

YS68FXXX(X)

数据手册（版本 V1.8）

8 位 Touch Flash MCU

1 器件概述

- 基于 8051 指令流水线结构的 8 位单片机
- Flash Rom: 8K 字节, 100,000 次可擦除, Firmware 可访问, 用户数据保护, 支持 16bit CRC 校检
- **RAM:**
 - IRAM 空间: 256 字节
 - XRAM 空间: 829 字节
 - SFR 空间: 128B
- **工作电压:**
 - VDD=1.8V~5.5V
- **振荡器:**
 - Fosc=12MHZ \pm 1% (内部/外部振荡)
 - Losc=32KHZ (内部/外部振荡)
- 最多 54 个 GPIO
- 16路Touch Key, 可通过软件单独对每路调节灵敏度
- 4 个通用 16 位定时器/计数器: T1、T2、T3、T4。
 - T2,T3,T4 具有 PWM、PPG 功能
 - T3,T4具有Capture 功能
- RTC计数器可计秒、分、时、星期和日期
- 看门狗定时器WDT
- **12通道12位ADC**
 - 精度 10bit/12bit ADC
 - ADC参考电压可软件选择为内部 (可选2V、3V、4V, \pm 1%误差) 或外部参考
 - 可内部测量1/2VDD
- **唤醒源**
 - 中断唤醒
 - 定时器唤醒
 - IO唤醒
 - I2C唤醒
 - 触摸唤醒
- **中断源: 2 级中断优先级**
 - 定时器
 - IO 沿中断
 - UART/SPI/I2C
 - Touch
- **串口通信:**
 - UART、SPI、I2C
- **LCD驱动器:**
 - 最高支持8COM*24SEG
 - 可调节显示频率
 - 支持1/3、1/4、1/5、1/6、1/8五种Duty可选择
 - 仅支持1/3 Bais
- **LED驱动器:**
 - 4*8段
 - 8*8段
- **工作模式:**
 - 正常模式, 电流< 2mA
 - IDLE模式, 电流<12uA (有触摸)
 - STOP模式, 电流< 5uA (无触摸)
- 支持在线仿真和烧写程序
- 封装: SSOP20、SOP24、SSOP24、SOP28、SSOP28、LQFP44、LQFP64

器件	VDD	ROM	RAM			I/O	Touch (通道数)	Timer	Interface	LCD	Package
		FLASH (word)	IRAM (word)	XRAM (word)	SFR (word)						
YS68F911N	1.8V~5.5V	8K	256	829	128	16	11	4*16bit	SPI/I2C/UART	0	SOP20
YS68F912	1.8V~5.5V	8K	256	829	128	20	12	4*16bit	SPI/I2C/UART	0	SOP24
YS68F916	1.8V~5.5V	8K	256	829	128	20	16	4*16bit	SPI/I2C/UART	0	SOP24
YS68F9908	1.8V~5.5V	8K	256	829	128	24	8	4*16bit	SPI/I2C/UART	0	SOP28
YS68F905	1.8V~5.5V	8K	256	829	128	21	5	4*16bit	SPI/I2C/UART	4Com*11Seg	SSOP28
YS68F910	1.8V~5.5V	8K	256	829	128	37	10	4*16bit	SPI/I2C/UART	4Com*20Seg	LQFP44
YS68F9916	1.8V~5.5V	8K	256	829	128	54	16	4*16bit	SPI/I2C/UART	3Com*29Seg	LQFP64

目录

1 器件概述	1
1.1 概述	5
1.2 系统结构图	6
1.3 封装脚位图	7
1.4 引脚说明	11
2 存储器构成	16
2.1 数据存储空间 (RAM)	16
2.2 特殊功能寄存器分布	17
2.3 FLASH 程序存储器	17
2.4 编程控制寄存器 (PSCTL)	18
2.5 FLASH 编程开锁寄存器 (FLKEY)	19
2.6 扇区和地址的关系	19
2.7 看门狗定时器 (WDT)	21
3 系统时钟和振荡器	22
3.1 概述	22
3.2 时钟缺失检测模式	22
3.3 时钟切换流程	22
3.4 快慢切换	22
3.5 由外部时钟切换到内部时钟	23
3.6 切换到外部时钟注意	23
3.7 时钟源选择寄存器	23
3.8 谐振器负载电容选择	24
4 功耗模式	25
4.1 正常模式	26
4.2 待机模式	26
4.3 睡眠模式	26
4.4 深度睡眠模式	26
4.5 功耗模式选择寄存器	26
5 复位源	28
5.1 概述	28
5.1 外部复位、上电复位	28
5.2 低电压侦测复位	28
5.3 看门狗溢出复位	28
5.4 过度电应力复位	28
5.5 软复位	29
5.6 复位状态寄存器 (RSTSRC)	29
5.7 复位源使能寄存器 (RSTEN)	29
6 IO 输入/输出端口	30
6.1 概述	30

6.2 功能控制寄存器 (FCTR)	32
6.3 上拉控制寄存器	33
6.4 模拟控制寄存器.....	35
6.5 端口方向控制寄存器.....	37
6.6 数据寄存器	40
7 中断系统.....	43
7.1 中断源	43
7.2 中断寄存器	44
8 定时器.....	51
8.1 定时器 1	51
8.2 定时器 2	55
8.3 定时器 3	59
8.4 定时器 4	63
9 实时时钟 (RTC)	69
9.1 RTC 寄存器	69
10 IIC.....	71
10.1 I2C 操作步骤.....	74
10.2 I2C 寄存器.....	74
11 增强型串行外设接口 (SPI)	76
11.1 引脚说明	76
11.2 SPI 通信.....	77
11.3 SPI 中断源	79
11.4 SPI 寄存器	79
12 UART	82
12.1 波特时钟与波特率	82
12.2 工作模式	83
12.3 UART 寄存器	85
13 模/数转换器 (ADC)	87
13.1 AD 寄存器	88
13.2 工作模式	95
14 触摸按键 (TOUCH KEY)	101
15 LCD/LED 驱动.....	102
15.1 LCD 驱动.....	102
15.2 LCD RAM 配置.....	103
15.3 LED 驱动器	109
15.4 LED RAM 配置	110
15.5 LCD/LED 驱动器.....	111
16 电气特性.....	114

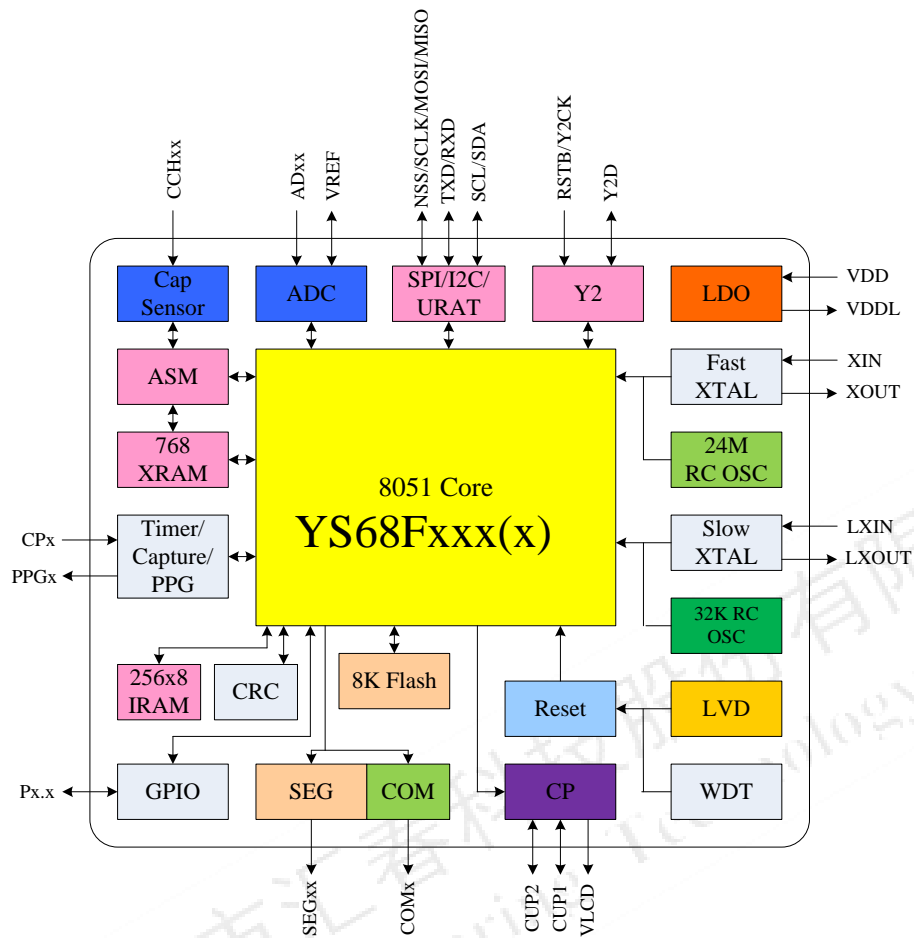
17 封装尺寸.....	116
17.1 SSOP20.....	116
17.2 SOP24.....	117
17.3 SSOP24.....	117
17.4 SOP28.....	118
17.5 SSOP28.....	119
17.6 LQFP44.....	119
17.7 LQFP 64.....	120
18 汇春知识产权政策.....	121
18.1 专利权.....	121
18.2 著作权.....	121

深圳市汇春科技股份有限公司
Shenzhen Yspring Technology Co., Ltd.

1.1 概述

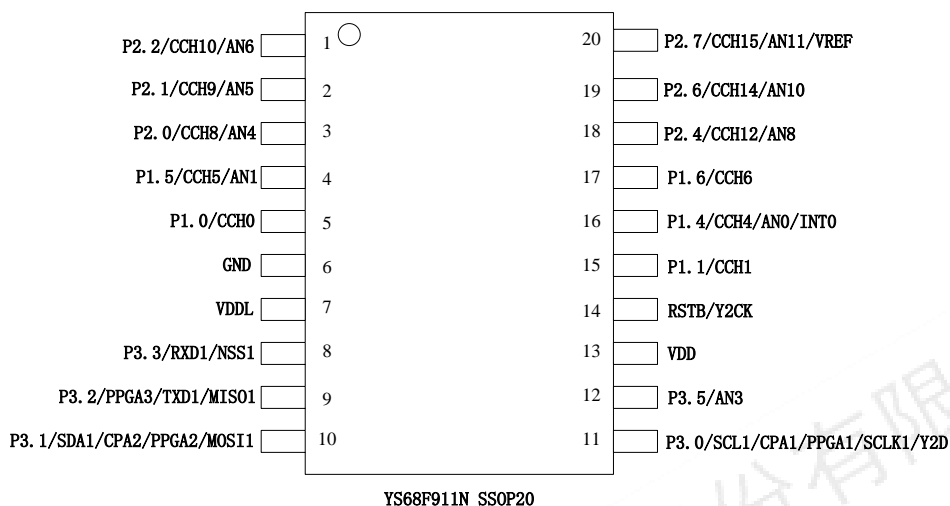
YS68F9XX(X)系列芯片是一款基于 1T 8051 内核的 8 位 FLASH MCU，内置 8K 字节 FLASH 程序存储器。在相同振荡频率下，运行速度较传统的 8051 芯片运行更快速的优越特性。除了保留标准 8051 芯片的基本特性外，还集成了 PWM、I2C、SPI、LCD/LED 驱动器、ADC、触摸通道、WDT、RTC 等模块，并能做到超低功耗。主要应用在各种车载音响、家用音响、蓝牙音箱、小家电、运动器材、马达控制、医疗保健、工业控制等方面。

1.2 系统结构图

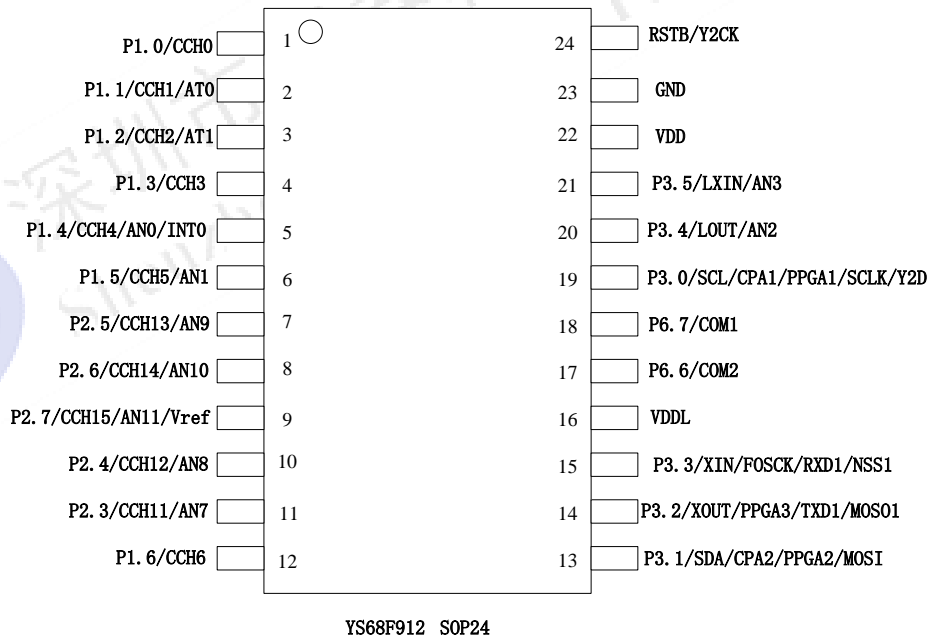


1.3 封装脚位图

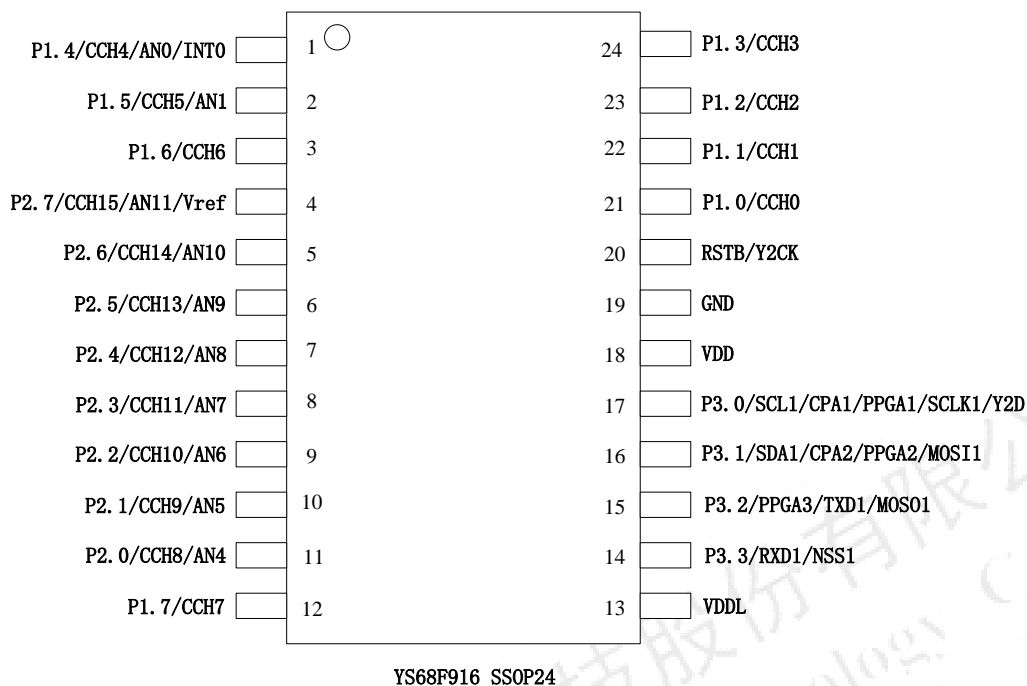
1.3.1 YS68F911N_SSOP20 引脚图



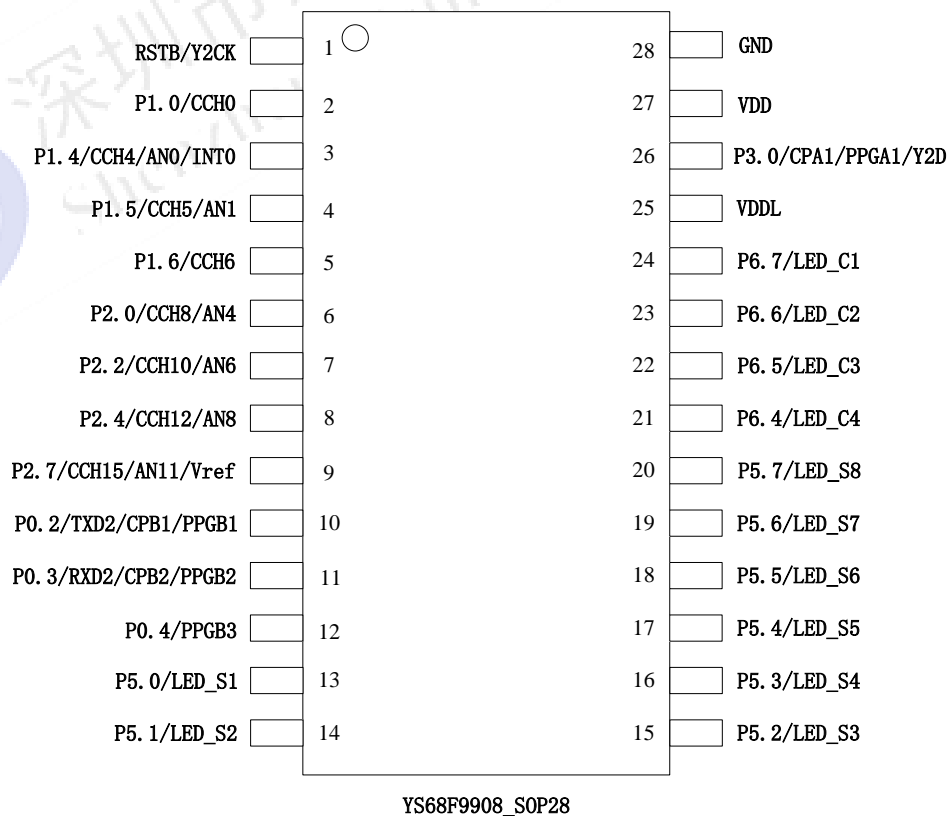
1.3.2 YS68F912 SOP24 引脚图



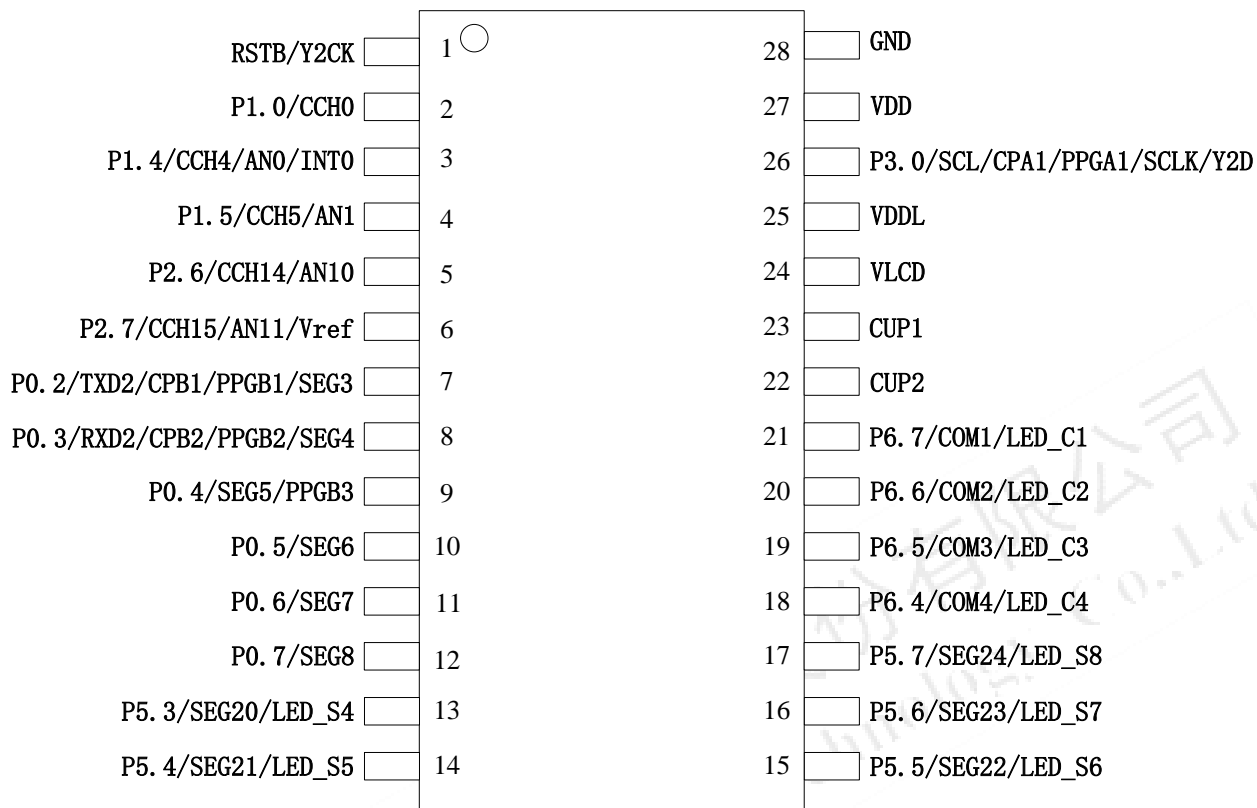
1.3.3 YS68F916 SSOP24 引脚图



1.3.4 YS68F9908 SOP28 引脚图

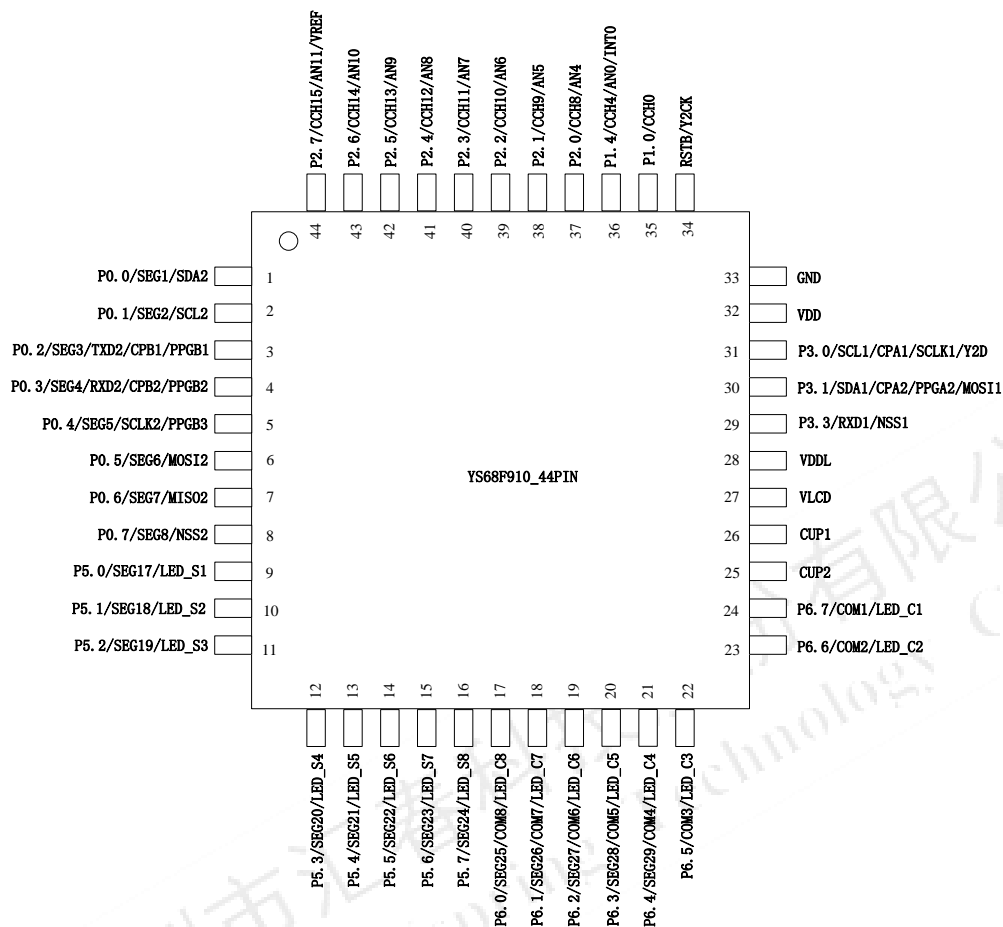


1.3.5 YS68F905 SSOP28 引脚图

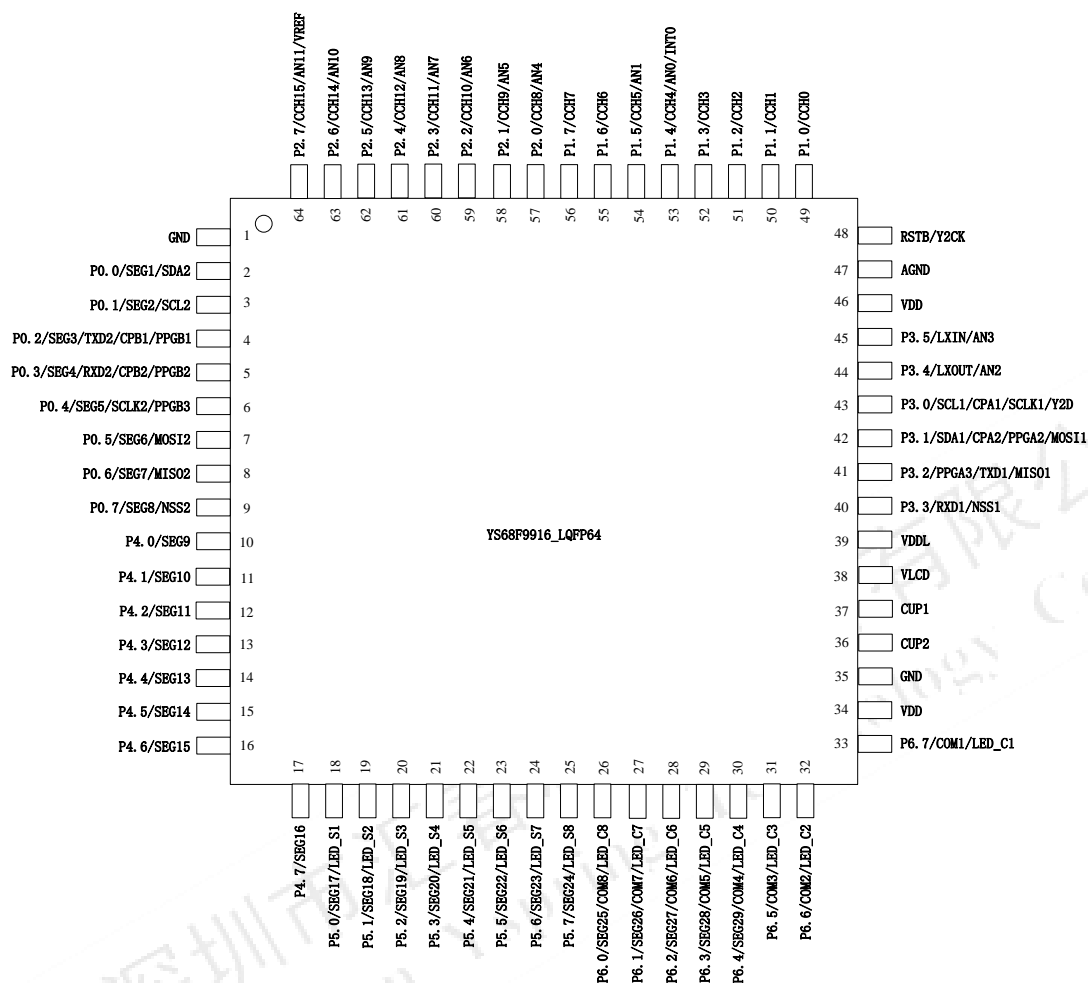


YS68F905 SSOP28

1.3.6 YS68F910_LQFP44 封装引脚图



1.3.7 YS68F9916_LQFP64 封装引脚图



1.4 引脚说明

Pin No.64	名称	Type	Description
1	GND	PWR	数字地
2	P0.0/ SEG1/ SDA2/	B	P0.0 可编程 IO, 可设置端口变化产生中断 LCD Segment 1 功能转移后的 I2C 之数据端
3	P0.1/ SEG2/ SCL2/	B	P0.1 可编程 IO, 可设置端口变化产生中断 LCD Segment 2 功能转移后的 I2C 之时钟端
4	P0.2/ SEG3/ TXD2/ CPB1/ PPGB1	B	P0.2 可编程 IO, 可设置端口变化产生中断 LCD Segment 3 功能转移后的 UART 之发射端 功能转移后的 CAPTURE 输入 1, 配合 timer3 使用 功能转移后的 PWM 输出 1, 配合 timer3 使用
5	P0.3/ SEG4/	B	P0.3 可编程 IO, 可设置端口变化产生中断 LCD Segment 4

	RXD2/ CPB2/ PPGB2		功能转移后的 UART 之接收端 功能转移后的 CAPTURE 输入 2, 配合 timer4 使用 功能转移后的 PWM 输出 2, 配合 timer4 使用
6	P0.4/ SEG5/ SCLK2/ PPGB3	B	P0.4 可编程 IO, 可设置端口变化产生中断 LCD Segment 5 功能转移后的 SPI 之时钟端 功能转移后的 PWM 输出 3, 配合 timer2 使用
7	P0.5/ SEG6/ MOSI2	B	P0.5 可编程 IO, 可设置端口变化产生中断 LCD Segment 6 功能转移后的 SPI 数据端, Master 模式输出, Slave 模式输入
8	P0.6/ SEG7/ MISO2	B	P0.6 可编程 IO, 可设置端口变化产生中断 LCD Segment 7 功能转移后的 SPI 数据端, Master 模式输入, Slave 模式输出
9	P0.7/ SEG8/ NSS2	B	P0.7 可编程 IO, 可设置端口变化产生中断 LCD Segment 8 功能转移后的 SPI 选择端口
10	P4.0/ SEG9	B	P4.0 可编程 IO LCD Segment 9
11	P4.1/ SEG10	B	P4.1 可编程 IO LCD Segment 10
12	P4.2/ SEG11	B	P4.2 可编程 IO LCD Segment 11
13	P4.3/ SEG12	B	P4.3 可编程 IO LCD Segment 12
14	P4.4/ SEG13	B	P4.4 可编程 IO LCD Segment 13
15	P4.5/ SEG14	B	P4.5 可编程 IO LCD Segment 14
16	P4.6/ SEG15	B	P4.6 可编程 IO LCD Segment 15
17	P4.7/ SEG16	B	P4.7 可编程 IO LCD Segment 16
18	P5.0/ SEG17/ LED_S1	B	P5.0 可编程 IO LCD Segment 17 LED Segment 1
19	P5.1/ SEG18/ LED_S2	B	P5.1 可编程 IO LCD Segment 18 LED Segment 2
20	P5.2/ SEG19/ LED_S3	B	P5.2 可编程 IO LCD Segment 19 LED Segment 3
21	P5.3/ SEG20/ LED_S4	B	P5.3 可编程 IO LCD Segment 20 LED Segment 4
22	P5.4/	B	P5.4 可编程 IO

