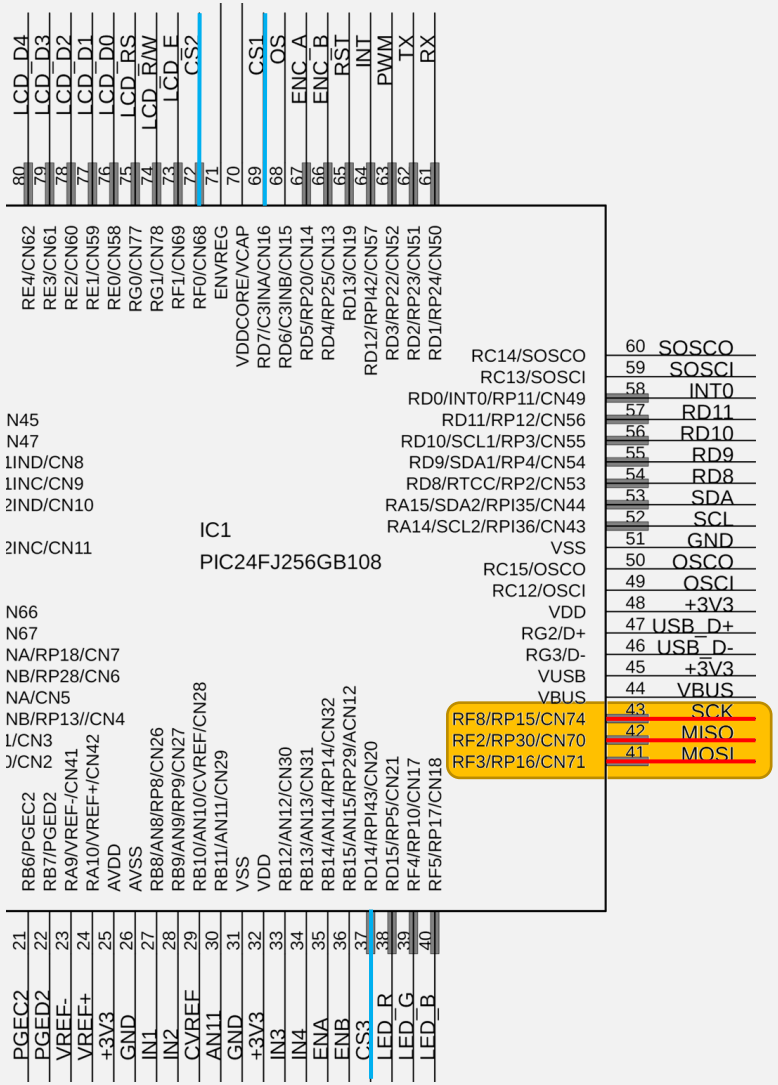
SPI modul alkalmazása

SPI modul

- 3 független SPI modul használható
- Master vagy Slave mód
- 8 vagy 16 bites adatátvitel
- 3 vagy 4 vezetékes (Framed SPI) üzemmód
- Regiszterei:
  - SPIxSTAT Státusz és kontroll regiszter
  - SPIxCON1 Kontroll regiszter1
  - SPIxCON2 Kontroll regiszter2
  - SPIxBUF Beérkező és küldendő adatbuffer

