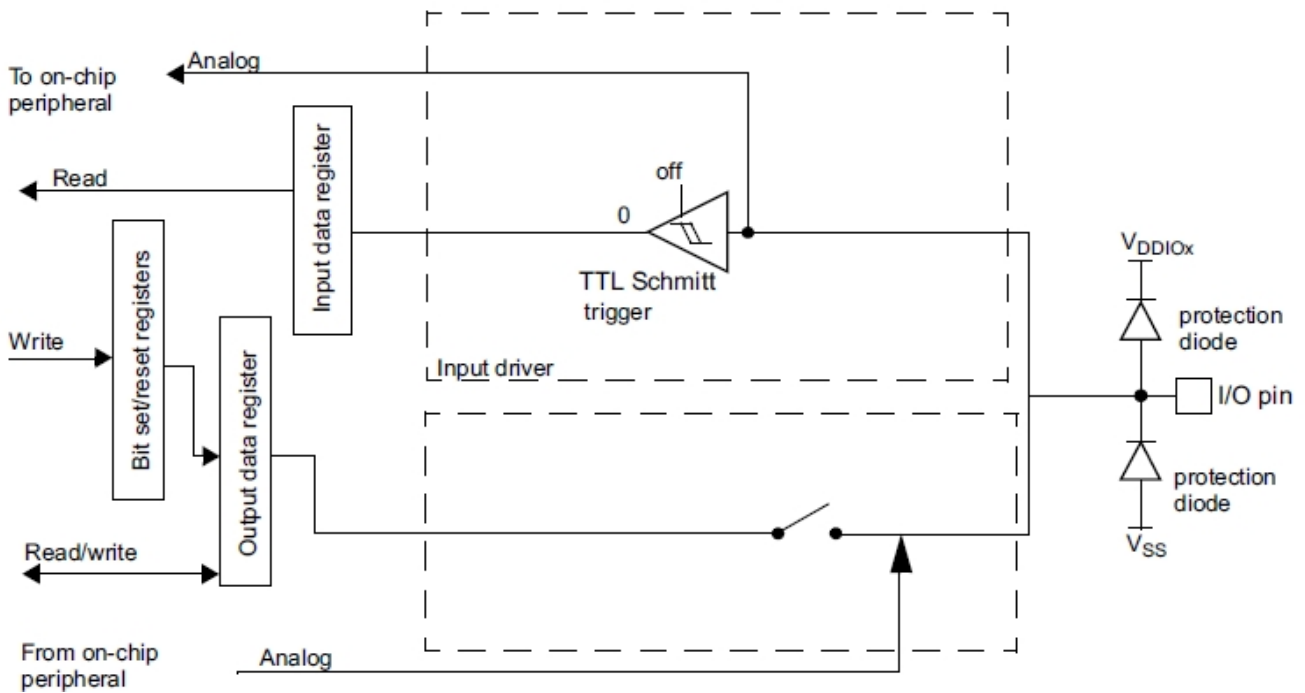


图 32 模拟规划

#### 8.4.11 使用 HXTAL 或 LXTAL 振荡器引脚作为 GPIOs

- ✓ 当 HXTAL 或 LXTAL 振荡器关闭时，相关的振荡器引脚可用作普通 GPIO。
- ✓ 当 HXTAL 或 LXTAL 振荡器开关打开时，振荡器将控制其关联的引脚，并且这些引脚的 GPIO 规划不起作用。



## 9 USB

### 9.1 主要概述

USB 外围实现了全速 USB 2.0 总线和 APB 总线之间的接口。支持 USB 暂停/恢复(Suspend and Resume), 允许停止设备时钟以降低功耗。

- 兼容 USB 2.0 全速操作(Full Speed)
- 兼容 USB 音频装置类规格 V1.0 (Audio Device Class Spec V1.0)
- 1 个控制端点 (Endpoint 0), IN/OUT 各有 64B FIFO (8/16/32/64B 可程序设定)
- 7 个通用端点 (Endpoint 1~6 和 Endpoint 9), 可程序化设定为 IN/OUT、INT/Bulk。
- 2 个同步端点 (Isochronous Endpoint 7、8), 支持 DMA 传输方式( SRAM 和 USB FIFO 之间)
- 端点(Endpoint) 0~9 的 FIFO 总共大小:1024B+576B
  - Endpoint 7 & 8: 共享 1024B (支援 DMA)
  - Endpoint 0: In 64B, out 64B
  - Endpoints 1~ 6, 9: 64B/each
- 支持 USB 暂停(Suspend)、恢复(Resume)和远程唤醒(Remote-Wakeup)
- 非 USB 应用时可关闭 USB 功能

### 9.2 方块图

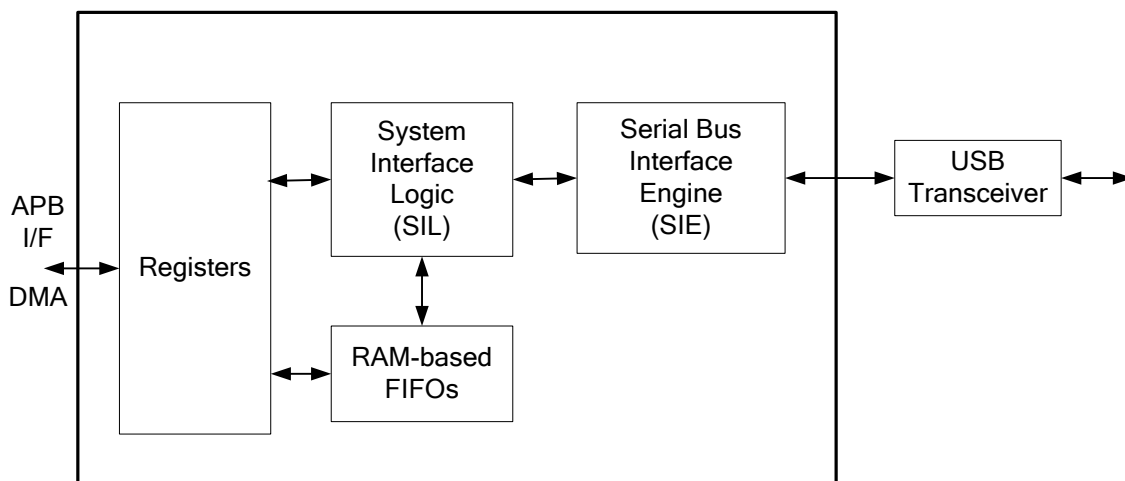


图 33 USB 方块图

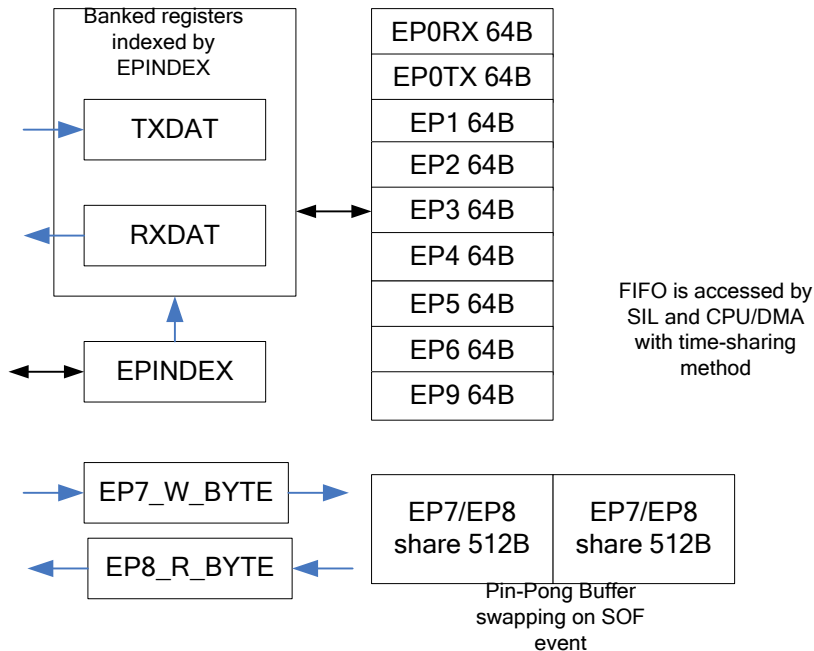


图 34 USB Endpoints FIFO

### 9.3 缓存器描述

表格 20 USB 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>USB_FADDR: USB Function Address</b>					
00h	31:7	-	-	-	保留
	6:0	R/W	0	fa	7 位可程序化之 USB 功能地址
<b>USBFI: USB Function Interrupt</b>					
04h	31:11	-	-	-	保留
	10	R	0	usbx9int	USB 端点9之 Tx/Rx 工作完成旗标
	9:8	-	-	-	保留
	7	R	0	usbx6int	USB 端点6之 Tx/Rx 工作完成旗标
	6	R	0	usbx5int	USB 端点5之 Tx/Rx 工作完成旗标
	5	R	0	usbx4int	USB 端点4之 Tx/Rx 工作完成旗标
	4	R	0	usbx3int	USB 端点3之 Tx/Rx 工作完成旗标
	3	R	0	usbx2int	USB 端点2之 Tx/Rx 工作完成旗标
	2	R	0	usbx1int	USB 端点1之 Tx/Rx 工作完成旗标
	1	R	0	usbrx0int	USB 端点0之 Rx 工作完成旗标
0	R	0	usbtx0int	USB 端点0之 Tx 工作完成旗标	
<b>USB_USBFIE: USB Function Interrupt Enable</b>					
08h	31:11	-	-	-	保留
	10	R/W	0	usbx9int_ie	USB 端点9之 Tx/Rx 工作完成中断致能设置

Index	Bit	R/W	Default	Name	Description
	9:8	-	-	-	保留
	7	R/W	0	usbx6int_ie	USB 端点 6 之 Tx/Rx 工作完成中断致能设置
	6	R/W	0	usbx5int_ie	USB 端点 5 之 Tx/Rx 工作完成中断致能设置
	5	R/W	0	usbx4int_ie	USB 端点 4 之 Tx/Rx 工作完成中断致能设置
	4	R/W	0	usbx3int_ie	USB 端点 3 之 Tx/Rx 工作完成中断致能设置
	3	R/W	0	usbx2int_ie	USB 端点 2 之 Tx/Rx 工作完成中断致能设置
	2	R/W	0	usbx1int_ie	USB 端点 1 之 Tx/Rx 工作完成中断致能设置
	1	R/W	0	usbrx0int_ie	USB端点0之Rx工作完成中断致能设置
	0	R/W	0	usbtx0int_ie	USB端点0之Tx工作完成中断致能设置
<b>USB_SIEI: SIE Interface</b>					
0ch	31:6	-	-	-	保留
	5	R/W	0	usben	USB功能致能设置
	4	R/W	0	dm	DM上拉电阻致能设置
	3	R/W	0	psofen	致能Pseudo-SOF 产生器
	2	-	-	-	保留
	1	R/W	0	dp	DP上拉电阻致能设置
	0	W	0	wakeup	USB功能使用此位启动远程唤醒。Firmware设置为1可产生唤醒讯号于USB讯号在线，藉此可唤醒上层的HUB或HOST，当讯号产生完成后，硬件会自动将此位清为0
<b>USB_EPINDEX: Endpoint Index</b>					
14h	31:4	-	-	-	保留
	3:0	R/W	0	epinx	端点索引缓存器，用来选择某个端点传送与接收 FIFO 以及某个端点的特定缓存器库 (USB_TXDAT、USB_TXCON、USB_TXFLG、USB_TXCNT、USB_TXSTAT、USB_RXDAT、USB_RXCON、USB_RXFLG、USB_RXCNT、USB_RXSTAT、USB_EPCON、USB_EPNUM)。当发生USB复位以及硬件复位，此缓存器会回到复位值 0 = 端点0. 1 = 端点1. 2 = 端点2. 3 = 端点3. 4 = 端点4. 5 = 端点5. 6 = 端点6. 7 = 端点7. (only has USB_EPNUM reg) 8 = 端点8. (only has USB_EPNUM reg) 1. = 端点9. Others = 保留
<b>USB_EPCON: Endpoint Control (Endpoint-indexed)</b>					
18h	31:8	-	-	-	保留
	7	R/W	0	rxstl	接收端点 Stall 设置: 将此位设置为 1 可令接收端点进入 Stall 状态, 需由 Host 透过中断从端点 0 下 USB 命令来进行清除; 当此位被设置为 1 同时 RXSETUP 为 0, 此时若 Host

Index	Bit	R/W	Default	Name	Description
					下一个有效的 OUT token 时, 将会响应 Stall; 当此位被设置为 1 同时 RXSETUP 为 1, 此时接收端点将会回应 NAK。此位并不会影响控制端点所收到的 SETUP token。
	6	R/W		txstl	传送端点 Stall 设置: 将此位设置为 1 可令传送端点进入 Stall 状态, 需由 Host 透过中断从端点 0 下 USB 命令来进行清除; 当此位被设置为 1 同时 RXSETUP 为 0, 此时若 Host 下一个有效的 IN token 时, 将会响应 Stall; 当此位被设置为 1 同时 RXSETUP 为 1, 此时接收端点将会回应 NAK。
	5	R/W	0/1	ctlep	控制端点设置: 将此位设置为 1 让此端点为控制端点, 只有控制端点可接收 SETUP tokens
	4	-	-	-	保留
	3	R/W	0	rxie	接收输入致能设置: 将此位设置为 1 以致能 USB 数据写入至接收 FIFO 缓冲区, 假如清为 0, 此端点将不会将接到的数据写入接收 FIFO 缓冲区, 若此时数据为有效的 OUT 封包且 RXSTL 并未被设置, 则在数据接完成后将会回应 NAK。此位不会影响有效的 SETUP token, 当接收到一个有效的 SETUP token 与数据封包, 若此时此位为 0, 则会无视此位且会将收到的数据写入接收 FIFO 缓冲区
	2	R/W	0/1	rxepen	接收端点致能设置: 将此位设置为 1 以致能接收端点; 当此位清除为 0, 此时收到有效的 SETUP 或 OUT tokens 都不会有任何响应。此位为 hardware read-only 且具有最高优先权高于 RXIE 与 RXSTL。需注意的是若为端点 0, 此位需在复位后马上设置为 1 以进行接收 USB 命令。
	1	R/W	0	txoe	传送输出致能设置: 将此位设置为 1 以致能 USB_TXDAT 内的数据可以被传送出去, 若此位清除为 0, 此端点在收到有效的 IN token 且此时 TXSTL 不为 1 时会回应 NAK
	0	R/W	0/1	txepen	传送端点致能设置 将此位设置为 1 以致能传送端点; 当此位清除为 0, 此时即使收到有效的 IN token 亦不会有任何的响应。此位为 hardware read-only。需注意的是若为端点 0, 此位需在复位后马上设置为 1。
<b>USB_TXDAT: Transmit FIFO Data (Endpoint-indexed)</b>					
20h	31:8	-	-	-	保留
	7:0	W	-	txdat	传送数据字节 (唯写): 将一个字节却传送的数据写入传送 FIFO 缓冲区内, 写入完成后, 写入指标 (Write-Point) 会自动加 1
<b>USB_TXCON: Transmit FIFO Control (Endpoint-indexed)</b>					
24h	31:8	-	-	-	保留

Index	Bit	R/W	Default	Name	Description
	7	W	0	txclr	传送 FIFO 缓冲区清除设置: 设置为 1 可清除传送 FIFO 缓冲区内的数据、复位读取指针与写入指标、将 USB_TXFLG.EMPTY 设置为 1、并清除 USB_TXFLG 内除了 USB_TXFLG.EMPTY 之外的其它位。当清除完成, 硬件会自动清除此位。
	6:0	-	-	-	保留
<b>USB_TXFLG: Transmit FIFO Flag (Endpoint-indexed)</b>					
28h	31:4	-	-	-	保留
	3	R	1	txemp	传送 FIFO 缓冲区内容为空之旗标 (只读): 当 SIL 已经将传送 FIFO 缓冲区内的数据全部读出时, 硬件会自动将此位设置为 1, 同时, 当硬件发现传送 FIFO 缓冲区内容不再为空时则会自重将此位清除为 0。此位总是追踪着目前传送 FIFO 缓冲区的状态, 即使传送了长度为 0 的封包时, 此位亦会被设置为 1。
	2	R	0	txfull	传送 FIFO 缓冲区内容已满之旗标 (只读): 当 Firmware 设置寄存器 USB_TXCNT, 则此位会自动设置为 1 以反映出此时传送 FIFO 缓冲区的状况, 当此笔数据被成功送出之后, 硬件会自动清除此位。
	1	R/W c	0	txurf	传送 FIFO 缓冲区数据欠载旗标 (只读、只可清除): 当传送 FIFO 缓冲区内数据已空时, 又额外再进行读取, 此时硬件会将此位设置为 1, 此时当收到 IN token 会响应 NAK, 需透过 Firmware 将此位清除为 0。当传送 FIFO 缓冲区发生欠载的状况, 此时读取指标将不会再往前增加, 会锁定在空的位置上。
	0	R/W c	0	txovf	传送 FIFO 缓冲区数据过载旗标 (只读、只可清除): 当传送 FIFO 缓冲区内数据已满时 (TXFULL = 1), 又额外再写入数据, 此时硬件会自动将此位设置为 1, 此时当收到 IN token 会响应 NAK, 需透过 Firmware 将此位清除为 0。当传送 FIFO 缓冲区发生过载的状况, 此时写入指标将不会再往前增加, 会锁定在满的位置上。
<b>USB_TXCNT: Transmit FIFO Byte Count (Endpoint-indexed)</b>					
2ch	31:7	-	-	-	保留
	6:0	W	0	txcnt	传送字节数量 (唯写): 将已写入传送 FIFO 缓冲区 (透过寄存器 USB_TXDAT) 内的字节数量填入此寄存器以进行传送, 当其被进行填值时, TXFULL 会自动被设置为 1。
<b>USB_TXSTAT: Endpoint Transmit Status (Endpoint-indexed)</b>					
30h	31:8	-	-	-	保留
	7	R/W c	0	txseq	传送顺序位 (只读、只能清除): 此位将于下一个 PID 时进行传送, 之后当收到有效 ACK 会自动在 0 与 1 之间进行切换。当收到一个有效的 SETUP token 后, 硬件会自动切换此位, SIE

Index	Bit	R/W	Default	Name	Description
					会自动处理与追踪此顺序位，所以此位只需在初始化一个新的设置或接口才需要进行清除。
	6:3	-	-	-	保留
	2	R	0	txvoid	传送空白状态 (只读): 在响应了一个有效的 IN token 之后就会发生传送空白状态，此状态与传送完成后所响应的 NAK 或 STALL 有非常紧密的关连，这是由于此状况会造成传送 FIFO 缓冲区无法或尚未准备好传送。使用此位来检查传送后的响应为 NAK 或是 STALL。此位并不会影响 USBTxINT、TXERR 或 TXACK 位，且硬件会在完成一个异步传送结束后自动更新此位。
	1	R	0	txerr	传送错误状态 (只读): 此位为 1 时指出传送时发生错误，全部或部份的数据已被传送，此错误可能为以下 2 种状况: 1. 数据传送完成，但没有接收到 handshake 2. 传送 FIFO 缓冲区在传送时发生数据欠载的状况 在数据传送结束时，硬件会自动更新此状态位与 TXACK 状态位，且此位与 TXACK 状态位乃与互斥状态
	0	R	0	txack	传送承认状态 (只读): 此位为 1 时指出传送完成且正确被接收，在数据传送结束时，硬件会自动更新此状态位与 TXERR 状态位，且此位与 TXERR 状态位乃与互斥状态
<b>USB_EPNUM: Endpoint Number (Endpoint-indexed)</b>					
34h	31:4	-	-	-	保留
	3:0	R/W		epnm	USB 复位后，Endpoint 1~9 的默认值为 1~9。用户可以更改该值以更改显示给 USB 主机的映像 Endpoint 编号。
<b>USB_DBCON: Double Buffer Control (Endpoint-indexed, for EP1~6 and 9)</b>					
38h	31:8	-	-	-	保留
	7	R/W	0	dben	Double buffer 启用 (for EP1~6 and 9)
	6:5	-	-	-	保留
	4:0	R/W		dbbank	双缓冲库编号。每区块(bank)占用 64 个字节。使用此缓存器可以分配第二个缓冲区的地址。
<b>USB_NOTEBOOK: General purpose 8-bit register</b>					
3ch	31:8	-	-	-	保留
	7:0	R/W	0	notebook	通用之 8 位缓存器
<b>USB_USBFI_CLR: USB Function Interrupt Clear</b>					
40h	31:11	-	-	-	保留
	10	W		usbx9int_clr	USB 端点 9 中断清除设置。写 1 以清除 USB 端点 9 中断
	9:8	-	-	-	-
	7	W		usbx6int_clr	USB 端点 6 中断清除设置。写 1 以清除 USB 端点 6 中断
	6	W		usbx5int_clr	USB 端点 5 中断清除设置。写 1 以清除 USB 端点 5 中断

Index	Bit	R/W	Default	Name	Description
	5	W		usbx4int_clr	USB 端点 4 中断清除设置。写 1 以清除 USB 端点 4 中断
	4	W		usbx3int_clr	USB 端点 3 中断清除设置。写 1 以清除 USB 端点 3 中断
	3	W		usbx2int_clr	USB 端点 2 中断清除设置。写 1 以清除 USB 端点 2 中断
	2	W		usbx1int_clr	USB 端点 1 中断清除设置。写 1 以清除 USB 端点 1 中断
	1	W		usbrx0int_clr	USB 端点 0 接收中断清除设置。写 1 以清除 USB 端点 0 接收中断
	0	W		Usbtx0int_clr	USB 端点 0 传送中断清除设置。写 1 以清除 USB 端点 0 接收中断
<b>USB_USBFI2: USB Function Interrupt Register 2</b>					
44h	31:8	-	-	-	保留
	7	R/W c	0	usbint	USB 复位中断旗标
	6	R/W c	0	resume	USB SIE 检测到 USB 线路上的 RESUME 信号。此中断用于终止低功耗模式。
	5	R/W c	0	suspend	USB SIE 检测到 USB 线路之上之暂停信号。相应的 ISR 应将整个芯片置于低功耗模式。
	4	-	-	-	保留
	3	R/W c	0	sofint	SOF/PSOF 中断旗标
	2:0	-	-	-	保留
<b>USB_USBFI2: USB Function Interrupt Enable Register 2</b>					
48h	31:8	-	-	-	保留
	7	R/W	0	nakint_ie	NACK 中断启用: 设置此位以启用 USB NACK 中断, 此时不管回应 ACK、NAK、或 STALL 皆会发生中断
	6	R/W	0	resume_ie	USB RESUME 中断启用。
	5	R/W	0	suspend_ie	USB SUSPEND 中断启用。
	4	R/W	0	stallint_ie	STALL 中断启用: 设置此位以启用 USB stall 中断, 当回应 STALL 时则会发生中断。
	3	R/W	0	tgerr_ie	RX-TOGGLE-ERROR 中断启用: 设置此位以启用 RX-TOGGLE-ERROR 中断, 当接收数据的 toggle 位发生错误时会发生中断 (RX_ERR/RXACK 会清为 0)
	2	R/W	0	sof_ie	SOF/PSOF 中断启用
	1	R/W	0	usbrt_ie	USB RESET 中断启用
	0	R/W	0	resume_ie2	USB RESUME 中断启用 2, DP=0 时唤醒 MCU。
<b>USB_DBSTAT: Double Buffer Status (Endpoint-indexed, for EP1~6 and 9)</b>					
54h	31:8	-	-	-	保留
	7	R	0	dfull_1	第二个缓冲区已满状态旗标
	6	R	0	dfull_0	第一个缓冲区已满状态旗标
	5	R	0	sil_bi	SIL 将使用之缓冲区: 0: 第一个缓冲区, 1: 第二个缓冲区
	4	R	0	sil_cpu	CPU 将使用之缓冲区:

Index	Bit	R/W	Default	Name	Description
					0: 第一个缓冲区, 1: 第二个缓冲区
	3	-	-	-	保留
	2	R	0	rtfull_cpu	CPU 的缓冲区已满状态。(== dfull[sil_cpu])
	1:0	-	-	-	保留
<b>USB_RXDAT: Receive FIFO Data (Endpoint-indexed)</b>					
60h	31:8	-	-	-	保留
	7:0	R	-	rxdat	接收数据字节 (只读): SIL 收到数据后会透过这个缓存器写入至接收 FIFO 缓冲区; 而 CPU 则利用这个缓存器来从接收 FIFO 缓冲区中读取数据。在进行写入或读取之后, 写入指标与读取指标自动进行累加。
<b>USB_RXCON: Receive FIFO Control (Endpoint-indexed)</b>					
64h	31:8	-	-	-	保留
	7	W	0	rxclr	清除接收 FIFO 缓冲区: 设置此位可将接收 FIFO 缓冲区内数据完全清除, RXFLG 中的全部旗标皆会回到复位值 (即 RXEMP = 1, 而其它的旗标 = 0), 当清除完成后, 硬件会自动清除此位。
	6:5	-	-	-	保留
	4	W	0	rxffrc	接收FIFO 读取完成设置: 当一组数据从接收FIFO缓冲区中完全读出之后, 将此位设置为 1 来释放 FIFO 缓冲区, 此时 USB_RXFLG.RXFULL 会被清为 0, 之后硬件会自动将此位清为 0。当要设置此位时需确认全部的数据已经读出且只有在 STOVW 与 EDOVW 为 0 的情况下方能动作。
	3:0	-	-	-	保留
<b>USB_RXFLG: Receive FIFO Flag (Endpoint-indexed)</b>					
68h	31:8	-	-	-	保留
	7	R/W c	0	sendnak	NAK 旗标 (只读、清除): 1: SIE 回应 NAK 至 Host 端 (当 NAKINT_IE = 1 时方会被设置)
	6	R/W c	0	sendstall	STALL 旗标 (只读、仅清除): 1: 回应 STALL 至 host 端 (当 STALLINT_IE = 1 时方会被设置)
	5	R/W c	0	tgerr	RX-TGERR 旗标 (只读、仅清除): 1: 接收数据 toggle 错误 (当 TGERR_IE = 1 时方会被设置)
	4	-	-	-	保留
	3	R	1	rxemp	接收FIFO缓冲区空之旗标 (read-only): 1: 当一组数据从接收FIFO缓冲区被全部读出时, 硬件会自动将此位设置为 1; 同时当接收FIFO缓冲区不再为空时, 硬件亦会自动将此位清除为 0。此位非一黏着位且总是随着目前的状态进行改变。此位在收到一个长度为 0 的数据时亦会被设置为 1。
	2	R	0	rxfull	接收 FIFO 全旗标 (只读): 此旗标指示接收 FIFO 中存在数据集。硬件在成功接收数据集时设置此位。写入 RXCNT 后将清除此



Index	Bit	R/W	Default	Name	Description
					位，以反映数据集的状况。同样，在设置 RXFFRC 位后将清除此位。
	1	R/W c	0	rxurf	接收FIFO缓冲区数据欠载旗标 (只读、仅清除) 1: 当从空的接收FIFO缓冲区中再读取额外的字节，则硬件会自动将此位设置为1，此时读取指标将会被锁定在空的位置而不会往前。清除的方式需利用firmware将此位清除为0。当此位为1，全部的传输将会响应NAK 注意: 只有 EP0 具有此旗标。
	0	R/W c	0	rxovf	接收 FIFO 缓冲区数据过载旗标 (只读、仅清除) 1: 接收 FIFO 缓冲区已满 (即 RXFULL = 1)，此时若 SIL 再写入一额外的字节数据至缓冲区内，则硬件会自动将此位设置为 1，此时写入指标将会被锁定在满的位置。此为一黏着位且需利用 firmware 将此位清除为 0，即使当收到 SETUP 封包时，硬件亦会自动清除此位。当此位为 1，全部的传输将会响应 NAK 注意: 只有 EP0 具有此旗标。
<b>USB_RXCNT: Receive FIFO Byte Count (Endpoint-indexed)</b>					
6ch	31:7	-	-	-	保留
	6:0	R	0	rxcnt	接收FIFO缓冲区字节数量 (只读): 接收FIFO缓冲区内之字节数量。当此缓存器被写入时，RXFULL会到成功响应ACK之后才会被硬件设置为1。当SIL写入一组数据至接收FIFO缓冲区之后，会将数据的数量写入此缓存器，此时CPU读取此缓存器来得知接收FIFO缓冲区内有多少字节的数据。
<b>USB_RXSTAT: Endpoint Receive Status (Endpoint-indexed)</b>					
70h	31:8	-	-	-	保留
	7	R/W c	0	rxseq	接收 Endpoint 序列位 (读取，仅清除): 当收到 OUT 数据封包且正确响应 ACK，则此位会自动进行 0 与 1 之切换；而当收到 SETUP 封包时，硬件会自动将此位设置为 1。SIL 会自动切换此位，所以只有在初始化一个设置或接口时才需要使用到此位，假如不想改变连续位的值，可在写入此缓存器时将此位设置为 1
	6	R/W c	0	rxsetup	接收 SETUP 封包(Setup Token) (只读、清除): 当接到一笔正确之 SETUP 封包时，硬件会自动将此位设置为 1。当 SIL 将此位设置为 1 时，此后的 IN 或 OUT token 都将会自动应答 NAK 直到此位被清除才允许一控制传输的进行，此时即使此端点为 STALL (RXSTL 或 TXSTL) 状态，IN 或 OUT Token 依然会响应 NAK 以允许系统透过控制传输来将此端点之 STALL 状态进行清除。当 firmware 已完成 SETUP 数据处理后将此位进行清除
	5	R	0	stovw	开始覆写旗标 (read-only): 硬件在开始接收SETUP封包之前会将此位设置为 1，以通知目前正有一笔新的SETUP封包正在写入

Index	Bit	R/W	Default	Name	Description
					至接收FIFO缓冲区。当此位被设置为1，接收FIFO状态 (RXFULL与读取指标) 将会被复位与锁定直到EDOVW被设置为1，如此可避免当正在进行接收FIFO缓冲区的清除以及写入新的数据的时候，刚好firmware也正在读取接收FIFO缓冲区造成读取指标被篡改。当接收完SETUP之数据且响应ACK之后，硬件会自动将此位清除为0，此位只使用于控制端点。
	4	R/W c	0	edovw	结束覆写旗标 (只读、只读): 每当接收完 SETUP 之数据且处于交握阶段时，硬件会自动将此位设置为 1。当要读取接收 FIFO 缓冲区的数据前，需先将此位清为 0。当此位为 1 时，接收 FIFO 缓冲区之状态 (RXFULL 与读取指标) 将会被锁定直到此位被清除为止，如此可避免当正在写入新的数据至接收 FIFO 缓冲区的时候，刚好 firmware 也正在读取接收 FIFO 缓冲区造成读取指标被篡改。此位只使用于控制端点。 注意: 当要读取接收 FIFO 缓冲区的数据前，需先将此位清为 0
	3	-	-	-	保留
	2	R	0	rxvoid	避免接收状态 (只读): 当一 SETUP 或 OUT Token 来了但却没有有效的数据被接收时此位会被设置为 1，一般为以下状况: 1. 接收 FIFO 缓冲区仍然是被锁定的状态 2. 缓存器 USB_EP7CON.RXSTL 被设置为 1 此位是由硬件来进行设置与清除，当硬件响应一有效的 OUT Token 之后，在传输结束时更新此位。
	1	R	0	rxerr	接收错误状态 (只读): 当接收发生错误时，此位会被设置为 1，此时全部或部份的数据已写入至接收 FIFO 缓冲区，且不会响应交握数据，此错误发生于以下状况: 1. 数据之 CRC 检验错误 2. 位填塞 (Bit stuffing) 错误 3. 当接收时接收 FIFO 缓冲区发生过载或欠载的状况 此位会在收到有效的 SETUP 或 OUT token 传输结束时由硬件进行更新，同时相对应的接收完成位亦会被设置。此位与 RXACK 会在数据接收完成时进行更新，且与 RXACK 为互斥之状态。
	0	R	0	rxack	接收确认状态 (只读): 当数据完成接收存入至收 FIFO 缓冲区且响应 ACK 之后，此位会被设置为 1，此位会在收到有效的 SETUP 或 OUT token 传输结束时由硬件进行更新，同时相对应的接收完成位亦会被设置。此位与 RXERR 会在数据接收完成时进行更新，且与 RXERR 为互斥之状态。
<b>USB_EP7_CON: EP7 Control</b>					
80h	31:2	-	-	-	保留

Index	Bit	R/W	Default	Name	Description
	1	R/W	0	ep7_dma_en	1: 启用端点7之DMA传输
	0	R/W	0	ep7_en	0: 禁能端点7功能 1: 启用端点7功能
<b>USB_EP7_STAT: EP7 Status</b>					
84h	31:13	-	-	-	保留
	12	R	0	ep7_tx_full	端点7传送FIFO缓冲区已满状态
	11:10	-	-	-	保留
	9:0	R	0	ep7_tx_len	端点7传送FIFO缓冲区内之字节数量
<b>USB_EP7_BASE: EP7 Base</b>					
88h	31:9	-	-	-	保留
	8:0	R/W	0	ep7_base	USB端点7传送FIFO缓冲区之起始指标 端点7与端点8共享512个字节之FIFO缓冲区，以此缓存器来设置端点7之传送FIFO缓冲区起始指针之位置
<b>USB_EP7_LIMIT: EP7 Limit</b>					
8ch	31:10	-	-	-	保留
	9:0	R/W	100h	ep7_limit	端点7之传送FIFO缓冲区大小设置: 端点7与端点8共享512个字节之FIFO缓冲区，以此缓存器来设置端点7之传送FIFO缓冲区之大小
<b>USB_EP7_W_BYTE: EP7 Write Data Byte</b>					
90h	31:8	-	-	-	保留
	7:0	W	-	ep7_w_byte	端点7 传送FIFO缓冲区数据写入缓存器
<b>USB_EP8_CON: EP8 Control</b>					
a0h	31:2	-	-	-	保留
	1	R/W	0	ep8_dma_en	1: 致能端点8之DMA传输
	0	R/W	0	ep8_en	0: 禁能端点8功能 1: 致能端点8功能
<b>USB_EP8_STAT: EP8 Status</b>					
a4h	31:12	-	-	-	保留
	11	R	0	ep8_rx_empty	端点8接收FIFO缓冲区为空之状态
	10	R	-	ep8_rx_err	端点8错误状态(CRC Error)
	9:0	R	0	ep8_rx_len	端点8 接收FIFO缓冲区内数据之长度
<b>USB_EP8_BASE: EP8 Base</b>					
a8h	31:9	-	-	-	保留
	8:0	R/W	100h	ep8_base	端点8接收FIFO缓冲区之起始指标设置: 端点7与端点8共享512个字节之FIFO缓冲区，以此缓存器来设置端点8之接收FIFO缓冲区起始指针之位置
<b>USB_EP8_LIMIT: EP8 Limit</b>					
ach	31:10	-	-	-	保留
	9:0	R/W	100h	ep8_limit	端点8 之接收FIFO缓冲区大小设置: 端点7与端点8共享512个字节之FIFO缓冲区，以此缓存器来设置端点8之传送FIFO缓冲区之大小
<b>USB_EP8_R_BYTE: EP8 Read Data Byte</b>					
b0h	31:8	-	-	-	保留
	7:0	R	-	ep8_r_byte	端点8接收FIFO缓冲区数据读取缓存器
<b>USB_FRAME: Frame Status</b>					

Index	Bit	R/W	Default	Name	Description
c0h	31:13	-	-	-	保留
	12	R	0	frame_miss	USB 讯框 (Frame) 消失状态 1: 未收到预期该来的SOF (只有当PSOF致能时方为有效)
	11	R	0	frame_error	USB 讯框 (Frame) 错误状态 1: 上一笔SOF封包发生错误
	10:0	R	0	frame	USB 讯框号码

## 9.4 功能描述

### 9.4.1 主要功能模块

USB 模块由四个部份组成，即 USB 全速收发器、串行接口引擎 (SIE)、系统接口逻辑 (SIL)、以及传送/接收 FIFO 缓冲区，其中 USB 收发器提供一个实体接口与 USB 讯号线进行通讯，SIE 负责 USB 通讯协议，SIL 负责数据的传输，同时为 SIE、FIFO 缓冲区、以及 CPU 核心提供一个通讯的接口。

USB 模块中的主要模块包括：

2. **全速 USB 收发器**: 此为一内建之收发器，带有一组差动讯号驱动器，可用来传送数据至 USB 讯号线或者由 USB 讯号线接收由 USB Host 所发出来的数据，此 USB 讯号线即一般 USB 所定义的 D+ 与 D- 两条差动讯号。
3. **串行总线接口引擎 (SIE)**: SIE 负责 USB 协议中的前端功能，举凡时钟 (Clock) 与数据 (Data) 讯号的分离、同步讯号的识别、NRZI-NRZ 的转换、标记封包的译码、数据位的拆解、NRZ-NRZI 的转换、CRC5 校验、以及 CRC16 的产生与校验，此外也负责侦测 USB 讯号线所发生的 USB 复位 (Reset)、休眠 (Suspend)、以及唤醒 (Resume) 等讯号及可远程唤醒上游的 USB Port 如 HUB 或 Host 等。SIE 同时亦负责将 USB 收发器所接收的串行数据转换为八位之并列数据给 SIL 并同时为 SIL 的八位并列数据转换成串行数据以利于 USB 收发器进行发送。
4. **系统接口逻辑 (SIL)**: SIL 为 CPU 提供一接口，让 CPU 得以控制 FIFO 缓冲区以进行数据的传送与接收。SIL 同时具有以下功能如监控 USB 传输的状态、透过中断机制来提供 CPU 控制数据的传输、于 USB 讯号线在线产生唤醒讯号 (系统处于休眠状态时)。CPU 可透过 USB 特殊缓存器来控制 SIL。
5. **装置功能**: USB 装置功能具有 10 个端点且支持 4 个 USB 数据传输模式：控制传输模式、中断传输模式、巨量传输模式、以及异步传输模式。传送 FIFO 缓冲区提供给 CPU 进行数据写入，然后 SIL 再进行数据读取并进行数据传送；接收 FIFO 缓冲区则是提供给 SIL 进行数据写入，然后 CPU 再进行数据读取以取得收到数据。端点 0 支持控制传输模式，提供客户端软件与装置之间进行设置、命令、以及状态通讯流程；端点 1~6 与端点 9 支持中断及巨量传输模式；端点 7 支持同步输入传输；端点 8 支持同步输出传输。

### 9.4.2 功能端点

USB 模块支持 10 个装置功能端点 (Endpoint)，端点 0 包含两个 FIFO 缓冲区，即传送 FIFO 与接收 FIFO 缓冲区，负责控制型数据模式的传输；端点 1~6 与端点 9 可个别程控为传送或接收，可程序规画为中断或巨量传输模式；端点 7 与端点 8 专用于异步传输模式。各个端点的特殊缓存器库乃是透过特殊缓存器 USB\_EPINDEX 进行切换。

### 9.4.3 传输 FIFOs

#### 9.4.3.1 传输 FIFOs 概述

USB 传输 FIFO 是具有以下功能的数据缓冲区 (参见图 35)

- USB 支持一个不大于 8/16/32/64 字节的数据集 (可利用程序进行设置)。
- 一个字节计数缓存器, 用来储存数据集中所含数据之字节数
- 当 FIFO 中已有完整数据时, 会防止被新数据集覆盖
- 当传输因故未成, 能自动重新传输当前数据集

