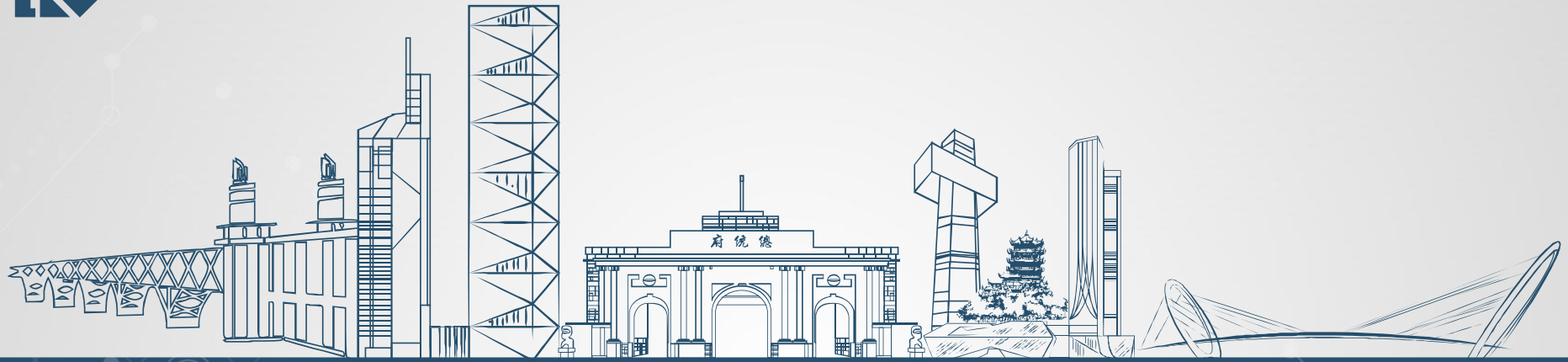


南京凌鸥创芯电子有限公司

用芯打造电控专用平台

CONTENTS

01 公司介绍

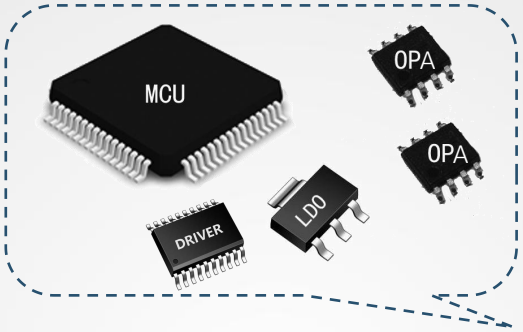
运动控制专用系列芯片

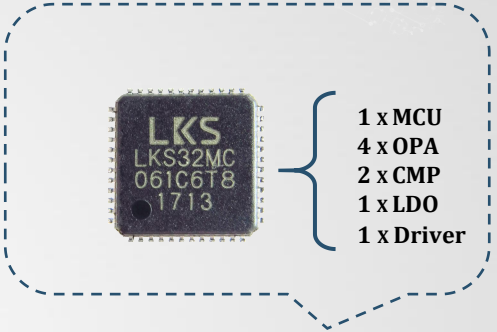
南京凌鸥创芯电子有限公司

南京凌鸥创芯电子有限公司成立于2015年8月，是一家专注于运动控制领域的集成电路设计与生产，并提供总体解决方案的国家高新技术企业。公司主营业务是运动控制核心芯片，目标市场主要为电动车辆、伺服控制、机器人、电动工具、家用电器等。

03 创新性

- 仪表级全差分可编程增益放大器+差分ADC，不需要做电压偏置就可以处理正负电流信号
- 高速12bit SAR ADC双路同步采样，速度3Msps
- 集成千分之五电压基准源
- MCU+DSP双核心，主频96MHz
- CORDIC硬件三角函数计算模块，100nS内完成SIN、COS、ARCTAN、开方运算计算；
- 使用DSP加速电机控制核心算法后，FOC电流内环可以在1.2uS内完成。
- 内部RC全温度范围时钟偏差1%以内





	竞品	凌鸥产品
工作温度	-40℃~85℃	-40℃~105℃
抗静电级别	4KV	6KV
工作频率	72MHZ	96MHZ
Flash	32K	32~64KB
ADC	2路	2路
MOSFET内阻电流采样钳位电路	无	支持
差分PGA	无	4路
DAC	无	1路
动态增益调节	不支持	支持
比较器	无	2路
电机HALL接口	无	1个

02

LKS32MC08系列

性能

- 96MHz 高性能Cortex-M0内核
- 集成自主指令集电机控制专用DSP
- 超低功耗睡眠模式，低功耗休眠电流30uA
- 工业级工作温度范围
- 超强抗静电和群脉冲能力

存储器

- 64kB/32KB Flash，带加密功能
- 2KB NVR　1KB存校正参数，1KB用户区
- 8kB RAM

工作范围

- 2.2V~5.5V电源供电，内部集成1个LDO，为1.5V数字电路供电
- 温度范围:-40至105°C

时钟

- 内置4MHz高精度RC时钟，-40~105℃范围内精度在±1%之内
- 内置低速32KHz 低速时钟，供低功耗模式使用
- 4MHz外部晶振
- 内部PLL最高可提供96MHz时钟

外设模块

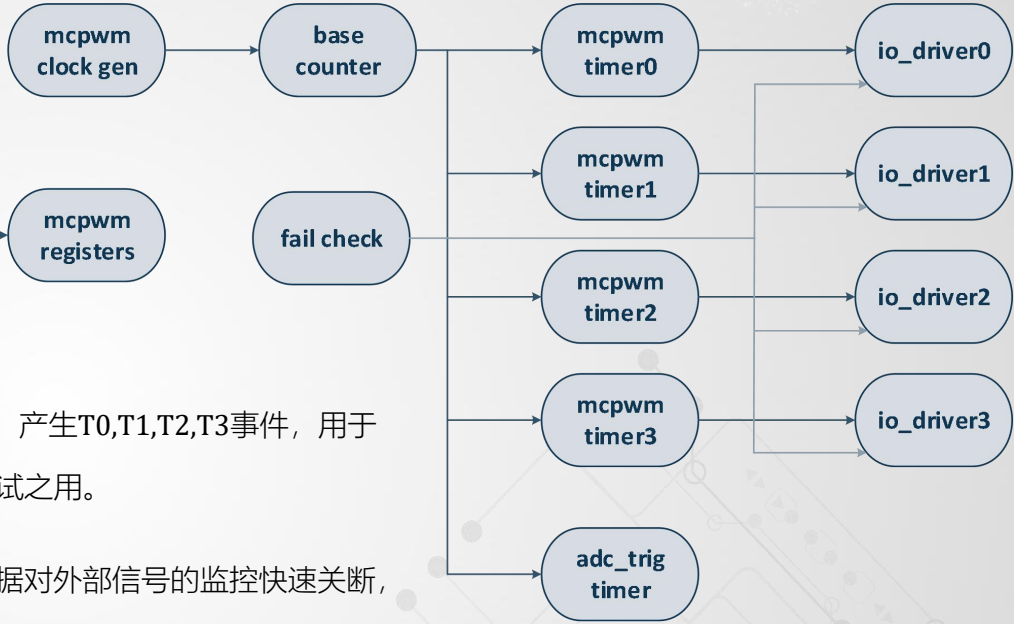
- 集成UART/SPI/I2C/CAN串行通信接口
- 2个通用16位Timer，支持捕捉和边沿对齐PWM功能
- 2个通用32位Timer，支持捕捉和边沿对齐PWM功能；支持正交编码输入，CW/CCW输入，脉冲+符号输入
- 电机控制专用PWM模块，支持8路PWM输出，独立死区控制
- Hall信号专用接口，支持测速，去抖功能
- 硬件看门狗
- 低功耗唤醒模块，支持定时唤醒和4个GPIO唤醒。
- 最多 4 组 16bit GPIO，16个GPIO可作为外部中断源输入

模拟模块

- 集成1路12bit SAR ADC，同步双采样，最高采样率3MSps，共20通道
- 集成4路运算放大器，可设置为差分PGA模式，内置反馈电阻，放大倍数软件可调
- 集成两路比较器，可设置滞回模式
- 集成12bit DAC数模转换器
- 内置±2℃温度传感器
- 内置高精度电压基准源
- 内置1路低功耗LDO和电源监测电路
- 集成高精度、低温飘高频RC时钟
- 集成晶体起振电路
- 集成32K RC时钟
- 集成96M Hz PLL

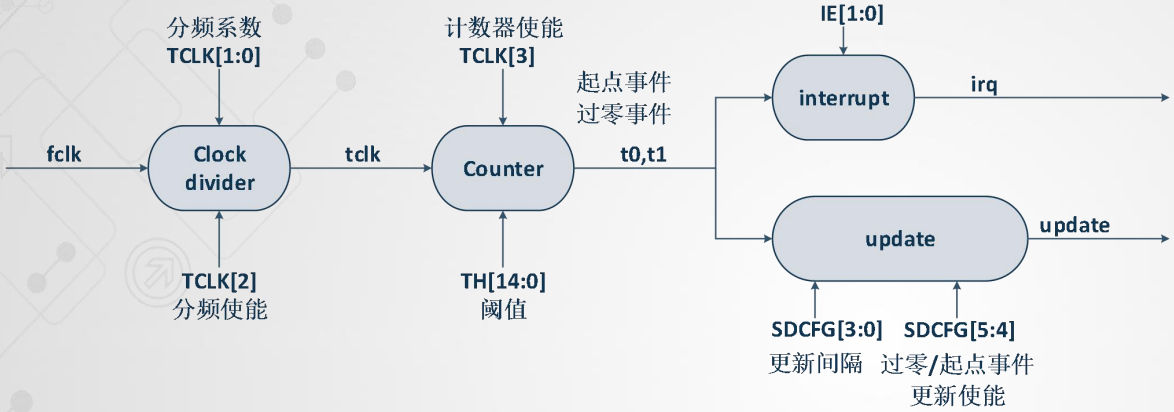
03 芯片模块MCPWM

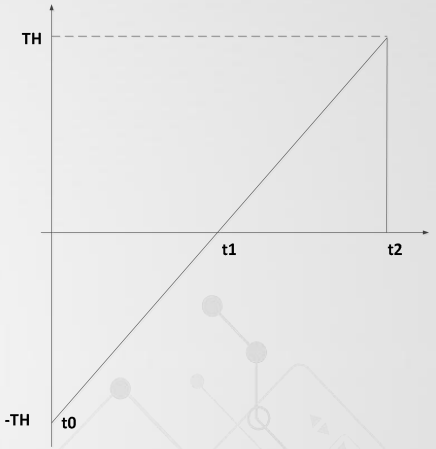
- 支持边沿对齐PWM (8路独立)，中心对齐PWM (4对互补信号)
- 移相PWM (4对互补信号)
- 计数器的时钟分频有 1/2/4/8 四种选项，产生的的分频时钟频率分别为 96MHz、48MHz、24MHz 和 12MHz。



- 包含四组Timer比较模块。产生4路和MCPWM同时基的定时信息，产生T0,T1,T2,T3事件，用于触发ADC模块同步采样。该触发信号可同时输出到GPIO，便于调试之用。
- 内部短路保护，避免因为配置错误导致短路。外部短路保护，根据对外部信号的监控快速关断，外部短路保护包含一组急停保护模块，用于快速关断MCPWM模块输出而不依赖MCU的处理。MCPWM模块可输入4路急停信号，其中两路来自外部IO，两路来自片内比较器。
- MCPWM的每个输出IO支持两种控制模式----PWM硬件控制或者软件直接控制（用于电机的刹车控制，或BLDC方波换相控制）

03 MCPWM主计数器



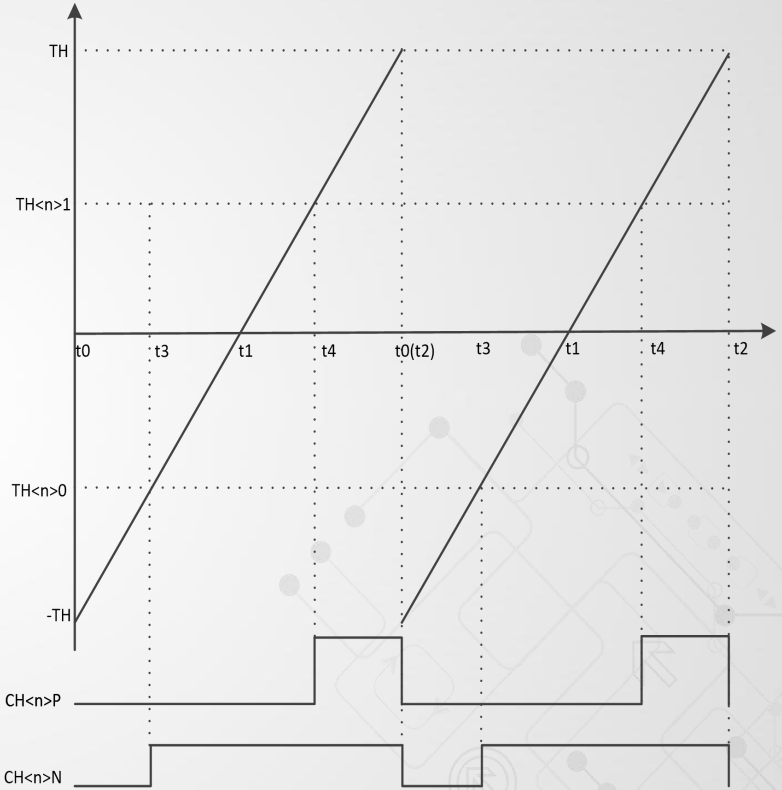


- 8路MCPWM共用一个基础周期。主计数器是由一个16位递增计数器组成，其计数门限值为TH，计数器从t0开始从-TH递增计数（UP），在t1过0，在t2计数到TH完成一次计数循环，回到-TH，重新开始计数。计数周期为（2xTH+1）/fclk。fclk是计数时钟频率。

- 周期寄存器和占空比寄存器，可实现手动更新，也可以硬件自动更新，更新周期可通过MCPWM_SDCFG[3:0]设置。硬件更新可配置为t0或t1时刻更新，以及t0 t1时刻都更新三种模式，硬件把加载寄存器的值载入到实际运行的寄存器中。如需要立刻更新，需要直接操作UPDATE寄存器，更新间隔可设置，最大16次 t1事件更新1次寄存器。

03 MCPWM边沿对齐模式

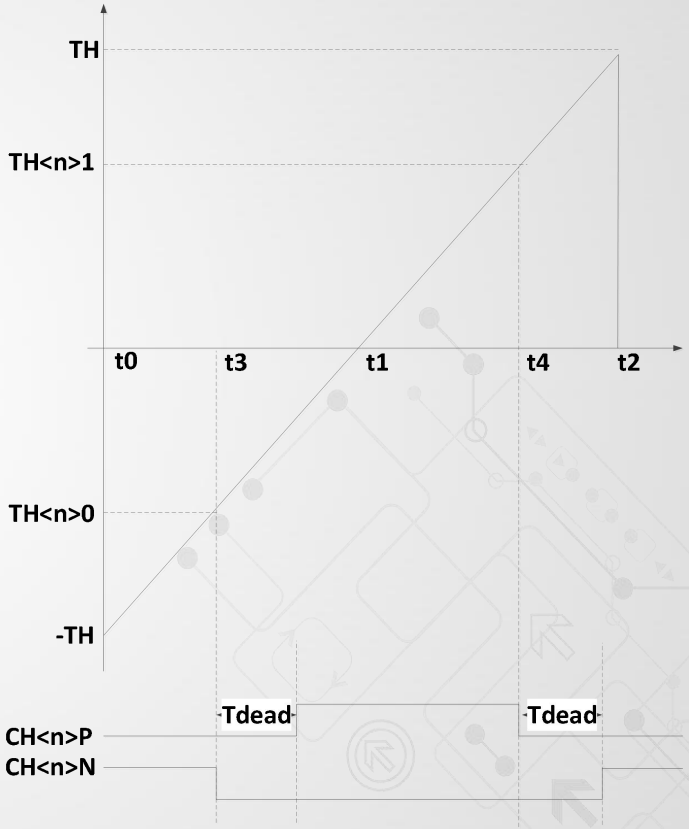
- 边沿对齐模式中，在 t0 时刻 CH<n>P/CH<n>N 同时置 0，在 t3 时刻，CH<n>N 变高，在 t4 时刻，CH<n>P 变高。
- 寄存器TH<n>0控制T3时刻，寄存器TH<n>1控制T4时刻。
- 通常边沿对齐模式，不需要死区控制。



03

MCPWM中心对齐模式

- 中心对齐模式中，采用TH<n>0和TH<n>1控制第<n>个MCPWM IO的启动、关闭动作，n为1/2/3/4。
- 寄存器TH<n>0控制T3时刻，寄存器TH<n>1控制T4时刻。
- 当计数器CNT值向上计数达到TH<n>0时，在t3时刻关闭CH<n>_N，经过死区延时Tdead，打开CH<n>_P。
- 当计数器CNT值向上计数达到TH<n>1时，在t4时刻关闭CH<n>_P，经过死区延时Tdead，打开CH<n>_N。
- 采用独立死区时间控制（8个寄存器），共享数据更新事件。死区延时保证 CH<n>P/CH<n>N 不会同时为高，避免短路发生。
- 采用独立的启动和关闭时间控制，可以提供相位控制的能力。
- t3/t4 时刻均会产生相应中断。

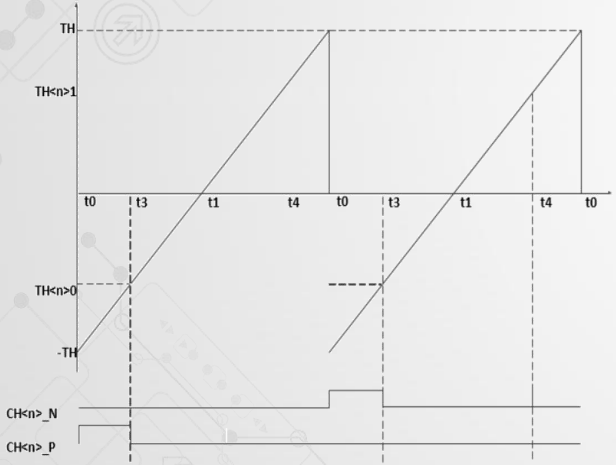


03

MCPWM推挽输出模式

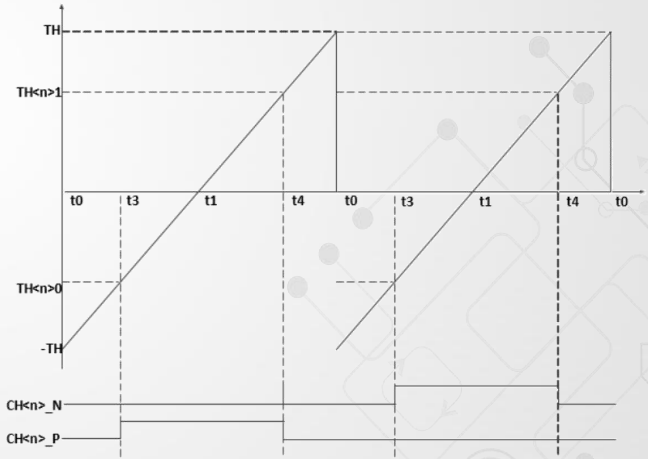
- 边沿对齐推挽模式。第一个周期内，在 t0 时刻 CH<n>P 置 1，在 t3 时刻，CH<n>P 变低。第二个周期内，在 t0 时刻 CH<n>N 置 1，在 t3 时刻，CH<n>N 变低。

- t0/t3 均会产生相应中断。



- 中心对齐互补推挽模式。第一个周期内，在 t3 时刻 CH<n>P 置 1，在 t4 时刻，CH<n>P 变低。第二个周期内，在 t3 时刻 CH<n>N 置 1，在 t4 时刻，CH<n>N 变低。

- t3/t4 均会产生相应中断。

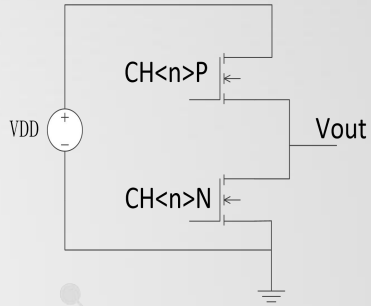


◆ **推挽输出模式主要用于开关电源应用的移相全桥控制**

03 MCPWM死区控制

- MCPWM IO 是一对互斥控制信号 CH<n>P/CH<n>N，控制如右图所示的电路，

CH<n>P状态	CH<n>N状态	输出状态
高	低	Vout 输出高（VDD）
低	高	Vout 输出低（VSS）
低	低	Vout 输出不确定
高	高	Vout 输出不确定，但是会产生 VDD 到 VSS 的短路

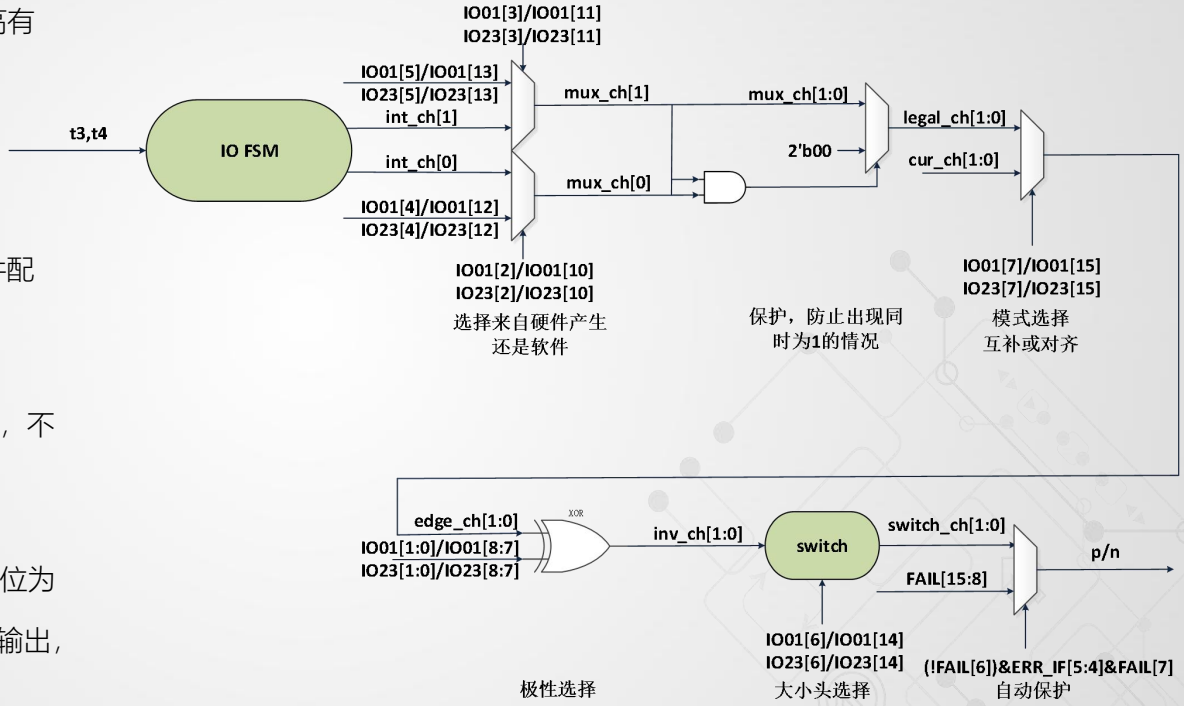


MCPWM IO 控制示意图

- 必须避免 CH<n>P/CH<n>N 同时为低的情况，死区的引入，可以有效避免 VDD 到 VSS 的短路。
- 四组 MCPWM IO 的死区宽度可独立调整。对于互补模式 MCPWM IO 自动插入死区。对于边沿对齐模式，MCPWM IO 无死区。
- MCPWM IO 也可通过软件配置的方式输出，此时，死区控制通过软件实现，如果 PWM 模式为互补，仍然由硬件保证不同时为高或者为低。
- 在 IO Driver 模块中增加 CH<n>P/CH<n>N 冲突检测，发生冲突时，自动将 IO 拉低，同时给出错误中断（中断保持，直到 MCU 写 0）。

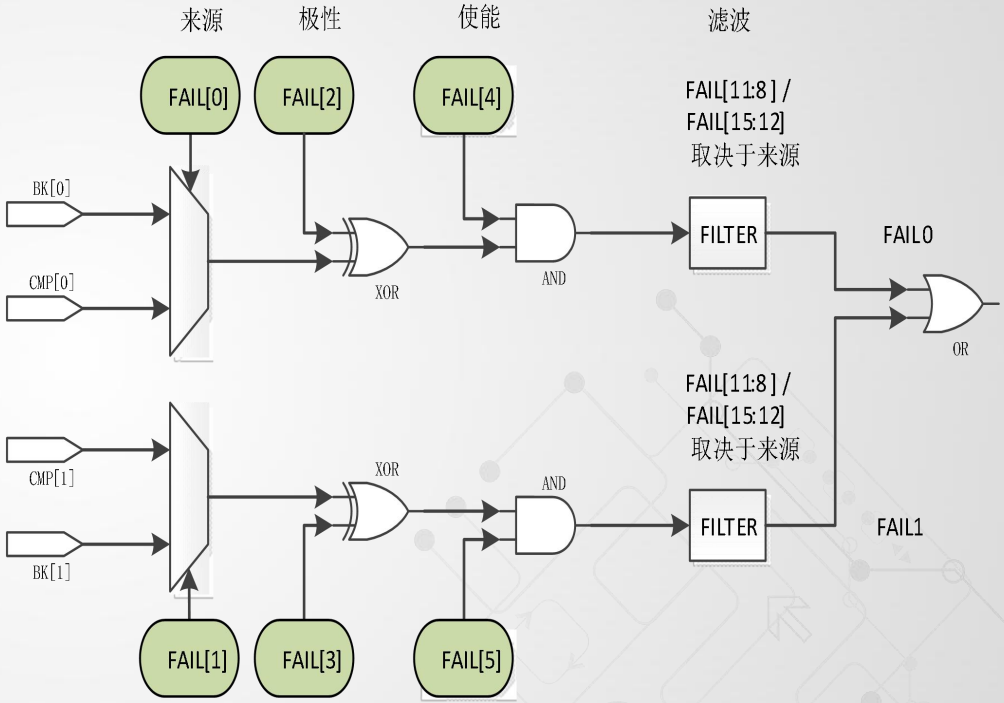
03 MCPWM输出极性控制

- CH<n>_P/CH<n>_N的有效电平可以配置为高有效/低有效。每个IO的有效电平单独可配。
- CH<n>_P/CH<n>_N信号极性可取反输出。
- CH<n>_P/CH<n>_N输出到IO的位置通过软件配置可以互换。
- 具有硬件保护逻辑，保证在中心对齐模式下，不会出现上下管同时导通情况。
- FAIL[7]控制芯片Debug时刻的输出状态，此位为0时，在芯片调试中MCU Halt时，PWM停止输出，输出FAIL[15:8]的默认电平值。此位为1时，MCPWM保持正常工作，IO口输出PWM波形。



03 MCPWM紧急停控制

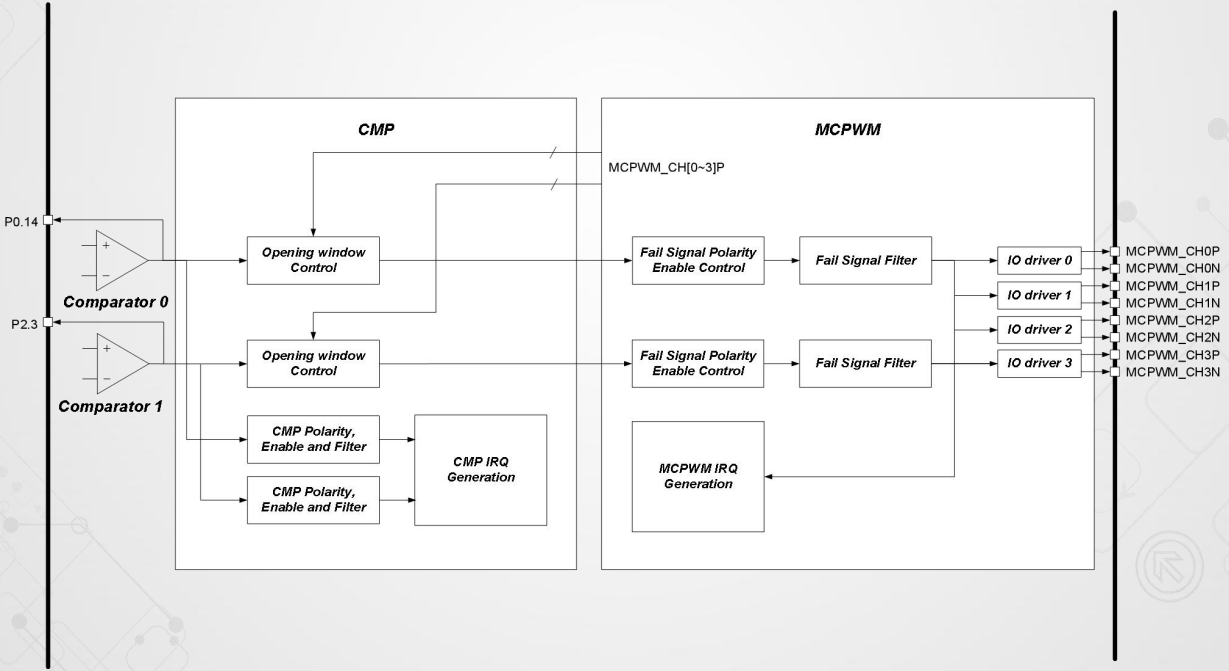
- MCPWM急停控制模块，实现硬件快速关断PWM的输出，不需要MCU干预。
- 共有两个通道FAIL0和FAIL1，有4个源头BK[1:0]和CMP[1:0]。BK来自IO（MCPWM_BKIN0/MCPWM_BKIN1），CMP来自芯片内部的比较器模块。
- MOE位（FAIL[6]）为PWM模块的总开关，为1时，IO输出电平受到MCPWM模块控制，输出PWM波形。MOE为0时，IO输出默认电平。当发生FAIL事件，硬件立刻将P/N端输出自动切换到默认电平状态，同时清零MOE位。IO口默认电平由寄存器FAIL[15:8]控制。



