FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA ZAGREB
Zavod za automatiku i procesno računarstvo

Dr.sc. Ivan Petrović

Upute za korištenje programirljivog logičkog kontrolera SIMATIC S7-314IFM

Zagreb, 1999.

1. UVOD

Programirljivi logički kontroleri najčešće su primjenjivani uređani za automatizaciju industrijskih postrojenja i procesa. Njihove su osnovne značajke sklopovska modularnost i programska fleksibilnost, koje ih čine primjenivim za rješavanje raznorodnih automatizacijskih zadaća.

Smatrajući važnim da studenti Automatike ovladaju korištenjem programirljivih logičkih kontrolera, na nekoliko je laboratorijkih vježbi iz predmeta "Automatizacija postrojenja i procesa" predviđena njihova primjena.

"Upute za korištenje programirljivog logičkog kontrolera SIMATIC S7-314IFM" ukratko opisuju osnovne značajke toga PLC-a te daju osnove njegova programiranja pomoću programskog paketa STEP 7. Za detaljnije podatke studentima će biti na raspolaganju izvorne upute proizvođača.

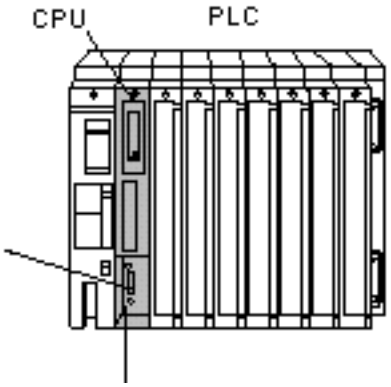
Ovo je prva inačica uputa pa se autor unaprijed ispričava studentima za moguće nedostatke i cijenit će svaku sugestiju za njihovo otklanjanje.

2. PROGRAMIRLJIVI LOGIČKI KONTROLER (PLC)

2.1 Osnovne značajke

Povećanjem automatizacije procesa pri upravljanju, identifikaciji i analizi, zatim alarmiranju, nadzoru, signalizaciji i dojavama, javlja se potreba za brzim, efikasnim i snažnim elektroničkim uređajima. Upravo takvi uređaji su programirljivi logički kontroleri (engl. **P**rogrammable **L**ogic **C**ontrollers-**PLC**). Osnovne značajke PLC-ova su pouzdanost, male dimenzije (za razliku od ormara s relejima i sklopnicima), te najvažnije programirljivost i fleksibilnost.

PLC S7 300 SA PROCESOROM CPU 314 IFM



Sl. 2.1 Izgled PLC-a.

Osnovne značajke PLC-a iz porodice SIMATIC S7-300 sa središnjim procesorom CPU 314 IFM su:

Izvor napajanja PS 307 2A pretvara 230V (120V) AC u istosmjerni napon od 24V maksimalne struje 2A.

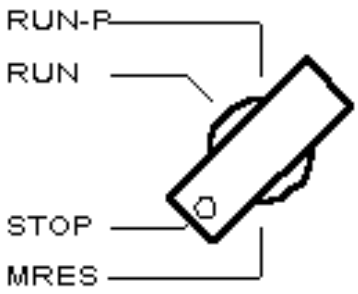
Središnji procesorski modul.

Ovaj modul sadržava ugrađen procesor, spremnik baterije, priključak napajanja, MPI port, diode za prikaz stanja procesora i preklopku za izbor načina rada. Sam procesor ima naravno i memoriju iz koje čita i u koju sprema podatke.

Preklopka za izbor načina rada

Ova preklopka omogućava četiri položaja koji postavljaju procesor u željeno stanje (slika 2.2.).

- **STOP** - procesor miruje to jest ne izvršava program. Ovaj način rada služi za prijenos podataka između CPU-a i računala i obratno.
- **RUN** - procesor izvršava program. Ovdje je moguće prenositi podatke samo iz PLC-a na računalo.
- **RUN-P** - u ovom modu procesor izvršava program i moguća je dvosmjerna komunikacija. I čitanje i pisanje u PLC.
- **MRES** - brisanje memorije PLC-a.



Sl. 2.2. Preklopka za izbor načina rada.

2.2 Memorijska područja CPU-a

Memorijska područja se sastoje od: **Load** memorije koja sadrži korisnički program. **Work** (radne) memorije koja sadrži dijelove S7 programa koji su trenutno potrebni za izvršavanje. Kao što su na primjer logički i podatkovni blokovi. **System** (sistemska) memorija sadrži elemente bitne za izvođenje programa. Kao što su ulazno izlazna tablica podataka, bit memorija, brojači vremena (engl. timer) i brojači. Također sadrži blok 'stog' i prekidni (engl. interrupt) stog. Radna i sistemska memorija smještena je u RAM memoriji i to je mjesto gdje se program izvodi, dok 'load' memorija može biti smještena još i u FEPROM-u ili na posebno dodanoj memorijskoj kartici.

Memorija S7 CPU-a podijeljena je na adresna područja (vidi tablicu 2.1.). Upotrebom instrukcija korisničkog programa izravno se obraća pojedinom dijelu adresnog područja predviđenog za te podatke.

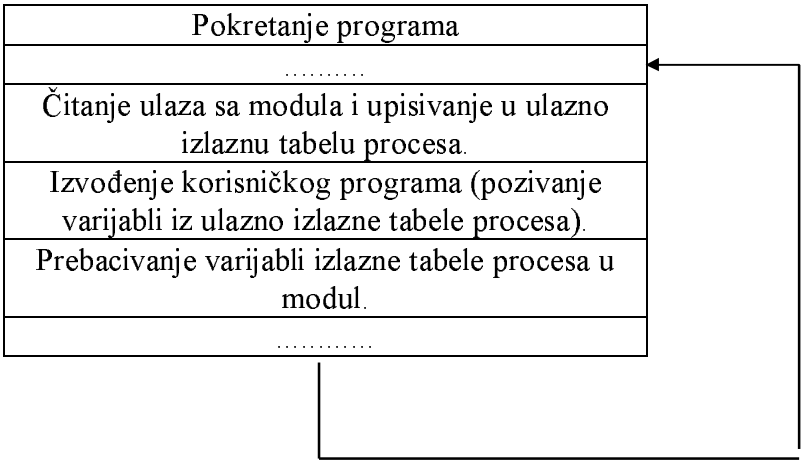
Memorijsko područje	**Tip (veličina) podatka**		**Identifikator**
Slika stanja ulaza (Proccess-image input)	Ulazni bit	(Input BIT)	I
	Ulazni bajt	(Input BYTE)	IB
	Ulazna riječ	(Input WORD)	IW
	Ulazna dvostruka riječ	(Input DWORD)	ID
Slika stanja izlaza (Proccess-image output)	Izlazni bit	(Output BIT)	Q
	Izlazni bajt	(Output BYTE)	QB
	Izlazna riječ	(Output WORD)	QW
	Izlazna dvostruka riječ	(Output DWORD)	QD
Memorija u užem smislu	Memorijski bit	(Memory BIT)	M
	Memorijski bajt	(Memory BYTE)	MB
	Memorijska riječ	(Memory WORD)	MW
	Memorijska dvostruka riječ	(Memory DWORD)	MD
Periferni ulazi	Periferni ulazni bajt	(Peripherial Input BYTE)	PIB
	Periferna ulazna riječ	(Peripherial Input WORD)	PIW
	Periferna ul. dvostruka riječ	(Peripherial Input DWORD)	PID

Periferni izlazi	Periferni izlazni bajt Periferna izlazna riječ Periferna izl. dvostruka riječ	(Peripherial Output BYTE) (Peripherial Output WORD) (Peripherial Output DWORD)	PQB PQW PQD
Zajednički podatkovni blokovi DBX (Shared)	Bit Bajt Riječ Dvostruka riječ	(Data BIT) (Data BYTE) (Data WORD) (Data DWORD)	DBX DBB DBW DBD
Posebni podatkovni blokovi DIX (Instance)	Bit Bajt Riječ Dvostruka riječ	(Data BIT) (Data BYTE) (Data WORD) (Data DWORD)	DIX DIB DIW DID
Privremeni lokalni stog	Lokalni bit Lokalni bajt Lokalna riječ Lokalna dvostruka riječ	(Local BIT) (Local BYTE) (Local WORD) (Local DWORD)	L LB LW LD

Tablica 2.1. Adresna područja.

2.3 Slika stanja ulaza i izlaza

Izlazi (Q) i ulazi (I) potrebni za izvršavanje programa to jest njihove vrijednosti smješteni su u memoriji. Ulazi se očitavaju sa modula na početku svakog ciklusa programa, a izlazi se postavljaju na kraju ciklusa. Prednosti ovakvog načina rada sa ulazima i izlazima je u tome što za vrijeme jednog ciklusa imamo konstantne podatke koji se ne mijenjaju tako da ne dođe do različitih interpretacija nad istom varijablom. Također je ovakav način dosta brži u odnosu na pojedino pristupanje modulu ulazno izlaznih podataka pri svakom pozivu na određenu varijablu. Prikaz tijeka pojedinih događaja vidljiv je na slici 2.3.



