FAKULTETA ZA ELEKTROTEHNIKO

UNIVERZA V LJUBLJANI

RAČUNALNIŠKO VODENJE PROCESOV

Praktikum

ZALOŽBA FE

VITO LOGAR

GAŠPER MUŠIČ

Univerza v Ljubljani
Fakulteta za elektrotehniko

RAČUNALNIŠKO VODENJE PROCESOV
Praktikum

Vito Logar
Gašper Mušič

Ljubljana, 2017

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani
COBISS.SI-ID=292593920
ISBN 978-961-243-346-8 (pdf)

URL: http://msc.fe.uni-lj.si/Download/Logar/RVP_skripta_2017.pdf

Založnik: Založba FE, Ljubljana
Izdajatelj: Fakuleta za elektrotehniko, Ljubljana
Urednik: prof. dr. Sašo Tomažič
Recenzent: izr. prof. dr. Gregor Klančar
2. izdaja

Predgovor

Praktikum je namenjen študentom tretjega letnika smeri Avtomatika na prvostopenjskem visokošolskem strokovnem študijskem programu Aplikativna elektrotehnika. Vsebuje navodila za izvajanje laboratorijskih vaj pri predmetu Računalniško vodenje procesov. Za lažjo izvedbo laboratorijskih vaj so v gradivu razložene potrebne tehnike programiranja, podani so osnovni podatki o uporabljeni krmilno-regulacijski opremi ter dodana kratka navodila za delo s pripadajočo programsko opremo.

Vsebinsko je snov razdeljena v štiri tematske sklope, ki tvorijo štiri laboratorijske vaje. Vsaka vaja obsega več nalog in se predvidoma razteza preko več terminov za delo v laboratoriju. Prva vaja je namenjena seznanjanju z načrtovanjem in izvedbo logičnega in sekvenčnega vodenja. Študentje se s programiranjem osnovnih logičnih in števno-časovnih operacij seznanijo ob uporabi pravih industrijskih krmilnikov in pripadajočih orodij za programiranje. Druga vaja obravnava izvedbo zahtevnej ših koračnih krmilij z lestvičnim diagramom. Študentje se seznanijo s sistematično izvedbo prehajanja stanj v lestvičnem diagramu in preizkušajo tako načrtana krmilja na laboratorijskem modelu modularnega proizvodnega sistema. Tretja vaja obravnava izvedbo zahtevnej ših krmilij s strukturiranim tekstom na laboratorijskem modelu modularnega proizvodnega sistema. Četrta vaja obravnava izvedbo regulacijskih sistemov, pri kateri se študentje seznanijo s konfiguriranjem in parametriranjem programirljivih logičnih krmilnikov za namen izdelave sistemov zveznega vodenja.

Za nastanek dela so zaslužni sodelavci Laboratorija za modeliranje, simulacijo in vodenje ter Laboratorija za avtonomne mobilne sisteme. Vsem iskreno hvala.

Vito Logar, Gašper Mušič

Kazalo

Laboratorijska vaja št. 1

Osnove programiranja programirljivih logičnih krmilnikov

Pri vaji bomo spoznali programirljive logične krmilnike (PLK), orodja za programiranje krmilnikov in osnove programiranja krmilnikov z lestvičnimi diagrami.

1.1 Lestvični diagram

Lestvični diagram (angl. Ladder Diagram - LD) je eden najbolj razširjenih načinov programiranja PLK. Kot eden od grafičnih programskih jezikov je vključen tudi v mednarodni standard IEC 61131-3.

Lestvični diagram temelji na relejskih vezalnih shemah, ki predstavljajo električni načrt relejskega krmilja. V takšni shemi so narisani kontakti, katerih vzporedne in/ali zaporedne vezave predstavljajo testiranje logičnih pogojev, in tuljave, ki se aktivirajo, če vezje kontaktov med posamezno tuljavo in električnim napajanjem prevaja. Za boljšo preglednost takšnega električnega načrta se je uveljavil način risanja z navpičnima napajalnima vodnikoma in vodoravno usmerjenimi vezji kontaktov in tuljav med njima. Celotno vezje ob tem spominja na obliko lestve in od tu ime lestvični diagram.

Z razvojem programirljivih logičnih krmilnikov se je pojavila potreba po kar najenostavnejšem načinu programiranja in zato so obliko lestvičnega diagrama prenesli v programski jezik. Simbole gradnikov električnih vezij so zamenjali s simboli programskih elementov, pri čemer sta simbola za kontakt in tuljavo ostala enaka. Simbole zlagamo v »prečke«, podobne vezjem v relejskih lestvičnih logičnih shemah. Prečke so levo in desno omejene z napajalnima vodnikoma (angl. power rails). To sta navpični črti, pri čemer leva predstavlja vir napajanja in desna predstavlja ponor. Desna črta se lahko opušča. Pri interpretaciji logičnega delovanja programa privzamemo, da je napajalni vodnik v stanju 1. Poleg kontaktov, tuljav in napajalnih vodnikov sestavljajo lestvični diagram še povezave ter dodatni gradniki, kot so funkcijski bloki in funkcije.

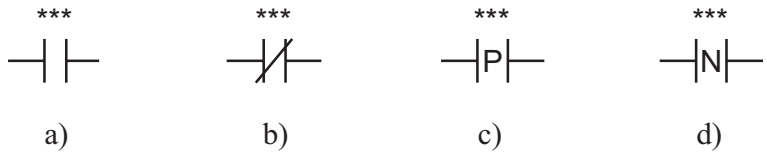
Vsi simboli lestvičnih diagramov imajo smisel le, če so prirejeni določeni vhodno/izhodni sponki krmilnika ali notranji pomnilniški lokaciji, s čimer dobijo nek pomen glede na proces, ki ga s krmilnikom vodimo. Za prireditev programskih elementov k fizikalnim uporabimo prireditveno tabelo.

Da bi tudi funkcionalno čimbolj posnemali delovanje električnih vezij, kjer se vse veje vezja »izvajajo« hkrati, se je uveljavil način izvajanja programa v ciklih. Na začetku vsakega cikla se preberejo stanja binarnih vhodov krmilnika in se shranijo v pomnilnik v t.i. »vhodno sliko«. Program nato obdeluje »zamrznjeno« sliko vhodov in rezultate vpisuje v pomnilnik, rezerviran za izhode, t.i. »izhodno sliko«. Ob koncu cikla, ko se obdelajo vse veje diagrama, se vsebina izhodne slike prepiše na izhodne sponke krmilnika. Na ta način dosežemo, da se izhodi, ki so posledica sočasnih sprememb vhodov, tudi v resnici spremenijo sočasno, ne glede na vrstni red prečk. Ker je čas izvajanja enega cikla programa zelo kratek (običajno nekaj delcev sekunde), je navzven videti, kot da se vse prečke izvajajo hkrati.

Kadar rezultat izvajanja ene prečke vpliva na pogoje v drugih prečkah, je vseeno potrebno nekaj pazljivosti pri vrstnem redu risanja prečk.

1.1.1 Kontakti

Normalno odprt oz. delovni kontakt (slika 1.1 a) predstavlja preverjanje, če je logična spremenljivka v visokem stanju. Bolj natančno, stanje povezave na levi se preslika preko kontakta (kontakt prevaja), če je stanje pripadajoče logične spremenljivke (na sliki označene z ***) enako 1. Če temu ni tako, je stanje povezave desno od kontakta enako logični 0.



Slika 1.1: Kontakti v lestvičnem diagramu

Normalno zaprt oz. mirovni kontakt (slika 1.1 b) predstavlja preverjanje, če je logična spremenljivka v nizkem stanju. Bolj natančno, stanje povezave na levi se preslika preko kontakta (kontakt prevaja), če je stanje pripadajoče logične spremenljivke enako 0. Če temu ni tako, je stanje povezave desno od kontakta enako logični 0.

Programsko lahko interpretiramo delovni kontakt kot logični pogoj $x = 1$, npr.: IF $x = 1$ THEN ... Pri tem je x logična spremenljivka, prirejena kontaktu. Mirovni kontakt interpretiramo kot logični pogoj $x = 0$, npr.: IF $x = 0$ THEN ...

Prehodno odprti kontakti omogočajo zaznavanje spremembe stanja logične spremenljivke. Če uporabimo detektor pozitivnega prehoda (slika 1.1 c), bo stanje pove-

zave na desni 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 0.

Detektor negativnega prehoda (slika 1.1 d) postavi stanje povezave na desni na 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.

Programsko interpretiramo detektor pozitivnega prehoda kot logični pogoj $(x_k = 1)$ AND $(x_{k-1} = 0)$, pri čemer je x_k trenutno stanje logične spremenljivke in x_{k-1} stanje spremenljivke v prejšnjem programskem ciklu. Detektor negativnega prehoda interpretiramo kot logični pogoj $(x_k = 0)$ AND $(x_{k-1} = 1)$.

1.1.2 Tuljave

Normalna, trenutno delujoča tuljava (slika 1.2 a) preslika stanje povezave na levi v stanje pripadajoče logične spremenljivke. Programsko jo interpretiramo kot prireditveni stavek: $y := x$; pri tem je y logična spremenljivka, prirejena tuljavi, x pa je stanje povezave levo od tuljave.

a) b) c) d)

Slika 1.2: Tuljave v lestvičnem diagramu

Negirana oz. inverzno delujoča tuljava (slika 1.2 b) preslika negirano stanje povezave na levi 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. Programsko jo interpretiramo kot prireditveni stavek z negacijo: $y :=$ NOT x;

Tuljave z zadržanim delovanjem povzročijo spremembo vrednosti spremenljivke, ki se ohrani, tudi ko vhodni pogoj ni več izpolnjen. Tuljava SET (zapah) (slika 1.2 c) povzroči, da se pripadajoča logična spremenljivka 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.

Tuljava RESET (slika 1.2 d) postavi pripadajočo logično spremenljivko 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.

Programsko interpretiramo tuljavi SET in RESET kot pogojni prireditveni stavek

(y je logična spremenljivka, prirejena tuljavi, x pa stanje povezave, na katero je tuljava priključena):

tuljava SET:	tuljava RESET:
IF x THEN	IF x THEN
$y := 1$;	$y := 0$;
END_IF;	END_IF;

Tuljavi tipa SET in RESET spremenita vrednost spremenljivke y le tedaj, kadar so vhodni pogoji izpolnjeni (vezje kontaktov pred tuljavo prevaja oz. $x = 1$), normalna in negirana tuljava pa v vsakem primeru priredita spremenljivki bodisi vrednost 0 bodisi vrednost 1, odvisno od tega, ali vezje prevaja ali ne. Normalni ali negirani tuljavi pripadajoči prireditveni stavek bi namreč lahko programsko interpretirali tudi kot:

normalna tuljava:	negirana tuljava:
IF x THEN	IF x THEN
$y := 1$;	$y := 0$;
ELSE	ELSE;
$y := 0$;	$y := 1$;
END_IF;	END_IF;

Razlika v primerjavi s tuljavama SET in RESET je torej v tem, da normalna in negirana tuljava vselej vplivata na vrednost prirejene logične spremenljivke y, tudi takrat, ko logični pogoj ni izpolnjen (del zgornjega stavka, ki sledi besedi ELSE).

S tuljavama tipa SET in RESET lahko spreminjamo vrednost iste spremenljivke večkrat in na različnih mestih v programu, seveda pa je to smiselno le, če se prirejeni vhodni pogoji medsebojno izključujejo. Določeno spremenljivko v povezavi z navadno ali negirano tuljavo pa lahko uporabimo vedno **le na enem mestu v programu**. Vsa vezja diagrama se namreč izvajajo hkrati in moramo zagotoviti, da ne pride do konfliktov med zahtevanimi stanji ene in iste logične spremenljivke.

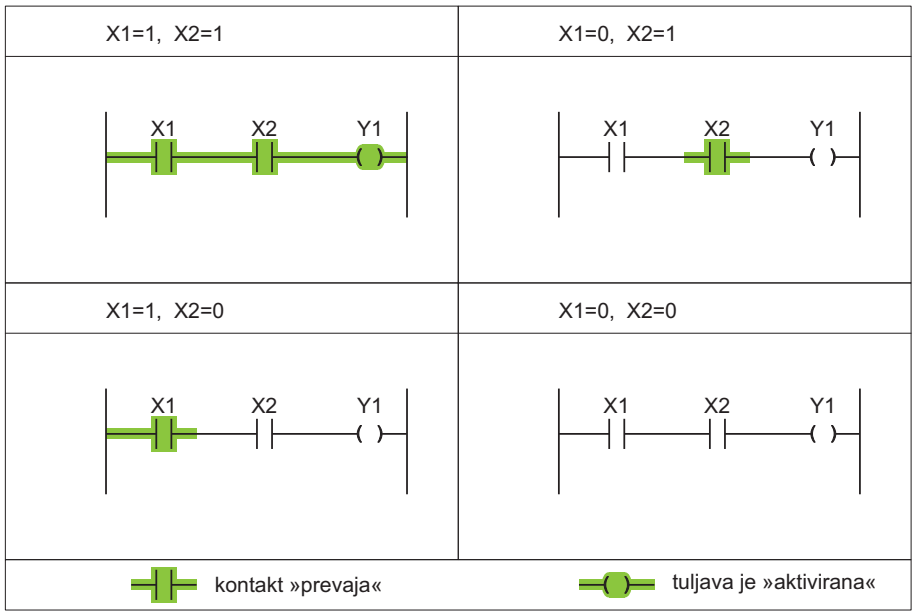
1.1.3 Povezave

S povezavami sestavljamo pogojne elemente v logične pogoje in jih povezujemo na izvršne elemente. Zaporedna vezava pogojnih elementov predstavlja logični IN, vzporedna vezava pa logični ALI (sliki 1.3 in 1.4).

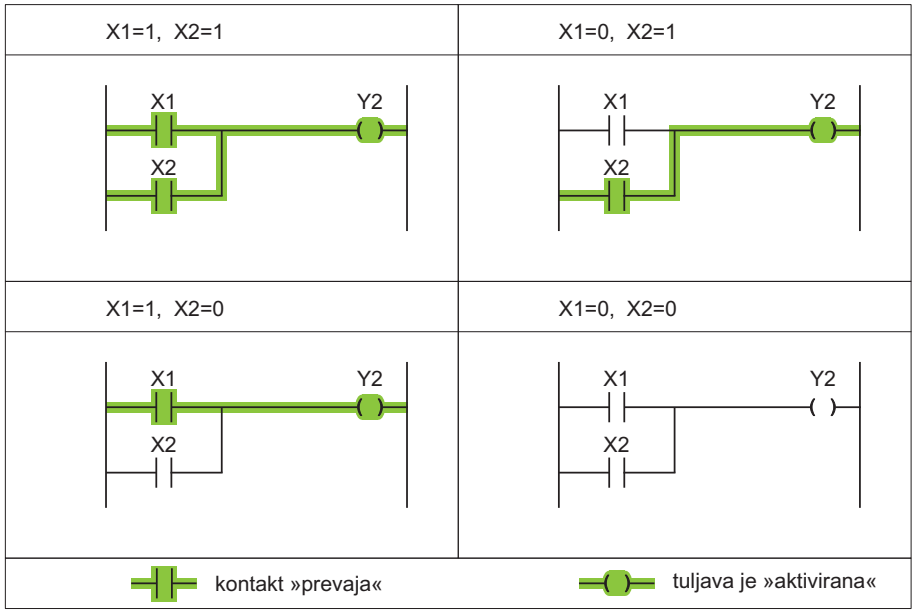
1.1.4 Časovniki

Slika 1.5 prikazuje simbol in delovanje enega od standardiziranih časovnikov, ki jih lahko uporabimo v lestvičnem diagramu.

Prikazan je časovnik z zakasnjenim vklopom **TON**, ki vključi nanj priključen izvršni element z določeno zakasnitvijo po prehodu stanja na vhodu časovnika z 0 na 1. Bolj natančno, deluje tako, da prehod v logično stanje 1 na vhodu IN sproži časovnik. Po času, določenem s PT, se izhod Q spremeni z 0 na 1. Izhod nato ostane



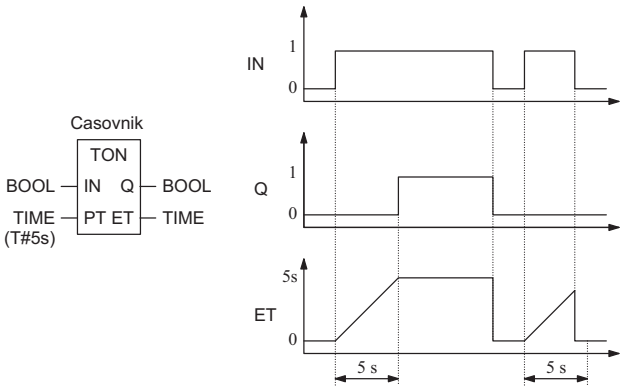
Slika 1.3: Pomen zaporedne vezave v lestvičnem diagramu



Slika 1.4: Pomen vzporedne vezave v lestvičnem diagramu

v stanju 1, dokler vhod IN ne pade na 0. Na izhodu ET lahko odčitamo pretečeni čas od trenutka sprožitve časovnika. Če stanje na vhodu IN pade na 0 še pred iztekom zakasnitve, se časovnik resetira in izhod se ne postavi.

Poleg časovnika **TON** sta standardizirana še dva časovnika, **TP** in **TOF**. Časovnik



Slika 1.5: Standardizirani časovnik z zakasnjenim vklopom

tipa **TP** je pulzni časovnik. Prehod v logično stanje 1 na vhodu IN sproži časovnik. Ob tem se postavi na 1 tudi izhod Q. Po času, določenem s PT, pade izhod Q nazaj na 0. Če se stanje na vhodu IN spremeni še pred iztekom zakasnitve, časovnik na to ne reagira. Na izhodu ET lahko odčitamo pretečeni čas od trenutka sprožitve časovnika.

Časovnik tipa **TOF** je časovnik z zakasnjenim izklopom. Deluje tako, da prehod v logično stanje 1 na vhodu IN postavi na 1 tudi izhod Q. Časovnik se sproži ob prehodu vhoda IN z 1 na 0. Po času, določenem s PT, pade na 0 tudi izhod Q. Pretečeni čas od trenutka sprožitve časovnika lahko odčitamo na izhodu ET. Če stanje na vhodu IN ponovno preide v 1 še pred iztekom zakasnitve, se časovnik resetira in izhod ne pade na 0, ampak ostane na 1.

1.1.5 Števci

Standardizirane so tudi tri vrste števcev. Števec **CTD** je ob zagonu v stanju 0, izhod Q je tedaj 1. Ob visokem stanju vhoda LOAD se postavi v stanje, ki ga določa vhod PV, izhod Q pade na 0. Ob vsakem prehodu vhoda CD z 0 na 1 se stanje števca zmanjša za 1. Ko stanje doseže 0, se postavi izhod Q na 1. Ob nadaljnjih prehodih vhoda CD z 0 na 1 stanje števca narašča v negativni smeri.

Števec **CTU** se ob zagonu ali ob visokem stanju vhoda RESET postavi v stanje 0. Ob vsakem prehodu vhoda CU z 0 na 1 se stanje števca poveča za 1. Ko stanje doseže vrednost vhoda PV, se postavi na 1 izhod Q. Ob nadaljnjih prehodih vhoda CU z 0 na 1 stanje števca še naprej narašča.

Števec **CTUD** združuje delovanje obeh opisanih števcev v enem bloku, pri čemer se stanje števca lahko povečuje ali zmanjšuje. Izhod QU preide v stanje 1, če stanje števca prekorači vrednost PV, izhod QD pa preide v stanje 1, če stanje števca doseže 0. Stanje števca lahko preberemo na izhodu CV, če nanj povežemo celoštevilsko spremenljivko.

1.2 Programsko orodje TIA Portal

Programski paket **Siemens TIA Portal** (TIA - Totally Integrated Automation) predstavlja inženirsko okolje, ki omogoča izvedbo različnih rešitev v avtomatizaciji procesov, od planiranja, zagona, obratovanja, vzdrževanja do nadgradnje obstoječih sistemov avtomatizacije. TIA Portal v enem programskem paketu omogoča programiranje krmilnikov, porazdeljenih vhodno/izhodnih enot, vmesnikov človek-stroj, pogonov ter drugih industrijskih sistemov. Njegov namen je povezati vse glavne gradnike preko vseh štirih nivojev avtomatizacije: 1. nivo upravljanja, 2. nivo operaterjev, 3. nivo krmilnikov ter 4. nivo izvrševanja. TIA Portal sloni na mednarodnem standardu IEC 61131-3, ki določa programsko strukturo in programske jezike za programiranje logičnih krmilnikov in ostalih naprav. Programski paket deluje v operacijskih sistemih Windows, zaradi česar je način dela v njem podoben kot pri ostalih programih v tem okolju. Naštejmo samo nekaj funkcionalnosti, ki jih omogoča TIA Portal:

- vnos, urejanje in shranjevanje projektov za logično, sekvenčno in zvezno vodenje,
- prenos projektov na naprave (krmilnik, ...),
- sprotno spremljanje delovanja programa,
- spremljanje in spreminjanje stanj spremenljivk v krmilniku,
- spreminjanje programa med izvajanjem.

1.2.1 Zagon programskega orodja

Programsko orodje zaženemo preko menija Start z zaporedjem: **Start** > **All Programs** > **Siemens Automation** > **TIA Portal V13**. Odpre se glavno okno programa, kjer lahko odpremo obstoječi projekt, naredimo nov projekt ali pa premaknemo projekt na drugo lokacijo. V menijski vrstici lahko dostopamo tudi do zavihkov s pomočjo, splošnimi informacijami o programu ter naprav, do katerih lahko dostopamo.