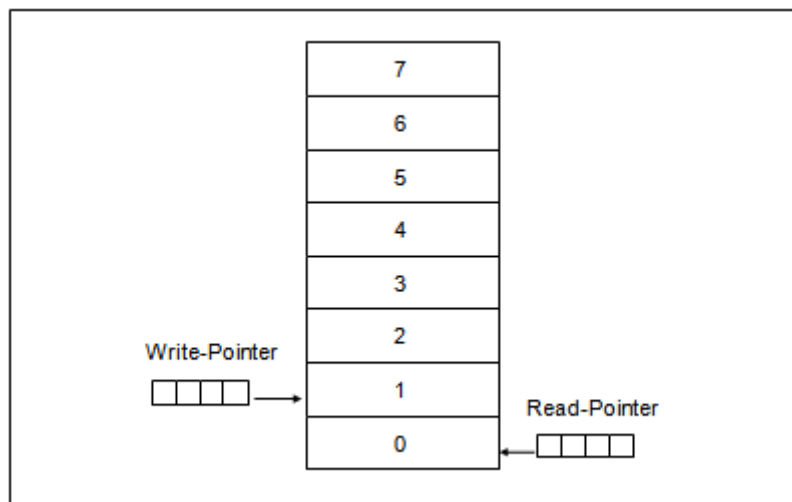


图 35 传输 FIFO 缓冲区概观 (以 8 字节 FIFO 为例)

当 CPU 要将数据写入传送 FIFO 缓冲区, 会将数据重复写入一个特定缓存器, 即 USB\_TXDAT, 每写入一个字节的数, 则写入指针(Write-Point) 会自动加 1; 而当每读取一个字节的数时, 写入指针(Write-Point) 会自动减 1。读取指标(Read-pointer) 则指向下一个 SIL 准备读取之数据字节, 当 SIL 读取一个字节的数, 则读取指针(Read-pointer) 会自动加 1。传送 FIFO 缓冲区在内容为空或者数据写入未完成时会禁止 SIL 进行数据的读取。

#### 9.4.3.2 传输数据集管理

当缓存器 USB\_TXFLG.TXFULL = 1 代表数据已经完成写入至传送 FIFO 缓冲区且可以进行传送; 当发生复位时, USB\_TXFLG.TXFULL = 0 且 USB\_TXFLG.TXEMP = 1, 此时代表传送 FIFO 缓冲区内无任何数据; 在此例中(即使用 8 字节 FIFO 为例), 当要传送的数据大于 8 个字节时, 则只有最前面的 8 个字节会被写至传送 FIFO 缓冲区内, 此时 USB\_TXFLG.TXFULL 尚不等于 1, 直到设置 TXCNT = 8 之后, USB\_TXFLG.TXFULL 才会为 1, 在此需注意的是 USB\_TXCNT 的内容值决定了有多少字节的数据会传送至 USB 讯号在线, 当写入传送 FIFO 缓冲区内数据数量若与填入 USB\_TXCNT 内的数量不同的话, 会发生不可预期的结果。此外, 当传送 FIFO 缓冲区为空或者是 USB\_TXFLG.TXFULL = 0 时, 传送 FIFO 缓冲区是禁止读取的。

以下两个事件会导致 USB\_TXFLG.TXFULL 的值进行更新:

- 新的数据集写入至传输 FIFO 缓冲区: 当 CPU 写入几个字节之数据至缓存器 USB\_TXDAT, 亦将字节的数量写入至缓存器 USB\_TXCNT, 则只有在设置了缓存器 USB\_TXCNT 之后 USB\_TXFLG.TXFULL 会被设置为 1; 若将 0 填入 USB\_TXCNT 之中, 则代表一组长度为 0 的数据传送出去, 此时 USB\_TXFLG.TXFULL 依然会被设置为 1, 同时 USB\_TXFLG.TXEMP 则无改变, 依然指出传送 FIFO 缓冲区内为空, 请参照下面表格 22
- 成功传输 FIFO 中的数据集: SIL 从传送 FIFO 缓冲区内读取数据并传送出现, 此时若传送成功且收到 Host 所下的 Ack 封包, 则 USB\_TXFLG.TXFULL 会被清为 0 且同时 TXFLG.TXEMP 会被设为 1。

表格 21 写入字节计数缓存器

TXFULL	TXEMP	零长度之传输	写入数据集至 TXDAT	数据集写入	TXFULL	TXEMP
0	0	No		Yes	1	0
0	1	No		Yes	1	0
0	1	Yes	写入数量至 TXCNT	No	1	1
1	-	-		写入忽略	1	-

当数据正常被传送完成, 读取指标 (Read\_Pointer) 与写入指标 (Write\_Pointer) 皆会被设置至传送 FIFO 缓冲区的起始位置, 以准备进行下一笔数据; 反之, 当传送过程发生问题, 此时读取指标 (Read\_Pointer) 会回到传送 FIFO 缓冲区的起始位置, 同时 SIL 会重新读取此笔数据以进行重送。数据指针完全是由硬件进行控制, 表格 23 列出传送完成后依据 USB\_TXSTAT.TXERR 与 USB\_TXSTAT.TXACK 的不同, 读取指标 (Read\_Pointer) 与写入指标 (Write\_Pointer) 有何相对应的动作。

表格 22 传输 FIFO 管理真值表

TXERR	TXACK	传输完成后之动作
0	0	无动作
0	1	读取指标 (Read_Pointer) 与写入指标 (Write_Pointer) 皆会被设置至传送 FIFO 缓冲区的起始位置
1	0	读取指标 (Read_Pointer) 会被设置至传送 FIFO 缓冲区的起始位置

#### 9.4.4 接收 FIFOs

##### 9.4.4.1 接收 FIFOs 概述

接收 FIFO 是具有以下功能的数据缓冲区 (参见图 36):

- 支持一个不大于 8/16/32/64 字节的数据集
- 一个字节计数缓存器，用来储存数据集中的字节数量
- 具相关之旗标以告知此时接收 FIFO 缓冲区已满或为和空
- 当接收因故未完成，能自动重新接收最后一个数据集

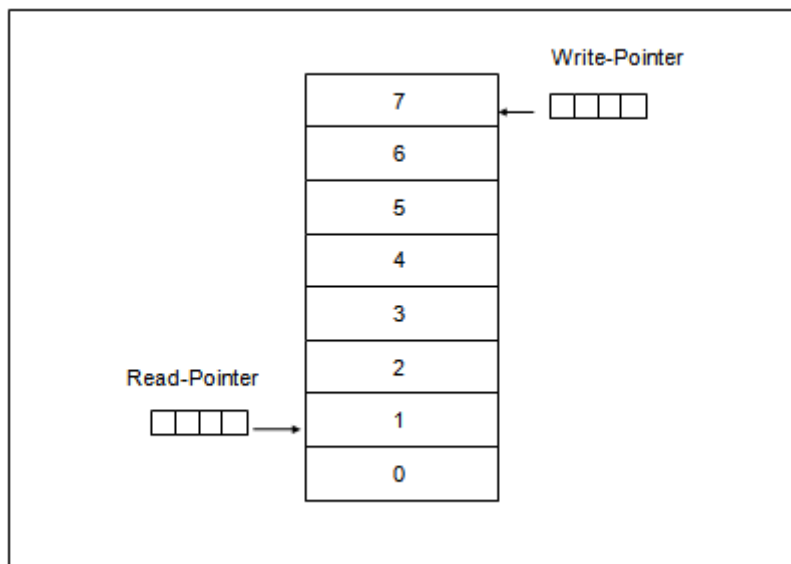


图 36 接收 FIFO 概观 (以 8 字节 FIFO 为例)



当 SIL 将数据写入接收 FIFO 缓冲区，此时每写入一个字节，则写入指针 (Write-Point) 会自动加 1；而当每读取一个字节的的数据时，写入指针 (Write-Point) 会自动减 1。读取指标 (Read-pointer) 则指向下一个 CPU 准备读取之数据字节，当 CPU 读取一个字节的的数据，则读取指针 (Read-pointer) 会自动加 1。接收 FIFO 缓冲区在内容为空或者数据写入未完成时会禁止 CPU 进行数据的读取。当 SIL 侦测到收到的数据是属于 SETUP 数据封包时，SIL 会自动清除接收 FIFO 缓冲区并开始将 SETUP 数据内容写入至接收 FIFO 缓冲区内，即使此时 CPU 正在读取接收 FIFO 缓冲区内的数据。

#### 9.4.4.2 接收数据集管理

当缓存器 USB\_RXFLG.RXFULL = 1 代表数据已经完成写入至接收 FIFO 缓冲区且可以进行读取；当发生复位时，USB\_RXFLG.RXFULL = 0 且 USB\_RXFLG.RXEMP = 1，此时代表接收 FIFO 缓冲区内无任何数据；在此例中(即使用 8 字节 FIFO 为例)，当要接收的数据大于 8 个字节时，则只有最前面的 8 个字节会被写至接收 FIFO 缓冲区内，此时 RXFLG.RXFULL 尚不等于 1，直到此笔数据接收完成且响应 HOST ACK 之后，USB\_RXFLG.RXFULL 才会为 1。当 CPU 成功从接收 FIFO 缓冲区内将数据读出之后，需透过 Firmware 将 USB\_RXCON. RXFFRC 设置为 1 来清除将 USB\_RXFLG.RXFULL 清为 0。在此需注意的是 CPU 在开始读取接收 FIFO 缓冲区之前，要先读取缓存器 USB\_RXCNT 的值来确定接收 FIFO 缓冲区内有多少字节的数据，当 CPU 读取超过接收 FIFO 缓冲区内所含的数据时，所读取到超过的部份是无意义的。

**表格 23 Status of the Receive FIFO Data Set**

0	0	SIL 正在将数据写入接收 FIFO 缓冲区
RXFULL	RXEMP	状态
0	1	接收 FIFO 缓冲区无数据在其中
1	0	SIL 已经将数据写入接收 FIFO 缓冲区
1	1	接收到长度为 0 之数据封包

当数据正常被接收完成，且 CPU 完成读取之后，Firmware 需将 USB\_RXCON. FFRC 设置为 1 来将写入指标 (Write\_Pointer) 与读取指标 (Read\_Pointer) 设置为接收 FIFO 缓冲区的起始位置，以准备进行下一笔数据；当接收过程发生问题，写入指标 (Write\_Pointer) 将回到接收 FIFO 缓冲区的起始位置，同时 SIL 会自动重新接收此笔数据并将此笔数据再次写入接收 FIFO 缓冲区。数据指针完全是由硬件进行控制，表格 25 列出接收完成后依据 USB\_RXSTAT.RXERR 与 USB\_RXSTAT.RXACK 的不同，读取指标 (Read\_Pointer) 与写入指标 (Write\_Pointer) 有何相对应的动作。

**表格 24 接收 FIFO 管理的真值表**

RXERR	RXACK	传输完成后之动作
0	0	无操作
0	1	当 Firmware 将 RXCON. FFRC 设置为 1，读取指标 (Read_Pointer) 与写入指标 (Write_Pointer) 皆会被设置至传送 FIFO 缓冲区的起始位置
1	0	写入指标 (Write_Pointer) 会被设置至接收 FIFO 缓冲区的起始位置

#### 9.4.5 Setup Token 接收 FIFO 缓冲区处理

主机端会从端点 0 下 SETUP token，收到之后必须响应 ACK 即使此时的接收 FIFO 缓冲区内尚有数据。当 SIL 侦测到 SETUP token，SIL 会清除接收 FIFO 缓冲区，同时将 USB\_RXSTAT. STOVW 设置为 1 以清除与锁定接收 FIFO 缓冲区之读取指标 (Read\_Pointer)，这是为了防止当 CPU 正在读取接收 FIFO 缓冲区时，因为 SIL 清除接收 FIFO 缓冲区造成 USB\_RXFLG. RXURF 以及读取指标 (Read\_Pointer) 会被进行设置；当 SETUP 封包数据成功接收且响应 ACK 之后，RXSTAT. STOVW 会被自动清除，同时 USB\_RXSTAT. EDOVW 会被设置为 1，此时接收 FIFO 缓冲区之读取指标 (Read\_Pointer) 依然是被锁定的，直到 USB\_RXSTAT. STOVW 与 USB\_RXSTAT. USB\_EDOVW 皆为 0 为止，Firmware 在读取 SETUP 封包数据之前，必须先清除 USB\_RXSTAT.EDOVW，否则将无法从接收 FIFO 缓冲区读取数据；在处理完 SETUP 封包之后，Firmware 应该要先检查 USB\_RXSTAT. STOVW 与 USB\_RXSTAT. EDOVW 时状态之后再行 USB\_RXCON. RXFFRC 设置，这是因为当 SETUP 封包已经在 FIFO 之内或是正在接收的时候，此时 USB\_RXSTAT. STOVW 与 USB\_RXSTAT. EDOVW 两者有任何一个不为 0，此时对 USB\_RXCON. RXFFRC 进行设置是无效的。

#### 9.4.6 暂停与恢复 (Suspend and Resume)

为了降低电源消耗，当 CPU 发现 USB 讯号超过 3ms 没有任何动作时会自动进入暂停(Suspend)状态，在此状态中，CPU 以及所有的外围皆处于低耗电(Power down)模式，此时会开启中断以致能恢复(Resume)唤醒的功能(亦可同时致能远程唤醒(Remote wakeup)的功能)，且整个 CPU 的耗电会降至 500μA 以下。

在暂停(Suspend)模式下，当 USB 讯号有动作时，WT32L064/032 会离开暂停(Suspend)模式。一般 USB 装置同时亦可以产生 USB 唤醒讯号来要求 HOST 离开暂停(Suspend)模式或选择性的暂停(Suspend)模式，此即为远程唤醒之功能，此功能可以选择是否开启，而 CPU 亦可允许 HOST 来致能或禁能此功能(远程唤醒)，下面图 37 描述了这些电源状态。

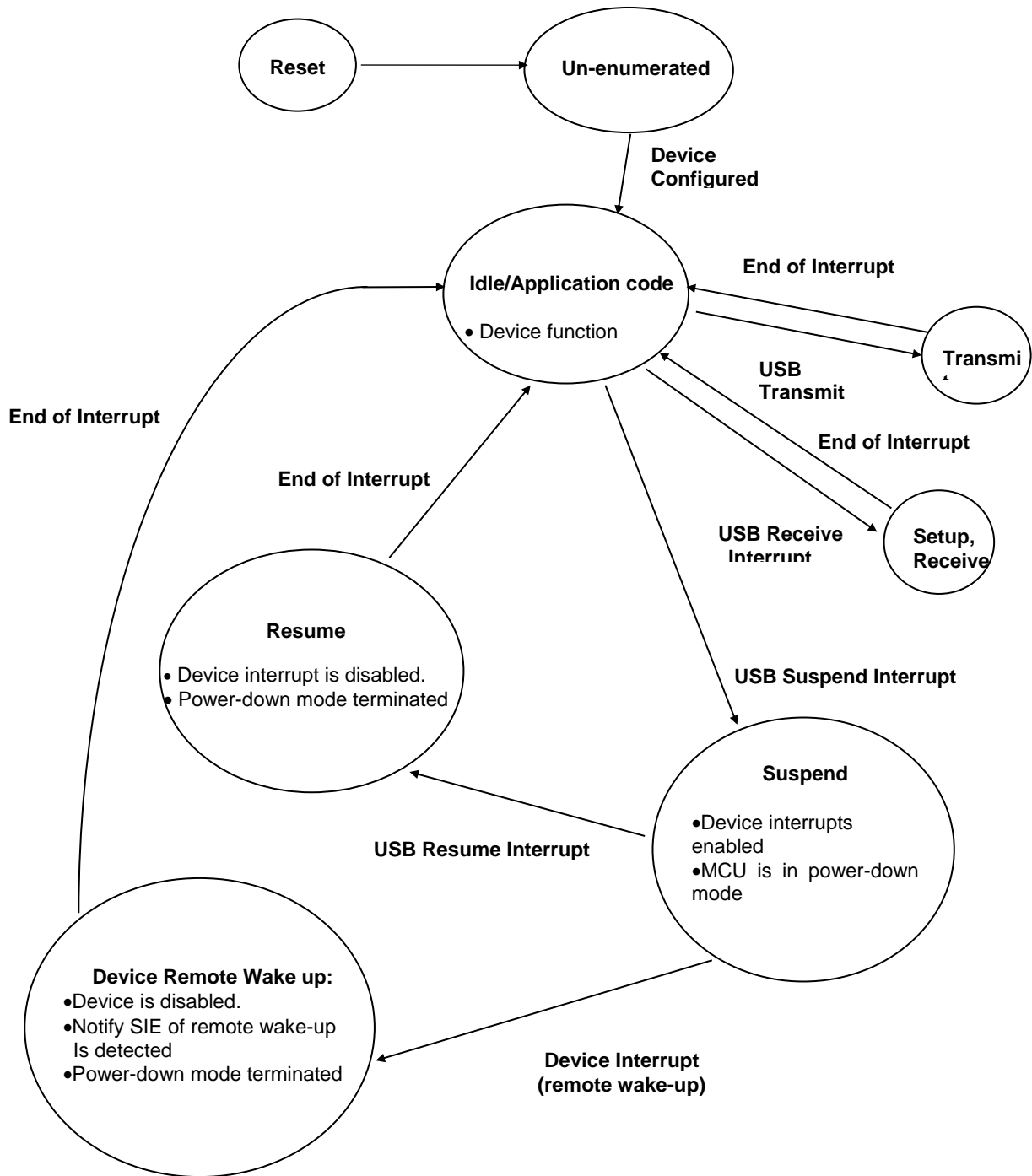


图 37 暂停 (Suspend)和恢复 (Resume)状态图

**9.4.7 FIFO 内存地址映像**

在初始状态下，每个 Endpoint 都有其自己的 FIFO 内存空间，如果未使用某些端点时，则可以通过正确程序设置 DBCON 寄存器的 DBBANK 字段，将其内存空间重新用作其它支持双缓冲区端点的第二个缓冲区。

**表格 25 FIFO 内存地址映像**

bank	address	default EP0~6 and EP9 buffer space	Default EP7 & EP8 Isochronous buffer space
0	000h ~ 03fh	EP0_TX	
1	040h ~ 07fh	EP0_RX	
2	080h ~ 0bfh	EP1	
3	0c0h ~ 0ffh	EP2	
4	100h ~ 13fh	EP3	
5	140h ~ 17fh	EP4	
6	180h ~ 1bfh	EP5	
7	1c0h ~ 1ffh	EP6	
8	200h ~ 23fh		EP7
9	240h ~ 27fh		EP7
10	280h ~ 2bfh		EP7
11	2c0h ~ 2ffh		EP7
12	300h ~ 33fh		EP8
13	340h ~ 37fh		EP8
14	380h ~ 3bfh		EP8
15	3c0h ~ 3ffh		EP8
16	400h ~ 43fh		EP7
17	440h ~ 47fh		EP7
18	480h ~ 4bfh		EP7
19	4c0h ~ 4ffh		EP7
20	500h ~ 53fh		EP8
21	540h ~ 57fh		EP8
22	580h ~ 5bfh		EP8
23	5c0h ~ 5ffh		EP8
24	600h ~ 63fh	EP9	

 $start\_addr(bank) = 64 * bank$ 
 $start\_addr\_1^{st}(EP7) = 200h + EP7\_BASE, start\_addr\_1^{st}(EP8) = 200h + EP8\_BASE$ 
 $start\_addr\_2^{nd}(EP7) = 400h + EP7\_BASE, start\_addr\_2^{nd}(EP8) = 400h + EP8\_BASE$

## 10 时钟还原系统

### 10.1 主要概述

时钟恢复系统 (CRS) 是一种先进的数字控制器，用于内部精细可调节(trimming)RC振荡器HSI48。CRS基于与可选同步信号的比较，为振荡器输出频率评估提供了有力的手段。它能够根据测量的频率误差值自动调整振荡器调节(trimming)，同时保持手动调节(trimming)的可能性。

CRS 非常适合为 USB 外围提供精确的时钟。在这种情况下，同步信号可以从 USB 总线上的帧启动(start-of-frame, SOF) 的数据报与产生信号化(signalization)，该信号指令由 USB 主机以精确的 1 毫秒间隔发送。

同步信号也可以从LSE振荡器输出或外部引脚产生，也可以由用户软件产生。

#### 主要功能

- 可选择同步源，具有可程序设定的预除器(prescaler)和极性：
  - External pin
  - LSE oscillator output
  - USB SOF packet reception
- 通过软件产生同步脉冲的可能性
- 自动振荡器调节(trimming)功能，无需 CPU 操作
- 手动控制选项，可加快启动收敛
- 16 位频率误差计数器，具有自动错误值捕获和重载
- 用于自动频率误差值评估和状态报表的可程序设定限制
- 可屏蔽的中断与事件：
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

10.2 方块图

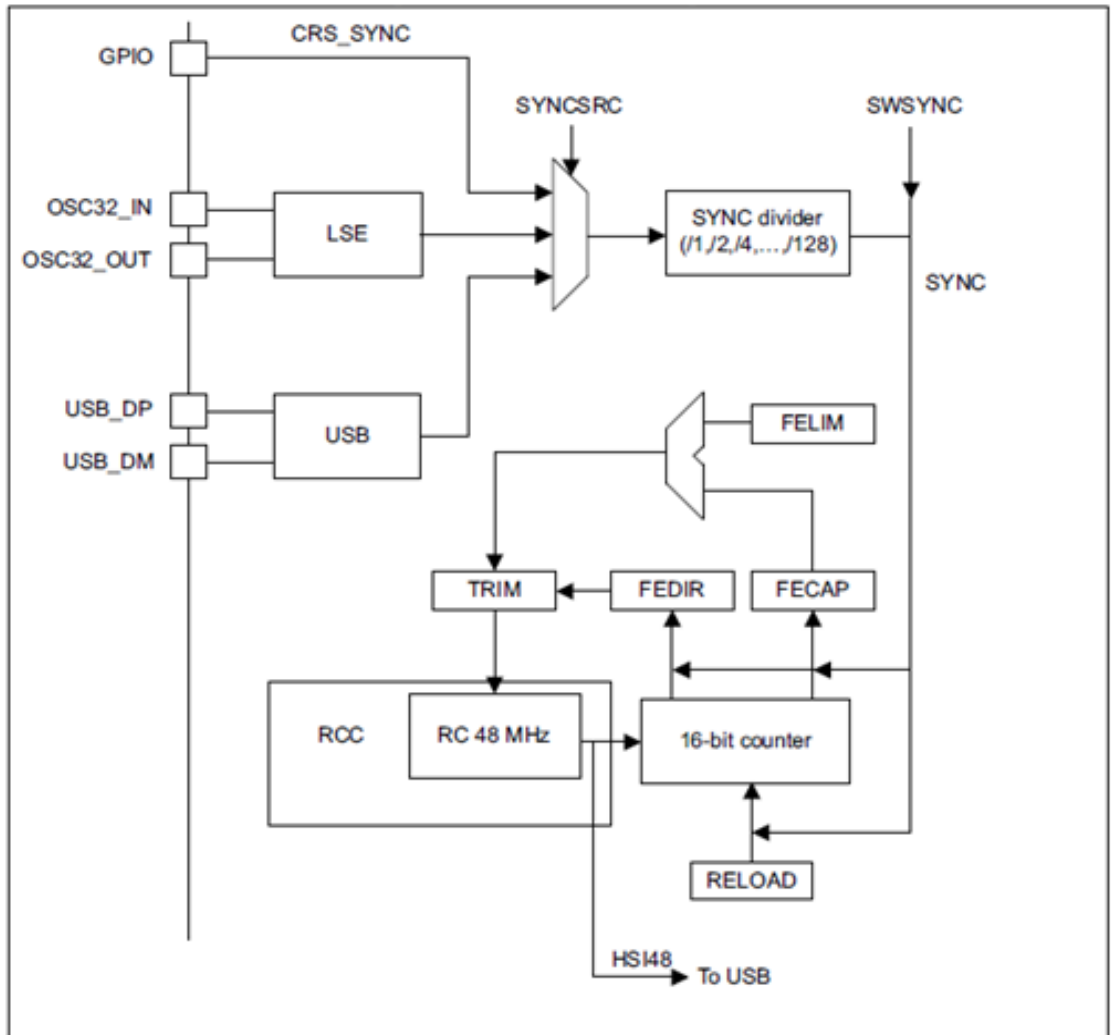


图 38 CRS 方块图

**10.3 缓存器描述**

表格 26 CRS 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>CRS_CR: CRS control register</b>					
00h	32:14	-	-	-	保留
	13:8	R/W	20h	TRIM	HSI48 振荡器平滑调节(trimming)。当 AUTOTRIMEN=1 时, 此缓存器是只读的
	7	R/W	0	SWSYNC	产生软件 SYNC 事件
	6	R/W	0	AUTOTRIMEN	Automatic trimming 致能
	5	R/W	0	CEN	频率误差计数器启动
	4	-	-	-	保留
	3	R/W	0	ESYNCIE	预期同步中断启动
	2	R/W	0	ERRIE	Synchronization or trimming 错误中断启动
	1	R/W	0	SYNCWARNIE	同步 SYNC 警告中断启动
0	R/W	0	SYNCOKIE	SYNC 事件确定中断启动	
<b>CRS_CFGT: CRS configuration register</b>					
04h*	31	R/W	0	SYNCPOL	SYNC 极性选择。 0: 同步在上升沿启动 (预设)。 1: 同步在下降沿启动。
	30	-	-	-	保留
	29:28	R/W	2	SYNCSRC	SYNC 信号源选择。 0: GPIO 被选为 SYNC 信号源。 1: 选择 LSE 作为 SYNC 信号源。 2: USB SOF 选择为 SYNC 信号源 (默认)。 3: 保留。
	27	-	-	-	保留
	26:24	R/W	0	SYNCDIV	SYNC 除频 (divider) 0: SYNC not divided (default) 1: SYNC divided by 2 2: SYNC divided by 4 3: SYNC divided by 8 4: SYNC divided by 16 5: SYNC divided by 32 6: SYNC divided by 64 7: SYNC divided by 128
	23:16	R/W	22h	FELIM	Frequency error limit
	15:0	R/W	bb7fh	RELOAD	Counter reload value
<b>CRS_ISR: CRS interrupt/status register</b>					
08h	31:16	R	0	FECAP	Frequency error capture
	15	R	0	FEDIR	Frequency error direction
	14:11	-	-	-	保留
	10	R	0	TRIMOVF	Trimming overflow or underflow
	9	R	0	SYNCMISS	SYNC missed
	8	R	0	SYNCERR	SYNC error
	7:4	-	-	-	保留
	3	R	0	ESYNCF	Expected SYNC 旗标 (flag)
	2	R	0	ERRF	Error 旗标 (flag)
1	R	0	SYNCWARNF	SYNC warning 旗标 (flag)	

Index	Bit	R/W	Default	Name	Description
	0	R	0	SYNCOKF	SYNC event OK 旗标 (flag)
<b>CRS_ICR: CRS clear flag register</b>					
0ch	31:4	-	-	-	保留
	3	W	0	ESYNCC	预期同步 SYNC 清除旗标
	2	W	0	ERRC	Error clear 旗标 (flag)
	1	W	0	SYNCWARNC	SYNC 警告清除旗标
	0	W	0	SYNCOKC	SYNC 事件确定清除旗标

\* 注意:

1. TRIM register is read-only when AUTOTRIMEN=1
2. CRS\_CFG register is read-only when CEN=1



## 10.4 功能描述

### 10.4.1 同步输入

CRS 同步 (SYNC) 源可利用 CRS\_CFGR 寄存器选择, 可以是来自外部 CRS\_SYNC 引脚、LSE 时钟或 USB SOF 信号的信号。为了更好的 SYNC 输入的强健性(robustness), 采用了一个简单的数字滤波器 (由 HSI48 时钟采样), 以过滤掉任何故障。该源信号也具有可规划的极性, 然后可以通过可程序设定二进制预除器(prescaler), 以获得适当频率范围 (通常约为 1 kHz)的同步信号。还可以通过软件通过 CRS\_CR 寄存器中设置 SWSYNC 位来产生同步事件。

### 10.4.2 频率错误量测

频率误差计数器是一个 16 位向下/向上计数器, 在每个 SYNC 事件上重载 RELOAD 值。它开始倒计时, 直到达到零值, 其中产生 ESYNC (预期同步)事件。然后, 它开始计数到 OUTRANGE 限制, 最终停止 (如果没有 SYNC 事件)并产生 SYNCMISS 事件。OUTRANGE 限制定义为频率误差限制 (CRS\_CFGR 寄存器的 FELIM 字段)乘以 128。

检测到 SYNC 事件时, 频率误差计数器及其计数方向的实际值存储在 CRS\_ISR 寄存器的 FECAP (频率误差捕获)字段和 FEDIR (频率误差方向)位中。当在向下计数阶段 (达到零值之前)检测到 SYNC 事件时, 这意味着实际频率低于目标 (因此, TRIM 值应增加), 而在向上计数阶段检测到该值时, 则表示实际频率应增加表示实际频率较高 (并且 TRIM 值应递减)。

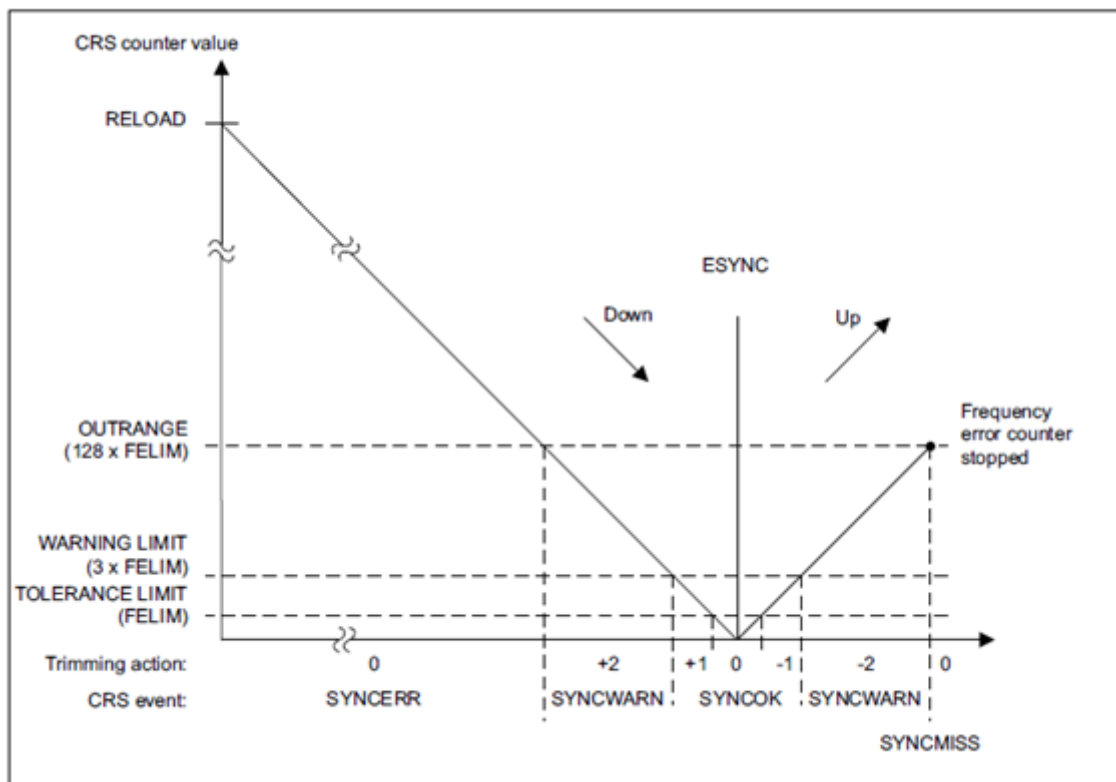


图 39 CRS Counter Behavior

### 10.4.3 频率误差评估和自动修补

测量的频率误差通过将其值与一组限制进行比较来评估:

- 在 CRS\_CFGR 缓存器的 FELIM 字段中直接给出的容差限制
- 警告限制, 定义为  $3 * \text{FELIM}$  值
- OUTRANGE (误差限制), 定义为  $128 * \text{FELIM}$  值

此比较的结果用于产生状态指示, 并用于通过在 CRS\_CR 缓存器中设置 AUTOTRIMEN 位来控制启动的自动修整:

- 当频率误差低于容差限制时, 这意味着 TRIM 字段中的实际调节(trimming)值是最佳值, 因此无需执行调节(trimming)操作。
  - SYNCOK status indicated
  - 在 AUTOTRIM 模式下未更改 TRIM 值
- 当频率误差低于警告限制但高于或等于容差限制时, 这意味着需要执行一些调节(trimming)操作, 但一个调节(trimming)步骤的调整足以达到最佳 TRIM 值。
  - SYNCOK status indicated
  - 在自动 TRIM 模式下通过一个调节(trimming)步骤调整 TRIM 值
- 当频率误差高于或等于警告限制但低于误差限制时, 这意味着需要采取更强烈的调节(trimming)操作, 并且下一个期间可能会达到最佳 TRIM 值。
  - SYNCWARN status indicated
  - 在 AUTOTRIM 模式下通过两个调节(trimming)步骤调整 TRIM 值
- 当频率误差高于或等于误差限制时, 则表示频率超出 TRIM 范围。当 SYNC 输入不干净或缺少某些 SYNC 脉冲时 (例如, 当一个 USB SOF 损坏时), 也会发生这种情况。
  - SYNCERR or SYNCMISS status indicated
  - 在 AUTOTRIM 模式下未更改 TRIM 值

注意: 如果 TRIM 字段的实际值非常接近其限制, 以至于 AUTOTRIM 将强制其溢出或下溢, 则 TRIM 值仅设置为限制, 并指示 TRIMOVF 状态。

在 AUTOTRIM 模式 (在 CRS\_CR 缓存器中设置自动位)中, CRS\_CR 的 TRIM 字段由硬件调整, 并且是只读的。

### 10.4.4 CRS 初始化和规划

#### RELOAD value

应根据预缩放后目标频率与同步源频率之间的比率选择 RELOAD 值。然后, 它减少 1, 以达到零值上的预期同步。公式如下:

$$\text{RELOAD} = (\text{fTARGET} / \text{fSYNC}) - 1 \text{ RELOAD}$$

字段的复位值对应于目标频率 48 MHz 和同步信号频率为 1 kHz (来自 USB 的 SOF 信号)。

**FELIM value**

FELIM 值的选择与 HSI48 振荡器特性及其典型的调节(trimming)步长大小密切相关。最佳值对应于调节节距(trimming step)大小的一半, 以 HSI48 振荡器时钟刻度数表示。可以使用以下公式:

$$FELIM = (fTARGET / fSYNC) * STEP[\%] / 100\% / 2$$

结果应始终四舍五入到最接近的整数值, 以便获得最佳调节(trimming)响应。如果应用程序中不需要频繁调节(trimming)操作, 则调节滞(trimming hysteresis)后可以通过略微增加 FELIM 值而增加。FELIM 字段的复位值对应于  $(fTARGET / fSYNC) = 48000$  和典型调节节距(trimming step)大小 0.14%。

**警告:** RELOAD 和 FELIM 字段规划错误, 无法提供硬件保护, 从而导致调节(trimming)回应不稳定。预期操作模式需要正确设置 RELOAD 值 (根据同步源频率), 该值也大于 128 = FELIM 值 (OUTRANGE 限制)。

**10.4.5 CRS 省电模式 (low-power modes)**

表格 27 低功耗模式对 CRS 的影响

Mode	Description
Sleep	无效. CRS interrupts cause the device to exit the Sleep mode.
Stop	<b>CRS 缓存器已冻结。</b>
Standby	<b>在停止或待机模式退出和 HSI48 振荡器重新启动之前, CRS 停止运行。</b>

**10.4.6 CRS 中断 (interrupts)**

表格 28 CRS 中断控制位

Interrupt event	Event 旗标(flag)	致能 control bit	Clear 旗标(flag) bit
Expected synchronization	ESYNCF	ESYNCE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization 或 trimming 错误 (TRIMOVF、SYNCMIS、SYNCERR)	ERRF	ERRIE	ERRC

## 11 I<sup>2</sup>C

### 11.1 主要概述

I<sup>2</sup>C (Inter-Integrated Circuit)总线接口做为微控制器和串行 I<sup>2</sup>C 总线之间的接口。它提供多主机功能，并控制所有 I<sup>2</sup>C 总线特定顺序、协议、仲裁和时序。它支持标准和快速模式。

- 多主机功能: 同一接口可以充当主机接口或从机接口
- I<sup>2</sup>C Master features:
  - Clock generation
  - Start and Stop generation
- I<sup>2</sup>C Slave features:
  - 可程序设定 I<sup>2</sup>C 地址检测
  - 双地址功能, 可确认 2 个从机地址
  - 停止位检测
- 生成和检测 7bit/10 bit 寻址和常规呼叫
- 支持不同的通信速度:
  - 标准速度模式 (高达 100 kHz)
  - 快速模式 (高达 400 kHz)
  - 快速模式 Plus (高达 1MHz)
- Status 旗标(flags):
  - 传送器/接收器模式旗标
  - End-of-Byte 传输旗标(flag)
  - I<sup>2</sup>C 忙碌旗标(flag)
  - 主接收器返回 ACK (NACK) 旗标。
- 错误旗标:
  - 主机模式的仲裁丢失条件
  - 如果 SCL 在 Smbus 模式下保持 Low 25 ms (超时)
  - 检测错位的启动或停止条件
  - 如果禁用 clock stretching, 则溢出或不足
- 具有各种源的中断向量:
  - 地址/数据通信的中断
  - 错误/警示状况的中断
- 可选择的时钟延展
- 1-byte buffer with DMA capability
- 可配置 CRC (循环冗余检查)产生
  - CRC 值可以在 Tx 模式下作为最后一个字节传输
  - CRC 错误检查上次接收的字节