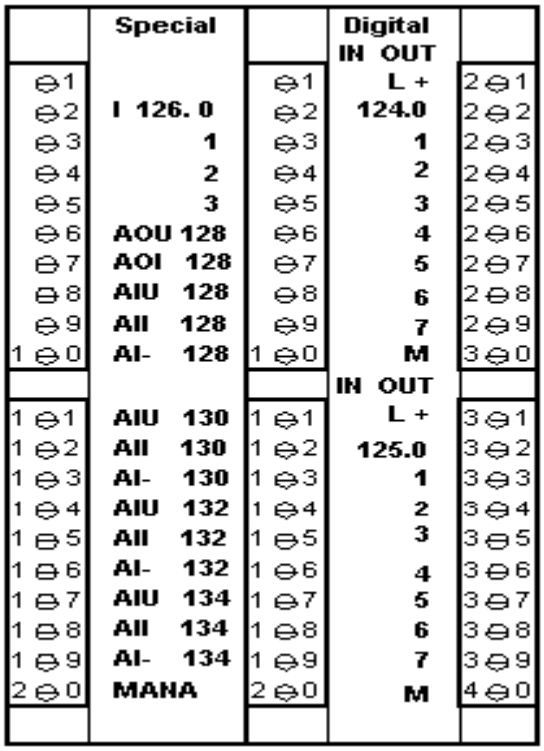
Sl. 2.3. Prikaz tijeka stanja osvježavanja ulazno izlazne tabele procesa.

Jedini izuzetak ovog pravila je analogno-digitalna i digitalno analogna pretvorba. Pretvorba se pokreče pri svakom pozivu ulaza ili pisanja na analogni izlaz.

2.4 Integrirani digitalni i analogni ulazi i izlazi

Modul integriranih digitalnih i analognih ulaza i izlaza prikazan je na slici 2.4. Taj modul se postavlja uz procesor i s njim je direktno povezan, što znači da je njegova direktna veza sa određenim procesom. U slijedećih pet točaka navedeno je s kojim ulazima i izlazima ovaj modul raspolaže:

- 4 posebna (prekidna) digitalna ulaza s adresama od I 126.0 do I 126.3 (koriste ih integrirane funkcije);
- 1 analogni izlaz (naponski - AOU, tj. strujni - AOI) s adresom PQ128;
- 4 analogna ulaza (naponska - AIU, tj. strujna - AII) s adresama PI 128, PI 130, PI 132, PI 134;
- 16 digitalnih ulaza s adresama od I 124.0 do I 124.7. odn. od I 125.0 do I 125.7;
- 16 digitalnih izlaza s adresama od Q 124.0 do Q 124.7, odn. od Q 125.0 do Q 125.7.



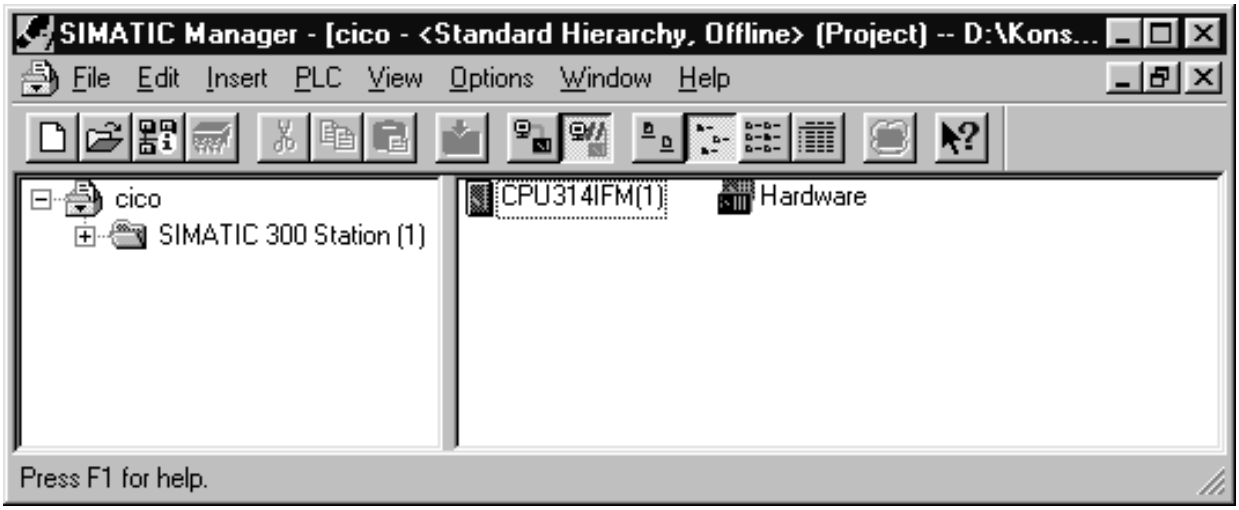
Sl. 2.4. Prikaz integriranih ulaza i izlaza.

3. PROGRAMSKI PAKET 'STEP 7'

Programiranje PLC-a SIMATIC S7 obavlja se pomoću programskog paketa STEP 7. Program radi pod Windows 95 okruženjem, te ga to čini mnogo preglednijim u odnosu na prijašnja programsko sučelja (na primjer 'STEP 5'). Primjer je vidljiv na slici 3.1. Samo programiranje u širem smislu obuhvaća stvaranje projekta i definiranje stanica sa kojima će se raditi (može ih biti više unutar jednog projekta), zatim odabir sklopovlja i programiranje u užem smislu to jest definiranje blokova u koje se upisuje sam kod programa za izvršavanje.

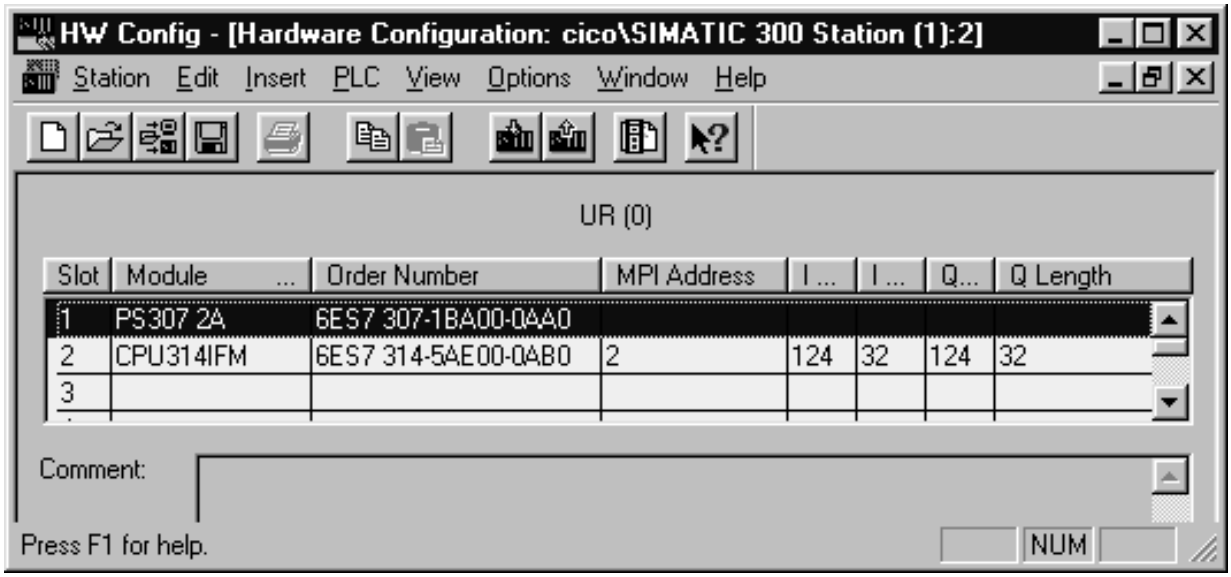
3.1 Definiranje projekta

Otvaranje novog projekta vrši se pomoću **File** → **New** → **Project** (Sve naredne naredbe bit će prikazivane kao prethodna, to jest pomoću padajućih izbornika). Nakon otvaranja projekta i zadavanja njegovog imena, pomoću **Insert** → **Station** → **SIMATIC 300 Station** postavlja se stanicu sa kojom se radi.

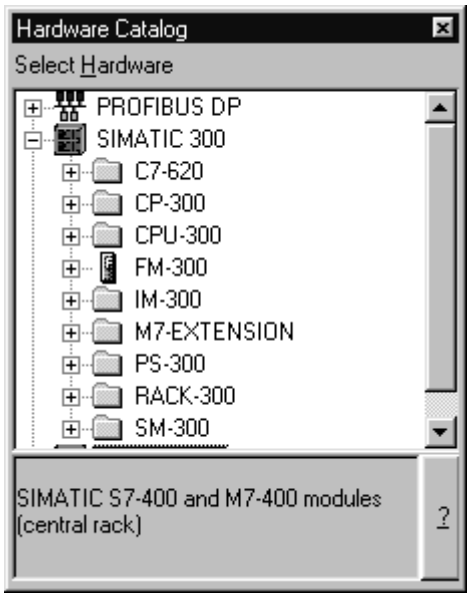


Sl. 3.1. Prikaz osnovnog prozora SIMATIC Managera.

Zatim otvaranjem datoteke **Hardware** (slika 3.2.) te naredbom **Insert** → **Hardware components** prikazuje se lista to jest popis komponenata (katalog) vidi sliku 3.3. iz kojeg se u slot 0 ubacuje **RACK 300** → **Rail**, u slot 1 napajanje **PS 300** → **PS 307 2A**, te u slot 3 procesor **CPU 300** → **CPU 314 IFM**.