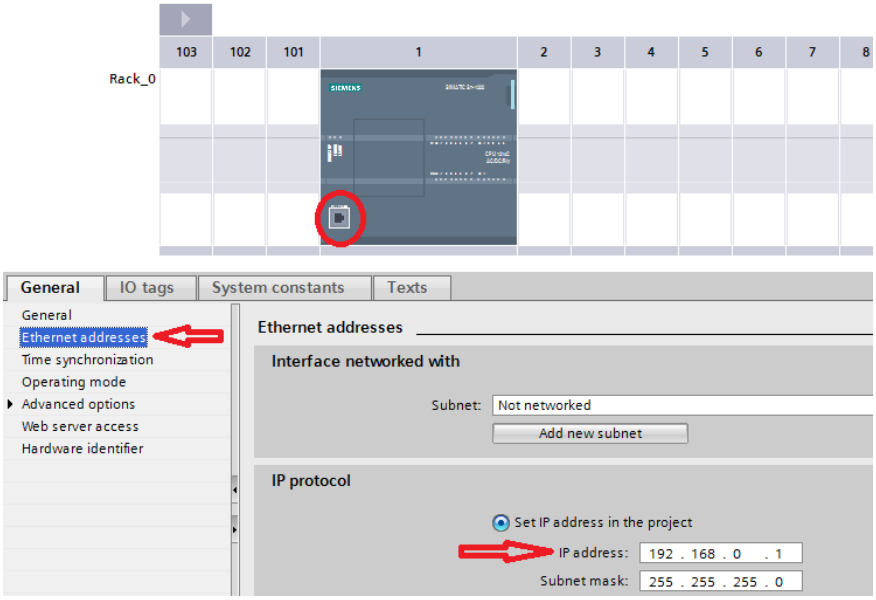
1.2.2 Izdelava novega projekta

Projekt je zaključena celota, sestavljena iz osnovnih gradnikov, kot so organizacijski bloki, funkcijski bloki, funkcije, opravila in seznami spremenljivk, ki jih uporabljamo v programskih blokih. Izdelavo novega projekta pričnemo tako, da v glavnem oknu programa izberemo **Create new project** ter izberemo želeno ime projekta ter mesto, kamor ga bomo shranili. Izbiro potrdimo s tipko **Create**. V naslednjem pogovornem oknu izberemo **Open the project view**, kjer najprej dodamo napravo oz. krmilnik, ki

ga bomo uporabljali. To storimo v levem stranskem meniju z dvoklikom na opcijo **Add new device** > **Controllers** > **SIMATIC S7-1200** > **CPU** > **CPU 1214C AC/DC/Rly** > **6ES7 214-1BG40-0XB0** in pritisnemo **OK**. S tem se v projektu pojavi okno (slika) krmilnika. Ko je krmilnik dodan v projekt, je potrebno v programu določiti še, katerega od petih krmilnikov želimo programirati, in sicer tako, da v program vnesemo njegov IP-naslov. IP-naslovi na vseh krmilnikih so fiksni in so dodeljeni vsaki postaji posebej:

- 1. Postaja: 192.168.222.111
- 2. Postaja: 192.168.222.112
- 3. Postaja: 192.168.222.113
- 4. Postaja: 192.168.222.114
- 5. Postaja: 192.168.222.115

IP-naslov krmilnika, do katerega bomo dostopali, nastavimo v podoknu krmilnika **Device configuration** v levem stranskem meniju, kjer dvokliknemo ikono  in sledimo navodilom, kot je prikazano na sliki 1.6.



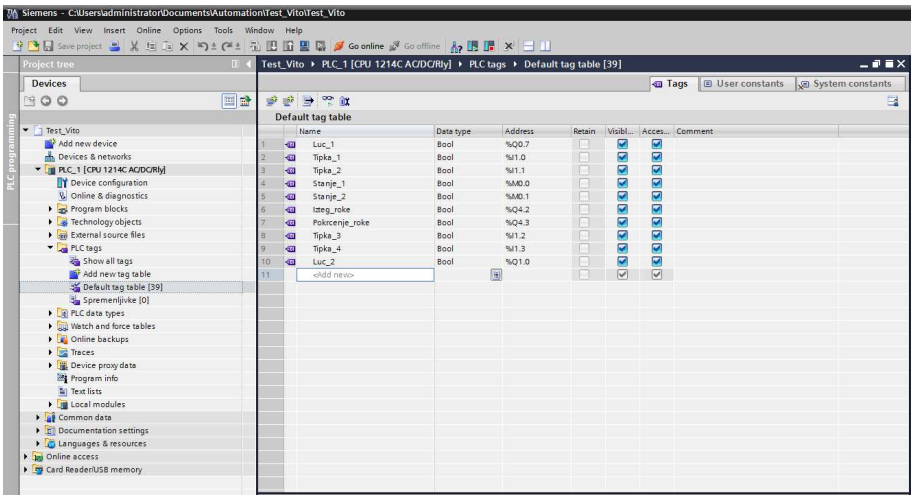
Slika 1.6: Nastavljanje IP-naslova krmilnika

Pomembno: Pri nastavljanju IP-naslova je potrebno paziti, da naslavljamo pravi krmilnik, saj lahko v nasprotnem primeru dostopamo in nalagamo program na druge krmilnike.

Vnos spremenljivk

Zaradi preglednosti ter potrebe po dodatnih internih spremenljivkah (stanja sistema, zastavice), bomo spremenljivke v projektu naslavljali simbolično. Programska oprema sicer omogoča tudi direktno (absolutno) naslavljanje spremenljivk, npr. `%I0.0` predstavlja prvi vhod na krmilniku.

Vse spremenljivke, ki nastopajo v projektu, je potrebno vnesti v tabelo spremenljivk oz. **PLC tags**. To storimo s klikom na izbran krmilnik **PLC1 [CPU 1214C AC/DC/Rly]** > **PLC tags** > **Add new tag table**. Vpišemo želeno ime tabele ter v oknu, ki se odpre, začnemo vnašati spremenljivke. V polje **Add new** vpišemo želeno ime spremenljivke, v **data type** izberemo tip spremenljivke (npr. BOOL), v **Address** pa vpišemo pripadajoč naslov na krmilniku (npr. `%I0.0`, `%Q0.2`, `%M0.5`). Pri tem **I** pomeni vhod v krmilnik, **Q** pomeni izhod iz krmilnika, **M** pa spomin, ki ga uporabljamo za npr. določevanje aktivnih stanj ali zastavic. Seznam spremenljivk v TIA Portalu prikazuje slika 1.7



Slika 1.7: Tabela spremenljivk

Spremenljivk, ki definirajo časovnike in števce, ne vnašamo v tabelo spremenljivk, temveč jih definiramo, ko želimo ustvariti nov časovnik oz. števec. Pri dodajanju novega bloka tipa časovnik ali števec program sam vpraša po želenem imenu tega bloka. Te spremenljivke se nahajajo v zavihku **PLC1 [CPU 1214C AC/DC/Rly]** > **Program blocks** > **System blocks** > **Program resources**.

Opis nekaterih osnovnih gradnikov

Device configuration: je zavihek, s katerim dostopamo do konfiguracije krmilnika. V njem lahko nastavljamo različne parametre procesne enote, kot npr. nastavitve IP-naslova krmilnika, nastavitve dodatnih razširitvenih modulov (signal boards), pregled I/O-povezav (imena spremenljivk, ki so vezane na vhode oz. izhode), idr.

Online & diagnostics: je zavihek, s katerim dostopamo do nastavitev krmilnika, kot npr. nastavitev trenutnega časa, nastavitev IP-naslova krmilnika, resetiranje krmilnika na tovarniške nastavitve, idr.

Program blocks: predstavlja programske bloke, pod katere spadajo organizacijski bloki (**Organization blocks - OB**), funkcijski bloki (**Function blocks - FB**), funkcije (**Functions - FC**) ter podatkovni bloki (**Data blocks - DB**). S pomočjo programskih blokov zgradimo strukturo programa, pri čemer vsak blok pripada določenem programskem nivoju od najvišjega - OB, preko srednjega - FB, do najnižjega - FC. S klikom na **Add new block** v projekt dodamo nov programski blok.

PLC tags: predstavlja seznam tabel spremenljivk (**PLC tags**), ki jih uporabljamo pri izgradnji programa. V seznam spremenljivk je potrebno vnesti spremenljivke, ki jih bomo uporabljali pri programiranju (vhode, izhode in notranje pomožne spremenljivke - stanja in zastavice).

Hiter pregled programskega okolja TIA Portal razkriva, da le-ta vsebuje še mnogo drugih gradnikov, ki pa jih za izvedbo vaj ne bomo potrebovali.

Pomen in ustvarjanje programskih blokov

Pri programiranju krmilnika je potrebno ukaze oz. programsko kodo zapisati v programske bloke. V okolju TIA Portal imamo na izbiro tri možne bloke, v katere zapisujemo programsko kodo, in ki se med seboj ločijo po namenu in hierarhiji. Programski jezik, ki je uporabljen v posameznem programskem bloku, določi uporabnik in je lahko lestvični diagram - LD, strukturiran tekst - SCL ali funkcijski bločni diagram - FBD. Vsi programski bloki so medsebojno kompatibilni, neglede na uporabljen programski jezik.

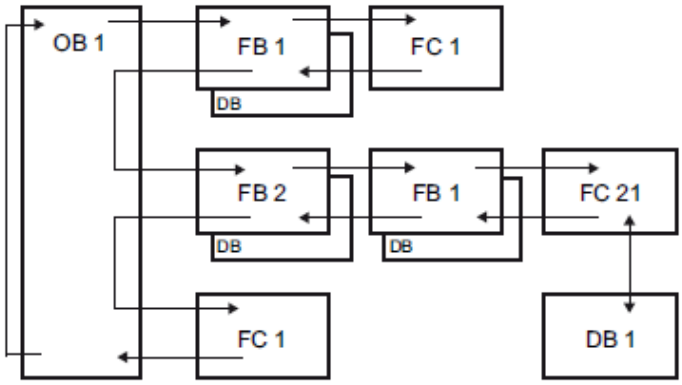
- **Organizacijski bloki (OB)** predstavljajo najvišji programski nivo in so vmesnik med operacijskim sistemom krmilnika ter uporabniškim programom. Organizacijski bloki skrbijo za pravilno delovanje krmilnika in se uporabljajo za izvajanje določenih akcij:
  - pri zagonu krmilnika,
  - pri cikličnem izvajanju,
  - kadar se pojavi napaka,
  - kadar se pojavi prekinitev.

  Samo delovanje krmilnika je odvisno od več kot 50 organizacijskih blokov, katerih delovanje presega obseg tega predmeta.
  Za izdelavo programske kode, kot jo bomo potrebovali za izvedbo vaj, je pomembno, da projekt vsebuje le en organizacijski blok, in sicer OB1. OB1 predstavlja organizacijski blok, ki ga krmilnik izvaja ciklično in vsebuje celotno strukturo uporabniškega programa, vključno z vsemi funkcijskimi bloki ter funkcijami. OB1 je običajno že ustvarjen, ko odpremo nov projekt.

- **Funkcijski bloki (FB)** predstavljajo naslednji programski nivo in vsebuje kodo, ki se izvaja s klicem iz drugih programskih blokov (OB, FB ali FC). Ena od lastnosti funkcijskih blokov je tudi, da imajo spomin. To pomeni, da se trenutne vrednosti posameznih spremenljivk ohranjajo pri vsakem klicu za izvajanje programa. Funkcijski bloki običajno vsebujejo glavno kodo celotnega programa. Nov funkcijski blok dodamo s klikom na izbran krmilnik ter **Program blocks** $>$ **Add new block**. V oknu, ki se odpre, vpišemo želeno ime bloka ter izberemo programski jezik, v katerem bo blok programiran.

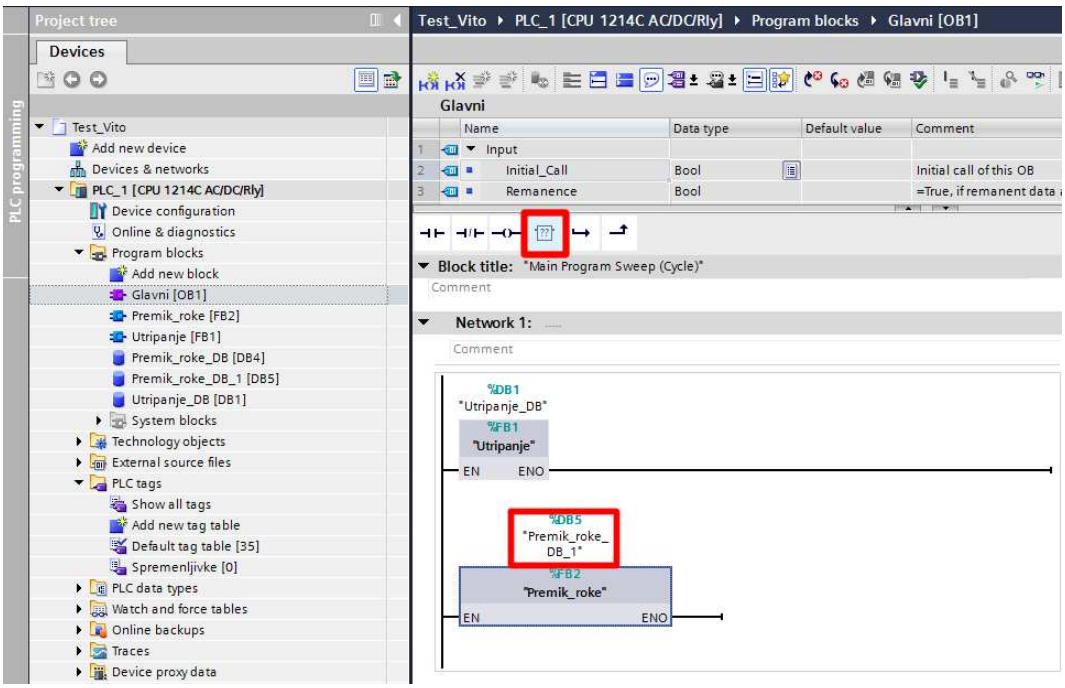
- **Funkcije (FC)** predstavljejo najnižji programski nivo in običajno vsebujejo podrutine oz. funkcije, ki jih kličejo ostali programski bloki (OB, FB ali FC). Funkcija, za razliko od funkcijskega bloka, nima spomina, kar pomeni, da se vrednosti spremenljivk pri vsakem klicu funkcije ponastavijo na začetne. Novo funkcijo dodamo s klikom na **Program blocks** $>$ **Add new block**. V oknu, ki se odpre, vpišemo želeno ime funkcije ter izberemo programski jezik, v katerem bo funkcija programirana.

Strukturiranje programa v posameznem projektu se običajno izvede na podoben način, kot prikazuje slika 1.8. Pri tem velja poudariti, da je lahko celotna programska koda krmilnika zapisana zgolj v organizacijskem bloku OB1, vendar pa je zaradi večje preglednosti programa bolje, da se program strukturira v različne FB-je in po potrebi FC-je.



Slika 1.8: Tipična struktura programa

Ko je nov programski blok, npr. FB, ustvarjen, ga moramo dodati na ustrezno mesto v programski blok, iz katerega bo klican, npr. OB1. To storimo tako, da odpremo blok OB1 ter s klikom na ikono dodamo nov blok v program. S klikom na ime dodanega bloka iz menija izberemo, kateri blok želimo dodati v program (slika 1.9).



Slika 1.9: Dodajanje programskih blokov (FB) v organizacijski blok OB1

Vnos lestvičnega diagrama

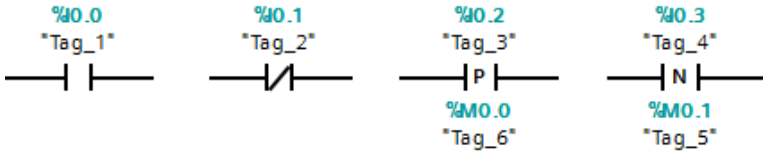
Z do sedaj opisanimi koraki pripravimo projekt do faze, ko lahko pričnemo z vnašanjem programa v obliki lestvičnega diagrama. Program sestavljamo v grafičnem načinu, in sicer tako, da izbiramo in povezujemo elemente lestvičnega diagrama. Ko imamo ustvarjen FB ali FC, dvakrat kliknemo na želeni programski blok, v katerega bomo vnašali diagram. Odpre se okno, v katerega rišemo diagram tako, da iz orodne vrstice izbiramo potrebne elemente.

Za izvedbo vaje potrebujemo naslednje elemente lestvičnega diagrama:

- branje vhodne logične spremenljivke oz. kontakte (normalno - delovni kontakt, negirano - mirovni kontakt, zaznavanje spremembe stanja - detektor prehoda),
- postavitev stanja izhodne logične spremenljivke oz. tuljave (normalna, negirana, postavitev stanja na 1 (SET), postavitev stanja na 0 (RESET)),
- časovnike,
- števce,
- povezave.

Branje logične spremenljivke

Simboli za normalno branje, negirano branje, detekcijo pozitivnega in negativnega prehoda so prikazani na sliki 1.10.

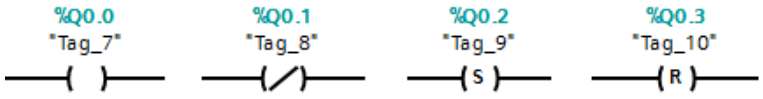


Slika 1.10: Simbol za branje vrednosti - kontakt

Za dodajanje kontakta v program v orodni vrstici pritisnemo ⊣⊢ in povlečemo s kurzorjem na mesto, kjer želimo postaviti element. Privzeta oblika elementa je vedno delovni kontakt. Z dvoklikom na element lahko kontakt spremenimo v mirovnega, detektor pozitivnega prehoda ali detektor negativnega prehoda. Pri detektorjih prehoda moramo paziti, da se spominski biti (%Mx.x) ne ponavljajo. Z zaporednim in vzporednim povezovanjem kontaktov sestavljamo logične pogoje za postavitev stanja notranjih in izhodnih logičnih spremenljivk krmilnika.

Postavitev stanja logične spremenljivke

Simboli normalne tuljave, negirane tuljave, tuljave za postavitev stanja na 1 (SET) in tuljave za postavitev stanja na 0 (RESET) so prikazani na sliki 1.11.

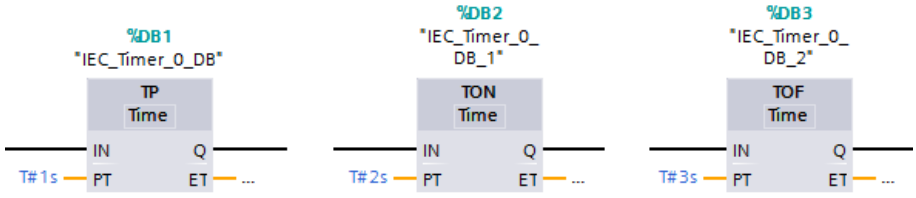


Slika 1.11: Simbol za pisanje vrednosti - tuljava

Za dodajanje tuljave v program v orodni vrstici pritisnemo in povlečemo s kurzorjem na mesto, kjer želimo postaviti element. Z dvoklikom na element, lahko izbiramo med normalno ali negirano tuljavo ter postavitvijo na 1 (SET) ali postavitvijo na 0 (RESET).

Časovniki

V skladu s standardom IEC 61131-3 imamo tri vrste časovnikov s simboli, kot jih prikazuje slika 1.12.

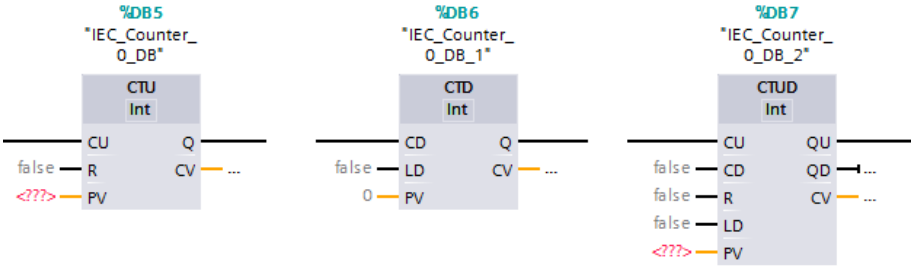


Slika 1.12: Časovniki

Časovnik lahko v diagram dodamo tako, da v desni stranski vrstici, v zavihku **Instructions** > **Basic instructions** > **Timer operations** kliknemo na želen časovnik, npr. **TP**, **TON** ali **TOF** in ga s kurzorjem povlečemo na želeno mesto. Odpre se okno **Call options**, kjer vpišemo želeno ime časovnika in potrdimo s klikom na **OK**. S tem se definicija časovnika pojavi tudi v stranskem meniju izbranega krmilnika (**System blocks** > **Program resources**), kjer jo lahko po potrebi spreminjamo. Parameter **PT** (preset time) privzeto vnašamo v milisekundah (vnos številke 2000 pomeni 2000 ms oz. 2 s), spremenimo pa ga z dvoklikom na <??? >.

Števci

V skladu s standardom IEC 61131-3 imamo tri vrste števcev s simboli, kot jih prikazuje slika 1.13.



Slika 1.13: Števci

Števec lahko v diagram dodamo tako, da v desni stranski vrstici, v zavihku **Instructions** > **Basic instructions** > **Counter operations** kliknemo na želen časovnik, npr. **CTU**, **CTD** ali **CTUD** in ga s kurzorjem povlečemo na želeno mesto. Odpre se okno **Call options**, kjer vpišemo želeno ime števca in potrdimo s klikom na **OK**. S tem se definicija števca pojavi tudi v stranskem meniju izbranega krmilnika (**System blocks** > **Program resources**), kjer jo lahko po potrebi spreminjamo. Parameter **PV** (preset value) spremenimo z dvoklikom na $<???>$, definiran pa je kot celoštevilska konstanta.

Povezave

V orodni vrstici s klikom in povlekom ikon ↪ | ⮥ dodajamo nove in povezujemo obstoječe povezave.

Primer: Prikazana je izdelava programa, ki bo zagotavljal spodaj opisano delovanje krmilnika:

S pritiskom tipke START, kateri pripada spremenljivka *Tipka_1*, prične utripati lučka STOP (*Luc_1*), in sicer pri vsakem utripu je lučka 1 s prižgana in 1 s ugasnjena. S tipko RESET, kateri pripada spremenljivka *Tipka_2*, lahko utripanje predčasno izključimo.

