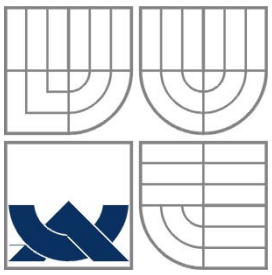
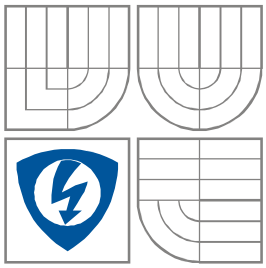


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

PLC ŘÍDÍCÍ SYSTÉM PRO MANIPULAČNÍ RAMENO

PLC CONTROLLED SYSTEM FOR MANIPULATED ARM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MIROSLAV SLAVÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN PÁSEK, CSc
SUPERVISOR

BRNO 2011

ORIGINÁLNÍ ZADÁNÍ DIPLOMOVÉ / BAKALÁŘSKÉ PRÁCE

Abstrakt

V této práci je řešena automatizační úloha řízení manipulačního ramene pomocí řídícího systému Siemens Simatic 200 v prostředí MicroWin. Pohyb jednotlivých částí ramene zajišťují servopohony. Připojení servopohonů k PLC je přes plošný spoj elektroniky, který byl pro tento účel vytvořen. Program pro PLC je hlavním tématem práce. Řídící systém je přes komunikaci PPI napojen na SCADA. Vizualizace procesu je v programu Moravských přístrojů ControlWeb 5.
K manipulačnímu ramenu je připojen dopravník, na kterém jsou dopravovány kostky.
Z kostek je postavena pyramida.

Klíčová slova

ControlWeb, manipulátor, PLC, PWM, servopohon, OPC, Simatic, SCADA.

Abstract

In this bachelor´s thesis is solved automatic task of PLC controled manipulated arm. For this controled is used Siemens Simatic 200 with programing enviroment MicroWin. The servos are moving with parts of arm. The servos are conected to the PLC trought PCB of electronics interface. PCB is created specially for this aim. Design of PCB is described. Details description is PLC´s program. Comunication between PLC and SCADA is trought a PPI cable. Vizualition of proces is runing on the ControlWeb by Moravians instrument, this is descripted also. To the Manipulated arm is instaled conveyor. Boxes are transporting by conveyor. Pyramid is build from boxes.

Keywords

ControlWeb, manipulated arm, PLC, PWM, servo unit, OPC, Simatic, SCADA.

Bibliografická citace:

Slavík, M. PLC řídící systém pro manipulační rameno. Brno: Vysoké učení technické v Brně , Fakulta elektrotechniky a komunikačních technologií, 2011. 45s. Vedoucí bakalářské práce byl Ing. Jan Pásek, CSc.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma PLC řídící systém pro manipulační rameno jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 30. května 2011 ……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Janu Páskovi, CSc za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: 30. května 2011 ……

podpis autora

Obsah

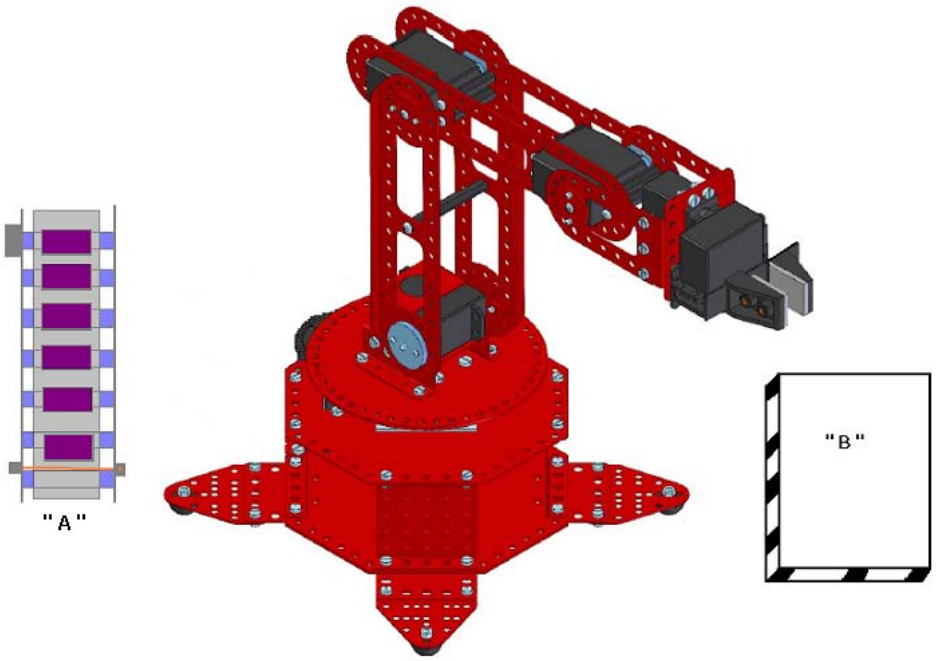
1 ÚVOD

Jak v každodenním životě, tak i v automatizaci řešíme pohyb z jednoho místa na jiné. Ve výrobních procesech řešíme manipulování s předměty s velkou a opakovanou přesnosti a značnou rychlostí. Jindy jsou to pohyby velmi hmotných předmětů, zde již lidská síla nestačí. Zde správně nastupuje automatizační technika. Přesun předmětů z bodu „A“ do bodu „B“ obstarávají manipulátory. Manipulátorů je celá řada, s možností velmi specifických řešení [1]. Většinou jde o jednoúčelové aplikace. S novými trendy se můžeme seznámit v [2]. Univerzálnější jsou manipulační ramena, která mají klouby a jsou schopny vykonávat pohyby jako lidská ruka. Zde jsme již limitování s hmotností břemene, kterou mohou přenášet, ale volnost pohybu je značná.

V této práci se budeme se zabývat modelem manipulačního ramene Merkur. Servopohony VS 18 MB a HS 311 pohybující ramenem budeme ovládat z programovatelného automatu Simatic S7 200. Pro připojení servopohonů vytvoříme plošný spoj s elektronikou. Vytvoříme program v prostředí MicroWin 4.0. Automat připojíme k počítači PC a v programu ControlWeb vytvoříme vizualizaci ve které budeme moci sledovat pohyby ramene, resp. natáčení jednotlivých částí. K tomu sestrojíme dopravník spolupracující s manipulačním ramenem.

2 MANIPULAČNÍ RAMENO MERKUR

Merkur je elektrotechnická stavebnice, ze které je postaven skelet ramene. Pohyb je v pěti osách zajištěn pěti servopohony. Šestý servopohon je určen pro otevírání a zavírání koncového členu (efektoru), v našem případě dvou prstů.



Obr. 1: Manipulační rameno Merkur [4]

Automatizační úlohu přesunu kostek z jednoho místa a stavění pyramidy na druhém, budeme realizovat na zmíněném modelu Merkur. Nicméně koncepce zpracování této úlohy bude obecně platná i pro aplikování na reálných prototypech.

Na Obr.1 vidíme rameno Merkur, dopravník a paletu. Dopravník bude posouvat kostky na konec pásu k místu optické závory, kde bude náš bod „A“. Paleta bude bod „B“, kde se budou kostky stavět. Modelovým příkladem je pyramida, ta bude složena ze čtrnácti kostek ve třech poschodích.

Pěkně viditelné jsou servopohony. První servo je v základně manipulátoru a přes ozubený převod a axiální ložisko otáčí ramenem kolem svislé osy. Druhé servo natáčí první část ramene ve svislé poloze. Třetí servo natáčí druhou část ramene ve vodorovné poloze. Čtvrté servo otáčí ve svislé poloze koncovou třetí část. Na konci třetí části je páté servo, které otáčí koncovým efektorem kolem vodorovné osy. Šesté servo je připojeno na mechanický převod mezi otočným a posuvným pohybem na uchopovací prsty.

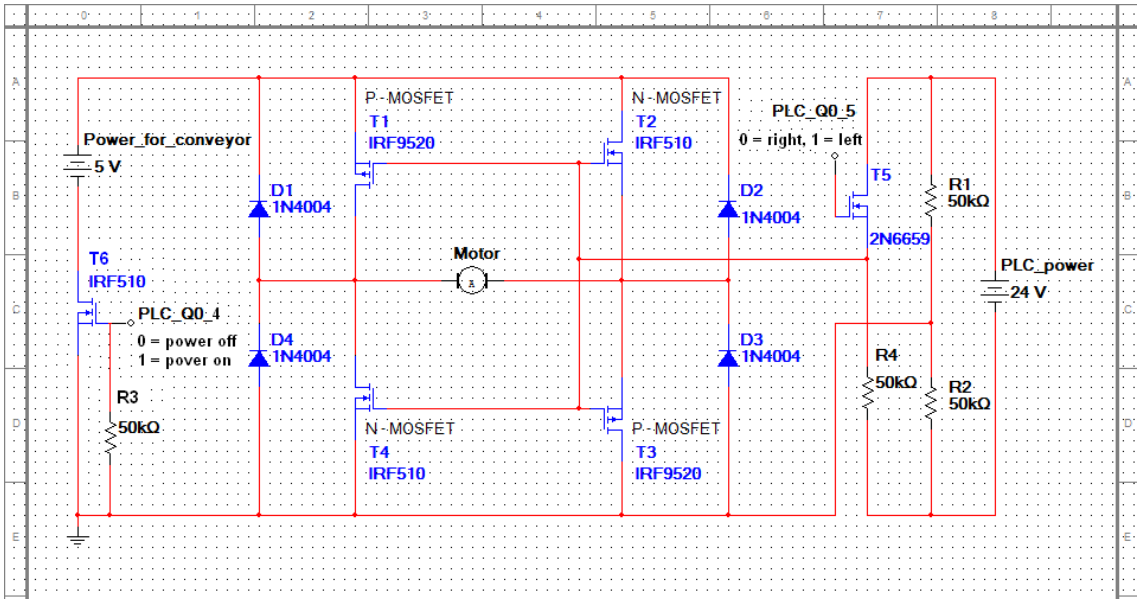
3 DOPRAVNÍK

Dopravník je poháněn stejnosměrným (DC) motorem. Přes převodové kolo je hřídel spojena na hnací válec dopravníkového pásu. Zapnutím dopravníku tlačítkem start, pás posouvá kostku dokud nepřeruší optickou závoru. Po přerušení optické závory se pás zastaví a vyšle se informace pro rameno Merkur, že je předmět k odebrání. Po odebrání kostky se po krátkém prodlení pás znovu rozjede dokud kostka opět nepřeruší optickou závoru. Časové prodlení je zde kvůli případnému zákmitu optické závory a rozjetí pásu dříve než bude kostka bezpečně odebrána. Směr pohybu pásu bude ve směru k optické závoře. Při demontáži pyramidy, ale bude potřeba kostky odvážet zase zpátky. Motor dopravníku je tedy potřebovat reverzovat.

3.1 Elektrické schéma dopravníku

Dopravník má snímač (sensor) a pohon (actor). Snímačem je optická závora, ta má část vysílací tvořenou vysoce svítivou úzkovyzařující LED diodou. Přijímací část obsahuje fototranzistor KP101 připojený mezi kolektor a bázi tranzistoru NPN BC 339. Z kolektoru se odebírá výstupní signál stavu optické závory. Bude-li paprsek LED diody přerušen kostkou, bude tranzistor NPN uzavřen a výstup bude +24 V a logická 1 pro vstup PLC. Nebude-li kostka mezi vysílačem a přijímačem optické závory, bude tranzistor NPN otevřen a výstup bude 0 V a logická 0 pro vstup PLC.

Motor dopravníku je umístněn v tzv. H-můstku. Toto zapojení se vyznačuje tím, že lze jednoduše motor reverzovat, tedy otočit směr otáčení.



Obr. 2: H - můstek

Na Obr. 2 vidíme, že motor je zapojen mezi čtyři unipolární tranzistory. Jsou-li řídícím signálem sepnuty tranzistory T1 a T3, motor se otáčí doprava. Jsou-li sepnuty tranzistory T2 a T4, motor se otáčí doleva. Toto přepínání lze uskutečnit, když motor stojí. Zapnutí / vypnutí motoru určuje výstup PLC Q0.4. Pří výstupu 24 V motor běží. Tranzistory T1, T3 jsou P-MOSFET a T2, T4 jsou N-MOSFET. MOSFET tranzistoru mají hned několik výhod. První je, že v sepnutém stavu mají malý odpor okolo půl ohmu, záleží na typu. Největší předností je, že mají řídící elektrodu G izolovanou od přechodu DS. Nemusíme nastavovat žádné bázové proudy. A elektrody můžeme jednoduše pospojovat ! Rozdílnost typů vodivostního kanálu umožňuje realizovat zapojení, ve kterém řídíme směr otáčení jen výstupem PLC Q0.5. Záporné předpětí otevře T1, T3 a uzavře T2, T4. Sepnutím tranzistoru T5 se přivede na řídící elektrody G kladné napětí, které uzavře T1, T3 a otevře T2, T4.