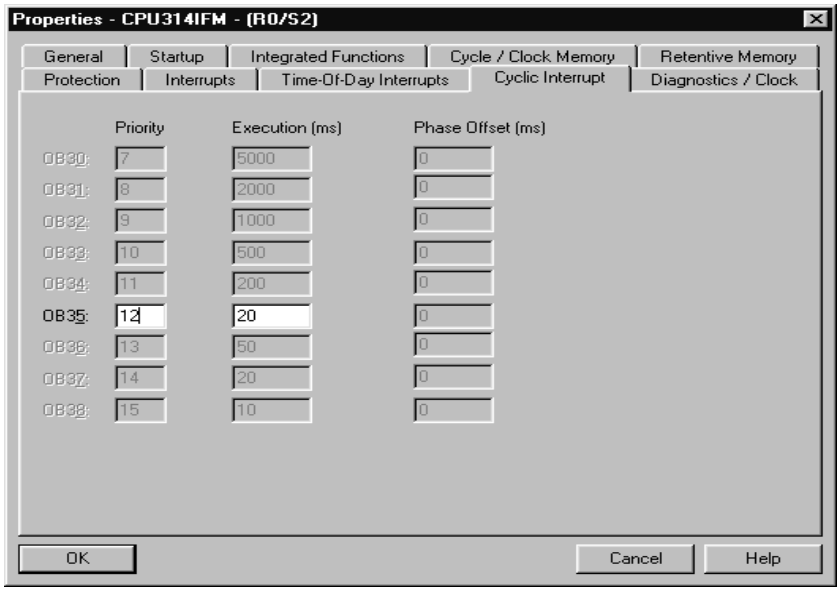
Sl. 3.2. Prozor za definiranje komponenata sklopovlja.



Sl. 3.3. Hardware Catalog prikazuje popis svih mogućih komponenata PLC-a. (Odavde se odabiru komponente da bi definirali PLC.)

Još jedna vrlo bitna stavka je namještanje parametara procesora, a taj se prozor dobiva naredbom **Edit** → **Object Properties** iz prozora sa slike 3.1. Object Properties omogućava namještanje nekih osnovnih značajki bitnih za rad procesora i komunikacije s njim. Jedna od njih je postavljanje vremena ciklusa za organizacijski blok 'OB35' koji se dobiva otvaranjem mape **Cyclic Interrupt** (vidljivo na slici 3.4.) i zadaje se u milisekunama. **Priority** to jest razinu prioriteta za ovaj procesor nije dozvoljeno mijenjati.

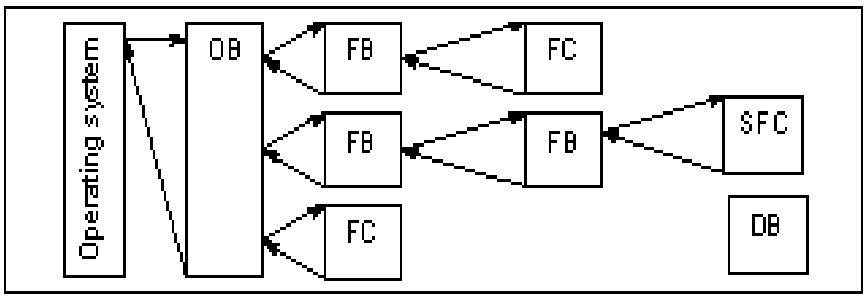
Još jedna interesantna opcija je **MPI** koja se nalazi u mapi **General** a služi za definiranje kod više procesorskih jedinica bit će pojašnjena u dijelu o komunikaciji. Ostale postavke trenutno nisu od velike važnosti i mogu se ostaviti pretpostavljene vrijednosti od strane proizvođača.



Sl. 3.4. Prikaz prozora za namještanje parametara bitnih za izvođenje programa.

3.2 Struktura korisničkog programa

Korisnički program je organiziran kroz programske blokove koji međusobno povezani tvore jednu programsku cjelinu. Osnovna struktura programa je definirana i mora se poštivati prilikom kreiranja programa, slika je:



Sl. 3.5. Struktura programa.

Općenito korisnički program sastoji se od sljedećih blokova :

- organizacijski blokovi (OB)
- funkcijski blokovi (FB)
- funkcije (FC)
- sistemski funkcijski blokovi (SFB)
- sistemske funkcije (SFC)
- podatkovni blokovi(DB)

Nazivi blokova, moguća veličina bloka i broj pojedinih blokova dani su u tablici 3.1.

Tip bloka	Naziv bloka	Veličina bloka	Broj bloka
OB	Organizacijski blok	8192 bajta	-
FB	Funkcijski blok	8192 bajta	0 – 127
FC	Funkcija	8192 bajta	0 - 127
DB	Podatkovni blok	8192 bajta	1 - 127 (0 rezervirano)
SFB	Sistemski funkcijski blok	-	-
SFC	Sistemska funkcija	-	-

Tablica 3.1. Vrste blokova.

3.3 Organizacijski blokovi

Oni ostvaruju vezu između korisničkog programa i operativnog sistema CPU-a, tako da operativni sistem pokreće OB, a on dalje poziva funkcije i funkcijske blokove. OB obavlja izvršavanje korisničkog programa na način koji je odabran, a može biti :

- ciklički po pokretanju
- u određenim vremenskim intervalima
- u zadano vrijeme dana
- nakon zadanog vremena
- nakon pojave greške
- nakon pojave prekida

Prije početka izvršavanja programa operativni sistem pročita sve podatke sa slike stanja ulaza i sa perifernih ulaza, a po završetku izvršavanja programa operativni sistem zapiše sve potrebne podatke u sliku stanja izlaza, odakle se šalju na periferne izlaze.

Cikličko Izvršavanje (OB1)

Operativni sistem izvršava OB1 po pokretanju, te nakon što dođe do kraja počinje ga izvršavati ponovo. Maksimalno predviđeno vrijeme izvršavanja je postavljeno na 150ms, no može se promijeniti na drugu vrijednost. Postoji i minimalno vrijeme za koje operativni sistem zakasni pokretanje programa. Ono se također pojavljuje kao parametar koji se može promijeniti. Ukoliko se premaši maksimalno predviđeno vrijeme izvršavanja operativni sistem će pozvati OB80. Ako OB80 nije programiran CPU će otići u STOP mod.

Izvršavanje na vrijeme dana (OB10 - Ob17)

S pomoću organizacijskih blokova OB10 do OB17 omogućeno je pokretanje programa u točno vrijeme dana određenog datuma, u sljedećim vremenskim periodima izvršavanja :

- jednom
- svake minute
- svakog sata
- dnevno
- tjedno
- mjesečno
- godišnje

Vrijeme i datum pokretanja, period izvršavanja, te aktiviranje OB-a, može se postaviti konfiguriranjem parametara hardware-a unutar projekta. Drugi način je pozivanjem sistemskih funkcija unutar samog programa i to za setiranje - SFC28 (SET_TINT), i aktiviranje - SFC30 (ACT_TINT). Nakon pokretanja OB-a CPU će na osnovu zadanog perioda proračunati vrijeme sljedećeg starta. Eventualno zaustavljanje izvršavanja OB-a moguće je napraviti unutar programa pozivanjem sistemske funkcije - SFC29 (CAN_TINT).

Izvršavanje sa kašnjenjem (OB20 - OB23)

Zakašnjelo izvršavanje programa obavlja se pozivom sistemske funkcije SFC32 (SRT_DINT) u kojoj se vremensko kašnjenje upisuje kao ulazni parametar. Moguće je zaustaviti izvršavanje koje još nije aktivirano pomoću funkcije SFC33 (CAN_DINT).

Cikličko izvršavanje programa (OB30 - OB38)

Koristeći organizacijske blokove OB30-OB38 program će se izvršavati u točno određenim vremenskim intervalima. Izvršavanje OB-a je pokrenuto programskim generiranjem prekida. Intervali za pojedini OB su postavljeni po definiciji i kreću se od 5sec za OB30 do 10ms za OB38, a po potrebi mogu se promijeniti u konfiguraciji hardware-a u projektu. Svaki OB ima određeni prioritet izvršavanja koji se u S7 300 ne može mijenjati. Potrebno je osigurati da se izvršenje samog programa obavi prije isteka vremenskog intervala. U protivnom operativni sistem će pokrenuti OB80.

3.4 Funkcijski blokovi i funkcije

Funkcijski blok sadrži dio programa koji za sebe predstavlja jednu izvršnu cjelinu. Za funkcijski blok potrebno je definirati tip varijabli s kojima se radi iz razloga pridjeljivanja odgovarajućeg memorijskog prostora. U tu svrhu se popunjava deklaracijska tablica u koju treba upisati i o kojoj vrsti varijable se radi (ulazna, izlazna, ulazno-izlazna, statička ili privremena).

Funkcijski blok se poziva iz organizacijskog ili drugog funkcijskog bloka. Na taj način je ostvareno strukturirano programiranje, te bolja preglednost cijelog programa. Prilikom poziva funkcijskog bloka potrebno je formalnim parametrima deklariranim u funkcijskom bloku pridružiti aktualne parametre iz programa ili neke konkretne vrijednosti. Nakon obrade funkcijskog bloka, vrijednosti varijabli se spremaju u pridijeljeni podatkovni blok s tim da se izlazne varijable prenose u logički blok u koji se program vraća kao rezultat obrade funkcijskog bloka. Iznimka su privremene varijable čija se vrijednost gubi nakon povratka u glavni program. Ukoliko se prilikom poziva ne napravi pridruživanje aktualnih parametara formalnim, tada će se njima pridružiti odgovarajuće vrijednosti iz pridijeljenog podatkovnog bloka. To isto mogu biti i početne vrijednosti upisane u deklaracijskoj tabeli.