- FAIL0,FAIL1 会对信号进行 16 个滤波时钟的滤波，即只有信号稳定时间超过 16 个滤波周期才能通过滤波器。即滤波宽度=滤波时钟周期*16。

03 MCPWM注意事项

来自比较器的FAIL为模拟比较器输出的原始信号，未经过比较器数字接口模块的滤波处理，但是可以被MCPWM_CHnP进行开窗控制，开窗控制设置见比较器数字接口模块。FAIL信号进入MCPWM模块后，可以通过MCPWM_TCLK进行滤波。



03 MCPWM注意事项

MCPWM默认电平通过MCPWM_FAIL[15:8]设置，当发生 FAIL 事件或 MOE 为 0 时，相应通道出默认输平。默认电平输出不受 MCPWM_IO01 和 MCPWM_IO23 的BIT0、BIT1、BIT8、BIT9、BIT6、BIT14 通道交换和极性控制的影响，直接控制通道输出。

MCPWM特殊输出状态

电机控制中经常会用到全零和全 1 输出状态，以下互补模式设置可以得到期望的输出。

1. 如果 THn0≥THn1，芯片处于恒 0 状态（CH<n>P 关闭，CH<n>N 开启），无死区

2. 如果 THn0=-TH，THn1=TH，芯片处于恒 1 状态（CH<n>P 开启，CH<n>N 关闭），无死区

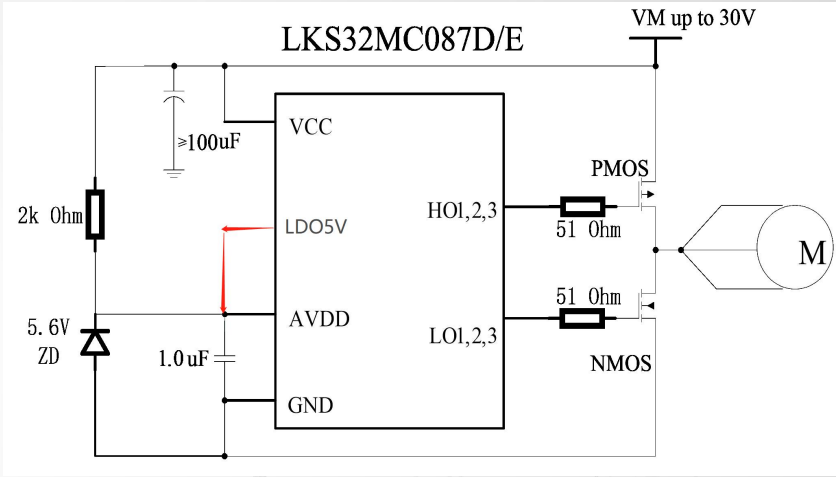
驱动模块的输出引脚信号 LO1/HO1 对应 MCU MCPWM_CH0N/MCPWM_CH0P 的 MCPWM 功能输出，LO2/HO2对应 MCU GPIO MCPWM_CH1N/MCPWM_CH1P 的 MCPWM 功能输出，LO3/HO3 对应 MCU MCPWM_CH3N/MCPWM_CH3P 的 MCPWM 功能输出，同时 P3.13/ P1.12/ P1.15 这 3 个 GPIO 需设置为输出态。

03 电源管理系统

- 电源管理系统由 LDO15 模块、上电/掉电复位模块（POR）组成。
- 该芯片由 3.3~5V 单电源供电，以节省芯片外的电源成本。芯片内部集成一路 LDO15 给内部所有数字电路、PLL 模块供电。
- LDO 上电后自动开启，无需软件配置，但 LDO 输出电压可通过软件实现微调。
- LPOR 模块监测 LDO15 的电压，在 LDO15 电压低于 1.25V 时（例如上电之初，或者掉电时），为数字电路提供复位信号以避免数字电路工作产生异常。
- HPOR 模块监测 AVDD 的电压，在 AVDD 电压低于 2.3V 时（例如上电之初，或者掉电时），为数字电路提供复位信号以避免数字电路工作产生异常。
- 08x自带PVD 模块对 5V 输入电源进行检测，如低于某一设定阈值，则产生报警（中断）信号以提醒 MCU。中断提醒阈值可通过寄存器PVDSEL[1:0]设置为不同的电压（3.6V，3.9V，4.2V，4.5V）。PVD模块可通过设置PD_PDT=1关闭。当发生电压过低事件时，会触发产生电源电压过低中断，对应中断号 17。

03 电源管理系统 – 内置3P3N驱动

- 部分芯片自带LDO5V/30mA，使用该LDO5V代替外部LDO对芯片和外供电时，如果芯片发热严重推荐电路设计方案如下：
- 在 VCC 高于 20V、且芯片无需休眠的应用场合，建议在 VCC 和 AVDD 之间加一个 1k~2k 欧姆的分流电阻，此电阻并在内部 5V LDO 的输入和输出端之间，用于旁路电流，降低芯片自身发热。电阻需放置在离开芯片一定距离的位置以分散热量。
- 电阻阻值的计算需遵循如下公式：R>=(VCC-AVDD)/I。
- 其中 I 为 5V 电源上的总功耗，包括 MCU 的功耗、5V 外围器件（例如 HALL）的功耗。外部跨接分流电阻的情况下，在 AVDD 脚应放一个 5.6V 的稳压管。
- VCC 引脚到地之间必须有一个大于等于 100uF 的去耦电容。
- 正常情况不推荐对外供电，芯片发热严重。

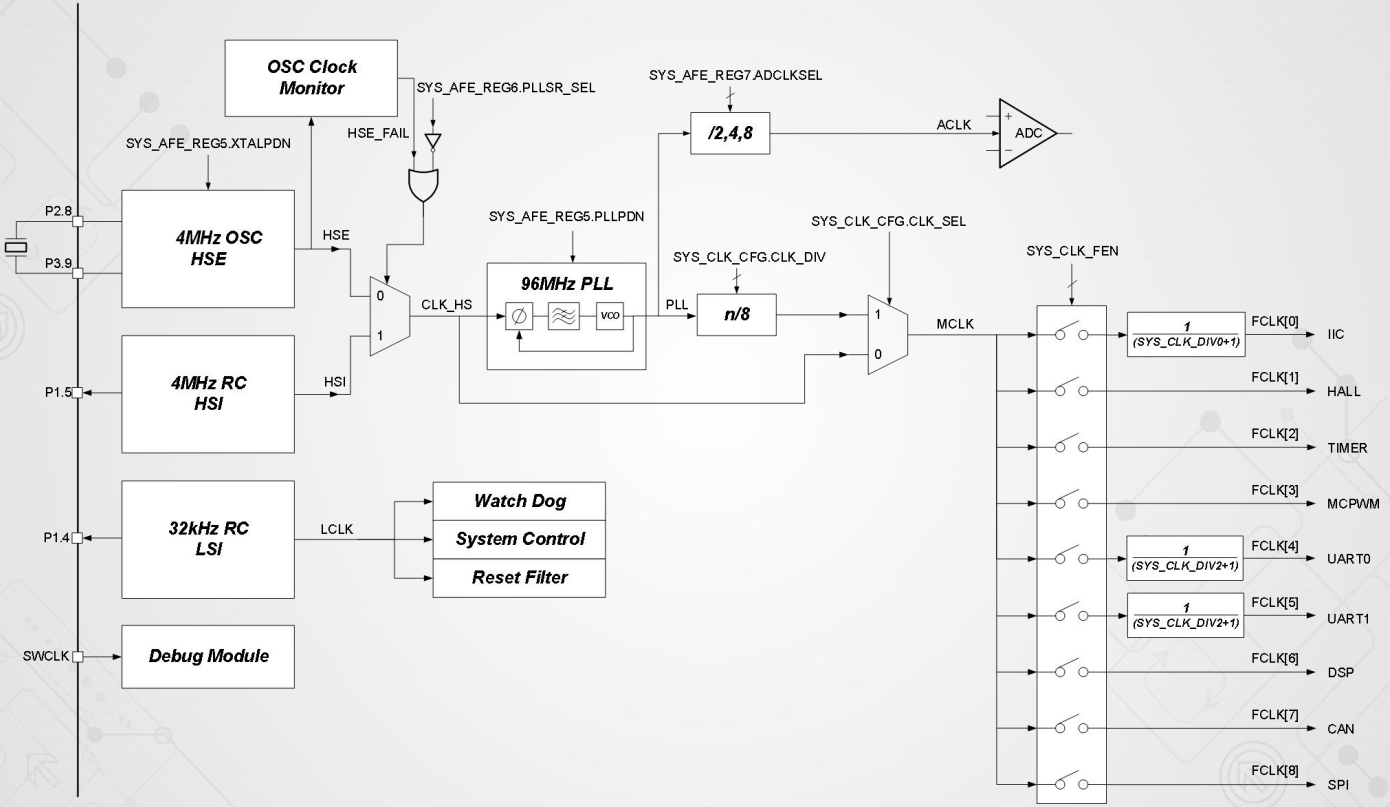


驱动模块典型应用图

03 CLOCK 时钟系统

时钟源	频率	来源	误差	说明
LSI	32KHz	内部RC振荡器	23KHz~42KHz	内部系统管理时钟，用于WDT，复位信号的滤波和展宽
HSI	4MHz	内部RC振荡器	-40~105° ±1% -085和086 -40~125°±1.5%	可作为PLL源时钟
PLL	96MHz	PLL时钟	同HSI一致	PLL输出时钟，以HSI/HSE作为输入，输出是HSI/HSE时钟的24倍频，作为系统主时钟。
HSE	4MHz	外部晶体振荡器	--	外部晶体，在对时钟精度有严格要求(例如ppm级别的精度要求)的应用下，可使用HSE作为PLL输入时钟来产生96M的系统主时钟
SWD	1MHz	调试器	--	SWD的JTAG时钟

03 CLOCK 时钟系统

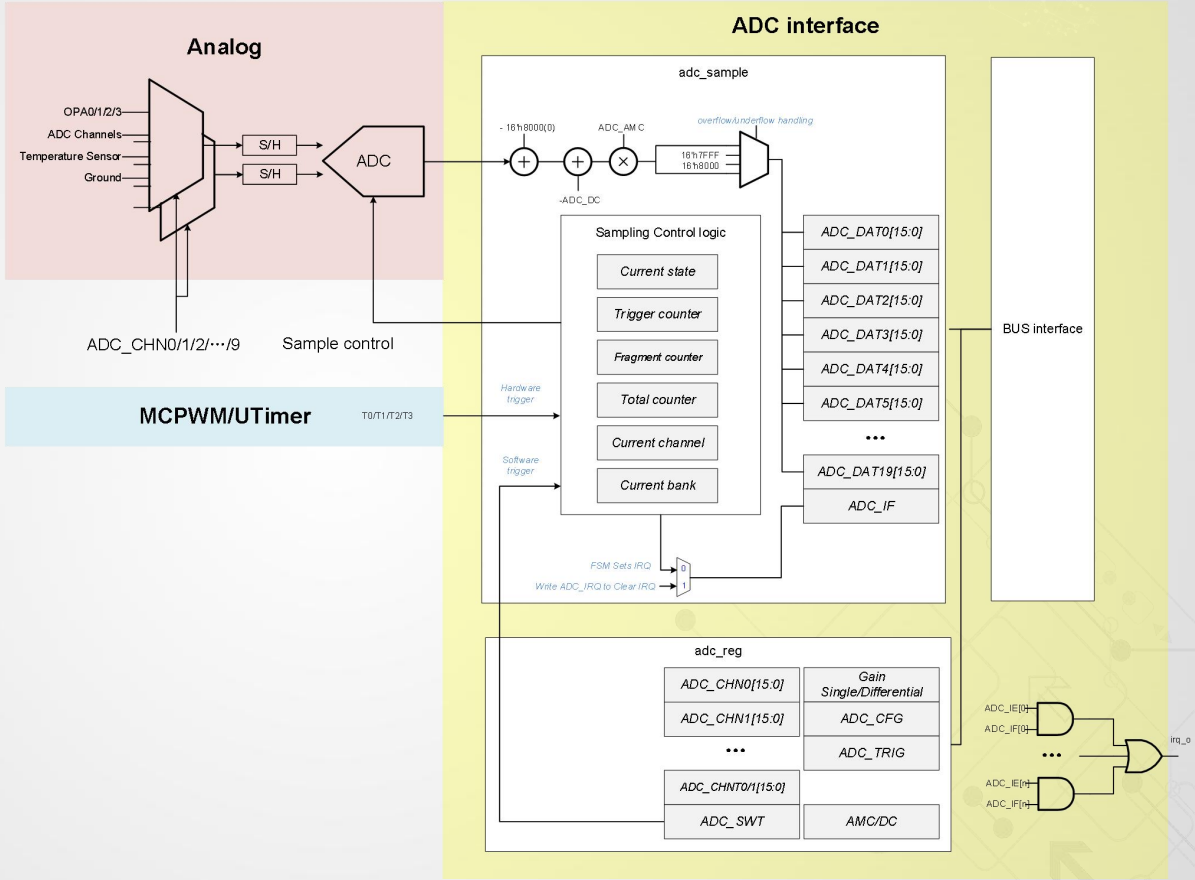


03 BGP 基准电压源

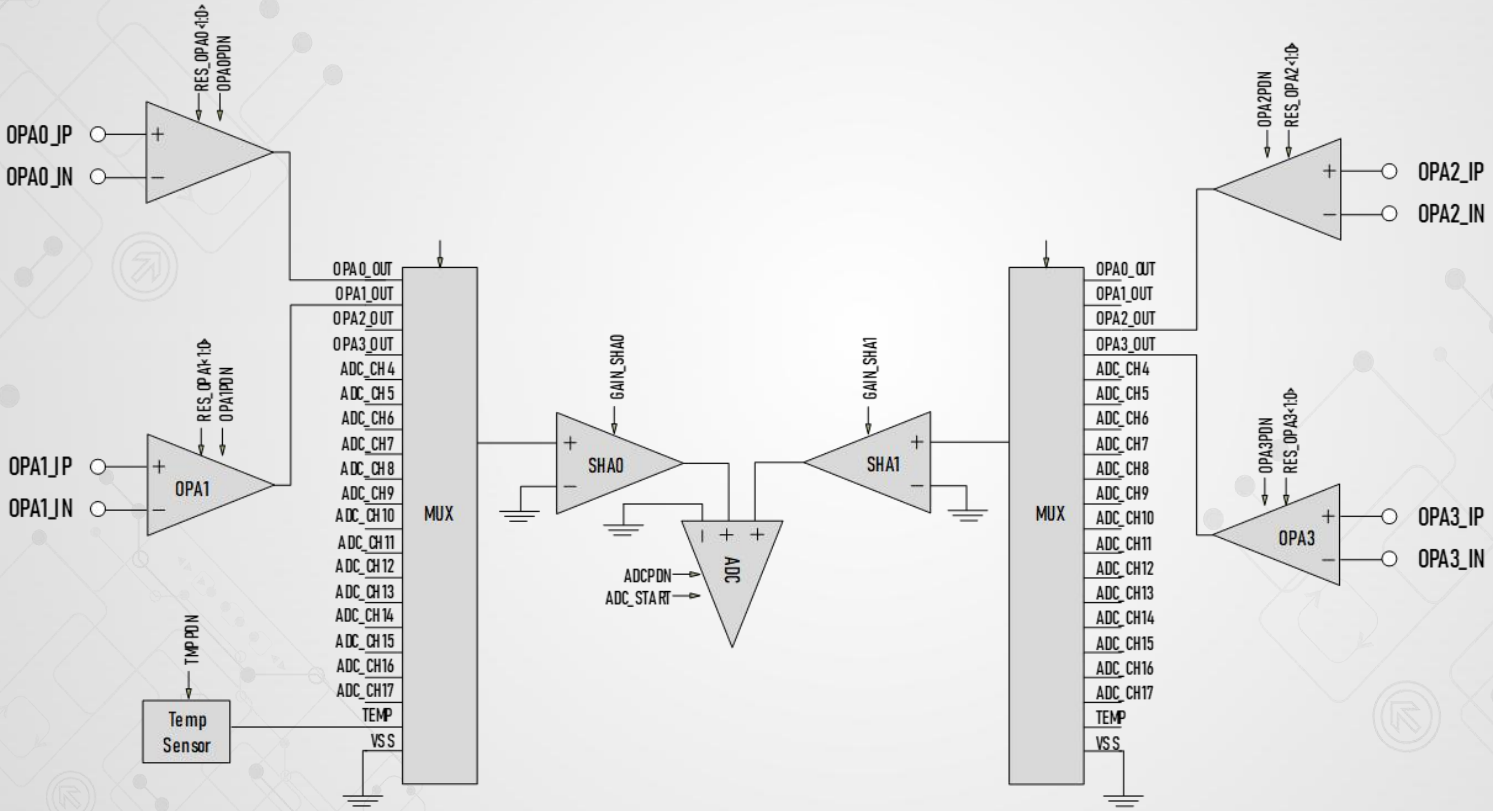
- 基准源电路(BGP REF: Bandgap reference)为 ADC、DAC、RC 时钟、PLL、温度传感器、运算放大器、比较器和 FLASH 提供基准电压和电流，使用上述任何一个模块之前，都需要开启 BGP 基准电压源。
- 芯片上电的默认状态下，BGP 模块是开启的。通过设置 BGPPD ='0'将基准源打开，从关闭到开启，BGP 需要约 2us 达到稳定。BGP 输出电压约 1.2V，精度为±0.5%。
- 基准电压源的电压大小可通过寄存器 REFTRIM_L<1:0>、REF_LTRIM、REFTRIM <2:0>进行设置，芯片出厂前基准源已经过校正，一般情况下，用户不需要额外配置这些寄存器。如需微调电压，需要读取原配置值，在此基础加上微调量，再将对应的配置值填入相应的寄存器。
- 基准源可通过设置 REF_AD_EN=1，将基准电压送至 IO P2.3 进行测量。

03 ADC模块

- 12bit SAR ADC
- 20路输入通道
- 3MHz转换速率
- 支持同步双采样，即同时采样两个通道信号，然后分别先后转换成数字量。
- 触发源：
  - MCPWM触发
  - UTIMER触发
  - 软件触发
- 工作模式
  - 单段触发
  - 两段触发
  - 四段触发
  - 支持连续采样
  - DAT0具有阈值中断机制



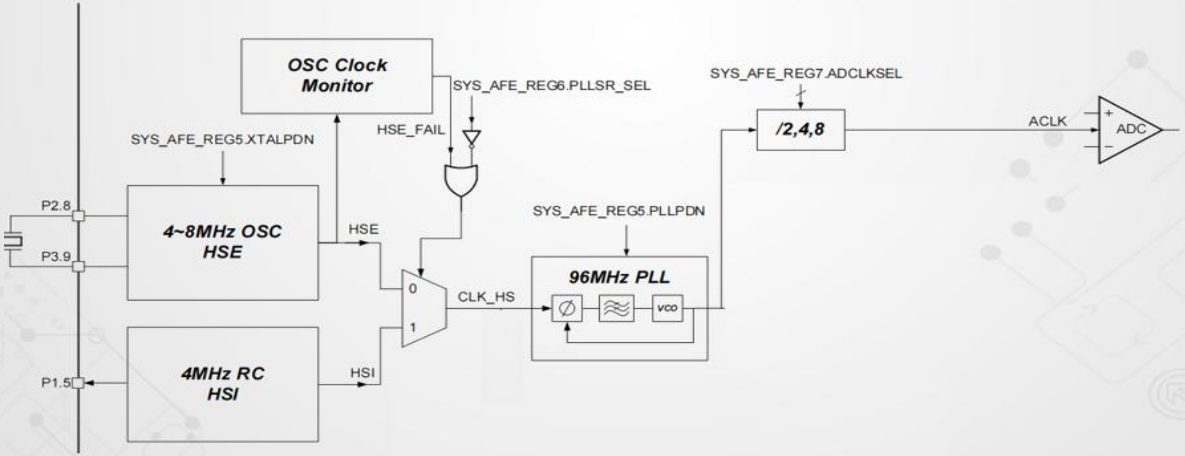
03 08x_ADC通道



通道	信号来源
0	OPA0_OUT
1	OPA1_OUT
2	OPA2_OUT
3	OPA3_OUT
4	ADC_CH4
5	ADC_CH5
6	ADC_CH6
7	ADC_CH7
8	ADC_CH8
9	ADC_CH9
10	ADC_CH10
11	ADC_CH11
12	ADC_CH12
13	ADC_CH13
14	ADC_CH14
15	ADC_CH15
16	ADC_CH16
17	ADC_CH17
18	Temp Sensor
19	VSS

03 08x_ADC工作时钟

- ADCCLK来自模拟PLL时钟分频，可通过配置ADCLKSEL@SYS_AFE_REG7来进行分频选择2/4/8倍分频，最高频率48MHz；
- ADC数字接口电路总线测时钟与CPU时钟同源，最高96MHz，与ADCCLK作异步时钟处理；
- ADC的工作周期为16个ADC时钟，即，对于最高48MHz的ADCCLK，ADC的转换频率为3MHz，每次转化时间为333nS。其中3个周期为采样时间，12个周期为转换时间，二者之间有1个周期的空档。采样和转换流水进行，ADC的第一次转换，数据在19个周期后得到，之后每16个周期获得一个ADC转化值。



03 08x_ADC通道选择

- 每个ADC模块有10个通道信号来源寄存器，控制采样序列0~19的信号选择。
- ADC_CHN0控制采样0~1，ADC_CHN2控制采样2~3，…，ADC_CHN9控制序列18~19
- 每个序列选择范围都是0~19，对应通道0~19，也就是可以对某一个通道进行多次采样。
- 每一个采样对应一个结果寄存器，转换结束后，可以直接到对应结果寄存器中获取到ADC采样结果。

ADC采样序列	对应结果寄存器	对应信号来源选择寄存器
第0次采样	ADC_DAT0	ADC_CHN0[4: 0]
第1次采样	ADC_DAT1	ADC_CHN0[12: 8]
第2次采样	ADC_DAT2	ADC_CHN1[4: 0]
第3次采样	ADC_DAT3	ADC_CHN2[12: 8]
第4次采样	ADC_DAT4	ADC_CHN2[4: 0]
第5次采样	ADC_DAT5	ADC_CHN2[12: 8]
	……	
	……	
第18次采样	ADC_DAT18	ADC_CHN9[4: 0]
第19次采样	ADC_DAT19	ADC_CHN9[12: 8]

03 ADC触发方式

- **软件触发**：软件触发，向ADC_SWT写入0x5AA5可以产生一次软件触发信号。
- **UTIME 触发**：硬件触发，ADC 的触发来源为 UTimer 的比较事件，ADC 四段触发采样事件对应 UTimer_T0/ UTimer_T1/ UTimer_T2/ UTimer_T3；其依次对应 Timer2 通道 0/1、Timer3 通道 0/1 的比较事件。
- **MCPWM 触发**：硬件触发，MCPWM 可以提供 ADC 采样控制。当计数器计数到 MCPWM_TMR0/ MCPWM_TMR1/ MCPWM_TMR2/ MCPWM_TMR3 时，可产生定时事件MCPWM_T0/ MCPWM_T1/ MCPWM_T2/ MCPWM_T3，触发 ADC 采样。该触发信号可同时输出到 GPIO，便于调试之用。
- ADC无论被配置为几段转换模式，每一段都需要一个触发事件才能进行开始采样转换，否则ADC处于等待状态。单段触发模式可以是一次硬件事件即触发，也可以是达到一定次数的硬件事件才触发采样，4个MCPWM/UTimer触发事件均可触发或参与计数；但两段触发和四段触发，4个MCPWM/UTimer触发事件只能按顺序触发ADC采样。

触发源	单段触发	两段触发	四段触发
MCPWM/UTimer	C次T0 C次T1 C次T2 C次T3 C次T0/T1/T2/T3	第一段T0 第二段T1	第一段T0 第二段T1 第三段T2 第四段T3
软件触发	软件触发	第一段软件触发 第二段软件触发	第一段软件触发 第二段软件触发 第三段软件触发 第四段软件触发

03 ADC转换模式

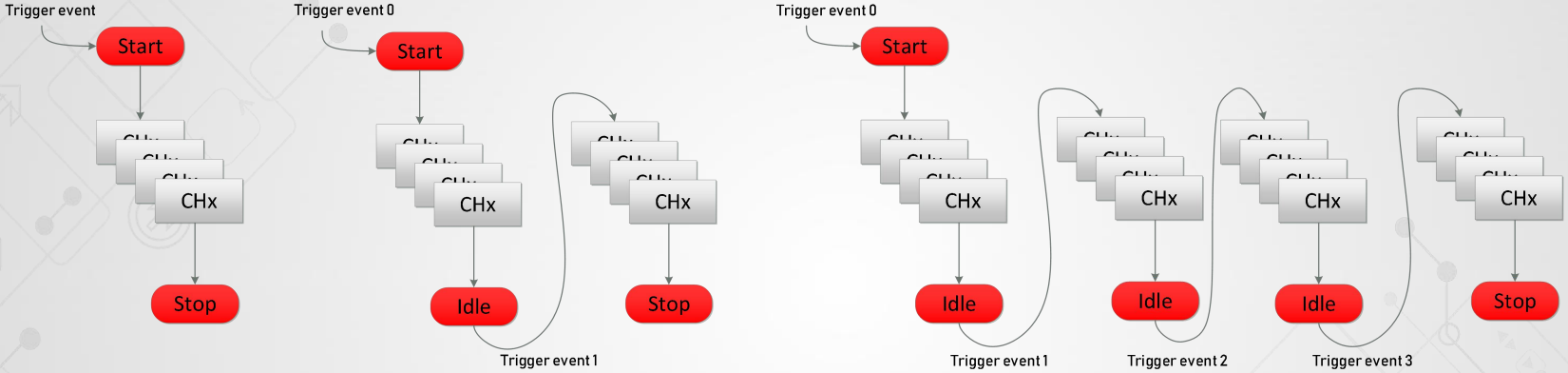
- ADC有单段触发、两段触发、四段触发四种转换模式
- 各段采样的通道数由ADC_CHNT0/1进行控制。(1表示1个通道，2表示2个通道，…，12表示12个通道)
- ADC具备连续采样模式，在连续采样模式下（通过ADC_TRIG[14]设置）无需触发信号，ADC完成一轮采样后自动开始下一轮采样

例如，配置四段触发模式如下表：

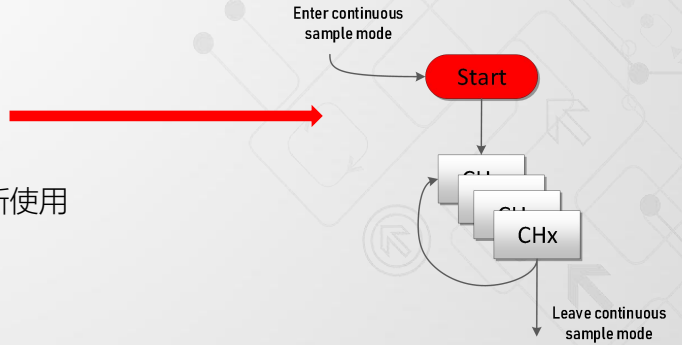
第n段	ADC_CHNT0/1	寄存器数值	采样的通道数
1	ADC_CHNT0[4: 0]	4	4
2	ADC_CHNT0[12: 8]	1	1
3	ADC_CHNT1[4: 0]	6	6
4	ADC_CHNT1[12: 8]	1	1