11.2 方块图

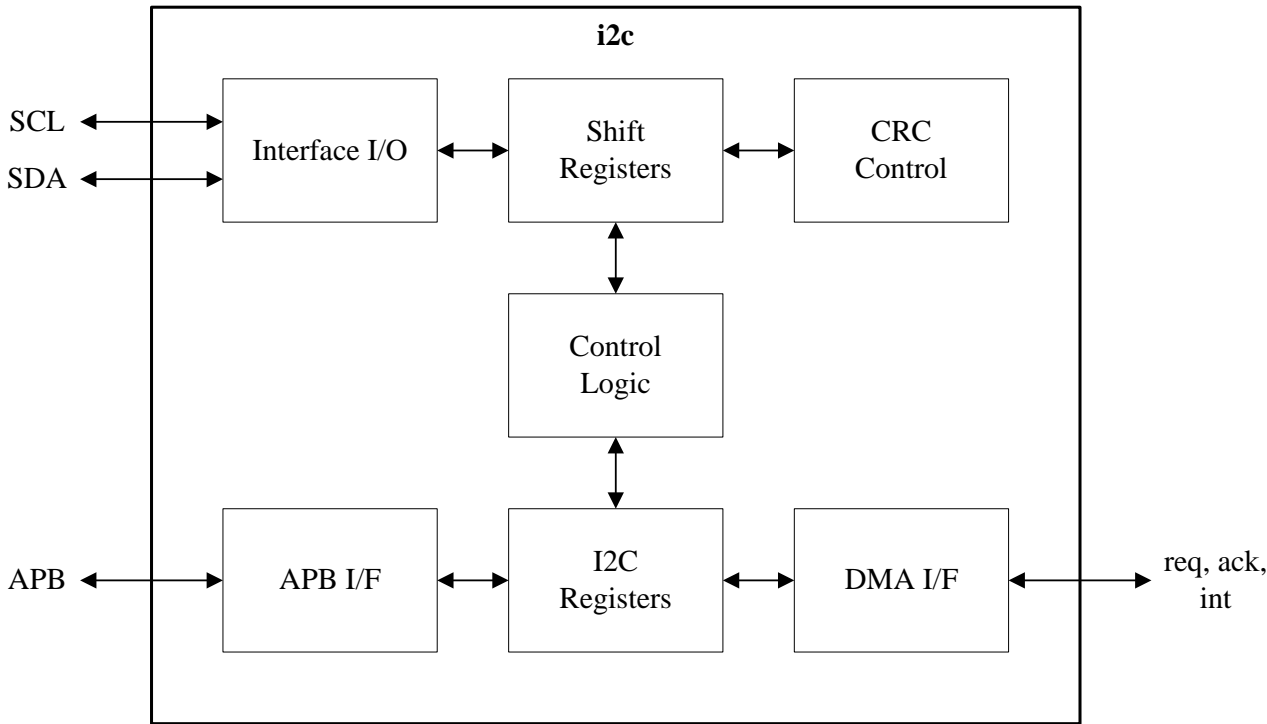


图 40 I<sup>2</sup>C 方块图

11.3 缓存器描述

表格 29 I<sup>2</sup>C 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>I2C_CR1: I<sup>2</sup>C control register 1</b>					
00h	31	R/W	0	I2cen	I2C 致能设置 0: 禁能 I <sup>2</sup> C 功能 1: 致能 I <sup>2</sup> C 功能.
	30:23	-	-	-	保留
	22	R/W	0	mnackopt	主机模式收到 NACK 时的动作选项设置 0: 自动停止 1: 继续 TX
	21	R/W	0	snackopt	从机模式传输数据收到 NACK 时的动作选项设置 0: 停止 RX 1: 继续 RX
	20	R/W	0	nosarb	从机模式仲裁设置 0: 有仲裁 1: 无仲裁
	19:17	-	-	-	保留
	16	R/W	0		0: 为 I <sup>2</sup> C 应用选择 CMOS I/O 级别。
	13	R/W	0	smbhen	SMBus Host address (0001000x) 设置

Index	Bit	R/W	Default	Name	Description
					0: 禁能. 1: 致能.
	12	R/W	0	gcen	General call address (00000000) 设置 0: 禁能. 1: 致能.
	11	R/W	0	NOSTRETCH	时钟延展设置 (在从机模式下) 0: 时钟延展致能 1: 时钟延展禁能 通过将 SCL 线拉低来暂停一个传输.直到释放 SCL 线为高电位,传输才继续进行.
	10	R/W	0	sbc	从机字节控件 0: 禁能 1: 致能
	9	R/W	0	rxdmaen	DMA 接收请求设置 0: 禁能. 1: 致能.
	8	R/W	0	txdmaen	DMA 传输请求设置 0: 禁能 1: 致能
	7	R/W	0	errie	错误中断设置 0: 禁能 1: 致能 注意: 这些错误中的任何一个都会产生一个中断:arlo[0], berr[0], ovr[0], 超时[0], pecerr[0], 警报[0].
	6	R/W	0	tcie	传输完整中断设置 0: 禁能. 1: 启动. 注意: 这些事件中的任何一个都会产生中断:tc[0], tcr[0]
	5	R/W	0	stopie	停止检测中断设置 0: 禁能. 1: 启动.
	4	R/W	0	nackie	未确认已接收的中断设置 0: 禁能. 1: 启动.
	3	R/W	0	addrie	地址匹配中断设置 0: 禁能. 1: 启动.
	2	R/W	0	rxie	RX 中断(rxne[0]) 设置 0: 禁能. 1: 启动.
	1	R/W	0	txie	TX 中断(txis[0]) 设置 0: 禁能. 1: 启动.
	0	-	-	-	保留
<b>I2C_OAR1: I<sup>2</sup>C addr mode and oa1</b>					
04h	31:12	-	-	-	保留

Index	Bit	R/W	Default	Name	Description
	11	R/W	0	addr_mode	0: 7-bit 地址模式 1: 10-bit 地址模式
	10:1	R/W	0	oa1	oa1[9:0] for 10-bit 地址 oa1[6:0] for 7-bit 地址
	0	R/W	0	oa1_en	oa1 致能/禁能
<b>I2C_OAR2: I<sup>2</sup>C oa1</b>					
08h	31:11	-	-	-	保留
	10:8	R/W	0	oa2msk	oa2 masks 000: no mask, all oa2[6:0] 比较 001: only oa2[6:1] 比较 010: only oa2[6:2] 比较 011: only oa2[6:3] 比较 100: only oa2[6:4] 比较 101: only oa2[6:5] 比较 110: only oa2[6] 比较 111: all (除了保留) 7-bit 收到的地址确认
	7:1	R/W	0	oa2	7-bit 地址
	0	R/W	0	oa2_en	oa2 致能/禁能
<b>I2C_TXDR: I<sup>2</sup>C 8-bit transmit data</b>					
0ch	31:8	-	-	-	保留
	7:0	R/W	0	tx_data	8-bit 传送数据的缓冲区
<b>I2C_RXDR: I<sup>2</sup>C 8-bit receive data</b>					
10h	31:8	-	-	-	保留
	7:0	R	0	rx_data	8-bit 接收数据的缓冲区
<b>I2C_SADDR: I2C saddr</b>					
14h	31:12	-	-	-	保留
	11	R/W	0	add10	0: 7-bit 地址模式 1: 10-bit 地址模式
	10:1	R/W	0	saad	saad[9:0] for 10-bit 地址 saad[6:0] for 7-bit 地址
	0	R/W	0	rd_wrn	0: 写入 1: 读取
<b>I2C_BCNT: I2C counter setting</b>					
18h	31:10	-	-	-	保留
	9	R/W	0	auto_end	自动结束模式 (主机模式) 0: 软件结束模式:在传输 NBYTES 数据时设置 tc[0] 旗标, 将 SCL 延展到低准位 1: 自动结束模式:当 NBYTES[7:0] 数据传输时, 自动发送停止条件 注意: 此位在从机模式下或 reload [0]位时不起作用
	8	R/W	0	reload	NBYTES 重载模式 0: 传输在 NBYTES[7:0] 数据传输后完成 (停止或 RESTART 将随后完成) 1: 在 NBYTES[7:0] 数据传输 (NBYTES[7:0] 将重载)后, 传输未完成。Tcr[0] 旗标是在传输 NBYTES 数据时设置的, 将 SCL 延展到低准位
	7:0	R/W	0	NBYTES	字节数 传输/接收的字节数在此处设定。此字段在 sbc[0]=0

Index	Bit	R/W	Default	Name	Description
					的从机模式下不关心
<b>I2C_CR2: I2C control register 2</b>					
1ch	31:4	-	-	-	保留
	3	R/W	0	swrst	软件重置 I2C 模块
	2	R/W	0	nack	NACK 产生 (从机模式) 此位由软件设置, 在发送 NACK 时或收到停止条件或匹配地址时由硬件清除 0: ACK 在当前接收字节后发送 1: 当前接收字节后发送 NACK 注意: 将“0”写入此位不起作用
	1	R/W	0	stop	“停止”设置 (主机模式) 此位由软件设置, 在检测到停止条件时由硬件清除 0: 无“停止”产生 1: 在当前字节传输后产生“停止”
0	R/W	0	start	“开始”设置 此位由软件设置, 在发送“开始”后由硬件清除 0: 无“开始”产生 1: “开始”/“重新开始”产生	
<b>I2C_ISR: I2C status and control</b>					
20h	31:16	-	-	-	保留
	15	R/W1c	0	busy	总线繁忙 此标志表示总线上正在进行通信。当检测到“开始”条件时, 由硬件设置。当检测到“停止”条件时, 硬件会清除它
	14	-	-	-	保留
	12	R/W1c	0	timeout	超时或 $t_{low}$ 检测标志 当发生超时或延展时钟超时时, 硬件会设置此标志
	11	-	-	-	保留
	10	R/W1c	0	ovr	超出/不足 (从机模式, Overrun/Underrun)。 1: 超出或不足 0: 无超出/不足
	9	R/W1c	0	arlo	仲裁丢失 1: 检测到仲裁丢失 0: 未检测到仲裁丢失
	8	R/W1c	0	berr	总线错误 1: 错位的“开始”或“停止”条件 0: 无错位的“开始”或“停止”条件
	7	R/W	0	tcr	传输完成重载 此标志由硬件在 reload[0]=1 和 NBYTES[7:0] 数据传输时设置。当 NBYTES 写入非零值时, 由软件清除
	6	R/W	0	tc	传输完成 (主机模式) 此标志由硬件设置, 当 reload[0]=0, auto_end[0]=0 和 NBYTES[7:0] 数据已传输 当设置“开使”位或“停止”位时, 由软件清除它
5	R/W1c	0	stopf	停止条件标志 当在总线上检测到停止条件且外围设备参与此传输时, 此标志由硬件设置: - 如果停止条件由外围产	



Index	Bit	R/W	Default	Name	Description
					生，则作为主旗标。- 或作为从机，前提是外围设备在此传输期间之前已解决
	4	R/W1c	0	nackf	未确认收到的旗标 当字节传输后收到 NACK 时，硬件会设置此旗标
	3	R/W1c	0	addr	地址匹配 (从机模式) 一旦收到的从机地址与启用的从机地址之一匹配，硬件就会设置此位
	2	R/W	0	rxne	接收数据缓存器不为空 当接收的数据复制到 I2C_RXDR 缓存器中时，此位由硬件设置，并可供读取。读取 I2C_RXDR 缓存器时已清除
	1	R/W	0	txis	传输中断状态 要传输的数据必须写入 I2C_TXDR 缓存器，当 I2C_TXDR 缓存器为空时，硬件会设置此位。 当下一个要发送的数据写入 I2C_TXDR 缓存器时，清除此位。 当仅使用 NOSTRETCH[0]=1 时，软件可以将此位写入"1"，以便产生 txis[0] 事件。
	0	R/W	0	txe	传输数据缓存器为空。 当 I2C_TXDR 缓存器为空时，此位由硬件设置。 当下一个要发送的数据写入 I2C_TXDR 缓存器时，清除此位。此位可以通过软件写入"1"，以便刷新传输数据缓存器 I2C_TXDR。
<b>I2C_SR2: I2C status register</b>					
24h	31:17	-	-	-	保留
	16	R	0	alertp	SMBA 输入状态 (主机模式)。 1: 已启动 (引脚处于低电平)。
	15	R/W	0	busy	Bus 忙碌(busy). 0: Not busy 1: Busy
	14	-	-	-	保留
	13	R/W	0	sdap	已滤波的 SDA 引脚输入状态
	12	R/W	0	sclp	已滤波的 SCL 引脚输入状态
	11	R/W	0	notactive	外围设备未处于启动状态。 0: 启动。 1: 未启动。
	10	R/W	0	slvactived	已匹配从机地址，在停止条件后自动清除。 0: 不匹配。 1: 匹配。
	9	R/W	0	master	主机处于启动状态。 0: 未启动。 1: 启动。
	8	R/W	0	mrw	主机 R/W (接收/传输)状态。 0: 传输。 1: 接收。
	7:1	R/W	0	addcode	地址匹配代码 (从机模式)。 当发生地址匹配事件 (addr[0]=1)时，这些位会使用接收的地址进行更新。在 10 bit 地址的情况下，

Index	Bit	R/W	Default	Name	Description
					addcode[6:0] 提供 10 bit 标头, 后跟地址的 2 个 MSB。
	0	R/W	0	dir	传输方向 (从机模式)。 0: 写入传输, 从机进入接收机模式。 1: 读取传输, 从机进入发射机模式。
<b>I2C_PECR: I2C packet error checking</b>					
28h	31:8	-	-	-	保留
	7:0	R	0	pec	封包错误检查缓存器。 此字段包含内部 PEC。
<b>I2C_TIMINGR1: I2C timing register 1</b>					
2ch	31:20	-	-	-	保留
	19:16	R/W	0	filter	用于 SCL/SDA 输入的数字噪声滤波器。 过滤器将过滤峰值, 最长为 filter[3:0]*t <sub>CLK</sub>
	15:8	R/W	0	scldel	数据设定时间(Data setup time) t <sub>SCLDEL</sub> = (scldel[7:0]+1)*t <sub>CLK</sub>
	7:0	R/W	0	sdadel	数据保持时间(Data hold time) t <sub>SCLDEL</sub> = (sdadel[7:0]+1)*t <sub>CLK</sub>
<b>I2C_TIMINGR2: I2C timing register 2</b>					
30h	31:24	-	-	-	保留
	23:12	R/W	0	sclh	SCL 维持高准位的周期 (主机模式). T <sub>SCLH</sub> = (sclh[11:0]+1)*t <sub>CLK</sub>
	11:0	R/W	0	scll	SCL 维持低准位的周期(主机模式). T <sub>SCLL</sub> = (scll[11:0]+1)*t <sub>CLK</sub>
<b>I2C_TIMEOCTR1: I2C time-out register 1</b>					
34h	31:16	-	-	-	保留
	15	R/W	0	timouten	时钟超时设置。 0: 禁能 SCL 超时检测。 1: 致能 SCL 超时检测。
	14:13	-	-	-	保留
	12	R/W	0	tidle	闲置时钟超时检测。 0: timeouta [11:0] 用于检测 SCL 低准位超时。 1: timeouta [11:0] 用于检测自动程序 SCL 和 SDA 高准位超时 (总线闲置条件)
	11:0	R/W	0	timeouta	总线超时 A。 此字段用于配置: - SCL 低准位超时条件 t <sub>TIMEOUT</sub> 时 tidle[0] = 0。 t <sub>TIMEOUT</sub> = (timeouta[11:0] + 1)*2048*t <sub>CLK</sub> - The bus idle condition (both SCL and SDA high) when tidle[0] = 1. T <sub>IDLE</sub> = (timeouta[11:0] + 1)*4*t <sub>CLK</sub>
<b>I2C_TIMEOCTR2: I2C time-out register 2</b>					
38h	31:16	-	-	-	保留
	15	R/W	0	texten	扩展时钟超时 设置 0: 禁能。 1: 致能。
	14:12	-	-	-	保留
	11:0	R/W	0	timeoutb	总线超时 B t <sub>LOW:EXT</sub> = (timeoutb[11:0] + 1)*2048*t <sub>CLK</sub>

Note: R/W1C: 读取 与 写“1”清除

## 11.4 功能描述

除了接收和传输数据外，此接口还将其从串行格式转换为并行格式，反之亦然。软件启用或禁用中断。接口通过数据引脚 (SDA) 和时钟引脚 (SCL) 连接到 I2C 总线。它可以与标准 (高达 100 kHz)、快速模式 (高达 400kHz)或快速模式 Plus (高达 1MHz)I2C 总线连接。

### 11.4.1 模式选择

接口可以在以下模式之一下运行:

- 从机传输 (Slave transmitter)
- 从机接收 (Slave receiver)
- 主机传输 (Master transmitter)
- 主机接收 (Master receiver)

#### 11.4.1.1 通信流程

在主机模式下，I<sup>2</sup>C 接口启动数据传输并产生时钟信号。串行数据传输始终以 START 条件开始，以 STOP 条件结束。开始和停止条件均由软件在主机模式下产生。

在从机模式下，接口能够识别自己的地址 (7bit 或 10bit)和常规呼叫地址。一般呼叫地址检测可能由软件启用或禁用。保留的 SMBus 地址也可以由软件启用。

数据和地址以 8 字节 (MSB 优先)传输。启动条件之后的第一个字节包含地址 (一个在 7bit 模式中，两个在 10 bit 模式下)。地址始终只能在主机模式下传输。

第 9 个时钟脉冲遵循字节传输的 8 个时钟周期，在此期间接收器必须向发射器发送确认位。请参阅下图。

在主机模式下，软件可能会启用或禁用确认位。主机的 RX 在上次传输中返回 NACK。I<sup>2</sup>C 接口地址 (双寻址 7bit/10 bit 或者常规呼叫地址)可以通过软件选择。

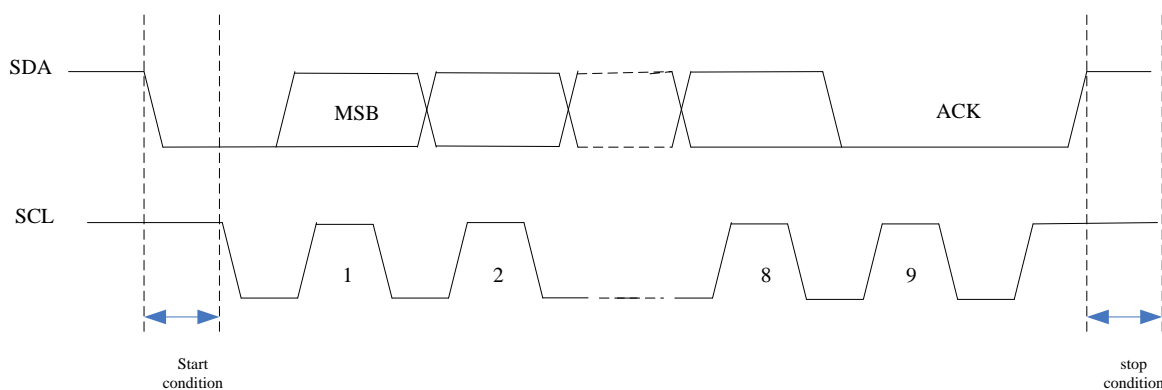


图 41 I2C Bus Protocol

## 11.4.2 I2C 初始化

### 11.4.2.1 噪声滤波

在启用 I2C 功能之前，必须配置数字噪声滤波器 (视情况)。启用数字滤波器后，SCL 或 SDA 线路的电平仅在其保持稳定且超过 I2C\_TIMINGR1 寄存器中滤波器设置的程序设定长度时，才会在内部更改。

### 11.4.2.2 时序

必须配置计时，以确保在主机模式和从机模式中使用的正确数据保持和设置时间。这是通过在 I2C\_TIMINGR1 寄存器中程序设定 scldel[7:0] 和 sdael[7:0]位来完成的。此外，在主机模式下，必须通过在 I2C\_TIMINGR1 寄存器中程序设定 sclh[11:0] 和 scll[11:0]位来配置 SCL 时钟高准位和低准位。

### 11.4.2.3 软件复位

可以通过在 I2C\_CR2 寄存器中设置 swrst[0]位来执行软件复位。内部状态机被复位和通信控制位，以及状态位返回其默认值。配置寄存器不受影响。

## 11.4.3 数据传输

数据传输通过传输和接收数据缓存器和移位寄存器进行管理。

### 11.4.3.1 接收

SDA 输入移位寄存器。当收到完整的数据字节时，如果移位寄存器为空 (rxne[0]=0)，则将移位寄存器复制到 I2C\_RXDR 缓存器中。如果 rxne[0]=1，这意味着之前接收的数据字节尚未读取，则 SCL 线将延展到 I2C\_RXDR 缓存器读取，如果启用时钟延展 (NOSTRETCH[0]=0)。延展将插入第 8 和第 9 个 SCL 脉冲之间 (在确认脉冲之前)。

### 11.4.3.2 传送

如果 I2C\_TXDR 缓存器不为空 (txe[0]=0)，则其内容在第 9 个 SCL 脉冲 (确认脉冲)后复制到移位寄存器中。然后，移位寄存器内容在 SDA 行上移出。如果 txe[0]=1，这意味着在 I2C\_TXDR 缓存器中尚未写入任何数据，则 SCL 行将延展到低准位，直到数据写入 I2C\_TXDR 缓存器。延展在第 9 个 SCL 脉冲后完成。

### 11.4.3.3 硬件传输管理

I2C 在硬件中嵌入了一个字节计数器，以便管理字节传输并在各种模式下关闭通信，例如：

- 在主机模式下产生 NACK、STOP 和 RESTART
- ACK 控制于从机接收模式

字节计数器始终在主机模式下使用。默认情况下，它在从机模式下禁用，但可以通过在 I2C\_CR1 寄存器中设置 sbc[0] (从机字节控制)来启用。

要传输的字节数在 I2C\_BCNTNTR 寄存器中的 nBYTE[7:0]位字段中进行程序设定。如果要传输的字节数 (nBYTE[7:0])大于 255，或者如果接收方想要控制接收到的数据字节的确认值，则必须通过在 i2c\_bcncntr 缓存器中设置 reload[0]位来选择重载模式。在此模式下，当以 nBYTE [7:0] 为单位程序设定的字节数已传输时，将设置 i2c\_iscr 缓存器中的 tcr[0] 旗标，如果设置了 tcie[0]，则产生中断。只要设置 tcr[0] 旗标，SCL 就被拉伸。当 nTB[7:0] 写入非零值时，软件会清除 tcr[0]。当 nBYTE [7:0] 计数器使用最后一个字节数重载时，必须清除

reload [0]位。

在主机模式下重新 reload[0]=0 时，计数器可在 2 种模式下使用：

- 自动结束模式 (auto\_end[0] = '1' in the i2c\_bcncr register)。在此模式下，一旦传输 nNBYTE[7:0]位字段中程序设定的字节数，主服务器将自动发送 STOP 条件。
- 软件结束模式(auto\_end[0] = '0' in the i2c\_bcncr register)。在此模式下，一旦传输 nNBYTE[7:0]位字段中程序设定的字节数，则预期软件操作；设置 tc[0] 旗标，如果设置了 tcie[0]位，则产生中断。只要设置 tc[0] 旗标，SCL 信号就被拉伸。在 I2CX\_CR2 缓存器中设置 START 或 STOP 位时，软件会清除 tc[0] 旗标。当主服务器想要发送 RESTART 条件时，必须使用此模式。

#### 11.4.4 I2C 从机模式

一旦检测到 START 条件，地址就会从 SDA 线路接收并发送到位移缓存器。然后将其与缓存器 I2C\_OAR1 (bit1~bit10) 的地址以及缓存器 I2C\_OAR2 (bit1~bit7, if oa2\_en[0] bit=1) 地址或一般呼叫地址 (如果缓存器 i2c\_cr1 bit0 (gcn[0]) = 1)进行比较。通过配置 i2c\_oar2 缓存器中的 oa2msk[2:0] 位，可以屏蔽多达 7 个 oa2 的 LSB。只要 oa2msk[2:0] 不等于 0，oa2[6:0]的地址比较器就不包括 I<sup>2</sup>C 地址 (0000 XXX 和 1111 XXX)，这些地址也不会被硬件检测到。如果 oa2msk[2:0] =7，则确认所有接收的 7-bit 地址 (保留的地址除外)。Oa2[6:0] 始终是一个 7-bit 地址。

注意：在 10bit 寻址模式下，比较包括标头序列 (11110xx0)，其中 xx 表示地址的两个最重要的位。

- 地址不匹配：接口忽略它，并等待另一个 START 条件。
- 地址匹配：接口按顺序产生：
  - 如果设置了 I2C\_CR1 缓存器中的 addrie，则产生中断。
  - 如果启用了多个地址，则必须读取 I2C\_SR2 缓存器中的 addcode[6:0]位，以便检查匹配的地址。还必须检查 dir[0] 旗标，以便知道传输方向。

在 10 bit 模式下，接收地址序列后，从站始终处于接收机模式。它将在接收重复的 START 条件后，进入发射器模式，然后是具有匹配地址位和最低显著性位集 (11110xx1) 的标头序列。

I2c\_sr2 中的 dir[0] 旗标指示从站处于接收器还是发射器模式。

#### 从机字节控制

为了在从机接收模式下允许字节 ACK 控制，必须通过在 I2C\_CR1 缓存器中设置 sbc[0] 位来启用从机字节控制模式。这必须符合 SMBus 标准。要控制每个字节，必须在 ADDR 中断子程序中初始化为 0x1，并在每个接收字节后重载到 0x1。当收到字节时，设置 tcr[0]位，在第 8 个和第 9 个 SCL 脉冲之间拉伸 SCL 信号低电平。

可以从 I2C\_RXDR 缓存器中读取数据，然后通过 I2C\_CR2 缓存器中配置 nack[0]位来决定是否确认它。SCL 拉伸通过程序设定 nNBYTE [7:0] 释放到非零值：发送确认或不承认，并可以接收下一个字节。nNBYTE [7:0] 可以加载大于 0x1 的值，在这种情况下，接收流在 nNBYTE [7:0] 资料接收期间是连续的。

注 1：当禁用 I2C 或未寻址从机时，或当 addr[0]=1 时，必须配置 sbc[0]位。当 addr[0]=1 或 tcr[0]1 时，可以更改重载位值。

注 2：从属字节控制模式与 NOSTRETCH 模式不兼容。不允许 NOSTRETCH[0]=1 时再去设置 sbc[0]。

#### 从机传送模式

当 I2C\_TXDR 缓存器变为空时，将产生传输中断状态(txis[0])。如果在 I2C\_CR1 缓存器中设置了 txie[0]位，

则产生中断。当使用要传输的下一个数据字节写入 I2C\_TXDR 寄存器时，将清除 txis[0]位。

收到 NACK 时，在 I2C\_ISCR 寄存器中设置 nackf[0]位，如果设置了 I2C\_CR1 寄存器中的 nackie[0]位，则产生中断。从站自动释放 SCL 和 SDA 行，以便让主服务器执行停止或 RESTART 条件。接收 NACK 时未设置 txis[0]位。

当收到停止并在 I2C\_CR1 寄存器中设置 stopie[0]位时，在 I2C\_ISCR 寄存器中设置 stopf[0] 旗标并产生中断。在大多数应用中，sbc[0]位通常被程序设定为“0”。在这种情况下，如果 txe[0] = 0 时接收从机地址 (addr[0]=1)，则可以选择将 I2C\_TXDR 寄存器的内容作为第一个数据字节发送，或者通过设置 txe[0] 位来刷新 I2C\_TXDR 寄存器，以便对新的数据字节进行程序设定。

在从机字节控制模式 (sbc[0]=1) 中，要传输的字节数必须在地址匹配中断子程序 (addr[0]=1) 中的 NBYTES[7:0] 中程序设定。在这种情况下，传输期间的 txis[0] 事件数对应于 NBYTES[7:0] 中程序设定的值。

### 从机接收模式

当 I2C\_RXDR 寄存器已满时，rxne[0] 设置在 I2C\_ISCR 寄存器中，如果 I2C\_CR1 寄存器中的 rxie[0] 已设置，则产生中断。读取 I2C\_RXDR 寄存器时，将清除 rxne[0]。当收到停止并在 I2C\_CR1 寄存器中设置 stopie 时，在 I2C\_ISCR 寄存器中设置 stopf[0]并产生中断。

### 11.4.5 I2C 主机模式

在启用 I2C 之前，软件必须在 I2C\_TIMINGR1 寄存器中配置 SCCH 和 SCLL 以设置主时钟频率。为了支持多主机环境和从机时钟延展，实现了时钟同步机制：

- 使用 SCLL 计数器从 SCL 低准位内部检测开始计算时钟的低准位。
- 使用 SCLH 计数器从 SCL 高准位内部检测开始计算时钟的高准位。

#### 11.4.5.1 通信初始化

要启动通信，必须为 I2C\_SADDR 寄存器中的寻址从站程序设定以下参数：

- 寻址模式 (7bit or 10bit): addr10[0]
- 要发送的从机地址: - 140 -nt[9:0]
- 传输方向: rd\_wrn[0]
- 要传输的字节数: NBYTES[7:0]。如果字节数等于或大于 255 字节，则 NBYTES[7:0] 最初必须用 0xFF 填充。

然后必须在 I2C\_CR2 寄存器中设置起始位。设置 start[0]位时，不允许更改上述所有位。当主机检测到总线闲置 (busy[0] = 0)时，将自动发送 START 条件，然后发送从属地址。在仲裁丢失的情况下，主机会自动切换回从机模式，并且如果将其寻址为从机，则可以确认自己的地址。

#### 11.4.5.2 主机发射模式

对于写入传输，txis[0] 旗标设置在每个字节传输后，在第 9 个 SCL 脉冲之后，当收到 ACK 时。如果设置了 I2C\_CR1 寄存器中的 txie[0] 位，将产生中断。写入 I2C\_TXDR 寄存器的数据将清除旗标。

传输期间的 txis[0] 事件数对应于 NBYTES 中程序设定的值[7:0]。如果要发送的数据字节总数大于 255，则必须通过在 I2C\_BCNTNTR 寄存器中设置 reload[0]位来选择重载模式。在这种情况下，当 NBYTES[7:0] 数据已传输时，将设置寄存器 I2C\_ISCR 中的 tcr[0] 旗标，并将 SCL 线延展到低准位，直到 NBYTES[7:0] 写入非零值。

收到 NACK 时未设置 txis[0] 旗标。在 NACK 接收后自动发送停止条件。Nackf[0] 旗标设置在 I2C\_ISCR 缓存器中，如果设置了 nackie[0]位，则产生中断。

### 11.4.5.3 主机接受模式

对于读取传输，rxne[0] 旗标设置在每个字节接收后，在第 8 个 SCL 脉冲之后。如果 rxie[0]位设置在 I2C\_CR1 缓存器中，则 rxne[0] 事件将产生中断。读取 I2C\_RXDR 时，将清除旗标。如果要接收的数据字节总数大于 255，则必须通过在 I2C\_BCNTNTR 缓存器中设置重载位来选择重载模式。在这种情况下，当 NBYTES[7:0] 数据已传输时，将设置 tcr[0] 旗标，并将 SCL 线延展到低准位，直到 NBYTES[7:0] 写入非零值。

当 reload[0]=0 和 NBYTES[7:0] 数据传输时：

- 在自动结束模式 (auto\_end[0]=1)中，在上次接收的字节之后自动发送 NACK 和 STOP。
- 在软件结束模式 (auto\_end[0]=0) 中，在上次接收的字节后自动发送 NACK，设置 tc[0] 旗标，并将 SCL 线拉低准位，以便允许软件操作

可以通过在 I2C\_CR2 缓存器中设置具有正确的从机地址配置和要传输的字节数来请求 RESTART 条件。设置起始位将清除 tc[0] 旗标，并在总线上发送 START 条件，然后是从机地址。可以通过在 I2C\_CR2 缓存器中设置停止位来请求停止条件。设置停止[0]位将清除 tc[0] 旗标，并在总线上发送 STOP 条件。

## 11.4.6 DMA 要求

### 11.4.6.1 使用 DMA 传送

通过在 I2C\_CR1 缓存器中设置 txdmaen[0]位，可以启用 DMA 模式进行传输。每当设置 txis[0]位时，数据将从使用 DMA 配置的内存区域加载到 I2C\_TXDR 缓存器。只有数据通过 DMA 传输。

- 在主机模式下：初始化、从机地址、方向、字节数和 START 位由软件程序设定。使用 DMA 传输所有数据时，必须先初始化 DMA，然后再设置 START 位。传输结束使用 NBYTES[7:0] 计数器进行管理。
- 在从机模式下：
  - 使用 NOSTRETCH[0]=0，当使用 DMA 传输所有数据时，必须在地址匹配事件之前或在 ADDR 中断子程序中初始化 DMA，然后才能清除 addr[0]。
  - 使用 NOSTRETCH[0]=1 时，必须在地址匹配事件之前初始化 DMA。

### 11.4.6.2 使用 DMA 接收

藉由在 I2C\_CR1 缓存器中设置 rxdmaen[0]位，可以启用 DMA 模式进行接收。每当设置 rxne[0]位时，数据将从 I2C\_RXDR 缓存器加载到使用 DMA 配置的内存区域。只有数据 (包括 pec[0])与 DMA 一起传输。

- 在主机模式下，初始化、从机地址、方向、字节数和 START 位由软件程序设定。使用 DMA 传输所有数据时，必须先初始化 DMA，然后再设置 START 位。传输结束使用 NBYTES[7:0] 计数器进行管理。
- 在具有 NOSTRETCH[0]=0 的从机模式下，当使用 DMA 传输所有数据时，必须在地址匹配事件之前或在 ADDR 中断子程序中初始化 DMA，然后才能清除 addr[0] 旗标。

## 12 UART

### 12.1 主要概述

- 全双工异步通信
- NRZ standard format
- 可规划 16 或 8 个时钟的过采样方法
- A baud rate 产生器
- 8 bit 或 9 bit 数据字长度
- 1 or 2 stop bits
- 发射器或接收器的单独启动位
- 使用 DMA 进行可规划的通信
  - 在保留 SRAM 中缓冲接收/传输的字节
- IrDA SIR ENDEC
  - 支持正常模式的 3/16 bit 持续时间
- 单线半双工通信
- 传输侦测旗标
  - Receive buffer full
  - Transmit buffer empty
  - End of Transmission
- 奇偶校验控制
  - 偶数、奇数或无同位产生和检测
- 四个错误侦测旗标
  - Overrun error
  - Noise error
  - Frame error
  - Parity error
- 使用旗标中断源
  - 传输数据缓存器为空
  - Transmit complete
  - 接收数据缓存器已满
  - Overrun error
  - Frame error
  - Noise error
  - Parity error



## 12.2 方块图

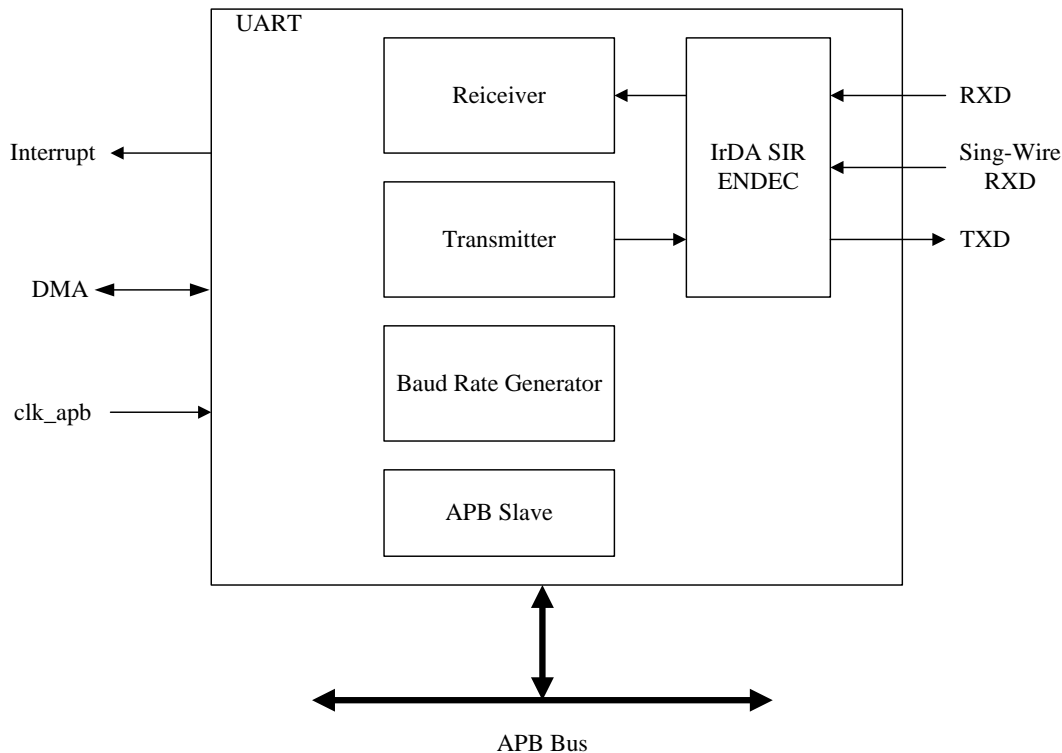


图 42 UART 方块图

## 12.3 缓存器描述

表格 30 UART 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>UART_CR1: 控制缓存器 1</b>					
00h	31	R/W	0	UE	UART 致能 0: UART 预除器和输出禁能 1: UART 启动
	30:16	-	-	-	保留
	15	R/W	0	TE	发射器启动。 0: 发射器已关闭。 1: 发射器已启动。
	14	R/W	0	RE	接收器启动。 0: 接收器已关闭。 1: 接收器已启动并开始搜索起始位。
	13	R/W	0	OVER8	过采样模式。 0: 超过采样 16。 1: 超过采样 8。 注意: 在 IrDA 中, 当 IREN =1 时, 超过采样 8 不可用, 然后由硬件强制 OVER8 为"0"。

Index	Bit	R/W	Default	Name	Description
	12	R/W	0	WORD_L	字长度。 0: 1 个起始 bit, 8 个数据 bit, n 停止 bit。 1: 1 个起始 bit, 9 个数据 bit, n 停止 bit。
	11	R/W	0	DMAT	DMA 启动发射器。 1: DMA 模式已启动传输。 0: DMA 模式因传输而关闭。
	10	R/W	0	DMAR	DMA 启动接收器。 1: DMA 模式已启动接收。 0: DMA 模式为接收关闭。
	9:3	-	-	-	保留
	2	R/W	0	PCE	Parity 控制致能。 0: 禁能。 1: 致能。
	1	R/W	0	PS	Parity 选择。 0: Even。 1: Odd。
	0	R/W	0	STOP	0: STOP = 1bit 1: STOP = 2bit
<b>UART_CR2: 控制缓存器 2</b>					
04h	31:7	-	-	-	保留
	6	R/W	0	TXEIE	TXE 中断启动。 0: 中断被抑制。 1: 每当 TXE=1 时都会产生 UART 中断。
	5	R/W	0	TCIE	传输完全中断启动。 0: 中断被抑制。 1: 每当 TC=1 时都会产生 UART 中断
	4	R/W	0	RXNEIE	RXNE 中断启动。 0: 中断被抑制。 1: 每当 ORE=1 或 RXNE=1 时都会产生 UART 中断
	3	R/W	0	EIE	错误中断启动。 0: 中断被抑制。 1: 每当 FER=1 或 ORE=1 或 NE=1. 时都会产生中断。
	2:1	-	-	-	保留
	0	R/W	0	PEIE	PE 中断启动。 0: 中断被抑制。 1: 每当 PE=1 时都会产生 UART 中断
<b>UART_STS: 状态缓存器</b>					
08h	31:7	-	-	-	保留
	6	R/W	0	TXE	传输数据缓存器为空。 0: 数据不会传输到班次缓存器。 1: 数据传输到班次缓存器。
	5	R/W	0	TC	UART: 传输完成。 0: 传输未完成。 1: 传输完成。当传输数据缓存器不为空时, 通过 S/W 写入 1 或 H/W 清除。