Funkcije za razliku od funkcijskih blokova nemaju pridijeljeni podatkovni blok; praktički nemaju svoj memorijski prostor za upisivanje vrijednosti proračunatih varijabli. Zbog toga je prilikom pozivanja funkcija obavezno pridruživanje aktualnih parametara formalnim. Funkcije se koriste za proračunavanje rezultata nekih matematičkih jednadžbi, ili za neke jednostavnije upravljačke zadaće.

3.5. Sistemski funkcijski blokovi i funkcije

Programom STEP 7 su predviđene neke radnje i operacije koje nije potrebno programirati jer se već nalaze integrirane unutar CPU-a u obliku sistemskih blokova i funkcija. Dobrim poznavanjem i korištenjem ovih logičkih blokova znatno je olakšano kreiranje projekta i samo programiranje.

Sistemski funkcijski blokovi

Predstavljaju dio operativnog sistema i da bi im se pristupilo potrebno ih je učitati iz CPU-a. Kao i funkcijski blokovi moraju imati pridijeljeni podatkovni blok S7 CPU sadrži sistemske funkcijske blokove namijenjene rukovanju protokolima za komunikaciju konfiguriranih sistema, te nekim drugim funkcijama specijalne namjene.

Sistemske funkcije

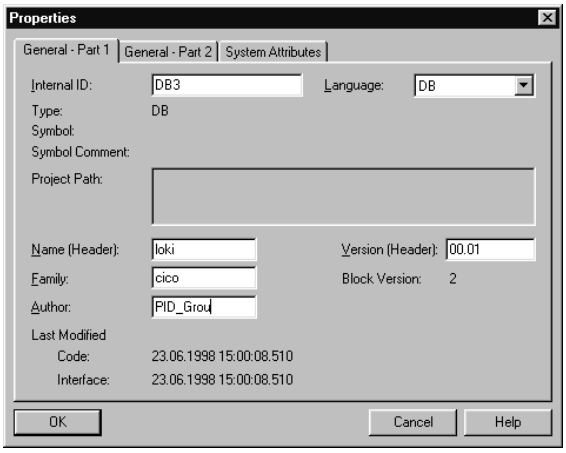
Za sistemske funkcije nije potreban posebni podatkovni blok, a način pozivanja je isti kao i za obične funkcije. S7 CPU sadrži sistemske funkcije sljedeće namjene :

- provjera programa
- rukovanje satom i mjeračima vremena
- prenošenje podataka
- prenošenje naloga iz jednog CPU-a u druge u višeračunalnom (engl. multicomputing) načinu rada
- rukovanje OB-ovima na vrijeme dana i OB-ovima sa kašnjenjem
- rukovanje sinkronim i asinkronim greškama, te prekidima
- dijagnostika sistema
- osvježavanje stanja procesa i procesiranje polja bitova
- adresiranje modula
- komunikacija globalnih podataka
- komunikacija u nekonfiguriranom sistemu
- generiranje poruka vezanih za blokove.

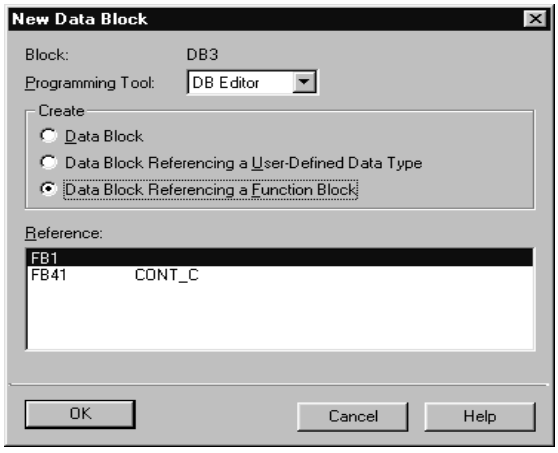
3.6 Podatkovni blokovi

Podatkovni blokovi služe za smještanje podataka koje upotrebljava korisnički program. Oni za razliku od logičkih blokova ne sadrže STEP 7 instrukcije.

Podaci ostaju u podatkovnim blokovima i onda kada se ti blokovi ne izvršavaju. Postoje dvije vrste podatkovnih blokova, to su posebni (instance) podatkovni blokovi i zajednički (shared) podatkovni blokovi. Definiranje jednog ili drugog bloka vrši se u zaglavlju bloka (vidi sliku 3.1.) nakon otvaranja podatkovnog bloka iz SIMATIC Managera naredbom **Insert → S7 Block → 1 Data Block** otvara se zaglavlje kao na slici 3.6. u kojem treba zadati ime, porodicu i autora. Pri prvom otvaranju bloka pojavljuje se prozor kao na slici 3.7. u kojem se definira kakav će biti tip bloka. Ako se pravi posebni ili pridijeljeni podatkovni blok tada treba odabrati funkcijski blok kojem je on pridružen. To govori da prvo treba napraviti i snimiti funkcijski blok te mu potom napraviti posebni podatkovni blok.



Sl. 3.6. Zaglavlje podatkovnog bloka



Sl. 3.7. Odabir vrste podatkovnog bloka

Posebni podatkovni blokovi

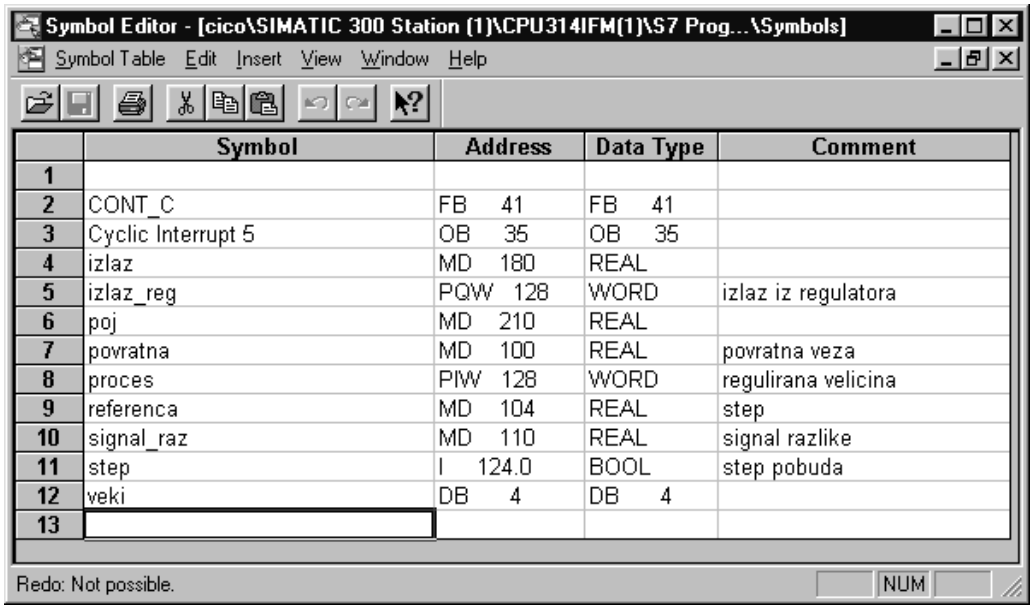
Nakon što se popuni deklaracijska tablica u zaglavlju funkcijskog bloka s varijablama relevantnim za taj blok, kreira se posebni podatkovni blok koji je u stvari preslikana deklaracijska tablica, ali sa rezerviranim mjestom u memoriji. Na taj način se proračunate vrijednosti varijabli spremaju na točno određena mjesta. Dakle, svaki funkcijski blok ima barem jedan pridijeljeni (posebni) podatkovni blok. No, moguće je da jedan funkcijski blok ima i više pridijeljenih podatkovnih blokova. Ako npr. funkcijski blok vrši upravljanje motorom tada se može taj blok koristiti za upravljanje više motora iste strukture ali različitih parametara koji su spremljeni u različitim podatkovnim blokovima. Zbog toga se prilikom poziva funkcijskog bloka uvijek navede i njemu pridijeljeni podatkovni blok, te se izvrši i pridruživanje aktualnih parametara formalnim.

Zajednički podatkovni blokovi

Za razliku od posebnih podatkovnih blokova, zajednički blokovi sadrže podatke kojima može pristupiti bilo koji logički blok, pa se stoga formiraju neovisno. U zajedničkom podatkovnom bloku svi su podaci statičkog tipa. Mogu se upisivati simbolički ili izravnom adresom, sa ili bez početne vrijednosti.

3.7. Tablica simbola

Osim simbola navedenih u deklaracijskoj tablici mogu se pridijeliti simbolička imena za varijable koje se koriste u više blokova, i imena za same blokove. Simbolička imena varijabli i blokova upisuju se u tablicu simbola prikazanu na slici 3.8.



	Symbol	Address	Data Type	Comment
1				
2	CONT_C	FB 41	FB 41	
3	Cyclic Interrupt 5	OB 35	OB 35	
4	izlaz	MD 180	REAL	
5	izlaz_reg	PQW 128	WORD	izlaz iz regulatora
6	poj	MD 210	REAL	
7	povratna	MD 100	REAL	povratna veza
8	proces	PIW 128	WORD	regulirana velicina
9	referenca	MD 104	REAL	step
10	signal_raz	MD 110	REAL	signal razlike
11	step	I 124.0	BOOL	step pobuda
12	veki	DB 4	DB 4	
13				

Sl. 3.8. Tablica simbola

Tablica simbola otvara se iz blok editora naredbom **Options→Symbol Table**. Tablica simboličkih imena sastoji se od polja naziva bloka (Symbol), polja adrese bloka (Address), polja tipa bloka (Data Type), i polja komentara (Comment).

