USER

MANUAL

32 位 RSIC 架构的微控制器

TS32F020 系列

用户手册

V1.0.6

修订历史记录

变更类型：A - 增加　M - 修订　D - 删除

变更版本号	日期	变更类型	修改人	审核	摘要
V0.0.1	2018/8/1		liuhao		Initial version.
V0.0.2	2019/1/30	M	SZQ、LQJ		Modified version.
V1.0.0	2019/3/19	M	LD		Modified chip name
V1.0.1	2019/6/12	M	WHY		PC0、PC1 pin function
V1.0.2	2019/6/19	M	LHT		Modified version.
V1.0.3	2019/10/9	M	LWB		Pin map Package information
V1.0.4	2019/11/4	A&M	LHT		Modified LED ADDR specification Add TK_CAP Pin Define
V1.0.5	2020/1/13	M	LHT		GPIO-PB/C speed fixed
V1.0.6	2020/4/29	M	LHT		Modified page header

目录

1 TS32F020 简介

- 内核与系统
  - 32-Bit RISC 架构的 MCU
  - 工作最大主频：52MHz
  - 单周期 32 位乘法指令
  - 32 个中断源，可配置 4 层中断优先级
  - 支持位带操作
  - 支持两线 DBG 调试接口
- 存储器
  - 高达 32K 字节的闪存程序存储器
  - 高达 3K 字节的 SRAM
  - Boot loader 支持片内 Flash、支持单/双 pin UART 在线用户编程（IAP）/ 在线系统编程（ISP）
- 时钟、复位和电源管理
  - 2.7V ~ 5.5V 供电
  - 上电/断电复位（POR/PDR）、可编程电压监测器（PVD）
  - 外部 1MHz ~ 26MHz 晶体振荡器
  - 内嵌经出厂调校的 26MHz(+/- 1.5%) 高速振荡器
  - 内嵌 32KHz 低速振荡器
  - PLL 支持 CPU 最高运行在 52MHz
  - 内置时钟安全系统（CSS）
  - WDT 复位
- DMA 支持
  - 支持的外设： EFLASH，UART, SPI/IIC ,CRC，TK，LED driver 和 ADC
- 多达 38 个快速 I/O 端口
  - 所有 I/O 口可以触发边沿或电平响应中断，唤醒低功耗模式
  - 所有端口均可输入输出 5V 信号
- 通讯接口外设
  - 2 个 SPI 高速串行接口,最高支持 26Mbit/s，支持 1/2/4 线主从模式，支持 IIC 模式
  - 2 个 UART 接口，支持 RS232/RS485 协议
- LED 显示控制器
  - 支持共阴、共阳方式推 LED 屏
  - LED 自动扫描控制
  - 段码显示最大支持 8*12
- 人机交互输入
  - Key (GPIO)
  - Touch Key 最多支持 20 Keys
  - 支持低功耗触摸按键唤醒
- 8 个定时器
  - 5 个 16 位定时器,1 个 32 位定时器，每个定时器支持 1 个 IC/OC，可组合用于 IR 控制编解码
  - 1 个看门狗定时器

- 1 个系统时间定时器：24 位自减型计数器

- 高安全性
  - 支持 AES 加解密程序，防止程序被盗
  - 支持 5/7/8/16/32 bit CRC 校验，保证数据准确性
- 低功耗
  - 支持 IDLE,STOP,SLEEP 低功耗模式
- 1 个 12 位高速模数转换器
  - 支持最高 150K 采样率
  - 多达 26 个输入通道
  - 转换范围： 0 ~ V_{AVCC}
- 2 个比较器
  - 支持 4 个正端输入，4 个负端输入。
  - 比较器 1 和 12 位 ADC、12 位 DAC 分时复用
- 1 个运算放大器
  - 内置失配校正检测电路，校正后失配 1mV
  - 运放正端接 3 个 IO，一个内部 DAC 输出电压；运放负端输入接 4 个 IO
- 1 个 6bit 精度 DAC
- 1 个 12bit 精度 DAC
- 内置温度传感器
- 高可靠性
  - ESD HBM 8KV
  - EFT ±4KV
  - Latch-up ±100mA @105℃
- 96 位的芯片唯一 ID（UID）
- 封装
  - SOP20/SOP24/SOP28
  - SSOP20/SSOP24/SSOP28
  - LQFP48
- 工业级温度范围

  -40 ºC ~ 105ºC

2 终端配置与功能

2.1 管脚图

（SSOP20/SSOP24/SSOP28 和 SOP20/SOP24/SOP28 引脚排布和功能定义是同样的）

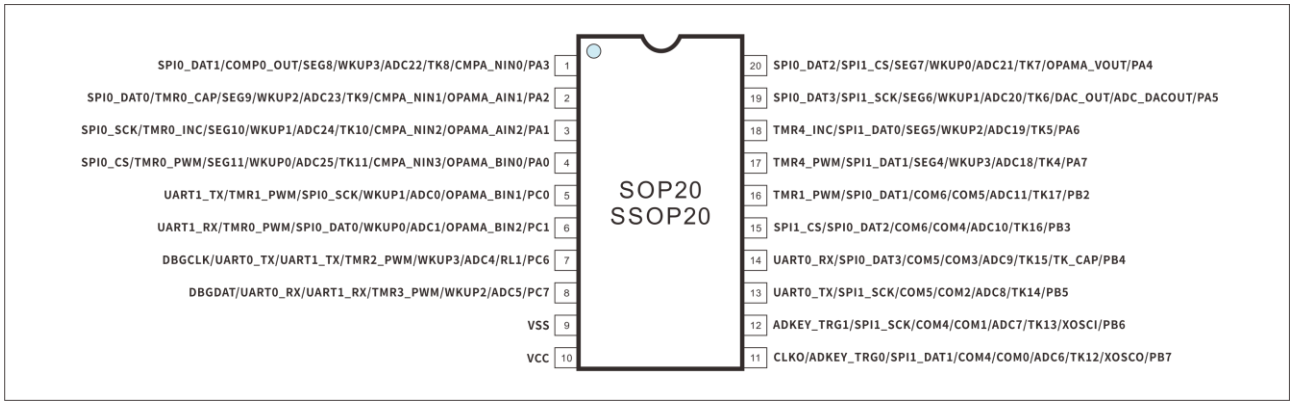


Figure 2-1 TS32F020 SOP20/SSOP20 package

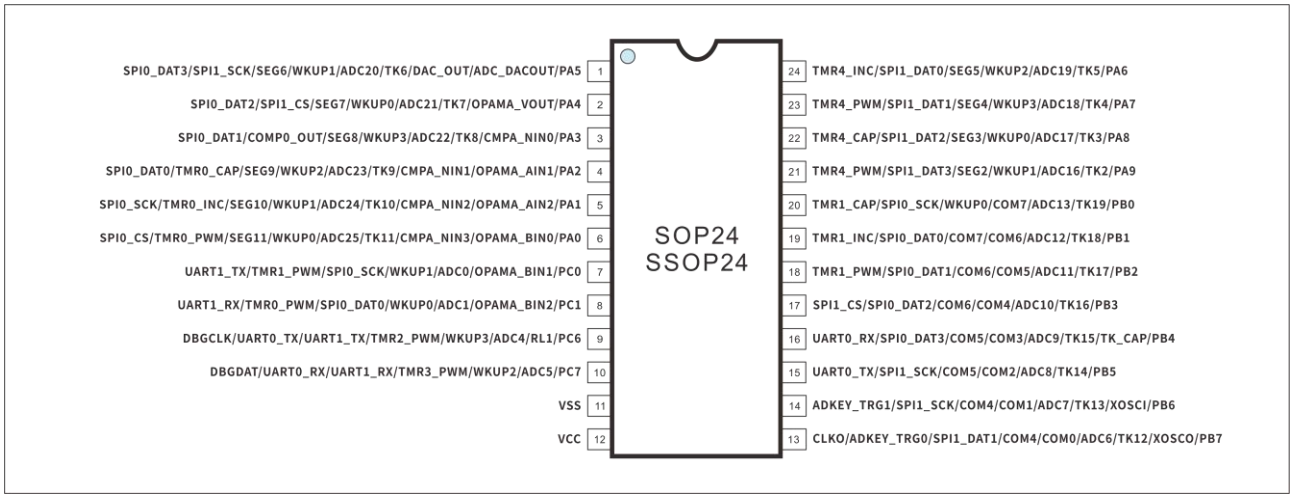


Figure 2-2 TS32F020 SOP24/SSOP24 package

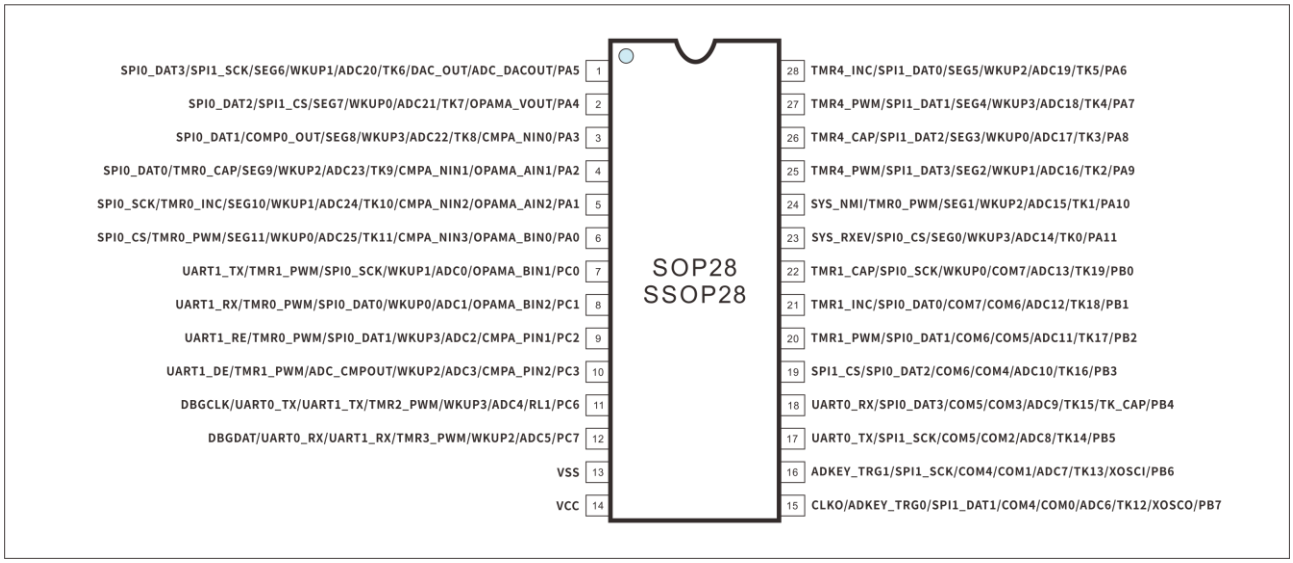


Figure 2-3 TS32F020 SOP28/SSOP28 package

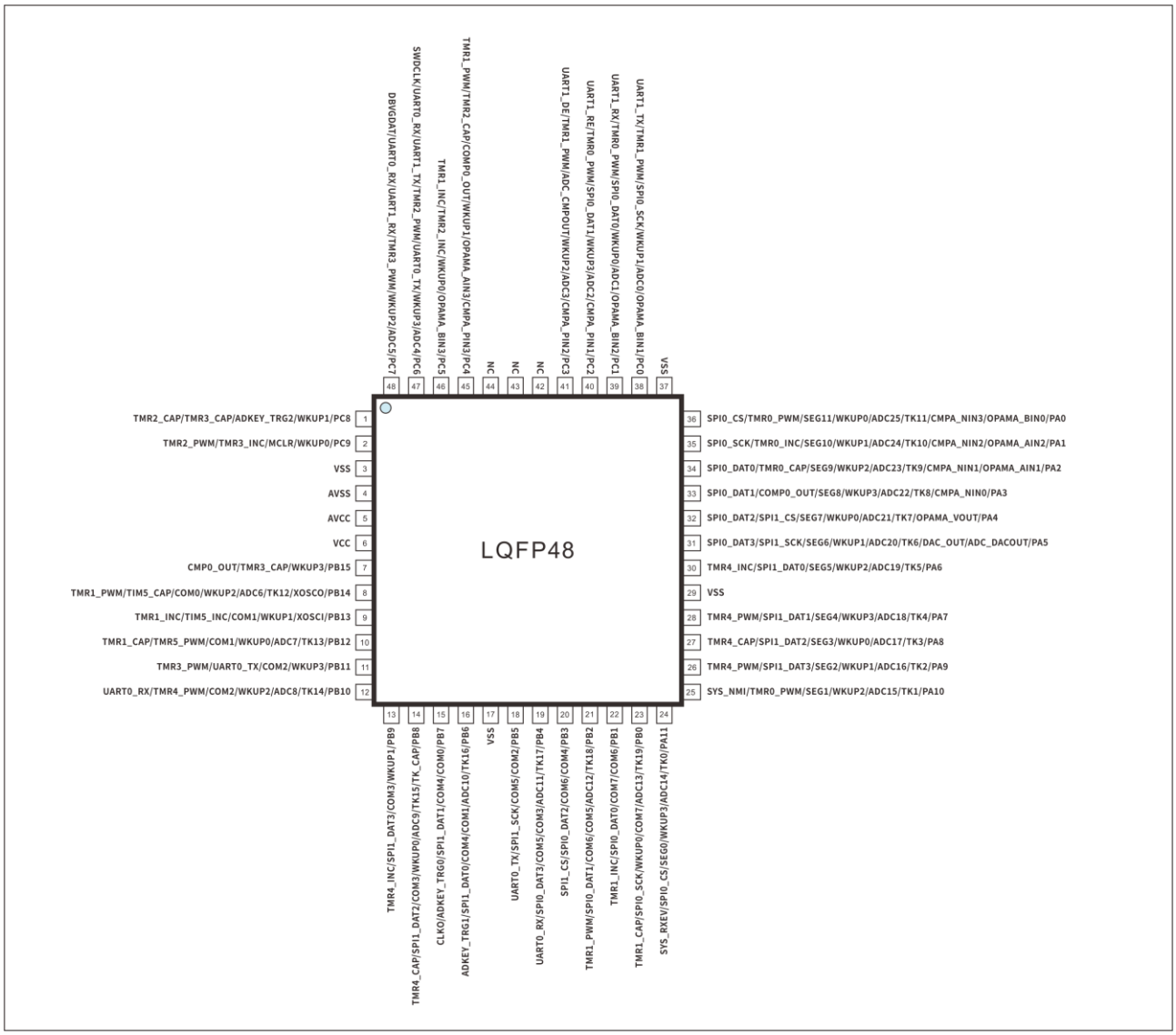


Figure 2-4 TS32F020 LQFP48 package

2.2 管脚定义

（SSOP20/SSOP24/SSOP28 和 SOP20/SOP24/SOP28 引脚排布和功能定义是同样的）

LQF48 引脚定义

Pin number	Pin name (function after reset)	Pin type	I/O structure	Notes	Pin functions	
LQFP48					Alternate Functions	Additional Functions
1	PC8	I/O	IO0	-	TMR2_CAP, TMR3_CAP, ADKEY_TRG2, WKUP1	-

2	PC9	I/O	IO0	-	TMR2_PWM, TMR3_INC, MCLR, WKUP0	-
3	AVSS	S	-	-	-	-
4	VSS	S	-	-	-	-
5	AVCC	S	-	-	-	-
6	VCC	S	-	-	-	-
7	PB15	I/O	IO0	-	CMP0_OUT, TMR3_CAP, COM0, WKUP3	-
8	PB14	I/O	IO0	-	TMR1_PWM, TIM5_CAP, COM0, WKUP2	ADC6, TK12 XOSCO
9	PB13	I/O	IO0	-	TMR1_INC, TIM5_INC, COM1, WKUP1	XOSCI
10	PB12	I/O	IO0	-	TMR1_CAP, TMR5_PWM, COM1, WKUP0	ADC7, TK13
11	PB11	I/O	IO0	-	TMR3_PWM, UART0_TX, COM2, WKUP3	-
12	PB10	I/O	IO0	-	UART0_RX, TMR4_PWM, COM2, WKUP2	ADC8, TK14
13	PB9	I/O	IO0	-	TMR4_INC,	-

					SPI1_DAT3, COM3, WKUP1	
14	PB8	I/O	IO0	-	TMR4_CAP, SPI1_DAT2, COM3, WKUP0	ADC9, TK15 TK_CAP:电荷转移方式下该引脚需接电容
15	PB7	I/O	IO0	-	CLKO, ADKEY_TRG0, SPI1_DAT1, COM4, COM0	-
16	PB6	I/O	IO0	-	ADKEY_TRG1, SPI1_DAT0, COM4, COM1	ADC10 TK16
17	VSS	S	-	-	-	-
18	PB5	I/O	IO0	-	UART0_TX, SPI1_SCK, COM5, COM2	-
19	PB4	I/O	IO0	-	UART0_RX, SPI0_DAT3, COM5, COM3	ADC11, TK17
20	PB3	I/O	IO0	-	SPI1_CS, SPI0_DAT2, COM6, COM4	-
21	PB2	I/O	IO0	-	TMR1_PWM, SPI0_DAT1, COM6, COM5	ADC12, TK18

22	PB1	I/O	IO0	-	TMR1_INC, SPI0_DAT0, COM7, COM6	-
23	PB0	I/O	IO0	-	TMR1_CAP, SPI0_SCK, WKUP0, COM7	ADC13, TK19
24	PA11	I/O	IO1	-	SYS_RXEV, SPI0_CS, SEG0, WKUP3	ADC14, TK0
25	PA10	I/O	IO1	-	SYS_NMI, TMR0_PWM, SEG1, WKUP2	ADC15, TK1
26	PA9	I/O	IO1	-	TMR4_PWM, SPI1_DAT3, SEG2, WKUP1	ADC16, TK2
27	PA8	I/O	IO1	-	TMR4_CAP, SPI1_DAT2, SEG3, WKUP0	ADC17, TK3
28	PA7	I/O	IO1	-	TMR4_PWM, SPI1_DAT1, SEG4, WKUP3	ADC18, TK4
29	VSS	S	-	-	-	-
30	PA6	I/O	IO1	-	TMR4_INC, SPI1_DAT0, SEG5, WKUP2	ADC19, TK5
31	PA5	I/O	IO1	-	SPI0_DAT3,	ADC20,

					SPI1_SCK, SEG6, WKUP1	TK6, DAC_OUT, ADC_DACOUT
32	PA4	I/O	IO1	-	SPI0_DAT2, SPI1_CS, SEG7, WKUP0	ADC21, TK7, OPAMA_VOUT
33	PA3	I/O	IO1	-	SPI0_DAT1, COMP0_OUT, SEG8, WKUP3	ADC22, TK8, CMPA_NIN0
34	PA2	I/O	IO1	-	SPI0_DAT0, TMR0_CAP, SEG9, WKUP2	ADC23, TK9, CMPA_NIN1, OPAMA_AIN1
35	PA1	I/O	IO1	-	SPI0_SCK, TMR0_INC, SEG10, WKUP1	ADC24, TK10, CMPA_NIN2, OPAMA_AIN2
36	PA0	I/O	IO1	-	SPI0_CS, TMR0_PWM, SEG11, WKUP0	ADC25, TK11, CMPA_NIN3, OPAMA_BIN0
37	VSS	S	-	-	-	-
38	PC0	I/O	IO0	-	UART1_TX, TMR1_PWM, SPI0_SCK, WKUP1	ADC0, OPAMA_BIN1
39	PC1	I/O	IO0	-	UART1_RX, TMR0_PWM, SPI0_DAT0, WKUP0	ADC1, OPAMA_BIN2
40	PC2	I/O	IO0	-	UART1_RE, TMR0_PWM,	ADC2, CMPA_PIN1

					SPI0_DAT1, WKUP3	
41	PC3	I/O	IO0	-	UART1_DE, TMR1_PWM, ADC_CMPOUT, WKUP2	ADC3, CMPA_PIN2
45	PC4	I/O	IO0	-	TMR1_PWM, TMR2_CAP, COMP0_OUT, WKUP1	OPAMA_AIN3, CMPA_PIN3
46	PC5	I/O	IO0	-	TMR1_INC, TMR2_INC, WKUP0	OPAMA_BIN3
47	PC6	I/O	IO0	-	DBGCLK, UART0_TX UART1_TX, TMR2_PWM, WKUP3	ADC4
48	PC7	I/O	IO0	(1)	DBGDAT, UART0_RX, UART1_RX, TMR3_PWM, WKUP2	ADC5

SOP28、SOP24、SOP20 引脚定义（SSOP 和 SOP 相同）

Pin Number			Pin name (function after reset)	Pin type	I/O structure	Notes	Pin functions	
SOP28	SOP24	SOP20					Alternate Functions	Additional Functions
1	1	19	PA5	I/O	IO1	-	SPI0_DAT3, SPI1_SCK, SEG6, WKUP1	ADC20, TK6, DAC_OUT, ADC_DACOUT

2	2	20	PA4	I/O	IO1	-	SPI0_DAT2, SPI1_CS, SEG7, WKUP0	ADC21, TK7, OPAMA_VOUT
3	3	1	PA3	I/O	IO1	-	SPI0_DAT1, COMP0_OUT, SEG8, WKUP3	ADC22, TK8, CMPA_NIN0
4	4	2	PA2	I/O	IO1	-	SPI0_DAT0, TMR0_CAP, SEG9, WKUP2	ADC23, TK9, CMPA_NIN1, OPAMA_AIN1
5	5	3	PA1	I/O	IO1	-	SPI0_SCK, TMR0_INC, SEG10, WKUP1	ADC24, TK10, CMPA_NIN2, OPAMA_AIN2
6	6	4	PA0	I/O	IO1	-	SPI0_CS, TMR0_PWM, SEG11, WKUP0	ADC25, TK11, CMPA_NIN3, OPAMA_BIN0
7	7	5	PC0	I/O	IO0	-	UART1_TX, TMR1_PWM, SPI0_SCK, WKUP1	ADC0, OPAMA_BIN1
8	8	6	PC1	I/O	IO0	-	UART1_RX, TMR0_PWM, SPI0_DAT0, WKUP0	ADC1, OPAMA_BIN2
9	-	-	PC2	I/O	IO0	-	UART1_RE, TMR0_PWM, SPI0_DAT1, WKUP3	ADC2, CMPA_PIN1
10	-	-	PC3	I/O	IO0	-	UART1_DE, TMR1_PWM, ADC_CMPOUT, WKUP2	ADC3, CMPA_PIN2
11	9	7	PC6	I/O	IO0	-	DBGCLK, UART0_TX UART1_TX, TMR2_PWM, WKUP3	ADC4, OPAMA_BIN3, OPAMA_AIN3, CMPA_PIN3

12	10	8	PC7	I/O	IO0	(1)	DBGDAT, UART0_RX, UART1_RX, TMR3_PWM, WKUP2	ADC5
13	11	9	AVSS	S	-	-	-	-
			VSS	S	-	-	-	-
14	12	10	VCC	S	-	-	-	-
			AVCC	S	-	-	-	-
15	13	11	PB7	I/O	IO0	-	CLKO, ADKEY_TRG0, SPI1_DAT1, COM4, COM0	ADC6, TK12, XOSCO
16	14	12	PB6	I/O	IO0	-	ADKEY_TRG1, PI1_DAT0, COM4, COM1	ADC7, TK13, XOSCI
17	15	13	PB5	I/O	IO0	-	UART0_TX, SPI1_SCK, COM5, COM2	ADC8, TK14
18	16	14	PB4	I/O	IO0	-	UART0_RX, SPI0_DAT3, COM5, COM3	ADC9, TK15 TK_CAP:电荷转移方式下该引脚需接电容
19	17	15	PB3	I/O	IO0	-	SPI1_CS, SPI0_DAT2, COM6, COM4	ADC10, TK16
20	18	16	PB2	I/O	IO0	-	TMR1_PWM, SPI0_DAT1, COM6, COM5	ADC11, TK17
21	19	-	PB1	I/O	IO0	-	TMR1_INC, SPI0_DAT0, COM7, COM6	ADC12, TK18
22	20	-	PB0	I/O	IO0	-	TMR1_CAP, SPI0_SCK, WKUP0, COM7	ADC13, TK19

23	-	-	PA11	I/O	IO1	-	SYS_RXEV, SPI0_CS, SEG0, WKUP3	ADC14, TK0
24	-	-	PA10	I/O	IO1	-	SYS_NMI, TMR0_PWM, SEG1, WKUP2	ADC15, TK1
25	21	-	PA9	I/O	IO1	-	TMR4_PWM, SPI1_DAT3, SEG2, WKUP1	ADC16, TK2
26	22	-	PA8	I/O	IO1	-	TMR4_CAP, SPI1_DAT2, SEG3, WKUP0	ADC17, TK3
27	23	17	PA7	I/O	IO1	-	TMR4_PWM, SPI1_DAT1, SEG4, WKUP3	ADC18, TK4
28	24	18	PA6	I/O	IO1	-	TMR4_INC, SPI1_DAT0, SEG5, WKUP2	ADC19, TK5

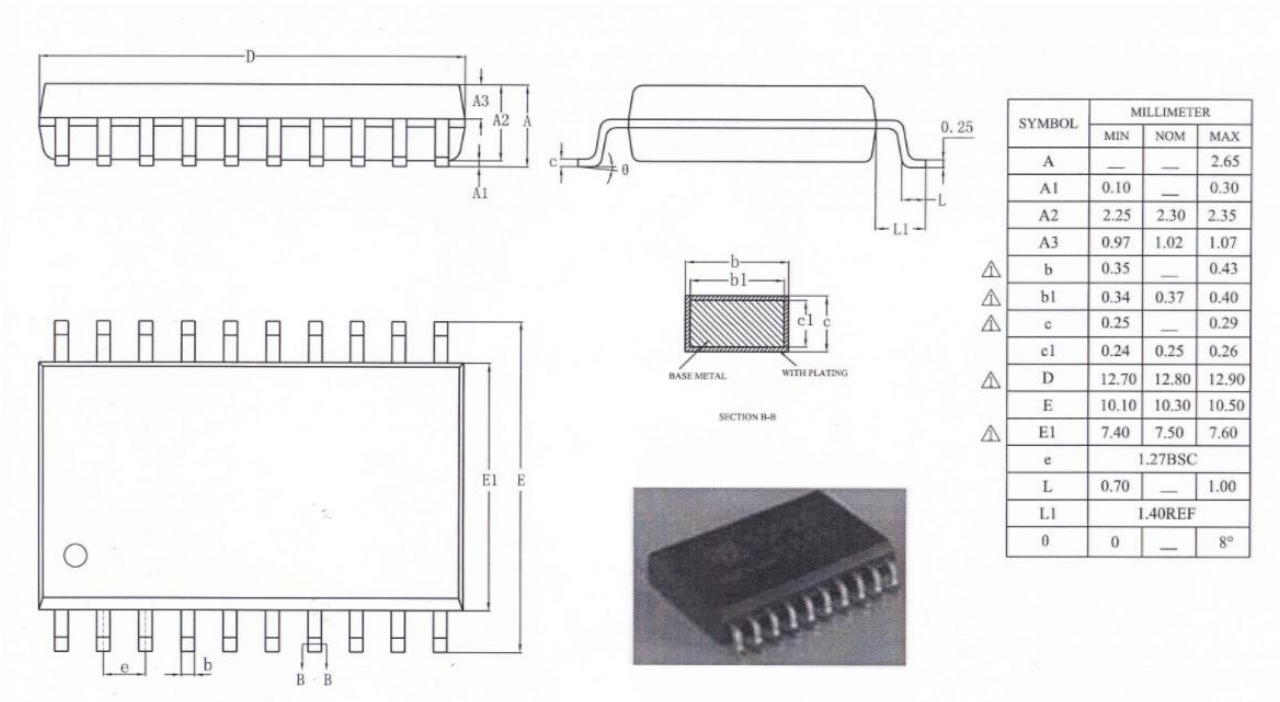
Table. Alternate function AF0 to AF3

Port		AF0	AF1	AF2	AF3
PA	PA0	SPI0_CS	TMR0_PWM	SEG11	WKUP0
	PA1	SPI0_SCK	TMR0_INC	SEG10	WKUP1
	PA2	SPI0_MOSI	TMR0_CAP	SEG9	WKUP2
	PA3	SPI0_MISO	COMP_OUT	SEG8	WKUP3
	PA4	SPI0_DAT2	SPI1_CS	SEG7	WKUP0
	PA5	SPI0_DAT3	SPI1_SCK	SEG6	WKUP1
	PA6	TMR4_INC	SPI1_MOSI	SEG5	WKUP2
	PA7	TMR4_PWM	SPI1_MISO	SEG4	WKUP3
	PA8	TMR4_CAP	SPI1_DAT2	SEG3	WKUP0
	PA9	TMR4_PWM	SPI1_DAT3	SEG2	WKUP1
	PA10	SYS_NMI	TMR0_PWM	SEG1	WKUP2
	PA11	SYS_RXEV	SPI0_CS	SEG0	WKUP3
PB	PB0	TMR1_CAP	SPI0_SCK	COM7	WKUP0
	PB1	TMR1_INC	SPI0_MOSI	COM7	COM6
	PB2	TMR1_PWM	SPI0_MISO	COM6	COM5
	PB3	SPI1_CS	SPI0_DAT2	COM6	COM4
	PB4	UART0_RX	SPI0_DAT3	COM5	COM3
	PB5	UART0_TX	SPI1_SCK	COM5	COM2
	PB6	ADKEY_TRG1	SPI1_MOSI	COM4	COM1

	PB7	ADKEY_TRG0	SPI1_MISO	COM4	COM0
	PB8	TMR4_CAP	SPI1_DAT2	COM3	WKUP0
	PB9	TMR4_INC	SPI1_DAT3	COM3	WKUP1
	PB10	TMR4_PWM	UART0_RX	COM2	WKUP2
	PB11	TMR3_PWM	UART0_TX	COM2	WKUP3
	PB12	TMR1_CAP	TMR5_PWM	COM1	WKUP0
	PB13	TMR1_INC	TMR5_INC	COM1	WKUP1
	PB14	TMR1_PWM	TMR5_CAP	COM0	WKUP2
	PB15	COMP_OUT	TMR3_CAP	COM0	WKUP3
PC	PC0	UART1_TX	TMR1_PWM	SPI0_SCK	WKUP1
	PC1	UART1_RX	TMR0_PWM	SPI0_MOSI	WKUP0
	PC2	UART1_RE	TMR0_PWM	SPI0_MISO	WKUP3
	PC3	UART1_DE	TMR1_PWM	ADKEY_COMPOUT	WKUP2
	PC4	TMR1_PWM	TMR2_CAP	COMP_OUT	WKUP1
	PC5	TMR1_INC	TMR2_INC	-	WKUP0
	PC6	UART1_TX	TMR2_PWM	UART0_TX	WKUP3
	PC7	UART1_RX	TMR3_PWM	UART0_RX	WKUP2
	PC8	TMR2_CAP	TMR3_CAP	ADKEY_TRG2	WKUP1
	PC9	TMR2_PWM	TMR3_INC	-	WKUP0

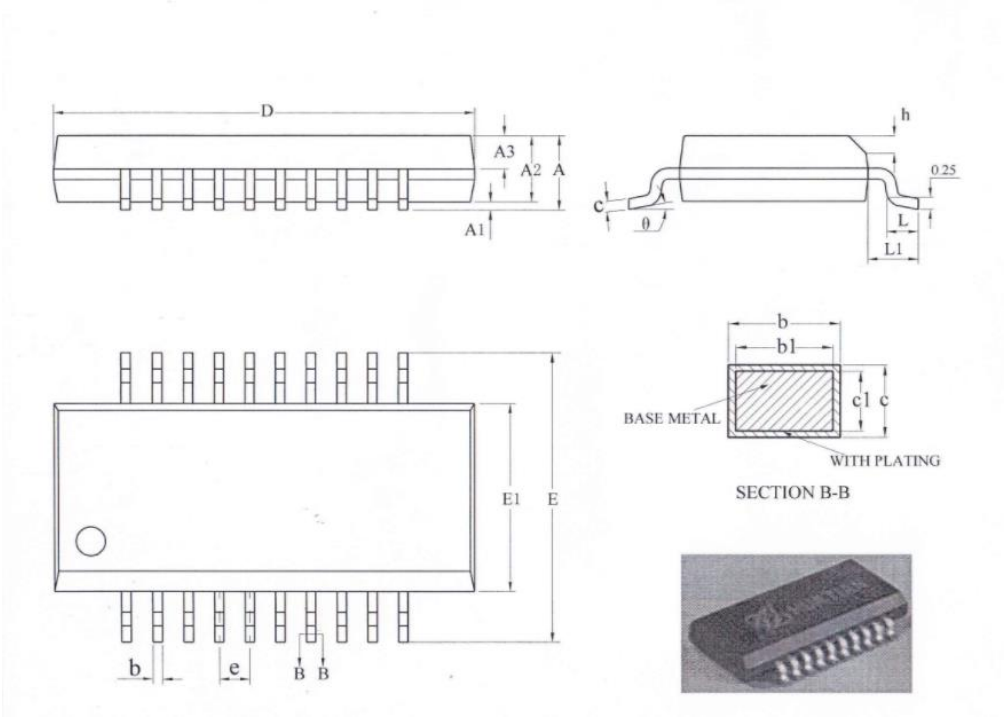
2.3 封装信息

2.3.1 **SOP20**



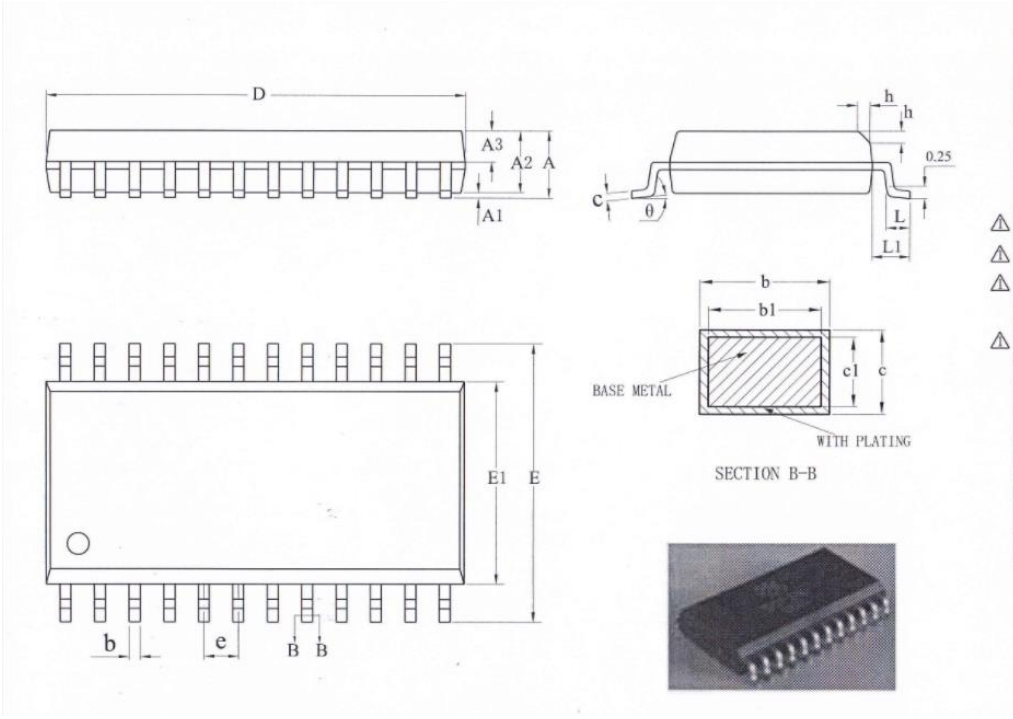
SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	2.65
A1	0.10	—	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
⚠ b	0.35	—	0.43
⚠ b1	0.34	0.37	0.40
⚠ c	0.25	—	0.29
c1	0.24	0.25	0.26
⚠ D	12.70	12.80	12.90
E	10.10	10.30	10.50
⚠ E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	—	1.00
L1	1.40REF		
θ	0	—	8°

2.3.2 **SSOP20**



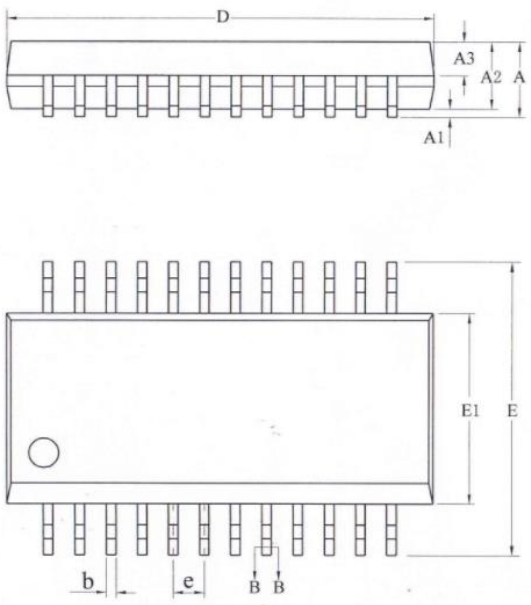
SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.75
A1	0.10	0.15	0.25
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.23	—	0.31
b1	0.22	0.25	0.28
c	0.20	—	0.24
c1	0.19	0.20	0.21
D	8.55	8.65	8.75
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	0.635BSC		
h	0.30	—	0.50
L	0.50	—	0.80
L1	1.05REF		
θ	0	—	8°

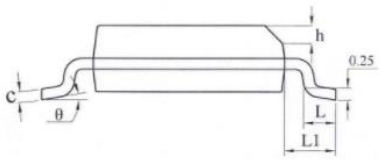
2.3.3 **SOP24**

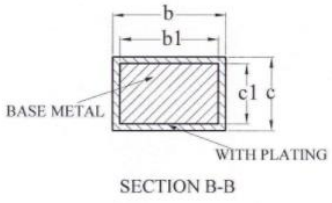


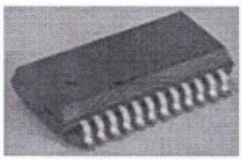
SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	2.36	2.54	2.64
A1	0.10	0.20	0.30
A2	2.26	2.30	2.35
A3	0.97	1.02	1.07
⚠ b	0.39	—	0.47
⚠ b1	0.38	0.41	0.44
⚠ c	0.25	—	0.29
c1	0.24	0.25	0.26
⚠ D	15.30	15.40	15.50
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	—	1.00
L1	1.40REF		
h	0.25	—	0.75
θ	0	—	8°

