1380 串行时钟芯片

1．特点

（1） 工作电压：2.0V～5.5V

（2） 最大输入串行时钟：2.0V 时　　500KHz

5.0V 时　　2MHz

（3） 工作电流:2.0V 时至少 300nA

5.0V 时至少 1μA

（4） 与 TTL 兼容

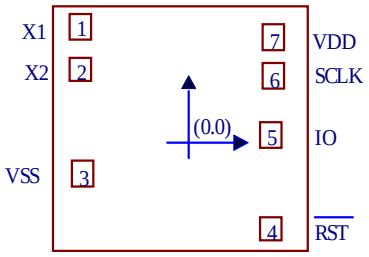
（5） 串行 I/O 口传送

（6） 两种数据传送方式：单字节传送

多字节传送（字符组方式）

（7） 所有寄存器都以 BCD 码格式存储

表面贴装(尺寸见下表)



Chip Size:2480 x 2390(um)

Unit:um

Pad NO.	X	Y	Pad NO.	X	Y
1	-1060.5	1000	5	1050.6	54.15
2	-1060.5	683.13	6	1050.8	472.16
3	-1060.5	-236.14	7	1050.4	770.21
4	1050.62	-710			

2．用途

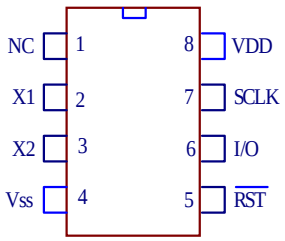
（1）微型计算机串行时钟　　（2）时钟和日历

3．概述

H1380 是一个带秒、分、时、日、日期、月、年的串行时钟保持芯片，每个月多少天以及闰年能自动调节，HT1380 低功耗工作方式，HT1380 用若干寄存器存储对应信息，一个 32.768KHz 的晶振校准时钟，为了使用最小引脚，HT1380 使用一个 I/O 口与微信息处理机相连，仅使用三根引线（1）/RST 复位；（2）SCLK 串行时钟；（3）I/O 口数据就可以传送 1 字节或 8 字节的字符组。

4．引脚定义

符号	管脚号	描　　述
NC	1	空脚
X1	2	振荡器输入
X2	3	振荡器输出
V_{SS}	4	地
/RST	5	复位引脚
I/O	6	数据输入/输出引脚
SCLK	7	串行时钟
V_{DD}	8	正电源



5．绝对最大范围

工作电压 ： -0.3V – 5.5V　　储藏温度 ： -50℃ – 125℃

输入电压 ： V_{SS}-0.3V – V_{DD}+0.3V　　工作温度：0℃ – 70℃

6．直流特性

符号	参数	测试条件		最小值	典型值	最大值	单位
		V_{DD}	条件				
V_{DD}	工作电压	–	–	2	–	5.5	V
I_{STB}	静态电流	2V	–	–	–	100	nA
		5V		–	–	100	nA
I_{DD}	工作电流	2V	空载	–	–	0.3	uA
		5V		–	–	1.0	uA
I_{OH}	输出电流	2V	V_{OH}=1.8V	-0.2	-0.4	–	mA
		5V	V_{OH}=4.5V	-0.5	-1.0	–	mA
I_{OL}	灌电流	2V	V_{OL}=0.2V	0.7	1.5	–	mA
		5V	V_{OL}=0.5V	2.0	4.0	–	mA
V_{IH}	“高”电平	5V	–	2	–	–	V
V_{IL}	“低”电平	5V	–	–	–	0.8	V
F_{OSC}	系统频率	5V	32.768KHz 晶体	–	32.768	–	KHz
F_{SCLK}	串行时钟	2V	–	–	–	0.5	KHz
		5V		–	–	2	MHz

