ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

FAKULTA
STROJNÍ

ZÁVĚREČNÁ PRÁCE

2019

VLADYSLAV
GRECHKA

České vysoké učení technické v Praze

Fakulta strojní

Ústav přístrojové a řídící techniky

Obor: Informační a automatizační technika

Nástroj pro výuku programování PLC podle normy IEC EN 61131-3

Tool for PLC programming according to IEC EN 61131-3

BAKALÁŘSKÁ PRÁCE

Vypracoval: Vladyslav Grechka
Vedoucí práce: prof. Ing. Milan Hofreiter, CSc.
Rok: 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Grechka** Jméno: **Vladyslav** Osobní číslo: **440547**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídicí techniky**

Studijní program: **Strojírenství**

Studijní obor: **Informační a automatizační technika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Nástroj pro výuku programování PLC podle normy IEC EN 61131-3

Název bakalářské práce anglicky:

Tool for PLC programming according to IEC EN 61131-3

Pokyny pro vypracování:

- seznamte se se zásadami programování podle normy IEC 61131-3 a s vývojovým systémem Mosaic
- navrhněte soubor příkladů a jejich řešení programy v jazycích podle normy
- příklady opatřete vysvětlujícím komentářem, doplňte úlohy k procvičení a kontrolní otázky
- vytvořte operátorské rozhraní a uživatelskou příručku

Seznam doporučené literatury:

[1] ŠMEJKAL, L., ČERNÝ, J.: Esperanto programátorů PLC: programování podle normy IEC/EN 601131-3 (souhrnné vydání částí stejnojmenného seriálu), Automa, Teco a. s., 2017 (elektronická verze je dostupná na www.tecoacademy.cz).
[2] KOHOUT, L.: Norma pro řídicí systémy IEC 61 131. Edumat, Kutna Hora, 2011 (dostupné na www.edumat.cz).
[3] Programování PLC Tecomat podle normy IEC 61131-3 v prostředí Mosaic. Teco a. s., Kolin, 2007 (dostupné na www.tecomat.com).
[4] Začínáme v prostředí Mosaic. Teco a. s., Kolin, 2010 (dostupné na www.tecomat.com).
[5] ŠMEJKAL, L. , MARTINASKOVA, M.: PLC a automatizace 1. Základní pojmy, úvod do programování. BEN, Technická literatura, Praha, 1999.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

prof. Ing. Milan Hofreiter, CSc., U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.04.2019** Termín odevzdání bakalářské práce: **12.06.2019**

Platnost zadání bakalářské práce: _____

prof. Ing. Milan Hofreiter, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

26-04-2019

Datum převzetí zadání

Podpis studenta

Prohlášení:

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího bakalářské práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

V Praze dne: ……. Podpis: …….

Poděkování

Chtěl bych poděkovat především panu Ing. Ladislavu Šmejkalovi, CSc. a panu prof. Ing. Milanu Hofreiterovi, CSc. za odborné rady a cenné připomínky, kterými přispěli k vypracování této bakalářské práce.

Dále bych rád poděkoval svojí rodině, která mi poskytla podporu během celého studia.

Grechka Vladyslav

Název: Nástroj pro výuku programování PLC podle normy IEC EN 61131-3

Abstrakt:

Bakalářská práce je věnována vytvoření nástroje pro výuku programování PLC podle normy IEC EN 61131-3.

Teoretická část zahrnuje seznámení s normou IEC EN 61131-3, a s vývojovým systémem Mosaic. Větší pozornost je věnována programovacím jazykům pro PLC a některým nástrojům v systému Mosaic.

V praktické části byl vytvořen program jako uživatelská příručka, která obsahuje teorii, příklady a kontrolní otázky.

Klíčova slova: programovaní PLC; norma IEC 61131; programovací jazyk; Mosaic; programovací nástroj.

Title: Tool for PLC programming according to IEC EN 61131-3

Abstract:

The bachelor thesis is devoted to creating a tool for teaching PLC programming according to IEC EN 61131-3.

The theoretical part includes familiarization with the IEC EN 61131-3 standard and the Mosaic development system. More attention is paid to programming languages for PLC and some tools in the Mosaic system.

In the practical part was created a program as a user manual, which contains theory, examples and control questions.

Keywords: PLC programming; standart IEC 61131; programming language; Mosaic; programming tool.

Obsah

Úvod

Téma diplomové práce „Nástroj pro výuku programování PLC podle normy IEC EN 61131-3“ jsem si vybral, protože jsem chtěl vytvořit materiál, který by dal studentovi příležitost setkat se s vývojovým systémem Mosaic, a seznámit se s zásadami programování podle normy IEC 61131-3. V této době probíhá ve světě čtvtrtá průmyslová revoluce, a s ní související automatizace výroby, která sebou přinese změny na trhu práce. V rámci této průmyslové revoluce se předpokládá, že podniky přejdou na nový typ procesního řízení, kdy propojené stroje budou autonomně a koordinovaně měnit výrobní procesy v souladu se společným úkolem tak, aby technologický proces byl rychlý a efektivní.

Motivací pro psaní bakalářské práce je seznámení studentů i již zaměstnaných pracovníků s programováním PLC, které v dnešní době rozvoje techniky a IT prochází revolucí.

Mým cílem je zjednodušení cesty pro studenty mezi teoretickými znalostmi, které dostali na vysoké škole a budoucí prací, která je více zaměrena na vyřešení praktických úloh ve vývojových systémach. Moderních vývojových systémů, které by sloužily k programování PLC podle normy IEC EN 61131-3 je mnoho. Jedním z nich je systém Mosaic. Jedná se o vývojové prostředí pro tvorbu a ladění programů pro programovatelné logické systémy TECOMAT ® z produkce firmy Teco.

1 Norma IEC 61131-3

Norma IEC 61 131 pro programovatelné řídící systémy má pět základních částí a představuje souhrn požadavků na moderní řídící systémy. Je nezávislá na konkrétní organizaci či firmě a má širokou mezinárodní podporu. Jednotlivé části normy jsou věnovány jak technickému tak programovénu vybavení těchto systémů.

V ČR byly přijaty jednotlivé části této normy pod následujícími čísly a názvy:

ČSN EN 61 131-1 Programovatelné řídicí jednotky - Část 1: Všeobecné informace

ČSN EN 61 131-2 Programovatelné řídicí jednotky - Část 2: Požadavky na zařízení a zkoušky

ČSN EN 61 131-3 Programovatelné řídicí jednotky - Část 3: Programovací jazyky

ČSN EN 61 131-4 Programovatelné řídicí jednotky - Část 4: Podpora uživatelů

ČSN EN 61 131-5 Programovatelné řídicí jednotky - Část 5: Komunikace

ČSN EN 61 131-7 Programovatelné řídicí jednotky - Část 7: Programování fuzzy řízení

V Evropské unii jsou tyto normy přijaty pod číslem EN IEC 61 131.

Norma IEC 61 131-3 je třetí částí z rodiny norem IEC 61 131 a definuje programovací jazyky. Norma IEC představuje první vážný pokus o standardizaci programovacích jazyků pro průmyslovou automatizaci. Norma IEC 61131-3 zavádí pět programovacích jazyků PLC, tři grafické a dva textové . Norma byla původně nazývána IEC 1131-3 a byla vydána v roce 1993. V roce 1997 IEC přešla na nový zápis a do názvu normy bylo přidáno číslo „6“. Na její tvorbě pracovalo sedm mezinárodních společností. Nakonec byla norma přijata jako směrnice u většiny významných výrobců PLC. Výsledkem je specifikace syntaxe a sémantiky unifikovaného souboru programovacích jazyků, včetně obecného softwarového modelu a strukturujícího jazyka [3].

Hlavním cílem standardu bylo zvýšit rychlost a kvalitu vývoje softwaru pro PLC a vytvořit programovací jazyky orientované na technology kteří zajistí, že PLC splní ideologii otevřených systémů, čímž eliminuje další fázi školení při změně typu PLC.

Programovací systémy založené na IEC 61131-3 jsou charakterizovány následujícími ukazateli:

- Spolehlivost vytvořeného softwaru - Spolehlivost je zajištěna tím, že programy PLC jsou vytvořeny pomocí speciálně navrženého vývojového prostředí, které obsahuje všechny potřebné nástroje pro psaní, testování a ladění programů pomocí emulátorů a reálných PLC, stejně jako mnoho hotových kódových fragmentů;
- možnost jednoduché úpravy programu a zvýšení jeho funkčnosti;
- přenositelnost projektu z jednoho PLC do druhého;
- schopnost znovu použít fragmenty programu;
- jednoduchost jazyka a omezení počtu jeho prvků.

1.1 Základní pojmy

Základním pojmem při programování podle normy IEC 61 131-3 je termín Programová Organizační Jednotka nebo zkráceně POU (Program Organisation Unit). POU je nejmenší nezávislá část uživatelského programu. POU mohou být dodávány od výrobce řídícího systému nebo je může napsat uživatel. Každá POU může volat další POU a při tomto volání může volitelně předávat volané POU jeden nebo více parametrů. Existují tři základní typy POU:

- Funkce (function, FUN);
- funkční blok (function block, FB);
- program (program, PROG).

Nejjednodušší POU je funkce, jejíž hlavní charakteristikou je to, že pokud je volána se stejnými vstupními parametry, musí produkovat stejný výsledek (funkční hodnotu). Funkce může vracet pouze jeden výsledek. Dalším typem POU je funkční blok, který si na rozdíl od funkce může pamatovat některé hodnoty z předchozího volání (např. stavové informace). Ty pak mohou ovlivňovat výsledek. Hlavním rozdílem mezi funkcí a funkčním blokem je tedy schopnost funkčního bloku vlastnit paměť pro zapamatování hodnot některých proměnných. Funkce tuto schopnost nemají a jejich výsledek je tedy jednoznačně určen vstupními parametry při volání funkce. Funkční blok může také, na rozdíl od funkce, vracet více než jeden výsledek. Posledním typem POU je program, který představuje vrcholovou programovou jednotku v uživatelském programu. Centrální jednotka PLC může zpracovávat více programů a programovací jazyk ST obsahuje prostředky pro definice spouštění programů [3].

Každá POU se skládá ze dvou základních částí - deklarační a výkonné. V deklarační části POU se definují proměnné potřebné pro její činnost. Výkonná část pak obsahuje vlastní příkazy pro realizaci požadovaného algoritmu [3].

Deklarační část POU

Deklarační část POU obsahuje definice proměnných potřebných pro činnost POU. Proměnné jsou používány pro ukládání a zpracování informací. Každá proměnná je definována jménem proměnné a datovým typem. Datový typ určuje velikost proměnné v paměti a zároveň do značné míry určuje způsob zpracování proměnné. Pro definice proměnných jsou k dispozici standardní datové typy (BOOL, BYTE, INT, apod.). Použití těchto typů závisí na tom, jaká informace bude v proměnné uložena (např. typ BOOL pro informace typu ANO-NE, typ INT pro uložení celých čísel se znaménkem). Uživatel má samozřejmě možnost definovat svoje vlastní datové typy. Umístění proměnných v paměti PLC systému zajišťuje automaticky programovací prostředí. Pokud je to potřeba, může umístění proměnné v paměti definovat i uživatel.

Proměnné můžeme rozdělit podle použití na globální a lokální. Globální proměnné jsou definovány vně POU a mohou být použity v libovolné POU, jsou viditelné z libovolné POU. Lokální proměnné jsou definovány uvnitř POU a v rámci

této POU mohou být používány (z ostatních POU nejsou viditelné). A konečně proměnné jsou také používány pro předávání parametrů při volání POU. V těchto případech mluvíme o vstupních resp. výstupních proměnných [3].

Výkonná část POU

Výkonná část POU následuje za částí deklarační a obsahuje příkazy a instrukce, které jsou zpracovány centrální jednotkou PLC. Ve výjimečných případech nemusí definice POU obsahovat žádnou deklarační část, a potom je výkonná část uvedena bezprostředně za definicí začátku POU. Příkladem může být POU, která pracuje pouze s globálními proměnnými. Toto řešení není ideální, ale může se použít. Výkonná část POU může obsahovat volání dalších POU. Při volání mohou být předávány parametry pro volané funkce resp. funkční bloky [3].

1.2 Společné prvky

Každý program pro PLC se skládá ze základních jednoduchých prvků, určitých nejmenších jednotek. Z těchtno jednotek se vytvářejí deklarace a příkazy. Jednoduché prvky dělíme na:

- Oddělovače (Delimiters);
- identifikátory (Identifiers);
- literály (Literals);
- klíčová slova (Keywords);
- komentáře (Comments).

Oddělovače jsou speciální znaky (=, (,), ;, :, mezera, apod.). Každý oddělovač má různý význam.

Identifikátory jsou alfanumerické řetězce znaků (TIME#14.7h, Spinac__Off, Pohyb__dolu apod.). Identifikátory slouží pro vyjádření jmen uživatelských funkcí, návěští nebo programových organizačních jednotek.

Literály slouží pro přímou reprezentaci hodnot proměnných (TRUE; 25; 3,79; modra atd.).

Klíčová slova jsou standardní identifikátory (BOOL, FUNCTION, REAL, VAR__OUTPUT, apod.). Jejich přesný tvar a význam odpovídá normě IEC 61 131-3. Klíčová slova se nesmějí používat pro vytváření jakýchkoli uživatelských jmen. Pro zápis klíčových slov mohou být použita jak velká tak malá písmena resp. jejich libovolná kombinace.

Komentáře nemají syntaktický ani sémantický význam. Komentáře slouží jako dokumentace programu. Při překladu jsou komentáře ignorovány. Mohou obsahovat znaky národních abeced. Překladač rozeznává dva druhy komentářů :

- Obecné komentáře - řetězce znaků začínající znaky (* a ukončené *);

- řádkové komentáře - začínající znaky // a ukončeny koncem řádku.

Datové typy. Pro programování v některém z jazyků podle normy IEC 61 131-3 jsou definovány datové typy: elementární, předdefinované, (Elementary data types) rodové datové typy (Generic data type) pro příbuzné skupiny datových typů, odvozené (uživatelské) datové typy (Derived data type, Type definition).

Elementární datové typy jsou charakterizované šířkou dat (počtem bitů) nebo rozsahem hodnot. Důležitým principem při programování podle normy IEC 61 131-3 je existence inicializační (počáteční) hodnoty pro všechny proměnné v programu. Pokud uživatel neuvede jinak, bude proměnná inicializována implicitní (předdefinovanou, default) hodnotou podle použitého datového typu [3]. Přehled podporovaných datových typů je uveden na obrazku č. 1.

Klíčové slovo	Anglicky	Datový typ	Bitů	Rozsah hodnot
BOOL	Boolean	Boolovské číslo	1	0,1
SINT	Short integer	Krátké celé číslo	8	–128 až 127
INT	Integer	Celé číslo	16	–32 768 až +32 767
DINT	Double integer	Celé číslo, dvojnásobná délka	32	–2 147 483 648 až +2 147 483 647
USINT	Unsigned short integer	Krátké celé číslo bez znaménka	8	0 až 255
UINT	Unsigned integer	Celé číslo bez znaménka	16	0 až 65 535
UDINT	Unsigned double integer	Celé číslo bez znaménka, dvojnásobná délka	32	0 až +4 294 967 295
REAL	Real (single precision)	Číslo v pohyblivé řádové čárce (jednoduchá přesnost)	32	±2.9E-39 až ±3.4E+38 Podle IEC 559
LREAL	Long real (double precision)	Číslo v pohyblivé řádové čárce (dvojnásobná přesnost)	64	Podle IEC 559
TIME	Duration	Trvání času	24d 20:31:23.647	
DATE	Date (only)	Datum	Od 1.1.1970 00:00:00	
TIME_OF_DAY nebo TOD	Time of day (only)	Denní čas	24d 20:31:23.647	
DATE_AND_TIME nebo DT	Date and time of day	„Absolutní čas“	Od 1.1.1970 00:00:00	
STRING	String	Řetězec	Max.255 znaků	
BYTE	Byte(bit string of 8 bits)	Sekvence 8 bitů	8	Není deklarován rozsah
WORD	Word (bit string of 16bits)	Sekvence 16 bitů	16	Není deklarován rozsah
DWORD	Double word (bit string of 32 bits)	Sekvence 32 bitů	32	Není deklarován rozsah

Obr. č. 1: Elementární datové typy [3].

1.3 Programovací jazyky

Jazyky podle normy IEC 61131-3 nevznikly jako teoretický vývoj, ale jako výsledek analýzy mnoha jazyků používaných v praxi a na požadavcích nabízených

výrobcům PLC. Jejich sémantika i syntaxe je přesně definována a neponechává žádný prostor pro nepřesné vyjadřování. Zvládnutím těchto jazyků se tak otevírá cesta k používání široké škály řídicích systémů, které jsou na tomto standardu založeny [3]. Standard stanoví pět programovacích jazyků s následujícími názvy:

- Jazyk strukturovaného textu (ST - Structured Text);
- jazyk seznamu instrukcí (IL - Instruction List);
- jazyk sekvenčního funkčního diagramu (SFC - "Sequential Function Chart");
- jazyk funkčního blokového schématu (FBD - Function Block Diagram);
- jazyk příčkového diagramu (jazyk kontaktních schémat) (LD - Ladder Diagram).