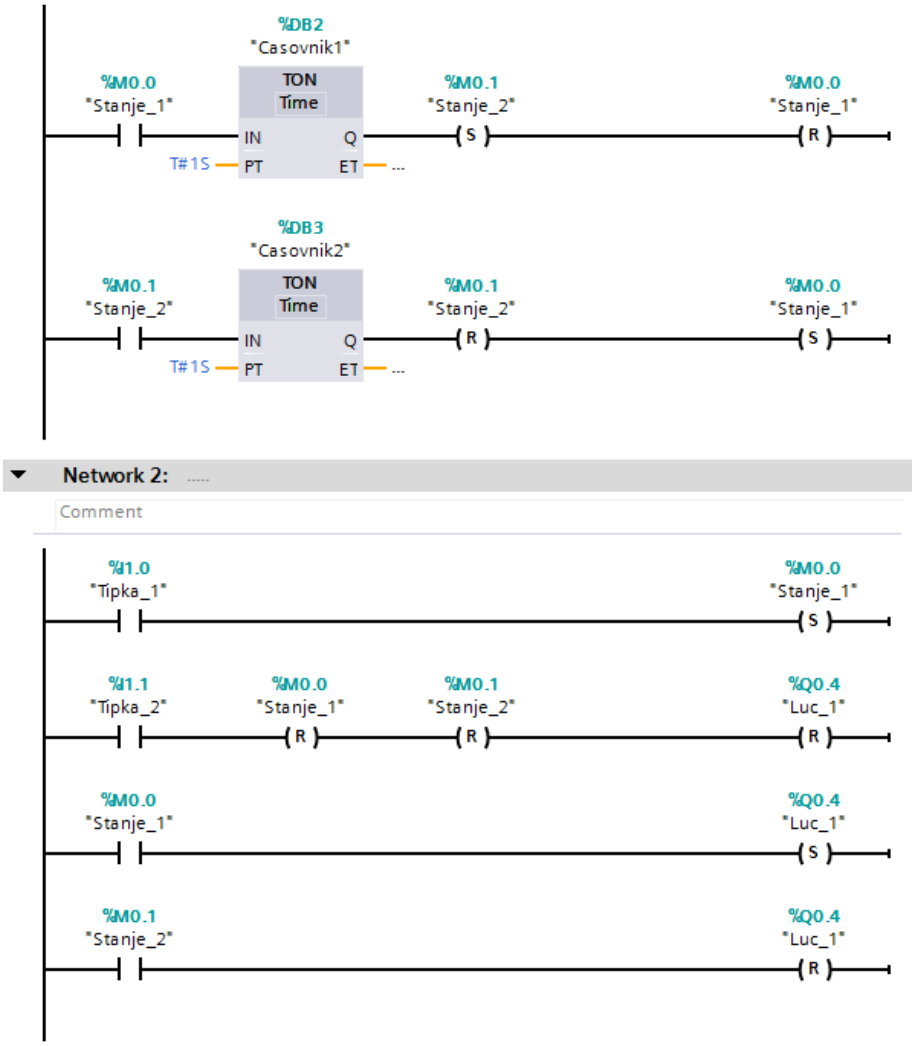
Izdelava programa:

- Odpremo nov projekt in ga poimenujemo.
- V levem stranskem meniju dodamo nov krmilnik (**Add new device**) in potrdimo z **OK**.
- Z dvoklikom na ikono  na sliki PLK-ja prikličemo okno z omrežnimi nastavitvami krmilnika, v katerega vpišemo IP-naslov krmilnika.
- Z dvoklikom na izbran krmilnik ter **PLC tags** > **Add new tag table** ustvarimo novo tabelo spremenljivk in jo poimenujemo.
- Odpremo ustvarjeno tabelo spremenljivk in začnemo vnašati potrebne spremenljivke. Tipki sta vhodni spremenljivki, spremenljivka, povezana z lučko, pa izhodna (slika 1.14). Potrebujemo še dve interni spremenljivki *Stanje_1* in *Stanje_2*, ki sta potrebni za izvedbo utripanja.

	Name	Data type	Address	Retain	Visibl...	Acces...
1	Luc_1	Bool	%Q0.7	☐	☑	☑
2	Tipka_1	Bool	%I1.0	☐	☑	☑
3	Tipka_2	Bool	%I1.1	☐	☑	☑
4	Stanje_1	Bool	%M0.0	☐	☑	☑
5	Stanje_2	Bool	%M0.1	☐	☑	☑

Slika 1.14: Seznam spremenljivk

- V levem stranskem meniju dodamo nov programski blok. To storimo s klikom na izbran krmilnik ter **Program blocks** > **Add new block**. Tip bloka naj bo **function block - FB**, njegovo ime *Utripanje*, izbran programski jezik pa **Ladder diagram**. Pričnemo s sestavljanem lestvičnega diagrama, kot prikazuje slika 1.15



Slika 1.15: Lestvični diagram funkcijskega bloka *Utripanje*

- Odpremo glavni organizacijski blok **OB1** in vanj dodamo prej ustvarjeni funkcijski blok *Utripanje*.
- Projekt shranimo, prevedemo in naložimo na krmilnik.

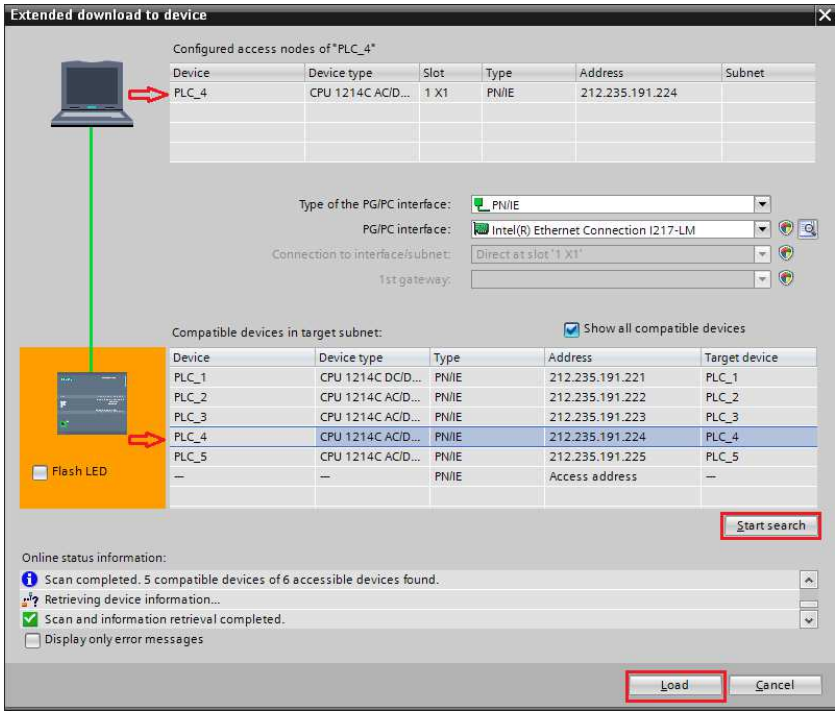
1.2.3 Prevajanje programa

Program je potrebno prevesti v obliko, ki je primerna za prenos na krmilnik. To lahko storimo na dva načina. Prvi način je, da v levem stranskem meniju z desnim gumbom kliknemo na izbran krmilnik ter izberemo **Compile** > **Hardware and software**. Druga, hitrejša možnost je, da v orodni vrstici kliknemo na ikono  . Prevajanje

hkrati odkriva napake, ki jih posebej označi in si jih lahko tudi ogledamo, če dvakrat kliknemo na vrstico z opisom napake. Če prevajanje javi, da v programu ni napak, lahko prenesemo program na krmilnik.

1.2.4 Prenos programa na krmilnik

Program lahko prenesemo na krmilnik na dva načina. Prvi način je, da v levem stranskem meniju z desnim gumbom kliknemo na izbrani krmilnik ter izberemo **Download to device** > **Hardware and software**. Druga, hitrejša možnost je, da v orodni vrstici kliknemo na ikono  . Odpre se okno **Extended download to device**, kjer s klikom na tipko **Start search** dobimo seznam priključenih krmilnikov na omrežje. Kliknemo na izbrani krmilnik (isti IP) in pritisnemo **Load** (slika 1.16). Pred preizkušanjem programa na krmilniku moramo preveriti ali je krmilnik v stanju *Run* (zelena - *Run* oz. oranžna - *Stop* lučka na krmilniku).



Slika 1.16: Prenos programa na krmilnik

1.2.5 Opazovanje stanj programa

Izvajanje programa na krmilniku opazujemo tako, da odpremo okno, v katerem je program, ter začnemo opazovanje s klikom na ikono  .

1.3 Izvedba osnovnih logičnih in števno-časovnih krmilij

1.3.1 Osnovna krmilja

Sestavite enostavne logično-sekvenčne programe, ki so opisani v nadaljevanju in s katerimi ilustriramo uporabo osnovnih gradnikov lestvičnega diagrama.

Vse programe napišite v okviru enega projekta, ki naj vsebuje en organizacijski blok ter več funkcijskih blokov, ki jih po potrebi vključujte v organizacijski blok. Vsak funkcijski blok, ki ga zahteva vaja, naj bo samostojen programski modul. Pri testiranju programov nato z nastavitvami pripadnosti organizacijskemu bloku določite, kateri programski modul se bo prevajal, naložil v krmilnik ter izvajal, in kateri ne.

Programe preizkusite tako, da vsakega posebej naložite v krmilnik in preizkusite, če se delovanje krmilnika ujema z zahtevami. Vhodne signale generiramo s pritiskom tipk, izhodne signale pa opazujemo na priključnih sponkah, ki so opremljene s svetlečimi diodami, ter na lučkah na komandnem pultu.

Narišite pripadajoče rešitve.

Programi:

1. Start/stop vezje: ob pritisku tipke START (%I1.0) naj se izhod LUC (%Q1.1) postavi na 1 in ostane na 1, ko tipko spustimo. Ob pritisku tipke RESET (%I1.1) naj se postavi na 0 in ostane na 0, ko tipko spustimo. Sestavite dve različici programa: brez uporabe tuljav SET in RESET in z uporabo teh dveh vrst tuljav.

Rešitev 1. naloge (brez SET/RESET):

Rešitev 1. naloge (s SET/RESET):

2. Zakasnitev vklopa s časovnikom: 4 s po pritisku tipke START se prižge lučka na izhodu %Q1.1. Lučka se prižge ne glede na to, ali tipko držimo dalj časa ali le kratek čas. S pritiskom tipke RESET lučko ugasnemo. Če tipko RESET pritisnemo še pred iztekom zakasnitve, se lučka sploh ne prižge.

Rešitev 2. naloge:

3. Vklop za določen čas: Ob pritisku tipke START se prižge lučka na izhodu %Q1.1 in ostane prižgana 3 s. Lučka mora po 3 s ugasniti, tudi če tipko START držimo dalj časa. S pritiskom tipke RESET lahko lučko predčasno ugasnemo, s pritiskom tipke START pa ponovno prižgemo za 3 s.

Rešitev 3. naloge:

4. Utripanje: Ob pritisku tipke START pričneta izmenično utripati lučki na izhodih %Q0.7 in %Q1.0. Vsaka naj bo 1 s prižgana in 1 s ugasnjena. Utripanje lahko ugasnemo s tipko RESET, s tipko START pa ga nato lahko ponovno vključimo.

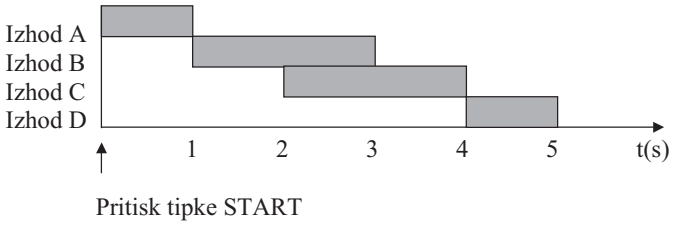
Rešitev 4. naloge:

1.3.2 Enostavno sekvenčno krmilje

Sestavite sekvenčno krmilje, ki ustreza naslednjim zahtevam:

Po pritisku tipke START naj se izhodi A (%Q0.6), B (%Q0.7), C (%Q1.0) in D (%Q1.1) vklapljajo tako, kot prikazuje časovni diagram na sliki 1.17. Celotna sekvenca vklopov naj se ponovi trikrat. Po koncu naj bo možen ponoven vklop s tipko START, med trajanjem sekvence pa tipka START ne sme vplivati na delovanje.

Izpišite program na tiskalnik in dodajte izpis kot prilogo k rezultatom vaje.



Slika 1.17: Zaporedje vklopov

Laboratorijska vaja št. 2

Programiranje koračnih krmilij z lestvičnimi diagrami

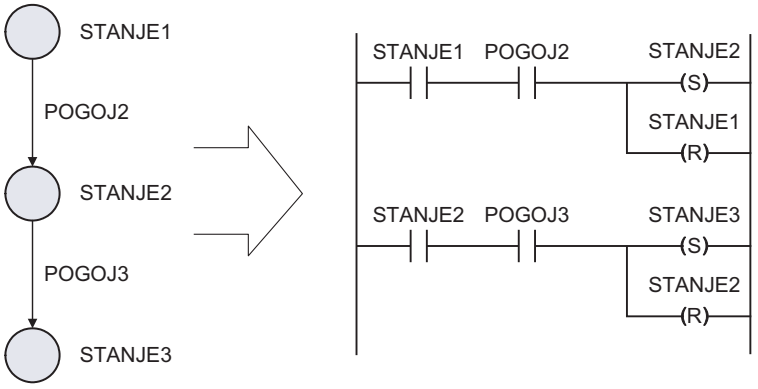
Pri vaji bomo spoznali načrtovanje logičnega in sekvenčnega vodenja z lestvičnimi diagrami. Načrtovali bomo vodenje laboratorijskega modularnega proizvodnega sistema z uporabo programskega orodja Siemens TIA Portal. Programsko orodje poleg ostalih programskih jezikov omogoča tudi programiranje logičnih krmilnikov z lestvičnimi diagrami (Ladder Diagram - LD) po standardu IEC 61131-3. Z osnovami lestvičnih diagramov ste se že srečali pri 1. vaji, pri tej vaji pa boste vaše znanje še nadgradili.

2.1 Prehajanje stanj v lestvičnem diagramu

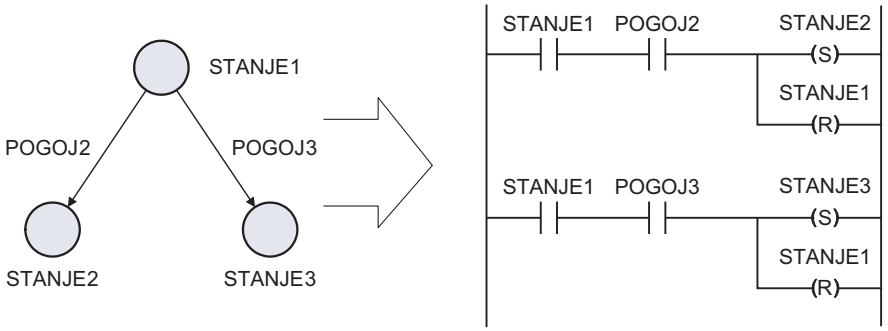
Značilnost koračnih krmilij je, da je odziv krmilja na spremembo vhodov odvisen od tega, kaj se je v krmilju prej dogajalo. Krmilje ima notranja stanja, ki pomnijo preteklo dogajanje. Za pravilno delovanje krmilja je pomembno, da zagotovimo pravilno prehajanje stanj.

Pri nekaterih jezikih za programiranje PLK-jev, kot npr. sekvenčnem funkcijskem diagramu, je prehajanje stanj določeno s povezavo korakov in prehodov. Pri lestvičnem diagramu je prehajanje stanj težje razvidno iz same grafične sheme. Klasičen pristop k izvedbi koračnih krmilij z lestvičnimi diagrami je uporaba pomožnih logičnih spremenljivk (t.i. markerjev), s katerimi si program zapomni, da je neko operacijo že izvedel, in to upošteva pri pogojih za naslednje operacije.

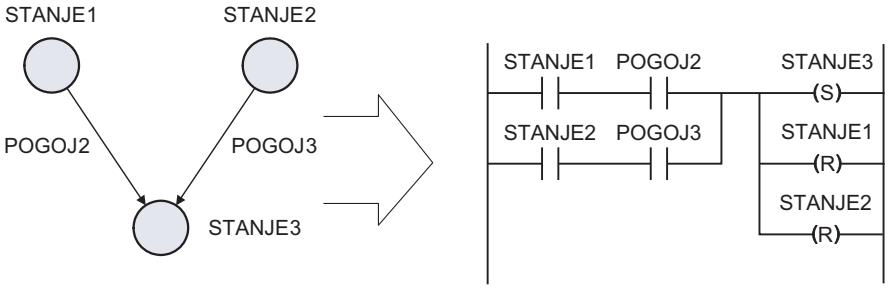
Tak pristop deluje, vendar pa so programi zelo nepregledni in je v njih težko odkriti morebitne napake. Bolj sistematično se izvedbe koračnega krmilja lotimo tako, da najprej narišemo diagram prehajanja stanj krmilja. Vsakemu stanju priredimo notranjo spremenljivko krmilnika, nato pa programiramo pogoje za prehode med temi stanji in pripadajočo logiko za postavljanje in brisanje stanj. V drugem delu programa programiramo še krmilne akcije, ki se izvajajo glede na to, katero stanje je aktivno. Precejšen del diagrama je namenjen zgolj zagotavljanju pravilnega prehajanja stanj. Tudi tu torej uporabimo pomožne logične spremenljivke, glavna razlika je v tem, da ločimo del programa, ki zagotavlja pravilno prehajanje stanj, in del programa, kjer pro-



Slika 2.1: Pretvorba zaporedja stanj v LD - enostavno zaporedje



Slika 2.2: Pretvorba zaporedja stanj v LD - izstop iz stanja v več smereh



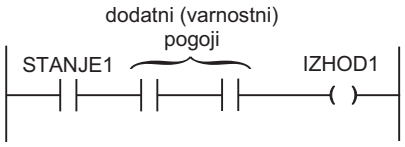
Slika 2.3: Pretvorba zaporedja stanj v LD - vstop v stanje iz več smeri

gramiramo operacije, ki se v določenem stanju izvajajo. Na ta način je programiranje lažje, program pa preglednejši.

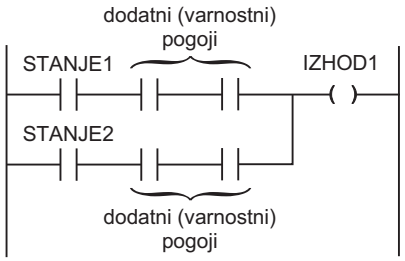
Na slikah 2.1 do 2.3 je prikazanih nekaj primerov, kako posamezne dele diagrama prehajanja stanj pretvorimo v prečke lestvičnega diagrama.

Na slikah 2.4 in 2.5 je prikazano, kako povežemo izhode z notranjimi stanji krmilja. Pri tem je priporočljivo dodati v pogoj za vklop izhoda poleg samega stanja še dodatne pogoje, ki preprečujejo, da bi v procesu prišlo do škodljivih ali celo nevarnih situacij.

Dodatni pogoji so npr. signali z zaščitnih ali končnih stikal. Na ta način zagotovimo varno obratovanje procesa tudi v primeru, ko pride do napake pri programiranju logike prehajanja stanj.



Slika 2.4: Postavljanje izhodov - izhod je aktiven samo v enem stanju



Slika 2.5: Postavljanje izhodov - izhod je aktiven v dveh stanjih

Iz slik je razvidno, da notranja stanja v programu spreminjamo glede na kombinacijo trenutnih stanj in vhodov, ki predstavljajo pogoje za prehod na naslednji korak. Izhodi pa so odvisni od notranjih stanj in vhodov, ki predstavljajo dodatne varnostne omejitve. Pomembno se je držati tudi naslednjih pravil:

- pogoji za spremembo stanja morajo vedno vključevati trenutno stanje; stanja postavljamo zadržano (SET/RESET oz. tuljava S/R),
- izhodi so odvisni od kombinacije notranjih stanj in jih praviloma ne postavljamo zadržano (v LD navadna tuljava namesto S/R).

Postopek izvedbe koračnih krmilij z lestvičnimi diagrami bi lahko strnili v naslednje točke:

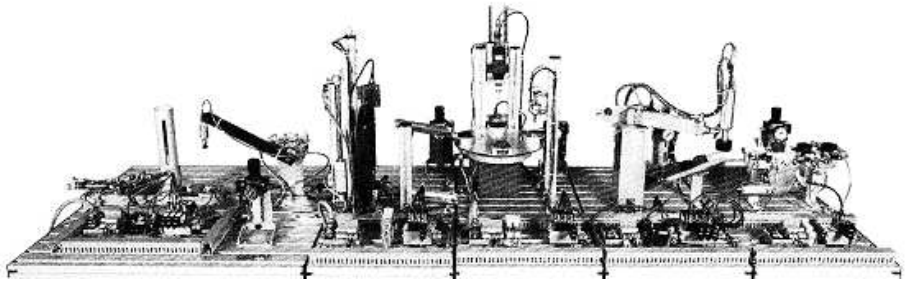
1. Narišemo diagram prehajanja stanj.
2. Vsakemu stanju v diagramu priredimo notranje stanje – logično spremenljivko oz. marker.
3. Za vsako notranje stanje poiščemo pogoje za vstop v stanje in sestavimo prečko lestvičnega diagrama, ki postavlja stanje (SET).
4. Pri vsakem stanju v pogoje za vstop vključimo aktivnost predhodnega stanja in v izvršni del prečke dodamo brisanje predhodega stanja (RESET).

5. Z notranjimi stanji sestavimo pogoje za aktivnost izhodnih signalov.

Prednost pristopa s stanji je manjša občutljivost na šumne signale in motnje: prehod v novo stanje se izvede le ob določenih pogojih; na ostale spremembe krmilje ne odgovarja. Enostavnejše je tudi odkrivanje napak, popravljanje programa pa lažje. Slabost tega pristopa pa je, da so programi obsežnejši.

