带 LED 串行点阵驱动、12BIT-ADC 的触控 MCU

GDMC181S16A/ GDMC181S20A / GDMC181S28A

版本：V1.6 日期：2018 年 06 月 28 日

第1章、GDMC181SXXA 触控 MCU 整体介绍

1.1、特性简介

- 工作电压：2.7～5.5V
- 工作温度：-40℃~+85℃
- 储存温度：-40℃~+125℃
- 16K FLASH
- 256(内部)+512(外部)字节 SRAM
- 内置 RC 振荡电路(1MHz)
- 高速 8051，基于标准 8051 指令流水线结构
- 工作频率（软件配置）：24M、12M、6M、1.5M 系统时钟，一个时钟/机器周期，速度比普通 8051 快 9~12 倍
- 最大封装支持 18 个电容按键，均可复用为 I/O
- 最大封装支持 26 个双向 I/O
- 一路 8 位 PWM 输出模块,支持二组 I/O 映射
- 8*8、7*8、7*7、6*7、6*6、5*5、4*4 LED 串行点阵驱动
- 3 个 16 位定时器，具有溢出中断,Timer2 时钟源为内部 32KHz 或外部 32768Hz 晶振可选
- 3 个外部中断(上升沿、下降沿、双沿)
- 支持空闲模式，唤醒时间 18ms ~ 2.304s
- 最多支持 4 路 12bit ADC 检测
- 支持多键低功耗模式下任意键快速唤醒(<20uA @100ms)
- IIC 硬件从机通信，支持标准模式 100KHz 或快速模式 400KHz
- UART 通信，可配置波特率(4800、9600、19200、57600、115200)，支持三组 I/O 映射
- 两级中断优先级可选
- 中断源
  - ——电容按键中断
  - ——ADC 中断
  - ——外部中断 0,1,2
  - ——Timer0，Timer1,Timer2
  - ——串口中断
  - ——IIC 通信中断
  - ——看门狗（WDT）中断
  - ——LVD 中断
  - ——LED 中断
- 支持掉电复位，掉电电压 2.5V
- 低电压检测 2.7V/3.0V/3.6V/4.2V 可选
- 看门狗定时器，溢出时间 18ms 到 2.304s
- 各按键的灵敏度可独立设置，配置灵活
- 封装型号： SOP16/SOP20/SOP28

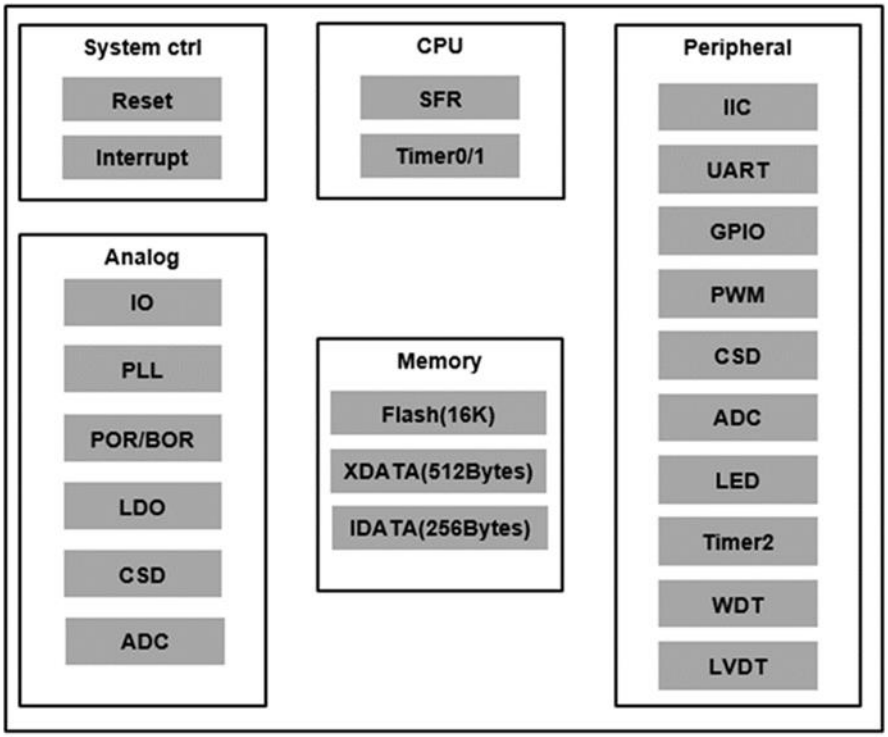
1.2、概述

GDMC181SXXA 采用高速 8051 内核，1T 指令周期，相比于标准的 8051(12T)指令周期，具有更快的运行速度，同时兼容标准 8051 指令。

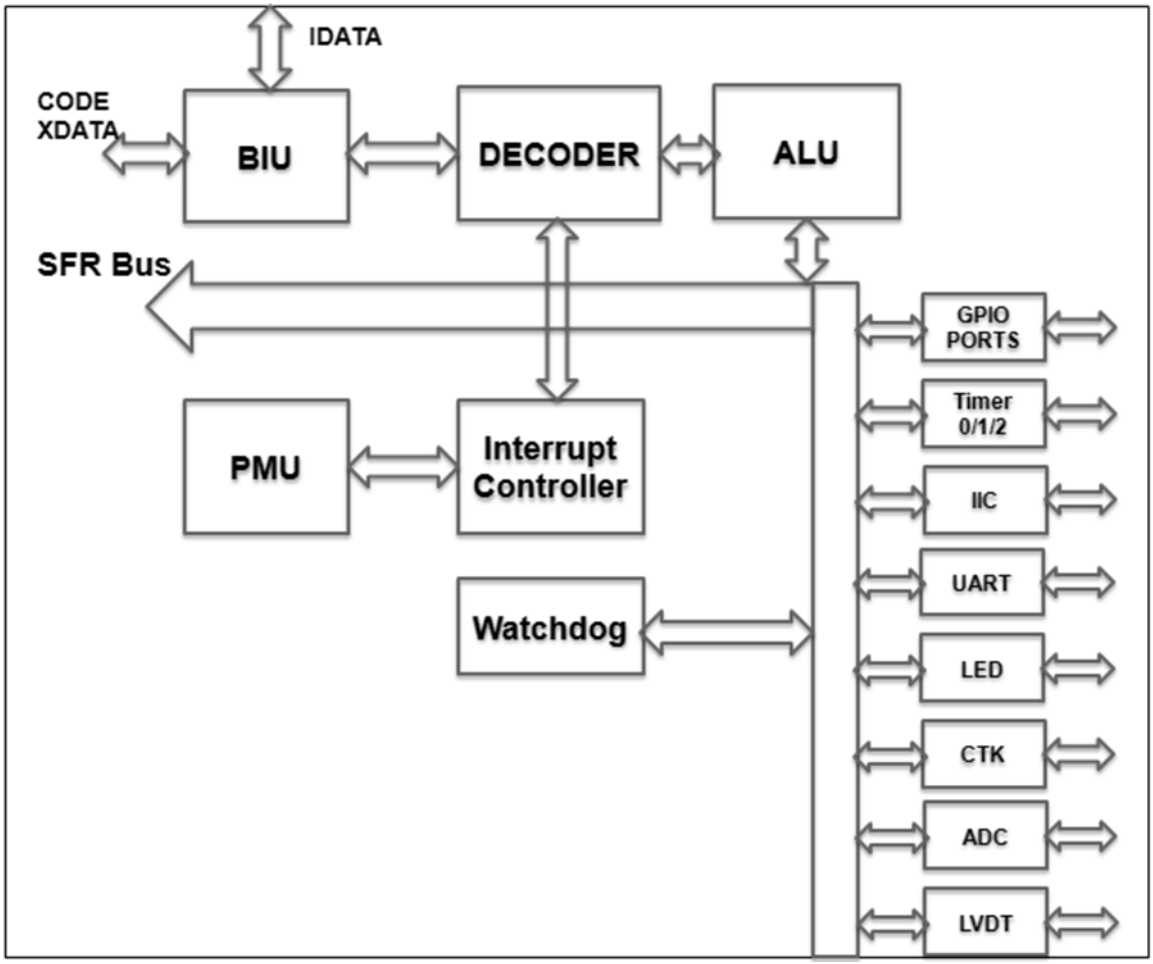
GDMC181SXXA 集成（4/10/18）电容检测通道，它可以用来检测近距离感应或者触摸。其内置 MCU，可灵活配置；通过配置可实现按键、滚轮、滑条等多种应用。（4/10/18）按键都能独立的运行，并且每个按键都能通过对相应的特殊功能寄存器来调节灵敏度。

GDMC181SXXA 包含外设有看门狗、按键检测、LED 串行点阵驱动、IIC、UART、低电压检测、1 路 8bit PWM、Timer0、Timer1、Timer2、12bit 逐次逼近 ADC。

1.3、系统结构

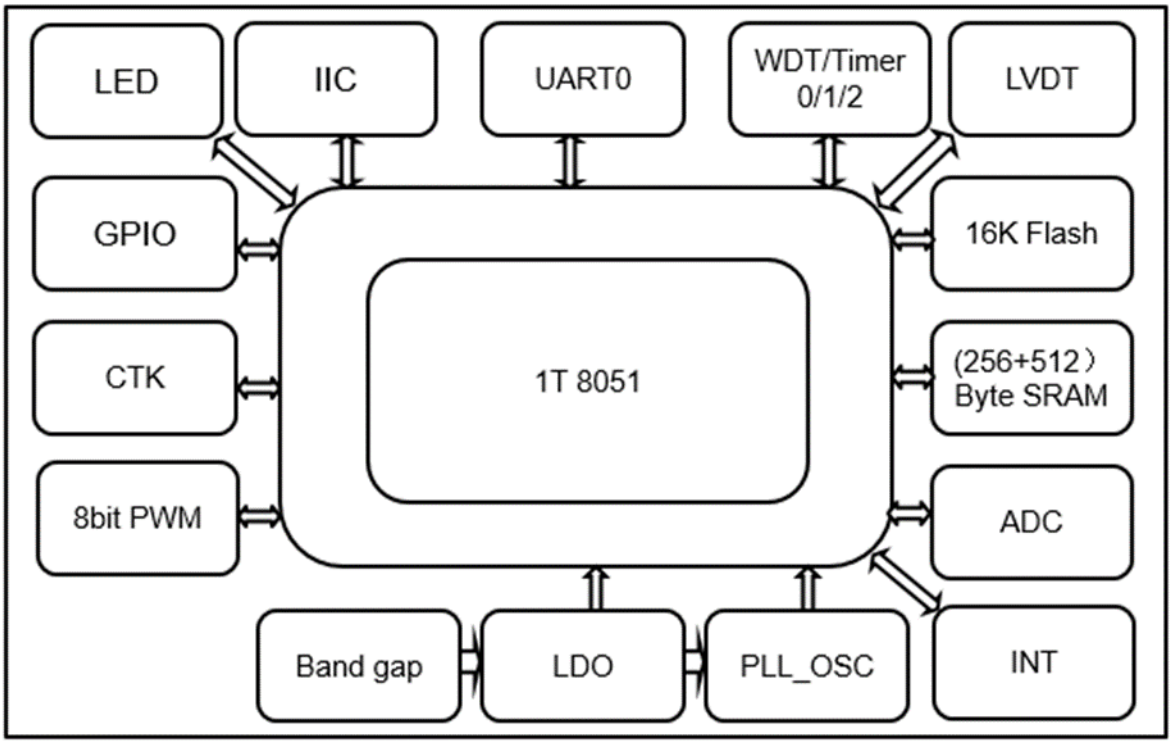


系统框图


IDATA
CODE
XDATA
BIU
DECODER
ALU
SFR Bus
GPIO
PORTS
Timer
0/1/2
IIC
UART
LED
CTK
ADC
LVDT
PMU
Interrupt
Controller
Watchdog

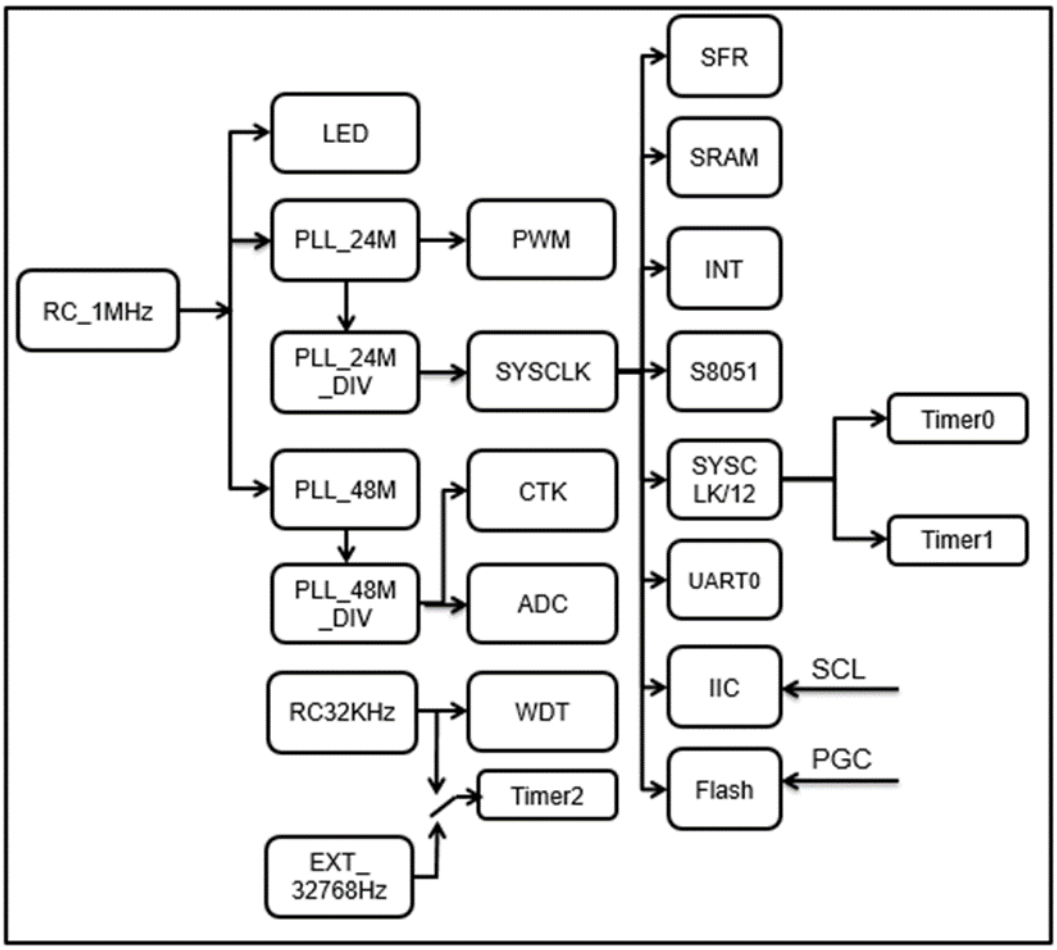
系统总线

1.4、模块方框图



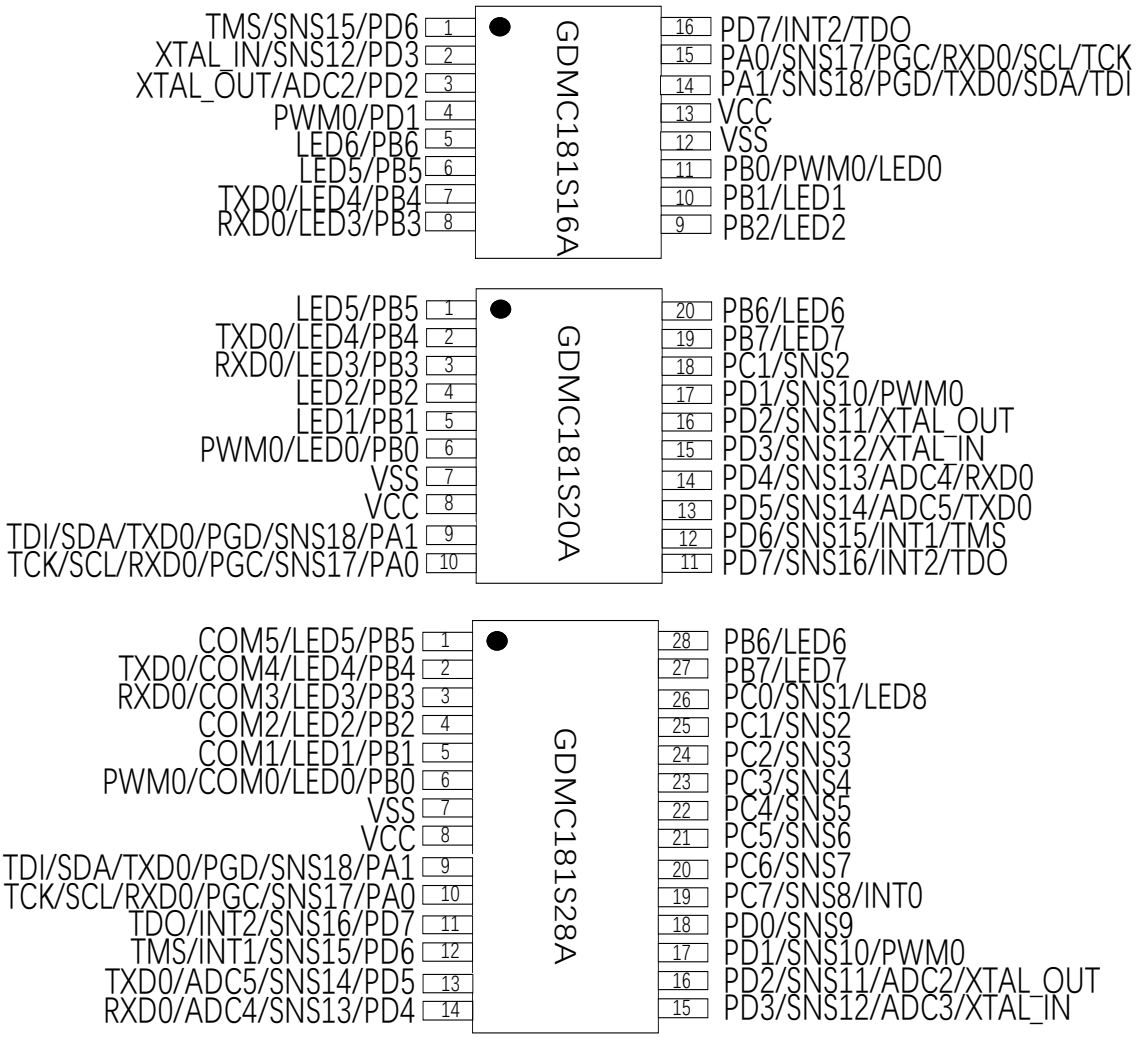
系统模块框图

1.5、时钟方框图



时钟方框图

1.6、引脚配置


GDMC181S16A
1 TMS/SNS15/PD6
2 XTAL_IN/SNS12/PD3
3 XTAL_OUT/ADC2/PD2
4 PWM0/PD1
5 LED6/PB6
6 LED5/PB5
7 TXD0/LED4/PB4
8 RXD0/LED3/PB3
16 PD7/INT2/TDO
15 PA0/SNS17/PGC/RXD0/SCL/TCK
14 PA1/SNS18/PGD/TXD0/SDA/TDI
13 VCC
12 VSS
11 PB0/PWM0/LED0
10 PB1/LED1
9 PB2/LED2
GDMC181S20A
1 LED5/PB5
2 TXD0/LED4/PB4
3 RXD0/LED3/PB3
4 LED2/PB2
5 LED1/PB1
6 PWM0/LED0/PB0
7 VSS
8 VCC
9 TDI/SDA/TXD0/PGD/SNS18/PA1
10 TCK/SCL/RXD0/PGC/SNS17/PA0
20 PB6/LED6
19 PB7/LED7
18 PC1/SNS2
17 PD1/SNS10/PWM0
16 PD2/SNS11/XTAL_OUT
15 PD3/SNS12/XTAL_IN
14 PD4/SNS13/ADC4/RXD0
13 PD5/SNS14/ADC5/TXD0
12 PD6/SNS15/INT1/TMS
11 PD7/SNS16/INT2/TDO
GDMC181S28A
1 COM5/LED5/PB5
2 TXD0/COM4/LED4/PB4
3 RXD0/COM3/LED3/PB3
4 COM2/LED2/PB2
5 COM1/LED1/PB1
6 PWM0/COM0/LED0/PB0
7 VSS
8 VCC
9 TDI/SDA/TXD0/PGD/SNS18/PA1
10 TCK/SCL/RXD0/PGC/SNS17/PA0
11 TDO/INT2/SNS16/PD7
12 TMS/INT1/SNS15/PD6
13 TXD0/ADC5/SNS14/PD5
14 RXD0/ADC4/SNS13/PD4
28 PB6/LED6
27 PB7/LED7
26 PC0/SNS1/LED8
25 PC1/SNS2
24 PC2/SNS3
23 PC3/SNS4
22 PC4/SNS5
21 PC5/SNS6
20 PC6/SNS7
19 PC7/SNS8/INT0
18 PD0/SNS9
17 PD1/SNS10/PWM0
16 PD2/SNS11/ADC2/XTAL_OUT
15 PD3/SNS12/ADC3/XTAL_IN

引脚配置

1.7、选型列表

型　号	VDD(V)	Flash	SRAM	PWM	IO	LED	定时器	AD	TK	INT	IIC/ UART	封装
GDMC181S16A	2.7~5.5	16K	768	1*8bit	14	6*7	3*16bit	1	4	1	1	SOP16
GDMC181S20A	2.7~5.5	16K	768	1*8bit	18	7*8	3*16bit	2	10	2	1	SOP20
GDMC181S28A	2.7~5.5	16K	768	1*8bit	26	8*8	3*16bit	4	18	3	1	SOP28

1.8、引脚说明

SOP16 NO.	名称	第一功能	第二功能	第三功能	第四功能	第五功能	第六功能
1	PD6/SNS15/TMS	GPIOD<6>	触摸通道	JTAG-模式选择	-	-	
2	PD3/SNS12/XTAL_IN	GPIOD<3>	触摸通道	外部晶振端口	-	-	-
3	PD2/ADC2/XTAL_OUT	GPIOD<2>	ADC 通道	外部晶振端口	-	-	-
4	PD1/PWM0	GPIOD<1>	PWM0	-	-	-	-
5	PB6/LED6	GPIOB<6>	LED 驱动口	-	-	-	-
6	PB5/LED5	GPIOB<5>	LED 驱动口	-	-	-	-
7	PB4/LED4/TXD0	GPIOB<4>	LED 驱动口	TXD0	-	-	-
8	PB3/ LED3/RXD0	GPIOB<3>	LED 驱动口	RXD0	-	-	-
9	PB2/LED2	GPIOB<2>	LED 驱动口	-	-	-	-
10	PB1/LED1	GPIOB<1>	LED 驱动口	-	-	-	-
11	PB0/LED0/PWM0	GPIOB<0>	LED 驱动口	PWM0	-	-	-
12	VSS	地	-	-	-	-	-
13	VCC	电源	-	-	-	-	-
14	PA1/SNS18/TXD0/PGD/SDA/TDI	GPIOA<1>	触摸通道	烧录数据端口(PGD)	UART0-TXD0	IIC_SDA	JTAG-数据输入
15	PA0/SNS17/RXD0/PGC/SCL/TCK	GPIOA<0>	触摸通道	烧录数据端口(PGC)	UART0-RXD0	IIC_SCL	JTAG-时钟
16	PD7/INT2/TDO	GPIOD<7>	INT2	JTAG-数据输出	-	-	-

SOP20 PIN NO.	名称	第一功能	第二功能	第三功能	第四功能	第五功能	第六功能
1	PB5/LED5	GPIOB<5>	LED 驱动口	-	-	-	-
2	PB4/LED4/TXD0	GPIOB<4>	LED 驱动口	TXD0	-	-	-
3	PB3/LED3/RXD0	GPIOB<3>	LED 驱动口	RXD0	-	-	-
4	PB2/LED2	GPIOB<2>	LED 驱动口	-	-	-	-
5	PB1/LED1	GPIOB<1>	LED 驱动口	-	-	-	-
6	PB0/LED0/PWM0	GPIOB<0>	LED 驱动口	PWM0	-	-	-
7	VSS	地	-	-	-	-	-
8	VCC	电源	-	-	-	-	-
9	PA1/SNS18/TXD0/PGD/SDA/TDI	GPIOA<1>	触摸通道	烧录数据端口(PGD)	UART0-TXD0	IIC_SDA	JTAG-数据输入
10	PA0/SNS17/RXD0/PGC/SCL/TCK	GPIOA<0>	触摸通道	烧录数据端口(PGC)	UART0-RXD0	IIC_SCL	JTAG-时钟
11	PD7/SNS16/INT2/TDO	GPIOD<7>	触摸通道	INT2	JTAG-数据输出		
12	PD6/SNS15/INT1/TMS	GPIOD<6>	触摸通道	-	INT1	JTAG-模式选择	
13	PD5/SNS14/ADC5/TXD0	GPIOD<5>	触摸通道	ADC 通道	UART0-TXD0	-	-
14	PD4/SNS13/ADC4/RXD0	GPIOD<4>	触摸通道	ADC 通道	UART0-RXD0	-	-
15	PD3/SNS12/XTAL_INT	GPIOD<3>	触摸通道	-	外部晶振端口	-	-
16	PD2/SNS11/XTAL_OUT	GPIOD<2>	触摸通道	-	外部晶振端口	-	-
17	PD1/SNS10/PWM0	GPIOD<1>	触摸通道	PWM0	-	-	-
18	PC1/SNS2	GPIOC<1>	触摸通道	-	-	-	-
19	PB7/LED7	GPIOB<7>	-	LED 驱动口	-	-	-
20	PB6/LED6	GPIOB<6>	LED 驱动口	-	-	-	-

SOP28 PIN NO.	名称	第一功能	第二功能	第三功能	第四功能	第五功能	第六功能
1	PB5/LED5/COM5	GPIOB<5>	LED 驱动口	LCD-COM	-	-	-
2	PB4/LED4/COM4/TXD0	GPIOB<4>	LED 驱动口	LCD-COM	TXD0	-	-
3	PB3/LED4/COM3/RXD0	GPIOB<3>	LED 驱动口	LCD-COM	RXD0	-	-
4	PB2/LED2/COM2	GPIOB<2>	LED 驱动口	LCD-COM	-	-	-
5	PB1/LED1/COM1	GPIOB<1>	LED 驱动口	LCD-COM	-	-	-
6	PB0/LED0/COM0/PWM0	GPIOB<0>	LED 驱动口	LCD-COM	PWM0	-	-
7	VSS	地	-	-	-	-	-
8	VCC	电源	-	-	-	-	-
9	PA1/SNS18/TXD0/PGD/SDA/TDI	GPIOA<1>	触摸通道	烧录数据端口(PGD)	UART0-TXD0	IIC_SDA	JTAG-数据输入
10	PA0/SNS17/RXD0/PGC/SCL/TCK	GPIOA<0>	触摸通道	烧录数据端口(PGC)	UART0-RXD0	IIC_SCL	JTAG-时钟
11	PD7/SNS16/INT2/TDO	GPIOD<7>	触摸通道	INT2	JTAG-数据输出		
12	PD6/SNS15/INT1/TMS	GPIOD<6>	触摸通道	-	INT1	JTAG-模式选择	
13	PD5/SNS14/ADC5/TXD0	GPIOD<5>	触摸通道	ADC 通道	UART0-TXD0	-	-
14	PD4/SNS13/ADC4/RXD0	GPIOD<4>	触摸通道	ADC 通道	UART0-RXD0	-	-
15	PD3/SNS12/ADC3/XTAL_IN	GPIOD<3>	触摸通道	ADC 通道	外部晶振端口	-	-
16	PD2/SNS11/ADC2/XTAL_OUT	GPIOD<2>	触摸通道	ADC 通道	外部晶振端口	-	-
17	PD1/SNS10/PWM0	GPIOD<1>	触摸通道	-PWM0	-	-	-
18	PD0/SNS9	GPIOD<0>	触摸通道	-	-	-	-
19	PC7/SNS8/INT0	GPIOC<7>	触摸通道	INT0	-	-	-
20	PC6/SNS7	GPIOC<6>	触摸通道	-	-	-	-
21	PC5/SNS6	GPIOC<5>	触摸通道	-	-	-	-
22	PC4/SNS5	GPIOC<4>	触摸通道	-	-	-	-
23	PC3/SNS4	GPIOC<3>	触摸通道	-	-	-	-
24	PC2/SNS3	GPIOC<2>	触摸通道	-	-	-	-
25	PC1/SNS2	GPIOC<1>	触摸通道	-	-	-	-
26	PC0/SNS1/LED8	GPIOC<0>	触摸通道	LED 驱动口	-	-	-
27	PB7/LED7	GPIOB<7>	-	LED 驱动口	-	-	-
28	PB6/LED6	GPIOB<6>	LED 驱动口	-	-	-	-

注：“*”表示工作时必须使用的管脚，“-”表示工作时未使用的管脚,“#”表示特殊芯片特殊功能脚

第 2 章、电气特性

2.1 、AC 特性

下表列出了 OSC 时钟特性参数：

参数	符号	条件	OSC时钟	单位
基频	RC1M	正常工作27℃	1M±1%	Hz
		环境温度-40℃~85℃	1M±3%	
WDT时钟	RC32K	环境温度-40℃~125℃	32K±30%	Hz

2.2 、DC 特性

下表列出了在电压范围：2.7V～5.5V，温度范围：-40℃<TA<85℃下各参数的最大值与最小值,典型值为在27℃条件下的测量值。

参数	符号	条件	最小值	典型值	最大值	单位
工作电压	VCC		2.7	-	5.5	V
电流	Iactive	@5V,系统时钟24M	-	7.6	9	mA
		@5V,系统时钟12M	-	4.7	6	mA
		@5V,系统时钟6M	-	3.2	5	mA
		@5V,系统时钟1.5M	-	2	4	mA
		@3.3V,系统时钟24M	-	7.4	9	mA
		@3.3V,系统时钟12M	-	4.4	6	mA
		@3.3V,系统时钟6M	-	3	5	mA
		@3.3V,系统时钟1.5M	-	1.8	4	mA
	Iidle	@5V,WDT_CTRL=7，唤醒，1%工作时间，IO输出低，关闭其它所有功能	-	12	20	μA
		@5V,Timer2 外部晶振2S唤醒，1%工作时间，IO输出低，关闭其它所有功能	-	12	20	μA
		@3.3V,WDT_CTRL=7，唤醒，1%工作时间，IO输出低，关闭其它所有功能	-	11	18	μA
		@3.3V,Timer2 外部晶振2S唤醒，1%工作时间，IO输出低，关闭其它所有功能	-	11	18	μA

		@5V,CTK并联中断唤醒，1%工作时间，IO输出低，关闭其它所有功能	-	15	25	μA
		@3.3V,CTK并联中断唤醒，1%工作时间，IO输出低，关闭其它所有功能	-	12	23	μA
	Sleep	@5V PCON = 0x01,BOR关闭，IO输出低，关闭其它所有功能	-	5	8	μA
		@3.3V PCON = 0x01,BOR关闭，IO输出低，关闭其它所有功能	-	4.7	7	μA
输入低电压	V_{IL}	VCC=3.3~5.5V	-	-	0.3*VCC	V
输入高电压	V_{IH}	VCC=3.3~5.5V	0.7*VCC	-	-	V
INT0/1/2输入低电压	V_{INTL}	VCC=3.3~5.5V	-	-	0.3*VCC	V
INT0/1/2输入高电压	V_{INTH}	VCC=3.3~5.5V	0.7*VCC	-	-	V
输出低电压	V_{OL}	I_{OL}=4mA@VCC=3.3V I_{OL}=10mA@VCC=5V	-	-	0.1*VCC	V
输出高电压	V_{OH}	I_{OH}=4mA@VCC=3.3V I_{OH}=10mA@VCC=5V	0.9VCC	-	-	V
IO灌电流	I_{OL}	V_{OL} =0.1VCC	10	16	20	mA
IO源电流	I_{OH}	V_{OH} =0.9VCC	10	16	20	mA
输入漏电流	I_{IH} I_{IL}	VCC=5V	-	1	5	μA
IO内部上拉	Pull_up Res	VCC=5V	-	4.7	-	K

2.3、极限参数

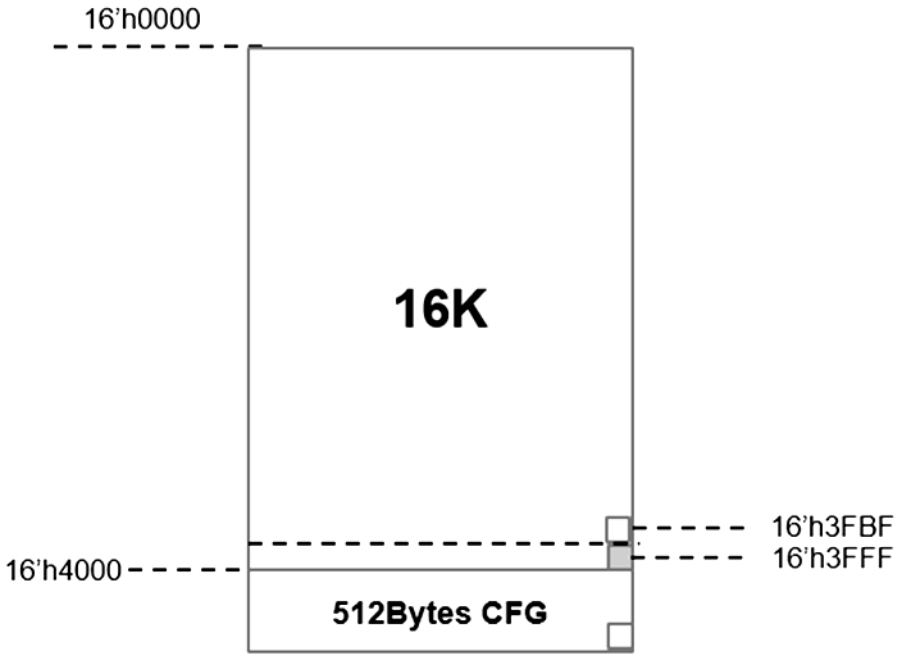
参数	符号	最小值	典型值	最大值	单位
非工作状态储存温度	T_{stg}	-40	-	125	℃
工作温度	T_{otg}	-40	-	85	℃
I/O输入电压	Vin	VSS-0.5	-	VCC+0.5	V
I/O锁存电流	LU	200	-	-	mA
端口静电放电电压	ESD(HBM)	4000	-	-	V
工作时供电电压	VCC	VSS+2.7	-	VSS+5.5	V

注：超过极限参数所规定的范围将对芯片造成损害，无法预期芯片在上述标示范围外的工作状态，而且若长期在标示范围外的条件下工作，可能影响芯片的可靠性。

第 3 章、存储器和特殊功能寄存器(SFRs)

3.1、Flash 程序存储器

Flash 结构示意图：



Falsh 存储结构

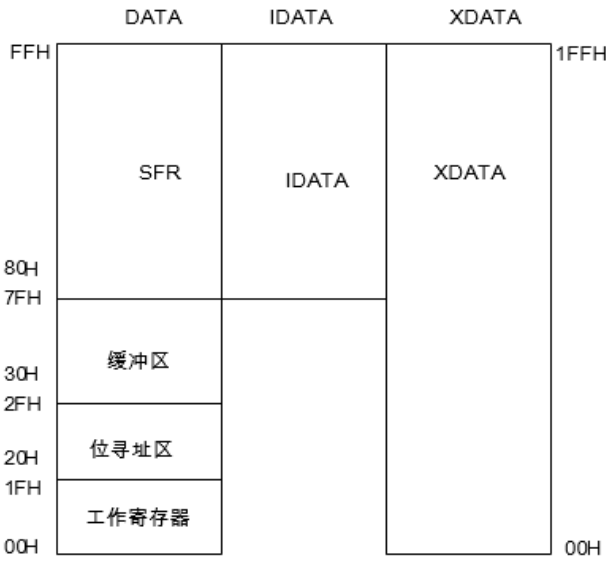
- Flash 内存包括两个 block：main_block 和 information_block；
  main_block(0000H~3FBFH)大小为 16k Bytes-64Bytes，
  information_block(4000H~41FFH)大小为 512 Bytes；
- 通过专用烧录器能进行正常的读写操作(擦写次数 1 万次)，应用代码不可操作。
- 支持 main_block 读保护和 information_block 写保护。
- 烧录端口附 8 位 CRC 校验码。
- 常温下数据保持年限为 10 年。

3.2、RAM

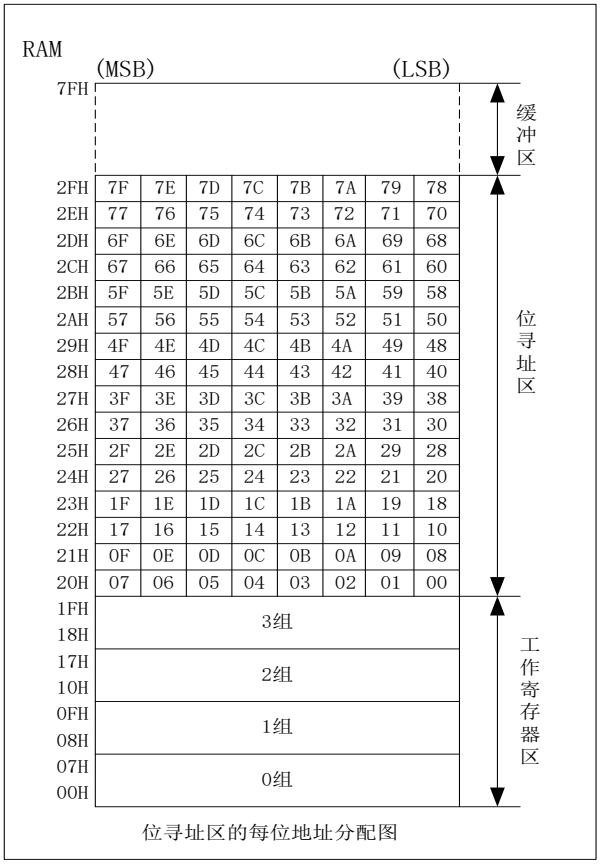
根据数据的取值方式，可将 RAM 分为三个模块：

- data 区：共有 256 字节，地址为 00H~FFH，其中包括工作寄存器组、位寻址区、缓冲以及 SFR，其中缓冲区包含了堆栈区。通过立即寻址方式来读取与写数据。
- idata 区：共有 128 字节，地址为 80H~FFH，该区域用户可以完全使用。通过工作寄存器间接寻址方式来读取与写数据。
- xdata 区：共有 512 字节，地址为 0000H~01FFH，该区域用户可以完全使用。通过数据指针或者工作寄存器寻址方式来读取与写数据。

在编写程序时注意预留堆栈空间，避免堆栈溢出导致程序跑飞。在使用 C 语言编程时，堆栈首地址由程序自动分配，但是一定存放在 data 或者 idata 里。Keil 中可在 STARTUP.A51 中设置堆栈的首地址。



数据存储器RAM地址分配



位寻址区的每位地址分配图

下表中列出了 RAM 中三个模块的取值方式：

DATA	MOV	A,direct
	MOV	direct,A
	MOV	direct,#data
	MOV	direct1,direct2
	MOV	Rn,direct
	MOV	direct,Rn
IDATA	MOV	A,@Ri

	MOV @Ri ,A
	MOV direct,@Ri
	MOV @Ri,derect
	MOV @Ri,#data

XDATA	MOVX @DPTR,A MOVX A,@DPTR

上表中，n 取值 0~7，i 取值 0~1。

3.3、SFR 总表

地址	名称	写/读	默认/复位	功能说明
80H	PWM_DIV	W/R	x000_0000	PWM 控制寄存器
81H	SP	W/R	0000_0111	堆栈指针寄存器
82H	DPL	W/R	0000_0000	数据指针寄存器 0 低 8 位
83H	DPH	W/R	0000_0000	数据指针寄存器 0 高 8 位
87H	PCON	(arrr_rraa)	0011_0000	低功耗模式选择寄存器
88H	TCON	(rara_rara)	0000_0101	定时器控制寄存器
89H	TMOD	(-aaa_-aaa)	0000_0000	定时器模式寄存器
8AH	TL0	W/R	0000_0000	定时器 0 计时器低 8 位
8BH	TL1	W/R	0000_0000	定时器 1 计时器低 8 位
8CH	TH0	W/R	0000_0000	定时器 0 计时器高 8 位
8DH	TH1	W/R	0000_0000	定时器 1 计时器高 8 位
8EH	PWM_DUTY	W/R	0000_0000	PWM 占空比配置寄存器
90H	PWM_CS	(----_---a)	xxxx_xxx0	PWM 输出使能寄存器
91H	WDT_CTRL	(----_-aaa)	0000_0000	WDT 时间配置寄存器
92H	WDT_EN	W/R	0000_0000	WDT 使能寄存器
93H	TIMER2_CFG	(----_-aaa)	0000_0000	TIMER2 控制寄存器
94H	TIMER2_SET_H	W/R	0000_0000	定时器 2 计时器高 8 位
95H	TIMER2_SET_L	W/R	0000_0000	定时器 2 计时器低 8 位
96H	REG_ADDR	(----_aaaa)	0000_0000	二级总线地址寄存器
97H	REG_DATA	W/R	0000_1111	二级总线数据读写寄存器
9AH	TRISA	(----_aaaa)	0000_1111	PA 口方向寄存器
9BH	TRISB	W/R	1111_1111	PB 口方向寄存器
9CH	TRISC	W/R	1111_1111	PC 口方向寄存器
9DH	TRISD	W/R	1111_1111	PD 口方向寄存器
9EH	DATAA	(----_aaaa)	0000_1111	PA 口数据寄存器
9FH	DATAB	W/R	1111_1111	PB 口数据寄存器
A1H	DATAC	W/R	1111_1111	PC 口数据寄存器
A2H	DATAD	W/R	1111_1111	PD 口数据寄存器
A3H	SNS_IO_SELL	W/R	0000_0000	SENSOR 使能寄存器
A6H	SNS_IO_SELH	W/R	0000_0000	SENSOR 使能寄存器
A7H	PU_PA	(----_aaaa)	0000_0011	PA 口上拉电阻使能寄存器
A8H	IENO	W/R	0000_0000	中断使能寄存器
AAH	PU_PC	W/R	0000_0000	PC 口上拉电阻使能寄存器
ABH	COM_EN	(---a_aaaa)	xxx0_0000	COM 口选择使能寄存器
ACH	ADC_IO_SEL	(-aaa_aaaa)	x000_0000	ADC 功能选择寄存器
ADH	PERIPH_IO_SEL	(-aaa_aaaa)	x000_0000	IO 口复用功能选择寄存器