I_{STB}特指 SCLK，I/O，/RST 开漏，且时钟停止位必须设置为逻辑 1（振荡停止）

7．交流特性

符号	参数	VDD	最小值	最大值	单位
t_{DC}	数据到时钟建立	2V	200	-	ns
		5V	50	-	
t_{CDH}	时钟到数据保持	2V	280	-	ns
		5V	70	-	
t_{CDD}	时钟到数据延时	2V	-	800	ns
		5V	-	200	
t_{CL}	时钟低时间	2V	1000	-	ns
		5V	250	-	
t_{CH}	时钟高时间	2V	1000	-	ns
		5V	250	-	
f_{CLK}	时钟频率	2V	1000	0.5	MHz
		5V	250	2.0	
t_R t_F	时钟上升及下降时间	2V	-	2000	ns
		5V	D.C	500	
t_{CC}	复位到时钟建立	2V	-	-	us
		5V	-	-	
t_{CCH}	时钟到复位保持	2V	240	-	ns
		5V	60	-	
t_{CDZ}	复位静止时间	2V	4	-	us
		5V	1	-	
t_{CDZ}	复位到 I/O 由高变低时间	2V	-	280	ns
		5V	-	70	

8．特征概述

HT1380 主要**包括**以下几点：

（1）一个数据移位寄存器组存储时钟/日历数据，命令控制逻辑，振荡器电路和读时钟。两种数据传送方式：单字节和多字节。

（2）在开始发送数据之前，先把/RST 置高，发送一个带地址和命令信息的 8 位命令字，紧跟命令之时钟/日历数据传送至相应的寄存器中或从相应寄存器传送出来（读）。

（3）/RST 引脚在数据传送完毕应保持低电平。

（4）所有数据的输入是在 SCLK 的上升沿有效，输出在 SCLK 的下降沿有效。

（5）单字节传送需要 16 个 SCLK 时钟脉冲，多字节传送需要 72 个 SCLK 时钟脉冲，输入、输出数据都是从 0 位开始。

HT1380 还包括两个附加位：时钟停止位（CH）和写保护位（WP），这些位控制振荡器的工作和数据能否写入寄存器中，这两位应首先规定（为了读、写寄存器组）。

9．命令字节

（1）每个数据的传送都是在命令字节中规定寄存器的工作方式，是读、写还是测试，是单字节还是多字节传送。

（2）命令字节的格式

7	6	5	4	3	2	1	0
1	0	0	A3	A2	A1	A0	R/W

A_0～A_2:寄存器地址

A_3: A_3=1 时测试模式, A_3=0 时常规模式

R/W:为 1 时读,为 0 时写

当命令字节为 1001xxx1 时,HT1380 设置在测试模式,这种仅为半导体公司测试时使用,如果使用普通的,不可预测条件的变化。

10．时钟停止位

当把秒寄存器的第 7 位（时钟停止位）设置为 1 时，时钟振荡器停止，HT1380 进入低功耗方式，当该位写入 0 时，起动时钟开始。

下表是寄存器的地址和数据格式

寄存器地址 A_0～A_2	特征	命令地址	读写控制	数据(BCD)	寄存器定义 7	6	5	4	3	2	1	0
0	秒	80	写	00～59	CH	10 秒			秒			
		81	读									
1	分	82	写	00～59	0	10 分			分			
		83	读									
2	12 小时	84	写	01～12	12/24	0	AP	HR	时			
	24 小时	85	读	00～23		0	10	HR				
3	日期	86	写	01～31	0	0	10 日期		日期			
		87	读									
4	月	88	写	01～12	0	0	0	10 月	月			
		89	读									
5	日	8A	写	01～07	0	0	0	0	星期			
		8B	读									
6	年	8C	写	00～99	10 年				年			
		8D	读									
7	写保护	8E	写	00～80	WP	通常 0						
		8F	读									

