

# Controlling the Read/Write Function of the HT24 Series EEPROM using the HT49 Series MCUs

D/N : HA0017E

## Introduction

The read and write function of the HT24 series EEPROM is controlled by the I<sup>2</sup>C protocol. The interface of the HT49 series MCU uses simple I/O ports that will make it more convenient to control the peripheral components by I<sup>2</sup>C protocol.

The HT24 series EEPROM has 8 pins. Among them, three are the chip address pins A0, A1, A2. While operating, data input from SDA has to correspond with the external A0, A1, A2 addresses. Another pin is the write-protection pin WP. When the WP pin is low, the R/W line can be controlled, when the WP pin is high only the read function is valid. The other two pins are the power pins VCC and VSS.

When using MCUs to control the HT24 series EEPROMs, the external pins of the HT24 series VCC, VSS, WP, A0, A1, A2 should correspond with one another. SDA and SCL should connect to the MCU pins.

Pin	I/O	Function Description
A0~A2	I	Address input
VSS	I	Negative power input
SDA	I/O	Serial data input/output
SCL	I	Serial data transmission clock signal input
WP	I	Write-protection
VCC	I	Positive power input

The HT24 series EEPROM capacity differs according to their part numbers. When the EEPROM size is bigger than 1 page (256 bytes) or 2048bits, the HT49 series MCUs need to control pins A0, A1, A2 to ensure the correct HT24 series EEPROM is addressed. The sizes of the HT24 series EEPROM are as follows.

Part No.	Usage of Pin A0, A1 and A2	Size
HT24LC02	A0, A1, A2 pins as component address input. Data input from SDA must correspond with each pin status	2K (256×8)
HT24LC04	A1, A2 pins as component address input. A1, A2 data input from SDA must correspond with A1 and A2 pin status. A0 pin keeps empty.	4K (512×8, 2 pages)
HT24LC08	A2 pin as component address input. A2 data input must correspond with A2 pin status, and the rest empty.	8K (1024×8, 4 pages)
HT24LC16	A0, A1, A2 need no connection	16K (2048×8, 8 pages)

## Usage

The following note uses the HT24LC04 which is controlled by the HT49R30A-1 as an example. I<sup>2</sup>C protocol can be operated simply by two lines: SCL and SDA. When controlling the HT24LC04 with an MCU, the HT24LC04 external pins VCC, VSS, WP, A1 and A2 should be connected correspondingly. In this example A1 and A2 are connected to VSS, namely A1, A2=00. The SDA and SCL pins connect to the MCU control pins. Here the SCL pin is connected to PA.3 and the SDA pin is connected to PA.1. Initially write 55H to some address in the EEPROM and then read the data out to make a comparison with 55H. If the data is incorrect, the program will jump to fail\_out. If the data is correct, the program will jump to ok\_end.

In the program, the write action is only executed to some specific address of page 0 before reading the address. As for the HT24 series EEPROM, just operate carefully according to the time order and notice if the A0, A1, A2 data input from SDA corresponds to the A0, A1, A2 pins correctly.

## Program Example

```

; file name:      4924_WR.asm
; writer :       Holtek(shanghai) Semiconductor Inc. Software Dept.
; Purpose :      Control HT24 series EEPROM by HT49 MCUs
; Note: in mask_option, PA0-PA3 must choose NMOS pull-high
resistance

include ht49r30a-1.inc
;-----