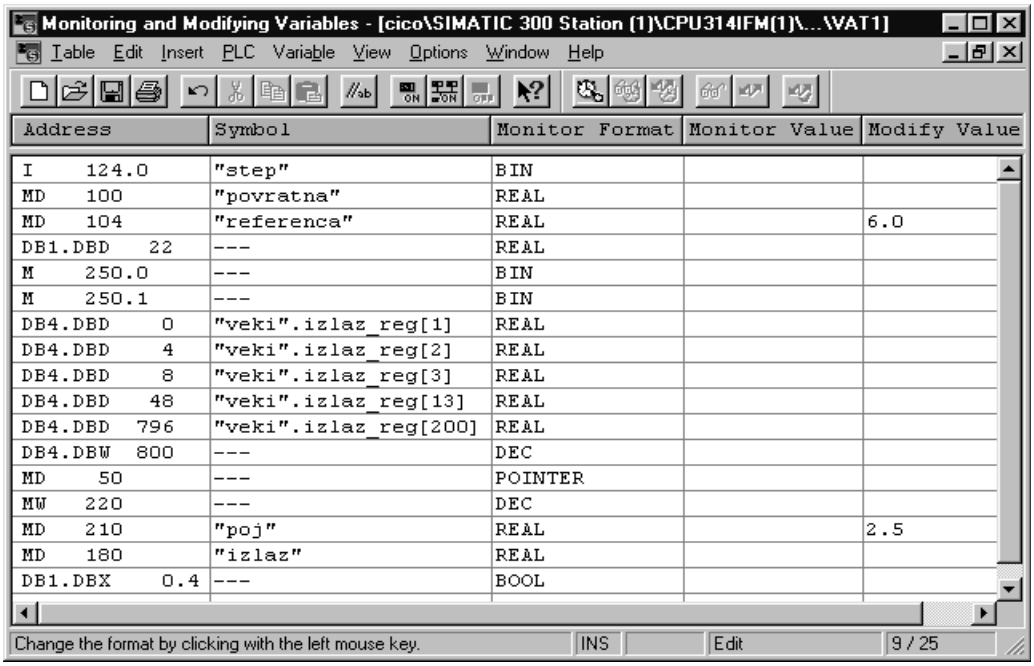
Simboli postaju aktivni tek nakon snimanja tablice. Ako je tokom programiranja potrebno mijenjati tablicu simbola, mora se nakon izmjene tablice zatvoriti projekt i ponovno ga otvoriti kako bi novi simboli bili upotrebljivi u programskim blokovima. STL Editor će simboličke nazive blokova kao i adrese podataka iz zajedničkih DB-a označavati sa navodnicima (″..″), a varijable iz deklaracijskih tablica znakom # ispred varijable.

3.8. Tablica varijabli

Prilikom izvođenja programa potrebno je imati uvid u ponašanje pojedinih varijabli i mogućnost njihove promjene. To se može postići uvrštavanjem varijabli u tablicu varijabli (slika 3.9.), koja se dobiva iz SIMATIC Managera pomoću **Insert→S7 Block→Variable Table (VAT)**.

Tablica se popunjava tako da se upiše adresa podatka kojeg treba pratiti ili mijenjati, te se odabere oblik prikaza. Nakon povezivanje tablice s odgovarajućom konfiguracijom CPU-a, treba uključiti opciju za praćenje varijabli u On-line režimu rada naredbom **Variable→Monitor**.

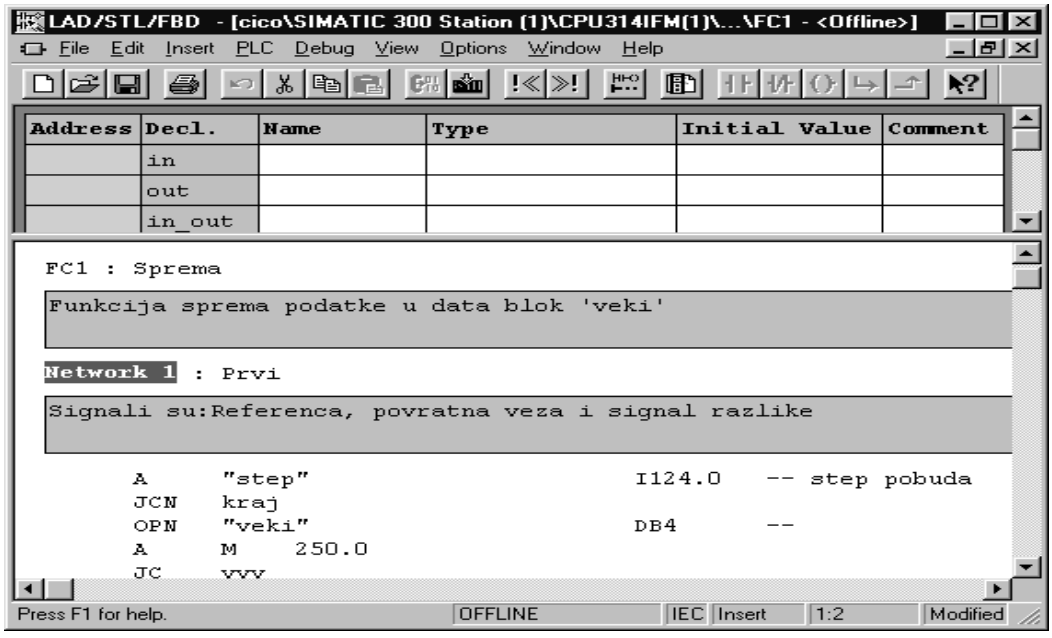
U tablici postoji opcija koja dozvoljava promjenu vrijednosti varijabli u On-line režimu rada, tzv. forsiranje varijabli. Izvodi se tako da se u polje *Modify Value* upiše nova vrijednost varijable i aktivira naredbom **Variable→Modify**.



Sl. 3.9. Tablica varijabli.

3.9. Naredbe i instrukcije programa 'STEP 7'

Otvaranjem funkcijskih blokova (FB) ili funkcija (FC), to jest dvostrukim pritiskom miša u SIMATIC Manageru otvara se novi prozor koji omogućava rad sa naredbama (editor). Editor je prikazan na slici 3.10.



Sl. 3.10. Editor za pisanje instrukcija logičkih blokova.

Pošto se otvori određeni logički blok (u prikazanom slučaju funkcijski blok FB), prvo se može (a i ne mora) ispuniti zaglavlje bloka u kojemu se upisuje ime bloka i komentar. Ispod toga se upisuje ime 'mreže' (engl. network) i njen komentar. On obično sadrži funkciju mreže i popis varijabli koje su tu korištene.

Zatim se ispunjava deklaracijska tablica u kojoj se navode varijable s kojima se radi u tom bloku. Zadaje se ime, tip, početna vrijednost i komentar varijable. Adresa varijable se postavlja sama, i to po redu punjenjem slijedećeg slobodnog memorijskog prostora i njih se ne može mijenjati. Tip varijable se postavlja prema njenoj funkciji i potrebi. Važno je napomenuti da je veoma bitno paziti na tipove varijabli kod programiranja i operacija nad njima. Mogući tipovi podataka njihova veličina to jest zauzeće memorije i raspon vrijednosti u kojima se kreću prikazani su u tablici 3.2. Početna vrijednost mora biti valjano zapisana prema tipu varijable. Ako se ne zada, sama će se postaviti na nultu vrijednost. Komentar se postavlja prema želji ako to olakšava rad i naknadno razumijevanje.

Tip podatka	Veličina u bitovima	Raspon vrijednosti (od najmanje do najveće)
BOOL	1	0 / 1
BYTE	8	2#00000000 - 2#11111111 ili B#16#0 - B#16#FF
WORD	16	W#16#0 - W#16#FFFF
DWORD	32	DW#16#0 - DW#16#FFFFFFFF
CHAR	8	'A' - 'Z'
INT	16	-32 768 - +32767
DINT	32	L#-2147483648 - L#2147483647
REAL	32	- 3.402823E+38 - -1.175494E-38 0.0 +1.175494E-38 - + 3.402823E+38
ARRAY	Automatski se postavlja	Ovisno o tipu podataka.

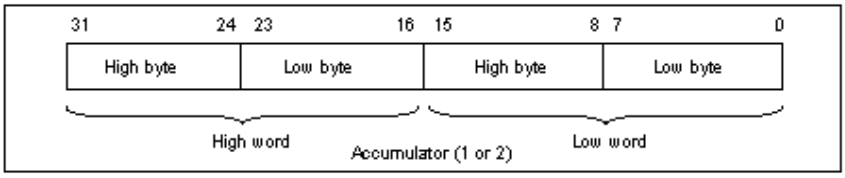
Tablica. 3.2. Tipovi podataka varijabli.