	SEG21/ LED_S5		LCD Segment 21 LED Segment 5
23	P5.5/ SEG22/ LED_S6	B	P5.5 可编程 IO LCD Segment 22 LED Segment 6
24	P5.6/ SEG23/ LED_S7	B	P5.6 可编程 IO LCD Segment 23 LED Segment 7
25	P5.7/ SEG24/ LED_S8	B	P5.7 可编程 IO LCD Segment 24 LED Segment 8
26	P6.0/ SEG25/ COM8/ LED_C8	B	P6.0 可编程 IO LCD Segment 25 LCD COM 8 LED COM 8
27	P6.1/ SEG26/ COM7/ LED_C7	B	P6.1 可编程 IO LCD Segment 26 LCD COM 7 LED COM 7
28	P6.2/ SEG27/ COM6/ LED_C6	B	P6.2 可编程 IO LCD Segment 27 LCD COM 6 LED COM 6
29	P6.3/ SEG28/ COM5/ LED_C5	B	P6.3 可编程 IO LCD Segment 28 LCD COM 5 LED COM 5
30	P6.4/ SEG29/ COM4/ LED_C4	B	P6.4 可编程 IO LCD Segment 29 LCD COM 4 LED COM 4
31	P6.5/ COM3/ LED_C3	B	P6.5 可编程 IO LCD COM 3 LED COM 3
32	P6.6/ COM2/ LED_C2	B	P6.6 可编程 IO LCD COM 2 LED COM 2
33	P6.7/ COM1/ LED_C1	B	P6.7 可编程 IO LCD COM 1 LED COM 1
34	VDD	I	电源输入, 1.8~5.5V
35	GND	I	地
36	CUP2	O	Charge Pump 输出 2
37	CUP1	O	Charge Pump 输出 1 与 CUP2 之间接 0.1uF 电容
38	VLCD	O	LCD 电源, 接 1uF 电容到地

39	VDDL	O	1.8V 数字电源，内建 LDO，外接 1uF 电容到地
40	P3.3/ XIN/ FOSCK/ RXD1/ NSS1	B	P3.3 可编程 IO 快时钟晶振输入 外部快时钟输入 UART 数据接收端 SPI 选择端口
41	P3.2/ XOUT/ PPGA3/ TXD1/ MISO1	B	P3.2 可编程 IO 快时钟晶振输出 功能转移前 PWM 输出 3，配合 timer2 使用 UART 数据发射端 SPI 数据端，Master 模式输入，Slave 模式输出
42	P3.1/ SDA1/ CPA2/ PPGA2/ MOSI1	B	P3.1 可编程 IO 功能转移前 I2C 之数据端 功能转移前 Capture 输入 2，配合 timer4 使用 功能转移前 PWM 输出 2，配合 timer4 使用 功能转移前 SPI 数据端，Master 模式输出，Slave 模式输入
43	P3.0/ SCL1/ CPA1/ PPGA1/ SCLK1/ Y2D	B	P3.0 可编程 IO 功能转移前 I2C 之时钟端 功能转移前 Capture 输入 1，配合 timer3 使用 功能转移前 PWM 输出 1，配合 timer3 使用 功能转移前 SPI 时钟端 Y2D 端口
44	P3.4/ LXOUT/ SOSCK/ AN2	B	P3.4 可编程 IO 低频晶振输出，外接 32768 晶振 外部慢时钟输入 ADC 输入通道 2
45	P3.5/ LXIN/ AN3	B	P3.5 可编程 IO 低频晶振输入，外接 32768 晶振 ADC 输入通道 3
46	VDD	I	电源输入，1.8~5.5V
47	AGND	I	模拟地
48	RSTB/ Y2CK	I	片外 Reset 脚，低电平复位，内置上拉 Y2CK 端口
49	P1.0/ CCH0	B	P1.0 可编程 IO 电容触摸传感输入通道 0
50	P1.1/ CCH1/ AT0	B	P1.1 可编程 IO 电容触摸传感输入通道 1 AT0 测试端口
51	P1.2/ CCH2/ AT1	B	P1.2 可编程 IO 电容触摸传感输入通道 2 AT1 测试端口
52	P1.3/ CCH3	B	P1.3 可编程 IO 电容触摸传感输入通道 3
53	P1.4/ CCH4	B	P1.4 可编程 IO，可设置上升沿或下降沿产生中断 电容触摸传感输入通道 4

	AN0/ INT0		ADC 输入通道 0 中断 0 输入
54	P1.5/ CCH5/ AN1	B	P1.5 可编程 IO 电容触摸传感输入通道 5 ADC 输入通道 1
55	P1.6 / CCH6	B	P1.6 可编程 IO 电容触摸传感输入通道 6
56	P1.7 / CCH7	B	P1.7 可编程 IO 电容触摸传感输入通道 7
57	P2.0 / CCH8/ AN4	B	P2.0 可编程 IO 电容触摸传感输入通道 8 ADC 输入通道 4
58	P2.1 / CCH9/ AN5	B	P2.1 可编程 IO 电容触摸传感输入通道 9 ADC 输入通道 5
59	P2.2 / CCH10/ AN6	B	P2.2 可编程 IO 电容触摸传感输入通道 10 ADC 输入通道 6
60	P2.3 / CCH11/ AN7	B	P2.3 可编程 IO 电容触摸传感输入通道 11 ADC 输入通道 7
61	P2.4 / CCH12/ AN8	B	P2.4 可编程 IO 电容触摸传感输入通道 12 ADC 输入通道 8
62	P2.5 / CCH13/ AN9	B	P2.5 可编程 IO 电容触摸传感输入通道 13 ADC 输入通道 9
63	P2.6 / CCH14/ AN10	B	P2.6 可编程 IO 电容触摸传感输入通道 14 ADC 输入通道 10
64	P2.7 / CCH15/ AN11/ Vref	B	P2.7 可编程 IO 电容触摸传感输入通道 15 ADC 输入通道 11 ADC 参考电压输入或者输出

Notes:

VDD: 1.8~5.5V

VDDL: 1.8V Core voltage

VLCD: 3.0V

所有端口在复位时为数字高阻输入状态，复位完成后有配置寄存器或程序决定其端口的驱动模式。

Direction: I: Input; O: Output; B: Bi-direction;

VDD 与 VSS 之间必须要并一个 10uF 电容

2 存储器构成

YS68F9XX(X)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。YS68F9XX(X)只支持高速 12MHz 和慢速 32KHz 内部振荡，在出厂时已经被校准为 12MHz，其精度在整个温度和电压范围内为±1%。

YS68F9XX(X)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（RAM）

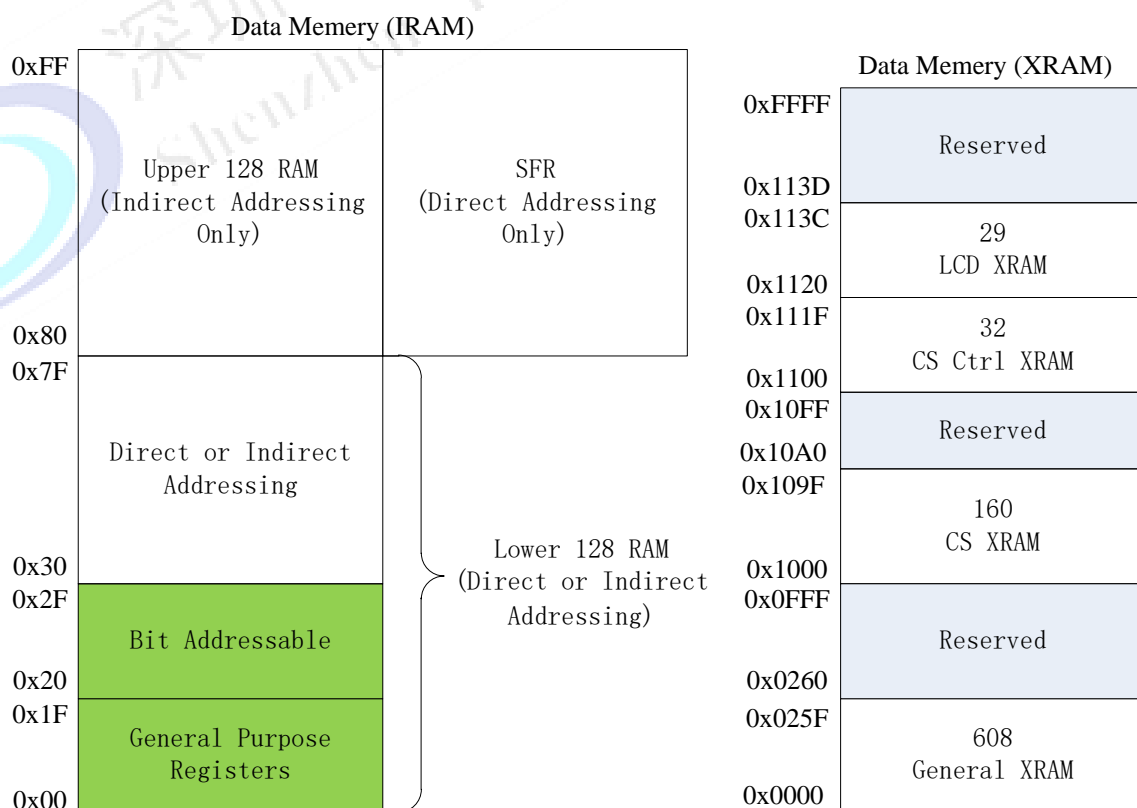
特性：

YS68F9XX(X)为数据存储提供了内部 RAM 和外部 RAM。下列为存储器空间分配：

- 低位 128 字节的 RAM（地址从 00H 到 7FH）可直接或间接寻址；
- 高位 128 字节的 RAM（地址从 80H 到 FFH）只能间接寻址；
- 特殊功能寄存器（SFR，地址从 80H 到 FFH）只能间接寻址；

高位 128 字节的 RAM 占用的地址空间和 SFR 相同，但在物理上与 SFR 的空间是分离的。当一个指令访问高于地址 7FH 的内部位置时，CPU 可根据访问的指令类型来区分是访问高位 128 字节数据 RAM 还是访问 SFR。

YS68F9XX(X)在外部数据空间额外提供了 4 块独立的 XRAM，其中 0x0000-0x025F 是通用 XRAM，供软件使用；0x1000~0x109F 是 CS XRAM（触摸章节有详细介绍）；0x1100~111F 是 CS CTRL XRAM；0x1120-0x113C 是 LCD XRAM。



● XRAM

一共有 4 块独立的 XRAM, 其中 0x0000-0x025F 是通用 XRAM, 供软件使用; 0x1000~0x109F 是 CS XRAM; 0x1100~0x111F 是 CS CTRL XRAM; 0x1120-0x113C 是 LCD XRAM。

● 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 128 字节。

● 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 0 或者 8 结尾的 SFR 既可以按字节寻址也可以按位寻址。SFR 空间中未使用的地址保留为将来使用, 请避免使用, 访问这些地址会产生不确定的结果。

2.2 特殊功能寄存器分布

地址	0(8)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)
0xF8	SPICTL					DISP_CFG	DISP_EN	CLKEY
0xF0	B	TMR4RLH	TMR4RLH	TMR4BLL	TMR4BLH		PER_EN	
0xE8	EIE1	RSTEN	CLKCF	CLKCF2				RSTSRC
0xE0	ACC	TMR3RLH	TMR3RLH	TMR3BLL	TMR3BLH			
0xD8	EIP1							
0xD0	PSW	P0DIR	P1DIR	P2DIR	P3DIR	P0PU	P1PU	PXC
0xC8	TMR2CN	P4DR	P5DR	P6DR	P3AN	P0AN	P1AN	P2AN
0xC0	IRQ0	P4DIR	P5DIR	P6DIR				MCDST
0xB8	IP	FCTR	I2C_DIN	I2C_DOUT	I2C_SLAD	I2C_STAT	RTCL	RTCH
0xB0	P3DR	TMR3CN	TMR4CN					FLKEY
0xA8	IE	ADC0L	ADC0H					ANA_CTRL
0xA0	P2DR	SPICFG	SPISCR	SPIDAT		ADC0PWR	ADC0TK	ADC0MX
0x98	SCON0	SBUF0						REF0CN
0x90	PIDR	TMR2RLH	TMR2RLH	TMR2BLL	TMR2BLH	ADC0CN	ADC0CF	ADC0AC
0x88	TCON	TMOD	TLO	TL1	TH0	TH1	CKCON	PSCTL
0x80	PODR	SP	DPL	DPH	P0SEL	WKSRC	WDTCF	PCON

2.3 Flash 程序存储器

- Flash 存储器包括 8*1KB 区块, 总共 8KB;
- 在工作电压范围内都能进行编程和擦除操作;
- 在线编程 (ICP) 操作支持写入、读取和擦除操作;
- 支持整体/扇区擦除和编程;
- 低功耗

YS68F9XX(X)有 8K 字节的 Flash, 一共有 128 个扇区, 每个扇区 64 字节。其中主程序区是 128 扇区控制器支持内部软件编程和 Y2 接口编程, 对编程时系统时钟的频率没有要求。本 Flash 可支持一次对一个扇区的编程。软件读 Flash 用 MOVX 指令, 软件擦除、编程用的是 MOVX 指令。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。

Flash 扇区擦除后, 数据是 0x00 而不是常见的 0xFF! 写 Flash 可以把数据位置 1, 但不能使数据位清 0。如果某一字节已经编程过了, 要对它重写就必须先擦除其所在扇区, 再执行写入操作。

写和擦除均由硬件自动定时, 以确保操作正确, 无须进行数据查询来操作的完成与否。

扇区擦除步骤:

- 禁止中断;
- 置 FLA_SEC (PSCTL.1) 和 FLA_WEN (PSCTL.0) 为 1, 以允许软件扇区擦除;
- 顺序往 FLKEY 写 0xA5 和 0xF1, 进入开锁状态;

- 4、加入五条 nop 指令；
- 5、向待擦除扇区的任一地址写任意值；
- 6、向 FLA_ACT (PSCTL.2) 写 1，使 Flash 进入擦除状态；
- 7、清除 FLA_SEC 禁止擦除操作。
- 8、重新使能中断。

注：

YS68F9XXX(X)扇区擦除后，数据是 0x00 而不是常见的 0xFF。

FLASH 写步骤：

- 1、禁止中断；
- 2、清“0”FLA_SEC (PSCTL.1)，置“1”FLA_WEN (PSCTL.0)；
- 3、顺序往 FLKEY 写 0xA5 和 0xF1，进入开锁状态；
- 4、加入五条 nop 指令；
- 5、往 FLASH 的目标地址写入数据（多个数据可以连续写入）；
- 6、向 FLA_ACT (PSCTL.2) 写 1，使 Flash 进入编程状态
- 7、清除 FLA_WEN 禁止 Flash 写；
- 8、重新使能中断。

注意：

- 1、Flash 写期间禁止中断,以避免出错
- 2、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。
- 3、要开锁一定要先写 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，则此功能将冻结，直到下一次系统复位！
- 4、程序过程出现跨页时，硬件会记录在 PSCTL.3。当出现这种情况时，FLASH 控制器将放弃这一操作，故用户程序应检查这一位。
- 5、在已经开锁的状态不要往 FLKEY 写数据，否则会进入冻结状态。如果要重新上锁，则需要往执行一次 PSCTL 写操作，可以是任何数值。
- 6、如果出现跨页，会造成整个程序丢失。

2.4 编程控制寄存器 (PSCTL)

寄存器 8FH：编程控制寄存器 (PSCTL)

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	FLA_ACT	FLA_SEC	FLA_WEN
bit7						bit0	

图注：

R = 可读位 W = 可写位 U = 未实现位，读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7-3 未实现：读为0

bit2 FLA_ACT：空寄存器位
 1 = 开始Flash操作，如编程，擦除
 0 = 无效，读永远是0

bit1 FLA_SEC：扇区擦除使能位
 1 = 使能扇区擦除
 0 = 禁止扇区擦除

bit0 FLA_WEN: 编程使能位
 1 = 使能扇区编程
 0 = 禁止扇区编程
 只有在FLA_WEN为高时, FLA_SEC才起作用

2.5 FLASH 编程开锁寄存器 (FLKEY)

寄存器 B7H: FLASH编程开锁寄存器 (FLKEY)

R/W	R/W	R/W	R/W	R/W	R/W-0	R/W-0	R/W-0
FLKEY							
bit7							bit0

图注:

R = 可读位 W = 可写位 U = 未实现位, 读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7-0 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!
 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

2.6 扇区和地址的关系

0	0x0000 – 0x003f	64	0x1000 – 0x103f
1	0x0040 – 0x007f	65	0x1040 – 0x107f
2	0x0080 – 0x00bf	66	0x1080 – 0x10bf
3	0x00c0 – 0x00ff	67	0x10c0 – 0x10ff
4	0x0100 – 0x013f	68	0x1100 – 0x113f
5	0x0140 – 0x017f	69	0x1140 – 0x117f
6	0x0180 – 0x01bf	70	0x1180 – 0x11bf
7	0x01c0 – 0x01ff	71	0x11c0 – 0x11ff
8	0x0200 – 0x023f	72	0x1200 – 0x123f
9	0x0240 – 0x027f	73	0x1240 – 0x127f
10	0x0280 – 0x02bf	74	0x1280 – 0x12bf
11	0x02c0 – 0x02ff	75	0x12c0 – 0x12ff
12	0x0300 – 0x033f	76	0x1300 – 0x133f
13	0x0340 – 0x037f	77	0x1340 – 0x137f
14	0x0380 – 0x03bf	78	0x1380 – 0x13bf
15	0x03c0 – 0x03ff	79	0x13c0 – 0x13ff
16	0x0400 – 0x043f	80	0x1400 – 0x143f
17	0x0440 – 0x047f	81	0x1440 – 0x147f
18	0x0480 – 0x04bf	82	0x1480 – 0x14bf
19	0x04c0 – 0x04ff	83	0x14c0 – 0x14ff

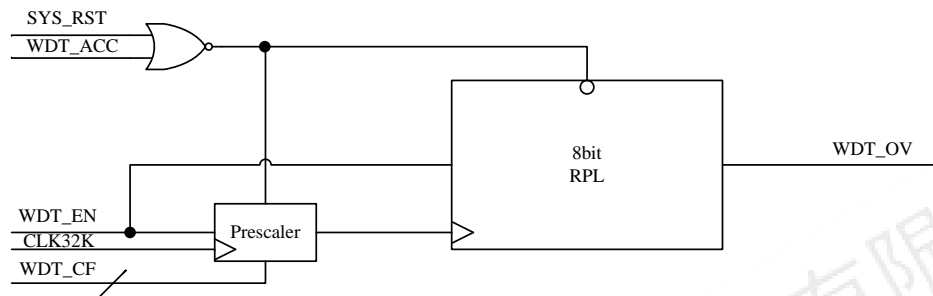
20	0x0500 – 0x053f	84	0x1500 – 0x153f
21	0x0540 – 0x057f	85	0x1540 – 0x157f
22	0x0580 – 0x05bf	86	0x1580 – 0x15bf
23	0x05c0 – 0x05ff	87	0x15c0 – 0x15ff
24	0x0600 – 0x063f	88	0x1600 – 0x163f
25	0x0640 – 0x067f	89	0x1640 – 0x167f
26	0x0680 – 0x06bf	90	0x1680 – 0x16bf
27	0x06c0 – 0x06ff	91	0x16c0 – 0x16ff
28	0x0700 – 0x073f	92	0x1700 – 0x173f
29	0x0740 – 0x077f	93	0x1740 – 0x177f
30	0x0780 – 0x07bf	94	0x1780 – 0x17bf
31	0x07c0 – 0x07ff	95	0x17c0 – 0x17ff
32	0x0800 – 0x083f	96	0x1800 – 0x183f
33	0x0840 – 0x087f	97	0x1840 – 0x187f
34	0x0880 – 0x08bf	98	0x1880 – 0x18bf
35	0x08c0 – 0x08ff	99	0x18c0 – 0x18ff
36	0x0900 – 0x093f	100	0x1900 – 0x193f
37	0x0940 – 0x097f	101	0x1940 – 0x197f
38	0x0980 – 0x09bf	102	0x1980 – 0x19bf
39	0x09c0 – 0x09ff	103	0x19c0 – 0x19ff
40	0x0a00 – 0x0a3f	104	0x1a00 – 0x1a3f
41	0x0a40 – 0x0a7f	105	0x1a40 – 0x1a7f
42	0x0a80 – 0x0abf	106	0x1a80 – 0x1abf
43	0x0ac0 – 0x0aff	107	0x1ac0 – 0x1aff
44	0x0b00 – 0x0b3f	108	0x1b00 – 0x1b3f
45	0x0b40 – 0x0b7f	109	0x1b40 – 0x1b7f
46	0x0b80 – 0x0bbf	110	0x1b80 – 0x1bbf
47	0x0bc0 – 0x0bff	111	0x1bc0 – 0x1bff
48	0x0c00 – 0x0c3f	112	0x1c00 – 0x1c3f
49	0x0c40 – 0x0c7f	113	0x1c40 – 0x1c7f
50	0x0c80 – 0x0cbf	114	0x1c80 – 0x1cbf
51	0x0cc0 – 0x0cff	115	0x1cc0 – 0x1cff
52	0x0d00 – 0x0d3f	116	0x1d00 – 0x1d3f
53	0x0d40 – 0x0d7f	117	0x1d40 – 0x1d7f
54	0x0d80 – 0x0dbf	118	0x1d80 – 0x1dbf
55	0x0dc0 – 0x0dff	119	0x1dc0 – 0x1dff
56	0x0e00 – 0x0e3f	120	0x1e00 – 0x1e3f
57	0x0e40 – 0x0e7f	121	0x1e40 – 0x1e7f
58	0x0e80 – 0x0ebf	122	0x1e80 – 0x1ebf
59	0x0ec0 – 0x0eff	123	0x1ec0 – 0x1eff
60	0x0f00 – 0x0f3f	124	0x1f00 – 0x1f3f
61	0x0f40 – 0x0f7f	125	0x1f40 – 0x1f7f
62	0x0f80 – 0x0fbf	126	0x1f80 – 0x1fbf

63	0x0fc0 – 0x0fff	127	0x1fc0 – 0x1fff
----	-----------------	-----	-----------------

2.7 看门狗定时器（WDT）

YS68F9XX(X)内置一个 16 位的看门狗计数器，其中 8 位拆分为预分频计数，另外 8 位是主计数器，由行波计数器实现进一步降低功耗。它是一个递减计数器，由使用内部慢时钟作为时钟源，当它使能时，主计数器从 0xFF 计数到 0x00 时产生溢出，将系统复位。

看门狗模块图



2.7.1 WDTCF 寄存器

寄存器 86H: WDTCF寄存器

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
					WDTCF2	WDTCF1	WDTCF0
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-3 未实现：读为0

bit2:0 WDTCF<2:0>: WDT分频比选择位

000 = 1:1

001 = 1:4

010 = 1:8

011 = 1:16

100 = 1:32

101 = 1:64

110 = 1:128

111 = 1:256

对 WDTCF 的读或写均可清看门狗预分频器和计数器。

注:

1. 当软件向该寄存器执行读/写操作时，就会产生一个清零信号，将看门狗复位；
2. 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；
3. 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；

4. 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。
5. 在系统时钟使用快时钟的情况下，如果同时打开溢出中断和溢出复位，则在中断处理程序中，不能立即禁用看门狗。否则无法发生溢出复位。这是因为慢时钟还没有来得及处理溢出信号来产生复位，快时钟就已经把溢出信号拉低了。解决的办法是，要么禁止溢出中断，要么在禁用看门狗之前，先延迟一段时间(一个慢时钟周期的时间)。
6. WDT 的使能在烧录的时候由 OPTION 选择。

3 系统时钟和振荡器

3.1 概述

特性：

- 内外时钟切换
- 快慢时钟切换
- 晶体振荡模式下的时钟缺失检测模式
- 时钟门控

YS68F9XX(X)内部有两个时钟源，一个 12MHz 可编程精密快时钟振荡器和一个 32.768KHz 慢时钟振荡器，可选择内部或外部时钟。系统复位后默认选择内部快时钟，外部快慢时钟分为晶体振荡模式和外灌模式。

3.2 时钟缺失检测模式

为了增强系统的可靠性，芯片内置了一个时钟监控模块。开启这个模块（MCD_EN 置 1），如果外部时钟为系统主时钟并且停振，系统将自动切换到内部 RC 时钟，如果外部 32K 时钟停止振动，将产生中断（标志位 MCD_IRQ 置 1）告知 CPU。系统处于 STOP 或 LDLE 模式时，可通过时钟检测中断唤醒，防止系统死机。

3.3 时钟切换流程

时钟切换可以划分为两大类：一是快时钟和慢时钟的切换，二是内部时钟和外部时钟的切换。支持各种快慢、内外时钟之间的切换，内部设计保证了这个操作的安全性，不会导致意外死机。

下面举例从内部快时钟切换到外部慢时钟：

```

...           ;外部管脚为时钟输入
...           ;外部灌入慢时钟
MOV A, CLKCF  ;暂存 CLKCF
ORL A, 0x10   ;选择外部慢时钟
ORL A, 0x01   ;把 CKSEL 置 1
MOV CLKCF, A  ;进行内部快时钟到外部慢时钟的切换

```

3.4 快慢切换

由快切换到慢：

1. 使能慢时钟（如果使用外部慢时钟，则置相关引脚为时钟输入）；
2. 切换到慢时钟（CKSEL=1）；
3. 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 CK12M=1；如果使用外部快时钟，则停止时钟输入）

由慢切换到快：

1. 使能快时钟（如果使用内部快时钟，置 CK12M=0；如果使用外部快时钟，则置相关引脚为时钟输入）；

2. 切换到快时钟 (CKSEL=0);

3.5 由外部时钟切换到内部时钟

1. 使能内部时钟, CK12M=0;
2. 切换到内部时钟, FCK_EXT_EN/SCK_EXT_EN=0;
3. 过了两个外部时钟下降沿之后, 系统时钟切换到内部时钟;
4. 停止外部时钟输入; (可选)

由内部时钟切换到外部时钟

1. 设置外部引脚为时钟输入模式, 灌入相应频率的时钟;
2. 切换外部时钟, FCK_EXT_EN/SCK_EXT_EN=1;
3. 关闭内部快时钟, CK12M=1; (可选)

3.6 切换到外部时钟注意

如果需要确保程序段一定要工作在外部时钟, 可以通过判断 SYS_IND (CLKCF.1 位), 它为 1 时表示已经切换到外部时钟。时钟分配和控制由模块 CLKCF 实现, 主要涉及到 CPU、中断控制器及外设自身。所有外设由 CLKCF 统一分配。它有以下特点:

当 CPU 处于非正常模式, CPU 时钟就会被停止;

其它外设则取决于低功耗模式和该外设本身的配置、工作状态;

处于在线仿真的暂停状态时, 所有 timer 的时钟停止, RTC 时钟停止; LCD/LED、UART (由于 TIMER1 的停止, UART 通信受影响)、ADC、SPI 不受影响。

3.7 时钟源选择寄存器

CLKCF 地址为 0xEA。通过它可实现内外时钟和系统时钟的选择

位	名字	复位值	功能
7	NA	NA	保留位, 读 0
6	CK12M	0	内部快时钟使能, 低有效 0: 启用内部快时钟 1: 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=0 并 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清零 CK12M 而导致的死机。
5	FCK_EXT_EN	0	12MHz 快时钟源选择 1: 选择外部快时钟 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 0=选择内部慢时钟
3			
2	MCD_EN	0	MCD 模块控制 1=使能 MCD 模块, MCD 事件发生后发出中断请求 0=禁止
1	SYS_IND	0	只读, 系统时钟指示位 内、外时钟切换完成与否的标志 0, 表示系统时钟是内部时钟 1, 表示系统时钟是外部时钟
0	CKSEL	0	系统时钟选择 读写: 0 选择快时钟, 1 选择慢时钟

寄存器 EAH: 时钟源选择寄存器 (CLKCF)

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-0	R/W-0
	CK12M	FCK_EXT_EN	SCK_EXT_EN		MCD_EN	SYS_IND	CKSEL
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7 未实现: 读为0

bit6 CK12M: 内部快时钟使能位

0 = 使能内部快时钟

1 = 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=0并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清零CK12M而导致的死机。

bit5 FCK_EXT_EN: 12MHz 快时钟源选择位

1 = 选择外部快时钟

0 = 选择内部快时钟

bit4 SCK_EXT_EN: 32KHz 慢时钟源选择位

1 = 选择外部慢时钟

0 = 选择内部慢时钟

bit3 未实现: 读为0

bit2 MCD_EN: MCD 模块控制位

1 = 使能MCD模块, MCD事件发生后发出中断请求

0 = 禁止MCD模块

bit1 SYS_IND: 系统时钟指示位

1 = 表示系统时钟是外部时钟

0 = 表示系统时钟是内部时钟

bit0 CKSEL: 系统时钟选择位

1 = 选择慢时钟

0 = 选择快时钟

3.8 谐振器负载电容选择

晶体谐振器		
频率	C1	C2
32.768KHZ	20 pf	20 pf
4MHZ	10 pf	10 pf
12MHZ	8 pf	8 pf

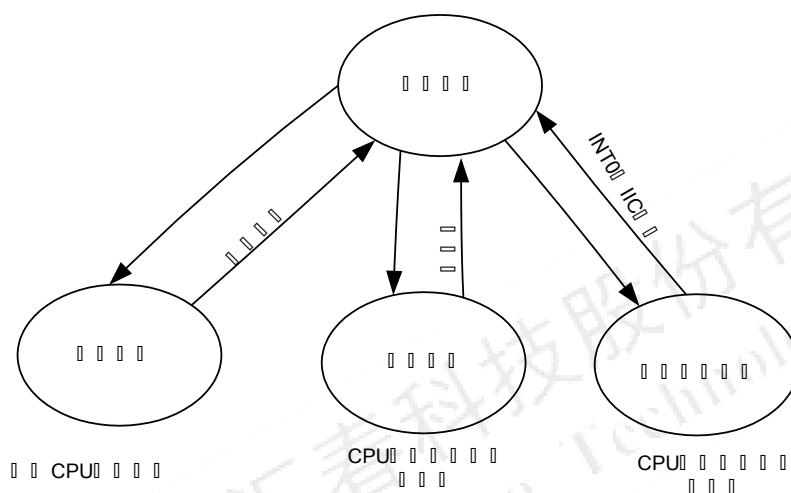
注意:

- 表中负载电容仅供参考!
- 以上电容值可通过谐振器基本的起振和运行测试, 并非最优值。

4 功耗模式

YS68F9XX(X)有 4 种功耗模式，分别是正常、待机、睡眠和深度睡眠。

各模式转换图：



各种功耗模式下的模块工作情况总结如下：

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 12MIPS 速度
待机	CPU 停止，其他功能模块关闭或工作，快慢时钟源均打开，Flash 待机	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，Flash 处于关机模式，其他功能模块关闭或工作，快时钟关闭（但若当前自容模块在工作时，快时钟不会立即关闭，待自容扫描结束，片内快时钟才会关闭），慢时钟打开	外部/IO 中断 RTC 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	快慢时钟关闭（但若当前自容模块或 ADC 在工作时，快慢时钟不会立即关闭，待自容扫描或 ADC 转换结束，片内快慢时钟才会关闭）	I2C 中断 外部/IO 中断 外部/软件复位	功耗最低

