Programirljivi logični krmilnik: programiranje - 2. del

Procesna avtomatika

Uroš Lotrič, Nejc Ilc

Funkcijski načrt: standard

Funkcijski načrt je eden od standardnih grafičnih jezikov

Nazorno prikazuje medsebojno povezanost funkcij in funkcijskih blokov

Podobni so električnim in blokovnim shemam iz analogne in digitalne tehnike

- Vsak blok ima vhode in izhode
- Povezave nakazujejo tok električnega toka v pravih vezjih

Bloki običajno predstavljajo kombinacijske funkcije

Lahko imajo tudi spomin

Standard dovoljuje tudi povratne povezave

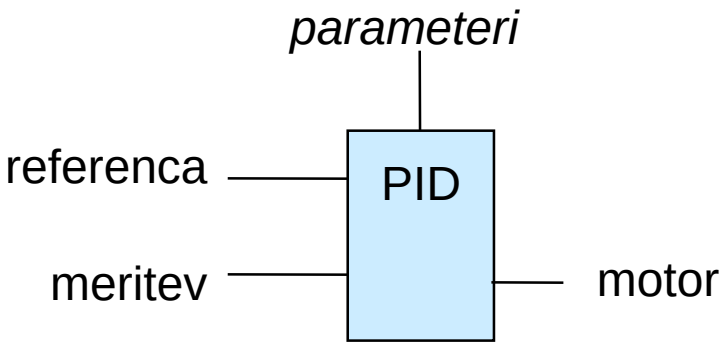
- Predpisuje definiranje vrstnega reda izvajanja povratno povezanih blokov, ne pa načina

Funkcijski načrt: standard

Funkcijski blok je definiran z

- Vmesnikom
  - Število in tipi vhodov in izhodov
- Funkcionalnostjo črne škatle (ang. black box)
  - Delovanje bloka je opisano grafično, tabelarično, s formulo, opisno

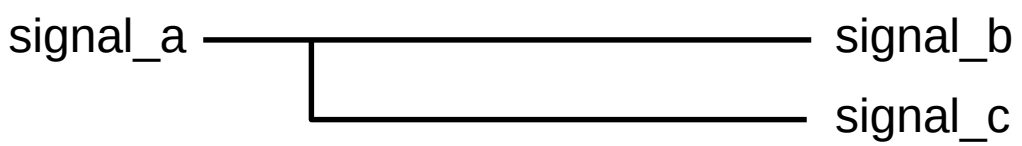
Primer:



Pravila povezovanja

- Vsak signal je povezan z natanko enim virom
- Vir, ponor in povezava morajo biti istega podatkovnega tipa

Primer:



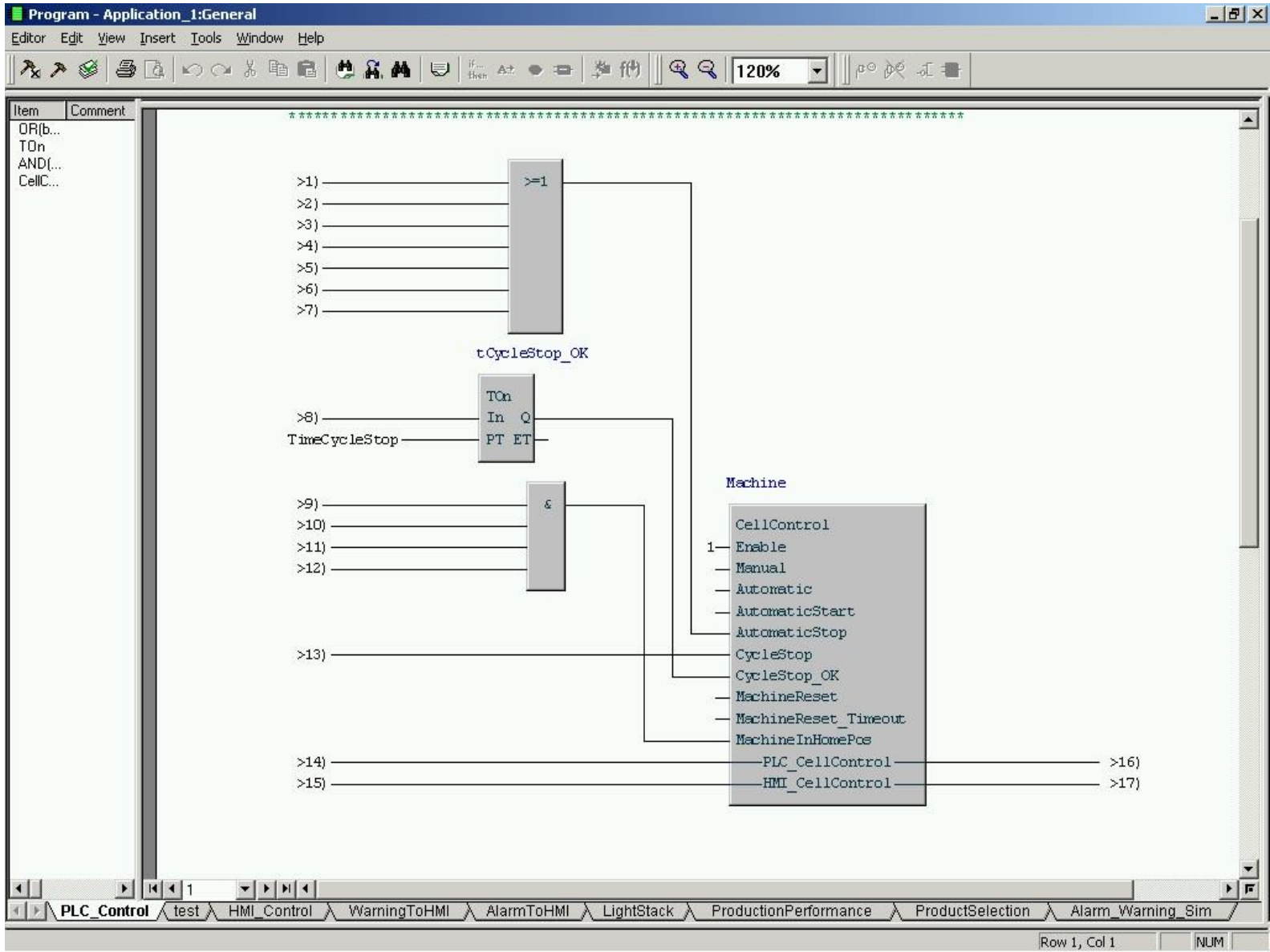
Funkcijski načrt: program

Izvajanje programa

- Od zgoraj navzdol
- Od leve proti desni
- Izjema so povratne povezave

Primer

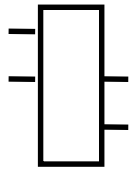
- ABB



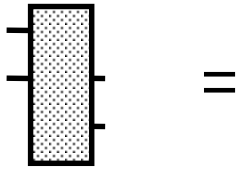
Funkcijski načrt: program

Dekompozicija problema

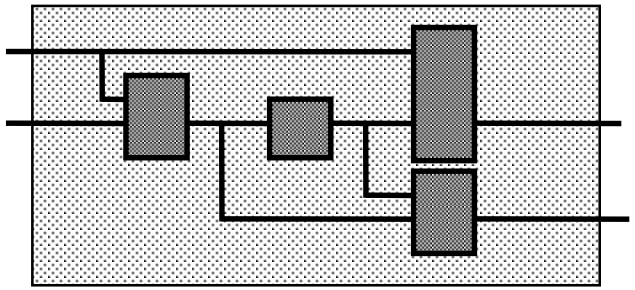
- Elementarni blok
  - Mikrokoda, zbirnik, STL, …



- Sestavljeni blok
  - Povezuje več elementarnih blokov v celoto
  - Funkcijski načrt