AEH	EXT_INT_CON	(--aa_aaaa)	xx01_0101	外部中断极性选择寄存器
AFH	INT_BOPO_STAT	(----_--aa)	xxxx_xx00	LVDT 中断使能控制寄存器
B0H	ODRAIN_EN	(----_aaaa)	xxxx_0000	PA 口开漏输出使能寄存器
B1H	SEG_SEL_1	W/R	0000_0000	LED 矩阵扫描配置寄存器 1
B2H	SEG_SEL_2	W/R	0000_0000	LED 矩阵扫描配置寄存器 2
B3H	SEG_SEL_3	W/R	0000_0000	LED 矩阵扫描配置寄存器 3
B4H	SEG_SEL_4	W/R	0000_0000	LED 矩阵扫描配置寄存器 4
B5H	SEG_SEL_5	W/R	0000_0000	LED 矩阵扫描配置寄存器 5
B6H	SEG_SEL_6	W/R	0000_0000	LED 矩阵扫描配置寄存器 6
B8H	IPL0	(-aaa_aaaa)	0000_0000	中断优先级寄存器 0
BAH	SEG_SEL_7	W/R	0000_0011	LED 矩阵扫描配置寄存器 7
BBH	SEG_SEL_8	W/R	0000_0111	LED 矩阵扫描配置寄存器 8
BCH	LED1_WIDTH	W/R	0000_0000	第一段 LED 周期配置寄存器
BDH	LED2_WIDTH	W/R	0000_0000	第二段 LED 周期配置寄存器
BEH	LED_SEG_POINT	(--aa_aaaa)	xx00_0000	两段 LED 分界坐标寄存器
BFH	LED_DRIVE	(---a_aaaa)	xxx0_0000	LED 口驱动能力配置寄存器
C0H	LED_MODE	(---a_aaaa)	xxx0_0000	LED 扫描模式及控制寄存器
C1H	ADC_SPT	(---a_aaaa)	xxx0_0010	ADC 采样时间配置寄存器
C2H	ADCCALC	(----_aaaa)	xxxx_0010	ADC 校准寄存器
C3H	ADC_SCAN_CFG	(----_aaaa)	xxxx_0000	ADC 通道选择及使能寄存器
C4H	ADCCKC	W/R	0000_0000	ADC 转换参数设置寄存器
C5H	ADC_RDATAH	W/R	0000_0000	ADC 扫描结果高 8 位寄存器
C6H	ADC_RDATAL	W/R	0000_0000	ADC 扫描结果低 8 位寄存器
C7H	ADC_I_SEL	(----_--aa)	xxxx_xx00	ADC 偏置电流设置寄存器
CEH	SNS_SCAN_CFG1	W/R	0111_0001	CTK 扫描参数设置寄存器 1
CFH	SNS_SCAN_CFG2	W/R	0000_0000	CTK 扫描参数设置寄存器 2
D0H	PSW	R	0000_0000	程序状态字寄存器
D1H	CSD_START	(----_---a)	xxxx_xxx0	CTK 扫描开启使能寄存器
D2H	RAWDATAL	W/R	0000_0000	CTK 扫描结果低 8 位寄存器
D3H	RAWDATAH	W/R	0000_0000	CTK 扫描结果高 8 位寄存器
D4H	PULL_I_SELA_H	(----_--aa)	xxxx_xx00	CTK 扫描电流源步进配置寄存器
D5H	PULL_I_SELA_L	W/R	0000_0000	CTK 扫描电流源大小配置寄存器
D6H	SNS_ANA_CFG	(-aaa_aaaa)	x101_0010	CTK 扫描模拟参数设置寄存器
D7H	RST_STAT	(-aaa_aaaa)	RST_STAT	复位状态寄存器
D8H	SOFT_RST	W/R	0000_0000	软件复位寄存器
D9H	SYS_CLK_CFG	(----_-aaa)	xxxx_x000	系统时钟分频及使能寄存器
DAH	SEL_LVDT_VTH	(----_--aa	xxxx_xx00	LVDT 阈值电压设置寄存器
DBH	PD_ANA	(-aaa_aaaa)	x111_1011	复用功能使能寄存器
DCH	UART_BDL	W/R	0000_0000	串口通信波特率设置低 8 位寄存器
DDH	UART_CON1	(-aaa_aaaa)	x000_0000	串口通信参数设置寄存器 1

DEH	UART_CON2	(----_aaaa)	xxxx_1100	串口通信参数设置寄存器 2
DFH	UART_STATE	(-aaa_aaaa)	x000_0000	串口通信状态标志寄存器
E0H	ACC	W/R	0000_0000	累加器
E1H	IRCON2	W/R	0000_0000	中断标志寄存器 2
E2H	UART_BUF	W/R	1111_1111	串口通信数据寄存器
E3H	IICADD	(aaaa_aaar)	0000_0000	IIC 地址寄存器
E4H	IICBUF	W/R	0000_0000	IIC 读写缓存寄存器
E5H	IICCON	(--aa_aaaa)	xx01_0000	IIC 控制寄存器
E6H	IEN1	W/R	0000_0000	中断使能寄存器 1
E7H	IEN2	W/R	0000_0000	中断使能寄存器 2
E8H	IICSTAT	(rrrr_rraa)	0100_0100	IIC 通信状态寄存器
E9H	IICBUFFER	W/R	0000_0000	IIC 数据发送缓存寄存器
EAH	PU_PB	W/R	0000_0000	PB 口上拉电阻使能寄存器
EBH	PU_PD	W/R	0000_0000	PD 口上拉电阻使能寄存器
ECH	SNS_IO_SELO	(----_-aaa)	xxxx_x000	SENSOR 使能寄存器
F0H	B	W/R	0000_0000	B 寄存器
F1H	IRCON1	W/R	0000_0000	中断标志寄存器 1
F4H	IPL2	W/R	0000_0000	中断优先级寄存器 2
F6H	IPL1	W/R	0000_0000	中断优先级寄存器 1

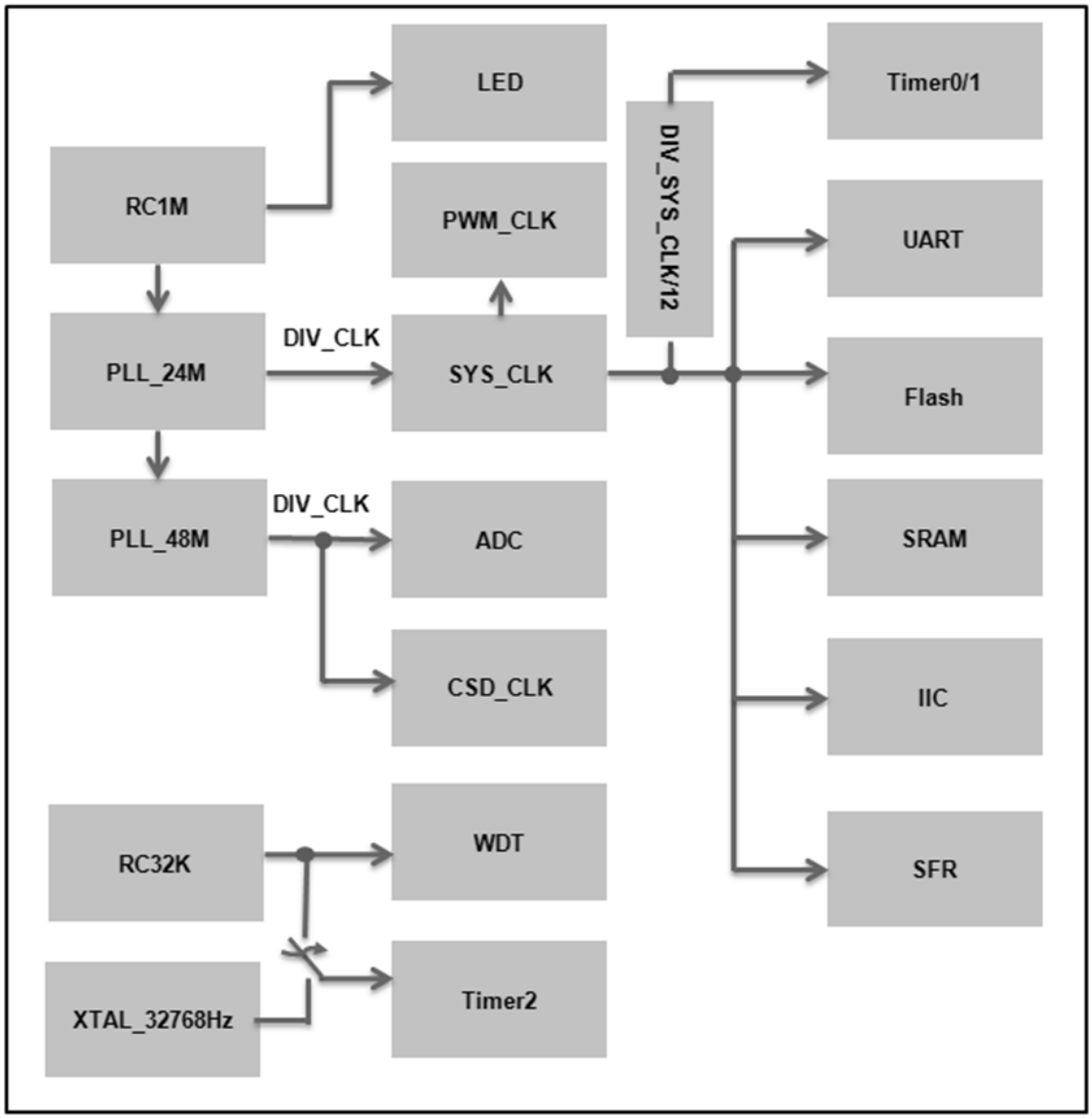
第 4 章、时钟、复位、WDT、省电模式

4.1、系统时钟和振荡器

支持内部 RC 振荡器:

■内建 1MHz RC 振荡器

■内建 RC32K 振荡器



时钟分布图

4.1.1、时钟定义

GDMC181SXXA 内部时钟定义如下：

RC1MHz：内置 RC 振荡器，频率为 1MHz,该时钟作为 LED Driver 时钟，及 PLL 时钟。

RC32KHz：内置独立 RC 振荡 32KHz 时钟，该时钟作为看门狗时钟或 Timer2 时钟。

PLL_24MHz：锁相环倍频后的 24MHZ 时钟。

SYS_CLK:这个时钟为 CPU 指令周期的时钟，由 PLL_24MHz 分频得到(24MHz、12MHz、8MHz、1.5MHz)作为系统时钟、串口时钟源、PWM、SFR、SRAM、Flash、INT 的时钟源。

SYS_CLK/12:系统时钟（SYSCLK）分频后的时钟,作为 timer0 和 timer1 的时钟源。

PLL_48MHz:由锁相环倍频后的时钟 48MHz,该时钟分频后作为 ADC 和 CSD 的时钟源。

SCL:IIC 主机时钟，由 IIC Master 主机发出，作为 IIC 通信时钟。

PGC:编程时钟，编程烧录程序时的下载时钟。

4.1.2、系统时钟设置

通过设置不同的分频系数可以设置芯片的 SYSCLK 时钟。

系统时钟寄存器 SYS_CLK_CFG—0xD9h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SYS_CLK_CFG	-					PLL_CLK_SEL[1:0]		PD_SYS_CLK
POR	-					0	0	0

PLL_CLK_SEL：系统时钟分频选择寄存器；

00：24MHz；

01：12MHz；

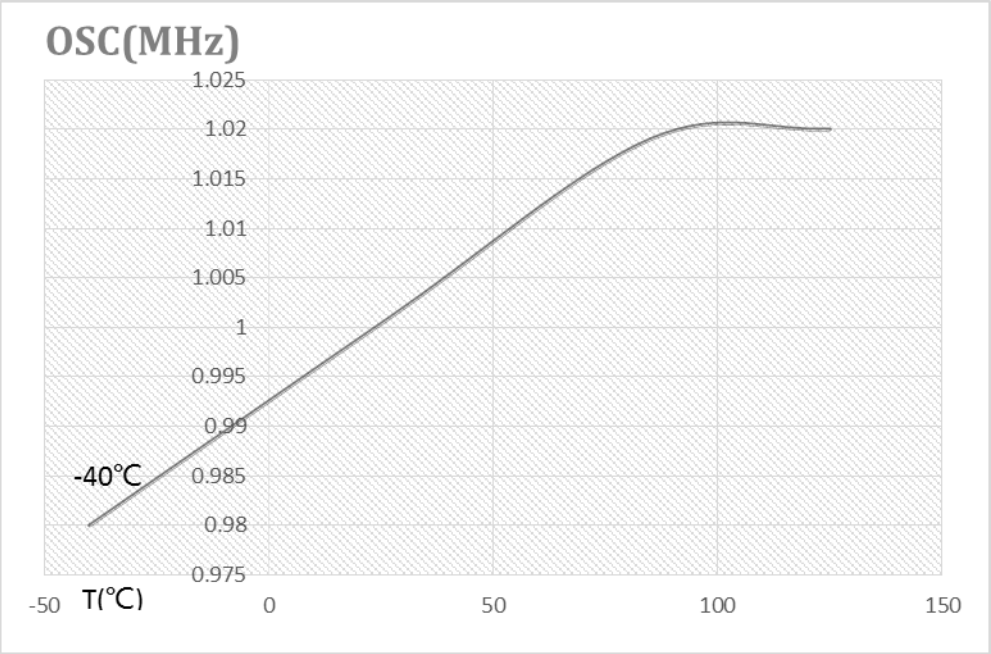
10：6MHz；

11：1.5MHz；

该寄存器在程序所有功能初始化之前配置。

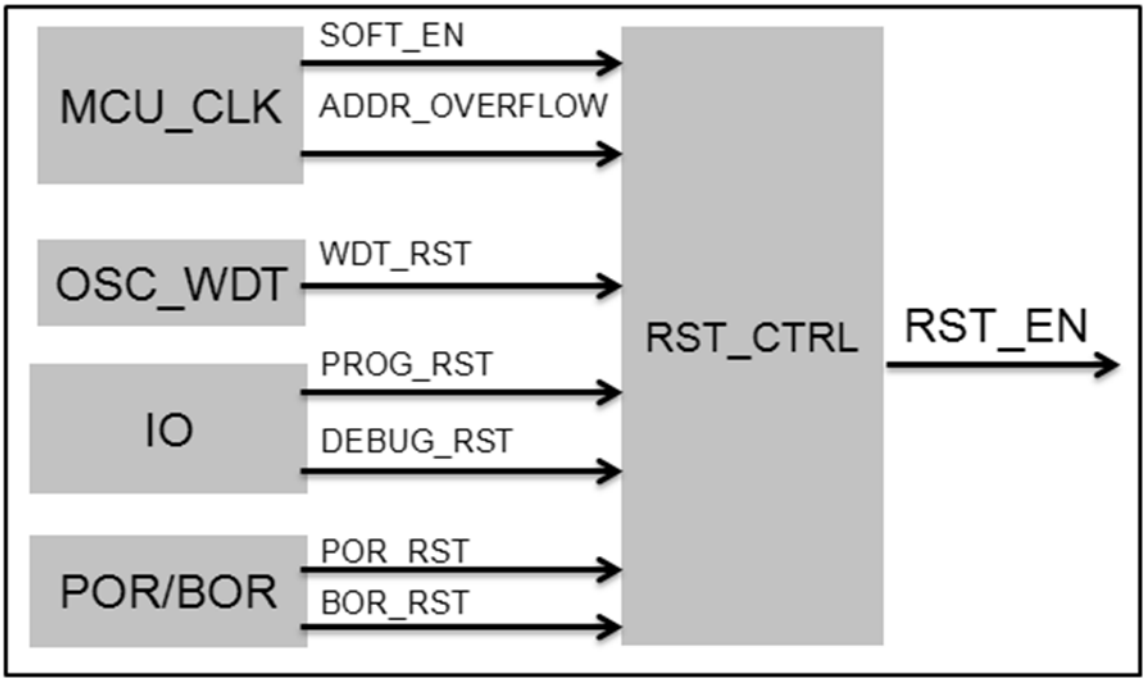
PD_SYS_CLK：系统时钟开启寄存器；1：关闭系统时钟；0：开启系统时钟。

4.1.3、OSC_1MHz 温度曲线



4.2 、复位

GDMC181SXXA 中有 7 种复位模式：看门狗定时器溢出复位（WDT_RST）、上电复位（POR）、掉电复位（BOR）、PC 指针溢出复位（ADDR_OVERFLOW）、flash 编程复位（POR_EN）、软件复位（SOFT_RST）、调试复位（DEBUG_RST）。只要其中任意一种复位发生，系统的全局复位信号就会让整个芯片复位。可由复位标志寄存器来确定芯片进行了何种复位，复位标志位需软件清零。



复位结构框图

RST_STAT 寄存器—0xD7h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RST_STAT	-	DEBUG_RST	SOFT_RST	POR_EN	ADDROF_F	BO_F	PO_F	WDTRST_F
POR	-	0	0	0	0	0	0	0

WDTRST_F：看门狗复位的标志位，在看门狗复位发生时，此位的值是 1。

BO_F：掉电复位的标志位，在掉电复位发生时，此位的值是 1。

PO_F：上电复位的标志位，在上电复位发生时，此位的值是 1。

ADDROF_F：地址溢出复位标志位，在地址溢出复位发生时，此位的值是 1。

POR_EN：Flash 编程复位标志位

SOFT_RST：软件复位标志位，当软复位标志位，当寄存器 SOFT_RST=0x55 时，复位完成后，此位的值是 1。

DEBUG_RST：进入 JTAG 调试模式时，芯片发生调试复位，此位的值为 1。

4.2.1、上/掉电复位

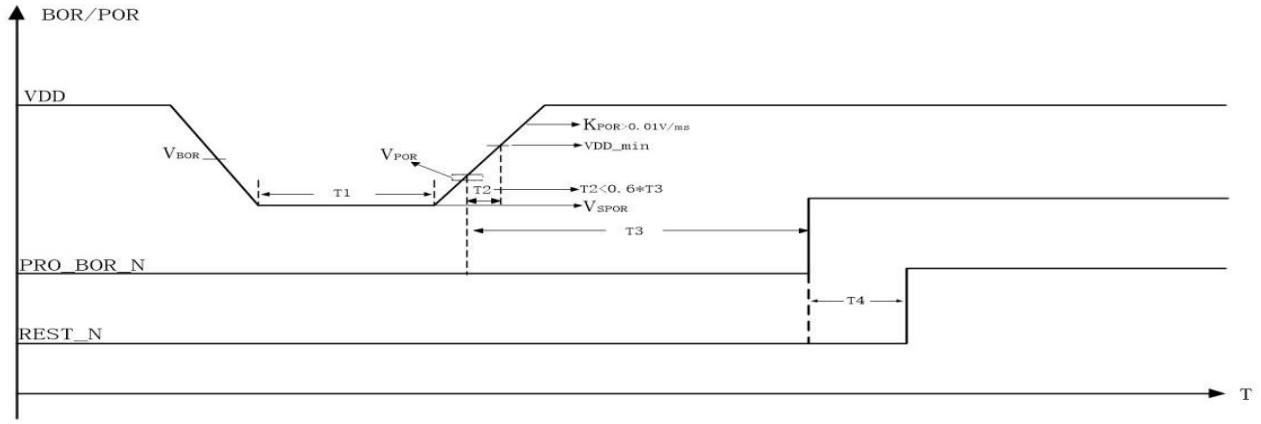
系统发生上/掉电复位后，系统产生 POR_BOR_N 信号（低电平）并持续 93ms。POR_BOR_N 为低时整个芯片处于复位状态，变高后全局复位信号继续有效 20ms.

PD_ANA 寄存器 -- 0xDBh

位	功能	R/W	复位值
PD_ANA.4	PD_BOR—BOR功能使能控制，0：开启，1：关闭，默认为1。	R/W	1

PD_ANA.4：0：BOR 功能开启(2.5V)；1：BOR 功能关闭；默认为 1；

上/掉电时序：

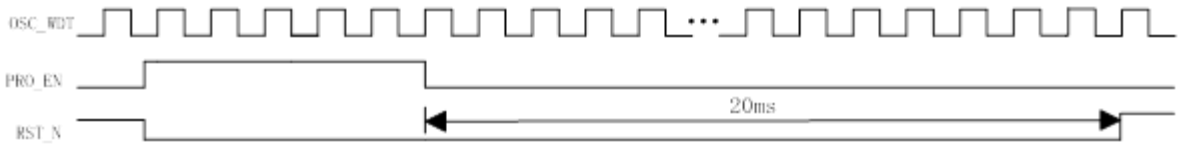


上/掉电复位参数：

符号	参数	测试条件(VDD)	最小	典型	最大	单位
V_{SPOR}	上电复位起始电压	—	—	—	300	mV
K_{PRO}	上电复位电压速率	—	0.01	—	—	V/ms
V_{POR}	上电复位电压	—	1.1	1.5	2.2	V
V_{BOR}	掉电复位电压(±10%),迟滞 0.2V	—	—	V_{BOR}	-	V
VDD_min	最小工作电压	—	2.7	-	-	V
T1	VDD 保持 V_{SPOR} 时间	—	0.1	—	—	ms
T2	V_{POR} 到 VDD_min 时间	—	-	-	0.6*T3	ms
T3	复位 POR_BOR_N 持续时间	—	55	93	131	ms
T4	全局复位有效时间	—	-	20	-	ms

4.2.2、FLASH 编程复位

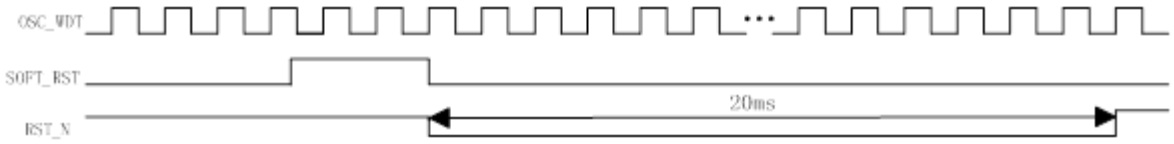
POR_EN 为高时为 flash 的编程模式，此时全局复位信号有效，变低后全局复位信号继续有效 20ms。



编程复位

4.2.3、软件复位

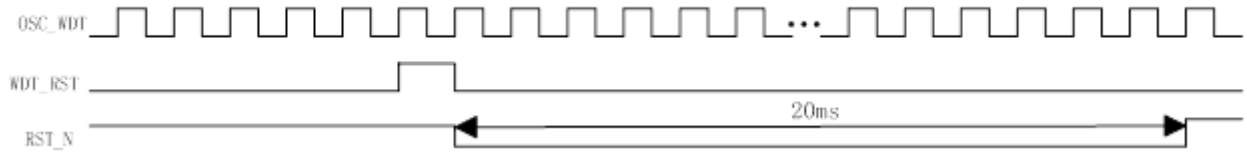
通过向 SOFT_RST SFR 中写入 0x55 使软复位信号 SOFT_RST 有效，使全局复位信号有效 20ms。



软件复位

4.2.4、看门狗定时器溢出复位

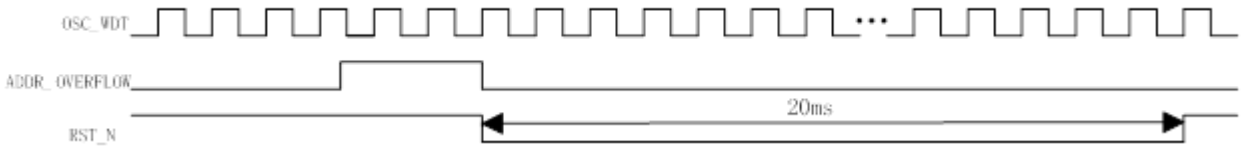
看门狗定时器溢出后 WDT_RST 信号变高，使全局复位 20ms



看门狗溢出复位

4.2.5、PC 指针溢出复位

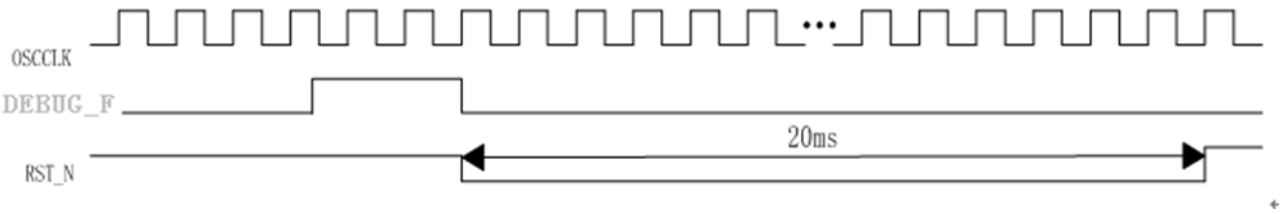
若 MCU 寻址程序存储器时 PC 指针超出了 flash 有效的地址范围，ADDR_OVERFLOW 信号变高，使全局复位 20ms。



PC 指针溢出复位

4.2.6、JTAG 调试复位

若 MCU 处于 JTAG 调试模式下，通过调试工具发送复位命令，将使全局复位信号有效 20ms。



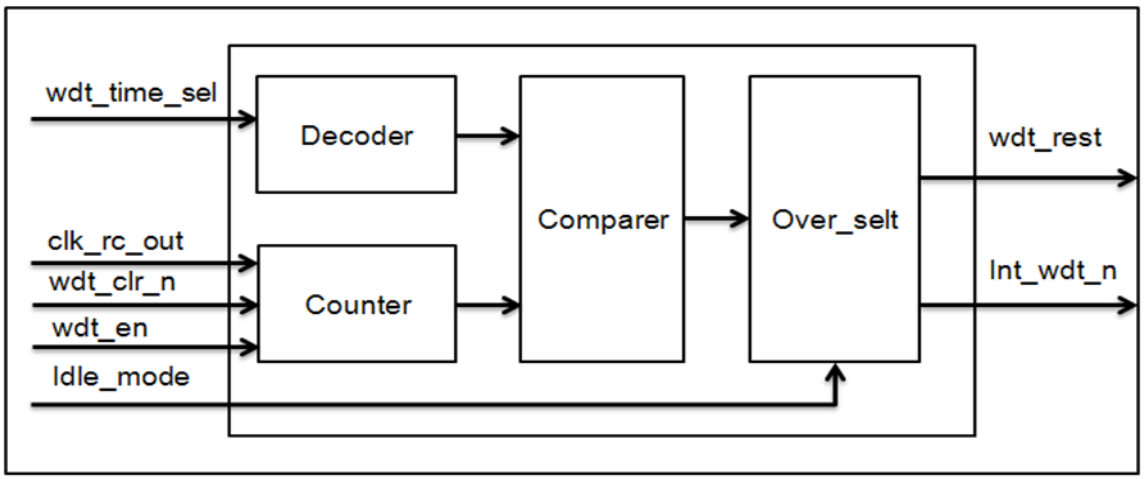
JTAG_DEBUG 复位

4.3 、WDT

GDMC181SXXA 内部有一个使用系统时钟的可编程看门狗定时器（WDT），时钟频率为 32K。MCU 处于 idle mode 时，看门狗溢出输出中断信号（INT_WDT）唤醒 MCU；其他情况下看门狗溢出输出复位信号（WDT_RST）使系统复位 20ms。

非 Idle 模式下，当看门狗定时器溢出时，WDT 将强制 CPU 进入复位状态。为了防止复位，必须在溢出发生前由应用软件重新触发 WDT。看门狗的功能由看门狗使能寄存器（WDT_EN）和看门狗定时器控制寄存器（WDT_CTRL）控制。

注意：JTAG 调试模式下 WDT 模块不工作。



WDT 框图

看门狗时钟寄存器:

看门狗用 32K 时钟完成定时功能可以实现从 18ms 到 2.3s 的定时。定时长度由 SFR（WDT_CTRL）控制，如下表所示：

看门狗时钟寄存器 WDT_CTRL - 0x91h

WDT_CTRL<3:0>	间隔
000	18ms
001	36ms
010	72ms
011	144ms
100	288ms
101	576ms
110	1152ms
111	2304ms

看门狗使能寄存器 WDT_EN - 0x92h

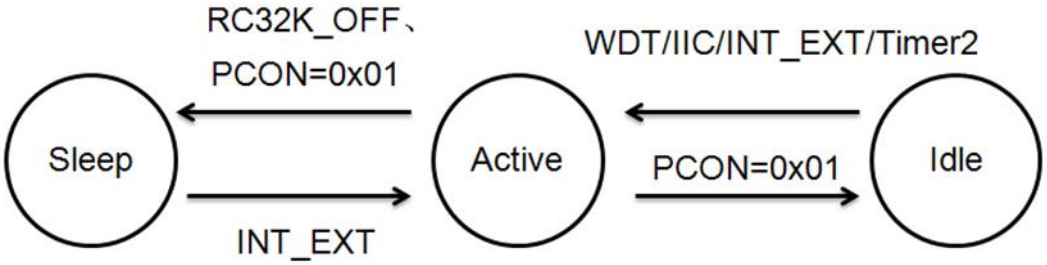
SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDT_EN	WDT_EN<7:0>							
SFR	0x00							

写 0x55 时关闭看门狗，写其他值开启看门狗。

看门狗定时器在复位结束后一直工作。看门狗定时器清零是通过写 WDT_CTRL 寄存器完成的，无论向此寄存器中写入何值都会使看门狗定时器清零，写操作 WDT_CTRL 后需要间隔 30us 才能再次写操作此寄存器。

4.4 、空闲模式和睡眠模式

在空闲模式、睡眠模式中，将没有使用的端口包括不用来唤醒的 SNS 设置为 IO 口并输出低可以降低功耗，在被唤醒后，再将状态还原。



工作模式

空闲模式：GDMC181SXXA 向特殊功能寄存器 PCON 的最低位写入 1 使 MCU 进入空闲模式（idle mode）。进入 idle 模式后，系统时钟停止，RC32K 时钟开启。复位或有效中断均可退出空闲模式。

在空闲模式下，GDMC181SXXA 可以由以下中断进行唤醒：INT_WDT（看门狗中断）、INT_EXT、INT_IIC（IIC 中断）、Timer2。由于在外部中断发生后要 WDT 时钟进行 1ms 定时等待 OSC 时钟稳定，所以在进入空闲模式之前建议将定时器以及串口中断关闭，在唤醒后再开启相应的中断。

在空闲模式下，开启按键扫描实现多按键唤醒，为了降低平均功耗，可以进入空闲模式前，将按键扫描配置为并联扫描模式，配置好按键扫描参数、WDT 唤醒时间，唤醒后芯片工作时间建议应大于等于 3 倍按键扫描时间，软件判断按键阈值是否退出空闲模式。

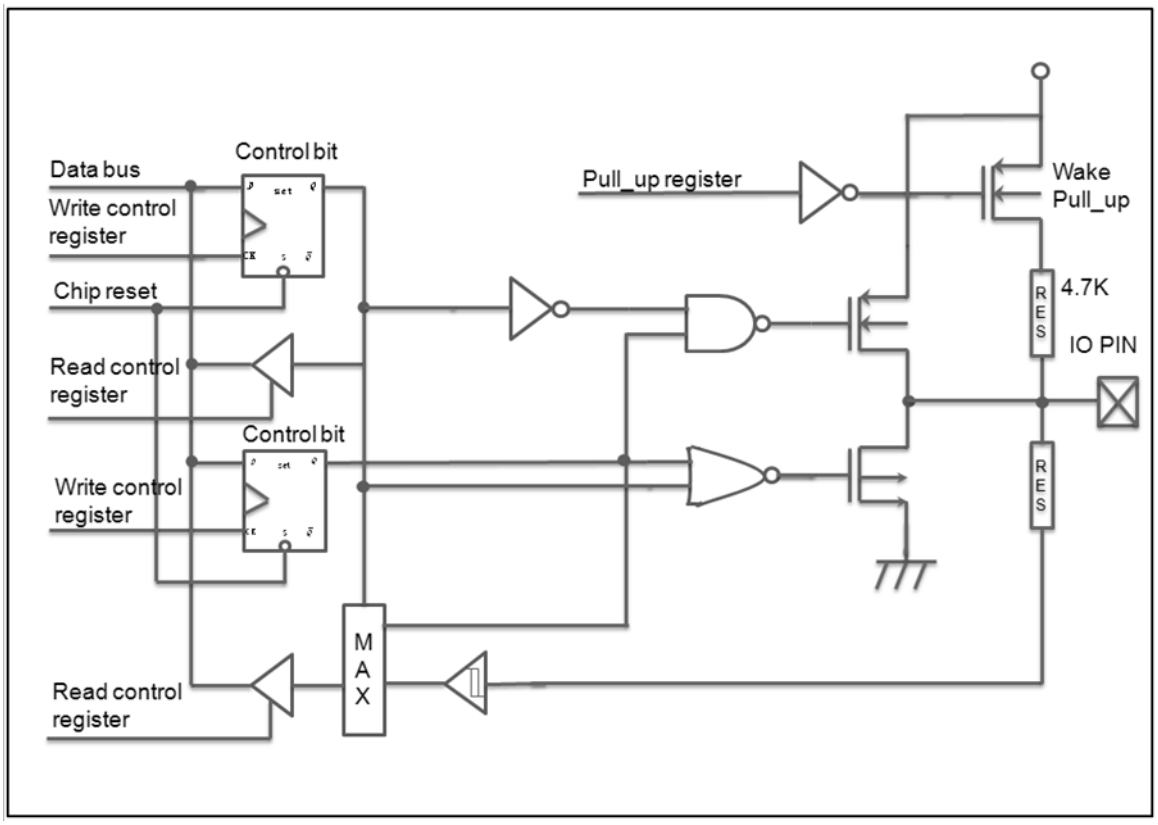
睡眠模式：RC32K 时钟关闭，其它模块关闭，同时向特殊功能寄存器 PCON 的最低位写入 1 使 MCU 进入睡眠模式（sleep mode）。进入睡眠模式后，OSC 时钟停止，睡眠模式下仅外部中断下降沿可以唤醒芯片。

在正常模式下，中断直接送入内核处理。在睡眠模式下有效中断到来时，开启 OSC 时钟，WDT 时钟开始计数，1ms 后把中断送入内核，内核检测到中断后退出睡眠模式。

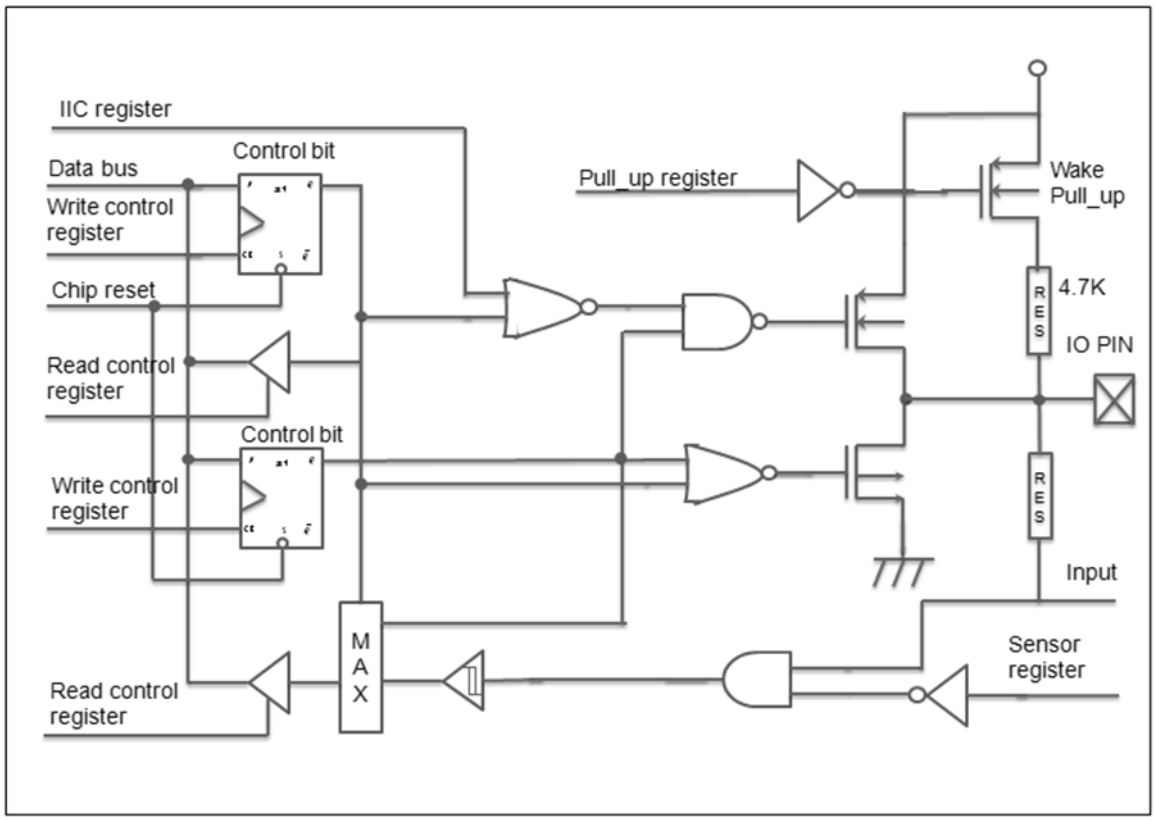
第 5 章、I/O 端口

5.1、I/O 端口

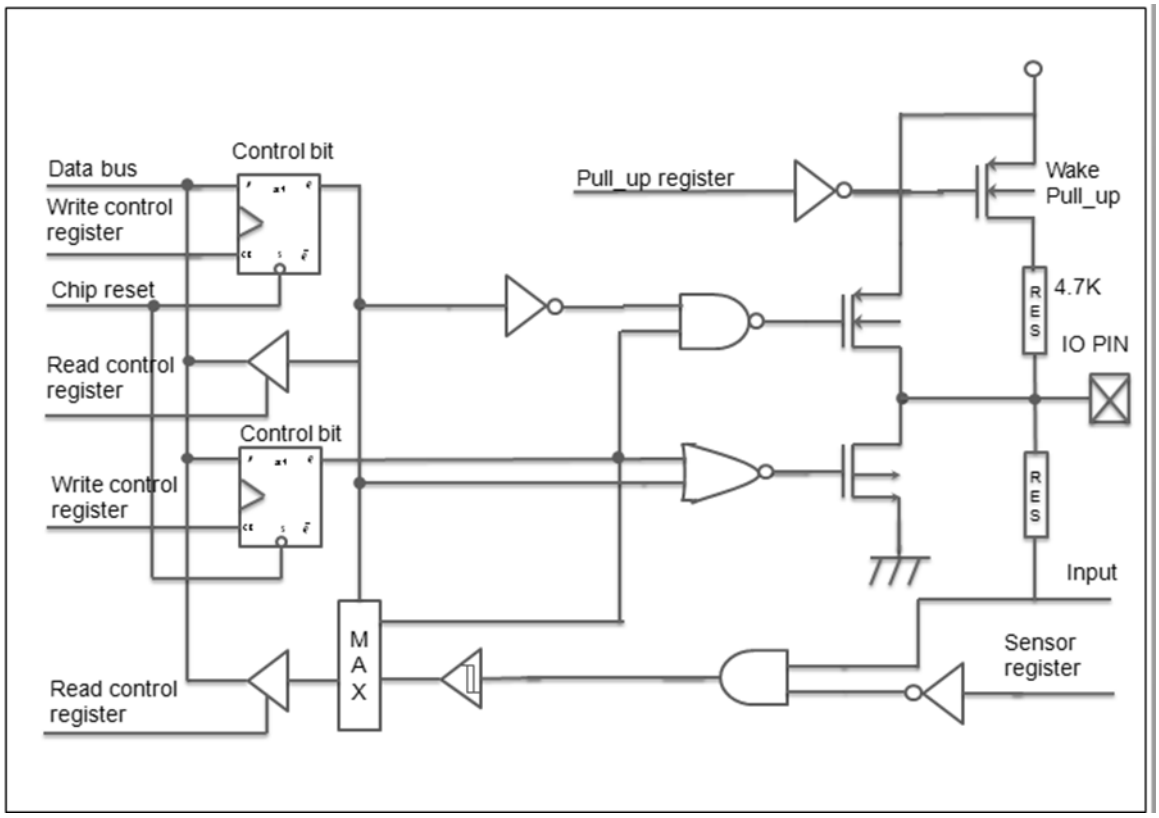
GPIO 端口的一些引脚和器件外设功能复用，当使能外设功能时，该引脚不能作为通用 IO 引脚使用。同一时间不能同时配置成多种功能，否则会引起功能错乱。PA0/PA1 作为 IIC 通信口时，开漏输出，需要接上拉电阻。