SPI modul bekötése

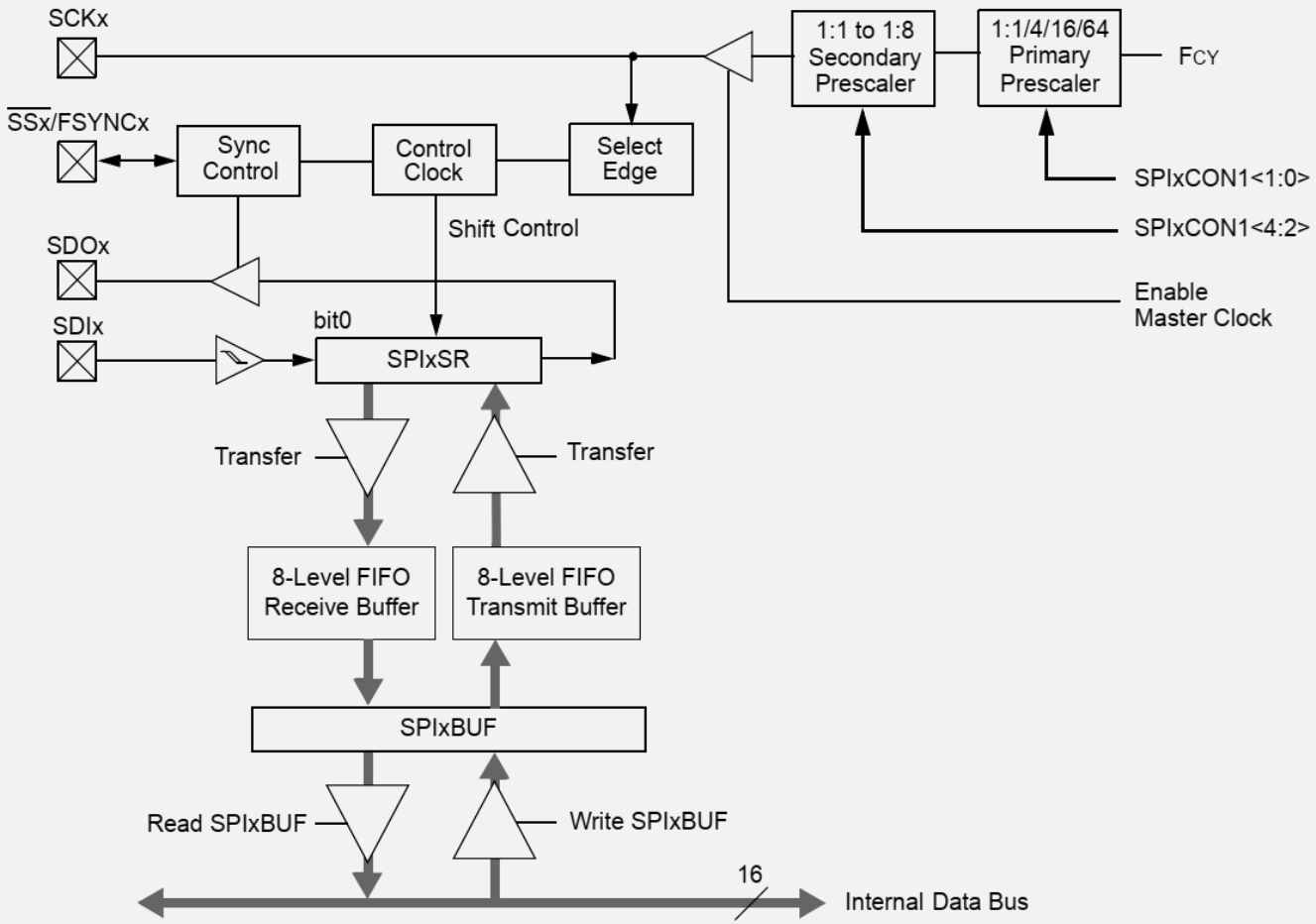


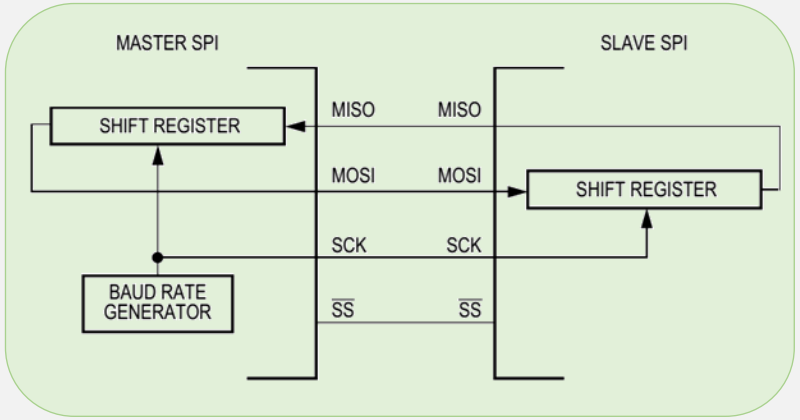
- Az SPI modul lábai a µMogi2 panelon:
  - SDI ⇨ RF2/RP30 (pin42)
  - SDO ⇨ RF3/RP16 (pin41)
  - SCK ⇨ RF8/RP15 (pin43)

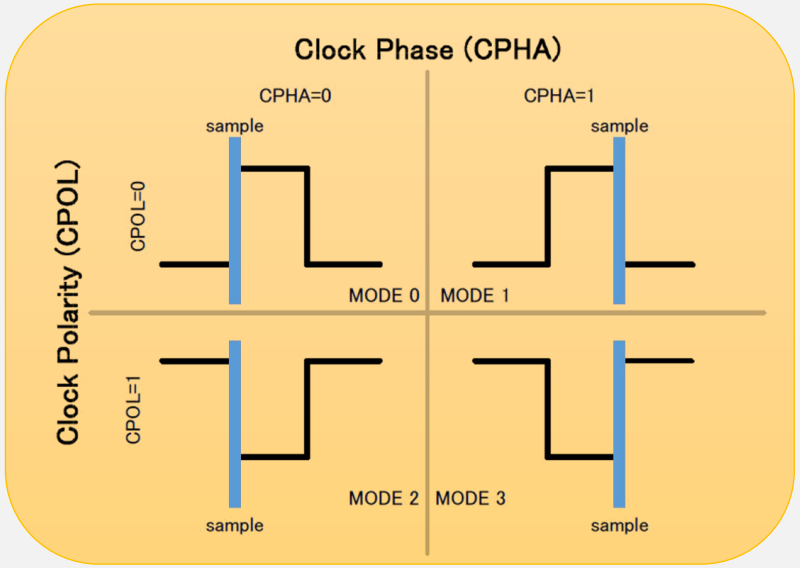
A megadott lábak 5V toleránsak!
A lábak PPS segítségével lesznek összekötve az SPIx modullal.
(más lábakra is átkonfigurálhatók)

- 3db CS lábak
  - CS1 ⇨ RD7 (pin69) ⇨ EEPROM
  - CS2 ⇨ RF0 (pin72) ⇨ microBUS
  - CS3 ⇨ RD14 (pin37) ⇨ SPI header

SPI modul működése







SPIxSTAT: SPIx Status and Control Register

R/W-0	U-0	R/W-0	U-0	U-0	R-0	R-0	R-0
SPIEN	—	SPISIDL	—	—	SPIBEC2	SPIBEC1	SPIBEC0
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

R-0	R/C-0 HS	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0
SRMPT	SPIROV	SRXMPT	SISEL2	SISEL1	SISEL0	SPITBF	SPIRBF
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

SPIEN: SPIx Enable bit
1 = Enables module and configures SCKx, SDOx, SDIx and SSx as serial port pins
0 = Disables module

SPISIDL: Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

SPIBEC<2:0>: SPIx Buffer Element Count bits (valid in Enhanced Buffer mode)
Master mode:
Number of SPI transfers pending.
Slave mode:
Number of SPI transfers unread.

