Emulátor vývojového prostředí pro programování PLC Saia

Emulator of developmental environment for PLC Saia programming

Bc. Stanislav Pěrka

Diplomová práce
2008

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2007/2008

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Stanislav PĚRKA**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Emulátor vývojového prostředí pro programovaní PLC Saia**

Zásady pro vypracování:

1. **Popište různé přístupy k programování programovatelných automatů (PLC) SAIA.**
2. **Vytvořte přehled jednotlivých knihoven a zdůrazněte funkce při výuce nejčastěji užívané v programovém prostředí SAIA PG5.**
3. **Vytvořte program umožňující vytvářet programy stejným způsobem jako v prostředí PG5.**
4. **Realizujte modul programu, který bude simulovat běžící program a jehož pomocí bude možno sledovat vstupy a výstupy na simulovaném PLC.**
5. **Připravte vzorové zadání několika úloh využitelných ve výuce.**
6. **Zpracujte podrobný manuál k programu.**

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **Brown, Charles, E.: The essential guide to Flex 2 with ActionScript 3.0, Berkeley Apress, 2007.**
2. **Lott, J., Schall, D., Peters, K.: ActionScript 3.0 Cookbook, Adobe Developer Library, 2006.**
3. **Sanders, W., Cumaranatunge, C.: ActionScript 3.0 Design Patterns, Adobe Developer Library, 2006.**
4. **Flash.cz -- server pro kreativní lidi. URL: http://www.flash.cz/portal/clanky.aspx?sekce=30.**
5. **Firemní literatura k PLC Saia.**
6. **Martinásková, M., Šmejkal, L.: Řízení programovatelnými automaty, Vydavatelství ČVUT, Praha, 1998.**
7. **Martinásková, M., Šmejkal, L.: Řízení programovatelnými automaty II, Vydavatelství ČVUT, Praha, 2000.**
8. **Šmejkal, L., Martinásková, M.: PLC a automatizace, Nakladatelství BEN – technická literatura, Praha, 1999.**

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

Datum zadání diplomové práce: **20. února 2008**

Termín odevzdání diplomové práce: **19. května 2008**

Ve Zlíně dne 20. února 2008

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Předkládaná diplomová práce si klade za cíl vyřešit nedostatek programovacího rozhraní Saia PG5 v nemožnosti simulace vytvořených aplikací bez fyzického propojení s reálným programovatelným automatem Saia PCD. Je tedy nutné vytvořit nové programovací rozhraní, které bude co nejvíce podobné vývojovému prostředí Saia PG5 a bude vněm možné programovat v jazyce FUPLA a simulovat vytvořené programy bez použití reálného automatu.

Klíčová slova: Saia PG5, Emulátor PG5, programovatelný automat

ABSTRACT

Aim of graduation thesis is solving poverty of programming environmental Saia PG5 in impossibility simulation of created applications without direct connection with real programmable automat Saia PCD. Its necessary to create new program interface, which will be really similar like Developmental environment Saia PG5 and possible programming in language FUPLA will be involved, also simulating of created programs without using real automat should be possible.

Keywords: Saia PG5, emulator PG5, programmable automat

Tímto bych chtěl poděkovat vedoucímu své práce panu Ing. Tomáši Sysalovi Ph.D za odborné vedení a spolupráci během řešení mé práce. A dále bych chtěl poděkovat celé své rodině za poskytnuté zázemí a podporu.

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

K vytváření uživatelských programů pro celý sortiment volně programovatelných automatů Saia®PCD se používá programovací komplet Saia®PG5. Programovací komplet Saia®PG5 obsahuje programovací jazyky, IL, Graftec, FUPLA, KOPLA. Programovací rozhraní bohužel neobsahuje simulaci chování programovatelného automatu Saia®PCD, tak je nutné při každé změně v programu, tento nahrát do automatu a až po té je možné sledovat jeho chování. Proto bylo nutné vytvořil software pro simulaci chování programovatelného automatu Saia®PCD pro jazyk FUPLA. V tomto prostředí je možné pomocí jazyka FUPLA vytvořit program a v reálném čase simulovat programové pochody, které by jinak nebylo možné odzkoušet bez fyzického propojení s automatem. To je velice vhodné například pro studenty, kteří si můžou rychle a jednoduše odzkoušet logiku tvorby programu v jazyce FUPLA a dopředu připravit strukturu programu, než ho fyzicky odzkouší na reálném automatu.

I. TEORETICKÁ ČÁST

1 PROGRAMOVATELNÉ AUTOMATY

Programovatelné automaty (PLC) jsou elektronická zařízení, která slouží pro řízení strojů nebo procesů. Přijímají signály na vstupech, zpracovávají je v souladu s programem a následně posílají signály na výstupy. Program se vytváří pomocí programovacího software, který je schopen vytvořit vazbu mezi vstupy a výstupy v jakékoli požadované posloupnosti, měřit čas nebo provádět výpočetní operace. Podstatnými parametry PLC jsou maximální počet vstupů/výstupů, kapacita paměti a rychlost zpracování výpočtů[6].

1.1 Programování PLC

1.1.1 Centrální jednotka

Poskytuje programovatelnému automatu inteligenci. Realizuje soubor instrukcí a systémových služeb, zajišťuje i základní komunikační funkce s vlastními i vzdálenými moduly, s nadřízeným systémem a s programovacím přístrojem. Paměťový prostor, který poskytuje uživateli je obvykle rozdělen na části. Prvá je určena pro uložení uživatelského program (PLC programu), datových bloků a tabulek. Její obsah se zadává v edičním režimu a během vykonávání programu se obvykle nemění (u "TECOMAT" se může měnit jen obsah tabulek). Druhá část je operační (zápisník). Jsou v ní lokalizovány uživatelské registry, čítače a časovače, obrazy vstupů a výstupů, komunikační, časové a jiné systémové proměnné (systémové registry). Obsah operační části se dynamicky mění působením uživatelského a systémového programu. Centrální jednotky současných programovatelných automatů obsahují mikroprocesor, mikrořadič nebo specializovaný řadič, zaměřený na rychlé provádění instrukcí. Jeho programem (systémovým programem) jsou realizovány všechny funkce, které má uživatel k dispozici, tj. kompletní soubor instrukcí programovatelného automatu, jeho systémové služby, časové a komunikační funkce. Jen výjimečně a za omezujících podmínek zpřístupňují některé PLC uživatelům programování na úrovni instrukcí mikroprocesoru. Typické je použití instrukcí a příkazů jazyka programovatelného automatu, který je přizpůsoben převažujícím úlohám a způsobu myšlení typického uživatele a programátora PLC[6].

1.1.2 Soubor instrukcí PLC

Protože programovatelné automaty byly původně určeny k realizaci logických úloh a k náhradě pevné logiky, nechybějí v žádném PLC instrukce pro základní logické operace s bitovými operandy. Tj. sejmutí pravdivostní hodnoty adresovaného bitu, operace logického součtu a součinu, negace, výlučného součtu a jiných kombinačních logických funkcí, instrukce pro realizaci paměťových funkcí a klopných obvodů, pro zápis výsledku a mezivýsledku na adresované místo, ale i instrukce čítačů, časovačů, posuvných registrů, krokových řadičů a jiných funkčních bloků. Současné PLC nabízejí instrukční soubor podstatně bohatší. V souboru instrukcí vyspělých PLC obvykle nechybí ani instrukce pro aritmetiku a operace s čísly (někdy jen nejzákladnější, např. sčítání, odčítání a porovnávání, jindy kompletní knihovny pro výpočty s pevnou nebo plovoucí řádovou čárkou), logické instrukce s číselnými operandy (paralelní operace s operandem v délce byte, slova nebo delším) a přenosy dat. Obvyklé jsou instrukce pro organizaci programu (např. skoky v programu, volání podprogramů a návraty)[6].

1.1.3 Výkonnost programovatelného automatu

se nejčastěji posuzuje podle doby vykonání instrukcí. Obvyklé hodnoty pro výkonné systémy jsou v řádu jednotek mikrosekund na instrukci, výjimečně méně (někdy se uvádějí doby na 1 000 instrukcí). U malých systémů bývají časy řádově jednotky až desítky mikrosekund na instrukci. Zde je třeba uvést, že toto hodnocení je mnohdy zavádějící. Uváděné časové údaje obvykle odpovídají jen nejzákladnějším logickým instrukcím. Doby ostatních instrukcí bývají mnohonásobně delší (často desetkrát i stokrát). Výpočetní výkon PLC zvyšují instrukce, které realizují komplexní funkční bloky (krokové řadiče, aritmetické knihovny pro práci v pevné nebo v pohyblivé řádové čárce, PID regulátor, fuzzy regulátor, generátor funkcí) nebo dovolují efektivní zpracování souborů dat a datových struktur. Významnou pomoc přestavují systémové služby. Odezvu systému na kritické události zkracuje i možnost přerušení a multiprogramování[6].

Výkonnost stejného systému bude záviset na typu převažujících úloh a na prostředcích, které k tomu použijeme, na míře kvalifikovanosti programátora, jak výkonné instrukce a postupy použije a které ze systémových služeb využijeme. Proto se doporučuje výkonnost PLC posuzovat nikoliv podle katalogových údajů, ale podle výsledků řešení reálných příkladů a typových úloh, zpracovaných kvalifikovaným programátorem[6].