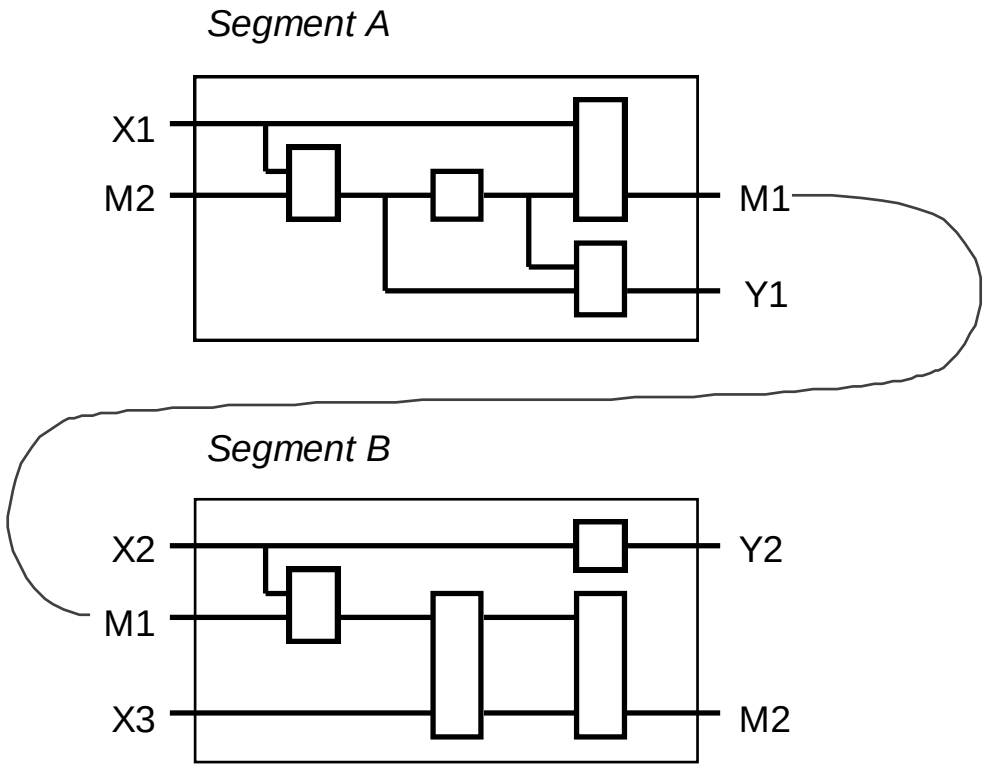
=



Funkcijski načrt: program

Segmentacija

- Zaradi večje preglednosti funkcijski načrt razdelimo v več segmentov
- Znotraj segmenta so povezave predstavljene grafično
- Segmente med seboj povezujejo spremenljivke (začasne)



Funkcijski načrt: posebnosti Siemens FBD

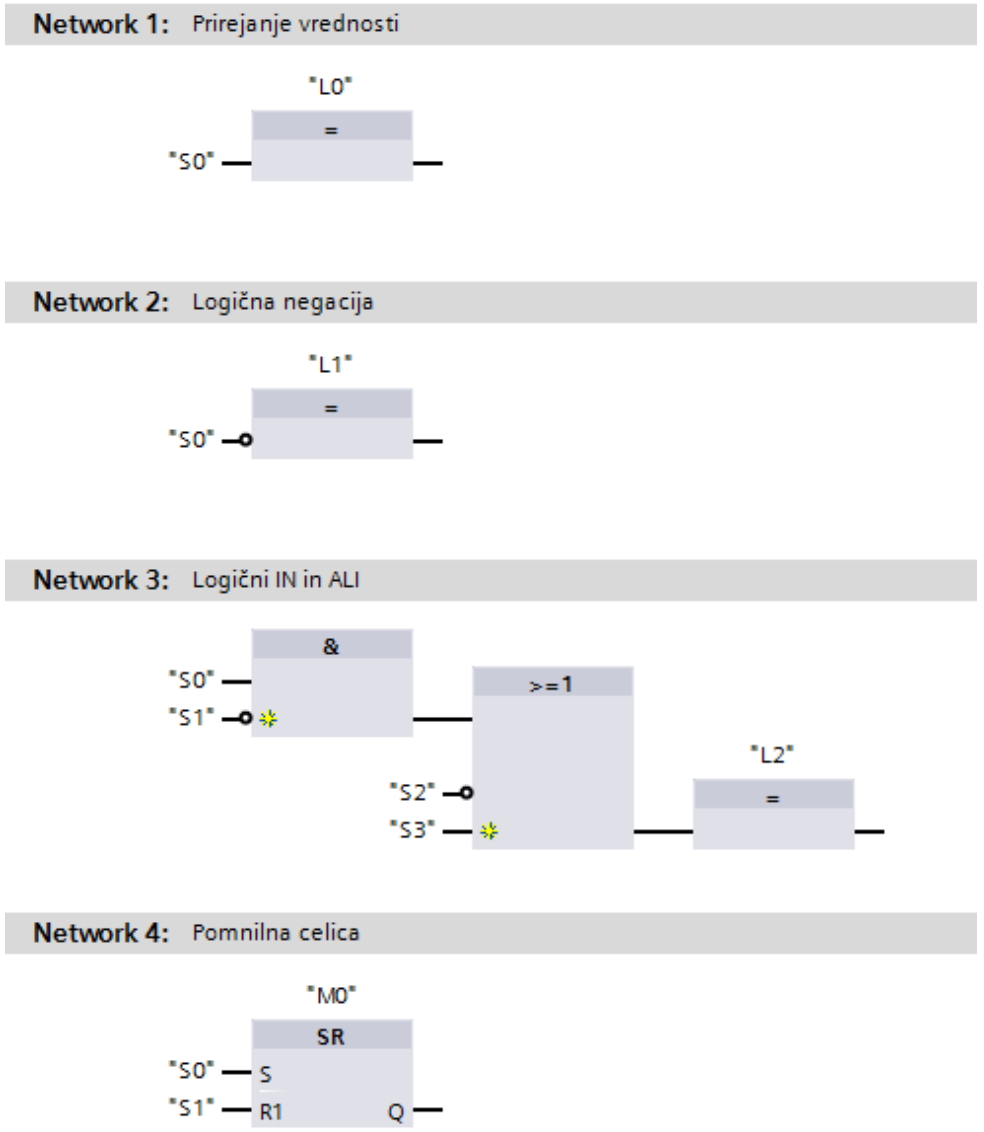
Koncept klinov (prečk) privzet iz lestvičnih diagramov

- Program se izvaja sekvenčno, klin za klinom, od leve proti desni

Dobra praksa:

- Vsi vhodi naj bodo definirani, da ne pride do neželenega obnašanja
- vsak izhod naj se v programu pojavi samo na enem mestu