Zapojení je navrženo a odsimulováno v prostředí NI Multisim 10.1 a soubor je přiložen na CD jako Dopravnik.ms10.

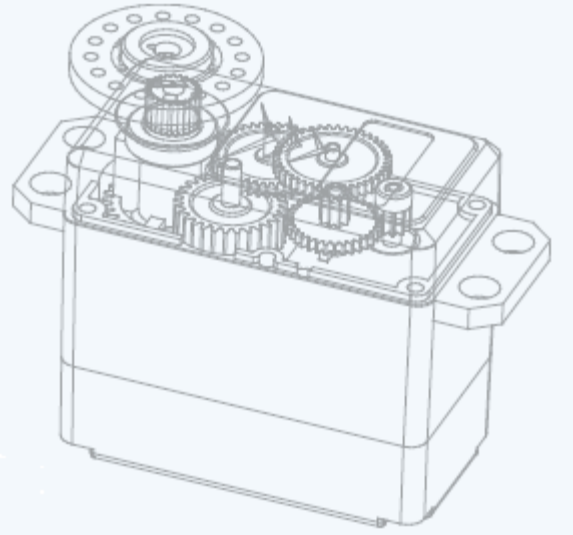
3.2 Rychlost dopravníku

Stejnosměrné motory mají tu výhodu, že se rychlost dá jednoduše měnit pulzní šířkovou modulací (PWM). Je-li střída signálu 1:1 točí se motor poloviční rychlostí. Poměrem střídy signálu můžeme řídit libovolně otáčky a tedy rychlost. Rychlost bude nemněná, ale lze ji pulzy z výstupu PLC Q0.4 v případě potřeby měnit. Náš motor je na napětí 6V a výkon 6W, lze jej provozovat i na napětí 5V, ale výkon bude nižší, ale pro nás stále postačující.

4 SERVOPOHON

Servopohon je ucelená jednotka, která obsahuje stejnosměrný motor (může se ale i jednat o střídavý), převodové soukolí a vlastní elektroniku.

My budeme používat modelářské tzv. analogové servopohony Vigor VS 18 MB [6] a Hitec HS 311 [7]. Servopohony Vigor mají kovové převodové soukolí a jsou schopny vyvinout větší sílu, potažmo moment. HS 311 mají soukolí z umělé hmoty. Jinak mají společné vlastnosti. Napájení od 4,8 V do 6 V. Převodový poměr soukolí je značný 1 : 247. Na výstupním kolu, které otáčí hřídelí je oproti hřídeli umístněn potenciometr snímající natočení. Je to tedy zpětná vazba, že se hřídel natočí, jak požadujeme.



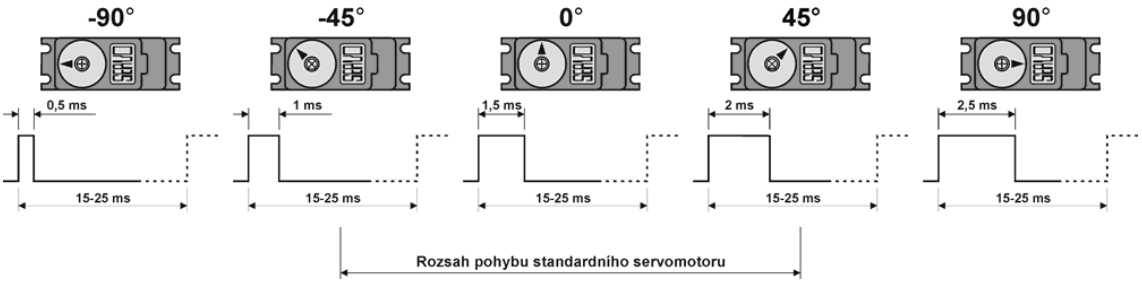
Obr. 3: Vigor VS 18 MB [6]

Obr. 4: Hitec HS 311

4.1 Ovládání servopohonu

Z návodu pro sestavení manipulačního ramene Merkur [4] se dozvídáme potřebné informace i o ovládání. Způsob ovládání je PWM. Pulzy jsou v rozmezí 500 μs do 2500 μs. Krajní poloze, kterou budeme mít 0° odpovídá první hodnota. Druhá hodnota je pro krajní úhel 180°. Neutrální a výchozí poloze 90° odpovídá impulz 1 500 μs. Tuto skutečnost jsem ověřil a zjistil jsem, že hodnoty pozicím odpovídají. Výrobci udávají menší rozsah pohybu, zřejmě s ohledem na udávaný moment hřídele. Dále záleží, jak nasadíme unášeč na tisícihran hřídele. Jsme například v neutrální poloze malinko vychýlení na jednu či druhou stranu. Toto ale nečiní potíže. Pulzy musí přicházet v časovém intervalu od 15 ms do 25 ms. Nedodržením tohoto rozsahu roste nebezpečí,

že se servo nebude natáčet, jak chceme nebo vůbec se nenatočí. S použitím PLC s tím nebudeme mít potíže.



Obr. 5: Polohy serva [4]

Ovládací pulzy se přivádějí o napětí 5V na žlutý vodič. Zbylé dva jsou napojeny na zdroj o napětí 5V a výkonem 40W, který je součástí elektrotechnické stavebnice.
Na Obr.5 vidíme, jak se natáčí servo při jednotlivých pulzech PWM. Natočení se uvádí od -90° do 90°, my budeme používat nezáporné úhlové hodnoty. Náš offset bude jen logický, bez vlivu na nastavení PWM. Samozřejmě lze polohovat i mezi vyznačené hodnoty. Výrobce uvádí „mrtvé pásmo“ 5 µs. My budeme přecházet z jedné polohy do druhé po stupni natočení 2° / 200 ms. Dvěma stupňům odpovídá přídavná šířka impulzu 22 µs.

$$\frac{(2500-500)\,\mu s}{180\,^\circ} = 11\,\mu s/1\,^\circ \tag{1}$$

4.2 Pohyb o 360 °

Jistě si všimneme, že je pohyb omezen na půlkruh. Ve skutečnosti můžeme jít ještě trochu dál, ale pak nám zabrání mechanický výstupek jednoho kola serva. Tento výstupek je demontovatelný [8]. Ale hlavní příčinou je zpětnovazební potenciometr řídící elektroniky, který se nemůže otáčet o celou otáčku. Ten sice můžeme demontovat a nahradit dvěma rezistory, jak je v [8] napsáno, ale docílíme tím jen, že se servo bude otáčet od neutrální polohy na stranu, dokud bude mít napětí, potažmo el. energii. Tedy ztratíme výhodu jednoduché a účinné zpětné vazby. Vzhledem k tomu, že jde o nevratný zásah a u serva Vigor i náročný (kovový výstupek), tento úkon neprovedeme a spokojíme se s „poloprostorem“, který je pro náš účel dostatečný.

4.3 Demultiplexer k servopohonům

Servopohony potřebujeme připojit a ovládat z PLC. Vlastnosti PLC Simatic S7 200 cpu 222 zjistíme z [9]. Zde vidíme, že maximální kmitočet spínání výstupů je 2 kHz, tedy 500 μs. Programově můžeme pracovat s nejkratším časem 1 ms. Je možnost se dvěma výstupy pracovat ve speciálním režimu v mikrosekundách.

Možná řešení :

- Na komunikačním portu 0 nastavit uživatelskou sériovou komunikaci. Vytvořit program pro komunikaci a vytvořit sběrnici pro serva.
- Připojit rozšiřující modul analogových výstupů. Výstupy připojit na převodník U / f . Kmitočet bude přímoúměrně odpovídat napětí a potažmo šířce pulzu PWM. Vytvoříme oscilátor s kmitočtem 50 Hz a pulzy namodulujeme.
- Využití principu časového multiplexu. Mnohé číslicové měřící přístroje tento způsob používají pro převod několika analogových signálů na číslicový s jedním A / D převodníkem.

My se budeme zabývat poslední uvedenou možností. Využijeme speciální konfiguraci výstupů PLC Q0.0 a Q0.1 v režimu PWM. Maximální pulz bude mít šířku 2 500 μs, při periodě 24 000 μs. (Zvolit můžeme i periodu ve spodním rozsahu 15 000 μs). Na každý výstup připojíme demultiplexor (DMX), který nám šířku periody rozdělí na tři intervaly 8 000 μs . DMX jako součástka 74F139 je dvakrát dekodér 1 ze 4 [10]. Má dva adresové vstupy, kterými se řídí přepínání výstupů. My budeme využívat jen tři výstupy.

Pulzy pro serva mají doporučenou amplitudu 5 V, proto použijeme TTL logiku. Vstupní signálky PWM a řídící signály z Q0.2 a Q0.3 musíme odporovým děličem přizpůsobit na napětí 5V. V manuálu [3] je uvedeno, že pro strmé hrany impulzů je potřebné výstupy zatížit alespoň na 10 % maximálního proudu výstupu. Maximální hodnota na jeden výstup je 800 mA. Vypočítáme tedy hodnoty odporů.

$$R_c = \frac{24\,V}{0{,}088\,A} \equiv 272\,\Omega \tag{2}$$

Celkový vypočtený odpor z rovnice 2 je přibližně 272Ω. Z této hodnoty vypočteme hodnoty děliče pro R_1 a R_2.

$$R_1 = \frac{5\,V}{0{,}088\,A} \equiv 56\,\Omega \tag{3}$$

$$R_2 = 272\,\Omega - 56\,\Omega = 216\,\Omega$$

(4)

Hodnota rezistoru s odporem 56 Ω je běžně k sehnání. Hodnotu 216 Ω musíme zaokrouhlit na 220 Ω. Vypočítáme výkon spotřebovaný na rezistorech. U R_1 nebude dle rovnice 5 potíž. U R_2 dle rovnice 6 již musíme počít s výkonem okolo 2 W a že se bude součástka dost zahřívat.

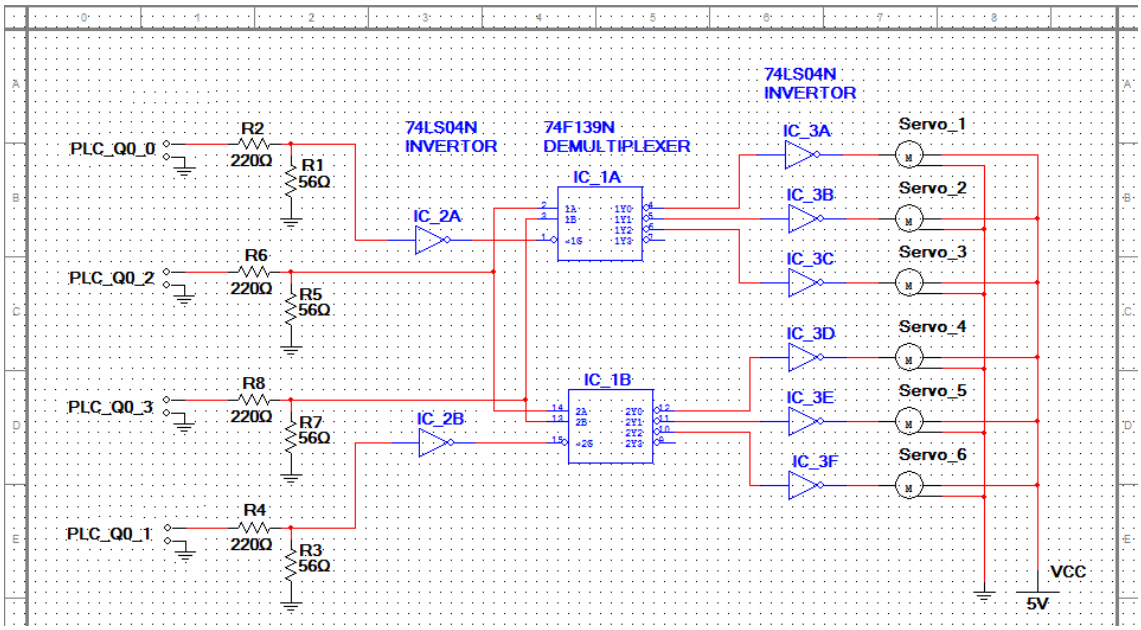
$$P_{R1} = 24\,V * 0{,}088\,A \equiv 2{,}11\,W$$

(5)

$$P_{R2} = 5\,V * 0{,}088\,A = 0{,}4\,W$$

(6)

Obvod 74F139 má kromě adresových vstupů, vstupy a výstupy jsou totiž negované a proto musíme přidat ještě dva invertory 74LS04.



Obr. 6: Elekrické schéma DMX

Elektrické schéma DMX vidíme na Obr.6. Toto schéma bylo navrženo a osdsimulováno v NI Multisim 10.1 a je na přiloženém CD jako DMX.ms10.

Z výstupu Q0.0 přivádíme přes invertor postupně po 8 ms pulzy na vstup enable IO. Adresování příslušného výstupu je určeno kombinací logických hodnot z Q0.2 a Q0.3.

