

Reading and Writing to the HT93LC46 with the HT49 MCU Series

D/N : HA0044E

Introduction

The HT93LC46 is a 1K capacity EEPROM. This kind of memory type is electrically erasable and can therefore be erased and overwritten. Normal usage for this type of memory is to save MCU data that does not change. In this project, the Holtek HT49 series is taken as an example to introduce the frequently used functions. The user only needs to place the code into their program in addition to connecting the CS/SK/DI/DO pins before use.

Functional Description

This projects uses the HT93LC46S.ASM file and OP16_93LC46S.ASM files. In the HT-IDE3000 development system environment, before using any of the functions, the following steps should be carried out:

Step 1: Add the OP16_93LC46S.ASM into the project – use Project/Edit instruction

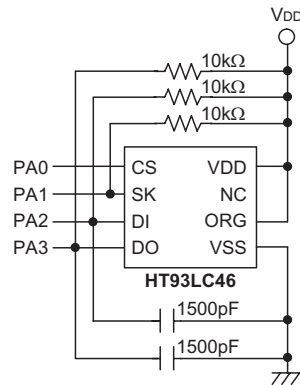
Step 2: According to the circuit, modify the HT93LC46S.INC file to connect the CS/SK/DI/DO pins

Step 3: Include the HT93LC46.ASM internal call interface functions

Note: Before using these functions, it is necessary to setup the input/output mode

Usage Explanation

Hardware Circuit



Configuration Options

PA0 ~ PA3 : NMOS

PA0 ~ PA3 : PULL HIGH

Program Example

First create a project using the HT49R30A-1 device and add the OP16_93LC46S.ASM file to the project.

In this example, first use the HT93LC46_EWEN function to write an enable to the HT93LC46. Afterwards use the HT93LC46_WRITE function to write the related address to the memory, then use the HT93LC46_READ instruction to read back the data just written and to check that the data is correct.

Program:

```

;-----
; filename : op16_93lc46s.asm
; Date : 2003/11/26
; MCU: HT49R30A-1
; EEPROM: HT93LC46, 64x16 bits
;-----
include ht49r30a-1.inc
include ht93lc46s.asm

data.section 'data'
temp db ?

```

```
main.section at 0 'code'
start:
    clr pa
    clr porta
    set porta.3
    mov a,porta
    mov pa,a
;-----
    call ht93lc46_ewen          ;ht93lc46 write enable
    nop
    mov a,3fh                  ;ORG pin connected to VCC,64x16
    mov temp,a
write:
    mov a, temp
    mov dataaddr, a
    mov data2, a
    cpl acc
    mov data3, a
    call ht93lc46_write        ;ht93lc46write
    dec temp
    snz temp.7
    jmp write
    call ht93lc46_ewds        ;ht93lc46 write disable
;-----
; start read and compare
    mov a,3fh                  ;ORG pin connected to 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
    dec temp
    snz temp.7
    jmp read
;-----
    call ht93lc46_ewen        ;ht93lc46 write enable
    nop
    call ht93lc46_eral        ;ht93lc46 erase all space
```

```
        nop
        mov  a,3fh                                ;ORG pin connected to 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
        dec  temp
        snz  temp.7
        jmp  read_1
;-----
        mov  a, 04bh
        mov  data2, a
        mov  a, 0c3h
        mov  data3, a
        call ht93lc46_wral                        ;ht93lc46 write all space
        nop
        mov  a,3fh                                ;ORG pin connected to 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
        dec  temp
        snz  temp.7
        jmp  read_2
;-----
        mov  a,3fh                                ;ORG pin connected to VCC, 64x16
```

```

        mov temp,a
write_1:
        mov a, temp
        mov dataaddr, a
        call ht93lc46_erase           ;ht93lc46 clear data at
;specified address
        dec temp
        snz temp.7
        jmp write_1
        nop
        mov a,3fh                    ;ORG pin connected to 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
        dec temp
        snz temp.7
        jmp read_3

        jmp $                         ;operation successful
fail:
        jmp $                         ;operation failed
;-----
;program op16_93lc46s.asm end
;-----

```

The following is the primary file 1 (HT93LC46S.ASM)
The primary file calling interface functions includes the 7
HT93LC46 instructions

```

;-----
; Filename:ht93lc46.ASM
; Date:2003/11/26
; ROM usage conditions:C8H
; RAM usage conditions:08H
;-----
#define ht93lc46_asm

```

```
;Control codes
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 the ORG pin is connected to VCC, then mask bit8,64x16
; If the ORG pin is connected to VSS, then mask bit16,128x8
;-----
#define bit16
;#define bit8

;-----
;The CS/SK/DI/DO pins can be defined according to individual
; circuit requirements
;-----
porta equ      [70h]
sk equ      porta.1
di equ      porta.2
do equ      porta.3
cs equ      porta.0
;pa equ      [12h]
doin equ     pa.3

;-----
;The following must not be changed
;-----
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 variables for external programs
public dataaddr ;data page internal address
public data3 ;saved high 8-bit data
public data2 ;saved low 8-bit data
```

```

;declare subroutines for external program use
public ht93lc46_eral ;
public ht93lc46_wral ;
public ht93lc46_ewen ;
public ht93lc46_ewds ;
public ht93lc46_write ;
public ht93lc46_read ;
public ht93lc46_erase ;
public delay

;-----
;data section
ht93lc46data .section 'data'
dataaddr db ? ;address
data3 db ? ;high 8-bit operation register
data2 db ? ; operation register
data1 db ? ;implement shift data register
movb db ? ;return shift number regsiter
reg db ? ;delay data register
reg1 db ? ;delay data register
;-----

;-----
;code section
ht93lc46code .section 'code'

;-----
; READ - read data
; description: read data from a specified EEPROM address
; input parameters dataaddr:byte indicated address
; output parameters : data2 :byte read data low 8-bits
; data3 :byte read data high 8-bits
; stack used: 1
;-----
ht93lc46_read proc
call ht93_start ;start signal
mov a, oc_read
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 data addr
endif
ifdef bit16
rl data1
mov a,6 ;write 6 data addr