普通 IO 结构图


IIC register
Control bit
Data bus
Write control register
Chip reset
Read control register
Control bit
Write control register
Read control register
MAX
Pull_up register
Wake Pull_up
4.7K
RES
IO PIN
RES
Input
Sensor register

PA0/PA1 结构



带 Sensor 功能的 IO 结构

TRISX 寄存器（方向寄存器）：TRISX 置 1 将对应的引脚配置为输入，清零将对应的引脚配置为输出

DATAX 寄存器（数据寄存器）：DATAX 置 1 将对应的引脚配置输出高，清零将对应的引脚配置输出低

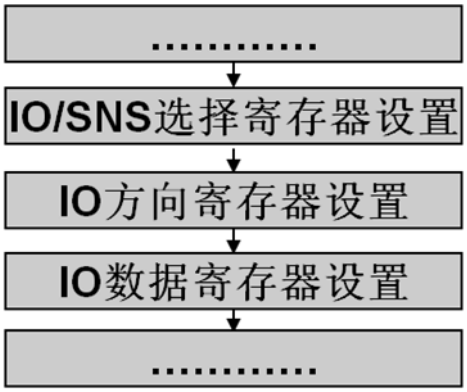
PU_PX 寄存器（上拉电阻使能寄存器）：PU_PX 置 1 对应的引脚上拉电阻使能，清零对应的引脚不使能上拉电阻。

ODRAIN_EN 寄存器（PA 口开漏输出使能）：ODRAIN_EN 置 1 对应的引脚使能开漏输出，清零则不使能开漏输出功能，使能 IIC 功能后自动开启开漏输出。

COM_EN 寄存器（PB 口 COM 使能寄存器）：COM_EN 置 1 对应的引脚使能 COM 口功能，清零则不使能 COM 口功能。

5.2、IO 配置

将端口设置为 GPIO 时，需要对以下 3 组寄存器都进行相应的设置。



IO 配置流程

5.3、PA 端口

PA 端口是 2 位宽的双向端口，PA0 和 PA1 选择为 IIC 通信口时，需外接上拉电阻。

PA 数据寄存器 DATAA - 0x9Eh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DATAA	-	-	-	-	-	-	PA<1>/SDA	PA<0>/SCL
POR	-	-	-	-	-	-	1	1

Bit7-2：保留

Bit1：PA<1>：通用 IO 口；SDA：IIC 数据线

Bit0：PA<0>：通用 IO 口；SCL：IIC 时钟线

PA 方向寄存器 TRISA - 0x9Ah

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	-	-	-	-	-		TRISA<1:0>	
POR	-	-	-	-	-	-	1	1

Bit[7:2]：保留

Bit1:

1:设置为 IO 输入

0:设置为 IO 输出

Bit0:

1:设置为 IO 输入

0:设置为 IO 输出

PA 上拉电阻控制寄存器 PU_PA - 0xA7h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PU_PA	-	-	-	-	-	-	PUA1	PUA0
POR	-	-	-	-	-	-	1	1

Bit[7:2]：保留

Bit1:

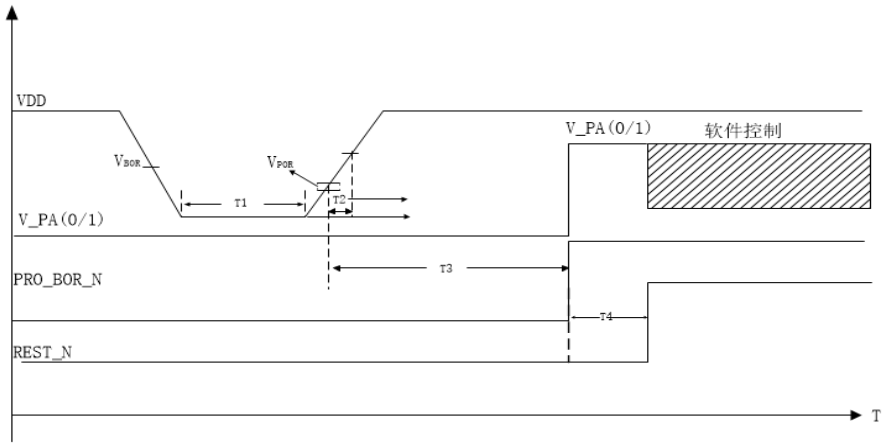
1:使能上拉

0:禁止使能上拉

Bit0:

1:使能上拉

0:禁止使能上拉



PA0/1 上电状态

注：PUA0/1 上电复位默认为 1，开启上拉电阻。在 T4(参考 4.2.1)时间内 PA0/1 口为高,T4 时间后由软件控制，应用时请注意上电默认状态，配合使用。

PA 开漏输出使能寄存器 ODRAIN_EN - 0xB0h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ODRAIN_EN	-	-	-	-	-	-	PA1	PA0
POR	-	-	-	-	-	-	0	0

Bit[7:2]：保留

Bit1:

1:使能开漏

0:禁止使能开漏

Bit0:

1:使能开漏

0:禁止使能开漏

5.4、PB 端口

PB 端口是 8 位有效的双向端口，每个端口都为推挽输出。

PB 数据寄存器 DATAB - 0x9Fh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DATAB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
POR	1	1	1	1	1	1	1	1

Bit[7:0]：数据；

PB 方向寄存器 TRISB - 0x9Bh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0
POR	1	1	1	1	1	1	1	1

Bit[7:0]：对应的位为 1：设置为 IO 输入；对应的位为 0：设置为 IO 输出；

PB 上拉电阻控制寄存器 PU_PB - 0xEAh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PU_PB	PUB7	PUB6	PUB5	PUB4	PUB3	PUB2	PUB1	PUB0
POR	0	0	0	0	0	0	0	0

Bit[7:0]：对应的位为 1：使能上拉；对应的位为 0：禁止使能上拉；

PB 口 LCD_COM 使能寄存器 COM_EN - 0xABh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
COM_EN	-	-	PB5	PB4	PB3	PB2	PB1	PB0
POR	-	-	0	0	0	0	0	0

Bit[7:6]：保留

Bit[5:0]：对应的位为 1：使能 LCD COM 驱动功能；对应的位为 0：禁止使能 LCD COM 驱动功能。

5.5、PC 端口

PC 端口是 8 位有效的双向端口，每个端口都为推挽输出。

PC 数据寄存器 DATAC - 0xA1h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DATAC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
POR	1	1	1	1	1	1	1	1

Bit[7:0]：数据；

PC 方向寄存器 TRISC - 0x9Ch

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
POR	1	1	1	1	1	1	1	1

Bit[7:0]：对应的位为 1：设置为 IO 输入；对应的位为 0：设置为 IO 输出；

PC 上拉电阻控制寄存器 PU_PC - 0xAAh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PU_PC	PUC7	PUC6	PUC5	PUC4	PUC3	PUC2	PUC1	PUC0
POR	0	0	0	0	0	0	0	0

Bit[7:0]：对应的位为 1：使能上拉；对应的位为 0：禁止使能上拉；

5.6、PD 端口

PD 端口是 8 位有效的双向端口，每个端口都为推挽输出。

PD 数据寄存器 DATAD - 0xA2h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DATAD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
POR	1	1	1	1	1	1	1	1

Bit[7:0]：数据；

PD 方向寄存器 TRISD - 0x9Dh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISD	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
POR	1	1	1	1	1	1	1	1

Bit[7:0]：对应的位为 1：设置为 IO 输入；对应的位为 0：设置为 IO 输出；

PD 上拉电阻控制寄存器 PU_PD - 0xEBh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PU_PD	PUD7	PUD6	PUD5	PUD4	PUD3	PUD2	PUD1	PUD0
POR	0	0	0	0	0	0	0	0

Bit[7:0]：对应的位为 1：使能上拉；对应的位为 0：禁止使能上拉;

5.7、GPIO 初始化 C 语言参考

```
//--------------------------------例--------------------------------//
//函数名称：  void Init_IO(void)
//函数功能：  IO 口初始化,将没有用到的或悬空的 IO 口，设置为 IO 输出为低
//输入参数：  无
//返 回 值：  无
//--------------------------------例--------------------------------//
#define SET_PA0_OD_OFF ODRAIN_EN &= ~0x01
#define SET_PA0_OD_ON ODRAIN_EN |= 0x01
#define SET_PA0_PU_ON PU_PA |= 0x01
void Init_IO(void)
{
    SET_PA0_OD_OFF;//不使能 PA0 开漏
    SET_PA0_PU_ON;//使能 PA0 上拉
    SET_PA0_IO_OUT;//设置为 IO 输出;
    SET_PA0_H;//输出为高;

    SET_PB0_IO_OUT;//设置为 IO 输出;
    SET_PB0_H;//输出为高;
    SET_PB1_IO_IN;//设置为 IO 输入;
}
```

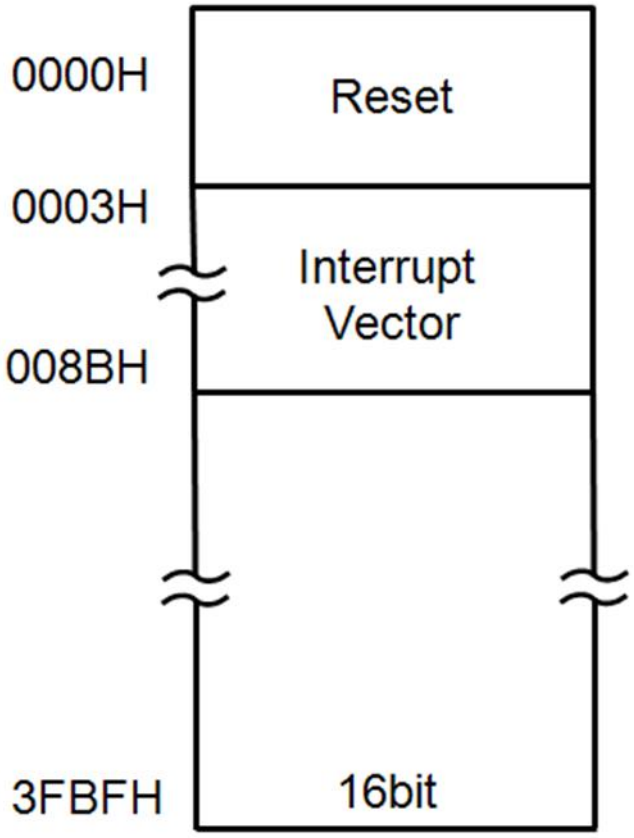
第 6 章、中断

6.1、中断源及入口地址

GDMC181SXXA中使用的中断信息如表所示：

中断源	中断向量	优先级	中断标志	使能位	优先级控制
INT_EXT0(外部中断 0)	03h	0	TCON.1(IE0)	IEN0.0	IPL0.0
TF0(timer0 中断)	0Bh	1	TCON.5(TF0)	IEN0.1	IPL0.1
INT_EXT1(外部中断 1)	13h	2	TCON.3(IE1)	IEN0.2	IPL0.2
TF1(timer1 中断)	1Bh	3	TCON.7(TF1)	IEN0.3	IPL0.3
INT_EXT2(外部中断 2)	4Bh	9	IRCON1.2(IE2)	IEN1.2	IPL1.2
INT_IIC(IIC 中断)	53h	10	IRCON1.3(IE3)	IEN1.3	IPL1.3
INT_ADC(ADC 中断)	5Bh	11	IRCON1.4(IE4)	IEN1.4	IPL1.4
INT_CTK (CTK 中断)	63h	12	IRCON1.5(IE5)	IEN1.5	IPL1.5
INT_LED (LED 中断)	6Bh	13	IRCON1.6(IE6)	IEN1.6	IPL1.6
INT_WDT(WDT 中断)	73h	14	IRCON1.7(IE7)	IEN1.7	IPL1.7
INT_TIMER2(Timer2 中断)	7Bh	15	IRCON2.0(IE8)	IEN2.0	IPL2.0
INT_LVDT(LVDT 中断)	83h	16	IRCON2.1(IE9)	IEN2.1	IPL2.1
INT_UART(UART 中断)	8Bh	17	IRCON2.2(IE10)	IEN2.2	IPL2.2
			UART_STATE.3	UART_CON2.2	-
			UART_STATE.4	UART_CON2.3	-

当芯片发生复位信号时，程序从0000H地址开始执行。当发生一个中断信号时，程序将跳转到中断向量程序地址执行中断服务程序。



程序存储结构

6.2、中断 SFR

中断控制中用到的 SFR 有：

- IEN0---SFR 地址 A8h
- IEN1---SFR 地址 E6h
- IEN2---SFR 地址 E7h
- IPL0---SFR 地址 B8h
- IPL1---SFR 地址 F6h
- IPL2---SFR 地址 F4h
- IRCON1---SFR 地址 F1h
- IRCON2---SFR 地址 E1h
- EXT_INT_CON---SFR 地址 AEh
- PERIPH_IO_SEL---SFR 地址 ADh

IEN0、IEN1、IEN2 寄存器用来使能相应的中断。IPL0、IPL1、IPL2 寄存器设置相应中断的优先级。IRCON1 和 IRCON2 寄存器为扩展的中断源提供标志位。EXT_INT_CON 寄存器用来选择外部中断 0、外部中断 1、外部中断 2 检测极性。PERIPH_IO_SEL 寄存器开启外部中断检测功能。

IEN0 寄存器--- 0xA8h

位	功能	R/W	复位值
IEN0.7	EA-中断允许位。EA=0 屏蔽所有的中断(EA 优先于中断源各自的中断使能位)。EA=1，中断打开，每个中断源的中断请求是允许还是被禁止，还需由各自的允许位确定。	R/W	0
IEN0.6-4	保留	aaa	000
IEN0.3	ET1-定时器 1 溢出中断允许位。ET1=0,禁止定时器 1(TF1)申请中断。ET1=1，允许 TF1 标志位申请中断。	R/W	0
IEN0.2	EX1-INT_EXT1 允许位。EX1=0，禁止 INT_EXT1 申请中断。EX1=1，允许 INT_EXT1 申请中断。	R/W	0
IEN0.1	ET0-定时器 0 溢出中断允许位。ET0=0,禁止定时器 0(TF0)申请中断。ET0=1，允许 TF0 标志位申请中断。	R/W	0
IEN0.0	EX0-INT_EXT0 允许位。EX0=0，禁止 INT_EXT0 申请中断。EX0=1，允许 INT_EXT0 申请中断。	R/W	0

IEN1 寄存器--- 0xE6h

位	功能	R/W	复位值
IEN1.7	EX7-INT_WDT 中断允许位。EX7=0，禁止 INT_WDT 申请中断。EX7=1，允许 INT_WDT 申请中断。	R/W	0

IEN1.6	EX6-INT_LED 中断允许位。EX6=0，禁止 INT_LED 申请中断。EX6=1，允许 INT_LED 申请中断。	R/W	0
IEN1.5	EX5-INT_CTK 中断允许位。EX5=0，禁止 INT_CTK 申请中断。EX5=1，允许 INT_CTK 申请中断。	R/W	0
IEN1.4	EX4-INT_ADC 中断允许位。EX4=0，禁止 INT_ADC 申请中断。EX4=1，允许 INT_ADC 申请中断。	R/W	0
IEN1.3	EX3-INT_IIC 中断允许位。EX3=0，禁止 INT_IIC 申请中断。EX3=1，允许 INT_IIC 申请中断。	R/W	0
IEN1.2	EX2- INT_EXT2 允许位。EX2=0，禁止 INT_EXT2 申请中断。EX2=1，允许 INT_EXT2 申请中断。	R/W	0
IEN1.1-0	保留	aa	00

IEN2 寄存器---0xE7h

位	功能	R/W	复位值
IEN2.7-3	保留	aaaaa	00000
IEN2.2	EX10-INT_UART 中断允许位。EX10=0，禁止 INT_UART 申请中断。EX10=1，允许 INT_UART 申请中断。	R/W	0
IEN2.1	EX9- INT_LVDT 允许位。EX9=0，禁止 INT_LVDT 申请中断。EX9=1，允许 INT_LVDT 申请中断。	R/W	0
IEN2.0	EX8- INT_TIMER2 允许位。EX8=0，禁止 INT_TIMER2 申请中断。EX8=1，允许 INT_TIMER2 申请中断。	R/W	0

IPL0 寄存器--- 0xB8h

位	功能	R/W	复位值
IPL0.7-4	保留	raaa	0000
IPL0.3	PX3-TF1(timer1 中断)优先级选择位。PX3=0 时 TF1(timer1 中断)为低优先级,PX3=1 时 TF1(timer1 中断)为高优先级。	R/W	0
IPL0.2	PX2- INT_EXT1 中断优先级选择位。PX2=0 时 INT_EXT1 为低优先级，PX2=1 时 INT_EXT1 为高优先级。	R/W	0
IPL0.1	PX1-TF0(timer0 中断)优先级选择位。PX1=0 时 TF0(timer0 中断)为低优先级,PX1=1 时 TF0(timer0 中断)为高优先级。	R/W	0
IPL0.0	PX0- INT_EXT0 中断优先级选择位。PX0=0 时 INT_EXT0 为低优先级，PX0=1 时 INT_EXT0 为高优先级。	R/W	0

IPL1 寄存器---0xF6h

位	功能	R/W	复位值
IPL1.7	PX9- INT_WDT 中断优先级选择位。PX9=0 时 INT_WDT 为低优先级，PX9=1 时 INT_WDT 为高优先级。	R/W	0
IPL1.6	PX8- INT_LED 中断优先级选择位。PX8=0 时 INT_LED 为低优先级，PX8=1 时 INT_LED 为高优先级。	R/W	0
IPL1.5	PX7- INT_CTK 中断优先级选择位。PX7=0 时 INT_CTK 为低优先级，PX7=1 时 INT_CTK 为高优先级。	R/W	0
IPL1.4	PX6- INT_ADC 中断优先级选择位。PX6=0 时 INT_ADC 为低优先级，PX6=1 时 INT_ADC 为高优先级。	R/W	0
IPL1.3	PX5- INT_IIC 中断优先级选择位。PX5=0 时 INT_IIC 为低优先级，PX5=1 时 INT_IIC 为高优先级。	R/W	0
IPL1.2	PX4- INT_EXT2 中断优先级选择位。PX4=0 时 INT_EXT2 为低优先级，PX4=1 时 INT_EXT2 为高优先级。	R/W	0
IPL1.1-0	保留	aa	00

IPL2 寄存器---0xF4h

位	功能	R/W	复位值
IPL2.7-3	保留	aaaaa	00000
IPL2.2	PX12- INT_UART 中断优先级选择位。PX12=0 时 INT_UART 为低优先级，PX12=1 时 INT_UART 为高优先级。	R/W	0
IPL2.1	PX11- INT_LVDT 中断优先级选择位。PX11=0 时 INT_LVDT 为低优先级，PX11=1 时 INT_LVDT 为高优先级。	R/W	0
IPL2.0	PX10- INT_Timer2 中断优先级选择位。PX10=0 时 INT_Timer2 为低优先级，PX10=1 时 INT_Timer2 为高优先级。	R/W	0

IRCON1 寄存器--- 0xF1h

位	功能	R/W	复位值
IRCON1.7	IE7-INT_WDT 中断标志位	R/W	0
IRCON1.6	IE6-INT_LED 中断标志位。	R/W	0
IRCON1.5	IE5-INT_CTK 中断标志位。	R/W	0
IRCON1.4	IE4-INT_ADC 中断标志位。	R/W	0
IRCON1.3	IE3-INT_IIC 中断标志位。	R/W	0
IRCON1.2	IE2-INT_EXT2 中断标志位。	R/W	0
IRCON1.1-0	保留	rr	00

IRCON2 寄存器---0xE1h

位	功能	R/W	复位值
IRCON2.7-3	保留	aaaaa	00000

IRCON2.2	IE10-INT_UART 中断标志位。	R/W	0
IRCON2.1	IE9-INT_LVDT 中断标志位。	R/W	0
IRCON2.0	IE8-INT_TIMER2 中断标志位。	R/W	0

EXT_INT_CON 寄存器--- 0xAEh

位	功能	R/W	复位值
EXT_INT_CON.7-6	保留	aa	00
EXT_INT_CON.5-4	外部中断 2(INT_EXT2)触发极性选择位： 01：下降沿触发 10：上升沿触发 00/11：上升沿、下降沿均触发	R/W	01
EXT_INT_CON.3-2	外部中断 1(INT_EXT1)触发极性选择位： 01：下降沿触发 10：上升沿触发 00/11：上升沿、下降沿均触发	R/W	01
EXT_INT_CON.1-0	外部中断 0(INT_EXT0)触发极性选择位： 01：下降沿触发 10：上升沿触发 00/11：上升沿、下降沿均触发	R/W	01

PERIPH_IO_SEL 寄存器--- 0xADh

位	功能	R/W	复位值
PERIPH_IO_SEL.7	保留	R/W	0
PERIPH_IO_SEL.6-5	UART 功能使能 IO 映射选择位： 01：PA0/PA1 使能 UART 功能 10：PD4/PD5 使能 UART 功能 11：PB3/PB4 使能 UART 功能 00：不使能 UART 功能	R/W	00
PERIPH_IO_SEL.4-3	PWM 功能使能 IO 映射选择位： 01：使能 PD1 口 PWM 功能 10：使能 PB0 口 PWM 功能 11：使能 PD1 口 PWM 功能 00：不使能 PWM 功能	R/W	00
PERIPH_IO_SEL.2	INT_EXT2 功能使能位(PD7)： 1：使能 INT_EXT2 功能 0：不使能 INT_EXT2 功能	R/W	0
PERIPH_IO_SEL.1	INT_EXT1 功能使能位(PD6)： 1：使能 INT_EXT1 功能 0：不使能 INT_EXT1 功能	R/W	0

PERIPH_IO_SEL.0	INT_EXT0 功能使能位(PC7)： 1：使能 INT_EXT0 功能 0：不使能 INT_EXT0 功能	R/W	0

6.3、中断响应

当发生中断申请时，CPU根据中断服务程序(ISR)来确定中断的种类。CPU完整的执行ISR，除非有优先级高的中断源申请中断。每个ISR后有RETI(中断返回)指令。执行RETI指令后，CPU继续执行在中断没有发生之前的程序。

ISR 只能被优先级更高的中断申请中断。也就是，低优先级的 ISR 能被高优先级的中断申请中断。高优先级的中断 ISR 能被掉电中断中断。

GDMC181SXXA 执行完当前指令后才响应中断请求。如果正在执行的指令是 RETI 指令，或者访问 IP、IE、EIP、EIE 寄存器时，需要执行一条其他指令后才会响应中断请求。

6.4、中断优先级

GDMC181SXXA 有两个中断优先级：中断级和默认优先级。中断级(最高级、高级和低级)优先于默认优先级。如果允许，掉电中断是唯一一个最高级的中断源。其他的中断源可以设置为高优先级或者低优先级。

每个中断源既可以分配优先级(高或者低)，还有默认的优先级。同一级别中的中断源(例如都为高优先级)的优先级由默认的优先级决定。

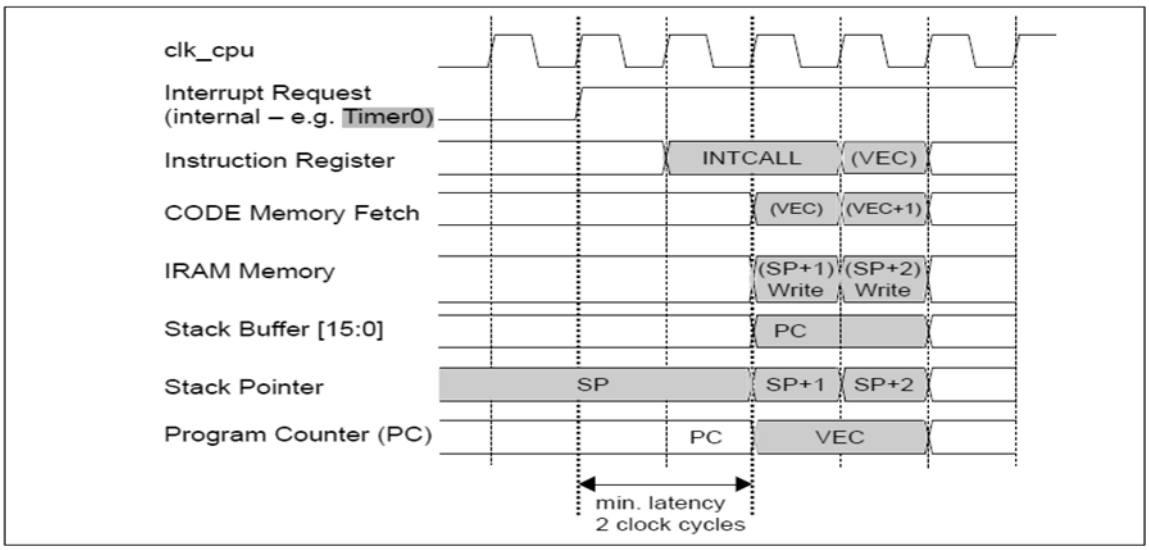
正在进行的中断服务程序只能被优先级高的中断请求中断。

6.5、中断采样

内部定时器和串口是通过各自的 SFR 中的中断标志位来发生中断请求。当每 1 个指令周期的第 1 个时钟周期(C1)结束时，在时钟的上升沿对外部中断进行采样。

INT_EXT 是低电平有效，并且可以通过 TCON SFR 中的 IT0 位来设置选择边缘触发或者电平触发。例如，当 IT0=0 时，INT_EXT 为边缘触发，在采集到 INT_EXT 脚出现由高到低的电平变化时， 外部中断标志位置 1。

为了确保边缘触发型的中断被检测到，相应的端口要首先保持 2 个时钟的高电平，然后保持 2 个时钟的低电平。



中断响应时序

6.6、中断等待

中断的响应时间由 GDMC181SXXA 当前状态决定。最快的响应时间是 5 个指令周期：1 个周期用来检测中断请求，其他 4 个用来执行长调用(LCALL)至 ISR。

当 GDMC181SXXA 在执行 RETI 指令，并且后面为 MUL 或者 DIV 指令时，中断等待的时间最长(13 个指令周期)。这 13 个指令周期分别为：1 个周期用来检测中断请求，3 个用来完成 RETI 指令，5 个用来执行 DIV 或者 MUL 指令，4 个用来执行长调用(LCALL)至 ISR。在这种情况下，响应时间为 13 个时钟周期。

6.7、中断初始化 C 语言参考

```
//-------------------------------------------------------------------//
//函数名称：  void ExtInt0Init(uchar TriggerMode)
//函数功能：  外部中断 0 初始化
//输入参数：
//外部中断 0 触发方式选择，uchar TriggerMode
//          TriggerMode = 0，触发方式为下降沿触发
//          TriggerMode = 1，触发方式为上升沿触发
//          TriggerMode = 2，触发方式为双边沿触发
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
#if ExtInt0_EN
#pragma message "编译，void ExtInt0Init(uchar TriggerMode)函数"
void ExtInt0Init(uchar TriggerMode)
{

    EA_OFF;//关总中断；

    INT0_FUN_SET(1)//0 选择为 IO 口功能，1 选择为 INT 中断功能
    IPL0  |= 0x01;//设置 IN0 中断优级为高
    INT0_INT_FLAG_CLR;//清除 INT0 中断标志

    switch(TriggerMode)
    {
        case 0:

            EXT_INT_CON |= 0X01; //设置为下降沿触发

        break;

        case 1:
            EXT_INT_CON &= ~0X01;
            EXT_INT_CON |= 0X02;     //设置为上升沿触发
        break;

        case 2:
```

```
            EXT_INT_CON |= 0X03; //设置为双边沿触发

        break;


        default:
            EXT_INT_CON |= 0X01; //设置为下降沿触发
        break;
    }
    INT0_IE_SET;//使能外部 INT0 中断
    EA_ON;//开总中断
}
//-------------------------------------------------------------------//
//函数名称:  void Ext0_ISR() interrupt 0
//函数功能:  外部中断 0 中断子函数
//输入参数:  无
//输出参数:  无
//返 回 值:  无
//-------------------------------------------------------------------//
void Ext0_ISR() interrupt 0
{

    INT0_INT_FLAG_CLR;//清除 INT0 中断标志

}
#endif

//-------------------------------------------------------------------//
//函数名称:  void ExtInt1Init(uchar TriggerMode)
//函数功能:  外部中断 1 初始化
//输入参数:
//外部中断 1 触发方式选择, uchar TriggerMode
//          TriggerMode = 0, 触发方式为下降沿触发
//          TriggerMode = 1, 触发方式为上升沿触发
//          TriggerMode = 2, 触发方式为双边沿触发
//输出参数:  无
//返 回 值:  无
//-------------------------------------------------------------------//
#if ExtInt1_EN
#pragma message "编译, void ExtInt1Init(uchar TriggerMode)函数"
void ExtInt1Init(uchar TriggerMode)
```

```
{
    EA_OFF;//关总中断;
    INT1_FUN_SET(1)//0 选择为 IO 口功能，1 选择为 INT 中断功能
    INT1_INT_FLAG_CLR;//清除 INT1 中断标志
    INT1_IP_SET;//设置 IN1 中断优级为高
    switch(TriggerMode)
    {
        case 0:

            EXT_INT_CON |= 0X04; //设置为下降沿触发

        break;

        case 1:
            EXT_INT_CON &= ~0X04;
            EXT_INT_CON |= 0X08;     //设置为上升沿触发
        break;

        case 2:
            EXT_INT_CON |= 0X0C; //设置为双边沿触发

        break;


        default:
            EXT_INT_CON |= 0X04; //设置为下降沿触发
        break;
    }
    INT1_IE_SET;//使能外部 INT1 中断
    EA_ON;//开总中断
}
//-------------------------------------------------------------------//
//函数名称:  void Ext1_ISR() interrupt 2
//函数功能:  外部中断 1 中断子函数
//输入参数:  无
//输出参数:  无
//返 回 值:  无
//-------------------------------------------------------------------//
void Ext1_ISR() interrupt 2
{
```

```
    INT1_INT_FLAG_CLR;//清除 INT1 中断标志
}
#endif


//-------------------------------------------------------------------//
//函数名称：  void ExtInt2Init(uchar TriggerMode)
//函数功能：  外部中断 2 初始化
//输入参数：
//外部中断 1 触发方式选择，uchar TriggerMode
//          TriggerMode = 0，触发方式为下降沿触发
//          TriggerMode = 1，触发方式为上升沿触发
//          TriggerMode = 2，触发方式为双边沿触发
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
#if ExtInt2_EN
#pragma message "编译，void ExtInt2Init(uchar TriggerMode)函数"
void ExtInt2Init(uchar TriggerMode)
{
    EA_OFF;//关总中断；
    INT2_FUN_SET(1)//0 选择为 IO 口功能，1 选择为 INT 中断功能
    INT2_INT_FLAG_CLR;//清除 INT2 中断标志
    INT2_IP_SET;//设置 IN2 中断优级为高
    switch(TriggerMode)
    {
        case 0:

            EXT_INT_CON |= 0X10; //设置为下降沿触发

        break;

        case 1:
            EXT_INT_CON &= ~0X10;
            EXT_INT_CON |= 0X20;     //设置为上升沿触发
        break;

        case 2:
            EXT_INT_CON |= 0X30; //设置为双边沿触发

        break;
```

```
        default:
            EXT_INT_CON |= 0X10; //设置为下降沿触发
        break;
    }
    INT2_IE_SET;//使能外部 INT2 中断
    EA_ON;//开总中断
}
//-----------------------------------------------------------------//
//函数名称：  void Ext2_ISR() interrupt 9
//函数功能：  外部中断 2 中断子函数
//输入参数：  无
//输出参数：  无
//返 回 值：  无
//-----------------------------------------------------------------//
void Ext2_ISR() interrupt 9
{
    INT2_INT_FLAG_CLR;//清除 INT2 中断标志
}
#endif
```

第 7 章、定时器

GDMC181SXXA 包含三个定时器（定时器 0、定时器 1 和定时器 2）。定时器 0、1 的时钟为系统时钟的 12 分频时钟，定时器 2 使用内部 RC(32KHZ)时钟或者外部晶振时钟。每个定时器包含一个 16 位的寄存器。在被访问时以两个字节的形式出现：一个低字节（TL0、TL1、TIMER2_SET_L）和一个高字节（TH0、TH1、TIMER2_SET_H）。通过设置 IEN0 寄存器中的 ET0 位使能定时器 0 中断，通过设置 IEN0 寄存器中的 ET1 位使能定时器 1 中断。通过设置 IEN2 寄存器中的 EX8 位使能定时器 2 中断。

●定时器 0 - TL0和TH0
●定时器 1 - TL1和TH1
●定时器 2 - TIMER2_SET_L和TIMER2_SET_H

7.1、定时器 0 和定时器 1

定时器 0/1有四种运行模式，由TMOD SFR和TCON SFR控制。
定时器0/1四种模式如下：
●13位定时器/计数器 (模式0)
●16位定时器/计数器 (模式1)
●自动重载初值的8位计数器(模式2)
●两个8位计数器 (模式3, 只用于定时器/计数器 0)

TMOD 寄存器 -- 0x89h

位	功能	R/W	复位值
TMOD.7	GATE-定时器1门控位。当GATE=0,TR1=1时，开启定时器1。	R/W	0
TMOD.6	保留。	R/W	0
TMOD.5-4	M1-定时器1模式选择Bit 1，M0-定时器1模式选择Bit 0， M1M0： 00=模式 0 - 13位定时器/计数器 01=模式 1 - 16位定时器/计数器 10=模式 2 - 自动重载初值的8位计数器 11=模式 3 - 两个8位计数器	aa	00
TMOD.3	GATE-定时器 0门控位。当GATE=0,TR0=1时，开启定时器0。	R/W	0
TMOD.2	保留。	R/W	0

TMOD.1-0	M1-定时器0模式选择Bit 1，M0-定时器0模式选择Bit 0 M1M0 00=模式 0 - 13位定时器/计数器 01=模式 1 - 16位定时器/计数器 10=模式 2 - 自动重载初值的8位计数器 11=模式 3 - 两个8位计数器	aa	00

TCON 寄存器 -- 0x88h

位	功能	R/W	复位值
TCON.7	TF1-定时器1溢出中断标志位。当定时器1计数溢出时该位置1，进入中断服务程序后，该位由硬件自动清0。	R/W	0
TCON.6	TR1-定时器1使能位。置1时开启定时器1；置0时关闭定时器1。由软件清零或置位，硬件不自动清零。	R/W	0
TCON.5	TF0-定时器0溢出中断标志位。当定时器0计数溢出时该位置1，当CPU进入中断服务程序时，该位由硬件自动清0。	R/W	0
TCON.4	TR0-定时器0使能位。当置1时开启定时器0；置0时关闭定时器0。由软件清零或置位，硬件不自动清零。	R/W	0
TCON.3	IE1- 外部中断1中断标志位。	R/W	0
TCON.2	保留	R/W	1
TCON.1	IE0- 外部中断0中断标志位。	R/W	0
TCON.0	保留	R/W	1

定时器 0 和定时器 1 中断使能寄存器 IEN0—0xA8h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IEN0	-	-	-	-	IEN0.3	-	IEN0.1	-
POR	0	0	0	0	0	0	0	0

IEN0.1：定时器 0 中断使能位；IEN0.1=1 允许定时器 0 中断；IEN0.1=0 禁止定时器 0 申请中断。

IEN0.3：定时器 1 中断使能位；IEN0.3=1 允许定时器 1 中断；IEN0.3=0 禁止定时器 1 申请中断。

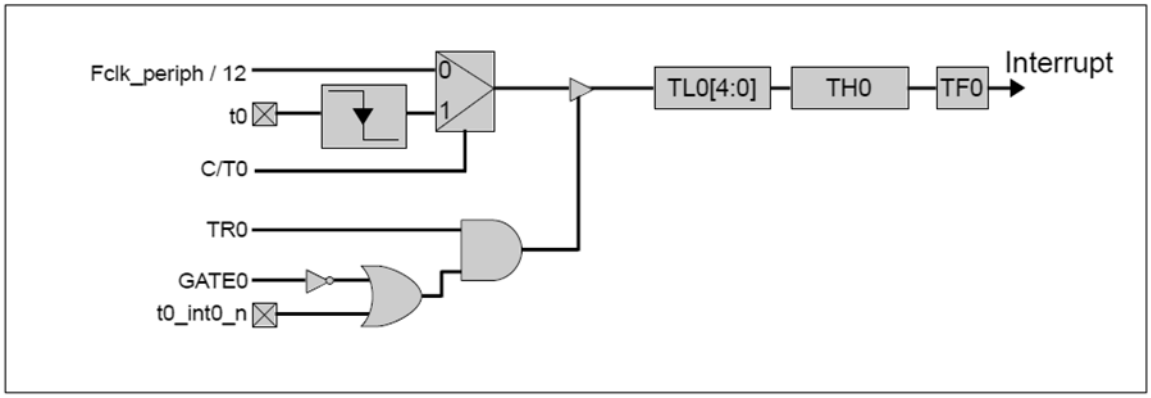
定时器 0 和定时器 1 中断优先级选择寄存器 IPL0—0xB8h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IPL0	-	-	-	-	IPL0.3	-	IPL0.1	-
POR	0	0	0	0	0	0	0	0

IPL0.1：定时器0中断优先级选择位；IPL0.1=1：定时器0中断高优先级；IPL0.1=0：定时器0中断低优先级。

IPL0.3：定时器1中断优先级选择位；IPL0.3=1：定时器1中断高优先级；IPL0.3=0：定时器1中断低优先级。

模式 0 :13 位定时器/计数器



Timer0/1-Mode0

在模式 0 下定时器 0 和定时器 1 的工作过程相同，如图所示。在模式 0 中，定时器为 13 位的计数器，其 0-4 位为 TL0(或者 TL1)，另外 8 位为 TH0(或者 TH1)。TCON 寄存器中的使能位(TR0/TR1)来控制定时器的开启和关闭。

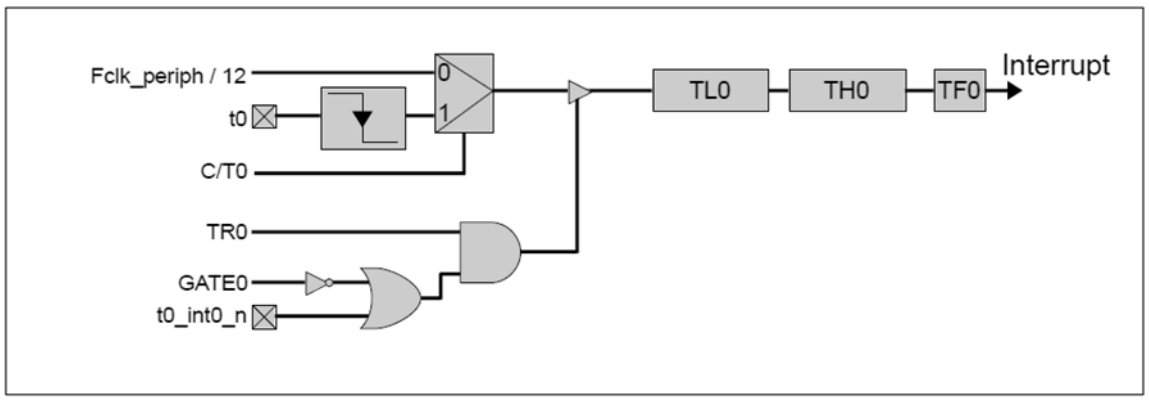
当 GATE=0 时，定时器对选定的时钟源(clk/12)进行计数；当 GATE=1 时，定时器 0 是否计数取决于 INT_EXT0 引脚的信号，当 INT_EXT0 由 0 变 1 时，开始计数；当 INT_EXT0 由 1 变 0 时，停止计数。

当 13 位计数器计数累积到 1FFFh(全 1 时)，计数器清 0(全 0)，并且 TF0(或者 TF1)置位。

在模式 0 中，TL0(或者 TL1)的高 3 位是不确定的，在读计数值时应屏蔽掉或忽略这 3 位。

模式 1 :16 位定时器/计数器

定时器 0 和定时器 1 的模式 1 是相同的，如图所示。在模式 1 中，定时器为 16 位的计数器。LSB 寄存器(TL0 或者 TL1)的所有 8 位都被使用。当计数器计数累计至 FFFFh 时，计数器清为全 0。除此之外，模式 1 和模式 0 是相同的。

