3. Programirljivi logični krmilniki

- Najpogostejši način izvedbe logičnega in sekvenčnega vodenja v industriji

  PLC - Programmable Logic Controllers

  SPS - Speicherprogrammierbare Steuerungen
- Definicija

  Programirljivi logični krmilnik je digitalno delujoča elektronska naprava, ki na podlagi ukazov, shranjenih v programirljivem pomnilniku, izvaja logične, sekvenčne, časovne in aritmetične operacije ter s tem vodi različne naprave in procese preko binarnih in analognih vhodov in izhodov

Izvedbe logičnega in sekvenčnega vodenja

- Krmilnik oziroma krmilje
  - sprejema binarne vhodne vrednosti
  - postavlja vrednosti binarnih izhodov
    (če se omejimo na sisteme brez analognih vhodno/izhodnih signalov)
- Različne izvedbe
  - relejska vezja
  - digitalna elektronska vezja
  - mikro-krmilniki
  - programirljivi logični krmilniki (PLK)

Relejska krmilja

- Starejša krmilja so bila izvedena z elektro-mehanskimi relejskimi vezji
  - enostavna realizacija Boolove preklopne logike
  - relativno zanesljivo delovanje
  - robustnost, neobčutljivost na motnje
  - enostavna montaža
  - enostavna diagnostika in vzdrževanje
  - načrtovanje s prilagojenimi električnimi shemami v obliki lestvičnih diagramov

Relejska krmilja /2

- Glavne slabosti
  - velika poraba prostora
  - kompleksno in drago ožičenje pri zahtevnejših krmiljih
  - težavno in zamudno vnašanje sprememb in popravkov v logično vezje
  - zamudna dokumentacija
  - težavna realizacija števnih in časovnih funkcij
  - relativno kratka življenjska doba mehanskih delov

Elektronska krmilja

- Digitalna elektronska logika
  - prednost -> miniaturizacija
  - logično vezje je načrtano za vsako krmilje posebej
  - težavno vnašanje sprememb in popravkov
  - v primerjavi z releji večja občutljivost na razmere v industrijskem okolju
  - uvajanje programirljivih funkcij
    - nastavitve števcev, časovnikov ipd.
    - elektronski programatorji
  - nadaljnji razvoj
    - -> programirljiva logična vezja (PLD): PAL, GAL, CPLD, FPGA
    - -> mikro-krmilniki
    - -> mikroprocesorski PLK

Mikroprocesorski PLK

- Posebna oblika procesnih računalnikov
  - prirejeni za logično in sekvenčno vodenje
  - enostavna vgradnja in priključitev signalov
  - enostavni za programiranje
  - programske operacije namesto relejskih kontaktov -> večja zanesljivost
  - programirne naprave (PC) avtomatsko generirajo dokumentacijo
  - hitro in enostavno vnašanje sprememb
  - razširjen nabor funkcij, npr. kompleksne aritmetične operacije
  - možnost komunikacije s drugimi napravami

Izbira tipa krmilja

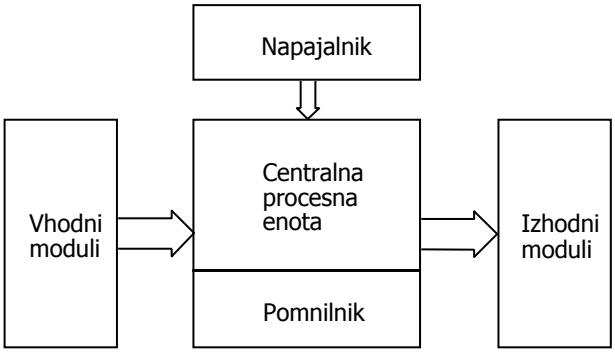
- Možnosti: releji, digitalna logična vezja, mikro-krmilniki, PLK , splošnonamenski računalniki
- Kriteriji
  - cena glede na funkcijo
  - dimenzije
  - hitrost delovanja
  - občutljivost na električne motnje
  - zahtevnost izvedbe (načrtovanja, instalacija, programiranje)
  - možnost izvajanja kompleksnih operacij
  - enostavnost spreminjanja funkcij
  - enostavnost vzdrževanja

3.1 Zgradba in lastnosti PLK

- Mikroračunalniki, prirejeni za delovanje v industrijskem okolju
  - mehanski tresljaji
  - električne motnje
  - temperaturne spremembe
  - vlažnost
  - korozivna atmosfera
- Dve izvedbi
  - kompaktna
  - modularna

Kompaktna in modularna izvedba PLK

Funkcionalne enote PLK



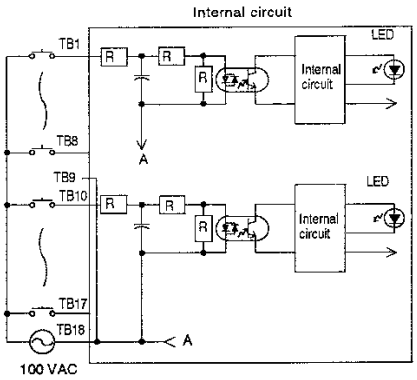
3.1.1 Centralna procesna enota

- En ali več mikroprocesorjev
  - lahko poseben enobitni procesor za izvajanje logičnih operacij
- 'watch dog' vezje
- Ura realnega časa
- Pomnilnik
  - sistemski del (ROM, EPROM)
  - uporabniški del (RAM, EPROM, EEPROM)
  - baterijska podpora

3.1.2 Vhodno/izhodni moduli

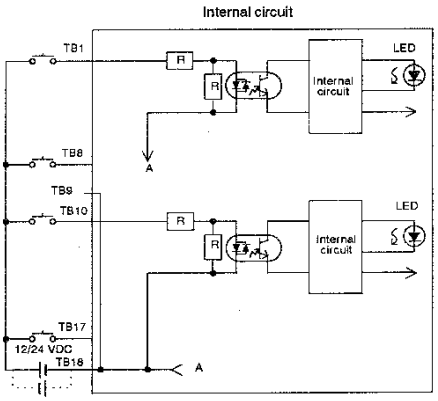
- Enostavni krmilniki - le binarni vhodi in izhodi
- Zmogljivejši krmilniki - širok izbor različnih modulov
  - binarni vhodi in izhodi
  - analogni vhodi in izhodi
  - komunikacijski moduli
  - posebni funkcijski moduli
- Napetostna ločitev, filtriranje, zaščita pred napetostnimi sunki

Binarni vhodi



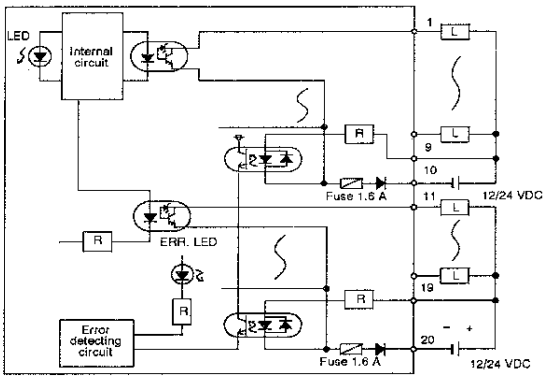
- Tipično 16 do 64 vhodnih signalov (točk)
- Napetostna ločitev, filtriranje odvečnih preklopov (odzivni čas cca. 20 ms pri AC, oz. 10 ms pri DC)
- Vhodne napetosti
  - 110/240 V AC
  - 5, 12, 24 V DC

Binarni vhodi /2



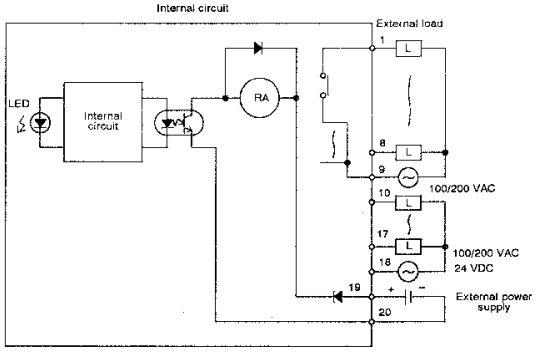
- Skupna masa v skupinah po 8 ali 16 vh. točk
- Vh. tok nekaj mA
- Vh. upornost > 1 kΩ
- Pogosto ne smejo biti vsi vhodi hkrati 1
- Zunanje napajanje senzorjev oz. vh. stikal
- Indikacija vklopa z LED

Binarni izhodi



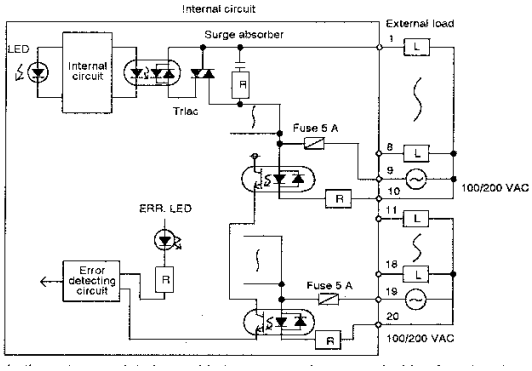
- Tipično 8 do 64 izhodnih točk
- Osnovni tipi
  - tranzistorski
  - relejski
  - triak
- Napetostna ločitev
- Absorbcija konic
- Zunanje napajanje bremen, pri relejskih še dodatno napajanje relejev

Binarni izhodi /2



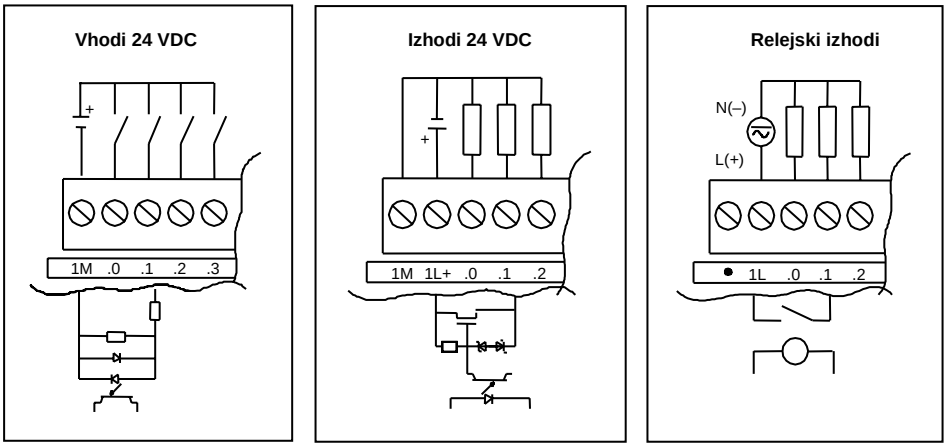
- Odzivni čas, izhodni tok in napetost odvisni od tipa
- Tranzistorski
  - odz. čas 2 ms
  - tok 0.1-2 A na točko oz. 0.8-4 A na grupo
  - napetost 12, 24 V DC
- Relejski
  - odz. čas 10 ms
  - tok tipično 2 A na točko, 8 A na grupo
  - napetost 24 V DC do 240 V AC

Binarni izhodi /3



- Triak
  - odz. čas vklopa: 1 polperioda + 1 ms (t.i. Zero Crossing Triac)
  - odz. čas izklopa 1 ms
  - max. tok tipično 0.5 A
  - omejen tudi min. tok npr. 10 mA pri 110 V, 20 mA pri 240 V
  - napetost 110 do 240 VAC

Vezava binarnih vhodov in izhodov



Analogni vhodi

- Tipično 2 do 16 analognih signalov
- Nastavljiva vhodna območja med -10 in +10 V oz. med 0 in 20 mA
- Napetostna ločitev
- Posebne izvedbe za priključitev standardnih senzorjev (Pt100, termočleni)
- Ločljivost 12 do 14 bitov
- Vhodna upornost > 100 kΩ (za napetostni signal) oz. 250 Ω za tokovni signal

Analogni izhodi

- Tipično 2 do 16 analognih signalov
- Nastavljiva vhodna območja med -10 in +10 V oz. med 0 in 20 mA
- Napetostna ločitev
- Dopustna obremenitev < 30 mA pri napetostnih izhodih oz. < 12 V pri tokovnih
- Zaščiteni pred kratkim stikom

3.1.3 Druge vrste modulov

- Komunikacijski moduli
- Posebni funkcijski moduli
  - hitri števni moduli
  - pozicijski moduli
  - regulacijski moduli
  - moduli z mehko logiko (fuzzy)
  - moduli za podporo umetnega vida
  - moduli za branje črtne kode
  - radio-frekvenčni moduli
  - moduli za komunikacijo z operaterjem
    - priključitev tipkovnice, zaslona, tiskalnika
    - nekatere izvedbe vsebujejo cel PC

