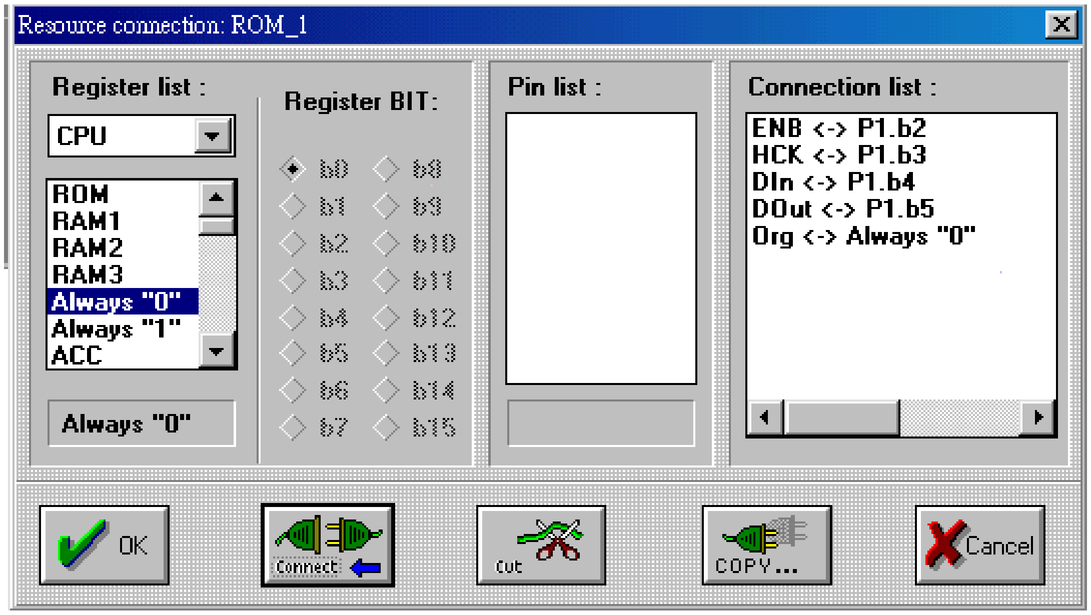
```
/*----------------------------------------------------------------------------------------------------|
| Name: 93C46.c                                                                                       |
| Purpose:                                                                                            |
| A demonstration program to illustrate how to program the 3 wire bus serial EEPROM (93C46 type).    |
| Device is first enable for write and erase (EWEN).                                                  |
| This program use 8 bit mode, ORG = 0                                                                |
|---------------------------------------------------------------------------------------------------*/
#include "config.h"

/*
    demonstrate the usage of the function of this program
*/
void main(void)
{
    unsigned char rcvdata;
    erase_write_enable();                               // enable erase/ write
    write_byte(0x31,0xaa);                              // write data to memroy
    rcvdata=read_byte(0x31);                            // read data from memory
    erase_byte(0x31);                                   // erase byte
    write_all(0xcc);                                    // write all with data
    erase_all();                                        // erase all
    write_all(0x99);                                    // write all with data
    write_byte(0x31,0xaa);                              // write data to memory
    erase_write_disable();                              // disable erase/ write

    while(1);
}
```

```
/*--------------------------------------------------------------------------------------------|
| Name: advanced.c                                                                            |
| Purpose:                                                                                    |
| The AT93C46/56/66 is enabled through the Chip Select pin (CS), and accessed via a 3-wire serial    |
| interface consisting of Data Input (DI), Data Output (DO), and Shift Clock (SK). Upon receiving a  |
| READ instruction at DI, the address is decoded and the data is clocked out serially on the data    |
| output pin DO. The WRITE cycle is completely self-timed and no separate. ERASE cycle is required   |
| before WRITE. The WRITE cycle is only enabled when the part is in the ERASE/WRITE ENABLE state.    |
| When CS is brought  “high”  following the initiation of a WRITE cycle, the DO pin outputs the      |
| READY/BUSY status of the part.                                                              |
|--------------------------------------------------------------------------------------------*/
#include "config.h"

/*
    write enable must precede all programming modes
*/
void erase_write_enable(void){
    setCS;                                      // chip select
    sendStartBit();                             // send start bit
    sendOpCode(EWEN);                           // op-code
    sendAddr(EWENADDR);                         // function code
    clrCS;                                      // chip disselect
}

/*
    disable all programming instructions
*/
void erase_write_disable(void){
    setCS;                                      // chip select
    sendStartBit();                             // send start bit
    sendOpCode(EWDS);                           // op-code
    sendAddr(EWDSADDR);                         // function code
    clrCS;                                      // chip disselect
}

/*
    erase memroy location eraddr
*/
void erase_byte(unsigned char eraddr){
    setCS;                                      // chip select
    sendStartBit();                             // send start bit
    sendOpCode(ERASE);                          // op-code
    sendAddr(eraddr);                           // functon code
    clrCS;                                      // chip disselect
    waitDone();                                 // wait process complete
}

/*
    erases all memory locations.
*/
void erase_all(void){
    setCS;                                      // chip select
    sendStartBit();                             // send start bit
    sendOpCode(ERAL);                           // op-code
    sendAddr(ERALADDR);                         // function code
    clrCS;                                      // chip disselect
    waitDone();                                 // wait process complete
}

/*
    reads data stored in memory, at specified address
*/
unsigned char read_byte(unsigned char rcvaddr){
    unsigned char mask,rcvdata;
    setCS;                                      // chip select
    sendStartBit();                             // send start bit
    sendOpCode(READ);                           // op-code
    sendAddr(rcvaddr);                          // address

    // I find this part can not use function to impletement instead, and must be written in this
    // form directly. The reason may involves the process while the function return the value to
    // read_byte or the complier problems.
    mask = 0x80;
    do{                                         // store 8 bit data
        setSK;                                  // negative edge clock data out
        if (DO==1)
          rcvdata |= mask;
        clrSK;
        mask = mask/2;
    }while (mask>0);
    clrCS;                                          // chip disselect
    return rcvdata;
}

/*
    writes memory location wraddr with byte wrdata
*/
void write_byte(unsigned char wraddr, wrdata){
    setCS;                                          // chip select
    sendStartBit();                                 // send start bit
    sendOpCode(WRITE);                              // op-code
    sendAddr(wraddr);                               // address
    sendByte(wrdata);                               // data
    clrCS;                                          // chip disselect
    waitDone();                                     // wait process complete
}
/*
    writes all memory loctions with data wrdata
*/
void write_all(unsigned char wrdata){
    setCS;                                          // chip select
    sendStartBit();                                 // send start bit
    sendOpCode(WRAL);                               // op-code
    sendAddr(WRALADDR);                             // function code
    sendByte(wrdata);                               // data
