Priemyselná automatizácia

Fakulta elektrotechniky a informatiky

Pavol Fedor a Daniela Perduková

Košice, 2016

Táto učebnica vznikla za finančnej podpory projektu KEGA 011TUKE-4/2013. Je určená študentom inžinierskeho štúdia študijného programu Automatizácia mechatronických systémov pre výučbu v predmete Riadenie montážnych liniek pomocou PLC, ako aj v ďalších predmetoch, kde sa využívajú riadiace systémy na báze PLC.

NÁZOV: Priemyselná automatizácia
AUTOR: prof. Ing. Pavol Fedor, PhD., prof. Ing. Daniela Perduková, PhD.
RECENZENT: doc. Ing. Ján Jadlovský, PhD., Ing. Peter Macko, PhD.
VYDAVATEĽ: Technická univerzita v Košiciach
ROK: 2016
VYDANIE: prvé
NÁKLAD 50 ks
ROZSAH: 102 strán
ISBN 978-80-553-2477-7

Obsah

Zoznam obrázkov

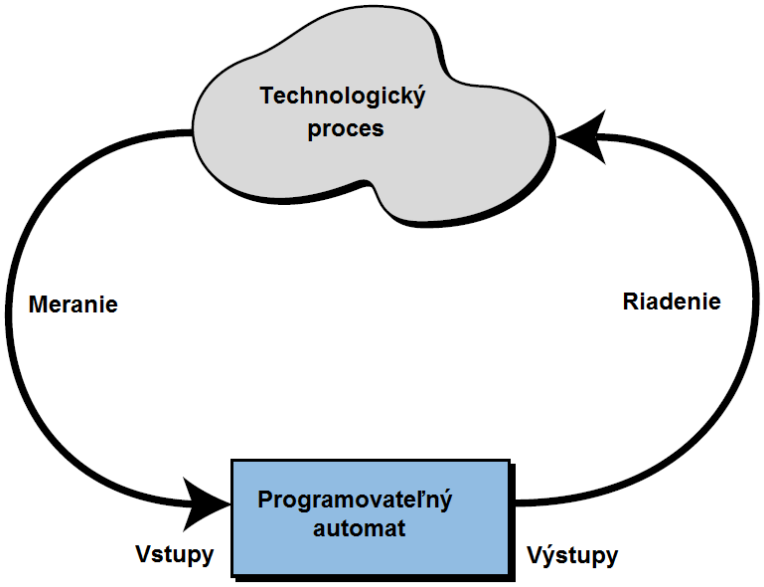
Zoznam skratiek

PA	programovateľný automat
PLC	Programmable Logic Controller (programovateľný logický automat)
CPU	Central Procesing Unit (centrálna procesorová jednotka)
IO	vstupy/výstupy
OP	operátorský panel
IS	inžinierska stanica
CPU	centrálna procesorová jednotka
HW	hardvér
SW	softvér
UPA	User Program Area (oblasť pre tvorbu užívateľských programov)
OT	Output table (tabuľka výstupov)
IT	Input table (tabuľka vstupov)
LD	Ladder Diagram (líniová schéma)
FBD	Function Block Diagram
IL	Instruction List (zoznam inštrukcií)
SFC	Sequential Function Chart (sekvenčný diagram)
Interface	rozhranie

ÚVOD

Zvyšujúca sa automatizácia technologických procesov si vyžaduje pre svoju realizáciu oveľa efektívnejšie technické prostriedky. Ešte v nedávnej minulosti postačovalo napr. pri automatizácii pohonu niekoľko tlačidiel, žiaroviek, stýkačov a jednoduchá miestna skrinka. Obsluha cez tlačidlo pohon naštartovala, podľa signálnej žiarovky videla jeho stav a niekoľko málo pomocných kontaktov stýkačov zabezpečilo základné blokácie pohonu. Postupne však začal narastať počet pohonov a meraní, ktoré bolo potrebné v danej technológii zahrnúť do automatizačných algoritmov. To vyplynulo jednak z rozširovania a skvalitňovania jednotlivých technológií, jednak zo stále rastúcich požiadaviek užívateľov na ich automatizáciu a vizualizáciu. Klasická reléovo-stýkačová automatika prestávala vyhovovať týmto požiadavkám, jednak pre svoju spoľahlivosť (ktorá silne klesá s rastúcim počtom použitých reléovo-stýkačových prvkov), jednak pre problematické prispôsobovanie sa stále rýchlejšiemu inovačnému cyklu.
S rozvojom mikropočítačovej techniky začala vznikať technická možnosť realizovať aj zložité automatizačné algoritmy programovými prostriedkami. Pretože klasický mikropočítač nemal periférne signály prispôsobené potrebám technológie, začali sa vyvíjať špecializované mikropočítače, ktorých technické prostriedky aj programové vybavenie boli určené predovšetkým pre riadenie technologických procesov. Takéto mikropočítače sa začali nazývať **programovateľné automaty**. Programovateľné automaty sú teda v podstate mikropočítače, špeciálne prispôsobené svojím hardvérom a softvérom pre riadenie technologických procesov logického, resp. kombinovaného (t.j. logického a analógového) typu.

Všetky odvetvia dnešného priemyslu (od výroby elektriny cez automobilový priemysel až po výrobu potravy) dnes potrebujú kvalitne a flexibilne riadiť svoju produkciu. Na tento cieľ veľmi často využívajú programovateľné automaty (PA), ktorých štrukturálna schéma je uvedená na obr.1.



Obr.1: Štrukturálna schéma riadenia technologického procesu pomocou PA

PA sa začali vyvíjať v 70-tych rokoch minulého storočia s primárnym cieľom eliminovať vysoké náklady, spojené s nevýhodami reléových riadiacich systémov (malá flexibilita, poruchovosť a pod.). Základné požiadavky boli:

- Nižšia cena ako porovnateľný reléový systém.
- Vhodnosť pre nasadenie do priemyselného prostredia.
- Ľahko modifikovateľný vstupno-výstupný (IO) interface.
- Možnosť komunikácie a prenosu údajov do nadradeného centrálneho systému.
- Čo najjednoduchšia forma modifikácie vlastností PA.

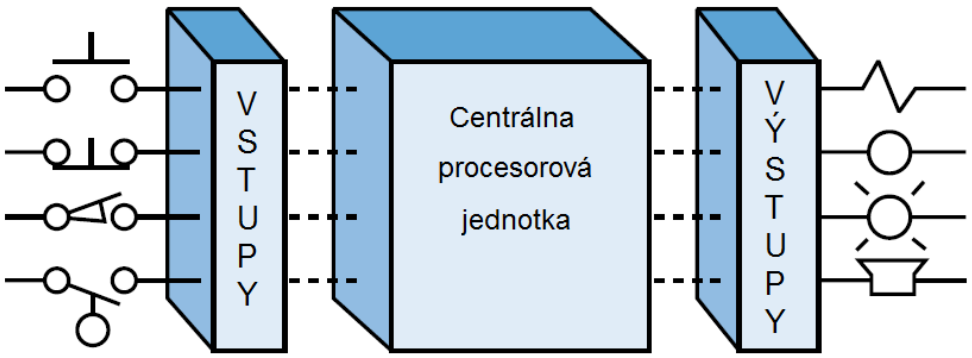
Táto učebnica je určená študentom inžinierskeho štúdia študijného programu Automatizácia mechatronických systémov pre výučbu v predmete Riadenie montážnych liniek pomocou PLC, ako aj v ďalších predmetoch, kde sa využívajú riadiace systémy na báze PLC a má za úlohu oboznámiť študentov s týmto univerzálnym a výkonným technickým prostriedkom používaným pre riadenie technologických procesov.

1 FILOZOFIA SYSTÉMOV S PROGRAMOVATEĽNÝMI AUTOMATMI

Aby sa vyhovelo vyššie uvedeným požiadavkám (hlavne rýchlej a pohodlnej modifikovateľnosti riadiaceho systému), zvolil sa mikropočítač ako jadro tohto systému.

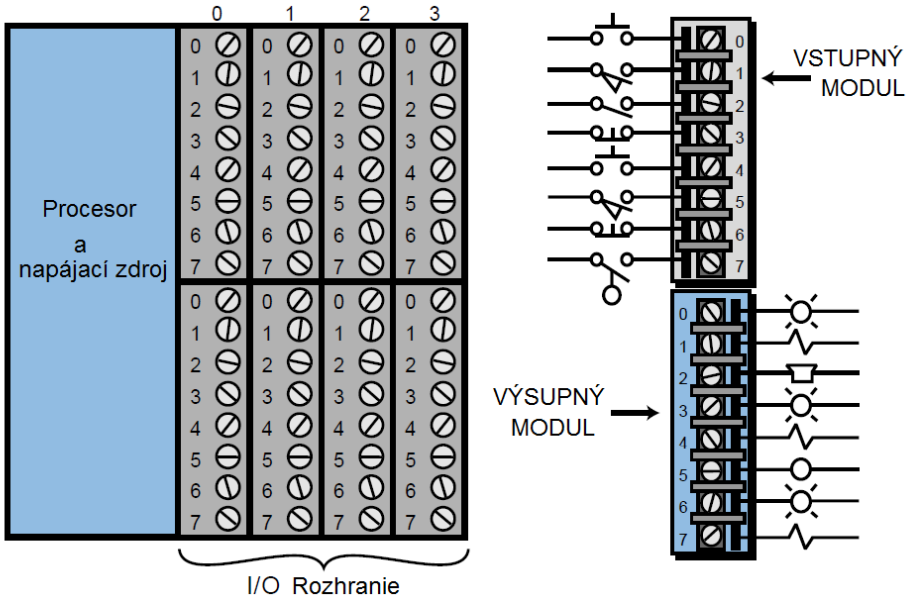
Definícia: **Programovateľný automat** je mikropočítač so špeciálne prispôsobeným hardvérom a softvérom pre riadenie technologických procesov (logického, resp. kombinovaného typu).

PA má dve hlavné hardvérové časti – **CPU** (Central Procesing Unit) a **IO interface** a je ukázaný na nasledujúcom obrázku (Obr. 1.1):



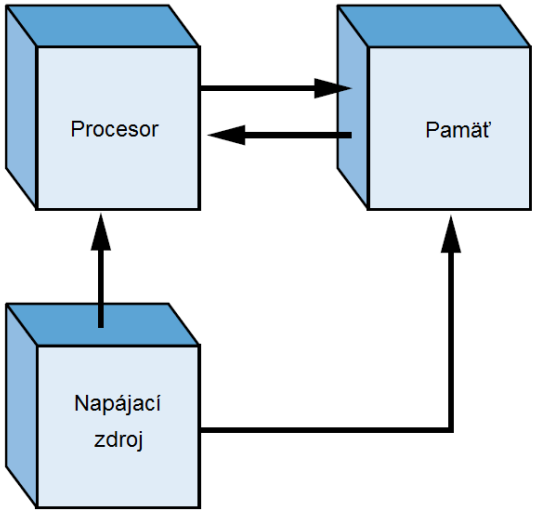
Obr. 1.1 Hardvérová štruktúra PA

IO interface má najčastejšie nasledujúcu štruktúru (Obr. 1.2):



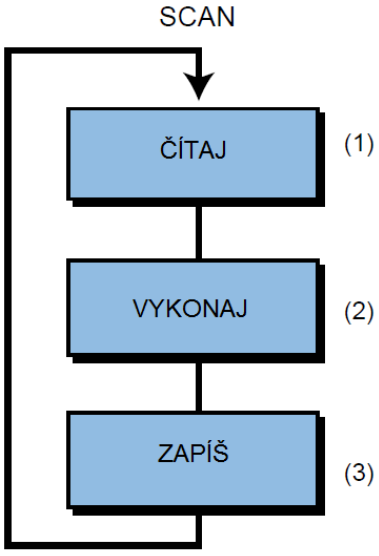
Obr. 1.2 Schematické znázornenie prepojenia PA na proces

Bloková schéma CPU pozostáva z **procesora**, **pamäte** a **napájania** (Obr. 1.3). Pretože sa v praxi vyskytujú PA s rôznym výkonom a rôznym počtom vstupov a výstupov, napájanie sa obvykle realizuje pomocou externého napájacieho modulu, ktorý vyrába všetky úrovne napätí (5V, 24V, 230V) pre všetky HW komponenty PA.



Obr. 1.3 Bloková schéma CPU programovateľného automatu

Počas svojej činnosti PA v zásade vykonáva cyklicky tri procesy – načítanie vstupov z vstupnej oblasti pamäte, spracovanie naprogramovaných príkazov a nastavenie oblasti pamäte pre výstupy. Jednému úplnému vykonaniu týchto troch procesov hovoríme **pracovný cyklus PA** (SCAN) (Obr. 1.4).

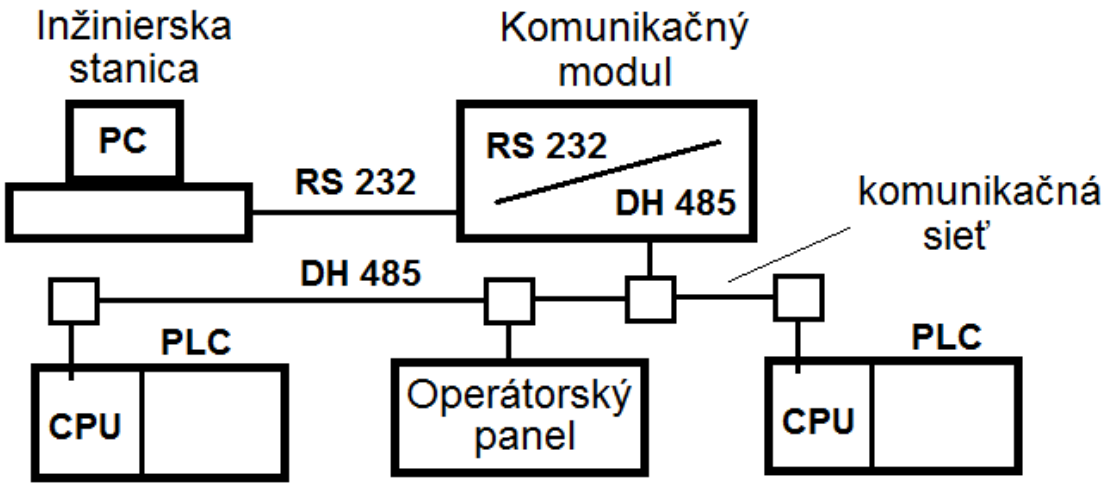


Obr. 1.4 Pracovný cyklus PA

Pre reálnu činnosť PA sú potrebné ešte ďalšie zariadenia, ktoré síce nie sú priamo súčasťou PA, ale umožňujú nasadiť PA do prevádzky, realizovať jeho údržbu a prepojenie s inými systémami (nadriadený riadiaci systém, operátor, iný PA a pod.). Preto sú PA sú obvykle skladané do systémov, ktoré obsahujú tieto hlavné prvky:

- Samotné **PA** rôzneho typu.
- **Operátorské panely** (OP) pre styk s obsluhou.
- **Inžinierske stanice** (IS) (dnes štandardné PC + programovacie prostredie pre PA).
- **Komunikačné moduly** (zabudované alebo externé).
- Tlačidlové a signalizačné moduly (ako náhrada klasických ovládacích pultov).
- Špeciálne moduly (napr. pre distribuovaný zber a prenos údajov z technológie).
- Ručné programátory (pre servisné účely).

Príklad typickej konfigurácie systému s PA je ukázaný na Obr. 1.5.



Príklad PLC systému s dvoma komunikačnými linkami
(DH 485 a ETHERNET)

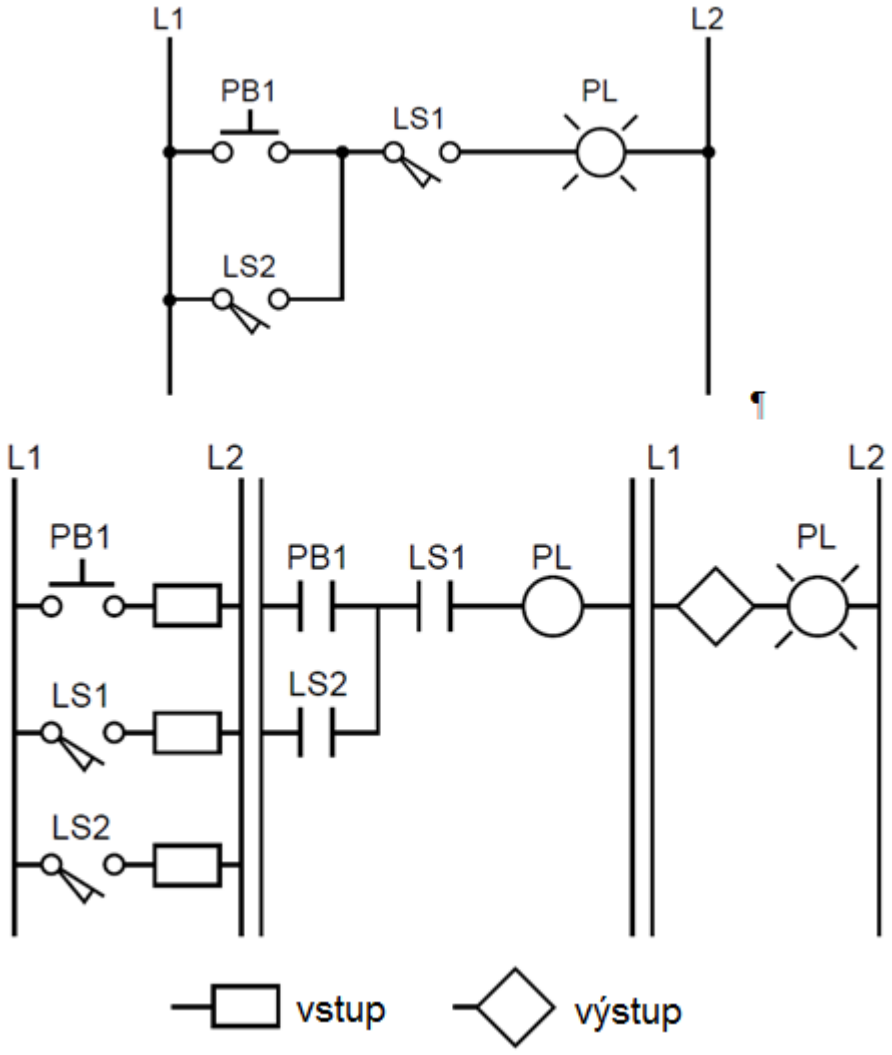
Obr. 1.5 Príklad typickej konfigurácie PA

Komunikácia medzi prvkami systému sa vykonáva:

- Po štandardizovaných komunikačných sieťach (RS232, RS485, Eethernet, CAN, MODBUS, Profibus...)
- Po špecializovaných sieťach jednotlivých výrobcov (DH+, Powerlink, ...)

Tradičný spôsob zápisu činnosti reléových riadiacich systémov bola a je elektrická líniová schéma (Obr. 1.6). Zobrazuje HW prepojenia a logiku riadenia nejakého zariadenia. Tento spôsob popisu činnosti bol primárne implementovaný aj do programového vybavenia (softvéru) pre PA.

Príklad:

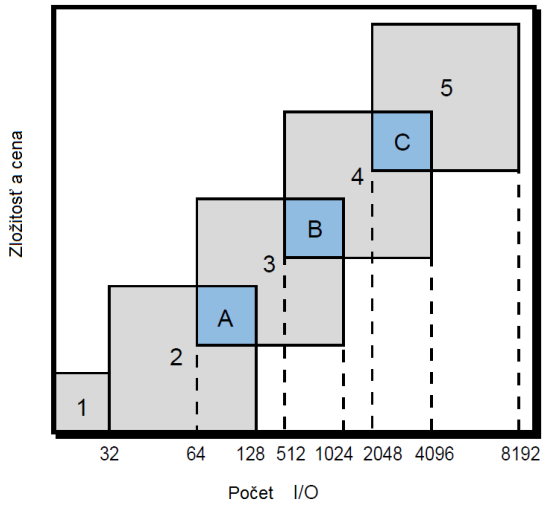


Obr. 1.6 Elektrická líniová schéma pre PA.

2 TECHNICKÉ PROSTRIEDKY PROGRAMOVATEĽNÉHO AUTOMATU

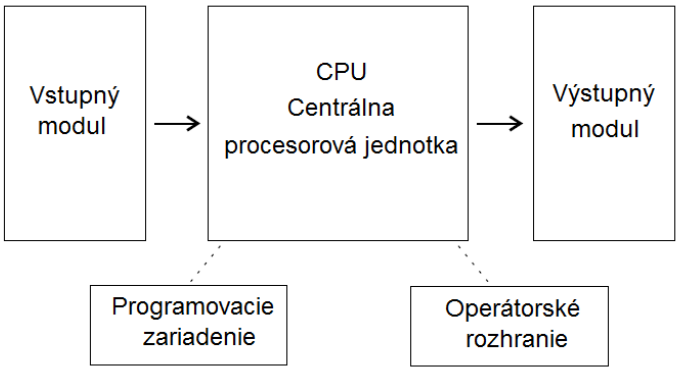
Podľa konštrukcie delíme PA na:
- Fixné (s pevne danou HW štruktúrou)
- Modulárne (s flexibilne upraviteľnou HW konfiguráciou v danom rozsahu)

PA delíme podľa veľkosti (počtu IO a zložitosti programu) približne takto (Obr. 2.1):
1. Mikro PA
2. Malé PA
3. Stredné PA
4. Veľké PA
5. Veľmi veľké PA (DCS – decentralizované riadiace systémy)



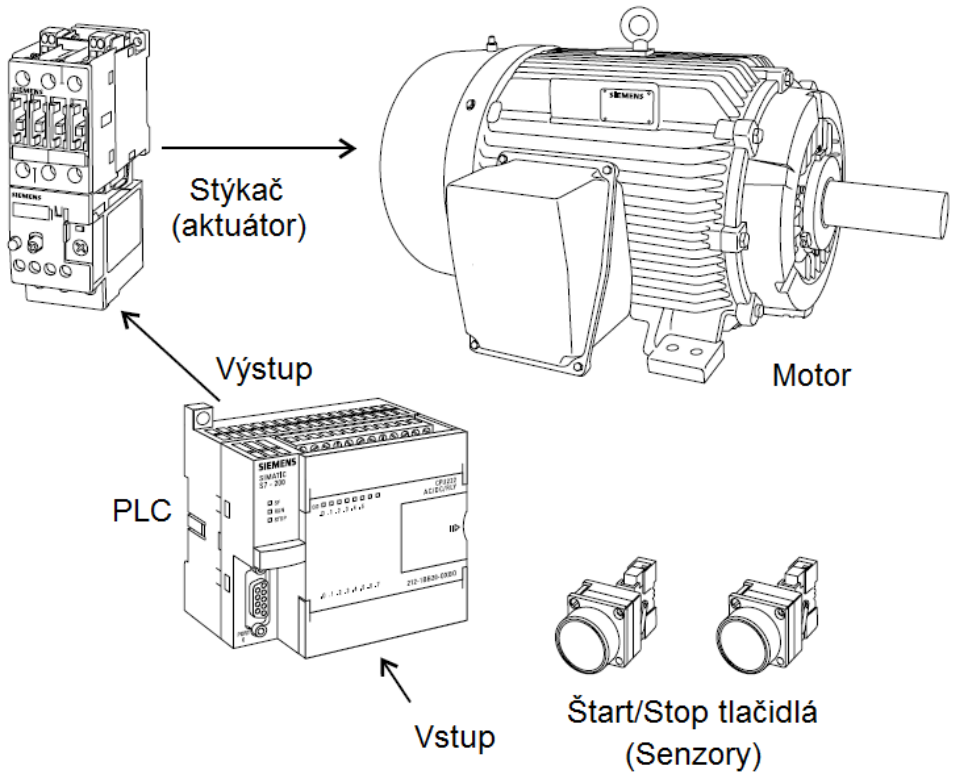
Obr. 2.1 Rozdelenie PA podľa veľkosti

Základná schematická predstava štandardného PA je uvedená na Obr.2.2.



Obr. 2.2 Schematické znázornenie základných technických prostriedkov PA

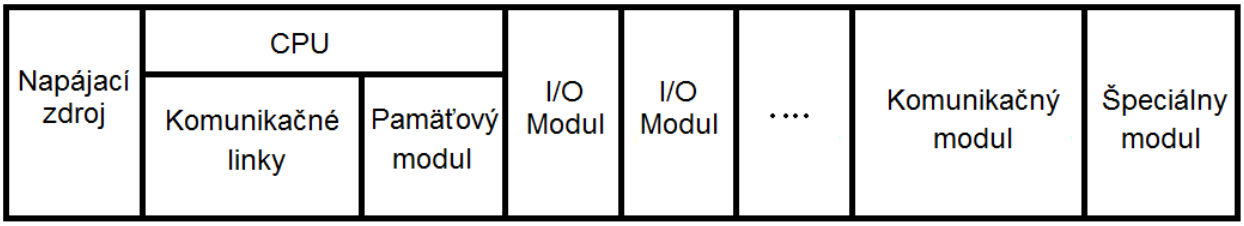
Na príklade z Obr. 2.3 vidíme, ako konkrétne môže vyzerať ovládanie a prípadne aj riadenie elektrického pohonu pomocou programovateľného automatu. Cez vstupné moduly PA načítava stav ovládacích tlačidiel pohonu, ktoré v tomto prípade slúžia ako operátorský interface. Cez svoje výstupné moduly PA ovláda aktuátor (napr. stýkač), ktorý pripája na motor elektrickú sieť.



Obr. 2.3 Príklad aplikácie PA ovládanie elektrického pohonu

Modulárny PA obsahuje najčastejšie nasledujúce komponenty (Obr. 2.4):

- CPU
- Napájacie moduly
- IO moduly
- Komunikačné moduly
- Špeciálne moduly

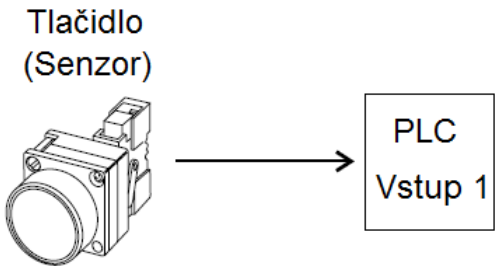


Obr. 2.4 Typická štruktúra modulárneho PA

Jednotlivé moduly sú fyzicky umiestnené buď do mechanického roštu (chassis) alebo sú prepájané konektormi, ktoré sú súčasťou každého modulu. Takto je zabezpečený prenos napájania na všetky moduly a aj komunikácia medzi nimi cez internú zbernicu.
Pretože PA je elektrické zariadenie a pracuje vnútorne s elektrickými signálmi (napätiami a prúdmi), musí IO interface transformovať fyzikálne signály z riadeného technologického procesu (teplotu, tlak, prietok, hladinu, rýchlosť, polohu ...) na štandardizované elektrické signály.

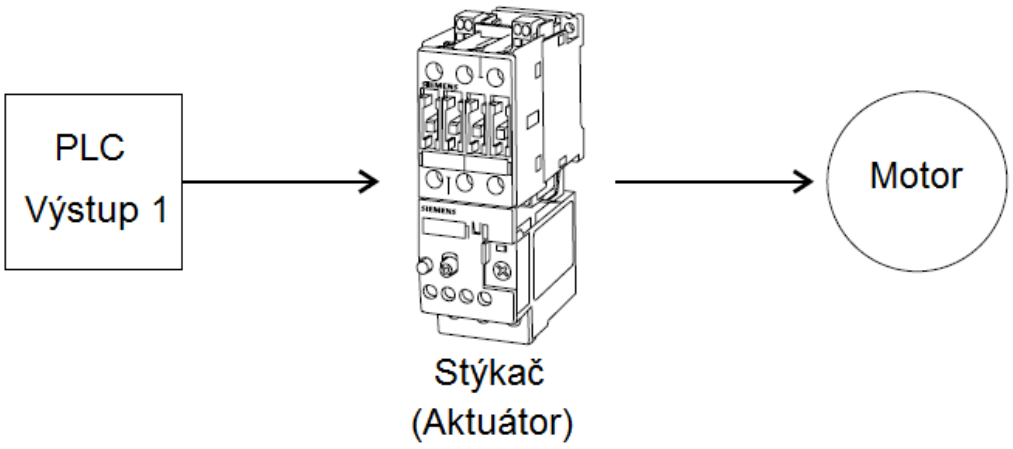
Vstupno-výstupné (IO) moduly programovateľných automatov sú prispôsobené elektrotechnickým prvkom (snímačom a akčným členom), ktoré sa štandardne vyskytujú pri riadení technologických procesov.

Snímač je zariadenie, ktoré konvertuje nejakú fyzikálnu podmienku na elektrický signál, použiteľný v PA (Obr. 2.5).



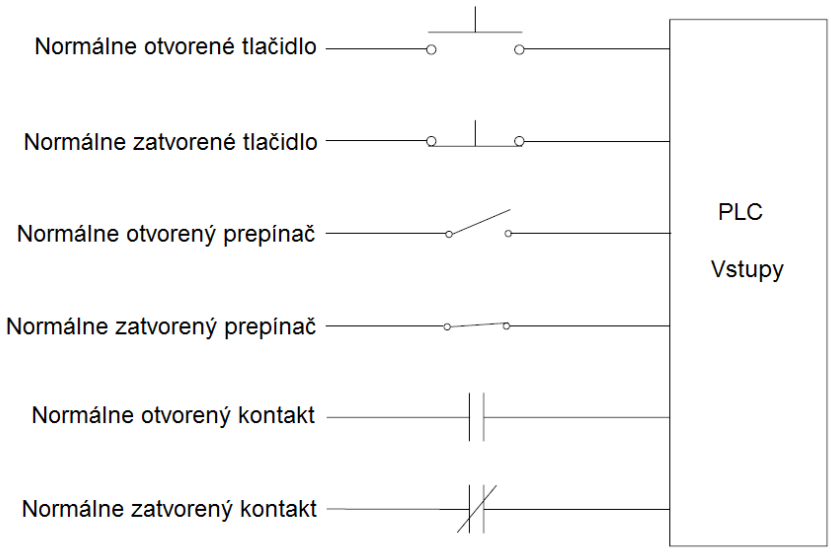
Obr. 2.5 Senzor

Aktuátor (akčný člen) je zariadenie, ktoré konvertuje elektrický signál z PLC na príslušnú fyzikálnu podmienku (Obr. 2.6).



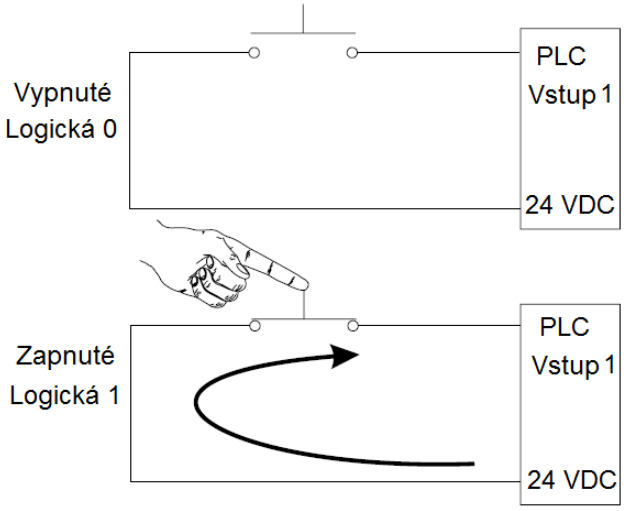
Obr. 2.6 Akčný člen, aktuátor

Diskrétny vstup (digitálny vstup) je taký vstup, ktorý nadobúda iba dve hodnoty (najčastejšie označované ON – logická 1 v PA a OFF – logická 0 v PA). Technicky je realizovaný napr. tlačidlom, prepínačom, koncovým spínačom, kontaktom a pod. (Obr. 2.7)



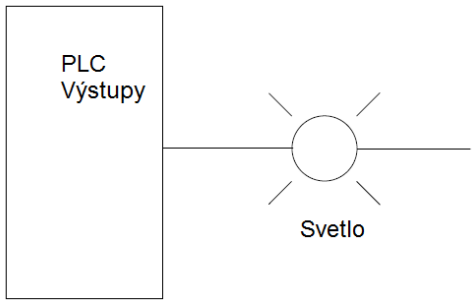
Obr. 2.7 Základné typy binárnych vstupov

Príklad zapojenia spínacieho tlačidla je na Obr. 2.8. Všimnime si, že toto zapojenie potrebuje pre svoju činnosť zdroj elektrického napätia.



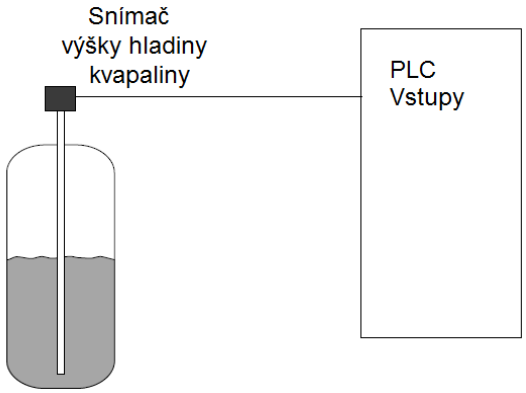
Obr. 2.8 Zapojenie tlačidla ako binárneho vstupu PA

Diskrétny výstup (digitálny výstup) je taký výstup, ktorý nadobúda iba dve hodnoty (ON a OFF). Technicky je realizovaný napr. solenoidom, stýkačom, kontaktným relé, žiarovkou a pod. (Obr. 2.9)



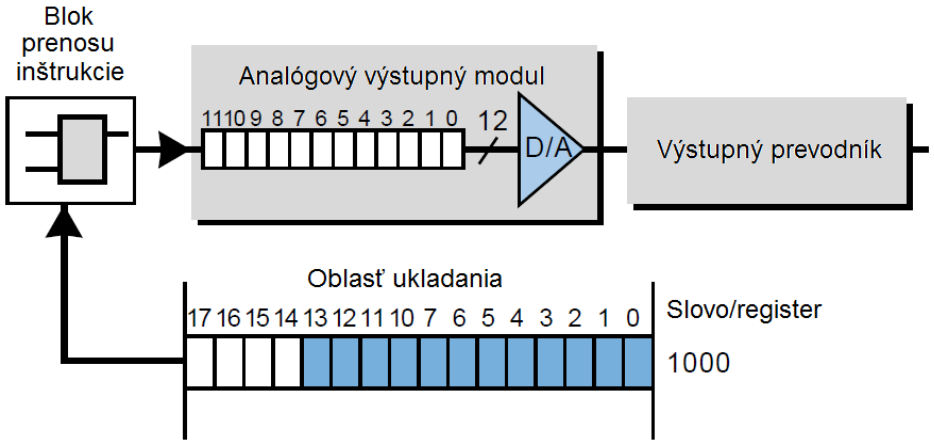
Obr. 2.9 Diskrétny výstup, spotrebič

Analógový vstup je taký, ktorý prenáša do PA spojitý signál z technológie. Na Obr. 2.10 je ako príklad uvedený spojitý snímač hladiny kvapaliny v nádrži.



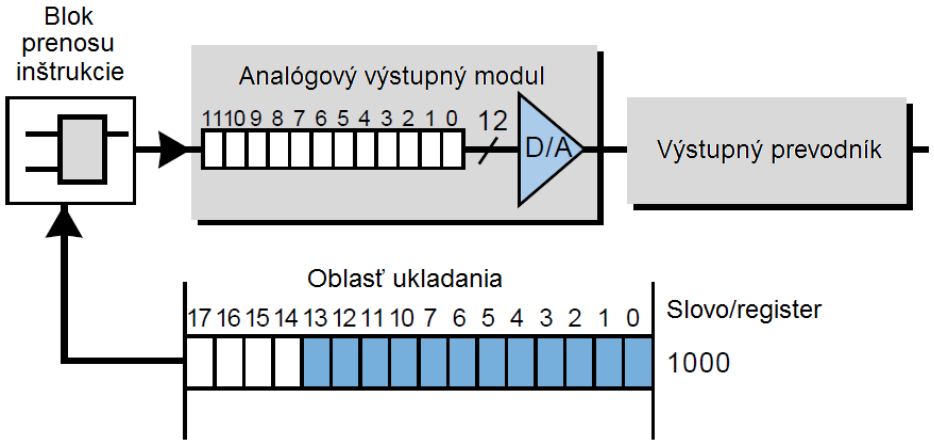
Obr. 2.10 Príklad analógového vstupného signálu do PA

Pretože mikropočítač pracuje vnútorne primárne s číslami v pevnej rádovej čiarke, musia byť v analógových moduloch spracované analógové signály cez príslušné prevodníky (Obr. 2.11).



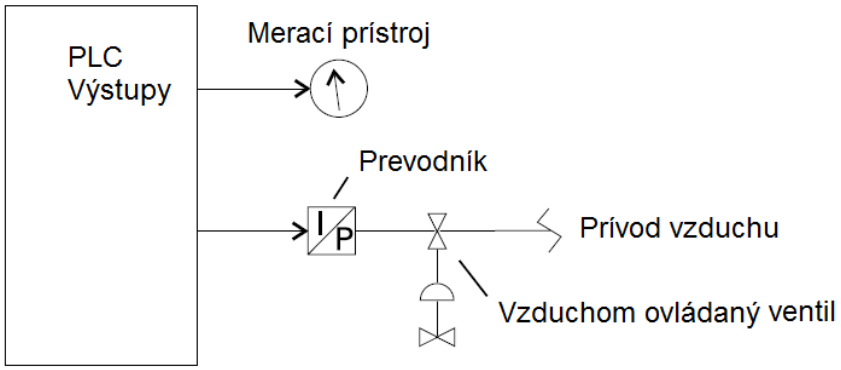
Obr. 2.11 Typická vnútorná štruktúra AO modulu

Analógový výstup je taký, ktorý prenáša z programovateľného automatu spojitý signál do technológie (Obr.2.12).



Obr. 2.12 Typická vnútorná štruktúra AO modulu

Tento signál môže nejaké technologický člen spracovať priamo (napr. zobrazovacie zariadenie) alebo cez vhodný výkonový prevodník (napr. elektrického prúdu na tlak vzduchu). Príklad pre takéto výstupy je na Obr. 2.13.