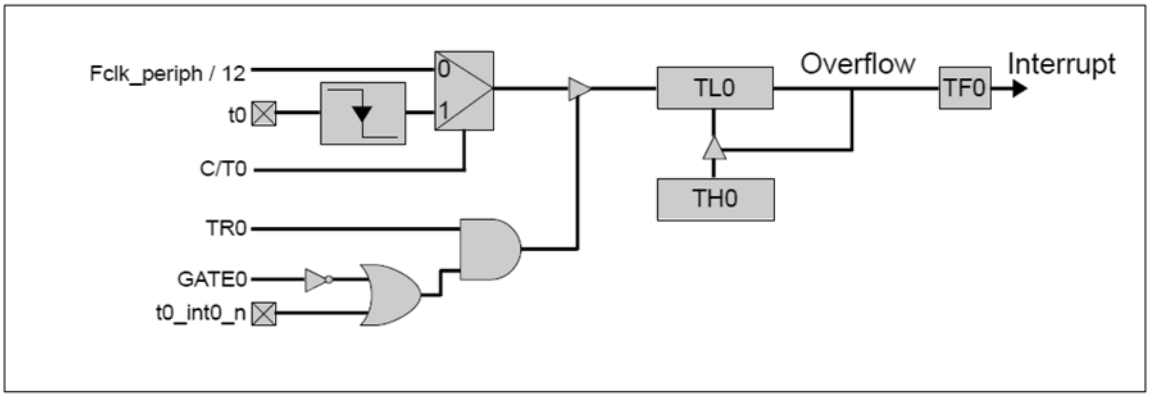


Timer0/1-Mode1

模式 2:自动重载初值的 8 位计数器

定时器 0 和定时器 1 的模式 2 是相同的。在模式 2 中，定时器为一个带有自动重载初值的 8 位计数器。这个计数器就是 LSB 寄存器(TL0 或者 TL1)，需要重载的初值保存在 MSB 寄存器(TH0 或者 TH1)中。

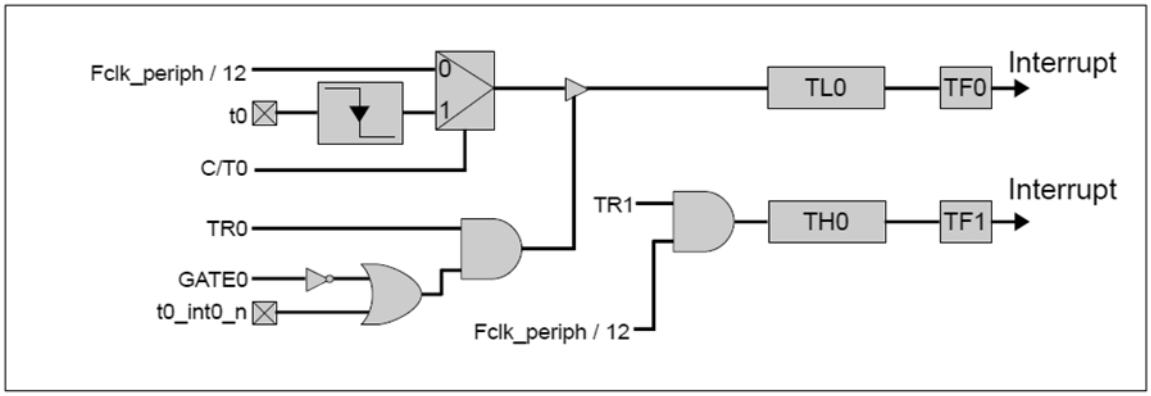
如图 4 所示，模式 2 的计数器控制和模式 0、模式 1 是一样的。但是，在模式 2 中，当 TL*n* 累计至 FFh，保存在 TH*n* 中的值重载至 TL*n*。



Timer0/1-Mode2

模式3 :两个8位计数器

在模式3中，定时器0为两个8位的计数器，此时定时器 1停止计数并且保存它的值。如图5所示，TL0是由定时器0的控制位来控制的8位寄存器。计数器用GATE作为使能端来控制INT_EXT信号接收。



Timer0/1-mode3

TH0 的是一个单独的 8 位计数器。TH0 只能用来计算时钟周期(12 分频)。定时器 1 的控制位和标志位(TR1 和 TF1)用来作为 TH0 的控制位和标志位。

当定时器 0 工作在模式 3 时，定时器 1 的使用受到限制，因为定时器 0 用到了定时器 1 的控制位(TR1)和中断标志位(TF1)。定时器 1 仍然能用来产生波特率，并且定时器 1 在 TL1 和 TH1 寄存器中的值依然有效。

当定时器 0 工作在模式 3 时，通过定时器 1 的模式位来控制定时器 1。要开启定时器 1，需要将定时器 1 设置为模式 0、1 或者 2。要关闭定时器 1，将定时器 1 的模式设置为 3。定时器 1 可以作为定时器(时钟为 clk/12)，但是由于 TR1 和 TF1 被借用，不能产生溢出中断。当定时器 0 工作在模式 3 时，定时器 1 的 GATE 有效。

7.2、定时器 2

TIMER2 模块起定时作用，其内部主要结构为一个 16 位的计数器，通过对输入时钟的计数达到定时的功能，TIMER2 的计数原则为累加计数，当计数器计数到设定值时产生中断；TIMER2 的计数时钟可选则外部 XTAL 时钟或内部 32K RC 时钟；TIMER2 有两种工作模式：单次定时模式和自动重装载模式，无论哪种模式，计时完成均会产生中断。

单次定时模式：

在一次定时完成后硬件会自动拉低 TIMER2_EN，停止计时。

自动重装载定时模式：

硬件将自动重载设置值，TIMER2_EN 继续维持高，重新开始下一次计时；软件通过向寄存器 TIMER2_EN 写 0 操作停止 TIMER2 计数或者中途修改定时周期。

注意：任意配置 TIMER2_SET_H, TIMER2_SET_L 或 TIMER2_CFG 均可清零计数器。

TIMER2_CFG 寄存器 -- 0x93h

位	功能	R/W	复位值
TIMER2_CFG.7-3	保留	aaaaa	00000
TIMER2_CFG.2	TIMER2_CLK_SEL- TIMER2时钟选择位，配置1选择外部XTAL时钟(32768HZ)，配置0选择内部RC时钟(32KHZ)。默认选择内部RC时钟。默认为0。	R/W	0
TIMER2_CFG.1	TIMER2_RLD- TIMER2计数模式选择位，配置1设置为自动重载模式，配置0为手动重载模式；自动重载模式下，如要精确计时，则不能在中断处理函数中配置定时相关寄存器，因为一旦有配置操作，会使计数器清零。默认为0。	R/W	0
TIMER2_CFG.0	TIMER2_EN- TIMER2计数使能选择位，配置1时开启TIMER2计数，配置0时停止TIMER2计数。手动重载模式下，定时完成，发出中断，中断下降沿自动清零该寄存器；自动重载模式下，定时完成，发出中断，使能寄存器保持不变，自动进行下一轮定时。计数过程中操作该位会重新计数。默认为0。	R/W	0

TIMER2_SET_H 寄存器 -- 0x94h

位	功能	R/W	复位值
TIMER2_SET_H.7-0	TIMER2计数值配置寄存器，高8位，计数过程中重新配置该寄存器会重新计数。默认为0x00。	R/W	0x00

TIMER2_SET_L 寄存器 -- 0x95h

位	功能	R/W	复位值
TIMER2_SET_L.7-0	TIMER2计数值配置寄存器，低8位，计数过程中重新配置该寄存器会重新计数。默认为0x00。	R/W	0x00

TIMER2 定时时长公式为：

$T_{TIMER2}=T_{TIMER2\_CLK}*(\{TIMER2\_SET\_H,\ TIMER2\_SET\_L\}+1)$;

注：$T_{TIMER2\_CLK(外部晶振)}=1/32768(S)$

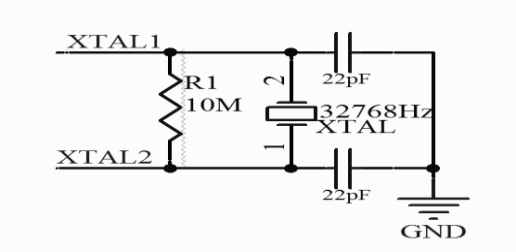
注：$T_{TIMER2\_CLK(内部RC)}=1/32000(S)$

使用外部晶振作为计数时钟配置顺序如下：

1. 设置 Timer1 中断优级为高，根据实际应运设置优先级；

2. 配置 TIMER2_SET_H ，TIMER2_SET_L，配置自动模式还是手动模式；配置 PD_ANA.3 寄存器为 0 打开外部晶振使能，配置 TIMER2_CFG.2 为 1 选择 TIMER2 计数时钟为外部 XTAL；

3. 开启计数。



外部晶振电路参考

PD_ANA 寄存器 -- 0xDBh

位	功能	R/W	复位值
PD_ANA.7	保留	R/W	0
PD_ANA.6	PD_LDO--TK扫描外部参考电压(LDO)关断控制，0：开启，1：关断，默认为1。	R/W	1
PD_ANA.5	PD_LVDT—LVDT功能使能控制，0：开启，1：关闭，默认为1。	R/W	1
PD_ANA.4	PD_BOR—BOR功能使能控制，0：开启，1：关闭，默认为1。	R/W	1

PD_ANA.3	PD_XTAL—外部晶振(32768hz)控制，0：开启，1：关闭，默认为1	R/W	1
PD_ANA.2	PD_OSC32K—内部RC32K时钟控制，0：开启，1：关闭，默认为0。PD_XTAL和PD_OSC32K两者只能有1位是0	R/W	0
PD_ANA.1	PD_TK—TK模块工作控制，0：TK模块工作，1：TK模块不工作。默认为1。	R/W	1
PD_ANA.0	PD_ADC—ADC模块工作控制，0：ADC模块工作，1：ADC模块不工作。默认为1。	R/W	1

定时器 2 中断使能寄存器 IEN2—0xE7h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IEN2	-	-	-	-	-	-	-	IEN2.0
POR	0	0	0	0	0	0	0	0

IEN2.0：定时器2中断使能位；IEN2.0=1：开启定时器2中断；IEN2.0=0：关闭定时器2中断。

定时器 2 中断优先级选择寄存器 IPL2—0xF4h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IPL2	-	-	-	-	-	-	-	IPL2.0
POR	0	0	0	0	0	0	0	0

IPL2.0：定时器2中断优先级选择位；IPL2.0=1：设置定时器2中断高优先级；IPL2.0=0：定时器2中断为低优先级。

定时器 2 中断标志位寄存器 IRCON2—0xE1h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IRCON2	-	-	-	-	-	-	-	IRCON2.0
POR	0	0	0	0	0	0	0	0

IRCON2.0：定时器2中断标志位；定时器2计数到设置值该位置1；中断服务函数中软件清零。

7.3、定时器配置流程

Tiemr0/1 配置流程


.....
关总中断
设置Timer2中断优先级
清除Timer2中断标志位
设置Timer2定时模式
设置Timer2定时初值
选择Tiemr2时钟RC32K或XTAL
使能Timer2中断
开启Timer2定时器
开启总中断
.....

Tiemr2 配置流程

7.4、定时器初始化 C 语言参考

```
//-------------------------------------------------------------------//
//函数名称:  void Timer0Init(uint Timer0Us)
//函数功能:  定时器 0 初始化
//输入参数:  uint Timer0Us:定时 Timer0Us us 时间，Timer0Us 取值范围 1~10000
//输出参数:  无
//返 回 值:  无
//-------------------------------------------------------------------//
#if Timer0_EN

    #pragma message "编译，void Timer0Init(uint Timer0Us)函数"
    void Timer0Init(uint Timer0Us)
    {
        EA_OFF;//关总中断;
        T0_IP_SET;//设置 Timer0 中断优级为高
        T0_INT_FLAG_CLR;//清除 Timer0 中断标志
        T0_CT_MODE(0); //Timer0 定时/计数模式选择 0 为定时模式，1 为计数模式
        T0_MODE_SET(1); //Timer0 定时模式选择:
                //0: 8bit 定时或计数模式,
                //1: 16bit 定时或 10bit 计数模式,
                //2: 8bit 重载定时或计数模式,
                //3:8bit 定时或计数和 8bit 定时模式 tiemr0_clk=(1/12)*sys_clk

        TH0 = ((65536 - 2 * Timer0Us) / 256);
        TL0 = ((65536 - 2 * Timer0Us) % 256);
        TH0_Reload = TH0;
        TL0_Reload = TL0;

        T0_IE_SET;//使能 Timer0 中断
        T0_RUN;//开启 Timer0
        EA_ON;//开总中断
    }
//-------------------------------------------------------------------//
//函数名称:  void Timer0_ISR() interrupt 1
//函数功能:  定时器 0 中断子函数
//输入参数:  无
//输出参数:  无
```

```
//返 回 值:  无
//-------------------------------------------------------------------//
void Timer0_ISR() interrupt 1
{
    T0_INT_FLAG_CLR;//清除 Timer0 中断标志
    TH0 = TH0_Reload;
    TL0 = TL0_Reload;
}
#endif
//-------------------------------------------------------------------//
//函数名称:  void Timer1Init(uint Timer1Us)
//函数功能:  定时器 0 初始化
//输入参数:  uint Timer1Us:定时 Timer1Us us 时间, Timer1Us 取值范围 1~10000
//输出参数:  无
//返 回 值:  无
//-------------------------------------------------------------------//
#if ((Timer1_EN == 1))
#pragma message "编译, void Timer1Init(uint Timer1Us)函数"
void Timer1Init(uint Timer1Us)
{
    EA_OFF;//关总中断;
    T1_IP_SET;//设置 Timer1 中断优级为高
    T1_INT_FLAG_CLR;//清除 Timer1 中断标志
    T1_CT_MODE(0); //Timer1 定时/计数模式选择 0 为定时模式, 1 为计数模式
    T1_MODE_SET(1); //设置为 16 位计算器, tiemr1_clk=(1/12)*sys_clk

    TH1 = ((65536 - 2 * Timer1Us) / 256);
    TL1 = ((65536 - 2 * Timer1Us) % 256);

    TH1_Reload = TH1;
    TL1_Reload = TL1;

    T1_IE_SET;//开 Timer1 中断使能
    T1_RUN;//开启 Timer1
    EA_ON;//开总中断
}
//-------------------------------------------------------------------//
//函数名称:  void Timer1_ISR() interrupt 3
//函数功能:  定时器 1 中断子函数
//输入参数:  无
//输出参数:  无
```

```
//返 回 值:  无
//-------------------------------------------------------------------//
void Timer1_ISR() interrupt 3
{
    T1_INT_FLAG_CLR;//清除 Timer1 中断标志
    TH1 = TH1_Reload;
    TL1 = TL1_Reload;
}
#endif



//-------------------------------------------------------------------//
//函数名称:  void Timer2Init(unsigned int Timer2Ms)
//函数功能:  定时器 2 初始化
//输入参数:  unsigned int Timer2Ms:定时 ims 时间, i 取值范围 1~1999
//输出参数:  无
//返 回 值:  无
//-------------------------------------------------------------------//
#if Timer2_EN

    #pragma message "编译, void Timer2Init(uint Timer2Ms)函数"
    void Timer2Init(unsigned int Timer2Ms)
    {
        unsigned int data dat;

        EA_OFF;//关总中断;

        T2_IP_SET;//设置 Timer2 中断优级为高
        T2_INT_FLAG_CLR;//清除 Timer2 中断标志
        T2_STOP;//bit0 为 0 时停止计时, bit0 为 1 时开启计时
        dat = (Timer2Ms * 32.768)-1;//内部 RC32.7KHz
        TIMER2_SET_H = (dat / 256);//设置 Timer2 定时初值高 8 位
        TIMER2_SET_L = (dat % 256);//设置 Timer2 定时初值低 8 位
        T2_MODE_SET(1);//bit1 为 0 时手动重载模式, bit[1] 1 为时自动重载模式
        #if (TIMER2_CLK_SEL == 0)//0 为内部 RC32K

            T2_XTAL_SET(1);//Timer2 外部晶振开关,1 为关, 0 为开
            RC_32K_SET(0);//WDT、Timer2 32K 时钟开关,1 为关 RC32K, 0 为开 RC32
            T2_CLK_SET(0);//Timer2 定时时钟选择:
                        //0: 内部 RC32.7KHz,
```

```
                    //1：外部晶振 32768Hz
        #else//1 为外部晶振 32768Hz
            T2_XTAL_SET(0);//Timer2 外部晶振开关,1 为关，0 为开
            RC_32K_SET(0);//WDT、Timer2 32K 时钟开关,1 为关 RC32K，0 为开 RC32
            T2_CLK_SET(1);//Timer2 定时时钟选择：
                          //0：内部 RC32.7KHz，
                          //1：外部晶振 32768Hz
        #endif


        T2_IE_SET;//开启 Timer2 中断使能
        T2_RUN;//开启 Timer2
        EA_ON;//开总中断

    }
//-----------------------------------------------------------------//
//函数名称：  void Timer2_ISR() interrupt 15
//函数功能：  定时器 0 中断子函数
//输入参数：  无
//输出参数：  无
//返 回 值：  无
//-----------------------------------------------------------------//
void Timer2_ISR() interrupt 15
{
    T2_INT_FLAG_CLR;//清除 Timer2 中断标志
}
#endif
```

第 8 章、IIC 通信

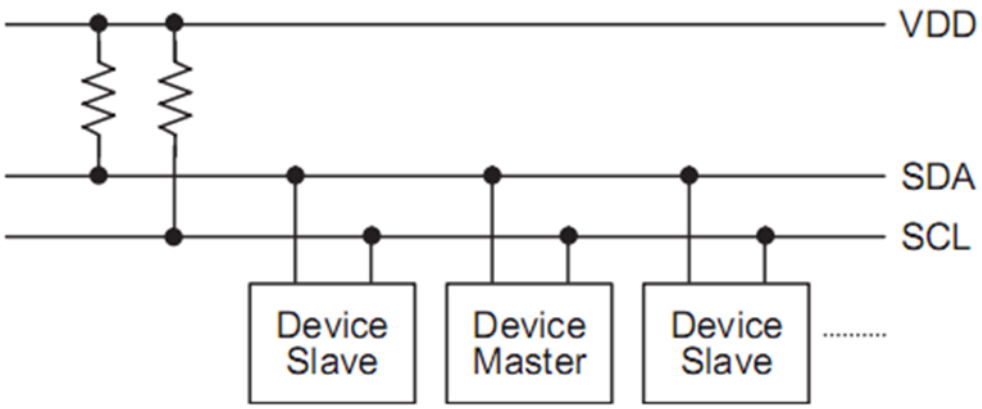
本章节介绍电容传感器的标准通信模式。GDMC181SXXA 支持标准 IIC 通信和快速 IIC 通信。

GDMC181SXXA 中 IIC 通信具有以下特点：

- IIC 从机支持 7 位的寻址方式
- 传输速率：100Kbps、400Kbps
- 中断机制，idle 模式可唤醒芯片
- 延长时钟低电平的功能
- 异常通信情况的诊断

在通信过程中，从机需要支持主机以下操作：

- 从 TS 设备读取数据
- 向 TS 设备发送命令



IIC 主从连接

如图，主机和从机之间由 SCL(串行时钟)线、SDA(串行数据)线连接，SCL、SDA 必须接上拉电阻(建议 4.7K~10K)。当 TS 设备有触摸相关的动作，比如触摸、滑动、手指离开等姿势发生时，主机可通过 IIC 通信来读取从机的触摸状态。

8.1、IIC SFR

IIC 状态寄存器 IICSTAT—0xE8h

位	功能	R/W	复位值
IICSTAT.7	STAT_S-- 开始信号状态位。当检测到开始信号，STAT_S=1。	R	0
IICSTAT.6	STAT_P-- 停止信号状态位。当检测到停止信号，STAT_P=1。	R	1
IICSTAT.5	STAT_RW-- 读写标志位。当主机进行读从机时，STAT_RW=1；当主机写从机时，STAT_RW=0。	R	0
IICSTAT.4	STAT_DA-- 地址数据标志位。当接收或者发送的字节是数据时，STAT_DA=1；当接收或者发送的字节是地址时，STAT_DA=0。	R	0
IICSTAT.3	STAT_BF-- 缓冲器满标志位。当缓冲器为满时，STAT_BF=1；当缓冲器为空时，STAT_BF=0。	R	0
IICSTAT.2	ACKSTAT-- 应答状态位。当主机发送有效应答时，ACKSTAT=0；当主机发送无效应答时，ACKSTAT=1。	R	1
IICSTAT.1	WCOL-- 写冲突标志位。当 IICBUF 满时，对 IICBUF 进行写操作，会造成写冲突，WCOL 置位，数据不能写入 IICBUF。此标志位需要软件清零。	R/W	0
IICSTAT.0	RECOV-- 接收溢出标志位。当 IICBUF 满时，IIC 接收到新的数据，会发生接收溢出，RECOV 置位，同时 IICBUF 里的数据不会被更新。此状态位需要软件清零。	R/W	0

该寄存器位状态详细变化请查看 IIC 从机时序图。

IIC 地址寄存器 IICADD—0xE3h

位	功能	R/W	复位值
IICADD.7-1	IIC地址寄存器，高7位可读可写。	R/W	0x00
IICADD.0	读写控制位。	R	0

IIC 地址寄存器，高 7 位可读可写，最低位为 RW 位。RW 位为 0 时表示主机将数据写入从机，为 1 时表示主机从从机读取数据。

IIC 发送接收缓冲器 IICBUF—0xE4h

位	功能	R/W	复位值
IICBUF.7-0	IIC数据收发缓冲器	R/W	0x00

IIC 数据收发缓冲器，可读可写的寄存器。在主机的同步时钟作用下，数据依次移位发送/接收，高位在前。

IIC 发送数据缓冲器 IICBUFFER—0xE9h

位	功能	R/W	复位值
IICBUFFER.7-0	IIC数据发送缓冲器	R/W	0x00

具体应用过程如下：

在 RD_SCL_EN 为 0 的情况下，主机读取数据时，在产生中断后 2 个 clk 之后将 IICBUFFER 中的数据送到从机发送缓存寄存器中，作为从机发送的数据。故在中断产生之前，IICBUFFER 中的数据要准备好，一般情况下是在上一个中断服务程序中准备好，设备地址产生中断发送数据为初始化中准备。

IIC 控制寄存器 IIC_CON—0xE5h

位	功能	R/W	复位值
IICCON.7-6	保留	R	0
IIC_CON.5	IIC_RST-- IIC 模块复位操作，当该位置 1 时，IIC 模块将发生复位，该位为 0 后，IIC 模块正常工作。	R/W	0
IIC_CON.4	RD_CLR_EN-- 主机读拉低时钟线控制位。1 为使能主机读中断拉低时钟线功能，0 为不使能。	R/W	1
IIC_CON.3	WR_CLR_EN-- 主机写拉低时钟线控制位。1 为使能主机写中断拉低时钟线功能，0 为不使能。	R/W	0
IIC_CON.2	SCLEN-- 时钟使能控制位。SCLEN=0 时钟线被锁定在低电平，SCLEN=1 释放时钟线。	R/W	0
IIC_CON.1	SR-- 转换速率控制位。SR=1 时转换速率控制关闭，端口适应于 100Kbps 的通信，SR=0 时转换速率打开，端口适应于 400Kbps 的通信。	R/W	0
IIC_CON.0	IICEN-- IIC 模块使能信号。只有在 IICEN=1 时，IIC 模块电路才工作。所有的 IIC 控制信号也必须在 IICEN=1 时才起作用。	R/W	0

IIC 中断使能寄存器 IEN1—0xE6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IEN1	-	-	-	-	IEN1.3	-	-	-
POR	0	0	0	0	0	0	0	0

IEN1.3：IIC中断使能位；IEN1.3=1：开启IIC中断；IEN1.3=0：关闭IIC中断。

IIC 中断优先级选择寄存器 IPL1—0xF6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IPL1	-	-	-	-	IPL1.3	-	-	-
POR	0	0	0	0	0	0	0	0

IPL1.3：IIC中断优先级选择位；IPL1.3=1：设置IIC中断高优先级；IPL1.3=0：IIC中断为低优先级。

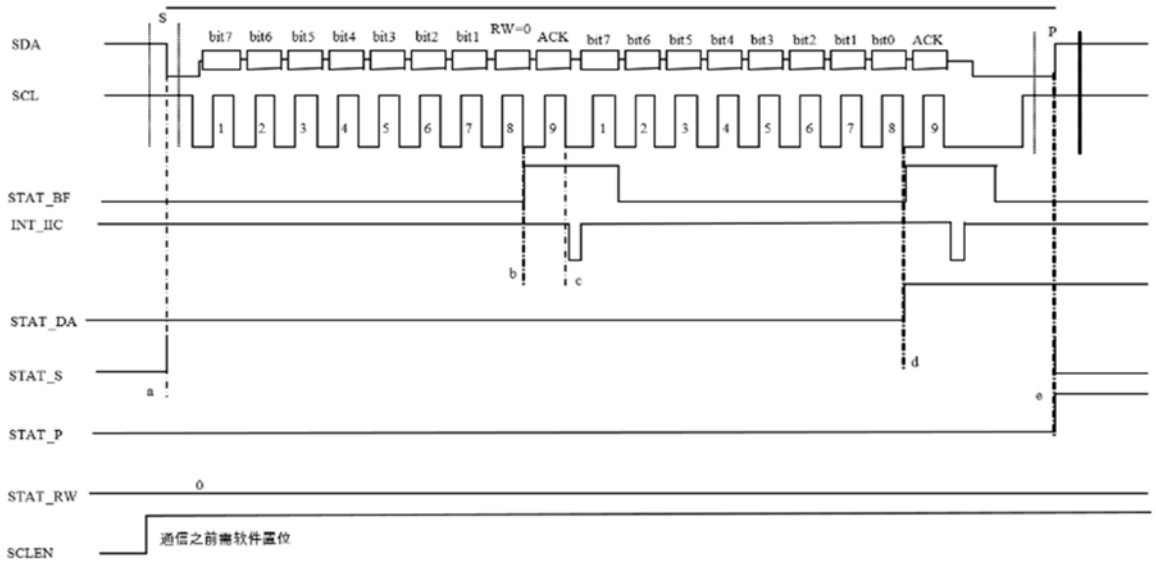
IIC 中断标志位寄存器 IRCON1—0xF1h

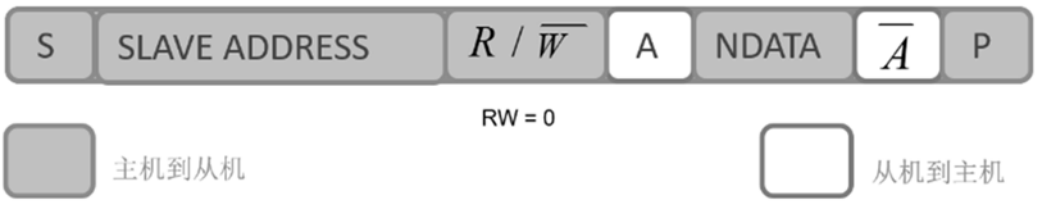
SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IRCON1	–	–	–	–	IRCON1.3	–	–	–
POR	0	0	0	0	0	0	0	0

IRCON1.3：IIC中断标志位；产生IIC中断时该位置1；中断服务函数中软件清零。

8.2、IIC 从机时序图

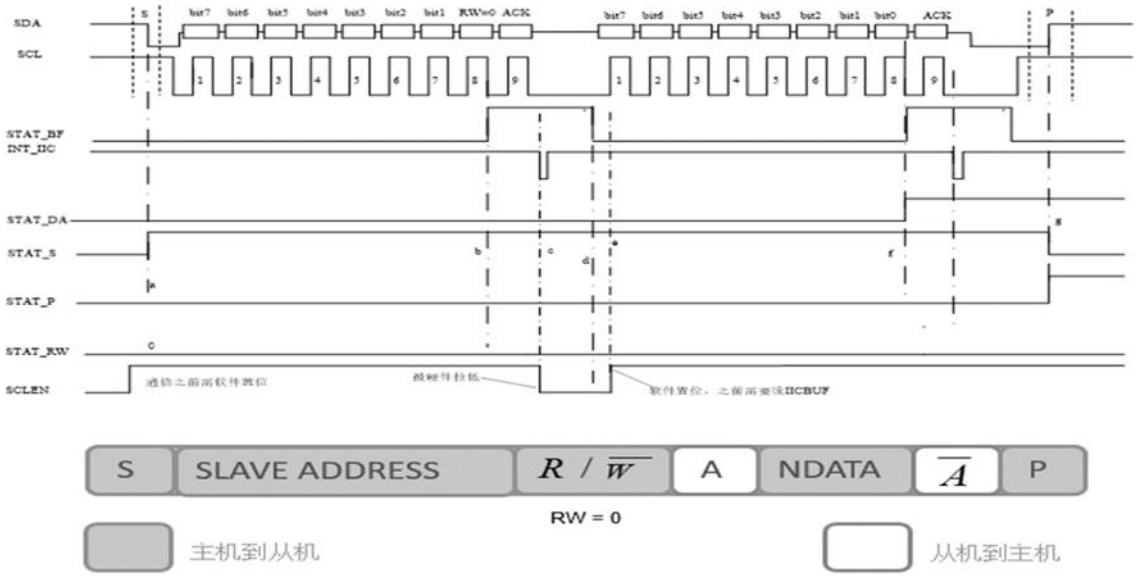
GDMC181SXXA 采用硬件从机。当主机读/写数据时，从机接收到地址后，如果地址匹配，则产生中断，发送有效应答信号。并在主机写数据的第八个时钟后产生中断，主机发送停止信号时将不会产生中断信号。下面是 IIC 通信的简单时序图：





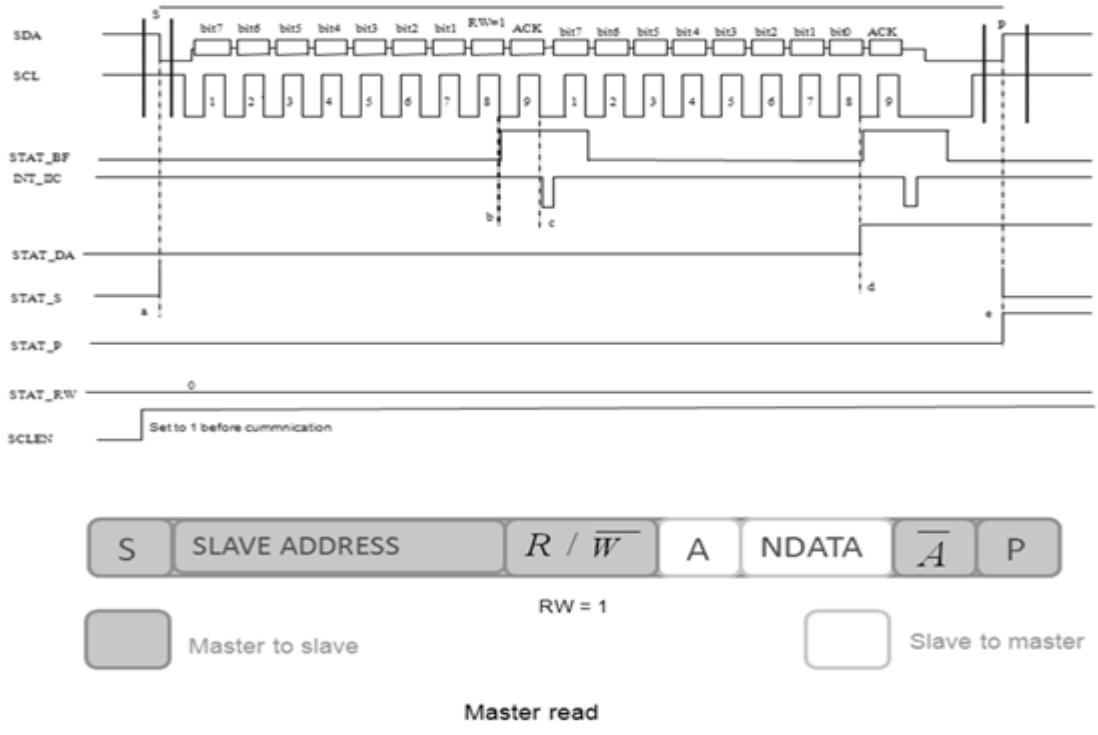
主机写示意图

主机写示意图



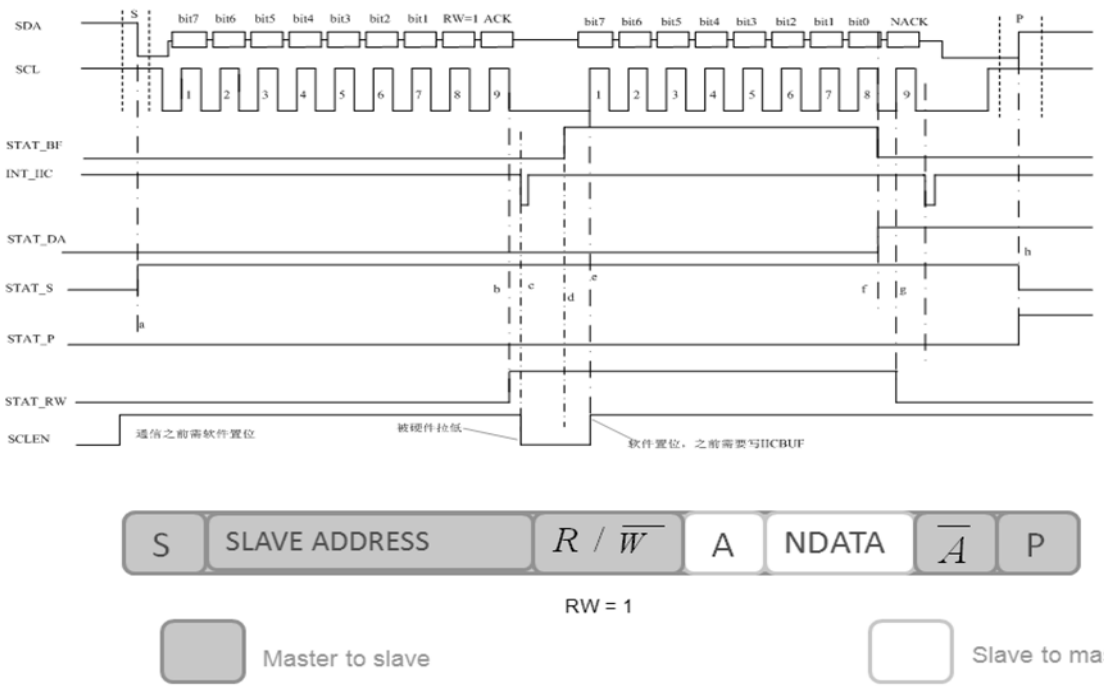
主机“写拉低”示意图

主机“写拉低”示意图

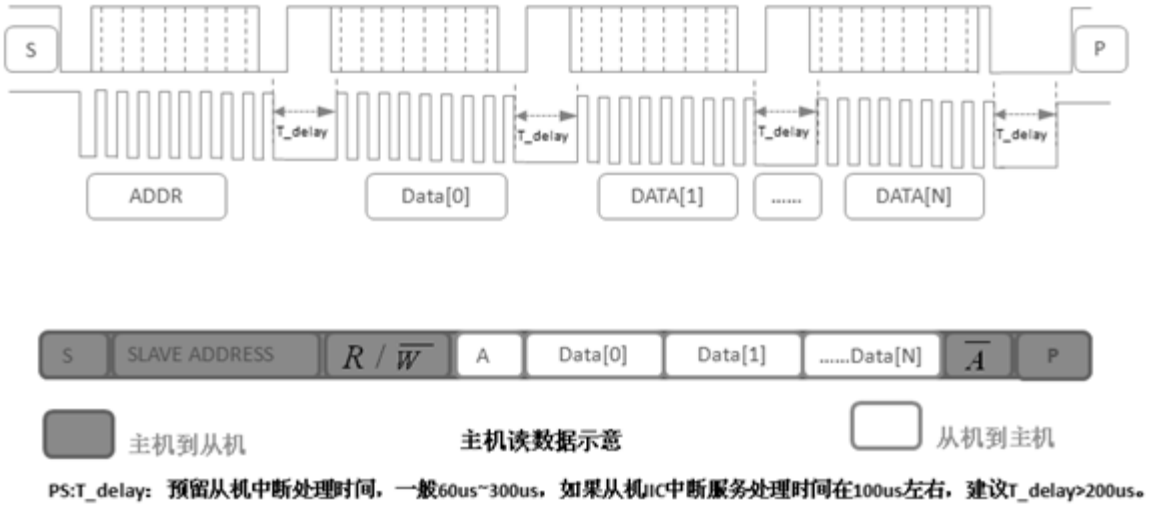


Master read

主机读示意图

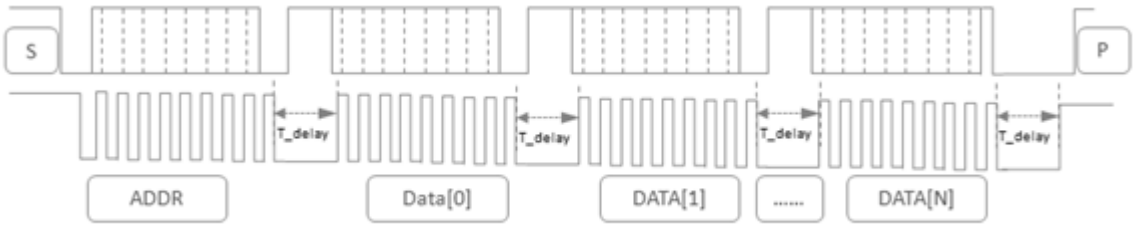


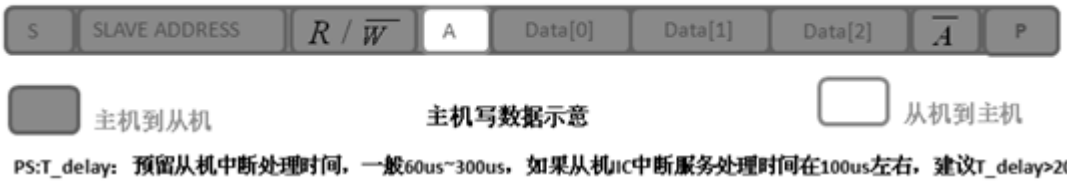
主机“读拉低”示意图



PS:T_delay：预留从机中断处理时间，一般60us~300us，如果从机IIC中断服务处理时间在100us左右，建议T_delay>200us。

从机在第八个时钟的下降沿给出 ACK 信号，第九个时钟的下降沿产生 IIC 中断，建议主机在发完第九个时钟下降沿的时候延时 60us~300us 预留从机 IIC 中断服务数据准备时间，然后再发时钟信号。





PS:T_delay：预留从机中断处理时间，一般60us~300us，如果从机IIC中断服务处理时间在100us左右，建议T_delay>200us。

8.3、IIC 配置流程

IIC 初始化配置流程

8.4、IIC 初始化 C 语言参考

```
//-------------------------------------------------------------------//
//函数名称：  void IIC_SlaveInit(uchar IICSlaveAddr)
//函数功能：  IIC 从机初始化
//输入参数：  uchar IICSlaveAddr:IIC 从机地址
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
void IIC_SlaveInit(uchar IICSlaveAddr)
{
    EA_OFF;//关总中断；
    IIC_IP_SET; //设置 IIC 中断优先级为高；
    IIC_INT_FLAG_CLR;//清除 IIC 中断标志位
    TRISA |= 0x03;//设置 SDA、SCL 为输入；
    IICADD = IICSlaveAddr;//设置 IIC 从机地址；
    IICSTAT = 0x00;//清除 IIC 状态寄存器；
    IICCON = (IIC_EN&(~IIC_SR)|IIC_SCLEN&(~IIC_W_SCL_EN)&(IIC_R_SCL_EN));
    //IIC 工作使能//IIC 转换速率为、400K//SLC 时钟正常工作//不使能写拉低//不使
能读拉低
    IIC_IE_SET; //开 IIC 中断使能
    EA_ON;//开总中断；
}
//-------------------------------------------------------------------//
//函数名称：  void MasterReadData()
//函数功能：  将发送的数据写入 IICBUF
//输入参数：  无
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
void MasterReadData()
{
    uchar tmp;
    tmp = IIC_Send[IIC_Send_Count];
    IICBUF = tmp;
    IIC_Send_Count++;
    SCLEN_SET;
    if(IIC_Send_Count == 2)
```

```
    {
        IIC_Send_Count = 0;
    }
}
//-------------------------------------------------------------------//
//函数名称：  void MasterWriteData()
//函数功能：  从 IICBUF 读取数据
//输入参数：  无
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
void MasterWriteData()
{
    uchar tmp;
    tmp = IICBUF;
}
//-------------------------------------------------------------------//
//函数名称：  void IIC_ISR_PA() interrupt 10
//函数功能：  PA 口 IIC 中断子函数
//输入参数：  无
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
void IIC_ISR_PA() interrupt 10
{
    uchar tmp;
    IIC_INT_FLAG_CLR;//清除 IIC 中断标志位

    if (IICS_WCOL)//写冲突标志位
    {
        IICS_WCOL_CLR;
    }
    if (IICS_RECOV)//读冲突标志位
    {
        IICS_RECOV_CLR;
        tmp = IICBUF;
    }
    if (IICS_AD == 0)//是地址
    {

        if (IICS_RW)//主机读
```

```
            {
                while (SCLEN);

                MasterReadData();
                SCLEN_SET;
            }
            else //主机写
            {
                tmp = IICBUF;
            }
        }
        else
        {
            if (IICS_RW)//主机读
            {
                while (SCLEN);
                MasterReadData();
                SCLEN_SET;
            }
            else//主机写
            {

                if (IICS_BF)
                {
                    MasterWriteData();
                }
            }
        }
    }
```