Primer:
Osnovne operacije



Funkcijski načrt: posebnosti Siemens FBD

Primer:
Vodenje motorja

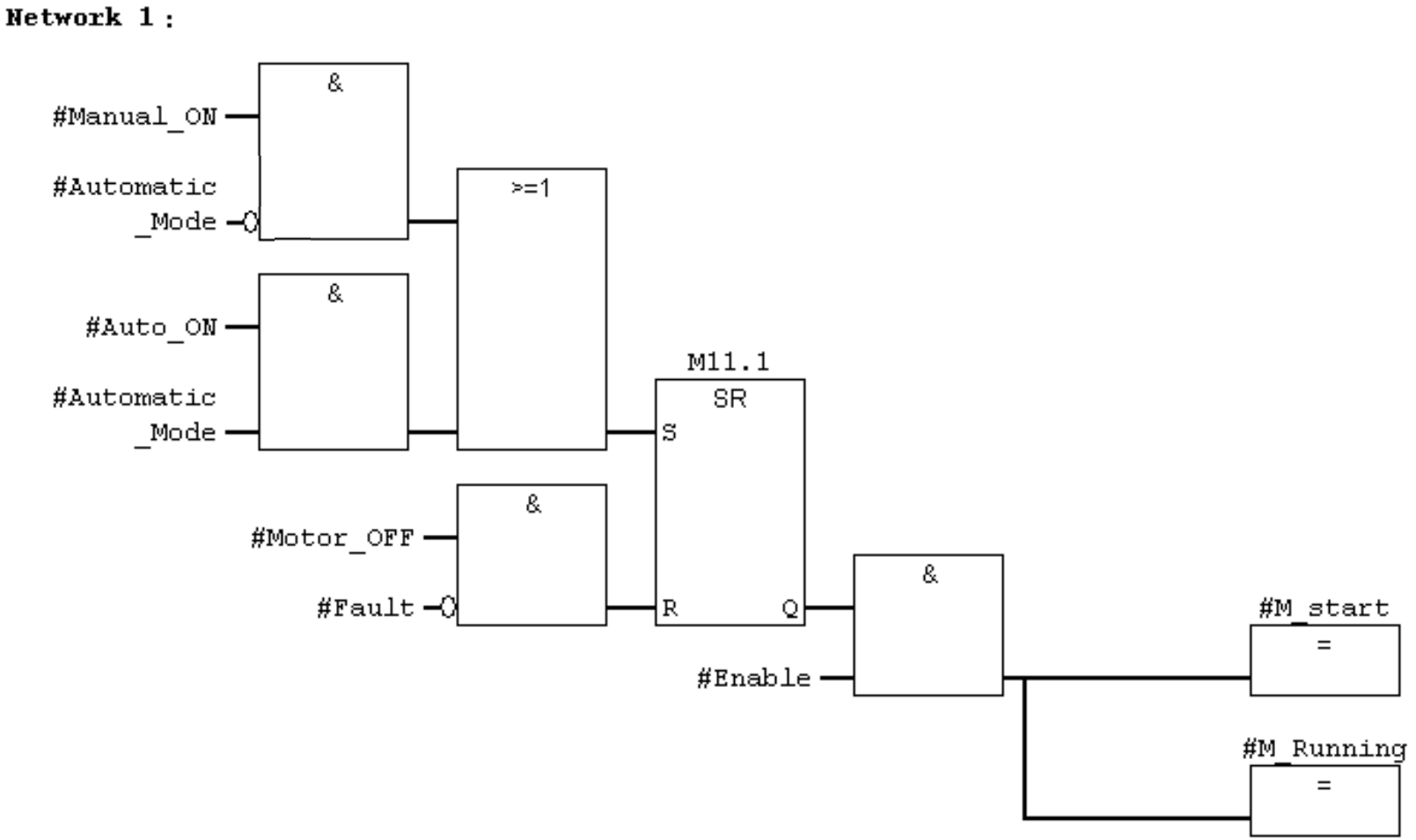


Diagram poteka: standard

SFC (ang. Sequential Flow Chart)

- Opisuje zaporedje operacij in interakcij med vzporednimi procesi
- Matematični temelji so v Petrijevih mrežah
- Stanja so povezana s prehodi

Žeton

- Stanje je aktivno ob prisotnosti žetona
- Žeton zapusti stanje, ko je izpolnjen pogoj za prehod
- Samo en prehod se lahko zgodi naenkrat
- Žeton je ob začetku programa v začetnih stanjih

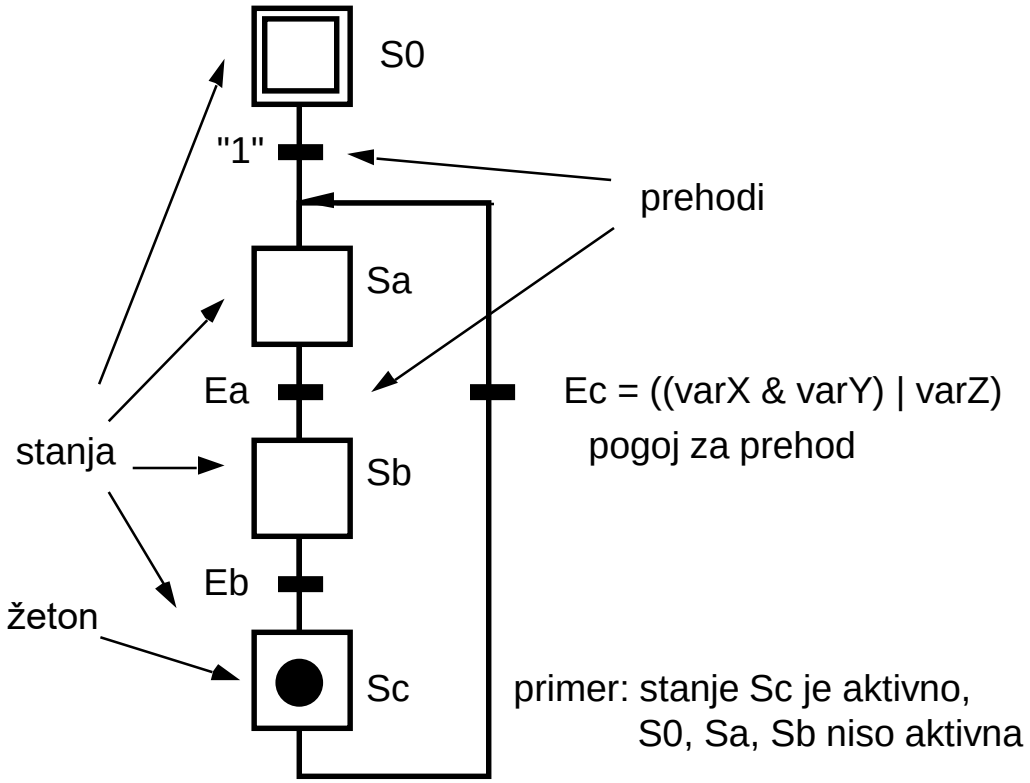


Diagram poteka: standard

Izvajanje programa

- Žeton prečka prvi aktivni prehod
  - V primeru, da sta Ea in Eb oba izpolnjena se upošteva nastavljene prioritete ali naključje
- Ko je pogoj Ee izpolnjen, se žeton razdeli na obe povezani stanji
- Ko so prisotni vsi razdeljeni žetoni in je pogoj Ef izpolnjen,
  pot nadaljuje en sam žeton