Obr. 2.13 Príklad analógového výstupného signálu z PA

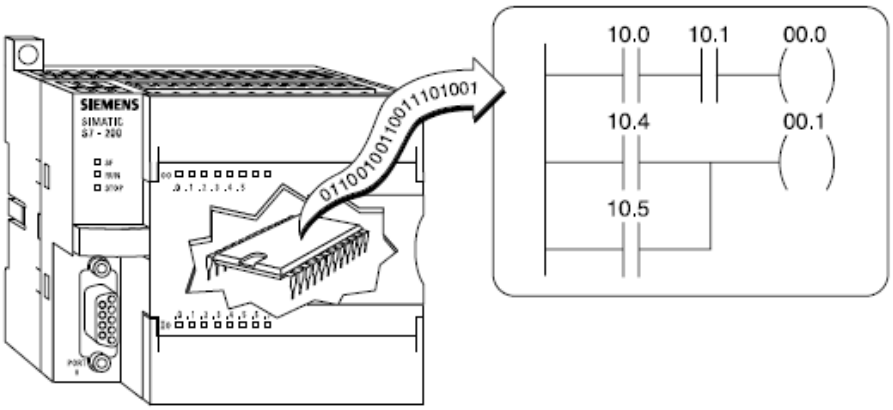
Najčastejšie typy IO modulov v programovateľných automatoch potom spracovávajú vyššie uvedené typy signálov. Z elektrického hľadiska sú dnes už signály z a do technológie štandardizované a preto sú štandardné aj typy IO modulov v PA:

- Binárne vstupy (24V DC, 230V AC)
- Binárne výstupy (realizované ako kontakt relé – *releové* alebo elektronický spínač - *tranzistorové*)
- Analógové vstupy a výstupy
  - prúdové 4 - 20mA
  - napäťové +/-10V

Pretože sa v priemyselnej praxi vyskytujú typy snímačov (resp. aktuátorov), ktoré sa hojne používajú, ale z elektrického hľadiska pracujú s neštandardnými typmi signálov, majú dnes prakticky všetci výrobcovia PA k dispozícii aj špeciálne IO moduly, ktoré vieme priamo pripojiť na takéto zariadenia (napr. termočlánky, impulzné snímače polohy a rýchlosti v pohonoch, šírkovo-impulzne modulované akčné členy a pod.). Najčastejšie používané typy špeciálnych IO modulov sú:

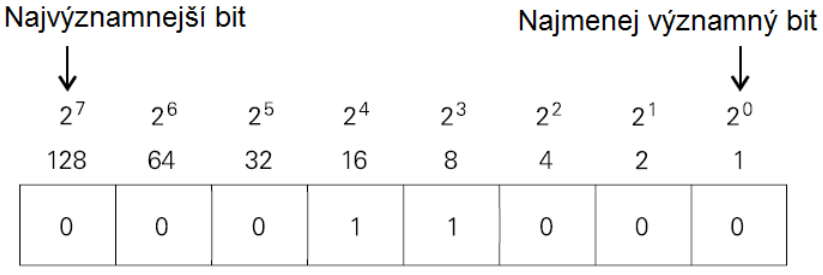
- Rýchle impulzné vstupy a výstupy (rýchle čítače, PWM modulované výstupy ...)
- Špeciálne moduly pre snímanie teplôt (pre termočlánky, odporové teplomery ...)

Centrálna procesorová jednotka CPU je mikroprocesorový systém, ktorý monitoruje vstupy PA a na základe naprogramovaných inštrukcií nastavuje jeho výstupy (Obr. 2.14).



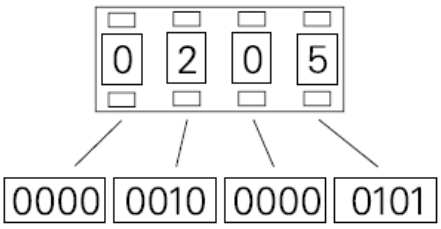
Obr. 2.14 Podstata činnosti CPU

Vnútorná reprezentácia všetkých typov signálov v CPU je binárne číslo vo forme núl a jednotiek, ktoré sú uložené v pamäti v tzv. bitoch. Tieto sú usporiadané vo forme bytov (8 bitov) alebo slov (niekoľko bytov, podľa typu procesora) (Obr. 2.15).



Obr. 2.15 Binárna reprezentácia informácie v CPU

Človek potrebuje reprezentovať číselné informácie v desiatkovej sústave, slovné vo forme textu, zobrazovať čísla pomocou kódovaných znakov v BCD sústave a pod (napr. v BCD sústave je každá číslica reprezentovaná 4-bitovým binárnym číslom). Preto CPU vykonáva okrem spracovania informácii vhodnú konverziu svojich vstupov a výstupov (Obr. 2.16)



Dekadické čísla	Binárne čísla
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Obr. 2.16 BCD konverzia informácie v CPU

Vnútorná reprezentácia analógových signálov v PA je číslo, zodpovedajúce danému signálu, pričom záporné hodnoty sú zobrazované v tzv. binárnom doplnku (Obr. 2.17):

Percento otvorenia	Výstupné napätie	Hodnota
0%	–10 VDC	–4095
10	–8	–3276
20	–6	–2457
30	–4	–1638
40	–2	–819
50	0	0
60	+2	+819
70	+4	+1638
80	+6	+2457
90	+8	+3276
100	+10	+4095

Obr. 2.17 Vnútorné zobrazenie analógových signálov v PA

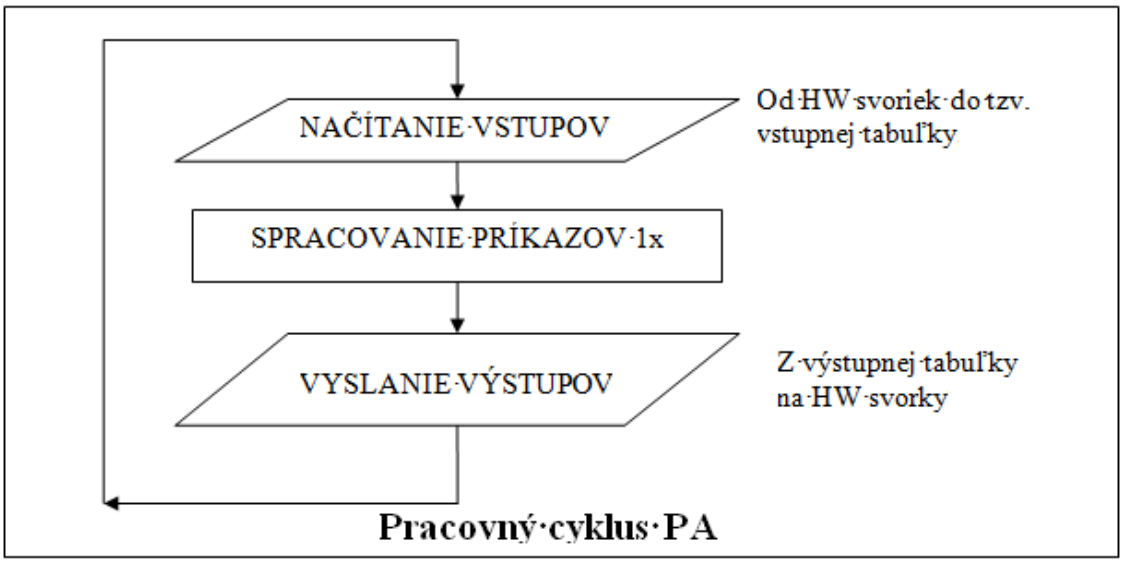
Činnosť, ktorú má PA vykonávať, je v pamäti CPU zapísaná vo forme príkazov, tzv. inštrukcií. Pretože táto činnosť sa dá veľmi jednoducho prepísať, je najväčšou výhodou PA jeho programová konfigurovateľnosť, ktorá je možná kvôli fyzickému oddeleniu prepojenia riadiaceho algoritmu od technologického procesu.

Vzhľadom na súčasné veľké rozšírenie PA v technologickej praxi boli postupy programovania PA štandardizované, a to európskou normou IEC 1131.

3 PROJEKT SYSTÉMU S PROGRAMOVATEĽNÝM AUTOMATOM

Programovateľný automat sa môže nachádzať v dvoch základných režimoch (módoch) činnosti:

- *Programovacom* (PROGRAM, STOP,...), v ktorom je možné zadávať zmeny v programe, prípadne zmeny v definícii HW (ináč povedané vytvárať projekt systému s PA)
- *Pracovnom* (RUN, MONITOR, ...), v ktorom PA vykonáva svoj tzv. **pracovný cyklus** PA. Ten sa vykonáva ***cyklicky*** buď na základe preddefinovaných systémových prerušení alebo v nekonečnej slučke.



Obr. 3.1 Pracovný cyklus PA

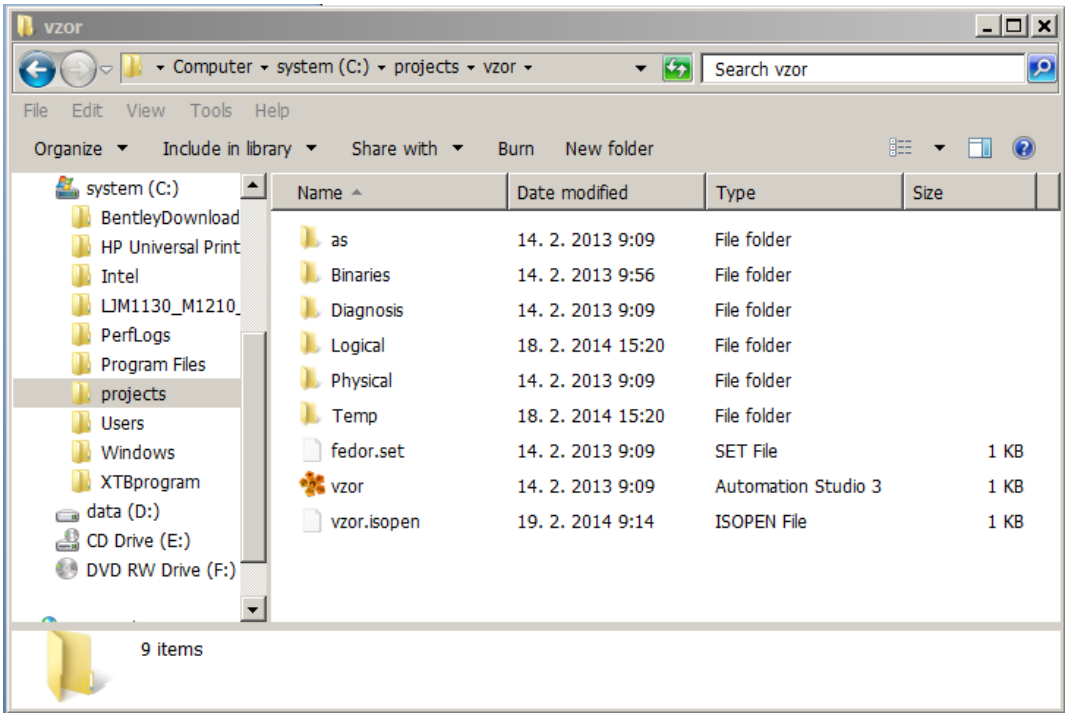
Nasadenie PA pre riadenie konkrétneho technologického procesu si vyžaduje konkrétne definovať nasledujúce informácie:

- HW konfiguráciu PA
- Sieťovú konfiguráciu PA (parametre komunikácie PA s okolím)
- SW konfiguráciu PA

Všetky tieto informácie je potrebné zadávať v rámci programovacieho prostredia konkrétneho automatu (napr. pre B&R je to *B&R Automation Studio*, pre Siemens *Simatic S7 Manager* a pod.)

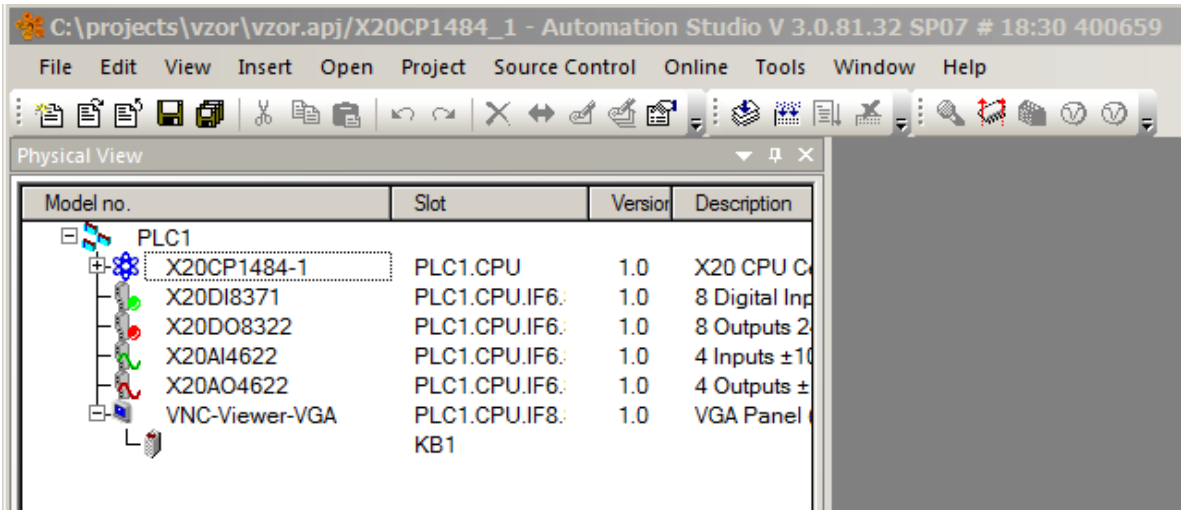
Množine týchto informácií pre riešenie danej úlohy hovoríme **PROJEKT**. Programovať PA teda znamená vytvoriť a popísať (konfigurovať) projekt, ktorý bude riešiť zadanú úlohu.

Projekt sa štandardne na inžinierskej stanici ukladá ako zložka, v ktorej sú uložené všetky potrebné súbory. Na obr.3.2 je uvedené štandardné uloženie projektu s názvom „**vzor**“ v prostredí PA od firmy B@R.

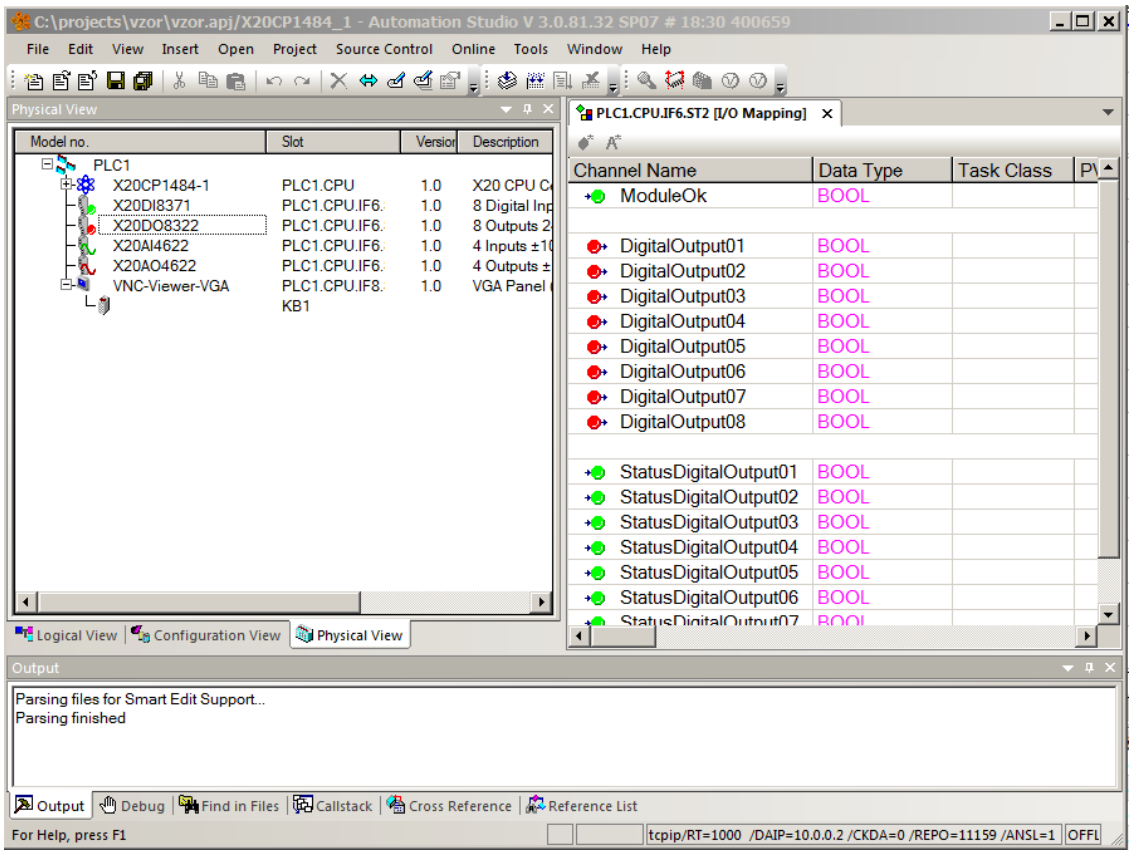


Obr. 3.2 Rozdelenie PA podľa veľkosti

Práca s týmito súbormi je v rámci programovacieho prostredia inžinierskej stanice umožnená cez menu, obvyklé v systéme Windows, ako to ukazuje obr. 3.3.



Obr. 3.3 Štandardné menu pre prácu s objektami projektu



Obr. 3.4 Základné okná pre zobrazenie projektu v PA

Aktuálny projekt je zložený z objektov a zobrazovaný v dvoch základných oknách – vľavo sú jednotlivé objekty projektu a vpravo obsah (vlastnosti a detaily) vybraného objektu. V spodnej časti prostredia môže byť zobrazené zvolené informačné okno.

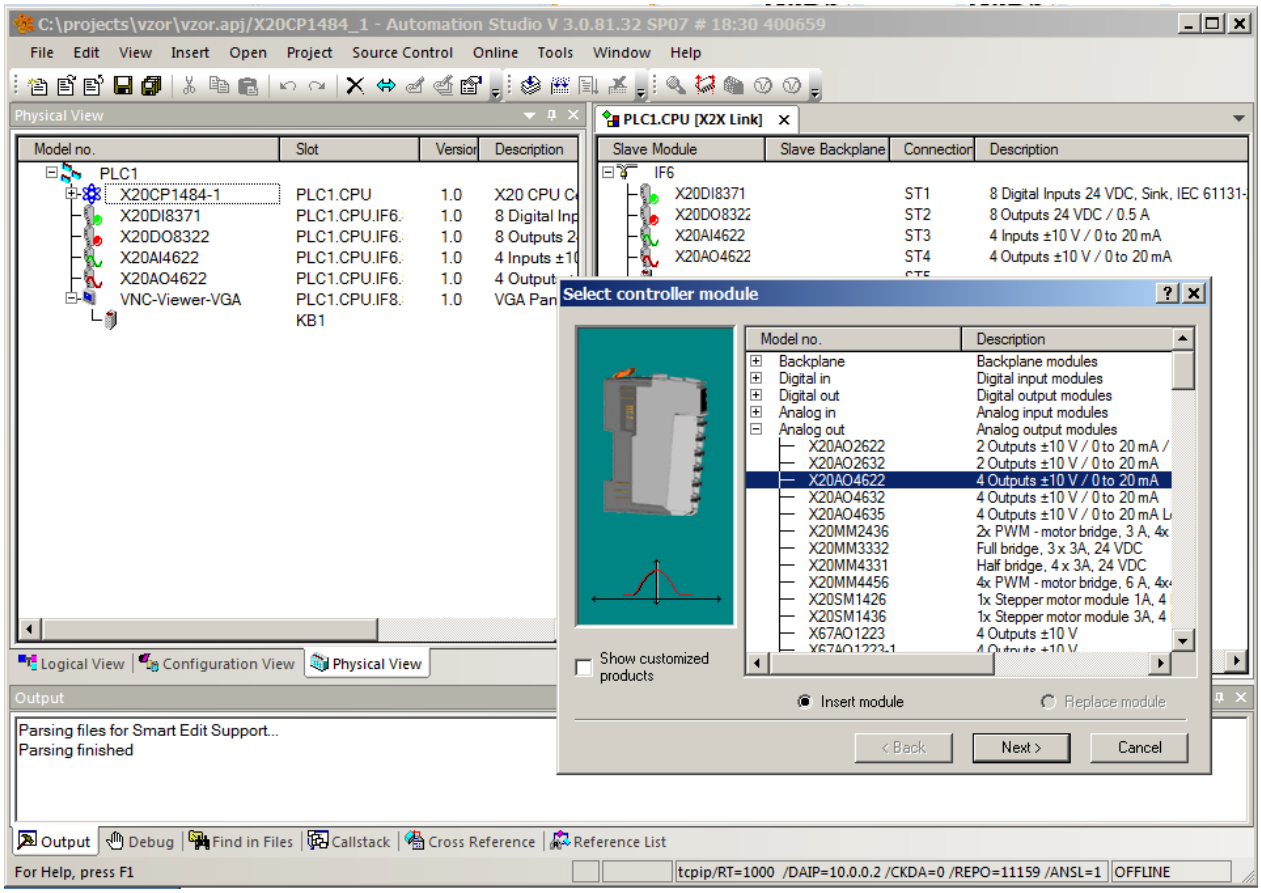
Na projekt sa v programovacom prostredí PA môžeme pozerať z troch základných hľadísk:

- **fyzického**, ktoré zobrazuje a umožňuje konfigurovať jednotlivé fyzické (hardvérové) komponenty systému s PA
- **logického**, ktoré zobrazuje logickú stavbu projektu , nezávislú od hardvéru. V podstate sa jedná o programové moduly, knižnice, definície premenných a ich typov a pod.
- **konfiguračného**, v ktorom sa definujú vlastnosti a parametre jednotlivých HW komponentov projektu.

3.1 Konfigurácia fyzických HW komponentov projektu

HW konfigurácia sa robí v okne fyzikálneho pohľadu vkladaním vybraných HW komponentov (napr. ďalších automatov, operátorských displejov,IO modulov a pod.) do aktuálneho projektu na vybrané pozície (sloty) v danom rošte (chassis, rack). Nastavuje sa obvykle na začiatku projektu a úzko súvisí s konkréntnym hardvérom riadenej technológie. Ak je HW už zapojený a PA

komunikuje s inžinierskou stanicou na PC, je možné túto reálne zapojenú konfiguráciu obvykle aj priamo načítať, čo značne uľahčuje prípravu projektu a znižuje možné chyby.

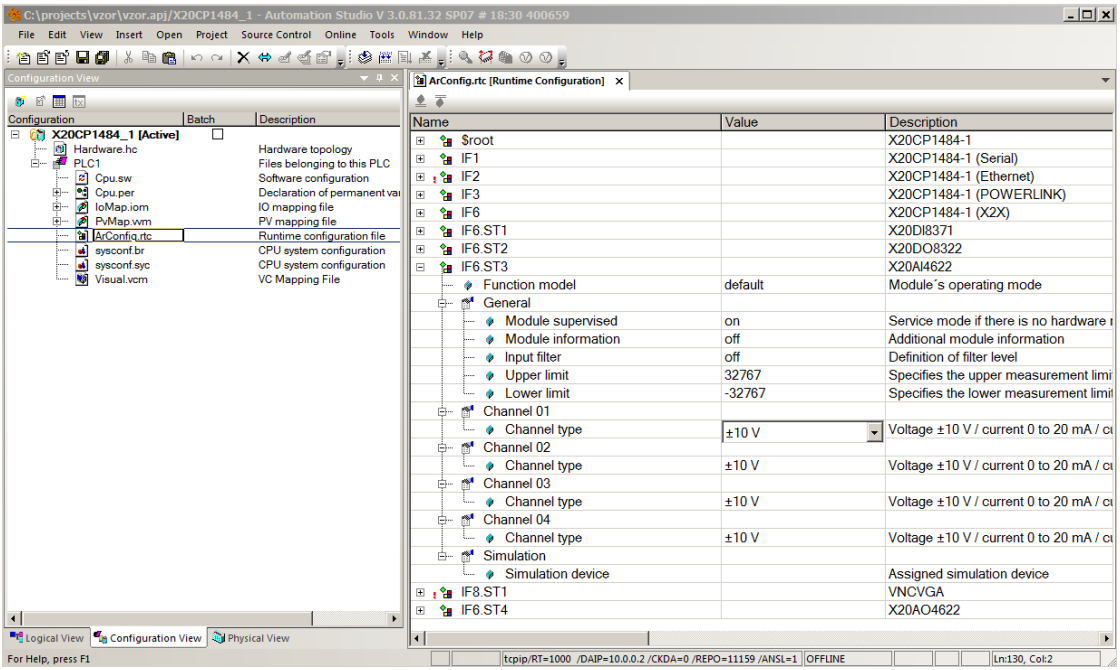


Obr. 3.5 Príklad vloženia AO modulu do HW konfigurácie PA

Na obr. 3.5 je do automatu s názvom PLC1, ktorý komunikuje s IO modulmi rady X20 cez svoj interfejs IF6 príklad, ako vložiť cez príslušné okno na zvolenú pozíciu analógový výstupný modul zvoleného typu.

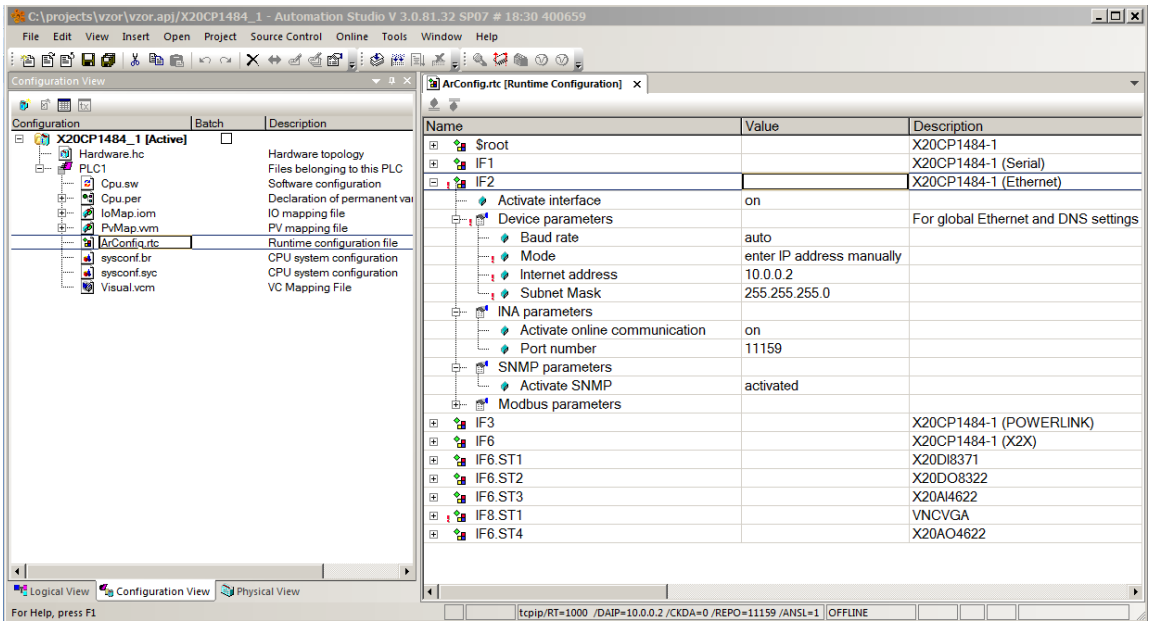
3.1.1 Konfigurácia parametrov HW komponentov PA

Praktický každý fyzický HW objekt programovateľného automatu (ako aj samotný PA) je dnes už „ inteligentný“ a jeho vlastnosti je možné nastaviť vhodnými parametrami. Tieto sa nastavujú v okne konfiguračného pohľadu na projekt. Napríklad PA od firmy B@R má k dispozícii niekoľko tzv. interfejsov IF, cez ktoré môže CPU komunikovať so svojím okolím.



Obr. 3.6 Príklad konfigurácie komunikácie CPU s analógovým kanálom č.1 na karte 3

Na obr. 3.6 je uvedený príklad, ako je možné na zvolenej analógovej vstupnej karte zmeniť napäťový typ vstupného signálu na kanáli č.1 za prúdový. Podobne je z obrázku zrejmé, že by bolo možné nastaviť vstupný filter proti zašumeniu analógového signálu, nastaviť limity pre výstup A/Č prevodu a podobne. Poznamenajme, že s touto AI kartou komunikuje CPU cez interfejs IF6.ST3. Je zrejmé, že keby sme chceli nastaviť napríklad komunikáciu CPU s nejakým zariadením cez ethernet, museli by sme nastaviť parametre interfejsu IF2, napr. podľa obr.3.7.



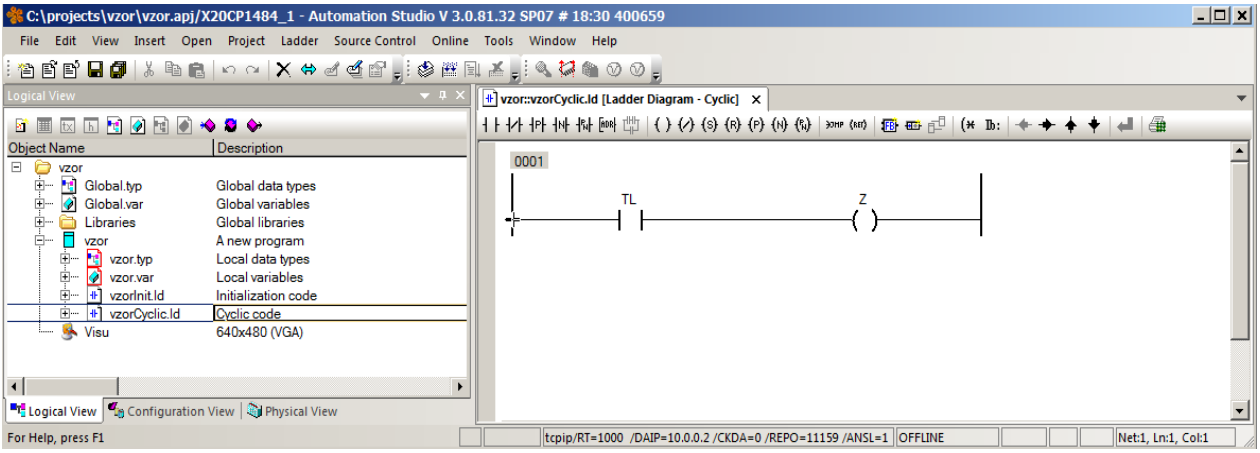
Obr. 3.7 Príklad konfigurácie komunikácie CPU cez sieť typu ethernet

3.2 Konfigurácia softvéru PA

Pod softvérom PA budeme rozumieť súbor programových objektov, ktoré projekt logicky obsahuje a ktoré vykonávajú predpísanú činnosť PA. Tieto objekty je možné manažovať v okne logického pohľadu na projekt, ktoré je „nezávislé“ od HW konfigurácie projektu. Medzi základné SW objekty patria:

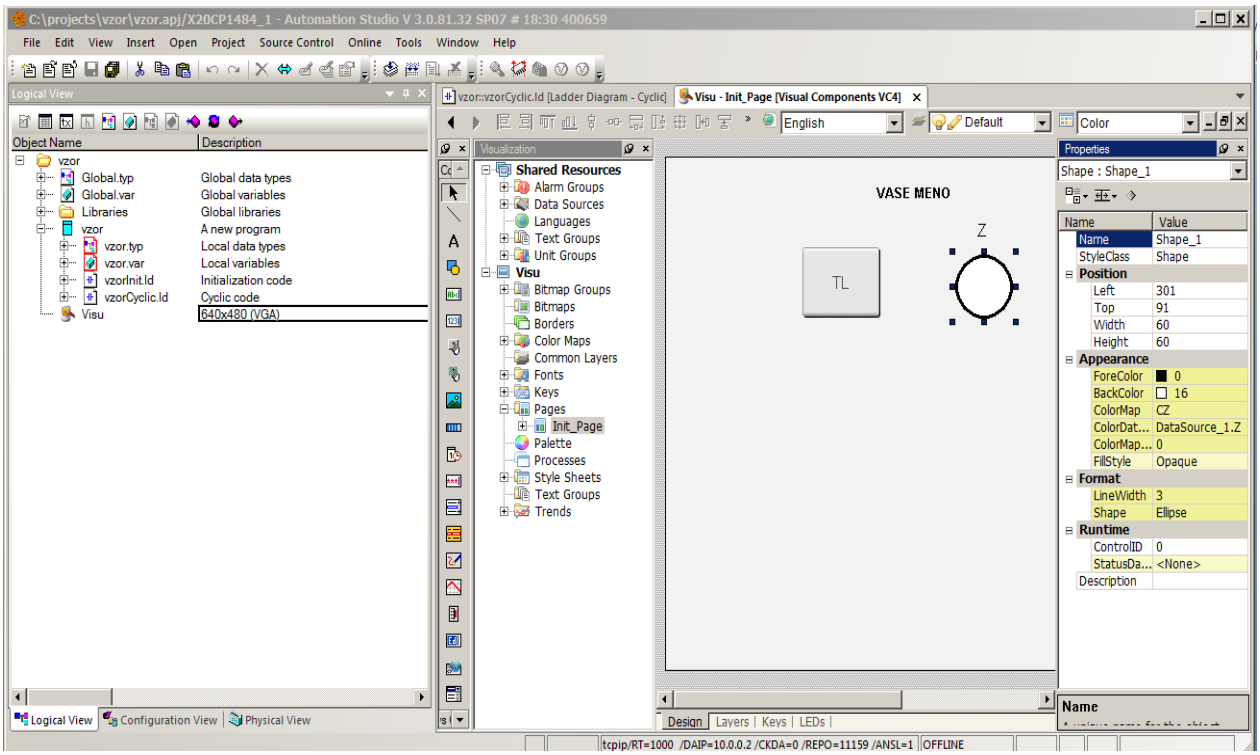
- Tabuľky datových typov
- Tabuľky premenných
- Knižnice
- Programové moduly
- Vizualizačné objekty
- Datové objekty
- Funkcie

Na obr.3.8 je uvedený príklad jednoduchého projektu a objektov, ktoré obsahuje. Každý typ objektu má kvôli prehľadnosti svoju ikonu. V pravom okne sa vieme pozrieť dovnútra objektu a editovať jeho funkcie.



Obr. 3.8 Objekty projektu VZOR z logického hľadiska

Na obr.3.9 je uvedený príklad, ako môže vyzerať editovacie prostredie pre vizualizačný objekt, ktorý je schopný vizualizovať na úrovni štandardnej VGA grafiky. Poznamenajme, že nie je rozhodujúce, či je táto vizualizácia realizovaná na špeciálnom operátorskom displeji alebo napr. na štandardnom PC s príslušným programovým vybavením.



Obr. 3.9 Editovacie prostredie vizualizačného programového objektu typu VGA

4 ZÁKLADY PROGRAMOVANIA PA

Celý program v PA (nazývame ho často „*aplikácia*“) sa skladá podľa normy IEC 61131-3 z týchto **základných programových častí**:

- Premenné
- Typy
- Konštanty
- Funkčné bloky
- Funkcie

Premenná je pamäťové miesto pre nejaký údaj aplikácie, pričom toto je deklarované (vyhradené) podľa typu premennej. Podľa toho, aké programové jednotky majú k danej premennej prístup, rozlišujeme ***globálne*** a ***lokálne*** premenné.

Typ (datový) určuje vlastnosti premennej, ako sú potrebný rozsah bytov v pamäti, presnosť zobrazenia, povolené operácie s touto premennou a pod.

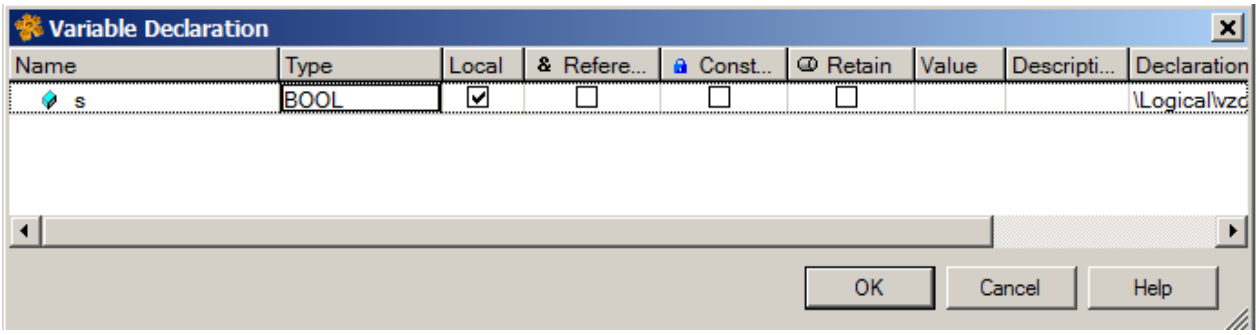
Konštanta je premenná, ktorej obsah sa počas behu programu nemení.

Funkčný blok je organizačná programová jednotka, ktorá po svojom vykonaní vracia jednu alebo viac hodnôt. Všeobecne má viac vstupov a viac výstupov.

Funkcia je organizačná programová jednotka, ktorá po svojom vykonaní vracia len jeden datový element, hoci môže mať viac vstupov.