2.2 Modularni proizvodni sistem

Model modularnega proizvodnega sistema (MPS) predstavlja fleksibilen sistem za prikaz delovanja komponent, enot, postaj in kompletnega postrojenja proizvodnega sistema. Predstavlja industrijsko linijo s petimi postajami, od podajanja in kontrole obdelovancev, obdelave, transporta do skladiščenja (slika 2.6). Te funkcije so v industriji najbolj pogoste. Uporabljene komponente (senzorji, aktuatorji, ožičenje, krmilniki) so enake kot v industriji.



Slika 2.6: Linija modularnega proizvodnega sistema

2.2.1 Delovanje modularnega proizvodnega sistema

Obdelovanci posamezno vstopajo v sistem in potujejo od podajalne postaje preko postaje za razpoznavanje materiala in barve ter kontrolo dimenzij, postaje za obdelavo, na kateri je vrtalna enota in enota za kontrolo kvalitete obdelave, ter transportne postaje do postaje za sortiranje in skladiščenje.

Podajalni cilinder potisne obdelovanec, ki je padel iz vhodnega skladišča, v prevzemni položaj. Pnevmatski zasučni pogon zavrti ročico s prijemalom v prevzemni položaj ob vhodnem skladišču, kjer prijemalo s sesalno šobo prisesa obdelovanec. Obenem mora podajalni cilinder sprostiti obdelovanec, zato se začne pomikati nazaj. Zasučni pogon prestavi obdelovanec na ploščad dvigala kontrolne postaje in izklopi sesalno šobo na prijemalu. Izvede se razpoznavanje materiala (ali je obdelovanec kovinski, oziroma plastičen rdeče ali črne barve). Razpoznan obdelovanec dvigalo dvigne

nad vrtilno mizo naslednje postaje, kjer se preveri njegova višina. Če je obdelovanec primerne višine, ga potisni cilinder potisne na drčo, po kateri obdelovanec zdrsne na vrtilno mizo obdelovalne postaje. Če je obdelovanec previsok ali prenizek, se dvigalo spusti v začetni položaj, potisni cilinder pa potisne obdelovanec v skladišče odpada. Po prejemu obdelovanca se vrtilna miza zavrti in prenese obdelovanec do vrtalne enote. Tu se obdelovanec vpne z vpenjalnim cilindrom in vanj se izvrta luknja. Po vrtanju se vrtilna miza zavrti naprej, da pride obdelovanec do enote za kontrolo kvalitete obdelave (ali je izvrtina ustrezna ali ne). Vrtilna miza prestavi nato obdelovanec do prevzemnega mesta. Roka transportne postaje se pomakne k prevzemnemu mestu, se iztegne, spusti prijemalo s sesalno šobo in prisesa obdelovanec, ga dvigne in se pokrči. Če je obdelovanec pravilno obdelan, se prenese na tekoči trak za sortiranje in skladiščenje. Če je pri kontroli kvalitete ugotovljeno, da je obdelovanec brez izvrtine, ga roka prenese in spusti v drčo za odpad. Glede na rezultat razpoznavanja materiala (kovina, rdeča ali črna plastika) se na tekočem traku vklopi ustrezen cilinder za usmerjanje obdelovancev v določeno skladišče.

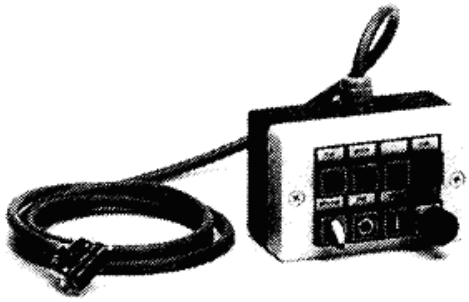
Z vgrajenimi krmilnimi elementi je mogoče realizirati različne načine delovanja sistema. Na vajah ne bomo realizirali popolnoma funkcionalne proizvodne linije, temveč bomo preizkusili le nekaj osnovnih operacij v sistemu.

2.2.2 Vodenje sistema in krmilni pult

Na vajah uporabljamo za vodenje pet krmilnikov, s katerimi krmilimo vsako postajo posebej. Uporabljamo programirljive logične krmilnike Siemens S7-1200, ki jih programiramo s programskim paketom Siemens TIA portal.

Za dajanje ukazov je na vsak krmilnik priključen krmilni pult (slika 2.7). Vsebuje 5 tipk, od katerih so 3 osvetljene, izbirno stikalo, stikalo za izklop v sili in svetlobni indikator:

- START (osvetljena tipka, zelena) - startna tipka
- RESET (osvetljena tipka, zelena) - tipka za reset
- osvetljena tipka, zelena - prosto nastavljiva funkcija
- AUTO / MAN (izbirno stikalo) - avtomatski / ročni način delovanja
- STOP (tipka, rdeča) - zaustavitev po končanem ciklu
- QUIT. (tipka, zelena) - razveljavitev izklopa v sili
- EMERGENCY STOP - izklop v sili
- svetlobni indikator - prosto nastavljiva funkcija



Slika 2.7: Krmilni pult

2.3 Varnostni napotki

Splošno

Študenti lahko delajo na postajah MPS le pod nadzorom asistenta. Upoštevajte navodila asistenta, posebno še napotke za varnost.

Elektrika

Priključitev in izključitev električnega napajanja je dovoljena le v mirujočem stanju vseh komponent. Uporabljamo samo nizko napetost, maks. 24 V.

Pnevmatika

Največji dovoljen tlak je 8 barov. Vključite napajanje s stisnjenim zrakom šele, ko so vzpostavljene vse cevne povezave, ki morajo biti tudi varne. Pri priklopu stisnjenega zraka bodite pozorni, kajti cilindri se lahko samodejno sunkovito premaknejo.

Mehanika

Vsi elementi morajo biti trdno pritrjeni na nosilno ploščo. Ročno poseganje na posamezne postaje je dovoljeno le takrat, kadar je postaja izključena.

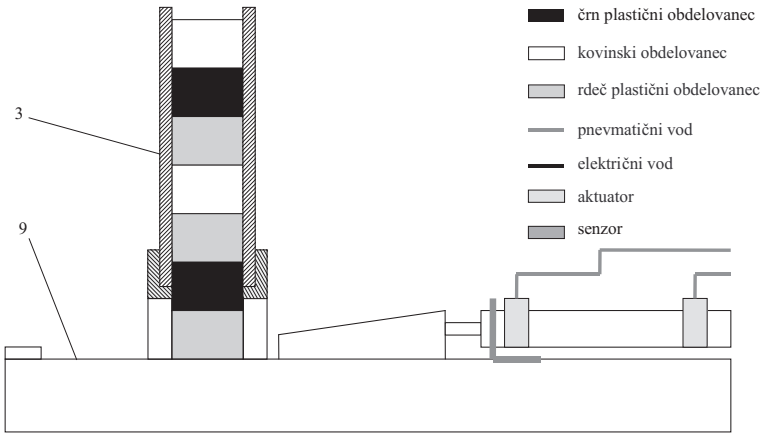
2.4 Izvedba vodenja modularnega proizvodnega sistema z lestvičnim diagramom

Napišite in preizkusite program za programirljivi logični krmilnik, ki bo krmilil eno od postaj modularnega sistema tako, kot določa funkcionalni opis za to postajo (podan v nadaljevanju).

Za programiranje uporabite **lestvični diagram**. Sestavite osnovno zaporedje prehajanja stanj v lestvičnem diagramu upoštevajoč ustrezne prehodne pogoje, ga dopolnite z ustreznimi akcijami in preizkusite na napravi. Izpišite program na tiskalnik in dodajte izpis kot prilogo k rezultatom laboratorijskih vaj.

2.4.1 Krmiljenje prve postaje

Sliki 2.8 in 2.9 prikazujeta podajalno enoto prve postaje v laboratorijskem modelu modularnega proizvodnega sistema. Naloga enote je podajanje obdelovancev iz vhodnega skladišča.

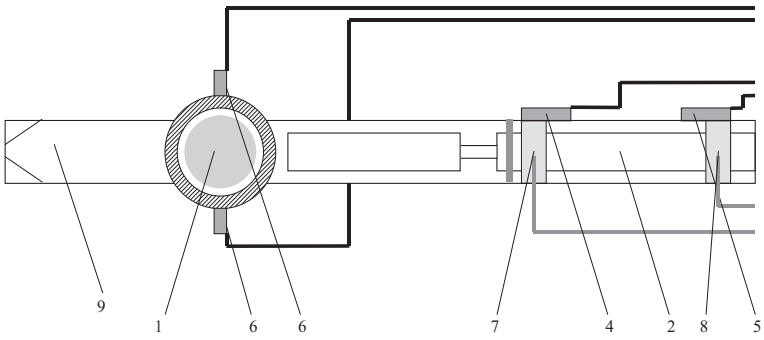


Slika 2.8: Podajalna enota

Pomen oznak na slikah opisuje tabela 2.1.

Funkcionalni opis

S pritiskom tipke START se postaja postavi v obratovalno stanje, kar signalizira signalna lučka LUC_START. Če je v skladišču vsaj en obdelovanec, ga podajalni cilinder potisne iz skladišča in se vrne v mirovni položaj. Podajalna enota nato počaka 5 s, v tem času mora naslednja enota prevzeti obdelovanec. Ko poteče 5 s, potisne cilinder iz skladišča naslednji obdelovanec, če je ta prisoten, in postopek se ponavlja.



Slika 2.9: Podajalna enota - pogled z vrha

Tabela 2.1: Pomen oznak na slikah podajalne postaje

1	obdelovanci (kovinski, plastični rdeči in plastični črni)
2	pnevmatski cilinder
3	skladišče
4	S1B2 podajalni cilinder spredaj (bližinsko stikalo)
5	S1B1 podajalni cilinder zadaj (bližinsko stikalo)
6	S1S1 obdelovanec prisoten v skladišču (optični senzor)
7	povlek podajalnega cilindra (A1Y1 = 0)
8	izteg podajalnega cilindra (A1Y1 = 1)
9	prevzemni položaj

S pritiskom tipke STOP ustavimo delovanje enote, pri čemer se tekoči delovni cikel izvede do konca, nato pa se postaja ustavi in jo lahko ponovno poženemo s tipko START. Od pritiska tipke STOP do popolne zaustavitve lučka LUC_START utripa s periodo 1 s.

Če naslednja enota ne prevzame obdelovanca v predpisanem času, podajalni cilinder ne more potisniti novega obdelovanca iz skladišča. V tem primeru se mora delovanje ustaviti, ob tem pa mora utripati lučka LUC_EMPTY, kar je znak za alarm. Ponoven zagon postaje je možen tako, da pritisnemo tipko RESET, odstranimo oba obdelovanca in nato pritisnemo tipko START.

Če je postaja v obratovalnem stanju ter v skladišču ni novih obdelovancev, lučka LUC_EMPTY gori brez utripanja.

Uporabljeni vhodno/izhodni signali so zbrani v tabeli 2.2.

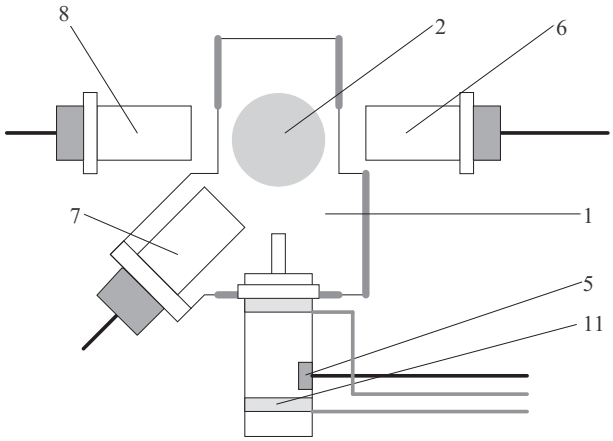
Tabela 2.2: Prireditvena tabela podajalne postaje

	oznaka na napravi	mirovno stanje	naslov v PLK
Vhodi krmilnika:			
Podajalni cilinder zadaj	S1B1	NO	%I0.0
Podajalni cilinder spredaj	S1B2	NO	%I0.1
Roka pri skladišču	S1S3	NO	%I0.2
Roka pri dvigalu	S1S4	NO	%I0.3
Obdelovanec v skladišču	S1S1	NO	%I0.5
Tipka Start	START	NO	%I1.0
Tipka Reset	RESET	NO	%I1.1
Tipka Potrditev	ACKN., OK/NOT OK	NO	%I1.2
Tipka Avotmatsko/Ročno	AUTO/MAN	NO	%I1.3
Tipka Stop	STOP	NO	%I1.4
Tipka Izhod	QUIT	NO	%I1.5
Izhodi krmilnika:			
Podajalni cilinder naprej	A1Y1	/	%Q0.0
Roka k skladiču	A1Y5	/	%Q0.1
Roka k dvigalu	A1Y4	/	%Q0.2
Izklop sesanja	A1Y3	/	%Q0.3
Vklop sesanja	A1Y2	/	%Q0.4
Signalna lučka Start	LUC_START	/	%Q0.6
Signalna lučka Reset	LUC_RESET	/	%Q0.7
Signalna lučka Acknowledge	LUC_ACKN	/	%Q1.0
Signalna lučka Empty	LUC_EMPTY	/	%Q1.1

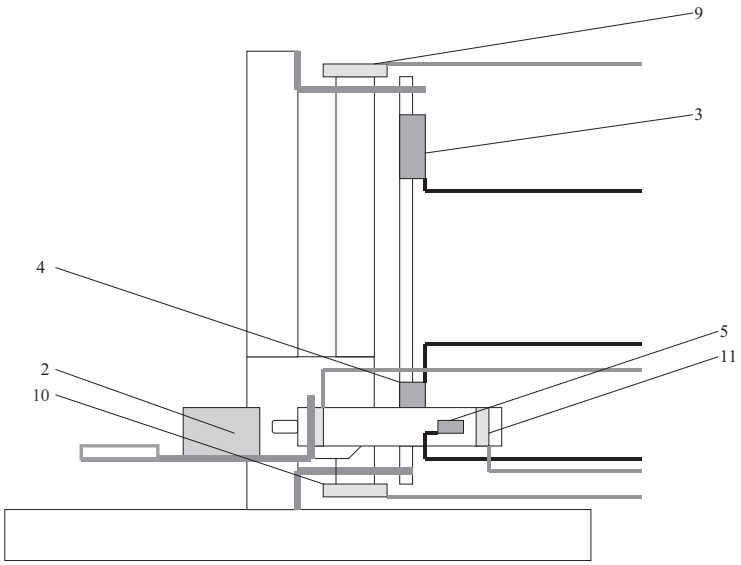
Legenda: mirovno stanje kontakta NZ - zaprto
NO - odprto

2.4.2 Krmiljenje druge postaje

Sliki 2.10 in 2.11 prikazujeta dvigalo na kontrolni postaji v laboratorijskem modelu modularnega proizvodnega sistema. Naloga dvigala je prenos obdelovancev z mesta testiranja na naslednjo postajo.



Slika 2.10: Ploščad dvigala in senzorji za razpoznavanje materiala



Slika 2.11: Dvigalo

Pomen oznak na slikah opisuje tabela 2.3.

Tabela 2.3: Pomen oznak na slikah kontrolne postaje

1	ploščad dvigala
2	obdelovanec
3	S2B1 dvigalo zgoraj
4	S2B2 dvigalo spodaj
5	S2B3 potisni cilinder na dvigalu zadaj
6	S2B5 indikator kovine (induktivni senzor)
7	S2B6 element na dvigalu (kapacitivni senzor)
8	S2B7 indikator barve (optični senzor)
9	A2Y1 spust dvigala
10	A2Y2 dvig dvigala
11	potisk potisnega cilindra na dvigalu naprej (A2Y3 = 1)

Funkcionalni opis

S pritiskom tipke START se postaja postavi v obratovalno stanje, kar signalizira signalna lučka LUC_START. Če je na dvigalu obdelovanec, ga dvigalo po zakasnitvi 1 s dvigne ter nato spusti cilinder merilnika višine na obdelovanec. Po dveh sekundah se merilnik vrne v mirovni položaj. Za tem potisni cilinder na ploščadi dvigala potisne obdelovanec proti naslednji postaji, in ko se potisni cilinder vrne v mirovno lego, se vrne v mirovno lego tudi dvigalo (mirovna lega dvigala je spodaj).

S pritiskom tipke STOP ustavimo delovanje enote, pri čemer se tekoči delovni cikel izvede do konca, nato pa se postaja ustavi in jo lahko ponovno poženemo s tipko START. Od pritiska tipke STOP do popolne zaustavitve lučka LUC_START utripa s periodo 1 s. Če je postaja v obratovalnem stanju ter dvigalo v mirovnem položaju in na njem ni obdelovanca, gori lučka LUC_EMPTY.

Uporabljeni vhodno/izhodni signali so zbrani v tabeli 2.4.

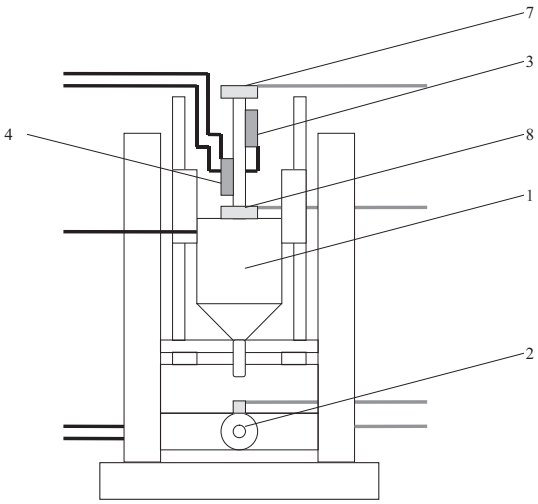
Tabela 2.4: Prireditvena tabela kontrolne postaje

	oznaka na napravi	mirovno stanje	naslov v PLK
Vhodi krmilnika:			
Obdelovanec na dvigalu	S2B6	NO	%I0.1
Dvigalo spodaj	S2B2	NO	%I0.3
Dvigalo zgoraj	S2B1	NO	%I0.4
Potisni cilinder v mir. legi	S2B3	NO	%I0.5
Merilnik višine spuščen	S2B4	NO	%I0.6
Tipka Start	START	NO	%I1.0
Tipka Reset	RESET	NO	%I1.1
Tipka Potrditev	ACKN., OK/NOT OK	NO	%I1.2
Tipka Avotmatsko/Ročno	AUTO/MAN	NO	%I1.3
Tipka Stop	STOP	NO	%I1.4
Tipka Izhod	QUIT	NO	%I1.5
Izhodi krmilnika:			
Spust dvigala	A2Y1	/	%Q0.0
Dvig dvigala	A2Y2	/	%Q0.1
Potisk cilindra na dvigalu	A2Y3	/	%Q0.2
Pomik višinomera navzdol	A2Y4	/	%Q0.3
Signalna lučka Start	LUC_START	/	%Q0.6
Signalna lučka Reset	LUC_RESET	/	%Q0.7
Signalna lučka Acknowledge	LUC_ACKN	/	%Q1.0
Signalna lučka Empty	LUC_EMPTY	/	%Q1.1

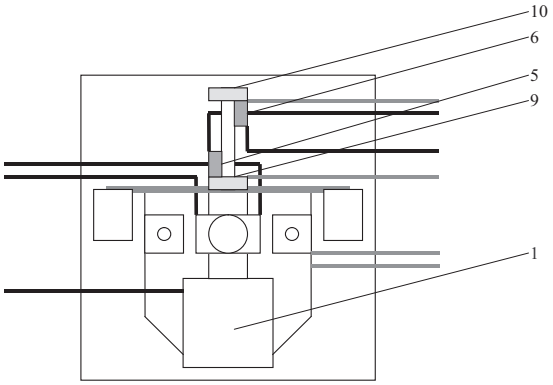
Legenda: mirovno stanje kontakta NZ - zaprto
NO - odprto

2.4.3 Krmiljenje tretje postaje

Sliki 2.12 in 2.13 prikazujeta vrtalno enoto na postaji obdelave v laboratorijskem modelu modularnega proizvodnega sistema.



Slika 2.12: Vrtalna enota



Slika 2.13: Pogled na vrtalno enoto z vrha

Pomen oznak na slikah opisuje tabela 2.5.

Tabela 2.5: Pomen oznak na slikah vrtalne enote

1	A3M1 vrtalni stroj
2	A3Y4 prijemalo obdelovanca
3	S3B1 vrtalni stroj dvignjen
4	S3B2 vrtalni stroj spuščen
5	S3B5 prijemalo iztegnjeno do konca
6	S3B6 prijemalo umaknjeno
7	A3Y1 spust vrtalnega stroja
8	A3Y2 dvig vrtalnega stroja
9	umik prijemala obdelovanca (A3Y4 = 1)
10	izteg prijemala obdelovanca (A3Y4 = 0)

Funkcionalni opis

S pritiskom tipke START se postaja postavi v obratovalno stanje, kar signalizira signalna lučka LUC_START. Če je na sprejemnem mestu vrtilne mize obdelovanec, se miza zavrti in prenese obdelovanec pod vrtalno enoto. Vključi se prijemalo, ki drži

obdelovanec. Nato se vključi vrtalnik in vrtalna enota se spusti do obdelovanca. Po dveh sekundah se vrtalnik vrne v mirovni položaj in izključi. Za tem cilinder prijemala popusti in postaja je pripravljena za sprejem naslednjega obdelovanca.

S pritiskom tipke STOP ustavimo delovanje enote, pri čemer se tekoči delovni cikel izvede do konca, nato pa se postaja ustavi in jo lahko ponovno poženemo s tipko START. Od pritiska tipke STOP do popolne zaustavitve lučka LUC_START utripa s periodo 1 s.

Če je postaja v obratovalnem stanju ter miza miruje in na sprejemnem mestu vrtilne mize ni obdelovanca, gori lučka LUC_EMPTY.

Opozorilo: Ko je aktivirano prijemalo, mize v nobenem primeru ni dovoljeno vrteti! V mirovnem stanju, ko ni tlaka oz. je signal A3Y4 enak 0, je prijemalo aktivirano (iztegnjeno)! **Pred vrtenjem mize je torej nujno postaviti signal A3Y4 na 1**!