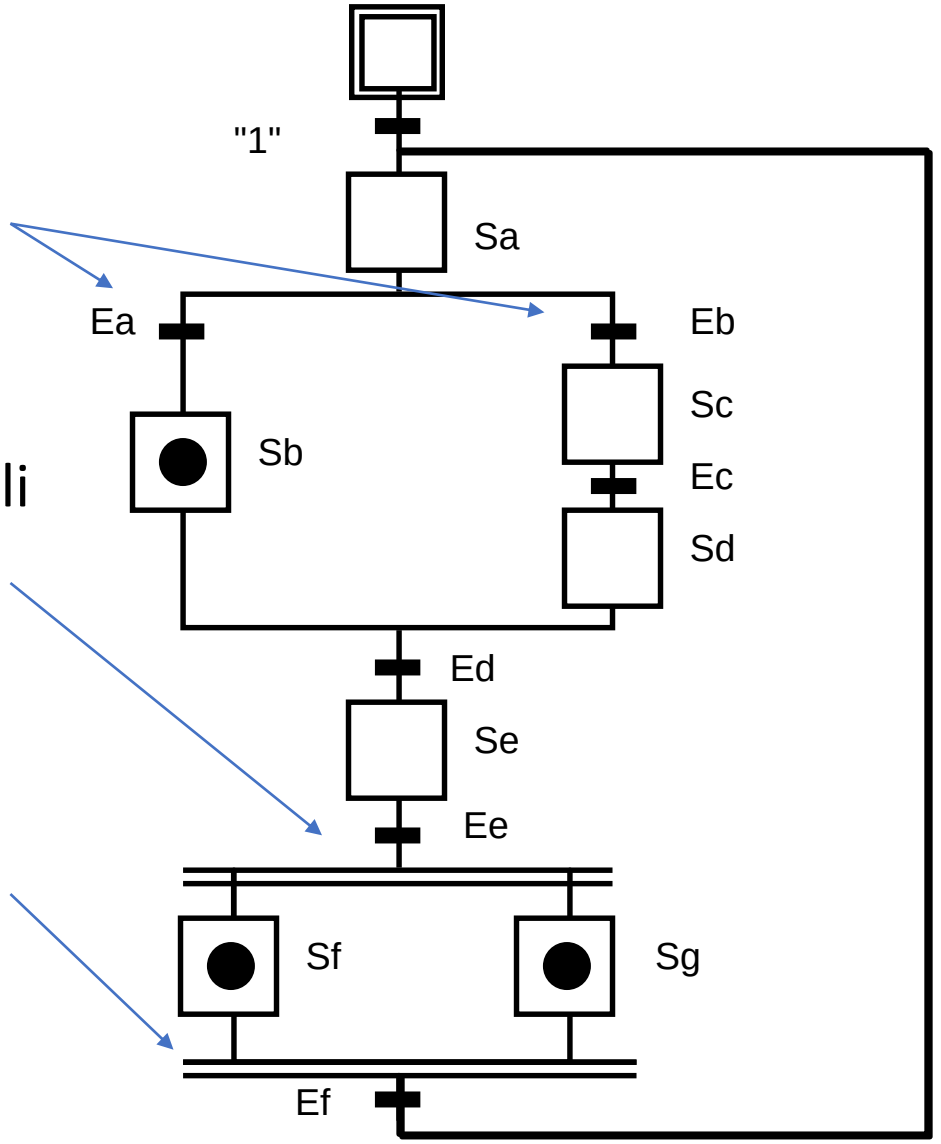


Diagram poteka: program

Zapleteni diagrami

- Smrtni objem (ang. deadlock)
- Nekontrolirano delo z žetoni

Rešitev

- Omejitve urejevalnika
- Funkcije urejevalnika

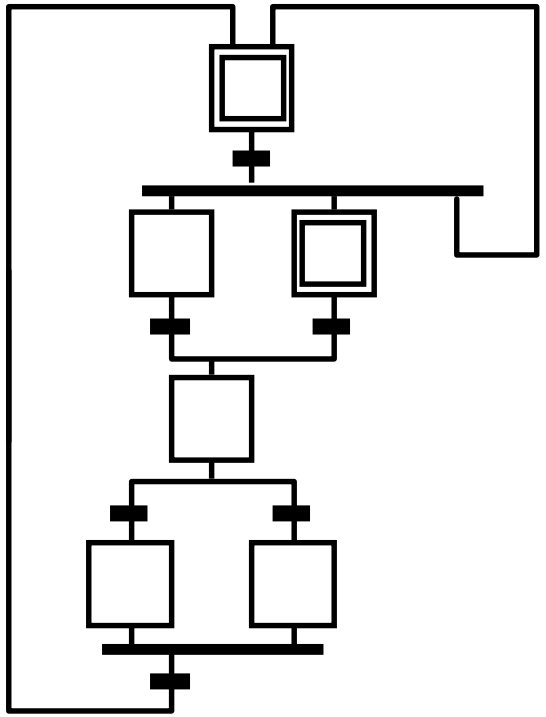
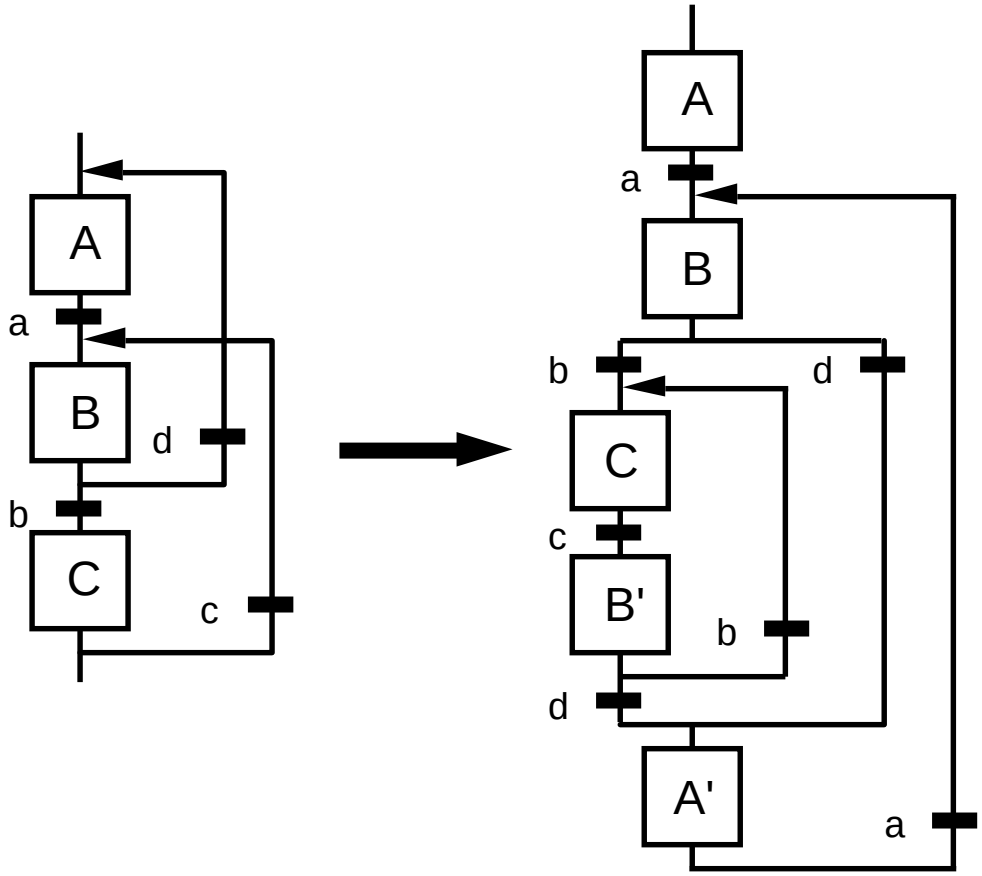


Diagram poteka: program

Zapis diagrama v strukturirano obliko

- Podvajanje stanj



Delo z izjemnimi dogodki

- Zaklepanje stanja (ang. interlock)
  - Ob podanem pogoju se akcije v stanju prekinejo
  - Prehod v novo stanje je mogoč
- Nadzorne napake
  - Ko pride do napake, prehod v naslednje stanje ni mogoč
  - Avtomat se ustavi

Diagram poteka: primerjava

Funkcijski bloki

- Zvezno vodenje, regulacija

Diagram poteka

- Koračno/sekvenčno vodenje, krmiljenje

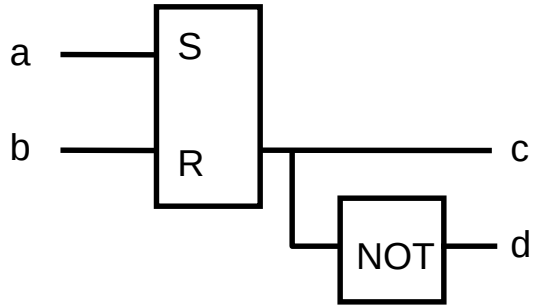
Velikokrat je najboljša izbira kombinacija obeh, zato mora biti komunikacija med njimi mogoča