1.2 Vykonávání programu PLC

1.2.1 Uživatelský program, cyklická aktivace

Program PLC je posloupnost instrukcí a příkazů jazyka. Typickým režimem jeho aktivace je cyklické vykonávání v programové smyčce. Na rozdíl od jiných programovatelných systémů se programátor PLC nemusí starat o to, aby po konci programu vrátil jeho vykonávání opět na začátek - zajistí to již systémový program. Naopak každé dlouhodobé setrvání programu v programové smyčce je "fatální chybou" a systém jej hlásí jako "překročení doby cyklu". Cyklické vykonávání programu je schematicky znázorněno na obrázku(obr. 1). Vždy po vykonání poslední instrukce uživatelského programu (např. E 0) je předáno řízení systémovému programu, který provede tzv. otočku cyklu. V ní nejprve aktualizuje hodnoty výstupů a vstupů: hodnoty uložené dosud v paměti jako obrazy výstupů (registry Y) přepíše do registrů výstupních periferních modulů a hodnoty ze vstupních modulů okopíruje do paměťových obrazů vstupů (registry X). Dále aktualizuje časové údaje pro časovače a systémové registry, ošetří komunikaci a provede ještě řadu režijních úkonů. Po otočce cyklu je opět předáno řízení prvé instrukci uživatelského programu (např. P 0)[6]

Obr. 1. Otočka cyklu

PLC program je vykonáván v cyklu. V jeho režijní části (v otočce cyklu) jsou nejprve na výstupy vyslány aktuálně vyčíslené hodnoty obrazů výstupů Y, pak jsou provedeny režijní operace systému, aktualizace systémových a časových proměnných, naplánována aktivace procesů pro další cyklus, ... apod.) a na závěr jsou sejmuty aktuální hodnoty , fyzických vstupů, které jsou pro celý následující cyklus konzervovány, jako obrazy vstupů X[6].

1.2.2 Obrazy vstupů a výstupů

Pro program PLC je typické, že nepracuje s aktuálními hodnotami vstupů a výstupů, ale s jejich "paměťovými konzervami" - s obrazy vstupů a výstupů, uloženými v zápisníkové paměti (regis-try X a Y). Aktualizace jejich hodnot - předání obrazů výstupů k řízenému objektu a sejmutí aktuálních vstupních hodnot od řízeného objektu se provede pouze ve fázi otočky cyklu. Tím je zajištěna synchronizace vstupních a výstupních dat s během programu a je tak omezena možnost chyb způsobených nevhodným souběhem měnících se hodnot (hazardních stavů). Obdobně jsou po dobu cyklu zmrazeny i časové údaje a hodnoty většiny systémových proměnných (například zpráv předávaných sériovou komunikací)[6].

1.3 Programovatelné automaty Saia PCD

Automaty Saia®PCD mají určité specifické vlastnosti, které je třeba zmínit.

1.3.1 Nevytváří se obraz procesu (process image)

V automatech Saia®PCD se negeneruje tzv. obraz procesu. S každým I/O se pracuje přesně v okamžiku, kdy ho uživatelský program čte nebo do něj zapisuje. Tento způsob přístupu k I/O signálům byl zvolen pro zajištění mnohem bezprostřednější a tím rychlejší komunikace mezi CPU a I/O a pro omezení interních přenosů dat. Ve výjimečných případech, kdy je vytvoření obrazu procesu výhodné pro odstranění možných hazardních stavů, ho lze snadno zajistit krátkou rutinou v uživatelském programu[7].

1.3.2 Integrovaný web server

Všechny novější stanice (uvedené na trh po roce 2000, tj. PCD2.M170, PCD2.M480, PCD3.Mxxxx, PCS1) mají v operačním systému integrován webový server. Pro přístup k webovému serveru je (v PC) potřebný program Web Connect a pro generování programu pro PCD využívajícího webový server je nutný program Web Builder. Oba tyto programové nástroje jsou součástí programovacího kompletu PG5 Controls Suite[7].

1.3.3 Komunikační protokol S-Bus

Komunikační protokol Saia®S-Bus je integrální součástí operačního systému každého automatu. Množství dalších protokolů je podporováno speciálními ovladači, které se začleňují v podobě tzv. FBoxů do uživatelského programu[7].

1.3.4 Komunikační moduly nevyžadují zvláštní prostor

Komunikační moduly a submoduly rozhraní se umisťují přímo na hlavní desku CPU, a nevyžadují tak žádný další prostor. Např.stanice PCD2.M480 může mít až 8 sériových rozhraní[7].

1.3.5 Komunikační moduly jsou vybaveny koprocesory

Pro komunikaci vysokými přenosovými rychlostmi a složitými protokoly jsou k dispozici speciální moduly, které pro tyto náročné úlohy využívají vlastní koprocesor[7].

1.3.6 Interní prvky automatu

Všechny automaty Saia®PCD obsahují totožné typy a počty interních prvků. Jsou to hodiny reálného času, poskytující informaci o hodinách, minutách, sekundách, dnu v měsíci, měsíc, roku, dnu v týdnu a týdnu v roce. Pro bitové informace je k dispozici 8192 tzv Flagů. Číselné informace (celá čísla i reálná čísla ve formátu Motorola s možností převodu do IEEE, ASCII znaky apod.) se ukládají do registrů, kterých je k dispozici 4096 a jsou 32bitové. V registrech mohou být i ASCII znaky nebo uživatelem definovaný obsah. Pro časování a čítání můžete využívat 1600 časovačů/čítačů s rozsahem 31 bitů. Všechny tyto prvky jsou umístěny ve zvláštní statické paměti RAM na desce CPU a mají své vlastní adresy[7].

1.3.7 Adresa I/O je dána jejich fyzickým umístěním

Všechny I/O staticky, podle své pozice na interní I/O sběrnici. Proto adresa každého použitého modulu je odvozena od jeho konkrétního umístění na určitou pozici. Každý modul má k dispozici 16 adres (i v případě, že nejsou všechny využité). Každá pozice pro modul má svou jedinečnou tzv.bázovou adresu[7].

1.3.8 Pro obsluhu analogových I/O modulů jsou k dispozici programové rutiny.

Analogové I/O, rychlé čítací moduly a moduly pro polohování se ovládají pomocí připravených programových rutin. Jsou to buď funkčních bloky, napsané v jazyce IL, nebo FBoxy pro grafické programování pomocí editoru Fupla. Tyto FB a FBoxy jsou součástí programovacího kompletu Saia®PG5 Controls Suite[7].

2 PG5

První automaty Saia® byly programovány bez počítačového programu. Jediným potřebným nástrojem byl přístroj s malou klávesnicí. Dalším krokem byl vývoj jednoduchého řádkového editoru pro operační systém PCM. V následujících desetiletích vzrostly nároky na programovací nástroje takovým způsobem, že dnes jsou srovnatelné s nároky na hardware jak důležitostí, tak objemem nákladů na jejich vývoj. Při tom je i současnými programovacími nástroji Saia® nejnovější generace stále ještě možné programovat i automaty, vyvinuté před 15 lety. Pro uživatele je tak zaručena možnost servisu a úprav programu jejich starého zařízení i po mnoha letech[8].

2.1 Programovací komplet - Saia®PG5.Controls Suite

Saia®PG5 je ústředním prvkem kompletu „Saia®PG5 Controls Suite“ a je mnohem víc, než jen dobrý servisní a programovací nástroj pro programátory PLC. Daleko překračuje funkce, požadované normou IEC 1131. Při programování v IL může být použit jako vývojový nástroj pro dedikované automaty, komunikační ovladače i IT funkce. Při tvorbě aplikačních programů pomáhají jeho grafické aplikační moduly uživatelům snadno implementovat i ty nejsofistikovanější automatizační úlohy, aniž by bylo nutné je programovat v jazycích KOPLA, IL či Graftec. Právě tímto způsobem se programuje většina aplikací pro Saia®PCD. Existující knihovny od společnosti Saia-Burgess a systémových partnerů poskytují výkonnou a komplexní základnu zejména pro projekty v oblasti automatizace infrastruktury. S pomocí programovacího nástroje Saia®FBox-Editor mohou být vyvíjeny vlastní grafické moduly, odpovídající specifickým potřebám určitých aplikací[8].

2.2 Hotové knihovny pro řízení a vizualizaci

Rozsáhlá knihovna pro TZB „Saia® DDC-Suite“ umožňuje významně snižovat pracnost a náklady při projektování a programování systému. Toho se dosahuje tím, že knihovna byla důsledně navržena jako objektově orientovaná. Dvě knihovny - pro řízení a vizualizaci - tak byly spojeny do jediného produktu. To znamená, že pro každý dostupný objekt v knihovně automatizace (např. ventilátor, motor nebo ohřívač) najde uživatel příslušný protějšek i v knihovně vizualizace[8].

