

YS62F0132

数据手册（版本 V1.0）

8-Bit Touch Flash MCU

器件概述

- 基于 8051 指令流水线结构的 8 位单片机。
- Flash Rom: 32K 字节, 100,000 次可擦除, Firmware 可访问, 用户数据保护, 支持 16bit CRC。
- **RAM:**
 - IRAM 空间: 256B。
 - XRAM 空间: 8KB (1KB 通用, 7KB 专用)。
 - SFR 空间: 128B。
- **工作电压:**
VDD=2.5V~5V Fosc=22MHZ / 32KHZ (内部振荡)。
- 最多 26 个 GPIO, 可配置 **VDDIO**=1.8/3.3V。
- 内部集成自电容和互电容触摸检测, 用户可选择其中之一进行扫描检测。
- 互电容触摸检测时报点率可达 125Hz。
- 支持互电容触摸通道数: 12RX * 20TX
- 支持互电容真实 5 点触控检测。
- 支持自电容 32 个触摸通道。
- 3 个通用 16 位定时器/计数器: T0、T1、T2。
- **运行模式**
 - 正常模式
 - 待机模式
 - 睡眠
 - 深度睡眠
 - 冬眠
- **唤醒源**
 - 中断唤醒
 - 定时器唤醒
 - I/O唤醒
 - I2C唤醒
- **中断源: 2 级中断优先级**
 - 定时器
 - IO 沿中断
 - UART/SPI/I2C
 - Touch
- 内建一个 32bit / 32bit 除法器 and 16 bit x 16bit 乘法器。
- 串口通信: UART、SPI、I2C。
- 支持在线仿真功能, 支持最多 4 个断点 debug。

器件	VDD	ROM	RAM			I/O	Touch (通道数)	Timer	Interface	唤醒功能 引脚数目	Package
		FLASH (Byte)	IRAM (Byte)	XRAM (Byte)	SFP (Byte)						
YS62F0132	2.6V~5V	32K	256	8K	128	26	32	3*16bit	SPI/I2C/UART	9	QFN48、 LQFP80

目 录

1.0 器件结构及封装	7
1.1 系统结构图	7
1.2 封装脚图	8
1.3 引脚说明	11
2.0 CPU 内核	14
2.1 数据存储空间	14
2.2 FLASH 程序存储器	16
2.2.1 FLSCL 寄存器	17
2.2.2 PSCTL 寄存器	17
2.2.3 FLKEY 寄存器	18
2.3 FLASH 安全管理（软件加密）	18
2.3.1 特性	18
2.3.2 安全保护等级	18
3.0 功耗模式	20
3.1 正常模式	20
3.2 待机模式	21
3.3 睡眠模式	21
3.4 深度睡眠模式	21
3.5 PCON 寄存器	21
4.0 复位源	22
4.1 外部复位、上电复位	22
4.2 低电压侦测复位	22
4.3 看门狗溢出复位	23
4.4 FEDR 复位	23
4.5 软复位	23
4.6 过度电应力复位	23
4.7 复位寄存器	23
4.7.1 RSTSRC 复位状态寄存器	23
4.7.2 RSTEN 复位源使能寄存器	24
5.0 BOOT 过程	24
6.0 循环冗余检查单元（CRC）	24
6.1 操作步骤	24
6.1.1 计算单个字节的 CRC	24
6.1.2 批量计算 ROM 数据 CRC	25
6.2 CRC 寄存器	25
6.2.1 CRC0CN 寄存器	25
6.2.2 CRC0IN 寄存器	27
6.2.3 CRC0DAT 寄存器	27
6.2.4 CRC0ST 寄存器	28

6.2.5	CRC0CNT 寄存器.....	28
7.0	输入/输出端口.....	29
7.1	对照表	29
7.2	I/O 寄存器说明.....	30
7.2.1	FCTR 寄存器.....	30
7.2.2	P0PU 寄存器.....	31
7.2.3	P1PU 寄存器.....	32
7.2.4	P23PU 寄存器.....	32
7.2.5	P0DIR 寄存器.....	33
7.2.6	P1DIR 寄存器.....	34
7.2.7	P2DIR 寄存器.....	35
7.2.8	P3DIR 寄存器.....	36
7.2.9	P0 寄存器.....	36
7.2.10	P1 寄存器.....	37
7.2.11	P2 寄存器.....	37
7.2.12	P3 寄存器	38
8.0	硬件去抖 (DEBOUNCE)	39
8.1	去抖功能的使用.....	39
8.2	寄存器	39
8.2.1	DEBCF 寄存器.....	39
8.2.2	DEBDLY 寄存器.....	39
9.0	中断系统.....	40
9.1	中断响应延时.....	40
9.2	中断源	41
9.3	中断寄存器	41
9.3.1	IE 寄存器.....	41
9.3.2	EIE1 寄存器.....	42
9.3.3	IP 寄存器.....	42
9.3.4	EIP1 寄存器.....	42
9.3.5	IRQ0 寄存器.....	43
9.3.6	P0SEL 寄存器.....	43
9.3.7	TCON 寄存器.....	44
10.0	系统时钟.....	45
10.1	快慢切换	45
10.2	系统时钟寄存器.....	45
10.2.1	CKSEL 时钟选择寄存器。	45
10.2.2	CKCTL (CKCON) 寄存器.....	45
10.2.3	PERCF 寄存器.....	46
11.0	定时器 0/定时器 1.....	47
11.1	TIMER0/TIMER1 工作模式.....	47
11.1.1	方式 0	48
11.1.2	方式 1	48

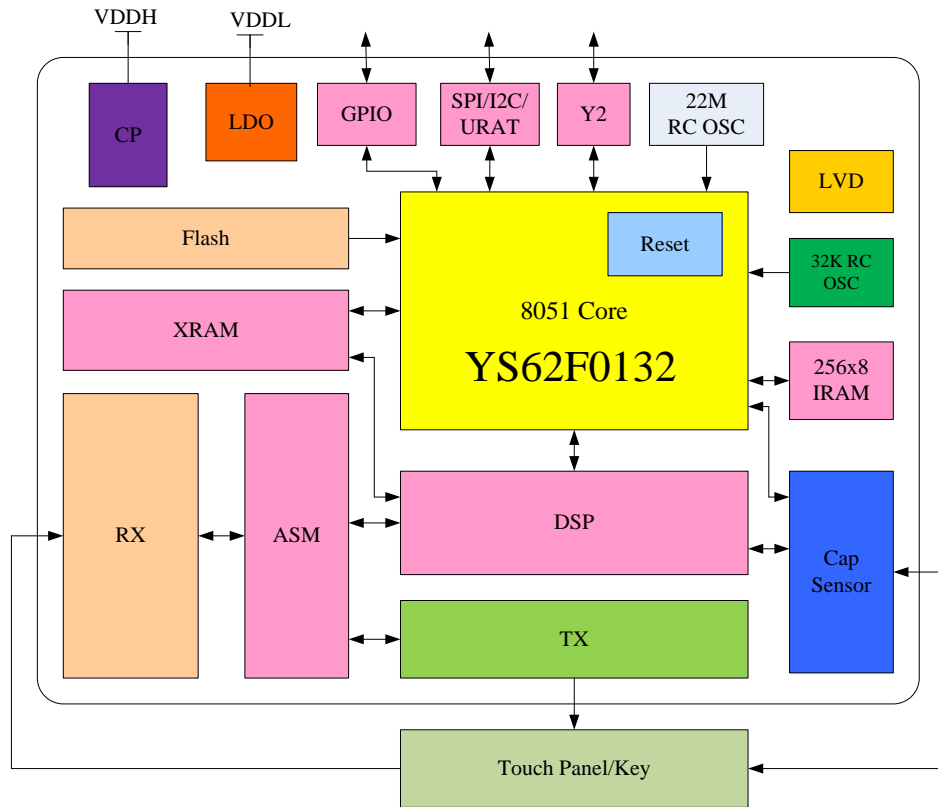
11.1.3	方式 2	48
11.1.4	方式 3	49
11.2	TIMER0/TIMER1 寄存器	49
11.2.1	CKCON 寄存器	49
11.2.2	TCON 寄存器	50
11.2.3	TMOD 寄存器	51
11.2.4	TL0 寄存器	52
11.2.5	TL1 寄存器	53
11.2.6	TH0 寄存器	53
11.2.7	TH1 寄存器	53
12.0	定时器 2	54
12.1	16 位自动重载定时器	54
12.2	8 位自动重载定时器	55
12.3	捕捉方式	56
12.4	TIMER2 寄存器	57
12.4.1	TMR2CN 寄存器	57
12.4.2	RMR2RLL 寄存器	58
12.4.3	TMR2RLH 寄存器	59
12.4.4	TMR2L 寄存器	59
12.4.5	TMR2H 寄存器	59
13.0	看门狗定时器	60
13.1	WDTCTR 寄存器	60
14.0	UART	61
14.1	波特时钟与波特率	61
14.2	工作模式	62
14.3	UART 寄存器	62
14.3.1	8 位 UART	62
14.3.2	9 位 UART	63
14.3.3	多处理器通讯	64
14.3.4	FCTR 寄存器	65
14.3.5	SCON0 寄存器	66
14.3.6	SBUF0 寄存器	67
15.0	增强型串行外设接口 (SPI)	67
15.1	信号说明	68
15.1.1	主输出、从输入 (MOSI)	68
15.1.2	主输入、从输出 (MISO)	68
15.1.3	串行时钟 (SCK)	68
15.1.4	从选择 (NSS)	68
15.2	SPI 主方式	69
15.3	SPI 从方式	70
15.4	SPI 中断源	70
15.5	SPI 寄存器描述	71
15.5.1	FCTR 寄存器	71

15.5.2	SPICFG 寄存器.....	71
15.5.3	SPICTL 寄存器.....	72
15.5.4	SPISCR 寄存器.....	73
15.5.5	SPIDAT 寄存器.....	73
16.0	I2C.....	74
16.1	时钟延长.....	75
16.2	I2C 操作步骤.....	75
16.3	I2C 寄存器.....	75
16.3.1	FCTR 寄存器.....	75
16.3.2	I2C_DIN 寄存器.....	76
16.3.3	I2C_DOUT 寄存器.....	77
16.3.4	I2C_SLAD 寄存器.....	77
16.3.5	I2C_STAT 寄存器.....	77
17.0	乘法器 16 位* 16 位	78
17.1	乘法器寄存器.....	78
17.1.1	MAC_CTRL 寄存器.....	78
17.1.2	MAC_STAT 寄存器.....	79
17.1.3	MAC0CF0 寄存器.....	79
17.1.4	MAC0AH 寄存器.....	80
17.1.5	MAC0AL 寄存器.....	80
17.1.6	MAC0BH 寄存器.....	80
17.1.7	MA0CBL 寄存器.....	81
17.1.8	MAC0ACC3 寄存器.....	81
17.1.9	MAC0ACC2 寄存器.....	81
17.1.10	MAC0ACC1 寄存器.....	81
17.1.11	MAC0ACC0 寄存器.....	81
18.0	除法器 32 位/32 位	82
18.1	除法寄存器	82
18.1.1	DIV_WA0 寄存器.....	82
18.1.2	DIV_WA1 寄存器.....	82
18.1.3	DIV_WA2 寄存器.....	83
18.1.4	DIV_WA3 寄存器.....	83
18.1.5	DIV_WB0 寄存器.....	83
18.1.6	DIV_WB1 寄存器.....	83
18.1.7	DIV_WB2 寄存器.....	83
18.1.8	DIV_WB3 寄存器.....	84
19.0	TOUCH	84
19.1	互容扫描	85
19.2	自容扫描	85
19.3	自电容寄存器列表.....	86
19.4	寄存器定义	89
19.5	自电容触摸操作流程.....	103

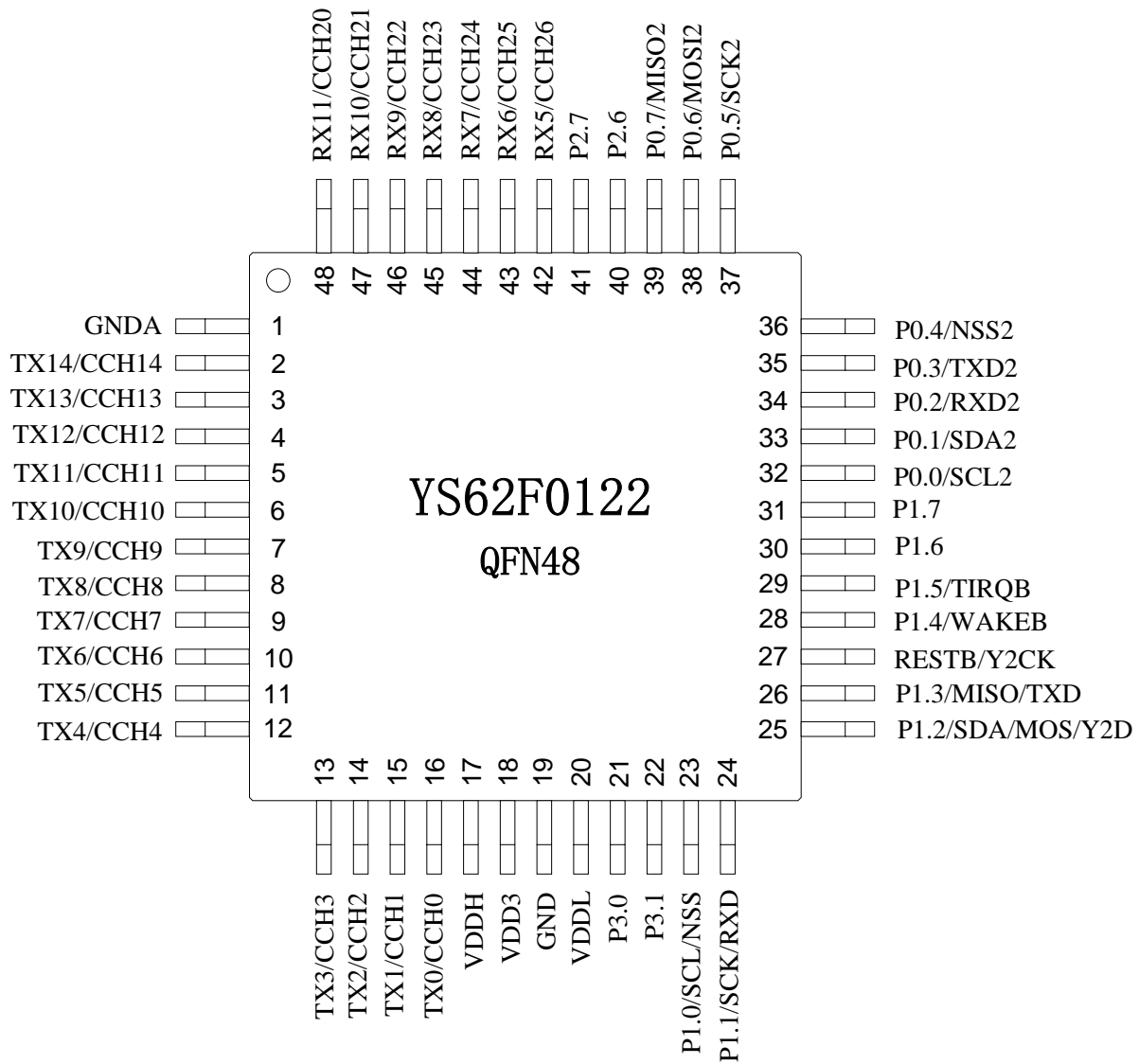
20.0 应用原理图.....	104
20.1 互容电路原理图.....	104
20.2 自容电路原理图.....	104
21.0 开发支持	105
21.1 仿真信息	105
21.1.2 软件:	105
21.1.2 硬件: YS-Link	106
21.1.3 仿真调试接口	106
21.2 烧录信息	107
21.2.1 烧录软件: YSpringPro	107
21.2.2 烧录器: YS-Writer	107
22.0 封装信息	108
22.1 QFN48L 0606x0.75-0.40	108
22.2 LQFP80L 1212x1.40	109
附录 1 FLASH 地址和块的对应关系	110
附录 2 FLASH 地址和页的对应关系	110
附录 3 自容扫描寄存器的 MCU 地址	114
23.0 汇春知识产权政策	115
23.1 专利权	115
23.2 著作权	115

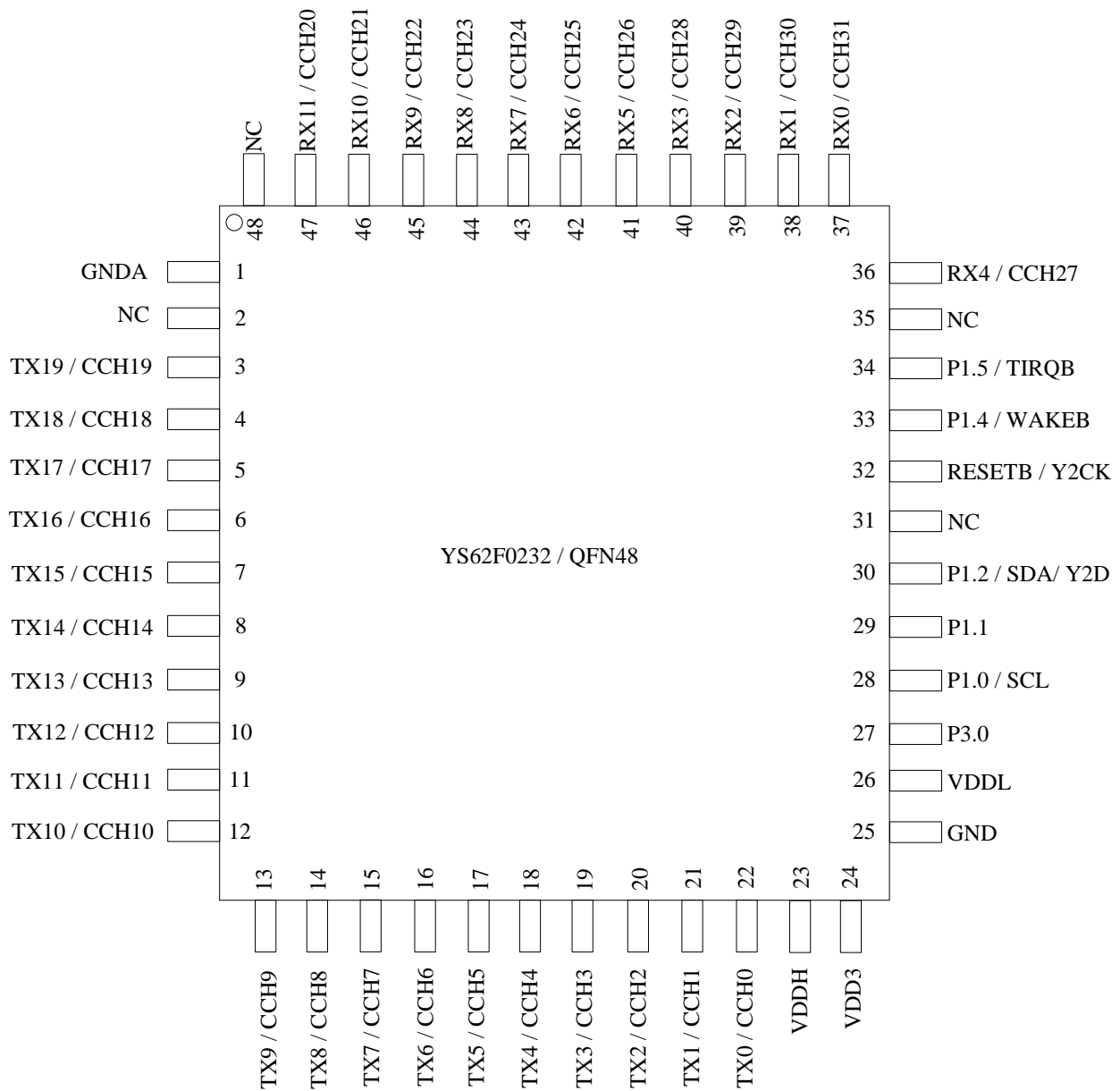
1.0 器件结构及封装

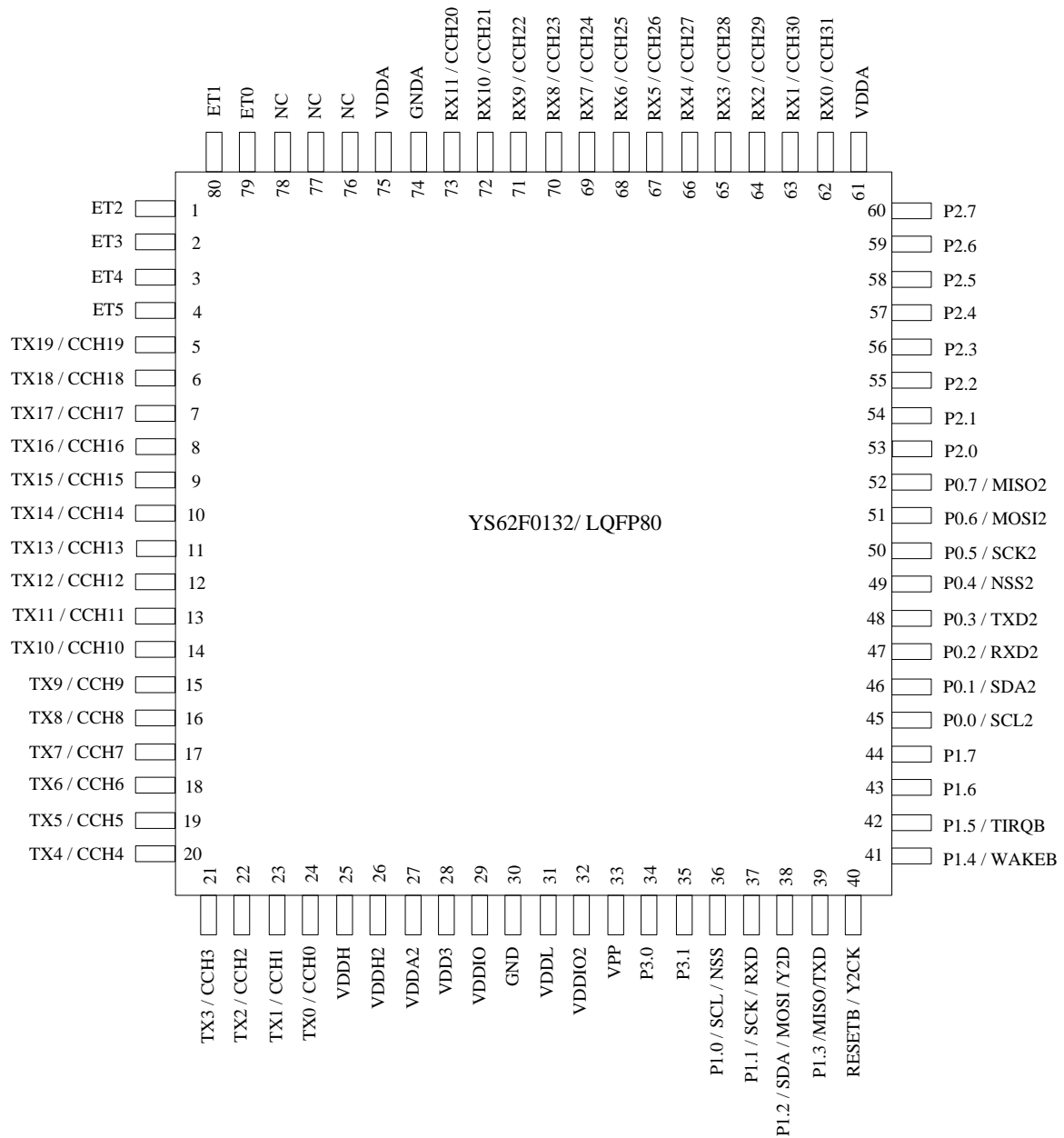
1.1 系统结构图



1.2 封装脚图







1.3 引脚说明

Name	Pin No.			Type	Description
	YS62F0122	YS62F0232	YS62F0132		
GNDA	1	1	74	PWR	Analog Ground
NC		2			Not Connected
TX19 / CCH19		3	5	O	Transmit output pin CS Channel 19
TX18 / CCH18		4	6	O	Transmit output pin CS Channel 18
TX17 / CCH17		5	7	O	Transmit output pin CS Channel 17
TX16 / CCH16		6	8	O	Transmit output pin CS Channel 16
TX15 / CCH15		7	9	O	Transmit output pin CS Channel 15
TX14 / CCH14	2	8	10	O	Transmit output pin CS Channel 14
TX13 / CCH13	3	9	11	O	Transmit output pin CS Channel 13
TX12 / CCH12	4	10	12	O	Transmit output pin CS Channel 12
TX11 / CCH11	5	11	13	O	Transmit output pin CS Channel 11
TX10 / CCH10	6	12	14	O	Transmit output pin CS Channel 10
TX9 / CCH9	7	13	15	O	Transmit output pin CS Channel 9
TX8 / CCH8	8	14	16	O	Transmit output pin CS Channel 8
TX7 / CCH7	9	15	17	O	Transmit output pin CS Channel 7
TX6 / CCH6	10	16	18	O	Transmit output pin CS Channel 6
TX5 / CCH5	11	17	19	O	Transmit output pin CS Channel 5
TX4 / CCH4	12	18	20	O	Transmit output pin CS Channel 4
TX3 / CCH3	13	19	21	O	Transmit output pin CS Channel 3
TX2 / CCH2	14	20	22	O	Transmit output pin CS Channel 2
TX1 / CCH1	15	21	23	O	Transmit output pin CS Channel 1

TX0 / CCH0	16	22	24	O	Transmit output pin CS Channel 0
VDDH	17	23	25	PWR	Generated internal 5V power supply, 1uF ceramic capacitor to ground is required
VDDH2			26	PWR	Generated internal 5V power supply, 1uF ceramic capacitor to ground is required
VDDA2			27	PWR	Generated internal 1.8V power supply, 1uF ceramic capacitor to ground is required
VDD3	18	24	28	PWR	3V power supply input
VDDIO			29	PWR	IO VDD
GND	19	25	30	PWR	Digital Ground
VDDL	20	26	31	PWR	Generated internal 1.8V power supply, 1uF ceramic capacitor to ground is required
VDDIO2			32	PWR	IO VDD
VPP			33	PWR	VPP
P3.0 /	21	27	34	B	P3.0 GPIO
P3.1	22		35	B	GPIO
P1.0 / SCL / NSS /	23	28	36	B	P1.0 GPIO I2C Clock SPI NSS
P1.1 / SCK / RXD /	24	29	37	B	P1.1 GPIO SPI Clock UART Receive Pin
P1.2 / SDA / MOSI / Y2D /	25	30	38	B	P1.2 GPIO I2C DATA pin SPI MOSI pin Y2 Data pin
NC		31			Not Connected
P1.3 / MISO / TXD /	26		39	B	P1.3 GPIO SPI MISO UART TXD
RESETB/ Y2CK	27	32	40	B	External Reset, low is active Y2 Clock
P1.4/ WAKEB	28	33	41	B	P1.4 GPIO External Interrupt from host
P1.5/ TIRQB	29	34	42	B	P1.5 GPIO External Interrupt to host
NC		35			Not Connected
RX4 / CCH27		36	66	I	Receive input pin CS Channel 27
RX0 / CCH31		37	62	I	Receive input pin CS Channel 31
RX1 / CCH30		38	63	I	Receive input pin CS Channel 30

RX2 / CCH29		39	64	I	Receive input pin CS Channel 29
RX3 / CCH28		40	65	I	Receive input pin CS Channel 28
RX5 / CCH26	42	41	67	I	Receive input pin CS Channel 26
RX6 / CCH25	43	42	68	I	Receive input pin CS Channel 25
RX7 / CCH24	44	43	69	I	Receive input pin CS Channel 24
RX8 / CCH23	45	44	70	I	Receive input pin CS Channel 23
RX9 / CCH22	46	45	71	I	Receive input pin CS Channel 22
RX10 / CCH21	47	46	72	I	Receive input pin CS Channel 21
RX11 / CCH20	48	47	73	I	Receive input pin CS Channel 20
NC		48			Not Connected
P1.6 /	30		43	B	P1.6 GPIO
P1.7 /	31		44	B	P1.7 GPIO
P0.0 / SCL2	32		45	B	P0.0 GPIO I2C_SCL2
P0.1 / SDA2	33		46	B	P0.1 GPIO I2C_SDA2
P0.2 / RXD2	34		47	B	P0.2 GPIO UART2 RXD
P0.3 / TXD2	35		48	B	P0.3 GPIO UART2 TXD
P0.4 / NSS2	36		49	B	P0.4 GPIO SPI2 NSS
P0.5 / SCK2	37		50	B	P0.5 GPIO SPI2 SCK
P0.6 / MOSI2	38		51	B	P0.6 GPIO SPI2 MOSI
P0.7 / MISO2	39		52	B	P0.7 GPIO SPI2 MISO
P2.0 /			53	B	P2.0 GPIO
P2.1 /			54	B	P2.1 GPIO
P2.2 /			55	B	P2.2 GPIO
P2.3 /			56	B	P2.3 GPIO
P2.4 /			57	B	P2.4 GPIO
P2.5 /			58	B	P2.5 GPIO
P2.6 /	40		59	B	P2.6 GPIO

P2.7 /	41		60	B	P2.7 GPIO
VDDA			61	PWR	Generated internal 1.8V power supply, 1uF ceramic capacitor to ground is required
VDDA			75	PWR	Generated internal 1.8V power supply, 1uF ceramic capacitor to ground is required
NC			76		
NC			77		
NC			78		
ET0			79	B	ESD TEST PIN0
ET1			80	B	ESD TEST PIN1
ET2			1	B	ESD TEST PIN2
ET3			2	B	ESD TEST PIN3
ET4			3	B	ESD TEST PIN4
ET5			4	B	ESD TEST PIN5

注： 1: VDD3: 2.6~3.6V
 2: VDDL、VDDA、VDDA2: 1.8V Core voltage
 3: VDDH: 5V
 4: 多功能 IO 的第一状态为复位状态，如复位状态为 GPIO 模式，则复位时为输入模式。
 5: Direction: I: Input; O: Output; B: Biput; A: Analog