3.10 Registri PLC-a

Akumulatori

Postoje dva 32-bitna akumulatora opće namjene za procesiranje bajtova, riječi i dvostrukih riječi (slika 3.11.). U akumulator se mogu očitavati konstante i vrijednosti sa memorijskih lokacija, te vršiti logičke i aritmetičke operacije nad njima. Prenošenje rezultata na mjesta u memoriji obavlja se sa prvim akumulatorom (ACCU 1). Operacije sa akumulatorima izvedene su na principu stoga :

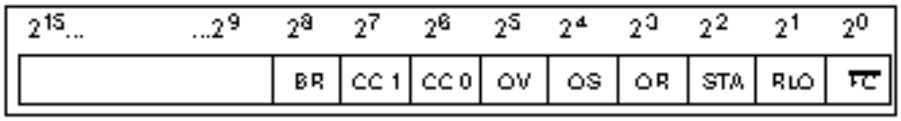
- Očitavanje se vrši uvijek na akumulator 1, a prijašnji sadržaj se pomiče na akumulator 2
- Instrukcije premještanja ne mijenjaju sadržaj akumulatora osim naredaba TAR1 i TAR2
- Instrukcija TAK izmjenjuje sadržaje akumulatora 1 i 2



Sl. 3.11. Akumulator.

Statusna riječ

Statusna riječ je 16-bitni registar pri čemu osnovnu riječ čini prvih devet bitova, dok su ostali bitovi bez značaja. Bitovi u statusnoj riječi predstavljaju informaciju o statusu i stanju procesora prilikom izvršavanja naredbi programa. Statusna riječ je prikazana na slici 3.12.



Sl. 3.12 Statusna riječ.

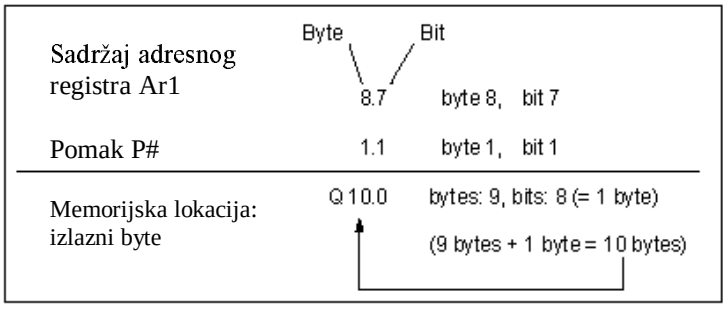
Programer može očitati i izmijeniti statusne bitove, te samo očitati bitovi za dojavu greške. Bitovi najvećeg značaja za programiranje zbog naredaba uvjetnih skokova su : CC0, CC1 i RLO. CC0 i CC1 (condition code) daju informaciju o sljedećim rezultatima ili bitovima.

- Rezultati matematičkih operacija
- Rezultati usporedbe
- Rezultati digitalnih operacija
- Bitovi koji su izbačeni van prilikom pomicanja ili rotiranja

RLO (result of logic operation) postavlja se kao rezultat logičkih instrukcija i operacija usporedbe.

Adresni registri

Adresni registri koriste se prilikom indirektnog adresiranja. Na raspolaganju su dva 32-bitna adresna registra (AR1 i AR2). S pomoću njih može se pristupiti bitovima, bajtovima, riječima i dvostrukim riječima bilo gdje u memorijskom prostoru. Adresni registar može sadržavati adresu memorijskog prostora prikazanu u formatu pokazivača. Tada je prilikom pristupa podatka na toj lokaciji moguće koristiti pomak adrese koji je također u formatu pokazivača. Na taj način je dosta olakšan sekvencijalan pristup podacima unutar nekog bloka memorije. Slika 3.13 prikazuje način proračuna memorijske lokacije ako je zadan izlaz Q[AR1, P#1.1].

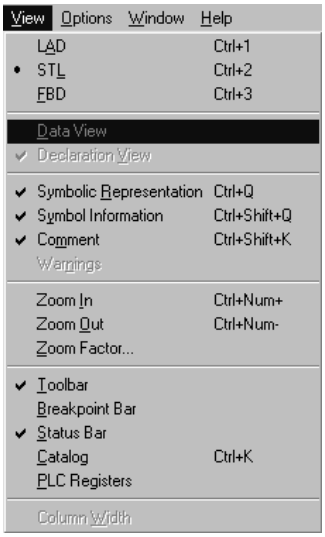


Sl. 3.13. Primjer proračuna memorijske lokacije izlaza Q [AR1, P#1.1].

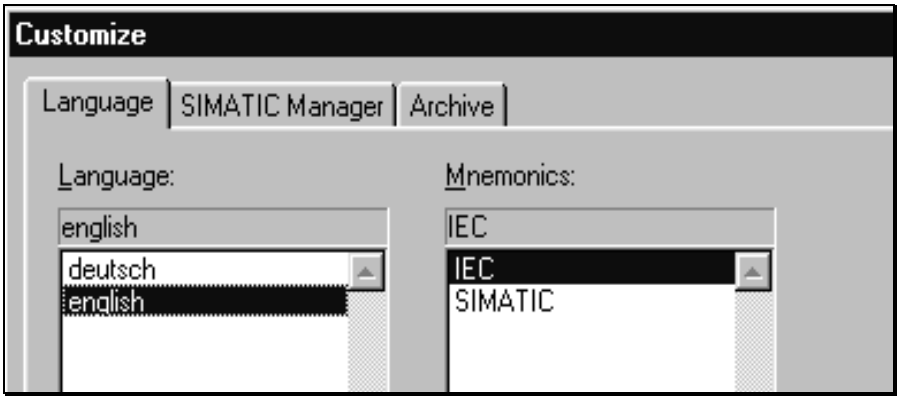
3. 11 Pisanje koda naredbe

Naredbe se mogu pisati u tri oblika s kojima raspolaže STEP 7. To su STL (Statement List programming), FBD (Function Block Diagram) i LAD (Ladder Logic programming). Do njih se dolazi pomoću naredbe **View** kao na slici 3.14. Za programiranje 'sekvencijalnog' to jest slijednog upravljanja preporuča se upotreba FBD dijagrama zbog puno veće preglednosti spajanja logičkih funkcija, dok za kompleksnije primjene zbog većih matematičkih mogućnosti treba upotrebljavati STL način pisanja koda koji je sličan 'asemblerskim' strojnim jezicima drugih procesora.

Za pisanje u STL-u potrebno je još namjestiti IEC međunarodni standard do kojeg se dolazi iz SIMATIC Managera naredbom **Options → Customize** i u potom otvorenom prozoru kao na slici 3.15. odaberemo IEC standard.



Sl. 3.14. Prikaz izbornika View.



Sl. 3.15. Prozor za namještanje IEC standarda pisanja naredbi.

STL je asemblerski program pa prema tome i same njegove naredbe tako izgledaju. Primjer linije koda na kojoj će se objasniti pojedine dijelove poslužit će sljedeća naredba:

```
L1:  L     MW10        //DOVOĐENJE STEPA
```

Na prvom mjestu se nalazi labela ako je ona potrebna i iza nje obavezno dolazi dvotočka. Zatim slijedi mnemonik to jest operacija. Iza operacije slijedi adresa operanda nad kojim se vrši operacija. Neke naredbe nemaju adresu ili se ona podrazumijeva. I na kraju kodne linije može stajati komentar koji je odvojen sa dvije kose crte. Komentar služi za lakše snalaženje u programu.

3.12 Tipovi instrukcija

U slijedećem odlomku bit će nabrojene neke grupe naredbi, bitnih za razumijevanje osnovnih značajki programiranja.

1. instrukcije za logičke operacije (nad bitovima i nad riječima),
2. instrukcije za pomicanje i rotiranje,
3. instrukcije za prijenos varijabli,
4. instrukcije za adresiranje,
5. instrukcije za cjelobrojnu aritmetiku,
6. instrukcije za aritmetiku pomičnog zareza,
7. instrukcije za pretvorbu veličina iz jednog zapisa u drugi,
8. instrukcije za usporedbu vrijednosti,
9. instrukcije za rad s brojilima i mjeračima vremena te
10. instrukcije za pozive logičkih blokova.

3.13 Adresiranje

Postoje tri načina na koja se može pristupiti nekom podatku to jest njegovoj memorijskoj lokaciji, to su neposredno, izravno (apsolutno) ili simboličko adresiranje i neizravno adresiranje.