CH: 时钟停止位

CH=0 振荡器工作允许

CH=1 振荡器停止

寄存器 2 的第 7 位：12/24 小时标志

bit7=1,12 小时模式

bib7=0,24 小时模式

WP: 写保护位

WP=0 寄存器数据能够写入

WP=1 寄存器数据不能写入

寄存器 2 的第 5 位:AM/PM 定义

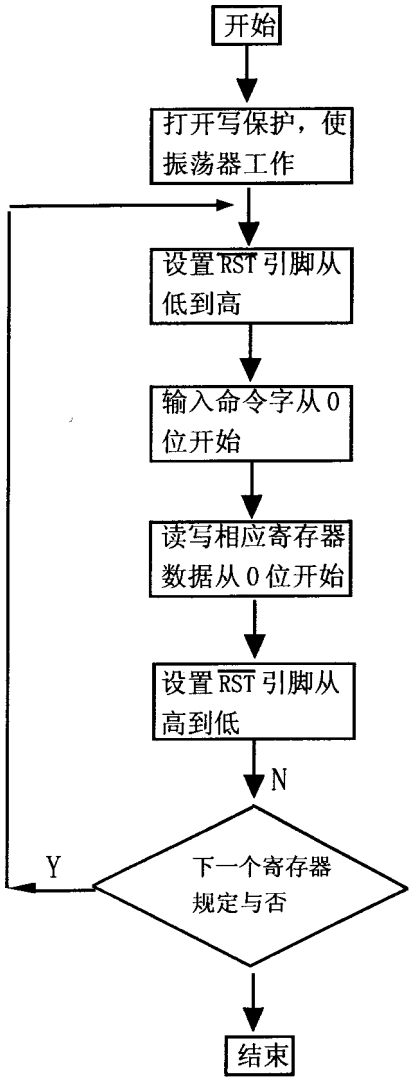
AP=1 下午模式

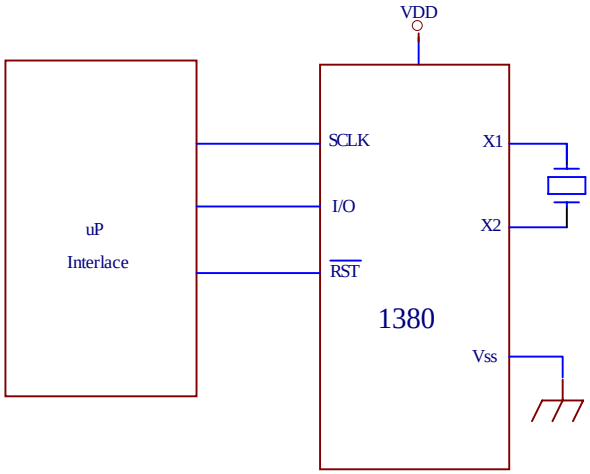
AP=0 上午模式

11．写保护寄存器

当写保护寄存器的高位为 0 时，允许数据写入寄存器，写保护寄存器可以通过命令字节 8E、8F 来

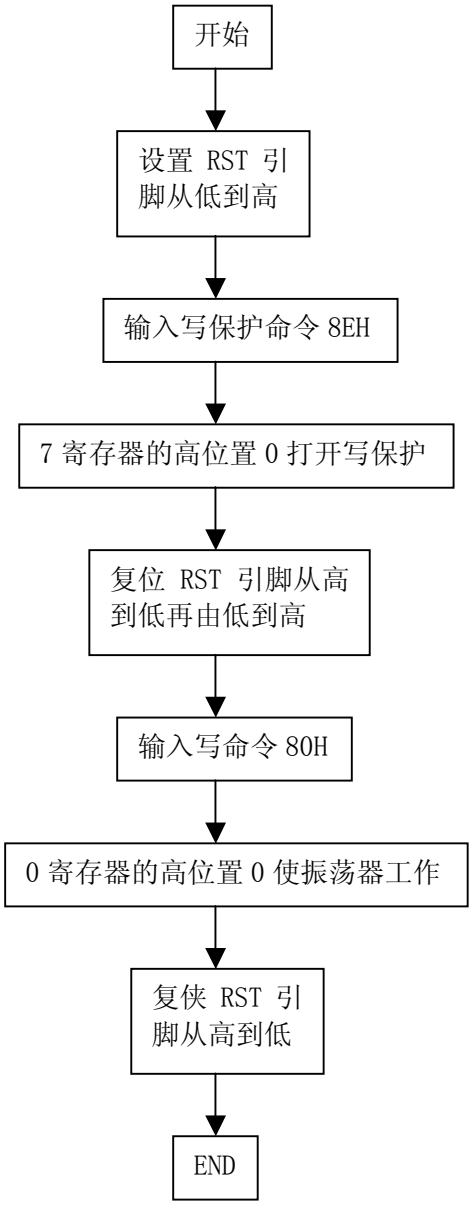
（4）单字节传送


开始
打开写保护，使振荡器工作
设置RST引脚从低到高
输入命令字从0位开始
读写相应寄存器数据从0位开始
设置RST引脚从高到低
N
Y
下一个寄存器规定与否
结束