03 ADC转换模式

- 单段触发、两段触发、四段触发共三种转换模式示意图如下图：



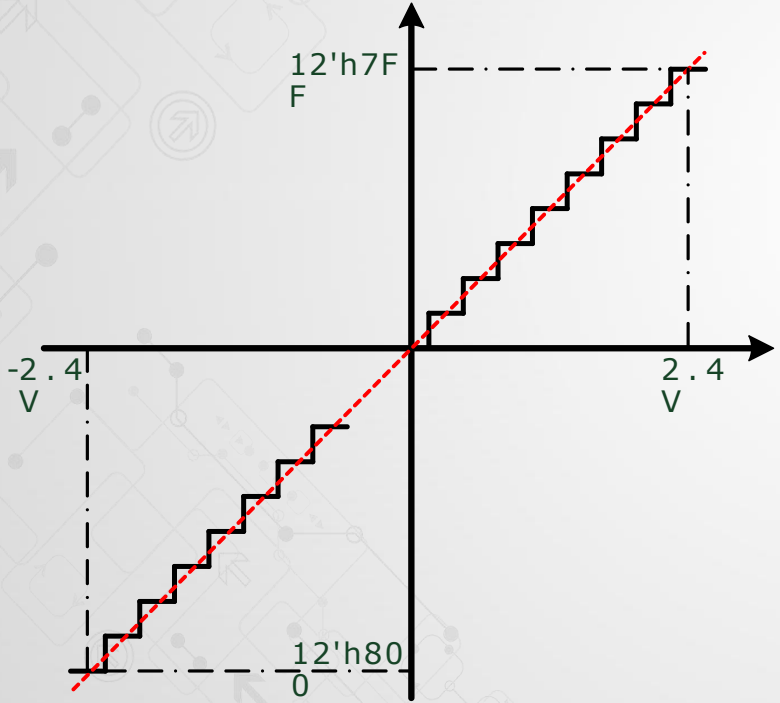
- **连续采样模式**如右图：
  连续采样模式的进入与退出通过ADC_TRIG[14]控制
  连续采样模式下ADC完成一轮采样后立刻开始下一轮采样
  因此通常CPU是来不及响应采样完成中断的，可以配合ADC_DAT0的阈值中断使用



03

ADC数据格式

- ADC输出数据为12bit补码。请注意：ADC采集差分输入信号，正负端电压的差值有正有负，所以ADC结果有正负号，单端输入信号范围-0.3V ~ 3.6V, GPIO 复用口输入的信号范围只能在-0.3V~AVDD+0.3V 之间。
- ADC接口数据寄存器为16bit，可选择左对齐或右对齐，由B[0]@ADC_CFG控制



一倍增益输入模拟量数值/V	2.4	0	-2.4
2/3倍增益输入模拟量数值/V	3.6	0	-3.6
转换后的数字量	12’h7FF	12’h000	12’h800
数据寄存器存储值(左对齐)	16’h7FF0	16’h0000	16’h8000
数据寄存器存储值(右对齐)	16’h07FF	16’h0000	16’hF800

03 ADC基准电压与量程

- **ADC 有两种基准电压：**ADC 有 2.4V/1.2V 两种基准电压。
- **ADC 有两种增益模式：**高增益（1 倍）和低增益（2/3 倍）。针对这两种增益，ADC 的量程也相应有所区别。1 倍增益模式下，对应最大±2.4V 的输入信号幅度，2/3 倍增益模式下，对应最大±3.6V的输入信号幅度。
- 使用外部输入电源作为 ADC REF，可设置ADC量程为4.85V。ADC量程有1.2V/3V/4.85V三种可选。

基准电压 / 增益	1.2V	2.4V
1倍	ADC量程为1.2V	ADC量程为2.4V
2/3倍	ADC量程为3.6V	ADC量程为3.6V

03

ADC校正

- ADC 硬件接口模块可以进行直流偏置校正与增益校正。

- 芯片出厂时已经过工厂标定，标定数据存放在 NVR 中，芯片上电会自动加载。芯片上电会自动加载。ADC 模块在初始化的时候，需要根据数据左右对齐模式配置 DC offset, 可以参看芯片供应商提供的库函数。

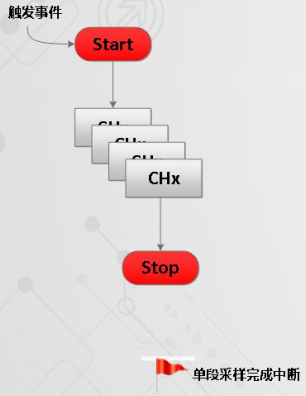
- ADC 有高增益和低增益两档配置，两种配置对应两套校正参数，每套校正数据分别包含一个 DC offset(以下记为 DCoffset)和一个增益校正值 AMPcorrection。同时每套校正参数包含两组 DC/AMC 分别对应采样电路 a/b。高增益对应的校正系数为 ADC_DC_A1/ADC_AMC_A1 和ADC_DC_B1/ADC_AMC_B1，低增益对应的校正系数为 ADC_DC_A0/ADC_AMC_A0 和
- ADC_DC_B0/ADC_AMC_B0。

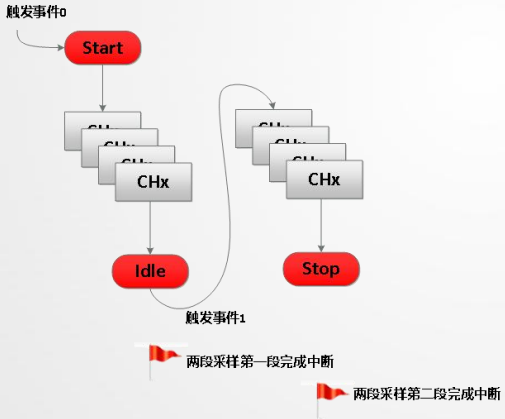
- 记 ADC 输出的数字量为 D_{ADC}，D_{ADC} 对应的真实值为 D，D_0 为编码数制的 0，则
- $D = (D_{ADC}-D_0-DC_{offset})*AMP_{correction}$

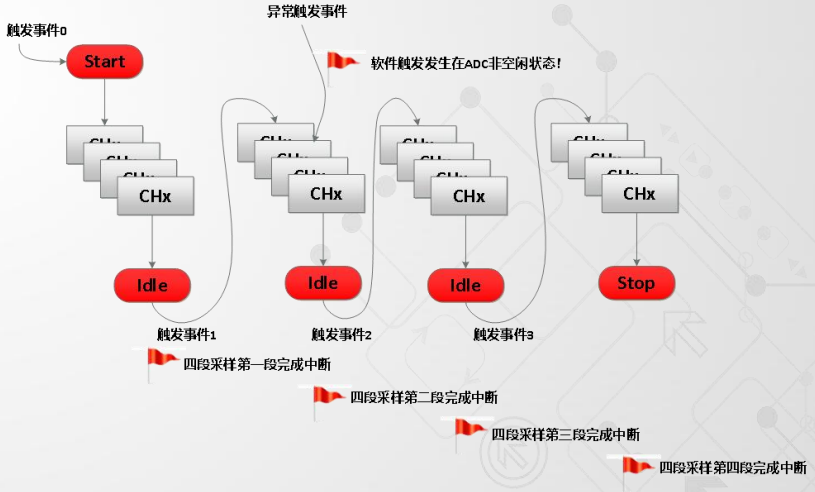
- 最终硬件会将进行校正后的 D 存入相应的采样数据寄存器。ADC 接口硬件电路会根据每个通道的增益配置(ADC_GAIN0/1)来自动选择 $AMP_{correction}$ 与 DC_{offset}。

03 ADC中断标志

- ADC中断信号在每段采样完成后置位
- 软件触发如果在ADC工作时发生，则置位异常触发中断
- ADC_DAT0具备阈值中断，阈值可以设置为上阈值/下阈值，通过ADC_CFG[1]设置，当ADC_DAT0超过ADC_DAT0_TH时发生中断

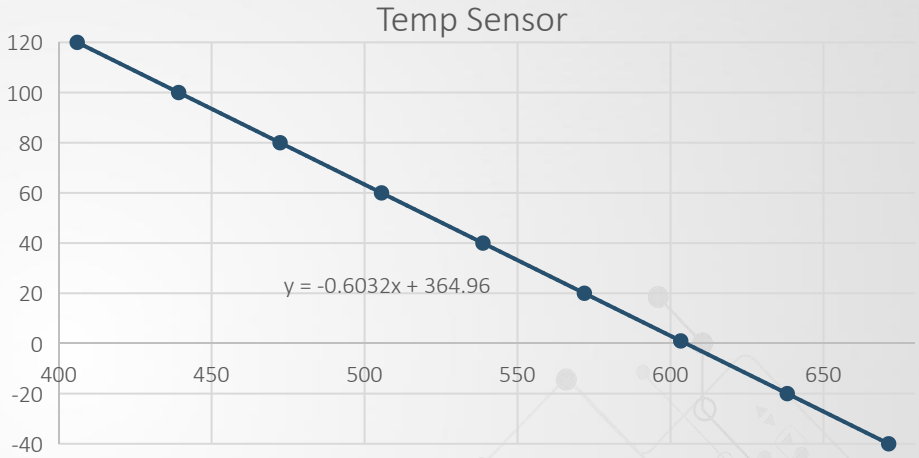






03 温度传感器

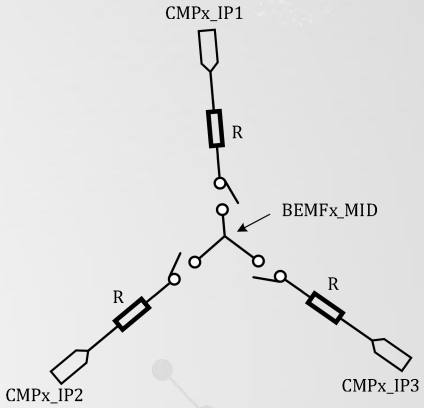
- 芯片内置温度传感器，-40~85℃范围内最大偏差为2℃。85~105 ℃范围内最大偏差为3℃。
- 芯片出厂前会经温度校正，校正值保存在flash info区。
- 芯片上电的默认状态下，温度传感器模块是关闭的。开启传感器之前，需要先开启BGP模块。
- 温度传感器通过设置TMPPDN=1打开，开启到稳定需要约2us，因此需在ADC测量传感器之前2us打开。
- 图中 X 轴为温度传感器的温度信号所对应的 ADC 值，Y 轴为传感器所处的温度。测温时，按照如上要求配置传感器相关寄存器，并得到 ADC 值后，将 ADC 值作为 X 代入公式:y=-0.6032x+364.96求得的 Y 值即为此时的温度。

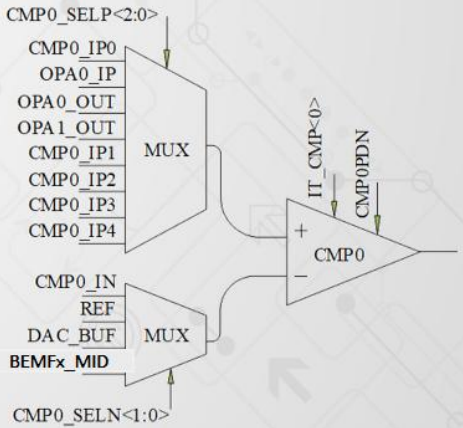


- 存储时，会将 b 系数小数点右移一位（乘 10）存入 info 区，小数点后第二位不进行保存。
- 将 a 系数小数点右移四位（乘 10000）存入 info 区。
- 上述计算公式，基于 ADC 右对齐实现。

03 比较器

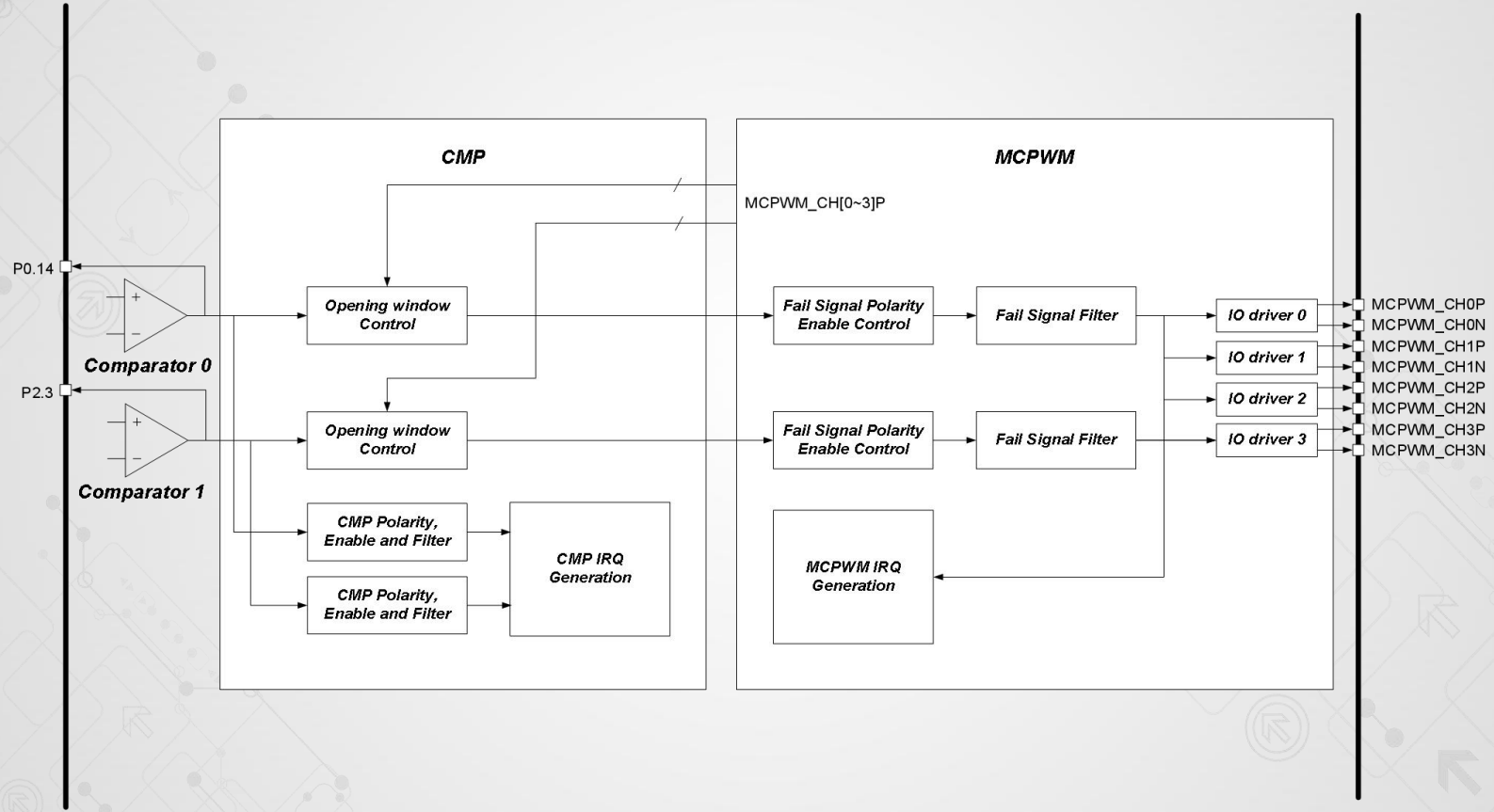
- 内置 2 路输入轨到轨（rail-to-rail） 比较器，比较器比较速度可编程、迟滞电压可编程、信号源可编程。
- 比较器的比较延时可通过寄存器 IT_CMP 设置为 0.15uS/0.6uS。迟滞电压通过 CMP_HYS 设置为20mV/0mV。
- 比较器正负两个输入端的信号来源都可通过寄存器 CMPx_SELP[2:0]和 CMPx_SELN[1:0]进行设置（x=0/1，代表 CMP0/CMP1 两个比较器）
- 两个比较器负输入端的 BEMFx_MID 信号，是对比较器正输入端信号 CMPx_IP1/CMPx_IP2/ CMPx_IP3 信号的平均，具体连接方式见右图。BEMFx_MID主要用于 BLDC 方波模式控制时，虚拟电机相线中心点电压，用于反电势过零点检测。其中电阻 R=8.2k 欧，图中的开关只有在比较器负输入端信号选择为 BEMFx_MID之后才会导通，否则开关都处于断开状态。
- 当CMPx_IP1/ CMPx_IP2/ CMPx_IP3 管脚连的是 HALL 信号时，通过将 HALL 信号与 BEMFx_MID信号进行比较，可快速得到 HALL 信号的状态 ，进行过零点检测。
- 模拟比较器未经滤波的原始输出值也可以通过配置 GPIO 的第二功能通过 P0.14 和 P2.3 送出。





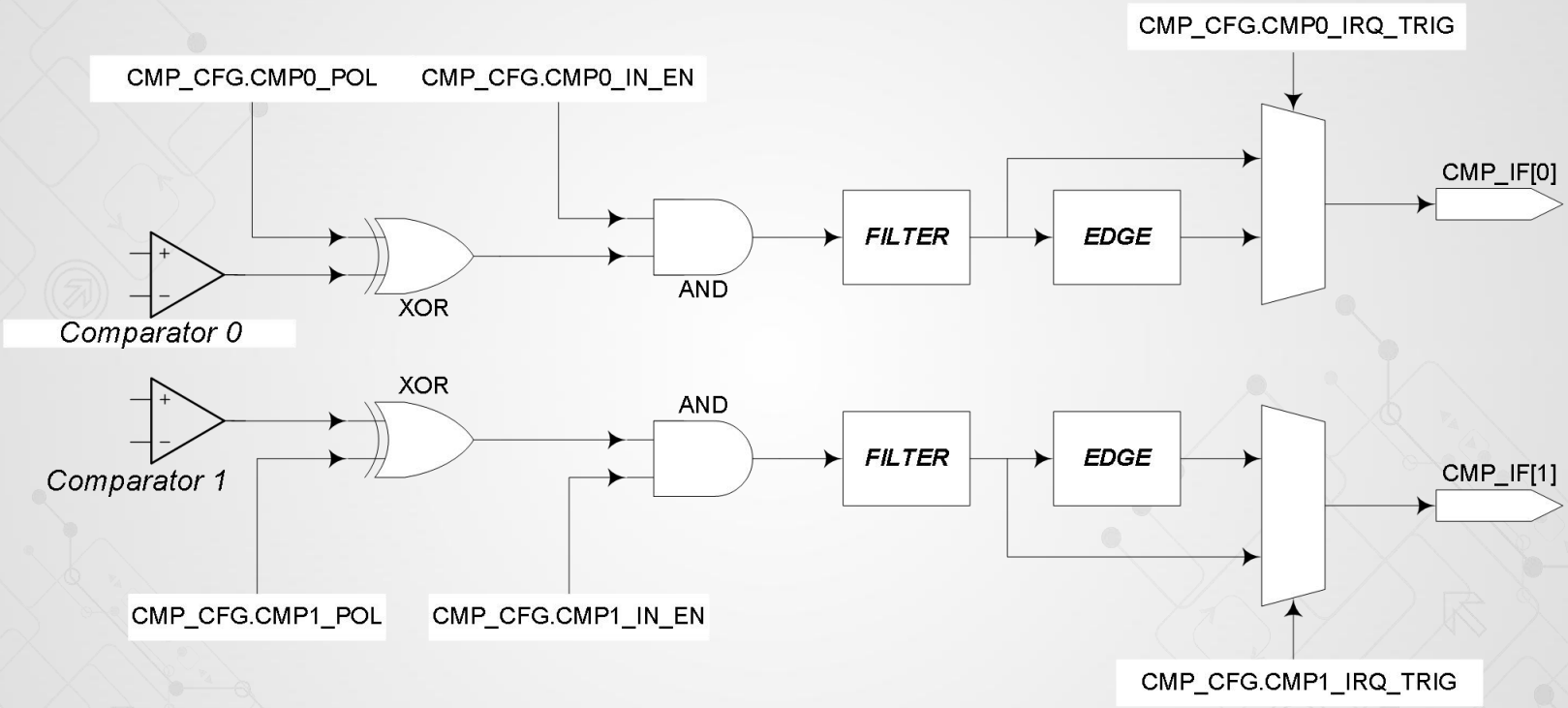
03

CMP与MCPWM 的联动



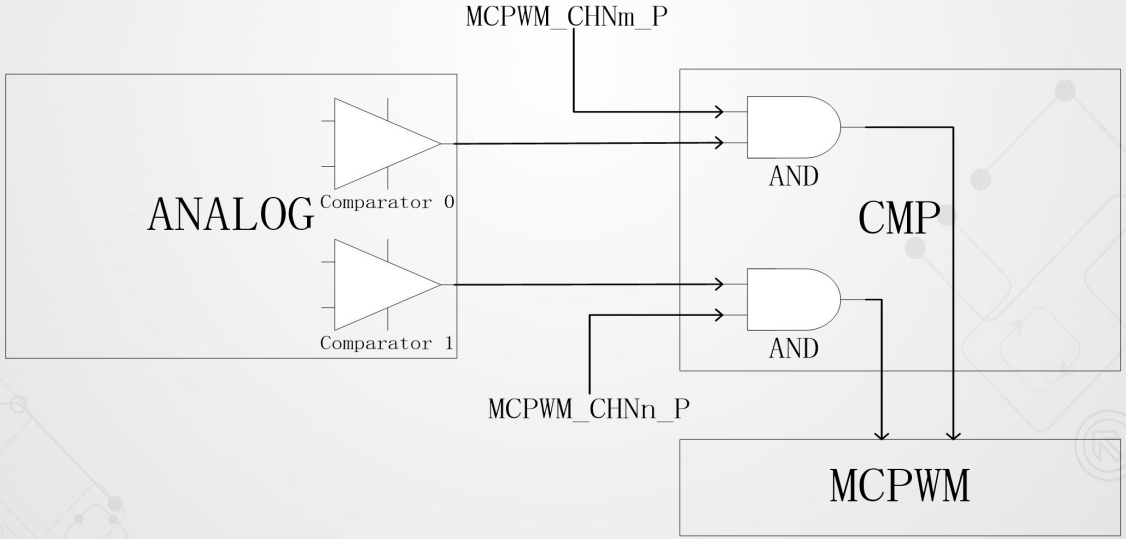
03

比较器控制及中断产生逻辑



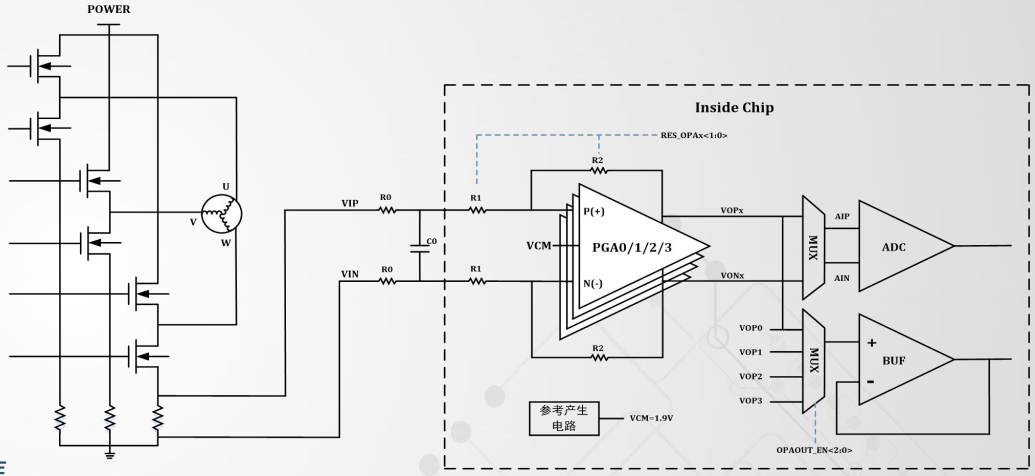
03 比较器开窗

- 对于比较器的开窗功能，若 CMP_CFG.CMP0_PWM_POL=1，则在对应 MCPWM CHNx_P 信号为 1时，比较器 0 可以产生比较信号输出，其他时刻比较信号为 0；
- 反之，若 CMP_CFG.CMP0_PWM_POL=0，则在对应 MCPWM CHNx_P 信号为 0 时，比较器 0 可以产生比较信号输出，其他时刻比较信号为 0。比较器 1 的开窗控制信号极性由 CMP_CFG.CMP1_PWM_POL 位进行控制，逻辑相同。
- 注意：CMP_CFG.CMP0_PWM_POL 和CMP_CFG.CMP1_PWM_POL 同时会影响送入 MCPWM 模块作为 fail 信号的比较器信号。



03 运放（OPA）

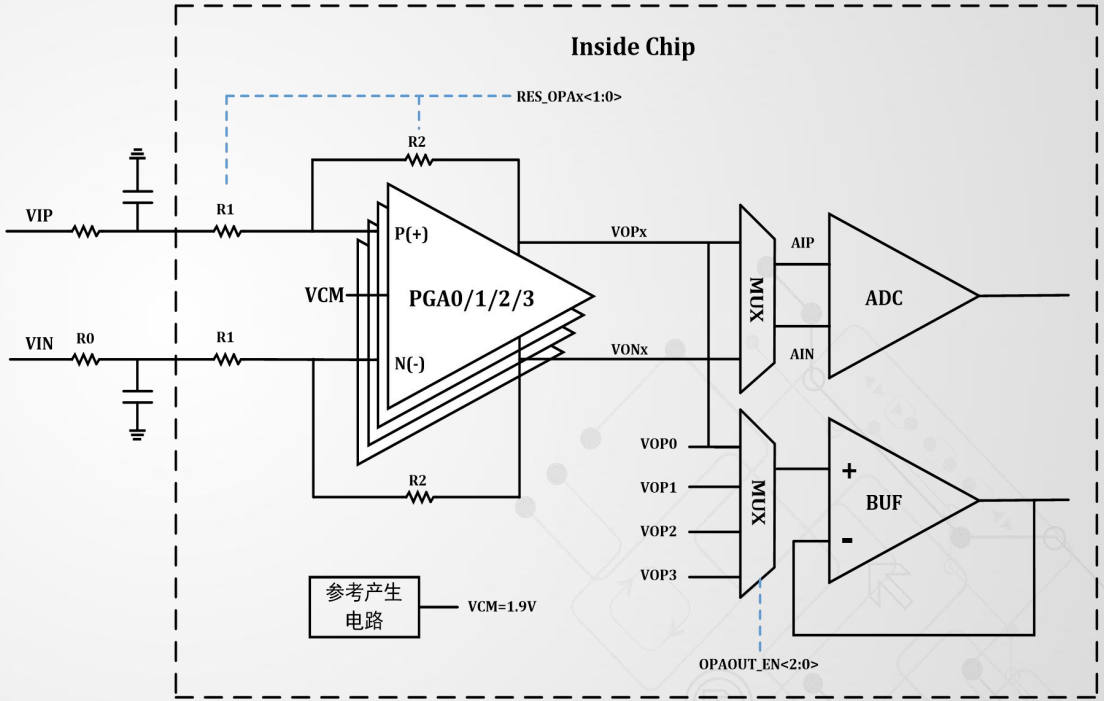
- 芯片集成 4 路输入输出轨到轨（rail-to-rail）运算放大器，内置反馈电阻，外部引脚上需串联一个电阻 R0 到信号源。反馈电阻 R2:R1 的阻值可通过寄存器 RES_OPAx[1:0]设置，以实现不同的放大倍数。
- 图中两个 R0 是片外需放置的电阻，阻值必须相等，最终的放大倍数为 R2/(R1+R0)。
- 对于 MOS 管电阻直接采样的应用，由于 MOS 下管关断、上管导通时信号会升高到数十 V 的电源电压，为减小此时往芯片引脚里流入的电流，建议接>20kΩ 的外部电阻。
- 对于康铜丝等取样电阻采样的应用，建议接 100~1K 欧的外部电阻。C0 为信号滤波电容，和 R0形成一阶RC 滤波电路。R0 的具体阻值可根据 R0*C0 的滤波常数而定。如果信号上噪声较小不需要滤波、或者信号需要很大的带宽（较快的响应速度），则 C0可以不加。
- 运放输入正负端内置钳位二极管，电机相线通过一个匹配电阻后直接接入输入端，从而简化了MOSFET 电流采样的外置电路。



放大器框图

03 运放差分和单端工作模式的区别

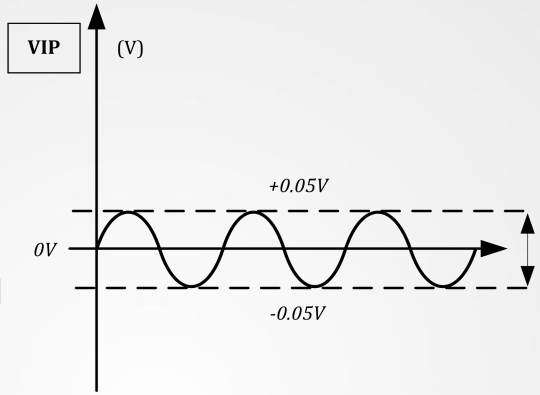
- 如果输入信号 VIP/VIN 之间的共模噪声特别大，则可将 1 个跨接电容 C0 改 为 2 个电容到地的电容。
- 放大器可通过设置 OPAOUT_EN<2:0>选择将 4 路放大器中某一路的正端输出信号 VOP 通过 BUFFER 送至 P2.7 IO 口进行测量和应用。
- 因为有 BUFFER 存在，在运放正常工作模式下也可以选择将其VOP 信号出来，此时该运放相当于同时作为差分运放和单端运放使用。
- 注意：BUFFER 的输出与ADC读取的不一样，ADC读取的是差分信号，而BUFFER 输出是单端信号。