Programovací jazyky se dělí do textovych a grafickych jazyků. Grafické jazyky jsou SFC, FBD, LD. Jazyky IL a ST jsou textové.

Do normy bylo zavedeno několik jazyků, tak, aby každý uživatel mohl použít jazyk, kterému rozumí nejvíce. Programátoři mnohem častěji volí jazyk IL (podobný na assembler) nebo ST, podobný vysoceúrovňovému jazyku Pascal. Odborníci se zkušenostmi v reléové logice volí jazyk LD, a specialisti v systémech automatického řízení (ASŘ) a inženýři obvodů si vybírají svůj obvyklý jazyk FBD.

Volba jednoho z jazyků je dána nejen uživatelskými preferencemi, ale také významem řešeného problému. Pokud je počáteční problém formulován z hlediska sekvenčního zpracování a přenosu signálů, pak je jednodušší a jasnější používat jazyk FBD. Pokud je úkol popsán jako posloupnost operací určitých kláves a relé, pak je jazyk LD nejzřejmější. Pro úkoly, které jsou zpočátku formulovány jako komplexní rozvětvený algoritmus, bude jazyk ST pohodlnější. Jazyk je volen především dle firemních stanardů nebo dle přání uživatele.

Jazyky normy IEC 61131-3 jsou založeny na následujících zásadách:

- Celý program je rozdělen do mnoha funkčních prvků - programových organizačních jednotek (POU), z nichž každý může sestávat z funkcí, funkčních bloků a programů. Jakýkoliv prvek programu IEC může být konstruován hierarchicky z jednodušších prvků;
- Standard vyžaduje jednotnou formu zadávání dat. Zadávání datových typů usnadňuje detekci většiny chyb v programu před jeho provedením;
- Existují prostředky pro provádění různých programových fragmentů v různých časech, při různých rychlostech a také pseudoparalelně. Každý PLC funguje v cyklickém režimu. Cyklus začíná sběrem dat ze vstupních modulů, pak je program PLC spuštěn a cyklus končí výstupem dat do výstupních zařízení. Proto hodnota cyklu regulátoru závisí na době provádění programu a na rychlosti procesoru. Jeden programový fragment může například skenovat koncový senzor s frekvencí 100 krát za sekundu, zatímco druhý fragment snímá teplotní senzor s frekvencí jednou za 10 sekund;

- Provádět operace ve specifickém sledu, který je definován body v čase nebo událostmi, s použitím speciálního jazyka sekvenčních funkčních diagramů (SFC);
- Norma podporuje struktury pro popis různých dat. Stav teploty, tlaku a zapnutí/vypnutí čerpadla lze například popsat pomocí jediné struktury „Pomp" a přenášet v rámci programu jako jeden datový prvek;
- Standard zajišťuje sdílení všech jazyků, proto pro každý úsek úkolu může být vybrán jakýkoli, nejvhodnější jazyk;
- Program napsaný pro jeden regulátor lze přenést do jakéhokoliv regulátoru, který je kompatibilní s normou IEC 61131-3.

1.3.1 Grafické jazyky

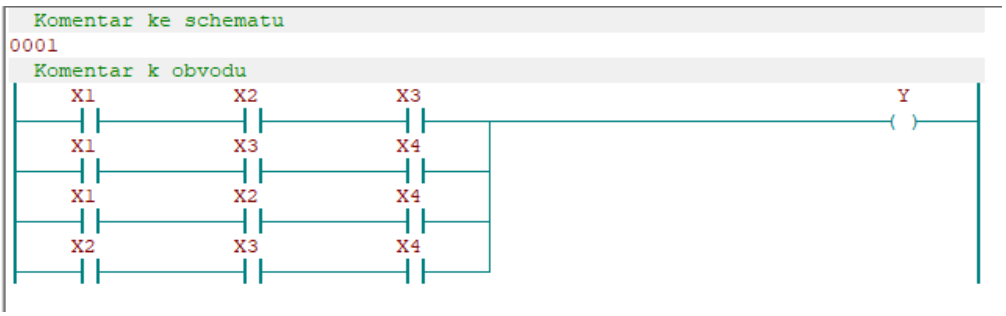
Jazyk příčkového diagramu (jazyk kontaktních schémat) (LD - Ladder Diagram).

První PLC byly používány v USA pro automatizaci výroby v automobilovém průmyslu v roce 1969. Protože v definici „programovatelného logického regulátoru" byla hlavní věc „programovatelného", téměř okamžitě vznikla otázka, jak programovat PLC? Algoritmické počítačové programovací jazyky té doby byly zaměřeny na řešení výpočetních problémů. Profese programátora byla považována za vzácnou a obtížnou, ve výrobě nebyli žádní odborníci. Ideální variantou by byl automatický překlad konceptů reléových automatů do PLC programů. Takto se v PLC objevil jazyk příčkového diagramu (LD - Ladder Diagram). Procesní technik mohl „překreslit" řídicí obvod na displeji programovací stanice PLC. Přirozeně, schéma nebylo znázorněno graficky, ale pomocí podmíněných symbolů. Jazyk příčkového diagramu byl intuitivní pro osoby, které jsou mírně obeznámeny s elektrotechnikou, a proto se ukázal jako nejběžnější v průmyslové automatizaci. Uživatel snadno zjistil poruchu v zařízení a sledoval signální cestu příčkového diagramu.

Jazyk LD je však problematický pro implementaci složitých algoritmů, zejména numerických, protože nepodporuje podprogramy, funkce, enkapsulace a další prostředky strukturování programů s cílem zlepšit kvalitu programování. Tyto nevýhody ztěžuje opakované použití softwarových komponent, což činí program dlouhým a obtížným pro údržbu.

Pro provádění komplexních aritmetických funkcí byly do jazyka LD přidány funkční bloky, které prováděly sčítání, násobení a průměrné výpočty. Obtížné výpočty v tomto jazyce nejsou výhodné. Další nevýhodou je, že pouze malá část programu se při programování vejde na počítačový monitor nebo ovládací panel.

I přes tyto nevýhody, patří jazyk LD mezi nejpoužívanější jazyky na světě, ačkoli ve většině případů je používán na jednoduché úkoly. Na obrázku č. 2 můžeme vidět příklad s použitím LD jazyka.



Obr. č. 2: Příklad programování v LD jazyce.

Jazyk funkčního blokového schématu (FBD - Function Block Diagram).

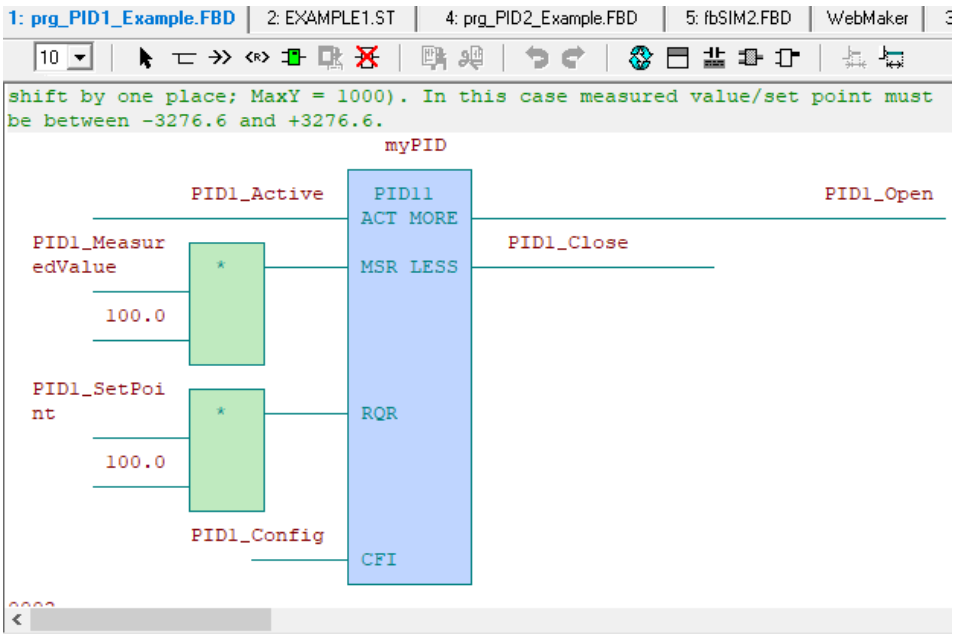
Jazyk funkčního blokového schématu (FBD) je grafický jazyk a je nejvhodnější pro programování procesů předávání signálů přes funkční bloky. FBD jazyk vznikl ve Francii. Jazyk FBD je vhodný pro obvodové techniky, kteřé dokáží snadno sestavit řídicí schéma na pevné logice s integrovanými obvody, ale nemají žádné zkušenosti s programováním.

Na obrázku č. 3 vidíme příklad programu napsaného ve funkčním jazyce FBD. Jak je vidět, takový obraz programu velmi jasně odráží algoritmus, což činí tento jazyk velmi jednoduchým a pohodlným pro řešení úloh v PLC.

V FBD je program tvořen určitým seznamem obvodů, které jsou prováděny po jednom seshora dolů. Program napsaný v grafickém jazyce FBD je soubor vzájemně propojených funkčních bloků, jejichž vstupy a výstupy jsou propojeny spojem. Spojení odráží určité programové proměnné, prostřednictvím spojovacích vodičů jsou mezi bloky data vyměňována.

Funkční bloky jsou fragmenty programů psaných v IL, SFC nebo jiných jazycích, které mohou být znovu použity v různých částech programu, a které odpovídají grafickému obrazu přijatému při vývoji funkčních diagramů elektronických zařízení.

Blok nese specifickou funkci (logický součin, negaci, čítač atd.). A jeden blok může mít několik vstupů a výstupů. Zpočátku jsou hodnoty proměnných nastaveny konstantami nebo pomocí speciálních vstupů, a jejich výstupy jsou dále připojeny k jiným programovým proměnným nebo k výstupům PLC.



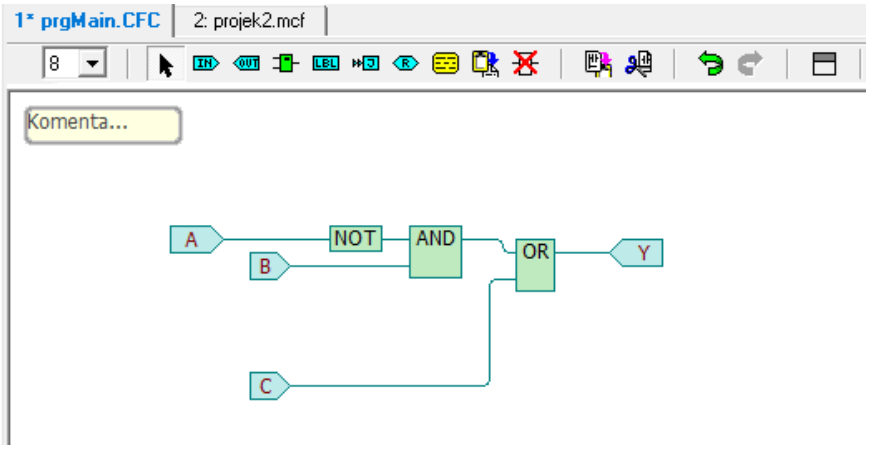
Obr. č. 3: Příklad programování v FBD jazyce.

Jazyk FBD lze použít k programování funkcí, funkčních bloků a programů. Dále také k popisu kroků a přechodů v jazyce SFC. Funkční bloky zapouzdřují data a metody, které se podobají objektově orientovaným programovacím jazykům, ale nepodporují dědičnost a polymorfismus.

Typickým použitím jazyka FBD je popis systémů pevné logiky, regulačních a dalších vypočetních algoritmu. Jazyk funkčních bloků je také vhodný pro vytváření a doplňování knihovny typických funkčních bloků, které lze znovu použít pro programování úloh v průmyslové automatizaci. Typické bloky zahrnují časovací jednotku, PID regulátor, sekvencorovou jednotku, klopný obvod, generátor impulsů, filtr a další.

Jazyk kontinuálního funkčního obvodu (CFC - "Continuous Flow Chart")

CFC je další vizuální programovací vysokoúrovňový jazyk. CFC se v podstatě vyvinul z jazyka FBD. CFC neni adoptován normou IEC 61131-3 ale je je velmi populární a rozšířený. Tento jazyk byl speciálně vytvořen pro návrh řídicích systémů pro kontinuální technologické procesy. Projektování přichází na výběr z knihoven hotových funkčních bloků, jejich umístění na ploše, navazování spojení mezi jejich vstupy a výstupy a nastavení parametrů vybraných bloků. Funkční bloky jazyka CFC na rozdíl od FBD provádějí nejen jednoduché matematické operace, ale jsou zaměřeny na správu celých technologických celků. Výhodou je logičnost a přehlednost pro malé programy, u větších programů se komplikuje.



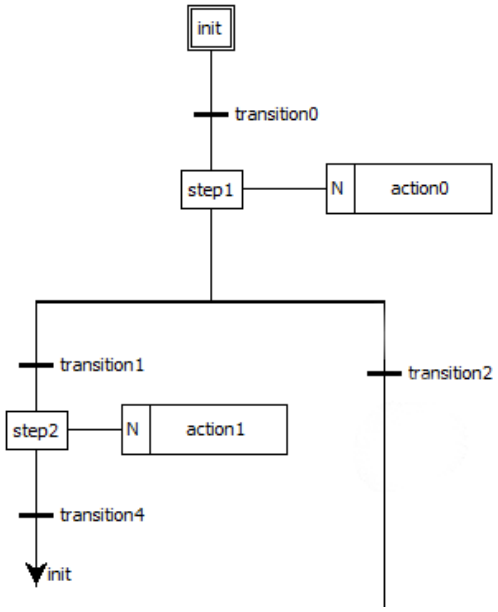
Obr. č. 4: Příklad programování v CFC jazyce.

Jazyk sekvenčního funkčního diagramu (SFC - "Sequential Function Chart")

SFC je nazýván jako programovací jazyk, ačkoli ve skutečnosti jazykem není, je to jen pomocný nástroj pro strukturování programů. Je navržen speciálně pro programování sledů akcí prováděných řídicím systémem, kdy tyto akce musí být prováděny ve specifikovaných časech nebo při výskytu určitých událostí. Je založen na reprezentaci řídicího systému s využitím konceptů stavů a přechodů mezi nimi.

Jazyk SFC je určen k popisu řídicího systému na nejvyšší úrovni abstrakce, například ve smyslu „Start", „Plnění nádoby", „Provádění stupně # 1", „Provádění stupně # 2", „Vyložení z nádoby". Jazyk SFC lze také použít k programování jednotlivých funkčních bloků, pokud je algoritmus jejich činnosti popsán přirozeným způsobem pomocí pojmů stavů a přechodů.

Na obrazku č. 5 je zobrazen fragment programu v jazyce SFC. Program se skládá z kroků a přechodových podmínek. Kroky jsou v diagramu znázorněny obdélníky, podmínky přechodu jsou označeny silnou příčnou čárou. Krok obsahuje akci, která bude prováděná, když je krok aktivní. Program běží seshora dolů. Počáteční krok v diagramu je zobrazen jako dvojitý obdélník. Podmínky přechodu jsou psány vedle jejich označení. Každý programový krok může být implementací komplexního algoritmu napsaného v jednom z jazyků IEC.



Obr. č. 5: Příklad programování v SFC jazyce.

1.3.2 Textové jazyky

Jazyk strukturovaného textu (ST - Structured Text).

Jazyk ST je textový jazyk na vysoké úrovni a velmi podobný jazyku Pascal. Jazyk ST má mnoho rozdílů od Pascalu, a je určen speciálně pro programování PLC. Obsahuje mnoho konstrukcí pro přiřazování hodnot proměnným, pro volání funkcí a funkčních bloků, pro psaní podmíněných výrazů větví, pro výběr operátorů a pro konstrukci iteračních procesů. Tento jazyk je určen především pro provádění složitých matematických výpočtů popisujících komplexní funkce, funkční bloky a programy. Příklad programu v jazyce ST je znázorněn na obrázku č. 6.

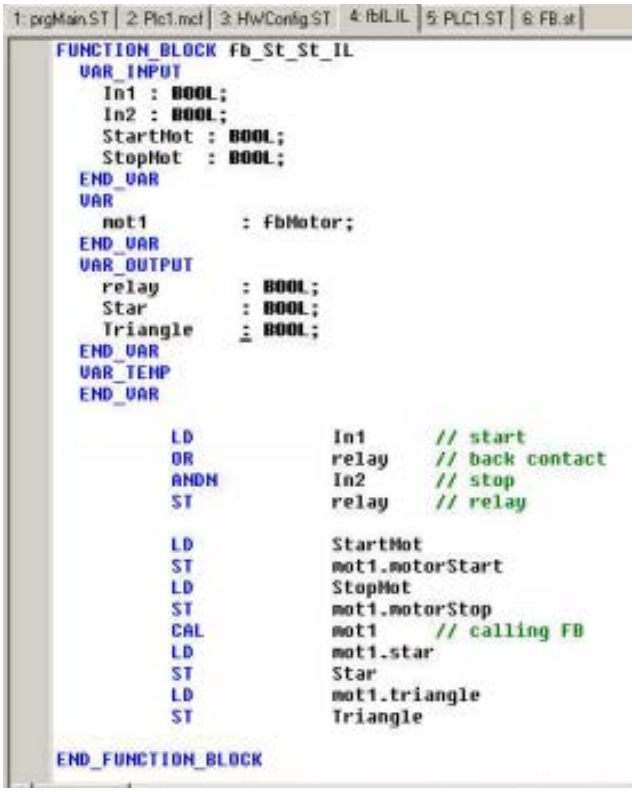
1: prgMain.ST | 2: fox2.mcf

```
PROGRAM prgMain
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR
  pokoj1 : fbTlacitko;
  pokoj2 : fbTlacitko;
  pokoj3 : fbTlacitko;


  END_VAR
  VAR_TEMP
  END_VAR
  pokoj1 (tl1:= vstup1(*BOOL*), tl2:= vstup2(*BOOL*),tl3:= vstup3(*BOOL*));
  vystup :=pokoj.sv;
  //      sc=> vystup

END_PROGRAM
```

Obr. č. 6: Příklad programování v ST jazyce.