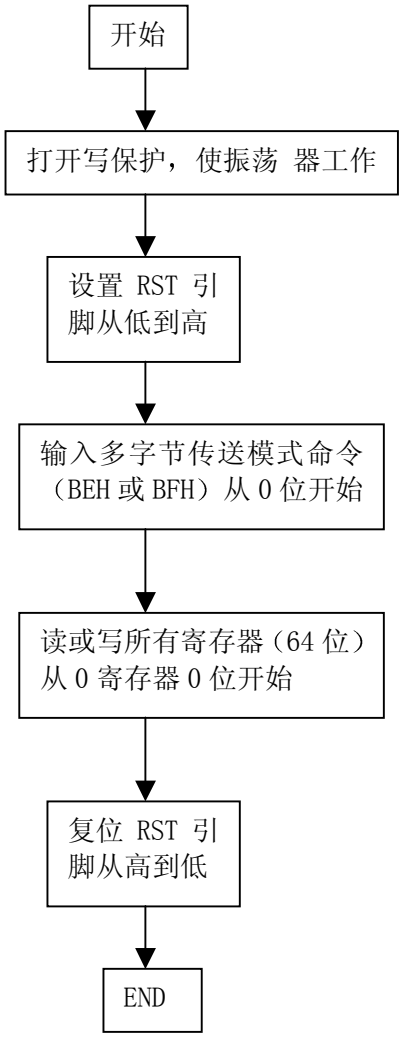

VDD
uP
Interlace
SCLK
I/O
RST
X1
X2
1380
Vss

流程图

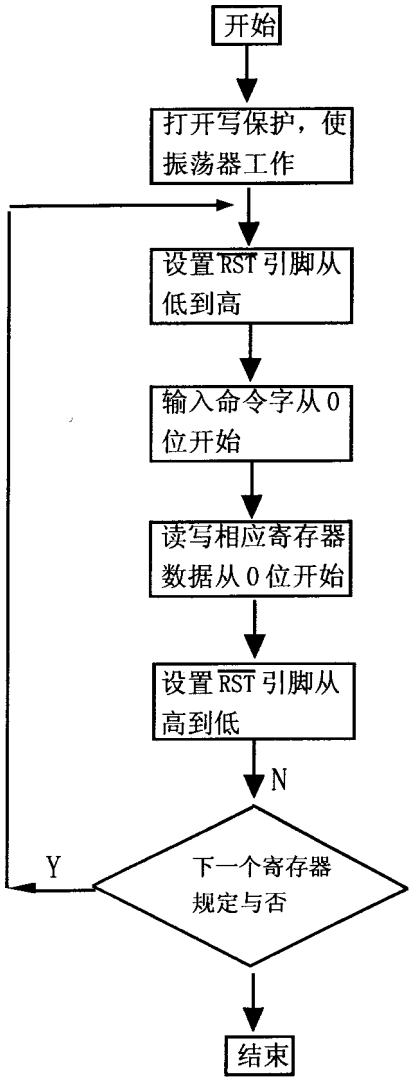
（1）振荡器工作（CH=0）
写保护打开（WP=0）

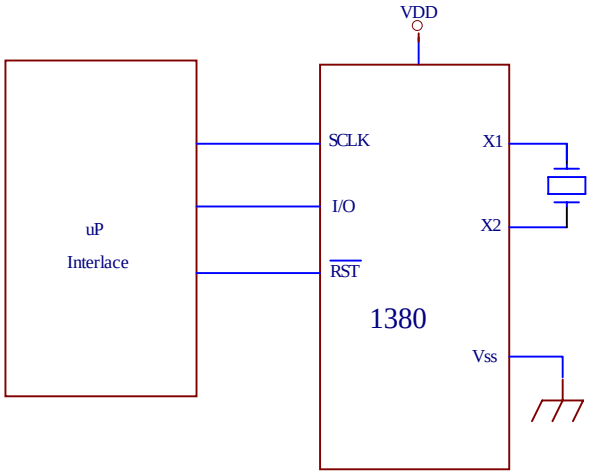


（2）多字节传送



（4）单字节传送


开始
打开写保护，使振荡器工作
设置RST引脚从低到高
输入命令字从0位开始
读写相应寄存器数据从0位开始
设置RST引脚从高到低
N
Y
下一个寄存器规定与否
结束