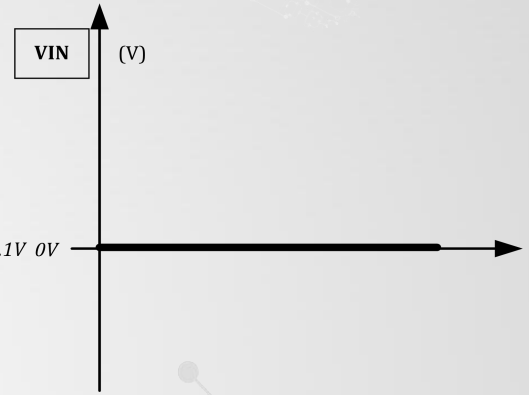


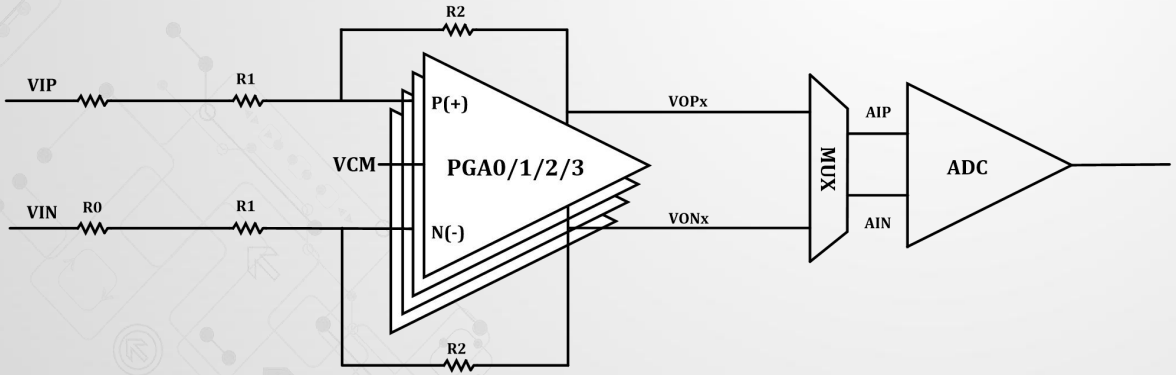
03 差分工作模式

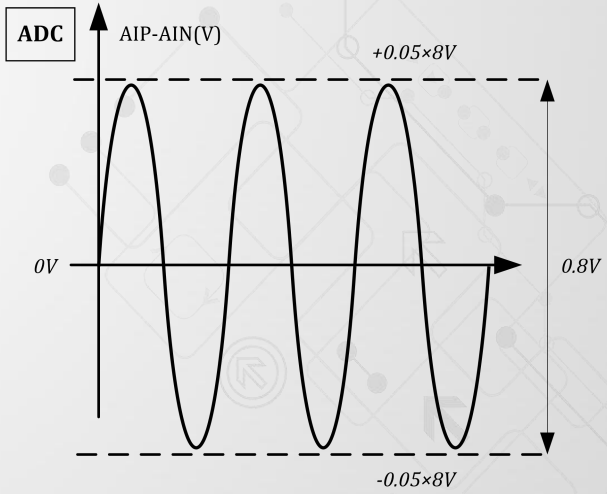
- 运放输出的差分信号：
- Vsig = VOPx - VONx = Gain * Vin +Gain * Voffset
  其中: Vin=VIP-VIN；
  Gain = R2/(R1+R0);
  Gain * Voffset 为运放经过放大后的零漂，对于应用来说 Gain * Voffset 这项需要剔除。

- 运放输出信号送给 ADC 采样的方式，是属于差分工作模式。差分工作模式的优点是精度高、抗干扰能力强，差分模式是芯片的默认工作模式。



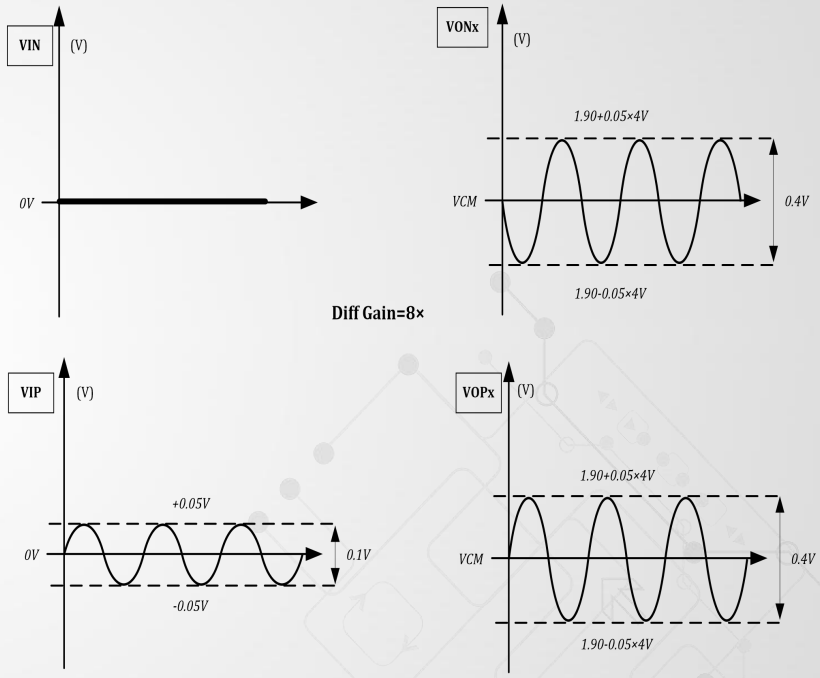






03 单端工作模式

- 除差分模式之外，运放还有两种单端应用模式：
- 将某路运放的正输出端 VOP，通过设置 CMPx_SELP 送至比较器 0/1 的正输入端，比较器的负输入端通过设置 CMPx_SELN<1:0>(x=0/1)连至DAC 模块的输出，这种应用方式，可将放大后的信号做过流保护使用。
- 通过设置 OPAOUT_EN<2:0>将 4 路放大器中某一路的正端输出信号(VOP)通过 BUFFER 送至 P2.7 IO 口，控制芯片外部的某些模块，或者将该信号在芯片外做滤波后得到信号平均值，重新送到芯片的另一个 ADC输入通道管脚，由ADC 去测量滤波后的信号。
- VOP/VON 的信号公式为：
  VOP = Vcm + Vsig/2
  VON = Vcm - Vsig/2
  其中,Vcm 是运放输出的共模电压，一般为 1.9V，Vsig 为运放输入信号经放大后的差分信号。



03

单端模式Vcm的校正

- 对于不同芯片运放的 Vcm，有离散性，大批量时，相比于 1.9V 的平均值，最大偏差可能达到一百多毫伏。因此有必要对 Vcm 做校正。
- 1）在芯片上电之初，电机未运行时，运放输入信号为 0V，此时将作为单端模式应用的那路运放 VOP 通过配置 OPAOUT_EN<2:0>送至 P2.7 IO 口，IO 口是ADC_CH11，用 ADC 对 CH11 进行采样，采样得到的值即为 Vcm 对应的 ADC 值。实际的物理值(即多少 V)则可根据 ADC 值计算得到。
- 注意如果 P2.7 IO 口在 PCB 上有阻容滤波，需要等待一段时间稳定才能采样。
- 2）上述部分校正了不同芯片之间运放共模电平 Vcm 的差异，但 Vcm 随温度还有约-0.5mV/℃的变化。即 125 度相比 25 度，Vcm 将下降 50mV。如果应用上要求精度较高，则有必要修正 Vcm 的温度变化量。
- 在芯片上电之初，将运放 VOP 送至 P2.7 IO 口进行 ADC 测量得到 Vcm0 的同时，测量芯片内部温度传感器的温度（详见 usermanual 温度传感器章节）T0。后面电机运行过程中，每隔几秒或者 1 分钟，测量一次温度值 T1，计算温度差VT = T1 - T0。此时的运放共模电压 Vcm = Vcm0 + VT * (-0.5m)。
- 但即使做过温度修正，运放单端模式的精度仍然不如差分模式，对于需要精确测量小电流信号的场合，建议还是使用单独一个运放的差分模式。

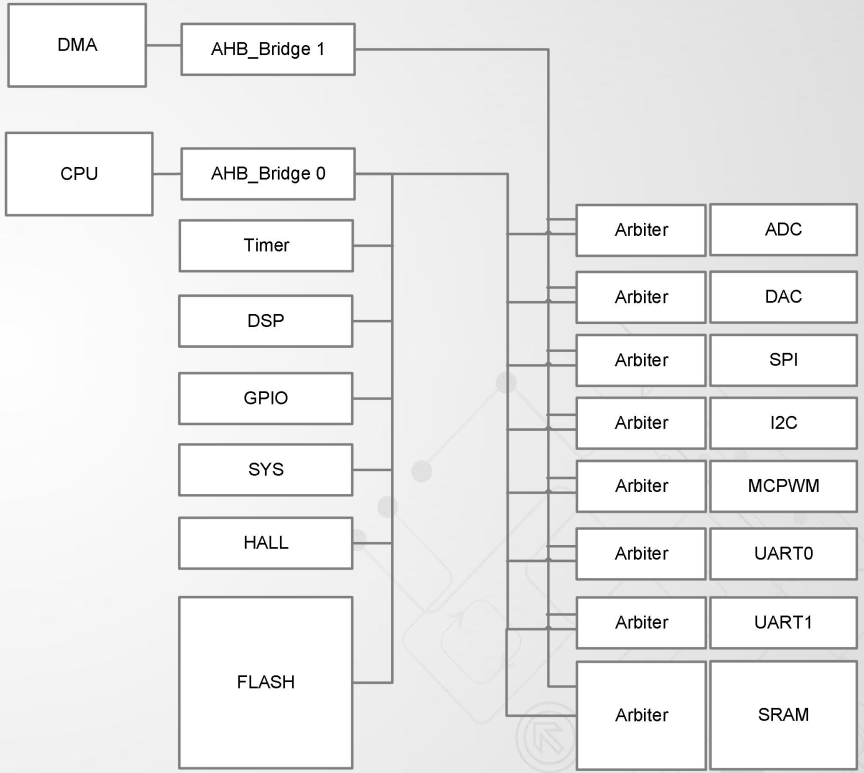
ADC_CH11/OPAx_OUT/LDO15/P2.7

03 DAC

- 08x芯片内置一路 12bit DAC，输出信号的最大量程可通过寄存器 DAC_GAIN<1:0>设置08x为1.2V/3V/4.85V。
- DAC可通过配置寄存器 DACOUT_EN=1，将 DAC 输出送至 P0.0 管脚，可驱动>5kΩ 的负载电阻和50pF 的负载电容。
- DAC 最大输出码率为 1MHz。
- 08x_DAC 的输入数字信号寄存器为 SYS_AFE_DAC，低 12BIT 有效。信号范围是 0x000~0xFFF。0x00 对 应零模拟量输出 0V，0xFFF 对应满量程模拟量输出为DACfs。每一档信号(LSB)所对应的模拟信号幅度为DACfs/4096。若 SYS_AFE_DAC 的数字值为 Din，则该数字信号所对应的 DAC 输出模拟信号为（DACfs/4096）* Din。
- DAC 自带校准硬件模块。DAC 输出校正遵循公式 y=ax+b。a,b校正值在芯片出厂前以存入芯片NVR区域，具体校正数据存储地址请查看芯片使用书册DAC章节，将不同量程的校正值从NVR区域读取出后写入SYS_AFE_DAC_AMC（a），SYS_AFE_DAC_DC(b)校正寄存器即可实现DAC输出校正。
- DAC 输出的模拟信号，除了可以送至 IO 口供外部模块使用外，还可通过配置寄存器连至芯片内部的 2 路比较器负端，作为比较器的基准信号使用。

03 DMA

- ADC/DAC/SPI/I2C/MCPWM/UART/SRAM等设备可以被DMA访问到。当CPU与DMA同时访问某外设时，CPU优先级更高
- 4个DMA通道时分复用一个DMA引擎，每个通道有3个硬件DMA请求和一个软件请求。
- 支持Byte/Half word/Word等位宽的传输，支持Source与Destination位宽不同。
- 各通道可配置进行单轮多次或多轮每轮多次搬运。
- DMA无法访问右图Timer,DSP,GPIO,SYS, HALL , FLASH模块。



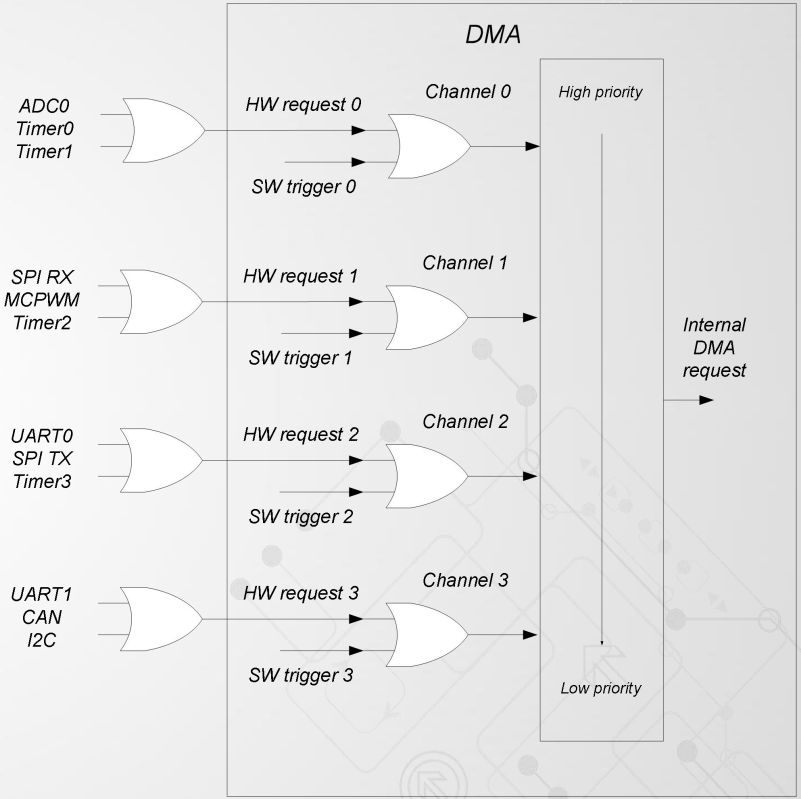
03 DMA请求及优先级

- 通道间优先级：

  Channel 0 > Channel 1 > Channel 2 > Channel 3

- 通道内优先级：

  在 DMA 各通道内部，通常有 3 个硬件请求事件和一个软件请求事件，硬件请求优先级高于软件请求。3 个硬件请求事件优先级相同，通常应用上面一个 DMA 通道配置一个硬件请求事件使能，多个硬件请求不应在一个通道内同时发生。

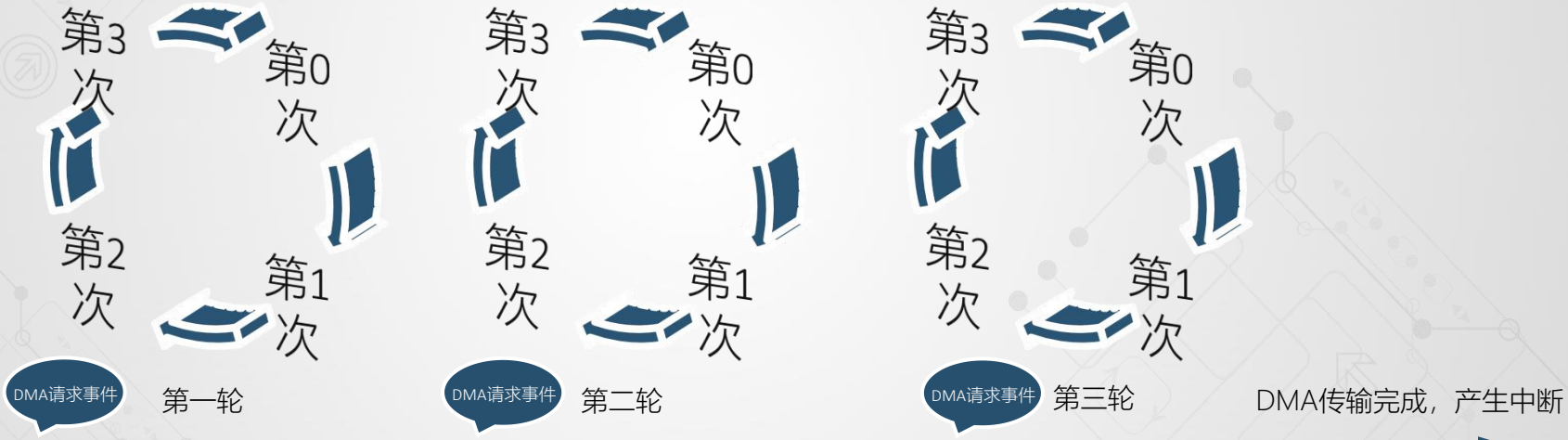


03 DMA通道搬运次数

- DMA_CTMS用于控制DMA通道的采样次数和轮数

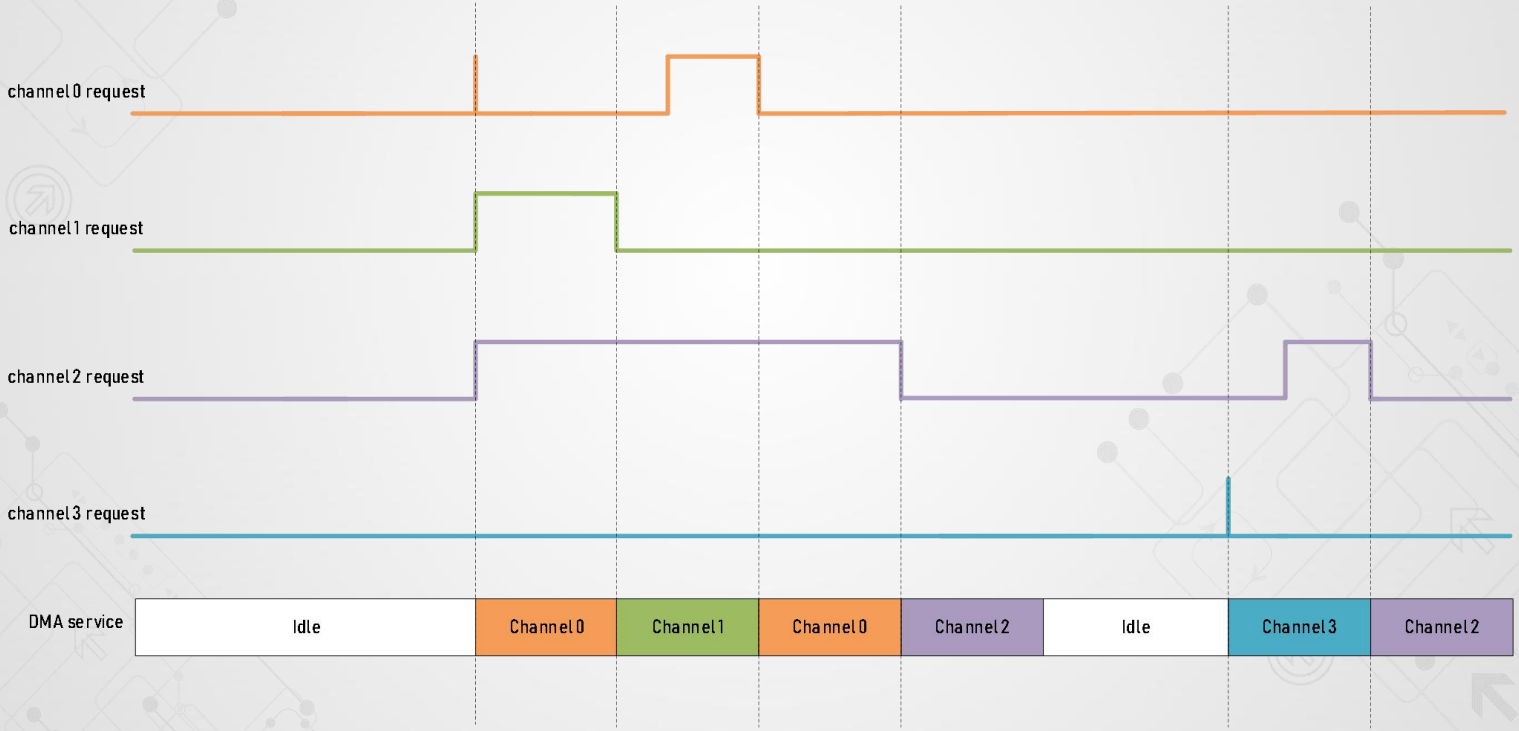
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								采样轮数																每轮采样次数							

以下，以DMA_CTMS[8:0] = 4， DMA_CTMS[23:16] = 3 为例。注意多轮采样，每轮触发仍需要DMA请求信号



03 DMA请求及优先级 | 通道间仲裁

- 当多个通道请求同时发生时，序号低的通道优先级高，首先得到DMA服务。需要注意的是，当DMA被占用时，即使有高优先级请求，也不会立即得到服务，须等到DMA引擎空闲。此外，未得到响应的请求会一直存在（request信号一直为高）直到收到ACK。



03 DMA传输地址

- DMA_CPAR[16:0]：

只存储地址的低17位，高15位恒为0x2000，高20位可能为0x40000(对应SYS寄存器)或0x4001*(对应外设寄存器)

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

- DMA_CMAR[12:0]：

DMA_CMAR只存储地址的低13位，对应SRAM 8kB地址空间。高19位恒为0x10000，对应RAM地址

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

03

DMA传输地址

- DMA_CCR[4]：DIR

  用于控制传输方向是外设到内存（P2M）还是内存到外设（M2P）

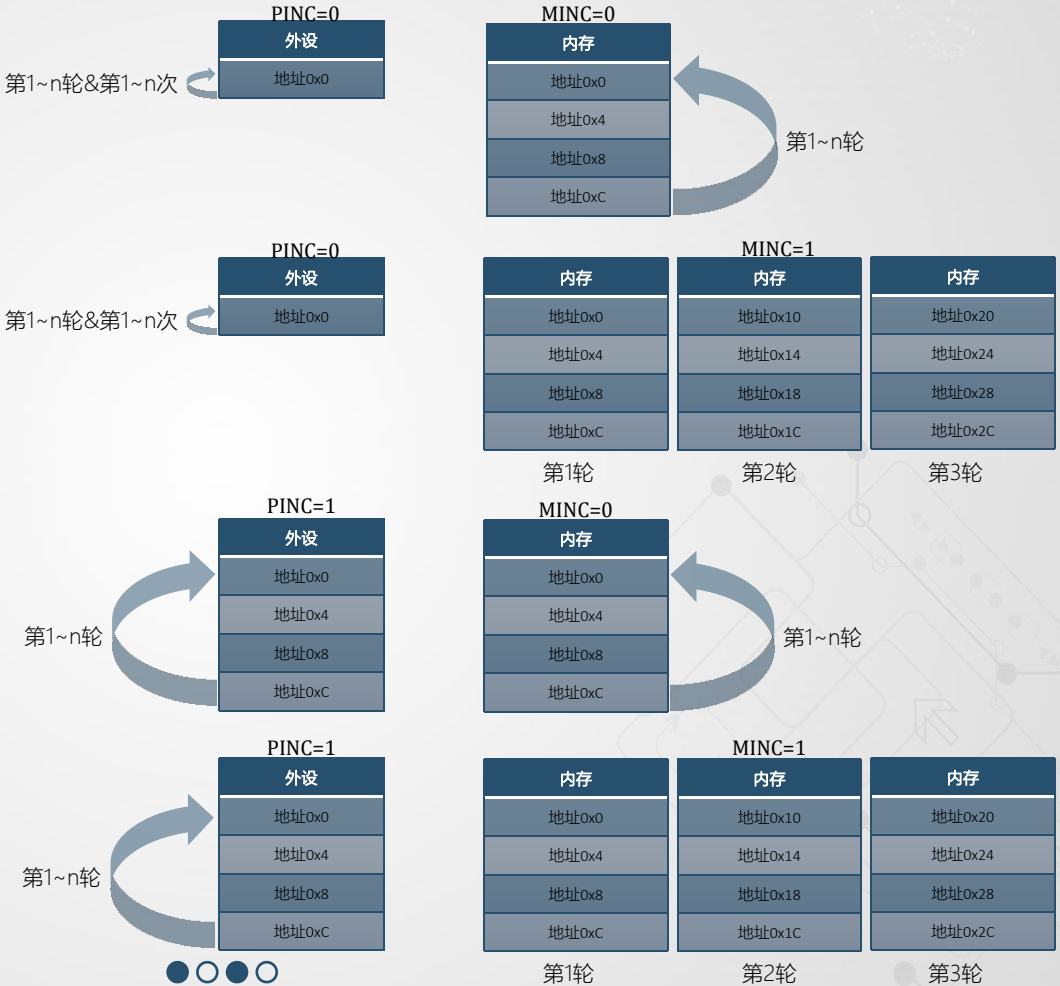
- DMA_CCR[6]：PINC

  用于控制外设地址每次传输是否递增，外设地址每轮传输一定不递增。

- DMA_CCR[7]：MINC

  用于控制内存地址第二轮是否在第一轮地址的基础上递增（轮内一定递增）。

右侧以采样4次，n轮为例，说明地址递增机制



03 DMA中断

- **注意：**DMA 的一个通道完成 DMA 操作后或出错则产生 DMA 中断。当 DMA 某一个通道完成操作后，会自动关闭该通道的使能为 **DMA_CCRx.EN**。下一次进行DMA传输时需要软件重新使能对应通道（ DMA_CCRx.EN），才可继续进行DMA传输。

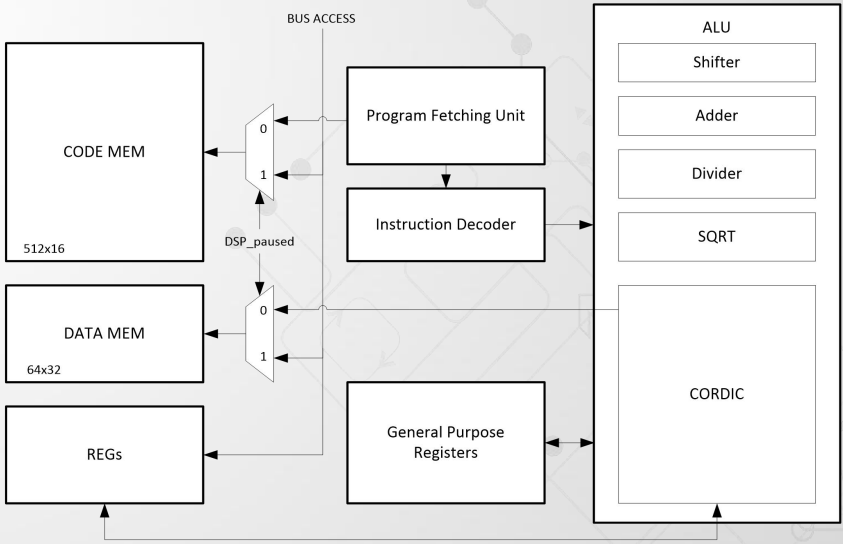
03 08x_DSP

- DSP模块使用自主设计的DSP指令集，可以进行加法、乘累加、移位、饱和等单周期算术指令，以及除法、开方、三角函数等多周期算术运算指令；具备load/store等访存指令，无条件跳转以及条件跳转等分支指令，还有中断提起等杂项指令。有断点指令和寄存器赋值等伪指令可以在模拟器上用于调试。

- 为充分灵活利用 DSP，在 DSP 暂停时允许 CPU 通过 DSP 寄存器接口直接访问 DSP 除法器、开方器、三角函数等运算模块，**即允许 CPU 将 DSP 当做简单的运算协处理模块使用。**

- 除法10个总线周期（96MHz）完成
- 开方8个总线周期（96MHz）完成
- 三角函数指令需要 8 个总线周期（96MHz）完成。
- 其余指令均为单周期指令

- 除法的被除数和商位宽均为 32 位有符号数，除数和余数为 16 位有符号数。
- 被开方数为 32 位无符号数，平方根为 16 位无符号数。
- 乘累加的两个乘数均为 16 位有符号数，加数和结果为 32 位有符号数。
- 三角函数 CORDIC 模块位宽为 16 位，Q15 定点数格式。

