



**UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET
ODSEK ZA RAČUNARSKU TEHNIKU I
INFORMATIKU**

DIPLOMSKI RAD

SERIJSKI KONTROLER

RAČUNARSKOG SISTEMA ZA RAD U

LABORATORIJI

profesor:
dr. Jovan Đorđević

student:
Ivan Palić 338/02

BGD, 2008.

SADRŽAJ

SADRŽAJ	I
UVOD	3
1 STRUKTURA SISTEMA	5
1.1. Opis sistema	5
2 ARHITEKTURA RAČUNARA	8
2.1. Arhitektura procesora	8
2.1.1. Skup programski dostupnih registara	8
2.1.2. Tipovi podataka	8
2.1.3. Načini adresiranja	8
2.1.4. Skup instrukcija	8
2.1.4.1. Opis instrukcija	8
Instrukcija bez dejstva	9
Instrukcija zaustavljanja	9
Instrukcije skoka	9
Instrukcije uslovnog skoka	9
Instrukcije prekida i povratka iz prekidne rutine	9
Instrukcije prenosa	10
Aritmetičke instrukcije	10
Logičke instrukcije	10
Instrukcije pomeranja i rotiranja	11
Instrukcije postavljanja indikatora u PSW	11
2.1.4.2. Kodiranje instrukcija	11
2.2. Mehanizam prekida	12
3 SOFTVERSKI PAKET IGOVSODS	13
3.1 IGoVSoDS - opšte napomene	13
3.2 Korisnički interfejs softverskog paketa IGoVSoDS	13
3.2.1 Glavne komponente programa	13
3.2.2 Osnovne funkcije programa	14
3.2.3 Biblioteka dostupnih kola	14
3.3 Funkcije grafičkog editora digitalne strukture	15
3.3.1 Funkcije za rad sa podlogom za prikaz strukture	15
3.3.2 Funkcije za rad sa objektima	16
3.3.3 Funkcije za rad sa signalima	17
3.3.4 Detaljni pregled funkcija	17
3.3.4.1. Portovi i simboli modula (Port Menager...)	17
3.3.4.2. Posebne vrste objekata	19
3.3.4.2.1. Moduli	19
3.3.4.2.2. Generatori signala	19
3.3.4.2.3. Memorijski moduli	20
3.3.4.3. Funkcija Expand Connectors	20
3.3.4.4. Pregled tipova signala i transformacija signala	20
3.3.4.4.1. Clone/PartClone signali	21
3.3.4.4.2. Magistrala	22
3.3.4.5. Povezivanje signala i modula	22
4 PERIFERIJA	25
4.1 ULAZNA PERIFERIJA	25
4.1.1 Operaciona jedinica	25
4.1.2 Upravljačka jedinica	26

4.1.2.1	Dijagram stanja ulazne periferije	26
4.1.2.2	Dijagram toka operacija	27
4.1.2.3	Algoritam generisanja upravljačkih signala	28
4.1.2.4	Struktura upravljačke jedinice	29
4.2	IZLAZNA PERIFERIJA	31
4.2.1	Operaciona jedinica	31
4.2.2	Upravljačka jedinica	32
4.2.2.1	Dijagram stanja izlazne periferije	32
4.2.2.2	Dijagram toka operacija	33
4.2.2.3	Algoritam generisanja upravljačkih signala	34
4.2.2.4	Struktura upravljačke jedinice	36
5	KONTROLER SA SERIJSKIM PRENOSOM.....	38
5.1	Arhitektura kontrolera periferije sa serijskim prenosom.....	38
	Registri kontrolera za serijski prenos podataka	38
5.2	Organizacija kontrolera periferije sa serijskim prenosom.....	41
5.2.1	Operaciona jedinica	43
5.2.1.1	Blok registri	43
5.2.1.2	Blok interfejs	45
5.2.2	Upravljačka jedinica	47
5.2.2.1	Upravljačka jedinica magistrale.....	48
5.2.2.1.1	Dijagram toka operacija	49
5.2.2.1.2	Algoritam generisanja upravljačkih signala	51
5.2.2.2	Struktura upravljačke jedinice magistrale	53
5.2.2.3	Upravljačka jedinica periferije	55
5.2.2.3.1	Dijagram toka operacija	56
5.2.2.3.2	Algoritam generisanja upravljačkih signala	58
5.2.2.3.3	Struktura upravljačke jedinice	60
6	LABORATORIJA	65
6.1	Inicijalizacija programa	65
6.2	Simulacija na nivou instrukcija	66
6.3	Simulacija na nivou signala.....	75
7	ZAKLJUČAK	81
	LITERATURA	82

UVOD

Današnji život gotovo da ne može da se zamisli bez računara. Računari su sve prisutniji i prisutniji ne samo kao odvojeni uređaji, već i kao sastavni delovi raznih predmeta koji su u svakodnevnoj upotrebi. Mobilni telefoni, automobili, brodovi, vozovi, avioni, navigacioni sistemi, uređaji koji kontrolišu razne procese u industriji svi oni u sebi imaju elemente koji čine računar.

Osnovna namena računara, kao što mu i samo ime govori je da obavlja neko računanje sa ulaznim podacima i da da neki rezultat. U tu svrhu nije neophodno da računar ima tastaturu, monitor, štampač kao periferni uređaje. Srce računara je procesor koji vrši obradu podataka, zatim potrebna je memorija i neki ulazni i izlazni uređaj. Ulazni uređaji osim tastature mogu da budu i razni senzori na osnovu čijih očitavanja će računar odlučiti šta treba da uradi kako bi korigovao trenutno stanje. Izlazni uređaji ne moraju da budu monitor i štampač, već mogu da budu razni sistemi npr. ventili koji kontrolišu pritisak pare u kotlu, gasa u gasovodu, nafte u naftovodu, servo uređaji za razne namene (pa i oni u automobilima).

Kako su računari sve prisutniji u običnom životu svakog čoveka potrebno je i obrazovati stručnjake u toj oblasti. U tu svrhu na Elektrotehničkom fakultetu univerziteta u Beogradu postoje dva odseka: Odsek za računarsku tehniku i informatiku i Odsek za softversko inženjerstvo. Kao sastavni deo nastavnog plana i programa izvodi se nastava iz predmeta koji imaju za cilj objašnjavanje principa na kojima se zasniva rad računara: Osnovi računarske tehnike 1 i 2, Arhitektura računara, Arhitektura i organizacija računara 1 i 2. Svaki od ovih predmeta kao deo nastave ima laboratorijske vežbe koje se izvode pokretanjem simulatora na računaru koji predstavlja traženi sistem. Za više informacija o ovim simulatorima pogledati literaturu [1-6].

Postojeći sistem koji se koristi za laboratorijske vežbe sastoji se od procesora, memorije i raznih periferija. Cilj ovog diplomskog rada bio je da se isprojektuje kontroler za serijski prenos podataka sa periferijom i interfejsom ka sistemskoj magistrali koji će se povezati na postojeći sistem, a zatim napravi i simulator upotrebom alata za pravljenje simulatora i napiše laboratorija sa test primerom kojim se pokazuje funkcionalnost isprojektovanog sistema. Simulator je konstruisan korišćenjem softverskog paketa IGoVSoDS. Primeri korišćeni za testiranje konstruisanog simulatora predstavljeni su u ovom radu i biće detaljno opisani u nastavku. Ovaj rad takođe predstavlja verifikaciju ovog softverskog paketa, gde su izložene prednosti i mane pomenutog alata IGoVSoDS.

U nastavku je dat kraći pregled šta sve i u kojim poglavljima čitalac može da očekuje u nastavku ovog diplomskog rada.

U prvoj glavi je dat pregled na koji način je organizovan računarski sistem za koji je konstruisan kontroler za serijski prenos. Delovi teksta i slike su preuzeti iz literature [7 i 8] uz saglasnost autora.

U drugoj glavi je dat pregled arhitekture računara kao i njegov sažeti prikaz. Delovi teksta i slike u ovoj glavi su preuzeti iz pomenute literature [7 i 8] uz saglasnost autora. Prvo je dat skup registara koji su programski dostupni, kao i njihova značenja. Zatim slede tipovi podataka i formati instrukcija. Slede načini adresiranja koji su primenjeni u simulaciji kao i skup instrukcija. Daje se opis instrukcija a zatim tabelarni pregled kodiranja instrukcija. Na kraju je dat opis prekida koji se mogu dogoditi u ovom računarskom sistemu.

U trećoj glavi je detaljno opisan softverski paket *IGoVSoDS*. Prvo je dat osnovni opis i osnovna funkcionalnost ovog programa, da bi se zatim obratila pažnja na detalje i sve pogodnosti koji ovaj program pruža. Opisan je skup funkcija koji su dostupni korisniku programa, zatim je opisan način rada sa objektima (modulima), signalima, kao i mogući načini povezivanja modula i signala (povezivanje signala i modula sa portovima i povezivanje signala i modula bez portova). U ovom poglavlju je iskorišćen deo teksta iz literature [9 i 10] uz saglasnost autora.

U četvrtoj glavi je dat dizajn i opis realizovanog simulatora periferije za serijski prenos uz pomoć ovog programa. Simulator i periferija su realizovani u skladu sa već projektovanim računarskim sistemom čija je realizacija data u [7 i 8] uz saglasnost autora. Dat je skup modula iz kojih se sastoji projektovana periferija sa detaljnim opisom svakog od njih. Date su slike svih realizovanih blokova i modula koji su konstruisani. Objašnjeni su signali koji se pojavljuju u tim blokovima.

U petoj glavi je dat dizajn i opis realizovanog simulatora kontrolera za serijski prenos uz pomoć ovog programa. Simulator i kontroler su realizovani u skladu sa već projektovanim računarskim sistemom čija je realizacija data u [7 i 8] uz saglasnost autora. Dat je skup modula iz kojih se sastoji projektovani kontroler sa detaljnim opisom svakog od njih. Date su slike svih realizovanih blokova i modula koji su konstruisani. Objašnjeni su signali koji se pojavljuju u tim blokovima.

U šestoj glavi je dat primer programa u assembleru koji je korišćen kao test primer za ispitivanje valjanosti rada konstruisanog simulatora kao i pregled izvršavanja same simulacije sa detaljnim opisom rada simulatora u koracima.

U sedmoj glavi, dat je zaključak u kome se u kratkim crtama opisuje sve ono što je u prethodnih šest glava detaljno rečeno vezano za sam program *IGoVSoDS* kao i za simulator realizovan u njemu. Najzad, predložena su sva zapažanja autora prilikom konstruisanja ovog simulatora, sa svim prednostima, odnosno manama koje ovaj program pruža.

1 STRUKTURA SISTEMA

U ovoj glavi je dat pregled na koji način je organizovan računarski sistem za koji je konstruisan kontroler za serijski prenos. Delovi teksta i slike su preuzeti iz literature [1] uz saglasnost autora. Dat je sažeti pregled arhitekture računara kao i u kratkim crtama njegova organizacija.

1.1. Opis sistema

Računarski sistem sadrži sledeće module: procesor **CPU**, memoriju **MEM**, ulazno/izlazne uređaje **U/I1** do **U/I7** i uređaj za kontrolu ispravnosti rada sistema **FAULT**. Procesor, memorija i ulazno/izlazni uređaji su međusobno povezani sistemskom magistralom **BUS** (slika 1). Moduli računarskog sistema rade sinhrono na isti signal takta **CLK**.

Arhitektura procesora od programski dostupnih registara ima registre opšte namene. Tipovi podataka sa kojima se radi su celobrojne 8-mo bitne veličine sa znakom i bez znaka. Format instrukcija je jednoadresni. Načini adresiranja uključuju registarsko direktno, registarsko indirektno, memorijsko direktno, memorijsko indirektno, registarsko indirektno sa pomerajem, bazno indeksno sa pomerajem, PC relativno sa pomerajem i neposredno adresiranje. Skup instrukcija uključuje instrukcije prenosa, aritmetičke instrukcije, logičke instrukcije, instrukcije pomeranja i rotiranja kao i instrukcije skoka. Prekidi uključuju unutrašnje i spoljašnje prekide sa maskiranjem i prioritiranjem prekida, pri čemu se kontekst procesora se čuva na steku, a adresa prekidne rutine utvrđuje tehnikom vektorisanog mehanizma prekida.

Organizacija procesora je tako odabrana da je čine dve odvojene jedinice i to operaciona jedinica i upravljačka jedinica. Operaciona jedinica se sastoji od blokova povezivanje na magistralu, čitanje instrukcije, formiranje adrese i čitanje operanda, izvršavanje operacija i opsluživanje prekida, međusobno povezanih direktnim vezama. Upravljačka jedinica je realizovana tehnikom ožičene realizacije sa brojačem koraka i dekoderom.

Procesor po linijama **intr₁** do **intr₇** dobija signale maskirajućih zahteva za prekid od ulazno/izlaznih uređaja **U/I1** do **U/I7**, respektivno. Procesor po liniji **inm** dobija signal nemaskirajućeg zahteva za prekid od uređaja za kontrolu ispravnosti rada sistema **FAULT**. Procesor po liniji **hreq** dobija signal zahteva korišćenja magistrale od ulazno/izlaznog uređaja **U/I2**, a po liniji **hack** šalje signal dozvole korišćenja magistrale ulazno/izlaznom uređaju **U/I2**.

Memorija je kapaciteta 60K 8-mo bitnih reči.

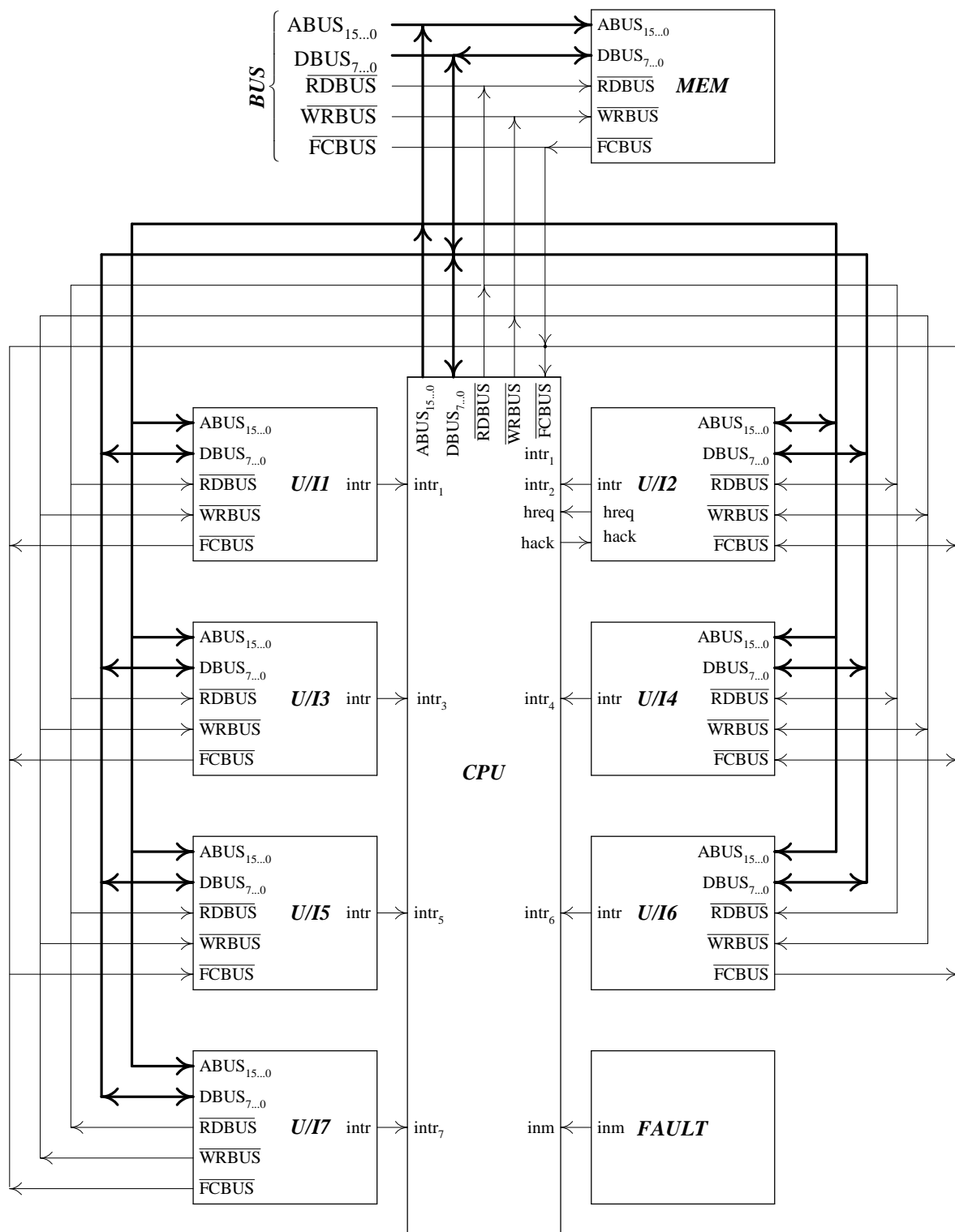
Ulazno/izlaznih uređaja ima 7. Ulazno/izlazni uređaj **U/I1** se sastoji od periferije i kontrolera bez direktnog pristupa memoriji povezanih paralelnim interfejsom. Ulazno/izlazni uređaj **U/I2** se sastoji od periferije i kontrolera za direktnim pristupom memoriji povezanih paralelnim interfejsom. Ulazno/izlazni uređaj **U/I3** se sastoji od periferije i kontrolera bez direktnog pristupa memoriji povezanih serijskim interfejsom. Ulazno/izlazni uređaji **U/I4** do **U/I7** se sastoje samo od kontrolera za generisanje impulsa.

Uređaj za kontrolu ispravnosti rada sistema **FAULT** generiše nemaskirajući prekid **inm** u slučaju otkrivanja neke neispravnosti.

Procesor, memorija i ulazno/izlazni uređaji su povezani asinhronom sistemskom magistralom koju čine adresne linije $ABUS_{15...0}$, linije podataka $DBUS_{7...0}$ i upravljačke linije **RDBUS**, **WRBUS** i **FCBUS**. Na magistrali mogu da se realizuju ciklus čitanja i ciklus upisa. Procesor realizuje cikluse čitanja prilikom čitanja instrukcija i podataka iz memorije i prilikom čitanja statusnih informacija i podataka iz registara kontrolera ulazno/izlaznih uređaja. Procesor realizuje cikluse upisa prilikom upisa podataka u memoriju i prilikom upisa upravljačkih informacija i podataka u registre kontrolera ulazno/izlaznih uređaja. Ulazno/izlazni uređaj sa kontrolerom za direktan pristup memoriji realizuje ciklus čitanja sa memorijom prilikom prenosa iz memorije u uređaj i ciklus upisa sa memorijom prilikom prenosa iz uređaja u memoriju.

Uređaj koji započinje neki ciklusa na magistrali naziva se gazda, a uređaj koji realizuje ciklus sluga. Pri ciklusu čitanja gazda šalje adresu na adresne linije $ABUS_{15...0}$ i signalom na upravljačkoj liniji **RDBUS** startuje čitanje u slugi. Po završenom čitanju sluga šalje očitani podatak na linije podataka $DBUS_{7...0}$ i signalom na upravljačkoj liniji **FCBUS** signalizira gazdi da je čitanje završeno i da je podatak raspoloživ. Pri ciklusu upisa gazda šalje adresu na adresne linije $ABUS_{15...0}$, podatak na linije podataka $DBUS_{17...0}$ i signalom na upravljačkoj liniji **WRBUS** startuje upis u slugi. Po završenom upisu sluga signalom na upravljačkoj liniji **FCBUS** signalizira gazdi da je upis završen.

Kako gazde na magistrali mogu da budu procesor i ulazno/izlazni uređaj sa kontrolerom za direktan pristup memoriji, postoji potreba za arbitracijom među njima. Usvojen je pristup kod koga procesor ima ulogu arbitratra. Ulazno/izlazni uređaj pre realizacije ciklusa na magistrali u kome je gazda signalom $hreq$ traži dozvolu korišćenja magistrale od procesora, a procesor mu signalom $hack$ daje dozvolu.



Slika 1 Konfiguracija sistema

2 ARHITEKTURA RAČUNARA

U ovoj glavi je data arhitektura računarskog sistema za koji je konstruisan kontroler za serijski prenos. Delovi teksta i slike su preuzeti iz literature [1] uz saglasnost autora. Dat je pregled arhitekture računara.

2.1. Arhitektura procesora

Arhitekturu procesora čine:

- skup programski dostupnih registara,
- tipovi podataka,
- načini adresiranja,
- skup instrukcija i
- mehanizam prekida.

U daljem tekstu biće razmotren svaki od ovih aspekata arhitekture procesora.

2.1.1. Skup programski dostupnih registara

Registri procesora koji su programski dostupni su:

- standardni 16-to razredni programski brojač PC,
- 8-mo razredni akumulator AB,
- 16-to razredni akumulator AW,
- 32 16-to razredna registra opšte namene GPR,
- 16-to razredni ukazivač na vrh steka SP,
- programska statusna reč PSW,
- 16-to razredni registar maske IMR i
- standardni 16-to razredni ukazivač na tabelu adresa prekidnih rutina IVTP.

2.1.2. Tipovi podataka

Tipovi podataka koji se koriste u ovom procesoru su celobrojne 8-mo bitne veličine sa znakom i bez znaka, 8-mo bitne binarne reči i 16-to bitne binarne reči.

2.1.3. Načini adresiranja

Koriste samo dva načina adresiranja i njihovo kodiranje i naziv su dati u narednoj tabeli:

AM	Način adresiranja
40h	memorijsko direktno
E0h	neposredno

2.1.4. Skup instrukcija

U ovom delu se navode instrukcije koje postoje u sistemu kao i opis i pregled kodiranja instrukcija koje se koriste u simulaciji.

2.1.4.1. Opis instrukcija

Instrukcije ovog procesora se mogu svrstati u sledećih deset grupa:

- instrukcija bez dejstva **NOP**,
- instrukcija zaustavljanja **HALT**,
- instrukcije skoka,
- instrukcije prenosa,
- aritmetičke instrukcije,
- logičke instrukcije,
- instrukcije pomeranja i rotiranja,
- instrukcije postavljanja indikatora u PSW.

Instrukcija bez dejstva

Instrukcija **NOP** ne proizvodi nikakvo dejstvo. Ova instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija zaustavljanja

Instrukcija **HALT** zaustavlja izvršavanje instrukcija. Ova instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcije skoka

Instrukcije skoka se svrstavaju u sledeće grupe:

- instrukcije uslovnog skoka,
- instrukcije bezuslovnog skoka **JMP**,
- instrukcije skoka na potprogram **JSR** i povratka iz potprograma **RTS** i
- instrukcija prekida **INT** i povratka iz prekidne rutine

Instrukcije uslovnog skoka

Instrukcije uslovnog skoka **BEQL disp**, **BNEQL disp**, **BNEG disp**, **BNNEG disp**, **BOVF disp**, **BNOVF disp**, **BCAR disp**, **BNCAR disp**, **BGRT disp**, **BGRTE disp**, **BLSS disp**, **BLSSE disp**, **BGRTU disp**, **BGRTEU disp**, **BLSSU disp** i **BLSSEU disp** realizuju relativni skok sa pomerajem *disp* u odnosu na programski brojač PC ukoliko je uslov specificiran kodom operacije ispunjen (tabela 1). Pomeraj *disp* je 8-mo bitna celobrojna veličina sa znakom. Ni jedna od ovih instrukcija ne utiče na indikatore N, Z, C i V registra PSW, a njihov format je format instrukcija relativnog skoka.

instrukcija	Značenje	uslov
BEQL	skok na jednako	Z = 1
BNEQ	skok na nejednako	Z = 0

Tabela 1: Instrukcije uslovnog skoka

Instrukcije prekida i povratka iz prekidne rutine

Instrukcija **INT br** programskim putem realizuje prekid i skok na prekidnu rutinu čija se adresa nalazi u ulazu *br* tabele sa adresama prekidnih rutina. Ulaz *br* je 8-mo bitna celobrojna veličina bez znaka. Ima format instrukcija prekida.

Instrukcija **RTI** realizuje povratak iz prekidne rutine tako što restaurira vrednosti programske statusne reči PSW i programskog brojača PC vrednostima sa steka. Ima format bezadresnih instrukcija.

Ni jedna od ovih instrukcija ne utiče na indikatore N, Z, C i V iz registra PSW.

Instrukcije prenosa

Instrukcija **LDB** *src* prenosi sadržaj 8-bitnog operanda *src* u akumulator AB.

Instrukcija **LDW** *src* prenosi sadržaj 16-bitnog operanda *src* u akumulator AW.

Instrukcija **STB** *dst* prenosi sadržaj akumulatora AB u 8-bitni operand *dst*.

Instrukcija **STW** *dst* prenosi sadržaj akumulatora AW u 16-bitni operand *dst*.

Instrukcija **POPB** prenosi sadržaj 8-bitnog operanda sa vrha steka u akumulatora AB.

Instrukcija **POPW** prenosi sadržaj 16-bitnog operanda sa vrha steka u akumulatora AW.

Instrukcija **PUSHB** prenosi sadržaj akumulatora AB u 8-bitni operand na vrh steka.

Instrukcija **PUSHW** prenosi sadržaj akumulatora AW 16-bitni operand na vrh steka.

Instrukcija **LDIVTP** prenosi sadržaj registra IVTP u akumulator AW.

Instrukcija **STIVTP** prenosi sadržaj akumulatora AW u registar IVTP.

Instrukcija **LDIMR** prenosi sadržaj registra IMR u akumulator AW.

Instrukcija **STIMR** prenosi sadržaj akumulatora AW u registar IMR.

Instrukcija **LOADSP** prenosi sadržaj registra SP u akumulator AW.

Instrukcija **STORESP** prenosi sadržaj akumulatora AW u registar SP.

Aritmetičke instrukcije

Instrukcija **ADD** *src* sabira celobrojnu 8-bitnu veličinu koja se nalazi u akumulatoru AB sa operandom *src* koji je celobrojna 8-bitna veličina, a rezultat koji je celobrojna 8-bitna veličina smešta u akumulator AB.

Instrukcija **SUB** *src* oduzima operand *src* koji je celobrojna 8-bitna veličina od celobrojne 8-bitne veličine koja se nalazi u akumulatoru AB, a rezultat koji je celobrojna 8-bitna veličina smešta u akumulator AB.

Instrukcija **INC** inkrementira celobrojnu 8-bitnu veličinu koja se nalazi u akumulatoru AB i rezultat koji je celobrojna 8-bitna veličina smešta u akumulator AB.

Instrukcija **DEC** dekrementira celobrojnu 8-bitnu veličinu koja se nalazi u akumulatoru AB i rezultat koji je celobrojna 8-bitna veličina smešta u akumulator AB.

Logičke instrukcije

Instrukcija **AND** *src* izračunava logičko I 8-bitne veličine koja se nalazi u akumulatoru AB i operanda *src* koji je 8-bitna veličina, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **OR** *src* izračunava logičko ILI 8-bitne veličine koja se nalazi u akumulatoru AB i operanda *src* koji je 8-bitna veličina, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **XOR** *src* izračunava logičko EKSLUZIVNO ILI 8-bitne veličine koja se nalazi u akumulatoru AB i operanda *src* koji je 8-bitna veličina, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **NOT** izračunava logičku NEGACIJU 8-bitne veličine koja se nalazi u akumulatoru AB, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcije pomeranja i rotiranja

Instrukcija **ASR** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB udesno za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **LSR** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB udesno za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **ROR** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB udesno za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **RORC** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB udesno za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **ASL** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB ulevo za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **LSL** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB ulevo za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **ROL** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB ulevo za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcija **ROLC** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB ulevo za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB.

Instrukcije postavljanja indikatora u PSW

Instrukcija **INTD** postavlja nulu u razred I registra PSW.

Instrukcija **INTE** postavlja jedinicu u razred I registra PSW.

Instrukcija **TRPD** postavlja nulu u razred T registra PSW.

Instrukcija **TRPE** postavlja jedinicu u razred T registra PSW.

2.1.4.2. Kodiranje instrukcija

OC _{7...0}	Oznaka	dužina u bajtovima
01h	HALT	1
05h	INTE	1
0Dh	RTI	1
10h	BEQL	2
11h	BNEQL	2
20h	LDB	2, 3 ili 4
21h	LDW	2 ili 4
22h	STB	2 ili 4
2Bh	STIMR	1
33h	DEC	1
34h	AND	1

Tabela 10: Kodiranje instrukcija

2.2 Mehanizam prekida

Kontroler periferija po liniji intr_3 generiše prekid da bi procesoru signalizirao spremnost za prenos podataka (maskirajući prekid). Utvrđivanje adrese prekidne rutine se realizuje na osnovu sadržaja tabele adresa prekidnih rutina (IV tabela) i broja ulaza u IV tabelu. Stoga je u postupku inicijalizacije celog sistema u memoriji, počev od adrese na koju ukazuje sadržaj registra procesora IVTP (*Interrupt Vector Table Pointer*), kreirana IV tabela sa 256 ulaza u kojima se nalaze adrese prekidnih rutina za sve vrste prekida. Na ulazu 11 nalazi se adresa prve instrukcije prekidne rutine za maskirajući prekid po liniji intr_3 . Ulazi 4 i 8 do 255 u IV tabeli su slobodni za adrese prekidnih rutina za prekide izazvane instrukcijom INT.

3 SOFTVERSKI PAKET

IGOVSoDS

U ovoj glavi se daje specifikacija funkcionalnosti programa *IGoVSoDS*. Na početku se navode osnovne karakteristike ovog programa, da bi se kasnije prešlo na detaljniji opis svih funkcija koje su dostupne korisniku prilikom generisanja simulatora kao i način na koji se te funkcije koriste. Autor ovog programa je **mr Nenad M. Grbanović**, a deo teksta je preuzet iz literature [2] uz saglasnost autora.

3.1 *IGoVSoDS* - opšte napomene

IGoVSoDS (**Interactive Generator of Visual Simulators of Digital Structures**) je program koji pruža korisnički interfejs za kreiranje i modifikaciju digitalnih struktura i omogućava detaljno podešavanje značajnih parametara svih digitalnih kola i modula, i obezbeđuje jednostavno upravljanje simulacijom i efikasan sistem pregleda rezultata rada simulacije projektovane digitalne strukture. To je alat koji je razvijen na Elektrotehničkom fakultetu u Beogradu sa ciljem da omogući jednostavan način generisanja svih vrsta simulatora koji će se koristiti za rad u laboratoriji na predmetima arhitekture i organizacije računara.

3.2 Korisnički interfejs softverskog paketa *IGoVSoDS*

U ovom poglavlju će biti dat najosnovniji pregled svega onog sto pruža ovaj softverski paket, a u narednim poglavljima detaljno su prikazane mogućnosti svih delova osnovnog prozora softverskog paketa, kao i svih postupaka za upravljanje radom softverskog paketa.

3.2.1 Glavne komponente programa

Glavne komponente softverskog paketa su:

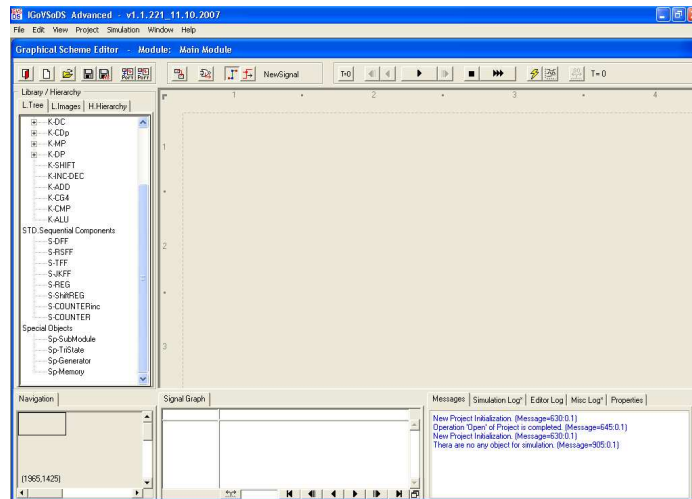
1) Osnovni prozor softverskog paketa, čiji su sastavni delovi:

- Podloga za prikaz digitalne strukture
- Glavni meni i grafički skup poziva funkcije
- Biblioteke kola i hijerarhijsko stablo modula
- Navigacija i pregled rezultata rada simulatora
- Pregled poruka i sadržaja strukture podataka
- Sistem za pomoć pri korišćenju softverskog paketa

2) Funkcije grafičkog editora digitalne strukture

3) Rad sa parametrima simulacije i upravljanje simulatorom

Na slici 35 je prikazan osnovni prozor softverskog paketa *IGoVSoDS*:



Slika 35: Osnovni prozor softverskog paketa *IGoVSoDS*

3.2.2 Osnovne funkcije programa

Osnovne funkcije koje program pruža su:

- Dodavanje, pomeranje i brisanje standardnih sekvencijalnih i kombinacionih kola, kao i specijalnih kola i modula;
- Dodavanje, pomeranje i brisanje signala i linija signala;
- Dodavanje, pomeranje i brisanje natpisa opšte namene (*General Purpose Labels*), natpisa naziva signala (*Signal Name Labels*) i natpisa logičkih stanja na linijama signala koji su širi od jednog bita (*Signal State Labels*);
- Poravnavanje kola u odnosu na odabrano referentno kolo po levoj, desnoj, gornjoj ili donjoj ivici, kao i poništavanje poslednjeg poravnavanja (funkcija *Alignment*);
- Promenu veličine i izgleda *Port Module*-a i *NonPort Module*-a;
- Jednostavnu zamenu *Port Module*-a i *NonPort Module*-a (funkcije *Replace With ALL Connections...* i *Replace Without ANY Connection...*).

Navedene funkcije se detaljno opisuju u poglavljima koji slede.

3.2.3 Biblioteka dostupnih kola

U programu je dostupna biblioteka realizovanih logičkih elemenata - kombinacionih i sekvencijalnih kola uz pomoć kojih je moguće konstruisati složenije elemente - module koji se kasnije mogu koristiti u konstruisanju jos složenijih struktura. Na raspolaganju su sledeći tipovi kola i modula:

- Standardna kombinaciona kola, sa označenim brojem ulaznih i izlaznih konektora: K-NOT, K-AND_i, K-NAND_i, K-OR_i, K-NOR_i, K-ExOR2, K-ExNOR2, K-DC_{k-m}, K-CDp_{k-m}, K-MP_{k-m}, K-DP_{k-m}, K-SHIFT, K-INC-DEC, K-ADD, K-CG4, K-CMP i K-ALU.

Nakon izbora iz biblioteke, na ulazne konektore ovih kola mogu da se povežu signali širine 1 bit, dok su izlazni signali širine 1 bit. Primenom određene funkcije *Expand Connectors...* moguće je promeniti širinu ovih konektora. O ovoj funkciji će biti više reči u nastavku.

- Standardna kombinaciona kola, sa modifikovanim ulaznim konektorima: K-gAND,

K-gNAND, K-gOR, K-gNOR, K-gDC_{k-m}, K-gCDp_{k-m}, K-gMP_{k-m} i K-gDP_{k-m}. Sva kola iz ove grupe imaju po jedan ulazni konektor na koji može da se poveže signal sa širinom od 1 do 32 bita, dok je izlazni signal uvek širine 1 bit. Nakon izbora iz biblioteke, ulazni konektori ovih kola prihvataju signale širine 16 bita. Primenom funkcije *Expand Connectors...* menjaju se širine samo ulaznih konektora.

- Standardna sekvencijalna kola: S-DFF, S-RSFF, S-TFF, S-JKFF, S-REG, S-ShiftREG, S-COUNTERinc i S-COUNTER. Nakon izbora flip-flopova iz biblioteke kola, na ulazne konektore mogu da se povežu signali širine 1 bit, dok su izlazni signali širine 1 bit. Primenom funkcije *Expand Connectors...* menjaju se širine svih konektora. Nakon izbora registara i brojača iz biblioteke kola, ulazni i izlazni signali za podatke su širine 16 bita, dok su svi kontrolni ulazni i izlazni signali širine 1 bit. Primenom funkcije *Expand Connectors...* menjaju se širine samo ulaznih i izlaznih signala za podatke.
- Specijalni moduli:
 - Sp-SubModule – *NonPort Module*.
 - Sp-TriState – trostatički bafer koji omogućava povezivanje signala na magistralu. Isključivo izlazni signal ovog modula može da se poveže na magistralu. Nakon izbora iz biblioteke svi ulazni i izlazni signali su širine 1 bit. Primenom funkcije *Expand Connectors...* menjaju se širine ulaznog i izlaznog signala, dok kontrolni signal zadržava širinu 1 bit.
 - Sp-Generator – generator signala.
 - Sp-Memory – memorijski modul.

3.3 Funkcije grafičkog editora digitalne strukture

U ovom poglavlju je dat detaljan uvid u sve mogućnosti koji pruža ovaj program. Dat je skup svih relevantnih funkcija uz pomoć kojih se konstruiše simulator kao i opis šta te funkcije rade. Biće takođe, prikazani postupci za rad sa objektima koji predstavljaju standardna kola, sa objektima koji predstavljaju module kao i sa signalima i linijama signala.