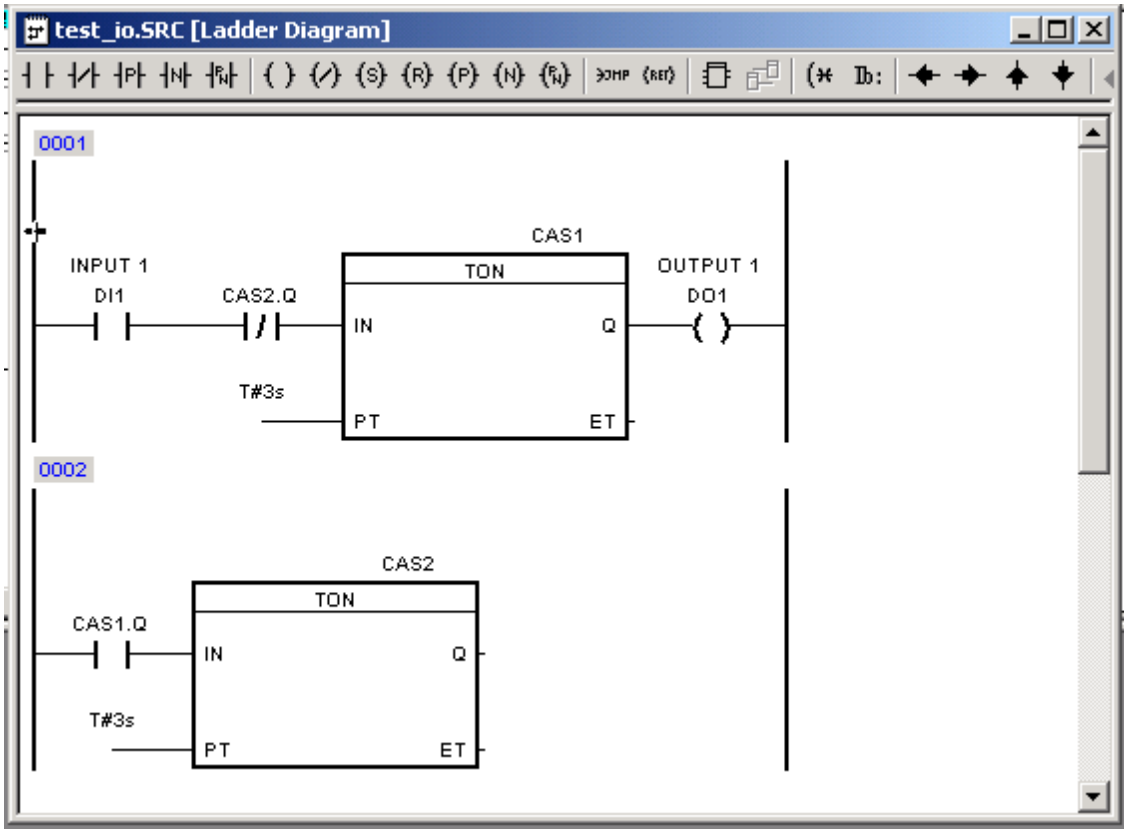
Programovanie nejakej aplikácie programovateľného automatu zahŕňa tieto tri základné činnosti:

- Definovanie databázy používaných údajov (vstupov, výstupov, premenných, konštánt, polí ...). Tým rozumieme definovanie ich jednoznačných mien, typov, počiatočných hodnôt a ďalších vlastností. Často sa premenné deklarujú v prehľadnej tabuľkovej forme, ako to ukazuje obr.4.1.



Obr. 4.1 Príklad deklaračnej tabuľky

- Definovanie organizačných programových jednotiek, riadiacich funkcií (napr. vo forme líniovej schémy, programovacieho jazyka, funkčnej schémy a pod.). Príklad programového modulu vo forme líniovej schémy je na obr.4.2.



Obr. 4.2 Príklad programovej organizačnej jednotky v tvare líniovej schémy

- Testovanie a ladenie funkčnosti programovej aplikácie

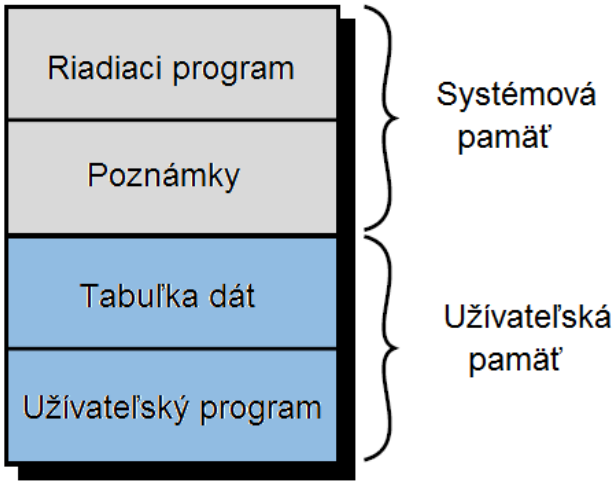
– *Formálnej stránky aplikácie*, vnútornej logiky programu – stavy a súvislosti medzi jednotlivými premennými,

– *Logickej správnosti* fungovania procesov medzi PA a technologiou – individuálne testovanie jednotlivých IO v tzv. režime **force** vstupov a výstupov a funkčné testovanie (monitoring) programu a premenných počas testovacej prevádzky aplikácie

Údajové typy, funkčné bloky, funkcie a prípadne aj konštanty si môže užívateľ vytvárať nanovo v rámci svojej aplikácie, alebo ich môže zaradiť do aplikácie z externej knižnice. Niektoré (systémové) knižnice sú automaticky pripojené prostredím k novovytváranej aplikácii, iné si užívateľ prilinkuje podľa potreby.

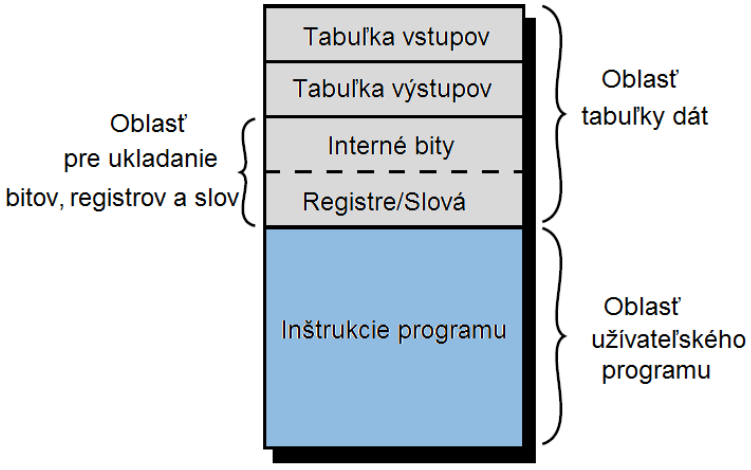
4.1 Organizácia pamäte a adresácia premenných v PA

Pamäť PA má dve základné časti – systémovú a užívateľskú, ako to ukazuje obr. 4.3.



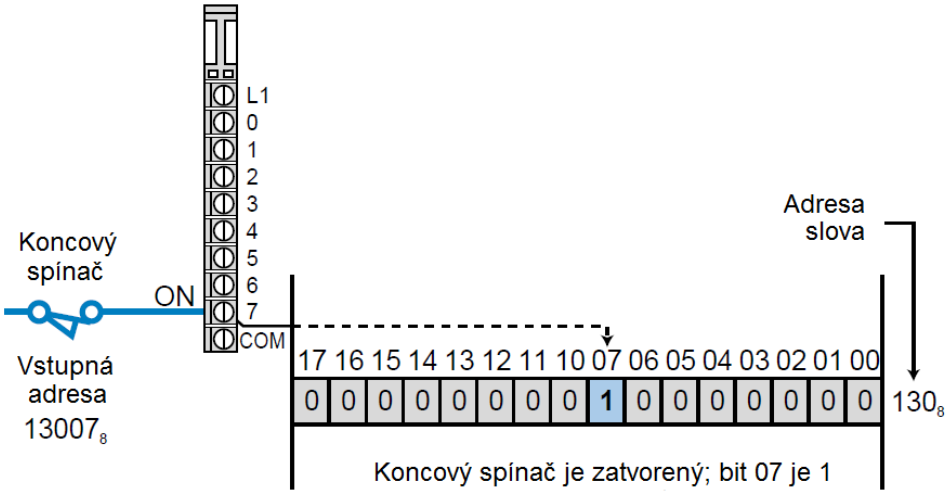
Obr. 4.3 Rozdelenie pamäte PA

Systémovú pamäť využíva PA pre svoje vnútorné funkcie a užívateľ z nej môže iba čítať niektoré vybrané informácie. V aplikácii programujeme aplikačnú časť pamäte, ktorej vnútorná štruktúra je na obr. 4.4.



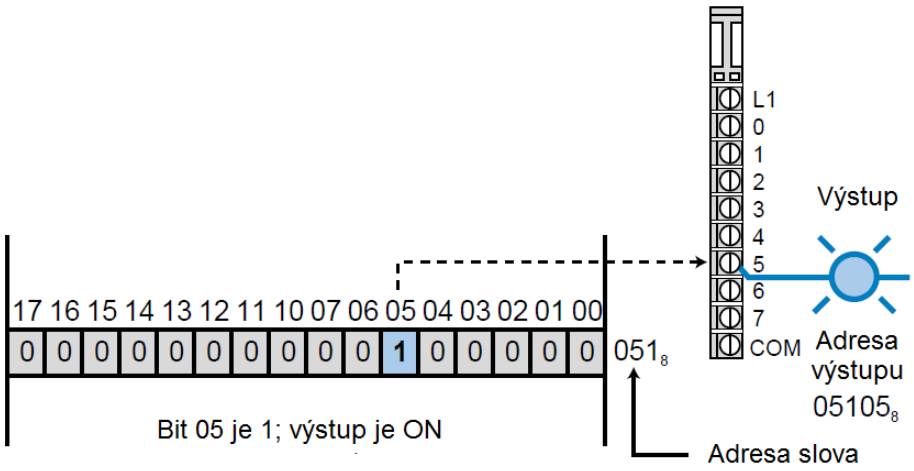
Obr. 4.4 Vnútorná štruktúra aplikačnej časti pamäte PA

Vstupná údajová tabuľka (Input table) zobrazuje a počas pracovného cyklu PA tiež uchováva stav signálov na jednotlivých svorkách vstupných modulov PA a ukladá ich vo forme bitov (resp. slov) na zodpovedajúce miesto v tejto tabuľke. Načítanie stavu svoriek sa deje cyklicky v prvej časti pracovného cyklu PA. Na obr. 4.5 je príklad zobrazenie binárneho signálu z koncového spínača, ktorý je pripojený na svorku 7 karty, konfigurovanej do adresy 130.



Obr. 4.5 Príklad zobrazenia binárneho signálu do vstupnej tabuľky

Výstupná údajová tabuľka (Output table) nastavuje a uchováva stav bitov (slov) na jednotlivých svorkách výstupných modulov O interface PA. Nastavovanie sa deje cyklicky sa to v tretej časti pracovného cyklu PA. Na obr. 4.6 je príklad nastavovania výstupu na svorke 5, ktorý je pripojený na výstupnú kartu konfigurovanú na adresu 51.



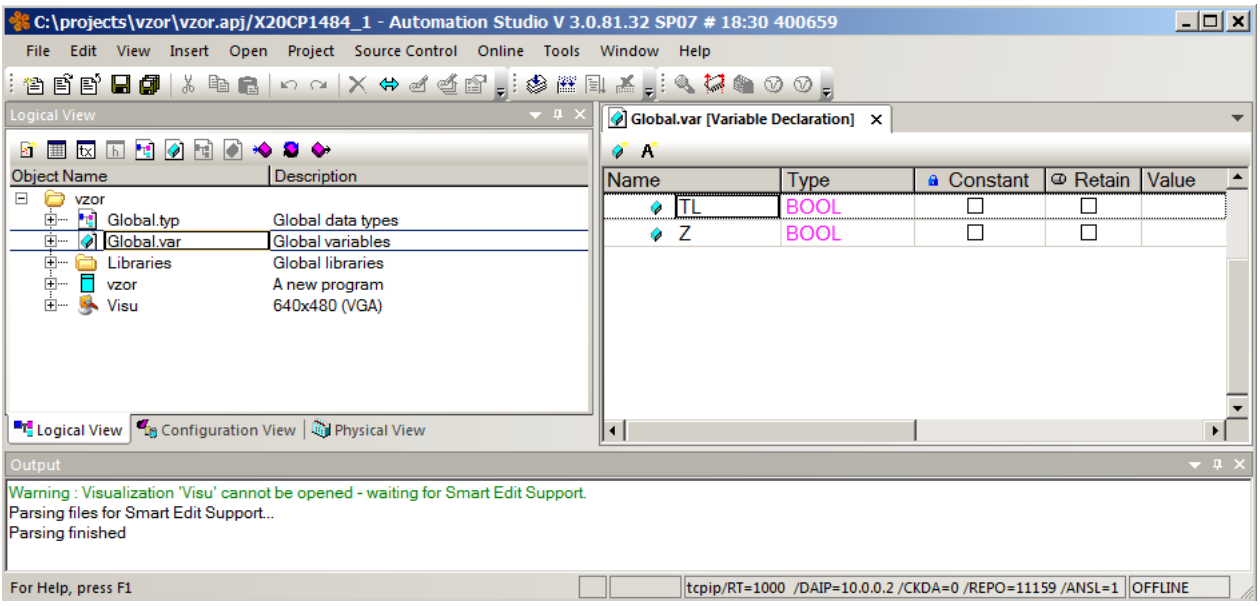
Obr. 4.6 Príklad nastavovania binárneho signálu do výstupnej tabuľky

Poznámka: Konfigurácia a priradenie jednotlivých IO modulov programovateľného automatu ku jednotlivým adresám vo vstupnej (resp. výstupnej) tabuľke úzko súvisí s fyzickou dispozíciou (polohou) jednotlivých modulov. Kvôli zrýchleniu komunikácie medzi CPU a jednotlivými modulmi majú tieto tabuľky pevnú veľkosť, ktorá sa dá meniť len mimo behu aplikácie.

Pamäť bitov a registrov je určená pre vnútorné programové premenné a operandy jednotlivých inštrukcií programu. Tento sa fyzicky ukladá do **užívateľskej oblasti** pamäte (User Program Area)

4.2 Programovanie komponentov aplikácie v PA

Užívateľský program PA (aplikácia) je modulárny (štrukturovaný). S jednotlivými komponentami aplikácie vieme pracovať v logickom náhľade. Príklad prostredia, ktoré nám to umožňuje, je na obr.4.7.



Obr. 4.7 Príklad nastavovania binárneho signálu do výstupnej tabuľky

Jednotlivé komponenty pridávame v ľavom okne prostredia, zatiaľ čo ich vlastnosti konfigurujeme v jeho pravom okne. V uvedenom príklade je v pravom okne zobrazená deklaračná tabuľka globálnych premenných, ktorý je druhým komponentom aplikácie „*vzor*“.

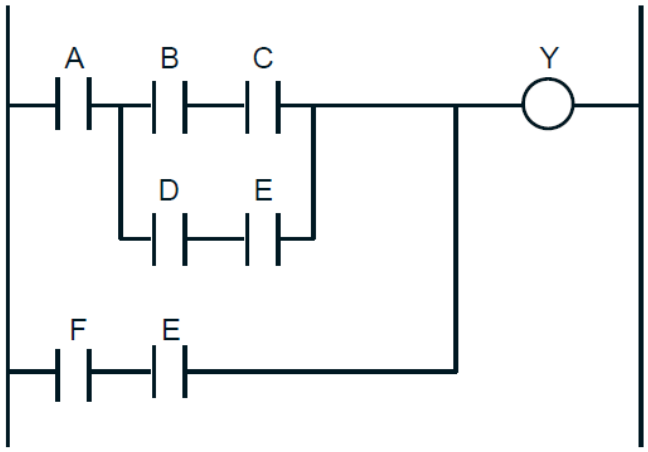
Jednotlivé programové moduly aplikácie je možné programovať rôznymi formami.
Najčastejšie formy zápisu činnosti (algoritmov) PA v užívateľských programových moduloch sú v súlade s európskou normou IEC 1131-3

A. Grafické

V tejto forme sú jednotlivé príkazy (inštrukcie) programu zobrazované svojou unikátnou značkou a vzťahy medzi nimi (požadované funkcie programu) sú znázorňované vhodnými prepojeniami.
Príklad: Chceme realizovať booleovskú funkciu v tvare.

$$Y = A \bullet (B \bullet C + D \bullet E) + (F \bullet E)$$

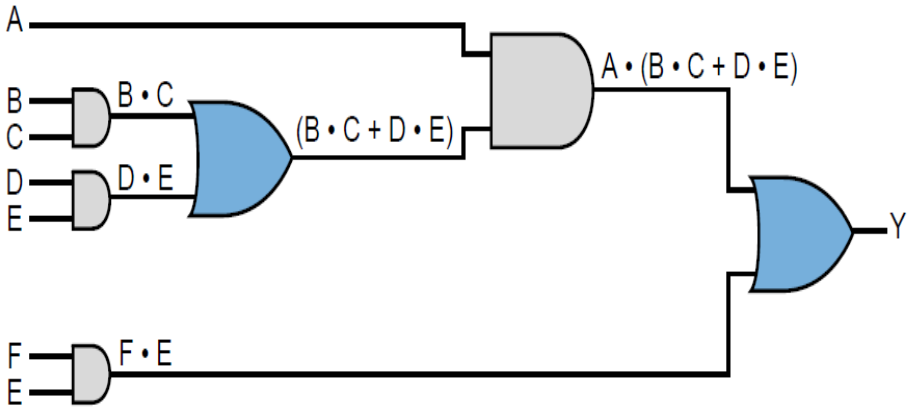
Základná forma, vychádzajúca z klasickej releovo-stykačovej logiky, je tzv. **líniová schéma** (Ladder Diagram LD). Obr.4.8 ukazuje príklad zobrazenie príslušnej líniovej schémy.



Obr. 4.8 Príklad líniovej schémy

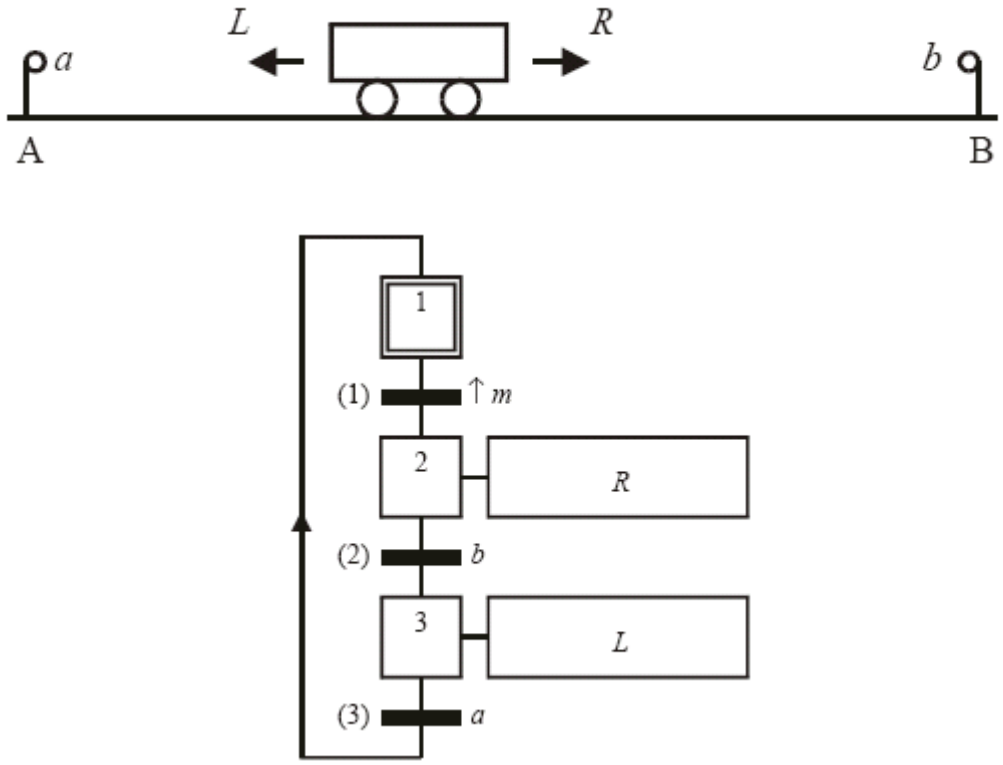
Na tomto obrázku je značkou označená inštrukcia pre test stavu každej premennej a sériovým (paralelným) prepájaním jednotlivých značiek sa realizujú logické funkcie AND (OR).

Druhou najčastejšie používanou možnosťou je **schéma z funkčných blokov** (Function Block Diagram FBD), ako to ukazuje obr.4.9.



Obr. 4.9 Príklad schémy z funkčných blokov

Mnoho výrobných procesov a liniek pracuje podľa určitých časových postupov v rámci definovaných krokov. Tieto sekvenčné procesy sa exaktne popisujú sekvenčných funkčných diagramov (Sequential Function Chart SFC). Obr.4.2.5 ukazuje príklad riadenia pohybu vozíka pomocou príslušnej sekvencie.



Obr. 4.10 Príklad sekvenčného funkčného diagramu pre riadenie vozíka

B. Algebraické

V tejto forme zápisu činnosti programu PA sú jednotlivé príkazy (inštrukcie) programu zapisované vo forme zvoleného formalizovaného jazyka. Norma IEC 61131-3 uvádza tieto najčastejšie formy:

- **Špecializovaný assembler** (Instruction List IL) – pozostáva zo série po sebe idúcich príkazov. Každý z nich je v novom riadku a obsahuje operátor, závislý od typu inštrukcie a príslušné operandy.

```
LD    Var1
AND(Var2
OR    Var3
)
ST    Var4
```

- **Štruktúrovaný text** (Structured Text ST) – príkazový jazyk vyššej úrovne, podobný C++,BASIC, Pascal

```
PROGRAM _CYCLIC
  (* plain text information of current step *)
  IF (pBrewingText <> 0) THEN
      strcpy(ADR(gBrewing.status.brewingStepText), pBrewingText);
  END_IF

  IF EDGEPOS(diStartCoffee) = TRUE THEN
    gMainLogic.status.progressStep := 0;      (* transporting cup to brewing station *)
  END_IF
```

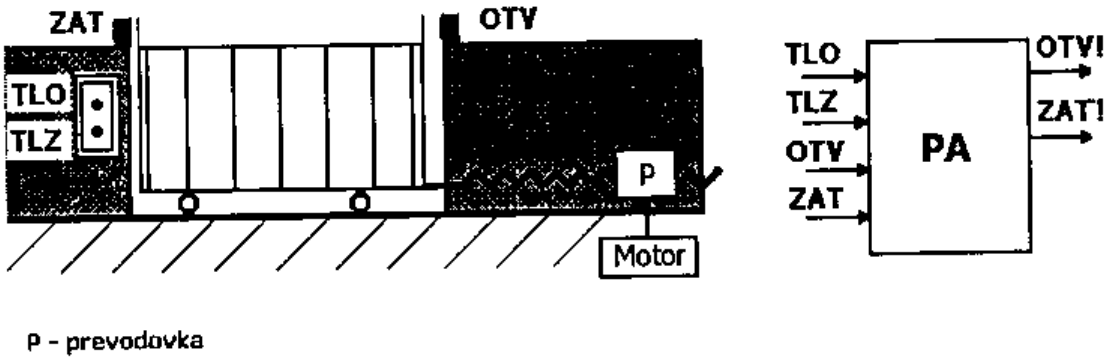
- **Ansi C++** – najčastejšie používaný jazyk pre problémovo orientované programové moduly, ktorý umožňuje využívať existujúce moduly pre riešenie už známych úloh.

```
#include <bur/plctypes.h>

#ifdef _DEFAULT_INCLUDES
 #include <AsDefault.h>
#endif

/* prototyping */
BOOL inWindow(REAL set, REAL act, REAL window);

void _INIT HeatingINIT( void )
{
  /* PID parameters for water flow heater */
 tempLCRPIDpara.WX_max  = 100.0;
 tempLCRPIDpara.WX_min  = 0.0;
 tempLCRPIDpara.invert  = 0;
```

Nie každý programovateľný automat ponúka všetky tieto možnosti. Každý PA ponúka minimálne programovanie pomocou líniových schém. Preto budeme ďalej robiť popis algoritmov týmto spôsobom cez LD.

4.3 Príklad – formy zápisu činnosti v PA

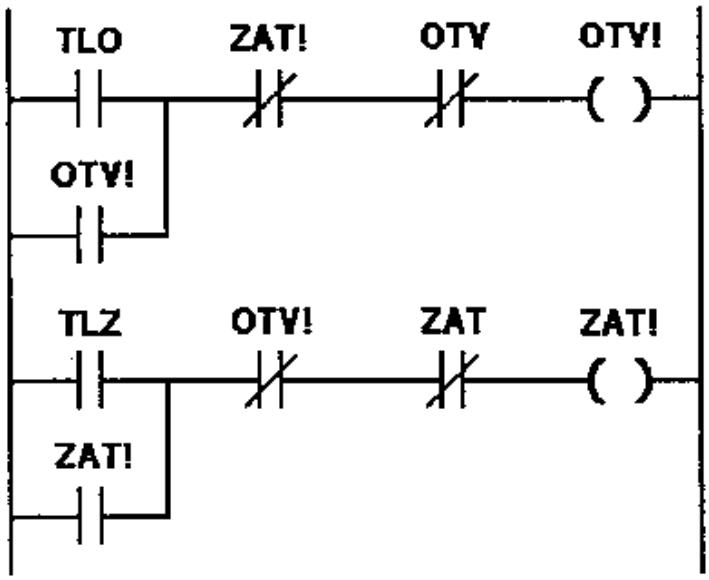
Programovateľný automat má za úlohu otvárať a zatvárať bránu, ktorú cez prevodový mechanizmus posúva asynchrónny motor. Podľa poradia fáz pripojených na motor sa brána otvára alebo zatvára. V koncových polohách brány sú umiestnené koncové spínače a pri bráne je umiestnená skrinka s dvoma tlačidlami pre jej otvorenie, resp. zatvorenie. Ovládenie brány si vyžaduje od PA spracovať 4 binárne vstupy a 2 binárne výstupy, ako tom ukazuje obr.4.11.



Obr. 4.11 Schéma brány a IO signálov programovateľného automatu

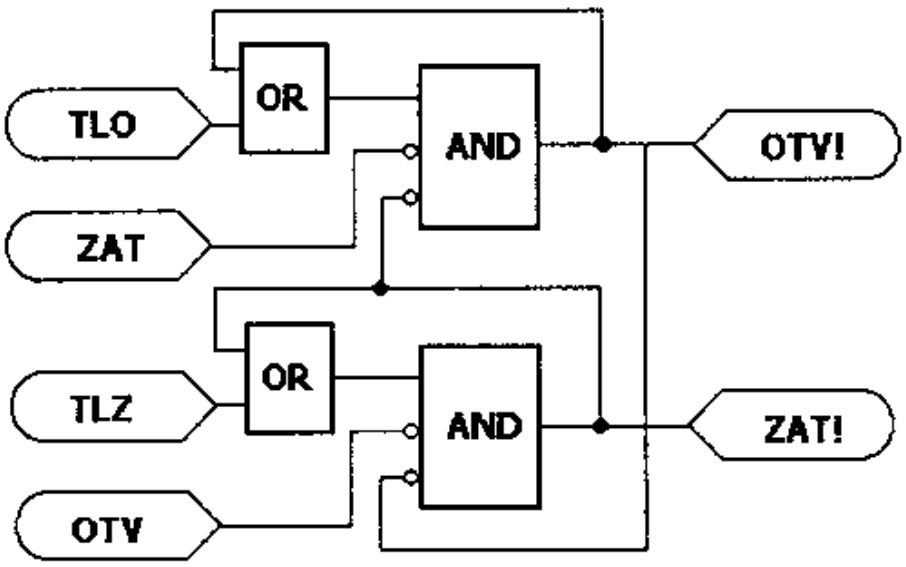
Slovný popis tejto úlohy môže byť:
- Ak je stlačené tlačidlo TLO, otváraj bránu, až kým nepríde signál od konc. spínača OTV
- Ak je stlačené tlačidlo TLZ, zatváraj bránu, až kým nepríde signál od konc. spínača ZAT

Algoritmus činnosti PA vo forme líniovej schémy by sme mohli znázorniť podľa obr. 4.12.



Obr. 4.12 Líniová schéma ako forma zápisu činnosti PA

Podobne by sme graficky vedeli túto úlohu pre PA zapísať vo forme funkčných blokov, ako to ukazuje obr.4.13.



Obr. 4.13 Schéma z funkčných blokov ako forma zápisu činnosti PA

Zápis inštrukcií vo forme najjednoduchšieho programovacieho jazyka (špecializovaného assembleru) by mohol mať nasledujúci tvar:

ld TLO
or OTV!
and not ZAT
and not ZAT!
out OTV!
...

Poznámka: Význam symbolických skratiek ld – načítaj vstup, out – vyšli výstup, or/and/not – vykonaj príslušnú logickú operáciu.

4.4 Všeobecný postup pri riešení úlohy pomocou PA:

Pri návrhu programovej aplikácie pre PA sa doporučuje používať nasledujúci postup, ktorý umožňuje systematicky riešiť vybranú úlohu:

1. Slovná formulácia úlohy – pomáha výrazne pri jej prepise do formálneho tvaru
2. Formalizovaný zápis úlohy (Definícia I/O v súlade s technológiou, resp. projektom, definícia formálnych mien a typov použitých premenných, zakreslenie algoritmu vo zvolenej forme) – toto je hlavná inžinierska tvorivá časť riešenia problému
3. Naprogramovanie formalizovaného zápisu úlohy v PC pod prostredím pre daný PA – tento krok vyžaduje hlavne poznanie konkrétneho PA a jeho programového prostredia

4. Testovanie funkčnosti programu po stránke:
   - formálnej
   - logickej

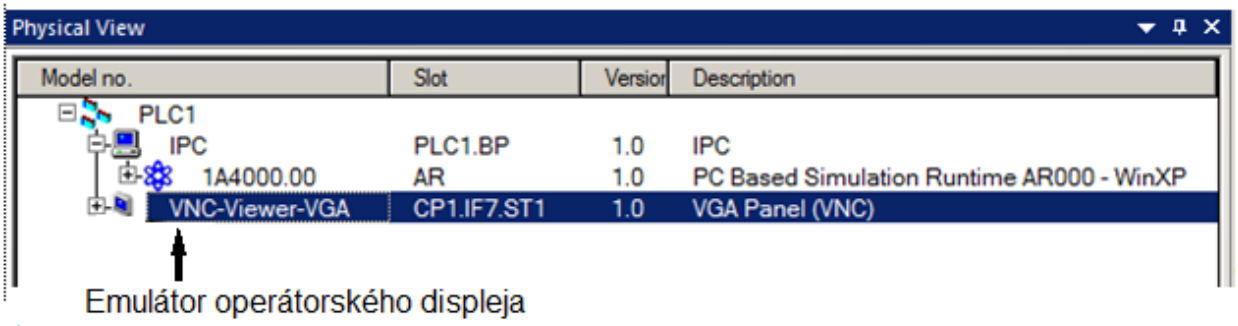
4.5 Príklad zostavenia jednoduchého projektu v PA

Všeobecný postup pri riešení úlohy pomocou PA má nasledujúce kroky:

1. *Slovná formulácia úlohy*

Pre otestovanie zvoleného binárneho výstupu PA urobte program, ktorý bude tento výstup cyklicky spínať a vypínať s periódou 3s. Tento výstup zobrazte ako blikajúci krúžok na operátorský panel.

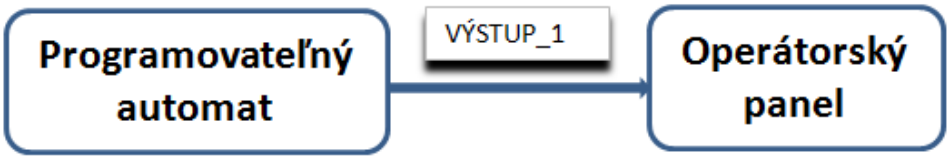
Fyzická zostava projektu systému s PA pre túto úlohu by vyzerala podľa obr.4.14.



Obr. 4.14 Fyzický pohľad na projekt príkladu 1

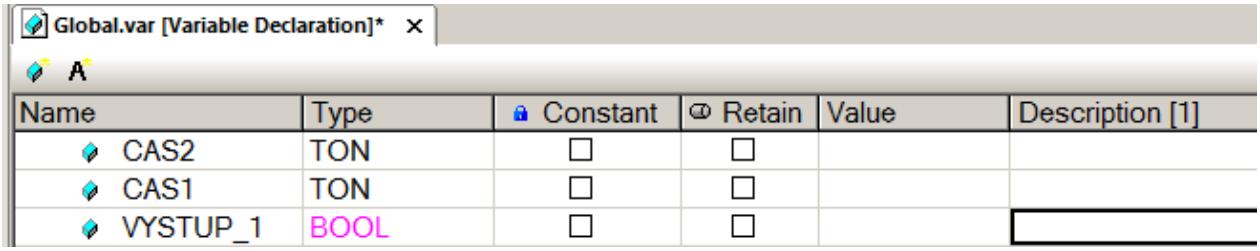
2. *Formalizovaný zápis úlohy (Definícia I/O v súlade s technológiou, resp. projektom, definícia formálnych mien a typov použitých premenných, zakreslenie algoritmu vo zvolenej forme)*

Je vhodné si na začiatku nakresliť blokovú predstavu úlohy, napr. podľa obr.4.15.



Obr. 4.15 Jednoduchá systémová predstava úlohy

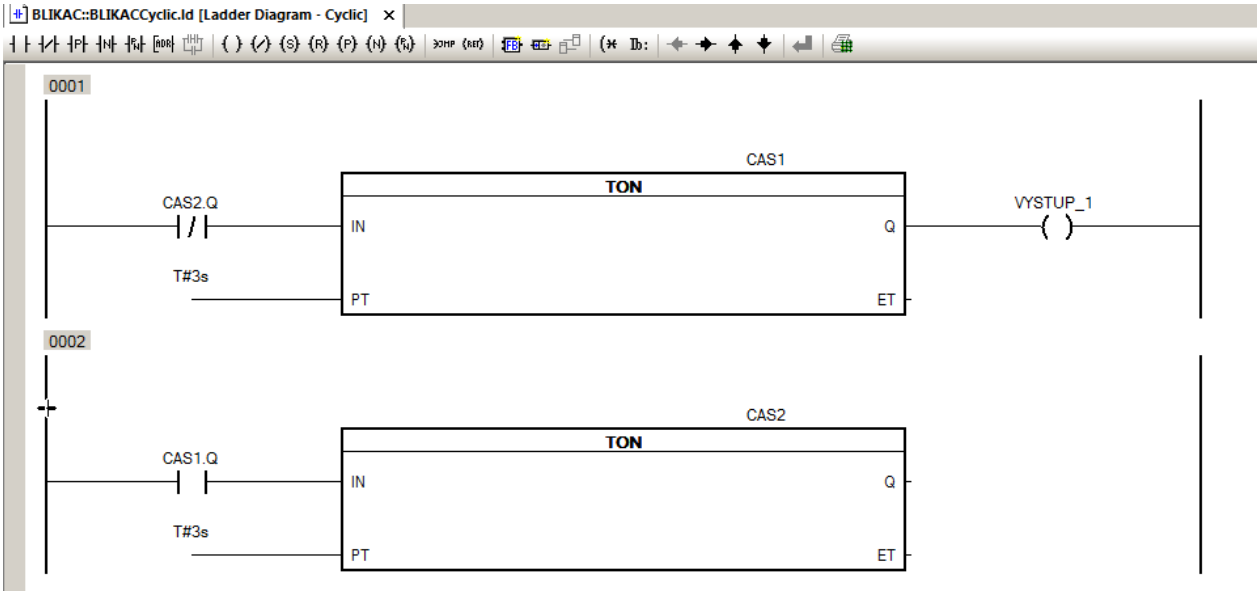
Pre tento prípad potrebujeme definovať jednu premennú typu *boolean*, ktorú sme nazvali **VYSTUP_1.** Pre meranie časov blikania ďalej potrebujeme dve premenné typu *časovač*. Definícia tabuľky globálnych premenných by mohla vyzerať nasledovne (obr.4.16):



Name	Type	Constant	Retain	Value	Description [1]
CAS2	TON	☐	☐		
CAS1	TON	☐	☐		
VYSTUP_1	BOOL	☐	☐		

Obr. 4.16 Deklaračná tabuľka premenných pre príklad 1

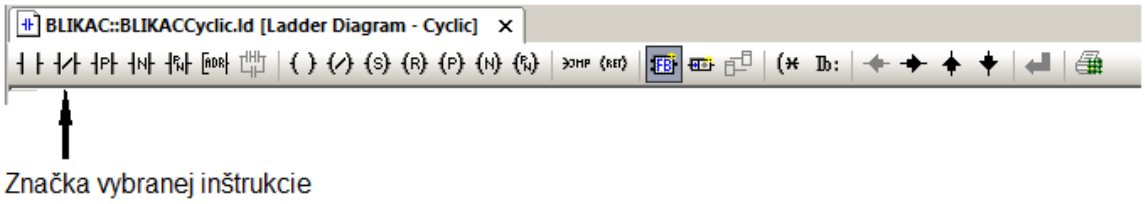
Algoritmus riadenia je jednoduchý a vieme ho nakresliť napr. v grafickej forme líniovej schémy podľa obr. 4.17. Skladá sa z dvoch línií a piatich inštrukcií.



