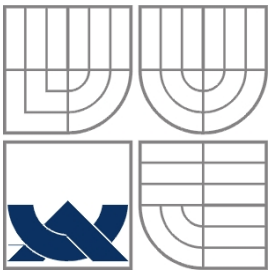


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

VYTVOŘENÍ VÝUKOVÝCH PODKLADŮ PRO PRÁCI VE VÝVOJOVÉM PROSTŘEDÍ PC WORX

TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ MICHALČÍK**
AUTHOR

VEDOUCÍ PRÁCE **ING. TOMÁŠ MARADA, PH.D.**
SUPERVISOR

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2007/08

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Michalčík Jiří

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Vytvoření výukových podkladů pro práci ve vývojovém prostředí PC WorX

v anglickém jazyce:

Teaching documents creation for work in the development system PC WorX

Stručná charakteristika problematiky úkolu:

Cílem práce je seznámit se s vývojovým prostředím PC WorX a vytvořit podklady použitelné pro výuku programování programovatelných automatů v tomto prostředí.

Cíle bakalářské práce:

1. Seznamte se s vývojovým prostředím PC WorX.
2. Seznamte se s programovatelnými automaty fy Phoenix Contact.
3. Proveďte popis vývojového prostředí PC WorX tak, aby byl použitelný pro výukové účely.
4. Vytvořte ukázkové úlohy s programovatelnými automaty fy Phoenix Contact. Tyto úlohy důkladně popište.

Seznam odborné literatury:

[1] Šmejkal, L., Martinásková, M., PLC a automatizace, Praha: BEN, 1999.
[2] Firemní materiály o programovatelných automatech fy Phoenix Contact.

Vedoucí bakalářské práce: Ing. Tomáš Marada, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2007/08.

V Brně, dne 11. 12. 2007

L.S.

doc. RNDr. Ing. Miloš Šeda, Ph.D.
Ředitel ústavu

doc. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

Licenční smlouva

Abstrakt

V této práci je proveden popis vývojového prostředí PC WorX fy Phoenix Contact , který bude použitelný jako výukový podklad pro podporu výuky. Součástí práce je soubor několika praktických úloh rozdílné obtížnosti, které byly vytvořeny pro praktické osvojení teoretických znalostí v oblasti programovatelných automatů. Tyto úlohy byly realizovány na programovatelném automatu ILC 150 ETH od fy Phoenix Contact, které jsou ve větším počtu zastoupeny v laboratoři logického řízení na Ústavu automatizace a informatiky Fakulty strojního inženýrství Vysokého učení technického v Brně.

Abstract

Thesis contains a description of a development system PC Worx by Phoenix Contact. This system was used as the educational support system. Document includes a set of practical examples with a different complexity. The purpose of examples is to improve the knowledge in programmable logic controller handling. In this case the model ILC 150 ETH (made by Phoenix Contact) was applied. This model is in common use at Institute of Automation and Computer Science laboratories.

Klíčová slova

PLC, Phoenix Contact, PC WorX, ILC 150 ETH Starter Kit, IEC 61131-3

Keywords

PLC, Phoenix Contact, PC WorX, ILC 150 ETH Starter Kit, IEC 61131-3

Poděkování

Na tomto místě bych chtěl poděkovat všem, kteří mi byli nápomocni při zhotovování bakalářské práce. Především děkuji vedoucímu mé bakalářské práce Ing. Tomášovi Maradovi, Ph.D. za cenné rady a připomínky.

Obsah

1 Úvod

Automatizační technika v poslední době prochází bouřlivým vývojem součástek a prostředků pro vývoj a tvorbu aplikací. Tento vývoj je úzce spjat s vývojem počítačové techniky a její pronikání do oblastí automatizace. Počítače se dnes již běžně využívají pro tvorbu aplikací a také samotná automatizační zařízení stále více začínají připomínat počítače. V současné době je kladen důraz na spolehlivost, rychlost, přesnost a hlavně komunikaci a tak je pevná logika na ústupu a nahrazují ji stále častěji programovatelné automaty, které již dávno nejsou symbolem drahých výrobních systémů. Stále častěji se setkáváme s těmito zařízeními i tam, kde bychom si je před pár lety jen těžko mohli představit. Podstatný vliv na tento rozvoj má i cena, která má klesající tendenci oproti minulým letem.

Programovatelné automaty se označují zkratkou PLC (Programmable Logic Controller) nebo SPS (Speicherprogrammierbare Steuerung). Jedná se o řídící systémy, které jsou přizpůsobeny pro řízení průmyslových a technologických procesů. Úlohy logického typu dnes obstarávají převážně levné a jednoduché modely.Lepší a dražší modely dokáží často mnohem více. Mohou zajišťovat sběr a vyhodnocování dat v širokém spektru a dokáží např. nahradit PID regulátory a jiné analogové součástky.

Mezi hlavní výhody těchto zařízení se řadí především jejich extrémní spolehlivost i v drsných průmyslových podmínkách, kde by klasické počítače jen těžko obstály, jsou odolné proti rušením i poruchám a často jsou vybaveny vnitřními diagnostickými funkcemi, které kontrolují činnost systému a včas dokáží lokalizovat případnou závadu nebo ji dokonce odstranit. Další výhodou je nesporně možnost postupného vylepšování programu nebo jeho případné modifikace často za chodu programu. S pevnou logikou toto není možné a neobejde se to bez finančních ztrát.

V současné době se konstrukčně dají PLC rozdělit jako kompaktní nebo modulární. Kompaktní PLC se vyznačují omezenou variabilitou v konfiguraci. Mají omezený počet vstupů a výstupů. Tato vlastnost se však pozitivně odráží v jejich konečné ceně. Nejmenší a nejlevnější modely se označují jako mikro PLC a jejich nasazení se vyplatí již při prostých úlohách, kdy nahradí několik relé či stykačů.

Modulární PLC nabízí nesrovnatelně vetší volnost ve volbě konfigurace. Tato obrovská výhoda je ovšem vykoupena i velmi vysokou cenou, která záleží na konečné konfiguraci. Rozšíření je realizováno pomocí sběrnic a moduly tak mohou být připojeny i na vzdálenosti stovek metrů.Aktuální je především rozvoj komunikace, kde se stále více objevuje ethernetové rozhraní nebo jeho průmyslová verze, která dává PLC netušené možnosti.Za zmínku stojí i rozvoj komunikace pomocí bezdrátových technologií. Modulů existuje celá řada a nezůstalo jen u digitálních nebo analogových typů. Nabídka je v současné době velmi široká a liší se podle výrobců. Mezi nejvýznamnější hráče na trhu patří bezesporu firmy jako Siemens, Phoenix Contact nebo Mitsubishi. [1].

2 ILC 150 ETH Starter Kit

2.1 Úvod

ILC 150 ETH v době svého uvedení na trh rozšiřoval řadu nejnižší řadu PLC od fy Phoenix Contact. Avšak jeho výkon a především ethernetové rozhraní dokázalo konkurovat dražším modelům od fy Siemens.

Starter Kit je souprava ILC 150 ETH včetně modulů analogových vstupů, výstupů, potenciometru a modulu přepínačů. Vše je umístěno na desce včetně zdroje, který se připojuje do el. sítě 230 V. Datová komunikace mezi PLC a PC je zprostředkována STP kabelem. Na obr. 1 je kompletně dodávaná sada.

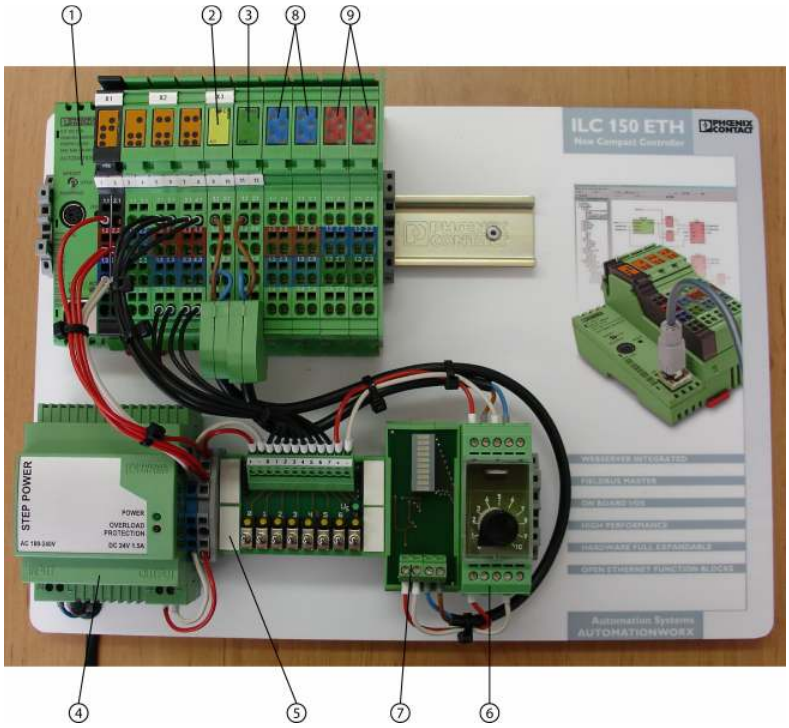
Obr.1: Starter Kit [4]

Hlavní součásti Starter Kitu ILC 150 ETH jsou uvedeny v tabulce(Obr.2)

Název	Označení	Pozice
Inline kontroler	ILC 150	1
Inline modul s jedním analogovým výstupem	IB IL AO 1/U/SF-PAC	2
Inline modul se dvěma analogovými vstupy	IB IL AI 2/SF-ME	3
Zdroj	STEP-PS-100-240AC/24DC/1.5	4
Modul přepínačů	UM 45-IB-DI/SIM8	5
Potenciometr	EMG 30-SPK-10K LIN	6
Bargraf		7
Inline modul se čtyřmi digitálními vstupy	IB IL 24 DI 4-ME	8
Inline modul se čtyřmi digitálními výstupy	IB IL 24 DO 4-ME	9

Obr.2: Položky Starter Kitu [4]

Školní verze Starter Kitu byly doplněny o moduly se čtyřmi digitálními vstupy (8) a čtyřmi digitálními výstupy (9). Od každého druhu 2 kusy.

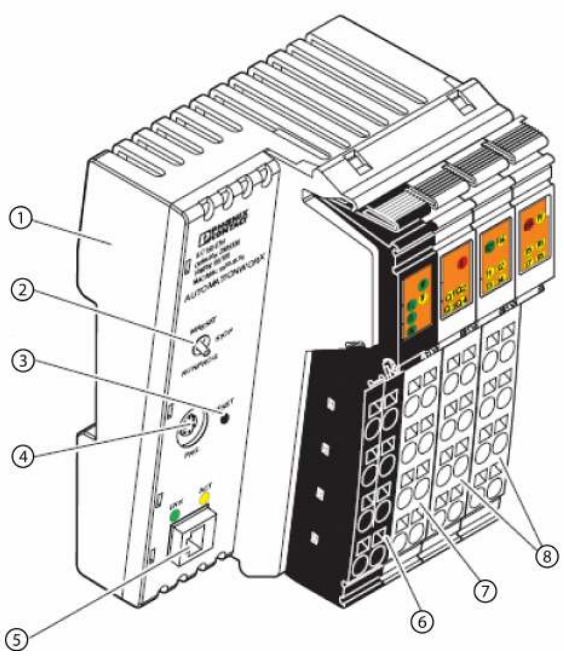


Obr.3: Starter Kit ILC 150 ETH

2.2 ILC 150 ETH

Toto zařízení umožňuje nejen efektivně a rychle řídit malé celky, ale také zapojit je do integrovaného systému řízení výrobního úseku nebo podniku. Stejně jako u ostatních komponent Inline, i u ILC 150 ETH je možné s využitím softwaru PC Worx nastavovat parametry, psát program v souladu s normou IEC 61131, předávat data prostřednictvím OPC serveru a komunikovat s ostatními zařízeními pomocí protokolu TCP/IP. [2]

Na obr.4 je samotný automat, kde můžeme vidět jeho nejdůležitější části. Kryt(1) chrání veškerou elektroniku včetně procesoru. Třípolohový přepínač(2) umožňuje přepínat automat do stavů RUN/STOP/RESET. Tento přepínač je možné ovládat softwarově přímo v PC WorXu, přičemž přepínač zůstává stále ve stejné poloze a nedochází k jeho opotřebování a následné destrukci. Taktéž je to výhodné, pokud se provádí např. úprava programu a automat není fyzicky dostupný. Tlačítko(3) pro „tvrdý“ RESET se nalézá vedle konektoru PS/2(4) pro připojení kabelu(RS-232). Na dostupném místě se nalézá ethernetový konektor(5)



Obr.4: ILC 150 ETH [4]

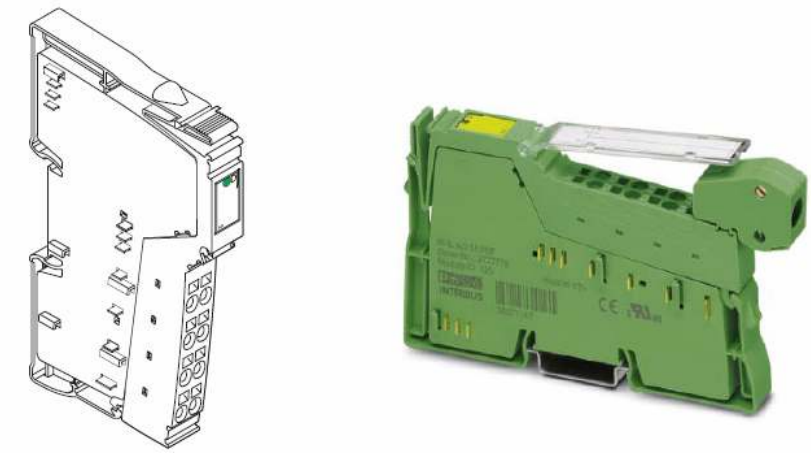
Konektor 1(6) slouží pro připojení zdroje. Konektor 2 (7) slouží pro připojení výstupů. Konektory 3 a 4 (oba 8) slouží pro připojení vstupů. Na každém z konektorů a u ethernetového konektoru je několik diod s označením. Slouží pro indikaci stavu komunikace, zdroje, diagnostiky kontroleru a vstupů/výstupů.