2.0 CPU 内核

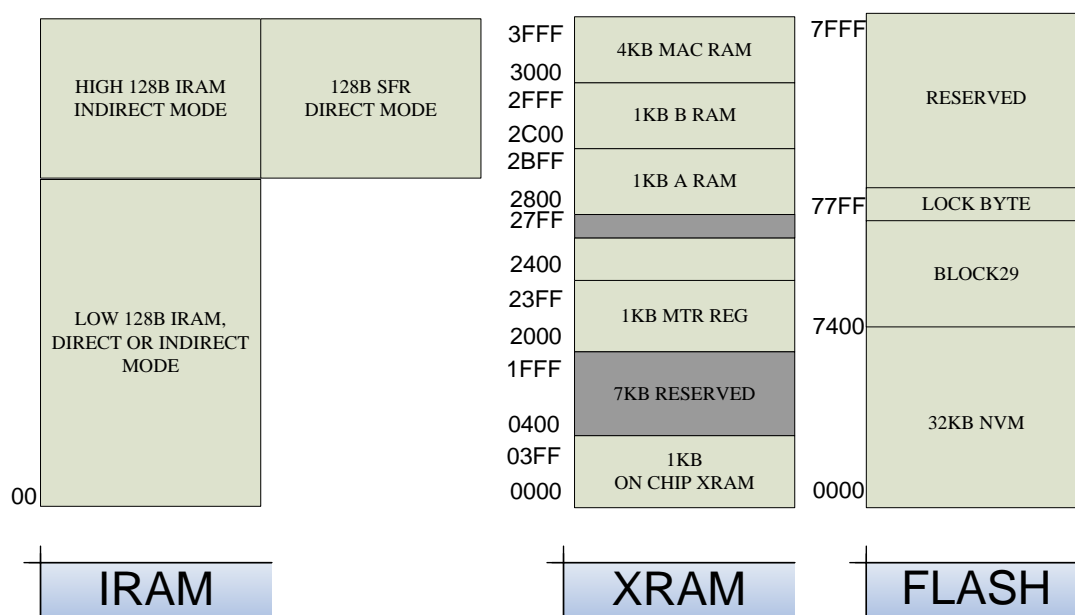
YS62F0132 的 MCU 核是流水线设计的 8051 核，指令集和标准 8051 全兼容，其流水线设计使得 70% 的指令可以在 1 到 2 个系统时钟内完成，最慢的除法指令也只需要 8 个系统时钟。YS62F0132 只支持高速 22MHz 和慢速 32KHz 内部振荡，在出厂时已经被校准为 22MHz，其精度在整个温度和电压范围内为 $\pm 1\%$ 。

YS62F0132 共有 111 条指令，下表列出了各种指令执行时间（指令执行时所需的系统时钟周期数）所对应的指令条数：

执行周期数	1	2	3	4	5	8
指令数	26	55	23	4	2	1

2.1 数据存储空间

YS62F0132 内部有 32KB 的可编程 FLASH 和 1KB 的通用的片内 XRAM，以及供 MAC 运算用的 4KB RAM。地址分配如下：



● FLASH

FLASH 地址范围是 0x0000~0x7FFF，共 32KB。其中 0x0000~0x77FF 是用户空间，共 30KB，最后一个字节 0x77FF 是安全锁字节，在引导过程会把其值映射到寄存器，以实现 FLASH 的访问控制；往上 1KB，0x7800~0x7BFF 存放的是参数表，引导过程写入参数 XRAM；最后 1KB 是保留区，软件不能 MOVX 进行访问。

● XRAM

一共有 5 块独立的 XRAM，其中低 1KB 是通用 XRAM，供软件使用；0x2800~0x2FFF 是 A RAM 和 B RAM（触摸章节有详细介绍）；0x3000~3FFF 是 MAC RAM（当 MAC 工作在 DMA 模式时，MAC_XRAM 不能被 CPU 访问）。

● IRAM

内部 RAM 大小为 256 字节，其中低 128 字节可以直接或间接寻址；高 128 字节的地址 0x80~0xFF 和 SFR 空间重叠，使用直接寻址指令将访问 SFR，间接寻址访问 IRAM 的高 128B。

● SFR

从 0x80 到 0xFF 的直接寻址存储器空间为特殊功能寄存器（SFR），地址以 0x00 或 0x08 结尾的 SFR 既可以按字节寻址也可以按位寻址，所有其它 SFR 只能按字节寻址。SFR 空间中未使用的地址保留为将来使用，访问这些地址会产生不确定的结果，应予避免。下表列出的每个寄存器的详细说明请参见本数据表的相关部分。

SFR地址映射

地址	0(8)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)
0xF8	SPICTL					DMA0A_AL	DMA0A_AH	
0xF0	B				P0SEL/LockB	RSTEN		
0xE8	EIE1					DMA0B_AL	DMA0B_AH	RSTSRC
0xE0	ACC	MAC0ACC0	MAC0ACC1	MAC0ACC2	MAC0ACC3	MAC0OVR	DMA0CI_AL	DMA0CI_AH
0xD8	EIP1					DMA0CO_AL	DMA0CO_AH	
0xD0	PSW	P0DIR	P1DIR	P2DIR	P3DIR	P0PU	P1PU	P23PU
0xC8	TMR2CN					DMA0_ITCL	DMA0_ITCH	
0xC0	IRQ0	DIV_WB0	DIV_WB1	DIV_WB2	DIV_WB3	MAC0CF0	MAC0CF1	MAC0CF2
0xB8	IP		I2C_DIN	I2C_DOUT	I2C_SLAD	I2C_STAT	SPR0	SPR1
0xB0	P3	DIV_WA0	DIV_WA1	DIV_WA2	DIV_WA3	FCTR	FLSCL	FLKEY
0xA8	IE	CKSEL	MAC0AL	MAC0AH	MAC0BL	MAC0BH	MAC_LEN	MISC
0xA0	P2	SPICFG	SPISCR	SPIDAT	PERCF		ADJ_PVB	OSC_S0
0x98	SCON	SBUF0	CRC0CN	CRC0IN	CRC0DAT	CRC0ST	CRC0CNT	MAC_STAT
0x90	P1	TMR2RL	TMR2RLH	TMR2L	TMR2H	DEBCF	DEBDLY	MAC_CTRL
0x88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCTL	PSCTL
0x80	P0	SP	DPL	DPH	TMR2CN	PUMP_CFG	WDTCTR	PCON

2.2 FLASH 程序存储器

YS62F0132 内部有可编程的 Flash 存储器，用于程序代码和非易失性数据存储。可以通过软件使用 MOVX 写指令对 FLASH 存储器进行写操作。一个 FLASH 位一旦被清 0，必须经过擦除才能再回到 1 状态。在进行重新写操作之前，一般要将 FLASH 字节擦除（置为 0xFF）。为了保证操作正确，写和擦除操作由硬件自动完成，不需要进行数据查询来判断写/擦除操作何时结束。在 FLASH 写/擦除操作期间，程序停止执行。

支持 3 种功耗模式，包括工作、待机及睡眠，其中睡眠模式允许 FLASH IP 掉电。通过 FLSCL 寄存器可配置为 22MHz、10MHz 以及 5MHz 频率的编程。

通过 LOCKBYTE，控制器还可以实现对 FLASH 内容的保护：软件读受保护的内容时，产生 FEDR 复位。

在使用 MOVX 指令对 Flash 写入之前，必须将程序写允许位 FLA_WEN（PSCTL.0）置 1，以允许写操作，它告诉了 MCU 之后的 MOVX 指令指向的是 Flash 而不是 XRAM。建议允许 Flash 写期间禁止中断，以免产生误写动作。写 Flash 可以清除数据位，但不能使数据位置 1。如果某一字节已经编程过了，要对它重写就必须先清除其所在扇区，再执行写入操作。软件读 Flash 用 MOVC 指令，MOVX 读操作总是指向 XRAM。

用软件对 Flash 编程的具体步骤如下：

- 禁止中断
- 置 SEC_EN（PSCTL.1）和 FLA_WEN（PSCTL.0）为 1，以允许软件扇区擦除（块擦除以 1KB 为单位）
- 顺序往 FLKEY 写 0xa5 和 0xf1，进入开锁状态
- 用 MOVX 指令向待擦除扇区的任一地址写任意值。擦除过程中程序自动停止，完毕后继续往下执

行，扇区擦除/块擦除耗时均约 4~5ms，擦除范围 0~0x73ff，即后 3 个块内不能擦除，否则程序复位。

- 清除 SEC_EN 禁止擦除操作
- 顺序往 FLKEY 写 0xa5 和 0xf1，重新进入开锁状态
- 用 MOVX 指令向刚擦除的扇区的目标地址写入数据，重复此步骤直到所有字节写入
- 清除 FLA_WEN 禁止 Flash 写
- 重新允许中断

注意：

- 1、擦除、写操作期间 CPU 停止执行指令，期间发生的中断也被挂起；为避免软件跑飞误对 Flash 擦除，只允许软件对块和扇区的擦除，而不允许软件对整片 Flash 擦除。
- 2、要开锁一定要先对 FLKEY 写 0xa5，后写 0xf1，写 0xa5 和 0xf1 中间允许 MCU 其它操作。“MOVX 编程 Flash”功能在没冻结的情况下，任何往 PSCTL 的写操作将使此功能重新上锁；若不按照这个写顺序或者数据不对，或者在开锁的状态下往 FLKEY 写数据，则此功能将冻结，直到下一次系统复位！

2.2.1 FLSCL 寄存器

FLSCL：FLASH 存储器定时预分频器，地址 0xB6，高 2 位可写，其保留。

位	名称	复位值	功能	
7:6	TM_CTRL	0	值	作用
			2'b10	10MHz (9M~11.2M)
			2'b01	5MHz (4.7M~5.6M)
			其它值	22MHz (20M~25M)
5:0	NA	NA	保留位，写无效。读=0x00。	

表 2.2.1 FLSCL 描述

2.2.2 PSCTL 寄存器

PSCTL：编程控制寄存器，地址 0x8F，低 3 位可写，高 5 位保留。

位	名称	复位值	功能
7:3	NA	NA	保留位，写无效。读=0x00。
2	BLK_EN	0	块擦除使能，高有效
1	SEC_EN	0	扇区擦除使能，高有效
0	FLA_WEN	0	编程使能，高有效 只有在 FLA_WEN 为高时，BLK_EN 和 SEC_EN 才起作用

表 2.2.2 PSCTL 描述

注意：

块擦除和扇区擦除是相互排斥的，扇区擦除优先级最高。假如 BLK_EN 和 SEC_EN 同时为高，编程命令执行的是扇区擦除。

2.2.3 FLKEY 寄存器

FLKEY: FLASH 编程开锁寄存器, 地址 0xB7

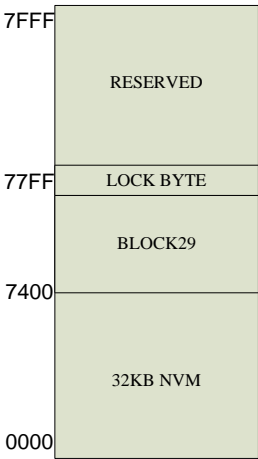
位	名称	复位值	功能
7:0	FLKEY	NA	<p>FLKEY 并没有实际的寄存器映射, 它负责解释来自软件写内容, 进而控制“软件编程 FLASH”的功能。</p> <p>写: 按先后顺序往 FLKEY 写 0xa5、0xf1 将开启“软件编程 FLASH”功能。若顺序不对或者写其他值将使此功能冻结, 直到下一次系统复位!</p> <p>读: 最低 2 位反映的是内部状态, 高 6 位返回的是 0x00: 00: 上锁 01: 0xa5 已经写入, 等待 0xf1 写入 10: 开锁 11: 冻结</p>

表 2.2.3 FLKEY 描述

2.3 FLASH 安全管理（软件加密）

位于代码区（0x0000~0x77ff）的最后一个字节是安全锁定字节, 器件擦除后对其编程可以实现对代码的保护, 其内容以反码形式存在。例如存放在 0x77ff 单元的值为 0xfe, 则其反码是 0x01, 表示 block0（0x0000~0x03ff）以及安全锁定字节所在块 block29 将被保护。

在上电过程 LockByte 值被映射到一个影子寄存器, 它决定了 Flash 的读写、擦除操作的合法性, 在执行完一次器件擦除, 或者 LockByte 的编程操作后, 这个影子寄存器将被更新, 无需重复上电过程。



FLASH 地址映射

2.3.1 特性

- 代码保护以 1KB 大小的块进行管理
- 软件的非法读写、擦除操作将产生 FEDR 复位

2.3.2 安全保护等级

访问 Flash 可能的方式有以下 4 种:

- 从加密块访问加密块

- 从加密块访问非加密块
- 从非加密块访问非加密块
- 从非加密块访问加密块

以上四种操作可能产生的影响如下表所示：

序号	行 为	用户固件执行位置	
		非加密块	加密块
1	读、写或擦除非加密块 (没有加密块的情况)	允许	允许
2	读、写或擦除加密块 (有加密块的情况)	不允许 产生FEDR复位	允许
3	读、写block29 包含LockByte (没有加密块的情况)	允许	允许
4	读、写block29 不含LockByte (有加密块的情况)	不允许 产生FEDR复位	允许
5	读取LockByte的值 (没有加密块的情况)	允许	允许
6	读取LockByte的值 (有加密块的情况)	不允许 产生FEDR复位	允许
7	擦除block29 (没有加密块的情况)	不允许 产生FEDR复位	不允许 产生FEDR复位
8	擦除block29 (有加密块的情况)	不允许 产生FEDR复位	不允许 产生FEDR复位
9	增加加密块 (改变LockByte的值)	不允许 产生FEDR复位	不允许 产生FEDR复位
10	减少加密块 (改变LockByte的值)	不允许 产生FEDR复位	不允许 产生FEDR复位
11	读、写或擦除保留区 19 (0x77ff以上的单元)	不允许 产生FEDR复位	不允许 产生FEDR复位

表 2.3.2 操作权限关系表

总结：

- 1、 LockByte 一旦被写入，涉及对其的编程、擦除操作都被禁止；
- 2、 只要某一块被加密保护， LockByte 所在的块 block29 也一齐被保护；
- 3、 不管加密情况如何，位于加密块的程序总是可以读取 LockByte 的值。

3.0 功耗模式

YS62F0132 支持四种功耗模式，分别是正常、待机、睡眠和深度睡眠。电源管理单元（PMU0）允许器件进入可用的电源方式和从某种方式唤醒。下表给出了这些工作方式的简要说明。

模式	描述	唤醒源	功耗性能
正常	除去被关掉的外设，其他模块全速工作，快时钟源可以选择关闭	NA	功耗较高，性能最好，最高达22MIPS速度
待机	CPU停止，其他功能模块关闭或工作，慢时钟打开，Flash待机，快时钟源可以选择关闭	任何中断 看门狗溢出复位 外部/软件复位	功耗低 性能灵活
睡眠	CPU停止，其他功能模块关闭或工作，快时钟关闭，慢时钟打开	外部/IO中断 看门狗溢出中断 看门狗溢出复位 触摸相关中断 I2C中断 外部/软件复位	功耗很低 性能灵活
深度睡眠	片内快慢时钟关闭	I2C中断 外部/IO中断 外部/软件复位	功耗最低

功耗模式描述

注：

- 1、在睡眠和深度睡眠模式下，若当前触摸模块在工作时，快时钟不会立即关闭，待触摸扫描结束，片内快时钟才会关闭；
- 2、慢时钟振荡器只有在深度睡眠模式下才会关闭，但若使能了硬件去抖（DEB_EN=1），慢时钟仍然会打开，但此时所有外设时钟被门控；
- 3、如不需要触摸扫描，快时钟可以在各种模式下关闭；
- 4、睡眠模式下，如果选择快时钟作为系统时钟源，Timer0/1/2 都将停止工作，所以不能使用这些中断作为唤醒源；如果选择慢时钟作为系统时钟，则以上几个中断将可作为唤醒源；
- 5、如果想要使用 UART/SPI 通信模块，必须选择快时钟为系统时钟，同时 MCU 处于正常工作或者待机模式。

3.1 正常模式

正常工作模式下所有功能模块均可处于工作状态下，外设可以通过配置相关 SFR 来禁止、开启，也可以把不用的模块时钟门控掉，节省部分功耗开销。此模式下性能最强，快时钟下指令速度可达到 22MIPS。

3.2 待机模式

此模式的“优先级”最低，即如果同时往 PCON.4/PCON.1/PCON.0 写 1，MCU 将进入深度睡眠模式。

通过置位 PCON 的最低位（IDLE 位）可使芯片进入此模式，CPU 工作时钟将被门控，其他功能模块的时钟不受此位的影响，它们可以工作或者关闭，取决于它们各自的配置 SFR。

内部振荡器不受影响，它们将保持进入待机模式之前的状态。

任何中断均可结束待机模式，中断发生后，PCON.0 由硬件清 0，CPU 从中断向量处取指令执行代码。RETI 返回后执行的是设置 PCON.0 为 1 的下一条指令。

如果进入待机之前使能了看门狗定时器并允许其中断，看门狗计数溢出后也会唤醒 CPU，这样可以避免因对 PCON.0 的错误写入而一直处于待机模式。

3.3 睡眠模式

标准的 80C51 里面，置位 PCON.1 位可使 MCU 进入一个 STOP 模式，该模式下 CPU 和所有的外设如 TIMER，UART 等功能模块都将停止工作。

3.4 深度睡眠模式

通过置位 PCON.4 可以使 MCU 进入此模式，该模式下所有数字外设的时钟均被门控；另外如果启用了外部中断的硬件去抖功能，硬件去抖模块的时钟则不受影响。

该模式可以通过外部中断、I2C 中断唤醒，唤醒后 PCON.4 被硬件清 0。

3.5 PCON 寄存器

PCON 地址为 0x87，此寄存器的第 4、第 1 和第 0 位用作功耗管理，由于唤醒后硬件自动清 0 的特性，这 3 位的读出来的值永远是 0。

位	名称	复位值	功能
7	GF4	0	通用标志位 4
6	GF3	0	通用标志位 3
5	GF2	0	通用标志位 2
4	DEEPPD	0	写 1 使芯片进入深度睡眠模式，唤醒后由硬件清 0
3	GF1	0	通用标志位 1
2	GF0	0	通用标志位 0
1	STOP	0	写 1 使芯片进入睡眠模式，唤醒后由硬件清 0
0	IDLE	0	写 1 使芯片进入待机模式，唤醒后由硬件清 0
{PCON.4,PCON.1,PCON.0}			功耗模式
1	x	x	深睡眠
0	1	x	浅睡眠
0	0	1	待机
0	0	0	正常

PCON 定义

注： 1、如果通过调试接口进行低功耗模式的切换，例如由正常模式切换到待机、睡眠或深睡眠的一种，CPU 将不能被中断唤醒，故上位机不应支持对 PCON 的写操作。

4.0 复位源

YS62F0132 有七个复位源：

- 上电复位
- 低电压检测复位
- 过度电应力复位（EOS 复位）
- FEDR 复位
- 软复位
- 看门狗溢出复位
- 外部复位

各复位事件对系统的影响

复位事件	受影响的电路
上电复位	数字电路所有模块
低电压复位	清除所有数字复位标志位
EOS复位	引发BOOT过程
外部复位	
看门狗复位	除配置寄存器（CFGxx）外所有模块 可配置是否产生BOOT过程
软复位	除断点、配置寄存器（CFGxx）、FPI、C2及EMU外所有模块
FEDR	不引发BOOT过程

复位标志可查询，记录在地址为 0xEF 的 SFR 里面，RSTSRC。最近一次的复位会把相关的位置 1，把其他各位清 0。

假设此前已有模拟复位标志位（外部、上电、EOS 或低电压）被置起，且没有被清 0，数字的复位事件（WDT、FEDR、SOFT_RST）不会把模拟复位标志位清掉。

相反，假设此前有数字复位事件发生，标志位被记录。之后发生了某次模拟复位事件，或者使能了看门狗 boot（WDT_BT_EN）并有看门狗复位事件，那么数字复位事件标志位会被清 0。

4.1 外部复位、上电复位

当 RST 管脚为低超过 20us 时，模拟侦测电路认为这是一次复位事件，把复位信号置为有效，MCU 将启动复位和 boot 过程。

4.2 低电压侦测复位

RSTEN.0 置为高时将允许 VDD 监测电路。如果电压降低到了复位阈值，监测电路发出 PWR_FAIL，此时复位产生模块将发出复位信号。

4.3 看门狗溢出复位

使能看门狗定时器后，如果在其计数溢出之前没有喂狗，计数器溢出之后将会引发系统复位。这个复位源可以避免程序跑飞或者由于错误写入 PCON 值而造成睡眠。向 PERCF.1 写高将使能看门狗模块，看门狗溢出事件记录在 IRQ0.7，如果 RSTEN.1 为 1，复位模块将产生系统复位，同时如果 RSTEN.4 为 1，溢出事件将通知模拟电路以产生 boot 过程。

看门狗复位可配置产生 boot 或者不产生 boot，配置位为 RSTEN 的第 4 位。

4.4 FEDR 复位

Flash 控制器提供了“MOVX 编程、擦除 Flash”功能，如果用该功能访问加密的扇区，或者是保留区，控制器都将发出 Flash 错误操作复位。FEDR 复位源一直使能，详情请参阅“FLASH 安全管理”一节。

4.5 软复位

软件复位是供片上调试模式使用的，当上位机往芯片写 Soft reset 命令时，经过同步电路处理后将产生系统复位。复位后 CPU 暂停运行，PC 值为 0x0000 地址。

4.6 过度电应力复位

EOS 是指芯片的电压、电流超出了芯片能承受的范围，当这种情况发生的时候而且 EOS 被使能（RSTEN.2 为 1），EOS 侦测电路会发出复位信号，启动系统复位和 boot 过程。

4.7 复位寄存器

4.7.1 RSTSRC 复位状态寄存器

RSTSRC 地址为 0xEF，页地址为 0x0，只读。通过此寄存器可以了解芯片发生了哪些复位，其各位都是高电平有效。

位	名称	复位值	功能
7	EXT_RST	NA	只读，1 表示发生过外部复位
6	EOS	NA	只读：1 表示发生了电过度应力复位
5	NA	NA	保留位，读 0
4	FEDR	NA	只读，1 表示此前有 MOVX 的非法操作，引发了复位
3	WDT_OV	NA	只读：1 表示看门狗溢出引发了复位
2	SOFT	NA	只读，1 表示发生过软件复位
1	POR	NA	只读，1 表示发生过上电复位
0	PWR_FAIL	NA	只读：1 表示曾经掉电，引发了复位

RSTSRC 定义

4.7.2 RSTEN 复位源使能寄存器

RSTEN 地址为 0xF5，页地址为 0x0。低 5 位可写，分别实现对 EOS、WDT 和 VDD_MON 复位源的使能。

任何对此寄存器的写均会把模拟复位标志清除，但数字复位标志不受影响。

位	名称	复位值	功能
7:5	Unused	NA	保留位，读出的值永远是 0 写没有意义
4	WDT_BT_EN	0	读写：1 允许看门狗复位事件引发 BOOT 0 禁止看门狗复位事件引发 BOOT
3	EOS_GATE	1	读写：1 允许 EOS 事件门控时钟； 0 禁止 EOS 事件门控时钟
2	EOS_RST_EN	1	读写：1 允许 EOS 模块发出复位，0 禁止
1	WDT_RST_EN	1	读写：1 允许看门狗模块发出复位，0 禁止
0	LVD_RST_EN	1	读写：1 允许电压检测发出复位，0 禁止

RSTEN 定义

5.0 Boot 过程

可引发 boot 过程的复位有：上电/外部复位、EOS 以及 LVR 复位，这些复位会把 boot_en 置高，送到 Flash 控制器，芯片在系统复位的同时会启动 boot 过程。

6.0 循环冗余检查单元（CRC）

YS62F0132 的 CRC 采用 8 位并行算法，与传统的 CRC 电路串行工作相比，速度可提高到原来的 8 倍。并行算法是通过分析串行算法在连续移位 8 次的过程，得出结果中每个 bit 与原始值的每个 bit 的关系，从而直接用组合逻辑来表示结果。CRC 能使用 16 位或 32 位多项式执行 CRC。CRC 接收写到 CRC0IN 寄存器的 8 位数据流，向一个内部寄存器输出 16 位或 32 位的结果。内部结果寄存器可以用 CRC0PNT 位和 CRC0DAT 寄存器间接访问。

6.1 操作步骤

6.1.1 计算单个字节的 CRC

- 向控制寄存器 CRC0CN 的 bit[1]写 0，使 CRC 模块工作在单字节计算模式下；
- 根据需要，向控制寄存器 CRC0CN 写入适当的值。例如，写入 0xFD，则 bit[7:5]由于是保留位，故不起作用；bit[0]为 1，使得 CRC 结果指针指向 16 位结果的高字节；bit[2]为 1，使得 CRC 结果初始化为 0xFFFF；bit[3]为 1 使得 CRC 结果初始化有效；

- 向输入数据寄存器 CRC0IN 写入一个数据，例如 0x63，则在下一个时钟周期内，CRC 结果马上就被计算出来，且结果为 0xBD35；
- 读取 CRC 结果：软件读取结果输出寄存器 CRC0DAT，得到高字节数据 0xBD；再读一次，得到低字节数据 0x35；合并起来就是正确的 CRC 结果。
- 如果在计算 CRC 之前，不想让初始化结果是 0x0000 或 0xFFFF，也可以先设定好 CRC 结果寄存器访问指针，然后向结果寄存器中写入自定义的初始化值。例如，先将指针定位在低字节，然后向 CRC0DAT 寄存器写入数据 0xCF，再次写入 0xD4，则 CRC 结果被初始化成 0xD4CF；向 CRC0IN 寄存器写入 0xD1，则在下一个时钟周期内，计算出 0xD1 的 CRC 值为 0x9FA5。

6.1.2 批量计算 ROM 数据 CRC

- 向控制寄存器 CRC0CN 的 bit[1]写 0，停止自动计算模式；
- 向控制寄存器 CRC0CN 的其它位(保持 bit[1]为 0)写入适当值，设置结果访问指针、结果初始化值，并使能初始化；
- 向自动控制寄存器 CRC0ST 写入适当值，设置起始地址；
- 向控制寄存器 CRC0CN 的 bit[1]写 1，保持其它位不变，会启动自动计算过程。

图 6.1.2-1 是进行 ROM 数据 CRC 计算时，访问的分区图。

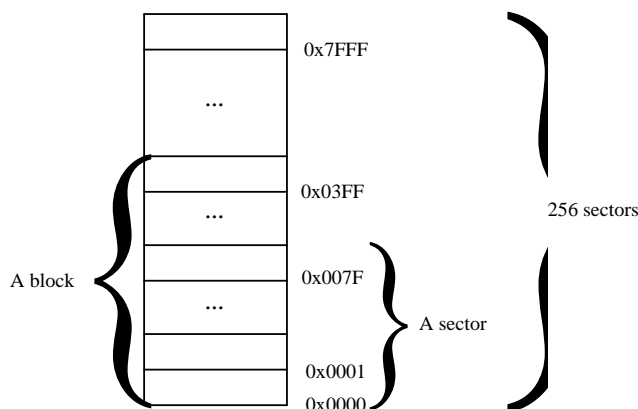


图 6.1.2-1 ROM 访问分区图

如图 6.1.2-1 所示。整个 ROM 共 32K 字节，分成 256 个 sector，编号从 sector0 到 sector255。每个 sector 包含 128 个字节。在进行 CRC 批量计算时，起始 sector 的值 CRC0ST 可以是 0x00~0xFF 之间的任何值，包括 0x00 和 0xFF；需要计算的 sector 总数的数值 CRC0CNT 可以是 0x00~0xFF，包括 0x00 和 0xFF。

需要注意的是，随着 CRC0ST 的值的增大，CRC0CNT 的值应该相应减小。例如，如果 CRC0ST 的值为 0xFF，则 CRC0CNT 的值只能是 0x00，即只能计算最后一个 sector 中数据的 CRC 值。此时，如果不小心将 CRC0CNT 的值设置为 0x01 或更大的值，则 CRC 控制器硬件会自动限制计算的字节数，使 CRC 引擎只计算最后一个 sector 中数据的 CRC 值。

6.2 CRC 寄存器

6.2.1 CRC0CN 寄存器

CRC0CN：CRC 控制寄存器，地址 0x9A

位	7	6	5	4	3	2	1	0
名称	(Reserved)			CRCDONE	CRC0INI	CRC0VAL	AUTOINT	CRC0PNT
类型	R	R	R	R	R/W	R/W	R/W	R/W
复位值	0	0	0	1	0	0	0	0

位	名称	功能
[7:5]	Reserved	保留位，只读，读为0。
[4]	CRCDONE	自动CRC计算完成标志。 在自动CRC计算模式过程中，硬件自动将这一位写0，并且软件代码也会停止执行；在其它情况下，硬件自动将这一位置为1，所以，软件读取这一位始终返回1。
[3]	CRC0INI	CRC结果初始化使能 0：初始化无效 1：初始化有效； 当软件向这一位写1时，硬件并没有真正将1写入此位，而是同步产生一个时钟周期的高电平脉冲，送到CRC引擎，作为CRC结果初始化的条件。所以，不管软件向这一位写入什么值，读取时总是返回0。
[2]	CRC0VAL	CRC结果初始化选择位。 0：将CRC结果初始化为0x0000 1：将CRC结果初始化为0xFFFF
[1]	AUTOINT	CRC自动计算使能。 当向此位写1时，会自动对Flash的某片连续的块中的数据进行CRC计算。计算的起始块为CRC0ST，共计算CRC0CNT个块。 注：在启用自动CRC计算功能之前，应先将其它位配置好，再将这一位写1。
[0]	CRC0PNT	CRC结果指针。 0：读取CRC0DATA寄存器时，访问的是16位CRC结果的低字节(7-0位) 1：读取CRC0DATA寄存器时，访问的是16位CRC结果的高字节(15-8位) 注：在应用时，如果要判断读取的是CRC结果的低字节还是高字节，应该要先读取CRC结果，再判断这一位是0还是1。如果是0，则说明刚才读取的是低字节；如果是1，则说明刚才读取的是高字节。即遵循“先读结果，再根据指针判断”的原则；而不是先看指针，再读CRC结果。 另外，在上位机单步运行调试时，由于上位机本身会读，如果软件再读，则会