SRMPT: Shift Register (SPIxSR) Empty bit (valid in Enhanced Buffer mode)
1 = SPIx Shift register is empty and ready to send or receive
0 = SPIx Shift register is not empty

SPIROV: Receive Overflow Flag bit
1 = A new byte/word is completely received and discarded. The user software has not read the previous data in the SPIxBUF register.
0 = No overflow has occurred

SRXMPT: Receive FIFO Empty bit (valid in Enhanced Buffer mode)
1 = Receive FIFO is empty
0 = Receive FIFO is not empty

SPIxSTAT: SPIx Status and Control Register

R/W-0	U-0	R/W-0	U-0	U-0	R-0	R-0	R-0
SPIEN	—	SPISIDL	—	—	SPIBEC2	SPIBEC1	SPIBEC0
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

R-0	R/C-0 HS	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0
SRMPT	SPIROV	SRXMPT	SISEL2	SISEL1	SISEL0	SPITBF	SPIRBF
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

SISEL<2:0>: SPIx Buffer Interrupt Mode bits (valid in Enhanced Buffer mode)
111 = Interrupt when SPIx transmit buffer is full (SPITBF bit is set)
110 = Interrupt when last bit is shifted into SPIxSR, as a result, the TX FIFO is empty
101 = Interrupt when the last bit is shifted out of SPIxSR, now the transmit is complete
100 = Interrupt when one data is shifted into the SPIxSR, as a result, the TX FIFO has one open spot
011 = Interrupt when SPIx receive buffer is full (SPIRBF bit set)
010 = Interrupt when SPIx receive buffer is 3/4 or more full
001 = Interrupt when data is available in receive buffer (SRMPT bit is set)
000 = Interrupt when the last data in the receive buffer is read, as a result, the buffer is empty (SRXMPT bit set)

SPITBF: SPIx Transmit Buffer Full Status bit
1 = Transmit not yet started, SPIxTXB is full
0 = Transmit started, SPIxTXB is empty
In Standard Buffer mode:
Automatically set in hardware when CPU writes SPIxBUF location, loading SPIxTXB.
Automatically cleared in hardware when SPIx module transfers data from SPIxTXB to SPIxSR.
In Enhanced Buffer mode:
Automatically set in hardware when CPU writes SPIxBUF location, loading the last available buffer location.
Automatically cleared in hardware when a buffer location is available for a CPU write.

SPIxSTAT: SPIx Status and Control Register

R/W-0	U-0	R/W-0	U-0	U-0	R-0	R-0	R-0
SPIEN	—	SPISIDL	—	—	SPIBEC2	SPIBEC1	SPIBEC0
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

R-0	R/C-0 HS	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0
SRMPT	SPIROV	SRXMPT	SISEL2	SISEL1	SISEL0	SPITBF	SPIRBF
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

SPIRBF: SPIx Receive Buffer Full Status bit
1 = Receive complete, SPIxRXB is full
0 = Receive is not complete, SPIxRXB is empty
In Standard Buffer mode:
Automatically set in hardware when SPIx transfers data from SPIxSR to SPIxRXB.
Automatically cleared in hardware when core reads SPIxBUF location, reading SPIxRXB.
In Enhanced Buffer mode:
Automatically set in hardware when SPIx transfers data from SPIxSR to buffer, filling the last unread buffer location.
Automatically cleared in hardware when a buffer location is available for a transfer from SPIxSR.

SPIXCON1: SPIx Control Register 1

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DISSCK	DISSDO	MODE16	SMP	CKE
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

DISSCK: Disable SCKx pin bit (SPI Master modes only)
1 = Internal SPI clock is disabled; pin functions as I/O
0 = Internal SPI clock is enabled

DISSDO: Disable SDOx pin bit
1 = SDOx pin is not used by module; pin functions as I/O
0 = SDOx pin is controlled by the module

MODE16: Word/Byte Communication Select bit
1 = Communication is word-wide (16 bits)
0 = Communication is byte-wide (8 bits)

SMP: SPIx Data Input Sample Phase bit
Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
Slave mode:
SMP must be cleared when SPIx is used in Slave mode.