```

```
endif
call      wbit
nop
call      rbit
ifdef    bit16
    mov    a, data2
    mov    data3, a
    call   rbit
endif
clr      cs
call     porta2pa
ret
ht93lc46_read    endp
;-----
; WRITE-write data
; description: write data to a specified EEPROM address
; input parameters :dataaddr:byte specified address
;                   data2       :byte write data low 8-bits
;                   data3       :byte write data high 8-bits
; output parameters: none
; stack used: 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 data addr
    endif
    ifdef bit16
        rl    data1
        mov    a,6                                ;write 6-bit data addr
    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 porta2pa
        call delay
        call mverify
        clr cs
        call porta2pa
        ret
ht93lc46_write      endp

;-----
; ERASE-- clear data
; description: write "1" data to the EEPROM at specified address
; input parameters : dataaddr:byte specified address
; output parameters : none
; stack used : 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 porta2pa
        call delay
        call mverify
        clr cs
        call porta2pa
        ret
ht93lc46_erase     endp

;-----
; EWDS--write disable
; description: write disable, EEPROM will not execute write operation
; input parameters: none
; output parameters: none
; stack used : 1
;-----

```

```
ht93lc46_ewds    proc
    call ht93_start
    mov a, oc_ewds
    mov data1, a
    mov a,8                      ;write 8-bit op-code
    call wbit
    ifdef bit8
        clr    sk
        call  porta2pa
        set    sk
        call  porta2pa
        nop
    endif
    clr sk
    call porta2pa
    clr cs
    call porta2pa
    ret
ht93lc46_ewds    endp
;-----
; EWEN--write enable
; description: write enable, permits EEPROM to execute write operations
; input parameters: none
; output parameters: none
; stack used: 1
;-----
ht93lc46_ewen    proc
    call ht93_start
    mov a, oc_ewen
    mov data1, a
    mov a,8                      ;write 8-bit op-code
    call wbit
    ifdef bit8
        clr    sk
        call  porta2pa
        set    sk
        call  porta2pa
        nop
    endif
    clr sk
    call porta2pa
    clr cs
    call porta2pa
    ret
ht93lc46_ewen    endp
;-----
; ERAL- clear all addresses
; description: write a "1" to all EEPROM locations
; input parameters: none
```

```
; output parameters: none
; stack used: 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
        call    porta2pa
        set     sk
        call    porta2pa
        nop
    endif
    clr sk
    call porta2pa
    clr cs
    call porta2pa
    call delay
    call mverify
    clr cs
    ret
ht93lc46_eral    endp
;-----
; WRAL--write to all addresses
; description: write data to all EEPROM locations
; input parameters: data2 :byte write data low 8-bits
;                  data3 :byte write data high 8-bits
; output parameters: none
; stack used: 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
        call    porta2pa
        set     sk
        call    porta2pa
        nop
    endif
    clr sk
    call porta2pa
    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 porta2pa
    call delay
    call mverify
    clr cs
    call porta2pa
    ret
ht93lc46_wral    endp
;-----
;start signalr
;-----
ht93_start      proc
    set cs
    call porta2pa
    clr sk
    call porta2pa
    set di
    call porta2pa
    nop
    nop
    set sk
    call porta2pa
    nop
    clr sk
    call porta2pa
    ret
ht93_start      endp
;-----
; write the n bit data subroutine, n is determined by the acc
;-----
wbit      proc
    mov movb, a
loop1:
    clr sk
    call porta2pa
    rl  data1
    snz data1.0
    jmp loop1_1
    set di
    call porta2pa
```

```

        jmp loop1_2
loop1_1:
        clr di
        call porta2pa
loop1_2:
        nop
        set sk
        call porta2pa
        nop
        sdz movb
        jmp loop1
        clr sk
        call porta2pa
        ret
wbit    endp
;-----
;read 8-bit data subroutine
;-----
rbit    proc
        mov a, 08h
        mov movb, a
loop_r:
        rl data2
        set do                                ;setup as input to receive data
        call porta2pa
        set sk
        call porta2pa
        nop
        snz doin
        jmp loops_0
        set data2.0
        jmp loops_1
loops_0:
        clr data2.0
loops_1:
        clr sk
        call porta2pa
        sdz movb
        jmp loop_r
        ret
rbit    endp

;-----
;check if D0 is HIGH signal, check if operation completed
;-----
mverify proc
        set cs
        call porta2pa
        nop

```

```
        nop
check:
        set     do                ;setup as input, receive answer
                                   ; (busy or ready)

        call  porta2pa
        snz   doin
        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
;-----
porta2pa proc
        mov  a,porta
        mov  pa,a
        jmp  $+1                ;HT49 I/O require a fixed delay
        jmp  $+1
        jmp  $+1
        jmp  $+1
        ret
porta2pa endp
;-----
;program ht93lc46s.ASM terminates
;-----
```

When writing this program, take note of the following points:

- Because the HT49 series have simple I/O structures, there are no I/O control registers. For the more complex I/O structures, the execution of a read-modify-write operation is simpler. The supplied program example uses a mapping register and the MOV instruction for I/O operations.
- Every time a SET instruction is executed on the I/Os, a short delay must be added afterwards; in the present example, each time an I/O changes, state a delay is added.
- In the circuit provided, the resistor and capacitor values are dependent upon the PCB used.