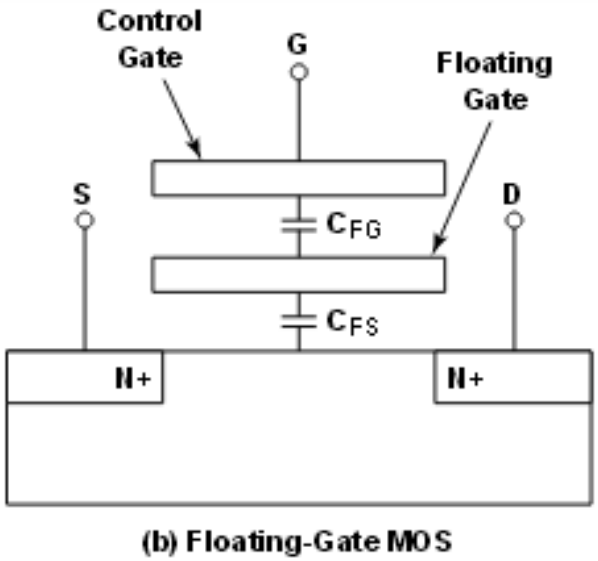
EEPROM Lesen/Schreiben über SPI-Bus

Experiment EEPROMtest 6

A.Schultze / DK4AQ 15.06.2013

Was ist ein EEPROM ? EEPROM = Electrical Erasable Programmable Read Only Memory

Ein EEPROM kann elektrisch geschrieben werden (programmiert) und hält seine Daten auch wenn keine Versorgungsspannung mehr anliegt (Datenspeicherung über ca. 200 Jahre !). In sofern verhält es sich wie ein Read-Only-Memory. Es kann jedoch eletrisch gelöscht werden und wieder beschrieben werden. Es sind ca. 1.000.000-mal beschrieben und gelöscht werden.



(b) Floating-Gate MOS

Quelle: http://web.eecs.umich.edu/~prabal/teaching/eecs373-f10/readings/rom-eprom-eeprom-technology.pdf

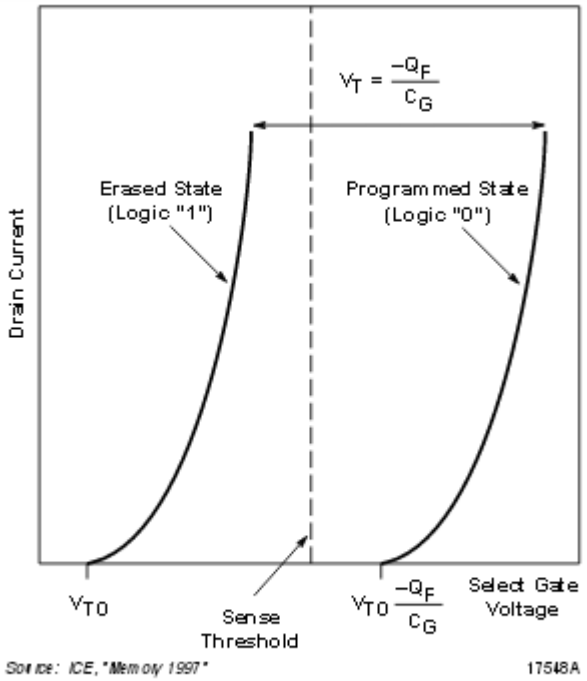


Figure 9-6. Electrical Characteristics of an EPROM

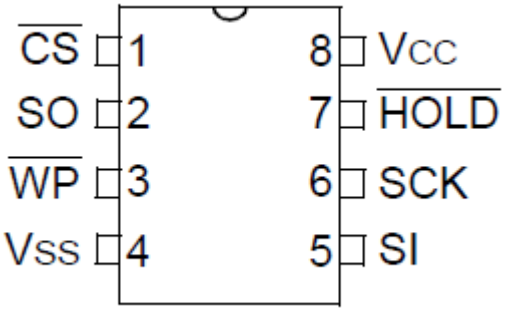
Das Prinzip einer EEPROM-Zelle beruht auf einem selbstsperrenden MOSFET-Feldeffekttransistor. In diesen Transistor ist eine zweite Elektrode eingebaut, die in beide Richtungen sehr gut isoliert ist. Ist dort eine Ladung vorhanden (Elektronen, so wirkt sie auf den sperrend auf den Kanal. Soll der Transistor nun geöffnet werden, so muss am Gate mehr Spannung angelegt werden (siehe Bild Kennlinienverschiebung). Der Transistor kann also nur mit mehr Spannung geöffnet werden. Diese Tatsache bedeutet, es war Ladung gespeichert und wird als logische „1 “ gewertet. Allerdings wird die Ladung auf der Zusatzelektrode dabei verändert. Es muss also nach Feststellen der „1“ der alte Zustand aktiv wieder hergestellt werden. Die Weise auf die die Ladung auf die Zusatzelektrode kommt ist nach Verfahren und Hersteller unterschiedlich. Ein Verfahren beruht auf den Eigenschaften des Isolationsmaterials zwischen Channel und Elektrode. Es wird z.B. ein Material benutzt, welches sog. Tunnel-Eigeneschaften hat, d.h. Bei Überschreiten einer Potentialdifferenz lässt das Material Ladungsträger aus dem Channel in die Zusatzelektrode durch.

Kern der Eigenschaften ist die Tatsache, dass auf der isolierten Zwischenelektrode Ladngsträger sehr lange erhalten bleiben. Die Ladungsverhältnisse um die Elektrode herum erlauben nicht das Entweichen, die Elektronen sind sozusagen „eingesperrt“.

Diese Vorgänge können wir von aussen nicht beobachten. Wir legen Daten an und lösen einen Schreibvorgang aus oder übergeben eine Leseanforderung und holen Daten ab. Alles andere läuft intern im EEPROM ab.

Quelle: Wikipedia http://common.ziffdavisinternet.com/encyclopedia_images/EEPROM.GIF

Serielles EEPROM 1kbit Microchip 25LC010A



Pin Function Table

Name	Function
$\overline{CS}$	Chip Select Input
SO	Serial Data Output
$\overline{WP}$	Write-Protect
Vss	Ground
SI	Serial Data Input
SCK	Serial Clock Input
$\overline{HOLD}$	Hold Input
Vcc	Supply Voltage

Features:

- 10 MHz max. clock frequency
- Low-power CMOS technology:
  - Max. Write Current: 5 mA at 5.5V, 10 MHz
  - Read Current: 5 mA at 5.5V, 10 MHz
  - Standby Current: 5 µA at 5.5V

128 x 8-bit organization

Write Page mode (up to 16 bytes)