Neposredno adresiranje je zapravo zadavanje samog podatka uz instrukciju.

```
L    12                    // punjenje akumulatora 1 cijelim brojem 12
```

Izravno ili apsolutno adresiranje je zadavanje memorijskog područja to jest lokacije na kojoj se podatak nalazi. Popis identifikatora memorijskog područja nalazi se u poglavlju 2.1. u tablici 2.1. Ovdje je dano nekoliko primjera iz kojih se vidi ideja izravnog adresiranja.

```
OPN    DB2                  // otvaranje data bloka broj 2

L        DB2,DBB4           // učitaj u aku 1 byte sa lokacije četiri iz data bloka dva
```

Moguće je koristiti i simbol, bilo definiran u deklaracijskoj tablici bloka ili u simbol tablici projekta (vidi simbol tabele u trećem poglavlju).

```
OPN    'veki'               // otvaranje data bloka dva simboličkog imena 'veki'

L        'veki'.'sig'       // učitaj u aku 1 podatak 'sig' iz data bloka 'veki'
```

Neizravno adresiranje sastoji se u tome da se vrši operacija nad lokacijom na koju pokazuje adresni registar koji se ranije napuni. Adresni registar sadrži adresu memorijske lokacije. Postoje dva adresna registra AR1 i AR2. Naredba se koristi zajedno sa pomakom P# koji zapravo

povećava vrijednost adresnog registra za vrijednost P#. Cjelobrojna vrijednost pomaka odnosi se na bayt, a decimalna na bit.

```
T     W[AR1,P#4.0]      // prijenos podatka tipa word iz aku 1 na mjesto AR1+4.0

L     D[AR1,P#8.0]      // punjenje aku 1 podatkom tipa DWORD (dvostrukom riječi) sa
                           lokacije AR1+8.0 (pomak od četiri bayta)
```

4. FUNKCIJSKI BLOK PID REGULATORA

4.1 Realizacija PID regulatora

Najčešće i najšire korišten algoritam regulacije, PID regulator, moguće je jednostavno realizirati u SIMATIC-u S7 korištenjem gotovih funkcijskih blokova STEP 7 programskog paketa. Postoje već unaprijed isprogramirani funkcijski blokovi za regulaciju koji se nalaze i unutar memorije S7 314IFM, koji se lako uključuju u korisnički program za konkretan problem regulacije. Postoje gotova tri funkcijska bloka: FB41 "CONT_C", FB42 "CONT_S" i FB43 "PULSGEN", kojima odgovaraju sistemski funkcijski blokovi SFB41, SFB42 i SFB43.

Funkcijski blok "CONT_C" koristi se u SIMATIC S7 programabilnim kontrolerima za regulaciju tehničkih procesa s kontinuiranim ulazom i izlazom.

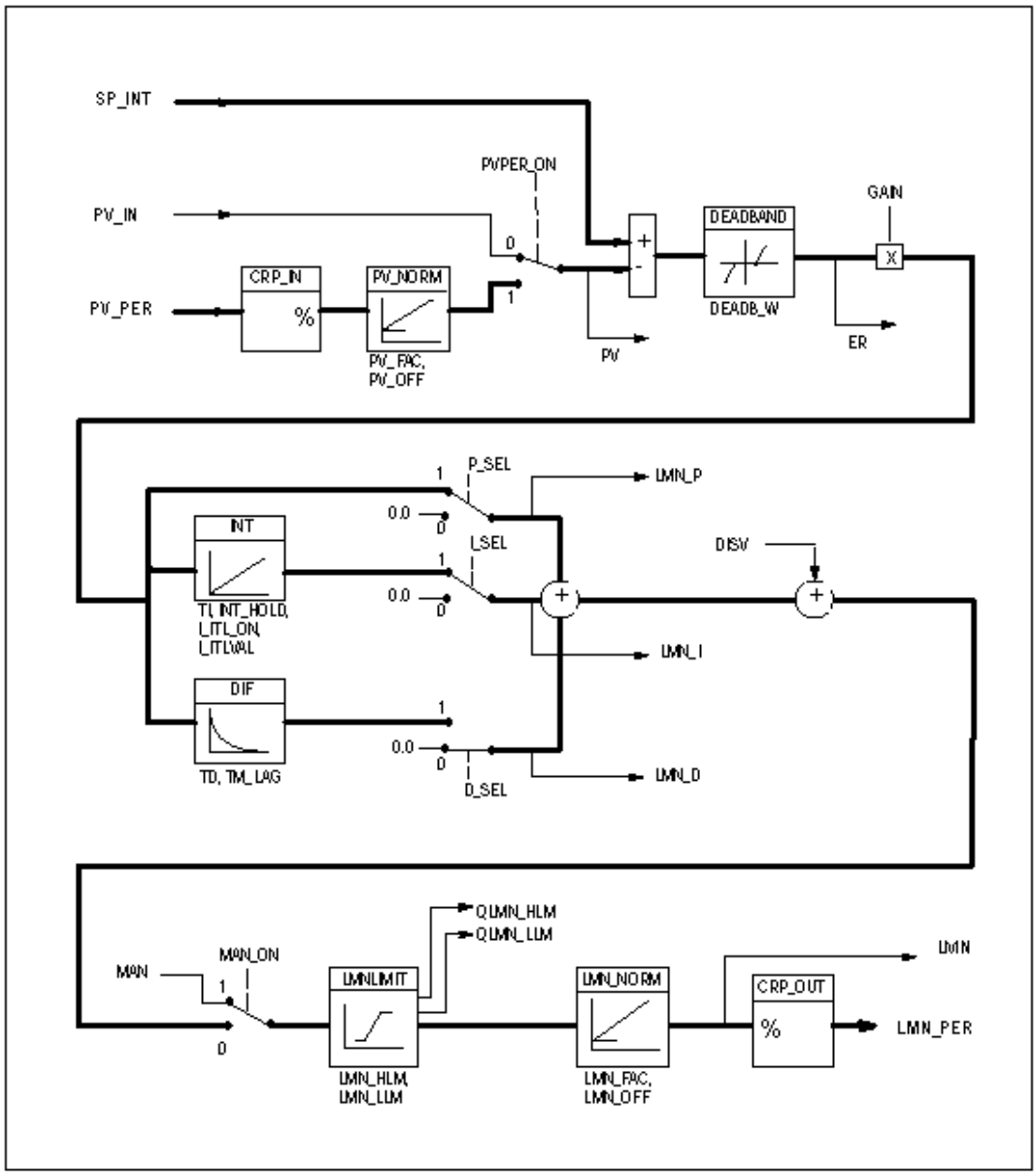
Funkcijski blok "CONT_S" koristi se u SIMATIC S7 programabilnim kontrolerima za regulaciju tehničkih procesa s digitalnim izlazom regulatora za pojačala snage s integralnim svojstvom. U ovom poglavlju neće se govoriti o tom funkcijskom bloku.

Funkcijski blok "PULSGEN" se koristi za PID regulatore s pulsnim izlazom za proporcionalna pojačala snage. PULSGEN pretvara amplitudu signala u trajanje pulsa radeći u biti pulsno širinsku modulaciju. Ovaj funkcijski blok se također neće obrađivati u ovom poglavlju.

4.2 Kontinuirana regulacija korištenjem FB41 "CONT_C"

Funkcijski blok 41 "CONT_C" se koristi u SIMATIC S7 programabilnim kontrolerima za regulaciju tehničkih procesa s kontinuiranim ulaznim i izlaznim varijablama. Tijekom pridruživanja parametara, te izvršavanja programa konkretne regulacije mogu se uključiti i isključiti različite podfunkcije PID algoritma, čime je regulator moguće prilagoditi procesu te koristiti neke naprednije algoritme adaptacije parametara PID regulatora, kao i promjene čitave strukture regulatora. Moguće je uključivanjem nekih ulaza FB-a dobiti P, PI, PD, I, PID, te i čisti D regulator.
Također moguće je koristiti ovaj regulator kao PID regulator za održavanje zadane vrijednosti, u kaskadnoj regulaciji, u regulaciji miješanja ili u regulaciji omjera veličina. Funkcija regulatora koja je temeljena na PID regulaciji može se proširiti dodavanjem pulsnog generatora.

Na slici 4.1. prikazana je blok shema funkcijskog bloka 41 "CONT_C" sa svim ulazima i izlazima. FB41 uključuje kompletan PID algoritam s kontinuiranim izlazom te mogućnošću postavljanja izlaza bloka ručno.



Sl. 4.1. Blok shema funkcijskog bloka FB41 “CONT_C” za kontinuiranu regulaciju.

Na blok shemi mogu se uočiti sljedeći dijelovi:

•**grana postavne vrijednosti**, u kojoj je postavna vrijednost upisana u formatu pomičnog zareza na SP_INT ulazu;

•**grana procesne varijable**, u kojoj procesna varijabla može biti u obliku pomičnog zareza ili ulazna vrijednost u I/O obliku na PV_PER ulazu. Funkcija CRP_IN pretvara periferni I/O zapis u oblik pomičnog zareza u intervalu -100% -+ 100% prema jednadžbi:

$$\text{Izlaz CRP\_IN} = \text{PV\_PER} * \frac{100}{27648}, \qquad (4\text{–}1)$$

funkcija PV_NORM normira izlaz funkcije CRP_IN prema jednadžbi:

$$\text{Izlaz PV\_NORM}=(\text{Izlaz CRP\_IN})*\text{PV\_FAC}+\text{PV\_OFF}; \quad (4\text{–}2)$$

- **signal razlike** predstavlja razliku između postavne vrijednosti i procesne varijable. Za sprečavanje malih konstantnih oscilacija uslijed kvantizacije izlaza regulatora ili uslijed šuma na ulazu koji se ne može otkloniti na signal razlike se primjenjuje mrtva zona (uz inicijalno isključeno stanje).
- **PID algoritam** funkcionira kao postavni algoritam. Proporcionalno, integralno (INT), i derivativno (DIF) djelovanje je spojeno paralelno te se može uključiti i isključiti svako zasebno, što omogućava različite konfiguracije regulatora, uključujući i čisto integralno i derivacijsko djelovanje;

- **Ručno zadanu vrijednost** izlaza regulatora je moguće postaviti prebacivanjem u ručni režim rada, čime je izlaz regulatora kratko spojen na tu vrijednost. Integrator (INT) je interno postavljen na LMN-LMN_P-DISV a derivator (DIF) je postavljen na 0. Time se osigurava da ponovno uključivanjem u automatski režim rada ne uzrokuje iznenadne promjene na izlazu regulatora;
- **Izlaz regulatora** se može ograničiti korištenjem funkcije LMNLIMIT. Signalni bitovi indiciraju kada je limit prekoračen. Funkcija LMN_NORM normira izlaz funkcije LMNLIMIT prema jednadžbi:

$$\text{LMN} = (\text{Izlaz LMNLIMIT})*\text{LMN\_FAC}+\text{LMN\_OFF}. \quad (4\text{–}3)$$

Izlaz regulatora se također može dobiti u I/O obliku za analogni izlaz. Funkcija CRP_OUT pretvara zapis pomičnog zareza vrijednosti LMN u periferni I/O zapis prema jednadžbi:

$$\text{LMN\_PER} = \text{LMN}*\frac{27648}{100}; \quad (4\text{–}4)$$

- **Unaprijedna regulacije (Feedforward)** Poremećajni signal se može dovesti na DISV ulaz. FB41 "CONT_C" sadrži rutinu za kompletan restart koja se izvršava kada je ulazni parametar COM_RST postavljen u TRUE. Tijekom startanja, integrator je interno postavljen na početnu vrijednost I_ITVAL. Kada se poziva u OB-u klase cikličkih prekida integrator tada nastavlja rad počevši od ove vrijednosti. Svi drugi izlazi se postavljaju na njihove početne vrijednosti. Ovaj FB ne provjerava greške interno. Izlazni parametar pogreške RET_VAL se ne koristi.