功耗模式描述

小结：

- 睡眠模式下，如果选择快时钟作为系统时钟源，Timer1/2/3/4 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；
- 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。
- 关于中断唤醒，通过配置 WKSRC，外部中断可以在没有使能总中断允许位 IE.7 和相关的独立中断使能位的情况下，唤醒 CPU。其它中断源则必须通过中断才能唤醒。

4.1 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 12MIPS。

4.2 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，而不是 IDLE 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。

任何中断标志的置起均可结束待机模式，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，看门狗计数溢出后复位也会唤醒 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。

4.3 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 TIMERS，WDT，UART 等功能模块都将停止工作，而且只能通过外部复位唤醒。

YS68F9XXX(X)的睡眠模式的进入方式跟标准的 80C51 兼容，但外设的工作状态、唤醒方式又有所不同，WDT，RTC 可以工作。如果系统时钟选择了慢时钟，Timers 也可以工作。

外部中断/I2C，RTC 或者是看门狗的溢出复位可以唤醒。

Flash 处于关机模式，典型情况下消耗电流约 1uA

4.4 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；内部快慢时钟源关闭，晶振电路不工作。

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

Flash 处于关机模式，典型情况下消耗电流约 1uA。

4.5 功耗模式选择寄存器

PCON 地址为 0x87。

位	名字	复位值	功能
7	NA	NA	保留，只读 0
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 1:0}			功耗模式
	1	x	深睡眠
	0	1	浅睡眠
	0	0	待机
	0	0	正常

寄存器 87H: 功耗模式选择寄存器 (PCON)							
U-0	R-0	R-0	R/W-0	R-0	R-0	R/W-0	R/W-0
—	GF3	GF2	DEEPPD	GF1	GF0	STOP	IDLE
bit7						bit0	
图注:							
R = 可读位		W = 可写位		U = 未实现位, 读为0		x = 未知	
-n = POR时的值		1 = 置1		0 = 清零			

bit7 未实现: 读为0

bit6 GF3: 通用标志位3
1 =
0 =

bit5 GF2: 通用标志位 2
1 =
0 =

bit4 DEEPPD: 深度睡眠模式选择位
1 = 芯片进入深度睡眠模式
0 = 唤醒后由硬件清0

bit3 GF1: 通用标志位 1
1 =
0 =

bit2 GF0: 通用标志位 0
1 =
0 =

bit1 STOP: 睡眠模式选择位
1 = 芯片进入睡眠模式
0 = 唤醒后由硬件清 0

bit0 IDLE: 待机模式选择位
1 = 芯片进入待机模式
0 = 唤醒后由硬件清0

{PCON 4, PCON 1:0}				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

注意:

- 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。
- 2) 此寄存器的第 5、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, 故这 4 位的读出来的值永远是 0。

5 复位源

5.1 概述

YS68F9XX(X)有七个复位源:

- 外部复位
- 上电复位
- 低电压侦测复位
- 过度电应力复位
- 看门狗溢出复位
- 软复位

复位标志可查询, 记录在 0xEF 寄存器里面。最近一次的复位会把相关的位置 1, 把其他各位清 0。模拟复位源 (外部/上电、低电压、过度电应力复位) 标志互斥, 数字的复位源 (WDT、FEDR、SOFTTRST) 标志也是互斥的。

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, 数字的复位事件不会把模拟复位标志位清掉。软件可以通过对 RSTEN 的写访问来把模拟复位标志清除。

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, 那么所有数字复位事件标志位会被清 0。

5.1 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。

同样, 芯片在上电过程中模拟电路也会向把 POR 置起, 启动复位。

5.2 低电压侦测复位

RSTEN.3 置为高时将允许 VDD 监测电路。如果 VDD 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。

5.3 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。

5.4 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.1 为 1), 侦测电路会发出复位信号。

5.5 软复位

通过设置相关寄存器，可以在程序控制下发出复位指令。

5.6 复位状态寄存器（RSTSRC）

寄存器 EFH: 复位状态寄存器（RSTSRC）							
R-	R-	R-	R-	R-	R-	R-	R-
EXTR	EOSR	VDD_STAT	FEDR	WDTR	SOFTR	POR	PWR_FAIL
bit7				bit0			
图注： R = 可读位 -n = POR时的值							
W = 可写位 1 = 置1		U = 未实现位，读为0 0 = 清零			x = 未知		

- bit7 EXTR: 外部复位状态位
1 = 发生过外部复位
0 = 未发生过外部复位
- bit6 EOSR: 过度电应力复位状态位
1 = 发生了电过度应力复位
0 = 未发生了电过度应力复位
- bit5 VDD_STAT: VDD 状态位
1 = 处于VDD状态
0 = 未处于VDD状态
- bit4 FEDR: 非法操作 MOVX 状态位
1 = 此前发生了MOVX的非法操作，引发了复位
0 = 此前并没有MOVX的非法操作，清零
- bit3 WDTR: 看门狗溢出复位状态位
1 = 发生了看门狗溢出复位
0 = 未发生看门狗溢出复位
- bit2 SOFTR: 软件复位状态位
1 = 发生过软件复位
0 = 未发生软件复位
- bit1 POR: 上电复位状态位
1 = 发生了软件复位
0 = 未发生软件复位
- bit0 PWR_FAIL: 掉电复位状态位
1 = 曾经发生了掉电复位
0 = 未发生掉电复位

5.7 复位源使能寄存器（RSTEN）

RSTEN 地址为 0xE9 复位源使能寄存器。它实现对 EOS、LVD 和 BOR 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。

寄存器 E9H: 复位源使能寄存器（RSTEN）

它实现对EOS、LVD和BOR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。

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	LVD_PD	EOS_GATE	EOS_RST_EN	BOR_PD
bit7				bit0			
图注： R = 可读位 -n = POR时的值				W = 可写位 1 = 置1		U = 未实现位，读为0 0 = 清零	
						x = 未知	

bit7-4 未实现：读为0

bit3 LVD_PD: LVD 模块使能位
1 = LVD 模块关闭
0 = LVD模块使能

bit2 EOS_GATE: EOS 事件门控时钟使能位
1 = 使能EOS事件门控时钟
0 = 禁止EOS事件门控时钟

bit1 EOS_RST_EN: EOS 复位使能位
1 = 允许EOS发出复位
0 = 禁止EOS发出复位

bit0 BOR_PD: BOR 模块使能位
1 = BOR模块关闭
0 = BOR模块使能

6 IO 输入/输出端口

6.1 概述

特性：

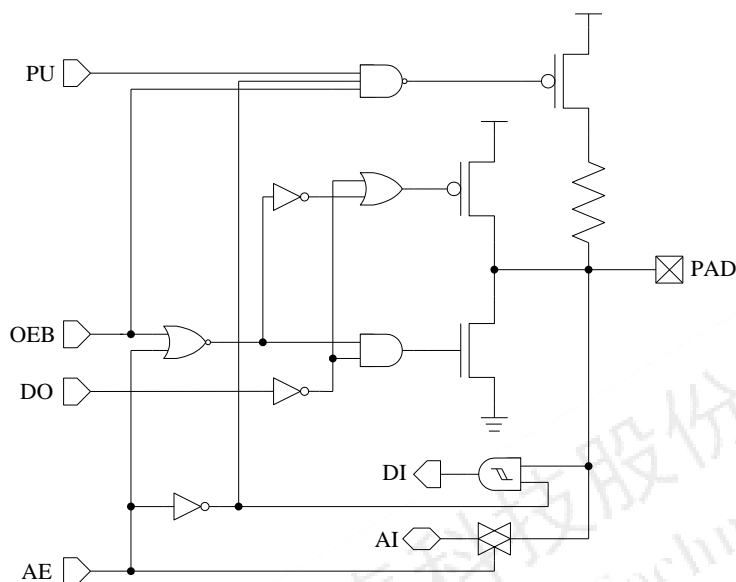
- 54 个双向 I/O 端口

■ I/O 端口可与其它功能共用

YS68F9XX(X)提供最多 54 个位可编程双向 I/O 端口，端口数据在寄存器 PxDR 中。每个 I/O 都有内部上拉电阻。端口控制寄存器（PxDIR）控制端口是作为输入或输出。

有些引脚具有复用功能,并且复用功能可以转移。当所有功能都允许时，在 CPU 中存在优先权以避免功能冲突。

IO 等效电路图：



寄存器名	偏移地址	有效位宽	复位值	说明
FCTR	0XB9		0X00	功能控制寄存器
P0PU	0XD5		0X00	P0 端口上拉控制寄存器
P1PU	0XD6		0X00	P1 端口上拉控制寄存器
PXC	0XD7		0X00	PX 端口（上拉）控制寄存器
P0AN	0XCD		0X00	P0 端口数字/模拟模式控制寄存器
P1AN	0XCE		0X00	P1 端口数字/模拟模式控制寄存器
P2AN	0XCF		0X00	P2 端口数字/模拟模式控制寄存器
P3AN	0XCC		0X00	P3 端口数字/模拟模式控制寄存器
P0DIR	0XD1		0XFF	P0 端口方向控制寄存器
P1DIR	0XD2		0XFF	P1 端口方向控制寄存器
P2DIR	0XD3		0XFF	P2 端口方向控制寄存器
P3DIR	0XD4		0X3F	P3 端口方向控制寄存器
P4DIR	0XC1		0XFF	P4 端口方向控制寄存器
P5DIR	0XC2		0XFF	P5 端口方向控制寄存器
P6DIR	0XC3		0XFF	P6 端口方向控制寄存器

P0DR	0X80		0XFF	P0 数据寄存器
P1DR	0X90		0XFF	P1 数据寄存器
P2DR	0XA0		0XFF	P2 数据寄存器
P3DR	0XB0		0X3F	P3 数据寄存器
P4DR	0XC9		0XFF	P4 数据寄存器
P5DR	0XCA		0XFF	P5 数据寄存器
P6DR	0XCB		0XFF	P6 数据寄存器

6.2 功能控制寄存器（FCTR）

寄存器 B9H：功能控制寄存器（FCTR）

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	UARTOIR	TIMER4FTR	TIMER3FTR	TIMER2FTR	UARTFTR	SPIFTR	I2CFTR
bit7							bit0

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7 未实现：读为0

bit6 **UARTOIR**：自动设置转换端口类型位

1 = 自动转换端口类型

0 = 不自动转换端口类型

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。

bit5 **TIMER4FTR**：TIMER4功能转移位

1 = TIMER4功能转移

0 = TIMER4功能不转移

控制 **TIMER4FTR** 功能是否转移。**TIMER4** 功能默认在 PAD43/P3.1 上，功能转移以后，在
PAD5/P0.3。

bit4 **TIMER3FTR**：TIMER3 功能转移位

1 = TIMER3功能转移

0 = TIMER3功能不转移

控制 TIMER3FTR 功能是否转移。TIMER3 功能默认在 PAD44/P3.0 上，功能转移以后，在 PAD4/P0.2。

bi3 TIMER2FTR: 指令错误复位状态位

1 = TIMER2功能转移

0 = TIMER2功能不转移

控制TIMER2FTR功能是否转移。TIMER2功能默认在PAD42/P3.2上，功能转移以后，在PAD6/P0.4。

bit2 UARTFTR: UART 功能转移位

1 = Uart功能转移

0 = Uart功能不转移

控制 Uart 功能是否转移。Uart 功能默认在 PAD42/P3.2 和 PAD41/P3.3 上，功能转移以后，分别在 PAD4/P0.2 和 PAD5/P0.3 上。

bit1 SPIFTR: SPI功能转移位

1 = SPI功能转移

0 = SPI功能不转移

控制 SPI 功能是否转移。SPI 功能默认在 PAD44/P3.0、PAD43/P3.1、PAD42/P3.2、PAD41/P3.3 上，功能转移以后，分别在 PAD6/P0.4、PAD7/P0.5、PAD8/P0.6、PAD9/P0.7 上。

bit0 I2CFTR: I2CFTR功能转移位

1 = I2C功能转移

0 = I2C功能不转移

控制 I2C 功能是否转移。I2C 功能默认在 PAD43/P3.1、PAD44/P3.0 上，功能转移以后，分别在 PAD2/P0.0 和 PAD3/P0.1 上。

6.3 上拉控制寄存器

6.3.1 P0 端口上拉控制寄存器 (P0PU)

寄存器 D5H: P0端口上拉控制寄存器 (P0PU)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P0PU<7:0>							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 P0PU<7:0> P0端口上拉使能位：P0PU[7:0]分别对应P0[7:0]的上拉使能。
 0=上拉禁止
 1=上拉使能
 注：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。

6.3.2 P1 端口上拉控制寄存器（P1PU）

寄存器 D6H：P1端口上拉控制寄存器（P1PU）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1PU<7:0>							
bit7				bit0			

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 P0PU<7:0> P0端口上拉使能位：P1PU[7:0]分别对应P1[7:0]的上拉使能。
 0=上拉禁止
 1=上拉使能
 注：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。

6.3.3 P2 端口上拉控制寄存器（P2PU）

寄存器 DFH：P2端口上拉控制寄存器（P2PU）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P2PU<7:0>							
bit7				bit0			

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 P2PU<7:0> P2端口上拉使能位：P2PU[7:0]分别对应P2[7:0]的上拉使能。

0=上拉禁止

1=上拉使能

注：当P2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P2DIR.x & P2PU.x=1），否则内部上拉不起作用。

6.3.4 PX 端口控制寄存器（PXC）

寄存器 D7H: PX端口控制寄存器（PXC）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OD_DRV	HDRV	COMPU	P2PU	P3PU[3:0]			
bit7				bit0			

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

- bit7 OD_DRV: P3[1:0]的 Open Drain 模式控制
1 = Open Drain模式，此模式下IO将会自动使用增强电流输出模式
0 = 推挽模式，此模式下IO的输出电流能力由HDRV决定
- bit6 HDRV: IO 输出电流驱动模式，适用于所有 IO
1 = 增强电流输出模式
0 = 正常电流输出模式
- bit5 COMPU: P3[5:4]、P4[7:0]、P5[7:0]、P6[7:0] 公共上拉控制使能位
1 = 上拉使能
0 = 上拉禁止
- bit4 P2PU: P2[7:0]公共上拉控制使能位
1 = 上拉使能
0 = 上拉禁止
- bit3-0 P3PU [3:0]: P3[3:0]端口上拉使能位，P3PU[3:0]分别对应 P3[3:0]的上拉使能。
1 = 上拉使能
0 = 上拉禁止

6.4 模拟控制寄存器

6.4.1 P0 模式控制寄存器（P0AN）

寄存器 CDH: P0模式控制寄存器（P0AN）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P0AN <7:0>							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 P0AN <7:0> P0端口模拟模式控制: P0AN[7:0]分别对应P0[7:0]的模拟模式控制。

0 = 数字模式

1 = 模拟模式

注:

P0[7:2]端口配置成模拟模式后, 只能用于 LCD 输出。

P0[1:0]端口配置成模拟模式后, 可用于LCD输出或者ADC采样输入, 由FCTR寄存器的SEG_AD位控制, SEG_AD为0时为LCD的SEG模式, SEG_AD为1时为ADC的采样通道。

6.4.2 P1 模式控制寄存器 (P1AN)

寄存器 CEH: P1模式控制寄存器 (P1AN)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1AN <7:0>							
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 P1AN <7:0> P0端口模拟模式控制: P0AN[7:0]分别对应P0[7:0]的模拟模式控制。

0 = 数字模式

1 = 模拟模式

注:

P0[7:2]端口配置成模拟模式后, 只能用于 LCD 输出。

P0[1:0]端口配置成模拟模式后, 可用于LCD输出或者ADC采样输入, 由FCTR寄存器的SEG_AD位控制, SEG_AD为0时为LCD的SEG模式, SEG_AD为1时为ADC的采样通道。

6.4.3 P2 模式控制寄存器 (P2AN)

寄存器 CFH: P2模式控制寄存器 (P2AN)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P2AN <7:0>							
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 P2AN <7:0> P2端口模拟模式控制: P2AN[7:0]分别对应P2[7:0]的模拟模式控制。

0 = 数字模式

1 = 模拟模式

注：P2[7:0]配成模拟模式后，如果其触摸通道的 MASK 为 1 则为触摸通道，否则为 ADC 的通道。

6.4.4 P3 模式控制寄存器 (P3AN)

寄存器 CCH: P3模式控制寄存器 (P3AN)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	P3AN <7:0>					
bit7						bit0	

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-6 保留位, 只读, 读为0。

bit5-4 P3AN <7:0> P3端口模拟模式控制: P3AN[5:4]分别对应P3[5:4]的模拟模式控制。

0 = 数字模式

1 = 模拟模式

bit3-0 保留位, 只读, 读为 0。

6.5 端口方向控制寄存器

6.5.1 P0 端口方向控制寄存器 (P0DIR)

寄存器 D1H: P0端口方向控制寄存器 (P0DIR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P0DIR <7:0>							
bit7						bit0	

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 P0端口数字输入、输出方向控制位: P0DIR[7:0]分别对应P0[7:0]的方向控制。

0 = 输出

1 = 输入

6.5.2 P1 端口方向控制寄存器 (P1DIR)

寄存器 D2H: P1端口方向控制寄存器 (P1DIR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1DIR <7:0>							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 P1端口数字输入、输出方向控制位: P1DIR[7:0]分别对应P1[7:0]的方向控制。

0 = 输出

1 = 输入

6.5.3 P2 端口方向控制寄存器 (P2DIR)

寄存器 D3H: P2端口方向控制寄存器 (P2DIR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P2DIR <7:0>							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 P2端口数字输入、输出方向控制位: P2DIR[7:0]分别对应P2[7:0]的方向控制。

0 = 输出

1 = 输入

6.5.4 P3 端口方向控制寄存器 (P3DIR)

寄存器 D4H: P3端口方向控制寄存器 (P3DIR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	P3DIR <7:0>					
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-6 保留位, 只读, 读为0。

Bit5-0 P3端口数字输入、输出方向控制位: P3DIR[5:0]分别对应P3[5:0]的方向控制。

0 = 输出

1 = 输入

6.5.5 P4 端口方向控制寄存器 (P4DIR)

寄存器 C1H: P4端口方向控制寄存器 (P4DIR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P4DIR <7:0>							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

Bit7-0 P4端口数字输入、输出方向控制位: P4DIR[7:0]分别对应P4[7:0]的方向控制。

0 = 输出

1 = 输入

6.5.6 P5 端口方向控制寄存器 (P5DIR)

寄存器 C2H: P5端口方向控制寄存器 (P5DIR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P5DIR <7:0>							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

Bit7-0 P5端口数字输入、输出方向控制位: P5DIR[7:0]分别对应P5[7:0]的方向控制。

0 = 输出

1 = 输入

6.5.7 P6 端口方向控制寄存器 (P6DIR)

寄存器 C3H: P6端口方向控制寄存器 (P6DIR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P6DIR <7:0>							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

Bit7-0 P6端口数字输入、输出方向控制位: P6DIR[7:0]分别对应P6[7:0]的方向控制。

0 = 输出
1 = 输入

6.6 数据寄存器

6.6.1 P0 数据寄存器 (P0DR)

寄存器 80H: P0数据寄存器 (P0DR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P0DR <7:0>							
bit7				bit0			

图注:

R = 可读位 W = 可写位 U = 未实现位, 读为0
-n = POR时的值 1 = 置1 0 = 清零 x = 未知

Bit7-0 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。

6.6.2 P1 数据寄存器 (P1DR)

寄存器 90H: P1数据寄存器 (P1DR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1DR <7:0>							
bit7				bit0			

图注:

R = 可读位 W = 可写位 U = 未实现位, 读为0
-n = POR时的值 1 = 置1 0 = 清零 x = 未知

Bit7-0 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。

6.6.3 P2 数据寄存器 (P2DR)

寄存器 A0H: P2数据寄存器 (P2DR)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P2DR <7:0>							
bit7				bit0			

图注:

R = 可读位 W = 可写位 U = 未实现位, 读为0
-n = POR时的值 1 = 置1 0 = 清零 x = 未知

Bit7-0

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。

6.6.4 P3 数据寄存器（P3DR）

寄存器 B0H: P3数据寄存器（P3DR）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	P3DR <5:0>					
bit7						bit0	

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

Bit7-0

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[5:0]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[5:0]输出。

6.6.5 P4 数据寄存器（P4DR）

寄存器 C9H: P4数据寄存器（P4DR）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P4DR <7:0>							
bit7						bit0	

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

Bit7-0

端口4数据寄存器

将端口 4 设置为通用输入端口时，此寄存器保存端口 P4[7:0]上的输入数据；将端口 4 设置为通用输出端口时，向此寄存器写入数据，则数据会从 P4[7:0]输出。

6.6.6 P5 数据寄存器（P5DR）

寄存器 CAH: P5数据寄存器（P5DR）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P5DR <7:0>							
bit7						bit0	

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

1 = 置1

0 = 清零

x = 未知

-n = POR时的值

Bit7-0 端口5数据寄存器

将端口 5 设置为通用输入端口时，此寄存器保存端口 P5[7:0]上的输入数据；将端口 5 设置为通用输出端口时，向此寄存器写入数据，则数据会从 P5[7:0]输出。

6.6.7 P6 数据寄存器（P6DR）

寄存器 CBH: P6数据寄存器（P6DR）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P6DR <7:0>							
bit7				bit0			

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

Bit7-0 端口6数据寄存器

将端口 6 设置为通用输入端口时，此寄存器保存端口 P6[7:0]上的输入数据；将端口 6 设置为通用输出端口时，向此寄存器写入数据，则数据会从 P6[7:0]输出。

7 中断系统

特性:

- 13 个中断源
- 2 层中断优先级

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。

7.1 中断源

一共有 13 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 Int0	0x0003	0	IE0(TCON.1)	是	IE.0	IP.0
Reserved	0x000b	1	NA	NA	NA	NA
IO 变化中断	0x0013	2	PI(TCON.3)	否	IE.2	IP.2
Timer1 溢出	0x001b	3	TF1(TMOD.1)	是	IE.3	IP.3
UART	0x0023	4	RI(SCON.0) TI(SCON.1)	否	IE.4	IP.4
SPI	0x002b	5	SPI_CNTL.7 SPI_CNTL.6 SPI_CNTL.5 SPI_CNTL.4	否	IE.5	IP.5
ADC	0x0033	6	ADC0CN.5	否	IE.6	IP.6
Timer2 溢出	0x003b	7	TMR2CN.7 TMR2CN.6	否	EIE1.0	EIP1.0
Reserved	0x0043	8	NA	NA	NA	NA
Timer3 溢出	0x004b	9	TMR3CN.7 TMR3CN.6	否	EIE1.2	EIP1.2
RTC	0x0053	10	IRQ0.3	否	EIE1.3	EIP1.3
Timer4 溢出	0x005b	11	TMR4CN.7 TMR4CN.6	否	EIE1.4	EIP1.4
CS	0x0063	12	IRQ0.5	否	EIE1.5	EIP1.5
I2C	0x006b	13	IRQ0.6	否	EIE1.6	EIP1.6
MCD	0x0073	14	IRQ0.7	否	EIE1.7	EIP1.7

注意:

- 除了外部中断、定时器 1 中断标志可以由硬件清 0 外, 其它 11 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 INT0 脚中断和 P0 端口变化中断外, 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)。

7.2 中断寄存器

7.2.1 IE 寄存器

寄存器 A8H: IE寄存器 (IE)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
EA	EADC	ESPI	ES0	ET1	EP0	—	EX0
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7 EA: 中断总开关

1 = 打开

0 = 禁止所有中断

bit6 EADC: ADC中断允许位

1 = 允许 ADC 中断

0 = 禁止ADC中断

bit5 ESPI : SPI 中断允许位

1 = 允许 SPI 中断

0 = 禁止SPI中断

bit4 ES0: UART 中断允许位

1 = 允许 UART 中断

0 = 禁止UART中断

bit3 ET1: Timer1中断允许位

1 = 允许 Timer1 中断

0 = 禁止Timer1中断

bit2 EP0: P0端口中断允许位

1 = 允许 P0 端口中断

0 = 禁止P0端口中断

bit1 未实现位: 读为0

bit0 EX0: 外部 (Int0) 中断允许位

1 = 允许外部 (Int0) 中断

0 = 禁止外部 (Int0) 中断

7.2.2 扩展中断允许寄存器 (EIE1)

寄存器 E8H: 扩展中断允许寄存器 (EIE1)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
--------	-------	-------	-------	--------	-------	-----	-------

EMCD	EI2C	ECS	ET4	ERTC	ET3	—	ET2
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7 EMCD: MCD中断允许位
1 = 允许 MCD 中断
0 = 禁止MCD中断

bit6 EI2C: I2C中断允许位
1 = 允许 I2C 中断
0 = 禁止I2C中断

bit5 ECS : 自容中断允许位
1 = 允许自容中断
0 = 禁止自容中断

bit4 ET4: T4 中断允许位
1 = 允许 T4 中断
0 = 禁止T4中断

bit3 ERTC: RTC中断允许位
1 = 允许 RTC 中断
0 = 禁止RTC中断

bit2 ET3: 定时器3中断允许位
1 = 允许定时器 3 中断
0 = 禁止定时器3中断

bit1 未实现位: 读为0

bit0 ET2: 定时器 2 中断允许位
1 = 允许定时器 2 中断
0 = 禁止定时器2中断

7.2.3 中断优先级控制寄存器 (IP)

通过中断优先级控制寄存器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。

寄存器 B8H: 中断优先级控制寄存器 (IP)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
—	—	PSPI	PS0	PT1	PP0	—	PX0
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-6	未实现位：读为0
bit5	PSPI：SPI 优先级 1= 允许 SPI 中断高优先级 0=默认优先级
bit4	PS0：UART 中断优先级 1= 允许 UART 中断高优先级 0=默认优先级
bit3	PT1：Timer1中断优先级 1= 允许 Timer1 中断高优先级 0=默认优先级
bit2	PP0：P0端口中断优先级 1= 允许 P0 端口中断高优先级 0=默认优先级
bit1	未实现位：读为0
bit0	PX0：外部（Int0）中断优先级 1 = 允许外部（Int0）中断高优先级 0 = 默认优先级

7.2.4 扩展中断优先级控制寄存器（EIP1）

寄存器 D8H：扩展中断优先级控制寄存器（EIP1）

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
PMCD	PI2C	PCS	PT4	PRTC	PT3	—	PT2
bit7							bit0

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7	PMCD：MCD优先级 1 = 允许 MCD 中断高优先级 0 = 默认优先级
bit6	PI2C：I2C 优先级 1 = 允许 I2C 中断高优先级 0 = 默认优先级
bit5	PCS：自容中断优先级 1 = 允许自容中断高优先级 0 = 默认优先级
bit4	PT4：定时器4中断优先级 1 = 允许定时器 4 中断高优先级 0 = 默认优先级

bit3	PRTC: RTC中断优先级 1 = 允许 RTC 中断高优先级 0 = 默认优先级
bit2	PT3: 定时器3中断优先级 1 = 允许定时器 3 中断高优先级 0 = 默认优先级
bit1	未实现位: 读为 0
bit0	PT2: 定时器 2 中断优先级 1 = 允许定时器 2 中断高优先级 0 = 默认优先级

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 (Priority Within Level, “同级优先级”) 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。

7.2.5 中断标志位寄存器 0 (IRQ0)

寄存器 C0H: 中断标志位寄存器0 (IRQ0)

R/W -0	R/W-0	R/W-0	U-0	R/W -0	U-0	U-0	U-0
MCD_IRQ	I2C_IRQ	CS_IRQ	—	RTC_IRQ	—	—	—
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7	MCD_IRQ: MCD 中断标志 1= 发生了 MCD 中断 (只有当时钟缺失条件消失后, 软件才能把它清 0) 0= 未发生MCD中断
bit6	I2C_IRQ : I2C 中断标志 只能由软件清0, 往此位写0即可
bit5	CS_IRQ: 自容触摸中断标志 只能由软件清0, 往此位写0即可
bit4	未实现位: 读为0
bit3	RTC_IRQ: RTC中断标志 只能由软件清0, 往此位写0即可
bit2-0	未实现位: 读为 0

7.2.6 P0 沿中断引脚选择寄存器 (P0SEL)

寄存器 84H: P0沿中断引脚选择寄存器 (P0SEL)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
P0.7_SEL	P0.6_SEL	P0.5_SEL	P0.4_SEL	P0.3_SEL	P0.2_SEL	P0.1_SEL	P0.0_SEL
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

- bit7 P0.7_SEL: IO 沿中断触发源选择位
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
- bit6 P0.6_SEL : IO 沿中断触发源选择位
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
- bit5 P0.5_SEL: IO 沿中断触发源选择位
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
- bit4 P0.4_SEL: IO沿中断触发源选择位
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
- bit3 P0.3_SEL: IO沿中断触发源选择位
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
- bit2 P0.2_SEL: IO沿中断触发源选择位
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
- bit1 P0.1_SEL: IO 沿中断触发源选择位
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
- bit0 P0.0_SEL: IO 沿中断触发源选择位
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

YS68F9XX(X)的 P0.0~P0.7 一共 8 个管脚口具有电平变化中断功能, 使用它的条件是:

- P0 相关管脚处于数字输入状态;
- 软件读取 P0 口, 把值锁存到后级寄存器;
- 置 P0SEL 的相关位;

同样可以看到, 满足以下条件之一才能清端口变化中断标志:

- 1) 外部端口恢复原来的电平;
- 2) 软件重新读取一下端口, 刷新寄存器值;
- 3) 满足以上条件之一后, 向 TCON.3 写 0;

设置 P0.n 为 IO 沿中断触发步骤：

- 1) 设置 P0.n 为数字输入口；
- 2) 读 P0；
- 3) 置“1”P0SEL.n。

清 P0 沿中断标志步骤：

- 1) 读 P0（或 P0.n 恢复原来电平）；
- 2) 清“0”IE1。

7.2.7 定时器控制寄存器（TCON）

寄存器 88H：定时器控制寄存器（TCON）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
TF1	TR1	TF0	TR0	IE1	—	IE0	IT0
bit7							bit0

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7 TF1：定时器 1 溢出标志

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

bit6 TR1：定时器 1 使能
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

bit5 TF0：IO 沿中断触发源选择位
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

bit4 TR0：IO沿中断触发源选择位
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

bit3 IE1：IO沿中断触发源选择位
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

bit2 未实现位：读为0

bit1 IE0：引脚 Int0 中断标志
硬件检测到 Int0 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

bit0 IT0：Int0 引脚的上升下降沿检测选择位
1 = 选择上升沿
0 = 选择下降沿

7.2.8 唤醒源配置寄存器（WKSRC）

寄存器 85H：唤醒源配置寄存器（WKSRC）

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	U-0	R/W -0	R/W-0
—						INT1WK	INT0WK
bit7						bit0	