    clrCS;                                          // chip disselect
    waitDone();                                     // wait process complete
}
```

```
/*-----------------------------------------------------------------------------------------------------|
| Name: basic.c                                                                                        |
| Purpose:                                                                                             |
| The fundamental function of a specific bit pattern for the usage of the advanced.c                  |
| PS: function waitDone() is a critical function. the function may by pass because the reach of the  |
| processing time. At this conditions, user can increase the valuse of MAXPROCESSTIME defined in      |
| config.h                                                                                             |
|----------------------------------------------------------------------------------------------------*/
#include "config.h"

/*
    a start bit of all instruction
*/
void sendStartBit(void){
    clrSK;                                      // a high bit before op-code
    setDI;
    setSK;
    clrSK;
}

/*
    op-code for the instrruction
*/
void sendOpCode(bit a, bit b){
    if(a){                                      // first op-code
        setDI;
    }
    else{
        clrDI;
    }
    setSK;                                      // toggle a clock
    clrSK;

    if(b){                                      // second op-code
        setDI;
    }
    else{
        clrDI;
    }
    setSK;                                      // toggle a clock
    clrSK;
}

/*
    sends lower 7 bit of address to a serial eeprom
*/
void sendAddr(unsigned char b){
    unsigned char mask;
    mask = 0x80;
    b*=2;                                       // shift b left
    do{                                         // send 7 bits
        if ( b & mask ){
          setDI;                                // bit is high
        }
        else{
          clrDI;                                // bit is low
        }
        setSK;                                  // toggle a clock
        clrSK;
        mask = mask/2;                          // shift mask right
    }while (mask>1);
}

/*
    sends one byte of data to a serial eeprom
*/
void sendByte(unsigned char b){
    unsigned char mask;
    mask = 0x80;
    do{                                         // send 8 bits
        if ( b & mask ){
          setDI;                                // bit is high
        }
        else{
          clrDI;                                // bit is low
        }
        setSK;                                  // toggle a clock
        clrSK;
        mask = mask/2;                          // shift mask right
    }while (mask>0);
}

/*
    check the status of serial eeprom. If the eeprom is busy, DO is low. If the eeprom
    is ready, DO is high. This function will wait until the operation is completed, or
    the overflow of the MAXPROCESSTIME which limit the process time, and avoid the dead
    loop of the program. Note that if the processing cycle is too short, the busy ack may
    not catched. The limitation of the processing time can help to let the next instruction
    to continue carry on without dead loop.
*/
void waitDone(void){
    unsigned int i;
    setDO;                                      // release DO for receive the status
    setSK;                                      // a clock pulse
    clrSK;
    setCS;                                      // set the chip select
    i=0;
    while(DO&i<MAXPROCESSTIME){                 // wait until the appear of busy state
        setSK;                                  // or the overflow of the i
        clrSK;
        i++;
    }
    i=0;
    while(!DO&i<MAXPROCESSTIME){                // wait until the appear of ready state
        setSK;                                  // or the overflow of the i
        clrSK;
        i++;
    }
    clrCS;                                      // eeprom operation completed
}
```

```
/*----------------------------------------------------------------------------------------------------|
| Name: 93C46.c                                                                                        |
| Purpose:                                                                                             |
| The hardware pin setting to connect the program and target. The definition of the preprocessor for |
| easier programming and the setting of include marcus.                                                |
| PS: If the real hardware design find the advance function can not complete, users can increase the |
| value of MAXPROCESSTIME to increase the processing time.                                             |
|----------------------------------------------------------------------------------------------------*/
#include <reg51.h>
#include <stdio.h>

/* set or clr of important PIN */
#define setCS       CS=1;while (CS!=1);                 // chip select
#define clrCS       CS=0;
#define setSK       SK=1;while (SK!=1);                 // clock
#define clrSK       SK=0;
#define setDI       DI=1;while (DI!=1);                 // data in
#define clrDI       DI=0;
#define setDO       DO=1;while (DO!=1);                 // data out
#define clrDO       DO=0;

/* Op-Code */
#define READ        1,0
#define EWEN        0,0
#define ERASE       1,1
#define WRITE       0,1
#define ERAL        0,0
#define WRAL        0,0
#define EWDS        0,0

/* X8 Addr for special function */
#define EWENADDR    0x60
#define ERALADDR    0x40
#define WRALADDR    0x20
#define EWDSADDR    0x00

/* the maxiumum allowance of the processing time */
#define MAXPROCESSTIME  500

/* Pin definition */
sbit    CS = P1^2;                                      // chip select (ENB)
sbit    SK = P1^3;                                      // clock       (HCK)
sbit    DI = P1^4;                                      // data in     (DI)
sbit    DO = P1^5;                                      // data out    (Dout)

// basic.c
void sendStartBit(void);
void sendOpCode(bit a, bit b);
void sendAddr(unsigned char b);
void sendByte(unsigned char b);
void waitDone(void);

// advance.c
void erase_write_enable(void);
void erase_write_disable(void);
void erase_byte(unsigned char erAddr);
void erase_all(void);
unsigned char read_byte(unsigned char rcvaddr);
void write_byte(unsigned char wraddr, wrdata);
void write_all(unsigned char wrdata);
```