CKE: SPIx Clock Edge Select bit(3)
1 = Serial output data changes on transition from active clock state to Idle clock state (see bit 6)
0 = Serial output data changes on transition from Idle clock state to active clock state (see bit 6)

SSEN: Slave Select Enable (Slave mode) bit(4)
1 = SSx pin used for Slave mode
0 = SSx pin not used by module; pin controlled by port function

CKP: Clock Polarity Select bit
1 = Idle state for clock is a high level; active state is a low level
0 = Idle state for clock is a low level; active state is a high level

SPIXCON1: SPIx Control Register 1

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DISSCK	DISSDO	MODE16	SMP	CKE
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

MSTEN: Master Mode Enable bit
1 = Master mode
0 = Slave mode

SPRE<2:0>: Secondary Prescale bits (Master mode)
111 = Secondary prescale 1:1
110 = Secondary prescale 2:1
...
000 = Secondary prescale 8:1

PPRE<1:0>: Primary Prescale bits (Master mode)
11 = Primary prescale 1:1
10 = Primary prescale 4:1
01 = Primary prescale 16:1
00 = Primary prescale 64:1

Master eszköz órajele:

$$FSCK = \frac{FCY}{\text{Primary Prescaler} * \text{Secondary Prescaler}}$$

FCY = 16 MHz		Secondary Prescaler Settings				
		1:1	2:1	4:1	6:1	8:1
Primary Prescaler Settings	1:1	(Invalid)	8000	4000	2667	2000
	4:1	4000	2000	1000	667	500
	16:1	1000	500	250	167	125
	64:1	250	125	63	42	31

Az értékek kHz-ben értendőek.

SPIXCON2: SPIx Control Register 2

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
FRMEN	SPIFSD	SPIFPOL	—	—	—	—	—
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	SPIFE	SPIBEN
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

FRMEN: Framed SPIx Support bit
1 = Framed SPIx support enabled
0 = Framed SPIx support disabled

SPIFSD: Frame Sync Pulse Direction Control on SSx pin bit
1 = Frame sync pulse input (slave)
0 = Frame sync pulse output (master)

SPIFPOL: Frame Sync Pulse Polarity bit (Frame mode only)
1 = Frame sync pulse is active-high
0 = Frame sync pulse is active-low

SPIFE: Frame Sync Pulse Edge Select bit
1 = Frame sync pulse coincides with first bit clock
0 = Frame sync pulse precedes first bit clock

SPIBEN: Enhanced Buffer Enable bit
1 = Enhanced Buffer enabled
0 = Enhanced Buffer disabled (Legacy mode)

SPI modul inicializálása Master módban

```
// Periferia - lab osszerendeles PPS (pp.135)
//PPSUnLock
__builtin_write_OSCCONL(OSCCON & 0xbf);
    //SPI
    RPINR20bits.SDI1R = 30;    //42-es láb SDI1
    RPOR8bits.RP16R = 7;       //41-es láb SDO1
    RPOR7bits.RP15R = 8;       //43-as láb SCLK1
//PPSLock
__builtin_write_OSCCONL(OSCCON | 0x40);
```

```
// SPI inicializálása 2MHz-es órajellel
void spiInit(){
   SPI1CON1 = 0x013A;  // master mód, CKP=0, CKE=1, 8-bites, 1:2, 1:4
   SPI1STAT = 0x8000;  // SPI engedélyezése, státuszok törlése
}
```

SPI modul adatátvitele

- Az adat kiküldése és beérkezése szimultán megy végbe.
  - Ha adatot küldünk ki, akkor a visszatérő érték eldobható
  - Ha csak adatot akarunk fogadni, akkor egy dummy értéket kell küldeni

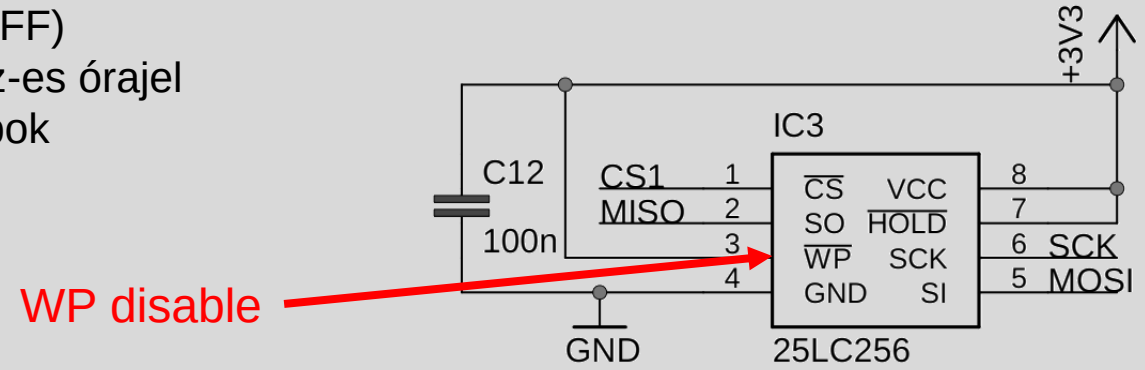
```
#include <stdint.h>

// 1 bájt küldése és fogadása
uint8_t spiWrite(uint8_t b) {
    SPI1BUF = b;                    // buffer írása küldésre
    while(!SPI1STATbits.SPIRBF);    // várakozás az átvitel befejezéséig
    return SPI1BUF;                 // beérkező adat kiolvasása
}
```

