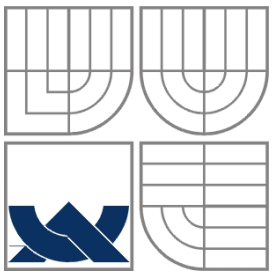


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ŘÍZENÍ MODELŮ PROCESŮ EDU-MOD PROGRAMOVATELNÝM AUTOMATEM ŘADY FX3U

EDU-MOD MODEL CONTROLL BY PROGRAMMABLE LOGIC CONTROLLER FX3U

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE **MARTIN KŘIVKA**
AUTHOR

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

BRNO 2010

ZADÁNÍ ZÁVĚREČNÉ PRÁCE

(na místo tohoto listu vložte originál a nebo kopii zadání Vaš práce)

LICENČNÍ SMLOUVA

(na místo tohoto listu vložte vyplněný a podepsaný list formuláře licenčního ujednání)

ABSTRAKT

Tato bakalářská práce vznikla jako podpora výuky předmětu Programovatelné automaty na Ústavu automatizace a informatiky fakulty Strojního inženýrství Vysokého učení technického.

Cílem této práce bylo seznámit se s modely soustav EDU-mod, které se využívají jako podpora výuky. Následně se seznámit s programovatelným automatem Mitsubishi FX3U-32M a jeho vývojovým prostředím GX IEC Developer 7.01. Poté v tomto prostředí navrhnout řízení pro zadané modely a ověřit funkčnost pomocí PLC.

ABSTRACT

This bachelor´s thesis was created as a support for Programmable logic controllers course at the Institute of Automation and Computer Science of the Faculty of Mechanical Engineering at Brno University of Technology.

The aim of this study is to get acquainted with the models of the EDU-mod systems serving as a teaching aid, and to learn about Mitsubishi PLC FX3U-32M and its development environment GX IEC Developer 7.01. Subsequently, the control for the specified models was designed in this environment and the functionality was verified via PLC.

KLÍČOVÁ SLOVA

PLC, EDU-mod, Mitsubishi – FX3U, GX IEC-Developer 7.01.

KEYWORDS

PLC, EDU-mod, Mitsubishi – FX3U, GX IEC-Developer 7.01.

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce Ing. Tomášovi Maradovi Ph.D.a také Ing. Karlu Jandovi za cené rady a připomínky.

Obsah:

1 ÚVOD

Soustava modelů EDU-mod je soubor modelů, které simulují různé technologické procesy. Tyto modely jsou určeny predevším k praktické výuce logických systémů pomocí programovatelných automatů (PLC), řídících počítačů nebo stavebnicemi logických obvodů.[1]

Programovatelný logický automat je v podstatě digitální počítač, který je určený pro automatizaci elektromechanických procesů. Mezi tyto procesy patří například ovládání výrobních linek, řízení strojů, popřípadě i některé jednodušší mechanismy jako jsou ovládání dveří, či ovládání kotle pro vytápění. Mezi hlavní přednosti programovatelných automatů patří to, že jsou určeny do průmyslových oblastí, proto jsou vyráběny tak, aby byly pokud možno co nejvíce odolné vůči vnějším vlivům jako jsou vlhkost, vibrace, prach, chlad, výkyvy napětí či magnetické pole.

Mezi další vlastnosti patří to, že automaty vykonávají program, který je v nich nahraný cyklicky a v reálném čase. V reálném čase z toho důvodu, aby nedocházelo k nechtěným problémům způsobených zpožděnou reakcí na změnu na vstupech.

Automat je schopný pracovat s velkým počtem vstupů a výstupů. Vstupy bývají většinou připojeny na různé snímače, hodnoty z těchto vstupů automat následně vyhodnotí, podle toho následně posílá signály na výstupy na které jsou většinou připojeny akční členy (motory, elektromagnetická relé, elektromagnety či hydraulické nebo pneumatikcé písty atd.). Každý automat má určitý počet integrovaných vstupů a výstupů, popřípadě může být jejich počet rozšířen pomocí přídavných modulů. [4]

2 MODELY SOUSTAV EDU-MOD

V této kapitole se seznámíme trochu blíže s modely procesů EDU-mod. Jak bylo napsáno v úvodu, jsou tyto modely určeny především k praktické výuce logických systémů a hlavně k řízení těchto modelů pomocí PLC (programovatelného automatu). Mohou být ale také řízeny pomocí řídících počítačů nebo také stavebnicemi logických obvodů. V laboratoři A1/731a jsou ze soustavy těchto modelů k dispozici modely křižovatky, hydraulické posuvové jednotky, mísicí jednotky a automatické pračky. Jedná se o modely EDU- mod řady 24V tzn., že logické signály s úrovní 24V ss umožňují univerzální použití pro libovolný typ PLC systému. Tato práce je zaměřena na navržení řízení pro dva tyto modely a to hydraulickou posuvovou jednotku a mísicí jednotku. [1]

2.1 Hydraulická posuvová jednotka

Tento model simuluje pohyb suportu, tento pohyb je signalizován 10 led diodami, z toho čtyři slouží jako snímače (K1, K2, K3, K4). Jsou zde také 3 výstupní signální bity a to EM1 (pro pohyb vpravo), EM2 (pro pohyb vlevo) a EM3 (pro ovládání rychlosti posuvu EM3=1 zpomalený pracovní posuv).

Dále je zde červená dioda ERR, která signalizuje chybové stavy (2 stavy). První chybový stav nastane, když suport přejede jeden z krajních snímačů (K1, K4), tento stav je signalizován rozsvícením červené diody ERR a rozsvícením diod u snímačů (K1 až K4). Druhý chybový stav nastane, pokud jsou současně spuštěny elektromotory EM1 a EM2, tento stav je signalizován blikáním červené diody ERR.

Součástí modelu je také tlačítko RESET, po jehož stlačení se automaticky nastaví inicializační stav, pozice na snímači K1. Tento stav se také nastaví po zapnutí napětí.[2]

Obr. 1: Hydraulická posuvová jednotka [2].

2.2 Mísicí jednotka

Jedná se o model s vlastní inteligencí, který simuluje funkci technologie složené ze čtyř nádrží (tří plnících tanků a jedné mísicí nádoby). Tato jednotka má 8 vstupů, kterými jsou snímače hladiny v tancích H1 až H8 (H3, H5 a H8 – minimální množství hladiny v tancích, H2 a H7 – tanky 1 a 3 jsou z poloviny zaplněny, H1, H4 a H6 – tanky jsou zcela naplněny).

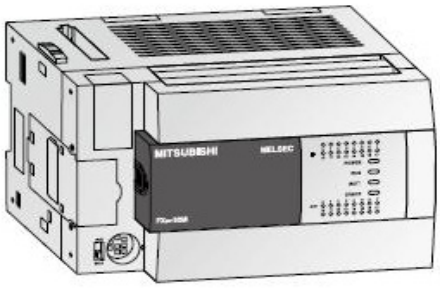
Dále je zde 6 výstupů a to SV1 až SV5 a MIX. Po sepnutí ventilů SV1, SV2 a SV3 se začnou plnit příslušné tanky. SV4 slouží jako napouštěcí ventil pro mísicí nádobu a zároveň vypouštěcí pro tanky 1,2,3. Výstup MIX spouští mixér, který promíchává obsahy jednotlivých tanků. Jako poslední je ventil SV5, tento ventil vypouští obsah mísicí nádoby. Přetečení jakékoliv nádoby je bráno jako chybový stav a je signalizováno rozsvícením červené LED diody ERR. Po stisknutí tlačítka reset se model vrátí do inicializačního stavu, kdy jsou všechny nádoby prázdné a rozbliká se červená LED dioda ERR. Ta zhasne až po vybuzení signálu na některém z ventilů.[3]

Obr. 2: Mísicí jednotka [3]

3 MITSUBISHI FX3U-32M

Údaje z této kapitoly je převzaty z materiálů firmy Mitsubishi.[5]

FX3U je řada kompaktních automatů od firmy Mitsubishi electric. Tato řada automatů patří již do třetí generace kompaktních PLC, které firma Mitsubishi distribuuje. Byly vyvinuty pro mezinárodní trh a jejich význačným rysem je druhý systém "adaptérové sběrnice", která doplňuje stávající systém sběrnic a používá se pro rozšíření, speciální funkce a síťové moduly. K této nové adaptérové sběrnici může být připojeno až deset dodatečných modulů.



Obr. 3: Programovatelný automat Mitsubishi FX3U-32M [5]

3.1 Specifikace

Programovatelný automat Mitsubishi FX3U-32M má následující parametry:

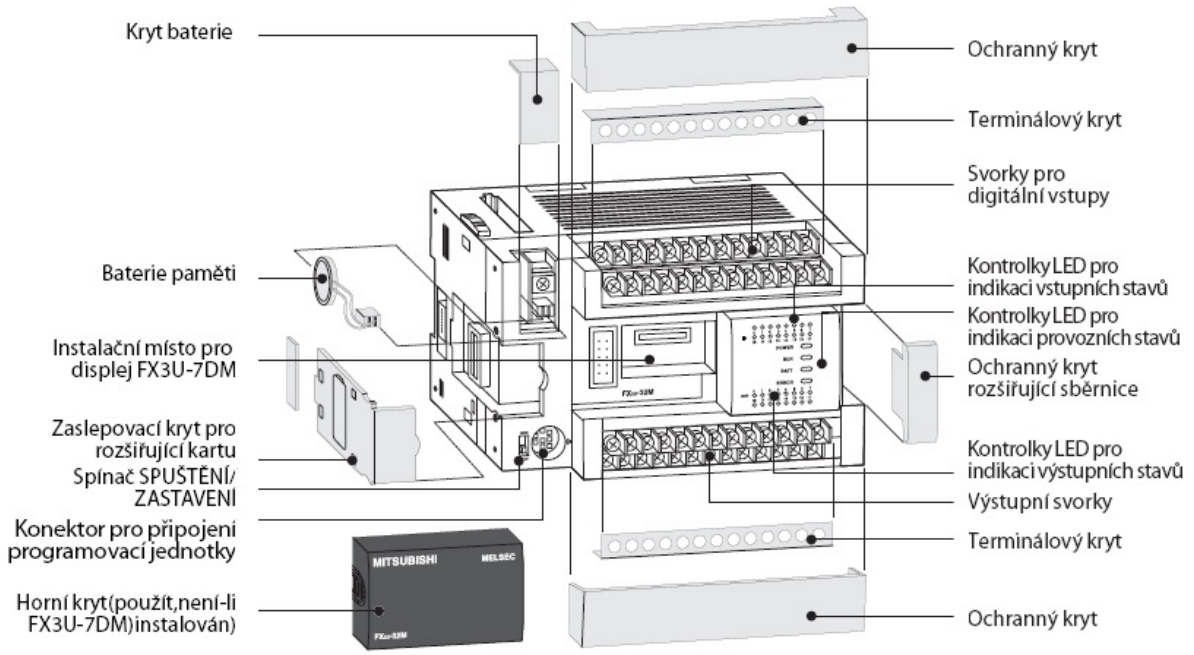
I/O	Vstupy (I)	Výstupy (O)	Zdroj napájení	Typ výstupu
32	16	16	24V DC	Tranzistor

Tab. 1: Parametry Mitsubishi FX3U-32M

Řada FX3U má ještě následující parametry:

- Maximální počet I/O obvodů: 128
- Možnosti rozšiřování (max. možný počet I/O): 384
- Programová paměť (kroky): 64000
- Čas cyklu na logický příkaz (μs): 0,065
- Počet příkazů (standard/ krokově/ speciální funkce): 27/ 2/ 209
- Maximální počet speciálních funkčních modulů: 8 vpravo, 10 vlevo
- Komunikace: port RS-485, port RS-232, Ethernet (TCP/UDP), Profibus-DP, CCLink, DeviceNet, CANopen, AS-interface
- Rozměry: 150 x 90 x 86 [mm]

3.2 Nákres základní jednotky MELSEC FX3U



Obr. 4: Nákres základní jednotky MELSEC FX3U [5]

- Konektor pro připojení programovací jednotky: může být použit pro připojení příruční programovací jednotky FX-20P-E, PC nebo notebooku s programovacím softwarem

- EEPROM: přepisovatelná paměť, do které lze programovacím softwarem zapisovat a vyčítat PLC program

- Rozšiřovací sběrnice: zde mohou být připojeny oba přídavné I/O rozšiřovací moduly nebo moduly speciálních funkcí, které přidávají PLC další možnosti

- Digitální vstupy: používají se pro řídící signály přicházející z připojených spínačů, tlačítek a senzorů, tyto vstupy čtou hodnoty ON (zdroj signálu zapnut) a OFF (zdroj signálu vypnut)

- Digitální výstupy: k těmto výstupům můžete připojit řadu různých ovládacích členů a dalších zařízení v závislosti na povaze aplikace a typu jejího výstupu

- Kontrolky LED pro indikaci stavu vstupů: tyto LED ukazují, které vstupy jsou v dané době připojeny ke zdroji signálu, resp. definovanému napětí, při vstupu signálu se rozsvítí odpovídající LED a indikuje, že na vstupu je stav ON

- Kontrolky LED pro indikaci stavu výstupů: tyto LED ukazují stav ON/OFF na digitálních výstupech, tyto výstupy mohou spínat řadu různých napětí a proudů v závislosti na typu modelu a výstupu

- Kontrolky LED pro indikaci provozních stavů: kontrolky LED RUN, POWER a ERROR ukazují stávající stav automatu, POWER ukazuje zapnutý zdroj, RUN svítí při provádění programu a ERROR svítí při chybě nebo poruše

- Baterie paměti: baterie chrání obsah RAM paměti PLC v případě, že dojde k selhání zdroje, chrání nastavené rozsahy časovačů, čítačů a relé, dále poskytuje energii integrovaným hodinám v případě vypnutí zdroje napájení pro PLC

- Spínač RUN/STOP: PLC má dva operační módy, RUN a STOP, spínač RUN/STOP umožňuje manuální přepnutí mezi těmito dvěma módy, v módu RUN PLC provádí program uložený ve své paměti, v módu STOP je provádění programu zastaveno a je možné automat programovat

4 GX IEC DEVELOPER 7.01

PLC (Programmable Logic Controller) neboli programovatelný logický automat. Mezi tyto automaty patří i model Mitsubishi FX3U-32M, pro programování tohoto automatu se používá software GX Developer. V této práci byla použita verze GX IEC Developer 7.01. Tento software, jak je obsaženo v jeho názvu je programovací a dokumentační software dle normy IEC 61131-3. Což je mezinárodní standard pro programování PLC systémů definovaný mezinárodní elektrotechnickou komisí (IEC).