2.3 Saia® FBox-Builder

Pomocí nástroje Saia® FBox-Builder mohou uživatelé rychle, pohodlně a nezávisle vytvářet vlastní speciální funkce (FBoxy) pro své oblasti aplikací. S těmito funkcemi se pak zachází a jsou v knihovnách dokumentovány stejně, jako ostatní grafické objekty. Začleněním a opakovaným využíváním takovýchto speciálních funkcí v programech, specifických pro danou technologii, dosahuje DDC-Plus neomezených možností rozšiřování a využití[8].

2.4 Hlavní programovací nástroje

Pro co nejefektivnější vytváření programů obsahuje PG5 několik editorů různých programovacích jazyků a také nástroje pro konfiguraci síťových komunikací. Podle potřeb konkrétních aplikací může být uživatelský program editován, generován a zaváděn různými způsoby, z nich každý má své specifické výhody[8].

Fupla - Grafické programování které šetří čas díky používání předdefinovaných FBoxů, plnících určité specifické úlohy (např. zpracování analogových hodnot, ovládání teploty v místnosti, komunikace prostřednictvím modemů, regulátory a mnoho dalších)

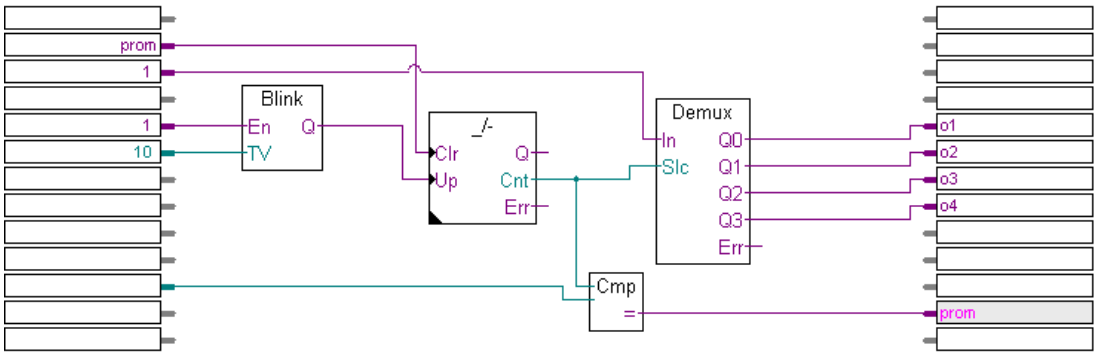
Graftec - Grafický editor pro pohodlné programování sekvenčních, událostmi řízených procesů (např.ovládání manipulátorů, montážních automatů a pod.).

Instruction List - Popisné programovací prostředí, kde programátor vytváří program zápisem jednotlivých instrukcí z velmi bohatého a výkonného souboru instrukcí pro PCD (tzv. makroasembler).

HMI Editor - Nástroj pro pohodlné programování levných („neinteligentních“) terminálů pro styk s obsluhou, jako jsou např. terminály řady PCD7.D23x.

2.5 FUPLA

FUPLA je programovací jazyk vyvinutý firmou *Saia-Burgess Controls* pro programování rodiny automatů Saia®PCD. Tento jazyk je specifický tím, že nemusíte napsat ani řádku kódu, protože je založen na grafické propojování logických schémat, pomocí kterých se vytváří celá logika automatu(obr. 2).



Obr. 2. Ukázka programu v prostředí PG5

Mezi logickými obvody se vyskytují běžně známé členy jako je AND, OR, XOR, ale také moduly, které reprezentují různé matematické, ale i složitější předdefinované funkce jako je např. převod hodnoty Float na Integer, dělení dvou hodnot, čítač atd.. Dále se zde vyskytují moduly, které reprezentují fyzické součásti automatu jako je např. A/D a D/A převodníky, pomocí niž je možné zjišťovat nebo posílat data mimo automat. Programování je poměrně jednoduché a intuitivní a zvládne ho i méně zdatný programátor.

II. PRAKTICKÁ ČÁST

3 EMULÁTOR PG5

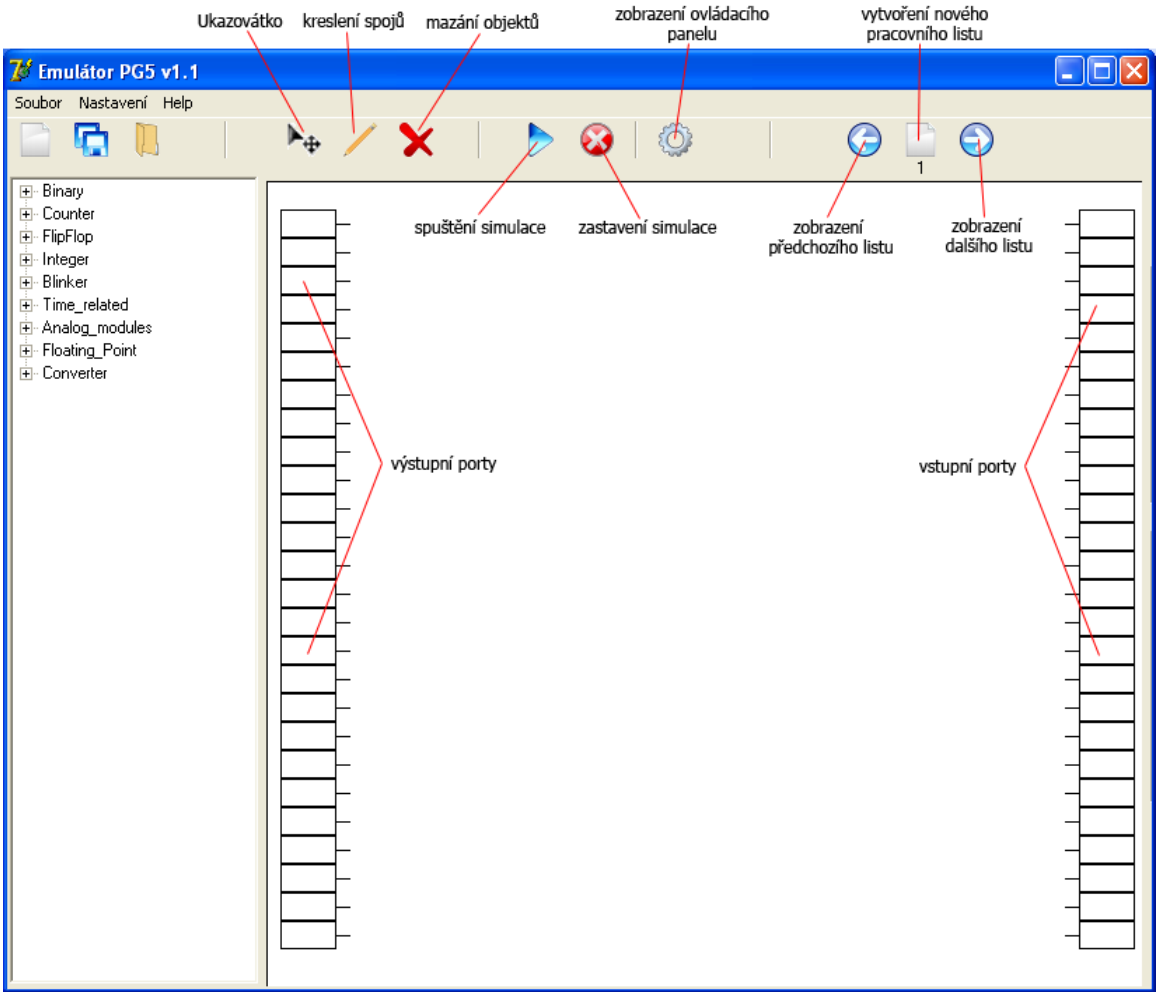
Vytvořil jsem emulátor, který věrně kopíruje programovací rozhraní SAIA PG5 pro jazyk FUPLA a v kterém je možné okamžitě simulovat chod programu bez nutnosti fyzického propojené s reálným automatem.

3.1 Uživatelské rozhraní

Emulátor PG5 se skládá z dvou hlavních částí: programovací rozhraní a ovládací panel.

3.1.1 Programovací rozhraní

První část je samotné programovací rozhraní, kde je možné kreslit logická schémata stejně jako v prostředí PG5. Pracovní prostředí je rozloženo na část kde je seznam dostupných obvodů, panel nástrojů a pracovní plocha pro kreslení samotných schémat(Obr. 2).



Obr. 3. Programovací rozhraní emulátoru PG5

Seznam dostupných obvodů je uspořádaný do složek označující typ skupiny FBoxů, které jsou v ní obsaženy. Po rozbalení příslušné složky se ukážou názvy definovaných FBoxů a ty je možné vybrat a následně vložit na pracovní plochu.