Obr. 4.17 Líniová schéma blikača

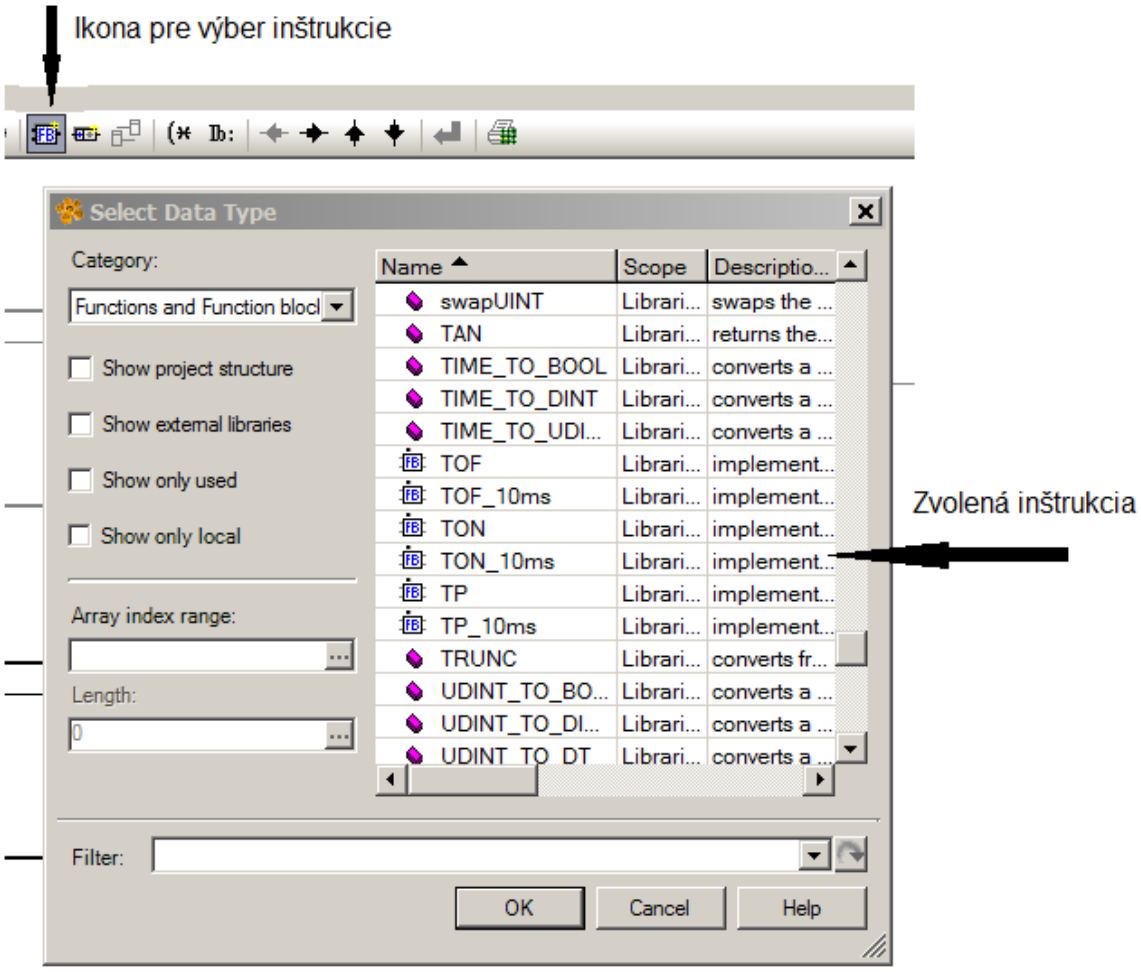
3. *Naprogramovanie formalizovaného zápisu úlohy v PC pod prostredím pre daný PA a jeho prenos do automatu*

Túto deklaráciu premenných a líniovú schému nakreslíme napr. v prostredí *B&R Automatin Studio*. K tomu slúži panel nástrojov, s ktorým sa pracuje dne štandardizovaným a známym spôsobom (obr.4.18).



Obr. 4.18 Typický panel nástrojov pre kreslenie

Výber inštrukcie časovača je možný zo zoznamu inštrukcií, ktoré sú k dispozícii v prilinkovaných knižniciach (obr.4.19).

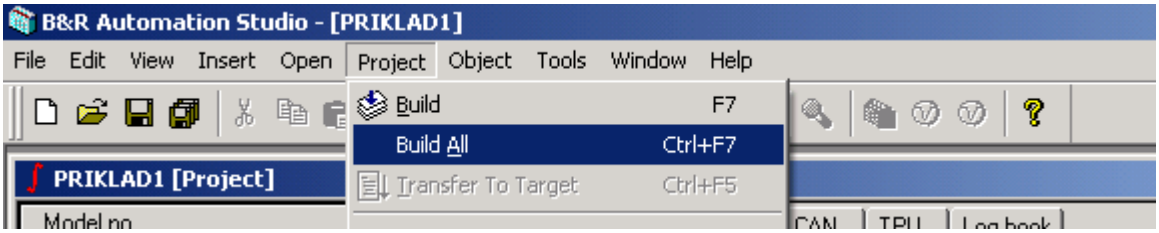


Obr. 4.19 Typický panel nástrojov pre kreslenie

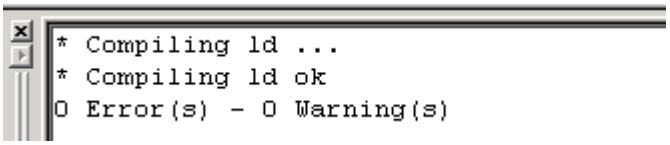
4. *Testovanie funkčnosti programu*

- *Formálne* - pomocou nástrojov v rámci prostredia PA

Formálne chyby sa prejavia väčšinou už pri preklade projektu do cieľového kódu pre PA. V rámci prostredia sa tento preklad vykoná vhodným príkazom (Build, compile a pod.).

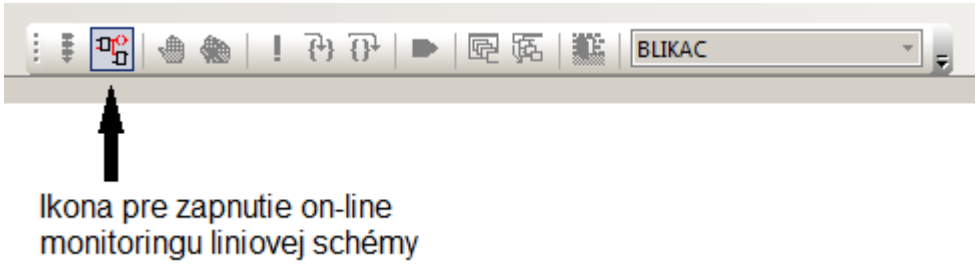


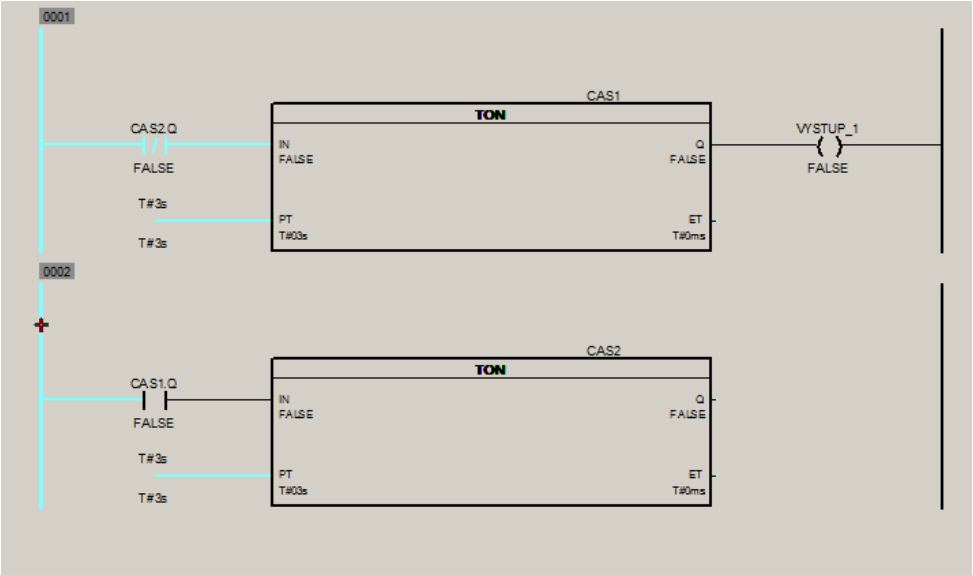
Výsledok tohto prekladu sa objaví v príslušnom okne. Pokiaľ sa tam vyskytnú formálne chyby, je potrebné ich odstrániť a preklad opakovať.



- *Logickej* - pomocou nástrojov v rámci prostredia PA

V tejto fáze vývoja projektu sa testuje jeho činnosť, väčšinou bez propojenia na reálnu technológiu pomocou simulácie vybraných premenných. Stav jednotlivých signálov a premenných je v rámci prostredia farebne vyznačený. Príklad ukazuje obr.4.20.



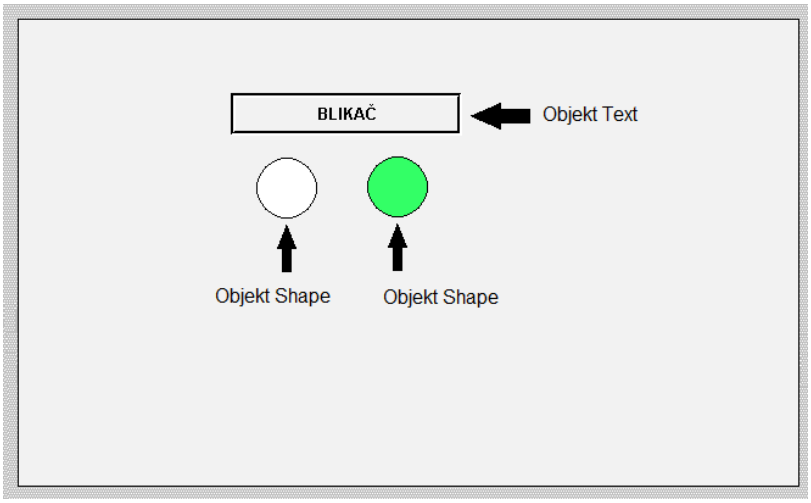


Obr. 4.20 Monitorovanie činnosti líniovej schémy

Konfigurácia operátorského panela

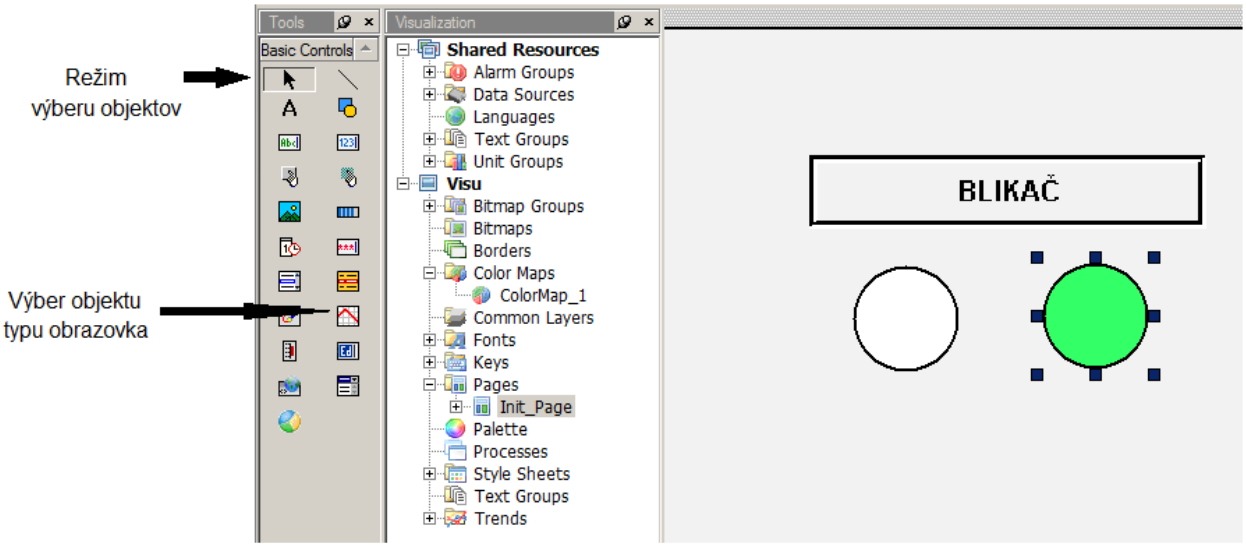
Naprogramovať (presnejšie nakonfigurovať) nejakú aplikáciu pre operátorský panel je potrebné urobiť v nasledujúcich krokoch:

1. *Nakresliť si základnú predstavu obrazovky*, resp. obrazoviek, ktoré budú slúžiť pre styk PA s operátorom. Pre tento príklad môže obrazovka vyzerať podľa obr.4.21. Tu je treba hlavne rozhodnúť, koľko bude obrazoviek, koľko na nich bude objektov a určiť ich typ.



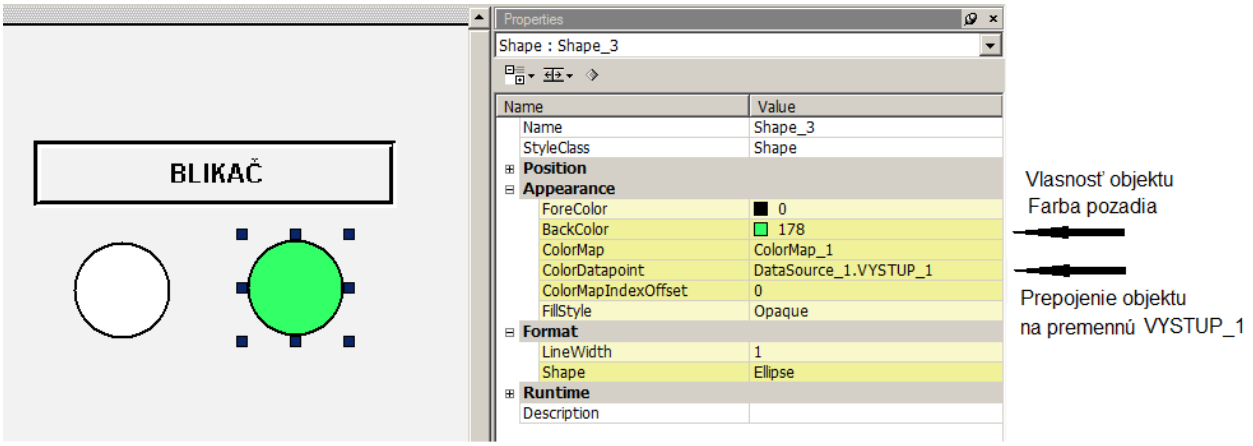
Obr. 4.21 Obrazovka pre zobrazenie blikača

2. *Vytvoriť navrhnuté obrazovky a objekty*. Každá obrazovka a každý objekt má svoje vlastnosti, ktorých počet závisí od typu objektu. Prácu s objektami (vytváranie, mazanie, kopírovanie, zmena polohy a pod.) umožňujú príslušné panely nástrojov (obr. 4.22).



Obr. 4.22 Panely nástrojov pre vytváranie objektov OD

3. *Definovať parametre jednotlivých objektov*. Najdôležitejšie sú formátové parametre (ako bude objekt vyzerať, jeho tvar, farby, rámček a pod.) a prepojenie objektu na príslušné premenné v programovateľnom automate (obr.4.23).



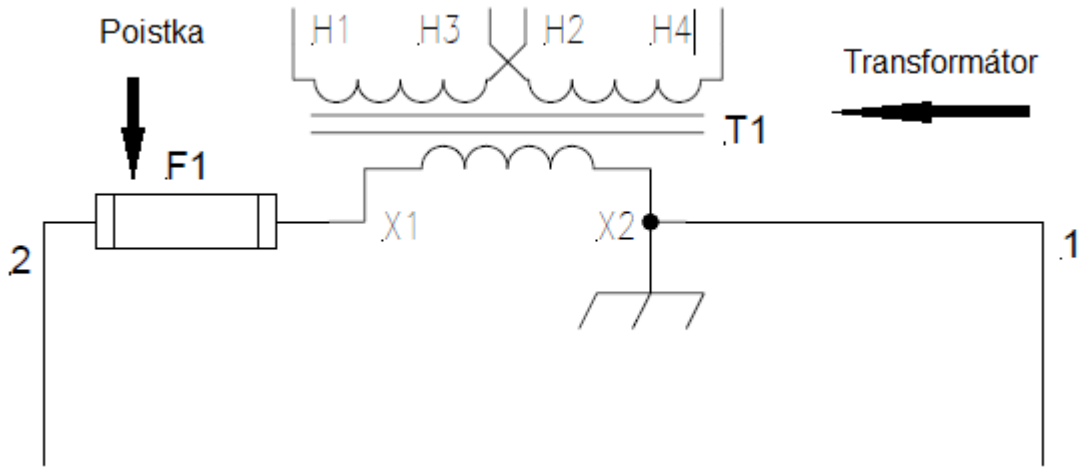
Obr. 4.23 Okno pre prácu s vlastnosťami objektu

5 LÍNIOVÁ SCHÉMA – LADDER DIAGRAM

Líniová schéma bola v minulosti používaná pre zakresľovanie automatizačných obvodov, realizovaných pomocou releovej logiky. Pretože PA vznikali pôvodne ako náhrada tejto logiky, prevzal sa aj základný spôsob zápisu ich činnosti vo forme *Ladder Diagramov*.

5.1 Líniová schéma elektrického obvodu

Každý elektrický akčný člen (elektrospotrebič) potrebuje mať napájanie, takže základom každej elektrickej schémy je zdroj vhodnej elektrickej energie. Tento je spoločný pre celú skupinu spotrebičov a jeden jeho vodič je takisto spoločný (v našom prípade sa nazýva nulový vodič a je obvykle uzemnený).



Riadenie technologického procesu musí uskutočňovať definované operácie nad stavmi procesu a na základe nich aktivovať príslušné spotrebiče. Pre riadenie logického typu to znamená realizovať základné logické operácie AND a OR.

Ak chceme rozsvietiť žiarovku, ak budú súčasne zopnuté dva spínače, znamená to realizovať boolovskú funkciu AND

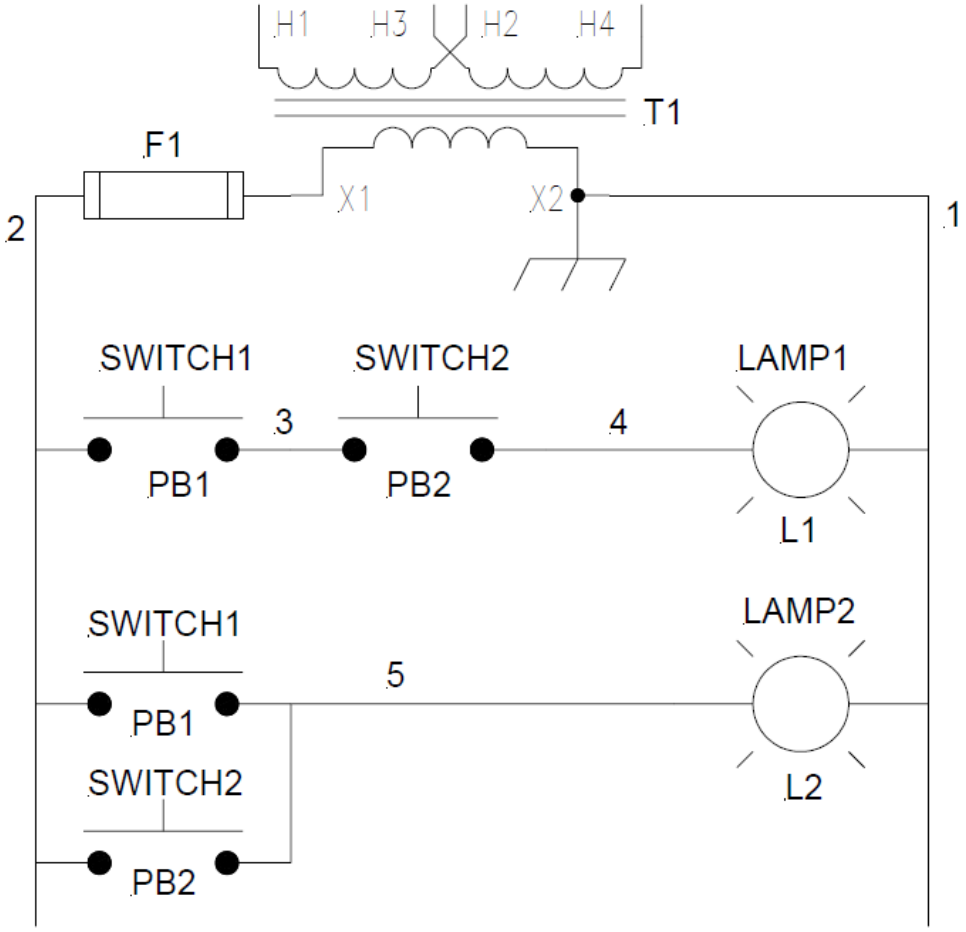
$Lamp1 = (Switch1) \bullet (Switch2)$

alebo sériové zapojenie príslušných spínačov.

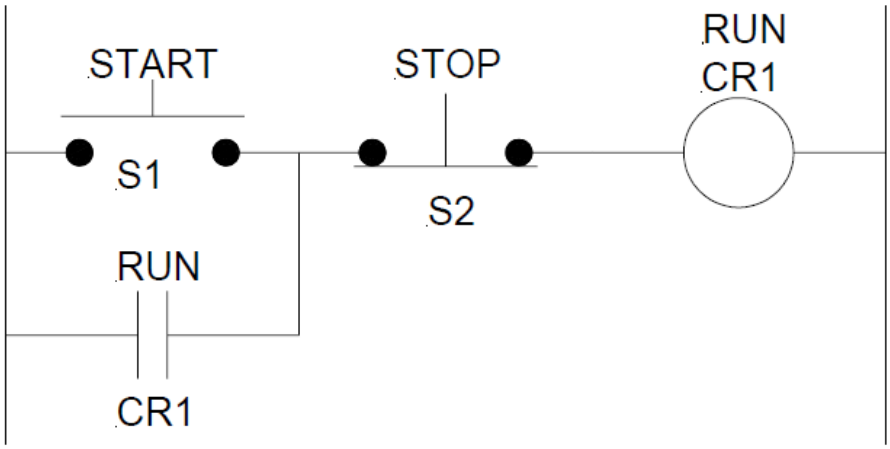
Ak chceme rozsvietiť žiarovku, ak budú zopnutý aspoň jeden za spínačov, znamená to realizovať boolovskú funkciu OR

$Lamp2 = (Switch1) + (Switch2)$

alebo paralelné zapojenie príslušných spínačov.



Veľmi často je potrebné zapamätať si stlačenie nejakého tlačidla, čo v zásade vieme vytvoriť zapojením so spätnou väzbou a nazývame ho samoprídrž (latch kontakt).

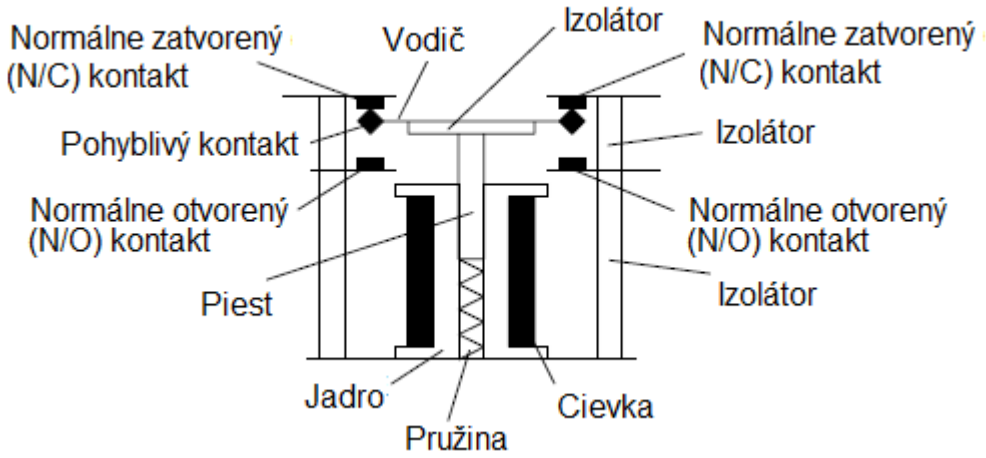


Základné stavebné prvky v elektrických riadiacich obvodoch sú:

- ***Tlačidlá***

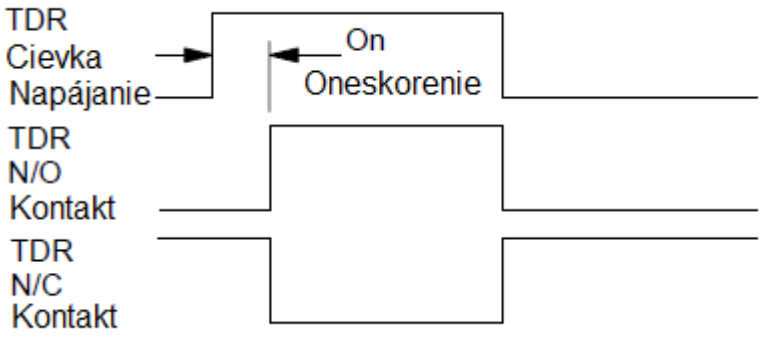
- ***Koncové spínače***

- ***Indikačné žiarovky***
- ***Relé***

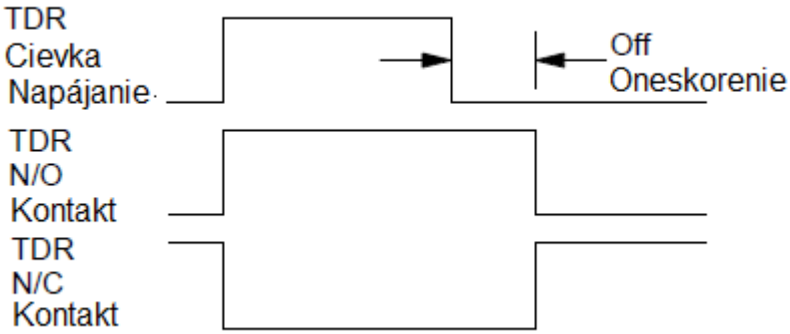


- ***Relé s časovým oneskorením***

  - s oneskorením po nábehu (TON)

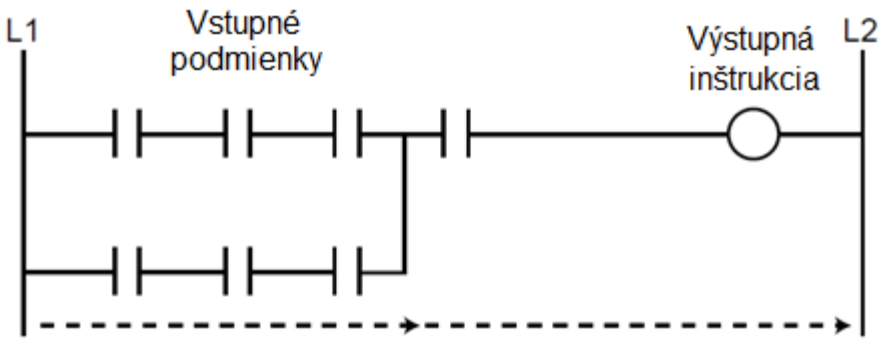


  - s oneskorením po odpade (TOF)



5.2 Líniová schéma (Ladder Diagram) v programovateľnom automate

Líniová schéma je grafické zobrazenie činnosti PA. Pozostáva z nezávislých vetiev (riadkov, línií), v ktorých sú grafickými značkami zobrazované jednotlivé inštrukcie (príkazy). Zároveň je možné sériovým alebo paralelným radením príkazov jednoducho realizovať základné logické funkcie AND a OR Booleovskej algebry. Každá vetva má vstupnú a výstupnú časť.

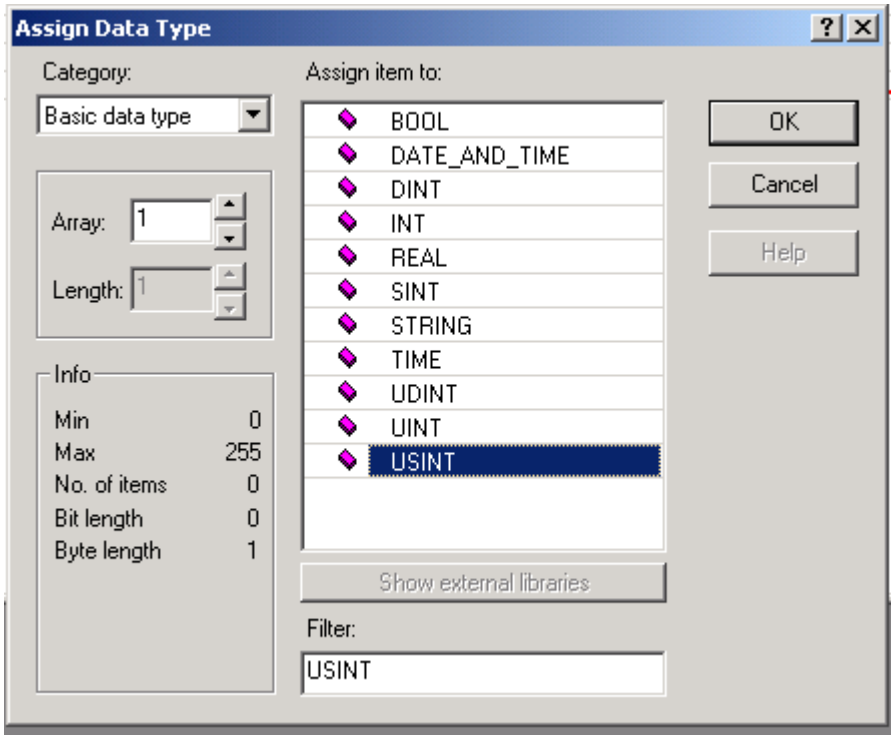


V klasickej releovej logike bol na výstupe jednej vetvy vždy jeden spotrebič (cievka relé, stykača, žiarovka a pod.). Vykonávanie príkazov v schéme sa deje 1x za pracovný cyklus PA sériovo zľava doprava a zhora dole a nie paralelne ako v elektrickej schéme (!!! teda záleží na poradí jednotlivých vetiev !!!).

Príkazy v líniovej schéme majú definovanú svoju **grafickú značku, mnemotechnickú značku** a **operandy**, nad ktorými vykonávajú svoju funkciu. Operandy môžu byť konštanty, pamäťové miesta príslušného typu (premenné) alebo celé oblasti pamäte. Adresácia týchto operandov môže byť v zásade priama alebo nepriama. Pre prehľadnosť programu a uľahčenie práce programátora sa operandy často označujú **symbolickými menami** (TAG, SYMBOL) a programovacie prostredie umožňuje pohodlne deklarovať (vytvárať databázu mien, adries a typov) a používať tieto mená.

5.2.1 Operandy

Pretože PA je vlastne mikroprocesor, pracuje vo svojej podstate v pevnej rádovej čiarke s operandami typu WORD. Pri riadení ovšem využívame aj iné typy operandov a preto je potrebné definovať u každého operandu jeho typ. Základné typy používaných datových údajov sú:



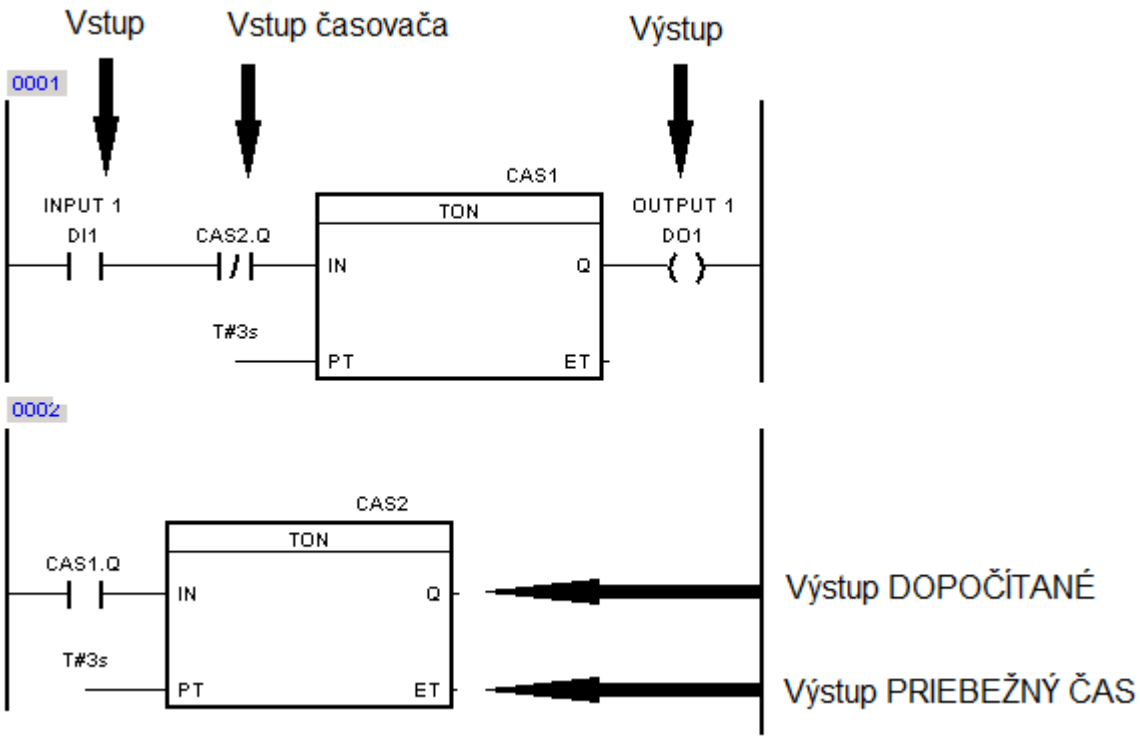
Jednotlivé operandy musia mať pridelenú presnú adresu v rámci pamäťového priestoru PA. Napríklad adresy vstupných a výstupných svoriek jednotlivých HW modulov PA, ktoré sú čítané a zapisované počas pracovného cyklu, súvisia presne s fyzickou polohou modulu a vstupnej svorky a sú umiestnené v tzv. I/O súbore (I/O tabuľke).

Dnešné PA umožňujú okrem základných datových typov používať aj užívateľom definované typy, ako sú polia, štruktúry a pod.

Úloha: Zistite, aké typy údajov môžete používať vo vašom PA a pre každý typ vyskúšajte príklad jeho adresovania pri deklarácii.

5.2.2 Inštrukcie

Inštrukcia líniovej schémy má svoj vstup, výstup a funkciu. Funkcia sa vykonáva iba vtedy, ak na vstupe inštrukcie je hodnota TRUE.



Pozn. Každá línia vyjadruje vlastne všeobecnú časovo-logickú funkciu s jedným výstupom, ktorú vieme slovne popísať, napr.:

Výstup DO1 nastav(=), AK *vstup DI1=1* **A** *CAS2 nedopočítal* **A** *CAS1 dopočítal*

alebo obrátene

AK *(vstup DI1=1* **A** *CAS2 nedopočítal* **A** *CAS1 dopočítal)* **POTOM** *(nastav DO1)*

Prepis slovného zadania úlohy do líniovej schémy pre PA potom vlastne znamená zaformulovať danú úlohu v takomto tvare pre jej jednotlivé výstupy.

6 TYPY INŠRUKCIÍ – PRÍKAZOV

Činnosť PA je definovaná pomocou inštrukcií (príkazov), ktoré pracujú nad operandami z pamäťového priestoru automatu. Pretože týchto inštrukcií je dnes už veľmi veľa, delíme ich do týchto hlavných skupín:

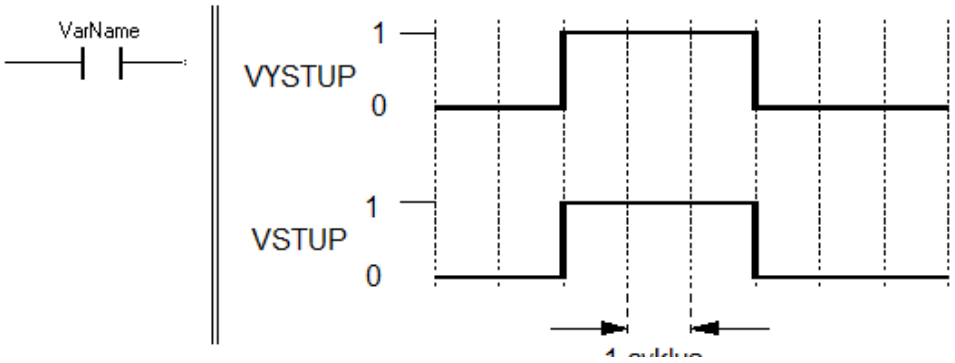
- Inštrukcie pre releovú (binárnu) logiku
- Príkazy pre časovače a čítače
- Porovnávacie príkazy
- Príkazy pre posuv operandov v pamäti
- Matematické a logické príkazy
- Príkazy pre riadenie programu
- Príkazy pre okamžité vstupy a výstupy
- Konverzné príkazy

Úlohu pre automat zapíšeme vo forme líniovej schémy pomocou vhodných inštrukcií a ich vzájomného prepojenia. Ako pri každom programovaní, aj tu existuje viacero možných líniových schém pre vykonanie tej istej úlohy. Pri úlohách riadenia elektrických pohonov existujú štandardné ovládacie líniové schémy pre jednotlivé typy pohonov. Pre syntézu zložitejších úloh je možné použiť známa matematické prostriedky z Booleovskej logiky, ako sú napr. Karnaughove mapy, ale väčšinou sa tieto úlohy riešia intuitívne. Celá úloha sa rozdelí na menšie programové celky, ktoré sa programujú a ladia samostatne a potom sa spoja do ucelenej aplikácie. Pre takýto postup je prispôsobené aj užívateľské prostredie PA, v ktorom je možné systematicky deliť úlohu na menšie časti (moduly, súbory, bloky a pod.).

6.1 Príkazy pre releovú logiku

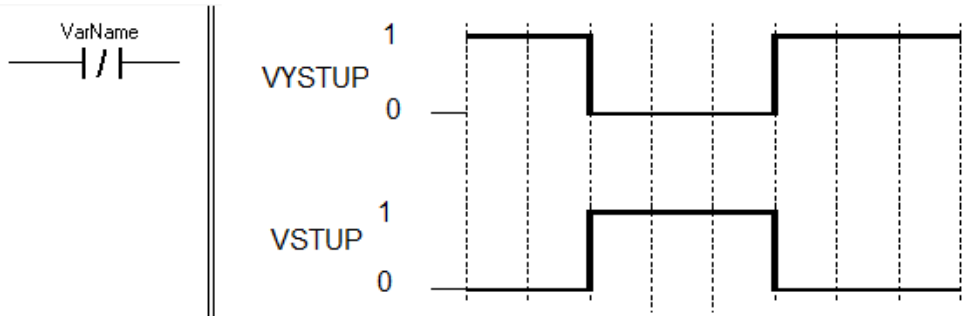
Používajú sa pre prácu so signálmi booleovského typu. Sú to tieto najčastejšie príkazy:

Test operandu na ON



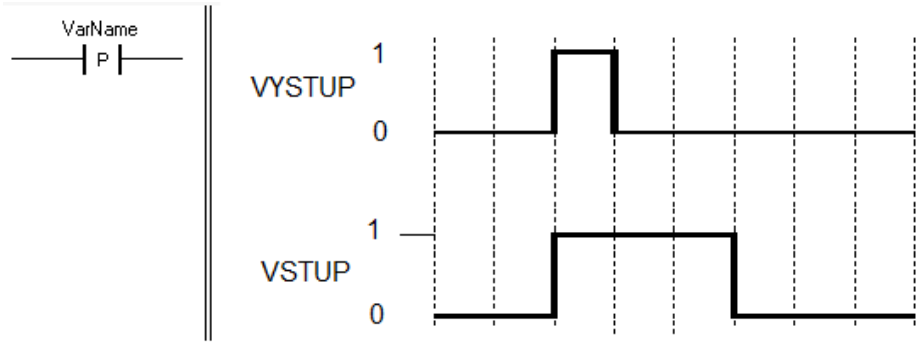
Schematická značka inštrukcie pripomína spínací kontakt relé a má aj podobnú funkciu. Ak je hodnota premennej *VarName* ON, príkaz prepustí svoj vstup na svoj výstup pre ďalšiu časť líniovej schémy.