第 9 章、UART 通信

GDMC181SXXA 提供了一个串口供用户使用，可以选择 3 路 IO 映射，PB3/PB4、PD4/PD5 或 PA0/PA1，默认使用 PA0/PA1。GDMC181SXXA 提供的 UART 通信具有以下特点：

- 支持全双工、半双工通信
- 具有独立的双缓冲接收器和单缓冲发射器
- 可编程波特率(10 位模数分频器)
- 支持串口中断或轮询操作
- 支持发送完成、接收满、接收溢出、奇偶校验错误和帧错误检测
- 支持硬件奇偶校验生产和检查
- 可设置 8 位或 9 位字符长度
- 可设置 1 位或 2 位 STOP 位
- 支持多处理器模式

9.1、UART 相关 SFR

UART_BDL 寄存器—0xDCh

位	功能	R/W	复位值
UART_BDL.7-0	UART_BDL寄存器为串口通信波特率控制器低8位，该寄存器与UART_CON2[1:0]共同构成波特率设置的10位分频数据。	R/W	0x00

UART_CON1 寄存器—0xDDh

位	功能	R/W	复位值
UART_CON1.7	保留	R/W	0
UART_CON1.6	UART_EN-- UART模块使能控制位，只有UART_EN=1时UART电路模块才工作，所有的UART设置只有在UART_EN=1时才有效。	R/W	0
UART_CON1.5	REC_EN-- 接收使能控制位，当该位置1时UART模块的接收器才开打工作，当该位置0时无法接收数据。	R/W	0
UART_CON1.4	MULTI_MODE-- 多处理器模式使能控制位，该位置1时打开多处理器模式，该位置0时关闭多处理器模式，需要设置多处理器模式时需要设置9位数据模式。	R/W	0
UART_CON1.3	STOP_MOD-- 停止位位宽设置，该位置1时选择2位位宽，清0时选择1位位宽。	R/W	0

UART_CON1.2	DATA_MOD-- UART通信数据位长度设置，该位置1时选择9位数据长度模式，该位清0时选择8位数据长度模式；开启奇偶校验功能或多处理器模式时需要选择9位数据长度模式。	R/W	0
UART_CON1.1	PAR_EN-- 奇偶校验功能使能控制位，该位置1时开启奇偶校验功能，清0则关闭奇偶校验功能。	R/W	0
UART_CON1.0	PAR_SE-- 奇偶检验选择位，该位置1选择奇校验，清0则选择偶校验。需要开启奇偶校验使能该功能才有效。	R/W	0

UART_CON2 寄存器—0xDEh

位	功能	R/W	复位值
UART_CON2.7-4	保留	aaaa	0000
UART_CON2.3	TX_IE_EN-- UART发送中断使能，该位置1时使能发送中断，清0时关闭发送中断。	R/W	1
UART_CON2.2	RX_IE_EN-- UART接收中断使能，该位置1时使能接收中断，清0时关闭接收中断。	R/W	1
UART_CON2.1	UART_BDH1-- UART通信波特率设置第10位，需要和UART_BDL寄存器共同构成10位分频数据。	R/W	0
UART_CON2.0	UART_BDH0-- UART通信波特率设置第9位，需要和UART_BDL寄存器共同构成10位分频数据。	R/W	0

UART_STATE 寄存器—0xDFh

位	功能	R/W	复位值
UART_STATE.7	保留	R	0
UART_STATE.6	R8--接收器接收的第9个数据。	R	0
UART_STATE.5	T8-- 发射器的第9个数据。	R/W	0
UART_STATE.4	TX_EMPTY_IF—发送中断标志位；1：发送缓存为空；0：发送缓存已满。该位需要软件清零。	R/W	0
UART_STATE.3	RX_FULL_IF—接收中断标志位；1：接收缓存已满；0：接收缓存为空。该位需要软件清零。	R/W	0
UART_STATE.2	RX_OVERFLOW_IF—接收溢出标志；1：数据溢出(数据丢失)；0：数据未溢出。该位需要软件清零。	R/W	0
UART_STATE.1	FRAME_ERR_IF—帧错误标志；1：检测到帧错误；0：未检测到帧错。该位需要软件清零。	R/W	0
UART_STATE.0	PARITY_ERR_IF—奇偶校验错误；1：接收器奇偶校验错误；0：奇偶校验正确。该位需要软件清零。	R/W	0

UART_BUF 寄存器—0xE2h

位	功能	R/W	复位值
UART_BUF.7-0	UART通信数据寄存器，该寄存器实际由两部分组成：一个发送缓冲器和一个接收缓冲器。当数据写入该寄存器时，它进入移位发送缓冲器等待串行发送。向该寄存器写入一个字节的数据即启动一次发送过程。读取该寄存器时数据来自接收缓冲器。	R/W	0xFF

9.2、UART 波特率配置

波特率生产模数：bandrate={UART_BDH [1:0],UART_BDL [7:0]}，即 UART_CON2 寄存器的 bit1 和 bit0 与 UART_BDL 寄存器的 8 位组成的 10 位分频数据。

当 UART_CON2 寄存器的 bit1 和 bit0 均为 0 且 UART_BDL 为 0x00，即 bandrate=0 时，不产生波特率时钟。当 bandrate=1~1023 时串口波特率时钟公式如下：

串口波特率(Buad rate)=SYS_CLK/(16*bandrate)

SYS_CLK 是系统工作时钟，GDMC181SXXA 芯片的系统工作时钟根据设置可能为(24Mhz、12Mhz、6Mhz 和 1.5Mhz)。

波特率和系统时钟对照表如下：

波特率		24Mhz 系统时钟			12Mhz 系统时钟		
序号	波特率	实际	bandrate 值	误差%	实际	bandrate 值	误差%
1	115200	115384	13	0.16%	不支持	-	-
2	57600	57692	26	0.16%	57692	13	0.16%
3	38400	38461	39	0.16%	37500	20	2.34%
4	19200	19230	78	0.16%	19230	39	0.16%
5	9600	9615	156	0.16%	9615	78	0.16%
6	4800	4807	312	0.16%	4807	156	0.16%
7	2400	2400	625	0	2403	312	0.16%
波特率		6Mhz 系统时钟			1.5Mhz 系统时钟		
序号	波特率	实际	bandrate 值	误差%	实际	bandrate 值	误差%
1	115200	不支持	-	-	不支持	-	-
2	57600	不支持	-	-	不支持	-	-
3	38400	37500	10	2.34%	不支持	-	-
4	19200	18750	20	2.34%	18750	5	2.34%
5	9600	9615	39	0.16%	9375	10	2.34%
6	4800	4807	78	0.16%	4934	19	2.79%
7	2400	2403	156	0.16%	2403	39	0.16%

9.3、多处理器模式

当MULTI_MODE（UART_CON1.4）被置位，通过使用第九数据位和软件识别支持一个主处理器与一个或多个从处理器之间的多机通信。在多处理器模式下，接收到的第九位数据被保存到R8(UART_STATE.6)，当接收到停止位后，串口中断只有在R8=1时才被激活。

一个典型的多机通信应用就是当主机想要向一个或多个从机发送数据数据时，它先发送一个用于选择目标从机的地址字节。地址字节与数据字节的区别是：地址字节的第九位为逻辑1；数据字节的第九位总是设置为逻辑0。

当MULTI_MODE（UART_CON1.4）=1时，主机发送数据字节，从机将不产生中断。但是主机发送地址字节，所有的从机都将产生一个中断，从机在这个中断服务程序中判断本机是否被寻址，从机的地址分配由软件实现。被寻址的从机清除MULTI_MODE（UART_CON1.4）位，等待接收数据。而未被寻址的从机将保持MULTI_MODE（UART_CON1.4）为1以忽略主机发送的数据字节。

9.4、UART 配置流程

Uart0 配置流程

9.5、UART 初始化 C 语言参考

```
//-------------------------------------------------------------------//
//函数名称：  void UART0_Init(void)
//函数功能：  UART0 初始化
//输入参数：  bit enableInterrpt:中断使能 0 为关串口中断，1 为开串口中断;
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
void UART0_Init(void)
{
    EA_OFF;//关总中断;
     UART_IP_SET;//设置 UART0 中断优级为高
    UART_INT_FLAG_CLR;//清除 UART0 中断标志
    PERIPH_IO_SEL |= UART_PORT;
    //UART_PA(RXD0-TXD0)映射为 PA0/1 为串口，
    //UART_PB(RXD0-TXD0)射为 PB3/4 为串口，
    //UART_PD(RXD0-TXD0)射为 PD4/5 为串口。

    #if (UART_PORT == UART_PA)
        TRISA &= ~0x02;//设置 TXD0 为输出
        TRISA |= 0x01;//设置 RX0 为输入
    #elif(UART_PORT == UART_PB)
        TRISB &= ~0x10;//设置 TXD0 为输出
        TRISB |= 0x08;//设置 RX0 为输入
    #elif(UART_PORT == UART_PD)
        TRISD &= ~0x20;//设置 TXD0 为输出
        TRISD |= 0x10;//设置 RX0 为输入
    #else

    #endif

    UART_BDL = (uchar)M_HEX_OF_BR(CFG_BAUD_RATE);
    //设置波特率=BUSCLK/(16*(UART_BDH[0:1],UART_BDL))
    UART_CON2 =
(((((uint)(M_HEX_OF_BR(CFG_BAUD_RATE)))>>8)&0x03)|(UART_RX_EN|UART_TX_EN));
    //设置 UART_BDH[0:1]高两位//接收中断使能//发送中断使能
    UART_CON1 = (UART_EN|UART_MULTI|UART_RE_EN)&(~UART_SEND_MODE);
```

```
    //使能 UAERT 模块//使能多处理器通信//开启接收使能 //发送数据模式
bit[3]：0：八位模式：1：9 位模式模式
    UART_STATE &= ((~UART_RI)&(~UART_TI));//清除接收中断标志位
    //清除发送中断标志位
    UART_IE_SET;//开 UART 中断使能
    EA_ON;//开总中断
}


void UART0_SendWord(uint willSendWord)
{
    UART_STATE &= ~UART_TI;//清除发送中断标志位

     UART_BUF = willSendWord>>8;

     while (!(UART_STATE & UART_TI)); // UART_TI 为宏定义
     UART_STATE &= ~UART_TI;//清除发送中断标志位

     UART_BUF = (uchar)willSendWord;

    while (!(UART_STATE & UART_TI)); // UART_TI 为宏定义
     UART_STATE &= ~UART_TI;//清除发送中断标志位
}


void UART0_SendByte(uchar willSendByte)
{
     UART_STATE &= ~UART_TI;//清除发送中断标志位
     UART_BUF = willSendByte;

    while (!(UART_STATE & UART_TI)); // UART_TI 为宏定义
     UART_STATE &= ~UART_TI;//清除发送中断标志位
}


/* 串口 0 中断服务子程序 */
void UART0_ISR() interrupt 17
{
     unsigned char temp;
    UART_INT_FLAG_CLR;//清除 UART0 中断标志
     if ((UART_STATE & UART_RI))
```

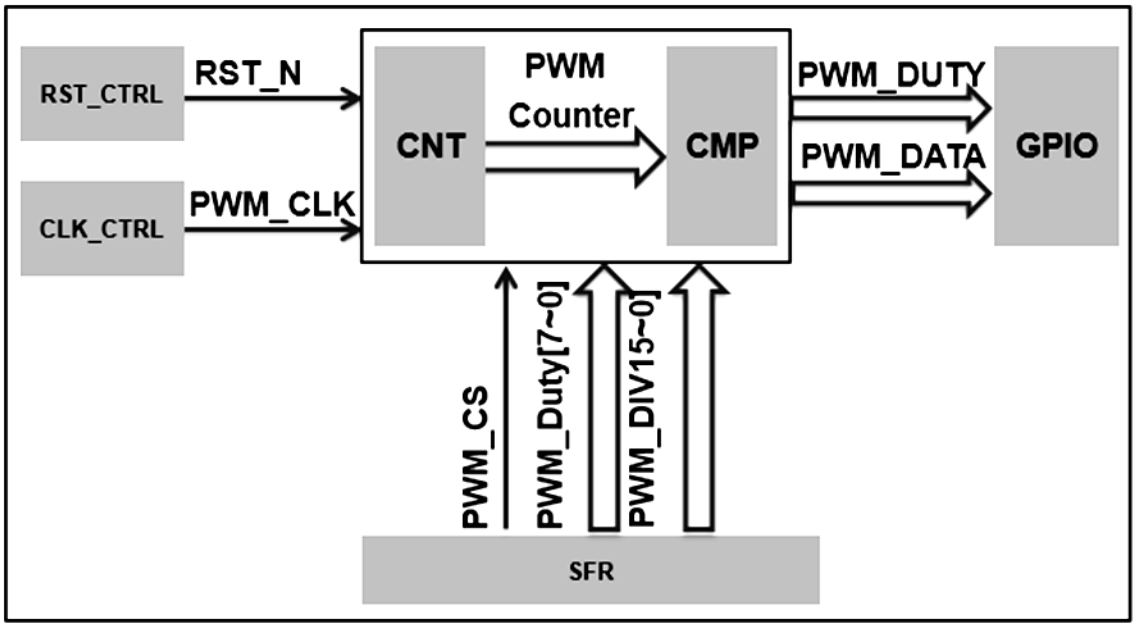
```
{
     temp = UART_BUF;
     UART_STATE &= ~UART_RI;//清除接收中断标志位
}
if((UART_STATE & UART_TI))
{
    UART_STATE &= ~UART_TI;//清除发送中断标志位
}
}
```

注：串口接收数据为 8bit、无校验、1 位 Stop 位模式时，若主机连续发送数据，建议主机发送完 Stop 位后，至少延时两个波特率时间 T=(2/Buad rate)s。

第 10 章、PWM

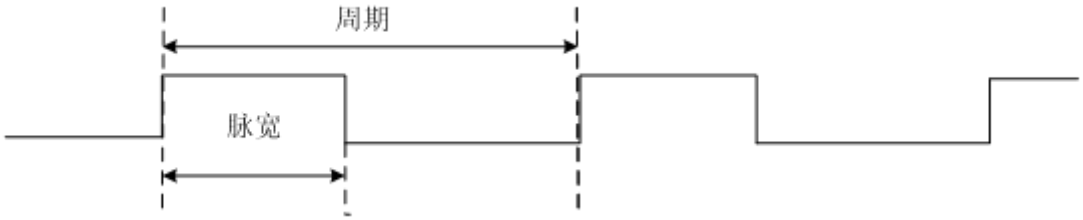
10.1、PWM 模块及 PWM 寄存器描述

PWM 脉宽调制模块输出包含一个时基（周期）和一段输出高电平的时间（脉宽）。PWM 脉宽调制模块的周期和脉宽都可以通过编程进行配置。PWM 的周期控制寄存器为 6 位寄存器，占空比控制寄存器为 8 位寄存器。PWM 相关寄存器有 PWM_DIV、PWM_DUTY、PWM_CS 和 PERIPH_IO_SEL 共 PWM 结构框图如下：



PWM 结构框图

PWM 输出波形如下图所示：



IO 映射寄存器 PERIPH_IO_SEL—0xADh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PERIPH_IO_SEL	-	-	-	PWM_IO_SEL[1:0]		-	-	-
POR	-	0	0	0	0	0	0	0

PWM_IO_SEL：PWM 功能使能，PWM_IO_SEL[1]对应 PB0 管脚；PWM_IO_SEL[0]对应 PD1 管脚；

01：对应 PD1 开启 PWM 功能；

10：对应 PB0 开启 PWM 功能；

11：对应 PD1 开启 PWM 功能；

00：PB1 和 PD1 关闭 PWM 功能。

PWM 周期与模式寄存器 PWM_DIV—0x80h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM_DIV	-	PWM_MODE	PWM_DIV					
POR	-	0	0	0	0	0	0	0

PWM_MODE：PWM 模式选择，1：8 倍预分频输出模式；0：512 倍预分频输出模式

PWM_DIV：PWM 周期设置寄存器，PWM 的 2 种模式周期计算公式如下：

8 倍预分频输出模式 PWM 周期计算公式：

PWM 周期 T= 8*（PWM_DIV +64）*(1/PLL_24MHz)

512 倍预分频输出模式 PWM 周期计算公式：

PWM 周期 T= 512*（PWM_DIV +1）*(1/ PLL_24MHz)

PWM 占空比设置寄存器 PWM_DUTY—0x8Eh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM_DUTY	PWM_DUTY[7:0]							
POR	0	0	0	0	0	0	0	0

PWM 的 2 种模式占空比计算公式如下：

8 倍预分频输出模式 PWM 占空比计算公式：

占空比 D = PWM_DUTY[2:0]/8

当 PWM_DUTY[2:0] = 0x00 时，PWM 输出占空比为 0%；当 PWM_DUTY[2:0] = 0x03 时，输出占空比为 100%。

512 倍预分频输出模式 PWM 占空比计算公式：

占空比 D = PWM_DUTY[7:0]/256

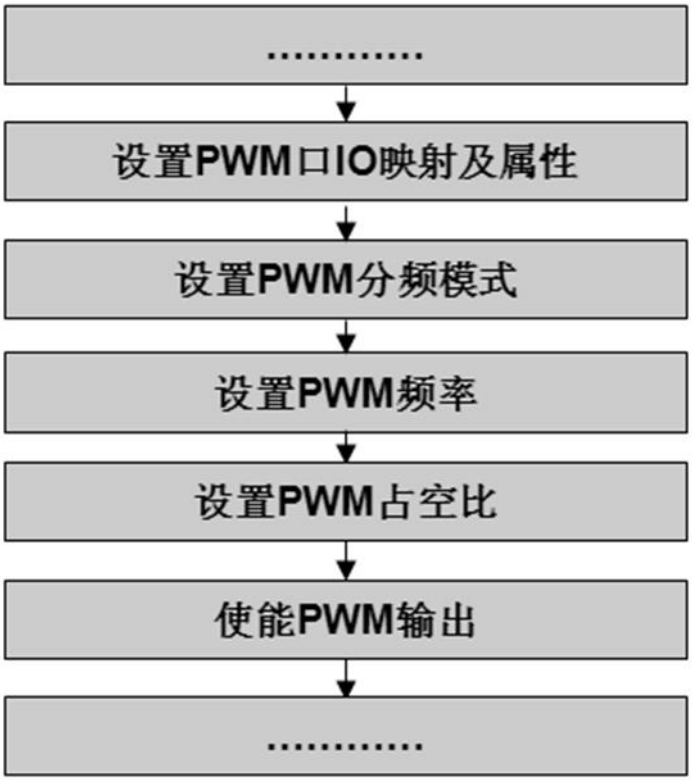
当 PWM_DUTY[7:0] = 0x00 时，PWM 输出占空比为 0%；当 PWM_DUTY[7:0] = 0xFF 时，输出占空比为 100%

PWM 模块使能寄存器 PWM_CS—0x90h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM_CS	-	-	-	-	-	-	-	PWM_CS
POR	-	-	-	-	-	-	-	0

PWM_CS：1：使能 PWM 模块功能；0：不使能 PWM 模块功能。

10.2、PWM 配置流程



PWM 模块配置流程

10.3、PWM 初始化 C 语言参考

```
//函数名：void PWMInit(bit pwm_mode,bit pwm_io_sel,uchar pwm_div, uchar pwm_duty)
//功　能：初始化 PWM 标准程序
//参　数：PWM_IO,pwm_div 频率，pwm_duty 占空比
//返回值：无
//PWM_MODE_SEL = 1 时,PWM 周期 = 8*(PWM_DIV+64)*TMCU(us),Duty_PWM = PWM_DUTY[2:0]/8
//PWM_MODE_SEL = 0 时，PWM 周期 = 512*PWM_DIV*TMCU(us),Duty_PWM = PWM_DUTY[7:0]/256
//PWM_MODE_SEL = 1 时，[(0x0F,0x04)--(38K,50%)](PWM_DUTY:0-0%,1-12.5%,
//2-25%,3-37.5%,4-50%,5-62.5%,6-75%,7-100%)
//PWM_MODE_SEL = 0 时,[(0x2F,0x80)--(1K,50%)]-[(0x17,0x80)--(2.03K,50%)]-
//[(0x10,0x80)--(2.92K,50%)]-[(0x0A,0x80)--(4.68K,50%)]
//[(0x09,0x80)--(5.20K,50%)]
    void PWMInit(uchar pwm_div, uchar pwm_duty)
    {
        #if(PWM_PORT == PWM_PD1)
            PERIPH_IO_SEL |= 0x08;//选择 PD1 为 PWM 功能
            SET_PD1_IO_IN;//设置 PD1 为入 IO 输入
        #elif(PWM_PORT == PWM_PB0)
            PERIPH_IO_SEL |= 0x10;//选择 PB0 为 PWM 功能
            SET_PB0_IO_IN;//设置 PB0 为入 IO 输入
        #else

        #endif
        #if(PWM_MODE_SEL == PWM_MODE0)
            PWM_DIV &= ~0x40;//512 倍预分频模式
            PWM_DIV = pwm_div;//设置 PWM 频率,设置 pwm_duty 占空比
            PWM_DUTY = pwm_duty;
            PWM_CS |= 0x01;//使能 PWM 输出
            #elif(PWM_MODE_SEL == PWM_MODE1)
                PWM_DIV |= 0x40;//8 倍预分频模式
                PWM_DIV = pwm_div;//设置 PWM 频率,设置 pwm_duty 占空比
                PWM_DUTY = pwm_duty;
                PWM_CS |= 0x01;//使能 PWM 输出
            #else

            #endif
    }
```

第 11 章、CTK

GDMC181SXXA通过一系列的寄存器来实现多种功能的应用。电容检测相关量与SFR值的关系如下：

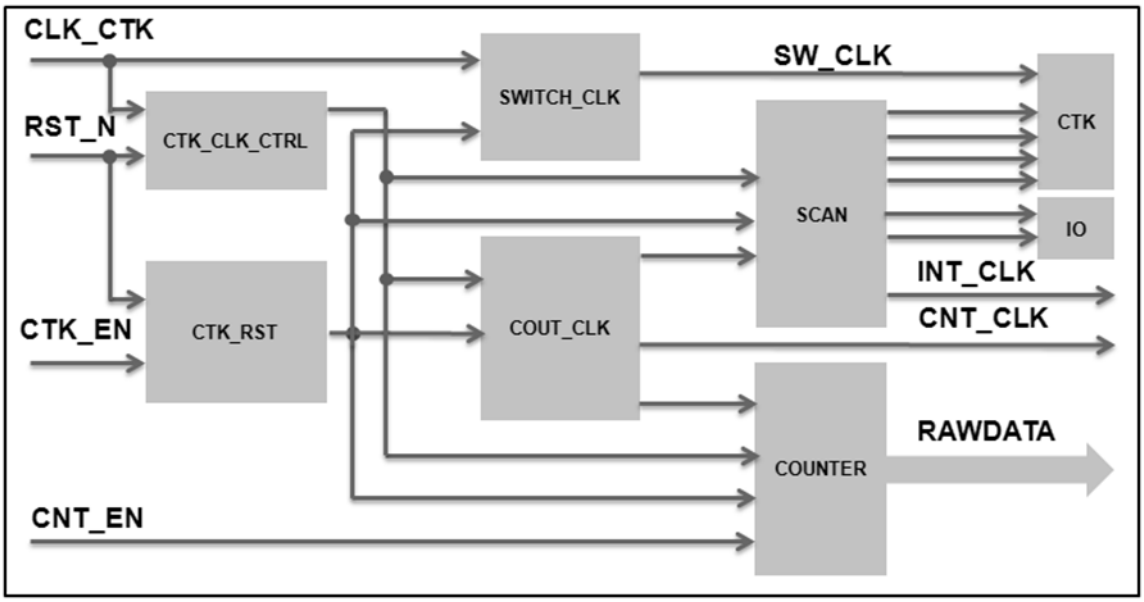
计数值大小与RESO、Rb电阻、PULL_I_SELA_H成正比,与VTH_SEL成反比。在保证充放电完全的情况下，与通过PRS_DIV设定的充放电频率成正比。

通道触摸变化量与RESO、Rb成正比，与VTH_SEL成反比。在保证充放电完全的情况下，与通过PRS_DIV设定的充放电频率与触摸变化量成正比。

触摸检测的信噪比与VTH_SEL成正比，PULL_I_SELA_L,与CSD_DS成反比。在充放电不完全时，与通过PRS_DIV设定的充放电频率与信噪比成反比。

单个按键检测的时间与RESO、与CSD_DS有关。

注：配置参数时应保证按键充放电完全。



CTK模块框图

11.1、CTK 相关寄存器描述

CTK 扫描功能选择寄存器 1　SNS_IO_SELL—0xA3h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SNS_IO_SELL	SEL_SENSOR[7:1]							-
POR	0	0	0	0	0	0	0	-

SNS_IO_SELL：对应的 bit 位为 1，IO 口选择为 sensor 功能，对应的 bit 位为 0，IO 口选择为 GPIO 功能。配置时应注意任意 IO 口同一时间只能使能其中一个功能。

CTK 扫描功能选择寄存器 2　SNS_IO_SELH—0xA6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SNS_IO_SELH	SEL_SENSOR[15:8]							
POR	0	0	0	0	0	0	0	0

SNS_IO_SELH：对应的 bit 位为 1，IO 口选择为 sensor 功能，对应的 bit 位为 0，IO 口选择为 GPIO 功能。配置时应注意任意 IO 口同一时间只能使能其中一个功能。

CTK 扫描功能选择寄存器 3　SNS_IO_SELO—0xECh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SNS_IO_SELO	-					SEL_SENSOR [18]	SEL_SENSOR [17]	SEL_SENSOR [16]
POR	-					0	0	0

SNS_IO_SELO：对应的 bit 位为 1，IO 口选择为 sensor 功能，对应的 bit 位为 0，IO 口选择为 GPIO 功能。配置时应注意任意 IO 口同一时间只能使能其中一个功能。

按键对应关系如下表所示：

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SNS_IO_SELL	SNS7	SNS6	SNS5	SNS4	SNS3	SNS2	SNS1	-
SNS_IO_SELH	SNS15	SNS14	SNS13	SNS12	SNS11	SNS10	SNS9	SNS8
SNS_IO_SELO	-					SNS18	SNS17	SNS16

功能控制寄存器　PD_ANA—0xDBh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PD_ANA	-	-	-	-	-	-	PD_CSD	-
POR	-	1	1	1	1	0	1	1

PD_CSD：CTK 模块使能控制位，PD_CSD=0 时 CTK 模块工作；PD_CSD=1 时 CTK 模块不工作。

CTK 扫描参数设置寄存器 1 SNS_SCAN_CFG1—0xCEh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SNS_SCAN_CFG1	SYNCH	RESO[2:0]			PRS_EN	PRS_DIV[2:0]		
POR	0	1	1	1	0	0	0	1

SYNCH：SNS通道IO同步控制寄存器；1：同步；0：非同步。

RESO：CTK计数位数选择寄存器：

000：9位；001：10位；001：11位；011：12位；

100：13位；101：14位；110：15位；111：16位。

PRS_EN：CTK扫描前端充放电控制寄存器；

1：关闭前端充放电使能；0：开启前端充放电。

PRS_DIV：CTK扫描前端充放电频率控制寄存器；

000：400KHz；001：1MHz；010：2MHz

011：3MHz；100：4MHz；101：6MHz；

110：最高频率3MHz，最低频率1MHz,中心频率1.5MHz，正态分布；

111：最高频率3MHz，最低频率1MHz,中心频率1.5MHz，均匀分布；

根据不同的应用，配置合适的CTK充放电频率，合适的PRS_DIV配置可以提高SNR。当处于嘈杂的环境时，保证按键充放电完全，PRS_DIV值增大，SNR会有所提高。为了保持采样波形不失真，建议$f_{CSD_DS} \geqq 2f_{PRS_DIV}$。

CTK 扫描参数设置寄存器 2 SNS_SCAN_CFG2—0xCFh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SNS_SCAN_CFG2	CSD_DS		PARALLEL_EN	CSD_ADDR				
POR	0	0	0	0	0	0	0	0

CSD_DS：CTK扫描计数频率选择寄存器；

00：24MHz；01：12Mhz；10：6MHz；11：4MHz。

PARALLEL_EN：CTK扫描SNS通道并联使能寄存器；

1：多通道并联模式；

0：单通道模式。

多通道并联模式时SNS_IO_SEL寄存器置1的通道均同时扫描；主要实现任意按键唤醒功能。

CSD_ADDR：CTK扫描通道地址寄存器；对应的通道号可设置1~16。

CTK 扫描模拟参数设置寄存器 SNS_ANA_CFG—0xD6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SNS_ANA_CFG	-	RB_SEL[2:0]			VTH_LDO5_SEL	-	VTH_SEL	
POR	-	0	0	0	0	-	1	0

RB_SEL：CTK扫描Rb电阻大小选择寄存器；

000：20K；001：40K；010：60K；011：80K；

100：150K；101：200K；110：250K；111：300K

VTH_LDO5_SEL：CTK扫描内部参考电压VTH和外部参考电压VTH选择寄存器；

1：选择外部VTH；

0：选择内部参考VTH。

VTH_SEL：CTK扫描内部参考电压VTH大小选择寄存器；

00：1.89V；01：2.22V；

10：2.56V；11：2.73V。

CTK 扫描电流源步进寄存器 PULL_I_SELA_H—0xD4h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PULL_I_SELA_H	-					PULL_I_SELA_H[2:0]		
POR	0	0	0	0	0	0	0	0

PULL_I_SELA_H：CTK 扫描电流源步进大小选择寄存器；

000：0.5uA；001：0.667uA；

010：1uA；011：2uA；100：0.75uA；

101：1uA；110：1.5uA；111：3uA。

CTK 扫描电流源级数大小设置寄存器 PULL_I_SELA_L—0xD5h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PULL_I_SELA_L	PULL_I_SELA_L[7:0]							
POR	0	0	0	0	0	0	0	0

PULL_I_SELA_L：CTK 扫描电流源大小设置寄存器,等于 0xFF 时关闭电流。

电流源大小=(255 - PULL_I_SELA_L)*$I_{PULL\_I\_SELA\_H}$，上拉电流源越小，计数值越小。

CTK 瞬时计数值寄存器(RAWDATAL—0xD2h、RAWDATAH—0xD3h)

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RAWDATAL	RAWDATA<7:0>							
POR	0x00							
RAWDATAH	RAWDATA<15:8>							
POR	0x00							

该寄存器保存当前检测通道的瞬时计数值。默认值：0x00

CTK 扫描使能寄存器 CSD_START—0xD1h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CSD_START	-							CSD_START
POR	-	-	-	-	-	-	-	0

CSD_START： CTK扫描使能。CSD_START=1在开启CTK扫描，在一次扫描结束后，硬件置0；若要开启下一次扫描，则需要软件重新置1。CTK扫描过程中如CSD_START清0，则此次扫

描立即结束。

CTK 扫描中断使能寄存器 IEN1—0xE6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IEN1	-	-	IEN1.5	-	-	-	-	-
POR	0	0	0	0	0	0	0	0

IEN1.5：CTK扫描中断使能位；IEN1.5=1：开启CTK扫描中断；IEN1.5=0：关CTK扫描中断。

CTK 扫描中断优先级选择寄存器 IPL1—0xF6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IPL1	-	-	IPL1.5	-	-	-	-	-
POR	0	0	0	0	0	0	0	0

IPL1.5：CTK扫描中断优先级选择位；IPL1.5=1：CTK中断高优先级；IPL1.5=0：CTK中断低优先级。

CTK 扫描中断标志位寄存器 IRCON1—0xF1h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IRCON1	-	-	IRCON1.5	-	-	-	-	-
POR	0	0	0	0	0	0	0	0

IRCON1.5：CTK扫描中断标志位；CTK扫描完成该位置1；中断服务函数中软件清零。

11.2、CTK 参数设置

通过灵敏度参数配置得到较好信噪比的一组参数，从而提高按键判断的准确性。

1、**RESO**：0~7 CTK 电容扫描分辨率，计数器位数：**（RESO + 9)位**，CTK 电容扫描分辨率越大， Rawdata 向下变化量越大，同时引入的噪声也会随之增大，反之相反。

2、**VTH_SEL**：0~7，参考电压越小，Rawdata 的变化量越大，同时引入的噪声也会随之增大，反之相反。

3、**CSD_DS**：检测速度 **0:24M,1:12M,2:6M,3:4M**，检测速度越小，采样 Rawdata 的时间越慢，反之相反。建议默认 24M 最快，检测速度至少为 2 倍的 PRS 时钟。

4、**RB_SEL**：Rb 电阻选择：**0:20K,1:40K,2:60K,3:80K,4:150K,5:200K,6:250K,7:300K**;电阻越大，Rawdata 的变化量越大，同时引入的噪声也会随之增大，反之相反。

5、**PRS_DIV**：设置 PRS 时钟 **400K,1M,2M,3M,4M,6M**,PRS 时钟越大，Rawdata 的变化量越大，同时引入的噪声也会增大，反之相反。

6、**PULL_I_SELA_L**:配置通道的电流源。当PULL_I_SELA_L为0xFF时，电流源关闭。电流电流源大小=(255 - PULL_I_SELA_L)*$I_{PULL\_I\_SELA\_H}$，电流源越小，计数值越小。默认值：0x00。

7、**PULL_I_SELA_H**：配置通道的电流源步进值。默认值：0x00。

注：

a) **Rawdata 为 CTK 电容计数器的实时原始计数值。**

b) **实际应用中需要通过烧录调试软件查看数据并进行参数对比得到信噪比较好的一组参数。**

c) **芯片供电电压与参考电压关系：VCC-VTH>0.8V。**

11.3、CTK 配置流程

CTK按键扫描为中断方式，首先，配置CTK参数；然后，开启CTK扫描；最后，在CTK中断获取并保存CTK数据，软件算法进行数据的处理及按键的输出判断。单个按键扫描时间= 776t+(2^N/f_{CSD_DS})。其中t为系统时钟周期，N为扫描分辨率，当芯片系统时钟为24MHz时t=(1/24)us；f_{CSD_DS}即通过CSD_DS寄存器设置的计数频率。

CTK配置流程

11.4、CTK 初始化 C 语言参考

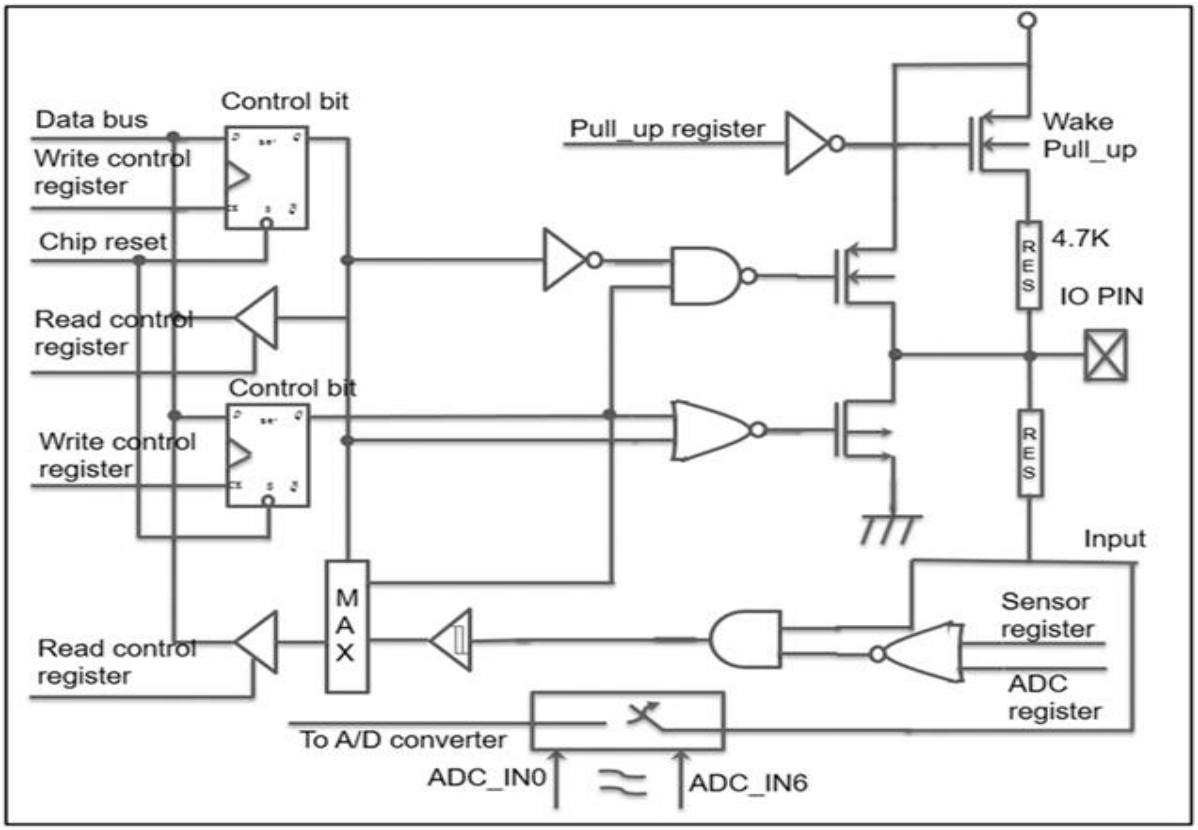
```
//-------------------------------------------------------------------//
//函数名称： void CTK__Init(void)
//函数功能： CTK 初始化，配置所有通道参数，并保存基线
//输入参数： 无
//输出参数： 无
//返 回 值： 无
//-------------------------------------------------------------------//
void CTK _Init(void)
{
        SNS_IO_SELL |= ((CFG_KEYS_MASK) & 0xFF);
        SNS_IO_SELH |= ((CFG_KEYS_MASK>>8) & 0xFF);
        SNS_IO_SELO |= ((CFG_KEYS_MASK>>16) & 0x07);//配置 IO 与 SNS 功能

        TRISB |= ((CFG_KEYS_MASK&0x01)<<7);
        TRISC |= ((CFG_KEYS_MASK>>1) & 0xFF);
        TRISD |= ((CFG_KEYS_MASK>>9) & 0xFF);
        TRISA |= ((CFG_KEYS_MASK>>16) & 0x03);//设置为输入

        PULL_I_SELA_H = 0;  //设置步进
        PULL_I_SELA_L = 0xFF; //设置关闭电流源自适应

        CTK_LDO_SET(1);//CTK LDO 选择 0 为外部 LDO,1 为内部 LDO
        CTK_PD_SET(0);  //为 0 时 CTK 工作，为 1 时 CTK 不工作
        CTK_SYN_SET(CFG_SYN);//0 为非同步模式，1 为同步模
        CTK_RESO_SET(CFG_CTK_RESOLUTION); //0~7 CTK 电容扫描分辨率，
        //计数器位数：(CFG_CTK_RESOLUTION+9)位。
        CTK_PRS_SET(CFG_PRS_DIV); //设置 PRS 时钟
        //0:400K,1:1MHz,2:1MHz,3:3MHz,4:4MHz,5:6MHz,
        CTK_DS_SET(CFG_DETECT_SPEED); //检测速度 0:24MHz,1:12MHz,2:6MHz,3:4MHz
        CTK_PAR_SET(CFG_PAR); //0 为单通道模式，1 为多通道并联模式
        CTK_VTH_SEL(0);//0 为内部参考，1 外部参考
        CTK_RB_SET(CFG_RB);//选择 RB 电阻大小 0:20K,1:40K,2:60K,3:80K,
        //4:150K,3:200K,6:250K,7:300K。
        CTK_VTH_SET(CFG_VREF);//0~7 参考电压
        //(0:1.9V),(1:2.22V),(2:2.56V),(3:2.73V)
```

```
    CTK_IE_CLR;//不使能 CTK 中断
    CTK_STOP;    //CTK 模块停止扫描

    EA_OFF; //关总中断
    CTK_IP_SET;//设置 CTK 中断优先级为高
    CTK_IE_SET;//使能 CTK 中断
    EA_ON;  //开总中断

    currentIndex = 0;

    SetAccessIndexes();//设置通道索引

    for(i = 0;i < CH_MAX;i++)
    {
        if(accessIndex[i] != 0x7F)
        {
            currentIndex = i;
            currentIndex_parallel = currentIndex;
            CTK_ADDR_SET(accessIndex[currentIndex]);//查找第一个开启的通道
            break;
        }
    }

    SET_CTKEN();
    CurrentSourceInit()
    BaselineInit();
    SET_CTKEN();
}
```