Jazyk seznamu instrukcí (IL - Instruction List).

Jazyk IL připomíná assembler a slouží k implementaci funkce, funkčních bloků a programů, jakož i kroků a přechodů v jazyce SFC. Hlavní výhodou jazyka je jeho snadné učení, ale je náročný na programování neboť se blíží strojovému kodu. Jazyk IL se nejčastěji používá v případech, kdy je nutné získat optimalizovaný kód pro implementaci kritických částí programu, jakož i pro řešení malých problémů s malým počtem větví algoritmu. IL vznikl ve Německu. V dnešní době je málo efektivní a moc se nepoužívá. Příklad programu v jazyce IL je znázorněn na obrázku č. 7.



Obr. č. 7: Příklad programování v IL jazyce.

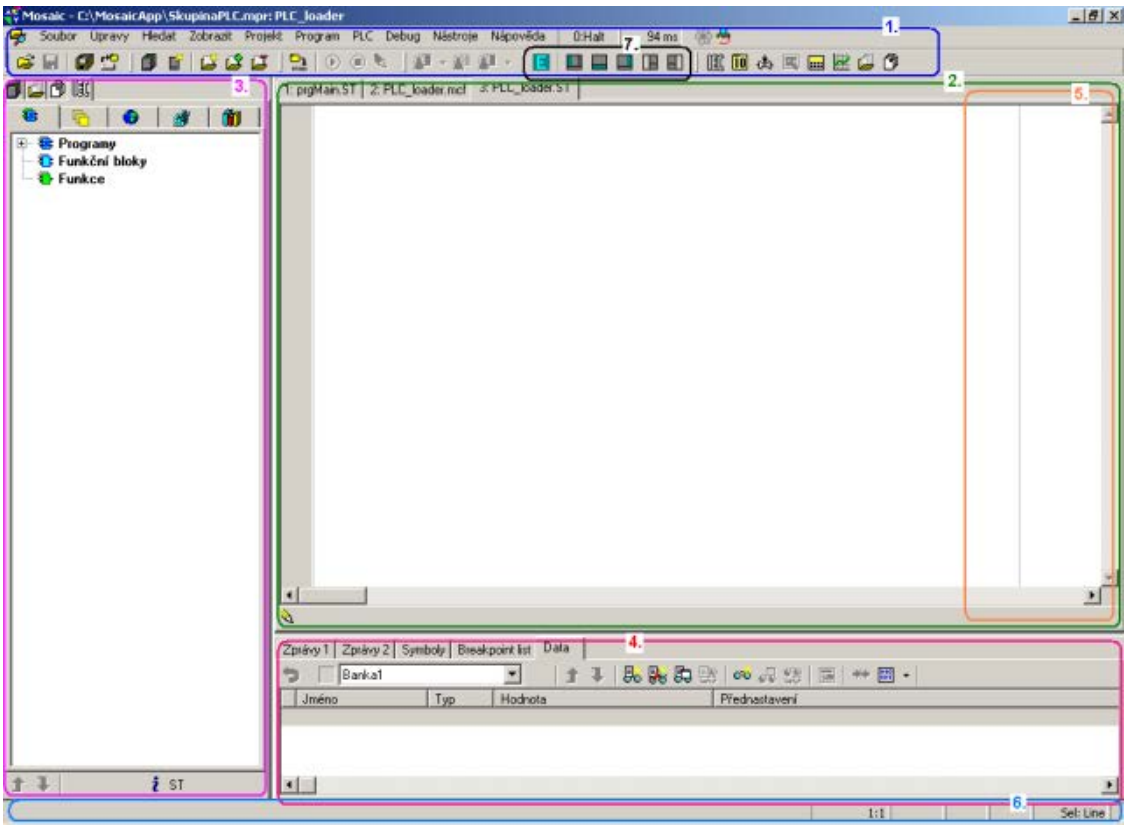
2 Vývojovy systém Mosaic

Mosaic je vývojové prostředí pro tvorbu a ladění programů pro programovatelné logické systémy (PLC, Programmable Logic Controller) TECOMAT z produkce firmy Teco a.s. Kolín. Program Mosaic je dodáván od roku 2000. Prostředí je vyvíjeno ve shodě s mezinárodní normou IEC EN-61131-3, která definuje strukturu programů a programovací jazyky pro PLC [4].

Společnost TECO dává každému možnost získat bezplatnou verzi Mosaicu, který funguje ve verzi Lite, která poskytuje plnou funkčnost programování pro všechny nástroje v Mosaicu bez omezení. Verze Lite umožňuje přístup k simulace a k programování nejmenších PLC z řady PLC TECOMAT. Pro větší typy PLC už je potřebný HW klíč. Mosaik můžeme nainstalovat na neomezený počet počítačů. Vývojové prostředí Mosaic se neustále zlepšuje, a nové verze jsou vydávány několikrát ročně, pričemž vše co bylo ve starých verzích zůstává v nových, ale objevují se nové funkční možnosti, jako je možnost programovat nové typy PLC z produkce Teco. Poslední platnou verzi Mosaicu můžeme stáhnout na stránce www.tecomat.cz.

2.1 Základní popis prostředí Mosaic

Začneme s popisem hlavního okna v prostředí Mosaic. Pro názornost se můžeme podívat na obrázek č. 8.



Obr. č. 8: Pracovní okno prostředí Mosaic [4].

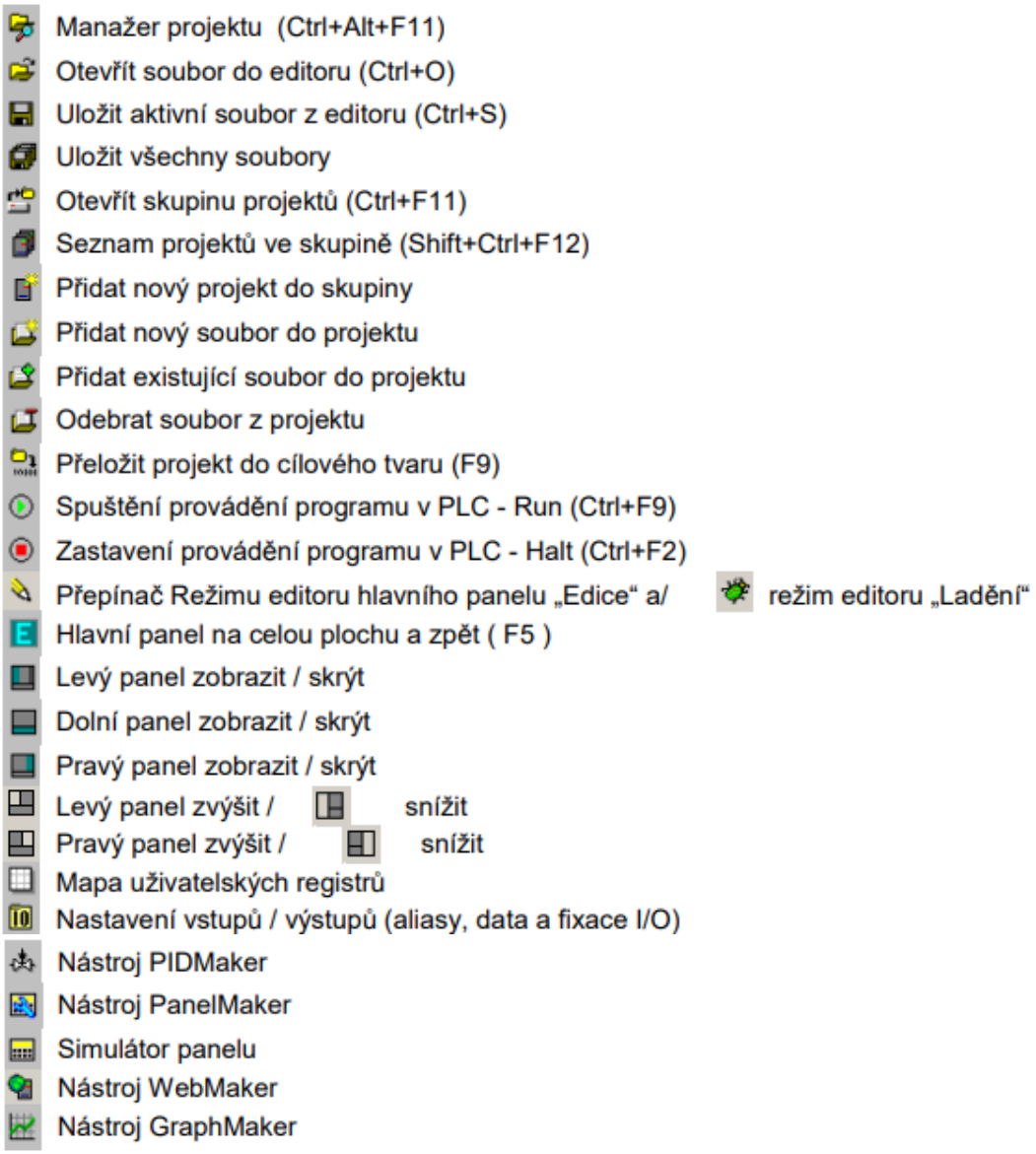
Pod čísly 1-7 z obrázku č. 8 popíšeme vzhled hlavního okna v prostředí Mosaic:

1- Hlavní menu, textové menu Mosaicu a hlavní panel nástrojů s grafickými ikonami.
2- Hlavní dokovací panel s otevřenými soubory.
3- Dokovací panel pro okna pomocných organizačních nástrojů.
4- Dolní dokovací panel pro okna informačních nástrojů.
5- Pravý dokovací panel pro okna nástrojů náhledů na oblasti paměti a proměnných v PLC.
6- Informační řádek pro informační texty a v pravé části je informace z aktivního editoru.
7- Ikony pro změnu rozměrů jednotlivých panelů.

Rozměry dokovacích panelů mužeme měnit uchopením jejich rozhraní a tažením. Můžeme měnit umístění jednotlivých oken nástrojů a editorů („dokováním"). Pro rychlé přepínaní mezi jednotlivými okny můžeme oknům přiřadit čísla od 1 do 9, a pomocí zkratky Alt+číslo mezi okny přepínat.

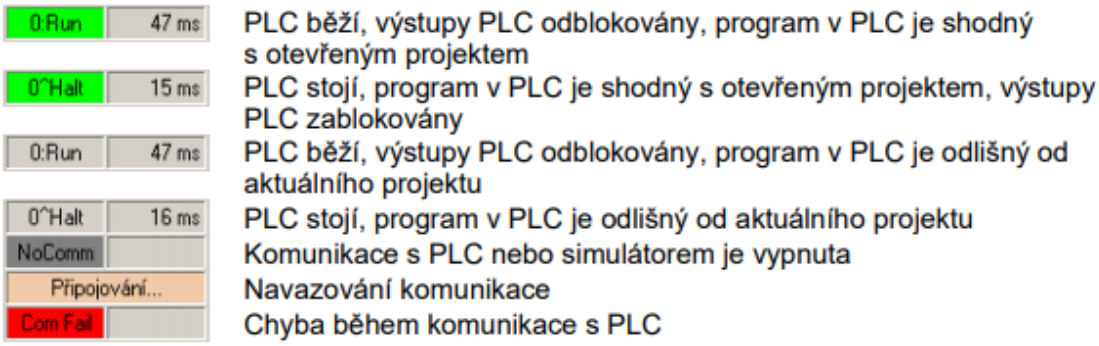
Hlavní menu prostředí Mosaic zahrnuje roletové menu, informace o stavu PLC, grafickou nástrojovou lištu a ikonu manažeru projektu.

Na obrázku č. 9 vidíme grafické ikony v nástrojové liště hlavního menu a jejich význam.



Obr. č. 9: Grafické ikony v nástrojové liště hlavního menu [4].

V menu prostředí Mosaic mužeme vidět informace o stavu PLC. Stavy PLC a jejich popis jsou na obrazku č. 10.



Obr. č. 10: Stavy PLC [4].

2.2 Přehled nástrojů prostředí Mosaic

Manažer Projektu. S tímto nástrojem můžeme definovát typ PLC, nastavit funkce jednotlivých modulů PLC, nastavit obecné funkce SW, drivery pro komunikace, datové propojení mezi jednotlivými projekty PLC.

Nastavení vstupů/výstupů. Okno pro data vstupů a výstupů, můžeme vstupním a výstupním signálům přiřadit jména a zafixovat hodnoty do libovolných stavů. Zobrazuje po překladu výsledné absolutní adresy vstupů a výstupů. Zviditelňuje také výsledné vstupní a výstupní adresy po překladu.

IEC manažer. Nástroj je určen pro organizaci a editaci položek v uživatelském programu podle normy IEC 61 131-3. Má několik záložek:

- „POU" - programovatelné organizační jednotky;
- „Typy" - typy proměnných;
- „Globální proměnné" - globálně dostupné proměnné;
- „Konfigurace"- organizace úloh a instancí v program;
- „Knihovny" - přehled zařazených knihoven a jejich obsahu.

PIDMaker. Používá se pro jednoduchou implementaci, ladění a řízení regulačních algoritmů.

PanelMaker. Nástroj definuje obsah obrazovek pro textové operátorské panely.

Grafický PanelMaker. Podobá se nástroji PanelMaker, ale definuje obsah obrazovek pro grafické operátorské panely.

POU Inspektor. Když je PLC v režimu RUN, používá se pro základní prohlížení programu.

WebMaker. Nástroj určený pro zobrazování a nastavování proměnných v Mosaicu, a pro vytváření XML stránek pro webový server v centrálních jednotkách a základních modulech.

GraphMaker. Vytváří grafické zobrazení 16 proměnných PLC ve formě časového diagramu.

Simulátor textových panelu. Nástroj je určen pro testování programové obsluhy operátorského panelu.

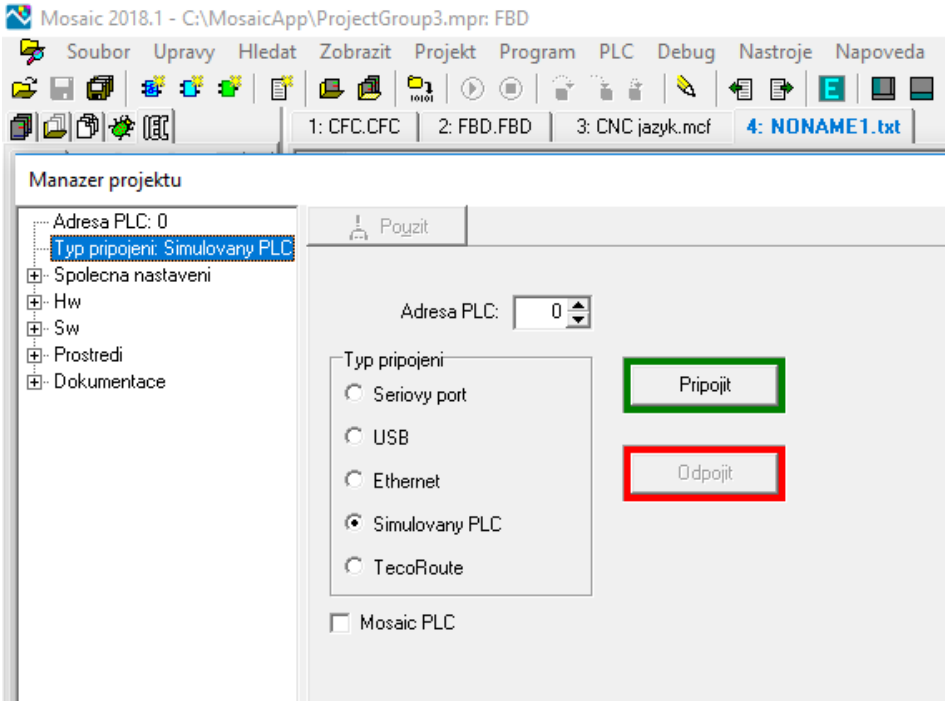
Nástroj Panel. Starší verze „WebMakeru". Slouží k semigrafickému zobrazování a nastavování proměnných v programu.

Mapa uživatelských registrů. Slouži k zobrazování obsazené paměti s uživatelskými registry, a umožňuje kontrolu překrytí definic proměnných.

2.3 Manažer projektu

Manažer projektu slouží k definování typu PLC, nastavení obecných funkcí SW driverů pro komunikaci, sestavení PLC a nastavení funkcí jednotlivých modulů, datového spoje mezi jednotlivými projekty PLC navzájem, a k zahrnutým do této skupiny projektů textovým operátorským panelům.