```

```
;equ definition section
scl          equ    pa.3      ;clock signal pin definition
sda          equ    pa.1      ;serial data pin definition
read_out     equ    [70h]     ;write register
write_in     equ    [71h]     ;read register
word_address equ    [72h]
data_8       equ    [73h]

;-----

;-----
;macro definition
;delay macro, delay 100MS
d_1          macro
    local label
    mov      a,64h
    mov      delay,a
label:
    sdz      delay
    jmp      label
endm
;-----

;-----
;data section
e2prom .section 'data'
    delay   db    ?
;-----

;-----
;code section
EEPROMc .section 'code'
    org    00h
    jmp    start
start:
    mov    a,055h                ;set write value 055H
    mov    write_in,a
    mov    a,14h                 ;write 14H as the EEPROM operation
                                ;address
    mov    word_address,a
random_write_cycle:
    set    sda
d_1
    set    scl
    d_1
    clr    sda                    ;start signal
```

```
clr    scl
set    sda                ;1
d_1
set    scl
d_1

clr    scl
clr    sda                ;0
set    scl
d_1

clr    scl
set    sda                ;1
d_1
set    scl
d_1

clr    scl
clr    sda                ;0
set    scl
d_1

clr    scl
clr    sda                ;a2,a1,a0=0
set    scl
d_1

clr    scl
set    scl
d_1

clr    scl
set    scl
d_1

clr    scl
clr    sda                ;0 write mode
set    scl
d_1

clr    scl
set    sda                ;1 for ack, set as input port for
                        ;receiving acknowledge signals
d_1
set    scl                ;read_modify_write
d_1
```

```
skch:
    sz    sda                ;reply signal
    jmp   skch
    clr   scl
    mov   a,08h
    mov   data_8,a          ;8 bit for one byte
write_address_in:
    clr   sda
    sz    word_address.7
    set   sda
    d_1
    set   scl
    d_1

    clr   scl
    rl    word_address
    sdz   data_8
    jmp   write_address_in
    set   sda
    d_1
    set   scl
    d_1

wdow:
    sz    sda
    jmp   wdow
    clr   scl
    mov   a,08h
    mov   data_8,a
write_data_in:
    clr   sda
    sz    write_in.7
    set   sda
    d_1
    set   scl
    d_1

    clr   scl
    rl    write_in
    sdz   data_8
    jmp   write_data_in

    clr   sda
    set   scl
    d_1
    clr   scl
    set   scl
    d_1
```

```
        set    sda                ;stop signal
        d_1
        clr    scl

;-----read
read_random_1:
        set    sda
        d_1
        set    scl
        d_1
        clr    sda                ;start signal

        clr    scl
        set    sda                ;1
        d_1
        set    scl
        d_1

        clr    scl
        clr    sda                ;0
        set    scl
        d_1

        clr    scl
        set    sda                ;1
        d_1

        set    scl
        d_1

        clr    scl
        clr    sda                ;0
        set    scl
        d_1

        clr    scl
        clr    sda                ;a2,a1,a0=0,0,0
        set    scl
        d_1

        clr    scl
        set    scl
        d_1

        clr    scl
        set    scl
        d_1

        clr    scl
```

```
    clr    sda                ;0 write mode
    set    scl
    d_1

    clr    scl
    set    sda                ;for ack
    d_1
    set    scl
    d_1
f1e1:
    sz     sda
    jmp    read_random_1
    clr    scl
    mov    a,08h
    mov    data_8,a
read_address_in:
    clr    sda
    sz     word_address.7
    set    sda
    d_1
    set    scl
    d_1
    clr    scl
    rl     word_address
    sdz    data_8
    jmp    read_address_in

    set    sda                ;for ack
    d_1
    set    scl
    d_1

skco:
    sz     sda
    jmp    skco
    clr    scl
restart:
    set    sda
    d_1
    set    scl
    d_1
    clr    sda                ;start signal

    clr    scl
    set    sda                ;1
    d_1
    set    scl
    d_1
```

```
    clr    scl
    clr    sda                ;0
    set    scl
    d_1

    clr    scl
    set    sda                ;1
    d_1
    set    scl
    d_1

    clr    scl
    clr    sda                ;0
    set    scl
    d_1

    clr    scl
    clr    sda                ;a2,a1,a0=0
    set    scl
    d_1

    clr    scl
    set    scl
    d_1

    clr    scl
    set    scl
    d_1

    clr    scl
    set    sda                ;1 read mode
    d_1
    set    scl
    d_1

    clr    scl
    set    sda                ;for ack
    d_1
    set    scl
    d_1
ewfp:
    sz     sda
    jmp    ewfp
    mov    a,08h
    mov    data_8,a
flow_out:
    clr    scl
```

```

set    sda                ;set as input port
d_1
clr    read_out.7
sz     sda
set    read_out.7
d_1
set    scl
d_1
rl     read_out
sdz    data_8
jmp    flow_out

clr    scl
clr    sda
set    scl
d_1
set    sda                ;stop signal
d_1

mov    a,055h            ;compare data read with 55H
xor    a,read_out
snz    z
jmp    fail_out
jmp    ok_end

fail_out:
jmp    $                ;operation fail
ok_end:
jmp    $                ;operation succeed

;HT49R30A-1 pass

```

- 
- Note:**
1. It is important to note the usage of A0 in the HT24 series EEPROM. For the HT24LC04 A0 is used for address.
  2. Read data occurs on the falling edge. Write data occurs on the rising edge.
  3. The HT49 series I/O ports are a simple type. Between the PCB and the HT24 series EEPROM is a parasite capacitance C. Between the PCB, the HT24 series EEPROM and the I/O is the trace equivalent resistor R. As this time constant,  $T=RC$  may cause an error on clock signal and data lines, so add a delay after every instruction set.
  4. This line is for read and write the specific address 0 for the HT24 series EEPROM. To work on other address, change the corresponding A0, A1 and A2 data from SDA.
-