Specifikace:

- Programovací systém: PC WorX 5
- Diagnostický program: Diag+ od verze 1.14
- rychlost: 150 ms pro 1K instrukce
- nejkratší časový cyklus pro cyklické úlohy: 1ms
- programová paměť: 256 KB
- datová paměť: 256 KB
- rozměry: 80x122x71,5 [mm]
- zdroj: 24V
- digitální vstupy:8
- digitální výstupy:4
- komunikace: RS-232-C(PS/2) , Ethernet 10/100(RJ-45)

2.3 IB IL AO 1/U/SF-PAC

Moduly analogových výstupů jsou používány v aplikacích, ve kterých jsou adresovány analogové akční členy. S těmito terminály mohou být často rozsahy výstupních napětí a proudů individuálně konfigurovány. Analogové signály jsou generovány s rozlišením 16 bitů. [4]



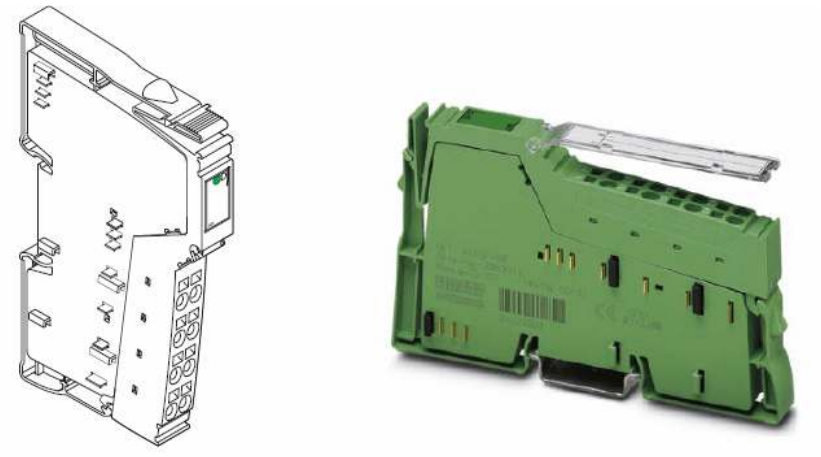
Obr.5: IB IL AO 1/U/SF-PAC [4]

Specifikace:

- Rozhraní: Inline local bus
- Přenosová rychlost:500 kBit/s
- Vedení vyrobeno z mědi
- Počet výstupů: 1
- Rozlišení: 16 bitů
- Rozsah výstupního napětí: 0..10 V
- Rychlost D/A konverze: <100 ms
- Základní chybový limit: 0.05%

2.4 IB IL AI 2/SF-ME

Blok je vyroben pro použití v Inline stanicích. Jsou používány pro získaní analogových napěťových a proudových signálů. Hodnoty lze získat s rozlišením 12 bitů.[4]



Obr.6:

Obr.7: IB IL AI 2/SF-ME[4]

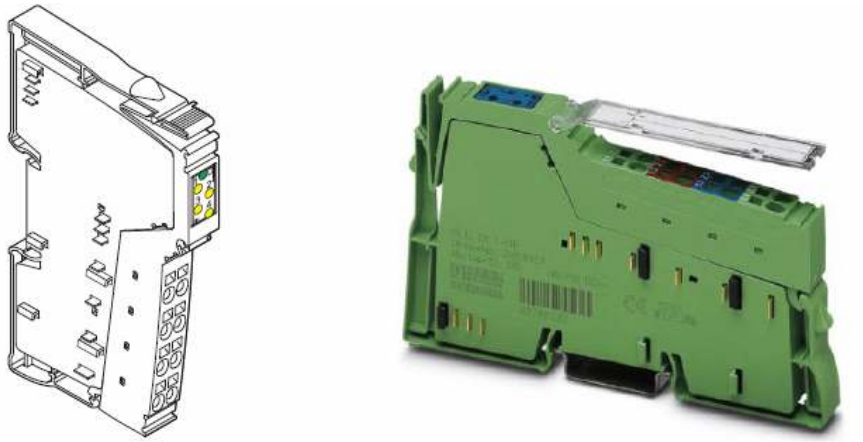
Specifikace:

- Rozhraní: Inline local bus
- Přenosová rychlost:500 kBit/s
- Vedení vyrobeno z mědi

- Počet vstupů: 2
- Rozlišení: 12 bitů
- Rozsah vstupních napětí: -10..+10 V(±10V) , 0 V ... 10 V
- Rozsah vstupních proudů: 0 mA ... 20 mA, 4 mA ... 20 mA , -20 mA ... 20 mA
- Rychlost A/D konverze: <120 µs
- Základní chybový limit: 0.3%

2.5 IB IL 24 DI 4-ME

Inline modul digitálních vstupů ve verzi ME(Machine Edition) s metodou 3 vodičového připojení. Umožňuje připojení pomocí 2 a 3-vodičové metody.. Lze připojit maximálně 4 digitální senzory. Je vybaven zkratovou ochranou a ochranou proti přetížení výstupů. Obsahuje diagnostické a stavové indikátory. [4]



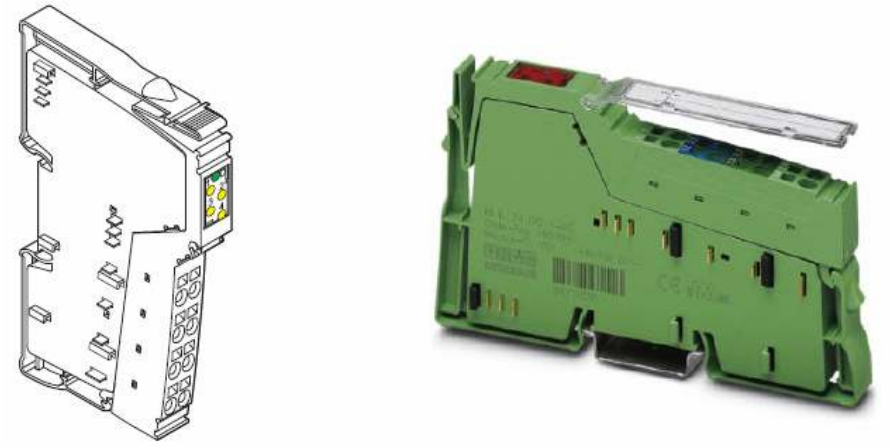
Obr.8: IB IL 24 DI 4-ME [4]

Specifikace:

- Rozhraní: local bus
- Přenosová rychlost:500 kbaud
- Počet vstupů: 4
- Vstupní napětí: 24 V DC
- Reakční doba <1ms
- Maximální přípustná zátěž na senzor: 250 mA
- Maximální přípustná zátěž na modul: 1 A

2.6 IB IL 24 DO 4-ME

Inline modul digitálních výstupů ve verzi ME (Machine Edition). Umožňuje připojení pomocí 2 a 3-vodičové metody. Lze připojit maximálně 4 digitální akční členy. Je vybaven zkratovou ochranou a ochranou proti přetížení výstupů. Obsahuje diagnostické a stavové indikátory. [4]



Obr.9: IB IL 24 DO 4-ME[4]

Specifikace:

- Rozhraní: local bus
- Přenosová rychlost:500 kbaud
- Počet výstupů: 4
- Maximální výstupní proud na 1 výstup: 500 mA
- Maximální výstupní proud na modul : 2 A

3 PC WorX

3.1 Úvod

PC WorX je vývojový software pro všechny programovatelné automaty(dále jen PLC) firmy Phoenix Contact - obecně použitelný pro programování a otevřený v komunikaci. Kromě konfigurace sběrnice INTERBUS lze komfortně konstruovat, budovat a uvádět do provozu i systémy vstupů / výstupů PROFINET IO.

Vlastnosti:

- Programování dle IEC 61131-3 (viz. Kapitola 4.8)
- Konfigurace sítí INTERBUS a PROFINET
- Zadávání parametrů zařízení a přístrojů INTERBUS a PROFINET
- Diagnostika sítí INTERBUS a PROFINET

Je doporučená verze PC WorX 5.10 a vyšší se Servis Packem 2. V této práci bude popsána demoverze, která obsahuje MSFC (Machine Sequential Function Chart) kompilátor a je omezena na vstupní a výstupní informaci o velikosti 8 bytů.

PC WorX je součástí balíku programů, který se označuje jako AUTOMATIONWORX. [4]

3.1.1 Programovací moduly

Firma Phoenix Contact poskytuje k dispozici velký počet funkčních modulů, které snižují nároky na programování a tím přispívají k tomu, že se při programování zařízení šetří čas i náklady. Kromě toho se použitím modulů podrobených rozsáhlému testování vyloučí chyby v programování.

Tyto funkční moduly jsou napsány speciálně pro programovací nástroj PC WORX dle IEC 61131 a v souhrnné formě jsou k dispozici na CD se softwarovou knihovnou CD AUTOMATIONWORX Software Library. [4]

3.2 Systémové požadavky

3.2.1 Podporované operační sytémy

- -Microsoft Windows 2000
- -Microsoft Windows XP (doporučeno)

3.2.2 Hardwarové požadavky

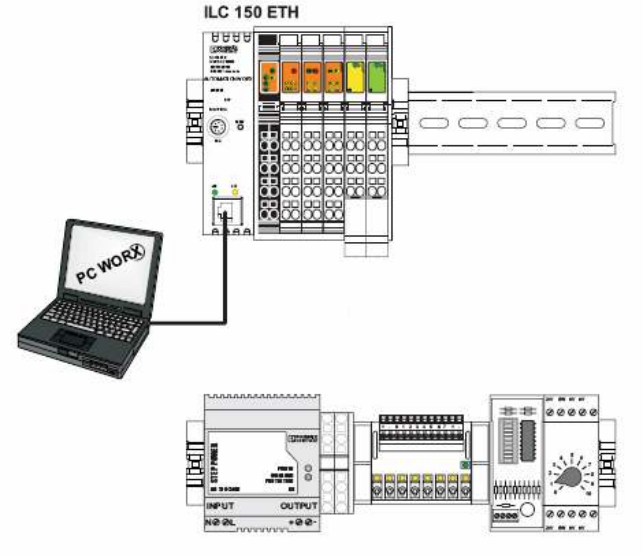
- CPU: Pentium III 800 MHz , 1 GHz (doporučeno)
- Operační paměť : 128 MB , 256 MB (doporučeno)
- Místo na Hard disku: 500 MB volných
- CD ROM mechanika : ano
- Rozhraní : sériové(1x), Ethernet
- Monitor: SVGA , rozlišení 800x600 pixelů (minimální), 1024x768 (doporučeno)
- Periferie : Klávesnice, myš
- Verze firmware ILC 150 ETH : 2.00 nebo vyšší

3.2.3 Hardwarová instalace

Starterkit ILC 150 ETH disponuje konektorem pro připojení stíněného(STP) kabelu s koncovkou RJ-45. Tento kabel je součástí starterkitu. Tímto kabelem se propojí starterkit s PC nebo notebookem, na kterém je nainstalován v požadované verzi software PC WorX.PC nebo notebook musí být vybaven příslušným typem síťové karty. Starterkit se zapojí do elektrické sítě 230 V.

Další možností je nepřipojit ILC 150 ETH přímo, ale vzdáleně pomocí průmyslového přepínače, lze tak docílit ovládání z libovolného místa na Zemi, které je připojeno k síti Internet.

Těmito jednoduchými úkony je hardwarové připojení dokončeno a zbývá jen spustit program PC WorX a začít s konfigurací PLC a vlastním programováním.



Obr.10:Zapojení Starterkitu ILC 150 ETH [4]

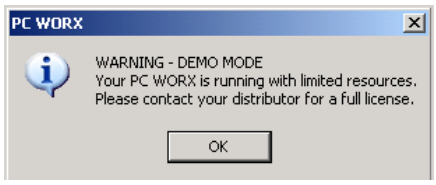
4 Vývojové prostředí PC WorX

4.1 Úvod

Tato část je věnována popisu vývojového prostředí a také náležitostí, které vedou k vytvoření funkčního programu. Budou zde probrány pouze nezbytně nutné součásti tohoto rozsáhlého prostředí s ohledem na tvorbu jednoduchých praktických úloh.

4.2 Úvodní obrazovka

Po úspěšné instalaci software a jeho SP nám nic nebrání v jeho spuštění. Na pracovní ploše se nalézá zástupce, pomocí kterého spustíme program. Ihned po jeho spuštění se objeví hlášení (obr.11) o tom, že se jedná o demoverzi, která je limitována. Pro školní použití však plně postačuje i tato demoverze.



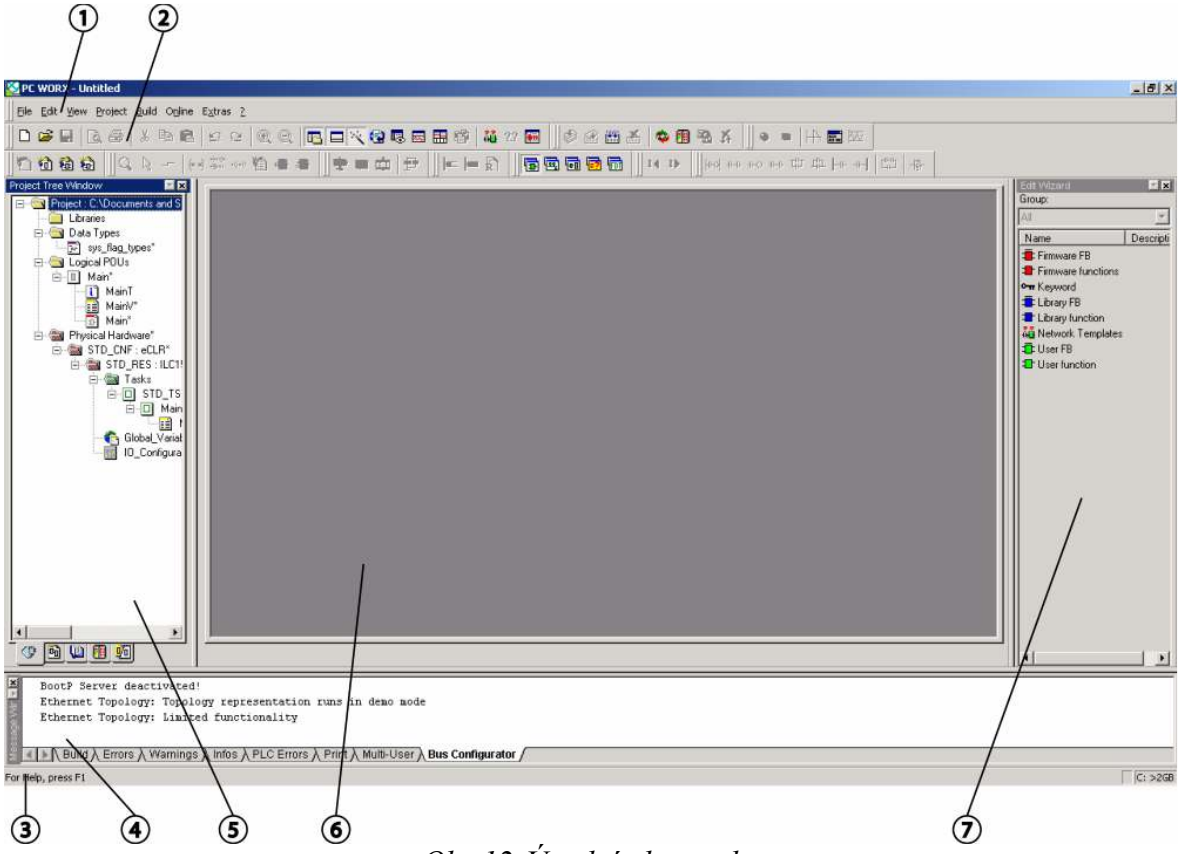
Obr.11:Demoverze

Pro pokračování chodu programu je nutné kliknout na tlačítko „OK“ jinak se program nespustí.

S velkou pravděpodobností bude v době výuky nainstalována plná verze a to znamená, že odpadne předchozí krok, který při častém spouštění programu může uživatele obtěžovat.

Po chvilce , která závisí na rychlosti PC, se program spustí a objeví se úvodní obrazovka (obr.12). Vzhled programu působí velmi přehledně a je strukturován podobně jako většina vývojových programů jako je např. AutoCAD nebo Autodesk Inventor.

Pod pozicí 1 je hlavní menu programu. Obsahuje 8 položek včetně nápovědy. Položka „File“ slouží k manipulaci s projektem jako takovým. Pomocí „File“ se projekt ukládá, zakládá, maže, exportuje, importuje a také lze vytvářet a smazat šablony. Dále hlavní menu obsahuje „Edit“, který dle názvu umožňuje editace v projektu jako je např. kopírování, vkládání, vyjmutí nebo pohyb kroků vpřed a vzad. Položka „View“ pak ovlivňuje celkový vzhled programu, respektive umožňuje nastavit zobrazování částí, které budeme pravděpodobně při tvorbě projektu využívat více a naopak součásti, které nebudeme potřebovat můžeme skrýt. Záložka „Project“ souvisí se samotným projektem, ale její použití je na vyšší uživatelské úrovni a proto se vkládání knihoven apod. nebudeme věnovat. Velice důležitou záložkou v menu je „Build“, kterou budeme hojně využívat při dokončování projektu, více v kapitole 4.11. Dále pak menu obsahuje ještě položky „Online“, „Extra“ a již zmiňovanou nápovědu (symbol otazníku).



Obr.12: Úvodní obrazovka

Pozici 2 na obr. 3 zaujímají lišty s nejčastěji používanými ikonami, aby se usnadnila a urychlila práce a minimalizoval se vstup do menu. Nejdůležitější a nejpoužívanější ikony budou probrány v jednotlivých kapitolách podle svých funkcí.