4.1 Hlavní panel nástrojů

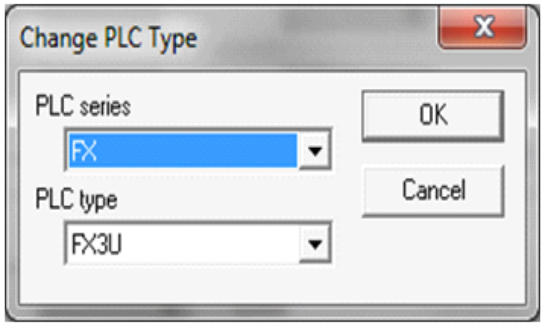
V následující kapitole budou uvedeny prvky, které je možné najít v panelu nástrojů. U těch nejdůležitějších bude uveden význam a použití.

Otevřít Projekt	**Uložit Projekt**	**Tisk**	**Náhled Tisku**	**Vyjmout**	**Kopírovat**
					
Vložit	**Zpět**	**Vpřed**	**Vlastnosti**	**Procházet**	**Změna typu PLC**
					
Check	**Build**	**ReBuild All**	**Open MXChange database**	**Open projekt from MXChange**	**Enable/disable monitoring mode**
					
Online Change mode	**Download Project**	**Upload project**	**New POU**	**New DUT**	**New TASK**
					

Tab. 2: Prvky hlavního panelu nástrojů

Změna typu PLC: toto tlačítko slouží ke změně typu používaného PLC, typ PLC se dá také nastavit hned při vytváření nového projektu. Nastavuje se série do jaké používaný automat patří a následně přímo typ použitého automatu. Změna se dá také provést v záložce:

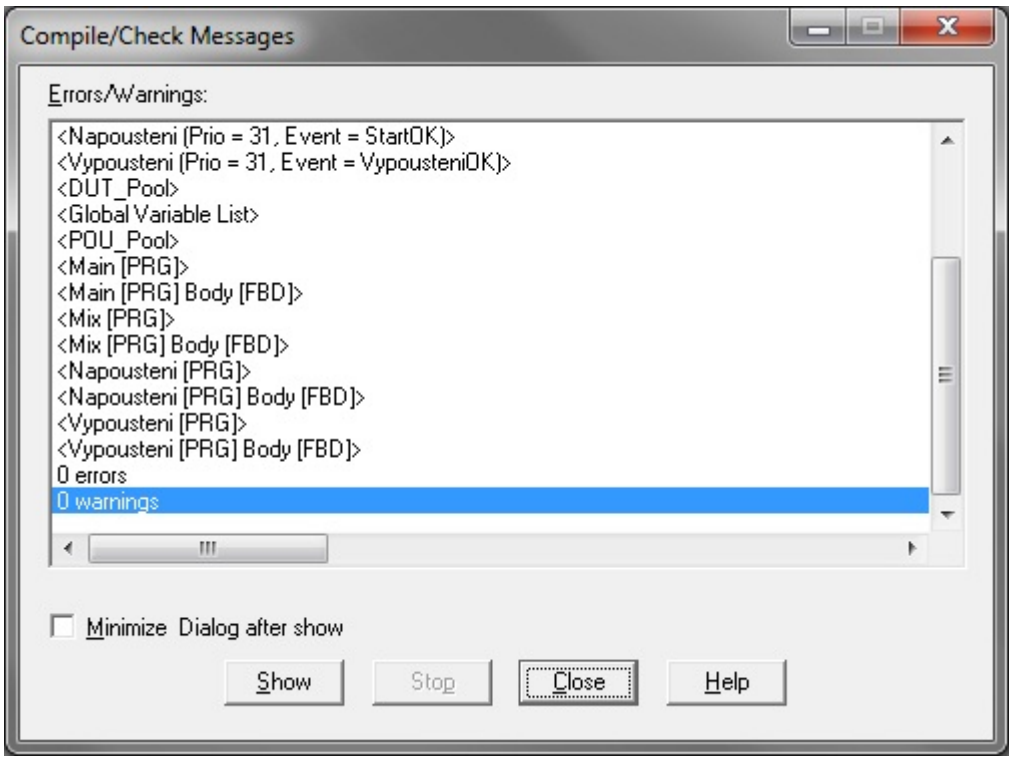
Project / Change PLC type



Obr. 5: Okno Change PLC type

Check: slouží ke kontrole syntaxe, tato kontrola se může použít jak na jednotlivé objekty projektu (Task, POU…), tak na celý projekt. Lze jej použít i pro kontrolu tabulek proměnných (Global vars, Header). Výsledky se zobrazí v okně Compile/Check messages. Tato kontrola lze také spustit v záložce:

Object / Check



Obr. 6: Okno Compile/Check messages

Build: slouží k převedení programovacího jazyka do strojového kódu, tato procedura musí být použita, aby bylo možné nahrát výsledný program do automatu. Používá se u objektů, ve kterých došlo k nějaké změně, nebo u prvků projektu, které jsou s těmito objekty nějak propojeny. Výsledky této operace se rovněž zobrazí v okně Compile/Check messages. Operaci lze také spustit v záložce:

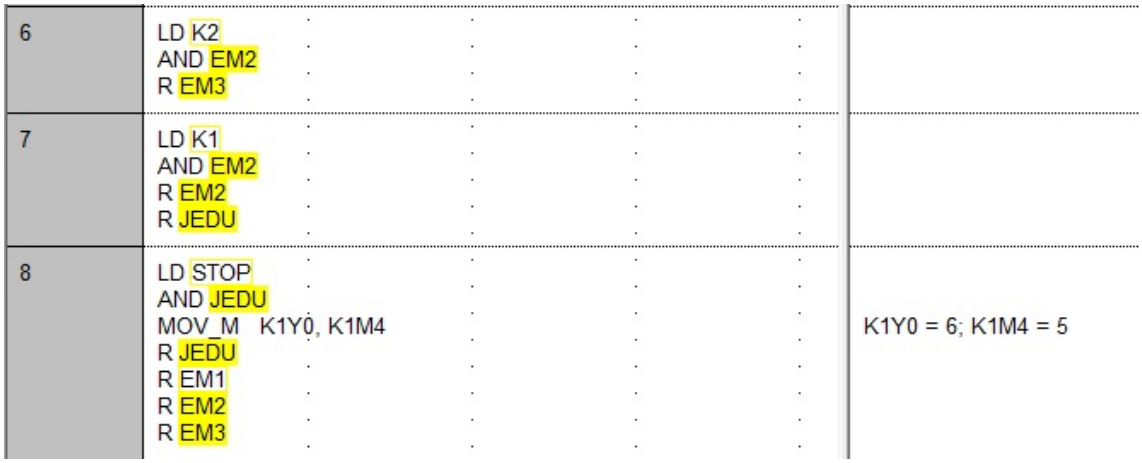
Project / Build

Rebuild All: znovu převede celý projekt napsaný v některém z programovacích jazyků do strojového kódu. Tato procedura je nutná, aby bylo možné nahrát výsledný program do automatu. I výsledky této operace se zobrazí v okně Compile/Check messages. Operaci lze také spustit v záložce:

Project / Rebuild All

Enable/Disable monitoring mode: slouží k zapnutí/vypnutí monitorovacího módu, je to mód, při kterém můžeme pozorovat, jak se při běhu programu mění jednotlivé vstupy, výstupy, paměťové bity atd. Monitorovací mód lze také spustit ze záložky:

Online / Monitoring mode



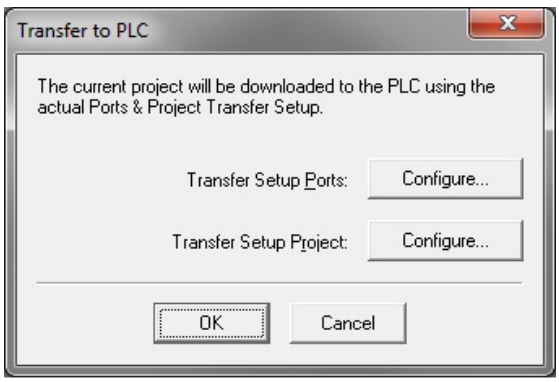
Obr. 7: Ukázka spuštěného monitorovacího módu

Download project: zkopíruje zkontrolovaný a převedený projekt z počítače do programovatelného automatu. Při tomto kopírování musí být automat v režimu STOP. Download lze také spustit v záložce:

Project / Transfer / Download to PLC

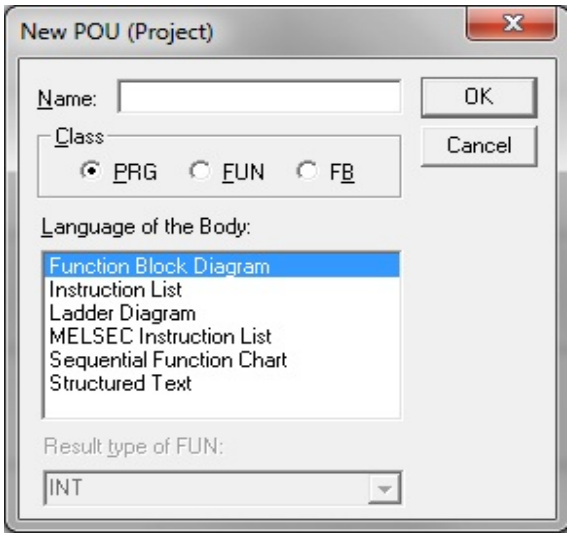
Upload project: zkopíruje do počítače program, který je nahraný v paměti programovatelného automatu. I pro toto kopírování musí být automat v režimu STOP. Upload lze také spustit v záložce:

Project / Transfer / Upload from PLC



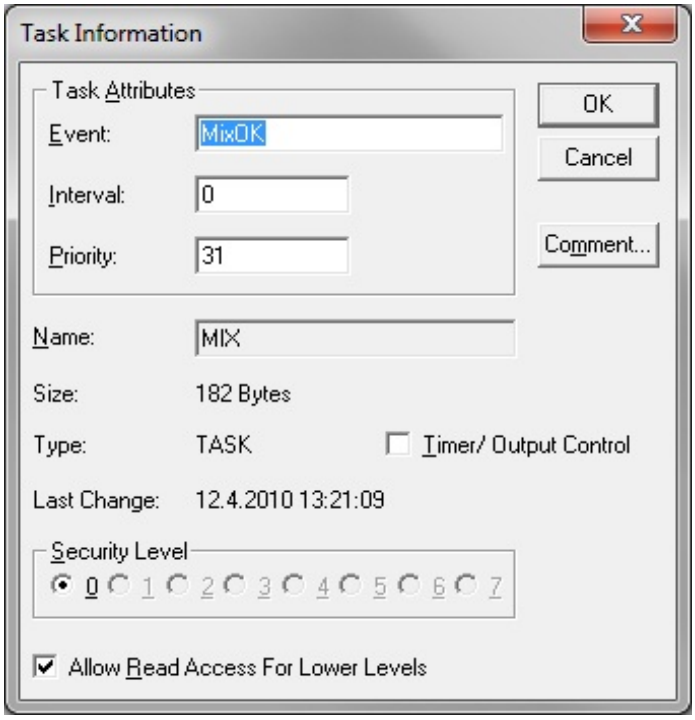
Obr. 8: Transfer programu

New POU: slouží k vytvoření nové POU (program organisation unit) programové jednotky. Nastavuje se název, dále třída (program, funkce, funkční blok) a poté jazyk jaký bude obsahovat tělo nové POU.



Obr. 9: Okno New POU

New TASK: slouží pro vytvoření nové úlohy (TASK), je to řídící prvek, který umožňuje volání POU. Nastavení Tasku se provádí kliknutím pravým tlačítkem myši na název Tasku a vybráním Properties. Zde lze následně nastavit událost EVENT, tato událost spustí POU, která je v Tasku nastavena. POU se nastavuje dvojklikem levého tlačítka myši na název Tasku. Dále lze nastavit Interval a Prioritu, která rozhoduje o tom, která POU bude vykonávána dřív.