Komunikacijski moduli

- Komunikacija z
  - drugimi krmilniki
  - senzorji in izvršnimi členi
  - nadrejenimi računalniki
- Najpogostejši vmesniki
  - RS 232C, RS 485, Ethernet, specializirani vmesniki
- Standardni protokoli
  - področna vodila (Foundation Fieldbus, Profibus, P-net, WorldFIP, ControlNet, Interbus, SwiftNet, Fieldbus Foundation’s high speed Ethernet; CAN, LonWorks ipd.)
  - TCP/IP

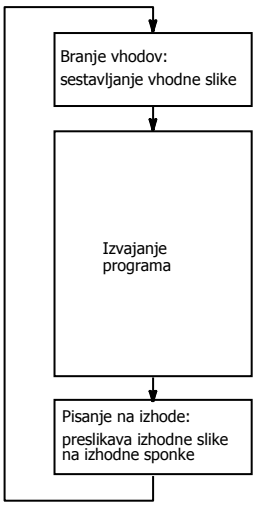
Posebni funkcijski moduli

- Hitri števni moduli
  - za štetje pulzov, ki so prehitri za priključitev na standardne binarne vhode; priključitev optičnih senzorjev za štetje, rotacijskih in translacijskih enkoderjev ipd.
- Pozicijski moduli
  - poleg števnega vhoda še regulator in analogni izhod; pozicioniranje servo-pogonov ipd.
- Regulacijski moduli
  - analogni vhodi, izhodi
  - samostojno izvajanje PID-algoritma

Posebni funkcijski moduli /2

- Moduli z mehko logiko (fuzzy)
  - prenos znanja operaterja v krmilnik
- Moduli za podporo umetnega vida
  - kontrola delovnih postopkov
  - testiranje produktov
- Moduli za branje črtne kode
  - avtomatska identifikacija sestavnih delov
  - razvrščanje
- Radio-frekvenčni moduli
  - avtomatska identifikacija delov
  - sledenje produktom

3.1.4 Način delovanja PLK



- Branje vhodov
  - stanje vhodov se preslika v pomnilnik
  - »vhodna slika«
- Izvajanje programa
  - izračun programiranih logičnih izrazov
  - rezultati se zapisujejo v »izhodno sliko«
- Pisanje na izhode
  - izhodi se postavijo v skladu z vsebino pomnilnika

3.1.5 Zmogljivostni razredi PLK

- Mikro PLK
  - do 32 vh./izh.
  - avtomatizacija manjših strojev in naprav, binarni ali univerzalni binarni/analogni vhodi in izhodi, običajno brez potencialne ločitve
  - kompaktne izvedbe, tipično do 8 kB pomnilnika
  - vgrajen prikazovalnik in nekaj tipk
- Mali PLK
  - do 128 vh./izh.
  - nadomestilo relejske logike, pogosto le binarni vhodi in izhodi
  - kompaktne izvedbe, tipično do 32 kB pomnilnika

Zmogljivostni razredi PLK /2

- Srednji PLK
  - do 1024 vh./izh.
  - tako sekvenčno vodenje kot enostavnejše regulacije
  - velika hitrost, do 128 kB pomnilnika
  - dobre komunikacijske zmožnosti
- Veliki PLK
  - več tisoč vh./izh.
  - kompleksne operacije,
  - do več MB, tudi več procesorjev

Primeri

3.2 Programiranje logičnih krmilnikov

- Tehnike načrtovanja logičnega in sekvenčnega vodenja so povezane z načinom izvedbe
- Načrtovanje relejskih vezij:
  - prilagojena oblika električnih vezalnih shem
- Načrtovanje krmilij v obliki logičnih vezij:
  - metode sinteze digitalnih vezij
- Načrtovanje vodenja s PLK - programiranje:
  - kombinacija uveljavljenih pristopov
  - dodatni programski jeziki

Programski jeziki za programiranje logičnih krmilnikov

- Osnova - programski jezik, podoben zbirniku
- Ukazi za branje vhodov, logične operacije, pisanje na izhode
- Programirne naprave
  - specializirane naprave
  - osebni računalniki
- Poleg tekstovnih tudi grafični programski jeziki
  - preklopne funkcije vnašamo preko ekvivalentnih logičnih vezij ali električnih vezalnih shem
  - programirna naprava prevede program v obliko, primerno za izvajanje

Grafični programski jeziki

- Lestvični diagram (LD)
  - pregledna predstavitev vezja relejskih krmilij
  - simbolična predstavitev konkretnih elementov krmilnega vezja
  - kot program PLK: zamenjava fizičnih simbolov z simboli programskih elementov
- Funkcijski blokovni diagram (FBD)
  - krmilje predstavimo kot digitalno vezje
  - realizacija logičnih vrat s programom
  - dodatne funkcije

Grafični programski jeziki /2

- Skupni elementi
  - povezave:
    - predstavljajo pretok električne moči - LD
    - predstavljajo pretok signalov - FBD
  - smer
    - pretok moči le od leve proti desni
    - smer signalov od izhoda bloka (desno) k vhodu naslednjega bloka (levo)
  - gradniki:
    - kontakti, tuljave, funkcijski bloki in funkcije

Problematika programiranja logičnih krmilnikov

- Večina PLK podpira programiranje v več programskih jezikih
- Nekaj uveljavljenih jezikov, vendar v več različicah
- Razlike med PLK različnih proizvajalcev
- Težavno programiranje različnih PLK
- Težaven prenos programov med različnimi tipi
- Potreba po poenotenju
  - mednarodni standard

Primer razlik v programiranju

Zahteva: Izhod št. 0 naj se vključi, če je prižgan vhod št. 1 ali vhod št. 2, pri tem pa ne sme biti prižgan vhod št. 0. Če pogoj za vključitev ni izpolnjen, mora biti izhod izključen (v stanju 0)

SIEMENS:

```
A(
O     I0.1
O     I0.2
)
AN    I0.0
=     Q0.0
```

MITSUBISHI:

```
LD    X1
OR    X2
ANI   X0
OUT   Y10
```

IEC IL:

```
LD    %IX1
OR    %IX2
ANDN  %IX0
ST    %QX0
```

3.2.1 Mednarodni standard IEC 61131

- IEC - International Electrotechnical Commission
- Standard IEC 61131 združuje številna določila v zvezi s PLK
- Namen: poenotenje delovanja, uporabe, programiranja PLK in pripadajočega dokumentiranja
- Prenosljivost programov
- Povezljivost v komunikacijska omrežja

Deli standarda IEC 61131 (1-2)

- IEC 61131-1 (sprejet 1992, 2. verzija 2003)
  - splošni del, terminologija, definicije pojmov, ki se uporabljajo v ostalih delih standarda
  - splošne informacije o lastnostih in uporabi PLK
- IEC 61131-2 (sprejet 1992, 2. verzija 2003)
  - definira zahteve za strojno opremo
  - električne, mehanske in funkcionalne zahteve
  - pogoji servisiranja, skladiščenja in transporta
  - postopki za preverjanje skladnosti s standardom

Deli standarda IEC 61131 (3-5)

- IEC 61131-3 (sprejet 1993, 2. verzija 2003)
  - programiranje, struktura programa, programski jeziki, sintaksa in semantika
- IEC 61131-4 (sprejet 1995, v pripravi 2. verzija)
  - uporabniške smernice, analiza in specifikacija zahtev, izbira, izvedba in vzdrževanje sistemov s PLK
- IEC 61131-5 (sprejet 2000)
  - komunikacije, definira komunikacijske funkcijske bloke, ki jih lahko uporabljamo pri programiranju PLK

Dodatni deli standarda

- IEC 61131-6 (opuščen)
  - komunikacija med PLK preko področnih vodil
- IEC 61131-7 (sprejet 2000)
  - programiranje PLK z uporabo mehke logike
- IEC 61131-8 (sprejet 2000, 2. verzija 2003)
  - smernice za uporabo in implementacijo programskih jezikov, ki jih definira 3. del standarda

IEC 61131-3: mednarodni standard za programiranje PLK

- Dva vsebinska sklopa
  - skupna programska osnova
  - standardizirani programski jeziki
- Skupna programska osnova
  - struktura programa
    - konfiguracija, viri in opravila
    - programske organizacijske enote oz. programski moduli (Program Organization Units - POU)
      - programi, funkcijski bloki, funkcije
  - podatkovni tipi in spremenljivke
  - standardne funkcije in funkcijski bloki
  - vsi elementi, ki so skupni vsem programom, ne glede na uporabljen programski jezik

IEC 61131-3: mednarodni standard za programiranje PLK /2

- Standardizirani programski jeziki
  - štirje osnovni programski jeziki
    - dva tekstovna jezika:
      IL **seznam ukazov (Instruction List)**
      ST **strukturiran tekst (Structured Text)**
    - dva grafična jezika:
      LD **lestvični diagram (Ladder Diagram)**
      FBD **funkcijski blokovni diagram (Function Block Diagram)**
  - dodaten programski jezik, namenjen strukturiranju opravil in programskih modulov:
    SFC **sekvenčni funkcijski diagram (Sequential Function Chart)**

3.2.2 Standardizirani programski jeziki

- Seznam ukazov (IL)
  - osnovni programski jezik, primer:

```
LD    %IX1
OR    %IX2
ANDN  %IX0
ST    %QX0
```

  - omogoča uporabo simboličnih imen in komentarjev:

```
        LD    R1          (* Nalozi vrednost R1 *)
        JMPC  RESET       (* Skoci, ce 'True' *)
        LD    PRESS_1
        ST    MAX_PRESS
RESET:  LD    0
        ST    A_X43
```

Standardizirani programski jeziki /2

- Strukturiran tekst (ST)
  - višjenivojski programski jezik
  - sintaksa podobna Pascalu
  - primer:

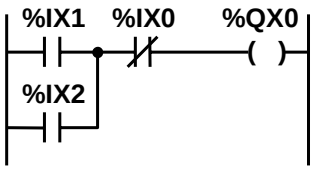
```
%QX0 := (%IX1 OR %IX2) AND NOT %IX0;
```

  - deklaracija spremenljivk, npr: **VAR Stevilo: INT;**
  - sestavljeni stavki

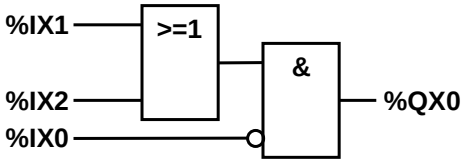
```
IF ... THEN ... ELSE ... END_IF;
FOR ... DO ... END_FOR;
WHILE ... DO ... END_WHILE;
REPEAT ... UNTIL ... END_REPEAT;
```

Standardizirani programski jeziki /3

- Lestvični diagram (LD)
  - izhaja iz relejske tehnike
  - primer:

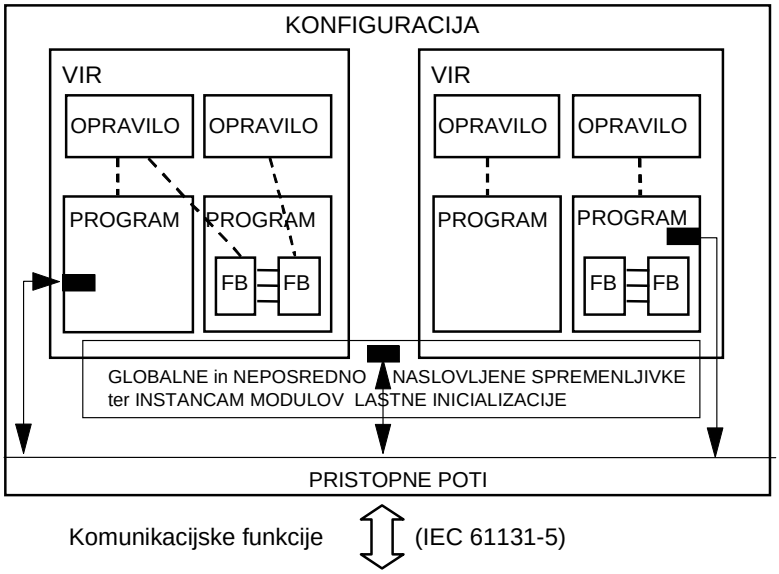


- Funkcijski blokovni diagram (FBD)
  - izhaja iz simbolov digitalne tehnike
  - primer:



- Sekvenčni funkcijski diagram (SFC)
  - predstavlja stanja sistema in prehode med njimi
  - običajno se uporablja v povezavi z ostalimi jeziki