- **Ulazni parametri**: Tablica 4.1. sadrži opis ulaznih parametara FB41 "CONT_C".

Parametar	Tip	Područje vrijednosti	Default	Opis
COM_RST	BOOL		FALSE	POTPUN RESTART Blok sadrži rutinu za potpun restart koja se izvršava kada je ulaz COM_RST postavljen.
MAN_ON	BOOL		TRUE	UKLJUČENJE RUČNO POSTAVLJENOG IZLAZA Ako je ovaj ulaz aktivan regulacijski krug je prekinut. Na izlaz regulatora se dovodi ručno postavljena vrijednost izlaza.
				UKLJUČENJE PROCESNE VELIČINE Ako se procesna veličina čita s I/O, ulaz PV_PER

PVPER_ON	BOOL		FALSE	mora biti spojen na I/O i ulaz PVPER_ON mora biti uključen.
Parametar	**Tip**	**Područje vrijednosti**	**Default**	**Opis**
P_SEL	BOOL		TRUE	UKLJUČENJE **P** KANALA Pojedine akcije PID algoritma mogu se zasebno aktivirati ili isključiti. P komponenta se aktivira postavljanjem P_SEL ulaza.
I_SEL	BOOL		TRUE	UKLJUČENJE **I** KANALA Pojedine akcije PID algoritma mogu se zasebno aktivirati ili isključiti. I komponenta se aktivira postavljanjem I_SEL ulaza.
INT_HOLD	BOOL		FALSE	ZADRŽI INTEGRALNO DJELOVANJE Izlaz integratora može se zalediti postavljanjem ovog ulaza.
I_ITL_ON	BOOL		FALSE	INICIJALIZACIJA INTEGRATORA Postavljanjem ovoga ulaza integrator se može spojiti na ulaz I_ITL_VAL.
D_SEL	BOOL		FALSE	UKLJUČENJE **D** KANALA Pojedine akcije PID algoritma mogu se zasebno aktivirati ili isključiti. D komponenta se aktivira postavljanjem D_SEL ulaza.
CYCLE	TIME	>=1ms	T#1s	VRIJEME UZORKOVANJA Vrijeme između poziva bloka mora biti konstantno. Ovaj ulaz određuje to vrijeme.
SP_INT	REAL	-100.0…100. 0(%) ili fizik. Veličina 1)	0.0	UNUTRAŠNJA POSTAVNA VRIJEDNOST Ovaj ulaz određuje veličinu postavne vrijednosti u postotcima područja veličina A/D pretvornika.
PV_IN	REAL	-100.0…100. 0(%) ili fizik. Veličina 1)	0.0	PROCESNA VELIČINA Procesnoj veličini može se pridružiti početna vrijednost na ovom ulazu ili se može zadati vrijednost u formatu pomičnog zareza.
PV_PER	WORD		W#16#00 00	MJERENA PROCESNA VELIČINA Izlaz procesa u I/O formatu spoja se na PV_PER ulazu u regulator.
MAN	REAL	-100.0…100. 0(%) ili fizik. veličina 2)	0.0	RUČNO ZADANA VRIJEDNOST Ovaj ulaz se koristi za ručno zadavanje izlaza regulatora korištenjem funkcija za sučelje prema operatoru.
GAIN	REAL		2.0	PROPORCIONALNO POJAČANJE Ovaj ulaz određuje pojačanje regulatora.
TI	TIME	>=CYCLE	T#20s	INTEGRALNA KONSTANTA Određuje vremensku konstantu integratora (Ti).
TD	TIME	>=CYCLE	T#10s	DERIVATIVNA KONSTANTA Određuje vremensku konstantu derivativnog kanala (Td).
TM_LAG	TIME	>=CYCLE/2	T#2s	VREMENSKO KAŠNJENJE **D** KANALA Algoritam D kanala sadrži kašnjenje koje se može postaviti na ovom ulazu.
DEADB_W	REAL	>=0.0(%) ili fizikalna veličina 1)	0.0	ŠIRINA MRTVE ZONE Mrtva zona se primjenjuje na signal razlike. Ovaj parametar određuje njenu širinu.
LMN_HLM	REAL	LMN_LLM… 100(%) ili fizikal. vel.2)	100.0	GORNJI LIMIT IZLAZA REGULATORA Izlaz regulatora je uvijek ograničen gornjom i donjom granicom. Ovaj ulaz određuje gornju granicu.
LMN_LLM	REAL	-100(%)… LMN_HLM ili fizikal. vel.2)	0.0	DONJI LIMIT IZLAZA REGULATORA Izlaz regulatora je uvijek ograničen gornjom i donjom granicom. Ovaj ulaz određuje donju granicu.
PV_FAC	REAL		1.0	FAKTOR PROCESNE VELIČINE Ovim faktorom se množi procesna veličina ako se vrši prilagodba opsega ulaza.
PV_OFF	REAL		0.0	POMAK PROCESNE VELIČINE Pomak procesne veličine se dodaje ako se vrši prilagodba opsega ulaza.
				FAKTOR IZLAZA REGULATORA Ovim faktorom se množi izlaz regulatora ako se vrši

LMN_FAC	REAL		1.0	prilagodba opsega izlaza regulatora.
LMN_OFF	REAL		0.0	POMAK IZLAZA REGULATORA Pomak izlaza regulatora se dodaje ako se vrši prilagodba opsega izlaza regulatora.
Parametar	**Tip**	**Područje vrijednosti**	**Default**	**Opis**
I_ITLVAL	REAL	-100.0…100.0(%) ili fizikalna veličina 2)	0.0	POČETNA VRIJEDNOST INTEGRATORA Izlaz integratora može se ulazom I_ITL_ON spojiti na neku početnu vrijednost zadanu ovim ulazom.
DISV	REAL	-100.0…100.0(%) ili fizikalna veličina 2)	0.0	POREMEĆAJNA VELIČINA Za unaprijednu regulaciju na ovaj ulaz se dovodi poremećajna veličina.

Tablica 4.1. Opis ulaznih parametara funkcijskog bloka "CONT_C".

1) Parametri u grani postavne vrijednosti i procesne veličine s istom jedinicom.
2) Parametri u grani regulatora s istom jedinicom.

- **Izlazni parametri**: Tablica 4.2. sadrži opis izlaznih parametara FB41 "CONT_C".

Parametar	**Tip**	**Područje vrijednosti**	**Default**	**Opis**
LMN	REAL		0.0	IZLAZ IZ REGULATORA Efektivni izlaz regulatora je u formatu pomičnog zareza nalazi se na ovom izlazu.
LMN_PER	WORD		W#16 #0000	PRETVORENI IZLAZ REGULATORA Izlaz regulatora u I/O formatu može se spojiti na proces sa ovog izlaza preko analognog izlaza.
QLMN_HLM	BOOL		FALSE	DOSEGNUTA GORNJA GRANICA IZLAZA REG. Izlaz regulatora je uvijek ograničen gornjom i donjom granicom. Ovaj izlaz pokazuje da je dosegnuta gornja granica izlaza regulatora.
QLMN_LLM	BOOL		FALSE	DOSEGNUTA DONJA GRANICA IZLAZA REG. Izlaz regulatora je uvijek ograničen gornjom i donjom granicom. Ovaj izlaz pokazuje da je dosegnuta donja granica izlaza regulatora.
LMN_P	REAL		0.0	PROPORCIONALNA KOMPONENTA Ovaj izlaz sadrži izlaz iz proporcionalnog kanala regulatora.
LMN_I	REAL		0.0	INTEGRALNA KOMPONENTA Ovaj izlaz sadrži izlaz iz integralnog kanala regulatora.
LMN_D	REAL		0.0	DERIVATIVNA KOMPONENTA Ovaj izlaz sadrži izlaz iz derivativnog kanala regulatora.
PV	REAL	0	0.0	PROCESNA VARIJABLA Efektivna procesna varijabla se nalazi na ovom izlazu bloka.
ER	REAL		0.0	SIGNAL RAZLIKE Na ovom izlazu nalazi se efektivni signal razlike.

Tablica 4.2. Opis izlaznih parametara funkcijskog bloka "CONT_C".