3.3.1 Funkcije za rad sa podlogom za prikaz strukture

Ova grupa funkcija je dostupna korisniku kada se pokazivač misa nalazi na slobodnoj površini podloge za prikaz strukture i kada se aktivira desni taster misa. Obezbeđene su funkcije koje vrše:

- 1) Prelazak na prikaz hijerarhijski nadređenog modula;
- 2) Prelazak na prikaz hijerarhijski najvišeg modula, odnosno glavnog modula;
- 3) Rad sa natpisima opšte namene: kreiranje natpisa (*Create*), modifikacija teksta natpisa (*Edit*), kreiranje natpisa koji je identičan natpisu iznad koga je otvoren padajući meni (*Create Copy*), smanjenje i povećanje veličine slova natpisa (*Font Size Down* i *Font Size Up*, respektivno), modifikacija izgleda slova natpisa (*Bold*, *Italic* i *Underline*) i brisanje natpisa (*Delete*);
- 4) Rad sa natpisima za prikaz naziva signala: kreiranje natpisa (*New...*), modifikacija teksta natpisa (*Edit*), poravnavanje natpisa po vertikali i horizontali sa linijom signala (*Align to Line*), centriranje natpisa sa sredinom linije signala (*Align to Center*) i brisanje natpisa (*Delete...*);
- 5) Rad sa natpisima za prikaz stanja signala, za signale veće širine od jednog bita:

kreiranje natpisa (*New...*), aktiviranje binarnog i heksadecimalnog formata natpisa (*Binary* i *Hexadecimal*, respektivno), poravnavanje natpisa po vertikali i horizontali sa linijom signala (*Align to Line*), centriranje natpisa sa sredinom linije signala (*Align to Center*) i brisanje natpisa (*Delete...*);

6) Otvaranje dijaloga za kreiranje, modifikaciju i brisanje podataka o portovima trenutno aktivne strukture. Ovako definisani portovi koriste se samo kada se trenutno aktivna digitalna struktura integriše u neku drugu digitalnu strukturu, aktiviranjem funkcije *Add Module/SubModule with Ports...*;

7) Otvaranje dijaloga za modifikaciju i podesavanje karakteristike modula.

3.3.2 Funkcije za rad sa objektima

Ova grupa funkcija je dostupna korisniku kada se pokazivač misa postavi na neko kolo ili modul i aktivira desni taster misa. Obezbeđene su:

1) Skup funkcija za rad sa modulima:

- *Edit Symbol Title* – otvaranje polja za modifikaciju teksta natpisa na simbolu koji predstavlja modul;
- *Replace With ALL Connections...* – zamena modula trenutno aktivnog projekta drugim modulom, SA očuvanjem konekcija modula koji se zamenjuje, na modul koji se učitava;
- *Replace Without ANY Connection...* – zamena modula trenutno aktivnog projekta drugim modulom, BEZ očuvanja konekcija modula koji se zamenjuje, na modul koji se učitava;

2) Skup funkcija za rad sa generatorom signala (generator signala je element koji proizvodi signal koji ima vrednost koja je zadata funkcijom):

- *Function...* – otvaranje dijaloga za definisanje parametara funkcije kojom se generiše signal na izlazu generatora signala;
- *Switch Value* – funkcija koja za generator signala čiji izlaz ima samo jedan bit prebacuje vrednost iz vrednosti logičke nule u vrednost logičke jedinice, i obrnuto. Funkcija nije dostupna za generator signala čiji je izlazni signali veće širine od jednog bita;
- *None, Unknown* i *TriState* – parametri koji određuje da li je na izlazu generatora vrednost signala koja je saglasna rezultatu rada funkcije generatora signala (*None*), ili je nametnuta jedna od specijalnih vrednosti: nepoznata vrednost (*Unknown*) ili visoka impedansa (*TriState*);

3) Skup funkcija za rad sa memorijskim modulom (*Memory Module*): otvaranje dijaloga za pregled sadržaja memorijskih lokacija (*HEX Dump List...*), otvaranje dijaloga za pregled i modifikaciju sadržaja memorijskih lokacija (*Change Data...*), kao i dijalozi za učitavanje i snimanje sadržaja memorijskih lokacija iz, odnosno u datoteku (*Import Memory Data...* i *Export Memory Data...*, respektivno). Funkcija memorijskog modula će biti opisana u nastavku;

4) Funkcija za modifikaciju broja bitova na konektorima kola i modula (*Expand Connectors...*). Funkcija nije dostupna za *Port Module* i *NonPort Module* objekte. Detaljni opis mogućnosti funkcije *Expand Connectors...* se daje u nastavku;

5) Funkcija za dodavanje (*CREATE*) i brisanje (*REMOVE*) kružića koji se integriše sa konektorom kola i modula, kojim se ostvaruje negacija signala na konektoru

(*Connector Not Sign*). Funkcija nije dostupna za specijalni modul Sp–Generator, kao i za *Port Module* i *NonPort Module* objekte.

3.3.3 Funkcije za rad sa signalima

Ova grupa funkcija je dostupna korisniku kada se pokazivač misa postavi na neku liniju signala i aktivira desni taster misa. Obezbeđeni su:

- 1) Funkcija za transformaciju signala u magistralu (*Make New BUS*). Funkcija je dostupna samo za signale koji nemaju izvor vrednosti;
- 2) Funkcija koja vrši promenu broja bitova magistrale (*Expand BUS...*). Funkcija je dostupna samo za signale koji su transformisani u magistrale, kada nema signala koji su povezani na magistralu;
- 3) Funkcije koje vrse brisanje određene linije signala (ako se signal sastoji iz više linija) i brisanje celog signala (svih linija signala);
- 4) Funkcija koja na pravoj liniji signala kreira prelomnu tačku (*New Break Point*), nakon čega je moguće da se napravi složenija putanja linije signala. Prelomna tačka koja se kreira nalazi se na mestu gde je otvoren meni (aktiviran taster misa);
- 5) Funkcije za brisanje poslednje linije signala na signalu koji se sastoji iz više linija, kao i dodavanje jedne linije signala na kraj linija koji čine jedan signal;
- 6) Funkcija za otvaranje dijaloga za podešavanje osnovnih parametara vezanih za signal nad kojim je funkcija aktivirana (*Properties...*).

3.3.4 Detaljni pregled funkcija

U poglavljima koja slede daje se detaljni opis najznacajnijih karakteristika i funkcija koje postoje u ovom programu. Karakteristike koje će biti opisane su:

- Portovi i simboli modula (*Port Menager...*);
- Posebne vrste objekata: moduli, generatori signala, memorijski moduli;
- Funkcija *Expand Connectors...* ;
- Osobine signala i kreiranje *Clone/PartClone* signala;

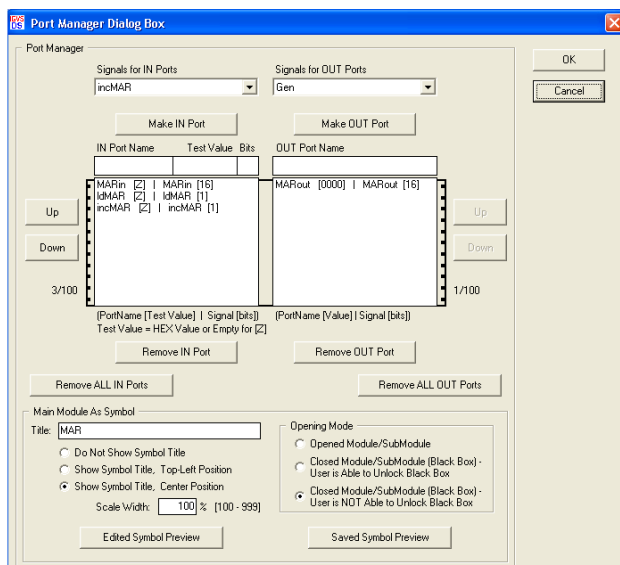
3.3.4.1. Portovi i simboli modula (*Port Menager...*)

Jedna od prednosti ovog programa je mogućnost kreiranja modula na osnovu trenutno aktivne digitalne strukture. Tako kreiran modul može da se integriše u neku drugu digitalnu strukturu. Na slici 12 prikazan je dijalog za kreiranje i modifikaciju podataka o portovima. U tom dijalogu se nalazi lista kandidata za ulazne i izlazne portove (*Signal for IN ports* i *Signal for OUT ports*). Aktiviranjem tastera *Make IN Port* i *Make OUT Port* signal koji je trenutno odabran u odgovarajućoj listi kandidata promovise se u ulazni (*IN*), odnosno izlazni (*OUT*) port, respektivno. Signali koji se odrede za ulazne i izlazne portove mogu da se obrisu čime se uklanjaju iz liste portova. U ovom dijalogu postoji mogućnost za podešavanje izgleda modula. Obezbeđena je modifikacija natpisa na simbolu, mesta gde će biti napisan naziv, procentualne vrednosti za deformaciju veličine simbola, kao i parametra koji određuje da li će korisnik moći da uđe i pregleda strukturu modula, ili to neće biti dopušteno (*Opening Mode*). Omogućen je izbor tri vrednosti parametra *Opening Mode*:

- *Opened Module/SubModule* – dozvoljeno je da se uđe i pregleda struktura modula
- *Closed Module/SubModule (Black Box), User is Able to Unlock Black Box* – nije

dozvoljeno da se uđe i pregleda struktura modula, ali korisnik može da promeni dozvolu za ulazak u modul

- *Closed Module/SubModule (Black Box), User is NOT Able to Unlock Black Box* – nije dozvoljeno da se uđe i pregleda struktura modula



Slika 36: Dijalog za kreiranje i modifikaciju podataka o portovima

Ukoliko su liste ulaznih i izlaznih portova prazne, modul će se koristiti kao *NonPort Module*. Prema dijalogu sa slike 36 kreiran je *Port Module*, čiji je izgled prikazan na slici 37.



Slika 37: Jedan mogući prikaz *Port Module*-a

Postupak integracije modula u drugu elektronsku digitalnu strukturu obavlja se aktiviranjem funkcija *Add Module/SubModule with Ports...* i *Add Module/SubModule w/o Ports...* Ukoliko je aktivirana funkcija *Add Module/SubModule with Ports...* modul će biti integrisan kao simbol sa portovima samo ako su portovi definisani, u suprotnom, modul će biti integrisan kao simbol bez portova. Ukoliko je aktivirana funkcija *Add Module/SubModule w/o Ports...* modul će biti integrisan kao simbol bez portova, bez obzira da li su portovi definisani ili nisu definisani.

Pored definisanja portova koji se koriste za povezivanje signala i modula, moguće je i povezivanje signala i modula bez definisanih portova. Moduli na koje su povezani signali mogu da se zamene drugim modulima tako da se ostvarene konekcije sačuvaju ili da se raskinu. Zamena modula sa portovima (*Port Module*) ostvaruje se korišćenjem informacija o portovima, ali takve informacije ne postoje za module bez portova (*NonPort Module*). Za potrebe ostvarivanja zamene modula bez portova, sa mogućnošću da se očuvaju uspostavljene konekcije, obezbeđena je mogućnost numeracije signala u modulu.

3.3.4.2. Posebne vrste objekata

3.3.4.2.1. Moduli

Jedna od važnijih funkcija koji ovaj program pruža je i zamena odabranog modula na digitalnoj strukturi drugim modulom (funkcije *Replace With ALL Connections...* i *Replace Without ANY Connection...*).

Osnovna namena funkcija za zamenu modula je obezbeđivanje mogućnosti da se u okviru aktivne digitalne strukture proizvoljni modul zameni nekim drugim modulom uz očuvanje ostvarenih konekcija signala van modula i signala unutar modula, ili bez očuvanja navedenih konekcija. Ponašanje funkcija delimično se razlikuje za module sa portovima (*Port Module*) i module bez portova (*NonPort Module*).

U zavisnosti od tipa funkcije i postojanja portova modula, razlikuju se četiri slučaja zamene module.

- zamena modula sa očuvanjem konekcija (*Replace With ALL Connections...*), za module sa portovima (*Port Module*) i za module bez portova (*NonPort Module*):
- za module sa portovima (*Port Module*): proverava se da li novi modul ima dovoljno portova, tako da sve postojeće konekcije signala sa starim modulom mogu da se ostvare sa novim modulom. Saglasnost portova se ne utvrđuje po nazivima portova, već po poziciji portova u nizu ulaznih, odnosno izlaznih portova, posebno za svaki tip portova.
- za module bez portova (*NonPort Module*): proverava se da li stari i novi modul imaju numerisane signale, tako da sve postojeće konekcije signala sa starim modulom mogu da se ostvare sa novim modulom. Novi modul može da ima proizvoljan broj signala, ali svi numerisani signali starog modula na koje su povezani signali van modula, moraju da imaju odgovarajući signal u novom modulu, sa identičnom numeracijom.

Modul koji je nasledio položaj starog modula u hijerarhijskoj strukturi modula, integrisan je u aktivnu digitalnu strukturu na identičan način kao i modul koji je zamenjen.

- zamena modula bez očuvanja konekcija (*Replace Without ANY Connection...*), za module sa portovima (*Port Module*) i za module bez portova (*NonPort Module*):

Modul koji je nasledio položaj starog modula u hijerarhijskoj strukturi modula, integrisan je u aktivnu digitalnu strukturu na identičan način kao i modul koji je zamenjen, osim konekcija koje ne postoje.

3.3.4.2.2. Generatori signala

U ranijem delu teksta je već napomenuto, u kraćim crtama, šta je generator signala. Generator signala je realizovan kao modul koji ima samo jedan izlazni signal, gde širina signala može da se podešava. Izlazni signal menja vrednost u svakom ciklusu signala takta pridruženog modulu kome generator pripada. Na raspolaganju su tri funkcije koje upravljaju radom generatora signala:

- *Constant* – izlazni signal ne menja vrednost
- *Counter* – izlazni signal menja vrednost – povećanje (*Increment*), odnosno smanjivanje (*Decrement*) vrednosti izlaznog signala za vrednost polja *Increment Value*, odnosno *Decrement Value*, respektivno
- *Oscillator* – izlazni signal menja vrednost tako što se svi bitovi izlaznog signala

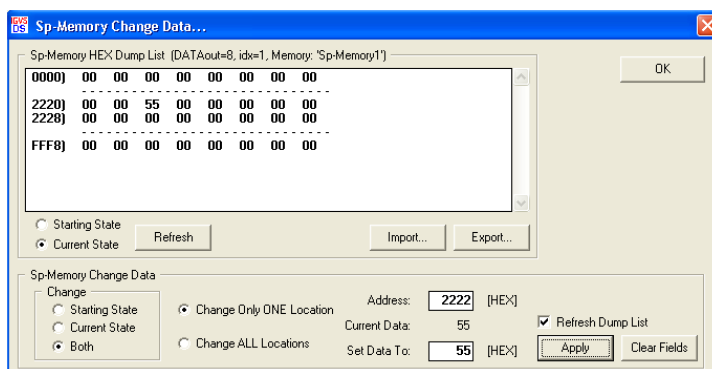
invertuju

3.3.4.2.3. Memorijski moduli

Memorijski modul je realizovan kao modul sa standardnim kontrolnim ulazima, ulazima za adresu i podatak, kao i izlazom za podatak koji je pročitao. Ulazne linije za adrese i podatke imaju po 16 bita. Širine ulaznih i izlaznih linija za podatke mogu da se podešavaju sa dozvoljenim vrednostima broja linija na ulaznim i izlaznim linijama za podatke 8, 4, 2 i 1.

Skup funkcija za rad sa sadržajem memorijskih modula (*Object/Memory Module*), obezbeđuje funkcije:

- *HEX Dump List...* – funkcija za otvaranje dijaloga za pregled sadržaja memorijskih lokacija memorijskog modula. Dijalog ima izgled kao gornji deo dijaloga prikazanog na slici 14. Taster *Change* (taster *Refresh* na dijalogu prikazanom na slici 14) transformiše *HEX Dump List...* dijalog u *Change Data...* dijalog.
- *Change Data...* – funkcija za otvaranje dijaloga za pregled i modifikaciju sadržaja memorijskih lokacija memorijskog modula.



Slika 38: Funkcija *Memory Module/Change Data...*

Postoji mogućnost da se promeni sadržaj samo jedne ili svih memorijskih lokacija.

3.3.4.3. Funkcija *Expand Connectors...*

Funkcija *Expand Connectors...* omogućava promenu širine signala na konektorima od 1 do 32 bita, u koracima po 1 bit. Funkcija može da se koristi proizvoljan broj puta za svako kolo koje nema povezane signale na svojim konektorima, bez obzira da li su postojali povezani signali koji su obrisani.

Postoje dve grupe kola i modula na kojima ova funkcija drugacije vrši modifikaciju širina signala. Jednoj grupi kola funkcija *Expand Connectors...* menja sve širine konektora, dok kod druge grupe menja širinu samo nekih konektora. Time se postiže velika raznovrsnost kola koji su dostupni iz osnovne biblioteke.

3.3.4.4. Pregled tipova signala i transformacija signala

U zavisnosti od načina kreiranja signala, korišćenja funkcija za povezivanje kola, modula i signala, kao i upotrebe funkcija za transformaciju tipova signala, signali mogu da pripadaju jednom od sledećih tipova signala:

- *Undefined* – signal koji nema izvor vrednosti (konektor kola ili signal iz modula)
- *Allocated* – signal koji ima izvor vrednosti (konektor kola ili signal iz modula)

- *BUS* – magistrala
- *External/Internal BUS* – poseban tip magistrale koja se koristi pri povezivanju magistrala koje se nalaze na različitim modulima i podmodulima
- *Clone* – signal čija vrednost je identična vrednosti nekog drugog signala (*Master Signal*)
- *PartClone* – signal čiji pojedini bitovi dobijaju vrednosti identične vrednostima bitova drugih signala (*Master Signal*)

Kao mogući izvori vrednosti za *Undefined* i *Allocated* signale označeni su signali iz modula.

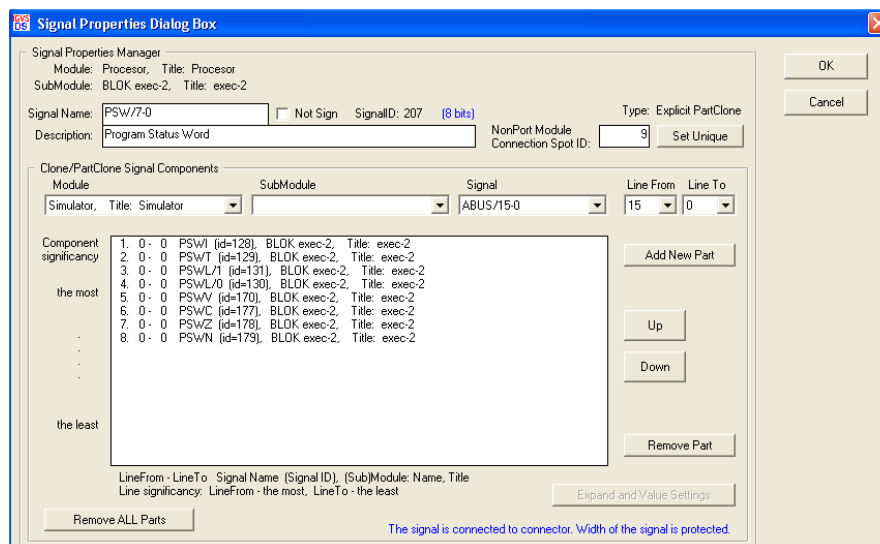
Undefined signali imaju najmanje jedan slobodan kraj na podlozi za prikaz strukture, dok drugi kraj može biti slobodan ili povezan na ulazni konektor kola ili ulazni port modula. *Allocated* signali imaju jedan kraj povezan na izlazni konektor kola ili izlazni port modula. *BUS* signal ne može kao izvor vrednosti da ima izlazni konektor proizvoljnog kola ili izlazni port modula, već kao izvor vrednosti može da ima isključivo izlazni signal *Sp-TriState* kola. *External/Internal BUS* signali se koriste isključivo pri povezivanju magistrala u jedinstvenu magistralu koja se prostire na više modula i podmodula. *Clone* i *PartClone* signali kao izvore vrednosti imaju sve bitove jednog signala (*Clone*) ili pojedine bitove koji ne moraju da pripadaju samo jednom signalu (*PartClone*). Signali čiji bitovi postaju komponente *Clone* ili *PartClone* signala imaju status *Master Signal*, ali ovakav status ne prevodi signal u posebnu vrstu signala.

Pored osnovnih tipova signala, na raspolaganju je i poseban skup tipova signala za potrebe povezivanja signala i *Port Module* objekata. Na raspolaganju su *IN Port* i *OUT Port* signali, za realizaciju ulaznih i izlaznih portova *Port Module* objekata, respektivno. *IN Port* signal može da se kreira korišćenjem *Undefined* signala, dok *OUT Port* može da se kreira korišćenjem *Allocated*, *Clone* i *PartClone* signala.

3.3.4.4.1. *Clone/PartClone* signali

Verovatno najznačajnija funkcija softverskog paketa *IGoVSoDS* je upravo mogućnost dodele vrednosti signalima na osnovu vrednosti drugih signala. Realizacijom ove funkcije, obezbeđena je mogućnost kreiranja signala sastavljenih od proizvoljnih delova svih vidljivih signala. Ukoliko zavisni signal ima samo jednu komponentu u vidu svih linija jednog signala, dobija status *Clone Signal*, dok u svim ostalim slučajevima dobija status *PartClone Signal*. Signali čije se linije koriste kao komponente u kreiranju *Clone* ili *PartClone* signala dobijaju status *Master Signal*. Signali u statusu *Clone* i *PartClone* mogu da budu *Master* signali za druge *Clone* i *PartClone* signale.

Ova funkcija praktično omogućava da jedan signal iz jednog modula, čija je širina recimo 32 bita, ima vrednost sastavljenu od 32 različitih signala širine jednog bita koji se nalaze u različitim modulima. Na slici 39 je prikazan jedan takav slučaj gde se vrednost jednog signala sastoji od vrednosti 8 različitih signala.



Slika 39: Primer realizacije *PartClone* signala

3.3.4.4.2 Magistrala

Magistrala (*BUS Signal*) je posebna vrsta signala koja se koristi za povezivanje više drugih signala. Za magistrale se primenjuju posebna pravila za kreiranje, promenu broja bitova, povezivanje drugih signala, kao i povezivanje magistrale sa kolima i modulima.

Za ostvarivanje funkcije povezivanja signala na magistralu, u biblioteci kola je obezbeđen specijalni modul *Sp-TriState*. Isključivo izlazni signal specijalnog modula *Sp-TriState* može da se poveže na magistralu.

Magistrala može da se povezuje na ulazne konektore kola i ulazne portove *Port Module* objekata, kao i svaki drugi signal. Posebne osobine koje ima magistrala, u odnosu na ostale signale, ne prenose se na signal koje se nalazi iza ulaznog porta. Isto tako, magistrala može da se povezuje sa signalima unutar *NonPort Module* objekta, i to na dva načina. Prvi način je da se magistrala poveže kao i svaki drugi signal, čime se posebne osobine magistrale ne prenose na signal unutar *NonPort Module* objekta (*Internal NonBUS signal*). Drugi način je da se magistrala poveže sa magistralom unutar *NonPort Module* objekta (*Internal BUS signal*), čime se dve magistrale integrišu u jedinstvenu magistralu koja se prostire na više modula i podmodula.

3.3.4.5 Povezivanje signala i modula

Povezivanje signala i modula je jedna od najznačajnijih funkcija softverskog paketa *IGoVSoDS*. Ovom funkcijom omogućava se razvoj hijerarhijske organizacije modula aktivne digitalne strukture. Realizovane su dve vrste modula: moduli sa portovima (*Port Module*) i moduli bez portova (*NonPort Module*).

Port Module objekti imaju definisane ulazne i izlazne portove, za povezivanje ulaznih i izlaznih signala, respektivno. Nijedan signal iz *Port Module* objekta nije vidljiv za nadređene module. Isto tako, nijedan signal van *Port Module* objekta nije vidljiv unutar tog objekta. Stoga, povezivanje signala i *Port Module* objekta može da se ostvari jedino povezivanjem signala na ulazne i izlazne portove.

NonPort Module objekti nemaju definisane portove, i nije moguće da se signali i moduli povezuju preko unapred određenih tačaka za povezivanje. Stoga je dopušteno da se signal van modula poveže sa bilo kojim signalom unutar modula, uz poštovanje pravila za

povezivanje odgovarajućih tipova signala. Svaki signal van modula koji ima izvor vrednosti (*Allocated*, *Clone* i *PartClone*), kao i svaka magistrala van modula (*BUS*), može da se poveže sa svakim signalom unutar modula, koji nema izvor vrednosti (*Undefined*). Isto tako, svaka magistrala van modula (*BUS*) može da se poveže sa svakom magistralom unutar modula (*BUS*), i na taj način može da se kreira magistrala koja se prostire na više modula. Osnovni uslov za povezivanje je da signali koji se povezuju imaju jednak broj bitova. Signali *Port Module* objekata, podređenih *NonPort Module* objektu sa kojim treba povezati signal, ne mogu da se koriste za povezivanje, saglasno pravilima za vidljivost signala koji pripadaju *Port Module* objektima.

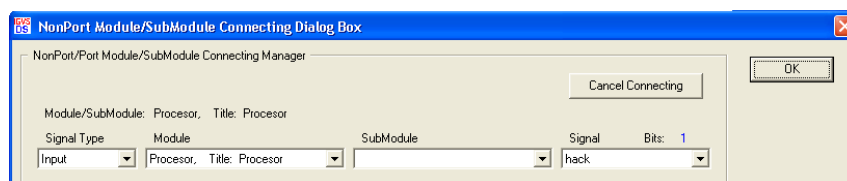
Povezivanje signala i modula je realizovano kroz dva postupka:

- povezivanje signala i modula sa portovima (*Port Module*)
- povezivanje signala i modula bez portova (*NonPort Module*)

Kreiranje linija signala koje će povezivati ulazne i izlazne portove *Port Module* objekta, obavlja se na identičan način kao i za konektore kola i specijalnih modula. Jedino ograničenje u odnosu na konektore je da nije moguće kreirati kružić za negaciju signala na portu.

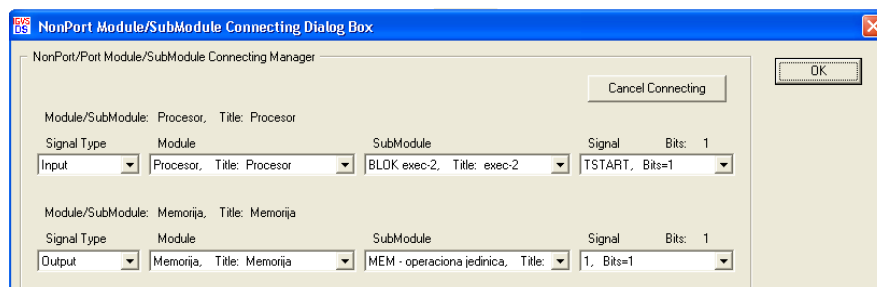
NonPort Module objekti nemaju unapred definisane portove za povezivanje signala. Stoga je omogućeno da se ostvari veza svakog signala van modula sa odgovarajućim signalom unutar modula, koji je vidljiv van modula.

Prilikom povezivanja nekog signala *Port Module*-a i *NonPort Module*-a otvara se dijalog prikazan na slici 40. U takvom dijalogu je moguće izabrati jedan signal iz *NonPort Module*-a koji će se povezati sa ulaznim ili izlaznim portom *Port Module*-a, do koga je dovučena linija signala. Ukoliko korisnik proba da poveže dva ulazna ili dva izlazna signala, greška će biti prijavljena.



Slika 40: Funkcija za povezivanje signala i *NonPort Module*-a

Prilikom povezivanja signala između dva *NonPort Module*-a otvara se dijalog prikazan na slici 41. U takvom dijalogu je moguće izabrati jedan izlazni signal iz jednog *NonPort Module*-a koji će se povezati samo sa ulaznim signalom drugog *NonPort Module*-a. Ukoliko korisnik proba da poveže dva ulazna ili dva izlazna signala, greška će biti prijavljena.



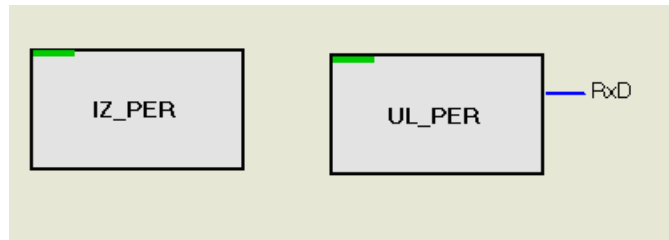
Slika 41: Funkcija za povezivanja signala dva *NonPort Module*-a

Za potrebe povezivanja magistrala i signala unutar *NonPort Module*-a, na raspolaganju su dve vrste povezivanja. Određuje se da li se magistrala povezuje bez

prenošenja osobina magistrale na signal unutar *NonPort Module*-a (*Internal NonBUS signal*), ili se magistrala povezuje sa prenošenjem osobina magistrale na signal unutar *NonPort Module*-a (*Internal BUS signal*). Nakon izbora vrste povezivanja, otvara se dijalog prikazan na slici 16, sa listama odgovarajućih signala unutar *NonPort Module*-a. Ukoliko je odabrana vrsta povezivanja *Internal NonBUS signal*, postavlja se *Signal Type=Input*, i prikazuje se lista *Undefined* signala unutar *NonPort Module*-a. Na ovaj način, magistrala postaje izvor vrednosti za *Undefined* signala unutar *NonPort Module*-a. Ukoliko je odabrana vrsta povezivanja *Internal BUS signal*, postavlja se *Signal Type=Output*, i prikazuje se lista *BUS* signala unutar *NonPort Module*-a. Nakon povezivanja, magistrala unutar *NonPort Module*-a postaje sastavni deo magistrale koja se povezuje na *NonPort Module*. Magistrala van *NonPort Module*-a dobija dodatni status *External BUS*, dok magistrala unutar *NonPort Module*-a dobija dodatni status *Internal BUS*. Obe magistrale se prikazuju i ponašaju kao da postoji samo *External BUS* magistrala.

4 PERIFERIJA

U ovoj glavi se daje pregled dela simulatora koji se odnosi na periferiju. Simulator je izrađen prema projektovanom sistemu opisanom u glavama 3 i 4 i povezan je sa simulatorom ostatka sistema već projektovanog računarskog sistema prikazanog u [1] uz saglasnost autora.

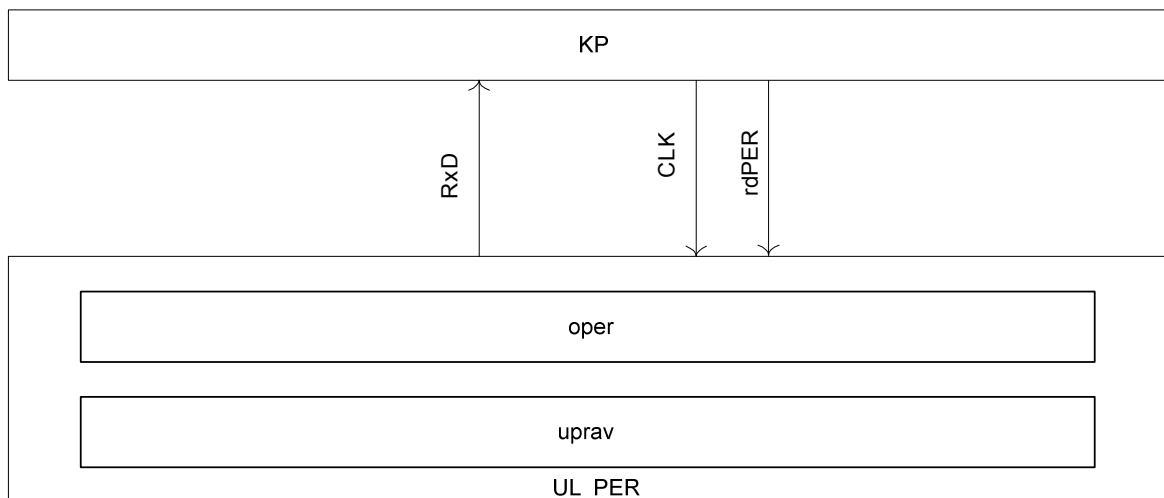


Slika 42 organizacija periferije

Na slici 42 prikazana je organizacija periferije koja se sastoji od ulazne periferije *UL_PER* i izlazne periferije *IZ_PER*.

4.1 ULAZNA PERIFERIJA

Ulazna periferija *UL_PER* (slika 10) se sastoji iz operacione jedinice *oper* i upravljačke jedinice *uprav*.



Slika 10 Organizacija ulazne periferije *UL_PER*

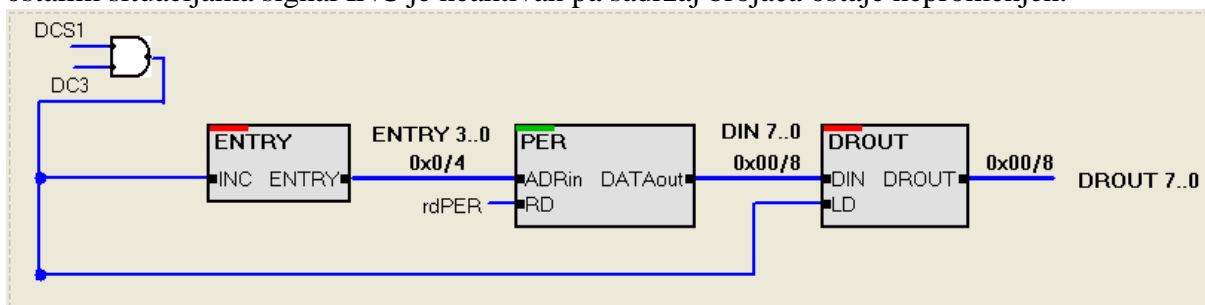
Operaciona jedinica *oper* je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova. Upravljačka jedinica *uprav* je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala na osnovu algoritma generisanja upravljačkih signala operacione jedinice i vrednosti signala logičkih uslova.

Struktura i opis operacione i upravljačke jedinice se daju u daljem tekstu.

4.1.1 Operaciona jedinica

Operaciona jedinica *oper* sadrži brojač $ENTRY_{4...0}$, prihvatni registar podatka $DROUT_{7...0}$ i memoriju PER (slika 11).

Brojač $ENTRY_{3..0}$ je 4-to razredni brojač čiji se sadržaj koristi za adresiranje memorijskih lokacija memorije PER prilikom čitanja iz periferije. U zavisnosti od vrednosti signala na ulazu INC brojača sadržaj brojača se može inkrementirati ili može ostati nepromenjen. Ova dva režima rada brojača se realizuju generisanjem aktivne vrednosti na ulazu INC onda kada sadržaj brojača treba inkrementirati i neaktivne vrednosti na ulazu INC onda kada sadržaj brojača treba da ostane nepromenjen. Prilikom svakog čitanja iz memorije PER periferije generisanjem aktivne vrednosti signala INC sadržaj brojača se inkrementira da bi ukazivao na sledeću memorijsku lokaciju memorije PER iz koje treba pročitati sledeći podatak. U svim ostalim situacijama signal INC je neaktivan pa sadržaj brojača ostaje nepromenjen.



Slika 11 Operaciona jedinica *oper*

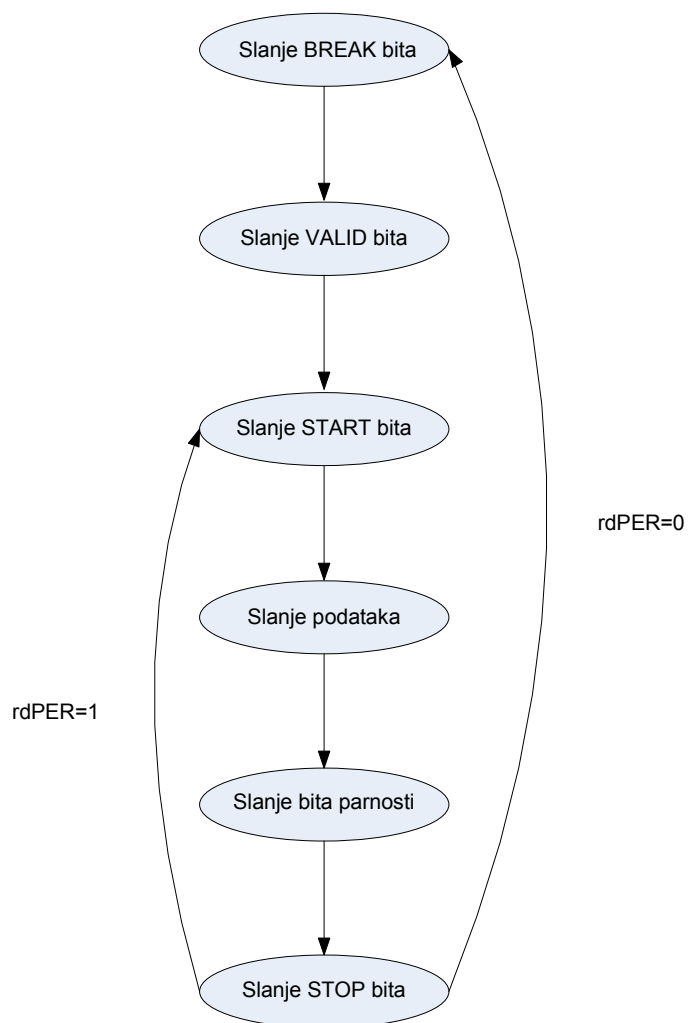
Memorija PER služi za simulaciji periferije kapaciteta 2^4 8-mo bitnih reči. Sadržaj memorijske lokacije adresirane sadržajem brojača $ENTRY_{3..0}$ se pojavljuje na izlaznim linijama podataka $DOUT_{7..0}$ memorije PER jedino ukoliko signal **rdPER** kontrolera KP ima aktivnu vrednost. Ovaj sadržaj se upisuje u prihvatni registar podatka $DROUT_{7..0}$. Sadržaj ovog registra se vodi na ulazne linije podataka $DROUT_{7..0}$ upravljačke jedinice ulazne periferije periferije *UL_PER*.