		发现读出来的永远是高字节或低字节。
--	--	-------------------

注： 1、由于 CRC 计算过程分为两大类，一类是单个字节的 CRC 计算，一类是 ROM 数据批量 CRC 自动计算。向控制寄存器 CRC0CN 的 bit[1]写入 1，会立即启动 CRC 自动计算过程。如果要计算软件写入 CRC0IN 寄存器中的单个字节的 CRC 值，则 CRC0CN 寄存器的 bit[1]不能为 1。

6.2.2 CRC0IN 寄存器

CRC0IN: CRC 输入数据寄存器，地址：0x9B

位	7	6	5	4	3	2	1	0
名称	CRC0IN[7:0]							
类型	W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	CRC0IN	<p>CRC模块输入数据。</p> <p>每次向此寄存器写入一个数据时，CRC模块就自动在现有CRC结果的基础上，根据输入数据计算出新的CRC结果，并覆盖原CRC结果。</p> <p>注：此寄存器是一个虚拟寄存器，写入的数据并不保存。读取此地址时返回0x00。</p>

6.2.3 CRC0DAT 寄存器

CRC0DAT: 结果输出寄存器地址，0x9C

位	7	6	5	4	3	2	1	0
名称	CRC0DAT[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	CRC0DAT	<p>CRC结果输出。</p> <p>每次读、写此寄存器时，会根据控制寄存器CRC0CN中的结果指针CRC0PNT来决定访问的是CRC结果的高字节还是低字节。需要注意的是，每次读或写此寄存器，都会导致指针CRC0PNT变化一次。</p>

注： 1：由于此寄存器的值除了直接由软件决定以外，还可由其它信号导致发生变化，所以直接放在 CRC 模块内部，而不放在寄存器专用模块里。

6.2.4 CRC0ST 寄存器

CRC0ST：自动计算起点寄存器，地址 0x9D

位	7	6	5	4	3	2	1	0
名称	CRC0ST[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	CRC0ST	<p>自动计算CRC的ROM起始sector。</p> <p>计算的起始地址是：$CRC0ST \times \text{Sector Size}$</p> <p>例如：如果CRC0ST[7:0]的值是1，每个Sector size是128个字节，则自动CRC计算的起始地址是：$1 \times 128 = 128$，实际上是从第二个sector的第一个字节开始。</p>

6.2.5 CRC0CNT 寄存器

CRC0CNT：自动计算块数寄存器，地址 0x9E

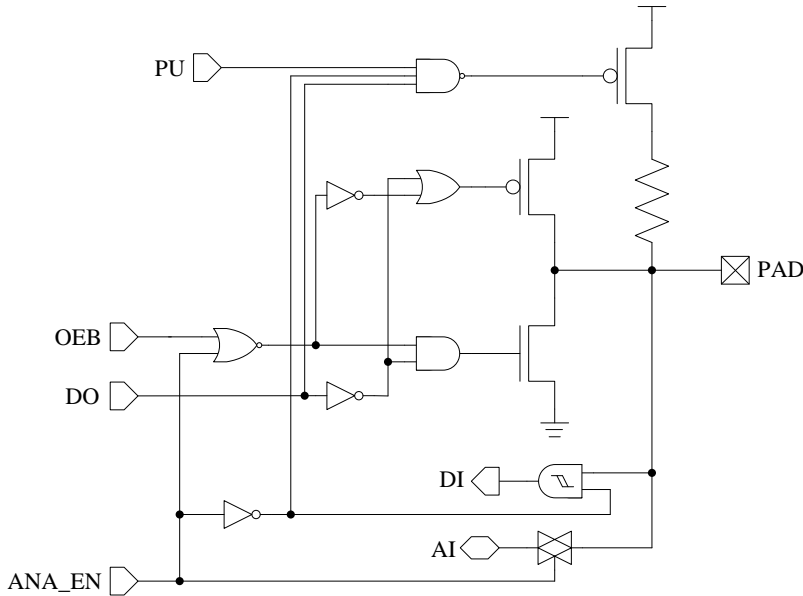
位	7	6	5	4	3	2	1	0
名称	CRC0CNT[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	CRC0CNT	<p>自动CRC计算sector总数。</p> <p>此值定义了需要计算CRC值的ROM中sector总数，例如：0x00表示1个sector；0xFF表示256个sector；需要计算CRC的最后一个sector的起始地址是：</p> <p>$(CRC0ST + CRC0CNT) \times \text{SectorSize}$</p>

注： ROM 访问分区示意图如图 6.1.2-1 所示。

7.0 输入/输出端口

YS62F0132 提供最多 26 个双向双模（数，模）I/O 端口。UART/I2C/SPI 可功能转移，与 P0[7:0]、P1[3:0] 复用。P1.4 除用作 GPIO 还可用作 Wake 口，具有可编程的硬件去抖功能。



双向IO等效电路图

7.1 对照表

地址	第7位	第6位	第5位	第4位	第3位	第2位	第1位	第0位	说明
0x80	P0[7:0]								P0数据寄存器
0x90	P1[7:0]								P1数据寄存器
0xA0	P2[7:0]								P2数据寄存器
0xB0	Reserved						P3[1:0]		P3数据寄存器
0xB5	Reserved	UARTOIR	I2CTO	I2CEN	I2CDDO	UARTFTR	SPIFTR	I2CFTR	FCTR功能控制寄存器（详见说明）

0xD1	P0DIR [7:0]			P0控制寄存器
0xD2	P1DIR [7:0]			P1控制寄存器
0xD3	P2DIR [7:0]			P2控制寄存器
0xD4	Reserved	P3DIR1	P3DIR0	P3控制寄存器
0xD5	P0PU[7:0]			P0上拉控制寄
0xD6	P1PU[7:0]			P1上拉控制寄存器
0xD7	Reserved	P3PU	P2PU	P2、P3口上拉控制寄存器

7.2 I/O 寄存器说明

7.2.1 FCTR 寄存器

FCTR：功能控制寄存器，地址：0xB5

位	7	6	5	4	3	2	1	0
名称	(Reserved)	UARTOIR	I2CTO	I2CEN	I2CDDO	UARTFTR	SPIFTR	I2CFTR
类型	0	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7]	(保留位)	只读，读为0。
[6]	UARTOIR	Uart相关寄存器，详细说明请查看14节
[5]	I2CTO	I2C相关寄存器，详细说明请查看16节
[4]	I2CEN	
[3]	I2CDDO	
[2]	UARTFTR	控制Uart功能是否转移。Uart功能默认在PAD47/P1.1和PAD49/P1.3上，功能转移以后，分别在PAD57/P0.2和PAD58/P0.3上。 0: Uart功能不转移 1: Uart功能转移
[1]	SPIFTR	控制SPI功能是否转移。SPI功能默认在PAD46/P1.0、PAD47/P1.1、PAD48/P1.2、PAD49/P1.3上，功能转移以后，分别在PAD59/P0.4、PAD60/P0.5、PAD61/P0.6、PAD62/P0.7上。
[0]	I2CFTR	控制I2C功能是否转移。I2C功能默认在PAD46/P1.0、PAD48/P1.2上，功能转移以后，分别在PAD55/P0.0和PAD56/P0.1上。

7.2.2 P0PU 寄存器

P0 端口上拉控制寄存器，地址：0xD5

位	7	6	5	4	3	2	1	0
名称	P0PU[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	P0PU	P0 上拉控制寄存器，高有效，其中每一位独立控制每一个 pin 脚。 当 P0 被配置为输入口，并且相应的上拉控制位为 1 时，上拉才有效（即 P0DIR.x & P0PU.x=1），否则内部上拉不起作用。

7.2.3 P1PU 寄存器

P1 口上拉控制寄存器地址：0xD6

位	7	6	5	4	3	2	1	0
名称	P1PU[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	P1PU	P1 上拉控制寄存器，高有效，其中每一位独立控制每一个 pin 脚。 当 P1 被配置为输入口，并且相应的上拉控制位为 1 时，上拉才有效（即 P1DIR.x & P1PU.x=1），否则内部上拉不起作用。

7.2.4 P23PU 寄存器

P2、P3 口上拉控制寄存器，地址：0xD7

位	7	6	5	4	3	2	1	0
名称	NA						P3PU	P2PU
类型	NA						R/W	R/W
复位值	NA						0	0

位	名称	功能
[7:2]	NA	保留位，写无效，读出来的值永远是 0。
[1]	P3PU	P3 上拉控制寄存器，高有效，此位控制 P3 口所有脚的上拉。 当 P3 被配置为输入口，并且 P3PU 为 1 时，上拉才有效（即 P3DIR.x & P3PU=1），否则内部上拉不起作用。
[0]	P2PU	P2 上拉控制寄存器，高有效，此位控制 P2 口所有脚的上拉。 当 P2 被配置为输入口，并且 P2PU 为 1 时，上拉才有效（即 P2DIR.x & P2PU=1），否则内部上拉不起作用。

7.2.5 P0DIR 寄存器

P0DIR 寄存器，地址：0xD1

位	7	6	5	4	3	2	1	0
名称	P0DIR7	P0DIR6	P0DIR5	P0DIR4	P0DIR3	P0DIR2	P0DIR1	P0DIR0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

位	名称	功能
[7]	P0DIR7	P0[7]数字输入、输出方向控制。 0 输出 1 输入
[6]	P0DIR6	P0[6]数字输入、输出方向控制。 0 输出 1 输入
[5]	P0DIR5	P0[5]数字输入、输出方向控制。 0 输出 1 输入
[4]	P0DIR4	P0[4]数字输入、输出方向控制。 0 输出 1 输入
[3]	P0DIR3	P0[3]数字输入、输出方向控制。 0 输出 1 输入
[2]	P0DIR2	P0[2]数字输入、输出方向控制。 0 输出 1 输入
[1]	P0DIR1	P0[1]数字输入、输出方向控制。 0 输出 1 输入
[0]	P0DIR0	P0[0]数字输入、输出方向控制。 0 输出 1 输入

7.2.6 P1DIR 寄存器

P1DIR 寄存器，地址：0xD2

位	7	6	5	4	3	2	1	0
名称	P1DIR7	P1DIR6	P1DIR5	P1DIR4	P1DIR3	P1DIR2	P1DIR1	P1DIR0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

位	名称	功能
[7]	P1DIR7	P1[7]数字输入、输出方向控制。 0 输出 1 输入
[6]	P1DIR6	P1[6]数字输入、输出方向控制。 0 输出 1 输入
[5]	P1DIR5	P1[5]数字输入、输出方向控制。 0 输出 1 输入
[4]	P1DIR4	P1[4]数字输入、输出方向控制。 0 输出 1 输入
[3]	P1DIR3	P1[3]数字输入、输出方向控制。 0 输出 1 输入
[2]	P1DIR2	P1[2]数字输入、输出方向控制。 0 输出 1 输入
[1]	P1DIR1	P1[1]数字输入、输出方向控制。 0 输出 1 输入
[0]	P1DIR0	P1[0]数字输入、输出方向控制。 0 输出 1 输入

注：如果将内部慢时钟切换到外部慢时钟，则 P1.1 始终为输入端口。

7.2.7 P2DIR 寄存器

P2DIR 寄存器，地址：0xD3

位	7	6	5	4	3	2	1	0
名称	P2DIR7	P2DIR6	P2DIR5	P2DIR4	P2DIR3	P2DIR2	P2DIR1	P2DIR0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

位	名称	功能
[7]	P2DIR7	P2[7]数字输入、输出方向控制。 0 输出 1 输入
[6]	P2DIR6	P2[6]数字输入、输出方向控制。 0 输出 1 输入
[5]	P2DIR5	P2[5]数字输入、输出方向控制。 0 输出 1 输入
[4]	P2DIR4	P2[4]数字输入、输出方向控制。 0 输出 1 输入
[3]	P2DIR3	P2[3]数字输入、输出方向控制。 0 输出 1 输入
[2]	P2DIR2	P2[2]数字输入、输出方向控制。 0 输出 1 输入
[1]	P2DIR1	P2[1]数字输入、输出方向控制。 0 输出 1 输入
[0]	P2DIR0	P2[0]数字输入、输出方向控制。 0 输出 1 输入

7.2.8 P3DIR 寄存器

P3DIR 寄存器，地址：0xD4

位	7	6	5	4	3	2	1	0
名称	(Reserved)						P3DIR1	P3DIR0
类型	只读，读为 0。						R/W	R/W
复位值	0	0	0	0	0	0	1	1

位	名称	功能
[7:2]	Reserved	保留位。
[1]	P3DIR1	P3[1]数字输入、输出方向控制。 0 输出 1 输入
[0]	P3DIR0	P3[0]数字输入、输出方向控制。 0 输出 1 输入

注：如果将内部快时钟切换到外部快时钟，则 P3.0 始终是输入状态。

7.2.9 P0 寄存器

P0 数据寄存器，地址：0x80

位	7	6	5	4	3	2	1	0
名称	P0[7:0]							
类型	R/W							
复位值	1	1	1	1	1	1	1	1

位	名称	功能
[7:0]	P0	端口 0 数据寄存器。 将端口 0 设置为通用输入端口时，此寄存器保存端口 P0[7:0]上的输入数据；将端口 0 设置为通用输出端口时，向此寄存器写入数据，则数据会从 P0[7:0]输出。

7.2.10 P1 寄存器

P1 数据寄存器，地址：0x90

位	7	6	5	4	3	2	1	0
名称	P1[7:0]							
类型	R/W							
复位值	1	1	0	0	0	0	0	0

位	名称	功能
[7:0]	P1	端口 1 数据寄存器。 将端口 1 设置为通用输入端口时，此寄存器保存端口 P1[7:0]上的输入数据；将端口 1 设置为通用输出端口时，向此寄存器写入数据，则数据会从 P1[7:0]输出。

7.2.11 P2 寄存器

P2 数据寄存器，地址：0xA0

位	7	6	5	4	3	2	1	0
名称	P2[7:0]							
类型	R/W							
复位值	1	1	1	1	1	1	1	1

位	名称	功能
[7:0]	P2	端口 2 数据寄存器。 将端口 2 设置为通用输入端口时，此寄存器保存端口 P2[7:0]上的输入数据；将端口 2 设置为通用输出端口时，向此寄存器写入数据，则数据会从 P2[7:0]输出。

7.2.12 P3 寄存器

P3 数据寄存器，地址：0xB0

位	7	6	5	4	3	2	1	0
名称	Reserved						P3[1:0]	
类型	只读，读为 0。						R/W	
复位值	0	0	0	0	0	0	1	1

位	名称	功能
[7:2]	(Reserved)	保留位，只读，读为 0。
[1:0]	P3	端口 3 数据寄存器。 将端口 3 设置为通用输入端口时，此寄存器保存端口 P3[1:0]上的输入数据；将端口 3 设置为通用输出端口时，向此寄存器写入数据，则数据会从 P3[1:0]输出。

8.0 硬件去抖（debounce）

P1.4 引脚可用作 wakeUp 功能，同时具有可编程的硬件去抖功能。当外围干扰大，容易造成误触发的情况下，使用 YS62F0132 特有的硬件去抖功能，可避免系统因外围干扰而不断被唤醒。

当工作在 32KHz 频率下，最大可过滤约 7.8ms 的抖动。

8.1 去抖功能的使用

由于去抖模块跟 CPU 有可能是异步的（当 CPU 选择快时钟的情况下），使用它需要遵守以下步骤（硬件对 DEB_EN 进行同步），否则可导致前面几个周期计数不准：

- 把 P1.4 设置为数字输入口；
- 把所需去抖计数值写入寄存器 DEBDLY；
- 写 IT0（TCON.0），选择检测上升沿或下降沿；
- 写 DEBCF，把 DEB_EN 置 1。

8.2 寄存器

8.2.1 DEBCF 寄存器

DEBCF：去抖配置寄存器，地址为 0x95，最低位可用，控制去抖模块的使能。

位	名称	复位值	功能
7:1	NA	NA	写无效，读恒为 0
0	DEB_EN	0	去抖模块的使能，高有效 为低时去抖模块时钟被门控

表 8.2.1 DEBCF 寄存器定义

8.2.2 DEBDLY 寄存器

DEBDLY，去抖模块延时寄存器，地址为 0x96。当去抖电路检测到 wake 脚有沿变化并且沿变化之后 wake 电平的维持时间大于等于 $DEBDLY * T_{CLK}$ 时，该模块发出沿有效信号。
沿有效信号大约滞后外部沿变化 $(DEBDLY + 4) * T_{CLK}$ 。

位	名称	复位值	功能
7:0	DEBDLY	0	延时配置计数值，可读写

表 8.2.2-1 DEBDLY 寄存器定义

DEBDLY 值	说明
1~255	1、当沿变化后维持电平时间大于 $(DEBDLY+1)*T_{clk}$ 时，模块发出沿有效信号； 2、当沿变化后维持电平时间小于 $DEBDLY*T_{clk}$ 时，模块认为是一次毛刺，过滤掉； 3、当沿变化后维持电平时间大于 $DEBDLY*T_{clk}$ 而小于等于 $(DEBDLY+1)*T_{clk}$ 时，模块可能认为是毛刺也可能认为是沿有效，取决于维持电平和慢时钟的相位关系。
0	1、当沿变化后维持电平时间大于 $1*T_{clk}$ 时，模块发出沿有效信号； 2、当沿变化后维持电平时间小于等于 $1*T_{clk}$ 时，模块可能认为是毛刺也可能认为是沿有效，取决于维持电平和慢时钟的相位关系。

表 8.2.2-2 DEBDLY 配置值和过滤毛刺的关系

9.0 中断系统

YS62F0132 包含一个扩展的中断系统，支持多个中断源和两个优先级。其中高优先级中断可以打断正在执行的低优先级中断，实现了中断嵌套；同级别的中断不能相互打断。

所有中断均拥有相应的标志位，当标志位为 1 时并且对应的中断位被允许以及中断总开关 EA 为 1 时，CPU 在执行完当前指令后产生一条 LCALL 跑到中断向量处取指，同时当前 PC 入栈，PSW、ACC 的值要靠 PUSH 指令压栈。每一个中断处理程序都必须以 RETI 结束。

要注意的是退出中断处理程序前必须确保中断标志位已经被清 0，否则在 CPU 执行完返回后的第一条指令后又再次进入该中断。

硬件对中断标志位的清除不尽相同，有的中断由硬件自动清 0，有的则依靠用户清 0，具体内容请看“中断源”一节。

9.1 中断响应延时

中断响应时间取决于中断发生时 CPU 的状态。中断系统在每个系统时钟周期对中断请求标志采样并对优先级译码。最快的响应时间为 7 个系统时钟周期：一个周期用于检测中断，一个周期用于执行一条指令，5 个周期用于完成对 ISR 的长调用（LCALL）。如果中断标志有效时 CPU 正在执行 RETI 指令，则需要再执行一条指令才能进入中断服务程序。因此，最长的中断响应时间（没有其它中断正被服务或新中断具有较高优先级）发生在 CPU 正在执行 RETI 指令，而下一条指令是 DIV 的情况。在这种情况下，响应时间为 19 个系统时钟周期：1 个时钟周期检测中断，5 个时钟周期执行 RETI，8 个时钟周期完成 DIV 指令，5 个时钟周期执行对 ISR 的长调用（LCALL）。如果 CPU 正在执行一个具有相同或更高优先级的中断的 ISR，则新中断要等到当前 ISR 执行完（包括 RETI 和下一条指令）才能得到服务。

9.2 中断源

一共有 14 个中断源，优先级和向量地址关系如下表：

中断源	向量地址	默认优先级	标志位	是否硬件清标志位	中断允许控制	优先级控制
复位	0x0000	最高	NA	NA	一直允许	最高
外部中断 Wake	0x0003	0	IE0(TCON.1)	是	IE.0	IP.0
Timer0 溢出	0x000b	1	TF0(TCON.5)	是	IE.1	IP.1
IO 沿中断	0x0013	2	PI(TCON.3)	是	IE.2	IP.2
Timer1 溢出	0x001b	3	TF1(TCON.7)	是	IE.3	IP.3
UART	0x0023	4	RI(SCON.0) TI(SCON.1)	否	IE.4	IP.4
Timer2 溢出	0x002b	5	IRQ0.6	否	IE.5	IP.5
SPI	0x0033	6	SPI_CNTL.7 SPI_CNTL.6 SPI_CNTL.5 SPI_CNTL.4	否	IE.6	IP.6
WDT	0x003b	7	IRQ0.7	否	EIE1.0	EIP1.0
保留	0x0043	8	NA	NA	NA	NA
MAC0	0x004b	9	MAC0STA.6	否	EIE1.2	EIP1.2
MTR	0x0053	10	IRQ0.2	否	EIE1.3	EIP1.3
CS	0x005b	11	IRQ0.3	否	EIE1.4	EIP1.4
SLP	0x0063	12	IRQ0.4	否	EIE1.5	EIP1.5
I2C	0x006b	13	IRQ0.5	否	EIE1.6	EIP1.6

注意：

- 除了外部中断、端口中断、定时器 0、定时器 1 中断标志可以由硬件清 0 外，其它 10 个中断标志位必须由软件清 0，方法是往相应标志位写 0。
- 除了 WAKE 脚中断和 P0 口中断外，软件向相关中断标志位置 1 时可引发中断（IE.7=1 和相应的中断允许情况下）。
- 配置 P0 口为中断源时，软件应写好 TCON.2 再写 P0SEL，这个顺序不能倒过来，否则有可能产生误产生多一次中断。

9.3 中断寄存器

9.3.1 IE 寄存器

总中断控制寄存器，地址 0xA8，页 0x0，高有效。

位	名称	复位值	功能
7	EA	0	中断总开关。 1：打开

			0: 禁止所有中断
6	ESPI	0	SPI 中断允许
5	ET2	0	Timer2 中断允许
4	ES0	0	UART 中断允许
3	ET1	0	Timer1 中断允许
2	EP	0	端口中断允许
1	ET0	0	Timer0 中断允许
0	EX0	0	外部 (Wake) 中断允许

9.3.2 EIE1 寄存器

扩展中断允许寄存器，地址 0xE8，页 0x0，高有效。

位	名称	复位值	功能
7	NA	NA	保留位，写无效，读 0
6	EI2C	0	I2C 中断允许
5	ESLP	0	睡眠定时器中断允许
4	ECS	0	自容中断允许
3	EMTR	0	互容中断允许
2	EMAC	0	MAC 中断允许
1	NA	NA	保留位，写无效，读 0
0	EWDT	0	看门狗中断允许

9.3.3 IP 寄存器

中断优先级控制，地址 0xB8，通过它可以改变中断的优先级关系。某一位设置为 1，表示该位对应的中断拥有高优先级，否则按默认优先级。

位	名称	复位值	功能
7	NA	NA	保留位，读=0，写忽略
6	PSPI	0	SPI 优先级
5	PT2	0	Timer2 优先级
4	PS0	0	UART 中断优先级
3	PT1	0	Timer1 中断优先级
2	PP	0	端口中断优先级
1	PT0	0	Timer0 中断优先级
0	PX0	0	外部 (Wake) 中断优先级

9.3.4 EIP1 寄存器

扩展中断优先级控制，地址 0xD8。

位	名称	复位值	功能
7	NA	NA	保留位，读=0，写忽略
6	PI2C	0	I2C 优先级
5	PSLP	0	睡眠定时器中断优先级

4	PCS	0	自容中断优先级
3	PMTR	0	互容中断优先级
2	PMAC	0	MAC 中断优先级
1	NA	NA	保留位，写无效，读 0
0	PWDT	0	看门狗中断优先级

关于中断优先级：

- 1、如果没有手动设置优先级，中断嵌套不会发生；默认的中断优先级是当多个中断来时处理的顺序，中断处理过程中它们都被视作同一级，所以就不存在中断被打断、挂起的情况。
- 2、当设置了中断优先级（IP/EIP），多个中断一齐到来时，优先级高的中断会先得到处理；高优先级的中断可以打断当前中断。

9.3.5 IRQ0 寄存器

中断标志位寄存器 0，地址 0xC0。

位	名称	复位值	功能
7	WDT_IRQ	0	WDT 中断标志。 只能由软件清 0，往此位写 0 即可
6	TM2_IRQ	0	Timer2 中断标志。 只能由软件清 0，往此位写 0 即可
5	I2C_IRQ	0	I2C 中断标志。 只能由软件清 0，往此位写 0 即可
4	SLP_IRQ	0	睡眠定时器中断标志。 只能由软件清 0，往此位写 0 即可
3	CS_IRQ	0	自容触摸中断标志。 只能由软件清 0，往此位写 0 即可
2	MTR_IRQ	0	互容触摸中断标志。 只能由软件清 0，往此位写 0 即可
1	MAC0_IRQ	0	只读 清要向 MAC0STA.6 写 0
0	NA	NA	保留位，写无效，读 0

注意： 为避免在配置过程中产生的误中断触发：在写 DEBCF 的时硬件会把 Wake 中断标志位（TCON.1）清掉，在写 P0SEL 时硬件会把 IO 沿中断标志位（TCON.3）清掉。

9.3.6 P0SEL 寄存器

P0 沿中断引脚选择寄存器，地址 0xF4。

位	名称	复位值	功能
7	P0.7_SEL	0	1：选择 P0.7 为 IO 沿中断触发源 0：禁止 P0.7 作为 IO 沿中断触发源
6	P0.6_SEL	0	1：选择 P0.6 为 IO 沿中断触发源 0：禁止 P0.6 作为 IO 沿中断触发源
5	P0.5_SEL	0	1：选择 P0.5 为 IO 沿中断触发源 0：禁止 P0.5 作为 IO 沿中断触发源

4	P0.4_SEL	0	1: 选择 P0.4 为 IO 沿中断触发源 0: 禁止 P0.4 作为 IO 沿中断触发源
3	P0.3_SEL	0	1: 选择 P0.3 为 IO 沿中断触发源 0: 禁止 P0.3 作为 IO 沿中断触发源
2	P0.2_SEL	0	1: 选择 P0.2 为 IO 沿中断触发源 0: 禁止 P0.2 作为 IO 沿中断触发源
1	P0.1_SEL	0	1: 选择 P0.1 为 IO 沿中断触发源 0: 禁止 P0.1 作为 IO 沿中断触发源
0	P0.0_SEL	0	1: 选择 P0.0 为 IO 沿中断触发源 0: 禁止 P0.0 作为 IO 沿中断触发源

9.3.7 TCON 寄存器

定时器控制寄存器，地址 0x88。

位	名称	复位值	功能
7	TF1	0	定时器 1 溢出标志 当 TIMER1 溢出时硬件自动置 1，CPU 处理中断时硬件自动清 0，也可以用软件清 0。
6	TR1	0	定时器 1 使能 1 使能 TIMER1，0 禁止 TIMER1
5	TF0	0	定时器 0 溢出标志 当 TIMER0 溢出时硬件自动置 1，CPU 处理中断时硬件自动清 0，也可以用软件清 0。
4	TR0	0	定时器 0 使能 1 使能 TIMER0，0 禁止 TIMER0
3	IE1	0	IO 沿中断标志 硬件检测到 P0 口有由 IT1 设置的沿事件时，此位被置起。CPU 处理中断时硬件自动清 0，也可以用软件清 0
2	IT1	0	IO 沿触发设置 1 选择上升沿触发，0 选择下降沿触发
1	IE0	0	引脚 wake 中断标志 硬件检测到 wake 引脚有由 IT0 设置的沿事件时，此位被置起。CPU 处理中断时硬件自动清 0，也可以用软件清 0
0	IT0	0	Wake 引脚的上升下降沿检测选择，为 1 是选择上升沿，为 0 时选择下降沿

10.0 系统时钟

YS62F0132支持内部振荡，分别是高速22MHz和慢速32KHz，如要使用外部晶振，必须使用有源晶体。上电默认为高速22MHz，在出厂时已经校准，其精度在整个温度和电压范围内为 $\pm 1\%$ 。

10.1 快慢切换

1: 由快切换到慢:

- 使能慢时钟（如果使用外部慢时钟，则需使用有源晶体，并置相关 P1.1 引脚为时钟输入）；
- 切换到慢时钟（CKSEL=0）；
- 根据需要，可关闭快时钟，以节省功耗；（如果当前使用内部快时钟，置 CK25M=0；如果使用外部快，则停止时钟输入）。

2: 由慢切换到快:

- 使能快时钟（如果使用内部快时钟，置 CK25M=1；如果使用外部快时钟，则需使用有源晶体，并置 P3.0 引脚为时钟输入）；
- 切换到快时钟（CKSEL=1）。

10.2 系统时钟寄存器

10.2.1 CKSEL 时钟选择寄存器。

时钟选择寄存器，地址为 0xA9

位	名称	复位值	功能
7:1	NA	NA	保留位，读=0，写忽略
0	CKSEL	1	系统时钟选择 1: 选择快时钟 0: 选择慢时钟

10.2.2 CKCTL (CKCON) 寄存器

CKCTL 地址为 0x8E，页地址为 0x0，其最高位实现对内部快时钟源的控制，其余位是定时器的配置。

位	名称	复位值	功能
7	CK25M	1	内部快时钟使能 1: 启用内部快时钟 0: 禁用内部快时钟，在正常或待机模式下，如果选择了内部快时钟作为系统时钟（即 CKSEL=1 并 FCK_EXT_EN=0），内部快时钟将强制开启，这些做是为了避免用户在切换系统时钟时忘记置位 CK25M 而导致的死机。