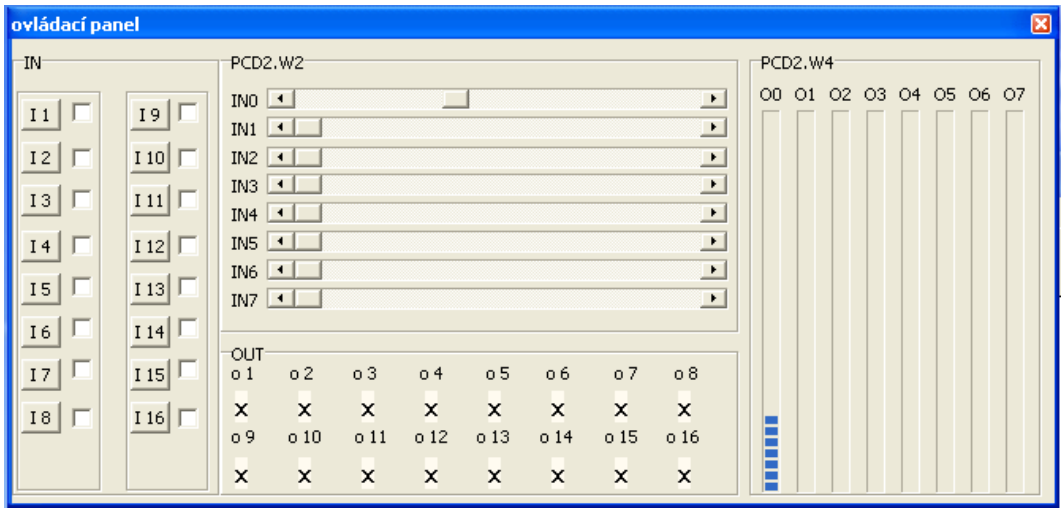
V panelu nástrojů nalezneme několik ikon:

- Vytvoření nového projektu – Vytvoří prázdnou pracovní plochu.
- Uložení projektu – Uložení vytvořeného projektu.
- Otevření projektu – Otevření uložených projektů.
- Ukazovátko – Slouží pro přemisťování FBoxů na pracovní ploše.
- Kreslení spojů – Používá se na kreslení spojů pro spojování vložených FBoxů vodícími čárami.
- Mazání objektů – Používá se pro mazání FBoxů.
- Spuštění simulace – Spuštění simulaci vytvořeného programu.
- Zastavení simulace – Zastavení simulace
- Zobrazení ovládacího panelu – Po stisknutí se zobrazí ovládací panel.
- Zobrazení předchozího listu – Pokud existuje, zobrazí se předchozí list pracovní plochy.
- Vytvoření nového listu – Vytvoří se nový prázdný list pracovní plochy.
- Zobrazení dalšího listu – Pokud existuje, zobrazí se následující list pracovní plochy.

Na pracovní ploše nalezneme nalevo několik výstupních portů a napravo stejný počet vstupních portů. Ty slouží hned k několika účelům. Můžeme do nich vepisovat své pomocné proměnné, konstanty nebo je využít pro propojení s ovládacím panelem pomocí předdefinovaných proměnných (více v části 3.3 Simulace běhu programu).

3.1.2 Ovládací panel

Tento panel je modul, který nahrazuje hardware Saia®PCD. Obsahuje bitové a analogové vstupy a také bitové a analogové výstupy (Obrázek 4).



Obr. 4. Ovládací panel

Jednobitové vstupy IN reprezentují porty vedoucí do programovatelného automatu pomocí kterých se může zasahovat do programu v reálném čase. Část z názvem PCD2.W2 je náhrada A/D převodníku, pomocí kterého můžeme v reálném automatu sledovat různá napětí vedoucí z analogových zařízení. Tato napětí jsou pak převedena na 10 bitové číslo, které můžeme pomocí FBoxu PCD2.W2 sledovat.

Při spuštění programu se tažením slideru mění hodnoty vedoucí do PCD2.W2 a tím se simulují hodnoty napětí, které by vstupovali do reálného automatu.

Část ovládacího panelu pod označením PCD2.W4 je náhrada D/A převodníku, který se u reálného automatu používá pro ovládání zařízení reagující na analogové hodnoty. V ovládacím panelu můžeme tyto hodnoty sledovat pomocí zásobníku, který reprezentuje velikost analogové hodnoty. Stejně jako u PCD2.W2 je tento převodník reprezentován příslušným FBoxem. Jako poslední na panelu zbývá část OUT. Jedná se o jednobitové výstupy, které v automatu fungují jako spínače pro jiná externí zařízení. V ovládacím panelu jsou reprezentovány textovým polem, které zobrazuje hodnotu 1 nebo 0.

3.2 Nejpoužívanější Fboxy

Emulátor obsahuje 50 nejpoužívanějších Fboxů pro vývoj aplikací pro programovatelné automaty Saia PG5. Jedna se o FBoxy, které jsou nezbytné pro vývoj aplikací, založené na čtení binárních vstupů a ovládání binárních výstupů.. Dále se zde vyskytuje několik Fboxů pro práci s A/D a D/A převodníky, které můžou být k programovatelnému automatu připojeny.

3.2.1 Binary

Jedná se o skupinu FBoxů pro práci s binárními daty. Výpis a funkce jednotlivých FBoxů jsou uvedeny níže.

AND - Logický člen AND.

OR - Logický člen OR.

XOR – Logický člen XOR

ODD - Tento logický člen dává na výstupu logickou 1 pokud na vstupech má lichý počet logických 1.

Demux integer selection – Jedná se o demultiplexor, který přepíná hodnotu vstupu In na výstupní porty Q0 až Q7 v závislosti na hodnotě vstupního portu slc.

Mux integer selection – Jedná se o multiplexor, kterým pomocí vstupního portu slc můžeme vybírat, který port vstupu i0 až i7 bude přepnut na výstupní port out.

3.2.2 Counter

Tato skupina FBoxů obsahuje členy, které umožňují reagovat na kladné hrany na vstupních portech a tím čítat impulsy. Hodnoty čítání je pak možné sledovat na výstupních portech.

Up with clear – Jedná se o klasický čítač se dvěma vstupy clr a up. Up reaguje na kladné hrany impulsu a tak zvyšuje svoji vnitřní proměnnou a vysílá ji na výstup cnt. Vstup clr pak slouží pro vynulování této proměnné.

Down with press – Tento čítač oproti předchozímu snižuje svoji vnitřní proměnnou od hodnoty nastavené na vstupu IC. Vnitřní proměnnou pro čítání je pak možné nastavit znovu pomocí vstupu SET.

Up down with press and clear – Tento čítač je kombinací dvou předchozích čítačů. Obsahuje jak vstupy Up a Down, tak i SET a Clear. Je tedy možné pomocí něho čítat v obou směrech a nastavit nebo vynulovat jeho vnitřní proměnnou.

3.2.3 Flip-Flop

Skupina Fboxů založených na klopných obvodech.

Type RS – Klasický klopný obvod RS. Pokud pošleme na vstupní port S logickou 1 obvod se překlopí a bude mít na výstupu logickou 1. Pokud pošleme logickou 1 na vstup R obvod se překlopí zpět a na výstupu tak bude stále logická 0.

3.2.4 Integer

Jedná se o skupinu FBoxů pro práci s daty typu integer.

Is equal to – Porovnává dvě vstupní proměnné datového typu integer. Pokud se rovnají, je na výstupu logická 1. V opačné případě 0.

Is greater or equal to – Porovnává dvě vstupní proměnné datového typu integer. Pokud je první větší nebo rovna druhé je na výstupu logická 1. V opačné případě 0.

Is greater then - Porovnává dvě vstupní proměnné datového typu integer. Pokud je první větší než druhá, pak je na výstupu logická 1. V opačné případě 0.

Is smaller or equal to - Porovnává dvě vstupní proměnné datového typu integer. Pokud je první menší nebo rovna druhé je na výstupu logická 1. V opačné případě 0.

Is smaller then - Porovnává dvě vstupní proměnné datového typu integer. Pokud je první menší než druhá, pak je na výstupu logická 1. V opačné případě 0.

Is zero - Fbox obsahuje pouze jeden vstup. Pokud je roven 0, pak je na výstupu logická 1.

Multiply - Násobí dvě vstupní proměnné a tuto hodnotu dává na výstup.

Divide - Dělí dvě vstupní proměnné se zbytkem. Na výstupu A/B je hodnota dělení se zbytkem. Na výstupu A%B je zbytek po dělení.

Add - Obsahuje dva vstupní porty. Na výstupu je součet těchto portů.

Subtract - Obsahuje dva vstupní porty. Na výstupu je rozdíl prvního od druhého.

Maximum - Na výstupu je nejvyšší hodnota na vstupních portech.

Minimum - Na výstupu je nejnižší hodnota na vstupních portech.

3.2.5 Floating point

Jedná se o skupinu FBoxů pro práci s daty typu Float. Ve skupině jsou obsaženy totožné FBoxy jako v předcházející skupině Integer, jenom s tím rozdílem že se pracuje s datovým

typem podporující plovoucí desetinou čárku. Níže uvedený FBox je jediný, který má odlišnou funkci oproti ekvivalentnímu Fboxu ve skupině Integer.

Divide – Obsahuje dva vstupní porty. Na výstupu je pak hodnota po dělení hodnoty na prvním portu s druhým bezezbytku.

3.2.6 Blinker

V této skupině jsou obsaženy jedny s velice důležitých FBoxů. Pomocí nich můžeme vytvořit signál vysílající v nastavených časových intervalech binární hodnoty. V obvodech tak můžeme mít kontrolu nad časem, který uběhl v průběhu aplikace.