Združevanje na nivoju funkcijskih blokov

Diagram poteka: primerjava

Primer

- Povezovanje funkcijskih blokov



- Diagram poteka

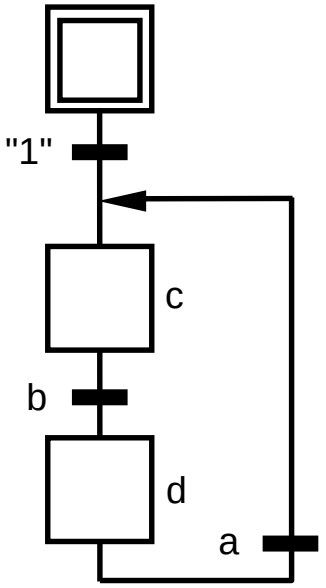
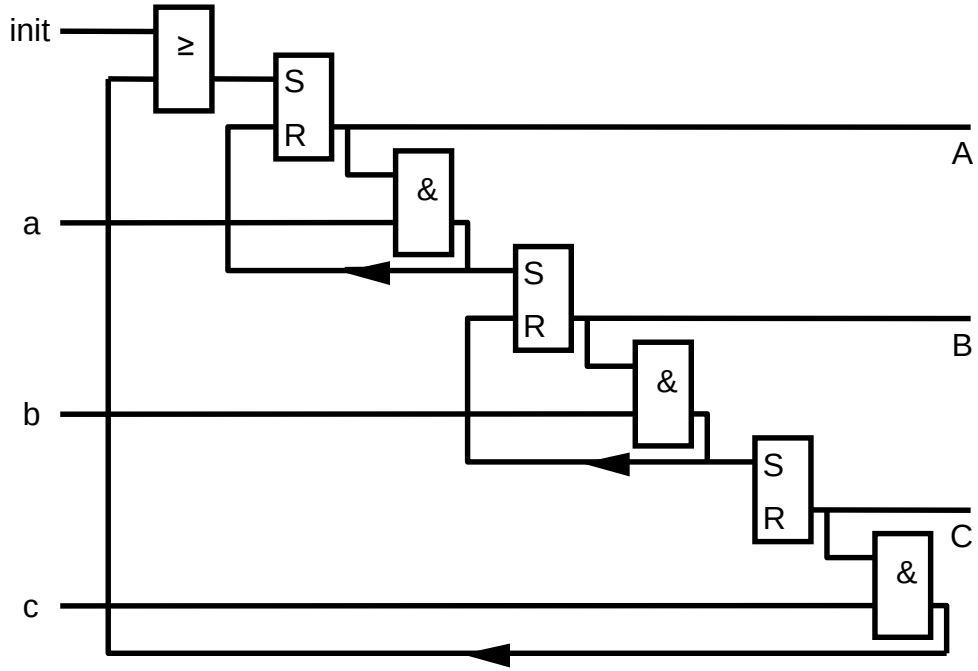