6	TGATE	0	用于 gate0 和 gate1 的辅助控制。 0: 禁用 gate 辅助功能, 完全关闭 gate0 和 gate1; 1: 打开 gate 辅助功能, gate0 和 gate1 是否有效, 还要看其它配置。
5	T2MH	0	Timer2 高字节时钟选择 选择提供给 Timer2 的计数时钟。具体来说, 是在 8 位分离计数模式下, 选择高 8 位计数器的计数时钟。 0: Timer2 高字节使用 TMR2CN 寄存器中的 T2XCLK 定义的时钟来计数; 1: Timer2 高字节使用系统时钟来计数。
4	T2ML	0	Timer2 低字节时钟选择 选择提供给 Timer2 的计数时钟。具体来说, 是在 8 位分离计数模式下, 选择低 8 位计数器的计数时钟。 0: Timer2 低字节使用 TMR2CN 寄存器中的 T2XCLK 定义的时钟来计数; 1: Timer2 低字节使用系统时钟来计数。
3	T1M	0	Timer1 时钟选择 选择 Timer1 的计数时钟源。当 C/T1 为 1 时, 则忽略此设置。 0: 计数器/定时器 1 使用分频配置位 SCA[1:0]选择的时钟计数; 1: 计数器/定时器 1 使用系统时钟计数。
2	T0M	0	Timer0 时钟选择 选择 Timer0 的计数时钟源。当 C/T0 为 1 时, 则忽略此设置。 0: 计数器/定时器 0 使用分频配置位 SCA[1:0]选择的时钟计数; 1: 计数器/定时器 0 使用系统时钟计数。
1:0	SCA[1:0]	0	Timer0/1 分频比选择。 00: 系统时钟 12 分频 (默认配置) 01: 系统时钟 4 分频 10: 系统时钟 48 分频 11: 外部时钟 8 分频

10.2.3 PERCF 寄存器

PERCF 地址为 0xA4, 页地址为 0x0。通过它可实现时钟源、除法器 DIV32、WDT 和 VDD_MON 模块的工作控制。

位	名称	复位值	功能
7	DIV_ERR	1	只读, 写没有意义 使用 32 位除法器时, 如果除数是 0, 此位被置 1
6	DIV_START	1	读 (DIV_FINISH): 1 表示 32 位除法器空闲 0 表示 32 位除法器忙 写 (DIV_START): 向此位写 1 表示让除法器开始运算, 写 0 无意义
5	FCK_EXT_EN	0	22MHz 快时钟源选择 1: 选择外部快时钟 (必须为有源晶振, 接在 P3.0) 0: 选择内部快时钟

4	SCK_EXT_EN	0	32KHz 慢时钟源选择 1: 选择外部慢时钟（必须为有源晶振，接在 P1.1） 0: 选择内部慢时钟
3:2	Unused	NA	保留位，读出的值永远是 2'b00 写没有意义
1	WDT_EN	0	读写：1 使能看门狗模块，0 禁止
0	Unused	NA	保留位，读出的值永远是 0

表 10.2.3 PERCF 定义

11.0 定时器 0/定时器 1

YS62F0132 内部有 3 个计数器/定时器：T0/T1 有两个 16 位计数器/定时器与标准 8051 中的计数器/定时器兼容，T2 是一个 16 位或两个 8 位自动重装载定时器。定时器 0 和定时器 1 几乎完全相同，有四种工作方式。

定时器0/定时器1工作方式	定时器2工作方式
13位计数器/定时器	16 位自动重装载定时器
16 位计数器/定时器	
8 位自动重装载的计数器/定时器	两个8 位自动重装载定时器
两个8 位计数器/定时器 (仅限于定时器0)	

11.1 Timer0/Timer1 工作模式

定时器/计数器 T0 可以设定为 13 位、16 位、8 位重装和两个独立 8 位计数器 4 种工作方式，这由寄存器 TMOD[1:0]两位状态设定。T0 的定时器模式/计数器模式由 TMOD[2]决定：

- 若 TMOD[2]=0，则 T0 设定为定时器模式，计数脉冲由 SCA[1:0]和 t0m 选择的时钟 Clk_sel 送来。
- 若 TMOD[2]=1，则 T0 为计数器模式，计数脉冲从 T0 输入引脚 P0.6（对于 T1 则是来自 P0.7）上送来。在每个 T0 的下降沿，计数器增加 1。

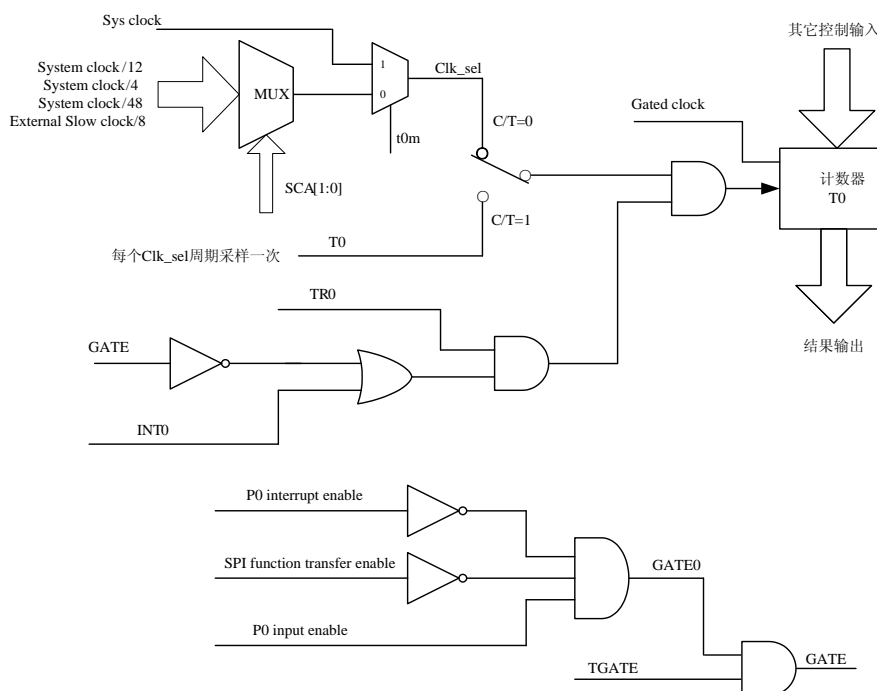


图 11.1-1 timer0 计数控制逻辑

11.1.1 方式 0

在此方式下，timer0/timer1 可按 13 位加 1 计数器工作，这 13 位由 TH 中的高 8 位和 TL 中的低 5 位组成，其中 TL 中的高 3 位是不用的。

在 timer0/timer1 启动工作前，CPU 先要为它装入方式控制字，以设定其工作方式，然后再为它装入定时器/计数器初值，并通过指令启动其工作。13 位计数器按加 1 计数器计数，当计数从 0xFF 溢出到 0x00 时自动向 CPU 发溢出中断请求，并再次计数。Timer 再次计数过程中，如果软件在溢出中断服务程序中为它重装时间常数初值，则 timer 以此重载值为起始值开始计数，否则从 0 开始计数。

在此方式下，如果读取 timer 的值，则返回的值是 timer 高 8 位与低 5 位构成的。

11.1.2 方式 1

在此方式下，timer0/timer1 是按 16 位加 1 计数器工作的，该计数器由高 8 位 TH 和低 8 位 TL 组成。定时器/计数器在方式 1 下的工作情况和方式 0 时相同，只是最大定时/计数值是方式 0 的 8 倍。

11.1.3 方式 2

在此方式下，timer0/timer1 被拆成一个 8 位寄存器 TH (TH0/TH1) 和一个 8 位计数器 TL (TL0/TL1)，CPU 对它们初始化时必须送相同的定时时间常数初值/计数器初值。当定时器/计数器启动后，TL 按 8 位加 1 计数器计数，每当它计满回零时，一方面向 CPU 发出溢出中断请求，另一方面从 TH 中重载获得时间常数初值并启动计数。

显然，定时器/计数器在方式 2 下工作时是不同于前两种方式的，定时器/计数器在方式 0 和方式 1 下计满回零时需要通过软件为它们重装定时初值/计数初值，而在方式 2 下 TL 回零能自动重载 TH 中的初值，但方式 2 下计数器长度仅有 8 位，最大计数值只有 256。

11.1.4 方式3

在前三种工作模式下，T0 和 T1 的功能是完全相同的，但在方式 3 下，T0 和 T1 功能就不相同了，而且只有 T0 才能设定方式 3。此时，TH0 和 TL0 按两个独立的 8 位计数器工作，T1 只能按不需要中断的方式 0、1 或 2 工作。如果将 T1 设置为方式 3，则相当于将 T1 禁用了，但软件仍然可以对 T1 初始化。

在方式 3 下的 TH0 和 TL0 是有区别的：TL0 可以设定为定时器或计数器模式工作，仍由 TR0 控制启动或停止，并采用 TF0 作为溢出中断标志，同时还受 GATE0、INT0 控制；TH0 只能按定时器模式工作，且不受任何 GATE 或 INT 控制，只受 TR1 控制使能或禁用，它借用 TF1 来存放溢出中断标志。因此，T1 就没有控制位可用了，故 TL1 在计满回零时是会产生溢出中断请求的。

显然，T0 设定为方式 3 实际上也就设定了 T1 的工作方式，相当于设定了 3 个 8 位计数器同时工作，其中 TH0 和 TL0 为两个由软件重装的 8 位计数器，TH1 和 TL1 为自动重装的 8 位计数器，但无溢出中断请求产生。由于 TL1 工作于无中断请求状态，故可用它来作为串行口可变波特率发生器。

注：T0 工作在方式 3 时，T1 仍然可以工作在方式 0、1 或 2，并且仍然受 CT1、GATE1、TR1、INT1 等控制。但如果用于串口可变波特率发生器，则 T1 必须配置为工作在方式 2 下，并且应该保证 GATE1 为 0（使 T1 的使能与禁用与 INT1 无关），CT1 也配置为 0（定时器模式）。

11.2 Timer0/Timer1 寄存器

11.2.1 CKCON 寄存器

CKCON 时钟控制寄存器，地址：0x8E

位	7	6	5	4	3	2	1	0
名称	CK25M	TGATE	T2MH	T2ML	T1M	T0M	SCA[1:0]	
类型	W/R	W/R	W/R	W/R	W/R	W/R	R/W	
复位值	1	0	0	0	0	0	0	0

位	名称	功能
7	CK25M	内部快时钟使能 1：启用内部快时钟 0：禁用内部快时钟，在正常或待机模式下，如果选择了内部快时钟作为系统时钟（即 CKSEL=1，并且 FCK_EXT_EN=0），内部快时钟将强制开启，这样做是为了避免用户在切换系统时钟时，忘记置位 CK25M 而导致死机现象。正确流程应该是：先开启内部快时钟，再选择内部快时钟。
[6]	TGATE	用于 gate0 和 gate1 的辅助控制。原理详见图 4.1.2.1-1。 0：禁用 gate 辅助功能，完全关闭 gate0 和 gate1； 1：打开 gate 辅助功能，gate0 和 gate1 是否有效，还要看其它配置。详见 4.1.2.1

		小节的描述。
[5]	T2MH	<p>Timer2 高字节时钟选择</p> <p>选择提供给 Timer2 的计数时钟。具体来说，是在 8 位分离计数模式下，选择高 8 位计数器的计数时钟。</p> <p>0: Timer2 高字节使用 TMR2CN 寄存器中的 T2XCLK 定义的时钟来计数；</p> <p>1: Timer2 高字节使用系统时钟来计数。</p>
[4]	T2ML	<p>Timer2 低字节时钟选择</p> <p>选择提供给 Timer2 的计数时钟。具体来说，是在 8 位分离计数模式下，选择低 8 位计数器的计数时钟。</p> <p>0: Timer2 低字节使用 TMR2CN 寄存器中的 T2XCLK 定义的时钟来计数；</p> <p>1: Timer2 低字节使用系统时钟来计数。</p>
[3]	T1M	<p>Timer1 时钟选择</p> <p>选择 Timer1 的计数时钟源。当 C/T1 为 1 时，则忽略此设置。</p> <p>0: 定时器 1 使用分频配置位 SCA[1:0]选择的时钟计数；</p> <p>1: 定时器 1 使用系统时钟计数。</p>
[2]	T0M	<p>Timer0 时钟选择</p> <p>选择 Timer0 的计数时钟源。当 C/T0 为 1 时，则忽略此设置。</p> <p>0: 定时器 0 使用分频配置位 SCA[1:0]选择的时钟计数；</p> <p>1: 定时器 0 使用系统时钟计数。</p>
[1:0]	SCA[1:0]	<p>Timer0/1 分频比选择。</p> <p>00: 系统时钟 12 分频（默认配置）</p> <p>01: 系统时钟 4 分频</p> <p>10: 系统时钟 48 分频</p> <p>11: 外部慢时钟 8 分频</p>

11.2.2 TCON 寄存器

TCON 时钟控制寄存器，地址：0x88

位	7	6	5	4	3	2	1	0
名称	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
7	TF1	<p>timer1/counter1 溢出标志。</p> <p>当 timer1/counter1 上溢（由计数最大值变为 0）时，硬件自动把这一位置 1。此标</p>

		志位可用软件来清除，但是当 CPU 进入相应的中断向量后，会自动清除此标志，所以在中断处理程序中去读取这一位时，始终返回 0。
6	TR1	timer1/counter1 运行控制。 设置为 1 时，timer1/counter1 开始运行；设置为 0 时暂停运行。 注：如果开始将 TR1 设置为 1，然后又将 TR1 设置为 0，则 Timer1 内部所有计数器原先的值保持不变。下次再将 TR1 设置为 1 时，Timer1 在原来旧值的基础上继续计数。TR0 下同。
5	TF0	timer0/counter0 溢出标志。 当 timer0/counter0 上溢（由计数最大值变为 0）时，硬件自动把这一位置 1。此标志位可用软件来清除，但是当 CPU 进入相应的中断向量后，会自动清除此标志，所以在中断处理程序中去读取这一位时，始终返回 0。
4	TR0	timer0/counter0 运行控制。 设置为 1 时，timer0/counter0 开始运行；设置为 0 时暂停运行。
3	IE1	IO 沿中断标志。当硬件检测到 P0 口有由 IT1 设置的沿事件时，此位被置起。CPU 处理中断时硬件自动清 0，也可以用软件清 0。
2	IT1	IO 沿触发设置 0：选择下降沿触发 1：选择上升沿触发
1	IE0	引脚 wake 中断标志。当硬件检测到 wake 引脚有 IT0 设置的沿事件时，此位被置起。CPU 处理中断时硬件自动清 0，也可以用软件清 0。
0	IT0	Wake 引脚的上升沿或下降沿检测选择。 0：选择下降沿 1：选择上升沿

11.2.3 TMOD 寄存器

TMOD 时钟模式寄存器，地址：0x89

位	7	6	5	4	3	2	1	0
名称	IN1PL	C/T1	T1M[1:0]		IN0PL	C/T0	T0M[1:0]	
类型	R/W	R/W	R/W		R/W	R/W	R/W	
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7]	IN1PL	int1_int 极性设置。用于决定外部引脚（P0.5）上的电平是高有效还是低有效。 0：低电平有效 1：高电平有效

[6]	C/T1	<p>counter1/timer1 选择。</p> <p>0: timer 功能: 按照 SCA[1:0]选择的时钟递增 1。</p> <p>1: counter 功能: 系统时钟 TS 对外部引脚 (P0.7) 上的电平进行检测, 当检测到下降沿时递增 1。</p> <p>注: SCA[1:0]在寄存器 CKCON[1:0]里面。</p>
[5:4]	T1M[1:0]	<p>timer1/counter1 模式选择。</p> <p>00: 模式 0, 即 13 位 counter/timer</p> <p>01: 模式 1, 即 16 位 counter/timer</p> <p>10: 模式 2, 即 8 位自动重载 counter/timer</p> <p>11: 模式 3, 即 Timer1 禁止</p>
[3]	IN0PL	<p>int0_int 极性设置。用于决定外部引脚 (P0.4) 上的电平是高有效还是低有效。</p> <p>0: 低电平有效</p> <p>1: 高电平有效</p>
[2]	C/T0	<p>counter0/timer0 选择。</p> <p>0: timer 功能: 按照 SCA[1:0]选择的时钟递增 1。</p> <p>1: counter 功能: 系统时钟对外部引脚 (P0.6) 上的电平进行检测, 当检测到下降沿时递增 1。</p> <p>注: SCA[1:0]在寄存器 CKCON[1:0]里面。</p>
[1:0]	T0M[1:0]	<p>timer0/counter0 模式选择。</p> <p>00: 模式 0, 即 13 位 counter/timer</p> <p>01: 模式 1, 即 16 位 counter/timer</p> <p>10: 模式 2, 即 8 位自动重载 counter/timer</p> <p>11: 模式 3, 即两个 8 位的 counter/timers</p>

11.2.4 TL0 寄存器

Timer0 低字节寄存器, 地址: 0x8A

位	7	6	5	4	3	2	1	0
名称	TL0[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
---	----	----

[7:0]	TL0[7:0]	16 位 Timer0 的低字节值。
-------	----------	--------------------

注：由于复位时，Timer0 和 Timer1 都是默认 13 位计算模式，所以 TL0 的高 5 位默认不可写，读为 0。如果要使高 5 位可读可写，则需要先将 Timer0 设置成其它计数模式。寄存器 TL1 同理。

11.2.5 TL1 寄存器

Timer1 低字节寄存器，地址：0x8B

位	7	6	5	4	3	2	1	0
名称	TL1[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	TL1[7:0]	16 位 Timer1 的低字节值。

11.2.6 TH0 寄存器

Timer0 高字节寄存器，地址：0x8C

位	7	6	5	4	3	2	1	0
名称	TH0[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	TH0[7:0]	16 位 Timer0 的高字节值。

11.2.7 TH1 寄存器

Timer1 高字节寄存器，地址：0x8D

位	7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---	---

名称	TH1[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	TH1[7:0]	16 位 Timer1 的高字节值。

12.0 定时器 2

定时器 2 是一个 16 位的定时器，由两个 8 位的 SFR 组成：TMR2L（低字节）和 TMR2H（高字节）。定时器 2 可以工作在 16 位自动重载方式或 8 位自动重载方式（两个 8 位定时器）。T2SPLIT 位（TMR2CN.3）定义定时器 2 的工作方式。定时器 2 还可被用于捕捉方式，以测量内部慢时钟和外部慢时钟的周期。

12.1 16 位自动重载定时器

Timer2 的 16 位自动重载模式控制原理如 图 12.1-1 所示。当 T2SPLIT（TMR2CN.3）为 0 时，Timer2 工作在 16 位自动重载模式。其计数时钟源可以是 SYSCLK、SYSCLK/12、外部振荡器的 8 分频或者内部慢时钟源。当 16 位计数器从 0xFFFF 溢出到 0x0000 时，位于重载寄存器 TMR2RLL 和 TMR2RLH 中的值自动分别载入 TMR2L 和 TMR2H 中，并且会将高字节溢出标志置 1。如果使能了 Timer2 的中断，则每当 Timer2 溢出时就会产生中断。

另外，如果使能了 Tiemr2 的中断，并且 TF2LINT（TMR2CN.5）为 1，则每当低字节（TMR2L）从 0xFF 溢出到 0x00 时，也会产生中断。

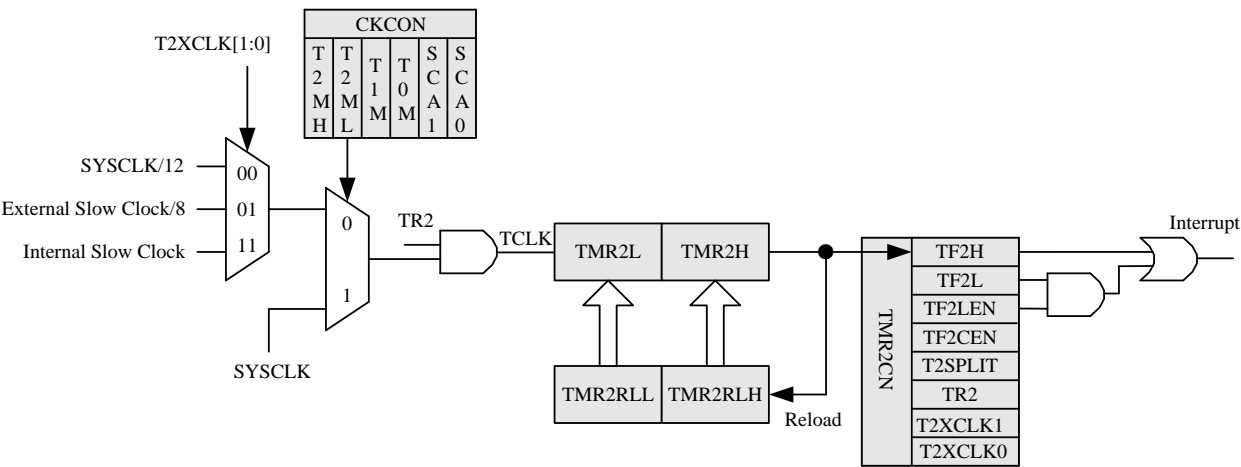


图 12.1-1 16 位自动重载模式控制示意图

12.2 8 位自动重载定时器

当 T2SPLIT 为 1 时, Timer2 被分成二个单独的 8 位计数器 TMR2H 和 TMR2L, 它们都具有自动重载功能, 如图 12.2-1 所示。TMR2CN 寄存器中的 TR3 控制 TMR2H 是否工作, 而 TMR2L 在 8 位模式下始终处于工作状态。

每个 8 位计数器都可配置成使用 SYSCLK、SYSCLK/12 或内部慢时钟源。Timer2 时钟选择位 (CKCON 寄存器中的 T2MH 和 T2ML) 用于控制选择 SYSCLK 或“其它时钟源”, “其它时钟源”由寄存器 TMR2CN 中的 timer2 外部时钟选择位按以下方式控制:

T2MH	T2XCLK[1:0]	TMR2H Clock Source
0	00	SYSCLK/12
0	01	Internal Slow Clock
0	10	Reserved
0	00	Reserved
1	X	SYSCLK

T2ML	T2XCLK[1:0]	TMR2L Clock Source
0	00	SYSCLK/12
0	01	Internal Slow Clock
0	10	Reserved
0	00	Reserved
1	X	SYSCLK

当 TMR2H 从 0xFF 溢出到 0x00 时，就将 TF2H 置 1；当 TMR2L 从 0xFF 溢出到 0x00 时，就将 TF2L 置 1。如果使能了 Timer2 的中断，则每当 TMR2H 溢出时就产生中断。如果使能了 TF2LINT，则 TMR2L 也会产生溢出中断，所以软件必须查询 TF2H 和 TF2L 这两个标志位，以便知道中断究竟是哪个计数器产生的。硬件不会自动清除这两个标志位，必须由软件来清除。

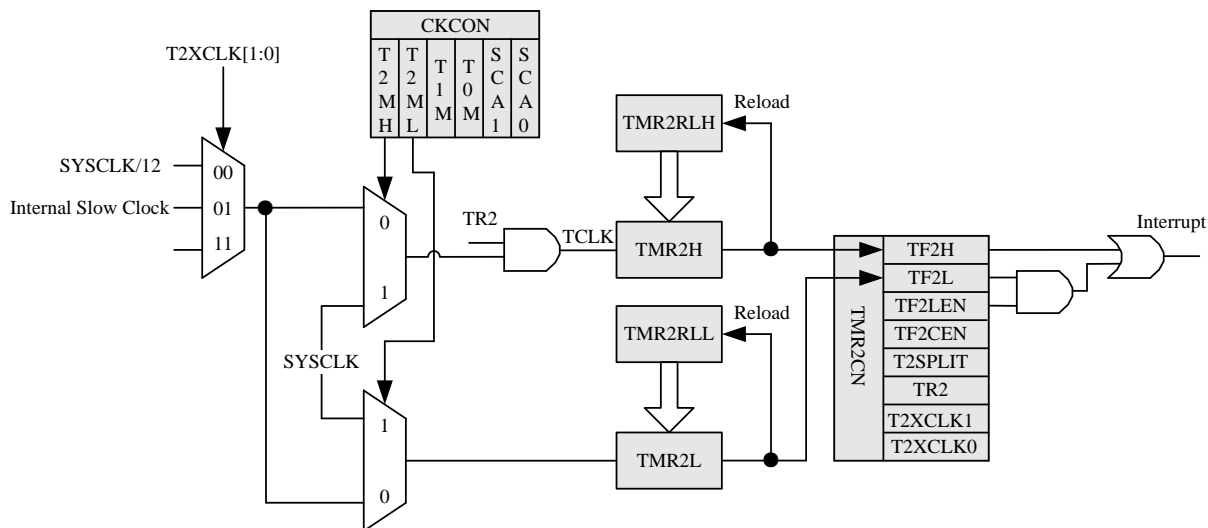


图 12.2-1 8 位自动重载模式控制示意图

12.3 捕捉方式

“抓捕”是指 timer2 在 TCLK 信号驱动下计数，在遇到 capture 信号的上升沿时，就将 16 位计数值写到重载寄存器中。如图 12.3-1 所示。此模式对测量慢时钟的频率特别有用。

在抓捕模式下，可以用系统时钟或系统时钟的 12 分频来测量内部慢时钟或外部慢时钟周期（P1.1 引脚输入）。内部慢时钟和外部慢时钟也可以相互测量。需要注意的是，被测量的信号频率必须远远小于系统时钟（Timer2 的驱动时钟）频率，这样才能保证测量精度。

启用抓捕模式，必须将 TMR2CN 寄存器中的 TF2CINT 置 1。同时 T2SPLIT 置 0，使 timer2 同时工作在 16 位计数模式下。

在启用抓捕模式以后，会在每个内部慢时钟上升沿产生一次抓捕事件。当产生抓捕事件时，timer2 的计数值（TMR2H 和 TMR2L）会保存到 timer2 重载寄存器（TMR2RLH 和 TMR2RLL）中，并且会将寄存器 TMR2CN 中的 TF2H 置 1。此时，如果使能了 timer2 的中断，则会产生一次中断。通过记录连续两次抓捕值，可以根据 timer2 的驱动时钟周期来计算出内部慢时钟的周期。

例如，如果 T2ML=1b，T2XCLK[1]=0b，TF2CINT=1b，则 timer2 会在每个 SYSCLK 上升沿递增 1，并且在每个内部慢时钟上升沿产生一次抓捕事件。如果 SYSCLK 的频率是 24.5MHz，经过记录两次连续抓捕值的差是 350，则内部慢时钟周期如下：

$$350 * (1/24.5\text{MHz}) = 14.2\mu\text{s}$$

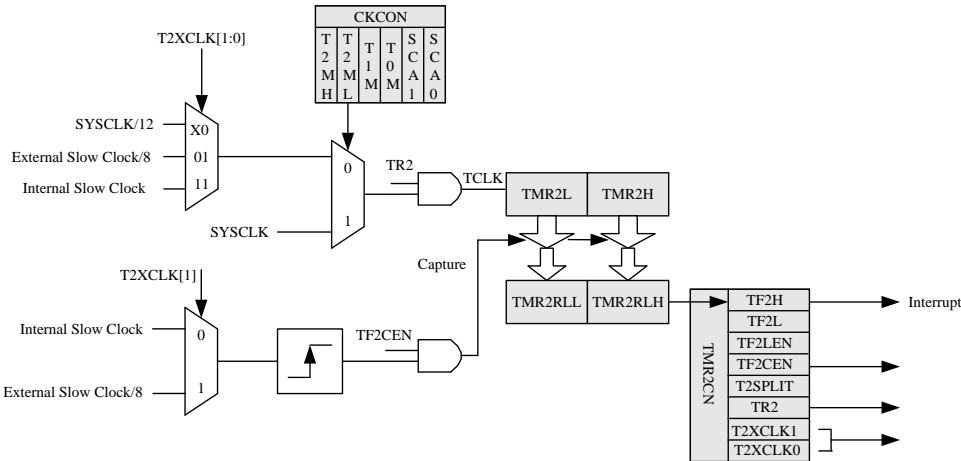


图 12.3-1 抓捕模式控制示意图

12.4 Timer2 寄存器

12.4.1 TMR2CN 寄存器

Timer2 控制寄存器, 地址: 0x84

位	7	6	5	4	3	2	1	0
名称	TF2H	TF2L	TF2LEN	TF2CEN	T2SPLIT	TR2	T2XCLK[1:0]	
类型	R	R	R/W	R/W	R/W	R/W	R/W	
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7]	TF2H	timer2 高字节溢出标志。 当 timer2 的高字节从 0xFF 溢出到 0x00 时，硬件自动置 1。在 16 位模式下，当 timer2 从 0xFFFF 溢出到 0x0000 时，硬件自动置 1。如果使能了 timer2 的中断，则在此位置 1 时，CPU 会进入中断向量。硬件不会自动清除此位。 注：抓捕模式同时也是 16 位模式，但仅在抓捕时产生中断，溢出不产生中断。 此标志位只能由硬件产生，不能由软件写入。
[6]	TF2L	timer2 低字节溢出标志。 当 timer2 的低字节从 0xFF 溢出到 0x00 时，硬件自动置 1。不管 timer2 处于哪种工作模式，只要低字节溢出时，此位都会置 1。硬件不会自动清除此位。 此标志位只能由硬件产生，不能由软件写入。
[5]	TF2LEN	timer2 低字节中断使能。 如果设置为 1，则使能 timer2 的低字节中断。如果同时使能了 timer2 的中断，则当 timer2 的低字节溢出时，就会产生 CPU 中断。

[4]	TF2CEN	timer2 抓捕使能。 0: 禁用抓捕功能 1: 开启抓捕功能 注: 如果要使用抓捕功能, 必须同时将 timer2 设置为 16 位计数模式。
[3]	T2SPLIT	timer2 分开模式使能。 0: timer2 工作在一个 16 位计数器模式。 1: timer2 工作在两个单独的 8 位计数器模式。
[2]	TR2	timer2 工作使能。 0: timer2 禁止工作 1: timer2 允许工作
[1:0]	T2XCLK[1:0]	timer2 外部时钟选择。 选择“外部”和“抓捕”时钟信号。如果 timer2 为 8 位模式, 则同时为高字节和低字节计数器选择“外部”时钟。timer2 时钟选择位 (CKCON 中的 T2MH 和 T2ML) 可以进一步为各字节选择“外部”时钟或系统时钟。 注: 外部时钟源均同步到系统时钟源。 00: 计数时钟为系统时钟 12 分频, 抓捕信号为内部慢时钟 8 分频; 01: 计数时钟为外部慢时钟 8 分频, 抓捕信号为内部慢时钟; 10: 计数时钟为系统时钟 12 分频, 抓捕信号为外部输入信号 (P1.1); 11: 计数时钟为内部慢时钟, 抓捕信号为外部输入信号 (P1.1);