Blink delay T - Tento FBox střídá v pravidelných intervalech logickou hodnotu 1 a 0 na výstupu. Tento interval je možné nastavit pomocí vstupního portu TV. Délka časového intervalu je rovna hodnotě vstupu TV*100ms. Druhý vstupní port En slouží pro zapnutí generování výstupního signálu.

Sample – FBox vysílá na výstup v pravidelných intervalech krátký impulz. Velikost intervalu je možná opět nastavit pomocí vstupu TV.

Blink delay T0/T1 – Má podobnou funkci jako má Blink delay T, ale s tím rozdílem že je možno nastavit zvlášť délku trvání výstupní hodnoty logická 1 a logická 0 pomocí vstupních portů T0 a T1.

3.2.7 Time related

Skupina obsahuje zpožďovací členy, které dokážou zpozdit vstupní binární signál podle nastavených kritérii.

On delay – Pokud je na vstupu IN logická 1, bude tato hodnota vyslána na výstup Q za čas nastavený na portu TV.

Off delay – Pokud je na vstupu IN, třeba i krátký impulz log.1 bude tato hodnota podržena na výstupu po dobu nastavenou na portu TV.

3.2.8 Converter

Skupina obsahuje FBoxy pro převody mezi datovými typy.

Int to float - Převede datový typ integer na float.

Float to int - převede datový typ float na integer.

3.2.9 Analog modules

Poslední skupina FBoxů je určena pro propojení s některými analogovými zařízení.

PCD2.W2 - Slouží pro propojení a čtení hodnot s A/D převodníku PCD2.W2.

PCD2.W4 - Slouží pro propojení a nastavení hodnot do D/A převodníku PCD2.W4.

3.3 Simulace běhu programu

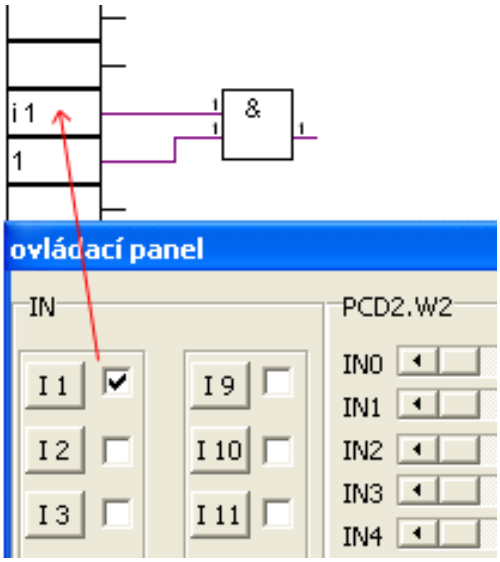
Programování aplikace pomocí emulátoru PG5 je velmi podobné programování ve vývojovém prostředí PG5. Přesunováním a následným spojováním FBoxů na pracovní ploše se vytváří logika programu. Při zapnutí simulace můžeme okamžitě sledovat chod programu na zobrazených hodnotách u vstupních a výstupních portů FBoxů. Tento způsob sledování simulace je zvláště vhodný při ladění programu, protože máme stálou kontrolu nad tím, jaká data se přenáší mezi každým členem.

Pokud se chceme simulací přiblížit co nejvíce reálnému automatu, můžeme k tomu využít přídavný ovládací panel. Ten nám poslouží k zobrazení hodnot, které jsou na něj přivedené pomocí jednobitových výstupů nebo D/A převodníku a také k zásahu do chodu vytvořeného programu změnou některých proměnných pomocí jednobitových vstupů nebo A/D převodníku.

3.3.1 Jednobitové vstupy In

Jednobitové vstupy IN reprezentují porty vedoucí do programovatelného automatu, které můžeme pomocí programu sledovat a reagovat na ně. Pokud stiskneme tlačítka reprezentující příslušný vstup bude na něm logická 1, v opačném případě logická 0. Pokud by nastal případ, při kterém je potřeba mít naráz na více vstupech nastaveny hodnoty, můžeme k tomu využít přidružené checkboxy. Pokud u tlačítka zaškrtneme chcekbox, bude tento vstup stále držet hodnotu logická 1, v opačném případě můžeme hodnotu měnit stiskem příslušného tlačítka.

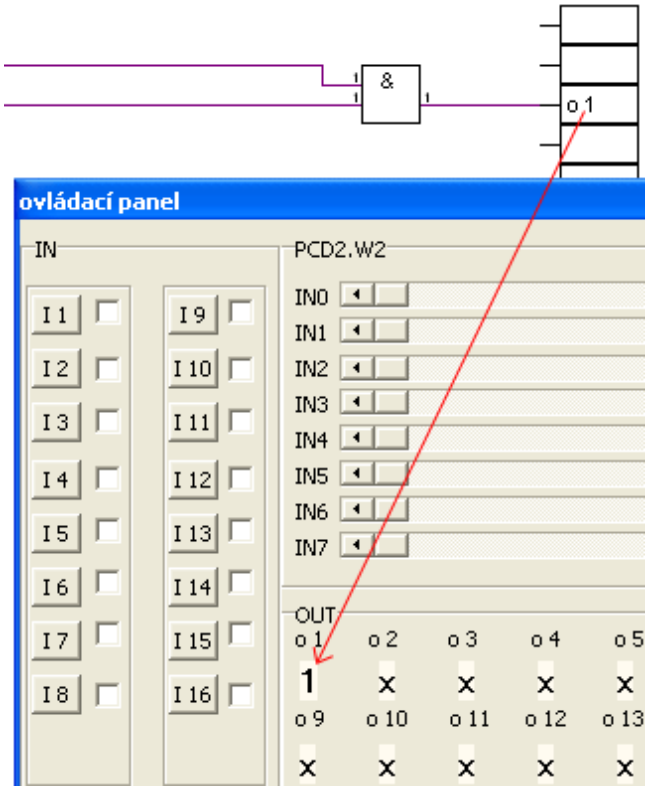
Propojení vstupů IN k vytvořenému programu je velice jednoduché. Slouží k tomu předdefinované proměnné ‘i 1‘ až ‘i 16‘. Tyto proměnné stačí vepsat do jednoho z výstupních portů, který tak bude data z ovládacího panelu převádět na tento port.



Obr. 5. Propojení jednobitových vstupů ovládacího panelu s porty na pracovní ploše

3.3.2 Jednobitové výstupy OUT

Jednobitové výstupy OUT reprezentují porty jdoucí z programovatelného automatu, které slouží pro ovládání externích zařízení reagující na jednobitové hodnoty. K propojení slouží předdefinované proměnné ‘o 1‘ až ‘o 16‘, které stačí vepsat do některého ze vstupních portů na pravé straně pracovní plochy. Data jdoucí do takto definovaného portu se při simulaci projeví na ovládacím panelu.

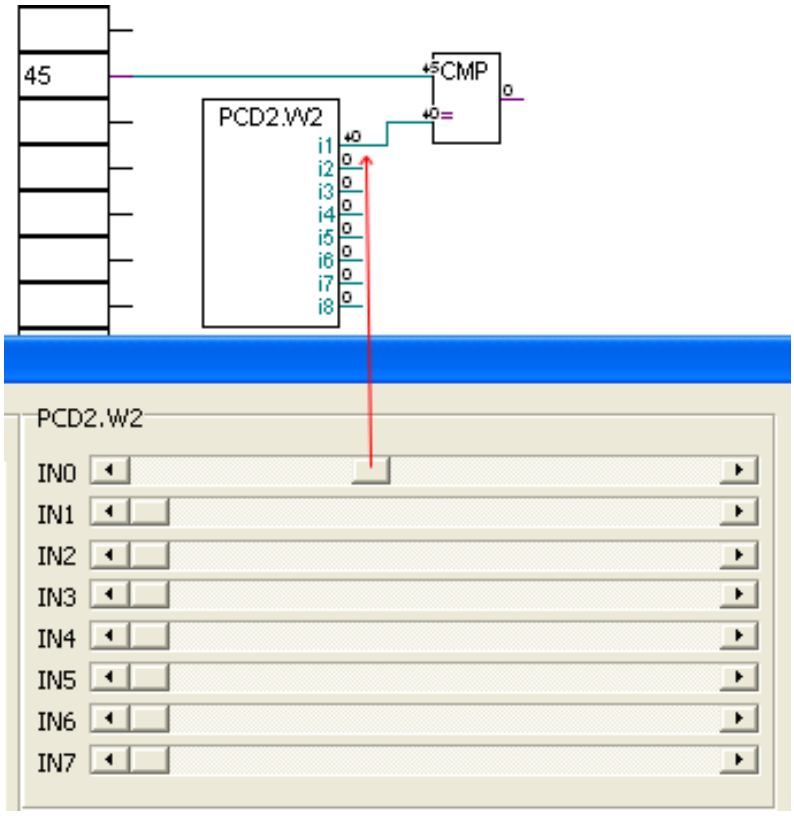


Obr. 6. Propojení jednobitových výstupů

s porty na pracovní ploše

3.3.3 A/D převodník PCD2.W2

Část ovládacího panelu pod označením PCD2.W2 je náhrada A/D převodníku, pomocí kterého můžeme v reálném automatu sledovat různá napětí vedoucí z analogových zařízení. Tato napětí jsou pak převedena na 10 bitové číslo, které můžeme pomocí FBoxu PCD2.W2 sledovat. V této části ovládacího panelu je 8 sliderů IN0 až IN7, které simulují vstupní napětí A/D převodníku. Jejich tažením simulujeme hodnotu vstupního napětí a tato hodnota je převedena na 10-ti bitové číslo, které je vyvedeno na výstupní porty FBoxu PCD2.W2.