3.2.3 Skupna programska osnova



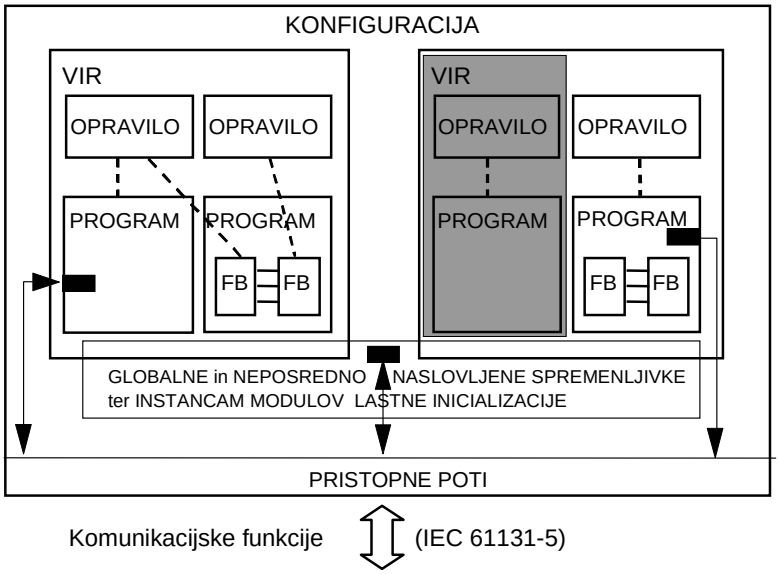
Elementi programskega modela

- Konfiguracija
  - celotna programska oprema sistema PLK, ki je potrebna za rešitev konkretnega problema vodenja
  - pogled z najvišjega nivoja
- Vir
  - znotraj konfiguracije lahko definiramo več virov
  - vir je lahko ena procesna enota v sistemu
- Opravilo
  - vir vsebuje eno ali več opravil
  - opravila nadzorujejo izvajanje različnih delov programa

Elementi programskega modela /2

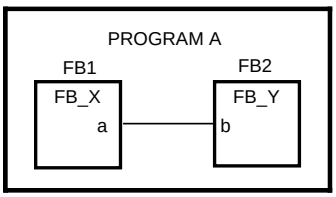
- Programi
  - napisani v različnih programskih jezikih
- Funkcije in funkcijski bloki
  - osnovni gradniki programov
  - vsebujejo podatkovne strukture in algoritme
- Programi lahko izmenjujejo podatke
  - preko globalnih in neposredno naslovljenih spremenljivk
  - preko komunikacijskih funkcij

Primerjava z običajnim PLK

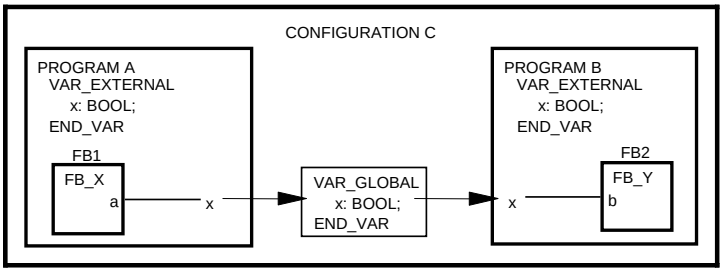


Komunikacijski model

- Podatkovna povezava znotraj programa

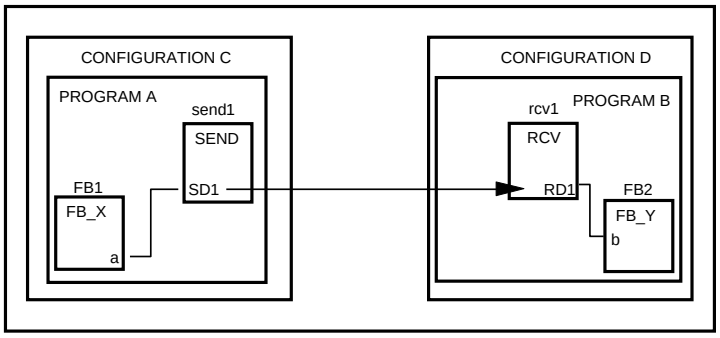


- Povezava preko globalnih spremenljivk



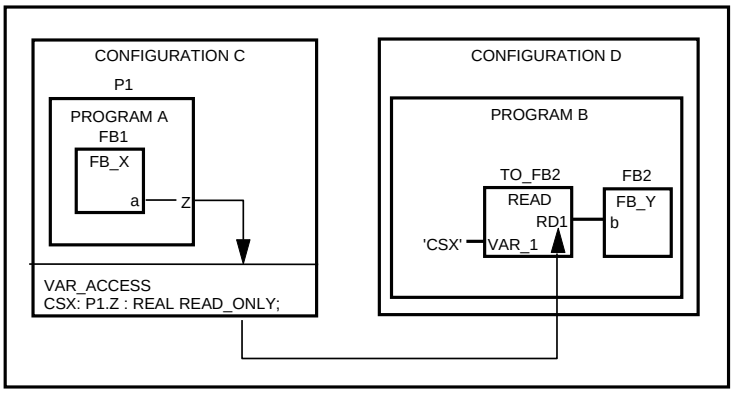
Komunikacijski model /2

- Komunikacijski funkcijski bloki



Komunikacijski model /3

- Komunikacija preko pristopnih poti



Predstavitev vrednosti

- Zapis števil
  - 12, -23, 123_456, 0.456, -1.23E-12
  - 2#1101_0111, 16#E5,
  - 0, 1, FALSE, TRUE
- Zapis znakovnih nizov
  - posebni znaki: ‘$E8’, ‘$L’
- Zapis časa
  - t#14ms, T#12.3s, t#5d12h15m18s3.5ms
  - TOD#15:36:15.12, D#1999-03-08
  - DT#1999-12-31-23:59:59.99

Podatkovni tipi

- Pomen podatkovnih tipov je predvsem v izogibanju oziroma zgodnjem odkrivanju programerskih napak
- Definicija tipa omogoča ločevanje med spremenljivkami, ki imajo enak bitni zapis, a predstavljajo različne podatke npr. znakovni niz, datum, celo število ter 16 bitov digitalnih vhodov
- Jasnejši pomen spremenljivk - pomembno predvsem pri skupinskem deli več ljudi, ki spremenljivke poimenujejo na različne načine

Osnovni podatkovni tipi

- BOOL, BYTE, INTEGER, REAL
- DATE, TIME_OF_DAY, STRING
- preprečujejo napake kot npr. deljenje datuma s celim številom ipd.
- poleg tega lahko definiramo lastne tipe, ki jih lahko nato uporabljamo podobno kot osnovne tipe

Pregled osnovnih tipov

Št.	Oznaka	Podatkovni tip	Bitov	Območje
1	**BOOL**	Boolean	1	0,1; FALSE, TRUE
2	**SINT**	Short integer	8	-128 ÷ +128
3	**INT**	Integer	16	-32768 ÷ +32767
4	**DINT**	Double integer	32	-2147483648 ÷ +2147483647
5	**LINT**	Long integer	64	$-2 \cdot 10^{63} \div +2 \cdot 10^{63}-1$
6	**USINT**	Unsigned short integer	8	0 ÷ 255
7	**UINT**	Unsigned integer	16	0 ÷ 65535
8	**UDINT**	Unsigned double integer	32	0 ÷ 4294967295
9	**ULINT**	Unsigned long integer	64	$0 \div 2 \cdot 10^{64}-1$
10	**REAL**	Real numbers	32	$\pm 10^{\pm 38}$
11	**LREAL**	Long reals	64	$\pm 10^{\pm 308}$

Pregled osnovnih tipov /2

Št.	Oznaka	Podatkovni tip	Bitov	Območje
12	**TIME**	Duration	*	*
13	**DATE**	Date (only)	*	*
14	**TIME_OF_DAY** or **TOD**	Time of day (only)	*	*
15	**DATE_AND_TIME** or **DT**	Date and time of Day	*	*
16	**STRING**	Variable-length single-byte character string	*	-
17	**BYTE**	Bit string of length 8	8	-
18	**WORD**	Bit string of length 16	16	-
19	**DWORD**	Bit string of length 32	32	-
20	**LWORD**	Bit string of length 64	64	-
21	**WSTRING**	Variable-length double-byte character string	*	-
* število bitov oziroma območje vrednosti je odvisno od implementacije				

Deklaracija novih tipov

Št.	Značilnost / tekstovni primer
1	Direktna izpeljava iz osnovnih tipov, npr.: TYPE RU_REAL : REAL ; END_TYPE
2	Naštevni podatkovni tipi, npr.: TYPE ANALOG_SIGNAL_TYPE : (SINGLE_ENDED, DIFFERENTIAL) ; END_TYPE
3	Intervalni podatkovni tipi, npr.: TYPE ANALOG_DATA : INT (-4095..4095) ; END_TYPE
4	Polja podatkov, npr.: TYPE ANALOG_16_INPUT_DATA : ARRAY [1..16] OF ANALOG_DATA ; END_TYPE
5	Strukturirani podatkovni tipi, npr.: TYPE ANALOG_CHANNEL_CONFIGURATION : STRUCT RANGE : ANALOG_SIGNAL_RANGE ; MIN_SCALE : ANALOG_DATA ; MAX_SCALE : ANALOG_DATA ; END_STRUCT ; ANALOG_16_INPUT_CONFIGURATION : STRUCT SIGNAL_TYPE : ANALOG_SIGNAL_TYPE ; FILTER_PARAMETER : SINT (0..99) ; CHANNEL : ARRAY [1..16] OF ANALOG_CHANNEL_CONFIGURATION ; END_STRUCT ; END_TYPE

Spremenljivke

- Simbolična predstavitev preko oznak
  - niz črk, števil in podčrtajev
  - prvi znak mora biti črka ali podčrtaj
  - ni ločitve med malimi in velikimi črkami (razen v komentarjih in znakovnih nizih)
  - v imenih ne sme biti presledkov
- Vh/izh preslikava je definirana na nivoju konfiguracije, vira ali programa
- Ločitev programa in aparaturne opreme

Enostavne spremenljivke

- Podatkovni element določenega tipa
- Simbolične
  - predstavljene z oznakami
- Neposredno naslovljene spremenljivke
  - vsebina pomnilne celice v vhodnem, izhodnem ali notranjem pomnilnem prostoru PLK
  - naslov: znak "%", koda lokacije, koda velikosti, eno ali več celih števil ločenih s pikami
  - npr.: %QX75, %IW215, %MD48, %IX3.5
  - interpretacija naslova je odv. od konkretnega PLK

Kode lokacije in velikosti

Št.	Koda	Pomen	Privzet tip
1	I	Input location	
2	Q	Output location	
3	M	Memory location	
4	X	Single bit size	BOOL
5	Brez	Single bit size	BOOL
6	B	Byte (8 bits) size	BYTE
7	W	Word (16 bits) size	WORD
8	D	Double word (32 bits) size	DWORD
9	L	Long (quad) word (64 bits) size	LWORD

Sestavljene spremenljivke

- Polje (array)
  - več elementov istega tipa
  - naslavljamo jih z enim ali več indeksov
  - primer: TEST[2,15]
- Struktura (structure)
  - zbirka elementov različnih tipov
  - naslavljamo jih z imenom spremenljivke, piko in imenom elementa
  - primer: AN_VHODI.OBMOCJE

Inicializacija spremenljivk

- Izvede se ob startu vira ali konfiguracije
- Možne začetne vrednosti
  - vrednost, ki je je imela spremenljivka ob zaustavitvi (če je deklarirana kot trajna)
  - začetna vrednost, ki jo je definiral uporabnik
  - privzeta začetna vrednost podatkovnega tipa
- Različni tipi zagona
  - "topli" zagon - trajne spr. ohranijo vrednost
  - "hladni" zagon - vse sprem. se reinicializirajo
- Inicializacija vhodnih spr. je odvisna od izvedbe

Deklaracija spremenljivk

Koda	Pomen
VAR	Notranja (lokalna) spremenljivka
VAR_INPUT	Zunanja spremenljivka, v tem programskem modulu je ne moremo sprem.
VAR_OUTPUT	Izhodna spremenljivka programskega modula
VAR_IN_OUT	Zunanja spremenljivka, v tem programskem modulu jo lahko spreminjamo
VAR_EXTERNAL	Zunanja spremenljivka, ki pripada konfiguraciji preko **VAR_GLOBAL;** v tem programskem modulu jo lahko spreminjamo
VAR_GLOBAL	Deklaracija globalne spremenljivke
VAR_ACCESS	Deklaracija pristopne poti
VAR_TEMP	Začasna spremenljivka
VAR_CONFIG	Dinamično dodeljevanje naslovov in začetnih vrednosti
RETAIN	Trajna spremenljivka
NON_RETAIN	Spremenljivka, ki ni trajna
CONSTANT	Konstanta (vrednosti ne moremo spremeniti)
AT	Prireditev naslova

3.2.4 Programski moduli

- Program organization units (POU)
- Tri vrste
  - programi
  - funkcijski bloki
  - funkcije
- Določene programske module pripravi proizvajalec v obliki knjižnic, preostale sprogramira uporabnik
- Rekurzivna uporaba ni dovoljena