12.4.2 RMR2RLL 寄存器

Timer2 重载寄存器低字节, 地址 0X91

位	7	6	5	4	3	2	1	0
名称	TMR2RLL[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	TMR2RLL[7:0]	Timer2 重载寄存器低字节。 TMR2RLL 保存 timer2 的重载值的低字节。

12.4.3 TMR2RLH 寄存器

Timer2 重载寄存器高字节，地址 0X92

位	7	6	5	4	3	2	1	0
名称	TMR2RLH[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	TMR2RLH[7:0]	Timer2 重载寄存器高字节。 TMR2RLL 保存 timer2 的重载值的高字节。

12.4.4 TMR2L 寄存器

Timer2 计数值低字节，地址 0X93

位	7	6	5	4	3	2	1	0
名称	TMR2L[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	TMR2L[7:0]	Timer2 计数值低字节。 在 16 位模式下，此寄存器保存着 16 位计数器的低字节值。在 8 位模式下，此寄存器保存着低字节计数器的值。

12.4.5 TMR2H 寄存器

Timer2 计数值高字节，地址 0X94

位	7	6	5	4	3	2	1	0
名称	TMR2H[7:0]							
类型	R/W							
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	TMR2H[7:0]	Timer2 计数值高字节。 在 16 位模式下，此寄存器保存着 16 位计数器的高字节值。在 8 位模式下，此寄存器保存着高字节计数器的值。

13.0 看门狗定时器

YS62F0132 有一个增强型看门狗定时器（WDT）。主要特性如下：

- 单独由信号同步电路和溢出产生电路组成
- 3 种工作模式
 - 中断
 - 系统复位
 - 中断与系统复位
- 从 7.8ms 到 8s 的可选择溢出周期

13.1 WDTCTR 寄存器

WDTCTR: 看门狗控制寄存器，地址 0X86

位	7	6	5	4	3	2	1	0
名称	(Reserved)				scaler			
类型	R 读为0				R/W			
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:4]	Reserved	保留位，不可写，读为0。
[3:0]	scaler	预分频器分频值： 0000 16分频，溢出时间125ms(复位后默认值；32.768K时钟驱动，下同)； 0001 不分频，溢出时间7.8ms； 0010 2分频，溢出时间15.6ms； 0011 4分频，溢出时间31ms； 0100 8分频，溢出时间63ms； 0101 32分频，溢出时间250ms； 0110 64分频，溢出时间500ms； 0111 128分频，溢出时间1s； 1000 256分频，溢出时间2s；

		1001 512分频，溢出时间4s； 1010, 1011,1100,1101,1110,1111 1024分频，溢出时间8s。
--	--	---

- 注：
- 1: 当软件向该寄存器执行写操作时，就会产生一个清零信号，将看门狗复位；
 - 2: 看门狗复位使能 `rst_en` 和看门狗使能控制 `s_en` 分别由寄存器 `RSTEN[1]` 和 `PERCF[1]` 控制；
 - 3: 看门狗溢出状态标志位在 `RSTSRC` 寄存器的 `bit[3]`；
 - 4: 进入中断处理程序后，如果不禁用 `WDT`，则其内部计数器仍然从 0 开始计数，直到溢出；软件清 `WDT` 中断标志位，会导致 `WDT` 清 0 复位；
 - 5: 如果既不允许 `WDT` 产生复位，也不允许产生中断，则其内部计数器会循环地从 0 开始计数，直到溢出；
 - 6: 在 `WDT` 正常运行过程中，在其溢出之前，如果禁用，则内部计数器清 0，并停止计数。

14.0 UART

YS62F0132 的 UART 支持 8 位数据位和 9 位数据位 2 种工作模式，后者主要是增加了奇偶校验位用于查错。UART 模块由发射子模块 TX module 和接收子模块 RX module 组成，共用一个系统时钟，但它们是两个独立的子模块，任何一个子模块均可单独工作，不影响另一个子模块。

UART 有两个相关的特殊功能寄存器：串行控制寄存器（`SCON0`）和串行数据缓冲器（`SBUF0`）。用同一个 `SBUF0` 地址可以访问发送寄存器和接收寄存器。写 `SBUF0` 时总是访问发送寄存器；读 `SBUF0` 时总是访问接收寄存器，不可能从发送寄存器中读数据。

如果 UART 中断被允许，则每次发送完成（`SCON0` 中的 `TI0` 位被置 1）或接收到一个数据字节（`SCON0` 中的 `RI0` 位被置 1）时将产生中断。当 CPU 转向中断服务程序时硬件不清除 UART 中断标志。中断标志必须用软件清除，这就允许软件判断 UART 中断的原因（发送完成或接收完成）。

14.1 波特时钟与波特率

UART 波特率由定时器 1 工作在 8 位自动重装载方式产生。TX 时钟由 TL1 产生；RX 时钟是由与 TL1 一样的时钟产生器（图 14.1-1 中以 RX Timer 表示）生成的，但其计数值对用户是不可见的。TX 和 RX 定时器的溢出被 2 分频，用于生成 TX 和 RX 的波特时钟。在使能 UART 的接收数据功能时，RX 定时器就开始运行，并与 TX 使用同一个重载值。

当检测到 RX pin 上有 START 条件时，RX 定时器被迫重载，然后开始工作。也就是说，不管 TX 的 Timer 处于何种状态，RX 只要检测到 START，RX Timer 就自动工作，保证 Uart 可以开始接收数据。需要注意的是，Uart 接收时，必须要使能 Timer1 才行。

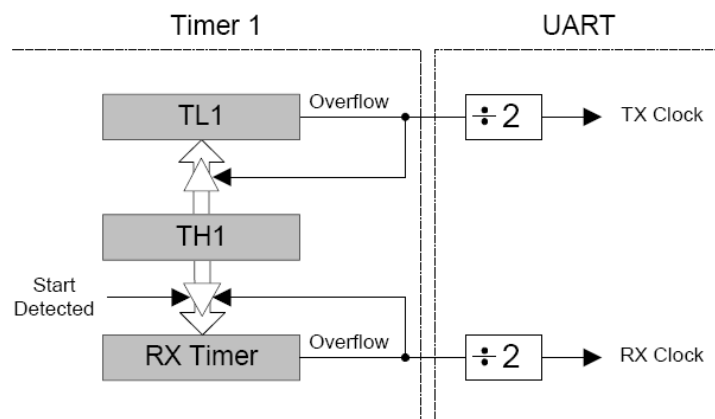


图 14.1-1 波特时钟生成图示意图

Timer1 要配置成模式 2，即 8 位自动重载。Timer1 的重载值应该这样设置：Timer1 的上溢频率是 UART 波特率频率的 2 倍。由于系统时钟源只有内部的 22M 和 32K，故 Timer1 的驱动时钟源只能是 22M，所以 UART0 的波特率由以下二个公式（公式 14.1）决定：

$$\text{A) UART 波特率} = \text{T1 溢出率} / 2$$

$$\text{B) T1 溢出率} = \text{T1}_{\text{CLK}} / (256 - \text{TH1})$$

其中 T1CLK 是定时器 1 的时钟频率，由 22M 时钟分频（分频比可在时钟控制寄存器 CKCON 中配置）得到，TH1 是 Timer1 的重载值高字节。

14.2 工作模式

UART 提供标准的异步、全双工通信，其工作模式（8 位或 9 位）通过 SOMODE 位（SCON0.7）来选择。典型的 UART 连接方式如图 14.2.1 所示。

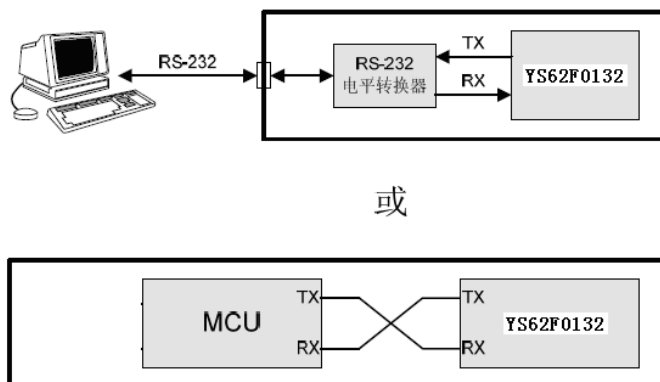


图14.2.1 UART 连接图

14.3 UART 寄存器

14.3.1 8 位 UART

在 8 位 UART 模式下，传送每个字节数据共需要 10 位：1 个起始位，8 个数据位（先传低位）以及 1 个停止位。数据按照从低位到高位顺序，从 TXD 引脚发送出去，并从 RXD 引脚接收。在接收数据时，8 个数据位存

入 SBUF0 寄存器，停止位进入 SCON0 寄存器的第 2 位 (RB80)。

当软件向 SBUF0 寄存器写入一个字节数据时，就开始发送数据。在发送结束（开始发送停止位）时，将发送中断标志位 TI0 (SCON0.1) 置 1。

任何时候当接收使能位 RINT0 (SCON0.4) 设置为 1 时，即开始接收数据。

注： 此时只是开始对接收信号进行检测，只有当接收信号电平和时序满足 UART 协议要求以后，才真正开始接收数据。

在接收完停止位以后，如果满足以下条件：

- RI0 为逻辑 0；
- 如果 MCE0 为逻辑 1，则停止位必须为逻辑 1。

则数据字节装入 SBUF0 接收寄存器。如果接收到的数据溢出，则先收到的 8 位锁存到 SBUF0 接收寄存器，而后来数据则丢失。

如果满足这些条件，则 8 位数据存入 SBUF0，停止位存入 SCON0 寄存器的第 2 位 (RB80)，SCON0 寄存器的第 0 位 RI0 置 1。如果这些条件不满足，则相应的数据不会存入 SBUF0 和 RB80，并且 RI0 标志位也不会置 1。如果使能了中断，并且 TI0 或 RI0 为 1，则会产生中断。

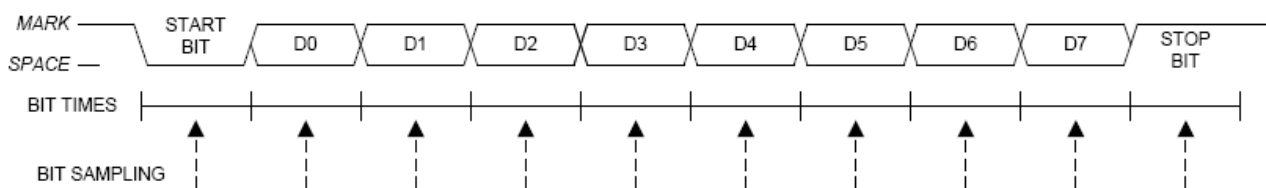


图 14.3.1-1 8 位 UART 时序图

14.3.2 9 位 UART

在 9 位 UART 模式下，传送每个字节需要 11 位：1 个起始位，8 个数据位（先传最低位），1 个可编程的第 9 个数据位，以及 1 个停止位。

在发送状态下，第 9 个发送数据位的值由 TB80 (SCON0.3) 决定。在接收状态下，第 9 个数据位进入 RB80 (SCON0.2)，而停止位被忽略。

当指令向 SBUF0 寄存器写入数据字节时，就开始了发送数据操作。发完数据（开始发送停止位时）后，发送中断位 TI0 (SCON0.1) 置 1。一旦将 UART 的接收使能位 RINT0 (SCON0.4) 置 1 时，就开始数据接收操作。同样地，此时只是开始对接收信号进行检测，只有当接收信号电平和时序满足 UART 协议要求以后，才真正开始接收数据。

在接收完停止位以后：

- 接收中断标志位 RI0 (SCON0.0) 为逻辑 0；
- 如果 MCE0 位 (SCON0.5) 为逻辑 1，则第 9 位必须为逻辑 1；如果 MCE0 位为逻辑 0，则第 9 个数据位无所谓。

如果满足以上条件，则 8 位数据字节存入 SBUF0 接收寄存器，第 9 位存入 RB80 位 (SCON0.2)，并且 RI0 标志置 1。如果以上条件不满足，则相应的数据不会存入 SBUF0 寄存器和 RB80 位，并且 RI0 标志位也不会置 1。当中断使能时，如果 TI0 位或者 RI0 位为 1，就会产生中断。

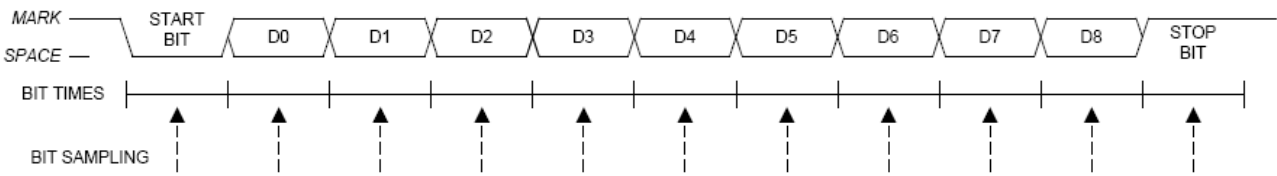


图 14.3.2-1 9 位 UART 时序图

14.3.3 多处理器通讯

9 位 UART 模式通过对第九位数据的特殊应用，支持在一个主机与一个或多个从机之间进行多处理器通讯。在主机向从机发数据之前，首先发送一个地址来选择目标从机。地址帧与数据帧通过第九位数据来区别：当第九位数据为 1 时为地址帧，而数据帧的第九位数据永远为 0。

将从机处理器的 MCE0 位(SCON0.5)置 1，则从机 Uart 接收到停止位时，如果已收到的第九位数据是 1(RB80=1)，则会产生中断。因为这意味着当前是一个地址帧，在 Uart 的中断处理程序里，软件会将接收到的地址与自己的 8 位地址进行比较。如果地址匹配，则从机将 MCE0 位清 0，以便在后续收到数据帧时能产生中断。

如果主机在后续传输过程中改变了地址（保证每个字节传输完整），并且原先被寻址的从机的 MCE0 位仍然为 0，则在改变地址后，新发送的数据（不管是地址帧还是数据帧）会同时被原先寻址的从机和新寻址的从机接收。所不同的是，新发送的数据被原先寻址的从机统一当成数据帧处理，并忽略第九位数据；而新寻址的从机会区别对待地址帧和数据帧。

没有被寻址的从机则保持自己的 MCE0 为 1 状态，并且在收到后续的数据帧时不产生中断，也就是忽略数据帧。被寻址的从机在接收完所有消息以后，会将 MCE0 清 0，从而忽略所有的后续传输，直到下一次收到寻址帧。

接收数据产生中断的条件，具体请参考表 14.3.3-1。

单个从机可以分配多个地址，单个地址也可以分配给多个从机。后者可以用广播的形式同时向多个从机发信息。可以将多处理配置成接收所有的消息传输，或者使用某种协议，将主机与从机的角色临时转换过来，以便在原始的主机与从机之间进行半双工传输。

表 14.3.3-1 接收数据产生中断的条件

工作模式	MCE0值 (是否使能多处理器通讯功能)	RB80值 (接收到的第9位数据值)
9位模式	1 (是)	1
	0 (是)	0或1
8位模式	0或1	0或1

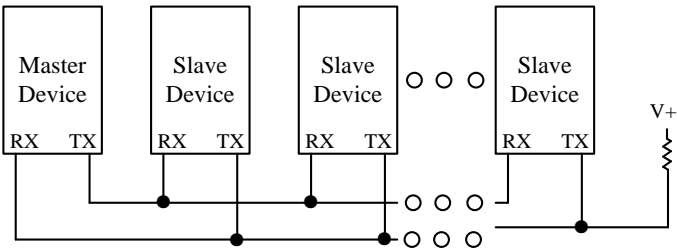


图 14.3.3-2 UART 多处理器模式通讯图

14.3.4 FCTR 寄存器

FCTR: 功能控制寄存器, 地址: 0xB5

位	7	6	5	4	3	2	1	0
名称	(Reserved)	UARTOIR	I2CTO	I2CEN	I2CDDO	UARTFTR	SPIFTR	I2CFTR
类型	0	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7]	(保留位)	只读, 读为0。
[6]	UARTOIR	用于控制Uart在发送完一帧数据后, 是否将数据发送端口(TX)自动设置为输入类型。此功能 是用于多处理器通讯的情况, 如果某个Uart从机的TX始终发送高电平, 则其它从机无法发送 数据。 0: 不自动转换端口类型 1: 自动转换端口类型。 注: 这一位实际上是由Uart模块使用, I/O控制模块并不直接使用这一位。
[5]	I2CTO	I2C功能章节介绍
[4]	I2CEN	
[3]	I2CDDO	
[2]	UARTFTR	控制Uart功能是否转移。Uart功能默认在P1.1和P1.3上, 功能转移以后, 分别在P0.2和P0.3上。 0: Uart功能不转移 1: Uart功能转移
[1]	SPIFTR	SPI功能章节介绍
[0]	I2CFTR	I2C功能章节介绍

14.3.5 SCON0 寄存器

UART 控制寄存器，可位寻址的特殊寄存器，地址 0X98。

位	7	6	5	4	3	2	1	0
名称	S0MODE	UART_EN	MCE0	RINT0	TB80	RB80	TI0	RI0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7]	S0MODE	UART的操作模式选择。 0: 8位UART模式 1: 9位UART模式
[6]	UART_EN	UART模块使能。 0: 发送功能和接收功能均禁用。 1: 发送功能启用，要启用接收功能，还必须将SCON0[4]（RINT0）写1。
[5]	MCE0	多处理器通讯使能。 对于8位UART模式：用于检查合法的停止位。 0: 忽略停止位 1: 当停止位为1时，RI0有效 对于9位UART模式：多处理器通讯使能 0: 忽略第九位数据 1: 当第九位数据为1时，将RI0置1，并产生中断。
[4]	RINT0	接收使能。 0: UART0不能接收数据。 1: UART0可以接收数据。 注：没有寄存器对发送使能进行控制。因为发送使能要结合发送数据进行同步控制。当要发送的数据准备好以后，才给出发送使能。
[3]	TB80	第9个数据发送位的值。此值在9位UART模式下，自动传送到第9个数据位上向外发送。在8位UART模式下，这一位没有用处。
[2]	RB80	第9个数据接收位的值。在9位UART模式下，这一位存储第9个接收数据位的值。在8位UART模式下，此位置1。
[1]	TI0	发送中断标志。 在发送完一个字节的的数据以后，此位由硬件自动置1。在开始发送停止位时就被视为发送完毕。如果使能了串口中断，则当这一位置1时，会导致CPU进入中断向量。进入中断服务程序后，这一位必须由软件来清除。
[0]	RI0	接收中断标志。

		在接收完一个字节的的数据以后，此位由硬件自动置1。在对停止位采样时就视为接收完了一个字节的数据。如果使能了串口中断，则当这一位置1时，会导致CPU进入中断向量。进入中断服务程序后，这一位必须由软件来清除。
--	--	--

14.3.6 SBUF0 寄存器

数据缓存寄存器，此寄存器是一个特殊功能寄存器，地址 0X99。

位	7	6	5	4	3	2	1	0
名称	SBUF0[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7:0]	SBUF0	串口数据缓冲位7:0（MSB-LSB） 通过此寄存器可以访问2个寄存器：1个发送移位寄存器和1个接收锁存寄存器。 当向SBUF0写入数据时，数据进入发送移位寄存器用于串行发送。向SBUF0写入一个字节会初始化发送操作。读取SBUF0会返回接收锁存器的值。

15.0 增强型串行外设接口（SPI）

YS62F0132 的 SPI 可以作为主器件或从器件工作，可以使用 3 线或 4 线方式，并可在同一 SPI 总线上支持多个主器件和从器件。从选择信号（NSS）可被配置为输入以选择工作在从方式的 SPI_{In}，或在多主环境中禁止主方式操作，以避免两个以上主器件试图同时进行数据传输时发生 SPI 总线冲突。NSS 可以被配置为片选输出（在主方式），或在 3 线操作时被禁止。在主方式，可以用其他通用端口 I/O 引脚选择多个从器件。

15.1 信号说明

SPI 接口有 4 个信号：MOSI/P1.2、MISO/P1.3、SCK/P1.1、NSS/P1.0，或 MOSI/P0.6、MISO/P0.7、SCK/P0.5、NSS/P0.4，2 组只能选择其中一组使用。

15.1.1 主输出、从输入（MOSI）

主出从入（MOSI）信号是主器件的输出和从器件的输入，用于从主器件到从器件的串行数据传输。当 SPI 作为主器件时，该信号是输出；当 SPI 作为从器件时，该信号是输入。数据传输时最高位在先。当被配置为主器件时，MOSI 由移位寄存器的 MSB 驱动。

15.1.2 主输入、从输出（MISO）

主入从出（MISO）信号是从器件的输出和主器件的输入，用于从从器件到主器件的串行数据传输。当 SPI 作为主器件时，该信号是输入；当 SPI 作为从器件时，该信号是输出。数据传输时最高位在先。当 SPI 被禁止或工作在 4 线从方式而未被选中时，MISO 引脚被置于高阻态。当作为从器件工作在 3 线方式时，MISO 总是由移位寄存器的 MSB 驱动。

15.1.3 串行时钟（SCK）

串行时钟（SCK）信号是主器件的输出和从器件的输入，用于同步主器件和从器件之间在 MOSI 和 MISO 线上的串行数据传输。当 SPI 作为主器件时产生该信号。在 4 线从方式，当从器件未被选中时（NSS=1），SCK 信号被忽略。

15.1.4 从选择（NSS）

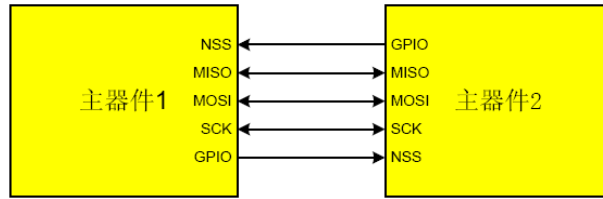
从选择（NSS）信号的功能取决于 SPICN 寄存器中 NSSMD1 和 NSSMD0 位的设置。

有 3 种可能的方式：

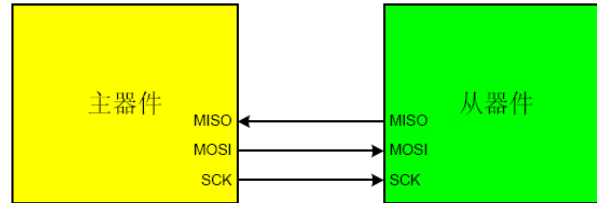
- NSSMD[1:0] = 00：3 线主方式或从方式：SPI 工作在 3 线方式，NSS 被禁止。当作为从器件工作在 3 线方式时，SPI 总是被选择。由于没有选择信号，SPI 工作在 3 线方式时必须是在总线唯一的从器件。这种情况用于一个主器件和一个从器件之间点对点通信。
- NSSMD[1:0] = 01：4 线从方式或多主方式：SPI 工作在 4 线方式，NSS 被使能为输入。当作为从器件时，NSS 选择 SPI_{In} 器件。当作为主器件时，NSS 信号的负跳变禁止 SPI 的主器件功能，以便可以在同一个 SPI 总线上使用多个主器件。
- NSSMD[1:0] = 1x：4 线主方式：SPI 工作在 4 线方式，NSS 被使能为输出。NSSMD 的设置值决定 NSS 引脚的输出逻辑电平。这种配置只能在 SPI 作为主器件时使用。

注意： NSSMD 位的设置影响器件的引脚分配。有关通用端口 I/O 和交叉开关的信息见“端口输入/输出”。

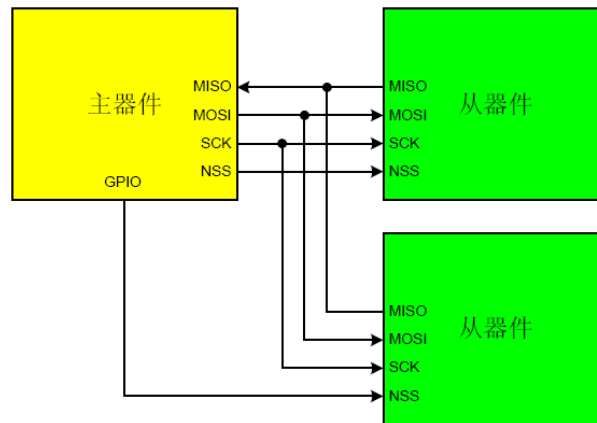
不同工作方式下的典型连接图：



多主方式连接图



3 线单主方式和3 线单从方式连接图



4 线单主方式和 4 线从方式连接图

15.2 SPI 主方式

SPI 总线上的所有数据传输都由 SPI 主器件启动。通过将主允许标志 (MSTEN, SPI_CFG.6) 置 1 将 SPI 置于主方式。当处于主方式时, 向 SPI 数据寄存器 (SPI_DAT) 写入一个数据字节时是写发送缓冲器。如果 SPI 移位寄存器为空, 发送缓冲器中的数据字节被传送到移位寄存器, 数据传输开始。SPI 主器件立即在 MOSI 线上串行移出数据, 同时在 SCK 上提供串行时钟。在传输结束后 SPIF (SPI_CNTL.7) 标志被置为逻辑 1。如果中断被允许, 在 SPIF 标志置位时将产生一个中断请求。在全双工操作中, 当 SPI 主器件在 MOSI 线向从器件发送数据时, 被寻址的 SPI 从器件可以同时在 MISO 线上向主器件发送其移位寄存器中的内容。因此, SPIF 标志既作为发送完成标志又作为接收数据准备好标志。从从器件接收的数据字节以 MSB 在前的形式传送到主器件的移位寄存器。当一个数据字节被完全移入移位寄存器时, 便被传送到接收缓冲器, 处理器通过读 SPI_DAT 来读该缓冲器。

当被配置为主器件时, SPI 可以工作在下面的三种方式之一: 多主方式、3 线单主方式或 4 线单主方式。当 NSSMD1 (SPI_CNTL.3) = 0 且 NSSMD0 (SPI_CNTL.2) = 1 时, 是默认的多主方式。在该方式, NSS 是器件的输入, 用于禁止主 SPI, 以允许另一主器件访问总线。在该方式, 当 NSS 被拉为低电平时, MSTEN (SPI_CNTL.6) 和 SPIEN (SPI_CNTL.0) 位被清 0, 以禁止 SPI 主器件, 且方式错误标志 (MODF, SPI_CNTL.5) 被置 1。如果中断被允许, 将产生方式错误中断。在这种情况下, 必须用软件重新使能 SPIIn。在多主系统中, 当器件不作为系统主器件使用时, 一般被默认为从器件。在多主方式, 可以用通用 I/O 引脚对从器件单独寻址 (如果需要)。图 24.2 给出了两个主器件在多主方式下的连接图。

当 NSSMD1 (SPI_CNTL.3) =0 且 NSSMD0 (SPI_CNTL.2) =0 时, SPI 工作在 3 线单主方式。在该方式, NSS 未被使用, 也不被交叉开关映射到外部端口引脚。在该方式, 应使用通用 I/O 引脚选择要寻址的任何从器件。图 24.3 给出了一个 3 线主方式主器件和一个从器件的连接图。

当 NSSMD1 (SPI_CNTL.3) =1 时, SPI 工作在 4 线单主方式。在该方式, NSS 被配置为输出引脚, 可被用作从选择信号去选中一个 SPI 器件。在该方式, NSS 的输出值由 NSSMD0 (SPI_CNTL.2) 控制 (用软件)。可以用通用 I/O 引脚寻址另外的从器件。图 24.4 给出了一个 4 线主方式主器件和两个从器件的连接图。

15.3 SPI 从方式

当 SPI_{In} 被使能而未被配置为主器件时, 它将作为 SPI 从器件工作。作为从器件, 由主器件控制串行时钟 (SCK), 从 MOSI 移入数据, 从 MISO 引脚移出数据。SPI 逻辑中的位计数器对 SCK 边沿计数。当 8 位数据经过移位寄存器后, SPIF 标志被置为逻辑 1, 接收到的字节被复制到接收缓冲器。通过读 SPI_{In}DAT 来读取接收缓冲器中的数据。从器件不能启动数据传送。通过写 SPIDAT 来预装要发送给主器件的数据到移位寄存器。写往 SPI_{DAT} 的数据是双缓冲的, 首先被放在发送缓冲器。如果移位寄存器为空, 发送缓冲器中的数据会立即被传送到移位寄存器。当移位寄存器中已经有数据时, SPI 将在下一次 (或当前) SPI 传输的最后一个 SCK 边沿过去后再将发送缓冲器的内容装入移位寄存器。

当被配置为从器件时, SPI 可以工作 4 线或 3 线方式。当 NSSMD1 (SPI_CNTL.3) =0 且 NSSMD0 (SPI_CNTL.2) =1 时, 是默认的 4 线从方式。在 4 线方式, NSS 被分配到一个端口引脚并被配置为数字输入。当 NSS 为逻辑 0 时, SPI 被使能; 当 NSS 为逻辑 1 时, SPI 被禁止。在 NSS 的下降沿, 位计数器被复位。注意, 对应每次字节传输, 在第一个有效 SCK 边沿到来之前, NSS 信号必须被驱动到低电平至少两个系统时钟周期。图 24.4 给出了两个 4 线方式从器件和一个主器件的连接图。