Obr. 10: Okno pro vytvoření nového TASKu

4.2 Změna režimu PLC

Jak už bylo napsáno v kapitole 3.2 má automat dva operační módy a to RUN a STOP. Mezi těmito módy se lze přepínat dvěma způsoby a to buď mechanicky přímo na automatu tlačítkem a nebo softwarově v programu GX IEC Developer 7.01. A to v záložce:

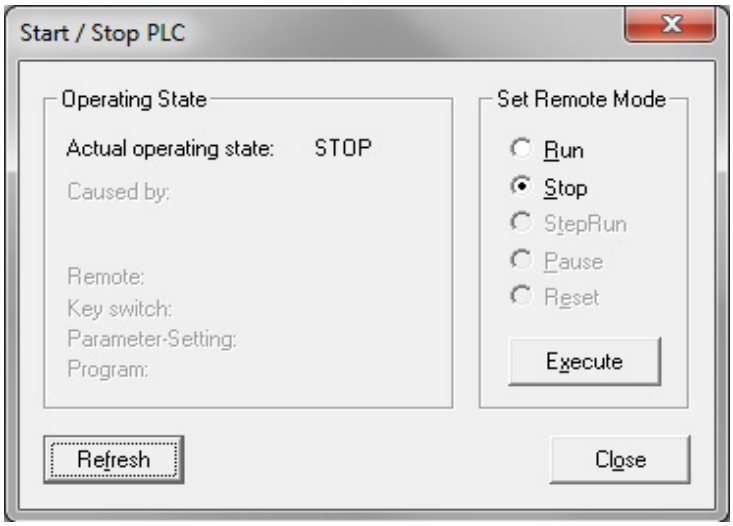
Online / Start/Stop PLC

REFRESH: tlačítko, které slouží ke zjištění aktuálního stavu PLC.

RUN: zaškrtávací políčko, které slouží k přepnutí automatu do režimu RUN (spustit). Je nutné, aby hardwarové tlačítko na automatu bylo v režimu RUN.

STOP: zaškrtávací políčko, které slouží k přepnutí automatu do režimu STOP (zastavit). Je nutné, aby hardwarové tlačítko na automatu bylo v režimu RUN.

Execute: slouží k potvrzení vybrané volby a následnému odeslání změn do automatu. Je nutné, aby hardwarové tlačítko na automatu bylo v režimu RUN. A fungovalo spojení mezi počítačem a automatem.



Obr. 11: Okno pro změnu režimu automatu

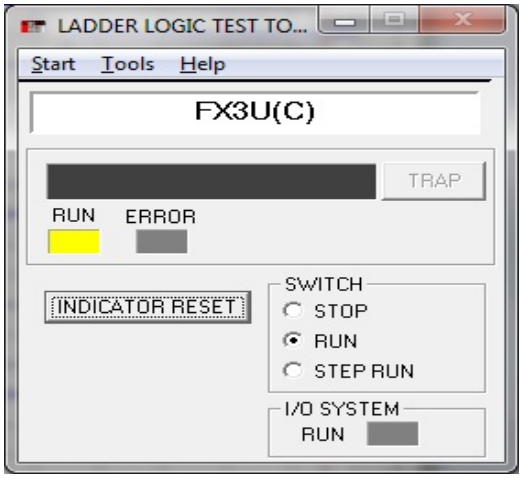
4.3 GX Simulator

GX Simulator je virtuální programovatelný automat, který je určen především k testování napsaného programu bez nutnosti připojení ke skutečnému automatu. Tímto se také eliminuje případné poškození tohoto automatu.

Můžeme si tak tedy snadno ověřit, zda nám program správně funguje, protože v GX Simulátoru si můžeme nasimulovat jednotlivé vstupy, výstupy, paměťové bity atd. Spouštění GX Simulatoru se provádí v záložce:

Online / GX Simulator

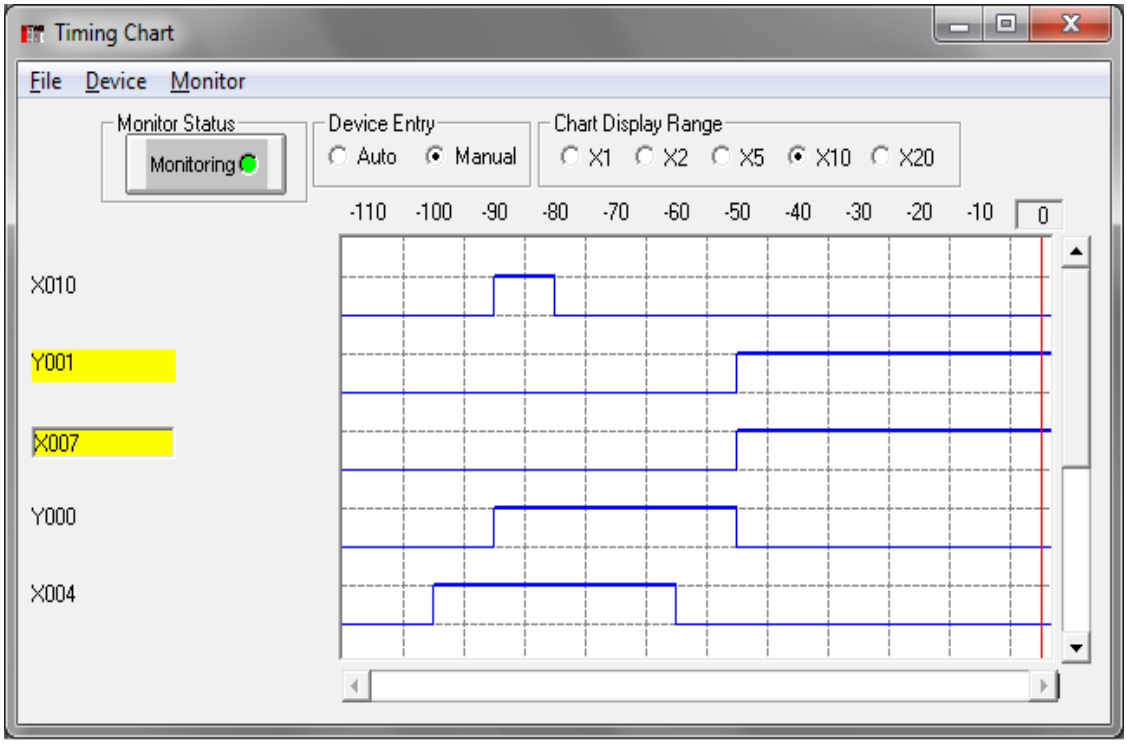
Ovšem pro jeho správné spuštění musí být GX Simulator nainstalován, GX IEC Developer se nesmí nacházet ve stavu online a celý projekt musí být řádně zkompilován. Po splnění těchto podmínek a po spuštění se objeví dialogové okno GX Simulatoru:



Obr. 12: Okno GX Simulatoru

V tomto dialogovém okně můžeme vidět typ simulovaného automatu, v tomto případě FX3U, následně zde můžeme vidět v jakém stavu se automat nachází, popřípadě tyto stavy měnit. Ovšem jednou z nejdůležitější částí GX Simulatoru je Timing Chart display, ten se spouští v záložce:

Start / Monitoring function / Timing chart display



Obr. 13: Okno Timing chart display

V tomto okně můžeme pozorovat a měnit hodnoty jednotlivých vstupních proměnných a sledovat jejich závislost na čase. Proměnné, které jsou označené žlutou barvou, obsahují hodnotu logická 1 a nezvýrazněné proměnné obsahují hodnotu logická 0. Změnit hodnotu jednotlivých proměnných lze provést dvojklikem levého tlačítka myši na danou proměnnou.

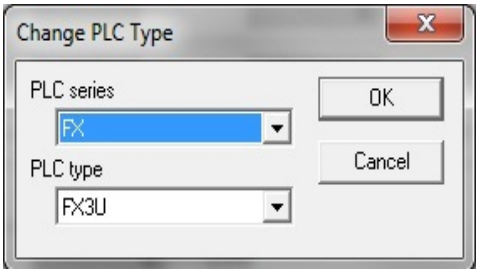
Při takto spuštěném GX simulátoru nelze provádět žádné programové změny.

4.4 Nový Projekt

Nový projekt můžeme založit v záložce:

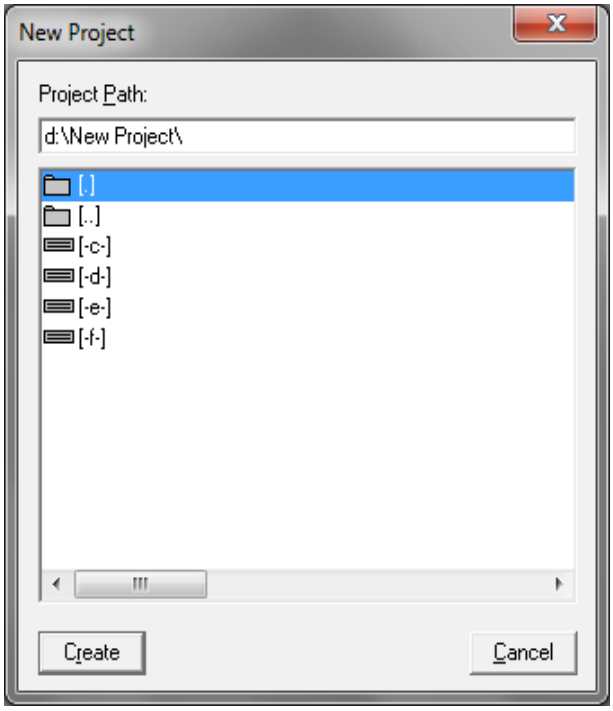
Project / New

Po vybrání této možnosti se nám zobrazí okno, ve kterém vybíráme sérii do které automat, jenž budeme programovat patří. Dále je ještě nutné vybrat přesný typ automatu.



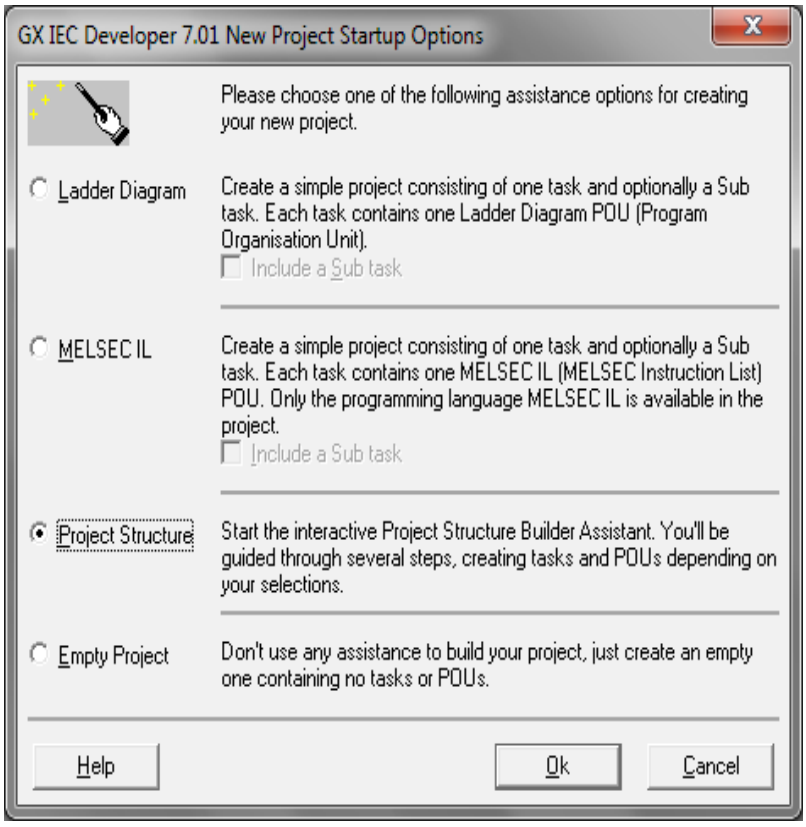
Obr. 14: Okno Change PLC Type

Jako další se nám zobrazí okno, ve kterém vybíráme kam chceme náš nově vytvořený projekt uložit a pod jakým názvem.



