

KOMUNIKACIJE - SPI

SPI je standard za serijsku komunikaciju razvijen od Motorole. Radi se o veoma elegantnom re{enju za brz i sinhroniziran serijski prenos podataka.

■ SPI standard za serijsku komunikaciju je razvijen od strane Motorolinih in`enera, za potrebu brzog i pouzdanog seriskog prenosa podataka.

■ Pri SPI transferu, podatci se {alju preko sifit registra sa seriskim izlazom, a primaju preko {ift registra sa seriskim ulazom. U bilo kom trenutku mo`e postojati najvi{e jedan master da bi se osigurala pravilna komunikacija.

■ Ovo je primer koji opisuje SPI logiku, a pri tome ne ulazi u pojedinosti interne periferije samog mikrokontrolera. Relativno je lako da se po{alje poruka preko SPI kanala od master-a do slave-a, pomo-u tehni-ke literature koja prati Motoroline mikrokontrolere.

■ Asemblerski program koji sledi, -ita deo sadr`aja SPI EEPROM-a i pro-itani sadr`aj upisuje u RAM mikromikrokontrolera.

SPI standard za serisku komunikaciju je razvijen od strane Motorolinih in`enera, za potrebu brzog i pouzdanog seriskog prenosa podataka. Od samog imena (Serial Peripheral Interface) saznajemo da je to standard primarno predvi`en za komunikaciju sa perifernim ure|ajima, ali je mogu}a i me|umikrokontrolerska komunikacija. Postoji {iroka paleta SPI baziranih perifernih ure|aja, od obi-nih TTL {ift registra, do SPI bazirane memorije, LCD-i, A/D konvertori i.t.d.

Da naglasimo na po-etku da se radi o sinhronoj komunikaciji, tako da je mogu}e postizanje velikih brzina prenosa, ali zbog toga su fizi-ka rastojanja izme|u ure|aja mala.

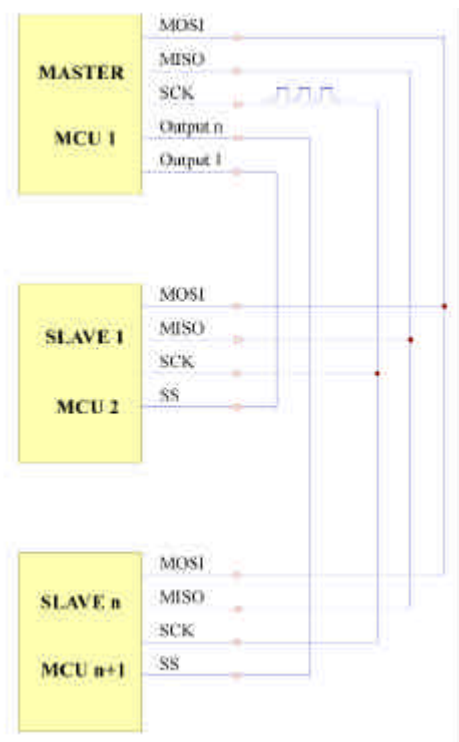
Tip komunikacije	Maksimalna brzina	Komentar
SCI	125 Mb/s	MCU takt = 2MHz,
SPI	1 Mb/s	MCU takt = 2MHz, <i>Master</i>
SPI	2 Mb/s	MCU takt = 2MHz, <i>Slave</i>

Tabela 1. SPI protokol ▲

Iz Tabele 1 vidimo da je SPI komunikacija oko 10 puta br`a od SCI komunikacije kada mikrokontroler radi kao master, i 20 puta br`a kada mikrokontroler radi kao slave. Tako|e se vidi da kada mikrokontroler radi kao slave, mogu}a je brzina prenosa podataka jednakva internom taktu mikrokontrolera.

SPI PROTOKOL

Pri SPI transferu, podatci se {alju preko sifit registra sa seriskim izlazom, a primaju preko {ift registra sa seriskim ulazom. U bilo kom trenutku mo`e postojati najvi{e jedan master da bi se osigurala pravilna komunikacija. Kao {to vidimo na Slici 1 potrebne su 4 linije odnosno koriste se 4 pina za ovaj tip komunikacije.