This documentation will demonstrate the usage of the umps software to implement the simulation of the 93C46 3 wire bus serial eeprom. 93C46.bin will be used in this documentation.


Resource connection: ROM_1

Register list : CPU

- ROM
- RAM1
- RAM2
- RAM3
- Always "0"
- Always "1"
- ACC

Always "0"

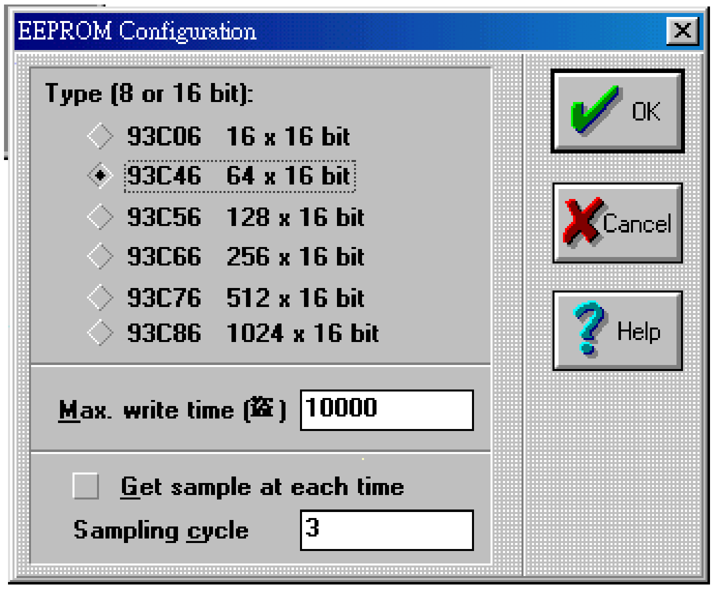
Register BIT: [✓ b0] b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15

Pin list :

Connection list :

```
ENB <-> P1.b2
HCK <-> P1.b3
DIn <-> P1.b4
DOut <-> P1.b5
Org <-> Always "0"
```

Open the UMPS software, select configure, resource, click the icon of add and select EEPROM 93C046/46/56. Click the icon of connection and configure the setting for pin connection as above.


EEPROM Configuration

Type (8 or 16 bit):

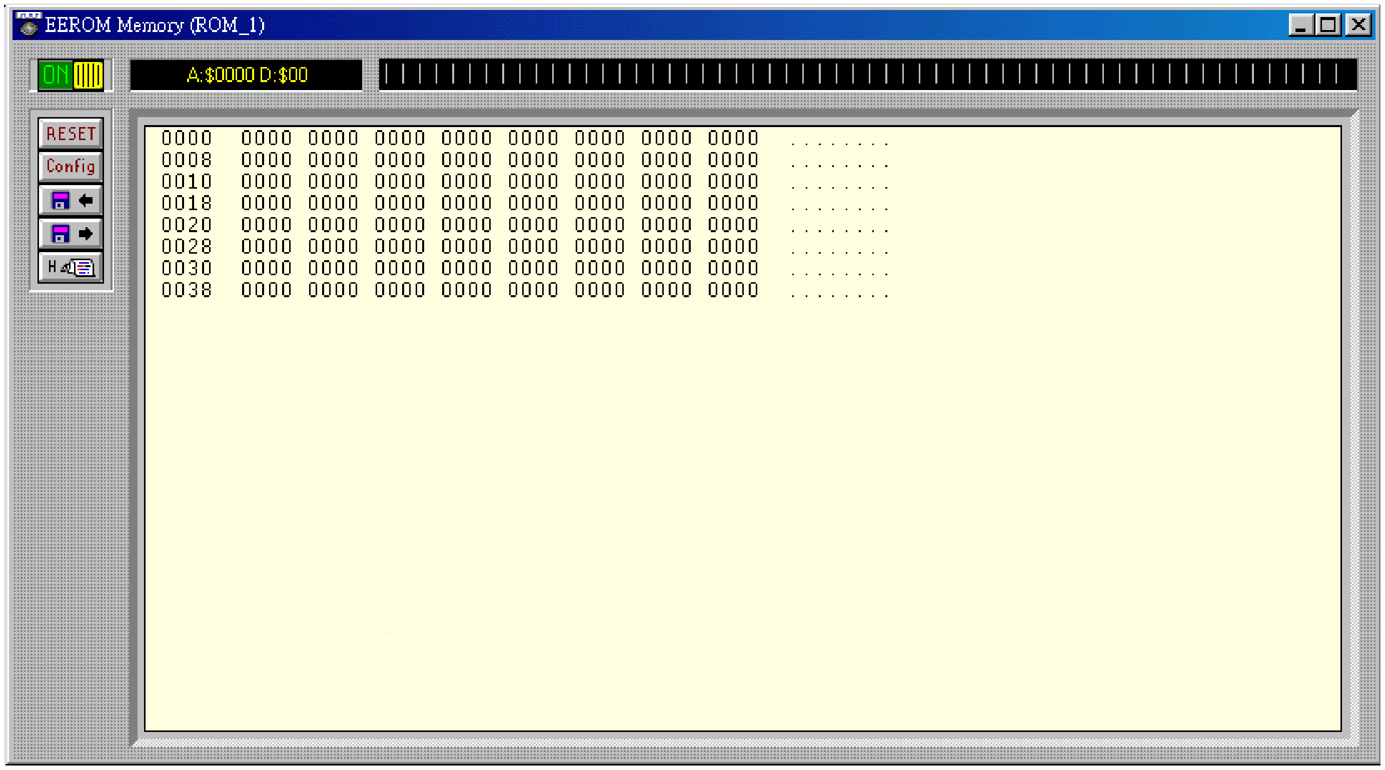
- 93C06 16 x 16 bit
- [✓ 93C46 64 x 16 bit]
- 93C56 128 x 16 bit
- 93C66 256 x 16 bit
- 93C76 512 x 16 bit
- 93C86 1024 x 16 bit