Obr. 15: Okno New Project

Po takto založeném projektu se nám zobrazí další okno a to okno New Project Startup Options, ve kterém je nám umožněno vybrat si jakou bude mít náš nový projekt strukturu. Jsou zde 4 možnosti a to:



Obr. 16: Okno New Project Startup Options

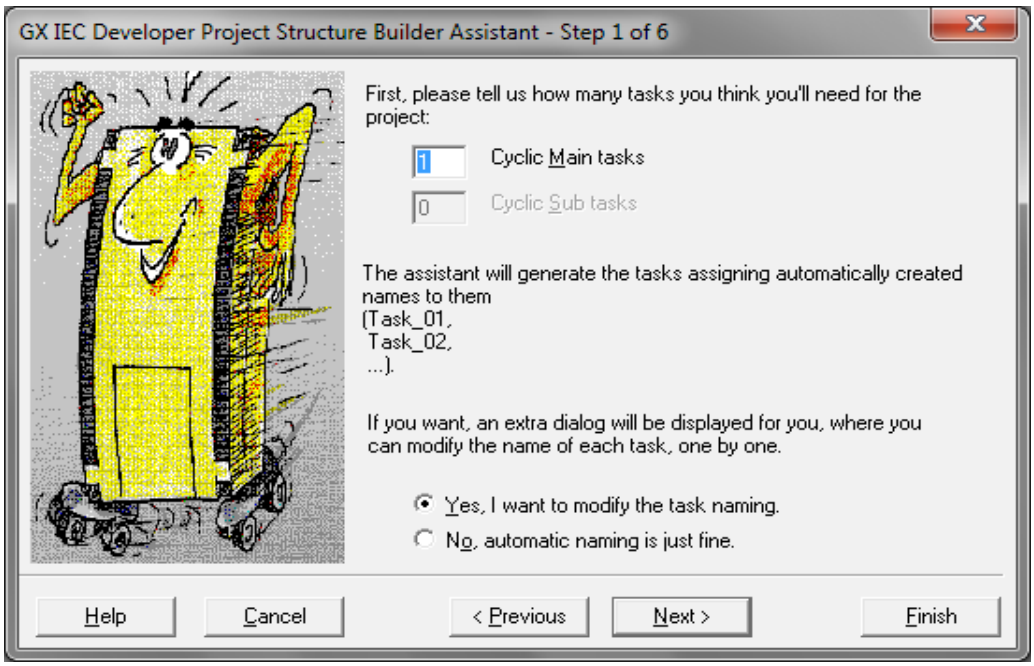
- Ladder Diagram: po vybrání této možnosti nám asistent vytvoří projekt obsahující jeden Task a volitelně ještě jeden SubTask. Každý tento Task obsahuje jednu POU pro programování v jazyku kontaktních schémat (Ladder Diagram).

- MELSEC IL:při vybrání této možnosti nám asistent opět vytvoří projekt obsahující jeden Task a volitelně jeden SubTask. Každý Task obsahuje jednu POU pro programování v jazyku Melsec instrukční list (Melsec IL).

- Empty Project: tato možnost slouží k vytvoření projektu bez použití asistenta, je vytvořen prázdný projekt, který neobsahuje žádné Tasky ani POU a uživatel si je musí vytvořit a definovat sám.

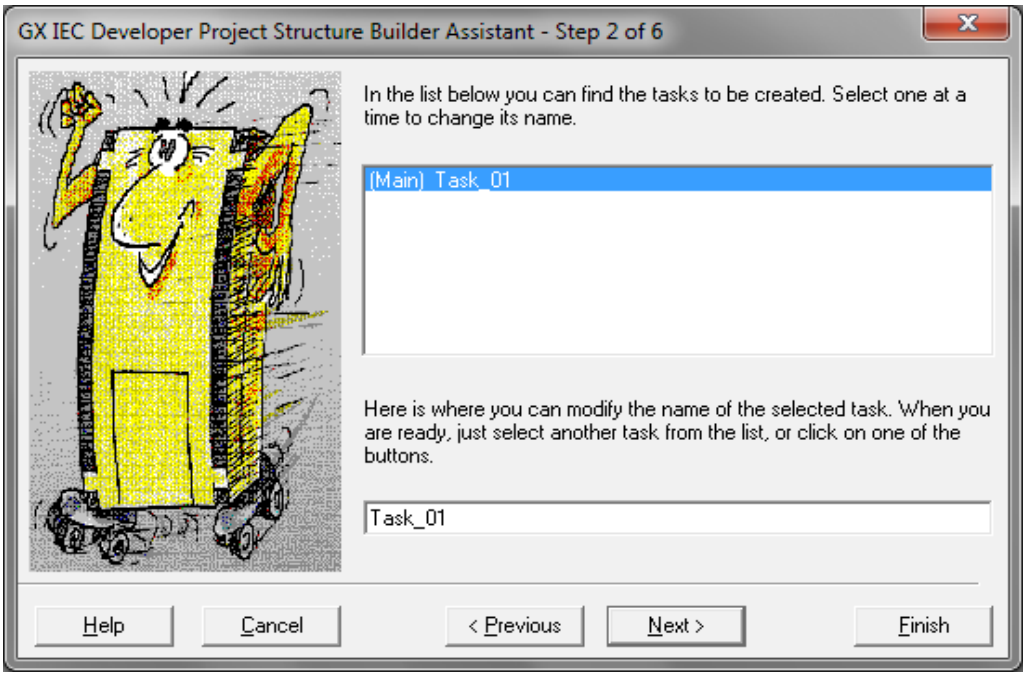
- Project Structure: při vybrání této možnosti se spustí asistent, kterým budeme provedeni dalšími kroky vytváření nového projektu.

Po vybrání možnosti Project Structure se zobrazí 1. Ze 6 oken asistenta pro vytvoření nového projektu. V tomto okně si můžeme zadat počet Tasků podle toho, kolik jich v projektu budeme potřebovat a také, jestli si budeme zadávat jejich názvy a nebo zda postačí automatické pojmenovávání.



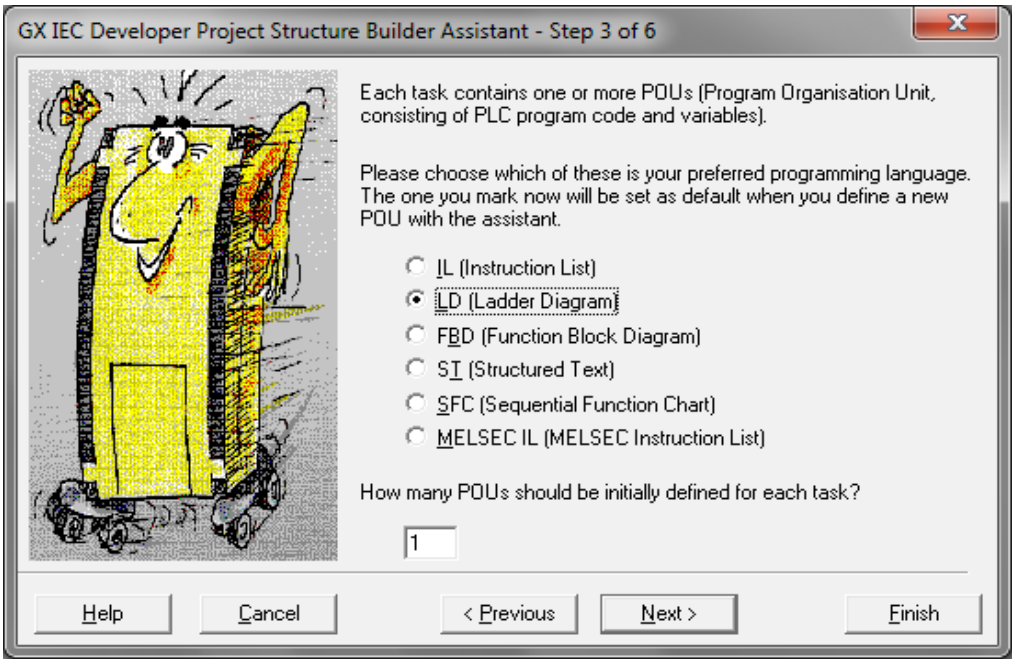
Obr. 17: GX IEC Developer Project Structure Builder assistant – Step 1 of 6

Pokud jsme vybrali možnost, že budeme názvy Tasků zadávat sami, je tak možno učinit v 2. kroku vytváření projektu.



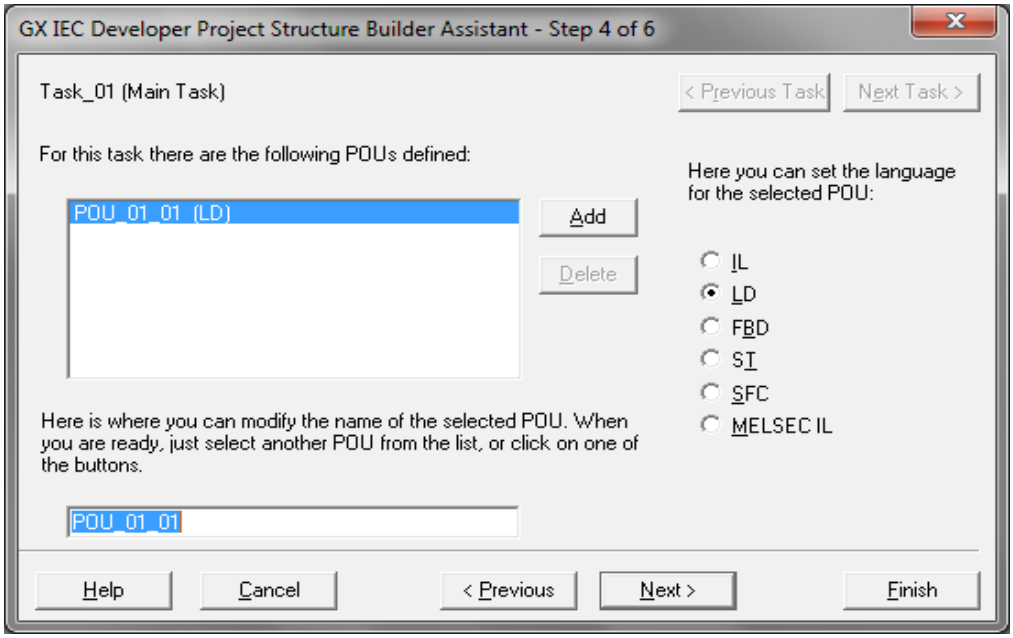
Obr. 18: GX IEC Developer Project Structure Builder assistant – Step 2 of 6

V dalším kroku si vybíráme který z programovacích jazyků budeme v novém projektu upřednostňovat a to z 5 jazyků normy IEC 61131-3 (Instruction List, Ladder Diagram, Function Block Diagram, Structured Text, Sequential Function Chart), jako další je zde ještě programovací jazyk, který je určen pro programování automatů firmy Mitsubishi řady FX a to Melsec IL. Dále je zde možnost nastavit si kolik POU by měl při výchozím stavu obsahovat každý Task.



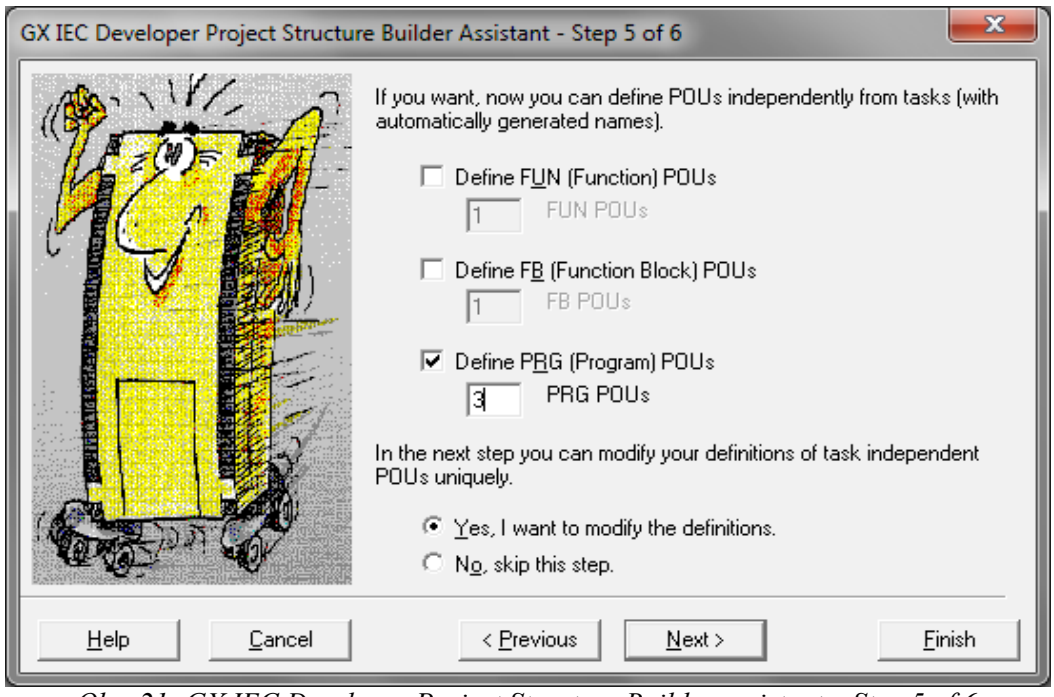
Obr. 19: GX IEC Developer Project Structure Builder assistant – Step 3 of 6

Ve 4. kroku je právě možnost pro POU jednotlivých Tasků změnit název, případně přidat další POU a pro každou POU nastavit programovací jazyk, který tato POU bude preferovat.