Okno Manažeru projektu se nachází v levé části pracovní plochy Mosaic a obsahuje rozbalovací strom s přehledem všech skupin nastavitelných základních parametrů, a plochu s objekty sloužícími k nastavování parametrů. Rozbalením jednotlivých uzlů a výběrem položky se otevře okno nastavení parametrů.



Obr. č. 11: Okno Manažeru projektu.

Společná nastavení obsahuje 3 okna:

- Okno které informuje o aktivních programových modulech, pro rozšíření funkčních možnosti prostředí Mosaic.
- Okno globální volby pro nastavení volby projektu na tovární hodnoty, nebo jako výchozí pro nové projekty.
- Okno nastavení složek na nastavení implicitních adresářů pro ukládání projektů a archivních kopií.

Konfigurátor HW obsahuje také 3 okna:

- Výběr řady PLC pro základní výběr řídícího systému;
- Konfigurace HW pro nastavování komunikačních kanálů, a nastavení parametrů periferních modulů;

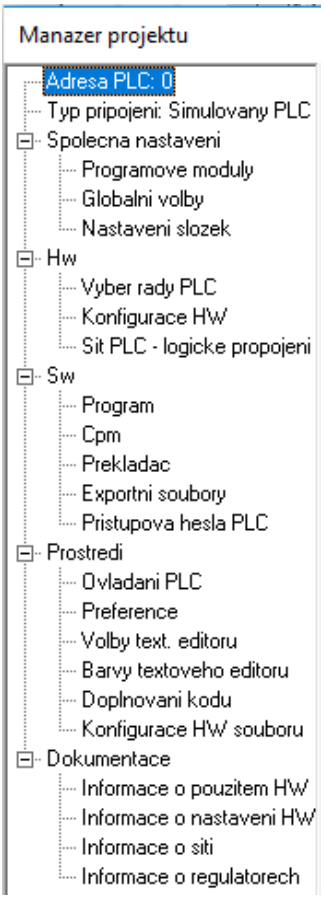
- Síť PLC - logické propojení pro grafický popis sítí PLC a dalších objektů, jako je nadřazené PC, switche, zobrazovací panely, huby, zařízení na CanOpen, Profibus DP apod.

Konfigurátor SW obsahuje 5 oken:

- Program je informační okno o aplikačním programu a knihovně;
- Cpm je okno pro nastavení centrálního modulu PLC;
- Překladač je okno nastavení překladače;
- Exportní soubory je okno pro různé typy souborů (pux, exp, pub);
- Přístupová hesla PLC (povolena pouze s připojeným HW klíčem).

Prostředí:
Obsahuje okna: ovládání PLC, preference, volby textového editoru, barvy textového editoru, doplňování kódu, konfigurace HW souboru. Zde nastavujeme parametry chování prostředí.

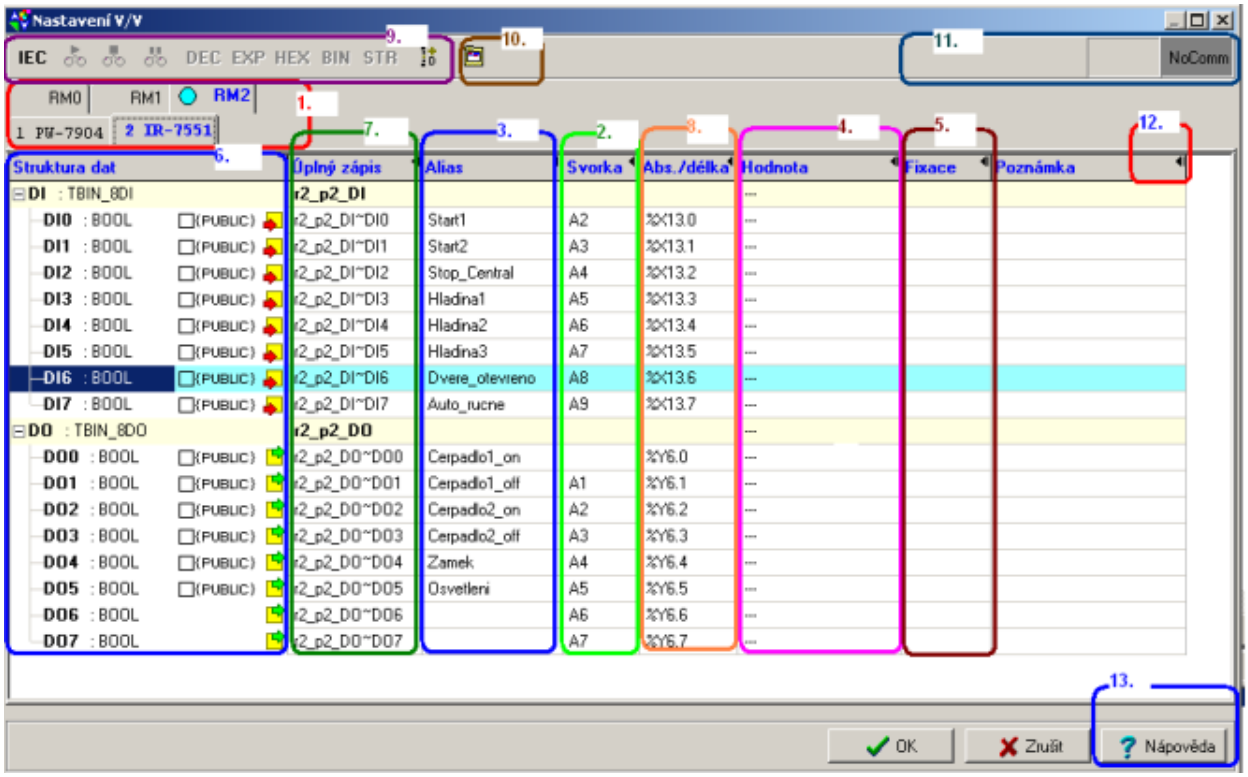
Dokumentace:
Okno, ve kterém jsou zobrazena nastavení HW a SW PLC.



Obr. č. 12: Strom Manažeru projektu.

2.4 Nastavení vstupů a vystupů

Kliknutím na ikonu „Nastavení vstupů/výstupů" otevřeme nástroj pro komplexní nastavení a správu vstupů/výstupů. Nástroj zobrazuje datovou strukturu periferních modulů a umožňuje, aby jednotlivým proměnným byla přiřazena vlastní jména (Alias), která později budeme používat při programování. Pokud je PLC připojen v režimu RUN, pak zobrazuje aktuální hodnoty všech proměnných. Pokud je to nutné, můžeme během ladění fixovat jejich hodnoty do zvoleného stavu. Po překladu zobrazuje výsledné absolutní adresy vstupů a výstupů, povoluje přiřadit vstupům a výstupům pevné absolutní adresy.



Obr. č. 13: Nástroj nastavení V/V [4].

1- Záložky, jsou struktura sestavy řídicího systému ve více rámech, která slouží pro vybrání periferního modulu podle jeho pozice v určitém rámu.
2- Svorka. Označení svorky na konektoru modulu.
3- Alias. Sloupec pro zadání vlastního jména proměnné přiřazené konkrétnímu vstupu a výstupu Změny aliasu jsou přijímány pouze po překladu programu a po jeho zapsání do PLC.
4- Hodnota. Sloupec ukazuje aktuální hodnotu připojeného nebo simulovaného I/O PLC
5- Fixace. Slouží k nastavení fixovaných hodnot proměnných během odlaďování algoritmů.
6- Struktura dat. Stromová struktura dat dostupná na vybraném modulu: stavová informace, řídicí slova apod.
7- Úplný zápis. Systémové jméno proměnné ve struktuře, přidělené automaticky nebo implicitně.
8- Abs/délka. Ukazuje absolutní adresu proměnné.
9- Ovládací lišta. Používá se pro volbu způsobu zobrazení dat: IEC (zápis absolutních jmen podle IEC normy (%I %Q) nebo podle syntaxe Tecomat (%X %Y)) , Start, Stop, Zmrazení, DEC EXP HEX BIN STR (formát dat ve sloupci Hodnota), Signum (se znaménkem/bez znaménka).

10- Mapa obsazení vstupů a výstupů. Používá se pro přehled obsazeni V/V prostoru PLC a ruční změnu adresace V/V modulů.

11- Stavové informace. Zobrazují signalizace zapnutého režimu fixace, signalizace platnosti zobrazovaných dat a pracovní režim připojeného PLC (RUN/HALT) a stavu komunikace.

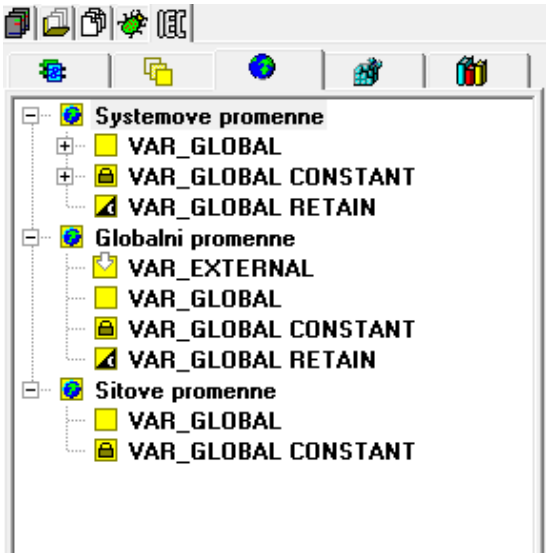
12- Tlačítka pro zobrazení sloupců tabulky.

13- Nápověda k proměnným v datové struktuře zvoleného I/O modulu.

2.5 IEC Manažer

Nástroj, který je určen k organizaci a úpravě prvků v uživatelském programu v souladu s normou IEC 61 131-3. IEC Manažer je automaticky vložen do levého panelu, a je rozdělen do pěti záložek:

- POU (programovatelné organizační jednotky). Předpisy programovatelných organizačních jednotek;
- Typy (typy proměnných). Systémové typy, které jsou definované v projektu;
- Globální proměnné (globálně dostupné proměnné). Struktura globálně dostupných proměnných;
- Konfigurace (organizace úloh, a instancí v programu). Organizace úloh a instancí v projektu;
- Knihovny (přehled zařazených knihoven a jejich obsahu). Zobrazení zařazených knihoven a jejich obsahu.



Obr. č. 14: IEC Manažer.

Na obrázcích znázornující tabulky, jsou sepsány všechny klávesové zkratky, které usnadňují práci s programem Mosaic.

Obecné:

F1	Vyvolává nápovědu kontextově k pozici kurzoru
Ctrl+F11	Otevřít projektovou skupinu
Ctrl+N	Vytvořit nový dokument
Ctrl+O / F3	Otevřít existující dokument
Ctrl+F4	Zavřít existující dokument
Alt+F4	Zavřít nedokované okno
Ctrl+F4 / Alt+F4	Zavřít dokované okno
Ctrl+S / F2	Uložit aktivní dokument
Shift+F12	Seznam otevřených oken editorů
Alt+1 az Alt+9	Přímý přístup na okno číslo 1 az 9
F6	Následující aktivní editor
Shift+F6	Předchozí aktivní editor
Ctrl+Tab	Následující zadokované okno v panelu
Shift+Ctrl+Tab	Předchozí zadokované okno v panelu
Ctrl+F12	Seznam souborů projektu
Shift+Ctrl+F12	Seznam projektů ve skupině
Escape	Zavřít modální okno (obyčejně okno s dialogem pro nastavování)

Ovládání PLC:

Alt+F2	Připojit / odpojit komunikaci s PLC
Ctrl+F2	Halt PLC
Ctrl+F9	Run PLC
F9	Přeložit projekt
Shift+F9	Vyslat kód do PLC
Alt+F6	Zapnutí / Výpnutí ladění
Ctrl+F5	Vyčíslit / Nastavit proměnnou
Ctrl+F7	Přidat položku do okna Data

V okně Textových editorů:

Ctrl+P	Tisknout aktivní dokument
Ctrl+X / Shift+Del	Vystřihnout text z dokumentu do textové schránky
Ctrl+C / Ctrl+Ins	Kopírovat text z dokumentu do textové schránky
Ctrl+V / Shift+Ins	Vložit text z textové schránky do aktivního dokumentu
Ctrl+A	Vybrat všechen text v aktivním dokumentu
Ctrl+ PgUp	Přejít na první řádek aktivního dokumentu
Ctrl+ PgDn	Přejít na poslední řádek aktivního dokumentu
Shift + PgUp	Vybrat až na první řádek aktivního dokumentu
Shift + PgDn	Vybrat až na poslední řádek aktivního dokumentu
Ctrl+B	Vybrat blok textu v aktivním dokumentu
Shift +Home	Vybrat do začátku řádku
Ctrl+ Home	Přejít na začátek aktivního dokumentu
Shift+Ctrl+ Home	Vybrat do začátku aktivního dokumentu
Shift +End	Vybrat do konce řádku
Ctrl+End	Přejít na konec aktivního dokumentu
Shift+Ctrl+End	Vybrat do konce aktivního dokumentu
Enter / Shift+ Enter	Zalomit řádek
Ins	Mód vkládání/přepisování (toggle mode)
Ctrl+Z / Alt+BackSpace	Vrátit předchozí akci, je-li to možné
Shift+Ctrl+Z / Shift+Alt+BackSpace	Obnovit opět předchozí akci, je-li to možné
BkSpace	Smazat poslední znak z textu
Shift + BkSpace	Smazat poslední znak z textu
Ctrl+ BkSpace	Smazat poslední slovo z textu
Ctrl+ T	Smazat do konce slova v textu
Del / Ctrl+Del	Smazat znak nebo označený blok textu z dokumentu
Ctrl+Y	Smazat celý řádek v aktivním dokumentu
Shift+Ctrl+Y	Smazat do konce řádku v aktivním dokumentu
Shift+Ctrl+0 az 9	Položit zarážku v textu 0 az 9
Ctrl+0 az 9	Skok na zarážku v textu 0 az 9
Shift+Ctrl+M	Přepnutí funkce označení sloupcových bloků a zpět
Shift+Ctrl+N	Přepnutí funkce označení sloupcových bloků a zpět
Shift+Ctrl+L	Zapnutí funkce označení řádkových bloků (vypnutí Shift+Ctrl+N)
Tab	Vložení tabelátoru (pracuje i pro označeny blok řádků)
Shift+Tab	Vyjmutí tabelátoru (pracuje i pro označeny blok řádků)
Home	Kurzor na první znak v řádku (podruhé na začátek řádku)
End	Kurzor za poslední znak v řádku

Obr. č. 15: Klávesové zkratky č. 1.[4]

Hledání a nahrazení:

Ctrl+F	Hledat v aktivním dokumentu
Ctrl+R	Nahradit v aktivním dokumentu
Ctrl+L	Opakovat poslední hledání / nahrazení v aktivním dokumentu
Ctrl+G	Jdi na řádek v aktivním dokumentu
Shift+F3	Hledat ve všech dokumentech
Shift+Alt+F3	Hledat ve všech dokumentech jako výstup(pro*.MOS a *.MAS)

IEC asistent:

Ctrl+D	Definovat IEC proměnnou
Shift+Ctrl+V	Vložit již definovanou proměnnou
Ctrl+I	Hledat proměnnou v IEC manžéru
Ctrl+J	IEC asistent
Ctrl+Space	Doplnit IEC kód

V okně IEC manžeru:

Alt+Enter	Vlastnosti
Ctrl+A	Přejít k originálu (pro aliasy na proměnné)
Ctrl+C	Kopírovat
Ctrl+F	Hledat
Ctrl+I	Přejít k textové reprezentaci položky
Ctrl+L	Najít další
Ctrl+T	Přejít k definici typu
Shift+Ctrl+I	Přejít k instancím (dostupné pouze pro POU a uživatelské typy)
Insert	Přidat programovou organizační jednotku / Přidat úlohu
Shift+Insert	Přidat proměnnou / Přidat instanci programu
Delete	Vymazat položku

Kontext ke zprávám:

Alt+F7	Kontext na předchozí událost z okna zprávy
Alt+F8	Kontext na následující událost z okna zprávy

Ovládání oken nástrojů:

Alt+0	Seznam otevřených oken
Ctrl+F12	Seznam souboru v projektu
Ctrl+Alt+F11	Manažér projektu
Ctrl+Alt+M	Okno Zprávy
Ctrl+Alt+W	Okno Data
Ctrl+Alt+Y	Okno Paměť
Ctrl+Alt+A	Okno Akumulátory
Ctrl+Alt+S	Okno Symboly
Ctrl+Alt+B	Okno Breakpoints list
Ctrl+M	Mapa registrů

Ovládání dokovacích panelů:

F5	Maximalizovat / Obnovit hlavní dokovací panel
Ctrl+Alt+Left	Zobrazit / Skrýt levý dokovací panel
Ctrl+Alt+Right	Zobrazit / Skrýt pravý dokovací panel
Ctrl+Alt+Down	Zobrazit / Skrýt dolní dokovací panel

Ladění:

Escape	Opustit POU inspektor
Shift+Alt+F11	Freeze all POU inspectors
Shift+Ctrl+F8	Pripojit/odpojit polozene pasti
Shift+F8	Past povolit/zakazat
F4	Polozit past
Shift+F4	Polozit podminenou past

Obr. č. 16: Klávesové zkratky č. 2.[4]

3 Uživatelska přiručka pro výuku

3.1 Python

Python je vysokoúrovňový skriptovací programovací jazyk. Nabízí dynamickou kontrolu datových typů a podporuje různá programovací paradigmata, včetně objektově orientovaného, imperativního, procedurálního nebo funkcionálního [8].