2.3.4 **SSOP24**



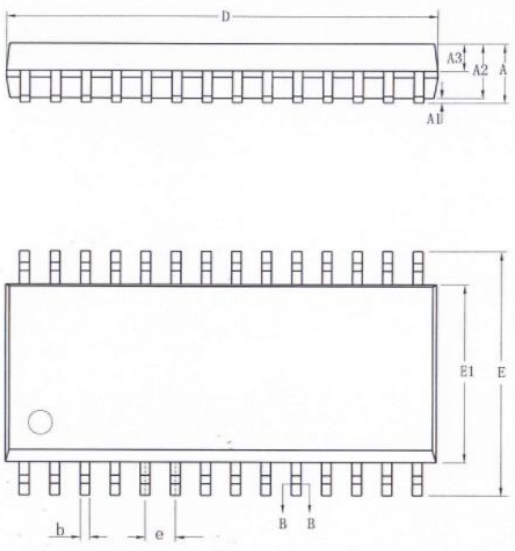


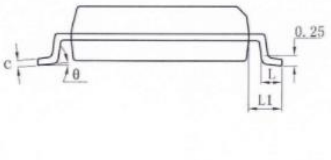


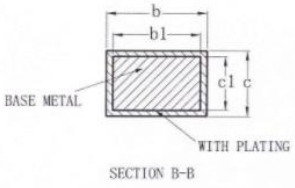


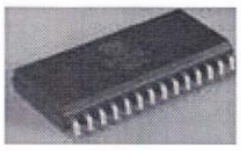
SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.75
A1	0.10	0.15	0.25
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.23	—	0.31
b1	0.22	0.25	0.28
c	0.20	—	0.24
c1	0.19	0.20	0.21
D	8.55	8.65	8.75
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	0.635BSC		
h	0.30	—	0.50
L	0.50	—	0.80
L1	1.05REF		
θ	0	—	8°

2.3.5 **SOP28**



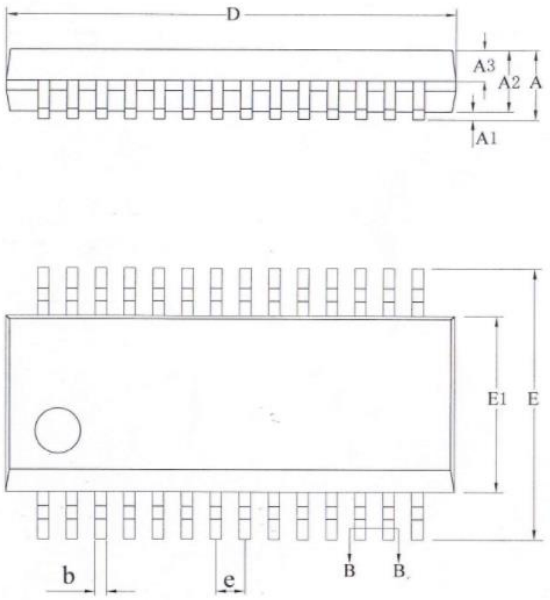


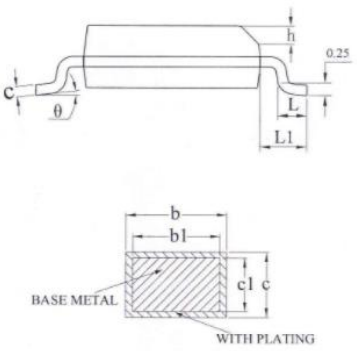


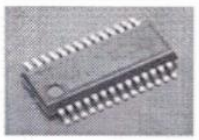


	SYMBOL	MILLIMETER		
		MIN	NOM	MAX
	A	—	—	2.65
	A1	0.10	—	0.30
	A2	2.25	2.30	2.35
	A3	0.97	1.02	1.07
⚠	b	0.39	—	0.47
⚠	b1	0.38	0.41	0.44
⚠	c	0.25	—	0.29
	c1	0.24	0.25	0.26
⚠	D	17.90	18.00	18.10
	E	10.10	10.30	10.50
⚠	E1	7.40	7.50	7.60
	e	1.27BSC		
	L	0.70	—	1.00
	L1	1.40REF		
	θ	0	—	8°

2.3.6 **SSOP28**

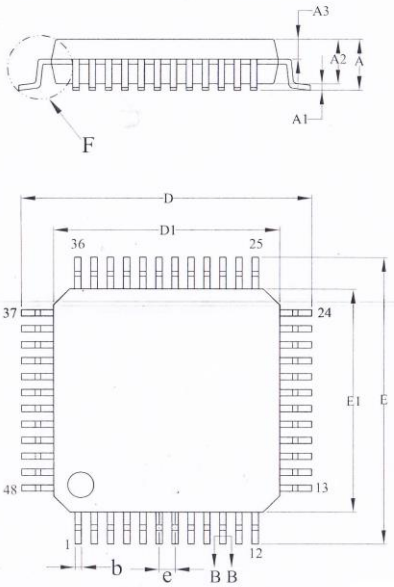


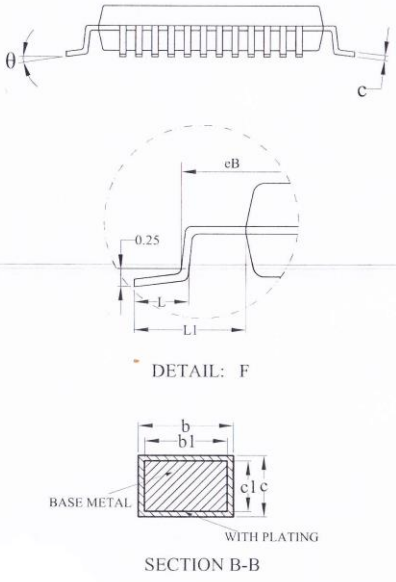




SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.75
A1	0.05	—	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.23	—	0.31
b1	0.22	0.25	0.28
c	0.20	—	0.24
c1	0.19	0.20	0.21
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	0.635BSC		
h	0.25	—	0.50
L	0.50	—	0.80
L1	1.05BSC		
θ	0°	—	8°

2.3.7 **LQFP48**





SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.60
A1	0.05	—	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.18	—	0.26
b1	0.17	0.20	0.23
c	0.13	—	0.17
c1	0.12	0.13	0.14
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
eB	8.10	—	8.25
e	0.50BSC		
L	0.45	—	0.75
L1	1.00REF		
θ	0	—	7

3 系统及存储器架构

TS32F020 器件是基于 RISC 的 32 位处理器，采用了哈佛结构。

3.1 系统架构

TS32F020 主系统由以下两部分构成：

- 二个驱动单元：
  - CPU 内核系统总线（S-bus）
  - DMA 总线
- 三个被动单元
  - 内部闪存存储器
  - 内部 SRAM
  - AHB 到 APB 的桥（AHB2APB0/1），它连接所有的 APB 设备

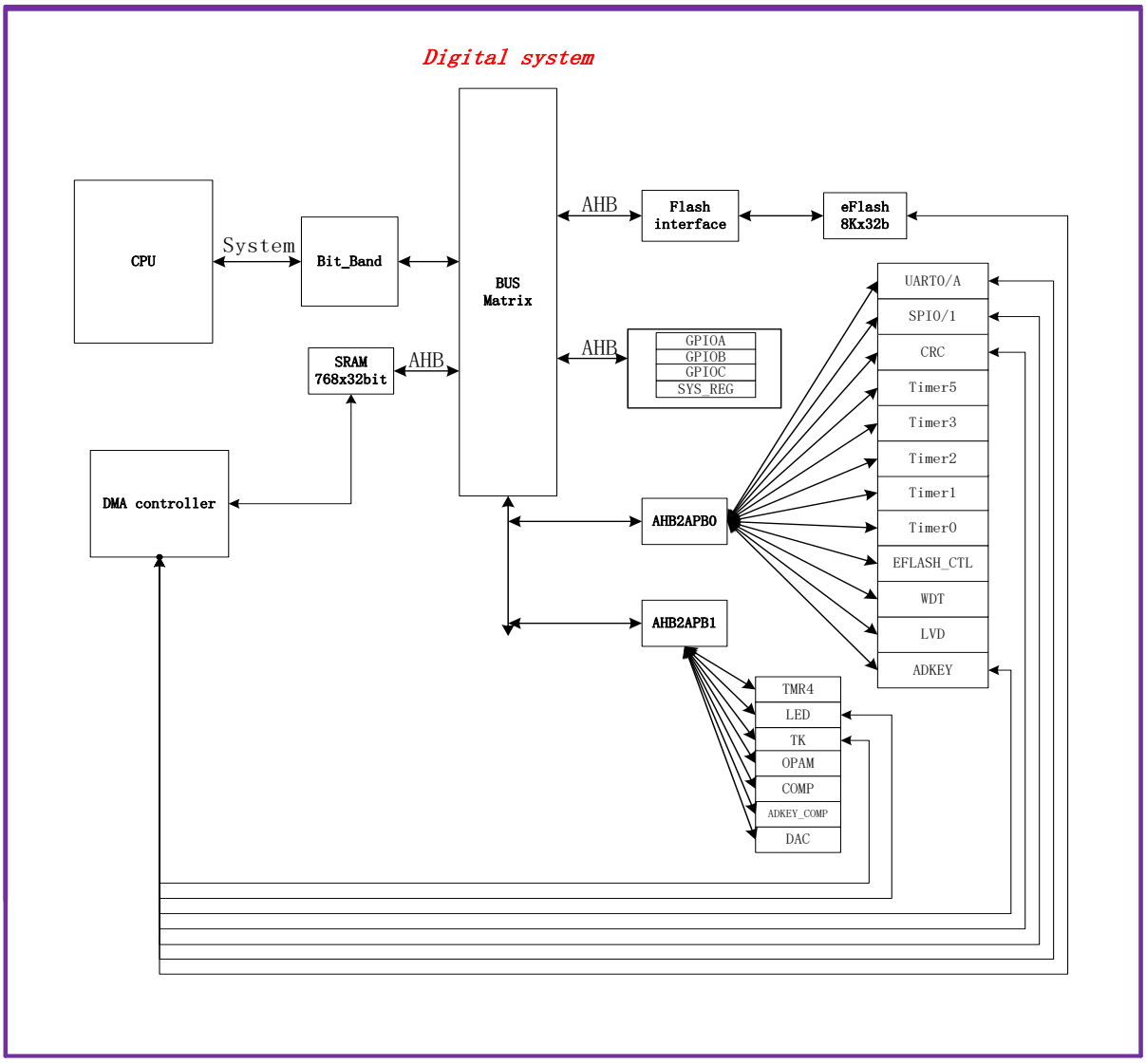


图 3-1 TS32F020 总线系统架构图

系统总线

此总线连接 CPU 内核的系统总线（外设总线）到总线矩阵，总线矩阵协调着内核和各个高速部件间的访问。

DMA 控制器

此总线将 CPU 与各外设模块访问相联竞争，协调访问优先级，仲裁等。

Module	Priority	
CPU	0	Highest
SPI0	1	
SPI1	2	
UART0	3	
ADKEY	4	
UART1	5	
CRC	6	
TOUCH KEY	7	
LED	8	
Eflash	9	Lowest

表 3-1 DMA 优先级

总线矩阵（BusMatrix）

总线矩阵管理着内核系统总线与各外设模块的访问仲裁，总线矩阵由主模块总线及从模块总线组成。

AHB 外设通过总线矩阵与系统总线相连。

AHB 到 APB 桥（AHB2APB bridges-APB）。

AHB 到 APB 桥在 AHB 与 APB 总线间提供同步连接。

注：当对 APB 寄存器进行 8 位或者 16 位访问时，该访问会被自动转换成 32 位的访问：桥会自动将 16 位或者 8 位的数据扩展以配合 32 位的宽度。

3.2 存储器映射

此 32 位 RISC 处理器采用同一套总线来读取指令和加载/存储数据。指令代码和数据都位于相同的存储器地址空间，但在不同的地址范围。程序存储器，数据存储器，寄存器和 I/O 端口都在同一个线性的 4 GB 的地址空间之内。这是 32 位 RISC 的最大地址范围，因为它的地址总线宽度是 32 位。此外，为了降低不同客户在相同应用时的软件复杂度，存储映射是按 32 位 RISC 处理器提供的规则预先定义的。在存储器映射表中，一部分地址空间由 32 位 RISC 的系统外设所占用，且不可更改。此外，其余部分地址空间可由芯片供应商定义使用。***表 3-2***. TS32F020 器件的存储器映射表显示了 TS32F020 器件的存储器映射，包括代码、SRAM、外设和其他预先定义的区域。简化了每个外设的地址译码。

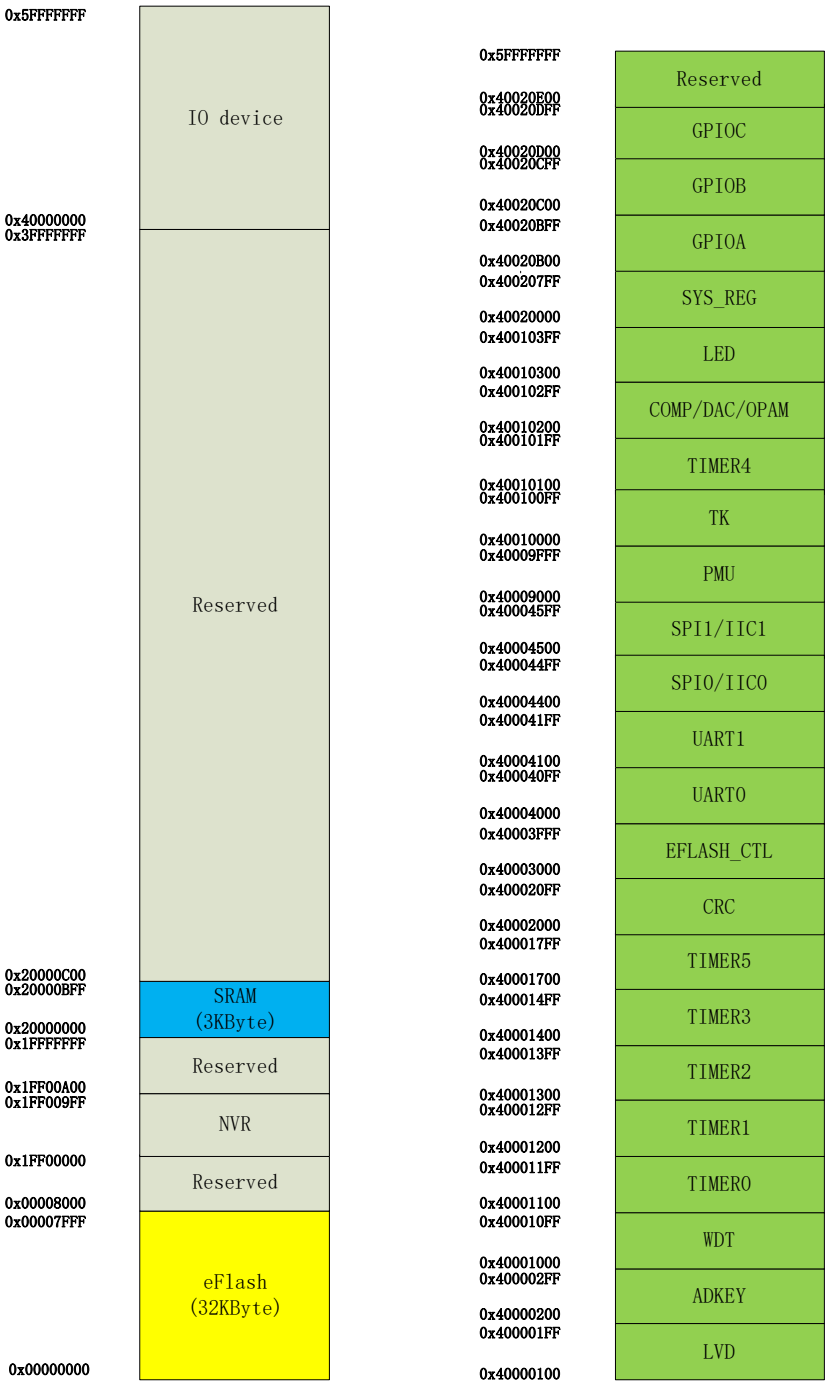


表 3-2 TS32F020 存储器映射表

3.2.1 位带操作

为了减少“读-改-写”操作的次数，32 位 RISC 处理器提供了一个可以执行单原子比特操作的位带功能。存储器映射包含了两个支持位带操作的区域。其中一个是 SRAM 区的最低 1MB 范围，第二个是片内外设区的最低 1MB 范围。这两个区域中的地址除了普通应用外，还有自己的“位带别名区”。位带别名区把每个比特扩展成一个 32 位的字。当用户访问位带别名区时，就可以达到访问原始比特的目的。

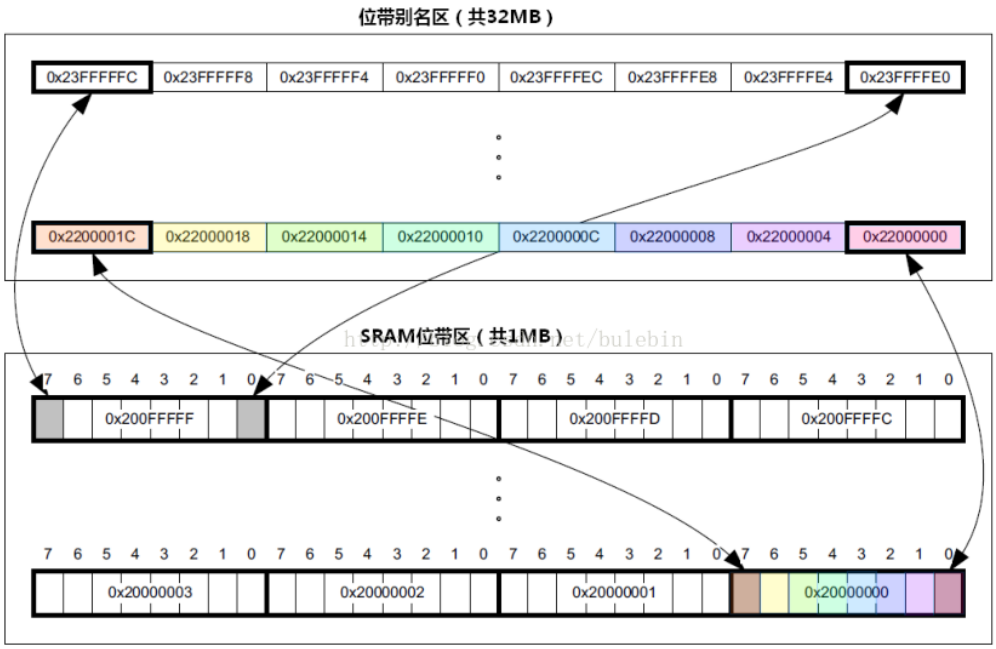


图 3-2 SRAM 区中的最低 1MB 的位带区与位带别名区的扩展对应关系

下面的公式表明了位带别名区中的每个字如何对应位带区的相应比特或目标比特。

bit_word_addr =bit_band_base +(byte_offset×32)+(bit_number×4) (式 3-1)

说明：“*4”表示一个字为 4 个字节，“*32”表示每个地址有 8 个比特位，8*4=32。

其中：

- bit_word_addr 指的是位带区目标比特对应在位带别名区的地址；
- bit_band_base 指的是位带别名区的起始地址；
- byte_offset 指的是位带区目标比特所在的字节的字节地址偏移量；
- bit_number 指的是目标比特在对应字节中的位置(0-7)。

例如，要想访问 0x2000 0200 地址的第 7 位，可访问的位带别名区地址是：

bit_word_addr = 0x2200 0000 + (0x200 * 32)+ (7 * 4)= 0x2200 401C (式 3-2)

如果对 0x2200 401C 进行写操作，那么 0x2000 0200 的第 7 位将会相应变化；如果对 0x2200 401C 进行读操作，那么视 0x2000 0200 的第 7 位状态而返回 0x01 或 0x00。

3.2.2 片上 SRAM

TS32F020 内置最大可到 3K 字节的静态 SRAM。它可以以字节（8 位）、半字（16 位）或字（32 位）进行访问。SRAM 起始地址为 0x2000 0000。数据总线上最大可到 3K 字节的 SRAM。可以被 CPU 或者 DMA 用最快的系统时钟且不插入任何等待进行访问。

DMA 支持访问外 TK,ADKEY,CRC,SPI0/1,UART0/1,LED,EFLASH。

3.2.3 片上 FLASH 概述

闪存存储器有两个不同存储区域：

- 主闪存存储块，它包括应用程序和用户数据区（若需要时）
- 信息块，其包含两个部分：

- 选项字节（Option bytes）—内含硬件及存储保护用户配置选项。
- 系统存储器（System memory）—其包含 Boot loader 代码。参见内置闪存存储器章节。

闪存接口基于 AHB 协议执行指令和数据存取。其预取缓冲的功能可加速 CPU 执行代码的速度。

3.2.4 引导配置

器件复位后，通过客户自己在主闪存存储块的配置，配置 PC6 默认工作是上拉还是拉下。在启动延

迟之后， CPU 从地址 0x0000 0000 获取堆栈顶的地址，并从启动存储器的 0x0000 0004 指示的地址开始执行代码。

内嵌的自举程序存放在系统存储器，由厂家在生产时写入。该程序可以通过 UART0 对闪存进行重新编程。

3.3 引脚复位（MCLR）功能

默认状态下 TS32F020 是不支持 MCLR 复位功能的。MCLR 指在某个特定 IO 引脚上输入一个持续 1.7ms 以上的低电平导致系统复位，如同重新上电复位一样。当用户通过 eflash 中用户自定义 bit 来使能 MCLR 功能后。PC9 会变成 MCLR pin。即默认上拉，往 PC9 上输入一个 1.7ms 以上低电平，导致 MCLR 复位。default 状态下 PC9 是 GPIO。既不上拉也没有下拉。详细情况请参考闪存存储器中关于用户自定义区域的描述。

4 嵌入式闪存

4.1 闪存主要特性

- 高达 32K 字节闪存存储器
- 存储器结构：
  - 主闪存空间：32K 字节
  - 副闪存空间（系统存储器）：2K 字节
- 带预取缓冲器的读接口
- 闪存编程和擦除操作
- 访问和写保护
- 低功耗模式

4.2 闪存功能描述

4.2.1 闪存结构

闪存空间由 32 位宽的存储单元组成，既可以存代码又可以存数据。主闪存块按 32 页（每页 1K 字节）分块，以页为单位设置写保护（参见存储保护相关内容）。

模块	名称	地址	大小（字节）
主闪存空间	页 0	0x0000_0000 – 0x0000_03FF	1K
	页 1	0x0000_0400 – 0x0000_07FF	1K
	页 2	0x0000_0800 – 0x0000_0BFF	1K
	页 3	0x0000_0C00 – 0x0000_0FFF	1K
	…	…	1K
	页 30	0x0000_7800 – 0x0000_7BFF	1K
	页 31	0x0000_7C00 – 0x0000_7DFF	512
	用户配置区	0x0000_7E00 – 0x0000_7FFF	512
副闪存空间	扇区 0	0x1FF0_0000 – 0x1FF0_01FF	512
	扇区 1	0x1FF0_0200 – 0x1FF0_03FF	512
	扇区 2	0x1FF0_0400 – 0x1FF0_05FF	512
	芯片信息区	0x1FF0_0600 – 0x1FF0_06FF	256
闪存寄存器接口	CTRLR0	0x4000_3000 – 0x4000_3003	4
	KST	0x4000_3004 – 0x4000_3007	4
	DONE	0x4000_3008 – 0x4000_300B	4
	PROG_ADDR	0x4000_3010 – 0x4000_3013	4
	PROG_DATA	0x4000_3018 – 0x4000_301B	4
	TIME_REG0	0x4000_3030 – 0x4000_3033	4
	NVR_PASSWORD	0x4000_3050 – 0x4000_3053	4
	MAIN_PASSWORD	0x4000_3054 – 0x4000_3057	4
	CRC_ADDR	0x4000_3058 – 0x4000_305B	4
	CRC_LEN	0x4000_305C – 0x4000_305F	4

	CRC_OUT	0x4000_3060 – 0x4000_3063	4

4.2.2 闪存读保护

读操作在整个产品工作电压范围内都可以完成，用于存放指令或者数据，Flash运行在26MHz的工作频率上，若工作频率提升到30MHz以上，需要让Flash的读时序执行分频。

芯片带有cache缓冲区和预取缓冲区，提升Flash的访问效率。

若用户配置区经过自定义的配置后，当UART/Debug接口连接上时，会自动对Flash执行保护机制。

读操作由下列2个寄存器完成：

- ✧ 配置寄存器（CTRLR0)
- ✧ 时序0寄存器(TIME_REG0)

4.2.3 闪存烧写和擦除操作

烧写和擦除操作在整个产品工作电压范围内都可以完成。

烧写和擦除操作有下列6个寄存器完成，先根据烧写的时钟配置好烧写时序（TIME_REG1），再配置烧写密码，配置好编程地址，最后配置好编程数据，即可开始执行烧写，然后等烧写结束。

- ✧ 时序1寄存器（TIME_REG1)
- ✧ 密码0寄存器（NVR_PASSWORD)
- ✧ 密码1寄存器（MAIN_PASSWORD)
- ✧ 编程地址寄存器（PROG_ADDR)
- ✧ 编程数据寄存器（PROG_DATA)
- ✧ 状态寄存器（DONE)

在对 Flash 进行写/擦除操作的同时，任何对 Flash 的访问都会令总线停顿，直到写/擦除操作完成后才会继续执行，这意味着在写/擦除 Flash 的同时不可以对它取指和访问数据。

4.2.4 闪存寄存器接口

4.2.4.1 配置寄存器 (CTRLR0)

Address offset: 0x00

Width	Name	Reset	Property	Description
31:17	Reserved	0	RO	保留，始终读为 0
16	Program Clock select	0	RW	Eflash 烧写时钟源选择，推荐使用 RC 时钟 1'b0: 高速 RC 时钟 2 分频 1'b1: 晶振，若 RC 不准时才使用
15:12	Reserved	0	RO	保留，始终读为 0
11	LVD Program Disable	0	RW	在 LVD 断电时，是否允许打断 eflash program 和 erase，一般在断电时，立刻把重要数据保存在 eflash 上，就打开此功能，用于快速让 eflash 处于空闲状态 1'b0: 不允许 1'b1: 允许
10	Program RAM mode	0	RW	直接把 RAM 的数据写到 eflash，并在 eflash 最后自动添加 CRC 校验值，具体影响到几个寄存器 PROG_ADDR、CRC_ADDR、CRC_LEN

				0：CRC 模式 1：RAM to EFLASH 模式
9	Readm1	0	RW	Eflash 测试使用，正常情况下配置成 0
8	Readm0	0	RW	Eflash 测试使用，正常情况下配置成 0
7	SRAM Directly Oouput	0	RW	Eflash 的数据直接输出，用于系统时钟低频时，且 cache 和 prefetch 都不使能时使用 0：禁止； 1：使能。
6	Reserved	0	RO	保留，始终读为 0
5	Block request Mode	0	RW	Program/Erase 的请求缓冲模式 0：使能。 1：禁止；
4	Cache Write Back Mode	1	RW	Program/Sector Erase 自动回写到 cache 0：禁止； 1：使能。
3	Reserved	1	RO	保留，始终读为 1
2	Prefetch Enable	0	RW	预取使能位 0：禁止； 1：使能。
1	Reserved	0	RO	保留，始终读为 0
0	Cache Enable	0	RW	cache 使能位 0：禁止； 1：使能。

4.2.4.2 触发寄存器 (KST)

Address offset: 0x04

Width	Name	Reset	Property	Description
31:27	Reserved	0	RO	保留
26	CRC Kick Enable	0	WO	EFLASH CRC Check Enable
25:21	Reserved	0	RO	保留
20	Cache Clear Kick Enable	0	WO	Cache 初始化触发使能
19:11	Reserved	0	RO	保留
10	CRC Kick Start	0	WO	EFLASH CRC Check，和第 26 位同时写‘1’触发
9:5	Reserved	0	RO	保留
4	Cache Clear Kick Start	0	WO	Cache 初始化触发，和第 20 位同时写‘1’触发
3:0	Reserved	0	RO	保留，始终读为 0

4.2.4.3 状态寄存器 (DONE)

Address offset: 0x08

Width	Name	Reset	Property	Description
31:13	Reserved	0	RO	保留
12	Chip Erase OK Flag	1	RO	Chip Erase 成功标志 0：擦除失败

				1: 擦除成功
11	Program OK Flag	1	RO	Program 成功标志 0：烧写失败 1: 烧写成功
10	CRC Done	1	RO	CRC 结束标志 0：进行中 1：空闲状态
9:7	Reserved	0	RO	保留
6	Program Done	1	RO	Program 结束标志 0：进行中 1：空闲状态
5	Reserved	0	RO	保留
4	Cache Clear Done	1	RO	Cache 初始化标志 0：进行中 1：空闲状态
3:2	Reserved	0	RO	保留
1	Chip Erase Done	1	RO	Main 区域全擦除标志 0: 正在运行 1：空闲状态
0	Sector Erase Done	1	RO	扇区(512 byte)擦除标志 0: 正在运行 1：空闲状态

4.2.4.4 编程地址寄存器（PROG_ADDR)

Address offset: 0x10

Width	Name	Reset	Property	Description
31:30	Program Byte	0	RO	默认一次 program 是 4 byte
29	Program NVR Select	0	RW	编程的地址是否 NVR 区域 1'b0: MAIN 区域 1'b1: NVR 区域
28:2	Program Address	0	RW	eflash 编程的地址，以 word 为单位进行 program Note: 在配置寄存器里的第 10 位有效时，此寄存器作为 eflash 编程的开始地址
1:0	Program address	0	RO	低 2 位地址固定为 0，word 对齐

4.2.4.5 编程数据寄存器（PROG_DATA)

Address offset: 0x18

Width	Name	Reset	Property	Description
31:0	Program Data	0	RW	eflash 编程的数据，需要配置好地址才可进行 Note: 在配置寄存器里的第 10 位有效时，此寄存器作为 eflash 编程的数据

4.2.4.6 擦除控制寄存器 (ERASE_CTRL)

Address offset: 0x20

Width	Name	Reset	Property	Description
31	Chip Erase Kick Start	0	RO	Chip 全擦除的触发，写“1”触发，需要先配置密码
30	Sector Erase Kick Start	0	RO	Sector 擦除的触发，写“1”触发，需要先配置密码
29	NVR Sector Enable	0	RW	NVR 的 sector 使能位 0：MAIN 区域 1：NVR 区域
28:7	Reserved	0	RO	保留
6:0	Erase Sector Address	0	RW	擦除的 sector 选择，范围从 0-63 个

4.2.4.7 时序 0 寄存器 (TIME_REG0)

Address offset: 0x30

Width	Name	Reset	Property	Description
31:20	Reserved	0	RO	保留，始终读为 0
19:16	PGH	1	RW	WEb low to PROG2 high 保持时间最小为 15ns
15:12	ADS	1	RW	字节/地址/数据设置的最小时间是 15ns
11:8	ADH	1	RW	字节/地址/数据保持时间最小为 15ns
7:4	RW	8	RW	PROG/ERASE 后的下一个操作的延迟时间为 100ns
3:0	RC	0	RW	读取周期最小时间为 25/30ns

4.2.4.8 时序 1 寄存器 (TIME_REG1)

Address offset: 0x34

Width	Name	Reset	Property	Description
31:20	Reserved	0	RO	保留，始终读为 0
18:8	1ms unit	1000	RW	1ms 的时间配置值，以 1us 为单位
7:0	1us unit	13	RW	1us 的时间配置，默认烧写频率为 13MHz

4.2.4.9 密码 0 寄存器 (NVR_PASSWORD)

Address offset: 0x50

Width	Name	Reset	Property	Description
31:10	NVR password	0	RW	密码为 0x20150931，只有打开密码后，才能对 NVR 进行擦除和编程

4.2.4.10 密码 1 寄存器 (MAIN_PASSWORD)

Address offset: 0x54

Width	Name	Reset	Property	Description
31:10	Main password	0	RW	密码为 0x20170230，只有打开密码后，才能对 Main 进行擦除和编程

4.2.4.11 CRC 地址配置寄存器 (CRC_ADDR)

Address offset: 0x58

Width	Name	Reset	Property	Description
31:30	Reserved	0	RO	保留
29	NVR Select	0	RW	地址是否 NVR 区域 1'b0: MAIN 区域 1'b1: NVR 区域
28:2	DMA Address	0	RW	CRC DMA 的开始地址，只指向 eflash，且是物理地址 Note: 在配置寄存器里的第 10 位有效时，此寄存器配置成 RAM 的地址
1:0	Reserved	0	RO	低 2 位地址固定为 0，word 对齐

4.2.4.12 CRC 长度配置寄存器 (CRC_LEN)

Address offset: 0x5C

Width	Name	Reset	Property	Description
31:2	DMA length	0	RW	CRC DMA 的长度 Note: 在配置寄存器里的第 10 位有效时，此寄存器配置成 RAM 的长度
1:0	Reserved	0	RO	低 2 位地址固定为 0，word 对齐

4.2.4.13 CRC 结果寄存器 (CRC_OUT)

Address offset: 0x60

Width	Name	Reset	Property	Description
31:00	CRC Result	0	RO	CRC 的结果，多项式 CRC-32 如下，出来的结果取反就是官方的结果，写入 EFLASH 的 CRC 需要官方的值取反表示!!! x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1

4.2.5 闪存用户配置区

4.2.5.1 用户配置 0（0x7FC0）

Width	Name	Reset	Property	Description
31:3	Reserved	29'h1FFFFFFF	RO	保留
2:0	NVR Write Permission	3'h7	RO	NVR 的写权限,每一位代表一个 sector

4.2.5.2 用户配置 1（0x7FC4）

Width	Name	Reset	Property	Description
31:0	MAIN Write Permission	32'hFFFFFFFF	RO	MAIN 的写权限，根据 EFLASH 的 memory 大小确定每一位控制多少位 32Kbyte： 每一位代表 1 Kbyte

4.2.5.3 用户配置 2（0x7FC8）

Width	Name	Reset	Property	Description
31:12	reserved	21'hFFFFF	RO	保留
11:8	Read Region Size	4'hF	RO	DATA RAM 区域的大小 EFLASH 为 32Kbyte 时： 3'd0: 1 Kbyte 3'd1: 2 Kbyte 3'd2: 3 Kbyte 3'd3: 4 Kbyte 3'd4: 5 Kbyte 3'd5: 6 Kbyte 3'd6: 7 Kbyte 3'd7: 8 Kbyte Others：无效
7:1	reserved	7'hFF	RO	保留
0	MAIN Key Select	1'b1	RO	MAIN 区域的密钥来源 1'b0：片外密钥 0（由 UID 产生） 1'b1：片内密钥 0

4.2.5.4 用户配置 3（0x7FCC）

Width	Name	Reset	Property	Description
31:17	reserved	15'h7FFF	RO	保留
16	NVR0/1/2 Backup Disable	1'b1	RO	NVR 区域是否需要备份到 MAIN 区域中 1'b0：备份 1'b1：不备份
15:8	reserved	8'hFF	RO	保留
7:0	NVR 的备份地址	8'hFF	RO	在 MAIN 区域中备份的开始 sector 地址，总共 1.5Kbyte，从开始 sector 后 3 个 sector 需要占用。一旦备份有效，备份区无法通过软件进行 sector 读写擦

4.2.5.5 用户配置 4（0x7FD0）

Width	Name	Reset	Property	Description
31:15	reserved	17'h1FFFF	RO	保留
14	DBG Mode enable	1'b1	RO	DBG 模式是否有效，只要 main 配置区域校验通过后才允许关闭 1'b0：关闭 1'b1：打开
13	UART0 update mode enable	1'b0	RO	UART0 升级模式使能位，只要 main 配

				置区域校验通过后才允许关闭 1'b0：关闭 1'b1：打开
12	MCLR disable	1’b1	RO	MCLR 使能位，只要 main 配置区域校验通过后才允许打开 1'b0：打开 1'b1：关闭
11:10	reserved	2’h3	RO	保留
9	DBG Pullup or Pulldown	1’b1	RO	DBG 复位上拉或下拉，根据 DBG mapping 先选择哪个 port 作为 DBG，然后再根据此位来确定默认为复位上拉或下拉 1'b0：下拉 1'b1：上拉
7:0	UART mapping	7’hFF	RO	UART Port mapping 8’d0: PA11 8’d1: PB0 8’d2: PB2 8’d3: PB4 8’d4: PB6 8’d5: PB8 8’d6: PB10 8’d7: PB12 8’d8: PB14 8’d9: PC7 8’d10: PC6 8’d11: PC3 8’d12: PC2 8’d13: PC1 8’d14: PC0 8’d15: PA0 8’d16: PA1 8’d17: PA2 8’d18: PA3 8’d19: PA4 8’d20: PA5 8’d21: PA6 8’d22: PA7 8’d23: PA8

				8’d24: PA9 8’d25: PA10 Others: Disable

4.2.5.6 用户配置 5（0x7FD4）

Width	Name	Reset	Property	Description
31	MAIN code check enable	1’b1	RO	MAIN 区域程序 CRC 检验的使能位 1'b0：关闭 1'b1：打开
30:0	MAIN code Length	31’hFFFF	RO	MAIN 区域程序的长度，word 为单元

4.2.5.7 用户配置 6（0x7FFC）

Width	Name	Reset	Property	Description
31:0	CRC	32’hFFFFFFFF	RO	用户配置区的 CRC 校验码

4.2.6 闪存芯片信息区

4.2.6.1 芯片信息 0（0x1FF00600）

Width	Name	Reset	Property	Description
31:0	UID0	32'hFFFFFFFF	RO	UID 信息 0

4.2.6.2 芯片信息 1（0x1FF00604）

Width	Name	Reset	Property	Description
31:0	UID1	32'hFFFFFFFF	RO	UID 信息 1

4.2.6.3 芯片信息 2（0x1FF00608）

Width	Name	Reset	Property	Description
31:0	UID2	32'hFFFFFFFF	RO	UID 信息 2

4.2.6.4 芯片信息 3（0x1FF00620）

Width	Name	Reset	Property	Description
31:29	Reserved	3’h7	RO	保留
28:16	ADC Offset 1	13'h1FFF	RO	ADC Offset 值
15:13	Reserved	3’h7	RO	保留
12:0	ADC Offset 0	13'h1FFF	RO	ADC Offset 值

4.2.6.5 芯片信息 4（0x1FF00624）

Width	Name	Reset	Property	Description
31:30	Reserved	2’h3	RO	保留
29:16	ADC ALL Offset	14'h3FFF	RO	ADC Offset 值，第 29 位为有效位判定，若为‘1’，则认为此数据是无效值
15:13	Reserved	3’h7	RO	保留