Pozice 3 je „Statusbar“, který usnadňuje orientaci v programu. Toho se snaží dosáhnout tím, že spolupracuje s ukazatelem myši a zobrazuje popis místa, kde se ukazatel myši aktuálně nalézá. Při delším setrvání ukazatele myši např. na ikoně se zobrazí krátký popis.

Pozice 4 je „Message Window“, které se velmi hodí hlavně při kompilaci a ladění projektu. Obsahuje několik záložek z nichž nejvíce využijeme standardně zobrazenou „Build“ a „Errors“ a „Warnings“.

Pozice 5 je „Project Tree Window“. Toto okno je velice důležité a umožňuje pohyb v projektu. Typickým příkladem může být přepnutí mezi pracovní plochou(pozice 6), kde se programuje a tabulkou proměnných, která se zobrazí na pracovní plochu nebo přepínání mezi podprogramy a hlavním programem.

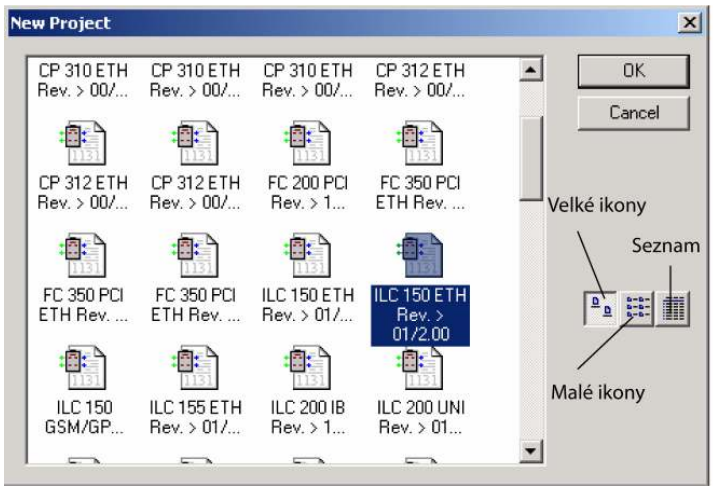
Pozice 6 je okno „Main Window“, které se využívá jako pracovní plocha, jak již bylo zmíněno výše. Při pohybu ve stromu projektu se pod „Main Window“ vytvářejí záložky právě navštívených položek ve stromu projektu, které velice usnadňují práci. Přepínání mezi programem a tabulkou proměnných je tak velice rychlé.

Pod pozicí 7 najdeme „Edit Wizard“ okno , které umožňuje vkládat funkční bloky a umožňuje spravovat nově vytvořené bloky či doinstalované knihovny bloků. Bloky jsou zde členěny podle skupin, což je opět krok k usnadnění práce za předpokladu, že uživatel ví, co hledá.

4.3 Založení nového projektu

Při opětovném spouštění PC WorX se nahrává i poslední otevřený projekt. Aby se předešlo nešťastné události, kdy si můžeme přepsat původní soubor, tak se z tohoto důvodu doporučuje založit nový projekt. To provedeme pomocí „File/New Project…“.

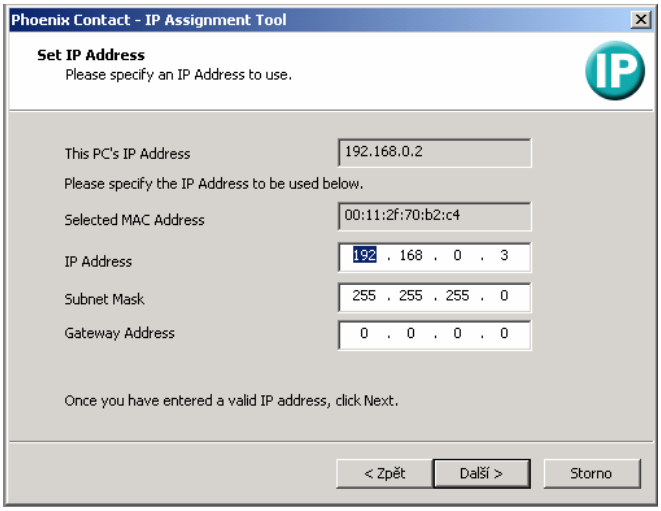
Otevře se okno(obr.13), které slouží pro výběr PLC, pro které se bude program vytvářet. V tomto případě se jedná o ILC 150 ETH a firmware ve verzi 2.00. Tato verze je na obrázku označena myší. Pro lepší orientaci a rychlejší vyhledání verze PLC je vhodné přepnout zobrazení ze standardních velký ikon na menší ikony nebo seznam. Výběr se provede poklepáním nebo označením a poté potvrdit tlačítkem OK.



Obr.13:New Project

4.4 Přidělení IP adresy

Jelikož ILC 150 komunikuje přes ethernetové rozhraní je nutné pro komunikaci, aby každé zařízení v síti mělo IP adresu. Toho lze velice snadno dosáhnout programem IP Assignment, který je také produktem firmy Phoenix Contact. Program není nutno instalovat a v několika málo krocích lze IP adresu PLC přidělit. Po spuštění IP Assignment se řídíme pokyny. Hned v druhé obrazovce provedeme výběr MAC adresy zařízení a pokud neznáme MAC adresu síťové karty PC, tak je lépe ponechat zaškrtnutou položku „Show Only Phoenix Contact devices“ , která zobrazuje pouze MAC adresy zařízení fy Phoenix Contact, protože pokud vybereme špatnou MAC adresu, tak se přirozeně IP adresu nepodaří přidělit. Pokud se MAC adresy nezobrazují je potřeba vypnout PLC ze sítě a opětovně jej zapnout. Na obrázku 5 je krok, kde zvolíme IP adresu. IP adresa musí být v rámci sítě unikátní. Po vyplnění potřebných údajů (obr.14) přejdeme na další obrazovku, kde se opět opakuje zapnutí-vypnutí PLC a adresa je poté přidělena, opět se řídíme instrukcemi IP Assignment.

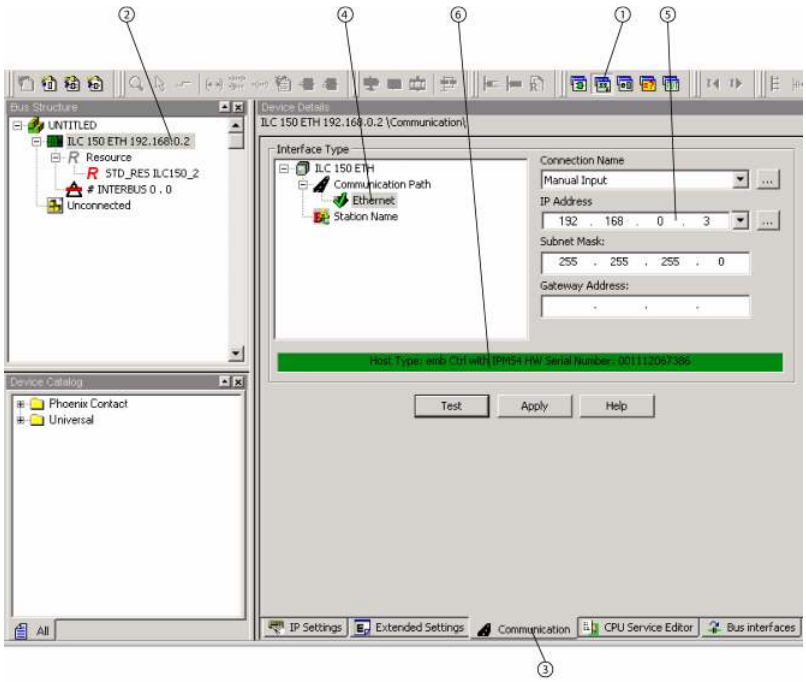


Obr.14:Přidělení IP adresy

4.5 Test komunikace a nastavení IP adresy

V odstavci 2.4 bylo popsáno přidělení IP adresy automatu. Nyní je potřeba otestovat komunikaci mezi PC a automatem a následně v PC WorX nastavit IP adresu(stejnou jako má automat).

Pomocí ikony „Bus Configuration Workspace“ (pozice 1 na obr.14) se přepneme do konfigurace sběrnice. Vybereme příslušné zařízení (2), kde je v jeho názvu i IP adresa , pomocí které komunikuje. Mezi záložkami vybereme „Communication“ (3) v okně „Device Details“ se doklikáme na „Ethernet“(4) jako komunikační cestu . Pak zapíšeme IP adresu (5), kterou jsem v odstavci 4.4 přidělili automatu. Poté tlačítkem „Test“ spustíme test komunikace. O průběhu testu nás PC WorX informuje v rámečku(6), pokud test probíhá v pořádku, je pozadí rámečku zelené. Po úspěšném testu klikneme na „Apply“.

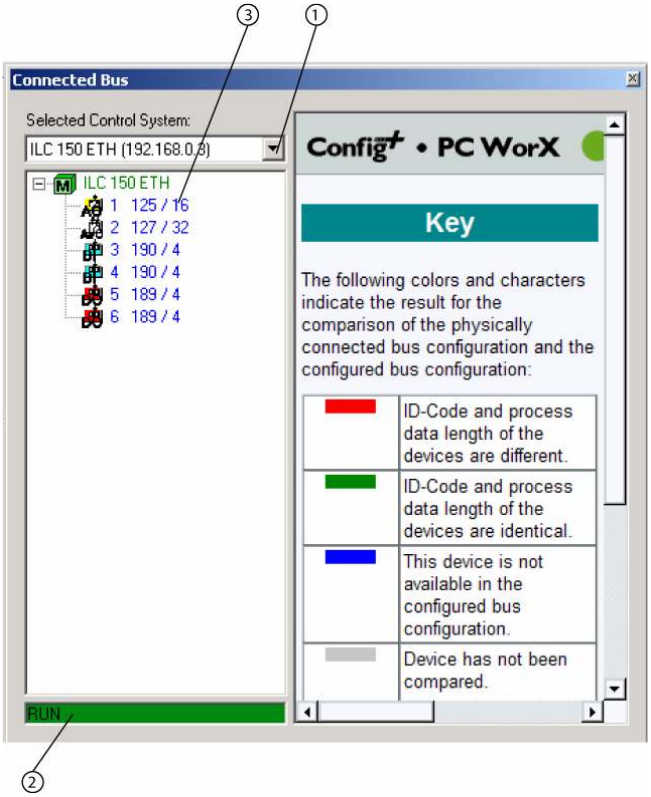


Obr.15:Test komunikace

4.6 Konfigurace přídavných modulů v prostředí PC WorX

Přestože je PC WorX na velmi vysoké úrovni i tak je potřeba některé věci udělat ručně nebo programu napomoci. Konfigurací modulů se ukončí základní nastavení před samotným započetím tvorby projektu. Automat s moduly v tomto případě komunikuje po Inline sběrnici a automat pomocí ní získá údaje o připojených modulech. Tyto informace předá PC WorXu, ale program dokáže rozlišit pouze skupinu, do které je patřičný modul zařazen. Toto řazení se provádí podle identifikačního kódu modulu a „Process Data Length“. Kombinaci těchto kódů odpovídá vždy několik modulů a v označení se mohou lišit třeba jen v jednom písmenku nebo číslici. Špatně identifikovaný modul může mít následky v podobě nefunkčnosti programu (Program např. může odkazovat výstupy na modul, který v podstatě není připojen nebo naopak získávat data z modulu). V takovém případě se modul chová jakoby se ho program netýkal a PC WorX přitom žádnou chybu nehlásí. Proto je potřeba věnovat náležitou pozornost právě této činnosti a zbytečně se nedopouštět chyb. Správnou konfigurací modulů se předejde nepříjemnostem v podobě nové konfigurace a úpravám v projektu, především proměnných, které se odkazují na patřičný modul/y .

Samotné nastavení se provádí pomocí „Bus Configuration Workspace“(pozice 1 na obr. 15) a poté v menu „View / Connected Bus“.

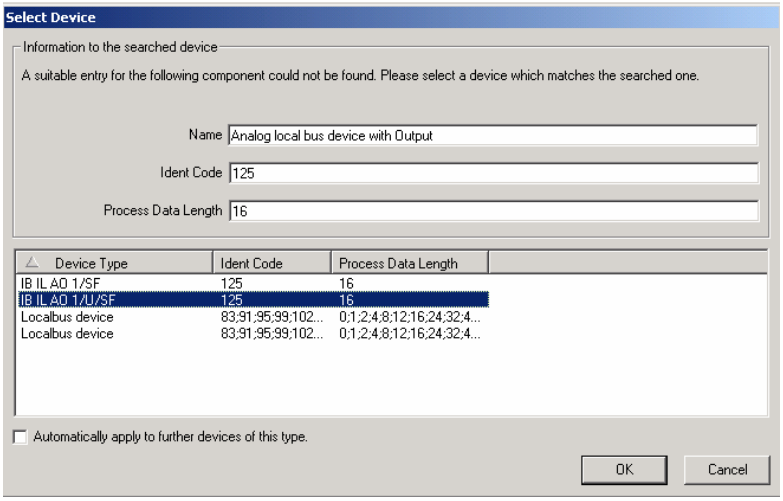


Obr.16:Konfigurace modulů

Na obr.16 je okno „Connected Bus“ , ve kterém se provádí konfigurace modulů. V rozbalovacím menu(1) se vybere zařízení s příslušnou IP adresou. Po vybrání zařízení se v informačním okně (2) provádí kontrola komunikace a načtení připojených modulů. Ty jsou pak zobrazeny ve struktuře stromu(3) přesně v takovém pořadí, jak jsou připojeny. Každý modul, který není nakonfigurován je reprezentován barevně odlišenou ikonou se zkratkou modulu(DO,DI,AO atd.). Barva ikony koresponduje se skutečným barevným

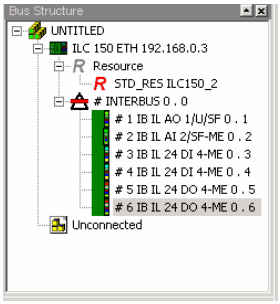
označením modulu. Číslice za ikonou označuje pořadí modulu dle připojení. ID-Code/ Process Data Length je dvojice čísel oddělená lomítkem.

Pravým tlačítkem myši klikneme na první modul a vybereme „Import to Project“ / „With Device Description“ .



Obr.17: Výběr zařízení

Na obr.17 je okno „Select Device“, ve kterém se provádí výběr modulů. Výběr probíhá postupně podle očíslování. Pod oknem výběru je položka, která se dá zaškrtnout. Ta usnadní práci pokud máme více stejných modulů tím, že je modul přidán automaticky. Tedy v našem případě ji necháme zaškrtnutou pro poslední dva moduly.



Obr.18: Struktura sběrnice

Po úspěšné konfiguraci modulů jejich označení změní barvu z modré na zelené a také dojde k zařazení modulů do struktury sběrnice(obr.18). Tímto je vše nakonfigurováno.

4.7 Šablona a uložení projektu

Po úspěšném absolvování kroků z kapitol 4.4 až 4.6 je velice výhodné toto nastavení uložit jako šablonu. Hlavní výhoda spočívá v tom, že předchozí kroky nebude nutné absolvovat, pokud se rozhodneme programovat pro stejný automat. Šablona se bude poté nalézat ve výběru „New Project“ viz. 4.3. Uložení šablony se provede pomocí „File“ a „Save As Template…“, zadáme jméno šablony a potvrdíme stiskem „OK“ . Doporučeno je taktéž uložit projekt. To opět provedeme v menu „File“ a „Save Project As…“ , další ukládání se provádí pomocí ikony diskety na liště. Tento způsob ukládání převládá v naprosté většině programů.