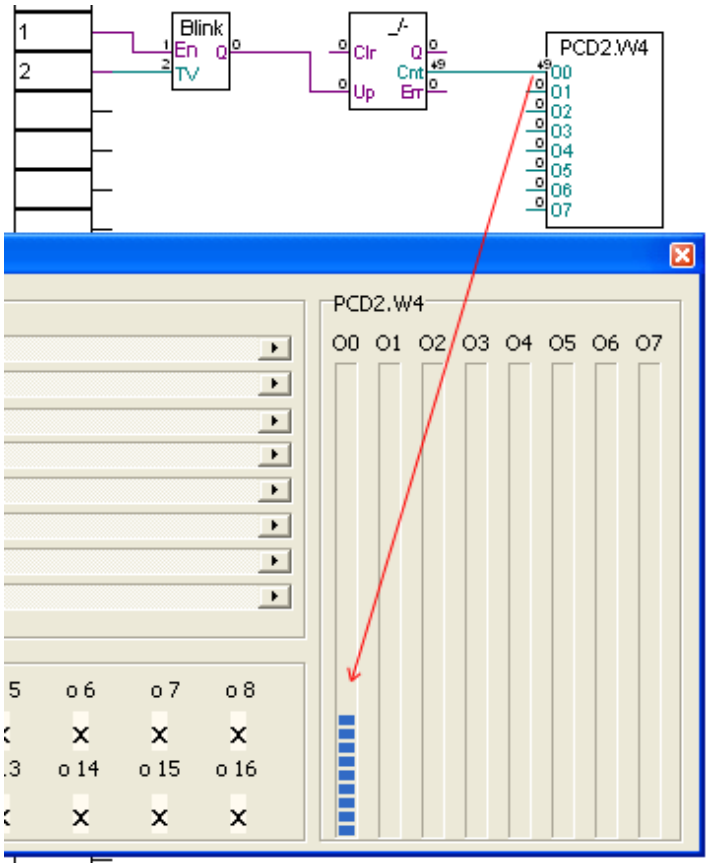


Obr. 7. Simulace A/D převodníku PCD2.W2

3.3.4 D/A převodník PCD2.W4

Část ovládacího panelu pod označením PCD2.W4 je náhrada D/A převodníku, který se u reálného automatu používá pro ovládání externích zařízení reagující na analogové hodnoty. Převodník přijímá hodnoty v rozsahu 8-bitů a ty převádí na výstupní napětí.

V emulátoru je D/A převodník reprezentovaný FBoxem PCD2.W4 obsahující 8 vstupů o0 až o7. Tyto vstupy jsou následně převedeny na simulovanou analogovou hodnotu, kterou můžeme sledovat na jednotlivých zásobnících o0 až o1.



Obr. 8. Simulace D/A převodníku PCD2.W4

4 ALGORITMY EMULÁTORU

4.1 Jádro

Jádro emulátoru se stará o převody hodnot jednotlivých FBoxů mezi sebou vždy od výstupu na vstup. Tyto operace musí provádět v reálném čase. Proto je jádro rozděleno na dvě hlavní části. V první části těsně po spuštění simulace proběhne kompilace celého programu a v druhé části se ohodnocují vstupy a výstupy všech FBoxů v reálném čase na základě zpracovaných dat kompilace.

4.1.1 Kompilace

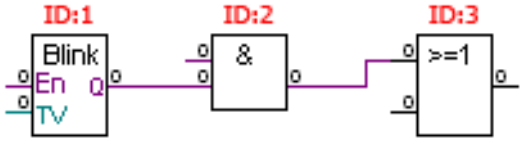
Kompilace se skládá ze tří částí a to, vytvoření pole relací mezi objekty, nastavení pořadového čísla objektu a uspořádání pole relací.

4.1.1.1 Vytvoření pole relací

V první části se nejprve vytvoří pole relací mezi objekty a toto pole je poté vstupem pro druhou část kompilace. Algoritmus tedy hledá, které dva FBoxy jsou spolu spojeny vstupem a výstupem. Pokud jsou takové dva objekty nalezeny, zapíšou se jejich údaje do podobného pole jaké znázorňuje tabulka 1. Do pole pak stačí zapsat pouze ID FBoxu a číslo jeho portu a s těmito údaji dále pracuje druhá část kompilace. Pro lepší pochopení algoritmu uvádím obrázek(obr.9), který znázorňuje propojené FBoxy z kterých je vytvořena tabulka(tab.1) .

Tab. 1. Ukázka seznamu relací mezi FBoxy

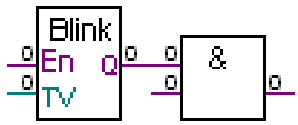
ID výstupního FBoxu	Číslo výstupního portu	ID vstupního FBoxu	Číslo vstupního portu
1	1	2	2
2	1	3	1



Obr. 9. Relace mezi objekty

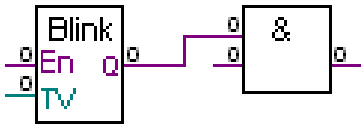
Při zjišťování relací mezi FBoxy existují 3 případy jak můžou být FBoxy mezi sebou propojené. Algoritmus tedy hledá postupně různé způsoby propojení a zapisuje je do pole relací. Níže jsou uvedeny 3 názorné ukázky způsobu propojení.

První hledá objekty, které jsou spolu spojeny přímo pomocí vlastních portů.



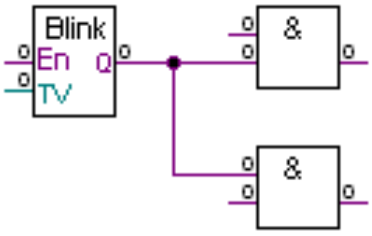
Obr. 10. FBoxy s přímím propojením

Druhý hledá objekty, které jsou spolu spojeny pomocí vodících čar.



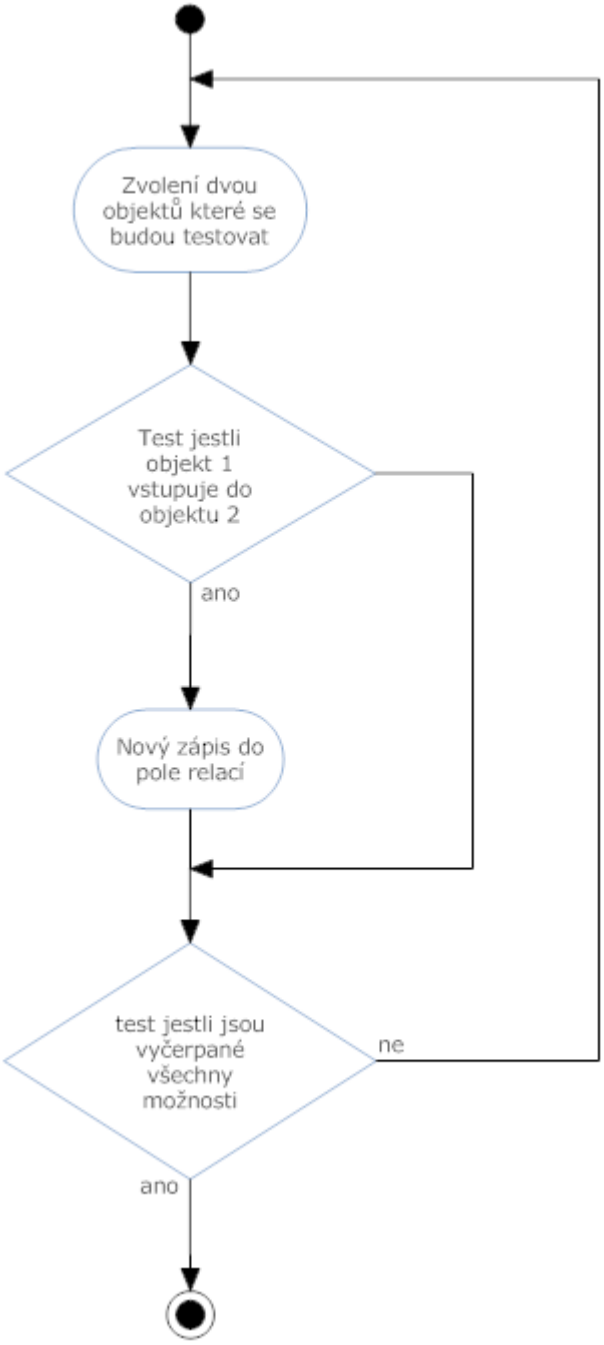
Obr. 11. FBoxy s propojením pomocí vodících čar

Třetí hledá objekty na které je napojena vodící čára vyvedená z uzlu jiné vodící čáry.