PLC	Q0.0	Q0.2	Q0.3			
IO	E	B	A	S1	S2	S3
	H	L	L	H	L	L
	H	L	H	L	H	L
	H	H	L	L	L	H
	L	x	x	L	L	L

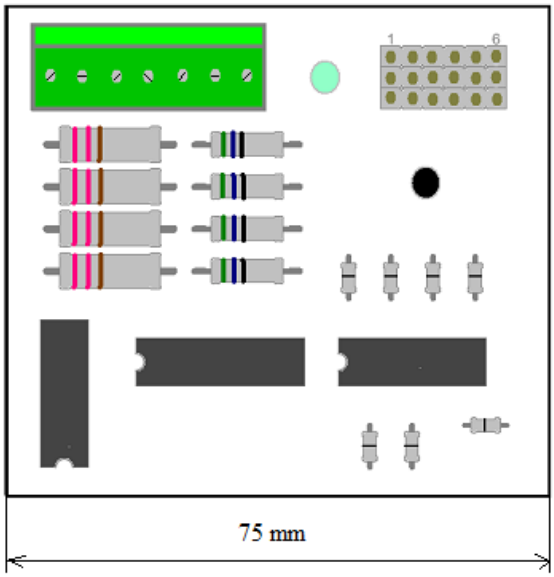
Tab. 1: Funkční tab. pro serva 1, 2 a 3

PLC	Q0.1	Q0.2	Q0.3			
IO	E	B	A	S4	S5	S6
	H	L	L	H	L	L
	H	L	H	L	H	L
	H	H	L	L	L	H
	L	x	x	L	L	L

Tab. 2: Funkční tab. pro serva 4, 5 a 6

Funkční tabulka je uvažována i s invertory a pro tři výstupy. Při spuštění programu a generování PWM pulzu přijde úroveň H na vstup IO E. V prvních 8 ms mají vstupy úroveň L. Úroveň H se přesuny na výstup pro servo S1a S4 . Po přechodu signálu PWM na úroveň L, přenese se tato hodnota i na servo S1a S4, tím polohovací povel je pro první časový interval pro toto servo splněn. V průběhu od 9 ms do 16 ms přejde adresový vstup na úroveň H a přepne na výstup pro servo S2 a S5. V PLC se přiřadí jiná šířka impulzu pro PWM a ta se přes vstup IO E přenese na výstup. V průběhu od 17 ms do 24 ms bude vstup IO A v úrovni L a vstup IO B v úrovni H a poveluje se výstup pro serva S3 a S6. Pro další 24 ms intervaly se činnost opakuje.

Plošný spoj byl navržen v trialové verzi programu EAGLE a je taktéž přiložena na CD jako DMX.brd.



Obr. 7: Elektronika DMX

5 PLC SIMATIC 200

Řízení manipulátoru budeme realizovat s PLC Simatic 200 CPU 222 DC/DC/DC [9], mající 8 digitálních vstupů a 6 digitálních tranzistorových výstupů. Paměť pro program je 4 096 bytů a pro uživatelská data 2 048 bytů. Tento automat je relativně malý (Obr. 8), s možností připojení dvou rozšiřujících modulů. Výrobce řadu 200 označuje dokonce za mikro PLC.

Obr. 8: PLC S7 200 CPU 222

5.1 Program manipulátoru

Programování řady 200 je v prostředí MicroWin. Máme k dispozici verzi 4.0 SP8. Náš program máme možnost vytvořit v programovacích nástrojích LAD, STL nebo FBD. Programovací nástroj je stejný pro celý program, ale dá se ve většině případech přepínat.

LAD je historicky starší. Operuje se zde z reléovými kontakty a cívkami, s možností doplnit bloky FBD. Vhodné pro program silové části.

V FBD pracujeme s logickými obvody. Program se sestává z jednotlivých elementů, které propojujeme čárou (wire). Výhoda je za běhu programu, při hledání chyby, že jednotlivé bloky mají barvu podle logické hodnoty. Ovšem nevýhoda je, že program často sahá „až za roh“ monitoru. A oč je elementárnější, tím je více práce s přetahováním bloků a jejich propojování.

STL je textové programování. Pro zkušené programátory je rychlý a přehledný způsob psaní kódu. Navíc je kód vzájemně přenositelný do poznámkového bloku txt.
A tím je možno vytvořit i dokumentaci.

Pro náš program máme k dispozici jeden hlavní blok OB1 a 64 SBR bloků. Subroutine je vlastně podprogram. Dále máme 8 podprogramů přerušení INT.

Pro programování nás budou především zajímat paměťové oblasti. Máme k dispozici standardní paměť pro vstupy I a výstupy Q. Budeme pracovat převážně se zápisníkovou pamětí M do velikosti 32 bytů. Dále paměť pro proměné V do velikosti 2 047 bytů. PLC obsahuje speciální paměť SM. Zde budeme používat bit SM0.0 jako stabilní logickou jedna. Bit SM0.1 je aktivní pouze při prvním cyklu programu po spuštění do RUNu. Pro časování máme k dispozici paměť T pro 256 časovačů. Z toho máme 4 časovače po 1 ms , 16 po 10 ms a 236 po 100 ms. Paměť C je pro 256 čítačů. Obě pamětí mají datový typ integer. Ještě máme paměť pro rychlé čítače. Ta je spojena s aktivací vstupů do režimu rychlého čítání.

5.1.1 Symbolická tabulka

Nejprve si nadefinujeme symbolickou tabulku. Tedy přiřazení názvů proměných, pro lepší práci s nimi.

5.1.1.1 Vstupy a výstupy

	Symbol	Address	Comment
1	Start	I0.0	Celkový start
2	Stop	I0.1	Celkový stop / Emergency
3	Manipulator_i	I0.2	Manipulator 0=beh, 1=stop
4	Conveyor_i	I0.3	Dopravník 0=beh, 1=stop
5	Optic_gate	I0.4	Optická závora
6	Palette_OK	I0.5	Palleta na miste
7	Conveyor_back	I0.6	Reverzace dopravníku
8	Position_ctrl	I0.7	Synchronyzace

Tab. 3: Symbolické proměné - vstupy

Činnost manipulátoru se zahájí stisknutím tlačítka Start. Tím se nasetuje merker M_Start. Nouzovým vypnutím Stop je M_Start resetován. V případě současného aktivního stavu má přednost Stop. Manipulátor se rozeběhne z programu, pokud nebude blokován. Taktéž motor dopravníku. Optická závora nám dává signál o přítomnosti kostky na odebírací pozici dopravníku. Paleta OK signalizuje, že je přítomna paleta, na niž budou kostky stavěny. Reverzace dopravníku je možná při zastaveném dopravníku. Změna směru je dáno přepnutím přepínače na vstupu I0.6. Poslední vstup je synchronizace. Časem bude dobré ověřit, jestli souřadnice skutečně odpovídají programovým.

Symbol	Address	Comment
PWM_0	Q0.0	PWM impulzy pro serva 1, 3, 5
PWM_1	Q0.1	PWM impulzy pro serva 2, 4, 6
A_line	Q0.2	A vstup DMX
B_line	Q0.3	B vstup DMX
Conveyor_power	Q0.4	Chod dopravníku
Conveyor_direction	Q0.5	Smer dopravníku
Power_Technology	Q0.6	Zapnutí stykacu (Serv, motoru,..)

Tab. 4: Symbolické proměné - výstupy

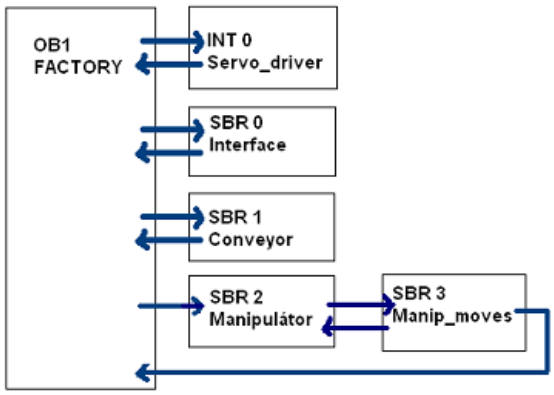
Pulzní výstupy s nastavitelnou šířkou v mikrosekundách lze nastavit dle [11]. Uvedené řešení je dosti složité. Proto jsem hledal lepší řešení. V nabídce *tools* > *Position Control Wizard* zatrhneme *Configurate the onbord PTO/PWM*. Vybereme výstup Q0.0. Nastavíme PWM a pak microseconds. Totéž zopakujeme pro Q0.1. Výstupy Q0.2 a Q0.3, jak je zmíněno v 4.3, jsou adresovací pro DMX. Výstup Q0.4 zapíná dopravník. S Q0.5 je možno ho reverzovat. Výstupem Q0.6 zapínáme „silovou“ část.

5.1.2 Vývojový diagram

Vývojový diagram strukturovaně graficky ukazuje činnost manipulátoru. Najdeme ho v příloze.

5.1.3 Hlavní program FACTORY (OB1)

V hlavním programu inicializujeme merkery. Nulujeme proměné. Nastavujeme časovače a voláme podprogramy.



Obr. 9: Volání podprogramů

Na Obr.10 vidíme nulování kroků a nastavení počátečních rychlostí pro dopravník a manipulátor. Dále inicializujeme přerušení 8 ms pomocí speciálního časovače SMB 34 s prioritou 10.

Na Obr.11 je část programu pro PWM. Bit L60.0 Je lokální a povoluje funkci. L63.7 je spuštění. PWM funkci pro Q0.0 voláme a přiřazujeme šířku intervalu a v proměné Set_Q0 nastavujeme délku trvání pulzu v logické jedna. MB 29 je byt pro číslo chyby. Pro Q0.1 používáme Set_Q1 a nastavení je obdobné.

```
Network 1 // MAIN program
// První scan
LD      SM0.1
R       S0.0, 255

Network 2
// Nastavení rychlostí
LD      SM0.1
MOVW    10, MW25
MOVW    10, MW27
Network 3
// Přerušení
LD      SM0.1
MOVB    8, SMB34
ATCH    INT0, 10
ENI
```

Obr. 10: OB1 (část a)

```
LD      M_Start:M0.0
=       L60.0
=       L63.7
LD      L60.0
CALL    PWM0_RUN:SBR1, L63.7, 8000, Set_Q0:VW22, Err_PWM_0:MB29
```

Obr. 11: OB1 (část b)

V hlavním programu použijeme tři časovače. Jeden časovač, kód je na Obr. 12, slouží pro událost přerušení. Přerušení je sice již inicializováno s SMB34. Tento byt je ovšem speciální a systémový. Z programu nemáme přístup, kromě počátečního nastavení. Podprogram přerušení Servo_driver je volán cyklicky po 8 ms po startu programu. My chceme ale události v přerušení mít ve „svých rukou“. Startovacím bitem aplikace je bit M_Start. V ntw.6 startujeme časovač T32 s 8*1 ms. Po uběhnutí nastavené doby se nasetuje pomocný byt M20.0. Je-li nasetován M20.0, nuluje se časovač. Je-li RLO časovače 0, resetuje se pomocný bit. Dále si vytvoříme hodiny 8 ms. Bit M20.2 je v prvních 4 ms časovače v 1 a pak je po zbytek času v 0. Tento signál prodloužené hrany využijeme pro čítač C0, kterým budeme řídit události v přerušení. O přerušení bude pojednáno v 5.1.3.1. Zbylé dva časovače jsou pro dopravník a manipulátor. U dopravníku hodiny prodloužené hrany nastavují délku hrany PWM a tím i rychlost. Funkce časovače pro manipulátor bude popsána v 5.1.3.5.

```
Network 6
LD      M_Start:M0.0
TON     CLK_8_ms:T32, 8

Network 7
LD      CLK_8_ms:T32
S       G_CLK:M20.0, 1

Network 8
LD      G_CLK:M20.0
R       CLK_8_ms:T32, 1

Network 9
LDN     CLK_8_ms:T32
R       G_CLK:M20.0, 1

Network 10
LDW<=   CLK_8_ms:T32, 4
=       Extended_edge_CLK:M20.2
```

Obr. 12: OB1 (část c)

Podprogramy se volají příkazem CALL. Podprogram Manipulátor je volán za podmínky, že počet kusů zakázky nebyl dosažen. Podprogram Manip_moves je volán z podprogramu Manipulátor jako funkce. Více v 5.1.3.6.

5.1.3.1 Podprogram přerušení Servo_driver

Tímto podprogramem přiřazujeme po osmi milisekundách postupně hodnoty pozic pro serva S1, S2 a S3 do Set_Q0. Pro serva S4, S5 a S6 do Set_Q1. Postupné přepínání je dáno hodnotou čítače C0 v OB1. Ten čítá od 0 do 2, tedy tři hodnoty. V přerušení vyhodnocujeme podmínku na rovnost hodnoty C0 a události, viz. ntw1. Na Q0.2 a Q0.3 nastavíme 0. Příkazem MOVW zapíšeme hodnoty pozic pro serva S1 a S2 do PWM výstupů Q0.0 a Q0.1. Setováním a resetováním Q0.2 a Q0.3 řídíme adresu přepínání pro elektroniku DMX. Postupně tímto způsobem ve 24 ms ovládáme všechny serva. O skutečné nastavení polohy se stará již elektronika serv.