4.1.2 Upravljačka jedinica

U ovom odeljku se daju dijagram stanja ulazne periferije, dijagram toka operacija, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice *uprav*.

4.1.2.1 Dijagram stanja ulazne periferije

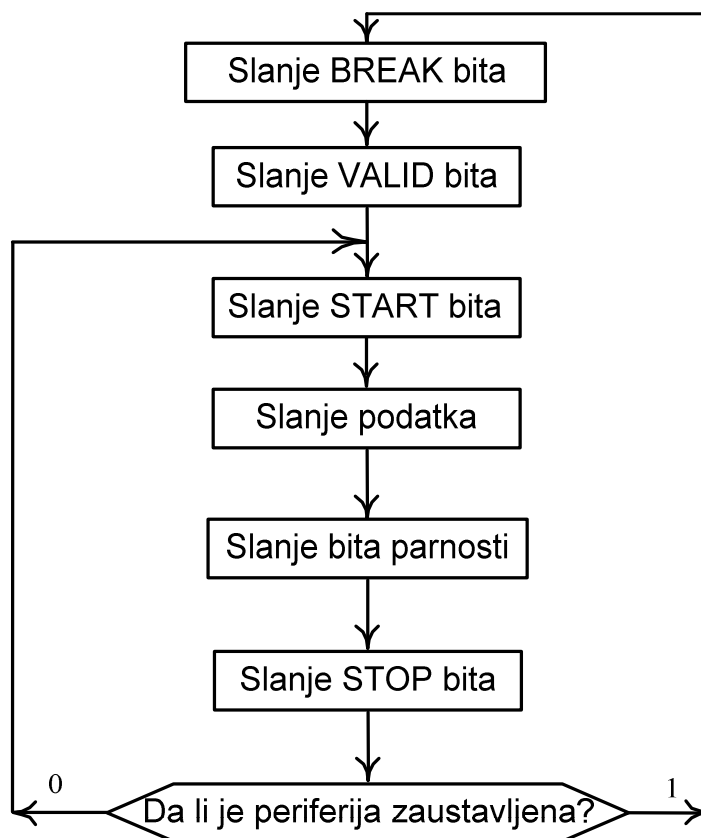
Protokol za slanje po liniji *RxD* predstavljen je dijagramom stanja na slici 12. Kada ulazna periferija nije startovana linija *RxD* ima neaktivnu vrednost. Kada se startuje ulazna periferija prvo se ostaje u stanju *Slanje BREAK bita* četiri perioda signala takta i za to vreme se šalje bit **BREAK** koji je predstavljen logičkom nulom u trajanju od četiri perioda signala takta. Zatim se prelazi u stanje *Slanje VALID bita* u kome se šalje bit **VALID** koji je predstavljen logičkom jedinicom u trajanju od četiri perioda signala takta. Posle ovoga se prelazi u stanje *Slanje START bita* u kome se šalje bit **START** koji je predstavljen logičkom nulom u trajanju od četiri perioda signala takta. Nakon toga se prelazi u stanje *Slanje podataka* u kome se šalje bit po bit podatka s tim da se vrednost svakog bita drži na liniji za slanje *RxD* u trajanju od četiri perioda signala takta. Kada se pošalje svih osam bitova podatak prelazi se u stanje *Slanje bita parnosti* u kome se šalje bit parnosti u trajanju od četiri perioda signala takta. Kada je poslat bit **PARNOSTI** prelazi se u stanje *Slanje STOP bita* u kome se šalje **STOP** bit koji je predstavljen logičkom jedinicom u trajanju od četiri perioda signala takta. Ako je periferija zaustavljena neaktivnom vrednošću signala **rdPER** prelazi se u stanje *Slanje BREAK bita* i odatle prenos nastavlja na već opisani način. U suprotnom prelazi se u stanje *Slanje START bita* i dalji prenos nastavlja na već opisani način.



Slika 12 Dijagram stanja ulazne periferije

4.1.2.2 Dijagram toka operacija

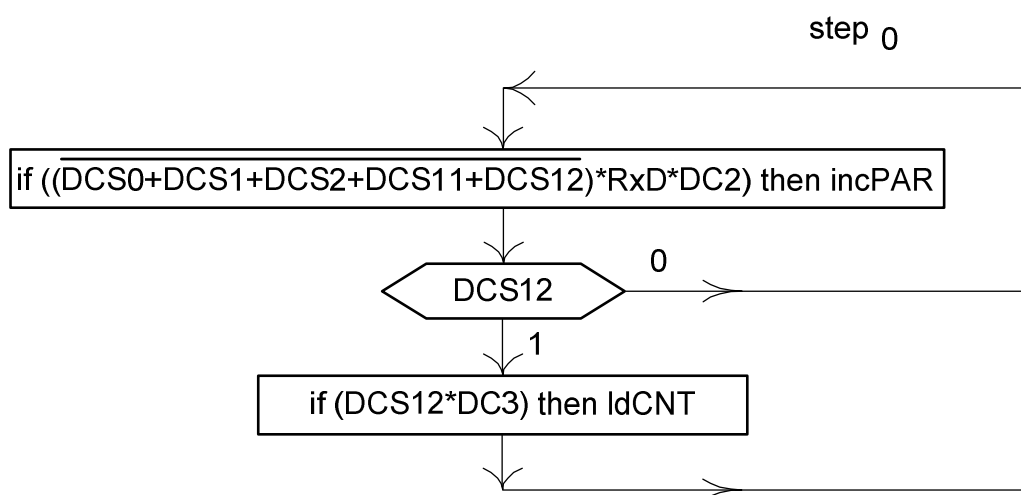
Dijagram toka operacija je predstavljen operacionim i uslovnim blokovima (slika 13) i u potpunosti odgovara dijagramu stanja ulazne periferije (slika 12). U operacionim blokovima se nalaze opisi mikrooperacija koje treba realizovati. U uslovnim blokovima se nalaze opisi logičkih uslova koji definišu grananja algoritma.



Slika 13 Dijagram toka operacija

4.1.2.3 Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu dijagrama toka operacija (slika 13) i dat u obliku dijagrama toka upravljačkih signala (slika 14) i sekvence upravljačkih signala (tabela 12).



Slika 10 Dijagram toka upravljačkih signala

Dijagram toka upravljačkih signala je predstavljen operacionim i uslovnim blokovima (slika 10). U operacionom bloku se nalazi upravljački signal i uslov pod kojima se on generiše. U uslovnim blokovima se nalaze signali logičkih uslova.

U sekvenci upravljačkih signala po koracima se koristi iskaz za signal. Iskaz za signal je oblika

if uslov then signal.

Ovaj iskaz sadrži uslov i upravljački signal kontrolera periferije *KP* i operacione jedinice *oper* i određuje pod kojim uslovima signal treba da bude generisan.

Objašnjenja vezana za generisanje upravljačkih signala su data zajednički za dijagram toka upravljačkih signala (slika 10) i sekvencu upravljačkih signala (tabela 12) i to u okviru sekvence upravljačkih signala.

Tabela 12 Sekvenca upravljačkih signala

! U koraku $step_0$ se ostaje sve vreme. U ovom koraku se generiše aktivna vrednost signala **incPAR** upravljačke jedinice *uprav* trajanja jedna perioda signala takta za vreme slanja svakog pojedinog bita podatka ako se šalje podatak koji je logička jedinica. Po slanju svih osam bita podataka generise se aktivna vrednost signala **ldCNT** kojim se ponovo prelazi na slanje sledećeg podatka. U ovom koraku se ne generiše aktivna vrednost signala **incPAR** sve dok je signal **rdPER** kontrolera neaktivan. Signal **rdPER** postaje aktivan kada kontroler uđe u stanje u kome treba da realizuje čitanje iz periferije i ostaje aktivan sve dok se čitanje ne završi i pročitani podatak prebaci iz periferije u kontroler. Aktivna vrednost ovog signala startuje čitanje iz memorije PER operacione jedinice. Zbog toga se pri aktivnoj vrednosti signala **rdPER** inkrementira brojač **BR** koji služi kao delitelj učestanosti za inkrementiranje brojača **CNT** koji služi za slanje bita na liniju **RxD** u skladu sa protokolom za slanje (slike 8 i 9) opisanim u poglavlju 4.1.2.1. Kada inkrementiranjem sadržaj brojača **CNT** dostigne vrednost 12 postaje aktivan signal **ldCNT**. Signalom **ldCNT** se u slučaju da periferija nije zaustavljena u brojač **CNT** upisuje vrednost 2. Ako je učitana vrednost 2 šalje se sledeći podatak na već opisani način. Ako je periferija zaustavljena ni brojač **CNT** ni brojač **BR** se ne inkrementiraju sve dok signal **rdPER** ne dobije aktivnu vrednost već se u njih upisuje vrednost 0.

- $step_0$ *if $((DCS0 + DCS1 + DCS2 + DCS11 + DCS12) * RxD * DC2)$ then **incPAR***
 *if $(DCS12 * DC3)$ then **ldCNT***

4.1.2.4 Struktura upravljačke jedinice

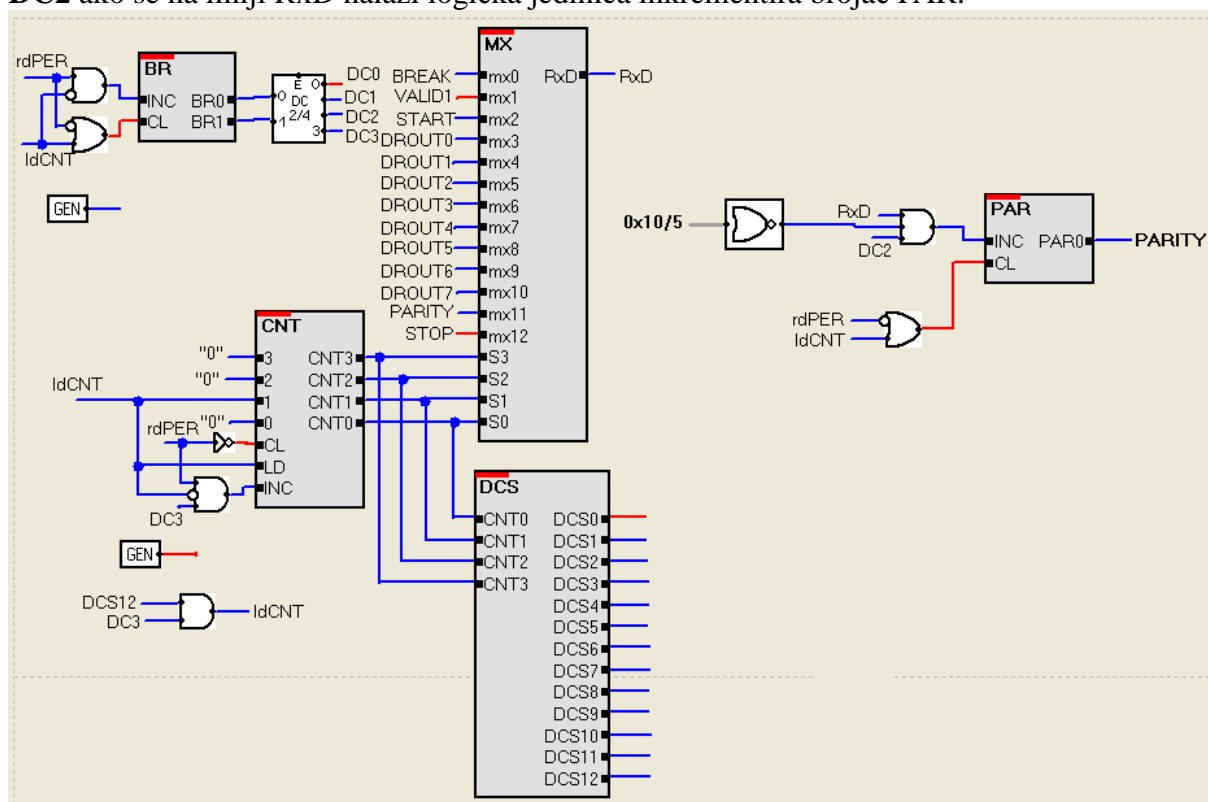
Struktura upravljačke jedinice je prikazana na slici 15. Upravljačka jedinica se sastoji od brojača **BR**_{1..0}, brojača **CNT**_{3..0}, brojača **PAR**_{3..0}, dekodera **DC**, dekodera **DC_S**, multipleksera **MX** i logičkih elemenata koji na osnovu signala **rdPER** kontrolera periferije *KP* generišu signale **ldCNT** i **incPAR** upravljačke jedinice *uprav* kao i signale **DC1**, **DC2**, **DCS0** i **DCS12** operacione jedinice *oper*.

Brojač **BR**_{1..0} je brojač po modulu četiri i koristi se kao delitelj učestanosti za brojač **CNT**_{3..0} da bi se brojač **CNT**_{3..0} inkrementirao na svaki četvrti signal takta **CLK**. Dok signal kontrolera periferije *KP* **rdPER** nije aktivan brojač **BR**_{1..0} ostaje u stanju 0. Dokle god je signal **rdPER** aktivan brojač **BR**_{1..0} broji po modulu četiri. Izlaz brojača **BR**_{1..0} se vodi na ulaze dekodera **DC**.

Dekoder **DC** dekodira izlaze brojača **BR**_{1..0} da bi generisao signale **DC0**, **DC1**, **DC2** i **DC3** potrebne za ostala kola u sistemu.

Brojač $CNT_{3..0}$ je brojač po modulu dvanaest i koristi se da selektuje jedan od ulaza multiplexera MX kojim se pušta odgovarajući sadržaj na liniju **RxD** u skladu sa protokolom za slanje opisanim u poglavlju 3.1.2.1. Ovaj brojač se inkrementira na aktivnu vrednost signala **DC3**.

Izlaz brojača $CNT_{3..0}$ se vodi na ulaze dekodera DCS koji služi za dekodiranje vrednosti brojača $CNT_{3..0}$ čime se dobija 12 stanja kroz koja prolazi ulazna periferija. Dok signal kontrolera periferije **KP rdPER** nije aktivan brojač $BR_{1..0}$ ostaje u stanju 0 (aktivan je signal **DC0** dekodera **DC**). Time brojač $CNT_{3..0}$ ostaje u stanju 0 (aktivan je signal **DCS0** dekodera **DCS**) a na liniju **RxD** se šalje logička nula (**BREAK**). Kad su aktivni signali **DCS0** i **DC2** iz operacione jedinice ulazne periferije **oper** učitava se 8-mo bitni podatak iz memorije ulazne periferije **PER** sa lokacije nula, od koje počinje čitanje iz ulazne periferije **UL_PER**, u registar **DROUT_{7..0}**. Kad je aktivan signal **DCS1** na liniju **RxD** se šalje logička jedinica koja predstavlja bit **VALID1**. Kad je aktivan signal **DCS2** na liniju **RxD** se šalje logička nula koja predstavlja bit **START**. Kad su aktivni signali **DCS3..DCS10** na liniju **RxD** se šalju biti podatka iz registra **DROUT** operacione jedinice **oper** za to vreme se na svaki aktivan signal **DC2** ako se na liniji **RxD** nalazi logička jedinica inkrementira brojač **PAR**.



Slika 15 Struktura upravljačke jedinice *uprav*

Brojač **PAR** se koristi za generisanje bita parnosti **PARITY**. Usvojeno je da se radi sa parnom parnosti što znači da će bit **PARITY** imati aktivnu vrednost samo u slučaju da je poslat neparan broj jedinica. Ovaj bit se generiše na sledeći način brojač **PAR** prebrojava poslate jedinice. Po obavljenom brojanju bit **PAR₀** predstavlja **PARITY** bit ako je on nula poslat je paran broja jedinica i vrednost **PARITY** bita je nula, u suprotnom poslat je neparan broj jedinica i **PARITY** bit ima vrednost jedan.

Po završenom slanju bita podataka šalje se **PARITY** bit kada je aktivan signal **DCS11** posle koga se šalje bit **STOP** kada je aktivan bit **DCS12**. Aktivnim vrednostima signala **DCS12** i **DC1** inkrementira se brojač $ENTRY_{3..0}$ čime se na njegovom izlazu dobija adresa

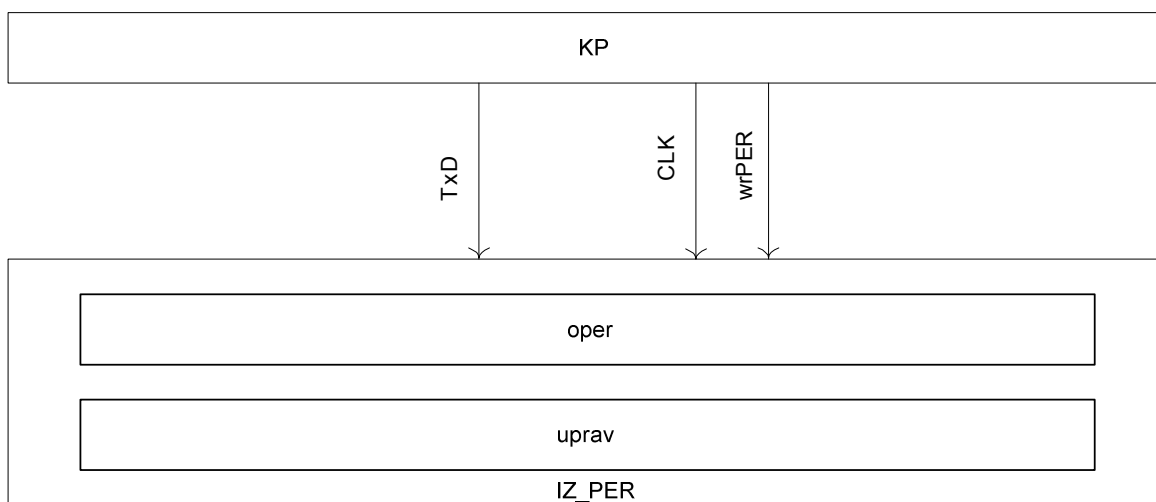
sledeće lokacije memorije periferije *PER* iz koje se čita podatak. Aktivnim vrednostima signala **DCS12** i **DC2** podatak se iz memorije periferije *PER* upisuje u registar DROUT_{8..0}. Aktivne vrednosti signala **DCS12** i **DC3** generišu aktivnu vrednost signala **ldCNT** kojom se u brojač *CNT* upisuje vrednost 2 ako periferija nije zaustavljena. Ako je periferija zaustavljena na izlazu dekodera DCS aktivan je signal **DCS0** koji na liniju RxD šalje vrednost 0 (BREAK) odatle se slanje podataka nastavlja na već opisani način.

Upravljački signali **ldCNT** i **incPAR** koji se koriste u upravljačkoj jedinici periferije *uprav*:

- **ldCNT** = DCS12*DC3
- **incPAR** = $\overline{\text{DCS0} + \text{DCS1} + \text{DCS2} + \text{DCS11} + \text{DCS12}} * \text{RxD} * \text{DC2}$

4.2 IZLAZNA PERIFERIJA

Izlazna periferija *IZ_PER* (slika 16) se sastoji iz operacione jedinice *oper* i upravljačke jedinice *uprav*.



Slika 16 Organizacija periferije *PER*

Operaciona jedinica *oper* je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova. Upravljačka jedinica *uprav* je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala na osnovu algoritma generisanja upravljačkih signala operacione jedinice i vrednosti signala logičkih uslova.

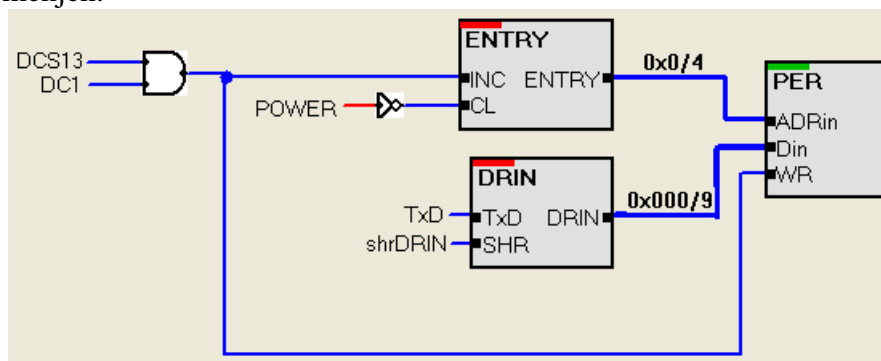
Struktura i opis operacione i upravljačke jedinice se daju u daljem tekstu.

4.2.1 Operaciona jedinica

Operaciona jedinica *oper* sadrži brojač ENTRY_{4...0} i memoriju PER (slika 17).

Brojač ENTRY_{3...0} je 4-to razredni brojač čiji se sadržaj koristi za adresiranje memorijskih lokacija memorije PER prilikom upisa u periferiju. U zavisnosti od vrednosti signala na ulazu INC brojača sadržaj brojača se može inkrementirati ili može ostati nepromenjen. Ova dva režima rada brojača se realizuju generisanjem aktivne vrednosti na ulazu INC onda kada sadržaj brojača treba inkrementirati i neaktivne vrednosti na ulazu INC onda kada sadržaj brojača treba da ostane nepromenjen. Pri neaktivnoj vrednosti signala **POWER** kada periferija nije uključena na ulazu CL će biti aktivna vrednost pa se tada u brojač upisuju sve nule. Ovo se koristi da se pri uključivanju periferije *PER* sadržaj brojača ENTRY_{3...0} dovede na početno stanje sve nule. Pri aktivnoj vrednosti signala **POWER** na ulazu CL će biti neaktivna

vrednost, dok će na ulazu INC biti aktivna vrednost. Ovo se koristi da se prilikom svakog upisa u memoriju PER periferije generisanjem aktivne vrednosti signala INC sadržaj brojača inkrementira da bi ukazivao na sledeću memorijsku lokaciju memorije PER u koju treba upisati sledeći podatak. U svim ostalim situacijama signal INC je neaktivan pa sadržaj brojača ostaje nepromenjen.



Slika 17 Operaciona jedinica *oper*

Memorija PER služi za simulaciji periferije kapaciteta 2^4 9-to bitnih reči. Najstariji bit ovih lokacija ima oznaku *P* i predstavlja bit parnosti primljen od kontrolera periferije *KP*. Ovaj bit se postavlja pri upisu podatka u periferiju. Na memorijsku lokaciju adresiranu sadržajem brojača $ENTRY_{3...0}$ se upisuje sadržaj registra *DRIN*. Registar *DRIN* je pomerački registar sa pomeranjem u desno koji prihvata podatak bit po bit sa linije *TxD* koja se dovodi iz upravljačke jedinice izlazne periferije *IZ_PER*. Kad se upiše svih 9 bitova u ovaj registar podatak je primljen i upisuje se u memorijsku lokaciju adresiranu sadržajem brojača $ENTRY_{3...0}$.

4.2.2 Upravljačka jedinica

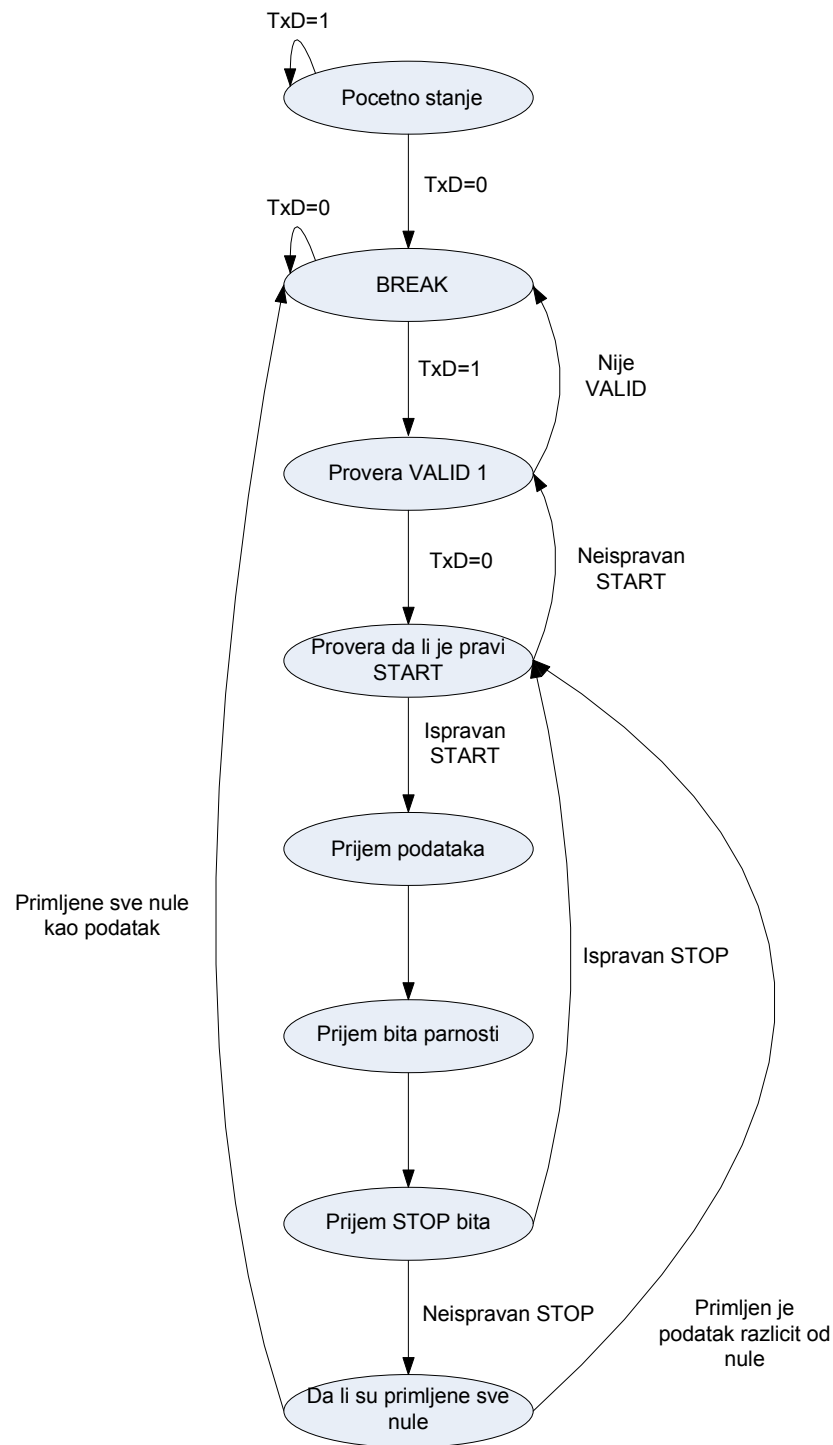
U ovom odeljku se daju dijagram stanja izlazne periferije, dijagram toka operacija, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice *uprav*.

4.2.2.1 Dijagram stanja izlazne periferije

Protokol za prijem po liniji *TxD* predstavljen je dijagramom stanja na slici 18. Usvojeno je da slanje svakog bita traje 4 perioda signala takta. Prijem bita se obavlja tako što se vrednost proverava na sredini intervala. Kada izlazna periferija nije startovana vrednost linije *TxD* nema uticaja na rad periferije.

Kada se startuje izlazna periferija prvo se ostaje u stanju *Pocetno stanje* sve dok se na liniji *TxD* ne pojavi logička jedinica i tada se prelazi u stanje *BREAK*. Da bi se iz stanja *BREAK* prešlo u sledeće stanje *Provera VALID 1* mora se na liniji *TxD* prvo detektovati jedinica i ponovo proveriti ta vrednost na sredini intervala prijema. Ako se ponovi vrednost 1 i na sredini intervala prijema prelazi se u stanje *Provera VALID 1*; u suprotnom se vraća u stanje *BREAK*. Da bi se iz stanja *Provera VALID 1* prešlo u stanje *Provera START* mora se na liniji *TxD* prvo detektovati nula i ponovo proveriti ta vrednost na sredini intervala prijema. Ako se ponovi vrednost 0 i na sredini intervala prijema prelazi se u stanje *Prijem podataka*; u suprotnom se vraća u stanje *Provera VALID 1*. U stanju *Prijem podataka* ostaje se dok se ne primi osam bita podataka, nakon toga se prelazi u stanje *Prijem bita parnosti*. Kad se primi bit parnosti prelazi se u stanje *Prijem STOP bita*. Ako je primljen ispravan bit STOP prelazi se u stanje *Provera START* i prijem se nastavlja na već opisani način. Ako nije primljen ispravan STOP prelazi se u stanje *Da li su primljene sve nule* u kome se proverava da li su kao podatak primljene sve nule. Ako je to slučaj prelazi se u stanje *BREAK* odakle se prijem nastavlja na

već opisani način. U suprotnom se prelazi u stanje *Provera START* odakle se prijem nastavlja na već opisani način.

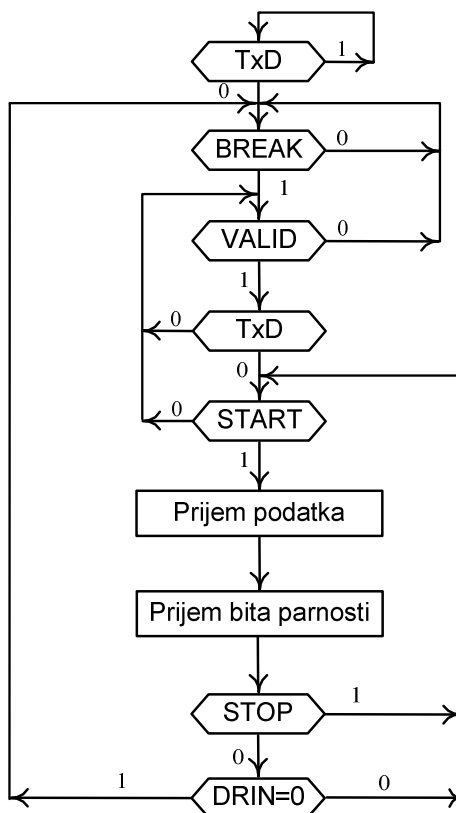


Slika 18 Dijagram stanja izlazne periferije

4.2.2.2 Dijagram toka operacija

Dijagram toka operacija je predstavljen operacionim i uslovnim blokovima (slika 19) i u potpunosti odgovara dijagramu stanja ulazne periferije (slika 18). U operacionim blokovima

se nalaze opisi mikrooperacija koje treba realizovati. U uslovnim blokovima se nalaze opisi logičkih uslova koji definišu grananja algoritma.



Slika 19 Dijagram toka operacija

4.2.2.3 Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu dijagrama toka operacija (slika 19) i dat u obliku dijagrama toka upravljačkih signala (slika 20) i sekvence upravljačkih signala (tabela 13).

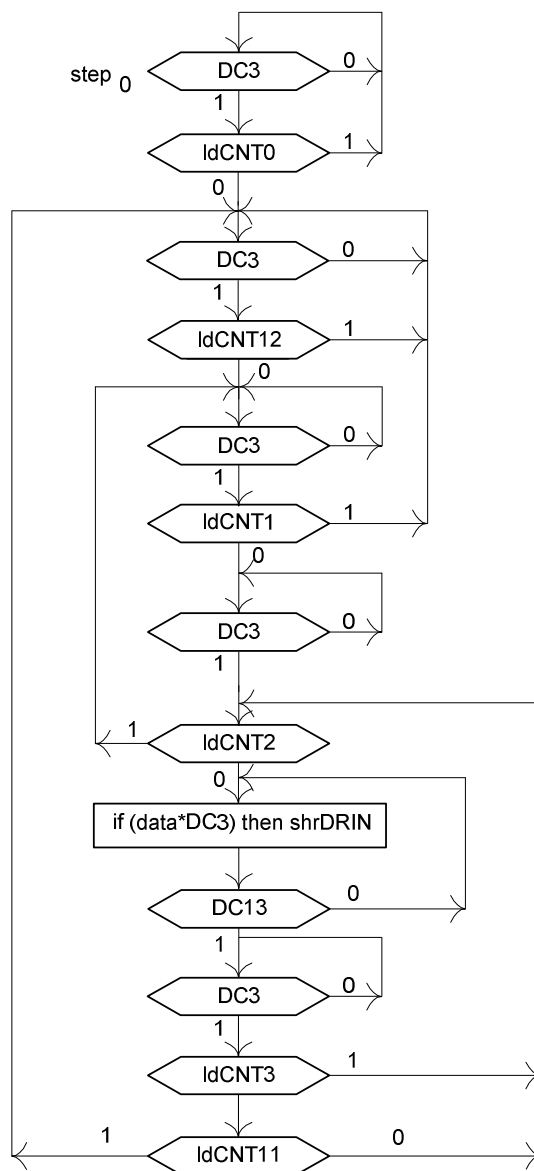
Dijagram toka upravljačkih signala je predstavljen operacionim i uslovnim blokovima (slika 20). U operacionom bloku se nalazi upravljački signal i uslov pod kojim se on generiše. U uslovnim blokovima se nalaze signali logičkih uslova.

U sekvenci upravljačkih signala po koracima se koristi iskaz za signal. Iskaz za signal je oblika

if uslov then signal.

Ovaj iskaz sadrži uslov i upravljački signal kontrolera periferije *KP* i operacione jedinice *oper* i određuje pod kojim uslovima signal treba da bude generisan.

Objašnjenja vezana za generisanje upravljačkih signala su data zajednički za dijagram toka upravljačkih signala (slika 20) i sekvencu upravljačkih signala (tabela 13) i to u okviru sekvence upravljačkih signala.



Slika 20 Dijagram toka upravljačkih signala

Tabela 13 Sekvenca upravljačkih signala

! U koraku $step_0$ se ostaje sve vreme. U ovom koraku se ne generiše ni jedan signal dok nije aktivan signal **START_IZ** periferije. Signal **START_IZ** postaje aktivan kada se startuje periferija. Aktivna vrednost ovog signala startuje upis u memoriju PER operacione jedinice. Zbog toga se pri aktivnoj vrednosti signala **START_IZ** inkrementira brojač **BR** koji služi kao delitelj učestanosti za inkrementiranje brojača **CNT** čiji se izlazi vode na ulaz dekodera **DCS** koji generiše signale potrebne za prijem bita na liniji **TxD** u skladu sa protokolom za prijem (slike 18 i 19) opisanim u poglavlju 3.2.2.1. Kada inkrementiranjem sadržaj brojača **BR** dostigne vrednost **3** postaje aktivan signal **DC3** tada se u slučaju da nije aktivan signal **IdCNT** inkrementira brojač **CNT**. Ako je aktivan signal **IdCNT** brojač **CNT** se ne inkrementira već učitava novu vrednost sa svog ulaza u zavisnosti od toga u koje stanje treba da pređe. Signal **IdCNT** dobija aktivnu vrednost u slučaju da treba odstupiti od inkrementiranja brojača **CNT** i generiše se kao ili funkcija signala **IdCNT0**, **IdCNT1**, **IdCNT2**, **IdCNT3**, **IdCNT11** i **IdCNT12**. Signal **IdCNT0** dobija aktivnu vrednost ako brojač **CNT** treba da ostane u stanju *Početno stanje* jer nije primljen bit **TxD** sa vrednošću 0 za prelazak u stanje *BREAK*. Signal

ldCNT1 dobija aktivnu vrednost ako brojač **CNT** treba da se vrati u stanje *BREAK* iz stanja *Provera VALID 1* jer je primljen neispravan bit **VALID** u skladu sa protokolom za prijem opisanim u poglavlju 3.2.2.1. Signal **ldCNT2** dobija aktivnu vrednost ako brojač **CNT** treba da se vrati u stanje *Provera VALID 1* iz stanja *Provera START* jer je primljen neispravan bit **START** u skladu sa protokolom za prijem opisanim u poglavlju 3.2.2.1. Signal **ldCNT3** dobija aktivnu vrednost ako brojač **CNT** treba da se vrati u stanje *Provera START* u skladu sa protokolom za prijem opisanim u poglavlju 3.2.2.1. u slučaju da je po prijemu podatka ispravno primljen **STOP** bit ili nije ispravno primljen **STOP** a primljen je podatak različit od nule. Signal **ldCNT11** dobija aktivnu vrednost ako brojač **CNT** treba da se vrati u stanje *BREAK* u skladu sa protokolom za prijem opisanim u poglavlju 3.2.2.1. u slučaju da je po prijemu podatka neispravno primljen bit **STOP** i ako su kao podatak primljene sve nule. Signal **ldCNT12** dobija aktivnu vrednost ako brojač **CNT** treba da ostane u stanju *BREAK* jer nije primljen bit **TxD** sa vrednošću 0 za prelazak u stanje *Provera START*. Signal **ldDRIN** dobija aktivnu vrednost po prijemu svakog bita podatka da bi se u prihvatni pomerački registar podatka **DRIN** operacione jedinice izlazne periferije *oper* upisao bit po bit podatka koji se prima. Signal **shrDRIN** dobija aktivnu vrednost po prijemu svakog bita podatka sem poslednjeg da bi se izvršilo pomeranje sadržaja prihvatnog pomeračkog registra **DRIN** operacione jedinice izlazne periferije *oper*. Posle ovoga se prijem nastavlja od stanja *Provera START* na već opisani način.