Index	Bit	R/W	Default	Name	Description
	4	R/W	0	RXNE	UART: 读取数据缓存器不为空。 0: 未收到资料。 1: 接收的数据已准备好可供读取。通过 S/W 读取接收数据缓存器或 S/W 写入 1 进行清除。
	3:0	-	-	-	保留
<b>UART_TXDR: 传送数据缓存器</b>					
0Ch	31:9	-	-	-	保留
	8:0	W	0	TDR[8:0]	传输 UART 数据值
<b>UART_RXDT: Receive data register</b>					
10h	31:9	-	-	-	保留
	8:0	R	0	RDR[8:0]	Receive Data value
<b>UART_BRR: Baudrate 缓存器</b>					
14h	31:16	-	-	-	保留
	15:4	R/W	0	Mantissa[11:0]	波特率发生器的尾数
	3:0	R/W	0	Fraction[3:0]	波特率发生器的分数
<b>UART_CR3: 控制缓存器 3</b>					
18h	31:15	-	-	-	保留
	14	R/W	0	IREN	IrDA mode 致能。 0: IrDA 禁能。 1: IrDA 致能。
	13	R/W	0	HDSEL	Half-duplex 选择。 0: Half-duplex 禁能。 1: Half-duplex 致能。
	12:0	-	-	-	保留
<b>UART_FERE: Framing error 侦测</b>					
24h	31:4	-	-	-	保留
	3	R/W	0	FERE	帧错误侦测启动。 0: 帧错误侦测被关闭。 1: 启动帧错误侦测。
	2:0	-	-	-	保留
<b>UART_ERR: 错误旗标缓存器</b>					
30h	31:8	-	-	-	保留
	7	R/W	0	ORE	溢出错误。 0: 无溢出错误。 1: 检测到溢出错误。设置此位后, RDR 缓存器内容不会丢失, 但班次缓存器将被覆盖。通过 S/W 写入 1 清除。
	6	R/W	0	NE	噪声错误旗标。 0: 未检测到噪音。 1: 检测到噪音。通过 S/W 写入 1 清除。
	5	R/W	0	PE	UART 奇偶校验位错误旗标。 0: 无同位错误。 1: 同位错误。通过 S/W 写入 1 清除
	4	-	-	-	保留
	3	R/W	0	FER	框架错误旗标。

Index	Bit	R/W	Default	Name	Description
	2:0	-	-	-	0: 未检测到帧错误。 1: 检测到帧错误。通过 S/W 写入 1 清除。 错误侦测条件:接收过程中每个数据字节的停止位较低。 保留

## 12.4 功能描述

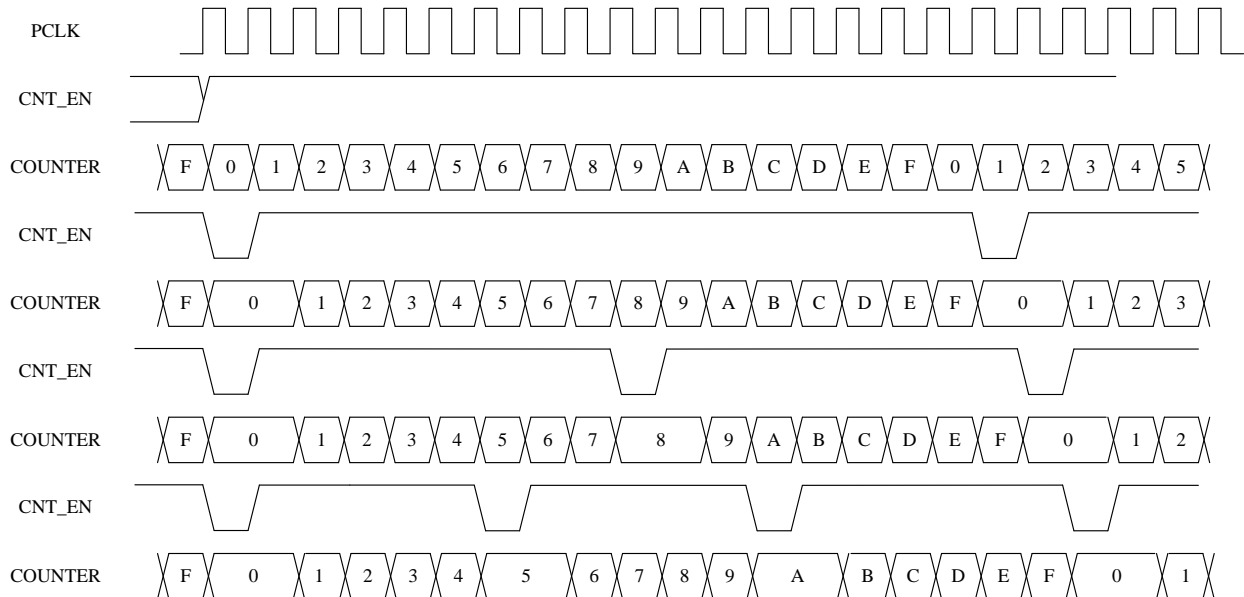
### 12.4.1 分数波特率 (baud rate) 产生

$$\text{Baud rate} = \frac{\text{Sysclk}}{8 * (2 - \text{OVER8}) * (\text{Mantissa} + \frac{\text{Fraction}}{16})}$$

表格 31 UART Baud Rate Table

Speed	Sysclk 16MHz (OVER8=0)			Sysclk 16MHz (OVER8=1)		
	Baud Rate Register	Actual	Error	Baud Rate Register	Actual	Error
2.4 K	416.625	2400	0.0%	833.3125	2400	0.0%
9.6 K	104.125	9604	0.04%	208.3125	9601	0.01%
19.2 K	52.0625	19208	0.04%	104.125	19208	0.04%
38.4 K	26.0625	38369	0.08%	52.0625	38415	0.04%
57.6 K	17.3125	57762	0.28%	34.75	57554	0.08%
115.2 K	8.625	115942	0.64%	17.375	115108	0.08%
230.4 K	4.3125	231884	0.64%	8.6875	230216	0.08%
921.6K	NA	NA	NA	2.1875	914286	0.8%
2 M	NA	NA	NA	1	2M	0.0%
3 M	NA	NA	NA	NA	NA	NA

Example:  $\frac{Pclk}{16 * (1 + \frac{1}{16})}$ ,  $\frac{Pclk}{16 * (1 + \frac{2}{16})}$ ,  $\frac{Pclk}{16 * (1 + \frac{3}{16})}$



Example: 
$$\frac{Pclk}{16 * (3 + \frac{3}{16})}$$

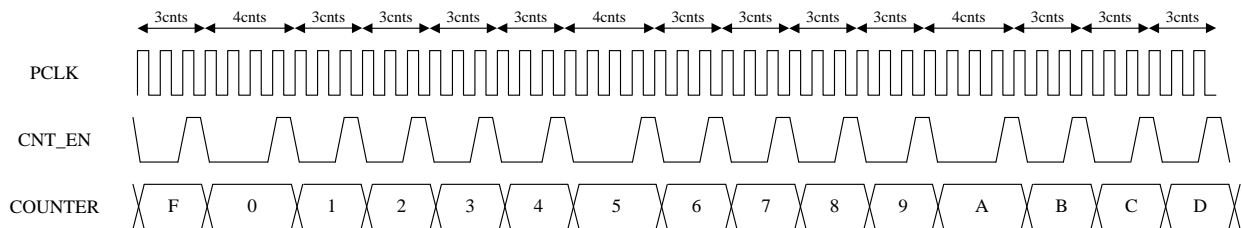


图 43 波特率发生器示例

### 12.4.2 接收机

#### Overrun error

未复位 RXNE 时收到字符时，将发生溢出错误。在清除 RXNE 位之前，无法将数据从移位寄存器传输到 RDR 寄存器。

#### Noise error

资料采样时过采样 16，采样值为 bit 8、9、10。

资料采样时过采样 8，采样值为 bit 4、5、6。

噪声误差标志位设置在 RXNE 位的上升沿。

**表格 32**    **UART 噪声误差**

Sample value	Noise error	Received bit
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

**Framing error**

当接收过程中每个数据字节的停止位较低时，检测到帧错误。

**12.4.3 奇偶校验控制**
**表格 33**    **同位(Parity)控制表**

WORD_L bit	PCE bit	UART Data Frame
0	0	Start + 8 bit data + Stop
0	1	Start + 7 bit data + parity bit + Stop
1	0	Start + 9 bit data + Stop
1	1	Start + 8 bit data + parity bit + Stop

**12.4.4 单线半双工通信**

通过设置 HDSEL 位选择单线半双工模式。在此模式下，CLKEN 和 IREN 位必须保持清除。

当 HDSEL 写入 1:

6. TX 和 RX 线路在内部连接。
7. 不再使用 RX 引脚。
8. 当没有传输数据时，TX 引脚是浮动输入。

## 13 SPI

### 13.1 主要概述

串行外围接口 (SPI) 模块是半双工同步串行接口，可用于与其它外围或微控制器设备通信。

- 主机操作或从机操作
- 可程序设定时钟极性和相位
- 带 MSB 优先或 LSB 优先移位的可程序设定数据顺序
- 具有中断功能的专用传输和接收旗标
- 三条线路上的全双工同步传输
- 在两条带或不带双向数据线的两条线路上进行单路同步传输
- 8 bits 或 16 bits 传输帧格式选择
- 多主机模式功能与总线繁忙状态旗标
- 主机模式或从机模式波特率高达  $CLK\_APB / 2$
- 通过硬件或软件对主机和从机进行 NSS 管理
- 硬件 CRC 功能，具有可配置的多项式，提供可靠的通信
- 具有中断功能的杂项错误旗标
- 具有 DMA 功能的传输和接收缓冲器
- 能够进行下一次访问延迟配置
- 能够用主机 RX-only 进行伪(pseudo)计数

### 13.2 方块图

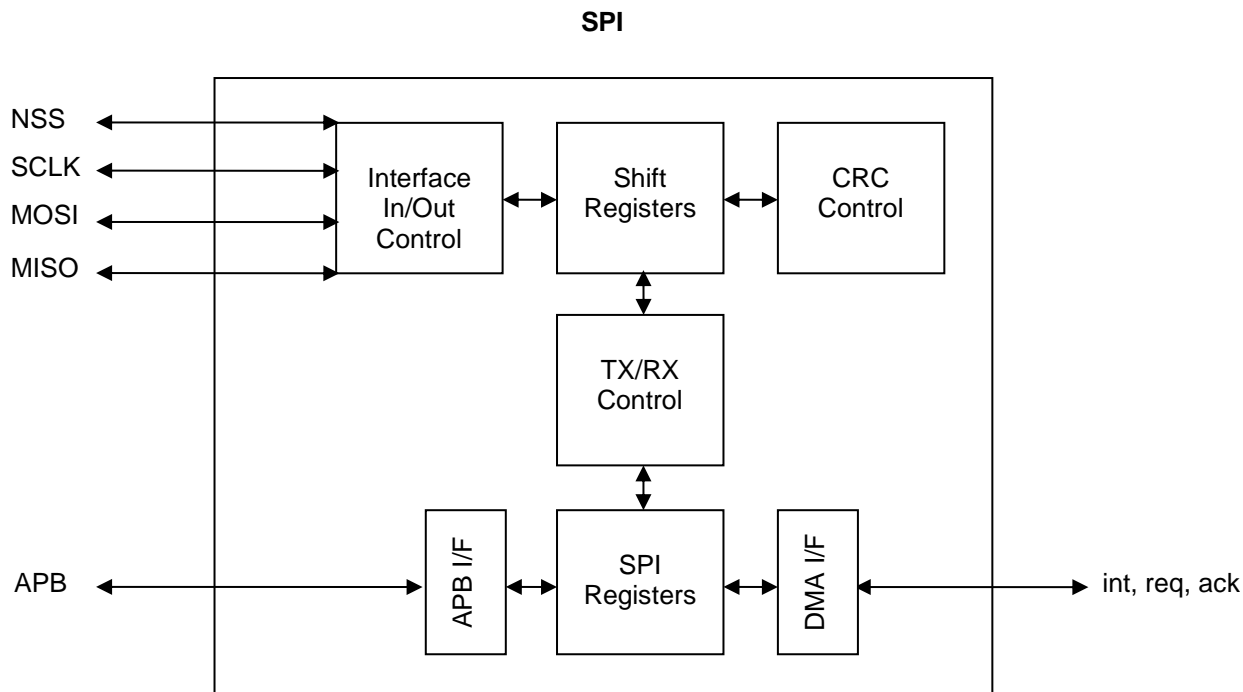


图 44 SPI module 方块图

### 13.3 缓存器描述

表格 34 SPI 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>SPI_CR1: SPI 控制缓存器 1</b>					
00h	15	R/W	0	SPE	SPI 致能设置, 如果发生 MODFIF, 将清除此位。 注意: 在关闭 SPE 时, SCLK 应保持闲置状态
	14	R/W	0	HSCC	高速时钟补偿。 设置 HSCC (如果 SCLK ≥ 12MHz)。
	13	R/W	0	BIDIMODE	双向数据模式设置 (单数)
	12	R/W	0	BIDIOE	双向模式输出设置。 0: RX. 1: TX
	11	R/W	0	CRCE	硬件 CRC 设置
	10	R/W	0	CRCNEXT	下一个 CRC 传输。 1: 在写入最后一个 TX 数据之后或读取倒数第二个数据之后。 注意: 在产生/检查 CRC 后, 将自动清除此位。
	9:8	-	-	-	保留
	7	R/W	0	MSTR	主机选择设置
6	R/W	0	SSOE	从机选择输出设置 (主机模式), 如果设置了	



Index	Bit	R/W	Default	Name	Description
					SSOE, 则忽略 NSS 输入, 并且不应在主机模式下设置 MODFIF
	5	R/W	0	SSM	软件从机模式。 0: 主机模式:检测到的 NSS 信号由 SPE 控制。从机模式:SPI 检测来自 NSS 端口的 NSS 信号。如果检测到的 NSS 处于非活动状态,则 SPI 处于保持状态。 1: SPI 检测来自 SSI 设置的 NSS 信号。
	4	R/W	0	SSI	软件从机输入。 软件 NSS 信号准位, 它仅有效 SSM = '1'
	3	R/W	0	MODE16	1 帧数据长度设定 (8/16 bits) 0: 8 1: 16
	2	R/W	0	LSBFE	LSB-first 设置
	1	R/W	0	CPOL	clock 极性(polarity) 0: 当总线闲置时, sclk = '0'。 1: 当总线闲置时, sclk = "1"。
	0	R/W	0	CPHA	时钟相位。 0: 第一个时钟转换边缘捕获数据。 1: 第二个时钟转换边缘捕获数据。
<b>SPI_CR2: SPI 控制寄存器 2</b>					
04h	15:12	-	-	-	保留
	11:8	R/W	0h	NAD	下一次访问延迟 (字到字延迟, 单位:0.5 位)包括 NSS 到 SCLK 到 NSS 延迟 (仅限主机模式)。注意:在 NAD 期间, BUSY 旗标处于非活动状态
	7:0	R/W	01h	BRS	比特率选择, 有效范围为 00h 到 ffh。 $SCLK = CLK\_APB / ((BRS+1)*2)$ 注意: $CLK\_AHB / ((BRS+1)*2) \leq 24MHz$
<b>SPI_EN: SPI counter and enable bit</b>					
08h	15:8	R/W	00h	PSCNT	主机 rx-only 伪(pseudo)计数器 (用于 sclk 产生)
	7	R/W	0	TXNFIE	TX 不为满的中断设置
	6	R/W	0	RXNEIE	RX 不为空中断设置
	5	R/W	0	MODFIE	模式故障中断设置
	4	R/W	0	CRCFIE	CRC 故障中断设置
	3	R/W	0	TXOVRIE	TX 溢出中断设置
	2	R/W	0	RXOVRIE	RX 溢出中断设置
	0	R/W	0	RXDMAEN	RX DMA 设置
<b>SPI_STS: SPI 状态旗标</b>					
0ch	15:8	-	-	-	保留
	7	R	1	TXNFIF	TX 不为满旗标, 写入 BUF_DAT 以清除
	6	R	0	RXNEIF	RX 不为空旗标, 读取 BUF_DAT 以清除
	5	R/W	-	MODFIF	模式故障旗标, 写入"1"以清除。 故障条件: 主机模式: NSS = INPUT-0 从机模式: NSS = HIGH, 但帧不完整
	4	R/W	-	CRCFIF	CRC 故障旗标, 写入"1"以清除 (注意)
	3	R/W	-	TXOVRIF	TX 溢出旗标, 写入"1"以清除

Index	Bit	R/W	Default	Name	Description
	2	R/W	-	RXOVRIF	RX 溢出旗标, 写入"1"以清除
	1	R	-	BUSY	TX/RX 忙碌状态
	0	-	-	-	保留
<b>SPI_DAT: SPI data buffer</b>					
10h	15:0	R/W	00h	BUF_DAT	TX/RX 的缓冲区数据
<b>SPI_CRC_TX: SPI CRC TX register</b>					
18h	15:0	R	ffffh	CRC_TX	CRC TX 资料
<b>SPI_CRC_RX: SPI CRC RX 缓存器</b>					
1ch	15:0	R	ffffh	CRC_RX	CRC RX 资料

Note: illegal CRCFIF condition under unidirectional TX should be handled by FW.

## 13.4 功能描述

### 状态旗标 (Status Flags)

为应用程序提供了三个状态旗标, 以完全监视 SPI 总线的状态。

#### TX 缓冲区未满载旗标 (TX buffer not full flag, TXNF)

设置该标记时, 此旗标指示 TX 缓冲区未满载, 并且要传输的下一个数据可以加载到缓冲区中。写入 BUF\_DAT 缓存器时, 将清除 TXNF 旗标。

#### RX 缓冲区不为空 (RX buffer not empty, RXNE)

设置时, 此旗标指示 RX 缓冲区中有有效的接收数据。读取 BUF\_DAT 时将清除它。

#### BUSY flag

此 BUSY 旗标由硬件设置和清除 (写入此旗标不起作用)。BUSY 旗标指示 SPI 通信层的状态。设置 BUSY 后, 它表示 SPI 正忙于通信。

如果软件想要禁能 SPI 并进入停止模式 (或禁用外围时钟), BUSY 旗标可用于检测传输的结束。这样可以避免损坏上次传输。

BUSY 旗标还可用于避免多主机系统中的写入冲突。在传输开始时设置 BUSY 旗标。

可清除如下:

- 传输完成后
- 禁能 SPI 时
- 当主机模式故障发生时 (MODFIF=1)

当通信不是连续的时, 每个通信之间的 BUSY 旗标都为低。

通信连续时:

- 在主机模式下, 在传输过程中, BUSY 旗标保持为高
- 在从机模式下, BUSY 旗标在每次传输之间的半 SPI 时钟周期中处于低位

## 错误旗标

### Mode fault (MODFIF)

当主机在从机设备数据传输过程中检测到其输入 NSS 引脚拉低或停用 NSS 时，会发生模式故障，这会自动设置 MODFIF 位。模式故障以下列方式影响 SPI 外围：

- 设置 MODFIF 位，如果设置了 ERRIE 位，则产生 SPI 中断。
- SPE 位已清除。这将阻止设备的所有输出，并禁用 SPI 接口。

将“1”写入 MODFIF 位以清除模式故障状态。为了避免在包含多个 MCUs 的系统中出现多个从机冲突，必须在 MODFIF 位清除序列期间将 NSS 引脚拉高。MODFIF 引起的中断不受 SPE 设置的控制。

作为安全性，硬件不允许在设置 MODFIF 位时设置 SPE 位。

在从机中，如果正在处理不完整的帧，并且 NSS 从低到高时被停用，则将设置 MODFIF 位元。中断例程可用于通过执行复位或返回到预设状态来从此状态中干净地恢复。

### CRC error

此旗标用于验证在控制缓存器中接收的 CRCE 位时接收的值的正确性。如果收到的 CRC 值不正确，则设置 CRCERR 旗标缓存器。CRCFIF 响应 CRC 检查结果后 CRCNEXT 被硬件清除。

### 不足和溢出条件 (Underrun and overrun condition)

#### TXUDRIF

当设备在帧传输开始之前未填充数据字节时，将发生运行不足的情况。当出现不足情况时：

- 设置 TXUDRIF 位，如果设置了 TXUDRIE 位，则产生中断。

在这种情况下，发射机缓冲区没有从设备传输的内容，在这种情况下，如果发生 TXUDRIF，传输的数据将毫无意义。清除 TXUDRIF 位是通过将“1”写入 TXUDRIF 缓存器来完成的。

#### RXOVRIF

当 TX 设备已发送数据字节且 RX 设备未清除由之前传输的数据字节导致的 RXNE 位时，将发生溢出情况。发生超运行情况时：

- 设置 RXOVRIF 位，如果设置 RXOVRIE 位，则产生中断。在这种情况下，接收方缓冲区内容将不会使用由主机新接收的数据进行更新。从 BUF\_DAT 缓存器读取返回此字节。所有其它随后传输的字节将丢失。清除 RXOVRIF 位是通过将“1”写入 RXOVRIF 缓存器来完成的。

## CRC Calculation

实施两个单独的 CRC 计算器（关于传输和接收数据流），以检查传输和接收数据的可靠性。SPI 提供 CRC8 或 CRC16 计算，具体取决于通过模式选择位选择的数据格式。CRC 使用 CRC\_POLY 缓存器中程序设定的多项式连续计算。

通过设置 CRCE 缓存器启用 CRC 计算，CRCE 的低到高转换可复位 CRC 逻辑。计算在 CPHA 和 CPOL 设置定义的采样时钟边上处理。计算出的 CRC 值在数据块的末尾自动检查，以及 CPU 或 DMA 管理的传输。当检测到在接收数据上内部计算的 CRC 与发射机发送的 CRC 之间出现不匹配时，CRCERR 旗标将设置为指示数据损坏错误。处理 CRC 计算的正确过程取决于 SPI 配置和所选的传输管理。对于 8 bit 模式，仅使用 CRC[POLY[7:0]]，初始 CRC 值和 XOR CRC 输出均固定为 0。

Reference CRC polynomials:

8-bit:  $C(x) = x^8 + x^2 + x^1 + x^0$ , CRC\_POLY = 16'h0007

16-bit:  $C(x) = x^{16} + x^{12} + x^5 + x^0$ , CRC\_POLY = 16'h1021

## SPI Interrupts

中断事件	事件旗标	致能控制位
Transmit buffer not full flag	TXNFIF	TXNFIE
Receive buffer not empty flag	RXNEIF	RXNEIE
Mode fault flag	MODFIF	MODFIE
CRC error flag	CRCFIF	CRCFIE
TX underrun error flag	TXUDRIF	TXUDRIE
RX overrun error flag	RXOVRIF	RXOVRIE

SPI 中断由 SPE 设置、中断启用和中断旗标控制，但独立于 SPE 设置的中断源的 MODFIF 除外。

## 使用 DMA 的 SPI 通信 (SPI communication using DMA)

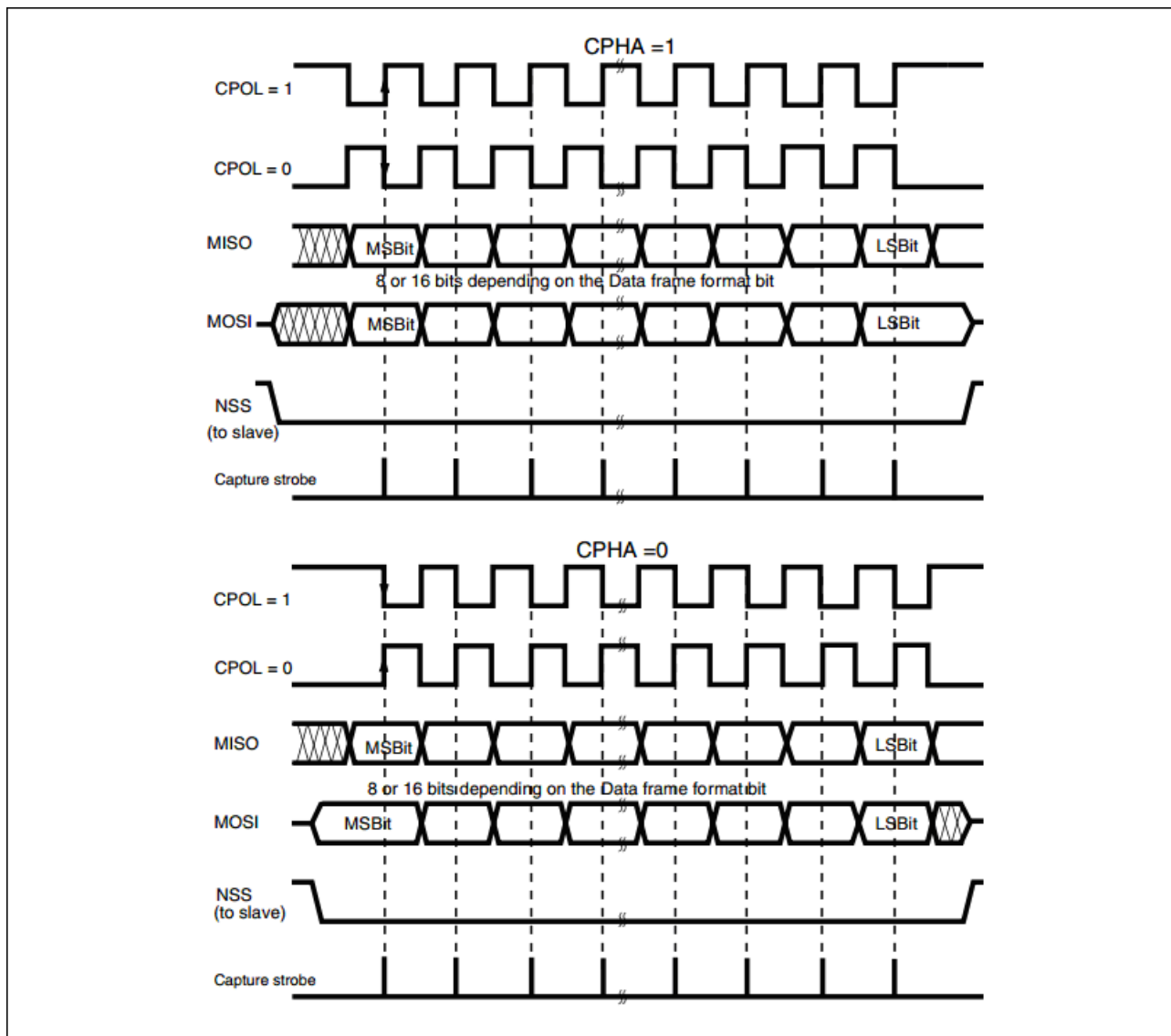
为了以最大速度运行，SPI 需要随传输数据一起馈送，并且应读取 Rx 缓冲区上接收的数据，以避免不足/溢出。为了便于传输，SPI 具有实现简单请求/确认协议的 DMA 功能。启用位打开时，将请求 DMA 访问。必须向 TX 和 RX 缓冲区发出单独的请求：

- 在传输中，每次 TXNF 设置为 1 时都会发出 DMA 请求。然后，DMA 写入 BUF\_DAT 缓存器 (这将清除 TXNF 旗标)。
- 在接收中，每次将 RXNE 设置为 1 时都会发出 DMA 请求。然后，DMA 读取 BUF\_DAT 缓存器 (这将清除 RXNE 旗标)。

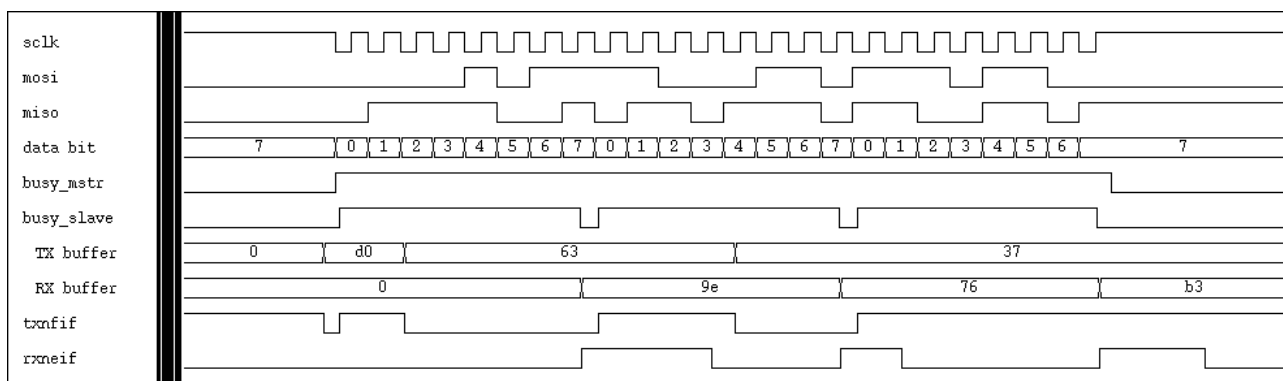
当 SPI 仅用于传输数据时，可以仅启用 SPI TX DMA 通道。在这种情况下，将设置 RXOVRIF 旗标，因为无法读取接收的数据。当 SPI 仅用于接收数据时，可以通过设置 PSCNT 缓存器仅启用 SPI RX DMA 信道。

在传输模式下，当 DMA 写入要传输的所有数据 (标记 TCIF 设置在 DMA 缓存器中) 时，可以监视 BUSY 旗标以确保 SPI 通信完成。这是为了避免在禁用 SPI 或进入停止模式之前损坏最后一个传输所必需的。软件必须首先等待 TXNF=1，然后等到 BUSY=0。

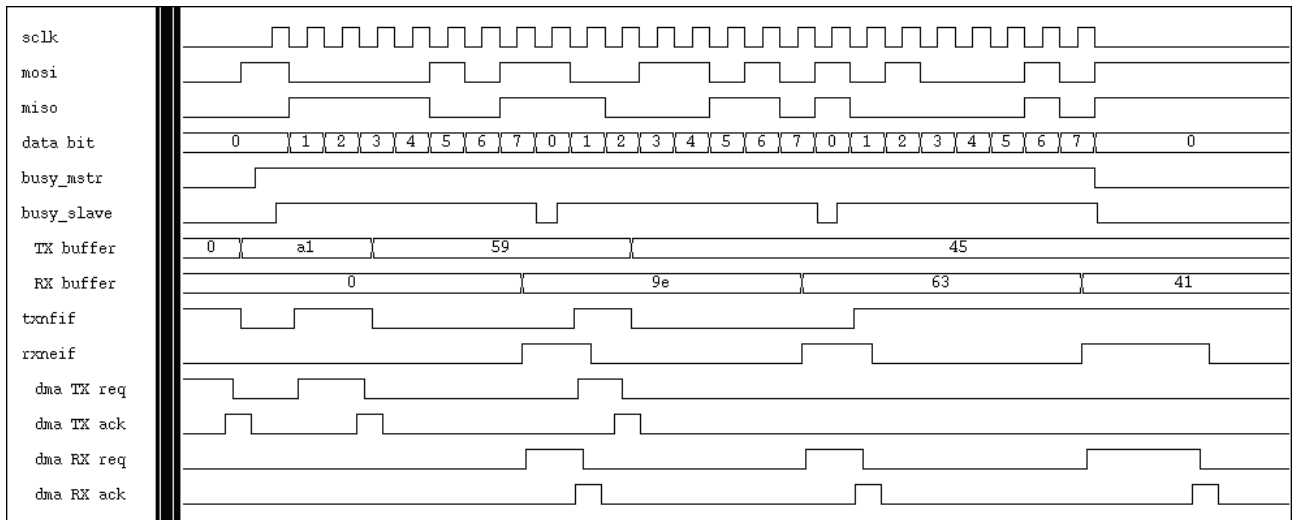
**13.5 SPI Timing Diagram**



Data clock timing diagram for LSBFE = 0



Signal behavior in master/slave mode (LSBFE = CPOL = CPHA = 1, MODE16 = 0)



使用 DMA 传输/接收 (LSBFE = 1, mode16 = CPOL = CPHA = 0)

注意: 在从机模式下, 下降的 NSS 和第一个 SPI 时钟转换, 最后的 SPI 时钟转换和上升的 NSS 之间需要至少 3 个系统时钟。

## 14 I2S

### 14.1 主要概述

- 兼容 APB 接口
- I2S 可工作于为主机端或从机端
- 数据长度支持 16、24、32 位
- 支持 I<sup>2</sup>S 和 MSB-justified 数据格式
- 支持一个输出信道和一个输入信道 (共享 BCLK 和 LRCK)
- 提供两个 8 字组 (WORD) FIFO 数据缓冲区，一个用于传输，另一个用于接收
- 支持两个 DMA 请求，一个用于传输，另一个用于接收
- 当缓冲区内之数据大小大于程序设定之边界时(可设置为 1、2、3、4 字组)，可产生中断要求
- 位时钟(BCLK)和左/右时钟(LRCK)的比率可设置为 32/48/64

### 14.2 方块图

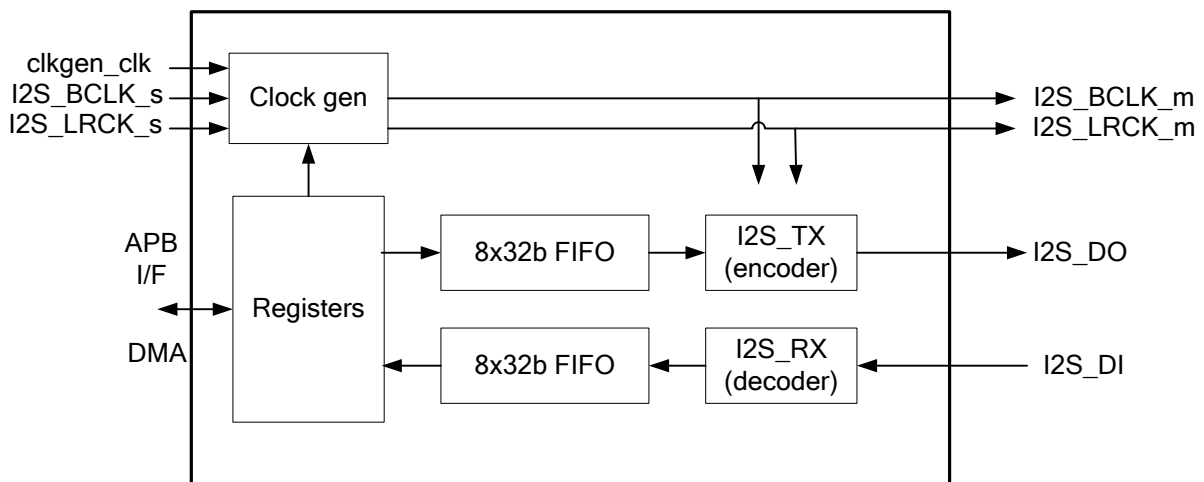


图 45 I<sup>2</sup>S 方块图

**14.3 缓存器描述**
**表格 35 I<sup>2</sup>S 控制缓存器**

Index	Bit	R/W	Default	Name	Description
<b>I2S_CFG: I2S 设置缓存器</b>					
00h	31:20	-	-	-	保留
	19:18	R/W	0	BCLKGEN_SRC	位时钟(BCLK)产生器的时钟来源选择: 00: MCU_CLK 01: AUX_PLL 10: XTAL 11: 保留
	17:11	R/W	05h	BCLK_LOW_DUTY	位时钟(BCLK) 低位准之周期 (duty) 设置 ((BCLK_LOW_DUTY[6:0]+1)+(BCLK_HIGH_DUTY[6:0]+1))
	10:4	R/W	05h	BCLK_HIGH_DUTY	位时钟(BCLK) 高位准之周期 (duty) 设置 ((BCLK_LOW_DUTY[6:0]+1)+(BCLK_HIGH_DUTY[6:0]+1))
	3:2	R/W	0	BCLK_RATE	BCLK/ LRCK 比率设置 ( $f_{BCLK}/f_{LRCK}$ ) 0 or 1: 32, 2: 48, 3: 64
	1	R/W	0	LJ_MODE	1: left justified, 0: I2S mode
	0	R/W	1	MASTER_SLAVE	1: 主机模式(Master), 0: 从机模式(Slave)
<b>I2S_DCFG: I2S 资料设置</b>					
04h	31:20	-	-	-	保留
	19:18	R/W	0	RX0_THRESHOLD	I2S RX0 频道产生中断/DMA 要求所需之接收数据长度设置 0: 1 word, 1: 2 words, 2: 3 words, 3: 4 words
	17:16	R/W	0	RX0_BLEN	I2S RX0 频道之音频位长度设置。 0 或 1: 16 位、2: 24 位、3: 32 位
	15:4	-	-	-	保留
	3:2	R/W	0	TX0_THRESHOLD	I2S TX0 频道产生中断/DMA 要求所需之数据长度设置。 0: 1 word, 1: 2 words, 2: 3 words, 3: 4 words
	1:0	R/W	0	TX0_BLEN	I <sup>2</sup> S TX0 频道之音频位长度设置 0 or 1: 16-bit, 2: 24-bit, 3: 32-bit
<b>I2S_CR: I2S TX/RX 控制缓存器</b>					
08h	31:20	-	-	-	保留
	19	R/W	0	RX0_FLUSH	写入 1 以清除接收 FIFO 缓冲区内之数据(RX0)
	18	R/W	0	RX0_INT_EN	启用中断 (RX0 通道)
	17	R/W	0	RX0_DMA_EN	启用 DMA 传输 (RX0 通道)
	16	R/W	0	RX0_EN	启用 I2S 功能 (RX0 通道)
	12:4	-	-	-	保留
	3	R/W	0	TX0_FLUSH	写入 1 以清除传送 FIFO 缓冲区内之数据(TX0)
	2	R/W	0	TX0_INT_EN	启用中断 (TX0 通道)
	1	R/W	0	TX0_DMA_EN	致能 DMA 传输 (TX0 通道)
0	R/W	0	TX0_EN	致能 I2S 功能 (TX0 信道)	
<b>I2S_TXSR: I2S TX 状态</b>					
0Ch	31:8	-	-	-	保留
	7	R/W1C	-	TX0_ERROR	TX0 频道发生错误 (数据过载或欠载), 软件写 1 来



Index	Bit	R/W	Default	Name	Description
					进行清除
	6	R	1	TX0_EMPTY	TX0 频道之传送 FIFO 缓冲区为空
	5	R	0	TX0_FULL	TX0 频道之传送 FIFO 缓冲区已满
	4	R	1	TX0_INT	TX0 频道发生中断
	3:0	R	0	TX0_CNT	TX0 频道之 FIFO 缓冲区内之数据长度 (以字组为单位)
<b>I2S_RXSR: I2S RX 状态</b>					
10h	31:8	-	-	-	保留
	7	R/W1C	0	RX0_ERROR	RX0 频道发生错误 (数据过载或欠载), 软件写 1 来进行清除
	6	R	0	RX0_EMPTY	RX0 频道之接收 FIFO 缓冲区为空
	5	R	0	RX0_FULL	RX0 频道之接收 FIFO 缓冲区已满
	4	R	1	RX0_INT	RX0 频道发生中断
	3:0	R	0	RX0_CNT	RX0 频道之 FIFO 缓冲区内之数据长度 (以字组为单位)
<b>I2S_TX_FIFO: I2S TX FIFO</b>					
20h	31:0	W	-	TX0_FIFO	TX0 频道之传送数据缓存器 16 位数据: [31:16]为右声道, [15:0]为左声道 24 位数据: [23:0]为右声道或左声道 (左声道数据先送出) 32 位数据: [31:0]为右声道或左声道 (左声道数据先送出)
<b>I2S_RX_FIFO: I2S RX FIFO</b>					
30h	31:0	R	-	RX0_FIFO	I <sup>2</sup> S RX0 频道之接收数据缓存器 16 位数据: [31:16]为右声道, [15:0]为左声道 24 位数据: [23:0]为右声道或左声道 (左声道数据在前) 32 位数据: [31:0]为右声道或左声道 (左声道数据在前)

R/W1C: 读取与写“1”清除

注意: 在主机端模式下取样率进行改变 (即 BCLK 之高/低准位周期发生改变), 或者进行主机/从机端切换之前必须先将 I2S 禁能, 待完成设定改变后再将 I2S 致能。