Funkcije

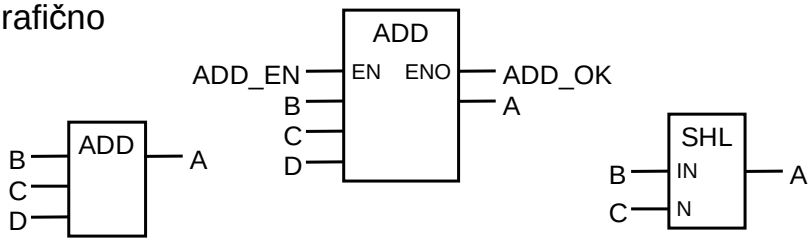
- Rezultat funkcije je vrednost ene spremenljivke (ki pa je lahko sestavljena)
- V tekstovnem jeziku lahko funkcijo uporabljamo kot operand v izrazih
- Funkcija nima notranjih stanj - klic funkcije z enakimi vrednostmi parametrov da vedno enak rezultat
- Poljubno obstoječo funkcijo lahko uporabimo pri deklaraciji novega programskega modula (funkcije, funkcijskega bloka ali programa)

Funkcije /2

- Tekstovna predstavitev
- Grafična predstavitev
  - blok v obliki kvadrata ali pravokotnika
  - smer izvajanja od leve proti desni (vhodni parametri so na levi, izhodni na desni)
- Kontrola izvajanja (vhod EN, izhod ENO)
- Standardne funkcije
  - pretvorba tipov, numerične in aritmetične funkcije
  - operacije nad nizi bitov, znakov
  - logične, primerjalne in preklopne funkcije
  - operacije nad časovnimi spremenljivkami

Klic funkcije

- Grafično



- Tekstovno

ST:
```
A := ADD(B,C,D);
A := A + B + C;

A := SHL(IN := B, N := C);
```

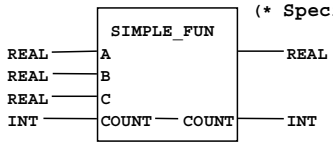
IL:
```
SHL(
IN := B
N := C
)
ST      A
```

Deklaracija funkcije

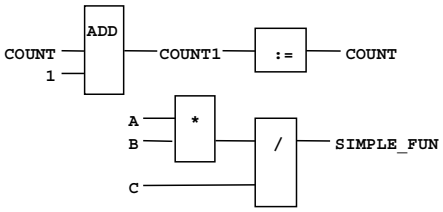
```
(* a) Tekstovna deklaracija v jeziku ST *)

FUNCTION SIMPLE_FUN : REAL
  VAR_INPUT
    A,B : REAL ;        (* Specifikacija uporabniskega vmesnika *)
      C : REAL := 1.0;
  END_VAR
  VAR_IN_OUT COUNT : INT ; END_VAR
  VAR COUNT1 : INT ; END_VAR

  COUNT1 := ADD(COUNT,1);  (* Specifikacija delovanja funkcije *)
  COUNT := COUNT1 ;
     SIMPLE_FUN := A*B/C;
  END_FUNCTION
```

Deklaracija funkcije /2

```
(* b) Graficna deklaracija v jeziku FBD *)

  FUNCTION
                                    (* Specifikacija uporabn. vmesnika *)
```




```
(* Specifikacija delovanja funkcije *)
```




```
  END_FUNCTION
```

Funkcijski bloki

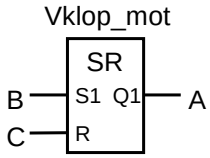
- Funkcijski blok vrne kot rezultat eno ali več vrednosti
- Lahko kreiramo več kopij (instanc) enega funkcijskega bloka
- Vsaka instanca ima svojo oznako in lastno podatkovno strukturo z notranjimi in izhodnimi spremenljivkami bloka
- Vrednost notranjih spremenljivk se med klici bloka ohranja - klic z istimi parametri da lahko različne rezultate

Funkcijski bloki /2

- Notranje spremenljivke bloka uporabniku bloka niso dostopne
- Poljuben obstoječ funkcijski blok lahko uporabimo pri deklaraciji novega programskega modula (funkcijskega bloka ali programa)
- Tekstovna predstavitev
- Grafična predstavitev - podobno kot pri funkcijah, dodatno še ime instance nad blokom
- Deklaracija

Klic funkcijskega bloka

- Grafično



- Tekstovno

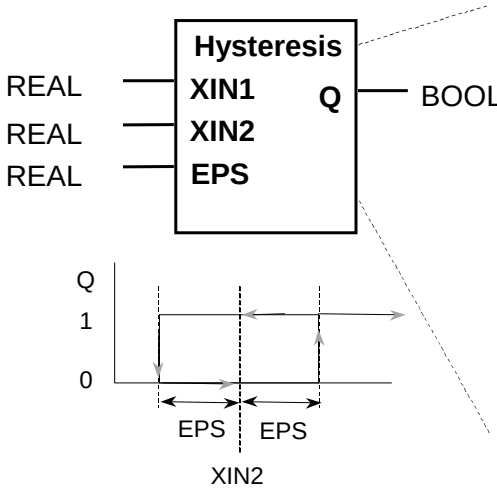
VAR Vklop_mot: SR; (* Neodvisno od jezika *)
END_VAR

.
.
.

ST: **Vklop_mot(S1 := B; R := C);**
A := Vklop_mot.Q1;

IL: **CAL Vklop_mot(**
S1 := B
R := C
)
LD Vklop_mot.Q1
ST A

Deklaracija funkcijskega bloka

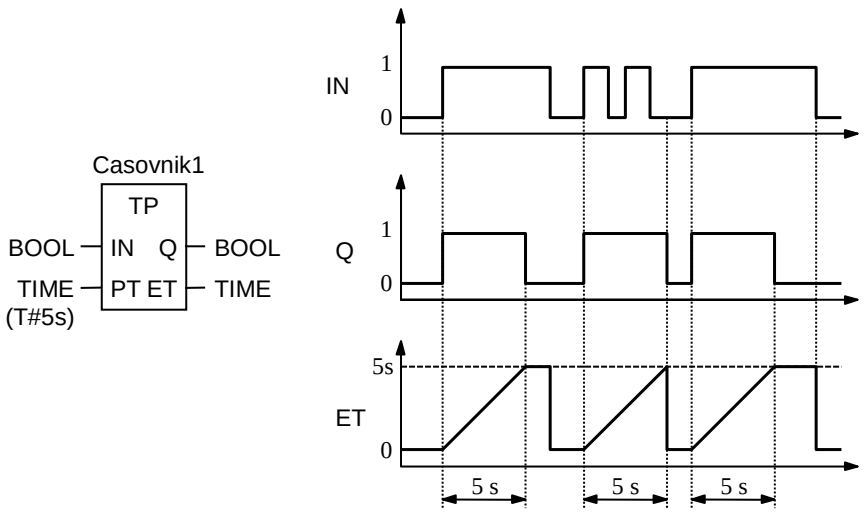



```
FUNCTION_BLOCK HYSTERESIS
 VAR_INPUT
   XIN1, XIN2 : REAL;
   EPS : REAL; (* Histereza *)
 END_VAR
 VAR_OUTPUT
   Q : BOOL := 0
 END_VAR
 IF Q THEN
   IF XIN1 < (XIN2-EPS) THEN
     Q := 0 (* XIN1 upada *)
   END_IF;
 ELSIF XIN1 > (XIN2 + EPS ) THEN
   Q := 1; (* XIN1 narasca *)
 END_IF;
END_FUNCTION_BLOCK
```

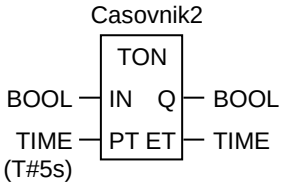
Funkcijski bloki /3

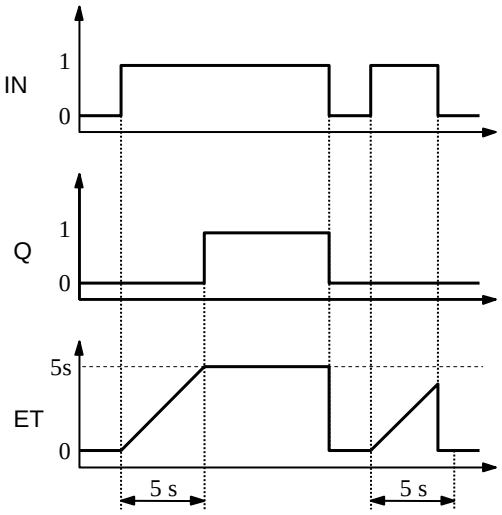
- Standardni funkcijski bloki
  - bistabilni elementi: SR, RS, semafor
  - detekcija preklopov 0->1, 1->0
  - števci: gor, dol, gor-dol
  - časovniki: **TP, TON, TOF** <- ***
- Komunikacijski funkcijski bloki
  - definirani v IEC 61131-5
  - osnovne funkcije: preverjanje delovanja naprav, zajem podatkov s pozivanjem, programiran zajem podatkov, upravljanje in zaščita povezav itd.

Časovniki

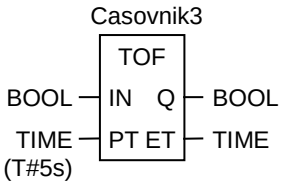


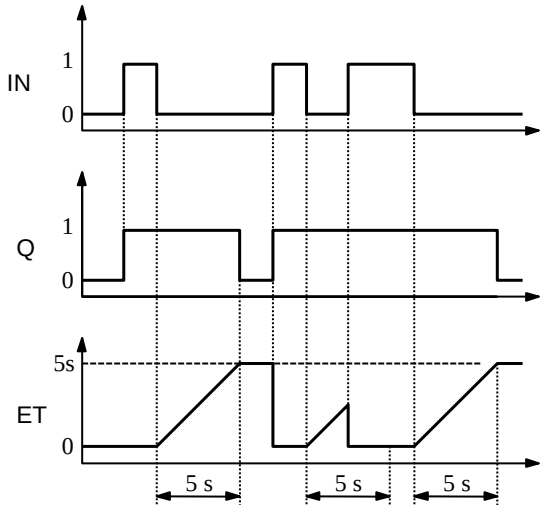
Časovniki /2





Časovniki /3





Programi

- Program je skupek elementov in sestavov programskega jezika, ki opravlja neko funkcijo vodenja
- Deklaracija in uporaba programov je podobna funkcijskim blokom, razlike:
  - znotraj programa so lahko definirane pristopne poti do spremenljivk programa
  - lahko deklariramo globalne in neposredno naslovljene spremenljivke
  - programi se lahko pojavljajo le znotraj vira, ne pa tudi znotraj funkcijskih blokov

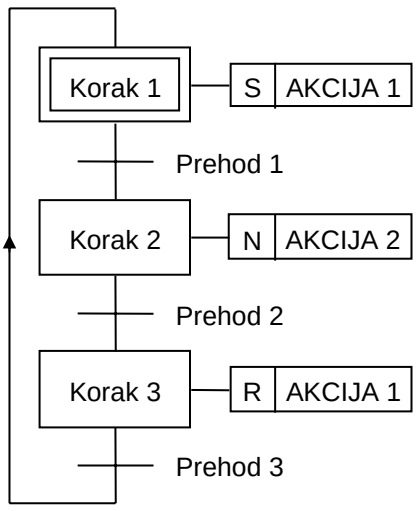
Strukturiranje

- Z uporabo funkcij in funkcijskih blokov lahko organiziramo program PLK kot vezje teh osnovnih gradnikov
- Na ta način lahko kompleksen program razgradimo na več enostavnih blokov
- Razgradnjo lahko nadaljujemo v več stopnjah
- Pri logičnem in sekvenčnem vodenju je pogosto zahtevano zaporedje operacij
- Dodaten programski jezik za strukturiranje takšnih zaporedij - SFC

3.2.5 Sekvenčni funkcijski diagram

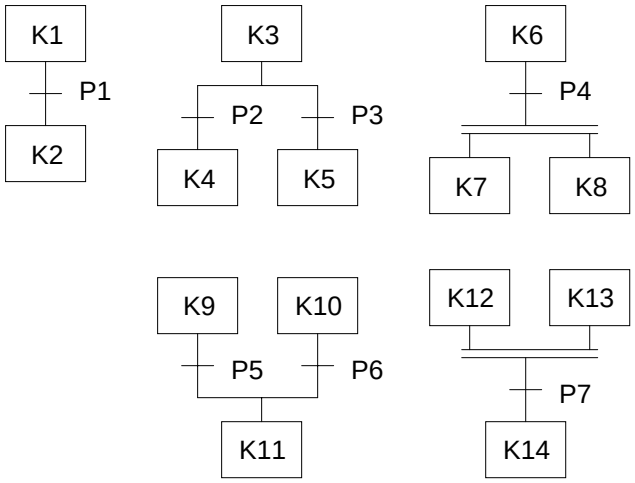
- Definicija se naslanja na standard IEC 60848
  - standard za specifikacijo in dokumentiranje postopkov v sistemih vodenja
  - izhaja iz francoskega Grafcet-a
  - osnova je teorija Petrijevih mrež
- Namen
  - strukturiranje programskih modulov, napisanih v različnih jezikih, za potrebe sekvenčnega vodenja
- Osnovni elementi
  - koraki, prehodi, akcije, prehodni pogoji