Slika 1. Povezivanje jednog mastera i dva slejva u SPI komunikaciji ▲

MOSI (Master Output Slave Input). Kada mikrokontroler radi kao master, onda je ovo linija za slanje podataka, a kada radi kao slave, ovo je linija za primanje podataka.

MISO (Master Input Slave Output). Kada mikrokontroler radi kao master, onda je ovo linija za primanje podataka, a kada radi kao slave, ovo je linija za slanje podataka. U slu-aju da je komunikacija isklju-ivo jednosmerna, svakako da se mo`e izvesti sa tri linije (otpadaju MOSI ili MISO zavisno od slu-aja).

SCK (SPI Clock). Ovo je linija za takt pod kojim se izvodi komunikacija. Takt daje master, a {ift registar slave-a prima podatke,

odnosno o-itava ulaz na promenu ovog takta.

SS (Slave Select). Ovaj pin mikrokontrolera se ve`e na liniju za selektiranje slave-a i visoko nivo na ulazu je neaktivno. Ovo zna-i da kada mikrokontroler radi kao slave, i kada je ovaj (u slu~aju ulazni) pin na visokom nivou, onda takt sa SCK linije i podaci sa MOSI linije se ignoriraju. Tek kada se dovede SS na aktivno niskom nivou, SPI sistem slave-a prati promene na pinovima koji su vezani za ovaj tip komunikacije i moze da primi i po{alje informaciju. Kada mikrokontroler radi kao master, SS pin ima dve opcije. Mo`e da se konfigurira kao ulazni pin i da setuje odre|eni flag t.j da inicira interapt u slu~aju da bude doveden na niskom nivou. To je dobra opcija za me|uprosorsku komunikaciju. Na ovaj na-in mo`e uspe{no da se odstarni mogu}nost da dva ili vise mikrokontrolera u datom trenutku poku{aju da budu master-i. Ovo svakako treba re{iti softverski. SS pin tako|e moze da se konfigurira kao izlazni pin, i da selektira neki slave, ili da vrshi bilo kakvu funkciju nezavisnu od SPI podsistema mikrokontrolera.

U slu~aju povezivanja vi{e mikrokontrolera preko SPI kanala, svi MISO pinovi se ve`u zajedno na jednu liniju, svi MOSI pinovi se ve`u zajedno na drugu liniju, i svi SCK pinovi se ve`u zajedno na tre}u liniju kao sto je prikazano na Slici 1. Slika 1 je primer vezivanja kada samo jedan mikrokontroler moze da bude master. Ako uloga mastera

treba da bude promenljiva, onda svaki mikrokontroler treba da ima pojedina~nu vezu sa SS pinovima ostalih mikrokontrolera.

Zamislmo dva {ift registra vezani kao na Slici 2. Neka sa strane sa koje su povezani me|usobno, registri imaju seriske ulaze i izlaze. Kakvi su ulazi i kakav je hardver logike rada, to nas ne interesuje. Pinovi za podatke (DATA1) su povezani zajedno, i to je jednosmerna linija sa koje {ift registar 1 daje podatke {ift registru 2. Pinovi za podatke (DATA2) su povezani zajedno, i to je jednosmerna linija sa koje {ift registar 2 daje podatke {ift registru 1. Pinovi za takt (CLOCK) su povezani zajedno, i to je jednosmerna linija sa koje {ift registar 1 daje takt bez razlike u kojem smeru idu podaci. Pinovi za omogu}avanje komunikacije (ENABLE) su vezani zajedno, i to je jednosmerna linija preko koje {ift registar 1 omogu}ava t.j. dozvoljava {ift registru 2 da radi. Svakako u analogiji sa SPI mikromikrokontrolerskom komunikacijom, {ift 1 radi kao master, a {ift 2 radi kao slave.