- step₀ *if* (data*DC3) *then shrDRIN*

4.2.2.4 Struktura upravljačke jedinice

Struktura upravljačke jedinice je prikazana na slici 21. Upravljačka jedinica se sastoji od brojača BR_{1..0}, brojača CNT_{3..0}, dekodera DC, dekodera DC_S, multipleksa MXCNT, RS flip-flopora, D flip-flopora i logičkih elemenata koji na osnovu signala **rdPER** kontrolera periferije *KP* generišu signal **ldCNT** upravljačke jedinice *uprav* kao i signale **DC1**, **DCS12**, **ldDRIN**, **shrDRIN** i **DR** operacione jedinice *oper*.

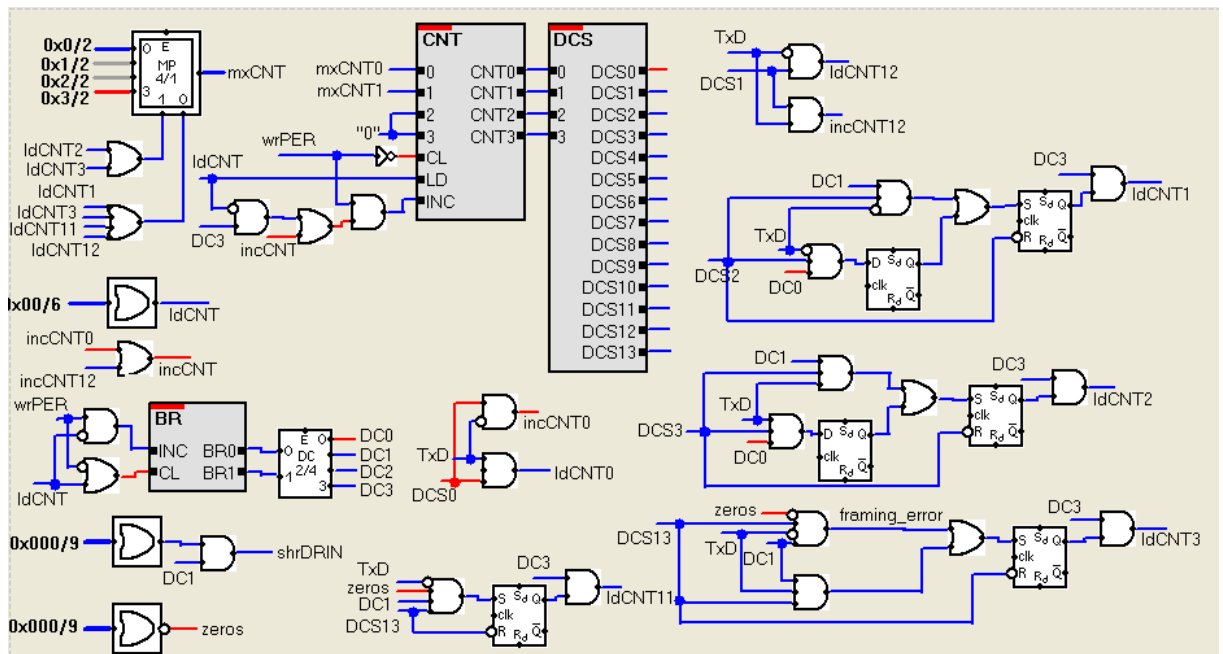
Brojač BR_{1..0} je brojač po modulu četiri i koristi se kao delitelj učestanosti za brojač CNT_{3..0} da bi se brojač CNT_{3..0} inkrementirao na svaki četvrti signal takta CLK. Dok signal kontrolera periferije *KP* **POWER** nije aktivan brojač BR_{1..0} ostaje u stanju 0. Dokle god je signal **POWER** aktivan brojač BR_{1..0} broji po modulu četiri. Izlaz brojača BR_{1..0} se vodi na ulaze dekodera DC.

Dekoder DC dekodira izlaze brojača BR_{1..0} da bi generisao signale **DC0**, **DC1**, **DC2** i **DC3** potrebne za ostala kola u sistemu. Na aktivnu vrednost signala **DC3** brojač CNT se ili inkrementira ili prelazi u neko od stanja.

Brojač CNT_{3..0} je brojač po modulu dvanaest. Izlaz brojača CNT_{3..0} se vodi na ulaze dekodera DCS koji služi za dekodiranje vrednosti brojača CNT_{3..0} čime se dobija 13 stanja kroz koja prolazi izlazna periferija. Dok signal **POWER** nije aktivan brojač BR_{1..0} ostaje u stanju 0 (aktivan je signal **DC0** dekodera **DC**). Time brojač CNT_{3..0} ostaje u stanju 0 (aktivan je signal **DCS0** dekodera **DCS**) a na liniju DR se ne šalje ništa (**Pocetno stanje**). Kad signal **POWER** postane aktivan brojač BR_{1..0} počinje sa brojanjem. Kad je aktivan signal **DC1** proverava se vrednost linije TxD i ako je njena vrednost 1 postavlja se aktivna vrednost signala **ldCNT0** čime se brojač CNT vraća u stanje 0 (**Pocetno stanje**) na aktivnu vrednost signala **DC3**. Ako je vrednost linije TxD 0 brojač se inkrementira i prelazi u stanje 1 (**BREAK**). U ovom stanju se na aktivnu vrednost signala **DC1** proverava vrednost linije TxD i ako je njena vrednost 0 postavlja se aktivna vrednost signala **ldCNT12** čime se brojač CNT vraća u stanje 1 (**BREAK**) na aktivnu vrednost signala **DC3**. Ako je vrednost linije TxD 0

brojač se inkrementira i prelazi u stanje 2 (**Provera VALID 1**). U ovom stanju se na aktivnu vrednost signala **DC0** proverava vrednost linije TxD i pamti njena vrednost u D flip flopu. Na aktivnu vrednost signala **DC1** proverava se vrednost linije TxD i poredi sa vrednošću sa izlaza D flip-flopa. Ako su obe vrednosti 1 brojač se inkrementira i prelazi u sledeće stanje 3 (**Provera START**). U suprotnom se postavlja aktivna vrednost signala **ldCNT1** čime se brojač CNT vraća u stanje 1 (**BREAK**) na aktivnu vrednost signala **DC3**. U stanju 3 (**Provera START**) se na aktivnu vrednost signala **DC0** proverava vrednost linije TxD i pamti njena vrednost u D flip flopu. Na aktivnu vrednost signala **DC1** proverava se vrednost linije TxD i poredi sa vrednošću sa izlaza D flip-flopa. Ako su obe vrednosti 0 brojač se inkrementira i prelazi u sledeće stanje 4 (**Prijem podataka**). U suprotnom se postavlja aktivna vrednost signala **ldCNT2** čime se brojač CNT vraća u stanje 2 (**Provera VALID 1**) na aktivnu vrednost signala **DC3**. U stanjima 4 do 12 (aktivne vrednosti jednog od signala DCS4, ..., DCS12) se na aktivnu vrednost signala **DC2** generiše aktivna vrednost signala **ldDRIN** čime se u prihvatni pomerački registar podatka **DRIN** upisuju bitovi podatka i bit parnosti. U stanjima 4 do 11 (aktivne vrednosti jednog od signala DCS4, ..., DCS11) se na aktivnu vrednost signala **DC3** generiše aktivna vrednost signala **shrDRIN** čime se vrši pomeranje sadržaja prihvatnog pomeračkog registra podatka **DRIN** za jedno mesto udesno. Na aktivne vrednosti signala **DCS13** i **DC3** inkrementira se vrednost brojača ENTRY i vrši se upis podatka u memoriju izlazne periferije PER iz registra DRIN na adresu na koju ukazuje vrednost brojača ENTRY. Inkrementirajne i istovremeni upis u memoriju periferije PER je moguće jer će vrednost brojača po inkrementiranju biti raspoloživa tek na sledeći signal takta kada je podatak već upisan u memoriju periferije PER čime se održava da se podaci prvo upisuju pa se tek onda inkrementira vrednost brojača ENTRY.

Multiplexer MXCNT se koristi da u zavisnosti od toga u koje stanje treba da pređe brojač CNT upiše odgovarajuću vrednost u taj brojač. Selekcija vrednosti koja će se upisati u brojač se obavlja aktivnim vrednostima signala **mxCNT0** i **mxCNT1**.



Slika 21 Struktura upravljačke jedinice *uprav*

Upravljački signali koji se koriste u upravljačkoj jedinici *uprav*:

- **ldCNT** = (ldCNT0+ldCNT1+ldCNT11+ldCNT12+ldCNT2+ldCNT3)*DC3
- **mxCNT0** = ldCNT1+ldCNT11+ldCNT12+ldCNT3
- **mxCNT1** = ldCNT2+ldCNT3

- $\text{zeros} = \overline{\text{DRIN0} + \text{DRIN1} + \text{DRIN2} + \text{DRIN3} + \text{DRIN4} + \text{DRIN5} + \text{DRIN6} + \text{DRIN7}}$
- $\text{shrDRIN} = (\text{DCS4} + \text{DCS5} + \text{DCS6} + \text{DCS7} + \text{DCS8} + \text{DCS9} + \text{DCS10} + \text{DCS11} + \text{DCS12}) * \text{DC3}$

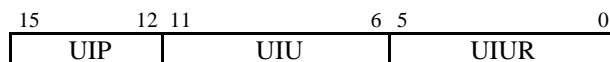
5 KONTROLER SA SERIJSKIM PRENOSOM

U ovoj glavi se razmatra arhitektura i organizacija kontrolera za serijski prenos podataka. Pri tome se pod arhitekturom sistema za serijski prenos podataka podrazumevaju svi oni njegovi elementi koji treba da budu poznati da bi za njega mogao da se napiše asemblerski program koji će se uspešno izvršavati i uvek davati isti rezultat bez obzira na to kako je dati sistem za serijski prenos podataka realizovan.

5.1 Arhitektura kontrolera periferije sa serijskim prenosom

Arhitekturu čine 8-mo bitni programski dostupni registri kontrolera ulazno/izlaznog uređaja kapaciteta 4K reči, koji zauzimaju opseg od F000h do FFFFh memorijskog adresnog prostora. Iz registara kontrolera se čita i u registre kontrolera se upisuje na identičan način na koji se čita iz memorijskih lokacija i upisuje u memorijske lokacije. Dozvoljeno je čitanje iz svih registara kontrolera i upisivanje u sve registre kontrolera. Adresa registra kontrolera se dobija na identičan način na koji se dobija adresa memorijske lokacije.

Adresiranje pojedinih registara kontrolera se obezbeđuje preko tri komponente koje formiraju adrese registara (slika 22): adresiranje ulazno-izlaznog adresnog prostora (UIP), adresiranje uređaja u ulazno-izlaznom adresnom prostoru (UIU) i adresiranje registara unutar adresiranog uređaja (UIUR).



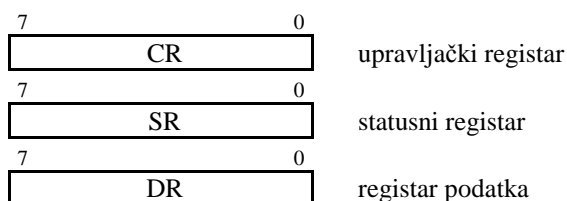
Slika 22 Formiranje adrese registara kontrolera periferije

Usvojeno je da je za ulazno-izlazni adresni prostor rezervisano najviših 2^{12} adresa te je sadržaj polja UIP jednak 1111b. Ova vrednost polja UIP je ista za sve ulazno-izlazne uređaje i njihove registre. Polje UIU je širine 6 bita, što omogućava adresiranje do 2^6 ulazno-izlaznih uređaja. Ova vrednost se postavlja mikroprekidačima pri povezivanju ulazno-izlaznog uređaja na računarski sistem. Poljem UIUR se adresira do 2^6 registara unutar ulazno-izlaznog uređaja. Ovakav format adresiranja registara ulazno-izlaznih uređaja je usvojen za sve uređaje.

U ovom odeljku se razmatraju programski dostupni registri kontrolera za serijski prenos podataka.

Registri kontrolera za serijski prenos podataka

Programski dostupni registri kontrolera su CR, SR i DR (slika 23).



Slika 23 Programski dostupni registri kontrolera za serijski prenos podataka

Registar CR je upravljački registar širine 8 bita od kojih se koriste samo najmlađa 3 bita. Ovaj registar se koristi da se programskim putem upisivanjem odgovarajućih vrednosti kontroler startuje i zadaje režim rada i da se kontroler zaustavlja. Bitovi registra CR su:

- CR₀ koji vrednošću
 - 0 ukazuje da je zadat režim rada sa izlaznom periferijom i
 - 1 ukazuje da je zadat režim rada sa ulaznom periferijom,
- CR₁ koji vrednošću
 - 0 ukazuje da zadat režim rada bez generisanja prekida i
 - 1 ukazuje da zadat režim rada sa generisanjem prekida i
- CR₂ koji vrednošću
 - 0 ukazuje da je kontroler zaustavljen i
 - 1 ukazuje da je kontroler startovan.

Bit CR₀ se prilikom startovanja kontrolera postavlja na neaktivnu vrednost da bi se zadao režim rada sa izlaznom periferijom i na aktivnu vrednost da bi se zadao režim rada sa ulaznom periferijom. Kada se zada režim rada sa izlaznom periferijom, programskim putem se podatak najpre prenosi iz memorije u registar DR kontrolera, a potom kontroler prenosi podatak iz registra DR u izlaznu periferiju. Kada se zada režim rada sa ulaznom periferijom, kontroler najpre prenosi podatak iz ulazne periferije u registar DR, a potom se podatak prenosi programskim putem iz registra DR u memoriju. Bit CR₀ se referiše kao bit *ulaz*.

Bit CR₁ se prilikom startovanja kontrolera postavlja na neaktivnu vrednost da bi se zadao režim rada bez generisanja prekida i na aktivnu vrednost da bi se zadao režim rada sa generisanjem prekida. Režim rada bez generisanja prekida se zadaje u slučaju organizacije ulaza/izlaza sa ispitivanjem bita SR₀ statusnog registra kontrolera. Režim rada sa generisanjem prekida se zadaje u slučaju organizacije ulaza/izlaza sa prekidom. Kada prilikom ulaza ili izlaza registar DR postane raspoloživ, na šta ukazuje aktivna vrednost bita SR₀ statusnog registra kontrolera, prekid se ne generiše ukoliko je bit CR₁ neaktivan i generiše ukoliko je bit CR₁ aktivan. Bit CR₁ se referiše kao bit *prekid*.

Bit CR₂ se programskim putem postavlja na aktivnu vrednost da bi se startovao kontroler i na neaktivnu vrednost da bi se zaustavio kontroler. Kontroler koji je startovan za režim rada sa ulaznom periferijom, prenosi podatke iz periferije u registar DR, a kontroler koji je startovan za režim rada sa izlaznim uređajem prenosi podatke iz registra DR u periferiju. Kontroler koji je zaustavljen prekida prenos podataka. Bit CR₂ se referiše kao bit *start*.

Registar SR je statusni registar širine 8 bita od kojih se koriste samo najmlađa 3 bita. Ovaj registar se koristi da se programskim putem čitanjem njegovog sadržaja dobijaju informacije o stanju u kome se nalazi kontroler. Bitovi statusnog registra je:

- SR₀ koji vrednošću
 - 0 ukazuje da registar DR nije raspoloživ i
 - 1 ukazuje da je registar DR raspoloživ,
- SR₁ koji vrednošću
 - 0 ukazuje da se nije desila greška na prijemu pri proveru bita parnosti i
 - 1 ukazuje da se desila greška na prijemu pri proveru bita parnosti,
- SR₂ koji vrednošću
 - 0 ukazuje da se nije desila greška na prijemu pri proveru stop bita i
 - 1 ukazuje da se desila greška na prijemu pri proveru stop bita.

Bit SR_0 se postavlja na aktivnu i neaktivnu vrednost po određenim pravilima u zavisnosti od toga da li je prilikom startovanja kontrolera zadat režim sa ulaznom ili izlaznom periferijom. Bit SR_0 se referiše kao bit *spremnost*.

Ukoliko je prilikom starovanja kontrolera u bit CR_0 upisana neaktivna vrednost, čime je zadat režim rada sa izlaznom periferijom, kontroler postavlja bit SR_0 na aktivnu vrednost koja služi kao indikacija da je registar DR raspoloživ da se programskim putem u njega upiše podatak. Prilikom upisa podatka u registar DR kontroler postavlja bit SR_0 na neaktivnu vrednost koja služi kao indikacija da registar DR više nije raspoloživ da se programskim putem u njega upiše novi podatak. Kontroler po završetku prenosa podatka iz registra DR u izlaznu periferiju postavlja bit SR_0 na aktivnu vrednost koja služi kao indikacija da je registar DR ponovo raspoloživ da se programskim putem u njega upiše podatak.

Ukoliko je prilikom starovanja kontrolera u bit CR_0 upisana aktivna vrednost, čime je zadat režim rada sa ulaznom periferijom, kontroler postavlja bit SR_0 na neaktivnu vrednost koja služi kao indikacija da registar DR nije raspoloživ da se programskim putem iz njega čita podatak. Kontroler po završetku prenosa podatka iz ulazne periferije u registar DR postavlja bit SR_0 na aktivnu vrednost koja služi kao indikacija da je registar DR raspoloživ da se programskim putem iz njega čita podatak. Prilikom čitanja podatka iz registra DR kontroler postavlja bit SR_0 na neaktivnu vrednost koja služi kao indikacija da registar DR više nije raspoloživ da se programskim putem iz njega čita podatak. Kontroler po završetku prenosa sledećeg podatka iz ulazne periferije u registar DR postavlja bit SR_0 na aktivnu vrednost koja služi kao indikacija da je registar DR ponovo raspoloživ da se programskim putem iz njega čita podatak.

Bit SR_1 se postavlja na aktivnu i neaktivnu vrednost po određenim pravilima u zavisnosti od toga da li se prilikom rada kontrolera u režimu sa ulaznom periferijom desila greška pri proveru bita parnosti. Bit SR_1 se referiše kao bit *parnost*.

Ukoliko je prilikom starovanja kontrolera u bit CR_0 upisana aktivna vrednost, čime je zadat režim rada sa ulaznom periferijom, kontroler postavlja bit SR_1 na neaktivnu vrednost koja služi kao indikacija da se nije dogodila greška pri proveru bita parnosti. Kontroler po završetku prenosa podatka iz ulazne periferije u registar DR postavlja bit SR_1 na aktivnu vrednost koja služi kao indikacija da se desila greška pri proveru bita parnosti ako je to slučaj u suprotnom bit SR_1 se postavlja na neaktivnu vrednost.

Bit SR_2 se postavlja na aktivnu i neaktivnu vrednost po određenim pravilima u zavisnosti od toga da li se prilikom rada kontrolera u režimu sa ulaznom periferijom desila greška pri proveru stop bita. Bit SR_2 se referiše kao bit *stop*.

Ukoliko je prilikom starovanja kontrolera u bit CR_0 upisana aktivna vrednost, čime je zadat režim rada sa ulaznom periferijom, kontroler postavlja bit SR_2 na neaktivnu vrednost koja služi kao indikacija da se nije dogodila greška pri proveru bita parnosti. Kontroler po završetku prenosa podatka iz ulazne periferije u registar DR postavlja bit SR_2 na aktivnu vrednost koja služi kao indikacija da se desila greška pri proveru bita parnosti ako je to slučaj u suprotnom bit SR_2 se postavlja na neaktivnu vrednost.

Registar DR je registar podatka širine 8 bita. Ovaj registar se koristi kao prihvatni registar za podatke koje razmenjuju kontroler i procesor. Slanje podatka u izlazni uređaj se realizuje programskim putem izvršavanjem instrukcija koje upisuju podatak u registar DR. Unos podatka iz ulaznog uređaja se realizuje programskim putem izvršavanjem instrukcija koje čitaju podatak iz registra DR.

U tabeli **Error! Reference source not found.** su date relativne adrese svih programski dostupnih registara kontrolera za serijski prenos u okviru opsega od 64 adrese datog kontrolera.

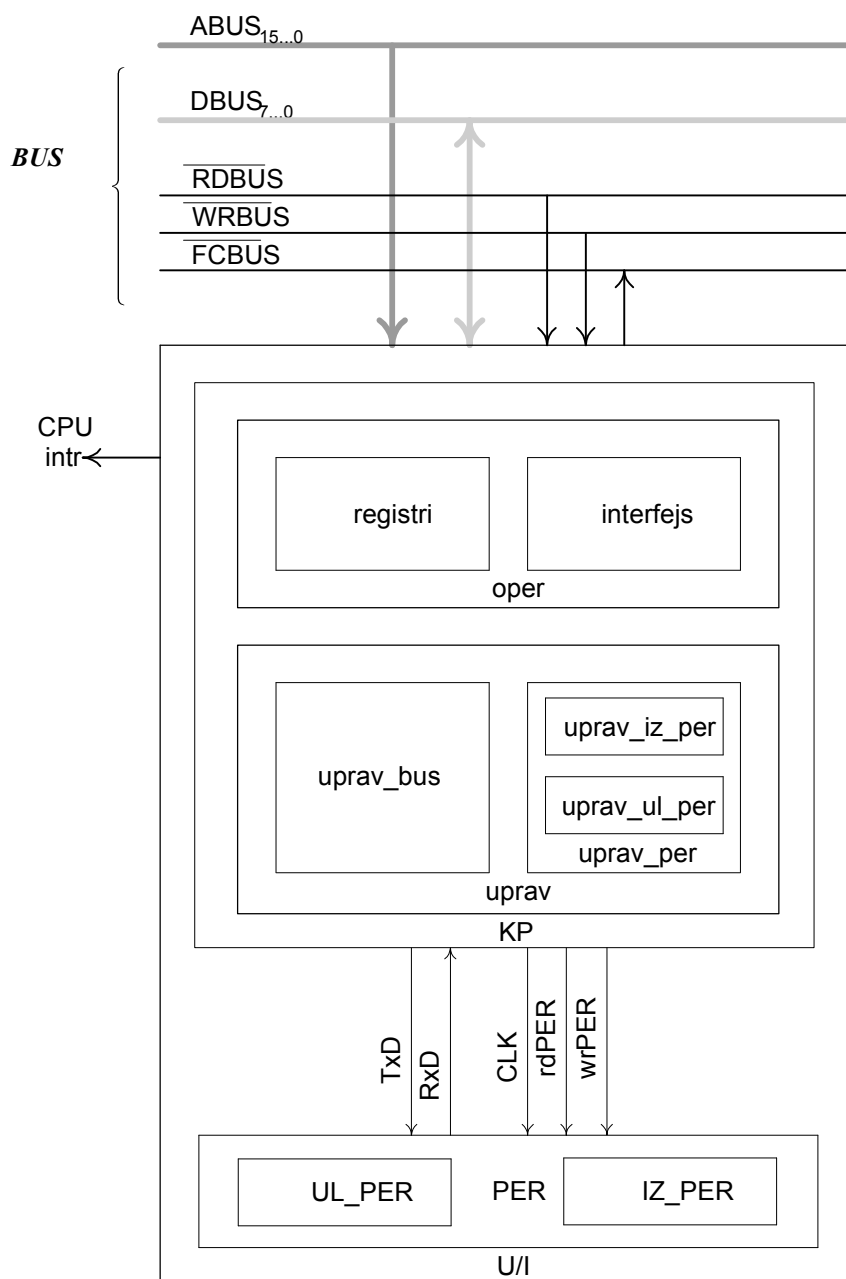
Tabela 24 Relativne adrese registara kontrolera bez direktnog pristupa memoriji

Registar	Adresa
CR	0
SR	1
DR	2

5.2 Organizacija kontrolera periferije sa serijskim prenosom

Kontroler periferije *KP* (slika 24) se sastoji iz operacione jedinice ***oper*** i upravljačke jedinice ***uprav***.

Operaciona jedinica ***oper*** je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova.



Slika 24 Organizacija kontrolera periferije *KP*

Upravljačka jedinica **uprav** se sastoji iz dva dela i to upravljačke jedinice magistrale *uprav_bus* i upravljačke jedinice periferije *uprav_per*. Upravljačka jedinica magistrale *uprav_bus* generiše upravljačke signale neophodne da kontroler periferije *KP* kao sluga realizacije cikluse na magistrali **BUS** kojima se prenose podaci između procesora *CPU* i kontrolera periferije *KP*. Upravljačka jedinica periferije *uprav_per* se sastoji iz dva dela i to upravljačke jedinice ulazne periferije *uprav_ul* i upravljačke jedinice izlazne periferije *uprav_iz*. Upravljačka jedinica periferije *uprav_per* generiše upravljačke signale neophodne za prenos podataka između periferije *PER* i kontrolera periferije *KP*. Upravljačke jedinice *uprav_bus* i *uprav_per* rade istovremeno i omogućuju paralelan rad kontrolera periferije *KP* kao sluge sa magistralom **BUS** i kontrolera periferije *KP* sa periferijom *PER*. Svaka od upravljačkih jedinica *uprav_bus* i *uprav_per* je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala prema algoritmu generisanja upravljačkih signala operacione jedinice **oper** i signala logičkih uslova.

Struktura i opis operacione i upravljačke jedinice se daju u daljem tekstu.

5.2.1 Operaciona jedinica

Operaciona jedinica (slika 24) se sastoji od sledećih blokova:

- blok *registri* i
- blok *interfejs*.

Blok *registri* (slike 25) služi za čuvanje podataka, upravljačkih i statusnih informacija. Blok *interfejs* (slike 26) služi za realizaciju ciklusa na magistrali u kojima je kontroler sluga i za realizaciju prekida.

Struktura i opis blokova operacione jedinice *oper* se daju u daljem tekstu.

5.2.1.1 Blok registri

Blok *registri* (slika 25) sadrži registre DR_{7...0}, DRIN_{7...0} i DROUT_{7...0} sa multiplekserom MP1, registre CR_{2...0} i SR_{2...0} i flip-flopove WRDR i RDDR.

Registri DR_{7...0} i DROUT_{7...0} su 8-mo razredni registar podatka i pomoćni registar podatka, respektivno, dok je registar DRIN_{7...0} pomerački prihvatni registar podatka (slika 25). Ovi registri zajedno sa multiplekserom MP1 služe za prenos podataka između periferije *PER* i memorije *MEM*.

Pri prenosu podataka iz periferije u memoriju podatak koji dolazi iz periferije po liniji RxD se, najpre, upisuje bit po bit uz pomeranje u pomoćni registar podatka DRIN_{7...0}. Potom se podatak iz registra DRIN_{7...0} propušta kroz multiplekser MP1 i upisuje u registar podatka DR_{7...0}. Na kraju se podatak iz registra DR_{7...0} preko bafera sa tri stanja propušta na linije podataka DBUS_{7...0} magistrale *BUS* i upisuje u memoriju. Ovakvim načinom prenosa podataka je omogućeno da se istovremeno prenosi tekući podatak iz registra DR_{7...0} u memoriju i sledeći podatak iz periferije u registar DRIN_{7...0}. Pri tome se sledeći podatak prenosi iz registra DRIN_{7...0} u registar DR_{7...0}, tek pošto je tekući podatak prenet iz registra DR_{7...0} u memoriju.

Pri prenosu podataka iz memorije u periferiju podatak koji dolazi iz memorije po linijama podataka DBUS_{7...0} magistrale se, najpre, propušta kroz multiplekser MP1 i upisuje u registar podatka DR_{7...0}. Potom se podatak iz registra DR_{7...0} upisuje u pomoćni registar podatka DROUT_{7...0}. Na kraju se podatak iz registra DROUT_{7...0} po linijama POUT_{7...0} šalje u upravljačku jedinicu izlazne periferije i upisuje u periferiju bit po bit. Ovakvim načinom prenosa podataka je omogućeno da se istovremeno prenosi tekući podatak iz registra DROUT_{7...0} u periferiju i sledeći podatak iz memorije u registar DR_{7...0}. Pri tome se sledeći podatak prenosi iz registra DR_{7...0} u registar DROUT_{7...0}, tek pošto je tekući podatak prenet iz registra DROUT_{7...0} u periferiju.

Registar DR_{7...0} je 8-mo razredni registar u koji se generisanjem aktivne vrednosti signala **ldDR** upisuje sadržaj sa izlaza multipleksa MP1. Pri neaktivnoj vrednosti signala **ldDR** sadržaj sa linija podataka DBUS_{7...0} magistrale se propušta kroz multiplekser MP1 i upisuje u registar DR_{7...0}. Pri aktivnoj vrednosti signala **ldDR** sadržaj sa izlaza registra DRIN_{7...0} se propušta kroz multiplekser MP1 i upisuje u registar DR_{7...0}. Sadržaj registra DR_{7...0} se pri aktivnoj vrednosti signala **DRout** propušta kroz bafere sa tri stanja na linije podataka DBUS_{7...0} magistrale. Pored toga sadržaj registra DR_{7...0} se vodi na ulaze registra DROUT_{7...0}.

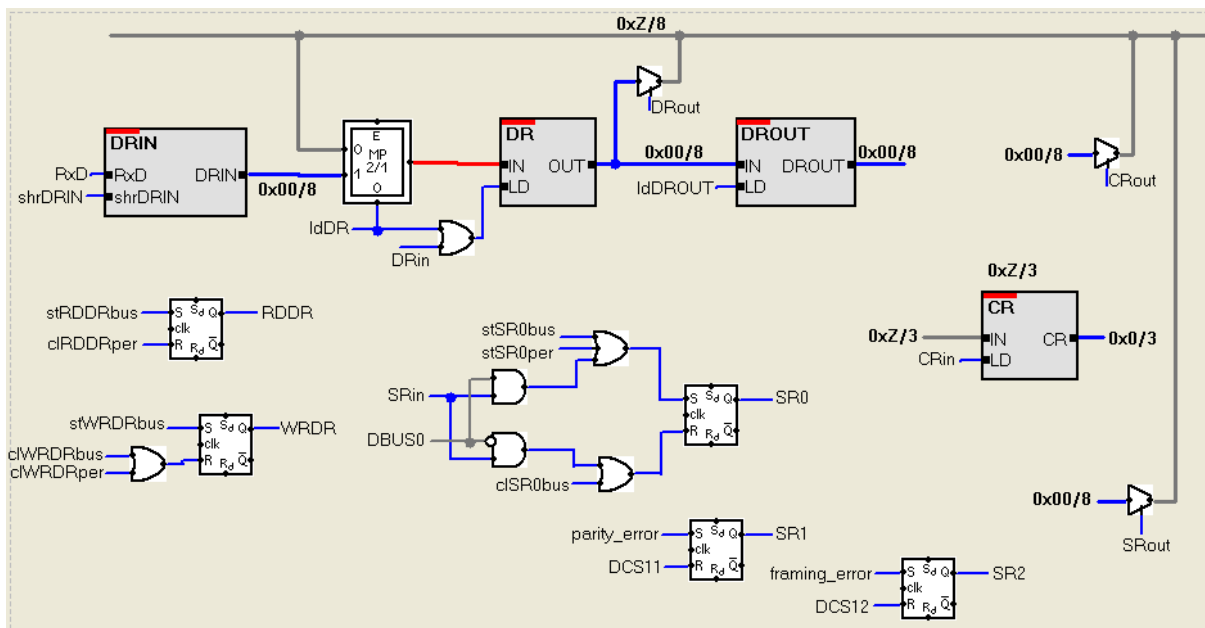
Multiplekser MP1 se sastoji od 8 multipleksa sa dva ulaza. Na ulaze 0 i 1 multipleksa MP1 dovodi se sadržaj sa linija podataka DBUS_{7...0} magistrale i sadržaj sa izlaza registra DRIN_{7...0}, respektivno. Selektovanje sadržaja se obavlja signalom **ldDR**.

Registar DRIN_{7...0} je 8-mo razredni registar u koji se, generisanjem aktivne vrednosti signala **ldDRIN** upisuje sadržaj sa linije RxD. Pri aktivnoj vrednosti signala **shrDRIN** vrši se pomeranje sadržaja registra DRIN_{7...0} za jedno mesto udesno. Pri aktivnoj vrednosti signala **ldDDR** sadržaj registra DRIN_{7...0} se propušta kroz multiplexer MP1 i upisuje u registar DR_{7...0}.

Registar DROUT_{7...0} je 8-mo razredni registar u koji se, generisanjem aktivne vrednosti signala **ldDROUT** upisuje sadržaj registra DR.

Registar CR_{2...0} je 3-bitni upravljački registar koji sadrži bitove start, u/i i prekid, respektivno. U registar CR_{2...0} se, generisanjem aktivne vrednosti signala **CRin** upisuje sadržaj sa linija podataka DBUS_{2...0} magistrale. Sadržaj registra CR_{2...0} proširen nulama do dužine 8 bitova se pri aktivnoj vrednosti signala **CRout** propušta kroz bafere sa tri stanja na linije podataka DBUS_{7...0} magistrale.

Registar SR_{2...0} je 3-bitni upravljački registar koji sadrži bite framing_error, parnost i spremnost, respektivno. Bit SR₀ se naziva bit *spremnost*. U bit SR₀ se, generisanjem aktivne vrednosti signala **SRin**, upisuje sadržaj sa linija podataka DBUS₀ magistrale. Sadržaj registra SR₀ proširen nulama do dužine 8 bitova se pri aktivnoj vrednosti signala **SRout** propušta kroz bafere sa tri stanja na linije podataka DBUS_{7...0} magistrale. Pri prenosu podataka iz periferije u memoriju njegova aktivna vrednost je indikacija da je novi podatak prebačen iz registra DRIN_{7...0} u registar DR_{7...0} i da je raspoloživ da se programskim putem prebaci iz registra DR_{7...0} u memoriju. S toga se na početku, pri startovanju kontrolera periferije za prenos iz periferije u memoriju, flip-flop registra SR₀ signalom **clSRbus** postavlja na neaktivnu vrednost. Posle toga se, po prenosu podatka iz registra DRIN_{7...0} u registar DR_{7...0}, flip-flop SR₀ signalom **stSR0per** postavlja na aktivnu vrednost, a po prenosu podatka iz registra DR_{7...0} u memoriju signalom **clSR0bus** postavlja na neaktivnu vrednost. Pri prenosu podataka iz memorije u periferiju njegova aktivna vrednost je indikacija da je sadržaj registra DR_{7...0} prebačen u registar DROUT_{7...0} i da je registar DR_{7...0} raspoloživ da se u njega programskim putem prebaci novi podatak iz memorije. S toga se na početku, pri startovanju kontrolera periferije za prenos iz memorije u periferiju, flip-flop SR₀ signalom **stSR0bus** postavlja na aktivnu vrednost. Posle toga se, pri prenosu podatka iz memorije u registar DR_{7...0}, flip-flop SR₀ signalom **clSR0bus** postavlja na neaktivnu vrednost, a pri prenosu podatka iz registra DR_{7...0} u registar DROUT_{7...0} signalom **stSR0per** postavlja na aktivnu vrednost. Bit SR₁ se naziva bit *parnost*. Flip-flop SR₁ se signalom **parity_error** postavlja na aktivnu vrednost, što je indikacija da se desila greška pri proveru bita parnosti; a aktivnim signalom **DCS11** postavlja na neaktivnu vrednost pre provere bita parnosti svakog primljenog podatka. Bit SR₂ se naziva bit *framing_error*. Flip-flop SR₂ se signalom **framing_error** postavlja na aktivnu vrednost, što je indikacija da se desila greška pri proveru



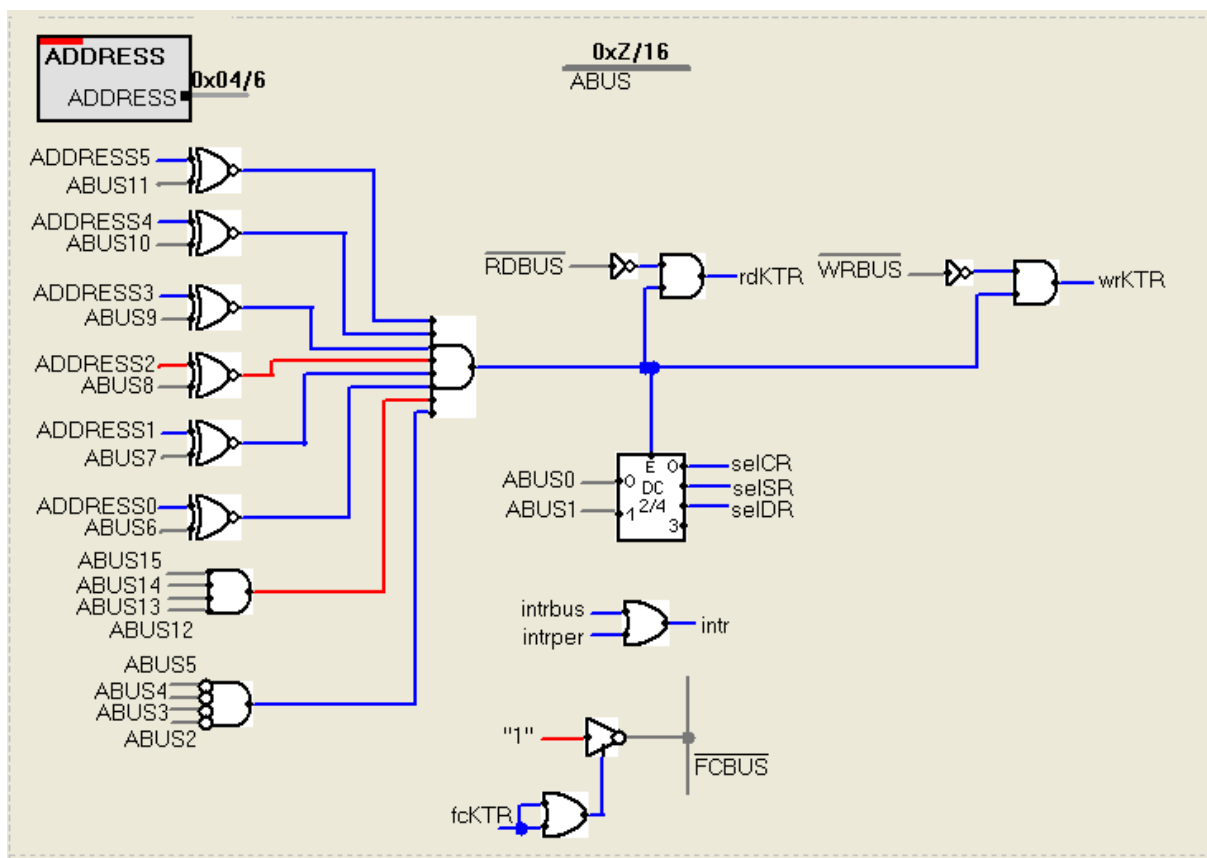
Slika 25 Blok registri

bita parnosti; a aktivnim signalom **DCS12** postavlja na neaktivnu vrednost pre provere bita framing_error svakog primljenog podatka.