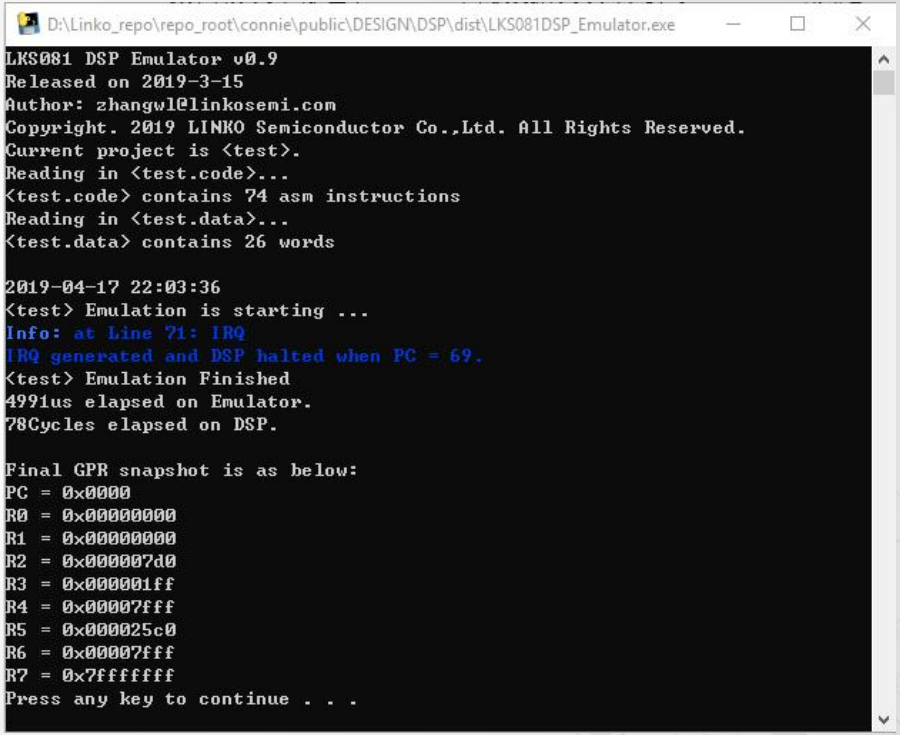


08x_DSP | ISA

Operation	Description	Assembler		Cycles
Add		ADD	Rd1 Rs1 Rs2	1
	5bit Immediate	ADDI	Rd1 Rs1 #<Imm>*1	1
Subtract		SUB	Rd1 Rs1 Rs2	1
Shift	Arithmetic right shift	ASR	Rd1 Rs1 Rs2	1
	5bit Immediate	ASRI	Rd1 Rs1 #<Imm>	1
	Logical left shift	LSL	Rd1 Rs1 Rs2	1
	5bit Immediate	LSLI	Rd1 Rs1 #<Imm>	1
Multiply and accumulation		MAC	Rs1 Rs2 Rs3	1
	5bit Immediate	MACI	Rs1 Rs2 #<Imm>	1
Divide		DIV	Rd1 Rs1 Rs2	10
Saturation		SAT	Rd1 Rs1 Rs2	1
	4bit Immediate	SATI	Rd1 #<Imm1> #<Imm2>	1
Cordic	SIN/COS	SIN_COS	Rd1 Rd2 Rs1	8
	Arctan/Module	ARCTAN	Rd1 Rs1 Rs2	8
Square root	Square root	SQRT	Rd1 Rs1	8
Memory access	Load word	LDRWI	Rd1 #<Imm>	1
	Load double half words	LDRDHI	Rd1 Rd2 #<Imm>	1
	Store word	STRWI	Rs1 #<Imm>	1
	Store double half words	STRDHI	Rs1 Rs2 #<Imm>	1
Branch	Unconditional Jump	JUMP	Rs1	2
	Immediate	JUMPI	#<Imm>	2
	Jump if less than or equal to	JLE	Rs1 Rs2 Rs3	2
	Immediate	JLEI	Rs1 Rs2 #<Imm>	2
Miscellaneous	Generate IRQ and Pause DSP	IRQ		1

03 08x_DSP | Emulator on Windows

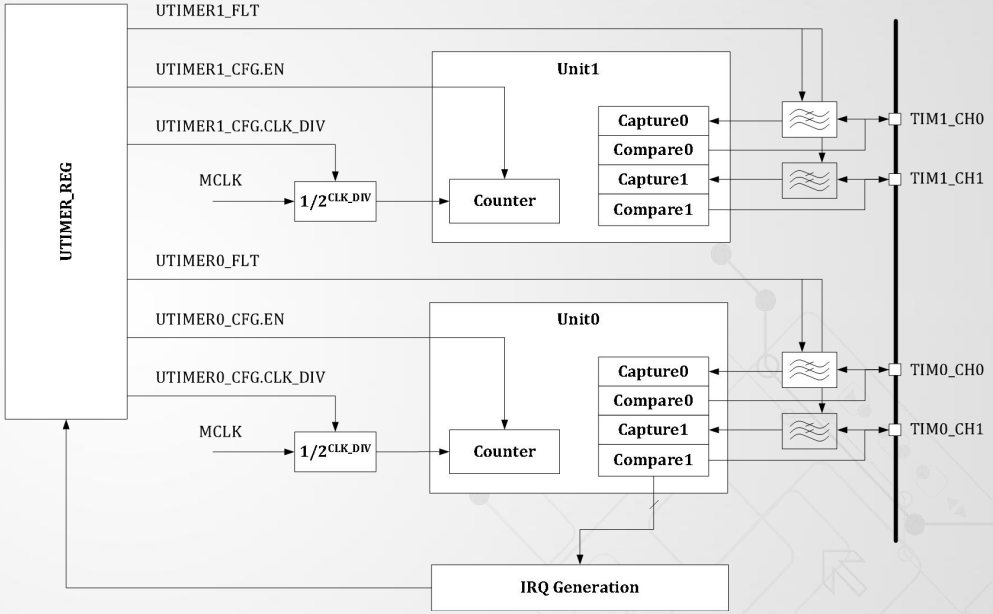
- 支持DSP汇编指令模拟
- 支持断点指令
- 支持寄存器运行中改写
- 支持核心寄存值打印
- 支持文件访问
- 支持指令trace
- 支持寄存器值trace



03

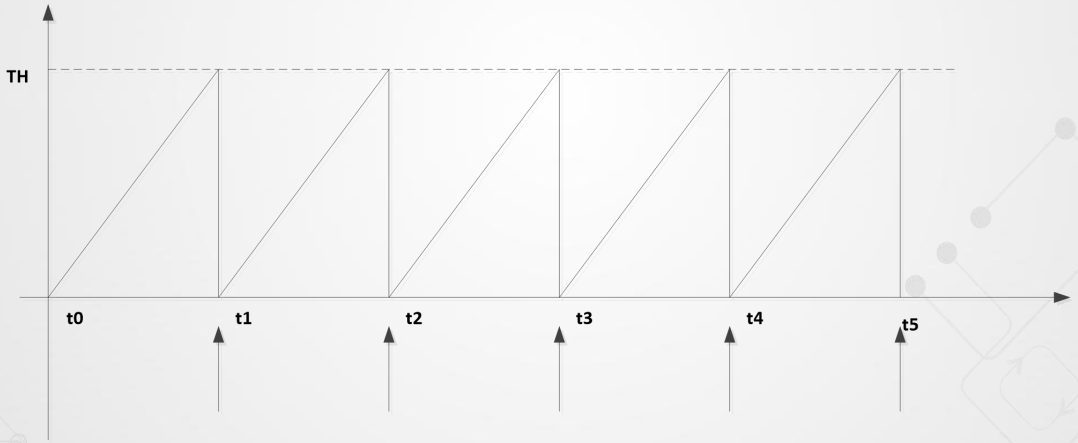
Timer(定时器)

- 1个SysTick定时器，24位。
- 08x通用定时器共4路，两路16bit，两路32bit；独立工作，可工作在不同频率下。
- 每个通用定时器处理2个外部输入信号（捕获模式），或者产生2个输出信号（比较模式，可用于产生边沿对齐 PWM/定时中断。
- 每个Timer有两个输入信号，可以进行最大120个系统主时钟的滤波，当芯片工作在96MHz时钟频率下时，可以滤除1.25uS宽度一下毛刺。



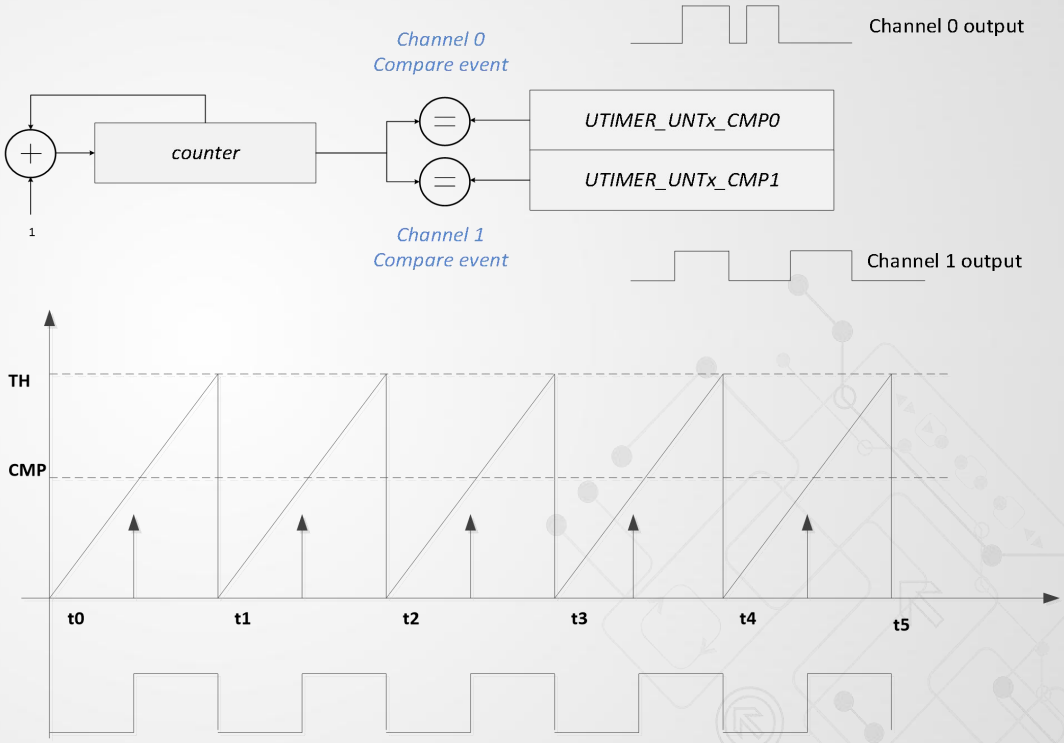
03 Timer工作模式

- Timer中的计数器采用向上计数模式，周期为TH+1。
- 计数器从0计数到TH值，再回到0重新开始计数，计数器回到0时，产生回零中断。



03 Timer工作模式 | 比较模式

- 比较模式下，计数器计数到CMP值时，产生比较中断。
- 比较模式可以驱动一个比较脉冲，用来产生PWM。
- 在计数器回零时，输出一个电平（可配置极性），在比较事件发生时，电平翻转。
- 比较模式下，可以产生两路定时事件，并可以产生中断。



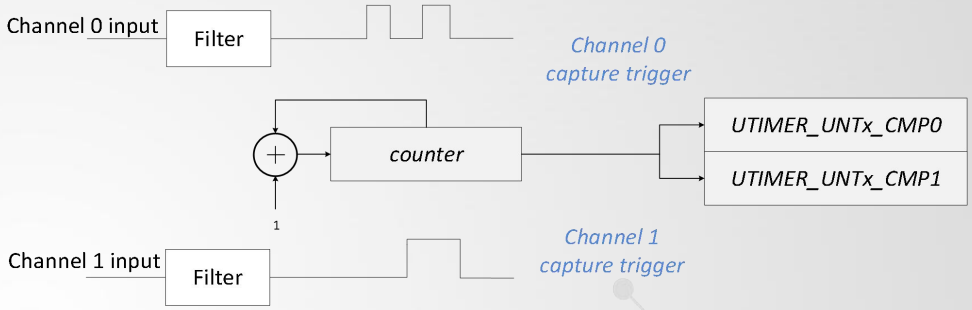
03 Timer工作模式 | 比较模式

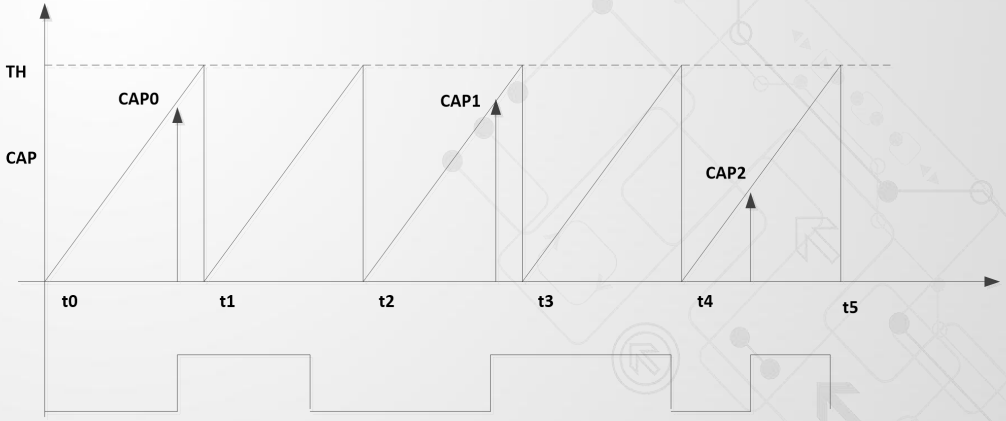
- 如果要实现某个 Timer 通道 0 输出全 0：
- 1）可以设置UTIMER_UNTx_CFG.CH0_POL=0
- 2）并设置 UTIMER_UNTx_CMP0=UTIMER_UNTx_TH+1。即 Timer 计数值回 0 时，通道输出 0，且Timer 计数值全程不命中 CMP0。或设置 UTIMER_UNTx_CMP0=0，即 Timer 命中 CMP0 和回零事件为相同事件。

- 如果要实现某个 Timer 通道 0 输出全 1：
- 1）可以设置 UTIMER_UNTx_CFG.CH0_POL=1
- 2）并设置 UTIMER_UNTx_CMP0=UTIMER_UNTx_TH+1。即 Timer 计数值回 0 时，通道输出 1，且Timer 计数值全程不命中 CMP0。

03 Timer工作模式 | 捕获模式

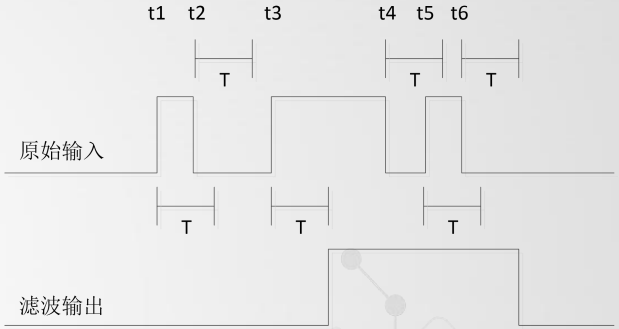
- 捕获模式下，可以捕获输入信号的上升/下降或者双沿，发生捕获事件时，定时器计数值存入CMP寄存器，并产生捕获中断。
- 如右图所示，定时器设置为上升沿捕获。在CAP0/CAP1/CAP2三个时刻点，捕获到输入信号发生上升沿变化，对应时刻点的定时器计数值将存入UTIMER_UNTx_CMP寄存器中。





03 Timer捕获滤波

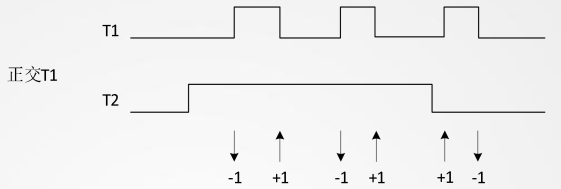
- 定时器模块共有 8 个/4 对通道输入，定时器可以对每个输入进行不同程度的滤波。
- 通过配置滤波寄存器可以调整滤波宽度，0~120 个系统时钟宽度。
- 输入信号滤波始终使用系统高速时钟，通常为 96MHz PLL 时钟，UTIMER_UNTx_CFG.CLK_DIV对 Timer 计数时钟的分频对滤波时钟没有影响。
- 原始输入信号在 t1~t6 几个时刻发生了翻转，滤波器宽度配置成 T。可以看到只有 t3 和 t6 时刻发生的翻转维持了大于 T 的时间，因此从滤波器的输出看，信号仅发生了两次翻转。



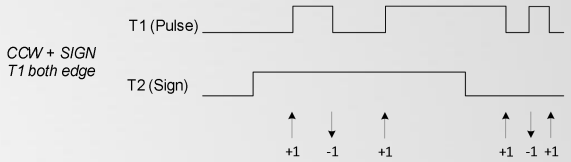
滤波示意图

03 Timer工作模式 | 编码器模式

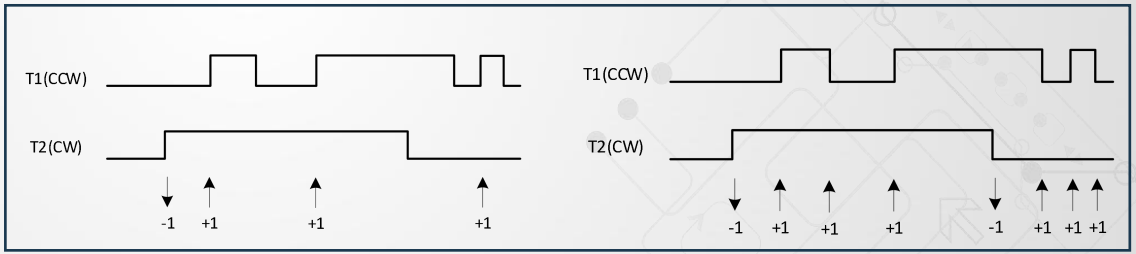
- 支持正交编码信号
- 符号加脉冲信号
- CW/CCW双脉冲信号
- 支持任意模值，达到模值后自动回零计数
- 定时器模块中集成了 2 个编码器模块。
- 其中编码器的输入分别来自 Timer2 的通道 0/1 和 Timer3 的通道 0/1。



正交编码模式



符号加脉冲编码模式



CW/CCW双脉冲编码模式

03 休眠唤醒

- 休眠启动：向SYS_CLK_SLP寄存器写入密码 0xDEAD，系统关闭高速时钟，进入休眠状态。
- 休眠唤醒有两种方案：
- 第一种：定时唤醒，08x唤醒时间只有0.25S、0.5S、1S、2S、4S、8S、16S、32S。
- 第二种：IO唤醒（唤醒电平可选）。唤醒IO请查看芯片数据手册。
- 注意08x的定时唤醒默认是开启状态，无法关闭，如果应用中只需要IO唤醒不需要定时唤醒，软件需要对定时唤醒进行判断，如果当前唤醒是定时唤醒，则继续操作休眠函数进行芯片休眠，具体软件配置参考凌鸥官方提供的模块例程。
- 休眠时PLL时钟需要关闭，当休眠唤醒时需要开启PLL延时100uS后将主时钟切换为PLL时钟。强烈建议使用凌鸥官网提供的Switch2PLL（）;函数,具体说明在Wiki：https://linkosemi.wiki.zoho.com.cn/休眠唤醒及时钟切换注意事项.html

03

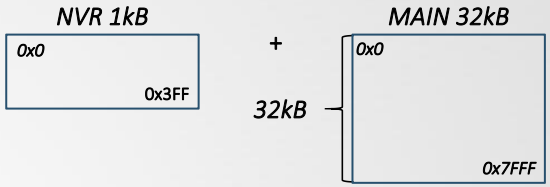
低功耗模式

LKS32MC08x 电路模块电流消耗 IDD

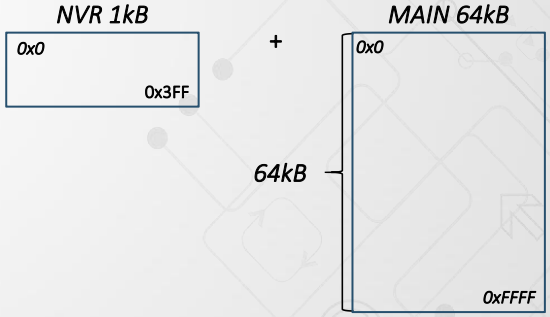
模块	Min	Typ	Max	单位
模拟比较器CMP(1个)		0.005		mA
运算放大器OPA(1个)		0.450		mA
模数转换器ADC		3.710		mA
数模转换器DAC		0.710		mA
温度传感器Temp Sensor		0.150		mA
带隙基准BGP		0.154		mA
4MHz RC时钟		0.105		mA
锁相环PLL		0.080		mA
CPU+flash+SRAM (96MHz)		8.667		mA
CPU+flash+SRAM (12MHz)		1.600		mA
CRC		0.070		mA
DSP		3.421		mA
UART		0.107		mA
DMA		1.340		mA
MCPWM		0.053		mA
TIMER		0.269		mA
SPI		0.500		mA
IIC		0.500		mA
CAN		2.200		mA
休眠	10	30	50	uA

03 FLASH

- FLASH 存储体包含两个部分：NVR 和 MAIN。08x系列NVR 用户使用区大小为 1kB，MAIN 为 32kB 或 64kB（不同型号）。
- 可反复擦除写入不低于 2万次。
- 室温数据保持长达 100 年 。
- 单字节编程时间最长 7.5us，Sector 擦除时间最长 5ms。
- Sector 大小 512 字节，可按 Sector 擦除写入，支持运行时编程，擦写一个 Sector 的同时读取访问另一个 Sector。
- 当对 FLASH 进行读取操作时，需要大于 1 个时钟周期才能完成数据的读出。为了加快数据的读出，FLASH 控制器增加了预取功能。
- Flash 数据防窃取（最后一个 word 须写入非 0xFFFFFFFF 的任意值）。



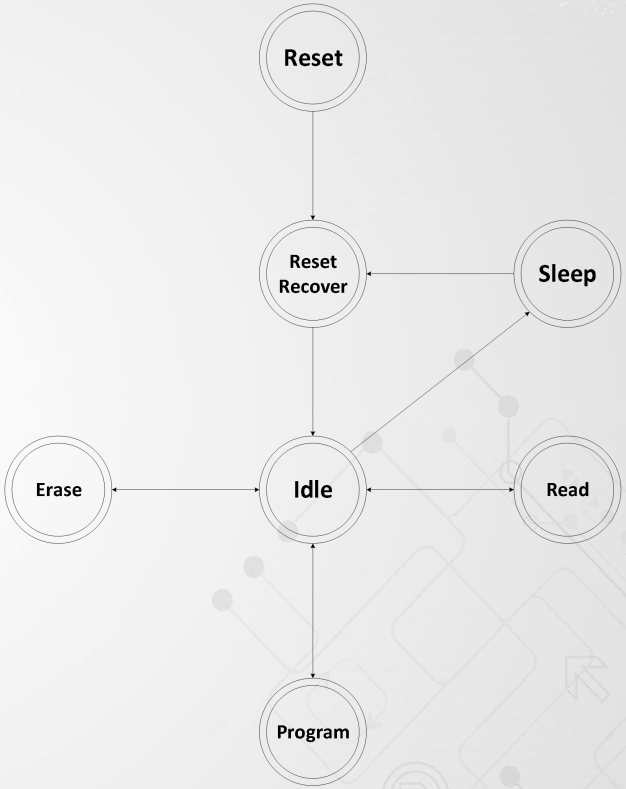
32kB flash空间划分



64kB flash空间划分

03 FLASH 功能特点

- FLASH 读取数据的操作，包括对 NVR 部分的读取和对 MAIN 部分的读取。
- FLASH 写入数据的操作，包括对 NVR 部分的写入和对 MAIN 部分的写入。
- FLASH 擦除操作，包括 CHIP 擦除和 SECTOR 擦除。NVR 部分仅支持 SECTOR 擦除，MAIN 部分支持 CHIP 擦除和 SECTOR 擦除。
- FLASH 深度休眠的操作，以降低芯片的休眠功耗。
- FLASH 存储体内容的加密操作。
- FLASH 的读取加速操作，以提升芯片整体运行效率。



FLASH 控制状态转换图

03 FLASH 读取操作

- 读操作为 FLASH 的基本操作。系统可通过两条路径访问 FLASH 内部的数据。

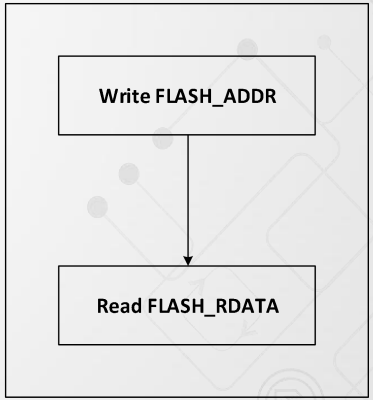
CPU 通过 AHB 总线，直接对 FLASH 执行**取指**取数操作。取指宽度为 32bit，且只能访问 MAIN 空间的数据。为了加快 MCU 的取指取数据的速度，硬件提供了加速的功能。

CPU 通过 AHB 总线，访问控制器的**寄存器**，间接实现读取 FLASH 内部数据的操作。可以访问 MAIN 和 NVR 空间的数据；若执行连续读取操作，硬件可自动完成地址累加，无需每次都更新地址寄存器的值。

- **FLASH_CFG.REGION 位指示当前访问空间。**

NVR(FLASH_CFG.REGION)	访问区域
0	MAIN区域
1	NVR区域

FLASH 访问空间分配表



FLASH 间接读取操作流程图

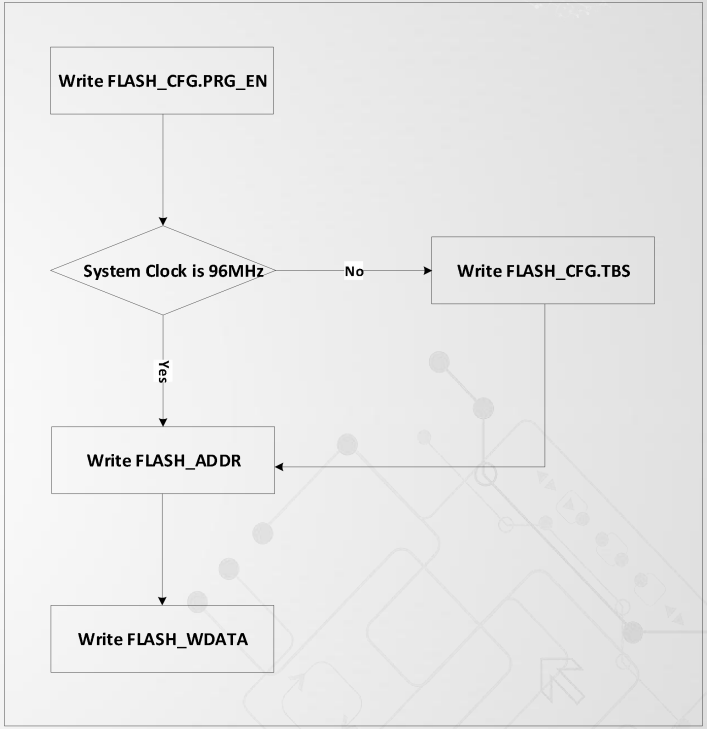
03 FLASH 编程操作

- FLASH 编程操作需要先执行擦除，然后才能执行数据编程操作。同时，只能通过访问 FLASH 控制器的寄存器，实现编程操作。
- FLASH编程操作具体流程为：

  控制寄存器 CFG，开启编程使能、

  地址寄存器 ADDR，写入编程地址、

  写数据寄存器 WDATA，写入编程数据。
- FLASH 写入/擦除操作的绝对时间是固定的，保证计数值的值×时钟频率等于恒定的时间。具体配置可查看FLASH_CFG.TBS说明。



FLASH 编程操作流程图

03 FLASH 擦除操作

- 擦除操作为 FLASH 的基本操作。系统只能通过访问 FLASH 控制器的寄存器实现。
- 执行对 FLASH 存储体的擦除操作。擦除分成 Sector 和 FullChip。分别对应，512Byte 的擦除和32KB/64kB 的擦除。通过配置 FLASH 控制寄存器选取执行Sector 或FullChip类型的擦除操作。
- NVR 区域只能实现 Sector 擦除；MAIN 区域可以实现 Sector 擦除和 FULL 擦除。

Name	Addresses	Size(Bytes)
Sector0	0x0000 0000 – 0x0000 01FF	512
Sector1	0x0000 0200 – 0x0000 03FF	512
Sector2	0x0000 0400 – 0x0000 05FF	512
…	…	…
Sector127	0x0000 FE00 – 0x0000 FFFF	512

FLASH Sector地址分配表

NVR(FLASH_CFG.REGION)	Sector Erase	NVR(FLASH_CFG.REGION)
0	Main区域	Main区域
1	NVR区域	Main区域

FLASH 擦除功能分配表

03 FLASH 在线升级(IAP)

- IAP 模式，实现中断向量表的重映射。在 LKS32MC08X 系列芯片中，包含了系统寄存器 VTOR，其地址为 0xE000_ED08。用于重新映射中断向量表入口地址。
- 默认值为 0x0，此时中断向量表入口地址为 0x0。当写入非 0 值时，中断向量表入口地址将映射到写入值对应的地址上，立即生效。
- 在 LKS32MC08X 系列芯片中，因为有 VTOR 寄存器。用户可根据自己需求，更新整个 FLASH 的内容。在线升级过程中可以使用中断，也可以关闭中断。