Max. write time ([illegible]) 10000

☐ Get sample at each time

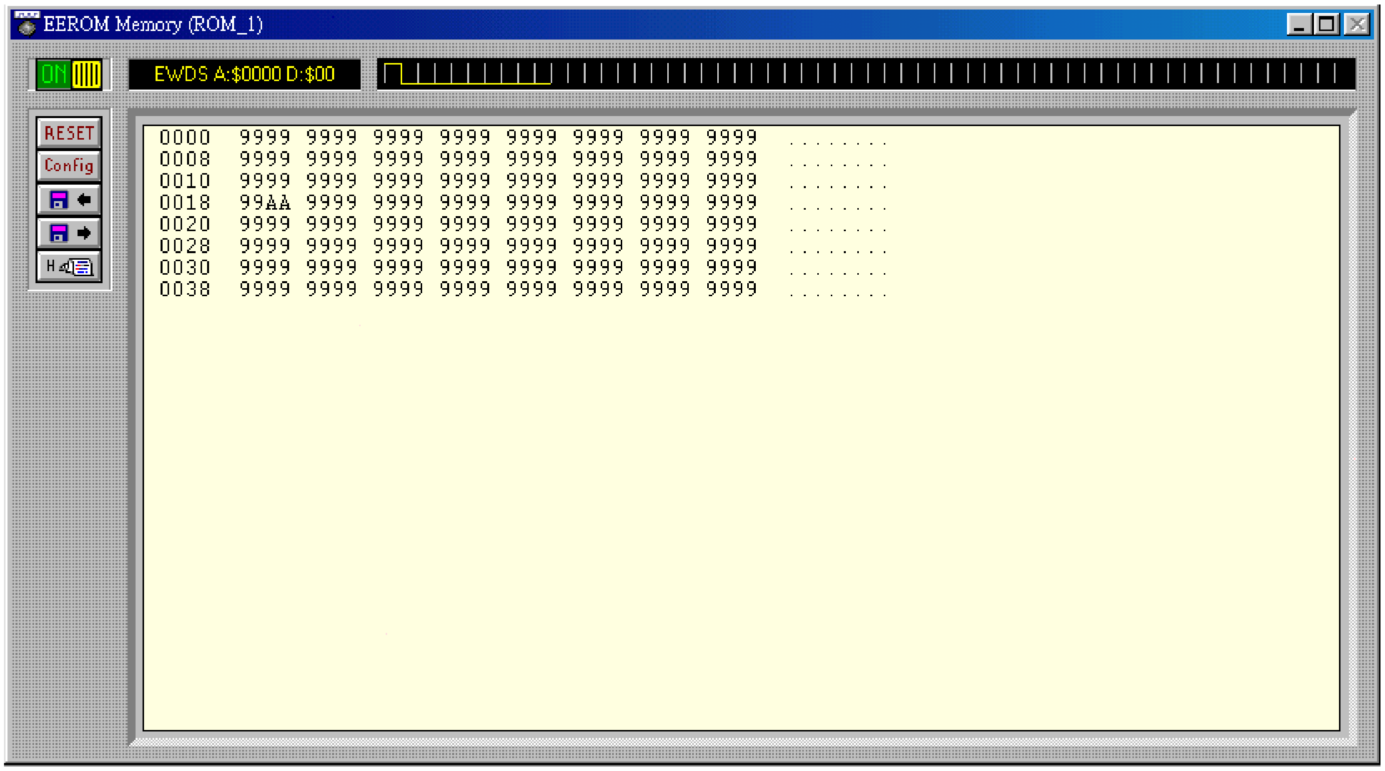
Sampling cycle 3

In resource, double click the ROM_1, select 93c46 64x16 bit.



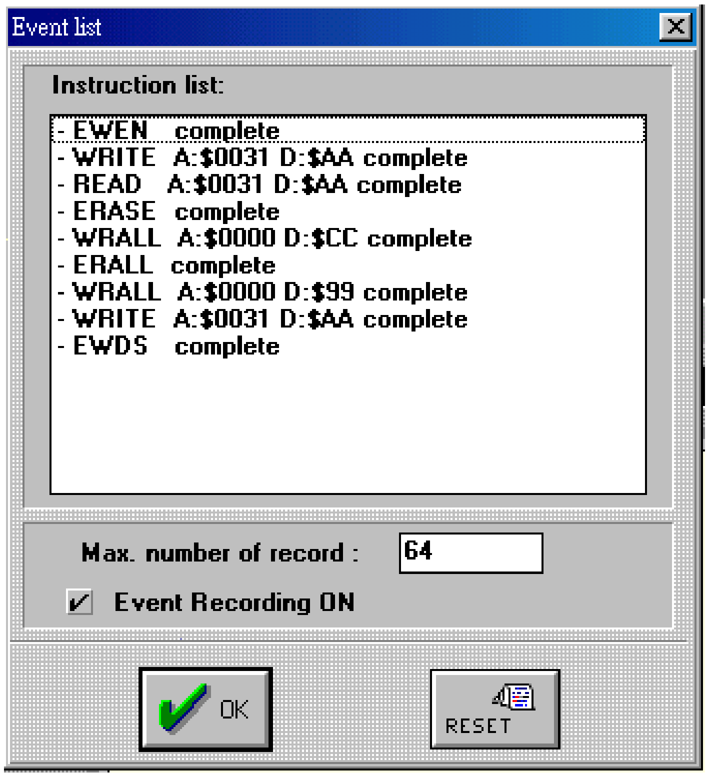
```
A:$0000 D:$00

0000   0000 0000 0000 0000 0000 0000 0000 0000   ........
0008   0000 0000 0000 0000 0000 0000 0000 0000   ........
0010   0000 0000 0000 0000 0000 0000 0000 0000   ........
0018   0000 0000 0000 0000 0000 0000 0000 0000   ........
0020   0000 0000 0000 0000 0000 0000 0000 0000   ........
0028   0000 0000 0000 0000 0000 0000 0000 0000   ........
0030   0000 0000 0000 0000 0000 0000 0000 0000   ........
0038   0000 0000 0000 0000 0000 0000 0000 0000   ........
```

Exit the resource, double click the device of ROM_1, this will open the eeprom memory area.



```
EWDS A:$0000 D:$00

0000   9999 9999 9999 9999 9999 9999 9999 9999   ........
0008   9999 9999 9999 9999 9999 9999 9999 9999   ........
0010   9999 9999 9999 9999 9999 9999 9999 9999   ........
0018   99AA 9999 9999 9999 9999 9999 9999 9999   ........
0020   9999 9999 9999 9999 9999 9999 9999 9999   ........
0028   9999 9999 9999 9999 9999 9999 9999 9999   ........
0030   9999 9999 9999 9999 9999 9999 9999 9999   ........
0038   9999 9999 9999 9999 9999 9999 9999 9999   ........
```

Load the bin file 93c46.bin. Press reset icon and then press run icon, the result should as above.



```
- EWEN   complete
- WRITE  A:$0031 D:$AA complete
- READ   A:$0031 D:$AA complete
- ERASE  complete
- WRALL  A:$0000 D:$CC complete
- ERALL  complete
- WRALL  A:$0000 D:$99 complete
- WRITE  A:$0031 D:$AA complete
- EWDS   complete
```

Max. number of record : 64

✓ Event Recording ON

Press the event list icon located in eeprom memory area left side.

Serial EEPROM/RAM (93C46 type)

Refresh style:
- on refresh breakpoint,
- at each N CPU cycles CPU (option *Get Sample at each time* checked) .