12:0	ADC Offset 2	13'h1FFF	RO	ADC Offset 值

4.2.6.6 芯片信息 5（0x1FF00628）

Width	Name	Reset	Property	Description
31:26	Reserved	6’h3F	RO	保留
25:8	UART0 Update Baud	18’h00a93	RO	升级 UART0 的波特率，默认频率为 26Mhz，波特率 9600
7:0	EFLASH Program Freq	8’d13	RO	EFLASH 烧写频率是 RC 频率值的 2 分频，默认为 RC 时钟为 26MHz

4.2.6.7 芯片信息 6（0x1FF0062C）

Width	Name	Reset	Property	Description
31:24	Reserved	8'hFF	RO	保留
23:16	OPAM	8’hFF	RO	OPAM
15:0	Sensor	16'hFFFF	RO	温度传感器值

5 中断和事件

5.1 嵌套向量中断控制器

特征

- 中断都可屏蔽（除了 NMI）
- 4 个可编程的优先等级
- 低延迟的异常和中断处理
- 电源管理控制
- 系统控制寄存器的实现

嵌套向量中断控制器（NVIC）和处理器核的接口紧密相连，可以实现低延迟的中断处理和高效地处理晚到的中断。

嵌套向量中断控制器管理着包括核异常等中断。

5.2 系统嘀嗒（SysTick）校准值寄存器

本芯片支持使用外部时钟计时 1ms。

5.3 中断功能描述

处理器和嵌套式矢量型中断控制器(NVIC)在处理(Handler)模式下对所有异常进行优先级区分以及处理。当异常发生时，系统自动将当前处理器工作状态压栈，在执行完中断服务子程序 (ISR)后自动将其出栈。

取向量是和当前工作态压栈并行进行的，从而提高了中断入口效率。处理器支持咬尾中断，可实现背靠背中断，大大削减了反复切换工作态所带来的开销。

Table 5-1 CPU 中的 NVIC 异常类型

异常类型	向量编号	优先级	向量地址	描述
-	0	-	0x0000_0000	保留
复位	1	-3	0x0000_0004	复位
NMI	2	-2	0x0000_0008	不可屏蔽中断
硬件故障（HardFault）	3	-1	0x0000_000C	各种硬件级别故障
-	7-10	-	0x0000_001C-0x0000_002B	保留
-	13	-	0x0000_0034	保留
系统节拍	15	可编程设置	0x0000_003C	系统节拍定时器

SysTick 校准值设为 26'hCB19，SysTick 时钟频率配置为 HCLK，此时若 HCLK 时钟被配置为 52MHz，则 SysTick 中断会 1ms 响应一次。

Table 5-2 中断向量表

中断编号	向量编号	外设中断描述	向量地址
IRQ0	16	lvd_int	0x0000_0040
IRQ1	17	tk_irq_int	0x0000_0044
IRQ2	18	tk_scan_done_int	0x0000_0048
IRQ3	19	tk_samp_ovf_int	0x0000_004C
IRQ4	20	tk_scan_ovt_int	0x0000_0050
IRQ5	21	uart0_int	0x0000_0054

IRQ6	22	uart1_int	0x0000_0058
IRQ7	23	-	0x0000_005C
IRQ8	24	-	0x0000_0060
IRQ9	25	-	0x0000_0064
IRQ10	26	spi0_int	0x0000_0068
IRQ11	27	spi1_int	0x0000_006C
IRQ12	28	gpioa_int	0x0000_0070
IRQ13	29	gpiob_int	0x0000_0074
IRQ14	30	gpioc_int	0x0000_0078
IRQ15	31	wkpnd_int	0x0000_007C
IRQ16	32	timer0_int	0x0000_0080
IRQ17	33	timer1_int	0x0000_0084
IRQ18	34	timer2_int	0x0000_0088
IRQ19	35	timer3_int	0x0000_008C
IRQ20	36	timer4_int	0x0000_0090
IRQ21	37	timer5_int	0x0000_0094
IRQ22	38	-	0x0000_0098
IRQ23	39	-	0x0000_009C
IRQ24	40	adkey_interrupt	0x0000_00A0
IRQ25	41	-	0x0000_00A4
IRQ26	42	-	0x0000_00A8
IRQ27	43	crc_dma_int	0x0000_00AC
IRQ28	44	comp_int	0x0000_00B0
IRQ29	45	wdt_interrupt	0x0000_00B4
IRQ30	46	-	0x0000_00B8
IRQ31	47	-	0x0000_00BC

5.4 外部中断/事件控制器（EXTI）

外部中断和事件控制器（EXTI）管理外部和内部异步事件/中断，并生成相应的事件请求到 CPU/中断控制器和到电源管理的唤醒请求。每个输入线可以独立地配置输入类型（脉冲或挂起）和对应的触发事件（上升沿或下降沿或者双边沿都触发）。每个输入线都可以独立地被屏蔽。挂起寄存器保持着状态线的中断请求。

5.4.1 主要特征

EXTI 控制器的主要特性如下：

- 每个中断/事件都有独立的触发和屏蔽
- 每个中断线都有专用的状态位
- 支持多达 20 个软件的中断/事件请求
- 检测脉冲宽度低于 APB0 时钟宽度的外部信号。参见数据手册中电气特性部分的相关参数。

5.4.2 唤醒事件管理

TS32F020 可以处理外部或内部事件来唤醒内核（WFE）。唤醒事件可以通过下述配置产生：

- 外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时在 CPU 的系统控制寄存器中使

能 SEVONPEND 位。当 CPU 从 WFE 恢复后，需要清除相应外设的中断挂起位和外设 NVIC 中断通道挂起位（在 NVIC 中断清除挂起寄存器中）。

- 配置一个外部或内部 EXTI 线为事件模式，当 CPU 从 WFE 恢复后，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

6 循环冗余校验计算单元（CRC）

6.1 简介

循环冗余校验（CRC）计算单元是根据自定义的生成多项式得到任一 32 位全字的 CRC 计算结果。在其他的应用中，CRC 技术主要应用于核实数据传输或者数据存储的正确性和完整性。CRC 计算单元可以在程序运行时计算出软件的标识，之后与在连接时生成的参考标识比较，然后存放在指定的存储器空间。

6.2 主要特性

- 支持 5/7/8/16/32 位等不同长度的多项式
- 支持自定义的多项式
- 默认是 32 位多项式:

  X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 +X8 + X7 + X5 + X4 + X2+ X +1
- 一个 32 位初始值配置寄存器用于输入，一个 32 位结果寄存器用于输出结果

6.3 寄存器定义

6.3.1 控制配置寄存器 (CRC_CFG)

Address offset: 0x00

Width	Name	Reset	Property	Description
31:14	Reserved	0	RO	保留，始终读为 0
13:8	POLY_WIDTH	32	RW	支持 5/7/8/16/32 等不同长度的多项式
7:2	Reserved	0	RO	保留，始终读为 0
1	BIT_ORDER_EN	0	RW	输入数据的顺序，from bit 7:0 to 0:7 0：不使能 1：使能
0	INT_EN	0	RW	CRC 中断 0: 不使能 1: 使能

6.3.2 初始值配置寄存器(CRC_INIT)

Address offset: 0x04

Width	Name	Reset	Property	Description
31:0	INIT_VALUE	32'hFFFFFFFF	RW	CRC 初始值

6.3.3 取反寄存器(CRC_INV)

Address offset: 0x08

Width	Name	Reset	Property	Description
31:0	INV_VALUE	32'hFFFFFFFF	RW	CRC 输出结果取反

6.3.4 多项式配置寄存器(CRC_POLY)

Address offset: 0x0C

Width	Name	Reset	Property	Description
31:0	POLY_VALUE	32'hedb88320	RW	CRC 多项式，默认为 32 位多项式

6.3.5 触发寄存器(CRC_KST)

Address offset: 0x10

Width	Name	Reset	Property	Description
31:1	Reserved	0	RW	保留，始终读为 0
0	Pending Clear	0	WO	清除 CRC 状态位

6.3.6 状态寄存器(CRC_STA)

Address offset: 0x14

Width	Name	Reset	Property	Description
31:1	Reserved	0	RO	保留，始终读为 0
0	Pending	0	RO	CRC 状态位

6.3.7 地址寄存器(CRC_ADDR)

Address offset: 0x1C

Width	Name	Reset	Property	Description
31:0	DMA Start Address	0	RW	CRC DMA 开始的物理地址 注意：物理地址，比如地址为 0x0 重定向 0x2000_0000

6.3.8 长度寄存器(CRC_LEN)

Address offset: 0x20

Width	Name	Reset	Property	Description
31:2	DMA Length	0	RW	CRC DMA 字长
1:0	Reserved	0	RO	4 byte 对齐

6.3.9 结果寄存器(CRC_OUT)

Address offset: 0x24

Width	Name	Reset	Property	Description
31:0	CRC Result	0	RO	CRC 输出结果

6.4 操作流程

1. 配置 CRC 初始值(CRC_INIT)和取反(CRC_INV)

2. 配置 CRC 多项式(CRC_POLY)，CRC 多项式的位宽(CRC_CFG)
3. 配置 DMA 的开始地址（物理地址），最后配置 DMA 的长度，一旦长度不为 0，即开始执行 CRC 校验
4. 等待 CRC 的 pending(CRC_STA)，得到 CRC 的结果(CRC_OUT)，最后清除 pending(CRC_KST)

7 电源控制（power control）

7.1 电源

芯片的工作电压（VCC） 为 2.7V ~ 5.5V。本芯片采用 CAPLESS 设计，无需在内置 LDO 输出上外挂电容。

7.1.1 电压调节器

复位后调节器总是使能的。在需要低功耗的场合，可以使能低功耗工作模式，PMUCON0. pmu_v2ien, PMUCON0. lpldo_en 两 bit 协同工作做功耗模式切换控制位。

7.2 电源管理器

7.2.1 上电复位（POR） 和掉电复位（PDR）

TS32F020 内部有一个完整的上电复位（POR）和掉电复位（PDR） 电路，当供电电压达到 2.7V 时系统既能正常工作。当 VDD 低于指定的限位电压 VPOR/VPDR 时，系统保持为复位状态，而无需外部复位电路。

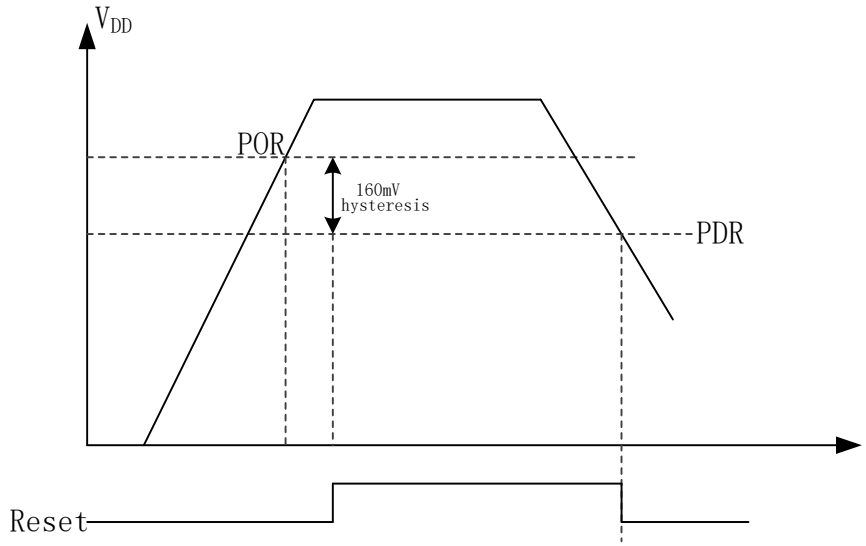


图 8-1 上电复位和掉电复位波形图

7.2.2 可编程电压监测器（PVD）

TS32F020 内部集成两个电压检测器，一个检测外部供电 VCC,一个检测内部 LDO 输出 VDD，LDO 采用 capless 结构，封装上 VDD 不可见。两种检测电压均阈值可选。当系统监测到 VCC 或 VDD 电压低于配置电压值时，可以选择触发系统复位或通过使能 PVD 中断进入中断子函数。这一特性可用于用于执行紧急关闭任务。检测信号可以选择经过毛刺滤波电路或直接检测，由 LVDCON. lvdvcc_bps_en 和 LVDCON. lvdvdd_bps_en 来控制。

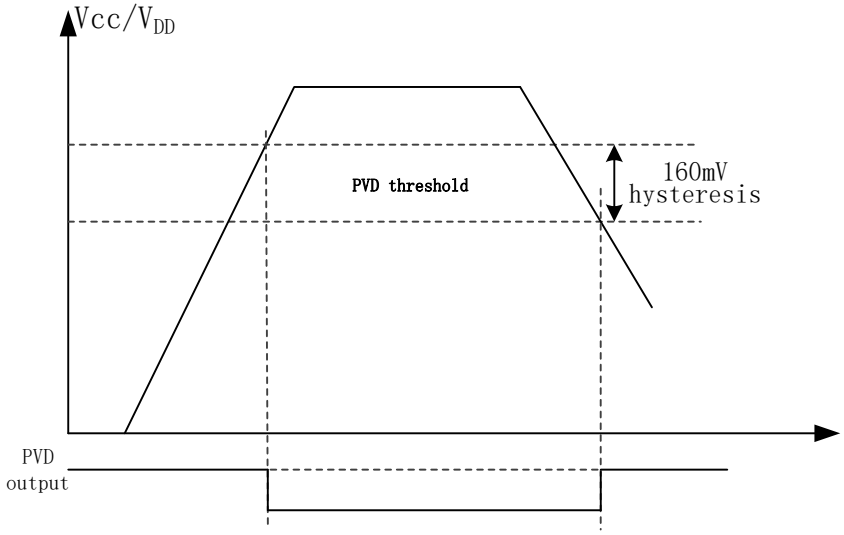


图 8-2 PVD 的阈值

7.3 低功耗模式

在系统或电源复位以后，微控制器处于运行状态，系统所用时钟为 256KHz 内部 RC。当 CPU 不需继续运行时，可以利用多种低功耗模式来节省功耗，例如等待某个外部事件时。用户需要根据最低电源消耗、最快速启动时间和可用的唤醒源等条件，选定一个最佳的低功耗模式。

TS32F020 有三种低功耗模式：

- 待机模式（Idle Mode）--（CPU 停止，所有外设包括 CPU 的外设，如 NVIC、系统时钟（SysTick）等仍在运行，所有的中断都可以唤醒）
- 停止模式（Stopclk Mode）--（CPU,大部分外设停止，可依赖其他事件唤醒此模式后芯片继续运行）
- 睡眠模式（Sleep Mode）--（所有时钟源都停止，依赖外部 IO 唤醒、复位芯片）

此外，在运行模式下，可以通过以下方式中的一种降低功耗：

- 降低系统时钟
- 关闭 APB 和 AHB 总线上未被使用的外设时钟。
- 合理配置 APB 与 AHB 的频率关系

7.4 电源控制寄存器

7.4.1 寄存器: LVDCON

地址： 0x4000A100

复位值： 0x0000 0767

Name	Bit	Reset	ACCESS	Description
lvdvdd_pending	31	0	R	VDD 电压低于设定阈值触发标志，写 0 清除这个标志
lvdvcc_pending	30	0	R	VCC 电压低于设定阈值触发标志，写 0 清除这个标志
dbs_lo_limit	29:22	0	RW	滤波检测电压信号低毛刺宽度，单位为系统时钟

				周期
dbs_hi_limit	21:14	0	RW	滤波检测电压信号高毛刺宽度，单位为系统时钟周期
lvdvcc_sync_dis	13	0	RW	检测到 VCC 低电压信号触发 GPIO 复位是否需要同步 0：需要同步 1：不同步
lvdvdd_bps_en	12	0	RW	采样去抖动后的 VDD 电压状态使能 0：不使能 1：使能
lvdvcc_bps_en	11	0	RW	采样去抖动后的 VCC 电压状态使能 0：不使能 1：使能
lvd_oe	10	1	RW	阈值判断触发条件后中断，以及复位功能使能 0：不使能 1：使能
lvdvdd_rst_en	9	1	RW	VDD 阈值判断触发后复位系统使能 0：中断，不复位 1：复位，不中断
lvdvcc_rst_en	8	1	RW	VCC 阈值判断触发后复位系统使能 0：中断，不复位 1：复位，不中断
llvd_s	7:6	2'h1	RW	VDD LVD setting: 00=1.05 01=1.25 10=1.35 11=1.40
llvd_en	5	1	RW	VDD LVD enable signal
hlvd_s	4:1	4'h3	RW	VCC LVD Setting 0000=1.8 0001=2.1 0010=2.4 0011=2.7 0100=3.0 0101=3.3 0110=3.6 0111=3.9 1000=4.2 1001=4.5 1010=4.8 1011=1.65 1100=1.95 1101=2.25 1110=2.55 1111=2.85
hlvd_en	0	1	RW	VCC LVD 使能

8 复位和时钟控制

8.1 复位

TS32F020 支持系统复位、电源复位和主复位

8.1.1 系统复位

系统复位将复位除某些复位状态寄存器和特殊功能寄存器之外的所有寄存器。

当以下事件中的一件发生时，产生一个系统复位：

1. SLEEP 模式下外部 IO 口唤醒
2. WDT 计数溢出复位
3. 系统复位请求复位
4. UART0 升级程序复位
5. 系统锁定复位

8.1.2 主复位

主复位能将部分系统复位无法复位的寄存器复位。

以下事件可以触发一个主复位：

1. 软件复位
2. PVD 检测到电压低事件，且控制器处于复位功能模式
3. 当芯片支持 MCLR（引脚）复位时，MCLR 指在某个特定 IO 引脚上输入一个持续 1.7ms 以上的低电平导致系统复位

8.1.3 电源复位

上电/掉电复位（POR/PDR 复位）都属于电源复位。电源复位将复位所有的逻辑和模拟模块。复位入口矢量被固定在地址 0x0000_0004。

8.2 时钟

8.2.1 XOSC 时钟

高速外部时钟信号（XOSC）由以下两种时钟源产生：

- 外部晶体/陶瓷谐振器
- 用户外部时钟

8.2.2 HIRC 时钟

HIRC 时钟信号由内部 26MHz 的振荡器产生，可直接作为系统时钟或作为 PLL 输入。 HIRC 振荡器能够在不需要任何外部器件的条件下提供系统时钟。它的启动时间比 XOSC 晶体振荡器短。然而，即使在校准之后它的时钟频率精度仍较差。需要出厂时校准，校准值写在 flash 系统存储区域。在程序使用这个时钟前，可以读取并配置出高精度的 HIRC 时钟。经过出厂效验后，常温状态下 HIRC 精度为 26MHz(±1.5%)。精确的频率在 0x1FF0_0600 – 0x1FF0_06FF 闪存区域有描述，用户可通过读取得到

HIRC 的精确频率。

如果 XOSC 晶体振荡器失效， HIRC 时钟会被作为备用时钟源。

8.2.3 PLL 配置

内部 PLL 可以用来倍频内部时钟源如 XOSC,HIRC,LIRC 等。可以作为系统时钟。

8.2.4 LIRC 时钟

LIRC 振荡器担当一个低功耗时钟源的角色，它作为系统启动时钟为看门狗和其他单元提供时钟。LIRC 时钟频率大约 256KHz（在 77KHz 和 435KHz 之间）。进一步信息请参考数据手册中有关电气特性部分。

8.2.5 系统时钟（SYSCLK）选择

四种不同的时钟源可被用来驱动系统时钟（SYSCLK）：

- 内部低速 256Khz LIRC
- 内部高速 26Mhz HIRC
- 外部高速晶振 XOSC
- 片内高速 PLL 时钟

8.2.6 毛刺滤波时钟源选择

四种不同的时钟源可被用来驱动 GPIO 的毛刺滤波时钟：

- 外部高速晶振 XOSC
- 内部高速 26Mhz HIRC 的分频时钟
- 系统时钟
- 内部低速 256Khz LIRC

8.2.7 比较器时钟选择

四种不同的时钟源可被用来驱动比较器的时钟（CMPCLK）：

- 系统时钟
- 内部高速 26Mhz HIRC 的分频时钟
- 内部低速 256Khz LIRC
- 外部高速晶振 XOSC

当不被使用时，任一个时钟源都可被独立地启动或关闭，由此优化系统功耗。

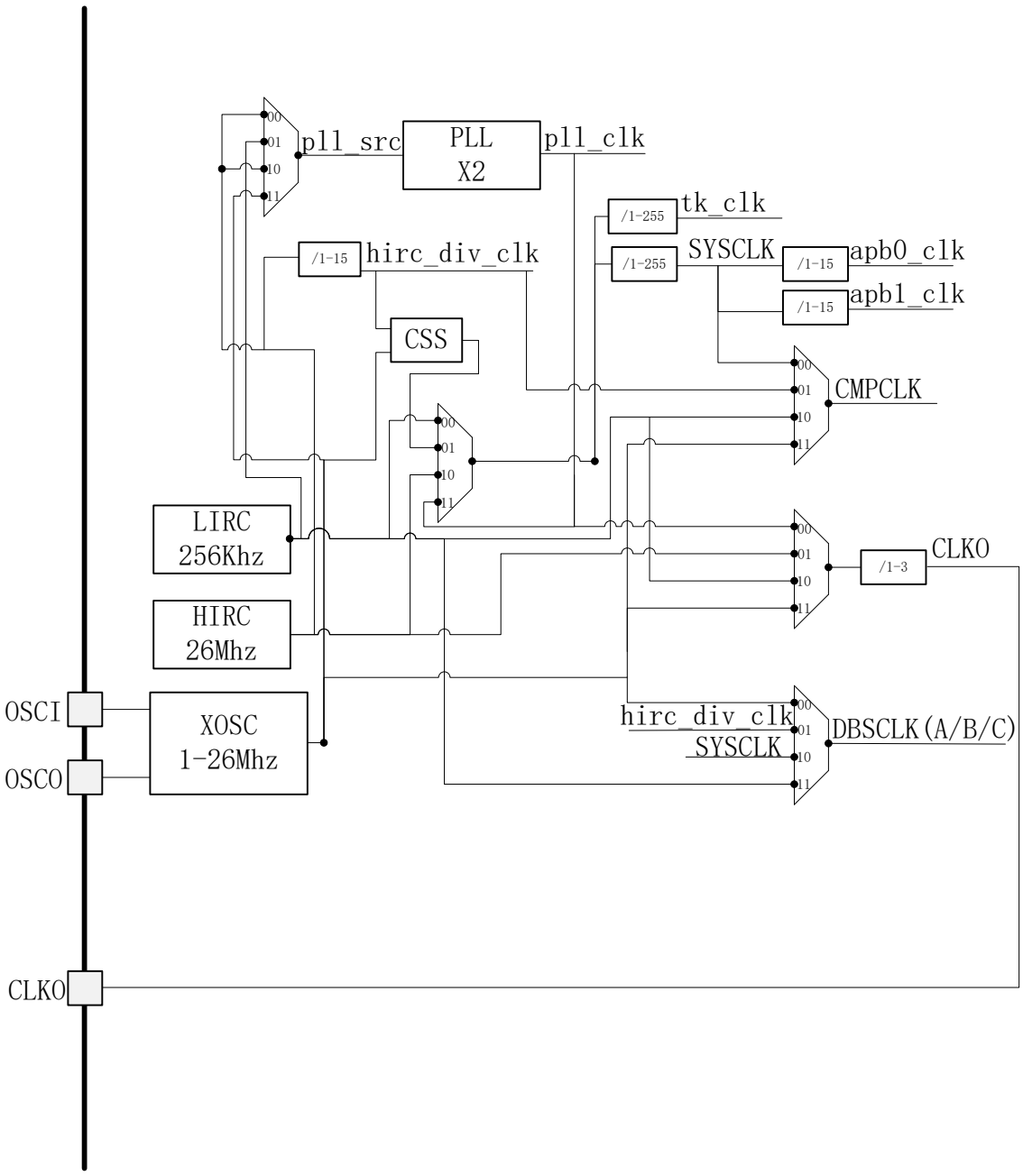


图 8-1 时钟结构

时钟安全系统（CSS）

时钟安全系统可以通过软件被激活。在使用这个功能时，请确保 XOSC 已经被成功使能。如果 XOSC 时钟发生故障，产生时钟安全中断 CSSI，允许软件完成营救操作。此 CSSI 中断连接到 CPU 的 NMI 中断。注：一旦 *CSS* 被激活，并且 *XOSC* 时钟出现故障， *CSS* 中断就产生，并且 *NMI* 也自动产生。 *NMI* 将被不断执行，直到 *CSS* 中断挂起位被清除。因此，在 *NMI* 的处理程序中必须通过设置时钟中断寄存器，（HOSC_MNT） 里的 hosc_loss_pending 位写 1 来清除 *CSS* 中断。通过寄存器使能，时钟故障将导致内部 XOSC 时钟自动切换到 hirc_ div_clk。

9 通用输入输出（GPIO）

9.1 简介

每组 GPIO 端口有四个 32 位配置寄存器(GPIOx_MODER,GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR)，两个 32 位数据寄存器(GPIOx_IDR and GPIOx_ODR)，一个 32 位置位/复位寄存器(GPIOx_BSR) 和一个 32 位翻转寄存器(GPIOx_TGL)。另外，所有 GPIO 有一个 32 位锁定寄存器(GPIOx_LCKR) 和两个复用功能选择寄存器 (GPIOx_AFRH and GPIOx_AFRL)。

9.2 GPIO 主要特征

- 输出状态：推挽或开漏+上下拉
- 输出寄存器状态值 (GPIOx_ODR) 或者复用功能输出
- 输入状态：浮空、上下拉、模拟
- 输入数据到数据寄存器(GPIOx_IDR) 或复用功能输入
- 独立置位/复位/翻转 IO 状态 (GPIOx_ BSR、GPIOx_TGL)
- 通过加锁配置 (GPIOx_LCKR) 锁定 IO 状态
- 模拟功能
- 复用功能

9.3 GPIO 功能描述

GPIO 的每一个端口可以通过软件独立配置成下面状态：

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟功能
- 开漏输出
- 推挽输出
- 复用功能

Port bit configuration table

MODE(i)	OTYPE(i)	OSPEED(i)	PUPD(i)		IO configuration	
01	0	SPEED[3:0]	0	0	GP output	PP
	0		0	1	Reserved	
	0		1	0		
	0		1	1		
	1		0	0	GP output	OD
	1		0	1	GP output	OD+PU
	1		1	0	Reserved	
	1		1	1		
10	0	SPEED[3:0]	0	0	AF	PP
	0		0	1	Reserved	
	0		1	0		
	0		1	1		
	1		0	0	AF	OD
	1		0	1	AF	OD+PU
	1		1	0	Reserved	

	1			1	1		
00	x	x	x	0	0	Input	PP
	x	x	x	0	1	Input	PP+PU
	x	x	x	1	0	Input	PP+PD
	x	x	x	1	1	Reserved	
11	x	x	x	0	0	Input/Output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1.GP = generate-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

9.3.1 通用 IO（GPIO）

复位期间和刚复位后，复用功能未开启，I/O 端口被配置成浮空输入模式。

当作为输出配置时，写到输出数据寄存器上的值（GPIOx_ODR）输出到相应的 I/O 引脚。可以以推挽或开漏模式使用输出驱动器。

输入数据寄存器（GPIOx_IDR）在每个 APB 时钟周期捕捉 I/O 引脚上的数据。

所有 GPIO 引脚有一个内部弱上拉和弱下拉，当配置为输入时，它们可以被激活也可以被断开。

9.3.2 单独的位操作

当对 GPIOx_ODR 的个别位编程时，软件不需要禁止中断：在单次 APB 写操作里，可以只更改一个或多个位。只需要通过对“置位/复位寄存器”（GPIOx_BSR）或“取反寄存器”（GPIOx_TGL）中想要更改的位写“1”来实现。没被选择的位将不被更改。

9.3.3 复用功能（AF）

芯片 IO 引脚通过多路选择器连接到片内外设，每个 IO 上同一时刻只能选通一个复用功能。每个 IO 引脚有一个 4 输入的多路选择器连接到复用功能（AF0~AF3），通过配置 GPIOx_AFRH/L 选择输入功能。如果把端口配置成复用输出功能，则引脚和输出寄存器断开，并和片上外设的输出信号连接。如果软件把一个 GPIO 脚配置成复用输出功能，但是外设没有被激活，它的输出将不确定。

9.3.4 GPIO 锁定机制

锁定机制允许在 GPIO 控制寄存器 GPIOx_LCKR 上执行一串锁定程序，然后把 GPIO 的状态锁定，一旦 GPIO 状态被锁定，将不可改变，直到 CPU 复位。被锁定的寄存器有（GPIOx_MODER, GPIOx_OTYPER,GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH）。

锁定序列参考 GPIOx_LCKR 寄存器描述。

9.3.5 输入配置

当 I/O 端口配置为输入时：

- 输出缓冲器被禁止

- 施密特触发输入被激活
- 根据输入配置（上拉、下拉或浮空）的不同，弱上拉和下拉电阻被连接
- 出现在 I/O 脚上的数据在每个 APB 时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到 I/O 状态

9.3.6 输出配置

当 I/O 端口被配置为输出时：

- 输出缓冲器被激活
  - ✓ 开漏模式：输出寄存器上的“0”激活 N-MOS，而输出寄存器上的“1”将端口置于高阻态（P-MOS 从不被激活）。
  - ✓ 推挽模式：输出寄存器上的“0”激活 N-MOS，而输出寄存器上的“1”将激活 P-MOS。
- 施密特触发输入被激活
- 弱上拉和下拉电阻被禁止
- 出现在 I/O 脚上的数据在每个 APB 时钟被采样到输入数据寄存器
- 在开漏模式时，对输入数据寄存器的读访问可得到 I/O 状态
- 在推挽模式时，对输出数据寄存器的读访问得到最后一次写的值。

9.3.7 模拟输入配置

当 I/O 端口被配置为模拟输入配置时：

- 输出缓冲器被禁止；
- 禁止施密特触发输入，实现了每个模拟 I/O 引脚上的零消耗。施密特触发输出值被强制为“0”；
- 弱上拉和下拉电阻被禁止；
- 读取输入数据寄存器时数值为“0”。

9.3.8 复用功能配置

对 I/O 端口进行编程作为复用功能时：

- 在开漏或推挽式配置中，输出缓冲器被打开
- 内置外设的信号驱动输出缓冲器 (复用功能输出)
- 施密特触发输入被激活
- 弱上拉和下拉电阻被禁止
- 在每个 APB 时钟周期，出现在 I/O 脚上的数据被采样到输入数据寄存器
- 开漏模式时，读输入数据寄存器时可得到 I/O 口状态
- 在推挽模式时，读输出数据寄存器时可得到最后一次写的值

9.3.9 GPIO 端口模式寄存器(GPIOx_MODE)

Address offset: 0x00

Width	Name	Reset	Property	Description
31:30	MODER15	0	RW	

29:28	MODER14	0	RW	Bits 2y+1:2y MODERy[1:0]: Port x configuration bits (y = 0..15) 这些位由软件编写以配置I/o模式 00: 输入模式 (复位状态) 01: 通用输出模式 10: 复用功能模式 11: 模拟模式
27:26	MODER13	0	RW	
25:24	MODER12	0	RW	
23:22	MODER11	0	RW	
21:20	MODER10	0	RW	
19:18	MODER9	0	RW	
17:16	MODER8	0	RW	
15:14	MODER7	0	RW	
13:12	MODER6	0	RW	
11:10	MODER5	0	RW	
9:8	MODER4	0	RW	
7:6	MODER3	0	RW	
5:4	MODER2	0	RW	
3:2	MODER1	0	RW	
1:0	MODER0	0	RW	

9.3.10 GPIO 端口输出类型寄存器(GPIOx_OTYPE)

Address offset: 0x04

Width	Name	Reset	Property	Description
31:16	Reserved	0	RW	OTy: Port x configuration bits (y = 0..15) 这些位可以通过软件配置I/O输出类型 0: Output 推挽 (复位状态) 1: Output 开漏
15	OT15	0	RW	
14	OT14	0	RW	
13	OT13	0	RW	
12	OT12	0	RW	
11	OT11	0	RW	
10	OT10	0	RW	
9	OT9	0	RW	
8	OT8	0	RW	
7	OT7	0	RW	
6	OT6	0	RW	
5	OT5	0	RW	
4	OT4	0	RW	
3	OT3	0	RW	
2	OT2	0	RW	
1	OT1	0	RW	
0	OT0	0	RW	

9.3.11 GPIO 端口输出低位速度寄存器(GPIOx_OSPEEDL)

Address offset: 0x08

Width	Name	Reset	Property	Description
31:28	OSPEED7	0	RW	Bits 31:0 OSPEEDy[3：0]: Port x configuration bits (y = 0..7) 这些位可以通过软件配置I/O输出速度 For GPIOA:
27:24	OSPEED6	0	RW	
23:20	OSPEED5	0	RW	
19:16	OSPEED4	0	RW	

15:12	OSPEED3	0	RW	normal mode : 0000: low speed 0001: speed1 0010: speed2 0011: speed3 0100: speed4 0101: speed5 0110: speed6 0111: high speed others : reserved electricity mode : (Must configure PMUCON0[11] == 1 and PMUCON0[13] == 1) 1000: low speed 1001: speed1 1010: speed2 1011: speed3 1100: speed4 1101: speed5 1110: speed6 1111: high speed others : reserved For GPIOB/C: 0xx0: low speed 0xx1: speed1 1xx0: speed2 1xx1:very high speed others : reserved
11:8	OSPEED2	0	RW	
7:4	OSPEED1	0	RW	
3:0	OSPEED0	0	RW	

9.3.12 GPIO 端口输出高位速度寄存器(GPIOx_OSPEEDH)

Address offset: 0x0C

Width	Name	Reset	Property	Description
31:28	OSPEED15	0	RW	Bits 31:0 **OSPEEDy[3:0]**: Port x configuration bits (y = 8…15) 这些位可以通过软件配置I/O输出速度 For **GPIOA**: normal mode : 0000: low speed 0001: speed1 0010: speed2 0011: speed3 0100: speed4 0101: speed5 0110: speed6 0111: high speed others : reserved
27:24	OSPEED14	0	RW	
23:20	OSPEED13	0	RW	
19:16	OSPEED12	0	RW	
15:12	OSPEED11	0	RW	
11:8	OSPEED10	0	RW	
7:4	OSPEED9	0	RW	
3:0	OSPEED8	0	RW	

				electricity mode : (Must configure PMUCON0[11] == 1 and PMUCON0[13] == 1) 1000: low speed 1001: speed1 1010: speed2 1011: speed3 1100: speed4 1101: speed5 1110: speed6 1111: high speed others : reserved For **GPIOB/C**: 0xx0: low speed 0xx1: speed1 1xx0: speed2 1xx1:very high speed others : reserved

9.3.13 GPIO 端口上拉/下拉寄存器(GOIOx_PUPD)

Address offset: 0x10

Width	Name	Reset	Property	Description
31	PD15	0	RW	Bits **PDy:** Port x configuration bits (y = 16...31) 这些位可以通过软件配置I/O下拉 0: 不下拉 1: 下拉
30	PD14	0	RW	
29	PD13	0	RW	
28	PD12	0	RW	
27	PD11	0	RW	
26	PD10	0	RW	
25	PD9	0	RW	
24	PD8	0	RW	
23	PD7	0	RW	
22	PD6	0	RW	
21	PD5	0	RW	
20	PD4	0	RW	
19	PD3	0	RW	
18	PD2	0	RW	
17	PD1	0	RW	
16	PD0	0	RW	
15	PU15	0	RW	Bits **PUy:** Port x configuration bits (y = 0…15) 这些位可以通过软件配置I/O上拉 0: 不上拉 1: 上拉 note:此位在GPIO配置为输入模式、输出开漏模式且输出电平较高时才有效
14	PU14	0	RW	
13	PU13	0	RW	
12	PU12	0	RW	
11	PU11	0	RW	
10	PU10	0	RW	
9	PU9	0	RW	
8	PU8	0	RW	

7	PU7	0	RW	
6	PU6	0	RW	
5	PU5	0	RW	
4	PU4	0	RW	
3	PU3	0	RW	
2	PU2	0	RW	
1	PU1	0	RW	
0	PU0	0	RW	

9.3.14 GPIO 端口输入数据寄存器(GPIOx_IDR)

Address offset: 0x14

Width	Name	Reset	Property	Description
31:16	Reserved	0	RW	
15	IDR15	0	RO	Bits 15:0 **IDRy:** Port input data bit (y = 0..15) 这些位只读. 它们包含相应 I/O 端口的输入值
14	IDR14	0	RO	
13	IDR13	0	RO	
12	IDR12	0	RO	
11	IDR11	0	RO	
10	IDR10	0	RO	
9	IDR9	0	RO	
8	IDR8	0	RO	
7	IDR7	0	RO	
6	IDR6	0	RO	
5	IDR5	0	RO	
4	IDR4	0	RO	
3	IDR3	0	RO	
2	IDR2	0	RO	
1	IDR1	0	RO	
0	IDR0	0	RO	

9.3.15 GPIO 端口输出数据寄存器(GPIOx_ODR)

Address offset: 0x18

Width	Name	Reset	Property	Description
31:16	Reserved	0	RW	
15	ODR15	0	RW	
14	ODR14	0	RW	
13	ODR13	0	RW	Bits 15:0 ODRy: Port output data bit (y = 0..15) 这些位可读可写。 对于初始位置位和复位，ODR寄存器的这些位可以单独置位或者通过写入GPIOx_BSRR或GPIOx_BRR寄存器进行复位(x = A..F)。
12	ODR12	0	RW	
11	ODR11	0	RW	
10	ODR10	0	RW	
9	ODR9	0	RW	
8	ODR8	0	RW	
7	ODR7	0	RW	
6	ODR6	0	RW	

5	ODR5	0	RW	
4	ODR4	0	RW	
3	ODR3	0	RW	
2	ODR2	0	RW	
1	ODR1	0	RW	
0	ODR0	0	RW	

9.3.16 GPIO 端口置位/复位寄存器(GPIOx_BSR)

Address offset: 0x1C

Width	Name	Reset	Property	Description
31	BR15	0	WO	Bits 31:16 BRy: Port x reset bit y (y = 0..15) 这些位只写。对这些位的读取将返回值 0x0000。 0: 对应的 ODRx 位没有动作 1: 复位相应的 ODRx 位 Note: 如果设置了 BSx 和 BRx，则 BRx 具有优先级。
30	BR14	0	WO	
29	BR13	0	WO	
28	BR12	0	WO	
27	BR11	0	WO	
26	BR10	0	WO	
25	BR9	0	WO	
24	BR8	0	WO	
23	BR7	0	WO	
22	BR6	0	WO	
21	BR5	0	WO	
20	BR4	0	WO	
19	BR3	0	WO	
18	BR2	0	WO	
17	BR1	0	WO	
16	BR0	0	WO	
15	BS15	0	WO	Bits 15:0 BSy: Port x set bit y (y= 0..15) 这些位只写。对这些位的读取将返回值 0x0000。 0: 对应的 ODRx 位没有动作 1: 置位相应的 ODRx 位
14	BS14	0	WO	
13	BS13	0	WO	
12	BS12	0	WO	
11	BS11	0	WO	
10	BS10	0	WO	
9	BS9	0	WO	
8	BS8	0	WO	
7	BS7	0	WO	
6	BS6	0	WO	
5	BS5	0	WO	
4	BS4	0	WO	
3	BS3	0	WO	
2	BS2	0	WO	
1	BS1	0	WO	
0	BS0	0	WO	