SPI alkalmazása

- µMogi2 panelon egy 25LC256 EEPROM található

- 256 Kbyte memória (0x0000-0x7FFF)
- 3,3V-on 5MHz-es órajel
- 64 byte-os lapok



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

```
// EEPROM CS láb
#define CSEE        _LATD7    // CS1 láb
#define TCSEE       _TRISD7   // CS1 láb iránya

void eeInit() {               // EEPROM lábainak inicializálása
    TCSEE = 0;                // CS1 kimenet
    CSEE = 1;                 // EEPROM nincs kiválasztva
}
```

25LC256 utasításkészlete

Instruction Name	Instruction Format	Description
READ	0000 0011	Read data from memory array beginning at selected addr
WRITE	0000 0010	Write data to memory array beginning at selected addr
WRDI	0000 0100	Reset the write enable latch (disable write operations)
WREN	0000 0110	Set the write enable latch (enable write operations)
RDSR	0000 0101	Read STATUS register
WRSR	0000 0001	Write STATUS register

```
//  25LC256 SPI EEPROM parancsai
#define SEE_WRSR      1               // státusz regiszter írása
#define SEE_WRITE     2               // írás parancs
#define SEE_READ      3               // olvasás parancs
#define SEE_WRDI      4               // írás tiltása parancs
#define SEE_RDSR      5               // státusz regiszter olvasása
#define SEE_WREN      6               // írás engedélyezése parancs
```

25LC256 Státusz regiszter

- A státusz regiszter tartalma:

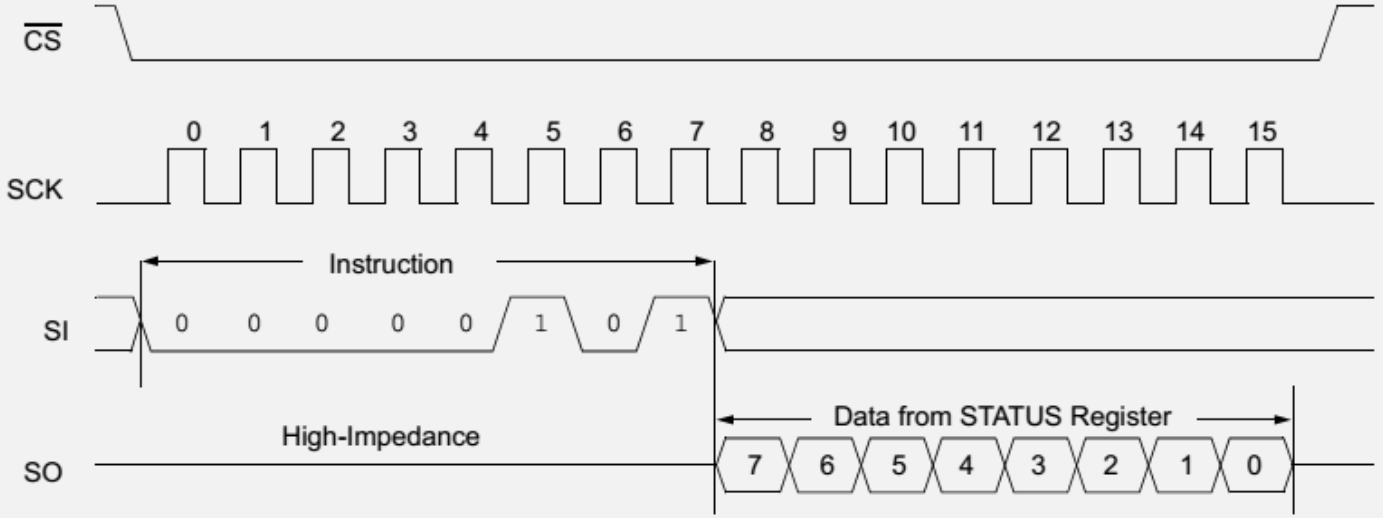
7	6	5	4	3	2	1	0
W/R	-	-	-	W/R	W/R	R	R
WPEN	x	x	x	BP1	BP0	WEL	WIP
W/R = writable/readable. R = read-only							

- WPEN: a hardwares írásvédelem engedélyezése (WP lábtól függ)
- BP1, BP0: blokk védelem beállítása (a védett terület csak olvasható lesz)

BP1	BP0	Array addres Write-Protected
0	0	none
0	1	upper 1/4 (6000h-7FFFh)
1	0	upper 1/2 (4000h-7FFFh)
1	1	all (0000h-7FFFh)

- WEL: írás engedélyezése állapot jelzése (1)
- WIP: írás folyamatban (1), írás befejeződött (0)

25LC256 Státusz regiszter olvasása




```
// státusz regiszter olvasása
uint8_t eeReadSR() {
    CSEE = 0;                   // EEPROM kiválasztása
    spiWrite(SEE_RDSR);         // státusz regiszter olvasása
    uint8_t SR = spiWrite(0);   // küldés/fogadás
    CSEE = 1;                   // EEPROM elengedése
    return SR;
}
```

Gyakorlás

Feladat

Olvassuk ki és írassuk ki a Státusz Regiszter tartalmát hexadecimális számként az LCD-re.
Milyen védelem van beállítva?