VDD
uP
Interlace
SCLK
I/O
RST
X1
X2
1380
Vss

1380 串行时钟芯片的原理与应用

：　　，　　。

（　　）　　。

一、概述

1380 是一款带秒　　分　　、　、时　日　星期　月　年的串行时钟保持芯片　每个月多少天以及闰年能自动调节，1380 具有低功耗工作方式并用若干寄存器存储对应信息　　，　一个 32.768KHz 的晶振校准时钟　为了使用最小引脚，1380 使用一个　　I/O 口与微信息处理机相连　仅使用三根引线（1）RST 复位（2）SCLK 串行时钟（3）I/O 口数据就可以传送 1 字节或 8 字节的字符组。因而，1380 是一种性价比极高的时钟芯片　　，它广泛应用于电话　传真　便携式仪器以及电池供电的仪器仪表等产品领域。下面将主要的性能指标作一综合：

（1）　工作电压：2.0V～5.5V

（2）　最大输入串行时钟:　2.0V 时　　500KHz

5.0V 时　　2MHz

（3）　工作电流:　2.0V 时至少 300nA

5.0V 时至少 1μA

（4）　与 TTL 兼容

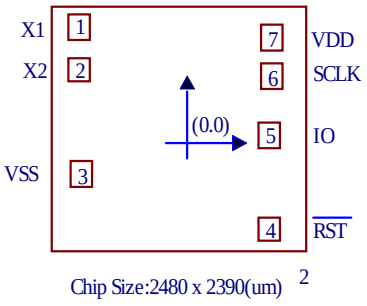
（5）　串行 I/O 口传送

（6）　两种数据传送方式 ：单字节传送

多字节传送（字符组方式）

（7）　所有寄存器都以 BCD 码格式存储

表面贴装(尺寸见下)



2

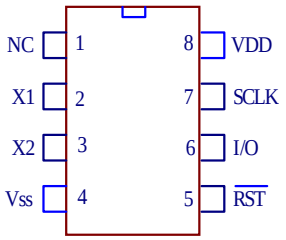
单位：微米

Pad　NO.	X	Y	Pad　NO.	X	Y
1	-1060.5	1000	5	1050.6	54.15
2	-1060.5	683.13	6	1050.8	472.16
3	-1060.5	-236.14	7	1050.4	770.21
4	1050.62	-710			

二、1380 的基本组成和工作原理

1.　1380 的管脚排列及描述如下图及表所示

符号	管脚号	描　　述
NC	1	空脚
X1	2	振荡器输入
X2	3	振荡器输出
V_{SS}	4	地
/RST	5	复位引脚
I/O	6	数据输入/输出引脚
SCLK	7	串行时钟
V_{DD}	8	正电源



2. 1380 内部寄存器

寄存器地址 A_0～A_2	特征	命令地址	读写控制	数据(BCD)	寄存器定义							
					7	6	5	4	3	2	1	0
0	秒	80	写	00～59	CH	10 秒			秒			
		81	读									
1	分	82	写	00～59	0	10 分			分			
		83	读									
2	12 小时	84	写	01～12	12/24	0	AP	HR	时			
	24 小时	85	读	00～23		0	10	HR				
3	日期	86	写	01～31	0	0	10 日期		日期			
		87	读									
4	月	88	写	01～12	0	0	0	10 月	月			
		89	读									
5	日	8A	写	01～07	0	0	0	0	星期			
		8B	读									
6	年	8C	写	00～99	10 年				年			
		8D	读									
7	写保护	8E	写	00～80	WP	通常 0						
		8F	读									