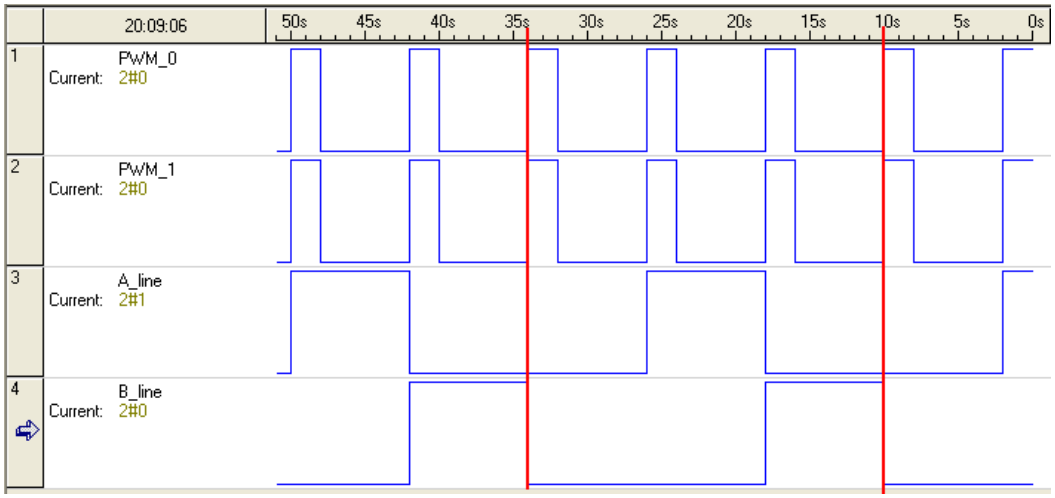
```
Network 1
LDW=    Counter_interrupts_even:C0, 0
=       Interrupt_0:M21.2

Network 2
LD      Interrupt_0:M21.2
R       A_line:Q0.2, 1
R       B_line:Q0.3, 1

Network 3
LD      Interrupt_0:M21.2
MOVW    O_Position_1:VW26, Set_Q0:VW22
MOVW    O_Position_2:VW28, Set_Q1:VW24
```

Obr. 13: Část podprogramu INT 0

Funkci programu si můžeme hned vyzkoušet. Pro ověření funkce jsem, ale transformoval časovou osu do makročasu, člověkem vnímatelného. Abychom průběhy viděli musíme ve Status Chart přidat položku Trend a do jednotlivých políček zadat proměnou. Zadání proměné je, po dvojkliku myši, do prvního okýnka. Časovou osu zadáme 1 sekundu. Proto tedy jsem změnil časovou osu. Po nahrátí programu do PLC. Spuštění do RUNu a v on-line watch (brýle) nastavením M0.0 = 1. V trendu pak vidíme průběhy zobrazené na Obr. 14.



Obr. 14: Časové průběhy výstupů PLC

Praktická zkouška na plošném spoji je jednoduchá. Místo serva zapojíme anodu LED diody na signál PWM a katodu přes rezistor na signální zem.

5.1.3.2 Podprogram Interface

Tento podprogram je rozhraním pro sdílení dat mezi vstupy PLC, kanály SCADA a řízení nastavení. Pro program jsou výstupem merkery.

První úlohou je přiřazení vstupům ekvivalenci. Př.: M0.0 = I0.0, atd. V programu používáme merkery. Důvodem je, že se může stát, že na vstupu přivede signál z čidla, které je NC místo NO. Při programování vycházíme z NO. Místo přepisování programu uděláme změnu, př.: `LDN I0.0` <enter> = `M0.0` <enter> .

Druhou úlohou a výhodou je možnost M0.0 setovat, oproti vstupu, toho v programu využijeme pro funkci tlačítka start.

Třetí úlohou je napojení kanálů SCADA a řízení přepínání režimu automat / manuál. Dále je to umožnění přístupu ze SCADA k proměným PLC, ve kterých jsou hodnoty poloh pro serva.

5.1.3.3 Podprogram Conveyor

Podprogram dopravníku má tři části: chod, reverzace a otáčky.

Chod dopravníku je následující: Po startu motoru se pohybuje pás s kostkou směrem k optické závoře. Při přerušení přijmu světla pro optotranzistor od světla LED diody kostkou, se dopravník zastaví. Nastaví se byt M15.1 Box_ready pro manipulátor. Po odejmutí kostky manipulátorem se dopravník po krátké chvíli rozjede. Zpoždění rozjezdu dopravníku je z důvodu bezpečného odebrání kostky manipulátorem. V případě „zákmitu“ optické závory by se dopravník hned rozjel, to nechceme.

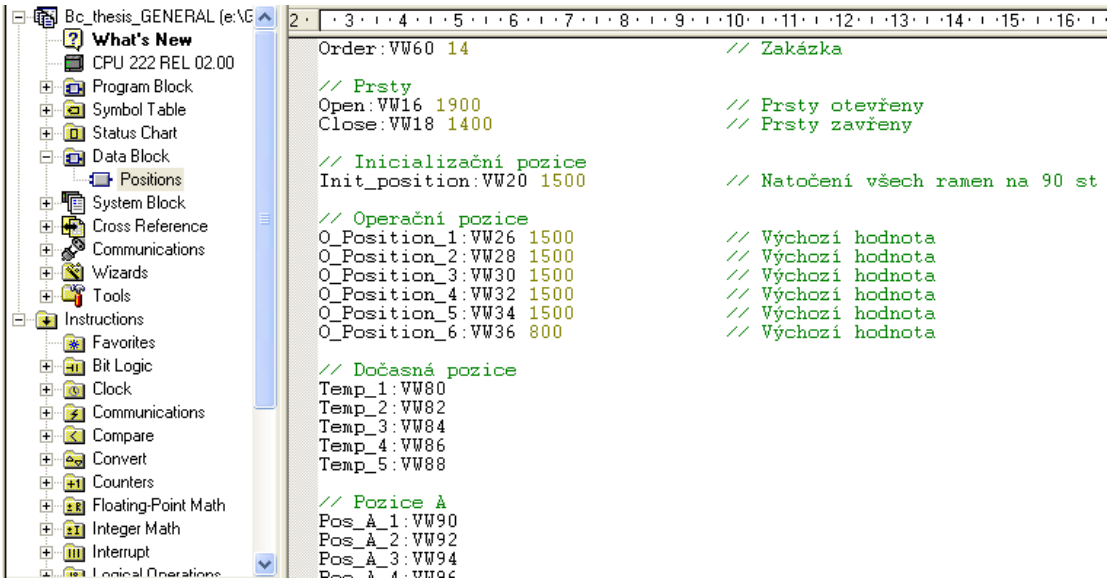
V případě potřeby máme možnost reverzace dopravníku. Motor musíme nejdříve zastavit. Hřídel motoru je brzděna dopravníkem a motor se zastaví téměř hned. Díky elegantně navržené elektronice pro dopravník (3.1) je změna směru uskutečněna nasetováním Q0.5.

Rychlost dopravníku lze měnit poměrem šířky pulzu PWM. Hodnotu pro rychlost jsme si již nadefinovali v OB1, je to B_RPM_motor. Tato hodnota je výchozí a lze ji změnit ve Status Chart MicroWinu nebo kanálem z ControlWebu. Nastavení Q0.4 je logickým součinem M_Start a RPM_motor. Je-li RPM_motor v 1, tak motor běží stále na maximum. Mění-li se hodnota bitu RPM_motor dle nastavení byte B_RPM_motor, tak na Q0.4 jsou pulzy PWM a motor má dle toho otáčky nižší.

5.1.3.4 Data Block

Paměť M slouží pro práci především s bity. Pro práci s byte, word a dword budeme používat paměť V. Výchozí hodnoty pro M jsme definovali v OB1. Výchozí, inicializační, proměné pro V nadefinujeme v datovém bloku. Zde zadáme hodnotu neutrální polohy 1 500. Stupeň natočení manipulátoru 22, jako dvojnásobek hodnoty

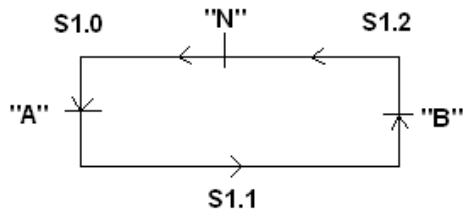
vypočtené viz. rovnice (1). Jednotlivé inicializační a koncové pozice pro manipulátor. Hodnoty Open a Close pro uchopovací prsty.



Obr.15: DB Positions

5.1.3.5 Podprogram Manipulator

V tomto podprogramu řešíme pracovní cyklus manipulátoru. Máme čítače C2 a C3. Čítač C3 počítá kostky, které jsou již přepraveny z „A“ do „B“. Účel C2 se složitější. V OB1 jsme si definovali hodnotu pro časovač T35, jako rychlost manipulátoru. Tato rychlost spočívá v tom, jak často budeme volat podprogram Manip_moves, kterým se bude postupně navyšovat hodnota aktuální operační pozice o stupeň natočení. Dle hodnoty C2, která poroste po časech T35, se bude počítat odchylka Displace od výchozí polohy. Odchylka bude vždy kladná. Směr budeme řídit tím, zda ji přičítáme, nebo odečítáme. Tak budeme postupně rozšiřovat (zužovat) délku logické 1 v PWM pulzu. Servo se bude dle toho postupně natáčet.



Obr. 16: Kroky manipulátoru

Na situačním náčrtku Obr.16 vidíme sled tří kroků. První krok S1.0 je z „N“ do „A“, druhý z „A“ do „B“ a třetí z „B“ do „N“. Po třetím kroku se nastaví Cnt_pieces, načež se přičte kostka do C2, jako doručená na místo.

Vycházíme z neutrální polohy, místa „N“. Je-li bit Box_ready v 1 a Enable_N2A_OK v 0, pak setujeme krok S1.0. Poté přesuneme pět hodnot operačních pozic pro „A“ do To příkazem BMW (v FBD: BLKMOV_W). Číslo 10 znamená počet bitů od adresy Pos_A_1 na adresu od To_1. Ve velikosti word je to pět dvoubytových slov: VW90, VW92, VW94, VW96 a VW98. Slovo pro nastavení prstů na otevřeno bude řešeno na konci vykonaného přesunu kostky.

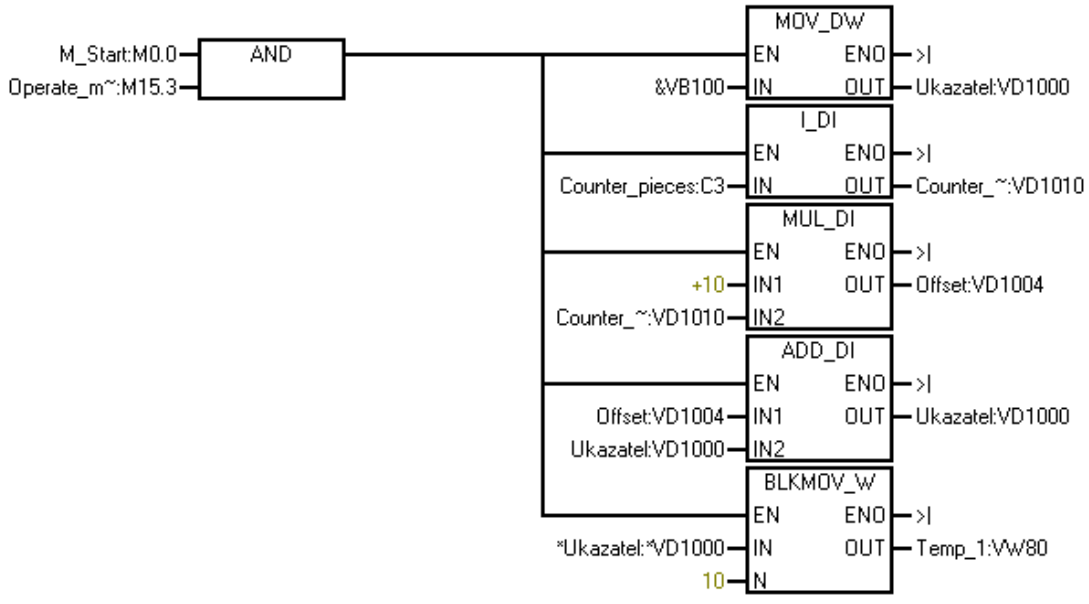
```
Network 6
LD      M_Start:M0.0
BMW     Pos_A_1:VW90, To_1:VW910, 10
MOVW    Init_position:VW20, Stand_1:VW920
MOVW    Init_position:VW20, Stand_2:VW922
MOVW    Init_position:VW20, Stand_3:VW924
MOVW    Init_position:VW20, Stand_4:VW926
MOVW    Init_position:VW20, Stand_5:VW928
```

Obr.17: Přesun pozic

Máme jednu inicializační hodnotu , ne pole. Proto 5x přesuneme hodnotu „N“ z VW20 do pole Stand. Je na snadě vytvořit inicializační pole, je to otázkou efektivity a vkusu. Význam polí To a Stand si vysvětlíme v 5.1.3.6.