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-2 未实现位：读为0

bit1

INT1WK:

0: 只有在打开相关中断位的情况下，P0 端口变化中断源才能唤醒 MCU

1: 在没有使能相关中断位的情况下，P0 端口变化中断源可以唤醒 MCU

bit0

INT0WK: Int0 引脚的上升下降沿检测选择位

1 = 在没有使能相关中断位的情况下，INT0 沿中断源可以唤醒 MCU

0 = 只有在打开相关中断位的情况下，INT0 沿中断源才能唤醒 MCU

注意：

如果要使用非中断唤醒 CPU 的低功耗模式（包括 IDLE/STOP/DEPD），软件必须在写完 PCON 的指令后面紧跟着三条 NOP 指令，否则程序会跑飞。

8 定时器

YS68F9XX(X)有 4 个 16 位的定时/计数器，它们分别是 Timer1、Timer2、Timer3 和 Timer4，每个定时/计数器都是由两个 8bit 的特殊功能寄存器构成。时钟源可以选择快时钟或慢时钟，只有 Timer1 带有预分频器，Timer2、Timer3 和 Timer4 只能选择固定的分频比。根据设置的不同，它们可工作在 16bit 模式、8bit 模式、PPG 模式以及 Capture 模式。

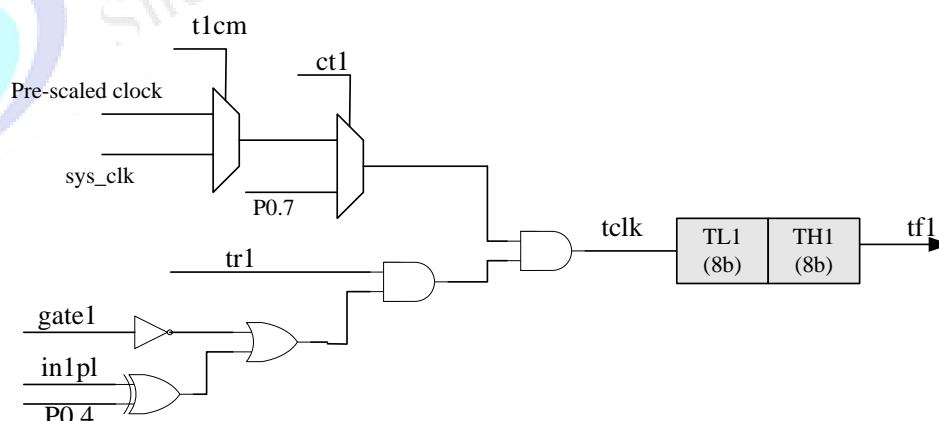
timer1	timer2	timer3	timer4
16 位模式定时/计数	16 位模式定时/计数+自动重载	16 位模式定时/计数+自动重载	16 位模式定时/计数+自动重载
8 位模式定时/计数+自动重载	2*8 位模式定时/计数+自动重载	2*8 位模式定时/计数+自动重载	2*8 位模式定时/计数+自动重载
	PPG 功能	PPG 功能	PPG 功能
	I2C 超时监测	捕获功能，抓捕开始时硬件自动清 0	捕获功能，抓捕开始时硬件自动清 0

8.1 定时器 1

Timer1 由高 8 位 TH1 和低 8 位 TL1 组成，可通过字节传送指令为它们分别设置初值，以便它们可以定时为不同的时间并获得所需的计数值，它可以工作于 16 bit 模式和 8 bit 自动重载模式。

8.1.1 16 bit 模式

功能框图：



在此方式下，timer1 是按 16 位加 1 计数器工作的，该计数器是由高 8 位 TH 和低 8 位 TL 组成。在 timer1 启动工作前，CPU 先要为其装入方式控制字，以设定其工作方式和时钟来源，在事件计数方式下，时钟来源是 P0.7。然后再为其装入定时器/计数器初值，并通过指令启动其工作。16 位计数器按加 1 计数器计数，当计数从 0xFFFF 溢出到 0x0000 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

8.1.3 Timer1 寄存器

8.1.3.1 时钟控制寄存器 (CKCON)

寄存器 8EH: 时钟控制寄存器 (CKCON)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
T4MH	T4ML	T3MH	T3ML	T2MH	T2ML	SCA[1:0]	
bit7							bit0

图注:

R = 可读位 W = 可写位 U = 未实现位, 读为0
-n = POR时的值 1 = 置1 0 = 清零 x = 未知

- bit7 T4MH: Timer4 高字节时钟选择位
选择提供给 Timer4 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=Timer4 高字节使用 TMR4CN 寄存器中的 T4XCLK 定义的时钟来计数;
1=Timer4高字节使用系统时钟来计数
- bit6 T4ML : Timer4 低字节时钟选择位
选择提供给 Timer4 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=Timer4 低字节使用 TMR4CN 寄存器中的 T4XCLK 定义的时钟来计数;
1=Timer4低字节使用系统时钟来计数。
- bit5 T3MH: Timer3 高字节时钟选择位
选择提供给 Timer3 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=Timer3 高字节使用 TMR3CN 寄存器中的 T3XCLK 定义的时钟来计数;
1=Timer3高字节使用系统时钟来计数。
- bit4 T3ML: Timer3 低字节时钟选择位
选择提供给 Timer3 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=Timer3 低字节使用 TMR2CN 寄存器中的 T3XCLK 定义的时钟来计数;
1=Timer3低字节使用系统时钟来计数。
- bit3 T2MH: Timer2 高字节时钟选择位
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。
- bit2 T2ML: Timer2 低字节时钟选择位
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。
- bit1-0 SCA[1:0]: Timer1 分频比选择位
00=系统时钟 4 分频
01=系统时钟 12 分频
10=系统时钟 48 分频
11=系统慢时钟 8 分频

8.1.3.2 模式寄存器 (TMOD)

寄存器 89H: 模式寄存器 (TMOD)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
GATE1	IN1PL	C/T1	T1M		T1CM	TF1	TR1
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7	GATE1: timer1 Gate Control 0=timer1 enabled when TR1=1 ,irrespective of INT1 logic level(P0.4) 1=timer1 enabled only when TR1=1 and INT1 is active as defined by bit IN1PL.
bit6	IN1PL : int_int 极性设置位 0=低电平有效 1=高电平有效
bit5	C/T1: Counter1Timer1 选择位。 0=timer 功能: 按照 SCA[1:0]选择的时钟递增 1。 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。 注: SCA[1:0]在寄存器CKCON[1:0]里面。
bit4	T1M: Counter1Timer1 模式选择位 0=模式 1, 即 16 位 counter/timer 1=模式 2, 即 8 位自动重载 counter/timer 注: 在8位自动重载模式下, 第一个计数周期并不是从重载值开始计数, 而是从初始值开始计数。初始值可以用软件直接设置。
bit3	未实现位: 读为0
bit2	T1CM: Timer1 时钟选择位 选择 Timer1 的计数时钟源。当 C/T1 为 1 时, 则忽略此设置。 0=定时器 1 使用分频配置位 SCA[1:0]选择的时钟计数; 1=定时器1使用系统时钟计数。
bit1	TF1: Timer1/Counter1 溢出标志 0=Timer1/Counter1 未溢出 1=Timer1/Counter1 已经溢出 (此标志位可用软件来清零) 注: 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。
bit0	TR1: Timer1/Counter1 运行控制位 1 = timer1/counter1 开始运行 0 = 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。

8.1.3.3 低字节寄存器 (TL1)

寄存器 8bH: 低字节寄存器 (TL1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TL1[7:0]							
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0

TL1[7:0]: 16位Timer1的低字节值

8.1.3.4 高字节寄存器 (TH1)

寄存器 8DH: 高字节寄存器 (TH1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T1H[7:0]							
bit7						bit0	

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0

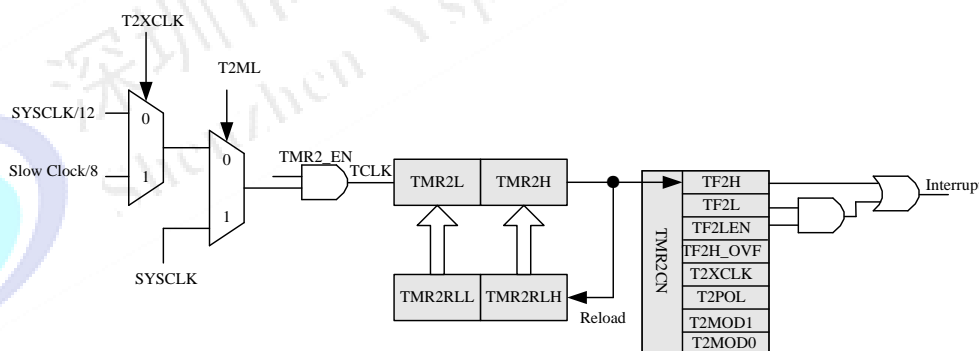
TH1[7:0]: 16位Timer1的高字节值。8位时用于保存TIMER1的自动重载值。

8.2 定时器 2

Timer2 是一个 16 bit 定时器, 由两个 8 bit 的特殊功能寄存器组成, 它们分别是 TMR2L 和 TMR2H。Timer2 可以工作在三种模式: 16 bit 自动重载模式、8 bit 自动重载模式、以及 PPG 模式, 另外它还可以用于 I2C 超时监控功能。

8.2.1 16Bit 自动重载模式

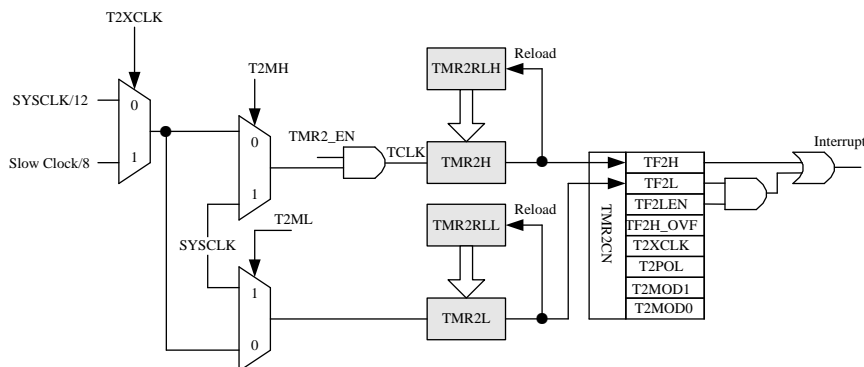
功能框图



在此模式下, Timer2 是按 16 位加 1 计数器工作的, 当计数发生溢出时 (从全'1'到 0x0000), 定时器溢出中断标志 TF2H 被置位, 重载寄存器 TMR2RLL 和 TMR2RLH 的值被重新载入到 Timer2 的寄存器中。在中断使能的情况下, CPU 会进入中断向量。如果 TF2LEN 被置一, 则使能 timer2 的低字节中断, 当 timer2 的低字节溢出时, 也会产生 CPU 中断。

8.2.2 8Bit 自动重载模式

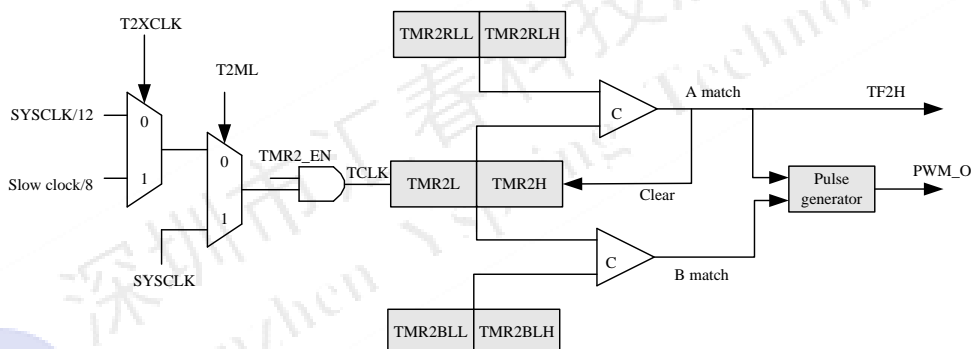
功能框图



将寄存器 TMR2CN[1:0] 设置成[00], 则 Timer2 工作在 8 bit 自动重载模式。此时 Timer2 被分成两个独立的 8 bit 的定时/计数器, 计数值分别在 TMR2L 与 TMR2H 之中。当 timer2 的高字节从 0xFF 溢出到 0x00 时, 重载寄存器 TMR2RLH 的值被重新载入到 TMR2H 之中, 且位 TF2H 由硬件自动置 1。如果使能了 timer2 的中断, CPU 会进入中断向量。当 timer2 的低字节从 0xFF 溢出到 0x00 时, 重载寄存器 TMR2RLL 的值自动载入到 TMR2L 之中, 由硬件自动置一 TF2L 位。中断允许, 则 CPU 会进入中断向量。注意, 在此模式下, 只有高字节需要使能信号才开始计数, 低字节不需要使能信号就开始计数。

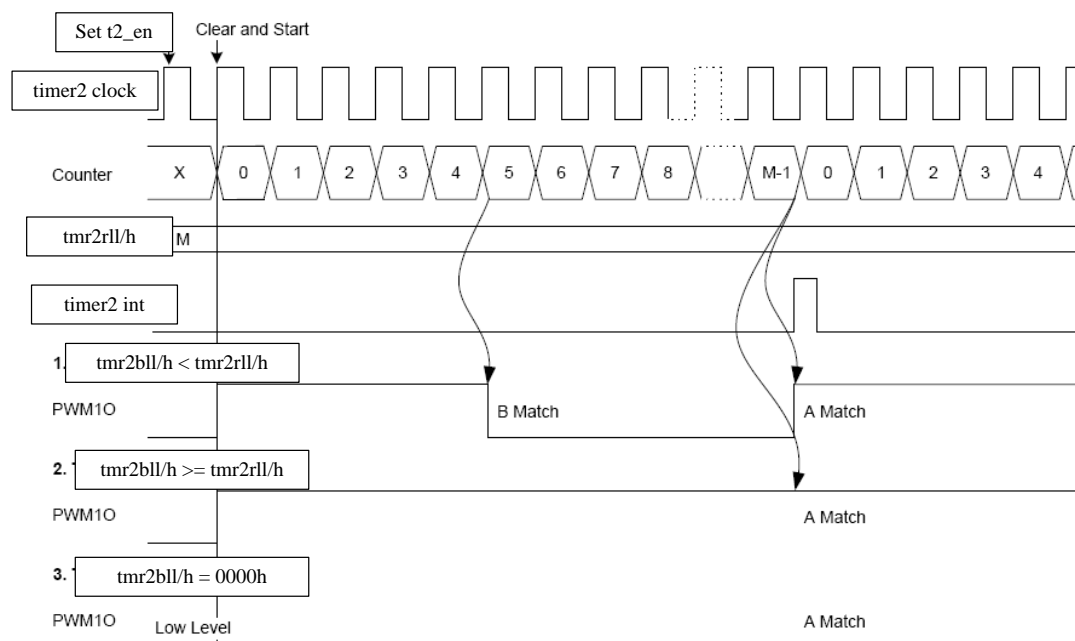
8.2.3 PPG 模式

功能框图



Timer2 在 PPG 模式下, 有三组 16 bit 寄存器, 分别是 PWM 输出周期寄存器 TMR2RLL 和 TMR2RLH 既 A 寄存器、计数寄存器 TMR2L 和 TMR2H, 以及占空比设置寄存器 TMR2BLL 和 TMR2BLH 既 B 寄存器。功能转移之前 PWM 的输出是 P3.2, 功能转移之后 PWM 的输出是 P0.4。当往 TMR2_EN 位写 1 时, 就开始了 PPG 的一个周期, 这个动作会清零计数寄存器 TMR2L 和 TMR2H, 并设置 PWM 输出为高。当计数值与 A 寄存器发生匹配时, 将清零计数器的值, 并置 PWM 的输出为高; 计数值与 B 寄存器发生匹配时, PWM 输出发生翻转。如果 B 寄存器的值大于 A 寄存器, 则 PWM 输出为高, B 寄存器的值为零时, PWM 输出为低。

Timer2 PPG 输出波形:



8.2.4 I2C 超时检测功能描述

Timer2 的超时监控功能必须用于 Timer2 工作于 16bit 模式，并且打开 Timer2。在 I2C 进行工作的时候打开边沿监测，检测到 I2C 的 start 信号以后，会在时钟 SCL 电平发生跳变的时候，对 Timer2 的计数值进行自动重载；如果 SCL 电平不变化，则 Timer2 一直递增计数，最终产生溢出中断。当 I2C 通信结束（收到 End 信号）以后，Timer2 停止监测，不再有重载操作，并一直递增计数，直到溢出后再发生重载操作。

8.2.5 Timer2 寄存器

8.2.5.1 Timer2 控制寄存器（TMR2CN）

寄存器 C8H: Timer2控制寄存器（TMR2CN）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
TF2H	TF2L	TF2LEN		T2XCLK	T2POL	T2MOD[1:0]	
bit7							bit0

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7

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。此标志位只能由硬件产生，不能由软件写入。

对于 PPG 模式：

在调试模式下（单步时或 CPU 暂停时），PWM 照常输出，但不会产生中断标志位。PWM 的占空比为进入调试模式之前的占空比，如果在单步时修改了占空比寄存器，则也会实时反应到当前输出的 PWM 波形上。在全速运行时，会在每个 PWM 输出周期产生一次中断标志。

bit6

TF2L : Timer2 低字节溢出标志

在 8 位计数模式或 16 位计数模式下，当 timer2 的低字节从 0xFF 溢出到 0x00 时，硬件自动置 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。

在 PPG 下不会产生溢出标志，也不会产生相应的中断。

bit5

TF2LEN: Timer2 低字节中断使能位

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。

bit4 未实现位：读为0:

bit3 T2XCLK: Timer2时钟选择位
0=计数时钟为系统时钟 12 分频
1=计数时钟为慢时钟8分频（慢时钟由CLKCF [4]决定）

bit2 T2POL: Timer2 PPG PWNOUT 极性选择
0=高占空比有效
1=低占空比有效

bit1-0 TF1: Timer2 模式选择
00: timer2 工作在两个单独的 8 位计数器模式。
01: timer2 工作在一个 16 位计数器模式。
1x: timer2 工作与 PPG 模式(16bit)
注：在 PPG 模式下，当 PWM 不使用功能转移时，PWM 波形从 P3[2]输出；使用功能转移时，从 P0[4]输出。

8.2.5.2 Timer2 重载寄存器低字节（TMR2RLL）

寄存器91H: Timer2重载寄存器低字节（TMR2RLL）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR2RLL[7:0]							
bit7				bit0			

图注：

R = 可读位 W = 可写位 U = 未实现位，读为0
-n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7-0 TMR2RLL[7:0]: Timer2重载寄存器低字节
TMR2RLL 保存 timer2 的重载值的低字节。
在PPG模式下，用于设置PWM输出的周期，即A寄存器

8.2.5.3 Timer2 重载寄存器高字节（TMR2RLH）

寄存器92H: Timer2重载寄存器高字节（TMR2RLH）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR2RLH[7:0]							
bit7				bit0			

图注：

R = 可读位 W = 可写位 U = 未实现位，读为0
-n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7-0 TMR2RLH[7:0]: Timer2 重载寄存器高字节。
TMR2RLH 保存 timer2 的重载值的高字节。
在PPG模式下，用于设置PWM输出的周期，即A寄存器

8.2.5.4 Timer2 PPG 模式占空比设置值低字节（TMR2BLL）

寄存器93H: Timer2 PPG模式占空比设置值低字节（TMR2BLL）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR2BLL[7:0]							
bit7				bit0			

图注：

R = 可读位 W = 可写位 U = 未实现位，读为0
1 = 置1 0 = 清零 x = 未知

-n = POR时的值

bit7-0 TMR2BLL[7:0]: Timer2 PPG 模式下设置占空比的低字节
在8B,16B模式下, 读此位将读出Timer2的值

8.2.5.5 Timer2 PPG 模式占空比设置值高字节 (TMR2BLH)

寄存器94H: Timer2 PPG模式占空比设置值高字节 (TMR2BLH)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR2BLH[7:0]							
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

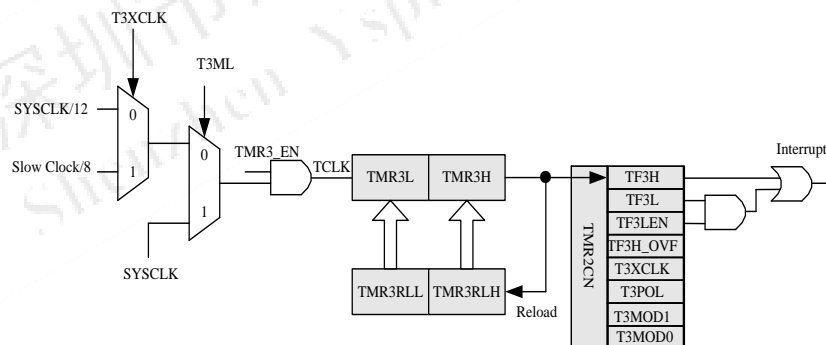
bit7-0 TMR2BLH[7:0]: Timer2 PPG 模式下设置占空比的高字节
在8B,16B模式下, 读此位将读出Timer2的值

8.3 定时器 3

Timer3 是一个 16 bit 定时器, 由两个 8 bit 的特殊功能寄存器组成, 它们分别是 TMR3L 和 TMR3H。Timer3 可以工作在四种模式: 16 bit 自动重载模式、8 bit 自动重载模式、PPG 模式、以及外部事件捕捉模式。Timer3 与 Timer2 的不同之处在于 Timer3 具有 Capture 模式, 而 Timer2 没有, 但 Timer2 具有 i2c 时钟超时检测而 Timer3 没有。

8.3.1 16 bit 自动重载模式

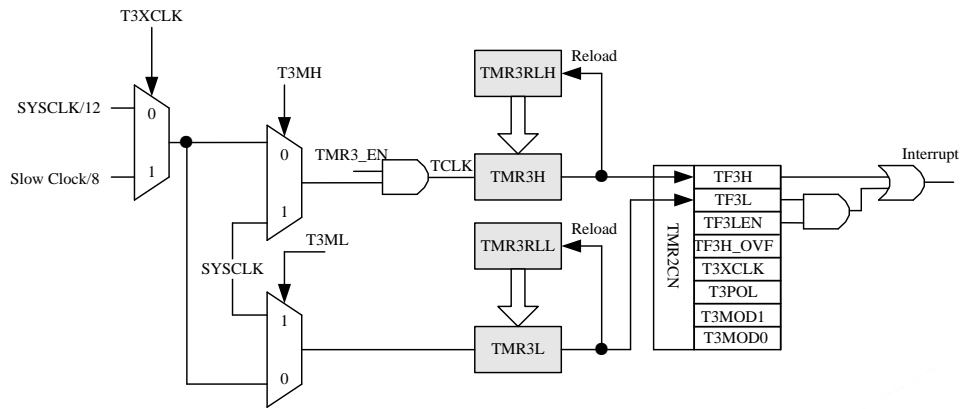
功能框图



详细介绍请参考 Timer2 16 bit 自动重载模式的章节。

8.3.2 8 bit 自动重载模式

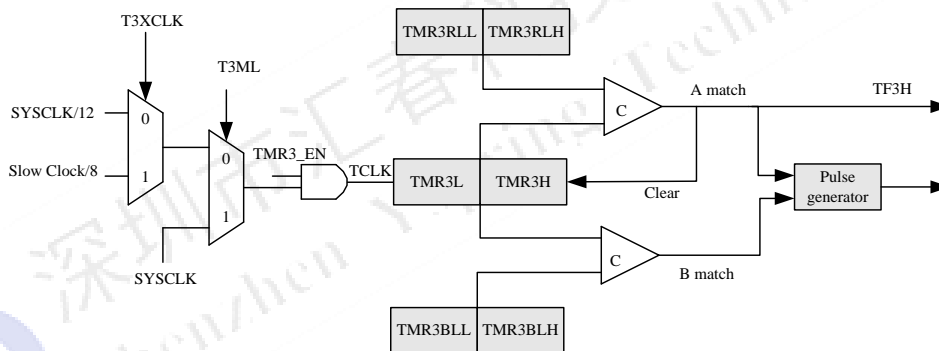
功能框图



详细介绍请参考 Timer2 8 bit 自动重载模式的章节。

8.3.3 PPG 模式

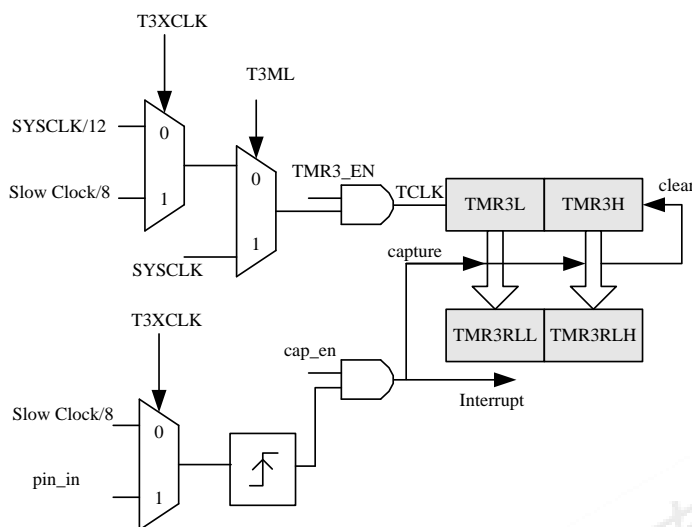
功能框图



Timer3 在此模式下，功能转移之前 PWM 的输出是 P3.0，功能转移之后 PWM 的输出是 P0.2。其他功能请参考 Timer2 PPG 模式的章节。

8.3.4 捕捉模式

功能框图



Timer3 工作在此模式下，可以对外部事件进行计数，功能转移之前外部事件引脚是 P3.0，功能转移后外部事件引脚为 P0.2。根据软件配置的不同，还可以对慢时钟的频率进行测量。当外部事件引脚的电平发生从 0 到 1 的变化，或者处于慢时钟的上升沿时，捕捉事件就发生。此时硬件会把计数器 TMR3L 和 TMR3H 的值捕捉到 TMR3RLL 和 TMR3RLH 寄存器，与此同时会对 TMR3L 和 TMR3H 清零，并置 TF3H 标志位，如果中断允许，则 CPU 进入中断向量。

8.3.5 Timer3 寄存器

8.3.5.1 Timer3 控制寄存器（TMR3CN）

寄存器B1H: Timer3 控制寄存器（TMR3CN）

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
TF3H	TF3L	TF3LEN	TF3H_OVF	T3XCLK	T3POL	T3MOD[1:0]	
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7

TF3H: Timer3 高字节溢出标志位

当 timer3 的高字节从 0xFF 溢出到 0x00 时，硬件自动置 1。在 16 位模式下，当 timer3 从 0xFFFF 溢出到 0x0000 时，硬件自动置 1。如果使能了 timer3 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：捕捉模式同时也是 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。

在抓捕模式下，当抓捕动作发生时，会产生此标志位。

对于 PPG 模式:

在调试模式下（单步时或 CPU 暂停时），PWM 照常输出，但不会产生中断标志位。PWM 的占空比为进入调试模式之前的占空比，如果在单步时修改了占空比寄存器，则也会实时反应到当前输出的 PWM 波形上。

在全速运行时，会在每个 PWM 输出周期产生一次中断标志。

bit6

TF3L : Timer3 低字节溢出标志位

在 8 位或 16 位模式下，当 timer3 的低字节从 0xFF 溢出到 0x00 时，硬件自动置 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。

在PPG或Capture mode下不会产生溢出标志，也不会产生相应的中断。

- bit5 **TF3LEN**: Timer3 低字节中断使能位
如果设置为1, 则使能timer3的低字节中断。如果同时使能了timer3的中断, 则当timer3的低字节溢出时, 就会产生CPU中断。
- bit4 **TF3H_OVF**: Timer3 高字节溢出标志位
当 timer3 的高字节从 0xFF 溢出到 0x00 时, 硬件自动置 1。软件清零。
只在capture mode时产生此标志, 但此标志不产生中断。
- bit3 **T3XCLK**: Timer3 外部时钟选择位
选择“外部”和“抓捕”时钟信号。如果 timer3 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。
Timer3 时钟选择位 (CKCON 中的 T3MH 和 T3ML) 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。
0=计数时钟为系统时钟 12 分频, 抓捕信号为慢时钟 8 分频;
1=计数时钟为慢时钟8分频, 抓捕信号为外部pin输入信号
- bit2 **T3POL**: Timer3 PPG PWNOUT 极性选择位
0= start high(T20/PWMO is low level at disable)
1= start low(T20/PWMO is high level at disable)
- bit1-0 **T3MOD[1:0]**: Timer3 模式选择位
00=timer3 工作在两个单独的 8 位计数器模式。
01=timer3 工作在一个 16 位计数器模式。
10=timer3 工作于抓捕模式(16bit)
11=timer3 工作与 PPG 模式(16bit)
注: 在 PPG 模式下, 当 PWM 不使用功能转移时, PWM 波形从 P3[0]输出; 使用功能转移时, 从 P0[2]输出。

8.3.5.2 Timer3 重载寄存器低字节寄存器 (TMR3RLL)

寄存器E1H: **Timer3重载寄存器低字节 (TMR3RLL)**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR3RLL [7:0]							
bit7							bit0

图注:

R = 可读位 W = 可写位 U = 未实现位, 读为0
-n = POR时的值 1 = 置1 0 = 清零 x = 未知

- bit7-0 **TMR3RLL [7:0]**: Timer3 重载寄存器低字节。**Cap 模式下, 将 T3 的值存入此寄存器。**
TMR3RLL 保存 timer3 的重载值的低字节。
在PPG模式下, 用于设置PWM输出的周期, 即A寄存器

8.3.5.3 Timer3 重载寄存器高字节寄存器 (TMR3RLH)

寄存器E2H: **Timer3重载寄存器高字节 (TMR3RLH)**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR3RLH [7:0]							
bit7							bit0

图注:

R = 可读位 W = 可写位 U = 未实现位, 读为0
-n = POR时的值 1 = 置1 0 = 清零 x = 未知

- bit7-0 **TMR3RLH [7:0]**: Timer3 重载寄存器高字节。**Cap 模式下, 将 T3 的值存入此寄存器**
TMR3RLH 保存 timer3 的重载值的高字节。
在PPG模式下, 用于设置PWM输出的周期, 即A寄存器

8.3.5.4 Timer3 PPG 模式占空比设置值低字节 (TMR3BLL)

寄存器E3H: Timer3 PPG模式占空比设置值低字节 (TMR3BLL)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
TMR3BLL [7:0]							
							bit0

图注:

R = 可读位 W = 可写位 U = 未实现位, 读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

TMR3BLL[7:0]: Timer3 PPG 模式下设置占空比的低字节
 在8B、16B模式下, 读此位将读出Timer3的值

8.3.5.5 Timer3 PPG 模式占空比设置值高字节 (TMR3BLH)

寄存器E4H: Timer3 PPG模式占空比设置值高字节 (TMR3BLH)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
TMR3BLH [7:0]							
							bit0