4.8 Programování v prostředí PC WorX

4.8.1 Programovací jazyky pro PLC

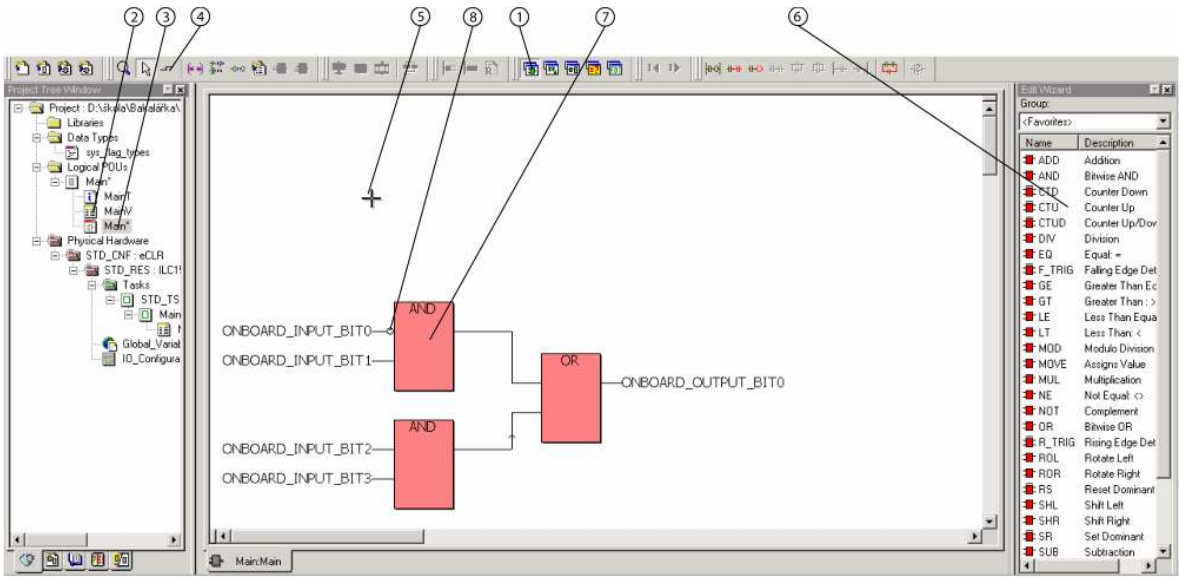
K programování nabízejí PLC systémy specializované jazyky, původně navržené pro snadnou, názornou a účinnou realizaci logických funkcí. Jazyky systémů různých výrobců jsou podobné, nikoliv však stejné. Přímá přenositelnost programů mezi PLC různých výrobců není možná, daří se to obvykle jen mezi systémy téhož výrobce. Mezinárodní norma IEC 1131-3 sjednocuje programovací jazyky pro PLC.[PLC a automatizace].

V rámci normy IEC 61131-3 jsou doporučovány čtyři programovací jazyky s přesně definovanou sémantikou a syntaxí: LD, FBD, IL a ST. Jako pátý programovací jazyk se často uvádí sekvenční funkční diagram – SFC, který však není v normě zařazen mezi jazyky, ale mezi tzv. společnými prvky, neboť tvoří jakousi nadstavbu pro strukturování celé aplikace. Dále existuje např. nabídka tzv. inženýrských nástrojů STEP7, kam patří S7-Graph, S7-HiGraph a CFC. To jsou však spíše grafické nástroje pro programování, nikoli jazyky s přesně definovanou syntaxí a sémantikou. [2]

4.8.2 Jazyk funkčního blokového schématu (FBD)

FBD (Function Block Diagram), je grafický jazyk, který vyjadřuje chování funkcí, funkčních bloků a programů jako soubor vzájemně provázaných grafických bloků podobně jako v elektronických obvodových diagramech. Jde o systém prvků, které zpracovávají signály. Často se zde používají standardní funkční bloky, jako jsou např. bistabilní prvky (paměti s dominantním vypnutím nebo sepnutím, semafor), prvky pro detekci náběžné a sestupné hrany, čítače, časovače a komunikační bloky definované v normě IEC 1131-5. Podle potřeby jsou doplňovány speciální bloky a každá firma nabízí ve svém programovacím prostředí poněkud odlišný soubor bloků (např. spínací hodiny týdenní, roční, generátory impulzů, komparátory apod.). [2]

Vlastní programování pomocí tohoto jazyka nám přiblíží obr.19., které opět zobrazíme pomocí ikony(1) „IEC Programming Workspace“

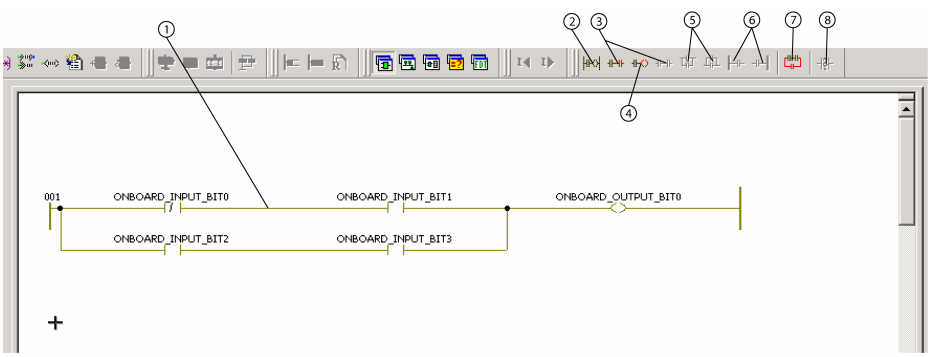


Obr.19:FBD

FBD je standardně nastavený programovací jazyk v prostředí PC WorX. V okně projektového stromu se nalézá složka „Logical POUs“. Jeho rozbalením se lze dostat do nastavení proměnných MainV(2) a pomocí Main(3) se symbolem FBD také programovat. Křížem(5), který umístíme levým tlačítkem myši v programovacím okně, lze stanovit polohu pro umístění funkčního bloku. Na pozici kříže bude následně levý horní roh bloku. Bloky lze vybírat v editačním okně(6), kde jsou členěny do několika skupin. Orientaci a výběr skupin umožňuje rozbalovací menu v témže okně. Spojování bloků se realizuje pomocí ikony „Connect“(4) se symbolem lomené čáry. Spojování bloků je intuitivní. Spojovat bloky lze pouze způsobem vstup/výstup. Vstupy jsou označeny modře, výstupy potom zeleně. Poklepáním na výstup bloku se vyvolá okno pro přiřazení proměnných(viz. 4.9). Ukázka programování pomocí FBD (7), kde jsou spojeny 3 bloky a na vstupy a výstup jsou přiřazeny lokální proměnné. Za zmínku stojí realizace negovaných vstupů/výstupů, která je zde symbolizována kružnicí(8) na hraně bloku. Negace, počet vstupů a velikost bloku lze nastavit velmi jednoduše poklepáním na konkrétní blok.

4.8.3 Jazyk příčkového diagramu (LD)

Grafický jazyk LD (Ladder Diagram) je někdy také nazýván jazykem kontaktních schémat a je založen na grafické reprezentaci reléové logiky. Organizační jednotka programu je vyjádřena sítí propojených grafických prvků. Síť v jazyku LD je zleva i zprava ohraničena svislými čarami, které se nazývají levá a pravá napájecí sběrnice. Mezi nimi je tzv. příčka, která může být rozvětvena. Každý úsek příčky, vodorovný nebo svislý, může být ve stavu on nebo off. Do příček mohou být včleněny kontakty (spínací, rozpínací apod.), cívky (cívka, negovaná cívka apod.) a dále funkce a funkční bloky. [2]



Obr.20:LD

Na obr.20 je vyjádřen pomocí LD jazyka ukázkový program totožný s tím na obr. 19, kde je vyjádření pomocí FBD. K realizaci těchto schémat se hojně využívá především skupiny ikon „Ladder“. Pozice vkládání prvků se opět určuje pomocí kříže. Po umístění kříže jsou aktivní některé ikony, které je možno použít. Začne se vytvořením sítě(2) Network. Pokud neprogramujeme přímo nový blok, tak se nám vytvoří kompletní síť, která na obou koncích obsahuje napájecí sběrnice a mezi nimi se nachází jeden kontakt a jedna cívka. Pro přidání kontaktů slouží ikony (3) „Conntact right“ a „Conntact left“. Cívka se připojí ikonou(4) „Coil right“. Paralelně lze přidat prvky pomocí ikon (5) „Parallel“(přidá prvek paralelně pod aktivní prvek) a „Add contact/coil above“(přidá prvek paralelně nad aktivní prvek). Levá a pravá napájecí sběrnice se přidá ikonami(6) „Left powerrail“ a „Right Powerrail“. Vytvoření celé větve programu můžeme provést ikonou(7) „Branch“. Ikonou(8) „Contact type“ lze měnit typ kontaktu(otevřený, zavřený) nebo typ cívky(set cívka, reset cívka, cívka, negovaná cívka).

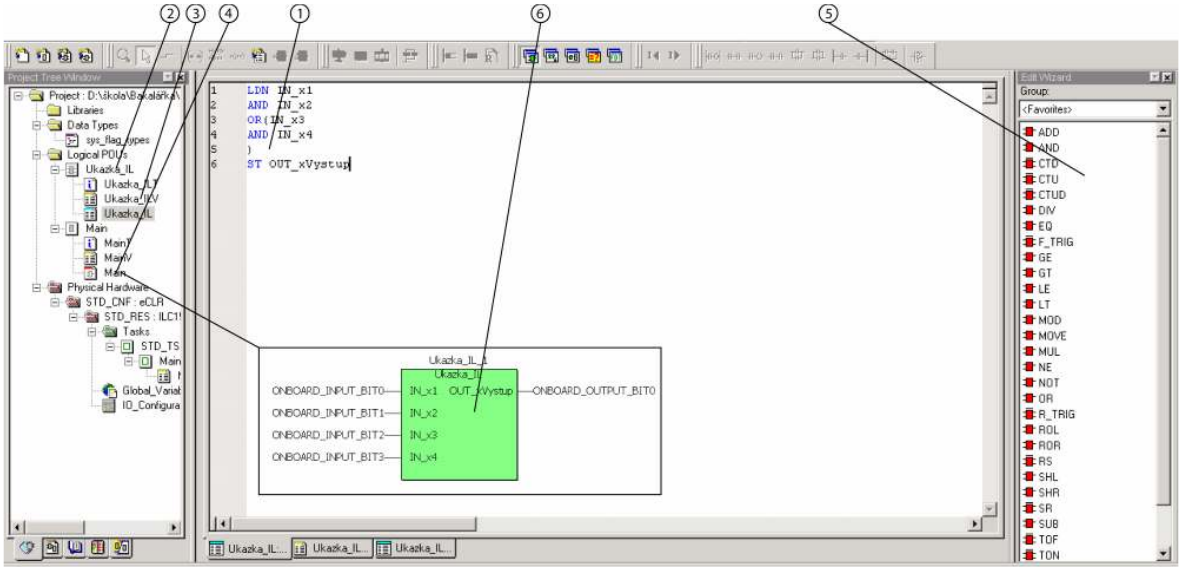
Na obr.21 jsou v tabulce uvedeny možné typy kontaktů a cívek v rámci programování pomocí jazyka LD.

Symbol	Jméno	Popis funkce
⊣ ⊢	Kontakt typu A (normally open contact)	Prvek, který kopíruje hodnotu na levé straně na stranu pravou, pokud je stav přiřazené funkce ve stavu logické 1.
⊣/⊢	Kontakt typu B (normally closed contact)	Prvek, který kopíruje hodnotu na levé straně na stranu pravou, pokud je stav přiřazené funkce ve stavu logické 0.
−( )−	Cívka (Coil)	Prvek, který hodnotu logické funkce kopíruje z levé strany na pravou stranu a do přiřazené funkce, kterou cívka reprezentuje.
−(/)−	Negovaná cívka (Negated coil)	Prvek, který vykonává stejnou funkci jako cívka, ale do přiřazené funkce kopíruje negovanou hodnotu logické funkce.
−(S)−	SET cívka (SET coil)	Prvek, který vykonává stejnou fukci jako cívka, ale nastavuje hodnotu přiřazené funkce, jestliže je na levé straně logická 1.
−(R)−	RESET cívka (RESET cívka)	Prvek, který vykonává stejnou fukci jako cívka, ale resetuje hodnotu přiřazené funkce, jestliže je na levé straně logická 1.

Obr.21: Cívky a kontakty

4.8.4 Jazyk seznamu instrukcí (IL)

Textový jazyk IL (Instruction List) označovaný také jako jazyk pokynů (povelů), seznam instrukcí (STL – Statement List) poněkud připomíná assembler. Programová organizační jednotka je složena ze sekvence instrukcí, z nichž každá začíná na novém řádku a může obsahovat návěští (nepovinné) ukončené dvojtečkou, operátor (např. AND, &, ADD, CAL apod.), který může být případně doplněn tzv. modifikátorem, operand a někdy také komentář (nepovinný). Pomocí modifikátorů se vyjadřují negace, podmíněnost a nepodmíněnost instrukce skoků, volání a návratů a priorita. [2].



Obr.22: IL

Na obr.22 je pomocí IL(1) zapsán stejný ukázkový program jako v předchozích případech. Pomocí tohoto jazyka není možné programovat v hlavním programu a lze tak pomocí něho programovat pouze programy, funkce a funkční bloky. Při programování

např. funkčního bloku se v POU vytvoří další složka(2). Proměnné tohoto bloku je potřeba zapsat ručně do složky proměnných(3). Samotný blok se po naprogramování zařadí do „Edit Wizard“ (5), který se vkládá do hlavního programu „Main“ (4). Ovšem i z „Edit Wizard“ je možné si vytáhnout pomocný zdrojový kód do podprogramu a následně ho upravit k obrazu svému. Výřez(6) z tohoto programu je také na obr.22

4.8.5 Jazyk strukturovaného textu (ST)

Textový jazyk ST (Structured Text) je výkonný vyšší programovací jazyk, který má kořeny v jazycích Pascal a C. Syntaxe jazyka je dána povolenými výrazy a příkazy. Vyhodnocením výrazu vyjde hodnota v některém z definovaných datových typů. Výraz se skládá z operátorů a operandů. Operandem může být konstanta, proměnná, funkce nebo jiný výraz. Operátory pro jazyk ST jsou definovány pro sedmnáct typů operací (vyhodnocení funkce, negace, násobení, booleovské funkce AND, XOR a OR apod.). Je definováno deset typů příkazů (přiřazení, vyvolání funkce, návrat, výběr apod.). Příkazy jsou odděleny středníkem a může jich být více na jednom řádku. Jazyk ST je vhodným nástrojem pro definování komplexních funkčních bloků, které pak mohou být použity v libovolném programovacím jazyku. [2].

```
1    AND_1 := not IN_x1 & IN_x2;
2    AND_2 :=     IN_x3 & IN_x4;
3    OUT_xVystup := AND_1 OR AND_2;
```

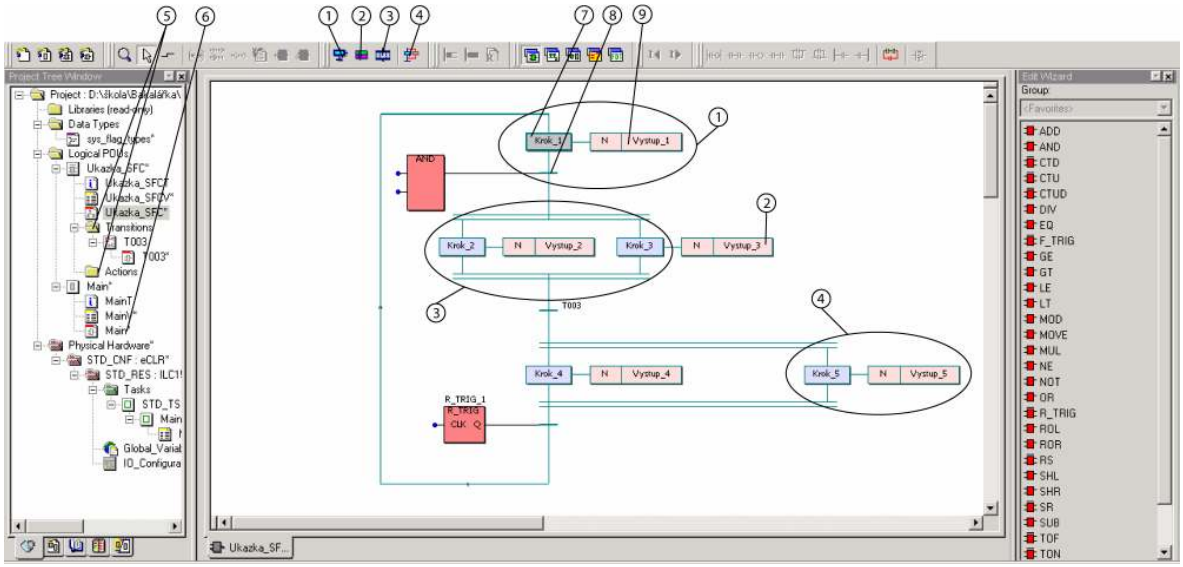
Obr.23:ST

Na obr.23 je opět zapsán stejný program, jako u ukázek ostatních jazyků. Bohužel i zde platí to co pro IL, takže nelze použít v hlavním programu. Schopnost velké efektivity programování pomocí ST tedy využijeme pouze při programování programů, funkcí a funkčních bloků. Způsob jeho programování je totožný s IL, proto na obr.23 je pouze ukázka zdrojového kódu. Ruční nastavení proměnných u podprogramů taktéž totožné jako IL stejně jako možnost využívání okna „Edit Wizard“ při psaní kódu.