Flip-floпови RDDR i WRDR se koriste za neophodnu sinhronizaciju pri prenosu podataka iz periferije u memoriju i iz memorije u periferiju, respektivno. Flip-flop RDDR se koristi pri prenosu podataka iz periferije u memoriju. Njegova aktivna vrednost je indikacija da novi podatak može da se prenese iz registra DRIN_{7...0} u registar DR_{7...0}, a neaktivna da tekući podatak još uvek nije prenet iz registra DR_{7...0} u memoriju i da novi podatak ne može da se prenese iz registra DRIN_{7...0} u registar DR_{7...0}. S toga se na početku pri startovanju kontrolera periferije za prenos iz periferije u memoriju flip-flop RDDR signalom **stRDDRbus** postavlja na aktivnu vrednost. Posle toga se po prenosu podatka iz registra DRIN_{7...0} u registar DR_{7...0} flip-flop RDDR signalom **clRDDRper** postavlja na neaktivnu vrednost, a po prenosu podatka iz registra DR_{7...0} u memoriju signalom **stRDDRbus** postavlja na aktivnu vrednost. Flip-flop WRDR se koristi pri prenosu podataka iz memorije u periferiju. Njegova aktivna vrednost je indikacija da se u registru DR_{7...0} nalazi podatak i da sadržaj registra DR_{7...0} može da se prenese u registar DROUT_{7...0}, a neaktivna da novi podatak još uvek nije prenet iz memorije u registar DR_{7...0} i da sadržaj registra DR_{7...0} ne može da se prenese u registar DROUT_{7...0}. S toga se na početku po startovanju kontrolera za prenos iz memorije u periferiju flip-flop WRDR signalom **clWRDRbus** postavlja na neaktivna vrednost. Posle toga se pri prenosu podatka iz memorije u registar DR_{7...0} flip-flop WRDR signalom **stWRDRbus** postavlja na aktivnu vrednost, a pri prenosu podatka iz registra DR_{7...0} u registar DROUT_{7...0} signalom **clWRDRper** postavlja na neaktivnu vrednost.

5.2.1.2 Blok interfejs

Blok *interfejs* (slika 26) sadrži kombinacione i sekvencijalne mreže za realizaciju ciklusa na magistrali u kojima je kontroler sluga i za generisanje prekida.



Slika 26 Blok interfejs

Kombinacione i sekvencijalne mreže za realizaciju ciklusa na magistrali u kojima je kontroler sluga formiraju signale **rdKTR** i **wrKTR** upravljačke jedinice *uprav_bus* i **FCBUS** magistrale **BUS** (slika 26). Signali **rdKTR** i **wrKTR** se formiraju na osnovu signala **ABUS_{15..0}** sa adresnih linija magistrale i **RDBUS** i **WRBUS** sa upravljačkih linija magistrale. Signal **FCBUS** se formira na osnovu signala **fcKTR** upravljačke jedinice *uprav_bus*.

Pri realizaciji ciklusa čitanja na magistrali procesor kao gazda otvara bafere sa tri stanja za adresne linije **ABUS_{15..0}** i upravljačku liniju **RDBUS** magistrale i na njih izbacuje adresu i aktivnu vrednost signala čitanja, respektivno, čime se u kontroleru kao slugi startuje čitanje adresiranog registra. Kontroler po završenom čitanju otvara bafere sa tri stanja za linije podataka **DBUS_{7..0}** i upravljačku liniju **FCBUS** magistrale i na njih izbacuje sadržaj adresiranog registra i aktivnu vrednost signala završetka operacije čitanja u kontroleru. Procesor prihvata sadržaj sa linija podataka i zatvara bafere sa tri stanja za adresne linije **ABUS_{15..0}** i upravljačku liniju **RDBUS** magistrale, dok kontroler zatvara bafere sa tri stanja za linije podataka **DBUS_{7..0}** i upravljačku liniju **FCBUS** magistrale.

Pri realizaciji ciklusa upisa na magistrali procesor kao gazda otvara bafere sa tri stanja za adresne linije **ABUS_{15..0}**, linije podataka **DBUS_{7..0}** i upravljačku liniju **WRBUS** magistrale i na njih izbacuje adresu, podatak i aktivnu vrednost signala upisa, respektivno, čime se u kontroleru kao slugi startuje upis u adresirani registar. Kontroler po završenom upisu otvara bafere sa tri stanja za i upravljačku liniju **FCBUS** magistrale i na nju izbacuje aktivnu vrednost signala završetka operacije upisa u kontroleru. Procesor zatvara bafere sa tri stanja za

adresne linije $ABUS_{15..0}$, linije podataka $DBUS_{7..0}$ i upravljačku liniju \overline{WRBUS} magistrale, dok kontroler zatvara bafere sa tri stanja za upravljačku liniju \overline{FCBUS} magistrale.

Signali **rdKTR** i **wrKTR** imaju vrednosti upravljačkih signala \overline{RDBUS} i \overline{WRBUS} magistrale, respektivno, kada je aktivna vrednost signala **select** i neaktivne vrednosti kada je neaktivna vrednost signala **select**. Signal **rdKTR** je kao i signal \overline{RDBUS} aktivan za vreme operacije čitanja nekog registra kontrolera, dok je signal **wrKTR** kao i signal \overline{WRBUS} aktivan za vreme operacije upisa u neki od registara kontrolera.

Signal **select** ima aktivnu vrednost ukoliko se na adresnim linijama $ABUS_{15..0}$ magistrale nalazi adresa iz opsega adresa dodeljenih registrima kontrolera periferije *KP*. Celokupan opseg adresa od 0000h do FFFFh podeljen je na opseg adresa od 0000h do EFFFh, koje su dodeljene memoriji *MEM*, i opseg adresa od F000h do FFFFh, koje su dodeljene registrima po svim kontrolerima periferija. Opseg adresa koje su dodeljene registrima po kontrolerima periferija je predviđen za adresiranje registara u najviše 64 kontrolera i to najviše 64 registra unutar određenog kontrolera. Pri tome prilikom adresiranja nekog od registara kontrolera sadržaj na adresnim linijama $ABUS_{15..0}$ magistrale ima sledeću strukturu: bitovi $ABUS_{15..12}$ su sve jedinice, bitovi $ABUS_{11..6}$ određuju broj kontrolera i bitovi $ABUS_{5..0}$ adresu registra unutar kontrolera. Broj kontrolera periferije za određeni kontroler periferije se postavlja mikroprekidačima na jednu od vrednosti u opsegu 0 do 63. Registrima $CR_{7..0}$, $SR_{7..0}$ i $DR_{7..0}$ su unutar opsega adresa datog kontrolera dodeljene adrese 0 do 2, pa bitovi $ABUS_{5..2}$ moraju da budu nule dok se njihovo pojedinačno adresiranje realizuje bitovima $ABUS_1$ i $ABUS_0$.

Na osnovu usvojene strukture adresa signal **select** ima aktivnu vrednost ukoliko su na adresnim linijama $ABUS_{15..12}$ magistrale sve jedinice, na adresnim linijama $ABUS_{11..6}$ se nalazi vrednost koja odgovara vrednosti postavljenoj mikroprekidačima i na adresnim linijama $ABUS_{5..2}$ sve nule. Signali **selCR**, **selSR** i **selDR** se dobijaju na izlazima 0 do 2 dekodera DC na osnovu vrednosti signala $ABUS_1$, $ABUS_0$ i **select** sa ulaza 1, 0 i E, respektivno. Pri neaktivnoj vrednosti signala **select** i signali **selCR**, **selSR** i **selDR** su neaktivni. Pri aktivnoj vrednosti signala **select** jedan od signala **selCR**, **selSR** i **selDR** postaje aktivan i to signal određen binarnom vrednošću signala $ABUS_1$ i $ABUS_0$.

Signal **select** ima neaktivnu vrednost ukoliko se na adresnim linijama $ABUS_{15..0}$ magistrale ne nalazi adresa iz opsega adresa dodeljenih registrima kontrolera periferije, pri čemu to može da bude ili adresa memorijske lokacije ili registra iz nekog drugog kontrolera periferije. Tada se formiraju neaktivne vrednosti signala **rdKTR** i **wrKTR** bez obzira na vrednosti signala \overline{RDBUS} i \overline{WRBUS} , respektivno.

Kombinaciona mreža za generisanje upravljačkog signala \overline{FCBUS} magistrale generiše ovaj signal na osnovu signala **fcKTR**. Pri neaktivnoj vrednosti signala **fcKTR** na liniji signala \overline{FCBUS} je stanje visoke impedance, dok je pri aktivnoj vrednosti signala **fcKTR** na liniji signala \overline{FCBUS} aktivna vrednost. Signal **fcKTR** ima neaktivnu ili aktivnu vrednost u zavisnosti od toga da li je u kontroleru operacija čitanja ili upisa u toku ili je završena, respektivno.

Kada kontroler treba u procesoru da izazove prekid, generiše se aktivna vrednost signala **intr**. Signala **intr** ima aktivnu vrednost ukoliko ili signal **intrbus** upravljačke jedinice *uprav_bus* ili signal **intrper** upravljačke jedinice *uprav_per* ima aktivnu vrednost.

5.2.2 Upravljačka jedinica

Upravljačka jedinica *uprav_jed* se sastoji iz dva dela:

- upravljačke jedinice magistrale *uprav_bus* i
- upravljačke jedinice periferije *uprav_per* koja se sastoji iz dva dela.

Upravljačke jedinice *uprav_bus* i *uprav_per* rade istovremeno i omogućuju paralelan rad kontrolera periferije *KP* kao sluge sa magistralom **BUS** i kontrolera periferije *KP* sa periferijom *PER*.

Upravljačka jedinica *uprav_bus* omogućuje da procesor **CPU** kao gazda realizuje u kontroleru periferije *KP* kao slugi upisivanje sadržaja u registre i čitanje sadržaja iz registara. Upisivanjem u upravljački registar sadržaja koji na poziciji bita start ima vrednost 1 startuje se kontroler za prenos podataka iz periferije u memoriju ili iz memorije u periferiju, dok se upisivanjem sadržaja koji na poziciji bita start ima vrednost 0, zaustavlja kontroler.

Prilikom startovanja kontrolera periferije za prenos podataka iz periferije u memoriju aktivira se upravljačka jedinica *uprav_per*. Upravljačka jedinica *uprav_per* prvo startuje periferiju da pročita podatak, a zatim podatak prenosi iz periferije, najpre, u pomoćni registar podatka kontrolera, a potom u registar podatka kontrolera. Podatak se prenosi iz registra podatka kontrolera u memoriju programskim putem sa dve instrukcije. Prvom instrukcijom se, uz aktiviranje upravljačke jedinice *uprav_bus*, podatak prenosi iz registra podatka kontrolera u procesor. Drugom instrukcijom se podatak prenosi iz procesora u memoriju. Prenos podatka iz registra podatka u memoriju i čitanje novog podatka iz periferije sa prenosom iz registra podatka periferije u pomoćni registar podatka kontrolera su preklapljeni. Upravljačka jedinica *uprav_per* novi podatak prenosi iz pomoćnog registra podatka u registar podatka tek pošto je upravljačka jedinica *uprav_bus* prethodni podatak prenela iz registra podatka u procesor. Po prenosu podatka iz pomoćnog registra podatka u registar podatka, bit spremnost statusnog registra se postavlja na aktivnu vrednost i generiše prekid ukoliko je pri startovanju zadat režim rada sa generisanjem prekida. Prenos podataka iz periferije u memoriju traje dok se kontroler periferije ne zaustavi programskim putem.

Prilikom startovanja kontrolera periferije za prenos podataka iz memorije u periferiju aktivira se upravljačka jedinica *uprav_per*. Podatak se prenosi iz memorije u registar podatka kontrolera programskim putem sa dve instrukcije. Prvom instrukcijom se podatak prenosi iz memorije u procesor. Drugom instrukcijom se, uz aktiviranje upravljačke jedinice *uprav_bus*, podatak prenosi iz procesora u registar podatka kontrolera. Posle toga upravljačka jedinica *uprav_per* podatak prenosi iz registra podatka kontrolera u pomoćni registar podatka kontrolera i startuje upis podatka u periferiju. Upis podatka iz pomoćnog registra podatka u periferiju i prenos novog podatka iz memorije u registar podatka kontrolera su preklapljeni. Upravljačka jedinica *uprav_per* novi podatak prenosi iz registra podatka kontrolera u pomoćni registar podatka kontrolera tek pošto prethodni podatak iz pomoćnog registra podatka kontrolera upiše u periferiju. Po prenosu podatka iz registra podatka u pomoćni registar podatka, bit spremnost statusnog registra se postavlja na aktivnu vrednost i generiše prekid ukoliko je pri startovanju zadat režim rada sa generisanjem prekida. Prenos podataka iz memorije u periferiju traje dok se kontroler periferije ne zaustavi programskim putem.

Struktura i opis upravljačkih jedinica *uprav_bus* i *uprav_per* se daju u daljem tekstu.

5.2.2.1 Upravljačka jedinica magistrale

U ovom odeljku se daju dijagram toka operacija, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice *uprav_bus*.

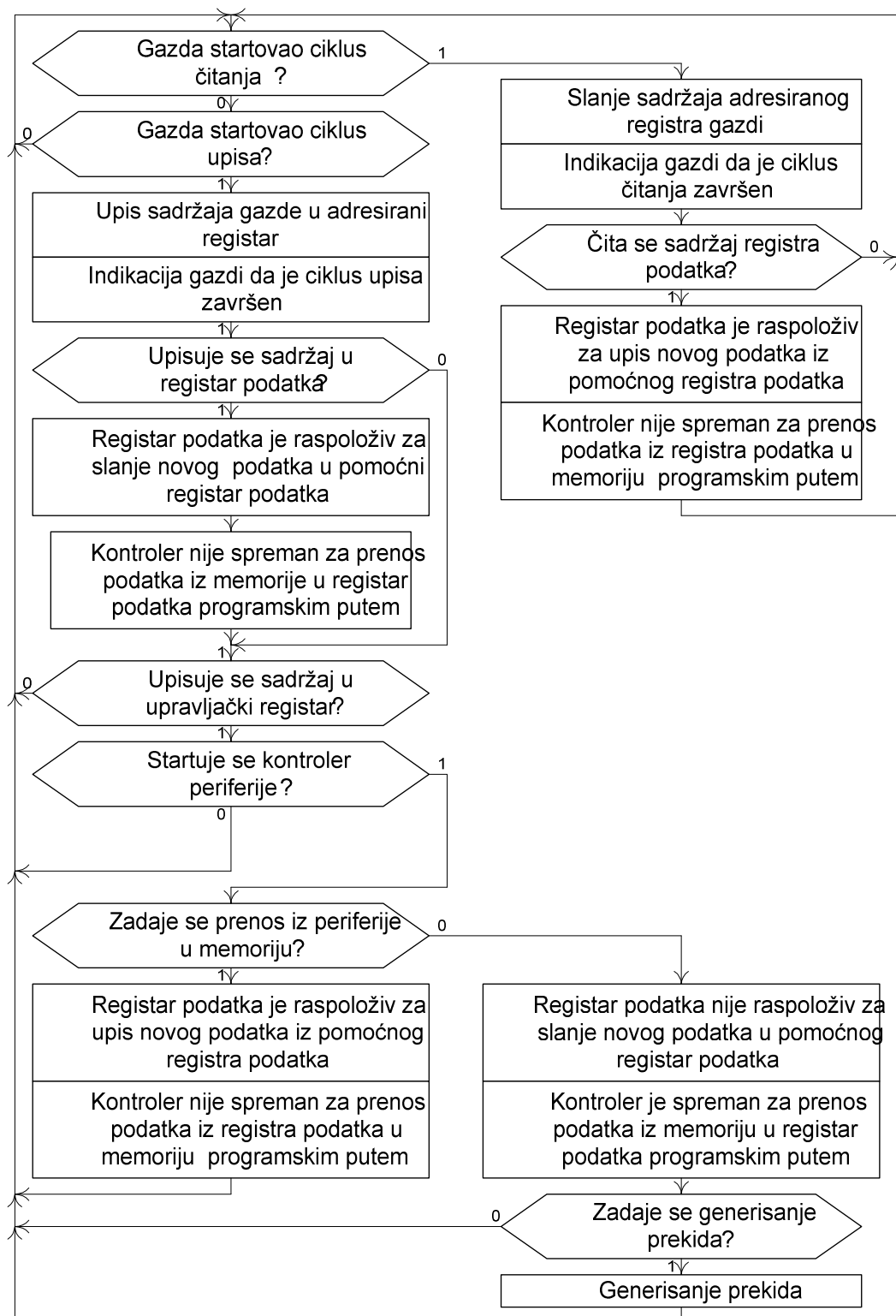
5.2.2.1.1 Dijagram toka operacija

Dijagram toka operacija je predstavljen operacionim i uslovnim blokovima (slika 27). U operacionim blokovima se nalaze opisi mikrooperacija koje treba realizovati. U uslovnim blokovima se nalaze opisi logičkih uslova koji definišu grananja algoritma.

U dijagrama toka operacija stalno se vrši provera da li je procesor startovao u kontroleru ciklus čitanja ili ciklus upisa, pri čemu se procesor ponaša kao gazda, a kontroler kao sluga. Ako nije startovan ni ciklus čitanja ni ciklus upisa nema izvršavanja mikrooperacija. Ako je startovan jedan od ova dva ciklusa izvršavaju se mikrooperacije saglasno ciklusu koji je startovan.

Ako je startovan ciklus čitanja, procesoru se kao gazdi šalje sadržaj adresiranog registra kontrolera i indikacija da je kontroler kao sluga završio čitanje. Pored toga, u slučaju kada se čita sadržaj registra podatka, postavljaju se i dve indikacije. Prva je interna indikacija da je sadržaj registar podatka prenet u memoriju i da je on raspoloživ da se u njega prenese novi podatak iz pomoćnog registra podatka. Druga je bit spremnost statusnog registra koji se postavlja na neaktivnu vrednost kao indikacija da je registar podatka prenet u memoriju i da ne treba da se čita dok se u njega ne prenese novi podatak iz pomoćnog registra podatka.

Ako je startovan ciklus upisa, u adresirani registar kontrolera se upisuje sadržaj koji procesor kao gazda šalje i gazdi šalje indikacija da je kontroler kao sluga završio upis. U slučaju kada se sadržaj upisuje u registar podatka, postavljaju se i dve indikacije. Prva je interna indikacija da je u registar podatka upisan novi podatak i da je on raspoloživ da se iz njega prenese novi podatak u pomoćni registar podatka. Druga je bit spremnost statusnog registra koji se postavlja na neaktivnu vrednost kao indikacija da je u registar podatka upisan podatak i da ne treba da se vrši upis novog podatka dok se podatak iz njega ne prenese u pomoćni registar podatka.

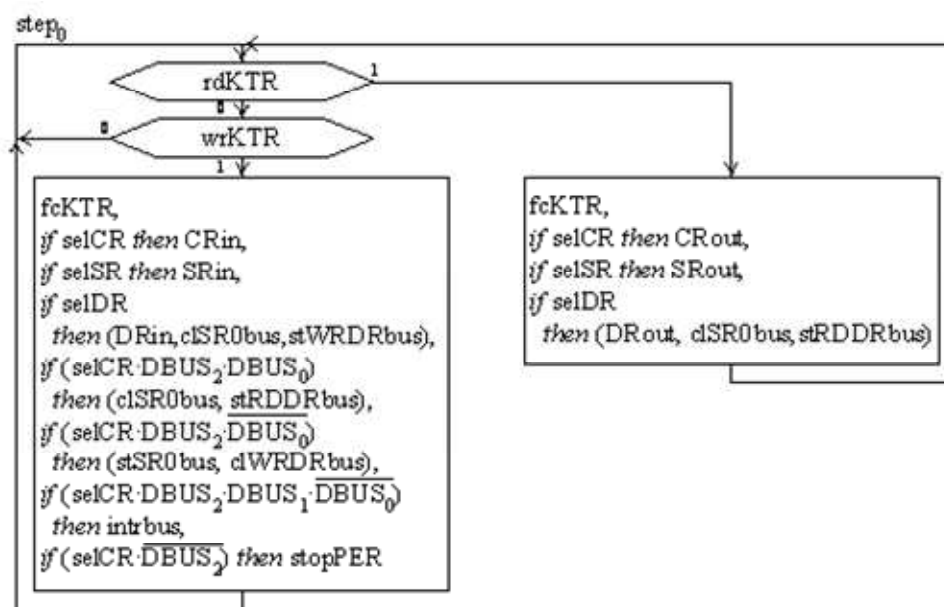


Slika 27 Dijagram toka operacija

U slučaju kada se sadržaj upisuje u upravljački registar, mikrooperacije koje se izvršavaju zavise od toga da li se kontroler periferije zaustavlja ili startuje. Ukoliko se vrši startovanje, mikrooperacije koje se izvršavaju zavise od toga da li se kontroler periferije startuje za prenos iz periferije u memoriju ili za prenos iz memorije u periferiju. Ukoliko se vrši startovanje za prenos iz periferije u memoriju postavljaju se i dve indikacije. Prva je interna indikacija da je registar podatka raspoloživ da se u njega prenese podatak iz pomoćnog registra podatka. Druga je bit spremnost statusnog registra koji se postavlja na neaktivnu vrednost kao indikacija da ne treba da se čita dok se u njega ne prenese podatak iz pomoćnog registra podatka. Ukoliko se vrši startovanje za prenos iz memorije u periferiju postavljaju se i dve indikacije. Prva je interna indikacija da u registar podatka nije upisan novi podatak iz memorije i da on nije raspoloživ da se iz njega prenese novi podatak u pomoćni registar podatka. Druga je bit spremnost statusnog registra koji se postavlja na aktivnu vrednost kao indikacija da u registar podatka može da se prenese podatak iz memorije. Pored toga, ukoliko se zadaje rad sa generisanjem prekida, generiše se i prekid.

5.2.2.1.2 Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu dijagrama toka operacija



(slika 27) i dat u obliku dijagrama toka upravljačkih signala (slika 28) i sekvence upravljačkih signala (tabela 15).

Slika 28 Dijagram toka upravljačkih signala

Dijagram toka upravljačkih signala je predstavljen operacionim i uslovnim blokovima (slika 28). U operacionim blokovima se nalaze upravljački signali i uslovi pod kojima se oni generišu. U uslovnim blokovima se nalaze signali logičkih uslova koji određuju grananja u algoritmu.

U sekvenci upravljačkih signala se koriste iskazi za signale (tabela 15). Iskazi za signale su oblika

if uslov then signali

Ovi iskazi sadrže koji sadrže uslov i spisak upravljačkih signala blokova operacione jedinice **oper** i određuje koji signali i pod kojim uslovima treba da budu generisani.

Objašnjenja vezana za generisanje upravljačkih signala su data zajednički za dijagram toka upravljačkih signala (slika 28) i sekvencu upravljačkih signala (tabela 15) i to u okviru sekvence upravljačkih signala.

Tabela 15 Sekvenca upravljačkih signala

! U koraku step_0 se ostaje sve vreme. U ovom koraku se ne generiše aktivna vrednost ni jednog od upravljačkih signala sve dok su oba signala **rdKTR** i **wrKTR** bloka *interfejs* neaktivni. Signal **rdKTR** postaje aktivan kada processor **CPU** prebaci adresne linije ABUS_{15..0} magistrale iz stanja visoke impedanse na vrednost adrese nekog od programski dostupnih registara CR_{2...0}, SR₀ ili DR_{7...0} bloka *registri* i upravljačku liniju **RDBUS** magistrale iz stanja visoke impedanse na aktivnu vrednost, čime započinje ciklus čitanja. Signal **wrKTR** postaje aktivan kada processor prebaci adresne linije ABUS_{15..0} i linije podataka DBUS_{7..0} magistrale iz stanja visoke impedanse na vrednost adrese registra i sadržaja za upis u neki od programski dostupnih registara CR_{2...0}, SR₀ ili DR_{7...0}, respektivno, i upravljačku liniju **WRBUS** magistrale iz stanja visoke impedanse na aktivnu vrednost, čime započinje ciklus upisa. Pri aktivnoj vrednosti jednog od signala **rdKTR** i **wrKTR** generiše se signal **fcKTR** bloka *interfejs*. Signalom **fcKTR** se obezbeđuje da upravljačka linija **FCBUS** magistrale pređe iz stanja visoke impedanse na aktivnu vrednost. U oba slučaja se, u zavisnosti od toga koji je od registara CR_{2...0}, SR₀ ili DR_{7...0} adresiran sadržajem na adresnim linijama ABUS_{15..0} magistrale, generiše aktivna vrednost jednog od signala **selCR**, **selSR** i **selDR**, respektivno, bloka *interfejs*. !

! Pri aktivnoj vrednosti signala **rdKTR**, a u zavisnosti od toga koji od signala **selCR**, **selSR** i **selDR** bloka *interfejs* ima aktivnu vrednost, generiše se aktivna vrednost jednog od signala **CRout**, **SRout** i **DRout** bloka *registri*, respektivno. Signalima **CRout**, **SRout** i **DRout** se obezbeđuje da linije podataka DBUS_{7...0} magistrale pređu iz stanja visoke impedanse na vrednost sadržaja jednog od registara CR_{2...0}, SR₀ ili DR_{7...0}, respektivno. Kada signal **rdKTR** postane neaktivan i signali **fcKTR**, **CRout**, **SRout** i **DRout** postanu neaktivni, pa upravljačka linija **FCBUS** magistrale i linije podataka DBUS_{7..0} magistrale se vraćaju u stanje visoke impedanse. !

! Pri aktivnim vrednostima signala **rdKTR** i **selDR**, što znači da se programskim putem čita sadržaj registra DR_{7..0} kao deo postupka prenošenja u memoriju, generišu se signali **clSR0bus** i **stRDDRbus** bloka *registri*. Signalom **clSR0bus** se razred SR₀ statusnog registra SR₀ postavlja na neaktivnu vrednost, što je indikacija da registar DR_{7..0} nije raspoloživ da se čita programskim putem dok se u njega ne prenese novi podatak iz registra DRIN_{7..0}. Signalom **stRDDRbus** se flip-flop RDDR postavlja na aktivnu vrednost, što je indikacija da je registar DR_{7..0} raspoloživ da se u njega prenese novi podatak iz registra DRIN_{7..0} bloka *registri*. !

! Pri aktivnoj vrednosti signala **wrKTR**, a u zavisnosti od toga koji od signala **selCR**, **selSR** i **selDR** ima aktivnu vrednost, generiše se aktivna vrednost jednog od signala **CRin**, **SRin** i **DRin** bloka *registri*, respektivno. Signalima **CRin**, **SRin** i **DRin** se obezbeđuje da se sadržaj sa linija podataka DBUS_{7...0} magistrale upiše u jedan od registara CR_{2...0}, SR₀ ili DR_{7...0}, respektivno. Kada signal **wrKTR** postane neaktivan i signali **fcKTR**, **CRin**, **SRin** i **DRin** postanu neaktivni, pa se upravljačka linija **FCBUS** magistrale vraća u stanje visoke impedanse. !

! Pri aktivnim vrednostima signala **wrKTR** i **selDR**, što znači da se programskim putem upisuje sadržaj u registar DR_{7..0} kao deo postupka prenošenja iz memorije, generišu se signali **clSR0bus** i **stWRDRbus** bloka *registri*. Signalom **clSR0bus** razred SR₀ statusnog registra SR₀ se postavlja na neaktivnu vrednost, što je indikacija da registar DR_{7...0} nije raspoloživ da se u njega programskim putem upiše novi podatak iz memorije dok se prethodni podatak iz njega ne prenese u registar DROUT_{7...0}. Signalom **stWRDRbus** flip-flop WRDR se postavlja na aktivnu vrednost, što je indikacija da je registar DR_{7...0} raspoloživ da se iz njega prenese novi podatak u registar DROUT_{7...0}. !

! Pri aktivnim vrednostima signala **wrKTR**, **selCR**, **DBUS₂** i **DBUS₀**, što znači da se programskim putem upisuje sadržaj u registar CR_{2...0} radi startovanja kontrolera za prenos iz periferije u memoriju,

generišu se signali **clSR0bus** i **stRDDRbus** bloka *registri*. Signalom **clSR0bus** razred SR_0 statusnog registra SR_0 se postavlja na neaktivnu vrednost, što je indikacija da registar $DR_{7...0}$ nije raspoloživ da se čita programskim putem dok se u njega ne prenese novi podatak iz registra $DRIN_{7...0}$. Signalom **stRDDRbus** se flip-flop RDDR postavlja na aktivnu vrednost, što je indikacija da je registar $DR_{7...0}$ raspoloživ da se u njega prenese novi podatak iz registra $DRIN_{7...0}$ bloka *registri*. !

! Pri aktivnim vrednostima signala **wrKTR**, **selCR** i **DBUS₂** i neaktivnoj vrednosti signala **DBUS₀**, što znači da se programskim putem upisuje sadržaj u registar $CR_{2...0}$ radi startovanja kontrolera za prenos iz memorije u periferiju, generišu se signali **stSR0bus** i **clWRDRbus** bloka *registri*. Signalom **stSR0bus** razred SR_0 statusnog registra SR_0 se postavlja na aktivnu vrednost, što je indikacija da je registar $DR_{7...0}$ raspoloživ da se u njega programskim putem upiše podatak iz memorije. Signalom **clWRDRbus** se flip-flop WRDR postavlja na neaktivnu vrednost, što je indikacija da je registar $DR_{7...0}$ nije raspoloživ da se iz njega prenese novi podatak u registar $DROUT_{7...0}$. !

! Pri aktivnim vrednostima signala **wrKTR**, **selCR**, **DBUS₂** i **DBUS₁** i neaktivnoj vrednosti signala **DBUS₀**, što znači da se programskim putem upisuje sadržaj u registar $CR_{2...0}$ radi startovanja kontrolera za prenos iz memorije u periferiju sa generisanjem prekida, generiše se signal prekida **intrbus** bloka *interfejs*. Signal prekida **intrbus** upravljačke jedinice *uprav_bus* sa signalom prekida **intrper** upravljačke jedinice *uprav_per* formira u bloku *interfejs* signal prekida **intr** procesora **CPU**, što je indikacija da je registar $DR_{7...0}$ raspoloživ da se u njega programskim putem skokom na prekidnu rutinu upiše podatak iz memorije. !

! Pri aktivnim vrednostima signala **wrKTR** i **selCR** i neaktivnoj vrednosti signala **DBUS₂**, što znači da se programskim putem upisuje sadržaj u registar $CR_{2...0}$ radi zaustavljanja kontrolera, generiše se neaktivna vrednost signala **rdPER** ulazne periferije *UL_PER*. Signalom **stopPER** se u periferiji zaustavlja prenos koji je u toku. !

```

step0  if (rdKTR + wrKTR) then fcKTR,
        if (rdKTR·selCR) then CRout,
        if (rdKTR·selSR) then SRout,
        if (rdKTR·selDR) then (DRout, clSR0bus, stRDDRbus),
        if (wrKTR·selCR) then CRin,
        if (wrKTR·selSR) then SRin,
        if (wrKTR·selDR) then (DRin, clSR0bus, stWRDRbus),
        if (wrKTR·selCR·DBUS2·DBUS0) then (clSR0bus, stRDDRbus),
        if (wrKTR·selCR·DBUS2· $\overline{DBUS_0}$ ) then (stSR0bus, clWRDRbus),
        if (wrKTR·selCR·DBUS2·DBUS1· $\overline{DBUS_0}$ ) then intrbus,
        if (wrKTR·selCR· $\overline{DBUS_2}$ ) then stopPER

```

5.2.2.2 Struktura upravljačke jedinice magistrale

Struktura upravljačke jedinice je prikazana na slici 29. Upravljačka jedinica se sastoji od logičkih elemenata koji na osnovu signala čitanja **rdKTR** i signala upisa **wrKTR** bloka *interfejs* i signala logičkih uslova generišu sledeće signale:

- signal **fcKTR** bloka *interfejs* završetka ili čitanja nekog od registara kontrolera $CR_{2...0}$, SR_0 ili $DR_{7...0}$ bloka *registri* ili upisa u neki od ovih registara kontrolera,
- signale **CRout**, **SRout** i **DRout** bloka *registri* čitanja registara kontrolera $CR_{2...0}$, SR_0 ili $DR_{7...0}$,
- signale **CRin**, **SRin** i **DRin** bloka *registri* upisa u registre kontrolera $CR_{2...0}$, SR_0 ili $DR_{7...0}$,
- signale **clSR0bus** i **stSR0bus** bloka *registri* postavljanja na neaktivnu i aktivnu vrednost, respektivno, bita spremnost SR_0 statusnog registra SR_0 kontrolera,

- signal **stRDDRbus** bloka *registri* postavljanja na aktivnu vrednost flip-flopa RDDR i signale **stWRDR** i **clWRDR** bloka *registri* postavljanja na aktivnu i neaktivnu vrednost, respektivno, flip-flopa WRDR,
- signal prekida **intrbus** bloka *interfejs* i
- signal zaustavljanja periferije **stopPER** bloka *interfejs*.

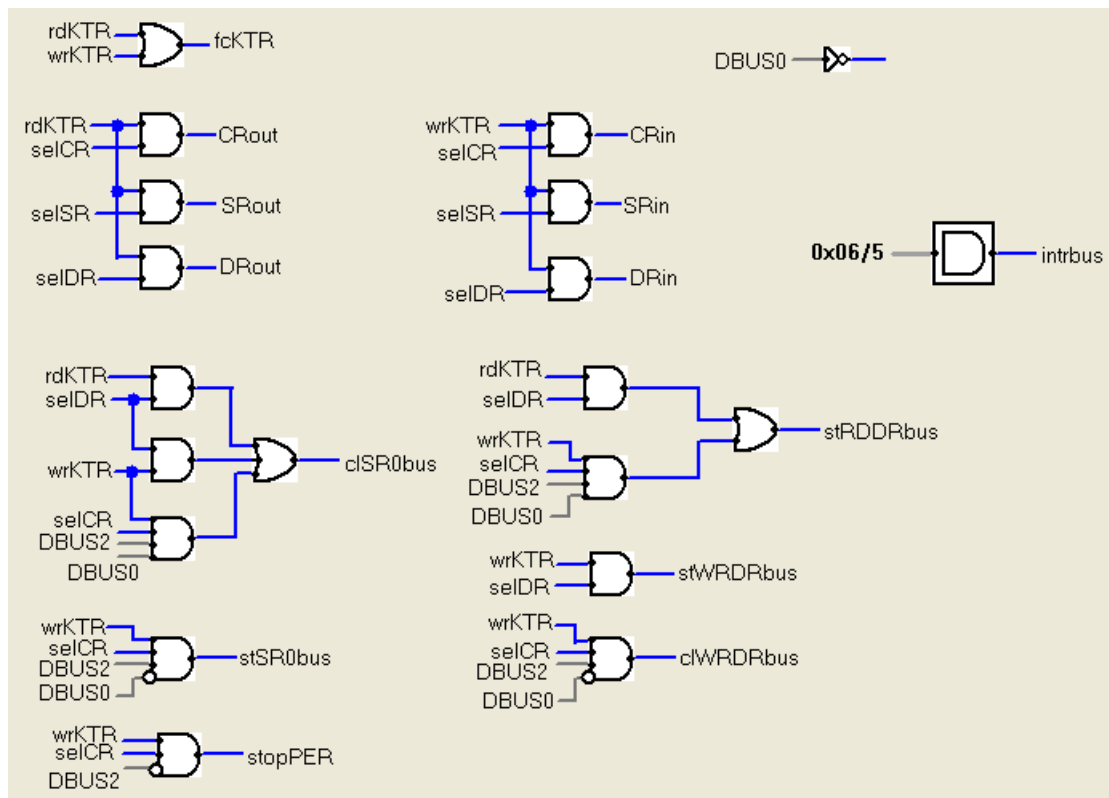
Signali **rdKTR** i **wrKTR** imaju aktivne vrednosti trajanja jedna perioda signala takta. To je posledica usvojenog načina generisanja signala **rdCPU** i **wrCPU** procesora *CPU*, na osnovu kojih se formiraju signali **RDBUS** i **WRBUS** magistrale *BUS* i **rdKTR** i **wrKTR** kontrolera *KP*, i signala **fcKTR**, na osnovu koga se generišu signali **FCBUS** magistrale *BUS* i **fcCPU** procesora *CPU*. Kod čitanja procesor na *i*-ti signal takta ulazi u stanje $step_i$ koje koristi da signal **rdCPU** postavi na aktivnu vrednost, što redom daje i aktivne vrednosti signala **RDBUS**, **rdKTR**, **fcKTR**, **FCBUS** i **fcCPU**. Kako je u procesoru aktivna vrednost signala **fcCPU** uslov za prelazak iz stanja $step_i$ u stanje $step_{i+1}$ i kako ta vrednost dolazi u koraku $step_i$, to procesor na $(i+1)$ -vi signal takta prelazi na korak $step_{i+1}$. Zbog toga se u koraku $step_i$ ostaje samo jedna perioda signala takta, što ima za posledicu da signali **rdCPU**, **RDBUS**, **rdKTR**, **fcKTR**, **FCBUS** i **fcCPU** imaju aktivnu vrednost trajanja jedna perioda signala takta. Iz istih razloga su kod upisa signali **wrCPU**, **WRBUS**, **wrKTR**, **fcKTR**, **FCBUS** i **fcCPU** trajanja jedna perioda signala takta. S obzirom na to da su signali **rdCPU** i **wrCPU** trajanja jedna perioda signala takta i da svi signali koje generiše upravljačke jedinice *uprav_bus* uključuju jedan od signala **rdKTR** i **wrKTR**, to i svi preostali signali imaju aktivne vrednosti trajanja jedna perioda signala takta.