图注:

R = 可读位 W = 可写位 U = 未实现位, 读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

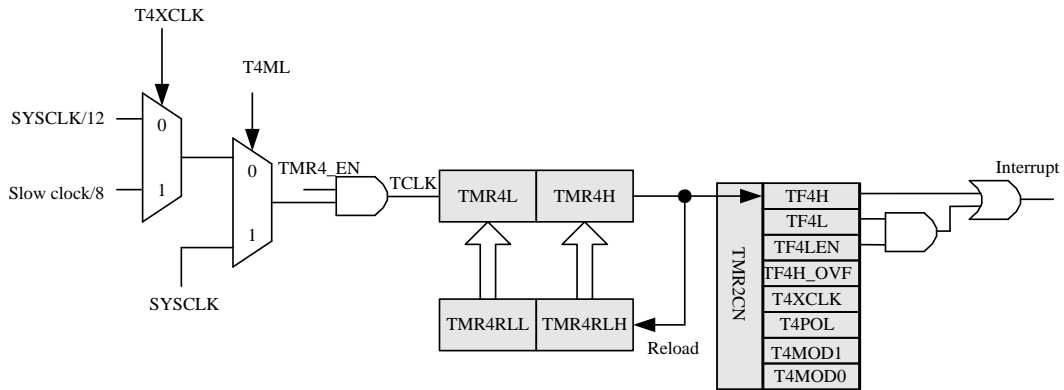
TMR3BLH[7:0]: Timer3 PPG 模式下设置占空比的高字节
 在8B、16B模式下, 读此位将读出Timer3的值

8.4 定时器 4

Timer4 是一个 16 bit 定时器, 由两个 8 bit 的特殊功能寄存器组成, 它们分别是 TMR4L 和 TMR4H。Timer4 可以工作在四种模式: 16 bit 自动重载模式、8 bit 自动重载模式、PPG 模式、以及外部事件捕捉模式。

8.4.1 16 bit 自动重载模式

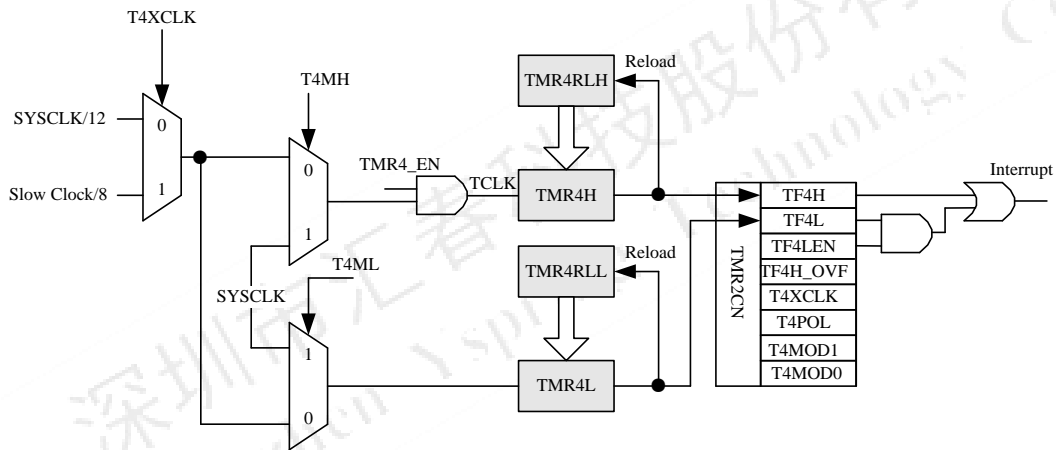
功能框图



请参考 Timer2 16 bit 自动重载模式的章节。

8.4.2 8 bit 自动重载模式

功能框图

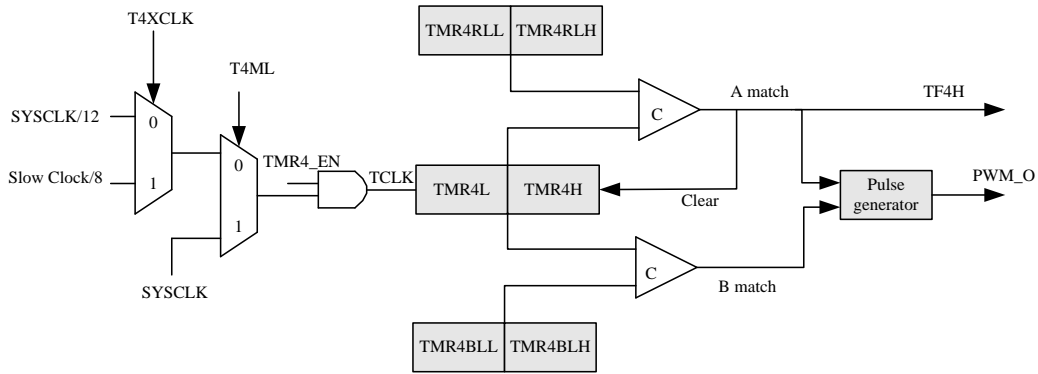


请参考 Timer2 8 bit 自动重载模式的章节。

8.4.3 PPG 模式

Timer4 在此模式下，功能转移之前 PWM 的输出是 P3.1，功能转移之后 PWM 的输出是 P0.3。
关闭 PPG 时，I/O 口状态由初始化决定。

功能框图

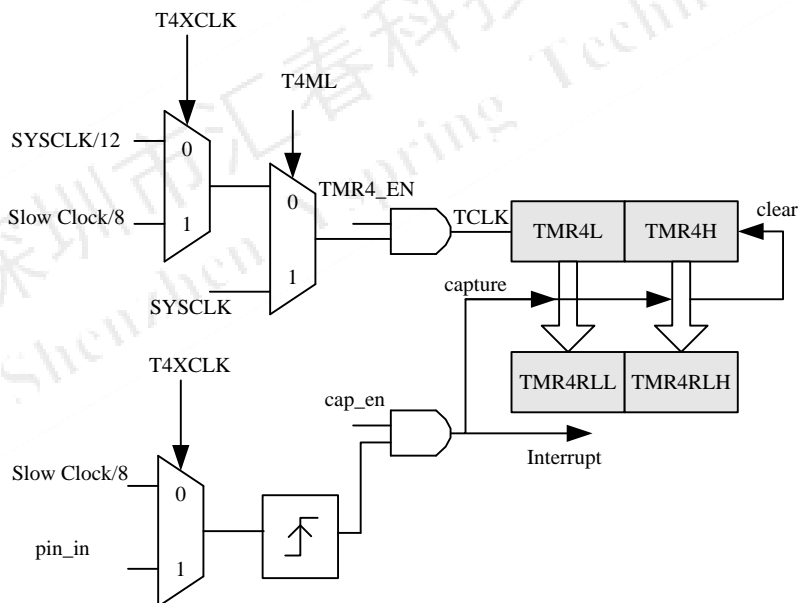


请参考 Timer2 PPG 模式的章节

8.4.4 捕捉模式

Timer4 工作在此模式下，可以对外部事件进行计数，功能转移之前外部事件引脚是 P3.1，功能转移后外部事件引脚为 P0.3。

功能框图



请参考 Timer3 捕捉模式的章节。

8.4.5 Timer4 寄存器

8.4.5.1 Timer4 控制寄存器 (TMR4CN)

寄存器B2H: Timer4 控制寄存器 (TMR4CN)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
TF4H	TF4L	TF4LEN	TF4H_OVF	T4XCLK	T4POL	T4MOD[1:0]	
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7

TF4H: Timer4 高字节溢出标志位

当 timer4 的高字节从 0xFF 溢出到 0x00 时, 硬件自动置 1。在 16 位模式下, 当 timer4 从 0xFFFF 溢出到 0x0000 时, 硬件自动置 1。如果使能了 timer4 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: 捕获模式同时也是 16 位模式, 但仅在捕获时产生中断, 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。

在捕获模式下, 当捕获动作发生时, 会产生此标志位。

对于 PPG 模式:

在调试模式下 (单步时或 CPU 暂停时), PWM 照常输出, 但不会产生中断标志位。PWM 的占空比为进入调试模式之前的占空比, 如果在单步时修改了占空比寄存器, 则也会实时反应到当前输出的 PWM 波形上。

在全速运行时, 会在每个 PWM 输出周期产生一次中断标志。

bit6

TF4L: Timer4 低字节溢出标志位

在 8 位或 16 位模式下, 当 timer4 的低字节从 0xFF 溢出到 0x00 时, 硬件自动置 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。

在 PPG 或 Capture mode 下不会产生溢出标志, 也不会产生相应的中断。

bit5

TF4LEN: Timer4 低字节中断使能位

如果设置为 1, 则使能 timer4 的低字节中断。如果同时使能了 timer3 的中断, 则当 timer3 的低字节溢出时, 就会产生 CPU 中断。

bit4

TF4H_OVF: Timer4 高字节溢出标志位

当 timer4 的高字节从 0xFF 溢出到 0x00 时, 硬件自动置 1。软件清零。

只在 capture mode 时产生此标志, 但此标志不产生中断。

bit3

T4XCLK: Timer4 外部时钟选择位

选择“外部”和“捕获”时钟信号。如果 timer4 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。

Timer4 时钟选择位 (CKCON 中的 T4MH 和 T4ML) 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。

0=计数时钟为系统时钟 12 分频, 捕获信号为慢时钟 8 分频;

1=计数时钟为慢时钟 8 分频, 捕获信号为外部 pin 输入信号

bit2

T4POL: Timer4 PPG PWNOUT 极性选择位

0= start high (T20/PWMO is low level at disable)

1= start low (T20/PWMO is high level at disable)

bit1-0

T4MOD[1:0]: Timer4 模式选择位

00=timer4 工作在两个单独的 8 位计数器模式。

01=timer4 工作在一个 16 位计数器模式。

10=timer4 工作于捕获模式(16bit)

11=timer4 工作与 PPG 模式(16bit)

注: 在 PPG 模式下, 当 PWM 不使用功能转移时, PWM 波形从 P3[1]输出; 使用功能转移时, 从 P0[3]输出。

8.4.5.2 Timer4 重载寄存器低字节 (TMR4RLL)

寄存器F1H: Timer3重载寄存器低字节 (TMR4RLL)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR4RLL [7:0]							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 TMR4RLL [7:0]: Timer4 重载寄存器低字节。Cap 模式下, 将 T4 的值存入此寄存器。
TMR4RLL 保存 timer4 的重载值的低字节。
在PPG模式下, 用于设置PWM输出的周期, 即A寄存器

8.4.5.3 Timer4 重载寄存器高字节 (TMR4RLH)

寄存器F2H: Timer4重载寄存器高字节 (TMR4RLH)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR4RLH [7:0]							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 TMR4RLH [7:0]: Timer4 重载寄存器低字节。Cap 模式下, 将 T4 的值存入此寄存器。
TMR4RLH 保存 timer4 的重载值的高字节。
在PPG模式下, 用于设置PWM输出的周期, 即A寄存器

8.4.5.4 Timer4 PPG 模式占空比设置值低字节 (TMR4BLL)

寄存器F3H: Timer4 PPG模式占空比设置值低字节 (TMR4BLL)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR4BLL [7:0]							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0 TMR4BLL [7:0]: Timer4 PPG 模式下设置占空比的低字节
在 8B,16B,CAP 模式下, 读此寄存器将读出 Timer4 的计数值低字节。
注: 如果timer4一直在计数, 则读出的计数值并不能实时反映出其当前真实值。

8.4.5.6 Timer4 PPG 模式占空比设置值高字节 (TMR4BLH)

寄存器F4H: Timer4 PPG模式占空比设置值高字节 (TMR4BLH)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR4BLH[7:0]							
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-0

TMR4BLH[7:0]: Timer4 PPG 模式下设置占空比的高字节

在 8B,16B,CAP 模式下, 读此寄存器将读出 Timer4 的计数值高字节。

注: 如果timer4一直在计数, 则读出的计数值并不能实时反映出其当前真实值。

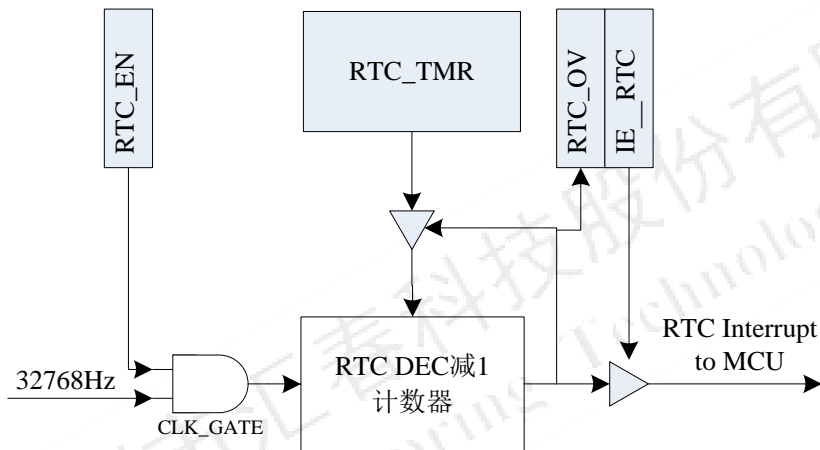
9 实时时钟（RTC）

RTC 模块是基于一个被 32768Hz 时钟驱动的 16bit 减计数器，用户配置好初始时间寄存器，使能后 RTC 开始计时，计时溢出后上报中断，并重新装载继续计时。

主要特性：

- 16 位减计数器，32768 时钟驱动，可实现最长 2 秒钟精确计时。
- 可使能中断，上报中断至 MCU
- 计数过程中可读取动态计数值
- MCU 休眠条件下，RTC 仍可正常工作

功能框图



9.1 RTC 寄存器

9.1.1 RTC 综合控制寄存器（PER_EN）

寄存器F6H：RTC综合控制寄存器（PER_EN）

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
TOUTEN	—	RTC_RST	I2C_EN	RTC_EN	TMR4_EN	TMR3_EN	TMR2_EN
bit7							bit0

图注：

R = 可读位 W = 可写位 U = 未实现位，读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7 TOUTEN: Touch 使能位
 1 = 使能触摸功能
 0 = 禁止触摸功能

bit6 未实现位：读为0

bit5 RTC_RST: RTC 复位
 RTC 运行过程中，置 RTC_RST = 1，RTC 将复位，重新装载 RTC 初值，并从该初值重新计数，装载完初值后 RTC_RST 由硬件自动清“0”。

bit4	I2C_EN: I2C 使能位 1 = 使能 I2C 0 = 禁止 I2C
bit3	RTC_EN: RTC 使能位 1 = 使能 RTC 计数功能, 计数溢出后产生中断; 0 = 禁用 RTC 功能。
bit2	TMR4_EN: 定时器 4 使能位 1 = 使能 Timer4 0 = 禁止 Timer4
bit1	TMR3_EN: 定时器 3 使能位 1 = 使能 Timer3 0 = 禁止 Timer3
bit0	TMR2_EN: 定时器 2 使能位 1 = 使能 Timer2 0 = 禁止 Timer2

9.1.2 RTC 计数/重载值低字节寄存器 (RTCL)

寄存器BEH: RTC计数/重载值低字节寄存器 (RTCL)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
RTCL[7:0]							
bit7							bit0

图注:

R = 可读位	W = 可写位	U = 未实现位, 读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit7	RTCL[7:0]: 16 位 RTC 的低字节值 读: 16 位 RTC 计数值低字节 写: 16 位 RTC 重载值低字节
------	---

9.1.3 RTC 计数/重载值高字节寄存器 (RTCH)

寄存器BFH: RTC计数/重载值高字节寄存器 (RTCH)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
RTCH[7:0]							
bit7							bit0

图注:

R = 可读位	W = 可写位	U = 未实现位, 读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit7	RTCH[7:0]: 16 位 RTC 的高字节值 读: 16 位 RTC 计数值高字节 写: 16 位 RTC 重载值高字节
------	---

RTC 重载值计算:

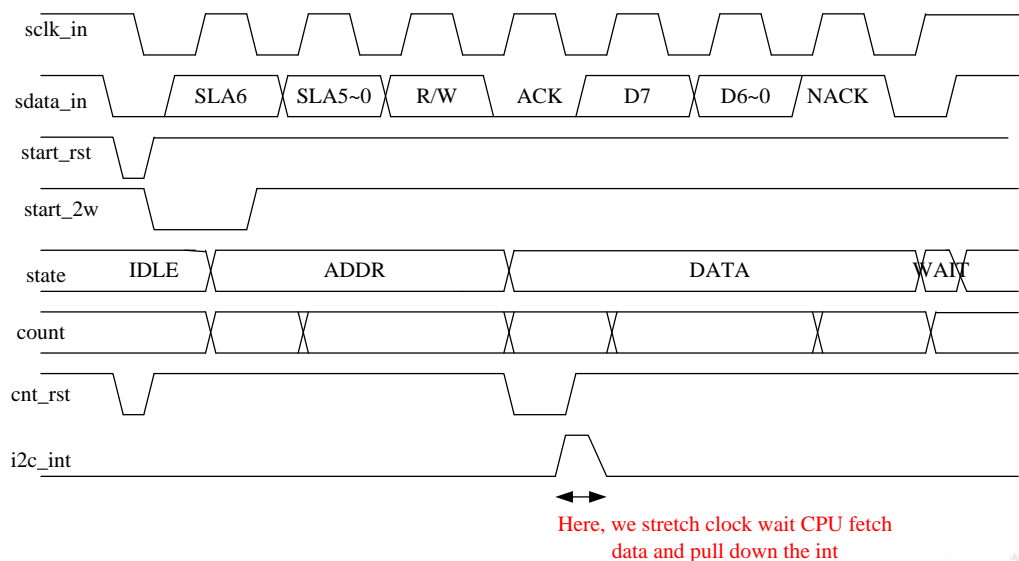
$$\text{RTC}(\text{RTCH}, \text{RTCL}) = 32768 * T \quad (T \text{ 为需要定时的时间, 单位: 秒})$$

10 IIC

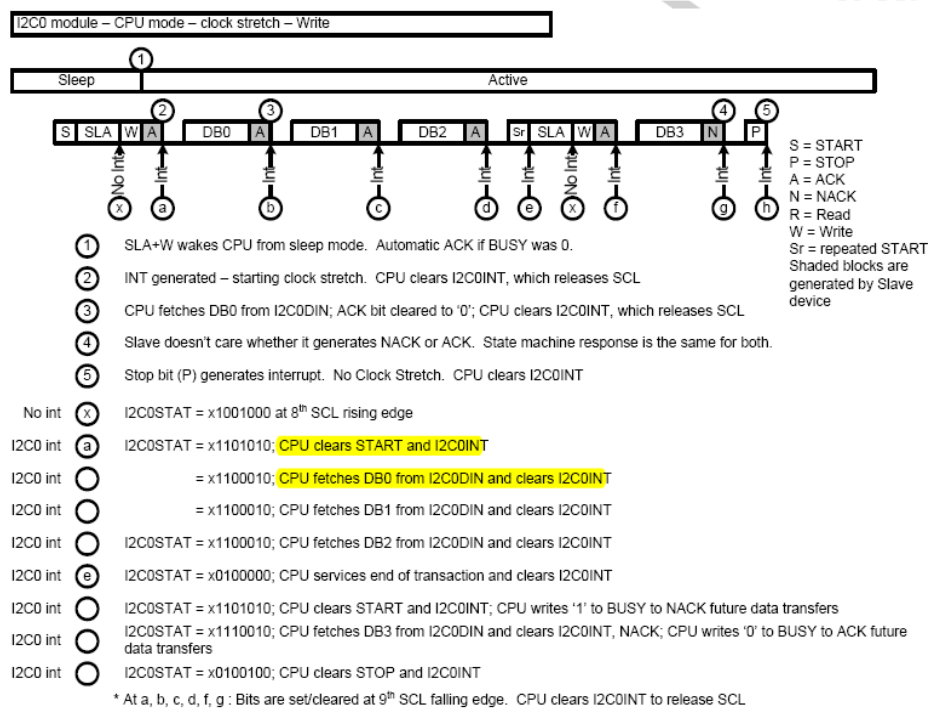
YS68F9XX(X)的 I2C 接口是一个双线，双向的串行接口。本设计与 I2C 协议定义 2.0 相匹配。它可以运行于高速模式，最高可达到 400K。CPU 可以通过寄存器控制 I2C 模块读写数据。本接口同时支持 clock stretching，可以允许 CPU 时时控制发送和接收的每个 Byte 的数据。本 I2C 是自时钟的，可以在 sleep 模式下工作，并且当接收的地址与本机相匹配时可以唤醒 CPU。

注意：YS68F9XX(X)的 I2C 接口只支持从机模式。

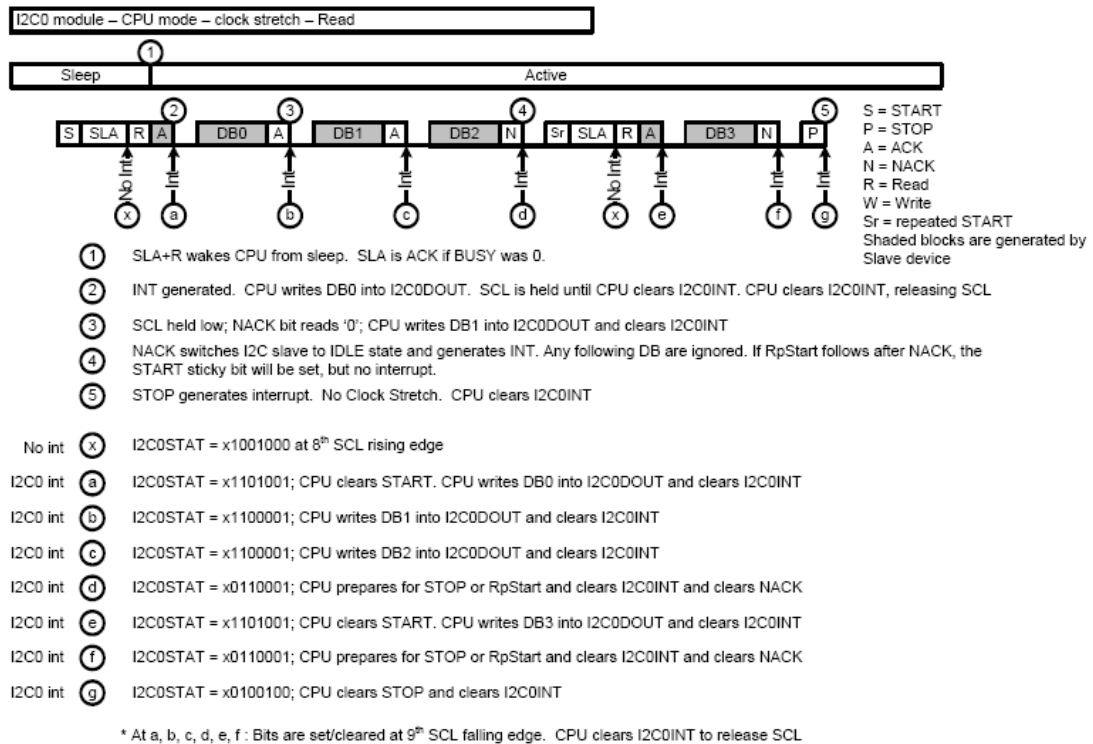
I2C接收波形图:



I2C 写时序图:



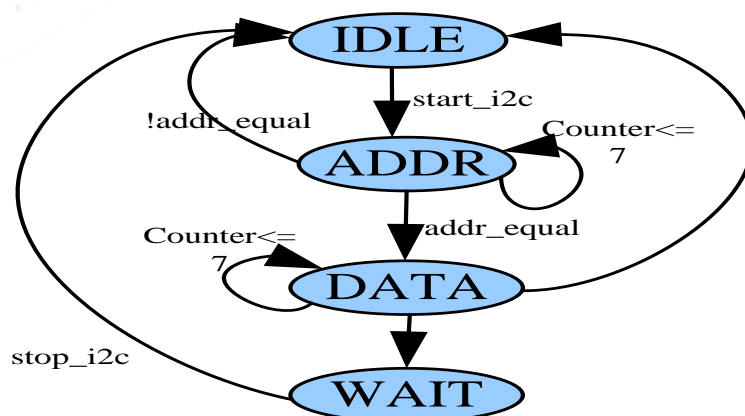
I2C 读时序图:



从上图我们可以看出，I2C 协议自动进行 Clock Stretching 在每个发送或接收的 ACK 或 NACKbit 的 SCL 时钟下降沿。此时，我们将相应的状态位拉高以告知 CPU 取数据或发送数据。CPU 在进行完上述动作后会将相应状态位拉低以结束 Clock Stretching，进入下一个步骤。

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。



10.1 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。

10.2 I2C 寄存器

I2C外设接口不能在Power on后直接使用，我们必须要先配置相应的寄存器来使能I2C。

I2C总线有5个寄存器，分别是FCTR、I2C_DIN、I2C_DOUT、I2C_SLAD、I2C_STAT。

10.2.1 I2C 接受数据寄存器（I2C_DIN）

寄存器BAH：I2C接受数据寄存器（I2C_DIN）

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
I2C_DIN[7:0]							
bit7				bit0			

图注：

R = 可读位 W = 可写位 U = 未实现位，读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7 I2C_DIN[7:0]: I2C 接口从 master 端接收的数据

10.2.2 I2C 发送数据寄存器（I2C_DOUT）

寄存器BBH：I2C发送数据寄存器（I2C_DOUT）

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
I2C_DOUT [7:0]							
bit7				bit0			

图注：

R = 可读位 W = 可写位 U = 未实现位，读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7 I2C_DOUT [7:0]: I2C 发送数据到 master 端

10.2.3 I2C 从机地址寄存器（I2C_SLAD）

寄存器BCH：I2C从机地址寄存器（I2C_SLAD）

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
I2C_SLAD [6:0]							
bit7				bit0			

图注：

R = 可读位 W = 可写位 U = 未实现位，读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7 未实现位：读为0
 bit6-0 I2C_SLAD[6:0]：I2C 从机地址

10.2.4 I2C 状态寄存器 (I2C_STAT)

寄存器BDH：I2C状态寄存器 (I2C_STAT)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
DMOD	ACTIVE	I2CINT	NACK	START	STOP	DATA	STRETCH
bit7							bit0

图注：

R = 可读位 W = 可写位 U = 未实现位，读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7 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

bit6 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。

bit5 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。

bit4 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。

bit3 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；

bit2 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。

bit1 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。

bit0

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。

11 增强型串行外设接口（SPI）

YS68F9XX(X)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。

11.1 引脚说明

SPI 接口有 4 个信号：SCK/P3.0、MOSI/P3.1、MISO/P3.2、NSS/P3.3 或 SCK/P0.4、MOSI/P0.5、MISO/P0.6、NSS/P0.7，2 组只能选择其中一组使用。

11.1.1 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。

11.1.2 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。

11.1.3 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。

11.1.4 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。

11.2 SPI 通信

11.2.1 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 从器件可以同时从 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 在先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 SPI。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 1 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 2 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 3 给出了一个 4 线主方式主器件和两个从器件的连接图。

11.2.2 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 NSSMD (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 3 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_CNTL3) = 0 且 NSSMD0 (SPI_CNTL2) = 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 2 给出了一个 3 线从器件和一个主器件的连接图。

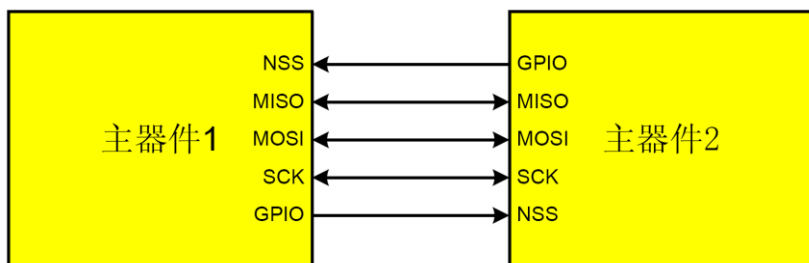


图1 多主方式连接图

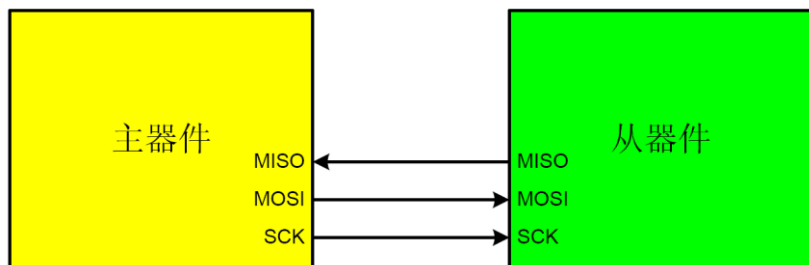


图2 线单主方式和3线单从方式

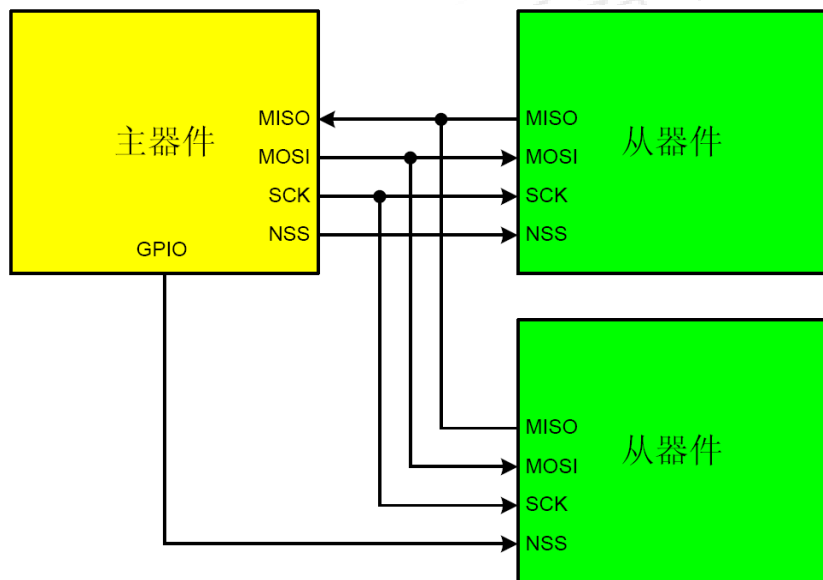


图3 线单主方式和4线从方式连接图

11.3 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。

11.4 SPI 寄存器

11.4.1 SPI 配置寄存器 (SPICFG)

寄存器A1H: SPI配置寄存器 (SPICFG)

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
SPIBSY	MSTINT	CKPHA	CKPOL	SLVSEL	NSSIN	SRMT	RXBMT
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7

SPIBSY:

SPI 工作中

此位在 SPI 工作时，会设置为 1

bit6

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

bit5

CKPHA: SPI 时钟相位选择位

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

bit4

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

bit3

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。

bit2	NSSIN: NSS 时时数据输入
bit1	SRMT: 移位寄存器空标志(只在从机模式时有效)
bit0	RXBMT: 接收暂存器空标志(只在从机模式时有效)