第 12 章、ADC

GDMC181SXXA 芯片包含一个单端、12 位线性逐次逼近的模数转换器(ADC)，ADC 的基准电压与芯片的 VCC 相连。4 个 ADC 通道都可以输入独立的模拟信号，ADC 模块每次转换 1 个通道，ADC_START=1 开启转换，转换完成后更新 ADC 结果寄存器并产生一个中断。GDMC181SXXA 芯片的 ADC 模块具有以下特性：

- 12 位分辨率线性逐次逼近 ADC；
- 4 路模拟信号输入通道；
- 单次转换模式；
- 采样时间和转换速度可配置；



ADC 结构图

12.1、ADC 相关寄存器描述

ADC 通道 IO 使能寄存器　ADC_IO_SEL—0xACh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADC_IO_SEL	-	-	ADC5	ADC4	ADC3	ADC2	-	-
POR	-	-	0	0	0	0	-	-

ADC_IO_SEL：对应的 bit 置 1 则使能对应的管脚选择 ADC 功能，清零则对应的管脚选择 IO 功能。配置时应注意任意 IO 口同一时间只能使能其中一个功能。

功能控制寄存器　PD_ANA—0xDBh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PD_ANA	-	-	-	-	-	-	-	PD_ADC
POR	-	1	1	1	1	0	1	1

PD_ANA.1：ADC 模块使能控制位，PD_ADC=0 时 ADC 模块工作；PD_ADC=1 时 ADC 模块不工作。

ADC 采样时间配置寄存器　ADC_SPT—0xC1h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADC_SPT	-			ADC_SPT[4:0]				
POR	-	-	-	0	0	0	1	0

ADC_SPT[4:0]：设置 ADC 采样时间；ADC 采样时间 T_{ADC_SPT}=(ADC_SPT+1)*T_{adc_clk}，其中 T_{adc_clk}通过寄存器 ADCCKC 进行设置。

ADC 校准寄存器　ADCCALC—0xC2h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCCALC	-				ALINE_SEL[2:0]			CAL_EN
POR	-	-	-	-	0	0	0	0

ALINE_SEL：校准位选择寄存器；000/110/111：校准 D12 位；001：校准 D11 位；
010：校准 D10 位；011：校准 D9 位；100：校准 D8 位；101：校准 D7 位。
CAL_EN：校准使能寄存器，1：使能校准，0：禁止校准。

ADC 参数配置寄存器　ADCCKC—0xC4h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCCKC	ADCCALW[1:0]		ADCWNUM[1:0]		ADCCKV[1:0]		ADCK[1:0]	
POR	0	0	0	0	0	0	0	0

ADCCALW：转换时校准修正位宽选择；
00/11：4 位校准位宽；
01：5 位校准位宽；
10：6 位校准位宽。

ADCWNUM[5：采样完毕后与转换开启间隔时间选择,

00：3*T_{adc_clk};

01：4* T_{adc_clk};

10：5 * T_{adc_clk};

11：6 *T_{adc_clk};

采样完毕后间隔时间应 $T_{ADCWNUM}$ >=4* T_{ADCCKV}。

ADCCKV：ADC 模拟时钟选择寄存器,

00：24MHz;

01：12MHz;

10：8MHz;

11：4MHz

ADCK：adc_clk 时钟选择寄存器,

00：8MHz;

01：6MHz;

10：4MHz;

11：3MHz;

配置时需要注意： 采样完毕后与转换开启间隔时间 $T_{ADCWNUM}$>=4* T_{ADCCKV}。例：当 ADC 模拟时钟选择 24MHz，adc_clk 选择 8MHz 时，采样完毕后与转换开启时间必须>=(1/6)us。

ADC 转换时间：

公式	说明
$T_{AD}=T_{ADC\_SPT}+T_{W1}+T_{W2}$	ADC 转换时间
$T_{W1}=(ADCWNUM+3)*T_{adc\_clk}$	转换完毕间隔时间
$T_{W2}=((3*(ADCCALW+4))+(9-ADCCALW))*T_{adc\_clk}$	位宽校准时间，(ADCCALW=0~2)
$T_{ADC\_SPT(us)} = (ADC\_SPT+1)/F_{adc\_clk(MHz)}$	ADC 采样时间
$F_{adc\_clk(MHz)}$	ADC 分频时钟

ADC 偏置电流大小设置寄存器 ADC_I_SEL—0xC7h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADC_I_SEL	-						ADC_I_SEL[1:0]	
POR	-	-	-	-	-	-	0	0

ADC 转换结果寄存器(ADC_RDATAL—0xC6h、ADC_RDATAH—0xC5h)

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADC_RDATAL	ADC_RDATAL<7:0>							
POR	0x00							
ADC_RDATAH	-	-	-	-	ADC_RDATAH<11:8>			
POR	-	-	-	-	0	0	0	0

该组寄存器保存 ADC 转换后的数据，默认值为 0x00。

ADC 配置使能寄存器 ADC_SCAN_CFG—0xC3h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADC_SCAN_CFG					ADC_ADDR[2:0]			ADC_START
POR	-	-	-	-	0	0	0	0

ADC_ADDR：ADC 通道地址寄存器；该寄存器配置转换通道。

ADC_START：ADC 转换开启使能控制寄存器；1：开启 ADC 转换，0：关闭 ADC 转换。向 ADC_START 寄存器写 1 开启一次 ADC 转换，转换完成硬件自动清零。若要开启下一次 ADC 转换，需要软件置 1；转换过程中向 ADC_START 写 0 则停止此次转换。

ADC 中断使能寄存器 IEN1—0xE6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IEN1	-	-	-	IEN1.4	-	-	-	-
POR	0	0	0	0	0	0	0	0

IEN1.4：ADC中断使能位；IEN1.4=1：开启ADC中断；IEN1.4=0：关闭ADC中断。

ADC 中断优先级选择寄存器 IPL1—0xF6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IPL1	-	-	-	IPL1.4	-	-	-	-
POR	0	0	0	0	0	0	0	0

IPL1.4：ADC中断优先级选择位；IPL1.4=1：设置ADC中断高优先级；IPL1.4=0：ADC中断为低优先级。

ADC 中断标志位寄存器 IRCON1—0xF1h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IRCON1	-	-	-	IRCON1.4	-	-	-	-
POR	0	0	0	0	0	0	0	0

IRCON1.4：ADC中断标志位；ADC转换完成该位置1；中断服务函数中软件清零。

REG_ADDR 二级总线地址寄存器 REG_ADDR—0x96h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
REG_ADDR	REG_ADDR[7:0]							
POR	0	0	0	0	0	0	0	0

REG_ADDR[7:0]：二级总线地址寄存器，读取ADC校准码时参照下表地址：

REG_ADDR	校准位
0x09	D12
0x0A	D11
0x0B	D10
0x0C	D9
0x0D	D8
0x0E	D7

REG_DATA 二级总线数据寄存器 REG_DATA—0x97h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
REG_DATA	REG_ADDR[7:0]							
POR	0	0	0	0	1	1	1	1

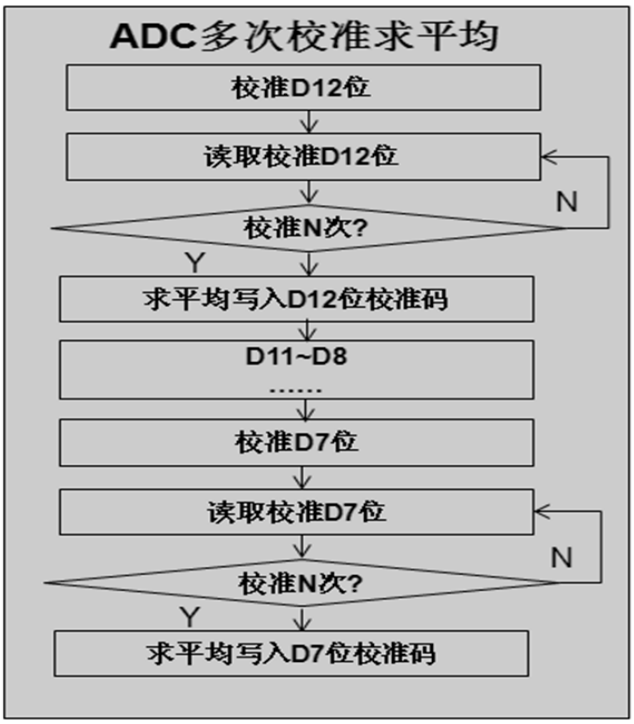
REG_ADDR[7:0]：二级总线数据寄存器，读取ADC校准码流程如下：

1、 REG_ADDR = addr;//写址到二级总线地址；

2、 Read_Tem = REG_DATA;//从二级总线读回对应地址数据；

3、 修改地址，依次读取校准码

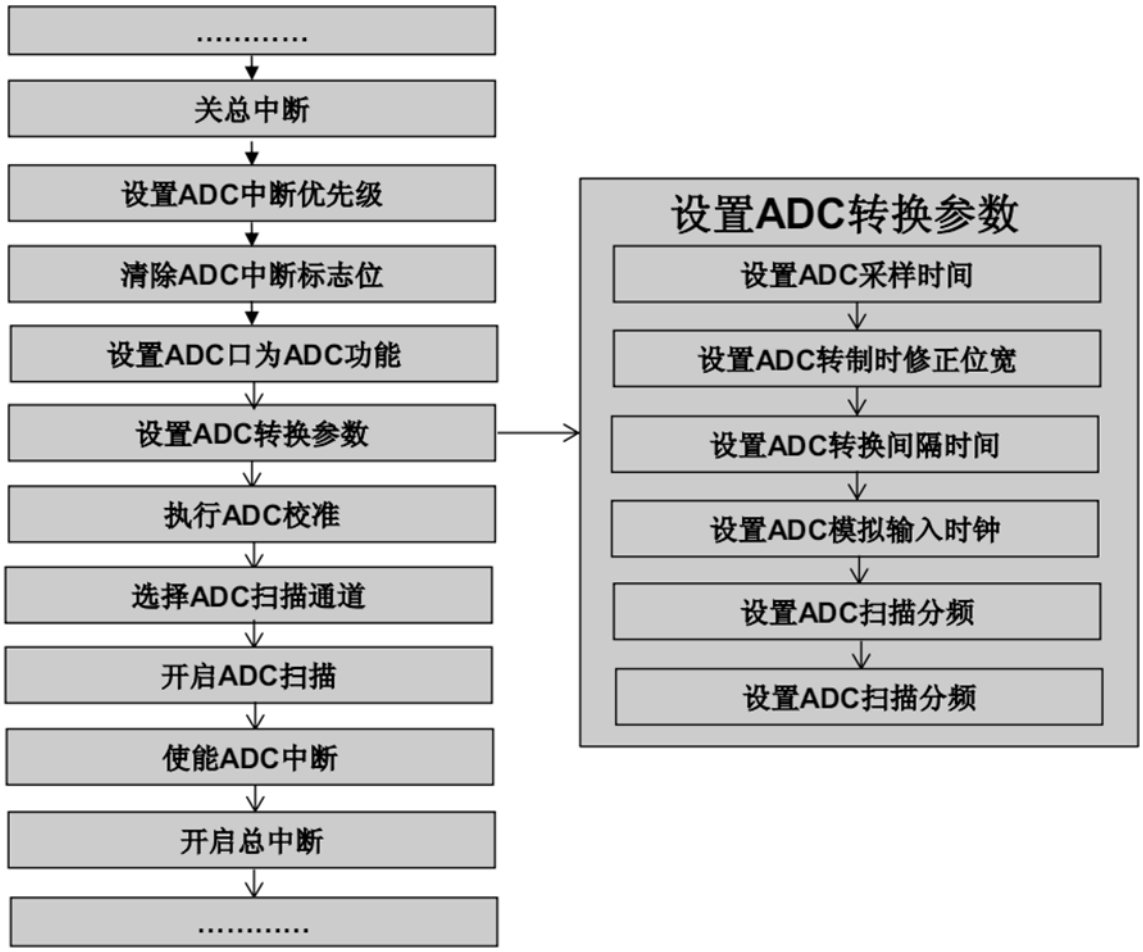
注：ADC校准可多次校准求平均后，写入校准寄存器，以提高校准精度。



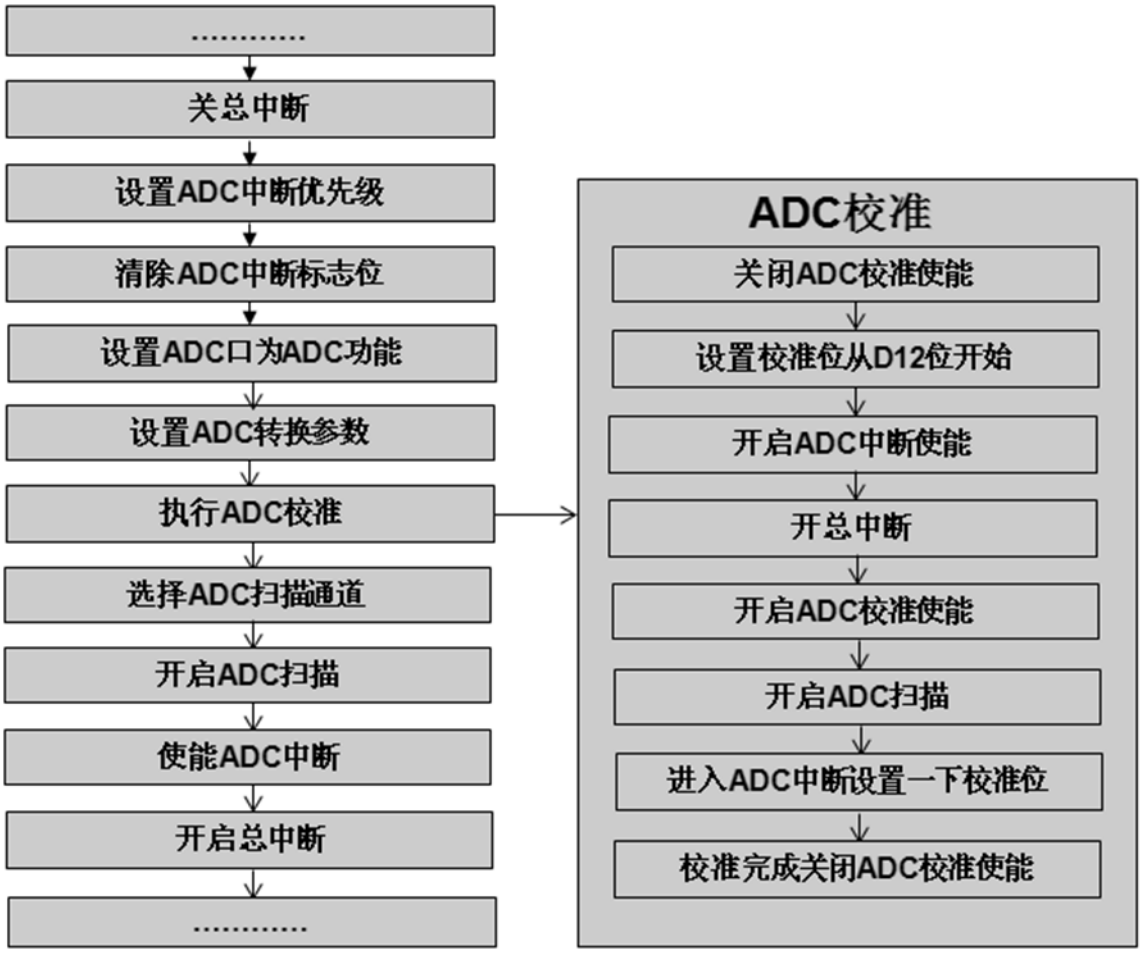
ADC多次校准求平均

12.2、ADC 配置流程

在进行 ADC 转换前，进行 ADC 校准，以提高检测精度，校准流程如下：



ADC 配置流程


.....
关总中断
设置ADC中断优先级
清除ADC中断标志位
设置ADC口为ADC功能
设置ADC转换参数
执行ADC校准
选择ADC扫描通道
开启ADC扫描
使能ADC中断
开启总中断
.....
ADC校准
关闭ADC校准使能
设置校准位从D12位开始
开启ADC中断使能
开总中断
开启ADC校准使能
开启ADC扫描
进入ADC中断设置一下校准位
校准完成关闭ADC校准使能

ADC 校准流程

12.3、ADC 特性

（除非另有规定，TA = +25° C）

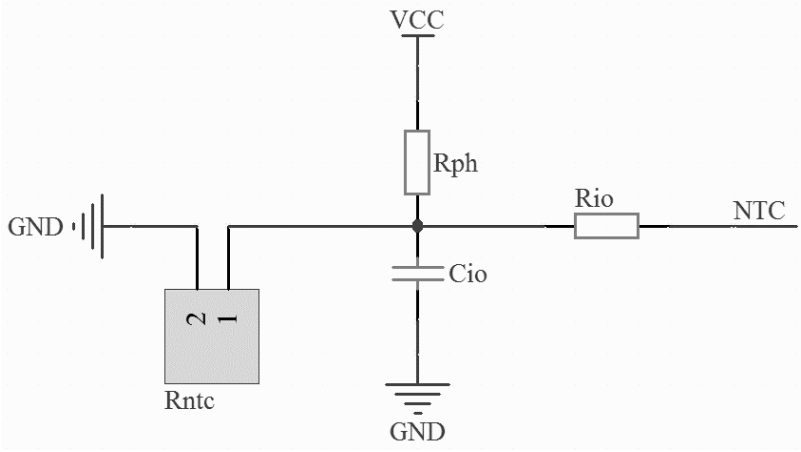
参 数 名 称	符 号	测 试 条 件	最小	典型	最大	单 位
工作电压	VDD		V_{LVR}		5.5	V
ADC 输入电压范围	V_{ADCIN}		0		VDD	V
分辨率	ADC_{RESO}		12			bit
ADC 时钟周期	T_{AD}	(ADC_SPT+1)*T_{adc_clk}	0.416			us
输入通道					7	通道
精度			10			Bit
无丢码			10			Bits
ADC 转换时间	T_{CON}	$T_{AD}=T_{ADC\_SPT}+T_{W1}+T_{W2}$	3.125			us
积分线性误差	E_{INL}			±2	±3	LSB
微分线性误差	E_{DNL}			±1	±2	LSB

12.4、ADC 初始化 C 语言参考

```
//-------------------------------------------------------------------//
//函数名称：  void ADC_Init(void)
//函数功能：  ADC 初始化
//输入参数：  无
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
void ADC_Init(void)
{
    EA_OFF;//关总中断；
    ADC_IP_SET;//设置 ADC 中断优级为高
    ADC_INT_FLAG_CLR;//清除 ADC 中断标志位
    ADC_IO_SEL(OPENADC);//ADC 与 IO 功能选择，
    //对应的位为 1 为 ADC 功能，0 为 IO 功能
    ADC_SAMP_SET(ADC_SAMPT);//ADC 采样时间设置 bit[4:0],
    //(0~32)采样时间为(ADC_SPT+1)*T_adc_clk
    ADC_CALW_SET(ADC_CALW);//转换时校准备修正位宽选择:
    //bit[7:6],转换修正位宽(0 和 3)--校准 4 位，(1-2)--校准 5-6 位
    ADC_WNUM_SET(ADC_WAIT);//bit[5:4],转换完毕距离转换时间选择,
    //(1~3-(X+3)*T_adc_clk>4*T_ADC_DCLK
```

```
    ADC_ANCK_SET(ADC_ANCK);//模拟输入信号时钟分频选择，
    //bit[3:2],(0~3)-(24MHz,12MhHz,8MHz,4MHz)
    ADC_DCLK_SET(ADC_DCLK);//ADC 分频选择，
    //bit[1:0],(0~3)-(8MHz,6MhHz,4MHz,3MHz)
    ADC_OFFSETI_SET(ADC_OFFSETI);//ADC 偏置电流大小选择，
    //bit[2:0],(0~3)

    ADC_Adjust();//ADC 校准
    for(ADC_Index_Start = 2;ADC_Index_Start < 19;ADC_Index_Start++)
    //查找第一个 ADC 通道
    {
        if(accessIndex[ADC_Index_Start] & 0x80)
        {
            break;
        }
    }

    ADC_ADDR_SET(accessIndex[ADC_Index_Start]);
    ADC_Index = ADC_Index_Start;
    ADC_SCAN_EN;//开启 ADC 扫描

    ADC_IE_SET;//开启 ADC 中断使能
    EA_ON;//开总中断
}
```

12.5、ADC 应用注意事项



温度检测参考电路

注意：

1、 主要应用于饭煲、压力锅、热水器、茶炉、养生壶等需温控场合，ADC 精度 8 位；

2、 建议 Rph 值不大于 100KΩ(精密电阻)、Rio=100Ω、Cio=102；

3、 Rio、Cio 起端口滤波作用，应尽量靠近芯片引脚；若不接需加强软件数字滤波；

4、 养生壶、茶炉：Rph 一般使用 27KΩ；

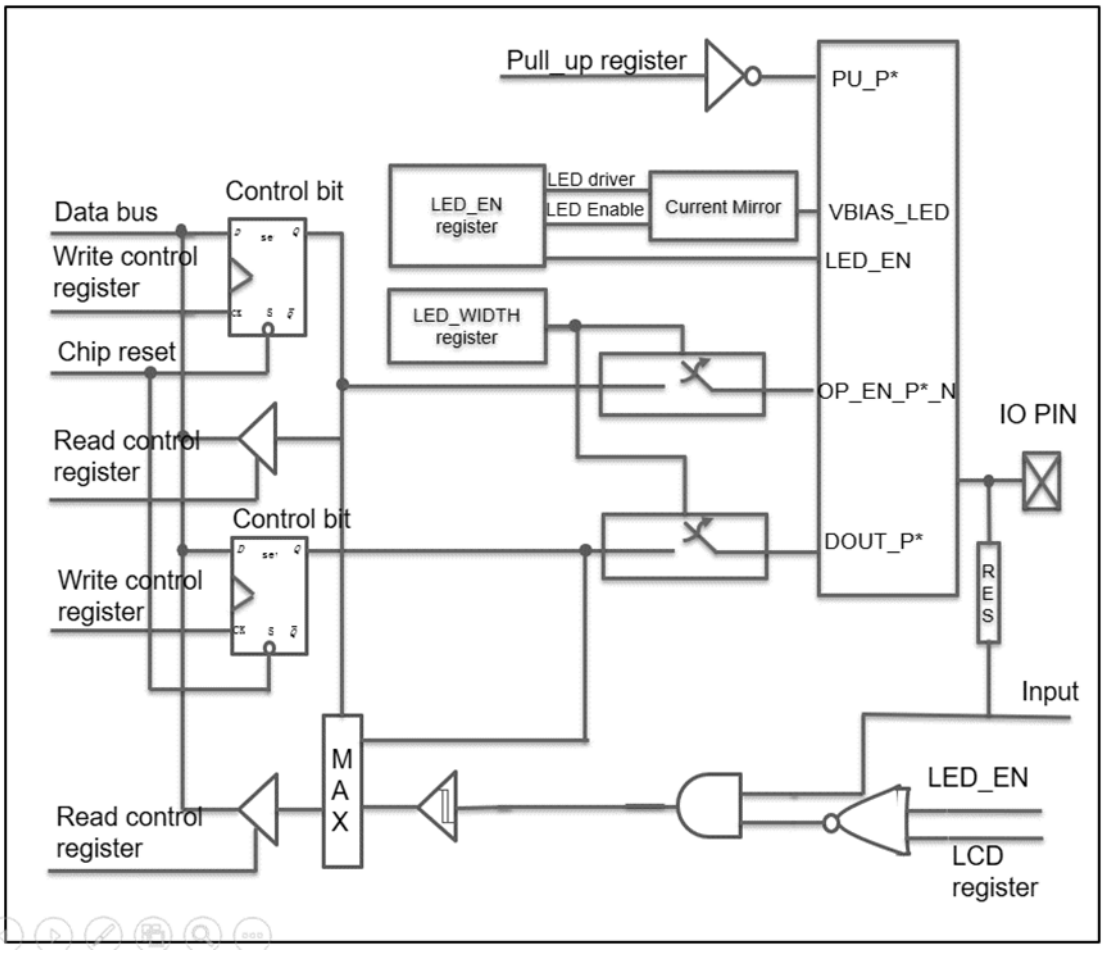
5、 压力煲：Rph 一般使用 5.6KΩ；

6、 电饭煲：Rph 一般使用 10KΩ；

第 13 章、LED/LCD 串行点阵驱动

LED 串行点阵驱动模块可通过软件配置实现 8*8、7*8、7*7、6*7、6*6、5*5、4*4 矩阵串行驱动。LED 导通时间可配置，可软件配置成循环扫描模式或中断扫描模式。GDMC181SXXA 中 LED 串行点阵驱动具有以下特点：

- 最大支持驱动 64 个 LED 灯
- 单个 LED 灯导通时间可设置；配置范围 8us-2.048ms，步进为 8us
- 设置间隔地址配置两种不同的驱动时间
- IO 口有多重复用关系，软件配置 LED 功能，矩阵模式自动开启对应管脚的 LED 功能
- 64 个灯具有唯一的地址，寄存器控制亮灭
- 扫描模式可配置：中断模式扫描完成产生中断；循环模式扫描一帧后自动开启下一帧扫描，不产生中断。



LED 结构图

13.1、LED 相关寄存器描述

LED 亮灯配置寄存器 1　SEG_SEL_1—0xB1h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEG_SEL_1	LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED0
POR	0	0	0	0	0	0	0	0

LED 亮灯配置寄存器 2　SEG_SEL_2—0xB2h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEG_SEL_2	LED15	LED14	LED13	LED12	LED11	LED10	LED9	LED8
POR	0	0	0	0	0	0	0	0

LED 亮灯配置寄存器 3　SEG_SEL_3—0xB3h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEG_SEL_3	LED23	LED22	LED21	LED20	LED19	LED18	LED17	LED16
POR	0	0	0	0	0	0	0	0

LED 亮灯配置寄存器 4　SEG_SEL_4—0xB4h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEG_SEL_4	LED31	LED30	LED29	LED28	LED27	LED26	LED25	LED24
POR	0	0	0	0	0	0	0	0

LED 亮灯配置寄存器 5　SEG_SEL_5—0xB5h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEG_SEL_5	LED39	LED38	LED37	LED36	LED35	LED34	LED33	LED32
POR	0	0	0	0	0	0	0	0

LED 亮灯配置寄存器 6　SEG_SEL_6—0xB6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEG_SEL_6	LED47	LED46	LED45	LED44	LED43	LED42	LED41	LED40
POR	0	0	0	0	0	0	0	0

LED 亮灯配置寄存器 7　SEG_SEL_7—0xBAh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEG_SEL_7	LED55	LED54	LED53	LED52	LED51	LED50	LED49	LED48
POR	0	0	0	0	0	0	0	0

LED 亮灯配置寄存器 8　SEG_SEL_8—0xBBh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEG_SEL_8	LED63	LED62	LED61	LED60	LED59	LED58	LED57	LED56
POR	0	0	0	0	0	0	0	0

上述 8 个寄存器用来控制每个 LED 的亮灭，如果希望某一个 LED 点亮则相应的位置 1，清零则对应的 LED 熄灭。如希望 LED7 灯点亮则 SEG_SEL_1 寄存器的 bit8 置 1。

注：没有焊接的 LED 对应寄存器位值为 0。

LED 亮灯时间配置寄存器 1　LED1_WIDTH—0xBCh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LED1_WIDTH	LED1_WIDTH[7:0]							
POR	0	0	0	0	0	0	0	0

LED1_WIDTH：第一段 LED 灯每个灯亮灯时间控制寄存器；第一段 LED 灯单个 LED 灯亮灯时间 T_{LED1}=(LED1_WIDTH+1)*8us。

LED 亮灯时间配置寄存器 2　LED2_WIDTH—0xBDh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LED2_WIDTH	LED2_WIDTH[7:0]							
POR	0	0	0	0	0	0	0	0

LED2_WIDTH：第二段 LED 灯每个灯亮灯时间控制寄存器；第二段 LED 灯单个 LED 灯亮灯时间 T_{LED2}=(LED2_WIDTH+1)*8us。

LED 段点坐标寄存器　LED_SEG_POINT—0xBEh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LED_SEG_POINT	-		LED_SEG_POINT[5:0]					
POR	-	-	0	0	0	0	0	0

LED_SEG_POINT：设置两段 LED 灯分界坐标；在 8*8 的矩阵中，如该寄存器设置 32 则表示：LED0-LED32 为第一段 LED，LED33-LED63 为第二段 LED。

合理配置 LED1_WIDTH、LED2_WIDTH 和 LED_SEG_POINT 寄存器可以实现两段 LED 呈现不同的亮度。

LED 驱动能力配置寄存器 LED_DRIVE—0xBFh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LED_DRIVE	-	-	-	LED_DRIVE[4:0]				
POR	-	-	-	0	0	0	0	0

LED_DRIVE：LED 驱动能力配置寄存器，驱动电流大小参照下表；通过调节该寄存器可以实现控制 LED 灯亮度。

LED_DRIVE	驱动电流(mA)	LED_DRIVE	I_LED 驱动电流(mA)
0	3	16	50
1	6	17	53
2	9	18	56
3	12	19	58
4	15	20	61
5	18	21	64
6	21	22	66
7	24	23	69
8	27	24	71
9	30	25	74
10	33	26	76
11	36	27	78
12	38	28	80
13	41	29	81
14	44	30	83
15	47	31	84

LED 串行点阵驱动单个 LED 灯的平均电流为：I_LED/(矩阵模式 LED 数)，例：8x8 矩阵，单个 LED 的平均电流为：I_LED /(8*8)。

注： I_LED 的设置建议小于 LED 灯标称的 Ifp 电流,所驱动的 LED 应选择正向电压 V_F 一致的 LED 灯。

LED 控制寄存器 LED_MODE—0xC0h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LED_MODE	-			LED_MATRIX_SEL[2:0]			LED_SCAN_MODE	LED_START
POR	-	-	-	0	0	0	0	0

LED_MATRIX_SEL：LED 驱动矩阵选择寄存器；000：不选择 LED 矩阵功能；
001：4x4 矩阵；010：5x5 矩阵；011：6x6 矩阵
100：6x7 矩阵；101：7x7 矩阵；110：7x8 矩阵
111：8x8 矩阵。无论选择哪种矩阵模式，始终以 LED0 为扫描起始点。选定相应的 LED 矩阵后，对应的 IO 管脚自动设置为 LED 功能。开启

LED 功能前需要取消对应 IO 口的复用功能。

LED_SCAN_MODE：LED 扫描模式选择寄存器，该位置 1 选择循环模式，清 0 则选择中断模式。循环模式下开启 LED 扫描后将一直循环扫描，当软件关闭扫描时立即停止后续 LED 的扫描。中断模式完成一帧扫描后停止扫描并产生中断。

LED_START：LED 扫描开启寄存器，向该寄存器写 1 时开启 LED 扫描，循环模式和中断模式向该寄存器写 0 则立即停止 LED 扫描。

LED 中断使能寄存器 IEN1—0xE6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IEN1	-	IEN1.6	-	-	-	-	-	-
POR	0	0	0	0	0	0	0	0

IEN1.6：LED中断使能位；IEN1.6=1：开启LED中断；IEN1.6=0：关闭LED中断。

LED 中断优先级选择寄存器 IPL1—0xF6h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IPL1	-	IPL1.6	-	-	-	-	-	-
POR	0	0	0	0	0	0	0	0

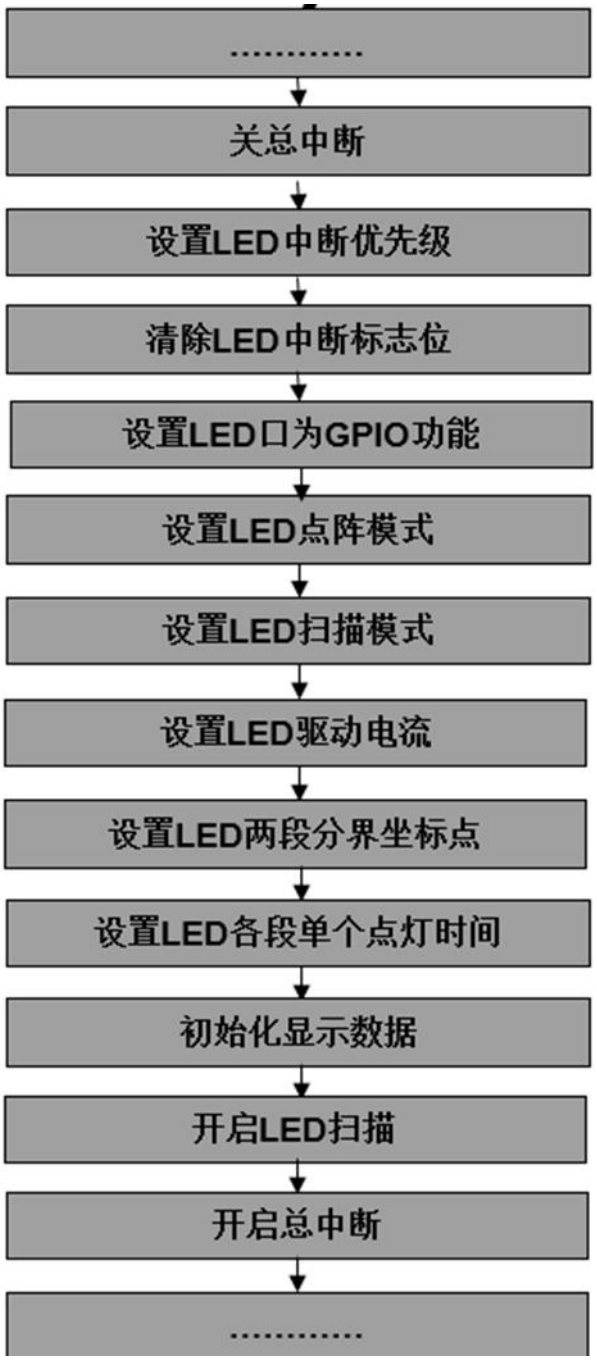
IPL1.6：LED中断优先级选择位；IPL1.6=1：设置LED中断高优先级；IPL1.6=0：LED中断为低优先级。

LED 中断标志位寄存器 IRCON1—0xF1h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IRCON1	-	IRCON1.6	-	-	-	-	-	-
POR	0	0	0	0	0	0	0	0

IRCON1.6：LED中断标志位；中断模式LED完成一帧扫描该位置1；中断服务函数中软件清零。

13.2、LED 配置流程



LED 配置流程

LED 两段点灯时间配置关系：

名称	公式	说明
T_LED(ms)	T_LED(ms)=T_LED1(ms)+T_LED2(ms)<=16.66ms	LED 总亮灯时间
T_LED1(ms)	T_LED1(ms)=T_LED(ms)-T_LED2(ms)	第一段亮灯时间
T_LED2(ms)	T_LED2(ms)=T_LED(ms)-T_LED1(ms)	第二段亮灯时间
T_LED1_N(us)	T_LED1_N(us)=T_LED1(ms)/(LED1_NUM,T_LED1_N(us)<=2.048ms	第一段单个亮灯时间
T_LED2_N(us)	T_LED2_N(us)=T_LED2(ms)/(X*Y-LED1_NUM)，(X*Y-LED1_NUM)=0 时，T_LED2_N(us)为 0T_LED2_N(us)<=2.048ms。	第二段单个亮灯时间
LED1_WIDTH	LED1_WIDTH=(T_LED1_N(us)/8-1)	第一段单个亮灯寄存器值
LED2_WIDTH	LED2_WIDTH=(T_LED2_N(us)/8-1)	第二段单个亮灯寄存器值
LED1_NUM	LED1_NUM	第一段点灯数
X*Y	4*4、5*5、6*6、6*7、7*7、7*8、8*8	矩阵行*矩阵列

各种矩阵模式扫描对应 X*Y 坐标点：

8*8 坐标							
0	15	16	31	32	47	48	63
1	14	17	30	33	46	49	62
2	13	18	29	34	45	50	61
3	12	19	28	35	44	51	60
4	11	20	27	36	43	52	59
5	10	21	26	37	42	53	58
6	9	22	25	38	41	54	57
7	8	23	24	39	40	55	56

7*8 坐标							
0	15	16	31	32	47	48	63
1	14	17	30	33	46	49	62
2	13	18	29	34	45	50	61
3	12	19	28	35	44	51	60
4	11	20	27	36	43	52	59
5	10	21	26	37	42	53	58
6	9	22	25	38	41	54	57
-	-	-	-	-	-	-	-

7*7 坐标							
0	15	16	31	32	47	48	-
1	14	17	30	33	46	49	-
2	13	18	29	34	45	50	-
3	12	19	28	35	44	51	-
4	11	20	27	36	43	52	-
5	10	21	26	37	42	53	-
6	9	22	25	38	41	54	-
-	-	-	-	-	-	-	-

4*4 坐标							
0	15	16	31	-	-	-	-
1	14	17	30	-	-	-	-
2	13	18	29	-	-	-	-
3	12	19	28	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-

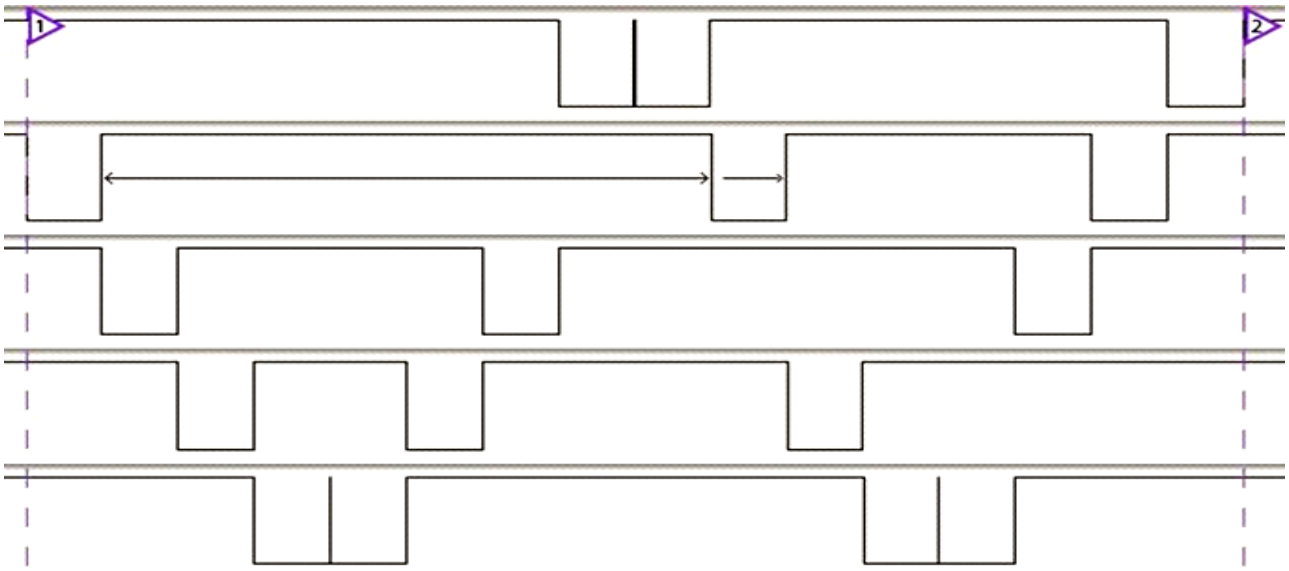
6*7 坐标							
0	15	16	31	32	47	48	-
1	14	17	30	33	46	49	-
2	13	18	29	34	45	50	-
3	12	19	28	35	44	51	-
4	11	20	27	36	43	52	-
5	10	21	26	37	42	53	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-

6*6 坐标							
0	15	16	31	32	47	-	-
1	14	17	30	33	46	-	-
2	13	18	29	34	45	-	-
3	12	19	28	35	44	-	-
4	11	20	27	36	43	-	-
5	10	21	26	37	42	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-

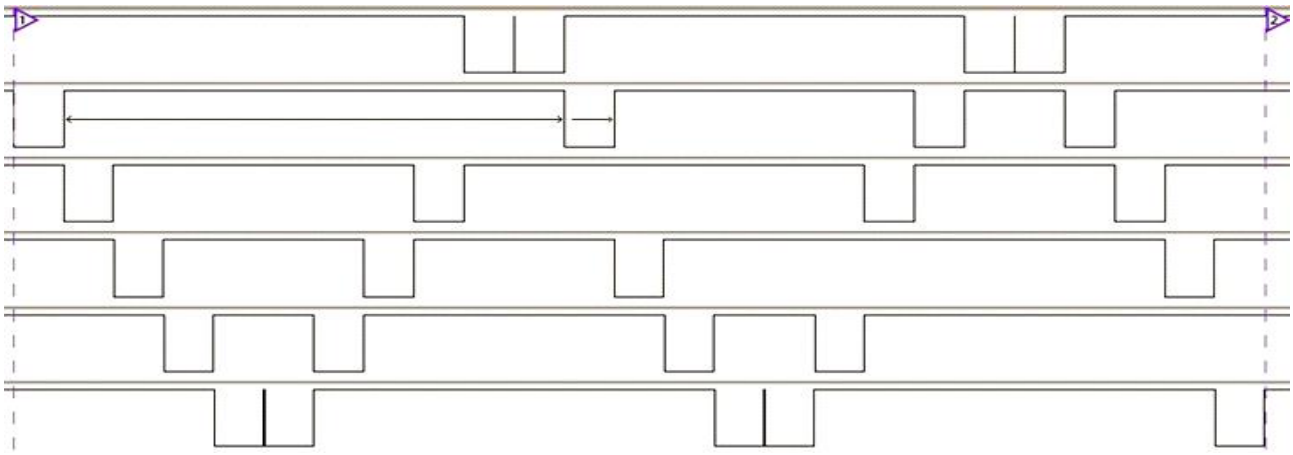
5*5 坐标							
0	15	16	31	32	-	-	-
1	14	17	30	33	-	-	-
2	13	18	29	34	-	-	-
3	12	19	28	35	-	-	-
4	11	20	27	36	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-