当 NSSMD1 (SPI_CNTL.3) =0 且 NSSMD0 (SPI_CNTL.2) =0 时, SPI 工作在 3 线从方式。在该方式, NSS 未被使用, 也不被交叉开关映射到外部端口引脚。由于在 3 线从方式无法唯一地寻址从器件, 所以 SPI 必须是总线上唯一的从器件。需要注意的是, 在 3 线从方式, 没有外部手段对位计数器复位以判断是否收到一个完整的字节。只能通过用 SPIEN 位禁止并重新使能 SPI 来复位位计数器。图 24.3 给出了一个 3 线从器件和一个主器件的连接图。

15.4 SPI 中断源

如果 SPI 中断被允许, 在下述 4 个标志位被置 1 时将产生中断。

注意: 这 4 个标志位都必须用软件清 0。

- 在每次字节传输结束时, SPI 中断标志 SPIF (SPI_CNTL.7) 被置 1。该标志适用于所有 SPI 方式。
- 如果在发送缓冲器中的数据尚未被传送到 SPI 移位寄存器时写 SPI_{DAT}, 写冲突标志 WCOL (SPI_CNTL.6) 被置 1。发生这种情况时, 写 SPIDAT 的操作被忽略, 不会对发送缓冲器写入。该标志适用于所有 SPI 方式。
- 当 SPI 被配置为工作于多主方式的主器件而 NSS 被拉为低电平时, 方式错误标志 MODF (SPI_CNTL.5) 被置 1。当发生方式错误时, SPI_CNTL 中的 MSTEN 和 SPIE 位被清 0, 以禁止 SPI 并允许另一个主器件访问总线。
- 当 SPI 被配置为从器件并且一次传输结束, 而接收缓冲器中还保持着上一次传输的数据未被读取时, 接收溢出标志 RXOVRN (SPI_CNTL.4) 被置 1。新接收的字节将不被传送到接收缓冲器, 允许前面接收的字节被读取。引起溢出的数据字节丢失。

15.5 SPI 寄存器描述

15.5.1 FCTR 寄存器

FCTR: 功能控制寄存器, 地址: 0xB5

位	7	6	5	4	3	2	1	0
名称	(Reserved)	UARTOIR	I2CTO	I2CEN	I2CDDO	UARTFTR	SPIFTR	I2CFTR
类型	0	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7]	(保留位)	只读, 读为0。
[6]	UARTOIR	UART功能章节介绍
[5]	I2CTO	I2C功能章节介绍
[4]	I2CEN	
[3]	I2CDDO	
[2]	UARTFTR	UART功能章节介绍
[1]	SPIFTR	控制SPI功能是否转移。SPI功能默认在P1.0、P1.1、P1.2、P1.3上, 功能转移以后, 分别在P0.4、P0.5、P0.6、P0.7上。
[0]	I2CFTR	I2C功能章节介绍

15.5.2 SPICFG 寄存器

SPI 配置寄存器, 地址 0xA1

位	7	6	5	4	3	2	1	0
名称	SPIBSY	MSTINT	CKPHA	CKPOL	SLVSEL	NSSIN	SRMT	RXBMT
类型	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	1	1	1

位	名称	功能
7	SPIBSY	SPI 工作中 此位在 SPI 工作时, 会设置为 1
6	MSTINT	主机模式使能位 0: 不使能主机模式, 工作于从机模式

		1: 使能主机模式, 工作于主机模式
5	CHPHA	SPI 时钟相位选择 0: 数据采样点位于 SCK 周期的第一个时钟延 1: 数据采样点位于 SCK 周期的第二个时钟延
4	CKPOL	SPI 时钟极性选择 0: sck 线在不工作时处于低电平 1: sck 线在不工作时处于高电平
3	SLVSEL	从机选择标志位 此位设为 1, 当 NSS pin 输入为低时, 表示从机是 SPI 主机选择的通讯对象。此位当 NSS 是高 (从机没有被选择) 时, 会被清为 0。
2	NSSIN	NSS 时时数据输入
1	SRMT	移位寄存器空标志(只在从机模式时有效)
0	RXBMT	接收暂存器空标志(只在从机模式时有效)

15.5.3 SPICTL 寄存器

SPI 控制寄存器, 地址 0XF8

位	7	6	5	4	3	2	1	0
名称	SPIF	WCOL	MODF	RXOVRN	NSSMID[1:0]		TXBMT	SPIEN
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	1	1	0

位	名称	功能
7	SPIF	SPI 中断标志位 当每次传输完一个数据 (8bit) 之后, 这位将由硬件拉高。此位必须由软件清 0
6	WCOL	写冲突标志位 当 TXBMT 为 0 时, 写入 SPIDAT 则将此位拉高, 表明写冲突。 此位必须由软件清 0
5	MODF	模式错误标志位 当检测到主机模式冲突的时候将此位置为 1T (NSS is low, MSTINT = 1 and NSSMD[1:0]=01)。 此位必须由软件清 0
4	RXOVRN	接收 overrun 标志(只在从机模式下有效)
3:2	NSSMD	从机选择模式 00: 3 线从方式或 3 线主方式。NSS 信号不连到端口引脚。 01: 4 线从方式或多主方式 (默认值)。NSS 是器件的输入。 1x: 4 线单主方式。NSS 信号被分配一个输出引脚并输出 NSSMD0 的值
1	TXBMT	发送暂存器空标志 当新数据被写入发送缓冲器时, 该位被清 0。当发送缓冲器中的数据被传送到 SPI 移位寄存器时, 该位被置 1, 表示可以安全地向发送缓冲器写新字节。
0	SPIEN	SPI 使能位

		0: 禁止SPIn 1: 使能SPI
--	--	-----------------------

15.5.4 SPISCR 寄存器

SPI 时钟速率寄存器，地址 0XA2

位	7	6	5	4	3	2	1	0
名称	SCR[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值								

位	名称	功能
7:0	SCR[7:0]	<p>SPI 时钟频率</p> <p>当SPI 模块被配置为工作于主方式时，这些位决定SCK 输出的频率。SCK 时钟频率是从系统时钟分频得到的，由下面的方程给出，其中：SYSCLK 是系统时钟频率，SPICKR 是spi_scr 寄存器中的8 位值。</p> $f_{SCK} = SYSCLK / 2 * (SPICKR + 1)$ <p>(0 ≤ spi_scr ≤ 255)</p> <p>例如：如果SYSCLK = 2MHz，SPICKR = 0x04，则</p> $f_{SCK} = 2000000 / 2 * (4 + 1)$ $f_{SCK} = 200 \text{ kHz}$

15.5.5 SPIDAT 寄存器

SPI 数据寄存器，地址 0XA3

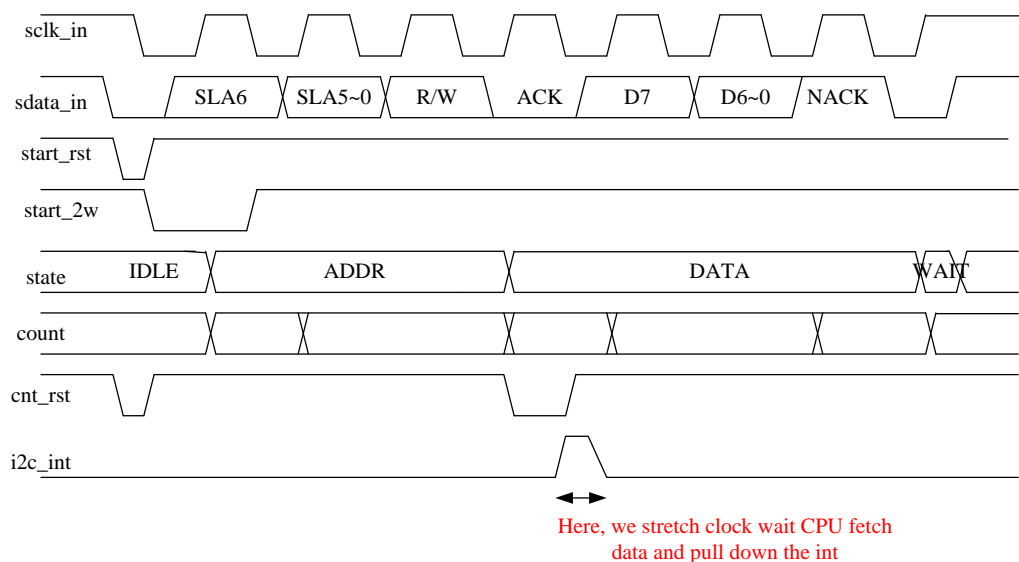
位	7	6	5	4	3	2	1	0
名称	SPIDAT							
类型	R	R	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
7:0	SPIDAT	<p>SPI 发送和接收数据寄存器。</p> <p>SPIDAT 寄存器用于发送和接收SPI 数据。在主方式下，向SPIDAT 写入数据时，数据被放到发送缓冲器并启动发送。读SPIDAT 返回接收缓冲器的内容。</p>

16.0 I2C

YS62F0132 的 I2C 接口是一个二线的双向串行总线，完全符合 I2C 协议定义 3.0，可运行于高速模式，最高可达到 400KBits/S。本模块只支持从机模式，由独立时钟控制，可以在 sleep 模式下工作，并且当接收的地址与本机相匹配时可以唤醒 CPU。

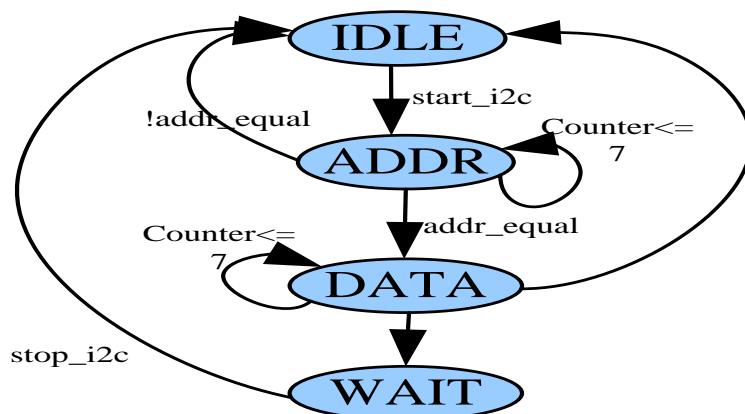
一次典型的 I2C 数据传输包括一个起始条件（START）、一个地址字节（7 位）、一个或多个字节的数据和一个停止条件（STOP）。每个接收的字节都必须用 SCL 高电平期间的 SDA 低电平来确认（ACK）。如果接收器件不确认（ACK），则发送器件将读到一个“非确认”（NACK），这用 SCL 高电平期间的 SDA 高电平表示。此时，硬件把 STRETCH(I2C_STAT.[0])置 1，来判断是否进入下一个状态。下图表示一次典型的 I2C 数据传输过程。



I2C接收波形图

I2C 外设接口在每次上电后必须要先经过配置相应的寄存器才能使用 I2C。当接收到开始的标志后，将 start_i2c 拉高，进入到 ADDR 状态，在这个状态下接收主机端发送的地址，如果地址匹配，则进入下一个状态。如果地址不匹配，则 I2C 不动作，退回到 IDLE 状态。在 ADDR 状态，我们还同时接收读写标志位，决定是发送还是接收数据。每接收完一个 byte 的数据后，可根据是否接收到结束的标志决定是停留在 DATA 状态还是返回到 IDLE 状态。

注： 为了更稳定工作，外部SCL和SDA引脚上最好加上拉电阻。



16.1 时钟延长

在任何 SCL 低电平阶段，I2C 总线上的任何器件都可以保持 SCL 线低电平以及延时，或暂停数据发送。此发送“延长”允许器件减慢总线上的通信速度。主器件必须不断对 SCL 线采样，以确保总线上的所有器件都已释放 SCL 以接收更多数据。

延长通常发生在发送 ACK 位之后，以延时发送下一字节的第 1 个位。

16.2 I2C 操作步骤

1. 设置 I2C_EN 为高，使能 I2C；
2. CPU 侦测 I2C 状态位，如果发现 I2C 中断拉高，CPU 将延长时钟并且接收或发送数据；
3. 当 CPU 完成上述工作，将复位 I2C 中断位，结束时钟延长；
4. 回到 步骤 2. 。

16.3 I2C 寄存器

I2C 总线有 5 个寄存器，分别是 FCTR,I2C_DIN,I2C_DOUT,I2C_SLAD,I2C_STAT，其中 FCTR 可设置 I2C 使能及 I2C 通讯超时监控等功能，请查阅 IO 输入/输出端口寄存器说明。

16.3.1 FCTR 寄存器

FCTR：功能控制寄存器，地址：0xB5

位	7	6	5	4	3	2	1	0
名称	(Reserved)	UARTOIR	I2CTO	I2CEN	I2CDDO	UARTFTR	SPIFTR	I2CFTR

类型	0	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能
[7]	(保留位)	只读，读为0。
[6]	UARTOIR	UART功能章节介绍
[5]	I2CTO	I2C通讯超时监控使能。 0: 禁用超时监控; 1: 启用超时监控。开启后, Timer2会自动监控I2C通讯是否超时。
[4]	I2CEN	I2C模块使能控制。 0: I2C模块禁用 1: I2C模块使能
[3]	I2CDDO	用于控制I2C在输出高电平时, 是否直接输出。(只影响SDA信号) 0: 不允许直接输出。I2C在需要输出高电平时, 自动将端口设置成输入状态, 通过外部上拉电阻形成高电平输出; 而在需要输出低电平时, 正常输出。 1: 允许直接输出。I2C在输出高、低电平时, 不需要通过外部电阻上拉控制, 直接输出电平。 注: 这一位对外是无效位。我们暂时做成固定为0。
[2]	UARTFTR	UART功能章节介绍
[1]	SPIFTR	SPI功能章节介绍
[0]	I2CFTR	控制I2C功能是否转移。I2C功能默认在P1.0、P1.2上, 功能转移以后, 分别在P0.0和P0.1上。

16.3.2 I2C_DIN 寄存器

I2C 接受数据寄存器, 地址 0XB8

位	7	6	5	4	3	2	1	0
名称	I2C_DIN[7:0]							
类型	R	R	R	R	R	R	R	R
复位值								

位	名称	功能
7:0	I2C_DIN[7:0]	I2C 接口从 master 端接收的数据

16.3.3 I2C_DOUT 寄存器

I2C 发送数据寄存器，地址 0XBB

位	7	6	5	4	3	2	1	0
名称	I2C_DOUT[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值								

位	名称	功能
7:0	I2C_DOUT[7:0]	I2C 发送数据到 master 端

16.3.4 I2C_SLAD 寄存器

I2C 从机地址寄存器，地址 0XBC

位	7	6	5	4	3	2	1	0
名称	I2C_SLAD[6:0]							
类型	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值								

SFR Page = 0xF; SFR Address = 0xAD

位	名称	功能
6:0	I2C_SLAD[6:0]	I2C 从机地址。

16.3.5 I2C_STAT 寄存器

I2C 状态寄存器，地址 0XBD

位	7	6	5	4	3	2	1	0
名称	DMOD	ACTIVE	I2CINT	NACK	START	STOP	DATA	STRETCH
类型	R	R	R	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	x	0	0	0	0

位	名称	功能
Bit 7	DMOD	用来指示当前 I2C 是读模式还是写模式； 0：写模式 1：读模式
Bit 6	ACTIVE	当前数据的地址匹配时，硬件自动置 1； 当 I2C 检测到 NACK 或 STOP 或者地址不匹配的 START 时，硬件清 0；

		I2C 没有被使能时，ACTIVE 恒为 0。
Bit 5	I2CINT	当 Bit[3:1]任何一位为高时，I2CINT 为高； 当 Bit[3:1]全为 0 时，I2CINT 为 0； I2C 没有被使能时，I2CINT 恒为 0。
Bit 4	NACK	用来指示当前发送/接收的字节的响应位； 0 表示 ACK； 1 表示 NACK； 当收到正确的地址后，硬件自动清 0； 写模式时，软件往此位写 1 将此位置 1，表示接下来的字节响应 NACK； 读模式时，此位反应的是主机发来的响应位。 CPU 确保 NACK 位写 0 无动作。
Bit 3	START	当收到匹配的地址字节后，硬件置 1； 软件往此位写 0 清除此位，I2C 没有被使能时，START 恒为 0；
Bit 2	STOP	当前数据的地址匹配时，收到 STOP，硬件置 1； 软件往此位写 0 清除此位，I2C 没有被使能时，STOP 恒为 0。
Bit 1	DATA	当前数据的地址匹配时，接收/发送完数据字节，硬件置 1； 软件往此位写 0 清除此位，I2C 没有被使能时，DATA 恒为 0。
Bit 0	STRETCH	在地址匹配的情况下，I2C 处理完每个字节的第 9 位时，硬件自动置 1，SCL 被硬件 Stretch；软件往此位写 0 清除此位，释放 SCL 的 Stretch。

注：I2C_STAT 寄存器在 I2C 时钟域；

I2C_STAT 寄存器 Bit[3:0]为异步复位，MCU 需保证复位信号无毛刺；

往 I2C_STAT 寄存器的 Bit[4]写 1 将其置 1，异步置位，MCU 需保证置位信号无毛刺。

17.0 乘法器 16 位* 16 位

YS62F0132 集成一个 16x16 的无符号数乘法器。使用时分为 MCU 模式和 DMA 模式。MCU 模式主要用于单次乘法操作，DMA 模式主要用于硬件加速操作。

MCU 模式时的操作步骤：

- 写 MAC0MOD=1；
- 分别写 MAC0AH/MAC0AL、MAC0BH/MAC0BL；
- 写 MAC0STAR=1；
- 读 MAC0STAR，直到其为 0，即表示本次乘法操作完成；
- 读取 MAC0ACC3~0 的值即为所求结果。

17.1 乘法器寄存器

17.1.1 MAC_CTRL 寄存器

MAC_CTRL:乘法器控制寄存器，地址 0X97

位	7	6	5	4	3	2	1	0
名称	MAC0MEMCTRL	MAC0MOD	/	/	/	/	/	/
类型	R	R/W	R	R	R	R/W	R	R
复位值	0	0	0	0	0	0	0	0

位	名称	功能描述
7	MAC0MEMCTRL	DMA 模式下，MAC 模块具有 SRAM 的控制权。 0: MCU 控制 SRAM 1: MAC 模块控制 SRAM。 =MAC0START && !MAC0MOD
6	MAC0MOD	0: MAC0 工作在 DMA 模式 1: MAC0 工作在 MCU 模式
5:0	Reversed	

17.1.2 MAC_STAT 寄存器

MAC_STAT: 乘法器状态寄存器，地址 0X9F

位	7	6	5	4	3	2	1	0
名称	/	MAC0INT	/	/	/	/	/	MAC0START
类型	R	R/W	R/W	R	R/W	R	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	名称	功能描述
7:1	Reversed	注：MCU 模式下，MAC0INT 不产生中断
0	MAC0START	MAC0 启动位。 向此位写 1，就可以触发 MAC0 开始以 MCU 模式或 DMA 模式工作。 1、在 MCU 模式下，MAC0 寄存器中已有的数据执行单个 MAC 操作，在操作完成后，将 MAC0START 清 0；

17.1.3 MAC0CF0 寄存器

MAC0CF0: 乘法器配置寄存器，地址 0XC5

位	7	6	5	4	3	2	1	0
名称			MAC0CA		MAC0MS			
类型	R	R	R/W	R	R/W			
复位值	0	0	0	0	0			

位	名称	功能描述
7:6	Reserve	
5	MAC0CA	累加器清 0（仅 MCU 模式有效） 用于在下次操作之前，将 MAC0 复位。当向这一位写 1 时，MAC0 累加器被清 0。除了 MAC0INT 和 MAC0ACC_RDY 以外，MAC0STA 寄存器中所有其它位都清 0。在操作结束时，这一位自动清 0。软件读取这一位时，永远都是 0。

4	Reserve	
3	MAC0MS	MAC0 模式选择。 用于选择 MAC 模式或仅仅是乘法模式。 设置为 1: 单独乘法模式
2:0	Reserve	

17.1.4 MAC0AH 寄存器

MAC0AH: 乘法器操作数 A 高位字节, 地址 0XAB

位	7	6	5	4	3	2	1	0
名称	MAC0AH[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
说明	在 MCU 模式下, 在 mcu 写寄存器或 MAC0BUSY 的情况下更新 REGA 的值							

17.1.5 MAC0AL 寄存器

MAC0AL: 乘法器操作数 A 低位字节, 地址 0XAA

位	7	6	5	4	3	2	1	0
名称	MAC0AL[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
说明	在 MCU 模式下, 在 mcu 写寄存器或 MAC0BUSY 的情况下更新 REGA 的值							

17.1.6 MAC0BH 寄存器

MAC0BH: 乘法器 B 操作数高位字节, 地址 0XAD

位	7	6	5	4	3	2	1	0
名称	MAC0BH[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
说明	在 MCU 模式下, 在 mcu 写寄存器或 MAC0BUSY 的情况下更新 REGB 的值							

17.1.7 MA0CBL 寄存器

MACBL: 乘法器 B 操作数低位字节, 地址 0XAC

位	7	6	5	4	3	2	1	0
名称	MAC0BL[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

17.1.8 MAC0ACC3 寄存器

MAC0ACC3: 乘法器累加器字节 3, 地址 0XE4

位	7	6	5	4	3	2	1	0
名称	MAC0ACC3[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
说明	在 MCU 模式下, 在 mcu 写寄存器的情况下更新 ACC 的值							

17.1.9 MAC0ACC2 寄存器

MAC0ACC2: 乘法器累加器字节 2, 地址 0XE3

位	7	6	5	4	3	2	1	0
名称	MAC0ACC2[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
说明	在 MCU 模式下, 在 mcu 写寄存器的情况下更新 ACC 的值							

17.1.10 MAC0ACC1 寄存器

MAC0ACC1: 乘法器累加器字节 1, 地址 0XE2

位	7	6	5	4	3	2	1	0
名称	MAC0ACC1[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
说明	在 MCU 模式下, 在 mcu 写寄存器的情况下更新 ACC 的值							

17.1.11 MAC0ACC0 寄存器

MAC0ACC0: 乘法器累加器字节 0, 地址 0XE1

位	7	6	5	4	3	2	1	0
名称	MAC0ACC0[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

复位值	0	0	0	0	0	0	0	0
说明	在 MCU 模式下，在 mcu 写寄存器的情况下更新 ACC 的值							

18.0 除法器 32 位/32 位

内置在 SFR 模块的 32 位除法器执行无符号数除法操作，由两组寄存器组成（DIV_WA/DIV_WB）。为节省 SFR 地址，商放在 DIV_WA，余数放在 DIV_WB。

使用方法：

- 软件把要做除法运算的被除数写到 DIV_WA，把除数写到 DIV_WB；
- 往 DIV_START（PERCF.6）写 1，启动 DIV32；
- 经过 14 个系统时钟后运算结束，这一步可通过查询 DIV_FINISH（PERCF.6）完成，DIV_FINISH=1 表示除法结束，否则还需等待；
- 查询到 DIV_FINISH=1，把结果取走。

注：

1. 如果做除法运算时除数 DIV_WB 为 0，除法器将产生一个 DIV_ERR（PERCF.7）标志，它将维持在高电平直到下一次做除法时除数不为 0；
2. 除法器在工作时（DIV_FINISH=0），商和余数结果都是不确定的（仿真时的值为 8'hxx），只有在除法器空闲时读取的商和余数才是稳定正确的；
3. 除法器在工作时（DIV_FINISH=0），改变除数或被除数的值都不会影响最后的结果，除非再来一次 DIV_START（往 PERCF.6 写 1）。

18.1 除法寄存器

18.1.1 DIV_WA0 寄存器

寄存器 DIV_WA0，地址 0XB1

DIV_WA[7:0]位，只写。读返回的是除法的商结果，DIV_QUOT[7:0]。

位	7	6	5	4	3	2	1	0
名称	DIV_WA0							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0

18.1.2 DIV_WA1 寄存器

寄存器 DIV_WA1，地址 0XB2

DIV_WA[15:8]位，只写。读返回的是除法的商结果，DIV_QUOT[15:8]。

位	7	6	5	4	3	2	1	0
名称	DIV_WA1							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0

18.1.3 DIV_WA2 寄存器

寄存器 DIV_WA2，地址 0XB3

DIV_WA[23:16]位，只写。读返回的是除法的商结果，DIV_QUOT[23:16]。

位	7	6	5	4	3	2	1	0
名称	DIV_WA2							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0

18.1.4 DIV_WA3 寄存器

寄存器 DIV_WA3，地址 0XB3

DIV_WA[31:24]位，只写。读返回的是除法的商结果，DIV_QUOT[31:24]。

位	7	6	5	4	3	2	1	0
名称	DIV_WA3							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0

18.1.5 DIV_WB0 寄存器

寄存器 DIV_WB0，地址 0XC1

DIV_WB[7:0]位，只写。读返回的是除法的余数结果，DIV_REM[7:0]。

位	7	6	5	4	3	2	1	0
名称	DIV_WB0							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0

18.1.6 DIV_WB1 寄存器

寄存器 DIV_WB1，地址 0XC2

DIV_WB[15:8]位，只写。读返回的是除法的余数结果，DIV_REM[15:8]。

位	7	6	5	4	3	2	1	0
名称	DIV_WB1							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0

18.1.7 DIV_WB2 寄存器

寄存器 DIV_WB2，地址 0XC3

DIV_WB[23:16]位，只写。读返回的是除法的余数结果，DIV_REM[23:16]。

位	7	6	5	4	3	2	1	0
名称	DIV_WB2							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0

18.1.8 DIV_WB3 寄存器

寄存器 DIV_WB3，地址 0XC4

DIV_WB[31:24]位，只写。读返回的是除法的余数结果，DIV_REM[31:24]。

位	7	6	5	4	3	2	1	0
名称	DIV_WB3							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0

19.0 Touch

YS62F0132 支持互电容扫描与自电容扫描，可通过寄存器 cap_mod 进行自由切换。自电容触控最多支持 32 触摸通道数，用作互容扫描最多支持 20 行*12 列（20 TX*12 RX）。能自动更新触摸环境，触摸灵敏度通过内部寄存器对其进行简单操作，不需外接任何元件。

内建 2 片 8bit*1KB XRAM，作为 A/B RAM（俗称“乒乓 Buffer RAM”），目的是为了提高了系统速度，当 MCU 在处理上一帧数据的同时，新一帧的扫描仍将正常进行，避免了因 MCU 处理数据而无法进行下一次扫描的情况。当关闭触摸功能时，A/B RAM 可作为 MCU 通用 XDATA RAM。

19.1 互容扫描

YS62F0132 自容和互容可自由切换，但不能同时使用。扫描最大支持 20 TX * 12 RX，可驱动 5 寸触摸屏，5 点触控检测，2 点/5 点触摸报点率分别可达 125Hz/100Hz。如使用互容功能，可调用汇春公司打包的算法，具体说明请查阅应用说明文档。

19.2 自容扫描

自电容检测(Self Capacitive Sensor of 16bit，简称 CS16)有 32 个自感电容检测通道，支持 3 种扫描模式，自动更新触摸门限以达到自适应检测功能。

下表扫描模式与自动更新门限：

扫描模式	auto_scan	Manu_scan	cs0mcen	Sth_upd_en	实现方式
single scan: 单独扫描	0	0	0	X	扫描一次 mcu 指定的任何一个通道
			1	X	短接所有 mask 通道,并扫描一次
manual scan: 手动扫描	0	1	0	X	逐一扫描一次所有 io_mask 的通道
			1	X	短接所有 io_mask 通道，并扫描一次，然后逐一扫描一次所有 io_mask 通道
auto scan: 自动反复扫描	1	X	0	0	自动逐一扫描所有 io_mask 通道，不累加，不更新 threshold。自动定时重复扫描
				1	自动逐一扫描所有 io_mask 的通道，并累加、更新 io_mask 的 threshold，不更新 mult_threshold。自动定时重复扫描
	1	X	1	0	短接所有 io_mask 通道，并扫描，不累加，不更新 threshold。自动定时重复扫描
				1	短接所有 io_mask 通道并扫描，更新 mult_threshold，并在累加时逐一扫描一次所有 io_mask 通道，并更新相应 io_mask 通道的 threshold。并且，当每次短接通道扫描的值超过 mult_threshold 时，随后紧接着扫描其它 io_mask 通道一次。自动定时重复扫描

19.3 自电容寄存器列表

Addr. 16'hxxx	BIT Fields . ASM top								Default 8'hxx
	7	6	5	4	3	2	1	0	
0x2000	asm_en		Aram_en	Bram_en	slp_en		gram_en	asm_ie	8'h01
0x2001	asm_busy	ram_ready	Aram_full	Bram_full	slp_rq	asm_eos	ram_handle	drop_flag	8'h00
0x2002				ie_slp	slp_int_clr				8'h08
0x2003	sleep[23:16]								8'h00
0x2004	sleep[15:8]								8'h06
0x2005	sleep[7:0]								8'h66
0x2006	cap_mod	AB_en	AB_mod		Reserve				8'h40
0x2007 ~ 0x201F	Reserve : 25 bytes								8'hXX
addr. 16'hxxx	BIT Fields. Self CAP scan								Default 8'hxx
	7	6	5	4	3	2	1	0	
0x2100	cs0en	cs0start				ie_self_eos	ie_self_cmp	self_ie	8'h00
0x2101	cs0cmpf	cs0eos_rq	cs0int				cs0busy	mult_result	8'h00
0x2102	cs0cmpen	cs0mcen	cs0cr[1:0]		cs0acu[2:0]			mult_pol	8'h10
0x2103	auto_scan	manu_scan							8'h00
0x2104									8'h00
0x2105	cs0mx[4:0]								8'h00
0x2106	chan_result[31:24]								8'h00
0x2107	chan_result[23:16]								8'h00
0x2108	chan_result[15:8]								8'h00
0x2109	chan_result[7:0]								8'h00
0x210A	io_mask [31:24]								8'h00
0x210B	io_mask [23:16]								8'h00
0x210C	io_mask [15:8]								8'h00

