

# QT8S737

**8BIT  
TK+AD 型  
MTP MCU**

**Version 1.06**

2020年6月

本公司不承担由本手册所涉及的产品或电路的运用和使用所引起的任何责任，本公司的产品不是专门设计来应用于外科植入、生命维持和任何本公司产品的故障会对个体造成伤害甚至死亡的领域。如果将本公司的产品应用于上述领域，即使这些是由本公司在产品设计和制造上的疏忽引起的，用户应赔偿所有费用、损失、合理的人身伤害或死亡所直接或间接产生的律师费用，并且用户保证本公司及其雇员、子公司、分支机构和销售商与上述事宜无关。

### 修正记录

版本	日期	描述
V1.00	2018-11-11	初版
V1.01	2018-12-25	修改波特率设置说明
V1.02	2019-3-5	增加 EE
V1.03	2019-3-11	勘误
V1.04	2019-7-10	增加指令集简述
V1.05	2019-9-10	增加模块功能范例
V1.06	2020-6-2	更新电性参数已经增加封装信息

## 目录

<b>1 产品简述</b> .....	<b>7</b>
1.1 特性.....	7
1.2 引脚图.....	8
1.2.1 SOP20.....	8
1.2.2 SOP16.....	8
1.3 引脚描述.....	9
<b>2 中央处理器（CPU）</b> .....	<b>11</b>
2.1 程序存储器.....	11
2.1.1 复位向量（0000H）.....	11
2.1.2 中断向量（0008H）.....	12
2.1.3 查表.....	12
2.2 EEPROM.....	14
2.2.1 EECON1 状态寄存器.....	14
2.2.2 EECON2 状态寄存器.....	14
2.3 数据存储器.....	16
2.3.1 数据存储器结构.....	16
2.3.2 数据存储器寻址模式.....	16
2.3.3 系统寄存器定义.....	17
2.3.4 INDF0 间接寻址寄存器 0.....	17
2.3.5 INDF1 间接寻址寄存器 1.....	17
2.3.6 FSR0 间接寻址指针 0.....	17
2.3.7 FSR1 间接寻址指针 1.....	17
2.3.8 HBUF 查表数据高 8 位.....	18
2.3.9 PCL 程序计数器指针低位.....	18
2.3.10 STATUS 状态寄存器.....	18
<b>3 复位</b> .....	<b>19</b>
3.1 复位方式.....	19
<b>4 系统时钟</b> .....	<b>20</b>
4.1 概述.....	20
4.2 OSCM 寄存器.....	20
4.3 系统时钟的工作模式.....	21
4.4 IRCCAL 寄存器.....	21
4.5 系统时钟结构框图.....	22
4.6 系统时钟高低频切换.....	23
<b>5 中断</b> .....	<b>24</b>
5.1 概述.....	24
5.2 OPTION 配置寄存器.....	24
5.3 IO 口变化中断使能寄存器.....	25

5.4 INTCR0 中断控制寄存器 0 .....	25
5.5 INTF0 中断标志寄存器 0 .....	26
5.6 INTCR1 中断控制寄存器 1 .....	27
5.7 INTF1 中断标志寄存器 1 .....	27
5.8 中断范例 .....	28
<b>6 端口.....</b>	<b>30</b>
6.1 IOA.....	30
6.2 IOB .....	30
6.3 IOC .....	32
6.4 IOIDS 端口驱动设置寄存器.....	33
<b>7 定时器 0/1(TC0/1).....</b>	<b>35</b>
7.1 概述 .....	35
7.2 TxCR 控制寄存器 .....	36
7.3 TCxCL TCx 计数器低 8 位/周期寄存器.....	36
7.4 TCxCH TCx 计数器高位 .....	37
7.5 定时器范例 .....	37
<b>8 脉宽调制模块 PWM0.....</b>	<b>39</b>
8.1 概述 .....	39
8.2 PWM0CR 控制寄存器 .....	39
8.3 PWM0D 数据高位.....	39
8.4 PWM0 范例 .....	40
<b>9 脉宽调制模块 PWM1.....</b>	<b>41</b>
9.1 概述 .....	41
9.2 PWM1CR 控制寄存器 .....	41
9.3 PWM1S 控制寄存器 .....	42
9.4 PWM1DH 数据高位.....	42
9.5 PWM1DL 数据低位 .....	42
9.6 PWMDEADT PWM 死区控制寄存器 .....	43
9.7 8+4 位分辨率模式.....	44
9.8 单脉冲模式 .....	45
9.9 PWM 输出波形示例.....	47
9.9.1 互补 PWM 输出 .....	47
9.9.2 带死区的互补 PWM 输出 .....	47
9.9.3 有效电平选择.....	48
9.10 PWM1 范例 .....	48
<b>10 通用串行通讯口 (USART) .....</b>	<b>50</b>
10.1 概述 .....	50
10.2 TXCR 发送控制寄存器 .....	50
10.3 TXREG 发送数据寄存器 .....	50
10.4 RXCR 接收控制寄存器 .....	51

10.5 RXREG 接收数据寄存器 .....	51
10.6 BRGDH 波特率寄存器高位 .....	51
10.7 BRGDL 波特率寄存器低位 .....	52
10.8 USART 使用说明 .....	52
10.8.1 波特率设置 .....	52
10.8.2 异步发送 .....	52
10.8.3 异步接收 .....	55
10.8.4 同步发送 .....	57
10.8.5 同步接收 .....	58
10.8.6 唤醒及休眠模式下通讯 .....	59
<b>11 串行通讯口 (I2C) .....</b>	<b>60</b>
11.1 概述 .....	60
11.2 通讯波形示意 .....	60
11.3 I2CCON I2C 控制寄存器 .....	61
11.4 I2ADDR I2C 控制寄存器 .....	61
11.5 I2CBUF 数据寄存器 .....	62
11.6 唤醒及休眠模式下通讯 .....	62
11.7 通讯波形图 .....	62
11.8 应用示例 .....	64
11.8.1 从机软件流程图 .....	64
11.8.2 例程 .....	64
<b>12 触摸按键 (CDC) .....</b>	<b>67</b>
12.1 概述 .....	67
12.2 原理框图 .....	67
12.3 TKCTR0 控制寄存器 0 .....	68
12.4 TKCTR1 控制寄存器 1 .....	68
12.5 TKCHSH/L 触摸按键通道选择寄存器 .....	69
12.6 TKCNTH/L 触摸按键计数寄存器 .....	69
12.7 操作说明 .....	70
<b>13 模数转换器(ADC).....</b>	<b>71</b>
13.1 概述 .....	71
13.2 ADCON0 寄存器 .....	71
13.3 ADCON1 寄存器 .....	72
13.4 ADCON2 寄存器 .....	72
13.4.1 ADC 调校功能 .....	73
13.4.2 ADC 小信号采集 .....	73
13.4.3 ADC 差分测量参考 .....	73
13.5 ADH ADC 数据高位 .....	74
13.6 ADL ADC 数据低位 .....	74
13.7 AD 转换时间 .....	75
13.8 ADC 范例 .....	75
<b>14 看门狗 (WDT) .....</b>	<b>77</b>

---

14.1 概述 .....	77
14.2 OPTION 配置寄存器 .....	77
14.3 WDTC 看门狗控制寄存器 .....	77
<b>15 芯片配置字 (OPTION BIT) .....</b>	<b>78</b>
<b>16 电性参数 .....</b>	<b>80</b>
16.1 极限参数 .....	80
16.2 直流特性 .....	80
16.3 温度对系统时钟的影响 .....	82
16.4 IO 口拉灌电流特性曲线 .....	82
16.5 ADC 参考电压随温度变化曲线 .....	83
<b>17 封装信息 .....</b>	<b>84</b>
17.1 SOP20 .....	84
17.2 SOP16 .....	85
<b>18 指令集简述 .....</b>	<b>86</b>
18.1 概述 .....	86
18.2 符号说明 .....	86
18.3 QT8Sxxxx 指令集表 .....	87

# 1 产品简述

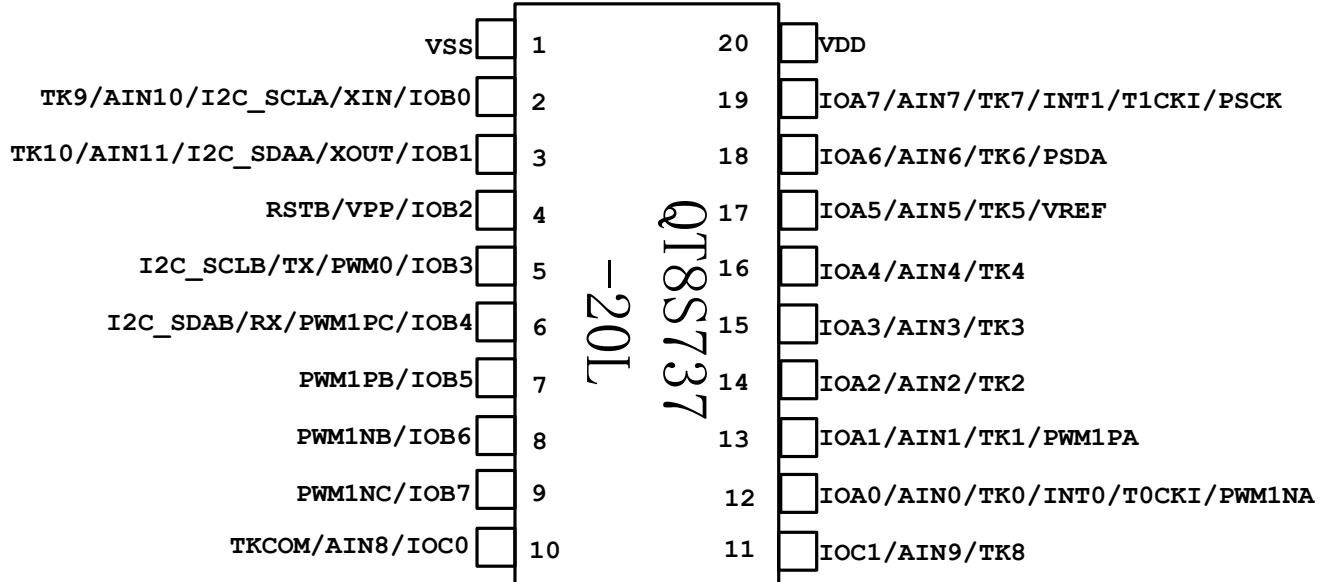
- QT8S737是一颗采用高速低功耗CMOS工艺设计开发的8位高性能精简指令单片机，内部有2K×16位多次擦写编程存储器（MTP，擦写次数1000次），128×8位EEPROM（擦写次数10000次），128×8位的数据存储器（RAM），18个双向I/O口，2个8/16位Timer定时器/计数器，1路UART，1路IIC SLAVE，1路8位PWM，1路8+4位分辨率的互补PWM，12+3路12位AD转换器，11路触摸按键，支持多种系统工作模式和多个中断源。

## 1.1 特性

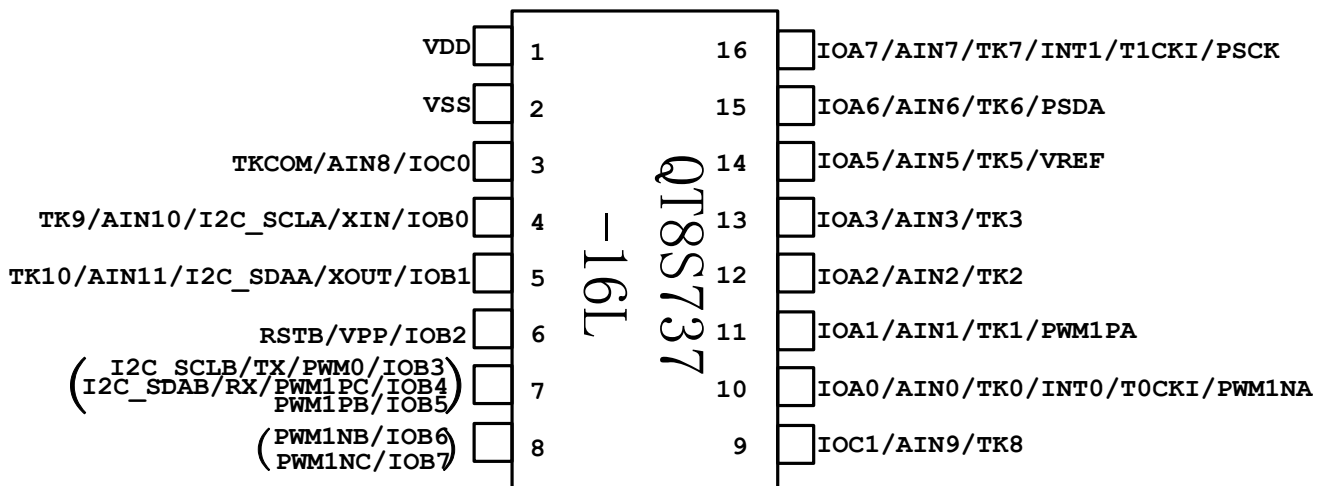
- CPU 特性
  - 高性能精简指令
  - 2K×16位的MTP程序存储器
  - 128×8位的EEPROM
  - 128×8位的数据存储器
  - 8级堆栈缓存器
  - 支持查表指令
- I/O 口
  - 最多18个双向I/O口
  - 所有端口可设置弱上拉
  - IOB端口可设置弱下拉
  - IOB口变化中断
  - 两路外部中断
- 2个定时器/计数器
  - TC0/TC1：具有自动装载功能的定时/计数器
- 2路 PWM
  - PWM0：8位分辨率PWM
  - PWM1：8+4位分辨率PWM，带死区控制及互补输出
- 系统时钟
  - 内部高速RC震荡器：16MHz
  - 内部低速RC震荡器：32KHz/500KHz
  - 外部高速晶体震荡器：500KHz -20MHz
  - 外部低速晶体振荡器：0-500KHz
- 系统工作模式
  - 普通模式：高低速时钟同时工作
  - 绿色模式：仅低速时钟工作
  - 休眠模式：高低速时钟都停止工作
- 12+3路12位ADC
  - 内嵌参考电压2V、3V、4V、VDD
  - 12路外部输入
  - 1路内部电源电压检测VDD/4
  - 1路内部GND电压检测
  - 1路内部参考电压检测
- 触摸按键模块
  - 休眠电流小于10uA
- 多路中断/唤醒源
  - 定时器中断：TC0/TC1
  - IOB口变化中断
  - 触摸按键中断
  - ADC转换中断
  - 外部中断
- 看门狗定时器
- 特殊功能
  - 可编程代码保护
  - ISP功能
- 封装形式
  - SOP20
  - SOP16

## 1.2 引脚图

### 1.2.1 SOP20



### 1.2.2 SOP16



注：引脚 7 是 IOB3、IOB4、IOB5 内部是短接在一起的，引脚 8 是 IOB6、IOB7 内部是短接在一起的，如果用到其中一个 IO 时，另外的 IO 用做输入。



### 1.3 引脚描述

名称	类型	说明
VDD, VSS	P	电源输入端
IOA0	I/O	输入/输出 IO, SMT, 上拉电阻
AIN0	A	AD 通道 0
TK0	A	触摸按键通道 0
INT0	I	外部中断
T0CKI	I	TC0 外部时钟输入
PWM1NC	O	PWM1 反相输出
IOA1	I/O	输入/输出 IO, SMT, 上拉电阻
AIN1	A	AD 通道 1
TK1	A	触摸按键通道 1
PWM1PC	O	PWM1 正相输出
IOA2	I/O	输入/输出 IO, SMT, 上拉电阻
AIN2	A	AD 通道 2
TK2	A	触摸按键通道 2
IOA3	I/O	输入/输出 IO, SMT, 上拉电阻
AIN3	A	AD 通道 3
TK3	A	触摸按键通道 3
IOA4	I/O	输入/输出 IO, SMT, 上拉电阻
AIN4	A	AD 通道 4
TK4	A	触摸按键通道 4
IOA5	I/O	输入/输出 IO, SMT, 上拉电阻
AIN5	A	AD 通道 5
TK5	A	触摸按键通道 5
VREF	A	AD 外部参考电压输入
IOA6	I/O	输入/输出 IO, SMT, 上拉电阻
AIN6	A	AD 通道 6
TK6	A	触摸按键通道 6
PSDA	I/O	编程用
IOA7	I/O	输入/输出 IO, SMT, 上拉电阻
AIN7	A	AD 通道 7
TK7	A	触摸按键通道 7
INT1	I	外部中断
T1CKI	I	TC1 外部时钟输入
PSCK	I/O	编程用
IOB0	I/O	输入/输出 IO, SMT, 上拉电阻
XIN	A	外部晶体振荡器接口
AIN10	A	AD 通道 10
TK9	A	触摸按键通道 9
I2C_SCLA	I/O	I2C 通讯时钟口
IOB1	I/O	输入/输出 IO, SMT, 上拉电阻

XOUT	A	外部晶体振荡器接口
AIN11	A	AD 通道 11
TK10	A	触摸按键通道 10
I2C_SDAA	I/O	I2C 通讯数据口
IOB2	I/O	输入/输出 IO, SMT, 上拉电阻, 只有上拉才能输出高
RSTB	I	外部复位输入, 上拉电阻
VPP	P	编程高压电源
IOB3	I/O	输入/输出 IO, SMT, 上拉电阻
PWM0	O	PWM0 输出
TX	O	通用异步串行通讯发送口
I2C_SCLB	I/O	I2C 通讯时钟口
IOB4	I/O	输入/输出 IO, SMT, 上拉电阻
PWM1PB	O	PWM1 正相输出
RX	I	通用异步串行通讯接收口
I2C_SDAB	I/O	I2C 通讯数据口
IOB5	I/O	输入/输出 IO, SMT, 上拉电阻
PWM1PA	O	PWM1 正相输出
IOB6	I/O	输入/输出 IO, SMT, 上拉电阻
PWM1NA	O	PWM1 反相输出
IOB7	I/O	输入/输出 IO, SMT, 上拉电阻
PWM1NB	O	PWM1 反相输出
IOC0	I/O	输入/输出 IO, SMT, 上拉电阻
AIN8	A	AD 通道 8
TKCOM0	A	触摸按键灵敏度电容口
IOC1	I/O	输入/输出 IO, SMT, 上拉电阻
AIN9	A	AD 通道 9
TK8	A	触摸按键通道 8

注: I = 输入      O = 输出    I/O = 输入/ 输出    P = 电源    A = 模拟端口

# 2 中央处理器（CPU）

## 2.1 程序存储器

地址	说明
0x0000	复位向量
0x0001 ~ 0x0007	用户区
0x0008	中断向量
0x0009 ~ 0x07EF	用户区
0x07F0 ~ 0x07FF	厂商保留区 (16x16 Bits)
0x0800 ~ 0x087F	EEPROM

### 2.1.1 复位向量（0000H）

QT8S737有以下四种复位方式

- 上电复位
- 看门狗复位
- 外部复位
- 欠压复位

发生上述任一种复位后，程序将从 0000H处重新开始执行，系统寄存器也将都恢复为初始默认值。

**例：定义复位向量**

```

        ORG      0000H
        GOTO     Main_Program    ;//跳转至用户程序开始
        ...
Main_Program:
        ...
Main:
        ...
        GOTO     Main            ;//用户主程序循环
    
```

## 2.1.2 中断向量（0008H）

QT8S737中断向量地址为0008H。一旦有中断响应，程序计数器PC的当前值就会存入堆栈缓存器并跳转到0008H处开始执行中断服务程序。

例：中断服务程序：

```

        ORG      0000H
        GOTO    Main_Program    ;//跳转到程序开始
        ORG      0008H
        GOTO    Interrupt      ;//发生中断后,跳转到中断子程序
Main_Program:
    ...
Main:
    ...
        GOTO    MainP          ;//主程序循环

Interrupt:
        PUSH                    ;//压栈、保存 A、STATUS
    ...
        POP                    ;//出栈、恢复A、STATUS
        RETIE

        END
    
```

## 2.1.3 查表

利用RDT指令可以读取程序区数据，其中读到的16位数据高位放在HBUF中，低位放在A寄存器中；FSR1的低3位和FSR0组成11位程序区数据寻址指针。

例：查找 ROM 地址为“DTAB”的值

```

        MOVIA   HIGH(DTAB)    ;//获取数据表地址高位
        MOVAR   FSR1         ;//设置数据表高位指针
        MOVIA   LOW(DTAB)    ;//获取数据表地址低位
        MOVAR   FSR0         ;//设置数据表低位指针
                                ;//若需读取表的其它数据,修改指针
        RDT                    ;//读取表的第一个数据0x0102
        MOVAR   TABDL        ;//将低位数据0x02放在TABDL
        MOVR    HBUF,A       ;//高位数据读入累加器A
        MOVAR   TABDH        ;//将高位数据0x01放在TABDH
    ...
DTAB:
        DW      0x0102
        DW      0x1112
    
```

利用加 PCL 地址来跳转，通过 GOTO 指令可以跳转不同的程序标号。

**例 2:+PCL GOTO 表**

MOVR	ADDRESS,A	;//获取表格地址
ADDAR	PCL,R	
GOTO	TAB1	;//PCL +0 处理程序
GOTO	TAB2	;//PCL +1 处理程序
GOTO	TAB3	;//PCL +2 处理程序
TAB1:		
	处理程序	
TAB2:		
	处理程序	
TAB3:		
	处理程序	

利用加PCL地址来跳转，通过RETIA指令可以读取数据表。

**例 3:+PCL RETIA 表**

MOVR	ADDRESS,A	;//获取地址
ADDAR	PCL,R	;//地址指针加 1
RETIA	0	;// PCL +0
RETIA	1	;// PCL +1
RETIA	2	;//PCL +2
	...	