Sekvenčni funkcijski diagram /2



- Koraki so povezani s stanji sistema
- Začetni korak
- Korakom so pridružene akcije
- Koraku vedno sledi prehod
- Prehodom so pridruženi prehodni pogoji

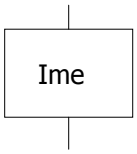
Povezave v diagramu



Koraki

- Korak je lahko aktiven (active) ali neaktiven (idle)
- Aktivnih korakov je lahko več, v danem trenutku aktivni koraki predstavljajo stanje programskega modula
- Grafična predstavitev
- Tekstovna predstavitev

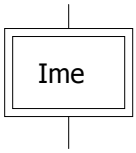
```
STEP Ime :
  (* telo koraka *)
END_STEP
```



Koraki /2

- Začetni korak
  - aktiven ob začetku izvajanja programskega modula
  - vsak diagram ima lahko le en začetni korak

```
INITIAL_STEP Ime :
  (* telo koraka *)
END_STEP
```



- Zastavica koraka
  - dvojiška spremenljivka Ime.X, ki ima vrednost 1, če je korak aktiven in 0, če je neaktiven
  - stanje zastavice je dostopno kot izhod na desni

Koraki /3

- Čas koraka
  - koraku pripada spremenljivka Ime.T tipa TIME, ki pove, koliko časa je korak že aktiven
  - ko korak preneha biti aktiven, spremenljivka zadrži vrednost do naslednjega aktiviranja
- Spremenljivki Ime.X in Ime.T sta lokalni
- Zastavica in čas koraka sta dodatni zahtevi, ki nista obvezni za združljivost s standardom
- Povezave vstopajo v korak navpično od zgoraj in izstopajo navpično navzdol

Prehodi

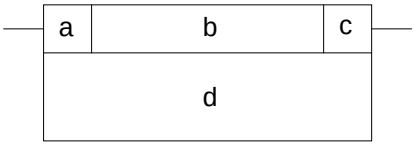
- Uravnavajo prenehanje aktivnosti koraka pred prehodom in pričetek aktivnosti koraka za prehodom - prehajanje stanj
- Vsakemu prehodu je pridružen prehodni pogoj ali dovzetnost
- Pogoj je lahko napisan v poljubnem jeziku standarda, rezultat mora biti dvojiška vrednost
- Standard dopušča več načinov povezovanja prehoda s pogojem

Akcije

- Koraku je lahko pridružena ena, nobena ali več akcij
- Korak brez akcije predstavlja čakanje na izpolnitev prehodnega pogoja
- Akcija je lahko
  - dvojiška spremenljivka
  - del programa v poljubnem jeziku - podprogram, ki se izvaja glede na aktivnost koraka
  - sekvenčni funkcijski diagram
- Standard dopušča več načinov deklaracije akcij

Akcije /2

- Grafična predstavitev



a - kvalifikator (določa tip akcije)
b - ime akcije
c - spremenljivka za signalizacijo
d - akcija v enem od jezikov standarda

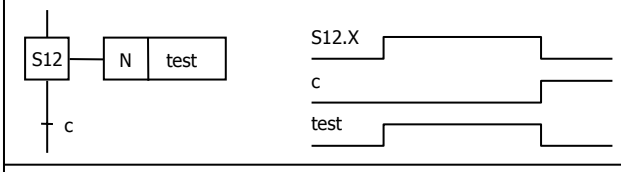
- Kvalifikatorji
  - tip akcije pove, kako je izvajanje akcije povezano z aktivnostjo pripadajočega koraka
  - nekateri tipi akcij potrebujejo poleg kvalifikatorja še dodatni parameter tipa TIME

Tipi akcij

Št.	Kvalifikator	Razlaga
1	None	Non-stored (null qualifier)
2	**N**	**N**on-stored
3	**R**	overriding **R**eset
4	**S**	**S**et (**S**tored)
5	**L**	time **L**imited
6	**D**	time **D**elayed
7	**P**	**P**ulse
8	**SD**	**S**tored and time **D**elayed
9	**DS**	**D**elayed and **S**tored
10	**SL**	**S**tored and time **L**imited
11	**P1**	**P**ulse (rising edge)
12	**P0**	**P**ulse (falling edge)

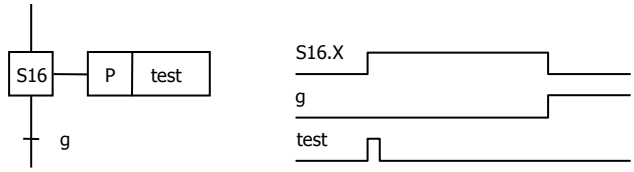
Trajanje akcij

a)



Spremenljivka 'test' je postavljena v visoko stanje, če je korak S12 aktiven

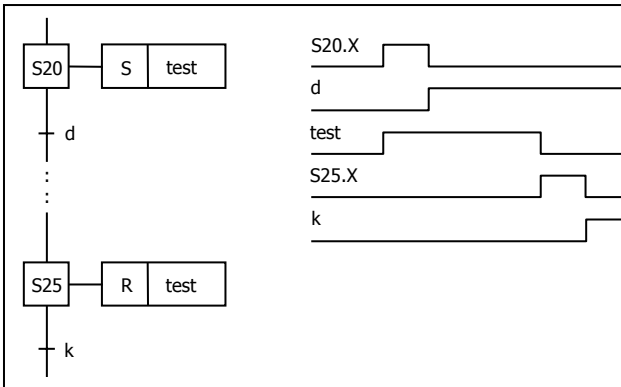
b)



Ko postane korak S16 aktiven, se spremenljivka 'test' postavi v visoko stanje za en cikel

Trajanje akcij /2

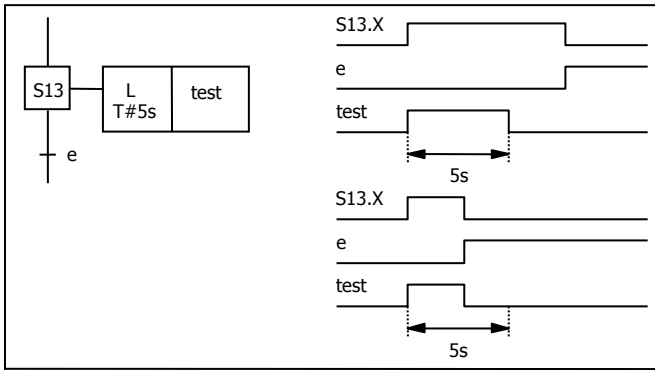
c)



Ko postane korak S12 aktiven, se spremenljivka 'test' postavi v visoko stanje in tam ostane, ne glede na trajanje aktivnosti koraka. Resetirati jo je možno le z akcijo tipa R.

Trajanje akcij /3

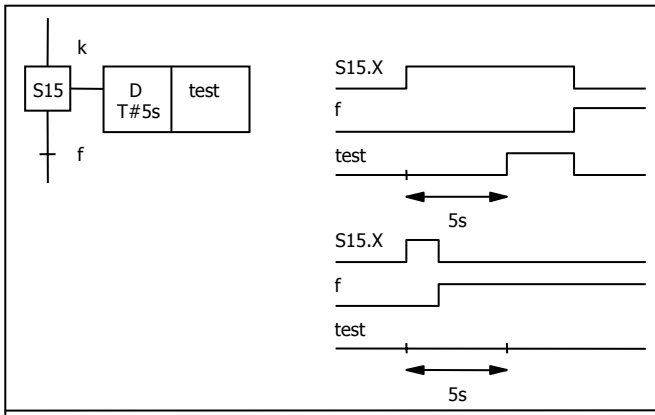
d)



Ko postane korak S13 aktiven, se spremenljivka 'test' postavi v visoko stanje in tam ostane predpisan čas, vendar kvečjemu do tedaj, ko se korak deaktivira.

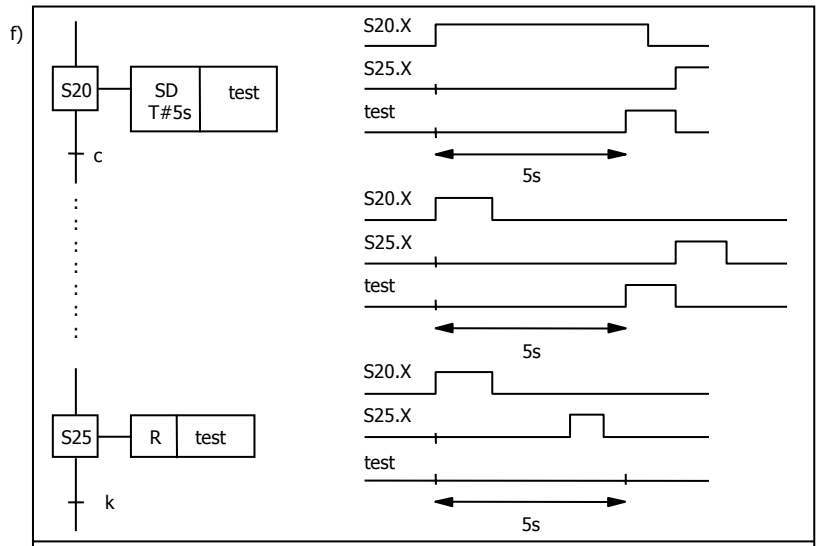
Trajanje akcij /4

e)



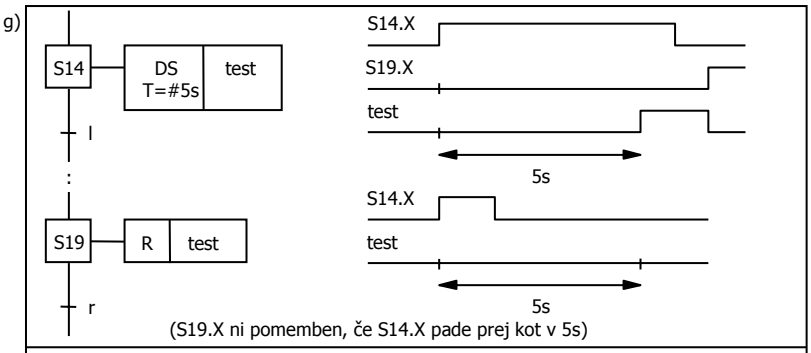
Spremenljivka preide v visoko stanje predpisan čas po aktiviranju koraka in ostane v visokem stanju, dokler je korak aktiven.

Trajanje akcij /5



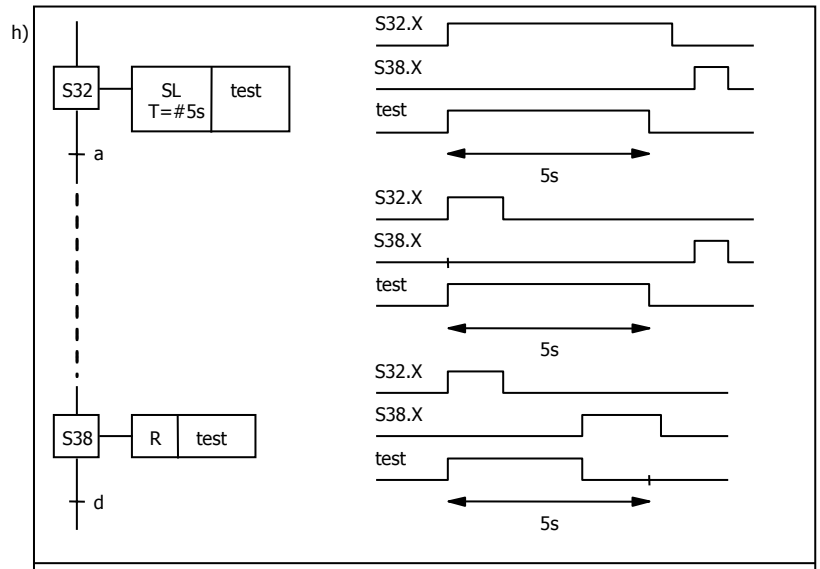
Spremenljivka se postavi v visoko stanje predpisan čas po aktiviranju koraka S20. To se zgodi, tudi če vmes korak S20 preneha biti aktiven. Spremenljivka ohrani stanje, dokler je ne resetira akcija tipa R.

Trajanje akcij /6



Deluje podobno kot akcija tipa SD, le da mora korak ostati aktiven do izteka zakasnitve, sicer spremenljivka ne bo prešla v visoko stanje. Ko pa se enkrat spremenljivka postavi, jo je možno resetirati le z akcijo tipa R.