#### 14.4 功能描述

I2S 模块具有两个时钟讯号即 BCLK (位时钟) 与 LRCK (左右声道时钟), 同时亦具有两个数据讯号线即 DI、DO, 其中每个数据讯号线皆包含两个声道之数据, 因为只有一组 BCLK 与 LRCK 所以同一时间只致能 I<sup>2</sup>S 编码器或 I<sup>2</sup>S 译码器, 除非连接的 DAC 或 ADC CODEC 具有相同的 BCLK 与 LRCK。当为 I<sup>2</sup>S 主机端时, BCLK 是利用系统时钟来产生, 此时 BCLK 的时钟符合 I<sup>2</sup>S 规格 ( $\pm 10\%$ ), BCLK 的高/低周期可透过 I<sup>2</sup>S 控制缓存器来进行设置以产生不同的采样频率 (Sampling Rate)。

### 14.4.1 I2S Bus 的基础

I<sup>2</sup>S 接口支持主机端 (Master) 与从机端 (Slave) 模式。在主机端模式下, BCLK 与 LRCK 的讯号乃由 MCU 所产生, 如图 46 (a); 反之, 在从机模式下, BCLK 与 LRCK 是由与 CPU 连接之 Codec 所提供, 如图 46(b)。

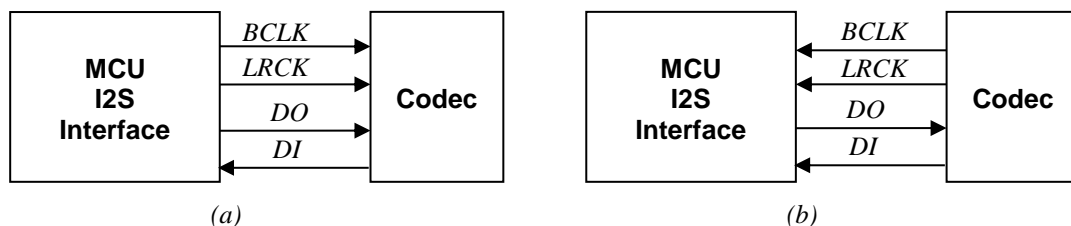


图 46 (a) I2S 主机模式 (b) I2S 从机模式

图 47 为 I<sup>2</sup>S 基本波形, 其中 LRCK 乃于 BCLK 的负缘产生且 LRCK 为 BCLK 的 1/64, 数据讯号线传输于 BCLK 的负缘端且取样于 BCLK 的正缘端。

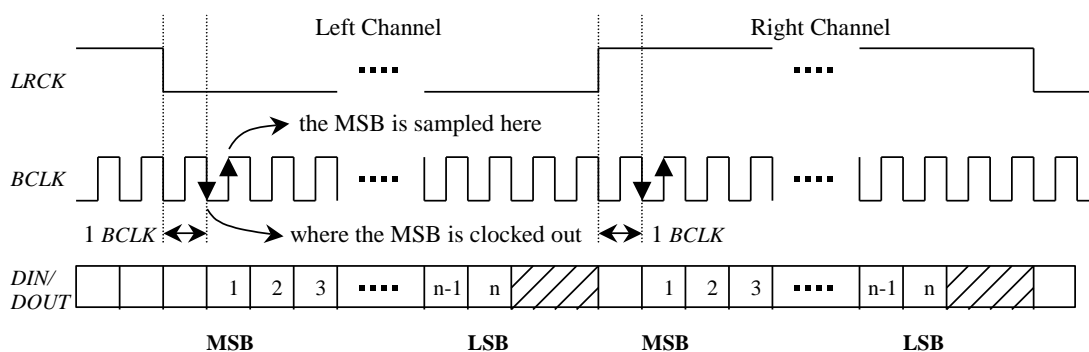


图 47 I2S 接口的基本计时图

针对 I<sup>2</sup>S DAC 控制器, 音频数据在传输前会先将并列格式之数据先转换成序列格式, 接着再由 MSB 开始一个位接着一个位平移输出至 DOUT 讯号在线; 相同的方式, 当从 DI 讯号在线接收来自于 ADC 控制器所输出的串行讯号, MCU 会将其转换为并列数据并储存于数据缓冲区内。

### 14.4.2 左对齐模式

在与 Left Justified 模式下的 I<sup>2</sup>S DAC 控制器连接时, MSB 数据位乃由 MCU 送出于 BCLK 之负缘, 同时对齐于 LRCK 发生转换时; 反之, 在与 Left Justified 模式下的 ADC 控制器连接时, MSB 数据位乃由 ADC 控制器输出, MCU 会于 LRCK 转换后的 BCLK 的正缘进行数据取样。在此模式下, 左声道数据传输时 LRCK 为高准位; 而右声道数据传输时 LRCK 为低准位。下图(图 48)即为此模式之时序图。

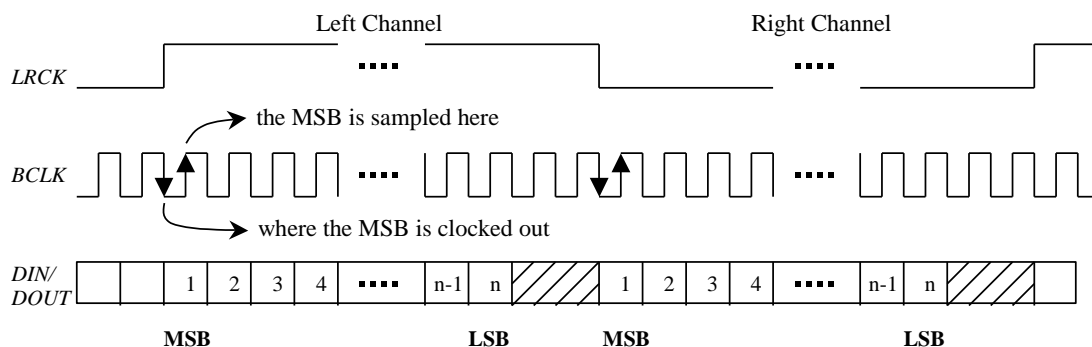


图 48 I2S 接口左对齐模式计时图

### 14.4.3 I2S Mode

在与 I<sup>2</sup>S 模式下的 I<sup>2</sup>S DAC 控制器连接时, MSB 数据位是由 MCU 送出于 LRCK 转换后的第一个 BCLK 之负缘; 反之, MCU 在与 I<sup>2</sup>S 模式下的 ADC 控制器连接时, MSB 数据位是由 ADC 控制器输出, WT32L064/032 会于 LRCK 转换后的第 2 个 BCLK 的正缘进行数据取样。在此模式下, 左声道数据传输时 LRCK 为低准位; 而右声道数据传输时 LRCK 为高准位。下图(图 49)即为此模式之时序图。

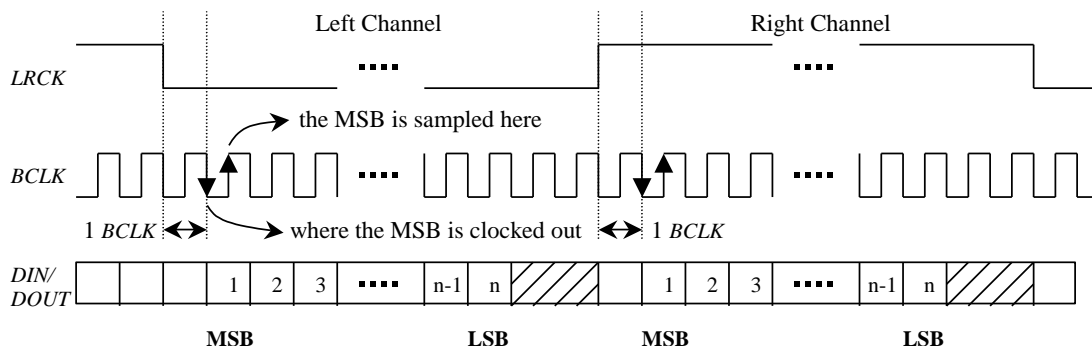


图 49 I2S 接口的 I2S 模式计时图

### 14.4.4 I2S 时钟产生器

在主机端模式下, I<sup>2</sup>S 的位时钟乃由系统时钟进行除频所产生, 相关的 BCLK 高/低准位之周期与取样率的关系如下表(表格 37), 其中计算方式为:

$$\text{Ideal bit clock rate} = (\text{sampling rate}) * (f_{\text{BCLK}} / f_{\text{LRCK}})$$

$$\text{Bit clock rate} = (\text{Main clock}) / ((\text{Clock high duty} + 1) + (\text{Clock low duty} + 1))$$

表格 36 I2S Sampling Rate

Main clock	Sampling rate	Clock high duty setting	Clock low duty setting	$f_{BCLK} / f_{LRCK}$	Bit clock rate	Ideal bit clock
18.432 MHz	48k	5	5	32	1536k	1536k
18.432 MHz	48k	3	3	48	2304k	2304k
18.432 MHz	48k	2	2	64	3072k	3072k
24.576 MHz	48k	7	7	32	1536k	1536k
24.576 MHz	48k	5	4	48	2234k	2304k
24.576 MHz	48k	3	3	64	3072k	3072k

 表格 37 I2S System Clock = 24MHz (assume  $f_{BCLK}=64fs$ )

Sampling rate	Clock high duty	Clock low duty	Bit clock rate	Ideal bit clock
8k	22	22	521.74k	512k
11.025k	16	16	705.88k	705.6k
12k	15	15	750k	768k
16k	11	11	1000k	1024k
22.025k	8	7	1411.76k	1411.2k
24k	7	7	1500k	1536k
32k	5	5	2000k	2048k
44.1k	3	3	3000k	2822.4k
48k	3	3	3000k	3072k
88.2k	1	1	6000k	5644.8k
96k	1	1	6000k	6144k
192k	0	0	12000k	12288k

 表格 38 I2S System Clock = 12MHz (assume  $f_{BCLK} = 64fs$ )

Sampling rate	Clock high duty	Clock low duty	Bit clock rate	Ideal bit clock
8k	11	10	521.74k	512k
11.025k	8	7	705.88k	705.6k
12k	7	7	750k	768k
16k	5	5	1000k	1024k
22.025k	4	3	1411.76k	1411.2k
24k	3	3	1500k	1536k
32k	2	2	2000k	2048k
44.1k	1	1	3000k	2822.4k
48k	1	1	3000k	3072k
88.2k	0	0	6000k	5644.8k
96k	0	0	6000k	6144k

## 15 实时计数器 (Real-Time Clock)

### 15.1 主要概述

系统时钟之实时计数器 (RTC) 是一个独立的 BCD 定时器/计数器。RTC 提供具有可程序设定报警中断的时钟与日历时间。RTC 还包括一个可程序设定的周期中断。缓存器包含秒、分钟、小时 (24 小时格式)、星期几、日期、月份和年份, 以二进制编码十进制格式 (BCD) 表示。自动执行每月 28 天、29 天 (闰年)、30 天和 31 天的效果。

- 带有秒、分钟、小时 (24 小时格式)、星期几、日期、月份和年份的日历。
- 带中断功能的可程序设定报警。报警可以由日历字段的任意组合触发。
- 可程序设定周期中断功能
- 闰年修正
- 需要外部 32.768 kHz 时钟晶体

15.2 方块图

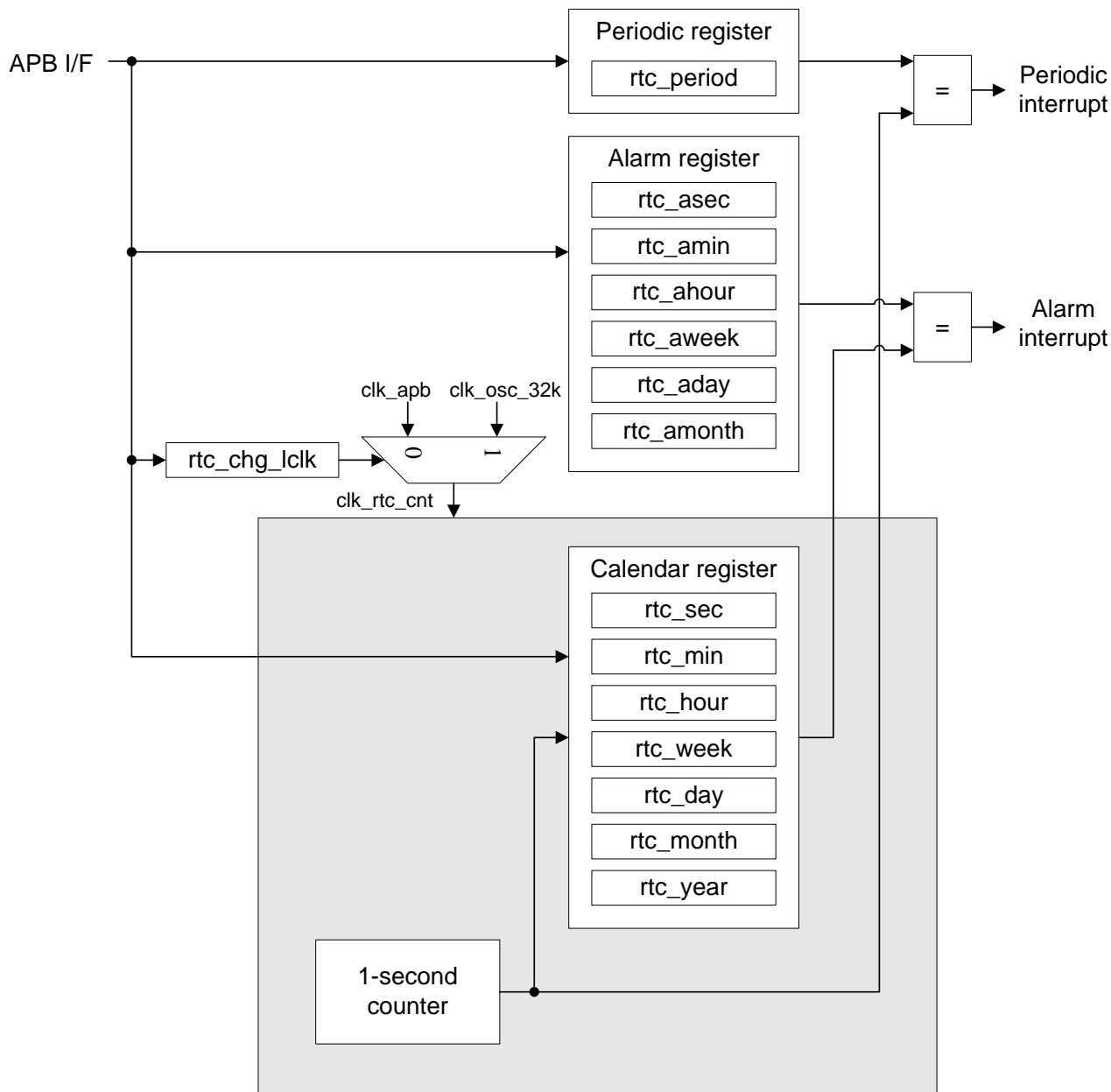


图 50 RTC 方块图

**15.3 缓存器描述**
**表格 39 RTC 控制缓存器**

Index	Bit	R/W	Default	Name	Description
<b>RTC_CR: RTC control</b>					
00h	31:16	-	-	-	保留
	15	R/W	0	rtc_en	RTC 致能 0: 禁能 RTC 1: 致能 RTC
	14:10	-	-	-	保留
	9	R/W	0	rtc_lse_en	选择 RTC 时钟源, LSE: 32768 Hz 或 LSI:37 kHz。 0: LSI 1: LSE
	8	R/W	0	rtc_chg_lclk	启用 RTC 时钟=32 kHz 低频时钟。 0: 可以读取/写入 RTC 缓存器。(缓存器和时钟源同步, 所以可正常读取/写入) 1: 无法读取/写入 RTC 缓存器。
	7:6	-	-	-	保留
	5:4	R/W	0	rtc_period	RTC 周期中断时间设置 0: 1 s (1Hz) 1: 62.5 ms (16Hz) 2: 31.25 ms (32Hz) 3: 15.625 ms (64Hz)
	3:2	-	-	-	保留
	1	R/W	0	rtc_alarm_ie	启用 RTC 报警中断。 0: 禁能报警中断。 1: 致能报警中断。
0	R/W	0	rtc_period_ie	启用 RTC 定期中断。 0: 禁能周期中断。 1: 致能周期中断。	
<b>RTC_ISR: RTC interrupt status register</b>					
04h	30:2	-	-	-	保留
	1	R/W1c	0	rtc_alarm_if	RTC 报警中断旗标。 0: 无报警中断事件。 1: 有报警中断事件。
	0	R/W1c	0	rtc_period_if	RTC 周期中断旗标。 0: 无周期中断事件。 1: 有周期中断事件。
<b>RTC_TIME: RTC time setting</b>					
10h	31:22	-	-	-	保留
	21:16	R/W	0	rtc_hour	在 BCD 中编码的小时, 范围为 0~23。 Rtc_hour[5:4]:十小时单元。 Rtc_hour [3:0]:小时单位。
	15	-	-	-	保留
	14:8	R/W	0	rtc_min	在 BCD 中编码的分钟数, 范围为 0~59。 Rtc_min[6:4]: 十分钟单元。 Rtc_min[3:0]: 分钟单位。
	7	-	-	-	保留

Index	Bit	R/W	Default	Name	Description
	6:0	R/W	0	rtc_sec	在 BCD 中进行第双步编码，范围为 0~59。 Rtc_sec[6:4]: 十秒单元。 Rtc_sec[3:0]: 秒单元。
<b>RTC_DATE: RTC date setting</b>					
14h	31:24	R/W	0	rtc_year	年份编码在 BCD 中，范围为 0~99。 Rtc_year[7:4]: 十年单位。 Rtc_year[3:0]: 年单位。
	23:21	-	-	-	保留
	20:16	R/W	1	rtc_month	月份编码在 BCD 中，范围为 1~12。 Rtc_month[4]: 十月单位。 Rtc_month[3:0]: 月单位。
	15:14	-	-	-	保留
	13:8	R/W	1	rtc_day	在 BCD 中编码的日数，范围为 1~31。 Rtc_day[5:4]: 十日单位。 Rtc_day[3:0]: 日单位。
	7:3	-	-	-	保留
	2:0	R/W	0	rtc_week	以 BCD 编码的周日，范围为 0~6。 0: 星期日 1: 星期一 2: 星期二 3: 星期三 4: 星期四 5: 星期五 6: 星期六
<b>RTC_A_TIME: RTC alarm time setting</b>					
18h	31:23	-	-	-	保留
	22	R/W	0	rtc_ahour_en	致能报警小时。 0: 禁能。 1: 致能报警设置，小时单位匹配。
	21:16	R/W	0	rtc_ahour	报警小时编码为 BCD，范围为 0~23。 Rtc_ahour[5:4]: 十小时单位。 Rtc_ahour[3:0]: 小时单位。
	15	R/W	0	rtc_amin_en	致能报警分钟。 0: 禁能。 1: 致能报警设置，分钟单位匹配。
	14:8	R/W	0	rtc_amin	报警分钟编码为 BCD，范围为 0~59。 Rtc_amin[6:4]: 十分钟单位。 Rtc_amin[3:0]: 分钟单位。
	7:0	-	-	-	保留
<b>RTC_A_DATE: RTC alarm date setting</b>					
1ch	31:22	-	-	-	保留
	21	R/W	0	rtc_amonth_en	致能报警月份。 0: 禁能。 1: 致能报警设置，月单位匹配。
	20:16	R/W	1	rtc_amonth	报警月份编码为 BCD，范围为 1~12。 Rtc_amonth [4]: 十月单位。 Rtc_amonth [3:0]: 月单位。
	15	-	-	-	保留



Index	Bit	R/W	Default	Name	Description
	14	R/W	0	rtc_aday_en	致能报警日期。 0: 禁能。 1: 致能报警设置, 日期单位匹配。
	13:8	R/W	1	rtc_aday	报警日编码为 BCD, 范围为 1~31。 Rtc_aday[5:4]: 十天单位。 Rtc_aday[3:0]: 日单位。
	7:4	-	-	-	保留
	3	R/W	0	rtc_aweek_en	启用报警周。 0: 禁能。 1: 致能报警设置, 星期单位匹配。
	2:0	R/W	0	rtc_aweek	报警周日编码为 BCD, 范围为 0~6。 0: 星期日。 1: 星期一。 2: 星期二。 3: 星期三。4: 星期四。5: 星期五。6: 星期六。
<b>RTC_LSE_CFG: RTC LSE configuration</b>					
20h	31:14	-	-	-	保留
	13:12	R/W	3h	rtc_lse_cur	选择 Bias Current Ratio to X'tal cir. Block for LSE X'tal pad. 3h: 默认值。
	11:10	-	-	-	保留
	9:8	R/W	3h	rtc_lse_bgo	从 Bandgap block 选择 LSE X'tal pad 的偏置电流。 3h: 默认值。
	7	-	-	-	保留
	6:4	R/W	0	rtc_lse_envt	选择 LSE X'tal pad 的 X'tal 反相的阈值电压驱动电平。 000: 默认值。
	3:2	-	-	-	保留
	1:0	R/W	0	rtc_lse_ed	选择 LSE X'tal pad 的驱动电流。00:默认值。

## 15.4 功能描述

### 15.4.1 可程序设定闹钟 (Programmable Alarm)

可利用 rtc\_alarm\_ie[0]寄存器启用可程序设定报警功能。如果日历分钟、小时、日期或星期与报警寄存器 (18h 和 1ch 寄存器)中程序设定的值匹配, 则 rtc\_alarm\_if[0] 设置为 1。每个日历字段可以利用这些寄存器的启用位 (rtc\_amin\_en [0]、rtc\_ahour\_en[0]、rtc\_aweek\_en[0]等)独立选择。利用 00h 寄存器中的 rtc\_alarm\_ie 位可启用报警中断。

### 15.4.2 周期中断 (Periodic Interrupt)

周期中断旗标由内部 15-bit, 1 秒计数器产生。通过设定 rtc\_period\_ie[0]致能周期中断功能, 在 00h 寄存器中。定期中断定时器可程序设定为产生 1s、62.5ms、31.25ms 和 15.63 ms 周期唤醒时间。

## 16 独立看门狗定时器

### 16.1 主要概述

独立看门狗 (IWDT) 有其自己的专用低速时钟 37 kHz 计时, 因此即使主时钟发生故障, 也能保持活动状态。IWDT 最适合需要看门狗作为主应用程序之外的完全独立运行的应用程序, 但时序精度限制较低。

- 自由运行的向下计数器
- 使用独立振荡器 (37 kHz) 时钟源
- 当向下计数器到达 000h 值时复位 (如果启用了看门狗)

### 16.2 方块图

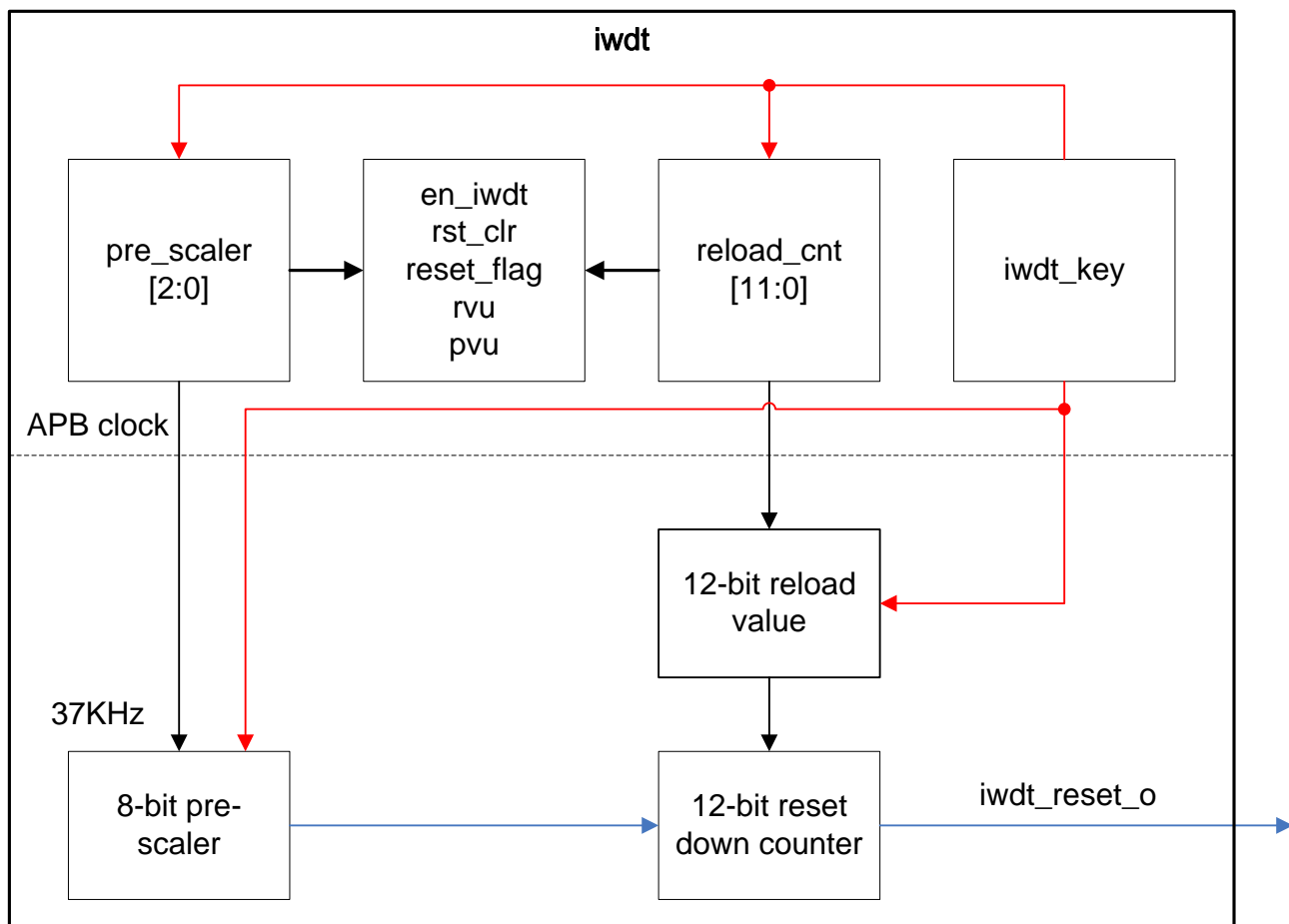


图 51 IWDT 方块图

**16.3 缓存器描述**

表格 40 IWDT 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>IWDT_KEY: IWDT key register</b>					
00h	31:16	-	-	-	保留
	15:0	W	-	iwdt_key	仅允许写入的缓存器，必须由软件定期编写以重载复位计数器，否则看门狗会在复位计数器达到 0 时产生复位。 Ccccch: 致能 IWDT (复位计数器启动) aaaah: 重载复位计数器和预除器复位 5555h: 致能存取预除器和复位计数器缓存器 (解除保护) 注意: 此缓存器访问之间的时间间隔必须至少为 60μs。
<b>IWDT_STS: IWDT status register</b>					
04h	31:8	-	-	-	保留
	7	R	0	en_iwdt	0: IWDT 致能 1: IWDT 禁能
	6:5	-	-	-	保留
	4	R	0	reload_work	0: 准备下次重置计数器重载。 1: 上一次重载过程正在进行，下次重置计数器重载无效。
	3	-	-	-	保留
	2	-	-	-	保留
	1	R	0	rvu	计数器重载值更新。 此位由硬件设置，以指示正在更新重载值。当重载值更新操作完成时，由硬件重置
	0	R	0	pvu	预除器值正在载入更新。 此位由硬件设置，以指示正在进行预除器值的更新。完成预除器更新操作后，由硬件复位
<b>IWDT_PRE_SCALER: IWDT pre-scaler for down counter</b>					
08h	31:3	-	-	-	保留
	2:0	R/W	0	pre_scaler	时钟预除器为向下计数器。此缓存器是受 iwdt_key[15:0] = 5555h 才能存取。 0: divide 4 1: divide 8 2: divide 16 3: divide 32 4: divide 64 5: divide 128 6, 7: divide 256 注意: pvu[0] 位必须复位后才能更改 pre-scaler[2:0]。
<b>IWDT_RELOAD: IWDT down counter reload value</b>					
0ch	31:12	-	-	-	保留
	11:0	R/W	FFFh	reload_cnt	向下计数器重载值。 此缓存器是受 iwdt_key[15:0] = 5555h 才能存取。它们由软件编写，用于定义每次在 iwdt_key 缓存

Index	Bit	R/W	Default	Name	Description
					器中写入值 <code>aaaah</code> 时要载入到看门狗计数器中的值。复位计数器从此值倒数计时。超时时间是此值和时钟预除器的函数。 <code>Rvu[0]</code> 位必须复位后, 才能更改重载值。

## 16.4 功能描述

当 IWDT 藉由在 KEY 缓存器中写入值 `cccch` (`iwdt_key[15:0]`)开始时, 计数器将从 `FFFh` 的复位值开始倒计。当它到达计数值(`000h`)末端时, 将产生复位信号 (IWDT 复位)。每当在 `iwdt_key[15:0]` 缓存器中写入 `aaaah` 时, `reload_cnt[11:0]` 值将重载到计数器中, 并防止看门狗复位。

对 `pre_scaler[2:0]` 和 `reload_cnt[11:0]` 缓存器的存取受到保护。要修改它们, 必须先写入 `iwdt_key[15:0]` 的代码 `5555h`。把不同值的写入此缓存器, 缓存器存取将再次受到保护。这意味着重载时 (写入 `aaaah`)的情况。状态缓存器 (`rvu[0]` 和 `pvu[0]`)可用于指示正在进行 `pre_scaler[2:0]` 或下数计数器重载正进行中。

37kHz 的最小/最大复位时间如下。

表格 41 IWDT Min./Max. 超时周期 使用 37 kHz 时钟源

<code>pre_scaler[2:0]</code>	Minimum timeout (ms) <code>reload_cnt[11:0]=000h</code>	Maximum timeout (ms) <code>reload_cnt[11:0]=fffh</code>
0	0.108	442.810
1	0.216	885.622
2	0.432	1,771.243
3	0.865	3,542.456
4	1.730	7,084.973
5	3.459	14,169.946
6 or 7	6.919	28,339.892

## 17 窗口看门狗定时器

### 17.1 主要概述

“窗口看门狗定时器” (WWDT) 通常用来监测由外部干扰或不可预见的逻辑条件造成应用程序背离正常的运行序列所产生的软件故障。如果递减计数器的值在位 6 变成 0 之前未被更新, 则窗口看门狗定时器会产生一个 MCU 复位。递减计数器的值在达到窗口缓存器的值之前被更新, 也会产生一个 MCU 复位。这表示窗口看门狗定时器的递减计数器需要在一个有限的时间窗口被更新。

- 可程序自由运行递减计数器
- 复位条件:
  - 当递减计数器值小于 40h 时复位 (如果 WWDT 致能)
  - 如果递减计数器在窗口外被重载 (如果 WWDT 致能)
- 早期唤醒中断 (EWI): 当向下计数器等于 40h 时触发 (如果致能 WWDT)。此功能可用于重载递减计数器并防止 WWDT 复位。

### 17.2 方块图

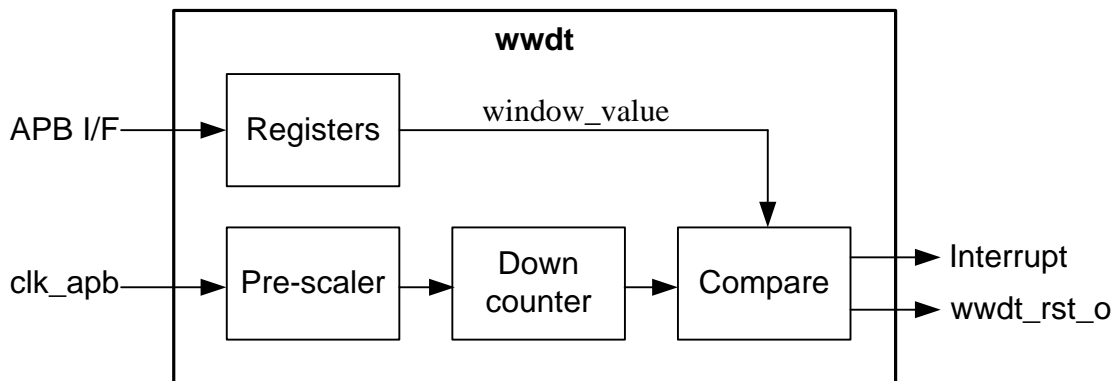


图 52 WWDT 方块图

### 17.3 缓存器描述

表格 42 WWDT 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>WWDT_CR1: WWDT control register 1</b>					
00h	31	R/W	0	wwdt_enable	致能 WWDT function
	30:7	-	-	-	保留
	6:0	R/W	7fh	down_cnt_time	递减计数器 (仅有效 wwdt_enable = 1)。Down_cnt_time = 3fh 时启动。
<b>WWDT_CR2: WWDT control register 2</b>					

Index	Bit	R/W	Default	Name	Description
04h	31:13	-	-	-	保留
	12	R/W	0	early_int_en	早期中断启用。 0:禁能早期 wwdt 中断。 1:致能早期 wwdt 中断。
	11:8	R/W	0	time_base	时钟基频 (clk_apb) 除以 4096 , 除以 2**{time_base}。 注意: time_base 应设置为 0~8。
	7	-	-	-	保留
	6:0	R/W	7fh	window_value	窗口值供递减计数器比较用
<b>WWDT_INT_F: WWDT early interrupt flag</b>					
08h	31:1	-	-	-	保留
	0	R/W1c	0	early_int_flag	早期中断旗标 ( 重置递减计数器= 40h )

RW1C: 读取 与 写“1”清除

## 17.4 功能描述

如果启用了 WWDT，并且当递减计数器从 40h 减到 3fh 时，它将产生一个复位。如果软件在计数器值大于窗口寄存器中的数值时重新装载递减计数器，也将产生一个复位。应用程序在正常运行过程中必须定期的写入 down\_cnt\_time 寄存器以防止发生复位。只有当递减计数器值小于窗口值寄存器(window\_value)的值时，才能进行写操作。储存在down\_cnt\_time 寄存器中的数值必须在ffh 和 c0h之间。

- 致能 WWDT: 系统复位后, WWDT 它通过设置 (wwdt\_enable = 1) 寄存器被致能。致能后, 除非复位, 否则不会再次禁能它。
- 递减计数器控制: 致能 WWDT 时, 必须设置 down\_cnt\_time[6]以防止立即复位。Down\_cnt\_time[5:0]位包含了 WWDT 产生复位之前的计数值。由于写入 time\_base 寄存器时预除器的未知状态, 因此计时在最小值和最大值之间有所不同。Window\_value 寄存器包含窗口的上限。为了防止复位, 当向下计数器的值低于窗口寄存器值且在设置 WWDT 致能后大于 3fh 时, 必须重载向下计数器 (wwdt\_enable = 1)。重载计数器的另一种方法是使用早期中断。当向下计数器达到值 40h 时, 将产生此中断, 并且可以使用相应的中断服务程序重载向下计数器以防止 WWDT 复位。通过设置 (early\_int\_flag)寄存器清除中断。软件复位也可以通过清除 clear\_cnt\_time[6] 当 WWDT 被启动(wwdt\_enable=1)
- 程序设定看门狗超时: WWDT 的计时图如图 53 所示。下面是计算超时值的公式:

$$T_{\text{WWDT}} = T_{\text{clk\_apb}} \times 4096 \times 2^{\text{time\_base}} \times (\text{down\_cnt\_time}[5:0] + 1) \text{ (ms)}$$

24.00MHz 时最小/最大超时值如表 44 所示。

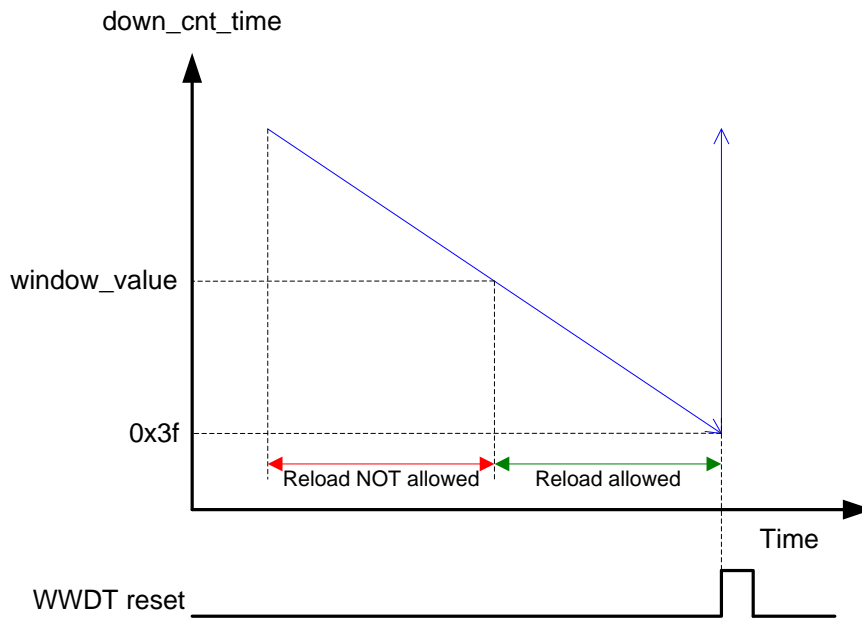


图 53 Timing Diagram of WWDT

表格 43 WWDT 最小值/最大超时值(Timeout), 24.00MHz

time_base	Minimun Timeout ( $\mu$ s) down_cnt_time = 40	Maximum Timeout (ms) down_cnt_time = 7f
0	170.67	10.92
1	341.33	21.85
2	682.67	43.69
3	1365.33	87.38
4	2730.66	174.76
5	5461.32	349.52
6	10922.64	699.04
7	21845.28	1398.08
8	43690.56	2796.16

## 18 PWM

### 18.1 主要概述

“脉冲宽度调制”(PWM)模块

- 四个独立周期 PWM
  - 独立用户定义的周期和占空比 (Duty)
  - 占空比 (Duty)范围从 0/65536 到 65535/65536
- 支持 PWM 占空比的双缓冲功能
- 一个中断输出