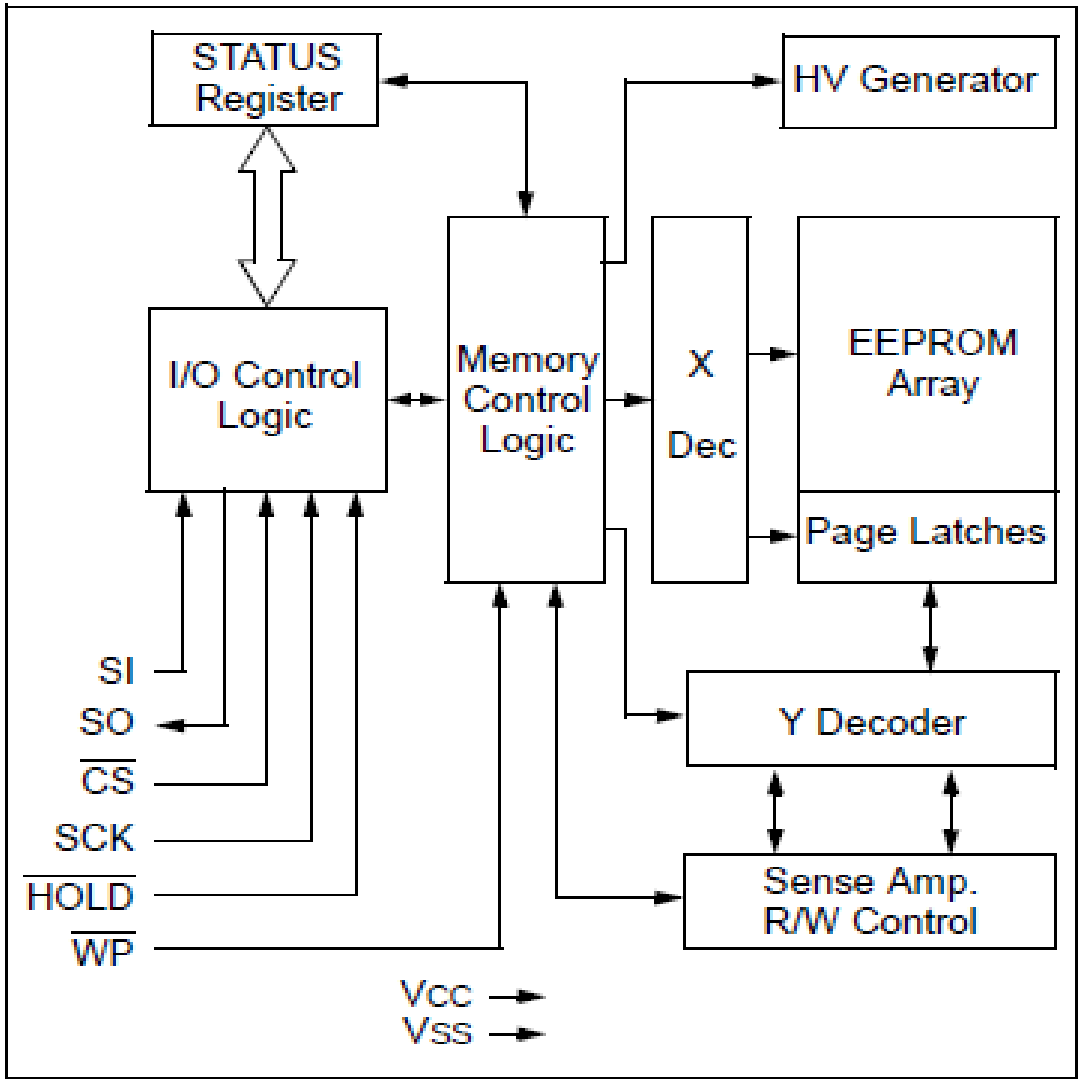
Sequential Read

Self-timed Erase and Write cycles (5 ms max.)

Param. No.	Sym.	Characteristic	Min.	Max.	Units	Test Conditions
D001	V_{IH1}	High-level Input Voltage	0.7 V_{CC}	V_{CC}+1	V	
D002	V_{IL1}	Low-level Input Voltage	-0.3	0.3 V_{CC}	V	$V_{CC} \geq 2.7V$ **(Note)**
D003	V_{IL2}		-0.3	0.2 V_{CC}	V	$V_{CC} < 2.7V$ **(Note)**
D004	V_{OL}	Low-level Output Voltage	—	0.4	V	I_{OL} = 2.1 mA
D005	V_{OL}		—	0.2	V	I_{OL} = 1.0 mA, $V_{CC} < 2.5V$
D006	V_{OH}	High-level Output Voltage	V_{CC} -0.5	—	V	I_{OH} = -400 µA

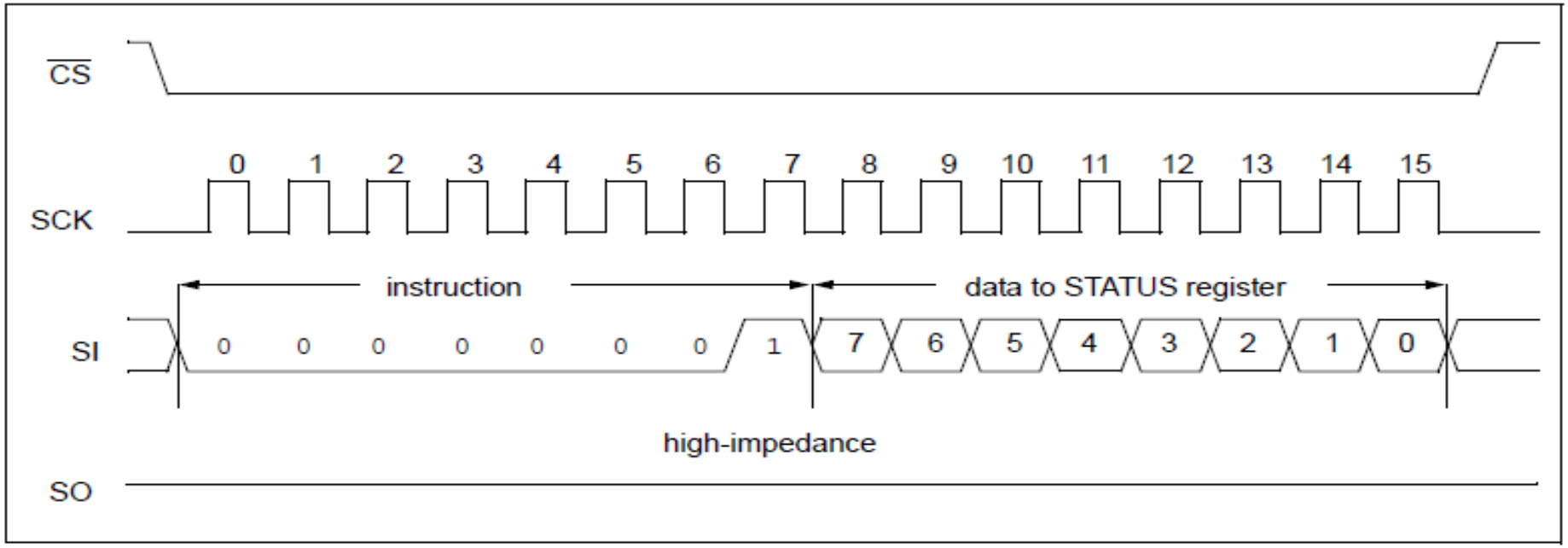
AC CHARACTERISTICS			Industrial (I): T_A = -40°C to +85°C V_{CC} = 1.8V to 5.5V Automotive (E): T_A = -40°C to +125°C V_{CC} = 2.5V to 5.5V			
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Max.**	**Units**	**Test Conditions**
1	F_{CLK}	Clock Frequency	—	10	MHz	$4.5V \leq V_{CC} < 5.5V$
			—	5	MHz	$2.5V \leq V_{CC} < 4.5V$
			—	3	MHz	$1.8V \leq V_{CC} < 2.5V$