Trajanje akcij /7

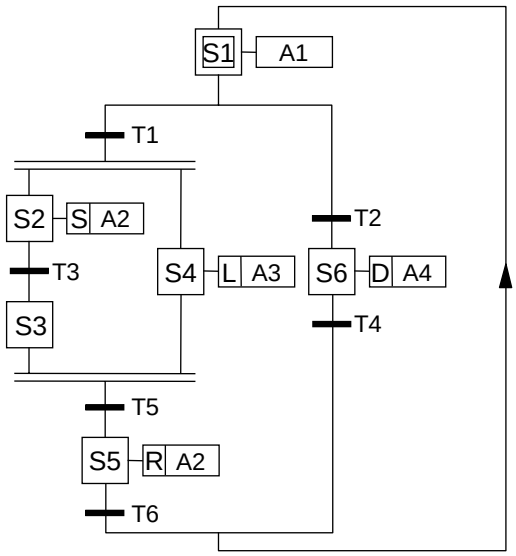


Spremenljivka preide v visoko stanje ob aktiviranju koraka in ostane v tem stanju predpisan čas, ne glede na čas trajanja koraka. Predčasno jo lahko resetiramo z akcijo tipa R.

Trajanje akcij /8

- V splošnem je akcija več kot le postavljanje spremenljivke, običajno je podprogram
- Podprogram se izvaja takrat, ko bi bila akciji prirejena spremenljivka v visokem stanju
  - po 'koncu izvajanja', se podprogram v vsakem primeru izvede ŠE ENKRAT!
  - pomembno za resetiranje časovnikov ipd.
  - vsaka, tudi pulzna akcija, se torej izvede vsaj dvakrat!

Primer diagrama



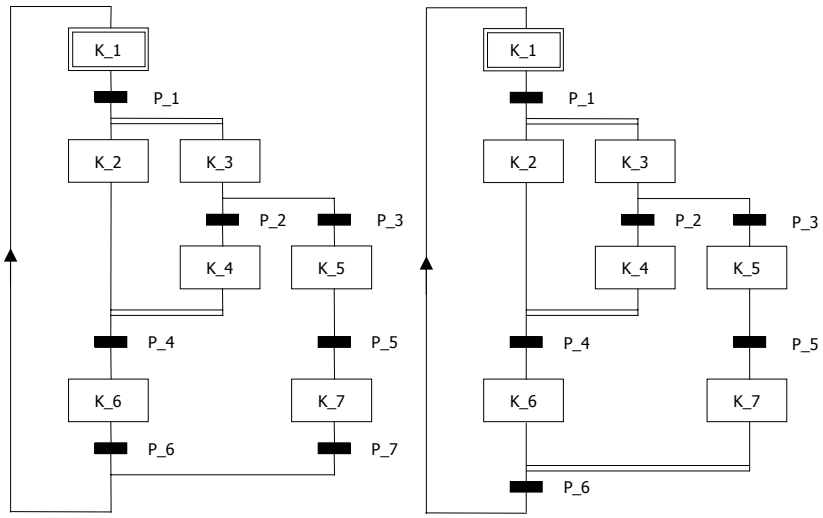
Izvajanje

- Začetno stanje je določeno z začetnim korakom
- Sprememba stanja korakov se zgodi ob sprožitvi prehoda
- Prehod je omogočen, če so aktivni vsi koraki, od katerih gredo povezave na ta prehod
- Prehod se lahko sproži, če
  - je omogočen
  - je dovzeten, kar pomeni, da je prehodni pogoj izpolnjen

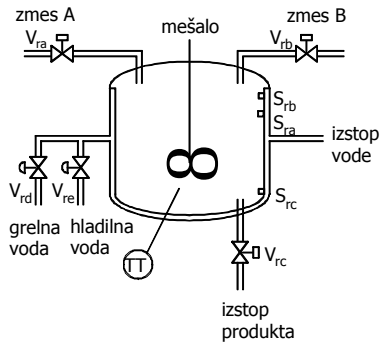
Izvajanje /2

- Ob sprožitvi prehoda postanejo vsi koraki pred njim neaktivni, vsi koraki za njim pa aktivni
- Vsi prehodi, ki se v danem trenutku lahko sprožijo, se sprožijo sočasno
- Stanje koraka je lahko nestabilno
  - takoj po aktiviranju se deaktivira
  - kljub temu se izvedejo pulzne in zadržane akcije
- Problem so situacije, ko korak omogoča dva ali več prehodov (S1 v prejšnjem primeru)

Nepravilni diagrami

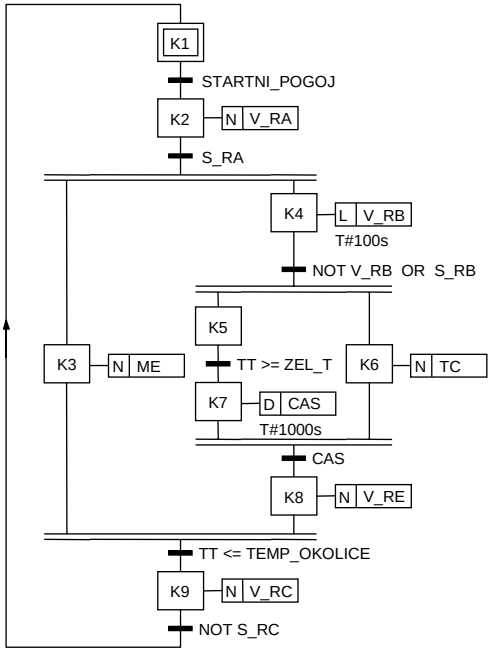


Primer



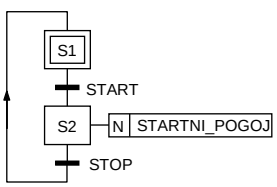
Tipki:
Start - pričetek obratovanja
Stop - zaustavitev po koncu šarže

- Reaktor v šaržnem sistemu
- Recept
  - napolnimo surovino A (do stikala S_{ra})
  - mešamo in napolnimo surovino B (100 s ali do S_{rb})
  - vključimo regulacijo temperature
  - čakamo 1000 s pri želeni temperaturi
  - hladimo vsebino
  - izpraznimo reaktor



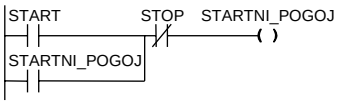
Za zagon/zaustavitev potreben vzporedno delujoč program v SFC ali LD:

a)



(* STARTNI_POGOJ je globalna spremenljivka *)

b)



3.2.6 Seznam ukazov (IL)

- Program vnašamo s pisanjem zaporedja ukazov, ki ponazarjajo osnovne operacije PLK
- Vsak ukaz se začne v novi vrstici in vsebuje operator z morebitno modifikacijo, in če je potrebno, sledi seznam operandov ločenih z vejicami
- Pred ukazom je lahko oznaka vrstice, ki jo od ukaza ločuje znak »:«
- Osnovna interpretacija ukazne vrstice je naslednja:

  novi_rezultat := trenutni_rezultat OPERATOR operand

  npr.:

  novi_rezultat := trenutni_rezultat AND %IX1

Nabor ukazov

Ukazi jezika Instruction List			
Št.	**UKAZ**	**MODIF.**	**POMEN**
1	**LD**	**N**	Naloži vrednost operanda kor trenutni rezultat
2	**ST**	**N**	Shrani rezultat v lokacijo operanda
3	**S**		Postavi operand na 1, če je trenutni rezultat 1
	R		Postavi operand na 0, če je trenutni rezultat 1
4	**AND**	**N, (**	Logični IN
5	**&**	**N, (**	Logični IN
6	**OR**	**N, (**	Logični ALI
7	**XOR**	**N, (**	Ekskluzivni logični ALI
7a	**NOT**		Logična negacija (eniški komplement)
8	**ADD**	**(**	Seštevanje
9	**SUB**	**(**	Odštevanje
10	**MUL**	**(**	Množenje

Nabor ukazov /2

Št.	UKAZ	MODIF.	POMEN
11	**DIV**	**(**	Deljenje
11a	**MOD**	**(**	Deljenje po modulu
12	**GT**	**(**	Primerjava: >
13	**GE**	**(**	Primerjava: >=
14	**EQ**	**(**	Primerjava: =
15	**NE**	**(**	Primerjava: <>
16	**LE**	**(**	Primerjava: <=
17	**LT**	**(**	Primerjava: <
18	**JMP**	**C, N**	Skok na oznako
19	**CAL**	**C, N**	Klic funkcijskega bloka
20	**RET**	**C, N**	Vračanje iz klicane funkcije, funkcijskega bloka ali programa
21	**)**		Zaključek izraza, ki se je pričel z »)«

Primeri /1

- Stanje izhoda %Q4.0 se postavi na 1 in ostane enako 1, če:
  - je stanje vhodov %I0.0 in %I0.1 enako 1
  - ali če je stanje vhoda %I0.2 enako 0
- Stanje izhoda %Q4. 0 se postavi na 0 in ostane enako 0, če:
  - je stanje vhodov %I0.3 in %I0.4 enako 1
  - ali če je stanje vhoda %I0.2 enako 1

IL:

```
LD    %I0.0
AND   %I0.1
ORN   %I0.2
S     %Q4.0
```

IL:

```
LD    %I0.3
AND   %I0.4
OR    %I0.2
R     %Q4.0
```

Primeri /2

- Stanje izhoda %Q4.1 je 1, če:
  - je stanje vhodov %I0.0 in %I0.1 enako 1
  - ali če je stanje vhoda %I0.2 enako 0
- Stanje izhoda %Q4.2 je 1, če:
  - je stanje vhodov %I0.0, %I0.1 in %I0.3 enako 1
  - ali če je stanje vhoda %I0.2 enako 0 in stanje vhoda %I0.3 enako 1
- Izvedba z IL:

```
LD      %I0.0
AND     %I0.1
ORN     %I0.2
ST      %Q4.1
AND     %I0.3
ST      %Q4.2
```

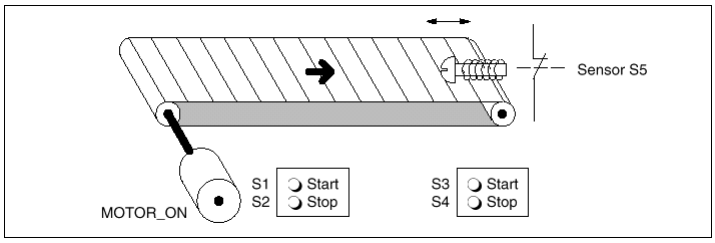
Primeri /3

- Stanje spremenljivke VKLOP je 1, če
  - je stanje vhoda VARNOST enako 1 in stanje vhoda ZAUSTAVITEV enako 0
  - in je stanje vhoda AUTO_START enako 1 ali stanje vhoda MAN_START enako 1
- Izvedba z IL:

```
LD      VARNOST
ANDN    ZAUSTAVITEV
AND(    AUTO_START
OR      MAN_START
)
ST      VKLOP
```

Primeri /4

- Krmiljenje tekočega traku



Komponenta sistema	**Absolutni naslov**	**Simbol**
Tipka Start	I 1.1	S1
Tipka Stop Switch	I 1.2	S2
Tipka Start Switch	I 1.3	S3
Tipka Stop Switch	I 1.4	S4
Senzor	I 1.5	S5
Motor	Q 4.0	MOTOR_ON

Primeri /5

- Zahteve pri krmiljenju tekočega traku:

  S pritiskom tipke Start na eni ali drugi strani vključimo motor traku. Motor ostane vključen, ko tipko Start spustimo, izključi pa se, če na eni ali drugi strani pritisnemo tipko Stop ali če se aktivira senzor na koncu traku.

- Izvedba z IL:

```
LD    S1
OR    S3
S     MOTOR_ON
LD    S2
OR    S4
ORN   S5
R     MOTOR_ON
```

3.2.7 Strukturiran tekst (ST)

- Program vnašamo s pisanjem zaporedja stavkov, ki ponazarjajo niz osnovnih operacij PLK
- Vsak stavek se zaključi z znakom »;«
- Stavek lahko vključuje enega ali več izrazov
- Izraze sestavljajo operatorji in operandi
- V vlogi operanda lahko nastopa nov izraz
- Primer:

```
IF VKLOP AND (POGOJ1 OR POGOJ2) THEN
        TEMP:= 2*SQRT(MAX(SENZOR1,SENZOR2));
END_IF;
```

Stavki

Stavki jezika ST		
Št.	**Tip stavka**	**Primeri**
1	Prireditveni stavek	`A := B;  CV := CV+1; C := SIN(X);`
2	Klic funkcijskega bloka in uporaba izhodov bloka	`CMD_TMR(IN:=%IX5, PT:=T#300ms) ;` `A := CMD_TMR.Q ;`
3	`RETURN`	`RETURN ;`

4	`IF`	`D := B*B - 4*A*C ;`
		`IF D < 0.0 THEN NROOTS := 0 ;`
		`ELSIF D = 0.0 THEN`
		`  NROOTS := 1 ;`
		`  X1 := - B/(2.0*A) ;`
		`ELSE`
		`  NROOTS := 2 ;`
		`  X1 := (- B + SQRT(D))/(2.0*A) ;`
		`  X2 := (- B - SQRT(D))/(2.0*A) ;`
		`END_IF ;`

Stavki /2

Št.	Tip stavka	Primeri

5	CASE	TW := BCD_TO_INT(THUMBWHEEL);
		TW_ERROR := 0;
		CASE TW OF
		  1,5: DISPLAY := OVEN_TEMP;
		  2: DISPLAY := MOTOR_SPEED;
		  3: DISPLAY := GROSS - TARE;
		  4,6..10: DISPLAY := STATUS(TW - 4);
		ELSE DISPLAY := 0 ;
		  TW_ERROR := 1;
		END_CASE;
		QW100 := INT_TO_BCD(DISPLAY);

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

Stavki /3

Št.	Tip stavka	Primeri
7	WHILE	J := 1; WHILE J <= 100 & WORDS[J] <> 'KEY' DO J := J+2 ; END_WHILE ;
8	REPEAT	J := -1 ; REPEAT J := J+2 ; UNTIL J = 101 OR WORDS[J] = 'KEY' END_REPEAT ;
9	EXIT	EXIT ;
10	Prazen stavek	;

Opomba: stavek **EXIT** je definiran v povezavi s ponavljalnimi stavki (**FOR**, **WHILE**, **REPEAT**).