方块图

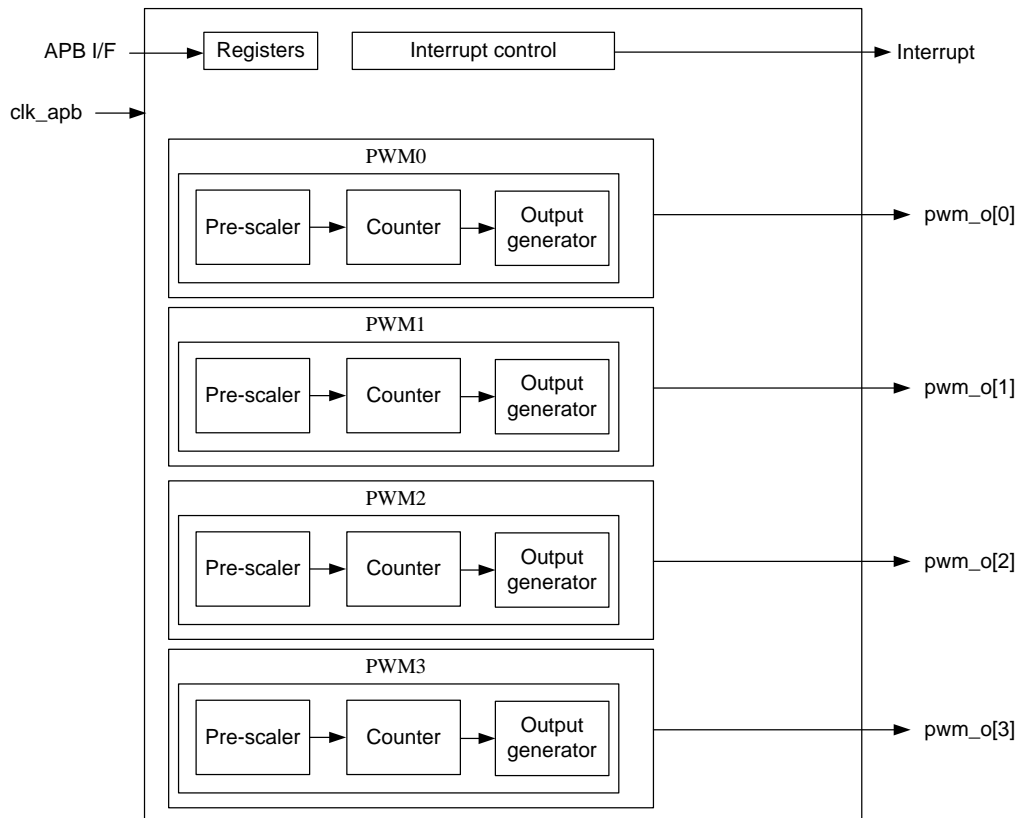


图 54 PWM 方块图



**18.2 缓存器描述**
**表格 44 PWM 控制缓存器**

Index	Bit	R/W	Default	Name	Description
<b>PWM_OEN: PWM 输出致能 &amp; 中断</b>					
00h	31:8	R/W	0	PWMO_EN	PWMx 输出设置位。 0: 禁能 PWMx 输出 1: 致能 PWMx 输出
	27:24	R/W	0	PWMIE	PWMx 中断设置位。 0: 禁能 PWMx 中断 1: 致能 PWMx 中断
	3:0	R/W	0	PWMIF	PWMx 中断旗标, 写入 1 以清除此位。 0: 无发生 PWMx 中断 1: 发生 PWMx 中断
<b>PWM_CR0: PWM 控制缓存器 0</b>					
04h	31	R/W	0	PWMS0	0: 停止 PWM0 功能 (默认值) 1: 启动 PWM0 功能
	30:4	-	-	-	保留
	3	R/W	0	PWMDB0	双缓冲功能。 0: 立即应用新的 PWMDV0/PWMCV0。 1: 在下一个 PWM 周期应用新的 PWMDV0/PWMCV0。
	2:0	R/W	0	PWMCK0	系统时钟的预分频器作为 PWM0 源时钟。 000: /2 <sup>11</sup> (默认值) 001: /2 <sup>9</sup> 010: /2 <sup>7</sup> 011: /2 <sup>5</sup> 100: /2 <sup>3</sup> 101: /2 <sup>2</sup> 110: /2 111: /1
<b>PWM_PWMCV0: PWM0 的周期设定</b>					
08h	31:16	-	-	-	保留
	15:0	R/W	0	PWMCV0	PWM0 的周期设置
<b>PWM_PWMDV0: PWM0 输出的 Duty 设置</b>					
0Ch	31:16	-	-	-	保留
	15:0	R/W	0	PWMDV0	PWM0 输出的 Duty 设置
<b>PWM_CR1: PWM 控制缓存器 1</b>					
10h	31	R/W	0	PWMS1	0: 停止 PWM1 功能 (默认值)。 1: 启动 PWM1 功能。
	30:4	-	-	-	保留
	3	R/W	0	PWMDB1	双缓冲功能。 0: 立即应用新的 PWMDV1/PWMCV1 1: 在下一个 PWM 周期应用新的 PWMDV1/PWMCV1
	2:0	R/W	0	PWMCK1	系统时钟的预分频器作为 PWM1 源时钟。 000: /2 <sup>11</sup> (默认值) 001: /2 <sup>9</sup>

Index	Bit	R/W	Default	Name	Description
					010: /2 <sup>7</sup> 011: /2 <sup>5</sup> 100: /2 <sup>3</sup> 101: /2 <sup>2</sup> 110: /2 111: /1
<b>PWM_PWMCV1: PWM1 的周期设置</b>					
14h	31:16	-	-	-	保留
	15:0	R/W	0	PWMCV1	PWM1 的周期设置
<b>PWM_PWMDV1: PWM1 输出的 Duty 设置</b>					
18h	31:16	-	-	-	保留
	15:0	R/W	0	PWMDV1	PWM1 输出的 Duty 设置
<b>PWM_CR2: PWM 控制寄存器 2</b>					
1Ch	31	R/W	0	PWMS2	0: 停止 PWM2 功能 (默认值)。 1: 启动 PWM2 功能。
	30:4	-	-	-	保留
	3	R/W	0	PWMDV2	双缓冲功能。 0: 立即应用新的 PWMDV2/PWMCV2 1: 在下一个 PWM 周期应用新的 PWMDV2/PWMCV2。
	2:0	R/W	0	PWMCK2	系统时钟的预分频器作为 PWM2 时钟源。 000: /2 <sup>11</sup> (default) 001: /2 <sup>9</sup> 010: /2 <sup>7</sup> 011: /2 <sup>5</sup> 100: /2 <sup>3</sup> 101: /2 <sup>2</sup> 110: /2 111: /1
<b>PWM_PWMCV2: PWM2 的周期设置</b>					
20h	31:16	-	-	-	保留
	15:0	R/W	0	PWMCV2	PWM2 的周期设置
<b>PWM_PWMDV2: PWM2 输出的 Duty 设置</b>					
24h	31:16	-	-	-	保留
	15:0	R/W	0	PWMDV2	PWM2 输出的 Duty 设置
<b>PWM_CR3: PWM 控制寄存器 3</b>					
28h	31	R/W	0	PWMS3	0: 停止 PWM3 功能 (默认值)。 1: 启动 PWM3 功能。
	30:4	-	-	-	保留
	3	R/W	0	PWMDV3	双缓冲功能。 0: 立即应用新的 PWMDV3/PWMCV3 1: 在下一个 PWM 周期应用新的 PWMDV3/PWMCV3
	2:0	R/W	0	PWMCK3	系统时钟的预分频器作为 PWM3 时钟源。 000: /2 <sup>11</sup> (default) 001: /2 <sup>9</sup> 010: /2 <sup>7</sup> 011: /2 <sup>5</sup> 100: /2 <sup>3</sup> 101: /2 <sup>2</sup>

Index	Bit	R/W	Default	Name	Description
					110: /2 111: /1
<b>PWM_PWMCV3: PWM3 的周期设置</b>					
2Ch	31:16	-	-	-	保留
	15:0	R/W	0	PWMCV3	PWM3 的周期设置
<b>PWM_PWMDV3: PWM3 输出的 Duty 设置</b>					
30h	31:16	-	-	-	保留
	15:0	R/W	0	PWMDV3	PWM3 输出的 Duty 设置

### 18.3 功能描述

#### 18.3.1 独立 PWM

Duty 范围: 0/65536 to 65535/65536

如果  $PWMDVx \leq PWMCVx$ , 则 PWMx 输出低准位

如果  $PWMDVx > PWMCVx$ , 则 PWMx 输出高准位

一般模式:

$PWMCVx \geq 1$  and  $PWMDVx \geq 1$

$PWMDVx \leq PWMCVx$

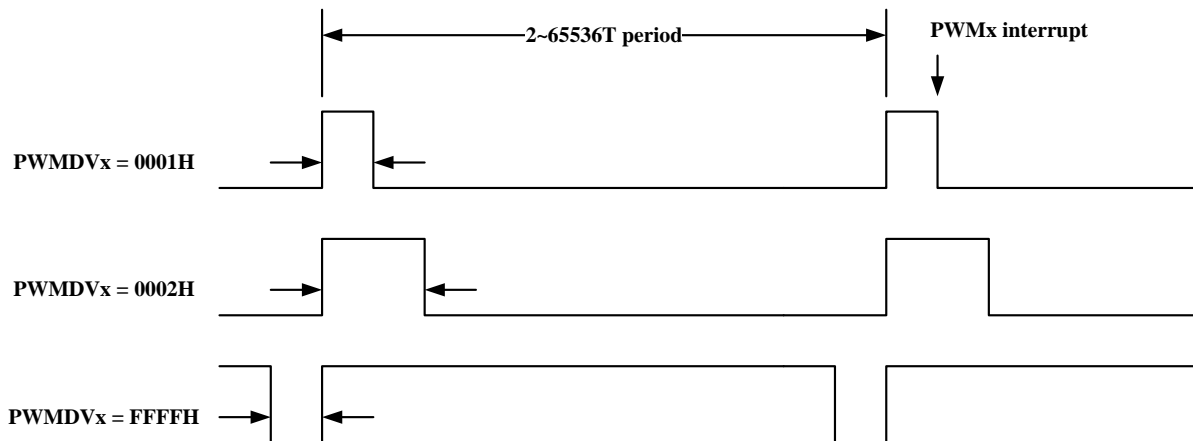


图 55 PWM0~3 功能

## 19 定时器

### 19.1 主要概述

通用定时器由一个可程序设定、预缩放(pre-scaler)驱动的 16 bit 计数器组成，它们是用来计算外围时钟 (PCLK) 或外部时钟的周期，并可用于各种用途，包括测量输入的脉冲长度信号 (输入捕获)或基于匹配缓存器产生输出波形 (输出比较和 PWM)。定时器还可以作为系统要求产生中断或 DMA 请求。

- 带 1 个可程序设定 16-bit 预除器(prescaler)的 16-bit 定时器/计数器
- 计数器或定时器操作
- 每个定时器有两个 16-bit 捕获通道，包括 PWM 输入检测
- 同步电路，用于使用外部信号和其它定时器控制定时器
- 在以下事件上中断/DMA 产生：
  - Input capture
  - Output match
- 四个 16-bit 匹配缓存器 (Match Register)，具以下功能：
  - 连续操作，匹配时可选产生中断
  - 与可选的中断产生匹配时停止定时器
  - 与可选的中断产生匹配时复位定时器
- 两个外部输出对应于匹配缓存器，具以下功能：
  - Set low on match
  - Set high on match
  - Toggle on match
- 配对匹配缓存器的组合产生 PWM 输出

## 19.2 方块图

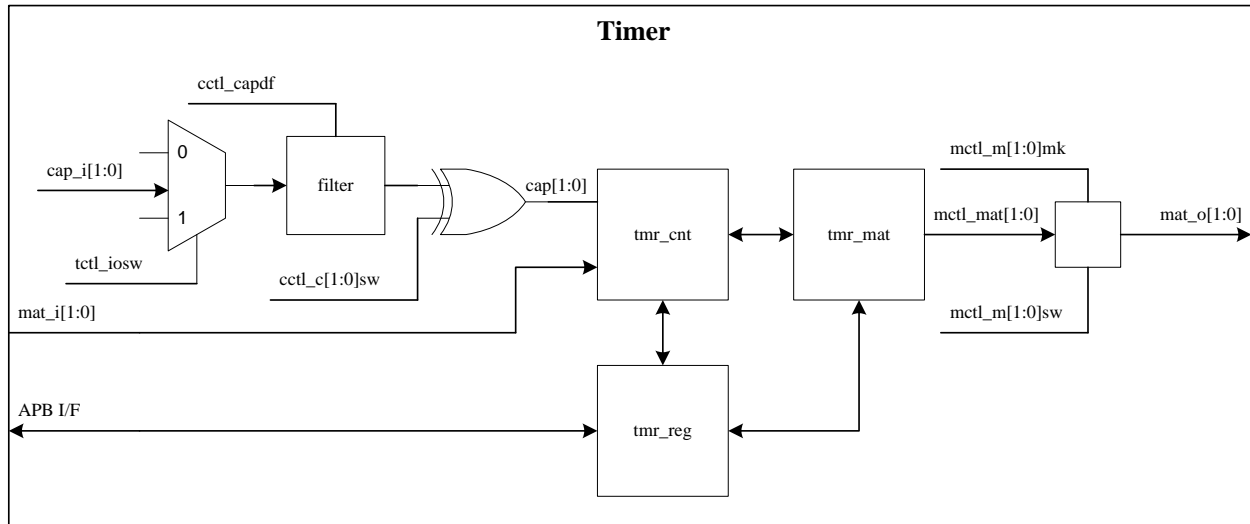


图 56 Timer Module 方块图

注意:

1. tmr[n].matox 连接至 tmr[n+1].matix 与输出 pads
2. capix 连接至输入 pads
3. 假设 tctl\_iosw = '0', capture input = cap\_i[1:0] = TMR\_CH[1:0],  
match output = mat\_o[1:0] = TMR\_CH[3:2]  
假设 tctl\_iosw = '1', capture input = cap\_i[1:0] = TMR\_CH[3:2],  
match output = mat\_o[1:0] = TMR\_CH[1:0]

## 19.3 缓存器描述

表格 45 Timer 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>TIMER_TCTL: Timer control register</b>					
00h	15	R/W	0	tctl_en	Timer 致能
	14:12	-	-	-	保留
	11:10	R/W	0	tctl_clkssel	Timer 时钟源选择. 00: clk_apb 01: clk_msi 10: clk_hsi 11: tmr_cap0 备注: Timer1 可使用上述 b'00~b'11 模式, Timer0/2 只可使用 b'00 模式 APB 时钟源。
	9	R/W	0	tctl_matisw	mati 来源交换. 0: mati = 原始匹配输入. 1: mati[1:0] = {lse_clk, lsi_clk}
	8	R/W	0	tctl_matco	mat0/mat1 控制关闭. 0: 写 mctl 相应地更改 mat1/mat0 值 1: 写入 mctl 不会影响 mat1/mat0 值。

本文件为伟诠电子股份有限公司机密数据，未经许可不得擅自复印或备份。

Index	Bit	R/W	Default	Name	Description
	7	R/W	0	tctl_de	调试启动。 0: 侦错模式下的定时器保持。 1: 定时器在侦错模式下工作正常。
	6	R/W	0	tctl_iosw	I/O 交换。 0: 无动作。 1: 交换 capix/matox I/O 定义和方向
	5:4	R/W	0	tctl_sel	counter 输入选择 00: cap0 01: cap1 10: mati0 11: mati1
	3:2	R/W	0	tctl_mode	计数器/定时器模式。 00: 定时器。 01: 上升边缘计数器。 10: 下降边缘计数器。 11: 双边计数器。
	1	R/W	0	tctl_rst	计数器/定时器复位 0: 定时器无动作 1: 复位{tcnt, pcnt}
	0	R/W	0	tctl_st	Timer 启动/停止控制 0: 停止 1: 启动
<b>TIMER_TCNT: Timer counter 缓存器</b>					
04h	15:0	R/W	0	tcnt	TCNT Timer 计数值等于预除器 PCNT 溢位触发的累进值, 请参考下面示意图
<b>TIMER_PSCL: Timer 预除器设定</b>					
08h	15:0	R/W	0	Psc1	预除器设定值 PCSL, 请参考下面示意图
<b>TIMER_PCNT: 预除器计数值缓存器</b>					
0ch	15:0	R/W	0	pcnt	预除器计数值 PCNT 等于时钟输入源比较 PSCL 设定范围, 超过后溢位让 TCNT 累进并归零, 请参考下面示意图
<b>TIMER_CCTL: capi 数字滤波器设置</b>					
10h	15:12	-	-	-	保留
	11:10	R/W	0	cctl_capdf	capi0 & capi1 的数字滤波器设置 00: 无滤波器 (1 时钟源) 01: 4 时钟源 10: 16 时钟源 11: 64 时钟源
	9	R/W	0	cctl_c1sw	invert capi1 for cap1
	8	R/W	0	cctl_c0sw	invert capi0 for cap0
	7:4	-	-	-	保留
	3	R/W	0	cctl_cap1f	在 cap 1 下降边缘启动捕获 (储存在 cap1f 中)
	2	R/W	0	cctl_cap1r	在 cap 1 上升边缘启动捕获 (储存在 cap1r 中)
	1	R/W	0	cctl_cap0f	在 cap 0 下降边缘启动捕获 (储存在 cap0f 中)
0	R/W	0	cctl_cap0r	在 cap0 上升边缘启动捕获 (储存在 cap0r 中)	
<b>TIMER_CICTL: capi 控制</b>					
14h	15:12	-	-	-	保留
	11:10	R/W	0	cictl_c1actx	cap1 在计数器上的扩展操作

Index	Bit	R/W	Default	Name	Description
					00: 无动作 01: 保持(hold)计数器于 cap1 低位 (low level) 10: 复位计数器于 cap1 下降边缘 (falling edge) 11: 保留
	9:8	R/W	0	cictl_c0actx	在计数器 cap 0 上扩展操作
	7:6	R/W	0	cictl_m1act	在计数器 mati1 上操作 (注-3)。 00: 无动作。 01: 在 mati1 高电平上保持计数器。 0: 在 mati1 上升边缘重置计数器。 11: 在 mati1 上升边缘 (设置 tctl_st)上的触发计数器。
	5:4	R/W	0	cictl_m0act	在计数器上操作 mati0
	3:2	R/W	0	cictl_c1act	在计数器上操作 cap1。 00: 无动作。 01: 在 cap1 高电平上保持计数器。 10: 在 cap1 上升边缘重置计数器。 11: cap1 上升边缘(设置 tctl_st)上的触发计数器。
	1:0	R/W	0	cictl_c0act	cap0 operation on counter
<b>TIMER_MCTL: mat 控制</b>					
18h	15:14	R/W	0	mctl_m1mk	matm1 output mask (note-4) 00: 和 mat1 相同 01: 保留 10: 清除 matm1 if cap0 = 1 11: 清除 matm1 if cap1 = 1
	13:12	R/W	0	mctl_m0mk	matm0 输出屏蔽 (mask)
	11	R/W	0	mctl_mat1	mat1 value
	10	R/W	0	mctl_mat0	mat0 value
	9	R/W	0	mctl_m1sw	invert mat1 for mato1
	8	R/W	0	mctl_m0sw	invert mat0 for mato0
	7:6	R/W	0	mctl_mat1b	匹配 mat1b (note-5)的控制。 00: 无操作。 01: 将相应的外部匹配位清除为 0。 10: 将相应的外部匹配位设置为 1。 11: 切换相应的外部匹配位。 注意: 使用 b'01 或 b'10 设置, 将对应到 moctl_m1br 执行复位或是 mat1a > mat1b 执行触发结果。
	5:4	R/W	0	mctl_mat1a	mat1a 的匹配控制
	3:2	R/W	0	mctl_mat0b	mat0b 的匹配控制
	1:0	R/W	0	mctl_mat0a	mat0a 的匹配控制
<b>TIMER_MOCTL: mo 控制</b>					
1ch	15:8	-	-	-	保留
	7	R/W	0	moctl_m1bs	mat1b 停止计数器 (清除 tctl_st)
	6	R/W	0	moctl_m1br	mat1b 复位计数器
	5	R/W	0	moctl_m1as	mat1a 停止计数器
	4	R/W	0	moctl_m1ar	mat1a 复位计数器
	3	R/W	0	moctl_m0bs	mat0b 停止计数器
	2	R/W	0	moctl_m0br	mat0b 复位计数器

Index	Bit	R/W	Default	Name	Description
	1	R/W	0	mocctl_m0as	mat0a 停止计数器
	0	R/W	0	mocctl_m0ar	mat0a 复位计数器
<b>TIMER_MAT0A: mat0a/ cap0r 缓存器</b>					
30h	15:0	R/W	0	mat0a (cap0r)	匹配输出匹配模式的 mat0a 缓存器。 用于输入捕获模式的捕获上限缓存器。
<b>TIMER_MAT0B: mat0b/ cap0f 缓存器</b>					
34h	15:0	R/W	0	mat0b (cap0f)	match mat0b 缓存器
<b>TIMER_MAT1A: mat1a/ cap1r 缓存器</b>					
38h	15:0	R/W	0	mat1a (cap1r)	match mat1a 缓存器
<b>TIMER_MAT1B: mat1b/ cap1f 缓存器</b>					
3ch	15:0	R/W	0	mat1b (cap1f)	match mat1b 缓存器
<b>TIMER_OFIF: 溢出及中断旗标</b>					
40h	15	R/W1c	0	of_tcnt	tcnt 的溢出旗标 (tcnt loop from ffffh to 0000h) (note-6)
	14:12	-	-	-	保留
	11	R/W1c	0	of_cap1f	cap1f 的溢出旗标 (twice or more interrupts occur)
	10	R/W1c	0	of_cap1r	cap1r 的溢出旗标
	9	R/W1c	0	of_cap0f	cap0f 的溢出旗标
	8	R/W1c	0	of_cap0r	cap0r 的溢出旗标
	7	R/W1c	0	if_mat1b	mat1b 的中断旗标
	6	R/W1c	0	if_mat1a	mat1a 的中断旗标
	5	R/W1c	0	if_mat0b	mat0b 的中断旗标
	4	R/W1c	0	if_mat0a	mat0a 的中断旗标
	3	R/W1c	0	if_cap1f	cap1f 的中断旗标
	2	R/W1c	0	if_cap1r	cap1r 的中断旗标
	1	R/W1c	0	if_cap0f	cap0f 的中断旗标
	0	R/W1c	0	if_cap0r	cap0r 的中断旗标
<b>TIMER_IE: 中断启动</b>					
44h	15:8	-	-	-	保留
	7	R/W	0	ie_mat1b	mat1b 的中断启动
	6	R/W	0	ie_mat1a	mat1a 的中断启动
	5	R/W	0	ie_mat0b	mat0b 的中断启动
	4	R/W	0	ie_mat0a	mat0a 的中断启动
	3	R/W	0	ie_cap1f	cap1f 的中断启动
	2	R/W	0	ie_cap1r	cap1r 的中断启动
	1	R/W	0	ie_cap0f	cap0f 的中断启动
0	R/W	0	ie_cap0r	cap0r 的中断启动	
<b>TIMER_RF: DMA 要求旗标</b>					
48h	15:8	-	-	-	保留
	7	R/W1c	0	rf_mat1b	mat1b 的 DMA 要求旗标 (note-7) (note-8)
	6	R/W1c	0	rf_mat1a	mat1a 的 DMA 要求旗标
	5	R/W1c	0	rf_mat0b	mat0b 的 DMA 要求旗标
	4	R/W1c	0	rf_mat0a	mat0a 的 DMA 要求旗标
	3	R/W1c	0	rf_cap1f	cap1f 的 DMA 要求旗标
	2	R/W1c	0	rf_cap1r	cap1r 的 DMA 要求旗标
	1	R/W1c	0	rf_cap0f	cap0f 的 DMA 要求旗标
0	R/W1c	0	rf_cap0r	cap0r 的 DMA 要求旗标	



Index	Bit	R/W	Default	Name	Description
<b>TIMER_RE: DMA 要求启动</b>					
4ch	15:8	-	-	-	保留
	7	R/W	0	re_mat1b	要求 DMA 启动于 mat1b
	6	R/W	0	re_mat1a	要求 DMA 启动于 mat1a
	5	R/W	0	re_mat0b	要求 DMA 启动于 mat0b
	4	R/W	0	re_mat0a	要求 DMA 启动于 mat0a
	3	R/W	0	re_cap1f	要求 DMA 启动于 cap1f
	2	R/W	0	re_cap1r	要求 DMA 启动于 cap1r
	1	R/W	0	re_cap0f	要求 DMA 启动于 cap0f
	0	R/W	0	re_cap0r	要求 DMA 启动于 cap0r

R/W1C: 读取 与 写“1”清除

Note-3: 重置计数器的优先级高于保留计数器

Note-4: 如果 tctl\_en 或 tctl\_st 处于关闭, 则函数无效

Note-5: 如果两个事件同时发生, mctl\_matxa 的优先级高于 mctl\_matxb

Note-6:

all if\_XXX/of\_XXX need to write '1' to clear

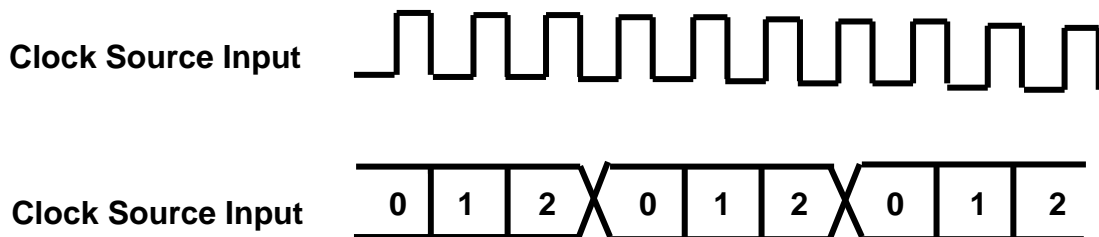
all if\_XXX/of\_XXX are cleared if tctl\_en=0

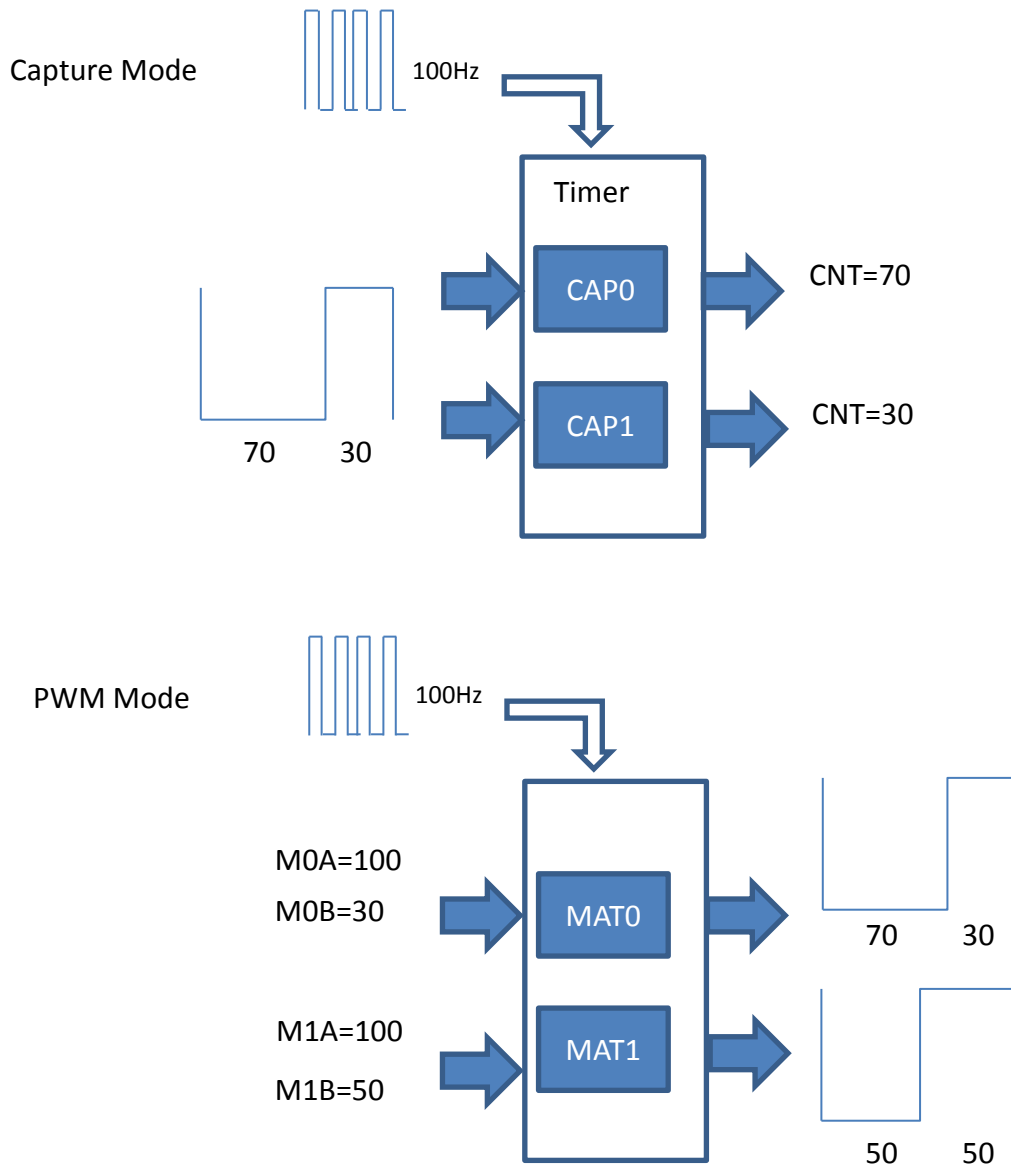
Note-7:

all rf\_XXX need to write '1' to clear

all rf\_XXX are cleared if tctl\_en=0

Note-8: reg\_rf 可用作关闭 reg\_re = ffh 和 DMA 通道的事件指示器





## 19.4 功能描述

### 19.4.1 多个捕获和匹配引脚

系统可以从多个引脚中选择用于 GPIO 缓存器中的捕获或匹配(Match)功能。当为匹配输出选择多个引脚时，所有此类引脚均由定时器的 MATOx 单独驱动。当为捕获输入选择多个引脚时，不同的引脚将由具有相应捕获设置的定时器内部逻辑进行过滤和处理。请注意，匹配条件可以在不使用设备引脚的情况下在内部使用。

### 19.4.2 中断 (Interrupts)

定时器中断源包括 4 个匹配中断的位和 4 个捕获中断的位。如果产生捕获/匹配缓存器中定义的捕获/匹配事件，并在 IE\_REG 中设置相应的中断启动缓存器，则 IF\_REG 中的相应位将很高。否则，位将低。将逻辑写入相应的 IF\_REG 位将复位中断。写入零不起作用。IF\_REG 中仅发出中断旗标处于非零状态。此外，IF\_REG 包含捕获事件和主计数器的溢出旗标。如果捕获事件发生两次或两次以上，而不按系统清除旗标，则意味着捕获的数据将被覆盖，必须放弃，并设置 OF\_CAPxx 旗标。在正常定时器操作中，主计数器不应从 0xff 转到 0x0000，否则可能会导致非法捕获数据或匹配输出，并设置 OF\_TCNT 旗标。系统应在设置溢出旗标 OF\_CAPxx 或 OF\_TCNT 旗标时，可利用分辨率调优或修复事件处理来逐步处理定时器错误。

### 19.4.3 DMA

DMA 函数包含 DMA 请求启动缓存器和 DMA 请求旗标缓存器，用于相应的定时器捕获或匹配事件。设置相应的 RF\_REG，并在设置相应的 DMA 请求并发生相应的定时器事件时，向 DMA 电路发出启动 DMA 操作的定时器 DMA 操作。将逻辑写入相应的 RF\_REG 位将清除 RF\_xxx 旗标并终止 DMA 操作。写入零不起作用。在 DMA 操作正常完成后，RF\_REG 也会被清除。请注意，单个定时器仅应用单个 DMA 通道，因此对于普通 DMA 应用而言，RE\_REG 的单位启动是合理的。

### 19.4.4 计数控制 (Count Control)

计数控制包括 TCTL、TCNT、PSCL 和 PCNT。缓存器 TCTL\_EN 控制整个定时器功能，IF\_xxx 和 RF\_xxx 全部清除，如果清除 TCTL\_ST 设置，则保留 [TCNT、PCNT] 丢弃 TCTL\_ST 设置。缓存器 TCTL\_MODE 控制运行 [TCNT、PCNT] 的定时器，此位也由 H/W 产生的捕获/匹配事件控制。缓存器 TCTL\_MODE 用于在定时器和计数器模式之间进行选择，并在计数器模式下选择引脚和边进行计数。当计数器模式选择为操作模式时，计数器输入 (由 TCTL\_SEL 选择)在 PCLK 时钟的每个上升边缘上采样。比较此 CAP 输入的两个连续样本后，可以识别以下四个事件之一：上升边缘、下降边缘、边之一或所选计数器输入的级别没有变化。仅当标识的事件发生且事件对应于 TCTL\_MODE 缓存器选择的事件时，[TCNT、PCNT] 缓存器才会递增。

TCNT 缓存器使用预缩放 PSCL 缓存器，如果 PCNT 等于 PSCL，TCNT 将递增，在定时器模式下每个 PCLK 的 PCNT 增量和在计数器模式下的每个计数勾选时，PCNT 缓存器将复位，而 PCNT 等于 PSCL 设置。实际捕获/匹配功能与 [TCNT、PCNT]、控件 [TCNT、PCNT] 配合使用，以进行保存、复位或计数。所有操作仅依赖于具有完整 PCNT 循环的 TCNT 值。

#### 19.4.5 捕捉控制 (Capture Control)

每个捕获寄存器都与设备引脚上的正边缘信号有相关联，当该引脚上发生指定事件时，可能会加载定时器计数器值。捕获启动寄存器 (CCTL\_CAPxx) 中的设置确定是否启动了捕获功能，以及捕获事件是发生在关联引脚的上升沿、下降沿还是两条边上。

捕获控制寄存器 (CCTL) 用于控制四个捕获(capture)寄存器 (CAP0R、CAP0F、CAP1R + CAP1F，并与 match 函数共享)，当捕获事件发生时 CCTL 可决定使用哪个捕获寄存器来加载定时器计数器中的数值，以及是否应用具有 2/4/8 时钟的数字滤波器来过滤来自外部设备的噪音。同时设置上升位和下降位是否为有效的规划，因此两种边缘都可以出现捕获事件。输入具有上升与下降的所有边缘都可以同时检测到并捕获。

捕获输入还可以通过信号电平或信号边缘来保持/复位计数器，如同寄存器 CICTL\_CxACTX 和 CICTL\_CxACT 的描述，匹配输入 (match input，来自定时器之前的输出)具有与 CICTL\_MxACT 描述类似的但更简单的功能。

如果启动了 CCTL\_CAPxx，则产生中断旗标，设置相应的 IE\_CAPxx 并发生捕获事件。

#### 19.4.6 配对控制 (Match Control)

匹配寄存器值 (包括 MAT0A、MAT0B、MAT1A 和 MAT1B)与定时器计数器值连续进行比较。当两个值相等时，可以自动触发操作。操作的可能性是产生中断、复位定时器计数器或停止定时器。操作由 MOCTL\_Mxxx 寄存器中的设置控制。

MCTL\_MATxx 寄存器用于控制当一个匹配寄存器与定时器计数器匹配时对定时器 MATOx 输出执行的操作。可以执行以下操作之一，包括清除 MATOx、设置 MATOx 或切换 MATOx。捕获输入可用于在捕获输入较高时，使用 MCTL\_MxMK 寄存器清除 MATOx 的中间阶段。

如果 TCNT 等于匹配寄存器 MATxx，并设置相应的 IE\_MATxx，则产生中断旗标。

## 20 ADC

### 20.1 主要概述

- 操作电压
  - VDA: 1.8V~3.6V @ VDA 参考电压
  - VDA: 2.3V~3.6V @ VBG:1.0V 参考电压
  - 最佳的 ENOB(>10bit) @VDA:3.3V & VDA 3.3V 参考电压
- 共 16 个信道的 12 位 ADC 控制接口
- 支援 DMA
- Conversion mode
  - when ADC clock  $\leq$  APB clock
    - ◆ 序列的单个转换
    - ◆ 序列的连续转换
  - when ADC clock  $>$  APB clock
    - ◆ 单次转换一个通道
    - ◆ 连续转换一个通道
- 模拟看门狗

### 20.2 方块图

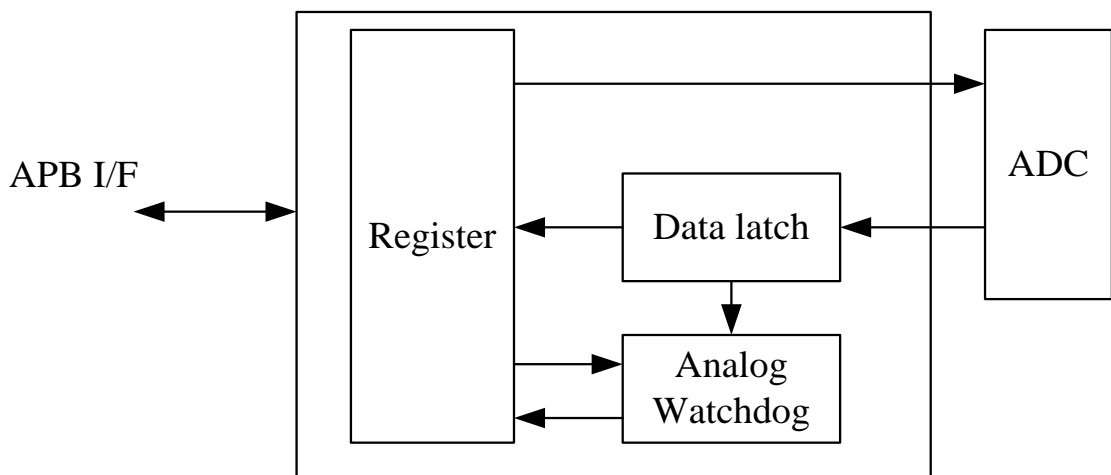


图 57 ADC 方块图

**20.3 缓存器描述**

表格 46 ADC 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>ADC_CFG1: ADC configuration</b>					
00h	31:13	-	-	-	保留
	12:11	R/W	10	ADC_REF_SEL	ADC 参考电压源选择 00: VDDA 01: V1P0 10: External Vref 11: External Vref *当 ADC_REF_SEL 设置为 2'b01 (VREF 选择 V1P0)时, IC 工作电流将增加 150μA, 因此在省电模式时, 请将 ADC_REF_SEL 设置为 2'b00。
	10	R/W	0	ADC_SLOW	0: 关闭, 当 ADC 时钟 = APB 时钟时。 1: 启动, 当 ADC 时钟 > APB 时钟
	9	R/W	0	ADC_EOCIE	0: 关闭 ADC EOC 中断。 1: 启动 ADC EOC 中断。
	8	R/W	0	ADC_WDIE	0: 关闭 ADC 看门狗中断。 1: 启动 ADC 看门狗中断。
	7	R/W	0	ADC_DMA	直接内存访问模式。 0: 关闭 DMA 模式。 1: 启动 DMA 模式。
	6	R/W	0	ADC_AWD	1: 启动 ADC 监视器。 0: 关闭 ADC 监视器。
	5	R/W	0	ADC_CONT	序列的单/连续转换。 0: 序列的单个转换。 1: 序列的连续转换。
	4:2	R/W	0	ADC_SLT_CLK	Select ADC clock: 000: 16MHz 001: 16MHz /2 010: 16MHz /4 011: 16MHz /8 100: 16MHz /16 101: 16MHz /32 110: 16MHz /64 111: 16MHz /128 注意: 软件只有在 ADC PD 关闭或向 ADC_STOP 写入"1"时才能对缓存器字段进行程序设定
	1:0	R/W	0	ADC_DO	AD data output format: (0,0): 12-bit (0,1): 10-bit (1,x): 8-bit
<b>ADC_DO_IDX_F: ADC data &amp; index 致能标旗</b>					
04h	31:16	R	0	ADC_DINDEX	ADC Data channel index 1: 致能 0: 禁能 bit16: ADC channel 0 .... Bit31: ADC channel 15
	11:0	R	0	ADC_DATA	ADC convert data