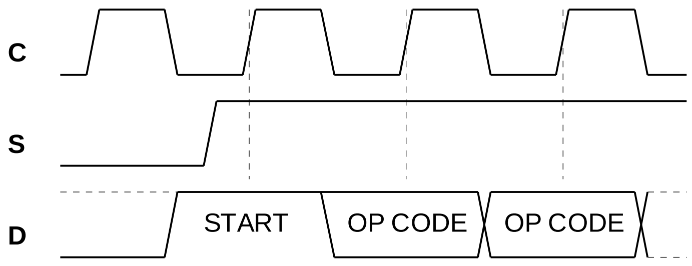
See also: Xicor EEPROM
Resources
Resource mechanism detail

93C46 can be configured to have 8 or 16 bits accesses with the ORG pin:
- ORG = 0 , 8 Bits mode
- ORG = 1 , 16 Bits mode

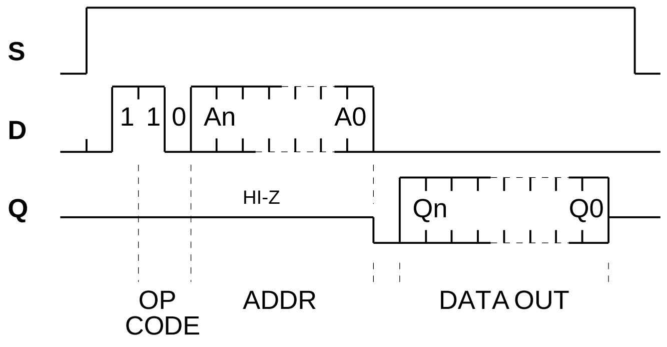
93C46 EEPROM has 7 instructions.

Instruction	**Description**	**Op-Code**	**X8 Addr (Org=0)**	**DATA**	**X16 Addr (Org=1)**	**Data**
READ	Read Data from memory	1 0	A6-A0	Q7-Q0	A5-A0	Q15-Q0
WRITE	Write Data to memory	0 1	A6-A0	Q7-Q0	A5-A0	Q15-Q0
EWEN	Erase/Write ENABLE	0 0	11XXXXX		11XXXX	
EWDS	Erase/Write DISABLE	0 0	00XXXXX		00XXXX	
ERASE	Erase Byte or Word	1 1	A6-A0		A5-A0	
ERAL	Erase ALL	0 0	10XXXXX		10XXXX	
WRAL	Write ALL with data	0 0	01XXXXX	D7-D0	01XXXX	D7-D0

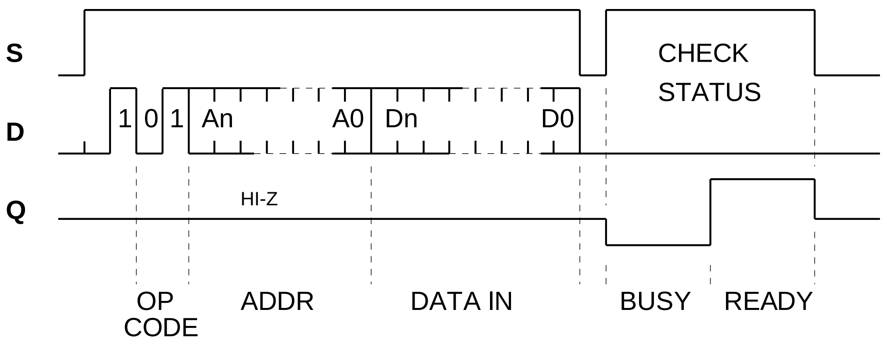
Start Conditions



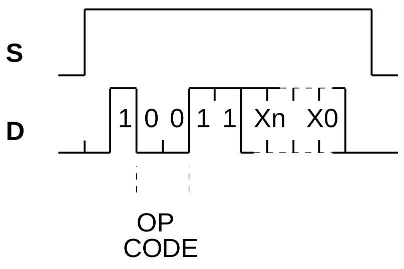
Read



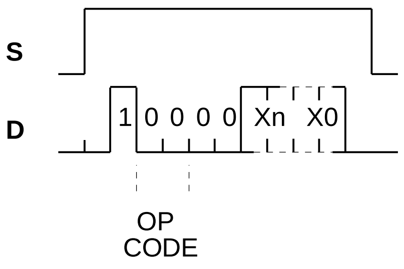
Write



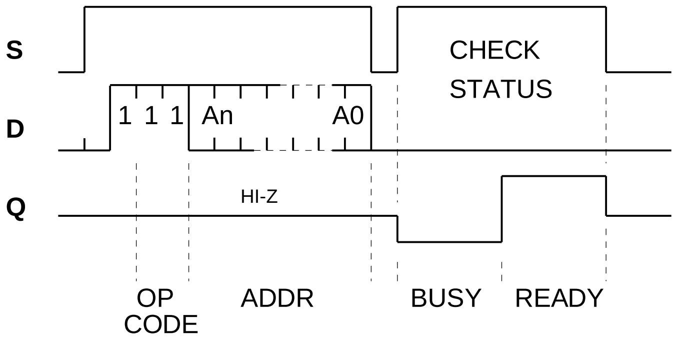
Erase Write Enable



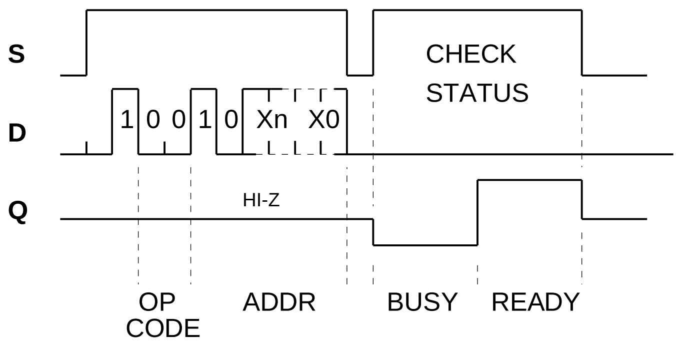
Erase Write Disable



Erase



Erase All


S
CHECK
STATUS
D
1 0 0 1 0 Xn X0
Q
HI-Z
OP
CODE
ADDR
BUSY
READY

Features

- Low-voltage and Standard-voltage Operation
  - 2.7 (V_{CC} = 2.7V to 5.5V)
  - 2.5 (V_{CC} = 2.5V to 5.5V)
  - 1.8 (V_{CC} = 1.8V to 5.5V)
- User-selectable Internal Organization
  - 1K: 128 x 8 or 64 x 16
  - 2K: 256 x 8 or 128 x 16
  - 4K: 512 x 8 or 256 x 16
- 3-wire Serial Interface
- 2 MHz Clock Rate (5V)
- Self-timed Write Cycle (10 ms max)
- High Reliability
  - Endurance: 1 Million Write Cycles
  - Data Retention: 100 Years
- Automotive Grade, Extended Temperature and Lead-Free Devices Available
- 8-lead PDIP, 8-lead JEDEC SOIC, 8-lead EIAJ SOIC, 8-lead MAP and 8-lead TSSOP Packages