11.4.2 SPI 控制寄存器（SPICTL）

寄存器F8H: SPI控制寄存器（SPICTL）

R -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-1	R/W-1	R/W-1
SPIF	WCOL	MODF	RXOVRN	NSSMID[1:0]	TXBMT	SPIEN	SPIF
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7	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
bit6	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
bit5	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
bit4	RXOVRN: 接收 overrun 标志(只在从机模式下有效)
bit3-2	NSSMID[1:0]: 从机选择标志位 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
bit1	TXBMT: 发送暂存器空标志位 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。
bit0	SPIEN: SPI 使能位 0 = 禁止SPI 1 = 使能 SPI

11.4.3 SPI 时钟速率寄存器 (SPISCR)

寄存器A2H: SPI时钟速率寄存器 (SPISCR)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
SCR [7:0]							
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7

SPI 时钟频率

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。

$$f_{SCK} = SYSCLK / 2 * (SPICKR + 1)$$

$$(0 \leq spi_scr \leq 255)$$

例如: 如果SYSCLK = 2MHz, SPICKR = 0x04, 则

$$f_{SCK} = 2000000 / 2 * (4 + 1)$$

$$f_{SCK} = 200 \text{ kHz}$$

11.4.4 SPI 数据寄存器 (SPIDAT)

寄存器A3H: SPI数据寄存器 (SPIDAT)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	U-0	R/W-0
SPIDAT [7:0]							
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7

SPIDAT [7:0]: SPI 发送和接收数据寄存器。

SPIDAT 寄存器用于发送和接收 SPI 数据。在主方式下, 向 SPI_DAT 写入数据时, 数据被放到发送缓冲器并启动发送。读 SPI_DAT 返回接收缓冲器的内容。

12 UART

YS68F9XX(X) 的UART支持8位数据位和9位数据位2种工作模式。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时自动访问发送寄存器；读SBUF0时自动访问接收寄存器，不可能从发送数据寄存器中读数据。

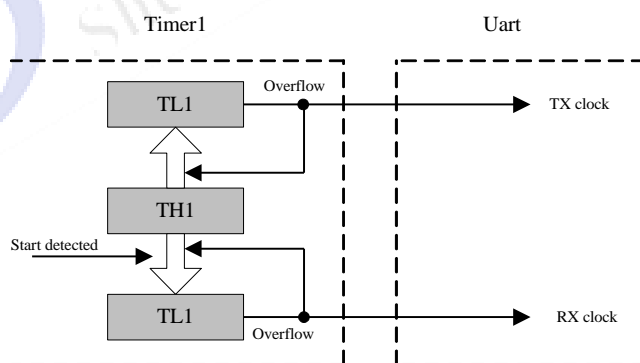
如果UART中断被允许，则每次发送完成（SCON0中的TI0位被置‘1’）或接收到数据字节（SCON0中的RI0位被置‘1’）时将产生中断。当CPU转向中断服务程序时硬件不清除UART0中断标志。中断标志必须用软件清除，这就允许软件查询UART0中断的原因（发送完成或接收完成）。

YS68F9XX(X)的实际值与理论值对比：

波特率	4800	9600	19200	38400
标准波形	208us	104us	52us	26us
YS68F9XXX(X) 波形	210us	100us	53us	27us

12.1 波特时钟与波特率

UART 的波特时钟是由 Timer1 在 8 位自动重载模式下产生的，如下图所示：



TX 时钟由 TL1 产生；RX 时钟是由与 TL1 一样的时钟产生器生成的，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。

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 12M 和 32K，故 Timer1 的驱动时钟源只能是 12M，所以 UART

的波特率由以下二个公式（公式 3.5）决定：

$$(A) \quad T1_Overflow_Rate = \frac{T1_{CLK}}{256 - TH1}$$

$$(B) \quad UartBaudRate = \frac{1}{2} \times T1_Overflow_Rate$$

其中， $T1_{CLK}$ 是驱动 Timer1 的时钟频率，由 12M 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。

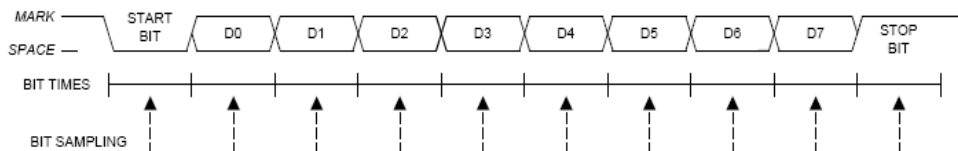
由于波特时钟信号单个调制状态对应的二进制位数为 $\frac{1}{16}$ ，故 Uart 的比特率为：

$$UartBitRate = \frac{1}{16} \times UartBaudClock$$

12.2 工作模式

12.2.1 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时开始数据发送。在发送结束时（停止位开始）发送中断标志 TI0（SCON0.1）被置‘1’。在接收允许位 REN0（SCON0.4）被置‘1’后，数据接收可以在任何时刻开始。

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。

收到停止位后，如果满足下述条件：

1: RI0 必须为逻辑‘0’；

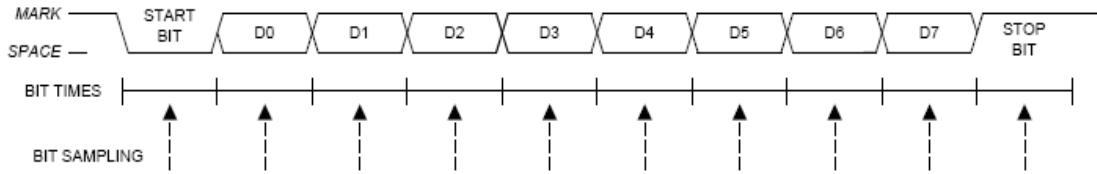
2: 如果 MCE0 为逻辑‘1’，则停止位必须为‘1’。

则数据字节将被装入到接收寄存器 SBUF0。

在发生接收数据溢出的情况下，先接收到的 8 位数据被锁存到 SBUF0，而后面的溢出数据被丢弃。

如果这些条件满足，则 8 位数据被存入 SBUF0，停止位被存入 RB80，RI0 标志被置位。如果这些条件不满足，则不装入 SBUF0 和 RB80，RI0 标志也不会被置‘1’。如果中断被允许，在 TI0 或 RI0 置位时将产生一个中断。

8 位 UART 时序图：



12.2.2 9 位 UART

在9位UART方式，每个数据字节共使用11位：一个起始位、8个数据位（LSB在先）、一个可编程的第九位和一个停止位。

在发送状态下第九位数据由TB80（SCON0.3）中的值决定，由用户软件赋值。在接收状态下，第九数据位进入RB80（SCON0.2），停止位被忽略。

当执行一条向SBUF0寄存器写一个数据字节的指令时开始数据发送。在发送结束时（停止位开始）发送中断标志TI0被置‘1’。

在接收允许位REN0（SCON0.4）被置‘1’后，数据接收可以在任何时刻开始。

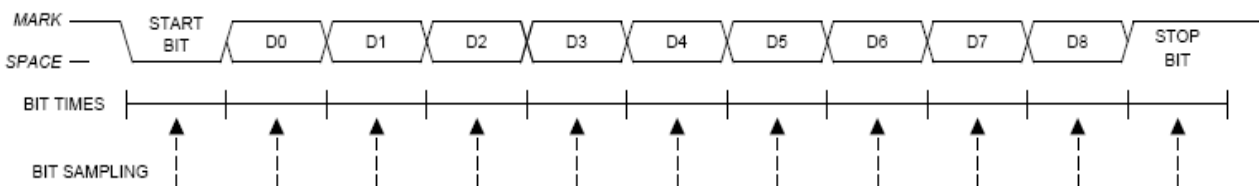
收到停止位后如果满足下述条件则数据字节将被装入到接收寄存器SBUF0=

1:RI0为逻辑‘0’；

2:如果MCE0为逻辑‘1’，则第九位必须为逻辑‘1’（当MCE0为逻辑‘0’时，第九位数据的状态并不重要）。

如果这些条件满足，则8位数据被存入SBUF0，第九位被存入RB80，RI0标志被置位。如果这些条件不满足，则不装入SBUF0和RB80，RI0标志也不会被置‘1’。如果中断被允许，在TI0或RI0置位时将产生一个中断。

9位UART时序图：



12.3 UART 寄存器

12.3.1 UART 控制寄存器 (SCON0)

寄存器98H: UART控制寄存器 (SCON0)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
S0MODE	TX_EN	MCE0	RX_EN	TB80	RB80	TI0	RI0
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

- bit7 S0MODE: UART 的操作模式选择
0=8 位 UART 模式
1=9 位 UART 模式
- bit6 TX_EN: 主机模式使能位
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] (RX_EN) 写 1。
- bit5 MCE0: 多处理器通讯使能位
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。不管第九位数据是 0 还是 1, 都会正常接收, 并产生中断;
1=当第九位数据为 1 时, 将 RI0 置 1, 并产生中断。如果第九位数据是 0, 则会丢弃当前已接收到的字节数据, 并重新等待下一字节的开始。
- bit4 RX_EN: 接收使能位
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。
- bit3 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。
- bit2 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。

bit1

TI0: 发送中断标志位

在发送完一个字节的数据以后，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。

bit0

RI0: 接收中断标志位

在接收完一个字节的数据以后，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。

12.3.2 数据缓存寄存器（SBUF0）

寄存器99H: 数据缓存寄存器（SBUF0）

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
SBUF0[7:0]							
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7

SBUF0[7:0]: 串口数据缓冲位 7:0 (MSB-LSB)

通过此寄存器可以访问 2 个 9 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。

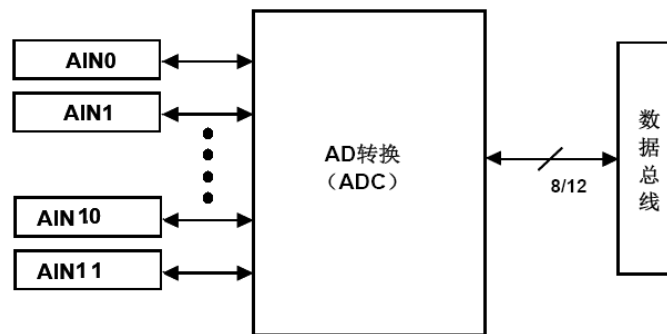
13 模/数转换器（ADC）

特性：

- 支持 8Bit/10Bit/12Bit 分辨率
- 三种参考电压选择：内部参考、VDD、外部输入
- 12 模拟通道输入

SAR ADC 为 YS68F9XX(X)系统中的逐次逼近型模数转换器，集成了跟踪保持电路、可编程窗口检测器和硬件累加器，支持 8bit/10bit/12bit 其转换速度为 300ksps（10bit）或 75ksps（12bit）。ADC 有一种特殊的突发方式（Burst mode），该方式能自动使能 ADC，采集和累加样本值，然后将 ADC 置于低功耗停机模式，不需 CPU 干预。

ADC 模块图：



ADC 相关寄存器：

寄存器	地址	复位值	功能
P1AN	0XCE	0X00	P1 模式控制寄存器
P2AN	0XCF	0X00	P2 模式控制寄存器
P3AN	0XCC	0X00	P3 模式控制寄存器
ADC0CN	0X95	0x00	ADC0 控制寄存器
ADC0CF	0X96	0XF8	ADC0 配置寄存器
ADC0AC	0X97	0X00	ADC0 累加器配置寄存器
ADC0PWR	0XA5	0X0F	ADC0 突发模式上电时间寄存器
ADC0TK	0XA6	0X1E	ADC0 突发模式跟踪时间寄存器
ADC0H	0XAA	0X00	ADC0 数据字高字节寄存器
ADC0L	0XA9	0X00	ADC0 数据字低字节寄存器
ADC0MX	0XA7	0X0F	ADC0 输入通道选择寄存器
REF0CN	0X9F	0X40	电压参考控制寄存器

13.1 AD 寄存器

13.1.1 ADC0 控制寄存器 (ADC0CN)

寄存器95H: ADC0控制寄存器 (ADC0CN)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
AD0EN	BURSTEN	AD0INT	AD0BUSY	-		AD0CM	
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7

AD0EN: ADC0 使能位

1'b0=ADC0 不使能 (低功耗关断)

1'b1=ADC0 使能 (使能且准备开始数据转换)

bit6

BURSTEN: ADC0 突发模式使能位

1'b0=ADC0 突发模式不使能

1'b1=ADC0 突发模式使能

bit5

AD0INT: ADC0 转换结束中断标志位

AD 转换完成 (BURSTEN=0/1) 后由硬件置 1。可触发中断, 需软件清零

bit4

AD0BUSY:

当 AD0CM[1:0]=2'b00, 将 AD0BUSY 写 1, 启动 AD 转换; 写 0, ADC0 不动作

读该 bit, 返回 ADC 当前状态:

1'b1=ADC0 BUSY, 1'b0=ADC0 IDLE

bit3-2

未实现位: 读为 0

bit1-0

AD0CM: ADC0 启动转换模式选择, 指定 ADC 启动转换源

2'b00=ADC0BUSY 写 1

2'b01=Timer1 溢出 (Timer1 8bit 模式时, 低字节溢出即为 Timer1 溢出)

2'b10=Timer2 溢出 (Timer2 8bit 模式时, 低字节溢出即为 Timer1 溢出)

2'b11=Timer3 溢出 (Timer3 8bit 模式时, 低字节溢出即为 Timer1 溢出)

13.1.2 ADC0 配置寄存器 (ADC0CF)

寄存器96H: ADC0配置寄存器 (ADC0CF)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
AD0SC					AD08BE	AD0TM	AMP0GN
bit7					bit0		

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-3

AD0SC: ADC0 SAR 转换时钟分频系数

BURSTEN=0: FCLK 是当前的系统时钟

BURSTEN=1: FCLK 是 12MHz 低功耗振荡时钟 (内部快时钟), 独立于系统时钟

 $CLK_{SAR} = FCLK / AD0SC$ (AD0SC=1~31) $CLK_{SAR} = FCLK / 32$ (AD0SC=0)

bit2

AD08BE: ADC0 8bit 模式使能

1'b0=ADC0 工作在 10bit 模式 (正常模式)

1'b1=ADC0 工作在 8bit 模式

bit1

AD0TM: ADC0 跟踪模式

1'b0=正常跟踪模式: 当 ADC0 使能, 在转换启动信号有效后立即开始 AD 转换

1'b1=延迟跟踪模式: 当 ADC0 使能, 在转换启动信号有效 3 个 SAR 时钟才开始 AD 转换。在延迟的时间里, ADC 允许进行跟踪

bit0

AMP0GN: ADC0 增益控制

0 = 片上 PGA 增益为 1

1 = 片上 PGA 增益为 0.5

13.1.3 ADC0 累加器配置寄存器 (ADC0AC)

寄存器97H: ADC0累加器配置寄存器 (ADC0AC)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
AD012BE	AD0AE	AD0SJST			AD0RPT		
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7

AD012BE: ADC0 12bit 模式使能

1'b0=12bit 模式不使能

1'b1=12bit 模式使能

注: 必须同时满足一下条件 12bit 模式才真正使能:

1、12bit 模式使能

2、10bit 模式使能

3、Burst 模式使能

bit6

AD0AE: ADC0 累加器使能

当突发模式不使能, 使能多次转换以实现累加

1'b0=ADC0H:ADC0L 为最新转换的结果

1'b1=ADC0H:ADC0L 为累加转换的结果; 软件需写 0x0000

到 ADC0H:ADC0L 以清除累加结果

bit5-3

AD0SJST: ADC0 累加器移位和对齐方式

指示 ADC0H:ADC0L 读回的数据格式

3'b000=右对齐; 没有移位

3'b001=右对齐; 右移 1 位

3'b010=右对齐; 右移 2 位

3'b011=右对齐; 右移 3 位

3'b100=左对齐; 没有移位

其他: 右对齐; 没有移位

注: 当重复次数大于 1, 必须右对齐; 当 AD08BE=1'b1, 必须左对齐。

bit2-0

AD0RPT: ADC0 重复次数

选择突发模式的转换和累加次数; 如果突发模式不使能, AD0RPT

必须配置为 3'b000/3'b110/3'b111

3'b000=1 次

3'b001=4 次

3'b010=8 次

3'b011=16 次

3'b100=32 次

3'b101=64 次

其他: 1 次

13.1.4 ADC0 突发模式上电时间寄存器 (ADC0PWR)

寄存器A5H: ADC0突发模式上电时间寄存器 (ADC0PWR)

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
AD0LPM	-	-	AD0PWR				
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7 AD0LPM:

bit6-5 未实现位: 读为 0

bit4-0 AD0PWR: ADC0 突发模式上电时间
设置 ADC0 从低功耗状态到上电的延时

1.当 BURSTEN=0=ADC0 电源状态由 ADC0EN 控制

2.当 BURSTEN=1&ADC0EN=1=ADC0 保持使能并且在转换完成后不进入低功耗模式; 在转换启动信号有效后, 启动立即开始

3.当 BURSTEN=1&ADC0EN=0=ADC0 在转换完成后进入低功耗模式; 在转换启动信号有效后, 需经历可编程的延时后才启动转换; 延时的计算公式如下, 参考图 10 时序:

$$T_{\text{startup}} = (\text{AD0PWR} + 1) * 8 * 84\text{ns}$$

注: YS68F9XX(X)的 burst_clk 频率为 12MHz, 周期约为 84ns

13.1.5 ADC0 突发模式跟踪时间寄存器 (ADC0TK)

寄存器A6H: ADC0突发模式跟踪时间寄存器 (ADC0TK)

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
-	-	AD0TK					
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-6 未实现位: 读为 0

bit5-0 AD0TK: AD0 突发模式跟踪时间
设置 ADC0 突发模式下, 连续转换之间的延时, 参考图 10

ADC0 突发模式跟踪时间 Ttrack:

$$T_{\text{track}} = (64 - \text{AD0TK}) * 84\text{ns}$$

注:

1. 如果 AD0TM 设置为 1, 在开始转换前额外插入 3 个 SAR 时钟周期

2. T_{track} 没有插入到第一次转换，对第一次转换来说，跟踪时间需由**突发模式上电时间** (T_{startup}) 满足
3. YS68F9XX(X)的 burst_clk 频率为 12MHz，周期约为 84ns。

13.1.6 ADC0 数据高字节寄存器 (ADC0H)

寄存器AAH: ADC0数据高字节寄存器 (ADC0H)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADC0H							
bit7							bit0

图注:

R = 可读位 W = 可写位 U = 未实现位，读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7-6

ADC0H: ADC0 数据字高字节

读: 16bit 的 ADC0 累加器数据格式由 AD0SJST[2:0]设置

写: 设置 16bit 的 ADC0 累加器的高字节

13.1.7 ADC0 数据低字节寄存器 (ADC0L)

寄存器A9H: ADC0数据低字节寄存器 (ADC0L)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADC0L							
bit7							bit0

图注:

R = 可读位 W = 可写位 U = 未实现位，读为0
 -n = POR时的值 1 = 置1 0 = 清零 x = 未知

bit7-6

ADC0L: ADC0 数据字低字节

读: 16bit 的 ADC0 累加器数据格式由 AD0SJST[2:0]设置

写: 设置 16bit 的 ADC0 累加器的低字节

13.1.8 ADC0 输入通道选择寄存器 (ADC0MX)

寄存器A7H: ADC0输入通道选择寄存器 (ADC0MX)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
-	-	-	-	AD0MX			
bit7				bit0			

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-4 未实现位: 读为 0

bit3-0 AD0MX: AMUX0 (模拟源选择器选择端)

0000=P1.4 ADC 输入通道 0

0001=P1.5 ADC 输入通道 1

0010=P3.4 ADC 输入通道 2

0011=P3.5 ADC 输入通道 3

0100=P2.0 ADC 输入通道 4

0101=P2.1 ADC 输入通道 5

0110=P2.2 ADC 输入通道 6

0111=P2.3 ADC 输入通道 7

1000=P2.4 ADC 输入通道 8

1001=P2.5 ADC 输入通道 9

1010=P2.6 ADC 输入通道 10

1011=P2.7 ADC 输入通道 11

1100=Ground

1101=Digital Supply Voltage (VREG0 Output, 1.7V Typical)

1110=V_{DD}

1111=关断模式输入电源电压

注:

因 ADC 输入通道 0~ADC 输入通道 11 与其他模拟功能复用 PAD, 当其他模拟功能打开时, 对应的 ADC 输入通道屏蔽。

复用关系如下:

✓ ADC 输入通道 0~1 复用电容触摸通道 4~5

✓ ADC 输入通道 2~3 复用外部慢时钟输入

ADC 输入通道 4~11 复用电容触摸通道 8~15

13.1.9 电压参考控制寄存器（REF0CN）

寄存器9FH：电压参考控制寄存器（REF0CN）

U-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
-	DEM_EN	TESTIREF	REFGND	REFSL		IREFSL	
bit7							bit0

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7 未实现位：读为 0

bit6 DEM_EN：ADC 电容阵列温度译码使能控制
0 = 不使能
1 = 使能

bit5 TESTIREF：选择内部电压参考输出到 PAD 用于测试
0 = 内部电压参考不输出到 PAD
1 = 内部电压参考输出到 PAD

bit4 REFGND：模拟地参考
选择 ADC0 地参考
1'b0=ADC0 地参考是 PAD1/GND pin
1'b1=ADC0 地参考是 PAD48/AGND pin

bit3-2 REFSL：电压参考选择
选择 ADC0 电压参考
00=ADC0 电压参考是 P1.0/VREF pin
01=ADC0 电压参考是 VDD pin
10=ADC0 电压参考是内部 1.8V 数字供电电压
11=ADC0 电压参考是内部电压参考

注：

因 VREF pin 与电容触摸通道 0 复用 P2.7：

- 当电容触摸通道 0 功能打开，REFSL=2'b00 时，电压参考选择 VDD pin
- 当电容触摸通道 0 功能不打开，REFSL=2'b00 时，电压参考选择 VREF pin

bit1-0 IREFSL：当 REFSL=11 时，内部电压参考选择（2V/3V/4V）
00=2V
01=3V
1x: 4V

注：当选择的内部参考电压高于 VDD 时，内部参考电压值会比当前 VDD 电压低 100mA~200 mA

13.2 工作模式

YS68F9XX(X) ADC0 有三个工作模式 8bit/10bit/12bit, ADC0 最大转换速度达 300kps;

BURSTEN=0 时, ADC0 转换时钟 `adc_clk` 从 `sys_clk` 分频而来; BURSTEN=1 时, ADC0 转换时钟 `adc_clk` 从低功耗振荡器时钟 `burst_clk` 分频而来; 分频系数由 ADC0CF 寄存器的 ADC0SC 决定。

13.2.1 转换启动方式

有 4 种 A/D 转换启动方式, 由 ADC0CN 中的 ADC0 转换启动方式位 `AD0CM[1:0]` 的状态决定采用哪一种方式。

转换的触发源有:

1. ADC0BUSY 写 1
2. Timer0 溢出
3. Timer2 溢出
4. Timer3 溢出

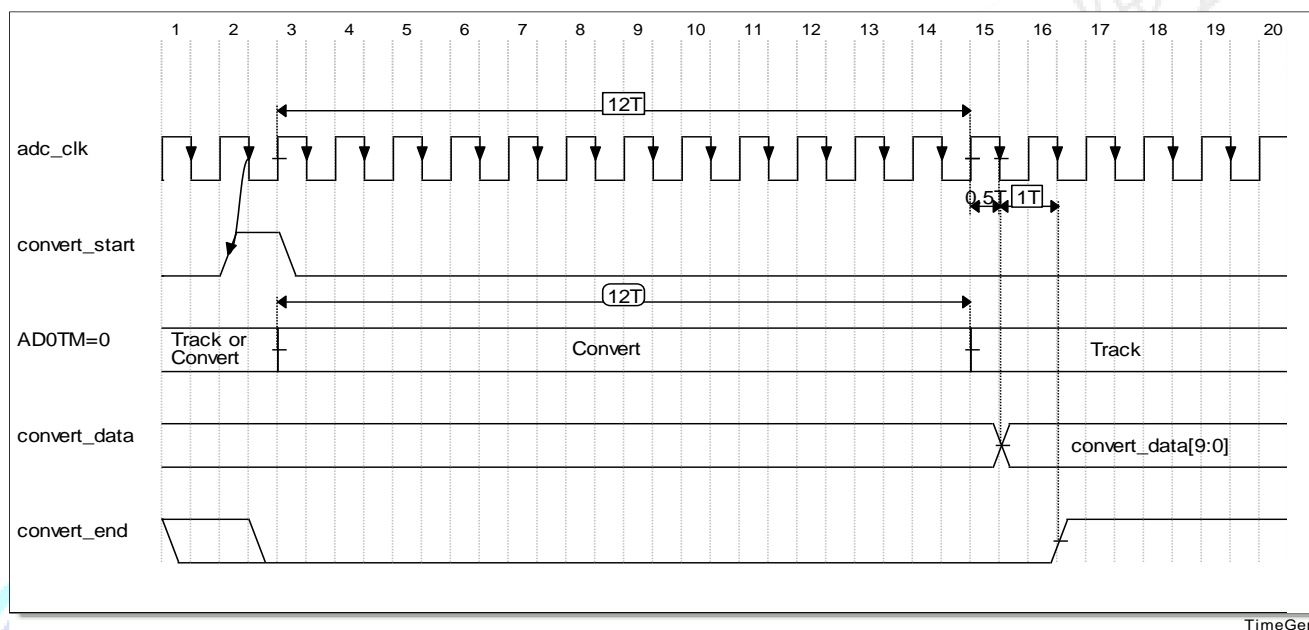
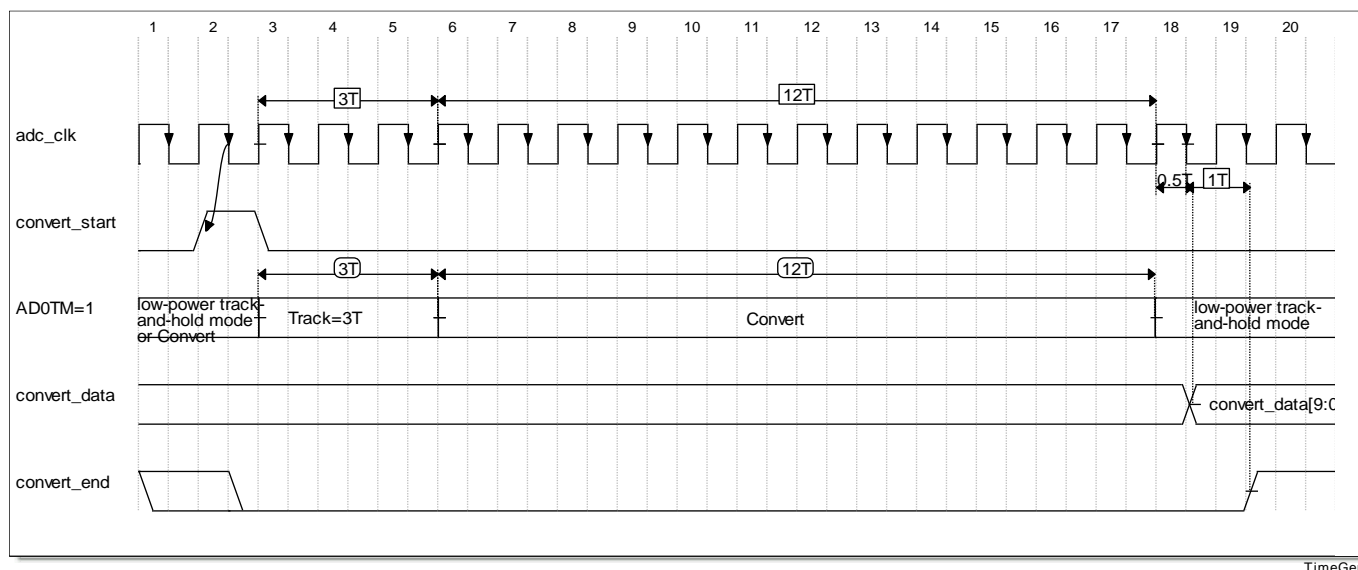
注:

- 1、ADC 转换进行中, 即使新的转换启动条件有效, ADC 并不中断当前转换, 硬件忽略新的转换启动。

13.2.2 跟踪方式

每次 ADC0 转换之前都必须有一个最小的跟踪时间, 以保证转换结果准确。AD0TM bit 控制 ADC0 跟踪和保持模式。当 `AD0EN=1&AD0TM=0`, 除非转换进行中, 否则 ADC0 输入持续被跟踪。当 `AD0EN=1&AD0TM=1`, ADC0 工作在低功耗跟踪和保持模式 (**low-power track-and-hold mode**), 在转换启动信号有效 3 个 SAR 时钟 (`adc_clk`) 才开始 AD 转换。

10bit 模式、BURSTEN=0 (正常模式), A/D 转换时序如下图所示:



YS68F9XX(X) SAR ADC 跟踪转换时序 (10bit 模式, BURSTEN=0)

上图所示, 10bit 模式下, Convert 时间为 12T (从 `adc_clk` 时钟下降沿检测到 `convert_start` 有效开始算起, 第 13 个 `adc_clk` 下降沿 `convert_data` 有效, 第 14 个 `adc_clk` 下降沿 `convert_end` 有效); 8bit 模式下, Convert 时间为 10T。当 12bit 模式使能, Convert 时间仍然由 ADC0CF 寄存器的 AD08BE 位决定。

ADC0 不同配置下的跟踪方式说明如下表所示:

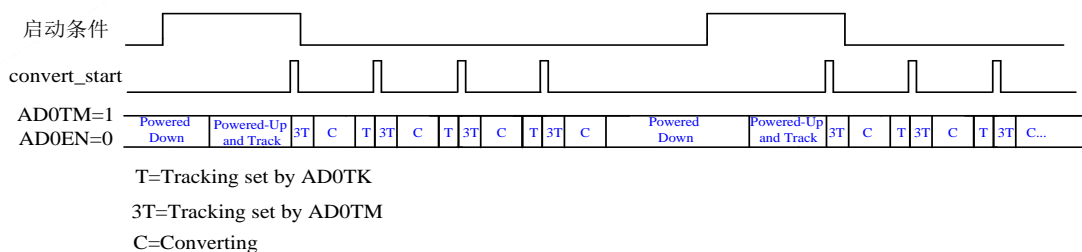
BURSTEN	AD0EN	AD0TM	跟踪方式
0	0	0	ADC0 不工作, 不跟踪
0	0	1	ADC0 不工作, 不跟踪
0	1	0	ADC0 不转换时, 一直跟踪; 在启动条件有效后, <code>convert_start</code> 立即有效并直接开始转换
0	1	1	ADC0 不转换时, 不跟踪 (低功耗跟踪和保持模式); 在