Obr. 12. Propojení FBoxů vodící čarou vevedenou z uzlu

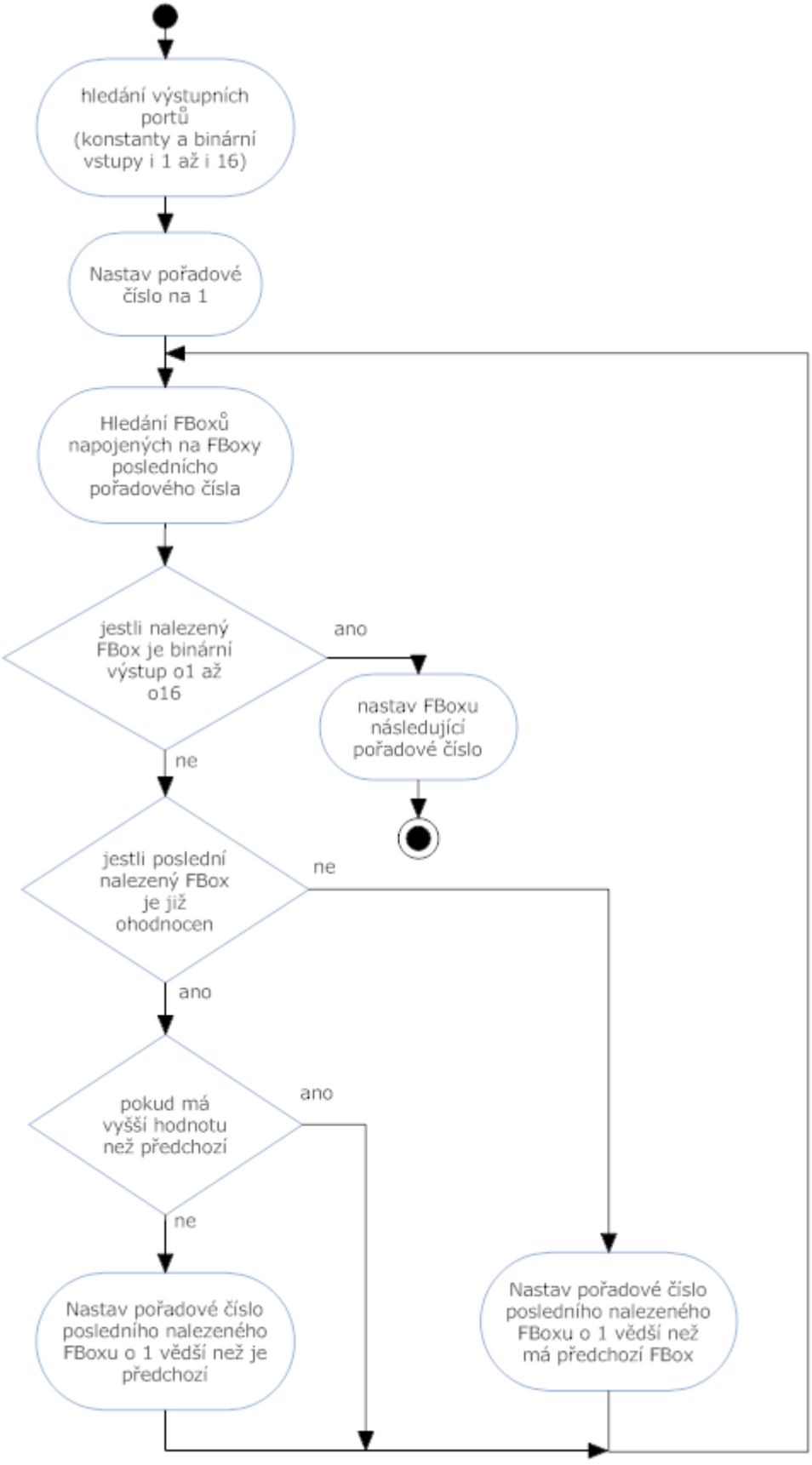
Zjednodušená podoba algoritmu je znázorněna v diagramu aktivit na obrázku(obr. 13).



Obr. 13. Zjednodušený diagram vytváření pole relací

4.1.1.2 Nastavení pořadového čísla objektu

Aby mohly být FBoxy vyhodnocovány ve správné pořadí, je třeba jim přiřadit pořadové číslo, které jednoznačně určí, která skupina FBoxů se vyhodnotí první a která další. Princip přiřazování pořadového čísla spočívá v tom, že jsou nejprve vyhledány výstupní porty s nastavenou konstantou a nebo porty, které mají nastavené propojení s jednobitovým vstupem i 1 až i 16. Těmto objektům je nastaveno pořadové číslo 1. V dalším kroku se pak hledají FBoxy, které jsou napojené na objekty s posledním pořadovým číslem. Těmto nalezeným objektům se nastaví nové pořadové číslo podle určitých kritérií. Pokud objekt vystupuje na FBox, který má již nastavené pořadové číslo, je ohodnocen pouze tehdy pokud by měl dostat pořadové číslo větší než má nyní. V opačném případě se jedná o FBox, který posílá své data dál do programu a je mu nastaveno pořadové číslo o jedničku větší než má předcházející objekt. Tento cyklus běží stále dokola, dokud se nenarazí na některý ze vstupních portů nebo na jiný vstupní FBox jako je například D/A převodník PCD2.W4. Celý algoritmus je znázorněný na obrázku(obr. 14).



Obr. 14. Diagram nastavení pořadového čísla

4.1.1.3 Uspořádání pole relací

V tomto posledním procesu kompilace se přeuspořádá pole relací na základě přiřazeného pořadového čísla. FBoxy se potom budou přepočítávat ve správném pořadí a algoritmus bude fungovat na podobném principu jako je běh programu reálného automatu.

4.1.2 Ohodnocení vstupů a výstupů

Algoritmus ohodnocení vstupů a výstupů slouží k nastavení hodnot na vstupech všech FBoxů a k následnému výpočtu výstupů a přiřazení těchto hodnot do navazujících vstupů dalších FBoxů. Algoritmus tedy projíždí postupně pole relací a spouští vnitřní algoritmy FBoxů. Vypočtené výstupní hodnoty poté přiřadí navazujícímu vstupu dalšího FBoxu. Tento algoritmus probíhá v reálném čase v pravidelných intervalech a tím simuluje otočky cyklu, které jsou využity u reálných programovatelných automatů. V emulátoru se otočka cyklu vykoná jednou za 100ms, což není stejná rychlost opakování jako v reálném automatu. Pro simulaci, testování a pochopení logiky jazyk FUPLA je tato rychlost naprosto dostačující a tím je možné i sledovat operace, které by z důvodu velké rychlosti opakování nebylo možné vidět.

5 PROGRAMOVÁNÍ FBOXŮ

Software je navržený tak, že se dá jednoduše pomocí knihoven rozšířit o další FBoxy, které by se nám mohly v programu později hodit. Programování takového dalšího členu je poměrně jednoduché a programuje se pomocí jazyka Object Pascal ve vývojovém prostředí DELPHI. Jak vytvořit vlastní přídavnou knihovnu si ukážeme na jednoduchém příkladu FBoxu, který funguje jako logický člen AND.

5.1 Popis stuktury

Knihovna každého členu se skládá ze 3 hlavních funkcí a dvou pomocných.

5.1.1 Funkce Ainit

První funkce Ainit slouží inicializace základních vlastností FBoxu a také jeho podoby. Proměnné width_clenu a height_clenu slouží k nastavení velikosti objektu, což je důležité pro správnou identifikaci objektu v prostoru a při určování jeho hranic pro správné vykreslování vodících čar. Tyto proměnné jsou poté přiřazeny objektu podoba. Tento objekt je typu TBitmap a nastavením jeho rozměrů nastavíme hranici vykreslovací plochy pro tento FBox. Pokud bychom tuto hranici nastavily větší nebo menší než je skutečná velikost FBoxu, objekt by se buď vekresloval jen z části a nebo by kolem sebe překresloval poblíž nacházející se objekty. Proměnné rozměru je tedy nutné nastavit přesně, jinak by nám to mohlo způsobovat problémy v hned několika algoritmech emulátoru. Rozměry se vždy musí nastavovat v násobcích deseti, protože jádro i grafické zobrazování emulátoru se pohybuje po sítí, která je rozložena po 10-ti pixelech.

Jako další proměnné jsou pocet_in a počet_out. Tyto proměnné inicializují kolik má FBox vstupů(pocet_in) a kolik výstupů(pocet_out). Bez správného nastavení by algoritmus nepoznal kolik vstupů či výstupů má hledat na FBoxu a to by zapříčinilo chyby v uchytávání vodících čar nebo by neuvedené vstupy/výstupy nebyly brané v úvahu při výpočtech u simulace.

Pole typ_in[] a typ_out[] slouží k nastavení datového typu vstupních nebo výstupních dat určitého portu. Porty jsou očíslované shora a počítá se od jedné. Tedy první vstupní port členu AND je typ_in[1], druhý typ_in[2]. Datové typy se určují přiřazením čísla 1 až 3. Výpis těchto typů jsou uvedené v tabulce 2.