25LC256 felülírás engedélyezése

- A véletlen felülírás ellen több védelemmel rendelkezik:

WEL (SR bit 1)	WPEN (SR bit 7)	$\overline{\text{WP}}$ pin	Protected Blocks	Unprotected Blocks	STATUS Register
0	x	x	Protected	Protected	Protected
1	0	x	Protected	Writable	Writable
1	1	0 (low)	Protected	Writable	Protected
1	1	1 (high)	Protected	Writable	Writable

  - minden írás előtt ki kell kapcsolni az írásvédelmet a (Write Enable– WREN) paranccsal – WEL értéke 1 lesz
  - A levédett blokkok csak a védettség (BP1, BP0) megszüntetése után írhatók

```
// írás engedélyezése
void eeWriteEnable() {
    CSEE = 0;             // EEPROM kiválasztása
    spiWrite(SEE_WREN);   // írás engedélyezése parancs
    CSEE = 1;             // EEPROM elengedése
}
```

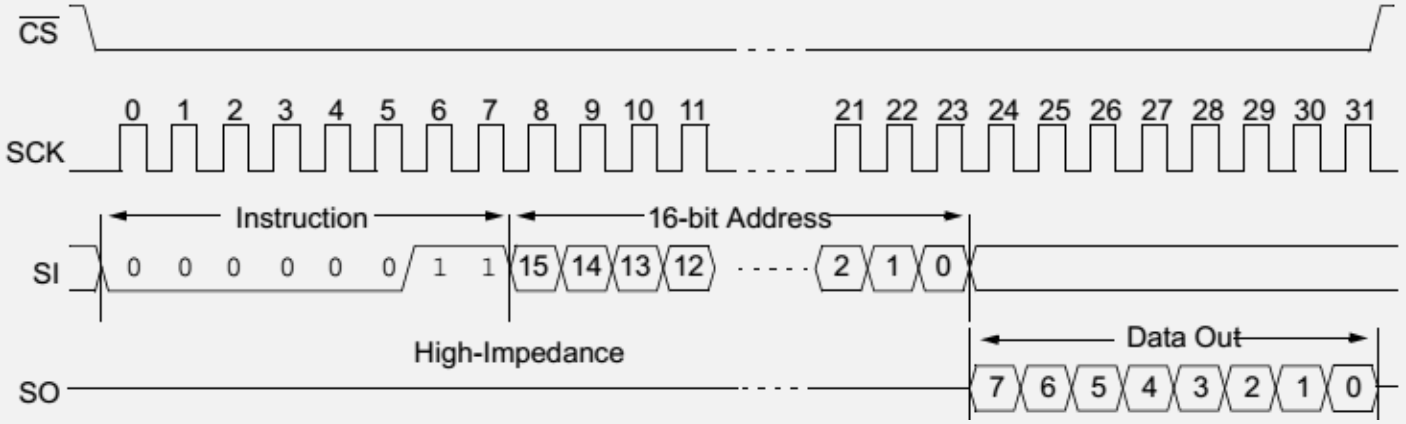
```
// várakozás az írás befejezésére
void eeWait() {
   // írás folyamatának vége (WIP)
   while(eeReadSR() & 0x01);
}
```

25LC256 védett tartalmak kezelése

```
// írás és blokk védelem törlése
void eeClearProtect() {
    eeWriteEnable();              // írás engedélyezése
    CSEE = 0;                     // EEPROM kiválasztása
    spiWrite(SEE_WRSR);           // státusz regiszter írása
    spiWrite(0);                  // védelem törlése
    CSEE = 1;                     // EEPROM elengedése
    eeWait();       // várakozás az írás folyamatának a végére (WIP)
}


// írás és blokk védelem beállítása
void eeSetProtect() {
    eeWriteEnable();              // írás engedélyezése
    CSEE = 0;                     // EEPROM kiválasztása
    spiWrite(SEE_WRSR);           // státusz regiszter írása
    spiWrite(0x8C);               // írás(0x80) és/vagy blokk(0x0C) védelem
    CSEE = 1;                     // EEPROM elengedése
    eeWait();       // várakozás az írás folyamatának a végére (WIP)
}
```

25LC256 adat olvasása

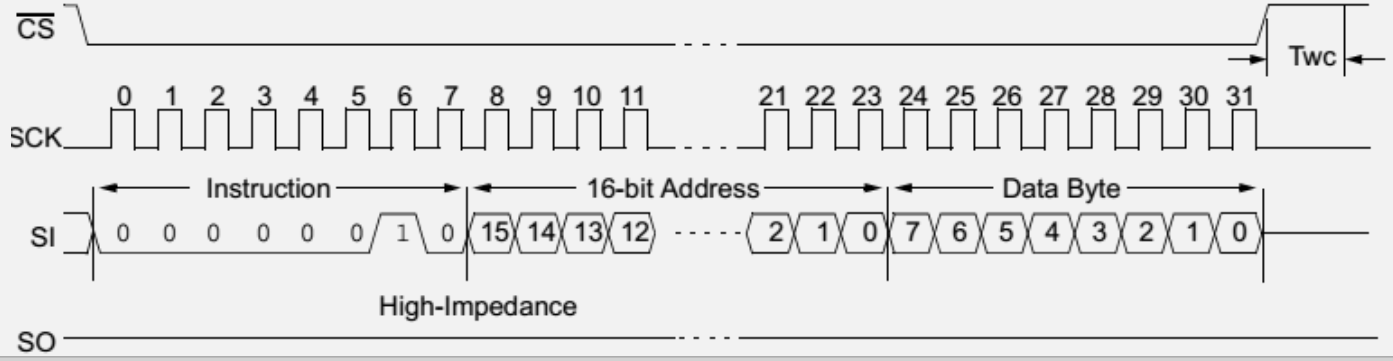



```
// 16 bites cím tartalmának olvasása
uint8_t eeRead(uint16_t addr) {
    CSEE = 0;                   // EEPROM kiválasztása
    spiWrite(SEE_READ);         // olvasás parancs
    spiWrite(addr>>8);          // a cím felső része (MSB)
    spiWrite(addr);             // a cím alsó része (LSB)
    uint8_t b = spiWrite(0);    // dummy érték küldése/érték beolvasása
    CSEE = 1;                   // EEPROM elengedése
    return b;
}
```