3.1.1 Tkinter

Tkinter je balíček Pythonu určený pro práci s knihovnou Tk. Knihovna Tk obsahuje grafické uživatelské rozhraní (GUI) napsané v programovacím jazyce Tcl.

Grafické uživatelské rozhraní (GUI) se vztahuje na všechna okna, tlačítka, textová pole, posuvníky, seznamy, přepínače a zaškrtávací tlačítka, které se zobrazují na obrazovce a otevírají aplikaci. S jejich pomocí komunikujeme s programem a dokážeme ho řídit. Všechny tyto prvky rozhraní se nazývají widgety (widgets). V současné době téměř všechny aplikace vytvořené pro koncového uživatele mají GUI.

Existuje mnoho GUI knihoven. Tk je jedena z nejpopulárnějších. S jejím použitím bylo napsáno mnoho projektů. Instalační soubor Pythonu obvykle obsahuje balíček tkinteru jako součást standardní knihovny spolu s dalšími moduly. Tkinter může být popsán jako překladatel z Pythonu do Tcl. Napíšeme program v Pythonu, a kód modulu tkinter převede vaše instrukce do Tcl, kterému Tk rozumí.

GUI aplikace jsou řízeny událostmi, což jsou akce orientované na události. To znamená, že jedna nebo jiná část programového kódu začíná být prováděna pouze tehdy, když se stane událost. Události mužou být různého charakteru. Muže jít o kliknutí tlačítka myši, stisknutí tlačítka Enter, začátek psaní textu, přepínání přepínacího tlačítka nebo posunutí stránky dolů/nahoru. Pokud je provedena některá z těchto akcí, ke které byl vytvořen odpovídající kód, je spuštěna část programu. Tohle všechno vede k odpovídajícímu výsledku.

Chceme-li napsat program GUI, postupujeme takto:

1. Tvorba hlavního okna.

2. Vytvoření widget a konfigurace jejich vlastností.

3. Identifikace události - nastavení na co bude program reagovat.

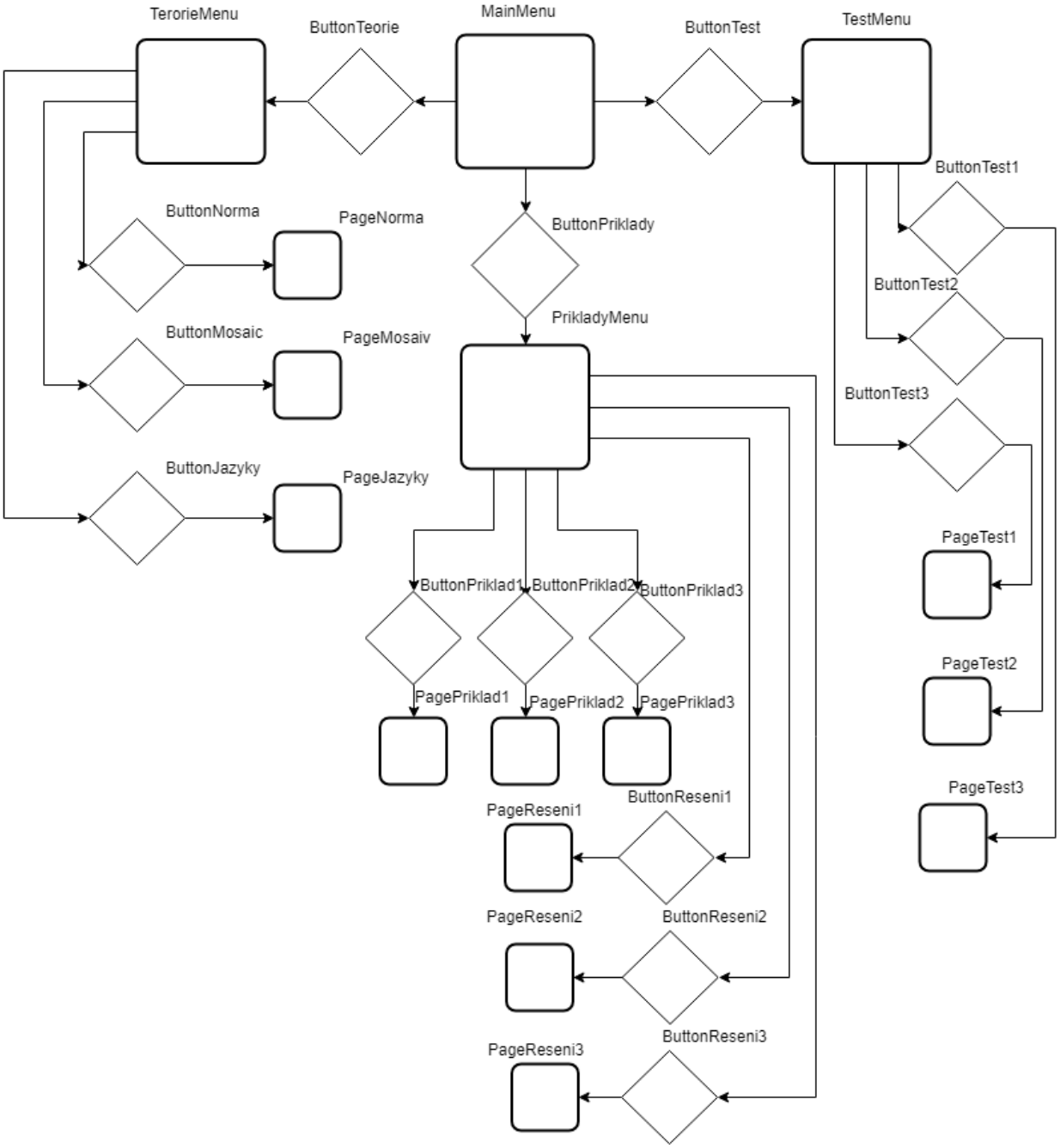
4. Definování kódů událostí - jak bude program reagovat.

5. Uspořádání widget v hlavním okně.

6. Spuštění cyklu událostí.

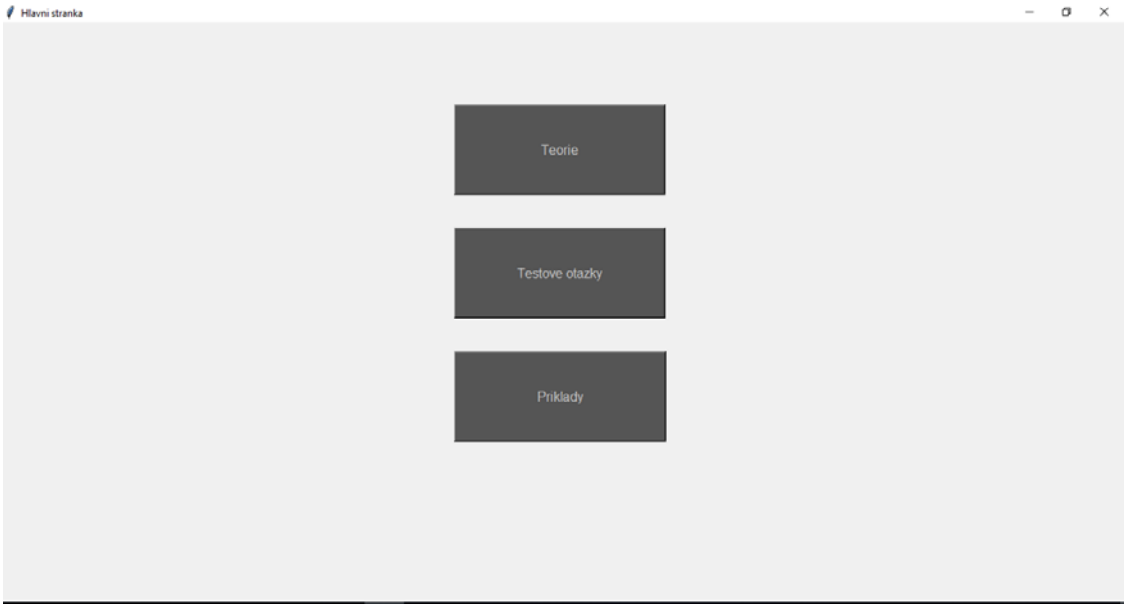
3.2 Struktura programu

Program, který jsem napsal v Pythonu, se skládá z mnoha oken a tlačítek, která spouštějí okna. Na obrázku č. 17 můžeme pozorovat strukturu programu.



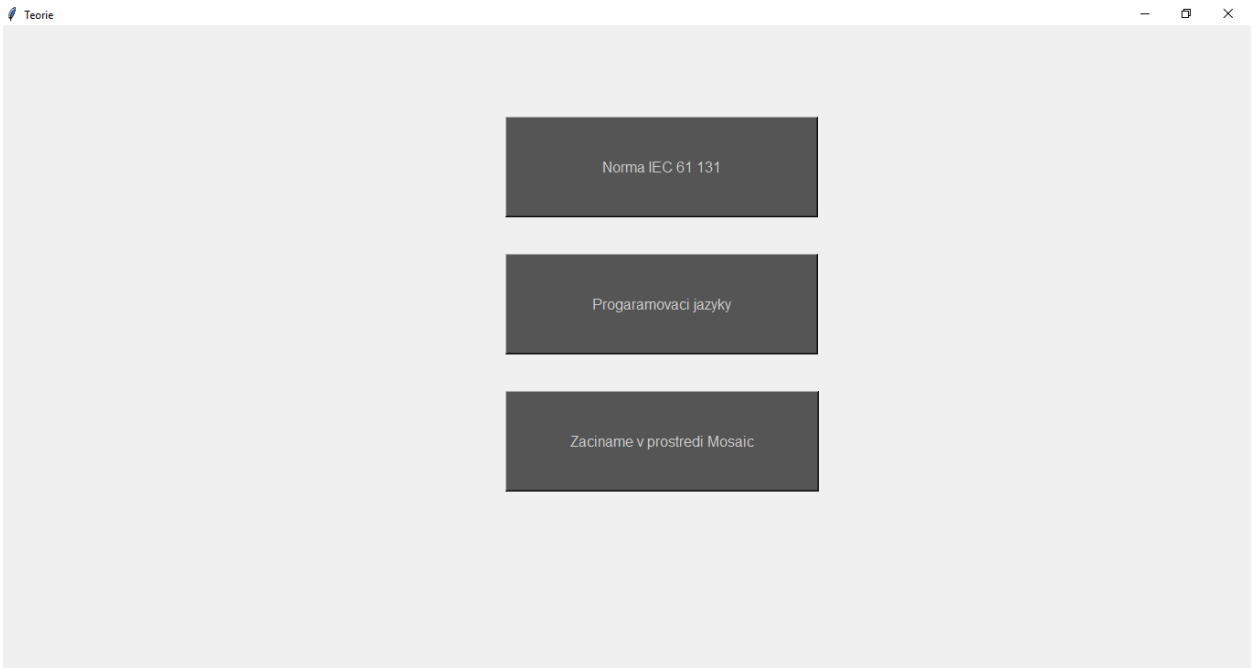
Obr. č. 17: Struktura programu.

Při spuštění programu vidíme hlavní menu, které obsahuje tři možné přechody, které můžeme aktivovat pomocí tlačítek a dostat se přes ně do jiných sekcí: menu teorie, menu testy, a menu příklady.



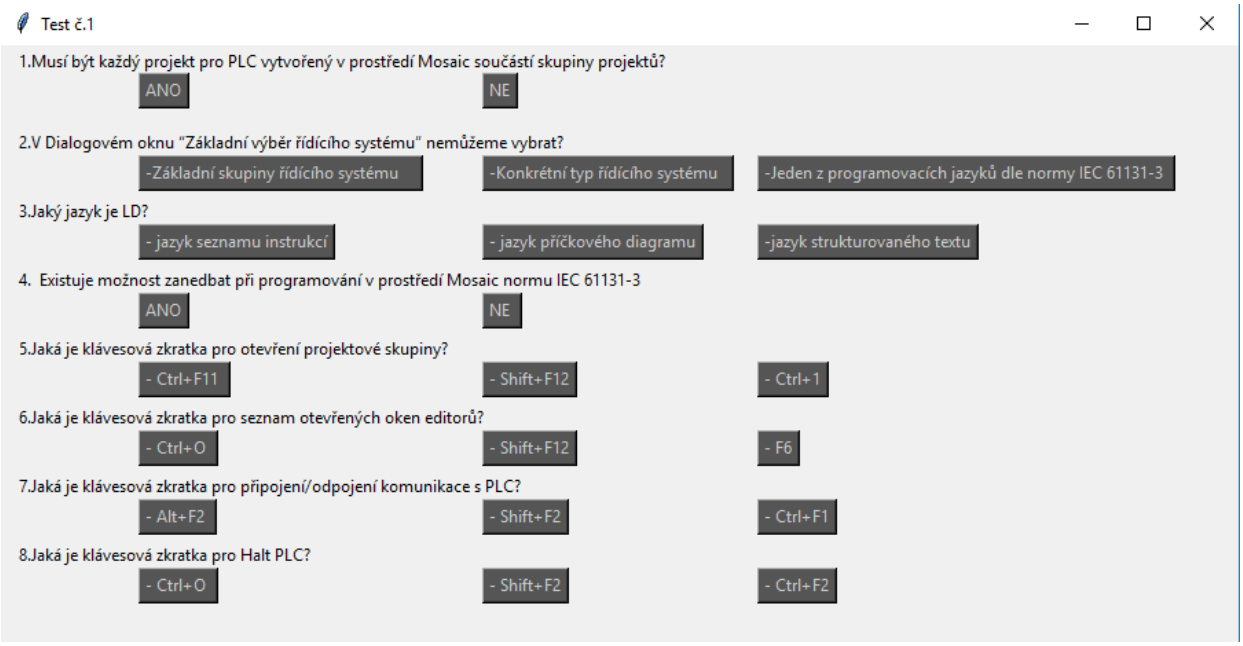
Obr. č. 18: Hlavní menu.

První sekce se nazývá Teorie. V této sekci se nalézá seznámení s normou IEC 61131-3, programovací jazyky a s vývojovým systémem Mosaic.



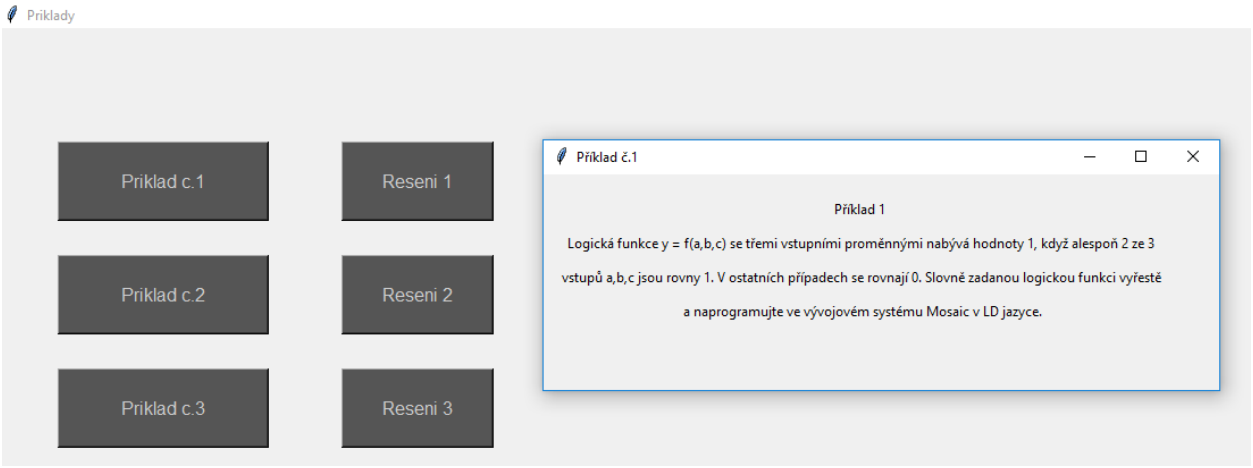
Obr. č. 19: Sekce Teorie.

Druhá sekce má název Testové otázky. Tato sekce má edukační funkci, kde student může zjistit jakou má úroveň znalostí. Testy jsou vytvořeny na principu otázky s variantami odpovědí. Ze zobrazených variant je pouze jedna správná. Pokud student vybere tlačítko se správnou odpovědí, program mu pošle okno s nápisem „Správně“, pokud vybere jinou odpověď bude mu posláno okno s nápisem „Špatně“.



Obr. č. 20: Sekce Test.

Třetí sekce nese název Příklady. Sekce obsahuje číselně značená tlačítka s příklady z předmětu Automatické řízení, vedle kterých jsou tlačítka s řešením jednotlivých příkladů v Mosaicu.



Obr. č. 21: Sekce Příklady.

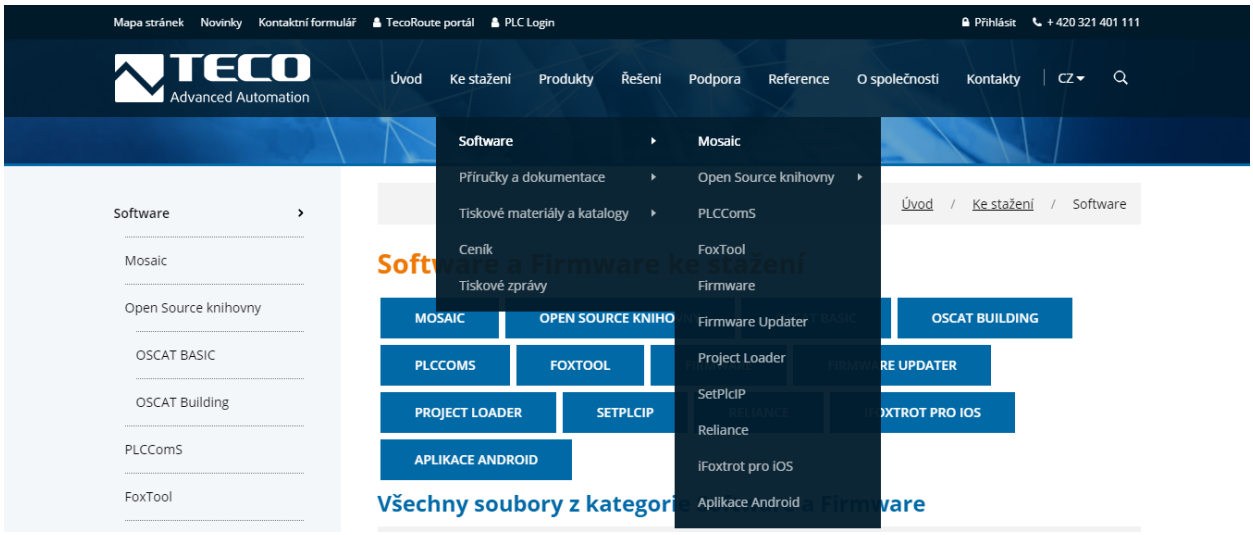
3.3 Teorie

Na stránce Teorie jsou tři tlačítka. Kliknutím na tlačítko se otevře stránka s teoretickým materiálem k danému tématu. Témata obsahují normu IEC 61131-3, jazyky programovaní a vývojový systém Mosaic.

V předchozich kapitolach výše jsme popsali normu IEC 61131-3 a programovací jazyky pro PLC, nyní popíšeme začátek práce v Mosaicu.

Projektem v Mosaicu je myšlen program pro PLC, včetně všech souvisejících souborů. Programy pro řídící systémy se skládají z jednotlivých souborů. Některé si vytváří sám programátor a jiné jsou tvořeny automaticky. Je důležité, že každý projekt pro PLC ve vývojovém prostředí Mosaic musí být součástí skupiny projektů. Skupina projektů obsahuje nejméně jeden či více projektů, které jsou součásti celé sítě řídicího systému. Projekty ve skupině mezi sebou mohou mít komunikační vazby a tak vytváří celek. Každý projekt je tvořen samostatnou složkou, ve které najdeme všechny potřebné soubory a informace pro programování jednoho řídicího systému.

K programování úloh musíme mít nainstalovaný program Mosaic. Program Mosaic je volně dostupný na stránkách výrobce zařízení Tecomat Foxtrot na adrese www.tecomat.com. Klikneme „Ke stažení", poté „Software", a „Mosaic". Na webové stránce stáhněme soubor „Mosaic 2019.1 SP1".



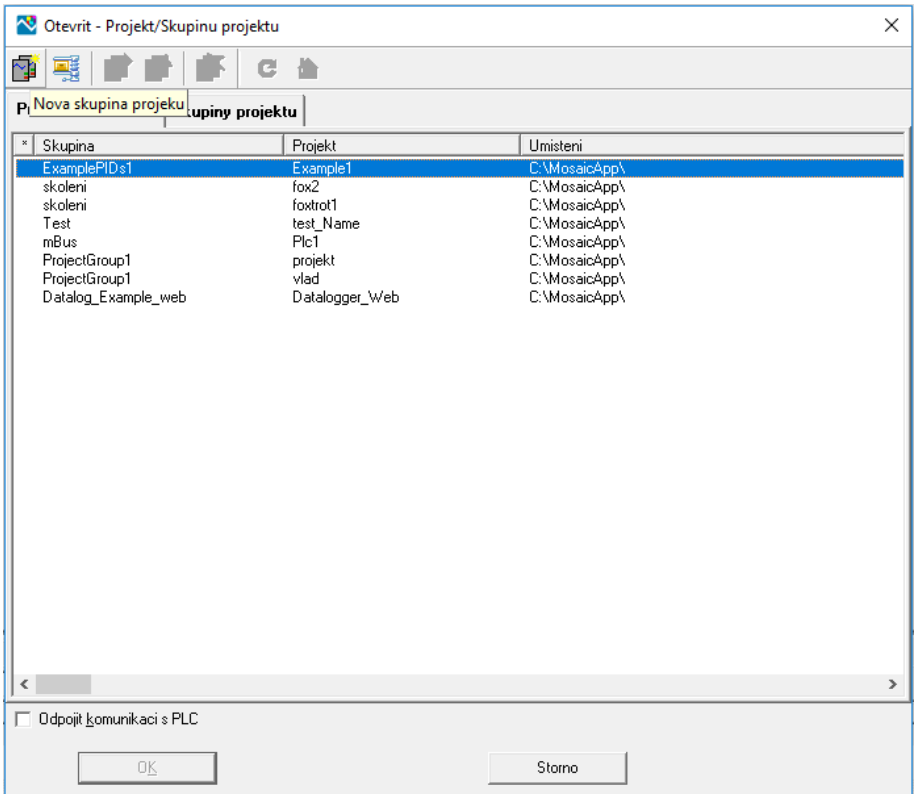
Obr. č . 22: Program Mosaic dostupný ke stažení [1].

Po stažení programu klikneme na ikonu Mosaic, poté naběhne úvodní obrázek, který můžeme vidět na obrázku č. 23. Po otevření těchto oken se zobrazí dialog pro založení nového projektu, nebo pro otevření dříve uložených projektů.

Obr. č. 23: Úvodní obrázky [4].

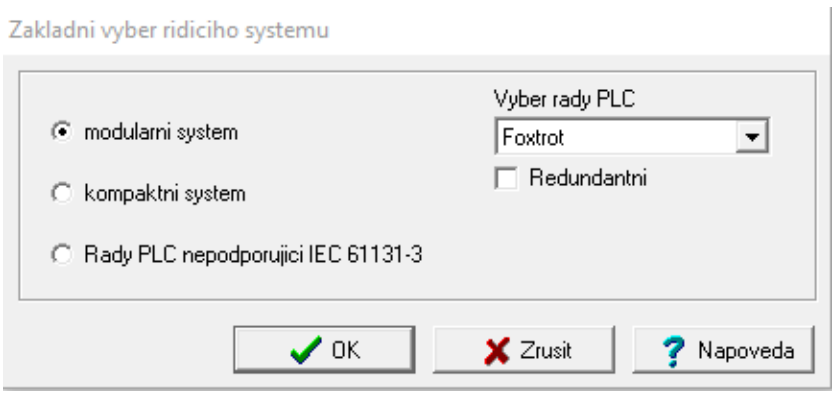
Vytvoření nové skupiny projektů

Po stisknutí tlačítka „OK" se otevře okno, kde uvidíme již existující projekty nebo skupiny projektů. Můžeme si zde založit novou skupinu projektů. Dále následuje posloupnost oken, která nás povede k vytvoření nového projektu. Zadáme jméno nové skupiny projektů. Zvolíme doporučenou konfiguraci (HW konfigurace) a zadáme jméno nového projektu.



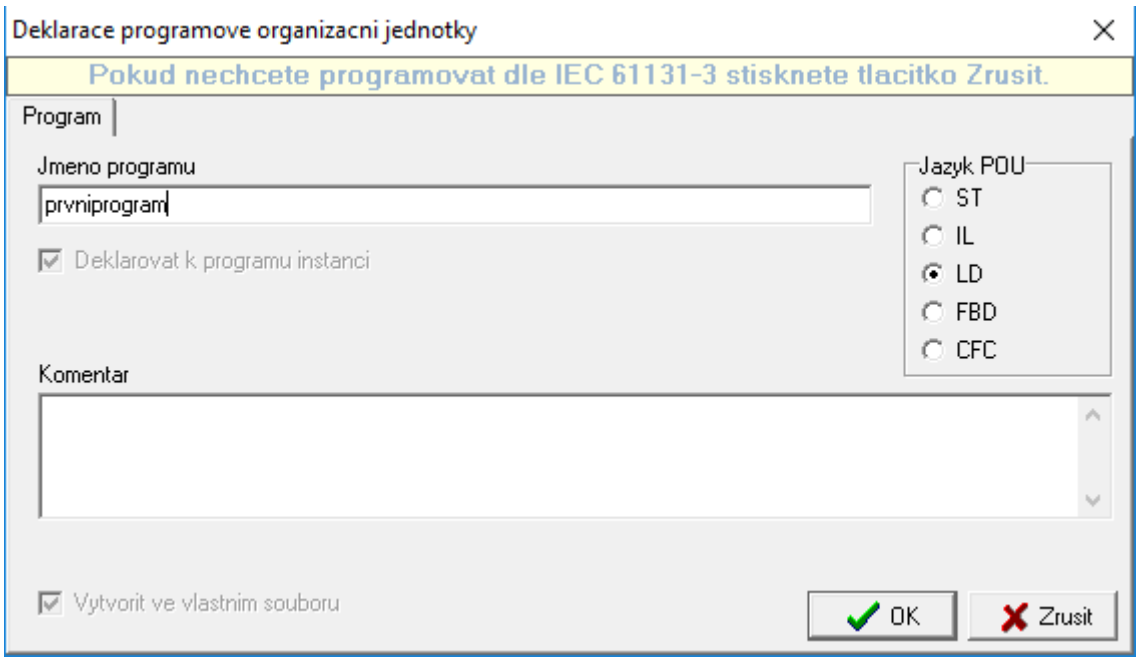
Obr.č. 24: Okno pro vytvoření nové skupiny projectů.

Dále bude následovat okno Základní výběr řídícího systému. Toto okno slouží k určení typu PLC produkce Teco, na němž program cílově poběží.

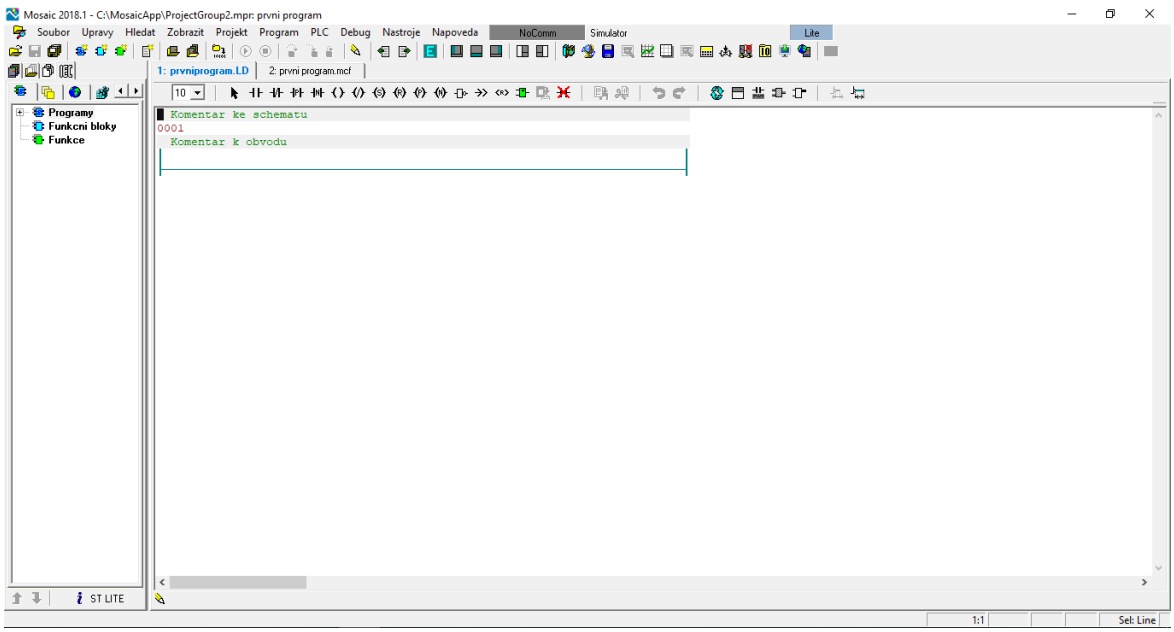


Obr. č. 25: Okno pro základní výběr řídícího systému [4].

Vybereme si modulární systém a výběr řady PLC - Foxtrot a stiskneme „OK“. Dále se nám otevře další okno Deklarace programové organizační jednotky (obr. č. 26). Zde zadáme jméno programu, a máme možnost výbrat jazyk, ve kterém budeme programovat. První program uděláme v LD (Ladder diagram). Stiskneme „OK“, poté znovu „OK“ a otevře se nám hlavní okno v prostředí Mosaicu, kde bude probíhat samotné programování (obr. č. 27).



Obr. č. 26: Okno pro deklarace programové organizační jednotky.



Obr. č. 27: Okno pro programování v LD.

Příklad v jazyce LD

Priklad 1.1

Světlo na chodbě je ovládáno binárním signálem Y ze dvou míst dvoupolohovými přepínači A, B. Přičemž informaci o poloze přepínačů A, B podávají signály I1, I2 (vstupy), a pro signál Y - Q1 (výstup).

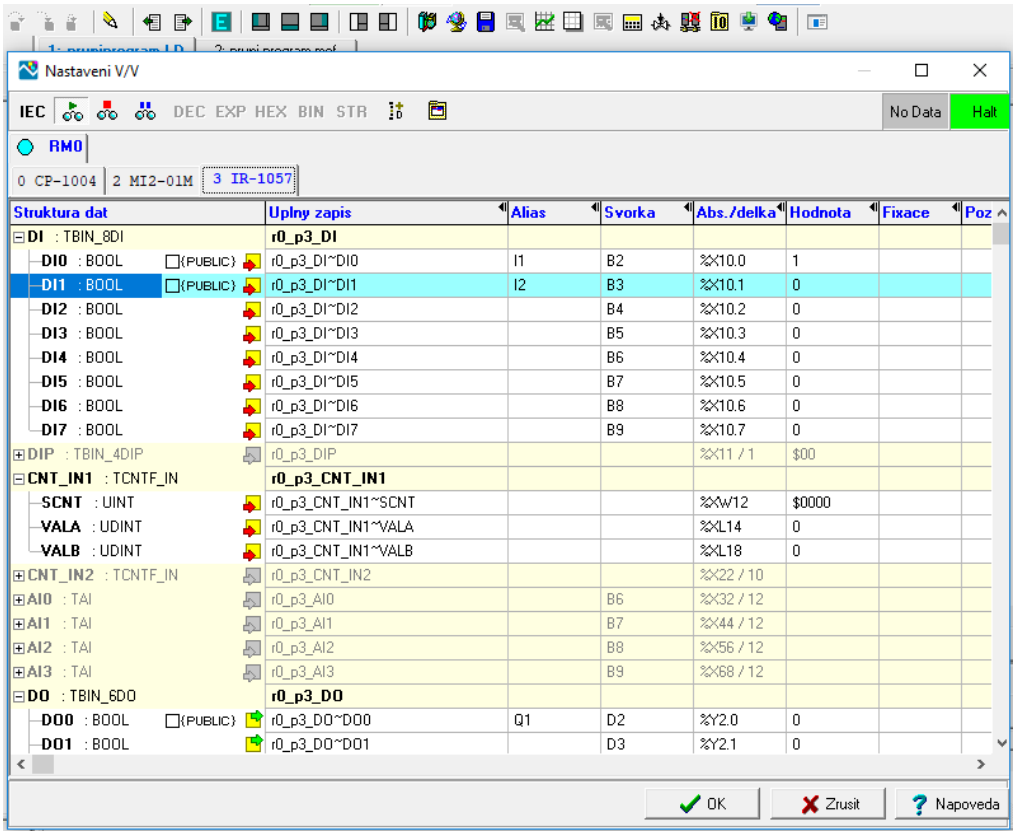
Rešení:

$Q1 = I1 * \overline{I2} + \overline{I1} * I2$

Pomocí nástroje Nastavení V/V provedeme adresaci proměnných.

Obr. č. 28: Okno pro programování.

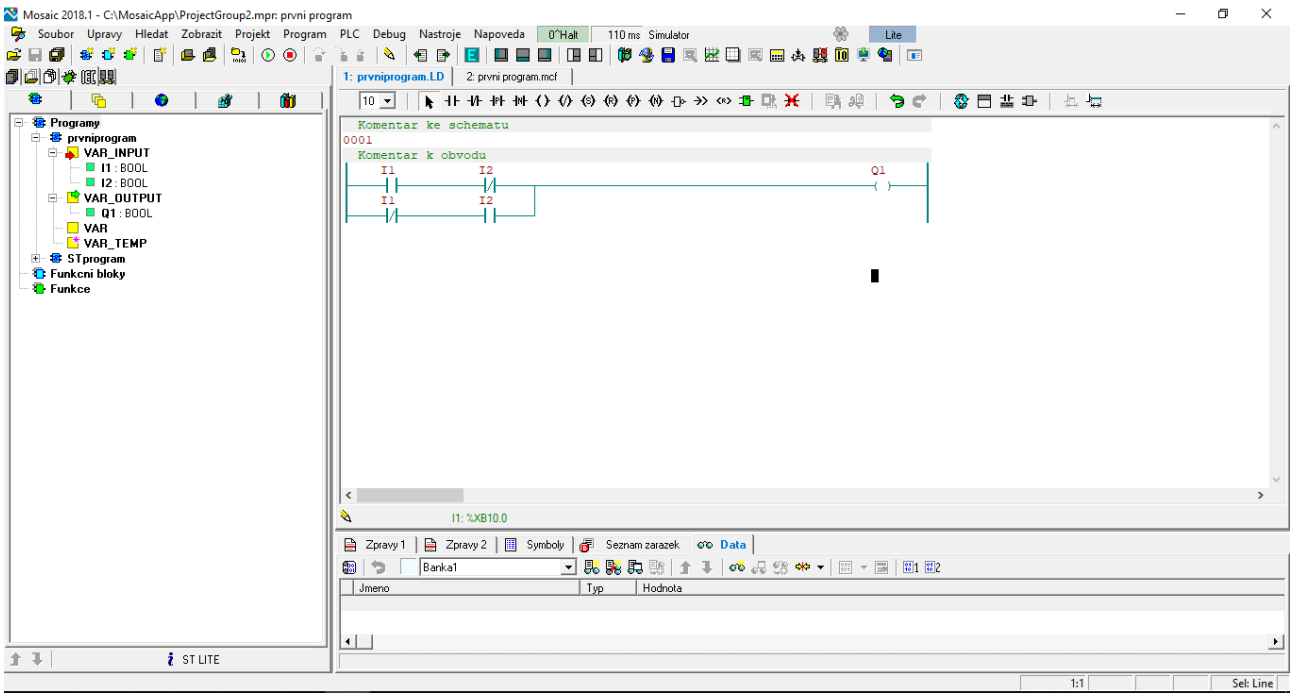
V režimu simulace můžeme nastavit hodnoty na vstupy.



Obr. č. 29: Nastavení V/V.

Dále je potřeba na pracovní ploše vytvořit kontaktní obvod, který v sobě bude mít paralelně zapojené větve, ze dvou sériově zapojených kontaktů. Klikneme levým tlačítkem myši na grafický symbol zvolené instrukce (vstup) ⊣⊢. V obvodech uvidíme dva body, v červené a modré barvě, pro umístění této instrukce.

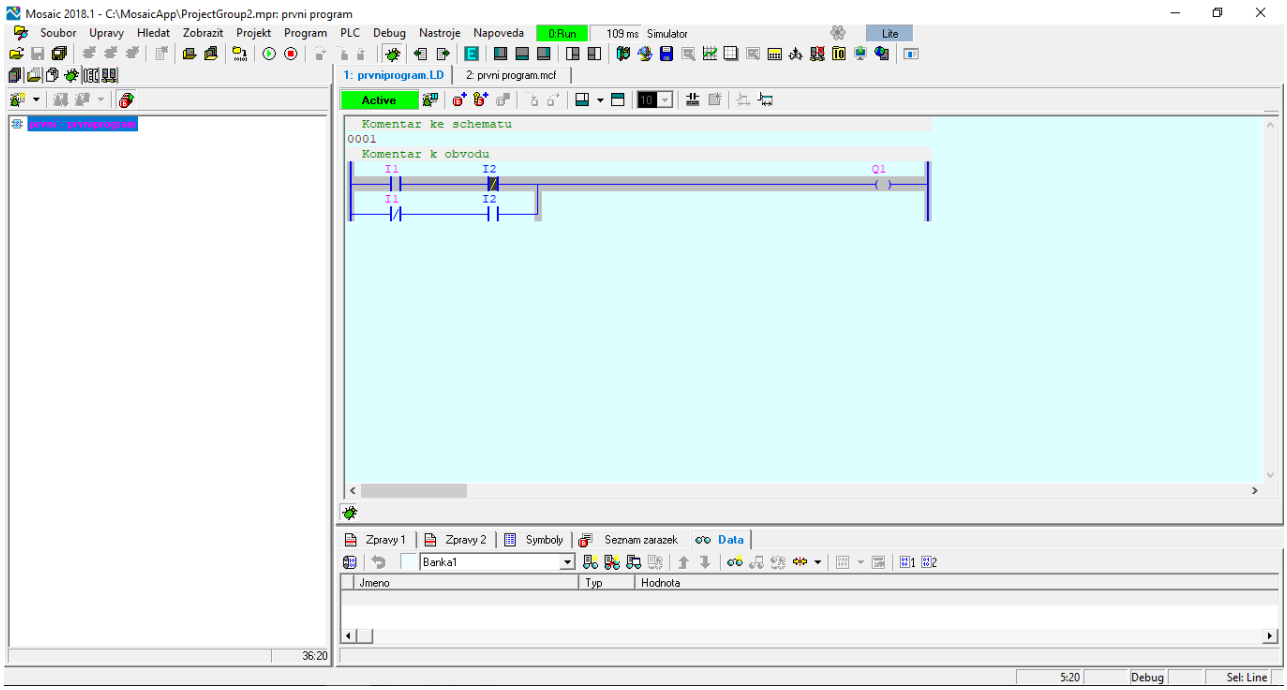
Modrý bod je pro paralelní zapojení, červený pro sériové zapojení. Uděláme první větev (seriově zapojeni), každému kontaktu přidáme jméno (I1 a I2). Pro I2 použijeme negovaný kontakt |/|. Pro vytvoření paralelní větve musíme stisknout levé tlačítko myši a tažením označit část schématu, ke kterému chceme přidat paralelní větev. V obvodu se objeví modrý bod, pomocí kterého obvod můžeme rozvětvit. V druhé větvi uděláme stejným postupem sériové zapojení, za pomoci červeného bodu. Obvod dokončíme přidáním výstupu (Q1) ( ) na konce.



Obr. č. 30: Okno pro programování v LD.

Dokončený obvod přeložíme (Spustíme v menu: Program – Přeložit) a spustíme Simulator (Run). Objeví se okno pro výběr typu restartu, kde následně zvolíme teplý nebo studený.

Na pracovní ploše vidíme aktivitu vstupů a výstupů. V simulaci můžeme měnit hodnoty vstupu dvojitým kliknutím myši na kontakt.



Obr. č. 31: Okno pro programování v LD.

Příklad v jazyce FBD

Pro jasnost budeme řešit stejný příklad v různých jazycích.

Rešení:

$Q1 = I1 * \overline{I2} + \overline{I1} * I2$

Pro vytvoření programu v jiném jazyce, opakujte všechny naše akce:

- Založíme novou skupinu projektů a zadáme jméno nové skupiny projektů

- Zvolíme doporučenou konfiguraci (HW)

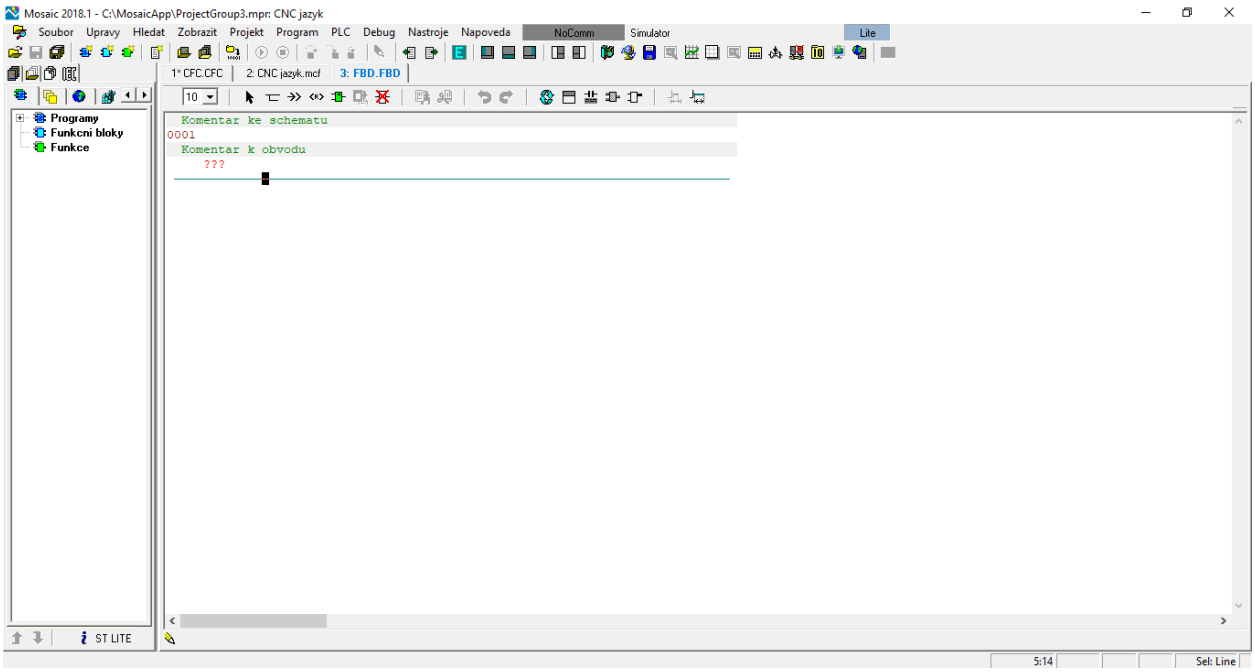
- Zadáme jméno nového projektu

- Vybereme si modulární systém a výběr řady PLC

- Zadáme jméno programu, a vybereme jazyk, ve kterém budeme programovat

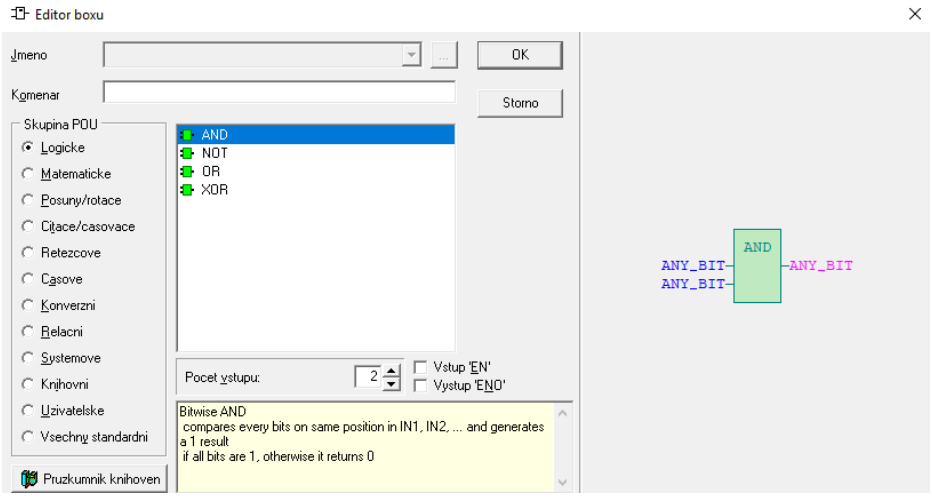
- Dvakrát stiskneme „OK“, poté se otevře okno Mosaic pro samotné programování v našem jazyce

- Definujeme vstupy a výstupy



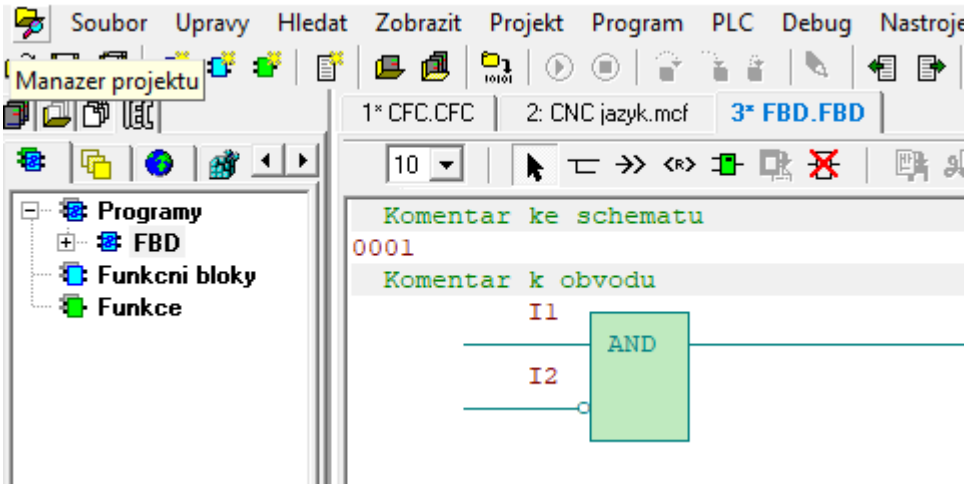
Obr. č. 32: Okno pro programování v FBD.

Zvolíme symbol Vlozit box (POU)  a umístíme ho do schématu kliknutím myši na červený bod. V okně, které se objeví vybereme skupinu logických POU a zvolíme blok AND s dvěma vstupy pro první součin.



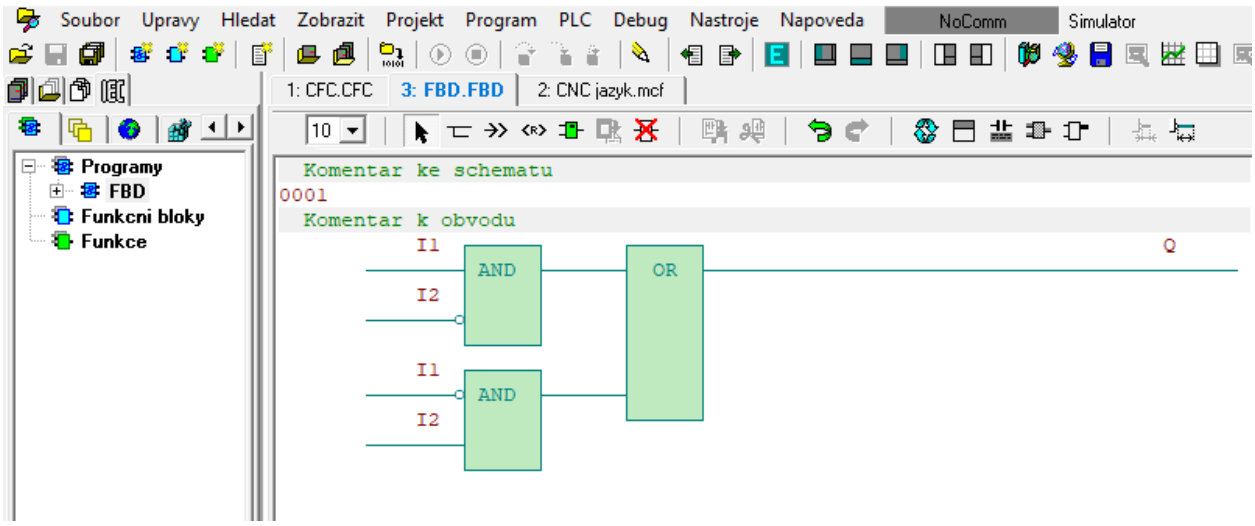
Obr. č. 33: Editor boxu.

Stiskneme tlačítko „OK", a na pracovní ploše se objeví součinové hradlo. Klikneme na vstupy našeho bloku a vyvoláme okno, kde zadáme naše proměnné (I1, $\overline{I2}$). Ve druhé proměnné musíme zaškrtnout negace ($\overline{I2}$). Takto jsme vytvořily první součinový blok.



Obr. č. 34: Součinový blok v FBD.

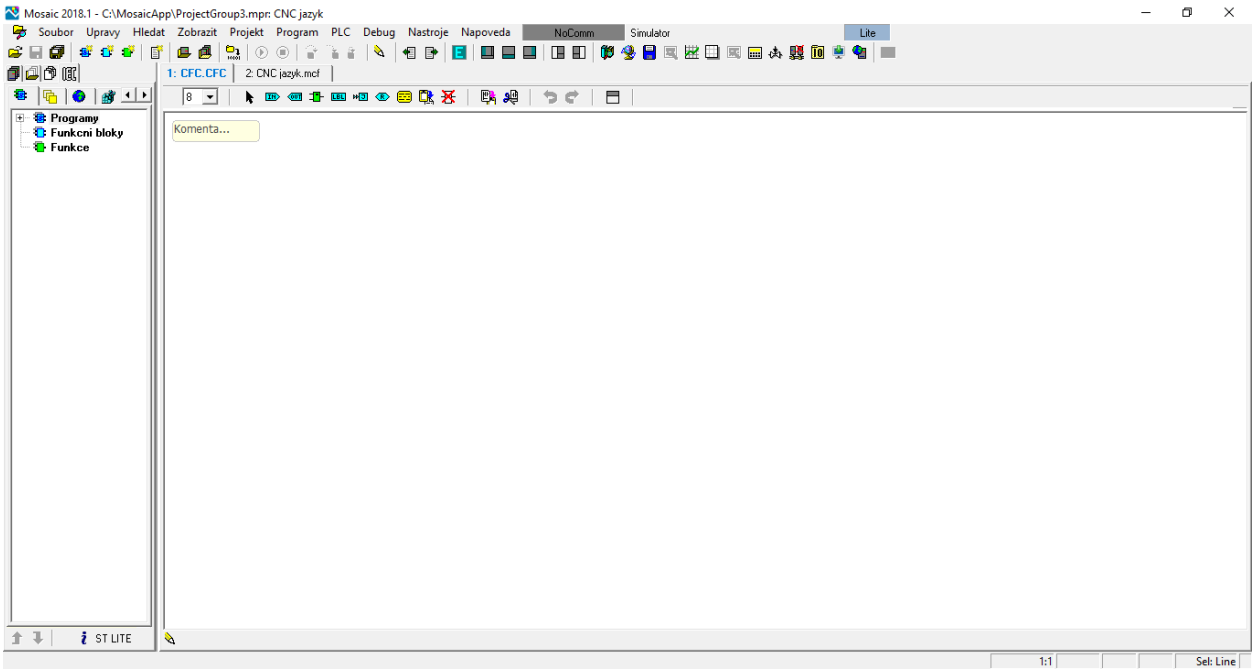
Vytvoříme další box logického součtu se dvěma vstupy a připojíme ho na výstup vytvořeného prvního součinového bloku. Další box připojíme na volný vstup součtového bloku. Vybereme blok logického součinu a zadáme naše proměnné ($\overline{I1}$*I2), nezapomeneme zaškrtnout negaci pro proměnnou I1. Klikneme na symbol Vystup a umístíme ho na výstup našeho součtového bloku a zadáme proměnnou Q. Dokončený projekt přeložíme, a spustíme Simulator.



Obr. č. 35: Okno pro programování v FBD.

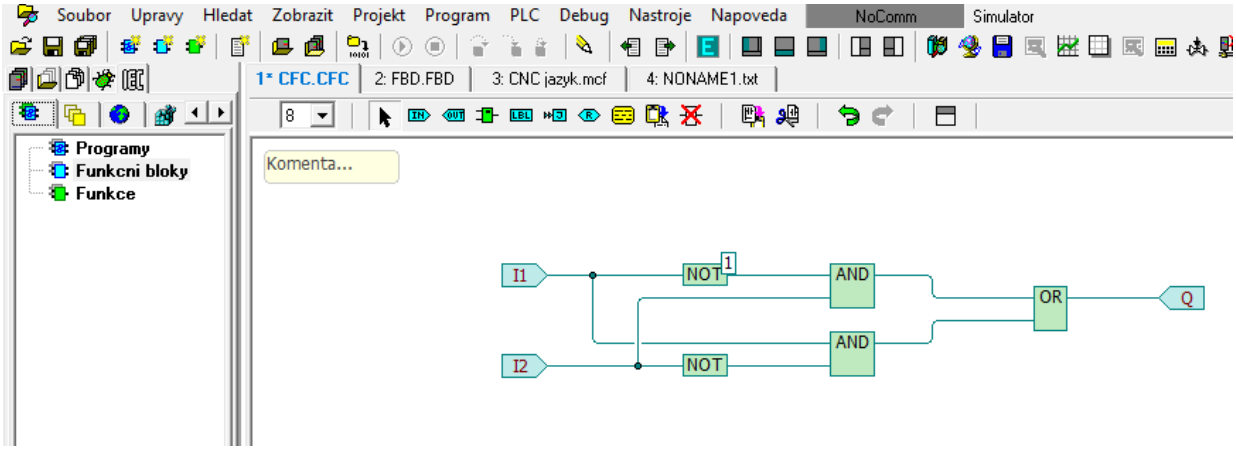
Příklad v jazyce CFC

Pro řešení našeho příkladu znovu vložíme nový projekt do skupiny. Poté založíme program, zvolíme programovací jazyk CFC a definujeme naše vstupy a výstupy.



Obr. č. 36: Okno pro programování v CFC.

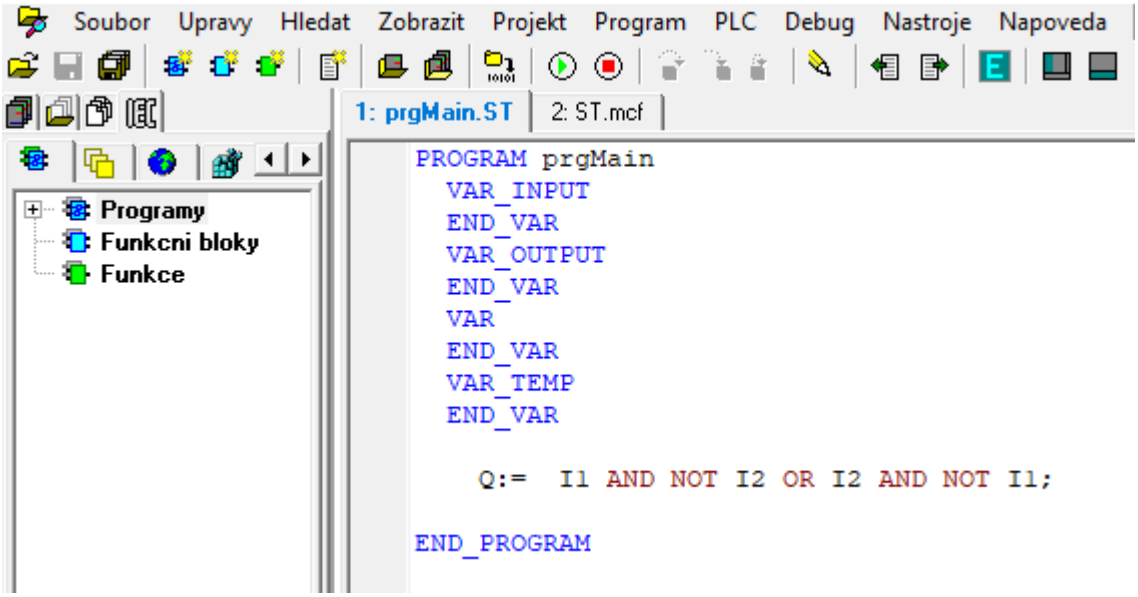
Zvolíme symbol Vlozit vstupni promennou , a umístíme ji na pracovní plochu. Zadáme jméno proměnné I1 a totéž uděláme pro druhou proměnnou I2. Dále zvolíme symbol Vlozit POU , a dvakrát zvolíme blok NOT (pro I1 a I2). Vložíme dva bloky logického součinu AND se dvěma vstupy, a další blok logického součtu OR se dvěma vstupy. Na konci vložíme vystup Q . Nyní připojíme všechny vstupy, bloky a výstup. Přeložíme program, a spustíme Simulator.



Obr. č. 37: Okno pro programování v CFC.

Příklad v jazyce ST

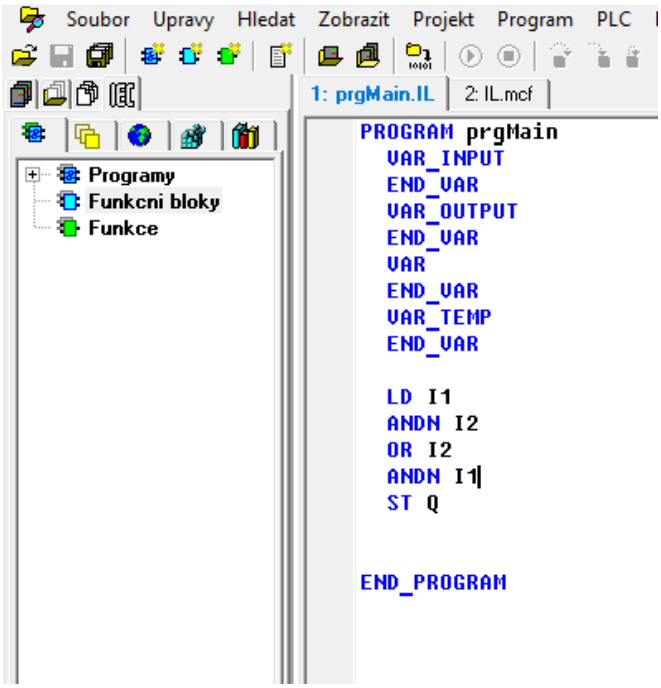
Opakováním akce popsaných výše založíme program, a zvolíme programovací jazyk ST. Náš příklad naprogramování v jazyce ST bude vypadat jako na obrázku č. 38. Projekt přeložíme, a spustíme Simulator.



Obr. č. 38: Okno pro programování v ST.

Příklad v jazyce IL

Opakováním akce popsaných výše založíme program, a zvolíme programovací jazyk IL. Náš příklad naprogramování v jazyce IL bude vypadat jako na obrázku č. 39. Projekt přeložíme, a spustíme Simulator.



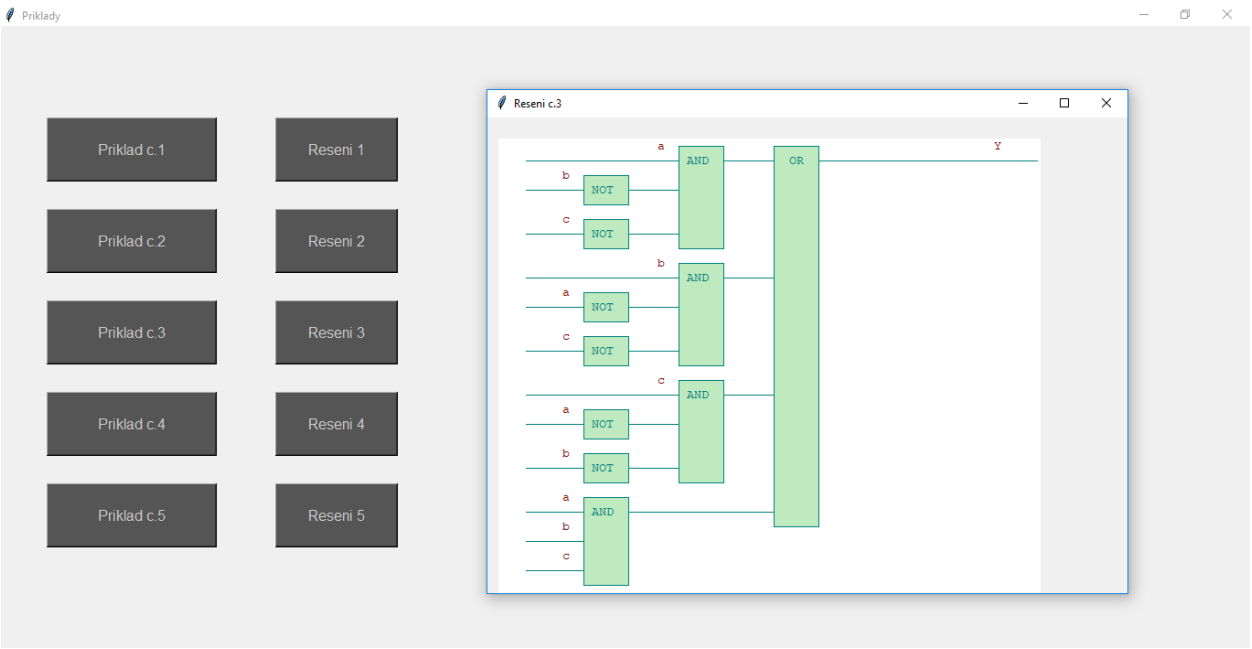
Obr. č. 39: Okno pro programování v IL.

3.4 Testy

Sekce Test je určena pro vlastní kontrolu znalostí studenta. Tato stránka obsahuje pět testů, kde v každém je maximálně deset testových otázek. Testy jsou zaměřeny na znalost normy IEC 61131-3, programovacích jazyků a základů programu Mosaic. Testy obsahují otázky na názvy nástrojů, jejich funkce a klávesové zkratky. Předpokladem testování je zkoušení znalostí studenta z předem nastudovaného materiálu týkajícího se problematiky popisované v této bakalářské práci.

3.5 Příklady

Sekce Příklady obsahuje příklady na kombinační logiku ve vývojovém systémemu Mosaic v různých programovacích jazycích. Většina příkladů je s řešením, ale existují i úkoly bez řešení.



Obr. č. 40: Sekce Příklady .

Otestovat příklady a testy je možné po instalaci Pythonu, a stažení kódu z přílohy do práce. Programovací jazyk Python je zdarma ke stažení na adrese https://www.python.org/downloads. Po staženi kódu musíme otevřít složku scratch_2.py, která se spustí pomocí Pythonu. Po otevření programu kliknutím na tlačítko „Příklady" se otevře stránka s příklady, kterou můžeme vidět na obrázku č. 40. Podobným způsobem můžeme se dostat do stránky „Testy", kde student může zjistit jakou má úroveň znalostí.

Pro lepší pochopení jsem vytvořil video na youtube, které může sloužt jako studijní materiál. V tomto videu krok po kroku názorně ukazuji jak naprogramovat řešení výše popisovaného příkladu ve vývojovém prostředí Mosaic. Příklad je řešen v jazyce LD. Video je k nalezení na stránce https://youtu.be/8cVnrijdIQ4 pod názvem „Programování v LD - program Mosaic".

5 Závěr

Ve této práci jsme se seznámili s normou IEC 61131-3 pro programování PLC a vývojovým prostředím Mosaic. V normě IEC 61131-3 byly uvedené základí pojmy a společné prvky programů pro PLC. Ve vývojovém prostředí Mosaic jsme provedli zakladní popis prostředí s uvedením hlavních nástrojů. Poté byly podle normy IEC 61131-3 rozebrány programovací jazyky. V každém programovacím jazyce byly uvedené jejich výhody i nevýhody. Bylo popsáno, který programovací jazyk je lepší použít pro vybrané úkoly. A nakonec bylo vysvětleno, který druh a jakým způsobem je jazyk implementován do praxe. Ve vývojovém prostředí Mosaic byly řešeny jednoduché příklady v každém programovacím jazyce s postupnim popisém jednotlivých kroků.

Dalé mnou byl vytvořen program pro studium a procvičení programovací logiky PLC ve vyvojovém prostředí Mosaic. Program byl napsán v Pythonu a má vzhled oken s tlačítky pod kterými se schovavaji následné funkce.

Program zahrnuje tři sekce: Teorie, Testové otázky a Příklady. Teorie umožňuje studentovi seznámit se s normou IEC 61131-3. Dále se naučit základní poznatky o každém z používaných programovacích jazyků pro PLC a naučit se základní práci ve vývojovém prostředi Mosaic.

Další částí jsou Testové otázky, zde student může otestovat své znalosti po nastudování teorie. Poslední částí jsou Příklady, kde jsou studentovi nabídnuty příklady z kombinační logiky pro, které musí dokázat vyřešit v Mosaicu. U některých příkladů má student možnost zkontrolovat své výsledky se správným řešením. Sekce Přiklady také má i příklady bez řešení pro samostatné procvičení studentovo logiky a znalostí.

Všechny úkoly jsou zaměřeny na kombinační logiku. Ta v praxi tvoří pouze malou součást programování PLC. Globálnější částí úkolů jsou sekvenční příklady, kterými se ale v této bakalářské práci nezabýváme. Tato práce spolu s mnou vytvořeným programem poskytuje vylepšení a zvětšení studijního materiálu.

Literatura

[1] ŠMEJKAL, L., ČERNÝ, J.: *Esperanto programátorů PLC: programování podle normy IEC/EN 601131-3* (souhrnné vydání částí stejnojmenného seriálu), Automa, Teco a. s., 2017. Elektronická verze je dostupná na www.tecoacademy.cz.

[2] KOHOUT, L.: *Norma pro řídicí systémy IEC 61 131.* Edumat, Kutna Hora, 2011. Dostupné na www.edumat.cz.

[3] *Programování PLC Tecomat podle normy IEC 61131-3 v prostředí Mosaic.* Teco a. s., Kolin, 2007. Dostupné na www.tecomat.com.

[4] *Začínáme v prostředí Mosaic*. Teco a. s., Kolin, 2010. Dostupné na www.tecomat.com.

[5] ŠMEJKAL, L. , MARTINASKOVA, M.: *PLC a automatizace 1. Základní pojmy, úvod do programování*. BEN, Technická literatura, Praha, 1999, ISBN 80-86056-58-9.

[6] LEWIS R.W. :*Programming industrial control systems using IEC 113-3,* Revised edition. - The Institution of Electrical Engineers, London, UK, 1998.

[7] HOFREITER, M.: *Základy automatického řízení – příklady*. ČVUT, FS, Praha, 2017, ISBN 978-80-01-05899-2.

[8] Elektroprumysl.cz. Python. [online]. [cit. 2019-05-15]. Dostupné z: https://www.elektroprumysl.cz/software/jaky-je-nejlepsi-programovaci-jazyk-pro-robotiku. ISBN 978-5-9596-0609-1.

[9] MINAEV, I. G., SAMOYLENKO, V. V.: *Programmable Logic Controller A practical guide for a novice engineer.* ARGUS, 2009, ISBN 978-5-9596-0609-1.

[10] PETROV, I.V.: *Programmable controllers. Standard languages and techniques of applied design.* SOLON-Press, 2015, ISBN 978-5-91359-151-7.

[11] BELOV, M.P.: *Technical means of automation and control.* SZTU, 2006.