3-wire Serial EEPROMs

1K (128 x 8 or 64 x 16)

2K (256 x 8 or 128 x 16)

4K (512 x 8 or 256 x 16)

AT93C46
AT93C56
AT93C66

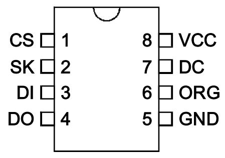
Description

The AT93C46/56/66 provides 1024/2048/4096 bits of serial electrically erasable programmable read only memory (EEPROM) organized as 64/128/256 words of 16 bits each, when the ORG pin is connected to VCC and 128/256/512 words of 8 bits each when it is tied to ground. The device is optimized for use in many industrial and commercial applications where low power and low voltage operations are essential. The AT93C46/56/66 is available in space-saving 8-lead PDIP, 8-lead JEDEC SOIC, 8-lead EIAJ SOIC, 8-lead MAP and 8-lead TSSOP packages. *(continued)*

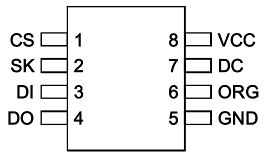
Pin Configurations

Pin Name	Function
CS	Chip Select
SK	Serial Data Clock
DI	Serial Data Input
DO	Serial Data Output
GND	Ground
VCC	Power Supply
ORG	Internal Organization
DC	Don't Connect

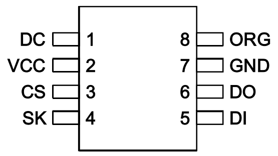
8-lead PDIP



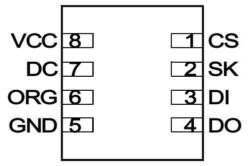
8-lead SOIC



8-lead SOIC
Rotated (R)
(1K JEDEC Only)

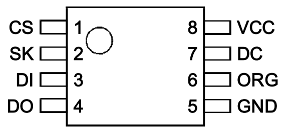


8-lead MAP



Bottom View

8-lead TSSOP



Rev. 0172T–SEEPR–6/03

Instruction Set for the AT93C46

Instruction	SB	Op Code	Address		Data		Comments
			x 8	x 16	x 8	x 16	
READ	1	10	A_6 - A_0	A_5 - A_0			Reads data stored in memory, at specified address.
EWEN	1	00	11XXXXX	11XXXX			Write enable must precede all programming modes.
ERASE	1	11	A_6 - A_0	A_5 - A_0			Erase memory location A_n - A_0.
WRITE	1	01	A_6 - A_0	A_5 - A_0	D_7 - D_0	D_{15} - D_0	Writes memory location A_n - A_0.
ERAL	1	00	10XXXXX	10XXXX			Erases all memory locations. Valid only at V_{CC} = 4.5V to 5.5V.
WRAL	1	00	01XXXXX	01XXXX	D_7 - D_0	D_{15} - D_0	Writes all memory locations. Valid only at V_{CC} = 4.5V to 5.5V.
EWDS	1	00	00XXXXX	00XXXX			Disables all programming instructions.

Note: The X's in the address field represent don't care values and must be clocked.

Instruction Set for the AT93C56 and AT93C66

Instruction	SB	Op Code	Address		Data		Comments
			x 8	x 16	x 8	x 16	
READ	1	10	A_8 - A_0	A_7 - A_0			Reads data stored in memory, at specified address.
EWEN	1	00	11XXXXXXX	11XXXXXX			Write enable must precede all programming modes.
ERASE	1	11	A_8 - A_0	A_7 - A_0			Erase memory location A_n - A_0.
WRITE	1	01	A_8 - A_0	A_7 - A_0	D_7 - D_0	D_{15} - D_0	Writes memory location A_n - A_0.
ERAL	1	00	10XXXXXXX	10XXXXXX			Erases all memory locations. Valid only at V_{CC} = 4.5V to 5.5V.
WRAL	1	00	01XXXXXXX	01XXXXXX	D_7 - D_0	D_{15} - D_0	Writes all memory locations. Valid only at V_{CC} = 5.0V ±10% and Disable Register cleared.
EWDS	1	00	00XXXXXXX	00XXXXXX			Disables all programming instructions.

Note: The X's in the address field represent don't care values and must be clocked.

Functional Description

The AT93C46/56/66 is accessed via a simple and versatile 3-wire serial communication interface. Device operation is controlled by seven instructions issued by the host processor. **A valid instruction starts with a rising edge of CS** and consists of a Start Bit (logic "1") followed by the appropriate Op Code and the desired memory Address location.

READ (READ): The Read (READ) instruction contains the Address code for the memory location to be read. After the instruction and address are decoded, data from the selected memory location is available at the serial output pin DO. Output data changes are synchronized with the rising edges of serial clock SK. It should be noted that a dummy bit (logic "0") precedes the 8- or 16-bit data output string.

ERASE/WRITE (EWEN): To assure data integrity, the part automatically goes into the Erase/Write Disable (EWDS) state when power is first applied. An Erase/Write Enable (EWEN) instruction must be executed first before any programming instructions can be carried out. Please note that once in the Erase/Write Enable state, programming remains enabled until an Erase/Write Disable (EWDS) instruction is executed or V_{CC} power is removed from the part.

ERASE (ERASE): The Erase (ERASE) instruction programs all bits in the specified memory location to the logical "1" state. The self-timed erase cycle starts once the ERASE instruction and address are decoded. The DO pin outputs the READY/BUSY status of the part if CS is brought high after being kept low for a minimum of 250 ns (t_{CS}). A logic "1" at pin DO indicates that the selected memory location has been erased, and the part is ready for another instruction.