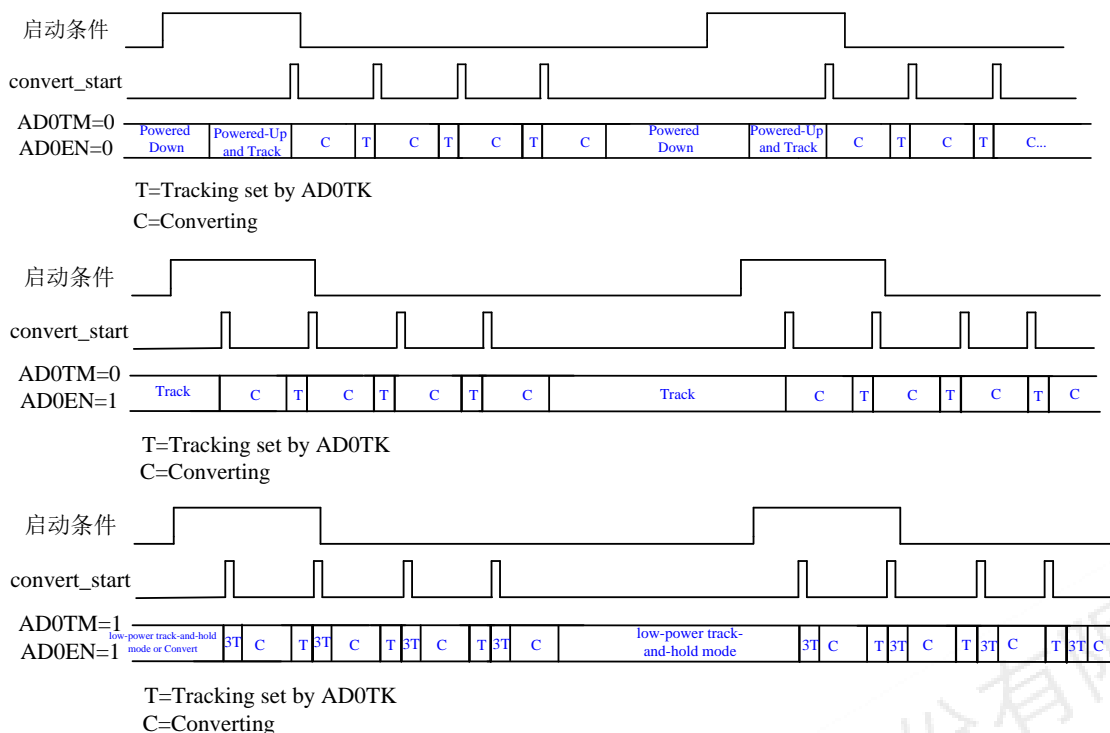
			启动条件有效后，convert_start 立即有效，需插入 3 个 adc_clk 时钟（用于跟踪）后才开始转换
1	0	0	ADC0 不转换时， Powered Down ，不跟踪；在启动条件有效后，计数（AD0PWR+1）个 burst_clk 周期（用于上电和跟踪）convert_start 才有效，convert_start 有效后直接开始转换
1	0	1	ADC0 不转换时， Powered Down ，不跟踪；在启动条件有效后，计数（AD0PWR+1）个 burst_clk 周期（用于上电和跟踪）convert_start 才有效，convert_start 有效后插入 3 个 adc_clk 时钟（用于额外的跟踪）后才开始转换
1	1	0	ADC0 不转换时，一直跟踪；在启动条件有效后，convert_start 立即有效并直接开始转换
1	1	1	ADC0 不转换时，不跟踪（ 低功耗跟踪和保持模式 ）；在启动条件有效后，convert_start 立即有效，需插入 3 个 adc_clk 时钟（用于跟踪）后才开始转换

13.2.3 突发模式

突发模式是一种节省功耗的功能特性，允许 ADC0 在两次转换期间保持低功耗状态。当突发模式被使能时，ADC0 从低功耗状态被唤醒，用内部突发模式时钟（burst_clk 约 12 MHz）累加 1、4、8、16、32 或 64 个采样值，然后又重新进入低功耗状态。由于突发模式时钟独立于系统时钟，ADC0 可以在一个系统时钟周期内完成多次转换并重新进入低功耗状态，即使系统时钟频率很低（YS68F9xxx(x) ADC 设计要求 ADC 工作期间送给 ADC 的 sys_clk 不能停止）。

将 BURSTEN 设置为逻辑 1 即使能突发模式。当工作在突发模式时，AD0EN 控制 ADC0 的空闲电源状态（即 ADC0 不跟踪也不执行转换时进入的状态）。如果 AD0EN 被设置为逻辑 0，ADC0 在每次突发转换后进入断电状态；如果 AD0EN 被设置为逻辑 1，ADC0 在每次突发转换后保持使能状态。每当转换启动条件有效，ADC0 被从其空闲电压状态唤醒。如果 ADC0 被断电（ADC0=0），它会自动上电并等待一个可编程的上电时间，该上电时间由 AD0PWR 位控制。否则，ADC0 会立即启动跟踪和转换。下图给出了重复次数为 4 时的突发模式示例。





YS68F9XX(X) SAR ADC 跟踪转换时序 (BURSTEN=1, AD0RPT=3'b001)

当突发模式被使能时，一次转换启动条件有效将进行多次转换，**ADC 模拟要求每次转换都要有对应的 convert_start 信号，convert_start 信号有效次数等于重复次数**。当突发模式被禁止时，每次转换都需要转换启动条件有效。在这两种情况下，在完成“重复次数”次转换和累加后，ADC0 转换结束中断会被置 1。

突发模式下，跟踪时间由 AD0PWR 和 AD0TK 寄存器配置。

注：

- 1、配置 AD0TM 为 1，在每次转换开始前将插入额外 3 个 SAR 时钟进行跟踪。
- 2、使用突发模式时，不能以高于 sys_clk 频率的 1/4 发出转换启动信号 convert_start。

13.2.4 建立时间要求

在进行一次精确的转换之前需要有一个最小的跟踪时间。该跟踪时间由 AMUX0 的电阻、ADC0 采样电容、外部信号源阻抗及所要求的转换精度决定。

注意：在低功耗跟踪模式 (AD0TM=1)，3 个 SAR 时钟 (adc_clk) 时钟用于在转换启动之前进行跟踪；对大多数应用来说，3 个 SAR 时钟可以满足最小跟踪时间要求；高的外部信号源阻抗将增加跟踪时间需求。

下图给出了等效的 ADC0 输入电路。对于一个给定的建立精度 (SA)，所需要的 ADC0 建立时间可以用方程 4.1 估算。当测量 VDD（相对于 GND）时， R_{TOTAL} 减小到 R_{MUX} 。

$$t = \ln(2n/SA) * R_{TOTAL} C_{SAMPLE} \quad (4.1)$$

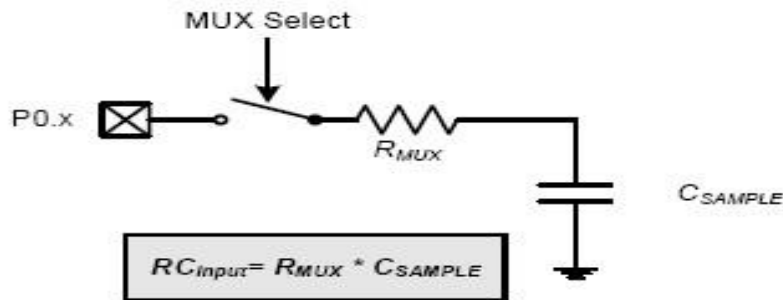
其中：

SA 是建立精度，用一个 LSB 的分数表示（例如，建立精度 0.25 对应 1/4 LSB）；

t 为所需要的建立时间，以秒为单位；

R_{TOTAL} 为 AMUX0 电阻与外部信号源电阻之和；

n 为 ADC 的分辨率，用比特表示（10）。



YS68F9XX(X) SAR ADC 输入等效电路

13.2.5 增益设置

ADC0 增益可设置为 1x 和 0.5x。在 1x 模式下，从 ADC0 读取的满量程由 V_{REF} 决定。在 0.5x 模式下，从 ADC0 读取的满量程由 $V_{REF} * 2$ 决定，这允许 V_{REF} 更低或者输入电压达到 $V_{REF} \sim V_{DD}$ 。

13.2.6 ADC 输出编码格式

ADC0 每次转换完成后，转换结果（数据）存储在 ADC0H 和 ADC0L 寄存器中。数据可以右对齐/左对齐，由 AD0SJST 决定。当重复次数（repeat count）为 1，转换码代表 10bit 无符号整数。模拟输入可为 $0 \sim V_{REF} * 1023/1024$ 。下表举例说明了输出编码方式，ADC0H 和 ADC0L 中无用的 bit 须为 0。

SAR ADC 模块输出编码方式（repeat count=1）

输入电压	右对齐 ADC0H:ADC0L (AD0LJST=3'b000)	左对齐 ADC0H:ADC0L (AD0LJST=3'b100)
$V_{REF} * 1023/1024$	0x03FF	0xFFC0
$V_{REF} * 512/1024$	0x0200	0x8000
$V_{REF} * 256/1024$	0x0100	0x4000
0	0x0000	0x0000

当重复次数大于 1，输出转换编码代表了转换累加的结果，且在最后一次转换完成后更新（先根据对齐方式累

加，最后移位存入寄存器）。重复次数由 ADC0AC 寄存器的 AD0RPT 配置。当重复次数大于 1，ADC 输出必须右对齐（AD0SJST=3'b0xx）；ADC0H 和 ADC0L 中无用的 bit 须为 0。下表列出了不同输入不同重复次数下右对齐的输出。注：当每次采样 ADC 输出同样的值，累加 2^n 次后的值等于单次输出左移 n 位。

SAR ADC 模块输出编码方式（repeat count=4/16/64）

输入电压	重复次数=4	重复次数=16	重复次数=64
VREF*1023/1024	0x0FFC	0x3FF0	0xFFC0
VREF*512/1024	0x0800	0x2000	0x8000
VREF*511/1024	0x07FC	0x1FF0	0x7FC0
0	0x0000	0x0000	0x0000

AD0SJST 用来配置 16bit 累加器的格式。累加的结果可以右移 1、2、3bit。基于过采样和平均原则，当过采样率增加因子 4，有效 ADC 精度增加 1bit。下表列出了在没有 CPU 干预的情况下如何增加 ADC 的有效精度（1、2、3bit）以使 ADC 有效精度达到 11bit、12bit、13bit。

SAR ADC 模块输出编码方式（repeat count=4/16/64，shift right=1/2/3）

输入电压	重复次数=4 右移 1bit 11bit 结果	重复次数=16 右移 2bit 12bit 结果	重复次数=64 右移 3bit 13bit 结果
VREF*1023/1024	0x07FE	0x0FFC	0x1FF8
VREF*512/1024	0x0400	0x0800	0x1000
VREF*511/1024	0x03FE	0x07FC	0x0FF8
0	0x0000	0x0000	0x0000

13.2.7 8bit/10bit/12bit 模式

设置 ADC08BE=1，ADC 处于 8bit 模式，转换完成的时间相对于 10bit 模式少了 2 个 SAR 时钟周期，降低了总体功耗。8bit 模式下，读寄存器 ADC0L 返回 ADC 转换启动前 ADC0L 的值（AD0SJST 必须配置为左对齐，convert_data 高 8bit 有效，低 2bit 始终为 0）。

ADC 还提供 12bit 精度，当配置为 12bit 转换精度，ADC 表现为 4 次 10bit 转换（每个转换的参考电压不同），组合转换结果为 12bit 值。配置 AD012BE=1 且重复 4 次的突发模式，12bit 模式使能。因为 12bit 转换的结果是由 4 次 10bit 转换的结果组合而成的，最大输出值是 $4*1023=4092$ 。为了提高精度，突发模式重复次数可以配置为 4 的倍数，举例来说，如重复次数为 16，ADC0 输出为 14bit（4 个 12bit 之和），具有 13bit 有效精度。

13.2.8 低功耗模式（Low Power Mode）

当 ADC 工作在低速 SAR 时钟时，SAR 转换提供了低功耗模式减少了电流消耗。配置 AD0LPM=1，使能低功耗模式。一般来说，当 SAR 转换时钟频率低于 4MHz 时，推荐使用低功耗模式。

13.2.9 ADC0 模拟多路选择器

AMUX0 选择 ADC 的输入通道。可参考寄存器 ADC0MX。

14 触摸按键（Touch Key）

YS68F9XX(X)系列芯片集成了具有汇春科技专利的 PureTouch 低功耗电容触摸技术，多达 16 通道的自电容触摸传感电路，可同时采样各个通道，软件独立调节各路触摸灵敏度，适用于各种触摸按键、滑条、滚轮的应用案例。

特性：

- 外围不需任何原器件
- 内部软件设置触摸灵敏度
- 近距离、多角度手机或对讲机干扰情况下，触摸响应灵敏度及可靠性不受影响
- 超低功耗
- 自动更新触摸门限以达到自适应检测功能

注：触摸使用手册请参阅《SelfCapLIB_CODE_V1.2.4.pdf》。

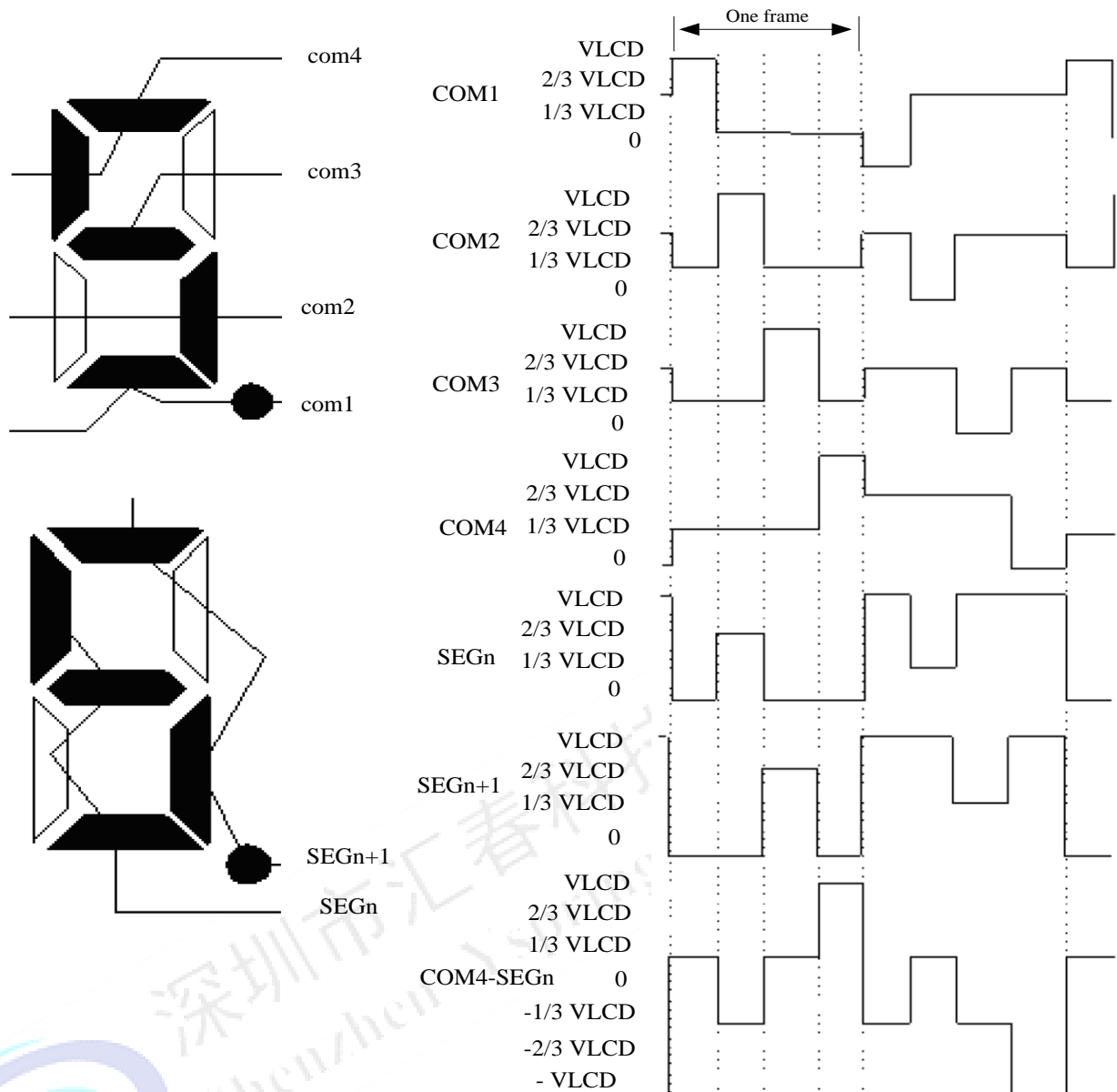
15 LCD/LED 驱动

15.1 LCD 驱动

特性:

- 仅支持 LCD 1/3 BIAS;
- 支持 LCD 3 Com*29 Seg、4 Com*28 Seg、5 Com*27 Seg、6 Com*26SEG、8 Com*24 Seg, 可选;
- 支持 LCD/LED 显示频率预分频 1/2、1/4、1/8 或 1/16, 最低 20Hz, 最高 128Hz;
- LCD 占空比为 1/3、1/4、1/5、1/6 或 1/8;
- LCD/LED 存储阵列, MCU 以 XRAM 方式访问;
- 支持 LCD/LED 功能, 可选 LCD 或 LED, 不能同时使用。

LCD 波形图 (1/4 DUTY, 1/3 BIAS)



15.2 LCD RAM 配置

LCD 1/3 Duty, 1/3Bias(COM1-3, SEG13-29)

XRAM	7	6	5	4	3	2	1	0
Addr.	COM8	COM7	COM6	COM5	COM4	COM3	COM2	COM1
0x1120	-	-	-	-	-	SEG1	SEG1	SEG1
0x1121	-	-	-	-	-	SEG2	SEG2	SEG2
0x1122	-	-	-	-	-	SEG3	SEG3	SEG3
0x1123	-	-	-	-	-	SEG4	SEG4	SEG4
0x1124	-	-	-	-	-	SEG5	SEG5	SEG5
0x1125	-	-	-	-	-	SEG6	SEG6	SEG6
0x1126	-	-	-	-	-	SEG7	SEG7	SEG7
0x1127	-	-	-	-	-	SEG8	SEG8	SEG8
0x1128	-	-	-	-	-	SEG9	SEG9	SEG9

0x1129	-	-	-	-	-	SEG10	SEG10	SEG10
0x112A	-	-	-	-	-	SEG11	SEG11	SEG11
0x112B	-	-	-	-	-	SEG12	SEG12	SEG12
0x112C	-	-	-	-	-	SEG13	SEG13	SEG13
0x112D	-	-	-	-	-	SEG14	SEG14	SEG14
0x112E	-	-	-	-	-	SEG15	SEG15	SEG15
0x112F	-	-	-	-	-	SEG16	SEG16	SEG16
0x1130	-	-	-	-	-	SEG17	SEG17	SEG17
0x1131	-	-	-	-	-	SEG18	SEG18	SEG18
0x1132	-	-	-	-	-	SEG19	SEG19	SEG19
0x1133	-	-	-	-	-	SEG20	SEG20	SEG20
0x1134	-	-	-	-	-	SEG21	SEG21	SEG21
0x1135	-	-	-	-	-	SEG22	SEG22	SEG22
0x1136	-	-	-	-	-	SEG23	SEG23	SEG23
0x1137	-	-	-	-	-	SEG24	SEG24	SEG24
0x1138	-	-	-	-	-	SEG25	SEG25	SEG25
0x1139	-	-	-	-	-	SEG26	SEG26	SEG26
0x113A	-	-	-	-	-	SEG27	SEG27	SEG27
0x113B	-	-	-	-	-	SEG28	SEG28	SEG28
0x113C	-	-	-	-	-	SEG29	SEG29	SEG29

LCD 1/4 Duty, 1/3Bias(COM1-4, SEG8-28)

XRAM	7	6	5	4	3	2	1	0
Addr.	COM8	COM7	COM6	COM5	COM4	COM3	COM2	COM1
0x1120	-	-	-	-	SEG1	SEG1	SEG1	SEG1
0x1121	-	-	-	-	SEG2	SEG2	SEG2	SEG2
0x1122	-	-	-	-	SEG3	SEG3	SEG3	SEG3
0x1123	-	-	-	-	SEG4	SEG4	SEG4	SEG4
0x1124	-	-	-	-	SEG5	SEG5	SEG5	SEG5
0x1125	-	-	-	-	SEG6	SEG6	SEG6	SEG6
0x1126	-	-	-	-	SEG7	SEG7	SEG7	SEG7
0x1127	-	-	-	-	SEG8	SEG8	SEG8	SEG8
0x1128	-	-	-	-	SEG9	SEG9	SEG9	SEG9
0x1129	-	-	-	-	SEG10	SEG10	SEG10	SEG10

0x112A	-	-	-	-	SEG11	SEG11	SEG11	SEG11
0x112B	-	-	-	-	SEG12	SEG12	SEG12	SEG12
0x112C	-	-	-	-	SEG13	SEG13	SEG13	SEG13
0x112D	-	-	-	-	SEG14	SEG14	SEG14	SEG14
0x112E	-	-	-	-	SEG15	SEG15	SEG15	SEG15
0x112F	-	-	-	-	SEG16	SEG16	SEG16	SEG16
0x1130	-	-	-	-	SEG17	SEG17	SEG17	SEG17
0x1131	-	-	-	-	SEG18	SEG18	SEG18	SEG18
0x1132	-	-	-	-	SEG19	SEG19	SEG19	SEG19
0x1133	-	-	-	-	SEG20	SEG20	SEG20	SEG20
0x1134	-	-	-	-	SEG21	SEG21	SEG21	SEG21
0x1135	-	-	-	-	SEG22	SEG22	SEG22	SEG22
0x1136	-	-	-	-	SEG23	SEG23	SEG23	SEG23
0x1137	-	-	-	-	SEG24	SEG24	SEG24	SEG24
0x1138	-	-	-	-	SEG25	SEG25	SEG25	SEG25
0x1139	-	-	-	-	SEG26	SEG26	SEG26	SEG26
0x113A	-	-	-	-	SEG27	SEG27	SEG27	SEG27
0x113B	-	-	-	-	SEG28	SEG28	SEG28	SEG28
0x113C	-	-	-	-	-	-	-	-

LCD 1/5 Duty, 1/3Bias(COM1-5, SEG11-27)

XRAM	7	6	5	4	3	2	1	0
Addr.	COM8	COM7	COM6	COM5	COM4	COM3	COM2	COM1
0x1120	-	-	-	SEG1	SEG1	SEG1	SEG1	SEG1
0x1121	-	-	-	SEG2	SEG2	SEG2	SEG2	SEG2
0x1122	-	-	-	SEG3	SEG3	SEG3	SEG3	SEG3
0x1123	-	-	-	SEG4	SEG4	SEG4	SEG4	SEG4
0x1124	-	-	-	SEG5	SEG5	SEG5	SEG5	SEG5
0x1125	-	-	-	SEG6	SEG6	SEG6	SEG6	SEG6
0x1126	-	-	-	SEG7	SEG7	SEG7	SEG7	SEG7
0x1127	-	-	-	SEG8	SEG8	SEG8	SEG8	SEG8
0x1128	-	-	-	SEG9	SEG9	SEG9	SEG9	SEG9
0x1129	-	-	-	SEG10	SEG10	SEG10	SEG10	SEG10
0x112A	-	-	-	SEG11	SEG11	SEG11	SEG11	SEG11

0x112B	-	-	-	SEG12	SEG12	SEG12	SEG12	SEG12
0x112C	-	-	-	SEG13	SEG13	SEG13	SEG13	SEG13
0x112D	-	-	-	SEG14	SEG14	SEG14	SEG14	SEG14
0x112E	-	-	-	SEG15	SEG15	SEG15	SEG15	SEG15
0x112F	-	-	-	SEG16	SEG16	SEG16	SEG16	SEG16
0x1130	-	-	-	SEG17	SEG17	SEG17	SEG17	SEG17
0x1131	-	-	-	SEG18	SEG18	SEG18	SEG18	SEG18
0x1132	-	-	-	SEG19	SEG19	SEG19	SEG19	SEG19
0x1133	-	-	-	SEG20	SEG20	SEG20	SEG20	SEG20
0x1134	-	-	-	SEG21	SEG21	SEG21	SEG21	SEG21
0x1135	-	-	-	SEG22	SEG22	SEG22	SEG22	SEG22
0x1136	-	-	-	SEG23	SEG23	SEG23	SEG23	SEG23
0x1137	-	-	-	SEG24	SEG24	SEG24	SEG24	SEG24
0x1138	-	-	-	SEG25	SEG25	SEG25	SEG25	SEG25
0x1139	-	-	-	SEG26	SEG26	SEG26	SEG26	SEG26
0x113A	-	-	-	SEG27	SEG27	SEG27	SEG27	SEG27
0x113B	-	-	-	-	-	-	-	-
0x113C	-	-	-	-	-	-	-	-

LCD 1/6 Duty, 1/3Bias(COM1-6, SEG10-26)

XRAM	7	6	5	4	3	2	1	0
Addr.	COM8	COM7	COM6	COM5	COM4	COM3	COM2	COM1
0x1120	-	-	SEG1	SEG1	SEG1	SEG1	SEG1	SEG1
0x1121	-	-	SEG2	SEG2	SEG2	SEG2	SEG2	SEG2
0x1122	-	-	SEG3	SEG3	SEG3	SEG3	SEG3	SEG3
0x1123	-	-	SEG4	SEG4	SEG4	SEG4	SEG4	SEG4
0x1124	-	-	SEG5	SEG5	SEG5	SEG5	SEG5	SEG5
0x1125	-	-	SEG6	SEG6	SEG6	SEG6	SEG6	SEG6
0x1126	-	-	SEG7	SEG7	SEG7	SEG7	SEG7	SEG7
0x1127	-	-	SEG8	SEG8	SEG8	SEG8	SEG8	SEG8
0x1128	-	-	SEG9	SEG9	SEG9	SEG9	SEG9	SEG9
0x1129	-	-	SEG10	SEG10	SEG10	SEG10	SEG10	SEG10
0x112A	-	-	SEG11	SEG11	SEG11	SEG11	SEG11	SEG11

0x112B	-	-	SEG12	SEG12	SEG12	SEG12	SEG12	SEG12
0x112C	-	-	SEG13	SEG13	SEG13	SEG13	SEG13	SEG13
0x112D	-	-	SEG14	SEG14	SEG14	SEG14	SEG14	SEG14
0x112E	-	-	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15
0x112F	-	-	SEG16	SEG16	SEG16	SEG16	SEG16	SEG16
0x1130	-	-	SEG17	SEG17	SEG17	SEG17	SEG17	SEG17
0x1131	-	-	SEG18	SEG18	SEG18	SEG18	SEG18	SEG18
0x1132	-	-	SEG19	SEG19	SEG19	SEG19	SEG19	SEG19
0x1133	-	-	SEG20	SEG20	SEG20	SEG20	SEG20	SEG20
0x1134	-	-	SEG21	SEG21	SEG21	SEG21	SEG21	SEG21
0x1135	-	-	SEG22	SEG22	SEG22	SEG22	SEG22	SEG22
0x1136	-	-	SEG23	SEG23	SEG23	SEG23	SEG23	SEG23
0x1137	-	-	SEG24	SEG24	SEG24	SEG24	SEG24	SEG24
0x1138	-	-	SEG25	SEG25	SEG25	SEG25	SEG25	SEG25
0x1139	-	-	SEG26	SEG26	SEG26	SEG26	SEG26	SEG26
0x113A	-	-	-	-	-	-	-	-
0x113B	-	-	-	-	-	-	-	-
0x113C	-	-	-	-	-	-	-	-

LCD 1/8 Duty, 1/3Bias(COM1–8, SEG8-24)

XRAM	7	6	5	4	3	2	1	0
Addr.	COM8	COM7	COM6	COM5	COM4	COM3	COM2	COM1
0x1120	SEG1	SEG1	SEG1	SEG1	SEG1	SEG1	SEG1	SEG1
0x1121	SEG2	SEG2	SEG2	SEG2	SEG2	SEG2	SEG2	SEG2
0x1122	SEG3	SEG3	SEG3	SEG3	SEG3	SEG3	SEG3	SEG3
0x1123	SEG4	SEG4	SEG4	SEG4	SEG4	SEG4	SEG4	SEG4
0x1124	SEG5	SEG5	SEG5	SEG5	SEG5	SEG5	SEG5	SEG5
0x1125	SEG6	SEG6	SEG6	SEG6	SEG6	SEG6	SEG6	SEG6
0x1126	SEG7	SEG7	SEG7	SEG7	SEG7	SEG7	SEG7	SEG7
0x1127	SEG8	SEG8	SEG8	SEG8	SEG8	SEG8	SEG8	SEG8
0x1128	SEG9	SEG9	SEG9	SEG9	SEG9	SEG9	SEG9	SEG9
0x1129	SEG10	SEG10	SEG10	SEG10	SEG10	SEG10	SEG10	SEG10
0x112A	SEG11	SEG11	SEG11	SEG11	SEG11	SEG11	SEG11	SEG11

0x112B	SEG12	SEG12	SEG12	SEG12	SEG12	SEG12	SEG12	SEG12
0x112C	SEG13	SEG13	SEG13	SEG13	SEG13	SEG13	SEG13	SEG13
0x112D	SEG14	SEG14	SEG14	SEG14	SEG14	SEG14	SEG14	SEG14
0x112E	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15
0x112F	SEG16	SEG16	SEG16	SEG16	SEG16	SEG16	SEG16	SEG16
0x1130	SEG17	SEG17	SEG17	SEG17	SEG17	SEG17	SEG17	SEG17
0x1131	SEG18	SEG18	SEG18	SEG18	SEG18	SEG18	SEG18	SEG18
0x1132	SEG19	SEG19	SEG19	SEG19	SEG19	SEG19	SEG19	SEG19
0x1133	SEG20	SEG20	SEG20	SEG20	SEG20	SEG20	SEG20	SEG20
0x1134	SEG21	SEG21	SEG21	SEG21	SEG21	SEG21	SEG21	SEG21
0x1135	SEG22	SEG22	SEG22	SEG22	SEG22	SEG22	SEG22	SEG22
0x1136	SEG23	SEG23	SEG23	SEG23	SEG23	SEG23	SEG23	SEG23
0x1137	SEG24	SEG24	SEG24	SEG24	SEG24	SEG24	SEG24	SEG24
0x1138	-	-	-	-	-	-	-	-
0x1139	-	-	-	-	-	-	-	-
0x113A	-	-	-	-	-	-	-	-
0x113B	-	-	-	-	-	-	-	-
0x113C	-	-	-	-	-	-	-	-