Primeri /1

- Stanje izhoda %Q4.0 se postavi na 1 in ostane enako 1, če:
  - je stanje vhodov %I0.0 in %I0.1 enako 1
  - ali če je stanje vhoda %I0.2 enako 0

```
ST:     IF %I0.0 AND %I0.1 OR NOT %I0.2 THEN
            %Q4.0 := 1;
        END_IF;
```

- Stanje izhoda %Q4. 0 se postavi na 0 in ostane enako 0, če:
  - je stanje vhodov %I0.3 in %I0.4 enako 1
  - ali če je stanje vhoda %I0.2 enako 1

```
ST:     IF %I0.3 AND %I0.4 OR %I0.2 THEN
            %Q4.0 := 0;
        END_IF;
```

Primeri /2

- Stanje izhoda %Q4.1 je 1, če:
  - je stanje vhodov %I0.0 in %I0.1 enako 1
  - ali če je stanje vhoda %I0.2 enako 0
- Stanje izhoda %Q4.2 je 1, če:
  - je stanje vhodov %I0.0, %I0.1 in %I0.3 enako 1
  - ali če je stanje vhoda %I0.2 enako 0 in stanje vhoda %I0.3 enako 1
- Izvedba s ST:

```
%Q4.1 := %I0.0 AND %I0.1 OR NOT %I0.2;
%Q4.2 := (%I0.0 AND %I0.1 OR NOT %I0.2) AND %I0.3;
```

Primeri /3

- Stanje spremenljivke VKLOP je 1, če
  - je stanje vhoda VARNOST enako 1 in stanje vhoda ZAUSTAVITEV enako 0
  - in je stanje vhoda AUTO_START enako 1 ali stanje vhoda MAN_START enako 1
- Izvedba s ST:

```
VKLOP := VARNOST AND NOT ZAUSTAVITEV AND
  (AUTO_START OR MAN_START);
```

Primeri /4

- Zahteve pri krmiljenju tekočega traku:

  S pritiskom tipke Start na eni ali drugi strani vključimo motor traku. Motor ostane vključen, ko tipko Start spustimo, izključi pa se, če na eni ali drugi strani pritisnemo tipko Stop ali če se aktivira senzor na koncu traku.

- Izvedba s ST:

```
IF S1 OR S3 THEN
        MOTOR_ON:=1;
END_IF;
IF S2 OR S4 OR NOT S5 THEN
        MOTOR_ON=0;
END_IF;
```

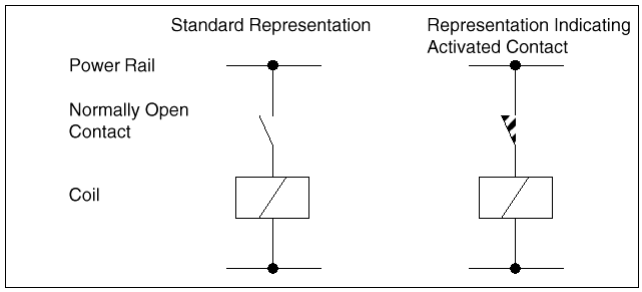
3.2.8 Lestvični diagram

- Program vnašamo z množico standardiziranih grafičnih elementov
- Simbole zlagamo v 'prečke', podobne elementom relejskih lestvičnih logičnih shem
- Prečke so levo in desno omejene z napajalnima vodnikoma (power rails)
  - navpični črti
  - leva predstavlja vir napajanja
  - desna predstavlja ponor
  - desna črta se lahko opušča

Kontakti

Statični kontakti		
Št.	Simbol	Opis
1 2	*** --| |-- ali *** --! !--	**Normalno odprt oz. delovni kontakt** Stanje povezave na levi se preslika preko kontakta (kontakt prevaja), če je stanje pripadajoče logične spremenljivke (označene z ***) enako **1**. Sicer je stanje povezave desno od kontakta enako logični **0**.
3 4	*** --|/|-- ali *** --!/!--	**Normalno zaprt oz. mirovni kontakt** Stanje povezave na levi se preslika preko kontakta (kontakt prevaja), če je stanje pripadajoče logične spremenljivke enako **0**. Sicer je stanje povezave desno od kontakta enako logični **0**.

Normalno odprt (delovni) kontakt

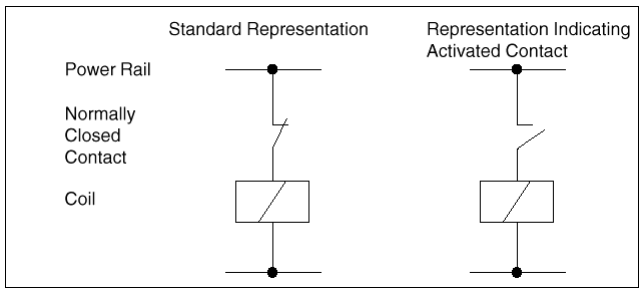


Programska interpretacija (logični pogoj):

x=1, npr.: IF x=1 THEN ...

x je logična spremenljivka, prirejena kontaktu

Normalno zaprt (mirovni) kontakt



Programska interpretacija (logični pogoj):

x=0, npr.: IF x=0 THEN ...

x je logična spremenljivka, prirejena kontaktu

Kontakti /2

Prehodno odprti kontakti		
5 **6**	*** **--|P|--** ali *** **--!P!--**	**Detektor pozitivnega prehoda** Stanje povezave na desni je enako **1** za čas enega programskega cikla od trenutka, ko se zazna prehod stanja pripadajoče logične spremenljivke z **0** na **1** ter je ob tem stanje na levi enako **1**. V vseh ostalih primerih je stanje povezave desno od kontakta enako logični **0**.
7 **8**	*** **--|N|--** ali *** **--!N!--**	**Detektor negativnega prehoda** Stanje povezave na desni je enako **1** za čas enega programskega cikla od trenutka, ko se zazna prehod stanja pripadajoče logične spremenljivke z **1** na **0** ter je ob tem stanje na levi enako **1**. V vseh ostalih primerih je stanje povezave desno od kontakta enako logični **0**

Programska interpretacija (logični pogoj):

P: $\mathbf{(x_k=1)\ AND\ (x_{k-1}=0)}$

N: $\mathbf{(x_k=0)\ AND\ (x_{k-1}=1)}$

Tuljave

Trenutno delujoče tuljave		
1	*** **--( )--**	**Tuljava** Stanje povezave na desni se preslika v stanje pripadajoče logične spremenljivke.
2	*** **--(/)--**	**Negirana oz. inverzno delujoča tuljava** Negirano stanje povezave na desni se preslika v stanje pripadajoče logične spremenljivke. To pomeni, da postane v primeru stanja povezave **1** vrednost logične spremenljivke enaka **0** in obratno, v primeru stanja povezave **0** postane vrednost logične spremenljivke enaka **1**.

Programska interpretacija (prireditveni stavek):

y := x; normalna tuljava,
y := NOT x; inverzno delujoča tuljava

y je logična spremenljivka, prirejena tuljavi

Tuljave /2

Tuljave z zadržanim delovanjem		
3	*** **--(S)--**	**Tuljava SET (zapah)** Pripadajoča logična spremenljivka se postavi na vrednost **1**, če je tudi povezava na levi v stanju **1**. Vrednost spremenljivke nato ostane nespremenjena, dokler je ne postavi na **0** tuljava tipa **RESET**.
4	*** **--(R)--**	**Tuljava RESET** Pripadajoča logična spremenljivka se postavi na vrednost **0**, če je povezava na levi v stanju **1**. Vrednost spremenljivke nato ostane nespremenjena, dokler je ne postavi na **1** tuljava tipa **SET**.

Programska interpretacija (pogojni prireditveni stavek):

SET:
```
IF x THEN
   y := 1;
END_IF;
```

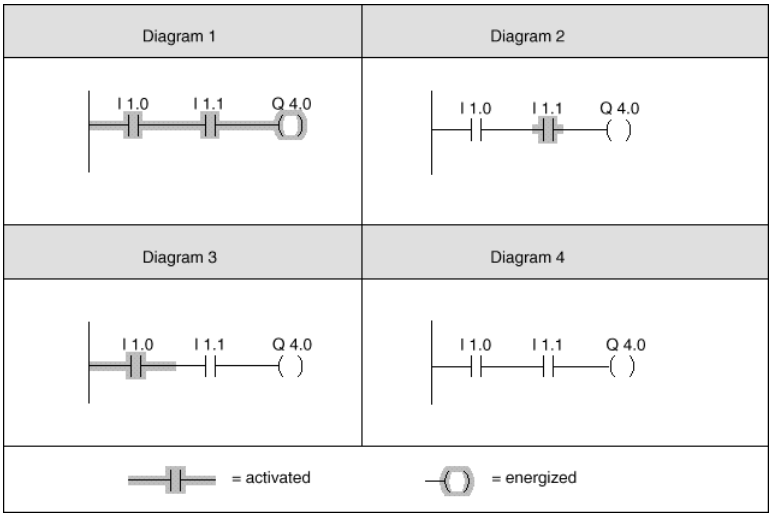
RESET:
```
IF x THEN
   y := 0;
END_IF;
```

y je logična spremenljivka, prirejena tuljavi

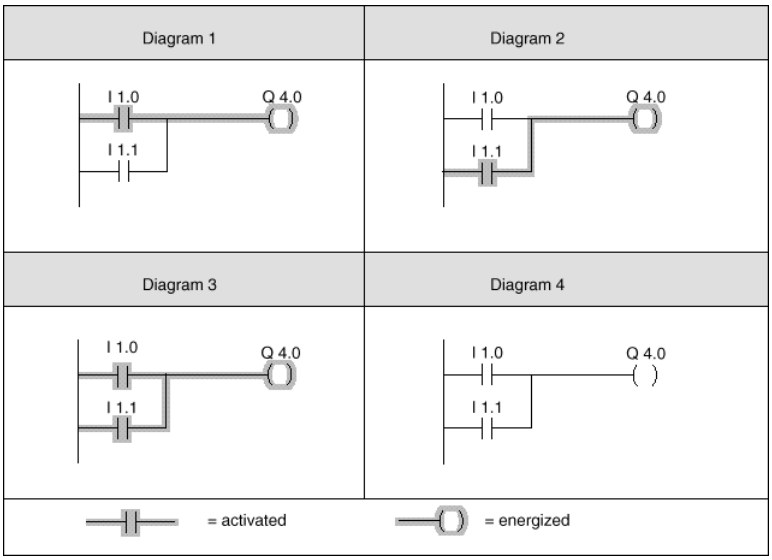
Povezave

- Povezave so lahko navpične ali vodoravne
- Vsaka povezava ima stanje
  - vključena/izključena oz. prevaja/ne prevaja
  - vir napajanja je vedno vključen
  - stanje ponora ni definirano
  - vodoravna povezava predstavlja prenos stanja
  - navpična povezava predstavlja logični ALI stanja vodoravnih povezav na njeni levi
  - stanje navpične povezave se prenese na vodoravne povezave na njeni desni