BLOCK DIAGRAM



- Kommandos an den Speicher werden in jedem Telegramm an das Status-/Command-Register übertragen
- Adressen für die Operation werden in jedem Telegramm gesendet und von der Memory Control-Logic über X- und Y-Decoder an eine Matrix von 1024 Bit-Speicherzellen angelegt.
- Bei jedem Lesezyklus werden Daten aus dem Status-Register mit übertragen.

FIGURE 2-7: WRITE STATUS REGISTER TIMING SEQUENCE (WRSR)



Note: An internal write cycle (TWC) is initiated on the rising edge of $\overline{CS}$ after a valid write STATUS register sequence.

TABLE 2-2: STATUS REGISTER

7	6	5	4	3	2	1	0
–	–	–	–	W/R	W/R	R	R
X	X	X	X	BP1	BP0	WEL	WIP
W/R = writable/readable. R = read-only.							

BP1/BP0: Block Protection Mode, Beschreibbar

TABLE 2-3: ARRAY PROTECTION

BP1	BP0	Array Addresses Write-Protected
0	0	none
0	1	upper 1/4 (60h-7Fh)
1	0	upper 1/2 (40h-7Fh)
1	1	all (00h-7Fh)

FIGURE 2-6: READ STATUS REGISTER TIMING SEQUENCE (RDSR)

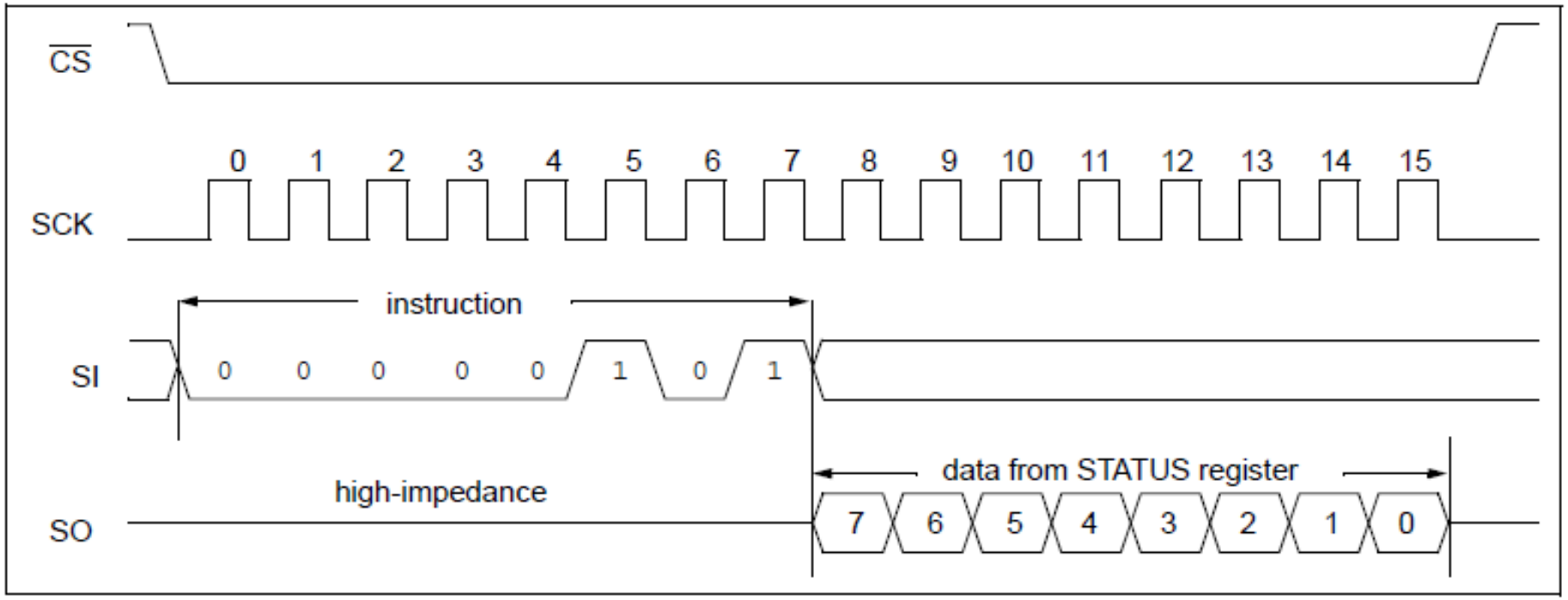


TABLE 2-2: STATUS REGISTER

7	6	5	4	3	2	1	0
–	–	–	–	W/R	W/R	R	R
X	X	X	X	BP1	BP0	WEL	WIP
W/R = writable/readable. R = read-only.							

WEL: Write Enable Latch (1: Schreiben möglich, 0: Schreibgesperrt), Read-Olny
WIP: Write in Process: Interner Schreibprozess läuft. Read-Only
BP1/BP0: Block Protection Mode, Lesbar

TABLE 2-1: INSTRUCTION SET

Instruction Name	Instruction Format	Description
READ	0000 x011	Read data from memory array beginning at selected address
WRITE	0000 x010	Write data to memory array beginning at selected address
WRDI	0000 x100	Reset the write enable latch (disable write operations)
WREN	0000 x110	Set the write enable latch (enable write operations)
RDSR	0000 x101	Read STATUS register
WRSR	0000 x001	Write STATUS register

x = don't care

Im Instruction Byte wird vorgegeben, was das EEPROM tun soll.