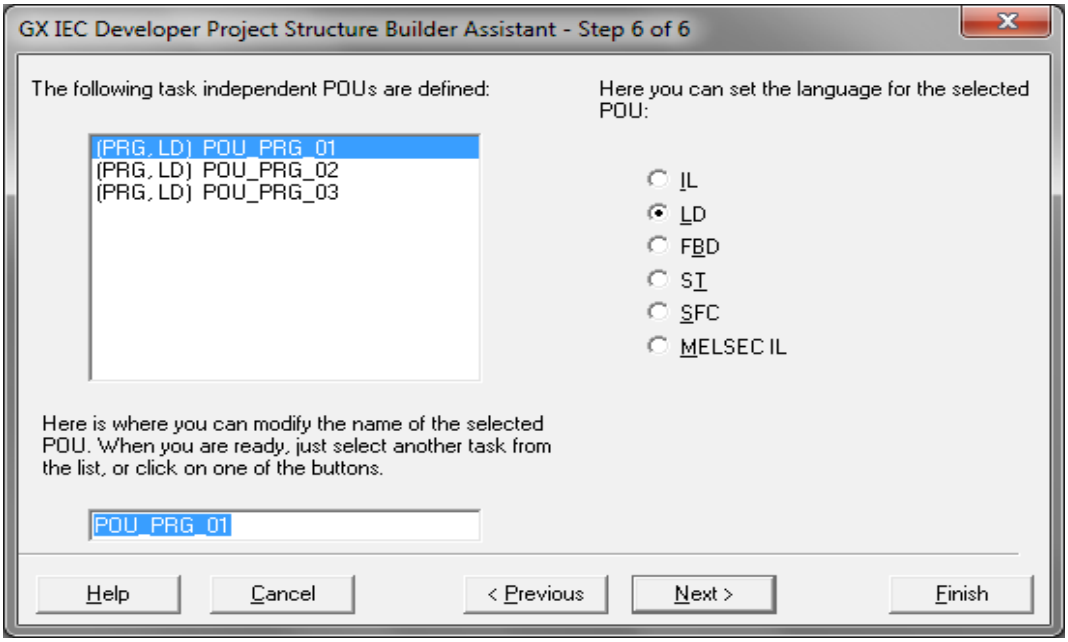
Obr. 20: GX IEC Developer Project Structure Builder assistant – Step 4 of 6

V předposledním kroku máme ještě možnost definovat další POU, které ovšem nebudou závislé na v předchozích krocích definovaných Tascích. U těchto POU také můžeme vybrat zda jde o FUN (funkci), FB (funkční blok) nebo PRG (program), dále se ještě nastavuje počet kolik kterých POU a jestli chceme upravit jejich definice.



Obr. 21: GX IEC Developer Project Structure Builder assistant – Step 5 of 6

V posledním kroku už pouze definujeme POU z předchozího kroku a nastavujeme programovací jazyk který bude v jednotlivých POU upřednostňovaný a můžeme upravit názvy těchto POU. Celý průvodce končí kliknutím na tlačítko Finish.



Obr. 22: GX IEC Developer Project Structure Builder assistant – Step 6 of 6

4.5 Programovací jazyky

- Instruction List (IL): Jazyk instrukcí, patří mezi textové programovací jazyky. Je podobný assembleru.
- Ladder Diagram (LD): Jazyk reléových schémat, patří mezi grafické programovací jazyky
- Function Block Diagram (FBD): Diagram funkčních bloků, patří mezi grafické programovací jazyky
- Structured Text (ST): Strukturovaný text, patří mezi textové programovací jazyky
- Sequential Function Chart (SFC): Patří mezi grafické programovací jazyky.
- Melsec IL: Melsec jazyk instrukcí, patří mezi textové programovací jazyky, je podobný jazyku IL a je určen pro programování PLC řady FX. Nejsou zde přípustné Instrukce normy IEC.

4.6 Struktura Projektu

Každý projekt založený v GX IEC Developeru obsahuje následující části.

4.6.1 Task Pool

Obsahuje jednotlivé tasky, neboli řídící prvky, které jsou schopny volat POU. Nový Task se dá vytvořit tak jak bylo popsáno v kapitole 4.1 nebo také kliknutím pravého tlačítka myši na Task pool a vybrat New Task. Poté se zadá název tohoto nového tasku.

4.6.2 Global Vars (tabulka globálních proměnných)

Tato tabulka obsahuje všechny vstupy, výstupy, proměnné, konstanty a funkční bloky, které jsou používány během celého programu. Důležitou vlastností je, že pro tyto proměnné lze nastavit symbolické, uživatelem definované označení, pod kterým bude možné tyto proměnné v celém projektu používat. Toho lze výhodně využít například u konstant, pokud dojde k nějaké změně, třeba v datovém typu, stačí provést změnu pouze v tabulce Global Vars a změna se provede v celém projektu najednou. Je ovšem nutné v této tabulce definovat funkční bloky, nedefinované funkční bloky by nebylo možné použít.

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial	Comment	Remark
0	VAR_GLOBAL	H1	X7	%IX7	BOOL	...	FALSE		
1	VAR_GLOBAL	H2	X6	%IX6	BOOL	...	FALSE		
2	VAR_GLOBAL	H3	X5	%IX5	BOOL	...	FALSE		
3	VAR_GLOBAL	H4	X4	%IX4	BOOL	...	FALSE		
4	VAR_GLOBAL	H5	X3	%IX3	BOOL	...	FALSE		
5	VAR_GLOBAL	H6	X0	%IX0	BOOL	...	FALSE		

Tab. 3: Tabulka globálních proměnných (Global Vars)

Class – přepínač globální proměnná/konstanta

Identifier – Identifikátor (symbolické, uživatelem definované označení)

MIT-Addr. – MELSEC systémová adresa

IEC-Addr. – IEC normovaná systémová adresa

Type – Datový typ proměnné

Initial – Výchozí hodnota

Comment - Komentář

Remark - Poznámka

4.6.3 POU pool

Tato část projektu obsahuje jednotlivé programové moduly POU (program organisation unit). Každá POU obsahuje jak tabulku lokálních proměnných (Header), tak i vlastní tělo programu Body.

Header (Hlavička) – obsahuje lokální proměnné, tedy proměnné, které jsou definovány pouze pro použití v dané POU. Z toho plyne že jednotlivé POU mohou používat proměnné se stejnými názvy, ale například s jinými hodnotami. Tohoto nelze využít u Global Vars, protože v Global Vars a Header nemůžou být stejně označené proměnné.

	Class	Identifier	Type		Initial	Comment
0	VAR	Timer	TON	...		
1	VAR_CONSTANT	Cas	TIME	...	T#15s	

Tab. 4: Tabulka lokálních proměnných (Header)

Class – Přepínač lokální proměnná/konstanta

Identifier – Identifikátor (symbolické, uživatelem definované označení)

Type – Datový typ proměnné

Initial – Výchozí hodnota

Comment – Komentář

Body (tělo programu) – do této části POU se zapisuje samotný program. Už při vytváření projektu máme možnost vybrat si z pěti programovacích jazyků, ve kterých budeme samotný program psát a to z jazyků grafických (LD – Ladder Diagram, FBD – Function Block diagram, SFC – Sequentional function chart) nebo jazyků textových (ST – Structured text, IL – Instruction List). Tyto jazyky jsou popsány normou IEC 61131-3, je zde ale ještě možnost psát program v jazyku Melsec IL (Melsec Instruction List), který slouží přímo pro programování automatů řady FX.

4.7 Použité prvky

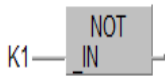
V této podkapitole budou popsány jednotlivé prvky, jež byly použity při vytváření vzorových úloh a jejich řešení. Budou uvedeny také nejčastěji používané prvky.

4.7.1 Vstupní proměnné

LD – Načte vstupní hodnotu proměnné na vrchol zásobníku. U ST a MELSEC IL nemůže být u vstupních proměnných použito symbolické označení.

Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	LD	-	LD		
Možný zápis	LD K1	IF X4...	LD X4		

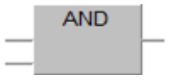
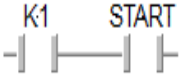
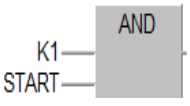
LDN – Načte negovanou vstupní hodnotu proměnné na vrchol zásobníku. U ST a MELSEC IL nemůže být u vstupních proměnných použito symbolické označení.

Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	LDN	-	LDI		-
Možný zápis	LDN K1	IF NOT X4...	LDI X4		

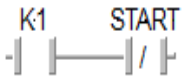
4.7.2 Logické funkce

Zde jsou popsány základní logické funkce a funkce použité v řešení úloh. U každého prvku nebo funkce bude napsáno označení popřípadě použití. V některých jazycích není pro některé prvky označení a tyto prvky musejí vzniknou spojením několika jiných prvků nebo jejich kombinací.

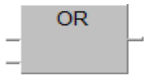
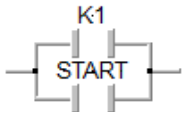
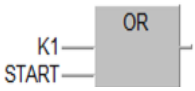
AND – Logický součin vstupních proměnných.

Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	AND	AND	AND	Prvky v sérii	
Možný zápis	LD K1 AND START	IF X4 AND X10	LD X4 AND X10		

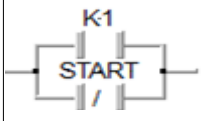
ANDN – Negovaný logický součin vstupní proměnné a bitu na vrcholu zásobníku.

Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	ANDN	AND NOT	ANI	Prvky v sérii	-
Možný zápis	LD K1 ANDN START	IF X4 AND NOT X10	LD X4 ANI X10		-

OR – Logický součet vstupní proměnné a bitu na vrcholu zásobníku.

Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	OR	OR	OR	Prvky paralelně	
Možný zápis	LD K1 OR START	IF X4 OR X10	LD X4 OR X10		

ORN – Logický součet negované vstupní proměnné a bitu na vrcholu zásobníku.

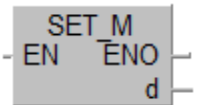
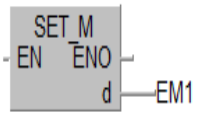
Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	ORN	OR NOT	ORI	Prvky paralelně	-
Možný zápis	LD K1 ORN START	IF X4 OR NOT X10	LD X4 ORI X10		-

4.7.3 Cívky (Coils)

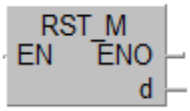
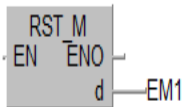
ST – Načte hodnotu na vrcholu zásobníku a přiřadí ji na výstup.

Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	ST	:=	OUT		
Možný zápis	LD K1 AND START ST EM1	POM1:=EM1;	LD X4 AND X10 OUT Y0		

SET – Jde o nastavení, když přijde na vstup logická 1, je na výstupu nastavena také logická 1, i když v dalším cyklu přijde na vstup logická 0, na výstupu je stále logická 1. SET lze vynulovat instrukcí RST.

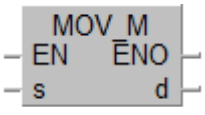
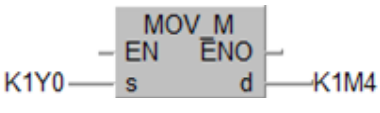
Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	S	SET_M(vstup, výstup);	SET		
Možný zápis	LD K1 AND START S EM1	SET_M(X4, EM1);	LD X4 AND X10 SET Y0		

RST – Opak funkce SET, nastaví na výstup logickou 0.

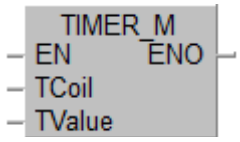
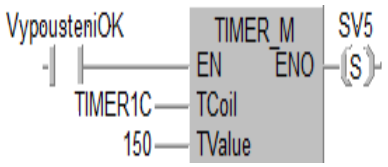
Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	R	RST_M(vstup, výstup);	RST		
Možný zápis	LD K1 AND START R EM1	RST_M(X4, EM1);	LD X4 AND X10 RST Y0		

4.7.4 Další použité funkce

MOV_M – tato funkce umožňuje přesunout data ze specifikované oblasti zdroje do specifikované oblasti cíle. Na vstup EN se přivádí spouštěcí podmínka, je-li podmínka splněna jsou hodnoty ze vstupu S přesunuty na výstup d. U jazyku Melsec IL se tato funkce jmenuje BMOV.

Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	MOV_M odkud, kam	-	BMOV odkud kam počet prvků		
Možný zápis	MOV_M K1Y0, K1M4	-	BMOV K1Y0 K1M4 K4		

TIMER_M – je to funkce časovače. Na vstup EN se přivádí spouštěcí podmínka, na vstup TCoil se nastavuje použitý timer (čaosvač) a do TValue se nastavuje potřebný čas.

Použitý jazyk	IL	ST	Melsec IL	LD	FBD
Symbolické označení	TIMER_M časovač,čas	-	-		
Možný zápis	TIMER_M TIMER1C,150	-	-		

TON – (zpožděné zapnutí) Tento časovač byl použit v jazyku strukturovaný text (ST). Po přivedení impulsu na vstupu IN, je časovač spuštěn a počítá do přednastavené hodnoty v PT (preset time). Uplynulý čas se zobrazuje v ET (established time), poté co se ET=PT je na výstupu Q logická 1.