4.8.6 Sekvenční funkční diagram (SFC)

SFC (Sequential Function Chart) popisuje sekvenční chování řídicího programu. Je odvozen ze symboliky Petriho sítí, ale liší se od nich tím, že grafická reprezentace se zde převádí přímo do souboru výkonných řídicích prvků. SFC strukturalizuje vnitřní organizaci programu a umožňuje rozložit úlohu řízení na zvládnutelné části a zachovat přitom přehled o chování celku. Sekvenční funkční diagram se skládá z kroků a přechodů. Každý krok reprezentuje stav řízeného systému a má k sobě přiřazen blok akcí. Přechod je spojen s podmínkami, které musí být splněny, aby mohl být deaktivován krok, který přechodu předchází, a naopak aktivován krok, který následuje. Každý prvek, tzn. přechod i blok akcí, může být naprogramován v libovolném jazyku definovaném v normě, včetně vlastního SFC. K základním strukturám SFC patří lineární sekvence, alternativní větvení se

spojením alternativních větví a paralelní souběh více větví s jejich následnou synchronizací. [2]



Obr.24:SFC

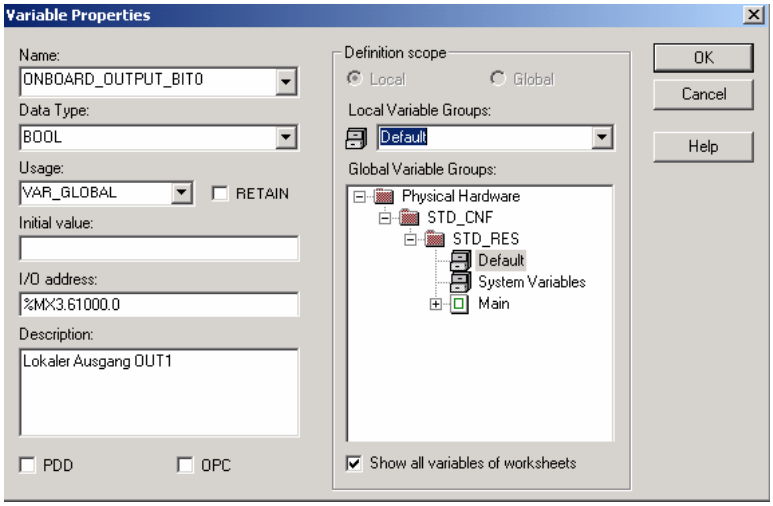
Obr.24 představuje jeden z mnoha možných typů SFC programu. K tomuto programování je určena skupina ikon (1 až 4). „Create step transition sequence (1) slouží k přidání jedné sekvence programu, která obsahuje Krok (7), Přechod (8) a Akci (9). Krok může být inicializační, skokový nebo koncový. Program pro přechod lze psát zvlášť do sekce „Logical POUs“, kde je sekce „Transitions“ nebo ho lze připojit přímo jako na tomto obrázku. Akce(9) slouží pro zápis funkce, která se má vykonat- toto se píše do „Actions“ v „Logica POUs“ nebo může sloužit jako výstup. Ikonou „Create Action“ (2) se dá vytvořit další reakce. Program může mít několik různých akcí na jeden krok. Ikonou „Insert Simultaneous/Alternative Divergence (3) lze program rozdělit. Ikona „Insert SFC Branch“ (4) vytvoří další větev v programu.

Pomocí SFC lze programovat funkce nebo funkční bloky a také programovat v hlavním programu „Main“ (6).

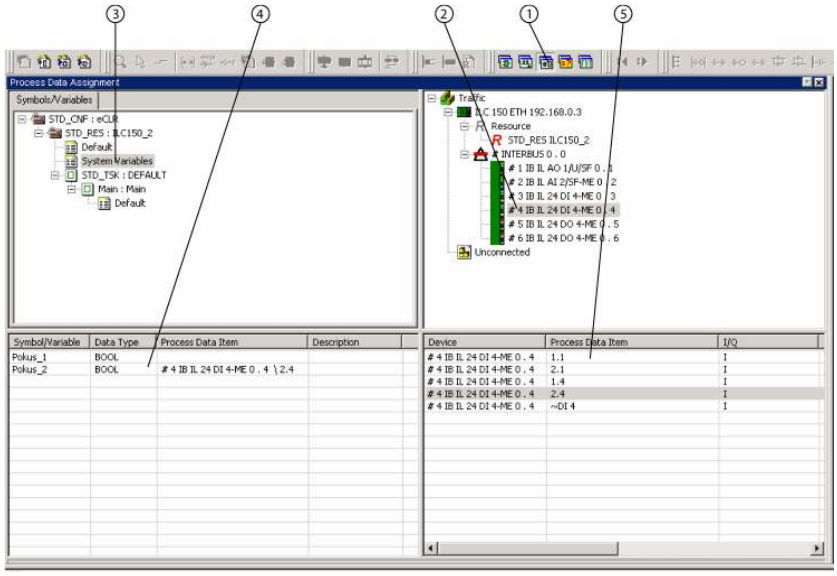
4.9 Proměnné

Proměnné se generují automaticky a jména se přidělují podle následujícího klíče:
<I or Q>_<IBS segment>_<IBS position>_<PD name>
I = Input; Q = Output
IBS= INTERBUS
PD = Process Data
Toto platí pouze pro přímé vstupy/výstupy.[4]

V tomto okně(obr.25) se provádí pojmenování, volba datového typu, volba typu proměnné, zvolení inicializační hodnoty, popis atd. Adresace proměnných přímo na výstupy/vstupy ILC 150 ETH se provádí také zde, ale adresace přímo na moduly se provádí pomocí ikony „Process Data Workspace“ (pozice 1 na obr. 26). Resp. v okně „Variable Properties“ lze název proměnné přiřadit na výstup připojeného modulu, ale v „Process Data Assignment“ je potřeba tuto proměnnou přiřadit konkrétnímu výstupu/vstupu modulu.



Obr.25:Nastavení proměnných



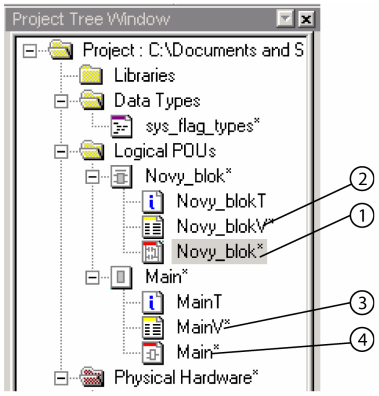
Obr.26:Process Data Assignment

Pokud chceme adresovat proměnné na připojené moduly je třeba, aby tyto proměnné byly globální tzn. v okně stromu projektu je položka „Global_Variables“ a sem je potřeba tyto proměnné zařadit. Na obr.26 můžeme vidět, že okno „Process Data Assignment“ je rozděleno na čtyři menší okna. V pravém horním okně lze vybrat ve struktuře sběrnice ILC včetně všech připojených modulů nebo si vybrat konkrétní modul(2). V okně nalevo je také strom, ale týká se proměnných. Zde je potřeba najít proměnné podle zařazení např. „Systém Variables“(3). Ve spodním okně(4) se vypíšou dostupné proměnné. Myší si označíme proměnnou, kterou chceme někam přiřadit. V posledním okně si vybereme výstup, na který chceme adresovat proměnnou. Pravým tlačítkem na výstupu rozbalíme menu a vybereme „Connect“ .

4.10 Programování vlastního bloku

Velkou výhodou programování v prostředí PC WorX je možnost naprogramovat si vlastní blok. Jeho vnitřní funkce může být naprogramována pomocí libovolného jazyka obsaženého v normě IEC 61131-3.

Pravým tlačítkem myši klikneme na „Logical POUs“, vybereme „Insert…“ a poté „Function block“. Otevře se okno, ve kterém vyplníme jméno a zvolíme si požadovaný jazyk pro programování.

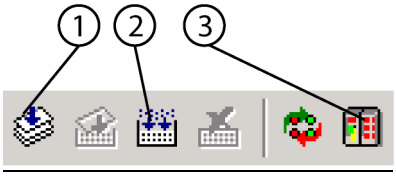


Obr.27:Project Tree Window

Pokud si zvolíme za název např. „Novy_blok“ , tak na obr.27 můžeme vidět „Project Tree Windows“, kde se vytvořil nový pracovní sešit. Pozice 1 označuje místo, kam se píše kód podprogramu. Lokální proměnné tohoto podprogramu se píšou do tabulky pro proměnné(2). V okně „Edit Wizard“ se objeví tento blok a bude označen světle zelenou barvou. Ten lze vložit do „Main“(4), kde ho lze použít pro další programování. Proměnné se nyní samy generují při kompilaci. V případě programovaní podprogramu pomocí LD se generují proměnné samy i v tabulce(2) pro podprogram.

4.11 Kompilace projektu a downloady do PLC

Předtím než hotový program nahrajeme do PLC, je potřeba zkompilovat projekt.



Obr.28:Kompilace

Ikona „Compile Worksheet“ na obr.28 (1) slouží pro kompilaci pracovního sešitu. Kompilují se typy, proměnné a zdrojový kód v „Main“. Tato ikona se užívá hlavně pro odhalení chyb v syntaxi programu. Ikona „Make“(2) je bez nadsázky nejdůležitější v celém vývojovém prostředí. Kontroluje sběrnici, vytváří konfigurační rámce, připojuje POU a především generuje kód, který se odesílá do PLC. Jinak také umí to co ikona „Compile Worksheet“. Průběh činnosti je vypisován do okna „Message Windows“.

Pro nahrání projektu slouží ikona „Project Control Dialog“(3). Na obr.29 je okno „Project Control Dialog“. Podává informaci o stavu automatu(RUN/STOP). Pro nahrání programu je potřeba být v režimu STOP, kdy PLC přestane vykonávat stávající program, který je v něm nahrán. Stisknutím „Download“ se spustí nahrávání projektu. Pokud je v PLC jiný program, než se pokoušíme právě nahrát, tak vyskočí varování, které nás upozorní na tuto situaci. Stisknutím „ANO“ započne nahrávání, jeho průběh je možné sledovat ve „Status Baru“. Po úspěšném nahrání programu, je možné opět PLC uvést do chodu příkazem „Cold“.

Nebývá zvykem označovat nahrávání projektu „někam“ jako „download“ ale spíše jako „upload“. Toto označení se používá především v počítačové terminologii. Zde je to opačně.

Obr.29:Project Control Dialog

5 Praktické úlohy

5.1 Úvod

Vývojové prostředí PC WorX využívá pro tvorbu programů všechny dostupné programovací jazyk obsažené v normě IEC 61131-3 a také jejich vzájemných kombinací což významným způsobem rozšiřuje schopnosti tohoto prostředí a také PLC.

Cílem těchto úloh je zaměřit se na programovací jazyky dle normy IEC 61131-3 a vytvořit jednoduché avšak zajímavé úlohy, které lze prakticky vyzkoušet na automatech fy Phoenix Contact.

Při programování je třeba mít na paměti, že je třeba dodržovat určité standarty, které se týkají označování proměnných a hlavně také dávat pozor na nešťastně nastavené proměnné vstupů(Obr.30).

Device	Input/Output	Signal at	Variable
ILC 150 ETH	Input I1	Connector 3 terminal point 1.1	ONBOARD_INPUT_BIT0
	Input I2	Connector 3 terminal point 2.1	ONBOARD_INPUT_BIT1
	Input I3	Connector 3 terminal point 1.4	ONBOARD_INPUT_BIT2
	Input I4	Connector 3 terminal point 2.4	ONBOARD_INPUT_BIT3
	Input I5	Connector 4 terminal point 3.1	ONBOARD_INPUT_BIT4
	Input I6	Connector 4 terminal point 4.1	ONBOARD_INPUT_BIT5
	Input I7	Connector 4 terminal point 3.4	ONBOARD_INPUT_BIT6
	Input I8	Connector 4 terminal point 4.4	ONBOARD_INPUT_BIT7
IB IL AO 1/U/SF-PAC	Output O1	Connector 5 terminal point 1.1	Output_Analog
IB IL AI 2/SF-ME	Input I9	Connector 6 terminal point 1.1	Input_Analog

Obr.30:Popis vstupů a výstupů[4]

Pro označování proměnných v projektu je vhodné volit názvy tak, aby nebyly zaměnitelné a držely se standartu(obr.31).

Datový typ	Označení	Příklad
REAL	R	IN_rVstup
BYTE	B	IN_bData
BOOLEAN	X	IN_xZapnuto

Obr.31:Označování proměnných

5.2 Realizace logické funkce dané tabulkou

5.2.1 Popis

Jedná se o jednoduchou úlohu zaměřenou na logické funkce. Cílem úlohy je zopakování znalostí z logického řízení především minimalizace pomocí KNF nebo DNF, Karnaughovy mapy, sestavení blokového schématu a programování této funkce v PC WorX.

5.2.2 Zadání

Minimalizujte logickou funkci tří proměnných danou pravdivostní tabulkou. Vhodně zvolte KNF nebo DNF při minimalizaci. Funkci taktéž minimalizujte pomocí Karnaughovy mapy. Výslednou logickou funkci zakreslete pomocí blokových schémat negace, konjunkce a disjunkce.

Realizujte danou funkci ve vývojovém prostředí PC WorX a poté její funkčnost prakticky vyzkoušejte na PLC ILC 150 ETH. Funkci realizujte pomocí FBD, LD a IL . Pro IL naprogramujte vlastní blok z důvodů zmíněných v odstavci 4.8.4 . Funkce je zadána následující pravdivostní tabulkou:

5.2.2.1 Pravdivostní tabulka

Vstup 1	Vstup 2	Vstup 3	Výstup
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Obr.32:Pravdivostní tabulka

Na obr.32 je klasická pravdivostní tabulka obsahující 3 vstupní proměnné a jednu výstupní. Počet kombinací výstupních hodnot je v tomto případě 2^3. V následujícím textu budou vstupní hodnoty označeny písmenem X a výstupní Y. Symboliku je samozřejmě volit libovolně.