Kada jedinica {alje poruku dvojki, redosled operacija je slede}i:

1. Preko linije ENABLE, jedinica omogu}ava rad dvojke ili preciznije omogu}ava da dvojka na svakom prelazu na liniji CLOCK iz visokog na nisko nivo (ili obrnuto zavisno od logike), pro~ita stanje na liniju DATA 1
2. Preko linije CLOCK jedinica daje takt pod kojim dvojka ~ita promene

koje daje jedinica na liniju za podatke DATA 1.

3. Kada jedinica zavr{i sa prenosom, onda stavlja linije CLOCK i ENABLE u stanje mirovanja (idle state).

Kada jedinica `eli da dobije informaciju od dvojke, preduslov koji treba da je ispunjen je da dvojka ve} ima spremnu informaciju koju treba da po{alje. Zadnja pretpostavka u analogiji mikromikrokontrolerske SPI komunikacije zna-i da smo u registar za slanje slave-a programski ve} postavili informaciju koja se trazi. Zna-i redosled operacija je:

1. Preko linije ENABLE, jedinica omogu}ava rad dvojke ili preciznije omogu}ava da dvojka na svakom prelazu na liniji CLOCK iz visokog na nisko nivo (ili obrnuto zavisno od logike), po potrebi promeni stanje na liniju DATA 2
2. Preko linije CLOCK jedinica daje takt pod kojim }e ~itati promene koje daje dvojka na liniju za podatke DATA 2. Ili obrnuto re~eno, dvojka preko linije CLOCK, dobija takt pod kojim }e slati signale.
3. Kada dvojka zavr{i sa prenosom, onda jedinica stavlja linije CLOCK i ENABLE u stanje mirovanja. Jedinica zna da je prenos završen kada (na primer kod osmobitnih informacija), na liniji CLOCK se zavr{i osmi ciklus.

Ovo je primer koji opisuje SPI logiku, a pri tome ne ulazi u pojedinosti interne periferije samog mikrokontrolera. Relativno je lako da se po{alje poruka preko SPI kanala od master-a do slave-a, pomo~u tehni~ke literature koja prati Motoroline mikrokontrolere. Mnogo je te`e i nerazumljivije da izvedemo komunikaciju u obrnutom smeru. Ne mogu da kazem za druge proizvo|a~e, jer nisam imao prilike i potrebe za rad sa SPI podsistemima drugih mikrokontrolera. Najbitnija stvar koju bi `eleo da naglasim ovim tekstom jeste na-in slanja podataka

Slika 2. Hardverska {ema SPI komunikacije



slave-master.

U kom trenutku treba slave da pošalje poruku, odlučuje master. On tada daje takt na liniji CLOCK. Da bi dao takt, zbog automatizovanosti SPI kontrolne i upravljačke logike, master u stvari mora da pošalje poruku na magistralu i normalno da selektira slave-a.

Sve navedeno opisuje suštinu SPI komunikacije. Svakako da ima mnogo detalja koji variraju u zavisnosti od mikrokontrolera i perifernog uređaja, ali to se mnogo lakše usvaja kada se znaju osnovni principi.

SERIJSKI SPI EEPROM

U novije vreme dosta su aktuelni serijski EEPROM-i. U slučaju paralelne spoljašnje memorije gubimo dragocene pinove, tj. cele porte, što u mnogim aplikacijama je veliki problem. Serijske spoljašnje memorije koriste serijski kanal mikrokontrolera koji može da opslužuje više uređaja. Znači serijske memorije mogu mnogo lakše da se priključe u sistem, a da pri tom ne okupiraju dodatne pinove. Postoje varijacije na temu serijske memorije u zavisnosti od tipa memorije i protokola za komunikaciju. Svakako tu su neprikosnoveni EEPROM-i od aspekta tipa memorije zbog lakoće pisanja i brisanja. Dosta popularan tip serijske komunikacije je SPI, a koriste ga mnogi proizvođači mikromikrokontrolera i ogroman broj proizvođača periferija. U narednom delu ovog pasusa biće ukratko objašnjeni princip

funkcioniranja SPI EEPROM-a kada želimo da pročitamo njegov sadržaj, i funkcije nekih pinova koji su bitni za ovaj članak.