CH: 时钟停止位　　寄存器 2 的第 7 位：12/24 小时标志

CH=0　振荡器工作允许　　bit7=1,12 小时模式

CH=1　振荡器停止　　bit7=0,24 小时模式

WP:写保护位　　寄存器 2 的第 5 位:AM/PM 定义

WP=0　寄存器数据能够写入　　AP=1　下午模式

WP=1　寄存器数据不能写入　　AP=0　上午模式

三、1380 与微控制器的接口软件及功能应用举例

下面首先给出基本的接口软件，然后举例说明各种功能的应用。

1. 写保护寄存器操作

当写保护寄存器的最高位为 0 时，允许数据写入寄存器，写保护寄存器可以通过命令字节 8E、8F 来规定禁止写入/读出。写保护位不能在多字节传送模式下写入。

```
Write_Enable:
        MOV     Command,#8Eh    ;命令字节为 8E
        MOV     ByteCnt,#1      ;单字节传送模式
        MOV     R0,#XmtDat      ；数据地址覆给 R0
        MOV     XmtDat,#00h     ；数据内容为 0（写入允许）
        ACALL   Send_Byte       ；调用写入数据子程序
        RET
```

当写保护寄存器的最高位为 1 时，禁止数据写入寄存器，

```
Write_Disable:
        MOV     Command,#8Eh    ;命令字节为 8E
        MOV     ByteCnt,#1      ;单字节传送模式
        MOV     R0,#XmtDat      ；数据地址覆给 R0
        MOV     XmtDat,#80h     ；数据内容为 80h（禁止写入）
        ACALL   Send_Byte       ；调用写入数据子程序
        RET
```

以上程序调用了基本数据发送(Send_Byte)模块及一些内存单元定义，其源程序清单在附录中给出。下面的程序亦使用了

这个模块。

2．时钟停止位操作

当把秒寄存器的第 7 位（时钟停止位）设置为 0 时，起动时钟开始。

```
Osc_Enable:
        MOV     Command,#80h        ；命令字节为 80
        MOV     ByteCnt,#1          ；单字节传送模式
        MOV     R0,#XmtDat          ；数据地址覆给 R0
        MOV     XmtDat,#00h         ；数据内容为 0（振荡器工作允许）
        ACALL   Send_Byte           ；调用写入数据子程序
        RET
```

当把秒寄存器的第 7 位（时钟停止位）设置为 1 时，时钟振荡器停止，HT1380 进入低功耗方式，

```
Osc_Disable:
        MOV     Command,#80h        ；命令字节为 80
        MOV     ByteCnt,#1          ；单字节传送模式
        MOV     R0,#XmtDat          ；数据地址覆给 R0
        MOV     XmtDat,#80h         ；数据内容为 80h（振荡器停止）
        ACALL   Send_Byte           ；调用写入数据子程序
        RET
```

3. 多字节传送方式

当命令字节为 BE 或 BF 时，1380 工作　　在多字传送模式　8 个时钟/日历寄存器从寄存器 0 地址开始连续读写从 0 位开始的数据。

例如：写入 00 年、6 月 21 日、星期三、13 时、59 分、59 秒，程序设置如下：

```
Write_Multiplebyte:
        MOV     Command,#0BEh       ；命令字节为 BEh
        MOV     ByteCnt,#8          ；多字节写入模式（此模块为 8 个）
        MOV     R0,#XmtDat          ；数据地址覆给 R0
        MOV     XmtDat,#59h         ；秒单元内容为 59h
        MOV     XmtDat+1,#59h       ；分单元内容为 59h
        MOV     XmtDat+2,#13h       ；时单元内容为 13h
        MOV     XmtDat+3,#21h       ；日期单元内容为 21h
        MOV     XmtDat+4,#06h       ；月单元内容为 06h
        MOV     XmtDat+5,#03h       ；星期单元内容为 03h
        MOV     XmtDat+6,#0         ；年单元内容为 00h
        MOV     XmtDat+7,#0         ；写保护单元内容为 00h
        ACALL   Send_Byte           ；调用写入数据子程序
          RET
```

读出寄存器 0-7 的内容，程序设置如下：

```
Read_Multiplebyte:
        MOV     Command,#0BFh       ；命令字节为 BFh
        MOV     ByteCnt,#8          ；多字节读出模式（此模块为 8 个）
        MOV     R1,#RcvDat          ；数据地址覆给 R1
        ACALL   Receive_Byte        ；调用读出数据子程序
        RET
```