9.3.17 GPIO 复用功能低位寄存器(GPIOx_AFRL)

Address offset: 0x24

Width	Name	Reset	Property	Description
31:28	AFR7	0	RW	Bits 31:0 **AFRy[3:0]:** Alternate function selection for port x pin y (y = 0…7) 这些位通过软件写入用于配置I/O口复用功能 AFSELy selection: 0000: AF0 0001: AF1 0010: AF2 0011: AF3 others: reserve
27:24	AFR6	0	RW	
23:20	AFR5	0	RW	
19:16	AFR4	0	RW	
15:12	AFR3	0	RW	
11:8	AFR2	0	RW	
7:4	AFR1	0	RW	
3:0	AFR0	0	RW	

9.3.18 GPIO 复用功能高位寄存器(GPIOx_AFRH)

Address offset: 0x28

Width	Name	Reset	Property	Description
31:28	AFR15	0	RW	Bits 31:0 **AFRy[3:0]:** Alternate function selection for port x pin y (y = 8…15) 这些位通过软件写入用于配置I/O口复用功能 AFSELy selection: 0000: AF0 0001: AF1 0010: AF2 0011: AF3 others: reserve
27:24	AFR14	0	RW	
23:20	AFR13	0	RW	
19:16	AFR12	0	RW	
15:12	AFR11	0	RW	
11:8	AFR10	0	RW	
7:4	AFR9	0	RW	
3:0	AFR8	0	RW	

9.3.19 GPIO 端口配置锁定寄存器(GPIOx_LCK)

当正确的写入序列应用于第16位(LCKK)时，此寄存器用于锁定端口位的配置。位 [15:0]用于锁定theGPIO的配置。在写入序列期间，LCKR[15:0]的值不能更改。当锁序列被应用在一个端口位上时，该端口位的值在下一次MCU复位或外设复位之前不能再更改。

Address offset: 0x20

Width	Name	Reset	Property	Description
31:17	Reserved	0	RW	
16	LCKK	0	RW	Bit 16 **LCKK**: 锁键 这个位可以在任何时候读取。它只能使用锁定键写入序列修改它 0: 端口配置锁定键不活动 1: 端口配置锁定键激活。GPIOx_LCK寄存器被锁定，直到下一次MCU复位或外围设备复位. LOCK key write sequence: WR LCK [16] = '1' + LCK [15:0] WR LCK [16] = '0' + LCK [15:0] RD LCK [16] = '1' (这个读取操作是可选的，但它确认锁是活动的) Note: 在锁键写入序列期间，LCK[15:0]的值

				不能改变. 锁定序列中的任何错误都会中止锁定 第一次锁后序列在任何端口,任何读访问 LCKK 的位将返回' 1 ',直到下一个单片机复位或外围重置。
15	LCK15	0	RW	Bits 15:0 **LCKy**: Port x lock bit y (y= 0..15) 这些位是读/写的，但只能在 LCKK 位为 0 写入。 0: 端口配置未锁定 1: 端口配置锁定
14	LCK14	0	RW	
13	LCK13	0	RW	
12	LCK12	0	RW	
11	LCK11	0	RW	
10	LCK10	0	RW	
9	LCK9	0	RW	
8	LCK8	0	RW	
7	LCK7	0	RW	
6	LCK6	0	RW	
5	LCK5	0	RW	
4	LCK4	0	RW	
3	LCK3	0	RW	
2	LCK2	0	RW	
1	LCK1	0	RW	
0	LCK0	0	RW	

9.3.20 GPIO 端口位切换寄存器(GPIOx_TGL)

Address offset: 0x2C

Width	Name	Reset	Property	Description
31:16	Reserved	0	RW	Bits 15:0 **TGy:** Port x toggle bit y (y = 0..15) 这些位是只写的。对这些位的读取将返回值 0x0000 0: 对应的 ODRx 位没有动作 1: 切换相应的 ODRx 位 Note: 如果设置了 BSx BRx 和 TGx，则 BRx 具有第一优先级，BSx 具有第二优先级。.
15	TG15	0	WO	
14	TG14	0	WO	
13	TG13	0	WO	
12	TG12	0	WO	
11	TG11	0	WO	
10	TG10	0	WO	
9	TG9	0	WO	
8	TG8	0	WO	
7	TG7	0	WO	
6	TG6	0	WO	
5	TG5	0	WO	
4	TG4	0	WO	
3	TG3	0	WO	
2	TG2	0	WO	
1	TG1	0	WO	
0	TG0	0	WO	

9.3.21 GPIO 端口中断屏蔽寄存器(GPIOx_IMK)

Address offset: 0x30

Width	Name	Reset	Property	Description
31:16	Reserved	0	RW	Bits 15:0 **IMKy**: Port x bit y interrupt mask (y = 0..15). 如果IMKy置1，IO输入切换时会出现中断. 0:禁止中断 1:使能中断
15	IMK15	0	RW	
14	IMK14	0	RW	
13	IMK13	0	RW	
12	IMK12	0	RW	
11	IMK11	0	RW	
10	IMK10	0	RW	
9	IMK9	0	RW	
8	IMK8	0	RW	
7	IMK7	0	RW	
6	IMK6	0	RW	
5	IMK5	0	RW	
4	IMK4	0	RW	
3	IMK3	0	RW	
2	IMK2	0	RW	
1	IMK1	0	RW	
0	IMK0	0	RW	

9.3.22 寄存器地址映射

name	GPIOA_MOD	GPIOA_OTYP	GPIOA_OSPEED	GPIOA_OSPEEDH	GPIOA_PU	GPIOA_IDR	GPIOA_OD	GPIOA_BS	GPIOA_LC
31			OSPEED7						
30									
29									
28									
27			OSPEED6		PD11			BR11	
26					PD10			BR10	
25					PD9			BR9	
24					PD8			BR8	
23	MODER11		OSPEED5		PD7			BR7	
22					PD6			BR6	
21	MODER10				PD5			BR5	
20					PD4			BR4	
19	MODER9		OSPEED4		PD3			BR3	
18					PD2			BR2	
17	MODER8				PD1			BR1	
16					PD0			BR0	LCKK
15	MODER7		OSPEED3	OSPEED11					
14									
13	MODE								

12	R6								
11	MODE	OT11	OSPEED	OSPEED	PU11	IDR11	ODR1	BS11	LCK11
10	R5	OT10	2	10	PU10	IDR10	ODR1	BS10	LCK1
9	MODE	OT9			PU9	IDR9	ODR9	BS9	LCK9
8	R4	OT8			PU8	IDR8	ODR8	BS8	LCK8
7	MODE	OT7	OSPEED	OSPEED	PU7	IDR7	ODR7	BS7	LCK7
6	R3	OT6	1	9	PU6	IDR6	ODR6	BS6	LCK6
5	MODE	OT5			PU5	IDR5	ODR5	BS5	LCK5
4	R2	OT4			PU4	IDR4	ODR4	BS4	LCK4
3	MODE	OT3	OSPEED	OSPEED	PU3	IDR3	ODR3	BS3	LCK3
2	R1	OT2	0	8	PU2	IDR2	ODR2	BS2	LCK2
1	MODE	OT1			PU1	IDR1	ODR1	BS1	LCK1
0	R0	OT0			PU0	IDR0	ODR0	BS0	LCK0

name	**GPIOA_AFRL**	**GPIOA_AFRH**	**GPIOA_TGL**	**GPIOA_IM**	**GPIOB_MODE**	**GPIOB_OTY**	**GPIOB_OSPEED**	**GPIOB_OSPEEDH**
31	AFR7				MODER15		OSPEED7	OSPEED15
30								
29					MODER14			
28								
27	AFR6				MODER13		OSPEED6	OSPEED14
26								
25					MODER12			
24								
23	AFR5				MODER11		OSPEED5	OSPEED13
22								
21					MODER10			
20								
19	AFR4				MODER9		OSPEED4	OSPEED12
18								
17					MODER8			
16								
15	AFR3	AFR11			MODER7	OT15	OSPEED3	OSPEED11
14						OT14		
13					MODER6	OT13		
12						OT12		
11	AFR2	AFR10	TG11	IMK11	MODER5	OT11	OSPEED2	OSPEED10
10			TG10	IMK10		OT10		
9			TG9	IMK9	MODER4	OT9		
8			TG8	IMK8		OT8		
7	AFR1	AFR9	TG7	IMK7	MODER3	OT7	OSPEED1	OSPEED9
6			TG6	IMK6		OT6		
5			TG5	IMK5	MODER2	OT5		
4			TG4	IMK4		OT4		

3	AFR0	AFR8	TG3	IMK3	MODER1	OT3	OSPEED 0	OSPEED 8
2			TG2	IMK2		OT2		
1			TG1	IMK1	MODER0	OT1		
0			TG0	IMK0		OT0		

name	**GPIOB _PUPD**	**GPIO B_IDR**	**GPIO B_OD**	**GPIO B_BS**	**GPIO B_LC**	**GPIO B_AF**	**GPIO B_AF**	**GPIOB _TGL**	**GPIOB_I MK**
31	PD15			BR15		AFR7	AFR1 5		
30	PD14			BR14					
29	PD13			BR13					
28	PD12			BR12					
27	PD11			BR11		AFR6	AFR1 4		
26	PD10			BR10					
25	PD9			BR9					
24	PD8			BR8					
23	PD7			BR7		AFR5	AFR1 3		
22	PD6			BR6					
21	PD5			BR5					
20	PD4			BR4					
19	PD3			BR3		AFR4	AFR1 2		
18	PD2			BR2					
17	PD1			BR1					
16	PD0			BR0	LCKK				
15	PU15	IDR15	ODR1	BS15	LCK1	AFR3	AFR1 1	TG15	IMK15
14	PU14	IDR14	ODR1	BS14	LCK1			TG14	IMK14
13	PU13	IDR13	ODR1	BS13	LCK1			TG13	IMK13
12	PU12	IDR12	ODR1	BS12	LCK1			TG12	IMK12
11	PU11	IDR11	ODR1	BS11	LCK1	AFR2	AFR1 0	TG11	IMK11
10	PU10	IDR10	ODR1	BS10	LCK1			TG10	IMK10
9	PU9	IDR9	ODR9	BS9	LCK9			TG9	IMK9
8	PU8	IDR8	ODR8	BS8	LCK8			TG8	IMK8
7	PU7	IDR7	ODR7	BS7	LCK7	AFR1	AFR9	TG7	IMK7
6	PU6	IDR6	ODR6	BS6	LCK6			TG6	IMK6
5	PU5	IDR5	ODR5	BS5	LCK5			TG5	IMK5
4	PU4	IDR4	ODR4	BS4	LCK4			TG4	IMK4
3	PU3	IDR3	ODR3	BS3	LCK3	AFR0	AFR8	TG3	IMK3
2	PU2	IDR2	ODR2	BS2	LCK2			TG2	IMK2
1	PU1	IDR1	ODR1	BS1	LCK1			TG1	IMK1
0	PU0	IDR0	ODR0	BS0	LCK0			TG0	IMK0

nam e	**GPIOC _MODE**	**GPIOC _OTYP**	**GPIOC_ OSPEED**	**GPIOC_ OSPEED**	**GPIO C_PU**	**GPIO C_IDR**	**GPIO C_OD**	**GPIO C_BS**	**GPIO C_LC**

31			OSPEED 7						
30									
29									
28									
27			OSPEED 6						
26									
25					PD9			BR9	
24					PD8			BR8	
23			OSPEED 5		PD7			BR7	
22					PD6			BR6	
21					PD5			BR5	
20					PD4			BR4	
19	MODE R9		OSPEED 4		PD3			BR3	
18					PD2			BR2	
17	MODE R8				PD1			BR1	
16					PD0			BR0	LCKK
15	MODE R7		OSPEED 3						
14									
13	MODE R6								
12									
11	MODE R5		OSPEED 2						
10									
9	MODE R4	OT9			PU9	IDR9	ODR9	BS9	LCK9
8		OT8			PU8	IDR8	ODR8	BS8	LCK8
7	MODE R3	OT7	OSPEED 1	OSPEED 9	PU7	IDR7	ODR7	BS7	LCK7
6		OT6			PU6	IDR6	ODR6	BS6	LCK6
5	MODE R2	OT5			PU5	IDR5	ODR5	BS5	LCK5
4		OT4			PU4	IDR4	ODR4	BS4	LCK4
3	MODE R1	OT3	OSPEED 0	OSPEED 8	PU3	IDR3	ODR3	BS3	LCK3
2		OT2			PU2	IDR2	ODR2	BS2	LCK2
1	MODE R0	OT1			PU1	IDR1	ODR1	BS1	LCK1
0		OT0			PU0	IDR0	ODR0	BS0	LCK0

name	**GPIOC_AFRL**	**GPIOC_AFRH**	**GPIOC_TGL**	**GPIOC_IMK**
31	AFR7			
30				
29				

28				
27	AFR6			
26				
25				
24				
23	AFR5			
22				
21				
20				
19	AFR4			
18				
17				
16				
15	AFR3			
14				
13				
12				
11	AFR2			
10				
9			TG9	IMK9
8			TG8	IMK8
7	AFR1	AFR9	TG7	IMK7
6			TG6	IMK6
5			TG5	IMK5
4			TG4	IMK4
3	AFR0	AFR8	TG3	IMK3
2			TG2	IMK2
1			TG1	IMK1
0			TG0	IMK0

10 通信接口外设 CSI

10.1 串行外设接口和内部集成电路（SPI_IIC）

10.1.1 SPI 简介

SPI 接口广泛用于不同设备之间的板级通讯，如扩展串行 Flash，ADC 等。许多 IC 制造商生产的器件都支持 SPI 接口。

10.1.2 SPI 特性

1. 支持主模式和从模式工作.
2. 支持 4 种模式
   - 模式 0：时钟 idel 为 0，上升沿采样，下降沿出数据
   - 模式 1：时钟 idel 为 0，下降沿采样，上升沿出数据
   - 模式 2：时钟 idel 为 1，下降沿采样，上升沿出数据
   - 模式 3：时钟 idel 为 1，上升沿采样，下降沿出数据
3. 支持 8bit、16bit、24bit and 32bit 传输
4. 传输数据顺序 MSB 在前
5. 支持标准模式，三线模式，两数据线模式和四数据线模式。
   - 标准模式：通信线有 CLK,CS,IO0(MOSI),IO1(MISO),一个 CLK 传输 1bit 数据
   - 三线模式：通信线有 CLK,CS,IO0,接收和发送都通过 IO0, 一个 CLK 传输 1bit 数据
   - 两数据模式：通信线有 CLK,CS,IO0,IO1，接收和发送都通过 IO0 和 IO1,一个 CLK 传输 2bit 数据
   - 四数据模式：数据线有 CLK,CS,IO0,IO1,IO2,IO3，接收和发送都通过 IO0,IO1,IO2,IO3, 一个 CLK 传输 4bit 数据
6. 发送完一个数据起来一个中断
7. 支持 DMA

10.1.3 IIC 简介

IIC（Inter－Integrated Circuit）总线是由 PHILIPS 公司开发的两线式串行总线，用于连接微控制器及其外围设备。它提供多主模式功能，可以控制所有 IIC 总线特定的序列、协议、仲裁和时序。是微电子通信控制领域广泛采用的一种总线标准。

根据器件的不同，可利用 DMA 功能来减轻 CPU 的工作量。

10.1.4 IIC 特性

1. 支持主模式和从模式
2. 主模式支持时钟同步和仲裁
3. 从模式支持在发送数据没有准备好或者接收 buffer 满时候拉低 SCL

4. 从模式支持 7bit 地址或者 10bit 地址

6. 支持 DMA

10.1.5 SPI 时序图

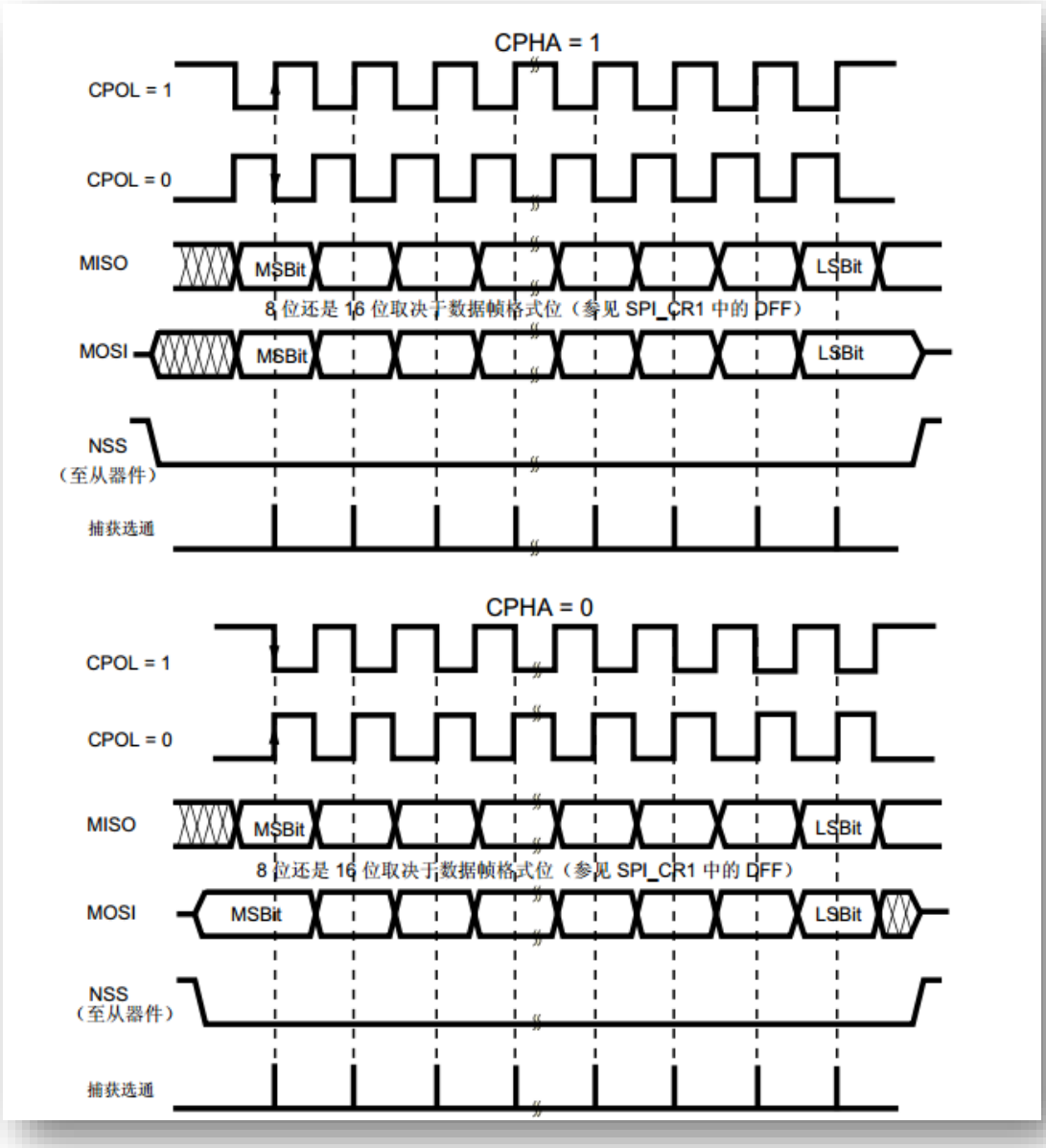


图 10-1 标准 SPI

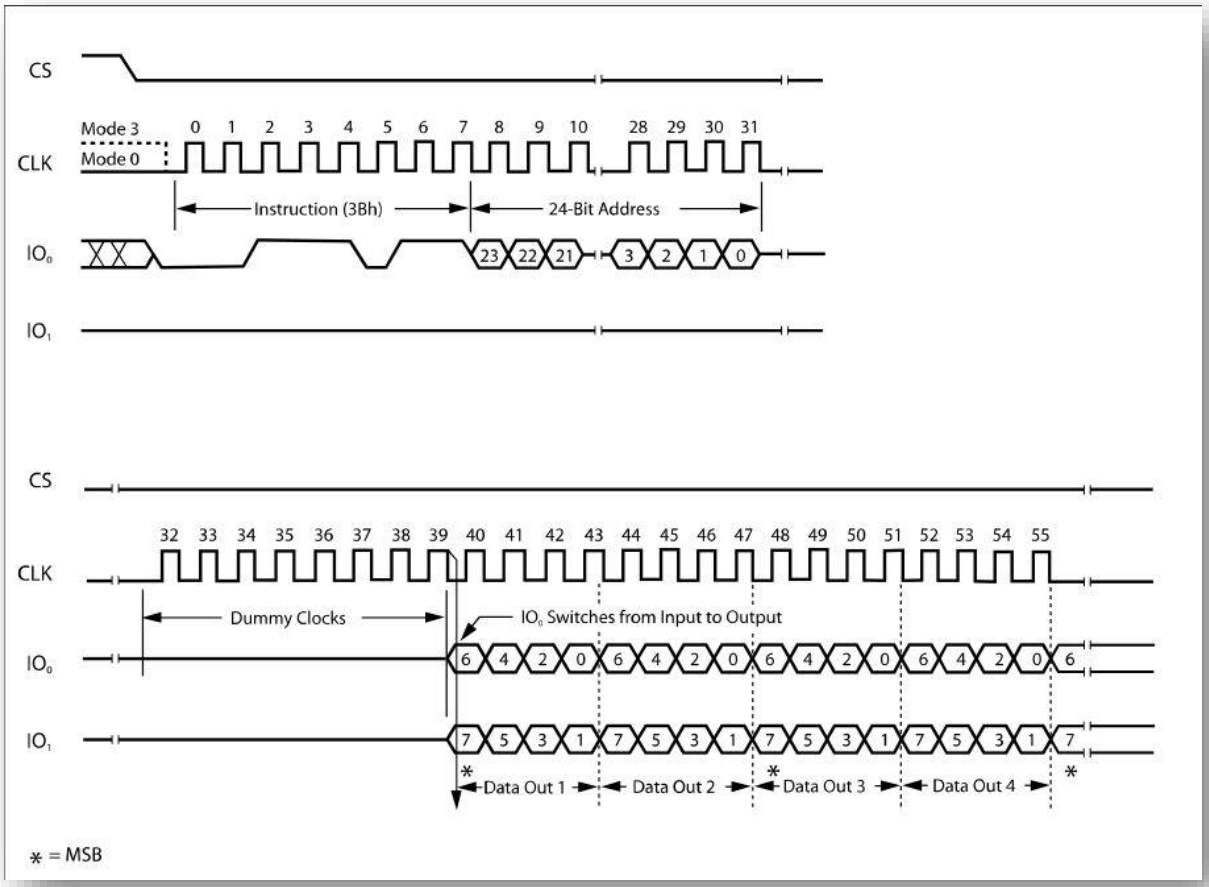


图 10-2 两数据线线 SPI

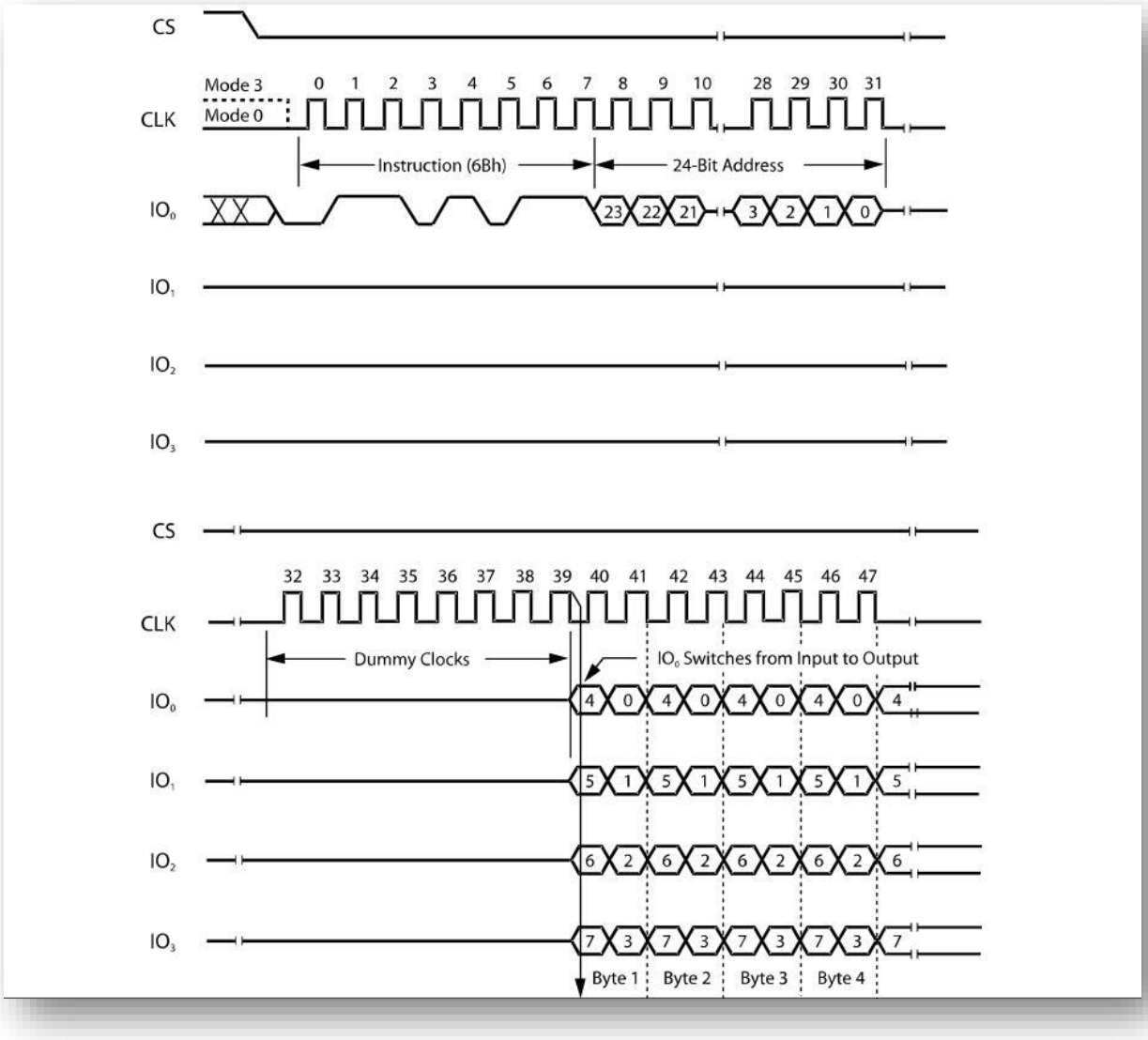


图 10-3 四数据线 SPI

10.1.6 IO 映射

SPI_IIC0:

SPI0_NSS： PA0(FUN0),PA11(FUN1)

SPI0_SCK/IIC0_SCL： PA1(FUN0),PB0(FUN1),PC0(FUN2)

SPI0_IO0/IIC0_SDA/MOSI： PA2(FUN0),PB1(FUN1),PC1(FUN2)

SPI0_IO1/MISO： PA3(FUN0),PB2(FUN1),PC2(FUN2)

SPI0_IO2： PA4(FUN0),PB3(FUN1)

SPI0_IO3： PA5(FUN0),PB4(FUN1)

SPI_IIC1:

SPI1_NSS： PA4(FUN1),PB3(FUN0)

SPI1_SCK/IIC1_SCL： PA5(FUN1),PB5(FUN1)

SPI1_IO0/IIC1_SDA/MOSI： PA6(FUN1),PB6(FUN1)

SPI1_IO1/MISO： PA7(FUN1),PB7(FUN1)

SPI1_IO2： PA8(FUN1),PB8(FUN1)

SPI1_IO3： PA9(FUN1),PB9(FUN1)

10.1.7 SPI/IIC 寄存器

Register 10-1 CON0（SPI 接口）

Name	Bit	Reset	ACCESS	Description
-	31:14	0	-	-
NSS_POS_IE	13	0	RW	SPI SLAVE SPI_NSS pin 上升沿中断使能 0：不使能 1：使能
SPI_NSS_EN	12	0	RW	SPI_NSS 引脚使能，次位仅在 SPI 模式下有效 0: 不使能, SPI 总线没有 NSS 引脚 1: 使能， SPI 总线有 NSS 引脚
SPI_NSS	11	0	RW	SPI NSS 引脚控制输出，次位仅在 SPI 主模式下有效 0: 输出低 1: 输出高
RXSEL	10：8	0	RW	SPI 主模式捕获串行数据 ： 3'd0: 没有延迟 3'd1: 延迟 1 个周期 3'd2: 延迟 2 个周期 ... 3'd7: 延迟 7 个周期
SLAVE_SYNC_EN	7	0	RW	SPI 从模式下，输入数据是否需要同步的选择位 0：不使能 1：使能
MASTER_SYNC_EN	6	0	RW	SPI 主模式下，输入数据是否需要同步的选择位 0：不使能 1：使能
DFS	5：4	0	RW	数据帧大小 00: 08 bit 01: 16 bit 10: 24 bit 11: 32 bit
WIREMODE	3：2	0	RW	SPI 线模式: 00: 标准模式 01: 三线模式 10: 两数据线模式 11: 四数据线模式
SPIMODE	1：0	0	RW	SPI 接口： SPI 模式选择 CPOL CPHA 00: mode 0：时钟 idel 为 0，上升沿采样，下降沿出数据

				01: mode 1：时钟 idel 为 0，下降沿采样，上升沿出数据 10: mode 2：时钟 idle 为 1，下降沿采样，上升沿出数据 11: mode 3：时钟 idel 为 1，上升沿采样，下降沿出数据

Register 10-2 CON0（IIC 接口）

Name	Bit	Reset	ACCESS	Description
-	31:23	0	-	-
RX_NACK_IE	22	0	RW	接收到 NACK 中断使能 0：不使能 1：使能
IIC_AL_IE	21	0	RW	主机仲裁丢失中断使能 0：不使能 1：使能
STOP_IE	20	0	RW	检测到线上有 STOP 信号中断使能，不管是从机主机都会起来 0：不使能 1：使能
ADR_MATCH_IE	19	0	RW	SLAVE 地址匹配中断使能 0：不使能 1：使能
IIC_FILTER_CNT	18:14	0	RW	IIC 每（IIC_FILTER_CNT +1）个 IIC 模块时钟，采样一次 SCL 和 SDA，用于滤除一定宽度的线路毛刺。IIC_FILTER_CNT 配置越大，虑掉的毛刺宽度越大。
BROADCAST_IE	13	0	RW	广播中断使能 0：不使能 1：使能
BROADCAST_EN	12	0	RW	IIC 作为 slave 时候，是否使能接收广播地址 0：忽略广播地址 1：接收到广播地址时，回应 ACK，并且将 BROADCAST_PEND 置起来，可以产生中断
SLAEV_ADR	11：2	0	RW	IIC 作为 SLAVE 时候，SLAVE 地址
TX_NACK	1	0	RW	IIC 在接收数据的时候，回应 NACK 还是 ACK 选择位 0：ACK 1：NACK
SLAVE_ADR_WIDTH	0	0	RW	IIC 作为 SLAVE 时候，SLAVE 地址宽度 0：7bit 1：10bit

Register 10-3CON1

Name	Bit	Reset	ACCESS	Description
-	31:10	0	-	-
DMA_IE	9	0	RW	DMA 完成中断使能 0：不使能 1：使能

BUF_OV_IE	8	0	RW	buffer 溢出，有数据丢失了中断使能
RBUF_NEMPTY_IE	7	0	RW	接收 buffer 不空中断使能 0：不使能 1：使能
TBUF_NFULL_IE	6	0	RW	发送 buffer 不满中断使能 0：不使能 1：使能
SSP_IE	5	0	RW	IIC 接口： IIC master 接收或者发送完成一帧数据（START + WRITE/READ(8bit+ack) + STOP）中断使能 IIC slave 接收或者发送完成一帧数据（8bit+ack）中断使能； 0：不使能 1：使能
DMA_EN	4	0	RW	发送 DMA 使能 0：不使能 1：使能 注：如果 SSP_TX_RX_EN==0，表示 DMA 接收；如果 SSP_TX_RX_EN==1，表示 DMA 发送；
SSP_TX_RX_EN	3	0	RW	接口发送使能 0：RX_EN 1：TX_EN
SLAVE	2	0	RW	SLAVE mode 1 = 从模式 0 = 主模式
SPI_IIC_SEL	1	0	RW	SPI 和 IIC 选择位 0：SPI 接口 1：IIC 接口
SSP_EN	0	0	RW	模块使能 0: 不使能 1: 使能

Register 100-4SSP_CMD_DATA

Name	Bit	Reset	ACCESS	Description
SSP_CMD_DATA	31:0	0	RW	SPI 接口： Write：将想要发送的数据写入这个寄存器，触发 SPI 发送 Read：读这个寄存器，获取收到的数据 IIC 接口： Write ： [7:0] 将想要发送的数据写入这 8bit； [8] START 使能位，在将这 byte 数据发出去前，先插入一个 start bit（只在 IIC_MASTER 模式有效）；

				[9] STOP 使能位，在将这 byte 数据发出去后，紧跟一个 stop bit（只在 IIC_MASTER 模式有效）； Read ：读这个寄存器，获取收到的数据（bit7~bit0） [31:10] 没有使用

Register 10-5BAUD

Name	Bit	Reset	ACCESS	Description
-	31:16	0	-	-
BAUD	15:0	0	RW	MASTER 模式时候： SPI 波特率=apb0_clock/(2*(BAUD+1)) IIC 波特率= apb0_clock/(4*（BAUD+1)) SLAVE 模式时候： SPI 没有用 IIC 用于定义从机准备好发送数据，再 delay(BAUD+1)个 apb0_clock 周期后才释放 SCL。

Register 10-6DMA_LEN

Name	Bit	Reset	ACCESS	Description
-	31:12	0	-	-
SPI_DMA_LEN	11:0	0	RW	接收和发送 DMA 模式：计划 DMA 的数据 byte 长度； 在 master 模式，RX_DMA_EN 为 0 时候，接收数据：表示计划接收的数据 byte 长度；

Register 100-7DMA_CNT

Name	Bit	Reset	ACCESS	Description
-	31:12	0	-	-
SPI_DMA_CNT	11:0	0	RO	接收和发送 DMA 模式：实际接收并且保存到 SRAM 的数据 byte 长度 在 master 模式，RX_DMA_EN 为 0 时候，接收数据：表示实际接收并且已经被 CPU 读走的数据 byte 长度； 注：在 RX_DMA_EN 上升沿，TX_DMA_EN 上升沿，以及

				MSTER 读上升沿的时候，被清零。

Register 10-8DMA_STADR

Name	Bit	Reset	ACCESS	Description
-	31:13	0	-	-
SPI_DMA_STADR	12:0	0	RW	DMA 地址，一帧数据是 32bit 时候，需要 32bit 对齐，其他时候可以任意配置

Register 10-9SPI_STA

Name	Bit	Reset	ACCESS	Description
-	31:28	0	-	-
SLV_ADDRED	27	0	RO	从机模式，从机被寻址标志位 1：从机已经被寻址 0：从机没有被寻址
STATE	26:24	0	RO	IIC STATE MASTER SLAVE 000：IDLE 空闲 空闲 001：START 发送 start 已经接收到 start，等待 SCL 变 0 010：TX 发送 1byte 数据 发送 1byte 数据 011：RX 接收 1byte 数据 接收 1byte 数据 100：STOP 发送 stop no use 101：ADR0 no use 等待接收 1byte 地址 110：ADR1 no use 等待接收 2byte 地址 111：no use
-	23:20	0	--	-
CLR_BUF_CNT	19	0	WO	Write0 ：没有任何作用 Write 1：清零 BUF_CNT Read：总是返回 0
BUF_CNT	18:16	0	RO	buffer 里面有多少 byte 有效数据，CPU 每读(RX_EN)/写(TX_EN)一次 CMD_DATA 寄存器，减/加一个帧数据宽度，8bit 数据减 1，16bit 数据减 2，24bit 和 32bit 数据减 4
MASTER_RX_BUSY	15	0	RO	在 MASTER 读从机模式，指示软件配置的连续读取 DMA_LEN byte 数据是否完成

				0：已经读完 1：还没有读完
IIC_RX_NACK	14	0	RO	IIC 主机或者从机，在发送数据的时候，在第 9bit 握手阶段，检测到 ACK 还是 NACK 0: ACK 1:NACK
IIC_SLAVE_RW	13	0	RO	IIC 从机，在地址阶段，接收到的主机发过来的读写标志 0：主机将写从机 1：主机将读从机
IIC_BUS_BUSY	12	0	RO	IIC 线路是否 busy 指示位 0：线路上一直没有出现 START，或者出现 START 后又出现了 STOP，线路空闲。 1：检测到线路上出现了 START,并且一直没有出现 STOP，线路忙。 注：只能硬件清
SPI_SLAVE_CS	11	1	RO	SPI SLAVE，CS 的状态
SSP_BUSY	10	0	RO	SPI 或者 IIC 正忙 0：MASTER 空闲 SLAVE 没有在发送数据，接收的时候不使用 1：MASTER 正在接收或者发送一帧数据 SLAVE 正在发送一帧数据，接收的时候不使用 注：硬件清，或者关掉 SPI 和 IIC（CON1[0]==0），或者在 slave 时候，CLR_BUF_CNT 会同时清掉 SSP_BUSY
AL_PEND	9	0	RC	IIC 主机，检测到仲裁丢失 0：没有仲裁丢失 1：仲裁丢失 注：只能软件清，必须清掉 IIC 才能正常工作
STOP_PEND	8	0	RC	IIC 接口，检测到线路上产生了 STOP 位 0：没有检测到 STOP 位。 1：检测到 STOP 位。 注：只能软件清
ADR_MTCH_PEND	7	0	RC	IIC 从机，接收到主机发送过来的对的从机地址 0：从机地址不匹配