FIGURE 2-1: READ SEQUENCE

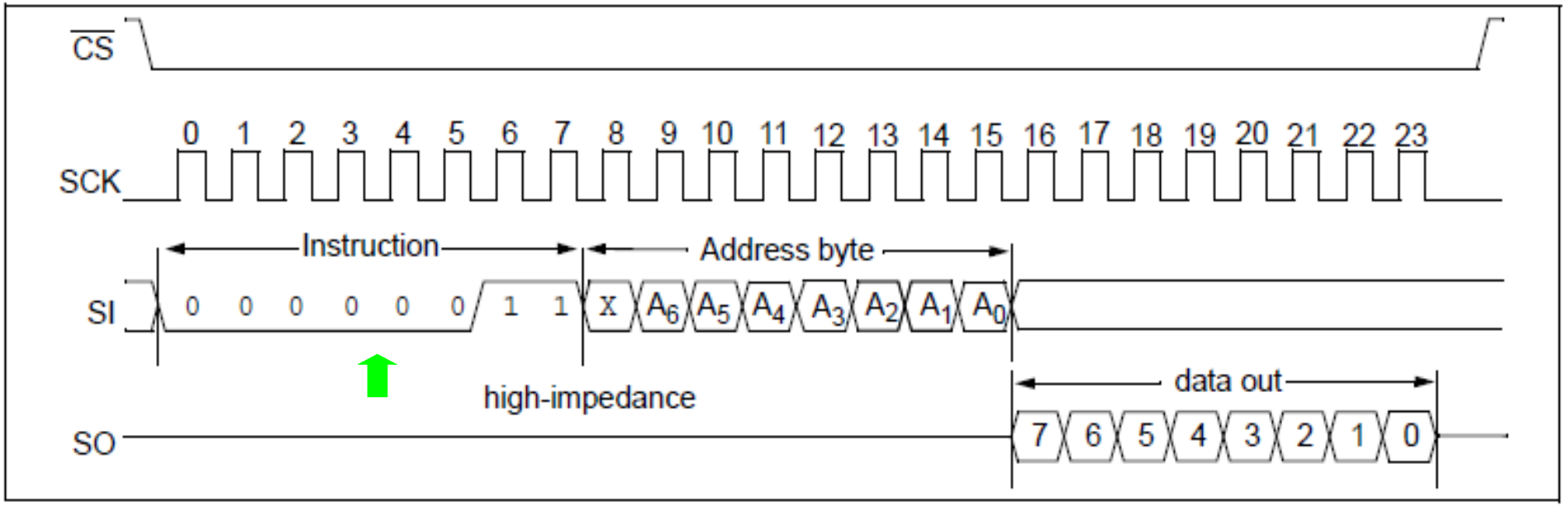


FIGURE 2-2: BYTE WRITE SEQUENCE

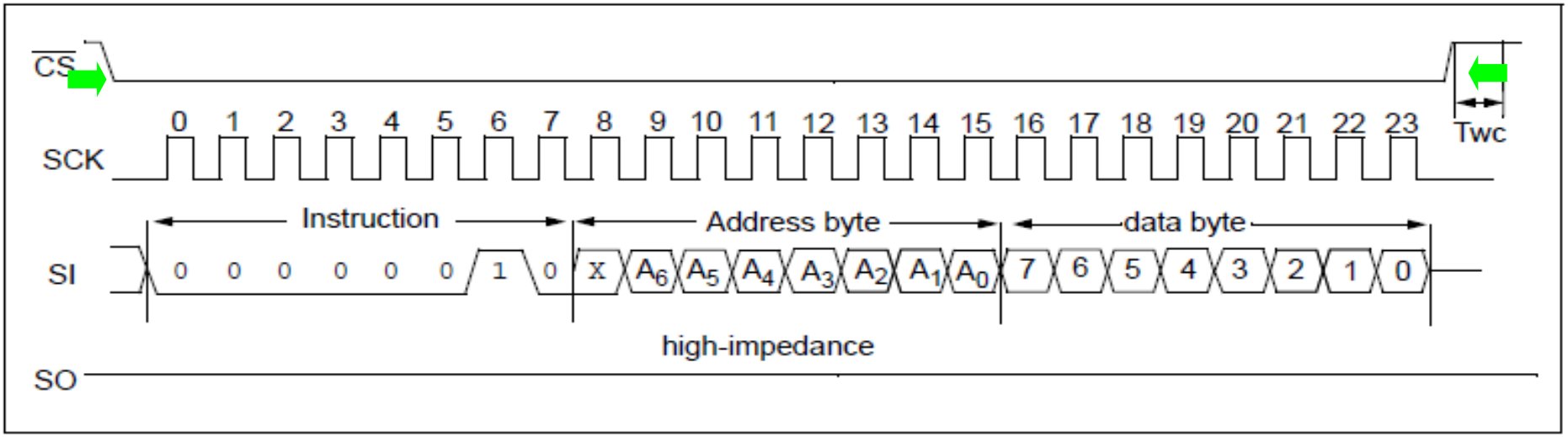
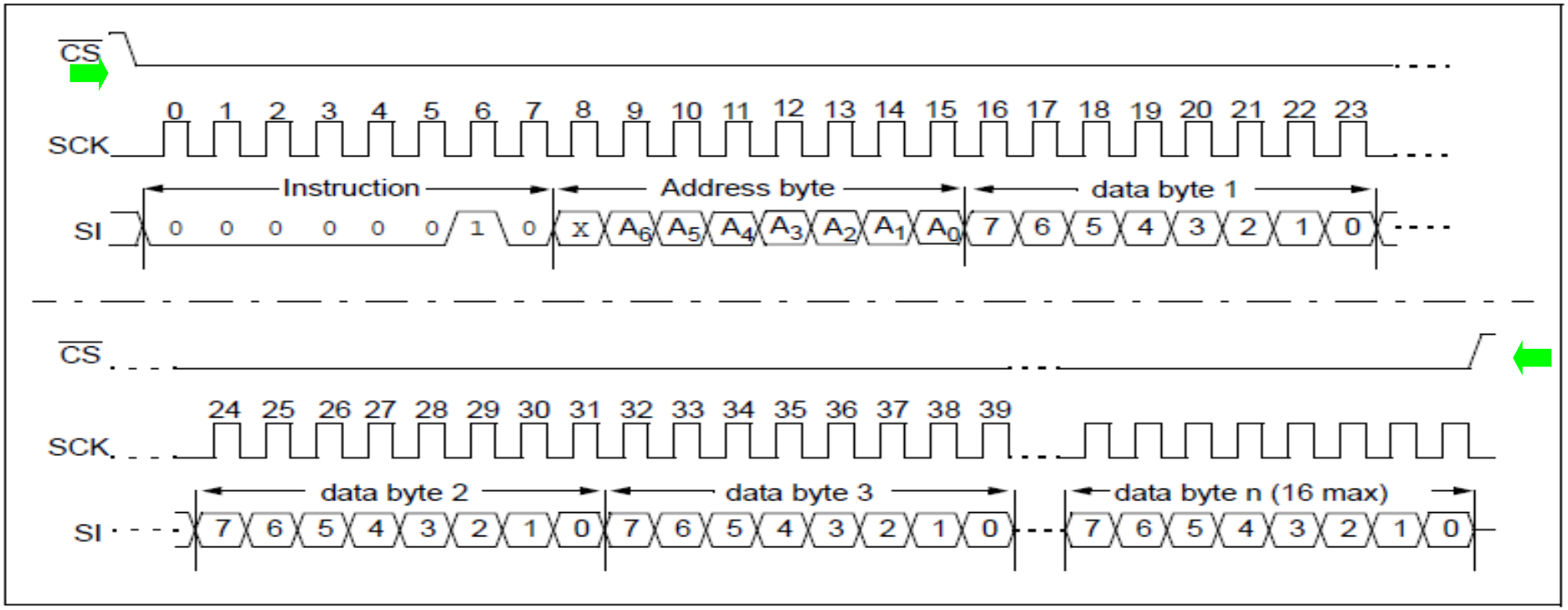