## 2.2 EEPROM

“Electrically Erasable Programmable Read Only Memory”为可擦可编程只读存储器，由于其非易失的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。EEPROM的写入电压为2.7V~5.5V，可以读写10000次。当不使用这个功能时不建议开启。

### 2.2.1 EECON1 状态寄存器

0F0H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEMOD	-	-	-	-	-	WERR	EEWEN
读/写	R/W	-	-	-	-	-	R	R/W
复位后	0	-	-	-	-	-	0	0

Bit 7 **EEMOD:** EEPROM写入模式使能

1 = 使能EEPROM写入模式

0 = 退出EEPROM写入模式

Bit 1 **WERR:** EEPROM写错误标志

1 = EEPROM写入失败

0 = 未发生EEPROM写入失败

Bit 0 **EEWEN:** EEPROM写入使能

1 = 使能EEPROM写入

0 = 退出EEPROM写入

注：（1）所有非EECON1和EECON2的寄存器写入操作，将使EEWEN清零  
 （2）在读写EEPROM时要关闭所有中断

### 2.2.2 EECON2 状态寄存器

0F1H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON2	-	-	-	-	EELOCK3	EELOCK2	EELOCK1	EEWE
读/写	-	-	-	-	W	W	W	W
复位后	-	-	-	-	0	0	0	0

Bit 3 **EELOCK3:** 解锁流程1

在EECON1[0]=1, EECON2 =00H时，使用置位指令置位EELOCK3，完成EE或MTP写入解锁流程1，完成该操作后，EECON2的值为08H

Bit 2 **EELOCK2:** 解锁流程2

在EECON1[0]=1, EECON2 =08H时，使用置位指令置位EELOCK2，完成EE或MTP写入解锁流程2，完成该操作后，EECON2的值为04H

Bit 1 **EELOCK1:** 解锁流程3

在EECON1[0]=1, EECON2 =04H时，使用置位指令置位EELOCK1，完成EE或MTP写入解锁流程3，完成该操作后，EECON2的值为02H

Bit 0 **EEWE:** 启动写入操作

在EECON1[0]=1, EECON2 =02H时，使用置位指令置位EEWE，启动EE写入操作，写入操作完成后，EECON2=00H

**例: 写 EE**

```

MOVIA    HIGH(DTAB)    ;//获取EE地址高位
MOVAR    FSR1          ;//设置EE地址高位指针
MOVIA    LOW(DTAB)    ;//获取EE地址低位
MOVAR    FSR0          ;//设置EE地址低位指针
MOVIA    0x55          ;//设置EE写入数据,即将0x55送入ACC 为写入
                        ;//的EE数据

BSET     bEEMOD        ;//读写EE/MTP时需置位
BSET     bEEWEN        ;//使能EE/MTP写入
BSET     bEELOCK3      ;//EE/MTP写入解锁流程1
BSET     bEELOCK2      ;//EE/MTP写入解锁流程2
BSET     bEELOCK1      ;//EE/MTP写入解锁流程3
BSET     bEEWR         ;//写入
                        ;//CPU暂停,等待EE写入结束

NOP
BCLR     bEEMOD        ;//写完EE,清除该位
JBTS0    bWERR
GOTO     WeeErrProcess ;//写入失败,跳转到错误处理程序
...
...
    
```

注: EEPORM或MTP写入时, 请关闭所有唤醒源, 不然唤醒源可能中断写入操作, 造成写入失败

Write Voltage    VDD: 2.7V~5.5V

**例: 读 EE**

```

MOVIA    HIGH(DTAB)    ;//获取数据表地址高位
MOVAR    FSR1          ;//设置数据表高位指针
MOVIA    LOW(DTAB)    ;//获取数据表地址低位
MOVAR    FSR0          ;//设置数据表低位指针
                        ;//若需读取表的其它数据,修改指针

BSET     bEEMOD        ;//读EE,需置位
RDT
BCLR     bEEMOD        ;//清除该位
MOVAR    TABDL         ;//将EE数据放在TABDL
    
```

## 2.3 数据存储器

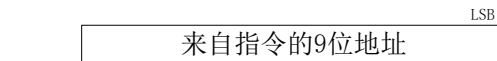
### 2.3.1 数据存储器结构

地址	间接寻址 INDF0	间接寻址 INDF1	间接寻址 INDF2	直接寻址
0X100 ~ 0x1FF	NO	NO	YES	YES
0x000 ~ 0x0FF	YES	YES		

注：此芯片间接寻址 INDF1 直接映射到间接寻址 INDF0

### 2.3.2 数据存储器寻址模式

9位数据寄存器地址组成



直接寻址模式

例: MOVAR 0X55 ;把A寄存器内容写入0x55地址



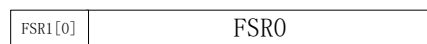
间接寻址模式0

例: MOVAR INDF0 ;把A寄存器内容写入FSR0指向寄存器



间接寻址模式1

例: MOVAR INDF1 ;把A寄存器内容写入FSR1指向寄存器



间接寻址模式2

例: MOVAR INDF2 ;把A寄存器内容写入FSR1/FSR0指向寄存器



### 2.3.3 系统寄存器定义

数据寄存器映射表								
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
0x000 ~ 0x070	GPR							
0x080 ~ 0x0A0	RESERVE							
0x0A8								
0x0B0	INDF0	FSR0	TKCTR0	TKCTR1	TKCNTL	TKCNTH	TKCHSL	TKCHSH
0x0B8	INDF1	FSR1	PCL	STATUS	OPTION	OSCM	WDTC	IOBICR
0x0C0	INDF2	HBUF	PDB		INTCR0	INTF0	INTCR1	INTF1
0x0C8	IOA	OEA	PUA	ANSA	IOB	OEB	PUB	ANSB
0x0D0	IOC	OEC	PUC	ANSC	IOIDS	I2CADR	I2CCON	I2CBUF
0x0D8	PWM0CR	PWM0D	TXCR	TXREG	RXCR	RXREG	BRGDH	BRGDL
0x0E0	ADCON0	ADCON1	ADL	ADH	ADCON2	T1CR	TC1CL	TC1CH
0x0E8	T0CR	TC0CL	TC0CH	PWMDEADT	PWM1S	PWM1CR	PWM1DL	PWM1DH
0x0F0	EECON1	EECON2						
0x0F8			VREFCAL		IRCCAL			

注：GPR 为通用寄存器

### 2.3.4 INDF0 间接寻址寄存器 0

访问INDF0寄存器时，实现间接寻址模式0，访问到的是FSR0寄存器所指向的寄存器内容，间接寻址模式0仅可寻址通用寄存器区0x0000~0x00FF空间

### 2.3.5 INDF1 间接寻址寄存器 1

访问INDF1寄存器时，实现间接寻址模式1，访问到的是FSR1寄存器所指向的寄存器内容，间接寻址模式1仅可寻址通用寄存器区0x0100~0x01FF空间

### 2.3.6 FSR0 间接寻址指针 0

利用间接寻址模式0访问通用寄存器时，FSR0为地址指针；当以间接寻址模式2访问通用寄存器时，FSR0作为地址指针的低位

### 2.3.7 FSR1 间接寻址指针 1

利用间接寻址模式1访问通用寄存器时，FSR1为地址指针；当以间接寻址模式2访问通用寄存器时，FSR1作为地址指针的高位

### 2.3.8 HBUF 查表数据高 8 位

利用RDT指令读取程序区数据时，读到的16位数据高8位放在HBUF中。

### 2.3.9 PCL 程序计数器指针低位

0BAH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCL	PCL7	PCL6	PCL5	PCL4	PCL3	PCL2	PCL1	PCL0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	1

Bit[7:0] **PCL[7:0]:** 程序计数器指针低位

用户将该PCL作为目的操作数做加法运算时（ADDAR PCL、ADCAR PCL），13位PC值参与运算，运算结果写入PC，实现程序的相对跳转；加法运算外的其它运算时，仅PCL参与运算，PCH保持不变，PCH不可寻址。

### 2.3.10 STATUS 状态寄存器

0BBH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	-	-	-	-	-	Z	DC	C
读/写	-	-	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	X	X	X

Bit 2 **Z:** 零标志

1 = 算术/逻辑运算的结果为零  
 0 = 算术/逻辑运算的结果非零

Bit 1 **DC:** 辅助进位标志

1 = 加法运算时低四位有进位，或减法运算后没有向高四位借位  
 0 = 加法运算时低四位没有进位，或减法运算后有向高四位借位

Bit 0 **C:** 进位标志

1 = 加法运算后有进位、减法运算没有借位发生或移位后移出逻辑“1”  
 0 = 加法运算后没有进位、减法运算有借位发生或移位后移出逻辑“0”

# 3 复位

## 3.1 复位方式

- 上电复位 (POR)
- 外部复位 (MCLR Reset)
- 欠压复位 (BOR)
- 看门狗定时器复位 (WDT Reset)

QT8S737 有以上4种复位方式，任何一种复位都会使PC程序计数器清零，让程序从0000H处开始运行，并且使系统寄存器值复位。

判断看门狗是否发生复位主要是通过寄存器OPTION的TO、PD两个位：

TO是超时位，当TO=1时，WDT上电复位或清除WDT，当TO=0时，WDT发生溢出；

PD是掉电位，当PD=1时，WDT上电复位或清除WDT，当PD=0时，WDT进入休眠模式。

# 4 系统时钟

## 4.1 概述

QT8S737支持双时钟系统：高速时钟和低速时钟。高速时钟由外部晶体振荡器或内置的16MHz RC振荡电路(IHRC 16MHz)提供,低速时钟由低速晶体振荡器(0-500KHz)或内置的低速RC振荡电路(ILRC 64KHz/500KHz)提供。两种时钟都可作为系统时钟源Fosc,系统工作在低速模式时,Fosc 2分频后为一个指令周期。低频系统时钟源和高频系统源可根据芯片配置字进行配置。

注：工作时勿在进行高低频切换同时 STOP CPU 操作，可能会造成系统紊乱。

## 4.2 OSCM 寄存器

工作模式控制寄存器 OSCM

0BDH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCM	STBH	STBL	HSPDX2	STOP	CLKM	STPH	LPSPD	STPL
读/写	RW	RW	R/W	RW	RW	RW	RW	RW
复位后	x	x	0	0	x	1	0	1

- Bit 7      **STBH:**    高频振荡器稳定标志
- Bit 6      **STBL:**    低频振荡器稳定标志
- Bit 5      **HSPDX2:**    高频振荡器倍频（仅用于定时器选用高频作为时钟）
  - 1 = 倍频使能
  - 0 = 不使能倍频
- Bit 4      **STOP:**    CPU工作状态标志位
  - 1 = CPU停止工作
  - 0 = CPU正常工作，所有复位唤醒
- Bit 3      **CLKM:**    系统时钟状态标志位
  - 1 = CPU运行于低频时钟
  - 0 = CPU运行于高频时钟
- Bit 2      **STPH:**    高频振荡器控制
  - 1 = 休眠状态或低速模式下关闭高频振荡器
  - 0 = 休眠状态或低速模式下高速振荡器仍然工作
- Bit 1      **LPSPD:**    低频振荡器频率选择
  - 1 = 内部低频振荡器频率500KHz
  - 0 = 内部低频振荡器频率32KHz
- Bit 0      **STPL:**    低频振荡器控制
  - 1 = 休眠状态下低频振荡器停止工作
  - 0 = 休眠状态下低频振荡器仍然工作

注：CLKM 的初始状态由配置字 SPDUP 决定。

### 4.3 系统时钟的工作模式

- 普通模式：**普通模式有两种分别是：1.高频时钟工作，低频时钟工作，不进 STOP  
 2.高频时钟工作，低频时钟不工作，不进 STOP
- 绿色模式：**绿色模式有两种分别是：1.高频时钟工作，低频时钟工作，进 STOP  
 2.高频时钟停止，低频时钟工作，进 STOP
- 所有中断唤醒，WDT 都可以唤醒绿色模式。
- 休眠模式：**高频时钟停止，低频时钟停止，进 STOP，可通过外部中断，WDT 唤醒。

注：省电建议，程序运行时跑高频，快速跑完程序然后进休眠，休眠下需设置高频停止工作。

### 4.4 IRCCAL 寄存器

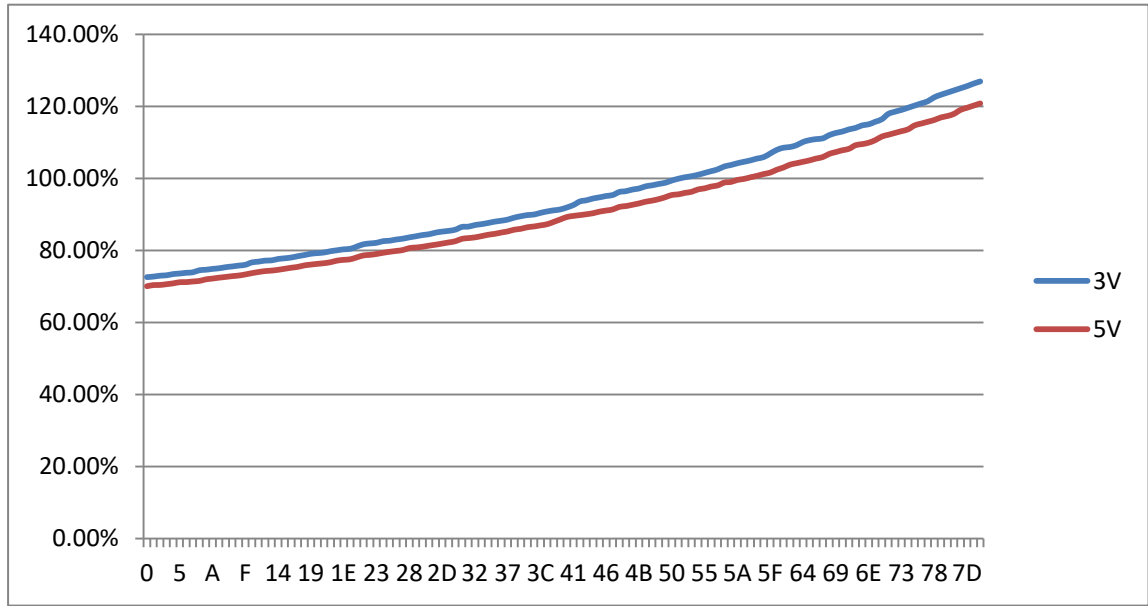
0FCH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IRCCAL	-	IRCAL6	IRCCAL5	IRCCAL4	IRCCAL3	IRCCAL2	IRCCAL1	IRCCAL0
读/写	-	RW	RW	RW	RW	RW	RW	RW
复位后	-	X	X	X	X	X	X	X

内置的高频 RC 震荡电路在芯片上电后频率为校准过的 16MHz，但程序中可以通过特殊的流程来调整此频率以满足特定应用需求。

#### 例：调整 IRC 频率

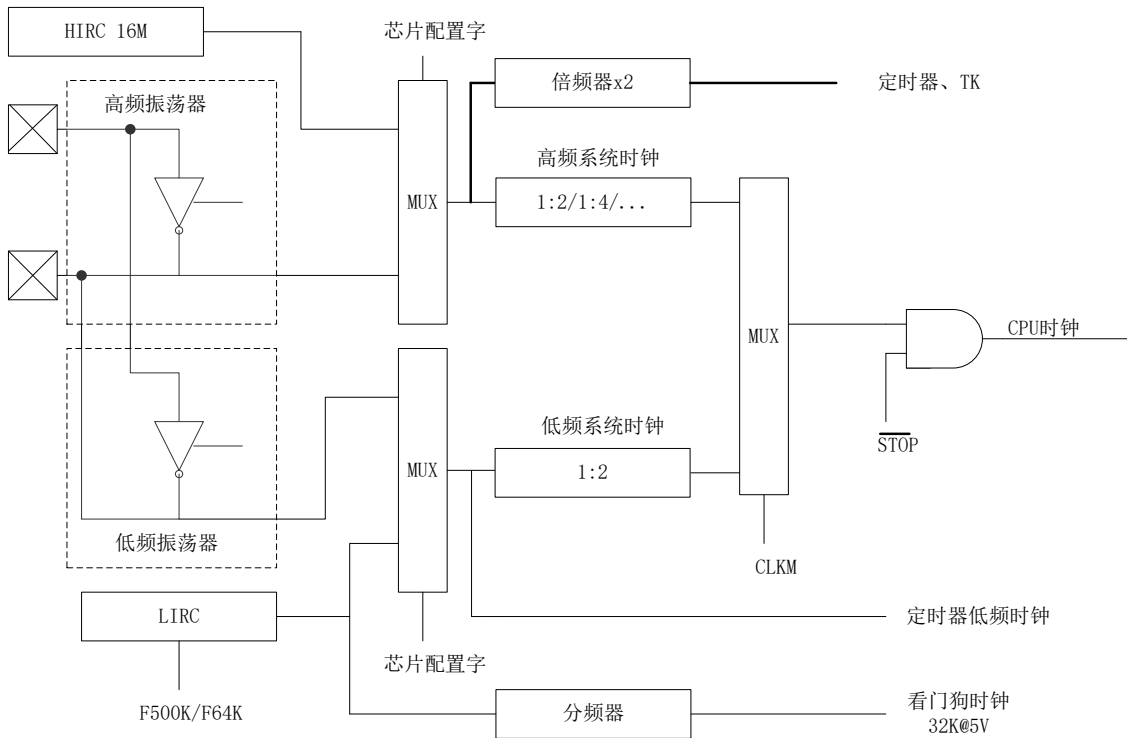
```

TASK_IRCCAL:
    MOVIA    0x55
    MOVAR    0x1F9           ;//1F9H地址写入055H
    MOVIA    0xAA
    MOVAR    0x1F9           ;//1F9H地址写入0AAH
    MOVIA    VALUE
    MOVAR    IRCCAL         ;//写入IRCCAL
    ...
;//若需继续在IRCCAL寄存器内写入其他值需要重复以上所有步骤
    
```

**IRC 调整频率图**


注：具体值不做设计保证

## 4.5 系统时钟结构框图



	高速运行模式 (CLKM=0)	低速运行模式 (CLKM=1)	休眠模式 (STOP=1)
高频振荡器	运行	由 STPH 决定	由 STPH 决定
低频振荡器	运行	运行	由 STPL 决定
WDT	配置字决定	由配置字决定	由配置字决定
TC0/TC1	TCxEN	若选择高速时钟，需 STPH=0	高速时钟源&STPH=0 或低 速时钟源&STPL=0

## 4.6 系统时钟高低频切换

高频振荡器稳定计数器：64 Clocks（内部 IRC 模式）/ 1024 Clocks（外部高频振荡器模式）

低频振荡器稳定计数器：16 Clocks（内部 RC 模式）/1024 Clocks（外部低频振荡器模式）

高低频切换时间

高频切低频：1 个低频时钟周期+1 个高频时钟周期

低频切高频&STBH=0：1 个低频时钟周期+高频振荡器起振时间+高频振荡器稳定时间

低频切高频&STBH=1：1 个低频时钟周期+1 个高频时钟周期

唤醒时间：

CLKM=0&STBH=0：高频振荡器起振时间+高频振荡器稳定时间

CLKM=0&STBH=1：64 Clocks

CLKM=1&STBL=0：低频振荡器起振时间+低频振荡器稳定时间

CLKM=1&STBL=1：16 Clocks

# 5 中断

## 5.1 概述

QT8S737有多路中断源：TC0/TC1, IOB口变化中断, UART发送/接收中断, ADC中断, INT0、INT1 中断。中断可以将系统从睡眠模式中唤醒，在唤醒前，中断请求被锁定。一旦程序进入中断，寄存器OPTION的位GIE被硬件自动清零以避免响应其它中断。系统退出中断后，硬件自动将GIE置“1”，以响应下一个中断。

设置 GIE 和中断控制寄存器 INTCR0/INTCR1 来使能中断，查询 INTF0/INTF1 中断标志寄存器判断中断是否发生。

## 5.2 OPTION 配置寄存器

0BCH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION	GIE	-	TO	PD	MINT11	MINT10	MINT01	MINT00
读/写	R/W	-	R	R	R/W	R/W	R/W	R/W
复位后	0	-	1	1	0	0	0	0

Bit 7 **GIE:** 全局中断控制位  
 1 = 总中断使能 (RETIE指令会将该位置1)  
 0 = 屏蔽所有中断 (响应中断后自动清零)