				1：从机地址匹配 注：只能软件清
IIC_BROADCAST_PEND	6	0	RC	IIC 作为 SLAVE 时候，检测到广播地址标志位 1: 检测到广播地址 0：没有检测到广播地址 注：只能软件清
SPI_NSS_POS	5	0	RC	检测到 SPI NSS 引脚上升沿 0：没有检测到上升沿 1：检测到上升沿 注：只能软件清
DMA_PEND	4	0	RC	发送 DMA 或者接收 DMA 完成标志 0：DMA 没有完成 1：DMA 已经完成 注：只能软件清
BUF_OV	3	0	RC	buffer 溢出，有数据丢失了 0： buffer 没有溢出 1：在 buffer 已经是满状态（buffer 容量是 5byte，满并不一定是 5byte，而是空间容不下一帧数据了），又有数据接收到。丢弃后来的数据。 注：只能软件清
BUF_EMPTY	2	1	RO	buffer 空标志 0：不空 1：空
BUF_FULL	1	0	RO	buffer 满标志位 0：不满 1：满
SSP_DONE	0	0	RC	0：没有完成 1： SPI 接口 SPI 已经完成一帧（8bit/16bit/24bit/32bit）数据的接收或者发送 IIC 接口 IIC MASTER 已经完成了一帧数据的接收或者发送（START（可选） + WRITE/READ（8bit+ack） + STOP（可选））

				IIC SLAVE 已经完成了一帧数据的接收或者发送（8bit+ack） 注：只能软件清

10.1.8 用户操作流程

10.1.8.1 SPI 操作流程

SPI MASTER 初始化:

1.IO mapping 配置，将数据通路打通

2.配置 BAUD

3.配置 CON0

4.配置配置 CON1

5.SPI MASTER 使能：SPIx->CON1 | = 0x01；

SPI MASTER 非 DMA 发送:

1.SSP_TX_EN 使能：SPIx->CON1 |= (1<<3); //tx enable

2.若初始化时 SPI_NSS_EN,需要将 NSS 拉低：SPIx->CON0 &= ~(1<<11);//nss negedge

2.判断发送 buffer 是否 full，没有 full 就可以往 CMD_DATA 填数据

3.将所有想发的数据都填入 CMD_DATA 后，等待发送 buffer 空，以及 ssp_busy 为 0。

4.若初始化时 SPI_NSS_EN,需要将 NSS 拉高：SPIx->CON0 |= (1<<11);//nss posedge

5.结束。

SPI MASTER DMA 发送:

1.SSP_TX_EN 使能：SPIx->CON1 |= (1<<3); //tx enable

2.若初始化时 SPI_NSS_EN,需要将 NSS 拉低：SPIx->CON0 &= ~(1<<11);//nss negedge

2.配置 DMA_STADR,DMA_LEN

3.使能 DMA: SPIx->CON1 |= (1<<4);// DMA EN\

4.等待 DMA 结束：while((SPIx->STA & (1<<4))==0);//WAIT DMA PEND

5.若初始化时 SPI_NSS_EN,需要将 NSS 拉高：SPIx->CON0 |= (1<<11);//nss posedge

6.结束。

SPI MASTER 非 DMA 接收:

1.SSP_RX_EN 使能：SPIx->CON1 &= ~(1<<3); //rx enable

2.若初始化时 SPI_NSS_EN,需要将 NSS 拉低：SPIx->CON0 &= ~(1<<11);//nss negedge

2.配置 DMA_LEN 为想接收的数据的 byte 数

3.写 CMD_DATA 触发接收开始。

4.查询接收 buffer 为非空。通过读 DMA_DATA 取走数据。

5.重复 4，直到读走 DMA_LEN 的数据

6.若初始化时 SPI_NSS_EN,需要将 NSS 拉高：SPIx->CON0 |= (1<<11);//nss posedge

7.结束。

SPI MASTER DMA 接收:

1.SSP_RX_EN 使能：SPIx->CON1 &= ~(1<<3); //rx enable

2.若初始化时 SPI_NSS_EN,需要将 NSS 拉低：SPIx->CON0 &= ~(1<<11);//nss negedge

2.配置 DMA_LEN 为想接收的数据的 byte 数

3.使能 DMA: SPIx->CON1 |= (1<<4);//

4.等待 DMA 结束：while((SPIx->STA & (1<<4))==0);//WAIT DMA PEND

5.若初始化时 SPI_NSS_EN,需要将 NSS 拉高：SPIx->CON0 |= (1<<11);//nss posedge

6.结束。

SPI SLAVE 初始化:

1.IO mapping 配置，将数据通路打通

2.配置 CON0

3.配置 CON1

4.SPI SLAVE 使能：SPIx->CON1 | = 0x05；

SPI SLAVE 非 DMA 发送:

1.SSP_TX_EN 使能：SPIx->CON1 |= (1<<3); //tx enable

2.判断发送 buffer 是否 full，没有 full 就可以往 CMD_DATA 填数据

3.将所有想发的数据都填入 CMD_DATA 后，等待发送 buffer 空，以及 ssp_busy 为 0。

4.结束。

SPI SLAVE DMA 发送:

1.SSP_TX_EN 使能：SPIx->CON1 |= (1<<3); //tx enable

2.配置 DMA_STADR,DMA_LEN

3.使能 DMA: SPIx->CON1 |= (1<<4);// DMA EN\

4.等待 DMA 结束：while((SPIx->STA & (1<<4))==0);//WAIT DMA PEND

5.结束。

SPI SLAVE 非 DMA 接收:

1.SSP_RX_EN 使能：SPIx->CON1 &= ~(1<<3); //rx enable

2.查询接收 buffer 为非空。通过读 DMA_DATA 取走数据。

3.重复 2，直到读走所有需要的数据

4.结束。

SPI SLAVE DMA 接收:

1.SSP_RX_EN 使能：SPIx->CON1 &= ~(1<<3); //rx enable

2.配置 DMA_LEN 为想接收的数据的 byte 数

3.使能 DMA: SPIx->CON1 |= (1<<4);

4.等待 DMA 结束：while((SPIx->STA & (1<<4))==0);//WAIT DMA PEND

5.结束。

10.1.8.2 IIC 操作流程

IIC MASTER 初始化:

1.IO mapping 配置，将数据通路打通

2.配置 BAUD

3.配置 CON0

4.配置配置 CON1

5.SPI MASTER 使能：SPIx->CON1 | = 0x03；

IIC MASTER 非 DMA 发送:

1.SSP_TX_EN 使能：SPIx->CON1 |= (1<<3); //tx enable

2.判断发送 buffer 是否 full，没有 full 就可以往 CMD_DATA 填数据

3.将所有想发的数据都填入 CMD_DATA 后，等待发送 buffer 空，以及 ssp_busy 为 0。

4.结束。

IIC MASTER DMA 发送:

1.SSP_TX_EN 使能：SPIx->CON1 |= (1<<3); //tx enable

2.配置 DMA_STADR,DMA_LEN

3.使能 DMA: SPIx->CON1 |= (1<<4);

4.等待 DMA 结束：while((SPIx->STA & (1<<4))==0);//WAIT DMA PEND

5.结束。

IIC MASTER 非 DMA 接收:

1.SSP_RX_EN 使能：SPIx->CON1 &= ~(1<<3); //rx enable

2.配置 DMA_LEN 为想接收的数据的 byte 数

3.写 CMD_DATA 触发接收开始。

4.查询接收 buffer 为非空。通过读 DMA_DATA 取走数据。

5.重复 4，直到读走 DMA_LEN 的数据

6.结束。

IIC MASTER DMA 接收:

1.SSP_RX_EN 使能：SPIx->CON1 &= ~(1<<3); //rx enable

2.配置 DMA_LEN 为想接收的数据的 byte 数

3.使能 DMA: SPIx->CON1 |= (1<<4);

4.等待 DMA 结束：while((SPIx->STA & (1<<4))==0);//WAIT DMA PEND

5.结束。

IIC SLAVE 初始化:

1.IO mapping 配置，将数据通路打通

2.配置 CON0

3.配置 CON1

4.SPI SLAVE 使能：SPIx->CON1 | = 0x07；

IIC SLAVE 非 DMA 发送:

1.SSP_TX_EN 使能：SPIx->CON1 |= (1<<3); //tx enable

2.判断发送 buffer 是否 full，没有 full 就可以往 CMD_DATA 填数据

3.将所有想发的数据都填入 CMD_DATA 后，等待发送 buffer 空，以及 ssp_busy 为 0。

4.结束。

IIC SLAVE DMA 发送:

1.SSP_TX_EN 使能：SPIx->CON1 |= (1<<3); //tx enable

2.配置 DMA_STADR,DMA_LEN

3.使能 DMA: SPIx->CON1 |= (1<<4);// DMA EN\

4.等待 DMA 结束：while((SPIx->STA & (1<<4))==0);//WAIT DMA PEND

5.结束。

IIC SLAVE 非 DMA 接收:

1.SSP_RX_EN 使能：SPIx->CON1 &= ~(1<<3); //rx enable

2.查询接收 buffer 为非空。通过读 DMA_DATA 取走数据。

3.重复 2，直到读走所有需要的数据

4.结束。

IIC SLAVE DMA 接收:

1.SSP_RX_EN 使能：SPIx->CON1 &= ~(1<<3); //rx enable

2.配置 DMA_LEN 为想接收的数据的 byte 数

3.使能 DMA: SPIx->CON1 |= (1<<4);// DMA EN\

4.等待 DMA 结束：while((SPIx->STA & (1<<4))==0);//WAIT DMA PEND

5.结束。

11 通用异步收发器 UART

11.1 简介

通用异步收发器(UART)提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。UART 利用分数波特率发生器提供宽度范围的波特率选择。使用多缓冲器配置的 DMA 方式，可以实现高速数据通信。

11.2 UART 特性

UART0:

- 支持 8bit 数据和 9bit 数据模式
- 支持奇偶校验，奇校验/偶校验可选
- 支持检测系统更新数据串（0x70->0x61->0x73->0x53->0xf5->0x1e->0xf4->0xec）
- 具有 4 帧数据的接收缓存
- 硬件检测接收 time out，time out 长度固定 5 *（1 start_bit + 8 data bit + 2 stop bit）波特率时间。

UART1:

- 支持 8bit 数据和 9bit 数据模式
- 支持奇偶校验，奇校验/偶校验可选
- 具有 4 帧数据的接收缓存
- 支持接收和发送 DMA
- 支持 RS485 模式
- 硬件检测接收 time out，time out 长度固定 5 *（1 start_bit + 8 data bit + 2 stop bit）波特率时间。

11.3 IO 映射

PIN MAP:

UART0 (RX/TX): PB4(fun0), PB10(fun1), PC7(fun2)（双线工作时作为 RX，单线工作时为半双工模式）

UART0 TX: PB5(fun0), PB11(fun1), PC6(fun2)

UART0 用作监测系统更新功能时候，UART0_RX:

PA0,PA1,PA2,PA3,PA4,PA5,

PA6,AP7,PA8,PA9,PA10,PA11,

PB0,PB2,PB4,BP6,PB8,PB10,PB12,PB14

PC0,PC1,PC2,PC3,PC6,PC7

PIN MAP :

UART1 TX: PC0(fun0),PC6(fun0)

UART1(RX/TX): PC1(fun0),PC7(fun0)

UART1 RE: PC2(fun0)

UART1 DE: PC3(fun0)

11.4 **UART** 寄存器

Register 111-1UARTCON

Name	Bit	Reset	ACCESS	Description
-	31:15	0	-	-
TMR_PWM_EN	14	0	RW	UART 的输出和 TMR 的 PWM 一起运算后才输出使能位（UART0 输出 timer0 的 PWM,UART1 输出 timer1 的 PWM） 0 = 不使能 1 = 使能
TO_IE	13	0	RW	Time out 中断使能 0 = 不使能 1 = 使能
TO_EN	12	0	RW	Time out 检测使能位。检测是否在经历了 5 *（1 start_bit + 8 data bit + 2 stop bit）时间，没有接收到数据。每次 TO_EN 后，需要等到接收到一 byte 数据才会开始检测。接收到 1byte 数据这个标志，会在每次清 TO_PEND 的时候被清掉，或者在 TO_EN 等于 0 的时候被清掉。 0 = 不使能 1 = 使能
FERR_IE	11	0	RW	帧出错中断使能位 0 = 不使能 1 = 使能
TX_IE	10	0	RW	发送完成中断使能位 0 = 不使能 1 = 使能
RX_IE	9	0	RW	接收完成中断使能位 0 = 不使能 1 = 使能
TX_INV	8	0	RW	发送输出信号取反选择位 0 = 不使能 1 = 使能
RX_INV	7	0	RW	接收输入信号取反选择位 0 = 不取反 1 = 取反
ODD_EN	6	0	RW	奇偶校验选择 0 =偶校验 1 =奇校验

PARITY_EN	5	0	RW	奇偶校验使能 0 = 不使能 1 = 使能 注：奇偶检验和 BIT9_EN 只能二选一
BIT9_EN	4	0	RW	UART 传输 9bit 数据选择位 0 = UART 传输 8bit 数据 1 = UART 传输 9bit 数据 注：奇偶检验和 BIT9_EN 只能二选一
STOP_BIT	3	0	RW	结束位使能位 0 = 一个结束位 1= 两个结束位
WORK_MODE	2:1	0	RW	00 = 双工，可以同时收发 01 = 单工发送 10 = 单工接收 11 = 一根数据线，硬件自动切换该 IO 方向；在不是 rs485M 模式，软件触发发送 IO 方向就是输出，没有数据发送 IO 方向是输入。在 RS485 模式，DE 为有效时候 IO 是输出，在 DE 为无效的时候 IO 是输入。
UART_EN	0	0	RW	UART 模块使能位 0 = 不使能 1 = 使能

Register 11-2UARTBAUD

Name	Bit	Reset	ACCESS	Description
-	31:18	0	-	-
UARTBAUD	17:0	0xa93	RW	UART 波特率设置 波特率=SYS_CLK /(UARTBAUD+1) 注：UARTBAUD 一定要配置为大于等于 6，否则输入信号会被内部滤波器滤掉。

Register 11-3UARTDATA

Name	Bit	Reset	ACCESS	Description
-	-	0	-	-
UARTDATA	8:0	XXX	RW	UART 传输数据 写 8/9 位数据到 UARTDATA ：触发 UART 模块开始发送数据。 读 UARTDATA ： 获取接收到的低 8/9bit 数据，读之前，先读取 PERR[0]获取奇偶校验信息

Register 11-4UARTSTA

Name	Bit	Reset	ACCESS	Description
-	31:15	0	-	-
UPDATE_DETECT_E	14	0	RO	系统升级检测使能位。检测是否接收到了下面的一串连续数

N				据 0x70->0x61->0x73->0x53->0xf5->0x1e->0xf4->0xec 0 = 不使能 1 = 使能 注：当 UPDATE_DETECT_EN 为 1，并且 UPDATE_DETECT_PEND 为 0 时候 UARTCON：将固定为 0x1005（单工接收，1stop，8bit 模式，没有奇偶校验，没有取反，所有中断不使能）； UARTBAUD：将由 EFLASH 模块配置，默认值会是 0xa93; UART_DMACON：将固定为 0x00（不使能 DMA）； UART_RS485_CON：将固定为 0x00（不使能 rs485）； 软件不可以改变。
UPDATE_DETECT_PEND_G	13	0	RO	这一位和 UPDATE_DETECT_PEND 一样，只是这位一旦为 1，除非 reset UART0，否则一直不会变为 0.
UPDATE_DETECT_PEND	12	0	RC	检测到系统升级标志位。表示接收到了下面的一串数据 0x70->0x61->0x73->0x53->0xf5->0x1e->0xf4->0xec。写 1 清零。 0：没有检测到系统升级 1：检测到系统升级 注：只有 UART0 有该位。只能软件清。
TO_PEND	11	0	RC	Time out 标志位。表示经历了 5 *（1 start_bit + 8 data bit + 2 stop bit）时间，没有接收到数据。写 1 清零。 0：time out 没有出现 1：time out 出现 注：只有 UART0 有该位。
PERR	10:7	0	RO	接收数据奇偶校验出错，4bit，分别对应 BUF 里面的 4 个数据。在读 UARTDATA 之前，需要先读这个寄存器，硬件会自动将 PERR[0]对应你读 UARTDATA 获取的 data。
RX_CNT	6:4	0	RO	RX BUF 有多少个数据 000：0 个数据 001：1 个数据 010：2 个数据 011：3 个数据 100：4 个数据 其他：不允许
FERR	3	0	RC	帧错误。表示在 STOPbit 期间检测到 RX_IN 出现了低电平。写 1 清零 0：没有帧错误 1：出现了帧错误
RX_BUF_OV	2		RC	接收 BUF 最多可以容纳 4 个 8/9bit 的数据，接收到 4 个数据后，如果软件还没有来得及读走，又收到一个数据，这个标志为会起来。写 1 清零。 0：接收了(0~4byte)数据 1：接收了(>4byte)的数据，BUF 里面只保存最开始的 4byte，其他丢弃
RX_BUF_NOT EMPTY	1	0	RC	接收 BUF 不空标志位。写 1 清零。 0：接收 BUF 没有数据 1：接收 BUF 有数据

TX_DONE	0	0	RC	发送完成标志位。写 1 清零，或者写 UARTDATA 清 0。 0：发送没有完成 1：发送完

Register 11-5UART_TSTADR（UART0 没有该寄存器）

Name	Bit	Reset	ACCESS	Description
-	31:13	0	-	-
TSTADRT	12:0	0	RW	发送 DMA 起始地址； 如果是 9bit 数据，需要 16bit 对齐； 如果是 8bit 数据，可以任意配置。

Register 11-6UART_RSTADR（UART0 没有该寄存器）

Name	Bit	Reset	ACCESS	Description
-	31:13	0	-	-
RSTADRT	12:0	0	RW	接收 DMA 起始地址 如果是 9bit 数据，需要 16bit 对齐； 如果是 8bit 数据，可以任意配置。

Register 11-7UART_TDMALEN（UART0 没有该寄存器）

Name	Bit	Reset	ACCESS	Description
-	31:12	0	-	-
TDMALEN	11:0	0	RW	计划发送 DMA byte 长度。 8bit 数据模式，uart 一帧数据等于 1byte； 9bit 数据，uart 一帧数据等于 2byte，TDMALEN 需要配置成 2 的倍数。

Register 11-8UART_RDMALEN（UART0 没有该寄存器）

Name	Bit	Reset	ACCESS	Description
-	31:12	0	-	-
RDMALEN	11:0	0	RW	计划接收 DMA byte 长度。 8bit 数据模式，uart 一帧数据等于 1byte； 9bit 数据，uart 一帧数据等于 2byte，RDMALEN 需要配置成 2 的倍数。

Register 11-9UART_TDMACNT（UART0 没有该寄存器）

Name	Bit	Reset	ACCESS	Description
-	31:12	0	-	-

TDMACNT	11:0	0	RO	已经发送 DMA byte 长度。 8bit 数据模式，uart 一帧数据等于 1byte； 9bit 数据，uart 一帧数据等于 2byte。

Register 11-10UART_RDMACNT

Name	Bit	Reset	ACCESS	Description
-	31:12	0	-	-
RDMACNT	11:0	0	RO	已经接收 DMA byte 长度。 8bit 数据模式，uart 一帧数据等于 1byte； 9bit 数据，uart 一帧数据等于 2byte。

Register 11-11UART_DMACON（UART0 没有该寄存器）

Name	Bit	Reset	ACCESS	Description
-	31:8	0	-	-
RX_DMA_PERR	7	0	RC	接收 DMA 一包数据出现至少一个数据奇偶校验错误标志位 0：没有数据出现奇偶校验错误 1：出现了至少一个数据奇偶校验错误
RX_DMA_PEND	6	0	RC	接收 DMA 完成标志位。指数据接收完成并保存到 SRAM 0：DMA 没有完成 1：DMA 已经完成
TX_DMA_PEND	5	0	RC	发送 DMA 完成标志位。指数据从 SRAM 取出并且通过 uart 完整发送出去。 0：DMA 没有完成 1：DMA 已经完成
RX_DMA_PERR_IE	4	0	RW	接收 DMA 一包数据出现至少一个数据奇偶校验错误，中断使能 0：不使能 1：使能
RX_DMA_IE	3	0	RW	接收 DMA 中断使能 0：不使能 1：使能
TX_DMA_IE	2	0	RW	发送 DMA 中断使能 0：不使能 1：使能
RX_DMA_EN	1	0	RW	接收 DMA 使能 0：不使能 1：使能
TX_DMA_EN	0	0	RW	发送 DMA 使能 0：不使能 1：使能

Register 11-12UART_RS485_CON（UART0 没有该寄存器）

Name	Bit	Reset	ACCESS	Description
-	31:10	0	-	-
RE_EN	9	0	RW	RE 使能 1：使能 0：不使能
DE_EN	8	0	RW	DE 使能 1：使能 0：不使能
-	7：4	0	0	0
RS485_MODE	3	0	RC	RS485 工作模式 1：硬件自动切换 DE 和 RE 模式，有数据需要发送切换到 DE，没有数据发送保持 RE 0：软件配置 DE_EN，DE 会一直有效，直到配置 DE_EN 为 0；软件配置 RE_EN，RE 会一直有效，直到配置 RE_EN 为 0；硬件会自动插入 DE_DAT，DE_AT，RE2DE_T，DE2RE_T 时间。这一位等于 0 的时候，DE_EN 和 RE_EN 不能同时置 1。
RE_POL	2	0	RC	RE 极性 1：低电平有效 0：高电平有效
DE_POL	1	0	RW	DE 极性 1：低电平有效 0：高电平有效
RS485_EN	0	0	RW	RS485 使能 1：使能 0：不使能

Register 11-13UART_RS485_DET（UART0 没有该寄存器）

Name	Bit	Reset	ACCESS	Description
-	31:25	0	-	-
DE_DAT	24:16	0	RW	STOP BIT 结束到 DE 无效之间的时间间隔，单位是 uart 模块时钟
-	15:9	0	-	-
DE_AT	8:0	0	RW	DE 有效到发送 START BIT 之间的时间间隔，单位是 uart 模块时钟

Register 11-14UART_RS485_TAT（UART0 没有该寄存器）

Name	Bit	Reset	ACCESS	Description
RE2DE_T	31:16	0	RW	RE 有效到 DE 有效之间的时间间隔，单位是 uart 模块时钟
DE2RE_T	15:0	0	RW	DE 有效到 RE 有效之间的时间间隔，单位是 uart 模块时钟

11.5 操作流程

DMA 模式不使能的 UART 发送：

1. 配置 UARTBAUD.
2. 配置 UART_RS485_CON,UART_RS485_DAT,UART_RS485_TAT;
3. 配置 UARTCON
4. 将需要发送的 8bit/9bit 数据写入 UARTDATA;
5. 等待发送完成:等待 TX_DONE(UARTSTA[0])会置 1
6. 如果还需要发送数据，重复步骤 4 和 5。

DMA 模式使能的 UART 发送：

1. 配置 UARTBAUD.
2. 配置 UART_RS485_CON,UART_RS485_DAT,UART_RS485_TAT;
3. 配置 UARTCON
4. 配置 DMACON,DMA_TSTADR,DMA_TDMALEN;
5. 等待 DMA 完成:UART_DMA_CON[0]清零，或者 UART_DMA_CON[5]为 1；

DMA 模式不使能的 UART 接收：

1. 配置 UARTBAUD.
2. 配置 UART_RS485_CON,UART_RS485_DAT,UART_RS485_TAT;
3. 配置 UARTCON
4. 等待接收完成:接收完成后，UARTSTA[1] ：RX_BUF_NOT EMPTY 会置 1；
5. 读取接收到的数据，通过读 UARTDATA 获取接收到的数据。
6. 如果还需要接收数据，重复步骤 4 和 5

DMA 模式使能的 UART 接收：

1. 配置 UARTBAUD.
2. 配置 UART_RS485_CON,UART_RS485_DAT,UART_RS485_TAT;
3. 配置 UARTCON
4. 配置 DMACON,DMA_RSTADR,DMA_RDMALEN;
5. 等待 DMA 完成:UART_DMA_CON[1]清零，或者 UART_DMA_CON[6]为 1；

12 比较器（COMP）

12.1 简介

芯片内嵌两个通用比较器 COMP0 和 COMP1，可独立使用，也可与定时器结合使用。比较器是将一个模拟电压信号与一个基准电压相比较的电路。比较器的两路输入（正极输入和负极输入）为模拟信号，输出则为二进制信号 0 或 1，当输入的电压的差值增大或减小且正负符号不变时，其输出保持恒定。

12.2 主要特性

- 轨对轨比较器
- 可复用 I/O，内部一端连接到 DAC 上
- 可编程迟滞电压
- 支持比较结果的滤波功能
- 输出端可以重定向到一个 I/O 端口或多个定时器输入端，可以触发定时器的捕获事件
- COMP0 有 4 个正向输入和 4 个反向输入
- COMP1 的正向输入可选择不同 IO 端口，反向输入连接到内部 12bit DAC 输出
- 每个比较器都可产生中断，并支持把 CPU 从睡眠模式和停机模式唤醒

12.3 功能描述

12.3.1 功能框图

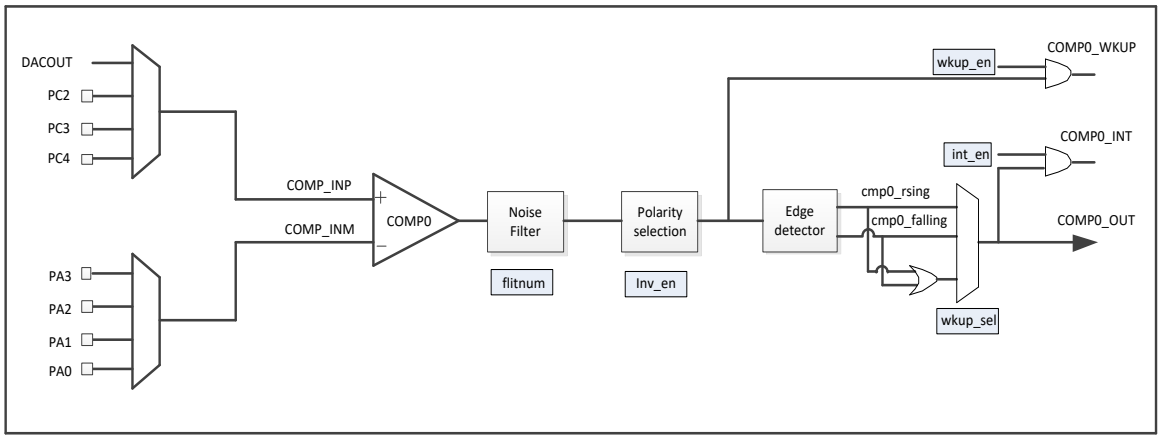


图 12-1 比较器 0

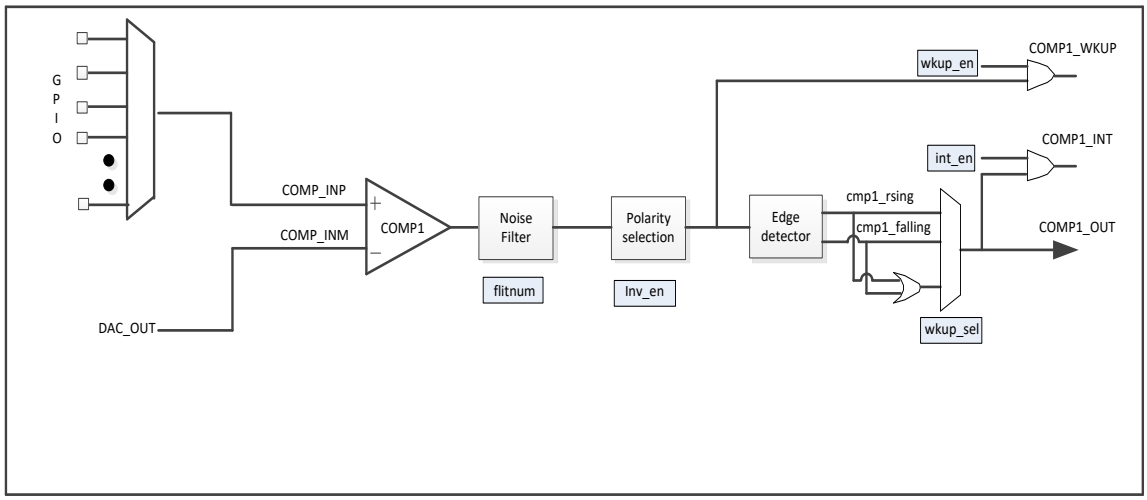


图 12-2 比较器 1 框图

12.3.2 比较器 0 输入和输出

作为比较器输入的 I/O 引脚必须在 GPIO 寄存器中设置为模拟模式。

比较器的输出可以内部重定向到各种定时器的输入：作为输入捕获，测量时序。

比较器的输出可以内部重定向到 DAC 的输入。

其中 DACOUT 为内部 6BitDAC 输出。

12.3.3 比较器 1 输入和输出

比较器 1 的正向输入可选择不同 IO 端口，反向输入连接到内部 DAC 输出。

其中 DAC_OUT 为内部 12BitDAC 输出。

12.3.4 中断和唤醒

比较器的输出可以内部连接到外部中断和时间控制器。每个比较器都有一个输出唤醒信号，能产生中断或事件，同样的机制能用来退出低功耗模式。

12.3.5 比较器锁定机制

比较器能用于安全的用途，比如过流保护或者过热保护。在某些特定的安全需求的应用中，有必要保证比较器设置不能被无效寄存器访问或者程序计数器破坏所改变。

为了这个目的，比较器控制和状态寄存器可以设为写保护。

一旦设置完成，LOCK 位必须设为 1，这导致整个 CMP_CON0 寄存器变为只读，包括 LOCK 位在内。

写保护只能被 MCU 复位清除。

12.3.6 比较器 0 迟滞现象

比较器 0 的可配置迟滞电压能防止无效的输出变化产生的噪声信号。在不需要强制迟滞电压的情况下迟滞现象可以被禁止。

12.4 寄存器描述

12.4.1 配置寄存器 0 (CMP_CON0)

Address offset: 0x00

Width	Name	Reset	Property	Description
31	lock	0	RW	比较器锁 该位只能写一次，由软件置“1”，由系统复位清零。 它令比较器的所有控制位为只读。 1：CMP_CON0 只读 0：CMP_CON0 可读可写
30:19	reserved	0	-	-
18:17	wkup_sel	0	RW	唤醒选择： 01：比较器 0 上升沿唤醒 10：比较器 0 下降沿唤醒 11：比较器上升沿和下降沿都能唤醒 others：保留
16:12	filt_num	0	RW	比较器输出滤波周期选择
11:10	hy_sel	0	RW	比较器电压迟滞选择： 00：没有迟滞 01：22*2mv 10：34*2mv 11：42*2mv
9:8	ch_psel	0	RW	输入正极选择： 00=DACOUT 01=PC2 10=PC3 11=PC4
7:6	ch_nsel	0	RW	输入负极选择： 00=PA3 01=PA2 10=PA1 11=PA0
5	reserved	0	-	-
4	lowp	0	RW	比较器低功耗使能
3	inv_en	0	RW	输出结果取反使能
2	wkup_en	0	RW	唤醒使能
1	int_en	0	RW	中断使能
0	enable	0	RW	比较器使能

12.4.2 配置寄存器 1 (CMP_CON1)

Address offset: 0x04

Width	Name	Reset	Property	Description
31	lock	0	RW	比较器锁 该位只能写一次，由软件置“1”，由系统复位清零。 它令比较器的所有控制位为只读。 1：CMP_CON0 只读

				0：CMP_CON0 可读可写
30:19	reserved	0	-	-
18:17	wkup_sel	0	RW	唤醒选择： 01：比较器 0 上升沿唤醒 10：比较器 0 下降沿唤醒 11：比较器上升沿和下降沿都能唤醒 others：保留
16:12	filt_num	0	RW	比较器输出滤波周期选择
11:4	reserved	0	-	-
3	inv_en	0	RW	输出结果取反使能
2	wkup_en	0	RW	唤醒使能
1	int_en	0	RW	中断使能
0	enable	0	RW	比较器使能

12.4.3 状态寄存器 (CMP_STA)

Address offset: 0x08

Width	Name	Reset	Property	Description
31:2	reserved	0	-	-
1	cmp1_pend	0	RO	比较器 1 输出标志
0	cmp0_pend	0	RO	比较器 0 输出标志

12.4.4 状态寄存器 (CMP_CLR)

Address offset: 0x0C

Width	Name	Reset	Property	Description
31:2	reserved	0	-	-
1	cmp1_pend_clr	0	WO	写“1”清比较器 1 输出标志
0	cmp0_pend_clr	0	WO	写“1”清比较器 0 输出标志

12.5 操作流程

1. 配置寄存器 CMP_CONx 选择比较器的正极和负极的输入引脚；
2. 根据比较器的工作方式配置对应模块功能（唤醒、输出到 IO、启动 ADC 采样等）；
3. 配置寄存器 CMP_CONx 使比较器工作；
4. 可以通过读取状态寄存器（CMP_STA）可以的到比较器的输出标志信号，并且配置状态寄存器(CMP_CLR）清除输出标志信号。

13 运算放大器（OPA）

13.1 简介

运算放大器是具有很高放大倍数的电路单元。在实际电路中，通常结合反馈网络共同组成某种功能模块。它是一种带有特殊耦合电路及反馈的放大器。

芯片内嵌一个运算放大器，运算放大器的输入和输出都连接到 I/O，通过共享 I/O 可以与 ADC、比较器相连。

13.2 主要特征

输出连接到 I/O 上。

13.3 寄存器描述

13.3.1 配置寄存器 (OPAM_CON)

Address offset: 0x00

Width	Name	Reset	Property	Description
31	lock	0	RW	运放锁 该位只能写一次，由软件置“1”，由系统复位清零。它令比较器的所有控制位为只读。 1：AMP_CON 只读 0：AMP_CON 可读可写
30	trdout	0	RO	AMP 输出
29:16	reserved	0	-	-
15:8	trims	0	RW	适配校准配置
7	trimen	0	RW	适配校准使能
6	reserved	0	-	-
5	lowp	0	RW	OPA 低功耗使能
4	enable	0	RW	OPA 使能，置“1”使能
3:2	binx_sel	0	RW	负极输入选择 00=PA0 01=PC0 10=PC1 11=PC5
1:0	ainx_sel	0	RW	正极输入选择 00=reserved 01=PA2 10=PA1 11=PC4

13.4 操作流程

例如：OPAM trimming Offset 测试

测试方法：

VCC 采用外部直流电源供电，打开 OPAM_TRIMEN_15V=1'b1;把 6Bit DAC 打开，输出 6'b000000

电压到 OPAM AIN0，然后调整 trimming 值，直到 OPAM_TRDOUT_15V 跳变为高则停止；记录下 trimming 值。

========== 获取适配校准值==========

```
/OPAM->CON |= BIT(7);       //OPA trim enable
OPAM->CON |= BIT(4);       //OPA enable
DAC->CON = (DAC->CON & ~(0x3f<<2)) | 0x1<<2;          //DAC data
OPAM->CON =(OPAM->CON & ~(0xff<<8)) | 0x19<<8 ;       //DAC trim
DAC->CON |= BIT(0);     //DAC enable
```

========== trimming 到值后，把值写回寄存 ==========

trimming 到值后，通过程序配置把值写回寄存器，然后把 opam 接成单位增益运放，AIN0 外部接一个固定电压 0.5/2.5/4.5，把 BIN0 和 OPAM_VOUT 短接一起，然后测量 VOUT 的电压值，都用同样万用表测试。

```
OPAM->CON |= BIT(4);                                    //OPA enable
OPAM->CON =(OPAM->CON & ~(0xFF<<8)) | 0x19<<8 ;   //trim to OPA
OPAM->CON =(OPAM->CON & ~3) | 2;                        //AIN0    select    PA1
OPAM->CON =(OPAM->CON & ~(3<<2) )| 2<<2;            //BIN0    select PC1
GPIOA->MODE |= GPIOA->MODE & (0x3<<2);              //PA1 analog
GPIOC->MODE |= GPIOC->MODE & (0x3<<2);              //PC1 analog
OPA output to PA2
ADKEY->STA    = (30<<4);                                //ADC select OPMA
ADKEY->CFG |= (1<<23);                                //output to     PA2
GPIOA->MODE |= (3<<4);                                  //PA2 analog
```

14 数模转换器（DAC）

14.1 简介

芯片内嵌两个 DAC，DAC0 是 6 位数字输入，电压输出的数模转换器。DAC1 是 12 位数字输入，电压输出的数模转换器。

14.2 主要特征

- DAC 输出可重定向到 PA5
- DAC0 可输出到 COMP0 的正端 AIP0，运放的 AIN0 端，以及 ADC 内部 channel 的第 3 端。
- DAC1 输出只能连接到 COMP1 的负端

14.3 寄存器描述

14.3.1 配置寄存器 (DAC_CON)

Address offset: 0x00

Width	Name	Reset	Property	Description
31	lock	0	RW	DAC 寄存器锁 该位只能写一次，由软件置“1”，由系统复位清零。它令比较器的所有控制位为只读。 0：DAC_CON 可读可写 1：DAC_CON 只读
30:8	reserved	0	-	-
7:2	dat_sel	0	RW	DAC 输入 data
1	diven	0	RW	VCC 除 4 使能位，置“1”有效
0	enable	0	RW	DAC 使能位，置“1”有效

DAC1 配置寄存器，参考 ADC 章节 ADKEY_CFG 和 ADKEY_DADAT 寄存器。

15 模数转换器(ADC)

15.1 功能简介

该模块是一个12bit的逐次逼近式的ADC控制器。ADC支持多种工作模式：单次转换和连续转换，并且可选择通道自动扫描。ADC的启动包括软件启动、外部引脚触发以及其他片内外设启动（timer触发启动）。

15.2 主要特征

- 最高12位可编程分辨率的SAR ADC，多达26路外部输入通道和7路内部通道
- 高达150Ksps转换速率
- 支持多种工作模式
  - 单次转换模式：A/D转换在指定通道完成一次转换
  - 单周期扫描模式：A/D转换在所有指定通道完成一个周期（从低序号通道到高序号通道或者从高序号通道到低序号通道）转换
  - 连续扫描模式：A/D转换连续执行单周期扫描模式直到软件停止A/D转换
- 通道采样时间，分辨率可软件配置
- 支持DMA传输。
- A/D转换开始条件
  - 软件启动
  - 外部IO触发启动
  - 内部timer触发启动