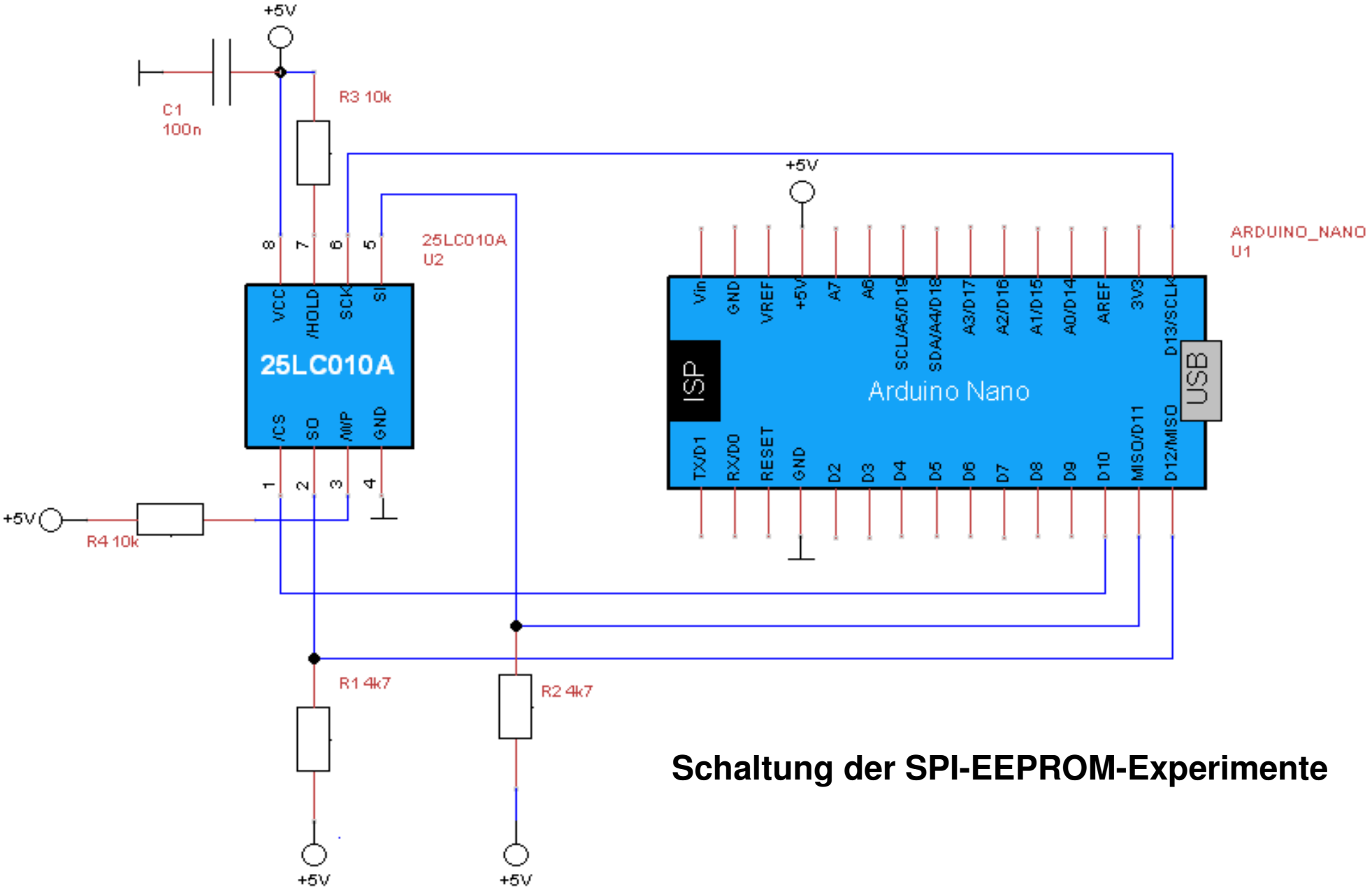
FIGURE 2-3: PAGE WRITE SEQUENCE



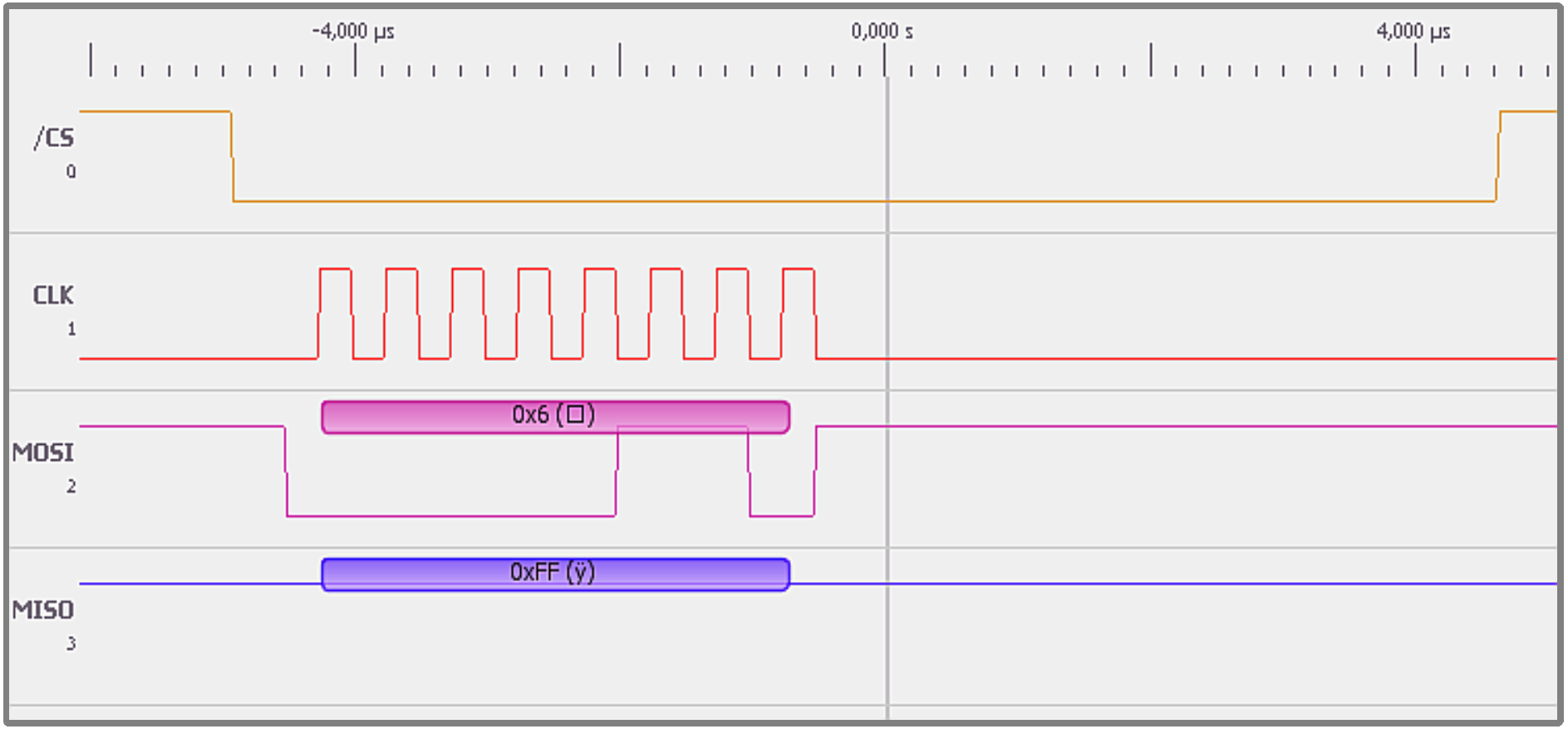
Adressierung der Speicherzellen und die Page-Einteilung