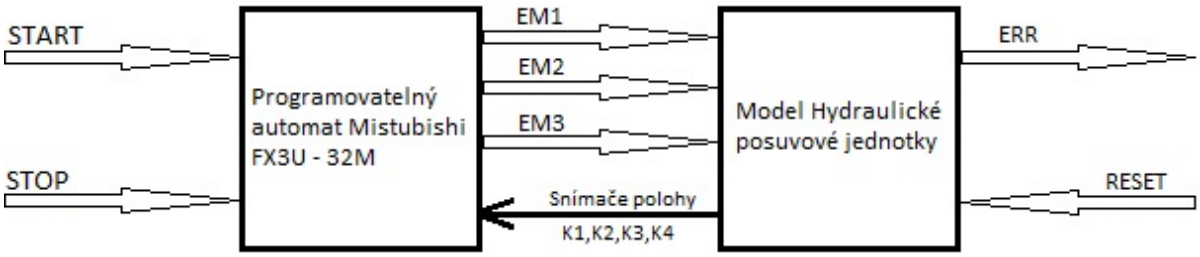
Použitý jazyk	ST
Symbolické označení	TON (IN:=,PT:=);
Možný zápis	TON(IN:=VypousteniOK ,PT:=Cas);

5 NAVRŽENÉ ÚLOHY A JEJICH ŘEŠENÍ

V následující kapitole budou vysvětlena jednotlivá zadání úloh pro modely procesů EDU-mod a to konkrétně pro Hydraulickou posuvovou jednotku a pro Mísicí jednotku. Ke každému z těchto modelů budou zadána 3 zadání a to seřazena vzestupně od nejjednoduššího až po nejtěžší. Následně bude ukázáno výsledné řešení těchto úloh a to v pěti programovacích jazycích (IL, FBD, LD, ST a MESLEC IL).

5.1 Hydraulická posuvová jednotka

Hlavní funkce modelu byly již popsány v kapitole 2.1, zde si tedy už jen shrneme to nejdůležitější. Pro řešení zadaných úloh budou potřeba 2 externí tlačítka a to tlačítko START a tlačítko STOP. Dále model hydraulické posuvové jednotky, programovatelný automat Mitsubishi FX3U-32M a počítač s nainstalovaným softwarem GX IEC Developer 7.01. V obrázku níže je zobrazeno blokové schéma, které znázorňuje komunikaci mezi automatem a modelem hydraulické posuvové jednotky.



Obr. 23: Blokové schéma komunikace Hydraulické posuvové jednotky s automatem

V následující tabulce je uvedeno, jak jsou propojeny jednotlivé vstupní a výstupní proměnné s automatem Mitsubishi FX3U-32M podle tabulky zapojení vodičů na boku modelu hydraulické posuvové jednotky.

Vstupní proměnné	Adresa	Popis
K1	X4	Snímač polohy K1
K2	X5	Snímač polohy K2
K3	X6	Snímač polohy K3
K4	X7	Snímač polohy K4
START	X10	Tlačítko START
STOP	X11	Tlačítko STOP

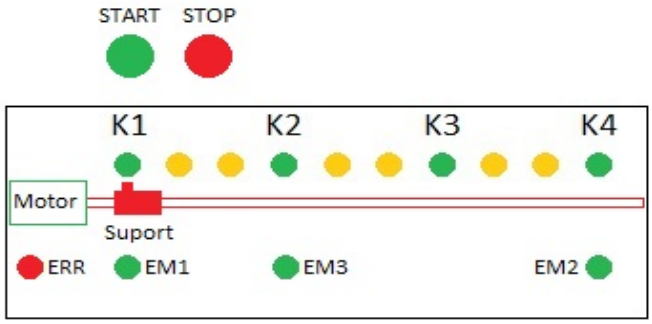
Výstupní proměnné	Adresa	Popis
EM1	Y0	Elektromotor1 (pohyb vlevo)
EM2	Y1	Elektromotor2 (pohyb vpravo)
EM3	Y2	Elektromotor3(pracovní pohyb)

Tab. 5: Propojení vstupních a výstupních proměnných s automatem

5.1.1 Zadání 1. úlohy

Naprogramujte řízení hydraulické posuvové jednotky tak, aby po stlačení tlačítka START a umístění suportu ve výchozí poloze K1 se spustil výstup EM1, následně až bude suport v poloze K4 se tento EM1 vypne a zapne se výstup EM2, suport se bude vracet zpět až do polohy K1 kde se následně vypne i EM2 a bude se čekat na opětovné stlačení tlačítka START.

Schéma funkce Hydraulické posuvové jednotky je uvedeno v následujícím obrázku:



Obr. 24: Schéma funkce Hydraulické posuvové jednotky

5.1.2 Řešení 1. úlohy (IL, ST, Melsec IL, FBD, LD)

Pro řešení této úlohy si nejprve vyplníme tabulku Global_Vars, která bude stejná pro řešení úlohy všemi zmíněnými programovacími jazyky a to tímto způsobem:

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type		Initial	Comment	Remark
0	VAR_GLOBAL	K1	X4	%IX4	BOOL	...	FALSE	Snímač K1 na vstupu X4	
1	VAR_GLOBAL	K4	X7	%IX7	BOOL	...	FALSE	Snímač K4 na vstupu X7	
2	VAR_GLOBAL	EM1	Y0	%QX0	BOOL	...	FALSE	EM1 na výstupu Y0	
3	VAR_GLOBAL	EM2	Y1	%QX1	BOOL	...	FALSE	EM2 na výstupu Y1	
4	VAR_GLOBAL	START	X10	%IX8	BOOL	...	FALSE	Tlačítko START na vstupu X10	

Tab. 6: Tabulka Global_Vars pro úlohu č. 1

Následuje řešení pomocí jazyka Instruction List (Instrukčního listu).

```
Network 1:
LD K1                    // impuls na snímači K1(suport je ve výchozí poloze)
AND START     // zároveň se objeví impuls ve START (stlačení tlačítka START)
S EM1                                                        // zapnut EM1
Network 2:
LD K4                    // impuls na snímači K4 (suport je v koncové poloze)
AND EM1                                          // a zároveň spuštěn EM1
R EM1                                              // je tento EM1 vypnut
S EM2                                                    // a spuštěn EM2
Network 3:
LD K1                                // impuls na snímači K1 (výchozí poloha)
AND EM2                                          // a zároveň je spuštěn EM2
R EM2                                              // je tento EM2 vypnut
```

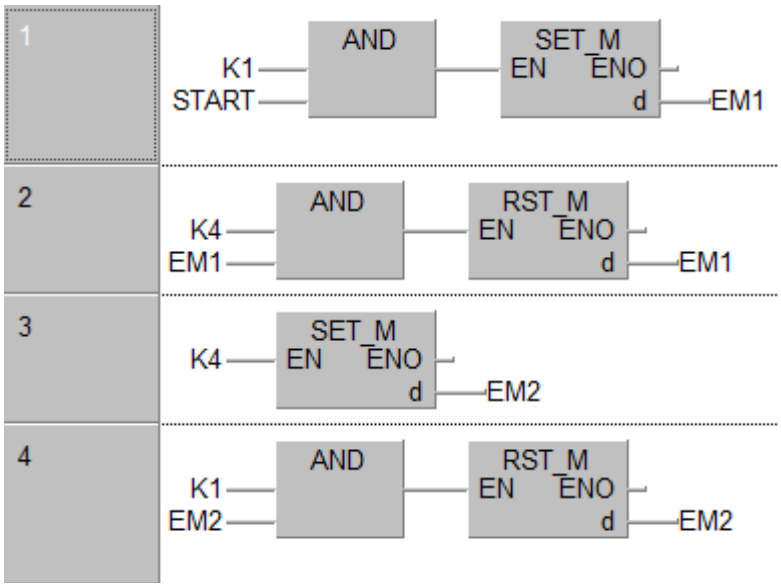
Výsledné řešení pomocí jazyka Structured Text (Strukturovaný text) by mohlo vypadat takto:

```
IF X4 AND START THEN  // impuls na snímači K1 (suport je ve výchozí poloze)
      EM1 := 1;     // zároveň se objeví impuls ve START (stlačeni tlačítka
                                                    // START) je zapnut EM1
ELSIF X7 AND EM1 THEN  // impuls na snímači K4 (suport je v koncové poloze)
                                                    // a zároveň zapnut EM1
      EM1 := 0;                                              // vypnut EM1
      EM2 := 1;                                              // zapnut EM2
ELSIF X4 AND EM2 THEN  // impuls na snímači K1 (výchozí poloha)  a zároveň
                                                  // je spuštěn EM2
      EM2:=0;                                                // EM2 vypnut
END_IF;
```

Výsledné řešení pomocí jazyka Melsec IL (Melsec Instrukčního listu):

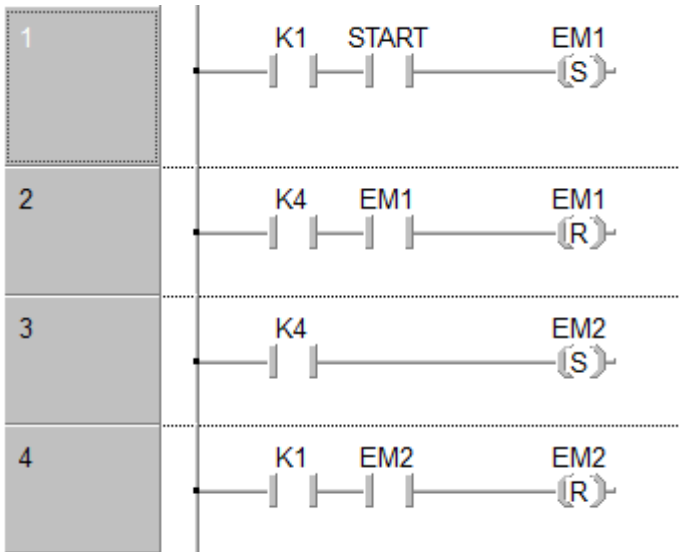
```
Network 1:
LD X4                  // impuls na snímači K1 (suport je ve výchozí poloze)
AND START    // zároveň se objeví impuls ve START (stlačení tlačitka START)
SET EM1                                                    // zapnut EM1
Network 2:
LD X7                  // impuls na snímači K4 (suport je v koncové poloze)
AND EM1                                          // a zároveň spuštěn EM1
RST EM1                                            // je tento EM1 vypnut
SET EM2                                                  // a spuštěn EM2
Network 3:
LD X4                                 // impuls na snímači K1 (výchozí poloha)
AND EM2                                          // a zároveň je spuštěn EM2
RST EM2                                            // je tento EM2 vypnut
```

Výsledné řešení pomocí jazyka FBD (jazyk blokových schémat):



Obr. 25: Výsledné řešení pomocí jazyka FBD

Výsledné řešení pomocí jazyka LD (jazyk kontaktních schémat):



Obr. 26: Výsledné řešení pomocí jazyka LD

5.1.3 Zadání 2. úlohy

V druhé úloze je zadání stejné jako v první, jen s tím rozdílem, že při pohybu vpravo od snímače K2 ke snímači K3 bude spuštěn pracovní pohyb (EM3) a poté při vracení se suportu do výchozí polohy bude opět od snímače K3 po snímač K2 spuštěn pracovní pohyb (EM3).

5.1.4 Řešení 2. úlohy (IL, ST, Melsec IL, FBD, LD)

Oproti 1. úloze jsou do tabulky Global_Vars doplněny následující proměnné.

5	VAR_GLOBAL	K2	X5	%IX5	BOOL	FALSE	Snímač K2 na vstupu X5	
6	VAR_GLOBAL	K3	X6	%IX6	BOOL	FALSE	Snímač K3 na vstupu X6	
7	VAR_GLOBAL	EM3	Y2	%QX2	BOOL	FALSE	EM3 na výstupu Y2	

Tab. 7: Doplněné proměnné do tabulky Global_Vars z úlohy 1.

Následuje řešení úlohy č. 2 pomocí jazyka Instuction List.

```
Network 1:
LD K1                  // impuls na snímači K1 (suport je ve výchozí poloze)
AND START    // zároveň se objeví impuls ve START (stlačení tlačitka START)
S EM1                                                          // zapnut
Network 2:
LD K2                                                  // Suport v poloze K2
AND EM1                    // zároveň je spuštěn EM1 (pohyb suportu vpravo)
S EM3                                  // je spuštěn EM3 - pracovní pohyb
Network 3:
LD K3                                                  // Suport v poloze K3
AND EM1                    // zároveň je spuštěn EM1 (pohyb suportu vpravo)
R EM3                            // zastaven EM3 - konec pracovního pohybu
```

```
Network 4:
LD K4                  // impuls na snímači K4 (suport je v koncové poloze)
AND EM1                                        // a zároveň je spuštěn EM1
R EM1                                            // je tento EM1 vyresetová
S EM2                                                        // spuštěn EM2
Network 5:
LD K3                                                     // Suport v poloze K3
AND EM2                     // zároveň je spuštěn EM2 (pohyb suportu vlevo)
S EM3                                        // spuštěn EM3 - pracovní pohyb
Network 6:
LD K2                                                     // Suport v poloze K2
AND EM2                     // zároveň je spuštěn EM2 (pohyb suportu vlevo)
R EM3                              // zastaven EM3 - konec pracovniho pohybu
Network 7:
LD K1                                  // Suport v poloze K1 (vychozi poloha)
AND EM2                                                // zároveň je spuštěn EM2
R EM2                                                // je tento EM2 vyresetován
```

Výsledné řešení pomocí jazyka Structured Text.