15.3 功能描述

15.3.1 ADC 开关控制

通过设置 ADCFG 寄存器的 ADEN 位可给 ADC 上电。当第一次设置 ADEN 位时，它将 ADC 从断电状态下唤醒。ADC 上电延迟一段时间后（tSTAB），设置采样模式、通道之后，设置 ADST 位开始进行转换。通过清除 ADST 位可以停止转换，设置 ADEN 位可置于断电模式。

15.4 通道选择

包含 26 路外部输入通道、7 路内部通道。每个外部输入通道都有独立的使能位，可通过设置 ADCHS 寄存器的对应位来设置。

15.4.1 ADC 工作模式

15.4.1.1 单次转换模式

在单次转换模式下，A/D 转换相应通道上只执行一次，具体流程如下：

- 软件配置 ADC_EN，给 ADC 上电。
- 通过软件、外部触发输入及定时器溢出置位 ADST，开始 A/D 转换。
- 当 A/D 转换完成，A/D 转换的数据值将存储于 A/D 的数据寄存器 ADDATA 中。
- A/D 转换完成，状态寄存器 ADSTA 的 ADIF 位置‘1’。若此时控制寄存器 ADCRL 的 ADIE 位置‘1’，将产生 AD 转换结束中断请求。
- A/D 转换期间，ADST 位保持为 1。A/D 转换结束，ADST 位自动清 0，A/D 转换器进入空闲模式。

15.4.1.2 单周期扫描模式

在单周期扫描模式下，将进行一次从被使能的最小序号通道向最大序号通道的 A/D 转换，可通过配置寄存器位 SCAN_DIR 选择扫描通道方向，操作步骤如下：

- 配置为 dma 模式。
- 软件或外部触发置位 ADST 开始，外部触发可软件配置触发延时，方向设置默认从最小序号通道到最大序号通道的 A/D 转换。
- 每路 A/D 转换完成后，A/D 转换数值将有序装载到相应的 DRAM 中，ADIF 转换结束标志被设置，如果设置了转换结束中断，则在所有通道转换都完成后产生中断请求。
- 转换结束后，ADST 位自动清 0，A/D 转换器进入空闲状态。

15.4.1.3 连续扫描模式

在连续扫描模式下，A/D 转换在 ADCHS 寄存器中的 CHENn 位被使能的通道上顺序进行（可通过配置寄存器位 SCAN_DIR 选择扫描通道方向），操作步骤如下：

- 软件或外部触发置位 ADST 开始，外部触发可软件配置触发延时，方向设置默认从最小序号通道到最大序号通道的 A/D 转换。
- 每路 A/D 转换完成后，A/D 转换数值将有序装载到相应的 DRAM 中。
- 当所有通道的 A/D 转换完成一遍后，ADIF 转换结束标志被设置。如果设置了转换结束中断，则在所有通道转换都完成后产生中断请求。
- 只要 ADST 位保持为 1，就重复步骤 2 到 3。当 ADST 位被清 0，A/D 转换停止，A/D 转换器进入空闲状态。当 ADST 清 0，A/D 转换结束。

15.4.2 DMA 请求

单周期扫描和连续扫描时最近一次转换的结果会保存在 ADDATA 寄存器中。DMA 传输时传输所有扫描通道的结果顺序存放到对应 DRAM 中，并且由 DATCNT 寄存器指示存放的采样个数。

15.4.3 采样频率设置

ADC 的时钟 ADCLK 由 PCLK 分频得到，分频系数可通过设置 ADCFG 寄存器的 ADCPRE 位来确定，即 PCLK/(N+1)分频后作为 ADC 时钟。

ADC 采样率= F_{pc1k} /(ADCPRE+1)/15

15.4.4 连续采样间隔时间可配置

对于连续扫描模式，连续两个通道采样之间可以配置采样延时（即在一个通道扫描完成之后，等待一段时间再进行下一通道扫描）。

15.4.5 外部触发转换

ADC 转换可以由外部事件触发（例如定时器、IO）。如果设置了 ADCR 寄存器的 TRGEN 位，就可以使用外部事件触发转换。通过设置 TRGSEL 位可以选择外部触发源。

具体的外部触发源选择情况，可以参考 AD 控制寄存器（ADCR）位[17]和[6：4] TRGSEL 的描述。

外部触发可设置延时控制，具体参考 ADCR[20:18]的 TRGSHIFT 的描述。

在触发信号产生后，延时 N 个 PCLK 的时钟周期再开始采样。如果是触发扫描模式，则只有第一个通道采样被延时，其他通道是在上一个采样结束后立即开始。

15.5 寄存器定义

15.5.1 配置寄存器 (ADKEY_CFG)

Address offset: 0x00

Width	Name	Reset	Property	Description
31:24	Reserved	0	RO	
25:23	ADDA_SET	0	RW	保留配置
22:17	Reserved	0	RO	
16:13	D2DCYC	0	RW	两次连续采样之间的间隔时间周期（n*ADCCLK）
12:10	Reserved	0	RO	
9:4	ADCPRE	‘hf	RW	ADC 预分频（ADC prescaler） 0=2 分频； 其他=（n+1）分频
3	Reserved	0	RO	
2	DACEN	0	RW	DAC 使能
1	Reserved	0	RO	
0	ADEN	0	RW	A/D 转换使能（ADC enable） 1 = 使能 0 = 禁用

15.5.2 数据寄存器 (ADKEY_CR)

Address offset: 0x04

Width	Name	Reset	Property	Description
31:29	Reserved	0	RO	
28	CALEN	1	RW	硬件数据校正使能，置“1”有效
27	SW_RST	0	WO	软件复位模块内部状态机，置“1”有效
26	DATASIGN	0	RW	数据扩张位选择，影响 ADKEY_DATA 的最高 3 位或最低 3 位
25	OVRIE	0	RW	overrun 中断使能
24	EOCS	0	RW	设置 ADIF 标志位，只有 scan mode 的时候有效： 1=在 scan mode 结束的时候产生 ADIF； 0=每次 sequence 完成产生 ADIF；
23:22	Reserved	0	RO	
21:19	TRGSHIFT	0	RW	外部触发延时采样（External trigger shift sample） 在触发信号产生后，延时 N 个 PCLK 的时钟周期再开始采样。 0：不延时 1：4 个周期 2：16 个周期 3：32 个周期 4：64 个周期 5：128 个周期 6：256 个周期 7：512 个周期
18:17	Reserved	0	RO	
16	SCANDIR	0	RW	ADC 扫描通道顺序（ADC scan direction） 在单周期扫描或者连续扫描方式时，设置扫描通

				道的顺序 0：ADC 通道选择寄存器按从低到高的顺序扫描 1：ADC 通道选择寄存器按从高到低的顺序扫描
15:12	Reserved	0	RO	
11	ALIGN	0	RW	数据对齐（Data alignment） 0：左对齐 1：右对齐
10:9	ADMD	0	RW	A/D 转换模式（ADC mode） 00：单次转换 01：单周期扫描 10：连续扫描 当改变转换模式时，软件要先禁用 ADST 位。
8	ADST	0	RW	ADST：A/D 转换开始（ADC start） 1 = 转换开始 0 = 转换结束或进入空闲状态 ADST 置位有下列两种方式： 在单次模式或者单周期模式下，转换完成后，ADST 将被硬件自动清除； 在连续扫描模式下，A/D 转换将一直进行，直到软件写 0 到该位或系统复位。
7:4	TRGSEL	0	RW	外部触发源选择 00 ： timer0 触发 01 ： timer1 触发 02 ： timer2 触发 03 ： timer3 触发 04 ： timer4 触发 05 ： timer5 触发 13 ： PB7 触发 14 ： PB6 触发 15 ： PC8 触发 其他 ： 保留
3	DMAEN	0	RW	DMA 使能（Direct memory access enable） 1 = DMA 请求使能 0 = DMA 禁止
2	TRGEN	0	RW	外部硬件触发源（External trigger enable） 1 = 使用外部触发信号启动 A/D 转换 0 = 不用外部触发信号启动 A/D 转换
1	Reserved	0	RO	
0	ADIE	0	RW	A/D 中断使能（ADC interrupt enable） 1 = 使能 A/D 中断 0 = 禁用 A/D 中断 如果 ADIE 置位，A/D 转换结束后产生中断请求。

15.5.3 数据寄存器 (ADKEY_CHS)

Address offset: 0x08

Width	Name	Reset	Property	Description

31:0	CHxEN	0	RW	连续扫描模式下 ADC 输入通道使能（Analog input channel 0 enable） 1 = 使能 0 = 禁用 CHxEN[0] ： PC0 CHxEN[1] ： PC1 CHxEN[2] ： PC2 CHxEN[3] ： PC3 CHxEN[4] ：PC6 CHxEN[5] ：PC7 CHxEN[6] ： PB14 CHxEN[7] ： PB12 CHxEN[8] ： PB10 CHxEN[9] ： PB8 CHxEN[10] ： PB6 CHxEN[11] ： PB4 CHxEN[12] ： PB2 CHxEN[13] ： PB0 CHxEN[14] ： PA11 CHxEN[15] ：PA10 CHxEN[16] ： PA9 CHxEN[17] ： PA8 CHxEN[18] ： PA7 CHxEN[19] ： PA6 CHxEN[20] ： PA5 CHxEN[21] ： PA4 CHxEN[22] ： PA3 CHxEN[23] ： PA2 CHxEN[24] ： PA1 CHxEN[25] ： PA0 CHxEN[26] ： BG CHxEN[27] ：温度 CHxEN[28] ： VCC/4 CHxEN[29] ：DAC 电压 CHxEN[30] ： OPMA CHxEN[31] ： VSS

15.5.4 数据寄存器 (ADKEY_STA)

Address offset: 0x10

Width	Name	Reset	Property	Description
31:10	Reserved	0	RO	
9:4	CHANNEL	0	RW	当前转换通道（Current conversion channel） 该 6 位在 BUSY = 1 时表示进行转换中的通道。 BUSY = 0 时表示可进行下次转换的通道。
3	Reserved	0	RW	
2	BUSY	0	RW	忙/空闲（Busy）

				1 = A/D 转换器忙碌 0 = A/D 转换器空闲
1	Reserved	0	RW	
0	ADIF	0	RW	A/D 转换结束标志位（ADC interrupt flag） 该位由硬件在通道组转换结束时置位，由软件清除。 1 = A/D 转换完成 0 = A/D 转换未完成 该标志位写‘1’清零。

15.5.5 数据寄存器 (ADKEY_DMAADR)

Address offset: 0x14

Width	Name	Reset	Property	Description
31:0	DMAADR	0	RW	DMA 起始地址

15.5.6 数据寄存器 (ADKEY_DATA)

Address offset: 0x18

Width	Name	Reset	Property	Description
31:24	Reserved	0	RO	
23	VALID	0	RO	有效标志位（只读）（Valid flag） 1 = DATA[11：0]位数据有效 0 = DATA[11：0]位数据无效 相应模拟通道转换完成后，将该位置位，读 ADDATA 寄存器后，该位由硬件清除.
22	OVERRUN	0	RO	数据覆盖标志位（只读）（Overrun flag） 1 = DATA [11：0]数据被覆盖 0 = DATA [11：0]数据最近一次转换结果 新的转换结果装载至寄存器之前，若 DATA[15：0]的数据没有被读取，OVERRUN 将置‘1’。读 ADDATA 寄存器后，该位由硬件清除。
21:16	CHANNELSEL	0	RO	该 5 位显示当前数据所对应的通道（Channel selection） n = 通道 n 的转换数据
15:0	DATA	0	RO	12 位 A/D 转换结果（Transfer data） 根据设置左对齐或者右对齐。

15.5.7 数据寄存器 (ADKEY_DADAT)

Address offset: 0x1C

Width	Name	Reset	Property	Description
31:12	Reserved	0	RO	
11:0	dacdat	0	RW	DAC 数据

15.5.8 数据寄存器 (ADKEY_DATCNT)

Address offset: 0x20

Width	Name	Reset	Property	Description
31:10	Reserved	0	RO	
9:0	dmadatcnt	0	RW	已经 dma 完成的数据长度

15.5.9 寄存器地址映射

Register name	offset	Reset value
ADKEY_CFG	0x00	0
ADKEY_CR	0x04	0
ADKEY_CHS	0x08	0
REVERSE	0x0C	
ADKEY_STA	0x10	0
ADKEY_DMAADR	0x14	0
ADKEY_DATA	0x18	0
ADKEY_DADAT	0x1C	0
ADKEY_DATCNT	0x20	0

15.6 操作流程及注意事项

15.6.1 注意事项：

在使用 ADC DMA 功能时候，需要配置 APB0 时钟与 AHB0 时钟 1：1 关系。

15.6.2 单次采样：

1. 配置寄存器 ADKEY_CFG 使能 ADC
2. 配置寄存器 ADKEY_CFG 选择对应工作的频率
3. 配置寄存器 ADKEY_STA 选择示例通道
4. 配置寄存器 ADKEY_CR 选择 A/D 转换模式为单次采样
5. 配置寄存器 ADKEY_CR 选择数据对齐格式
6. 配置寄存器 ADKEY_CR 使 A/D 转换开始
7. 等待转换完成：读取寄存器 ADKEY_STA 第 0 位的值进行判断
8. 清除 A/D 转换开始标志位，停止工作：配置寄存器 ADKEY_CR 使 A/D 转换进入空闲状态。

15.6.3 周期采样：

1. 配置寄存器 ADKEY_CFG 使能 ADC
2. 配置寄存器 ADKEY_CFG 选择对应工作的频率
3. 配置寄存器 ADKEY_CFG 选择相应的两次连续采样之间的间隔时间周期

4. 配置寄存器 ADKEY_CHS 选择扫描第一条通道
5. 配置寄存器 ADKEY_CHS1 选择扫描下一条通道
6. 配置寄存器 ADKEY_CR 选择扫描通道顺序
7. 配置寄存器 ADKEY_CR 选择 A/D 转换模式为单周期扫描
8. 配置寄存器 ADKEY_CR 选择数据对齐格式
9. 通过配置寄存器 ADKEY_DMAADR 配置 DMA 地址
10. 寄存器 ADKEY_CR 使能 DMA
11. 配置寄存器 ADKEY_CR 使 A/D 转换开始
12. 等待转换完成：读取寄存器 ADKEY_STA 第 0 位的值进行判断
13. 清除 A/D 转换开始标志位，停止工作：配置寄存器 ADKEY_CR 使 A/D 转换进入空闲状态。

15.6.4 扫描模式：

1. 配置寄存器 ADKEY_CFG 使能 ADC
2. 配置寄存器 ADKEY_CFG 选择对应工作的频率
3. 配置寄存器 ADKEY_CFG 选择相应的两次连续采样之间的间隔时间周期
4. 配置寄存器 ADKEY_CHS 选择扫描通道
5. 配置寄存器 ADKEY_CR 选择扫描通道顺序
6. 配置寄存器 ADKEY_CR 选择 A/D 转换模式为连续扫描
7. 配置寄存器 ADKEY_CR 选择数据对齐格式
8. 通过配置寄存器 ADKEY_DMAADR 配置 DMA 地址
9. 寄存器 ADKEY_CR 使能 DMA
10. 配置寄存器 ADKEY_CR 使 A/D 转换开始

15.6.5 外部 trigger 启动采样

1. 配置寄存器 ADKEY_CFG 使能 ADC
2. 配置寄存器 ADKEY_CFG 选择对应工作的频率
3. 配置寄存器 ADKEY_STA 选择示例通道
4. 配置寄存器 ADKEY_CR 选择 A/D 转换模式为外部触发
5. 配置寄存器 ADKEY_CR 选择数据对齐格式
6. 配置寄存器 ADKEY_CR 使能外部触发
7. 配置寄存器 ADKEY_CR 选择外部触发延时采样周期

16 定时器

TS32F020 一共包含 6 个普通 timer 和一个 watchdog。其中 timer0/1/2/3/5 为 16 位 timer，timer4 为 32 位 timer。

16.1 16 位定时器 0/1/2/3/5

16.1.1 简介

定时器 timer0/1/2/3 由一个 16 位的计数器组成。支持定时功能，可选择不同的计数源（系统时钟、内部低速 RC 时钟、外部时钟、GPIO 等），同时支持捕获和 PWM 输出功能,另外 timer5 支持红外发射功能。

16.1.2 主要特性

- 16 位递增计数器
- 支持选择 GPIO 作为计数时钟源
- 支持不同计数时钟源
- 捕获功能
- PWM 输出

16.1.3 功能描述

16.1.3.1 16 位定时器

时基单元包括：

- 计数器寄存器（TIMx_CNT）
- 周期寄存器（TIMx_PR）
- 分频寄存器（TIMx_CON[10:8]）

在计数模式下，计数器从 0 计数到周期值（TIMx_PR），然后重新从 0 开始计数，并且产生一个计数器溢出中断。

分频寄存器可以将计数器的时钟频率按 2 的 n 次方分频，从而提高计数的最大周期。

16.1.3.2 计数源选择

计数器时钟可由下列时钟源提供：

- 内部系统时钟
- 内部 32K RCOSC
- 外部高速晶振
- 外部 inc pin

16.1.3.3 支持 capture 工作模式

输入捕获模式，支持外部 Pin 触发或者内部比较器触发，捕获当前计数值并保存到内部寄存器（TIMx_PWM）。结构如下图：

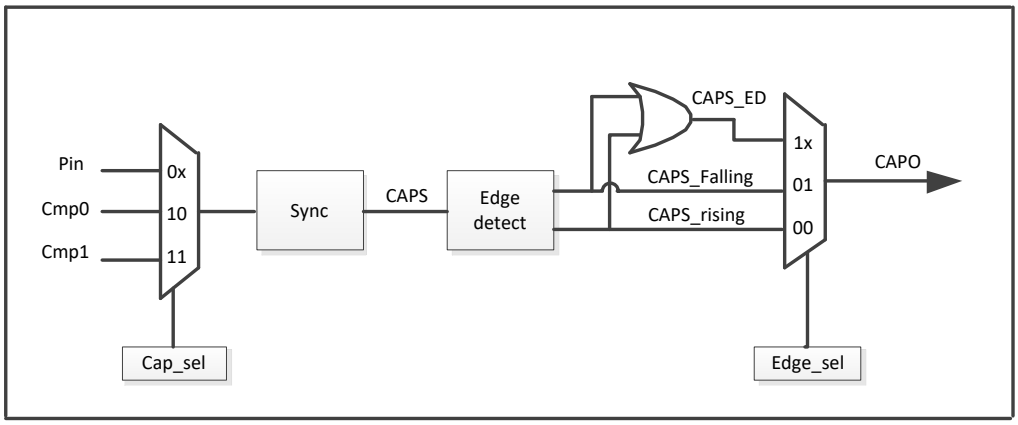


图 16-1 输入捕获通道

上图中 pin 对应 GPIO 输入：

timer0---PA2

timer1—PB0

timer2---PC4/PC8

timer3---PC8/PB15

CMP0 和 CMP1 分别是比较器 0/1 的比较输出信号。首先通过一个选择器选择输入触发通路后，经过同步器同步得到 CAPS，然后边沿检测器产生边沿触发信号，最后通过一个选择器选出最终捕获信号 CAPO。当捕获信号有效时，自动把当前计数器的值存入 TIMx_PWM 寄存器，并且产生捕获中断信号。

16.1.3.4 支持独立 PWM 输出工作模式

PWM 工作模式可以产生一个又 TIMx_PR 寄存器确定周期，TIMx_PWM 寄存器确定占空比的信号。

提前配置好对应 PWM 输出 GPIO 对应的复用功能选择，在 TIMx_PR 和 TIMx_PWM 寄存器中配置所需的 PWM 周期和占空比，然后配置 TIMx_CON 寄存器上的 mode_sel 启动 PWM 输出。这样，对应 IO 上就会一直产生期望的 PWM 波形信号，直到软件配置关闭 PWM 输出。

16.1.3.5 触发 ADC 采样

Timer 还有一个增强型的功能，可以在每次 timer 计数满的时候，触发一次 ADC 采样。具体参考 ADC 章节。

16.1.3.6 Timer5 红外发射功能

Timer5 通过配合 timer0 产生的高频载波，可以产生红外发生信号。具体配置如下图：

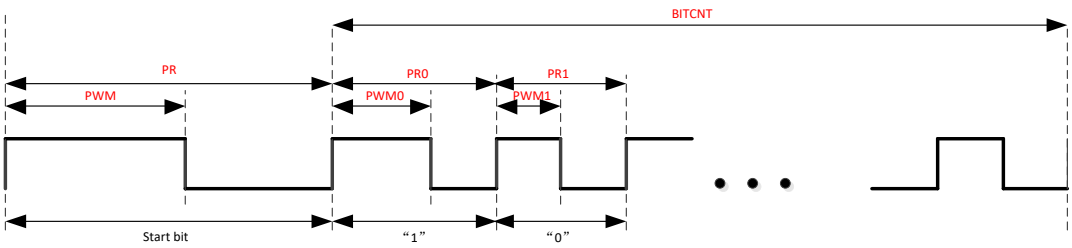


图 16-2 TIM5 产生 PWM 波形

下图是两个 timer 配合产生的红外发射波形:

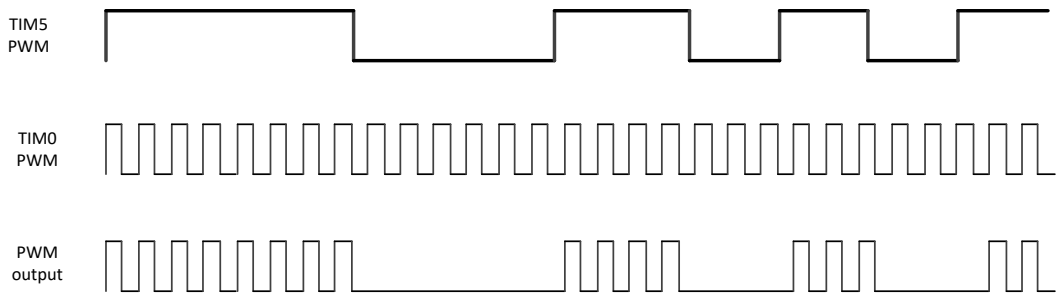


图 16-3 输出到 IO 的红外发射波形

16.1.4 寄存器描述

Register 16-1 控制寄存器 (TMRx_CON)

Address offset: 0x00

Name	Width	Reset	Property	Description
cap_sel	17:16	0	RW	捕获选择: 0x：GPIO 10：compare0 output 11：compare1 output
Timer pending	15	0	RW	定时器标志：（write 1 清除标志） 0：定时器未满标志 1：定时满标志
Capture pending	14	0	RW	捕获标志：（write 1 清除标志） 0：没有捕获事件 1：捕获事件发生
Tmr_ie	13	0	RW	定时器模式中断使能: 0：不使能 1：使能
Cap_ie	12	0	RW	捕获模式中断使能: 0：不使能 1：使能
Reserved	11	-	-	Reserved
psc	10:8	0	RW	Timer 预分频设置： 3’b000： 0 分频 3’b001： 2 分频 3’b010： 4 分频 3’b011： 8 分频 3’b100： 16 分频 3’b101： 32 分频 3’b110： 64 分频 3’b111： 128 分频
Edge_sel	7:6	0	RW	捕获模式边缘选择位: 2’b00：捕获上升沿 2’b01：捕获下降沿 2’b1x：捕获上升沿和下降沿
Mode_sel	5:4	0	RW	定时器模式选择位:

				2’b00： 定时器禁用 2’b01： 定时器计算模式 2’b10： 定时器 PWM 模式 2’b11： 定时器捕获模式
Reserved	3	-	-	Reserved
Inc_src_sel	2:0	0	RW	定时器计算源选择位: 3’b000 : timer inc pin 上升沿 3’b001 : timer inc pin 下降沿 3’b010： 内部高速 RC 除 2 时钟的上升沿和下降沿 3’b011： 内部 32KHz RCOSC 除 2 时钟的上升沿和下降沿 3’b100： 外部晶振除 2 时钟的上升沿和下降沿 others： 系统时钟上升沿

Register 16-2 计数寄存器 (TMRx_CNT)

Address offset: 0x04

Name	Width	Reset	Property	Description
CNT	15:0	16’h0	RW	计数寄存器

Register 16-3 周期寄存器 (TMRx_PR)

Address offset: 0x08

Name	Width	Reset	Property	Description
PR	15:0	16’hffff	RW	计数周期设置寄存器，当使用红外功能能，改值为红外发射第一个 PWM 周期的周期值

Register 16-4PWM 寄存器 (TMRx_PWM)

Address offset: 0x0c

Name	Width	Reset	Property	Description
PWM	15: 0	0	RW	PWM 设置寄存器，当使用红外功能能，改值为红外发射第一个 PWM 周期的占空比

Register 16-5 周期寄存器 (TMRx_PR0)（只有 timer5 有）

Address offset: 0x10

Name	Width	Reset	Property	Description
PR0	15:0	16’h0	RW	红外发射值为“1”的 PWM 周期，具体看图 16-3

Register 16-6PWM 寄存器 (TMRx_PWM0) （只有 timer5 有）

Address offset: 0x14

Name	Width	Reset	Property	Description
PWM0	15: 0	0	RW	红外发射值为“1”的 PWM 相位，具体看图 16-3

Register 16-7 周期寄存器 (TMRx_PR1) （只有 timer5 有）

Address offset: 0x18

Name	Width	Reset	Property	Description
PR1	15:0	16’h0	RW	红外发射值为“0”的 PWM 周期，具体看图 16-3

Register 16-8 PWM 寄存器 (TMRx_PWM1) （只有 timer5 有）

Address offset: 0x1c

Name	Width	Reset	Property	Description
PWM1	15: 0	0	RW	红外发射值为“0”的 PWM 相位，具体看图 16-3

Register 16-9 PWM 寄存器 (TMRx_BITCNT) （只有 timer5 有）

Address offset: 0x20

Name	Width	Reset	Property	Description
BITCNT	7: 0	0	RW	红外发射数据位宽，n 表示 n bit

Register 16-10 PWM 寄存器 (TMRx_TXDAT) （只有 timer5 有）

Address offset: 0x24

Name	Width	Reset	Property	Description
TXDAT	15: 0	0	RW	红外发射 data buffer（高位先发送），内部有一个 16bit 的 buffer，读 buffer 满标志，不满，则可以往这个寄存器写值

Register 16-11 PWM 寄存器 (TMRx_IRSTA) （只有 timer5 有）

Address offset: 0x28

Name	Width	Reset	Property	Description
reserve	15: 11	0	RO	
wdat_req	10	0	RO	写数据有效标志，软件读到该位为“1”，表示可以往 txda 写数据，写 txdat 该位自动清 0
tx_done	9	0	RW	发送完成标志，写 1 清
tx_buf_ferr	8	0	RW	buffer 满，但是继续写数据，导致数据覆盖。写 1 清
tx_buf_eerr	7	0	RW	buffer 空，但是没有及时填数据，导致数据出错。写 1 清
tx_buf_full	6	0	RO	buffer 满
tx_buf_empty	5	1	RO	buffer 空
rdatie	4	0	RW	读数据中断使能，当 buffer 空时产生中断
txie	3	0	RW	发送完成中断使能
xoren	2	0	RW	异或使能，配置该位，输出 PWM 信号与当前传输

				的值做一个异或，主要用于 RC5 红外发射协议
polsel	1	0	RW	PWM 极性选择
iren	0	0	RW	IR 使能，’1’有效

16.1.4.1 寄存器地址 mapping

Register name	offset	Reset value
TMRx_CON	0x00	0
TMRx_CNT	0x04	0
TMRx_PR	0x08	0xffff
TMRx_PWM	0x0C	0
TMRx_PR0	0x10	0
TMRx_PWM0	0x14	0
TMRx_PR1	0x18	0
TMRx_PWM1	0x1C	0
TMRx_BITCNT	0x20	0
TMRx_TXDAT	0x24	0
TMRx_IRSTA	0x28	0

16.1.5 操作步骤

16.1.5.1 定时器计数器模式操作流程：

1. 设置计数器的初始计数值：如 TMRx_CNT = 0x0000； （计数器从 0 开始计数）
2. 设置计数周期值：如 TMRx_PR = 0xffff； （计数器一个周期设置为 0xffff）
3. 设置 PWM 值：如 TMRx_PWM = 0x0； （PWM 设置为 0x0）
4. 选择计数器的时钟源：如 TMRx_CON【2:0】= 3’b101; (系统时钟上升沿)

   注意：如果有用到 timer inc pin 作为计数时钟源，请设置对应的 io map。
5. 选择计数器预分频设置：如 TMRx_CON【10:8】= 3’b000; (不预分频)
6. 选择计数器中断使能设置：如 TMRx_CON【13】= 1’b1; (打开 timer 内部计数器中断使能)
7. 选择计数器工作模式，计数器 kick start，开始工作：如 TMRx_CON【5:4】= 2’b01; (timer counter mode)

16.1.5.2 定时器 pwm 模式操作流程：

8. 设置 timer pwm 输出 pin 的 io map，不需要设置 io 方向，自动设置；（见文档后面的附录表）
9. 设置计数器的初始计数值：如 TMRx_CNT = 0x0000； （计数器从 0 开始计数）
10. 设置计数周期值：如 TMRx_PR = 0xffff； （计数器一个周期设置为 0xffff）
11. 设置 PWM 值：如 TMRx_PWM = 0xff； （PWM 设置为 0xff）
12. 选择计数器的时钟源：如 TMRx_CON【2:0】= 3’b101; (系统时钟上升沿)

    注意：如果有用到 timer inc pin 作为计数时钟源，请设置对应的 io map。
13. 选择计数器预分频设置：如 TMRx_CON【10:8】= 3’b000; (不预分频)

14. 选择计数器中断使能设置：如 TMRx_CON【13】= 1’b1; (打开 timer 内部计数器中断使能)
15. 选择 PWM 工作模式，计数器 kick start，开始工作：如 TMRx_CON【5:4】= 2’b10; (timer PWM mode)

16.1.5.3 定时器捕获模式操作流程：

1. 设置 timer cap 输入 pin 的 io map，及 io 方向设置为输入；（见文档后面的附录表）
2. 设置计数器的初始计数值：如 TMRx_CNT = 0x0000； （计数器从 0 开始计数）
3. 设置计数周期值：如 TMRx_PR = 0x00ff； （PWM 高电平设置为 0x00ff）
4. 设置 PWM 值：如 TMRx_PWM = 0x0； （clear pwm 到 0，capture 源的计数值会存入 PWM 寄存器中）
5. 选择计数器的时钟源：如 TMRx_CON【2:0】= 3’b100; (系统时钟上升沿)

   注意：如果有用到 timer inc pin 作为计数时钟源，请设置对应的 io map。
6. 选择 capture pin 的边沿：如 TMRx_CON【7:6】= 2’b00; (上升沿触发)
7. 选择计数器预分频设置：如 TMRx_CON【10:8】= 3’b000; (不预分频)
8. 选择 capture 中断使能设置：如 TMRx_CON【12】= 1’b1; (打开 captuer 中断使能)
9. 选择 capture 工作模式，计数器 kick start，开始工作：如 TMRx_CON【5:4】= 2’b11; (timer capture mode)
10. 等到 capture pending 为 1，读取 TMRx_PWM 的值就是在固定时间内抓到 capture pin 上事件的计数值。

16.1.5.4 定时器 5 红外发射模式操作流程：

1. 配置 Timer0 输出高频载波 PWM。
2. 设置 timer pwm 输出 pin 的 io map，不需要设置 io 方向，自动设置；（见文档后面的附录表）
3. 选择计数器的时钟源：如 TMRx_CON【2:0】= 3’b100; (系统时钟上升沿)

   注意：如果有用到 timer inc pin 作为计数时钟源，请设置对应的 io map。
4. 选择计数器预分频设置：如 TMRx_CON【10:8】= 3’b000; (不预分频)
5. 选择计数器中断使能设置：如 TMRx_CON【13】= 1’b1; (打开 timer 内部计数器中断使能)
6. 设置计数器的初始计数值：如 TMRx_CNT = 0x0000； （计数器从 0 开始计数）
7. 设置 PWM 第一个周期高电平计数值：如 TMRx_PWM = 0x00ff； （PWM 高电平设置为 0x00ff）（TMRx_PWM 和 TMRx_PR 不能配置为 0）
8. 设置 IR 发送第一个周期周期值：如 TMRx_PR = 0xffff； （PWM 周期设置为 0xffff）
9. 设置 IR 发送“1”周期周期值：如 TMRx_PR0 = 0xffff； （PWM 周期设置为 0xffff）
10. 设置 IR 发送“1”高电平计数值：如 TMRx_PWM0 = 0x00ff； （PWM 高电平设置为 0x00ff）
11. 设置 IR 发送“0”周期周期值：如 TMRx_PR1 = 0xffff； （PWM 周期设置为 0xffff）

12. 设置 IR 发送“0”高电平计数值：如 TMRx_PWM1 = 0x00ff； （PWM 高电平设置为 0x00ff）
13. 配置发送的数据位宽：TMRx_BITCNT = 0x55；（发送有效数据位 0x55）
14. 往 buffer 里面写要发送的数据，高位在前：TMRx_TXDAT = 0x55aa，TMRx_TXDAT = 0xbbcc； （第一次可以连续写两次，因为内部有 16bit 的缓存）
15. 根据不同需要配置中断和使能位：TMRx_IRSTA
16. 配置 IR 使能：TMRx_IRSTA |= BIT(0)；
17. 选择 PWM 工作模式，计数器 kick start，开始工作：如 TMRx_CON【5:4】= 2’b10; (timer PWM mode)
18. 等待中断，往 TXDAT 寄存器里面写剩下的值。

16.2 32 位定时器 4

16.2.1 简介

定时器 timer4 由一个 32 位的计数器组成。支持定时功能，可选择不同的计数源（系统时钟、内部低速 RC 时钟、外部时钟、GPIO 等），同时支持捕获和 PWM 输出功能。

16.2.2 主要特性

- 32 位递增计数器
- 支持选择 GPIO 作为计数时钟源
- 支持不同计数时钟源
- 捕获功能
- PWM 输出

16.2.2.1 32 位定时器

时基单元包括：

- 计数器寄存器（TIMx_CNT）
- 周期寄存器（TIMx_PR）
- 分频寄存器（TIMx_CON[10:8]）

在计数模式下，计数器从 0 计数到周期值（TIMx_PR），然后重新从 0 开始计数，并且产生一个计数器溢出中断。

分频寄存器可以将计数器的时钟频率按 2 的 n 次方分频，从而提高计数的最大周期。

16.2.2.2 计数源选择

计数器时钟可由下列时钟源提供：

- 内部系统时钟
- 内部 32K RCOSC
- 外部高速晶
- 外部 inc pin

16.2.2.3 支持 capture 工作模式

输入捕获模式，支持外部 Pin 触发或者内部比较器触发，捕获当前计数值并保存到内部寄存器（TIMx_PWM）。结构如下图：

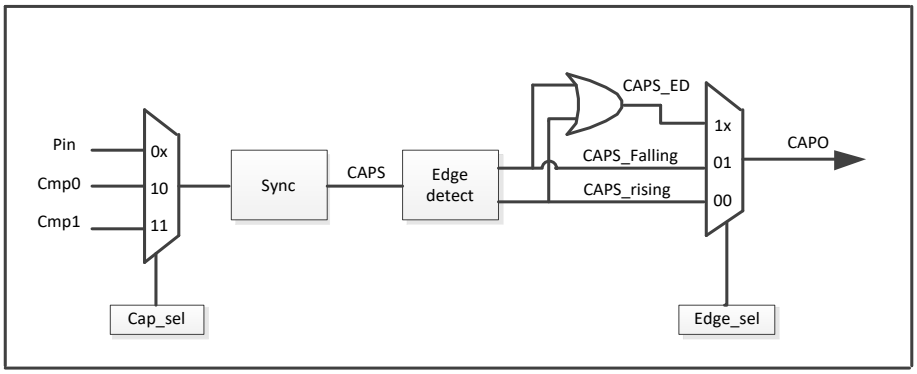


图 16-4 输入捕获通道

上图中 pin 对应 GPIO 输入：

timer4---PA8/PB8

cmp0 和 cmp1 分别是比较器 0/1 的比较输出信号。首先通过一个选择器选择输入触发通路后，经过同步器同步得到 CAPS，然后边沿检测器产生边沿触发信号，最后通过一个选择器选出最终捕获信号 CAPO。当捕获信号有效时，自动把当前计数器的值存入 TIMx_PWM 寄存器，并且产生捕获中断信号。

16.2.2.4 支持独立 PWM 输出工作模式

PWM 工作模式可以产生一个又 TIMx_PR 寄存器确定周期，TIMx_PWM 寄存器确定占空比的信号。

提前配置好对应 PWM 输出 GPIO 对应的复用功能选择，在 TIMx_PR 和 TIMx_PWM 寄存器中配置所需的 PWM 周期和占空比，然后配置 TIMx_CON 寄存器上的 mode_sel 启动 PWM 输出。这样，对应 IO 上就会一直产生期望的 PWM 波形信号，直到软件配置关闭 PWM 输出。

16.2.2.5 触发 ADC 采样

Timer 还有一个增强型的功能，可以在每次 timer 计数满的时候，触发一次 ADC 采样。具体参考 ADC 章节。

16.2.3 寄存器描述

Register 16-5 控制寄存器 (TMRx_CON)

Address offset: 0x00

Name	Width	Reset	Property	Description
cap_sel	17:16	0	RW	捕获选择: 0x：GPIO 10：compare0 output 11：compare1 output
Timer pending	15	0	RW	定时器标志：（write 1 清除标志） 1：定时满状态 0：定时器未满状态

Capture pending	14	0	RW	捕获事件标志：（write 1 清除标志） 1：捕获事件发生 0：捕获事件未发生
Tmr_ie	13	0	RW	定时器模式中断使能: 0：不使能 1：使能
Cap_ie	12	0	RW	捕获模式中断使能: 0：不使能 1：使能
Reserved	11	-	-	保留
psc	10:8	0	RW	Timer 预分频设置： 3’b000： 0 分频 3’b001： 2 分频 3’b010： 4 分频 3’b011： 8 分频 3’b100： 16 分频 3’b101： 32 分频 3’b110： 64 分频 3’b111： 128 分频
Edge_sel	7:6	0	RW	捕获模式边缘选择位: 2’b00 :捕获上升沿 2’b01 :捕获下降沿 2’b1x :捕获上升沿和下降沿
Mode_sel	5:4	0	RW	定时器模式选择位: 2’b00：定时器禁用 2’b01：定时器计数模式 2’b10：定时器 PWM 模式 2’b11：定时器捕获模式
Reserved	3	-	-	保留
Inc_src_sel	2:0	0	RW	定时器计数源选择位: 3’b000 : timer inc pin 上升沿 3’b001 : timer inc pin 下降沿 3’b010 ： 内部高速 RC 除 2 时钟的上升沿和下降沿 3’b011 :内部 32KHz RCOSC 除 2 时钟的上升沿和下降沿 3’b100：外部晶振除 2 时钟的上升沿和下降沿 others：系统时钟上升沿