以上程序调用了基本数据接收(Receive_Byte)模块及一些内存单元定义，其源程序清单在附录中给出。下面的程 序亦使用了这个模块。

4. 单字节传送方式

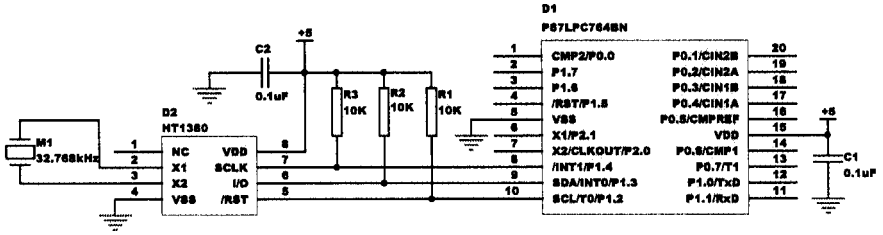
例如：写入 8 时（12 小时模式），程序设置如下：

```
Write_Singlebyte:
        MOV     Command,#84h        ；命令字节为 84h
        MOV     ByteCnt,#1          ；单字节传送模式
        MOV     R0,#XmtDat          ；数据地址覆给 R0
        MOV     XmtDat,#88h         ；数据内容为 88h
        ACALL   Send_Byte           ；调用写入数据子程序
        RET
```

上面所列出的程序模块“Write_Enable”、“Write_Disable”、“Osc_Enable”、“Osc_Disable”与单字节写入模块“Write_Singlebyte”的程序架构完全相同，仅只是几个入口参数不同，本文是为了强调功能使用的不同才将其分为不同模块，用户在使用中可灵活简略。

下面模块举例说明如何单字节读出“小时”单元的内容.

```
Read_Singlebyte:
        MOV     Command,#85h        ；命令字节为 85h
        MOV     ByteCnt,#1          ；单字节传送模式
        MOV     R1,#RcvDat          ；数据地址覆给 R1
        ACALL   Receive_Byte        ；调用读出数据子程序
        RET
```