Index	Bit	R/W	Default	Name	Description
<b>ADC_FLAG: AWD and EOC 旗标</b>					
08h	31:2	-	-	-	保留
	1	R/W c	0	AWD	模拟看门狗 dog flag. 它通过编写 1 到它的软件清除。
	0	R/W c	0	EOC	转换旗目标结尾。 它通过编写 1 的软件或读取 ADC_DATA 缓存器来清除。
<b>ADC_ADCCR: ADC 控制位</b>					
0Ch	31:18	-	-	-	保留
	17	R/W	0	ADC_START	ADC 控制启动。 注意: 仅当 ADC PD 关闭时, 软件才允许设置 ADC_START
	16	R/W	0	ADC_STOP	ADC control stop, write "1" to stop
	15:0	R/W	0	ADC_CH	AD 通道选择。 示例: 0000_0000_0000_0001: 通道 0 启动。 0000_0000_0000_0010: 通道 1 启动。 0000_0000_0000_1111: 通道 0~3 启动。 注意: 当 ADC_SLOW = 1 时, ADC_CH 仅设置一个通道, 而 ADC_CH 仅在将"1"写入 ADC_STOP 时才更改其它通道
<b>ADC_AWDCH: AWD 通道设置</b>					
10h	31:5	-	-	-	保留
	4	R/W	0	AWDSGL	在单个信道或所有信道上启动看门狗。 0: 在所有信道上启动模拟看门狗。 1: 单通道启动模拟看门狗。
	3:0	R/W	0	AWD_CH	模拟看门狗通道选择 0000: 通道 0 0001: 通道 1 0010: 通道 2 .... 1111: 通道 15
<b>ADC_THRHD: AWD 阈值设定</b>					
14h	31:28	-	-	-	保留
	27:16	R/W	0	AWD_HT	模拟看门狗更高的阈值
	15:12	-	-	-	保留
	11:0	R/W	0	AWD_LT	模拟看门狗阈值较低
<b>ADC_CFG2: ADC standby and VCM 设定</b>					
18h	31:3	-	-	-	保留
	2	R/W	0	ADC_STANDBY	ADC 待机模式。 当 ADC 以 240kHz 时钟速率运行时, 待机模式下停止运行的电流耗散将从 460μA 下降到 75μA
	1:0	R/W	10	ADC_VCM_LS	当 VDDA=1.8V、VDDA/2=0.9V=NMOS 阈值电压, 并让前置放大器输入 NMOS 关闭时, 因此 VCM 需要更多的+0.18V。 00: VCM=VDDA*(10/20) 01: VCM=VDDA*(11/20) 10: VCM=VDDA*(12/20)

Index	Bit	R/W	Default	Name	Description
					11: VCM=VDDA*(13/20)

Note: R/W1C: 读取与写“1”清除

## 20.4 功能描述

### 20.4.1 转换功能

#### 20.4.1.1 ADC\_SLOW=0, 当 ADC clock ≤ APB clock

在单转换模式，当 ADC\_CONT=0 & ADC\_SLOW=0 时，选择此模式使 ADC 执行单个转换序列，一次性转换当前所有通道。

在连续转换模式，当 ADC\_CONT=1 & ADC\_SLOW=0 时，选择此模式使 ADC 执行一系列转换，一次性转换当前所有通道，然后自动重新启动并持续执行相同的转换序列。

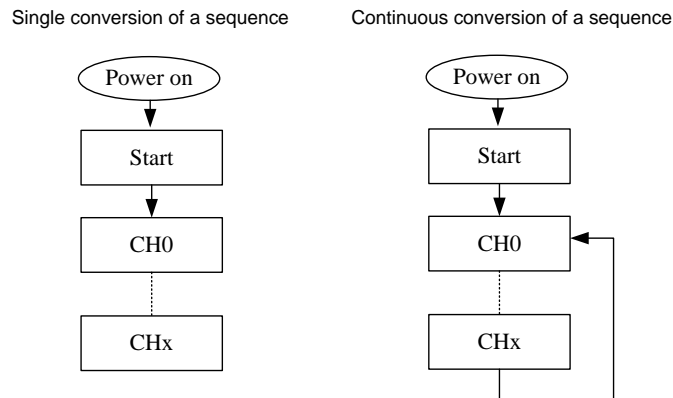


图 58 ADC\_SLOW=0 转换顺序

#### 20.4.1.2 ADC\_SLOW=1, 当 ADC clock > APB clock

在单转换模式，当 ADC\_CONT=0 & ADC\_SLOW=1 时，选择此模式使 ADC 执行单个转换。

在连续转换模式，当 ADC\_CONT=1 & ADC\_SLOW=1 时选择此模式使 ADC 执行一系列转换，一次性转换一个通道，然后自动重新启动并持续执行相同的转换序列。

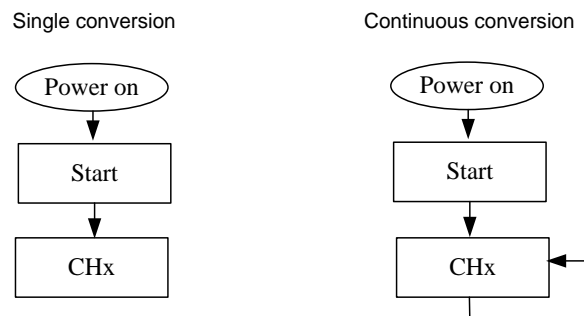


图 59 ADC\_SLOW\_1 的转换顺序



### 20.4.2 转换时序

$$t_{CONV} = t_{SMPL} + t_{SAR} = 2 * ADCCLK + 12 * ADCCLK = 14 * ADCCLK$$

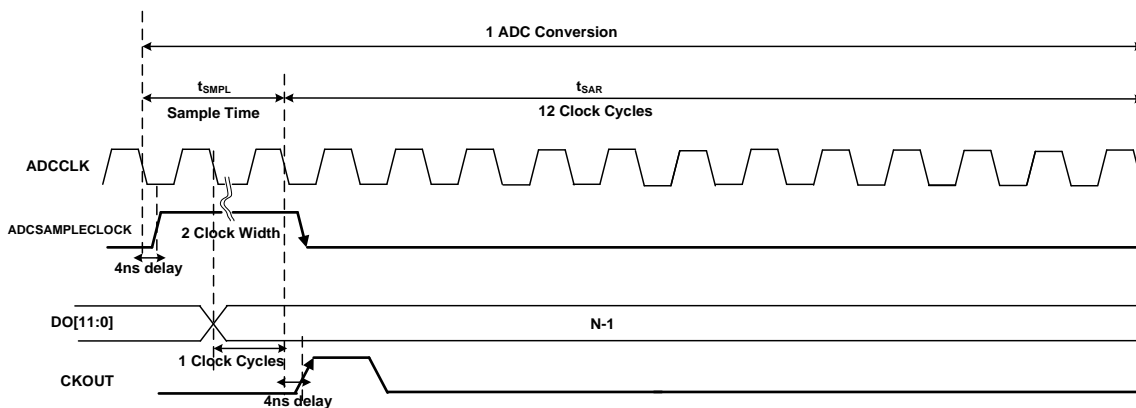


图 60 ADC 转换时间

### Example timing diagrams

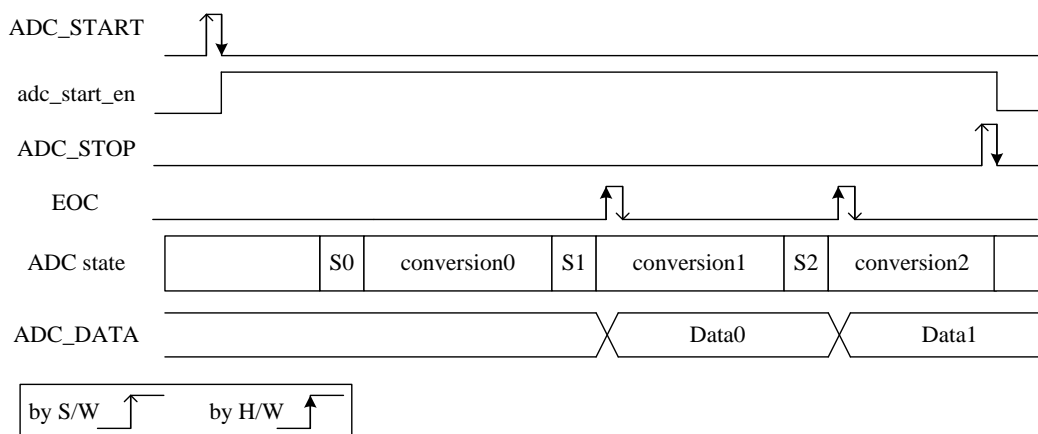


图 61 ADC\_SLOW=0 时序图

### 20.4.3 模拟看门狗的窗口

如果 ADC 转换的模拟电压低于或高于较高阈值，则设置 ADC\_WD 模拟观察器状态位。这些阈值在 HT[11:0] 和 LT[11:0]位中程序设定。可以通过设置 ADC\_WDIE 位来启动中断。

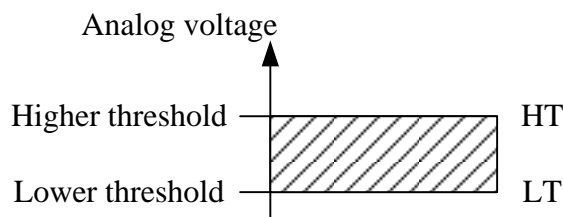


图 62 模拟看门狗窗口

## 21 DAC

### 21.1 主要概述

- 12-bit DAC
- DAC Rail-to-Rail Amplifier output

### 21.2 方块图

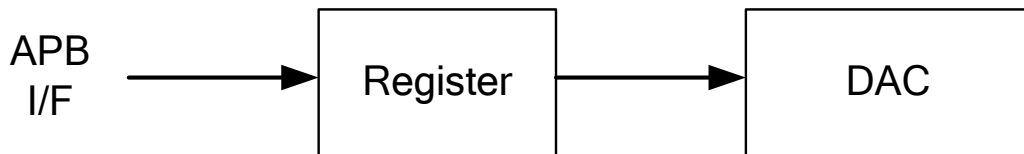


图 63 DAC 方块图

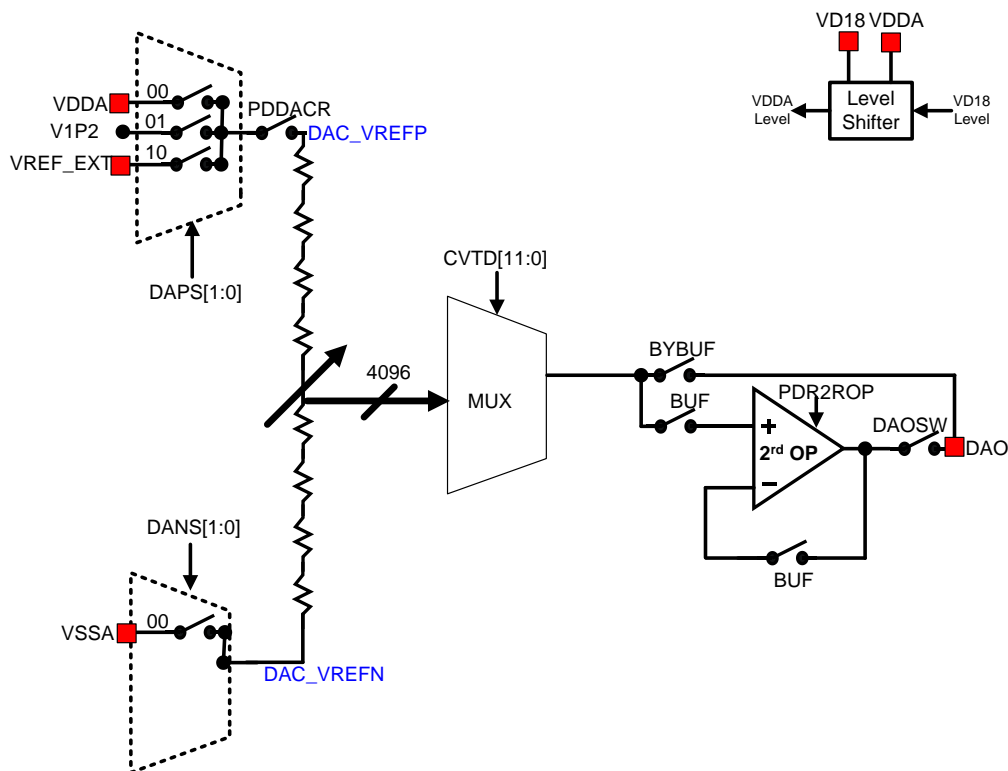


图 64 DAC 模拟方块图

**21.3 缓存器描述**

表格 47 DAC 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>DAC_CFG: DAC 设置</b>					
00h	31:7	-	-	-	保留
	6	R/W	0	DAC_BYBUF	DAC 输出旁路轨到轨放大器 (无驱动能力)
	5	R/W	0	DAC_BUF	DAC 输出 + 轨到轨放大器 (可驱动高达 1500pF 的输出电容)。*DAOSW 需要写入 1。
	4	R/W	0	DAOSW	轨到轨(Rail-to-Rail)放大器输出到 PAD
	3:2	R/W	0	DAPS	DAC 正输入选择 00: VDDA 01: V1P0V 10: VREF_EXT 11: 保留 *当 DAPS 设置为 2'b01 (VREF 选择 V1P0)时, IC 工作电流将增加 150μA, 因此在省电模式时, 请将 DAPS 设置为 2'b00。
1:0	R/W	0	DANS	DAC 负输入选择。 *必须设置 00。 选择 VSSA。	
<b>DAC_CVTD: DAC 12-bit 数据输入</b>					
04h	31:12	-	-	-	保留
	11:0	R/W	0	CVTD	DAC 12-bit 数据输入

## 22 超低功耗比较器

### 22.1 主要概述

WT32L064/032 内置两个电压比较器，其功能如下。

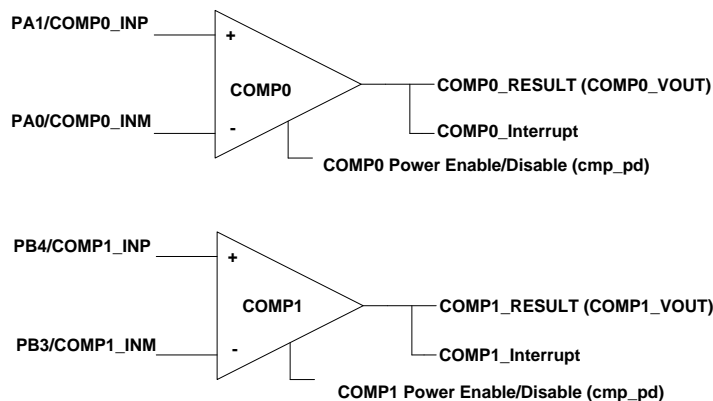
- 超低功耗（工作电流低于 10μA/单位）
- 比较器可以单独启动或关闭
- 没有比较器输出引脚，但可以通过缓存器读取输出状态
- 每个比较器输出更改可以产生中断

Range	Index	Function	Description
AHB	0x5001_F000~0x5001_FFFF	system control	

Index	Bit	R/W	Default	Name	Description
<b>SYSCON_CMP_OUT: 比较器输出</b>					
14h	1	R	-	CMP1_VOUT	Comparator 1 output
	0	R	-	CMP0_VOUT	Comparator 0 output

Range	Index	Function	Description
APB0	0x4001_A000~0x4001_AFFF	PMU	电源管理单位

Index	Bit	R/W	Default	Name	Description
<b>PMU_VD_CR: 电压侦测控制讯号</b>					
0Ch	3:2	R/W	11	cmp_hs	选择 CMP 的操作模式 高速模式: 1 低速模式: 0 cmp_hs[1]: CMP1 cmp_hs[0]: CMP0
<b>PMU_CLK_PD: Clock power 讯号</b>					
14h	8:7	R/W	11	cmp_pd	CMP0 & CMP1 省电开关 [7]: CMP0, [8]:CMP1 0: 开启。 1: 关闭。



## 23 CRC32

### 23.1 主要概述

循环冗余检查 (CRC) 是数字网络和储存装置中常用的错误侦测码，用于检测原始数据的意外变动。在 EN/IEC 60335-1 标准中，提供了一种验证闪存(Flash Memory)完整性的方法，CRC 计算单元可于运行期间计算软件的签名，并与链接时所生成且存储在给定的存储器位置的参考签名进行比较。

- CRC-32 polynomial (0x04C11DB7)  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- 单一缓存器 (同时支持输入与输出)
- CRC 运算时间为 4 个系统时钟的时间
- 支持字组(word)、半字组(half-word)、以及字节(byte)之数据大小(AHB-Lite I/F)
- 支持大端序(big endian)与小端序(little endian)之位阻排序方法

### 23.2 缓存器描述

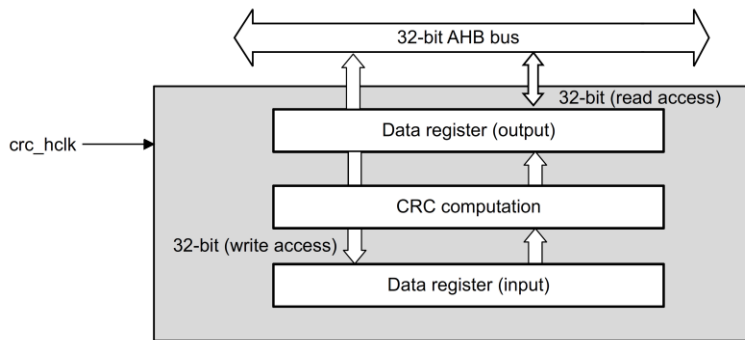
表格 48 CRC32 控制缓存器

Index	Bit	R/W	Default	Name	Description
<b>CRC32_DR: CRC32 数据缓存器</b>					
00h	31:0	R/W	0	DR	将新数据写入此缓存器以进行 CRC 计算。进行读取可取得上一个 CRC 计算结果时保留该结果。
<b>CRC32_RST: CRC32 reset bit</b>					
04h	31:2	-	-	-	保留
	1	R/W	0	LE	小端序 (Little endian) 致能: 0: 大端序 (big endian) 1: 小端序 (little endian)
	0	W	-	RST	写入 1 以复位 CRC 计算单元并将数据缓存器设置为 0x00。 完成后由硬件自动清除。

### 23.3 功能描述

CRC 计算具有单一 32 位数据缓存器，用作输入缓存器，用于在 CRC 计算器中输入新数据 (写入缓存器时)，并保存上一次 CRC 计算的结果 (读取缓存器时)。请注意，新的写入操作会暂停，直到上笔 CRC 计算结束。

每次将新数据写入数据缓存器中，皆会以先前的 CRC 值和新的数据来产生新的 CRC 值，MCU 可透过将 1 写入 RST 缓存器来复位 CRC 计算单元(数据缓存器会被重置为 0x00)。



### 23.4 CRC32 示例代码:使用 WT32L064/032

```
#define BUFFER_SIZE    64
static const uint32_t DataBuffer[BUFFER_SIZE] =
{
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //0
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //1
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //2
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //3
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //4
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //5
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //6
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //7
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //8
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //9
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //10
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //11
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //12
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //13
    0x00000001, 0x00000002, 0x00000003, 0x00000004, //14
    0x00000001, 0x00000002, 0x00000003, 0x00000004 //15
};
```

```
__IO uint32_t CRCValue = 0;
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_CRC32, ENABLE);
CRC_ResetDR();
CRCValue = CRC_CalcBlockCRC((uint32_t *)DataBuffer, 64);
printf("64 words, crc=0x%08x\r\n",CRCValue);
```

输出:

**64 words, crc=0x13388b4a**

### CRC 32 example code on PC

*/\* The result should be in accordance with ISO 3309, ITU – T V.42, Gzip and PNG.*

*This code is a translation from Ruby, with an adjustment to use 32-bit integers.*

*This code happens to resemble the examples from RFC 1952 section 8 and from PNG annex D, because those examples use an identical table. \*/*

```
uint32_t crc_table[256];
void CrcTable(void)
{
    uint32_t rem;
```

```

for (- 197 -nti = 0; i < 256; i++) {
    rem = i; /* remainder from polynomial division */
    for (int j = 0; j < 8; j++) {
        if (rem & 1) {
            rem >>= 1;
            rem ^= 0xedb88320;
        }
        else
            rem >>= 1;
    }
    crc_table[i] = rem;
}
}

uint32_t rc_crc32(uint32_t crc, unsigned char *buf, size_t len)
{
    uint8_t raw_byte;
    unsigned char *start_p, *end_p;
    uint32_t rotate_H8;
    uint32_t nibble_H8;
    uint32_t rem_result;

    crc = ~crc; //invert first CRC input
    end_p = buf + len;
    for (start_p = buf; start_p < end_p; start_p++)
    {
        raw_byte = *start_p; /* Cast to unsigned octet(8). */

        nibble_H8= (crc & 0xff) ^ raw_byte;
        rem_result = crc_table[nibble_H8];
        rotate_H8 = (crc >> 8);
        crc = rotate_H8 ^ rem_result;
    }

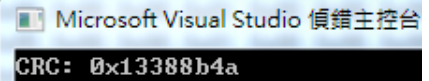
    return ~crc; //invert first CRC output
}

int main()
{
    uint32_t crc_result;
    unsigned char test_data[] =
    { //0 //1 //2
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //0
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //1
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //2
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //3
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //4
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //5
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //6
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //7
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //8
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //9
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //10
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //11
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //12
        0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //13
    }
}

```

```
    0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04, //14  
    0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x02,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x04 //15  
};  
  
CrcTable();  
crc_result = rc_crc32(0, test_data, sizeof(test_data));  
printf("CRC: 0x%08x\n", crc_result);  
return 0;  
}
```

输出:



Microsoft Visual Studio 偵錯主控台  
CRC: 0x13388b4a



## 24 Boot ROM & IAP

### 24.1 说明

WT32L064/032内建8K字节之Boot ROM，用来储存Boot Code之用，Boot Code存有IAP(In Application Programming)的程序，用来做Flash 区内User Code的远程更新用途。

### 24.2 进入 Boot Code IAP 的方式

通常当CPU复位(Reset)，会先执行此Boot Code，随后才跳至闪存 (Flash) 开始执行用户的程序。

此Boot code可以用USB/UART的应用程序来执行IAP，有以下几种方式来进入Boot code 的IAP功能。

1. 硬体会先判断Flash单元的控制缓存器nboot(0x40080028)，确认是否从Boot ROM开始执行程序(默认)? 或Flash开始执行程序? 或SRAM开始执行程序?
  
9. 检查PMU单元控制缓存器boot\_sel(0x4001A024)的Boot select是否enable?

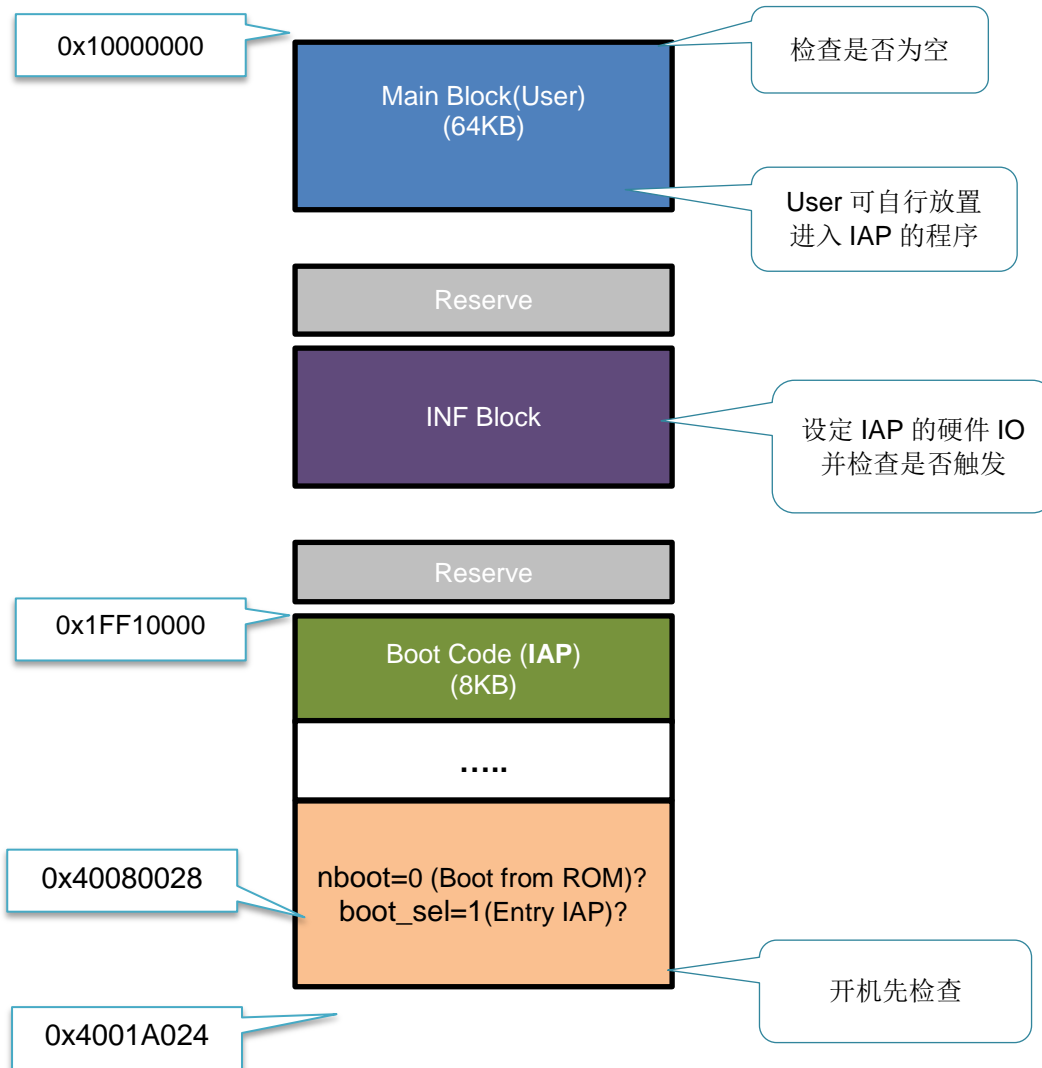
这是由User Code自行启动，再执行软件复位，复位后就会进入boot code，所以User Code事先包含有一个USB/UART的应用程序，在此程序收USB/UART到特殊指令时，软件会将PMU 单元boot\_sel 状态启动(enable)，接着做软件复位(Software Reset)，进入Boot Code，boot code内的IAP会检查boot\_sel 状态是否启动，若是则进入IAP进行User Code 远程升级并清除boot\_sel状态，若boot\_sel状态没有被启动，则离开Boot Code回到User Code。

10. 另一种进入IAP方式是在Information Block存有使用者定义的IAP enable的GPIO，可以定义任一GPIO当作boot pin，但在出厂时，就要事先刻录到 Information Block。  
例如，当定义GPIOA1 LOW时为IAP触发，在上电后即可直接进入IAP，若GPIOA1为HIGH则离开IAP回到User Code。
  
11. 检查MCU寻址的 SP (stack pointer)及PC (Program Counter)的值是否在合理范围内?  
若在范围内，代表是flash存在有效的程序，若值不在范围内，代表flash是清空的，也就是flash没有任何程序，此时也能进入IAP。

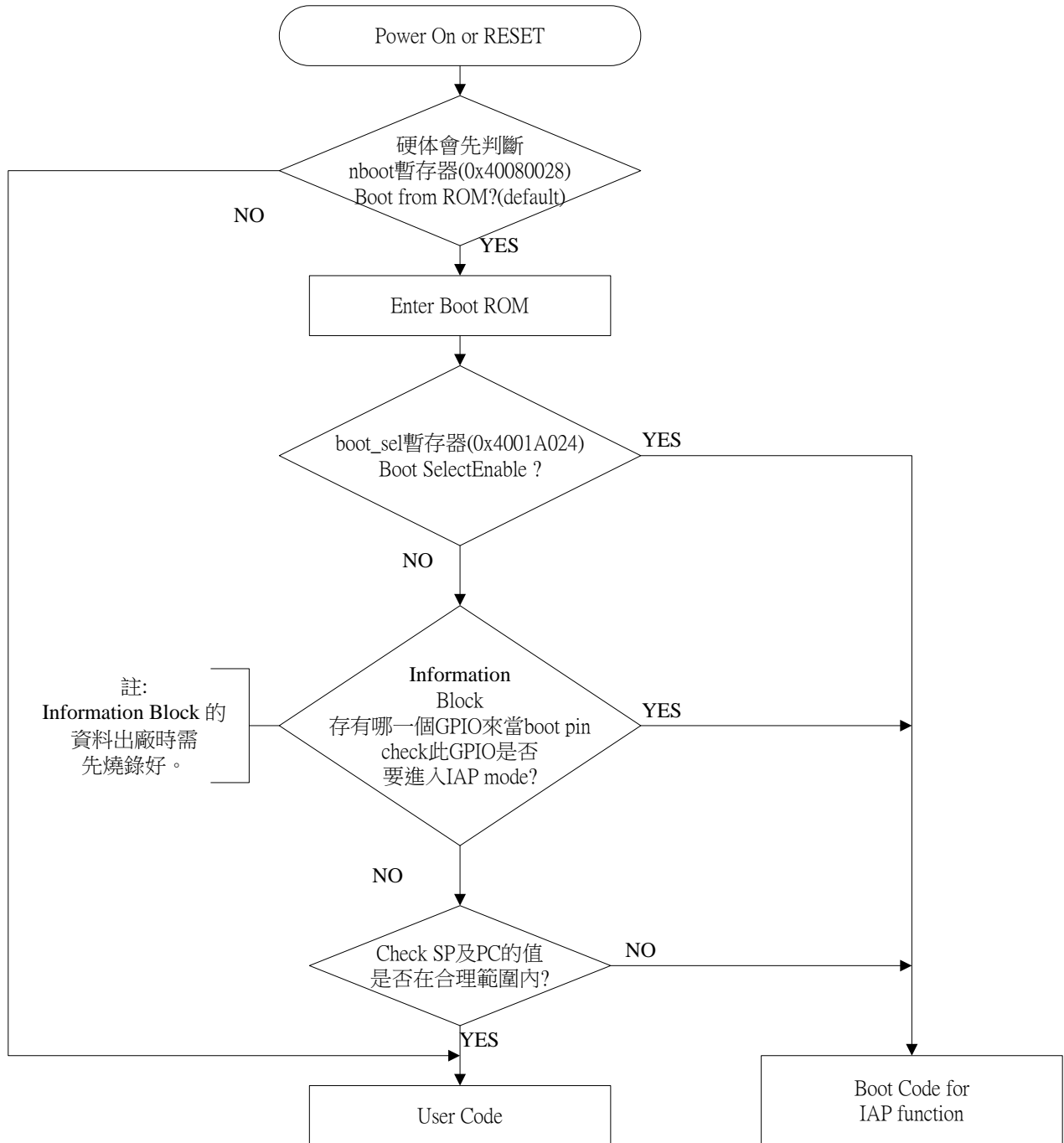
### 24.3 Boot 与 IAP 内存映像

下列描述为进入 IAP 的条件与相对位置及示意图

Index	Function	Description
0x1000_0000~0x1000_FFFF	embedded flash	64KB
0x1FF0_0000~0x1FF0_07FF	INF Block	2KB
0x1FF1_0000~0x1FF1_1FFF	boot ROM	8KB
0x4000_0000~0x4003_FFFF	APB0 memory space	256KB
0x4008_0000~0x400B_FFFF	AHB memory space	256KB



24.4 流程图



## 25 电气特性

### 25.1 极限参数

Parameter	Symbol	Condition	Range	Units
D.C. Supply Voltage	$V_{D33}$		-0.3 ~ 3.6	V
Input Voltage	$V_I$		-0.3 to $V_{D33} + 0.3$	V
Output Voltage	$V_O$		-0.3 to $V_{D33} + 0.3$	V
Total current source by all GPIO	$\Sigma I_{OH}$	125mA @3mA(6mA)/single pad	@-40°C ~ +85°C	mA
	$\Sigma I_{OH}$	85mA @3mA(6mA)/single pad	@-40°C ~ +105°C	mA
Total current sink by all GPIO	$I_{OL}$	125mA @3mA(6mA)/single pad	@-40°C ~ +85°C	mA
	$\Sigma I_{OH}$	85mA @3mA(6mA)/single pad	@-40°C ~ +105°C	mA
Ambient Temperature	$T_A$		-40 ~ 105	°C
Storage Temperature	$T_{STG}$		-60 ~ 125	°C

**Note: Stresses above those listed may cause permanent damage to the devices.**

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

Parameter	Symbol	Conditions	Min.	Max.	Units
Internal AHB clock frequency	fHCLK		0	32	MHz
Internal APB0 clock frequency	fPCLK1		0	32	MHz
Internal APB1 clock frequency	fPCLK2		0	32	MHz
Standard operating voltage	VD33	BOR detector disabled	1.65	3.6	V
		BOR detector enabled, at power-on	1.8	3.6	V
		BOR detector disabled, after power-on	1.65	3.6	V
Analog operating voltage (all features)	VDDA	Must be the same voltage as VD33	1.65	3.6	V
Digital power from Internal LDO1.2V~1.8V	VDD	need to connect capacitor 1μF	1.2	1.8	V
Standard operating voltage, USB domain	VD33 (USB)	USB peripheral used	3	3.6	V
		USB peripheral not used	1.65	3.6	V
VBUS: 5V input, USB domain	VDD_5V	USB peripheral used	4.5	5.5	V
Input voltage on 5V Tolerant I/O & reset pins	VIN		-0.3	5.5	V
Input voltage on 3.3V I/O pin			-0.3	VD33+0.3	V

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

**25.2 DC 电气特性**
**25.2.1 电源消耗**
**VD33=ADDA=3.0V, Vcore=1.8V**

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Run mode	IVDD32M-1	sysclk= (HIS 16M x2) =32M (PLL on) AHB=sysclk no load & peripheral off		8.9		mA
Run mode	IVDD32M-2	sysclk= (HIS 16M x2) =32M (PLL on) AHB=sysclk/2 no load & peripheral off		5.5		mA
Run mode	IVDD16M-1	sysclk= HSI=16M AHB=sysclk no load & peripheral off		5.2		mA
Run mode	IVDD16M-2	sysclk= HSI=16M AHB=sysclk/2 no load & peripheral off		3		mA
Low Power Run mode	IVDD1M	sysclk= MSI=1M AHB=sysclk/2 no load & peripheral off		202		μA
Low Power Run mode	IVDD131K	sysclk= MSI=131K AHB=sysclk/2 no load & peripheral off		57		μA
Sleep mode	Isleep16M	sysclk= HSI=16M AHB=sysclk/2 no load & peripheral off		1.1		mA
Sleep mode	Isleep1M	sysclk= MSI=1M AHB=sysclk/2 no load & peripheral off		113		μA
Stop mode	ISTOP1	SRAM(retain) , All clock (off) Flash (off) no load & peripheral off		0.6		μA
Stop mode+ RTC	ISTOP2	SRAM(retain), MSI, PLL, HSI, HSE (off) LSI, LSE (on) Flash (off) no load & peripheral off		1.3		μA
Standby mode	ISTBY1	SRAM(lost) All clock (off) Flash (off) Vcoer (off) no load & peripheral off		0.3		μA
Standby mode+ RTC	ISTBY2	SRAM(lost) MSI, PLL, HSI, HSE (off) LSI, LSE (on)		1.0		μA

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
		Flash (off) Vcoer (off) no load & peripheral off				

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

### 25.2.2 Digital I/O 电气特性

VDDA=3.3V

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Schmitt Trigger Low-to-High threshold Point	$V_{T+}$		0.65*Vdd			V
Schmitt Trigger High-to-Low threshold Point	$V_{T-}$				0.35*Vdd	V
Output High Voltage (DS=0)	$V_{OH}$	$I_{OH} = 3 \text{ mA}$	2.4			V
Output Low Voltage (DS=0)	$V_{OL}$	$I_{OL} = 3 \text{ mA}$			0.4	V
Output High Voltage (DS=1)	$V_{OH}$	$I_{OH} = 6 \text{ mA}$	2.4			V
Output Low Voltage (DS=1)	$V_{OL}$	$I_{OL} = 6 \text{ mA}$			0.4	V
Input Schmitt trigger hysteresis voltage	$V_{HYS}$		300			mV
Input Leakage Current	$I_{OZ}$	$V_O = 0V \text{ or } 3.3V$			$\pm 0.05$	$\mu\text{A}$
Pull-Up Resistor	$R_{PH}$			30		k $\Omega$
Pull-Down Resistor	$R_{PD}$			50		k $\Omega$

(\*): These parameters are presented for design guidance only and not tested or guaranteed

### 25.2.3 LVR 电气特性

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Low V1D35 Reset Voltage	$V_{LVR13\_H}$	VDDA=3.3V		1.40		V
Low V1D35 Reset Voltage	$V_{LVR13\_L}$	VDDA=3.3V		1.35		V

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Low V1D35 Reset Current	$I_{LVR13}$	VDDA=3.3V		< 2		$\mu\text{A}$
Low V1D35 Reset Voltage				$\pm 3$		%

(\*): These parameters are presented for design guidance only and not tested or guaranteed

#### 25.2.4 PLL 电气特性

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
PLL input reference frequency (after pre-divider)	$f_{REF}$		2		24	MHz
PLL VCO range	$f_{VCO}$	VDDA < 1.8V	32		96	MHz
		VDDA $\geq$ 1.8V	32		192	
PLL lock time (Note 1)	$t_{LOC}$	$f_{REF} = 16\text{MHz}$ , $f_{VCO} = 96\text{MHz}$		160		$\mu\text{s}$
PLL operation current	$I_{PLL}$	$f_{REF} = 4.2\text{MHz}$ , $f_{VCO} = 64\text{MHz}$		0.75		mA

(\*): These parameters are presented for design guidance only and not tested or guaranteed

#### 25.2.5 ADC 电气特性

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Resolution				12		bits
Integral Nonlinearity Error (including INL, DNL)		$V_{REF} = 3.3\text{V}$ , VDDA = 3.3V		$\pm 2$		LSB
Input Voltage	$V_{ADCIN}$		VSS		$V_{REF}$	V
ADC Reference Voltage ( $V_{REF}$ )	VDDA	$1.8\text{V} < \text{VDDA} < 3.6\text{V}$		VDDA		V
	V1P0	VDDA > 2.2V & $F_{ADC} \leq 7\text{MHz}$		1.0		V
	VREF_EXT	$1.8\text{V} < \text{VDDA} < 3.6\text{V}$	1.65		VDDA	V
ADC Frequency	$F_{ADC}$			7		MHz
ADC Current	$I_{ADC}$	ADC CLK = 6MHz		400		$\mu\text{A}$
		ADC CLK = 1MHz		314		$\mu\text{A}$
ADC Input impedance	$R_{AIN}$				20	K $\Omega$
Conversion time	$T_{conv}$	$f_{ADC} = 6\text{MHz}$		2.33		$\mu\text{s}$

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

Parameter	Condition	Specification			Units
		Min.	Typ.	Max.	
ENOB	VDDA=Vref=3.6V		10		bit
	VDDA=Vref=3.3V		10		bit
	VDDA=Vref=1.8V		9		bit

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

### 25.2.6 DAC 电气特性

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
Analog supply voltage	VDDA		2.2		3.6	V
VREFP=VDDA Channel	Resolution			10		bits
VREFP=Internal 1.0V Channel				9		bits
VREFP=External Channel				10		bits
Analog supply voltage	VDDA		2.2		3.6	V
VREFP=External Channel of 0.6V				10		bits
Current consumption on VDDA Supply, VDDA = 3.3 V @0X0800	IDAC			86.2		μA
Current consumption on VDDA Supply, VDDA = 3.3 V @0X0F1C	IDAC			118.7		μA
Differential non linearity	DNL	@VDDA=2.2V~3.6V		±2		LSB
Integral non linearity	INL	@VDDA=2.2V~3.6V		±4		LSB
Resistive Load	RL	DAC Output Buffer ON RL to VSSA	5			KΩ
Capacitive Load	CL	DAC Output Buffer ON CL to VSSA			50	pf

\* If DAC used at VD33 in 1.65V~2.2V,

External Channel must be provided 0.6V at PB0 for 8 bits resolution.