名称	复位值	偏移	位置	权限	说明
VTOR	0x0		[31:7]	RW	执行写入操作，写入中断向量表入口地址
			[6:0]	--	默认写0

IAP VTOR 寄存器表述

03 UART

➢UART 特征如下：

支持全双工工作

支持单线半双工工作

08x支持 7/8 位数据位，

支持 1/2 停止位

支持奇/偶/无校验模式

带 1 字节发送缓存

带 1 字节接收缓存

支持 Multi-drop Slave/Master 模式

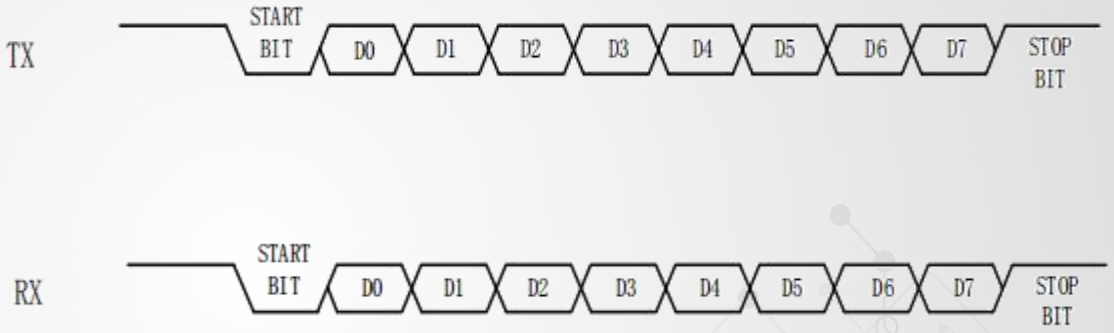
- 08x—UART 模块支持 DMA 操作。实现 DMA 搬移数据，能极大减轻 MCU 的负担。
- 波特率配置通过两级分频实现：

  UART 时钟=系统主时钟/(1+SYS_CLK_DIV2)

  波特率=UART 时钟/（256*UARTx_DIVH+UARTx_DIVL+1)
- UART 模块支持 Tx 与 Rx 端口互换。
- UART可实现单口半双工逻辑，
- UART数据发送：UART 包括一个字节发送缓冲区，当发送缓冲区有数据时，UART 将发送缓冲区的数据加载，一但数据进入发射队列，此时发送缓冲区空，当UARTx 将一个字节最后一位发送完毕，产生发送完成中断。
- UART数据接收：UART 包括一个字节的接收缓冲区，当完成一个字节的接收后，会产生接收中断，并将接收到字节存储到接收缓冲区，用户应当在 UART 接收完成下一个字节前完成此字节的读取，否则缓冲区会被写入新接收的字节。

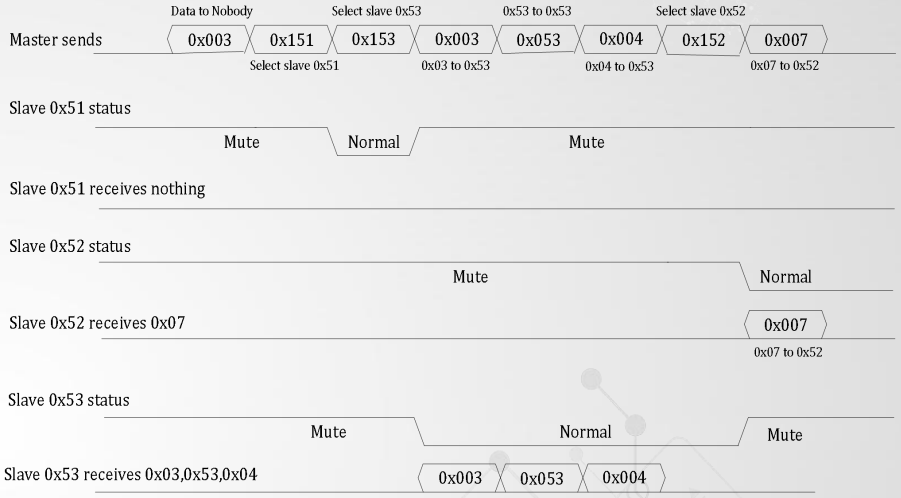
03 UART帧格式

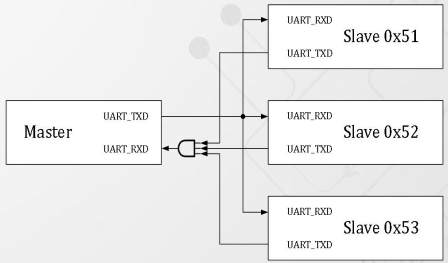
- UART 信号上收发的数据格式通常如下：
- 信号线空闲；
- 起始比特 (1 bit Start bit： 1 bit Zero)
- 数据字节 (data word, 8bits or 9bits, LSB first or MSB first)
- 停止比特 (1/2 bit Stop bit: 1/2bit Ones)



03 UART多机通信

- 多机通讯场景通常是一个设备作为主设备 master，另有若干个从设备 slave，主设备的UARTm_TXD 端口连接到所有从设备的 UARTs_RXD 端口，从设备的 UARTs_TXD 相与连接到主设备的UARTm_RXD 端口。
- 如图 所示，为一个主设备和 3 个从设备（地址分别为 0x51,0x52,0x53）的互联情况。
- 先设置从机的UARTx_ADR寄存器配置从机地址，只有与这个地址匹配的从机才能接收主机发送的数据，地址匹配后，主机发送的地址和数据是通过UARTx_BUFF发送，当UARTx_CTRL的BIT6为1时表示此时主机发送的是地址，当UARTx_CTRL的BIT6为0时表示主机发送的是数据，此外所有从机发送的数据主机是全部接收的。





UART 多机通讯示例

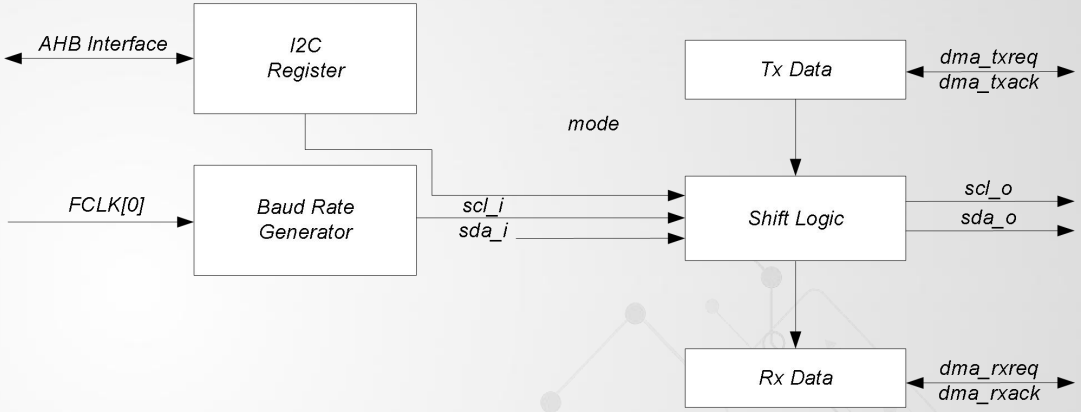
03

IIC

- IIC支持主机和从机模式。
- 支持标准（MCU）和快速两种模式（DMA）。
- 具单字节缓冲器的 DMA。
- 根据系统分频，实现不同的通讯速度。
- I2C 主设备功能：产生时钟、START 和 STOP 事件。
- I2C 从设备功能：可编程的 I2C 硬件地址比较（仅支持 7 位硬件地址）、停止位检测。
- IIC提供多主机功能，控制所有 I2C 总线特定的时序、协议、仲裁和定时。
- IIC支持四种传输方式：
- 从模式单字节发送，从模式 DMA 发送
- 从模式单字节接收，从模式 DMA 接收
- 主模式单字节发送，主模式 DMA 发送
- 主模式单字节接收，主模式 DMA 接收

03 IIC 功能描述

- I2C 接口同外界通讯只有 SCL 和 SDA 两根信号线。
- scl_i：时钟信号。当 I2C 接口配置为从模式时，此为 I2C 总线的时钟输入信号。
- sda_i：数据信号。当 I2C 接口接收数据时（无论主模式还是从模式），此为 I2C 总线的数据输入信号。
- scl_o：时钟信号。当 I2C 接口配置为主模式时，此为 I2C 总线的时钟输出信号。
- sda_o：数据信号。当 I2C 接口发送数据时（无论主模式还是从模式），此为 I2C 总线的数据输出信号。
- sda_oe：数据使能信号。当 sda_o 输出时，sda_oe 有效；当 sda_i 输入时，sda_oe 无效。



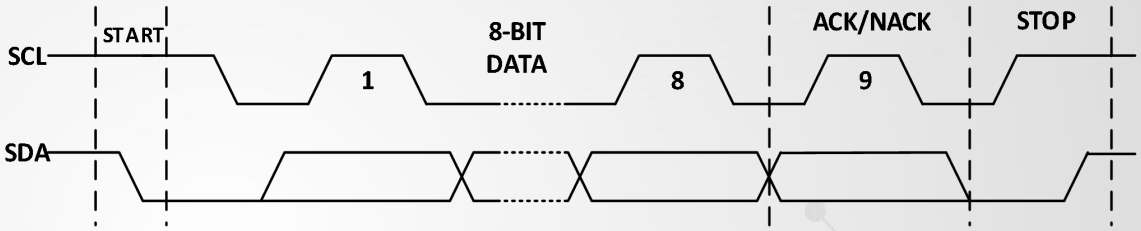
I2C 模块顶层功能框图

03 IIC 传输时序

- 数据和地址按 8 位/字节进行传输，高位在前，低位在后。跟在起始条件后的 1 个字节是地址。地址数据中高7位为地址数据，最低位为读写控制位，最低位为0表示主设备发送数据，为1表示设备主设备接收数据，注意地址只在主模式发送。

- 在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收器必须回送一个应答位(ACK)给发送器。软件可以开启或禁止应答(ACK)，并可以设置 I2C 接口的地址。

- 最后硬件自动产生STOP结束信号。



基本 I2C 传输时序图

03 IIC 通讯速度设置

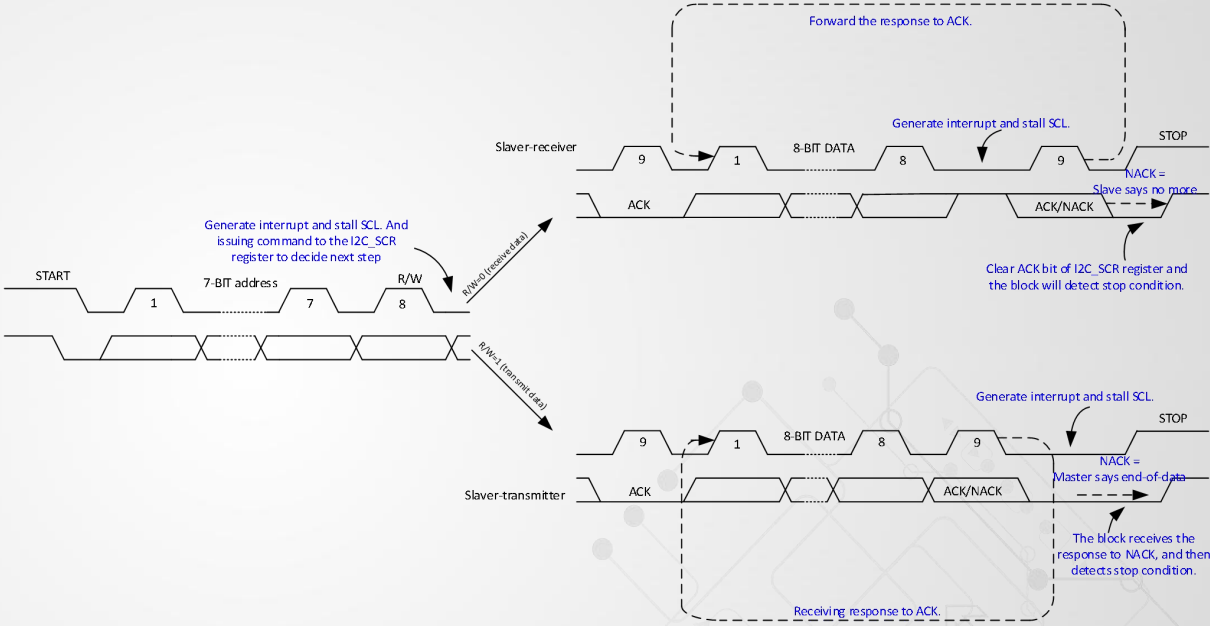
- I2C 接口的工作时钟来自系统时钟的分频，分频寄存器为 SYS 模块的 CLK_DIV0。
- I2C 接口采用同步设计，需要对外部设备的信号进行同步采样，同步时钟为 I2C 接口工作时钟。数据和时钟信号的时钟频率为接口工作时钟/16。
- I2C 模块工作时钟频率 = 系统频率 / (CLK_DIV0 + 1)。
- I2C 波特率 = I2C 模块工作时钟频率 / 17。

03 IIC 接口从模式

- 默认情况下，I2C 接口主模式和从模式均关闭。若工作在从模式，需使能从模式。
- 从模式具有硬件地址匹配功能。
- SYS_CLK_DIV0 是 I2C 接口工作时钟的分频系数。
- 从模式下，I2C 接口时刻在监控总线上的信号。一旦检测到起始条件，其将保存地址位数据和读写位数据。
- 从模式下，单字节接收模式。每次收到一个字节的数据后，产生中断，此时 I2C 接口可拉低 SCL，直至中断完成，继续后续操作。
- 从模式下，单字节发送模式。每次发送一个字节完毕后且收到响应（ACK/NACK），产生中断，此时 I2C 接口可拉低 SCL，直至中断完成，继续后续操作。
- **从模式下，DMA 接收模式。每次收到 SIZE 约定后的数据，产生中断，此时 I2C 接口可拉低 SCL，直至中断完成。**
- **从模式下，DMA 发送模式。每次发送 SIZE 约定后的数据并收到响应（ACK/NACK），产生中断，此时 I2C 接口可拉低 SCL，直至中断完成。**

03 IIC 从模式单字节传输（MCU）

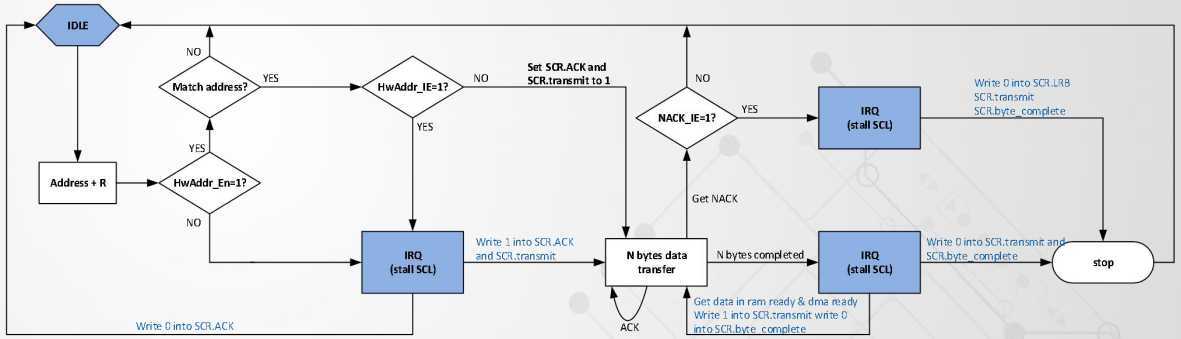
- 单字节传输，并不意味着仅仅传输一个字节数据，其含义为每次传输完一个字节的数据后，将产生中断判断是否还要继续传输。单字节传输的极端情况是仅传输一个字节的数据。下图为单字节传输的总线示意图。从图可知，一般单字节传输的流程如下：
- 地址匹配，产生地址匹配中断，准备开始传输。
- 若是接收模式，一个字节接收完毕后，产生中断，软件判断是否继续接收，返回 ACK/NACK 响应。
- 若是发送模式，一个字节发送完毕后，等待响应（ACK/NACK），产生中断，根据响应判断后续操作。
- 获得总线 STOP 事件，本次传输完成。



从模式下单字节传输示意图

03 IIC 从模式多字节发送（DMA）

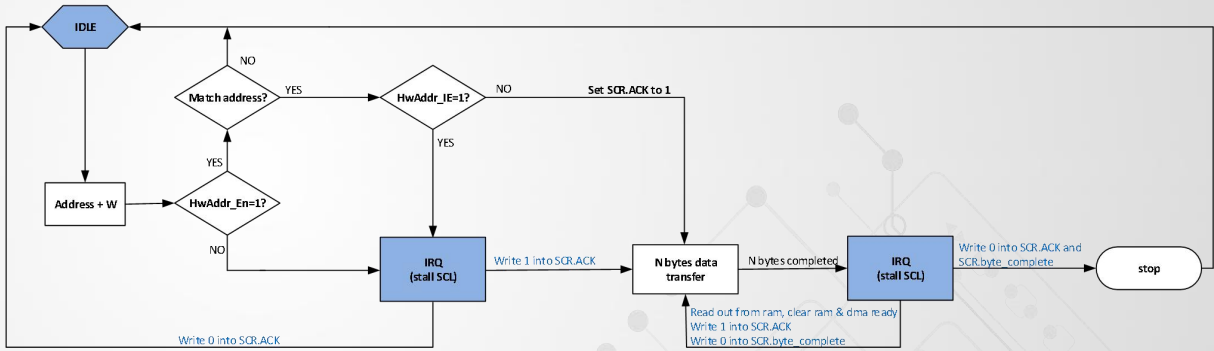
- 地址匹配后，配置完毕 DMA。通过发送 DMA 请求，将字节从 RAM 搬移到 I2C_DATA 寄存器，然后经由内部移位寄存器发送到 SDA 线上。在 I2C_DATA 数据没有准备好之前，从设备可拉底 SCL，直到待发送数据已写入 I2C_DATA 寄存器。从模式下发送数据，需要软件协助触发第一次 DMA 搬移，配置 I2C_BCR.BYTE_CMPLT 为 1 即可。
- I2C 接口在发送完毕 I2C_BCR.BURST_SIZE 约定的字节数据后都执行下列操作:
- 如果接收到 ACK 位，配置 DMA，准备下一批数据，继续传输。准备过程中，可拉低 SCL。
- 如果接收到 NACK 位，停止准备下一批数据，停止本次传输。



从模式下多字节发送示意图

03 IIC 从模式多字节接收

- 地址匹配后，配置完毕 DMA。配置好 DMA，从 SDA 线接收到的数据先存入 I2C_DATA 寄存器，然后通过 DMA搬移到 RAM。I2C 接口在接收到一个字节后都将执行 DMA 请求。
- 完成 I2C_BCR.BURST_SIZE 约定的数据传输
- 后，执行下列操作:
- 如果设置了 ACK 位，在I2C_BCR.BURST_SIZE 约定数据接收完毕后，产生一个 ACK 应答脉冲。
- 如果清除了ACK位，在 I2C_BCR.BURST_SIZE 约定数据接收完毕后，产生一个 NACK 应答脉冲。



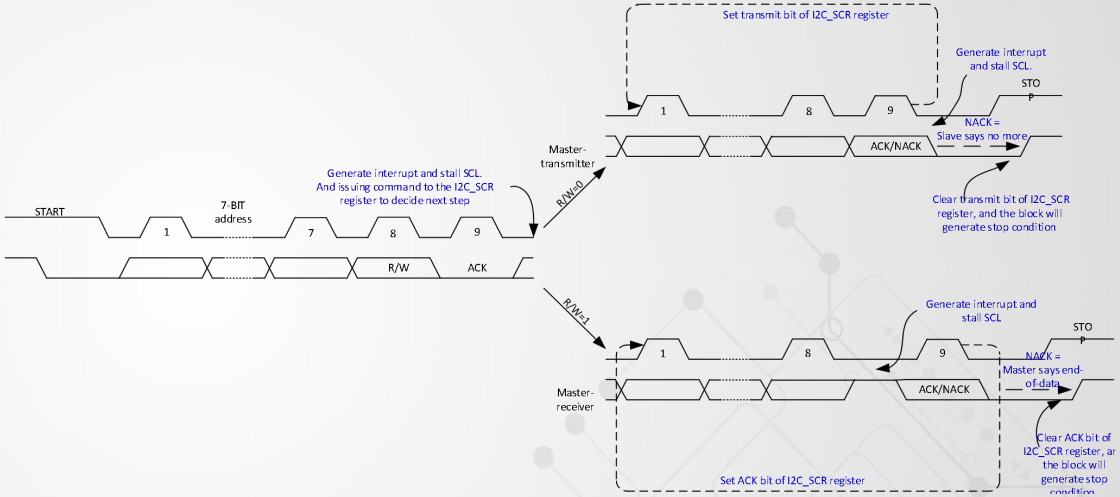
从模式下多字节发送示意图

03 IIC 接口主模式

- 默认情况下，I2C 接口主模式和从模式均关闭。
- 在系统寄存器 CLK_DIV0 中设定 I2C 接口的工作时钟。
- 主模式下，单字节接收模式。单字节传输，并不意味着仅仅传输一个字节数据，其含义为每次传输完一个字节的数据后，将产生中断判断是否还要继续传输。单字节传输的极端情况是仅传输一个字节的数据。
- 主模式下，单字节发送模式。I2C 接口将字节从 I2C_DATA 寄存器经由内部移位寄存器发送到 SDA 线上。在I2C_DATA 数据没有准备好之前，主设备可不产生 SCL 时钟信号，直到待发送数据已写入 I2C_DATA寄存器。
- 主模式下，DMA 发送模式。通过发送 DMA 请求，将字节从 RAM 搬移到 I2C_DATA 寄存器，然后经由内部移位寄存器发送到 SDA 线上。
- 主模式下，DMA 接收模式。总线空闲，配置好 DMA，从 SDA 线接收到的数据先存入 I2C_DATA 寄存器，然后通过 DMA 搬移到 RAM。I2C 接口在接收到一个字节后都将执行 DMA 请求。

03 IIC 主模式单字节传输

- 判断总线是否空闲，若空闲，准备开始传输。
- 若是接收模式，一个字节接收完毕后，产生中断，软件判断是否继续接收，返回 ACK/NACK 响应。
- 若是发送模式，一个字节发送完毕后，等待响应（ACK/NACK），产生中断，根据响应判断后续操作。
- 发送总线 STOP 事件，本次传输完成。



主模式下单字节传输示意图

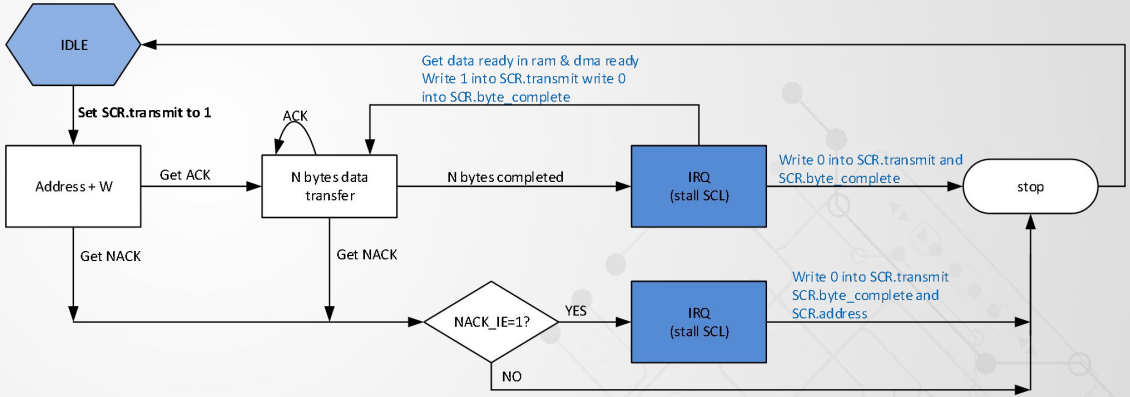
03 IIC 主模式多字节发送

- 总线空闲，配置完毕 DMA。通过发送 DMA 请求，将字节从 RAM 搬移到 I2C_DATA 寄存器，然后经由内部移位寄存器发送到 SDA 线上。在 I2C_DATA 数据没有准备好之前，主设备可不产生 SCL时钟，直到待发送数据已写入 I2C_DATA 寄存器。I2C 接口在发送完毕 SIZE 约定的字节数据后都执行下列操作:

- 如果接收到 ACK 位，配置 DMA，准备下一批数据，继续传输。准备过程中，可拉低 SCL。

- 如果接收到 NACK 位，停止加载下一批数据。

- 如果本次数据发送完毕，停止后续发送。产生 STOP 事件，停止本次传输。



主模式下多字节发送示意图

03

IIC 主模式多字节接收

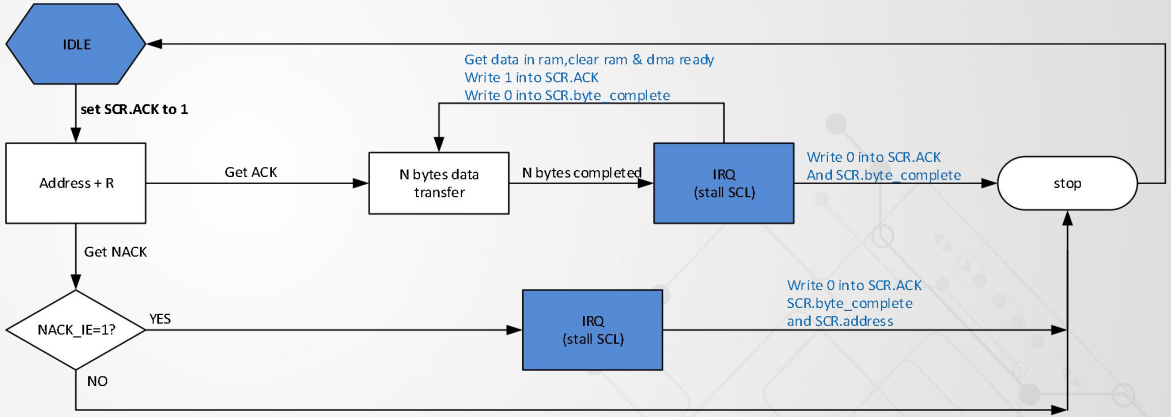
- 总线空闲，配置好 DMA 从 SDA 线接收到的数据先存入 I2C_DATA 寄存器，然后通过 DMA 搬移到 RAM。I2C 接口在接收到一个字节后都将执行 DMA 请求。

- 完成 I2C_BCR.BURST_SIZE 约定的数据传输后，执行下列操作:

- 如果设置了 ACK 位，在 I2C_BCR.BURST_SIZE 约定数据接收完毕后，产生一个 ACK 应答脉冲。

- 如果清除了 ACK 位，在 I2C_BCR.BURST_SIZE 约定数据接收完毕后，产生一个 NACK 应答脉冲。

- 产生 STOP 事件，停止本次传输。



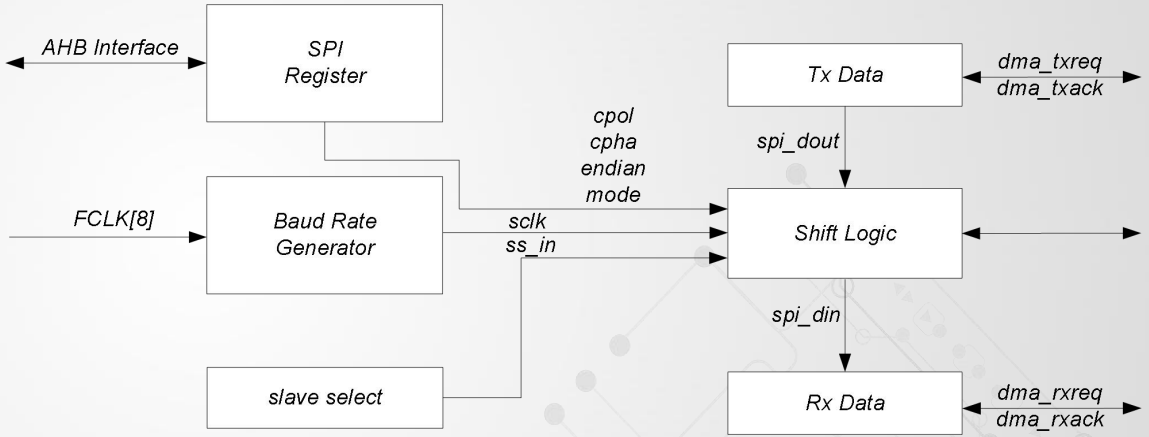
主模式下多字节接收示意图

03 IIC中断介绍

- I2C 接口包含三种类型的中断事件，分别是：数据传输完成事件，总线错误事件、STOP 事件、NACK 事件和硬件地址匹配事件。
- 数据完成事件。当前数据传输完成，高电平有效，对 I2C_SCR 的 BIT0 写 0 清除。
- 总线错误事件。传输过程中，总线产生错误的 START 事件/STOP 事件，高电平有效，对 I2C_SCR的 BIT7 写 0 清除。
- STOP 事件。当前数据传输完成，主设备发送 STOP 事件，从设备收到 STOP 事件并产生相应中断。高电平有效，对 I2C_SCR 的 BIT5 写 0 清除。
- 硬件地址匹配事件。从模式下接收到的地址同本设备地址匹配，产生相应中断。高电平有效，对 I2C_SCR 的 BIT3 写 0 清除。
- 使用 DMA 协助数据传输。若本模块为接收模式，I2C 收到数据后还需要通过 DMA 搬移到 RAM，此时数据最终完成是要看 DMA 是否搬移完成，若使用 I2C 的完成中断作为判断依据的话，推荐在中断处理函数中查询下 DMA 的状态。若本模块为发送模式，直接使用 I2C 的完成中断作为判断依据即可。