Poté voláme Manip_moves, s vrácenou hodnotu Manip_moves_OK =1 zavřeme prsty přesunem slova Close do O_Position_6. Nastavíme Displace do 0, C3 do 0, Enable_N2A_OK do 1 a vyresetujeme S1.0 a nastavíme krok S1.1.

Krok S1.1 je složitější. V krocích S1.0 a S1.2 řešíme opakovaně přesuny „N“ do „A a „B“ do „N“ . Zde řešíme přesun z jednoho bloku hodnot „A“ do několika bloků hodnot – 14 – „B“. Zde bychom mohly řešit 14x postup z S1.0. To, ale asi nebude to správné. Adresy „B“ máme pěkně seřazeny v DB a odstup (Offset) je 10 bytů. A toho se dá využít!



Obr. 18: Přesun adres bloků pomocí ukazatelů

Na Obr.18 vidíme, jak se dá elegantně vyřešit indexované (od C3) krokování s využitím ukazatelů (pointerů). Oproti předchozímu případu je názornější použít reprezentaci v FBD. Logický člen AND je jasný. Je-li start a je bit operačního času v 1, vykonávají se bloky napravo. Slovo VW100 pro Pos_B1_1 (Hodnota pozice B1 pro servo S1), začíná na adrese VB100. Ampersand & značí, že jde o adresu, ne o hodnotu, která je na ní. Adresu VB100 přesuneme na ukazatel VD1000. Ukazatel je typu dword a má 4 byty. Protože je to „velký" a speciální datový typ, použijeme horní část paměťové oblasti V. Druhý blok I_DI nám konvertuje hodnotu čítače C3 na double integer. Blok MUL_DI nám vytvoří odstupy po 10 bytech. Je-li C3 nula jsme na VW100, bude-li C3 = 1, budeme na VW110, pro Pos_B2_1. Offset uložíme do VD1004. Součet ADD_DI VD1000 (ukazatele na &VB100) a offsetu se uloží zpět do VD1000. BLKMOV_W přesune počáteční adresu, na kterou ukazuje *VD1000, do bloku začínající adresou Temp_1. V Temp_1 budeme mít postupně Pos_B1_1, Pos_B2_1,...Pos_B14_1. V Temp_2 bude postupně Pos_B1_2, Pos_B2_2,… Pos_B14_2.

Pole adres Temp přesune na adresy To. V tomto případě jsou Temp mezikrok a šlo přímo jít na To. (Zachoval jsem styl tvorby kódu). Pole adres Pos_A_1 přesuneme na adresy Stand. Následuje volání Manip_moves. Dále jsou operace v tomto místě programu jako u S1.0. Na konci se přesune Open do O_Position_6 a kostka je puštěna na příslušné místo „B".

Krokem S1.2 se dostaneme do výchozí polohy „N" Programový kód je velmi podobný kroku S1.0. Na konci kroku S1.2 setujeme bit M16.3 Cnt_pieces_edge pro čítač C3. M16.3 resetujeme v S1.0, tím máme potřebnou náběžnou a sestupnou hranu pro čítač.

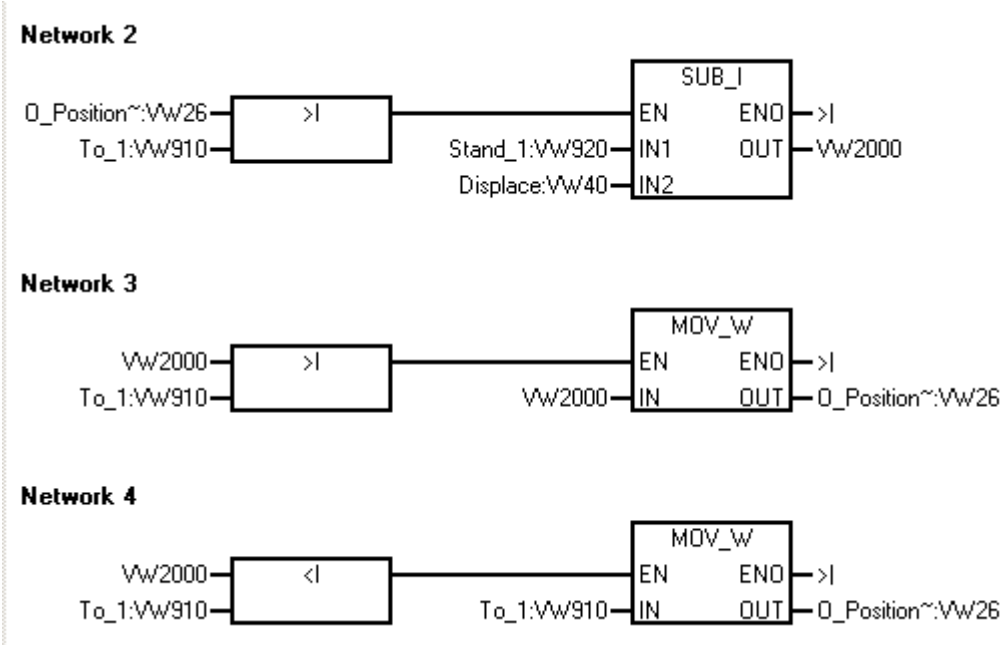
V místě „N" čekáme na Box_ready, pokud není. Jinak opakujeme kroky S1.0 až S1.2 do doby, kdy C3 = Order. Zakázka (Order) je nastavena v DB na 14.

5.1.3.6 Podprogram Manip_moves

Tento podprogram je obdobou FC v Simatic Manageru pro systémy 300. Zde řešíme plynulý pohyb po kroku, stupni natočení 2°, pro jednotlivá serva, z výchozí polohy do koncové. Zde pracujeme s pěti hodnotami operačních proměných (O_Position_1 až O_Position_5), kterými předáváme aktuální hodnotu pro Set_Q0 a Set_Q1. Dále máme pole To. To znamenají hodnoty pozic, kam jdeme. Postupně, dle kroků, jsou to pozice „A", „B [x]" a „N".

Vycházíme z operačních pozic. Počáteční hodnoty pozic jsou 1 500, ty se postupně mění dle kroků. Po třetím kroku jsou opět 1 500. Stand obsahuje pole adres, odkud jdeme. Při cestě z „N" do „A" jsou to „N". K jednotlivým „N" přičítáme odchylku Displace. Mezivýsledkem pro pozici serva S1 bude VW2000, bude to hodnota aktuální operační pozice upravená od odchylku Displace, pro přepis vstupní hodnoty operační pozice. V symbolice jazyka C : `O_Position_1 += Displace`.

Do cílové polohy můžeme jít zleva, nebo zprava, to nevíme. Explicitně pro konkrétní, jedinečný, případ – ano. To by bylo ale pro jiné zadání práce manipulátoru těžko modifikovatelné. Proto musíme řešit, jak se dostat do cíle z obou stran. To samozřejmě řešíme porovnáním - odkud `<I` kam, a odkud `>I` kam. Význam proměné VW2000 si vysvětlíme nyní. Jdeme z operační pozice O_Position_1 (na začátku „N“) do To_1 (ve které je Pos_A_1). Stojíme-li čelem před manipulátorem, je pozice Pos_A_1 nalevo a odpovídá jí příkladně 800 µs. Tedy je menší, než výchozí operační pozice 1 500 µs. Ve Stand_1 máme 1 500 µs. Od Stand_1 odečítáme, po časových intervalech 200 ms běhu programu, odchylku Displace (`Displace += Step_deg`). V kritické chvíli, blížíme-li se ke koncové hodnotě, hrozí případ : O_Position_1 je 818, nastane další porovnání s výsledkem pro novou O_Position_1 (1 500 – 32*22) = 796. Zapsáním této hodnoty do P_Position_1, bychom přešli cílovou polohu. Zprava bychom se museli vracet. Měli bychom Wind-up efekt. Použití výrazu `<=I` pro operaci `MOVW TO_1, O_Position_1` nejde! Tím bychom vyřadili směr zprava. Proto máme proměnou VW2000, kterou testujeme na podtečení. Nová vypočtená hodnota je nižší než výchozí, ale větší než cílová. VW2000 zapíšeme do O_Position_1. Případ druhý nová hodnota je menší, než cílová. Místo VW2000, se kterou bychom podtekli, zapíšeme To_1 do O_Position_1. Tím jsme nastavili zbývající část kroku natotčení k cílové pozici. Při rovnosti pozic se již žádné vyhodnocování menší / větší neprovádí.

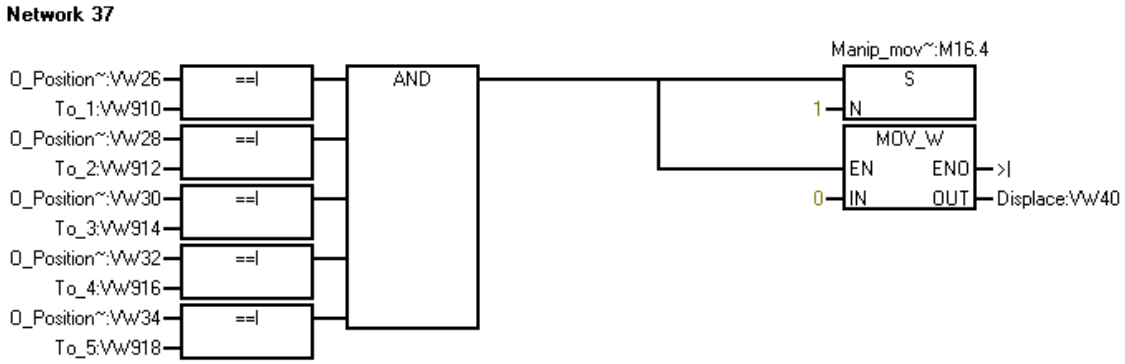


Obr.19: Část výpočtu nové hodnoty operační pozice

Jak mile tímto postupem projde všech pět operačních pozic, pak se nastaví Manip_moves_OK do 1 a do Displace se zapíše 0 (integer). Na Obr. 20 a Obr. 21 vidíme stejný kód v STL a FBD, pro srovnání.

```
Network 37
LDW=    O_Position_1:VW26, To_1:VW910
AW=     O_Position_2:VW28, To_2:VW912
AW=     O_Position_3:VW30, To_3:VW914
AW=     O_Position_4:VW32, To_4:VW916
AW=     O_Position_5:VW34, To_5:VW918
S       Manip_moves_OK:M16.4, 1
MOVW    0, Displace:VW40
```

Obr. 20: Dosažení cílové pozice (STL)



Obr. 21: Dosažení cílové pozice (FBD)

Zde se vracíme do příslušného kroku podprogramu Manipulátoru. Řešíme otevření, nebo zavření prstů. V dalším kroku, pro nové hodnoty, opět voláme Manip_moves.

Aby CPU stále nezpracovávala několikrát stejné hodnoty výpočtů, které se mění po časech T35, máme zavedený bit Operate_manip. Čas T35 nastavíme na 200 ms. Operate_manip bude aktivní prvních 20 ms pro volání Manip_moves, ve kterém se provedou výpočty minimálněkrát, nejméně vždy jednou.

Dle [3] jsem vyhledal nejdelší čas pro vykonání jedné operace. Je to operace *R, což používáme při práci s ukazateli v kroku S1.1 podprogramu Manipulator. Pro zpracování *R potřebuje CPU maximálně 166µs. Z toho vyplývá, že našich 20 ms odpovídá 120 operací *R. Časovač TON potřebuje 33µs . Za 20 ms se provede příkladně činnost 606 čítačů. Logické operace jsou v jednotkách mikrosekund. Z toho lze usuzovat, že naše kódy Manipulatoru a Manip_moves za nastavený čas stihneme. Výsledkem je ušetření výpočetní kapacity CPU, třeba pro komunikaci nebo ostatní podprogramy.

6 OPC

Aby bylo možno zpracovávat a nastavovat hodnoty pro technologii z vizualizace na PC, je potřeba mít rozhraní OPC. Pro počítače je rozhraní mezi systémem a periferiemi řešeno s OLE (Object Linking Embeded). Této myšlenky se použilo pro OLE for Proces Control, zkráceně OPC.

Každý výrobce má svůj jedinečný „balík“ pro komunikaci PLC se SCADA. Siemens používá vitualizaci WinCC, PLC Teco používá vizualizaci Reliance, apod. Je to zřejmě i marketingový tah, aby se k jejich PLC použila jejich SCADA. Tímto nekončí. Ještě vstupují do hry komunikační protokoly jako PPI, MPI, PROFIBUS, Ethernet, aj.

Výrobce špičkových PLC, ale nemusí mít špičkou SCADA, jako specializovaný dodavatel SCADA. Možnost propojit na jedné straně PLC Siemens s vizualizací ControlWeb lze speciálním ovladačem. Propojit Simatic s InTouch*em* lze též speciálně určeným ovladačem pro SW produkt Wonderware, apod. Druhá možnost je OPC server, který umí na jedné straně pracovat s různými PLC a libovolnou SCADA na straně druhé. OPC je složitou otázkou. Problematikou OPC se zabývá organizace OPC Fundation.