- Schreib-/Lese-Zugriff ist auf einzelne beliebige Adressen möglich
- Lese-Zugriff ist auf beliebige Adressen in beliebiger Länge möglich
- Schreibzugriffe mit mehreren Bytes Länge können ist nur innerhalb der Pages durchgeführt werden !
- Wird versucht beim Schreiben mehrerer Bytes ausserhalb der adressierten Page zu schreiben, so beginnt der Adresspointer wieder beim Page-Anfang !.

	Adresse (dez)	Adresse (hex)	Daten (hex) z.B.
	127	7F	90
	...	...	...
Page 1	011	0B	43
	010	0A	42
	009	09	41
	008	08	33
	007	07	32
	006	06	31
Page 0	005	05	AA
	004	04	3E
	003	03	55
	002	02	B2
	001	01	33
	000	00	1A



Schaltung der SPI-EEPROM-Experimente



Command Write Enable (06h)

Write Enable wird autom. gelöscht !

2.4 Write Enable (WREN) and Write Disable (WRDI)

The 25XX010A contains a write enable latch. See Table 2-4 for the Write-Protect Functionality Matrix. This latch must be set before any write operation will be completed internally. The WREN instruction will set the latch, and the WRDI will reset the latch.

The following is a list of conditions under which the write enable latch will be reset:

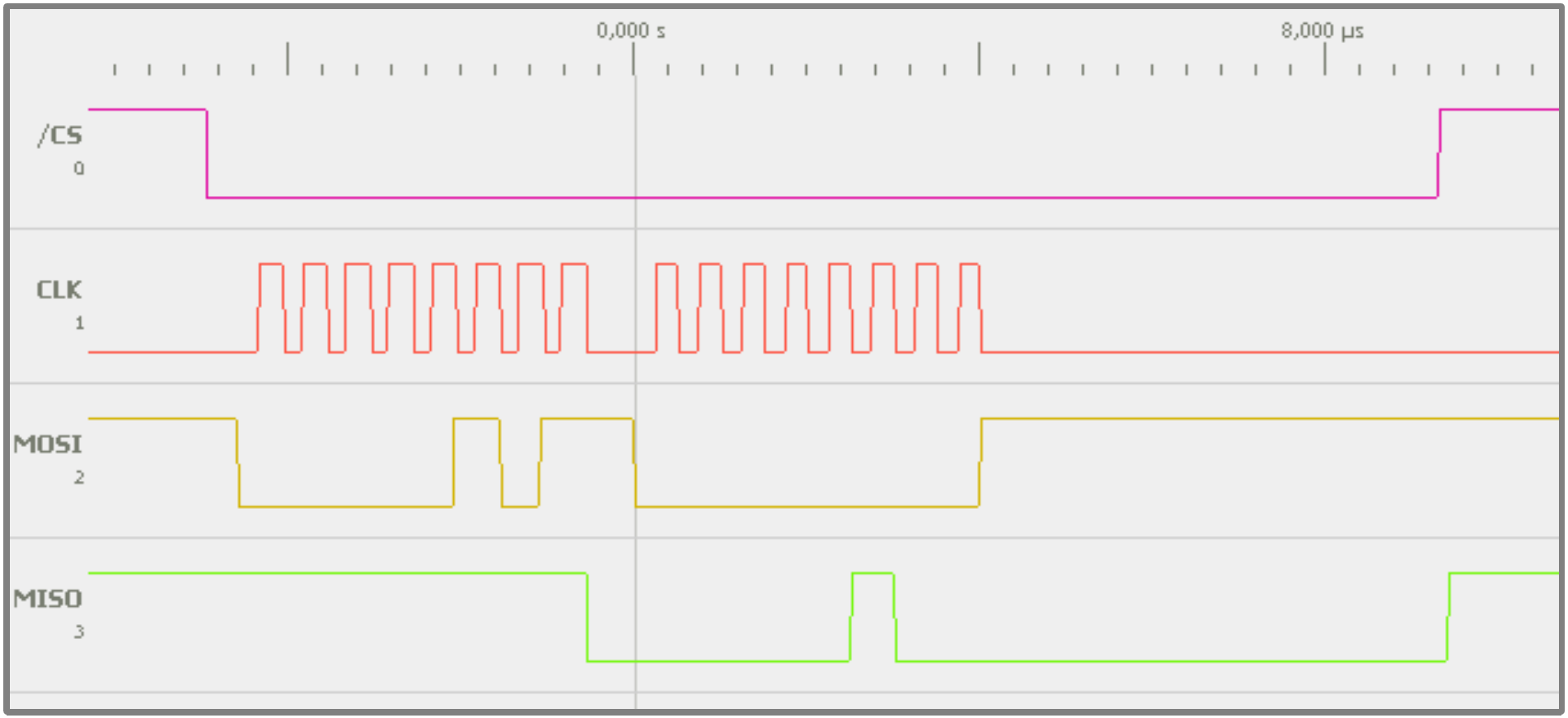
- Power-up
- WRDI instruction successfully executed
- WRSR instruction successfully executed
- WRITE instruction successfully executed
- $\overline{WP}$ pin is brought low

Wartezeiten nach Schreiben !