03 SPI

- SPI 支持 Master 和 Slave 工作模式，工作模式软件可选，默认为SPI Motorola 模式。
- 全双工传输，可根据应用情况，使用 3 根或者 4 根信号线。
- 支持半双工传输，可根据应用情况，使用 2 根信号线。
- 最快BAUD 率为系统最高时钟频率的 1/8。
- 片选信号均可选。Master 模式下，片选信号可以软件控制或硬件产生；Slave 模式下，片选信号可以恒定有效，也可以来自外界设备。
- 无本地 FIFO，支持 DMA 操作。包含溢出检测和片选信号异常检测



SPI 模块结构框图

03

SPI 全双工模式

- 默认情况下，SPI 接口配置为全双工模式。
- 接口为 Master 模式时：

  spi_din 为数据输入，接外部 Slave 设备的 MISO。

  spi_dout 为数据输出，接外部 Slave 设备的 MOSI。

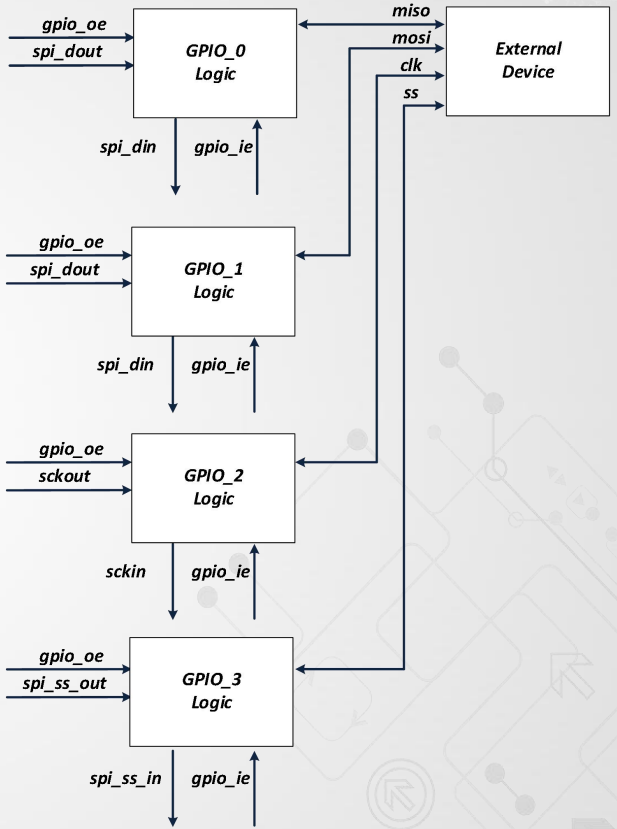
  spi_ss_out 为片选信号，根据应用情况选择是使用该信号还是软件控制其它 GPIO 实现。

- 接口为 Slave 模式时：

  spi_din 为数据输入，接外部 Master 设备的MOSI。

  spi_dout 为数据输出，接外部 Master 设备的MISO。

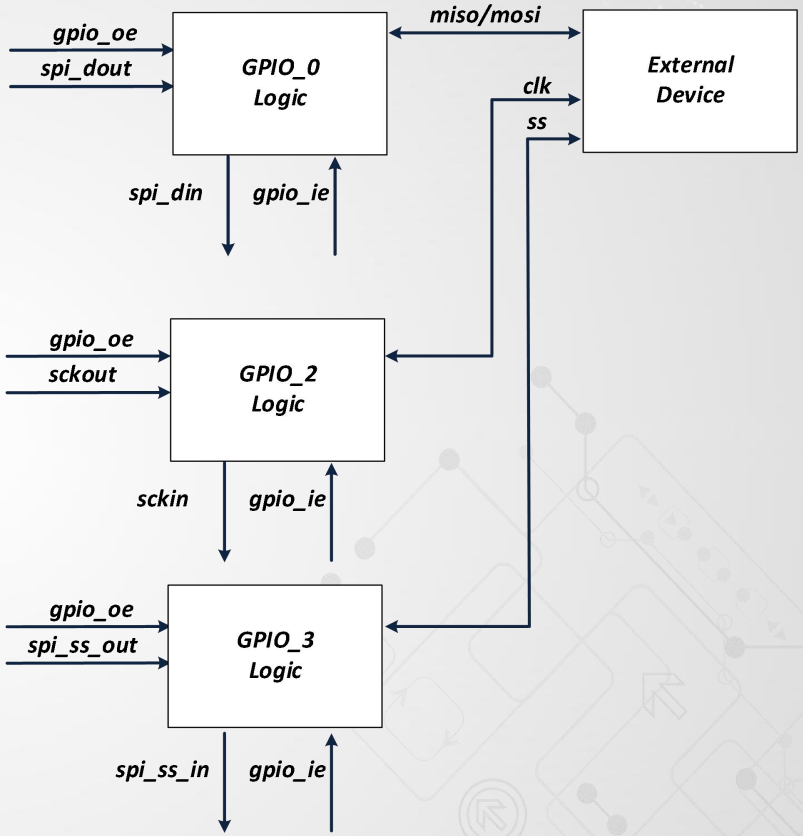
  spi_ss_in 为片选信号，根据应用情况是使用该信号还是片选恒有效。



SPI 接口全双工模式互连框图

03 SPI 半双工模式

- SPI 接口可配置为半双工模式。数据传输只需要一根数据线。数据信号的变化，发生在时钟信号的边沿，即同步于时钟信号。一次传输只能是一个方向的，要不就是发送，要不就是接收。
- 仅发送：GPIO_0 的 oe 使能，发送 spi_dout 数据到外界；GPIO_0 的 ie 关闭，spi_din 恒定输入为 0。此模式下，支持 DMA 传输、支持 Master/Slave 模式下的发送。
- 仅接收：GPIO_0 的 oe 关闭，spi_dout 无法发送数据到外界；GPIO_0 的 ie 开启，spi_din 接收来自外部的数据。此模式下，支持DMA 传输、支持 Master/Slave 模式下的接收。

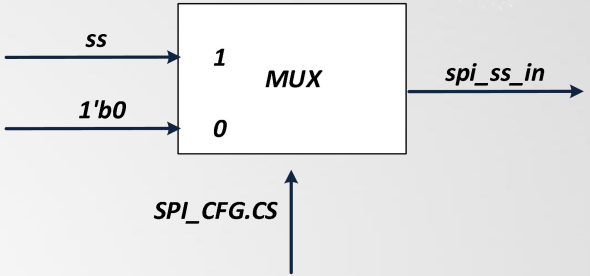


SPI 接口半双工模式互连框图

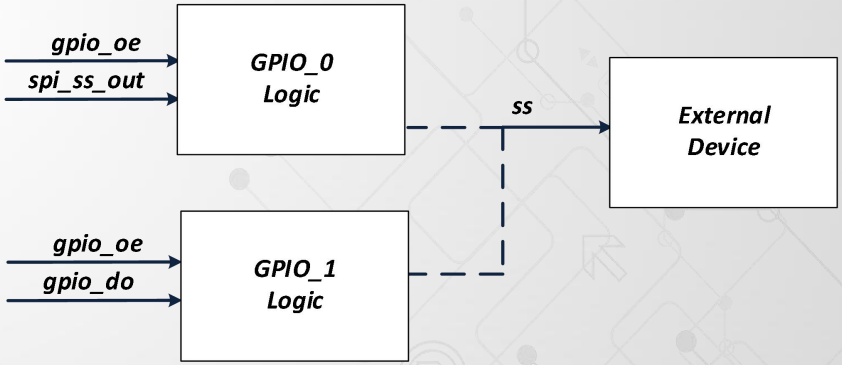
03

SPI 片选信号

- 本接口做Slave 模式时，片选信号可选，CFG[5]决定片选来源。ss 为 Master 设备发出的选通使能信号，低电平有效。
- 本接口做Master 模式时，片选信号亦可选。模块硬件产生了标准的片选信号，实际应用可屏蔽此信号通过软件操作额外的 GPIO 实现。
- 右图虚线仅表示不确定。若使用 spi_ss_out 为 ss 的源头，那么将 GPIO_0 同外界设备互连；若使用软件操作 GPIO 的方式，那么可将 GPIO_1 同外界设备互连。



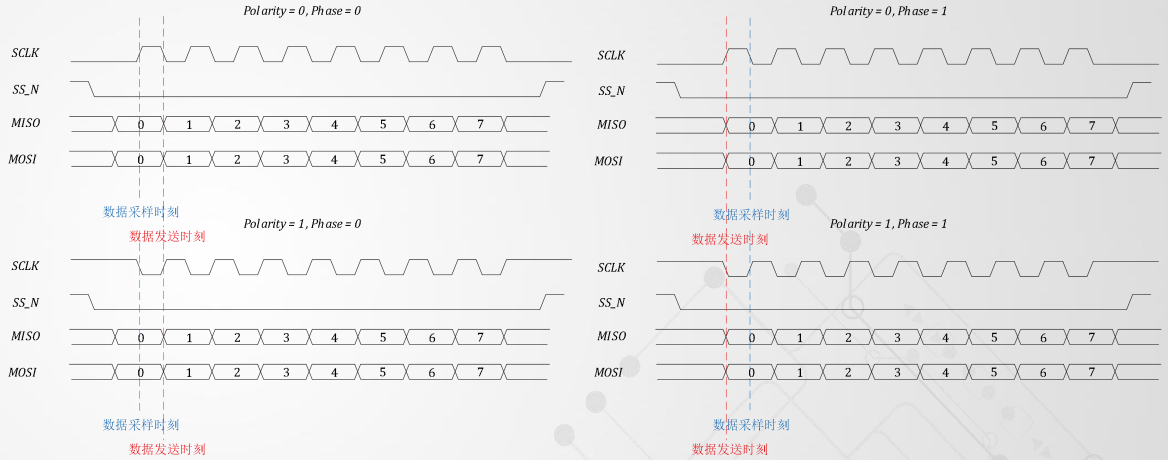
SPI 模块 Slave 模式片选信号选择



SPI 模块 Master 模式片选信号选择

03 SPI 通讯格式

- Polarity 控制了 SPI 时钟信号在默认情况下的电平状态。
- Polarity 为 0 时，默认时钟电平为低电平。
  Polarity 为 1 时，默认电平为高电平。
- Phase 控制了 SPI 数据的发送/接收时刻。
- Phase 为 0 时，时钟从默认电平到第一个跳变边沿为采样数据时刻。
- Phase 为 1 时，时钟从默认电平到第一个跳变边沿为发送数据时刻。
- SPI 数据传输格式分成两种：MSB 和 LSB。



SPI 通讯信号极性相位

03 SPI 波特率设置

- SPI 接口时钟通过对系统时钟分频获得，分频系数来自 BAUD[5:0]。分频范围是 1 ~ 128，对应的 BAUD[5:0]的值为 0 ~ 63。
- SPI 协议为半拍协议，上升沿发送数据，下降沿采集数据；或者下降沿发送数据，上升沿采用数据。
- SPI 接口采用同步设计，需要对外部设备的信号进行同步采样，同步时钟为系统时钟。数据和时钟信号（此时为 Slave 模式）的同步，需要两拍系统时钟。考虑到时钟相位的偏移，此时需要一拍系统时钟的冗余，由此推导出最快的 BAUD 率为系统时钟的 1/8，高电平周期为四拍系统时钟，低电平周期为四拍系统时钟。

SPI 波特率寄存器

位置	位名称	说明
[5:0]	BAUD	SPI 传输波特率配置，SPI 实际传输速度计算公式为： SPI 传输速度 = 系统时钟 / (2*(BAUD + 1)) 切记，BAUD 的配置值不能小于 3。

03 SPI 数据传输

DMA传输方式

- 在大容量数据传输应用下，SPI 接口支持 DMA 传输，减轻 MCU 的负担。一次传输，最大传输量255字节，最小传输量为 1 字节。在全双工模式下，接收和发送均可实现 DMA 传输；在半双工模式下，仅接收或发送实现 DMA 传输。

- 在接收到新的数据后，硬件自动产生 DMA 请求，通过 DMA 模块将数据搬移到 RAM 中。在发送新数据前，硬件自动产生 DMA 请求，通过 DMA 模块将数据从 RAM 中搬移到 SPI 接口。因 SPI 无 FIFO，SPI 发送一次 DMA 请求，DMA 只能搬移一个字节数据。若要实现多字节搬移，DMA 需配置为多轮搬移，每一轮搬移一个字节的方式。

DMA传输，推荐软件配置流程：

1、初始化 DMA 模块，将本次发送的数据来源，接收的数据去向配置好，传输长度配置完毕。

2、初始化 GPIO 模块，将 SPI 复用的 GPIO 配置完毕。

3、初始化 SPI 接口，IE/CFG/BAUD/SIZE 等寄存器配置完毕。

4、触发 SPI 接口，进入发送/接收状态。触发条件是 MCU 对 TX_DATA 寄存器执行写操作，因最终发送的数据来自 DMA，本次 MCU 写入的数据不会混入 SPI 发送流程。

03 SPI 数据传输

MCU传输方式

- MCU 传输，一次只能发送/接收 1 个字节，每次完成后需要通过中断或者轮询的方式判断传输是否完成。
- SPI 接口支持 DMA 传输，也支持 MCU 传输。MCU 传输，SPI 接口无需触发发送状态，只要MCU将数据搬移到 SPI 接口，就可开始发送数据到外界。

MCU传输，推荐软件配置流程：

1、初始化 GPIO 模块，将 SPI 复用的 GPIO 配置完毕。

2、初始化 SPI 接口，IE/CFG/BAUD/SIZE 等寄存器配置完毕。注意 SIZE 只能配置为 1。

3、MCU 对 TX_DATA 寄存器执行写操作，触发 SPI 接口进入发送流程，发送的数据来自 MCU 对TX_DATA 写入值。

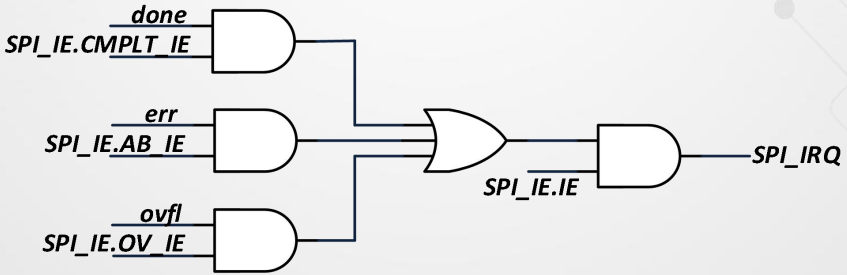
注意：若需要连续发送，则需要重新配置 SIZE 和 TX_DATA 寄存器。

03

SPI 中断处理

SPI 接口包含三种类型的中断事件，分别是：数据传输完成事件，异常事件和溢出事件。

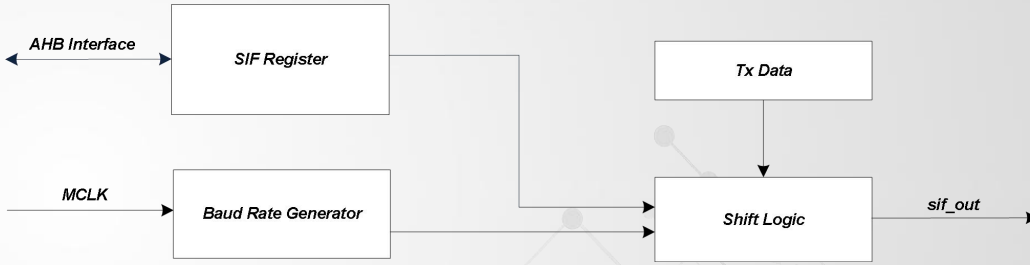
- 数据完成事件，当前数据传输完成。高电平有效，对 SPI_IE. CMPLT_IF 写 1 清除。
- 异常事件，SPI 接口为 Slave 模式，若在传输过程中片选信号受到干扰，被拉高，将产生片选异常事件。高电平有效，对 SPI_IE. AB_IF 写 1 清除。
- 溢出事件，SPI_RX_DATA 寄存器数据没有及时被读走，将产生溢出事件。高电平有效，对 SPI_IE.OV_IF 写 1 清除。
- 上述事件，默认是不触发 SPI 中断，可以通过配置 IE[6:4]使能事件产生中断。
- 发送模式下，DMA 先完成搬移操作，SPI 后发送完毕。SPI 发送完毕，可作为本次传输完成标识。
- 接收模式下，SPI 接收完毕，触发 DMA 搬移。DMA 搬移完成，可作为本次传输完成标识。



SPI模块中断选信号产生图

03 SIF

- SIF 总线接口，支持 SIF 协议，液晶显示器与电动车控制器之间的数据传输，实现其对运行状态和故障的检测。
- 采用国际标准 SIF 通信协议，接口通用方便。
- 主从方式采用单线单向传输，即只需要一根传输线路，电动车控制器为发送，仪表为接收方。
- 传输线与电动车控制故障运行灯共用 I/O 口，不占用额外资源。
- 传输波特率自适应范围宽，主机可以利用空闲时间发送数据。
- 时间基准单位范围广，32us <Tosc<320us。
- 数据的电平遵守 TTL 规范。一个中断向量。



SIF 模块顶层功能框图

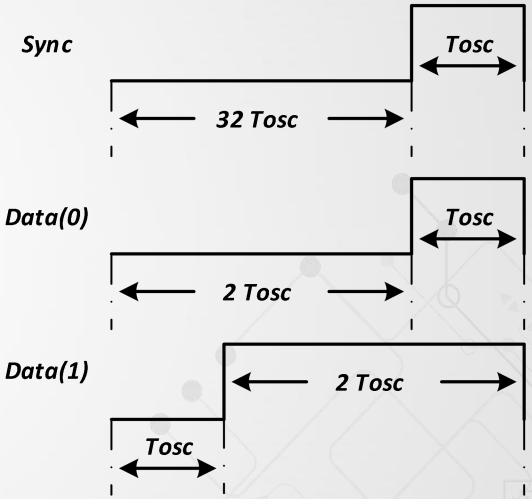
SIF 接口同外界通讯只有 sif_out 一根信号线。

03 SIF 传输时序

- SIF 接口仅有一种运行模式：主发送模式。因为 SIF 接口的传输速度很慢，不支持 DMA 传输。每次传输一个字节完毕后，产生中断信号。
- 占空比为 2:1，可选配为 3:1。SIF_FREQ[0]控制占空比的选择。

通讯速度设置

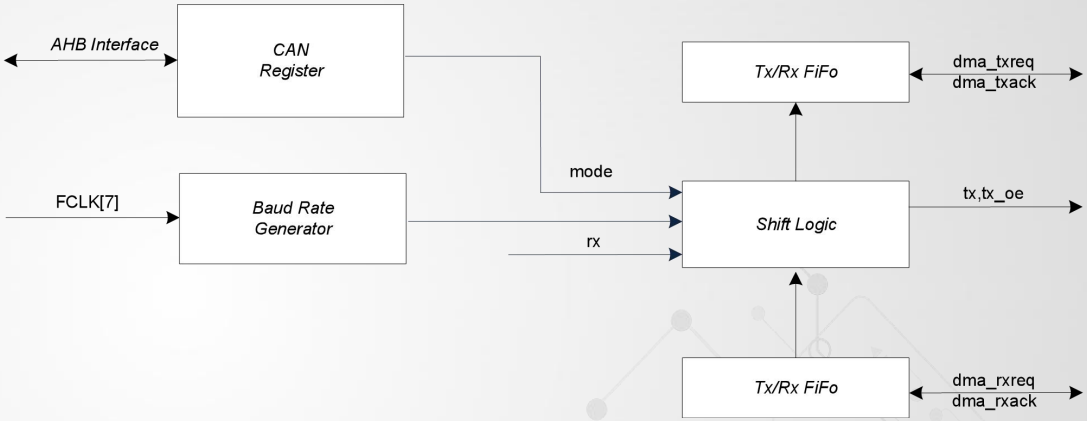
SIF 接口的基本时间单位是 32us--Tosc。Tosc 的产生来自系统时钟。在不同系统频率下，通过配置 SIF_FREQ 寄存器，获得我们需要的基准时间。



SIF 基本传输时序图

03 CAN

- CAN支持 BOSCH 2.0A 和 2.0B 协议。2.0A 等效 CAN1.2，包含了 11 位 ID 格式；2.0B 包含了 11 位ID 和 29 位 ID。
- 支持 SJA1000 大部分功能。
- 有两种模式：工作模式和复位模式。
- 支持监听和自测试。监听模式下，仅接收外界信号但不会返回任何信号。
- 包含 DMA 功能。
- CAN全温度工作时需要外接晶振提供工作时钟。



CAN 模块顶层功能框图

CAN 接口同外界通讯只有 tx 和 rx 两根信号线。

03

CAN工作模式

- CAN 模块主要包含两个工作模式：正常工作模式和复位模式。
- 复位模式作用：时序参数、ID 配置、错误统计值等均在此模式下设定好。CAN_MOD.0 为 1，是复位模式。硬件复位结束后，CAN 模块处于复位模式下。
- 正常工作模式，CAN_MOD.0 为 0。此时，可以正常响应 CAN 总线请求，CAN数据的收发均在正常模式下进行。
- CAN配置流程：先将CAN配置为复位模式，将ID 配置、错误统计值等均在此模式下设定好后，再将CAN配置为正常模式进行CAN数据的正常收发工作。
- CAN扩展了监听模式（Listen Only）和自测试模式（Self Test）。前者，类似一个搜集器，只接收 CAN 总线上的数据，不发送任何数据。后者，是内部自测试，发送的数据同时被自己接收，检测内部功能是否正确。

03 CAN传输方式

DMA传输方式

- 在大容量数据传输应用下，CAN 接口支持 DMA 传输，减轻 MCU 的负担。一次传输，SFF 为 11个字节，EFF 为 13 个字节。
- 在接收到新的数据后，硬件自动产生 DMA 请求，通过 DMA 模块将数据搬移到 RAM 中。在发送新数据前，硬件自动产生 DMA 请求，通过 DMA 模块将数据从 RAM 中搬移到 CAN 接口。
- 本 CAN 模块设计的 DMA 不同于其它模块的 DMA 搬移操作，需要 MCU 介入部分搬移操作。假定当前配置 CAN 模块发送 N 帧数据，那么第一帧数据需要 MCU 搬移到 CAN 模块寄存器中，后续帧（N-1）的数据可由 DMA 实现搬移。

MCU传输方式

- MCU 传输，根据 SFF 还是 EFF，搬移 11 个字节或者 13 个字节；完成后需要通过中断或者轮询的方式判断传输是否完成。

CAN传输，推荐软件配置流程如下：

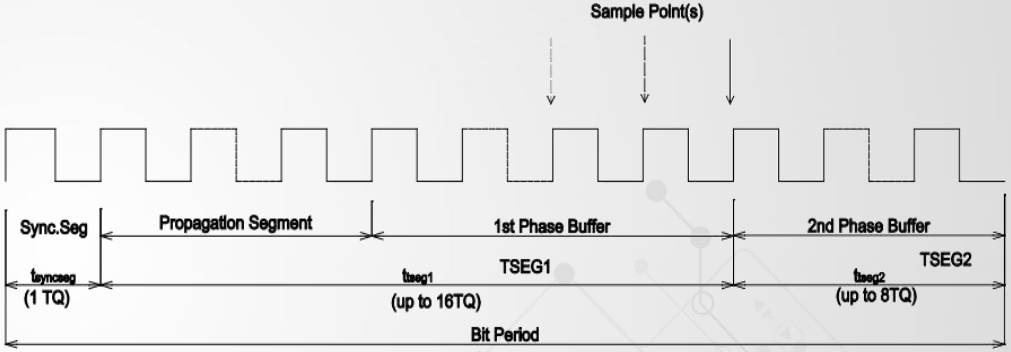
1、初始化 GPIO 模块，将 CAN 复用的 GPIO 配置完毕。

2、初始化 CAN 接口，控制寄存器配置完毕。使用DMA传输时，还需初始化DMA模块，将本次发送的数据来源，接收的数据去向配置好，传输长度配置完毕。

3、若MCU 触发 CAN 接口进入发送流程，发送的数据来自 MCU 对 CAN_TXDATA 写入值。若DMA触发 CAN 接口，直接进入发送状态即可。

03 CAN 通讯采样设置

- CAN 的波特率设定，主要依靠 CAN_BTR0 和 CAN_BTR1 两个寄存器配合完成。
- CAN_BTR0 主要是配置 TQ 参数，CAN_BTR1 主要处理 1-bit 数据的采样点、采样次数以及宽度信息。
- CAN_BTR0 设置传输基本时间单元参数 TQ：TQ = 2 * Tclk * (CAN_BTR0.BAUDRATE + 1)
- CAN_BTR1 设置波特率；同时，可调节 BIT 信息中各个部分的宽度（TSEG1、TSEG2 和 Sync.Seg），找到理想采样点。
- SEG1 段时间计算公式：Tseg1 = TQ*(CAN_BTR1.SEG1 + 1)
- SEG2 段时间计算公式：Tseg2 = TQ*(CAN_BTR1.SEG2 + 1)
- 波特率计算公式为：Can Baudrate = TQ * (1 + Tseg1 + Tseg2)



CAN 模块 Bit 周期介绍图

03 CAN 通讯采样设置

常规波特率索引值如下：

Can波特率	BTR0	BTR1
1Mbps	0x05	0x14
800Kbps	0x05	0x16
666Kbps	0x85	0xB6
500Kbps	0x05	0x1C
400Kbps	0x05	0xFA
250Kbps	0x0B	0x1C
200Kbps	0x05	0xFA
125Kbps	0x17	0x1C
100Kbps	0x1D	0x1C
83.33Kbps	0x17	0x6F
80Kbps	0x97	0xFF
66.66Kbps	0x1D	0x6F
50Kbps	0x3B	0x1C
40Kbps	0xAF	0xFF

03 CAN ID 滤波

- CAN 总线，可以挂载很多设备。通过 ID 号，不同设备可以知道当前总线上发送的帧是否需要被自己接收，或者发送出去的帧，是否有响应。
- 接收滤波器允许 CAN 控制器根据帧 ID 过滤接收帧（有时可以过滤帧的第一个数据字节和帧类型）。只有通过过滤的帧才能存储到接收 FIFO 中。
- CAN 帧的 ID 号码，有两种长度--11 位和 29 位。前者对应的 SFF（standard frame format），后者对应 EFF（extended frame format）。
- CAN ID滤波通过 CAN_ACR 和 CAN_AMR 判断当前 CAN 模块可接收的 ID范围。CAN_ACR 列出了一个特定的 ID，CAN_AMR 为 MASK 寄存器，标识 CAN_ACR 中对应位数据同接收到的 ID 对应位数据需完全匹配，哪些位可以不用。CAN_AMR 对应位是 0，表示接收到的 ID 对应位需要同 CAN_ACR 对应位匹配；CAN_AMR 对应位是 1，表示接收到的 ID 对应位不需要同 CAN_ACR 对应位匹配。
- CAN_MOD.3 决定了 CAN_ACR 包含两个滤波 ID 还是一个滤波 ID。为 1 的话，CAN_ACR 包含一个长的滤波 ID；为 0 的话，CAN_ACR 包含了两个短的滤波 ID。两个滤波 ID 情况下，只要接收帧的ID 同其中一个匹配，就会被 CAN 接收。

CAN_ACR（ID）	.... 1111 0001 1000
CAN_AMR	.... 0000 0000 0001
可以匹配ID	.... 1111 0001 100x

03 CAN SFF(标准帧) ID 滤波

SFF，单滤波ID数据格式

CAN_TXRX1	CAN_TXRX2	CAN_TXRX3	CAN_TXRX4
ID.28..ID21	ID.20..ID.18 RTR X X X X	Data Byte 1	Data Byte 2

滤波ID设置

ACR0[7:0]	ACR1[7:4] (ACR1[3:0] unused)	ACR2[7:0]	ACR3[7:0]
AMR0[7:0]	AMR1[7:4] (AMR1[3:0] unused)	AMR2[7:0]	AMR3[7:0]