1380 应用电　　路原理图（ P87LPC764 单片机选取内部振荡及内部复位电路）

附录：数据发送与接收模块源程序清单

```
; CPU 工作频率最大不超过 20MHz
;*********************************************************************************************
;                                P87LPC762/4 主控器发送接受数据程序
; 说明：本程序是利用 Philips 公司的 P87LPC764 单片机（任何具有 51 内核或其它合适的单片机都可在此
; 作为主控器）的普通 I/O 口(如 P1.2/P1.3/P1.4)实现总线的功能，对总线 上的器件（本程序采用 HT1380）
; 进行读写操作。命令字节在 Command， 传送字节数在 ByteCnt 中，所发送的数据在 XmtDat 中，所接收
; 的数据在 RcvDat 中。
;*********************************************************************************************
;P87LPC762/4 主控器总线发送接受数据程序头文件
;内存数据定义
BitCnt      DATA    30h     ；数据位计数器
ByteCnt     DATA    31h     ；数据字节计数器
Command     DATA    32h     ；命令字节地址
```

```
    RcvDat          DATA        40H         ; 接收数据缓冲区
    XmtDat          DATA        50H         ; 发送数据缓冲区
  ;端口位定义
    IO_DATA         Bit         P1.3        ; 数据传送总线
    SCLK            Bit         P1.4        ; 时钟控制总线
    RST             Bit         P1.2        ; 复位总线
  ;*********************************************************************************
  ;发送数据程序
  ;名称:Send_Byte
  ;描述:发送（ByteCnt）个字节给被控器1380
  ;命令字节地址在 Command 中
  ;所发送数据的字节数在 ByteCnt 中，发送的数据在 XmtDat 缓冲区中
  ;*********************************************************************************
Send_Byte:
        CLR     RST             ; 复位引脚为低电平，所有数据传送终止
        NOP
        CLR     SCLK            ; 清时钟总线
        NOP
        SETB    RST             ; 复位引脚为高电平，逻辑控制有效
        NOP
        MOV     A,Command       ; 准备发送命令字节
        MOV     BitCnt,#08h     ; 传送位数为 8
 S_Byte0:
        RRC     A               ; 将最低位传送给进位位 C
        MOV     IO_DATA,C       ; 位传送至数据总线
        NOP
        SETB    SCLK            ; 时钟上升沿，发送数据有效
        NOP
        CLR     SCLK            ; 清时钟总线
        DJNZ    BitCnt,S_Byte0  ; 位传送未完毕则继续
        NOP
S_Byte1:                        ; 准备发送数据
        MOV     A,@R0           ; 传送数据，过程与传送命令相同
        MOV     BitCnt,#08h
S_Byte2:
        RRC     A
        MOV     IO_DATA,C
        NOP
        SETB    SCLK
        NOP
        CLR     SCLK
        DJNZ    BitCnt,S_Byte2
        INC     R0              ; 发送数据的内存地址加 1
        DJNZ    ByteCnt,S_Byte1 ; 字节传送未完毕则继续
```

```
        NOP
        CLR     RST                 ；逻辑操作完毕，清 RST
        RET

;*********************************************************************
;接收数据程序;
;名称:Receive_Byte
;描述:从被控器1380 接    收（ByteCnt）个字节数据
;命令字节地址在 Command 中
;所接收数据的字节数在 ByteCnt 中，接收的数据在 RcvDat 缓冲区中
;*********************************************************************
Receive_Byte:
        CLR     RST                 ；复位引脚为低电平，所有数据传送终止
        NOP
        CLR     SCLK                ；清时钟总线
        NOP
        SETB    RST                 ；复位引脚为高电平，逻辑控制有效
        MOV     A,Command           ；准备发送命令字节
        MOV     BitCnt,#08h         ；传送位数为 8
R_Byte0:
        RRC     A                   ；将最低位传送给进位位 C
        MOV     IO_DATA,C           ；位传送至数据总线
        NOP
        SETB    SCLK                ；时钟上升沿，发送数据有效
        NOP
        CLR     SCLK                ；清时钟总线
        DJNZ    BitCnt,R_Byte0      ；位传送未完毕则继续
        NOP
R_Byte1:                            ；准备接收数据
        CLR     A                   ；清类加器
        CLR     C                   ；清进位位 C
        MOV     BitCnt,#08h         ；接收位数为 8
R_Byte2:
        NOP
        MOV     C,IO_DATA           ；数据总线上的数据传送给 C
        RRC     A                   ；从最低位接收数据
        SETB    SCLK                ；时钟总线置高
        NOP
        CLR     SCLK                ；时钟下降沿接收数据有效
        DJNZ    BitCnt,R_Byte2      ；位接收未完毕则继续
        MOV     @R1,A               ；接收到的完整数据字节放入接收内存缓冲区
        INC     R1                  ；接收数据的内存地址加 1
        DJNZ    ByteCnt,R_Byte1     ；字节接收未完毕则继续
        NOP
```

```
CLR     RST                    ；逻辑操作完毕，清 RST
RET
;
END
```

热点问题解答：

1. 初次使用1380 的用　户，经常会遇到振荡器“不起振”的问题；其实，绝大多数用户都是由于在调试过程中期望通过示波器观察振荡管脚，却未留意 X1（振荡器输入，PIN2）与 X2（振荡器输出，PIN3） 的区别，在此特别强调；示波器探极只有接到 X2 才能观察到振荡波形；如果接到 X1 脚，不仅观察不到振荡，同时也**使得振荡器“停振”**，另外，正确的程序设计也是能否起振的前提。

2. 已对1380 使用过一段　时间的用户提出，程序控制及硬件设计在以前都能通过，为什么有些振荡器能起振，而有些“不起振”？问题首先出在晶振本身；用户接着提出，所选取的 32.768KHz 的振荡器经频率计或在其它某些系统中使用时都正常，精度也能满足，怎么能说晶振有问题？ 这是因为1380 内部　振荡电路给外部振荡器提供的激励相对较弱，而市场上各厂家生产的晶振在工艺制造中又有优劣之分，有些晶振需要很强的激励才能起振；因此，用户认准一种品牌的晶振非常重要；对于在试验过程的用户来说，遇到这种问题有两个简单的解决办法：

- 选取以前已使用过并证明功能正常的1380 做　一个简易的测试板，以方便的方式将未起振的晶　振与之匹配，然后运行正确的程序，将能起振的此晶振重新移回您的系统板，这一过程类似“学习”。
- 在上电操作中用已加热的烙铁烫一下晶振的两个引脚，通过高温强迫起振，断电后再上电也能正常起振。

3. 读写不正确的用户，请您放心使用以上的程序模块就行了，前提是您的 CPU 工作频率最大不超过 20MHz。