

Controlling the I²C Bus with the HT48 & HT46 MCU Series

D/N : HA0005E

Introduction

The I²C bus includes only two lines, known as SCL and SDA, along which all communication takes place between any I²C bus device connected to these two lines. Among all devices connected to this bus, at least one will act as the bus master while the others will perform as slaves. The following application note shows the way in which the Holtek 8 bit RISC MCU is used as a bus master using software. The 2Kbit capacity, HT24LC02 EEPROM, is taken as an example here.

Circuit

The HT24LC02 pins A0, A1, A2, VSS and WP should be grounded. Pin VCC is connected to +5V while SCL is connected to PA3 and SDA is connected to PA2

Usage

Write data into the EEPROM first and then read the data out of the EEPROM to verify. If the data differs, then jump to fail_out; if the same, jump to ok_end.

The following note offers an original file OP_HT24.ASM and HT24.ASM with text. When applying, add OP_HT24.ASM to the user project and modify the functional setting of HT24.INC to build the SCL/SDA pins to go with the user circuit.

Program

```
-----  
; file name : op_ht24.asm  
; date : 2003/11/18  
; MCU: HT48R30A-1  
; EEPROM: HT24LC02  
; Fsys: 1MHz  
-----  
include ht24.asm  
  
data .section 'data'  
count db ?  
fail db ?  
success db ?  
  
-----  
main .section at 0 'code'  
clr count  
clr fail  
clr success  
  
clr pac  
clr pa  
  
write_cycle:  
mov a, 0a0h ;1010 0000  
mov deviceaddr, a  
mov a, count  
mov dataaddr, a  
mov data2, a  
  
call ht24_byte_write  
call write_delay ;delay for write  
siz count ;256 x 8 = 2K  
jmp write_cycle
```

```
read_cycle:
    mov     a, 0a0h           ;1010 0000
    mov     deviceaddr, a
    mov     a, count
    mov     dataaddr, a

    call    ht24_byte_read
    mov     a, data2
    xor     a, count
    sz     acc
    jmp     fail_out
    siz    count
    jmp     read_cycle

ok_end:
    set     success
    jmp     $

fail_out:
    set     fail
    jmp     $
```

In this program, HT24.INC(appendix 1) and HT24.ASM(appendix 2) will be executed.

Appendix 1 : HT24.INC

```
-----
; file name : HT24.INC
; date : 2003/11/18
-----

;-----
; modification of sda/sdac/scl/sclc pin definition according to the
; circuit is allowed
;-----

#define     port      [12H]
#define     portc     [13H]
sda        equ      port.2      ;data line
sdac       equ      portc.2     ;data control
scl        equ      port.3      ;clock line
sclc       equ      portc.3     ;clock control
```

```
-----  
;modification is not allowed in the parts below  
-----  
ifndef ht24_asm  
extern deviceaddr      :byte      ;element address and code  
extern dataaddr       :byte      ;address of the page  
extern data2          :byte      ;write and read byte data  
extern array          :byte      ;write and read page data  
extern num            :byte      ;page size  
extern ht24_byte_write :near      ;byte write  
extern ht24_byte_read  :near      ;byte read  
extern ht24_page_write :near      ;page write  
extern ht24_page_read  :near      ;page read  
extern write_delay     :near      ;delay about 5ms  
endif  
-----  
;End of file HT24.INC  
-----
```

Appendix 2 : HT24.ASM

```
-----  
; file name : ht24.asm  
; date : 2003/11/18  
; ROM status : 94H  
; RAM status : 16H  
-----  
#define          ht24_asm  
include         ht24.inc  
include         ht48r30a-1.inc  
;declare the function name for the external program  
public deviceaddr  
public dataaddr  
public data2  
public array  
public num  
;declare the subprogram name for the external program  
public ht24_byte_write  
public ht24_byte_read  
public ht24_page_write  
public ht24_page_read  
public write_delay
```

```

;-----
;data section
ht24data      .section          'data'
deviceaddr    db ?
dataaddr      db ?
data2         db ?
array         db 16 dup (?)
num           db ?
data1         db ?             ;temporary register
movb          db ?             ;8bit counter
;-----
;program section
ht48iiccode   .section          'code'
;-----
; decription: read a BYTE data of the EEPROM
; entrance argument: deviceaddr:byte
;                  dataaddr:byte
; exit argument:   data2 :byte
; STACK:         1
;-----
ht24_byte_read proc
    call    start                ;send start signal
    mov     a, deviceaddr        ;send device address & write
                                        ;op-code

    mov     data1, a
    call   byte_write
    mov     a,dataaddr          ;send data address
    mov     data1, a
    call   byte_write
    call   start
    mov     a, deviceaddr
    inc     acc                  ;send device address & read op-code
    mov     data1, a
    call   byte_write
    mov     a, offset data2
    mov     mp0, a
    call   byte_read            ;read designate address data
    call   stop                 ;send stop signal
    ret
ht24_byte_read      endp

```

```

;-----
; description: write a BYTE data into the EEPROM
; entrance argument: deviceaddr:byte
;                   dataaddr:byte
;                   data2:byte
; exit argument: none
; STACK: 1
;-----
ht24_byte_write      proc
    call    start          ;send start signal
    mov     a, deviceaddr  ;send device address & write
                                ;op-code

    mov     data1, a
    call    byte_write
    mov     a, dataaddr    ;send data address
    mov     data1, a
    call    byte_write
    mov     a, data2      ;send data
    mov     data1, a
    call    byte_write
    call    stop          ;send stop signal
    ret

ht24_byte_write      endp
;-----
; description: read a page data from the EEPROM
; entrance argument: deviceaddr:byte
;                   dataaddr:byte
; exit argument: array      :16 bytes
; STACK: 1
;-----
ht24_page_read      proc
    call    start          ;send start signal
    mov     a, deviceaddr
    mov     data1, a      ;address judgement, send address
                                ;and code

    call    byte_write
    mov     a, dataaddr
    mov     data1, a      ;send page address
    call    byte_write
    call    start
    mov     a, deviceaddr
    inc     acc
    mov     data1, a      ;page address judgement, send page
                                ;address and code

    call    byte_write
    mov     a, offset array ;save the data from the EEPROM to
                                ;ARRAY
    mov     mp0, a

```

```

more:
    call    byte_read        ;call read BYTE subprogram
    sdz    num
    jmp    cmack
    jmp    retu
cmack:
    call    mack
    inc    mp0
    jmp    more
retu:
    call    stop            ;send stop signal
    ret
ht24_page_read endp
;-----
; description: write a PAGE data to the EEPROM
; entrance argument: deviceaddr :byte
;                   dataaddr   :byte
;                   array      :16 bytes
; exit argument:    none
; STACK: 1
;-----
ht24_page_write    proc
    call    start          ;send start signal
    mov     a, deviceaddr
    mov     data1, a       ;send page address and write code
    call    byte_write
    mov     a, dataaddr
    mov     data1, a       ;send page address
    call    byte_write
    mov     a, offset array ;write the ARRAY data to the EEPROM
    mov     mp0, a
amore:
    mov     a, r0          ;send data to the EEPROM
    mov     data1, a
    call    byte_write
    nop
    sdz    num
    jmp    amore
    call    stop          ;send stop signal
    ret
ht24_page_write    endp

```

```

;-----
; description: write delay after the data, about 5ms (fsys=4MHz)
; entrance argument: none
; exit argument: none
; STACK: none
;-----
write_delay proc
    set    data1
    mov    a,06h
    mov    movb,a
lpy1:
    sdz    data1
    jmp    lpy1
    sdz    movb
    jmp    lpy1
    ret
write_delay endp
;-----
;send signal 0 to EEPROM
;-----
mack proc
    clr    sdac
    clr    sda
    nop
    set    scl
    nop
    nop
    clr    scl
    nop
    set    sda
    ret
mack endp
;-----
;start signal subprogram
;-----
start proc
    clr    sclc
    clr    sdac
    set    scl
    nop
    set    sda
    nop
    clr    sda
    nop
    clr    scl
    ret
start endp

```

```

;-----
;finish signal subprogram
;-----
stop   proc
        clr     sdac
        clr     sda
        nop
        set     scl
        nop
        set     sda
        nop
        clr     sda
        clr     scl
        ret
stop   endp
;-----
;BYTE subprogram
;-----
byte_write   proc
        mov     a, 08h           ;serial input 8 bit data
        mov     movb, a
loop:
        clr     sdac
        clr     scl             ;write data, scl as high
        rl     data1
        mov     a, data1
        snz    acc.0           ;data serial input
        jmp     loop_1
        set     sda
        jmp     loop_2
loop_1:
        clr     sda
loop_2:
        nop
        set     scl
        nop
        nop
        sdz    movb
        jmp     loop
        clr     scl
        set     sdac
        nop
        set     scl             ;read figure, scl as low
rep:
        sz     sda             ;check EEPROM has sent signal or not
        jmp     rep
        clr     scl
        ret
byte_write   endp

```

```
-----  
;byte data read subprogram  
-----  
byte_read      proc  
    mov     a, 08h           ;serial input 8 bit data  
    mov     movb, a  
    set     sdac  
    nop  
loops:  
    set     scl             ;read figure, scl as low  
    rl     r0  
    snz    sda             ;data serial input  
    jmp    loops_1  
    set     r0.0  
    jmp    loops_2  
loops_1:  
    clr     r0.0  
loops_2:  
    clr     scl  
    nop  
    sdz    movb  
    jmp    loops  
    ret  
byte_read      endp  
-----  
;End of the program  
-----
```