说明
LED 扫描时，按照 0~到（X*Y-1）的顺序进行扫描。

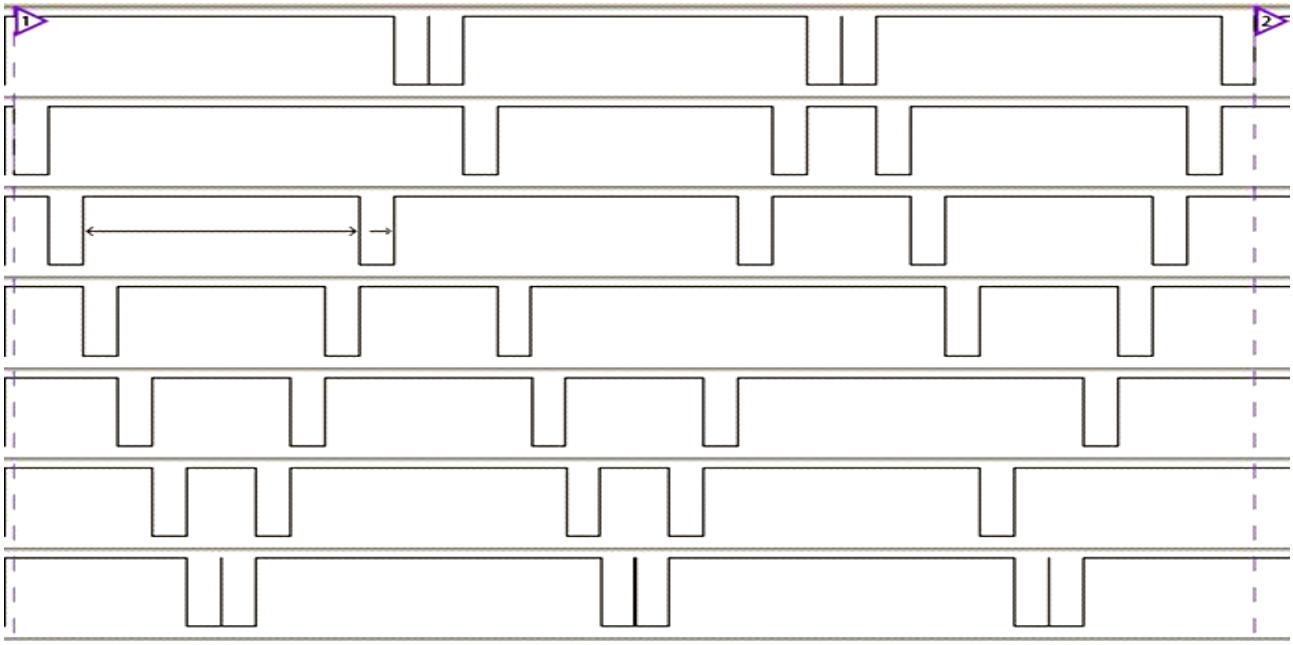
13.3、LED 扫描时序



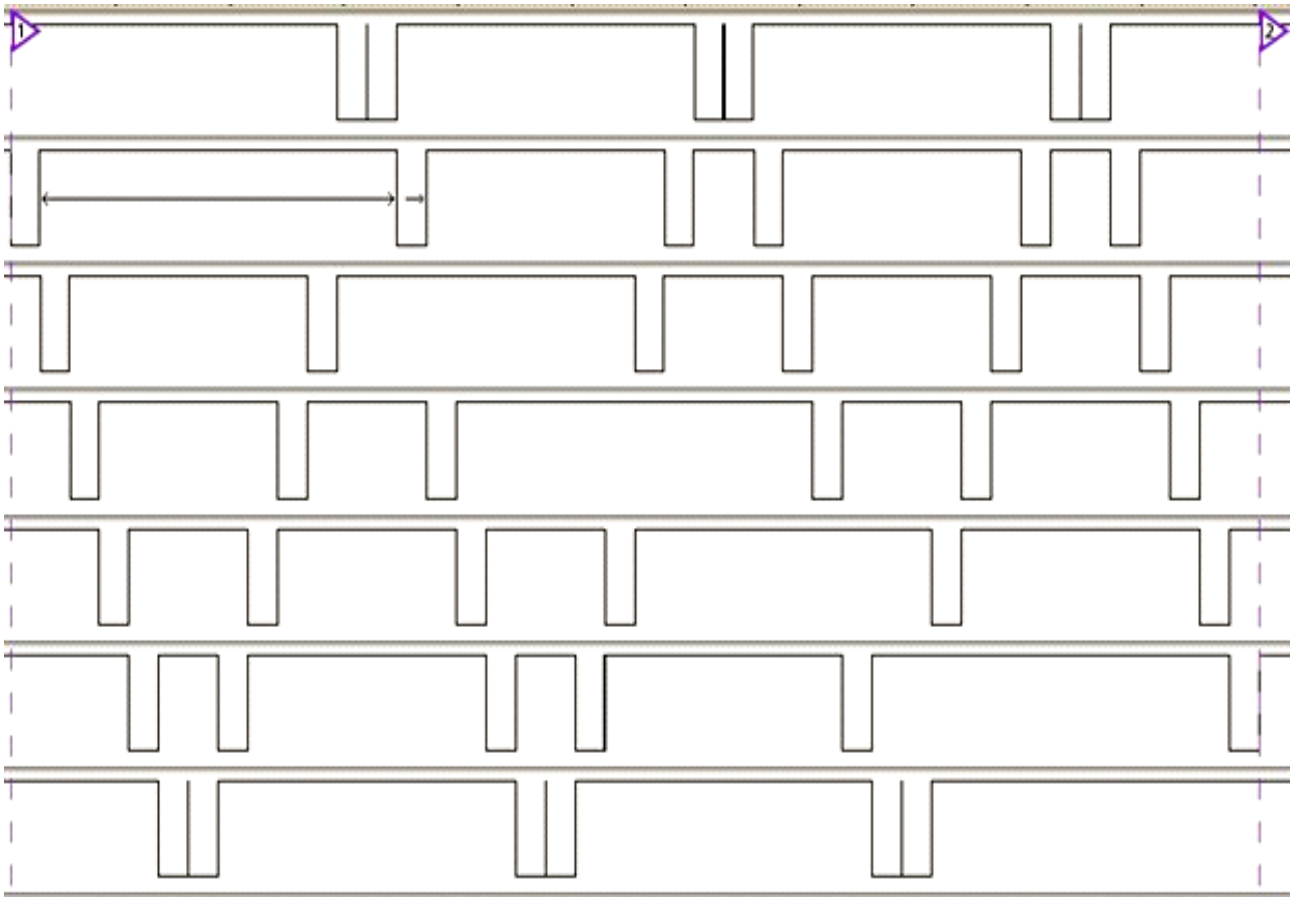
4*4



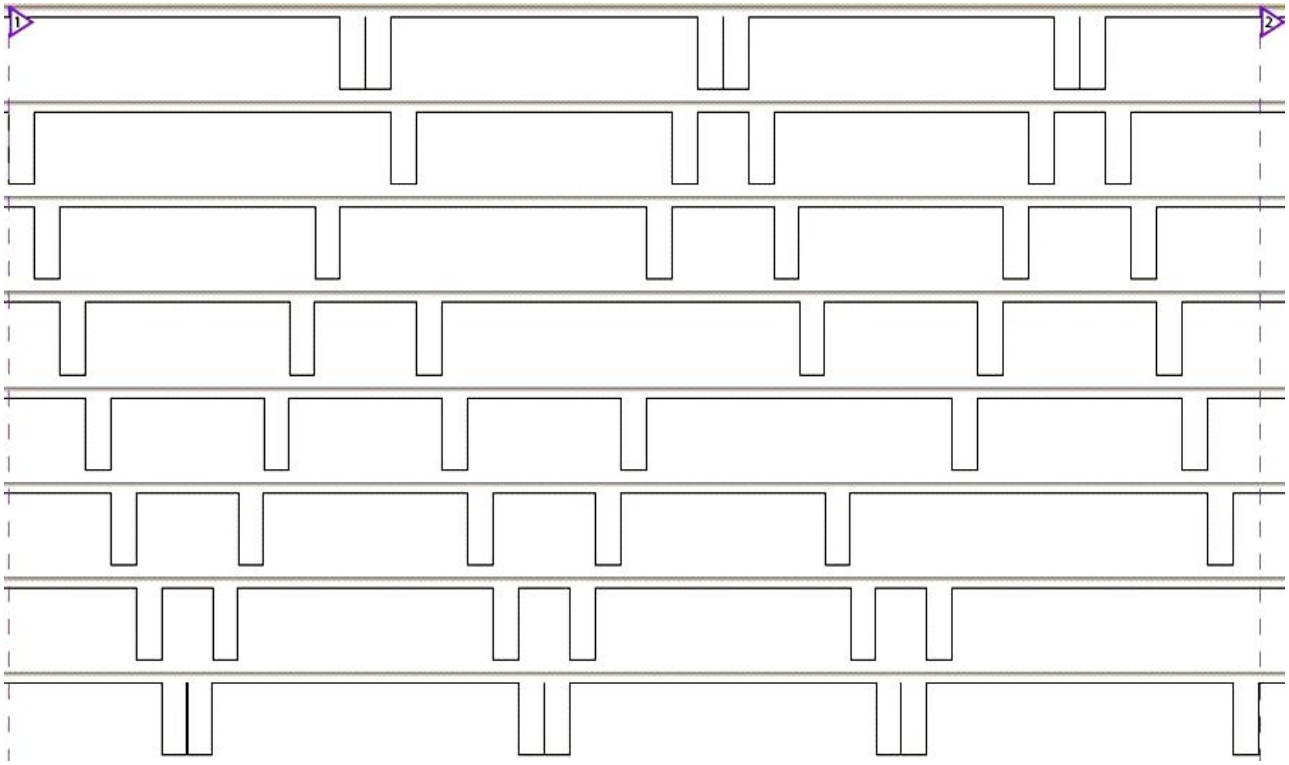
5*5



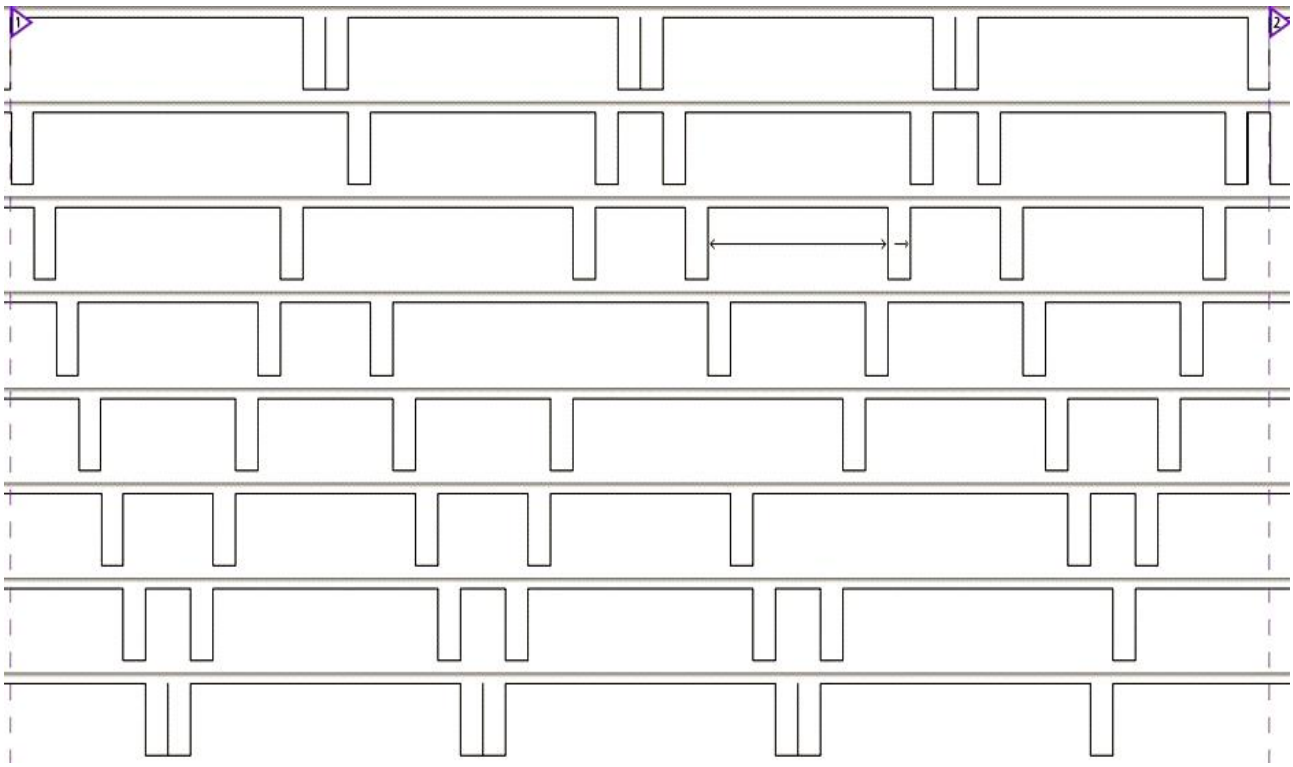
6*6



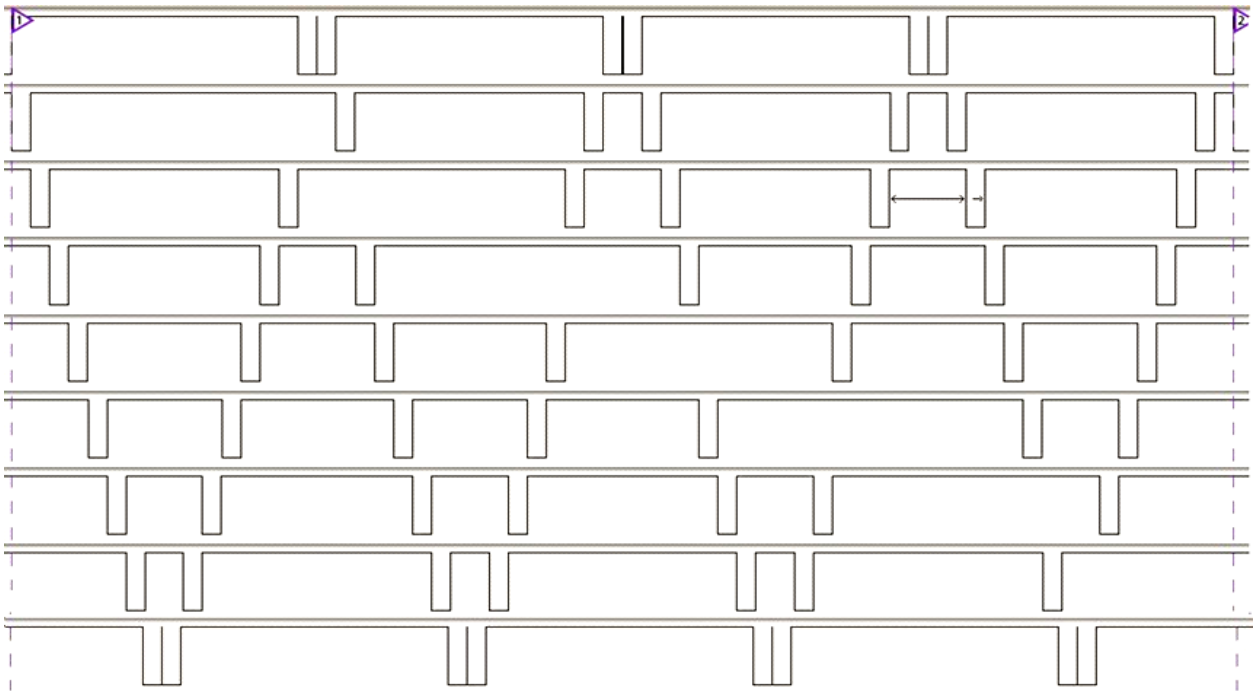
6*7



7*7



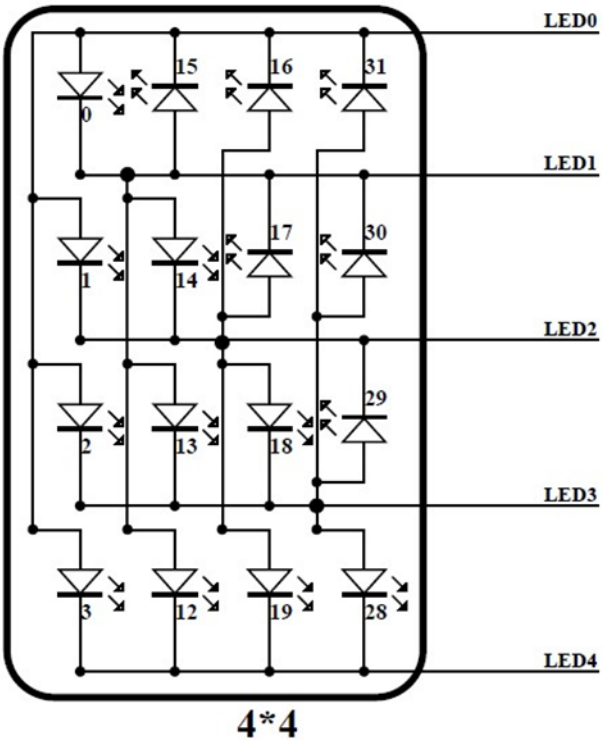
7*8



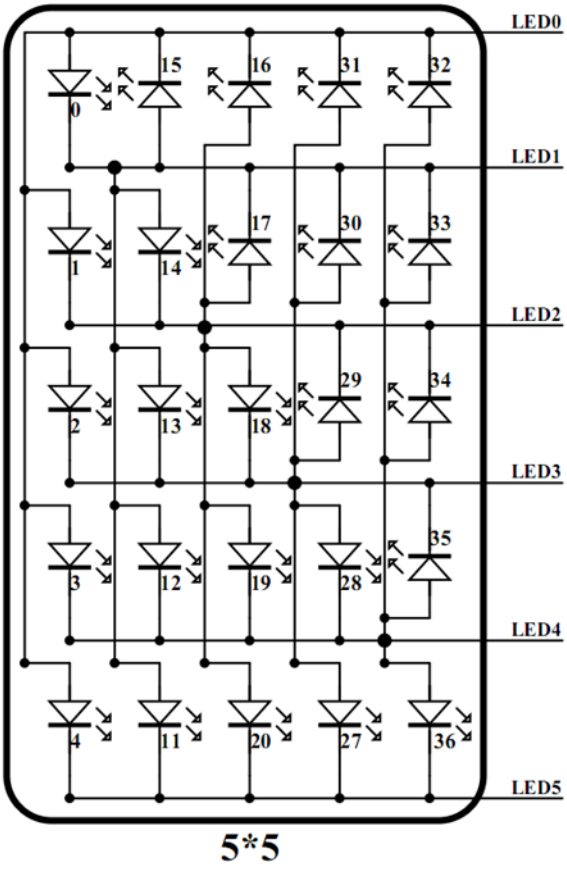
8*8

13.4、LED 点阵列灯

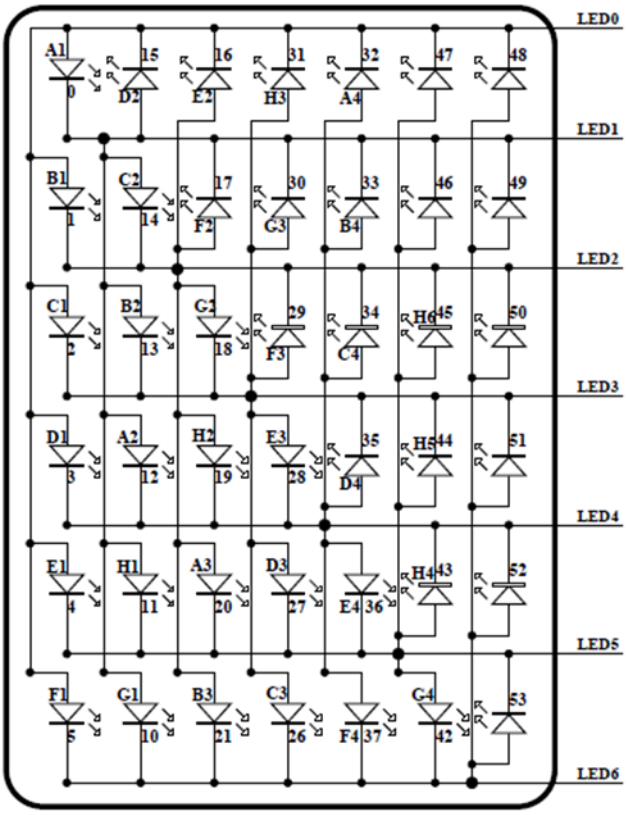
在 LED 扫描模式下，对应有 4*4、5*5、6*6、6*7、7*7、7*8、8*8 串行点阵模式，通过寄存进行配置来实现，根据应用需求来选择合适的点阵模式和对应的定制数码管。



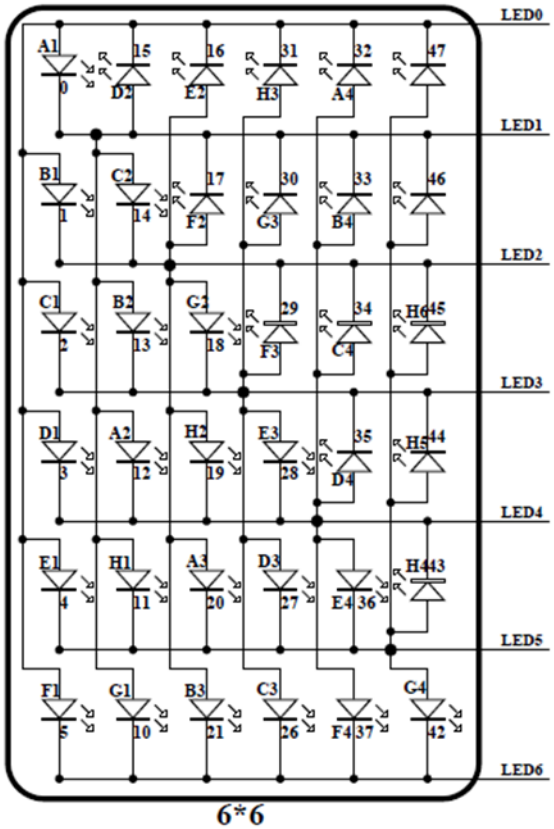
4*4



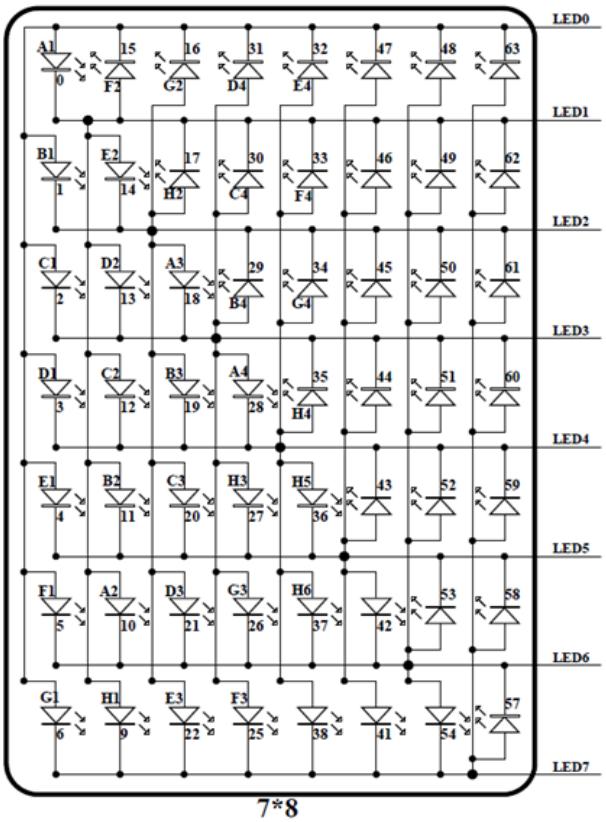
5*5


LED0
LED1
LED2
LED3
LED4
LED5
LED6

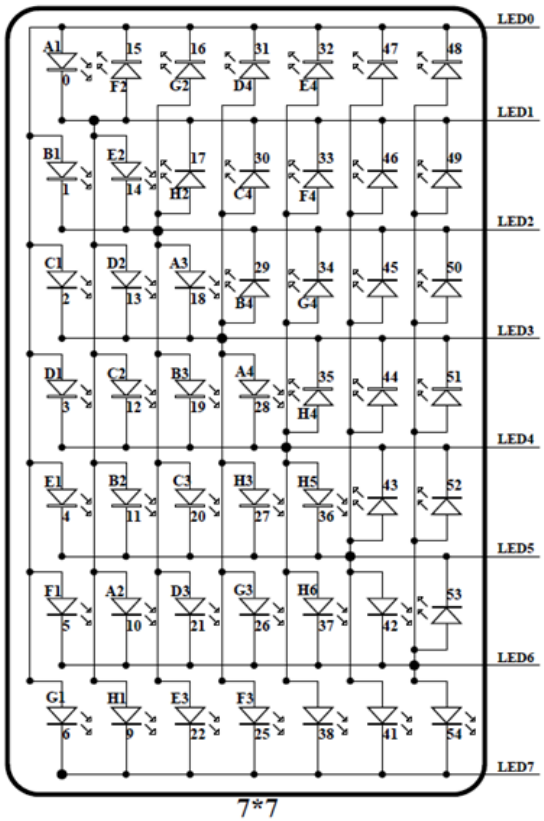
6*7


LED0
LED1
LED2
LED3
LED4
LED5
LED6

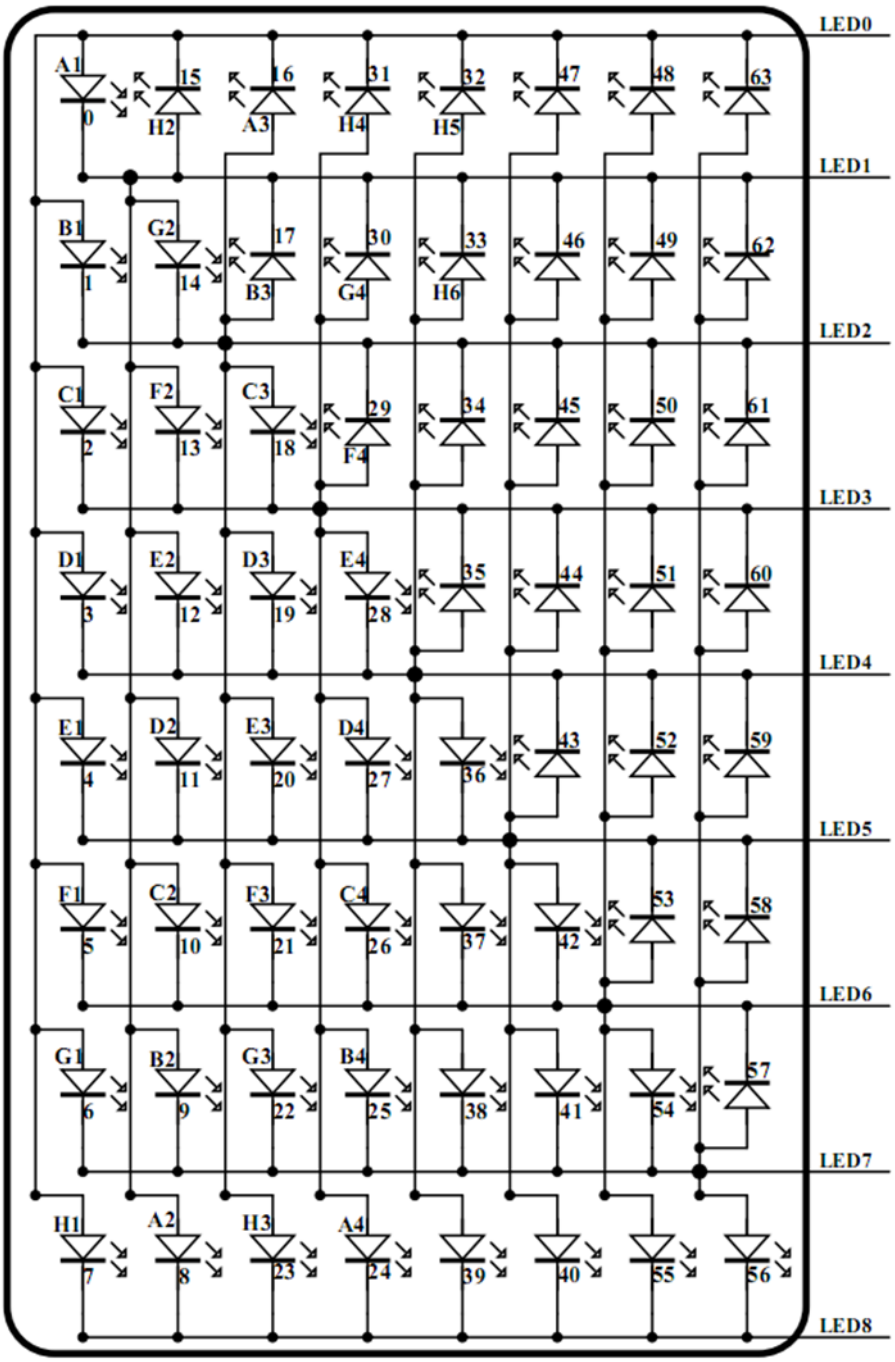
6*6


LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7

7*8


LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7

7*7


LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
A1 0 15 H2 16 A3 31 H4 32 H5 47 48 63
B1 1 G2 14 17 B3 30 G4 33 H6 46 49 62
C1 2 F2 13 C3 18 29 F4 34 45 50 61
D1 3 E2 12 D3 19 E4 28 35 44 51 60
E1 4 D2 11 E3 20 D4 27 36 43 52 59
F1 5 C2 10 F3 21 C4 26 37 42 53 58
G1 6 B2 9 G3 22 B4 25 38 41 54 57
H1 7 A2 8 H3 23 A4 24 39 40 55 56

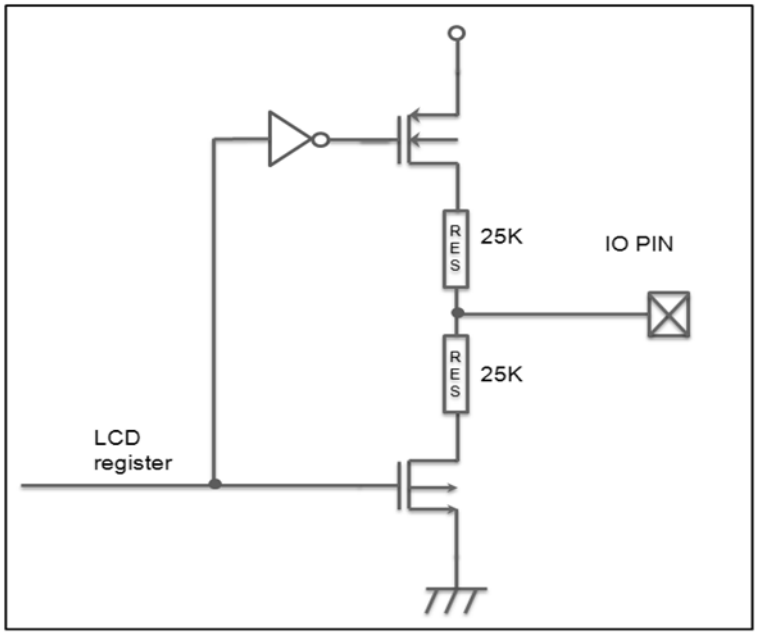
8*8

13.5、LED 初始化 C 语言参考

```
//-------------------------------------------------------------------//
//函数名称：  void LED_Init(void)
//函数功能：  LED 驱动初始化
//输入参数：  无
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
void LED_Init(void)
{
    EA_OFF;//关总中断；
    LED_IP_SET; //设置 LED 中断优先级为高
    LED_INT_FLAG_CLR; // 清除 LED 中断标志位

    LED_MATRIX_SET(4);//LED 串行点阵矩阵选择:
    //(X*Y) 0-无，1-4*4，2-5*5，3-6*6，4-6*7，5-7*7，6-7*8，7-8*8
    LED_MODE_SET(1);//LED 扫描模式选择，0 为中断模式，1 为循环模式

    LED_CURRENT_SET(15);//建议 15，LED 驱动能力配置，
    //参照规格书表格(0-3mA),(9-30mA),(21-64mA)
    LED_POINT_SET(53);//LED 两段坐标配置,第一段为(0~x),第二段为(x+1,63)

    LED_SEG1_WIDTH_SET(11);//第一段单个灯导通时间(建议 96us)，(x+1)*8us
    LED_SEG2_WIDTH_SET(0); //第二段单个灯导通时间，(x+1)*8us


    LED_Data(0xFFFFFFFF,0xFFFFFFFF);//LED 数据显示初始化

    LED_IE_SET;//开 LED 中断使

    LED_START_SET(1);//LED 扫描开始停止控制，0 为停止扫描，1 为开始扫描
    EA_ON;//开总中断
}
```

13.6、LCD 相关

LCD 驱动功能是通过配置 PB0~PB5 口，结合软件输出 LCD 驱动时序来驱动 LCD。PB0~PB5 内置上下拉电阻 25K 电阻，作为 1/2bais 的 LCD 驱动用。



LCD IO 驱动口结构

13.7、LCD 相关寄存器描述

LCD 1/2biase 输出使能寄存器 COM_EN—0xABh

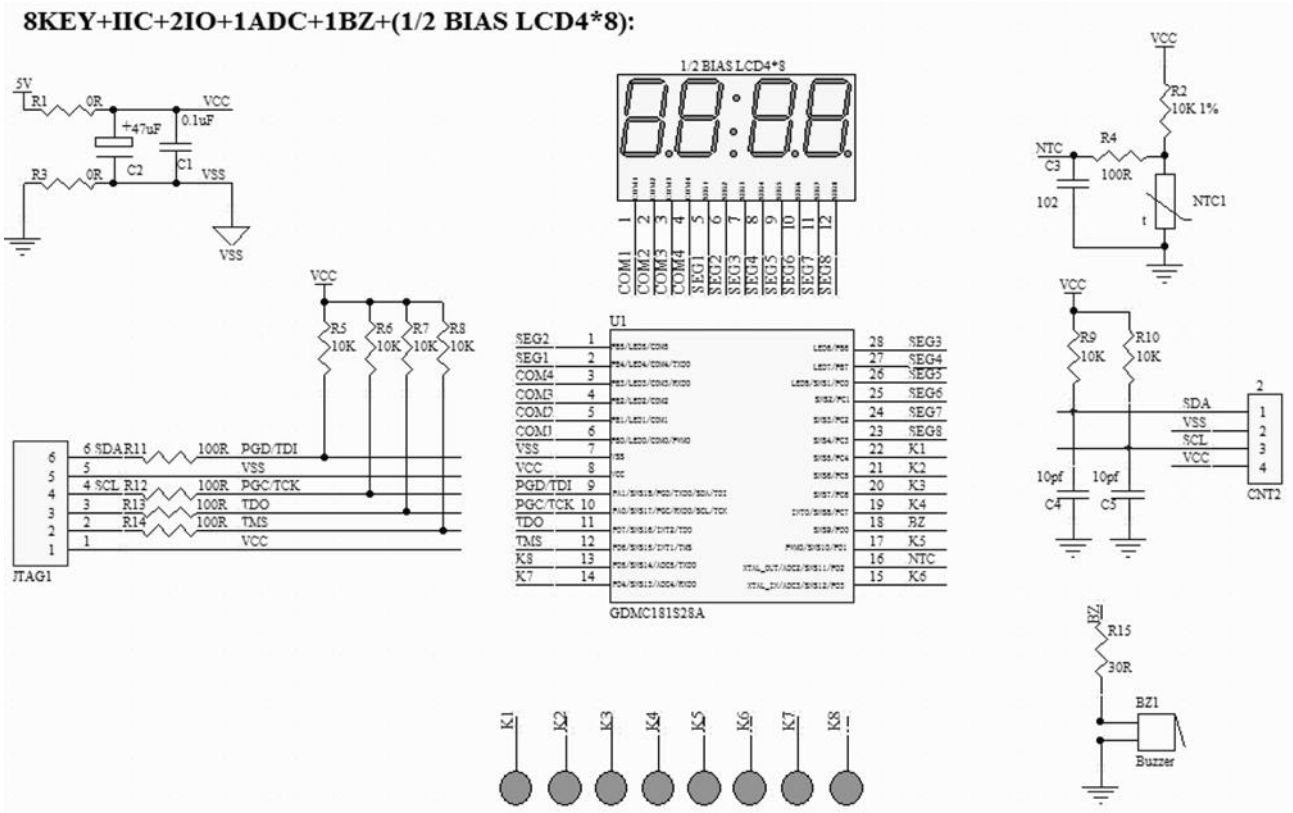
SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
COM_EN	保留	保留	COM_EN[5:0]					
POR	保留	保留	0	0	0	0	0	0

Bit[7:6]：保留

Bit[5:0]：对应的位为 1：使能 LCD COM 驱动功能；对应的位为 0：禁止使能 LCD COM 驱动功能;

将对应的 LCD COM 口设置为 IO 输出为高，IO 输出为低，IO 输入并开启对应的 LCD COM 驱动功能，软件控制输出 LCD 扫描驱动波形。

13.8、LCD 参考电路



LCD 参考电路

第 14 章、 LVDT

14.1、LVDT 相关寄存器

PD_ANA LVDT 控制寄存器 -- 0xDBh

位	功能	R/W	复位值
PD_ANA.7	保留	R/W	0
PD_ANA.5	PD_LVDT—LVDT功能使能控制，0：开启，1：关闭，默认为1。	R/W	1

PD_ANA.5: PD_LVDT—LVDT 功能使能控制，0：开启，1：关闭，默认为 1。

SEL_LVDT_VTH LVDT 低电检测配置寄存器 SEL_LVDT_VTH—0xDAh

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEL_LVDT_VTH	保留						SEL_LVDT_VTH[1:0]	
POR	0	0	0	0	0	0	0	0

SEL_LVDT_VTH[1:0]:设置低电压检测点:(0-2.7V)-(1-3.0V)-(2-3.6V)-(3-4.2V)。

IEN2 LVDT 中断使能寄存器---0xE7h

位	功能	R/W	复位值
IEN2.7-3	保留	aaaaa	00000
IEN2.1	EX9- INT_LVDT 允许位。EX9=0，禁止 INT_LVDT 申请中断。EX9=1，允许 INT_LVDT 申请中断。	R/W	0

IEN2.1：EX9- INT_LVDT 允许位。EX9=0，禁止 INT_LVDT 申请中断。EX9=1，允许 INT_LVDT 申请中断。

IRCON2 LVDT 中断标志位寄存器---0xE1h

位	功能	R/W	复位值
IRCON2.7-3	保留	aaaaa	00000
IRCON2.1	IE9-INT_LVDT 中断标志位。	R/W	0

RCON2.1： IE9-INT_LVDT 中断标志位。

INT_BOPO_STAT LVDT 升压/降压中断标志位寄存器---0xAFh

位	功能	R/W	复位值
INT_BOPO_STAT.7-3	保留	aaaaa	00000
INT_BOPO_STAT.1	LVDT 升压中断标志位。	R/W	0
INT_BOPO_STAT.0	LVDT 降压中断标志位。	R/W	0