WRITE (WRITE): The Write (WRITE) instruction contains the 8 or 16 bits of data to be written into the specified memory location. The self-timed programming cycle, t_{WP}, starts after the last bit of data is received at serial data input pin DI. The DO pin outputs the READY/BUSY status of the part if CS is brought high after being kept low for a minimum of 250 ns (t_{CS}). A logic "0" at DO indicates that programming is still in progress. A logic "1" indicates that the memory location at the specified address has been written with the data pattern contained in the instruction and the part is ready for further instructions. **A READY/BUSY status cannot be obtained if the CS is brought high after the end of the self-timed programming cycle, t_{WP}.**

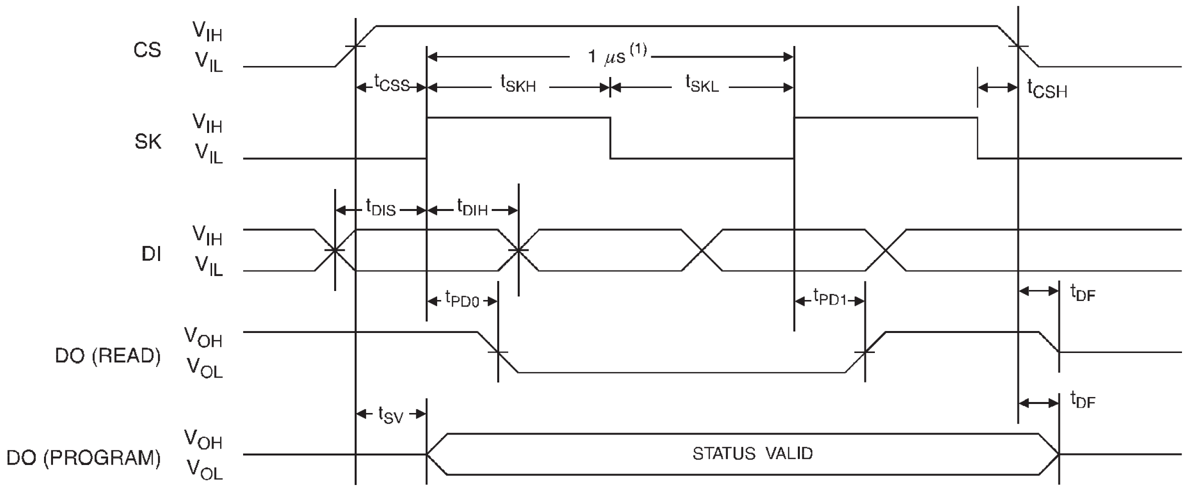
ERASE ALL (ERAL): The Erase All (ERAL) instruction programs every bit in the memory array to the logic "1" state and is primarily used for testing purposes. The DO pin outputs the READY/BUSY status of the part if CS is brought high after being kept low for a minimum of 250 ns (t_{CS}). The ERAL instruction is valid only at V_{CC} = 5.0V ± 10%.

WRITE ALL (WRAL): The Write All (WRAL) instruction programs all memory locations with the data patterns specified in the instruction. The DO pin outputs the READY/BUSY status of the part if CS is brought high after being kept low for a minimum of 250 ns (t_{CS}). The WRAL instruction is valid only at V_{CC} = 5.0V ± 10%.

ERASE/WRITE DISABLE (EWDS): To protect against accidental data disturb, the Erase/Write Disable (EWDS) instruction disables all programming modes and should be executed after all programming operations. The operation of the READ instruction is independent of both the EWEN and EWDS instructions and can be executed at any time.

Timing Diagrams

Synchronous Data Timing



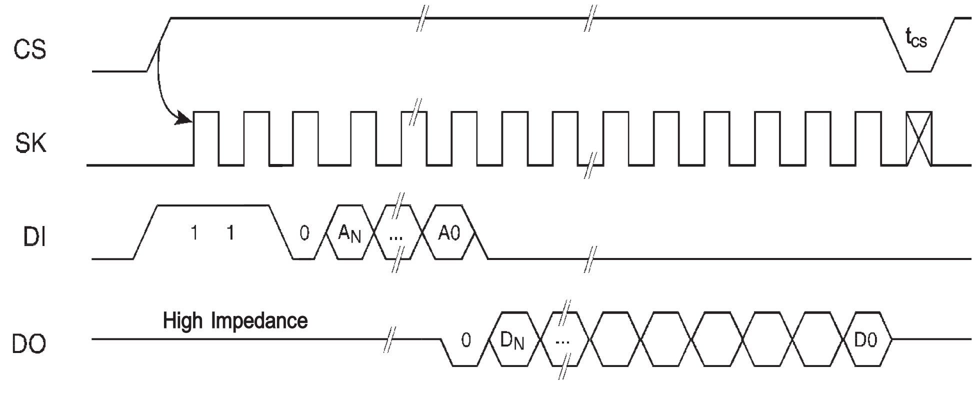
Note: 1. This is the minimum SK period.

Organization Key for Timing Diagrams

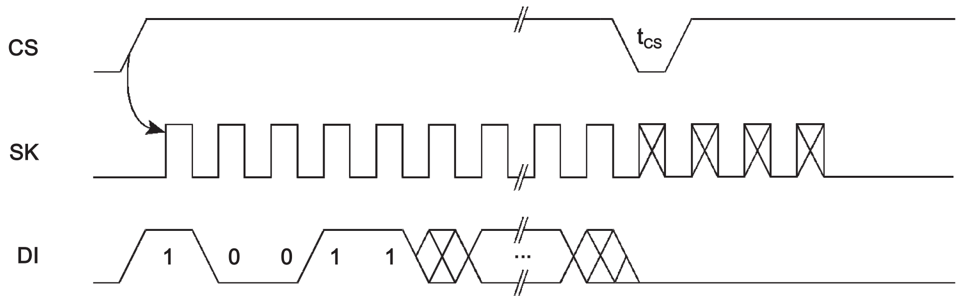
I/O	AT93C46 (1K) x 8	AT93C46 (1K) x 16	AT93C56 (2K) x 8	AT93C56 (2K) x 16	AT93C66 (4K) x 8	AT93C66 (4K) x 16
A_N	A_6	A_5	A_8(1)	A_7(2)	A_8	A_7
D_N	D_7	D_{15}	D_7	D_{15}	D_7	D_{15}

Notes: 1. A_8 is a DON'T CARE value, but the extra clock is required.
2. A_7 is a DON'T CARE value, but the extra clock is required.