Uporabljeni vhodno/izhodni signali so zbrani v tabeli 2.6.

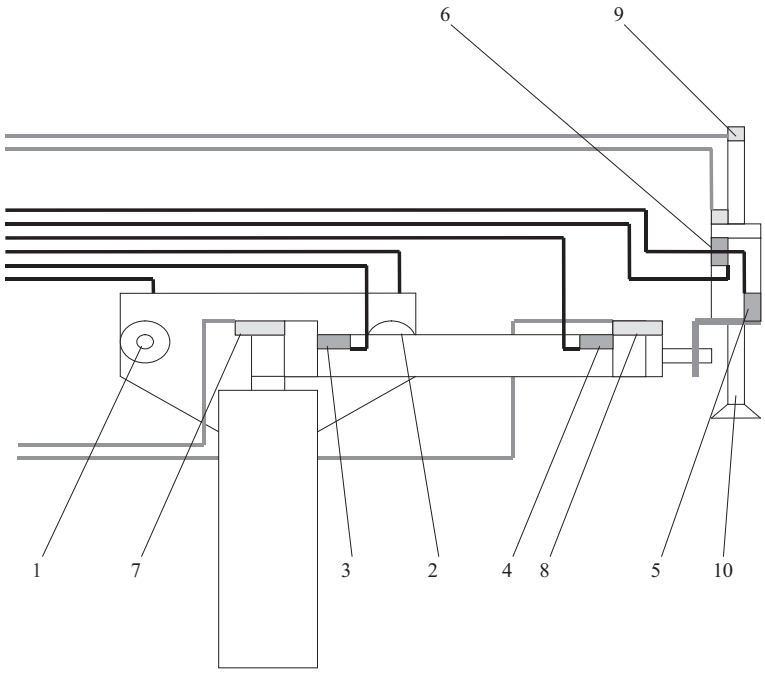
Tabela 2.6: Prireditvena tabela postaje obdelave

	oznaka na napravi	mirovno stanje	naslov v PLK
Vhodi krmilnika:			
Obd. na sprejemnem mestu	S3S1	NO	%I0.0
Senzor premika za 90°	S3B7	NO	%I0.1
Prijemalo umaknjeno	S3B6	NO	%I0.2
Pr. iztegnjeno do konca	S3B5	NO	%I0.3
Vrtalnik dvignjen	S3B1	NO	%I0.4
Vrtalnik spuščen	S3B2	NO	%I0.5
Tipka Start	START	NO	%I1.0
Tipka Reset	RESET	NO	%I1.1
Tipka Potrditev	ACKN., OK/NOT OK	NO	%I1.2
Tipka Avotmatsko/Ročno	AUTO/MAN	NO	%I1.3
Tipka Stop	STOP	NO	%I1.4
Tipka Izhod	QUIT	NO	%I1.5
Izhodi krmilnika:			
Pogon vrtalnika	A3M1	/	%Q0.1
Pogon mize	A3M2	/	%Q0.2
Spust vrtalnika	A3Y1	/	%Q0.3
Dvig vrtalnika	A3Y2	/	%Q0.4
Umik prijemala	A3Y4	/	%Q0.5
Signalna lučka Start	LUC_START	/	%Q0.6
Signalna lučka Reset	LUC_RESET	/	%Q0.7
Signalna lučka Acknowledge	LUC_ACKN	/	%Q1.0
Signalna lučka Empty	LUC_EMPTY	/	%Q1.1

Legenda: mirovno stanje kontakta NZ - zaprto
NO - odprto

2.4.4 Krmiljenje četrte postaje

Slika 2.14 prikazuje manipulator transportne postaje v laboratorijskem modelu modularnega proizvodnega sistema. Naloga te postaje je prenos obdelovancev iz obdelovalne mize v skladišče končnih izdelkov.



Slika 2.14: Manipulator na transportni postaji

Pomen oznak na sliki opisuje tabela 2.7.

Tabela 2.7: Pomen oznak na sliki manipulatorja

1	S4B2 pozicija prenašala (roke) pri delovni mizi
2	S4B3 pozicija prenašala (roke) pri skladišču
3	S4B5 linearni vodoravni cilinder (x-smer) povlečen
4	S4B4 linearni vodoravni cilinder (x-smer) zunaj
5	S4B6 prijemalo s sesalno šobo spuščeno
6	S4B7 prijemalo s sesalno šobo dvignjeno
7	A4Y2 potisk linearnega vodoravnega cilindra (x-smer)
8	A4Y1 povlek linearnega vodoravnega cilindra (x-smer)
9	A4Y7 spust linearnega dvižnega cilindra (z-smer)
10	A4Y5 prijemalo s sesalno šobo

Funkcionalni opis

Po pritisku tipke START mehanska roka prenese en obdelovanec z obdelovalne mize na postajo razvrščanja, kjer je skladišče končnih izdelkov. Če še ni pri obdelovalni mizi, se roka najprej zasuče v smer obdelovalne mize, nato se iztegne, vključi prijemalo s sesalno šobo in ga spusti k mizi. Prijemalo prisesa obdelovanec, roka ga dvigne, se pokrči ter zasuče v smer postaje razvrščanja. Nato se roka spet iztegne, spusti prijemalo navzdol ter izpusti obdelovanec. Zatem se prijemalo spet dvigne, roka pokrči ter vrne k obdelovalni mizi, kjer čaka na prenos novega obdelovanca. Od pritiska tipke START pa do vrnitve v začetno lego gori signalna lučka LUC_START. Nov prenos sprožimo z novim pritiskom tipke START.

Če pri poskusu prijema senzor v 2 s ne zazna, da je obdelovanec prijet, prijemalo popusti prijem, nato se izključijo vsi krmilni signali, vključi pa se alarm, kar označuje prižgana signalna lučka LUC_EMPTY. Isto se zgodi, če senzor prijema zazna predčasno izpustitev obdelovanca. Dokler je vključen alarm, s tipko START ne moremo sprožiti prenosa obdelovanca. Alarm lahko ugasnemo s tipko ACKNOWLEDGE.

Opozorilo: Pred premikanjem obdelovalne roke levo in desno, ko roka nekaj časa miruje, je potrebno oba cilindra najprej napolniti ročno!

Pomembno: Pri četrti postaji je potrebno zaradi števila izhodnih signalov v konfiguracijo krmilnika dodati še dodaten izhodni modul (**Signal board**). To storimo tako, da odpremo nastavitve krmilnika v **Device configuration**, nato pa iz v desnem stranskem meniju izberemo **Signal boards** $>$ **DQ** $>$ **DQ 4x24 VDC** $>$ **6ES7 222-1BD30-0XBO** ter s kurzorjem povlečemo na sliko krmilnika. S tem smo v konfiguracijo krmilnika dodali dodatne izhode. V primeru, da tega ne storimo, izhoda %Q4.0 ter %Q4.1 ne bosta delovala.

Uporabljeni vhodno/izhodni signali so zbrani v tabeli 2.8.

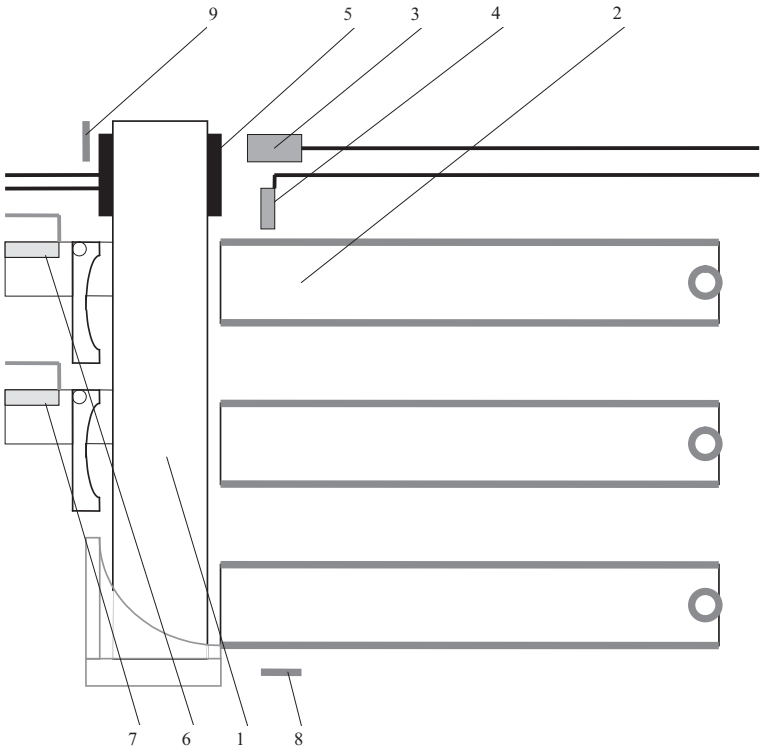
Tabela 2.8: Prireditvena tabela transportne postaje

	oznaka na napravi	mirovno stanje	naslov v PLK
Vhodi krmilnika:			
Obdelovanec prijet	S4S1	NO	%I0.0
Roka je pri postaji razvrščanja	S4B3	NO	%I0.1
Roka je pri obd. mizi	S4B2	NO	%I0.2
Roka iztegnjena	S4B4	NO	%I0.3
Roka pokrčena	S4B5	NO	%I0.4
Prijemalo spuščeno	S4B6	NO	%I0.5
Prijemalo dvignjeno	S4B7	NO	%I0.6
Tipka Start	START	NO	%I1.0
Tipka Reset	RESET	NO	%I1.1
Tipka Potrditev	ACKN., OK/NOT OK	NO	%I1.2
Tipka Avotmatsko/Ročno	AUTO/MAN	NO	%I1.3
Tipka Stop	STOP	NO	%I1.4
Tipka Izhod	QUIT	NO	%I1.5
Izhodi krmilnika:			
Popustitev prijema obdel.	A4Y6	/	%Q0.0
Prijem obdelovanca	A4Y5	/	%Q0.1
Spust prijemala navzdol	A4Y7	/	%Q0.2
Zasuk k skladišču	A4Y3	/	%Q4.2
Zasuk k mizi	A4Y4	/	%Q4.3
Pokrčenje roke	A4Y1	/	%Q4.0
Izteg roke	A4Y2	/	%Q4.1
Signalna lučka Start	LUC_START	/	%Q0.6
Signalna lučka Reset	LUC_RESET	/	%Q0.7
Signalna lučka Acknowledge	LUC_ACKN	/	%Q1.0
Signalna lučka Empty	LUC_EMPTY	/	%Q1.1

Legenda: mirovno stanje kontakta NZ - zaprto
NO - odprto

2.4.5 Krmiljenje pete postaje

Slika 2.15 prikazuje shemo postaje razvrščanja in skladiščenja v laboratorijskem modelu modularnega proizvodnega sistema. Naloga te postaje je razvrščanje obdelovancev v skladišče končnih izdelkov.



Slika 2.15: Postaja razvrščanja

Pomen oznak na sliki opisuje tabela 2.9.

Tabela 2.9: Pomen oznak na sliki postaje razvrščanja

1	sortirni trak
2	drča (skladišče)
3	obdelovanec prisoten na sortirnem traku
4	obdelovanec na poti v skladišče
5	A8M1 motor sortirnega traku
6	A8Y1 kretnica za drčo 1
7	A8Y2 kretnica za drčo 2
8	odsevnik senzorja 4
9	sprejemnik senzorja 3

Funkcionalni opis

S pritiskom tipke START se postaja postavi v obratovalno stanje, kar signalizira signalna lučka LUC_START.

Če senzor zazna obdelovanec na traku, se vključi pogon traku in glede na stanje stikala AUTO/MAN in tipke RESET se pomakne ustrezna kretnica.

- Če je stikalo v položaju AUTO (0), se kretnici vključujeta tako, da se enakomerno polnijo vse tri izhodne drče: pri prvem obdelovancu se vključi prva kretnica, pri drugem druga, pri tretjem nobena, nato se zaporedje ponavlja.
- Če je stikalo v položaju MAN (1), izhodno drčo izbiramo s pritiskanjem tipke RESET: preden prvič pritisnemo tipko, se polni prva drča, po enem pritisku druga, po drugem pritisku tretja, po naslednjem spet prva in tako naprej.

Ko senzor zazna, da je obdelovanec na poti v skladišče, se kretnici ter pogon traku izključijo. Če obdelovanec ne pride do poti v skladišče po 10 s od trenutka, ko ga je senzor zaznal na traku, se vključi alarm, kar označuje prižgana lučka LUC_EMPTY. Ob vključitvi alarma se trak ustavi ter kretnici izključita in postaja miruje, dokler je alarm vključen. Alarm lahko ugasnemo s tipko ACKNOWLEDGE.

S pritiskom tipke STOP ustavimo delovanje postaje, pri čemer se tekoči delovni cikel izvede do konca, nato pa se postaja ustavi. Od pritiska tipke STOP do popolne zaustavitve lučka LUC_START utripa s periodo 1 s. Po zaustavitvi lahko delovno postajo ponovno poženemo s tipko START, pri čemer se prične izbiranje kretnic od začetka, najprej se torej polni prva drča.

Uporabljeni vhodno/izhodni signali so zbrani v tabeli 2.10.

Tabela 2.10: Prireditvena tabela postaje razvrščanja

	oznaka na napravi	mirovno stanje	naslov v PLK
Vhodi krmilnika:			
Kretnica 1 ni aktivna	S8B1	NO	%I0.0
Kretnica 1 je aktivna	S8B2	NO	%I0.1
Kretnica 2 ni aktivna	S8B3	NO	%I0.2
Kretnica 2 je aktivna	S8B4	NO	%I0.3
Obdelovanec na traku	S8S1	NZ	%I0.4
Obdelov. na poti v skladišče	S8S2	NO	%I0.5
Tipka Start	START	NO	%I1.0
Tipka Reset	RESET	NO	%I1.1
Tipka Potrditev	ACKN., OK/NOT OK	NO	%I1.2
Tipka Avotmatsko/Ročno	AUTO/MAN	NO	%I1.3
Tipka Stop	STOP	NO	%I1.4
Tipka Izhod	QUIT	NO	%I1.5
Izhodi krmilnika:			
Kretnica 1 aktivna	A8Y1	/	%Q0.0
Kretnica 2 aktivna	A8Y2	/	%Q0.1
Motor traku	A8M1	/	%Q0.2
Signalna lučka Start	LUC_START	/	%Q0.6
Signalna lučka Reset	LUC_RESET	/	%Q0.7
Signalna lučka Acknowledge	LUC_ACKN	/	%Q1.0
Signalna lučka Empty	LUC_EMPTY	/	%Q1.1

Legenda: mirovno stanje kontakta NZ - zaprto
NO - odprto

Laboratorijska vaja št. 3

Programiranje krmilij s strukturiranim tekstom

Pri vaji bomo spoznali načrtovanje logičnega in sekvenčnega vodenja s strukturiranim tekstom. Načrtovali bomo vodenje laboratorijskega modularnega proizvodnega sistema.

3.1 Strukturiran tekst

Stukturiran tekst (angl. **Structured Control Language - SCL** ali **Structured Text - ST**) predstavlja enega od standardnih programskih jezikov za programiranje krmilnikov po standardu IEC 61131-3. Gre za visokonivojski programski jezik, ki po svoji sintaksi spominja na Pascal, na kateremu je tudi osnovan. Uporaba SCL-ja je še posebej primerna pri programiranju kompleksnih algoritmov, aritmetičnih funkcij ter obdelavi podatkov. Prednosti SCL-ja pred ostalimi programskimi jeziki, kot npr. lestvičnimi diagrami, funkcijskimi bločnimi diagrami ter seznami ukazov, so:

- enostavnejši, hitrejši in učinkovitejši razvoj programa z uporabo programskih zank in pogojev: IF...THEN...ELSE, WHILE...DO, FOR, ...,
- lažja implementacija aritmetičnih operacij in kompleksnejših algoritmov,
- večja preglednost programa, lažje strukturiranje,
- manjša dovzetnost za napake, enostavnejše testiranje ter lažje iskanje napak.

Strukturiran tekst je sestavljen iz nabora ukazov, zank in pogojev, podobno kot pri ostalih programskih jezikih, npr.:

```
IF "vrednost" < 7 THEN
    WHILE "vrednost" < 8 DO
        "vrednost" := "vrednost" + 1;
    END_WHILE;
END_IF;
```

3.1.1 Osnovna navodila za delo s strukturiranim tekstom

SCL je visokonivojski programski jezik za Siemens-ove CPE-je S7. SCL podpira bločno strukturo STEP 7 in organizacijo programov, pri kateri je možno znotraj istega projeka združiti programske bloke, ki so napisani v vseh podprtih programskih jezikih, npr. strukturiranem tekstu, lestvičnih diagramih ali funkcijskih bločnih diagramih. Ukazi SCL temeljijo na standardnih programskih operatorjih, kot so prireditev (:=) ter matematične funkcije (+ za seštevanje, - za odštevanje, * za množenje ter / za deljenje). SCL uporablja standardne PASCAL-ove operacije, kot so `IF-THEN-ELSE`, `CASE-OF`, `FOR-TO-DO`, `WHILE-DO`, `REPEAT-UNTIL`, `CONTINUE`, `GOTO` ter `RETURN`.

Operatorji

SCL podpira vse standardne programske operatorje. Nekateri od pomebnejših so prikazani v tabeli 3.1.

Tabela 3.1: Operatorji v SCL

Tip	Operacija	Operator	Prioriteta
Matematične	Potenca	**	1
	Množenje	*	2
	Deljenje	/	2
	Modulus	MOD	2
	Seštevanje	+	3
	Odštevanje	-	3
Primerjalne	Manj kot	<	4
	Manj ali enako	<=	4
	Več kot	>	4
	Več ali enako	>=	4
	Enako	=	5
	Ni enako	<>	5
Bitne	Negacija	NOT	2
	Logični IN	AND	6
	Ekskluzivni ALI	XOR	7
	Logični ALI	OR	8
Prireditev	Prireditev	:=	9

Kontrolni stavki

Kontrolni stavki so poseben tip izrazov v SCL-ju in se uporabljajo za:

- programske razvejitve,
- ponavljanje določenih odsekov kode,
- skoke na druge odseke programske kode,
- pogojno izvajanje kode.

Kontrolni stavki v SCL-ju zajemajo: `IF-THEN-ELSE`, `CASE-OF`, `FOR-TO-DO`, `WHILE-DO`, `REPEAT-UNTIL`, `CONTINUE`, `GOTO` ter `RETURN`.

Nekaj primerov kontrolnih stavkov:

- Primer zanke FOR-TO-DO:
```
FOR "x" := 0 TO "max" DO
   "sum" := "sum" + "x";
END_FOR;
```
- Primer zanke WHILE-DO:
```
WHILE "x" <= "max" DO
   "sum" := "sum" + "x";
   "x"   := "x" + 1;
END_WHILE;
```
- Primer pogoja REPEAT-UNTIL:
```
REPEAT
   "x" := "x" + 1;
UNTIL "x" = 10
END_REPEAT;
```
- Primer pogoja IF-THEN-ELSE:
```
IF "x" <= 10 THEN
   "n" := 0;
ELSIF "x" > 10
   "n" := 1;
END_IF;
```
- Primer pogoja CASE-OF:
```
CASE "x" OF
   1: "n" := 5;
   2: "n" := 8;
```

```
    3,4: "n":= 10;
    5..10: "n" := 15
ELSE:
    "n" := 0;
END_CASE;
```

Naslavljanje spremenljivk

SCL omogoča dve vrsti naslavljanja spremenljivk:

- simbolično,
- absolutno.

Simbolično naslavljanje spremenljivke pomeni, da spremenljivko na mestu uporabe kličemo po njenem imenu oz. tag-u, ki smo ga določili v tabeli spremenljivk **PLC tags**.

Primer:

- `"PLC_Tag_1"` - ime spremenljivke v tabeli spremenljivk,
- `"Data_block_1".Tag_1` - ime spremenljivke (`Tag_1`) v podatkovnem bloku (`"Data_block_1"`),
- `"Data_block_1".MyArray[#i]` - i-ti element v polju (`MyArray`) spremenljivke v podatkovnem bloku (`"Data_block_1"`).

Absolutno naslavljanje spremenljivke pomeni, da spremenljivko na mestu uporabe kličemo po njenem imenu, kot le-ta nastopa na krmilniku.

Primer:

- `%I0.0` - vhod 0.0 na krmilniku,
- `%Q0.5` - izhod 0.5 na krmilniu,
- `%M0.1` - notranja spremenljivka (spomin) 0.1 na krmilniku.

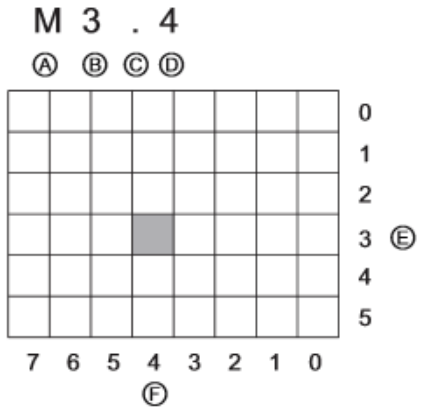
Pomembno: Zaradi večje preglednosti programske kode, v smislu pomena posameznih spremenljivk, v programu uporabljajte **simbolično naslavljanje** spremenljivk s pomensko ustreznim poimenovanjem. To storite z vnosom spremenljivk v tabelo spremenljivk **PLC tags** na enak način kot pri vaji 1 in 2.