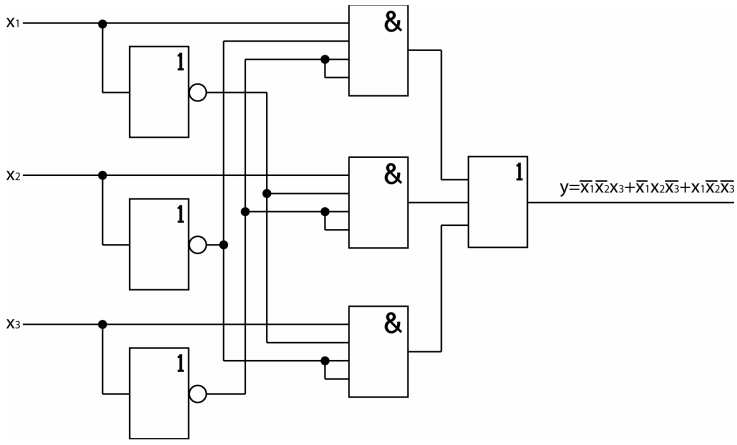
5.2.3 Řešení

Provedeme algebraický zápis pro DNF, ze kterého lze hned sestavit blokové schéma. V tomto případě je vhodné zvolit minimalizaci DNF. Důvod plyne z tabulky na obr.32, kdy se vyplatí kvůli délce algebraického zápisu funkce zvolit DNF. Zápis DNF je na obr.33.

$$y=\overline{x}_1\overline{x}_2x_3+\overline{x}_1x_2\overline{x}_3+x_1\overline{x}_2\overline{x}_3$$

Obr.33:Funkce

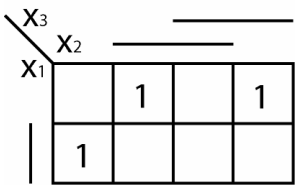
5.2.3.1 Blokové schéma



Obr.34:Blokové schéma

Obr.34 je blokové schéma logické funkce, které je tvořeno bloky negace, konjunkce a disjunkce.

5.2.3.2 Karnaughova mapa



Obr.35:Karnaughova mapa

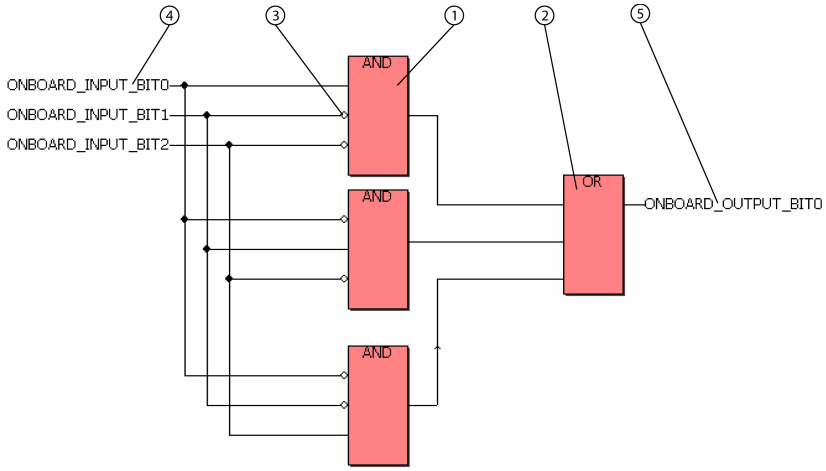
Z obr.35, na kterém je minimalizace pomocí Karnaughovy mapy, je vidět, že minimalizace není možná.

5.2.3.3 FBD realizace

	Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC	TB	I/Q	Connection Point
	Default											
	ONBOARD_INPUT_BIT0	BOOL	VAR_EXTERNAL	Lokaler Eingang IN1			☐	☐	☐	☐		
	ONBOARD_INPUT_BIT1	BOOL	VAR_EXTERNAL	Lokaler Eingang IN2			☐	☐	☐	☐		
	ONBOARD_INPUT_BIT2	BOOL	VAR_EXTERNAL	Lokaler Eingang IN3			☐	☐	☐	☐		
	ONBOARD_OUTPUT_BIT0	BOOL	VAR_EXTERNAL	Lokaler Ausgang OUT1			☐	☐	☐	☐		

Obr.36:Tabulka proměnných

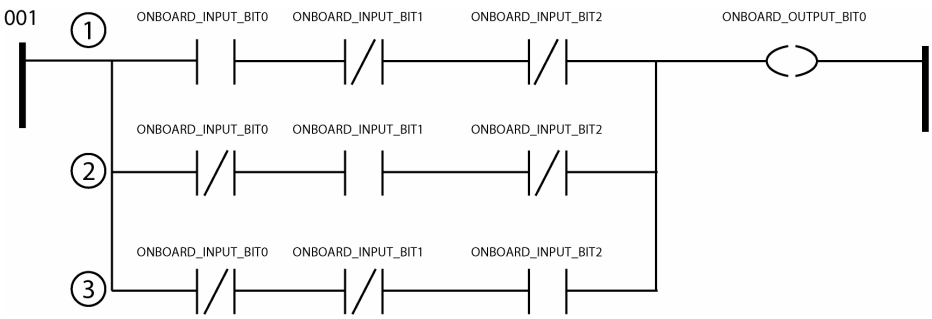
Na obr.36 je tabulka proměnných, která se nalézá v „MainV“ v „Project Three Windows“. Obsahuje 4 proměnné, z nichž 3 jsou vstupní a 1 výstupní. Všechny jsou typu BOOL, který reprezentuje booleovské funkce a taktéž všechny proměnné jsou lokální za předpokladu, že vstupy i výstupy jsou adresovány přímo na samotné PLC.



Obr.37:FBD schéma

Blokové schéma vytvořené v jazyce FBD je na obr.37. Je vytvořeno z 3 hradel AND(1) a jednoho hradla OR(2). Negace vstupů (3) je vytvořena dle kapitoly 4.8.2 . Dle kapitoly 4.9 jsou přiřazeny vstupy(4) a výstup (5). Funkce hradel je všeobecně známa a ani zde nemá jiný význam a tedy funkci programu není třeba blíže specifikovat.

5.2.3.4 LD realizace



Obr.38:LD schéma

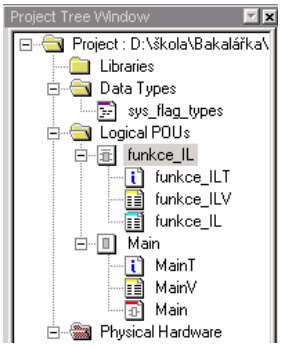
V tomto případě je tabulka proměnných stejná jako u realizace pomocí FBD a proto se odkážeme na obr.36

Pro lepší přehlednost je schéma LD překresleno na obr.38. Chod programu bude popsán v následujících větách.

Nejprve se vykoná přímá větev(1) programu. Cívka se sepne pokud kontakt x1(logická 1) bude sepnut a x2 a x3(logické 0) zůstanou rozepnuté. Jestliže nenastane přesná kombinace těchto vstupů, tak program přejde na paralelní větev(2). Ta má následující kombinaci: cívka se sepne, pokud x1 a x3(logické 0) zůstanou rozepnuté a sepne se kontakt x2(logická 1), pokud tato kombinace nevyhovuje, tak program přejde na další větev (3). Kombinace, která zajistí sepnutí cívky je následující: cívka se sepne, pokud se sepne kontakt x3(logická 1) a x1 a x2 zůstanou rozepnuté (logická 0).

5.2.3.5 IL realizace

Pro realizaci pomocí IL jazyka bylo potřeba naprogramovat nejprve funkční blok a ten poté umístit do „Main“ a připojit k němu vstupy a výstupy.



Obr.39:IL Project Tree

Na obr.39 je „Project Tree Windows“ programu realizovaného pomocí IL. Pomocný podprogram se jmenuje „funkce_IL“

	Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC	TB
	Default									
	IN_x3	BOOL	VAR_INPUT				☐	☐	☐	☐
	IN_x2	BOOL	VAR_INPUT				☐	☐	☐	☐
	IN_x1	BOOL	VAR_INPUT				☐	☐	☐	☐
	FU_IL	BOOL	VAR_OUTPUT				☐	☐	☐	☐

Obr.40:Tabulka proměnných

Na obr.40 je tabulka proměnných pro podprogram, která je sekci „funkce_ILV“ (viz.obr.39). Tyto proměnné je potřeba zapsat ručně, protože při kompilaci nejsou generovány. Tyto proměnné jsou pouze v rámci tohoto podprogramu.

```
1   LD    IN_x1    (*načtení a uložení hodnoty x1 na první pozici zásobníku*)
2   ANDN  IN_x2    (*načtení negované hodnoty x2 a provedení logického součinu
3                   s hodnotou x1, která je na první pozici zásobníku. Výsledek
4                   operace se uloží místo pozice x1*)
5   ANDN  IN_x3    (*načtení negované hodnoty x3 a provedení logického součinu
6                   s hodnotou , která je na poslední pozici zásobníku a uložení
7                   na tuto pozici*)
8
9   OR(   IN_x2    (*Provede se logický součet poslední hodnoty v zásobníku a *)
10  ANDN  IN_x1    (*hodnotou vzniklou operacemi v závorce(logický součin tří hodnot) *)
11  ANDN  IN_x3    (* a výsledek se uloží místo poslední hodnoty v zásobníku *)
12  )
13
14  OR(   IN_x3   (*Provede se logický součet poslední hodnoty v zásobníku a *)
15  ANDN  IN_x2   (*hodnotou vzniklou operacemi v závorce(logický součin tří hodnot) *)
16  ANDN  IN_x1   (* a výsledek se uloží na poslední hodnotu v zásobníku *)
17  )
18
19  ST FU_IL (*ulozeni konecneho vysledku v zásobníku do výstupu*)
```

Obr.41:IL

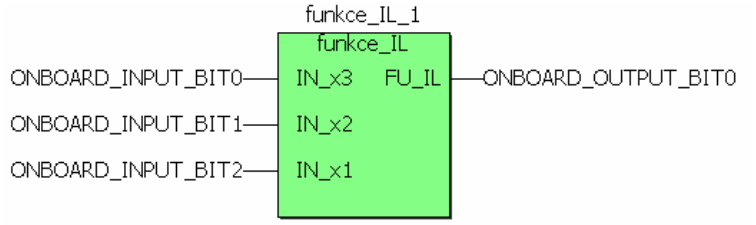
Obr.41 ukazuje zdrojový kód programu pro funkční blok. Popis částí kódu je umístěn ve formě komentářů hned za kód.

	Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC	TB	I/Q	Connection Point
	⊟ **Default**											
	funkce_IL_1	funkce_IL	VAR				☐	☐	☐	☐		
	ONBOARD_INPUT_BIT0	BOOL	VAR_EXTERNAL	Lokaler Eingang IN1			☐	☐	☐	☐		
	ONBOARD_INPUT_BIT1	BOOL	VAR_EXTERNAL	Lokaler Eingang IN2			☐	☐	☐	☐		
	ONBOARD_INPUT_BIT2	BOOL	VAR_EXTERNAL	Lokaler Eingang IN3			☐	☐	☐	☐		
	ONBOARD_OUTPUT_BIT0	BOOL	VAR_EXTERNAL	Lokaler Ausgang OUT1			☐	☐	☐	☐		

Obr.42:Tabulka proměnných

Tabulka proměnných na obr.42 se liší od obr.36 pouze jednou proměnnou navíc, která reprezentuje nově vytvořený blok. Tyto proměnné jsou generovány při kompilaci

5.2.3.6 Výsledný blok



Obr.43:IL blok

Na obr.43 je blok, který byl vytvořen pomocí sekce „Funkce_IL“(obr.39). Je vložen do „Main“ a jsou k němu připojeny lokální vstupy a výstup. V tomto případě to nemá na funkci programu vliv, ale je třeba dávat pozor, v jakém pořadí se generují vnitřní proměnné zapsané ručně při programování bloku. Na tomto obrázku je jasně vidět, že jsou generovány v opačném pořadí, než jsou poté připojeny vstupy. Může tedy snadno dojít k záměně vstupů a program nebude pracovat správně.

5.3 Realizace funkčního bloku „Flip Flop“

5.3.1 Popis

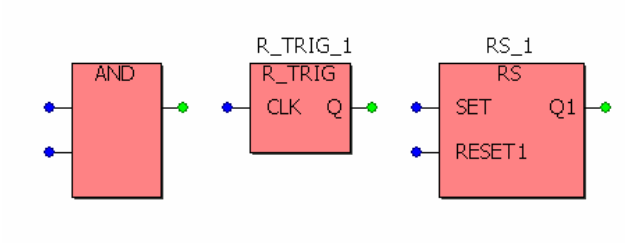
Úloha zaměřená na tvorbu vlastního funkčního bloku, který realizuje inverzi výstupu v závislosti na náběžné hraně na vstupu. Úloha v sobě kombinuje použití RS klopného obvodu a členu, který detekuje náběžnou hranu.

5.3.2 Zadání

Naprogramujte funkční blok v prostředí PC WorX, který může být použit v dalších projektech. Jeho funkčnost přímo ověřte na PLC od fy Phoenix Contact. Programování bloku proveďte pomocí FBD a LD.

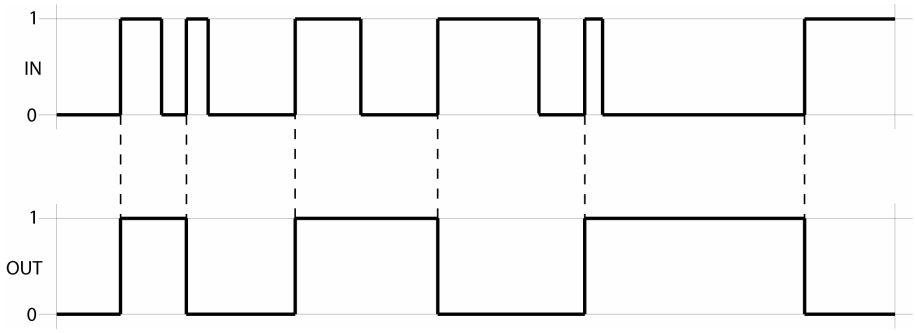
Funkce bloku je následující:Výstup(standartně logická 0) bloku je invertován, pokud je na vstupu detekována náběžná hrana(logická 1) impulsu.

Pro programování bloku použijte následující prvky na obr.44



Obr.44:Prvky

Blok bude pracovat podle tohoto časového průběhu(obr.45)



Obr.45:Časový průběh

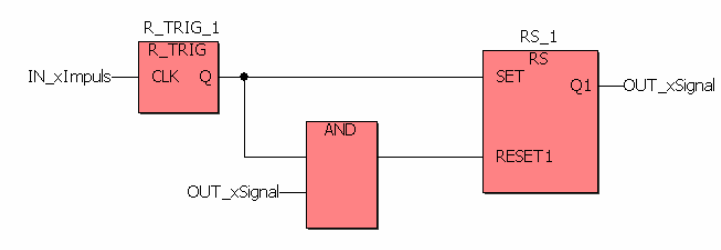
5.3.3 Řešení

5.3.3.1 FBD realizace

	Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC	TB
	Default									
	R_TRIG_1	R_TRIG	VAR				☐	☐	☐	☐
	RS_1	RS	VAR				☐	☐	☐	☐
	IN_xImpuls	BOOL	VAR_INPUT				☐	☐	☐	☐
	OUT_xSignal	BOOL	VAR_OUTPUT				☐	☐	☐	☐

Obr.46:Tabulka proměnných

Tabulka proměnných na obr.46 obsahuje proměnné, které jsou potřeba k chodu podprogramu na obr.47. Jsou to opět vnitřní proměnné a vyskytují se jen v rámci tohoto podprogramu. Opět je nutné je zapsat ručně, protože při kompilaci se generují proměnné pouze v „Main“.



Obr.47:FBD

Obr.47 ukazuje schéma podprogramu, vytvořeného pomocí FBD. Obsahuje 3 funkční bloky. Blok AND není třeba blíže popisovat.R-TRIG je funkční blok, který umožňuje detekovat náběžnou hranu signálu. Jestliže je detekována na vstupu bloku náběžná hrana, tak na výstupu se místo logické 0 generuje logická 1. Jestliže je blok zavolán v programu úplně poprvé, tak generuje na výstupu logickou 0 dokud není detekována první náběžná hrana.