Diagram poteka: primerjava

Primer

- Povezovanje funkcijskih blokov



- Diagram poteka

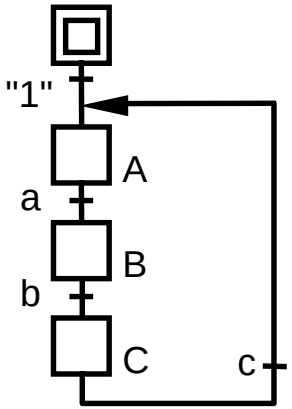


Diagram poteka: Siemens Graph

Vsak avtomat se v program vključi kot funkcijski blok

Pogoji za prehode

- Lestvični diagrami
- Funkcijski načrt

Stanja

- Komentar
- Akcije v stanju
  - Stalne
  - Vezane na dogodke

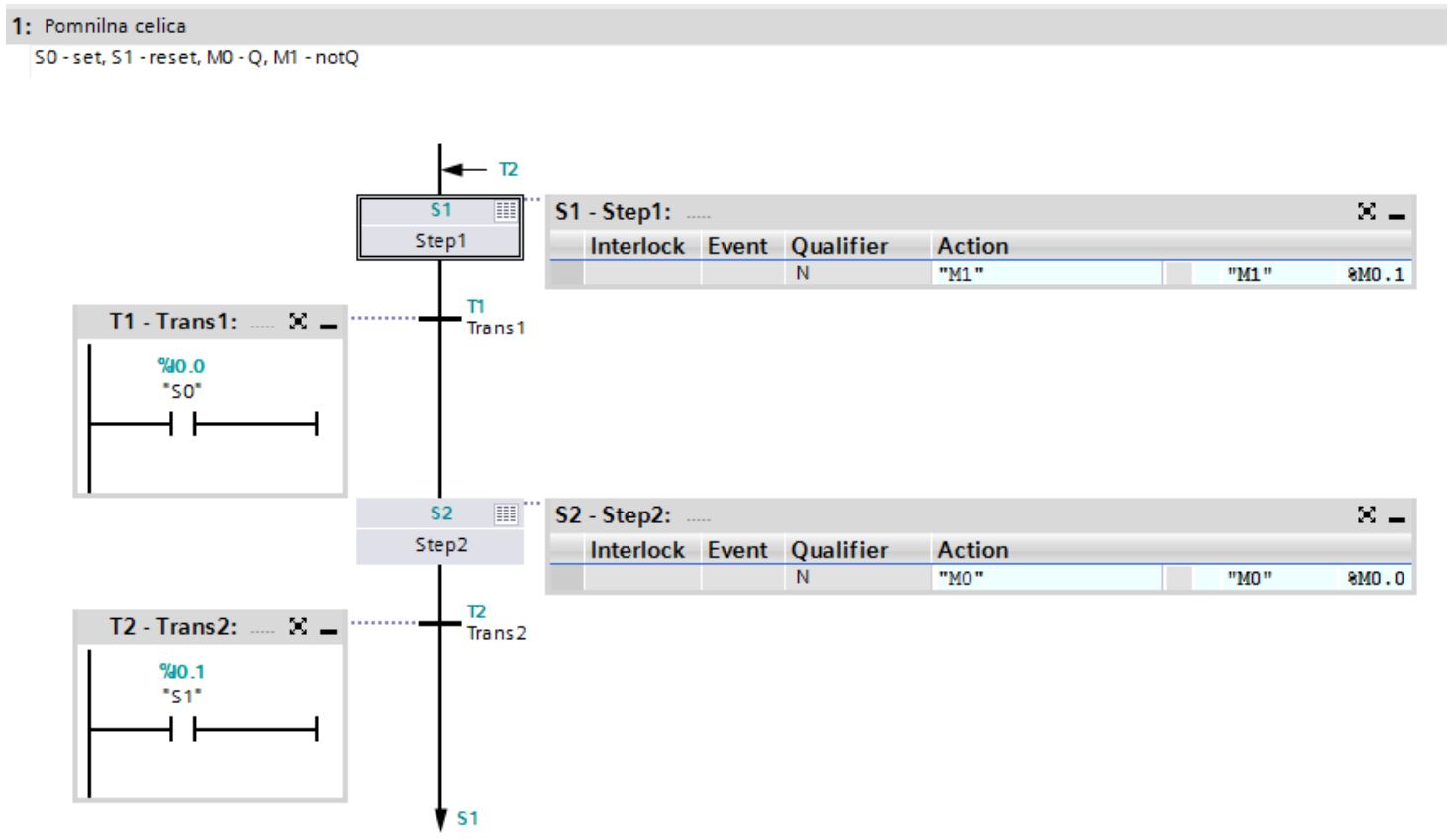


Diagram poteka: Siemens Graph

Dogodki

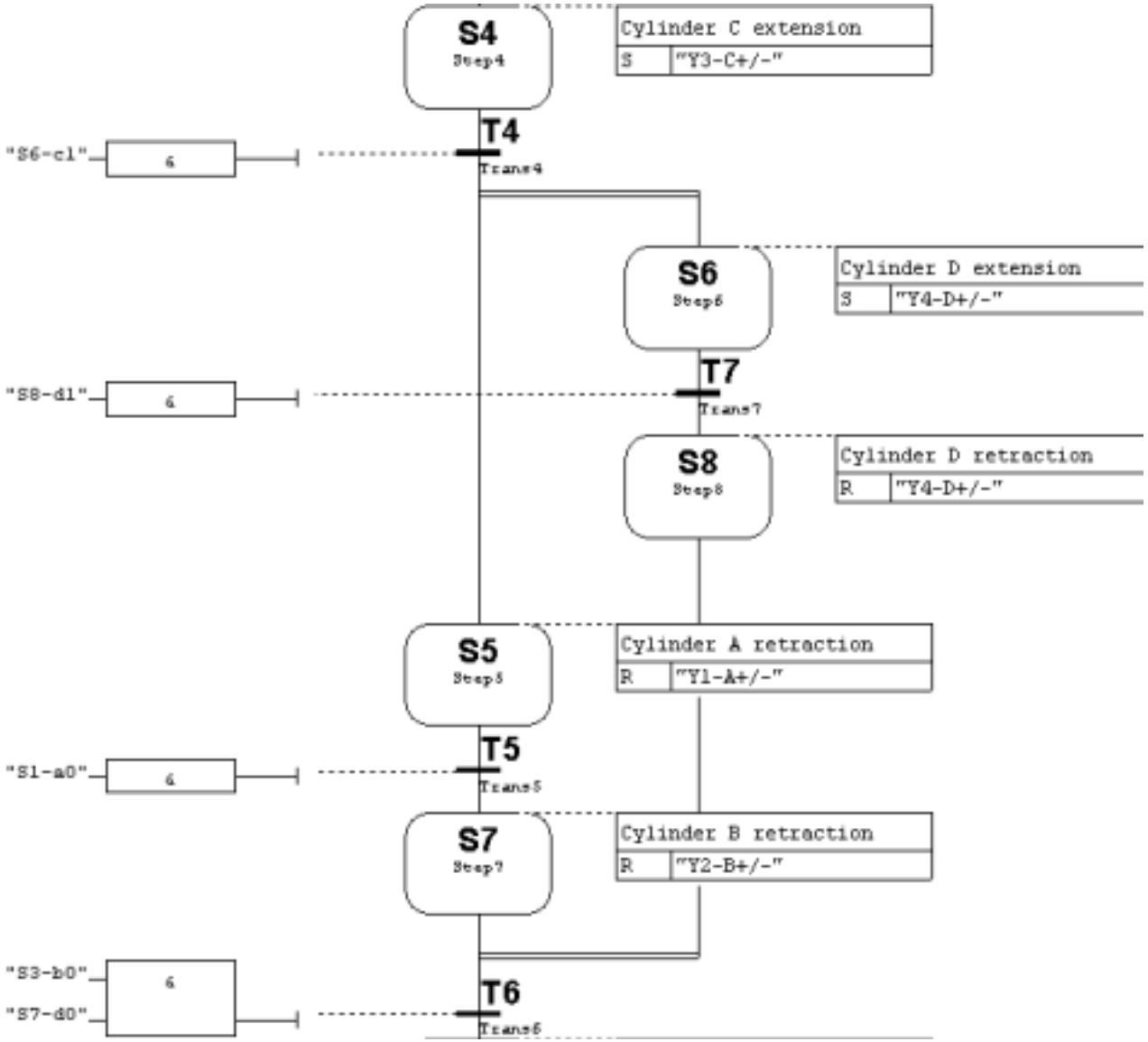
- S1/S0 – ob vstopu v stanje/izstopu iz stanja
- L1/L0 – pogoji za notranje zaklepanje stanja (ang. interlock) pri izstopu iz stanja/vstopu v stanje
  - Notranje zaklepanje – skupen pogoj za več spremenljivk v stanju
- V1/V0 – prišlo je do nadzorovane napake/napake ni več. Dokler je napaka aktivna, prehod v naslednje stanje ni mogoč
- …

Akcije v stanju

- N – vrednost 1, dokler je stanje aktivno
- S – ob prihodu v stanje se vrednost postavi
- R – ob prihodu v stanje se vrednost pobriše
- D – časovnik (on delay)
- L – pulz (pulse)
- CALL – klicanje FC ali FB
- itd.

Diagram poteka: Siemens Graph

Primer



Lista ukazov: standard

Nizkonivojski programski jezik

- Podoben zbirniku
- Neprijazen do uporabnika
  - Koda ni strukturirana
  - Šibka semantika
  - Odvisen od programirljivega krmilnika

Večinsko mnenje

- Osnovni jezik, ki naj bi ga podpirali programirljivi logični krmilniki, ki so združljivi z IEC
- Standard osnovnega jezika ne postavlja!

Namenjen izkušenim programerjem za

- Izdelavo učinkovite programske kode
- V ta jezik naj bi se prevedli vsi višjenivojski programski jeziki (tega standard ne zahteva)

Lista ukazov: standard

Vsak ukaz se začne v novi vrstici

Vsak ukaz je sestavljen iz

- Operacijske kode (operator / mnemonik)
- Operandov
  - Operandi so ločeni z vejicami

Oznake na začetku vrstice

- Zaključi se z dvopičjem

Komentar na koncu vrstice

Dovoljene so prazne vrstice

Oklepaji

Lista ukazov: standard

Osnovni ukazi

- 21 ukazov
- Rezultati operacij se shranjujejo v register RLO (Result of Logic Operation)
- Modifier
  - N – negacija rezultata
  - C – pogojna izvedba
  - (- zakasnitev rezultata

Nr.	Operator	Modifier	Operand	Definition
1	LD	N	Note 1	Sets the actual result of the operand
2	ST	N	Note 1	Stores the actual result in the operand address
3	S	Note 2	**BOOL**	Set Boolean operator to 1
	R	Note 2	BOOL	Reset Boolean operator to 0
4	AND	N, (	BOOL	Boolean AND
5	&	N, (	BOOL	Boolean AND
6	OR	N, (	BOOL	Boolean OR
7	XOR	N, (	BOOL	Boolean Exclusive-OR
8	ADD	(	Note 1	Addition
9	SUB	(	Note 1	Subtraction
10	MUL	(	Note. 1	Multiplication
11	DIV	(	Note 1	Division
12	GT	(	Note 1	Comparison: >
13	GE	(	Note 1	Comparison: >=
14	EQ	(	Note 1	Comparison: =
15	NE	(	Note 1	Comparison: <>
16	LE	(	Note 1	Comparison: <=
17	LT	(	Note 1	Comparison: <
18	JMP	C,N	**MARK**	Jump to the Mark
19	CAL	C,N	NAME	Call function block (Note 3)
20	RET	C,N		Return to a function or a function block
21	)			Processing reset operations

Note 1: The operations must be either loaded or given with a type.
The actual result and the operand must have the same type.
Note 2: The operations are only executed when the value of the actual result is a Boolean 1.
Note 3: A list of arguments in parenthesis follow the name of the function block

Lista ukazov: standard

Primeri

- AND %IX1 Rezultat := Rezultat AND %IX1

- AND(%IX1
  ORN %IX2
) Rezultat := Rezultat AND (%IX1 OR NOT %IX2)

- LD 15
  ST C10.PV
  LD %IX10
  ST C10.CU
  CAL C10 Klic funkcije: CAL C10(CU:=%IX10, PV:=15)

Lista ukazov: Siemens STL

STL (ang. Statement List)

Block title: Osnovne logične operacije

Comment

Network 1: Negacija

```
1    AN    #x1
2    =     #y_not
3
```

Network 2: Logični IN

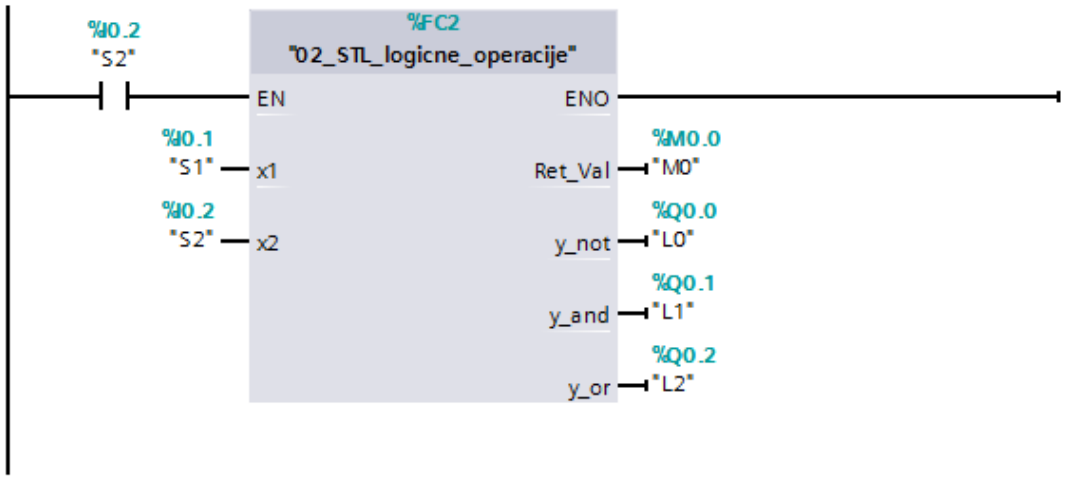
```
1    A     #x1
2    A     #x2
3    =     #y_and
```