		Name	Data type	Address	Retain	Visibl...	Acces...
1		Tipka_1	Bool	%I1.0	☐	☑	☑
2		Tipka_2	Bool	%I1.1	☐	☑	☑
3		Luc_1	Bool	%Q0.6	☐	☑	☑
4		Luc_2	Bool	%Q0.7	☐	☑	☑
5		Luc_3	Bool	%Q1.0	☐	☑	☑
6		Koncaj	Bool	%M0.5	☐	☑	☑
7		Blokiraj_start	Bool	%M0.3	☐	☑	☑
8		Postavi_stevec	Bool	%M0.4	☐	☑	☑
9		Stanje	Int	%MW1	☐	☑	☑
10		N1	Bool	%M0.0	☐	☑	☑
11		N2	Bool	%M0.1	☐	☑	☑
12		N3	Bool	%M0.2	☐	☑	☑
13		<Add new>			☐	☑	☑

Slika 3.1: Tabela spremenljivk

Dodeljevanje pomnilniških mest

Pri vnosu spremenljivk v tabelo spremenljivk **PLC tags** je potrebno vsaki spremenljivki dodeliti pomnilniško mesto. V primeru spremenljivke, ki je tipa **bool**, le-tej dodelimo eno pomnilniško mesto, npr. `%M0.0`, `%M0.4`, `%M1.4` itd. V primeru spremenljivk, ki v spominu zavzamejo več kot 1 bit informacije, npr. **int**, **real** itd., se jim v skladu z njihovo dolžino dodeli več pomnilniških mest, npr. `%MW1`, `%MW2` itd. Razdelitev pomnilniških mest prikazuje slika 3.2.



Slika 3.2: Razdelitev pomnilniških mest: A - identifikator področja pomnilnika (M), B - naslov bajta (bajt 3), C - ločilo (.), D - lokacija bita v bajtu (bit 4 od 8), E - bajti v spominu (3), F - biti v bajtu (4)

V skladu z zgornjo razdelitvijo pomnilniških mest bi 8-bitna spremenljivka tipa **int** na naslovu `%MW0` zavzela celotno prvo vrstico pomnilnika.

Pomembno: V primeru uporabe spremenljivk, ki v spominu zavzamejo več kot 1 bit informacije, npr. **int**, **real**, **char** itd., moramo pri dodeljevanju naslovov tem spremenljivkam paziti, da jim dodelimo mesto v pomnilniku, ki se ne prekriva s spominskimi mesti ostalih spremenljivk. Npr. spremenljivka tipa **bool** z naslovom `%M0.0` ter spremenljivka tipa **integer** z naslovom `%MW0` si delita prvi bit, s čimer lahko pride do neželenega delovanja. **Sistem nas na prekrivanje spominskih bitov ne opozori**!

Klicanje blokov

V programski kodi SCL-ja je možno klicanje drugih programskih blokov, npr. **funkcijskih blokov (FB)** ali **funkcij (FC)**. Klic drugega bloka enostavno izvedemo tako, da v kodo vnesemo njegovo ime (ali absolutni naslov), skupaj s parametri bloka.

Primer klicanja drugih blokov:

- `result := Function_Block(Input_A := 1, Input_B := 2);` - klic bloka, ki vrne vrednost v spremenljivko `result`, vhodna parametra pa sta `Input_A` in `Input_B`,
- `Function_Block();` - klic bloka, ki ne vrača vrednosti, prav tako pa nima vhodnih parametrov.

3.2 Programiranje s strukturiranim tekstom v orodju TIA Portal

Programsko orodje TIA Portal poleg ostalih programskih jezikov omogoča tudi programiranje logičnih krmilnikov s strukturiranim tekstom (Structured Control Language - SCL) po standardu IEC 61131-3.

Postopek ustvarjanja novega projekta ali odpiranja že obstoječega projekta je podoben kot pri programiranju z lestvičnim diagramom. Ravno tako je enak postopek izbire krmilnika ter njegovih nastavitev (glejte 1. in 2. vajo), razlika pa je pri ustvarjanju in urejanju programskih blokov.

3.2.1 Ustvarjanje novega programskega bloka

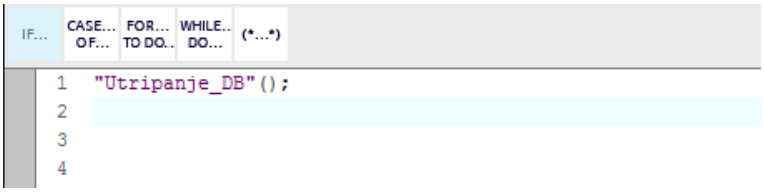
Ko ustvarimo nov projekt, nam program že sam zgenerira glavni organizacijski blok OB1, ki pa je tipa **ladder digram**. Ker bomo vajo izvedli v strukturiranem tekstu, moramo spremeniti tip organizacijskega bloka. To storimo tako, da obstoječi OB1

pobrišemo in s klikom na izbran krmilnik ter **Program blocks** > **Add new block** odpremo okno, v katerem izberemo **Organization block**, za programski jezik izberemo **SCL** ter ga poimenujemo.

Podobno kot pri predhodnih vajah, bo OB1 vseboval zgolj strukturo celotnega programa, ki jo tvorijo funkcijski bloki, ne pa dejanske kode.

Ko imamo ustvarjen glavni organizacijski blok OB1, ustvarimo še funkcijske bloke - FB, ki bodo vsebovali kodo dejanskega programa, ki se bo izvajal na krmilniku. To storimo s klikom na izbran krmilnik ter **Program blocks** > **Add new block**. V oknu, ki se odpre, izberemo **Function block**, za programski jezik izberemo **SCL** ter ga poimenujemo.

Če želimo, da se ustvarjeni funkcijski blok izvaja na krmilniku, ga moramo vključiti v organizacijski blok OB1. To storimo tako, da odpremo organizacijski blok OB1 ter vanj vstavimo klic funkcijskega bloka, kot prikazuje slika 3.3.



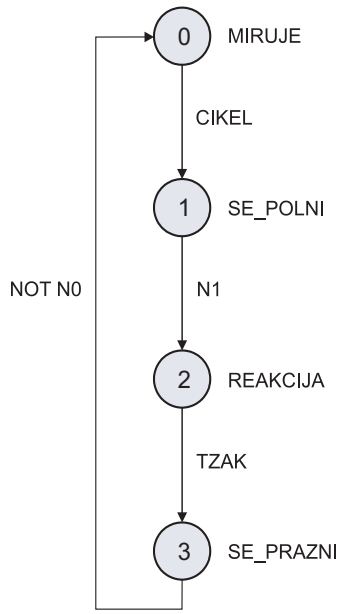
Slika 3.3: Vključitev funkcijskega bloka *Utripanje* v organizacijski blok OB1.

Ker blok ne vrača nobenih vrednosti, prav tako pa tudi nima vhodnih argumentov, ni potrebno nobeni spremenljivki prirediti izhodne vrednosti, oklepaj v klicu funkcije pa je prazen. Takšen način klicanja blokov je primeren za izvedbo večine blokov, ki jih bomo potrebovali za izvedbo vaje.

3.2.2 Prehajanje stanj v strukturiranem tekstu

Podobno kot pri realizaciji prehajanja stanj v lestvičnem diagramu pri vaji 2, moramo prehajanje stanj realizirati tudi pri strukturiranem tekstu. Možnih realizacij prehajanja stanj je več, v nadaljevanju pa si bomo podrobneje pogledali realizacijo z uporabo stavka `CASE-OF`.

Pogled na diagram prehajanja stanj ter pripadajočo kodo v programskem jeziku SCL razkriva, da za realizacijo diagrama prehajanja stanj potrebujemo zgolj eno spremenljivko - *STANJE*, ki definira stanje, v katerem se sistem nahaja. Spremenljivka *STANJE* lahko zavzame vrednosti od 0 do 3, s čimer so definirana stanja *MIRUJE*, *SE_POLNI*, *REAKCIJA* in *SE_PRAZNI*. Prehodi med posameznimi stanji so realizirani znotraj stavka `CASE-OF` z uporabo stavkov `IF-THEN`, s katerimi program preverja, ali je v danem stanju izpolnjen pogoj za prehod v naslednje stanje. Za stavkom `CASE-OF` sledi še realizacija akcij, ki se izvajajo v posameznem stanju.




```
CIKEL := (START or CIKEL) AND NOT STOP;
IF START_PLK THEN
   STANJE := 0;
END_IF;

//realizacija prehajanja stanj
CASE STANJE OF
   0: IF CIKEL THEN STANJE := 1; END_IF;
   1: IF N1 THEN STANJE := 2; END_IF;
   2: IF TZAK THEN STANJE := 3; END_IF;
   3: IF N0 THEN STANJE := 0; END_IF;
END_CASE;

//realizacija pripadajočih akcij
VIN := (STANJE = 1) AND NOT N1;
VOUT := (STANJE = 3) AND N0;
GR := (STANJE = 2) AND NOT TEMP;
ME := (STANJE = 2);
Casovnik(IN := (STANJE = 2),PT := T#30m);
TZAK := Casovnik.Q;
```

3.2.3 Izdelava programa

Za lažje razumevanje ter lažji pričetek programiranja v SCL-ju si na primeru poglejmo izdelavo programa s spodnjim funkcionalnim opisom. Tudi v tem primeru bomo uporabili realizacijo prehajanja stanj z uporabo stavka `CASE-OF`.

S pritiskom tipke START, kateri pripada spremenljivka *Tipka_1*, se pričnejo zaporedno prižigati lučke START (*Luc_1*), STOP (*Luc_2*) in ACKNOWLEDGE (*Luc_3*). Vsaka lučka se prižge za 1 s in nato ugasne. Ko lučka ugasne, se za 1 s prižge naslednja itd. S tipko RESET, kateri pripada spremenljivka *Tipka_2*, lahko utripanje predčasno izključimo. Po dveh ponovitvah sekvence se utripanje ustavi, s ponovnim pritiskom na tipko START pa lahko sekvenco znova poženemo. Med samo sekvenco pritisk na tipko START ne vpliva na delovanje.

Izdelava programa:

- Odpremo nov projekt in ga poimenujemo.
- V levem stranskem meniju dodamo nov krmilnik (**Add new device**) in potrdimo z **OK**.
- Z dvoklikom na ikono  na sliki PLK-ja prikličemo okno z omrežnimi nastavitvami krmilnika, v katerega vpišemo IP naslov krmilnika.
- S klikom na izbran krmilnik ter izbiro **PLC tags** > **Add new tag table** ustvarimo novo tabelo spremenljivk in jo poimenujemo.
- Odpremo ustvarjeno tabelo spremenljivk in začnemo vnašati potrebne spremenljivke. Tipki sta vhodni spremenljivki (*Tipka_1* in *Tipka_2*), spremenljivke, povezane z lučkami (*Luc_1*, *Luc_2* in *Luc_3*), pa izhodne. Potrebujemo še interno spremenljivko (*Stanje*) tipa **integer**, ki je potrebna za izvedbo prehajanja stanj ter tri dodatne spremenljivke za postavitev števca (*Postavi_stevec*), blokiranje tipke START (*Blokiraj_start*) ter končanje sekvence (*Koncaj*). Spremenljivke prikazuje slika 3.4.

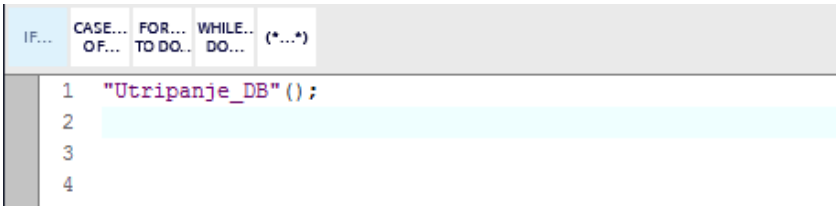
	Name	Data type	Address	Retain	Visibl...	Acces...
1	Tipka_1	Bool	%I1.0	☐	☑	☑
2	Tipka_2	Bool	%I1.1	☐	☑	☑
3	Luc_1	Bool	%Q0.6	☐	☑	☑
4	Luc_2	Bool	%Q0.7	☐	☑	☑
5	Luc_3	Bool	%Q1.0	☐	☑	☑
6	Koncaj	Bool	%M0.5	☐	☑	☑
7	Blokiraj_start	Bool	%M0.3	☐	☑	☑
8	Postavi_stevec	Bool	%M0.4	☐	☑	☑
9	Stanje	Int	%MW1	☐	☑	☑
10	N1	Bool	%M0.0	☐	☑	☑
11	N2	Bool	%M0.1	☐	☑	☑
12	N3	Bool	%M0.2	☐	☑	☑
13	<Add new>			☐	☑	☑

Slika 3.4: Seznam spremenljivk

Pomembno: Pazimo na pravilno naslavljanje spremenljivke **int**. Glej stran 72!

- V levem stranskem meniju najprej pobrišemo glavni organizacijski blok OB1, nato pa dodamo novega s klikom na izbran krmilnik ter (**Program blocks** > **Add new block**). V oknu, ki se odpre, izberemo tip bloka **Organization block**, ga poimenujemo *Main* ter izberemo tip **Program cycle** in programski jezik **SCL**.

- V levem stranskem meniju dodamo nov funkcijski blok s klikom na izbran krmilnik ter (**Program blocks** > **Add new block**). V oknu, ki se odpre, izberemo tip bloka **Function block**, ga poimenujemo *Utripanje* ter izberemo programski jezik **SCL**.
- Pričnemo s pisanem strukturiranega teksta v organizacijskem bloku *Main*, kjer moramo zgolj dodati funkcijski blok *Utripanje*, kot prikazuje slika 3.5



Slika 3.5: Vključitev funkcijskega bloka *Utripanje* v glavni organizacijski blok *Main*

- Pričnemo s pisanjem stukturiranega teksta v funkcijskem bloku *Utripanje*, kot prikazuje slika 3.6.

```
1  //start
2  IF "Tipka_1" = TRUE AND "Blokiraj_start" = FALSE THEN
3      "Stanje" := 1; "Blokiraj_start" := 1; "Postavi_stevec" := 1;
4  END_IF;
5
6  //reset
7  IF "Tipka_2" = TRUE OR "Koncaj" = TRUE THEN
8      "Stanje" := 0; "Blokiraj_start" := 0;
9  END_IF;
10
11 //prehajanje stanj
12 CASE "Stanje" OF
13     1: IF "N1" = TRUE THEN "Stanje" := 2; END_IF;
14     2: IF "N2" = TRUE THEN "Stanje" := 3; END_IF;
15     3: IF "N3" = TRUE THEN "Stanje" := 1; END_IF;
16 END_CASE;
17
18 //akcije
19 //stanje = 1; casovnik 1
20 "Luc_1" := ("Stanje" = 1);
21 "Casovnik_1".TON(IN := ("Stanje" = 1), PT := T#1s);
22 "N1" := "Casovnik_1".Q;
23
24 //stanje = 2; casovnik 2
25 "Luc_2" := ("Stanje" = 2);
26 "Casovnik_2".TON(IN := ("Stanje" = 2), PT := T#1s);
27 "N2" := "Casovnik_2".Q;
28
29 //stanje = 3; casovnik 3
30 "Luc_3" := ("Stanje" = 3);
31 "Casovnik_3".TON(IN := ("Stanje" = 3),
32                  PT := T#1s);
33 "N3" := "Casovnik_3".Q;
34
35 //stevec
36 "Stevec_1".CTD(CD := "N3", LD := "Postavi_stevec", PV := 2, Q => "Koncaj");
37 "Postavi_stevec" := 0;
```

Slika 3.6: Strukturiran tekst v funkcijskem bloku *Utripanje*

- Projekt shranimo, prevedemo in naložimo na krmilnik.

3.3 Izvedba vodenja modularnega proizvodnega sistema s strukturiranim tekstom

Napišite in preizkusite program za programirljivi logični krmilnik, ki bo krmilil eno od postaj modularnega sistema tako, kot določa funkcionalni opis za to postajo (podan

pri 2. vaji).

Za programiranje uporabite **SCL**. Sestavite osnovno zaporedje prehajanja stanj, ga dopolnite z ustreznimi akcijami in preizkusite na napravi. Izpišite program na tiskalnik in dodajte izpis kot prilogo k rezultatom laboratorijskih vaj.

Laboratorijska vaja št. 4

Izvedba zveznega vodenja procesov

Pri vaji bomo spoznali načrtovanje zveznega vodenja laboratorijskih naprav z uporabo programirljivih logičnih krmilnikov. Pri tem bomo uporabljali programsko orodje TIA Portal ter vgrajena orodja za konfiguriranje in parametriranje regulatorjev.

4.1 PLK Siemens S7-1200 in razširitveni modul SM 1232

Za izvedbo zveznega vodenja laboratorijske naprave bomo uporabili programirljivi logični krmilnik Siemens S7-1200. Potrebne specifikacije krmilnika so bile opisane v besedilu 1. vaje, zaradi česar na tem mestu podajamo samo opis razširitvenega modula **SM 1232 AQ 2x14 bit** ter lastnosti analognih vhodov na krmilniku. Razširitveni modul SM 1232 predstavlja modul z dvema analognima izhodoma, ki ju potrebujemo za izvedbo zveznega vodenja. Na krmilniku S7-1200 lahko zasledimo dva analogna vhoda, analognih izhodov pa ne, zaradi česar ta problem rešujemo z razširitveno kartico ali razširivenim modulom.

4.1.1 Analogni vhodi AI

Analogna vhoda, ki se nahajata na krmilniku S7-1200, se uporabljata z zajem tokovnih ali napetostnih signalov v območjih 0 - 20 mA ali 4 - 20 mA (tokovni) ter $\pm$10 V, $\pm$5 V ali $\pm$2,5 V (napetostni). Vgrajeni analogno digitalni pretvornik pretvori vhodni signal v desetiško število med -27648 in 27648.

4.1.2 Analogni izhodi AQ

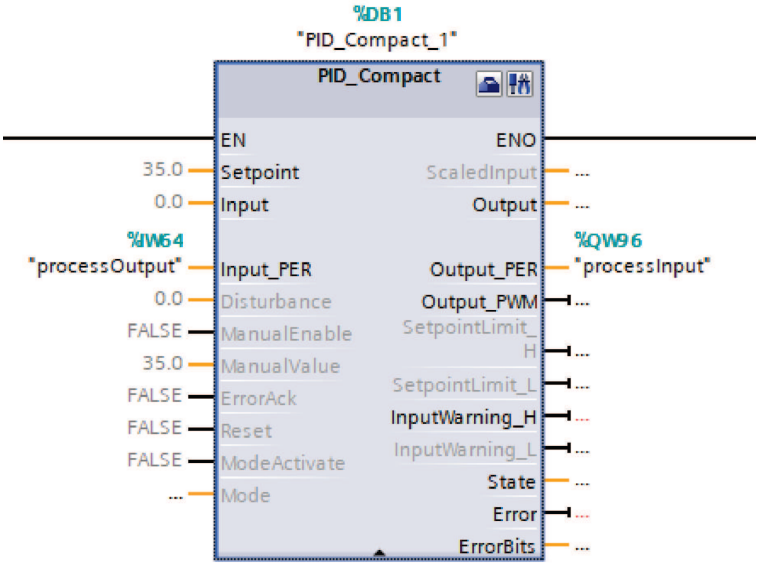
Analogna izhoda, ki se nahajata na razširitvenem modulu SM 1232, se uporabljata za pošiljanje tokovnih ali napetostnih signalov v območjih 0 - 20 mA (tokovni) ter $\pm$10 V (napetostni) na priklopljene naprave.

4.2 Programsko orodje TIA Portal

Za izvedbo zveznega vodenja laboratorijske naprave ter izgradnjo uporabniškega vmesnika (HMI) bomo uporabili že poznano okolje TIA Portal. Izdelava programske kode poteka na podoben način kot smo bili vajeni pri prvih treh vajah, obstajajo pa določene razlike, ki jih podajamo v nadaljevanju.

4.2.1 Blok PID_compact

Programirljivi logični krmilniki družine S7 poleg logičnega vodenja podpirajo tudi zvezno vodenje procesov. Na tem mestu se bomo osredotočili na proizvajalčevo najnovejšo rešitev s tega področja, t.j. Compact PID, ki ga podpirajo vsi krmilniki tipa S7-1200 ter S7-1500. Glavni blok oz. objekt iz omenjene knjižnice, ki ga bomo uporabljali za izvedbo vodenja je **PID_compact**, ki je prikazan na sliki 4.1.



Slika 4.1: Blok PID_compact

Opazimo lahko, da ima blok PID_compact veliko število vhodov in izhodov, kar omogoča veliko prilagodljivost ter enostavno vgradnjo bloka v sisteme vodenja. V tabeli 4.1 so opisani vhodi in izhodi bloka PID_compact.

	Ime	Tip	Opis
Vhodi	Setpoint	Real	Referenca
	Input	Real	Trenutna vrednost regulirane veličine
	Input_PER	Int	Trenutna vrednost regulirane veličine iz I/O (AI vhoda)
	Disturbance	Real	Motnja
	ManualEnable	Bool	Aktiviranje ročnega načina delovanja
	ManaulValue	Real	Vrednosti izhoda regulatorja v ročnem načinu
	ErrorAck	Bool	Potrditev napake
	Reset	Bool	Reset, ponovni zagon regulatorja
	ModeActivate	Bool	Potrditev načina delovanja
Izhodi	ScaledInput	Real	Skalirana vrednost trenutne regulirane veličine
	Output	Real	Izhod regulatorja
	Output_PER	Int	Izhod regulatorja na I/O (AQ izhod)
	Output_PWM	Bool	Izhod regulatorja v pulzno-širinski modulaciji
	SetpointLimit_H	Bool	Referenca je omejena na zgornjo mejo
	SetpointLimit_L	Bool	Referenca je omejena na spodnjo mejo
	InputWarning_H	Bool	Trenutna vrednost regulirane veličine je previsoka
	InputWarning_L	Bool	Trenutna vrednost regulirane veličine je prenizka
	State	Int	Stanje regulatorja (0=neaktiven, 1=SUT, 2=TIR, 3=auto, 4=manual)
	Error	Bool	Napaka v delovanju
	ErrorBits	DWord	Opis napake
Vh/Izh	Mode	Int	Izbira načina delovanja

Tabela 4.1: Vhodno/izhodni signali bloka PID_compact

Za namen vodenja laboratorijske naprave bomo uporabili le del omenjenih vhodov in izhodov, ki jih prikazuje tabela 4.2.