25LC256 lap olvasása (max 64 byte)

```
// 16 bites címtől n adat olvasása buff tömbbe
// 64 byte-os laphatárt ne lépjünk át!
void eeReadN(uint16_t addr, uint8_t *buff, int n) {
    CSEE = 0;                   // EEPROM kiválasztása
    spiWrite(SEE_READ);         // olvasás parancs
    spiWrite(addr>>8);          // a cím felső része (MSB)
    spiWrite(addr);             // a cím alsó része (LSB)
    for(int i = 0; i < n; i++)
        buff[i] = spiWrite(0);  // dummy érték küldése/érték beolvasása
    CSEE = 1;                   // EEPROM elengedése
}
```

25LC256 adat írása

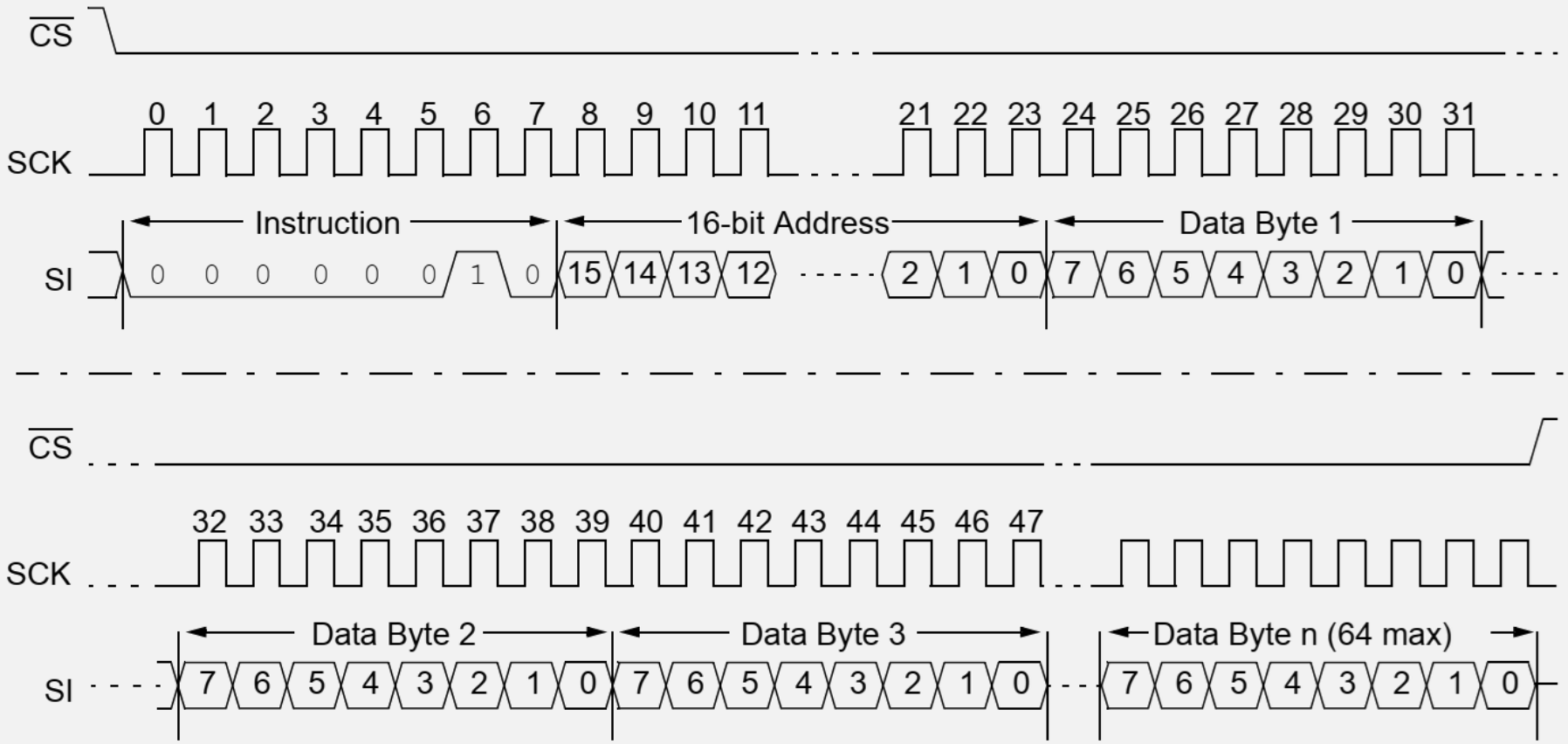



```
// adat írása a 16 bites címre
// csak nem védett adatterület írható, írás után WEL értéke újra 0 lesz
void eeWrite(uint16_t addr, uint8_t data) {
    eeWriteEnable();              // írás engedélyezése

    CSEE = 0;                     // EEPROM kiválasztása
    spiWrite(SEE_WRITE);          // írás parancs
    spiWrite(addr>>8);            // a cím felső része (MSB)
    spiWrite(addr);               // a cím alsó része (LSB)
    spiWrite(data);               // az adat küldése
    CSEE = 1;                     // EEPROM elengedése

    eeWait();                     // várakozás az írás folyamatának a végére (WIP)
}
```