Network 3: Logični ALI

```
1    O     #x1
2    O     #x2
3    =     #y_or
```

Network 1: Kličemo funkcijo "STL_logicne_operacije" ob pogoju, da je S2=1 (signal ENable)

```
1     A     "S0"                              %I0.0
2     =     #x1tmp
3     A     "S1"                              %I0.1
4     =     #x2tmp
5     A     "S2"                              %I0.2
6     JNB   SKIP
7     CALL  "02_STL_logicne_operacije"        %FC2
8        x1      :="S0"                       %I0.0
9        x2      :="S1"                       %I0.1
10       Ret_Val :="M0"                       %M0.0
11       y_not   :="L0"                       %Q0.0
12       y_and   :="L1"                       %Q0.1
13       y_or    :="L2"                       %Q0.2
14 SKIP: NOP 0
15
```



Lista ukazov: Siemens STL

Dva 32-bitna akumulatorja

- ACC1 in ACC2
- Delamo večinoma z ACC1
- Pred spremembo ACC1 se njegova stara vsebina prenese v ACC2

Dva naslovna registra AR1 in AR2

16-bitna statusna beseda (SW)

- /FC – First check: začetek novega logičnega izraza, ni povezave s prejšnjim RLO
- RLO – Result of Logic Operation: Oblikuje se po vsaki logični ali primerjalni operaciji
- STA – Status: Rezultat povpraševanja po vrednosti na posameznem bitu
- OR – Za realizacijo prednostnega izvajanje logične operacije AND pred OR
- OS – Overflow Stored: Preliv (aritmetične operacije), ohrani se do konca bloka
- OV – Overflow: Preliv, se briše pri prvi naslednji pravilno izvedeni aritmetični operaciji
- CC0 in CC1 – Condition Codes: Oblikujeta se glede na rezultat
  - 00 – nič, 01 – več od nič, 10 – manj od nič, 11 – neveljaven rezultat
- BR – Binary Result: zaključevanje vej v LAD in FBD programih

Strukturirano besedilo: standard

Jezik podoben Pascalu

Primeren za kompleksne obdelave podatkov

Prevajanje v listo ukazov

Spremenljivke, definirane v ST, lahko uporabljajo tudi drugi jeziki

Strukturirano besedilo: standard – izrazi

- Z izrazi določamo vrednosti na podlagi vrednosti spremenljivk in konstant
- Nujna je uporaba zahtevanih podatkovnih tipov
  - Pretvarjanje s funkcijami
  - Primer: REAL_TO_INT(..)
- Sestavljeni so iz operatorjev in operandov
  - Izračunavanje po prioriteti
  - V primeru enake prioritete od leve proti desni
  - Primer: X := (A+B-C)*ABS(D);

	Operacija	**Oznaka**	**Proriteta**
1	Oklepaj	(izraz)	Visoka
2	Function evaluation	ImeF(argumenti) LN(A), MAX(X,Y)	
3	Potenciranje	**	
4	Predznak	-	
5	Negacija	NOT	
6	Množenje	*	
7	Deljenje	/	
8	Modulo	MOD	
9	Seštevanje	+	
10	Odštevanje	-	
11	Primerjava	<,>,<=,>=	
12	Enakost	=	
13	Neenakost	<>	
14	IN	&	
15		AND	
16	Ekskluzivni ALI	XOR	
17	ALI	OR	Nizka

Strukturirano besedilo: standard – stavki

Stavek	Primer
Prireditev	`CV := CV + 1;` `C := SIN(X);`

Klic in uporaba funkcijskega bloka	`CMD_TIMER(IN := %IX5,`
	`  PT := T#300ms);`
	`A := CMD_TIMER.Q;`

Izhod iz funkcije	`RETURN;`

IF	`IF D < 0.0 THEN`
	`  NROOTS := 0;`
	`ELSIF D = 0.0 THEN`
	`  NROOTS := 1;`
	`ELSE`
	`  NROOTS := 2;`
	`END_IF;`

CASE	`TW := BCD_TO_INT(THUMBWHEEL);`
	`CASE TW OF`
	`  1:  DISPLAY := OVEN_TEMP;`
	`  2,3: DISPLAY := MOTOR_SPEED;`
	`ELSE`
	`  DISPLAY := 0;`
	`END_CASE;`

Stavek	Primer

FOR	`J := 101;`
	`FOR I := 1 TO 100 BY 2 DO`
	`  IF WORDS[I] = 'KEY' THEN`
	`  J := I;`
	`  EXIT;`
	`  END_IF;`
	`END_FOR;`

WHILE	`J := 1;`
	`WHILE J <= 100 DO`
	`  J := J + 2;`
	`END_WHILE;`

REPEAT	`J := -1;`
	`REPEAT`
	`  J := J + 2;`
	`UNTIL J = 101`
	`END_REPEAT;`

EXIT	`EXIT;`
Prazen stavek	`;`

Strukturirano besedilo: Siemens SCL

SCL = Structured Control Language

Osnovni podatkovni tipi:

Podatkovni tip	Pomen	Širina v bitih	Območje
BOOL	Bit	1	False, True
BYTE	Bajt	8	16#00 – 16#FF
WORD	Beseda	16	16#0000 – 16#FFFF
DWORD	Dvojna beseda	32	16#00000000 – 16#FFFFFFFF
INT	Celo število	16	-32768 – 32767
DINT	Dolgo celo število	32	-2147483648 – 2147483647
REAL	Realno število	32	$\approx \pm 3.4028235 \times 10^{38}$
CHAR	Znak	8	'A', 'B', … (znaki ASCII)
TIME	Čas v formatu IEC	32	T#-24d20h31m23s647ms – TIME#24d20h31m23s647ms
DATE	Datum	16	D#1990-01-01 – DATE#2168-12-31
TIME_OF_DAY	Čas v dnevu	32	TOD#00:00:00.000 – TIME_OF_DAY#23:59:59.999

Strukturirano besedilo: Siemens SCL

Sestavljeni podatkovni tipi

Podatkovni tip	Pomen	Širina v bitih	Območje (omejitve)
DATE_AND_TIME	Datum in čas	64	DT#1990-01-01-00:00:00.000 – DATE_AND_TIME#2168-12-31-23:59:59.999
STRING	Niz znakov	Spremenljiva	Največ 254 znakov (prvi znak definicija, drugi znak trenutno)
ARRAY	Polje	Spremenljiva	Največ šest dimenzij
STRUCT	Struktura	Spremenljiva	Zbirka komponent poljubnih podatkovnih tipov (UDT) Do šest nivojev struktur v strukturi

Ostali podatkovni tipi

Podatkovni tip	Pomen	Širina v bitih
TIMER	Časovnik	16
COUNTER	Števnik	16
FC, FB, DB, SDB	Programski bloki	16
POINTER	Kazalec na DB	48
ANY	Kazalec ANY	80

Strukturirano besedilo: Siemens SCL – primeri

RocnoAvto

Name	Data type	Offset	Default v

CASE... OF... | FOR... TO DO.. | WHILE.. DO... | (*...*)

```
1  // stavek if grdo
2  IF NOT "Avtomatsko" THEN
3      "Luc" := "Stikalo";
4  ELSE
5  END_IF;
6  IF "Avtomatsko" THEN
7      "Luc" := "Ura_1s";
8  END_IF;
9
10 // stavek if lepše
11 IF NOT "Avtomatsko" THEN
12     "Luc" := "Stikalo";
13 ELSE
14     "Luc" := "Ura_1s";
15 END_IF;
16
17 // brez if narobe
18 "Luc" := NOT "Avtomatsko" AND "Stikalo";
19 "Luc" := "Avtomatsko" AND "Ura_1s";
20
21 // brez if pravilno
22 "Luc" := (NOT "Avtomatsko" AND "Stikalo") OR
23           ("Avtomatsko" AND "Ura_1s");
```