Na Slici 3 je prikazana opšta shema jednog Microchip-ovog SPI EEPROM-a 25C160 povezanim sa mikromikrokontrolerom 68HC11F1. Oznake i funkcije pinova Microchip-ovog SPI EEPROM-a koji su na Slici 3 su:

CS (Chip Select) Ovaj pin mora da bude na niskom nivou da bi memorija bila selektovana.

SI (Serial Input) Serijski ulaz za instrukcije adrese i podatke. Podatci se latchingu (latched) na usponskoj ivici takta.

SO (Serial Output) Serijski izlaz. Podatci na ovom pinu se menjaju na padajuću ivicu takta.

SCK (Serial Clock) Ulaz za takt.

INITIJANJE

Prvo se selektira dovođenjem niskog nivoa na pin CS. Onda mora na pin SI da se pošalje instrukcija za inicijanje koja u slučaju je \$03. Posle instrukcije za inicijanje kontrolna logika memorije otključuje 16 bitnu adresu sa koje treba da pročita sadržaj. Posle korektno primljene instrukcije za inicijanje i adrese, traženi sadržaj se postavlja na izlazni shift registar povezan sa pinom SO. Ovaj sadržaj se šalje uz uslov da takt na pinu SCK produži. Podatak sa naredne adrese se čita sa produžavanjem takta na pinu SCK. Tada unutarnji adresni pointer se inkrementira automatski i novi podatak se izvodi preko SO. Ako se dostigne najveća adresa, adresni

broj dobija vrednost \$0000 i odatle se nastavlja inicijanje.

PRIMER POVEZIVANJA MIKROKONTROLERA I SERIJSKOG EEPROMA POMOJU SPI KOMUNIKACIJE

Asemblerski program koji sledi, čita deo sadržaja SPI EEPROM-a i pročita sadržaj upisuje u RAM mikromikrokontrolera. Da napomenemo samo da RAM kod 68HC11F1 počinje od adrese \$0000, EEPROM od \$FE00, a SPI EEPROM kod 25C160, od \$0000. U slučaju mikrokontrolera je pročitati sadržaj serijskog EEPROM-a od adrese \$0007 do \$000C i postaviti sadržaje istim redosledom u RAM mikromikrokontrolera od adrese \$0000 pa nadalje.

Na početku programa su određene sve potrebne inicijalizacije i definisanja adrese sistema. Kao što vidimo od linije 89, program počinje da se izvršava od linije 47 (Inicijalizacija programa). Zatim sledi inicijalizacija SPI kanala. Podešavaju se pinovi za ulaze ili izlaze, polaritet i faza takta i početno se prazni registar za podatke. Podešavanja polariteta i faze takta omogućuju da se prilagodi mikromikrokontroler na bilo kakvoj SPI periferiji.

Sa linije 60 započinje sekvenca inicijanja. U indeksni registar Y se stavlja početna adresa RAM-a gde će započeti upisivanje. Zatim se priprema adresa serijskog EEPROM-a \$0007 sa koje će početi inicijanje. U liniji 65 se selektira EEPROM dovođenjem niskog nivoa na njegov CS pin. Zatim u liniji 66 i 67 se šalje kod za inicijanje. Potprogram za primopredaju se nalazi od linije 82 do linije 85. Iz ovog dela vidimo da je sama SPI komunikacija potpuno automatizovana. Znači, pošto smo

Slika 3. SPI komunikacija mikrokontrolera i EEPROM-a





LISTING PROGRAMA

```

*****
* S.ASM je program u kome je dat osnovni primer
* za SPI komunikaciju izmedju mikrokontrolera
* MC68HC11F1 i Microchip-ovog SPI seriskog
* EEPROM-a 25C160
*
* Autor:          Ivica Gjorgjiev,dipl.inz
* Revision:       1.0
* Mikrokontroler: MC68HC11F1
* Seriski EEPROM: 25C160 SPI Protocol
*****

*POVEZIVANJE PINOVA EEPROM-a
*****
* 68HC11 PD2/MISO ---- 25C160 2 SO *
* 68HC11 PD3/MOSI ---- 25C160 5 SI *
* 68HC11 PD4/SCK ---- 25C160 6 SCK *
* 68HC11 PD5/SS ---- 25C160 1 CS *
* 68HC11 RESET ---- 25C160 3 WP *
* 68HC11 VCC ---- 25C160 7 HOLD *
* 68HC11 VCC ---- 25C160 8 VCC *
* 68HC11 GND ---- 25C160 4 VSS *
*****

0010          RAM      equ      $0010
03ff          STACK   equ      $03FF
1000          REG      equ      $1000
fe00          EEPROM  equ      $FE00
ffff          RESET   equ      $FFFE

0008          portd    equ      $08
0009          ddrd     equ      $09
0028          spcr     equ      $28
0029          spsr     equ      $29
002a          spdr     equ      $2A

0010          org      RAM

0010          ad1      rmb      1
0011          ad2      rmb      1

fe00          org      EEPROM

fe00 8e 03 ff  init     lds      #STACK
fe03 ce 10 00          ldw      #REG

* INICIJALIZACIJA SPI KANALA
fe06 86 3b          ldaa     #%00111011
fe08 a7 09          staa     ddrd,x
fe0a 86 ef          ldaa     #%11101111
fe0c a7 08          staa     portd,x
*I/O ports SS=1, SCK=0
fe0e 86 70          ldaa     #%01110000
fe10 a7 28          staa     spcr,x
*CPOL=0, CPHA=1
fe12 a6 29          ldaa     spsr,x
*Reset SPIF bit

* READ
fe14 18 ce 00 00   read     ldy      #$0000
fe18 86 00          ldaa     #$00
fe1a 97 10          staa     ad1
fe1c 86 07          ldaa     #$07
fe1e 97 11          staa     ad2
fe20 1d 08 20      bclr     portd,x %00100000
*selektira EEPROM
fe23 86 03          ldaa     #%00000011
fe25 bd fe 46      jsr     sendb
*salje kod za citanje
fe28 96 10          ldaa     ad1
fe2a bd fe 46      jsr     sendb
*salje "hi" adresu
fe2d 96 11          ldaa     ad2
fe2f bd fe 46      jsr     sendb
*salje "lo" adresu
fe32 bd fe 46      jsr     sendb
*cita sadrzaj adrese
fe35 18 a7 00      staa     0,y
*stavlja sadrzaj u RAM
fe38 18 08          iny
fe3a 18 8c 00 05   cpy     #$0005
fe3e 26 f2          bne     agn

fe40 1c 08 20      bset     portd,x %00100000
*deselektira EEPROM
fe43 7e fe 4f      jmp     nast

```

```

*SEND
fe46 a7 2a          sendb    staa     spdr, x
fe48 1f 29 80 fc   paus     brolr    spsr,x %10000000 paus
fe4c a6 2a          ldaa     spdr,x
fe4e 39            rts

fe4f 7e fe 14      nast     ovde   proizvava korisnicki program...

ffff             org     RESET
ffff fe 00        fdb     init
                                end

```

poslali kod za ~itanje, od linije 68 do 71 {aljemo 16 bitnu adresu sa koje }e po-eti ~itanje. Sa linije 72 se {alje ista poruka kao sa linije 71. To je bitno kod SPI komunikacije {to smo naglasili u obja{njavanju protokola. Potrebno je produ`avanje takta koji prima memorija i ona automatski nastavlja da {alje sadr`aje sa slede~ih memoriskih lokacija sve dok traje takt. Linije od 74 do 76 slu`e da se pro~ita onoliki broj lokacija koliko treba i da se iza|e iz petlje preko linije 79 naravno deselektiraju~i EEPROM u prethodnoj liniji. Linija 87 produ`ava program, i odavde mo`emo na primer da produ`imo program sa obradom pro~itanih podataka. ☒