Test operandu na OFF



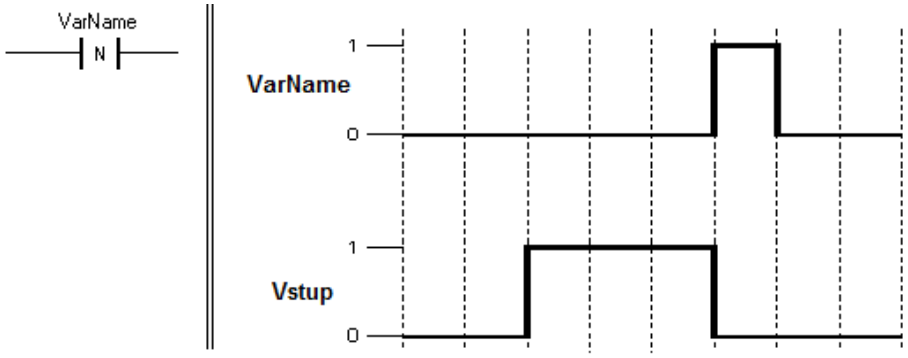
Schematická značka inštrukcie pripomína rozpínací kontakt relé a má aj podobnú funkciu. Ak je hodnota premennej *VarName* OFF, príkaz prepustí svoj vstup na svoj výstup pre ďalšiu časť líniovej schémy.

Test nábežnej hrany



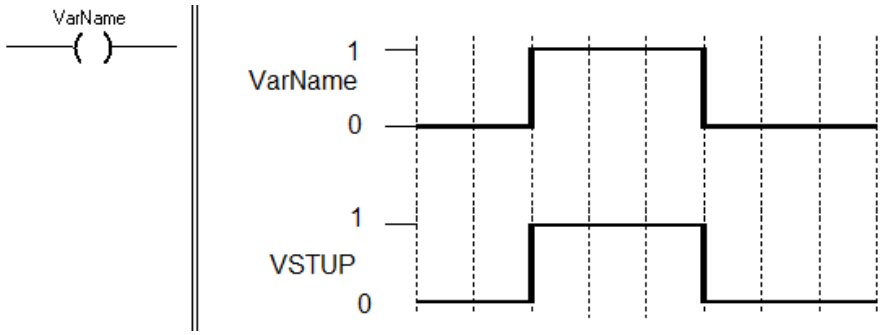
Inštrukcia testuje zmenu svojho vstupu z OFF na ON. Ak nastane táto zmena, nastaví svo výstup počas jedného pracovného cyklu automatu na ON. Premenná Varname slúži v tomto prípade ako pamäť predchádzajúcej hodnotu vstupu inštrukcie.

Test dobežnej hrany



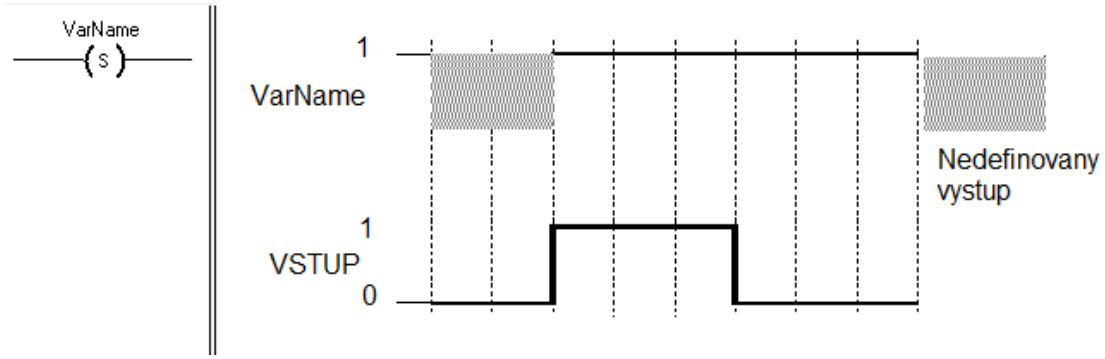
Inštrukcia testuje zmenu svojho vstupu z OFF na ON. Ak nastane táto zmena, nastaví svo výstup počas jedného pracovného cyklu automatu na ON. Premenná Varname slúži v tomto prípade ako pamäť predchádzajúcej hodnotu vstupu inštrukcie.

Nastavenie operandu podľa vstupu príkazu



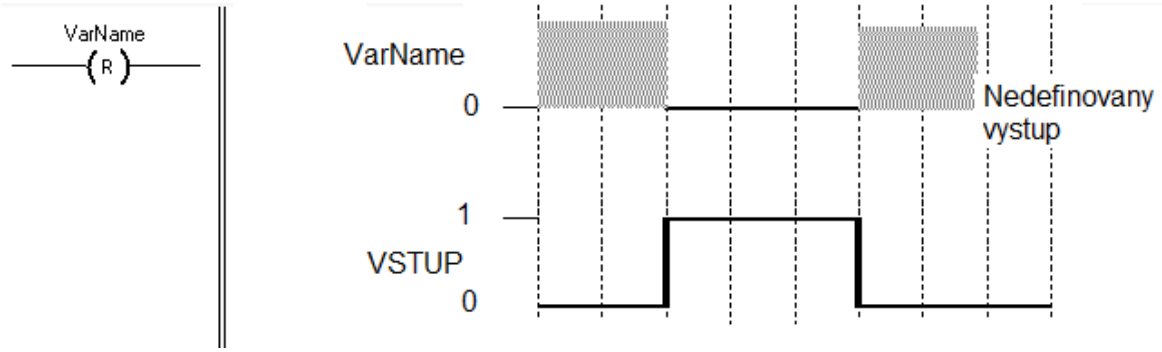
Ak má vstup inštrukcie hodnotu ON, nastavuje premennú *VarName* na ON. Ak má vstup inštrukcie hodnotu OFF, nastavuje premennú *VarName* na OFF. Jedná sa o výstupný príkaz, ktorý sa chová ako cievka rele.

Nastavenie operandu na ON



Ak má vstup inštrukcie hodnotu ON, nastavuje premennú *VarName* na ON. Ak má vstup inštrukcie hodnotu OFF, ponecháva operand v predchádzajúcom stave.

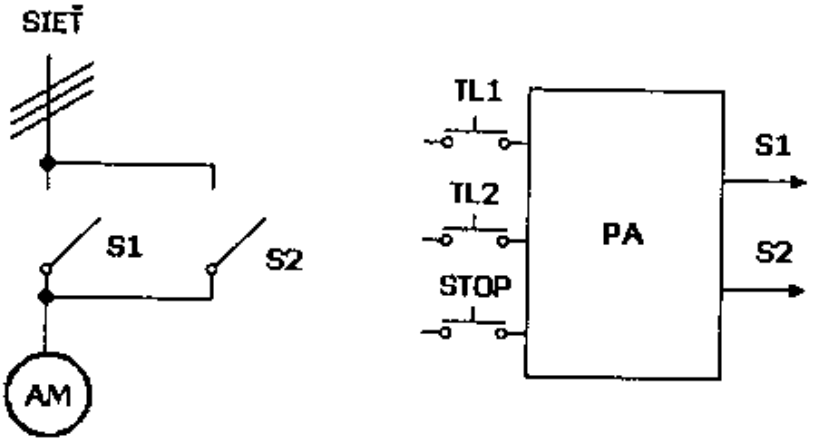
Nastavenie operandu na OFF



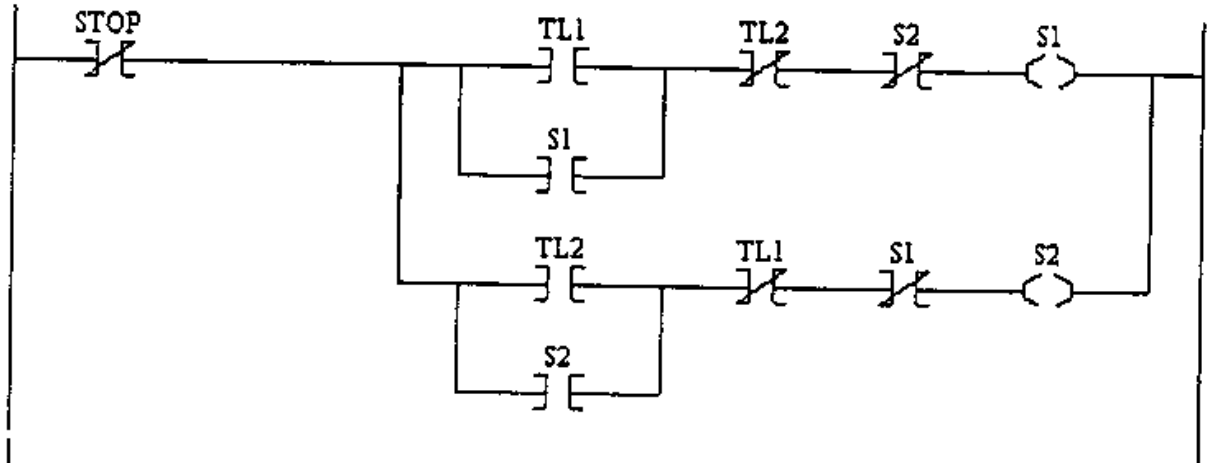
Ak má vstup inštrukcie hodnotu ON, nastavuje premennú *VarName* na OFF. Ak má vstup inštrukcie hodnotu OFF, ponecháva operand v predchádzajúcom stave. Je to komplementárna inštrukcia ku predchádzajúcej.

6.1.1 Riešený príklad:

Trojfázový asynchrónny motor je pripojený na sieť cez dva stykače S1 a S2. Stykač S1 pripája na motor jedno pradie fáz siete, stykač S2 iné poradie. PA má zabezpečiť rozbeh motora do jedného alebo druhého smeru podľa stlačenia tlačidiel TL1 a TL2. Pre zastavenie motora slúži tlačidlo STOP. Táto situácia je schématicky znázornená na nasledujúcom obrázku.



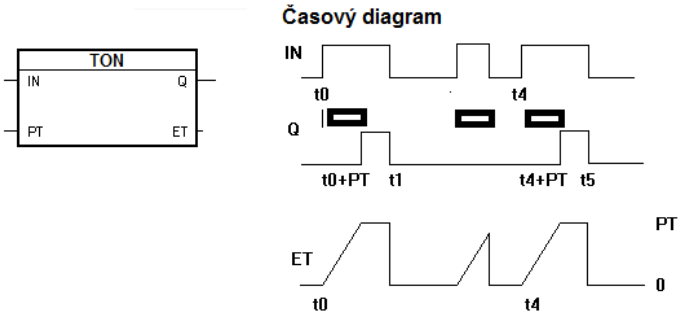
Možná líniová schéma s použitím inštrukcií pre binárnu logiku pre riešenie tejto úlohy je na nasledujúcom obrázku. Zo schémy je vidieť princíp vzájomného blokovania stykačov S1 a S2, ako aj samoprídrž tlačidiel TL1 a TL2.



6.2 Príkazy pre časovače a čítače

Tieto príkazy slúžia na meranie počtu udalostí. Časovače merajú internú zmenu počtu impulzov vnútorných hodín PA, čítače zmenu počtu externých udalostí.

- **TON - meranie času od nábežnej hrany vstupu (časové relé s oneskorením po nábehu)**

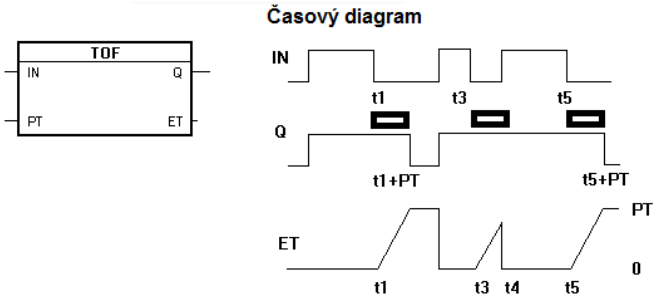


I/O	Parameter	Data Type	Description
IN	IN	BOOL	Vstupný signál
IN	PT	TIME	Časové oneskorenie
OUT	Q	BOOL	Výstupný signál, vzostupná hrana vstupného signálu je oenskorená o PT
OUT	ET	TIME	Uplynulý čas

Tento príkaz (funkčný blok) začína merať čas, prednastavený svojím vstupom PT od okamihu nábežnej hrany vstupu IN. Ak tento vstup má hodnotu ON počas celej doby merania času, výstup Q sa nastaví na ON. Ak vstup IN nadobudne hodnotu OFF, výstup Q sa zmení tiež na OFF. Operand tejto inštrukcie je zložený z viacerých hodnôt rôzneho typu (IN,Q,PT,ET ...).

- **TOF - meranie času od dobežnej hrany vstupu (časové relé s oneskorením po odpade)**

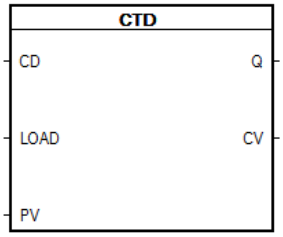
Tento príkaz (funkčný blok) začína merať čas, prednastavený svojím vstupom PT od okamihu dobežnej hrany vstupu IN. Ak tento vstup má hodnotu OFF počas celej doby merania času, výstup Q sa nastaví na OFF. Ak vstup IN nadobudne hodnotu ON, výstup Q sa zmení tiež na ON. Operand tejto inštrukcie je takisto zložený z viacerých hodnôt rôzneho typu (IN,Q,PT,ET ...).



I/O	Parameter	Data Type	Description
IN	IN	BOOL	Vstupný signál
IN	PT	TIME	Časové oneskorenie
OUT	Q	BOOL	Výstupný signál, zostupná hrana vstupného signálu je oenskorená o PT
OUT	ET	TIME	Uplynulý čas

- **CTD - počítanie dobežnej hrany vstupu**

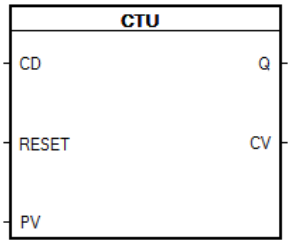
Príkaz počíta dobežné hrany na svojom vstupe CD. Keď napočíta počet, zadaný vstupom PV, nastaví svoj výstup Q na hodnotu ON. Vstup LOAD inicializuje čítač na počiatočný stav podľa PV a zároveň nuluje Q. Aktuálnu hodnotu napočítaných externých udalostí čítame na výstupe čítača CV.



I/O	Parameter	Data Type	Description
IN	CD	BOOL	Čítač CV je zmenešený o 1 zostupnou hranou na vstupe CD
IN	LOAD	BOOL	Ak vstup LOAD je TRUE, potom je čítač CV inicializovaný počiatočnou hodnotou PV
IN	PV	UINT	Počiatočná hodnota
OUT	Q	BOOL	Q je TRUE, ak CV je rovné 0. Inak je FALSE
OUT	CV	UINT	Čítač

- **CTU - počítanie nábežnej hrany vstupu**

Príkaz počíta nábežné hrany na svojom vstupe CD. Keď napočíta počet, zadaný vstupom PV, nastaví svoj výstup Q na hodnotu ON. Vstup LOAD inicializuje čítač na počiatočný stav podľa PV a zároveň nuluje Q. Aktuálnu hodnotu napočítaných externých udalostí čítame na výstupe čítača CV.



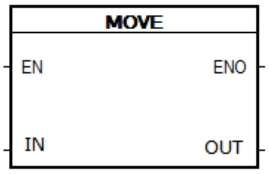
I/O	Parameter	Data Type	Description
IN	CU	BOOL	Čítač CV je zvýšený o 1 vzostupnou hranou na vstupe CU
IN	RESET	BOOL	Čítač CV je nastavený na 0, ak RESET je TRUE
IN	PV	UINT	Porovnávaná hodnota
OUT	Q	BOOL	Q je TRUE, ak čítač CV je väčší alebo rovný hodnote PV. Inak je FALSE
OUT	CV	UINT	Čítač

6.3 Príkazy pre posuv pamäte

Univerzálny príkaz pre presun časti pamäte je MOV. U niektorých typov automatov je pre každý typ operandu iný variant tohto príkazu.

- **MOV - presun (kopírovanie) časti pamäte**

Príkaz skopíruje premennú, pripojenú na vstup IN do premennej, pripojenej na výstup OUT. Rozsah vyhradenej pamäte pre vstupnú a výstupnú premennú musí byť rovnaký. Príkaz sa vykonáva len vtedy, ak na riadiaci vstup EN (Enable) je pripojený signál ON.

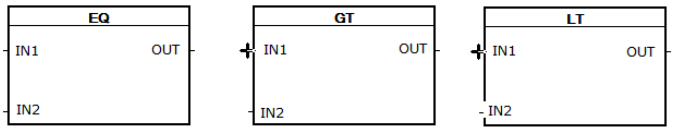


I/O	Parameter	Data Type
IN	IN	BOOL, SINT, INT, DINT, USINT, UINT, UDINT, REAL, TIME, DATE_AND_TIME, STRING, Array, Structure
OUT	OUT	BOOL, SINT, INT, DINT, USINT, UINT, UDINT, REAL, TIME, DATE_AND_TIME, STRING, Array, Structure

6.4 Porovnávacie príkazy

Tieto príkazy vykonávajú základné typy porovnania dvoch premenných a výsledok porovnania vyhodnotia výstupom logického typu (pravda/nepravda). U niektorých typov PA tieto príkazy nastavujú príslušné stavové príznaky v systémovej časti pamäte PA. Tieto je potom potrebné testovať hneď v ďalšej línii líniovej schémy, aby bol výsledok správny.

- **EQ, GT, LT,... – rôzne typy porovnania vstupov**

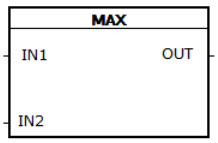


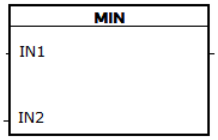
I/O	Parameter	Data Type	Description
IN	IN1	BOOL, SINT, INT, DINT, USINT, UINT, UDINT, REAL	Porovnávaná hodnota 1
IN	IN2	BOOL, SINT, INT, DINT, USINT, UINT, UDINT, REAL	Porovnávaná hodnota 2
OUT	OUT	BOOL	Výsledok porovnávania je TRUE, ak obidve porovnávané hodnoty sú =/>/< . Inak je FALSE

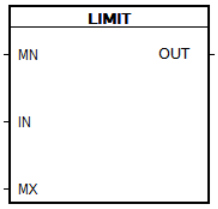
Medzi tieto príkazy môžeme zaradiť aj rôzne príkazy pre testovanie maxima/minima z dvoch premenných alebo obmedzenie hodnoty premennej na určité limity.

- **MIN,MAX,LIMIT – zistenie minimálnej,maximálnej hodnoty a obmedzenie premennej**

Činnosť týchto príkazov je zrejmá.







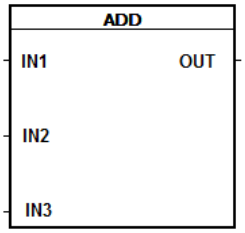
I/O	Parameter	Data Type	Description
IN	IN1	BOOL, SINT, INT, DINT, USINT, UINT, UDINT, REAL	Vstupná hodnota 1
IN	IN2	BOOL, SINT, INT, DINT, USINT, UINT, UDINT, REAL	Vstupná hodnota 2
OUT	OUT	BOOL, SINT, INT, DINT, USINT, UINT, UDINT, REAL	Výstupná hodnota obsahuje menšiu hodnotu z dvoch vstupov

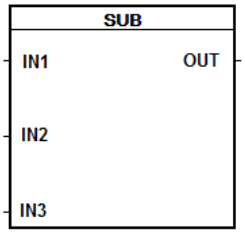
6.5 Matematické a logické príkazy

Tieto príkazy vykonávajú základné matematické operácie nad operandami rôzneho typu. Počet vstupov inštrukcie závisí od typu operácie, výstup je vždy jeden.

- **ADD,SUB – sčítanie, odčítanie operandov**

Tieto príkazy vukonávajú operáciu súčtu, resp. rozdielu na zvoleným počtom vstupov.

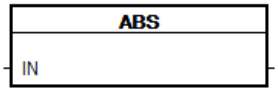


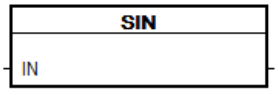


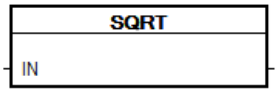
I/O	Parameter	Data Type	Description
IN	INx	BOOL, SINT, INT, DINT, USINT, UINT, UDINT, REAL, TIME	Vstupné hodnoty
OUT	OUT	SINT, INT, DINT, USINT, UINT, UDINT, REAL, TIME	Súčet vstupných hodnôt

- **ABS, SIN, SQRT,... – absolútna hodnota, sin,odmocnona ...**

Príkazy vykonávajú vybranú matematickú funkciu nad vstupným operandom.







I/O	Parameter	Data Type	Description
IN	IN	SINT, INT, DINT, USINT, UINT, UDINT, REAL	Vstupná hodnota
OUT	OUT	SINT, INT, DINT, USINT, UINT, UDINT, REAL	Absolútna hodnota

6.6 Konverzné príkazy

Operandy (premenné) potrebujeme častokrát spracovávať ako ten istý údaj, ale rôzneho typu. Napríklad pri práci s textovými reťazcami, dátumom a časom a pod. je potrebné ich zobrazovať operátorovi v nejakom štandardnom tvare, ale spracovávať ich je oveľa efektívnejšie v numerickej forme. Konverzné príkazy konvertujú navzájom operandy rôzneho typu medzi sebou.

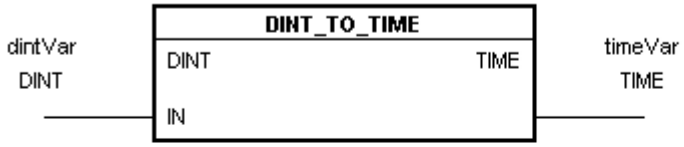
- **DINT_TO_TIME – konverzia 32bitového čísla na časový typ**

Čas je v PA zobrazený pre spracovanie ako numerický údaj,napríklad:

0 – T#0s

60000 – T#1m

3024 – T#3s24ms



I/O	Parameter	Data type	Description
IN	IN	DINT	Vstupná hodnota
OUT	OUT	TIME	Výstupná hodnota

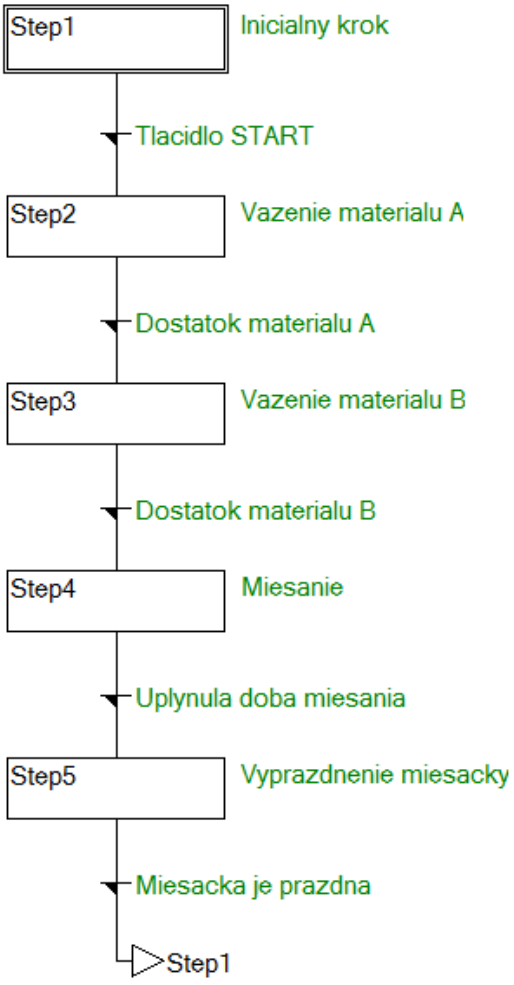
7 SEKVENČNÉ FUNKČNÉ DIAGRAMY

Riadenie takmer každej trochu zložitejšej technológie alebo linky sa vykonáva v určitých po sebe nasledujúcich etapách, stavoch, krokoch (*step*). V každom z týchto krokov je technológia v definovanom stave a jej riadenie vyžaduje od PA vykonávať určité akcie. Prechod do nasledujúceho stavu (kroku) nastáva po splnení určitej podmieky (podmienok), ktorú nazývame prechodová podmienka (transition). Takúto sériu krokov a postupných prechodov mezi nimi nazývame **sekvencia**.

Grafická forma zápisu takejto situácie, založená na štandarde normy IEC 61131-3, sa nazýva **sekvenčný funkčný diagram** (Sequential Function Chart – SFC).

Program pre PA, zapísaný v tejto forme, pozostáva zo série krokov, pospájaných navzájom líniami s prechodovou podmienkou.

Príklad: Veľmi často sa vyrábajú automatickými linkami rôzne zmesi materiálov podľa danej receptúry (miešanie farieb, výroba asfaltu, betónu, chemických lubrikácií a pod.). Automatické zmiešanie dvoch materiálov pomocou PA môžeme pomocou SFC zakresliť podľa obr. 7.1.



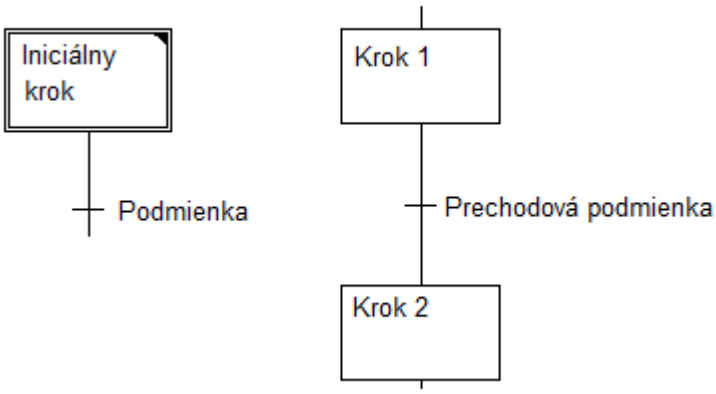
Obr. 7.1 Sekvenčný diagram pre zmiešanie dvoch materiálov

Základné prvky SFC sú:

- KROK (STEP)
- PRECHODOVÁ PODMIENKA (TRANSITION)
- SELEKTÍVNE VETVENIE (ALTERNATIVE BRANCH)
- SIMULTÁNNE VETVENIE (PARALEL BRANCH)
- SKOK NA STEP (JUMP TO STEP)

- **STEP**

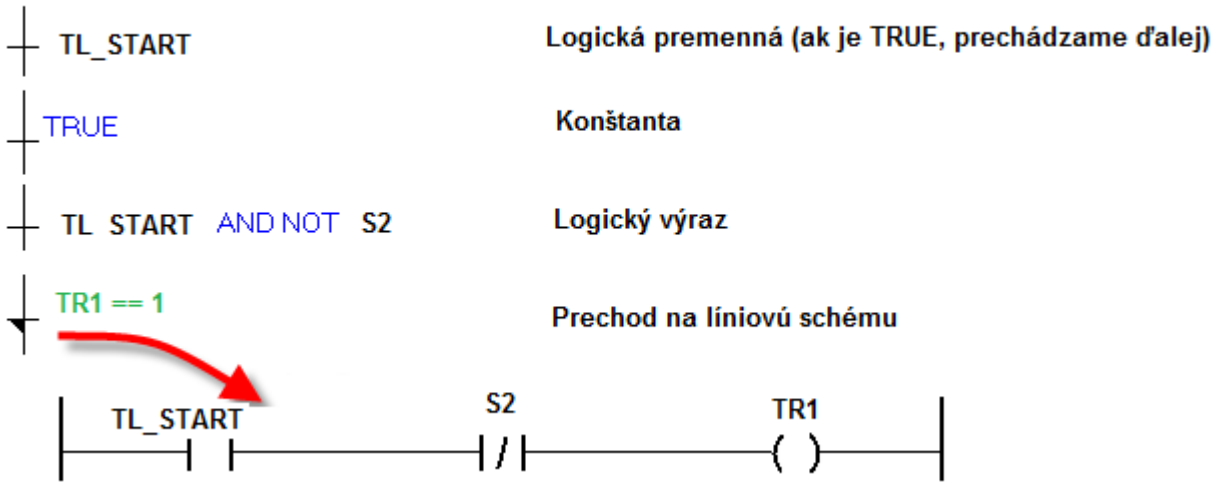
Akcie, vykonávané programovateľným automatom sú uložené v jednotlivých krokoch. Krok, ktorého akcie sú práve vykonávané, sa nazýva **aktívny krok**. Keď je krok aktívny, jeho akcie sa vykonajú raz za jeden pracovný cyklus PA. Krok, ktorý sa vždy vykoná ako prvý pri štarte PA, sa nazýva **iniciálny krok**.



Obr. 7.2 Iniciálny, resp. štandardný step

- **PRECHODOVÁ PODMIENKA (TRANSITION)**

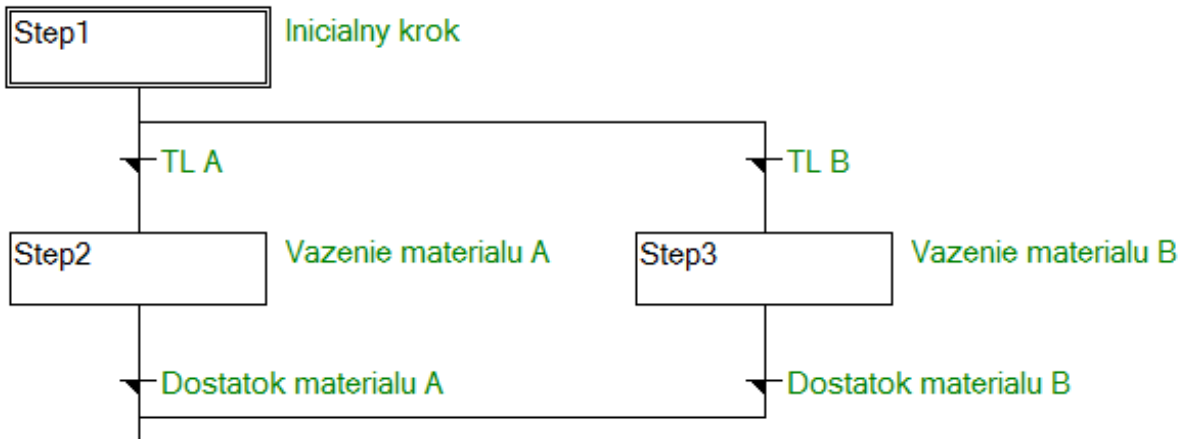
Je umiestnená medzi dvoma krokmi a v nej sú definované podmienky pre prechod na ďalší krok. Prechodová podmienka môže byť booleovská premenná, konštanta alebo skupina inštrukcií, nastavujúca zložitejšiu podmienku pre prechod (napr. vo forme líniovej schémy).



Obr. 7.3 Rôzne formy prechodovej podmienky

- **SELEKTÍVNE VETVENIE (ALTERNATIVE BRANCH)**

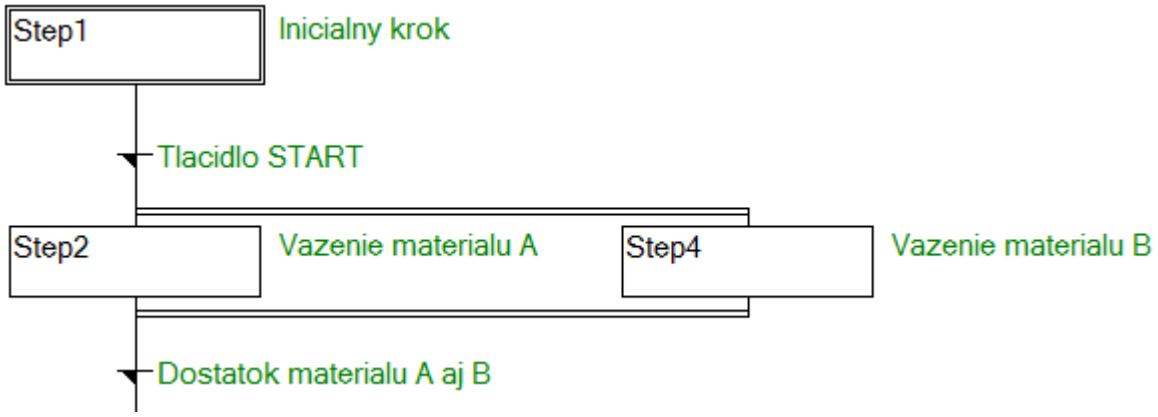
Vykonávanie jednotlivých krokov sa riadi v tomto prípade tým, ktorá podmienka bude splnená ako prvá. Kroky v jednotlivých vetvách musia mať vstupnú aj výstupnú prechodovú podmienku a vykoná sa vždy len jedna vetva, t.j. medzi krokmi je akoby funkcia ALEBO.



Obr. 7.4 Selektívne vykonávanie váženia

- **SIMULTÁNNE VETVENIE (PARALEL BRANCH)**

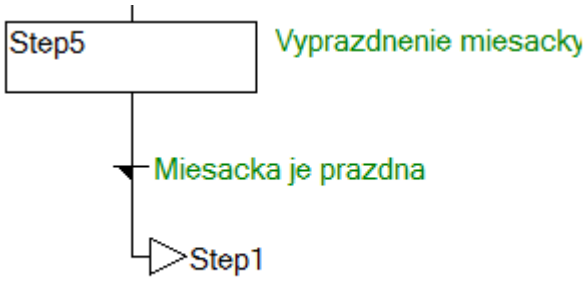
Všetky kroky v tomto prípade sa začnú vykonávať paralelne, ak sa splní ich spoločná vstupná prechodová podmienka a vykonávajú sa dovtedy, kým sa nesplní spoločná výstupná prechodová, t.j. medzi krokmi je akoby funkcia AND.



Obr. 7.5 Súčasné vykonávanie váženia

- **SKOK NA STEP (JUMP TO STEP)**

Skoky slúžia na riadenie toku stepov v rámci SFC. Skočiť je možné len na step, nie na prechodovú podmienku.

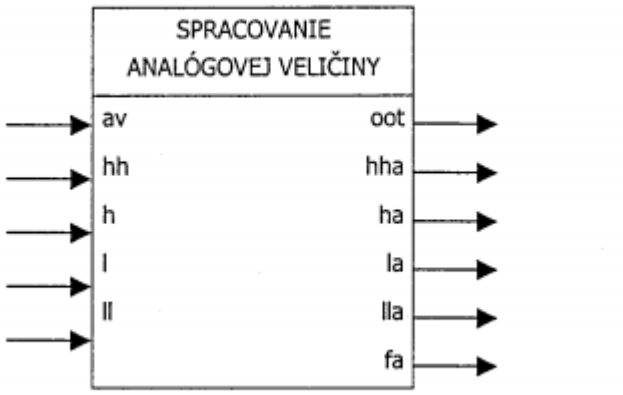


Obr. 7.6 Skok, zabezpečujúci pokračovanie sekvencie krokom Step1

8 ZÁKLADNÉ BLOKY LOGICKÉHO RIADENIA

8.1 Blok spracovania analógovej veličiny

Tento blok spracováva analógové vstupné veličiny a vytvára z nich binárne signály pre ďalšie logické spracovanie. Môžeme to znázorniť nasledujúcim obrázkom:



Obr. 8.1 Blok spracovania analógovej veličiny

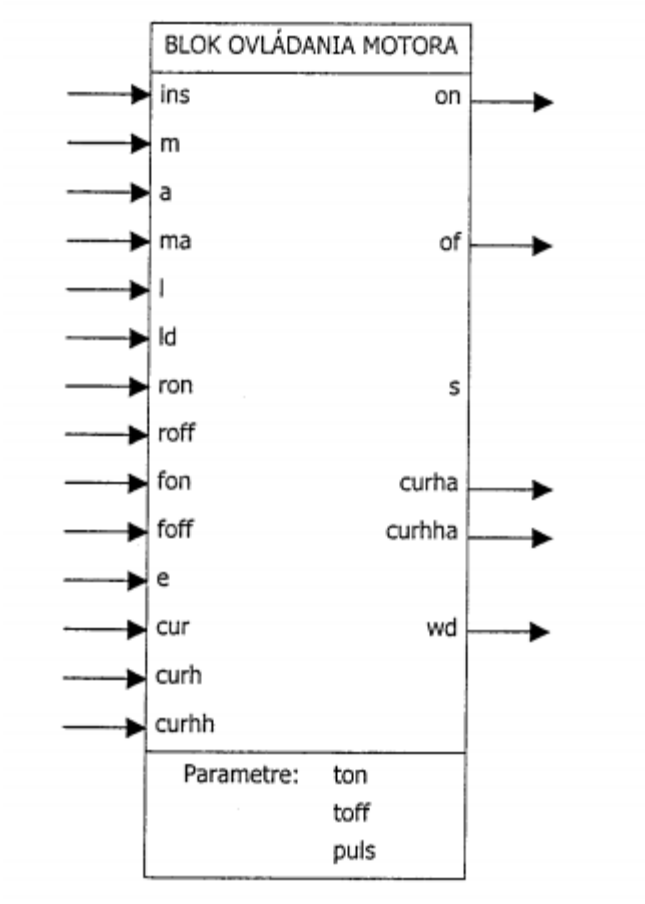
V tomto bloku jednotlivé signály majú nasledujúci význam:

- av: vstup analógovej (najčastejšie meranej) veličiny. Je to vlastne číslo typu INTEGER.
- hh: hranica kritického horného limitu
- h: hranica pracovného horného limitu
- l: hranica pracovného dolného limitu
- ll: hranica kritického dolného limitu
- out: upravený výstup meranej veličiny
- hha: alarm od hh
- ha: alarm od h
- la: alarm od l
- lla: alarm od ll
- fa: porucha merania (napr. prerušenie slučky, veličina mimo rozsah a pod.).

