用软件执行 I^2C 总线的控制功能的方法

作　者：　盛扬半导体（上海）有限公司软件部
日　期：　2001/8/6
适用单片机：HT48C50

简介：

I^2C 总线包括了两条串行总线（时钟线 SCL 和数据线 SDA），通过这两条总线能实现多个芯片之间的通信。在互相连接的芯片中，至少有一个芯片作为总线控制器，而其它芯片则作为从控制器。在本应用说明中，介绍了用 Holtek 的八位 RISC 结构的单片机作为单总线控制器的软件实现的方法。在本文的示例中，采用了一片 EEPROM（型号 HT24C01，1Kbit）作为从控制器参与测试。

使用说明：

为了使 I^2C 程序的使用更为简便，我们提供了一些宏指令和标志。

宏指令：

1．LOAD_ADDR_8　*SLAVE_ADDRESS*
　把从控器的地址装入内部寄存器中供以后使用。
2．I2C_TEST_DEVICE
　测试被控芯片是否工作。结果由位标志 *_SlaveActive* 显示。
3．RELEASE_BUS
　释放总线，进入高阻态。
4．I2C_WRITE_SUB *_BYTES_*, *_SourcePointer_*, *_SubAddress_*
　可任意把‘*_BYTES_*’字节的数据由源指针‘*_SourcePointer_*’指出的数据传送到从控制器由地址‘*_SubAddress_*’开始的页空间内。是否完成传送过程由位标志 *_TxmtSuccess* 显示。
5．I2C_READ_SUB *_BYTES_*, *_DestPointer_*, *_SubAddress_*
　可任意读出被控制器地址‘_SubAddress_’指出的‘*_BYTES_*’字节的数据，送入由‘*_DestPointer_*’指出的存储区中。是否完成该过程由 *_RevSuccess* 标志。
6．I2C_READ *_BYTES_*, *_DestPointer_*
　可任意读出被控器当前地址中‘*_BYTES_*’字节的数据，送到由‘*_DestPointer_*’指出的存储区中。是否完成该过程由位标志 *_RcvSuccess* 显示。

标志

位标志名	含义
*_Bus_*Busy	开始和停止位标志
*_Txmt_*Progress	传送过程是否正在进行标志
*_Rcv_*Pregress	接收过程是否正在进行标志
*_Txmt_*Success	是否成功传送标志
*_Rcv_*Success	是否成功接收标志
*_Ack_*Error	是否有标识错误标志
_SlaveActive	从控制器是否工作标志

本说明中提供了一个源文件 I2C.ASM 和一个包含文件 I2C.INC。 在应用时，要将 I2C.ASM 文

件添加到用户的 project 中，并修改 I2C.INC 文件中的变量设置，以建立 SCL/SDA 引脚来与用户的应用电路相匹配。为了能够使用上述宏指令和标志位，须在源文件中加入 I2C.INC。同时注意在输入状态时，SCL 和 SDA 引脚必须加上上拉电阻。

实例：

```
；Filename: Software Implementation of I2C Bus Master
；作者：ANDY YU
；目的：用软件实现对 I2C 总线的控制
```

在本例中，单片机 HT48C50 用作为总线控制器而 EEPROM HT24C01 用作为从控制器。HT48C50 的 PB0 和 PB1 引脚分别接到 SCL 和 SDA。首先，程序用宏指令 I2C_TEST_DEVICE 测试 HT24C01 是否工作,然后把相关的地址用指令 I2C_WRITE_SUB 写入整个 HT24C01 的内存中，最后用指令 I2C_READ_SUB 把它们重新读出以供核对确认。注意 HT24C01 把它的 1K 内存分割为 16 页，每页 8 字节。以下所述仅供页写时使用。

```
;-------------------------------------------------------------
; File Name: TEST.ASM
; Author: Andy Yu
; Date: July 22, 1998
; Purpose: test i2c routines
;-------------------------------------------------------------
include i2c.inc
include ht48c50.inc
;
MySlaveAddr EQU 0a2h
;
tdata .section 'data'
PageData  db 8 dup (?)
Addr      db ?
PageCount db ?
Temp      db ?
Count     db ?
;
main .section at 0 'code'
   jmp    start
;
start:
   call   InitI2CBusMaster
;
LOAD_ADDR_8 MySlaveAddr
;
   I2C_TEST_DEVICE
   snz    _SlaveActive
   jmp    Error
;
```

```
;将地址写入HT24C01中
;
    clr    Addr
    mov    A,16                          ;页计数器，由16递减到1
    mov    PageCount,A
PageWriteLp:
    call   FillPageData                  ;把地址装入pagedata
I2C_WRITE_SUB 8, offset PageData, Addr
    snz    _Txmt_Success
    jmp    Error
AckPoll:
    I2C_TEST_DEVICE
    snz    _SlaveActive
    jmp    AckPoll
    mov    A,8
    addm   A,Addr                        ;指向下页首址
    sdz    PageCount
    jmp    PageWriteLp
;
;读出核对
;
    clr    Addr
    mov    A,16                          ;页计数器 16递减到1
    mov    PageCount,A
PageReadLp:
I2C_READ_SUB 8, offset PageData, Addr
    snz    _Rcv_Success
    jmp    Error
    call   CheckPageData
    snz    Acc.0
    jmp    Error
    mov    A,8
    addm   A,Addr                        ;指向下页首址
    sdz    PageCount
    jmp    PageReadLp
;
    jmp    $                             ;成功
Error:
    jmp    $                             ;失败
FillPageData proc
; PageData=Addr,Addr+1,...,Addr+7
    mov    A,offset PageData             ;指针MP0指向PageData
    mov    MP0,A
    mov    A,Addr                        ;Temp=Addr
```

```
    mov    Temp,A
    mov    A,8
    mov    Count,A                          ;计数值Count 8递减到1
FillLp:
    mov    A,Temp
    mov    R0,A
    inc    Temp
    inc    MP0
    sdz    Count
    jmp    FillLp
    ret
FillPageData endp
;
CheckPageData proc                          ; PageData 必须为
                                            ;Addr,Addr+1,...,Addr+7
    mov    A,offset PageData                ;指针MP0指向PageData数组首址
    mov    MP0,A
    mov    A,Addr                           ;Temp = Addr
    mov    Temp,A
    mov    A,8
    mov    Count,A                          ; 计数值Count = 8 递减到1
CheckLp:
    mov    A,R0
    XOR    A,Temp
    snz    Z
    ret    A,0
    inc    Temp
    inc    MP0
    sdz    Count
    jmp    CheckLp
    ret    A,1
CheckPageData endp
```

程序执行：

在初始化过程中，SCL和SDA的输出锁存器中都写0，此时它们都被设置为输入模式进入高阻态。设置SCL或SDA为输出模式时被控端口输出低电平。另一方面，在输入模式时由于带有上拉电阻，被控端口输出高电平。

在这段注释中，执行了两个文件I2C.INC(附件1)和I2C.ASM(附件2)。

附件 1: I2C.INC

```
;-------------------------------------------------------
; File Name: I2C.INC
; Author: Andy Yu
; Date: July 22, 1998
;-------------------------------------------------------
```

```
;
; 可以修改以下等式配合用户的应用程序
;
PortAddr EQU 14H                              ; SCL=PB0, SDA=PB1
SCL_Bit EQU 0
SDA_Bit EQU 1
I2CPORTMASK EQU 03h
_IAR EQU [00h]
_MP EQU [01h]
;
; 装入延时代码
;
DELAY_tSETUP_START macro                      ;设置开始条件的时间(600ns
Endm                                          ;HT24C0x)
;
DELAY_tHOLD_START macro                       ;开始条件持续的时间(600ns
Endm                                          ;HT24C0x)
;
DELAY_tSETUP_STOP macro                       ;设置停止条件的时间(600ns
Endm                                          ;HT24C0x)
;
DELAY_tBUS_FREE macro                         ;总线空闲时间(1300ns
Endm                                          ;HT24C0x)
;
DELAY_tLOW macro                              ;时钟为低的时间(600ns
Endm                                          ;HT24C0x)
;
DELAY_tHIGH macro                             ;时钟为高的时间(1300ns
Endm                                          ;HT24C0x)
;------------------------------------------------------------
ifndef I2C_ASM
extern InitI2CBusMaster : near
extern _i2c_block_write : near
extern _i2c_block_read : near
extern SendData : near
extern TxmtStartBit : near
extern Txmt_Slave_Addr : near
extern IsSlaveActive : near
extern _Bus_Busy : bit
extern _Txmt_Progress : bit
extern _Rcv_Progress : bit
extern _Txmt_Success : bit
extern _Rcv_Success : bit
extern _Ack_Error : bit
```

```
extern _10BitAddr : bit
extern _Slave_RW : bit
extern _SlaveActive : bit
extern DataByte : byte
extern SlaveAddr : byte
extern tempCount : byte
extern TempStore : byte
endif
;------------------
; LOAD_ADDR_8
;-------
LOAD_ADDR_8 macro SLAVE_ADDRESS
    clr    _10BitAddr
    mov    A,(SLAVE_ADDRESS and 0ffh)
    mov    SlaveAddr,A
endm
;--------------------
; I2C_TEST_DEVICE
;-------
I2C_TEST_DEVICE macro
    call   IsSlaveActive
endm
;--------------------
; RELEASE_BUS
;-------
RELEASE_BUS macro
    set    SDA                          ; 置为高阻态
    set    SCL
endm
;--------------------
; I2C_WRITE_SUB
;-------
I2C_WRITE_SUB macro _BYTES_, _SourcePointer_, _SubAddress_
    mov    A,_SubAddress_
    mov    TempStore,A
    mov    A,_BYTES_                    ; 循环初值
    mov    tempCount,A
    mov    A,_SourcePointer_
    mov    _MP,A
    call _i2c_block_write
endm
;--------------------
; I2C_READ
;-------
```

```
I2C_READ macro _BYTES_, _DestPointer_
   mov    A,_BYTES_-1
   mov    tempCount,A
   mov    A,_DestPointer_
   mov    _MP,A
   call   _i2c_block_read
endm
;-------------------
; I2C_READ_SUB
;-------
I2C_READ_SUB macro _BYTES_, _DestPointer_, _SubAddress_
   clr    _Slave_RW
   call   TxmtStartBit
   call   Txmt_Slave_Addr
   mov    A,_SubAddress_
   mov    DataByte,A
   call   SendData
   I2C_READ _BYTES_, _DestPointer_
endm
```

附件 2: I2C.ASM

```
;------------------------------------------------------------
; File Name: I2C.ASM
; Author: Andy Yu
; Date: July 22, 1998
;------------------------------------------------------------
#define I2C_ASM
include i2c.inc
public InitI2CBusMaster
public _i2c_block_write
public _i2c_block_read
public SendData
public TxmtStartBit
public Txmt_Slave_Addr
public IsSlaveActive
public _Bus_Busy
public _Txmt_Progress
public _Rcv_Progress
public _Txmt_Success
public _Rcv_Success
public _Ack_Error
public _10BitAddr
public _Slave_RW
public _SlaveActive
```

```
public DataByte
public SlaveAddr
public tempCount
public TempStore
;--------------------
; General Equate
;--------------------
TRUE          EQU 1
FALSE         EQU 0
LSB           EQU 0
MSB           EQU 7
Carry         EQU [0ah].0
;--------------------
; Port Assignment
;--------------------
PortCtrlAddr  EQU (PortAddr+1)           ; 控制端口地址15H
I2CPORT       EQU [PortAddr]
SCL           EQU [PortAddr+1].SCL_Bit    ;SCL=[15H].0
SDA           EQU [PortAddr+1].SDA_Bit    ;SDA=[15H].1
PSCL          EQU [PortAddr].SCL_Bit     ;PSCL=[14H].0
PSDA          EQU [PortAddr].SDA_Bit     ;PSDA=[14H].1
;-----------------------------------------------
; Data Section
;-----------------------------------------------
I2CData .section 'data'
I2CStatus label byte
_Bus_Busy        dbit
_Abort           dbit
_Txmt_Progress   dbit
_Rcv_Progress    dbit
_Txmt_Success    dbit
_Rcv_Success     dbit
_Fatal_Error     dbit
_Ack_Error       dbit
I2CCtrl label byte
_10BitAddr       dbit
_Slave_RW        dbit
_Last_Byte_Rcv   dbit
_reserve3        dbit
_reserve4        dbit
_reserve5        dbit
_SlaveActive     dbit
_Time_Out        dbit
DataByte         db ?
```

```
DataByteCopy     db ?
BitCount         db ?
SlaveAddr        db ?                    ; 保留两个字节供十位地址使用
tempCount        db ?
TempStore        db ?
DelayCount       db ?
;-------------------------------------------------
; Code Section
;-------------------------------------------------
I2CCode .section 'code'
;-------------------
; IsSlaveAcitve
;-------
IsSlaveActive proc
   clr    _Slave_RW
   call   TxmtStartBit
   call   Txmt_Slave_Addr
   clr    _SlaveActive
   snz    _Ack_Error
   set    _SlaveActive
   call   TxmtStopBit
   ret
IsSlaveActive endp
;-------------------
; _i2c_block_write
;-------
_i2c_block_write proc
   call   TxmtStartBit                  ; 开始
   clr    _Slave_RW                     ; 从寄存器地址
   call   Txmt_Slave_Addr
   snz    _Txmt_Success
   jmp    ending
   mov    A,TempStore                   ; 从寄存器数据区首址
   mov    DataByte,A
   call   SendData
_block_wr1_loop:
   snz    _Txmt_Success
   jmp    ending
   mov    A,_IAR
   mov    DataByte,A
   inc    _MP
   call   SendData
   sdz    tempCount
   jmp    _block_wr1_loop
```

```
ending:
    call   TxmtStopBit                    ; 数据传送结束
    ret
_i2c_block_write endp
;-------------------
; _i2c_block_read
;-------
_i2c_block_read proc
    call   TxmtStartBit
    set    _Slave_RW
    clr    _Last_Byte_Rcv
    call   Txmt_Slave_Addr
    sz     _Txmt_Success
    jmp    _block_rd1_loop
    call   TxmtStopBit
    ret    A,FALSE
_block_rd1_loop:
    call   GetData
    mov    A,DataByte
    mov    _IAR,A
    inc    _MP
    sdz    tempCount
    jmp    _block_rd1_loop
;
    set    _Last_Byte_Rcv
    call   GetData
    mov    A,DataByte
    mov    _IAR,A
    call   TxmtStopBit
    ret    A,TRUE
_i2c_block_read endp
;-------------------
; InitI2CBusMaster
;-------
InitI2CBusMaster proc
    mov    A,I2CPORT                      ; 清锁存器
    and    A,NOT I2CPORTMASK
    mov    I2CPORT,A
    RELEASE_BUS
    clr    I2CStatus
    clr    I2CCtrl
    ret
InitI2CBusMaster endp
;-------------------
```

```
; TxmtStartBit
;-------
TxmtStartBit proc
   set    SDA
   set    SCL
   DELAY_tSETUP_START
   clr    SDA
   DELAY_tHOLD_START
   set    _Bus_Busy
   ret
TxmtStartBit endp
;--------------------
; TxmtStopBit
;-------
TxmtStopBit proc
   clr    SCL
   clr    SDA
   set    SCL
   DELAY_tSETUP_STOP
   set    SDA
   DELAY_tBUS_FREE
   clr    _Bus_Busy
   ret
TxmtStopBit endp
;--------------------
; SendData
;-------
SendData proc
TxmtByte:
   mov    A,DataByte
   mov    DataByteCopy,A
   set    _Txmt_Progress
   clr    _Txmt_Success
   mov    A,08h                          ; 数据位计数器8到1
   mov    BitCount,A
TxmtNextBit:
   clr    SCL
   rlc    DataByteCopy
   clr    SDA
   sz     Carry
   set    SDA
   DELAY_tLOW
   set    SCL
   DELAY_tHIGH
```

```
    sdz    BitCount
    jmp    TxmtNextBit
;--- 检查是否 ACK
    clr    SCL
    set    SDA
    DELAY_tLOW
    set    SCL
    DELAY_tHIGH
    sz     PSDA
    jmp    TxmtErrorAck
    clr    SCL
    clr    _Txmt_Progress
    set    _Txmt_Success
    clr    _Ack_Error
    ret
TxmtErrorAck:
    RELEASE_BUS
    set    SDA
    set    SCL
    clr    _Bus_Busy
    clr    _Txmt_Progress
    clr    _Txmt_Success
    set    _Ack_Error
    ret
SendData endp
;-------------------
; GetData
;-------
GetData proc
RcvByte:
    set    _Rcv_Progress
    clr    _Rcv_Success
    mov    A,08h
    mov    BitCount,A
RcvNextBit:
    clr    SCL
    set    SDA
    DELAY_tLOW
    set    SCL
    DELAY_tHIGH
    clr    Carry
    sz     PSDA
    set    Carry
    rlc    DataByte
```

```
    sdz     BitCount
    jmp     RcvNextBit                  ;产生ACK信号
    clr     SCL
    clr     SDA
    sz      _Last_Byte_Rcv
    set     SDA
    DELAY_tLOW
    set     SCL
    DELAY_tHIGH
RcvEnd:
    clr     SCL
    clr     _Rcv_Progress
    set     _Rcv_Success
    clr     _Ack_Error
    ret
GetData endp
;------------------
; Txmt_Slave_Addr
;-------
Txmt_Slave_Addr proc
    clr     _Ack_Error
    snz     _10BitAddr
    jmp     SevenBitAddr
    snz     _Slave_RW
    jmp     TenBitAddrWR
TenBitAddrRd:
    clr     _Slave_RW
    call    TenBitAddrWR
    snz     _Txmt_Success
    ret     A,FALSE
    call    TxmtStartBit
    set     _Slave_RW
    mov     A,SlaveAddr[1]
    mov     DataByte,A
    set     DataByte.LSB
    call    SendData
    jmp     _AddrSendTest
;
    snz     _Txmt_Success
    jmp     _AddrSendFail
TenBitAddrWR:
    mov     A,SlaveAddr[1]
    mov     DataByte,A
    clr     DataByte.LSB
```

```
    call   SendData
    snz    _Txmt_Success
    jmp    _AddrSendFail
    mov    A,SlaveAddr
    mov    DataByte,A
    jmp    EndTxmtAddr
SevenBitAddr:
    mov    A,SlaveAddr
    mov    DataByte,A
    clr    DataByte.LSB
    sz     _Slave_RW
    set    DataByte.LSB
EndTxmtAddr:
    call   SendData
_AddrSendTest:
    snz    _Txmt_Success
    jmp    _AddrSendFail
    ret    A,TRUE
_AddrSendFail:
    snz    _Ack_Error
    ret    A,FALSE
    call   TxmtStopBit
    ret    A,FALSE
Txmt_Slave_Addr endp
```