Bit[3:2] **MINT1[1:0]:** INT1中断模式选择

MINT1[1:0]	INT1 中断模式选择
00	上升沿中断
01	下降沿中断
1X	变化中断

Bit[1:0] **MINT0[1:0]:** INTO中断模式选择

MINT0[1:0]	INT0 中断模式选择
00	上升沿中断
01	下降沿中断
1X	变化中断



### 5.3 IO 口变化中断使能寄存器

0BFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOBICR	IOBICR7	IOBICR6	IOBICR5	IOBICR4	IOBICR3	IOBICR2	IOBICR1	IOBICR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7-Bit 0 **IOBICRn:** IO端口变化中断使能

1 = 使能IOB.n口电平变化中断

0 = 屏蔽IOB.n口电平变化中断

### 5.4 INTCR0 中断控制寄存器 0

0C4H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCR0	TKIE	ADIE	RXIE	TXIE	I2CIE	-	TC1IE	TC0IE
读/写	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
复位后	0	0	0	0	0	-	0	0

Bit 7 **TKIE:**

1 = 使能触摸按键中断1

0 = 屏蔽触摸按键中断1

Bit 6 **ADIE:**

1 = 使能ADC转换中断

0 = 屏蔽ADC转换中断

Bit 5 **RXIE:**

1 = 使能串行通讯接收中断

0 = 屏蔽串行通讯接收中断

Bit 4 **TXIE:**

1 = 使能串行通讯发送中断

0 = 屏蔽串行通讯发送中断

Bit 3 **I2CIE:**

1 = 使能I2C中断

0 = 屏蔽I2C中断

Bit 1 **TC1IE:**

1 = 使能TC1溢出中断

0 = 屏蔽TC1溢出中断

Bit 0 **TC0IE:**

1 = 使能TC0溢出中断

0 = 屏蔽TC0溢出中断

## 5.5 INTF0 中断标志寄存器 0

0C5H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF0	TKIF	ADIF	RXIF	TXIF	I2CIF	-	TC1IF	TC0IF
读/写	R/W	R/W	R	R	R/W	-	R/W	R/W
复位后	0	0	0	0	0	-	0	0

注：除 USART 之外的所有中断标志位需软件清零

- Bit 7      **TKIF:**  
 1 = 产生触摸按键中断1  
 0 = 未产生触摸按键中断1
- Bit 6      **ADIF:**  
 1 = 产生ADC转换中断  
 0 = 未产生ADC转换中断
- Bit 5      **RXIF:**  
 1 = 产生串行通讯接收中断  
 0 = 未产生串行通讯接收中断
- Bit 4      **TXIF:**  
 1 = 产生串行通讯发送中断  
 0 = 未产生串行通讯发送中断
- Bit 3      **I2CIF:** (I2C Master写数据)  
 1 = 缓冲区内接收到数据 (读SSPBUF清零)  
 0 = 未收到数据或数据已读
- Bit 1      **TC1IF:**  
 1 = 产生TC1溢出中断  
 0 = 未产生TC1溢出中断
- Bit 0      **TC0IF:**  
 1 = 产生TC0溢出中断  
 0 = 未产生TC0溢出中断

## 5.6 INTCR1 中断控制寄存器 1

0C6H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCR1	-	-	-	-	INT1IE	INT0IE	-	IOBCHIE
读/写	-	-	-	-	R/W	R/W	-	R/W
复位后	-	-	-	-	0	0	-	0

Bit 3      **INT1IE:**  
 1 = 使能外部端口中断1  
 0 = 屏蔽外部端口中断1

Bit 2      **INT0IE:**  
 1 = 使能外部端口中断0  
 0 = 屏蔽外部端口中断0

Bit 0      **IOBCHIE:**  
 1 = 使能端口B变化中断  
 0 = 屏蔽端口B变化中断

## 5.7 INTF1 中断标志寄存器 1

0C7H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF1	-	-	-	-	INT1IF	INT0IF	-	IOBCHIF
读/写	-	-	-	-	R/W	R/W	-	R/W
复位后	-	-	-	-	0	0	-	0

注：所有中断标志位需软件清零

Bit 3      **INTF1IF:**  
 1 = 产生外部中断INT1  
 0 = 未产生外部中断INT1

Bit 2      **INTF0IF:**  
 1 = 产生外部中断INT0  
 0 = 未产生外部中断INT0

Bit 0      **IOBCHIF:**  
 1 = 对应输入端口状态发生变化  
 0 = 对应输入端口状态未发生变化

## 5.8 中断范例

### 例: INT0 中断

```

//+++++
        ORG        0000H
        GOTO       Main_Program           ;//跳转到程序开始
        ORG        0008H
        GOTO       Interrupt             ;//发生中断后,跳转到中断子程序
//+++++
Main_Program:                               ;//程序开始
INT0_Init:                                  ;//INT0 初始化
//1、 端口设置
        BSET       PUA,0                 ;//INT0端口设置为上拉
        BCLR       OEA,0                ;//INT0端口设置为输入
//2、 中断模式选择
        MOVIA      0x31                  ;//INT0下降沿中断
        MOVAR      OPTION
//3、 中断使能
        BSET       INTCR1,INT0IE        ;//INT0中断使能
        BCLR       INTF1,INT0IF
//4、 总中断使能
        BSET       OPTION,GIE           ;//总中断使能
Main:                                       ;//程序主循环
        ...
        GOTO       Main
// ++++++
Interrupt:                                  ;//中断子程序
//中断进来
        PUSH       ;//压栈、保存 A、STATUS
//中断处理程序
        JBTS0      INTF1,INT0IF         ;//检测 INT0 标志位
        GOTO       Interrupt_INT0
Interrupt_End:
//中断结束
        POP        ;//出栈、恢复 A、STATUS
        RETIE
Interrupt_INT0:
        BCLR       INTF1,INT0IF         ;//清 INT0 标志位
//INT0 中断处理程序
        ...
        GOTO       Interrupt_End

        END
    
```

**例:IO 变化中断 (以 IOB0 口为例)**

```

//+++++
        ORG        0000H
        GOTO       Main_Program           ;//跳转到程序开始
        ORG        0008H
        GOTO       Interrupt             ;//发生中断后,跳转到中断子程序
//+++++
Main_Program:                               ;// 程序开始
IO_INTERRUPT_INIT:                          ;//IO 中断初始化
//1、端口设置
        BSET       PUB,0                 ;//IOB0 端口设置为上拉
        BCLR       OEB,0                ;//IOB0 端口设置为输入
//2、允许 B 口变化中断设置
        BSET       IOBICR,0             ;//IOB 口变化中断使能
//3、中断使能
        BSET       INTCR1, IOBCHIE
        BCLR       INTF1, IOBCHIF
//4、总中断使能
        BSET       OPTION,GIE           ;//总中断使能
Main:                                         ;//程序主循环
        ...
        GOTO       Main
// +++++
Interrupt:                                   ;//中断子程序
//中断进来
        PUSH       ;//压栈、保存 A、STATUS
//中断处理程序
        JBTS0      INTF1,IOBCHIF        ;//检测 IO 中断标志位
        GOTO       Interrupt_IO
Interrupt_End:
//中断结束
        POP        ;//出栈、恢复 A、STATUS
        RETIE
Interrupt_IO:
        BCLR       INTF1, IOBCHIF
//IO 中断处理程序
        ...
        GOTO       Interrupt_End

        END
    
```

# 6 端口

## 6.1 IOA

IOA 数据寄存器

0C8H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOA	IOA7	IOA6	IOA5	IOA4	IOA3	IOA2	IOA1	IOA0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

IOA 方向寄存器

0C9H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OEA	OEA7	OEA6	OEA5	OEA4	OEA3	OEA2	OEA1	OEA0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **OEA:** A口输出使能  
 1 = 输出  
 0 = 输入

IOA 上拉使能寄存器

0CAH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PUA	PUA7	PUA6	PUA5	PUA4	PUA3	PUA2	PUA1	PUA0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **PUA:** A口上拉使能  
 1 = 上拉使能  
 0 = 上拉关闭

IOA 端口模式控制寄存器

0CBH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSA	ANSA7	ANSA6	ANSA5	ANSA4	ANSA3	ANSA2	ANSA1	ANSA0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **ANSA:** A口模式控制  
 1 = 作为模拟端口 (I0输入功能屏蔽)  
 0 = 作为数字IO口

## 6.2 IOB

IOB 数据寄存器

0CCH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOB	IOB7	IOB6	IOB5	IOB4	IOB3	IOB2	IOB1	IOB0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

**IOB 方向寄存器**

0CDH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OEB	OEB7	OEB6	OEB5	OEB4	OEB3	OEB2	OEB1	OEB0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **OEB:** B口输出使能  
 1 = 输出  
 0 = 输入

注: IOB[2]作为输出口的注意事项

- (1) 需将 PUB2 置 1 才能输出高电平。
- (2) IOB[2]输出的高电平是由上拉电阻提供的, 所以驱动能力弱。
- (3) IOB[2]输出的低电平驱动能力比其他端口略弱一些, 输出低电平时内部电路会关闭上拉电阻。

**IOB 上拉使能寄存器**

0CEH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PUB	PUB7	PUB6	PUB5	PUB4	PUB3	PUB2	PUB1	PUB0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **PUB:** B口上拉使能  
 1 = 上拉使能  
 0 = 上拉关闭

**IOB 下拉使能寄存器**

0C2H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PDB	PDB7	PDB6	PDB5	PDB4	PDB3	PDB2	PDB1	PDB0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **PDB:** B口下拉使能  
 1 = 下拉使能  
 0 = 下拉关闭

**IOB 端口模式控制寄存器**

0CFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSB	-	-	-	-	-	-	ANSB1	ANSB0
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

Bit[1:0]      **ANSB:**    B口模式控制  
                   1 = 作为模拟端口（IO输入功能屏蔽）  
                   0 = 作为数字IO口

注：同一 IO 口上下拉电阻同时打开时，IO 口将自动屏蔽输入功能（读该端口状态为 0），此时端口电平接近于 VDD/2。

## 6.3 IOC

**IOC 数据寄存器**

0D0H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOC	-	-	-	-	-	-	IOC1	IOC0
读/写	-	-	-	-	-	-	RW	RW
复位后	-	-	-	-	-	-	X	X

**IOC 方向寄存器**

0D1H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OEC	-	-	-	-	-	-	OEC1	OEC0
读/写	-	-	-	-	-	-	RW	RW
复位后	-	-	-	-	-	-	0	0

Bit[7:0]      **OEC:**    C口输出使能  
                   1 = 输出  
                   0 = 输入

**IOC 上拉使能寄存器**

0D2H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PUC	-	-	-	-	-	-	PUC1	PUC0
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

Bit[7:0]      **PUC:**    C口上拉使能  
                   1 = 上拉使能  
                   0 = 上拉关闭



**IOC 端口模式控制寄存器**

0D3H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSC	-	-	-	-	-	-	ANSC1	ANSC0
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

Bit[1:0]      **ANSC:**    C口模式控制  
                   1 = 作为模拟端口 (IO输入功能屏蔽)  
                   0 = 作为数字IO口

## 6.4 IOIDS 端口驱动设置寄存器

0D4H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOIDS	IOBHIS	IOBLIS	IOAHIS	IOALIS	IOBHDS	IOBLDS	IOAHDS	IOALDS
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7]      **IOBHIS:**    IOB7-4翻转电平选择

IOBHIS	IOB[7:4] 电平选择
0	SMT
1	低反转点

Bit[6]      **IOBLIS:**    IOB3-0翻转电平选择

IOBLIS	IOB[3:0] 电平选择
0	SMT
1	低反转点

Bit[5]      **IOAHIS:**    IOA7-4翻转电平选择

IOAHIS	IOA[7:4] 电平选择
0	SMT
1	低反转点

Bit[4]      **IOALIS:**    IOA3-0翻转电平选择

IOALIS	IOA[3:0] 电平选择
0	SMT
1	低反转点

Bit[3]      **IOBHDS:**    IOB7-4输出驱动电流选择

IOBHDS	IOB[7:4] 输出驱动电流选择
0	大驱动 (IoL2\ IoH2)
1	小驱动 (IoL1\ IoH1)

Bit[2]      **IOBLDS:**    IOB3-0输出驱动电流选择

IOBLDS	IOB[3:0] 输出驱动电流选择
0	大驱动 (IoL2\ IoH2)
1	小驱动 (IoL1\ IoH1)

Bit[1]      **IOAHDS:**    IOA7-4输出驱动电流选择

IOAHDS	IOA[7:4]输出驱动电流选择
0	大驱动 (IOL2\ IOH2)
1	小驱动 (IOL1\ IOH1)

Bit[0]

**IOALDS:** IOA3-0输出驱动电流选择

IOALDS	IOA[3:0]输出驱动电流选择
0	大驱动 (IOL2\ IOH2)
1	小驱动 (IOL1\ IOH1)

注：IO 口的各个驱动能力参考电性参数表

# 7 定时器0/1(TC0/1)

## 7.1 概述

QT8S737 TC0/TC1 为带有可设置 1:128 预分频器及周期寄存器的 8 位/16 位定时计数器，具有休眠状态下唤醒功能。

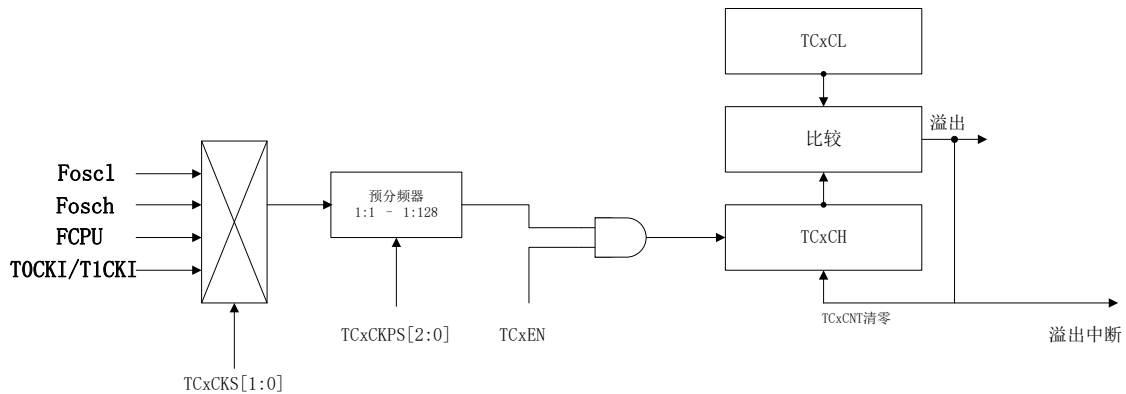
在 8 位模式下，TCxCL 作为 TCx 的周期寄存器，TCx 使能后，TCxCH 递加，当 TCxCH 与 TCxCL 数值相等时，TCx 溢出，将 TCxCH 清零重新开始计数，同时将中断标志位 TxIF 置 1。

在 16 位模式下，[TCOCH, TCOCL]作为 16 位的计数器，TC0 使能后，16 位计数器递加，当计数值等于 0xFFFF 时，16 位计数器将清零重新开始计数，同时将中断标志位 TCxIF 置 1。

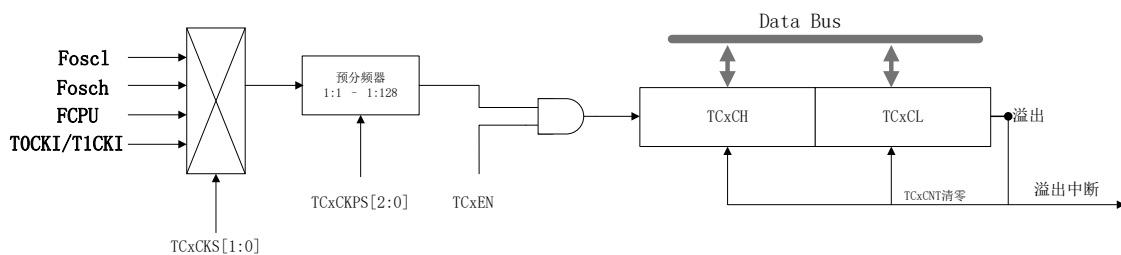
- 可选择时钟源，高频系统时钟 Fosch、低频系统时钟 Foscl 和指令时钟 Fcpu 、外部时钟 TxCKI
- 可选择 8 位模式和 16 位模式
  - ✓ 8 位模式下，通过设置周期寄存器，可任意设置 TC0 的周期
- 预分频比多级可选，最大可选择 1:128
- 溢出中断功能
- 溢出中断唤醒功能（当输入频率选择 Foscl 或 Fosch 时，若所选择的时钟源振荡器一直工作，此时 TC0 在休眠状态下依然工作，溢出中断可唤醒 CPU）

TC0/TC1 框图

8 位模式



16 位模式



## 7.2 TxCR 控制寄存器

0E8/0E5H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TxCR	TCxEN	TCxMOD	-	TCxCKS1	TCxCKS0	TCxCKPS2	TCxCKPS1	TCxCKPS0
读/写	R/W	R/W	-	R/W	R/W	R/W	R/W	R/W
复位后	0	0	-	0	0	0	0	0

Bit 7 **TCxEN:** TCx模块使能位

1 = 使能TCx

0 = 关闭TCx

Bit 6 **TCxMOD:** TCx模式选择位

1 = 16位模式

0 = 8位模式

Bit 5 未定义

Bit[4:3] **TCxCKS:** TCx时钟源选择

TCxCKS[1:0]	TCx 时钟源选择
00	Foscl (低频系统时钟)
01	Fosch (高频系统时钟)
10	Fcpu
11	TOCKI(TC0)/TICKI(TC1)

Bit[2:0] **TCxCKPS[2:0]:** TCx预分频比选择

TCxCKPS[2:0]	TCx 预分频比
000	1:1
001	1:2
010	1:4
011	1:8
100	1:16
101	1:32
110	1:64
111	1:128

## 7.3 TCxCL TCx 计数器低 8 位/周期寄存器

0E9/0E6H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCxCL	TCxCL7	TCxCL6	TCxCL5	TCxCL4	TCxCL3	TCxCL2	TCxCL1	TCxCL0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	X	x	x	x	x	x

## 7.4 TCxCH TCx 计数器高位

0EA/0E7H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCxCH	TCxCH7	TCxCH6	TCxCH5	TCxCH4	TCxCH3	TCxCH2	TCxCH1	TCxCH0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

## 7.5 定时器范例

例：以 TC0 为例，内置的 16MHz RC 震荡电路提供振荡频率，定时 125us 程序

```

;+++++
        ORG          0000H
        GOTO         Main_Program          ;//跳转到程序开始
        ORG          0008H
        GOTO         Interrupt            ;//发生中断后,跳转到中断子程序
;+++++
Main_Program:                               ;//程序开始
TC0_Init:                                   ;//TC0初始化
;//1、确保内置高频16M时钟在工作状态
        MOVIA        b'00000000'
        MOVAR        OSCM
;//2、配置T0CR控制寄存器
        MOVIA        b'00001100'          ;// 8位模式,高频系统时钟源,1:16分频
        MOVAR        T0CR                 ;//将模式位配置字写入T0CR
;//3、清零计数寄存器
        CLRR         TC0CH                ;//将TC0CH清零
;//4、配置自动加载计数器
        MOVIA        124                  ;//将124写入A
        MOVAR        TC0CL                ;// 16M/16/(124+1)=125us
;//5、使能TC0,开启TC0中断
        BCLR         INTF0,TC0IF          ;//TC0 中断标志清零
        BSET         T0CR,7               ;//使能 TC0
        BSET         INTCR0, TC0IE        ;//使能 TC0 溢出中断
Main:                                       ;//程序主循环
        ...
        GOTO         Main
;+++++
Interrupt:                                  ;//中断子程序
        PUSH         ;//压栈,保存 A,STATUS
;//中断处理程序
        JBTS0        INTF1, TC0IF         ;//检测 TC0 标志位
        GOTO         Interrupt_TC0
Interrupt_End:
        POP          ;//出栈,恢复 A,STATUS
        RETIE
Interrupt_TC0:
    
```

```
;//中断处理程序  
    NOP
```

# 8 脉宽调制模块PWM0

## 8.1 概述

QT8S737 有一个 8 位分辨率 PWM0，时基使用 TCO.PWM。

## 8.2 PWM0CR 控制寄存器

0D8H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0CR	PWM0EN	PWM0OE	-	-	-	-	-	-
读/写	R/W	R/W	-	-	-	-	-	-
复位后	0	0	-	-	-	-	-	-

Bit 7 **PWM0EN:** PWM0模块使能位

1 = 使能PWM0

0 = 关闭PWM0

Bit 6 **PWM0OE:** PWM0输出控制

1 = PWM0信号从管脚输出

0 = PWM0信号不从管脚输出，管脚用做IO

## 8.3 PWM0D 数据高位

0D9H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0D	PWM0D7	PWM0D6	PWM0D5	PWM0D4	PWM0D3	PWM0D2	PWM0D1	PWM0D0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