Parametry bloku:

- CLK – datový typ BOOL a detekuje pouze náběžnou hranu
- Q - datový typ BOOL, při detekování náběžné hrany mění logickou 0 na logickou 1

Blok RS je základní obvod, který je schopen setrvat v určitém stavu (logické 0 nebo 1) bez aplikace vnějších logických úrovní (mimo napájecí napětí ovšem). Jestliže je SET logická „1“, tak i výstup Q1 je „logická 1“. Q1 se nemění ani když SET přechází v logickou „0“ .Q1 je resetován, jestliže RESET1 nabývá logické „1“. Jestliže jsou oba vstupy logické „1“, tak Q1 je nastaven RESET1 na logickou „0“.

Parametry bloku:

- SET – datový typ BOOL
- RESET1 - datový typ BOOL
- Q1 – datový typ BOOL

Chování programu je následující. Pokud na blok R_TRIG, který detekuje náběžnou hranu, přivedeme signál s náběžnou hranou, tak na výstupu se vytvoří jednotkový impuls. Tento impuls přijde vstup SET RS obvodu a na výstupu Q1 nastaví 1. Tento výstup je zároveň vstupem pro hradlo AND, která má ovšem na výstupu 0, takže vstup RESET je také nula. Při další náběžné hraně se ovšem na výstupu hradla AND objeví 1. Výstup Q1 totiž stále generuje 1 a na druhý vstup AND přichází jednotkový impuls. Na vstup RESET je tedy přivedena 1 a výstup Q1 je nastaven na 0.

	Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC	TB	I/Q
	Default										
	FB_TFlipflop_FBS_1	FB_TFlipflop_FBS	VAR				☐	☐	☐	☐	
	ONBOARD_INPUT_BIT0	BOOL	VAR_EXTERNAL	Lokaler Eingang IN1			☐	☐	☐	☐	
	ONBOARD_OUTPUT_BIT0	BOOL	VAR_EXTERNAL	Lokaler Ausgang OUT1			☐	☐	☐	☐	

Obr.48:Tabulka proměnných bloku

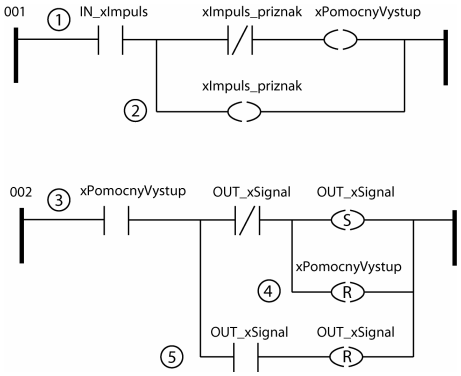
Na obr.48 je tabulka proměnných pro výsledný blok , ve které se vyskytuje lokální proměnná, která reprezentuje vnitřní podprogram a vstupní a výstupní proměnná bloku. Vstup a výstup jsou opět lokální, protože jsou připojeny přímo na PLC.

5.3.3.2 LD realizace

	Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC	TB
	Default									
	IN_xImpuls	BOOL	VAR_INPUT				☐	☐	☐	☐
	xImpuls_priznak	BOOL	VAR				☐	☐	☐	☐
	xPomocny_vystup	BOOL	VAR				☐	☐	☐	☐
	OUT_xSignal	BOOL	VAR_OUTPUT				☐	☐	☐	☐

Obr.49:Tabulka proměnných

V tabulce na obr.49 jsou pouze lokální proměnné, které jsou nutné pro chod programu, který je schématicky znázorněn na obr.50. Tyto proměnné se vyskytují pouze v rámci podprogramu, který je napsán pro funkční blok.



Obr.50:LD

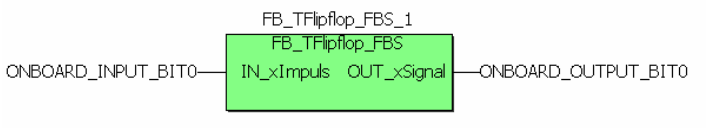
Na obr.50 je nakresleno schéma podprogramu. Samotný screenshot v tom případě není názorný. Podprogram má dvě větvě. Jedna slouží pro detekci náběžné hrany a druhá

vykonává nastavení výstupu a resetování pomocného výstupu, který byl zaveden uměle do větvě, která realizuje detekci náběžné hrany.

Větev, která realizuje detekci náběžné hrany je na obr.50 označena jako síť 001. Přímá větev programu se chová následovně. Aby došlo k sepnutí cívky „Pomocny_vystup“ je potřeba, aby byl sepnut kontakt „IN_xImpuls“ a zůstal rozepnut kontakt „xImpuls_priznak“. V dalším kroku programu je sepnuta cívka „xImpuls_priznak“, která zabrání průchodu signálu kontaktem „xImpuls_priznak“. Tímto je vyřešena detekce náběžné hrany pomocí impulsu.

Síť 002 se začne vykonávat, jestliže v síti 001 byla sepnuta cívka „xPomocny_vystup“. Pokud je výstup „OUT_xSignal“ ve stavu logické nuly, tak se provede přímá část programu a „OUT_xSignal“ se nastaví do logické 1. V dalším kroku dojde k resetování „xPomocny_vystup“. Pokud se výstup nalézal ve stavu logické 1, tak dojde k resetu „OUT_xSignal“. Resetováním „xPomocny_vystup“ se předešlo tomu, aby se při jedné náběžné hraně neměnil výstup víckrát.

5.3.3.3 Výsledný blok



Obr.51: Výsledný blok

Obr.51 prezentuje konečnou fázi v programování bloku. Blok má jeden vstup a výstup. Dle názvu vstupu/výstupu lze poznat, že vstup i výstup jsou připojeny lokálně přímo k PLC.

5.4 Realizace bloku „Clock Cascade“

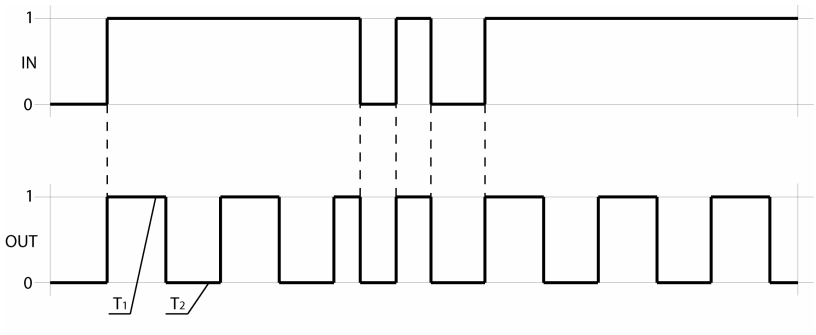
5.4.1 Popis

Úloha zaměřená na tvorbu vlastního funkčního bloku, který využívá časovačů-v tomto případě s pozdním sepnutím. Úloha je zaměřena na demonstraci využití různých typů jazyka s ohledem na složitost úlohy.

5.4.2 Zadání

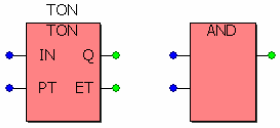
Naprogramujte funkční blok v prostředí PC WorX, který může být použit v dalších projektech. Jeho funkčnost přímo ověřte na PLC od fy Phoenix Contact. Programování bloku realizujte pomocí FBD, IL a ST.

Funkce bloku je následující:Pokud bude na vstupu detekována náběžná hrana po dostatečně dlouhou dobu, tak na výstupu se bude generovat signál obdélníkového charakteru. O jeho průběhu se lze názorně informovat z diagramu na následujícím obrázku.



Obr.52:Časový průběh

Pro programování bloku v FBD použijte prvky viz. obr.53



Obr.53:Prvky

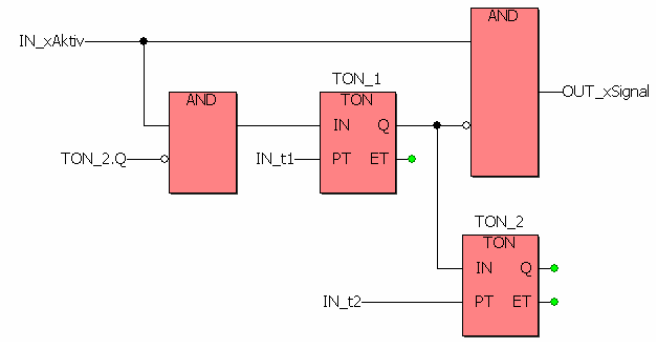
5.4.3 Řešení

5.4.3.1 FBD realizace

	Name	Type	Usage	Description	Address	Init	Retain	PDD
	Default							
	OUT_xSignal	BOOL	VAR_OUTPUT					
	IN_t1	TIME	VAR_INPUT					
	IN_t2	TIME	VAR_INPUT					
	TON_1	TON	VAR					
	TON_2	TON	VAR					
	IN_xAktiv	BOOL	VAR_INPUT					

Obr.54:Tabulka proměnných bloku

Tabulka proměnných bloku na obr.54 je v tomto případě stejná pro všechny jazyky. Obsahuje celkem 6 položek. Vstupní, výstupní dále 2 časové vstupní hodnoty a 2 vnitřní hodnoty pro časovače s pozdním sepnutím.



Obr.55:FBD

Na obr.55 je FBD schéma pro podprogram bloku. Obsahuje pouze 2 typy bloků. Jedná se o AND a blok s pozdním sepnutím TON.

TON je funkční blok realizují pozdní sepnutí. Jestliže se na vstupu (IN) detekuje náběžná hrana, tak se vstup přepne na PT a začne odpočet časové prodlevy. Po uplynutí stanovené doby se na výstupu Q objeví logická jednička za předpokladu, že mezitím nedojde ke změně na vstupu IN. Čas, který mezitím uběhl z celkově stanovené prodlevy je indikován na výstupu ET. IN a Q lze negovat.

Parametry bloku:

- IN – datový typ BOOL
- PT - datový typ TIME , slouží pro nastavení doby zpoždění
- Q – datový typ BOOL, logická jednička je na výstupu pouze tehdy, pokud je logická jednička na vstupu IN a ET je vetší nebo rovno PT
- ET - datový typ TIME

Funkce programu je dle obr.54 jednoduchá a není třeba ji více rozepisovat. Jazyk FBD je velmi názorný v tomto případě.

5.4.3.2 IL realizace

```
1    LD       IN_xAktiv    (*Načtení vstupního signálu na první pozici střadače*)
2    ANDN     TON_2.Q      (*Načtení negovaného výstupu z bloku TON_2 a jeho
3                          logický součin s předchozí pozicí ve střadači*)
4    ST       TON_1.IN     (*Uložení hodnoty předchozí operace na vstup bloku TON_1*)
5
6    LD       IN_t1        (*Načtení hodnoty vstupu IN_t1... *)
7    ST       TON_1.PT     (* ...a její uložení na vstup TON_1.PT*)
8
9    CAL      TON_1        (* "Zavolání" funkce bloku TON_1 *)
10
11   LD       TON_1.Q      (*Načtení hodnoty výstupu TON_1.Q ... *)
12   ST       TON_2.IN     (* ... a její uložení na vstup TON_2.IN*)
13
14   LD       IN_t2        (*Načtení hodnoty vstupu IN_t2... *)
15   ST       TON_2.PT     (* ... a její uložení na vstup TON_2.PT*)
16
17   CAL      TON_2        (* "Zavolání" funkce bloku TON_2 *)
18
19   LD       IN_xAktiv    (*Načtení vstupního signálu *)
20   ANDN     TON_1.Q      (*Načtení negovaného výstupu z bloku TON_1.Q
21                          a jeho logický součin s předchozí pozicí střadače*)
22   ST       OUT_xSignal (*uložení výsledku operace na poslední pozici střadače*)
```

Obr.56:IL

Na obr.56 je zdrojový kód jazyka IL a ve formě komentářů je doplněn o vysvětlení každého řádku kódu. Kód je rozdělen mezerami pro lepší přehlednost, ale je možné psát kód bez mezer.

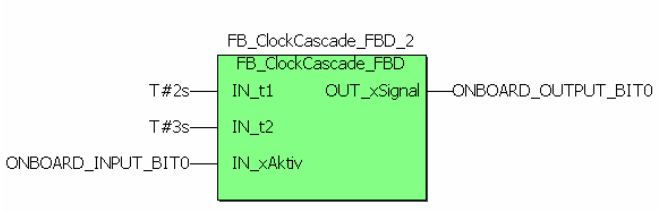
5.4.3.3 ST realizace

```
1    TON_1(IN:=IN_xAktiv & Not TON_2.Q,
2            PT:=IN_t1);
3
4    TON_2(IN:=TON_1.Q,
5          PT:=IN_t2);
6
7    OUT_xSignal := IN_xAktiv & Not TON_1.Q;
```

Obr.57:ST

Obr. 57 zobrazuje kompletní kód v jazyce ST. Taktéž jsou odděleny mezerami části kódu, které spolu souvisí pro větší přehlednost. Celý kód je možno v tomto případě zapsat na 3 řádky. Pro tuto úlohu se žádný jiný jazyk nehodí více než právě ST. Pro složitější úlohy je ST tou pravou volbou, především pro svou jednoduchost, která je prezentována na tomto příkladu.

5.4.3.4 Výsledný blok



Obr.58:Výsledný blok

Výsledný produkt předchozích kroků lze vidět na obr.58, kde je již blok připojen opět na lokální vstup i výstup. Kromě toho má dva časové vstupy, kde se čas zadává ve formátu T#*s, kde hvězdička je nahrazena číslicí. Tabulka proměnných zde nebude uvedena, protože obsahuje lokální proměnnou bloku a vstup a výstup.

5.5 Realizace řízení semaforu

5.5.1 Popis

Úloha zaměřená na prezentaci SFC schématu a využití sekvenčního způsobu programování .

5.5.2 Zadání

Pomocí sekvenčního diagramu SFC naprogramujte funkční blok v prostředí PC WorX, který může být použit v dalších projektech. Jeho funkčnost přímo ověřte na PLC od fy Phoenix Contact. Funkcí bloku bude jednoduché řízení semaforu bez poruchového chodu. K úloze využijte školní modul simulující chod křižovatky a jeho zapojení konzultujte s vyučujícím. Modul připojte k jednomu z modulů se 4 digitálními výstupy.

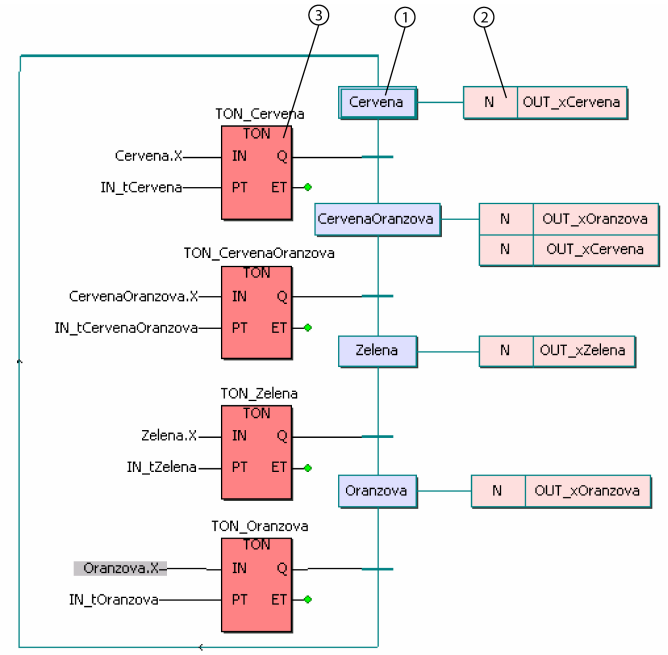
5.5.3 Řešení

5.5.3.1 Realizace SFC

	Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC	TB
	Default									
	TON_Cervena	TON	VAR							
	TON_Oranzova	TON	VAR							
	TON_Zelena	TON	VAR							
	TON_CervenaOranzova	TON	VAR							
	IN_tCervena	TIME	VAR_INPUT							
	IN_tOranzova	TIME	VAR_INPUT							
	IN_tZelena	TIME	VAR_INPUT							
	IN_tCervenaOranzova	TIME	VAR_INPUT							
	OUT_xCervena	BOOL	VAR_OUTPUT							
	OUT_xOranzova	BOOL	VAR_OUTPUT							
	OUT_xZelena	BOOL	VAR_OUTPUT							

Obr.59:Tabulka proměnných bloku

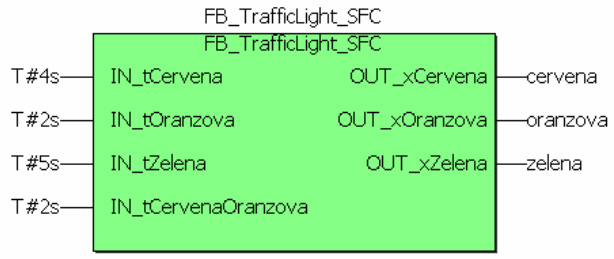
Na obr.59 je vyobrazena tabulka proměnných, kterou ke svému chodu potřebuje podprogram na obr.60. Obsahuje 4 časové vstupní proměnné, 4 proměnné pro časovače s pozdním sepnutím a 3 výstupní logické proměnné. Všechno jsou to lokální proměnné a vystupují pouze v rámci podprogramu.