8.2 Blok riadenia motora

Tento blok ovláda zariadenia (najčastejšie motory), ktoré pracujú v dvoch základných stavoch – zapnutý (ON) a vypnutý (OFF). Motor sa ešte môže nachádzať v prechodných stavoch štartovanie (OFF→ON) a zastavovanie (ON→OFF). Riadenie týchto stavov sa vykonáva na základe povelov na vstupoch *l, m, a*. Blok vyberá povel podľa módu *ma,* resp. *d*. Výstupy bloku môžu byť zakázané pomocou signálov *ron, roff* alebo vnútené pomocou vstupov *fon, foff*. Blok

ďalej vyhodnocuje poruchy, ako sú prekročenie doby rozbehu *ton*, akoby dobehu *toff*, nesprávne rozbehnutie motora, nesprávne zastavenie motora a pod. Jeho činnosť je zobrazená na nasledujúcom obrázku (Obr.8.2).



Obr. 8.2 Blok riadenia motora

Jednotlivé signály bloku majú nasledujúci význam:

- ins: stav motora podľa spätnej hlášky
- m: žiadosť pre mód manuál
- a: žiadosť pre mód auto
- ma: prepínanie módov m/a
- l: žiadosť pre mód local
- ld: prepínanie módov locl/diaľka
- ron: zákaz posielania povelu on
- roff: zákaz posielania povelu off
- fon: vnútenie povelu on
- foff: vnútenie povelu off
- e: vonkajšia poruch

- cur: prúd motora
- curh: hranica pracovného prúdu motora
- curhh: kritický prúd motora
- on: riadiaci signál pre zapínanie
- off: riadiaci signál pre vypínanie
- s: aktuálny stav pohonu
- curha: alarm od curh
- curhha: alarm od curhh
- wd: číslo poruchového stavu
- ton: čas rozbehu
- tof: čas dobehu
- puls: voľba typu pre výstupy on/off – trvalé alebo pulzné.

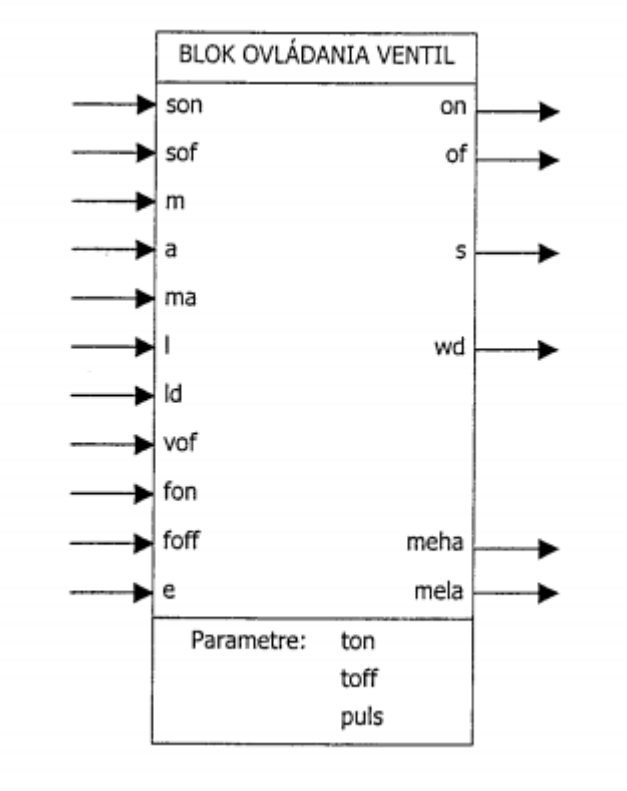
8.3 Blok riadenia ventilu

TENTO funkčný blok sa používa pre riadenie akčných členov, ktoré maj dva základné stavy – otvorený (OPEN) a zatvorený (CLOSE), ako sú napr. solenoidové ventily. Ventil sa ešte môže nachádzať v prechodných stavoch otváranie (CLOSE → OPEN) a zatváranie (OPEN → CLOSE). Riadenie týchto stavov sa vykonáva na základe povelov na vstupoch *l, m, a*. Blok vyberá povel podľa módu *ma*, resp. *ld*. Výstupy bloku môžu byť zakázané pomocou signálov *ron*, *roff* alebo vnútené pomocou vstupov *fon*, *foff*. Blok ďalej vyhodnocuje poruchy, ako sú prekročenie doby otvárania ton, doby zatvárania *toff*, nesprávne otvorenie ventilu, nesprávne zatvorenie ventilu a pod. Jeho činnosť je zobrazená na obr. 7.3.

Jednotlivé signály bloku majú nasledujúci význam:

- son: stav koncového spínača pre otvorený ventil
- sof: stav koncového spínača pre zatvorený ventil
- m: žiadosť pre mód manuál
- a:žiadosť pre mód auto
- ma: prepínanie módov m/a
- l: žiadosť pre mód local
- ld: prepínanie módov local/diaľka
- ron: zákaz posielania povelu on
- roff: zákaz posielanie povelu off
- fon: vnútenie povelu on
- foff: vnútenie povelu off
- e: vonkajšia porucha
- ov: riadiaci signál pre zapínanie
- of: riadiaci signál pre vypínanie
- s: aktuálny stav ventilu

- wd: číslo poruchového stavu
- ton: čas otvárania
- tof: čas zatvárania
- puls: voľba typu výstupy on/off – trvalé alebo pulzné



Obr. 8.3 Blok riadenia ventilu

8.4 Blok PID riadenia

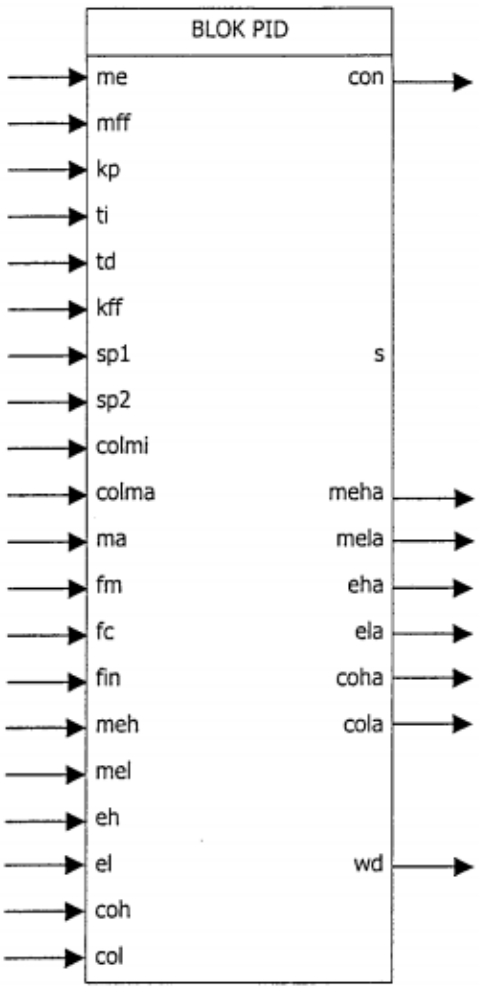
Hoci PA boli pôvodne určené pre riadenie logického typu, dnes už prakticky každý automat potrebuje vykonávať aj základné regulačné činnosti analógového typu, ako sú napr. regulácie tlakov, výšky hladiny, prietokov, teplôt a podobne. Štandardne si toto riadenie vyžaduje blok, ktorý vykonáva vlastne funkciu spojitého PID regulátora. Tento blok je buď súčasťou programového vybavenia PA (PA obsahuje takúto „inštrukciu“), alebo si ho musí užívateľ vytvoriť na základe inštrukcií matematických, posuvu a pod.
Funkčný blok PID generuje svoj riadiaci výstup *con* na základe aktívnej žiadanej hodnoty *spa* (z anglického *setpoint*) a meranej veličiny *me*. Tento výstupný signál bloku obvykle hýbe s polohovateľným ventilom alebo s rýchlosťou pohonu, ktoré slúžia ako akčné členy pri regulácii meranej veličiny. Blok PID sa snaží udržať regulačnú odchýlku *e* na nule pri pôsobení rôznych porúch. Štandardná rovnica pre PID blok je:

$$con = K_p\,(e + T_s/t_i \int e\,dt + t_d\,dme/dt) + k_{ff}\,m_{ff}$$

kde

- K_p je proporcionálne zosilnenie regulátora
- T_s je vykonávací interval bloku
- t_i je integračná časová konštanta
- t_d je derivačná časová konštanta
- k_{ff} je zosilnenie pre tzv. feedforward
- m_{ff} je veľkosť feedforwardu

Blok môže pracovať v dvoch módoch - manuálnom *m* a automatickom *a*. V manuálnom móde určuje žiadanú hodnotu obsluha cez vstup *sp1*, v automatickom je určuje iný blok alebo nadradený počítač cez vstup *sp2*. Módy *m* a *a* si prepína väčšinou obsluha. V prípade potreby (napr. prerušení meracieho obvodu) je možné blok nútene prepnúť do režimu *m* cez vstup *fm* alebo mu vnútiť výstup cez vstupy *fc*, *fin*. Cez ďalšie svoje vstupy blok umožňuje obmedziť hodnoty meranej veličiny, výstupu *con* a regulačnej odchýlky *e*.



Obr. 8.4 Blok PID regulátora

Jednotlivé signály bloku majú nasledujúci význam:

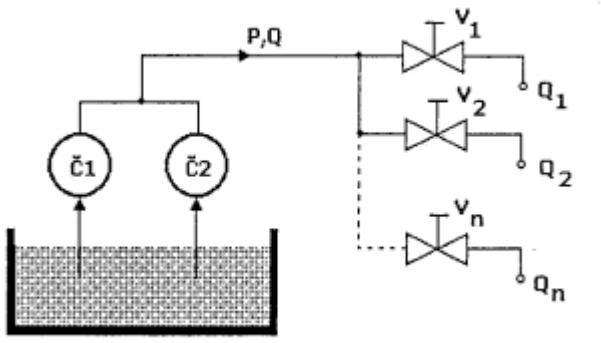
- me: meraná hodnota regulovanej veličiny
- mff: hodnota feedforwardu
- kp: zosilnenie regulátora
- ti: integračná časová konštanta
- td: derivačná časová konštanta
- kff: zosilnenie feedforwardu
- sp1: žiadaná hodnota pre mód manuál
- sp2: žiadaná hodnota pre mód auto
- colmi: minimálna hodnota výstupu con
- colma: maximálna hodnota výstupu con
- ma: prepínač módov manuál – auto
- fm: vnútenie manuálneho módu
- fc: vnútenie výstupu
- fin: hodnota vnúteného výstupu
- meh: hotný limit me
- mel: spodný limit me
- eh: horný limit e
- el: spodný limit e
- coh: horný linit con
- col: spodný limit con
- con: riadiaci výstup
- spa: aktuálny setpoint
- e: regulačná odchýlka
- meha: alarm od meh
- mela: alarm od mel
- eha: alarm od eh
- ela: alarm od el
- coha: alarm od coh
- cola: alarm od col
- wd: číslo vzniknutej poruchy

V tejto kapitole boli uvedené iba skutočne najpoužívanejšie funkčné bloky, ktoré sa používajú pri činnosti PA. Nie je tu napríklad uvedený blok riadenia polohovateľných ventilov, ktorého podrobnejší opis by si vyžadoval niekoľko desiatok strán. Pretože tieto bloky sa pri automatizácii technologických procesov opakovane využívajú, je vhodné (pokiaľ už samotný PA neobsahuje ich „inštrukciu“) pripraviť si ich univerzálnu verziu pre konkrétny PA, ktorá by pracovala nad dátovou štruktúrou, popísanou pre jednotlivé boky v tejto kapitole. Pre jednotlivé pohony, ventily a regulačné slučky potom stačí vyhradiť v pamäti PA príslušné dátové pole, ktoré iba napĺňame vstupmi a čítame z neho výstupy. Tento spôsob vytvárania programu pre PA vedie ku značnému zvýšeniu spoľahlivosti programu a zvyšuje jeho čitateľnosť, čo šetrí čas pri jeho ladení.

9 PRÍKLADY LOGICKÉHO RIADENIA V ELEKTRICKÝCH POHONOCH

9.1 Automatické udržiavanie tlaku v potrubí

Uvažujme časť technológie zobrazenú na obr. 9.1, ktorá predstavuje nádrž s vodou a hladinou *H,* čerpadlá *Č1* a *Č2*, ktoré sajú vodu z tejto nádrže a tlačia ju do spoločného výtlaku a vytvárajú v ňom tlak *P*. Na konci spoločného potrubia sú ručné ventily *V*1 až *Vn*, cez ktoré je možné odoberať množstvá vody *Q1* až *Qn*. Celkový odber vody z potrubia je daný súčtom týchto množstiev a je náhodný.



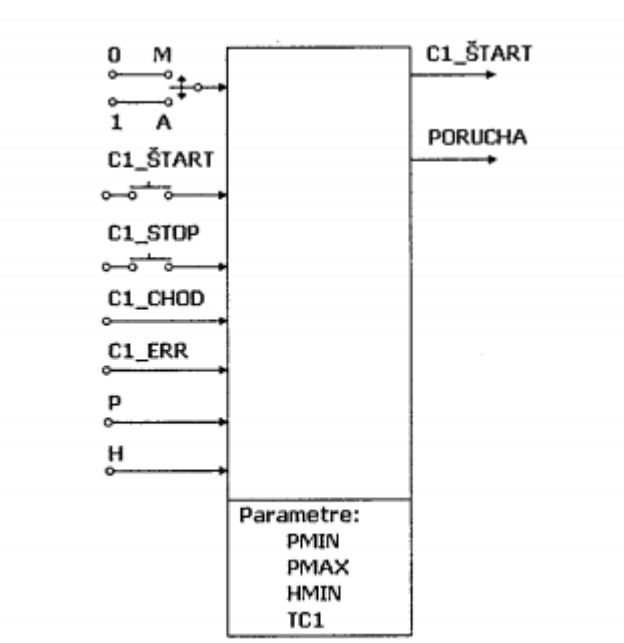
Obr. 9.1 Riadenie tlaku P v potrubí čerpadlami

Uvažujeme, že máme k dispozícii iba čerpadlo *Č1*, ktoré ma riadiť programovateľný automat.

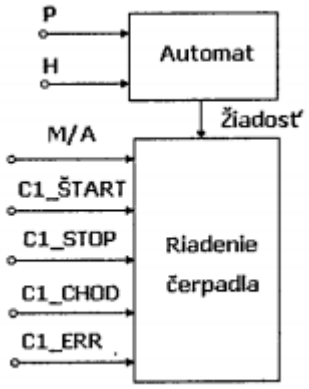
Slovný popis nech je nasledovný:
Čerpadlo môže pracovať v dvoch režimoch – ručnom (*M*) a automatickom (*A*). Voľbu režimu určí obsluha prepínačom. V ručnom režime sleduje obsluha na mieste zabudovaných ukazovateľoch stav tlaku v potrubí *P* a výšku hladiny v nádrži *H* a podľa týchto údajov cez tlačidlá ŠTART/STOP manipuluje s čerpadlom. V automatickom režime túto činnosť vykonáva PA na základe dvoch analógových vstupov, zodpovedajúcich *P* a *H*.

Cieľom práce čerpadla je udržiavať tlak v potrubí P v definovanom rozmedzí medzi hodnotami *Pmin* a *Pmax*. Zároveň čerpadla musí byť čerpadlo vypnuté pri určitej minimálnej hladine *Hmin,* aby nedošlo k jeho poškodeniu pri saní naprázdno. Obr. 9.2 znázorňuje vstupné a výstupné signály PA, ako aj jeho parametre.

Program v PA rozdeľme do dvoch blokov podľa obr. 9.3. Spodný (podradený) blok riadenie čerpadla ošetruje všetky jeho vstupno/výstupné signály, nadradený blok Automat podľa požiadaviek technológie generuje ŽIADOSŤ pre čerpadlo, podľa ktorej sa čerpadlo štartuje alebo zastavuje. Program pre PA je prakticky vždy rozdelený na takéto nadradené a podradené bloky.

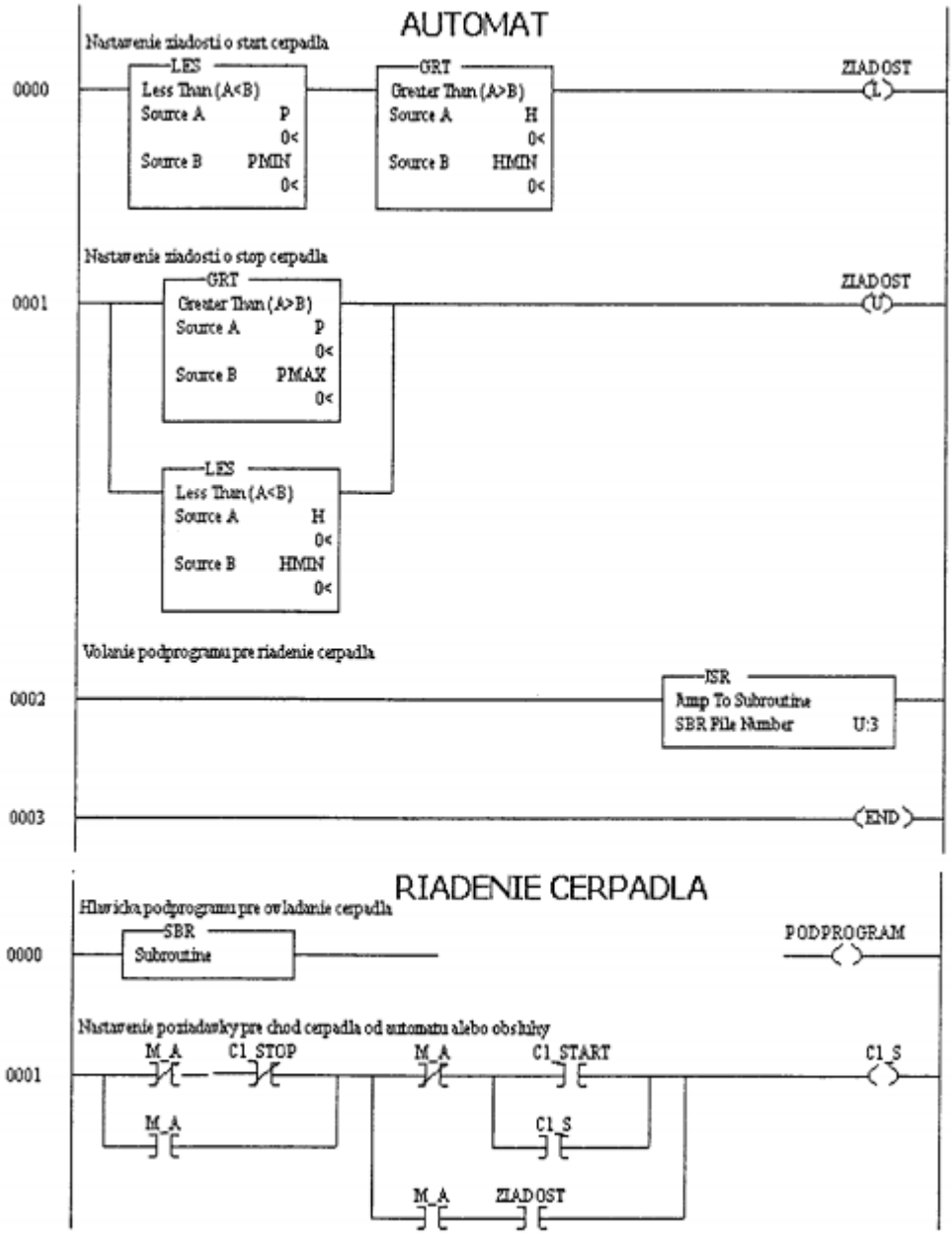


Obr. 9.2 PA pre riadenie tlaku v potrubí

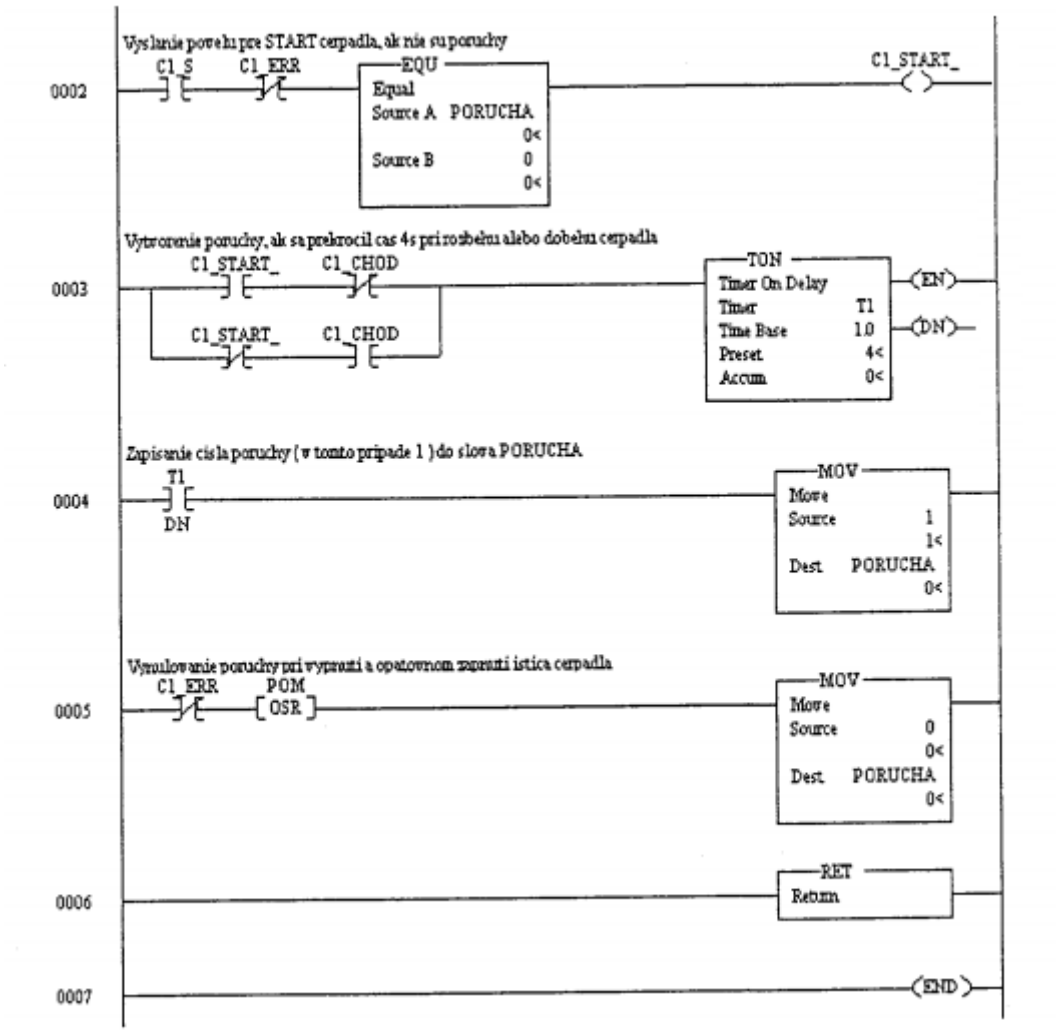


Obr. 9.3 Rozdelenie programu v PA na logické bloky

Programová realizácia uvedenej úlohy je na obr. 9.4 a obr. 9.5.



Obr. 9.4 Programová realizácia riadenia tlaku v potrubí

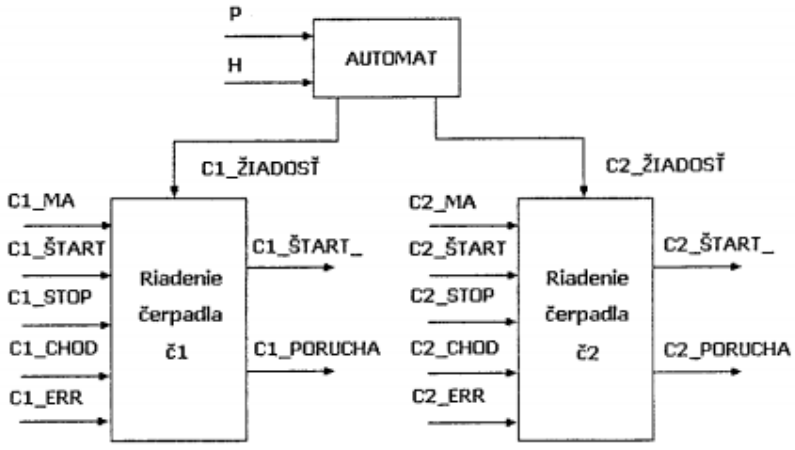


Obr. 9.5 Programová realizácia riadenia tlaku v potrubí

9.2 Prískok-odskok pohonov

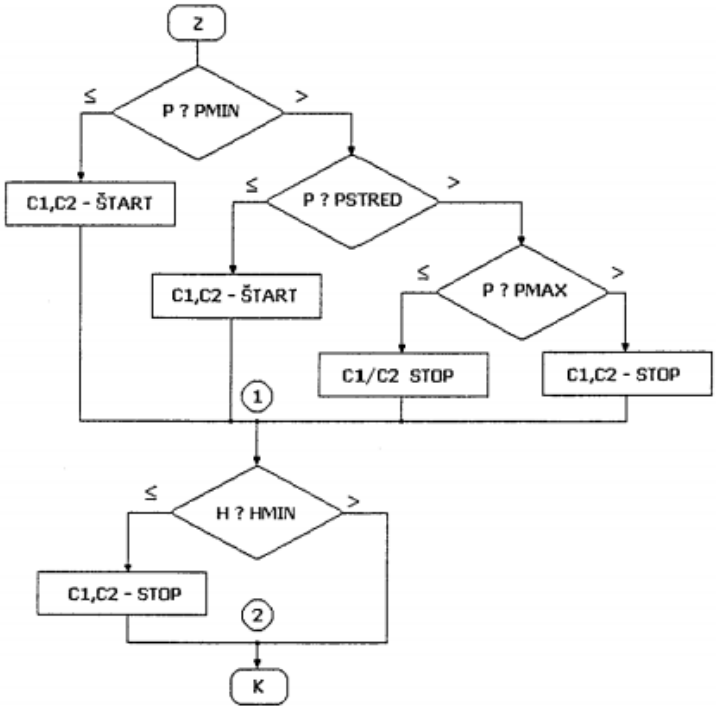
Často pracuje niekoľko menších pohonov ako čerpadlá do spoločného výtlaku. Umožňuje to znížiť celkové straty, zjednodušiť opravy a výmeny pohonov bez prerušenia prevádzky zariadenia a zálohovať pohony pre prípad poruchy. Obr. 9.1 ukazuje prípad dvoch čerpadiel *Č1* a *Č2*, ktoré pracujú do spoločného potrubia. V tomto prípade rozdelíme program v PA na logické celky podľa obr. 9.6.

Bloky riadenia jednotlivých čerpadiel ostávajú prakticky rovnaké ako v predchádzajúcom príklade, len do nich vstupujú zodpovedajúce signály od správnych čerpadiel. Pokiaľ je podobných čerpadiel málo (v našom príklade dve), realizuje sa blok riadenia pre každé z nich zvláštnym podprogramom, ktorého líniová schéma je rovnaká, menia sa iba adresy operandov. V prípade, že je podobných pohonov viac, realizujú sa jedným spoločným podprogramom, ktorému sa pri volaní presúvajú parametre príslušného pohonu.



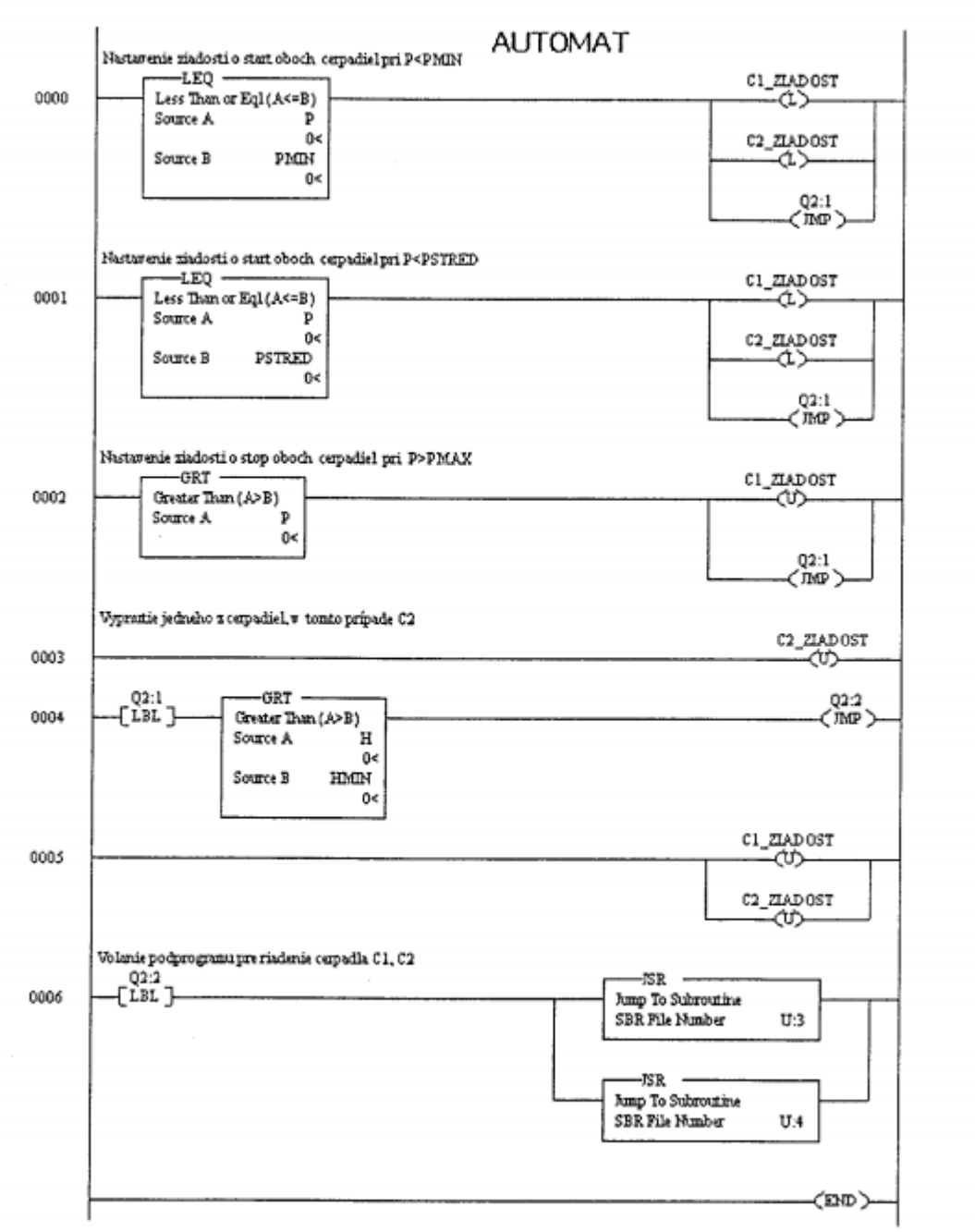
Obr. 9.6 Rozdelenie programu v PA

Programová realizácia samotného automatu je zložitejšia ako v predchádzajúcom príklade. Jej ideový vývojový diagram je uvedený na obr. 9.7. Aby sme vylúčili neustále zapínanie a vypínanie čerpadiel pri hraničných hodnotách tlaku PMIN, PMAX a PSTRED, je potrebné zaviesť do programu určitú hysteréziu, najjednoduchšie časovú. V tomto príklade nerozlišujeme, ktoré čerpadlo má priskočiť alebo odskočiť, bude to dané poradím vykonávania príkazov v líniovej schéme. V prípade viacerých čerpadiel môže záležať aj na poradí priskakovanie resp. odskakovanie, čo vedie na zložitejšiu programovú schému.



Obr. 9.7 Vývojový diagram pre prískok a odskok čerpadiel

Programová realizácia pomocou líniovej schémy je na obr. 9.8.



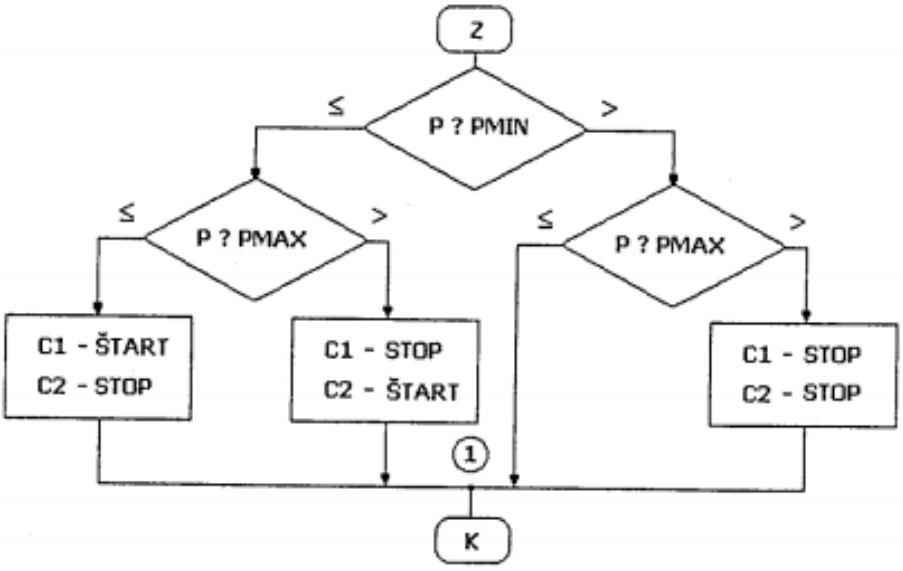
Obr. 9.8 Programová realizácia prískoku a odskoku čerpadiel

9.3 Záskok pohonov

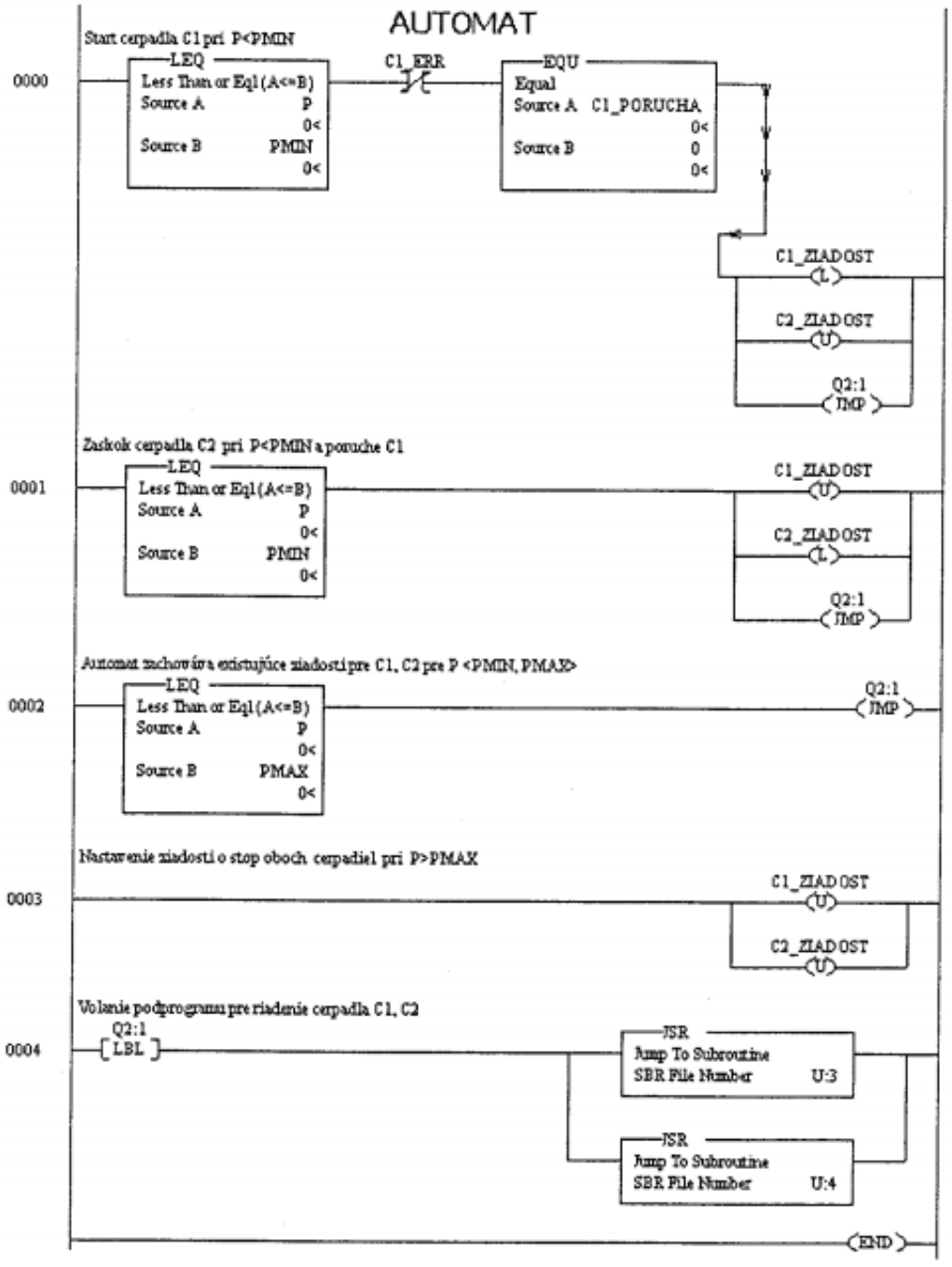
V predchádzajúcej podkapitole sme videli, že PA rozbiehal a zastavoval pohony z dôvodu dodržania technologického cieľa – tlaku v potrubí. Existujú prevádzky, v ktorých nie je možné prerušiť dodávku dopravovaného média (t.j. pohon musí stále bežať), aby nevznikli veľké (najčastejšie ekonomické) škody. Príkladom môže byť odsávanie spalín z priestoru horákov na kotle elektrárenského bloku, ktorého prerušenie vedie k odstaveniu celého bloku. Nový nábeh bloku stojí potom niekoľko stotisíc korún. Pretože však nie je možné zabezpečiť, aby pohon pracoval absolútne spoľahlivo (vždy môže dôjsť napríklad k jeho tepelnému preťaženiu a následne zastaveniu), obsahuje technológia v takýchto prípadoch niekoľko rovnakých pohonov, ktoré sa pri výpadkoch navzájom zaskakujú. PA v automatickom režime zabezpečuje potom vhodným spôsobom požiadavky na spustenie zaskakujúceho pohonu. V podstate je potrebné riešiť dve otázky:

- či vznikla požiadavka na záskok
- ktorý pohon (ak je ich viac ako dva) má zaskočiť

Pre jednoduchosť uvažujme opäť dve čerpadlá z technológie podľa obr. 9.1. Samotné riadenie jednotlivých čerpadiel bude obdobné ako v predchádzajúcich príkladoch. Vývojový diagram riadenia automatu je na obr. 5.8 a samotný program pre PA na obr. 5.9.



Obr. 9.9 Vývojový diagram automatu pre záskok pohonov

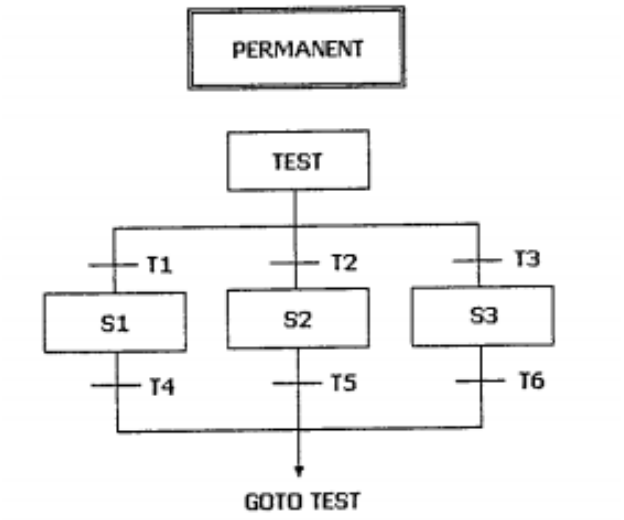


Obr. 9.10 Program pre záskok pohonov

9.4 Riadenie sekvencie

V predošlých prípadoch sme formálne popisovali činnosť PA pomocou vývojových diagramov, ako je to známe z „klasického“ programovania. Pre jednoduchšie úlohy tento spôsob popisu vyhovuje, pri zložitejších úlohách sú však vývojové diagramy veľmi neprehľadné a nečitateľné. Ich štruktúra vyhovuje skôr sériovému ako cyklickému vykonávaniu príkazov, ktorý je typický pre PA. Preto sa činnosť PA často popisuje vo forme tzv. sekvenčných diagramov, ktoré zobrazujú všetky možné stavy technologického procesu a prechody medzi nimi. Podrobnejšie o tomto spôsobe zápisu úloh pojednáva kapitola 7. V nasledujúcom príklade sa pokúsime ukázať, akým spôsobom je možné programom v PA realizovať riadenie takejto sekvenčne popísanej úlohy.

Uvažujeme ako príklad prískok a odskok pohonov podľa podkapitoly 9.2. Činnosť PA popisuje vývojový diagram na obr. 9.6. Vidíme, že automat z hľadiska výstupov generuje tri možné kombinácie svojich výstupov - štart oboch čerpadiel, štart iba jedného čerpadla a stop oboch čerpadiel. Rozhodovanie o aktuálnej kombinácii sa deje na základe okamžitých hodnôt tlaku a hladiny *H*. Sekvenčný popis by mohol mať tvar podľa obr. 5.10.



Obr. 9.11 Vzorový sekvenčný diagram činnosti automatu

Systematický popis jednotlivých krokov (stepov) sekvencie by bol nasledujúci:

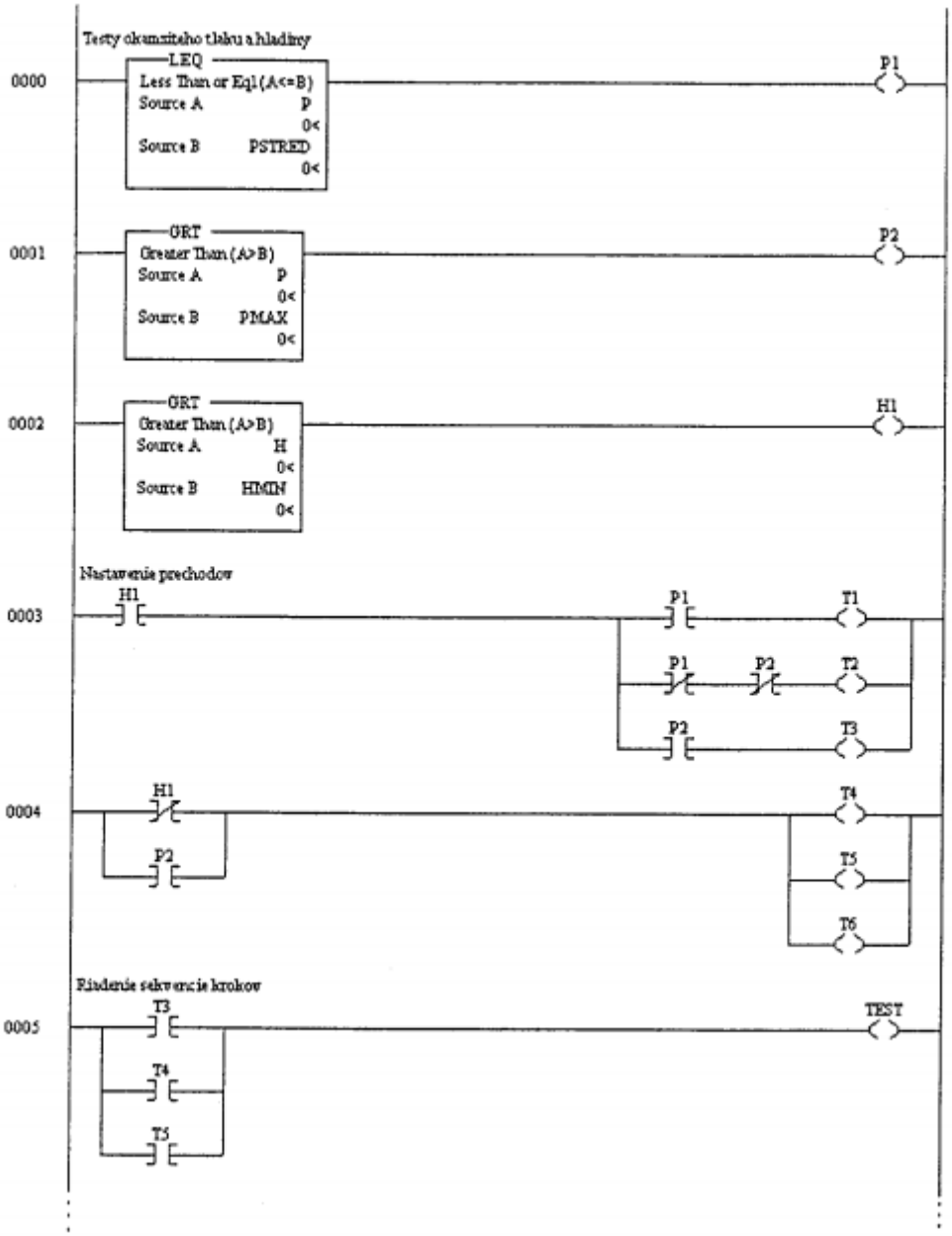
- PERMANENT – V každom programovom cykle testuje okamžité hodnoty tlaku *P* a hladiny *H*. Generuje binárne údaje *P1, P2, H1*, ktoré sa používajú pre určenie stavu jednotlivých prechodov *T1* až *T6*. Tieto údaje zodpovedajú jednotlivým pásmam tlaku *P* a hladiny *H* (napr. *P1* je *ON*, ak *P* < *STRED*, *P2* je *ON*, ak *P* je > *PMAX*, *H1* je *ON*, ak *H*>*HMIN*). Tento blok zabezpečuje aj permanentné riadenie stavov a prechodov sekvencie,. Pre tento účel má každý stav svoje číslo. Riadenie spočíva vo výbere správneho aktuálneho čísla (stavu) a realizácii jeho príkazov.
- TEST – je to umelý krok (t.j. nenastavuje žiadne výstupy), ktorý prebehne po vyvolaní vždy iba raz a umožní nastaviť automat do správneho stepu pri zmene *P*, *H*.

- S1 – krok, v ktorom sa žiada štart oboch pohonov
- S2 – krok, v ktorom sa žiada stop jedného pohonu
- S3 – krok , v ktorom sa žiada stop obidvoch pohonov.

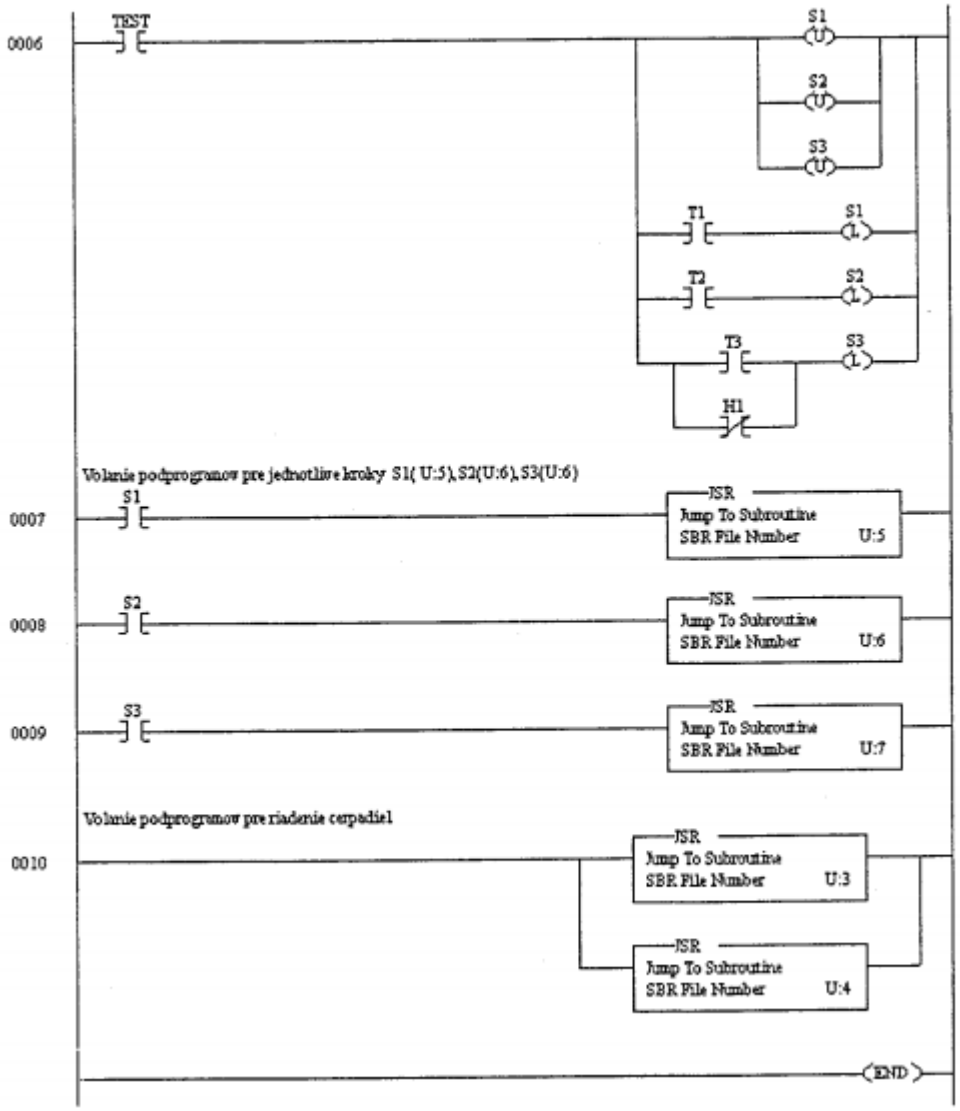
Podmienky pre aktualizáciu jednotlivých stavov sú v nasledujúcej tabuľke:

Č.	H1	P1	P2	Stavy
1	0	0	0	S3
2	0	0	1	S3
3	0	1	0	S3
4	0	1	1	zakázaný
5	1	0	0	S2/T2
6	1	0	1	S3/T3
7	1	1	0	S1/T1
8	1	1	1	zakázaný

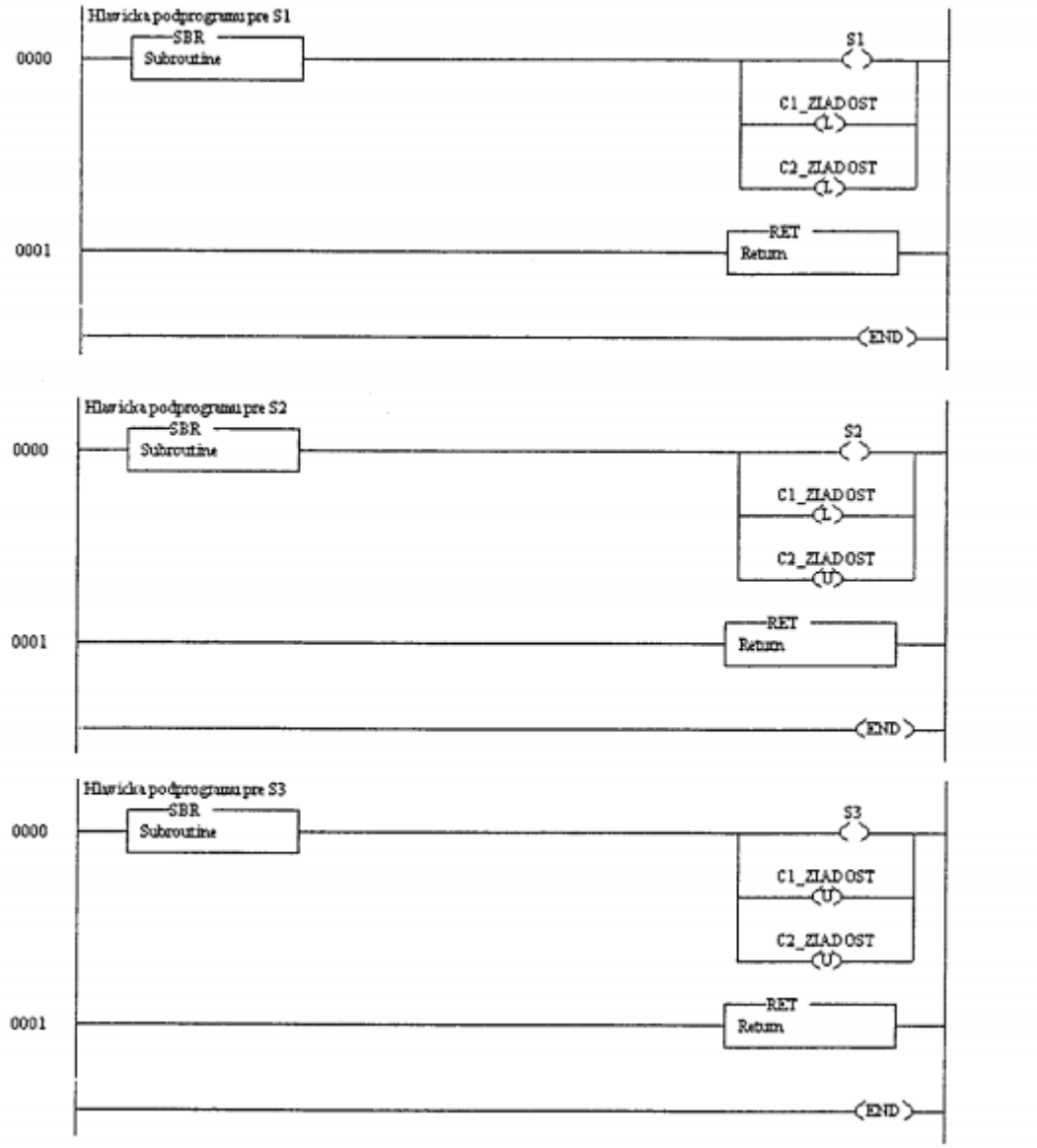
Všimnime si, že napr. stav č.8 je zakázaný, t.j. logicky nemôže nastať. Tlak *P* nemôže byť zároveň menší ako *PMIN* a väčší ako *PMAX*. Pri takomto systematickom ošetrení všetkých stavov je možné aj tento stav ošetriť (napr. chybovým hlásením), pričom podľa vývojového diagramu na obr. 9.6 tento stav vyvolá nedefinované chovanie PA. Program pre PA je ukázaný na obr. 5.11.



Obr. 9.12 Program pre riadenie sekvencie (1.časť)



Obr. 9.12 Program pre riadenie sekvencie (2.časť)

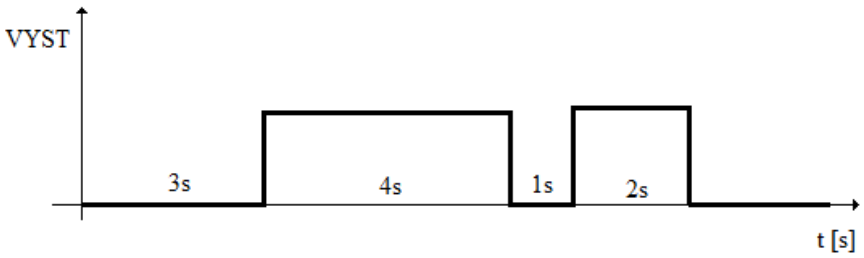


Obr. 9.12 Program pre riadenie sekvencie (3.časť)

10 PRÍKLADY A ZADANIA

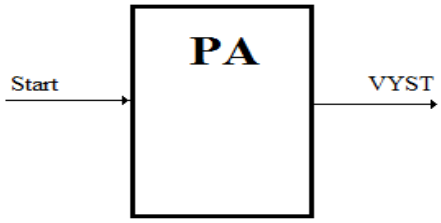
10.1 Príklad 1 - Generátor pulzov.

Pomocou PA generujte výstupný signál podľa obrázka, ktorý sa vygeneruje jednorázovo po stlačení tlačidla *Start* (obr. 8.1):



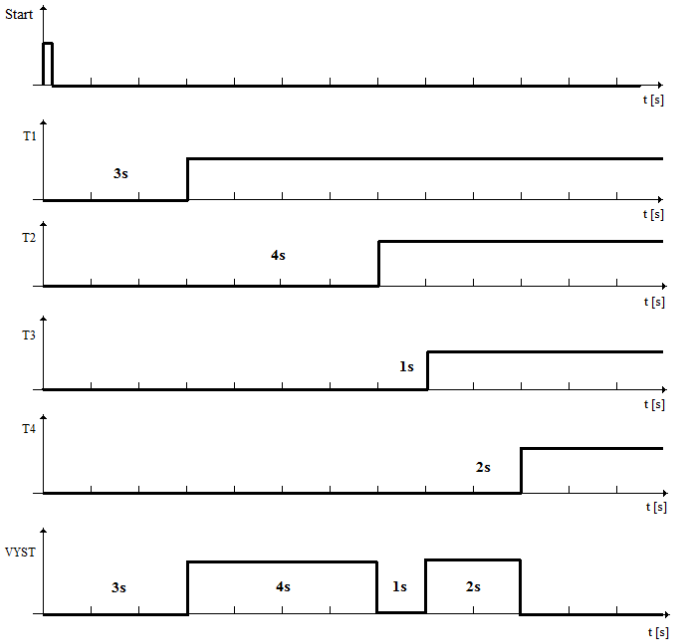
Obr. 10.1 Priebeh výstuoného signálu

Riešenie:
Programovateľný automat bude mať v tomto prípade jeden vstup a jeden výstup (obr. 8.2).



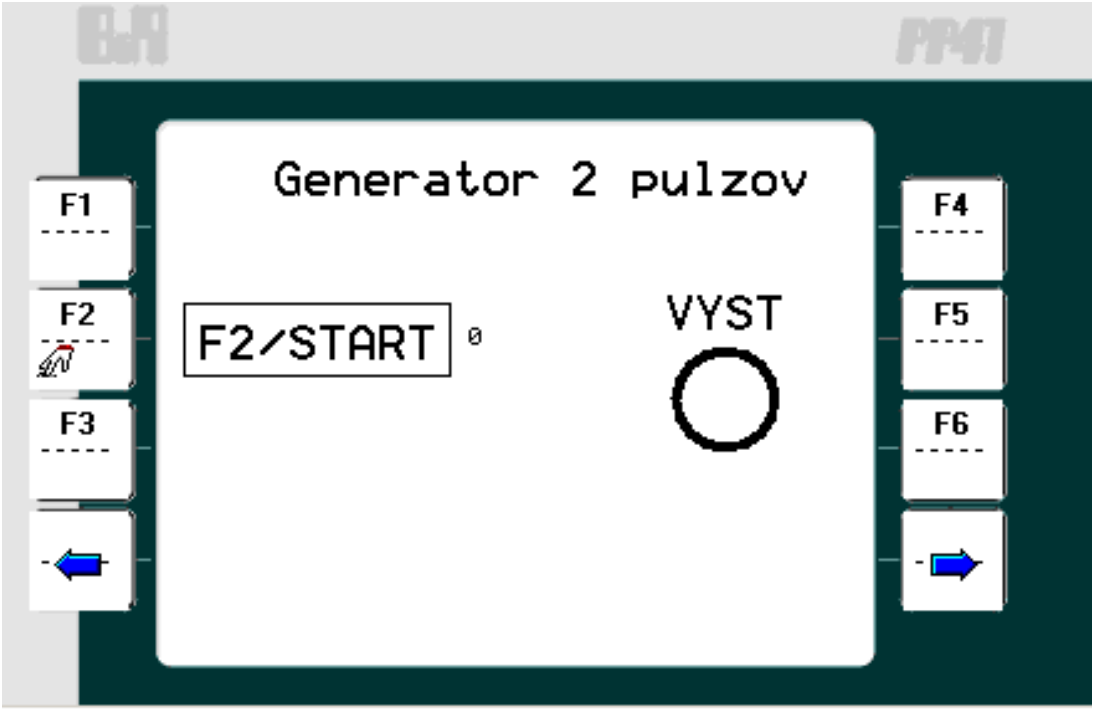
Obr. 10.2 Zobrazenie PA z hľadiska vstupov a výstupov

Časy jednotlivých intervalov budú merať štyri časovače podľa nasledujúceho diagramu (obr. 8.3):



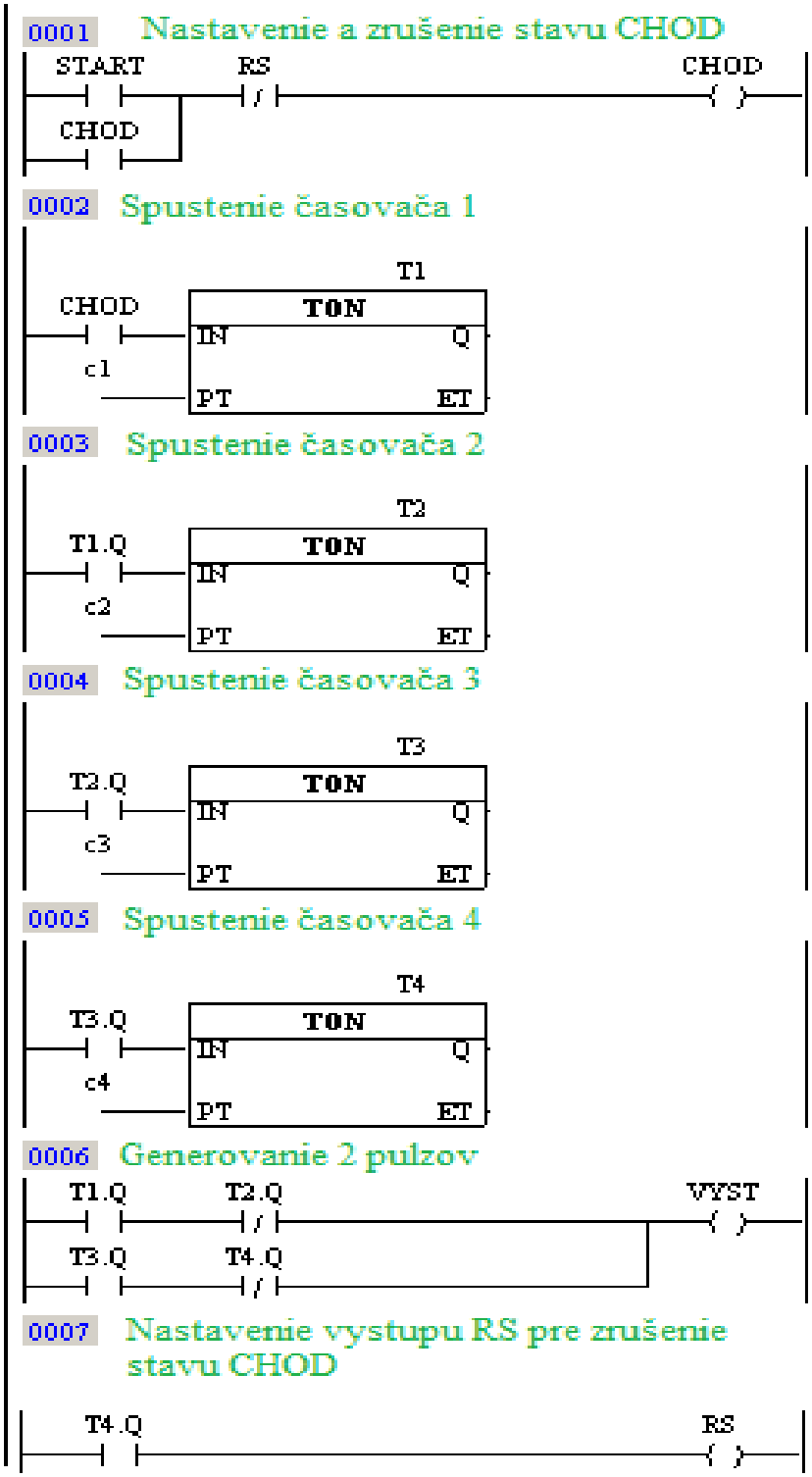
Obr. 10.3 Priebehy jednotlivých časovačov

Vzhľad vizualizačnej obrazovky može vyzerať podľa obrázka (obr. 8.4):



Obr. 10.4 Vzhľad vizualizačnej obrazovky

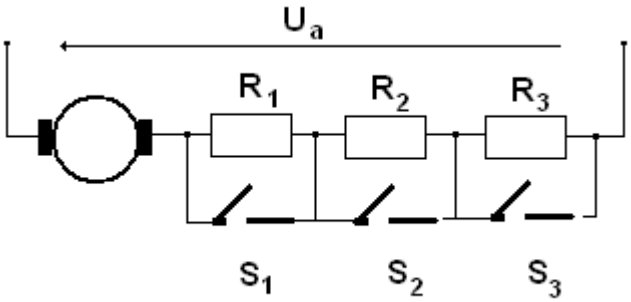
Príklad líniovej schémy, realizujúcej tento generátor, je uvedený na ďalšom obrázku (obr. 8.5).



Obr. 10.5 Riešenie príkladu formou líniovej schémy

10.2 Príklad 2 – Odporový spúšťač jednosmerného motora

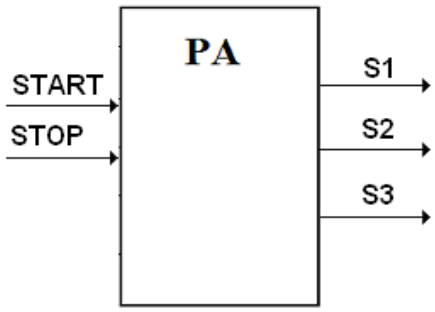
Pomocou PA navrhnite riadenie 3-stupňového odporového spúšťača JSCB motora podľa obrázka:



Popis činnosti:

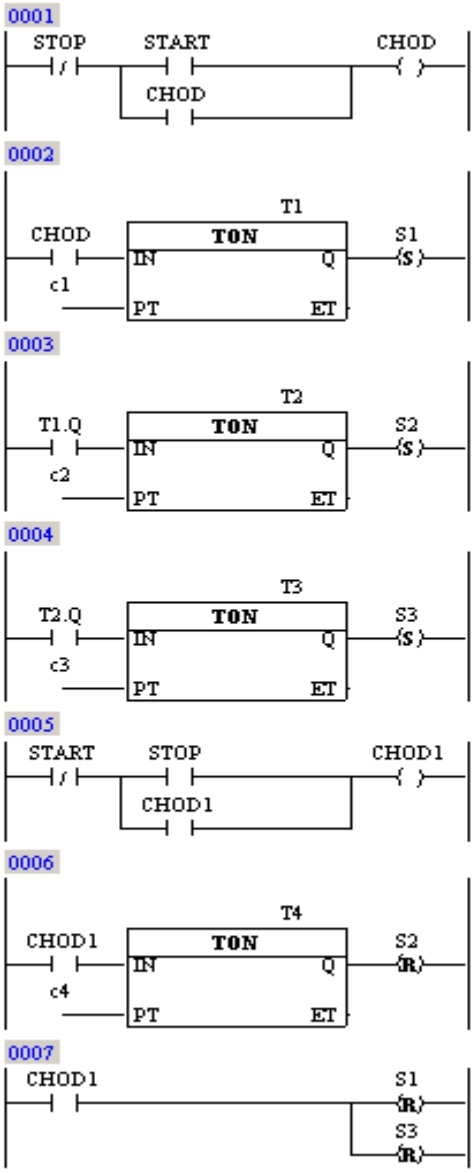
1. Rozbeh pohonu - po stlačení tlačidla START
   - odpor R1 sa vyradí po 4s
   - odpor R2 sa vyradí po 3s od vyradenia R1
   - odpor R3 sa vyradí po 2s od vyradenia R2
2. Brzdenie pohonu - po stlačení tlačidla STOP
   - sa okamžite zaradia odpory R1 a R3
   - po 5s sa zaradí odpor R2

Riešenie:
Programovateľný automat bude mať v tomto prípade dva vstupy pre tlačidlá a tri výstup pre spínanie jednotlivých stýkačov, ktoré skratujú príslušné odpory (obr. 8.6).



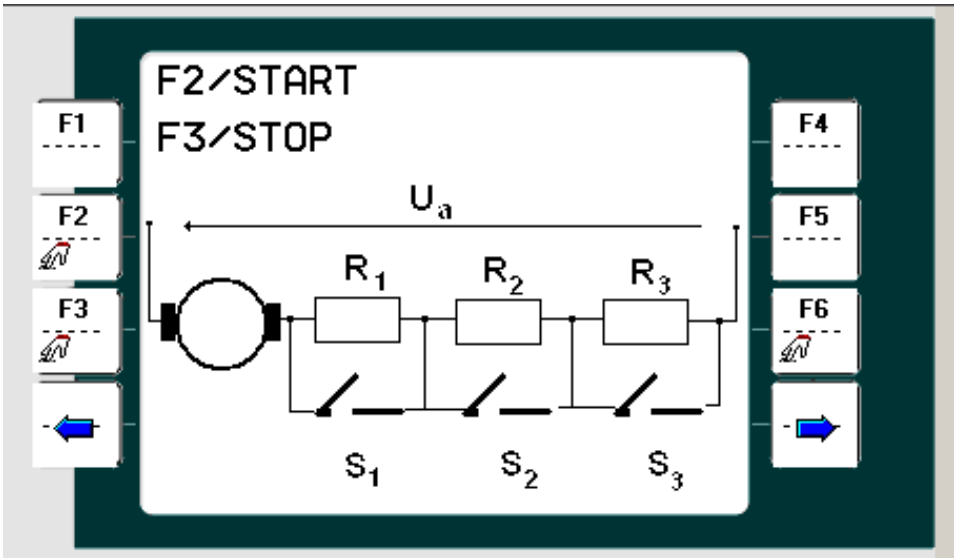
Obr. 10.6 Zobrazenie PA z hľadiska vstupov a výstupov

Príklad líniovej schémy, realizujúcej tento generátor, je uvedený na ďalšom obrázku (obr. 8.7).



Obr. 10.7 Riešenie úlohy formou líniovej schémy

Vzhľad vizualizačnej obrazovky pre typ automatu PP41 je uvedený na nasledujúcom obrázku (obr. 8.8). Stavy jednotlivých stýkačov je potrebné animovať!



Obr. 10.8 Vzhľad vizualizačnej obrazovky

10.3 Príklad 3 – Semafor pre chodca

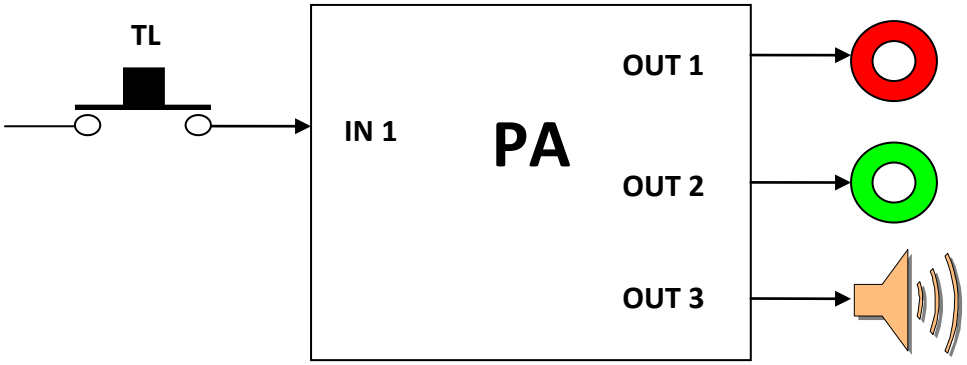
Všeobecný postup pri riešení úlohy pomocou PA:

1. **Slovná formulácia úlohy**

Naprogramujte ovládanie semafora pre chodca. Na semafore svieti štandardne červená. Keď chodec príde ku prechodu, stlačí tlačidlo. O 5s sa rozsvieti zelená, ktorá trvale svieti 5s a zároveň je spustený zvukový signál pre nevidiacich, potom 6s bliká v sekundovom intervale a nakoniec sa opäť rozsvieti červená.