Upravljačka jedinica *uprav_bus* sadrži kombinacione mreže koje pomoću signala **rdKTR** i **wrKTR**, koji dolaze sa bloka *interfejs*, signala logičkih uslova **selCR**, **selSR**, **selDR**, **DBUS₂**, **DBUS₁** i **DBUS₀**, koji dolaze iz bloka *interfejs* i magistrale *BUS*, i saglasno algoritmu generisanja upravljačkih signala (tabela 15) generiše upravljačke signale blokova operacione jedinice.

Upravljački signali blokova operacione jedinice *oper* se daju posebno za blok *registri* i blok *interfejs*.

Upravljački signali bloka *registri* se generišu na sledeći način:

- **CRout** = **rdKTR**·**selCR**
- **SRout** = **rdKTR**·**selSR**
- **DRout** = **rdKTR**·**selDR**
- **clSR0bus** = **rdKTR**·**selDR**
- **stRDDRbus** = **rdKTR**·**selDR**
- **CRin** = **wrKTR**·**selCR**
- **SRin** = **wrKTR**·**selSR**
- **DRin** = **wrKTR**·**selDR**
- **clSR0bus** = **wrKTR**·**selDR**
- **stWRDRbus** = **wrKTR**·**selDR**
- **clSR0bus** = **wrKTR**·**selCR**·**DBUS₂**·**DBUS₀**
- **stRDDRbus** = **wrKTR**·**selCR**·**DBUS₂**·**DBUS₀**
- **stSR0bus** = **wrKTR**·**selCR**·**DBUS₂**·**DBUS₀**
- **clWRDRbus** = **wrKTR**·**selCR**·**DBUS₂**·**DBUS₀**



Slika 29 Struktura upravljačke jedinice *uprav_bus*

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz bloka *interfejs* operacione jedinice *oper* i magistrala *BUS* i to:

- **rdKTR** — bloka *interfejs*.
- **wrKTR** — bloka *interfejs*
- **selCR** — bloka *interfejs*
- **selSR** — bloka *interfejs*
- **selDR** — bloka *interfejs*
- **DBUS₂** — magistrala *BUS* i
- **DBUS₀** — magistrala *BUS*.

Upravljački signali bloka *interfejs* se generišu na sledeći način:

- **intrbus** = $\text{wrKTR} \cdot \text{selCR} \cdot \text{DBUS}_2 \cdot \text{DBUS}_1 \cdot \overline{\text{DBUS}_0}$
- **stopPER** = $\text{wrKTR} \cdot \text{selCR} \cdot \overline{\text{DBUS}_2}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz bloka *interfejs* operacione jedinice *oper* i magistrala *BUS* i to:

- **wrKTR** — bloka *interfejs*
- **selCR** — bloka *interfejs*
- **DBUS₂** — magistrala *BUS*
- **DBUS₁** — magistrala *BUS* i
- **DBUS₀** — magistrala *BUS*.

5.2.2.3 Upravljačka jedinica periferije

U ovom poglavlju se daju dijagram toka operacija, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice.

5.2.2.3.1 Dijagram toka operacija

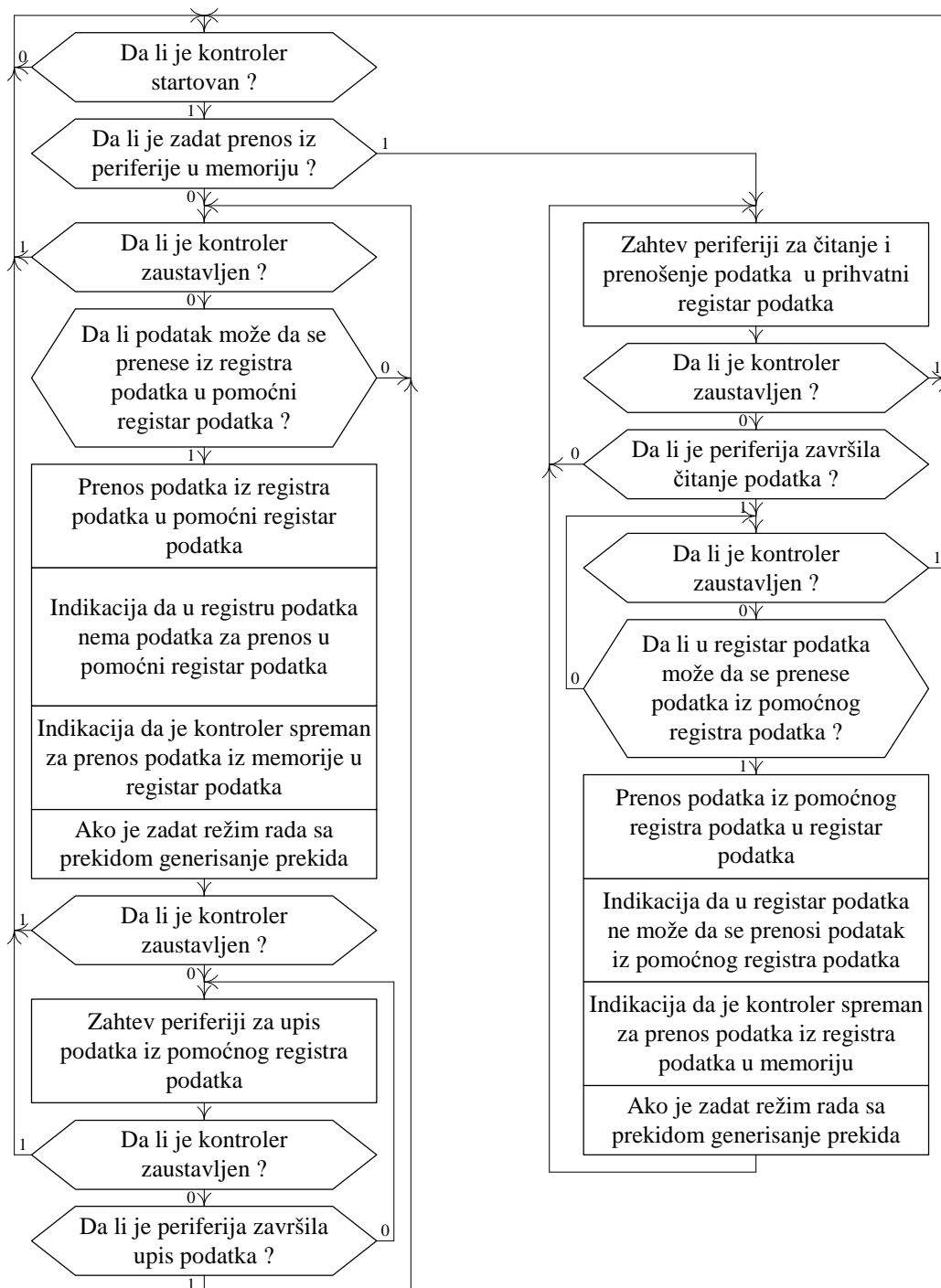
Dijagram toka operacija je predstavljen operacionim i uslovnim blokovima (slika 30). U operacionim blokovima se nalaze opisi mikrooperacija koje treba realizovati. U uslovnim blokovima se nalaze opisi logičkih uslova koji definišu grananja algoritma.

U početnom koraku vrši se provera da li je kontroler startovan ili ne. Ako kontroler nije startovan ostaje se u početnom koraku i čeka startovanje kontrolera. Ako je kontroler startovan vrši se provera da li je zadat prenos iz periferije u memoriju ili iz memorije u periferiju i prelazi na odgovarajući korak.

Ako je zadat prenos iz periferije u memoriju prelazi se na prenošenje podatka iz periferije u kontroler. Najpre se u petlji drži zahtev periferiji da pročita podatak i vrši provera da li je kontroler zaustavljen programskim putem upisivanjem neaktivne vrednosti u razred start upravljačkog registra i da li je periferija završila čitanje podatka. Ukoliko se utvrdi da je kontroler zaustavljen, prekida se prenošenje podataka iz periferije u kontroler, vraća se u početni korak i čeka da se ponovo programskim putem upisivanjem aktivne vrednosti u razred start upravljačkog registra startuje kontroler. Ukoliko se utvrdi da je periferija pročitala podatak izlazi se iz petlje i podatak prenosi iz periferije u pomoćni registar podatka kontrolera.

Potom se u petlji vrši provera da li je kontroler zaustavljen programskim putem i da li je registar podatka raspoloživ da se u njega prenese novi podatak iz pomoćnog registra podatka. Ukoliko se utvrdi da je kontroler zaustavljen, prekida se prenošenje podataka iz periferije u kontroler, vraća se u početni korak i čeka da se ponovo programskim putem startuje kontroler. Ukoliko se utvrdi da je registar podatka raspoloživ, izlazi se iz petlje i podatak prenosi iz pomoćnog registra podatka u registar podatka. Istovremeno se postavlja indikacija da registar podatka nije raspoloživ da se u njega prenese novi podatak iz pomoćnog registra podatka sve dok se programskim putem podatak ne prenese iz registra podatka u memoriju. Pored toga bit spremnost statusnog registra kontrolera se postavlja na aktivnu vrednost, čime se postavlja indikacija da je kontroler spreman za prenos podatka iz registra podatka u memoriju, i generiše prekid, ukoliko je pri startovanju zadat režim sa generisanjem prekida. Na kraju se vraća na korak u kome se postavlja zahtev periferiji da pročita sledeći podatak. Opisane aktivnosti se ponavljaju sve dok se kontroler ne zaustavi programskim putem.

Ako je u zadat prenos iz memorije u periferiju prelazi se na prenošenje podatka iz kontrolera u periferiju. Najpre se u petlji vrši provera da li je kontroler zaustavljen programskim putem i da li je registar podatka raspoloživ da se iz njega prenese novi podatak u pomoćni registar podatka. Ukoliko se utvrdi da je kontroler zaustavljen, prekida se prenošenje podataka iz kontrolera u periferiju, vraća se u početni korak i čeka da se ponovo programskim putem, startuje kontroler. Ukoliko se utvrdi da je registar podatka raspoloživ izlazi se iz petlje i podatak prenosi iz registra podatka u pomoćni registar podatka. Istovremeno se postavlja indikacija da registar podatka nije raspoloživ da se iz njega prenosi podatak u pomoćni registar podatka sve dok se programskim putem podatak ne prenese iz memorije u registar podatka. Pored toga na aktivnu vrednost se postavlja bit spremnost statusnog registra kontrolera, čime se postavlja indikacija da je kontroler spreman za prenos podatka iz memorije u registar podatka, i generiše prekid, ukoliko je pri startovanju zadat režim sa generisanjem prekida.



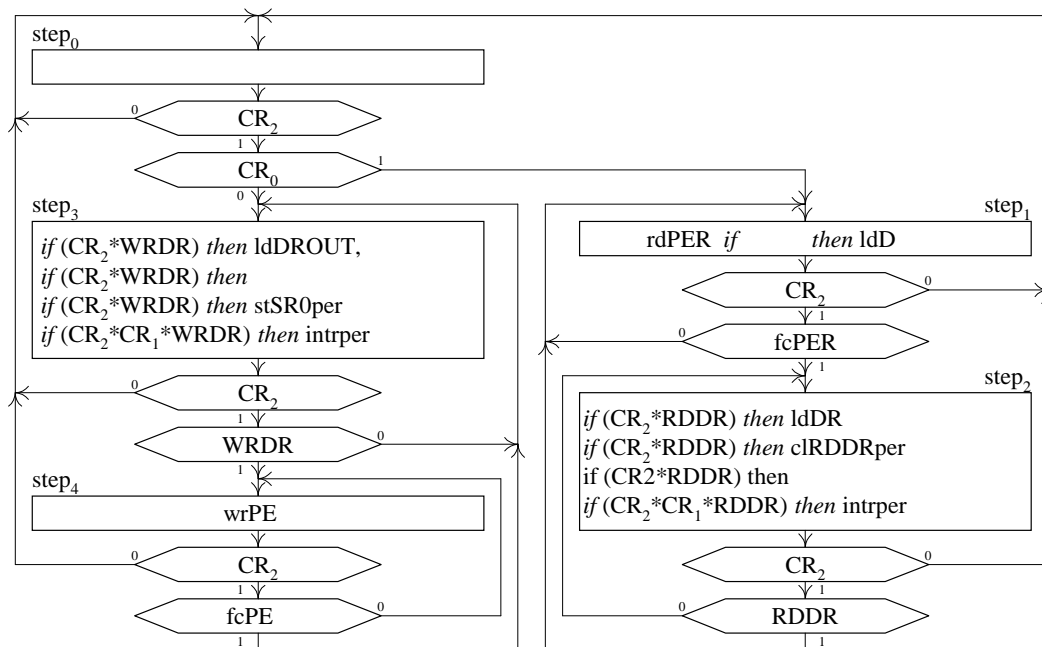
Slika 30 Dijagram toka operacija

Zatim se vrši provera da li je kontroler zaustavljen programskim putem. Ukoliko se utvrdi da je kontroler zaustavljen, prekida se prenošenje podataka iz kontrolera u periferiju, vraća se u početni korak i čeka da se ponovo programskim putem startuje kontroler. Ukoliko se utvrdi da kontroler nije zaustavljen u petlji se periferiji drži zahtev da upiše sledeći podatak i vrše provere da li je kontroler zaustavljen programskom putem i da li je periferija završila upis podatka. Ukoliko se utvrdi da je kontroler periferije zaustavljen, prekida se prenošenje podataka iz kontrolera u periferiju, vraća se u početni korak i čeka da se ponovo programskim putem startuje kontroler. Ukoliko se utvrdi da je periferija završila upis podatka iz pomoćnog registra podatka kontrolera, izlazi se iz ove petlje i vraća u petlju u kojoj se vrši provera da li je kontroler zaustavljen programskim putem i da li je registar podatka kontrolera raspoloživ da

se iz njega prenese novi podatak u pomoćni registar podatka kontrolera. Opisane sktivnosti se ponavljaju sve dok se kontroler ne zaustavi programskim putem.

5.2.2.3.2 Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu dijagrama toka operacija (slika 30) i dat u obliku dijagrama toka upravljačkih signala (slika 31) i sekvence upravljačkih signala po koracima (tabela 16).



Slika 31 Dijagram toka upravljačkih signala

Dijagram toka upravljačkih signala je predstavljen operacionim i uslovnim blokovima (slika 31). U operacionim blokovima se nalaze upravljački signali i uslovi pod kojima se oni generišu. U uslovnim blokovima se nalaze signali logičkih uslova.

U sekvenci upravljačkih signala po koracima je za svaki korak data simbolička oznaka samog koraka i iskazi za signale i skokove. Iskazi za signale se pojavljuju ukoliko u datom koraku treba da se generiše neki od upravljačkih signala operacione jedinice. Iskazi za signale sadrže spisak upravljačkih signala operacione jedinice koji se generišu bezuslovno i uslovno, a za signale koji se generišu uslovno i signale logičkih uslova pod kojima se signali generišu. Iskazi za skokove se pojavljuju ukoliko u treba odstupiti od sekvencijalnog generisanja upravljačkih signala operacione jedinice. Iskazi za skokove sadrže uslov i korak i određuje na koji korak i pod kojim uslovima treba preći.

Objašnjenja vezana za generisanje upravljačkih signala su data zajednički za dijagram toka upravljačkih signala i sekvencu upravljačkih signala i to u okviru sekvence upravljačkih signala.

Tabela 16 Sekvenca upravljačkih signala

! U koraku se step₀ se ostaje i ne generiše se aktivna vrednost ni jednog od upravljačkih signala sve dok je signal **CR₂** bloka *registri* neaktivan. Signal **CR₂** postaje aktivan tek kada se programskim putem startuje kontroler periferije. Tom prilikom se u upravljački registar **CR_{2...0}** bloka *registri* upisuje sadržaj koji na poziciji razreda **CR₂** ima aktivnu vrednost. Na poziciji razreda **CR₀** je aktivna ili neaktivna vrednost i zavisnosti od toga da li je zadat prenos iz periferije u memoriju ili iz memorije

u periferiju, respektivno. Na poziciji razreda CR_1 je aktivna ili neaktivna vrednost i zavisnosti od toga da li je zadat režim rada sa generisanjem ili bez generisanja prekida, respektivno. Pri aktivnim vrednostima signala CR_2 i CR_0 se prelazi na korak $step_1$ koji odgovara prenosu iz periferije u memoriju. Pri aktivnoj i neaktivnoj vrednosti signala CR_2 i CR_0 , respektivno, se prelazi na korak $step_3$, koji odgovara prenosu iz memorije u periferiju. !

$step_0$ *br* (if ($CR_2 \cdot CR_0$) then $step_1$),
 br (if ($CR_2 \cdot \overline{CR_2}$) then $step_3$);

! U korak $step_1$ se dolazi iz koraka $step_0$ ili koraka $step_2$. U ovom koraku se generiše signal **rdPER** koji predstavlja zahtev periferiji da pročita sledeći podatak. Ukoliko je neaktivna vrednost signala CR_2 , što znači da je programskim putem upisivanjem neaktivne vrednosti u razred CR_2 upravljačkog registra CR zaustavljen rad kontrolera, prelazi se u korak $step_0$. Ukoliko je aktivna vrednost signala CR_2 , ili se ostaje u koraku $step_1$ ili se prelazi u korak $step_2$ u zavisnosti od toga da li je neaktivna ili aktivna vrednost signala **fcPERrd** periferije *PER*, respektivno. Neaktivna vrednost signala **fcPERrd** je indikacija da čitanje podatka u periferiji još uvek traje, a aktivna vrednost da je podatak pročitani i da se nalazi u registru $DRIN_{7...0}$.!

$step_1$ **rdPER**,
 br (if $\overline{CR_2}$ then $step_0$),
 br (if ($CR_2 \cdot fcPERrd$) then $step_2$);

! U korak $step_2$ se dolazi iz koraka $step_1$. Ukoliko je neaktivna vrednost signala CR_2 prelazi se u korak $step_0$. Ukoliko je aktivna vrednost signala CR_2 , ili se ostaje u koraku $step_2$ ili se prelazi u korak $step_1$ u zavisnosti od toga da li je neaktivna ili aktivna vrednost signal **RDDR** bloka *registri*, respektivno. Neaktivna vrednost signala **RDDR** je indikacija da registar podatka $DR_{7...0}$ nije raspoloživ, jer prethodni podatak još uvek nije programskim putem prenet iz registra $DR_{7...0}$ u memoriju. Aktivna vrednost signala **RDDR** je indikacija da je registar podatka $DR_{7...0}$ raspoloživ, pa se generišu signali **ldDR**, **clRDDRper** i **stSR0per**. Signalom **ldDR** se sadržaj registra $DRIN_{7...0}$ bloka *registri* propušta kroz multiplekser $MP1$ bloka *registri* i upisuje u registar podatka $DR_{7...0}$. Signalom **clRDDRper** se u flip-flop **RDDR** upisuje neaktivna vrednost. Ova vrednost je indikacija da registar $DR_{7...0}$ nije raspoloživ da se u njega prenese novi podatak iz registra $DRIN_{7...0}$ sve dok se programskim putem prethodni podatak ne prenese iz registra $DR_{7...0}$ u memoriju. Signalom **stSR0per** se upisuje aktivna vrednost u razred SR_0 statusnog registra SR_0 bloka *registri*. Ovaj razred odgovara bitu spremnost i njegova aktivna vrednost pri prenosu iz periferije u memoriju služi kao indikacija procesoru da može programskim putem da prenese podatak iz registra $DR_{7...0}$ u memoriju. Ukoliko je pri aktivnim vrednostima signala CR_2 i **RDDR** i signal CR_1 aktivan, generiše se signal **intrper** bloka *interfejs*. Ovim signalom se pri prenosu iz periferije u memoriju generiše signala prekida **intr**, da bi procesor skočio na odgovarajuću prekidnu rutinu i u okviru nje programskim putem preneo podatak iz registra $DR_{7...0}$ u memoriju. !

$step_2$ *if* ($CR_2 \cdot RDDR$) then (**ldDR**, **stSR0per**, **clRDDRper**),
 if ($CR_2 \cdot CR_1 \cdot RDDR$) then **intrper**,
 br (if $\overline{CR_2}$ then $step_0$),
 br (if ($CR_2 \cdot RDDR$) then $step_1$);

! U korak $step_3$ se dolazi iz koraka $step_0$ ili koraka $step_4$. Ukoliko je neaktivna vrednost signala CR_2 prelazi se u korak $step_0$. Ukoliko je aktivna vrednost signala CR_2 , ili se ostaje u koraku $step_3$ ili se prelazi u korak $step_4$ u zavisnosti od toga da li je neaktivna ili aktivna vrednost signal **WRDR** bloka *registri*, respektivno. Neaktivna vrednost signala **WRDR** je indikacija da registar podatka $DR_{7...0}$ bloka *registri* nije raspoloživ, jer novi podatak još uvek nije programskim putem prenet iz memorije u registar $DR_{7...0}$. Aktivna vrednost signala **WRDR** je indikacija da je registar $DR_{7...0}$ raspoloživ jer je podatak programskim putem prenet iz memorije u registar podatka $DR_{7...0}$. Po upisu podatka u registar $DR_{7...0}$ upravljačka jedinica *uprav_bus* signalom **stWRDRbus** postavlja flip-flop **WRDR** na aktivnu vrednost. Pri aktivnoj vrednosti signala **WRDR** generišu se signali **ldDROUTin**, **clWRDRper** i **stSR0per**. Aktivnom vrednošću signala **ldDROUT** se sadržaj registra $DR_{7...0}$ registar podatka $DROUT_{7...0}$ bloka *registri*. Signalom **clWRDRper** se u flip-flop **WRDR** upisuje neaktivna vrednost. Ova vrednost je indikacija da registar $DR_{7...0}$ nije raspoloživ i da njegov sadržaj ne može da se prenese

u registar DROUT_{7...0} sve dok se programskim putem novi podatak ne prenese iz memorije u registar DR_{7...0}. Signalom **stSR0per** se upisuje aktivna vrednost u razred SR₀ statusnog registra SR₀ bloka *registri*. Ovaj razred odgovara bitu spremnost i njegova aktivna vrednost pri prenosu iz memorije u periferiju služi kao indikacija procesoru da može programskim putem da prenese podatak iz memorije u registar DR_{7...0}. Ukoliko je pri aktivnim vrednostima signala **CR₂** i **WRDR** i signal **CR₁** aktivan, generiše se signal **intrper** bloka *interfejs*. Ovim signalom se pri prenosu iz memorije u periferiju generiše signal prekida **intr**, da bi procesor skočio na odgovarajuću prekidnu rutinu i u okviru nje programskim putem preneo podatak iz memorije u registar DR_{7...0}. !

```

step3  if (CR2·WRDR) then (ldDROUT, clWRDRper, stSR0per),
        if (CR2·CR1·WRDR) then intrper,
        br (if CR2 then step0),
        br (if (CR2·WRDR) then step4);

```

! U korak step₄ se dolazi iz koraka step₃. U ovom koraku se realizuje upis u periferiju. Ukoliko je neaktivna vrednost signala **CR₂** prelazi se u korak step₀. Ukoliko je aktivna vrednost signala **CR₂**, ili se ostaje u koraku step₄ ili se prelazi u korak step₃ u zavisnosti od toga da li je neaktivna ili aktivna vrednost signal **fcPERwr** periferije *PER*, respektivno. Neaktivna vrednost signala **fcPERwr** je indikacija da je upis podatka iz registra DROUT_{7...0} u periferiju u toku, a aktivna vrednost da je podatak upisan. !

```

step4  wrPER,
        br (if CR2 then step0),
        br (if (CR2·fcPERwr) then step3);

```

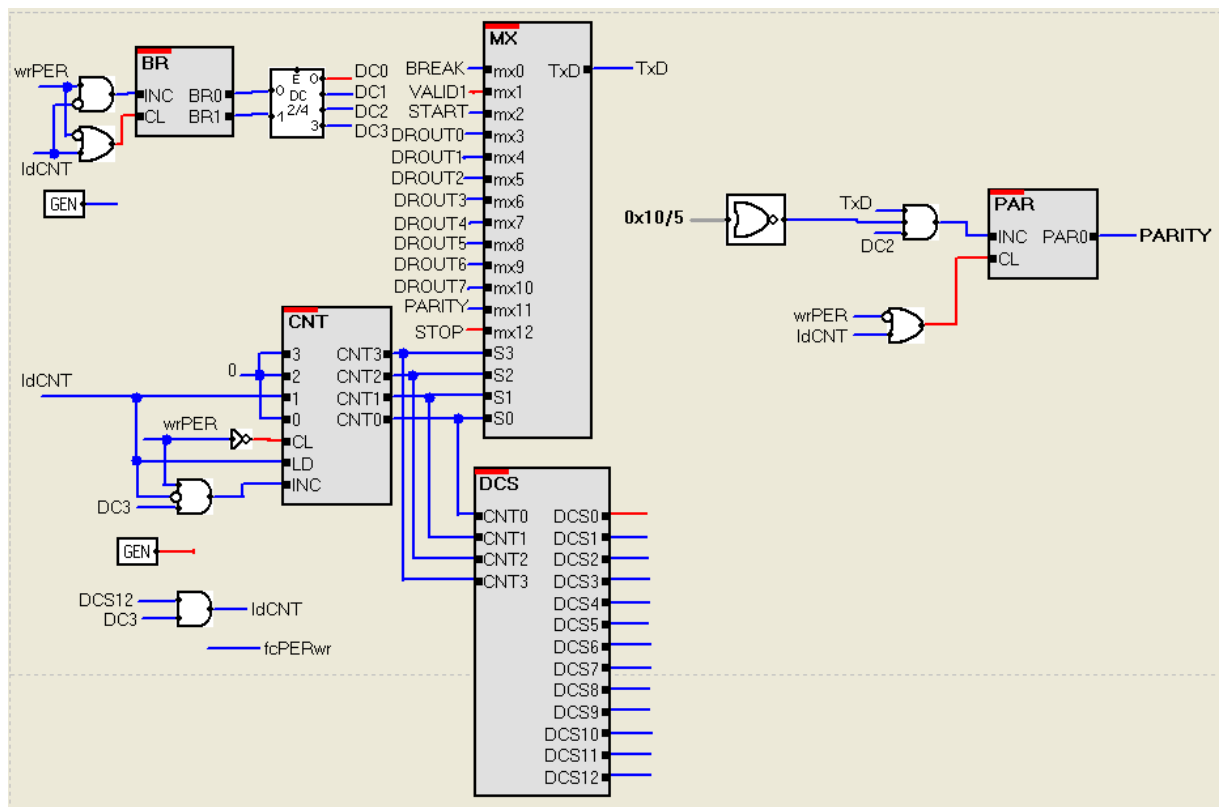
5.2.2.3.3 Struktura upravljačke jedinice

Upravljačka jedinica (slika 32) se sastoji od sledećih blokova:

- blok *generisanje nove vrednosti brojača koraka*,
- blok *brojač koraka*,
- blok *dekoder koraka*,
- blok *generisanje upravljačkih signala*,
- blok *upravljačka jedinica ulazne periferije* i
- blok *upravljačka jedinica izlazne periferije*.

Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.

Blok *generisanje nove vrednosti brojača koraka* služi za generisanje vrednosti koju treba upisati u brojač CNT. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog generisanja upravljačkih signala. Analizom algoritma generisanja upravljačkih signala operacione jedinice (poglavlje 4.4.2.2) se utvrđuje da su 0, 1 i 3 vrednosti koje treba upisati u brojač CNT da bi se realizovala odstupanja od sekvencijalnog generisanja upravljačkih signala. Te vrednosti se formiraju na ulazima 2, 1 i 0 brojača CNT pomoću signala **val₀**, **val₁** i **val₃** pri čemu signal **val₀** ima vrednost 1 samo onda kada treba upisati 0, signal **val₁** ima vrednost 1 samo onda kada treba upisati 1 i signal **val₃** ima vrednost 1 samo onda kada treba upisati 3. Time se obezbeđuje da na ulazima 2, 1 i 0 brojača CNT budu vrednosti 0, 0 i 0 onda kada signal **val₀** ima vrednost 1, vrednosti 0, 0 i 1 onda kada signal **val₁** ima vrednost 1 i vrednosti 0, 1, i 1 onda kada signal **val₃** ima vrednost 1.



Slika 34 Struktura upravljačke jedinice izlazne periferije

Blok *brojač koraka* sadrži brojač CNT. Brojač CNT svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač CNT može da radi u sledećim režimima:

- režim inkrementiranja,
- režim skoka i
- režim mirovanja.

U režimu inkrementiranja pri pojavi signala takta vrši se uvećavanje sadržaja brojača CNT za jedan. Ovim režimom se obezbeđuje sekvencijalno generisanje upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 5.2.2.3.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **incCNT**. Signal **incCNT** je uvek neaktivan sem kada treba obezbediti režim inkrementiranja.

U režimu skoka pri pojavi signala takta vrši se upis nove vrednosti u brojač CNT. Ovim režimom se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 5.2.2.3.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **ldCNT**. Signal **ldCNT** je uvek neaktivan sem kada treba obezbediti režim skoka.

U režimu mirovanja pri pojavi signala takta ne menja se vrednost brojača CNT. Ovaj režim rada se obezbeđuje neaktivnim vrednostima signala **incCNT** i **ldCNT**. Ovi signali su neaktivni kada se u koraku

- step₀ pri aktivnoj vrednosti signala T₀ čeka da kontroler bude startovan upisivanjem aktivne vrednosti u razred CR₂ upravljačkog registra kontrolera,
- step₁ pri aktivnoj vrednosti signala T₁ čeka da čitanje u periferiji bude završeno i signal fcPERrd postane aktivan,

- step₂ pri aktivnoj vrednosti signala T₂ čeka da registar podatka DR_{7...0} bude raspoloživ za prebacivanje podatka iz pomoćnog registra podatka DRIN_{7...0} i signal RDDR postane aktivan,
- step₃ pri aktivnoj vrednosti signala T₃ čeka da pomoćni registar podatka DROUT_{7...0} bude raspoloživ za prebacivanje podatka iz registra podatka DR_{7...0} i signal WRDR postane aktivan i
- step₄ pri aktivnoj vrednosti signala T₄ čeka da upis u periferiju bude završen i signal fcPERwr postane aktivan.

Blok *dekoder koraka* sadrži dekodera DC. Na ulaze dekodera DC vode se izlazi brojača CNT. Dekodovana stanja brojača CNT pojavljuju se kao signali T₀ do T₇ na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 4.4.2.2) dodeljen je jedan od ovih signala i to koraku step₀ signal T₀, koraku step₁ signal T₁, itd.

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala T₀ do T₇ koji dolaze sa bloka *dekoder koraka* upravljačke jedinice, signala logičkih uslova koji dolaze iz blokova operacione jedinice i saglasno algoritmu generisanja upravljačkih signala (poglavlje 5.2.2.3.2) generišu upravljačke signale. Upravljački signali se generišu na identičan način kao i upravljački signali procesora CPU (poglavlje 5.2.2.3.2).

Blok *generisanje upravljačkih signala* generiše tri grupe upravljačkih signala i to:

- upravljačke signale periferije *PER*,
- upravljačke signale blokova operacione jedinice *oper* i
- upravljačke signale upravljačke jedinice *uprav_per*.

Upravljački signali periferije *PER* se generišu na sledeći način:

- $rdPER = CR_2 \cdot T_1 \cdot \overline{stopPER}$
- $wrPER = CR_2 \cdot T_4 \cdot \overline{stopPER}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- CR₂ — blok *registri*.

Upravljački signali blokova operacione jedinice *oper* se daju posebno za blok *registri* i blok *interfejs*.

Upravljački signali bloka *registri* se generišu na sledeći način:

- $stSR0per = CR_2 \cdot RDDR \cdot T_2 + CR_2 \cdot WRDR \cdot T_3$
- $clRDDRper = CR_2 \cdot RDDR \cdot T_2$
- $clWRDRper = CR_2 \cdot WRDR \cdot T_3$
- $ldDR = CR_2 \cdot RDDR \cdot T_2$
- $ldDROUT = CR_2 \cdot WRDR \cdot T_3$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- CR₂ — blok *registri*,
- RDDR — blok *registri*,
- WRDR — blok *registri*.

Upravljački signali bloka *interfejs* se generiše na sledeći način:

- $intrper = CR_2 \cdot CR_1 \cdot RDDR \cdot T_2 + CR_2 \cdot CR_1 \cdot WRDR \cdot T_3$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i periferije *PER* i to:

- **CR₂** — blok *registri*,
- **CR₁** — blok *registri*,
- **RDDR** — blok *registri*,
- **WRDR** — blok *registri*.

Upravljački signali upravljačke jedinice *uprav_per* se generiše na sledeći način:

- **ldCNT** = **val0** + **val1** + **val3**
- **val0** = $\overline{\text{CR}_2} \cdot (\text{T}_1 + \text{T}_2 + \text{T}_3 + \text{T}_4)$
- **val1** = $\text{CR}_2 \cdot \text{RDDR} \cdot \text{T}_2$
- **val3** = $\text{CR}_2 \cdot \overline{\text{CR}_0} \cdot \text{T}_0 + \text{CR}_2 \cdot \text{fcPERwr} \cdot \text{T}_4$
- **incCNT** = $\text{CR}_2 \cdot \text{CR}_0 \cdot \text{T}_0 + \text{CR}_2 \cdot \text{fcPERrd} \cdot \text{T}_1 + \text{CR}_2 \cdot \text{WRDR} \cdot \text{T}_3$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz upravljačke jedinice *uprav* i pojedinih blokova operacione jedinice *oper* i to:

- **fcPERrd** — blok *upravljačka jedinica ulazne periferije*,
- **fcPERwr** — blok *upravljačka jedinica izlazne periferije*,
- **CR₀** — blok *registri*,
- **CR₂** — blok *registri*,
- **RDDR** — blok *registri* i
- **WRDR** — blok *registri*.

Upravljački signali blokova *upravljačka jedinica ulazne periferije* i *upravljačka jedinica izlazne periferije* upravljačke jedinice *uprav_per* se generišu na sledeći način:

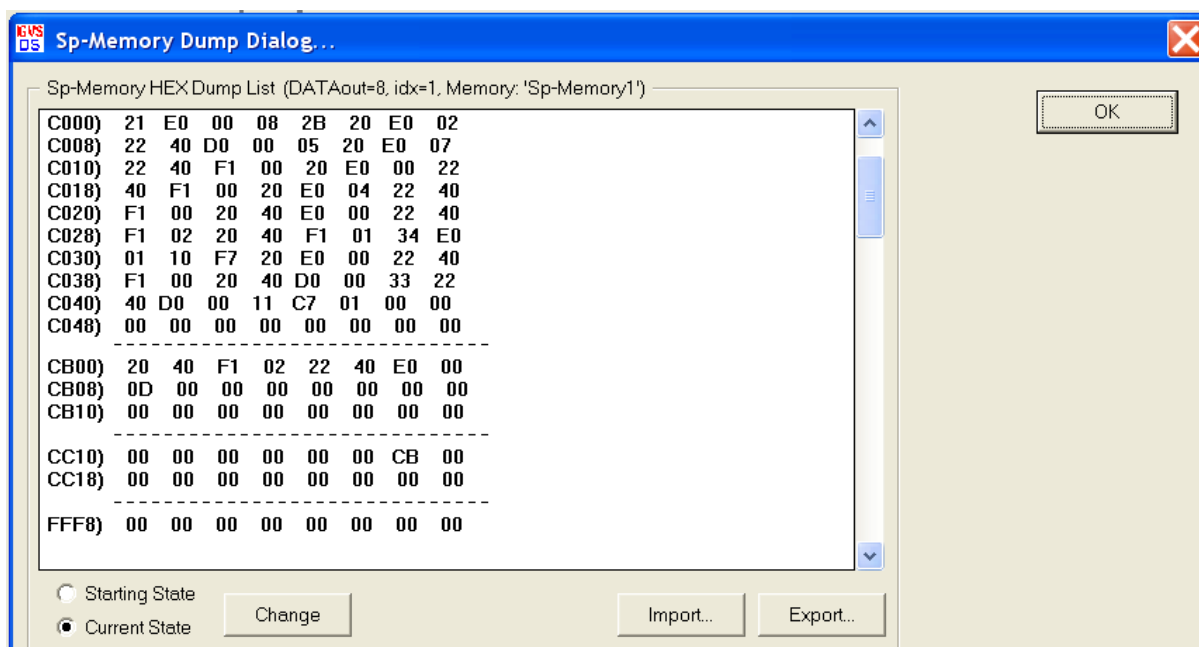
- **fcPERrd** = DCS13 * DC3
- **fcPERwr** = ldCNT

6 LABORATORIJA

U ovom poglavlju će biti prikazan i detaljno opisan primer programa koji se izvršava na ovako konstruisanom simulatoru i ilustruje ulaz sa periferije upotrebom kontrolera sa serijskim prenosom sa generisanjem prekida i izlaz na periferiju upotrebom kontrolera sa serijskim prenosom ispitivanjem bita spremnosti. Time će biti prikazana funkcionalnost samog sistema ali i simulatora koji je konstruisan u programskom paketu *IGoVSoDS*. Preduslov za izvršavanje ovog programa je da su memorija računara i registri opše namene procesora inicijalizovani odgovarajućim vrednostima.