86Product Specification (V1.0)www.vsprinatech.com

0x210D	io_mask [7:0]								8'h00
0x210E	self_tmr[15:8] / tmp_cap[15:8]								8'h06
0x210F	self_tmr[7:0] / tmp_cap[7:0]								8'h66
0x2110									8'h00
0x2111			cg_mx[2:0]						8'h38
0x2112 0x2113	Reserve:								/
0x2114	chan_pol[31:24]								8'h00
0x2115	chan_pol[23:16]								8'h00
0x2116	chan_pol[15:8]								8'h00
0x2117	chan_pol[7:0]								8'h00
0x2118	sth_upd_en	sth_mod	sth_force	sth_acc_rst					8'h00
0x2119	sth_gap[3:0]				sth_dead_scope[3:0]				8'h74
0x211a	/		/					/	/
0x211b~ 0x23FF	0x2119~ 0x23FF: 743 bytes Reserve								8'hxx
Addr. 16'hxxx	BIT Fields. RAM Space								Default
	7	6	5	4	3	2	1	0	8'hxx
0x2400~ 0x27FF	0x24e7 ~ 0x24ef: Gauss SRAM: 9 Bytes for MCU XDATA								8'hXX
	0x1570 ~ 0x166f: Gauss SRAM: 256 Bytes for gauss table								8'hXX
0x2800~ 0x2BFF	ARAM: 1024 Bytes								8'hXX
0x2C0 ~ 0x2FFF	BRAM: 1024 Bytes								8'hXX

Memory Space Summary:

ADDR.	SIZE	Content
2C00~2FFF	1KB	BRAM
2800~2BFF	1KB	ARAM
2400~27FF	1KB	GAUSS SRAM
2000~23FF	1KB	ASM SFR
Toal: 2000~2FFF	4KB	ASM

注: Reserve

MCU 只读

19.4 寄存器定义

地址	16'h2000 : asm_main_en, ASM top control							
位	7	6	5	4	3	2	1	0
名称	asm_en	/	Aram_en	Bram_en	slp_en	/	gram_en	asm_ie
类型	R/W	/	R/W	R/W	R/W	/	R/W	R/W
复位值	0	/	0	0	0	/	0	1
位	名称		功能					
[7]	asm_en		ASM main enable 0: disable main scan。 1: enable main scan。 Recommend : do not set it to 0 when asm is busy					
[6]	Reserved							
[5]	Aram_en		ARAM enable 0: disable ARAM for ASM 1: enable ARAM for ASM Aram_en=1, ASM 取得 ARAM 控制权, MCU 对 ARAM 无控制权 Aram_en=0, MCU 取得 ARAM 控制权, ASM 对 ARAM 无控制权。 Recommend: do not set to 0 when ARAM is bufferizing data					
[4]	Bram_en		BRAM enable 0: disable BRAM for ASM 1: enable BRAM for ASM Bram_en=1, ASM 取得 BRAM 控制权, MCU 对 BRAM 无控制权 Bram_en=0, MCU 取得 BRAM 控制权, ASM 对 BRAM 无控制权。 Recommend: do not set to 0 when BRAM is bufferizing data					
[3]	slp_en		休眠计数器使能 0: 关闭休眠计数器 1: 打开休眠计数器 本功能与 cap. sensor 无关。当 slp_rq_en=1 时, 数字休眠计数器以内部时钟 32768Hz 开始从 sleep 作减法计数, 当计数到等于 0 时, 产生 slp_rq 标志位。此后重新开始计数, 如此反复, 周期性产生 slp 中断。 ie_slp=1 时, slp_rq 产生 slp_int 中断, MCU 可写 slp_int_clr=0 清除 slp_rq 标志。slp_int 中断电平宽 1 个慢时钟周期。 ie_slp=0 时, 不产生 slp_int 中断, 但 slp_rq 标志位给出一个宽 32768Hz 的高电平。					
[2]	Reserved							
[1]	gram_en		Gauss ram operation enable 0 : 1K bytes Gauss ram 由 MCU 控制 1 : 1K bytes Gauss ram 由 ASM 控制 Recommend: do not set to 0 when mutu_scan busy or self_scan threshold updating					
[0]	asm_ie		ASM interrupt main enable 0: disable ASM main interrupt 1: enable ASM main interrupt					

地址	16'h2001 : asm_main_status, ASM top control							
位	7	6	5	4	3	2	1	0
名称	asm_busy	ram_ready	Aram_full	Bram_full	slp_rq	asm_eos	ram_handle	drop_flag
类型	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0
位	名称	功能						
[7]	asm_busy	ASM scan busy flag 0: ASM idle 1: ASM is busy, scanning						
[6]	ram_ready	当前模式下, ram 是否准备好 0: ram 未准备好 1: ram 已准备好						
[5]	Aram_full	ARAM 数据缓存完毕标志, MCU 写 Aram_en=0 时自动清该位为 0 0: ARAM 数据未缓存完毕。1: ARAM 数据已缓存完毕						
[4]	Bram_full	BRAM 数据缓存完毕标志, MCU 写 Bram_en=0 时自动清该位为 0 0: BRAM 数据未缓存完毕。 1: BRAM 数据已缓存完毕						
[3]	slp_rq	休眠计数器溢出标志位 0: 休眠计数器未溢出 1: 休眠计数器溢出 休眠计数器减法计数到达 0 时, 该标志位置 1。 当 ie_slp=0 时, 该标志位置 1 并仅延迟 1 个慢时钟周期, 之后自动清 0, 且不产生 slp_int 中断。当 ie_slp=1 时, 计数器溢出时该标志置 1, 直至 MCU 写 slp_int_clr=0 以清 0。slp_int 中断信号为 1 个 32KHz 的高电平, 自动清 0, 随着计数器的反复溢出而呈周期性中断。						
[2]	asm_eos	ASM end flag of scan 0: not end of current scan 1: complete current scan 新扫描开始时自动清 0, MCU 清中断时可清该位为 0						
[1]	ram_handle	ARAM/BRAM 数据缓冲标记 0: 最新一次扫描得到的数据缓冲在 ARAM 中 1: 最新一次扫描得到的数据缓冲在 BRAM 中						
[0]	drop_flag	扫描数据丢失标志。表示未被 MCU 处理过的扫描数据是否被新扫描数据覆盖。下次扫描未丢失时自动清 0。 0: 缓存于 A/B RAM 中的扫描数据均被 MCU 处理过, 数据未丢失。 1: 原扫描数据尚未被 MCU 处理就被新的扫描数据覆盖。						

地址	16'h2002 : asm_sleep_timer interrupt enable			
位	7:5	4	3	2:0
名称	/	ie_slp	slp_int_clr	/
类型	/	W/R	W/R	/
复位值	/	1'b0	1'b1	Reverse
位	名称	功能		
[7:5]	Reserved			
[4]	ie_slp	<p>slp 中断使能:</p> <p>0: 禁止 slp_int 中断 1: 允许 slp_int 中断</p> <p>当 ie_slp=0 时, slp_rq 标志位置 1 并仅延迟 1 个慢时钟周期, 之后自动清 0, 且不产生 slp_int 中断。</p> <p>当 ie_slp=1 时, slp_rq 标志位长时间置 1, 同时送出 slp_int 中断, 直至 MCU 写 slp_int_clr=0, 将同时清 slp_rq=0</p>		
[3]	slp_int_clr	<p>Sleep 中断清 0</p> <p>0: 24bit sleep 计数溢出时会使 slp_int 信号产生高电平中断、slp_rq 寄存器为 1'b1。MCU 可写 slp_int_clr 为 0, 立即清 slp_rq 寄存器为 1'b0, 1 个慢时钟周期内 slp_int_clr 自动恢复为 1。</p> <p>注意: MCU 在获知中断为 slp_int 时, 应写 slp_int_clr=0 以清除 slp_rq, 否则 slp_rq 无法自动清 0。slp_int 作为周期性中断信号向 MCU 发送中断电平, 电平宽度为 1 个慢时钟周期, 自动清 0。</p> <p>1: MCU 不清 sleep 中断。(default)</p>		
[2:0]	Reserved			

地址	16'h2003 ~ 16'h2005 : sleept		
位	16'h2003 - [7:0]	16'h2004 - [7:0]	16'h2005 - [7:0]
名称	sleep[23:16]	sleep[15:8]	sleep[7:0]
类型	R/W	R/W	R/W
复位值	8'h00	8'h06	8'h66
位	名称	功能	
[23:0]	sleept	<p>slp_ctrl 计数器计数设置。</p> <p>以内部 32768Hz sclk 为时钟进行计数，当计数值达到(sleept+1)时，计数完毕上报中断 slp_int 为高，并且置中断标志位 slp_rq=1，计数器继续计数，直至下一次溢出中断，自动重载并反复。</p> <p>MCU 获知该中断为 slp_int 中断后应写 slp_int_clr=0 以清除 slp_int/slp_rq，否则 slp_rq/slp_int 不会自动清 0。</p> <p>slp_rq_en=1 打开此功能，slp_rq_en=0 关闭此功能。</p> <p>ie_slp 为 slp_int 中断允许位。</p> <p>Slp_rq 为溢出标志位。</p>	

地址	16'h2006 : abram_mod				
位	7	6	5	4	3:0
名称	cap_mod	AB_en	AB_mod/	/	dtest[3:0]
类型	R/W	R/W	R/W	/	R/W
复位值	0	1'b1	1'b0	/	4'h0
位	名称	功能			
[7]	cap_mod	Capacitor sensor type mode 0: mutual capacitor sensor 1: self capacitor sensor Recommend : do not swith it when asm busy			
[6]	AB_en	A/B RAM function enable 0: disable A/B RAM function。将扫描后的数据仅保存于 ARAM。 1: enable A/B RAM function。将扫描后的数据保存于 ARAM 或 BRAM。当启动互电容自动更新门限功能或者自动比较功能时,MCU 需要释放 BRAM 控制权给 ASM 以保存中间数据、门限及比较标志位。			
[5]	AB_mod	A/B RAM switch mode 0: A/B RAM 手动切换模式。 Aram_full(或 Bram_full)=1 且 Bram_en(或 Aram_en)=0 时, 或者 A/Bram 均 full 时, 不再允许扫描数据缓存, 并且不启动扫描。自动扫描时若延时到达但因 MCU 尚未处理完上一帧数据, 而 ASM 未获得 A/B ram 控制权, 则 ASM 不扫描, 一旦获得 A/B ram 控制权, 即开始扫描。而手动扫描 MCU 必须先给予未 full 的 RAM 控制权才能开启扫描。 1: A/B RAM 自动切换模式。 Aram_full(或 Bram_full)=1 时, 仍将按 MCU 指令开启扫描, 但若 MCU 丢失一帧数据未处理, 则 drop_flag=1 作为标志。 比如, Aram_en=1,Aram_full=1,Bram_en=0 时, 若强行扫描则新的扫描数据会覆盖 ARAM 中的原数据, 无论这些原数据是否被 MCU 处理过。特别地, 当需要 MCU 休眠而 ASM 自动扫描时, MCU 可设置该位为 1, 这样 MCU 不需要重新设置 aram_en 或 bram_en 为 0/1 来准备所需的 RAM, ASM 会自动缓存数据到相应 RAM 中。			
[4]	Reserved				
[3:0]	Reserved				

地址	16'h2100: self_enable, self CAP sar control					
位	7	6	5:3	2	1	0
名称	cs0en	cs0start	/	ie_self_eos	ie_self_cmp	self_ie
类型	R/W	R/W	/	R/W	R/W	R/W
复位值	0	0	/	0	0	0
位	名称	功能				
[7]	cs0en	Self CAP scan enable. 0: cs16 disabled 1: cs16 enable and ready to convert 自电容扫描启动位。 cs0en=0 时禁止所有自电容扫描。 Recommend : do not set it to 0 when self CAP scan is busy				
[6]	cs0start	Self CAP : manual scan start, 手动扫描使能位。 0: disable manual scan 1: enable manual scan 当需要进行非自动扫描(即 auto_scan=0)时, mcu 可置 cs0en=1、cs0start=1 开始扫描, 扫描完毕后产生 cs0eos_rq 标志并上报中断, cs0start 自动清 0, 此后 mcu 若再次需要扫描则直接设置 cs0start=1 即可开始新的扫描。 该位不参与控制自动扫描。				
[5:3]	Reserve					
[2]	ie_self_eos	self CAP : interrupt enable of eos. 0: cs16 will not give a interrupt after it completed all channel scanning. 1: cs16 will give a interrupt after it completed all channel scanning.				
[1]	ie_self_cmp	Self CAP: Interrupt enable of digital comparor 0: disable of the interrupt of comparor 1: enable of the interrupt of comparor 数字比较器比较中断使能, 即触摸中断使能位。当 ie_cmp=1 时, 才允许数字比较器比较的结果产生中断, 若 ie_cmp=0, 即使当前电容值超过了门限值并且 cs0cmpf=1, 也不会产生触摸中断。 此位不会影响标志位 cs0cmpf 的置位情况。				
[0]	self_ie	Self CAP scan main interrupt enable 0 : disable all self CAP interrupt. 1: enable self CAP main interrupt.				

地址	16'h2101: self_status, self CAP sar control					
位	7	6	5	4:2	1	0
名称	cs0cmpf	cs0eos_rq	cs0int	/	cs0busy	mult_result
类型	R	R	R	/	R	R
复位值	0	0	0	/	0	0
位	名称	功能				
[7]	cs0cmpf	<p>cs16 digital compareator interrupt flag.</p> <p>0: 本次扫描的所有通道逼近结果值比该通道门限值小 1: 本次扫描的通道中有一个或一个以上的通道结果值比对应门限值大。</p> <p>所有通道扫描完毕后，ASM 根据扫描比较结果 chan_result[31:0]和 mult_result 的值置 cs0cmpf 位，高电位有效，下次扫描开始后自动清 0。</p> <p>该中断标志位受 cs0cmphen 使能，受 chan_pol/mult_pol 比较方向影响。不受中断使能位影响，self_ie=1 且 ie_self_cmp=1 时该位置 1 时会产生 self_int 中断</p>				
[6]	cs0eos_rq	<p>cs16 end of scan interrupt flag</p> <p>0: 本次扫描未结束 1: 本次扫描结束</p> <p>本次扫描的通道全部扫描完成后置 cs0eos_rq 为 1'b1，下次扫描开始后自动清 0。该标志位不受中断使能影响，当 self_ie=1,ie_self_eos=1 时，该标志位的置位会产生中断。</p>				
[5]	cs0int	<p>Channel scan completed flag</p> <p>通道扫描结束标志位</p> <p>每当一个通道扫描结束时，该位均会出现一个高电平作为标志，高电平宽度 1 个 charge clk(约 2.5us)。</p> <p>该位多用于调试。</p>				
[4:2]	Reserve					
[1]	cs0busy	<p>Self CAP. busy flag, self CAP. sensor is scanning</p> <p>0: Self CAP is not busy 1: Self CAP is busy</p>				
[0]	mult_result	<p>Multiple scan compare result</p> <p>多通道短接扫描比较结果标志位。</p> <p>当多通道短接扫描结束时，比较当前值 mult_val 与 mult_th，是否超过门限，若超过则置 mult_result 为 1，比较方向受 mult_pol 控制。</p> <p>该标志位保持到下次扫描开始时自动清零。cs0mcen=0 时不会进行多通道短接扫描，该位也不会被置 1。</p>				

Addr.	16'h2102: self_config, self CAP sar control				
Bit	7	6	5:4	3:1	0
Name	cs0cmpen	cs0mcen	cs0cr[1:0]	cs0acu[2:0]	mult_pol
Type	R/W	R/W	R/W	R/W	R/W
Reset	0	0	2'b01	3'b000	0
Bit	Name	Description			
[7]	cs0cmpen	cs16 digital comparator enable. 0: digital comparator disabled 1: digital comparator enabled 数字比较器打开使能，若打开则比较通道电容值 mxN_val[15:0] 与其门限 mxN_th[15:0]。比较器方向受 chan_pol/mult_pol 控制，default 情况下为正向：cap_val >= threshold 时置位。 cs0cmpen 可看作 chan_result[31:0]、mult_result 及 cs0cmpf 的使能，当 cs0cmpen=1'b0 时，比较器关闭，chan_result、mult_result 及 cs0cmpf 不会被置 1。			
[6]	cs0mcen	cs16 multiple channel enable 0: multiple channel feature is disable 1: multiple channel feature is enabled. 同时打开 io_mask 中的所有通道。			
[5:4]	cs0cr	cs16 conversion rate 2'b00: 12 位逐次逼近，低 4 位为 0 2'b01: 13 位逐次逼近，低 3 位为 0 2'b10: 14 位逐次逼近，低 2 位为 0 2'b11: 16 位逐次逼近			
[3:1]	cs0acu	cs16 accumulator mode select. 3'b000: accumulate 1 sample. 3'b001: accumulate 4 samples. 3'b010: accumulate 8 samples. 3'b011: accumulate 16 samples. 3'b100: accumulate 32 samples. 3'b101: accumulate 64 samples. 3'b11x: reserved.			
[0]	mult_pol	cs16 digital comparer polarity select: mult_pol CS16 数字比较器方向选择之多通道短接比较方向 0: Mult 正向比较(default) 当前 mult 扫描结果比门限值大(mult_val > mult_th)时，将视为 touch，置 mult_result 为 1，并置 cs0cmpf 为 1。 1: Mult 反向比较 当前 mult 扫描结果比门限值小(mult_val < mult_th)时，将视为 touch，置 mult_result 为 1，并置 cs0cmpf 为 1。			

Addr.	16'h2103: self_scan_mod, self CAP threshold update		
Bit	7	6	5:0
Name	auto_scan	manu_scan	/
Type	R/W	R/W	/
Reset	0	0	/
Bit	Name	Description	
[7]	auto_scan	automatic scan enable 0: 非自动扫描模式 1: 自动扫描模式。 设置该位为 1'b1 时打开扫描计时器，进入自动扫描模式，然后设置 cs0en=1 启动自动扫描。	
[6]	manu_scan	manual scan mode 0: 单独扫描模式 1: 手动扫描模式 auto_scan=0, manu_scan=0 : 单独扫描模式。对 cs0mx 指定的任意单一通道进行扫描。 auto_scan=0, manu_scan=1 : 手动扫描模式。对 io_mask 指定的任意通道进行逐一扫描。 auto_scan=1, manu_scan=X : 自动扫描模式。对 io_mask 指定的任意通道进行周期性自动扫描。 Recommend : if want to switch scan mode, should set cs0en =0 then switch scan mode and set cs0en=1 again to start new scan, do not switch scan mode when scan is busy.	
[5:0]	Reserved		

地址	16'h2106 ~ 16'h2109 : self_chan_result, Self_CAP channel scan result			
位	16'h2106 - [7:0]	16'h2107 - [7:0]	16'h2108 - [7:0]	16'h2109 - [7:0]
名称	chan_result[31:24]	chan_result[23:16]	chan_result[15:8]	chan_result[7:0]
类型	R	R	R	R
复位值	8'h00	8'h00	8'h00	8'h00
位	名称	功能		
[31:0]	chan_result	<p>scan channel compare result 通道扫描比较结果。</p> <p>当启动数字比较器功能后，扫描结束时，ASM 自动比较当前值与对应的 threshold，是否超过门限，若超过则置相应通道 chan_result 为 1。32 个通道共有 32 个结果，多通道短接时对应的是寄存器 mult_result。MCU 可利用此判断哪个通道触摸了。</p> <p>该扫描结果保持到下次扫描开始时自动清零，受 chan_pol 方向控制，受 cs0cmpen 使能</p>		

地址	16'h210A ~ 16'h210D : self_io_mask, Self_CAP channel scan MASK			
位	16'h210A - [7:0]	16'h210B - [7:0]	16'h210C - [7:0]	16'h210D - [7:0]
名称	io_mask[31:24]	io_mask[23:16]	io_mask[15:8]	io_mask[7:0]
类型	R/W	R/W	R/W	R/W
复位值	8'h00	8'h00	8'h00	8'h00
位	名称	功能		
[31:0]	io_mask	<p>Self_CAP channel scan mask 自电容通道扫描掩膜</p> <p>在手动扫描和自动扫描模式下，通道扫描掩膜被设置为 1 的通道才会被扫描。</p> <p>在单独扫描时，不受 io_mask 限制，MCU 可指定任一通道进行扫描(对应寄存器写 cs0mx[4:0])。</p>		

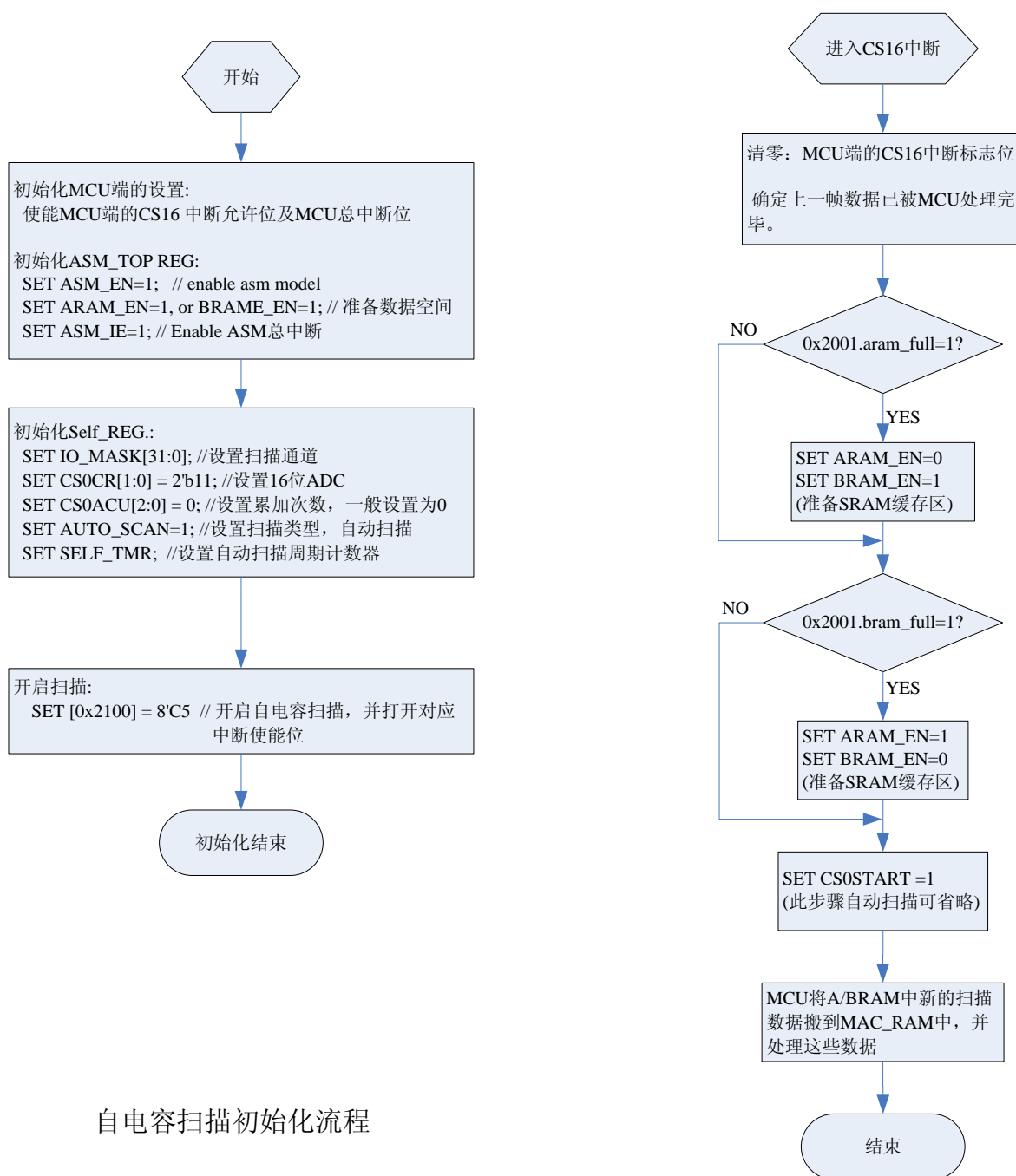
地址	16'h210E ~ 16'h210F : self_tmr, Self_CAP channel scan timer	
位	16'h210E - [7:0]	16'h210F - [7:0]
名称	self_tmr[15:8]	self_tmr[7:0]
类型	R/W	R/W
复位值	8'h06	8'h66
位	名称	功能
[15:0]	self_tmr	<p>Self_tmr : Self_CAP cs16 auto scan timer.</p> <p>cs16 在自动扫描模式时,两次扫描的间隔时间。以 32768Hz 时钟进行计时。Default 值为约 20Hz(50ms)扫描一次。</p>

地址	16'h2114 ~ 16'h2117 : self_chan_pol, Self_CAP 32 channels digital comparer polarity			
位	16'h2114 - [7:0]	16'h2115 - [7:0]	16'h2116 - [7:0]	16'h2117 - [7:0]
名称	chan_pol[31:24]	chan_pol[23:16]	chan_pol[15:8]	chan_pol[7:0]
类型	R/W	R/W	R/W	R/W
复位值	8'h00	8'h00	8'h00	8'h00
位	名称	功能		
[31:0]	chan_pol	<p>Self_CAP 32 channels digital comparer polarity 自电容 32 个通道数字比较器方向控制位。</p> <p>0: 通道正方向比较。当本次扫描 $mxN_val \geq mxN_th$ 时, 对应 chan_result 置 1, 并且任意一个 chan_result=1 时均会置 cs0cmpf=1 表示发生了触摸事件。(default)</p> <p>1: 通道反方向比较。当本次扫描 $mxN_val < mxN_th$ 时, 对应 chan_result 置 1, 并且任意一个 chan_result=1 时均会置 cs0cmpf=1 表示发生了触摸事件。</p>		

地址.	16'h2118: self_threshold_en, self CAP threshold update				
位	7	6	5	4	3:0
名称	sth_upd_en	sth_mod	sth_force	sth_acc_rst	/
类型	R/W	R/W	R/W	R/W	/
复位值	0	0	0	0	/
位	名称	功能			
[7]	sth_upd_en	<p>Self CAP .Threshold update enable 门限自动更新功能：使能位</p> <p>0: 33 路触摸门限值由 MCU 写入 1: 33 路触摸门限由 self_scan 自动更新，只有在自动扫描模式才能启动此功能。</p> <p>注意，启动自电容门限自动更新功能前，MCU 必须先写合适的初始门限值及门限差值至 Gauss_RAM 对应的空间，然后写 gram_en=1 将 Gauss_RAM(GRAM)的控制权交于 ASM。然后才能启动自电容门限自动更新功能</p>			
[6]	sth_mod	<p>Self CAP. Threshold update mode 门限自动更新功能：门限累加模式控制</p> <p>0: 累加 8 次当前值，平均后更新门限 1: 累加 16 次当前值，平均后更新门限</p>			
[5]	sth_force	<p>Self CAP. Threshold update force 门限自动更新功能：强制门限累加控制</p> <p>0: 在自动更新门限时，若遇到 touch 或者 touch_dead_area，不进行累加操作 1: 在自动更新门限时，无论是否有 touch 或 touch_dea_area，都将进行相关累加操作，并且一旦更新完门限值后，sth_force 立即会被自动清零。</p>			
[4]	sth_acc_rst	<p>Threshold accumulator reset 门限自动更新功能：门限累加复位控制</p> <p>0: 在自动更新时，若遇到 touch 或者 touch_dead_area，则累加器跳过，不进行累加操作，但不清零，之后继续累加。 1: 在自动更新时，若遇到 touch 或者 touch_dead_area，则累加器立即清零。之后，从零开始进行累加 注意，sth_force=1 时，sth_acc_rst 无效。</p>			
[3:0]	Reserved				

地址	16'h2119: self_th_config, self CAP threshold update	
位	7:4	3:0
名称	sth_gap[3:0]	sth_dead_scope[3:0]
类型	W/R	W/R
复位值	4'h7	4'h4
位	名称	Description
[7:4]	sth_gap	<p>Self CAP. Threshold gap 门限自动更新功能：门限累加采样间隔。</p> <p>前一次扫描累加之后，下一次累加是在之后的第(sth_gap + 1)次扫描结束后累加。 特别地，当 sth_gap=4'h0 时，两次累加之间没有间隔扫描，即两次连续扫描均参与累加运算。当 sth_gap=4'hF 时两次累加间隔为 15 次，第 16 次扫描值参与累加。</p>
[3:0]	sth_dead_scope	<p>Self CAP Touch dead scope 门限自动更新功能：死区范围</p> <p>发现触摸之后，对于门限累加器及间隔计数器而言产生了一个死区，在这个死区范围里不累加 gap_cnt 值，也不进行门限累加。死区范围为(sth_dead_scope + 1)次扫描。 并且，sth_acc_rst 的置位会使得当门限累加器一旦遇到 dead_area 会立即清零。而 sth_force 的置位会使得门限累加器不考虑 dead scope 和 sth_acc_rst 值的情况。</p>