\* It must be back to default channel of VDDA when DAC off to reduce current consumption of 150μA,



**25.2.7 LDO 电气特性**

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
LDO 3.3V Power Current @VDD5V = 5.0V	$I_{VDD5V}$	VD33 $\cong$ 3.0V		30		mA
LDO 3.3V Power Current @VDD5V = 4.5V	$I_{VDD5V45}$	VD33 $\cong$ 3.0V		20		mA
Voltage of LDO 3.3V @Loading=3mA	VD33		3.17	3.3	3.43	V
Voltage of LDO 3.3V Tolerance @Loading=3mA				$\pm 4$		%

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
LDO 1.8V Driven Current @VDD33 = 2.2V	$I_{VDDC22}$	BLDO ON VD18 $\cong$ 1.7V		30		mA
LDO 1.8V Driven Current @VDD33 = 2.2V	$I_{VDDC22S}$	BLDO OFF VD18 $\cong$ 1.7V		285		$\mu$ A
Voltage of LDO 1.8V @Loading=3mA	VD18		1.774	1.81	1.846	V
Voltage of LDO 1.8V Tolerance @Loading=3mA				$\pm 2$		%

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

**25.2.8 HSI 16 MHz**

Parameter	Symbol	Condition	Specification			
			Min	Typ	Max	Units
Frequency	fHSI16			16		MHz
HSI16 trimming step	TRIM			$\pm 0.4$		%
Duty cycle	DuCy		45		55	%
Accuracy of the HSI16 oscillator (factory calibrated)	ACC	VDDA=3V, TA = 25 °C	-1		1	%
		VDDA=3V, TA = 0~70 °C		$\pm 2$		%
		VDDA=3V, TA = -40 °C ~85 °C		$\pm 3$		%
		VDDA=3V, TA = -40 °C ~125 °C		$\pm 4$		%
		VDDA=1.65~3.6V, TA = -40 °C ~125 °C		$\pm 5$		%
HSI16 oscillator startup time	tsu				20	$\mu$ s
HSI16 oscillator power consumption	IDDA			100	140	$\mu$ A

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

**25.2.9 MSI 4.2 MHz**

Parameter	Symbol	Condition	Specification			
			Min	Typ	Max	Units
Frequency	$f_{MSI}$			4.2		MHz
Duty cycle	DuCy		45(49)		55(51)	%
Accuracy of the MSI oscillator (factory calibrated)	ACC	VDDA=3V, $T_A = 25\text{ }^\circ\text{C}$	-1		1	%
MSI oscillator startup time	$t_{su}$				60	$\mu\text{s}$
MSI oscillator power consumption	I <sub>DDA</sub>			15	20	$\mu\text{A}$

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

Parameter	Symbol	Condition	Specification					
			Min	Typ	Max	Units		
Frequency after factory calibration @VDDA=3.3V $T_A=25\text{ }^\circ\text{C}$	$f_{MSI}$	MSI range=0		65.625		kHz		
		MSI range=1		131.25				
		MSI range=2		262.5				
				MSI range=3		525		MHz
				MSI range=4		1.05		
				MSI range=5		2.1		
				MSI range=6		4.2		
MSI oscillator power consumption	I <sub>DDA</sub>	MSI range=0		7		$\mu\text{A}$		
		MSI range=1		7.5				
		MSI range=2		8				
		MSI range=3		9				
		MSI range=4		10				
		MSI range=5		12				
		MSI range=6		15				
MSI oscillator frequency drift $0\text{ }^\circ\text{C} \leq T_A \leq 85\text{ }^\circ\text{C}$	D <sub>TEMP</sub>			$\pm 3.0$		%		
MSI oscillator frequency drift VDDA=3.3V $-40\text{ }^\circ\text{C} < T_A < 110\text{ }^\circ\text{C}$		MSI range=0		$\pm 10$				
		MSI range=1		$\pm 10$				
		MSI range=2		$\pm 9$				
		MSI range=3		$\pm 8$				
		MSI range=4		$\pm 7$				
		MSI range=5		$\pm 6$				
MSI range=6		$\pm 5$						
MSI oscillator frequency drift $1.65\text{V} < \text{VDDA} < 3.6\text{V}$ $T_A=25\text{ }^\circ\text{C}$	D <sub>VOLT</sub>		-2.5		2.5	%		

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

**25.2.10 IRC 48MHz 电气特性**

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
RC Frequency	$F_{RC}$	VDDA = 2.2V~3.6V		48		MHz
Frequency Tolerance	$\Delta F_{RC}/F_{RC}$	Without crystal oscillator calibration		$\pm 4$		%
		With USB calibration		$\pm 0.25$		%
RCOSC Current	$I_{RCOSC}$	12/24/48 MHz		0.25		mA

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

**25.2.11 IRC 37kHz 电气特性**
**VDDA=3.3V**

Parameter	Symbol	Specification			Units
		Min.	Typ.	Max.	
LSI frequency	$f_{LSI}$	32.66		55.19	kHz
LSI oscillator frequency drift $0^{\circ}\text{C} \leq T_A \leq 85^{\circ}\text{C}$	$D_{LSI}$	-3.1	-	+8.1	%
LSI oscillator startup time	$t_{SU(LSI)}$	-	-	235	$\mu\text{s}$
LSI oscillator power consumption	$I_{DD(LSI)}$	-	393	640	nA

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

**VDDA=1.65V**

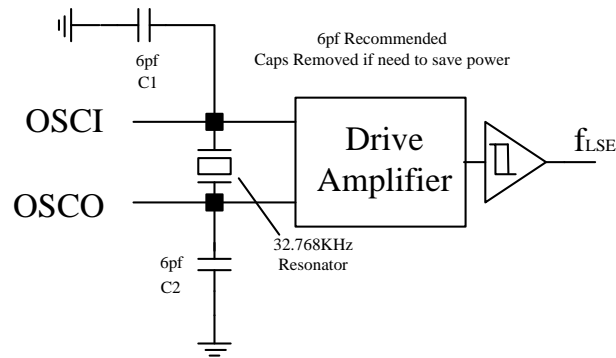
Parameter	Symbol	Specification			Units
		Min.	Typ.	Max.	
LSI frequency	$f_{LSI}$	32.56		55.1	kHz
LSI oscillator frequency drift $0^{\circ}\text{C} \leq T_A \leq 85^{\circ}\text{C}$	$D_{LSI}$	-3.4	-	+8.2	%
LSI oscillator startup time	$t_{SU(LSI)}$	-	-	166	$\mu\text{s}$
LSI oscillator power consumption	$I_{DD(LSI)}$	-	390	592	nA

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

**25.2.12 Crystal 32768 电气特性**

Parameter	Symbol	Condition	Specification			Units
			Min.	Typ.	Max.	
LSE oscillator frequency	$f_{LSE}$	-		32.768		kHz
Startup time	$t_{su(LSE)}$	$V_{DD}$ is stabilized		2		Sec.
Crystal transconductance	Gm			0.3	0.5	$\mu A/V$

(\*): These parameters are presented for design guidance only and not tested or guaranteed.



**25.2.13 COMP 电气特性**

Parameter	Symbol	Conditions	Specification			Units
			Min	Typ	Max	
Analog supply voltage	$V_{DDA}$	-	1.65	-	3.6	V
Comparator input voltage range	$V_{IN}$	-	+0.2	-	$V_{DDA}-0.2$	V
Comparator startup time	$t_{START}$	Fast mode		22	29.8	$\mu S$
		Slow mode		27.6	36.4	
Propagation delay in slow mode	$t_{d\ SLOW}$	$1.65\ V \leq V_{DDA} \leq 2.7\ V$		1.21	2.23	
		$2.7\ V \leq V_{DDA} \leq 3.6\ V$		1.65	2.13	
Propagation delay in fast mode	$t_{d\ fast}$	$1.65\ V \leq V_{DDA} \leq 2.7\ V$		0.43	0.72 $\mu$	
		$2.7\ V \leq V_{DDA} \leq 3.6\ V$		0.55	0.68 $\mu$	
Comparator offset error	$V_{offset}$			$\pm 4$	$\pm 20$	mv
Current consumption	$I_{COMP2}$	Fast mode		3.67	4.67	$\mu A$
		Slow mode		0.95	1.15	

(\*): These parameters are presented for design guidance only and not tested or guaranteed

**25.2.14 BOR/PVD 电气特性**

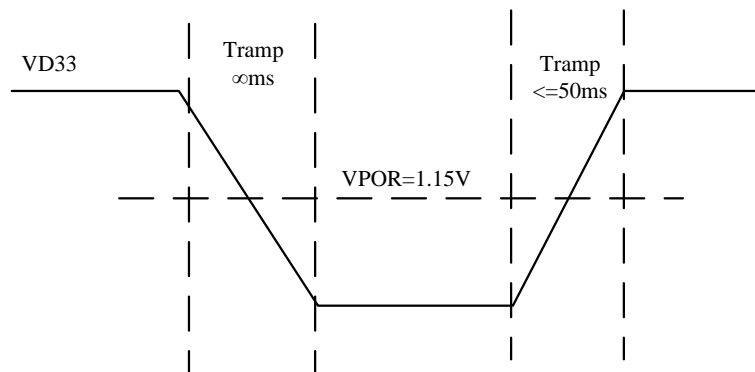
Parameter	Symbol	Conditions	Specification			Units
			Min.	Typ.	Max.	
VDD rising time rate	$t_{VDD}$	BOR detector enabled			15	ms/V
		BOR detector disabled			15	ms/V
VDD falling time rate		BOR detector enabled			$\infty$	ms/V
		BOR detector disabled			$\infty$	ms/V
Analog supply Operating voltage	$V_{DDA}$	-	1.65	-	3.6	V
BOR disabled			1.65		3.6	V
BOR enabled			1.8		3.6	V
Brown-out reset threshold 0	$V_{BOR0}$	Rising edge		1.75		V
		Falling edge		1.7		V
Brown-out reset threshold 1	$V_{BOR1}$	Rising edge		2.03		V
		Falling edge		1.93		V
Brown-out reset threshold 2	$V_{BOR2}$	Rising edge		2.4		V
		Falling edge		2.3		V
Brown-out reset threshold 3	$V_{BOR3}$	Rising edge		2.66		V
		Falling edge		2.55		V
Brown-out reset threshold 4	$V_{BOR4}$	Rising edge		2.9		V
		Falling edge		2.8		V
Programmable Voltage detector threshold 0	$V_{PVD0}$	Rising edge		1.95		V
		Falling edge		1.85		V
Programmable Voltage detector threshold 1	$V_{PVD1}$	Rising edge		2.14		V
		Falling edge		2.04		V
Programmable Voltage detector threshold 2	$V_{PVD2}$	Rising edge		2.34		V
		Falling edge		2.24		V
Programmable Voltage detector threshold 3	$V_{PVD3}$	Rising edge		2.54		V
		Falling edge		2.44		V
Programmable Voltage detector threshold 4	$V_{PVD4}$	Rising edge		2.74		V
		Falling edge		2.64		V
Programmable Voltage detector threshold 5	$V_{PVD5}$	Rising edge		2.93		V
		Falling edge		2.83		V
Programmable Voltage detector threshold 6	$V_{PVD6}$	Rising edge		3.15		V
		Falling edge		3.05		V
Programmable Voltage detector Tolerance	$V_{PVD}$			$\pm 3$		%

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

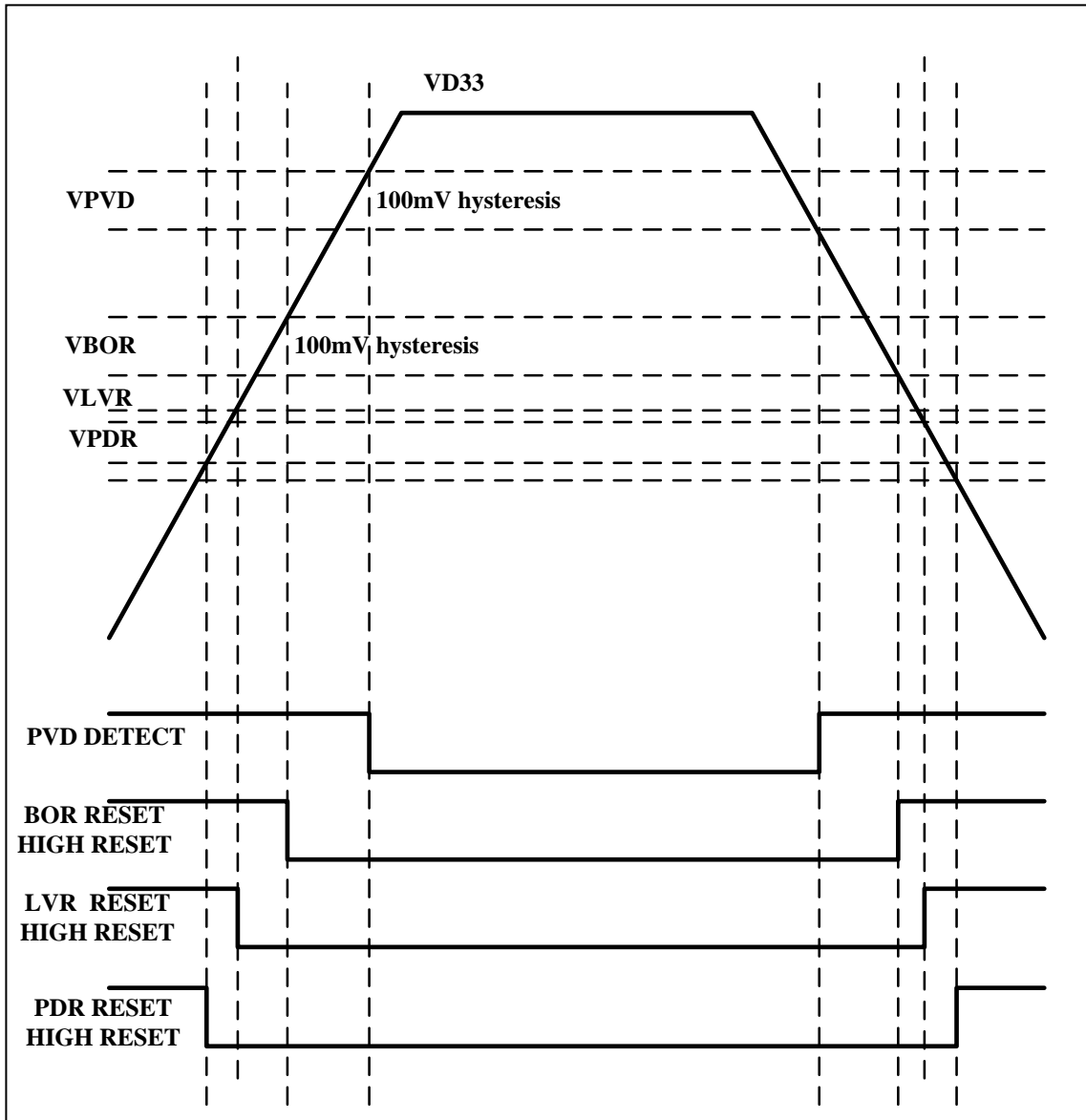
25.2.15 POR 电气特性

Parameter	Symbol	conditions	Specification			Units
			Min.	Typ.	Max.	
VDD Rising time rate	$t_{VDD}$				15	ms/V
VDD Falling time rate					$\infty$	ms/V
POR	$V_{POR}$	Rising		1.15		V
		Falling		0.4		V

(\*): These parameters are presented for design guidance only and not tested or guaranteed.



25.2.16 POWER Supply 电气特性



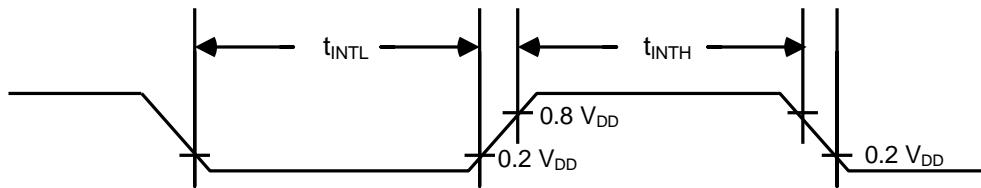
1. PVD is enabled or disabled by software
2. BOR is enabled or operating from 1.8V to 3.6V
3. PDR reset level is around at 1.0V. There is no BOR when VD33 operating from 1.65V to 3.6V and the reset is enabled when VD33 goes below PDR level.
4. LVR reset level is around at 1.35V.

**25.3 AC 电气特性**

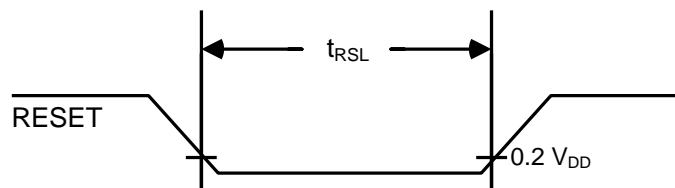
Parameter	Symbol	Pin/condition	Specification			Units
			Min.	Typ.	Max.	
Main Operation Frequency	$F_{MCP}$	$X_{IN}$	0.032		32	MHz
Main Crystal Stabilization Time(*)		$V_{D33} = 1.8V \sim 3.6V$ at HSE 16 MHz			10	ms
		$V_{D33} = 1.8V \sim 4.5V$ at MSI MHz			30	ms
		$V_{D33} = 4.5V \sim 5.5V$ at Hz		0.5	1	s
		$V_{D33} = 1.8V \sim 3.6V$ at 32768 Hz			10	s
Interrupt Input High, Low Width (IRQx)	$t_{INTH}$ , $t_{INTL}$	MCU clock = 12 MHz	167			ns
RESET Input Low Width	$t_{RSL}$	RST_NDF = 1, main clock = 12 MHz	334			ns

(\*): These parameters are presented for design guidance only and not tested or guaranteed.

**Input Timing for External Interrupts**



**Input Timing for RESET**





**25.4 Thermal 电气特性**

Parameter	Symbol	Feature	Typ.	Units	Condition
TH01	$\theta_{JA}$	Thermal Resistance (Junction to Air)	57	°C/W	64-pin LQFP package
TH02	$\theta_{JC}$	Thermal Resistance (Junction to Case)	15	°C/W	64-pin LQFP package
TH03	TJMAX	Maximum Junction Temperature	125	°C	64-pin LQFP package

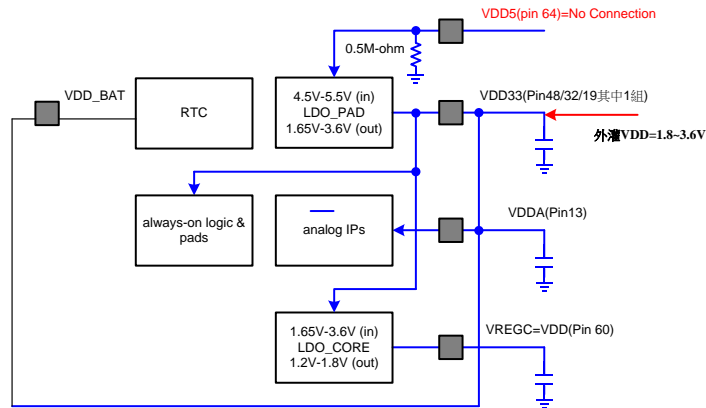
Parameter	Symbol	Feature	Typ.	Units	Condition
TH01	$\theta_{JA}$	Thermal Resistance (Junction to Air)		°C/W	48-pin LQFP package
TH02	$\theta_{JC}$	Thermal Resistance (Junction to Case)		°C/W	48-pin LQFP package
TH03	TJMAX	Maximum Junction Temperature		°C	48-pin LQFP package

Parameter	Symbol	Feature	Typ.	Units	Condition
TH01	$\theta_{JA}$	Thermal Resistance (Junction to Air)		°C/W	28-pin SOP package
TH02	$\theta_{JC}$	Thermal Resistance (Junction to Case)		°C/W	28-pin SOP package
TH03	TJMAX	Maximum Junction Temperature		°C	28-pin SOP package

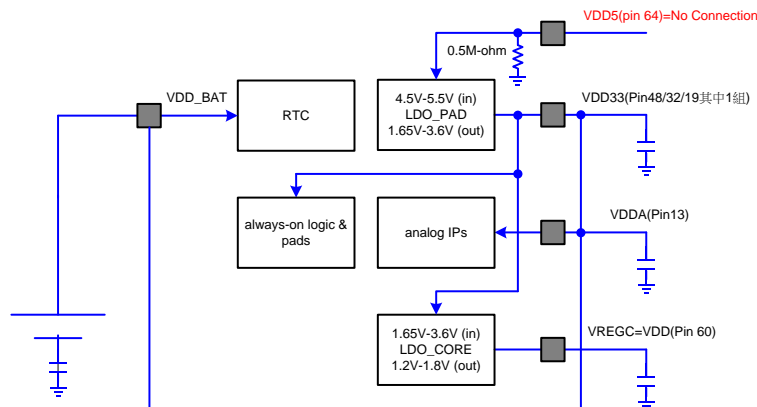
Parameter	Symbol	Feature	Typ.	Units	Condition
TH01	$\theta_{JA}$	Thermal Resistance (Junction to Air)		°C/W	20-pin SSOP package
TH02	$\theta_{JC}$	Thermal Resistance (Junction to Case)		°C/W	20-pin SSOP package
TH03	TJMAX	Maximum Junction Temperature		°C	20-pin SSOP package

## 26 应用线路

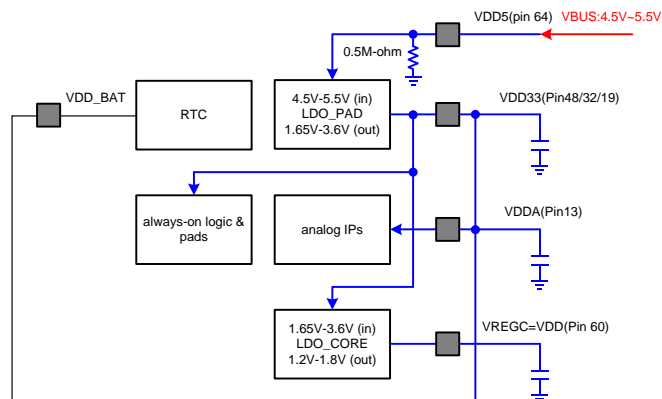
### 26.1 VDD33 power supply



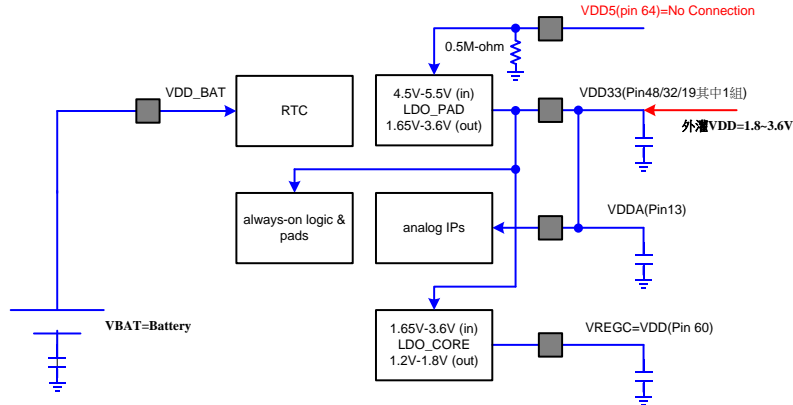
### 26.2 VDDBat power supply



### 26.3 USB Vbus 5V power supply



**26.4 VBAT & VDD33 dual power supply**



VD33	VBAT	RESET?	Current Leakage?	Note
<b>VBAT exist</b>				
N→Y	Y	Exception RTC are RESET	No	
Y→N	Y	Exception RTC are RESET	No	
<b>VD33 exist</b>				
Y	N→Y	RTC RESET The Others No RESET	No	
Y	Y→N	RTC RESET The Others No RESET	<b>No VD33 Domain</b>	<b>VBAT needs to change new battery</b>
<b>VD33 not exist</b>				
N	Y→N	All No RESET	No	All are no power supply
N	Y→N→Y	RTC RESET The Others No RESET	<b>No VBAT Domain</b>	<b>Unreasonable! VD33 needs to offer power supply for system operation</b>

## 27 产品命名规则

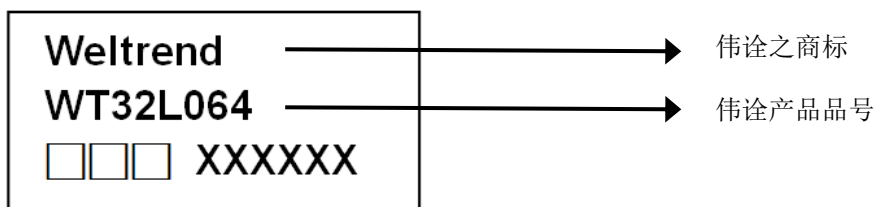
WT	Consumption market	Family	Core	Flash Size (K Bytes)		Note	Remarks
WT	32	L	0	6	4	M0/64KB Flash	32:32 位 MCU 嵌入式, 用于通用或消费市场相关产品。 L/F: L: 超低功耗系列 F:通用电源系列  0/3: 0: ARM Cortex M0 3: ARM Cortex M3
			0	3	2	M0/32KB Flash	
WT	32	F	0	6	4	M0/64KB Flash	
			0	3	2	M0/32KB Flash	
			0	3	3	M0/32KB Flash	
			3	C(12)	8	M3/128KB Flash	
			3	C(12)	9	M3/128KB Flash	
			3	6	4	M3/64KB Flash	

## 28 订单讯息

### 28.1 WT32L064

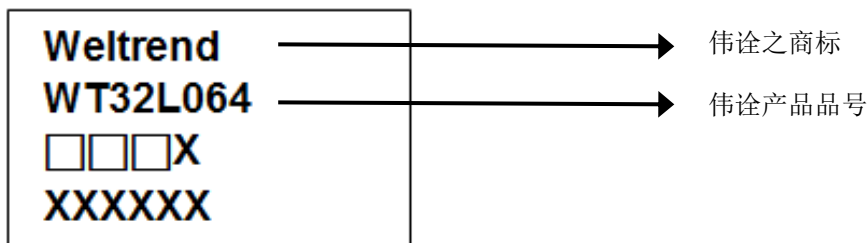
Package Type	Package Outline	Part Number
64-pin LQFP	7mm x 7mm	WT32L064-RG64AWT
48-pin LQFP	7mm x 7mm	WT32L064-RG48AWT
QFN32	5mm x 5mm	WT32L064-UG32BWT

#### 28.1.1 Top Marking – LQFP64/48



- Date Code
- X 产品追蹤码

#### 28.1.2 Top Marking – QFN32

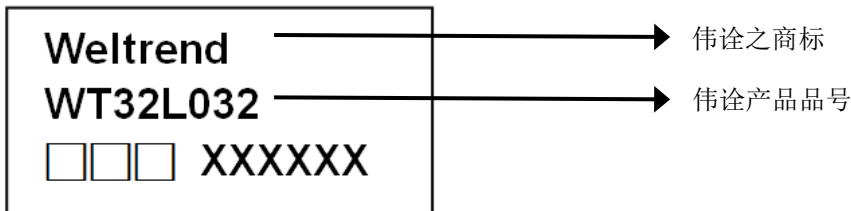


- Date Code
- X 产品追蹤码

## 28.2 WT32L032

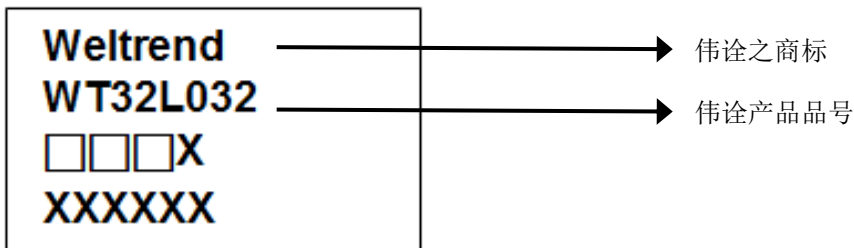
Package Type	Package Outline	Part Number
64-pin LQFP	7mm x 7mm	WT32L032-RG64AWT
48-pin LQFP	7mm x 7mm	WT32L032-RG48AWT
QFN32	5mm x 5mm	WT32L032-UG32BWT

### 28.2.1 Top Marking – LQFP64/48



- Date Code
- X 产品追蹤码

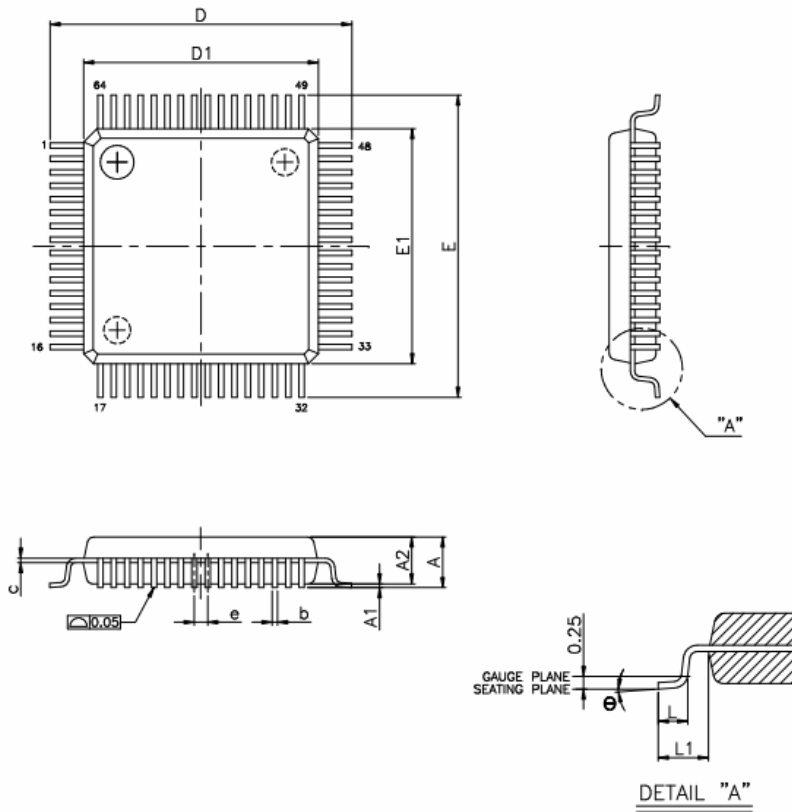
### 28.2.2 Top Marking – QFN32



- Date Code
- X 产品追蹤码

## 29 封装外观

### 29.1 LQFP-64 外观图示



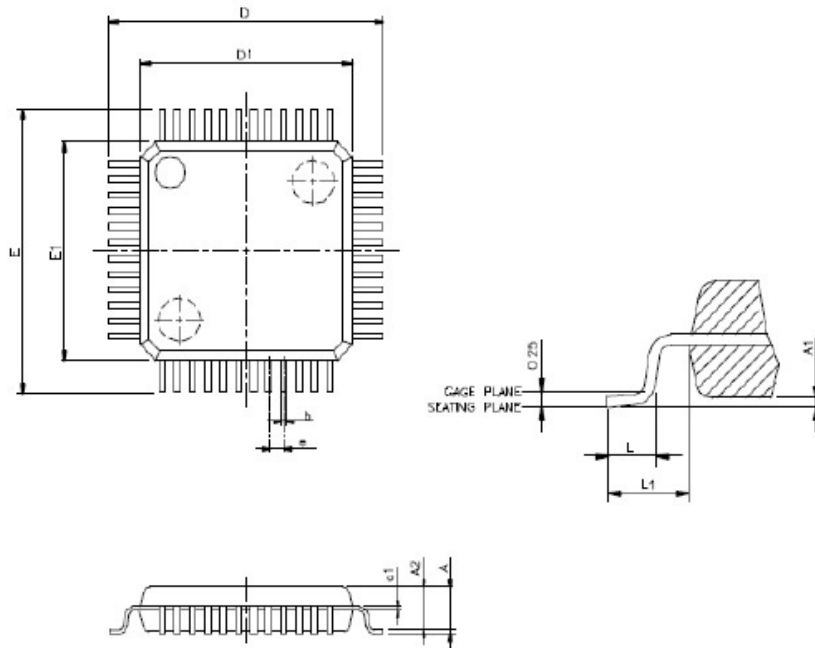
VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	—	—	1.60
A1	0.05	—	0.15
A2	1.35	1.40	1.45
b	0.13	0.18	0.23
c	0.09	—	0.20
D	9.00 BSC		
D1	7.00 BSC		
e	0.40 BSC		
E	9.00 BSC		
E1	7.00 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
$\theta$	0°	3.5°	7°

NOTES:

1. JEDEC OUTLINE : MS-026 BBD
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. D1 AND E1 ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08mm.

29.2 LQFP-48 外观图示



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	MAX.
A	--	1.6
A1	0.05	0.15
A2	1.35	1.45
c1	0.09	0.16
D	9.00 BSC	
D1	7.00 BSC	
E	9.00 BSC	
E1	7.00 BSC	
e	0.5 BSC	
b	0.17	0.27
L	0.45	0.75
L1	1 REF	

NOTES:

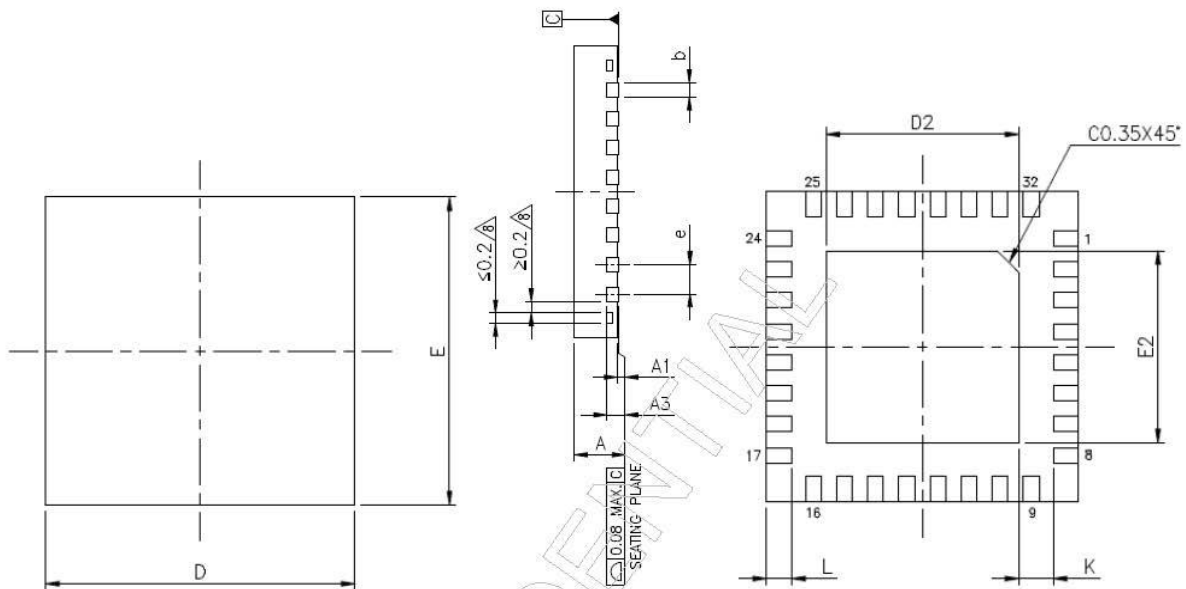
1. JEDEC OUTLINE: MS-026 BBC
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. D1 AND E1 ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08mm.



29.3 QFN-32 外观图示

Quad Flat No-Lead Plastic Package

QFN-32 PIN



SYMBOLS	MIN	NOR	MAX
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	0.203 REF		
b	0.18	0.25	0.30
D	5.00 BSC		
E	5.00 BSC		
e	0.50 BSC		
L	0.35	0.40	0.45
K	0.20	-	-
D2	3.10	3.20	3.25
E2	3.10	3.20	3.25

UNIT: mm

NOTES:

1. JEDEC outline : MO-220
2. Dimension "b" applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip. If the terminal has the optional radius on the other end of the terminal, the dimension "b" should not be measured in that radius area.
3. Bilateral coplanarity zone applies to the exposed heat sink slug as well as the terminals.

PREPARE	Cynthia	DATE: 2012/8/1
CHECK	Lawrence	DATE: 2012/8/1
APPROVE	Eric	DATE: 2012/8/1

### 30 开发工具

WT32L064/032 开发工具组可与编译程序软件 Keil MDK 协同工作。他们可以在 Windows 2000/XP/Vista/Win7/Win10 中执行电路内仿真器 (ICE) 和系统内程序设定 (ISP)。

开发工具组如下图 65 所示:

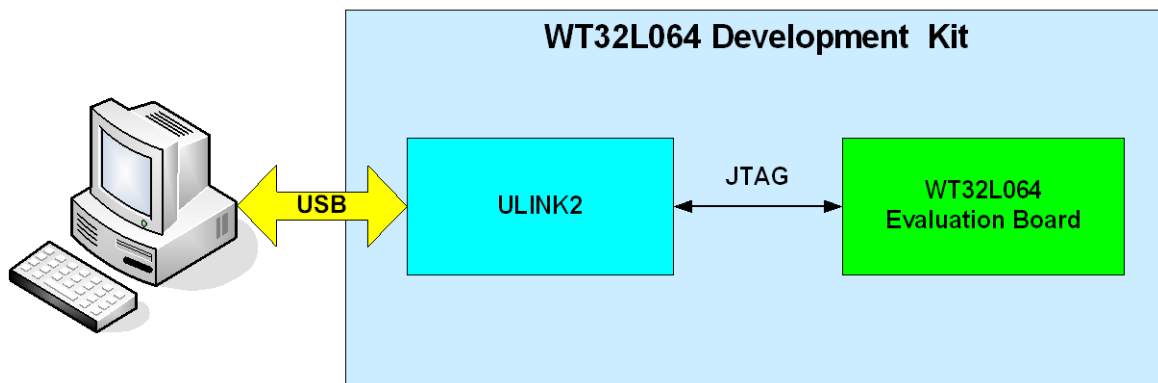


图 65 WT32L064/032 开发工具包

### 31 版本更改纪录

Version	Section/Page	Contents	Date
0.10	All	Initial version	Nov. 12, 2019
0.20	Pin description	Add A1 type statement during reset period	April 01, 2020
	All	Add WT32L032 part No.	
	RCC	Add RTC clock enable	
0.30	System clock tree & RCC & ADC & Electrical Characteristics	Add LSI:37kHz of RTC, Notice of RCC & ADC Add RTC low power clock enable bit	June 15, 2020
0.40	All		Sep. 8, 2020
0.50	7.1	Modify DMA descriptions	June 11, 2021
0.60	4.2.2.1	Update PLL descriptions	July 28, 2021
1.00	订单讯息	新增 Top Marking	Sep. 09, 2021
1.01	订单讯息	修改 Bonging 版本为 B 版 删除 “Wafer form or Chip form”	Sep. 16, 2021
	CH29	删除 “Pad 图标与坐标地址表”说明	
1.02	1.1	Update Features.	Feb. 10, 2022
	CH5	删除 “低功耗睡眠” 说明	
	20.3	更新 ADC 叙述	