Máme zde pojmy jako OPC server, OPC klient a ovladač. Pojmy si probereme detailněji s návazností na náš projekt.

6.1 OPC server

OPC server je program, který komunikuje s datovými oblastmi PLC prostřednictvím kanálů. Veličiny (hodnoty požadovaných proměných), které chceme sledovat, přiřadíme kanál. Pak s nimi můžeme pracovat, jak jsme zvyklí s daty na PC pracovat.

OPC server firmy Siemens je PC Access. Tento program je možné získat od Siemens*u* na webových stránkách v sekci download. Plně funkční je 60 dní nebo do 100x spuštění aplikace.

Prostředí je podobné jako MicroWin. K PLC jsme připojení PPI kabelem, který slouží pro komunikaci počítač jako DTE s RS232 a S7 200 jako DCE s RS 485.

V levé části si založíme projekt. K projektu si přiřadíme stanice PLC. Po nahrátí stanice do PC Access si můžeme přetáhnout (systémem chytni a táhni) symbolické tabulky do pravého horního okna programu, čímž vytvoříme kanály. Kanály můžeme přetáhnout do spodního okna. Stisknutím watch (stejné ikony jako v MicroWinu) vidíme stav, resp. hodnotu naší sledované proměné. To by bylo ale trochu málo. Proto je zde služba DDE (Dynamic Data Exchange). Tato služba slouží pro komunikaci mezi SW na PC. Tímto můžeme komunikovat s MS Excelem. V Excelu můžeme při podpoře VBA (Visual Basic Aplication) vytvářet jednoduché vizualizace.

6.2 OPC klient

V předchozí kapitole jsme již vlastně jednoho OPC klienta popsali. Je to MS Excel. Klientem je i ControlWeb nebo InTouch, aj. Každý vizualizační software je OPC klientem k OPC serveru.

6.3 Ovladač SIMATIC 200 PPI driver

OPC server, jak jsme si již popsali, je univerzální prostředek. Jeho cena tomu i odpovídá. V Excelu (součást kancelářského balíku) můžeme dělat vizualizaci. Prostředí Excel známe a víme, že pohyblivé obrázky s pěknou grafikou neuděláme.

Pro tuto práci máme k dispozici ControlWeb 5, produkt Moravských přístrojů. My potřebujeme propojit Simatic 200 a ControlWeb 5 (dále jen CW). OPC by bylo drahou záležitostí.

Přímo určeným nástrojem pro konkrétní PLC a konkrétní SCADA je ovladač. Ovladač je produktem výrobce SCADA. Pro S7 200 a CW existuje ovladač SIMATIC 200 PPI driver, v současné verzi v.4.7.

Ovladač je program s vlastním instalačním souborem. Nejdříve musíme mít nainstalován CW. Instalace ovladače zajistí všechno potřebné nastavení pro CW. Nastavit komunikaci se Simaticem a definovat data, která chceme vyčítat nebo zapisovat, musíme již my. Zde s nastavováním komunikace předběhneme seznámení se s CW, který bude přiblížen v kapitole 7.

6.3.1 Přidání ovladače

Ovladač SIMATIC 200 PPI driver má v CW podobu dynamické knihovny dll. V *Nástroje > Konfigurace ovladačů...* . Se nám objeví okno správce ovladačů. Zde se již nějaké ovladače ovladače nacházejí. Vidíme mimo jiné Virtuální generátor v 5.0, Ovladač DDE klient. Virtuální generátor slouží jako simulátor dat ovladače zařízení, když nemáme zařízení (PLC) nebo ten ovladač. S DDE klientem bychom mohli zkusit komunikovat s PC Access*em*. Máme k dispozici speciální ovladač s licencí a tak ho použijeme. Stisknutím tlačítka přidat, ho přidáme. Stačí poklepat na ovladač myší a zaškrtne se také pro použití. V *Nápověda >Licence* vidíme budíky. Jeden je pro CW a druhý pro ovladač. To znamená, že máme neomezenou funkci CW a ovladače.

V záložce Datové inspektory vidíme zelenou ikonu v podobě PCI karty. Kliknutím na nápis Ovladače se vpravo zobrazí tabulka. Do ní zapíšeme název ovladače, třeba S7. Dále se po nás bude chtít soubor dmf a par.

6.3.1.1 Mapovací soubor

Než budeme definovat ovladač a jeho kanály musíme mít vytvořený mapovací soubor. Náš mapovací soubor bude manip.dmf. Je to textový soubor a lze ho prohlížet v poznámkovém bloku. Do něho nic nezapisujeme. Do něho zapíše ovladač při definici kanálů. Obsahem budou čísla kanálů a jejich datový typ.

6.3.1.2 Parametrický soubor

Než budeme definovat ovladač a jeho kanály musíme mít vytvořený parametrický soubor. Náš parametrický soubor bude manip.par. Ten obsahuje informace o typu komunikace jako master nebo slave, číslo portu COM, adresu v PPI, rychlost přenosu v Bd, paritu. Některá další nastavení není ani nutné nastavovat.

```
[S200PPI]
  ComDriver             = cwcomm.dll, COM6
  TIMEOUT               = 100
  REPEAT                = 5
  ADDRESS               = 1
  MODE                  = master
  PDU_SIZE              = 112
  INTER_MESSAGE_DELAY   = 20

[comm]
  device                 = com6
  baudrate               = 9600
  parity                 = none
  databits               = 8
  stopbits               = 1
  cts_flow               = false
  dsr_flow               = false
  dtr_control            = disable
  rts_control            = disable
  dsr_sense              = low
  rx_interchar_timeout   = 20
  rx_char_timeout        = 20
  rx_timeout             = 20
  tx_char_timeout        = 20
  tx_timeout             = 20
  rx_buffer              = 500
  tx_buffer              = 500

[BLOCKS]
  block  = 2, 10,  8,   bitset, M, 0
  block  = 2, 18,  8,   bitset, M, 10
  block  = 2, 26,  300, word,   V, 0
  block  = 2, 326, 300, word,   V, 0
```

Obr.22: Obsah parametrického souboru

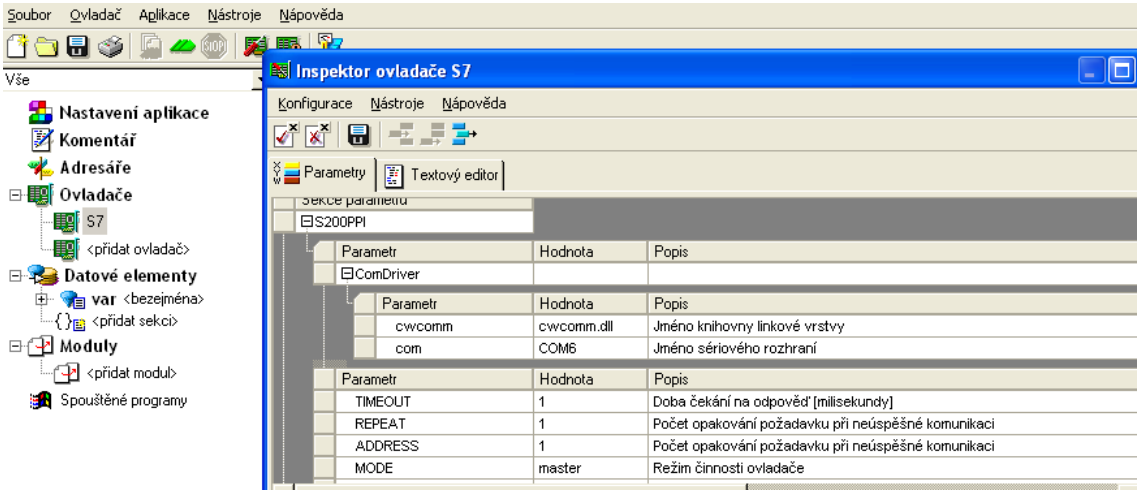
6.3.2 Nastavení ovladače.

Ovladač jsme přidali a přiřadili jsme mu i mapovací a parametrický soubor. Než však uvidíme jejich výpis, jak na Obr.22, musíme jej nastavit.

Na Obr.23 vidíme část inspektora ovladače, ve které nastavujeme komunikační port. Máme-li stolní počítač (DeskTop), je přiřazení sériového portu na COMu 1. Číslo 2 je pro paralelní port. Máme-li přenosný počítač (LapTop), nemáme v 99% fyzicky sériový port. COMy samozřejmě máme, ty jsou přiřazeny pro USB. V tom případě je našim úkolem zjistit, které USB má jaký COM. V našem případě COM pro CW je COM 6.

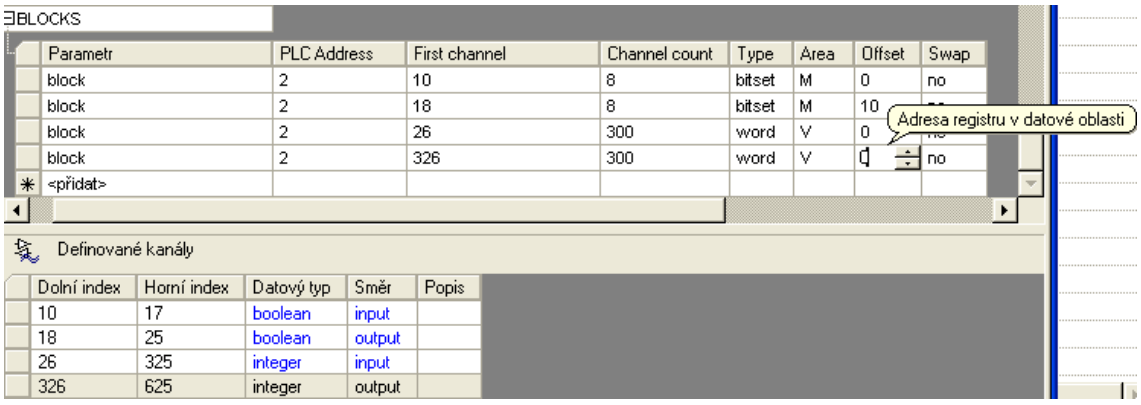
Tím tato část ještě není dokončena. Máme kabel PPI s konektorem D-SUB9M pro PLC a D-SUB9F pro PC. Tedy řešíme konverzi RS232 na USB. Běžně je k sehnání převodník i s krátkým kabelem. Převodník HL-340 se musí v PC nastavit jako USB zařízení. I když k němu dostaneme CD s ovladači, není vyhráno. Nainstalovat správný ovladač dá pěknou práci.

S přiřazením portu musíme dát pozor, že COM pro CW je jiný než COM pro MicroWin!



Obr. 23: Inspektor ovladače (část a)

Prodlevu (Timeout) pro komunikaci musíme zadat experimentálně a prakticky ověřit. Stejně tak i jedno opakování při neúspěšné komunikaci. Adresu v síti PPI (Point-to-point interface) si již zdaleka vybírat nemůžeme. Souvislost adresy je s režimem ovladače. V případě režimu slave, je vlastně jedno jaké číslo přiřadíme od 3 do 31. PLC bychom v tom případě museli nakonfigurovat v režimu FREEport, aby posílalo data do CW. Pro vyčítání dat CW od PLC volíme režim master. Simatic je mastr*em* v PPI a má adresu 2. CW může být mater*em*, ale musí mít nižší adresu. MicroWin má adresu 0 (obvykle) a pro CW zbývá už jen 1.



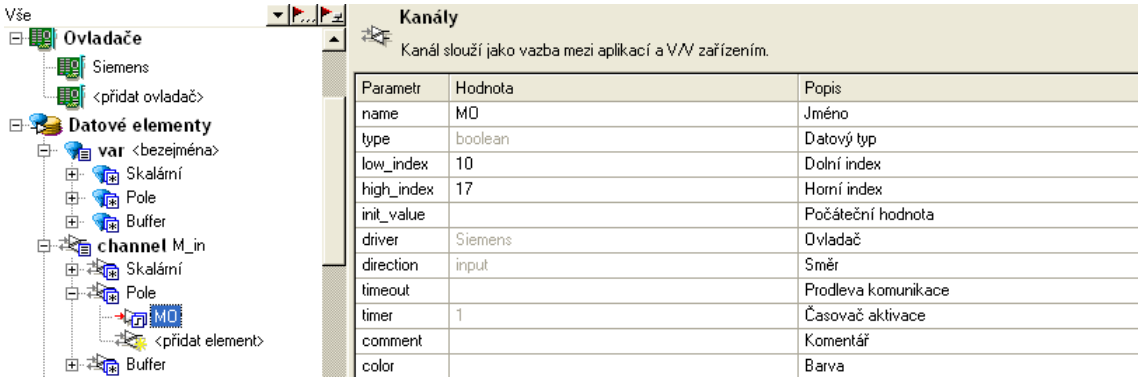
Obr.24: Inspektor ovladače (část b)

Součástí nastavení ovladače je vytvoření a definice kanálů. Proto, než jsme něco chtěli dělat s nastavením ovladače, museli jsme vytvořit soubory manip.dmf a manip.par. Prvním kanálem začínáme s číslem 10. Toto číslo není opět náhodné. Prvních 10 kanálů slouží pro přenos času, data do PLC a kontrole komunikace. Aplikace nebudeme mít v reálném čase a PLC nemáme s modulem reálného času, takže to opomineme. Kontrola komunikace nás potrápila. Chyba 2 je chyba spojení. Nejen špatné nadefinování komunikace stojí za touto chybou. Stačí změnit v systému přiřazení COMů, což může nastat i ovladečem našeho RS232 / USB převodníku.