AsimetricnaUra

Name	Data type
Input	
CasOn	Time
CasOff	Time
<Add new>	
Output	
Q	Bool
<Add new>	
InOut	
<Add new>	
Static	
Gor	TON
Dol	TON
Ura	Bool

CASE... OF... | FOR... TO DO.. | WHILE.. DO... | (*...*)

```
1  #Gor(IN:=#Ura,
2       PT:=#CasOn);
3  #Dol(IN:=NOT #Ura,
4       PT:=#CasOff);
5
6  IF #Gor.Q OR #Dol.Q THEN
7      #Ura := NOT #Ura;
8  END_IF;
9
10 #Q := #Ura;
```

Strukturirano besedilo: Siemens SCL – primeri

Blokada prehitre menjave smeri vrtenja motorja

Block_Motor

	Name	Data type	Offset	Default value	Visible in ...	Setpoint	Comment
	Input				☐	☐	
	i_levo	Bool	...	false	☑	☐	
	i_desno	Bool	...	false	☑	☐	
	t_value	Time	...	T#0ms	☑	☐	
	Output				☐	☐	
	o_vklop	Bool	...	false	☑	☐	
	o_smer	Bool	...	false	☑	☐	
	InOut				☐	☐	
	<Add new>				☐	☐	
	Static				☐	☐	
	t_levo	TOF	...		☑	☐	
	t_desno	TOF	...		☑	☐	
	o_levo	Bool	...	false	☑	☐	
	o_desno	Bool	...	false	☑	☐	

CASE... OF... | FOR... TO DO... | WHILE... DO... | (*...*)

```
(*  levo: smer = true
    input: i_levo (bool), i_desno (bool), t_value (bool)
    output: o_vklop (bool), o_smer (bool)
    static: t_levo (TOF), t_desno (TOF), o_levo (bool), o_desno (bool)
*)

#t_levo(IN := #o_levo, PT := #t_value);
#t_desno(IN := #o_desno, PT := #t_value);

#o_levo := #i_levo AND NOT #t_desno.Q;
#o_desno := #i_desno AND NOT #t_levo.Q;

#o_vklop := #o_levo OR #o_desno;

(* o_smer := o_levo je OK, če želimo ohranjat rele --> IF *)
IF #o_levo THEN
    #o_smer := true;
ELSIF #o_desno THEN
    #o_smer := false;
ELSE
    #o_smer := #o_smer;
END_IF;
```

Tehnike programiranja: uvedba stanj

Razlogi

- Določeni deli programa se lahko izvajajo samo ob določenih pogojih
- Potreba po zaklepanju klinov oz. delov programske kode

Razdelitev programa na logična stanja

- Stanja in prehodi med njimi morajo biti jasno določeni tako v ročnem kot avtomatskem načinu
  - Stanja določena glede na akcije izvršnih sistemov in vrednosti merilnih sistemov
- Lažje programiranje kompleksnih sistemov
- Lažji obratni inženiring
  - Koda za vsako stanje enostavnejša
  - Pogoji za prehajanje med stanji so veliko bolj očitni
  - Vsak programer piše na svoj način

Prednosti

- Skrajšanje zagona sistema zaradi napak v programu za 85 %
  - Predvsem na račun enostavnejših pogojev za zaklepanje klinov
  - V tipičnem lestvičnem diagram je velik del kode namenjenih zaklepanju klinov
    - 35 % pri procesni kontroli (zvezni procesi, regulacija)
    - 60 % pri sekvenčnem procesu

Tehnike programiranja: uvedba stanj

Programiranje

- Posnemanje diagrama poteka
- Ob izpolnjenem pogoju za prehod v novo stanje se:
  - aktivira ustrezna oznaka (žeton) za novo stanje (set)
  - deaktivira oznaka za trenutno stanje (reset)
    - Če je lahko hkrati aktivnih več stanj, je potrebno paziti, da se pri prehodu v novo skupno stanje deaktivirajo vsa trenutna stanja
- Označevanje stanj
    - Z biti: en bit ustreza enemu stanju (ang. one hot encoded)
    - Številčno: uporaba spremenljivke tipa integer (in primerjalnika)
- Ob zagonu sistema je potrebna logika, ki zna
  - ugotoviti, v katerem stanju se je sistem ustavil
    - preskakovanje stanj in ne njihovo zaporedno izvajanje, ki je lahko zelo nevarno!
  - postaviti sistem v začetno stanje ali
  - preprečiti njegovo delovanje, če ni v pravem stanju, in to ustrezno alarmirati (najlažje)

Tehnike programiranja: spremljanje materiala

Izdelava strukture (UDT – user data type), ki predstavlja logično sliko materiala

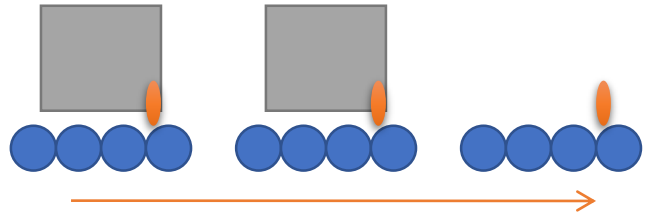
- Podatki o materialu: črtna koda (ID), fizične mere, neustreznost, …
- Ciljna lokacija
- Navodila za obdelavo (recept)
- Funkcije na trenutni lokaciji: zasedenost , premikanje, …

Sekvenčni proces

- Vsaka fizična enota (mesto) ima tudi svojo logično sliko

- Vsaka enota na enkrat lahko izpolnjuje le eno nalogo
  - Primer: trije tekoči trakovi
  - Enostavna komunikacija:
    zahteva, dovoljenje, akcija, potrditev, (alarm)

- Prepisovanje struktur
  - Ob izpolnjenih pogojih se celotna slika prenese iz enega mesta na drugega
    - Material je fizično na novem mestu
    - Senzorska slika ustreza bodoči logični sliki (dvojna kontrola)
  - Prepis naj NE bo vezan na fronto fotocelice



- Alarmiranje v primeru, da se logična in fizična shema po določenem času ne ujemata

Tehnike programiranja: organizacija programa

Branje senzorjev v strukture
- Umerjanje senzorjev, skaliranje analognih vrednosti, pretvorbe (NC→NO)

Proženje alarmov
- En alarm, en bit: postavitev ustreznega bita

Upravljanje alarmov
- Potrjevanje in ničenje alarmov

Priprava podatkov za vmesnike človek-stroj
- Pretvorbe (NC→NO), izračuni, ločene podatkovne strukture (DB) zaradi boljše preglednosti

Glavni program
- Avtomat prehajanja stanj

Sledenje materiala
- Glede na fizično sliko

Varnostne funkcije
- Varovanje človeka in opreme
- Zaradi varnosti neodvisne od glavnega programa
- Blokada prehitre menjave smeri vrtenja

Aktivacija izvršnih sistemov

Tehnike programiranja: Fischer technik

Linija z dvema napravama, Pnevmatski sistem:

- DB z vsemi lokacijami kjer je tipalo (fotocelica ali končno stikalo); "navidezne" lokacije (potiskač, vrtljiva miza, vhod na trak)
- Avtomat za vsako lokacijo ali izvršni člen
- Odvisnosti (predhodno, naslednje mesto)
- Vrtljiva miza (vodenje položaja, materiala na mizah)
- Spremljanje materiala
  - Črtna koda (ID)
  - Naloga/recept
  - uporaba Watch Table za vnos vrednosti na prvi lokaciji

Robot

- Hierarhija avtomatov:
  - Avtomat za vsako os (vrtenje, dvig, izteg, stisk)
  - Avtomat, ki povezuje vse osi
    - Pojdi na lokacijo
    - Pojdi na lokacijo in poberi
    - Pojdi na lokacijo in odloži
  - Avtomat, ki izvaja "program" premikov
- Spremljanje materiala