	Ime	Tip	Opis
Vhodi	Setpoint	Real	Referenca
	Input_PER	Int	Trenutna vrednost regulirane veličine iz I/O (AI vhoda)
	ManualEnable	Bool	Aktiviranje ročnega načina delovanja
	ManaulValue	Real	Vrednosti izhoda regulatorja v ročnem načinu
	ErrorAck	Bool	Potrditev napake
	Reset	Bool	Reset, ponovni zagon regulatorja
	ModeActivate	Bool	Izbira načina delovanja
Izhodi	Output_PER	Int	Izhod regulatorja na I/O (AQ izhod)
	InputWarning_H	Bool	Trenutna vrednost regulirane veličine je previsoka
	InputWarning_L	Bool	Trenutna vrednost regulirane veličine je prenizka
	State	Int	Stanje regulatorja (0=neaktiven, 1=SUT, 2=TIR, 3=auto, 4=manual)
	Error	Bool	Napaka v delovanju
	ErrorBits	DWord	Opis napake

Tabela 4.2: Izbrani vhodno/izhodni signali bloka PID_compact

4.2.2 Izdelava novega projekta za realizacijo zveznega vodenja

Za namen izdelave sistema zveznega vodenja bomo ustvarili nov projekt in ga ustrezno poimenovali. Začetni koraki, vključno z dodajanjem krmilnika in nastavljanjem naslova IP, so enaki kot pri vaji 1 (poglavje 1.2.2).

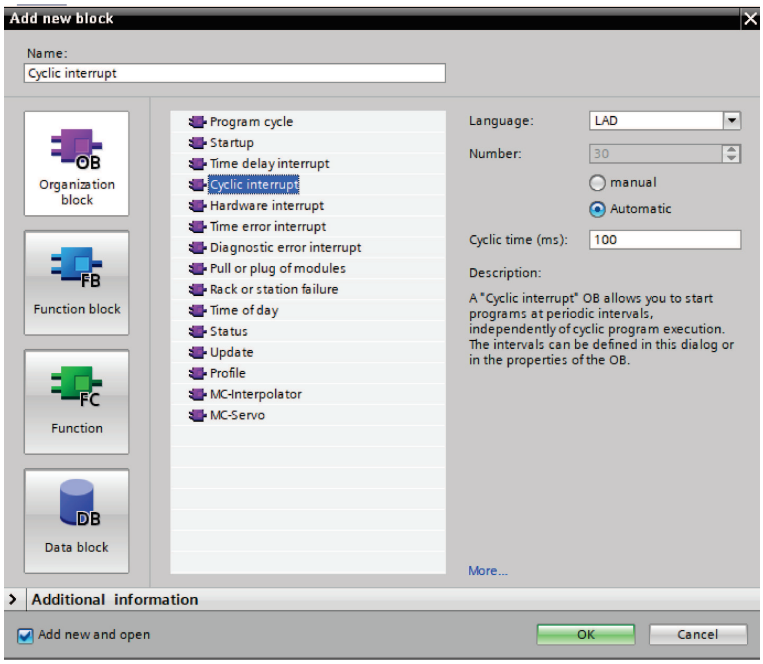
Ker bomo v konkretnem primeru poleg analognih vhodov potrebovali tudi analogne izhode, ki jih na samem krmilniku ni, je potrebno v konfiguracijo strojne opreme dodati tudi razširitveni modul SM 1232. To storimo tako, da v projekt najprej dodamo krmilnik (po znanem postopku), nato pa v desnem stranskem meniju odpremo zavihek **Hardware catalog**, v njem poiščemo razširitveni modul **AQ** $>$ **AQ 2x14BIT** $>$ **6ES7 232-4HB32-0XB0** in ga povlečemo na prosto mesto na desni strani krmilnika.

Preverimo še kakšna sta naslova analognega vhoda in izhoda, ki ju bomo uporabili za vodenje naprave. Za analogni vhod to storimo tako, da kliknemo na sliko krmilnika ter v spodnjem meniju izberemo zavihek **General** ter nadalje **AI 2** $>$ **analog inputs** $>$ **Channel 0**. Običajno je ta naslov **IW64**. Za analogni izhod pa naslov preverimo tako, da kliknemo na sliko razširitvenega modula SM 1232 ter v spodnjem meniju izberemo zavihek **General** ter nadalje **AQ 2** $>$ **analog outputs** $>$ **Channel 0**. Običajno je ta naslov **QW96**.

4.2.3 Vključitev bloka PID_compact v program

Pri prvih treh vajah smo za izdelavo programske kode primarno uporabljali en organizacijski (OB1) ter več funkcijskih blokov (FB). Za izvedbo zveznega vodenja z uporabo bloka PID_compact, pa je postopek nekoliko drugačen. Program, ki je nameščen na krmilniku se izvaja ciklično. To pomeni, da PLK prebere stanje vhodov, izvrši programsko kodo ter zapiše vrednosti na izhode, postopek pa se nato ponavlja. Čas, ki ga PLK potrebuje za izvedbo enega cikla se imenuje čas cikla in je v veliki meri odvisen od zahtevnosti programske kode, ki jo izvaja PLK ter se lahko spreminja. Pri izvajanju algoritmov zveznega vodenja je nujno, da je čas cikla vedno enak, saj so od njega odvisne nastavitve regulatorja. Zaradi tega se za izvedbo zveznega vodenja uporabljajo organizacijski bloki, ki imajo konstanten čas cikla. Te bloke imenujemo **ciklični prekinitveni blok OB30** (*ang. Cyclic interrupt*).

V obstoječi projekt moramo torej dodati ciklični prekinitveni blok. To storimo s klikom na izbran krmilnik **PLC1[CPU 1214C AC/DC/Rly]** $>$ **Program blocks** $>$ **Add new block**. V prikazanem oknu najprej izberemo **Organization block** ter iz seznama blokov **Cyclic interrupt**, kot prikazuje slika 4.2. Programski jezik naj bo LAD, čas cikla (vzorčenja) pa 100 ms.



Slika 4.2: Dodajanje bloka cyclic interrupt

V ustvarjeni prekinitveni blok nadalje dodamo block PID_compact. Le-tega lahko najdemo v desnem stranskem meniju, ki se nahaja v rubriki **Technology** > **PID control** > **Compact PID**.

4.2.4 Definicija spremenljivk

Vhodno-izhodne spremenljivke

Podobno kot pri prvih treh vajah, je potrebno tudi v tem primeru definirati spremenljivke, ki so vezane na krmilniške vhode in izhode. V glavni tabeli spremenljivk (**PLC tags**) definiramo dve spremenljivki, in sicer:

- processInput (tip: **int**, naslov: **%QW96**) - izhod krmilnika, vhod v proces,
- processOutput (tip: **int**, naslov: **%IW64**) - vhod krmilnika, izhod procesa.

Tipk in lučk na uporabniškem pultu v tem primeru ne bomo potrebovali, zato definicija spremenljivk teh vhodov in izhodov ni potrebna.

Pomožne spremenljivke

Za izgradnjo uporabniškega vmesnika HMI bomo potrebovali tudi ostale vhodno/izhodne spremenljivke bloka PID_compact. Ker so le-te med posameznimi objekti

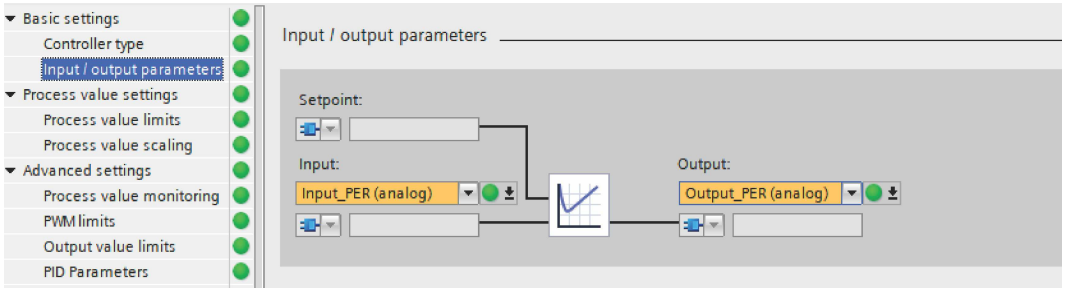
znotraj projekta dostopne, jih ni potrebno dodatno definirati v tabelah spremenljivk, saj so že definirane v bloku PID_compact.

4.2.5 Nastavitve bloka PID_compact

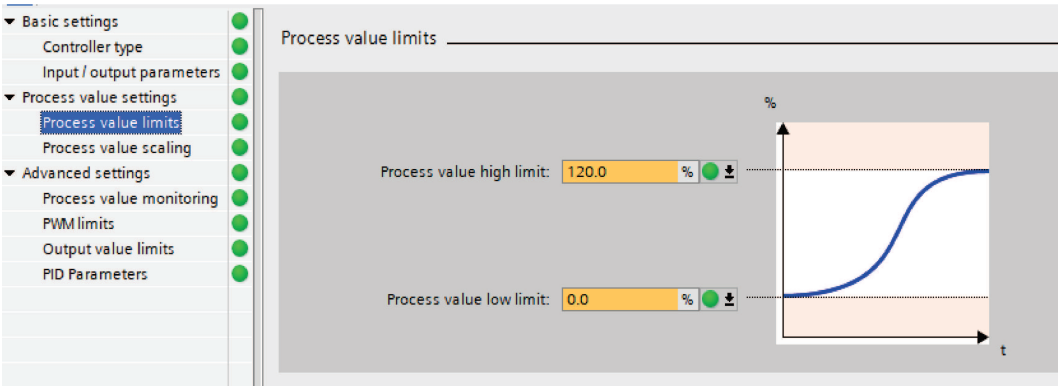
Za pravilno delovanje bloka PID_compact je potrebno le-tega konfigurirati. To storimo v levem meniju s klikom na **PLC1[CPU 1214C AC/DC/Rly]** > **Technology objects** > **PID_compact** > **Configuration**. Odpre se okno, v katerem nastavimo vhodne in izhodne spremenljivke, njihove omejitve, skaliranje, regulacijsko strukturo ipd. Za potrebe vodenja laboratorijske naprave je potrebno nastaviti naslednje:

- vhode ter izhode - **Input/output parameters** (Input: Input_PER, Output: Output_PER),
- omejitve - **Process value limits** (Process value high limit: 120 %, Process value low limit: 0 %),
- skaliranje vhodnega signala - **Process value scaling** (Input_Per: Enabled, Low: 0, High: 27648, Scaled low process value: 0.0 %, Scaled high process value: 100 %),
- strukturo regulatorja - **PID Parameters** (Control Structure).

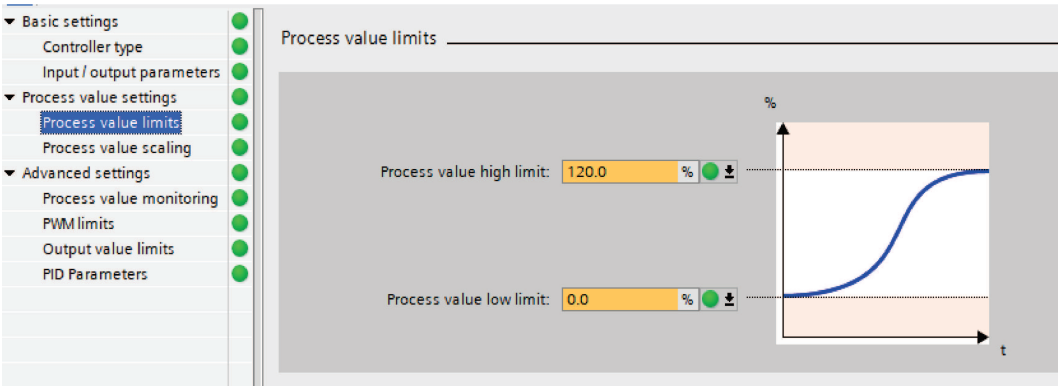
Nastavitve prikazujejo slike 4.3, 4.4 in 4.5.



Slika 4.3: Nastavitve bloka PID_compact (vhodi in izhodi)



Slika 4.4: Nastavitve bloka PID_compact (omejitve)



Slika 4.5: Nastavitve bloka PID_compact (skaliranje vhodnega signala)

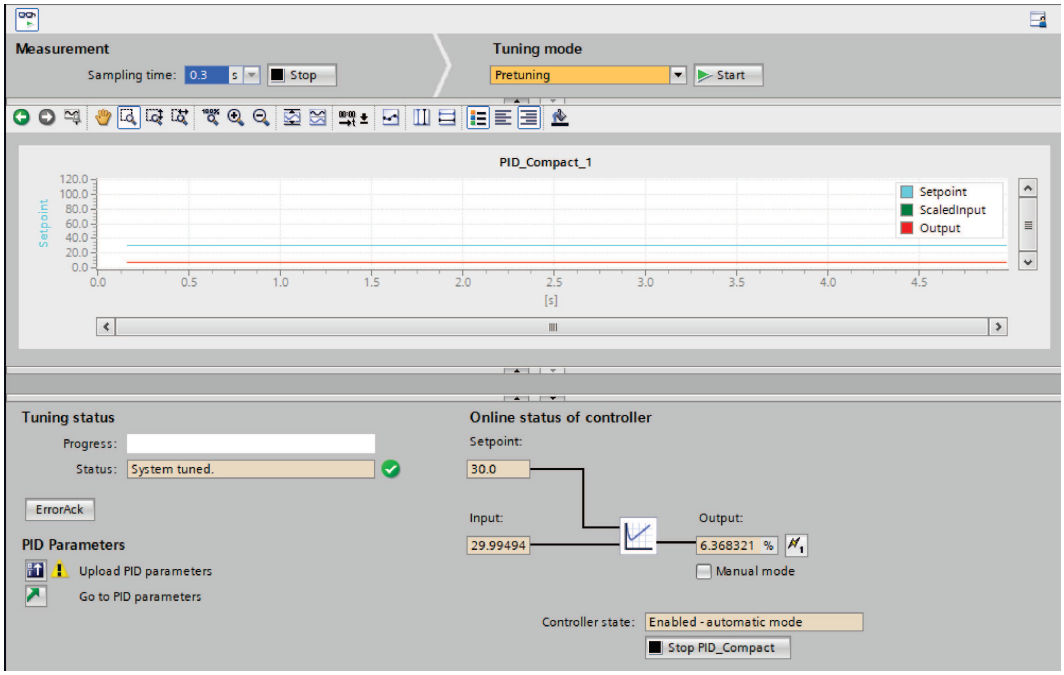
4.2.6 Povezava spremenljivk z vhodi in izhodi bloka PID_compact

Večina spremenljivk, ki jih bomo uporabljali za izdelavo vodenja je medsebojno dostopna tako bloku Cyclic Interupt OB 30 z vključenim PID_compact kot napravi HMI. Zaradi tega je potrebno na blok PID_compact povezati le vhodni in izhodni signal kot prikazuje slika 4.1.

4.2.7 Uporaba bloka PID_compact

Za uporabo bloka PID_compact uporabimo pot **PLC1[CPU 1214C AC/DC/Rly]** > **Technology objects** > **PID_compact** > **Commissioning**. Slika 4.6 prikazuje okno, ki omogoča upravljanje z regulatorjem.

Orodje zaženemo s klikom na gumb **Start** zgoraj levo (pod opcijo **Measurement**). Pri tem lahko nastavimo tudi čas vzorčenja, s katerim določimo čas osveževanja podatkov na zaslonu. Če orodje deluje, se okvir okna obarva oranžno. Gumb **Start**



Slika 4.6: Uporaba bloka PID_compact

zgoraj desno (pod opcijo **Tuning mode**) se uporablja za samonastavitev parametrov regulatorja.

V sredini okna se prikazujejo signali reference, skaliranega vhoda v regulator (skaliran izhod procesa) ter izhoda regulatorja (regulirna veličina oz. vhod v proces). Pod njimi se prikazujejo še stanje regulatorja in morebitne napake, ter trenutne vrednosti reference, vhoda in izhoda. Na tem mestu lahko vklopimo tudi ročni način delovanja regulatorja s klikom na polje **Manual mode**, ročno vnesemo vrednost izhoda regulatorja ter pošljemo vrednost na krmilnik s klikom na .

Trenutno stanje regulacijskega algoritma se izpisuje v polju **Controller state**. Pod njim lahko ročno zaženemo ali zaustavimo regulator.

V primeru, da smo za nastavljanje parametrov regulatorja uporabili samonastavitveni algoritem, je potrebno po kočani nastavitvi parametre poslati krmilniku s klikom na gumb **Upload PID parameters** .

4.2.8 Izgradnja uporabniškega vmesnika HMI

Uporabniški vmesnik HMI se uporablja za komunikacijo med operaterjem ter krmilnikom. S pomočjo uporabniškega vmesnika je operaterju omogočen vpogled v delovanje sistema, hkrati pa lahko vanj tudi posega. Običajno je vmesnik prikazan na uporabniškem zaslonu, ki je del operaterskega pulta. V našem primeru bomo namesto

dejanskega zaslona uporabili simulacijo, ki jo omogoča okolje TIA Portal.

Uporabniški vmesnik dodamo v projekt s klikom na gumb **Add new device** v levem stranskem meniju. V prikazanem oknu izberemo opcijo **HMI** $>$ **9"display** $>$ **KTP 900 Basic** $>$ **6AV2 123-2JB03-0AX0**. Odpre se okno za dodajanje novega HMI-ja, v katerem moramo določiti, s katerim krmilnikom bo povezan vmesnik. Izberemo krmilnik, ki smo ga predhodno dodali v projekt ter potrdimo s klikom na OK. S tem se v projektu prikaže nova naprava HMI.

Za izdelavo uporabniškega vmesnika je potrebno ustvariti sliko zaslona, ki se bo prikazovala na vmesniku. Glavna slika (**Root screen**) je že ustvarjena, če želimo ustvariti dodatne slike, pa to storimo s klikom na **Add new screen**. S klikom na izbrano sliko lahko pričnemo z izgradnjo izgleda vmesnika. Na voljo so različni predpripravljeni elementi, npr. gumbi, stikala, vnosna polja, prikazi signalov, prikazi alarmov, simboli procesnih elementov itd. Dostopni so na desnem stranskem meniju. Za potrebe izdelave vmesnika bomo primarno potrebovali naslednje elemetne: prikaz signalov, prikaz alarmov, gumbe ter vnosna polja.

V nadaljevanju je za primer prikazana konfiguracija gumba ter prikaza signalov.

Konfiguracija gumbov

V desnem stranskem meniju poiščemo gumb in ga povlečemo na sliko uporabniškega vmesnika. Pri vsakem gumbu moramo definirati kaj se zgodi pri pritisku nanj, za kar ima vsak takšen element namenjene različne dogodke, npr. klik (Click), pritisk (Press), spust (Release), itd. Izbira primernega dogodka je odvisna od tega, kaj želimo s pritiskom gumba doseči. Naprimer, za vklop samonastavitvenega algoritma regulatorja je konfiguracija gumba naslednja:

- **Properties** $>$ **Events** $>$ **Press**

  Nastavimo dogodke:

  - **SetTag** (Tag(Output): **Mode**, Value: **1**)
  - **SetBit** (Tag(Input/Output): **ModeActivate**)

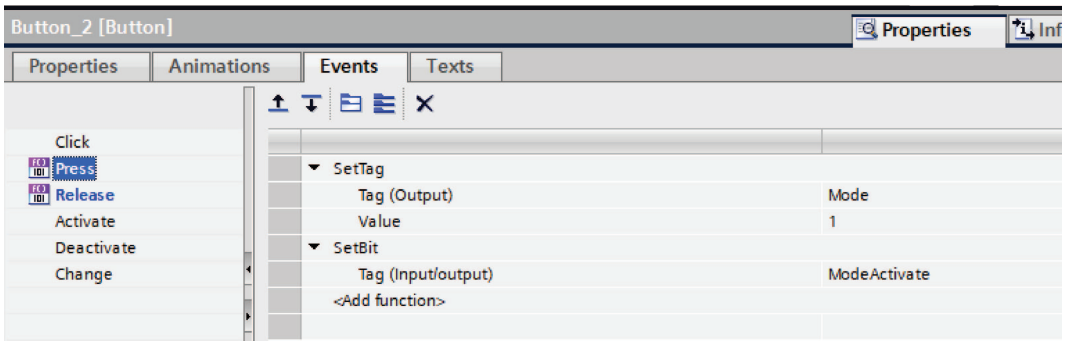
- **Properties** $>$ **Events** $>$ **Release**:

  Nastavimo dogodek:

  - **ResetBit** (Tag(Input/Output): **ModeActivate**)

Z ukazom **SetTag** nastavimo vrednost spremenljivke **Mode** na 1, kar pomeni izbiro stanja regulatorja za prednastavitev parametrov regulatorja (pretuning). Dejanski vklop tega stanja sprožimo s pulzom na spremenljivki **ModeActivate**. Za izvedbo

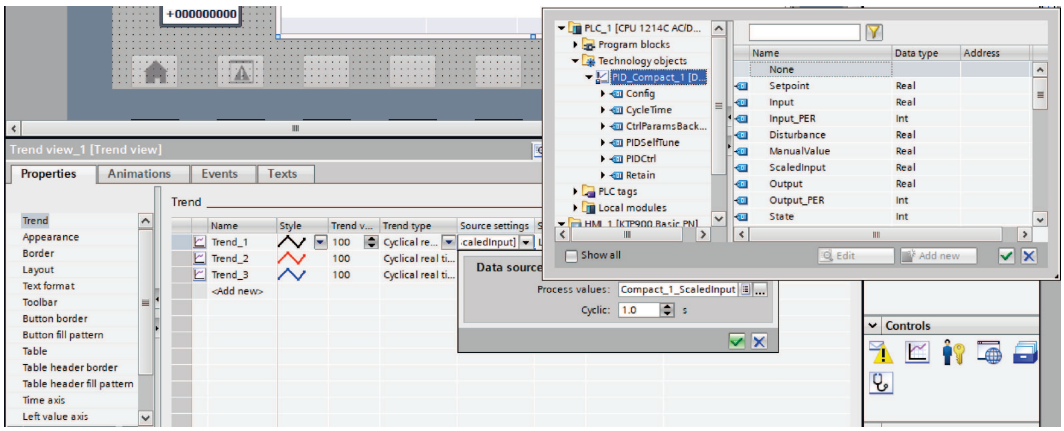
pulza uporabljamo ukaza **SetBit** ter **ResetBit**. Z ukazom **SetBit** postavimo vrednost spremenljivke **ModeActivate** na 1. Z ukazom ResetBit pa postavimo vrednost spremenljivke **ModeActivate** nazaj na 0. S takšno nastavitvijo gumba se ob pritisku nanj sproži samonastavitveni algoritem. Slika 4.7 prikazuje konfiguracijo gumba za vklop prednastavitve parametrov regulatorja.