## 8.4 PWM0 范例

例：PWM0 以 TC0 为时基，内置的 16MHz RC 震荡电路提供振荡频率，设置 50% 占空比，在 IOB3 口输出 1kHz 波形

```

;+++++
                ORG        0000H
                GOTO      Main_Program      ;//跳转到程序开始
                ORG        0008H
                GOTO      Interrupt         ;//发生中断后,跳转到中断子程序
;+++++
Main_Program:                                     ;//程序开始
PWM0_Init:                                       ;//PWM0初始化
;//1、IO端口设置:
                BSET      OEB,3              ;//设置 IOB3 为波形输出端口
;//2、时钟源设置:
                MOVIA     b'01001110'      ;// 8位模式,高频系统时钟源,1:64分频
                MOVAR     T0CR              ;//将模式位配置字写入
                CLRR      TC0CH            ;//将TC0CH清零
                MOVIA     249               ;//将249写入A
                MOVAR     TC0CL            ;//16MHz/250/64=1KHz
;//3、PWM0设置:
                MOVIA     b'01000000'      ;//设置PWM0信号从管脚输出
                MOVAR     PWM0CR           ;//端口输出波形
                MOVIA     (249+1)/2        ;//产生占空比为50%的PWM波形
                MOVAR     PWM0D            ;//设置数据高位
;//4、使能PWM0和TC0:
                BSET      T0CR,7           ;//使能定时器T0
                BSET      PWM0CR,7        ;//使能PWM0
Main:
                ...
                GOTO      Main
;+++++
Interrupt:                                         ;//中断子程序
                PUSH      ;//压栈,保存 A,STATUS
;//中断处理程序
                NOP
Interrupt_End:
                POP        ;//出栈,恢复 A,STATUS
                RETIE
                END
    
```



# 9 脉宽调制模块PWM1

## 9.1 概述

QT8S737 有 1 路带有死区控制的 PWM，时基使用 TC1，可独立进行设置，8+4 位分辨率。

- 正常 PWM 模式（定时器 1 需设置为 8 位模式）
- 单脉冲模式（定时器 1 需设置为 16 位模式）
- 互补输出
- 死区控制
- 12 位分辨率模式
- 8+4 分辨率模式
- 多个管脚输出

## 9.2 PWM1CR 控制寄存器

0EDH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1CR	PWM1EN	PWM1POE	PWM1NOE	PWM1PAS	PWM1NAS	-	PWM1M	PWM1T
读/写	R/W	R/W	R/W	R/W	R/W	-	RW	R/W
复位后	0	0	0	0	0	-	0	0

- Bit 7      **PWM1EN:**    PWM1 模块使能位  
             1 = 使能 PWM1  
             0 = 关闭 PWM1
- Bit 6      **PWM1POE:**    PWM1 正相波形输出使能位  
             1 = 端口输出 PWM1P 波形  
             0 = 端口用作 IO
- Bit 5      **PWM1NOE:**    PWM1 反相波形输出使能位  
             1 = 端口输出 PWM1N 波形  
             0 = 端口用作 IO
- Bit 4      **PWMxPAS:**    PWMxP 波形有效电平选择  
             1 = PWMxP 波形有效电平为低电平  
             0 = PWMxP 波形有效电平为高电平
- Bit 3      **PWM1NAS:**    PWM1N 波形有效电平选择  
             1 = 端口输出 PWM 波形高电平有效  
             0 = 端口输出 PWM 波形低电平有效
- Bit 1      **PWM1M:**    PWM1 模式选择  
             1 = 单脉冲模式（需将定时器 1 设置成 16 位模式）  
             0 = 正常 PWM 模式（需将定时器 1 设置成 8 为模式）
- Bit 0      **PWM1T:**    PWM1 触发事件设置  
             1 = TICK1 上升沿/下降沿  
             0 = TICK1 上升沿

### 9.3 PWM1S 控制寄存器

0ECH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1S	-	-	-	-	-	PWM1C	PWM1B	PWM1A
读/写	-	-	-	-	-	R/W	RW	RW
复位后	-	-	-	-	-	0	1	0

- Bit 2      **PWM1C:**    PWM1C 输出管脚设置  
             1 = PWM1PC/PWM1NC 用作PWM输出  
             0 = PWM1PC/PWM1NC 用作IO口
- Bit 1      **PWM1B:**    PWM1B 输出管脚设置  
             1 = PWM1PB/PWM1NB 用作PWM输出  
             0 = PWM1PB/PWM1NB 用作IO口
- Bit 0      **PWM1A:**    PWM1 输出管脚设置  
             1 = PWM1PA/PWM1NA 用作PWM输出  
             0 = PWM1PA/PWM1NA 用作IO口

### 9.4 PWM1DH 数据高位

0EFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1DH	PWM1DH7	PWM1DH6	PWM1DH5	PWM1DH4	PWM1DH3	PWM1DH2	PWM1DH1	PWM1DH0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

### 9.5 PWM1DL 数据低位

0EEH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1DL	PWM1DL7	PWMxDL6	PWMxDL5	PWMxDL4	-	-	-	-
读/写	R/W	R/W	R/W	R/W	-	-	-	-
复位后	0	0	0	0	-	-	-	-

## 9.6 PWMDEADT PWM 死区控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMDEADT	DEADTF3	DEADTF2	DEADTF1	DEADTF0	DEADTR3	DEADTR2	DEADTR1	DEADTR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:4] **DEADTF[3:0]:** 前死区宽度设置

DEADTF[3:0]	前死区时间设定
0000	无前死区
0001	前死区时间为 1*(时基时钟周期/2)
0011	前死区时间为 2*(时基时钟周期/2)
0010	前死区时间为 3*(时基时钟周期/2)
1010	前死区时间为 4*(时基时钟周期/2)
1011	前死区时间为 5*(时基时钟周期/2)
1001	前死区时间为 6*(时基时钟周期/2)
1101	前死区时间为 7*(时基时钟周期/2)
0101	前死区时间为 8*(时基时钟周期/2)
0100	前死区时间为 9*(时基时钟周期/2)
0110	前死区时间为 10*(时基时钟周期/2)
0111	前死区时间为 11*(时基时钟周期/2)
1111	前死区时间为 12*(时基时钟周期/2)
1110	前死区时间为 13*(时基时钟周期/2)
1100	前死区时间为 14*(时基时钟周期/2)
1000	前死区时间为 15*(时基时钟周期/2)

Bit [3:0] **DEADTR[3:0]:** 后死区宽度设置

DEADTR[3:0]	后死区时间设定
0000	无后死区
0001	后死区时间为 1*(时基时钟周期/2)
0011	后死区时间为 2*(时基时钟周期/2)
0010	后死区时间为 3*(时基时钟周期/2)
1010	后死区时间为 4*(时基时钟周期/2)
1011	后死区时间为 5*(时基时钟周期/2)
1001	后死区时间为 6*(时基时钟周期/2)
1101	后死区时间为 7*(时基时钟周期/2)
0101	后死区时间为 8*(时基时钟周期/2)
0100	后死区时间为 9*(时基时钟周期/2)
0110	后死区时间为 10*(时基时钟周期/2)
0111	后死区时间为 11*(时基时钟周期/2)
1111	后死区时间为 12*(时基时钟周期/2)
1110	后死区时间为 13*(时基时钟周期/2)
1100	后死区时间为 14*(时基时钟周期/2)
1000	后死区时间为 15*(时基时钟周期/2)

死区时间设置:

$$T_{deadr} = DEADTR * \text{时基时钟周期}$$

$$T_{deadf} = DEADTF * \text{时基时钟周期}$$

注:

- (1) 时基时钟周期即各 PWM 所选择的时钟源经预分频之后的时钟周期。
- (2) 3 路 PWM 共用同一档前/后死区宽度设置寄存器，每路 PWM 选择不同的时基时，区宽度计算是对应不同的时基时钟周期。

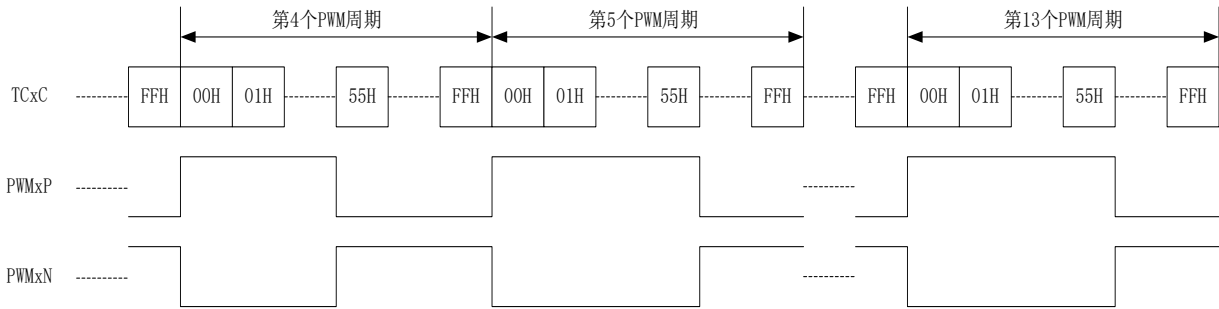
## 9.7 8+4 位分辨率模式

PWM1D[3:0]为 4 位扩展位, WMD[11:4]决定 PWM 脉冲基础宽度。每 16 个 PWM 周期循环中, 展位中的有效位对应的 PWM 周期, 出的 PWM 脉冲宽度为 (PWMD[11:4]+1), 其余的 PWM 周期, 出的 PWM 脉冲宽度为 (PWMD[11:4]), 样得到的 PWM 输出是等效的 12 位 PWM 分辨率效果。

PWM1D[3:0]对应的扩展周期序号:

PWM1D[3:0]	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
PWM1D3		●		●		●		●		●		●		●		●
PWM1D2			●				●				●				●	
PWM1D1					●								●			
PWM1D0									●							

例:: PWM1CR=11100000B, PWM1DH=55H, PWM1DL=2H, TC0CL=FFH



## 9.8 单脉冲模式

单脉冲模式下, TICKI 输入 n 个事件 (上升沿或下降沿), WM1DH、PWM1DL 作为 PWM 脉冲宽度。

-脉冲宽度设置:

$$\text{PulsWidth} = 62.5\text{ns} \times \text{定时器 1 的预分频比} \times \text{PWM1D}$$

若定时器 1 的分频比为 1:16, PWM1DH=50, WM1DL=0, PWM1D 数据高低位转换为二进制, PWM1D=1100100000, 化为十进制 PWM1D=800, 脉宽计算公式可得:  $\text{ulsWidth} = 62.5\text{ns} \times 16 \times 800 = 800\text{us}$ ; 设置分频比为 1:16,  $\text{PWM1D}(\text{max}) = 2^{12} = 4096$ , 设置脉宽的最大宽度为 4096us (约为 4ms)。

-滤波时间设置:

$$\text{滤波时间} = 62.5\text{ns} \times \text{定时器 1 的预分频比} \times \text{PWMDEADT}[2:0]$$

PWMDEADT[2:0] 作为触发信号的滤波时间设置, 发信号的采样时钟为定时器的计数时钟  
 若定时器 1 的分频比为 1:16, WMDEADT[2:0] 设置为 "010", 滤波时间 =  $62.5\text{ns} \times 16 \times 2 = 2\text{us}$ ;  
 若设置分频比为 1:16, PWMDEADT[2:0] 设置为 "111", 设置的最大滤波时间 =  $62.5\text{ns} \times 16 \times 8 = 8\text{us}$ 。

-PWMDEADT[7:3] 作为触发事件设置, 高可选 16 个触发事件中产生的脉冲个数。

PWMDEADT[7:3]	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
PWMDEADT[7]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
PWMDEADT[6]		●		●		●		●		●		●		●		●
PWMDEADT[5]			●				●				●				●	
PWMDEADT[4]					●							●				
PWMDEADT[3]									●							

**范例：单脉冲模式下，以 PWMDEADT3 为例，TC1 时基**

```

;// ++++++
        ORG          0000H
        GOTO        Main_Program      ;//跳转到程序开始
        ORG          0008H
        GOTO        Interrupt        ;//发生中断后,跳转到中断子程序
;//++++++
Main_Program:                ;// 程序开始
PWM1_Init:                  ;//PWM1初始化
;//1、 确保内置高频16M时钟在工作状态
        MOVIA      b'00000000'      ;//设置系统时钟为高频16M时钟
        MOVAR      OSCM
;//2、 IO 端口设置
        MOVIA      0x7F              ;//T1CKI 口作外接信号输入
        MOVAR      OEA
        MOVIA      0x03              ;//PWM1NA/PWM1PA 口输出 PWM1 波形
        MOVAR      OEA
        .....
;//3、 时钟源设置(以 TC1 为例,使其产生 1KHz 的波形)
        MOVIA      0xFF
        MOVAR      TC1CH
        MOVIA      0xFF
        MOVAR      TC1CL
        MOVIA      B'11001100'      ;//定时器使能,16 位模式,高频系统时钟源,1:16 分频
        MOVAR      T1CR              ;//定时器 T1
;//4、 PWM设置
        MOVIA      b'11000011'      ;//PWM 时基选择定时器 T1
        MOVAR      PWM1CR            ;//PWM1 使能,端口输出波形
        MOVIA      b'00001111'      ;//一个脉冲;滤波时间8us;
    
```

```

        MOVAR    PWM1DEADT
        MOVIA    b'00000001'      ;// PWM1PA/PWM1NA 用作PWM输出
        MOVAR    PWM1S
        MOVIA    50
        MOVAR    PWM1DH
        MOVIA    00
        MOVAR    PWM1DL

Main:                                       ;//程序主循环
        .....
        GOTO    Main

;//+++++
Interrupt:                               ;//中断子程序
        PUSH    ;//压栈,保存 A,STATUS
;//中断处理程序
        NOP
Interrupt_End:
        POP     ;//出栈,恢复 A,STATUS
        RETIE

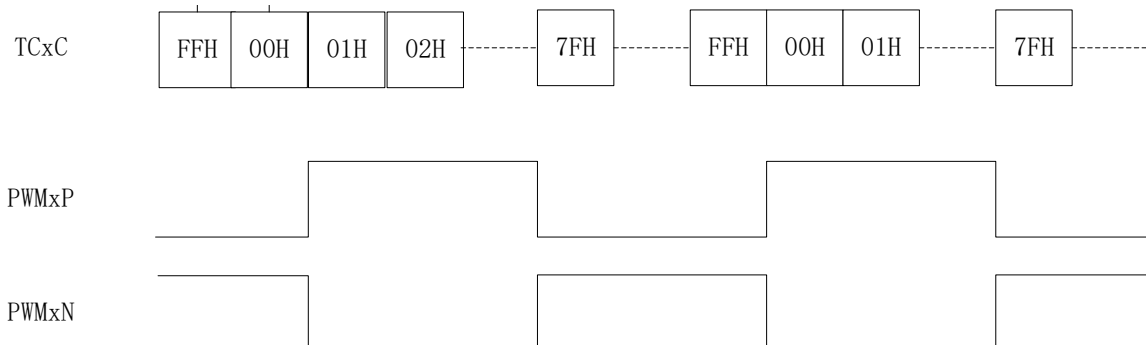
        END
    
```

注：单脉冲模式下，置 TC1 定时器时，钟源只能选用 Fosch，选用 Foscl、Fcpu 或 T1CKI 作为时钟源，无法产生相对应的现象

## 9.9 PWM 输出波形示例

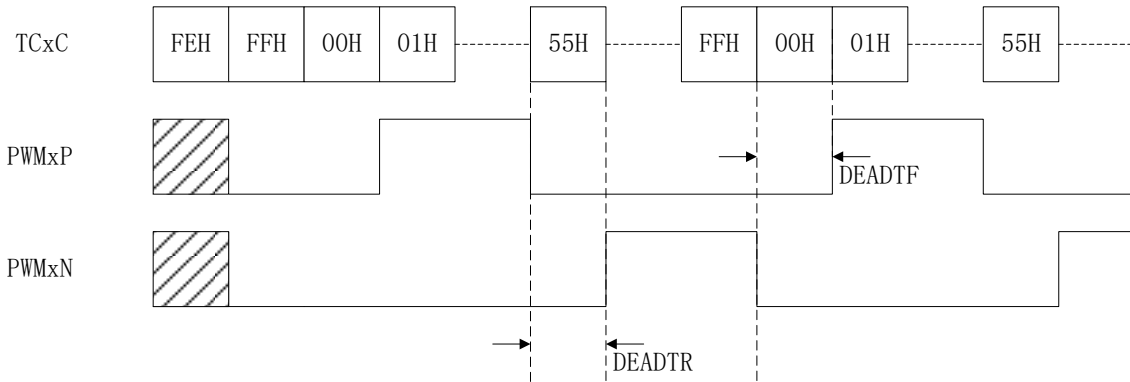
### 9.9.1 互补 PWM 输出

例：PWM1CR=11100000B，WM1DH=7FH，PWM1DL=0H，TC1CL=FFH



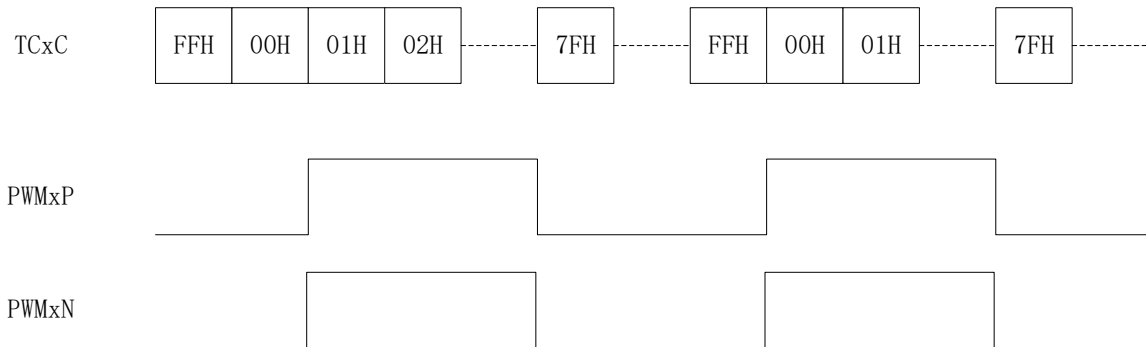
### 9.9.2 带死区的互补 PWM 输出

例：PWM1CR=11100100B，WM1DH=55H，PWM1DL=0H，TC1CL=FFH，WM1DEADT=00010001B



### 9.9.3 有效电平选择

例：PWM1CR=11101100B, PWM1DH=7FH, WM1DL=0H, TC1CL=FFH



### 9.10 PWM1 范例

例：以 TC1 时基，置的 16MHz RC 震荡电路提供振荡频率，置 50% 占空比，IOA0、IOA1 口输出 1kHz 波形

```

// ++++++
        ORG        0000H
        GOTO      Main_Program    ;//跳转到程序开始
        ORG        0008H
        GOTO      Interrupt      ;//发生中断后,跳转到中断子程序
// ++++++
Main_Program:                               ;//程序开始
PWM1_Init:                                  ;//PWM1初始化
; //1、IO 端口设置:
        BSET     OEA,0             ;//设置 IOA0、IOA1 为波形输出口
        BSET     OEA,1
; //2、时钟源设置:
        MOVIA    b'01001110'     ;// 8位模式,高频系统时钟,1:64分频
        MOVAR    T1CR             ;//将模式位配置字写入
        CLRR     TC1CH            ;//将 TC0CH 清零
    
```



```

MOVIA    249           ;//将249写入A
MOVAR    TC1CL        ;//16MHz/64/250=1KHz
;//3、PWM1设置:
MOVIA    b'01111000'
MOVAR    PWM1CR       ;// 端口输出PWM1P波形、PWM1N波形,高低电平有效
MOVIA    b'000000001' ;//PWM1PA/PWM1NA 用作PWM输出
MOVAR    PWM1S
MOVIA    0
MOVAR    PWM1DL       ;//低位清零
MOVIA    (249+1)/2    ;//产生占空比为50%的PWM波形
MOVAR    PWM1DH
;//4、使能PWM1、TC1:
BSET     T1CR,7       ;//使能定时器T0
BSET     PWM1CR,7     ;//使能PWM0
Main:
        .....
        GOTO     Main
;//+++++
Interrupt:
        PUSH     ;//压栈,保存 A,STATUS
;//中断处理程序
        NOP
Interrupt_End:
        POP      ;//出栈,恢复 A,STATUS
        RETIE
        END

```

# 10 通用串行通讯口 (USART)

## 10.1 概述

QT8S737 支持异步全双工模式和同步半双工模式。

## 10.2 TXCR 发送控制寄存器

0DAH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXCR	TXEN	TMCLR	SYNC	TX9	SLAVE	SPD1	SPD0	TXD9
读/写	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	1	0	0	0	0	0	0

- Bit 7      **TXEN:**  使能发送  
           1 = 使能USART发送功能  
           0 = 屏蔽USART发送功能
- Bit 6      **TMCLR:**  发送寄存器空标志  
           1 = 数据已发送, 移位寄存器空  
           0 = 正在发送数据, 移位寄存器不空
- Bit 5      **SYNC:**  同步模式  
           1 = 选择同步模式  
           0 = 选择异步模式
- Bit 4      **TX9:**  数据长度选择  
           1 = 9位数据  
           0 = 8位数据
- Bit 3      **SLAVE:**  同步发送/接收模式  
           1 = SLAVE  
           0 = Master

Bit[2:1]    **SPD[1:0]:**  发送接收速度选择

SPD[1:0]	波特率分频比(n)
11	256
10	64
01	16
00	4

Bit 0      **TXD9:**  发送数据第9位数据

## 10.3 TXREG 发送数据寄存器

0DBH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXREG	TX0D7	TX0D6	TX0D5	TX0D4	TX0D3	TX0D2	TX0D1	TX0D0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