Tab. 2. Datové typy FBoxu

ID datového typu	Datový typ
1	Byte
2	Integer
3	Float

Nesprávným nastavením těchto typů by se stalo, že by algoritmus pro kreslení vodících čar povolil spojení FBoxů různých datových typů a to by mohlo způsobit chyby ve výpočtech.

Poslední sada volaných funkcí Rectangle, pin a text slouží pro definování samotné podoby FBoxu a k tomu slouží již zmiňované pomocné funkce. Rectangle vykreslí do objektu podoba obdélník o nastavených rozměrech. Pin vykreslí vodorovný vstup/výstup na nastavené pozici. Poslední argument této funkce slouží k nastavení barvy pinu. Text vykreslí na pozici námi zvolený popisek. Nastavení vhodné barvy u pinu je důležité pouze pro správné zobrazení pinů u FBoxu, protože PG5 pro zobrazování vodících čar a pinů používá různé barvy pro datové typy, aby se zvýšila přehlednost programu. V následující tabulce jsou uvedeny správné barvy pro již zmiňované tři datové typy v normě RGB.

Tab. 3. Barvy datových typů

Datový typ	Barva
Byte	RGB(127,0,127)
Integer	RGB(0,127,127)
Float	RGB(127,127,0)

```
 1 procedure Ainit(podoba:TBitmap;var width_clenu, height_clenu,
   pocet_in, pocet_out:Byte; var typ_in, typ_out:Array of Byte);
 2 begin
 3 width_clenu:=50;
 4 height_clenu:=30;
 5 podoba.Width:=width_clenu;
 6 podoba.Height:=height_clenu;
 7
 8 pocet_in:=2;
 9 pocet_out:=1;
10
11 typ_in[1]:=1;
12 typ_in[2]:=1;
13 typ_out[1]:=1;
14
```

```
15 podoba.Canvas.Rectangle(10, 0,40, 30);
16 pin(0,10,podoba,RGB(127,0,127));
17 pin(0,20,podoba,RGB(127,0,127));
18 pin(40,20,podoba,RGB(127,0,127));
19 text(20,1,8,'&',podoba,0);
20 end;
```

5.1.2 Funkce Aprepocet

Tato funkce slouží pro výpočet výstupní hodnoty FBoxu na základě vstupních hodnot. Funkce má sedm argumentů, ale pro výpočet členu AND budou stačit pouze dva. Argumentů je použito více i když se nepoužívají z toho důvody, aby program mohl být lehce rozšířitelný a podoba knihovny byla v jádru stejná pro všechny možné druhy FBoxů. V tomto příkladu použijeme pouze pole hodnota_in[] a hodnota_out[], ale pokud chceme programovat například čítač, museli bychom brát v úvahu i proměnné cas a minula_hodnota_cas. Další proměnné jako je pole pomoc_prom[] slouží k libovolným účelů pro některé druhy složitějších FBoxů. V tomto příkladu je můžeme ignorovat, bez jakýchkoli následků.

Aplikace algoritmu pro AND je tedy velice jednoduchá, jak je vidět na příkladu níže.

```
1 procedure Aprepocet(hodnota_in:Array of Real;var
  hodnota_out,minula_hodnota_in:Array of Real;var
  cas,minula_hodnota_cas,pocitadlo:Word;var pomoc_prom:Array of Byte);
2 begin
3 if (hodnota_in[1] = 1) AND (hodnota_in[2] = 1) then
4 hodnota_out[1]:=1
5 else hodnota_out[1]:=0;
6 end;
```

Jedná se o podmínku, která porovnává vstup 1 a 2. Pokud jsou tyto vstupy rovny 1, tak výstupní hodnota je rovna 1 v opačném případě je výstup 0. Vstupy a výstupy jsou stejně číslované jako v případě nastavování typu proměnných.

5.1.3 Funkce AsetXY

Tato funkce slouží k nastavení pozic konců, lépe řečeno záchytných bodů vstupních a výstupních pinů FBoxu. Tato funkce probíhá pouze tehdy pokud objekt vložíme na

pracovní plochu nebo pokud s ním pohybujeme. Funkce nastaví do polí x_in[], y_in[], x_out[], y_out[] pozice všech pinů Fboxu a toho se využívá při kreslení vodících čar, aby bylo jasné na které místo se vodič zachytí. Také se v této funkci nastavují proměnné x_clenu a y_clenu, což jsou pozice levého horního rohu FBoxu. Ty slouží pro identifikaci objektu při jeho posunování nebo při vykreslování vodících čar. Následující ukázka zobrazuje celou definici funkce AsetXY pro FBox AND.

```
 1 procedure AsetXY(X,Y:Word;var x_clenu,y_clenu:Word;var
   x_in,y_in,x_out,y_out:Array of Word; pocet_in,pocet_out:Byte);
 2 begin
 3 x_clenu:=X;
 4 y_clenu:=Y;
 5
 6 x_in[1]:=X;
 7 y_in[1]:=Y+10;
 8 x_in[2]:=X;
 9 y_in[2]:=Y+20;
10
11 x_out[1]:=X+50;
12 y_out[1]:=Y+20;
13 end;
```

5.2 Připojení knihovny do Emulátoru

Připojení knihovny se provádí pomocí seznamu členů uložené ve formátu INI. Tento soubor se nachází v kořenovém adresáři emulátoru ve složce knihovny pod názvem seznam_clenu.ini. V tomto souboru najdeme úplný seznam FBoxů, které jsou použité v současné verzi emulátoru. Níže je ukázána část struktury soubory.

```
 1 [info]
 2 pocet_clenu=27
 3 ……
 4 [3]
 5 slozka=Counter
 6 nazev=up with clear
 7 index=3
 8 typ=Counter_UpWithClear
 9 dll=Counter_UpWithClear.dll
10 [4]
11 slozka=Binary
12 nazev=and
```

```
13 index=4
14 typ=and
15 dll=LAnd.dll
16 ……
```

Jako první musí být správně definovaný počet logických členů v seznamu. Pokud připíšeme nový FBox, tak stačí stávající číslo zvýšit o jedničku. Každý nový člen má svoje jednoznačné ID(index) a to musí být shodné s ID nového záznamu. V každém záznamu je 5 proměnných. První proměnná slozka značí do jaké skupiny FBoxů nově vytvořený FBox patří. O vytváření nových složek se zmíním později. Druhá proměnná název určuje pod jakým názvem se bude FBox zobrazovat v seznamu. Třetí je již zmiňovaný jednoznačný identifikátor index. Čtvrtá proměnná typ určuje jednoznačný identifikátor názvu FBoxu. A poslední proměnná dll určí z jaké knihovny se mají data načítat. Příslušnou knihovnu je nutné nahrát do stejné složky jako je seznam FBoxů.

ZÁVĚR

Při vývoji aplikací pro programovatelné automaty je vhodný simulátor nepostradatelnou součásti. Vývoj programů se tak značně zrychluje a zjednodušuje, přičemž není potřeba mít zapojený skutečný programovatelný automat. Emulátor není kompatibilní s originálním PG5 a vytvořené programy je možné použít pouze pro simulaci. Cílem práce bylo, aby studenti pochopili filozofii jazyka FUPLA a naučili se tak vytvářet jednoduché aplikace. Takto získané zkušenosti pak můžou využít při programování skutečného automatu.

ZÁVĚR V ANGLIČTINĚ

An appropriate stimulator is the indispensable part for the development of applications for programmable automats. Therefore, development of programmers is much faster and easier, whereas there is no need of having a real programmable automat plugged in. Emulator don't compatible with original PG5 and this created programs are uses only for simulation. The aim of the thesis was to make sure that students understand the philosophy of FUPLA language and have learned how to create simple applications. The experience gained this way could be used when programming a real automat.

SEZNAM POUŽITÉ LITERATURY

[1] Písek S. : Delphi - začínáme programovat, Nakladatelství Grada Publishing a.s., Praha, 2005.
[2] Procházka M., Strakoš M. : Borland Delphi – průvodce vývojářem Kniha I, Vydavatelství Computer Press a. s. , Praha, 2005
[3] Šmejkal, L., Martinásková, M.: PLC a automatizace, Nakladatelství BEN - technická literatura, Praha, 1999.
[4] Martinásková, M., Šmejkal, L.: Řízení programovatelnými automaty, Vydavatelství ČVUT, Praha, 1998.
[5] Martinásková, M., Šmejkal, L.: Řízení programovatelnými automaty II,Vydavatelství ČVUT, Praha, 2000.
[6] Kovář J. :Programovatelné automaty II [online]. 2006 [2006-06-25]. Dostupný z WWW:<http://www.spszl.cz/modules/wfdownloads/singlefile.php?cid=5&lid=20>
[7] Saia PCD - Programovatelné automaty [online]. Dostupný z WWW: <http://www.sbsys.cz/produkty/saia-pcd.php>
[8] Programovací nástroje Saia [online]. Dostupný z WWW: <http://www.sbsys.cz/produkty/saia-pcd-programovaci-nastroje.php>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

FBox Obecný název pro logický člen programovacího jazyka FUPLA

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH