Příprava podkladů pro kurz DVPP se zaměřením na programovatelné automaty

Filip Šimo

Bakalářská práce
2018

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2017/2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Filip Šimo**
Osobní číslo:	**A15073**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Příprava podkladů pro kurz DVPP se zaměřením na programovatelné automaty**
Téma anglicky:	**The Preparation of Materials for the DVPP Course with a Focus on Programmable Controllers**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na téma programovatelné automaty (PLC) a způsoby výuky jejich programování.**
2. **Vytvořte a utřiďte materiály specifické pro dvacetihodinový kurz určený pro pedagogické pracovníky v rámci dalšího vzdělávání pedagogických pracovníků (DVPP) využitelné v následujících bodech zadání.**
3. **Navrhněte vzorové úlohy zaměřené na různé oblasti pokročilého programování PLC.**
4. **Pro navržené úlohy vytvořte podrobné návody.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. PLC a automatizace, 1.díl. 1. Vyd. Praha: BEN technická literatura, 1999. ISBN 80-860-5658-9.**
2. **ŠMEJKAL, Ladislav. PLC a automatizace, 2.díl. 1. Vyd. Praha: BEN technická literatura, 2005. ISBN 80-7300-087-3.**
3. **MARTINÁSKOVÁ, Marie a Ladislav ŠMEJKAL. Řízení programovatelnými automaty. Vyd. 2. Praha: Vydavatelství ČVUT, 2004. ISBN 80-010-2925-5.**
4. **TECO [online]. [cit. 2017-11-23]. Dostupné z: http://www.tecomat.eu/index.php?ID=655**
5. **FOXON [online]. [cit. 2017-11-23]. Dostupné z: https://www.foxon.cz/blogs/category/17-kurzy-programovani-simatic-s7-300.html**

Vedoucí bakalářské práce: **Ing. Tomáš Sysala, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **15. prosince 2017**

Termín odevzdání bakalářské práce: **25. května 2018**

Ve Zlíně dne 15. prosince 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 21. 5. 2018

.....
podpis diplomanta

ABSTRAKT

Bakalárska práca je zameraná na vytvorenie učebného plánu pre rozšírenie vzdelania pedagogických pracovníkov v okruhoch programovania logických automatov (PLC).
Teoretická časť práce obsahuje všeobecný popis PLC, jeho históriou, využitie a jeho jednotlivé funkcie. V ďalšej časti práce sú popísané rôzne programovacie jazyky - textové (jazyk mnemokódu, štruktúrovaný text) a grafické (jazyk reléových schém, jazyk logických schém a Grafcet). Predposledná časť teoretickej časti je zameraná na PLC od výrobcu Tecomat, kde je popísaná štruktúra pamäte, zásobník, systém procesov, čitače, časovače a tabuľkové inštrukcie. Záver teoretickej časti práce je venovaný popisu programu Control Web.
V praktickej časti práce sú vytvorené programy v prostredí Mosaic, zamerané na dva programovacie jazyky, a to jazyk mnemokódu a štruktúrovaného textu. Úlohy sú rozdelené na kombinačné logické, štruktúrované logické, čítače, časovače, tabuľkové inštrukcie a v zložitejších úlohách je vysvetlený princíp zasielania e-mailov z PLC.
V závere práce sú zhrnuté výsledky podrobného popisu programov v prostredí Mosaic z priložených skrípt bakalárskej práce.

Kľúčové slova: PLC, jazyk mnemokódu, štruktúrovany text, reléové schémy, logické schémy

ABSTRACT

The bachelor thesis is aimed at creating a curriculum for the extension of pedagogical education in PLC.
The theoretical part of the thesis contains a general description of PLC, its history, usage and its individual functions. Different programming languages are described in the next part - text (mnemocode language, structured text) and graphical (language of relay schemes, language of logical schemes and Graftec). The penultimate of the theoretical part is focused on the Tecomat PLC, which describes memory structure, stack, process system, readers, timers and table instructions. The conclusion of the theoretical part is dedicated to the description of Control Web.
In the practical part of the thesis programs are created in the Mosaic environment, focusing on two programming languages, namely the language of the mnemocode and structured text.

Tasks are divided into combinational logic, structured logic, counters, timers, table instructions, and in more complex tasks the principle of sending emails from the PLC is explained.
At the end of the thesis are summarized the results of detailed description of programs in the Mosaic environment from the attached bachelor thesis script.

Keywords: PLC, mnemocode language, structured text, relay schemes, logic schemes

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Bakalárska práca sa zaoberá tvorbou podkladov pre intenzívny týždňový kurz zameraný na programovateľné automaty (ďalej len PLC). Hlavnou úlohou je vytvorenie špecifického učebného textu z dostupnej literatúry a vzorových úloh, podľa daných požiadaviek náplne kurzu. Práca je rozdelená na teoretickú a praktickú časť.

V teoretickej časti práce je zhrnutá história PLC, všeobecný popis, funkčnosť PLC, popis otočky cyklu, doby cyklu, spracovanie prerušení a ich jednotlivé typy. Ďalej sú popísané rôzne spôsoby programovania PLC, textové a grafické programovanie. Následne je spracovaný popis fungovania programu PLC od výrobcu Tecomat, jeho štruktúra, registre, štruktúra pamäte a zásobník. V ďalšej časti práce je popísaný proces fungovania čítačov, časovačov, tabuľkových inštrukcií a ich podrobné zamerania. V závere teoretickej časti práce je popísaný program Control Web.

V praktickej časti práce sú uvedené jednotlivé zadania úloh a popis ich riešení dvoma spôsobmi, pričom prvý spôsob je uvedený v textovom jazyku mnemokódu a druhý spôsob v jazyku štruktúrovaného textu. Úlohy sú postupne riešené od tých najjednoduchších až po zložitejšie. Zadané úlohy sa začínajú kombinačnými logickými úlohami, pokračujú sekvenčnými logickými úlohami a v zložitejších úlohách sú aplikované časovače, čítače a na záver aj tabuľkové inštrukcie.

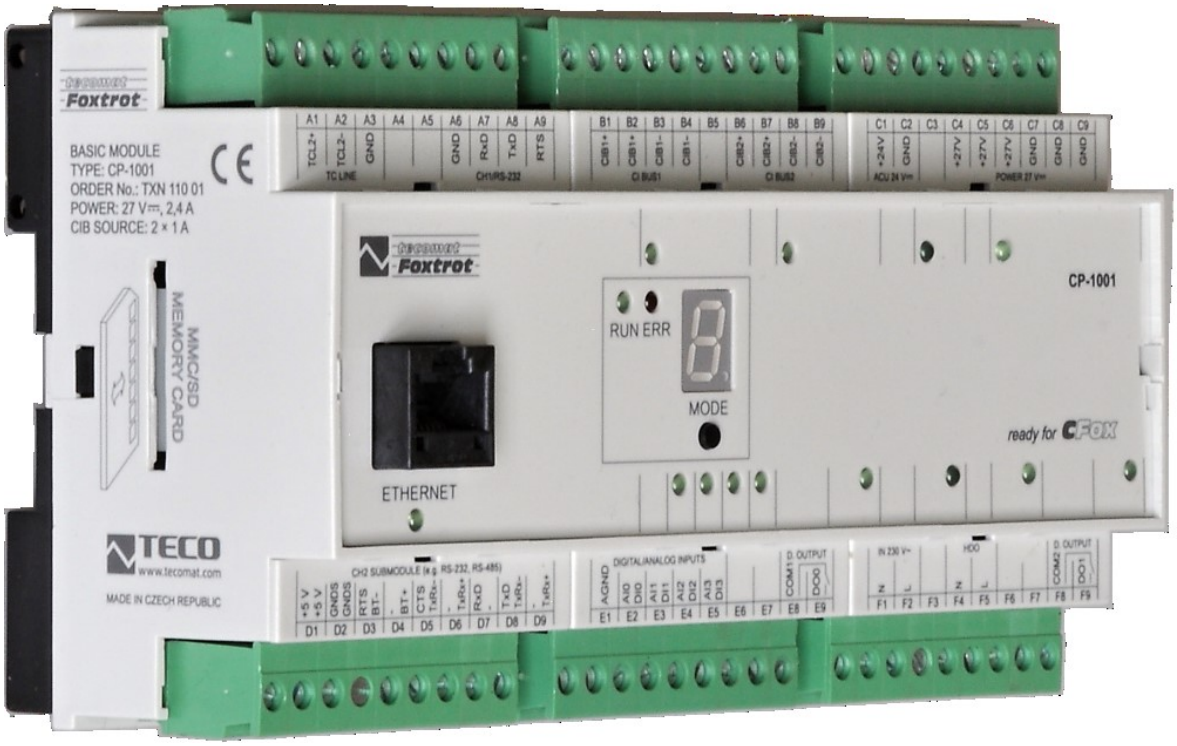
K úlohám je vytvorený študijný materiál, v ktorom je dôkladne popísaná práca v prostredí Mosaic a popis jednotlivých riešení daných úloh.

Študijný materiál je určený pre 20-hodinový kurz pedagogických pracovníkov v rámci ďalšieho vzdelávania.

I. TEORETICKÁ ČASŤ

1 PLC

Programovateľný logický automat alebo PLC (z anglického slova Programmable Logic Controller) je relatívne malý priemyslový počítač používaný pre automatizáciu procesov v reálnom čase – predovšetkým na riadenie strojov, výrobných liniek, semafory a ďalšie rôzne automaty. PLC je nahradzovaný výrazom PAC (z anglického Programmable Automation Controller), ale naďalej sa označuje PLC, čo je celosvetovo rozšírené [7].



Obr. 1. Vzor PLC Tecomat [12]

1.1 História

Číslicový počítač, ako univerzálne programovateľné zariadenie, bolo ihneď po svojom vzniku použité k riadeniu technologických procesov. Programovateľné logické automaty, boli schopné reálne spracovávať odchýlky medzi žiadanou a skutočnou hodnotou. Tieto automaty sa začaly objavovať okolo roku 1930. Skutočnej automatizácie dosiahla chemická spoločnosť Texaco v Port Arthuru s počítačovým riadením až v roku 1959. Ďalšiemu a väčšiemu rozšíreniu automatizácie doteraz bránila vysoká cena a relatívne veľké nároky na inštalačný priestor. K prelomu došlo až okolo roku 1970 s poklesom ceny a miniaturizáciou samotných prvkov PLC. Cena a miniaturizácia je vždy primeraná k danému obdobiu [7].

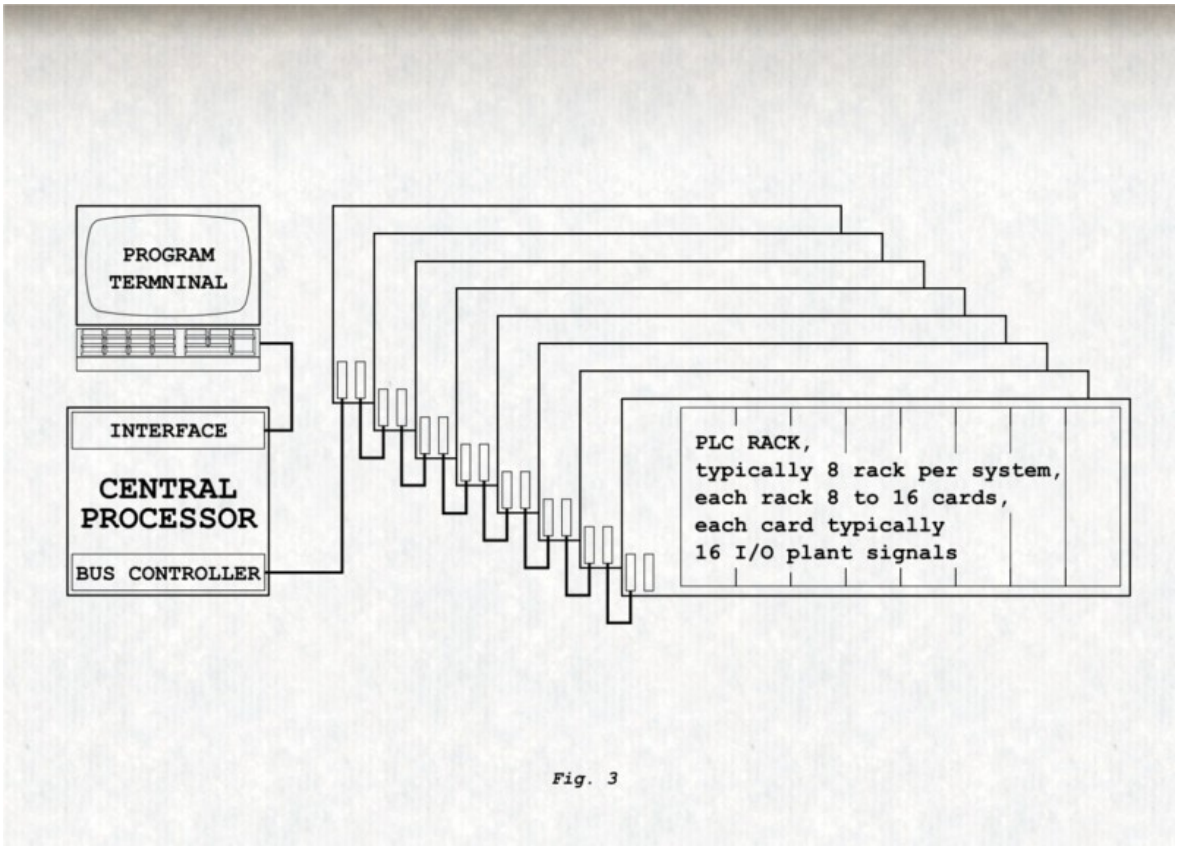
Prvé PLC pre svoj stabilný prevádzkový stav vyžadovali prostredie podobné sálovým počítačom, to znamená klimatizáciu, čisté prostredie, stabilnú dodávku elektrickej energie a

odrušenie od vonkajších vplyvov. K programovaniu bolo nutné mať školených špecialistov ktorí veľmi dobre poznali jadro PLC. Pre praktické využitie PLC však bolo nutné dosiahnuť podmienok takmer opačných [7].

PLC musí byť odolné proti vplyvom prostredia, jeho nasadenie by malo byť možné všade tam, kde je potreba. U PLC je vyžadované, aby ho bolo možno ľahko modifikovať podľa požadovaného počtu vstupov a výstupov. Základnou požiadavkou je spracovanie binárnych signálov. Používanie PLC nevyžaduje niekoľkoročné zaškolenie operátora. Reakčná doba na zmenu stavu procesu musí byť dostatočne rýchla tak, aby riadenie procesov bolo udržované v požadovateľnom stave a to podľa povahy procesu [7].

1.1.1 Prvé PLC

Po roku 1960 sa americký výrobca automobilov General Motors začal venovať myšlienke náhrady reléových riadiacich systémov s pevnou logickou štruktúrou a s počítačovými systémami schopnými pružnejšie reagovať na potrebné zmeny výroby, teda pružný automatizačný systém. V roku 1968 vyhlásila spoločnosť General Motors súťaž na dodávku počítačového riadenia pre svoje výrobné závody. Súťaž bola vyhlásená na základe referátu, ktorý predniesol Bill Stone na konferencii General Motors Corporation vo Westinghouse, kde sa riešili výrobné záležitosti a vznikol návrh na vytvorenie štandardu riadenia strojov. Do súťaže sa prihlásili štyri spoločnosti: Information Instruments, Inc (teraz Rockwell Automation), Digital Equipment Corp. (DEC), Century Detroit a Bedford Associates (neskôr Modicon). Spoločnosti navrhli (pravdepodobne podobné) riadiace systémy čiastočne podobné minipočítačom tej doby. Víťazom súťaže se stala spoločnosť Bedford Associates a v roku 1969 bol vyrobený prvý PLC [7].



Obr. 2. Prvé PLC [7]

Jadrom systému bol mikroprocesor (CPU) prispôsobený na prevádzku v priemyselnom prostredí a s okolitým prostredím bol prepojený cez rámy (racky), do ktorých sa inštalovali vstupné / výstupné moduly, každý až o 16 binárnych signáloch. Celková konfigurácia umožňovala k CPU pripojiť až 8 rámov, každý s 8 pozíciami pre moduly vstupu / výstupu po 16 signálov, teda 8 x 8 x 16 = 1024 signálov. Dôležitým rysom bolo to, že modul vstupu alebo výstupu bolo možné osadiť do ľubovolnej pozície ľubovolného rámu, podľa požiadaviek užívateľa. Prvé PLC dostalo označenie 084, ako 84 projekt spoločnosti Bedford Associates, ktoré sa následne začalo zaoberať vývojom, výrobou a servisom pre tento produkt, ktorý bol uvedený na trh pod názvom MODICON (skratka zo slova MO-dular DI-gital CON-troller) ako modulárný systém pre riadenie procesov. Číslica 84 na konci označenia sa ďalej používala ako synonymum pre PLC MODICON až do označenia 984. Jeden z prvých PLC 084, ktorý bol v prevádzke 20 rokov, je vystavený v sídle Modiconu, North Andover, Massachusetts. Jedným z vývojárov PLC bol Richard E. Morley a Dick Morley, ktorý je považovaný za "otca" PLC [7].

1.2 Čo je PLC

PLC je dnes relatívne malý počítač (v závislosti na tom, ako veľký objekt ním riadime) v priemyslovom prevedení, riadený mikroprocesorom s vlastným operačným systémom, prispôsobeným pre potreby riešenia automatizačných úloh v reálnom čase, s čo najkratšou dobou odozvy [7].

Pre komunikáciu s okolím je PLC vybavený vstupnými perifériami (vstupy), na ktoré je privedený signál z riadeného procesu, binárny signál v stave zapnuté/vypnuté (napr. snímanie polohy koncovým snímačom) alebo v podobe spojitých analógových signálov (napr. teplota, tlak, hladina ...). Na "opačnej" strane má PLC výstupné periférie (výstupy), ku ktorým sú pripojené akčné prvky riadeného procesu, opäť v podobe binárneho riadiaceho signálu zapnuté/vypnuté (napr. stykač motora, cievka ventilu...) alebo spojitého výstupného riadiaceho signálu analógovej veličiny (napr. pre riadenie rýchlosti, polohy regulačného ventilu...) [7].

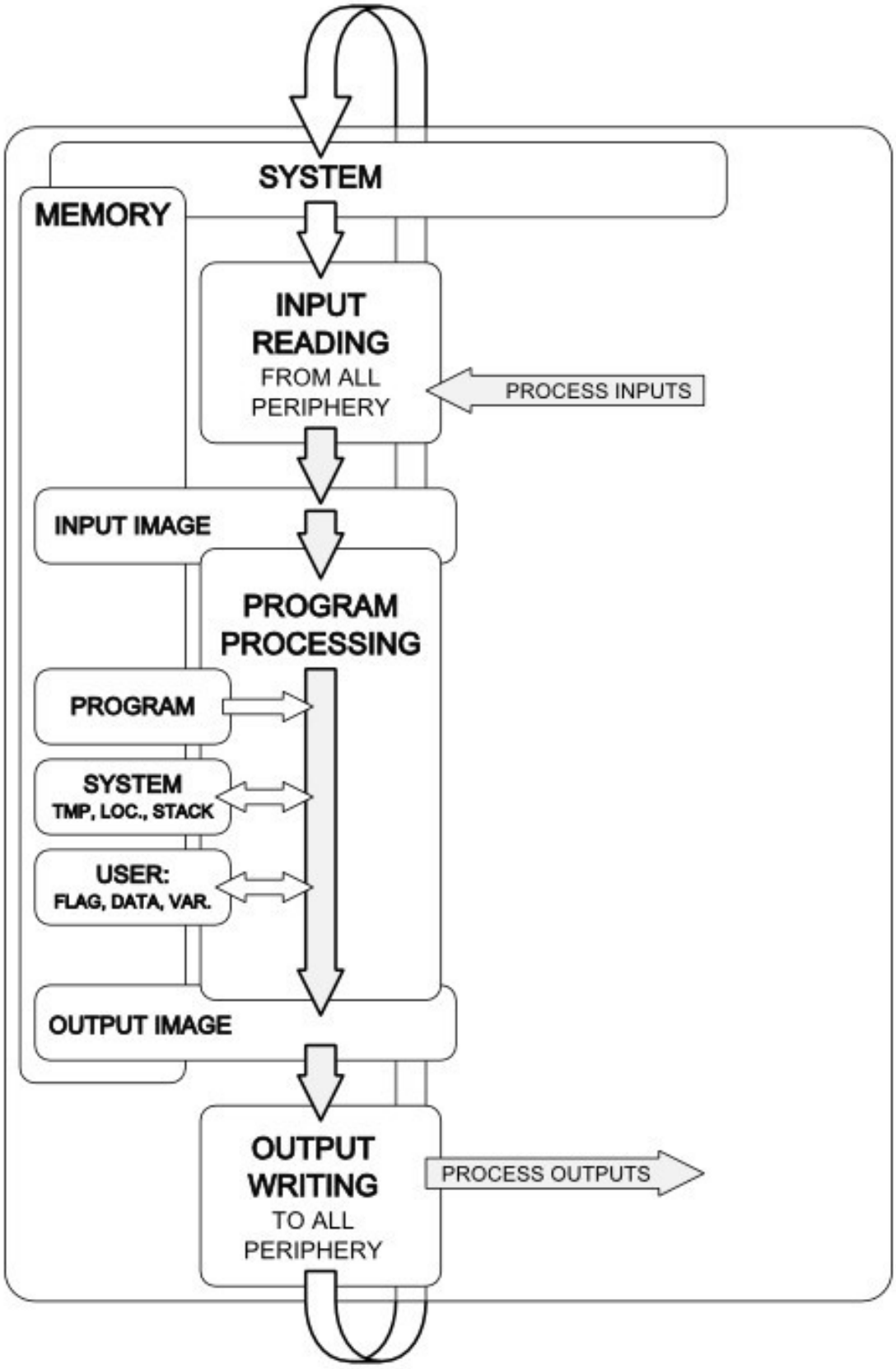
Medzi vstupom a výstupom sa "nachádza" riadiaca logika – CPU, ktorá na základe stavu vstupu ovláda výstup tak, aby bola dosiahnutá minimálna odchýlka od žiadaného alebo zadaného stavu celého zariadenia. Ako bude PLC reagovať na zmenu stavu vstupných signálov, určuje programátor tým, že vytvorí programový algoritmus - riešenie zadanej úlohy (skrátene program) a ten uloží do pamäti PLC. Operačný systém PLC tak zabezpečí, aby bol program opakovane (cyklicky) vykonávaný [7].

Mimo klasických periferií (binárnych, analógových) je PLC vybavený rozhraním (interface) pre komunikáciu s programátorom, rovnakým alebo ďalším rozhraním pre komunikáciu s obsluhou, ak je to treba. Ďalšou možnosťou je zapojenie PLC do siete, kde môže komunikovať s dalšími PLC perifériami, obecnými systémami v hierarchických sieťach [7].

1.3 Otočka cyklu

Cyklus PLC je základným rysom spracovania riadenia, algoritmu, kedy sú jednotlivé príkazy vykonávané v opakovanom cykle [7].

Cyklus je základným znakom priemyslového riadiaceho systému, kedy PLC stále opakovane - **cyklicky** vykonáva svoju činnosť, viď obrázok 1 [1, 7].



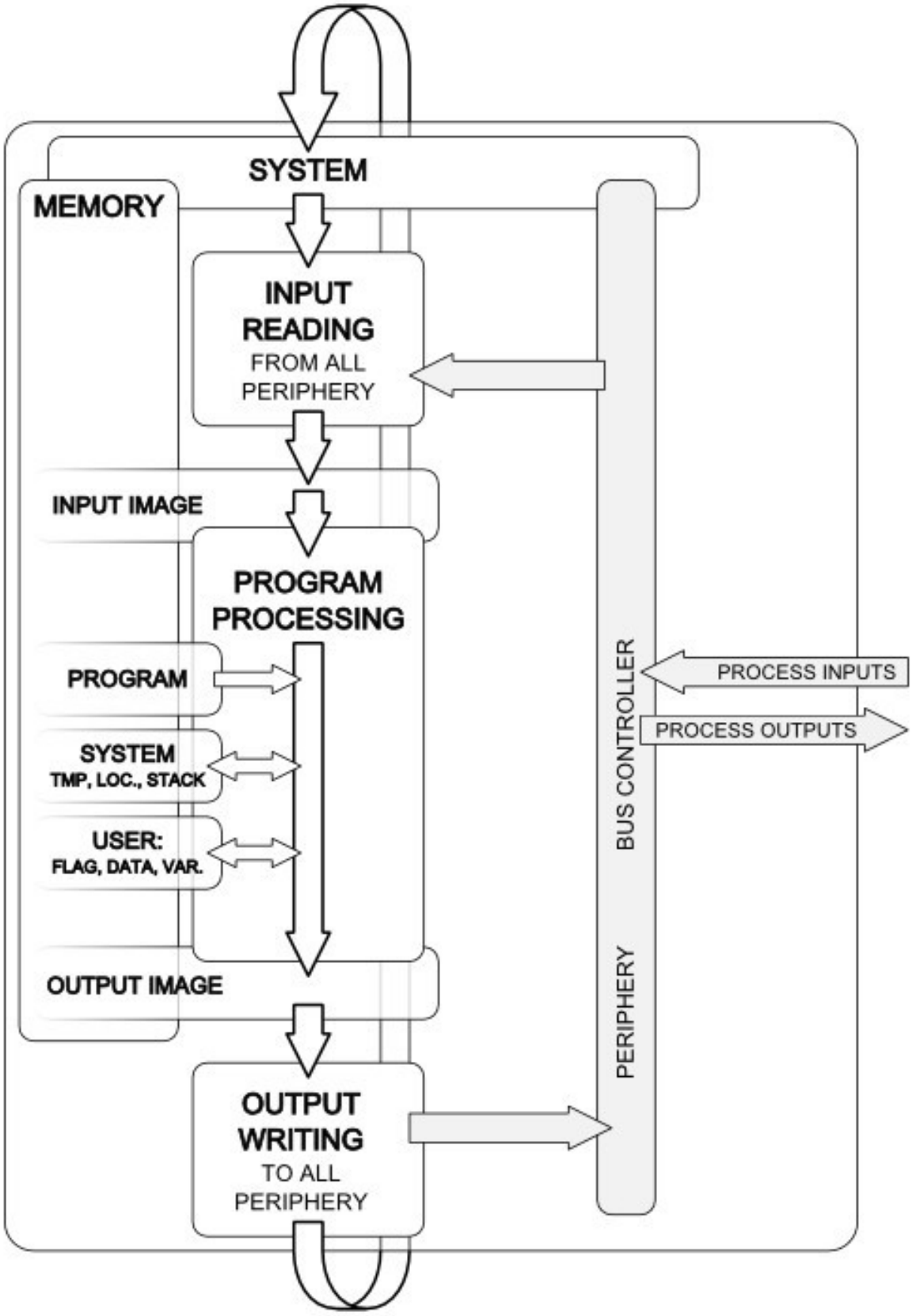
Obr. 3. Základný cyklus malého PLC [7]

Úvodným krokom na začiatku každého cyklu je vykonanie systémových operácií v CPU, ktoré užívateľ nemá možnosť ovplyvniť. Jedná sa o vnútornú kontrolu, komunikáciu s programátorom, manipuláciu s programovými blokmi [7, 1].

Nasledujúcim krokom je synchrónne načítanie stavu všetkých dostupných vstupných periférnych signálov a uloženie tohoto stavu do pamäti prerušení, ktorý je označovaný ako **obraz procesných vstupov (PII - Process Input Image)**. Pokiaľ je však v programe odkaz na nejaký vstupný signál, je jeho hodnota dosadená (prevzatá) práve z tohto obrazu vstupov. Toto ukladanie stavu vstupov sa zdá zbytočnou komplikáciou, ale dôvodom pre toto riešenie je to, aby počas jedného cyklu programu bola zaistená jednoznačnosť stavu vstupného signálu, hlavne u digitálnych vstupoch. V prípade, že by sa program v priebehu cyklu zakaždým spytoval na práve aktuálny stav vstupu, mohlo by dôjsť k situácii, že na začiatku programu by bola hodnota konkrétneho vstupu napr. "zapnuté", v strede programu "vypnuté" a na konci programu zasa "zapnuté". Spracovanie niekoľkých logických podmienok počas jedného cyklu programu s rôznym stavom toho istého signálu by logicky viedlo k nejednoznačným stavom a výsledkom. Načítanie stavu vstupov prebieha tzv. vzorkovaním. Po načítaní stavu vstupných signálov je zahájené **spracovanie programu**, uloženého v pamäťovo prerušenej oblasti vyhradenej pre program. Program je spracovávaný zhora dolu (spravidla v hlavnom organizačnom bloku) ako je napísaný a zostavený programátorom. Pre chod programu sa využíva pomocná pamäť pre ukladanie dočasných výsledkov spracovania, pamäť pre lokálne premenné, platné len po určitú dobu a nevyhnutný zásobník hĺbky vnorenia pre uloženie volania podprogramov, návratových adries, obnovu registrov a iné[7, 1].

Počas spracovania programu sa už podľa algoritmu generujú výstupné signály pre riadenie procesu. Stavy signálov sa nezapisujú priamo do výstupných periferií, ale sú ukladané do pamäti zvanej **obraz procesných výstupov (PQI - Process Output Image)**. Dôvod je rovnaký ako pri vstupných signáloch, a to zaistenie jednoznačnosti stavu. Po dokončení programu, je požadovaný stav výstupov synchronne zapísaný na fyzické výstupy periférií.

Vyššie uvedený popis, uvádzajúci synchrónne načítanie vstupov a zápis výstupov má ome-dzenie v počte signálov, ktoré sú spracovávané. Dôvodom je to, že v reálnom čase nie je možné presne v jednom okamihu aktualizovať stovky až tisíce signálov. Takže uvedené je možné považovať za reálne napríklad pri malých PLC systémoch. Pre väčšie modulárne PLC systémy (vrátane centrálnych i rozšírených) sa používá radič zbernice, ku ktorému sú periférie pripojené, viď obrázok 2 [7, 1].



Obr. 4. Základný cyklus väčšieho PLC [7]

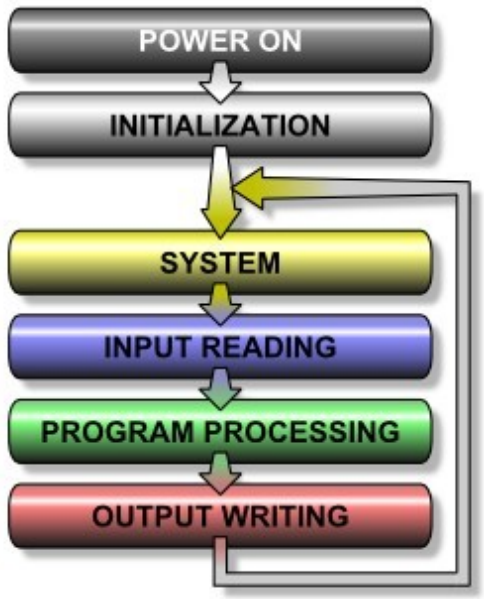
Jednotlivé periférie už môžu byť od seba viac vzdialené, a aj ich množstvo je také, že riadenie toku dát už vyžaduje samostatnú riadiacu logiku periférií, tj. Periphery Bus

Controller. Táto môže pracovať samostatne, ale napriek tomu je synchronizovaná so systémom CPU, takže, pokiaľ je spracovanie programu rýchlejšie ako načítanie aktuálneho stavu všetkých periférií, vyčká systém na dokončenie aktualizácie stavu. Množstvo i vzdialenosť periférií potom nepriaznivo ovplyvňuje rýchlosť odovzdávania signálov i synchronizovateľnosť medzi nimi [7, 1].

1.4 Doba cyklu

Doba cyklu udáva, za ako dlho CPU spracuje všetky potrebné operácie nutné k riadeniu procesu[1].

Základná doba cyklu PLC je taká doba, časový interval, počas ktorého PLC vykoná všetky potrebné operácie ku spracovaniu programu a vráti sa do rovnakého bodu činnosti. Pretože tento bod musí byť stabilne definovaný, predpokladajme, že sa jedná o okamih, kedy je dokončená inicializácia CPU a je zahájená činnosť potrebná pre spracovanie užívateľského programu [1].



Obr. 5. Cyklus PLC [7]

Do základnej doby cyklu PLC sa počítajú následujúce časové intervaly potrebné pre:

- aktualizáciu interných premenných systémov
- aktualizáciu stavu časovačov
- aktualizáciu stavu vstupov (vzorkovanie)
- uloženie obrazu vstupov
- spracovnie hlavného programu a absolútne volaných podprogramov
- načítanie obrazu výstupov

- aktualizáciu stavu výstupov

Táto doba potom určuje najkratšiu reakčnú dobu PLC medzi zmenou stavu signálov na vstupe a reakcií na výstupe, ak neuvažujeme o možnosti prerušenia a priameho prístupu k perifériám a ďalších možných funkciách systému [1, 2, 7].

1.5 Zostava

Zostava PLC sa skladá z niekoľkých modulov v prípade modulárnej zostavy zahrňujúcej CPU a periférie, alebo sa môže jednať i o kompaktné prevedenie PLC, ktoré má integrované potrebné komponenty.

CPU je hlavným modulom riadiaceho systému PLC a musí byť vždy v zostave riadiaceho systému [1, 7, 8].

SW výstavba CPU

- operačný systém pre chod PLC, rutiny pre nábeh (po zapnutí napájania) a ukončení behu (vypnutie napájania)
- synchronizácia taktu a spracovania pokiaľ CPU obsahuje viacej mikroprocesorov
- vnútorná diagnostika HW, kontrola integrity obsahu pamäti systému, programu a užívateľskej pamäti
- dátová výmena medzi programom, systémom a pamäťami
- spracovanie diagnostických prerušení HW, teda chybové stavy (závady) indikované perifériami alebo samotným CPU
- spracovanie diagnostických prerušení SW (programátora), napr. oslovenie neexistujúcej periférie, chyby konverzácie medzi číselnými sústavami, prístup programu k neexistujúcej oblasti pre dáta, chybné odskoky (neexistujúce návestie, podprogramy).
- spracovanie užívateľských prerušení ak je žiadané HW prerušenie z procesu, cyklické alebo kalendárne prerušenie
- spracovanie hlavného užívateľského programu, podprogramov, funkcií, logiky, algoritmov, regulácie ...
- načítanie stavu vstupných periférií a odovzdávanie hodnôt na výstupné periférie, buď na systémovej zbernici alebo decentrálnych perifériách pripojených cez priemyslovú sieť
- komunikácia cez programové rozhranie s programátorom [7]

HW výstavba CPU

- mikroprocesor pre chod PLC, jeden ale spravidla viacej podľa typu a prevedenia CPU, kedy je činnosť CPU rozdelená medzi niekoľko mikroprocesorov (programové, periférne, komunikačné)
- pamäť pre uloženie programu, internú / externú, principiálne typu ROM
- užívateľskú pamäť pre dáta, spravidla internú / ako rozširujúcu externú, principiálne ako RAM
- rozhranie pre komunikáciu s programátorom
- budič zbernice pre pripojenie periférií prípadne ďalších rozhraní pre sieťovú komunikáciu
- niektoré typy CPU disponujú i samostatnou ALU (Arithmetic Logic Unit) [7]

1.6 Periférie

Periférie sú jednotky, moduly PLC, určené pre komunikáciu s okolím. Medzi základné periférie patria digitálne t.j. také, ktoré spracovávajú dvojstavové signály charakteru zapnuté / vypnuté na vstupoch, a tento stav generujú na výstupoch. Ďalšími perifériami sú periférie analógové, zasa vstupné / výstupné, keď vstupné periférie prevádzajú spojitú analógovú hodnotu na diskrétnu číselnú a výstupnú naopak. Periférie, ktoré vykonávajú zložitejšiu činnosť než samotný prevod alebo konverziu signálov sa podľa konkrétneho prevedenia označujú ako funkčné moduly, inteligentné periférie, komunikačné procesory [1].

1.6.1 Centrálna periféria

Za centrálne periférie sú označované také, ktoré sú umiestnené v centrálnom alebo rozširujúcom ráme. V centrálnom, prípadne rozširujúcom ráme, môžu byť umiestnené typovo všetky periférie. Nejedná sa teda o špeciálnu funkčnosť periférií, ale len o označenie príslušnosti k časti systému [7].

1.6.2 Decentrálne periférie

Za decentrálne periférie sú označované také, ktoré sú rozmiestnené po stroji, technológii, obecne zariadení, teda nie sú umiestnené v centrálnom alebo rozširujúcom ráme. Z hľadiska osadenia modulov v decentrálnej periférii je mimo digitálnych a analógových modulov možné i použitie modulov generujúcich prerušenie i modulov s vyššími nárokmi na objemový prenos dát, platí však, že je nutné zvoliť vhodný interface na strane periférie [7].

1.7 Požiadavky

Požiadavky na automatizáciu priemyslu pomocou výpočtovej techniky, konkrétne PLC, má niektoré odlišnosti od bežných PC a ich aplikácií.

U klasického počítača (PC, notebook…) prebieha interakcia medzi počítačom a prostredím pomocou klávesnice (hlasom, ovládačom ...) teda spôsobom, ktorý je pre človeka blízky lebo je schopný sa ho naučiť a naopak, po spracovaní úlohy je výstup prezentovaný v grafickej podobe (znaky, symboly) na displeji zariadenia (optickou interakciou), hlasom (akusticky), hmatateľne (3D tlač, ale aj slepecké písmo). Bežný počítač tiež prevažnú dobu čaká, až ho používateľ k niečomu použije, zadá mu úlohu [7, 8].

1.7.1 Spracovanie signálov

Oproti tomu PLC, musí trvalo a naraz spracovávať veľké množstvo signálov riadenej technológie (automatizovaného procesu), rádovo je dnešný PLC schopný spracovávať 1000 - 10000 binárných signálov oproti 100-1000 v začiatkoch automatizácie. Jednoduchšie binárne signály (zapnuté / vypnuté) predstavujú 80-100% signálov, ktoré sú spracovávané. Menšiu spracovávanú skupinu potom tvoria analógové signály. Samozrejme, pri špecifických systémoch spojitých regulácií môže byť pomer i opačný. S rozvojom sieťových prostriedkov komunikácie je možno analógovú hodnotu digitalizovať priamo "pri zdroji" v snímači a ten prostredníctvom siete odovzdáva PLC informácie o analógovej hodnote vyjadrené číselne ako stavové slovo, alebo PLC prostredníctvom siete komunikuje s ďalšími zariadeniami [7, 8].

PLC spracováva nielen vonkajšie signály, ale aj signály interné. Tieto signály vznikajú ako väzby a kombinácie vychádzajúce z požadovanej logiky riadenia. Ak uvážime základnú možnosť vytvorenia stavovej kombinácie napr. 4 signálov, tj. 16 kombincií (2^4=16) je zrejmé, že s každým ďalším signálom narastá počet možných kombinácií. I pri vylúčení stavov, ktoré reálne nemôžu nikdy nastať, povedzme 95% (pri malých systémoch) a až 99,5% (pri veľkých systémoch) tak pri malom systéme. S 10 vstupnými signálmi dostávame 1024 možných kombinácií, s vylúčením stavov, ktoré nemôžu nastať. Potom 51 možných reálných kombinácií, interných signálov, samozrejme podľa konkrétnej aplikácie. Pokiaľ klasický počítač spracováva veľké množstvo informacií, tak si operátor na výsledok počká trochu dlhšie a v zásade sa nejedná o havarijný stav. Technológiou riadený proces ale nemôže čakať na výsledky zpracovania z PLC, teda nastává potreba kontroly rýchlosti spracovania [7, 8].

1.7.2 Programové riešenie

Kompilátory vyšších programovacích jazykov nie sú vhodné pre programovanie PLC. Málokedy majú funkcie pre prácu s bitmi. Zložito spracovávajú časové intervaly a generujú príliš veľký program v strojovom kóde inštrukčného súboru PLC. Pre účely programovania PLC boli teda vyvinuté vlastné programové jazyky blízke programovaniu v Assembleru. Pre určitý komfort programátora boli vyvinuté i grafické nadstavby zobrazenia PLC programov, ktoré sú názornejšie a prehľadnejšie pre spracovanie binárnej logiky. Príklady zápisu programu v jazyku C a pre PLC sú uvedené nižšie [1].

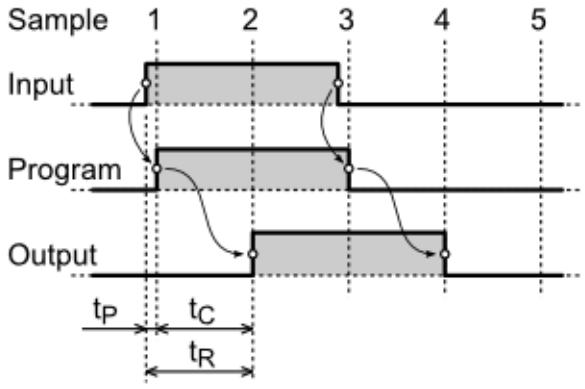
1.8 Reakčná doba

Reakčná doba je časový interval, ktorý uplynie od okamihu zmeny sledovaného signálu až po okamih zodpovedajúci reakcii systému na výstupe [1].

1.8.1 Optimálna reakčná doba

Pre vysvetlenie budeme pracovať s jednoduchým prenosom signálu zo vstupu na výstup s ideálnym vzorkovaním. Signál ideálne zmení stav na "zapnuté" tesne pred okamihom vzorkovania 1, teda je s minimálnym oneskorením zachytený a prevedený do programu ku spracovaniu. Stav zmeny je spracovaný a prenesie sa na zodpovedajúci výstup v okamihu 2. Rovnako tak prebieha prechod do stavu "vypnuté". Reakčná doba $t_{REACT} = t_{CYCLE} + t_{PRE}$ a doba trvania signálu na výstupe $t_{OUT} \approx t_{IN}$ nie je zaťažená časovou chybou oproti signálu na vstupe, ktorá by bola väčšia než doba cyklu, $\Delta t < t_{CYCLE}$.

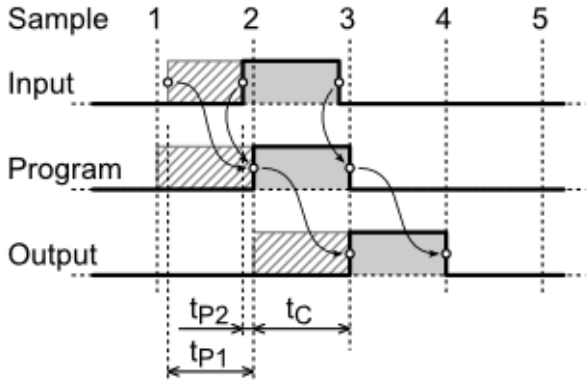
Zatiaľ čo vstupný signál môže a vzniká asynchrónne k cyklu PLC, výstup je synchronizovaný s cyklom PLC [7].



Obr. 6. Optimálna reakčná doba [7]

1.8.2 Odchýlka reakčnej doby

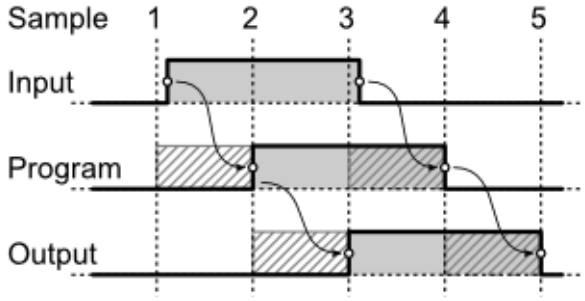
Odchýlkam v reakčnej dobe sa nedá vyhnúť a nejedná sa o chybové stavy. V určitých prípadoch je však nutné na tieto odchýlky brať zreteľ, napríklad, pokiaľ by boli používané snímače pre detekciu polohy využívajíce nezabezpečený binárný kód, napr. dvojkovej sústavy, BCD, Aikenov kód [7].



Obr. 7. Oneskorená detekcia nábehu signálu [7]

1.8.3 Oneskorenie detekcie nábehu signálu

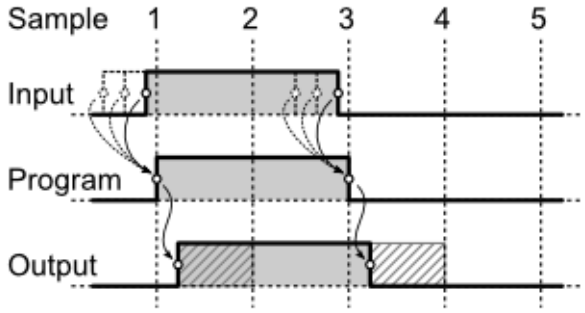
Pokiaľ sme v predchádajúcom prípade signál optimálne vzorkovali, tj. 2 vzorky v priebehu trvania stavu "zapnuté" a dôjde ku zmene periódy vzorkovnia alebo sa zmena stavu signálu "zapnuté" oneskorí za okamihom vzorkovnia 1, je táto zmena zaznamenaná až pri nasledujúcom vzorkovaní 2. Ku strate impulzu signálu síce nedôjde, ale dôjde ku skresleniu doby trvania stavu signálu na výstupe oproti dobe trvania signálu na vstupe [7].



Obr. 8. Zmena detekcie nábehu, ukončenie [7]

1.8.4 Skrátená reakčná doba

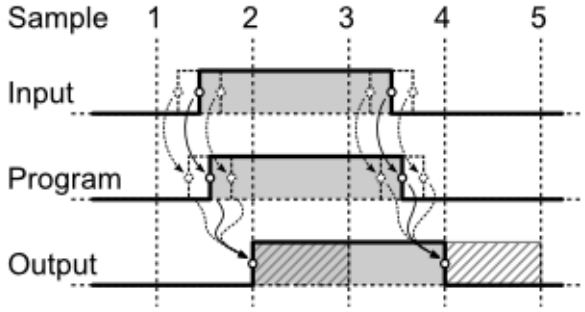
Skrátenie reakčnej doby je žiadúce v prípade, keď je nutné zaistiť minimálnu, najkratšiu, reakčnú dobu PLC na stav signálu [7].



Obr. 9. Skrátenie reakčnej doby [7]

1.9 Spracovanie prerušenia

Prerušenie je taký stav systému PLC, keď vznikne situácia, ktorá vyžaduje vykonanie okamžitej alebo žiadanej reakcie na tento stav [7].



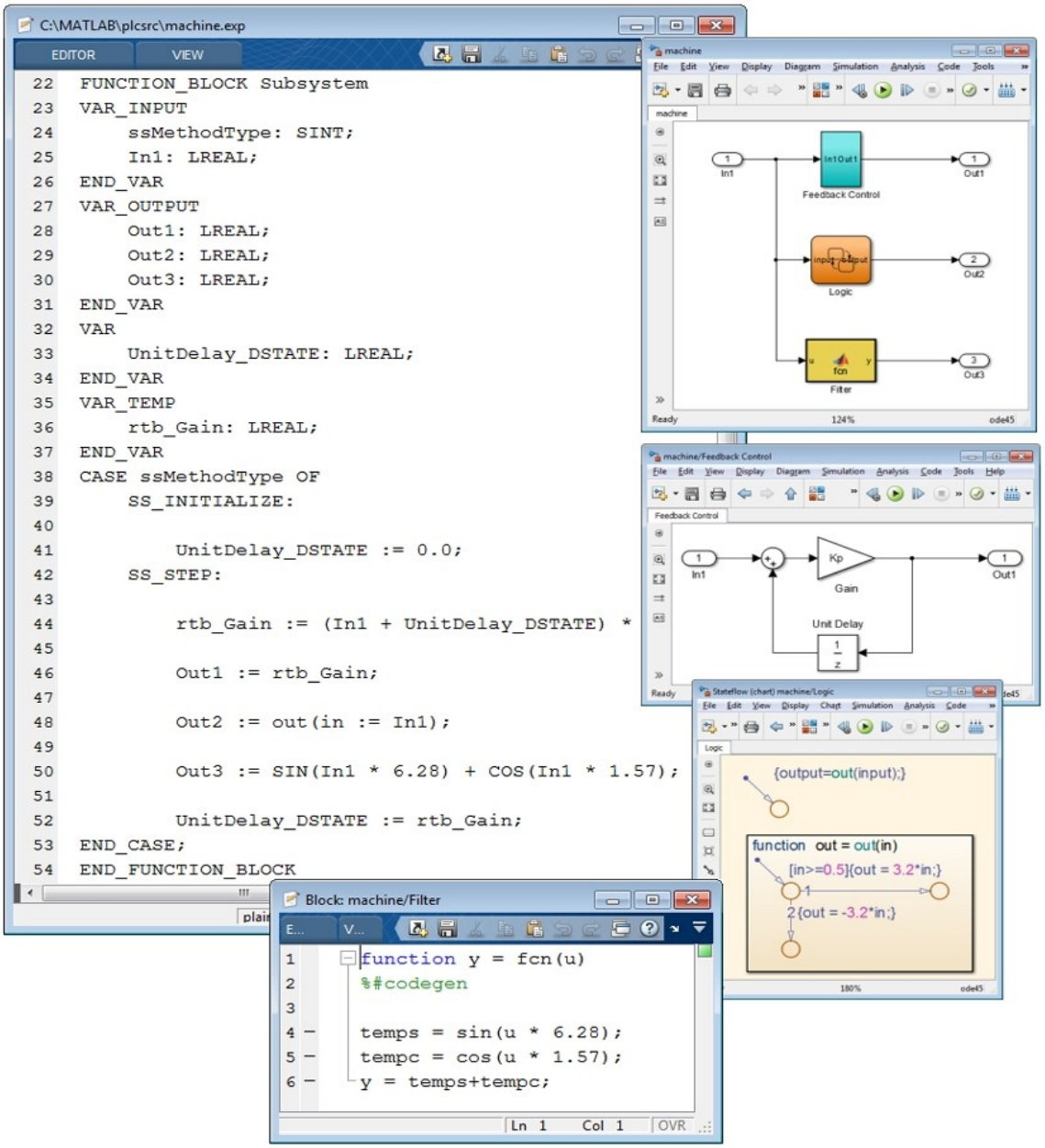
Obr. 10. Skrátenie reakčnej doby prerušením [7]

1.9.1 Typy prerušenia

- Užívateľské prerušenie
- Hardwareové / procesné prerušenie
- Cyklické prerušenie
- Synchrónne cyklické prerušenie
- Časovo oneskorené prerušenie [7]

2 SPÔSOBY PROGRAMOVANIA PLC

V rokoch 1993-2003 bol vydaný štandard IEC 61131, ktorý obsahoval v sekci IEC 61131-3 dokument o programovateľných jazykoch PLC. Hlavným dôvodom pre vytvorenie tohto štandardu bolo práve zjednotenie, štandardizácia existujúcich programovacích jazykov rôznych výrobcov PLC automatov. Aby vyrobcovia Mitsubishi, Siemens, Tecomat a ostatní mali rovnaký programovateľný jazykový štandard. PLC môžeme programovať viacerými spôsobmi: jazyk mnemokódu (IL), štrukturovaný text, jazyk kontaktných schém, jazyk logických schém a Grafcet [9].



Obr. 11. Prehľad programovacích jazykov [13]

2.1 Jazyk mnemokódu – Instruction list (IL)

Jazyk mnemokódu či Instruction list (IL) je programovací jazyk nižšej úrovne, ktorý je podobný assembler a je tiež strojovo orientovaný, to znamená že každá inštrukcia PLC systému zodpovedá rovnakému pomenovaniu príkazu jazyka. Tento typ programovacieho jazyka je vhodný skôr pre malé projekty a tiež pre tvorbu uzatvorenej funkcie. V dnešnej dobe sa kvôli svojej zložitosti iba veľmi málo používa keďže sa nájdu oveľa jednoduchšie a prehľadnejšie spôsoby programovania. Tento typ jazyka je vhodný pre Tecomat. Jazyk mnemokódu používajú predovšetkým profesionálni programátori. Dovoľuje najlepšie použiť možnosti PLC a vyťažiť maximum z inštrukčných súborov [9].

```
#reg bool auto1, auto2, nakladiak, pozor, pretazenie

P 0

LD auto1
AND auto2
OR nakladiak
WR pozor

LD auto1
AND nakladiak
LD auto2
AND nakladiak
OR
WR pretazenie

E 0
```

Obr. 12. Vzor jazyka mnemokódu [Vlastné]

2.2 Štrukturovaný text (ST)

Štrukturovaný text- skratka ST je programovací jazyk vyššej úrovne. Z pohľadu programovacieho jazyka ho môžeme priradiť k jazyku C, Pascal a ďalším. Štrukturovaný text si obľúbia skôr programátori, ktorí sú zvyknutí pracovať práve s programovacími jazykmi vyšších úrovní [9].

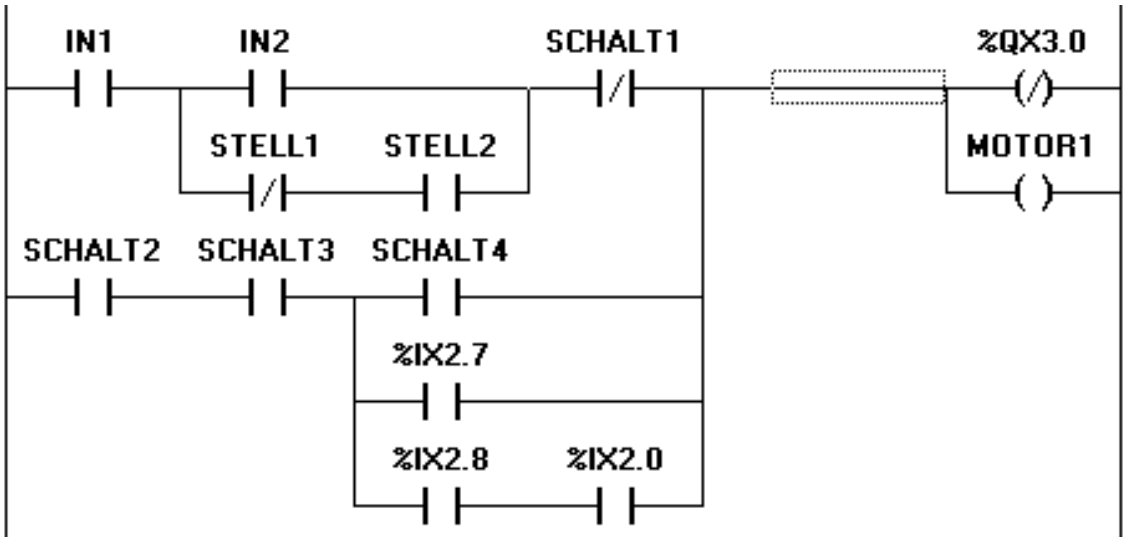
```
PROGRAM prgMain
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  sekunda      : USINT;
  minuta       : USINT;
  hodina       : USINT;
  END_VAR
  VAR

  zapm : ARRAY [0..1] OF USINT:= [19, 44];
  zaph : ARRAY [0..1] OF USINT:= [19, 6];
  vypm : ARRAY [0..1] OF USINT:= [20, 12];
  vyph : ARRAY [0..1] OF USINT:= [19, 8];
  zapvyp : BOOL;
  i : int;


  END_VAR
  VAR_TEMP
  END_VAR

  sekunda:=System_S.COUNTER_SECONDS;
  minuta:=System_S.COUNTER_MINUTES;
  hodina:=System_S.COUNTER_HOURS;

  FOR i:=0 TO 1 DO
  IF zapm[i]=System_S.COUNTER_MINUTES THEN
  IF zaph[i]=System_S.COUNTER_HOURS THEN
  zapvyp:=true;
  END_IF;
  END_IF;
  END_FOR;

  FOR i:=0 TO 1 DO
  IF vypm[i]=System_S.COUNTER_MINUTES THEN
  IF vyph[i]=System_S.COUNTER_HOURS THEN
  zapvyp:=false;
  END_IF;
  END_IF;
  END_FOR;


END_PROGRAM
```

Obr. 13. Vzor jazyku štruktúrovaného textu [Vlastné]

2.3 Jazyk kontaktných (reléových) schém – Ladder Diagrams (LD)

Jazyk kontaktných (reléových) schém je programovací jazyk, ktorý môžeme prirovnať k tvorbe elektrických reléových schém. Program je založený na logických operáciách, zobrazený obvykle vo forme pre kreslenie schém a s prácou s releovými a kontaktnými prvkami. Jazyk LD je grafický programovací jazyk na rozdiel od jazykov IL a ST. Tento programovateľný grafický jazyk je v dnešnej dobe veľmi obľúbený z dôvodu jednoduchosti

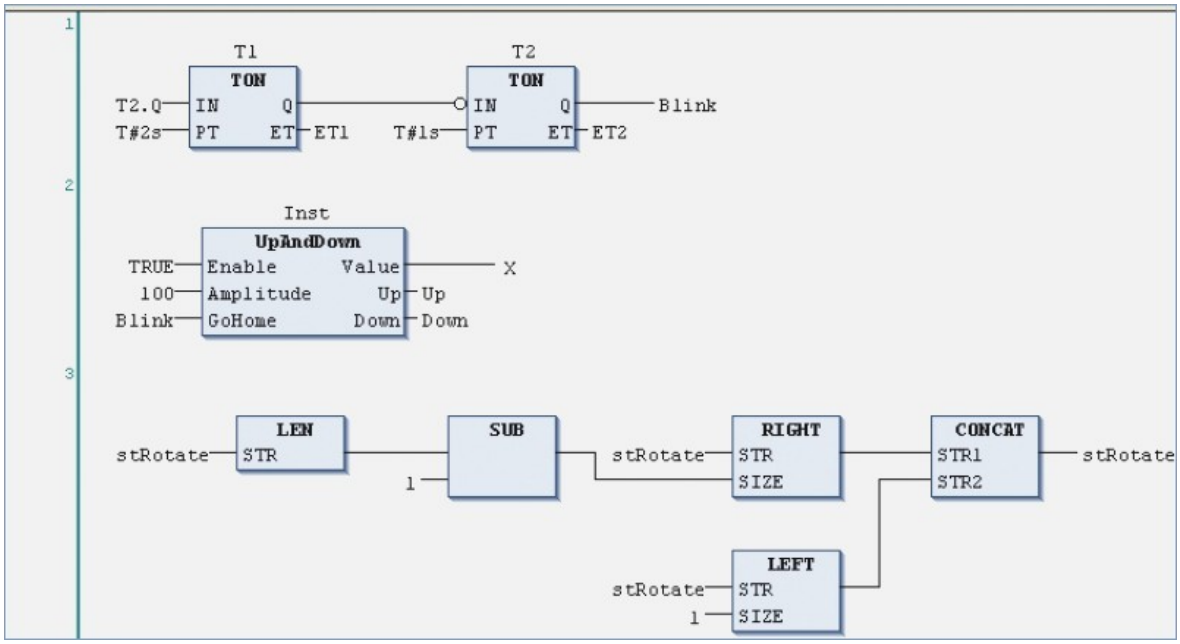
tvorby programu, prehľadnosti zápisu, čítania kódu a jednoduchej diagnostiky v prípade poruchy zariadenia [9].



Obr. 14. Vzor kontaktných schém [14]

2.4 Jazyk logických schém (Function block diagram)

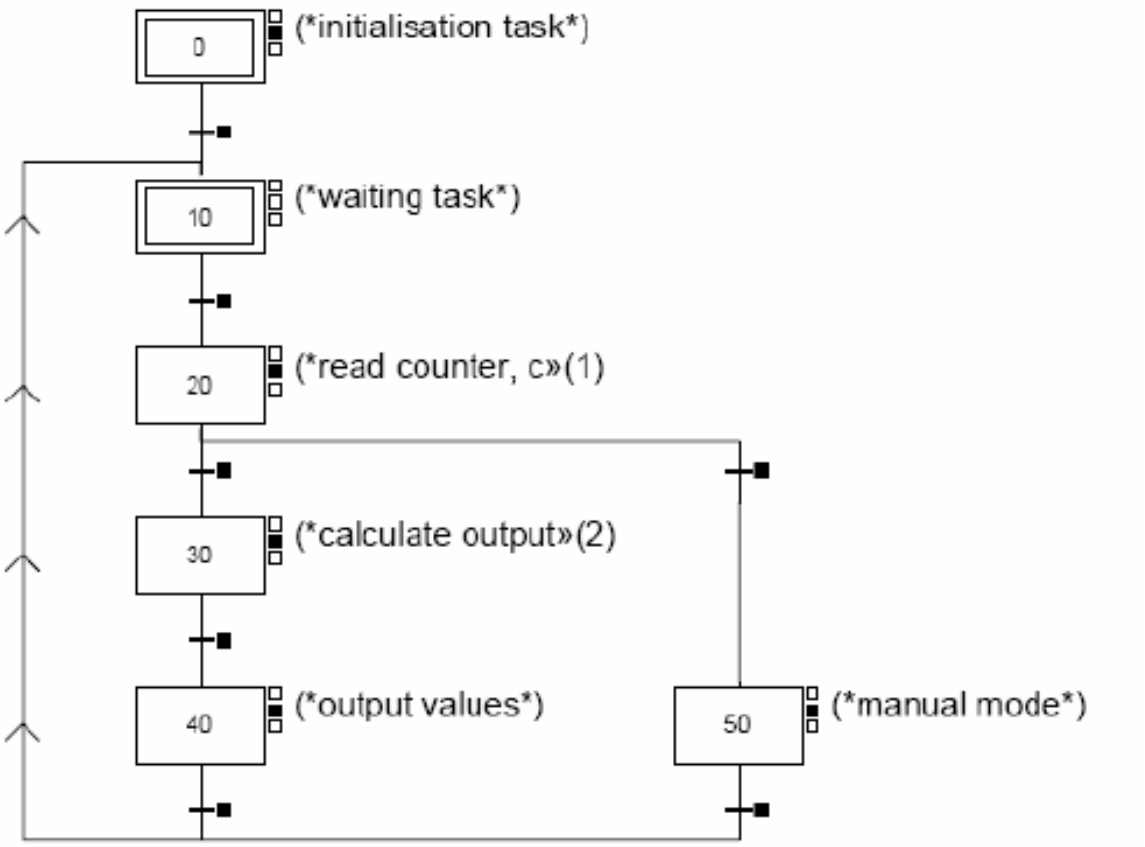
V jazyku logických schém ide o grafický spôsob programovania, kde využívame na programovanie funkčné bloky, na ktoré pripájame vstupy a výstupy. Základné logické operácie sú popísané obdĺžnikovými značkami. Výška značiek je prispôsobená počtu vstupov. Každá z funkcií (časovače, matematické operácie a mnohé ďalšie) má svoje označenie a svoj vlastný blok, ktorý následne spájame spolu s ostatnými blokmi s cieľom vytvorenia vlastného programu, podľa vopred nadefinovaného algoritmu [2, 9].



Obr. 15. Vzor jazyku logických schém [15]

2.5 Grafcet

Jedná sa o grafický spôsob sekvenčného programovania. Je to nadstavba nad popísaním jazyka. Dovoľuje stavový popis sekvenčných úloh v symbolike prechodového grafu konečných automatov a určitej triedy Petriho siete. K popisu štruktúry používame značky stavov, prechodu a vetveniu. Správanie v jednotlivých stavoch alebo definovanie podmienok prechodu sa dá obvykle popísať prostriedkami ktoréhokoľvek už predtým popísaného jazyka alebo ďalším vnorením sekvenčného grafu. Jazyk sekvenčných programovaní je veľmi názorný a podporuje systémový prístup k programovaniu. Programovanie má malý priestor k vytváraniu chaotických neusporiadaných programov. Pri programovaní je potrebné sa zamyslieť nad podstatou problému. Grafcet má možnosť systematicky popísať a realizovať. Väčšina riadených technológií je vo svojej podstate sekvenčných. Pomerne náročným sekvenčným problémom býva vyhodnotenie postupnosti tlačítiek a zásahu obsluhy [9].



Obr. 16. Vzor Grafcet [16]

3 PROGRAMOVANIE PLC TECOMAT

V tejto kapitole sa budeme zaoberať programovaním PLC od firmy Tecomat, v programovacom prostredí Mosaic. Pri programovaní PLC budeme využívať jazyk mnemokódu a štruktúrovaný text. Obidva tieto spôsoby spadajú pod textové programovanie. Základnou jednotkou programovania je tzv. programová organizačná jednotka (ďalej len POU), ktorá je najmenšou časťou užívateľského programu a je zároveň nezávislá. Termín POU spadá pod normu IEC 61 131-3.

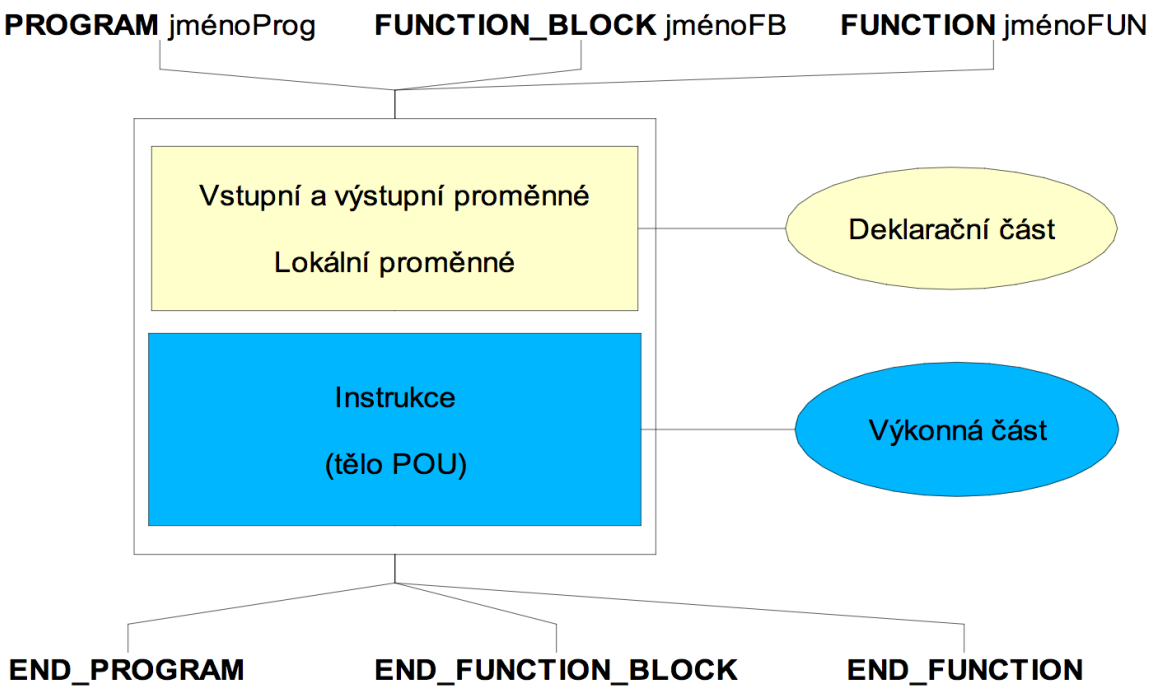
POU sa rozdeľuje na tri typy:

- funkcia (function, v skratke FUN)
- funkčný blok (function block, v skratke FB)
- program (program, v skratke PROG)

FUN produkuje rovnaký výsledok respektíve funkčnú hodnotu, pokiaľ je táto funkcia volaná s rovnakými vstupnými parametrami.

FB vracia viac než jeden výsledok. Pamätá si stavové informácie, ktoré môžu ovlyvniť výsledok. Rozdiel oproti predchádzajúcej funkcii spočíva v tom, že funkčný blok má schopnosť zapamätať si hodnoty požadovaných premenných. Funkcie tieto schopnosť nemajú, ich výsledok je určený vstupnými parametrami.

PROG v užívateľskom programe predstavuje vrcholovú jednotku. Súvisí to s centrálnou jednotkou, ktorá spracováva viac programov a programovací jazyk ST obsahuje definíce spúšťania programov [11].



Obr. 17. Základná štruktúra POU [11]

POU sa skladá z dvoch častí:

- deklaračnej
- výkonnej[11]

Deklaračná časť je potrebná k činnosti POU, definujú sa v nej premenné.

Výkonná časť obsahuje príkazy, ktoré slúžia na realizáciu nami požadovaného algoritmu [11].

3.1 Štruktúra pamäte

Štruktúra pamäte je úsekom pamäťového priestoru PLC. Tento úsek je prístupný na čítanie aj zapisovanie uživateľských dát. Pamäť je rozdelená na niekoľko častí, každá časť má vyhradený význam[11].

Pamäť je rozdelená na štyri časti:

1. obrazy vstupných signálov X
2. obrazy výstupných signálov Y
3. systémové registre S
4. užívateľské registre R

Řada centrálních jednotek	E, M	S	D, B	C, G	K
Obrazy vstupních signálů **X**	X0 : X15	X0 : X127	X0 : X127	X0 : X8191	X0 : X8191
Obrazy výstupních signálů **Y**	Y0 : Y15	Y0 : Y127	Y0 : Y127	Y0 : Y8191	Y0 : Y8191
Systémové registry **S**	S0 : S63	S0 : S63	S0 : S63	S0 : S6143	S0 : S6143
Uživatelské registry **R**	R0 : R255	R0 : R511	R0 : R8191	R0 : R40955	R0 : R64535

Obr. 18. Rozdelenia štruktúry pamäte [11]

3.1.1 Obrazy vstupných signálov X

Pred každým začiatkom cyklu programu zaisťuje centrálna jednotka aktualizáciu tejto oblasti zápisníkovej pamäti zo vstupných periferných jednotiek, na základe deklaračnej tabuľky zadanej v užívateľskom programe, ktorý popisuje priradenie medzi obrazy vstupov X a fyzickými adresami jednotlivých jednotiek.V prípade nedostatku pamäti v oblasti X, možno bez obmedzenia použiť k tomu istému účelu oblasť užívateľských registrov R [11].

3.1.2 Obrazy výstupných signálov Y

Po každom ukončení cyklu programu zabezpečuje centrálna jednotka presun výsledkov z tejto oblasti zápisníkovej pamäti do výstupov periférnych jednotiek na základe deklaračnej tabuľky zadanej v užívateľskom programe, ktorá popisuje priradenie medzi obrazy výstupov Y a fyzickými adresami jednotlivých jednotiek. V prípade nedostatku pamäti v oblasti Y, možno bez obmedzenia použiť k tomu istému účelu oblasť užívateľských registrov R [11].

3.1.3 Systémové registre S

Táto oblasť zápisníkovej pamäti je vyhradená pre špecifické použitie systémovým programom automatu a nedoporučuje sa ju používať pre iný účel. Niektoré bity a byte sú pravidelne v obrátke cyklu nastavované systémovým programom a sú vhodné iba pre čítanie. Niektoré bity naopak modifikujú svojim nastavením správanie systémového programu [11].

3.1.4 Užívateľské registre R

Užívateľské registre slúžia k zálohovaniu dát pri výpadku napätia.

Pri výpadku napájacieho napätia je časť obsahu zápisníka zálohovaná z náhradného zdroja (tzv. remanentná zóna v užívateľských registroch R). Pri opakovanom štarte môžu byť tieto zálohované hodnoty použité i pre ďalšie riadenie - záleží na spôsobe rozbehu a na ďalších

okolnostiach (neporušenosť zápisníka, nezmenený obsah užívateľského programu a pod.. Pri voľbe konfigurácie si môže užívateľ zvoliť veľkosť remanentnej zóny [11].

3.1.5 Prístup k zápisu

Počas doby cyklu užívateľského programu sú údaje zo zápisníka zmrazené, tieto údaje sa aktualizujú až po najbližšej otočke cyklu. Počas priebehu cyklu sa menia premenné, ktoré ovplyvňujú užívateľský program.
Jedná sa o premenné: WR, WRC, WRA, PUT, LET, BET, SET, RES. [11].

3.2 Zásobník

Pri aplikácii užívateľského programu pracuje PLC so zásobníkom, ktorý má 8 úrovní označených A0 až A7 (akumulátor, register výsledkov). Aktívna úroveň A0 označovaná tiež ako vrchol zásobníka je využitá v absolútnej väčšine inštrukcií [11].

PLC TECOMAT má dva modely zásobníka, ktoré sa od seba líšia šírkou jednej vrstvy. Rady B, D, E, M a S majú jednotlivé vrstvy zásobníka široké 16 bitov, zatiaľ čo rady C, G a K majú vrstvy zásobníka široké 32 bitov. Z toho plynú určité rozdiely medzi správaním jednotlivých modelov [11].

So zásobníkom pracujú logické operácie, aritmetické operácie, prenosové operácie, odovzdávajú sa v ňom logické a číselné parametre zložitejších inštrukcií a programov [11].

Zásobník je cyklický, môžeme si ho predstaviť ako bubnovú pamäť [11].

3.2.1 Osmice prepínateľných zásobníkov

Týchto zásobníkov máme k dispozícii celkom 8 (okrem centrálnych jednotiek rady E, kde je len 1 zásobník) označovaných písmenami A až H. Aktívny je vždy jeden a môžeme medzi nimi prepínať. Tým sú otvorené veľké možnosti v oblasti prenosu parametrov medzi funkciami v užívateľskom programe bez nutnosti menej prehľadného medziukladania parametrov do zápisníka [11].

3.2.2 Štruktúra zásobníka

Vrchol zásobníka

Vrchol zásobníka (A0, resp. A01) je aktívnou vrstvou, ktorá má význam registra (akumulátora). Ďalšie vrstvy (A1 až A7, resp. A23 až A67) obsahujú postupne sled predchádzajúcich hodnôt vrcholu zásobníka [11].

Posun zásobníka dopredu

Posun zásobníka dopredu spôsobujú inštrukcie čítania (LD, LDC, ...) a niektoré zložitejšie inštrukcie. Pri každom posune zásobníka dopredu o jednu úroveň sú hodnoty všetkých jeho vrstiev A0 až A6 presunuté do vrstiev s číslami o jednotku vyššími a vrchol zásobníka A0 je obsadený novou ukladanou hodnotou podľa nasledujúceho postupu:

A0 ← nové dáta

A1 ← pôvodný obsah A0

A2 ← pôvodný obsah A1

.....

A7 ← pôvodný obsah A6

Pôvodný obsah A7 sa nenávratne stráca (je prepísaný novým obsahom A0) [11].

Posun zásobníka vzad

Posun zásobníku vzad vykonáva inštrukcia POP a bezoperandové inštrukcie aritmetických a logických operácií. Spätný posun prebieha podľa nasledujúceho postupu:

A0 ← pôvodný obsah A1 alebo výsledok operácie medzi A0 a A1

A1 ← pôvodný obsah A2

A2 ← pôvodný obsah A3

.....

A7 ← pôvodný obsah A0 [11].

INTERPRETÁCIE DÁT NA ZÁSOBNÍKU - MODEL 16 BITOV

Každá vrstva zásobníka má šírku 16 bitov (2 bity). Celú vrstvu označujeme A0, A1, ... A7. Dáta typov dword, udint, dint a real zabierajú dve vrstvy. Hovoríme potom o tzv. dvojvrstve označovanej A01, A23, A45, A67 [11].

3.3 Užívateľské procesy

Užívateľský program sa skladá z užívateľských procesov. Teoreticky ich môže byť až 65, to znamená od P0 po P64, prakticky ich je oveľa menej. Na rozdiel od tradičných operačných systémov reálneho času pre počítače, v tomto užívateľ nemá takúto voľnosť pri ovládaní procesov. Podľa vopred definovaných pravidiel sú aktivované procesy. V rámci týchto pravidiel môžeme dodatočne ovplyvniť aktiváciu väčšiny procesov v priebehu uživateľského programu [11].

Procesy	Určenie
P0	základný proces
P1 až P4	štvorfázové aktivované procesy
P5 až P9	časovo aktivované procesy
P10 až P40	užívateľsky aktivované procesy
P41 až P48	prerušovacie procesy
P50 až P57	ošetrenie ladiaceho bodu
P60	balík podprogramov
P62	teplý reštart
P63	studený reštart
P64	záverečný proces cyklu

Obr. 19. Prehľad procesov užívateľských programov [Vlastné]

Centrálna jednotka rady E má možnosť naprogramovať len proces P0.

Užívateľ nemusí využívať všetky procesy. Pokiaľ mu vyhovuje klasicky jednoslučkové orámovánie riadenia, môže zadať iba proces P0. Proces môžeme vylúčiť troma spôsobmi:

- Proces nie je naprogramovaný, to znamená, že nie sú použité zátvorkové inštrukcie P a E príslušného procesu. Jedine proces P0 nemožno takto vylúčiť, môže však byť prázdny
- Proces je prázdný, to znamená že medzi zátvorkami inštrukciami P a E príslušneho procesu nie je ďalšia inštrukcia. Jeho aktivácia sa prejaví ako prázdna operácia.

- Aktivačná maska procesu je vynulovaná. Aktivačné masky procesov P10 až P48 sú obsiahnuté v systémových registroch S25 až S29. Proces s vynulovanou aktivačnou maskou bude potlačený v nasledujúcich cykloch, respektíve ihneď, ak sa jedná o prerušenie procesu P41 až P48. Ak systém spracováva procesy P0 až P9 nemožno ich takto potlačiť [11].

Pri vstupe do ktoréhokoľvek procesu P0 až P40, P62 až P64 je aktívny užívateľský zásobník vynulovaný [11].

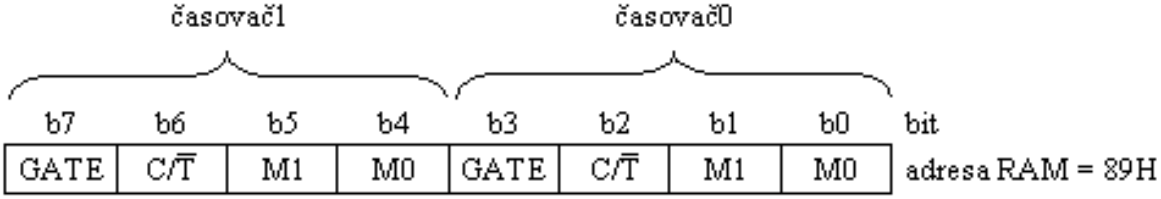
Len inštrukcia EOC je schopná prerušiť postupnosť procesov naplánovaných pre súčasnú slučku a priamo vykonať otočku cyklu. To môže byť účelné napr. pri obsluhe prerušenia alebo pri zaistení rýchlej odozvy na kritickú situáciu. Pri naplánovaní procesu pre ďalší obeh v takto vynútenej otočke cyklu sa nerešpektuje, že niektoré z pôvodne naplánovaných procesov neboli aktivované – začína sa plánovať znovu [11].

3.4 Čítače a časovače

Čítače a časovače tvoria nenahraditeľnú súčasť mikroprocesoru. Funkcia čítača/časovača je tu rovnaká ako u iných zariadení číslicovej techniky - čítač riadi určité pamäťové miesto, ku ktorému pripočíta jednotku na základe zistenej nábežnej alebo zostupnej hrany sledovaného signálu. Sledovaným signálom je u čítača nejaký vonkajší signál, napr. výstupný signál z indukčného alebo fotoelektrického čidla. Časovač tiež riadi určité pamäťové miesto, ku ktorému pripočíta jednotku. Funguje tiež ako čítač, ale na rozdiel od čítača je tu sledovaným signálom vnútorný signál so známym priebehom a konštantnou frekvenciou (tzv. hodinový signál). Časovač sa tiež používa k vytvoreniu určitého časového intervalu. Mikroprocesor 8051 má dva 16-bitové čítače/časovače. Pamäťovým miestom je tu register TH0,TL0 pre čítač/časovač 0 a TH1,TL1 pre čítač/časovač 1. TH predstavuje vyšší bit, TL nižší bit z celkového 16-bitového obsahu čítača/časovača. Ak je nastavená funkcia č/č ako čítač, tak sa v stave S5P2 každého strojového cyklu vykonáva vzorkovanie vstupov T0 a T1 (T0 je pre č/č0, T1 je pre č/č1). Ak je zistená v jednom cykle hodnota na vstupe ako log.1 a v ďalšom cykle ako log.0, potom sa k obsahu príslušného čítača (registra) pripočíta jednotka. Nová hodnota obsahu čítača sa nastaví v stave S3P1 cyklu následujúcom za cyklom, v ktorom bola zistená zmena. Pretože zistenie zmeny na vstupoch mikroprocesoru T0 a T1 trvá 2 strojové cykly, je maximálna čítaná frekvencia vonkajšieho signálu 1/24 frekvencie oscilátora mikroprocesora. Logická úroveň čítaného signálu musí zostať nezmenená vždy aspoň 1

strojový cyklus. Pri použití 12MHz kryštálu je maximálna vstupná frekvencia čítaného signálu 0,5MHz. Minimálna frekvencia čítaného signálu nie je obmedzená. Ak je nastavená funkcia č/č jako časovač, je zdrojom čítaného signálu vnútorný oscilátor mikroprocesora. Jednotka sa potom k obsahu registra TH a TL príslušného časovača pripočíta za každý strojový cyklus. Tento cyklus je tvorený 12 periódami oscilátora. Z toho vyplýva, že maximálna vstupná frekvencia časovača je 1MHz pri 12MHz kryštáli. Konfiguráciu čítačov/časovačov vykonávame nastavením SFR registra TMOD. Riadenia čítačov/časovačov vykonávame nastavením SFR registra TCON [6].

TMOD - Register režimov čítačov/časovačov (Timer/Counter Mode Control) Skladá sa z dvoch štvoríc bitov prislúchajúcich každému z dvoch čítačov/časovačov [6].



Obr. 20. Registre TMOD [6]

Popis jednotlivých bitov registra TMOD [6]:

GATE - Riadenie hradlovania. Ak je GATE=1, potom čítač/časovač n (n=0,1) je aktivovaný (číta) pri vstupe INTn=1 a TRn=1, kde TRn je bit z registra TCON. V tomto režime je činnosť čítača ovplyvňovaná nielen programovo pomocou bitu TRn, ale zároveň i pomocou vonkajšieho signálu privedeného na vstup INTn. Ak je GATE=0, potom čítač/časovač n je aktívny pre TRn=1 [6].

C/T - Voľba čítač/časovač. Bit rozhoduje o režime čítača/časovača. Je C/T=0, potom sa jedná o režim časovača, kedy sa na čítací vstup privádza hodinový signál, ktorý je vytvorený z vnútorného hodinového synchronizačného signálu mikroprocesora vydelením hodnotou 12. Ak je C/T=1, potom sa jedná o režim čítača externých udalostí na vstupe Tn (I/O pin mikroprocesora)[6].

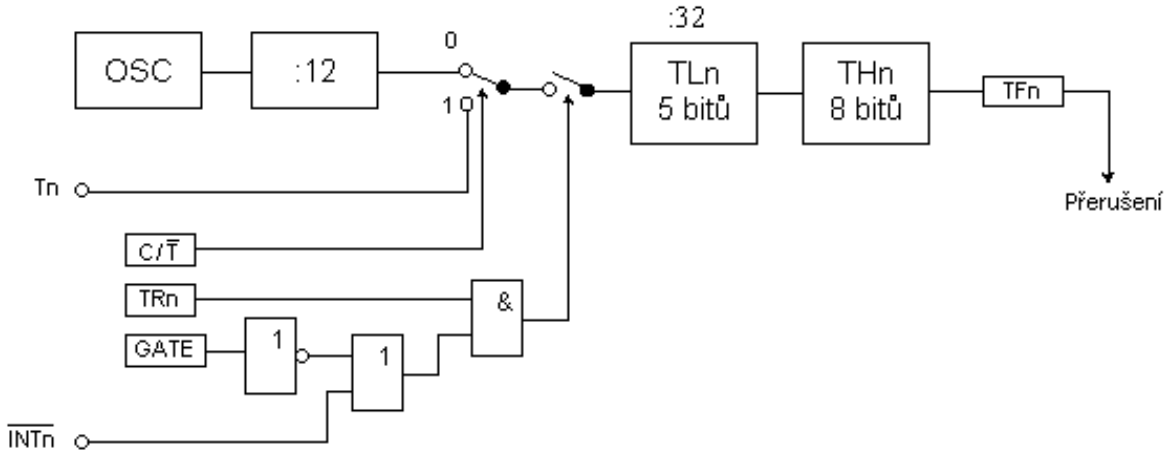
M1,M0 - Kombináciou týchto bitov sa volí jeden zo štyroch módov čítača/časovača. V módoch 0, 1 a 2 pracujú čítače/časovače 0 a 1 ako samostatné, na sebe nezávislé čítače/časovače, tzn. každý č/č má svoj bit TRn,Tn,INTn,TFn. Č/č0 takže môže pracovať v móde 0, č/č1 môže kľudne pracovať v móde 2 alebo môžu č/č0 i č/č1 pracovať v rovnakom

móde, napr. v móde 2. Výnimku ale predstavuje mód 3. Ak č/č0 pracuje v móde 3, je tento č/č rozdelený na 2 samostané 8-bitové čítače s tým, že jeden z týchto 8-bitových čítačov využíva riadiaci a príznakový bit č/č1, č/č1 potom z toho dôvodu nepracuje v plnohodnotnom režime - bližšie viď. popis módu 3. Mód 3 môže byť navolený iba u č/č0 [6].

M1	M2	Nastavený mód
0	0	mód 0
0	1	mód 1
1	0	mód 2
1	1	mód 3

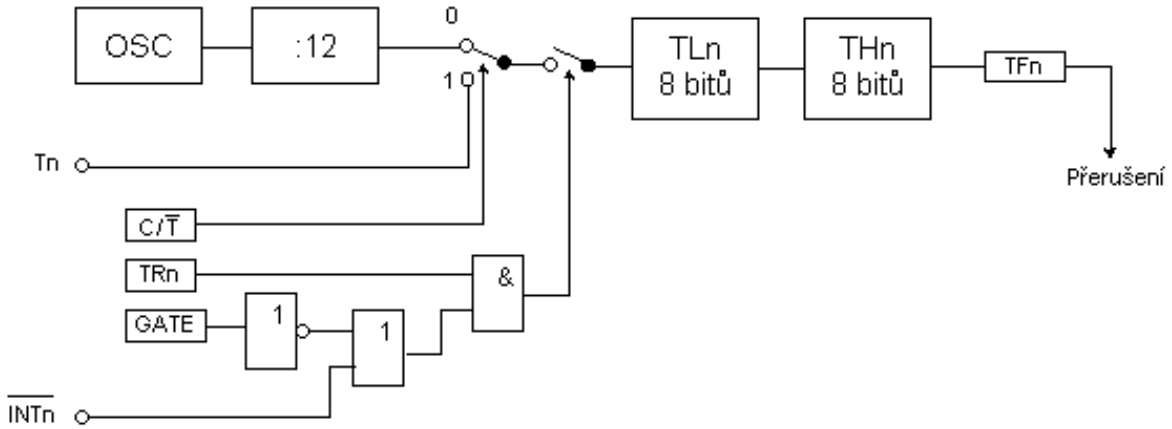
Obr. 21. Typy módov [Vlastné]

Mód 0 - čítač/časovač v tomto móde pracuje ako 8 bitový a je tvorený registrom THn. Tomu je predradený 5-bitový register TLn, ktorý pracuje ako delič 32. Osembitový čítač THn vtedy číta impulzy vydelené 5-bitovým preddeličom tvoreným spodnými bitmi registra TLn, je tak dosiahnutý efekt akoby u 13-bitového čítača. Ak dôjde k pretečeniu obsahu čítača tzn. prechod zo samých jednotiek na samé nuly v THn, nastaví sa príslušný príznakový bit TFn v registri TCON, ktorý môžeme využiť ako zdroj prerušenia mikroprocesoru. Vstup čítaného signálu do č/č je povolený vtedy, ak je TRn=1 a súčastne s ním je GATE=0 alebo INTn=1 [6].



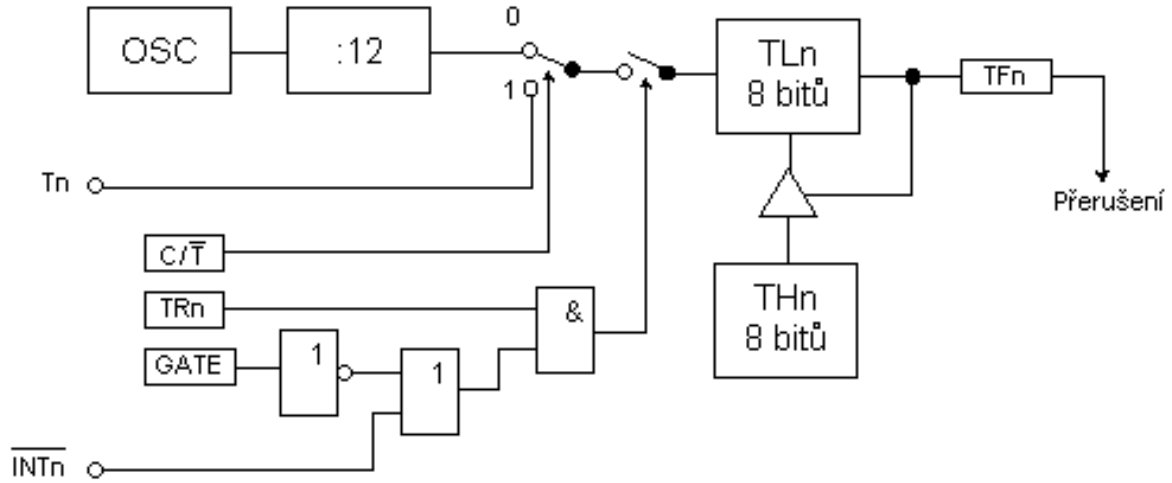
Obr. 22. Čítač/časovač v móde 0 [6]

Mód 1 - mód 1 je rovnaký ako mód 0 s tým rozdielom, že registre THn a TLn tvoriace čítač sú oba 8-bitové a vytvárajú tak 16-bitový čítač. Ak nastane pretečenie čítača, nastaví sa príznak TFn [6].



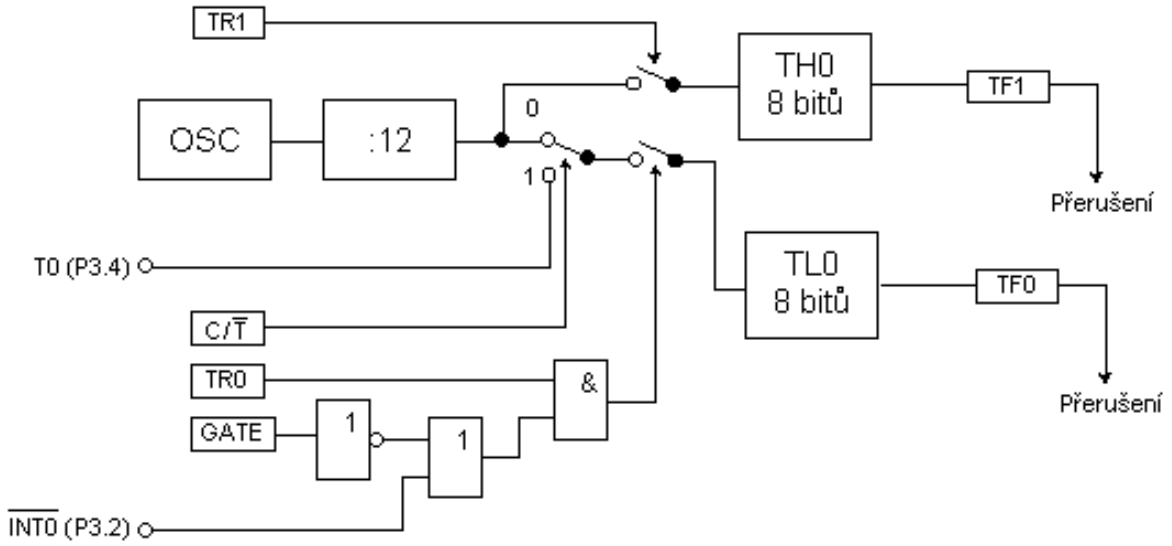
Obr. 23. Čítač/časovač v móde 1 [6]

Mód 2 - v móde 2 pracuje č/č ako 8-bitový čítač s prednastavením. Obsah čítača predstavuje register TLn, v registri THn je uložená predvoľba čítača. Po pretečení obsahu čítača (prechod zo samých jednotiek na samé nuly v TLn), je obsah čítača TLn automaticky nastavený na hodnotu THn. Programové nastavenie novej hodnoty v registri THn neovplyvňuje súčasný stav čítača TLn [6].



Obr. 24. Čítač/časovač v móde 2 [6]

Mód 3 - v móde 3 je čítač/časovač 0 rozdelený na dva samostatné 8-bitové čítače TH0 a TL0. Čítač TL0 využíva štandardné signály C/T, GATE, TR0, INT0 a TF0. Čítač TH0 pracuje vo funkcii časovača a je ovládaný iba riadiacim bitom TR1. Pri pretečení nastavuje príznak TF1. Ak pracuje čítač/časovač 0 v móde 3, potom čítač/časovač 1 môže iba generovať prenosovú rýchlosť pre sériový kanál alebo môže byť použitý v prípade, kedy nebudeme využívať prerušenie. Pretože bit TR1 je využitý pre riadenie č/č0, je zastavenie alebo spustenie č/č1 ovládané jeho nastavením do módu 3 alebo zrušením módu 3 [6].



Obr. 25. Čítač/časovač v móde 3 [6]

TCON - register riadenia čítača/časovača (Timer/Counter Control). Čísla 0 alebo 1 v symbolických označeniach bitov znamená číslo čítača/časovača, ku ktorému prislúcha [6].

b7	b6	b5	b4	b3	b2	b1	b0	bit
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	adresa RAM = 88H
8F	8E	8D	8C	8B	8A	89	88	bitová adresa (hex)

Obr. 26. Registre TCON [6]

TF0, TF1 - príznakový bit pretečenia čítača/časovača. Bit je hardwarovo nastavený pri prechode obsahu čítača z maximálnej hodnoty do nuly. Bit je automaticky vynulovaný pri prechode procesoru do zodpovedajúceho obslužného podprogramu prerušenia [6].

TR0, TR1 – spustenie čítača/časovača. Bit, ktorý ovládame programovo, zaisťuje spustenie alebo zastavenie príslušného čítača (TRn=1 –>> spustenie čítača, TRn=0 –>> zastavenie čítača). Ak je v registri TMOD príslušného čítača nastavený bit GATE=1, potom o spustení/zastavení čítača rozhoduje vstupný signál INTn [6].

Príslušný bit je nastavený pri zostupnej hrane alebo úrovni log.0 na vstupe vonkajšieho prerušenia INTn v závislosti na stave konfiguračného bitu ITn. Po prechode procesora do obslužného podprogramu príslušného prerušenia je bit automaticky vynulovaný [6].

IT0, IT1 - konfigurácia aktivácie vonkajšieho prerušenia. Ak je ITn=1, je žiadosť o vonkajšie prerušenie aktivovaná zostupnou hranou signálu na vstupe INTn. Ak je ITn=0, je žiadosť aktivovaná úrovňou log.0 na vstupe INTn. Ak je signál na vstupe INTn po dlhú dobu v log.0, môže byť prerušenie vyvolané i niekoľkokrát za sebou, pokiaľ doba vykonania

obslužného podprogramu prerušenia je kratšia ako doba, počas ktorej je vstup INTn v úrovni log.0. Bity sa nastavujú i nulujú programovo [6].

3.5 Tabuľkové inštrukcie

Tabuľkové inštrukcie nad tabuľkami T a nad tabuľkami v zápisníku umožňujú čítať z tabuľky LTB a zapisovať do tabuľky WTB, prehľadávať a triediť FTB, FTM, FTS vo formátoch bit, byte, word [1].

Inštrukcie LMS a WMS umožňujú sekvenčné čítanie a zápis do tabuľky vo formáte word. Tabuľka T je súčasťou užívateľského programu a má vždy predpísanú štruktúru, je to vždy rada hodnôt rovnakého formátu (bit, byte, word) s pridanými kontrolnými údajmi. Každej položke je priradené poradové číslo, ktoré sa nazýva index. Nulový index je najmenšia položka, index poslednej položky sa nazýva medz. Tabuľka T môže byť definovaná vo formáte bit, byte alebo word. S každou tabuľkou môže byť uložená iná medz (jednotka meraná v počte bytov). Tabuľky v zápise majú rovnakú štruktúru ako tabuľka T. Medz sa zadá ako prarameter v zásobníku [1].

Aparát tabuľkových inštrukcí dovoľuje realizovať jednoduchým spôsobom aj veľmi zložité funkcie, pričom riešenie býva úspornejšie a rýchlejšie oproti tradičnému spôsobu, vždy je však pružnejšie a prispôsobivejšie. Program je názorný a prehľadný [1].

Každá tabuľka sa dá interpretovať vo formáte bit, byte alebo word podľa použitia inštrukcie. Všetky tabuľky sú uložené bytovo nezavisle na tom, ako boli v užívateľskom programe zadané. Z toho vyplýva:

- bitové zadanie je doplnené v celom počte bytov nulovými bitmi
- pokiaľ interpretujeme vo forme word tabuľku s nepárnym počtom bytov, je posledný byt nedostupný [1].

Priklad tabuľky: #table bit tab_1 = 0, 0, 0, 1, 1, 1, 0, 1

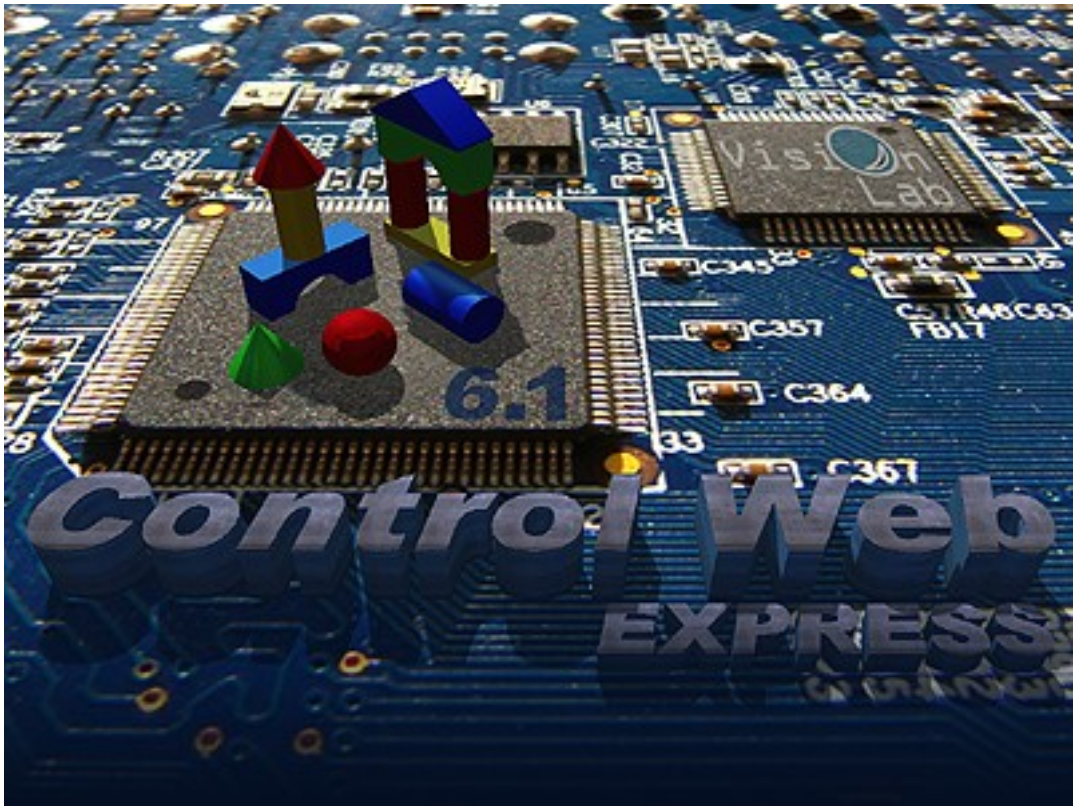
Inštrukcia LTB prečíta položku na zadanie pozícií (indexe) a uloží ju do zásobníku na úroveň A0 (ako inštrukcia LD). Typickým použitím je realizovať pravdivostnú tabuľku, ale aj tabuľku číselných hodnôt napr. cieľ prechodu, čas, nový stav…). Každá položka môže mať štyri bity, slabiky alebo slová [1].

Inštrukcia WTB zapisuje zadaný obsah na zadanej pozíci v tabuľke. Typickým použitím je vytváranie záznamov, zásobníkových štruktúr a dynamických tabuliek. Zapisovaná položka môže mať štyri bity, slabiky alebo slová [1].

Inštrukcia FTB prehľadáva tabuľky, pokiaľ nenájde hľadaný obsah, alebo pokiaľ nenarazí na koniec tabuľky. Odovzdáva index položky s hľadaným obsahom a príznak "nájdené". Pokiaľ obsah nenájde, je index odovzdaný do poslednej položky a zväčšený o jednotku. Hľadaná hodnota môže mať význam aktuálny, v kombinácii vstupných premenných (v šírke do 8 ale aj 16 bitov) [1].

Inštrukcia FTS rozumie položkám tabuľky ako súboru medzí, ktoré rozdeľuje hodnoty usporiadanej množiny napr. rady čísel, časové osy do súvisiacich úsekov, intervalov, skupín. Zadanú hodnotu porovnáva s medzami a odovzdáva poradové číslo, index, úsek, do ktorého hodnotu zaradil [1].
Existuje aj další typ tabuľky a inštrukcií FTM – prehľadávanie s maskovaním [1].

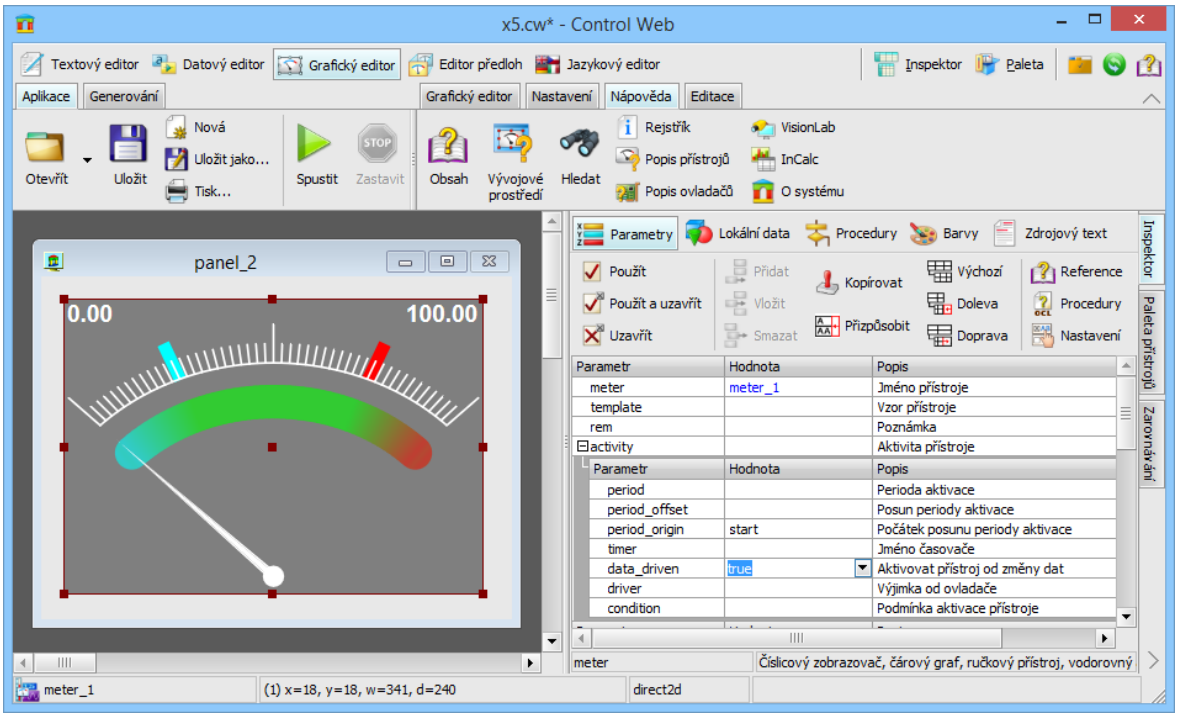
4 CONTROL WEB



Obr. 27. Control Web [17]

Je priemyslový informačný systém pre riadenie a vizualizáciu technologických procesov rozhrania človek – stroj. Je schopný sledovať a archivovať trendy, vytvárať distribuované sieťové aplikácie, ale je tiež vhodný pre výuku a laboratórny výskum. Môže pracovať v reálnom čase, každý V/V kanál je čítaný v čase, kedy to požaduje virtuálny prístroj. Beží na operačných systémoch Windows 95 a vyšších. Control Web 2000 má veľmi široké pole pôsobnosti. Jeho nasadenie je možné od menej či viac komplikovaných riadení a monitorovania strojov, celopodnikových informačných systémov alebo tak náročných štruktúr, ako sú jadrové elektrárne či automobilové podniky a iné. Tu je veľmi dobre rozvinutá podpora internetových a intranetových technológií. Systém obsahuje vstavaný HTTP server, pomocou ktorého je zabezpečená distribuovateľnosť a prepojenosť v počítačových sieťach. Potrebné dátové elementy môžu byť sprístupnené všetkým prepojeným aplikáciám pomocou internetu alebo po ľubovolnej TPC/IP. Sieťová komunikácia môže byť časovaná a aplikácie môžu byť volané a riadené pomocou siete pre optimálny výkon. Umožňuje vizualizáciu technologií pomocou štandardov HTTP a HTML, prostredníctvom ktorých dokáže technológiu i riadiť. I kvôli tomu je Control Web dostupný i v novších verziách Control Web 5 a Control Web 6. Ponúka veľkú škálu komponentov pre

tvorbu vizualizačných aplikácií, ako napríklad zobrazovacie a ovládacie prvky, alarmy, archívy, historické trendy a iné. Virtuálne prístroje sú programovateľné, dajú sa prispôsobovať konkrétnym aplikáciám, ako po vzhľadovej, tak funkčnej stránke, prípadne je možné vytvoriť nový virtuálny prístroj. Dokáže spolupracovať priamo s technológiami pripojenými na V/V kanáli pomocou ovládačov, ktoré je možné ľahko doplniť podľa potreby. Pre spojenie medzi aplikáciami a technológiami sú podporované najpoužívanejšie štandardy - COM/OLE, ActiveX, SQL [10].

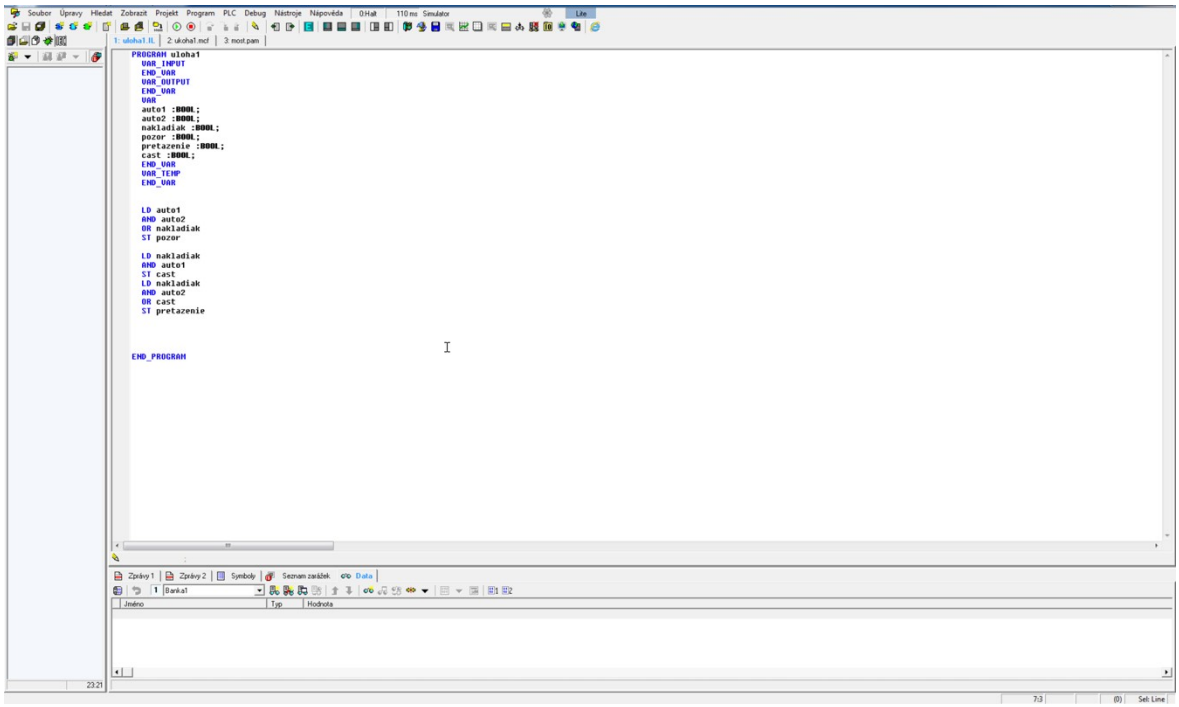


Obr. 28. Vzor práce Control Web [17]

II. PRAKTICKÁ ČASŤ

5 MOSAIC

Táto časť práce bude obsahovať tvorbu programov vo vývojovom prostredí Mosaic. Ďalej programy v jazyku mnemokódu a štruktúrovaný text. Príklady pre zadanie typov úloh: Kombinačné logické úlohy, sekvenčné logické úlohy, čítače, časovače a tabuľkové inštrukcie.



Obr. 29. Vzor prostredia Mosaic [Vlastné]

V prostredí Mosaic si vytvoríme nový skupinový projekt, ktorý pomenujeme a nastavíme podľa požiadavok. Následne v skupinovom projekte vytvoríme nový projekt, ktorý pomenujeme a nastavíme podľa požiadavok. Budeme vytvárať programy na dva spôsoby a to buď na prípony .st alebo .mos. Do stredového okna budeme tvoriť kód. Po dopísaní kódu bude potrebné preložiť kód buď s ikonou v hornom menu alebo s klávesou F9. Po preložení môžeme program spustiť, na to môžeme použiť ikonu v hornom menu alebo s klávesou CTRL+F9. Pre vizuálnosť funkčnosti programu použijeme grafický panel.

6 KOMBINAČNÉ LOGICKÉ ÚLOHY

Kombinačné logické úlohy (funkcie) je pravidlo priraďujúce každej kombinácií hodnôt 0 a 1. Priraďuje vstupným premenným z definičného oboru funkcie jedinú hodnotu výstupnej premennej. Na riešenie takýchto úloh sa pre zjednodušenie používajú Karnaughove mapy.

6.1 Zadanie

Vytvorte program, ktorý bude indikovať most pred preťažením a preťaženie mostu, s použitím dvoch osobných áut a jedného nákladného auta. Ak na moste budú dve osobné autá alebo nákladiak, bude most hlásiť upozornenie pred preťažením. Ak na moste budú všetky tri vozidlá alebo jedno osobné auto a nákladiak, bude most hlásiť preťaženie.

6.2 Riešenie v jazyku mnemokódu

Na začiatku bolo potrebné vytvoriť premenné typu bool (*auto1, auto2, nakladiak, pozor, pretazenie*). V programe sme si načítali premennú *auto1* a pomocou operátora AND sme vynásobili *auto1* s *autom2* a s operátorom OR sme pričítali *nakladiak*. Výsledok sme zapísali pomocou operátora WR do premennej *pozor*. Ďalej sme načítali pomocou operátora LD premennú *auto1*, vynásobili s *nákladiakom*. Následne sme načítali premennú *auto2* a vynásobili s *nákladiakom*. Celé sme to sčítali, výsledok zapísali do premennej *pretazenie*. Nakoniec sme si vytvorili panel na grafické zobrazenie.

6.3 Riešenie v štruktúrovanom texte

Na začiatku bolo potrebné vytvoriť premenné typu bool (*auto1, auto2, nakladiak, pozor, pretazenie*). Vynásobili sme premennú *auto1* a *auto2* s operátorom AND a pričítali premennú *nakladiak* s operátorom OR a zapísali do premennej *pozor*. Ďalej sme vynásobili *nakladiak* a *auto1* s operátorom AND, potom *nakladiak* s *autom2*. Celé sme to sčítali s operátorom OR a zapísali do premennej *preťaženie*. Nakoniec sme si vytvorili panel na grafické zobrazenie.

Obr. 30. Vzor grafického prevedenia programu most [Vlastné]

7 SEKVENČNÉ LOGICKÉ ÚLOHY

Sekvenčné logické úlohy pracujú tak, že priraďujú každej kombinácii hodnty 0 a 1. Hodnoty výstupov ovplyvňujú hodnoty vstupov a oproti kombinačným úlohám aj hodnoty minulého stavu.

7.1 Zadanie

Vytvorte program žeriavu, ktorý sa bude pohybovať vpravo a vľavo, ak príde na kraj, tak zmení smer na opačnú stranu. Pridajte dve tlačidlá pomocou ktorých budete môcť meniť smer a ďalšie tlačidlo pomocou ktorého budete môcť žeriav zastaviť.

7.2 Riešenie v jazyku mnemokódu

Na začiatku bolo potrebné vytvoriť premenné typu bool, uint a int. Načítali sme si premennú *ľavé* tlačidlo, ak sa rovnalo jednej, tak pomocou funkcie SET sme nastavili *smer* na jedna. Ďalej sme načítali *pravé* tlačidlo, ak sa rovnalo jednej, tak pomocou funkcie RES sme nastavili *smer* na nula. Následne sme kontrolovali rozsah pohybu žeriavu od aktuálnej *pozície* po pozíciu maximálnu. Ak sa obidve *pozície* rovnali, funkciou SET sme nastavili *smer*, ale ak sa aktuálna pozícia rovnala minimálnej pozícií, tak s funkciou RES sme nastavili *smer*. Ďalej sme nastavili časovač TON na jednu sekundu a skočili do podprogramu, kde sme časovač vynulovali. Podľa smeru sme buď od aktuálnej *pozície* odčítali alebo pripočítali jednotku a vrátili sa späť do programu. Ku koncu sme načítali tlačidlo *vľavo* a XORom nábežnú hranu vložili do tlačidla *vľavo* a tlačidlo *vľavo* sme vložili do nábežnej hrany. Toto isté sme vykonali s *pravým* tlačidlom. Nakoniec sme si vytvorili panel na grafické zobrazenie.

7.3 Riešenie v štruktúrovanom texte

Na začiatku bolo potrebné vytvoriť premenné typu bool, int, ton a RS, čo je klopný obvod pre funkciu SET a RES. Celý program sme vložili do podmienky IF pre spustenie a zastavenie žeriavu. Ďalej sme riešili, aby sa tlačidlá vracali do pôvodnej polohy. Pomocou IF sme pravé tlačidlo logicky sčítali s nábežnou hranou. Ak výsledok bol logická jednotka, tak sa *hrana* a *pravé* tlačidlo nastavili na logickú nulu. Ak ale vyšla logická nula, tak do *hrany* sa zapísala hodnota na *pravom* tlačidle. Toto isté sme vykonali pre *lavé* tlačidlo. Následne sme použili funkciu RS, ktorú sme nazvali obvod, keď sme do SET vložili hodnotu

maximálnej *pozície* a do RES sme vložili hodnoty minimálnej pozície. Výstupom bol smer chodu žeriavu.

Ešte raz sme použili obvod, keď sme do SET vložili tlačidlo *vľavo* a do RES sme vložili tlačidlo *vpravo*, kvôli zmene pohybu žeriavu. Následne sme použili časovač TON, ktorý sme nastavili na jednu sekundu. Po uplynutí času sa pozícia zmení o jednu hodnotu podľa smeru a vynuluje časovač.

Obr. 31. Vzor grafického prevedenia programu žeriav [Vlastné]

8 ČÍTAČE

Čítače slúžia k počítaniu impulzov signálu, ktoré spracováva program.

8.1 Zadanie

Vytvorte program pre výletnu loď, kde si nastavíte počet volných miest na lodi a program bude počítať koľko ľudí prišlo a koľko odišlo. Pri dovŕšení maximálneho počtu lodí sa spustí rampa aby nemohol už nikto nastúpiť.

8.2 Riešenie v jazyku mnemokódu

Na začiatku bolo potrebné vytvoriť premenné typu bool, uint a cnt. Použili sme čítač CNT, ktorý je obojstranný čítač a podľa nápovedy HELP sme ho nastavili. Načítali sme premennú *prichod, odchod* a *reset* operátorom LD. Potom sme použili CNT čítač a pridali počet miest na lodi. Následne sme načítali počet miest na lodi a porovnali s čítačom. Ak sa budú rovnať tak *rampa* sa zavrie. Ďalej sme porovnali čítač s počtom miest. Ak bude čítač väčší ako *počet* miest, ak skočí na podprogram keď zníži čítač o jednu hodnotu a vráti sa späť. Potom porovná najväčšie číslo UINT, čo je 65 535, s čítačom, a ak sa čítač bude rovnať, tak skočí na podprogram a ten zvýši hodnotu o jedna. Čítač sa dostane na nulu a skočí späť. Ku koncu sme riešili vrátenie tlačidiel do východzej polohy. Načítali sme tlačidlo a s operandom XOR sme pridali hranu a zapísali do tlačidla. Potom sme načítali tlačidlo a zapísali do hrany. Nakoniec sme si vytvorili panel na grafické zobrazenie.

8.3 Riešenie v štruktúrovanom texte

Na začiatku bolo potrebné vytvoriť premenné typu bool, int a ctud. Použili sme čítač CTUD, ktorý je obojstranný čítač a podľa nápovedy HELP sme ho nastavili. Vytvorili sme podmienku IF, keď sme porovnávali nastavenie počtu miest na lodi so skutočným počtom ľudí na lodi. Ak sa počet miest bude rovnať s počtom ľudí na lodi nastaví sa *rampa* na zatvorená. Ďalej sme riešili, aby sa tlačidlá vracali do východzej polohy. Kvôli tomu sme zisťovali nabežnú hranu. Použili sme podmienku IF. Ak *hrana* alebo *príchod* bude pravda, tak *hrana* a *príchod* sa nastavia na nepravda alebo sa *hrana* nastaví na *príchod*. Rovnakým spôsobom sme to riešili pre *príchod* aj *reset*. Nakoniec sme si vytvorili panel na grafické zobrazenie.

Obr. 32. Vzor grafického prevedenia programu loď [Vlastné]

9 ČASOVAČE

Časovače slúžia k odmeraniu časového intervalu potrebného v algoritme programu.

9.1 Zadanie

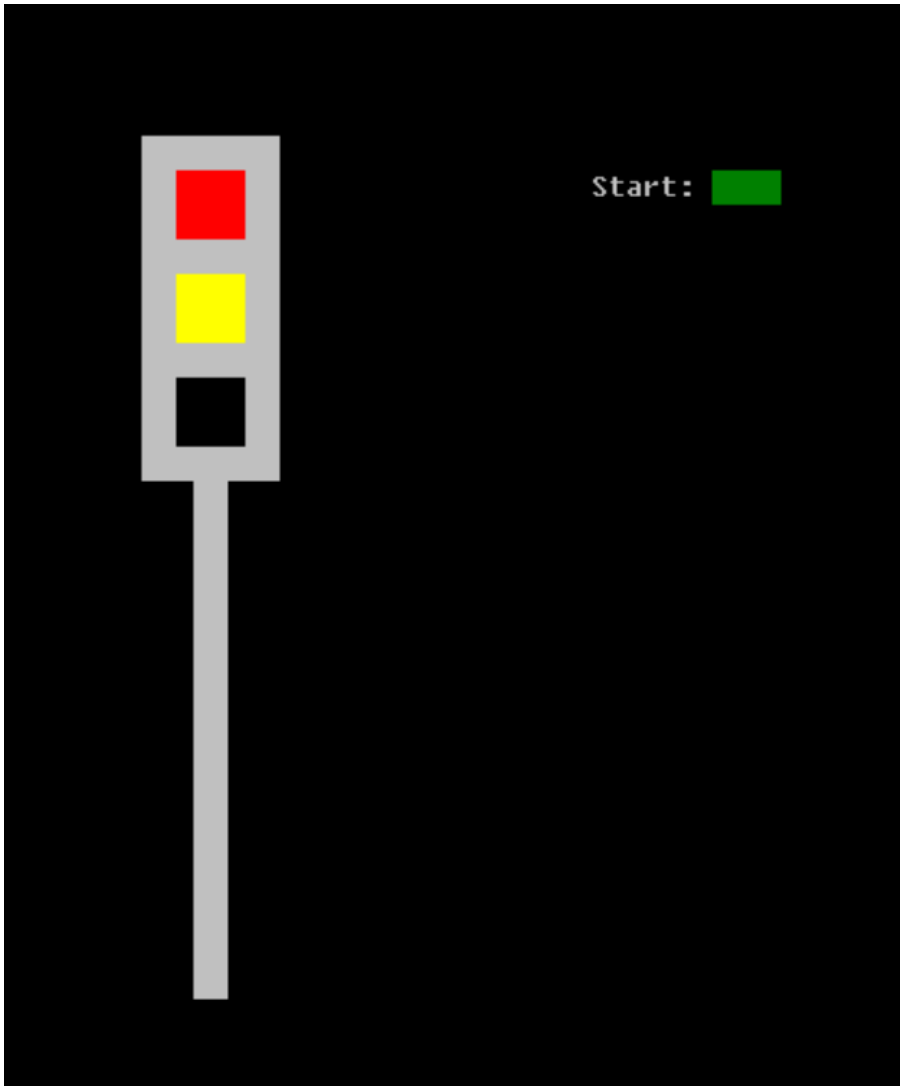
Vytvorte program Semafór, ktorý bude mať dve režimy. Režim pri vypnutom semafóre bude blikať oranžová farba a v druhom režime bude semafór postupne prepínať farby.

9.2 Riešenie v jazyku mnemokódu

Na začiatku bolo potrebné vytvoriť premenné typu bool a uint. Na začiatku programu sme si načítali premennú na *spustenie*. Ak bude pravda, tak skočí na semafór je režim " zapnutý", ale ak bude nepravda do *červenej* a *zelenej* sa zapíše nula a zapne sa časovač na jednu sekundu. Ak sekunda uplynie, tak *oranžová* sa zneguje. To znamená, že oranžová bude blikať. Ak semafór bude zapnutý, tak bude zisťovať, ktoré svetlá sú zapnuté, a podľa toho skočí na ďalšiu farbu. Napríklad, ak bude zapnutá *červená*, tak skočí na skok keď zapne *červenú* a *oranžovú*. Každá farba bude svietiť určitú dobu, na čo sme použili časovač TON. Nakoniec sme si vytvorili panel na grafické zobrazenie.

9.3 Riešenie v štruktúrovanom texte

Na začiatku bolo potrebné vytvoriť premenné typu bool a int. Celý program sme vložili do podmienky IF, keď riešime či je program zapnutý alebo vypnutý. Ak semafór bude vypnutý, tak bude blikať *oranžová*, a to každú sekundu, na čo sme použili časovač TON. Ak uplynie jedna skunda, tak sa *oranžová* zneguje, časovač sa vynuluje a program ide odzačiatku. V režime "zapnutý" sme použili funkciu CASE, a to keď naskočí *červená* na tri sekundy a skočí na ďalšiu časť keď zapne *oranžovú*. Počká sekundu a skočí na následný krok, keď vypne *oranžovú* a *červenú* a na tri sekundy sa zapne *zelená* a skočí ďalej. V poslednom skoku vypne *zelenú* a zapne *oranžovú* na sekundu a skočí na prvý skok, keď zase zapne *červenú* a vypne *oranžovú*. Nakoniec sme si vytvorili panel na grafické zobrazenie.



Obr. 33. Vzor grafického prevedenia programu semafór [Vlastné]

10 TABUĽKOVÉ INŠTRUKCIE

Programovateľné automaty Tecomat môžu používať aj systém tabuliek, kde dáta sú usporiadané do tabuľky a pomocou špecialnych inštrukcií môžeme s tabuľkami pracovať.

10.1 Zadanie

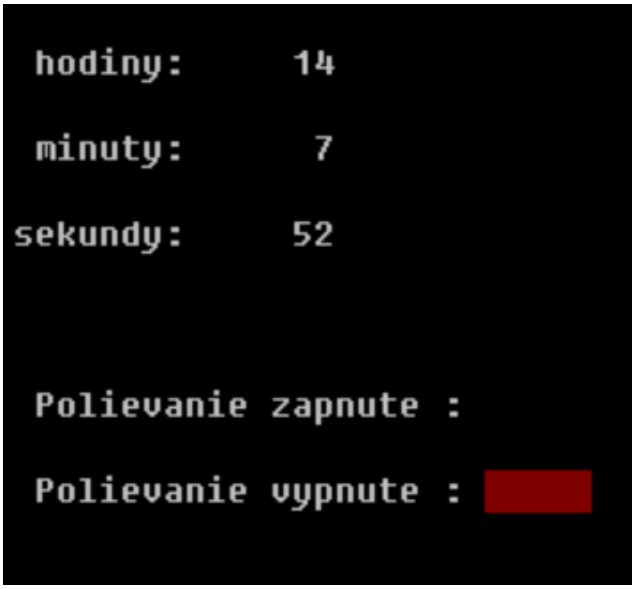
Vytvorte program pre závlahu zahrady, závlaha bude zapnutá od 21:38 od 23:49, opätovné zapnutie závlahy sa zapne od 5:20 do 8:15.

10.2 Riešenie v jazyku mnemokódu

Na začiatku bolo potrebné vytvoriť premenné typu bool, int a table. Vložili sme si do tabuľky *zapnúť*, kde chceme, aby sa závlaha spustila, a do tabuľky *vypnúť* sme vložili dáta kedy závlaha bude vypnutá. Načítali sme systémový čas na dva bity s príkazom LD %SW7, čo nám načítalo minúty a hodiny, ktoré sme porovnali s tabuľkou *zapnúť*. Ak sa zhodovali, závlaha sa spustila. Ďalej sme načítali čas a hľadali sme v tabuľke *vypnúť*. Ak sa čas zhodoval závlaha sa vypla. Následne sme nechali vypisovať systémový čas. Nakoniec sme si vytvorili panel na grafické zobrazenie.

10.3 Riešenie v štrukturovanom texte

Na začiatku bolo potrebné vytvoriť premenné typu bool, usint a array. Veľmi dôležitým krokom je pridanie knižnice do knižnice projektu. Je to potrebné z toho dôvodu, aby fungoval systémový čas. Do deklarácie premenných typu array sme si vložili časy pre spustenie zalievania a pre vypnutie zalievania. V ďalšom kroku sme si vložili systémový čas do premenných *hodiny, minúty* a *sekundy*. Následne sme začali porovnávať polia (array) so systémovým časom. Ak sa systémovy čas zhodoval s časom zapnutia závlah, tak sa závlaha zapla. Následne sme porovnávali systémový čas s časom *vypnutia* závlahy. Ak sa čas zhodoval, tak sa závlaha vypla. Nakoniec sme si vytvorili panel na grafické zobrazenie.



Obr. 34. Vzor grafického prevedenia príkladu závlahový systém [Vlastné]

ZÁVER

Hlavným cieľom bakalárskej práce je vytvorenie podkladov pre intenzívny týždňový kurz zameraný na programovateľné automaty. Cieľom práce je všeobecný popis programovateľného automatu.

V teoretickej časti bakalárskej práce je všeobecne popísaný programovateľný logický automat, jeho história a funkčnosť. Sú vysvetlené pojmy ako otočka cyklu, doba cyklu, periféria, spracovanie prerušení, a ich jednotlivé typy. Ďalšia stať je venovaná popisu rôznych spôsobov programovania PLC, textového a grafického programovania. Teoreticky je priblížený popis fungovania programu PLC od výrobcu Tecomat, jeho štruktúra, registre, štruktúra pamäte a zásobnik. V ďalšej časti práce je popísaný proces fungovania čítačov, časovačov a tabuľkových inštrukcií. Záver teoretickej časti práce je venovaný programu Control Web.

V praktickej časti práce sú uvedené jednotlivé zadania úloh a popis ich riešení dvoma spôsobmi, pričom prvý spôsob bol uvedený v textovom jazyku mnemokódu a druhý spôsob v jazyku štruktúrovaného textu. Úlohy sú riešené postupne, od tých najjednoduchších až po zložitejšie. Zadania úloh začínajú kombinačnými logickými úlohami a pokračujú sekvenčnými logickými úlohami. Na riešenie zložitejších zadaní sú použité časovače, čítače a v závere aj tabuľkové inštrukcie.

K úlohám je vytvorený študijný materiál, v ktorom je veľmi dôkladne popísaná práca vo vývojovom prostredí Mosaic a popis jednotlivých riešení zadaných úloh, či už pre jazyk mnemokódu alebo štruktúrovaného textu.

Tento materiál je určený pre 20-hodinový kurz pedagogických pracovníkov v rámci ďalšieho vzdelávania.

ZOZNAM POUŽITEJ LITERATÚRY

[1] ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. PLC a automatizace, 1.díl. 1. Vyd. Praha: BEN – technická literatura, 1999. ISBN 80-860-5658-9.

[2] ŠMEJKAL, Ladislav. PLC a automatizace, 2.díl. 1. Vyd. Praha: BEN – technická literatura, 2005. ISBN 80-7300-087-3.

[3] MARTINÁSKOVÁ, Marie a Ladislav ŠMEJKAL. Řízení programovatelnými automaty. Vyd. 2. Praha: Vydavatelství ČVUT, 2004. ISBN 80-010-2925-5.

[4] TECO [online]. [cit. 2017-11-23]. Dostupné z: http://www.tecomat.eu/index.php?ID=655

[5] FOXON [online]. [cit. 2017-11-23]. Dostupné z: https://www.foxon.cz/blogs/category/17-kurzy-programovani-simatic-s7-300.html

[6] *DHservis* [online]. [cit. 2018-04-26]. Dostupné z: http://www.dhservis.cz

[7] *PLC automatizace* [online]. [cit. 2018-04-26]. Dostupné z: http://www.plc-automatizace.cz

[8] *UPRT* [online]. [cit. 2018-04-26]. Dostupné z: http://uprt.vscht.cz

[9] MARTINÁSKOVÁ, Marie a Ladislav ŠMEJKAL. *Řízení programovatelnými automaty.* Vyd. 2. Praha: Vydavatelství ČVUT, 2004, 160 s. ISBN 80-01-02925-5

[10] *MII* [online]. [cit. 2018-04-26]. Dostupné z: http://www.mii.cz

[11] *Programování PLC podle normy IEC 61 131-3 v prostředí Mosaic* [online]. 2009 [cit. 2018-04-26].Dostupnéz: https://web.rcmt.cvut.cz/users/cerny/PLC_sup/TXV00321_(v11)_Programovani_PLC_TECOMAT_podle_IEC_61131-3.pdf

[12] *Mylms* [online]. [cit. 2018-05-21]. Dostupné z: https://www.mylms.cz/text-plc-tecomat-1-seznameni-a-prvni-program/

[13] *PLC programátor* [online]. [cit. 2018-05-21]. Dostupné z: https://www.romanjanku.cz/

[14] *Beckhoff* [online]. [cit. 2018-05-21]. Dostupné z: https://infosys.beckhoff.com/

[15] *Cathology* [online]. [cit. 2018-05-21]. Dostupné z: http://cathology.info/

[16] *ResearchGate* [online]. [cit. 2018-05-21]. Dostupné z: https://www.researchgate.net/

[17] *Mii* [online]. [cit. 2018-05-21]. Dostupné z: http://www.mii.cz/

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

ALU	Aritmeticko-logická jednotka
CPU	Centrálna procesorová jednotka
HTML	HyperText Markup Language
http	Hypertext Transfer Protocol
HW	Hardware
IEC	Medzinárodná elektrotechnická komisia
IL	Instraction list
LD	Ladder diagrams.
Napr.	Napríklad
PLC	Programovacie logické automaty
POU	Programová organizačná jednotka
SQL	Structured Query Language
ST	Štrukturovaný text
SW	Software
TCP/IP	Transmission Control Protocol/Internet Protocol

ZOZNAM OBRÁZKOV

ZOZNAM PRÍLOH

1. Príloha k práci: Příprava podkladů pro kurz DVPP se zaměřením na programovatelné automaty

PRÍLOHA P I: PRÍLOHA K PRÁCI: PŘÍPRAVA PODKLADŮ PRO KURZ DVPP SE ZAMĚŘENÍM NA PROGRAMOVATELNÉ AUTOMATY