Počet kanálů je jinými slovy šířka kanálu. Zadáme 8. Bitset znamená přístup po bitu v bytu. Příkladem je vstupní byte jako pole 8 bitů pro M0.0 až M0.7. Datová oblast, do které přistupuje je M s offsetem 0 v bytech. Tedy. Swap znamená přehodit, je určeno pro problematiku big-endien a litle-endien. Kanálu přiřadíme datový typ a směr. Typ je jasný – bool, směr bude input – do CW. Budeme vyčítat merkery vstupů PLC. Obdobně nastavíme kanál 18 pro 8 merkrů počínaje M1.0. Typ bude opět bool a směr output. Z vizualizace budeme těmito merkery zastupovat funkci osmici merkrů počínaje M0.0. Jak jsme si v rozhraní (5.1.3.2) řekli. Vizualizací tak v režimu manuál budeme řídit start / stop, reverzaci, aj. Dalšími kanály budeme mít pro čtení a zapisování proměných V. Typ máme word a začínáme na VB0. V nastavení zadáme datový typ integer a směr. Definice je pře celou oblast V, ve které máme data pro manipulátor. Při používání kanálu v 6.3.3 musíme nepoužívat některé výstupní kanály, abychom nenarušili vlastní operační pozice. Pro ty máme předávací proměné, které můžeme měnit z CW kdykoliv. Změna ale bude až při povolení zápisu do operačních proměných v ručním režimu, bez autonomního běhu programu PLC.

6.3.3 Přidání kanálů

V datových inspektorech, jak jsme přidávali ovladač, je o něco níže složka Datové elementy. Při kliknutí na tuto složku se vpravo objeví proměné CW. Jako speciální globální proměnou je kanál. V odstavci Kanály zvolíme přidat sekci. M_in bude vstupní kanál s obsahem 8 bitů. V kanálu M_in zadáme ovladač S7, smět input a čas se kterým budou z PLC vyčítány data. V kanálu si nadefinujeme proměné pro CW. Skalární proměná je jedna proměná na jeden kanál. Pole je několik proměných na kanál. Indexování je od počátečního čísla kanálu do jeho šířky. Jelikož máme nadefinovaný vstupní kanál M_in, již přiřadíme jen dolní a horní index.



Parametr	Hodnota	Popis
name	M0	Jméno
type	boolean	Datový typ
low_index	10	Dolní index
high_index	17	Horní index
init_value		Počáteční hodnota
driver	Siemens	Ovladač
direction	input	Směr
timeout		Prodleva komunikace
timer	1	Časovač aktivace
comment		Komentář
color		Barva

Obr.25: Pole proměných ve vstupním kanále

7 CONTOL WEB 5

ControlWeb 5 je produktem české firmy Moravské přístroje, která se specializuje na systém SCADA. SCADA je označení dohlížejících a řídících prostředků pro technologii a poskytuje i sběr a vyhodnocování dat. Prostředí CW je velice přátelské a ke každému kroku, který v programu uděláme, máme český komentář. V roce 2000 vyšla firemní publikace k systému ve verzi 2000 – Control Web 2000. Tato knížka je, bohužel, stěží sehnatelná už jen v technické knihovně. Novější produkt je CW6. Stále se používá CW5, který je dlouhodobě prověřen a je k němu celá řada podpor. K dispozici pro tuto práci máme verzi 5 unicode SP15.

Prostředí používá tzv. virtuální přístroje, s nimiž vytvoříme celou aplikaci. Přístroje jsou 2D i 3D. Pro vytvoření aplikace se můžeme inspirovat mnoha příklady ze složky Examples. V CW nejen uděláme aplikaci pro výměnu dat s PLC. Je zde podpora pro archivaci hodnot, alarmy. Tvorbu databáze dBase a SQL. Publikování na webu. Nastavení přístupových práv, to je zejména pro RUN-Time u koncového zákazníka. Lze vytvořit i aplikaci pro systémy reálného času.

7.1 Průvodce aplikace

Při startu prostředí se objeví průvodce nastavením nové aplikace. Hned zkraje učiníme závazné rozhodnutí, zda aplikaci budeme mít v režimu reálného času, nebo poběží jako řízená událostmi.

Reálný čas je používám, kdy z vizualizace řídíme PLC. Celý program pracuje s časovači. Přistroj, který je nečasován vlastně pro aplikaci neexistuje. V tomto režimu si musíme vše velmi dobře promyslet, protože pracujeme přímo s prostředky operačního systému.

Druhý režim je řízení aplikace událostmi. Zde je PLC autonomní a my sledujeme proces s minimem zásahů. Většinou zasahujeme v manuálním režimu do činnosti PLC. Řízení událostmi znamená, že až se něco ve virtuálním přístroji změní, pak teprve jsou aktivovány jiné přístroje, pro něž je výstup určen.

My zvolíme aplikaci volně běžící v závislou na změnách dat. Postupujeme stiskem tlačítka další. Velikost zobrazení můžeme ponechat. Panely zadáme 2. Jeden pro dopravník, druhý pro manipulátor. Přepínací záložku mezi panely si dáme nahoru.

7.2 Vizualizace

Na Panel_1 již můžeme umístňovat virtuální přístroje, ty máme v *Nástroje* > *Paleta přístrojů*. Základní dělení přístrojů je na zobrazovací a řídící. Jejich další členění je

binární, spojité a textové. Textové používat nebudeme. Co si však vytvoříme, bude místo barvy kontrolky, text. Kontrolka pro prsty OTV / ZAV. Běžně máme ikony ve smyslu zelená je OK, červená je chyba, apod. Vlastní lze vytvořit v Iconer.exe, který je v adresáři CW.

Zobrazovací binární reprezentuje žárovka, spojité - měřicí přístroj. Řídící binární přístroj je přepínač, spojitý je knoflík. Přístrojů je samozřejmě v jednotlivých kategoriích celá řada.

Abychom neumísťovali přístroje do „vzduchu“ nahrajeme si podkladový obrázek pro panel. Při poklepání na panel se nám objeví nabídka. V ní vyplníme dv_id, dv_file. Viz Obr.26. Nastavení panelu je bohaté, ostatně jako u všech přístrojů

Parametr	Hodnota	Popis
mode	normal	Způsob překreslování panelu a přístrojů do panelu vložených
icon		Soubor s ikonou pozadí
dv_id	.IMG	DataView identifikátor typu
dv_file	'DOPRAVNIK.BMP'	DataView soubor
container		DataView v kontejneru
follow_scrollbars		Přístroje budou sledovat pohyb rolovacích lišt
disable_controls		Zakázat ovládací prvky kontejneru
disable_scrollbars		Zakázat rolovací lišty kontejneru
load_on_show		Obsah DataView zaveden až při zobrazení
dither_bitmaps		Povolit ditrování barevných bitmap na pozadí panelu
allow_mouse_wheel		Povolit šíření událostí kolečka myši k přístrojům
blink_rate	normal	Frekvence blikání
blink		Podmínka blikání
minimize		Minimalizace
dv_page		Výraz nastavující stranu DataView

Obr.26: Přidání obrázku na pozadí panelu

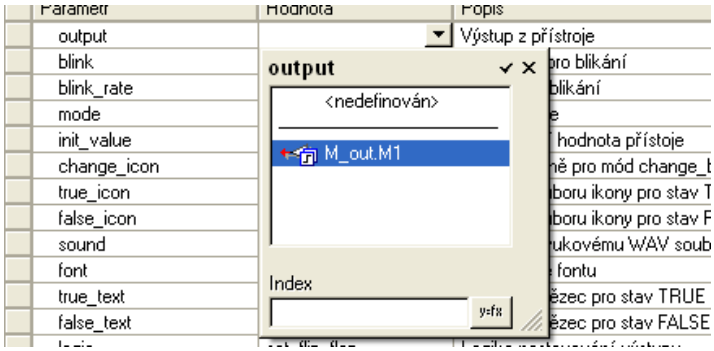
Než se pustíme do jednotlivých panelů, musíme se ještě seznámit s proměnými. S proměnou kanál jsme se již setkali v souvislosti s nastavováním ovladače do paměťových oblastí PLC. Proměná kanál je svým způsobem globální. Globální proměnou budeme používat pro předávání dat mezi přístroji. Lokální budeme používat pro přepočty a pomocné výpočty uvnitř přístroje. Systémové jsou převážně pro práci s časem a datumy. Konstanty máme také troje. V systémových máme příkladně konstantu Ludolfova čísla Pi.

U PLC jsme proměné nespecifikovali na globální a lokální. Ikdyž by to bylo vhodné vidět, která hodnota podprogramu je pro OB1 a více SBR, nebo která je jen pro svůj SBR.

7.2.1 Panel_1: Dopravník

Ve vrchní části máme tři bloky: pro přepnutí režimu z automatického na ruční, zapínání a vypínání dopravníku a změnu směru pohybu dopravníku. Změna režimu je uskutečněna až po stisku potvrzovacího tlačítka. Až po té je vyslána výstupní hodnota.

Tlačítku přiřadíme výstupní proměnou. Na Obr.27 vidíme, že se nám při kliknutí do políčka output zobrazí nabídka dostupných proměných daného typu.



Obr.27: Výstupní proměná

Po vložení proměné se ještě zobrazí hranaté závorky, do nichž zadáme 4. Je to pátý bit kanálu přistupující k bytové oblasti MB10 v PLC. M10.4 je merker, kterým přebíráme přístup k činnosti PLC. Program běží stále, s tím, že již můžeme zasahovat do činnosti manipulátoru.

Zadávání nastartování a vypnutí rovněž jako reverzaci nemůžeme mít přepínačem On-Off. Funkce musíme rozdělit na povel start(aktivní), start(neaktivní) a stop(aktivní) a stop(neaktivní). Každé tlačítko s funkcí set_flip_flop.

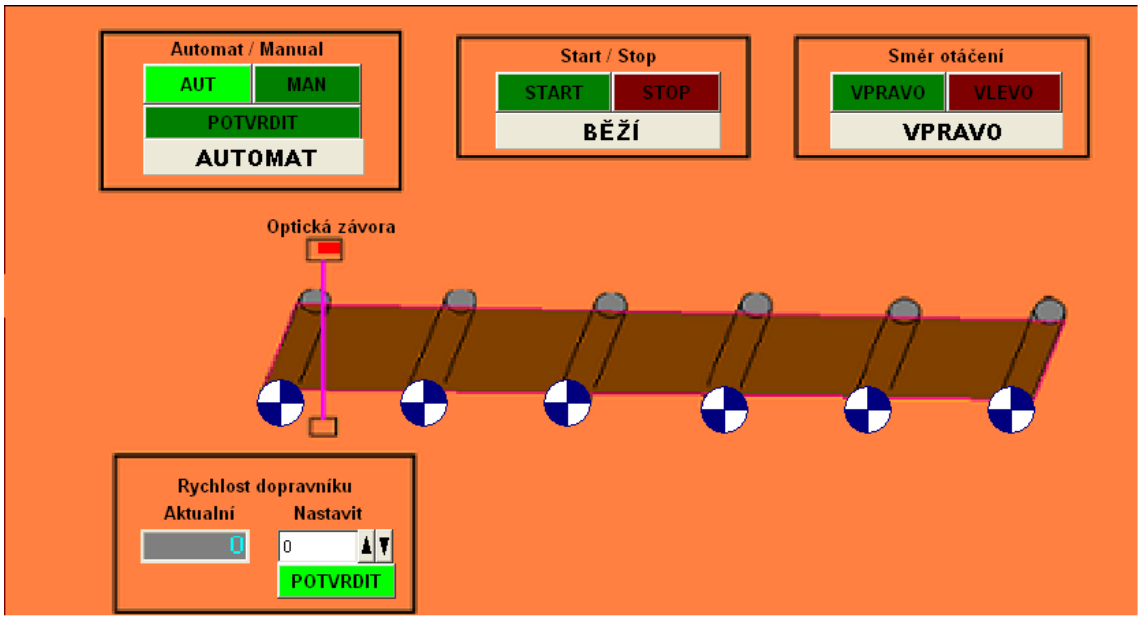
switch	switch_3	Jméno přístroje
⊞společné		Společné parametry přístroje
output		Výstup z přístroje
blink		Podmínka pro blikání
blink_rate	normal	Frekvence blikání
mode	text_button	Typ přístroje
init_value	false	Inicializační hodnota přístoje
change_icon		Cesta k ikoně pro mód change_box
true_icon		Cesta k souboru ikony pro stav TRUE
false_icon		Cesta k souboru ikony pro stav FALSE
sound		Cesta ke zvukovému WAV souboru
font	'Arial', 10, bold	Specifikace fontu
true_text	'START'	Textový řetězec pro stav TRUE
false_text	'BĚŽÍ'	Textový řetězec pro stav FALSE
logic	set_flip_flop	Logika nastavování výstupu
receivers		Seznam jmen objektů přijímajících zprávy
mono_time	6	Doba trvání aktivního stavu v sekundách
mono_level	true	Úroveň aktivního stavu
auto_update	false	Nastavení přístroje podle výstupního datového elementu bě
⊞colors		Nastavení barev
⊞blink_colors		Nastavení alternativních barev

Obr.28: Přepínač flip-flop

Vratným přepínačem – tlačítkem nastavíme bit na určitý čas. PLC tento bit převezme a nasetuje AutMan_Con. Hodnota true ve vizualizaci vyprší za 6 sekund, v našem případě. Bit AutMan_Con se po vrácení na režim automat dá tlačítky na vstupu PLC resetovat. Vizualizace bit nedrží.