INT_BOPO_STAT[1:0]：LVDT 检测升降压中断标志位。

14.2、LVDT 置流程

LVDT 低电压检测配置流程

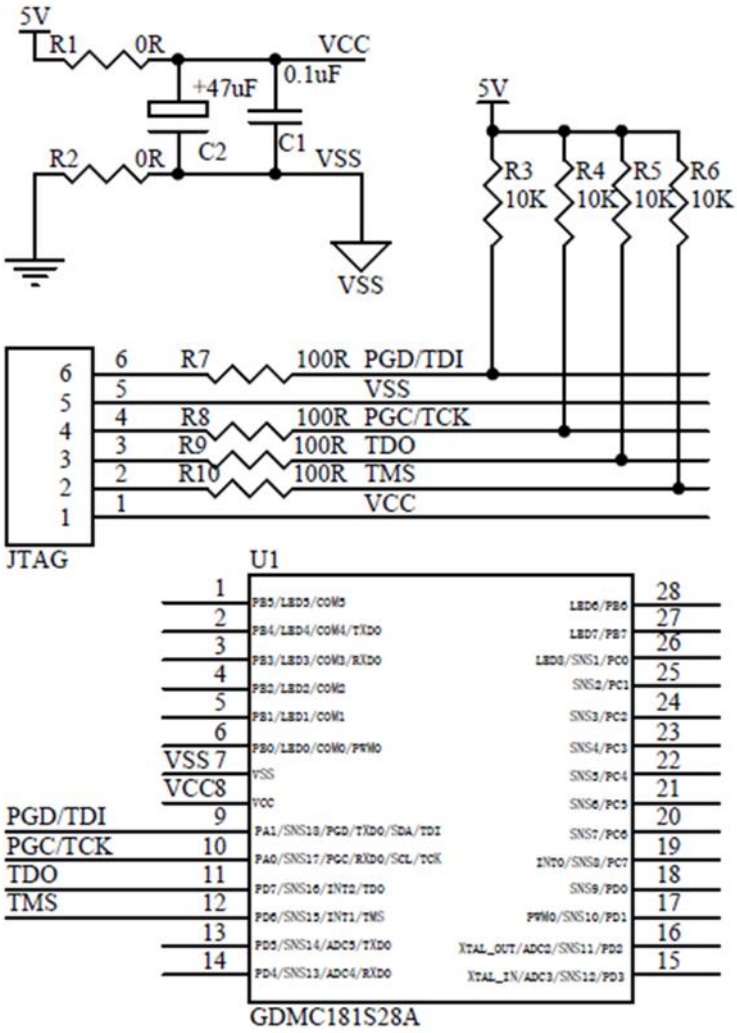
14.3、LVDT 初始化 C 语言参考

```
//-------------------------------------------------------------------//
//函数名称：  void LVDT_Init(void)
//函数功能：  降压升压检测初始化
//输入参数：  无
//输出参数：  无
//返 回 值：  无
//-------------------------------------------------------------------//
#if ((VolDet_EN == 1))
void LVDT_Init(void)
{
    EA_OFF;//关总中断；
    LVDT_IP_SET;//设置降压升压检中断优级为高
    LVDT_INT_FLAG_CLR;//清除低电压检测中断标志位
    LVDT_INT_DOWN_FLAG_CLR;//清除电压下降中断标志位
    LVDT_INT_UP_FLAG_CLR;//清除电压上升中断标志位 LVDT_ON;
    //开启低电压检测模块
    LVDT_ON;//开启降压升压检模块
    V_LVDT_SET(1);//设置低电压升检测
    //点:(0-2.7V)-(1-3.0V)-(2-3.6V)-(3-4.2V)
    LVDT_IE_SET;//使能降压升压检中断
    EA_ON;//开总中断
}
```

第 15 章、烧录调试

15.1 、JTAG 在线烧录调试

在进行仿真调试时，需要接 TDI、TCK、TMS、TDO、VCC、VSS 六根线，JTAG 调试模式下，JTAG 口的 IO 功能被屏蔽，建议不要操作 JTAG 调试 I/O 口，以免影响 JTAG 调试功能。烧录时仅接 TDI、TCK、VCC、VSS 四根线。



JTAG 连接示意图

15.2 、烧录调试

连接芯片 PGC、PGD、VCC、VSS 四根线，进入烧录界面时，选择对应型号的芯片，打开编译好的 HEX 文件，点击一键写 flash 等待烧录完成。

进入调试界面时，先烧录带调试数据发送模式的 HEX 文件，点击开始调试可以查看按键数据。

注：具体操作说明参照 TK 烧录调试指南。

第 16 章、指令系统

当 MCU 核运行时，以下几组寄存器需要注意使用：

PSW 寄存器-SFR D0h

位	功能
PSW.7	CY- 进位标志。当做加减运算时，有从最高位向更高位进位或由从更高位向最高位借位时，CY=1，否则 CY=0。在 CPU 进行移位操作时也会影响到 CY。
PSW.6	AC-辅助进位标志，又称为半进位标志。当进行加减运算时，有从 bit3 向 bit4 进位或有从 bit4 向 bit3 借位时，AC=1，否则 AC=0。
PSW.5	F0-用户标志位。该标志位由用户自己设定。一旦设定后，便可由用户程序检测，用以决定用户程序的流向。
PSW.4-3	RS1、RS0-寄存器选择位。在 8051 内部有四组工作寄存器，每组有八个 8 位的寄存器，分别命名为 R0,R1,….R7。这四组工作寄存器分别有自己的物理地址，并且利用 RS1 和 RS0 的编码来选择。具体选择下： RS1RS0　　R0~R7 组号各组 R0~R7 物理地址 00　　0　　00H~07H 01　　1　　08H~0FH 10　　2　　10H~17H 11　　3　　18H~1FH
PSW.2	OV-溢出标志位。当 CPU 进行算术运算产生溢出时，OV=1，否则 OV=0。即当累加器 A 中的运算结果小于-128 或大于+127 时，即发生溢出。
PSW.1	F1-用户标志位。该标志位又用户自己设定。一旦设定后，便可由用户程序检测，用以决定用户程序的流向。
PSW.0	P-奇偶标志位。若运算结果累加器 A 中 1 的个数为偶数，则 P=0，否则 P=1。

DPS 寄存器-SFR 86h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DPS	-							SEL

8051 增加了一组数据指针（DPTR1），地址为 84h 和 85h。当 SEL=0,指令使用的 DPTR 选择 DPL0 和 DPH0。当 SEL=1,指令使用的 DPTR 选择 DPL1 和 DPH1。

PCON 寄存器-SFR 86h

位	功能
PCON.7	保留
PCON.6	保留
PCON.5-4	保留，默认为 11
PCON. 3-2	GF1、GF0。通用标志位。
PCON.1	保留，必须写 0。
PCON.0	IDLE 位。当 IDLE=1 时,GDMC181SXXA 进入空闲模式;当 IDLE=0 时,GDMC181SXXA 不进入空闲模式。

SP 寄存器-SFR 86h

SFR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SP	SP<7~0>							

堆栈指针 SP。这是一个 8 位的寄存器。堆栈是在单片机内部 RAM 中定义的一个存储区域，用于临时存放数据。由于 8051 中 RAM 区的地址为 00H~7FH，堆栈只能在此范围内定义。堆栈操作总是遵循“先进后出”的原则。堆栈指针 SP 总是指向堆栈的顶。当栈中无数据时，栈顶与栈底重合。

16.1、指令编码

GDMC181SXXA 的指令分为单字节指令、双字节指令和三字节指令。

- 单字节指令：单字节指令由 8 位二进制编码构成。指令中只有指令操作码，没有指令操作数或者至指令操作数隐含于指令操作码。该类指令共有 49 条。
- 双字节指令：双字节指令由两个字节构成，一个为操作码，另一个为操作数(或操作数的地址)，在程序存储器中按顺序存放。该类指令共有 46 条。
- 三字节指令：三字节指令由一个字节的指令操作码和两个字节的操作数(或操作数的地址)构成。该类指令共有 16 条。

16.2、指令集

为了描述指令方便，在指令中使用了一些符号，这些符号的含义说明如下：

addr 11	低 11 位地址
addr 16	16 位地址
direct	直接寻址，8 位内部数据及地址(包括特殊功能寄存器)
bit	位地址
#data	8 位立即数
#data16	16 位立即数
rel	带符号的 8 位相对位移量
n	数字 0~7
Rn	当前寄存器组的 R0~R7 工作寄存器
i	数字 0、1
Ri	工作寄存器 R0、R1

@	寄存器间接寻址
←	数据传送方向
∧	逻辑“与”
∨	逻辑“或”
⊕	逻辑“异或”
√	对标志位有影响
×	对标志位无影响

提供使用的汇编指令、各指令的功能、占用的字节数、指令执行周期以及对相应标志位的影响如下表所示：

8 位数据传送类指令：

助记符		功能	对标志位影响				字节数	周期数
			P	OV	AC	CY		
MOV A	Rn	A←(Rn)	√	×	×	×	1	1
	direct	A←(direct)	√	×	×	×	2	1
	@Ri	A←((Ri))	√	×	×	×	1	1
	#data	A←data	√	×	×	×	2	1
MOV Rn	A	Rn←(A)	×	×	×	×	1	1
	direct	Rn←(direct)	×	×	×	×	2	2
	#data	Rn←data	×	×	×	×	2	1
MOV direct1,	A	direct1←(A)	×	×	×	×	2	1
	Rn	direct1←(Rn)	×	×	×	×	2	2
	direct2	direct1←(direct2)	×	×	×	×	3	2
MOV direct,	@Ri	direct←((Ri))	×	×	×	×	2	2
	#data	direct←data	×	×	×	×	3	2
MOV @Ri	A	(Ri)←(A)	×	×	×	×	1	1
	direct	(Ri)←(direct)	×	×	×	×	2	2
	#data	(Ri)←data	×	×	×	×	2	1

16 位数据传送类指令：

助记符	功能	对标志位影响				字节数	周期数
		P	OV	AC	CY		
MOV DPTR,#data16	DPTR←data16	×	×	×	×	3	2

外部数据传送与查表类指令：

助记符	功能	对标志位影响				字节数	周期数
		P	OV	AC	CY		
MOVX @DPTR,A	(DPTR)←(A)	×	×	×	×	1	2

MOVC A,	@A+DPTR	A←((A)+(DPTR))	√	×	×	×	1	2
	@A+PC	A←((A)+(PC))	√	×	×	×	1	2
MOVX A,	@DPTR	A←(DPTR)	√	×	×	×	1	2

注：MOVX 指令的周期数以及字节数可通过寄存器 CKCON<2:0>配置。

交换类指令：

助记符		功能	对标志位影响				字节数	周期数
			OV	AC		CY		
XCH A,	Rn	(Rn)←(A)	√	×	×	×	1	2
	direct	(A)←(direct)	√	×	×	×	2	1
	@Ri	(A)←((Ri))	×	×	×	×	1	1
XCHD A,@Ri		(A)3~0~((Ri))3~0	√	×	×	×	1	1
SWAP A		(A)7-4~(A)3-0	√	×	×	×	1	1

算术运算类指令：

助记符		功能	对标志位影响				字节数	周期数
			P	OV	AC	CY		
ADD A,	Rn	A←(A)+(Rn)	√	√	√	√	1	1
	direct	A←(A)+(direct)	√	√	√	√	2	1
	@Ri	A←(A)+((Ri))	√	√	√	√	1	1
	#data	A←(A)+data	√	√	√	√	2	1
ADDC A,	Rn	A←(A)+(Rn)+(C)	√	√	√	√	1	1
	direct	A←(A)+(direct) +(C)	√	√	√	√	2	1
	@Ri	A←(A)+((Ri)) +(C)	√	√	√	√	1	1
	#data	A←(A)+data+(C)	√	√	√	√	2	1
INC	A	A←(A)+1	√	×	×	×	1	1
	Rn	Rn←(Rn)+1	×	×	×	×	1	1
	direct	direct←(direct)+1	×	×	×	×	2	1
	@Ri	(Ri)←((Ri))+1	×	×	×	×	1	1
	DPTR	DPTR←((DPTR))+1	×	×	×	×	1	2
DA A		BCD 码调整	√	×	√	√	1	1
SUBB A	Rn	A←(A)-(Rn)-(C)	√	×	×	×	1	1
	direct	A←(A)-(direct)-(C)	√	√	√	√	2	1
	@Ri	(A)←(A)-((Ri))-(C)	√	√	√	√	1	1
	#data	A←(A)-data-(C)	√	√	√	√	2	1
DEC	A	A←(A)-1	√	×	×	×	1	1
	Rn	Rn←(Rn)-1	×	×	×	×	1	1
	direct	direct←(direct)-1	×	×	×	×	2	1
	@Ri	(Ri)←((Ri))-1	×	×	×	×	1	1
MUL AB		BA←(A)*(B)，执行乘法	√	√	×	0	1	4

	运算后，结果低字节存于 A，高字节存于 B						
DIV AB	A←(A)/(B) B←余数	√	√	×	0	1	4

注：DA 指令使用时，，调整规则如下：若累加器 A 低 4 位大于 9 或者 AC=1，则 A←A+06H;若累加器 A 高 4 位大于 9 或者 CY=1，则 A←A+60H。

逻辑运算类指令

助记符		功能	对标志位影响				字节数	周期数
			P	OV	AC	CY		
CLR A		A←00H	√	×	×	×	1	1
CPL A		A←$\overline{(A)}$	√	×	×	×	1	1
ANL A,	Rn	A←(A)∧(Rn)	√	×	×	×	1	1
	direct	A←(A)∧(direct)	√	×	×	×	2	1
	@Ri	A←(A)∧((Ri))	√	×	×	×	1	1
	#data	A←(A)∧data	√	×	×	×	2	1
ANL direct,	A	direct←(A)∧(direct)	×	×	×	×	2	1
	#data	direct←(direct)∧data	×	×	×	×	3	2
ORL A,	Rn	A←(A)∨(Rn)	√	×	×	×	1	1
	direct	A←(A)∨(direct)	√	×	×	×	2	1
	@Ri	A←(A)∨((Ri))	√	×	×	×	1	1
	#data	A←(A)∨data	√	×	×	×	2	1
ORL direct,	A	direct←(direct)∨(A)	×	×	×	×	2	1
	#data	direct←(direct)∨data	×	×	×	×	3	2
XRL A,	Rn	A←(A)⊕(Rn)	√	×	×	×	1	1
	direct	A←(A)⊕(direct)	√	×	×	×	2	1
	@Ri	A←(A)⊕((Ri))	√	×	×	×	1	1
	#data	A←(A)⊕data	√	×	×	×	2	1
XRL direct,	A	direct←(direct)⊕(A)	×	×	×	×	2	1
	#data	direct←(direct)⊕data	×	×	×	×	3	2

循环、移位类指令

助记符	功能	对标志位影响				字节数	周期数
		P	OV	AC	CY		

RL A	A 中内容循环左移一位	×	×	×	×	1	1
RLC A	A 中内容带进位循环左移一位	√	×	×	√	1	1
RR A	A 中内容循环右移一位	×	×	×	×	1	1
RRC A	A 中内容带进位循环右移一位	√	×	×	√	1	1

调用、返回类指令

助记符	功能	对标志位影响				字节数	周期数
		P	OV	AC	CY		
LCALL addr16	(PC)←(PC)+3, (SP)←(PC), (PC)←addr16	×	×	×	×	3	2
ACALL addr11	(PC)←(PC)+2, (SP)←(PC), ($PC_{10\sim0}$)←addr11	×	×	×	×	2	2
RET	(PC)←((SP))	×	×	×	×	1	2
RETI	(PC)←((SP))从中断返回	×	×	×	×	1	2

转移类指令

助记符	功能	对标志位影响				字节数	周期数
		P	OV	AC	CY		
LJMP addr16	PC←$addr_{15\sim0}$	×	×	×	×	3	2
AJMP addr11	$PC_{10\sim0}$←$addr_{10\sim0}$	×	×	×	×	2	2
SJMP rel	PC←(PC)+rel	×	×	×	×	2	2
JMP @A+DPTR	PC←(A)+(DPTR)	×	×	×	×	1	2
JZ rel	PC←(PC)+2, 若(A)=0, PC←(PC)+rel	×	×	×	×	2	2
JNZ rel	PC←(PC)+2, 若(A)≠0, PC←(PC)+rel	×	×	×	×	2	2
JC rel	PC←(PC)+2, 若(CY)=1, PC←(PC)+rel	×	×	×	×	2	2
JNC rel	PC←(PC)+2, 若(CY)=0, PC←(PC)+rel	×	×	×	×	2	2
JB bit,rel	PC←(PC)+3, 若(bit)=1, PC←(PC)+rel	×	×	×	×	3	2
JNB bit,rel	PC←(PC)+3, 若(bit)=0, PC←(PC)+rel	×	×	×	×	3	2
JBC bit,rel	PC←(PC)+3, 若(bit)=1, 则 bit←0, PC←(PC)+rel	×	×	×	×	3	2
CJN A, direct,rel	PC←(PC)+3,	×	×	×	×	3	2

E		若(A) ≠direct 则 PC(PC)+rel 若(A)<(direct),则 CY←1						
	A,#data,rel	PC←(PC)+3, 若(A) ≠data 则 PC(PC)+rel 若(A)<(data),则 CY←1	×	×	×	×	3	2
	Rn,#data,rel	PC←(PC)+3, 若(Rn) ≠data 则 PC←(PC)+rel 若(Rn)<(data),则 CY←1	×	×	×	×	3	2
	@Ri,#data,rel	PC←(PC)+3,若((Ri)) ≠ data 则 PC←(PC)+rel 若((Ri))<(data),则 CY←1	×	×	×	×	3	2
DJNZ	Rn,rel	PC←(PC)+2,Rn←(Rn)-1, 若(Rn) ≠0,则 PC←(PC)+rel	×	×	×	×	2	2
	direct,rel	PC←(PC)+3, (direct)←(direct)-1, 若(direct) ≠0, 则 PC←(PC)+rel	×	×	×	×	3	2

堆栈、空操作类指令

助记符	功能	对标志位影响				字节数	周期数
		P	OV	AC	CY		
PUSH direct	SP←(SP)+1,(SP)←(direct)	×	×	×	×	2	2
POP direct	direct←(SP),SP←(SP)-1	×	×	×	×	2	2
NOP	空操作	×	×	×	×	1	1

位操作类指令

助记符		功能	对标志位影响				字节数	周期数
			P	OV	AC	CY		
MOV	C,bit	CY←bit	×	×	×	√	2	1
	bit,C	bit←CY	×	×	×	×	2	1
CLR	C	CY←0	×	×	×	√	1	1
	bit	bit←0	×	×	×	×	2	1
SETB	C	CY←1	×	×	×	√	1	1
	bit	bit←1	×	×	×	×	2	1
CPL	C	CY←$(\overline{CY})$	×	×	×	√	1	1
	bit	bit←$(\overline{bit})$	×	×	×	×	1	1
ANL	C,bit	C←(C)∧(bit)	×	×	×	√	2	2
	C ,/bit	C←(C)∧$(\overline{bit})$	×	×	×	√	2	2

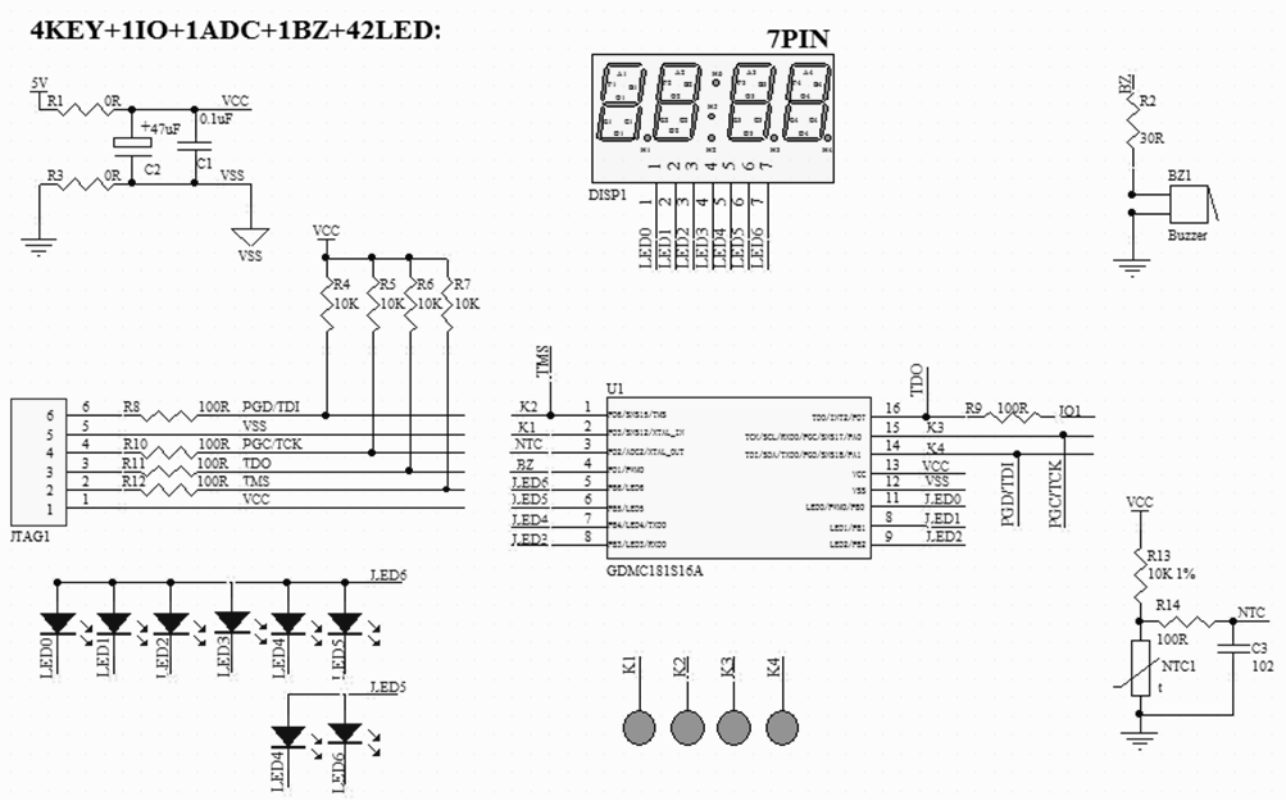
ORL	C,bit	C←(C)∨(bit)	×	×	×	√	2	2
	C,/bit	C←(C)∨($\overline{bit}$)	×	×	×	√	2	2

伪指令

助记符	指令格式	功能说明
ORG	【标号：】ORG addr16	规定标号的起始地址
EQU	标号 EQU 数值或标号	为标号赋值
DB	【标号：】 DB 项或项表	用于定义内存一个单元或一批单元的字节内容
DW	【标号：】 DW 项或项表	用于定义内存某两单元或多个两个单元构成的 16 位字内容
DS	【标号：】DS 表达式	规定从标号开始留下若干个存储单元
BIT	标号 BIT 位地址	把位地址赋给标号
END	END 放在汇编语言程序的最后，用以告诉汇编程序，源程序到此为止。没有 END 结束的源程序将进入死循环	

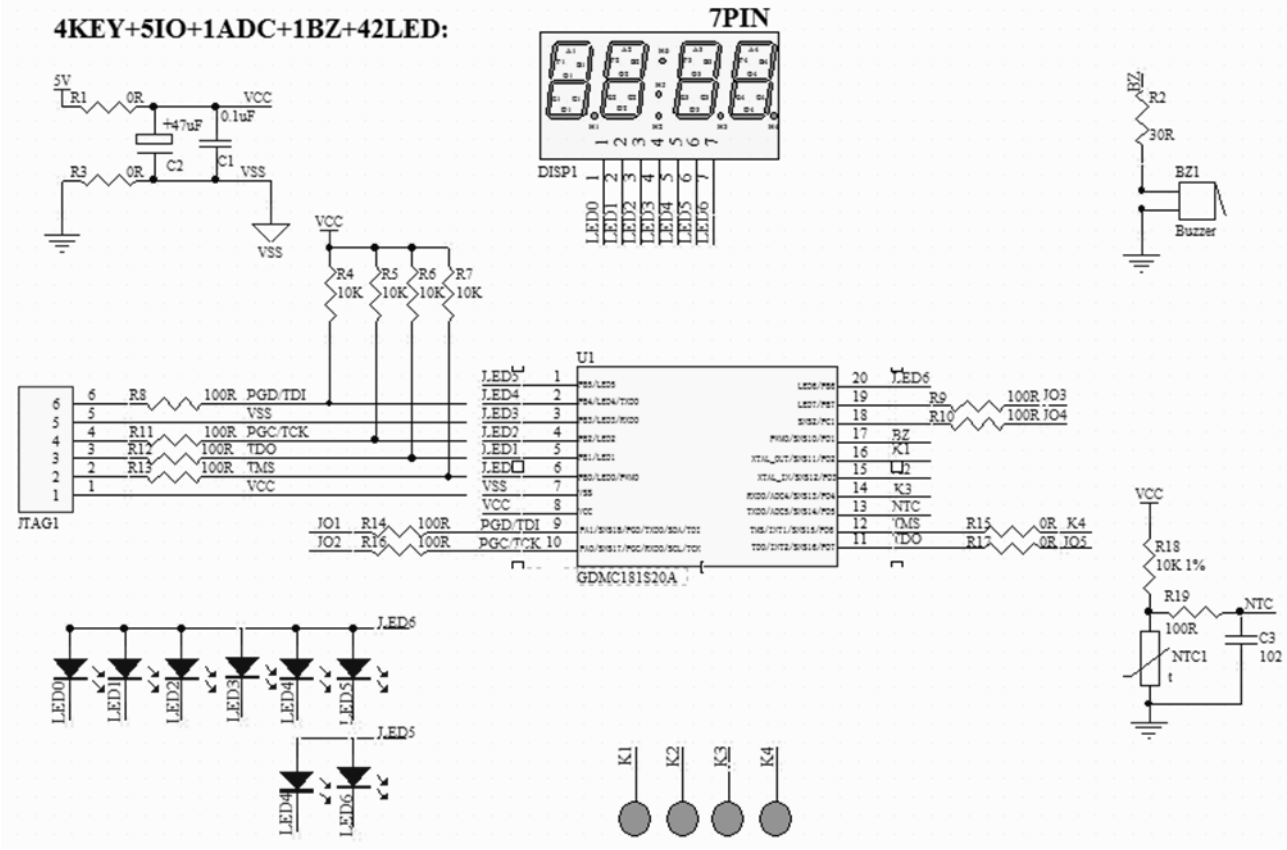
第 17 章、参考应用电路

17.1、GDMC181S16A 参考电路



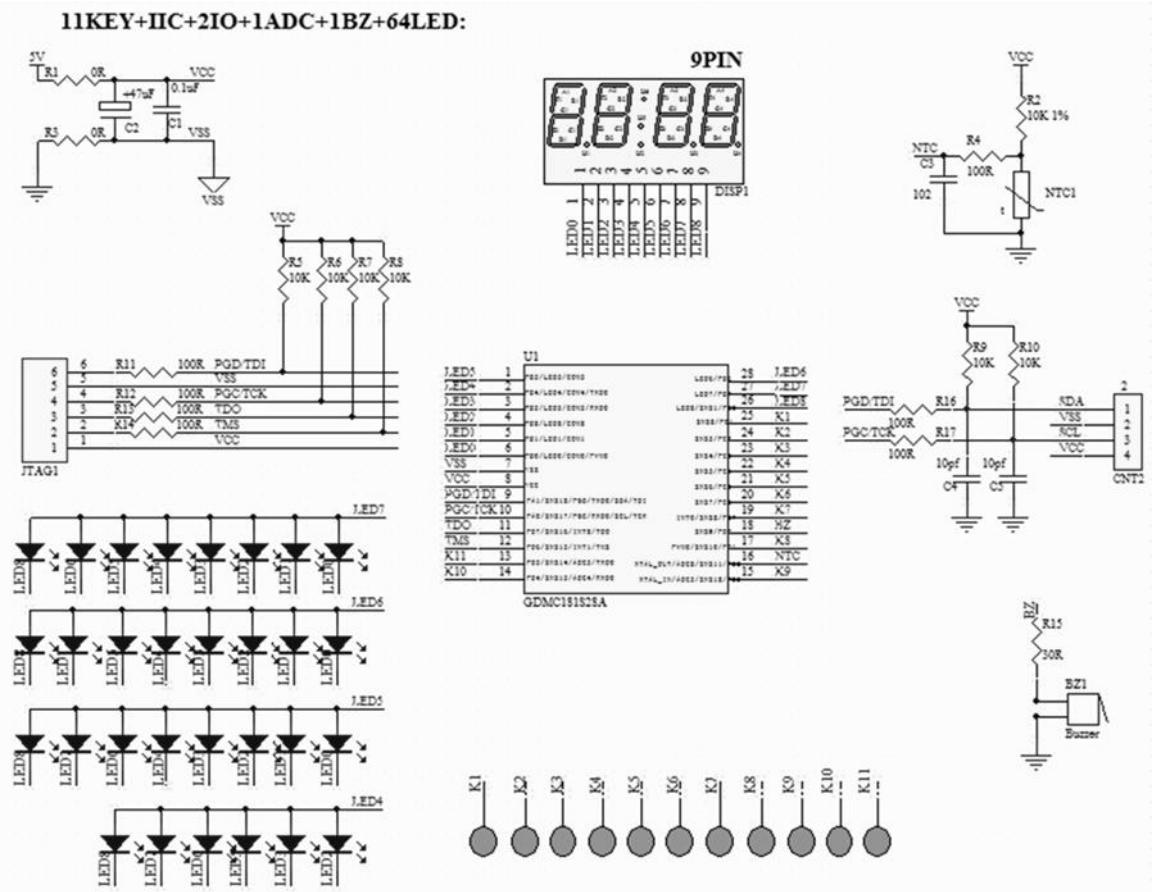
GDM181S16A 参考电路

17.2、GDMC181S20A 参考电路



GDM181S20A 参考电路

17.3、GDMC181S28A 参考电路

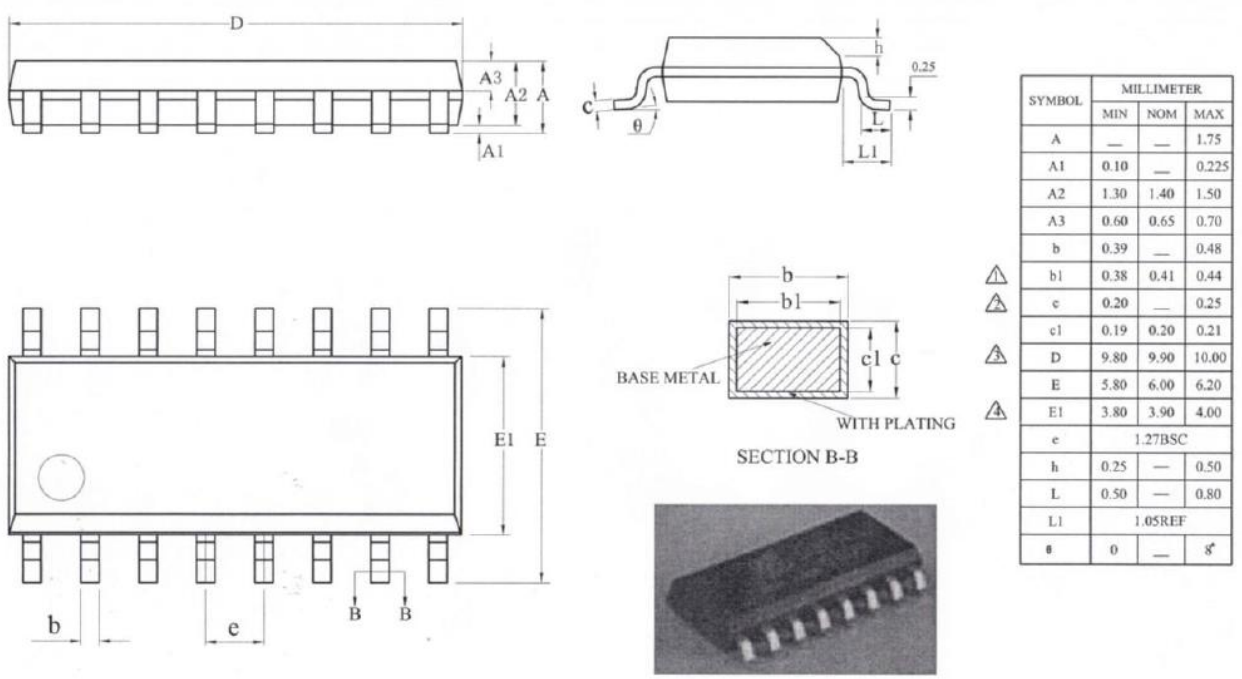


GDM181S28A 参考电路

注： 7PIN/8PIN/9Pin 数码管为定制数码管

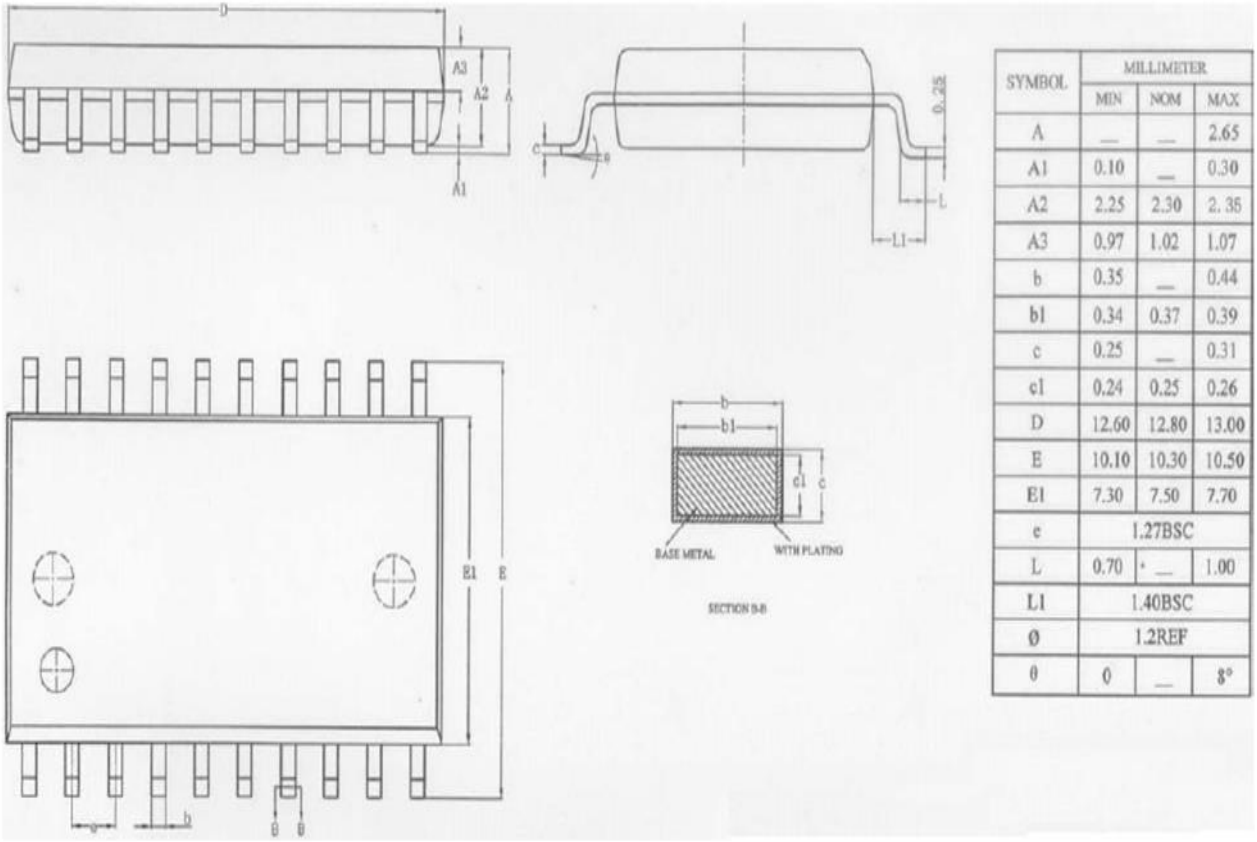
第 18 章、封装

18.1、SOP16-(天水华天)



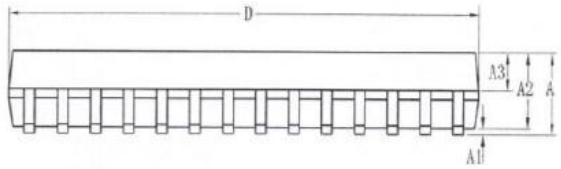
	SYMBOL	MILLIMETER		
		MIN	NOM	MAX
	A	—	—	1.75
	A1	0.10	—	0.225
	A2	1.30	1.40	1.50
	A3	0.60	0.65	0.70
	b	0.39	—	0.48
1	b1	0.38	0.41	0.44
2	c	0.20	—	0.25
	c1	0.19	0.20	0.21
3	D	9.80	9.90	10.00
	E	5.80	6.00	6.20
4	E1	3.80	3.90	4.00
	e	1.27BSC		
	h	0.25	—	0.50
	L	0.50	—	0.80
	L1	1.05REF		
	θ	0	—	8°

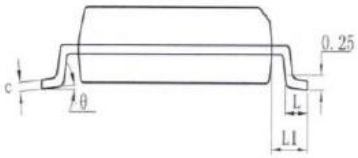
18.2、SOP20-(天水华天)



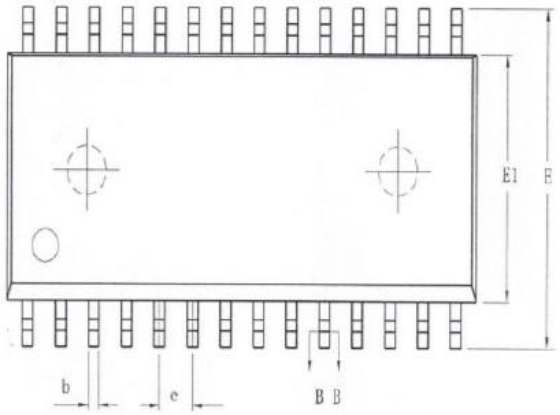
SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	2.65
A1	0.10	—	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
b	0.35	—	0.44
b1	0.34	0.37	0.39
c	0.25	—	0.31
c1	0.24	0.25	0.26
D	12.60	12.80	13.00
E	10.10	10.30	10.50
E1	7.30	7.50	7.70
e	1.27BSC		
L	0.70	—	1.00
L1	1.40BSC		
Ø	1.2REF		
θ	0	—	8°

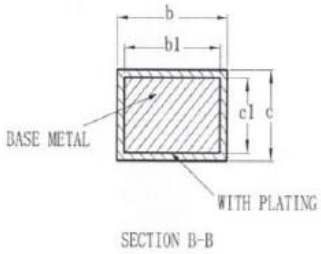
18.3、SOP28-(天水华天)

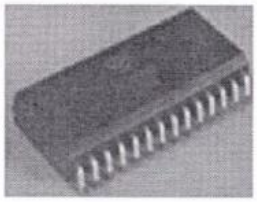




SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	2.65
A1	0.10	—	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
b	0.39	—	0.48
b1	0.38	0.41	0.43
c	0.25	—	0.31
c1	0.24	0.25	0.26
D	17.80	18.00	18.20
E	10.10	10.30	10.50
E1	7.30	7.50	7.70
e	1.27BSC		
L	0.70	—	1.00
L1	1.40BSC		
θ	0	—	8°







改版记录

改版日期	改版内容	改版人	备注
2018.04.02	初版	Xs，Zds	V1.0
2018.04.24	1、增加空闲模式，多按键唤醒描述； 2、JTAG 描述； 3、更新 Timer2 描述； 4、更新 PU_A 描述；	Xs，Zds	V1.1
2018.05.03	1. 更新 TCON、REG_DATA、PU_PC、EXT_INT_CON、ADC_SPT、SNS_SCAN_CFG1、UART_BUF 上电复位默认值描述； 2. 更新串口参考程序； 3. 更新选型列表；	Xs	V1.2
2018.05.09	1. 更新选型列表； 2. 更新 PA0/PA1 为 SNS 功能及 ADC 引脚资源描述； 3. 增加串口 8,N,1 模式，主机发送建议；	Xs	V1.3
2018.05.28	1. 修正 SEG_SEL_7/8 地址描述; 2. 更新 LED 驱动能力描述、增加 LED 单个灯平均电流描述； 3. 增加 VTH 设置建议描述； 4. 更新 Flash 擦写次数；	Xs	V1.4
2018.06.11	1、修正位宽校准时间公式描述； 2、增加 LED 驱动关于 LED 正向导通压降选择的建议； 3、更新 JTAG 烧录描述；	Xs	V1.5
2018.06.28	1. 增加 ADC 应用注意事项；	Xs	V1.6

免责声明

1、此文档中的信息可以在不通知用户时进行修改及更新
2、公司将竭尽最大的努力保证本公司产品的高质量与高稳定性。尽管如此，
由于一般半导体器件的电气敏感性及易受到外部物理伤害等固有特点，本公司产品有可能在这些情况下出现故障或失效。当使用本公司产品时，使用者有责任遵从安全规则来设计一个安全及稳定的系统环境。使用者可通过去除多余器件、故障预防及火灾预防等措施来避免可能发生的意外、火灾及公共伤害。在用户使用该产品时，请遵从本公司最新说明书上规定的操作步骤来使用该产品。
3、在此文档中的本公司的产品是为一般电气应用（电脑、个人工具、办公司
工具、测量工具、工业机械器件、家用电器等）所设计的。本公司该产品不能及禁止应用在一些需要极高稳定性及质量的特殊设备上，以免导致人员伤亡等意外发生。产品不能应用范围包括原子能控制设备、飞机及航空器件、运输设备、交通信号设备、燃烧控制设备、医药设备以及所有安全性设备等等。使用者在以上列举的非产品应用范围内使用时造成的损失与伤害，本公司概不负责。