## 10.4 RXCR 接收控制寄存器

0DCH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RXCR	RXEN	CKPS	-	RX9	SREN	RXOVF	FRER	RXD9
读/写	R/W	R/W	-	R/W	R/W	R	R	R
复位后	0	0	-	0	0	0	0	0

- Bit 7      **RXEN:** 使能接收  
           1 = 使能USART接收功能  
           0 = 屏蔽USART接收功能
- Bit 6      **CKPS:** 同步模式时钟模式选择  
           1 = 下降沿发送数据  
           0 = 上升沿发送数据
- Bit 4      **RX9:** 数据长度选择  
           1 = 9位数据  
           0 = 8位数据
- Bit 3      **SREN:** 同步接收开始  
           1 = 开始同步接收, 单字节接收模式下接收完一个字节自动清零  
           0 = 停止同步接受
- Bit 2      **RXOVF:** 接收缓冲区溢出标志  
           1 = 接收缓冲区溢出, 读缓冲区自动清零  
           0 = 接收缓冲区未发生溢出
- Bit 1      **FRER:** 接收数据格式错  
           1 = 当前接收数据格式错 (未收到停止位)  
           0 = 当前接收数据未发生格式错
- Bit 0      **RXD9:** 接收数据第9位数据

## 10.5 RXREG 接收数据寄存器

0DDH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RXREG	RX0D7	RX0D6	RX0D5	RX0D4	RX0D3	RX0D2	RX0D1	RX0D0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

## 10.6 BRGDH 波特率寄存器高位

0DEH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BRGDH	SBYTE	-	-	-	-	-	BRGD9	BRGD8
读/写	R/W	-	-	-	-	-	R/W	R/W
复位后	0	-	-	-	-	-	0	0

- Bit 7      **SBYTE:** 同步接收模式选择  
           1 = 单字节接收, 接收完一个字节后自动清除SREN  
           0 = 多字节接收

## 10.7 BRGDL 波特率寄存器低位

0DFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BRGDL	BRGD7	BRGD6	BRGD5	BRGD4	BRGD3	BRGD2	BRGD1	BRGD0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

## 10.8 USART 使用说明

### 10.8.1 波特率设置

通过设置 BRGD 和 SPD 来获得所需的波特率。

波特率计算公式：标波特率 =  $F_{osc} / ((BRGD+1) \times n)$

常用波特率设置 ( $F_{osc}=16MHz$ )

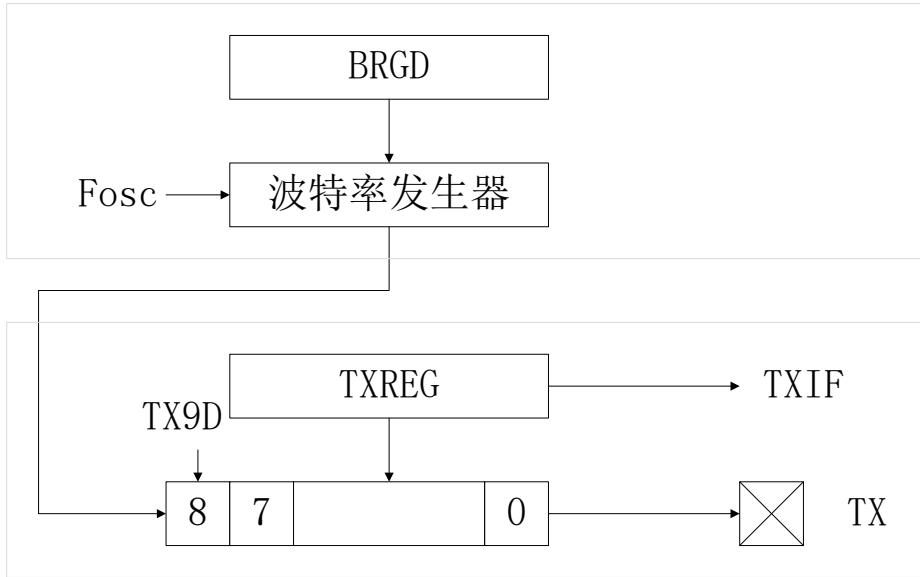
常用波特率	N (波特率分频比)	BRGD	偏差
300	256	0xCF	0.17%
600	256	0x67	0.17%
1200	64	0xCF	0.16%
2400	64	0x67	0.16%
4800	16	0xCF	0.16%
9600	16	0x67	0.16%
19200	4	0xCF	0.16%
38400	4	0x67	0.16%
57600	4	0x44	0.64%
115200	4	0x21	2.12%

### 10.8.2 异步发送

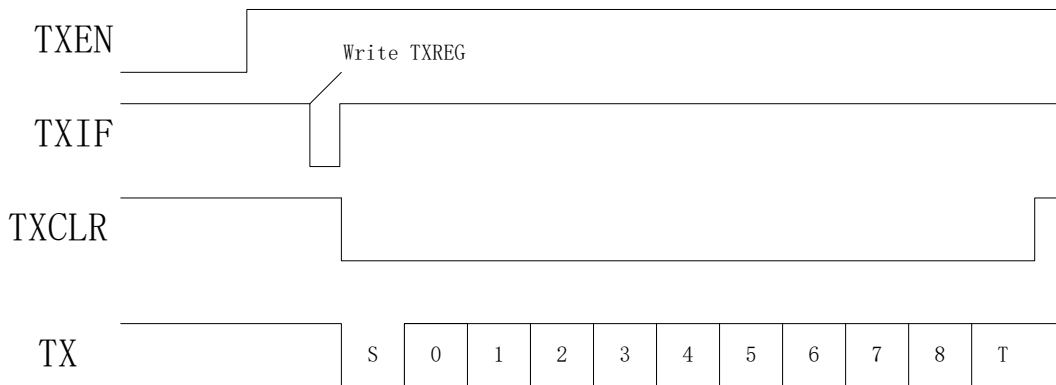
当 TXEN 使能时，XIF 中断标志为 1 说明 TXREG 发送寄存器为空，MCLR=1 说明发送移位寄存器为空，送器处于空闲状态。

空闲状态时写入 TXREG，入数据将立即装载到发送移位寄存器中，此时，XIF 为 1，MCLR=0，发送器进入发送状态。此时再次写入 TXREG，TXIF 将清零，说明 TXREG 有未发送数据，发送移位寄存器发送完毕后，TXREG 数据将自动载入发送移位寄存器继续发送，且 TXIF 为空。

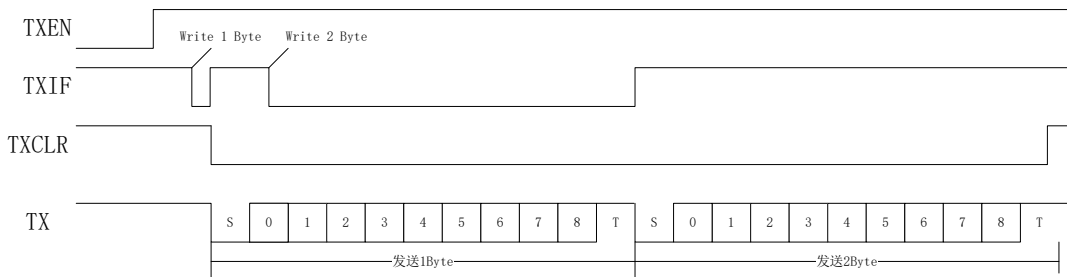
当 TXIF 为 0 时写入 TXREG，将覆盖上次写入数据。



单字节发送:



多字节发送:



- STEP1: 设置波特率 SSPD=X, BRGD=X, SYNC=0
- STEP2: 设置 TXEN=1, 设置数据模式 TX9=X
- STEP3: 写入数据高位 TXD9
- STEP4: 写入 TXREG, 启动发送
- STEP5: 当 TXIF=1 时, 写入 TXREG 发送下一个字节
- STEP6: 重复 STEP5, 直到该帧数据发送完成

**例: USART 异步发送**

```

;///+++++
                ORG        0000H
                GOTO      Main_Program    ;//跳转到程序开始
                ORG        0008H
                GOTO      Interrupt       ;//发生中断后,跳转到中断子程序
;///+++++
Main_Program:   ;//程序开始
USART_Init:    ;//USART初始化
;///1、端口设置
                BSET      OEB,3           ;//0:输入 1:输出
;///2、TXCR 设置:
                MOVIA     b'10000000'
                MOVAR     TXCR           ;//使能USART发送功能
;///3、波特率设置:
                MOVIA     0x00           ;//目标波特率 = Fosc/ ((BRGD+1) *n)
                MOVAR     BRGDH         ;//波特率寄存器高位清零
                MOVIA     0x67
                MOVAR     BRGDL         ;//波特率 38400
;///4、数据发送
                MOVIA     0x55           ;//以发送数据55为例
                MOVAR     TXREG
                JBTS1     TXCR,6
                GOTO      $-1
Main:           ;//程序主循环
                .....
                GOTO      Main
;///+++++
Interrupt:     ;//中断子程序
                PUSH      ;//压栈,保存 A,STATUS
;///中断处理程序
                NOP
Interrupt_End: ;//出栈,恢复 A,STATUS
                POP
                RETIE
                END
    
```



**例: USART 异步接收**

```

;+++++
        ORG          0000H
        GOTO        Main_Program    ;//跳转到程序开始
        ORG          0008H
        GOTO        Interrupt       ;//发生中断后,跳转到中断子程序
;+++++
Main_Program:                ;//程序开始
USART_Init:                  ;//USART初始化
;//1、端口设置
        BCLR        OEB,4           ;//0:输入 1:输出
;//2、TXCR 设置:
        MOVIA       b'00000000'
        MOVAR       TXCR           ;//屏蔽USART发送功能
;//3、RXCR 设置:
        MOVIA       b'10000000'
        MOVAR       RXCR           ;//使能USART接收功能
;//4、波特率设置:           ;//目标波特率 = Fosc / ((BRGD+1) * n)
        MOVIA       0x00
        MOVAR       BRGDH          ;//波特率寄存器高位清零
        MOVIA       0x67
        MOVAR       BRGDL          ;//波特率 38400
Main:                          ;//程序主循环
        .....
        GOTO        Main
;+++++
Interrupt:                    ;//中断子程序
        PUSH        ;//进栈、保存 A、STATUS
;//中断处理程序
        JBTS1       INTF0,5
        GOTO        Interrupt_End
        MOVR        RXREG,RXIF     ;// 读取 RXREG
Interrupt_End:
        POP         ;//出栈、恢复 A、STATUS
        RETIE
        END
    
```

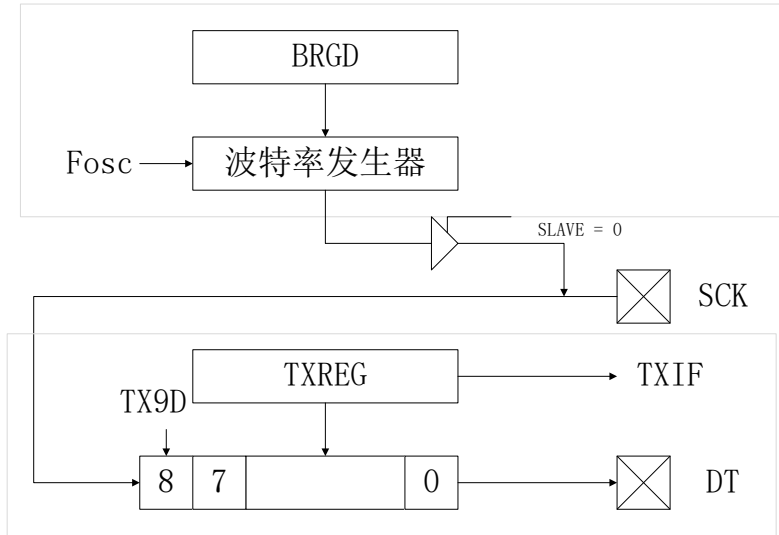


### 10.8.4 同步发送

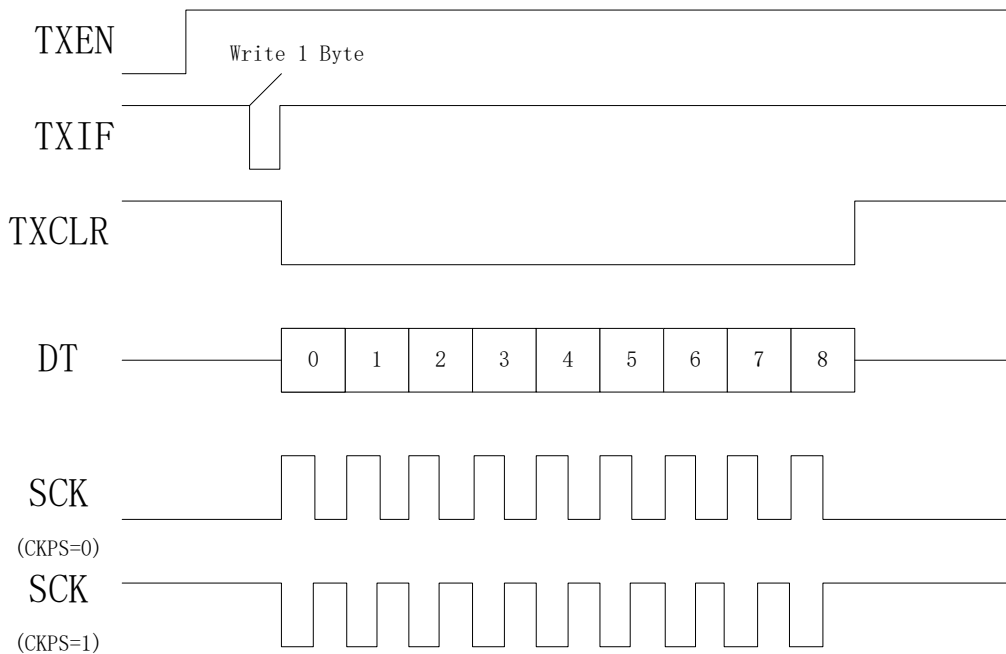
当 TXEN=1, SYNC=1 时, 使能同步发送功能。CKPS 选择发送时钟极性, TXIF 中断标志为 1 说明 TXREG 发送寄存器为空, TMCLR=1 说明发送移位寄存器为空, 发送器处于空闲状态。

空闲状态写入 TXREG, 写入数据将立即装载到发送移位寄存器中, 此时, TXIF 为 1, TMCLR=0, 发送器进入发送状态。此时再次写入 TXREG, TXIF 将清零, 说明 TXREG 有未发送数据, 发送移位寄存器发送完毕后, TXREG 数据将自动载入发送移位寄存器继续发送, 且 TXIF 为空。

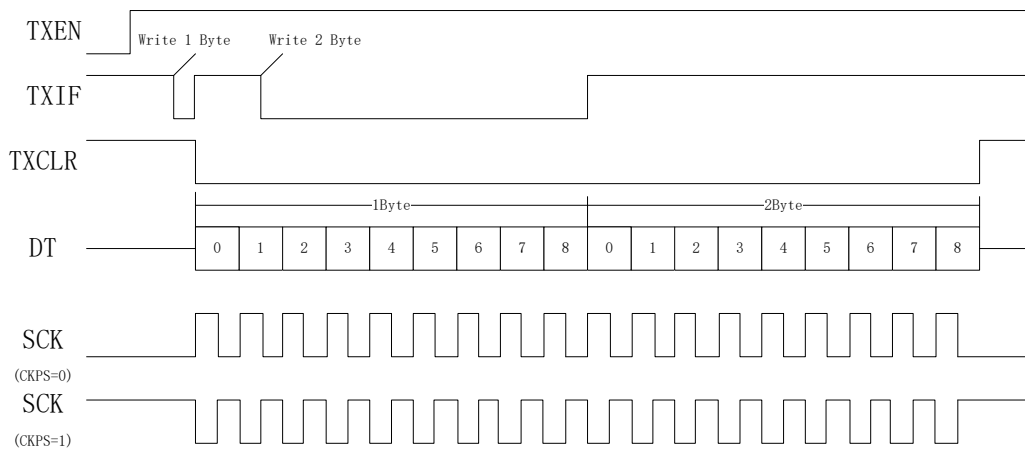
当 TXIF 为 0 时写入 TXREG, 将覆盖上次写入数据。



单字节发送:



多字节发送:



参考操作步骤 SLAVE=0:

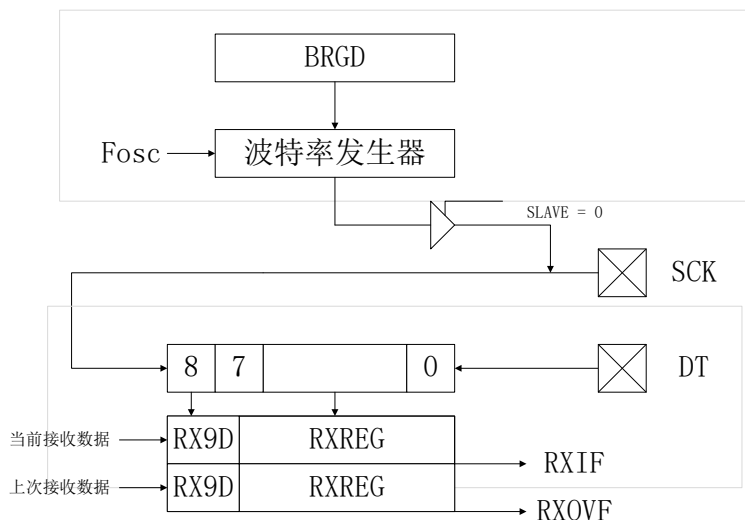
- STEP1: 设置波特率 SSPD=X, BRGD=X, SYNC=1
- STEP2: 设置 TXEN=1, 设置数据模式 TX9=X
- STEP3: 写入数据高位 TXD9
- STEP4: 写入 TXREG, 启动发送
- STEP5: 当 TXIF=1 时, 写入 TXREG 发送下一个字节
- STEP6: 重复 STEP5, 直到该帧数据发送完成

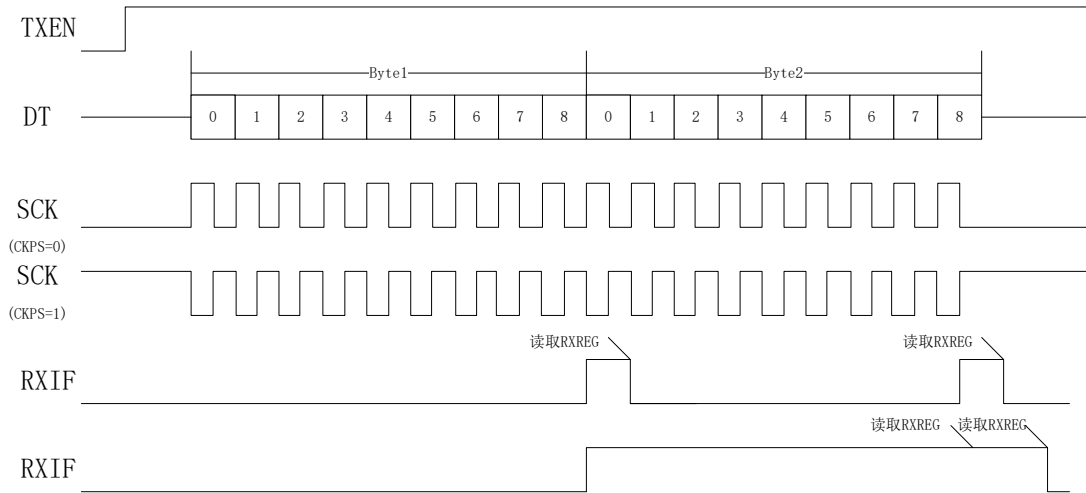
参考操作步骤 SLAVE=1:

- STEP1: 设置波特率 SSPD=X, BRGD=X, SYNC=1
- STEP2: 设置 TXEN=1, 设置数据模式 TX9=X
- STEP3: 当 TXIF=1 时, 写入数据高位 TXD9
- STEP4: 写入 TXREG 等待发送下一个字节
- STEP5: 重复 STEP3-4, 直到该帧数据发送完成

## 10.8.5 同步接收

设置同步 SYNC=1 模式, 使能 RXEN, 开始启动同步接收。RX 管脚处于高电平时, 接收器处于空闲状态, 当检测到 RX 变为低电平, 接收器检测该低电平是否有效起始位, 若为有效起始位, 则启动数据时钟恢复电路和数据恢复电路进行接收。1 个数据接收完成后, RXIF 置 1, 当接收 3 个数据未读取, RXOVF 置 1, 同时舍弃第三个接收数据。完全读取 RXREG 后 RXIF 自动清零。





参考操作步骤 SLAVE=0:

STEP1: 设置波特率 SSPD=X, BRGD=X, SYNC=0

STEP2: 设置 RXEN=1

STEP3: 写 SREN 启动接收

STEP4: 等待接收完成 RXIF=1

STEP5: 读取 RX9D

STEP6: 读取 RXREG, 单字节接收 (SBYTE=1) 重复 3-6; 多字节接收 (SBYTE=0) 重复 4-6

参考操作步骤 SLAVE=1:

STEP1: 设置波特率 SSPD=X, BRGD=X, SYNC=0

STEP2: 设置 RXEN=1

STEP3: 写 SREN 启动接收

STEP4: 等待接收完成 RXIF=1

STEP5: 读取 RX9D

STEP6: 读取 RXREG, 单字节接收 (SBYTE=1) 重复 3-6; 多字节接收 (SBYTE=0) 重复 4-6

## 10.8.6 唤醒及休眠模式下通讯

TXIE 置 1 时, TXIF 中断标志唤醒 CPU

RXIE 置 1 时, RXIF 中断标志唤醒 CPU

异步接收时, 检测到 START 位将自动使能高频振荡器, 接收完成后唤醒 CPU

同步接收时, 若作为主机, 则休眠状态下部工作; 作为从机, 则接收 1 个字节完成后唤醒 CPU

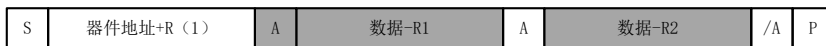
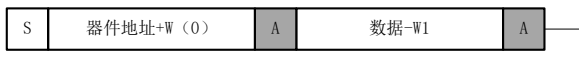
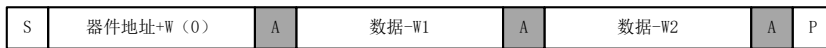
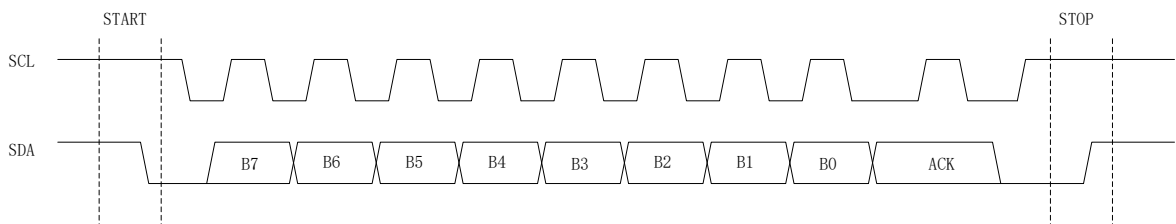
# 11 串行通讯口 (I2C)

## 11.1 概述

QT8S737 支持高速 I2C (400K) Slave。

注：(1) CPU 时钟选择 2T 时不支持。

## 11.2 通讯波形示意



### 11.3 I2CCON I2C 控制寄存器

0D6H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
I2CCON	I2CEN	R_W	D_A	BF	NACK	-	IICM	IICPS
读/写	R/W	R	R	R	R	-	R/W	R/W
复位后	0	0	0	0	0	-	0	0

- Bit 7      **I2CEN:** 使能发送  
 1 = 使能串行通讯功能  
 0 = 屏蔽串行通讯功能
- Bit 6      **R\_W:** 发送寄存器空标志  
 1 = Master读数据 (主机发接收数据命令)  
 0 = Master写数据 (复位后、START、STOP后)
- Bit 5      **D\_A:** 数据地址标志  
 1 = 主机发送的是数据(Start后发过的第一个数据)  
 0 = 主机发送的是地址
- Bit 4      **BF:** 数据缓冲区满  
 R\_W = 0  
         1 = 从机接收到数据, 未读  
         0 = 已读或未接受到数据  
 R\_W = 1  
         1 = 有数据待发送  
         0 = 数据已发送或正在发送
- Bit 3      **NACK:** Master读数据ACK  
 1 = Master停止接收数据  
 0 = Master继续接收下一个数据
- Bit 1      **IICM:** IIC 总线模式  
 1 = 从机在未完成接收(未读 I2CBUF)时 HOLD 总线(拉低 I2C\_SCL)  
 0 = 从机在未完成接收不 HOLD 总线(高速 Master 可能会造成通讯紊乱)
- Bit 0      **IICPINS:** I2C总线脚位选择  
 1 = 选择B组总线(I2C\_SCLB, I2C\_SDAB)  
 0 = 选择A组总选(I2C\_SCL\_A, I2C\_SDAA)

### 11.4 I2ADDR I2C 控制寄存器

0D5H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
I2CADR	I2CA7	I2CA6	I2CA5	I2CA4	I2CA3	I2CA2	I2CA1	I2CA0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
复位后	0	0	0	0	0	0	0	0

- Bit[7:0]    **I2CA[7:0]:** 从机地址  
 地址(I2C 写入): I2CADR  
 地址(I2C 读取): I2CADR+1

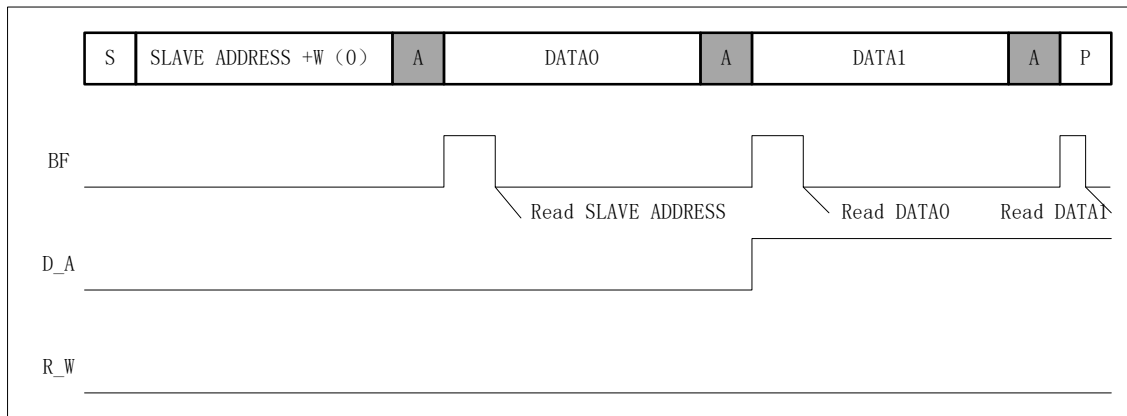
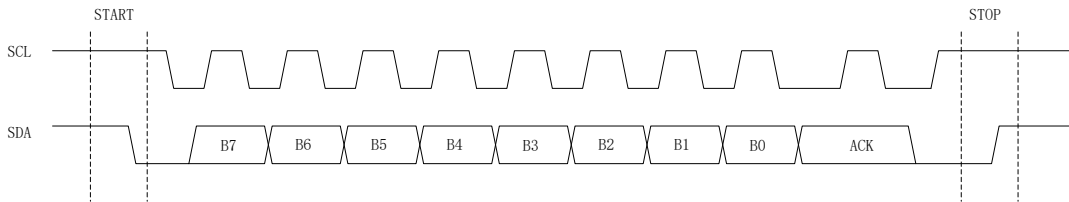
## 11.5 I2CBUF 数据寄存器

0D7H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
I2CBUF	I2CBUF7	I2CBUF6	I2CBUF5	I2CBUF4	I2CBUF3	I2CBUF2	I2CBUF1	I2CBUF0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

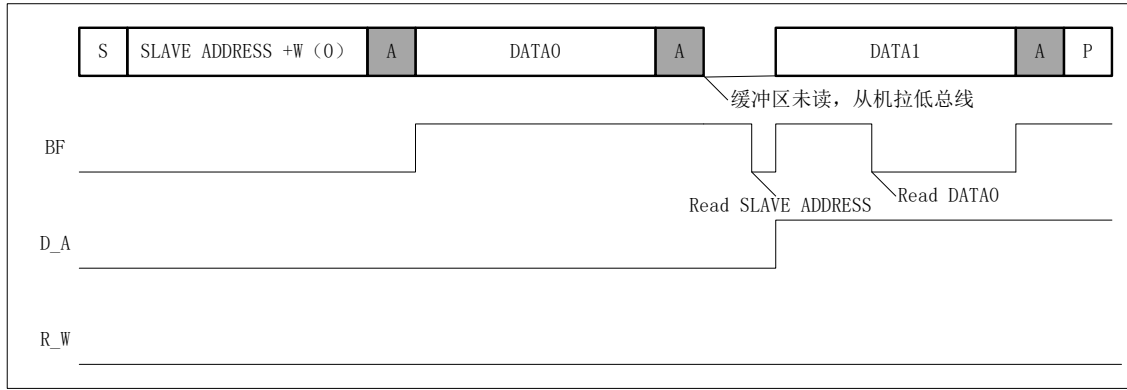
## 11.6 唤醒及休眠模式下通讯

当 I2CCON.7 为 1 时，SCL、SDA 线的低电平会唤醒 CPU，并开始通讯，通讯期间 CPU 无法进入休眠模式。

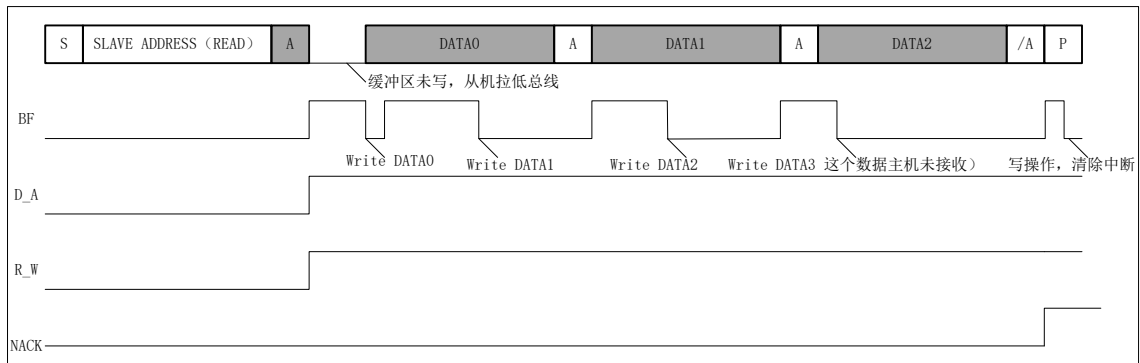
## 11.7 通讯波形图



地址匹配后，主机向从机写入两个数据



两个数据没有读取, 从机拉低 SCL, 等待数据读取

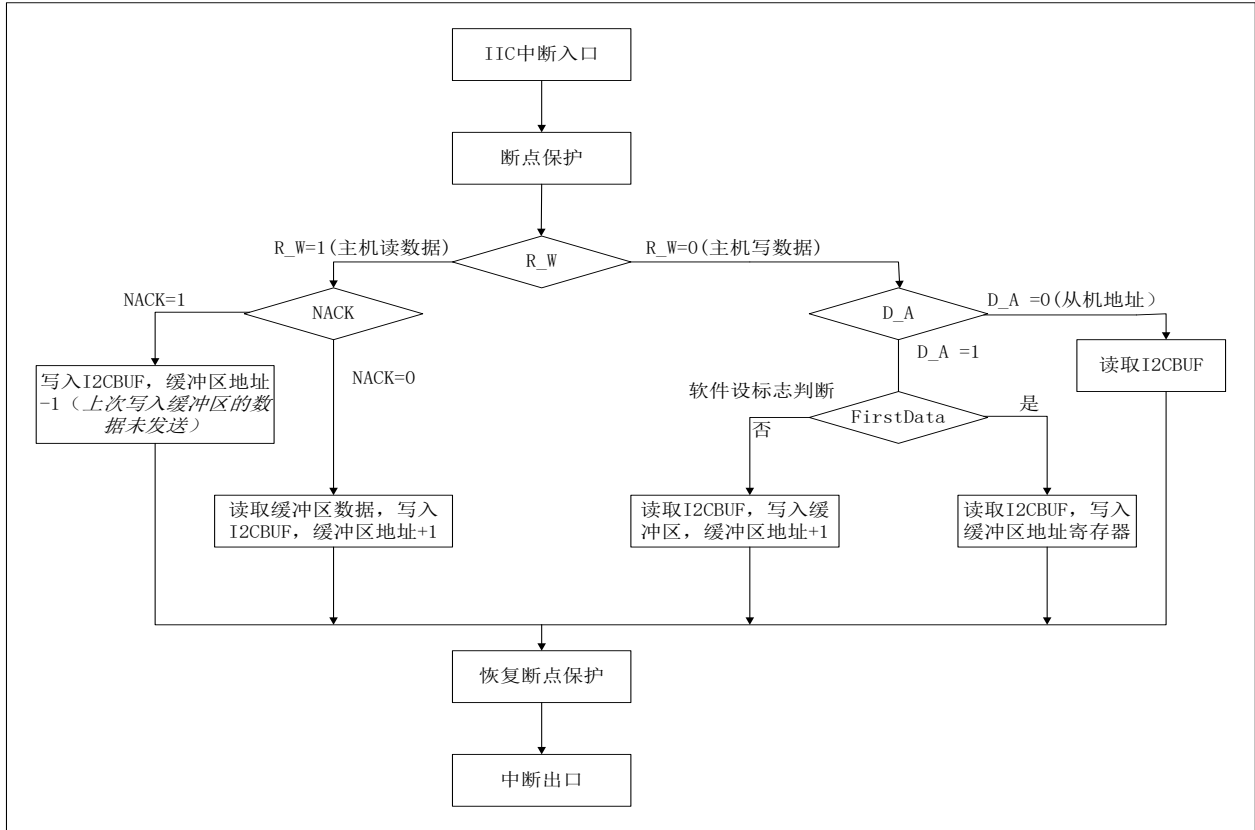


I2C Master 读数据操作

## 11.8 应用示例

M8E625 作为从机，允许主机通过 I2C 接口对其 RAM 进行读写操作。

### 11.8.1 从机软件流程图



### 11.8.2 例程

```

//+++++
LIST P= QT8S737
#include QT8S737.inc
//+++++
#define      bFirstByte      0x0f,0      //设置软件标志
#define      SlaveBuffAddr   0x0e      //缓冲区地址
#define      StartAdr 0x10
//+++++
        ORG      0
        GOTO     MainProgram
//+++++
//中断入口
        ORG      8
        GOTO     Interrupt
//+++++
    
```



**MainProgram:**

```

IIC_Init:                                     ;//IIC 初始化
    BSET      PUB,0                            ;//上拉使能
    BSET      PUB,1
    BCLR      OEB,0
    BCLR      OEB,1
    MOVIA     b'10000010'
    MOVAR     I2CCON
    MOVIA     0xA2                             ;//IICSLAVE 为 0xA2(写)/0xA3 (读)
    MOVAR     I2CADR
    BSET      bI2CIE                           ;//使能 IIC 中断
    BSET      OPTION,GIE
    
```

**Main:**

```

    CLRWDT
    GOTO      Main

;//+++++
;//中断处理子程序
    
```

**Interrupt:**

```

    PUSH
    MOVR      FSR0,A
    MOVAR     sFSR0                            ;//中断中用到 FSR0,保护
    JBTS0    intf2,0
    GOTO      sspint
    
```

**Interrupt\_End:**

```

    BCLR      FLAG
    MOVR      sFSR0,A
    MOVAR     FSR0                            ;//中断中用到 FSR0,保护
    POP
    RETIE
    
```

**sspint:**

```

    BSET      FLAG
    JBTS1    I2CCON,R_W ;//R_W
    GOTO      iicMasterWrite
    
```

**iicMasterRead:**

```

    JBTS0    I2CCON,NACK
    GOTO      iicMasterReadNoAck
    MOVR      SlaveBuffAddr,A
    MOVAR     FSR0
    MOVR      INDF0,A
    MOVAR     I2CBUF
    INCR      SlaveBuffAddr,R
    GOTO      Interrupt_End
    
```

**iicMasterReadNoAck:**

```

    ;//Master 读数据完毕
    DECR      SlaveBuffAddr,R                ;//上次写入数据未接收,地址减 1
    
```

```

    MOVAR    I2CBUF          ;//清除中断
    GOTO     Interrupt_End

iicMasterWrite:
    JBTS0    I2CCON,D_A
    GOTO     iicGetBufAddr

iicAddr:
    MOVR     I2CBUF,A        ;//读数据,清空缓冲区
    BSET     bFirstByte
    GOTO     Interrupt_End

iicGetBufAddr:
    JBTS1    bFirstByte
    GOTO     iicGetData
    MOVR     I2CBUF,A
    MOVAR    SlaveBuffAddr
    BCLR     bFirstByte
    GOTO     Interrupt_End

iicGetData:
    MOVR     SlaveBuffAddr,A
    MOVAR    FSR0
    MOVR     I2CBUF,A
    MOVAR    INDF0
    INCR     SlaveBuffAddr,R
    GOTO     Interrupt_End

;//+++++

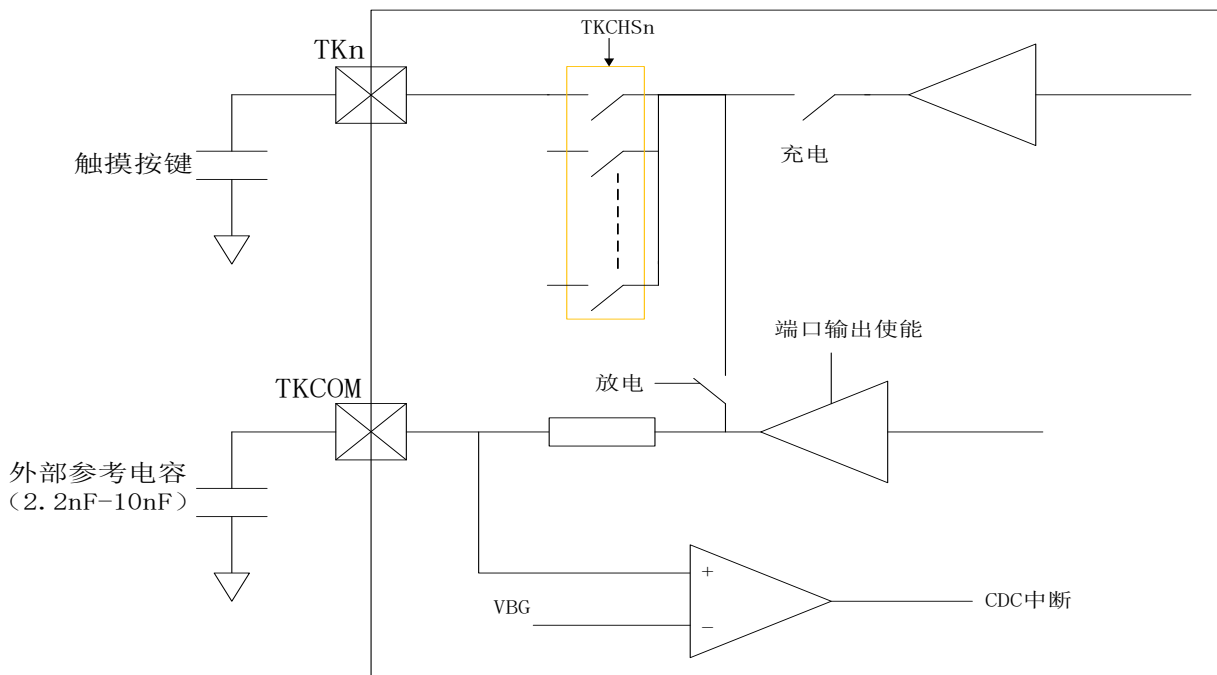
```

# 12 触摸按键 (CDC)

## 12.1 概述

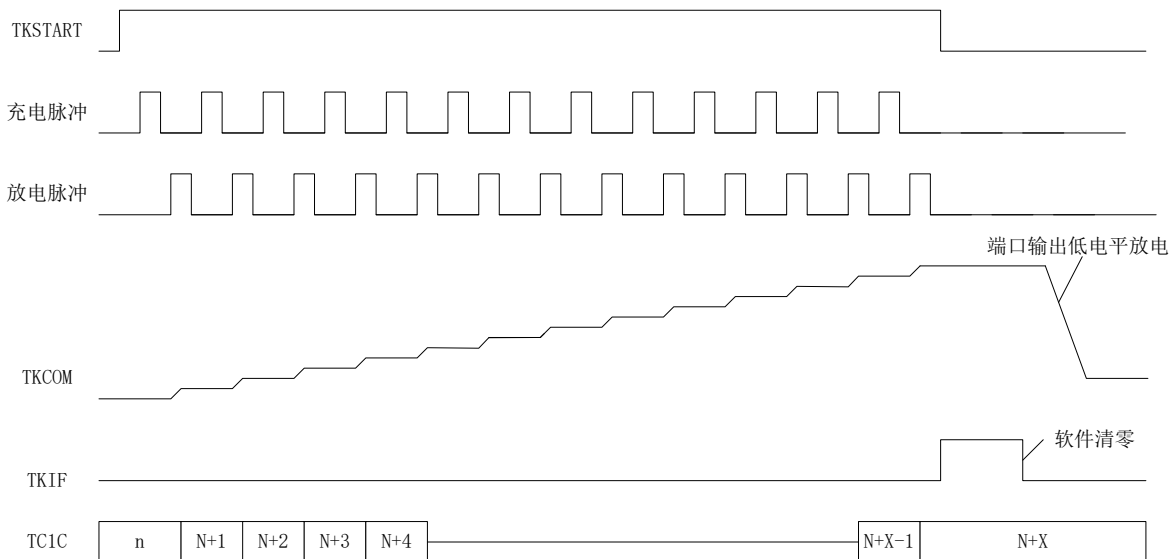
QT8S737 有 11 路触摸按键通道，灵敏度可通过外接电容调节，可替代机械式触摸按键，实现防水防尘，简单易用的操作接口。