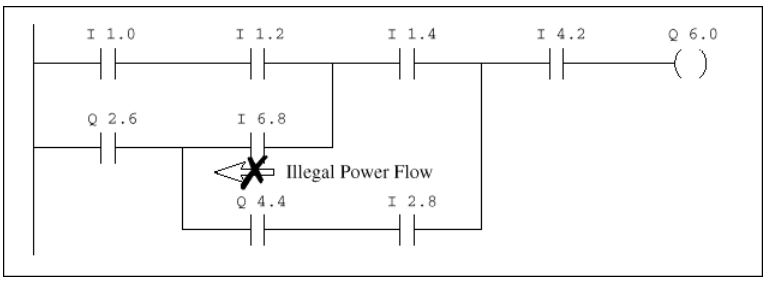
Zaporedna vezava



Vzporedna vezava

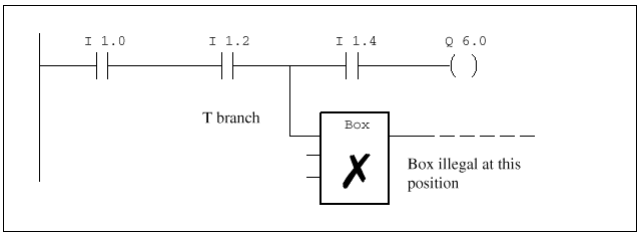


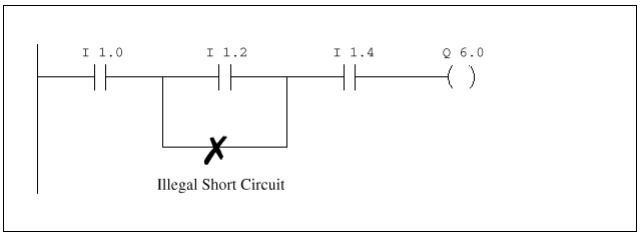
Nedovoljene vezave



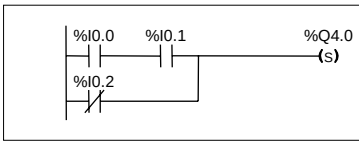
Če je I1.4 v stanju 0 in I6.8 v stanju 1, bi tok lahko tekel v označeni smeri, ki pa ni dovoljena

Nedovoljene vezave /2



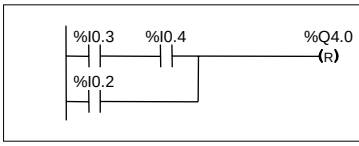


Primeri



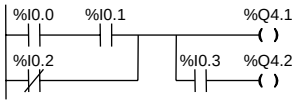
Stanje izhoda %Q4.0 se postavi na 1 in ostane enako 1, če:
- **je stanje vhodov %I0.0 in %I0.1 enako 1**
- **ali če je stanje vhoda %I0.2 enako 0**

```
IF %I0.0 AND %I0.1 OR NOT %I0.2 THEN
        %Q4.0 := 1;
END_IF;
```



Stanje izhoda %Q4. 0 se postavi na 0 in ostane enako 0, če:
- **je stanje vhodov %I0.3 in %I0.4 enako 1**
- **ali če je stanje vhoda %I0.2 enako 1**

```
IF %I0.3 AND %I0.4 OR %I0.2 THEN
        %Q4.0 := 0;
END_IF;
```



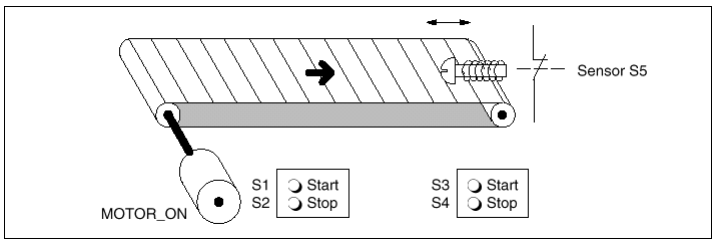
Stanje izhoda %Q4.1 je 1, če:
- **je stanje vhodov %I0.0 in %I0.1 enako 1**
- **ali če je stanje vhoda %I0.2 enako 0**

Stanje izhoda %Q4.2 je 1, če:
- **je stanje vhodov %I0.0, %I0.1 in %I0.3 enako 1**
- **ali če je stanje vhoda %I0.2 enako 0 in stanje vhoda %I0.3 enako 1**

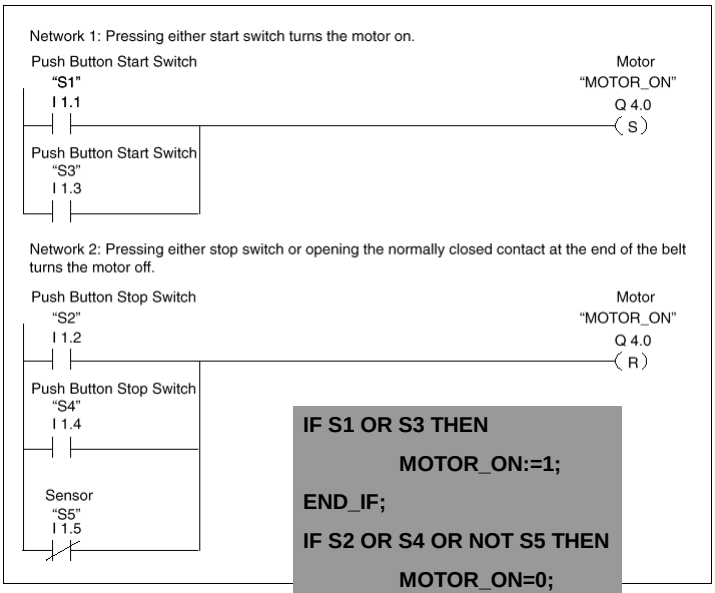
```
%Q4.1 := %I0.0 AND %I0.1 OR NOT %I0.2;
%Q4.2 := (%I0.0 AND %I0.1 OR NOT %I0.2) AND %I0.3;
```

Primeri /2

- Krmiljenje tekočega traku

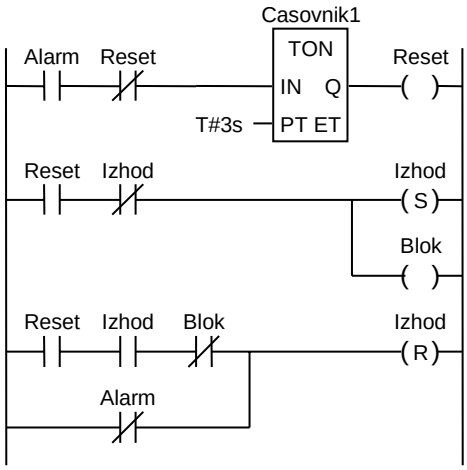


Komponenta sistema	**Absolutni naslov**	**Simbol**
Tipka Start	I 1.1	S1
Tipka Stop Switch	I 1.2	S2
Tipka Start Switch	I 1.3	S3
Tipka Stop Switch	I 1.4	S4
Senzor	I 1.5	S5
Motor	Q 4.0	MOTOR_ON



Primeri /3

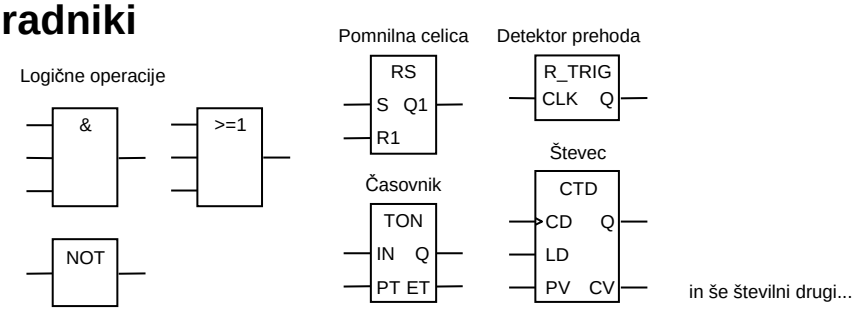
Generiranje periodičnega signala



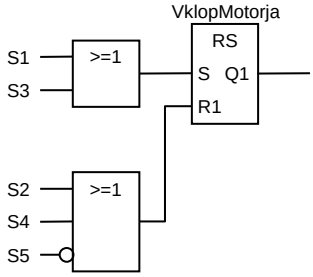
3.2.9 Funkcijski bločni diagram

- Program vnašamo z množico standardiziranih grafičnih elementov - blokov
- Bloke povezujemo v vezja, podobna digitalnim logičnim elektronskim vezjem
- Dodatna pravila
  - bloki so vedno predstavljeni s pravokotniki
  - vhodni signali vedno vstopajo v blok z leve
  - izhodni signali izstopajo na desni
  - ločitev med funkcijami in funkcijskimi bloki
  - vrstni red izvajanja blokov v zanki ni definiran

Gradniki



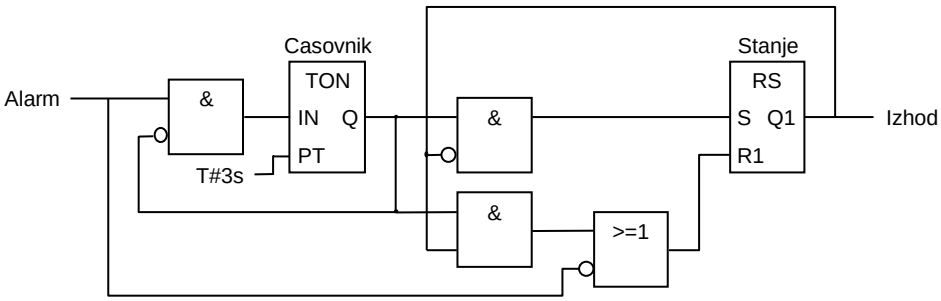
- Primer: krmiljenje tekočega traku (isti primer kot pri LD)



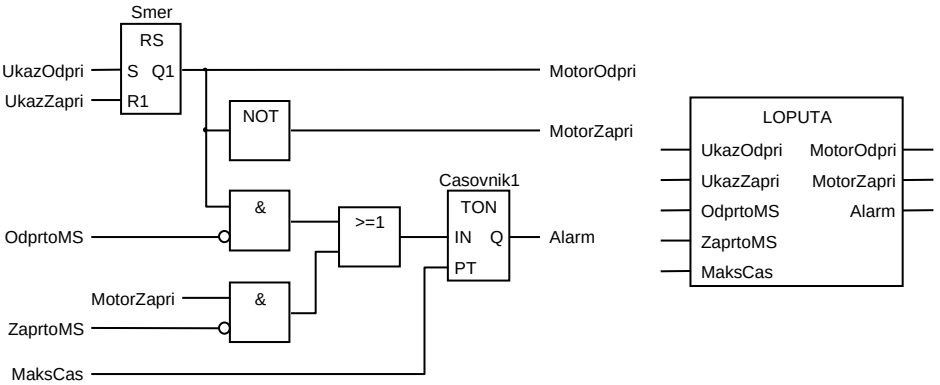

```
IF S1 OR S3 THEN
        MOTOR_ON:=1;
END_IF;
IF S2 OR S4 OR NOT S5 THEN
        MOTOR_ON=0;
END_IF;
```

Primer

- Generiranje periodičnega signala

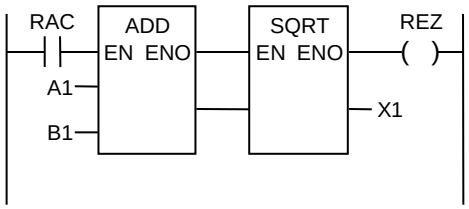


Združevanje blokov



Uporaba blokov v lestvičnem diagramu

- Funkcije in funkcijske bloke lahko uporabimo tudi v lestvičnem diagramu
- V ta namen imajo bloki lahko dodaten vhod (EN) in izhod (ENO)



3.2.10 Programska orodja s podporo IEC 61131-3

- SIMATIC STEP 7
  - v osnovni različici podpira IL, LD, FBD (ni povsem v skladu s standardom)
  - možnost dodatnih modulov za SFC in ST
- Nekatera odstopanja od standarda (v5.1)
  - v imenih spremenljivk ločuje med malimi in velikimi črkami
  - zapis šestnajstiških vrednosti: W#16#F23C
  - ne podpira nepredznačenih celoštevilskih tipov
  - od sestavljenih tipov podpira le strukture in polja
  - ne pozna globalnih spremenljivk, namesto tega podatkovni bloki (DB)

Programska orodja /2

- Odstopanja od standarda (nadaljevanje)
  - implementacija aritmetičnih funkcij odvisna od CPE
  - nestandardni števci in časovniki
  - v SFC le akcije tipov N, R, S, L, D, poleg tega še dodatni nestandardni tipi akcij
  - akcija je lahko le navadna spremenljivka ali klic podprograma (nestandarden, s CALL)
  - ne pozna konfiguracije,virov,opravil in programov
  - namesto tega organizacijski bloki (OB) in funkcijski bloki (FB)
  - v vseh blokih je dovoljeno neposredno naslavljanje vhodov, izhodov in pomnilniških lokacij

Programska orodja /3

- GX IEC Developer (prej Melsec Medoc plus)
  - podpira IL, ST, LD, FBD in SFC
- Nekatera odstopanja od standarda (v4.0)
  - ne podpira datumov in absolutnega časa
  - ne podpira nepredznačenih celoštevilskih tipov
  - od sestavljenih tipov podpira le strukture in polja
  - ne podpira inicializacije spremenljivk

Programska orodja /4

- Odstopanja od standarda (nadaljevanje)
  - ne podpira aritmetičnih funkcij ene spremenljivke (ABS, SQRT, LOG, SIN, COS ...)
  - ne podpira operacij nad nizi znakov, časi, datumi ...
  - v SFC ne podpira spremenljivk ime.X, ime.T
  - akcije v SFC so lahko le navadne (tipa N)
  - konfiguracija in viri niso vidni uporabniku, razvidni so le, če program izvozimo v ASCII datoteko