Slika 4.7: Konfiguracija gumba

Konfiguracija prikaza signalov

V desnem stranskem meniju poiščemo prikaz signalov in ga povlečemo na sliko uporabniškega vmesnika. Nadalje moramo definirati, kateri signali naj se izrisujejo na prikazu. V zavihku **Properties** izberemo opcijo **Trend**. V seznam vseh signalov dodamo novega s klikom na **Add new**. Pri tem lahko izbiramo barvo s katero se bo signal izrisoval na prikazu, območje prikazovanja, tip prikazovanja itd. Pri tem je pomembno, da na dodan element povežemo dejanski signal, ki ga želimo prikazovati (**Source settings**) kot prikazuje slika 4.8



Slika 4.8: Konfiguracija prikaza signalov

Simulacija uporabniškega vmesnika

V primeru, da fizičnega zaslona za prikaz uporabniškega vmesnika nimamo, lahko prikaz le-tega izvedemo s pomočjo simulacije. To storimo tako, da zgrajen uporabniški vmesnik najprej prevedemo (Compile) ter kliknemo na ikono  . S tem se na zaslonu računalnika pojavi novo okno, ki simulira prikaz uporabniškega vmesnika z vso funkcionalnostjo.

4.3 Izvedba regulacije

V okviru vaje bomo izvedli enostavno regulacijo procesa z enim vhodom in enim izhodom.

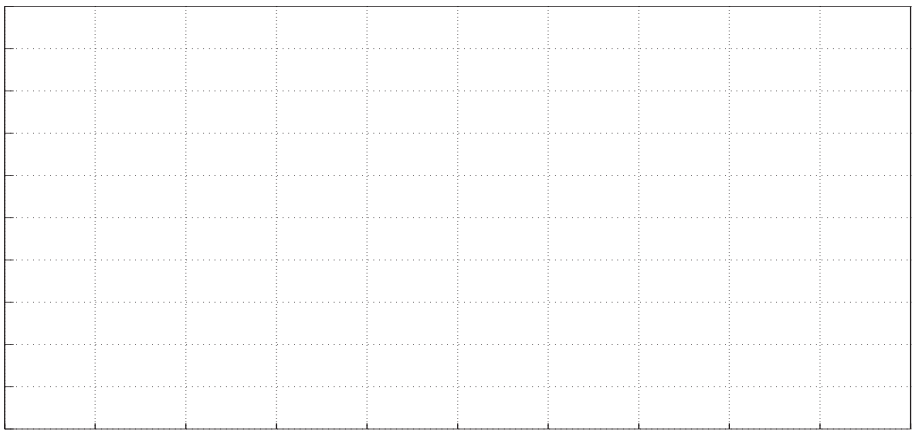
4.3.1 Preizkus delovanja regulatorja in povezava s procesom

Najprej bomo preizkusili delovanje regulatorja, programskega orodja za programiranje in pripadajoče komunikacijske povezave. V ta namen s programom TIA Portal ustvarite nov projekt in po prej opisanih korakih izdelajte program, ki bo omogočal uporabo bloka PID_compact ter ustrezno komunikacijo med krmilnikom in napravo. Program naložite na krmilnik in s pomočjo orodja **Commissioning** preverite pravilnost delovanja sheme.

V naslednjem koraku bomo preverite ustreznost delovanja naprave ter njene odzive na vhodne signale. To storite v orodju **Commissioning** tako, da regulator postavite v **ročni način** delovanja in spreminjate vrednost na njegovem izhodu. Hkrati na grafu opazujte odziv naprave. Pri tem bodite pozorni na območje vhodno/izhodnih signalov, t.j. pri kolikšni vrednosti izhoda regulatorja (vhoda na proces) je vhod regulatorja (izhod procesa) še v mejah merilnega območja.

4.3.2 Merjenje odprtozančnega odziva

1. Izmerite odprtozančni odziva procesa. Odziv izmerite tako, da sistem najprej pripeljete v izbrano delovno točko, potem pa naredite stopničasto spremembo izhoda regulatorja (vhoda v proces) in opazujete odziv. Skicirajte izmerjeni odziv na spodnjem diagramu. Narišite tako vzbujanje kot odziv procesa. Označite skalo obeh koordinatnih osi diagrama:



4.3.3 Nastavitev parametrov regulatorja in merjenje zaprtozančnega odziva

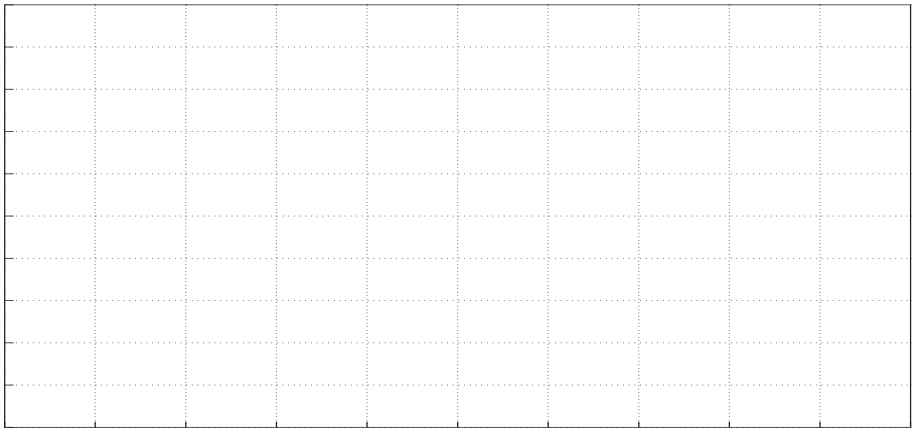
1. Na podlagi izmerjenega odprtozančnega odziva izberite ustrezno strukturo regulatorja, t.j. P, PI ali PID, ter nastavite pripadajoče parametre tako, da bo regulirana veličina čim bolje sledila spremembam reference: izberite ustrezno metodo za nastavljanje parametrov regulatorja (poglavje 4.3.5), utemeljite izbiro in zapišite rezultate.

   Izmerjeni parametri procesa: _____
   Izračunani parametri regulatorja: _____

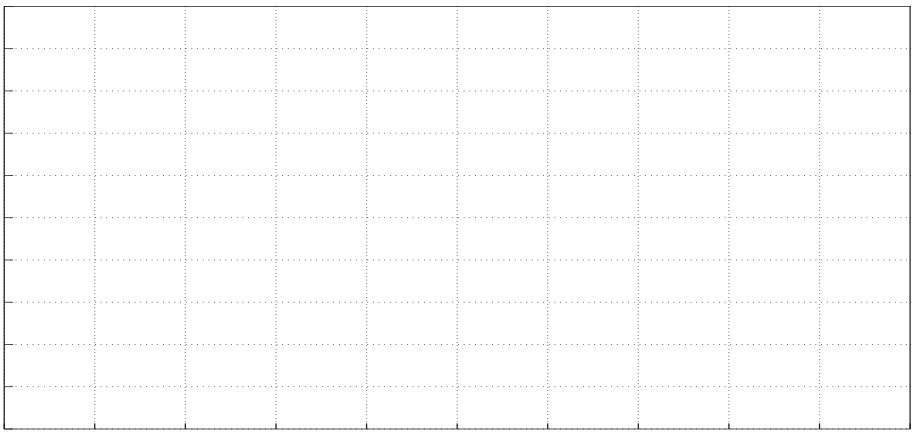
2. Prikažite dosežene rezultate - zaprtozančni odziv procesa ob spremembah reference in motnjah.

   Narišite izmerjeni odziv pri spremembi reference

   - prvi diagram naj prikazuje potek reference in regulirane veličine:

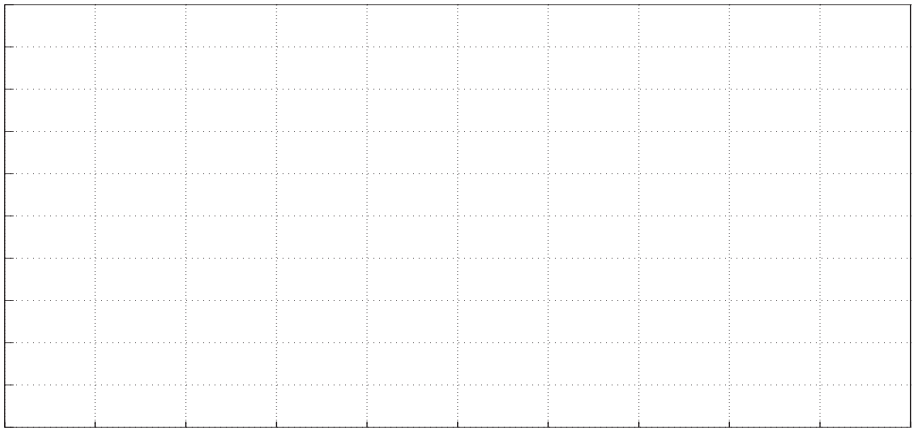


   - drugi diagram naj prikazuje potek regulirne veličine:

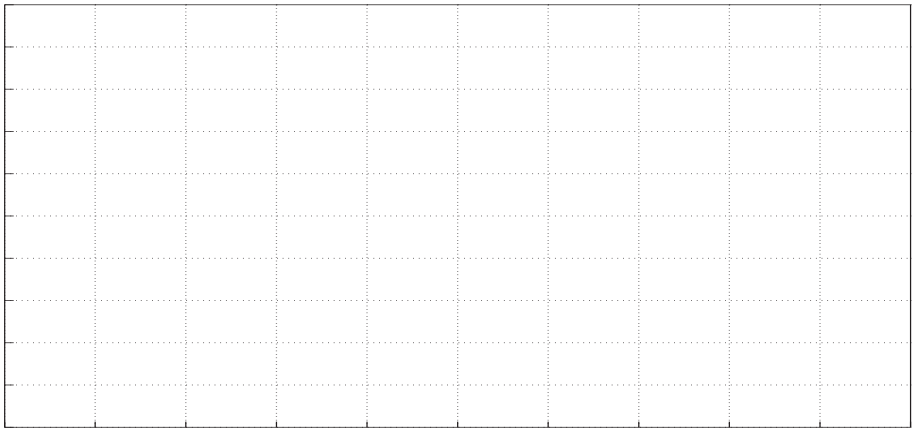


Na spodnja diagrama narišite še izmerjeni odziv pri motnji

- prvi diagram naj prikazuje potek reference in regulirane veličine:



- drugi diagram naj prikazuje potek regulirne veličine:



3. Pridobite parametre regulatorja z uporabo vgrajenega samonastavitvenega algoritma. Pri tem uporabite funkciji **Pretuning** ter **Fine tuning**. Ponovite vse zgornje poskuse (sprememba reference, motnja) ter v grafe dorišite ustrezne signale. Katera nastavitev regulatorja daje boljše rezultate? Odgovor utemeljite.

   Dobljeni parametri regulatorja z uporabo samonastavitvenega algoritma:

4.3.4 Izdelava uporabniškega vmesnika

Za izdelan projekt zgradite še uporabniški vmesnik. Vmesnik naj vsebuje naslednje 3 slike, in sicer glavno (Root screen), prikaz napak ter prikaz parametrov regulatorja.

Glavna slika naj vsebuje naslednje:

- izris signalov (referenca, izhod regulatorja, vhod regulatorja),
- izpis stanja regulatorja (0 - neaktiven, 1 - prednastavitev parametrov, 2 - fina nastavitev parametrov, 3 - avtomatski način, 4 - ročni način),
- vklop in izklop regulatorja,
- spreminjanje reference,
- preklop ročno/avtomatsko z možnostjo ročnega nastavljanja izhoda regulatorja,
- izpis napake (indikator),
- ustrezne gumbe za povezavo na ostale slike.

Slika za prikaz napak naj vsebuje naslednje:

- seznam vseh napak v delovanju,
- gumb za potrditev napak,
- ustrezne gumbe za povezavo na ostale slike.

Slika s parametri regulatorja naj vsebuje naslednje:

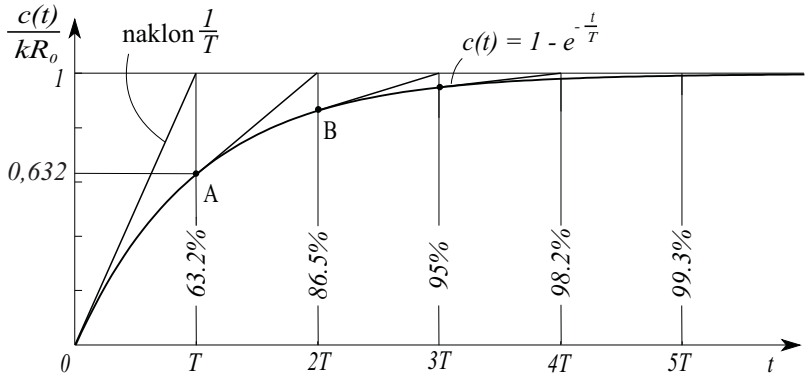
- možnost izbire prednastavitve ali fine nastavitve parametrov regulatorja,
- možnost ročnega vnosa parametrov regulatorja.
- ustrezne gumbe za povezavo na ostale slike.

DODATEK

4.3.5 Nastavitvena pravila

Metoda za P procese 1. reda

Za vodenje P procesa prvega reda lahko uporabimo P regulator (v tem primeru moramo predpisati želeno ojačenje zaprtozančnega sistema K_{zel} ali želeno časovno konstanto zaprtozančnega sistema T_{zel}) ali PI regulator (predpisati moramo T_{zel}), medtem ko je T časovna konstanta procesa 1. reda, K pa njegovo ojačenje. Ojačenje in časovno konstanto procesa odčitamo iz odziva sistema na stopnico kot prikazuje slika 4.9.



Slika 4.9: Merjenje odziva procesa na stopnico

Časovno konstanto sistema odčitamo iz presečišča tangente v koordinatnem izhodišču ter premice, ki podaja ustaljeno stanje sistema. Ojačenje procesa izračunamo kot kvocient spremembe izhodnega in vhodnega signala: $K = \Delta c/\Delta u$.

regulator	K_P	T_I	T_D
P	$\frac{T - T_{zel}}{KT_{zel}}$ ali $\frac{K_{zel}}{K(1 - K_{zel})}$	/	/
PI	$\frac{T}{T_{zel}K}$	T	/

Tabela 4.3: Nastavitvena pravila za P procese prvega reda

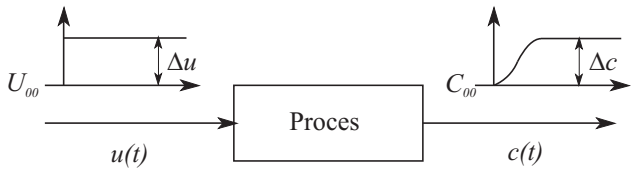
Gornja nastavitvena pravila za P regulator v tabeli lahko izpeljemo, če primerjamo želeno zaprtozančno prenosno funkcijo $G_{zel} = \frac{K_{zel}}{T_{zel}s+1}$ z izračunano zaprtozančno

prenosno funkcijo $G_z = \frac{K_P G}{1+K_P G}$, kjer je K_P proporcionalno ojačenje regulatorja in $G = \frac{K}{Ts+1}$ prenosna funkcija procesa. Iskano ojačenje K_P lahko dobimo s primerjavo časovnih konstant G_{zel} in G_z ali s primerjavo ojačenj G_{zel} in G_z. Ker ima P regulator na P procesu prvega reda vedno pogrešek v ustaljenem stanju, je smiselna izbira $0 < K_{zel} < 1$.

Nastavitvena pravila za PI regulator izpeljemo tako, da izberemo želeno zaprtozančno prenosno funkcijo $G_{zel} = \frac{1}{T_{zel}s+1}$ (I del regulatorja kompenzira pogrešek v ustaljenem stanju, torej $K_{zel} = 1$). Iz zaprtozančne prenosne funkcije $G_z = \frac{G_{PI} G}{1+G_{PI} G}$ izrazimo $G_{PI} = \frac{G_z}{G-G_z G}$, kjer za G_z vstavimo G_{zel}, in dobimo $G_{PI} = \frac{T}{T_{zel} K}(1 + \frac{1}{Ts})$.

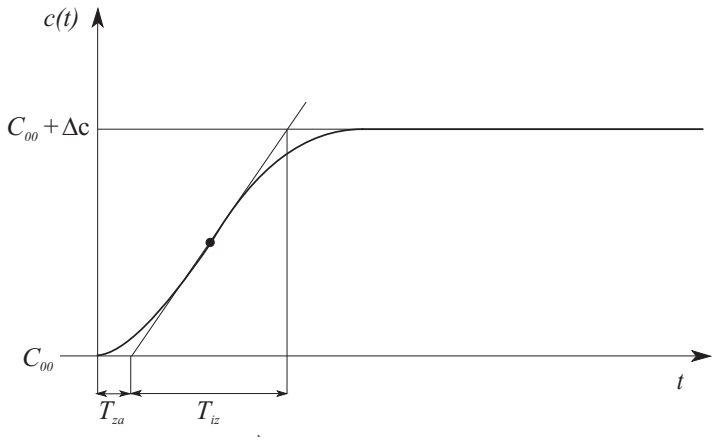
Metoda Chien-Hrones-Reswick za procese 2. reda

Metoda spada med t.i. odprtozančne metode, kar pomeni, da je potrebno opraviti eksperiment na odprtozančnem procesu (brez povratne zanke). Primerna je za proporcionalne procese višjega reda, ki morajo biti nekoliko nadkritično dušeni (brez prenihaja pri prehodnem pojavu pri odzivu na stopnico). Proces je potrebno vzbuditi s stopničasto spremembo vhoda in meriti odziv kot prikazuje slika 4.10.



Slika 4.10: Merjenje odziva procesa na stopnico

Odziv na stopnico lahko izmerimo bodisi na realnem sistemu ali pa na njegovem modelu, odvisno ali realni sistem dopušča tovrstne posege v njegovo delovanje ter ali model procesa sploh obstaja. Karakteristični odziv, ki je primeren za uporabo tega kriterija, t.j. brez prenihaja, prikazuje slika 4.11.



Slika 4.11: Odziv proporcionalnega procesa

Iz odziva, ki ga prikazuje slika 4.11 odčitamo oz. izračunamo naslednje parametre: ojačenje - K, čas zaostajanja - T_{za} ter čas izravnave - T_{iz}. Ojačenje procesa K izračunamo kot kvocient sprememb izhodnega in vhodnega signala v ustaljenem stanju:

$$K = \frac{\Delta c}{\Delta u}. \tag{4.1}$$

V prevojni točki odziva narišemo tangento ter s pomočjo presečišč tangente z abscisо in premico $c(t) = C_{00} + \Delta c$ določimo še čas zaostajanja T_{za} in čas izravnave T_{iz}. Na ta način dobimo tudi zelo poenostavljen model, ki ga opišemo s procesom 1. reda z mrtvim časom:

$$\frac{C(s)}{U(s)} = \frac{K\, e^{-T_{za}s}}{T_{iz}s + 1}. \tag{4.2}$$

Priporočila, ki so jih izdelali Chien, Hrones in Reswick upoštevajo ali bo regulacijski sistem primarno deloval v regulacijskem ali v sledilnem načinu delovanja (ali je vhodni signal referenca ali motnja na vhodu v proces), kot tudi kakšen naj bo odziv procesa. Izbiramo lahko ali bo odziv imel 20% prevzpon ali pa bo aperiodičen s čim krajšim umiritvenim časom. Priporočila za nastavitve podaja tabela 4.4.

Tabela 4.4: Uglaševanje PID regulatorja s pravili Chien-Hrones-Reswick

Regulator		Aperiodični odziv z najkrajšim umiritvenim časom		Najkrajši umiritveni čas z 20% prevzponom	
		motnja	referenca	motnja	referenca
P	K_P	$\frac{0.3}{K}\frac{T_{iz}}{T_{za}}$	$\frac{0.3}{K}\frac{T_{iz}}{T_{za}}$	$\frac{0.7}{K}\frac{T_{iz}}{T_{za}}$	$\frac{0.7}{K}\frac{T_{iz}}{T_{za}}$
PI	K_P	$\frac{0.6}{K}\frac{T_{iz}}{T_{za}}$	$\frac{0.35}{K}\frac{T_{iz}}{T_{za}}$	$\frac{0.7}{K}\frac{T_{iz}}{T_{za}}$	$\frac{0.6}{K}\frac{T_{iz}}{T_{za}}$
	T_I	$4T_{za}$	$1.2T_{iz}$	$2.3T_{za}$	T_{iz}
PID	K_P	$\frac{0.95}{K}\frac{T_{iz}}{T_{za}}$	$\frac{0.6}{K}\frac{T_{iz}}{T_{za}}$	$\frac{1.2}{K}\frac{T_{iz}}{T_{za}}$	$\frac{0.95}{K}\frac{T_{iz}}{T_{za}}$
	T_I	$2.4T_{za}$	T_{iz}	$2T_{za}$	$1.35T_{iz}$
	T_D	$0.42T_{za}$	$0.5T_{za}$	$0.42T_{za}$	$0.47T_{za}$

Tabela 4.5: Nastavitvena pravila po metodi Chien-Hrones-Reswick

Predno se lotimo nastavitve parametrov regulatorja, se odločimo kakšen regulator bomo uporabili. Izbiramo lahko med P, PI ter PID strukturami. Nadalje se odločimo ali bo sistem deloval v sledilnem ali regulacijskem delovanju. Na koncu se moramo odločiti še ali naj bo odziv sistema pri stopničasti referenci periodičen (s prevzponom) ali aperiodičen (brez prevzpona). V tabeli 4.4 tako poiščemo polje, ki ustreza vsem trem zahtevam in izračunamo ustrezne parametre po pripadajočih enačbah.