Obr.60: SFC

SFC schéma na obr.60 obsahuje 4 samostatné sekvence. Jako inicializační krok je zvolen „Cervena“(1). Přechod této sekvence je dán časovačem(3) s pozdním sepnutím. V okamžiku, kdy program dorazí na tento přechod, tak se spustí pozdní sepnutí časovače, které je dáno lokální časovou proměnnou. Po celou dobu akční doby této sekvence je aktivní i „Akce“(2). Než program přejde k dalšímu přechodu svítí červená. Program pokračuje cyklicky ve směru šipek.

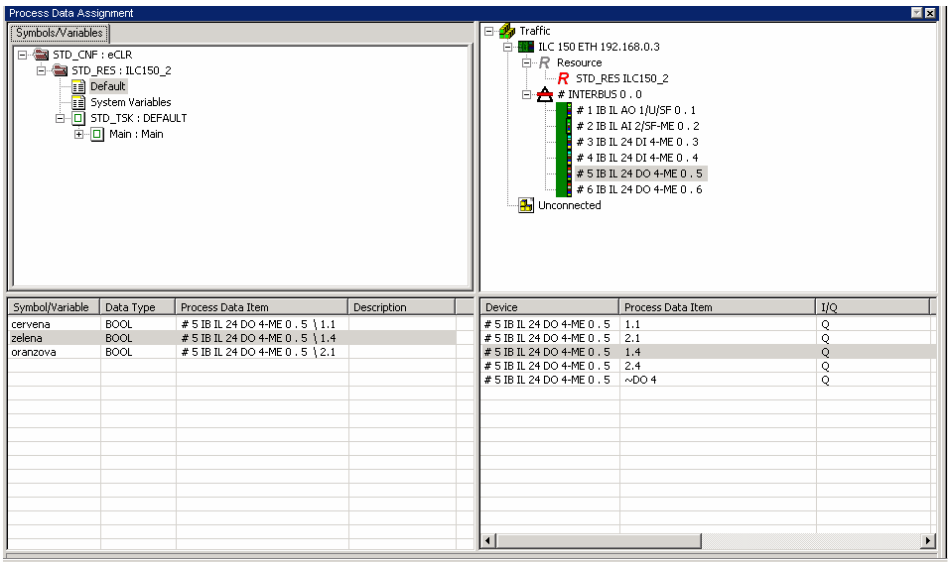
5.5.3.2 Výsledný blok



Obr.61: Výsledný blok SFC

Na obr.61 je výsledný blok, který má 4 lokální vstupy a 3 globální výstupy. Nebude zde uvedena tabulka proměnných „Main“, kde se vyskytuje pouze proměnná bloku a 3

globální proměnné, které reprezentují výstupy z tohoto bloku. Tyto výstupy jsou připojeny dle obr.62

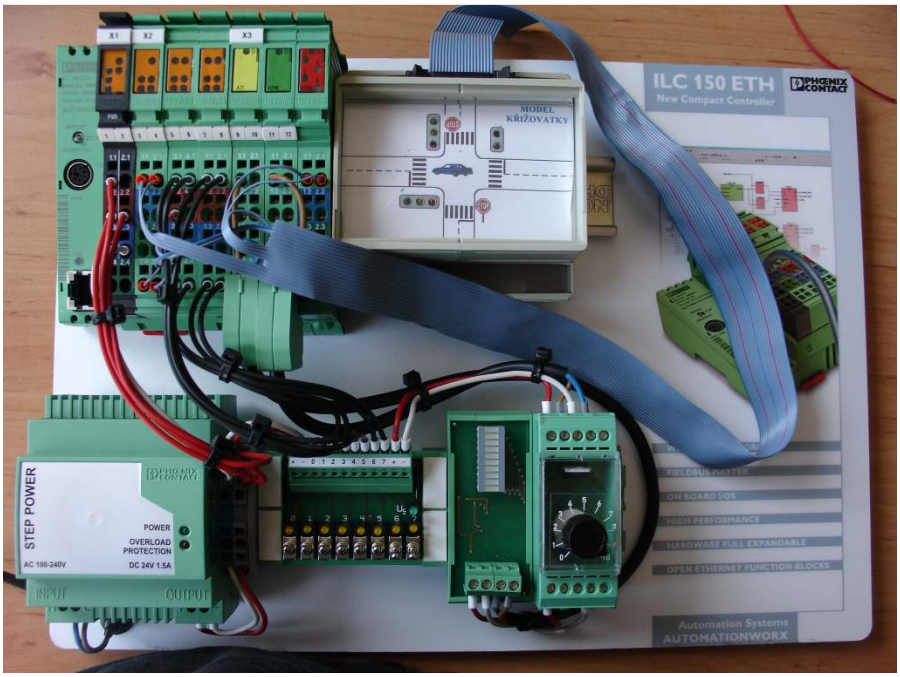


Obr.62:Přiřazení proměnných

Kapitola 4.9 pojednává o přiřazování globálních proměnných a proto zde je potřeba pouze zmínit to, že je nutno dávat pozor, kterou proměnnou kam přiřadíme.

5.5.4 Modifikace

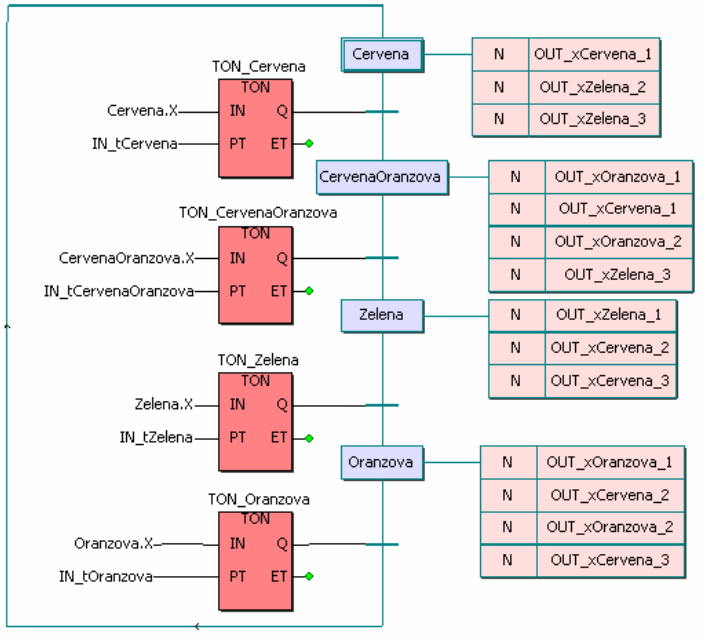
Proveďte úpravu SFC schématu, tak aby bylo možno ovládat kompletně celý model křižovatky na obr.63 .



Obr.63:Model křižovatky

Pro připojení křižovatky zvolte dva moduly s digitálními výstupy.

5.5.4.1 Realizace SFC



Obr.64:SFC

Kostra schématu zůstala stejná, pouze byly přidány akce. Akce jsou přiřazeny dle tabulky na obr.65. Jako Semafor 1 je zvolen semafor, který je na hlavní silnici. Semafor 2 je na vedlejší silnici a Semafor 3 je semafor pro chodce, kteří přecházejí hlavní silnici.

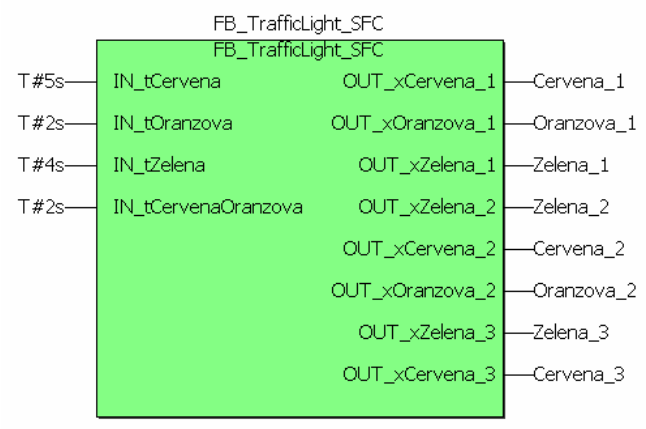
	Semafor 1	Semafor 2	Semafor 3
Krok 1	Červená	Zelená	Zelená
Krok 2	Červená+Oranžová	Oranžová	Zelená
Krok 3	Zelená	Červená	Červená
Krok 4	Oranžová	Červená+Oranžová	Červená

Obr.65:Tabulka křižovatky

Kompletní tabulka pro blok realizovaný pomocí SFC je na obr.66. V porovnání s obr. 59 je rozdíl pouze ve výstupních proměnných, kterých přibylo 5 v závislosti na akcích.

Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC	TB
Default									
TON_Cervena	TON	VAR							
TON_Oranzova	TON	VAR							
TON_Zelena	TON	VAR							
TON_CervenaOranzova	TON	VAR							
IN_tCervena	TIME	VAR_INPUT							
IN_tOranzova	TIME	VAR_INPUT							
IN_tZelena	TIME	VAR_INPUT							
IN_tCervenaOranzova	TIME	VAR_INPUT							
OUT_xCervena_1	BOOL	VAR_OUTPUT							
OUT_xOranzova_1	BOOL	VAR_OUTPUT							
OUT_xZelena_1	BOOL	VAR_OUTPUT							
OUT_xZelena_2	BOOL	VAR_OUTPUT							
OUT_xCervena_2	BOOL	VAR_OUTPUT							
OUT_xOranzova_2	BOOL	VAR_OUTPUT							
OUT_xZelena_3	BOOL	VAR_OUTPUT							
OUT_xCervena_3	BOOL	VAR_OUTPUT							

Obr.66: Tabulka proměnných bloku



Obr.67: Výsledný blok SFC

Výsledná úprava programu se promítla do bloku následovně. Na obr. 67 lze pozorovat zvýšený počet výstupů, na které jsou připojeny globální proměnné. Proměnné tohoto bloku v „MainV“ obsahují všechny výstupy, které jsou globální, protože jsou připojeny na Interbus moduly.

6 Závěr

Cílem této práce bylo seznámit se s vývojovým prostředím PC WorX a programovatelnými automaty fy Phoenix Contact. Dále bylo potřeba popsat toto vývojové prostředí, tak aby bylo použitelné jako výukový podklad pro podporu výuky. Jedním z hlavním cílů bylo vytvořit soubor několika praktických úloh odlišné obtížnosti, které by umožnily vyzkoušení schopností (především množství dostupných programovacích jazyků) programovatelného automatu a také možnost vyzkoušet si programování jazyky normy IEC 61131-3.

Práce byla rozdělan na 4 části. V úvodu byla krátce popsána současná situace v automatizační technice a situaci PLC v oblasti automatizace. Byla zmíněna konstrukce PLC, jejich využití a možnosti, které v současné době nabízejí.

Cílem druhé části bylo blíže seznámit čtenáře s programovatelným automatem fy Phoenix Contact ILC 150 ETH. Tento automat byl ve formě Starter Kitu, který obsahoval i přídavné moduly, které byly rovněž blíže popsány. Tento Starter Kit byl rovněž doplněn o nové moduly z nichž právě dvojice modulů s digitálními výstupy se výborně hodila pro realizaci jedné z úloh-ovládání křižovatky. V této kapitole získal čtenář informace technického charakteru o tomto PLC a jeho modulech. Modulů existuje široká škála a proto zde byly zmíněny jen ty, se kterými se lze fyzicky setkat v laboratoři na UAI.

Pro popis vývojového prostředí byla zvolena především grafická forma, tedy „screenshoty“, které byly doplněny vysvětlujícím textem. Tento popis byl vytvořen tak, aby pomocí něj byl schopen čtenář vytvořit funkční program. Cílem této části práce bylo vytvoření uceleného manuálu použitelného pro podporu výuky.

Poslední část práce byla věnovaná praktickým úlohám. Jejich cílem byla možnost praktického otestování programovacích jazyků. Byly vytvořeny čtyři praktické úlohy.

První z nich byla velice triviální a byla zaměřena na realizaci logické funkce, která byla dána tabulkou. Úkolem studenta bylo provést minimalizaci této funkce DNF nebo KNF, dále minimalizace Karnaughovou mapu. Dalším úkolem bylo nakreslit blokové schéma pomocí bloků negací, konjunkcí a disjunkcí. Zde se naskytla možnost porovnání tohoto schématu se schématem, které bylo vytvořeno pomocí FBD jazyka. Tato úloha byla dále realizována pomocí LD a IL.

Druhá úloha byla zaměřena na prvky detekující náběžnou hranu a RS-klopné obvody s cílem vytvořit jednoduchý program právě s těmito prvky. Program vykonával jednoduchou funkci, kdy s každou náběžnou hranou docházelo k invertování výstupní hodnoty. Tento program byl realizován jazyky FBD a LD.

Třetí úloha byla věnována časovačům s pozdním sepnutím. Program měl na výstupu generovat určitý obdélníkový průběh, který byl dán grafickým zadáním, pokud byla na vstup přivedena logická jednička. Úloha byla realizována pomocí jazyků FBD, IL a ST.

Poslední úloha byla věnována SFC jazyku a byla rozdělena na dvě části, kde druhá část byla modifikací původní úlohy a měla poukázat na velice snadnou rozšiřitelnost programu v tomto jazyce. Jednalo se o řízení semaforu, kde v první části bylo úkolem vytvořit schéma, podle kterého se měl řídit jeden semafor. Jednoduchou modifikací bylo možno úlohu snadno rozšířit na ovládání celé křižovatky, která obsahovala dva semafory pro dva různé směry a jeden semafor pro chodce.

Poslední část práce byla obohacena o velké množství obrázků, schémat, tabulek atd. za účelem snadnějšího pochopení programů.

Cíle práce se podařilo splnit v celém rozsahu.

Použitá literatura

[1] Šmejkal, L., Martinásková, M., PLC a automatizace, Praha: BEN, 2007
[2] Martinásková, M., AUTOMATIZACE[online]. 2004, červen [cit. 2. května 2008]. Dostupné na WWW: http://www.automatizace.cz/article.php?a=142
[3] Pantůček , E., Automa[online]. 2006, [cit. 10. ledna 2008].
Dostupné na WWW: http://www.odbornecasopisy.cz/index.php?id_document=31558
[4] Firemní materiály o programovatelných automatech fy Phoenix Contact. [online]. last modified: 16th of Januar, 2008, [cit. 3. května 2008].
Dostupné na WWW: http://www.phoenixcontact.cz/

Seznam příloh

Datový nosič CD-R obsahující:

- Funkční programy v komprimované verzi
- Bakalářskou práci v elektronické formě