AC CHARACTERISTICS			Industrial (I): TA = -40°C to +85°C VCC = 1.8V to 5.5V Automotive (E): TA = -40°C to +125°C VCC = 2.5V to 5.5V			
20	TWC	Internal Write Cycle Time (byte or page)	—	5	ms	**(NOTE 3)**

NOTE 3 TWC begins on the rising edge of $\overline{CS}$ after a valid write sequence and ends when the internal write cycle is complete.

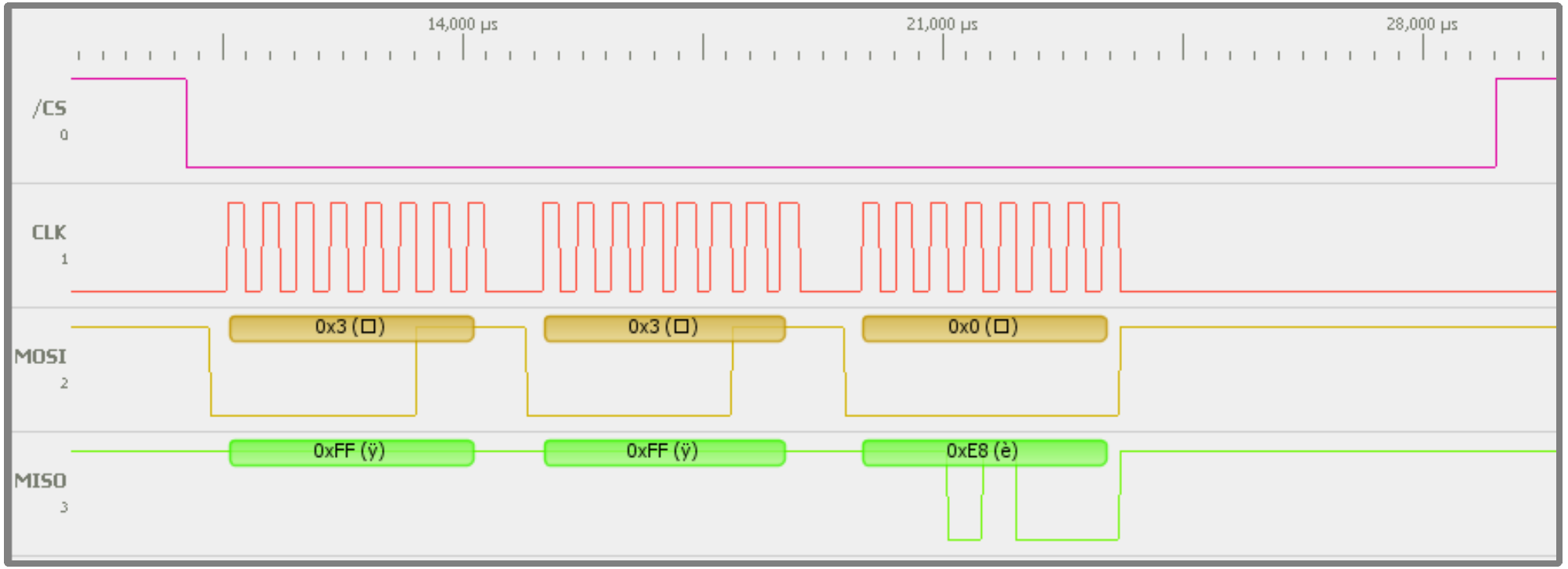
Lesen Status-register (06h) **Inhalt Status-register (04h)**



Programmteil:

```
digitalWrite(CS,LOW);
SPI.transfer(RDSR);
rd_byte = SPI.transfer(0);
digitalWrite(CS,HIGH);
```

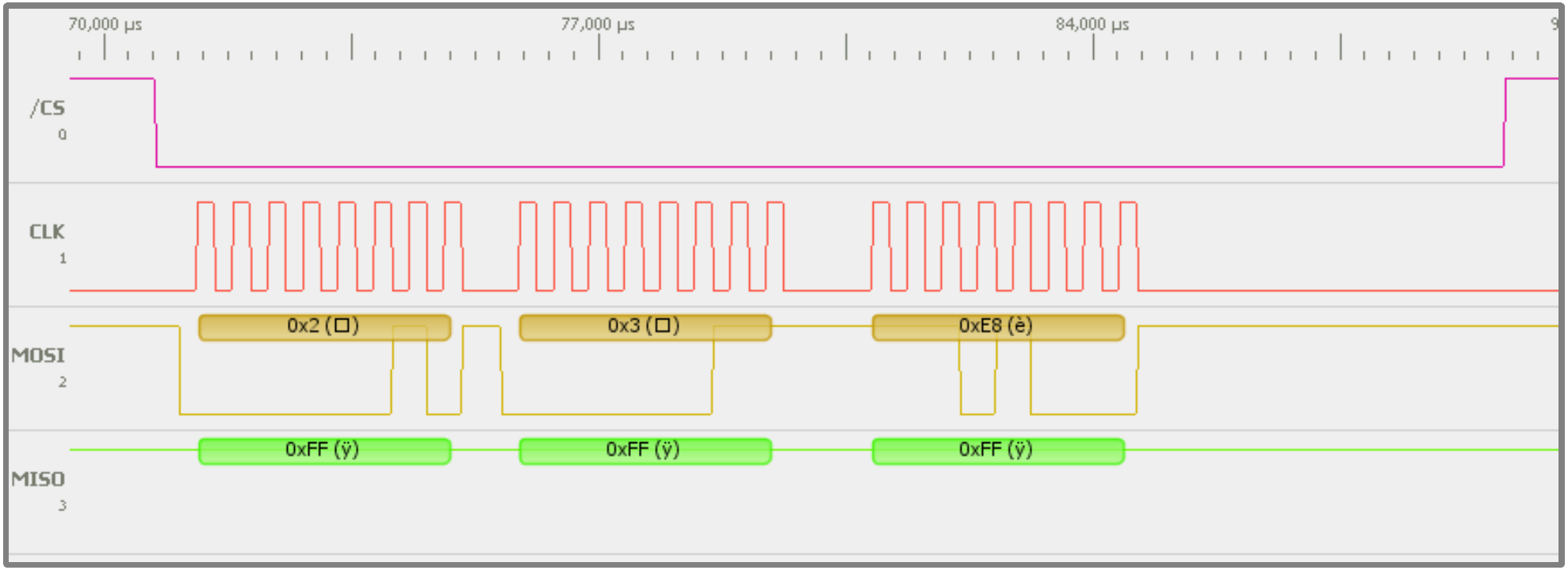
Schreiben EEPROM (03h) Adresse (03h) Daten (E8h)