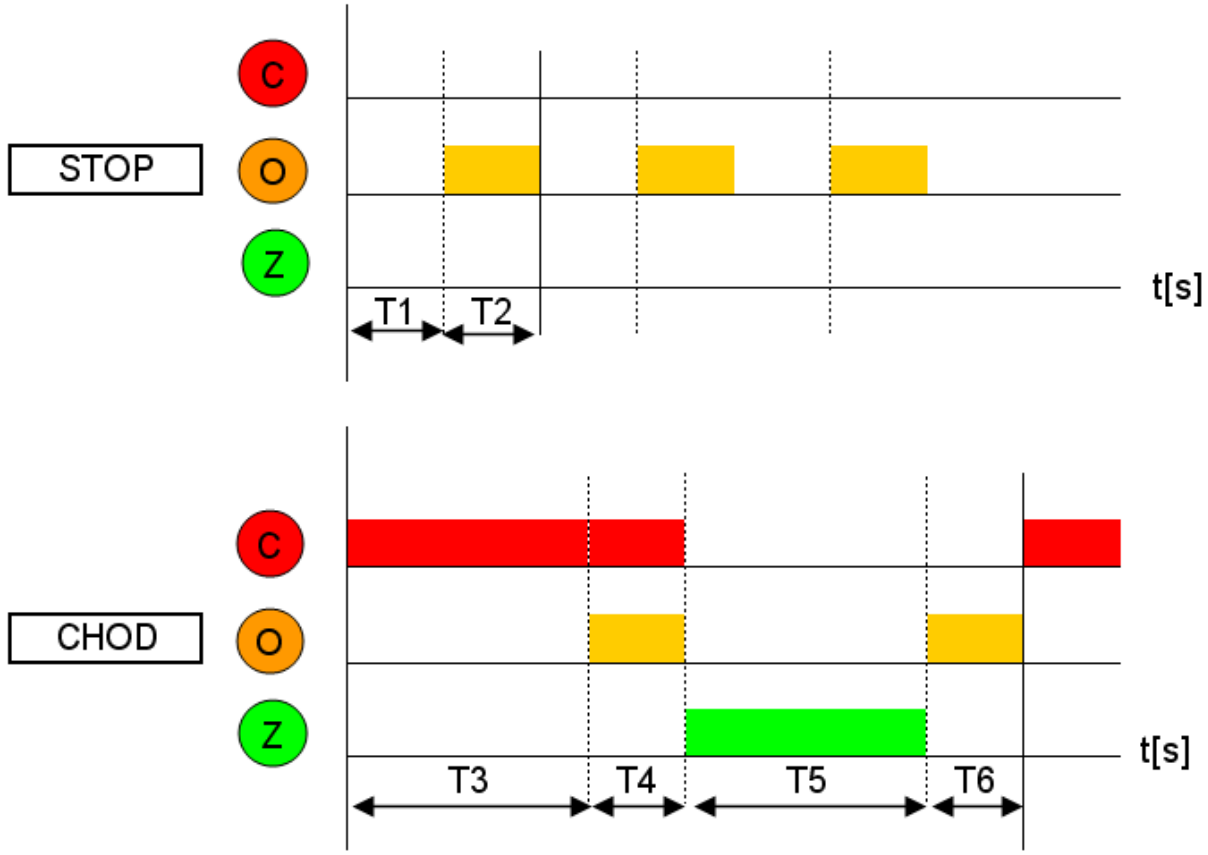
2. **Formalizovaný zápis úlohy** (Definícia I/O v súlade s technológiou, resp. projektom, definícia formálnych mien a typov použitých premenných, zakreslenie algoritmu vo zvolenej forme)

Je vhodné si na začiatku nakresliť blokovú predstavu úlohy, napr. (obr. 8.9)



Obr. 10.9 Bloková schéma riešenej úlohy

Časový priebeh jednotlivých signálov bude vyzerať (obr. 8.10):



Obr. 10.10 Časový priebeh jednotlivých signálov

Pre tento prípad potrebujeme napr. definovať tieto premenné (obr. 8.11):

▪ Stav tlačidla	TL	BOOLEAN	vstup
▪ Stav OUT1	CER	BOOLEAN	výstup
▪ Stav OUT2	ZEL	BOOLEAN	výstup
▪ Stav OUT3	HUK	BOOLEAN	výstup
▪ Časovač 5s pre CER	CAS_C	TON	int.pamäť
▪ Časovač 5s pre ZEL	CAS_Z	TON	int.pamäť
▪ Časovač 6s pre blikavú ZEL	CAS_BZ	TON	int.pamäť
▪ Časovače pre frekvenciu blik	CAS_1	TON	int.pamäť
	CAS_2	TON	int.pamäť
▪ Chod automatu	CHOD	BOOLEAN	int.pamäť

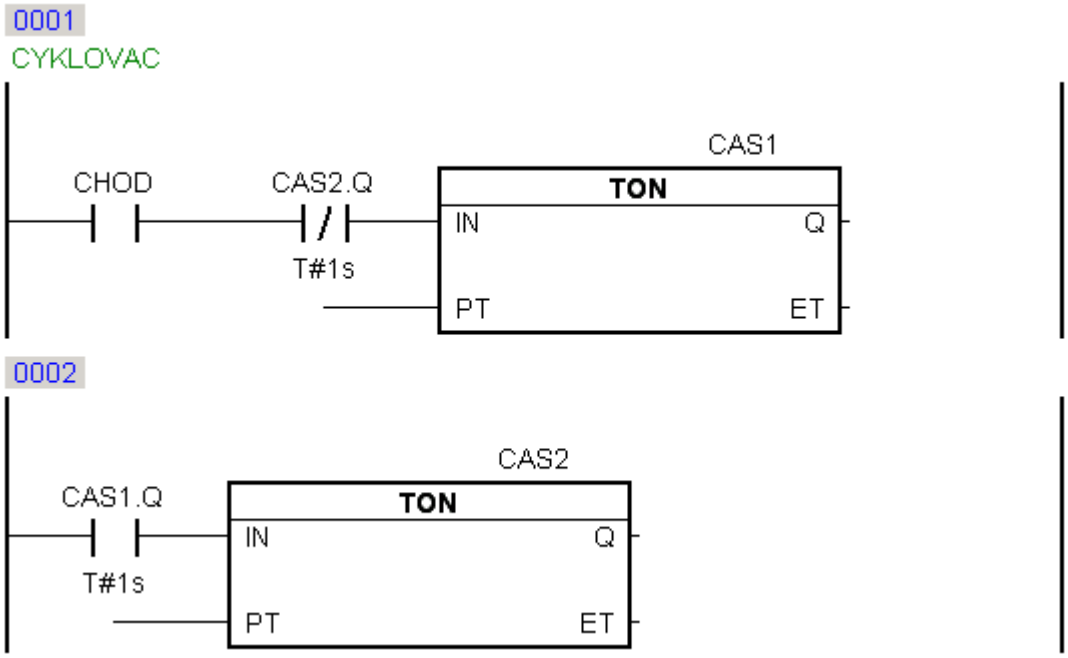
Name	Type	Scope	Attribute	Value	Owner	Remark
CAS1	TON	global	memory	* remanent		
CAS2	TON	global	memory	* remanent		
CAS_BZ	TON	global	memory	* remanent		
CAS_C	TON	global	memory	* remanent		
CAS_Z	TON	global	memory	* remanent		
CER	BOOL	global	QP3.02.01	* remanent		24 VDC, 0.4 A
CHOD	BOOL	global	memory	* remanent		
HUK	BOOL	global	QP3.02.03	* remanent		24 VDC, 0.4 A
TL	BOOL	global	memory	* remanent		
ZEL	BOOL	global	QP3.02.02	* remanent		24 VDC, 0.4 A

Obr. 10.11 Definovanie premenných v PA

Spresnený popis úlohy a z neho navrhnutá líniová schéma (obr. 8.12):

Blikač môžeme realizovať napr. takto:

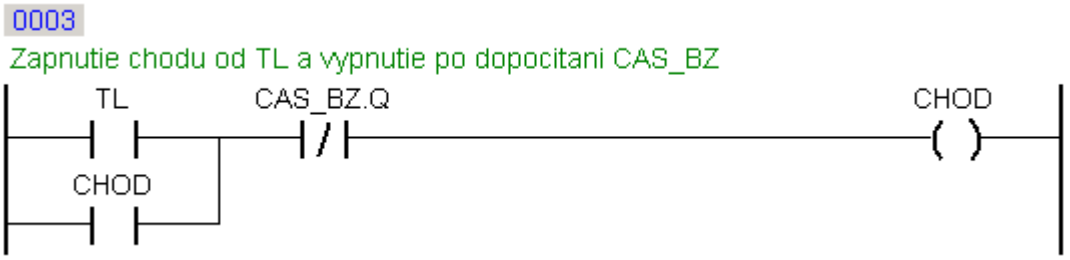
AK (je) CHOD A (nedopočítal) CAS_2 POTOM (počítaj) CAS_1
AK (dopočítal) CAS_1 POTOM (počítaj) CAS_2



Obr. 10.12 Realizácia blikania v PA formou líniovej schémy

Zapnutie a vypnutie chodu jedného cyklu semafora (obr. 8.13):

AK (niekto stlačí) TL POTOM (zapni, nastav) CHOD

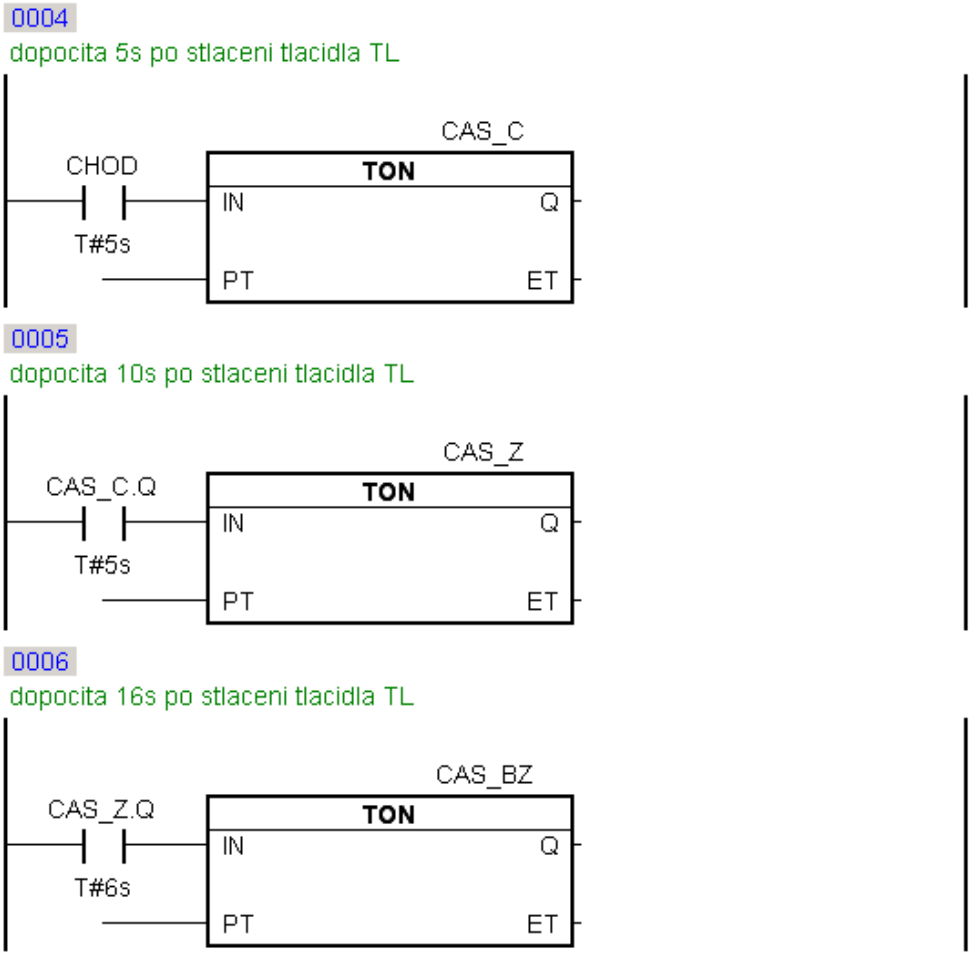


Obr. 10.13 Líniová schéma pre zapnutie a vypnutie jedného cyklu semafora

Meranie časov jednotlivých intervalov (obr. 8.14):

AK (niekto stlačí) CHOD
AK (dopočítal) CAS_C
AK (dopočítal) CAS_Z
AK (dopočítal) CAS_BZ

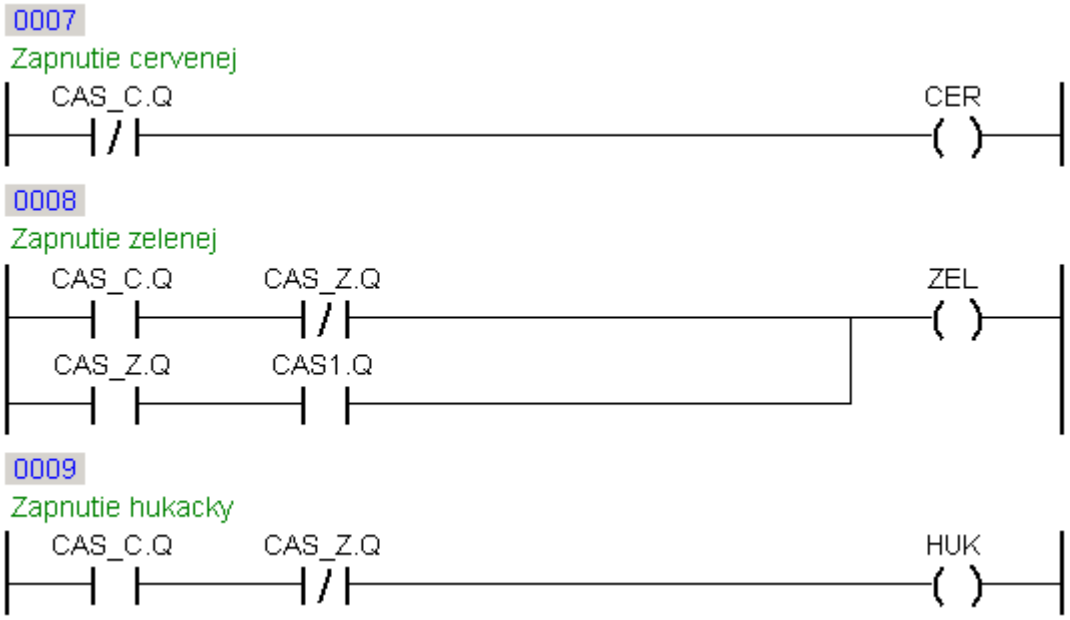
POTOM (počítaj čas 5s pre červenú) CAS_C
POTOM (počítaj čas 5s pre zelenú) CAS_Z
POTOM (počítaj čas 6s pre blikavú zelenú) CAS_BZ
POTOM (vypni) CHOD



Obr. 10.14 Línoivá schéma pre časovanie intervalov

Nastavovanie výstupov (obr. 8.15):

AK (nedopočítal) CAS_C — POTOM (zapni červenú) CER
AK (dopočítal) CAS_C A (nedopočítal) CAS_Z
ALEBO (dopočítal) CAS_Z A (blikač má 1) CAS_1 — POTOM (zapni) ZEL
AK (dopočítal) CAS_C A (nedopočítal) CAS_Z — POTOM (zapni) HUK



Obr. 10.15 Líniová schéma pre nastavovanie výstupov

3. **Naprogramovanie formalizovaného zápisu úlohy v PC pod prostredím pre daný PA a jeho prenos do automatu**

4. **Testovanie funkčnosti programu po stránke:**

   a. *Formálnej* - pomocou nástrojov v rámci prostredia PA

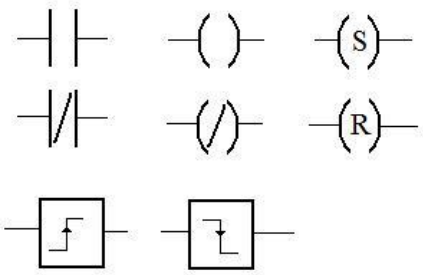
   b. *Logickej* - pomocou nástrojov v rámci prostredia PA

10.4 Príklad 4 – Kódovaný zámok

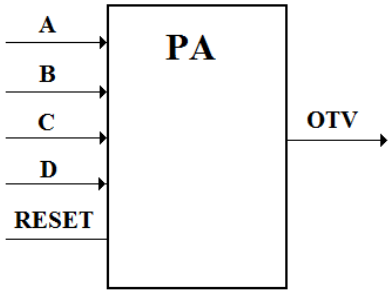
Pomocou PA a binárnych inštrukcií realizujte kódový zámok podľa nasledujúceho popisu:

- Zámok obsahuje 4 tlačidlá A, B, C, D.
- Zámok sa otvorí iba vtedy, ak stlačíme za sebou kombináciu kláves BADC.

Binárne inštrukcie:



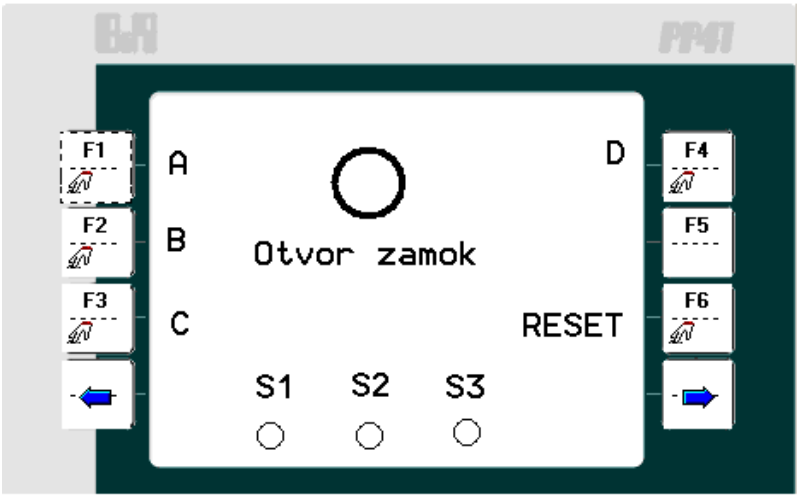
Riešenie:
Programovateľný automat bude mať v tomto prípade jeden 5 vstupov a jeden výstup (obr. 8.16).



Obr. 10.16 Zobrazenie riešenej úlohy z hľadiska vstupov a výstupov

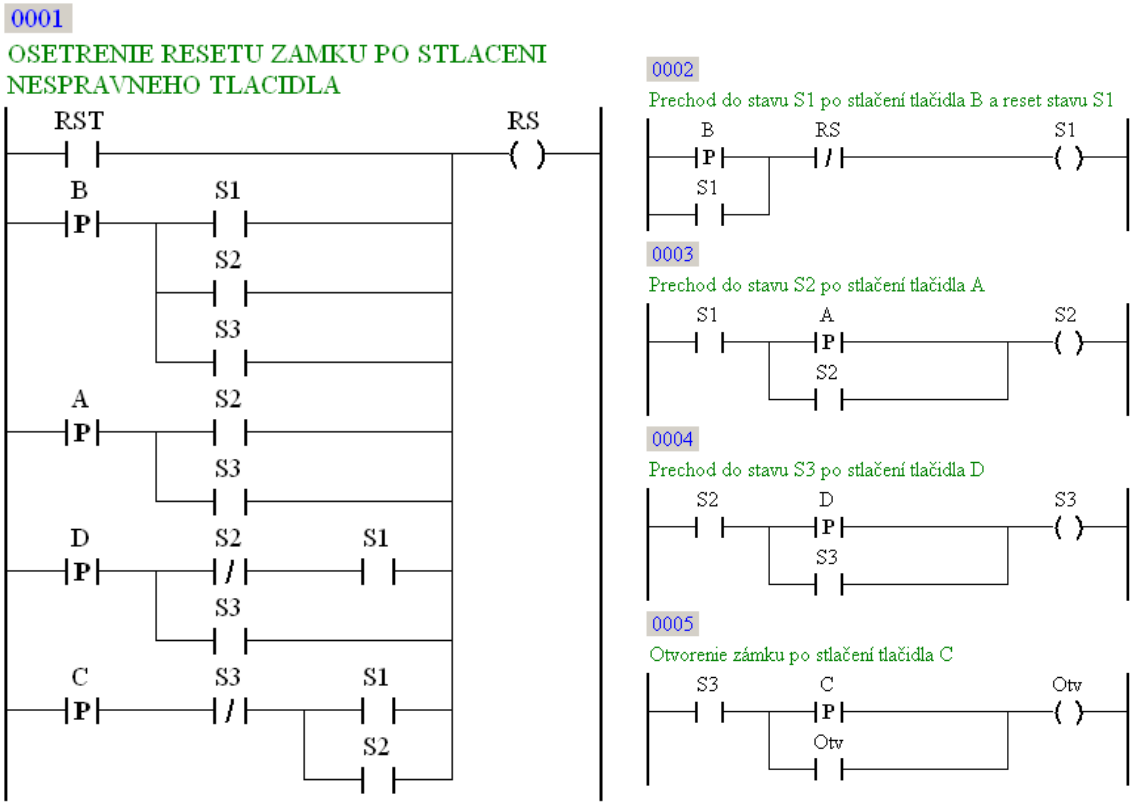
Riešenie pomocou PA B&R

- **Vzhľad vizualizačnej obrazovky** (obr. 8.17):



Obr. 10.17 Vizualizačná obrazovka pre PA B&R

- **Líniová schéma** (obr. 8.18):



Obr. 10.18 Líniová schéma v PA B&R

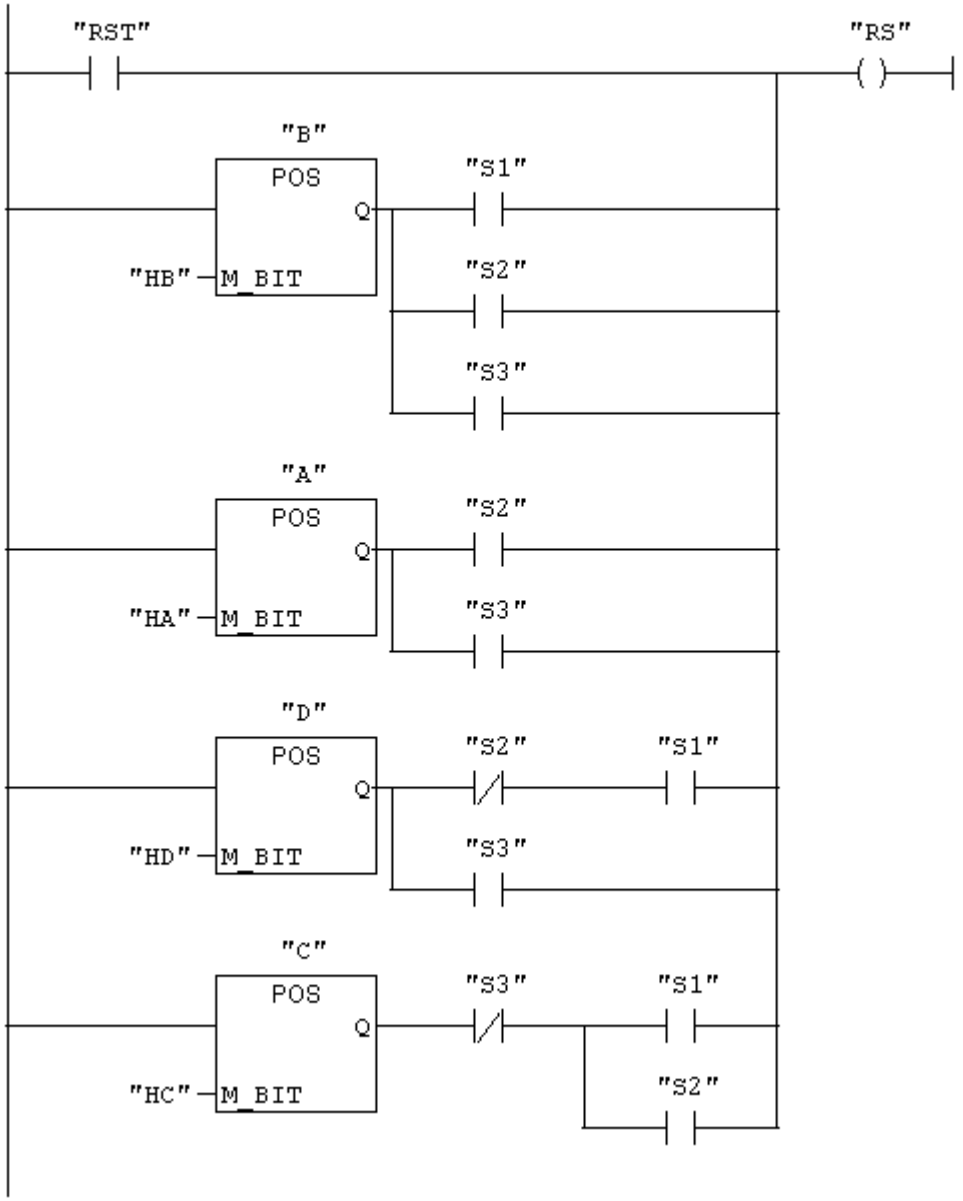
Riešenie pomocou PA Simatic S300

- **Vzhľad vizualizačnej obrazovky** (obr. 8.19):

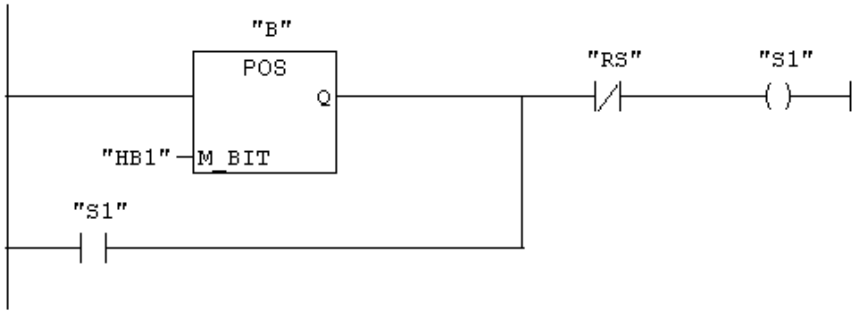
Obr. 10.19 Vizualizačná obrazovka pre PA Simatic S300

- **Líniová schéma pre PA Simatic S300** (obr. 8.20)

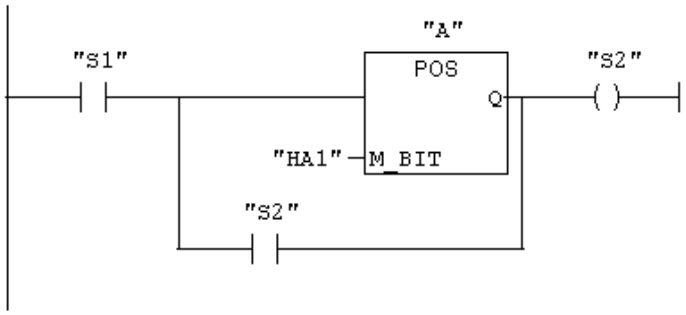
Network 1 : Ošetrenie resetu zamku postlačení nesprávneho tlačidla



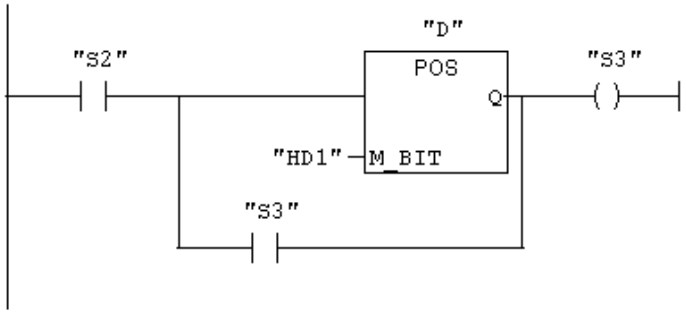
Network 2 : Prechod do stavu S1 po stlačení tlačidla B a reset stavu S1



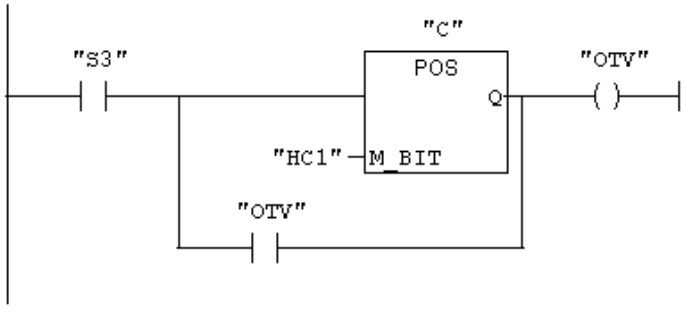
Network 3 : Prechod do stavu S2 po stlačení tlačidla A



Network 4 : Prechod do stavu S3 po stlačení tlačidla D



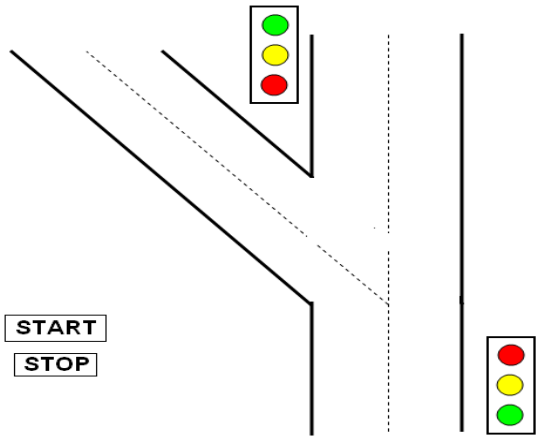
Network 5 : Otvorenie zámku po stlačení tlačidla C



Obr. 10.20 Líniová schéma riešenia úlohy pre PA Simatic S300

10.5 Zadanie 1 – Križovatka

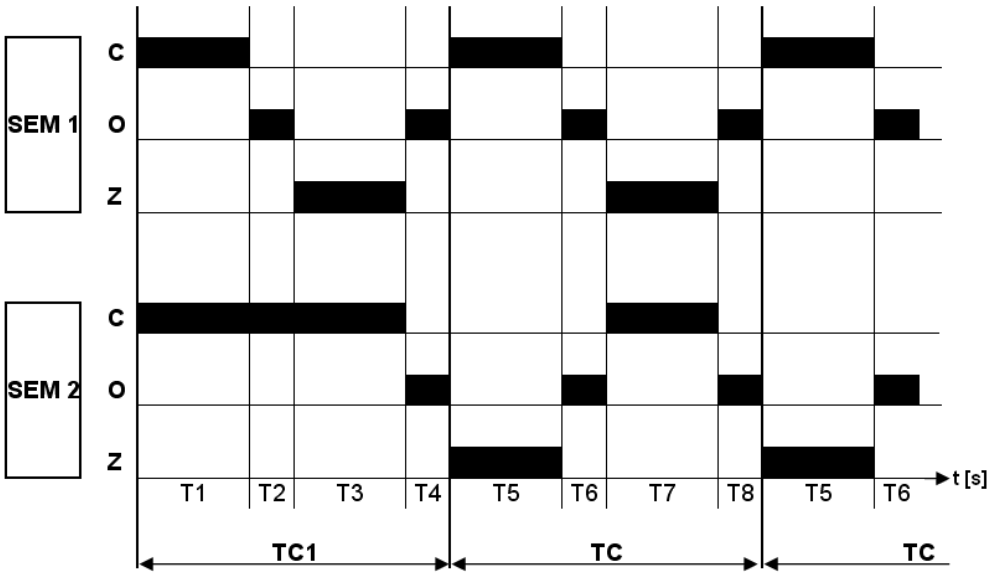
Pomocou PA realizujte riadenie križovatky s dvoma semaformi. Križovatku vizualizujte na operátorskom paneli (obr.8.21)



Obr. 10.21 Schéma križovatky

Križovatka pracuje v dvoch režimoch (obr. 8.22):

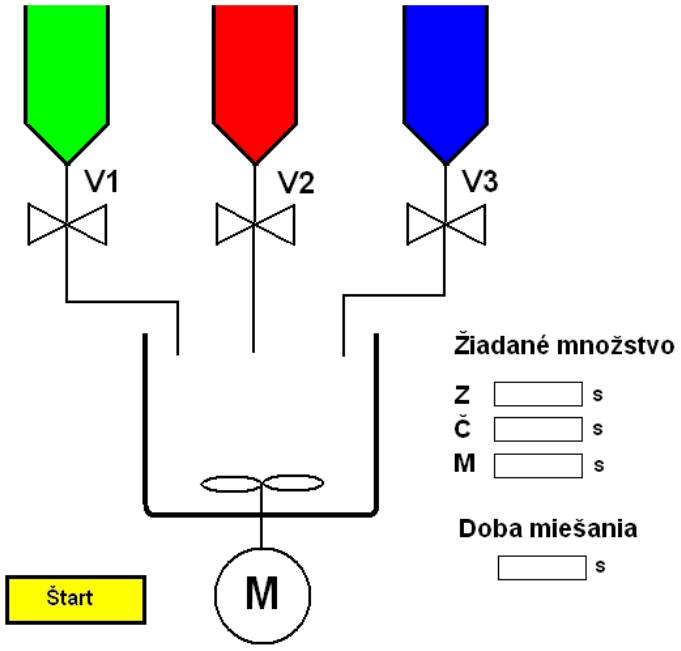
- STOP (po stlačení tlačidla **STOP**). V tomto režime na oboch semaforoch cyklicky bliká oranžová s periodou T=1 s.
- CHOD (po stlačení tlačidla **ŠTART)**. V tomto režime sa striedajú jednotlivé svetlá semaforov podľa nasledujúceho časového diagramu



Obr. 10.22 Časový priebeh signálov na semaforoch

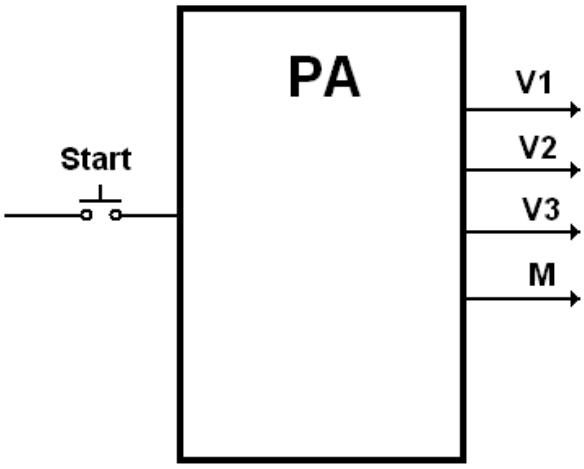
10.6 Zadanie 2 – Miešanie farieb

Pomocou PA realizujte automatické namiešanie vybranej farby podľa zadanej receptúry. Farba sa mieša na základe troch farieb: zelenej, červenej a modrej. Pre každú z uvedených farieb sa zadáva čas pre naplnenie zásobníka jej potrebným množstvom a celková doba miešania všetkých farieb (obr.8.23)



Obr. 10.23 Zobrazenie technológie miešania farieb

Programovateľný automat bude mať v tomto prípade jeden 1 vstup a štyri výstupy (obr. 8.24).



Obr. 10.24 Zobrazenie PA z hľadiska vstupov a výstupov

Záver

Predložená učebnica sa zaoberá základnou filozofiou malých programovateľných automatov PA), ich hlavnými technickými prostriedkami a možnosťami ich programovania. Vzhľadom na veľký počet výrobcov týchto automatov v súčasnosti nie je možné dosť dobre popísať všetky detaily jednotlivých typov a jednotlivých programovacích prostredí. Preto v tejto učebnici bola snaha popísať všeobecne platné vlastnosti, použiteľné pri čo najširšom spektre dostupných programovateľných automatov. Rovnako si treba uvedomiť, že vývoj v tejto oblasti napreduje prudkým tempom, čo prináša stále nové dokonalejšie typy programovateľných automatov, tak z hľadiska výkonu, ako aj z hľadiska nových funkcií a možností.

Z inžinierskeho pohľadu potom nie je až tak dôležité poznať detaily jednotlivých typov automatov, ale skôr vedieť formulovať algoritmy ich riadenia vo forme vhodnej pre ich programovanie, napríklad pomocou líniových schém a sekvenčných diagramov.

Použité príklady treba chápať ako možné riešenie daných (veľmi zjednodušených) úloh. Skutočné praktické nasadenie konkrétneho programovateľného automatu bude určite vyžadovať od programátora viac praktických skúseností.

Literatúra

[1] NIŽNÍK J.: Návrh technologickej časti riadiaceho systému zásobníka sypkých materiálov. Košice : FEI TU, 1998. Diplomová práca.

[2] KOHAN D.: Riadenie zásobníka sypkých materiálov pomocou logického automatu Schneider Modicon TSX a odpovedajúceho vizualizačného systému. Košice : FEI TU, 2001. Diplomová práca.

[3] FEDOR P. – PERDUKOVÁ D.: Programovateľné automaty v elektrických pohonoch. Košice : Mercury – Smékal, 2001. ISBN 80 - 968550 - 0 - X

[4] MUDRONČÍK D. - ZOLOTOVÁ I.: Priemyselné programovateľné regulátory. Košice : Elfa, s.r.o., 2000. ISBN 80 - 88964 - 45 - 8

[5] ŠMEJKAL, L.: Tři desetiletí PLC a standard IEC 1131-3, Automatizace č. 12, 1998

[6] URBAN, L.: Programování podle normy IEC 1131-3, Automatizace č. 10, 1998

[7] ANTSAKLIS, P. J.: Defining intelligent control. IEEE Control Syst. Mag., vol. 14, pp. 4–66, 1994

[8] BIEN, Z.: How to measure the machine intelligence quotient (MIQ):Two methods and applications. IEEE Trans. Syst., Man, Cybern. A, vol. 27, pp. 256–262

[9] BRYCHTA, J. – Šmejkal, L.: Výběr vizualizačního systému pro dispečerská pracoviště. Automatizace 39, č.2, 1996, s.47-52. ISSN 0005-125X

[10] DIX, A. et. al.: Human-Computer interaction. Pearson Education 2003.

[11] FOULLOY, L. – ZAVIDOVIQUE, B.: Toward symbolic process control. Automatica, vol. 30, no. 3, pp. 379–390, 1994.

[12] KIM, S.W. – KIM, B. K.: MIQ (Machine Intelligence Quotient) for process control system. IEEE Trans. Syst., Man, Cybern. , vol. 28, pp. 325–332

[13] LANDRYOVÁ, L. – ZOLOTOVÁ, I.: Integrated Environment of SCADA/HMI for Process Simulation. In: INES´99, 3rd IEEE International Conference on Intelligent Engineering Systems 1999, High Tatras, 1999, 469-472

[14] NEWMAN, W. M., Lamming, M G: Interactive System Design. Addison-Wesley 1995.

[15] PARK, H. J. et al.: Measuring the Machine Intelligence Quotient (MIQ) of Human-Machine Cooperative Systems. IEEE Trans. On systems, Man and Cybernetics, part A, vol. 31, No.2, 2001. pp. 89-96

[16] PARR, E. A.: Industrial Control Handbook, Industrial Press Inc., 1999. ISBN 0-8311-3085-7

[17] LAUGHTON, M. A. - WARNE D. J. (ed): Electrical Engineer's Reference book, 16th edition, Newnes, 2003 Chapter 16, Programmable Controller.

[18] HARMS, T. M. - KINNER, R. H. P.E.: Enhancing PLC Performance with Vision Systems. 18th Annual ESD/HMI International Programmable Controllers Conference Proceedings, 1989, p. 387-399.

[19] MAHER, M. J.: Real-Time Control and Communications. 18th Annual ESD/SMI International Programmable Controllers Conference Proceedings, 1989, p. 431-436.

[20] KINNER, P.E.: Designing Programmable Controller Application Programs Using More than One Designer. 14th Annual International Programmable Controllers Conference Proceedings, 1985, p. 97-110.

[21] BOLTON, W.: Programmable Logic Controllers. Fifth Edition, Newnes, 2009, Chapter 1. ISBN 978-1-85617-751-1,

[22] KELLER, W. L. Jr.: Grafcet, A Functional Chart for Sequential Processes. 14th Annual International Programmable Controllers Conference Proceedings, 1984, p. 71-96.

[23] GREGORY K. M. - DOUGLAS M. C. (ed): Process/Industrial Instruments and Controls. Handbook Fifth Edition, McGraw-Hill, 1999, Section 3: Controllers. ISBN 0-07-012582-1

[24] ERICKSON, K. T.: Programmable logic controllers. Institute of Electrical and Electronics Engineers. 1996

[25] IQBAL, S.: Programmable Logic Controllers (PLCs): Workhorse of Industrial Automation. IEEEP Journal, 2008, pp. 27–31.

[26] PETRUZELLA,, F. D.: Programmable logic controllers. Tata McGraw-Hill Education. 2005.

ZDROJE WWW

1. http://www.sigchi.org/
2. http://www.useit.com/
3. http://www.w3.org/TR/WAI-WEBCONTENT/
4. http://www.baddesigns.com/
5. http://www.iarchitect.com/mshame.htm
6. http://en.wikipedia.org/wiki/Programmable_logic_controller

7. http://de.wikipedia.org/wiki/Speicherprogrammierbare_Steuerung
8. http://www.technolead.com/articles/?ID=15&SubID=231
9. http://www.plcdev.com/schneider_electric_modicon_history
10. http://www.ceasiamag.com/article-4616-specialreportcontrolsystems-Asia.html
11. http://en.wikipedia.org/wiki/IEC_61131-3
12. http://www.iec.ch/
13. http://www.plcopen.org/