## 12.2 原理框图



注：外部参考电容为 NPO 或 X7R 等温度系数较良好的电容。

信号波形示意图:



## 12.3 TKCTR0 控制寄存器 0

0B2H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKCTR0	TKEN	TKSTART	TKCKS1	TKCKS0	-	-	-	-
R/W	R/W	R/W	R/W	R/W	-	-	-	-
POR	0	0	0	0	-	-	-	-

Bit 7      **TKEN:** CDC模块使能控制位

1 = 使能CDC模块

0 = 关闭CDC模块

Bit 6      **TKSTART:** 启动通道转换

1 = 启动通道转换

0 = 通道转换完成, 自动清零

Bit [5:4]      **TKCKS[1:0]:** CDC时钟分频选择位

TKCKS[1:0]	输入信号选择
00	Fosch/4
01	Fosch/8
10	Fosch/16
11	Fosch/32

## 12.4 TKCTR1 控制寄存器 1

0B3H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKCTR1	-	-	-	-	CAPS3	CAPS2	CAPS1	CAPS0
R/W	-	-	-	-	R/W	R/W	R/W	R/W
POR	-	-	-	-	0	0	0	0

Bit [3:0]      **CAPS:** 内嵌电容选择

## 12.5 TKCHSH/L 触摸按键通道选择寄存器

0B6H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKCHSL	TKCHS7	TKCHS6	TKCHS5	TKCHS4	TKCHS3	TKCHS2	TKCHS1	TKCHS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

0B7H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKCHSH	-	-	-	-	-	TKCHS10	TKCHS9	TKCHS8
R/W	-	-	-	-	-	R/W	R/W	R/W
POR	-	-	-	-	-	0	0	0

Bit [7:0] **TKCHSn:** 触摸通道连通

1 = 连通该通道

0 = 不连通该通道

注：(1) 可同时连通 0-11 个通道；用于检测组合键，按键唤醒等

(2) 连通的端口在 TKEN 有效时自动设置成模拟输入口

## 12.6 TKCNTH/L 触摸按键计数寄存器

0B4H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKCNTL	TKCNTL[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	X	X	X	X	X	X	X	X

0B5H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKCNTH	TKCNTH[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	X	X	X	X	X	X	X	X

注：按键扫描过程中，不允许写操作

## 12.7 操作说明

- 1- 开启触摸模块电源
- 2- 等待触摸模块电源稳定 ( $>100\mu\text{S}$ )
- 3- 使能 CDC 模块  $\text{TKEN}=1$
- 4- CDC 相关通道/转换时钟设置
- 5- 设置 TKCOM 管脚输出 0, 对外接电容放电 ( $>10\mu\text{S}$ )
- 6- 设置 TKCOM 管脚为输入模式
- 7- 清除触摸按键计数寄存器  $\text{TKxCNTH}/\text{TKxCNTL}$
- 8- 启动 CDC 转换 ( $\text{TKSTART}$  置 1)
- 9- 等待转换完成 ( $\text{TKSTART}==0$ ) /或使用中断模式 ( $\text{TKIF}$ )
- 10- 读取  $\text{TKCNTH}/\text{TKCNTL}$  的计数值, 判断是否有按键发生
- 11- 重复 4-10 对不同通道进行扫描

注: 触摸库函数请到公司网站查阅

# 13 模数转换器(ADC)

## 13.1 概述

QT8S737有一个12路外部通道(AIN0~AIN11)和3路内部通道(VDD\_DIV, VREF和GND) 12位分辨率的A/D转换器,可以将模拟信号转换成12位数字信号。进行AD转换时,首先要选择输入通道,然后启动AD转换。转换结束后,系统自动将EOC设置为“1”,并将转换结果存入寄存器ADH和寄存器ADL中。

## 13.2 ADCON0 寄存器

0E0H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADON	ADS	ADEOC	ADFM	CHS3	CHS2	CHS1	CHS0
读/写	R/W	R/W	R	R/W	RW	R/W	R/W	R/W
复位后	0	0	1	0	1	0	1	0

- Bit 7      **ADON:** ADC使能控制位  
 1 = 使能ADC  
 0 = 关闭ADC
- Bit 6      **ADS:** ADC 启动位  
 1 = 开始(每次写入1将重新启动ADC)  
 0 = 停止,转换完成自动清零
- Bit 5      **ADEOC:** ADC 状态控制位  
 1 = 转换结束,ADS 复位  
 0 = 转换过程中
- Bit 4      **ADFM:** 数据格式选择位  
 1 = ADRES = {ADH[3:0], ADL[7:0]}; ADH[7:4] = 0  
 0 = ADRES = {ADH[7:0], ADL[7:4]}; ADL[3:0] = 0
- Bit[3:0]   **CHS[3:0]:** ADC 输入通道选择位  
 [0000] ~ [1011] = AIN0 ~ AIN11  
 [1100] = VDD/4  
 [1101] = 内建 VREF 基准电平  
 [1111] = GND

注: (1) 若 ADENB = 1, 用户应设置 IOA.n/AINn 为无上拉电阻的输入模式, 系统不会自动设置。若已经设置了 ANSEL.n, IOA.n/AINn 的数字 I/O 功能(都是隔离开来的)。

(2) 休眠模式下, 关掉 ADEN, ADC 输入通道选择位不能为 VDD/4。

### 13.3 ADCON1 寄存器

0E1H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	-	ADCKS2	ADCKS1	ADCKS0	VREMS1	VREMS0	VHS1	VHS0
读/写	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	0	0	0	0	0	0	0

Bit6:4] **ADCKS[2:0]:** ADC 时钟源选择位

ADCKS[2:0]	ADC 时钟源选择
000	Fcpu
001	Fcpu/2
010	Fcpu/4
011	Fcpu/8
100	Fcpu/16
101	Fcpu/32
110	Fcpu/64
111	

Bit[3:2] **VREMS[1:0]:** ADC 参考电压模式选择位

VREMS[1:0]	ADC 参考电压模式
00	VDD
01	内部参考电压
10	外部参考电压
11	内部参考与外部参考连接

Bit[1:0] **VHS[1:0]:** ADC 内建基准电平选择位

VHS[1:0]	内建 VREF 基准电平
00	关闭内部参考
01	2.0V
10	3.0V
11	4.0V

注：(1) 若由 VHS[1:0]控制选择的内部 VREF 电平高于 VDD，内部 VREF 为 VDD。

例：VHS[1:0] = 11 (内部 VREF = 4.0V)，VDD = 3.0V，则实际内部 VREF = 3.0V。

(2) 休眠模式下 VHS[1:0]为 00，即关闭内部参考。

### 13.4 ADCON2 寄存器

0E4H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON2	-	-	-	-	ADVOS3	ADVOS2	ADVOS1	ADVOS0
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit[3:0] **ADVOS[3:0]:** ADC失调补偿寄存器



### 13.4.1 ADC 调校功能

ADC通道选择内部GND通道，先设置ADCON2为0x00，若ADC的GND通道转换值不为0，就把ADCON2加1，直到ADC的GND通道转换值为0时，ADCON2的值就是调校好的值，ADCON2值最大等于15。

```

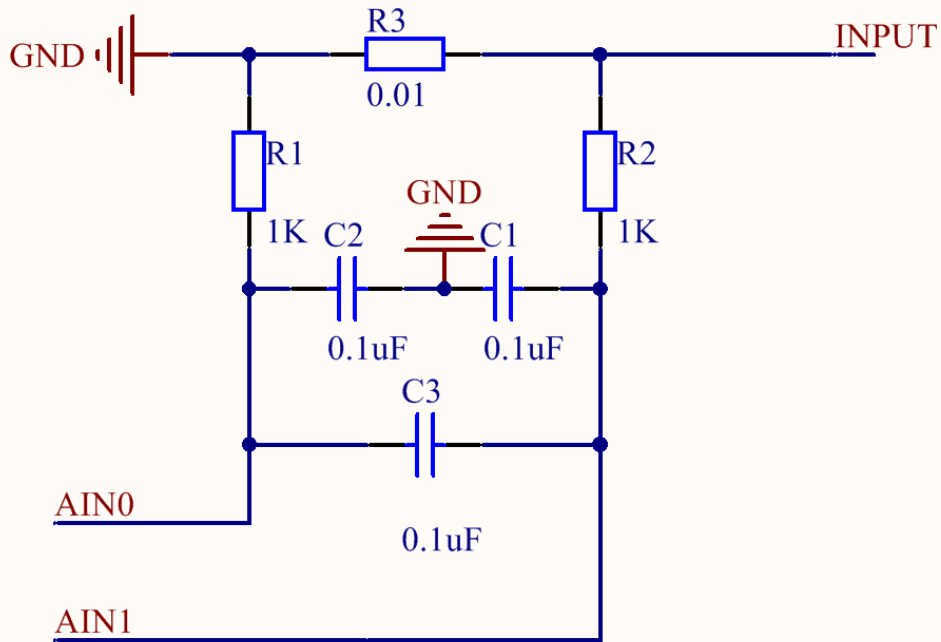
Main_Program:                                ;//程序开始
ADC_Init:
;//1、ADCON0 设置
        MOVIA    b'10011111'                ;//选择内部GND通道
        MOVAR    ADCON0
;//2、ADCON1 设置
        MOVIA    b'00000000'
        MOVAR    ADCON1
;//3、ADCON2 设置
        MOVIA    b'00000000'
        MOVAR    ADCON2
;//4、开启转换
ADC_CHANGE:
        BSET     ADCON0,6
        JBTS0    ADCON0,6
        GOTO     $-1
        MOVIA    0x00
        JCMPAR   ADH
        GOTO     ADC_CHANGE
        JCMPAR   ADL
        GOTO     $+2
        GOTO     Main                        ;//此时 ADCON2 的值调校好的值
        INCR     ADCON2,R
        GOTO     ADC_CHANGE
Main:
        .....
        GOTO     Main
    
```

### 13.4.2 ADC 小信号采集

将 ADCON2 设置为 0x0F，分别采集小信号通道的 ADC 值和内部 GND 通道的 ADC 值，将采集的小信号 ADC 值减去内部 GND 通道的 ADC 值，即为实际的小信号 ADC 值。

### 13.4.3 ADC 差分测量参考

如下图所示电路图，是一个测量经过 R3 电流的差分测试实例图，通过已知的 R3 的电阻值，根据 R3 两端的 ADC 通道采集的数据的差值（以 AIN0，AIN1 为例），计算出经过 R3 的电压值，进而可以算出经过 R3 的电流值。



### 13.5 ADH ADC 数据高位

0E3H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADH	-	-	-	-	-	-	-	-
读/写	R	R	R	R	R	R	R	R
复位后	X	X	X	X	X	X	X	X

### 13.6 ADL ADC 数据低位

0E2H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADL	-	-	-	-	-	-	-	-
读/写	R	R	R	R	R	R	R	R
复位后	X	X	X	X	X	X	X	X

注：ADH/ADL 的数据格式与 ADM 相关，当 ADFM=1 时，ADH[7:4] = 0，ADH[3:0] 存放高四位数据，ADL[7:0] 存放低 8 位数据；当 ADFM=0 时，ADH[7:0] 存放高 8 位数据，ADL[7:4] 存放低 4 位数据，AD[3:0] = 0。

## 13.7 AD 转换时间

12 位 AD 转换时间 =  $1/(\text{ADC clock}) * 16 \text{ sec}$

## 13.8 ADC 范例

例: ADC 模数转换器

将高位的值存在寄存器 ADC\_DATAH, 低位存在寄存器 ADC\_DATAH。

```

ADCDATAH    EQU    '00'           ;//特殊寄存器定义声明
ADCDATAH    EQU    '01'
;// ++++++
                ORG        0000H
                GOTO        Main_Program    ;//跳转到程序开始
                ORG        0008H
                GOTO        Interrupt       ;//发生中断后,跳转到中断子程序
;//+++++
Main_Program:                ;//程序开始
ADC_Init:                    ;//ADC初始化
;//1、端口设置
                BCLR        OEA,1
                BSET        ANSA,1         ;//AIN1设置为模拟输入
;//2、ADCON0 设置
                MOVIA       b'10010001'   ;//使能ADC,输入通道选择AIN1
                MOVAR       ADCON0
;//3、ADCON1 设置
                MOVIA       b'00000000'
                MOVAR       ADCON1
;//4、ADCON2 设置
                MOVIA       b'00000000'
                MOVAR       ADCON2
                NOP          ;//延时一段时间再采集
                ...
                NOP
;//5、开启转换
                BSET        ADCON0,6
                JBTS0       ADCON0,6
                GOTO        $-1
                MOVR        ADH,A
                MOVAR       ADC_DATAH
                MOVR        ADL,A
                MOVAR       ADC_DATAH
Main:                                ;//程序主循环
    
```

.....

GOTO Main

; //+++++

Interrupt: ; //中断子程序

PUSH ; //压栈,保存 A,STATUS

; //中断处理程序

NOP

Interrupt\_End:

POP ; //出栈,恢复 A,STATUS

RETIE

END

# 14 看门狗 (WDT)

## 14.1 概述

看门狗定时器的时钟为内部独立 RC 时钟。

配置字 WDTCN 设置看门狗定时器的三种工作状态：

- (1) 始终使能：在 STOP 模式下仍然工作，溢出可唤醒 STOP
- (2) STOP 下关闭
- (3) 始终关闭

配置字 TWDTCN 设置看门狗的四种溢出时间： 4.5ms、18ms、72ms 或 288ms。

注：看门狗正常溢出后，程序复位到 0000H，是在休眠模式下看门狗溢出程序是继续往下运行。

## 14.2 OPTION 配置寄存器

0BCH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION	GIE	-	TO	PD	-	-	-	-
读/写	R/W	-	R	R	-	-	-	-
复位后	0	-	1	1	0	0	0	0

- Bit 5      **TO:**  超时位  
           1 = 上电复位或清除WDT  
           0 = WDT发生溢出
- Bit 4      **PD:**  掉电位  
           1 = 上电复位或清除WDT  
           0 = 进入休眠模式

## 14.3 WDTC 看门狗控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTC	WDTC7	WDTC6	WDTC5	WDTC4	WDTC3	WDTC2	WDTC1	WDTC0
读/写	W(*)	W(*)	W(*)	W(*)	W(*)	W(*)	W(*)	W(*)
复位后	-	-	-	-	-	-	-	-

注： (1) WDTC 写入 0x5A 将清除 WDT 定时器，写入其他值无效。  
 (2) CLRWDT 指令也可清除 WDT 定时器。

# 15 芯片配置字 (OPTION BIT)

烧录选项	内容	说明
程序区写入保护	允许改写程序区	
	不允许改写程序区	
振荡器模式	HIRC+LIRC	双系统时钟
	HXT+LIRC	
	HIRC+LXT	
CPU 运行速度选择	2T (LVR $\geq$ 3.8V)	高频模式下 CPU 速度选择; 低频模式下固定为 2T
	4T (LVR $\geq$ 2.4V)	
	8T (LVR $\geq$ 2.2V)	
	16T (LVR $\geq$ 1.8V)	
	32T (LVR $\geq$ 1.4V)	
	64T (LVR $\geq$ 1.4V)	
	128T (LVR $\geq$ 1.4V)	
	256T (LVR $\geq$ 1.4V)	
复位电压选择	LVR=1.4V	系统高速运行时, 请选择相应较高的 LVR 电压, 以保证系统的可靠性
	LVR=1.5V	
	LVR=1.6V	
	LVR=1.7V	
	LVR=1.8V	
	LVR=1.9V	
	LVR=2.0V	
	LVR=2.1V	
	LVR=2.2V	
	LVR=2.3V	
	LVR=2.4V	
	LVR=2.5V	
	LVR=3.5V	
	LVR=3.6V	
LVR=3.7V		
LVR=3.8V		
外部复位端口	作为外部复位端口	
	作为 IO 口	
启动模式选择	高速启动	
	低速启动	
输出端口读取	从端口读取	
	从输出寄存器读取	
芯片代码加密	不使能	
	使能	
低频晶振供	芯片 VDD	

电	BIAS 供电	
EEPROM 写入保护	不允许 EE 写入	
	允许 EE 写入	
WDT 使能选择	屏蔽 WDT 功能	
	使能, 绿色或休眠模式下关闭	
	始终开启 WDT 功能	
WDT 溢出时间	WDT 溢出时间=4.5mS	VDD=5V 典型值
	WDT 溢出时间=18mS	
	WDT 溢出时间=72mS	
	WDT 溢出时间=288mS	
仿真电压选择	VDD 4.2V	
	VDD 3.0V	
	外供电源	
EEPROM 选择	下载时不擦除 EEPROM	
	下载时擦除 EEPROM	

# 16 电性参数

## 16.1 极限参数

储存温度.....	-50℃~125℃
工作温度.....	-40℃~85℃
电源供应电压.....	0V~5.5V
端口输入电压.....	VSS-0.3V~VDD+0.3V

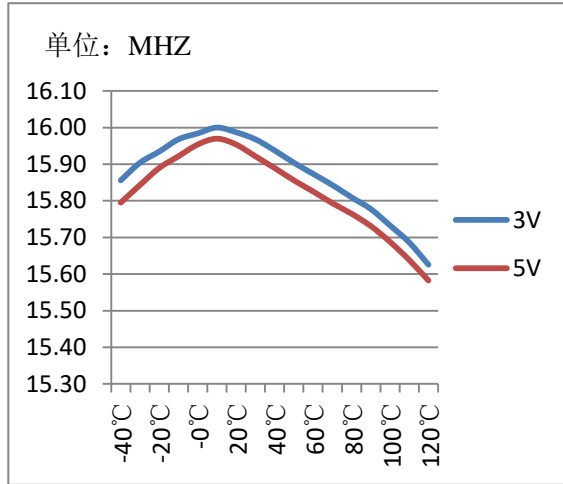
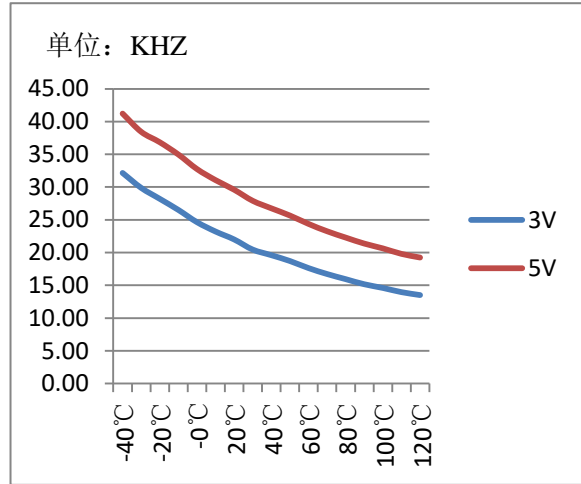
## 16.2 直流特性

符号	参数	测试条件		最小值	典型值	最大值	单位
		VDD	条件 (常温 25℃)				
VDD	工作电压	—	Fosc = 16MHz, 16T	1.8		5.5	V
IDD1	动态电流 1	3V	高频工作 (16M/4T), 低频工作(32K), 普通模式		1.7		mA
		5V		2.3			
IDD2	动态电流 2	3V	高频工作 (16M/8T), 低频工作(32K), 普通模式		1.3		mA
		5V		1.8			
IDD3	动态电流 3	3V	高频工作 (16M/16T), 低频工作(32K), 普通模式		0.9		mA
		5V		1.2			
IDD4	动态电流 4	3V	高频工作(16M/16T), 低频工作(32K), 绿色模式		140		uA
		5V		230			
IDD6	ADC 模块电流	3V	高频停止 (16M/16T), 低频工作(32K), 绿色模式		88.00		uA
		5V		215.0			
IDD7	CDC 模块电流	3V	高频停止 (16M/16T), 低频工作(32K), 绿色模式		220		uA
		5V		320			
ISP1	静态电流 1	3V	高频停止(16M/16T), 低频停止(32K), 休眠模式		2		uA
		5V		6			
ISP2	静态电流 2	3V	高频停止 (16M/16T), 低频工作 (32K), 绿色模式	4.5ms	3		uA
				18ms	2		
				72ms	2		
				288ms	2		
		5V		4.5ms	9		
				18ms	7		
				72ms	6		
			288ms	6			
VIL1	输入低电平	3V	施密特	0		0.3VDD	V