Register 166-6 计数寄存器 (TMRx_CNT)

Address offset: 0x04

Name	Width	Reset	Property	Description
CNT	31:0	32’h0	RW	计数寄存器

Register 166-7 周期寄存器 (TMRx_PR)

Address offset: 0x08

Name	Width	Reset	Property	Description
PR	31:0	32'hffffffff	RW	计数周期设置寄存器

Register 166-8PWM 寄存器 (TMRx_PWM)

Address offset: 0x0c

Name	Width	Reset	Property	Description
PWM	31: 0	0	RW	PWM 设置寄存器

16.2.4 操作步骤

16.2.4.1 定时器计数器模式操作流程：

1. 设置计数器的初始计数值：如 TMRx_CNT = 0x0000； （计数器从 0 开始计数）
2. 设置计数周期值：如 TMRx_PR = 0xffff； （计数器一个周期设置为 0xffff）
3. 设置 PWM 值：如 TMRx_PWM = 0x0； （PWM 设置为 0x0）
4. 选择计数器的时钟源：如 TMRx_CON【2:0】= 3'b101; (系统时钟上升沿)

   注意：如果有用到 timer inc pin 作为计数时钟源，请设置对应的 io map。
5. 选择计数器预分频设置：如 TMRx_CON【10:8】= 3'b000; (不预分频)
6. 选择计数器中断使能设置：如 TMRx_CON【13】= 1'b1; (打开 timer 内部计数器中断使能)
7. 选择计数器工作模式，计数器 kick start，开始工作：如 TMRx_CON【5:4】= 2'b01; (timer counter mode)

16.2.4.2 定时器 pwm 模式操作流程：

1. 设置 timer pwm 输出 pin 的 io map，不需要设置 io 方向，自动设置；（见文档后面的附录表）
2. 设置计数器的初始计数值：如 TMRx_CNT = 0x0000； （计数器从 0 开始计数）
3. 设置 PWM 高电平计数值：如 TMRx_PR = 0x00ff； （PWM 高电平设置为 0x00ff）
4. 设置 PWM 周期值：如 TMRx_PWM = 0xffff； （PWM 周期设置为 0xffff）
5. 选择计数器的时钟源：如 TMRx_CON【2:0】= 3'b101; (系统时钟上升沿)

   注意：如果有用到 timer inc pin 作为计数时钟源，请设置对应的 io map。
6. 选择计数器预分频设置：如 TMRx_CON【10:8】= 3'b000; (不预分频)
7. 选择计数器中断使能设置：如 TMRx_CON【13】= 1'b1; (打开 timer 内部计数器中断使能)
8. 选择 PWM 工作模式，计数器 kick start，开始工作：如 TMRx_CON【5:4】= 2'b10; (timer PWM mode)

16.2.4.3 定时器捕获模式操作流程：

1. 设置 timer cap 输入 pin 的 io map，及 io 方向设置为输入；（见文档后面的附录表）
2. 设置计数器的初始计数值：如 TMRx_CNT = 0x0000； （计数器从 0 开始计数）

3. 设置计数周期值：如 TMRx_PR = 0x00ff； （PWM 高电平设置为 0x00ff）
4. 设置 PWM 值：如 TMRx_PWM = 0x0； （clear pwm 到 0，capture 源的计数值会存入 PWM 寄存器中）
5. 选择计数器的时钟源：如 TMRx_CON【2:0】= 3'b101; (系统时钟上升沿)

   注意：如果有用到 timer inc pin 作为计数时钟源，请设置对应的 io map。
6. 选择 capture pin 的边沿：如 TMRx_CON【7:6】= 2'b00; (上升沿触发)
7. 选择计数器预分频设置：如 TMRx_CON【10:8】= 3'b000; (不预分频)
8. 选择 capture 中断使能设置：如 TMRx_CON【12】= 1'b1; (打开 captuer 中断使能)
9. 选择 capture 工作模式，计数器 kick start，开始工作：如 TMRx_CON【5:4】= 2'b11; (timer capture mode)
10. 等到 capture pending 为 1，读取 TMRx_PWM 的值就是在固定时间内抓到 capture pin 上事件的计数值。

16.3 看门狗

16.3.1 简介

Watchdog 工作时钟为 32K。默认定时 2S，可以通过修改分频系数改变看门狗超时时间。通过配置寄存器，可以选择当计时溢出时，复位系统或者产生中断。

16.3.2 寄存器描述

Register 16-9WDT_CON

Address offset: 0x00

Name	Bit	Reset	ACCESS	Description
reserved	31:10	23'b0	-	-
wake_en	9	1'b0	RO	唤醒使能
reserved	8:7	2'b0	-	-
wdt_pend	6	1'b0	RO	watchdog 计数满标志 WDG_KEY 写 0xaaaa，清除 wdt_pending；
int_enable	5	1'b0	RO	中断使能： 1=计数满时产生中断； 0=计数满时复位系统
wdte	4	1'b1	RW	写 WDT_KEY=0xCCCC，置位。 写 WDT_KEY=0xDDDD，复位。
wdt_psr	3:0	4'h8	RW	分频系数，每次配置该位域之前必须先写 WDG_KEY=0x5555 4'h0 ： 不分频 4'h1 ：2 4'h2 ：4 4'h3 ：8 4'h4 ：16 4'h5 ：32

				4’h6 ： 64 4’h7 ： 128 4’h8 ： 256 4’h9 ： 512 4’ha ： 1024 4’hb ： 2048 4’hc ： 4096 4’hd ： 8192 4’he ： 16384 4’hf ： 32768 看门狗复位时间=1/32K*256*分频系数

Register 16-10WDG_KEY

Address offset: 0x04

Name	Bit	Reset	ACCESS	Description
reserved	31:16	16'b0	-	-
KEY	15:0	16'b0	WO	KEY[15：0]：键值（只写寄存器，读出值为 0x0000）（Key value） 软件必须以一定的间隔写入 0xAAAA，否则，当计数器为 0 时，看门狗会产生复位。当 pending 为 1 的时候，写 0xAAAA 会清除 pending。 写入 0x5555 表示允许访问 WDG_PSR。 写入 0xCCCC，启动看门狗工作。 写入 0xDDDD，关闭看门狗。 写 0xaaaa，清除 wdt_pending； 写入 0x55AA，开启中断使能； 写入 0xAA55，关闭中断使能； 写入 0x5A5A，开启 wake up 使能； 写入 0xA5A5，关闭 wake up 使能；

17 系统寄存器

17.1 体统寄存器基地址表

Name	Base Address	
System Register	0x40020000	

Table 1 register list

Offset	Register Name	Description
0x0000	SYS_KEY	按键控制寄存器
0x0004	SYS_CON0	系统控制寄存器 0
0x0008	SYS_CON1	系统控制寄存器 1
0x000c	SYS_CON2	系统控制寄存器 2
0x0010	SYS_CON3	系统控制寄存器 3
0x0014	SYS_CON4	系统控制寄存器 4
0x0018	SYS_CON5	系统控制寄存器 5
0x001c	SYS_CON6	系统控制寄存器 6
0x0020	SYS_CON7	系统控制寄存器 7
0x0024	CLKCON0	时钟控制寄存器 0
0x0028	CLKCON1	时钟控制寄存器 1
0x002c	CLKCON2	时钟控制寄存器 2
0x0030	CLKCON3	时钟控制寄存器 3
0x0034	CLKCON4	时钟控制寄存器 4
0x0038	CLKCON5	时钟控制寄存器 5
0x003c	CLKCON6	时钟控制寄存器 6
0x0040	CLKCON7	时钟控制寄存器 7
0x0044	HOSC_MNT	XOSC fail monitor register（晶振失败监控寄存器）
0x0048	SYS_ERR	系统出错记录寄存器
0x004c	WKUP_CON	唤醒控制寄存器
0x0050	LP_CON	低功率控制寄存器
0x0060	MODE	特殊模式记录寄存器
0x0064	PMUCON0	PMU 控制寄存器
0x0068	RPCON	系统暂挂记录寄存器

17.2 存器描述

Register 17-1 SYS_KEY

Offset = 0x0000

Name	Bit(s)	Reset	R/W	Description
sys_key	31:0	1'b1	R/W	写入 0x3fac87e4 以使能所有系统寄存器写操作,写入除了这个数外都会将这个 bit 置 0。读取返回 sys_key 状态 0:锁定所有系统寄存器写操作 1:解锁所有系统寄存器写操作

Register 17-2SYS_CON0

Offset = 0x0004

Name	Bit(s)	Reset	R/W	Description
Reserved	31:16	-	-	读时全为 0
gpio_soft_rst_	15	1'b1	R/W	0: 复位 GPIOA/GPIOB/GPIOC 配置寄存器 1:完成复位
Reserved	14	-	-	读时为 0
adkey_soft_rst_	13	1'b1	R/W	0: 复位 KeyADC 配置寄存器. 1: 完成复位.
tk_soft_rst_	12	1'b1	R/W	0: 复位 TouchKey&Led driver 配置寄存器. 1: 完成复位.
Reserved	11	-	-	读时为 0
dbs_soft_rst_	10	1'b1	R/W	0: 复位 GPIO debounce relative 配置寄存器. 1: 完成复位.
crc_soft_rst_	9	1'b1	R/W	0: 复位 CRC relative 配置寄存器. 1: 完成复位.
wdt_soft_rst_	8	1'b1	R/W	0: 复位 WDT relative 配置寄存器. 1: 完成复位.
Reserved	7:6	-	-	Be read as all zeros
uart1_soft_rst_	5	1'b1	R/W	0: 复位 UART1 relative 配置寄存器. 1: 完成复位.
uart0_soft_rst_	4	1'b1	R/W	0: 复位 UART0 relative 配置寄存器. 1: 完成复位.
spi0_soft_rst_	3	1'b1	R/W	0: 复位 SPI0 relative 配置寄存器. 1: 完成复位.
spi1_soft_rst_	2	1'b1	R/W	0: 复位 SPI1 relative 配置寄存器. 1: 完成复位.
timer_soft_rst_	1	1'b1	R/W	0: 复位 TIMERs relative 配置寄存器. 1: 完成复位.
wdt_sys_soft_rst_	0	1'b1	R/W	0: 复位 WDT system domain relative 配置寄存器. 1: 完成复位.

Register 17-3SYS_CON1

Offset = 0x0008

Name	Bit(s)	Reset	R/W	Description
Reserved	31:11	-	-	读时全为 0
nmi_inv_sel	10	1'b0	R/W	NMI 引脚极性反向选择 0: 高电平引脚触发 NMI 1: 低电平引脚触发 NMI
Reserved	9	-	-	读时为 0
dbg_en	8	1’b1	R/W	DBG enable 0: 不使能 1: 使能
lvdvcc_rst_en	7	1’b0	R/W	VCC 低压检测器快速复位 GPIO 状态寄存器 0: 不使能 1: 使能
clk_test_oe	6	1’b0	R/W	使内部时钟源输出到 PB7
sys_err_resp_en	5	1’b0	R/W	启用系统总线访问出侧内存区域返回错误响应 0:不使能 1:使能
sys_err_int_en	4	1’b0	R/W	启用系统总线访问出侧内存区触发 NMI 中断 0:不使能 1:使能
hosc_loss_nmi_en	3	1’b0	R/W	监控 XOSC 丢失触发 NMI 中断 0:不使能 1:使能
rxev_enable	2	1’b0	R/W	使能将 PA11 作为接收方事件发送到 CPU 0:不使能 1:使能
nmi_int_enable	1	1’b0	R/W	使能 PA10 作为 CPU 的外部 NMI 输入 0:不使能 1:使能
lockup_enable	0	1’b0	R/W	使能系统锁定触发系统复位 0:不使能 1:使能

Register 17-4SYS_CON2

Offset = 0x000c

Name	Bit(s)	Reset	R/W	Description
pc_deb_en[3:0]	31:28	4'h0	R/W	PortC 去抖动使能寄存器 [3:0] for PC3-PC0
pb_deb_en	27:12	16'h0	R/W	PortB 去抖动使能寄存器 [15:0] for PB15-PB0
pa_deb_en	11:0	12'h0	R/W	PortA 去抖动使能寄存器 [11:0] for PA11-PA0

Register 17-5SYS_CON3

Offset = 0x0010

注：在 SOP20/SOP24/SOP28 封装的时候需要 enable remap PB，PC 时，同时模拟功能要用对应的 PB0,PB2，... ，PB12,PB14

Name	Bit(s)	Reset	R/W	Description
Reserved	31:18	-	-	读时全为 0
pc_remap_en[3]	17	1'b0	R/W	使能 PC9 重新映射 0:不使能 1:使能 PC9 重新映射 to PC7
pc_remap_en[2]	16	1'b0	R/W	使能 PC8 重新映射 0: 不使能 1: 使能 PC8 重新映射 to PC7
pc_remap_en[1]	15	1'b0	R/W	使能 PC5 重新映射 0:不使能 1:使能. PC5 重新映射 to PC6
pc_remap_en[0]	14	1'b0	R/W	使能 PC4 重新映射 0:不使能 1:使能. PC4 重新映射 to PC6
pb_remap_en[7]	13	1'b0	R/W	使能 PB14/PB15 重新映射 0:不使能 1:使能. PB14/PB15 重新映射 to PB7
pb_remap_en[6]	12	1'b0	R/W	使能 PB12/PB13 重新映射 0:不使能 1:使能. PB12/PB13 重新映射 to PB6
pb_remap_en[5]	11	1’b0	R/W	使能 PB10/PB11 重新映射 0:不使能 1:使能. PB10/PB11 重新映射 to PB5
pb_remap_en[4]	10	1’b0	R/W	使能 PB8/PB9 重新映射 0:不使能 1:使能. PB8/PB9 重新映射 to PB4
pb_remap_en[3]	9	1’b0	R/W	使能 PB6/PB7 重新映射 0:不使能 1:使能. PB6/PB7 重新映射 to PB3
pb_remap_en[2]	8	1’b0	R/W	使能 PB4/PB5 重新映射 0:不使能 1:使能. PB4/PB5 重新映射 to PB2
pb_remap_en[1]	7	1’b0	R/W	使能 PB2/PB3 重新映射 0:不使能 1:使能. PB2/PB3 重新映射 to PB1
pb_remap_en[0]	6	1’b0	R/W	使能 PB1 重新映射 0:不使能 1:使能. PB1 重新映射 to PB0
pc_deb_en[9:4]	5:0	6'h0	R/W	PortC 防反跳寄存器 [9:4] for PC9-PC4

Register 17-6 SYS_CON4

Offset = 0x0014

Name	Bit(s)	Reset	R/W	Description
Reserved	31:0	-	-	读时全为 0

Register 17-7 SYS_CON5

Offset = 0x0018

Name	Bit(s)	Reset	R/W	Description
Reserved	31:0	-	-	读时全为 0

Register 17-8 SYS_CON6

Offset = 0x001c

Name	Bit(s)	Reset	R/W	Description
Reserved	31:0	-	-	读时全为 0

Register 17-9 SYS_CON7

Offset = 0x0020

Name	Bit(s)	Reset	R/W	Description
Reserved	31:0	-	-	读时全为 0

Register 17-10 CLKCON0

Offset = 0x0024

Name	Bit(s)	Reset	R/W	Description
Reserved	31:17	-	-	读时全为 0
ringout_sel	16	1'b0	R/W	使 CLK 输出选择 HIRC 26M or RingClk 0:选择 HIRC 26M 1:选择 RingClk
comp_clk_sel	15:14	2'h0	R/W	比较器工作时钟选择 00:sys_clk 01: divider output from HIRC 26M 10: LIRC 32K 11:XOSC
pll_sel	13:12	2'h1	R/W	pll 输出时钟选择 00:选择阶段 4 输出 01:选择阶段 5 输出 10:选择阶段 6 输出 11:选择阶段 7 输出
gpioc_dbs_sel	11:10	2'h0	R/W	GPIOC 断开工作时钟选择 00: XOSC 01:divider output from HIRC 26M 10: sys_clk 11: LIRC 32K
gpiob_dbs_sel	9:8	2'h0	R/W	GPIOB 断开工作时钟选择 00:sys_clk 01: divider output from HIRC 26M 10: LIRC 32K 11:XOSC

gpioa_dbs_sel	7:6	2'h0	R/W	GPIOA 断开工作时钟选择 00:sys_clk 01: divider output from HIRC 26M 10: LIRC 32K 11:XOSC
pll_refclk_sel	5: 4	2'h0	R/W	Pll 参考时钟选择 00:HIRC 26M 01: LIRC 32K 10: HIRC 26M 11:XOSC
clock_to_io_sel	3:2	2'h0	R/W	输出时钟选择 00:pll_clk 01:HIRC 26M or RingClk 10: LIRC 32K 11:XOSC
sysclk_sel	1:0	2'h0	R/W	系统时钟选择 00: LIRC 32K 01: XOSC 10: HIRC 26M 11: pll_clk

Register 17-11 CLKCON1

Offset = 0x0028

Name	Bit(s)	Reset	R/W	Description
Reserved	31:30	-	-	读时全为 0
clk_to_io_div	29:28	0x0	R/W	时钟到 IO 分配器 0: divide by 1 1: divide by 2 2:divide by 3 3: 关闭时钟选择
hirc_clk_div	27:24	0x0	R/W	HIRC 26M clock divider 0:divide by 1 1:divide by 2 ... 0xe divide by 15 0xf 关闭时钟选择
tk_div	23:16	0x0	R/W	tk_hclk clock divider 0:divide by 1 1:divide by 2 ... 0xfe:divide by 255 0xff: 关闭 tk_hclk
apb1clk_div	15:12	0x0	R/W	APB1 clock divider 0:divide by 1 1:divide by 2 ...

				0xe divide by 15 0xf 关闭 apb1_clk
apb0clk_div	11:8	0x0	R/W	APB0 clock divider 0:divide by 1 1:divide by 2 ... 0xe divide by 15 0xf 关闭 apb0_clk
sysclk_div	7:0	0x0	R/W	System clock divider 0:divide by 1 1:divide by 2 ... 0xfe:divide by 255 0xff: close sys_clk

Register 17-12 CLKCON2

Offset = 0x002c

Name	Bit(s)	Reset	R/W	Description
reserved	31	-	-	读时全为 0
hxosc_tsout_en	30	0x0	R/W	使能 HXOSC 测试输出 0:不使能 1:使能
cp_clk_en	29	0x0	R/W	CP mode 使能 cpu clock 0:不使能 1:使能
test_clk_en	28	0x0	R/W	使能 ATE clock input 0:不使能 1:使能
comp_clk_en	27	0x1	R/W	使能 comparator clock 0:不使能 1:使能
Reserved	26	-	-	读时全为 0
crc_clk_en	25	0x1	R/W	使能 CRC clock 0:不使能 1:使能
eflash_mem_clk_en	24	0x1	R/W	使能 eflash erase/program clock 0:不使能 1:使能
Reserved	23:19	-	-	读时全为 0
uart1_clk_en	18	0x1	R/W	使能 uart1 clock 0:不使能 1:使能
uart0_clk_en	17	0x1	R/W	使能 uart0 clock 0:不使能 1:使能

spi1_clk_en	16	0x1	R/W	使能 spi1 clock 0:不使能 1:使能
spi0_clk_en	15	0x1	R/W	使能 spi0 clock 0:不使能 1:使能
Reserved	14	-	-	读时全为 0
timer5_clk_en	13	0x1	R/W	使能 timer5 clock 0:不使能 1:使能
timer3_clk_en	12	0x1	R/W	使能 timer3 clock 0:不使能 1:使能
timer2_clk_en	11	0x1	R/W	使能 timer2 clock 0:不使能 1:使能
timer1_clk_en	10	0x1	R/W	使能 timer1 clock 0:不使能 1:使能
timer0_clk_en	9	0x1	R/W	使能 timer0 clock 0:不使能 1:使能
wdt_clk_en	8	0x1	R/W	使能 watchdog clock 0:不使能 1:使能
wdt_sys_clk_en	7	0x1	R/W	使能 watchdog system clock 0:不使能 1:使能
timer4_clk_en	6	0x1	R/W	使能 timer4 clock 0:不使能 1:使能
Reserved	5	-	-	读时全为 0
led_clk_en	4	0x1	R/W	使能 LED driver clock 0:不使能 1:使能
tk_clk_en	3	0x1	R/W	使能 touchkey clock 0:不使能 1:使能
sram0_clk_en	2	0x1	R/W	使能 SRAM clock 0:不使能 1:使能
ahb1_clk_en	1	0x1	R/W	使能 ahb1 clock 0:不使能 1:使能
ahb0_clk_en	0	0x1	R/W	使能 ahb0 clock 0:不使能 1:使能

Register 17-13CLKCON3

Offset = 0x0030

Name	Bit(s)	Reset	R/W	Description
reserved	31:28	-	-	读时全为 0
ring_ringen	27	0	R/W	RingClk 使能 0:不使能 1:使能
ring_div2en	26	0	R/W	RingClk divide 2 使能 0:不使能 1:使能
hxosc_ldoen	25	0	R/W	XOSC internal LDO 使能 0:不使能 1:使能
hrcosc_vtest2	24	0	R/W	VDD 测试使能; 输出到 VTSOUT; 0:不使能 1:使能
hrcosc_vtest1	23	0	R/W	VDDOSC 测试使能; 输出到 VTSOUT; 0:不使能 1:使能
hrcosc_vsel	22	0x0	R/W	LDO 参考选择; 1'b0: 基电压; 1'b1: 电流
hrcosc_sc	21:15	0x40	R/W	Freq control; 0x0 :最低; 0x7f:最高;
hrcosc_ldos	14	0	R/W	RCOSC LDO 电压选择; 1'b0:1.5v; 1'b1:1.6v
hrcosc_en	13	1	R/W	RCOSC 使能 1'b0:不使能; 1'b1:使能
Reserved	12	1	RO	HIRC 26M 就绪状态位 0:未就绪 1:就绪
hxosc_sc	11:7	0x1a	R/W	Setting XOSCI(SC[3:2])/XOSCO(SC[1:0])
hxosc_res	6:5	0x1	R/W	XOSC 双针反馈电阻控制;
Reserved	4	-	-	读时为 0
hxosc_en	3	0x0	R/W	XOSC 使能 0:不使能 1:使能
hxosc_drv	2:0	0x3	R/W	驱动能力控制; 000:XTAL=12M 011: XTAL=24M/26M

Register 17-14CLKCON4

Offset = 0x0034

Name	Bit(s)	Reset	R/W	Description
Reserved	31:0			读时全为 0

Register 17-15CLKCON5

Offset = 0x0038

Name	Bit(s)	Reset	R/W	Description
Reserved	31:0			读时全为 0

Register 17-16CLKCON6

Offset = 0x003c

Name	Bit(s)	Reset	R/W	Description
Reserved	31:0			读时全为 0

Register 17-17CLKCON7

Offset = 0x0040

Name	Bit(s)	Reset	R/W	Description
Reserved	31:0			读时全为 0

Register 17-18HOSC_MNT

Offset = 0x0044

Name	Bit(s)	Reset	R/W	Description
high_limit	31:16	0x1450	R/W	xosc 正常工作计数高极限
hosc_mnt_en	15	0x0	R/W	使能 xosc 工作监控 0:不使能 1:使能
hosc_loss_pending	14	0x0	R/W	Read :外部 XOSC 出错状态标志 Write: 0:清除标志 1:不采取操作
hosc_loss_sw_en	13	0x0	R/W	使能当检测到外部 xosc 故障时，硬件自动将 xosc 时钟从 xosc 切换到 HIRC 0:不自动切换 1:自动切换
low_limit	12:0	0xa7	R/W	xosc 正常工作计数低极限

Register 17-19SYS_ERR

Offset = 0x0048

Name	Bit(s)	Reset	R/W	Description
Reserved	31:1	-	-	读时全为 0
clk_err	1	0	R/W	适中使用出错状态. 1: 时钟使用发生错误时，此位可由硬件设置。清除由软件 0: 适中使用没有错误
sys_err0	0	0	R/W	指示出侧存储器的总线操作的出错标志。如果设置 sys_err_int_en，这个标志将使 NMI 中断发生

clk_err condition			
	lirc disable	hirc disable	xosc disable | xosc ldo disable
sys_clk_sel_lirc	√		
sys_clk_sel_hirc		√	
sys_clk_sel_xosc			√
sys_clk_sel_pll & pll_ref_sel_lirc	√		
sys_clk_sel_pll & pll_ref_sel_hirc		√	
sys_clk_sel_pll & pll_ref_sel_xosc			√

Register 17-20WKUP_CON

Offset = 0x004c

Name	Bit(s)	Reset	R/W	Description
Reserved	31:28	-	-	读时全为 0
wkup_pnd	27:24	-	R	指示检测到的相对 IO 边缘 0:未检出 1:检测出
Reserved	23:20	-	-	读时全为 0
clr_pnd	19:16	-	W	写入 1 以清除相关标志
Reserved	15:12	-	-	读时全为 0
wkup_edge	11:8	0x0	R/W	选择要监视所选外部端口的边缘 0: 上升沿 1: 下降沿
Reserved	7:4	-	-	读时全为 0
wkup_en	3:0	0x0	R/W	使能选择要监视所选外部端口的边缘 0: 不使能 1:使能

Register 17-21LP_CON

Offset = 0x0050

Name	Bit(s)	Reset	R/W	Description
Reserved	31:7	-	-	读时全为 0
hirc_auto_enable	6	0x0	R/W	自动使能 HIRC 26M 当 xosc 监视发生丢失事件 0:不使能 1:使能自动打开 HIRC 26M
rc32k_soft_en	5	0x1	R/W	使能 LIRC 0:不使能 1:使能
rc32k_auto_dis	4	0x0	R/W	使能进入睡眠模式时自动关闭 LIRC 0:不使能 1:使能
hirc_auto_dis	3	0x0	R/W	使能 HIRC 26M 自动关机，停机或休眠模式 0:不使能 1:使能
sram0_auto_dis	2	0x0	R/W	使能自动关闭 sram CE 当进入停机或睡眠模式时 0:不使能 1:使能
stopclk	1	0x0	R/W	停机模式 0:不进入停机模式 1:进入停机模式
sleep	0	0x0	R/W	睡眠模式 0:不进入睡眠模式 1:进入睡眠模式

Register 17-22MBIST_CON

Offset = 0x0054

Name	Bit(s)	Reset	R/W	Description
Reserved	31:21	-	-	读时全为 0
mbist_fail_h	20	0	R/W	Read: 指示 MBIST 失败标志 Write: Write 1 to 清除这个失败标志
mbist_tst_done	19	0	R/W	Read: 指示 MBIST 已完成标志 Write: Write 1 to 清除这个完成标志
mbist_clk_en	18	0x0	R/W	Mbist 时钟使能 0:不使能 1:使能
Reserved	17:12	-	-	读时全为 0
mbist_rf1p_debugz	11	0x0	R/W	使能 rf1p 内存 mbist 调试 0:不使能 1:使能
mbist_rf1p_hold_l	10	0x0	R/W	rf1p mbist 测试保持 0: 保持

				1: 运行
mbist_rf1p_test_h	9	0x0	R/W	使能 rf1p mbist 测试 0:不使能 1:使能
Reserved	8:0	-	-	读时全为 0

Register 17-23MBIST_MISR

Offset = 0x0058

Name	Bit(s)	Reset	R/W	Description
Reserved	31:0	-	-	读时全为 0

Register 17-24CHIPID_DCN

Offset = 0x005c

Name	Bit(s)	Reset	R/W	Description
Reserved	31:24	-	-	读时全为 0
chip_dcn	23:16	0x0	R	设计变更说明版本
chip_id	15:0	0x1001	R	Chip id

Register 17-25MODE

Offset = 0x0060

Name	Bit(s)	Reset	R/W	Description
Reserved	31:2	-	-	读时全为 0
cp_mode	1	-	R	CP 模式标志 0: chip is not cp mode 1:chip is cp mode
Reserved	0	-	-	读时全为 0

Register 17-26PMUCON0

Offset = 0x0064

Name	Bit(s)	Reset	R/W	Description
Reserved	31:2	-	-	读时全为 0
led_ldrv	14	0x0	R/W	LED 低电流驱动 0:不使能 1:使能
led_bias	13	0x0	R/W	LED 基值使能 0:不使能 1:使能
tsen_en	12	0x0	R/W	温度传感器使能 0:不使能 1:使能

pmu_v2ien	11	0x1	R/W	PMU V2I 使能 0:不使能 1:使能
pmu_bglowp	10	0x0	R/W	BG in 低功率模式使能 0:不使能 1:使能低功耗模式
pmu_bgbuf	9	0x0	R/W	VBG Buffer 使能 0:不使能 1:使能
ldo_s	8:5	0x0	R/W	VDD 电压选择 0000=1.50@默认 0001=1.10 0010=1.15 0011=1.20 0100=1.25 0101=1.30 0110=1.35 0111=1.40 1000=1.45 1001=1.05 1010=1.55 1011=1.60 1100=1.65 1101=1.70 1110=1.75 1111=1.80
ldo_pd5ma	4	0x0	R/W	VDD LDO 下拉 5mA 选择使能 0:不使能 1:使能
ldo_pd2p5ma	3	0x1	R/W	VDD LDO 下拉 2.5mA 选择使能 0:不使能 1:使能
lpldo_en	2	0x0	R/W	PMU 低功耗工作模式使能 0:不使能 1:使能
bgtrim_s	1:0	0x0	R/W	VBG 微调控制; 00=1.206(默认 t) 01=1.181 10=1.232 11=1.257 step=25mv

Register 17-27RPCON

Offset = 0x0068

Name	Bit(s)	Reset	R/W	Description

Reserved	31:20	-	-	读时全为 0
uart0_reset_clr	19	0	W	Write 1 清除 uart0 更新复位标志
lockup_reset_clr	18	0	W	Write 1 清除锁定复位标志
soft_reset_clr	17	0	W	Write 1 清除软件复位标志
sleep_sta_clr	16	0	W	Write 1 清除睡眠标志
Reserved	15:4	-	-	读时全为 0
uart0_update_pending	3	0	R	Uart0 更新事件发生标志
lock_reset_pending	2	-	R	当发生锁定复位时，这个位将记录复位标志直到 cpu 清除它 0: 没发生锁定复位 1: 锁定复位已发生
soft_reset_pending	1	-	R/W	写入 1 到此位将使芯片复位。这个复位可以使 Eflash 重新获取 NVR 和主数据 0:不使能软件复位 1: 使能软件复位和记录这个复位标志
sleep_pending	0	-	R	写 LP_CON[0]将记录休眠模式标志，即使端口唤醒系统复位

18 LED

18.1 简介

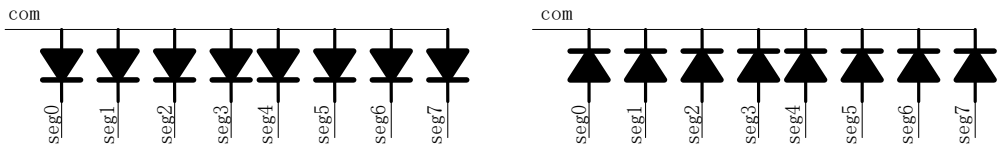
LED(Light Emitting Diode)，发光二极管，是一种能够将电能转化为可见光的固态的半导体器件，它可以直接把电转化为光。LED 的心脏是一个半导体的晶片，晶片的一端附在一个支架上，一端是负极，另一端连接电源的正极，使整个晶片被环氧树脂封装起来。

18.2 主要特性

- 支持最大 8 个 COM 口，12 个 SEG 口
- SEG 口支持与 TK 复用
- 支持按 COM 口扫描和按 SEG 口扫描
- 支持 LED、数码管共阳极和共阴极接法
- 支持调整扫描频率和调节亮度

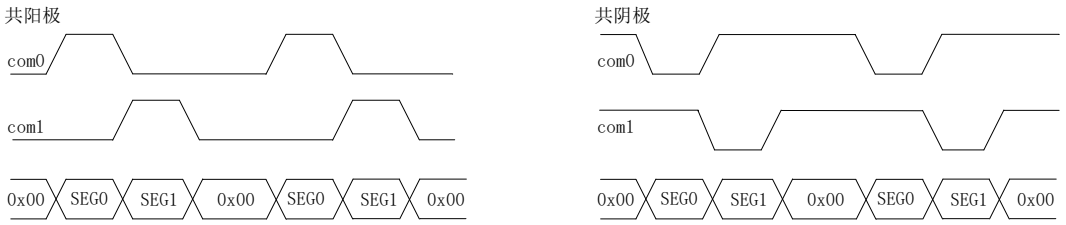
18.3 功能描述

以动态扫描的方式驱动数码管、LED，支持自动定时扫描或者与 TK 复用的扫描方式，区别是前者按照定时器定时启动扫描，后者是等待 TK 模块给出启动信号再启动扫描。通过控制 COM 和 SEG 口的高低电平来控制是否点亮。以下 LED、数码管共阳极是的共阳及共阴极接法：

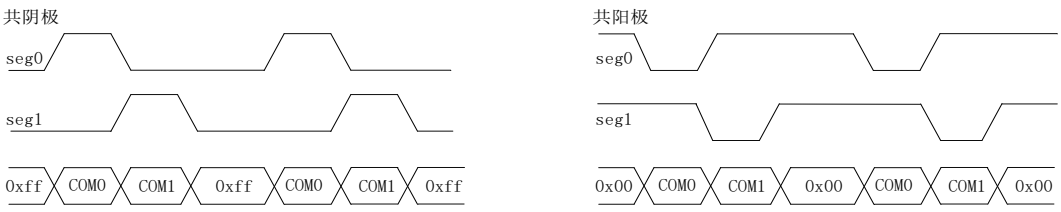


其中，扫描方式可以选择按 COM 扫描或者按 SEG 扫描，COM 口最大支持 8 个，按 COM 扫描时可以根据实际应用从支持的 8 个 COM 中选择任意的 1~8 个 COM。SEG 口总共 12 个，按 SEG 扫描时也可以根据实际应用选择支持的 12 个 SEG 中的任意 1~12 个。

按 COM 口扫描:



按 SEG 口扫描:



支持 SEG 口与 TK 共用 IO：

当 TK 与数码管 SEG 口复用 IO 时，复用的 IO 同一时间只能用来驱动 LED 或者用来扫描触摸按键。在驱动 LED 时，SEG 口输出高/低电平。在 TK 扫描阶段，SEG 口由硬件切换成模拟功能。

复用模式下 LED 的 COM 口在非 LED 驱动阶段，一直保持为关闭状态。

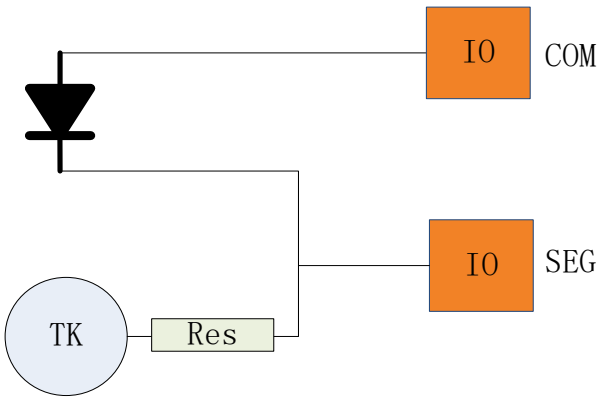


图 18-1 复用模式下的外围电路连接图

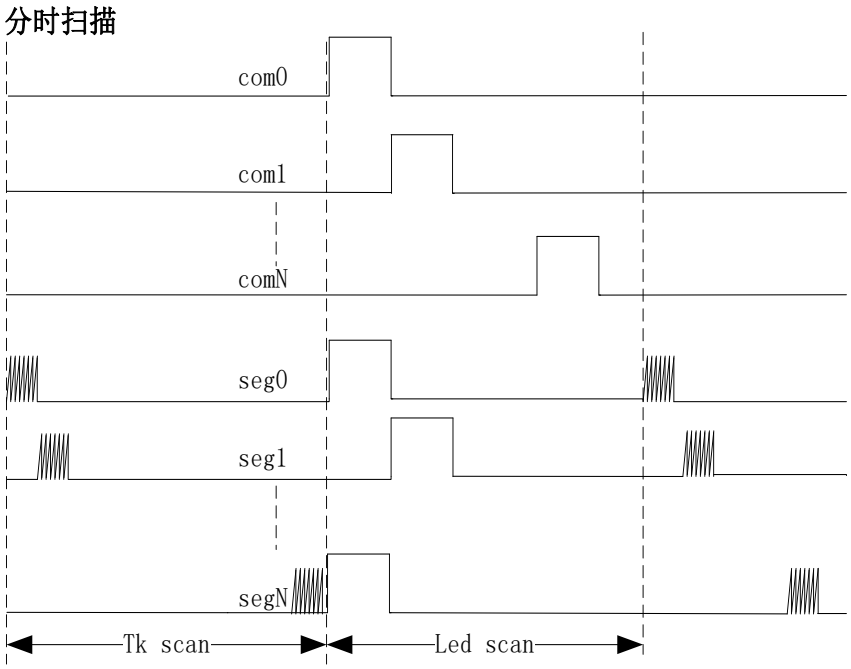


图 18-2 TK、LED 复用 IO 模式下 LED 按 COM 口分时扫描的时序示意图

分时扫描：扫描 TK 和驱动 LED 是完全分时进行的，如上图所示，TK scan 阶段把所有按键都扫描一遍，所有 COM 口都是关闭状态。Led scan 阶段，无论按键有没有和 SEG 口复用都停止扫描，这段时间只驱动 LED。