25LC256 lap írása (max 64 byte)



25LC256 lap írása (max 64 byte)

```
// adat írása a 16 bites címtől n adattal buff tömbből
// csak nem védett adatterület írható, írás után WEL értéke újra 0 lesz
void eeWriteN(uint16_t addr, uint8_t *buff, int n) {

    eeWriteEnable();              // írás engedélyezése

    CSEE = 0;                     // EEPROM kiválasztása
    spiWrite(SEE_WRITE);          // írás parancs
    spiWrite(addr>>8);            // a cím felső része (MSB)
    spiWrite(addr);               // a cím alsó része (LSB)
    for(int i = 0; i < n; i++)
        spiWrite(buff[i]);        // az adat küldése
    CSEE = 1;                     // EEPROM elengedése

    eeWait();       // várakozás az írás folyamatának a végére (WIP)

}
```

Gyakorlás

Feladat

Készítsük el egy lézernyomtató nyomtatási lapjainak a számlálását.
Az SW1 gomb megnyomására történjen egy nyomtatás.
Az EEPROM-ban a 0x10-es címen tárolja el az eddigi nyomtatások oldalszámát.
Az LCD kijelzőn jelenjen meg az eddigi nyomtatási oldalak száma.
Újraindítás után is folytatódjon a számlálás.

Feladat

A számláló legyen alkalmas legalább 300 nyomtatási oldal számlálására.

Gyakorlás

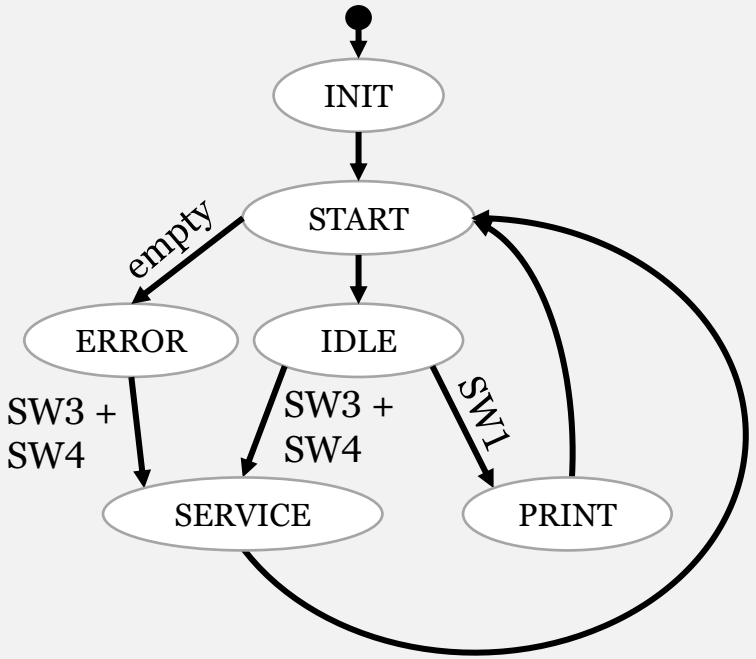
Feladat

Induláskor és nyomtatás közben is ellenőrizze le, hogy az oldalszám elérte-e a toner maximális kapacitását (pl.: 300 oldal).
Ha kiürült a toner, akkor további nyomtatás ne történjen, és írja ki, hogy üres.
A SERVICE az SW3 és SW4 gombok együttes lenyomására legyen meghívva. Ilyenkor jelenjen meg egy SERVICE üzenet 2 másodpercig, amely az EEPROM 0x20-as címétől kezdődik.
Toner csere után elölről kezdődjék a számlálás.

Oldja meg a feladatot állapotgéppel

```
enum States
    {   ST_INIT,
        ST_START,
        ST_IDLE,
        ST_PRINT,
        ST_SERVICE,
        ST_ERROR
    };

enum States State = ST_INIT;
```




```
switch(State){
case ST_INIT:
         // EEPROM betöltése
         State = ST_START;
         break;
case ST_START:
         // Kiírtatás LCD-re
         if(page >= MAXPAGE ) State = ST_ERROR;
         else State = ST_IDLE;
         break;
case ST_IDLE:
         // gombok figyelése
         break;
case ST_PRINT:
         //nyomtatási oldalak növelése és kimentése
         State = ST_START;
         break;
case ST_SERVICE:
         //Toner csere és mentés
         State = ST_START;
         break;
case ST_ERROR:
         if(SW3 && SW4) State = ST_SERVICE;
         break;
}
```