6.1 Inicijalizacija programa

Da bi se program izvršavao, na početku se mora uraditi inicijalizacija najvažnijih registara u procesoru. Kod programa čija će realizacija biti prikazana, od važnosti je jedino da programski brojač bude inicijalizovan. Pošto je zabranjeno koristiti adrese iz opsega *F000h-FFFFh* jer su rezervisane za periferije, uzeto je da se adresa početka izvršavanja programa (prve instrukcije) nalazi na adresi *C000h*, tako da je registar PC inicijalizovan ovom vrednošću. Dalje, mora se izvršiti inicijalizacija memorije, počev od adrese *C000h* odgovarajućim podacima koji će predstavljati instrukcije koje treba izvršavati. Takođe kako se u primeru koristi prekid mora se inicijalizovati ulaz 11 u tabeli prekidnih rutina, odnosno adresa *CC16h* vrednošću koja predstavlja adresu prve instrukcije prekidne rutine koja je u ovom slučaju *CB00h*. Dalje moraju se inicijalizovati i memorijske lokacije koje će predstavljati datu prekidnu rutinu. U nastavku, na slikama 65, 66 i 67 dat je primer programa u assembleru i prikaz memorije koja je inicijalizovana odgovarajućim vrednostima na osnovu algoritma koji se nalazi u glavi 2.



Slika 65 Sadržaj memorije nakon inicijalizacije

C000	LDW 08h	Acc16=08	20 E0 00 08
C004	STIMR	IMR=Acc16	2B
C005	LDB 02h	Acc8=02	20 E0 02

C008	STB D0 00	Mem[D000]=Acc8	20 40 D0 00
C00C	INTE	PSWI=1	5
C00D	LDB 07h	Acc8=07	20 E0 07
C010	STB F1 00	Mem[F100]=Acc8	22 40 F1 00
C014	LDB 00h	Acc8=00	20 E0 00
C017	STB F1 00	Mem[F100]=Acc8	22 40 F1 00
C01B	LDB 04h	Acc8=04	20 E0 04
C01E	STB F1 00	Mem[F100]=Acc8	22 40 F1 00
C022	LDB E0 00	Acc8=Mem[E000]	20 40 E0 00
C026	STB F1 02	Mem[F102]=Acc8	22 40 F1 02
C02A	LDB F1 01	Acc8=Mem[F101]	20 40 F1 01
C02E	AND 1	Acc8=Acc8 AND 01	34 E0 00
C031	BEQL F7	skok na adresu C02A	10 F7
C033	LDB 00h	Acc8=00	20 E0 00
C036	STB F1 00	Mem[F100]=Acc8	22 40 F1 00
C03A	LDB D0 00	Acc8=Mem[D000]	20 40 D0 00
C03E	DEC	Acc8=Acc8 - 1	33
C03F	STB D0 00	Mem[D000]=Acc8	22 40 D0 00
C043	BNEQ C7	skok na adresu C00C	11 C7
C045	HALT	zaustavljanje CPU	1

Slika 66 program koji se izvršava

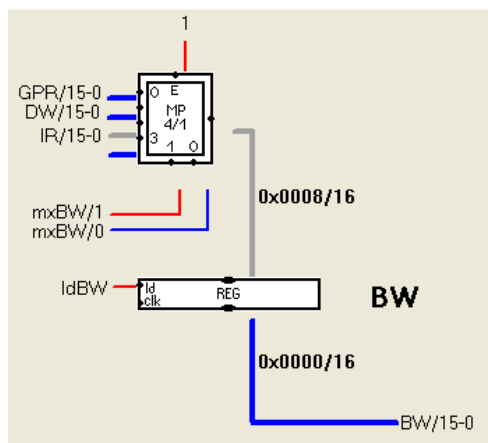
CB00	LDB F1 02	Acc8=Mem[F102]	20 40 F1 02
CB04	STB E0 00	Mem[E000]=Acc8	22 40 E0 00
CB08	RTI	povratak iz prekidne rutine	OD

Slika 67 prekidna rutina na adresi *CB00h* koja se izvršava

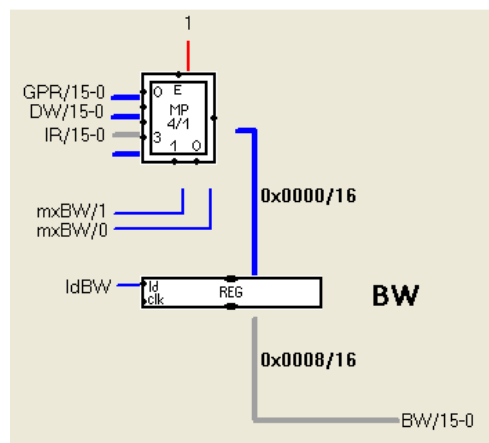
6.2 Simulacija na nivou instrukcija

Kako je cilj ovog diplomskog bio projektovanje i simulacija kontrolera za serijski prenos podataka u daljem tekstu se daje opis simulacije samo kontrolera uz osvrt na neke važne delove ostatka sistema dok se opis ostalih delova ovog sistema preskače. Opis izvršavanja programa daje se na instrukcijskom nivou.

Prva instrukcija *LDW 08h* vrši upis vrednosti 08h u akumulator *BW*. U trenutku $T=41$ aktivnom vrednostima signala *ldBW* i *mxBW/1* vrši se upis vrednosti 08h u akumulator *BW*. U trenutku $T=42$ vrednost 08h je upisana u akumulator.

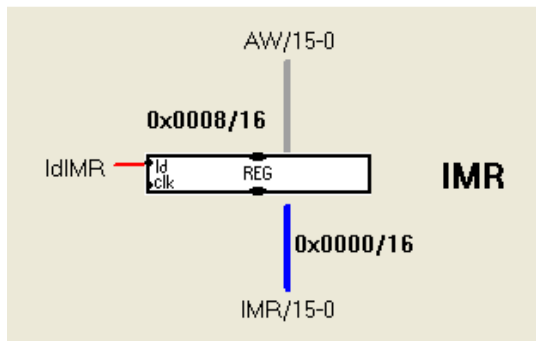


Slika 68 $T=41$ upis 08h u akumulator

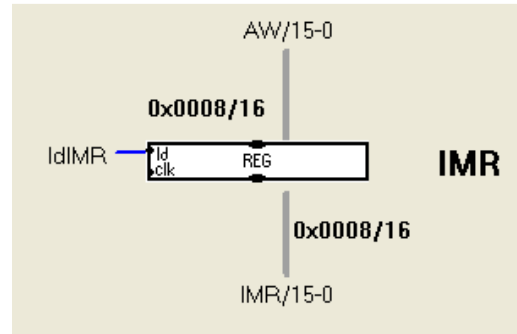


Slika 69 $T=42$ upisana vrednost u akumulatoru

Druga instrukcija *STIMR* vrši upis sadržaja akumulatora *AB* u registar *IMR* čime se dozvoljavaju maskirajući prekidi po liniji *intr₃* koje šalje kontroler za serijski prenos. U trenutku T=57 aktivnim signalom *ldIMR* vrši se upis vrednosti 08h u registar maske *IMR*. U trenutku T=58 vrednost 08h je upisana u registar *IMR*.

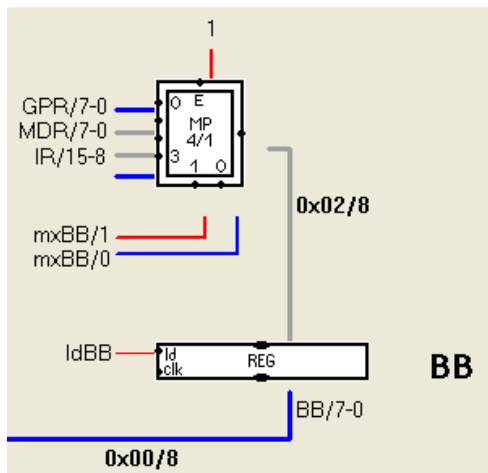


Slika 70 T=57 upis 08h u IMR

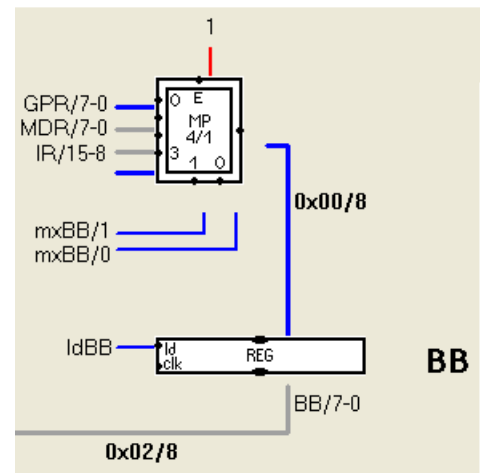


Slika 71 T=58 upisana vrednost u IMR

Treća instrukcija *LDB 02h* vrši upis vrednosti 02h, koja predstavlja broj podataka za prenos, u akumulator *AB*. U trenutku T=92 aktivnim vrednostima signala *mxBB/1* i *ldBB* upisuje se vrednost 02h u akumulator *BB*. U trenutku T=93 u akumulatoru se nalazi vrednost 02h.

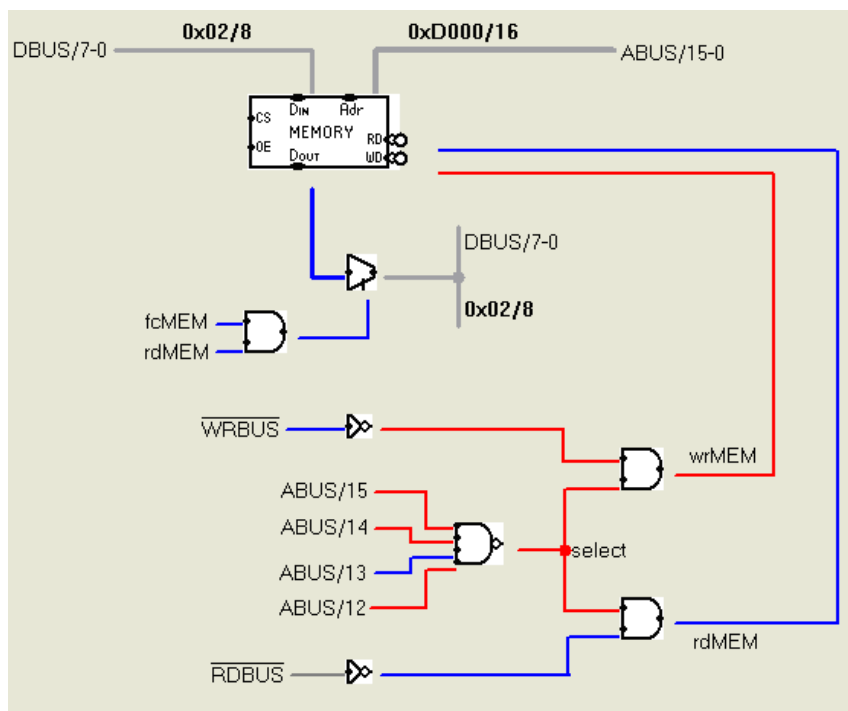


Slika 71 T=92 upis vrednosti 02h u akumulator



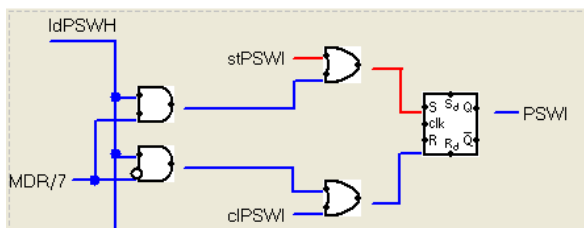
Slika 72 T=93 upisana vrednost u akumulator

Četvrta instrukcija *STB D0 00* vrši upis sadržaja akumulatora *AB* na memorijsku lokaciju *D0 00* koja će se koristiti za brojanje prenetih podataka. U trenutku T=142 neaktivnom vrednošću signala *WRBUS* vrši se upis vrednosti 02h, iz akumulatora, na adresu *D000* u memoriju.

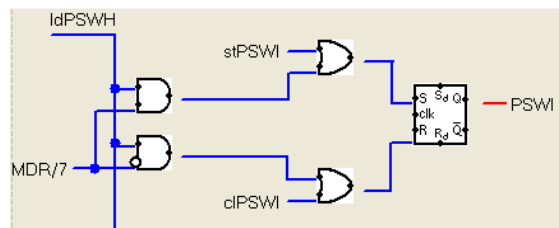


Slika 73 T=142 upis vrednosti 02h u memoriju na adresi D000h

Peta instrukcija *INTE* upisom vrednosti 1 u bit *I* registra *PSW* dozvoljava prekide. U trenutku T=161 aktivnom vrednošću signala *stPSWI* bit *PSWI* registra *PSW* se postavlja na vrednost 1 čime se omogućavaju prekidi. U trenutku T=162 bit *PSWI* je postavljen na vrednost 1.



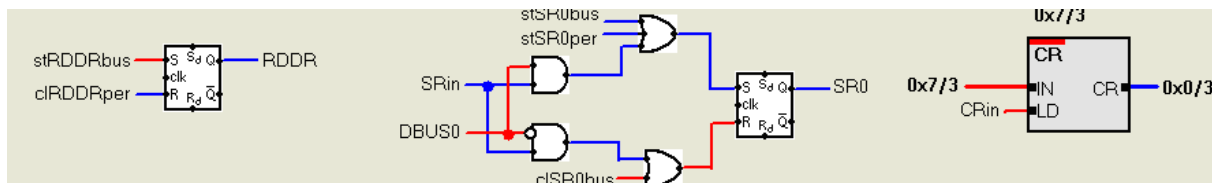
Slika 74 T=161 postavljanje bita *PSWI* na 1



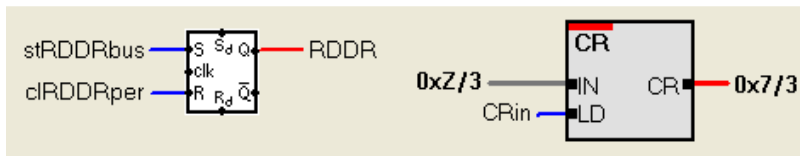
Slika 75 T=162 bit *PSWI* je postavljen na 1

Šesta instrukcija *LDB 07h* vrši upis vrednosti 07h u akumulator *AB*. Ova vrednost se koristi za programiranje kontrolera za rad sa ulaznom periferijom u režimu sa generisanjem prekida. U trenutku T=196 aktivnim vrednostima signala *mxBB/1* i *ldBB* upisuje se vrednost 07h u akumulator *BB*. U trenutku T=197 u akumulatoru se nalazi vrednost 07h.

Sedma instrukcija *STB F1 00* vrši upis sadržaja akumulatora u memorijsku lokaciju *F1 00* koja predstavlja adresu kontrolnog registra kontrolera periferije za serijski prenos. U trenutku (T=246) na sistemskoj magistrali se nalazi vrednost *F100* kojim se selektuje kontrolni registar kontrolera periferije i signal *selCR* postaje aktivan, dok neaktivnom vrednošću na liniji *WRBUS* postaje aktivan signal *wrKTR* (u bloku uprav_bus). Ovim aktivnim signalima postaje aktivan signal *CRin* kojim se vrši upis vrednosti 07h sa magistrale podataka *DBUS* u kontrolni registar *CR* čime je kontroler programiran za čitanje iz periferije sa generisanjem prekida, odnosno za rad sa ulaznom periferijom. Takođe aktivni su i signali *stRDDRbus*, *clSR0bus* i *fcKTR*. Aktivnom vrednošću signala *stRDDRbus* postaje aktivan signal *RDDR*.

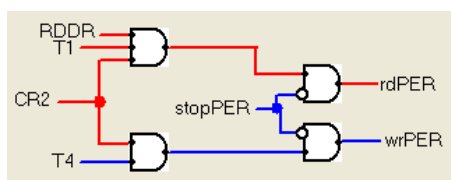


Slika 76 T=246 upis vrednosti 07h u kontrolni registar CR kontolera periferije

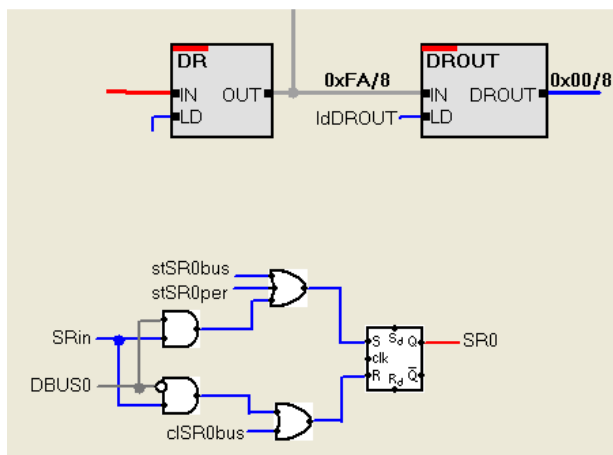


Slika 77 T=247 upisana vrednost u kontrolni registar CR kontolera periferije

Signalom *stRDDRbus* u trenutku (T=247) postaje sktivan signal *RDDR* takođe postaje aktivan i signal *incCNT* upravljačke jedinice periferije kojim se inkrementira brojač *CNT* upravljačke jedinice periferije *uprav_per* koji prelazi u stanje T1 u trenutku(T=248). U ovom stanju se generiše aktivna vrednost signala *rdPER* kojim se startuje čitanje u periferiji.

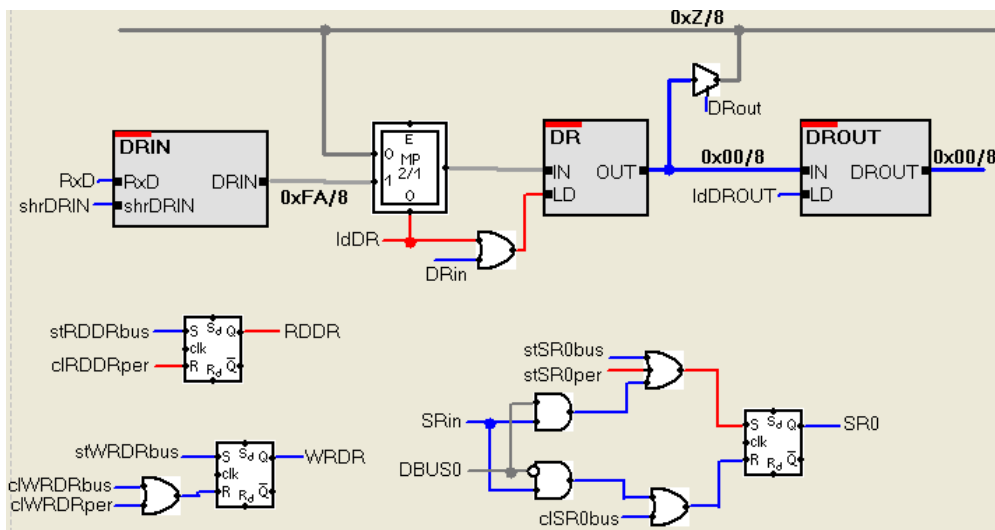


Slika 78 T=248 generisanje signala *rdPER*



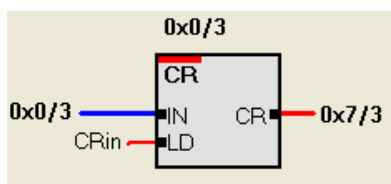
Slika 79 T=301 upisana vrednost u DR i postavljen statusni bit

U trenutku T=299 završeno je čitanje iz periferije i aktivan je signal *fcPERrd*. Kako su aktivni i signali *T1* i *CR2* aktivan je signal *incCNT* kojim se inkrementira brojač *CNT* upravljačke jedinice periferije *uprav_per* i prelazi u stanje T2 u trenutkuT=300 (slika 80). Kako su u ovom trenutku aktivni signali *T2*, *RDDR* i *CR2* postaju aktivni i signali *ldDR*, *clRDDRper* čime se sadržaj prihvatnog registra *DRIN* upisuje u registar podatka *DR* periferije. Takođe u istom trenutku se generiše i signal *intr* što je signal da je registar *DR* raspoloživ da se iz njega prenese podatak u memoriju. Međutim na prekidnu rutinu se skače tek pošto se pročita tekuća instrukcija. U trenutku T=301(slika 79) u registru *DR* se nalazi vrednost *FA* koja predstavlja podatak pročitan iz periferije. Time je čitanje iz periferije završeno.

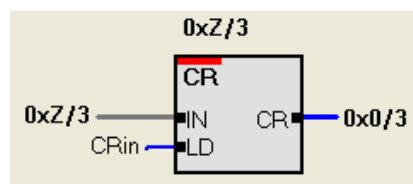


Slika 80 T=300 upisana vrednost u *DRIN*

U trenutku T=332 završeno je izvršavanje instrukcije *STB F100h* kojom se sadržaj akumulatora upisuje u kontrolni registar kontrolera periferije. Kako se u akumulatoru nalazi vrednost 0h kontroler se zaustavlja. U trenutku T=333 u kontrolnom registru *CR* nalazi se vrednost 0h.

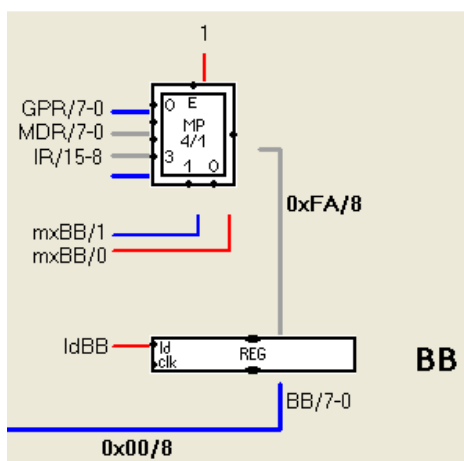


Slika 81 T=332 upis vrednosti 0 u *CR*

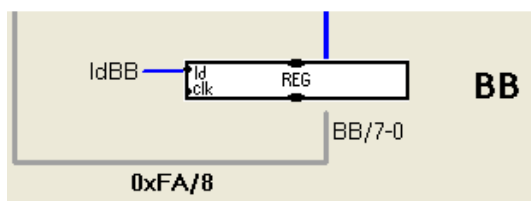


Slika 82 T=333 upisana vrednost 0 u *CR*

Sledeća instrukcija je skok na prekidnu rutinu na adresi *CB00h* u okviru koje se prvo na stek stavljaju *PSW* i *PC* a onda se izvršava prva instrukcija prekidne rutine *LDB F102* kojom se u akumulator upisuje vrednost sa lokacije *F102* koja predstavlja adresu registra *DR* kontrolera periferije sa serijskim prenosom. U trenutku T=334 (slika 83) aktivnim vrednostima signala *mxBB/0* i *ldBB* u akumulator se upisuje vrednost *FAh* pročitana iz registra *DR* kontrolera periferije. U trenutku T=335 (slika 84) u akumulatoru se nalazi vrednost *FAh*.

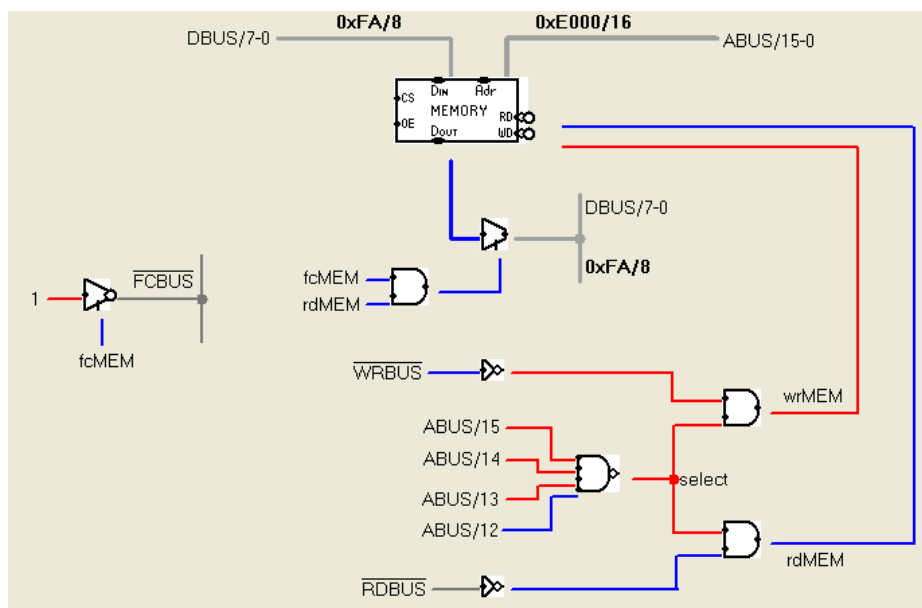


Slika 83 T=334 upis *FAh* u akumulator



Slika 84 T=335 upisana vrednost *FAh* u akumulatoru

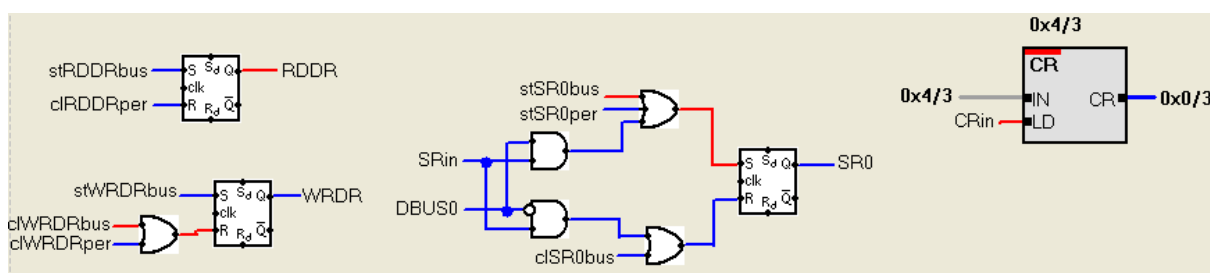
Nakon ove instrukcije izvršava se instrukcija *STB E000* kojom se sadržaj akumulatora upisuje na memorijsku lokaciju *E000*. U trenutku $T=484$ neaktivnom vrednošću signala *WRBUS* vrši se upis vrednosti *FAh*, iz akumulatora, na adresu *E000* u memoriju.



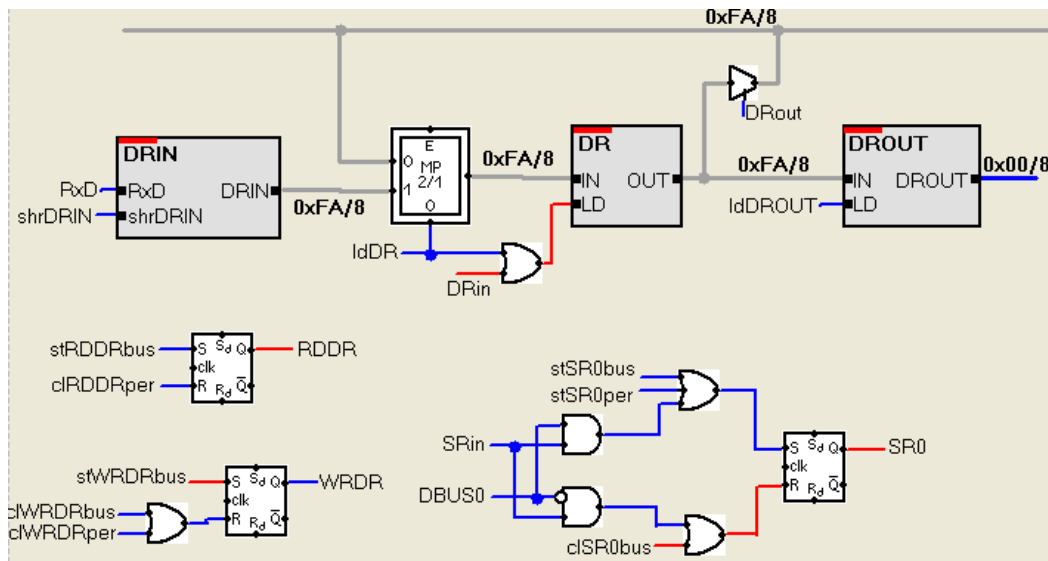
Slika 85 $T=484$ upis vrednosti *FAh* u memoriju na adresu *E000h*

Sljedeća instrukcija je *RTI* kojom se realizuje povratak iz prekidne rutine nakon čega se izvršava instrukcija *LDB 04h* kojom se u akumulator upisuje vrednost *04h*. Ova vrednost se koristi za programiranje kontrolera za rad sa izlaznom periferijom u režimu bez generisanja prekida, tj. sa proverom bita spremnosti. U trenutku $T=570$ aktivnim vrednostima signala *mxBB/1* i *ldBB* upisuje se vrednost *04h* u akumulator *BB*. U trenutku $T=571$ u akumulatoru se nalazi vrednost *04h*.

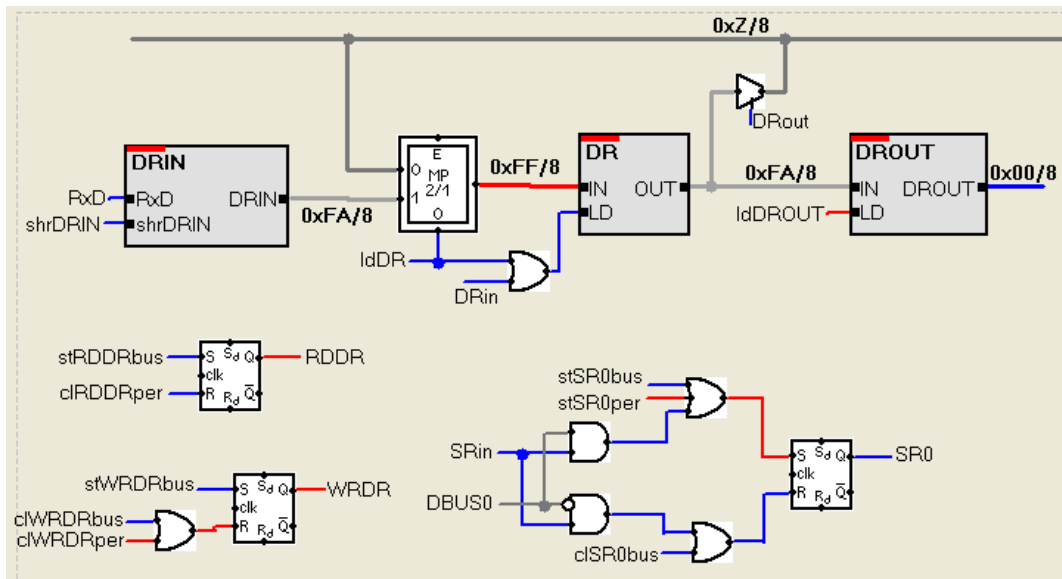
Sljedeća instrukcija *STB F100* vrši upis sadržaja akumulatora u memorijsku lokaciju *F1 00* koja predstavlja adresu kontrolnog registra kontrolera periferije za serijski prenos. U trenutku $T=620$ (slika 86) na sistemskoj magistrali se nalazi vrednost *F100* kojim se selektuje kontrolni registar kontrolera periferije i signal *selCR* postaje aktivan, dok neaktivnom vrednošću na liniji *WRBUS* postaje aktivan signal *wrKTR* (u bloku *uprav_bus*). Ovim aktivnim signalima postaje aktivan signal *CRin* kojim se vrši upis vrednosti *04h* sa magistrale podataka *DBUS* u kontrolni registar *CR* čime je kontroler programiran za upis u periferije bez generisanja prekida, odnosno za rad sa izlaznom periferijom ispitivanjem bita spremnosti. Takođe aktivni su i signali *clWRDRbus*, *stSR0bus* i *fcKTR*. Aktivnom vrednošću signala *clWRDRbus* postaje neaktivan signal *WRDR*, aktivnom vrednošću signala *stSR0bus* postavlja se na aktivnu vrednost bit *SR0* statusnog registra kontrolera periferije koji signalizira da je registar *DR* raspoloživ da se u njega prenese podatak iz memorije (slika 87).



Instrukcija *STB F102* vrši upis sadržaja akumulatora u memorijsku lokaciju *F102* koja predstavlja registar *DR* kontrolera periferije sa serijskim prenosom. U trenutku T=722 aktivni su signali *stWRDRbus*, *DRin* i *clSR0bus*. Signalom *stWRDRbus* postaje aktivan signal *WRDR*. Signalom *DRin* u registar *DR* upisuje se vrednost *FAh* iz akumulatora. Signalom *clSR0bus* bit *SR0* postavlja se na neaktivnu vrednost čime se signalizira da registar *DR* nije raspoloživ. Osim ovih signala u trenutku T=723 aktivan je i signal *ldDROUT* kojim se u registar *DROUT* upisuje sadržaj registra *DR* koji treba upisati u izlaznu periferiju.

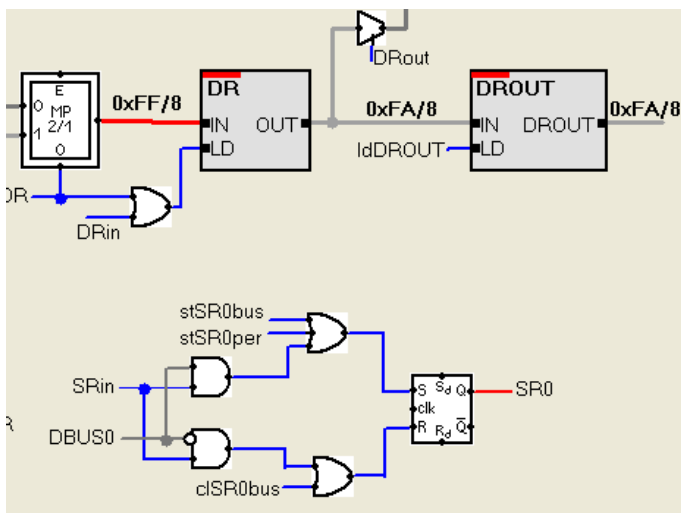


72



Slika 89 T=723 upisana vrednost *FAh* u registar *DR* i postavljen signal *WRDR*

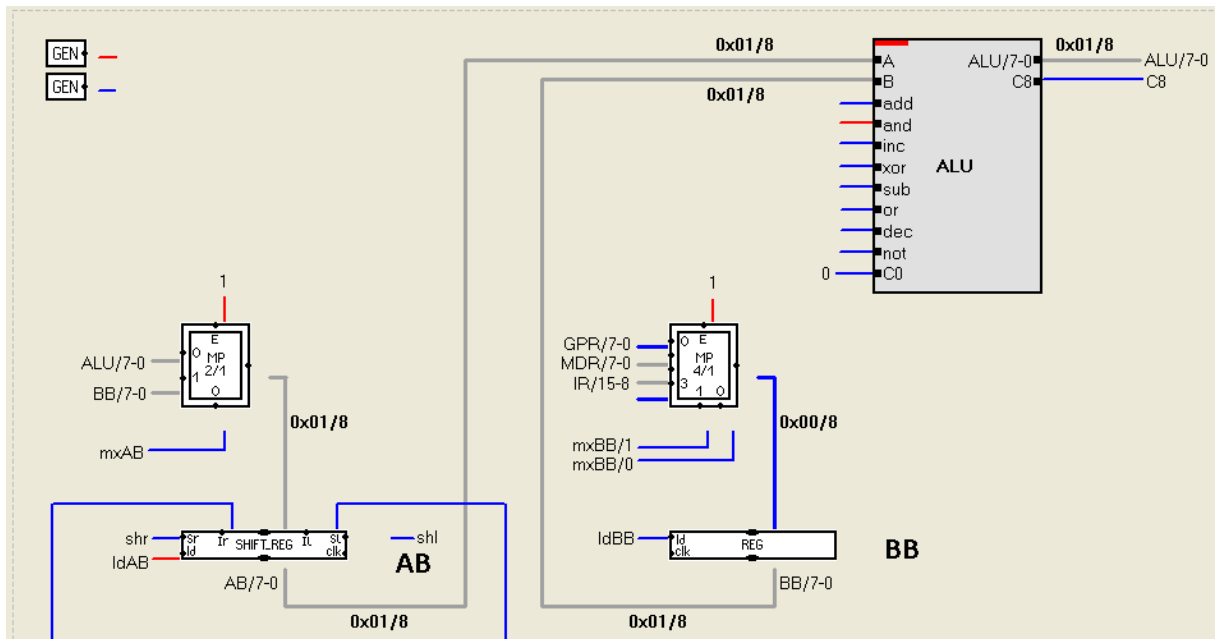
U trenutku T=724(slika 90) u registar *DROUT* upisana je vrednost *FAh* iz registra *DR*. Aktivan je signal *wrPER* kojim počinje upis u periferiju podataka iz registra *DROUT*. Aktivna vrednost bita *SR0* statusnog registra signalizira da je registar *DR* raspoloživ.