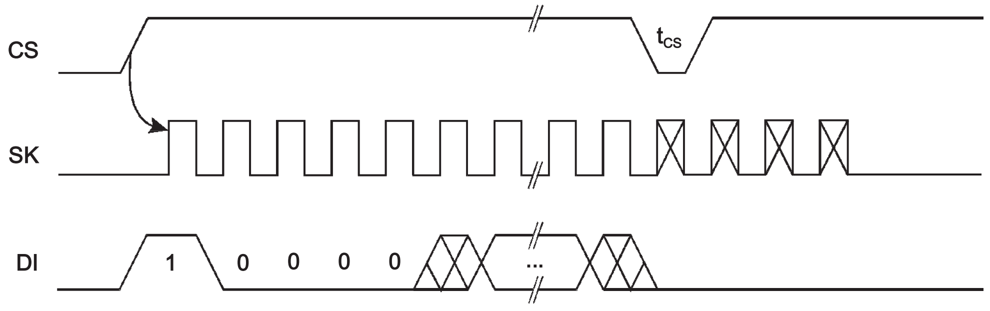
READ Timing



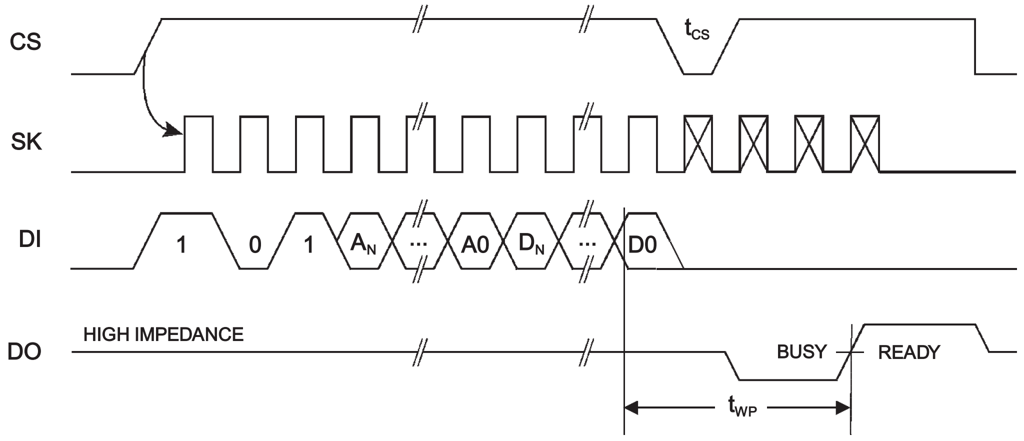
EWEN Timing



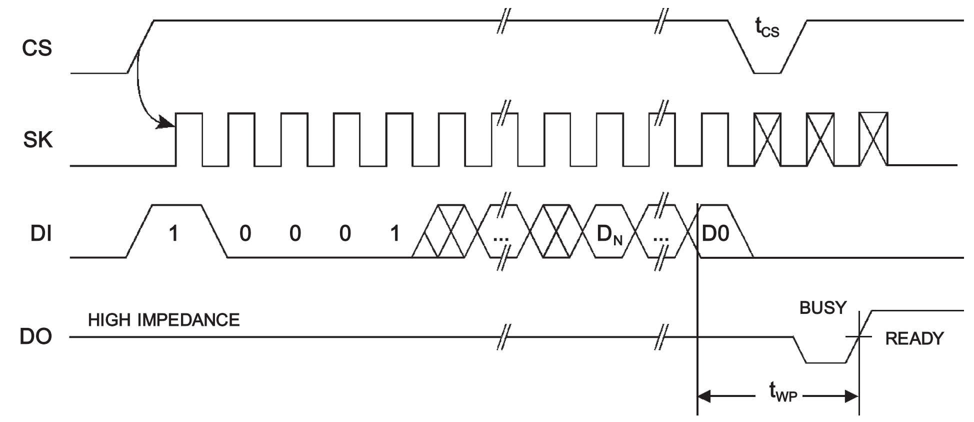
EWDS Timing



WRITE Timing

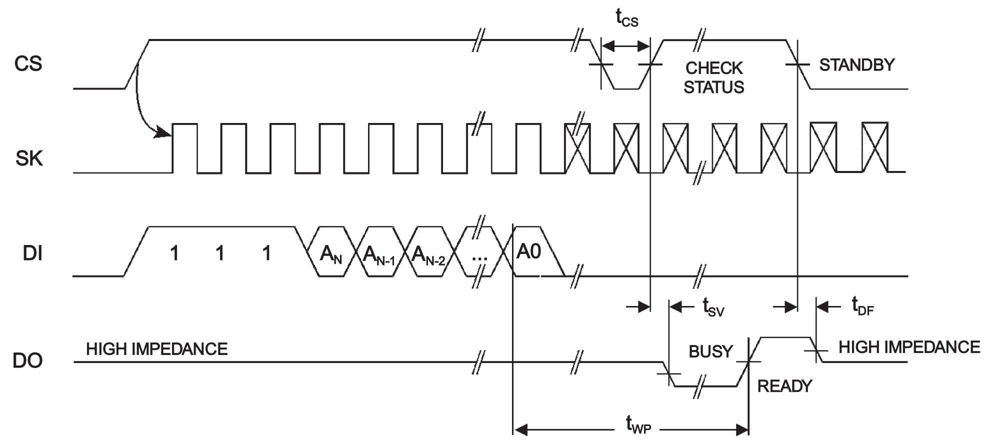


WRAL Timing(1)

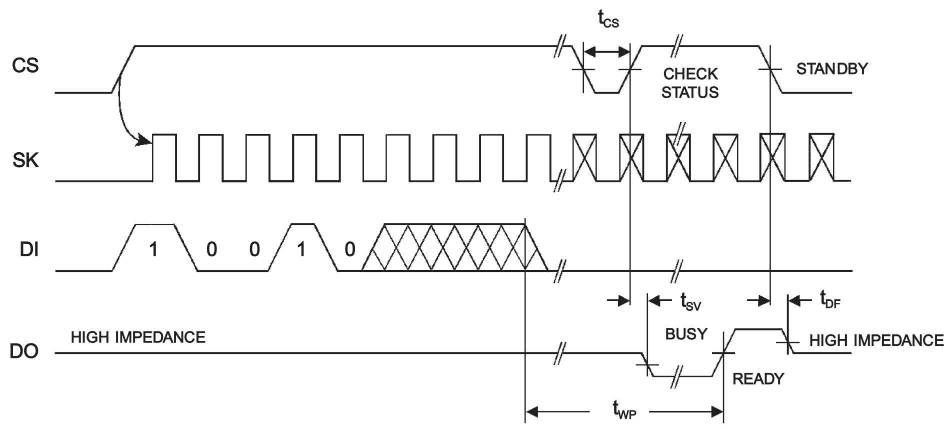


Note: 1. Valid only at V_{CC} = 4.5V to 5.5V.

ERASE Timing



ERAL Timing(1)



Note: 1. Valid only at V_{CC} = 4.5V to 5.5V.