SFF，双滤波ID数据格式

CAN_TXRX1	CAN_TXRX2	CAN_TXRX3	CAN_TXRX4
ID.28..ID21	ID.20..ID.18 RTR X X X X	Data Byte 1	Data Byte 2

滤波ID1

ACR0[7:0]	ACR1[7:0]	ACR3[3:0]
AMR0[7:0]	AMR1[7:0]	AMR3[3:0]

滤波ID2

ACR2[7:0]	ACR3[7:4]
AMR2[7:0]	AMR3[7:4]

03 CAN EFF（扩展帧） ID 滤波

EFF，单滤波ID数据格式

CAN_TXRX1	CAN_TXRX2	CAN_TXRX3	CAN_TXRX4
ID.28..ID21	ID.20..ID.13	ID.12..ID.5	ID.4..ID.0 RTR X X

滤波ID设置

ACR0[7:0]	ACR1[7:0]	ACR2[7:0]	ACR3[7:2]
AMR0[7:0]	AMR1[7:0]	AMR2[7:0]	AMR3[7:2]

EFF，双滤波ID数据格式

CAN_TXRX1	CAN_TXRX2	CAN_TXRX3	CAN_TXRX4
ID.28..ID21	ID.20..ID.13	ID.12..ID.5(not matched)	ID.4..ID.0 RTR X X(not matched)

滤波ID1

ACR0[7:0]	ACR1[7:0]
AMR0[7:0]	AMR1[7:0]

滤波ID2

ACR2[7:0]	ACR3[7:0]
AMR2[7:0]	AMR3[7:0]

03 CAN 发送帧格式

- 发送帧分成 ID 部分和数据部分。
- 第一个字节包含了帧分类等信息，确定是 SFF（标准）帧还是EFF（扩展）帧；确定是远程帧，还是数据帧；以及确定数据长度。
- SFF 的 ID 长度为 2 个字节。
- EFF 的 ID 长度为 4 个字节。
- 数据长度为最大 8 个字节。

SFF		EFF	
地址	域	地址	域
0x40	TX帧信息	0x40	TX帧信息
0x44	TX ID0	0x44	TX ID0
0x48	TX ID1	0x48	TX ID1
0x4C	TX DATA0	0x4C	TX ID2
0x50	TX DATA1	0x50	TX ID3
0x54	TX DATA2	0x54	TX DATA0
0x58	TX DATA3	0x58	TX DATA1
0x5C	TX DATA4	0x5C	TX DATA2
0x60	TX DATA5	0x60	TX DATA3
0x64	TX DATA6	0x64	TX DATA4
0x68	TX DATA7	0x68	TX DATA5
0x6C	unused	0x6C	TX DATA6
0x70	unused	0x70	TX DATA7

CAN 发送帧结构图

03 CAN 接收帧格式

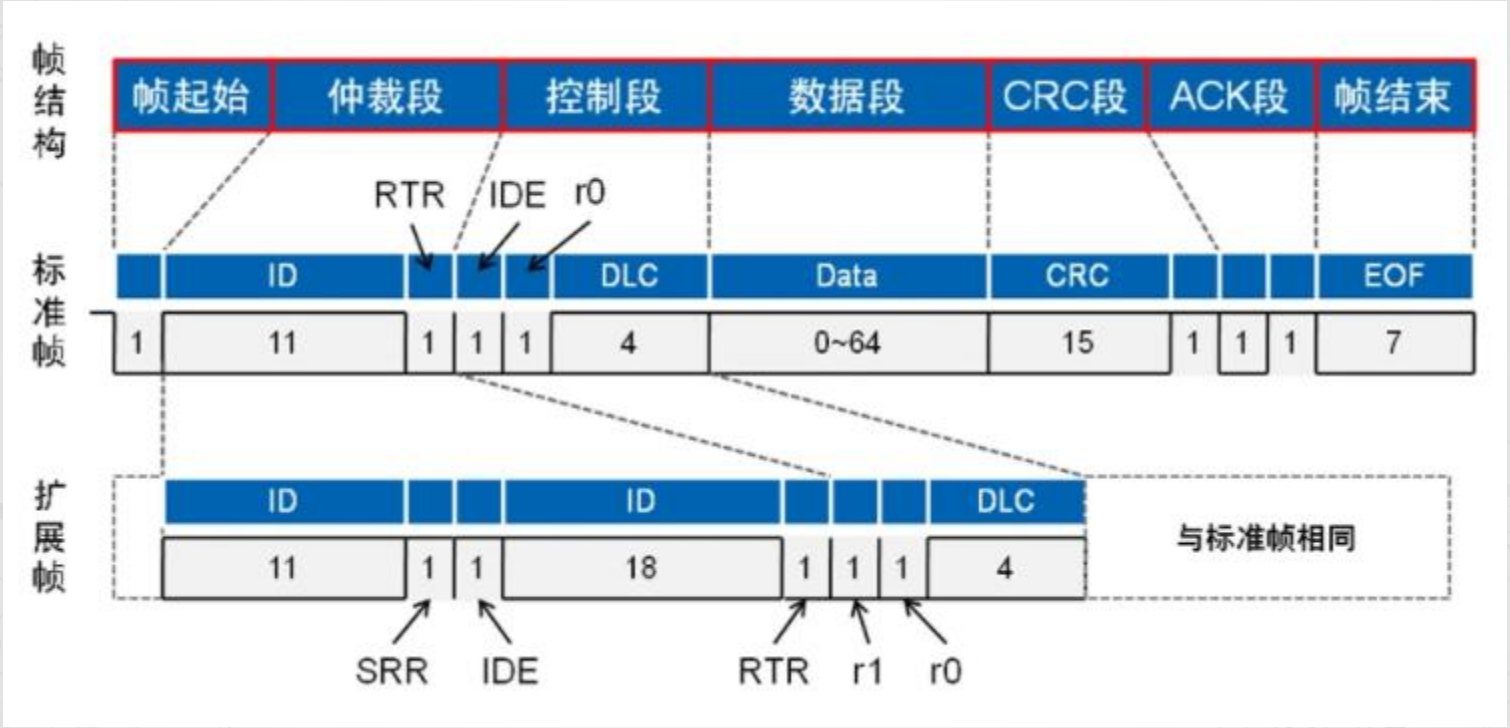
- 接收帧分成 ID 部分和数据部分。
- 第一个字节包含了帧分类等信息，确定是 SFF（标准）帧还是EFF（扩展）帧；确定是远程帧，还是数据帧；以及确定数据长度。
- SFF 的 ID 长度为 2 个字节。
- EFF 的 ID 长度为 4 个字节。
- 数据长度为最大 8 个字节。

SFF		EFF	
地址	域	地址	域
0x40	RX帧信息	0x40	RX帧信息
0x44	RX ID0	0x44	RX ID0
0x48	RX ID1	0x48	RX ID1
0x4C	RX DATA0	0x4C	RX ID2
0x50	RX DATA1	0x50	RX ID3
0x54	RX DATA2	0x54	RX DATA0
0x58	RX DATA3	0x58	RX DATA1
0x5C	RX DATA4	0x5C	RX DATA2
0x60	RX DATA5	0x60	RX DATA3
0x64	RX DATA6	0x64	RX DATA4
0x68	RX DATA7	0x68	RX DATA5
0x6C	unused	0x6C	RX DATA6
0x70	unused	0x70	RX DATA7

CAN 接收帧结构图

03

CAN 帧头信息


帧结构
帧起始
仲裁段
控制段
数据段
CRC段
ACK段
帧结束
RTR
IDE
r0
标准帧
ID
DLC
Data
CRC
EOF
1
11
1
1
1
4
0~64
15
1
1
1
7
扩展帧
ID
ID
DLC
11
1
1
18
1
1
1
4
与标准帧相同
SRR
IDE
RTR
r1
r0

03 CAN 帧头信息

- 发送或接收帧分成 ID 部分和数据部分。第一个字节包含了帧分类等信息，确定是 SFF（标准）帧还是EFF（扩展）帧；确定是远程帧，还是数据帧；以及，确定数据长度。SFF 的 ID 长度为 2 个字节，EFF 的 ID 长度为 4 个字节。数据长度为最大 8 个字节。
- FF（IDE）：1 表示是 EFF（扩展）帧，0 表示是 SFF（标准）帧。
- RTR：1 标识是 remote（远程）帧，0 表示是 data（数据）帧。
- SRR：替代远程请求位。
- DLC：表示此帧要发送数据的长度。最大为 8 个字节，最小为 0 个字节。
- ID：帧标识号。SFF 帧的 ID 长度为 11 位（ID.28 到 ID.18）。EFF 帧的 ID 长度为 29 位（ID.28到 ID.0）。高位优先发送。
- DATA：数据。字节间顺序，从大到小，即 TX DATA7 先发送。字节内顺序，从高位到低位。
- X2：最好同 RTR 值一致。
- X1：1 或 0 均可。

03

CAN 帧头信息

发送 SFF 头信息

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	X1	X1	DLC.3	DLC.2	DLC.1	DLC.0
0x44	DLC.28	DLC.27	DLC.26	DLC.25	DLC.24	DLC.23	DLC.22	DLC.21
0x48	DLC.20	DLC.19	DLC.18	X2	X1	X1	X1	X1

发送 SFF 头信息

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	X1	X1	DLC.3	DLC.2	DLC.1	DLC.0
0x44	DLC.28	DLC.27	DLC.26	DLC.25	DLC.24	DLC.23	DLC.22	DLC.21
0x48	DLC.20	DLC.19	DLC.18	DLC.17	DLC.16	DLC.15	DLC.14	DLC.13
0x4C	DLC.12	DLC.11	DLC.10	DLC.9	DLC.8	DLC.7	DLC.6	DLC.5
0x50	DLC.4	DLC.3	DLC.2	DLC.1	DLC.0	X2	X1	X1

03 CAN 数据发送

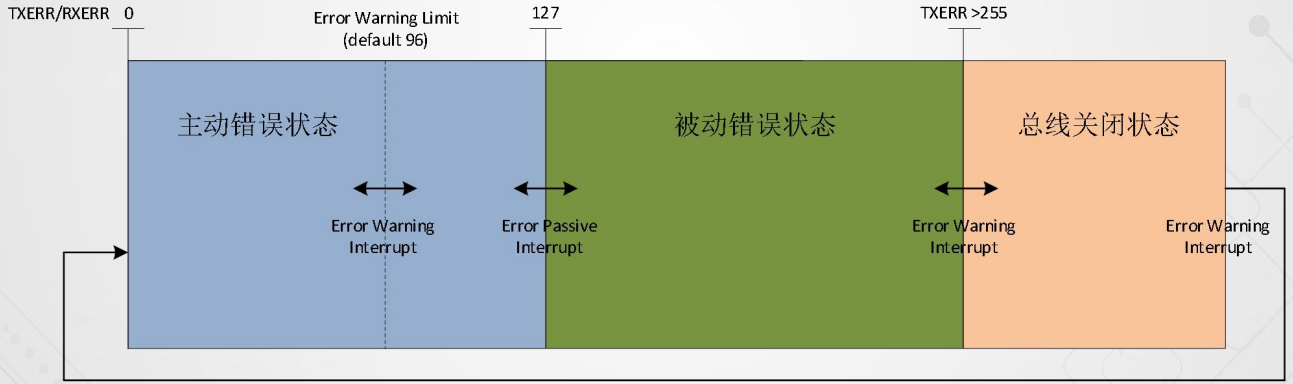
- CAN 模块要发送数据，必须在正常工作模式下。
- 执行发送操作，一般推荐开启发送中断源和错误警告数量中断源，而仲裁丢失和总线错误中断源头不强迫开启，CAN 模块有自动重发机制。
- 发送的数据，存放在 TX FIFO 中，TX FIFO 为 32-Byte，及一次可以存几帧数据，且 TX FIFO 中的数据一旦被成功发送出去，就被 CAN 模块释放掉供新的数据写入。一帧数据长度为 13 个字节，其考虑到最大数据量为 8 个字节。在发送之前，需确保此时模块处于空闲状态。
- 发送帧的格式，分成标准帧（SFF）和扩展帧（EFF）。CAN_CMR 的 BIT0 置 1，触发 CAN 模块发送数据。一旦 CAN 总线空闲，数据就会发出。传输完毕与否，一方面可以检查 CAN_SR 的 BIT2是否变成 1（1 表示TFIFO 空闲），也可以通过中断方式检查是否发送完毕。
- 在数据发送到总线前，设置 CAN_CMR 的 BIT1，可以停止发送，产生相应中断；通过 CAN_SR的 BIT3，可知当前数据是否发送完毕。

03 CAN 数据接收

- CAN 模块要接收数据，必须在正常工作模式下。
- 接收 FIFO 中存储的帧大小可以不同（3 ~ 13 byte 范围之间）。当接 FIFO 为满时（或剩余的空间不足以完全存储一帧），将触发溢出中断，后续的接收帧将丢失，直到接收 FIFO 中清除出足够的存储空间。接收 FIFO 中的第一条帧将被映射到 13-byte 的**接收缓冲器**中，直到该帧被清除（通过释放接收缓冲器指令）。清除后，接收缓冲器将继续映射接收 FIFO 中的下一条帧，接收 FIFO 中上一条已清除帧的空间将被释放。
- 接收，开始于侦测到 Start 帧。接收到的数据，执行 ID 匹配操作：
- 匹配成功，存入 RX FIFO，CAN_RMC 递增。RX FIFO 为 32-Byte，即一次可以存几帧数据，且 RX FIFO中的数据一旦被 DMA 成功取走，就被 CAN 模块释放掉供新的数据写入（非 DMA 模式，需要对CAN_CMR BIT2 写 1 释放 RX FIFO）。一帧数据长度为 13 个字节，其考虑到最大数据量为 8 个字节。RX FIFO 中有几个有效帧，通过 CAN_AMC 寄存器可知。通过读取 CAN_TXRX 寄存器可以获得最先被接收到的数据帧。若 RX FIFO 满了，抛弃此帧，RX FIFO 不接收，同时产生溢出中断（使能前提下）。
- 匹配失败，存入 RX FIFO，RMC 不会递增。因此，数据也会被后续帧覆盖。若 RX FIFO 满了，抛弃此帧，RX FIFO 不接收，不产生溢出中断（使能前提下）。

03 CAN 错误管理

CAN 协议要求每个节点中都包含发送错误计数和接收错误计数。这两个错误计数的数值决定了控制器当前的错误状态（如主动错误、被动错误、离线）。控制器将数值分别存储在CAN_RXERR 和 CAN_TXERR 中，MCU 可随时进行读取。除了错误状态之外，控制器还提供错误报警限制的功能，这个功能可在 CAN 控制器进入被动错误状态之前，提醒用户，当前发生的严重总线错误。



CAN 模块错误管理

03 CAN 离线状态与离线恢复

当发送错误的数值大于 255 时，CAN 控制器将进入离线状态。进入离线状态后，CAN 控制器将自动进行以下动作：

- 接收错误计数器的数值置为 0
- 发送错误计数器的数值置为 127
- CAN_IR.BUS_ERR_IF 位置 1
- 进入 CAN 的复位模式

离线恢复：

为了返回主动错误状态，CAN 控制器必须进行离线恢复。要启动离线恢复，首先需要退出复位模式，进入操作模式。然后要求 CAN 控制器在总线上检测到 **128 次 11 个连续隐性位**。每一次 CAN控制器检测到 11 个连续隐性位时，发送错误计数器的数值都将减小，以追踪离线恢复进程。当离线恢复完成后（发送错误的数值从 127 减小到 0），CAN_IR.BUS_ERR_IF 位将自动复位为 0，从而触发错误报警中断。

03 CRC模块

- 内置硬件CRC校验，可以再单周期内完成一次CRC-8，CRC16或CRC32运算，支持用户自定义CRC多项式、初值、输入数据反转等。
- 触发RESET清空CRC数据。
- 一般计算过程如下（以CRC16/IBM为例）

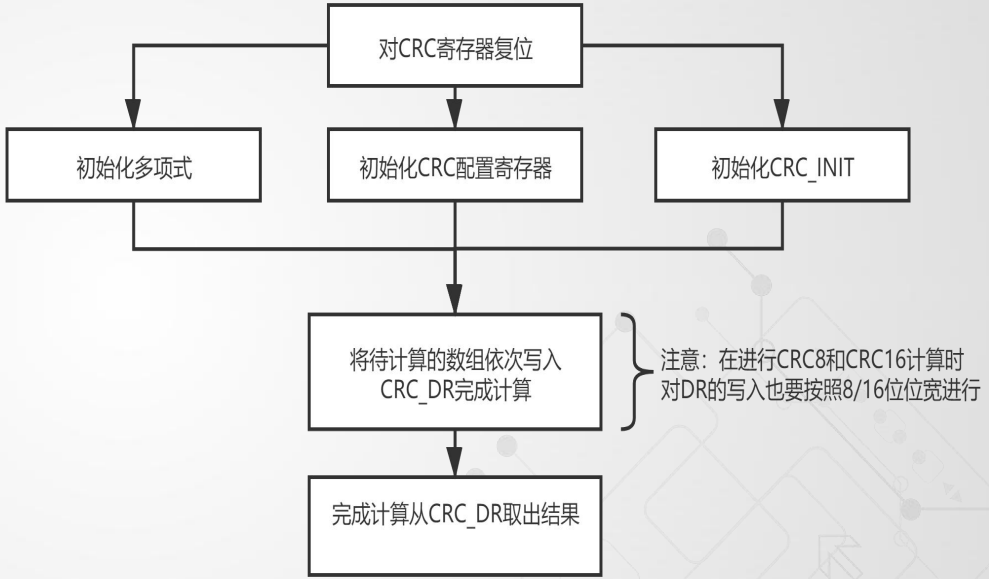
写入CRC多项式0x8005到CRC_POL

REV_IN_TYPE反转类型设置为半字反转

输出位宽设置为16写入初始值0x0000到CRC_INIT

将待转换的数据按半字方式依次写入CRC_DR

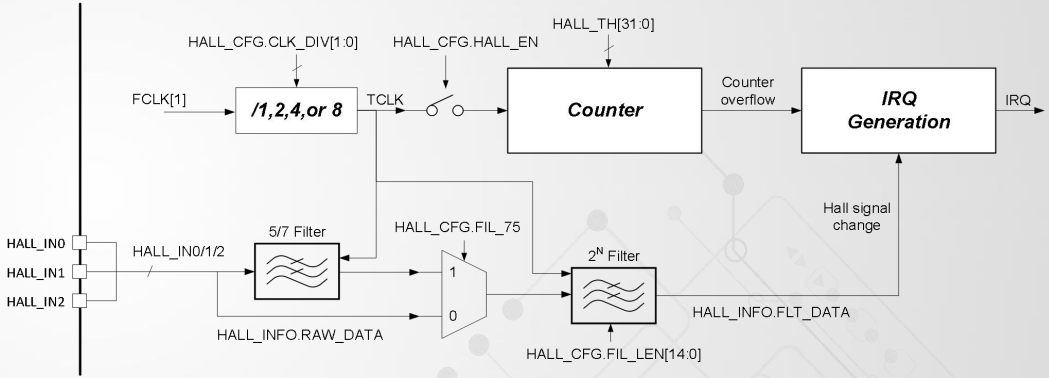
从CRC_DR取出数据，完成一组CRC计算



CRC 软件设计框图

03 HALL模块

- 芯片共支持 3 路 HALL 信号输入。
- 对于输入的 HALL 传感器信号，所进行的处理包括：
- 滤波，消除 HALL 信号毛刺的影响。
- 捕获，当 HALL 输入有变化时，记录当前的定时器值，并输出中断。
- 溢出，当 HALL 信号长时间不发生变化导致计数器溢出时，输出中断。
- HALL模块工作频率可调。通过配置HALL_CFG.CLK_DIV寄存器，可以选择系统主时钟的1/2/4/8分频作为 HALL 模块工作频率，滤波和计数均采用该频率工作。



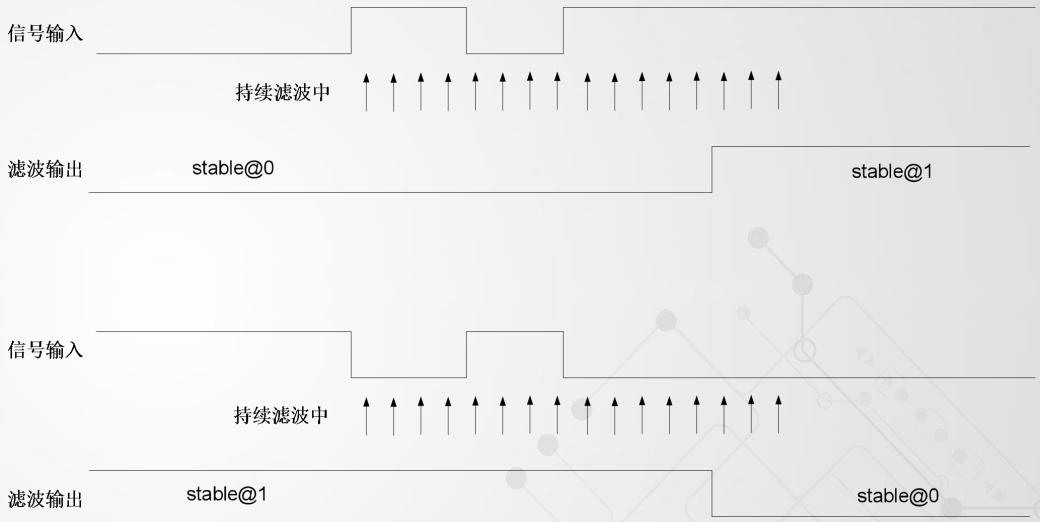
HALL 数据流程框图

03 HALL输入信号滤波

- 滤波包括两级滤波器：
- 第一级采用 7 判 5 进行滤波，即连续 7 个采样点中，如果达到超过 5 个 1 则输出 1，如果达到或超过 5 个 0 则输出 0，否则输出保持上一次的滤波结果。 7 判 5 滤波器如右图所示。
- 第二级采用连续滤波，在连续 N 个采样点中，如全为 0 则输出 0，如全为 1 则输出 1，否则输出保持上一次的滤波结果。滤波时间常数计算公式如下：

Tfit = Tclk * (HALL_CFG.FIL_LEN[14:0] + 1)

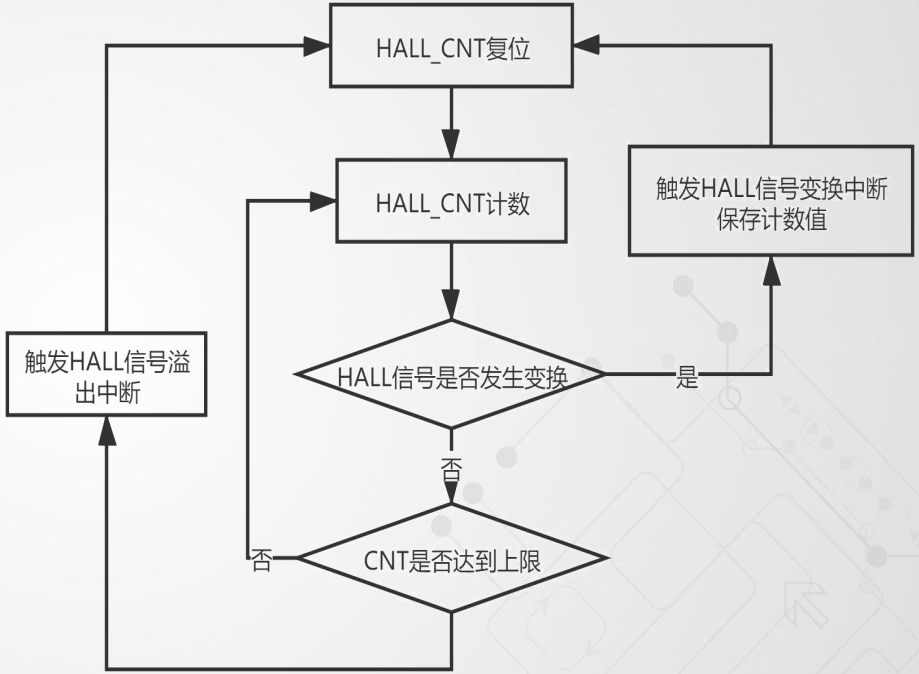
（Tclk为HALL模块时钟周期）。



7/5 滤波模块框图

03 HALL捕获及溢出

- 捕获模块用于测量两次 HALL 信号变化之间的时间，其核心为一个 24 位计数器，在主时钟 96MHz 工作频率下，HALL时钟选择主时钟8分频，此时最大可以记录约 1.39 秒的时间宽度，最高时间分辨率约为 10ns 。
- HALL_CNT 从 0 开始计数，当发生 HALL 信号变化时，将此时刻HALL_CNT 值保存到 HALL_WIDTH 寄存器，将此时刻的 HALL 信号保存到 HALL_INFO.FIL_DATA，输出 HALL 信号变化中断，HALL_CNT 重新从 0 开始计数。
- 当计数器计数值达到 HALL_TH 时，输出 HALL 计数器溢出中断，计数器重新从 0 开始计数。



HALL 软件实现流程图

03 08x中断

- 外设共有21个中断源。
- M0内核有5个中断源如下：

Reset_Handler 复位向量

HardFault_Handler 硬件错误中断

SVC_Handler SVC异常中断

PendSV_Handler 可挂起系统中断

SysTick_Handler 系统滴答定时中断

中断号	中断事件来源	中断号	中断事件来源
0	TIMER0	16	WAKEUP，系统唤醒中断
1	TIMER1	17	电源电压过低
2	TIMER2	18	DMA
3	TIMER3	19	CAN
4	ENCODER0	20	SIF
5	ENCODER1	21	Reserved
6	I2C	22	Reserved
7	GPIO	23	Reserved
8	UART0	24	Reserved
9	HALL	25	Reserved
10	SPI	26	Reserved
11	ADC	27	Reserved
12	DSP	28	Reserved
13	MCPWM	29	Reserved
14	UART1	30	Reserved
15	CMP	31	Reserved

03 看门狗

- 08x看门狗工作于低速 RC 时钟域 LSI，即使用 32kHz 进行计数。支持 2s、4s、8s、64s 四档复位时间可选。
- 复位控制寄存器 SYS_RST_CFG.WDT_EN 可用于使能或禁用看门狗，SYS_RST_CFG.WDT_EN=1 则使能看门狗模块。复位源记录寄存器 SYS_RST_SRC.WDT_RST_RCD 记录了看门狗复位事件，SYS_RST_SRC.WDT_RST_RCD 为高表示发生过看门狗复位。
- 看门狗复位是硬件全局复位，其作用域等同于外部引脚复位以及内部的上电复位。
- 看门狗清零前，需要向 SYS_WR_PROTECT 写入 0xCAFE，开启 WatchDog SYS_WDT_CLR 寄存器写入。

03 IO复用

- LKS08x 可实现 SWDIO和SWDCLK 复用为其它 IO 的功能，SWDIO复用的 IO 是 P0.0、P2.15，SWDCLK 复用的 IO P2.6。注意事项如下：
- 默认状态是不开启复用，需要软件开启复用。即芯片硬复位结束后，初始状态是 SWDIO 和SWDCLK用途，SWDIO和SWDCLK 在芯片内部有上拉（芯片内部上拉电阻约为 10K），应用对初始电平有要求的，需注意。
- LKS08x 可实现 RSTN 复用为其它 IO 的功能，复用的 IO 是 P0.2。

03 其他特点

- 电机控制类应用定制开发，集成度高、节约BOM成本；
- 丰富的模拟运放和比较器资源，可满足单电阻/双电阻/三电阻电流采样拓扑架构的不同需求；
- 内部集成高压钳位网络，允许高压共模信号直接输入，轻松实现MOSFET内阻直接电流采样，省去电流采样电阻并降低了系统功耗；
- 独特的Mosfet内阻温度漂移补偿电路，确保电流采样的精确度；
- 变增益控制技术兼顾高速和低速各种工况下的采样精度；
- -40℃ ~ 125℃工作温度范围，抗干扰能力强，稳定可靠；
- 单电源+2.2V ~ +5.5V供电，确保了系统供电的通用性;
- 适用于有感FOC/无感FOC/BLDC/两相步进电机等控制系统;
- 成熟的常见类型电机控制算法，提供商用级应用方案支持。

03

LKS网站资源

实时更新

- 各型号芯片资源表
- User_Manual用户使用手册，Datasheet芯片数据手册，勘误表
- IAR
  需要安装IAR插件：IAR环境配置.rar，KEIL器件库
- 各DEMO板原理图和PCB
- LKS离线烧录工具，DSP用户使用说明，应用笔记
- LKS32MC0XX_BSP各模块例程与配套例程说明文档

凌鸥官方网站地址：https://www.linkosemi.com

凌鸥WiKi地址：https://linkosemi.wiki.zoho.com.cn

為天地立芯
為控制塑魂

创芯驱动，领航电控未来！

正直诚信！利他共赢！成长超越！

江苏省南京市经济技术开发区兴智科技园B栋15层
http://www.linkosemi.com