LCD 1/3、1/4、1/5、1/6、1/8 Duty 依次对应的 29SEG、28SEG、27SEG、26SEG 及 24SEG

XRAM	7	6	5	4	3	2	1	0
Addr.	COM8	COM7	COM6	COM5	COM4	COM3	COM2	COM1
0x1120	SEG1	SEG1	SEG1	SEG1	SEG1	SEG1	SEG1	SEG1
0x1121	SEG2	SEG2	SEG2	SEG2	SEG2	SEG2	SEG2	SEG2
0x1122	SEG3	SEG3	SEG3	SEG3	SEG3	SEG3	SEG3	SEG3
0x1123	SEG4	SEG4	SEG4	SEG4	SEG4	SEG4	SEG4	SEG4
0x1124	SEG5	SEG5	SEG5	SEG5	SEG5	SEG5	SEG5	SEG5
0x1125	SEG6	SEG6	SEG6	SEG6	SEG6	SEG6	SEG6	SEG6
0x1126	SEG7	SEG7	SEG7	SEG7	SEG7	SEG7	SEG7	SEG7
0x1127	SEG8	SEG8	SEG8	SEG8	SEG8	SEG8	SEG8	SEG8
0x1128	SEG9	SEG9	SEG9	SEG9	SEG9	SEG9	SEG9	SEG9
0x1129	SEG10	SEG10	SEG10	SEG10	SEG10	SEG10	SEG10	SEG10
0x112A	SEG11	SEG11	SEG11	SEG11	SEG11	SEG11	SEG11	SEG11

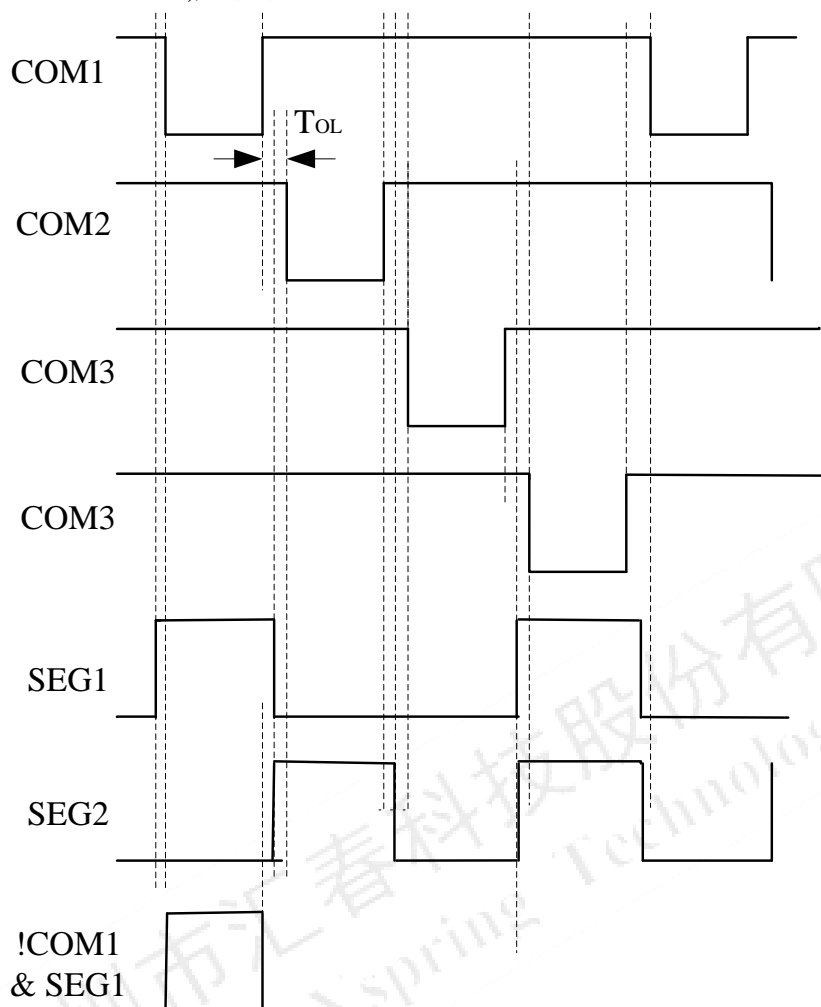
0x112B	SEG12	SEG12	SEG12	SEG12	SEG12	SEG12	SEG12	SEG12
0x112C	SEG13	SEG13	SEG13	SEG13	SEG13	SEG13	SEG13	SEG13
0x112D	SEG14	SEG14	SEG14	SEG14	SEG14	SEG14	SEG14	SEG14
0x112E	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15	SEG15
0x112F	SEG16	SEG16	SEG16	SEG16	SEG16	SEG16	SEG16	SEG16
0x1130	SEG17	SEG17	SEG17	SEG17	SEG17	SEG17	SEG17	SEG17
0x1131	SEG18	SEG18	SEG18	SEG18	SEG18	SEG18	SEG18	SEG18
0x1132	SEG19	SEG19	SEG19	SEG19	SEG19	SEG19	SEG19	SEG19
0x1133	SEG20	SEG20	SEG20	SEG20	SEG20	SEG20	SEG20	SEG20
0x1134	SEG21	SEG21	SEG21	SEG21	SEG21	SEG21	SEG21	SEG21
0x1135	SEG22	SEG22	SEG22	SEG22	SEG22	SEG22	SEG22	SEG22
0x1136	SEG23	SEG23	SEG23	SEG23	SEG23	SEG23	SEG23	SEG23
0x1137	SEG24	SEG24	SEG24	SEG24	SEG24	SEG24	SEG24	SEG24
0x1138	-	-	SEG25	SEG25	SEG25	SEG25	SEG25	SEG25
0x1139	-	-	SEG26	SEG26	SEG26	SEG26	SEG26	SEG26
0x113A	-	-	-	SEG27	SEG27	SEG27	SEG27	SEG27
0x113B	-	-	-	-	SEG28	SEG28	SEG28	SEG28
0x113C	-	-	-	-	-	SEG29	SEG29	SEG29

15.3 LED 驱动器

LED 最大支持 8 Com*8 Seg 共 64 个点的驱动，与 LCD 共用存储器和引脚，共 16 个输出，可通过调节 COM 的占空比来调节 LED 亮度，并可设置共阳或共阴，每个管脚最大提供 20mA 电流。

LED 波形图(LED_MOD=0, 1/4 duty):

LEDCOM=0,共阴极



注：TOL 为 LED COMMON 信号间的重叠时间，取值范围：20us-40us。

15.4 LED RAM 配置

LED 1/4 占空比(LED_C1-4,LED_S1-8)

XRAM Addr.		7	6	5	4	3	2	1	0
0x1120	COM1	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1

0x1121	COM2	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1
0x1122	COM3	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1
0x1123	COM4	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1

LED 1/8 占空比(LED_C1-8,LED_S1-8)

XRAM Addr.		7	6	5	4	3	2	1	0
0x1120	COM1	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1
0x1121	COM2	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1
0x1122	COM3	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1
0x1123	COM4	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1
0x1124	COM5	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1
0x1125	COM6	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1
0x1126	COM7	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1
0x1127	COM8	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1

15.5 LCD/LED 驱动器

15.5.1 显示配置寄存器 (DISP_CFG)

寄存器 FDH: 显示配置寄存器 (DISP_CFG)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	R/W-0	R/W-0
						DISP_MOD	LED_MOD
bit7						bit0	

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-2 未实现位: 读为0

bit2 DISP_MOD: LCD/LED Select control

0 = 启用 LCD 显示驱动程序, 禁用 LED 显示驱动程序 (默认)

1 = 启用 LED 显示驱动程序, 禁用 LCD 显示驱动程序

bit1-0 LED_MOD: LED COMMON mode LED 共模

0 = 共阴极 (default)

1 = 共阳极

15.5.2 显示使能寄存器 (DISP_EN)

寄存器 FEH: 显示使能寄存器 (DISP_EN)

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	R/W-0	R/W-0
DISP_ON	—	DUTY			LCD_FREQ		
bit7							bit0

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7 DISP_MOD: LCD/LED 使能位

0 = 禁用 LCD/LED 驱动程序

1 = 使能 LCD/LED 驱动程序

bit6 未实现位: 读为 0

bit5-3 DUTY :

LCD 占空比配置位

000: 1/3 duty, 1/3 bias, support LCD 29 SEG

001: 1/4 duty, 1/3 bias, support LCD 28 SEG

010: 1/5 duty, 1/3 bias, support LCD 27 SEG

011: 1/6 duty, 1/3 bias, support LCD 26 SEG

100: 1/8 duty, 1/3 bias, support LCD 24 SEG (default)

LED 占空比配置位

xx0: 1/4 duty

xx1: 1/8 duty

bit2-0 LCD_FREQ:

LCD/LED 驱动时钟预刻度为 32768Hz/32

FREQ[2:0]	DIV	Frame Frequency (Hz)				
		1/3 duty	1/4 duty	1/5 duty	1/6 duty	1/8 duty
000	1	341.3	256	204.8	170.7	128
001	1/2	170.7	128	102.4	85.3	64
010	1/4	85.3	64	51.2	42.7	32
011	1/8	42.7	32	25.6	21.3	16
100	1/16	21.3	16	12.8	10.7	8

Frame Frequency = Duty * DIV * 32768 Hz / 32

000: Frame frequency is : duty * 1 * 32768Hz / 32

001: Frame frequency is : duty * 1/2 * 32768Hz / 32 (default)

010: Frame frequency is : duty * 1/4 * 32768Hz / 32
 011: Frame frequency is : duty * 1/8 * 32768Hz / 32
 100: Frame frequency is : duty * 1/16 * 32768Hz / 32

注：当使能 LCD 时，P4 P5 P6 端口自动设置为 LCD 驱动口，不能用作通用 IO 端口，P0 需要将 P0AN 的相应位置 1 才能用作 LCD 的 seg 驱动口，否则 P0 端口即为通用 IO 端口。

15.5.3 模拟控制寄存器（ANA_CTRL）

寄存器 AFH: 模拟控制寄存器（ANA_CTRL）

R/W -0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0	R/W-0	R/W-0
—					EXT_CP	CPCK_CTL	
bit7					bit0		

图注：

R = 可读位

W = 可写位

U = 未实现位，读为0

-n = POR时的值

1 = 置1

0 = 清零

x = 未知

bit7-3 未实现位：读为 0

bit2 EXT_CP: VLCD 电压供电模式
 0 = VLCD 电压由芯片内部 Charge Pump 提供，默认值
 1 = VLCD 电压由片外提供

bit1-0 CPCK_CTL: Charge Pump 分频输入

CP_CK_CTL[1:0]	Charge Pump 频率
00 (Default)	8KHz
01	4KHz
10	2KHz
11	1KHz

16 电气特性

偏置电压下的环境温度.....	-40°C 至+85°C
储存温度.....	-65°C 至+150°C
VDD 引脚相对于 VSS 的电压.....	-0.3V 至+6.5V
MCLR 引脚相对于 Vss 的电压.....	-0.3V 至+13.5V
所有其他引脚相对于 VSS 的电压.....	-0.3V 至(VDD+0.3V)

注意： 如果运行条件超过了上述“绝对极限参数值”，即可能对器件造成永久性损坏。上述值仅为运行条件的极大值，我们不建议器件运行在该规范范围以外。器件长时间工作在绝对极限参数条件下，其稳定性可能受到影响。

直流电器特性 (VDD=1.8V-5.5V, GND=0V, T_A=+25℃, 除非另有说明)

直流特性		标准工作条件 工作温度 -40℃≤T _A ≤+85℃				
符号	特性	最小值	典型值 ⁽¹⁾	最大值	单位	条件
VDD	电源电压	1.8		5.5	V	
VDR	RAM 数据保持电压 ⁽²⁾	—	0.8*	—	V	器件处于休眠模式
VPOR	Vdd 起始电压确保能够产生上电复位信号	—	V _{SS}	—	V	
SVDD	Vdd 上升速率确保能够产生上电复位信号	0.05*	—	—	V/ms	
IDD 工作电流 ⁽³⁾		—	2.0	—	mA	电压 5V, 系统内部时钟为 12MHZ, 所有 IO 输出为低, 外设默认
IPD	掉电流 ⁽⁴⁾	—	5	—	uA	WDT Disable VDD=5V
ΔIWDT	WDT 电流 ⁽⁴⁾	—	1	—	uA	VDD=5V
VIL	输入低电平电压	VSS	1.2	—	V	3V SCHMITT
		VSS	1.76	—		5V SCHMITT
VIH	输入高电平电压	—	2.1	VDD	V	3V SCHMITT
		—	2.76	VDD		5V SCHMITT
IOL	正常模式输出灌电流	—	9	—	mA	VOL=0.7V 3V
		—	16	—		VOL=0.7V 5V
	增强模式输出灌电流	—	28	—		VOL=0.7V 3V
		—	45	—		VOL=0.7V 5V
IOH	正常模式输出拉电流	—	—	—	mA	VOH=1.7V 3V
		—	—	—		VOH=3.6V 5V
	增强模式输出拉电流	—	—	—		VOH=2.1V 3V
		—	—	—		VOH=3.6V 5V
Vlvr	低电压复位电压	1.7 -20%	1.7	1.7 +20%	V	
		2.1 -20%	2.1	2.1 +20%		
		2.3 -20%	2.3	2.3 +20%		
		2.1 -20%	3.8	3.8 +20%		

Rpu	上拉电阻	—	30	—	K	3V
		—	16	—		5V

注：“—”表示没有，或待定。

(1) 典型栏中数据均为 25℃ 条件下值，此部分数据仅供参考。

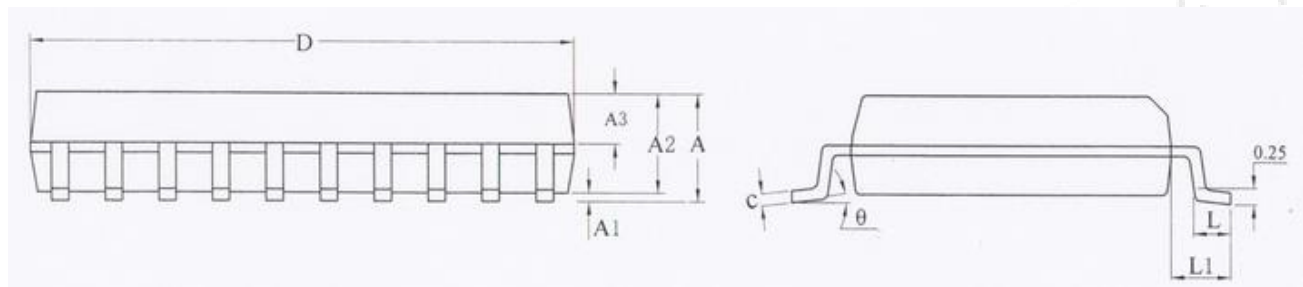
(2) 该电压是保证不丢失 RAM 数据的最小 VDD。

(3) 工作电流主要随工作电压和频率而变化。其它因素，如总线负载、总线速率、内部代码执行模式和温度也会影响电流消耗。

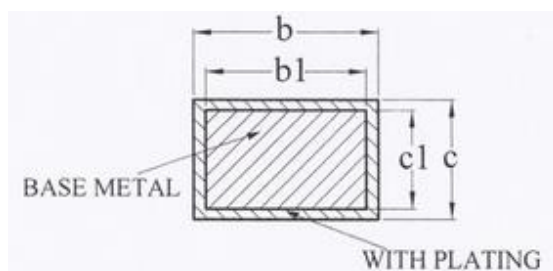
(4) 掉电电流是在器件休眠时，所有 I/O 引脚都处于高阻态并且连接到 Vdd 或 Vss 时测得。

17 封装尺寸

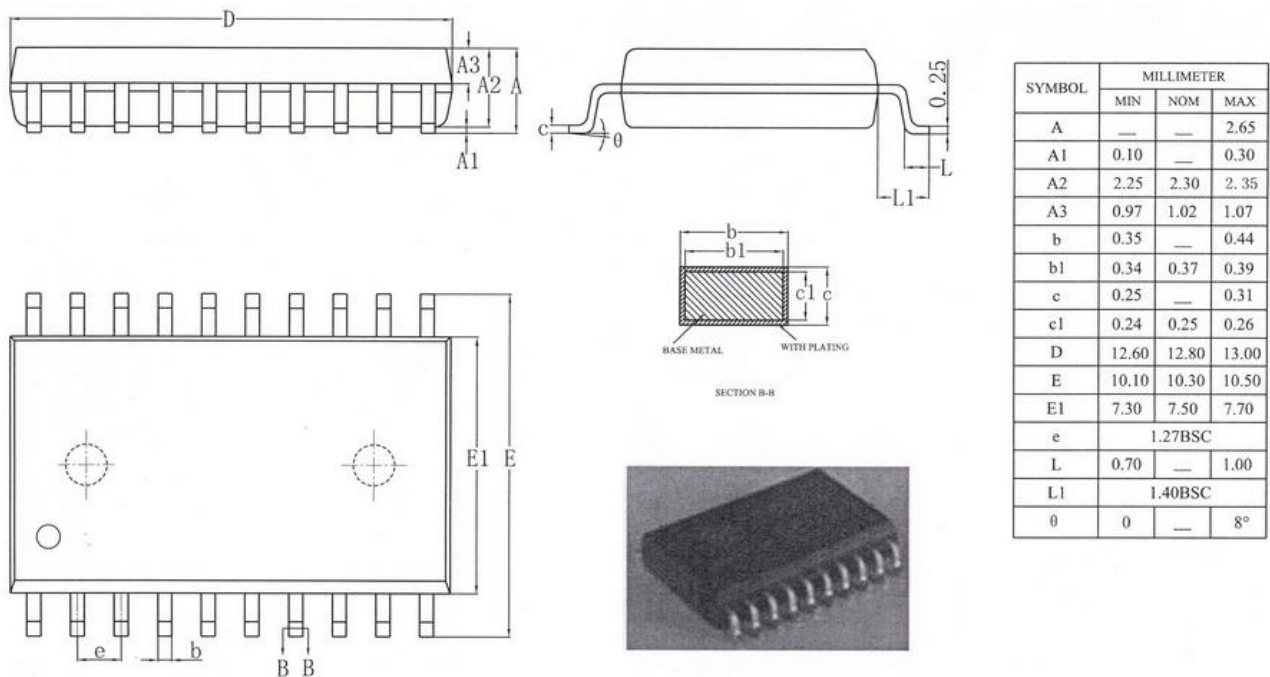
17.1 SSOP20



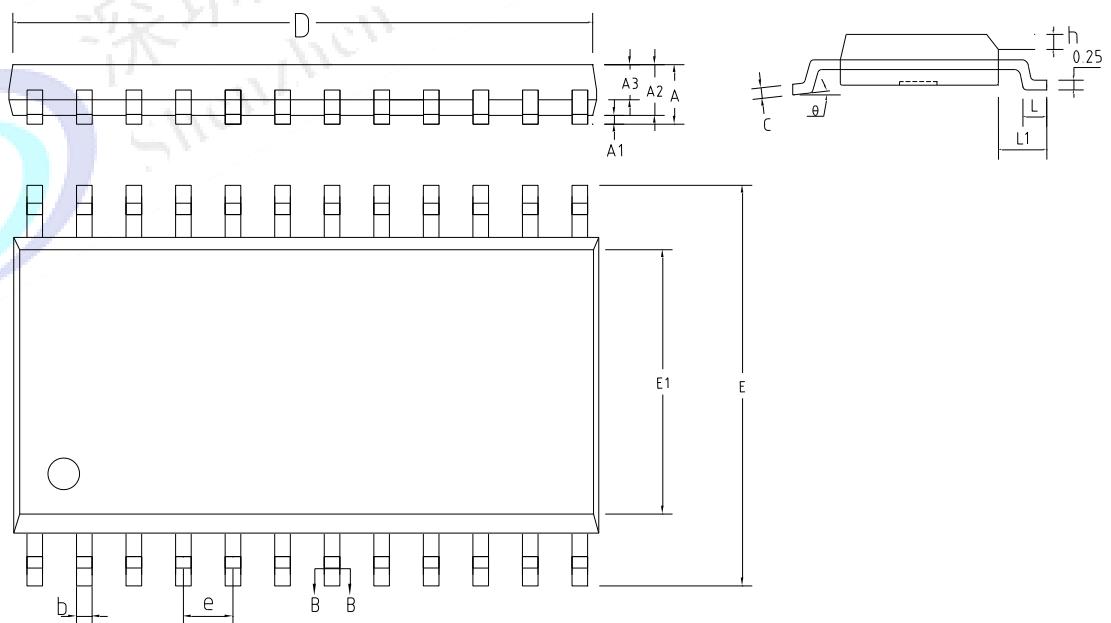
SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.85
A1	0.05	—	0.25
A2	1.40	1.50	1.60
A3	0.62	0.67	0.72
b	0.29	—	0.37
b1	0.28	0.30	0.33
c	0.15	—	0.20
c1	0.14	0.15	0.16
D	7.00	7.20	7.40
E	7.60	7.80	8.00
E1	5.10	5.30	5.50
e	0.65BSC		
L	0.75	—	1.05
L1	1.25BSC		
θ	0	—	8°
L/P 载体尺寸 (mil)	145*169		



17.2 SOP24



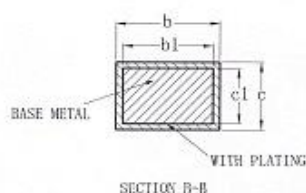
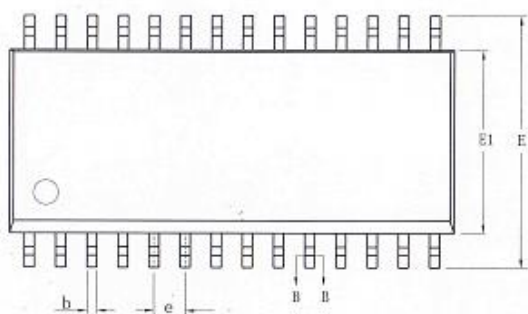
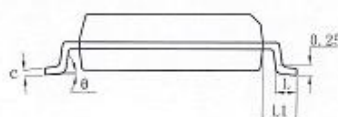
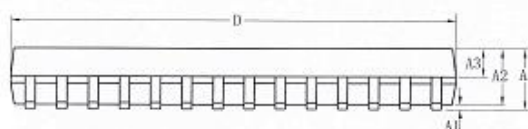
17.3 SSOP24



SYMBOLS	MIN	NOR	MAX	SYMBOLS	MIN	NOR	NOR
	(mm)				(mm)		
A	—	—	1. 75	E1	3. 80	3. 90	4. 00

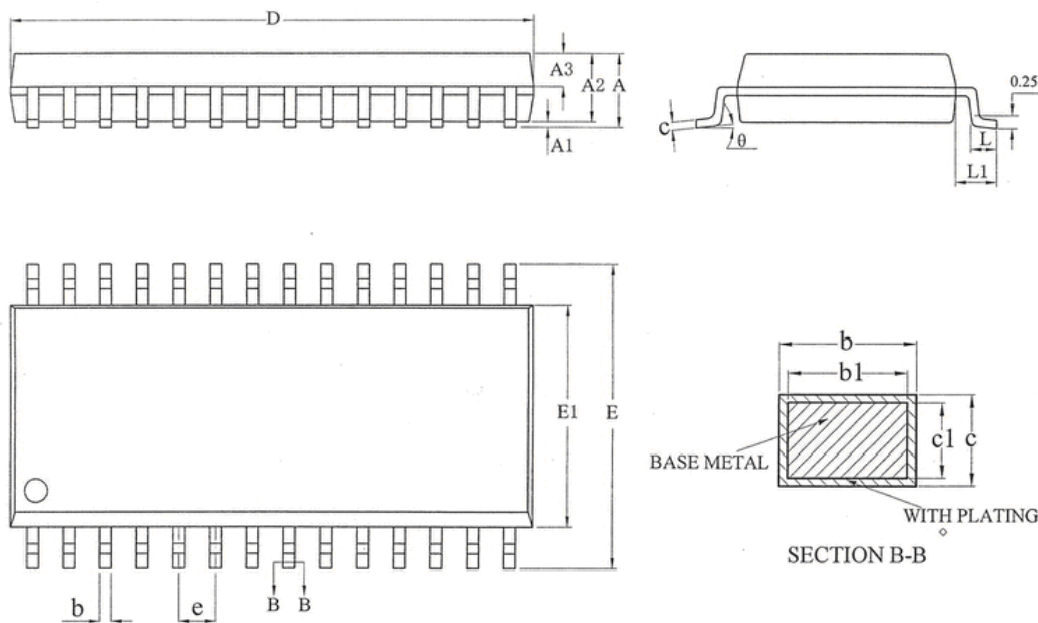
A1	0.10	0.15	0.25	e	0.635BSC		
A2	1.30	1.40	1.50	c	0.20	—	0.24
A3	0.60	0.65	0.70	L	0.50	—	0.80
D	8.55	8.65	8.75	L1	1.05REF		
h	0.30	—	0.50	c1	0.19	0.20	0.21
E	5.80	6.00	6.20	θ	0	—	8°
				b	0.23	—	0.31

17.4 SOP28



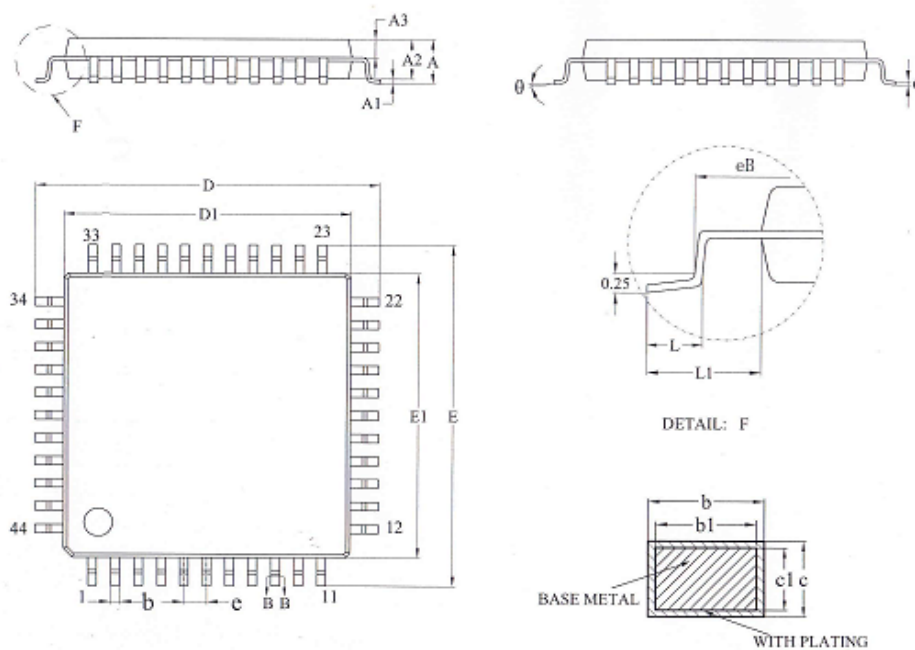
SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	2.65
A1	0.10	—	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
b	0.39	—	0.47
b1	0.38	0.41	0.44
c	0.25	—	0.29
c1	0.24	0.25	0.26
D	17.99	18.00	18.10
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	—	1.00
L1	1.40REF		
θ	0	—	8°

17.5 SSOP28



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	2.00
A1	0.05	—	0.25
A2	1.65	1.75	1.85
A3	0.75	0.80	0.85
b	0.29	—	0.37
b1	0.28	0.30	0.33
c	0.15	—	0.20
c1	0.14	0.15	0.16
D	10.00	10.20	10.40
E	7.60	7.80	8.00
E1	5.10	5.30	5.50
e	0.65BSC		
L	0.55	0.75	0.95
L1	1.25BSC		
θ	0	—	8°
L/F载体尺寸 (mil)	153*200		

17.6 LQFP44

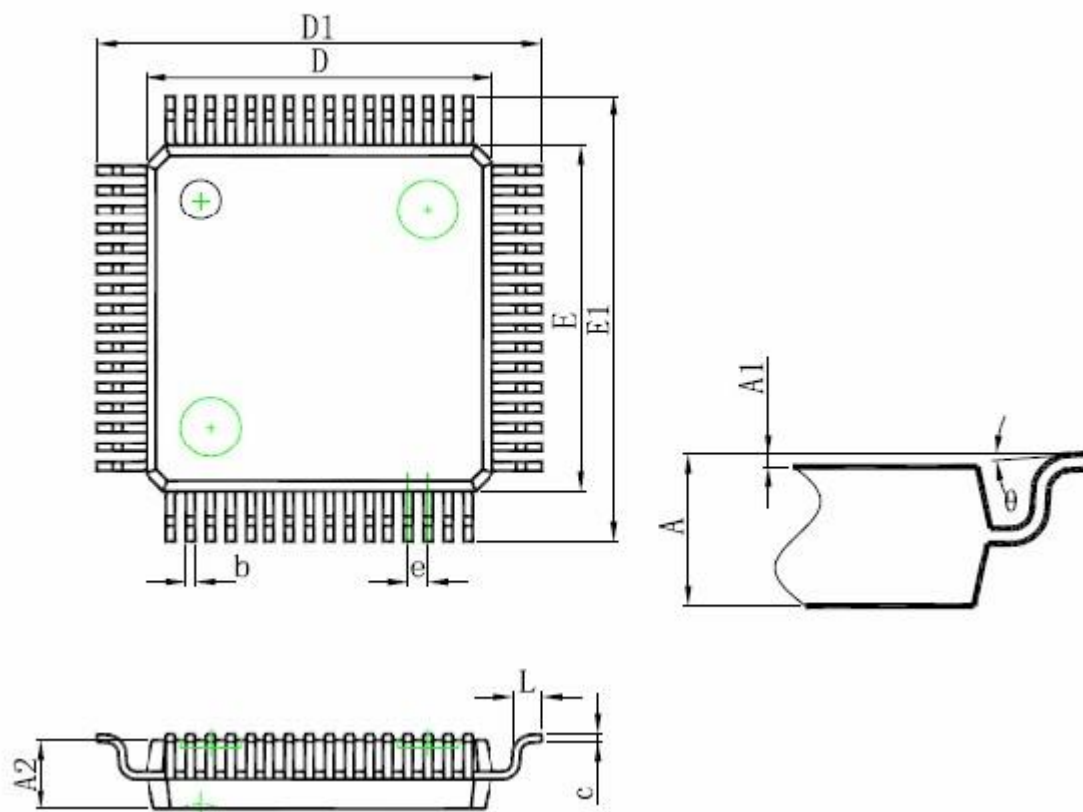


SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.60
A1	0.05	—	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.28	—	0.36
b1	0.27	0.30	0.33
c	0.13	—	0.17
c1	0.12	0.13	0.14
D	11.80	12.00	12.20
D1	9.90	10.00	10.10
E	11.80	12.00	12.20
E1	9.90	10.00	10.10
e	0.80BSC		
eB	11.25	—	11.45
L	0.45	—	0.75
L1	1.00REF		
θ	0	—	7°

17.7 LQFP 64

Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min	Max	Min	Max
A		1.600		0.063
A1	0.050	0.150	0.002	0.006
A2	1.350	1.450	0.053	0.057
b	0.170	0.240	0.007	0.009
c	0.090	0.200	0.004	0.008
D	6.900	7.100	0.272	0.280
D1	8.850	9.150	0.348	0.360
E	6.900	7.100	0.272	0.280
E1	8.850	9.150	0.348	0.360
e	0.400 (BSC)		0.016 (BSC)	
L	0.450	0.750	0.018	0.030
θ	1°	7°	1°	7°

LQFP64 (7×7) PACKAGE OUTLINE DIMENSIONS



18 汇春知识产权政策

18.1 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 160 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。

18.2 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>;