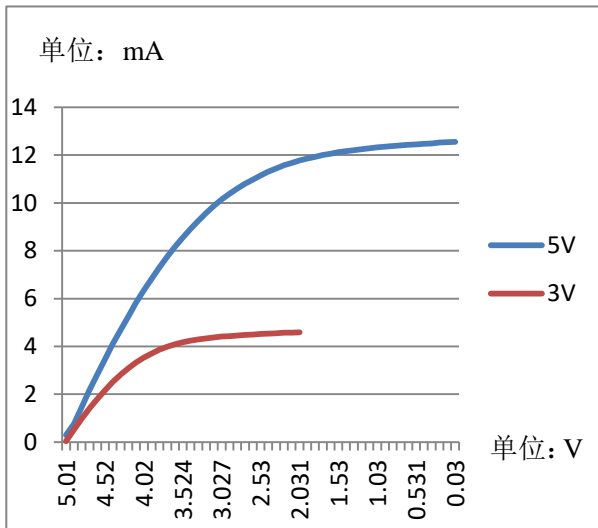
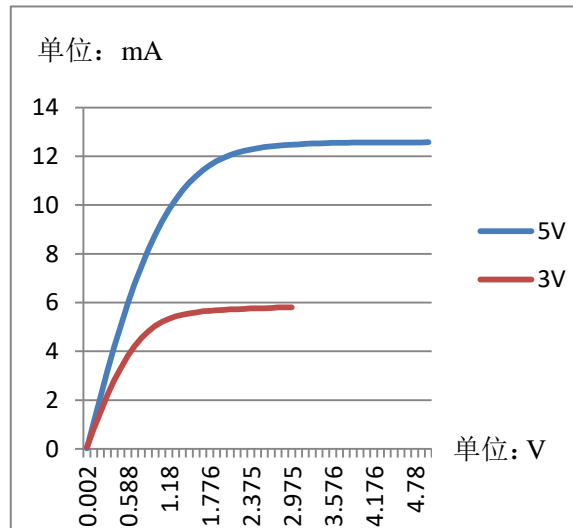
V <sub>IH1</sub>	输入高电平	3V	低翻转	0.7VDD		VDD	
V <sub>IL2</sub>	输入低电平	5V		0		0.3VDD	
V <sub>IH2</sub>	输入高电平	5V		0.6VDD		VDD	
V <sub>IL3</sub>	输入低电平	3V		0		0.2VDD	
V <sub>IH3</sub>	输入高电平	3V		0.3VDD		VDD	
V <sub>IL4</sub>	输入低电平	5V		0		0.1VDD	
V <sub>IH4</sub>	输入高电平	5V		0.2VDD		VDD	
I <sub>PH</sub>	上拉电阻	5V	输入到 GND		230		uA
		3V	输入到 GND		80		
I <sub>PL</sub>	下拉电阻	5V	输入到 VDD		230		
		3V	输入到 VDD		80		
I <sub>OL1</sub>	输出灌电流	5V	输出口, V <sub>out</sub> =V <sub>SS</sub> +0.6V		6		mA
		3V		4			
I <sub>OH1</sub>	输出拉电流	5V	输出口, V <sub>out</sub> =V <sub>DD</sub> -0.6V	—	4		
		3V		3			
I <sub>OL2</sub>	输出灌电流	5V	输出口, V <sub>out</sub> =V <sub>SS</sub> +0.6V		55		
		3V		35			
I <sub>OH2</sub>	输出拉电流	5V	输出口, V <sub>out</sub> =V <sub>DD</sub> -0.6V	—	25		
		3V		15			

## 16.3 温度对系统时钟的影响

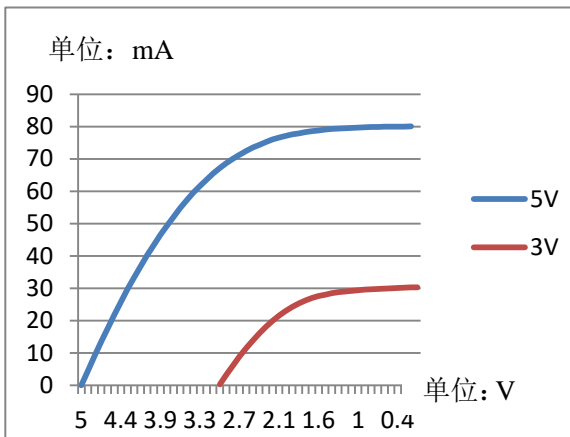
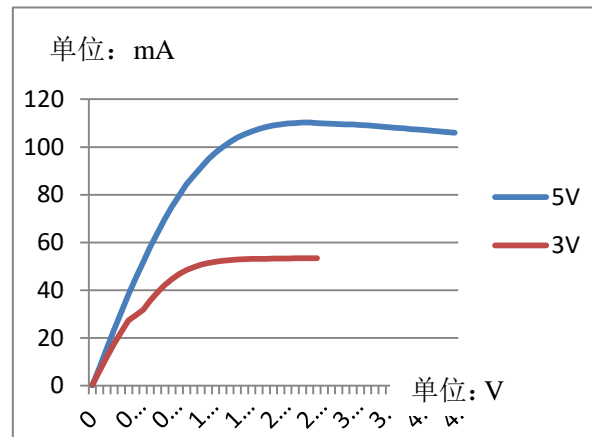
**高频(16M)**

**低频(32K)**


注: 具体值不做设计保证

## 16.4 IO 口拉灌电流特性曲线

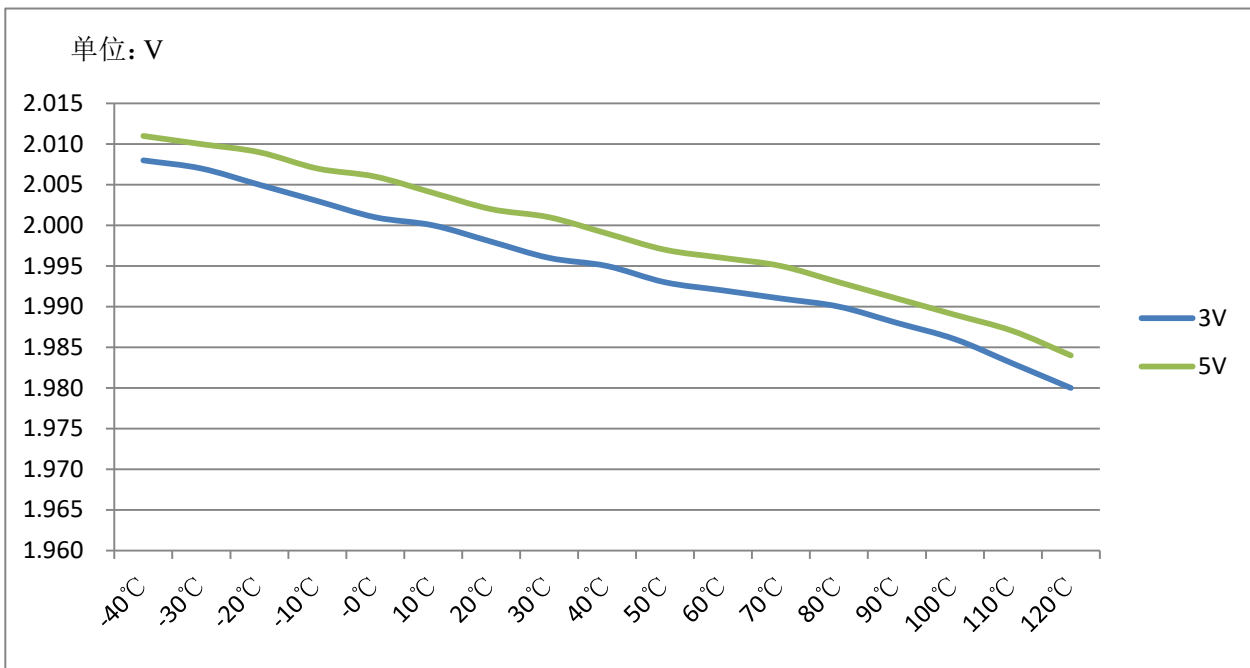
**IOH1**

**IOL1**


注: 具体值不做设计保证

**IOH2**

**IOL2**


注: 具体值不做设计保证

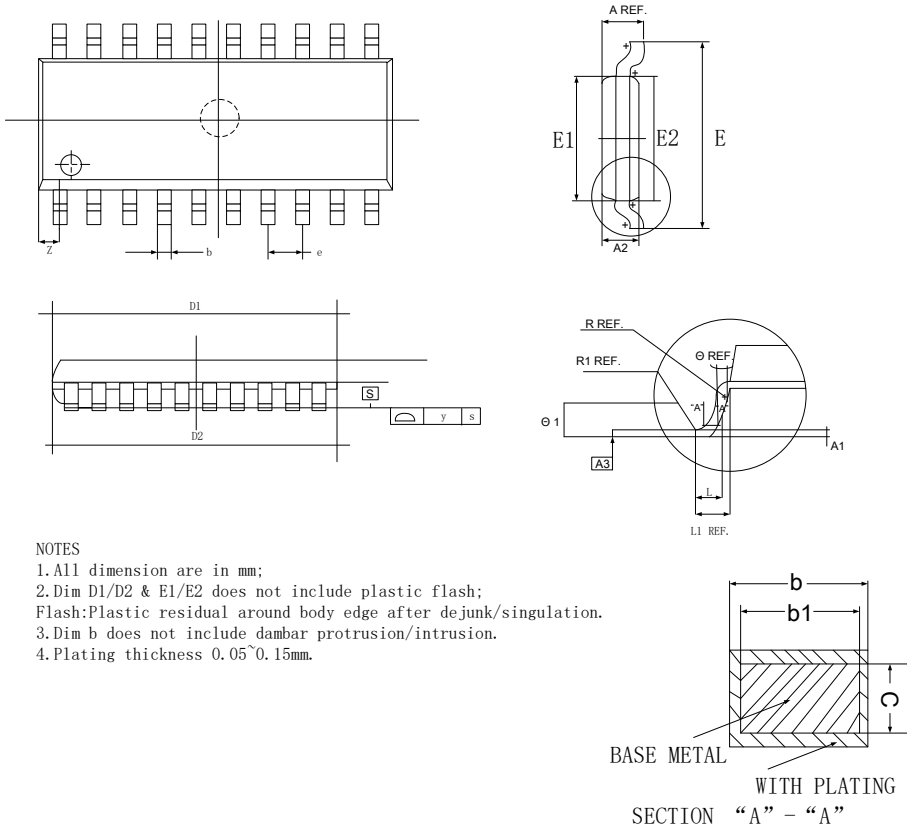
## 16.5 ADC 参考电压随温度变化曲线



注: 具体值不做设计保证

# 17 封装信息

## 17.1 SOP20

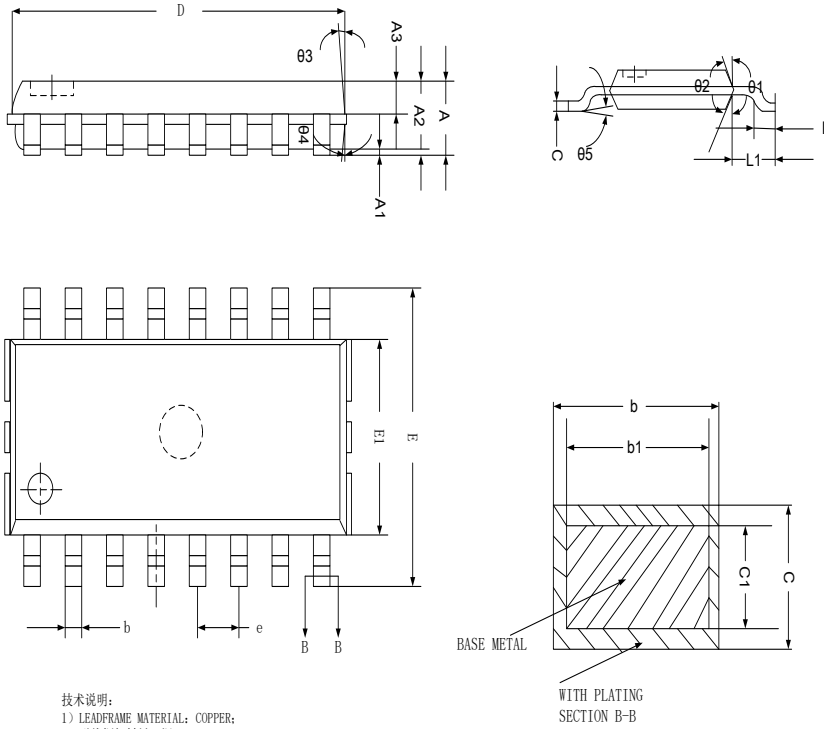

**NOTES**

1. All dimension are in mm;
2. Dim D1/D2 & E1/E2 does not include plastic flash;  
Flash:Plastic residual around body edge after dejunk/singulation.
3. Dim b does not include dambar protrusion/intrusion.
4. Plating thickness 0.05~0.15mm.

SYMBDL	MIN	NOM	MAX
A	2.465	2.515	2.565
A1	0.100	0.150	0.200
A2	2.100	2.300	2.500
A3	----	0.274	----
b	0.356	0.406	0.456
b1	0.366	0.426	0.486
c	----	0.254	----
D1	12.500	12.700	12.900
D2	12.550	12.750	12.950
E	10.206	10.306	10.406
E1	7.400	7.450	7.500
E2	7.450	7.500	7.550
e	----	1.270	----
L	0.800	0.864	0.900
L1	1.303	1.403	1.503
R	----	0.200	----
R1	----	0.300	----
$\theta$	0	----	----
$\theta 1$	0	----	10°
y	----	----	0.1
Z	----	0.660	----

## 17.2 SOP16

### SOP16 PACKAGE OUTLINE DIMENSIONS


**技术说明:**

- 1) LEADFRAME MATERIAL: COPPER;  
引线框架材料: 铜;
- 2) LEADFRAME THICKNESS: 0.203mm;  
引线框架厚度: 0.203mm;
- 3) BOTH PACKAGE LENGTH AND WIDTH DO NOT INCLUDE MOLD FLASH;  
塑封体长度及宽度尺寸不包括塑封溢胶;
- 4) REFERENCE: JEDEC MS-013, MS-012.  
参考标准: JEDEC MS-013, MS-012.

SYMBDL			
	MIN	NOM	MAX
A	--	--	1.70
A1	0.10	0.15	0.21
A2	1.40	1.45	1.50
A3	0.60	0.65	0.70
b	0.39	--	0.46
b1	0.38	0.41	0.44
c	0.20	--	0.24
c1	0.19	0.20	0.21
D	9.50	9.90	10.05
E	5.80	6.00	6.20
E1	3.75	3.85	4.00
e	1.27 (BSC)		
L	0.50	0.60	0.70
L1	1.10 (REF)		
θ1	6°	~	11°
θ2	6°	~	11°
θ3	5°	~	10°
θ4	6°	~	10°
θ5	0°	~	6°

# 18 指令集简述

## 18.1 概述

QT8Sxxxx系列指令集是一种精简指令集（RISC），指令宽度为16位，由操作码和0~2个操作数组成。指令按照功能可分为5类，即字节操作指令、位操作指令、立即数指令、分支指令、特殊控制指令。

一个指令周期由1个系统时钟周期组成，除非条件测试结果为真或指令执行改变了程序计数器的值，否则执行所有的指令都只需要一个指令周期。对于上述两种特征情况，指令执行需要两个指令周期。

任何一条指定文件寄存器作为指令一部分的指令都进行读-修改-写操作。读寄存器、修改数据并根据指令或目标标识符“d”存储结果。即使是写寄存器的指令也将先对改寄存器进行读操作。

## 18.2 符号说明

符号	范围	说明	符号	范围	说明
R/r	0-0x1ff	寄存器地址	C	-	进位标志
A	-	ACC 寄存器	DC	-	半进位标志
B/b	0-7	位地址	Z	-	零标志
I/i	0-0xff	立即数	d	0-1	目的操作数定义
K/k	0-0x1fff	标号	GIE	-	总中断使能位
TOS	-	栈顶	stkp	-	堆栈指针
PC	-	PC 指针			

## 18.3 QT8Sxxxx 指令集表

指令集表中, d=1, 目的操作数为 R; d=0, 目的操作数为 A

指令类型	助记符	指令说明	周期数	影响标志位	备注
寄存器操作指令	ADDAR R,d	$R+A \rightarrow d$	1	Z,DC,C	
	ADCAR R,d	$R+A+C \rightarrow d$	1	Z,DC,C	
	SUBAR R,d	$A-R \rightarrow d$	1	Z,DC,C	
	SBCAR R,d	$A-R-\overline{C} \rightarrow d$	1	Z,DC,C	
	SUBRA R,d	$R-A \rightarrow d$	1	Z,DC,C	
	SBCRA R,d	$R-A-\overline{C} \rightarrow d$	1	Z,DC,C	
	ANDAR R,d	$R\&A \rightarrow d$	1	Z	
	ORAR R,d	$R A \rightarrow d$	1	Z	
	XORAR R,d	$R^{\wedge}A \rightarrow d$	1	Z	
	COMR R,d	$\overline{R} \rightarrow d$	1	Z	
	MOVR R,d	$R \rightarrow d$	1	Z	
	MOVAR R	$A \rightarrow R$	1	-	
	CLRR R	$0 \rightarrow R$	1	Z	
	SWAPR R,d	R 半字节交换 $\rightarrow d$	1	-	
	RLR R,d	$R[7] \rightarrow C, \{R[6:0],C\} \rightarrow d$	1	C	
	RLRNC R,d	$\{R[6:0],0\} \rightarrow d$	1	-	
	RRR R,d	$R[0] \rightarrow C, \{C,R[7:1]\} \rightarrow d$	1	C	
	RRRNC R,d	$\{0,R[7:1]\} \rightarrow d$	1	-	
	DECR R,d	$R-1 \rightarrow d$	1	Z	
	DJZR R,d	$R-1 \rightarrow d, \text{SKIP if } 0$	1(2)	-	
	INCR R,d	$R+1 \rightarrow d$	1	Z	
	JZR R,d	$R+1 \rightarrow d, \text{SKIP if } 0$	1(2)	-	
	JNZR R,d	$R+1 \rightarrow d, \text{SKIP if } !0$	1(2)	-	(*1)
	DJNZR R,d	$R-1 \rightarrow d, \text{SKIP if } !0$	1(2)	-	(*1)
	JCMPPAR R	SKIP if $A=R$	1(2)	Z,C	(*1)
	JNCMPAR R	SKIP if $A \neq R$	1(2)	Z,C	(*1)
JGAR R	SKIP if $A \geq R$	1(2)	Z,C	(*1)	
JLAR R	SKIP if $A < R$	1(2)	Z,C	(*1)	
XCHAR R	$A \leftrightarrow R$	1	-	(*1)	
位操作指令	JBTS0 R,b	SKIP if $R[b]=0$	1(2)	-	
	JBTS1 R,b	SKIP if $R[b]=1$	1(2)	-	
	BCLR R,b	$0 \rightarrow R[b]$	1	-	
	BSET R,b	$1 \rightarrow R[b]$	1	-	

指令类型	助记符	指令说明	周期数	影响标志位	备注
立即数操作指令	ADDIA I	$I+A \rightarrow A$	1	Z,DC,C	
	ADCIA I	$I+A+C \rightarrow A$	1	Z,DC,C	(*2)
	SUBIA I	$I-A \rightarrow A$	1	Z,DC,C	
	SBCIA I	$I-A-C \rightarrow A$	1	Z,DC,C	
	SUBAI I	$A-I \rightarrow A$	1	Z,DC,C	
	SBCAI I	$A-I-C \rightarrow A$	1	Z,DC,C	
	ANDIA I	$A \& I \rightarrow A$	1	Z	
	ORIA I	$A   I \rightarrow A$	1	Z	
	XORIA I	$A \wedge I \rightarrow A$	1	Z	
	MOVIA I	$I \rightarrow A$	1	-	
	RETIA I	Stack $\rightarrow$ PC, $I \rightarrow A$	2	-	
	JCMPAI I	SKIP if A=I	1(2)	Z,C	(*1)
	JNCPAI I	SKIP if A $\neq$ I	1(2)	Z,C	(*1)
	特殊操作指令	RLA	$A[7] \rightarrow C, \{A[6:0], C\} \rightarrow A$	1	C
RLANC		$\{A[6:0], 0\} \rightarrow A$	1	-	
RRA		$A[0] \rightarrow C, \{C, A[7:1]\} \rightarrow A$	1	C	
RRANC		$\{0, A[7:1]\} \rightarrow A$	1	-	
DECA		$A-1 \rightarrow A$	1	Z	
DJZA		$A-1 \rightarrow A, \text{SKIP if } 0$	1(2)	-	
INCA		$A+1 \rightarrow A$	1	-	
JZA		$A+1 \rightarrow A, \text{SKIP if } 0$	1(2)	-	
RETIE		Stack $\rightarrow$ PC, $1 \rightarrow GIE$	2	-	
RETURN		Stack $\rightarrow$ PC	2	-	
NOP		None Operation	1	-	
RDT		ROM[ $\{fsr1, fsr0\}$ ] $\rightarrow$ {HBUF, A}	2	-	
DAA		加法后十进制调整	1	DC, C	
DSA		减法后十进制调整	1	DC, C	
PUSH		A, STATUS 压栈	1	-	(*1)(*2)
POP		A, STATUS 出栈	1	Z, DC, C	(*1)(*2)
CLRWDT		清除 WDT 寄存器	1	PD, TO	(*1)
分支指令	CALL I	$I \rightarrow PC, PC \rightarrow \text{Stack}$	2	-	
	GOTO I	$I \rightarrow PC$	2	-	
	ACALL I	$I \rightarrow PC, PC \rightarrow \text{Stack}$	2	-	
	AGOTO I	$I \rightarrow PC$	2	-	