Slika 90 T=724 upisana vrednost *FAh* u *DROUT* i postavljen bit *SR0*

Instrukcija *LDB F101* u akumulator upisuje sadržaj memorijske lokacije *F101h* koja predstavlja statusni registar kontrolera periferije *SR*. U trenutku T=770 aktivnim vrednostima signala *mxBB/0* i *ldBB* u akumulator se upisuje vrednost *01h* koja predstavlja sadržaj statusnog registra. U trenutku T=771 u akumulatoru se nalazi vrednost *01h* upisana u prethodnom taktu.

Instrukcija *AND 1* vrši logičku operaciju I nad sadržajem akumulatora i rezultat smešta u akumulator. Ovom instrukcijom se proverava vrednost bita *spremnosti* statusnog registra. U trenutku T=810 (slika 91) aktivni su signali *and* i *ldAB* kojima se u aritmetičkoj jedinici obavlja logička operacija i vrši upis rezultata u akumulator. U trenutku T=811 u akumulatoru se nalazi vrednost *01h* kao rezultat izvršene operacije.



Slika 91 T=810 izvršavanje logičke operacije I

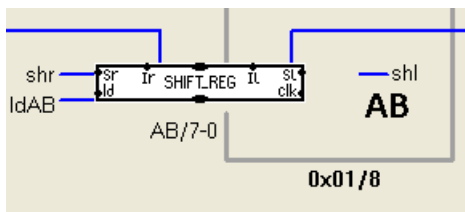
Instrukcijom *BEQL F7* se u slučaju da je ispunjen uslov skoka ($Z=1$) skače na adresu *C02A* na kojoj je instrukcija *LDB F101* kojom se proverava bit spremnost statusnog registra *SR*. Kako uslov skoka nije ispunjen ($Z=0$) nastavlja se sa izvršavanjem sledeće instrukcije.

Instrukcijom *LDB 00h* se u akumulator upisuje vrednost *00h*. U trenutku $T=870$ aktivnim vrednostima signala *mxBB/0* i *ldBB* u akumulator se upisuje vrednost *00h*. U trenutku $T=871$ u akumulatoru se nalazi vrednost upisana u prethodnom taktu.

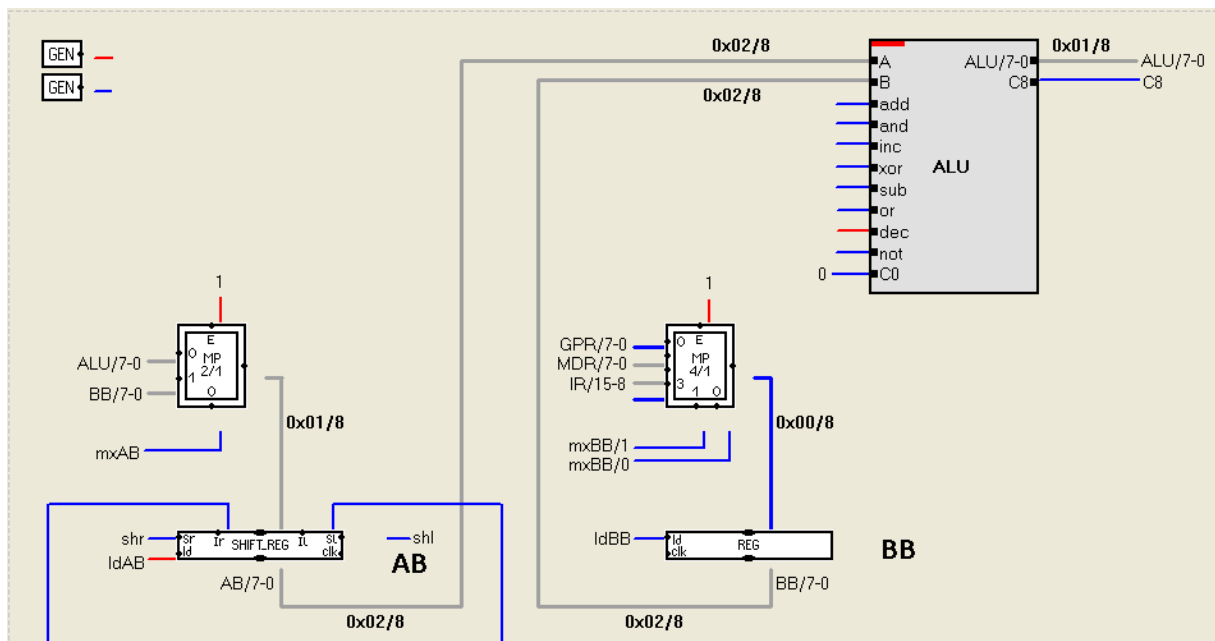
Instrukcijom *STB F100* se sadržaj akumulatora upisuje u memorijsku lokaciju *F100* koja predstavlja kontrolno registar kontrolera periferije *CR*. Kako se u akumulatoru nalazi vrednost *0h* kontroler se zaustavlja. U trenutku $T=921$ u kontrolnom registru *CR* nalazi se vrednost *0h*.

Instrukcijom *LDB D000* se u akumulator upisuje sadržaj memorijske lokacije *D000h* u kojoj se nalazi broj podataka za prenos. U trenutku $T=972$ aktivnim vrednostima signala *mxBB/0* i *ldBB* u akumulator se upisuje vrednost *02h*. U trenutku $T=973$ u akumulatoru se nalazi vrednost upisana u prethodnom taktu.

Instrukcija *DEC* vrši operaciju dekrementiranja nad sadržajem akumulatora i rezultat smešta u akumulator. Ovom instrukcijom se po obavljenom prenosu jednog podatka smanjuje broj preostalih podataka za prenos. U trenutku $T=989$ (slika 93) aktivni su signali *dec* i *ldAB* kojima se u aritmetičkoj jedinici obavlja logička operacija i vrši upis rezultata u akumulator. U trenutku $T=990$ (slika 92) u akumulatoru se nalazi vrednost *01h* kao rezultat izvršene operacije.



Slika 92 T=990 upisan rezultat operacije dekrementiranja u akumulator

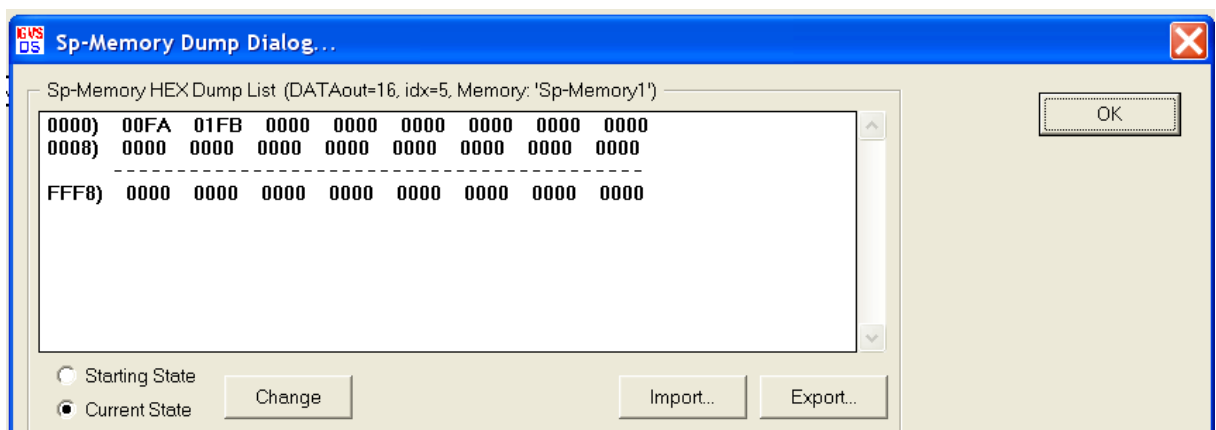


Slika 93 T=989 izvršavanje operacije dekrementiranja

Instrukcijom *STB D000* se vrši upis sadržaja akumulatora u memorijsku lokaciju *D000* u trenutku T=1037. U trenutku T=1038 u memorijsku lokaciju *D000* upisana je vrednost *01h* iz akumulatora.

Instrukcijom *BNEQL C7* se u slučaju da je ispunjen uslov skoka ($Z=0$) skače na adresu *C00C* na kojoj je instrukcija *INTE* kojom ponovo počinje ciklus prenosa sledećeg posatka iz periferije u memoriju i kasnije iz memorije u periferiju na već opisani način. Kako je uslov skoka ispunjen ($Z=0$) skače se na instrukciju *INTE*, omogućavaju se prekidi, čita se podatak iz periferije, upisuje se podatak u periferiju, ponovo se dekrementira broj preostalih podataka za prenos i kada dostigne nulu, u trenutku $T=1910$, dolazi se do instrukcije *HALT* kojom se završava izvršava programa i zaustavlja procesor.

Kao rezultat programa u izlaznoj periferiji na lokacijama *00h* i *01h* nalaze se vrednosti *FAh* i *1FBh* pročitane iz ulazne periferije sa doatkom bita parnosti.

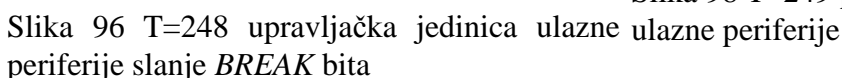


Slika 94 sadržaj memorije izlazne periferije po završenom programu

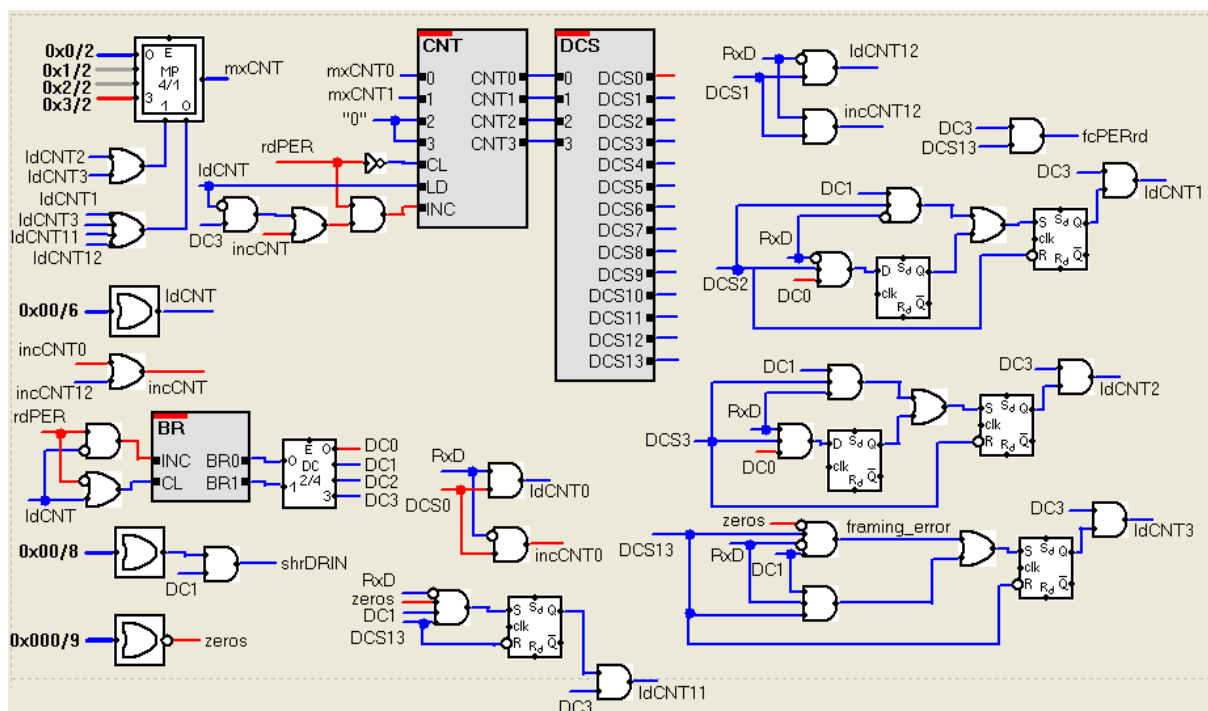
6.3 Simulacija na nivou signala

U prethodnom tekstu dat je opis simulacije na nivou instrukcije. U ovom delu navodi se deo simulacije od značaja za serijski prenos na nivou signala i to slanje i prijem bitova po

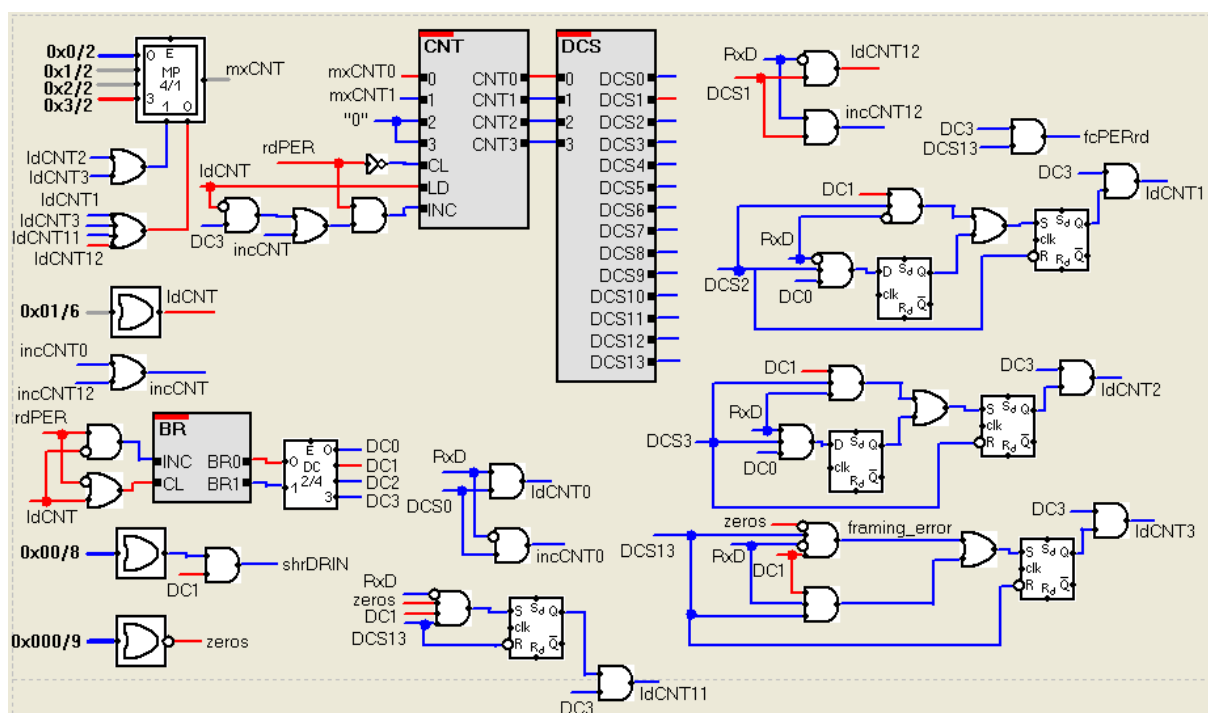
Čitanje se obavlja u skladu sa protokolima opisanim u 3.1.2.1 i 3.2.2.1. To znači da će se na liniji *RxD* pojaviti sledeći bitovi *BREAK*, *VALID1*, *START*, 8 bitova podataka, *PARITY* i *STOP* u skladu sa vrednošću koja se nalazi na liniji *RxD* upravljačka jedinica ulazne periferije kontrolera periferije prolazi kroz odgovarajuća stanja i generiše potrebne signale u skladu sa tim.



Slika 98 T=249 pročitani podatak iz memorije

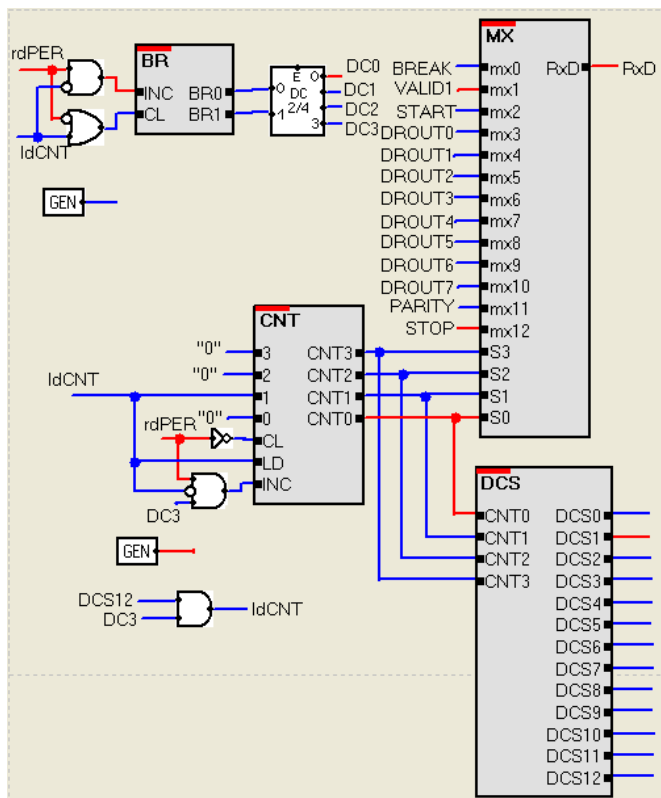


Slika 99 T=248 upravljačka jedinica ulazne periferije na strani kontrolera u stanju *POČETNO STANJE*

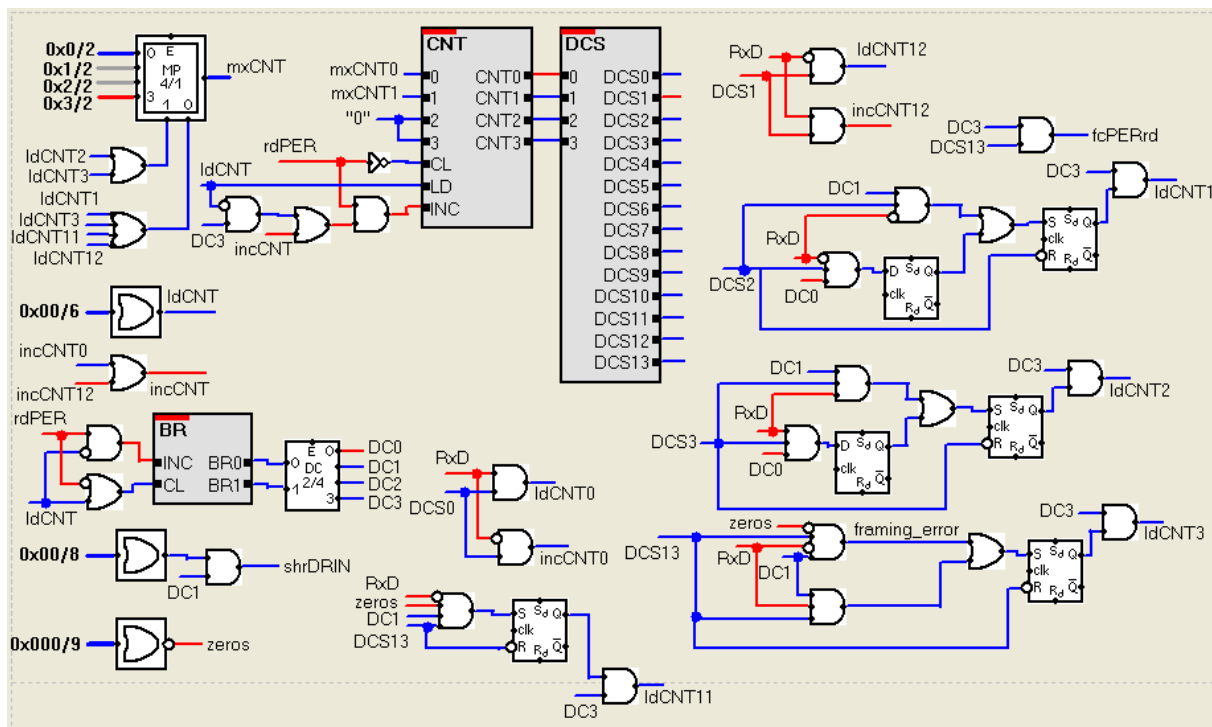


Slika 100 T=249 upravljačka jedinica ulazne periferije na strani kontrolera u stanju *BREAK*

U trenutku T=252 šalje se bit *VALID1* i u upravljačkoj jedinici ulazne periferije na strani kontrolera generiše se aktivna vrednost *incCNT12* čime signal *incCNT* postaje aktivan i upravljačka jedinica prelazi u stanje *VALID1*.



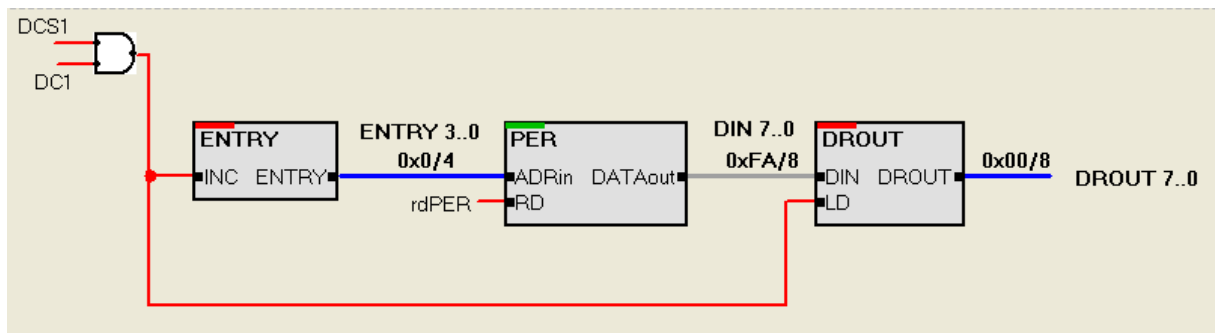
Slika 101 T=252 slanje bita *VALID1* na strani periferije



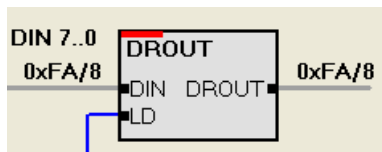
Slika 102 T=252 prelazak upravljačke jedinice kontrolera periferije u stanje *VALID1*

U trenutku T=253 generiše se aktivna vrednost signala *DC1* i *DCS1* i vrši se upis vrednosti *FAh* u registar *DROUT* ulazne periferije (slika 103).

U trenutku T=254 u regostru *DROUT* nalazi se vrednost *FAh* upisana u prethodnom taktu (slika 104). Takođe na linijama *DROUT*_{7...0} nalaze se bitovi podatka koji se nalazi u registru *DROUT*.

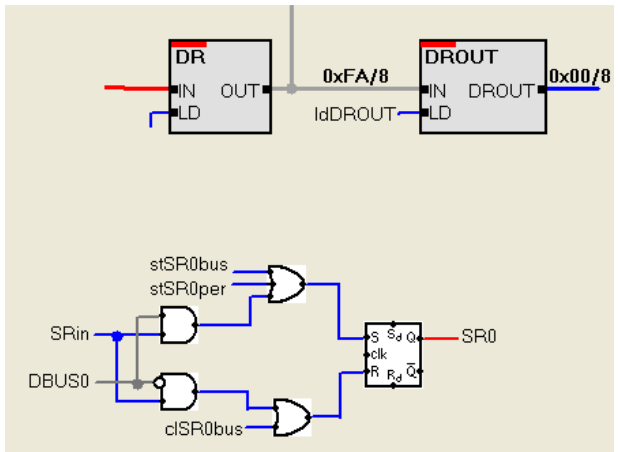


Slika 103 T=253 upis vrednosti *FAh* pročitane iz memorije ulazne periferije u prihvatni registar podatka *DROUT*

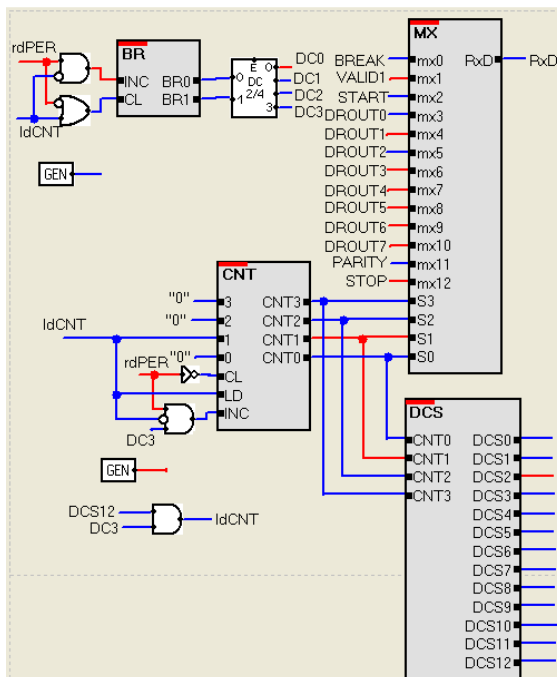


Slika 104 T=254 upisana vrednost u
DROUT

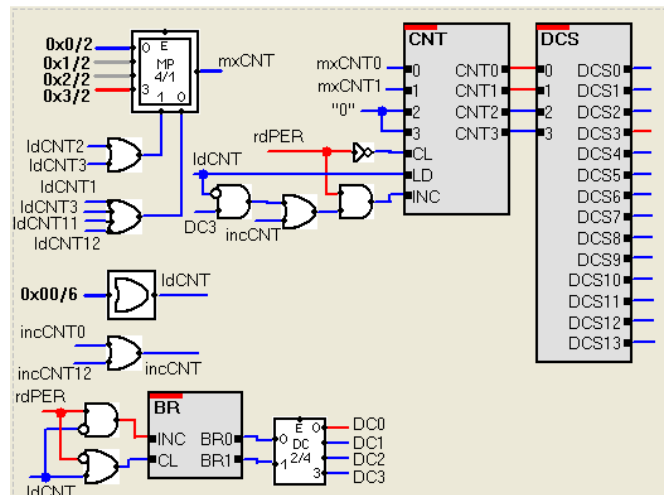
U trenutku T=256 (slika 106) na linijama *DROUT0-DROUT7* nalaze se bitovi podataka koji se šalje (*FAh*) takođe šalje se *START* bit i nadalje bitovi podataka, bit parnosti i bit *STOP*.



Slika 105 T=301 upisana vrednost u *DR* i postavljen statusni bit

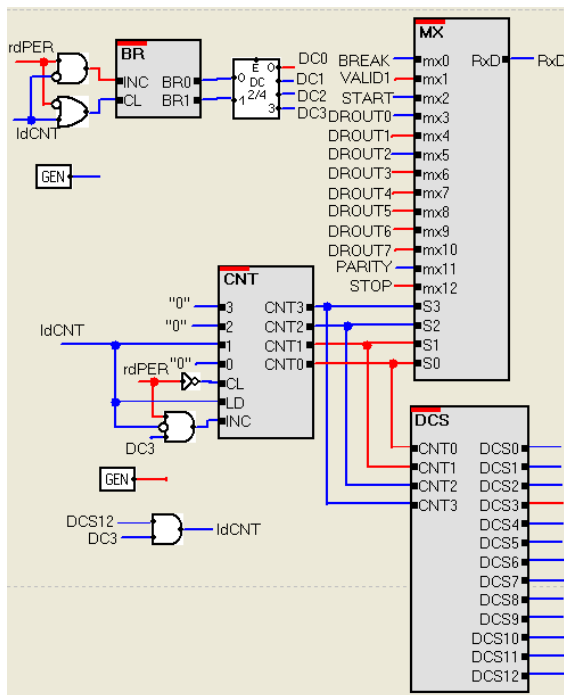


Slika 106 T=256 slanje *START* bita na strani periferije

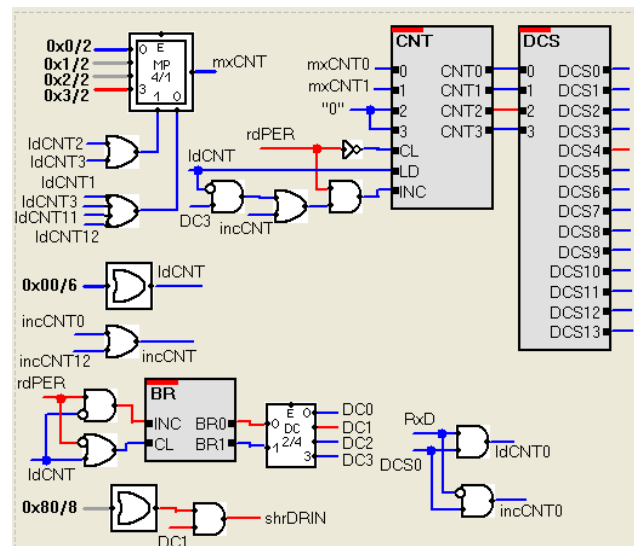


Slika 107 T=256 prijem *START* bita na strani kontrolera

U trenutku $T=260$ (slika 107) šalje se prvi bit podatka sa vrednošću nula. U trenutku $T=261$ (slika 108) vrši se prijem prvog bita podatka u kontroleru i generiše se signal *shrDRIN* kojim se vrši upis i pomeranje bita podatka u registru *DRIN*.

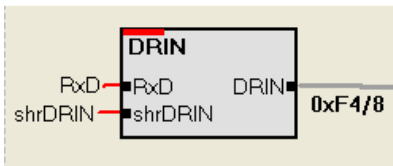


Slika 107 T=260 slanje prvog bita podatka na strani periferije

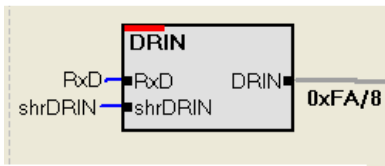


Slika 108 T=261 prijem prvog bita podatka na strani kontrolera

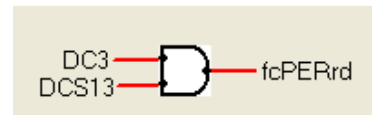
Na dalje se vrši prijem ostalih bitova podatka, bita parnosti i *STOP* bita na isti način. U trenutku T=289 (slika 109) signalom *shrDRIN* vrši se upis poslednjeg bita podatka u registar *DRIN*. U trenutku T=290 (slika 110) u registru *DRIN* nalazi se vrednost *FAh* upisana iz periferije.



Slika 109 T=289 upis poslednjeg bita u *DRIN*

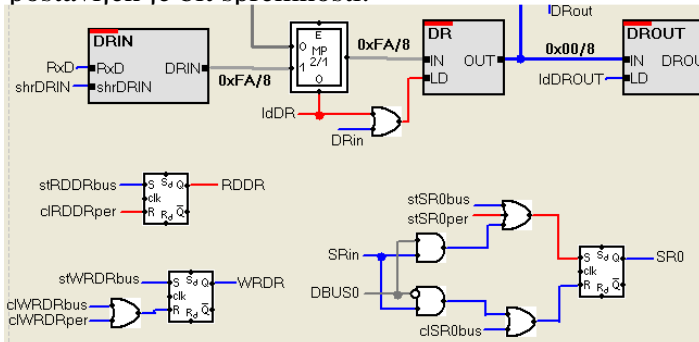


Slika 110 T=290 upisana vrednost *FAh* u *DRIN*

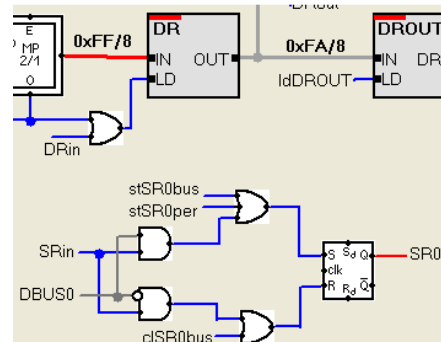


Slika 111 T=299 aktivna vrednost signala *fcPERrd*

U trenutku T=299 (slika 111) završeno je čitanje iz periferije i aktivan je signal *fcPERrd*. U trenutku T=300 (slika 112) aktivnom vrednošću signala *ldDR* vrši se upis vrednosti *FAh* u registar *DR* iz registra *DRIN*, takođe aktivnom vrednošću signala *stSR0per* postavlja se bit spremnosti statusnog registra i time signalizira da je registar *DR* raspoloživ da se iz njega prenese podatak u memoriju. Aktivnom vrednošću signala *clRDDRper* signal *RDDR* se postavlja na neaktivnu vrednost. U trenutku T=301 (slika 113) upisana je vrednost *FAh* i postavljen je bit spremnosti.



Slika 112 T=300 upis vrednosti *FAh* u registar *DR* i postavljanje bita spremnosti

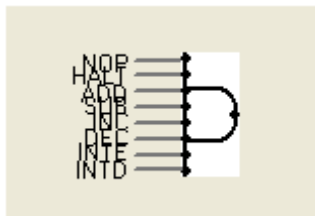


Slika 113 T=301 upisana vrednost u *DR* i postavljen bit spremnosti

7 ZAKLJUČAK

Cilj ovog rada je bio taj da se isprojektuje kontroler za serijsku prenos podataka i realizuje simulator u softverskom paketu IGoVSoDS koji će se koristiti u laboratoriji za učenje. Na samom početku je bio dat opis arhitekture i organizacije sistema nad kojim se simulator konstruiše. Opisana je, dakle, struktura samog sistema, iz kojih se delova sastoji kao i formati instrukcija i načini adresiranja koji se na njemu mogu izvršavati. Nakon ovoga, dat je detaljan uvid u to šta nudi softverski paket IGoVSoDS. Dat je prikaz svih karakterističnih funkcija koje ovaj program pruža. U nastavku, prikazan je sam simulator koji je konstruisan. Detaljno su opisani svi delovi sa svim relevantnim signalima. Sav opis je ilustrovan sa slikama realizovanog simulatora. Na samom kraju, dat je test primer koji ilustruje rad ovog simulaora. Autor veruje da su svi najvažniji elementi ovog sistema i simulatora testirani ovim primerima.

Osim ovog već pomenutog cilja koji je ovaj rad imao, simulator koji je ovde konstruisan takođe služi kao jedan pokazatelj efikasnosti softverskog paketa IGoVSoDS. Autor smatra da ovaj program pruža odličan interfejs za konstruisanje raznih simulatora i daje najbolju preporuku za njegovo dalje korišćenje. Međutim, autor je takođe primetio par nedostataka u ovom programu koji su otežavali konstrukciju simulatora. Jedan manji problem se ogleda u tome da su pojedina kola koja su dostupna u biblioteci dostupnih kola u programu veoma mala s obzirom na njihovu veličinu. Recimo, 8-mo ulazno I kolo je praktično neupotrebljivo jer se na 8 ulaza dovode signali koji su previše zbijeni pa je njihovo označavanje skoro nemoguće izvesti bez problema. Takođe i pri obeležavanju signala na 4-ro ulaznim kolima se javlja sličan problem nepreglednosti mada malo manje izražen.



Drugi nedostatak je taj da kada se spajaju magistrale između modula moraju da se spoje izlomljenim linijama kako bi se otvorio prozor sa menije za odabir signala koji se spajaju, što zna da bude frustrirajuće u prvi mah dok se ne otkrije način za spajanje magistrala.

LITERATURA

1. J. Djordjevic, A. Milenkovic, N. Grbanovic, *An Integrated Educational Environment for Teaching Computer Architecture and Organization*, IEEE MICRO, Vol. 20, No. 3, pp. 66-74, May/June **2000**.
2. J. Djordjevic, B. Nikolic, A. Milenkovic, *Flexible Web-based Educational System for Teaching Computer Architecture and Organization*, IEEE Transactions on Education, Vol. 48, No. 2, pp. 264-273, May **2005**.
3. J. Djordjevic, M. Mitrovic, B. Nikolic, *A Memory System for Education*, The Computer Journal, Vol. 48, No. 6, pp. 630-641, November **2005**.
4. A. Stojkovic, J. Djordjevic, B. Nikolic, *WASP – A Web-Based Simulator for an Educational Pipelined Processor*, International Journal on Electrical Engineering Education, Vol. 44, No. 3, **2007**, pp. 197-215.
5. J. Djordjevic, B. Nikolic, T. Borozan, A. Milenković, *CAL²: Computer Aided Learning in Computer Architecture Laboratory*, Computer Applications in Engineering Education, Vol. 16, No. 3, **2008**, pp. 172-188.
6. Boško Nikolić, Zaharije Radivojević, Jovan Djordjević, Veljko Milutinović, *A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization*, IEEE Transactions on Education, rad prihvacen
7. J. Đorđević, *Arhitektura i organizacija računara, Računarski sistem za rad u laboratoriji, Arhitektura i organizacija računarskog sistema*, Interni materijal u pripremi za štampu, Elektrotehnički fakultet Beograd, **2007**
8. J. Đorđević, *Arhitektura računara, edukacioni računarski sistem, Arhitektura i organizacija računarskog sistema*, Akademska misao, Beograd, **2003**
9. N. Grbanović, *Interaktivni generator vizuelnih simulatora digitalnih sistema*, Doktorska disertacija u izradi, Interni materijal u pripremi za štampu, **2007**
10. N. Grbanović, *Interaktivni Generator Vizuelnih Simulatora Digitalnih Struktura (IGoVSoDS) priručnik za upotrebu*, Elektrotehnički fakultet Beograd, **2007**
11. Đorđević, J., Grbanović, N., Nikolić, B., *Arhitektura računara, Edukacioni računarski sistem, Priručnik za simulaciju sa zadacima*, Elektrotehnički fakultet, Beograd, **2002**.
12. Lazić, B., *Logičko projektovanje računara*, Nauka—Elektrotehnički fakultet, Beograd, Srbija, Jugoslavija, **1994**