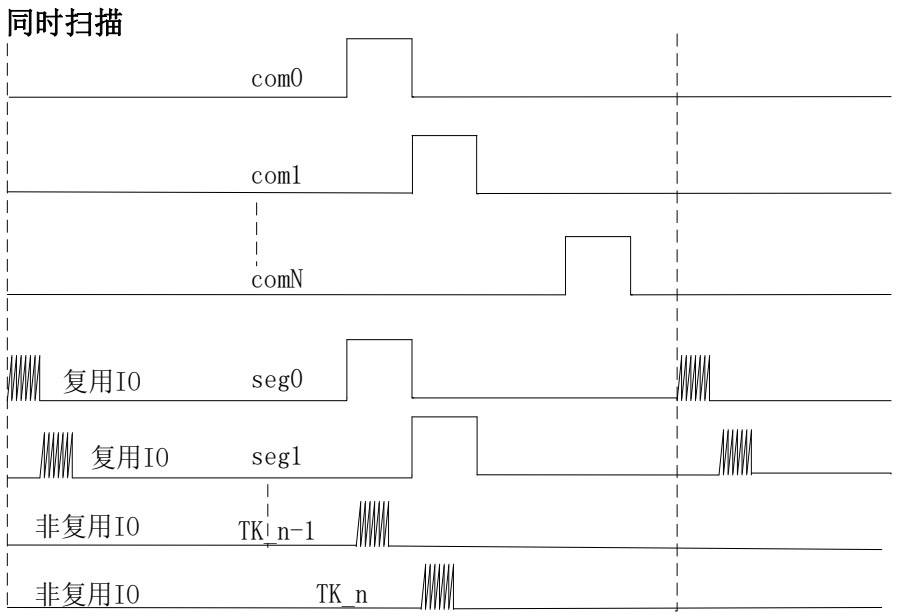


图 18-3 TK、LED 复用 IO 模式下 LED 按 COM 口同时扫描的时序示意图

同时扫描：同时扫描并不是同一个 IO 在同一时间同时用来做触摸按键和驱动 LED，而是 LED 驱动模块和 TK 扫描模块是同时在工作的，同一个 IO 同一时间还是只能用作其中一个功能的。

这种模式的好处是在有部分触摸按键没有和 LED 共用 IO 的应用中，在驱动 LED 的时间里，可以同时扫描那些没有共用 IO 的按键，节省时间。

支持 LED SEG 口驱动能力由硬件切换：

该功能是在 LED SEG 口与 TK 复用 IO 时使用，用作 TK 的 IO 在未扫描到该 IO 时需要用强驱动能力模式输出低电平以固定该 IO 的状态，但该 IO 作为 LED SEG 口扫描时可能只需要较弱的驱动模式，此时可以硬件把该 IO 的驱动能力切换成 LED 所需要的驱动能力模式。

（操作寄存器：LED_CON0->LED_DRV_EN 和 LED_CON0->LED_DRV）

18.4 寄存器描述

18.4.1 寄存器: LED_CON0

Address offset: 0x00

Name	Bit	Reset	ACCESS	Description
Reserved	31:30	0	RO	--
LED_DRV_EN	29	0	RW	LED SEG 口驱动能力硬件控制使能： 1：使能 0：不使能
LED_DRV	28:25	0	RW	LED SEG 口驱动能力选择，当 LED_DRV_EN=1 时，SEG 口的驱动能力由 LED_DRV 控制。
TMRTIM	24:4	0	RW	timer 定时时间，LED 自动模式时，根据设定的定时时间启动扫描。即，每间隔 TMRTIM 个 pclk 启动一次扫描。
AUTOMODE	3	0	RW	自动扫描模式，该模式下自动定时扫描数码管：

				0：非自动扫描 1：自动扫描
SCANMODE	2	0	RW	扫描方式： 0：按 COM 口扫描 1：按 SEG 口扫描
PLYMODE	1	0	RW	数码管公共极性选择： 0：共阴 1：共阳
LEDEN	0	0	RW	数码管控制使能： 0：不使能 1：使能

18.4.2 寄存器: LED_CON1

Address offset: 0x04

Name	Bit	Reset	ACCESS	Description
Reserved	31:23	0	R	--
COMDIVT	22	0	RW	按 COM 扫描时，使能 COMDIVT 让 12 个 SEG 口分时扫描，使能后，使能一个 COM 时，先扫描低 6 个 SEG，再扫描高 6 个 SEG。 0：不使能 1：使能
SCANTIM	21:4	0	RW	扫描时间，每一个 COM/SEG 口的点亮时间：点亮时间为 SCANTIM 个 led_clk。 当 COMDIVT 为 1 时，扫描时间应该设置成 COMDIVT 为 0 时的一半。
SCANNUM	3:0	0	RW	扫描 COM/SEG 口的个数,COM 口最大支持 8 个，SEG 口最大支持 12 个。 实际 scan number = SCANNUM+1

18.4.3 寄存器: LED_CON2

Address offset: 0x08

Name	Bit	Reset	ACCESS	Description
Reserved	31:20	0	R	--
LED_COM_MAP	19:12	0	RW	数码管 COM 口选择，把对应 bit 置 1，即使能该 COM 口。 （按 COM 口扫描时有效）

LED_SEG_MAP	11:0	0	RW	数码管 SEG 口选择，把对应 bit 置 1，即使能该 SEG 口。

18.4.4 寄存器: LED_BADR

Address offset: 0x0C

Name	Bit	Reset	ACCESS	Description
LEDBADR	31:0	0	RW	数码管显示数据缓存区域基地址，byte 地址。 **注意：**需要 4byte 对齐

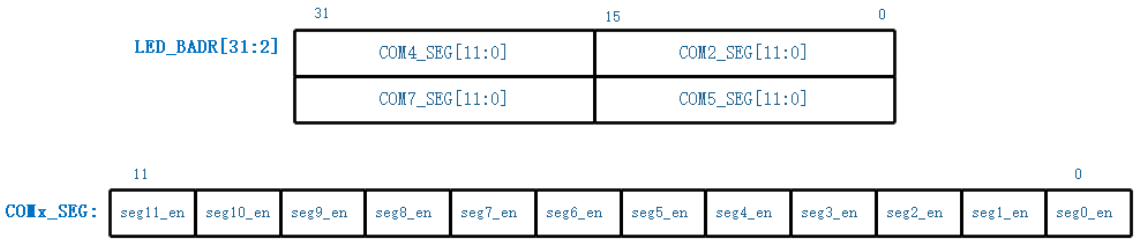
18.5 软件使用说明

18.5.1 LED 显存数据格式

LED 显示的数据从配置的 buffer 基地址 LED_BADR 开始存放，根据按 COM 和按 SEG 扫描有不同的存放格式要求。

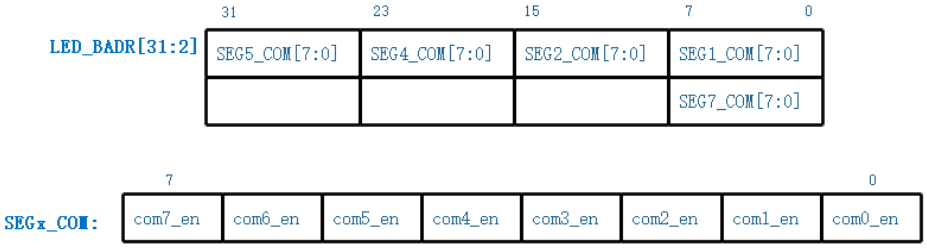
1. 按 COM 扫描时，存放的数据与 SEG 对应，总共 12 个 SEG 口，所以每个 COM 口对应有 12bit 数据。这 12bit 数据每 1bit 控制一个 SEG 口是否使能，1 为使能，0 则不使能。LED_BADR 中 COMx_SEG 组的占用的顺序为 COM 使能位的顺序从小到大排列的数据值（每 32bit 中装有两个 COM_SEG 组的数据，其中 0-11bit 为第一个 COM_SEG 数据，16-31bit 为下一个 COM_SEG 数据，依照顺序排列），而每一个 COM_SEG 组的数据对应的 seg 数据的使能位。

例如：数码管驱动按 COM 扫描，且 COM 口使能为 COM2，COM4，COM5，COM7,则 LED_BADR 中的数据将按如下排列：



2. 按 SEG 扫描时，存放的数据与 COM 对应，总共 8 个 COM 口，所以每个 SEG 口对应有 8bit 数据决定扫描该 SEG 口时，8 个 COM 口是否使能，1 为使能 SEG 口，0 则不使能。LED_BADR 中 SEGx_COM 组的占用的顺序为 SEG 使能位的顺序从小到大排列的数据值（每 32bit 中装有两个 seg 的数据，其中 0-7bit 为第一个 SEG_COM 数据，7-15bit 为第二个 SEG_COM 数据，15-23bit 为第三个 SEG_COM 数据，23-31bit 为第四个 SEG_COM 数据，依照顺序排列），而每一个 SEG_COM 组的数据对应的 com 数据的使能位。

例如：数码管按 SEG 扫描，且 SEG 口使能为 SEG1，SEG2，SEG4，SEG5，SEG7,LED_BADR 中的数据将按如下排列：



18.5.2 配置流程

1. 配置 IO 模式和 IO function。
2. 配置显存 buffer 基地址：LED_BADR，注意需要 4byte 对齐。
3. 初始化显存 buffer 数据。
4. 配置 LED COM、SEG IO 选择：操作 LED_CON2。
5. 配置需要扫描的 COM/SEG 口数量、扫描 COM/SEG 时的使能时间和按 COM 扫时是否分时扫描（COMDIVT）：操作 LED_CON1。
6. 配置自动扫描的定时时长、自动模式、扫描模式、共阳/共阴模式、驱动能力配置及使能 led_en：操作 LED_CON0

19 触摸按键

19.1 简介

本芯片采用的是电容式触摸按键。它可以穿透绝缘材料外壳 5mm (玻璃、塑料等等)以上，准确无误地侦测到手指的有效触摸。并保证了产品的灵敏度、稳定性、可靠性等不会因环境条件的改变或长期使用而发生变化，并具有防水和强抗干扰能力，超强防护，超强适应温度范围。适用于遥控器、灯具调光、各类开关以及车载、小家电和家用电器控制界面等应用中。

19.2 主要特性

- 支持最多 20 个触摸按键
- 按键基准值自动跟踪校准
- 按键状态输出防抖处理
- TK 支持软件触发扫描
- TK 支持硬件自动定时扫描
- TK 支持和 LED 的 SEG 口共用 IO

19.3 功能描述

电容式触摸按键是依据人体或带电物体靠近传感极点时，导致自电容变化的原理，通过检测电容变化来确定按键是否被触摸。（注意当使用电容式触摸时：20Pin、24Pin、28Pin 芯片的 PB4（48Pin 芯片的 PB8 引脚）需外挂陶瓷电容，必须使用 10%高精度的 NPO 或 X7R 材质的电容，容值为 1nF~10nF 之间，推荐使用 6.8nF 电容，可根据不同方案进行调节。）

在电荷转移模式下，外部触摸弹簧通过 IO 接到模拟电路。开始扫描时，先对外部寄生电容充电，然后断开充电开关 sw1，再闭合外部触摸弹簧与参考电容的连接的开关 sw2，使扫描通道上的电荷与参考电容之间进行电荷平衡。平衡后，将再次对寄生电容充电，然后再次平衡，直到参考电容的电压比比较器另一端的电压大，比较器输出翻转。比较器输出由高电平变成低电平时，代表对外部电容充电完成，此时闭合参考电容对地的开关，使参考电容放电，放电完后再次执行上述步骤充电。当人体靠近感应电极时，外部电容增加，即每次电荷平衡时外部电容所带的电荷量增加，使得对参考电容充满电的时间缩短。系统时钟 counter 对充电时间计数，当计数值变小超过一定阈值时，认为有手指触摸。

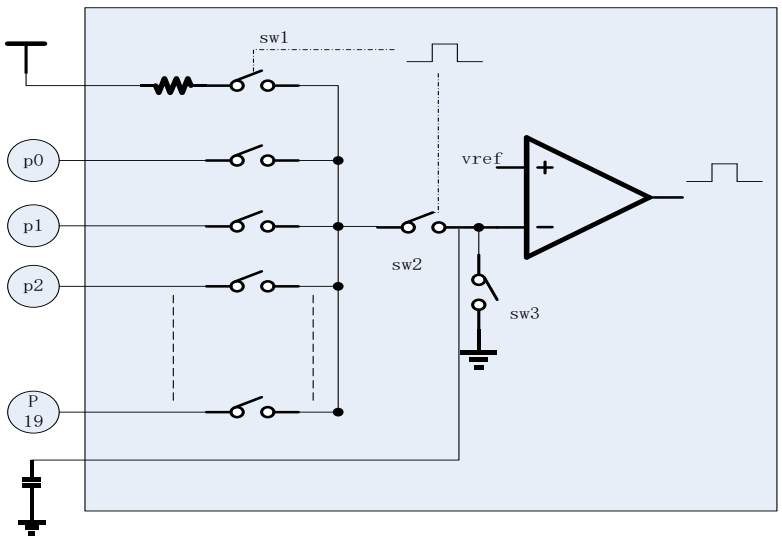


图 19-1 电荷转移模式电路

电荷转移结构实现的功能是通过把电容转化成方波的形式给数字模块处理。TK 的数字处理模块利用固定频率的时钟采样软件设定的方波周期数，并这段时间内让计数器持续计数，最终的计数值就反映了当前电容值，这就是对每个按键的电容转换成数值的过程。数字滤波模块对这个数值作滤波运算，当滤波后的数值超过设定的阈值时，认为有按键被触发。

19.4 寄存器描述

19.4.1 寄存器: TKCON

Name	Bit	Reset	ACCESS	Description
Reserved	31:30	0	RW	--
SAMPTIMEDT	29	0	RW	采样超时检测使能，在非固定采样时间模式下，使能后可以检测采样是否超时。 1：使能采样超时检测 0：不使能采样超时检测
BFLTMODE	28:26	3	RW	基准值滤波模式选择： BFLTMODE 可设置为 1~7，数值越大，响应时间越长，滤波效果越好,响应速度越慢。 BFLTMODE 与滤波算法公式中的 N 对应关系如下： N=BFLTMODE+1 注意：不能设置成 0
FLTMODE	25:23	2	RW	采样值滤波模式： FLTMODE 可设置为 0~6，数值越大，响应时间越长，滤波效果越好，响应速度越慢。 FLTMODE 与滤波算法公式中的 N 对应关系如下： N=FLTMODE+1 注意：不能设置成 7
DISCHTIM	22:15	0	RW	放电时间控制：（电荷转移模式下有效）,保证电荷转移模式下能够把外部电容的电放完。 放电时间为 DISCHTIM*T(tk_clk)
MMFLTMODE	14:13	0	RW	Max Min 滤波模式选择： 0：不做去除最大最小值滤波 1：去掉一个最大和一个最小值 2：去掉两个最大和两个最小值
Reserved	12:11	0	RO	--
FSDIV	10:7	0	RW	电荷转移模式充电开关分频比：对 tk_clk 分频，作为充电开关信号。分频比=2*(FSDIV+1)
RCINV	6	0	RW	触摸按键 RC 信号取反 0：不取反 1：取反
CHMODE	5	0	RW	振荡器模式： 0：无 1：电荷转移模式

OFFSETMODE	4	0	RW	触摸按键阈值模式 0：绝对值模式 1：百分比模式
AUTOSCAN	3	0	RW	自动扫描模式： 0：非自动扫描，每一次扫描都需要软件或者硬件触发。 1：自动扫描，扫描完一轮之后接着进行下一轮扫描。
FIXPR	2	0	RW	按键扫描时间模式: 0：每个按键扫描时间不固定，根据实时震荡频率变化。 1：每个按键扫描时间固定，由软件设定。
TKBUSY	1	0	RW	启动触摸按键扫描，软件写 1 触发扫描 Read: 0：触摸按键处于非扫描状态 1：触摸按键处于扫描状态
TKEN	0	0	RW	触摸按键使能 0：不使能 1：使能

19.4.2 寄存器:OFFSET0

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C1_OFFSET	23:12	0	RW	通道 1 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096
C0_OFFSET	11:0	0	RW	通道 0 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.3 寄存器: OFFSET1

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C3_OFFSET	23:12	0	RW	通道 3 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096
C2_OFFSET	11:0	0	RW	通道 2 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.4 寄存器: OFFSET2

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C5_OFFSET	23:12	0	RW	通道 5 触发阈值，由 OFFSETMODE 决定是百分比或绝对值

				百分比：OFFSET/4096
C4_OFFSET	11:0	0	RW	通道 4 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.5 寄存器: OFFSET3

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C7_OFFSET	23:12	0	RW	通道 7 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096
C6_OFFSET	11:0	0	RW	通道 6 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.6 寄存器: OFFSET4

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C9_OFFSET	23:12	0	RW	通道 9 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096
C8_OFFSET	11:0	0	RW	通道 8 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.7 寄存器: OFFSET5

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C11_OFFSET	23:12	0	RW	通道 11 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096
C10_OFFSET	11:0	0	RW	通道 10 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.8 寄存器: OFFSET6

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C13_OFFSET	23:12	0	RW	通道 13 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

C12_OFFSET	11:0	0	RW	通道 12 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.9 寄存器: OFFSET7

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C15_OFFSET	23:12	0	RW	通道 15 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096
C14_OFFSET	11:0	0	RW	通道 14 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.10 寄存器: OFFSET8

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C17_OFFSET	23:12	0	RW	通道 17 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096
C16_OFFSET	11:0	0	RW	通道 16 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.11 寄存器: OFFSET9

Name	Bit	Reset	ACCESS	Description
Reserved	31:24	0	RO	--
C19_OFFSET	23:12	0	RW	通道 19 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096
C18_OFFSET	11:0	0	RW	通道 18 触发阈值，由 OFFSETMODE 决定是百分比或绝对值 百分比：OFFSET/4096

19.4.12 寄存器: STBTHR

Name	Bit	Reset	ACCESS	Description
reserved	31:20	0	RW	--
BDECMODE	19:17	0	RW	基准值采样率倍速模式： 当手指频繁触摸按键时基准值可能向采样值方向靠拢，设置 BDECMODE 让基准值在手指松开时倍速回到稳定值。加速的方法是在原本设定的

				降采样基础上提高采样率。 基准值采样率加速倍数=2^BDECMODE **注意：**BDECMODE 需要和 BSDLYNUM 配合使用，要求 2^BDECMODE<BSDLYNUM
RCDBNUM	16:13	0	RW	触摸按键振荡器信号去抖次数：数字模块对模拟 TK_RC 做 RCDBNUM+1 次去抖后才用来采样。
TKDBNUM	12:10	0	RW	触摸按键输出状态去抖次数：对按键输出状态做 TKDBNUM+1 次去抖后才输出。
MMSTBTHR	9:0	0	RW	当前采样稳定阈值：对当前 N 个采样值的最大最小值作差，当差值大于 MMSTHTHR 时，认为当前采样不稳定，丢弃采样值，不做滤波计算及触摸事件判断。

19.4.13 寄存器: SAMPCON

Name	Bit	Reset	ACCESS	Description
Reserved	31:30	0	RW	--
CHSWUP_4	29	0	RW	电荷转移模式下，可以选择充电开关翻转 N 次后进入加速模式，加速模式可以选择加速 2 倍或 4 倍： 0:加速 2 倍 (FSDIV/2) 1:加速 4 倍 (FSDIV/4) 注意：需要和 FSDIV 配合使用
CHSWUP	28:27	0	RW	电荷转移模式下，可以选择充电开关翻转 N 次（按下降沿计算）后进入加速模式： 0：不加速 1：翻转 256 次后加速 2：翻转 512 次后加速 3：翻转 1024 次后加速
SW_EN_RDY	26	0	WO	软件对该 bit 写 1，使所有按键通道强行 ready。
BASESTBNUM	25:23	0	RW	基准值稳定系数： 当基准值滤波器的输出维持稳定 BASESTBNUM 次不变时认为基准值稳定，此时 ready 标志信号置 1。
BSDLYNUM	22:15	0	RW	基准值降采样级数，可设置 0~255 级降采样。
SPSTEPNUM	14:10	0	RW	对每个按键每次扫描时采样次数（>0）：该配置必须大于 0 1：采样 1 次 2：采样 2 次 ..... 31：采样 31 次
RCPOSNUM	9:5	0	RW	每次采样 tk_rc 上升沿的个数： 0：采样 1 个上升沿 1：采样 2 个上升沿 2：采样 3 个上升沿 .....

				31：采样 32 个上升沿
PRECYCNUM	4:0	0	RW	模拟 RC 信号稳定周期数： 对每一个通道采样时，前 PRECYCNUM+1 个 CYCLE 为等待时间，不进行采样计数。 (电荷转移模式时只有 PRECYCNUM[2:0]有效)

19.4.14 寄存器: SPTIM

Name	Bit	Reset	ACCESS	Description
TKWTIM	31:16	0	RW	触摸按键等待时间： LED 和触摸按键复用时： LED 扫描结束后等待 TKWTIM 个 tk_clk 才开始下一轮按键扫描。 如果不需要等待，则把 TKWTIM 设置成 0
SAMPTIM	15:0	0	RW	每个按键的扫描时间为 SAMPTIM 个 tk_hclk 周期。 扫描按键时如果在设定时间内没完成扫描会产生扫描超时 pending

19.4.15 寄存器: KEYEN

Name	Bit	Reset	ACCESS	Description
Reserved	31:25	0	RW	--
CHANNUM	24:20	0	RW	按键使能个数： 实际按键个数=CHANNUM+1
KEYEN	19:0	0	RW	Key0~key19 对应的开关控制位： 0：不使能 1：使能

19.4.16 寄存器: TKIE

Name	Bit	Reset	ACCESS	Description
Reserved	31:4	0	R	--
SCDN_IE	3	0	RW	单按键扫描完成中断使能： 0：不使能 1：使能
SPOVF_IE	2	0	RW	采样值溢出中断使能： 0：不使能 1：使能
SCNOVT_IE	1	0	RW	扫描时间超时中断使能： 0：不使能 1：使能
KEYIRQ_IE	0	0	RW	按键触发中断使能 0：不使能 1：使能

19.4.17 寄存器: INTPND

Name	Bit	Reset	ACCESS	Description
reserved	31:4	0	R	--
SCDN_PND	3	0	RW	单按键扫描结束中断 pending，硬件置 1，软件写 1 清零。
SPOVF_PND	2	0	RW	采样累加值溢出中断 pending，硬件置 1，软件写 1 清零。
SCANOVT_PND	1	0	RW	单按键扫描超时中断 pending，硬件置 1，软件写 1 清零。
KEYIRQ_PND	0	0	RW	按键触发请求中断 pending，硬件置 1，软件写 1 清零。

19.4.18 寄存器: KEYPND

Name	Bit	Reset	ACCESS	Description
Reserved	31:20	0	R	--
KEYPND	19:0	0	RW	按键触发事件 pending 标志位(硬件置 1，软件写 1 清零) bit0~bit19 对应 key0~key19 0：没有按键触发请求 1：有按键触发请求

19.4.19 寄存器: SCOVPND

Name	Bit	Reset	ACCESS	Description
Reserved	31:20	0	R	--
Scan over time pend	19:0	0	RW	扫描超时 pending 标志(硬件置 1，软件写 1 清零) bit0~bit19 对应 key0~key19 0:没有扫描超时事件 1:有扫描超时事件

19.4.20 寄存器: SCDOPND

Name	Bit	Reset	ACCESS	Description
Reserved	31:20	0	R	--
SCAN DONE PEND	19:0	0	RW	扫描完成 pending 标志(硬件置 1，软件写 1 清零) bit0~bit19 对应 key0~key19 0:没有扫描完成事件 1:有扫描完成事件

19.4.21 寄存器: SAMPVAL

Name	Bit	Reset	ACCESS	Description

Reserved	31:16	0	R	--
SAMPVAL	15:0	0	RO	按键采样值，单个按键采样完成后会起 pending，并把采样值保存到该寄存器上

19.4.22 寄存器:TKFILTVAL

Name	Bit	Reset	ACCESS	Description
reserved	31:16	0	RO	--
FILTVAL	15:0	0	RO	按键采样值经过滤波器后的输出值，由于时多个按键分时扫描，所以根据 SCDOPND 判断当前 FILTVAL 对应哪个按键。

19.4.23 寄存器: BASEVAL

Name	Bit	Reset	ACCESS	Description
reserved	31:16	0	RO	--
BASEVAL	15:0	0	RO	按键基准值经过滤波器后的输出值，由于时多个按键分时扫描，所以根据 SCDOPND 判断当前 BASEVAL 对应哪个按键。

19.4.24 寄存器:TKKEYRDY

Name	Bit	Reset	ACCESS	Description
reserved	31:20	0	RO	--
KEYRDY	19:0	0	RO	按键准备状态标志，每一 bit 对应一个按键 Read: 0：基准值初始化未完成 1：基准值初始化完成 Write: 1:清除 ready 标志位，让硬件重新校准基准值

19.4.25 寄存器:KEYSTA

Name	Bit	Reset	ACCESS	Description
Reserved	31:20	0	R	--
KEY_STA	19:0	0	RO	按键状态，每一 bit 对应一个按键 0：没有触摸状态 1：有触摸状态

19.4.26 寄存器: TKBADR

Name	Bit	Reset	ACCESS	Description

TKBADR	31:0	0	RW	触摸按键数据缓存基地址，byte 地址。 注意：需要 4byte 对齐。

19.4.27 寄存器: TKANACON0

Name	Bit	Reset	ACCESS	Description
Reserved	31:22	0	RO	--
TK_TEST_OE	21	0	RW	使能模拟 RC 信号输出，使能后从 PC1 输出 TK 模拟振荡器信号。 0：不使能 1：使能
TKEY_HWDIS	20	0	RW	TKEY_RCEN 和 TKEY_CCEN 硬件控制使能: 0:TKEY_RCEN 和 TKEY_CCEN 只由软件控制 1：TKEY_RCEN 和 TKEY_CCEN 在 TKEY_CHAN_EN 为 0 时由硬件强制关闭（TKEY_CHAN_EN 由硬件控制）
Reserved	19	0	RO	
TKEY_CCEN	18	0	RW	模拟甩电荷模式使能： 0：不使能 1：使能
TKEY_VREFSEL	17	0	RW	TKEY LDO 参考模式选择: 0=基电压; 1=电流
TKEY_TSEN2	16	0	RW	TKEY Cap-less VREF output to ATSOUT
TKEY_TSEN1	15	0	RW	TKEY Cap-less VDDTK output to ATSOUT
Reserved	14:13	0	RO	--
TKEY_RCFTUNE1	12	0	RW	TKEY RC mode; RC 频率微调选择;
TKEY_RCFTUNE0	11	0	RW	
Reserved	10:8	0	RO	--
TKEY_RCCAP	7	0	RW	TKEY RC mode inner cap 使能: 0： 不使能 1： 使能
TKEY_RCBANK1	6	1	RW	TKEY RC mode; RC 频率微调选择
TKEY_RCBANK0	5	1	RW	
TKEY_CCVRS1	4	0	RW	TKEY charge cap mode: 比较器输入电压选择 2’b00=4/8*VDDTK 2’b01=3/8*VDDTK 2’b10=5/8*VDDTK 2’b11=7/8*VDDTK
TKEY_CCVRS0	3	0	RW	

TKEY_CCFBEN	2	0	RW	TKEY charge cap mode，当地反馈使能 0：不使能 1：使能
TKEY_LDOS	1	0	RW	TKEY LDO 电压选择 0=1.5v 1=1.6v
TKEY_LDOEN	0	0	RW	TKEY LDO 使能 0=不使能 1=使能

19.4.28 寄存器:TK_NOISETHR

Name	Bit	Reset	ACCESS	Description
BASNOISEPND	31	0	RO	基准值噪声检测标志位 Read: 0：无噪声 1：有噪声
FLTNOISEPND	30	0	RO	采样值噪声检测标志位 Read: 0：无噪声 1：有噪声
BASEDBNUM	29:25	0	RW	基准值去抖次数：当采样值超过基准值的去抖阈值时，需要满足 BASEDBNUM 次去抖后才更新基准值。
BASEDBTHR	24:13	0	RW	基准值去抖阈值：按百分比计算，当采样值与基准值的差值超过 BASEVAL*BASEDBTHR/4096 时，对基准值更新做防抖处理。（电荷转移模式是基准值变大做防抖）
NOISE_DT_EN	12	0	RW	噪声检测使能： 0：不能使 1：使能
NOISETHR	11:0	0	RW	噪声阈值，按百分比计算，当采样值偏差大于原来的滤波值的NOISETHR/4096时，认为是噪声，将忽略当前采样值。

19.4.29 寄存器:TK_NOISETHRLT

Name	Bit	Reset	ACCESS	Description
reserved	31:24	0	RO	--
BASEDBTHRLT	23:12	0	RW	与 LED 复用 IO 的 TK 的基准值防抖阈值，按百分比计算，当采样值小于

				BASEVAL*BASEDBTHRLT/4096 时，对基准值更新做防抖处理。
NOISETHRLT	11:0	0	RW	与 LED 复用 IO 的 TK 的噪声阈值，按百分比计算，当采样值偏差大于原来的滤波值的 NOISETHRLT/4096 时，认为是噪声，将忽略当前采样值。

19.4.30 寄存器: TK_NOISETHRSEL

Name	Bit	Reset	ACCESS	Description
reserved	31:20	0	RO	--
NOISETHR_SEL	19:0	0	RW	按键基准值防抖阈值和噪声阈值选择： 0：选择 TK_NOISETHR 的阈值 1：选择 TK_NOISETHRLT 的阈值

19.4.31 寄存器: TK_ANAFRQCON0

Name	Bit	Reset	ACCESS	Description
reserved	31:30	0	RO	--
TKEY5_RCCIS	29:27	4	RW	TKEY5 RC mode; RC 充电器电流选择;
TKEY5_RCFTUNE	26:25	2	RW	TKEY5 RC mode; RC 频率微调选择;
TKEY4_RCCIS	24:22	4	RW	TKEY4 RC mode; RC 充电器电流选择;
TKEY4_RCFTUNE	21:20	2	RW	TKEY4 RC mode; RC 频率微调选择;
TKEY3_RCCIS	19:17	4	RW	TKEY3 RC mode; RC 充电器电流选择;
TKEY3_RCFTUNE	16:15	2	RW	TKEY3 RC mode; RC 频率微调选择;
TKEY2_RCCIS	14:12	4	RW	TKEY2 RC mode; RC 充电器电流选择;
TKEY2_RCFTUNE	11:10	2	RW	TKEY2 RC mode; RC 频率微调选择;
TKEY1_RCCIS	9:7	4	RW	TKEY1 RC mode; RC 充电器电流选择;
TKEY1_RCFTUNE	6:5	2	RW	TKEY1 RC mode; RC 频率微调选择;
TKEY0_RCCIS	4:2	4	RW	TKEY0 RC mode; RC 充电器电流选择;
TKEY0_RCFTUNE	1:0	2	RW	TKEY0 RC mode; RC 频率微调选择;

19.4.32 寄存器: TK_ANAFRQCON1

Name	Bit	Reset	ACCESS	Description
reserved	31:30	0	RO	--
TKEY11_RCCIS	29:27	4	RW	TKEY11 RC mode; RC 充电器电流选择;

TKEY11_RCFTUNE	26:25	2	RW	TKEY11 RC mode; RC 频率微调选择;
TKEY10_RCCIS	24:22	4	RW	TKEY10 RC mode; RC 充电器电流选择;
TKEY10_RCFTUNE	21:20	2	RW	TKEY10 RC mode; RC 频率微调选择;
TKEY9_RCCIS	19:17	4	RW	TKEY9 RC mode; RC 充电器电流选择;
TKEY9_RCFTUNE	16:15	2	RW	TKEY9 RC mode; RC 频率微调选择;
TKEY8_RCCIS	14:12	4	RW	TKEY8 RC mode; RC 充电器电流选择;
TKEY8_RCFTUNE	11:10	2	RW	TKEY8 RC mode; RC 频率微调选择;
TKEY7_RCCIS	9:7	4	RW	TKEY7 RC mode; RC 充电器电流选择;
TKEY7_RCFTUNE	6:5	2	RW	TKEY7 RC mode; RC 频率微调选择;
TKEY6_RCCIS	4:2	4	RW	TKEY6 RC mode; RC 充电器电流选择;
TKEY6_RCFTUNE	1:0	2	RW	TKEY6 RC mode; RC 频率微调选择;

19.4.33 寄存器: TK_ANAFRQCON2

Name	Bit	Reset	ACCESS	Description
reserved	31:30	0	RO	--
TKEY17_RCCIS	29:27	4	RW	TKEY17 RC mode; RC 充电器电流选择;
TKEY17_RCFTUNE	26:25	2	RW	TKEY17 RC mode; RC 频率微调选择;
TKEY16_RCCIS	24:22	4	RW	TKEY16 RC mode; RC 充电器电流选择;
TKEY16_RCFTUNE	21:20	2	RW	TKEY16 RC mode; RC 频率微调选择;
TKEY15_RCCIS	19:17	4	RW	TKEY15 RC mode; RC 充电器电流选择;
TKEY15_RCFTUNE	16:15	2	RW	TKEY15 RC mode; RC 频率微调选择;
TKEY14_RCCIS	14:12	4	RW	TKEY14 RC mode; RC 充电器电流选择;
TKEY14_RCFTUNE	11:10	2	RW	TKEY14 RC mode; RC 频率微调选择;
TKEY13_RCCIS	9:7	4	RW	TKEY13 RC mode; RC 充电器电流选择;
TKEY13_RCFTUNE	6:5	2	RW	TKEY13 RC mode; RC 频率微调选择;
TKEY12_RCCIS	4:2	4	RW	TKEY12 RC mode; RC 充电器电流选择;
TKEY12_RCFTUNE	1:0	2	RW	TKEY12 RC mode; RC 频率微调选择;

19.4.34 寄存器: TK_ANAFRQCON3

Name	Bit	Reset	ACCESS	Description
reserved	31:10	0	RO	--
TKEY19_RCCIS	9:7	4	RW	TKEY19 RC mode; RC 充电器电流选择;
TKEY19_RCFTUNE	6:5	2	RW	TKEY19 RC mode; RC 频率微调选择;

TKEY18_RCCIS	4:2	4	RW	TKEY18 RC mode; RC 充电器电流选择;
TKEY18_RCFTUNE	1:0	2	RW	TKEY18 RC mode; RC 频率微调选择;

19.5 软件使用说明

19.5.1 配置流程

a. 初始化时钟

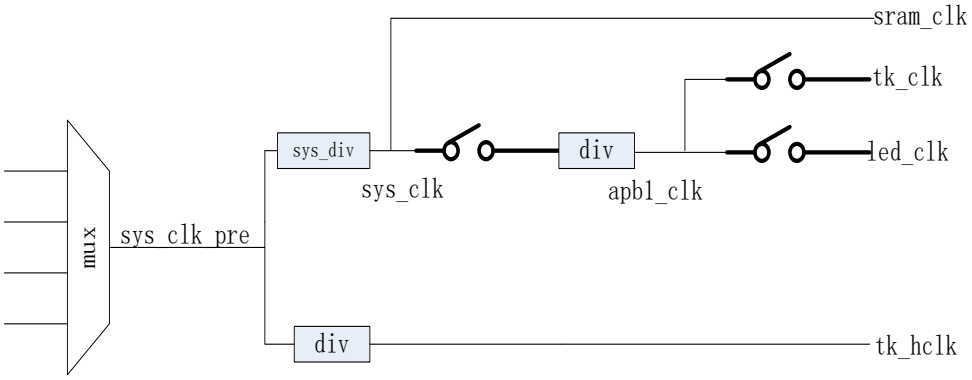


图 19-4 TK 的时钟结构

b. 初始化 IO 模式：TK 没有与 LED 共用 IO 时，需要设置使该 IO 输出低电平。

c. 初始化 TK 模拟模块

d. 初始化 TK 中断函数

e. 给 TK 开辟 buffer 空间，并设定基地址。

f. 初始化 TK buffer，全部填 0.

g. 配置 TK 数字模块

h. 使能 TK 启动扫描。

19.5.2 关于 TK buffer 说明

TK buffer 用于存放滤波器数据，当 TK 和 LED 都使能且都工作在 auto 模式时，需要开辟 45 个 word 空间，否则需要开辟空间大小为：2*N+ceil(N/4)个 word。

19.5.3 关于 TK 和 LED auto 模式配置的补充说明：

1. 当 TK 或者 LED 单独工作时，配置成 auto 模式
2. 当 TK 和 LED 复用 IO 工作时，TK 和 LED 只能同时为 auto 模式或者同时为非 auto 模式。

20 器件电子签名

电子签名存放在闪存存储器模块的系统存储区域，可以通过 JTAG、 DBG 或者 CPU 读取。它所包含的芯片识别信息在出厂时编写，用户固件或者外部设备可以读取电子签名，用以自动匹配不同配置的 TS32F020 系列微控制器。

20.1 存储器容量寄存器

20.1.1 产品唯一身份标识寄存器（96 位）

产品唯一的身份标识非常合适：

- 用来作为序列号（例如 USB 字符序列号或者其他的终端应用）
- 用来作为密码，在编写闪存时，将此唯一标识与软件加解密算法结合使用，提高代码在闪存存储器的安全性
- 用来激活带安全机制的自举过程

96 位的产品唯一身份标识所提供的参考号码对任意一个 TS32F020 系列微控制器，在任何情况下都是唯一的。用户在何种情况下，都不能修改这个身份标识。

关于这个寄存器的详细描述，请参考嵌入式闪存一章。

21 调试支持（DBG）

21.1 概述

TS32F020 系列内核内含硬件调试模块，支持两线 Debug 接口。硬件调试模块允许内核在取指（指令断点）或访问数据（数据断点）时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。

当 TS32F020 系列微控制器连接到调试器并开始调试时，调试器将使用内核的硬件调试模块进行调试操作。

CPU 内核提供集成的片上调试功能。它由以下部分组成：

- DBG-DP：串行调试端口
- AHP-AP： AHB 访问端
- ITM：执行跟踪单元
- FPB：闪存指令断点
- DWT：数据触发
- TPUI：跟踪单元接口

21.2 引脚分布和调试端口脚

TS32F020 微控制器的不同封装有不同的有效引脚数。因此，某些与引脚相关的功能可能随封装而不同。

21.2.1 DBG 调试端口脚

TS32F020 的 1 组普通 I/O 口可用作 DBG-DP 接口引脚。这些引脚在所有的封装里都存在。

DBG-DP 端口引脚名称	DBG 调试接口		引脚分配
	类型	调试功能	
DBGIO	输入/输出	串行数据输入/输出	PC7
DBGCLK	输入	串行时钟	PC6

21.2.2 DBG 脚上的内部上拉和下拉

保证 DBG 的输入引脚不是悬空的是非常必要的，因为他们直接连接到 D 触发器控制着调试模式。必须特别注意 DBGCLK 引脚，因为他们直接连接到一些 D 触发器的时钟端。

为了避免任何未受控制的 I/O 电平， TS32F020 在 DBG 输入脚上嵌入了内部上拉和下拉。

DBGIO：默认输入浮空

DBGCLK：内部上/下拉可选

一旦 DBG I/O 被用户代码释放， GPIO 控制器再次取得控制。这些 I/O 口的状态将恢复到复位时的状态。

软件可以把这些 I/O 口作为普通的 I/O 口使用。

21.3 MCU 调试

为了支持低功耗模式下（STOP/SLEEP）的调试，当 DBG 连接成功后，如果芯片处于 STOP 模式，则会被唤醒，如果芯片处于 SLEEP 模式，则会被复位。此复位不会让 DBG 失连。