

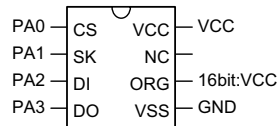
Communicating Between the HT48 & HT46 Series MCUs and the HT93LC46 EEPROM

D/N : HA0003E

Introduction

The HT93LC46 is Holtek's 1K bit series EEPROM (Electrically Erasable Programmable Read Only Memory) Such EEPROM memory is commonly used in the fixed data storage of microcontrollers. The following application takes Holtek's 8 bit MCU as an example to introduce the general operating routines for this device. Users only have to add the code into their program and connect the CS/SK/DI/DO pins before using the HT93LC46 device.

Circuit



Usage

The following example uses the HT93LC46 EEPROM which will be controlled by the HT48R30A-1. When the ORG pin is connected to VCC, the memory will be organized into 64×16-bits. An original assembly file OP16_93LC46.ASM will be used in this example. Within the HT-IDE environment, the following steps should be implemented before using the interface function :

Step1 : add OP16_93LC46.ASM to the project (use [Project/Edit] instruction)

Step2 : modify the HT93LC46.ASM file and connect CS/SK/DI/DO pins according to the program

Step3 : include the HT93LC46.ASM file and call the interface function

note : set properly the input/output mode before using these functions

Create a HT48R30A-1 project, and add the OP16_93LC46.ASM file into the project. In this case, use the HT93LC46_EWEN function first to write enable the HT93LC46 and then use the HT93LC46_WRITE function to write the relative address into the memory. Lastly use the HT93LC46_READ instruction to read and check the data just written.

Program

```
-----  
; file name : op16_93lc46.asm  
; date : 2003/11/21  
; MCU: HT48R30A-1  
; EEPROM: HT93LC46, 64x16-bit  
-----  
include ht48r30a-1.inc  
include ht93lc46.asm           ;see appendix 1  
  
data .section      'data'  
temp db ?  
  
main .section at 0  'code'  
start:  
    clr    csc  
    clr    skc  
    clr    dic  
    set    doc  
  
;-----  
    call   ht93lc46_ewen      ;ht93lc46 write enable  
    nop  
    mov    a,3fh              ;ORG pin VCC,64x16  
    mov    temp,a  
write:  
    mov    a, temp  
    mov    dataaddr, a  
    mov    data2, a  
    cpl    acc  
    mov    data3, a  
    call   ht93lc46_write     ;ht93lc46 write in  
    sdz    temp  
    jmp    write  
    call   ht93lc46_ewds     ;ht93lc46 write in disable  
;-----
```

```

; start reading data and compare
mov     a,3fh                ;ORG pin VCC,64x16
mov     temp,a
read:
clr     data2
clr     data3
mov     a, temp
mov     dataaddr, a
call    ht93lc46_read        ;ht93lc46 read

mov     a, temp              ;compare data
xor     a, data2
snz     z
jmp     fail
cpla    temp
xor     a, data3
snz     z
jmp     fail
sdz     temp
jmp     read

;-----
call    ht93lc46_ewen        ;ht93lc46 write enable
nop
call    ht93lc46_eral        ;ht93lc46 clear all space
nop
mov     a,3fh                ;ORG pin VCC,64x16
mov     temp,a
read_1:
clr     data3
clr     data2
mov     a, temp
mov     dataaddr, a
call    ht93lc46_read        ;ht93lc46 read

mov     a, 0ffh              ;compare data
xor     a, data2
snz     z
jmp     fail
mov     a, 0ffh
xor     a, data3
snz     z
jmp     fail
sdz     temp
jmp     read_1

;-----
mov     a, 04bh
mov     data2, a
mov     a, 0c3h

```

```

        mov     data3, a
        call   ht93lc46_wral      ;ht93lc46 write to all locations
        nop
        mov     a, 3fh           ;ORG pin VCC,64x16
        mov     temp,a
read_2:
        clr     data2
        clr     data3
        mov     a, temp
        mov     dataaddr, a
        call   ht93lc46_read     ;ht93lc46 read

        mov     a, 04bh         ;compare data
        xor     a, data2
        snz     z
        jmp     fail
        mov     a, 0c3h
        xor     a, data3
        snz     z
        jmp     fail
        sdz     temp
        jmp     read_2

;-----
        mov     a, 3fh           ;ORG pin VCC,64x16
        mov     temp,a
write_1:
        mov     a, temp
        mov     dataaddr, a
        call   ht93lc46_erase   ;ht93lc46 clear designated address
        sdz     temp
        jmp     write_1
        nop
        mov     a, 3fh           ;ORG pin VCC,64x16
        mov     temp,a
read_3:
        clr     data3
        clr     data2
        mov     a, temp
        mov     dataaddr, a
        call   ht93lc46_read     ;ht93lc46 read

        mov     a, 0ffh         ;compare data
        xor     a, data2
        snz     z
        jmp     fail
        mov     a, 0ffh
        xor     a, data3
        snz     z
        jmp     fail

```

```

        sdz      temp
        jmp      read_3

        jmp      $                ;successful operation
fail:   jmp      $                ;operation failed
;-----
;program op16_93lc46.asm over
;-----

the following is the function call source file of appendix 1,
including seven instructions of HT93LC46.
;-----
; file name:ht93lc46.ASM
; date:2003/11/21
; ROM status : 9DH
; RAM status : 07H
;-----
#define      ht93lc46_asm

;operating code
oc_read      equ      10000000b
oc_erase     equ      11000000b
oc_write     equ      01000000b
oc_ewen      equ      00110000b
oc_ewds      equ      00000000b
oc_eral      equ      00100000b
oc_wral      equ      00010000b
;-----
;if ORG pin VCC, then mask bit8, 64x16
;if ORG pin VSS, then mask bit16, 128x8
;-----
#define      bit16
#define      bit8
;-----
;modify CS/SK/DI/DO definition according to the actual circuit
;-----
port        equ [12H]
portc       equ [13H]
#define     sk port.1
#define     skc portc.1
#define     di port.2
#define     dic portc.2
#define     do port.3
#define     doc portc.3
#define     cs port.0
#define     csc portc.0
;-----
;the following parts cannot be modified

```

```

;-----
ifndef ht93lc46_asm
extern dataaddr          :byte
extern data3             :byte
extern data2             :byte
extern ht93lc46_write    :near
extern ht93lc46_read     :near
extern ht93lc46_ewen    :near
extern ht93lc46_ewds    :near
extern ht93lc46_eral    :near
extern ht93lc46_wral    :near
extern ht93lc46_erase   :near
endif
;-----
;declare the variables provided to the external program
public dataaddr          ;page address of data
public data3             ;read high 8 bit data
public data2             ;save low 8 bit data
;declare the subroutine provided to the external program
public ht93lc46_eral    ;
public ht93lc46_wral    ;
public ht93lc46_ewen    ;
public ht93lc46_ewds    ;
public ht93lc46_write   ;
public ht93lc46_read    ;
public ht93lc46_erase   ;
;-----
;data section
ht93lc46data .section      'data'
dataaddr     db ?          ;operating address
data3        db ?          ;operand data register, high 8 bit
data2        db ?          ;operand data register
data1        db ?          ;rolling bit register
movb         db ?          ;loop transpose accumulator
;register
reg           db ?          ;delay data register
reg1         db ?          ;delay data register
;-----
;code section
ht93lc46code .section      'code'
;-----
; READ—read data
; description: read data from EEPROM designated address
; entrance function: dataaddr:byte redesignated address
; exit function:      data2          :byte read low 8 bit data
;                   data3          :byte read high 8 bit data
; stack:             1
;-----

```

```

ht93lc46_read      proc
    call    ht93_start      ;start signal
    mov     a, oc_read
    mov     data1, a
    mov     a,2             ;write in 2 bit op-code
    call    wbit
    mov     a, dataaddr
    mov     data1, a
    rl      data1
    ifdef   bit8
        mov     a,7             ;write in 7 bit dataaddr
    endif
    ifdef   bit16
        rl      data1
        mov     a,6             ;write in 6 bit dataaddr
    endif
    call    wbit
    nop
    call    rbit
    ifdef   bit16
        mov     a, data2
        mov     data3, a
        call    rbit
    endif
    clr     cs
    ret
ht93lc46_read endp
;-----
; WRITE—write data
; description: write data from EEPROM designated address
; entrance function: dataaddr:byte designated address
;                 data2 :byte write low 8 bit data
;                 data3 :byte write high 8 bit data
; exit function:   none
; stack:         1
;-----
ht93lc46_write proc
    call    ht93_start
    mov     a, oc_write
    mov     data1, a
    mov     a,2             ;write 2 bit op-code
    call    wbit
    mov     a, dataaddr
    mov     data1, a
    rl      data1
    ifdef   bit8
        mov     a,7             ;write 7 bit dataaddr
    endif
endif

```

```

        ifdef    bit16
            rl    data1
            mov   a,6           ;write 6 bitdataaddr
        endif
        call    wbit
        nop
        ifdef    bit16
            mov   a, data3
            mov   data1, a
            mov   a,8
            call  wbit
        endif
        mov   a, data2
        mov   data1, a
        mov   a,8
        call  wbit
        clr   cs
        call  delay
        call  mverify
        clr   cs
        ret
ht93lc46_write endp
;-----
; ERASE—erase data
; description:write data 1 to EEPROM designated address
; entrance function: dataaddr:byte designate address
; exit function: none
; stack: 1
;-----
ht93lc46_erase proc
    call    ht93_start
    mov     a, oc_erase
    mov     data1, a
    mov     a,2           ;write 2 bit op-code
    call    wbit
    mov     a, dataaddr
    mov     data1, a
    rl     data1
    ifdef   bit8
        mov     a,7           ;write 7 bit dataaddr
    endif
    ifdef   bit16
        rl     data1
        mov     a,6           ;write 6 bit dataaddr
    endif
    call    wbit
    clr     cs
    call    delay

```

```

        call    mverify
        clr     cs
        ret
ht93lc46_erase endp
;-----
; EWDS—write disable
; description: write disable, disable the write function of the EEPROM
; entrance function: none
; exit function: none
; stack: 1
;-----
ht93lc46_ewds proc
    call    ht93_start
    mov     a, oc_ewds
    mov     data1, a
    mov     a, 8
    call    wbit
    call    wbit
    ;write 8 bit op-code
    ifdef  bit8
        clr     sk
        set     sk
        nop
    endif
    clr     sk
    clr     cs
    ret
ht93lc46_ewds endp
;-----
; EWEN—write enable
; description: write enable, enable the write function of the EEPROM
; entrance function: none
; exit funxtion: none
; stack: 1
;-----
ht93lc46_ewen proc
    call    ht93_start
    mov     a, oc_ewen
    mov     data1, a
    mov     a, 8
    call    wbit
    call    wbit
    ;write 8 bit op-code
    ifdef  bit8
        clr     sk
        set     sk
        nop
    endif
    clr     sk
    clr     cs
    ret
ht93lc46_ewen endp

```

```

;-----
; ERAL—clean all address
; description:write all addresses into EEPROM data1
; entrance function: none
; exit function: none
; stack: 1
;-----
ht93lc46_eral proc
    call    ht93_start
    mov     a, oc_eral
    mov     data1, a
    mov     a,8                               ;write 8 bit op-code
    call    wbit
    ifdef   bit8
        clr     sk
        set     sk
        nop
    endif
    clr     sk
    clr     cs
    call    delay
    call    mverify
    clr     cs
    ret
ht93lc46_eral endp
;-----
; WRAL—write all addresses
; description:write all data into EEPROM space
; entrance function: data2 :byte write low 8 bit data
;                   data3 :byte write high 8 bit data
; exit function: none
; stack: 1
;-----
ht93lc46_wral proc
    call    ht93_start
    mov     a, oc_wral
    mov     data1, a
    mov     a,8                               ;write 8 bit op-code
    call    wbit
    ifdef   bit8
        clr     sk
        set     sk
        nop
    endif
    clr     sk

```

```

        ifdef    bit16
            mov    a, data3
            mov    data1, a
            mov    a, 8
            call   wbit
        endif
        mov    a, data2
        mov    data1, a
        mov    a, 8
        call   wbit
        clr    cs
        call   delay
        call   mverify
        clr    cs
        ret
ht93lc46_wral endp
;-----
; start signal
;-----
ht93_start    proc
        set     cs
        clr     sk
        set     di
        nop
        nop
        set     sk
        nop
        clr     sk
        ret
ht93_start    endp
;-----
; write n bit data subroutine, n decided by acc
;-----
wbit          proc
        mov     movb, a
loop1:
        clr     sk
        rl     data1
        snz    data1.0
        jmp    loop1_1
        set    di
        jmp    loop1_2
loop1_1:
        clr     di

```

```
loop1_2:
    nop
    set    sk
    nop
    sdz   movb
    jmp   loop1
    clr   sk
    ret
wbit   endp
;-----
;read 8 bit data subroutine
;-----
rbit   proc
    mov   a, 08h
    mov   movb, a
loop_r:
    rl    data2
    set   sk
    nop
    snz   do
    jmp   loops_0
    set   data2.0
    jmp   loops_1
loops_0:
    clr   data2.0
loops_1:
    clr   sk
    sdz   movb
    jmp   loop_r
    ret
rbit   endp

;-----
;test if DO has HIGH signal, namely operation complete
;-----
mverify   proc
    set    cs
    nop
    nop
    nop
check:
    snz   do
    jmp   check
    ret
mverify   endp
```

```
-----  
delay proc  
    set     reg1  
    mov     a, 06h  
    mov     reg, a  
lpy:  
    sdz     reg1  
    jmp     lpy  
    sdz     reg  
    jmp     lpy  
    ret  
delay endp  
-----  
;End of program ht93lc46.ASM  
-----
```