Kontrolky mají v nastavení expression, místo output, tedy hodnotu nepředávají, ale reagují na její změnu. Při hodnotě true bitu přiřazené kanálové proměné optické závory je kontrolka zelená.

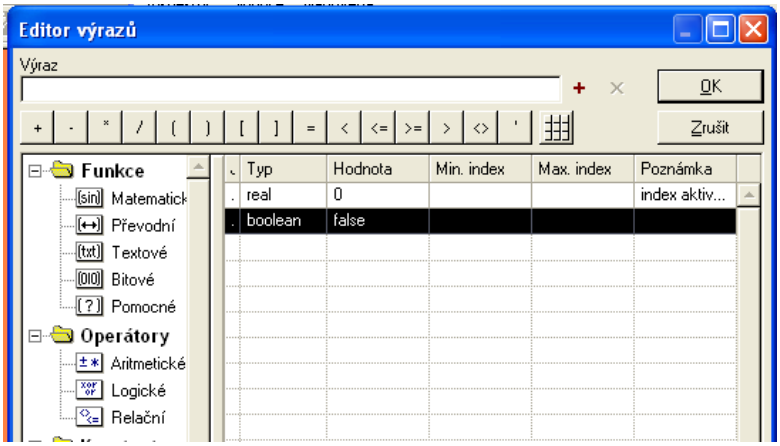
V okýnku rychlost dopravníku vidíme rychlost v inženýrských jednotkách $m*s^{-1}$. Rychlost nastavujeme také v $m*s^{-1}$. Při nastavování hodnot je třeba hlídat meze. Použitím datového typu cardinal, což je unsignet integer, zápornou rychlost nenastavíme. Maximální rychlost dopravníku je dána elektrickými a fyzikálními zákony. Náš dopravník nepoběží 80km/h, protože to není možné.



Obr.29: Panel_1 Dopravník

7.2.2 Panel_2: Manipulátor

V horní části obrazovky máme již známé tlačítka pro změnu režimu a start / stop. U jednotlivých os vidíme stupeň natočení v rozmezí od 0° po 180°.Tato hodnota není přímo vstupní do expression. Musíme ji přepočítat. Při kliknutí na ikonu y=fx se nám ukáže editor výrazů Obr.30. Je příjemné mít proměné, operátory a funkce přehledně u sebe.

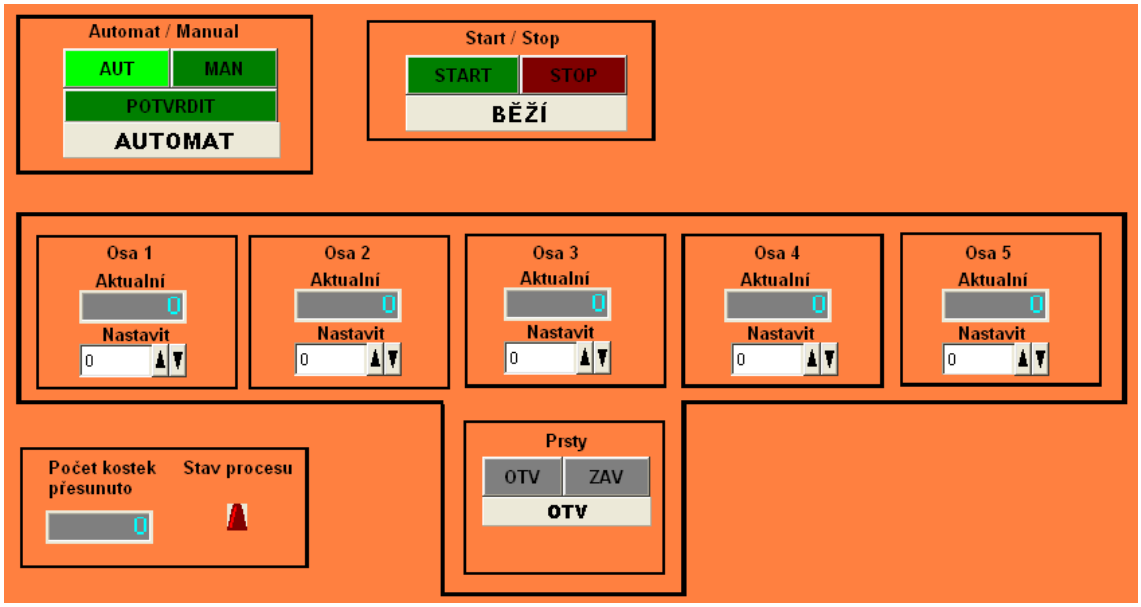


Obr. 30: Editor výrazů

Do něho zadáme přepočet. Pro osu 1 a čtenou O_Position_1 je určen výpočet (7). Pro zadávání hodnoty pro O_positon_1 z CW je určen přepočet (8).

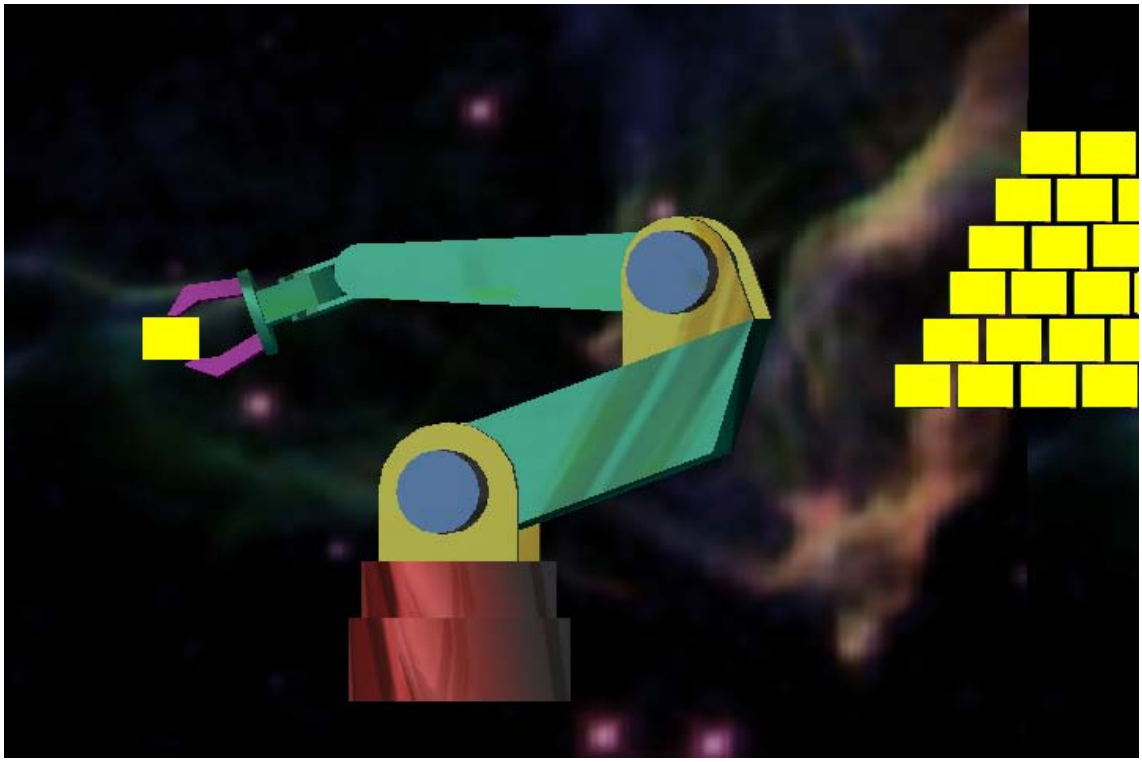
$$\frac{10*(Vin[26]-500)}{111}=OPin[1] \tag{7}$$

$$\frac{(OPout[1]*111)}{10}+500=Vout[42] \tag{8}$$



Obr.31: Panel_2 Manipulátor

Řekli jsme si, že v CW lze použít i 3D přístroje. V souboru Example je dokonce i 3D animace svářecího robota. Tato animace by se pro náš účel dala docela dobře s úpravami použít. Úkolem by bylo přidat manipulaci s kostkami. Zde, jsme již v prostorovém modelování, které je nad rámec běžné standardní vizualizace technologického procesu.



Obr.32: Ilustrační obrázek

8 ZÁVĚR

V této práci jsme navrhli způsob řízení servopohonů, vytvořili elektronické schéma a zhotovili plošný spoj. Pro PLC Simatic jsme zhotovili program ovládání servopohonů a uživatelský program, dle kterého manipulátor postaví pyramidu o základně 3*3 kostky. Program jsme strukturovali na podprogramy. Kvůli omezené kapacitě paměti PLC jsme při programování použili ukazatele, pro zjednodušení práce s polem proměných. Pohyb jsme vizualizovali v ControlWebu, v němž můžeme on-line sledovat pohyby ramen a prstů. V ručním režimu můžeme z vizualizace též pohyb manipulátoru přímo řídit. K manipulátoru jsme navrhli dopravníky, jako část navazující technologie na manipulátor.

V práci jsme použili PLC S7-200 CPU 222 DC/DC/DC, tedy s tranzistorovými výstupy. Při použití jiného typu, př. cpu 214 AC/DC/reley, který má dva komunikační porty (port 0 a port 1) a výstupy reléové, pak je potřeba řízení servopohonů realizovat sériovou komunikací z PLC v režimu FREEport. V praxi bývá často na portu 0 operátorský panel a na portu 1 SCADA klient, databázový klient, nebo systémy řízení podniku. Zde již se vracíme opět k řešení, které jsme zvolili.

Čitatel mající zkušenost s programováním Simaticu 300 ve Step 7 si jistě povšiml, že program vipadá jinak, než by očekával. Je to tím, že program Step 7 MicroWin a Step 7 Simatic Manager se od sebe liší. Proto společnost Siemens zahájila projekt Total Integrated Automation, ve kterém bude jeden programovací styl a kompatibilita se všemi svými produkty nabízející pro Automatizační technologie.

Ve vizualizaci lze pokračovat v 3D designu modelu manipulátoru. Určitě bude přínosné nahradit jednotlivých pět souřadnic přímo jednou xyz souřadnicí. Pro tuto metodu lze využít Denavit-Hartenbergovu transformaci souřadnic. Dále vytvoření databáze pozic pro manipulátor a jejich archivace. Pokračovat se dá připojením vizualizace na internet a přístupem k ní přes web.

Literatura

[1] B. Chvála, R. Maticka, J. Talácko: Prumyslové roboty a manipulátory
[2] MM Spektrum 11/2005
[3] S7-200cz - Manuál Siemens v pdf
[4] Beta - Montážní manuál Merkur v pdf
[6] Vigor VS 18 MB – Datasheet výrobce v pdf.
[7] Hitech HS 311 – Datasheet výrobce v pdf
[8] ÚPRAVY MODELÁŘSKÝCH SERVOMECHANIZMŮ [online: www.robotika.cz].
[9] S7 200 CPU 222 – Referenční příručka v pdf
[10] P.Jedlička: TTL7400

Seznam obrázků

Seznam tabulek

Seznam zkratek

CW	Control Web – vizualizační software
NC	Normaly close – spínač ve výchozím stavu sepnutý
NO	Normaly open – spínač ve výchozím stavu rozepnutý
OPC	Object Linking Embeded for Proces Control – softwarový přístup k hardwaru řídícího technologický proces
PLC	Programable Logic Controler – programovatelní automat
PPI	Potint to point interface – sběrnice typu master – slave
Simatic	Siemes Automatic – označení PLC
SCADA	Supervision Control and Data Acquisition – dohlížecí a řídící software se sběrem dat

Seznam příloh

Příloha 1. Vývojový diagram..
Příloha 2. CD
- Manipulator.mwp
- Manipulator.cwu
- DMX.ms10
- DMX.brd
- Dopravmik.ms10
- S7 200cz.pdf
- S7 200 CPU 222.pdf
- Beta.pdf
- VS 18 MB.pdf
- HS 311.pdf

Příloha 1: Vývojový diagram činnosti manipulátoru