Programmteil:

```
digitalWrite(CS,LOW);
SPI.transfer(WRITE);
SPI.transfer(BASE_ADR+i);
SPI.transfer(testwert[i]);
digitalWrite(CS,HIGH);
```

Lesen EEPROM (03h) **Adresse (03h)** **Daten (E8h)**



Programmteil:

```
digitalWrite(CS,LOW);
 SPI.transfer(READ);
 SPI.transfer(BASE_ADR+i);
 lesewert[i]=SPI.transfer(i);
 digitalWrite(CS,HIGH);
```

Experiment SPI_EEPROMtest(1)

```
// SPI_EEPROMtest
// Serielles EEPROM 25LC010A ueber SPI
// A.Schultze/DK4AQ, 15.06.2013


#include <SPI.h>
// definition der Portpins
#define CS 10

// Instruction Set 25L010A
//Lesen
#define READ 0x03
//Schreiben
#define WRITE 0x02
//Schreiben sperren
#define WRI 0x04
//Schreiben freigeben
#define WREN 0x06
//Lesen Statusregister
#define RDSR 0x05
//Schreiben Statusregister
```

SPI-Library

Definiton der Bitmuster für die Kommandos an das EEPROM

Experiment SPI_EEPROMtest(2

```
//kein Schreibschutz
#define WP_NO 0b00000000
//Schreibschutz oberes Viertel (60h-7Fh)
#define WP_UQ 0b00000100
//Schreibschutz obere Haelfte (40h-7Fh
#define WP_UH 0b00001000
//Schreibschutz alles (00h-7Fh
#define WP_ALL 0b00001000
```

Bitmuster für die Schreibschutz-Modi

```
//Adresse fuer Schreib-Lese-Versuche
#define BASE_ADR 3
```

Interne EEPROM-Adresse

```
byte wr_byte = 0; //Zwischenspeicher Schreib-Byte
byte rd_byte = 0; //Zwischenspeicher Lese-Byte
byte ee_adr = 0; // interne Adresse EE-Speicher
byte lesewert[8];// Daten, die empfangen wurden
byte testwert[8];// Daten, die geschrieben werden sollen
byte i = 0;//Zaehlvariable
```

Deklaration globale Variablen

Experiment SPI_EEPROMtest(3

```
void setup()
{
  pinMode(CS,OUTPUT);
  digitalWrite(CS,HIGH);

  Serial.begin(9600);
  SPI.begin();

  //Parameter fuer SPI-Biliothek
  SPI.setBitOrder( MSBFIRST);
  SPI.setClockDivider(SPI_CLOCK_DIV8);
  SPI.setDataMode(SPI_MODE0 );

  //Zufallswerte in Array füllen
  for(i=0;i<10;i++)
  {
   testwert[i]= random(255);
  }
```

Ausgangspin D10 für CS-Signal als Ausgang deklarieren und ais HIGH setzen, CS inaktiv.

Libraries initialisieren

MSB zuerst senden (Bitreihenfolge)

Clock = Quarzfreq./8

Phasen/Flanken-Auswahl Mode 0

Testwerte vorbereiten

Experiment SPI_EEPROMtest(4

```
Serial.println("EEPROMTEST");
Serial.println("===========");

//EEPROM initalisieren, Write Enable
digitalWrite(CS,LOW);
SPI.transfer(WREN);
digitalWrite(CS,HIGH);
delay(5);

//EEPROM Kommando memory Protection aufheben
digitalWrite(CS,LOW);
wr_byte = WP_NO;
SPI.transfer(WRSR); //Komando-Byte
SPI.transfer(wr_byte);// Daten-Byte
digitalWrite(CS,HIGH);
delay(5);

Serial.println("Commandbyte");
Serial.println(wr_byte,BIN);
```

EEPROM Kommando: Write Enable

EEPROM Kommando: keine Memory Protection

Experiment SPI_EEPROMtest(5)

```
  digitalWrite(CS,LOW);
  SPI.transfer(WREN);
  digitalWrite(CS,HIGH);
  delay(5);

  //EEPROM Status lesen
  digitalWrite(CS,LOW);
  SPI.transfer(RDSR);
  rd_byte = SPI.transfer(0);
  digitalWrite(CS,HIGH);

  Serial.println("Statusbyte");
  Serial.println(rd_byte,BIN);

}
```

EEPROM- Kommando Write Enable (Grund: durch Schreiben wird Schreibschutz aufgehoben)

EEPROM-Kommando Statusregister Lesen

Ende Setup()

Experiment SPI_EEPROMtest(6)

```
void loop()
{
  //EEPROM ab BASEADR mit Werten beschreiben
  for(i=0;i<8;i++)
  {
    digitalWrite(CS,LOW);
    SPI.transfer(WRITE);
    SPI.transfer(BASE_ADR+i);
    SPI.transfer(testwert[i]);
    digitalWrite(CS,HIGH);
    delay(5);

    //EEPROM Kommando Write Enable
    digitalWrite(CS,LOW);
    SPI.transfer(WREN);
    digitalWrite(CS,HIGH);
    delay(5);
  }
```

8 Bytes ab der internen EEPROM-Adresse BASE_ADR mit dem Wert aus testwert[ ] schreiben

EEPROM-Kommando Write Enable (weil durch Schreiben in Memory aufgehoben

Experiment SPI_EEPROMtest(7)

```
//EEPROM lesen
for(i=0;i<8;i++)
{
    digitalWrite(CS,LOW);
    SPI.transfer(READ);
    SPI.transfer(BASE_ADR+i);
    lesewert[i]=SPI.transfer(i);
    digitalWrite(CS,HIGH);
}
```

8 Bytes aus EEPROM lesen und jeweils in den entsprechenden lesewert[ ] einfüllen

```
//Ausgabe Ergebnis
Serial.println("Schreiben");
for(i=0;i<8;i++)
{
  Serial.print(testwert[i]);
  Serial.print(' ');
}
Serial.println(" ");
```

Ausgabe des Arrays testwert (Quelle) und des Arrays lesewert[] (Ziel)

Experiment SPI_EEPROMtest(8)

```
    Serial.println("Lesen");
  for(i=0;i<8;i++)
  {

    Serial.print(lesewert[i]);
    Serial.print(' ');
  }
  Serial.println(" ");

  //Warten auf Eingabe ueber seriellen Monitor
  while (Serial.available() == 0)
  {
    Serial.read();
  }
}
```

Ende loop()

```
EEPROMTEST
==========
Commandbyte
0
Statusbyte
10
Schreiben
232 19 158 38 175 197 114 68
Lesen
232 19 158 38 175 197 114 68
```

Ausgabe des Tests SPI_EEPROMtest auf dem seriellen Monitor

Testwerte im Array testwert[ ] als Quelle

Ergebniswerte im Array lesewert[ ] als Ziel

Das Rücklesen nach dem Schreiben erbringt gleiche Ergebnisse !