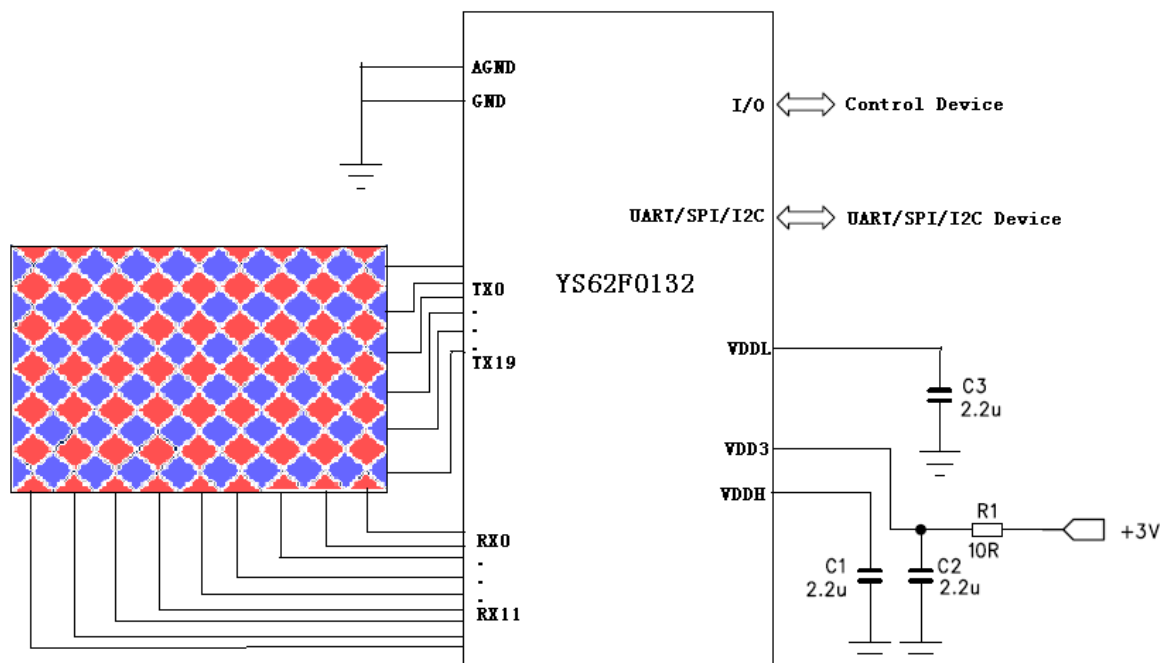
19.5 自电容触摸操作流程



自电容扫描初始化流程

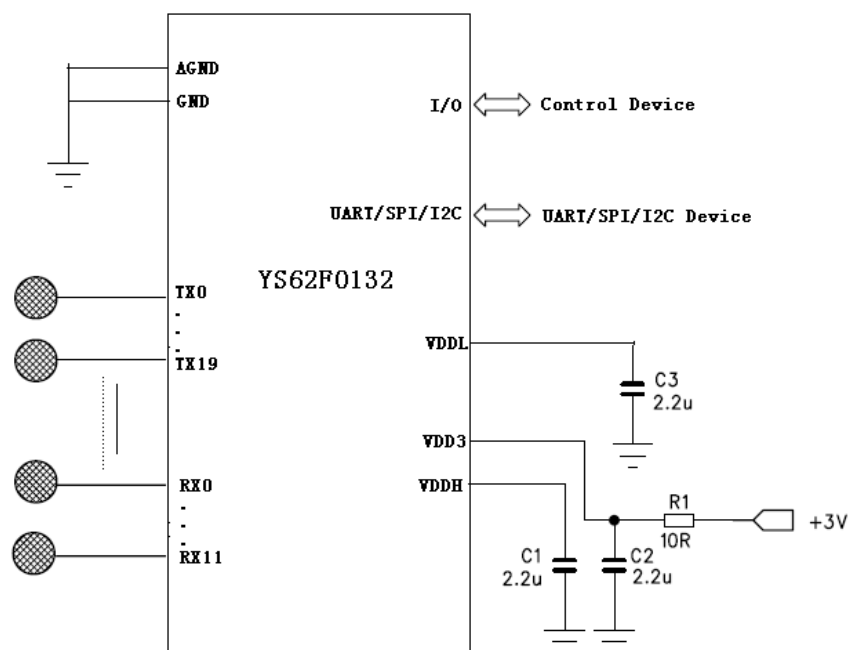
20.0 应用原理图

20.1 互容电路原理图

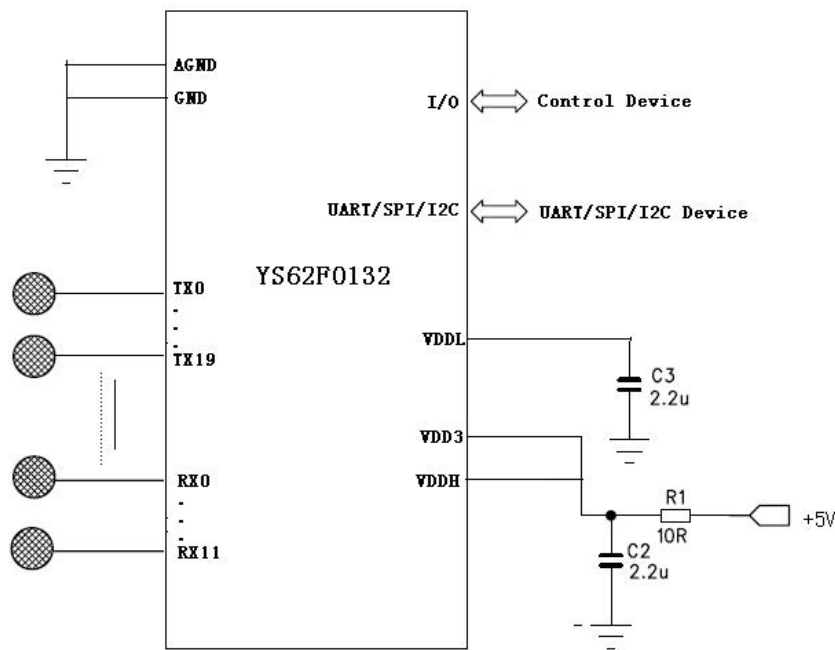


20.2 自容电路原理图

工作电压为 3V



工作电压为 5V



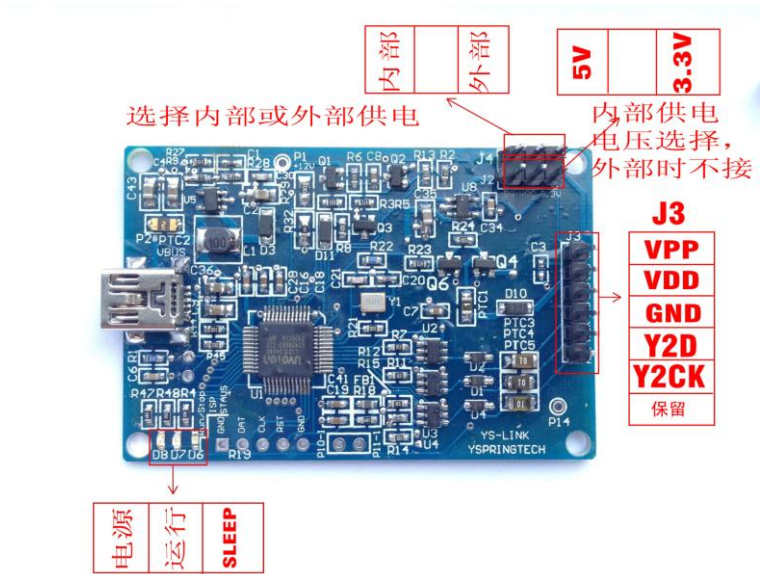
21.0 开发支持

21.1 仿真信息

21.1.2 软件：

- 支持 SFR/IRAM/XRAM 以及 Flash 的读写访问
- 支持 5 个用户断点
- 支持软复位暂停、单步进入 (step into)、单步跳过 (step over)、单步跳出 (step out)、全速运行以及运行到光标处的操作
- 支持取消所有断点命令

21.1.2 硬件：YS-Link



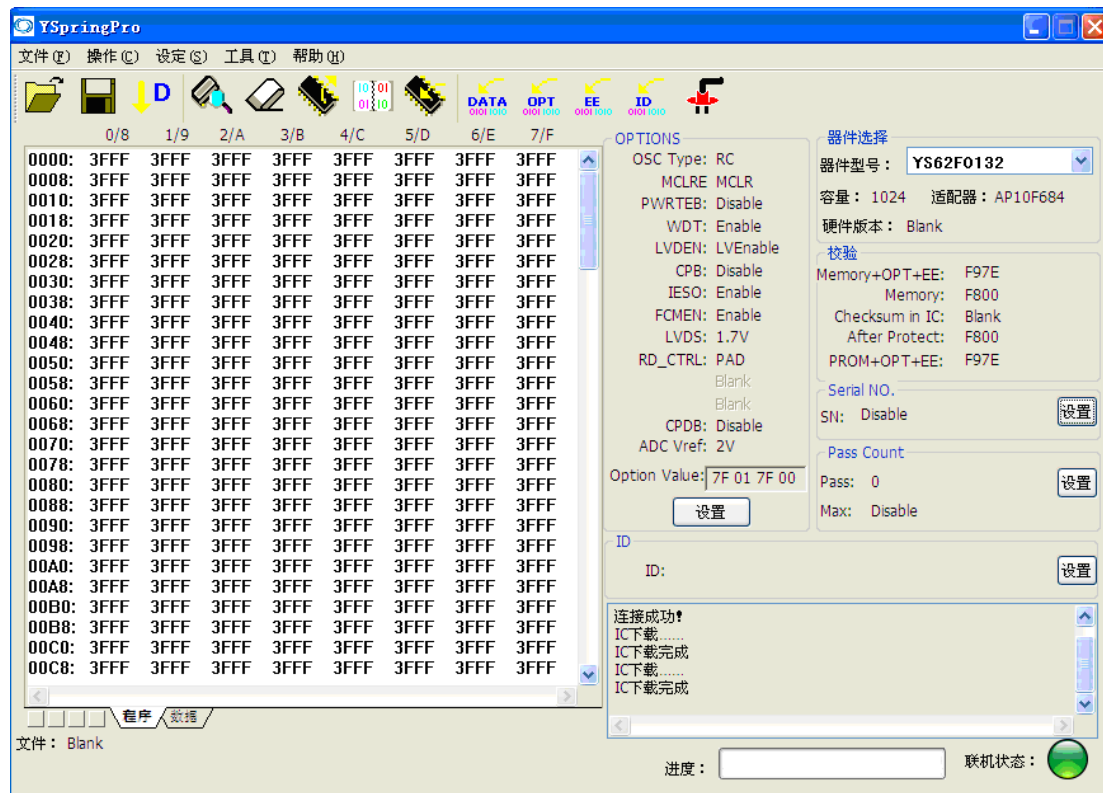
21.1.3 仿真调试接口

YS62F0132 具备片内仿真功能，可通过串口模式仿真调试程序，串口接口如下：

YS62F0132	YS-Link
VDD	VDD
GND	GND
RSTB	Y2K
P12	Y2D

21.2 烧录信息

21.2.1 烧录软件：YSpringPro

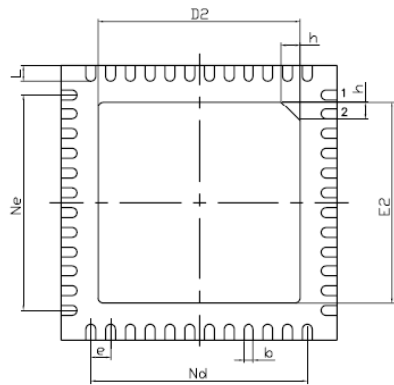
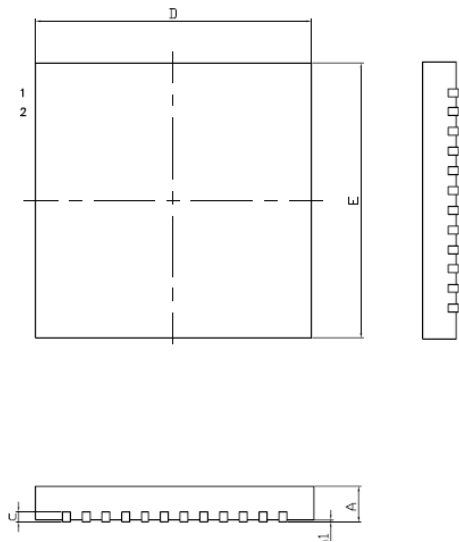


21.2.2 烧录器：YS-Writer



22.0 封装信息

22.1 QFN48L 0606x0.75-0.40



SYMBOL	MILLIMETER	
	MIN	MAX
A	0.70	0.80
A1	—	0.05
b	0.15	0.25
c	0.18	0.23
D	5.90	6.10
D2	4.10	4.30
e	0.40BSC	
Ne	4.40BSC	
Nd	4.40BSC	
E	5.90	6.10
E2	4.10	4.30
L	0.35	0.45
h	0.30	0.40

Technical drawing of a square plate with dimensions and a table of values.

Dimensions and Labels:

- Top View:** Square plate with side length D . Dimensions include $D1$, E , $E1$, 60 , 41 , 61 , 40 , 21 , 20 , 1 , b , e , a , $a1$, $(3X)$, 80 .
- Side View:** Shows the profile of the plate with dimensions c and s .
- Section B-B:** Cross-section showing **BASE METAL** and **WITH PLATING** layers. Dimensions include b , $b1$, c , and $c1$.
- Detail F:** Circular detail showing a cross-section with dimensions eB , L , $L1$, and 0.25 .
- Bottom View:** Shows the plate with dimensions A , $A1$, $A2$, $A3$, and F .

Table of Values (MILLIMETER):

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	0.17	—	0.27
b1	0.17	0.20	0.23
c	0.09	—	0.20
c1	0.09	0.13	0.16
D	13.80	14.00	14.20
D1	11.90	12.00	12.10
E	13.80	14.00	14.20
E1	11.90	12.00	12.10
eB	13.05	—	13.25
e	0.50BSC		
L	0.45	0.60	0.75
L1	1.00BSC		
θ	0	—	7°
\varnothing	$\varnothing 1.20 \times 0.05 \sim 0.10$ (DP)		
$\varnothing 1$	$\varnothing 1.80 \times 0.05 \sim 0.10$ (DP)		
1/4 板体尺寸 (mm)	240*240		

DETAIL: F

附录 1 FLASH 地址和块的对应关系

块	地址	块	地址
0	0x0000 – 0x03ff	16	0x4000 – 0x43ff
1	0x0400 – 0x07ff	17	0x4400 – 0x47ff
2	0x0800 – 0x0bff	18	0x4800 – 0x4bff
3	0x0c00 – 0x0fff	19	0x4c00 – 0x4fff
4	0x1000 – 0x13ff	20	0x5000 – 0x53ff
5	0x1400 – 0x17ff	21	0x5400 – 0x57ff
6	0x1800 – 0x1bff	22	0x5800 – 0x5bff
7	0x1c00 – 0x1fff	23	0x5c00 – 0x5fff
8	0x2000 – 0x23ff	24	0x6000 – 0x63ff
9	0x2400 – 0x27ff	25	0x6400 – 0x67ff
10	0x2800 – 0x2bff	26	0x6800 – 0x6bff
11	0x2c00 – 0x2fff	27	0x6c00 – 0x6fff
12	0x3000 – 0x33ff	28	0x7000 – 0x73ff
13	0x3400 – 0x37ff	29	0x7400 – 0x77ff
14	0x3800 – 0x3bff	30	0x7800 – 0x7bff
15	0x3c00 – 0x3fff	31	0x7c00 – 0x7fff

附录 2 FLASH 地址和页的对应关系

0	0x0000 – 0x007f	128	0x4000 – 0x407f
1	0x0080 – 0x00ff	129	0x4080 – 0x40ff
2	0x0100 – 0x017f	130	0x4100 – 0x417f
3	0x0180 – 0x01ff	131	0x4180 – 0x41ff
4	0x0200 – 0x027f	132	0x4200 – 0x427f
5	0x0280 – 0x02ff	133	0x4280 – 0x42ff
6	0x0300 – 0x037f	134	0x4300 – 0x437f
7	0x0380 – 0x03ff	135	0x4380 – 0x43ff
8	0x0400 – 0x047f	136	0x4400 – 0x447f
9	0x0480 – 0x04ff	137	0x4480 – 0x44ff
10	0x0500 – 0x057f	138	0x4500 – 0x457f
11	0x0580 – 0x05ff	139	0x4580 – 0x45ff
12	0x0600 – 0x067f	140	0x4600 – 0x467f
13	0x0680 – 0x06ff	141	0x4680 – 0x46ff
14	0x0700 – 0x077f	142	0x4700 – 0x477f
15	0x0780 – 0x07ff	143	0x4780 – 0x47ff
16	0x0800 – 0x087f	144	0x4800 – 0x487f

17	0x0880 – 0x08ff	145	0x4880 – 0x48ff
18	0x0900 – 0x097f	146	0x4900 – 0x497f
19	0x0980 – 0x09ff	147	0x4980 – 0x49ff
20	0x0a00 – 0x0a7f	148	0x4a00 – 0x4a7f
21	0x0a80 – 0x0aff	149	0x4a80 – 0x4aff
22	0x0b00 – 0x0b7f	150	0x4b00 – 0x4b7f
23	0x0b80 – 0x0bff	151	0x4b80 – 0x4bff
24	0x0c00 – 0x0c7f	152	0x4c00 – 0x4c7f
25	0x0c80 – 0x0cff	153	0x4c80 – 0x4cff
26	0x0d00 – 0x0d7f	154	0x4d00 – 0x4d7f
27	0x0d80 – 0x0dff	155	0x4d80 – 0x4dff
28	0x0e00 – 0x0e7f	156	0x4e00 – 0x4e7f
29	0x0e80 – 0x0eff	157	0x4e80 – 0x4eff
30	0x0f00 – 0x0f7f	158	0x4f00 – 0x4f7f
31	0x0f80 – 0x0fff	159	0x4f80 – 0x4fff
32	0x1000 – 0x107f	160	0x5000 – 0x507f
33	0x1080 – 0x10ff	161	0x5080 – 0x50ff
34	0x1100 – 0x117f	162	0x5100 – 0x517f
35	0x1180 – 0x11ff	163	0x5180 – 0x51ff
36	0x1200 – 0x127f	164	0x5200 – 0x527f
37	0x1280 – 0x12ff	165	0x5280 – 0x52ff
38	0x1300 – 0x137f	166	0x5300 – 0x537f
39	0x1380 – 0x13ff	167	0x5380 – 0x53ff
40	0x1400 – 0x147f	168	0x5400 – 0x547f
41	0x1480 – 0x14ff	169	0x5480 – 0x54ff
42	0x1500 – 0x157f	170	0x5500 – 0x557f
43	0x1580 – 0x15ff	171	0x5580 – 0x55ff
44	0x1600 – 0x167f	172	0x5600 – 0x567f
45	0x1680 – 0x16ff	173	0x5680 – 0x56ff
46	0x1700 – 0x177f	174	0x5700 – 0x577f
47	0x1780 – 0x17ff	175	0x5780 – 0x57ff
48	0x1800 – 0x187f	176	0x5800 – 0x587f
49	0x1880 – 0x18ff	177	0x5880 – 0x58ff
50	0x1900 – 0x197f	178	0x5900 – 0x597f
51	0x1980 – 0x19ff	179	0x5980 – 0x59ff
52	0x1a00 – 0x1a7f	180	0x5a00 – 0x5a7f
53	0x1a80 – 0x1aff	181	0x5a80 – 0x5aff
54	0x1b00 – 0x1b7f	182	0x5b00 – 0x5b7f
55	0x1b80 – 0x1bff	183	0x5b80 – 0x5bff
56	0x1c00 – 0x1c7f	184	0x5c00 – 0x5c7f
57	0x1c80 – 0x1cff	185	0x5c80 – 0x5cff
58	0x1d00 – 0x1d7f	186	0x5d00 – 0x5d7f
59	0x1d80 – 0x1dff	187	0x5d80 – 0x5dff

60	0x1e00 – 0x1e7f	188	0x5e00 – 0x5e7f
61	0x1e80 – 0x1eff	189	0x5e80 – 0x5eff
62	0x1f00 – 0x1f7f	190	0x5f00 – 0x5f7f
63	0x1f80 – 0x1fff	191	0x5f80 – 0x5fff
64	0x2000 – 0x207f	192	0x6000 – 0x607f
65	0x2080 – 0x20ff	193	0x6080 – 0x60ff
66	0x2100 – 0x217f	194	0x6100 – 0x617f
67	0x2180 – 0x21ff	195	0x6180 – 0x61ff
68	0x2200 – 0x227f	196	0x6200 – 0x627f
69	0x2280 – 0x22ff	197	0x6280 – 0x62ff
70	0x2300 – 0x237f	198	0x6300 – 0x637f
71	0x2380 – 0x23ff	199	0x6380 – 0x63ff
72	0x2400 – 0x247f	200	0x6400 – 0x647f
73	0x2480 – 0x24ff	201	0x6480 – 0x64ff
74	0x2500 – 0x257f	202	0x6500 – 0x657f
75	0x2580 – 0x25ff	203	0x6580 – 0x65ff
76	0x2600 – 0x267f	204	0x6600 – 0x667f
77	0x2680 – 0x26ff	205	0x6680 – 0x66ff
78	0x2700 – 0x277f	206	0x6700 – 0x677f
79	0x2780 – 0x27ff	207	0x6780 – 0x67ff
80	0x2800 – 0x287f	208	0x6800 – 0x687f
81	0x2880 – 0x28ff	209	0x6880 – 0x68ff
82	0x2900 – 0x297f	210	0x6900 – 0x697f
83	0x2980 – 0x29ff	211	0x6980 – 0x69ff
84	0x2a00 – 0x2a7f	212	0x6a00 – 0x6a7f
85	0x2a80 – 0x2aff	213	0x6a80 – 0x6aff
86	0x2b00 – 0x2b7f	214	0x6b00 – 0x6b7f
87	0x2b80 – 0x2bff	215	0x6b80 – 0x6bff
88	0x2c00 – 0x2c7f	216	0x6c00 – 0x6c7f
89	0x2c80 – 0x2cff	217	0x6c80 – 0x6cff
90	0x2d00 – 0x2d7f	218	0x6d00 – 0x6d7f
91	0x2d80 – 0x2dff	219	0x6d80 – 0x6dff
92	0x2e00 – 0x2e7f	220	0x6e00 – 0x6e7f
93	0x2e80 – 0x2eff	221	0x6e80 – 0x6eff
94	0x2f00 – 0x2f7f	222	0x6f00 – 0x6f7f
95	0x2f80 – 0x2fff	223	0x6f80 – 0x6fff
96	0x3000 – 0x307f	224	0x7000 – 0x707f
97	0x3080 – 0x30ff	225	0x7080 – 0x70ff
98	0x3100 – 0x317f	226	0x7100 – 0x717f
99	0x3180 – 0x31ff	227	0x7180 – 0x71ff
100	0x3200 – 0x327f	228	0x7200 – 0x727f
101	0x3280 – 0x32ff	229	0x7280 – 0x72ff
102	0x3300 – 0x337f	230	0x7300 – 0x737f

103	0x3380 – 0x33ff	231	0x7380 – 0x73ff
104	0x3400 – 0x347f	232	0x7400 – 0x747f
105	0x3480 – 0x34ff	233	0x7480 – 0x74ff
106	0x3500 – 0x357f	234	0x7500 – 0x757f
107	0x3580 – 0x35ff	235	0x7580 – 0x75ff
108	0x3600 – 0x367f	236	0x7600 – 0x767f
109	0x3680 – 0x36ff	237	0x7680 – 0x76ff
110	0x3700 – 0x377f	238	0x7700 – 0x777f
111	0x3780 – 0x37ff	239	0x7780 – 0x77ff
112	0x3800 – 0x387f	240	0x7800 – 0x787f
113	0x3880 – 0x38ff	241	0x7880 – 0x78ff
114	0x3900 – 0x397f	242	0x7900 – 0x797f
115	0x3980 – 0x39ff	243	0x7980 – 0x79ff
116	0x3a00 – 0x3a7f	244	0x7a00 – 0x7a7f
117	0x3a80 – 0x3aff	245	0x7a80 – 0x7aff
118	0x3b00 – 0x3b7f	246	0x7b00 – 0x7b7f
119	0x3b80 – 0x3bff	247	0x7b80 – 0x7bff
120	0x3c00 – 0x3c7f	248	0x7c00 – 0x7c7f
121	0x3c80 – 0x3cff	249	0x7c80 – 0x7cff
122	0x3d00 – 0x3d7f	250	0x7d00 – 0x7d7f
123	0x3d80 – 0x3dff	251	0x7d80 – 0x7dff
124	0x3e00 – 0x3e7f	252	0x7e00 – 0x7e7f
125	0x3e80 – 0x3eff	253	0x7e80 – 0x7eff
126	0x3f00 – 0x3f7f	254	0x7f00 – 0x7f7f
127	0x3f80 – 0x3fff	255	0x7f80 – 0x7fff

附录 3 自容扫描寄存器的 MCU 地址

自电容 通道编号	电容值(16'h) H:L		GAUSS_RAM		
	ARAM	BRAM	门限累加值 (16'h) H:M:L	门限值 (16'h) H:L	门限差值 (16'h) H:L
Mult	2BBE : 2BBF	2FBE : 2FBF	2400 : 2401 : 2402	2463 : 2464	24A5 : 24A6
5'h00	2BC0 : 2BC1	2FC0 : 2FC1	2403 : 2404 : 2405	2465 : 2466	24A7 : 24A8
5'h01	2BC2 : 2BC3	2FC2 : 2FC3	2406 : 2407 : 2408	2467 : 2468	24A9 : 24AA
5'h02	2BC4 : 2BC5	2FC4 : 2FC5	2409 : 240A : 240B	2469 : 246A	24AB : 24AC
5'h03	2BC6 : 2BC7	2FC6 : 2FC7	240C : 240D : 240E	246B : 246C	24AD : 24AE
5'h04	2BC8 : 2BC9	2FC8 : 2FC9	240F : 2410 : 2411	246D : 246E	24AF : 24B0
5'h05	2BCA : 2BCB	2FCA : 2FCB	2412 : 2413 : 2414	246F : 2470	24B1 : 24B2
5'h06	2BCC : 2BCD	2FCC : 2FCD	2415 : 2416 : 2417	2471 : 2472	24B3 : 24B4
5'h07	2BCE : 2BCF	2FCE : 2FCF	2418 : 2419 : 241A	2473 : 2474	24B5 : 24B6
5'h08	2BD0 : 2BD1	2FD0 : 2FD1	241B : 241C : 241D	2475 : 2476	24B7 : 24B8
5'h09	2BD2 : 2BD3	2FD2 : 2FD3	241E : 241F : 2420	2477 : 2478	24B9 : 24BA
5'h0A	2BD4 : 2BD5	2FD4 : 2FD5	2421 : 2422 : 2423	2479 : 247A	24BB : 24BC
5'h0B	2BD6 : 2BD7	2FD6 : 2FD7	2424 : 2425 : 2426	247B : 247C	24BD : 24BE
5'h0C	2BD8 : 2BD9	2FD8 : 2FD9	2427 : 2428 : 2429	247D : 247E	24BF : 24C0
5'h0D	2BDA : 2BDB	2FDA : 2FDB	242A : 242B : 242C	247F : 2480	24C1 : 24C2
5'h0E	2BDC : 2BDD	2FDC : 2FDD	242D : 242E : 242F	2481 : 2482	24C3 : 24C4
5'h0F	2BDE : 2BDF	2FDE : 2FDF	2430 : 2431 : 2432	2483 : 2484	24C5 : 24C6
5'h10	2BE0 : 2BE1	2FE0 : 2FE1	2433 : 2434 : 2435	2485 : 2486	24C7 : 24C8
5'h11	2BE2 : 2BE3	2FE2 : 2FE3	2436 : 2437 : 2438	2487 : 2488	24C9 : 24CA
5'h12	2BE4 : 2BE5	2FE4 : 2FE5	2439 : 243A : 243B	2489 : 248A	24CB : 24CC
5'h13	2BE6 : 2BE7	2FE6 : 2FE7	243C : 243D : 243E	248B : 248C	24CD : 24CE
5'h14	2BE8 : 2BE9	2FE8 : 2FE9	243F : 2440 : 2441	248D : 248E	24CF : 24D0
5'h15	2BEA : 2BEB	2FEA : 2FEB	2442 : 2443 : 2444	248F : 2490	24D1 : 24D2
5'h16	2BEC : 2BED	2FEC : 2FED	2445 : 2446 : 2447	2491 : 2492	24D3 : 24D4
5'h17	2BEE : 2BEF	2FEE : 2FEF	2448 : 2449 : 244A	2493 : 2494	24D5 : 24D6
5'h18	2BF0 : 2BF1	2FF0 : 2FF1	244B : 244C : 244D	2495 : 2496	24D7 : 24D8
5'h19	2BF2 : 2BF3	2FF2 : 2FF3	244E : 244F : 2450	2497 : 2498	24D9 : 24DA
5'h1A	2BF4 : 2BF5	2FF4 : 2FF5	2451 : 2452 : 2453	2499 : 249A	24DB : 24DC
5'h1B	2BF6 : 2BF7	2FF6 : 2FF7	2454 : 2455 : 2456	249B : 249C	24DD : 24DE
5'h1C	2BF8 : 2BF9	2FF8 : 2FF9	2457 : 2458 : 2459	249D : 249E	24DF : 24E0
5'h1D	2BFA : 2BFB	2FFA : 2FFB	245A : 245B : 245C	249F : 24A0	24E1 : 24E2
5'h1E	2BFC : 2BFD	2FFC : 2FFD	245D : 245E : 245F	24A1 : 24A2	24E3 : 24E4
5'h1F	2BFE : 2BFF	2FFE : 2FFF	2460 : 2461 : 2462	24A3 : 24A4	24E5 : 24E6
	ARAM:	BRAM:	GAUSS_RAM: 16'h2400 ~ 16'h27FF		

	16'h2800 ~ 16'h2BFF	16'h2C00 ~ 16'h2FFF	
--	------------------------	------------------------	--

23.0 汇春知识产权政策

23.1 专利权

汇春公司在全球各地区已核准和申请中之专利权至少有 16 件以上,享有绝对之合法权益。与汇春公司 MCU 或其它产品有关的专利权并未被同意授权使用,任何经由不当手段侵害汇春公司专利权之公司、组织或个人,汇春将采取一切可能的法律行动,遏止侵权者不当的侵权行为,并追讨汇春公司因侵权行为所受之损失、或侵权者所得之不法利益。

23.2 著作权

Copyright 2013 by INC.

规格书中所出现的信息在出版当时相信是正确的,然而汇春对于规格内容的使用不负责任。文中提到的应用其目的仅仅是用来做说明,汇春不保证或不表示这些应用没有更深入的修改就能适用,也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。汇春产品不授权使用于救生、维生器件或系统中做为关键器件。汇春拥有不事先通知而修改产品的权利,对于最新的信息,请参考我们的网址

<http://www.yspringtech.com;>

版本编号

版本	日期	更新内容	作者
V1.0	2013-01-11	原始版本	Maggie