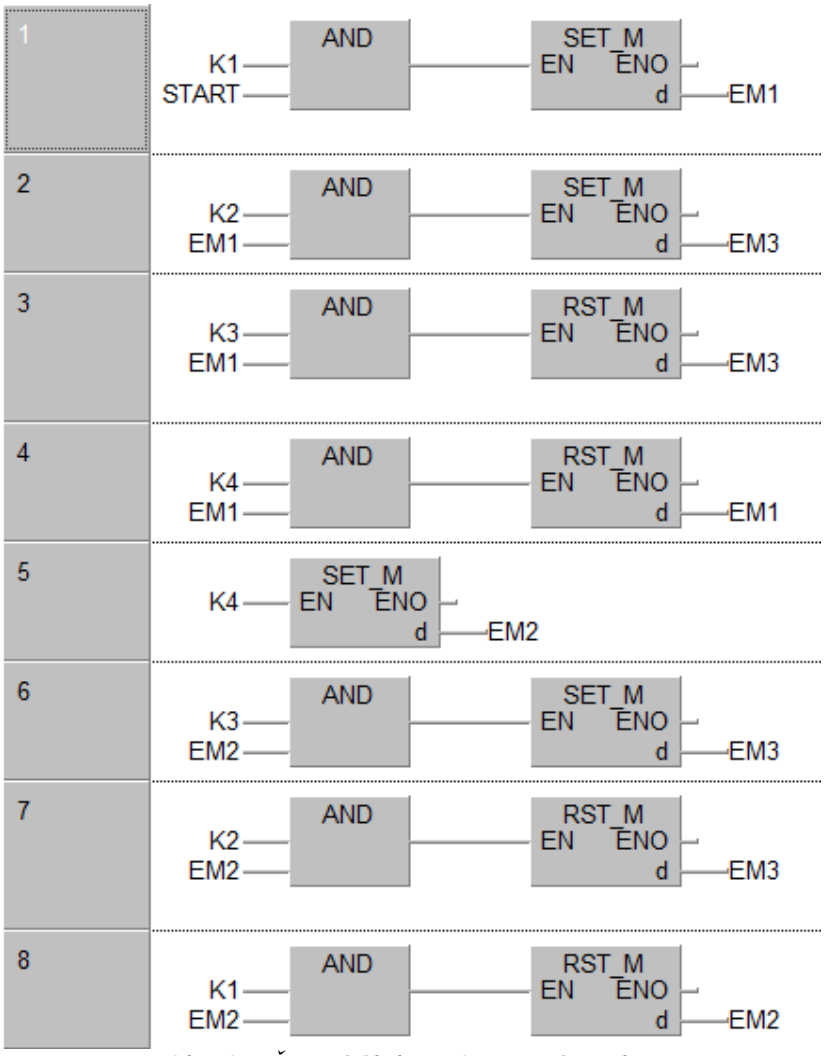
```
IF X4 AND START THEN      // impuls na snímači X4(K1) (suport je ve vychozí
      EM1 := 1;    // poloze) a zároveň se objeví impuls ve START (stlačeni
                                                  // tlačitka START) je zapnut EM1
ELSIF X5 AND EM1 THEN    // Suport v poloze X5(K2) a zároveň je spuštěn EM1
      EM3 := 1;       // (pohyb suportu vpravo)spuštěn EM3 – pracovní pohyb

ELSIF X6 AND EM1 THEN    // Suport v poloze X6(K3) a zároveň je spuštěn EM1
      EM3 := 0;              //(pohyb suportu vpravo) zastaven EM3 - konec
                                                        // pracovního pohybu
ELSIF X7 AND EM1 THE // impuls na snímači X7(K4- suport v koncové poloze) a
      EM1 := 0;                        // zároveň spuštěn EM1 je EM1 vypnut
      EM2 := 1;                                  // spuštěn EM2 je zapnut

ELSIF X6 AND EM2 THEN // Suport v poloze K3 a zároveň je spuštěn EM2 (pohyb
      EM3 := 1;              // suportu vlevo)spuštěn EM3 – pracovní pohyb

ELSIF X5 AND EM2 THEN // Suport v poloze K2 a zároveň je spuštěn EM2 (pohyb
      EM3 := 0;    // suportu vlevo) zastaven EM3 - konec pracovního pohybu

ELSIF X4 AND EM2 THEN      // Suport se vrací do polohy X4(K1) a zároveň je
   EM2:=0;                                       // spuštěn EM2 je EM2 vypnut

END_IF;
```

Výsledné řešení pomocí jazyka Melsec IL.

```
Network 1:
LD X4                  // impuls na snímači K1 (suport je ve vychozí poloze)
AND START    // zároveň se objeví impuls ve START (stlačení tlačítka START)
SET EM1                                                      // zapnut EM1
Network 2:
LD X5                                                 // Suport v poloze K2 a
AND EM1                     // zároveň je spuštěn EM1 (pohyb suportu vpravo)
SET EM3                                  // je spuštěn EM3 - pracovní pohyb
LD X6                                                    // Suport v poloze K3
AND EM1                     // zároveň je spuštěn EM1 (pohyb suportu vpravo)
RST EM3                            // zastaven EM3 - konec pracovniho pohybu
```

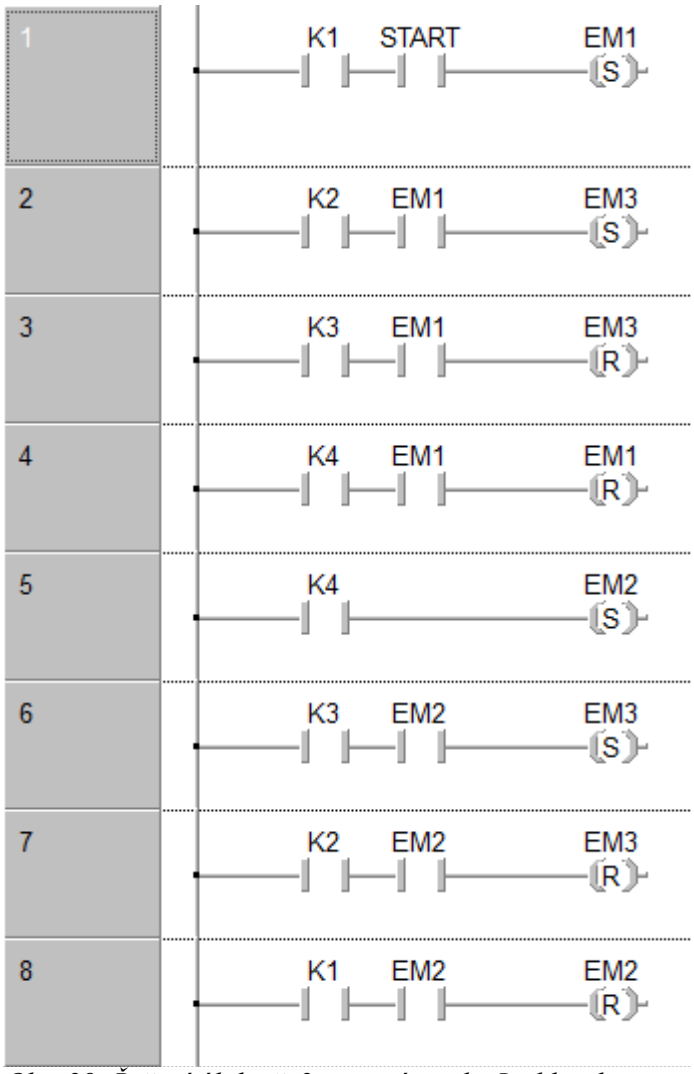
```
Network 3:
LD X7                    // impuls na snímači K4 (suport je v koncové poloze)
AND EM1                                          // a zároveň je spuštěn EM1
RST EM1                                          // je tento EM1 vyresetován
SET EM2                                                    // spuštěn EM2
Network 4:
LD X6                                                  // Suport v poloze K3
AND EM2                  // zároveň je spuštěn EM2 (pohyb suportu vlevo)
SET EM3                                      // spuštěn EM3 - pracovni pohyb
LD X5                                                  // Suport v poloze K2
AND EM2                  // zároveň je spuštěn EM2 (pohyb suportu vlevo)
RST EM3                        // zastaven EM3 - konec pracovního pohybu
Network 5:
LD X4                                // Suport v poloze K1 (výchozi poloha)
AND EM2                                            // zároveň je spuštěn EM2
RST EM2                                          // je tento EM2 vyresetován
```

Výsledné řešení pomocí jazyka FBD.



Obr. 27: Řešení úlohy č. 2 pomocí jazyka FBD

Výsledné řešení pomocí jazyka Ladder Diagram.



Obr. 28: Řešení úlohy č. 2 pomocí jazyka Ladder diagram

5.1.5 Zadání 3. úlohy

Ve 3. úloze jen rozšíříme zadání z úlohy 2. a to tak, že přidáme externí tlačítko STOP, po jehož stlačení se zastaví pohyb suportu v libovolné poloze. Poté po stlačení tlačítka START bude suport pokračovat v předchozím pohybu z místa, kde byl zastaven.

5.1.6 Řešení 3. úlohy (IL, ST, Melsec IL, FBD, LD)

Do tabulky Global_Vars přidáme následující 2 proměnné a to externí tlačítko STOP a pomocnou proměnnou JEDU (je to paměťový bit na adrese M0, který bude signalizovat zda je suport v pohybu 1, či zda stojí 0):

8	VAR_GLOBAL	STOP	X11	%IX9	BOOL	...	FALSE	Tlačítko STOP na vstupu X11
9	VAR_GLOBAL	JEDU	M0	%MX0.0	BOOL	...	FALSE	Paměťový bit na adrese M0

Tab. 8: Doplněné proměnné do tabulky Global_Vars z úlohy 2.

Výsledné řešení úlohy č. 3 jazykem Instruction List:

```
Network 1:
LD K1                  // impuls na snímači K1 (suport je ve výchozí poloze)
AND START    // zároveň se objeví impuls ve START (stlačení tlačitka START)
S EM1                                                        // zapnut EM1
S JEDU                          // nastavena logická 1 v paměťovém bitu JEDU
Network 2:
LD K2                                                // Suport v poloze K2 a
AND EM1                      // zároveň je spuštěn EM1 (pohyb suportu vpravo)
S EM3                                    // je spuštěn EM3 - pracovni pohyb
Network 3:
LD K3                                                 // Suport v poloze K3
AND EM1                      // zároveň je spuštěn EM1 (pohyb suportu vpravo)
R EM3                              // zastaven EM3 - konec pracovniho pohybu
Network 4:
LD K4                  // impuls na snímači K4 (suport je v koncové poloze)
AND EM1                                          // a zároveň je spuštěn EM1
R EM1                                             // je tento EM1 vyresetován
S EM2                                                        // spuštěn EM2
Network 5:
LD K3                                                 // Suport v poloze K3
AND EM2                       // zároveň je spuštěn EM2 (pohyb suportu vlevo)
S EM3                                       // spuštěn EM3 - pracovní pohyb
Network 6:
LD K2                                                 // Suport v poloze K2
AND EM2                       // zároveň je spuštěn EM2 (pohyb suportu vlevo)
R EM3                              // zastaven EM3 - konec pracovního pohybu
Network 7:
LD K1                                  // Suport v poloze K1 (výchozi poloha)
AND EM2                                                // zároveň je spuštěn EM2
R EM2                                             // je tento EM2 vyresetován
R JEDU                                       // vyresetován paměťový bit JEDU
Network 8:
LD STOP                           //  impuls na STOP (stlačení tlačítka STOP)
AND JEDU                                        // a zároveň je v JEDU logická 1
MOV_M   K1Y0, K1M4     // jsou pomocí funkce MOV_M přesunuty hodnoty z 1. 4
                          //výstupních proměnných do paměťových bitů M0-M4
R JEDU                                       // vyresetován paměťový bit JEDU
R EM1                                                        // vypnut EM1
R EM2                                                        // vypnut EM2
```

```
R EM3                                                    // vypnut EM3
Network 9:
LD START                          // impuls na START (stlačení tlačítka START)
ANDN JEDU                                    // a zároveň je v JEDU logická 0
MOV_M   K1M4, K1Y0   // jsou pomoci funkce MOV_M přesunuty hodnoty z 1.4
            // paměťových bitů M0-M4 zpět do 1. 4 výstupních proměnných a
           // pokračuje se v pohybu, který byl před stlačenim tlačítka STOP
S JEDU                            // nastavena logicka 1 v paměťovém bitu JEDU
```

Výsledné řešení úlohy č. 3 jazykem Structured text: Při řešení této úlohy vytvořímei další 3 proměnné (POM1, POM2, POM3), které slouží jako pomocné, do kterých se po stlačení tlačítka start uloží aktuální hodnota z výstupních proměnných EM1, EM2 a EM3. Musíme tedy doplnit tabulku Global_Vars o následující řádky:

VAR_GLOBAL	POM1	M2	%MX0.2	BOOL	FALSE
VAR_GLOBAL	POM2	M3	%MX0.3	BOOL	FALSE
VAR_GLOBAL	POM3	M4	%MX0.4	BOOL	FALSE

Tab. 9: Doplnění tabulky Global_Vars pro ST

Výsledné řešení úlohy č. 3 jazykem Structured text:

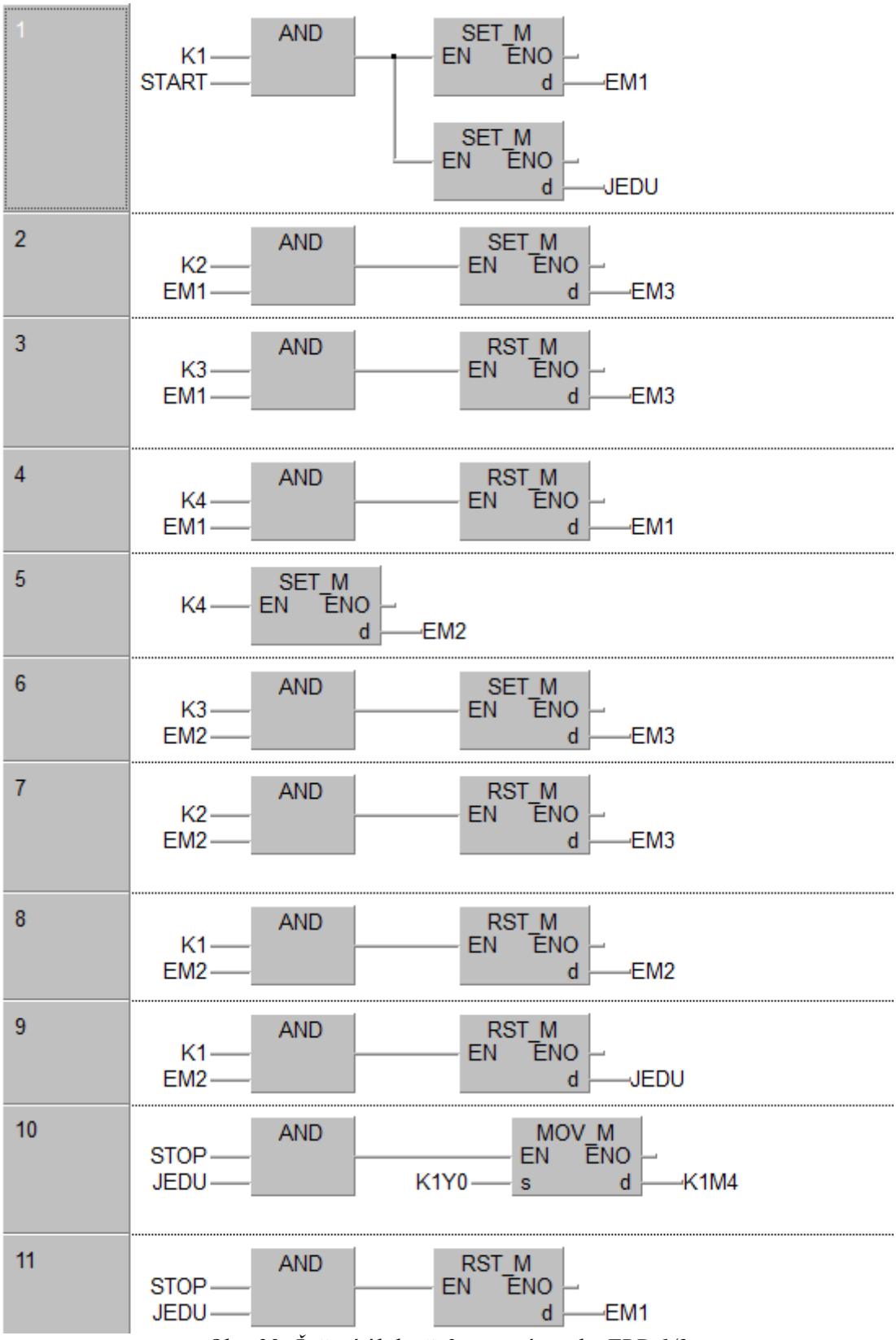
```
IF X4 AND START THEN      // impuls na snímači X4(K1) (suport je ve výchozí
      EM1 := 1;              // poloze) a zároveň se objeví impuls ve START
      JEDU:=1;                  // (stlačení tlačítka START) je zapnut EM1
                             //  je nastavena logická 1 v paměťovém bitu JEDU
ELSIF X5 AND EM1 THEN    // Suport v poloze X5(K2) a zároveň je spuštěn EM1
      EM3 := 1;       // (pohyb suportu vpravo)spuštěn EM3 - pracovní pohyb

ELSIF X6 AND EM1 THEN    // Suport v poloze X6(K3) a zároveň je spuštěn EM1
      EM3 := 0;                        // (pohyb suportu vpravo)zastaven EM3
                                                  // konec pracovního pohybu
ELSIF X7 AND EM1 THE   // impuls na snímači X7(K4- suport v koncové poloze)
      EM1 := 0;                     // a zároveň spuštěn EM1 je EM1 vypnut
      EM2 := 1;                                          // EM2 je zapnut

ELSIF X6 AND EM2 THEN        // Suport v poloze K3 a zároveň je spuštěn EM2
      EM3 := 1;        //(pohyb suportu vlevo) spuštěn EM3 - pracovní pohyb

ELSIF X5 AND EM2 THEN        // Suport v poloze K2 a zároveň je spuštěn EM2
      EM3 := 0;                  // zastaven EM3 - konec pracovního pohybu

ELSIF X4 AND EM2 THEN      // Suport se vraci do polohy X4(K1) a zároveň je
      EM2:=0;                                  // spuštěn EM2 je EM2 vypnut
      JEDU:=0;       // vyresetován paměťový bit JEDU - nastavena logická 0

ELSIF STOP AND JEDU THEN       // impuls na STOP a zároveň v JEDU logická 1
   POM1:=EM1;         // zkopirována hodnota z proměnné EM1 do proměnné POM1
   POM2:=EM2;         // zkopirována hodnota z proměnné EM2 do proměnné POM2
   POM3:=EM3;         // zkopirována hodnota z proměnné EM3 do proměnné POM3
   EM3 := 0;                        // Je vyresetována výstupní proměnná EM3
   EM2 := 0;                        // Je vyresetována výstupní proměnná EM2
   EM1 := 0;                        // Je vyresetována výstupní proměnná EM1
   JEDU :=0;      // Je vyresetován paměťový bit JEDU (nastavena logická 0)

ELSIF START AND NOT JEDU THEN // impuls na START a zároveň v JEDU logická 0
```

```
   EM1:=POM1;    // zkopirována hodnota z proměnné POM1 zpět do proměnné EM1
   EM2:=POM2;    // zkopirována hodnota z proměnné POM2 zpět do proměnné EM2
   EM3:=POM3;    // zkopirována hodnota z proměnné POM3 zpět do proměnné EM3
   JEDU :=1;             // nastaven paměťový bit JEDU (nastavena logická 1)

END_IF;
```

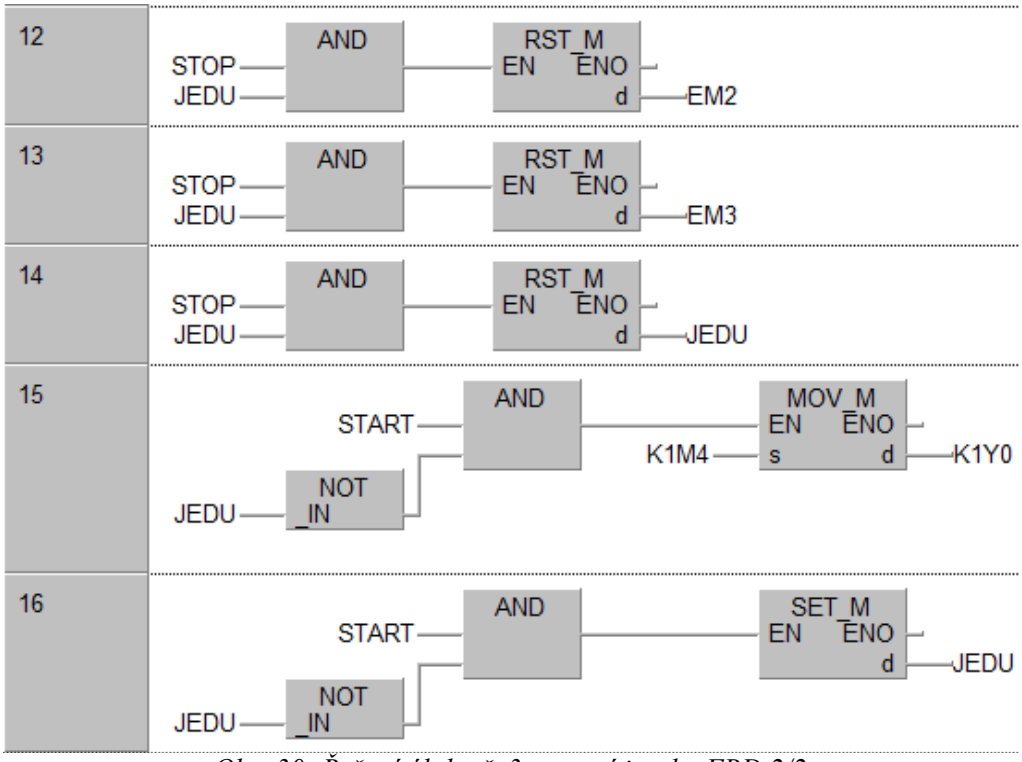
Výsledné řešení pomocí jazyka Melsec IL.

```
Network 1:
LD X4                  // impuls na snímači K1 (suport je ve výchozi poloze)
AND START    // zároveň se objeví impuls ve START (stlačení tlačítka START)
SET EM1                                                        // zapnut EM1
SET JEDU                       // nastavena logicka 1 v paměťovém bitu JEDU
Network 2:
LD X5                                                // Suport v poloze K2 a
AND EM1                     // zároveň je spuštěn EM1 (pohyb suportu vpravo)
SET EM3                                  // je spuštěn EM3 - pracovní pohyb
LD X6                                                 // Suport v poloze K3
AND EM1                     // zároveň je spuštěn EM1 (pohyb suportu vpravo)
RST EM3                            // zastaven EM3 - konec pracovního pohybu
Network 3:
LD X7                  // impuls na snímači K4 (suport je v koncové poloze)
AND EM1                                          // a zároveň je spuštěn EM1
RST EM1                                          // je tento EM1 vyresetován
SET EM2                                                      // spuštěn EM2
Network 4:
LD X6                                                 // Suport v poloze K3
AND EM2                      // zároveň je spuštěn EM2 (pohyb suportu vlevo)
SET EM3                                    // spuštěn EM3 - pracovní pohyb
LD X5                                                 // Suport v poloze K2
AND EM2                      // zároveň je spuštěn EM2 (pohyb suportu vlevo)
RST EM3                            // zastaven EM3 - konec pracovniho pohybu
Network 5:
LD X4                                 // Suport v poloze K1 (výchozí poloha)
AND EM2                                            // zároveň je spuštěn EM2
RST EM2                                          // je tento EM2 vyresetován
RST JEDU             // vyresetován paměťový bit JEDU (nastavena logická 0)
Network 6:
LD STOP                           // impuls na STOP (stlačení tlačítka STOP)
AND JEDU                                    // zároveň je v JEDU logická 1
BMOV K1Y0 K1M4 K4       // jsou pomocí funkce BMOV přesunuty hodnoty z 1. 4
                          // výstupních proměnných do paměťových bitů M0-M4
RST JEDU                                    // vyresetován paměťový bit JEDU
RST EM1                                                        // vypnut EM1
RST EM2                                                        // vypnut EM2
RST EM3                                                        // vypnut EM3
Network 7:
LD START                        // impuls na START (stlačení tlačítka START)
ANI JEDU                                    // a zároveň je v JEDU logická 0
BMOV K1M4 K1Y000 K4     // jsou pomoci funkce BMOV přesunuty hodnoty z 1. 4
// paměťových bitu M0-M4 zpět do 1. 4 výstupnich proměnných a pokračuje se
// v pohybu který byl pred stlačením tlačítka STOP
SET JEDU                       // nastavena logická 1 v paměťovém bitu JEDU
```

Výsledné řešení úlohy č. 3 jazykem FBD:

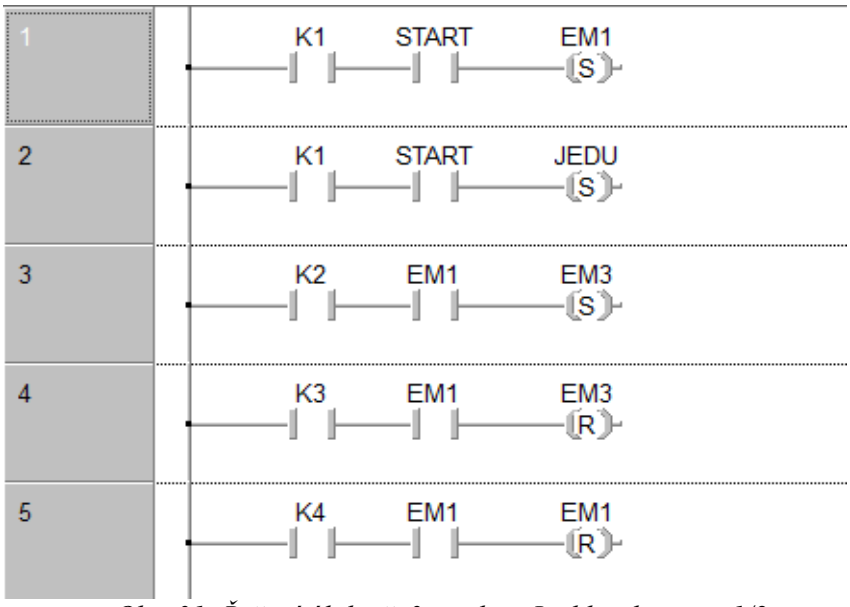


Obr. 29: Řešení úlohy č. 3 pomocí jazyka FBD 1/2

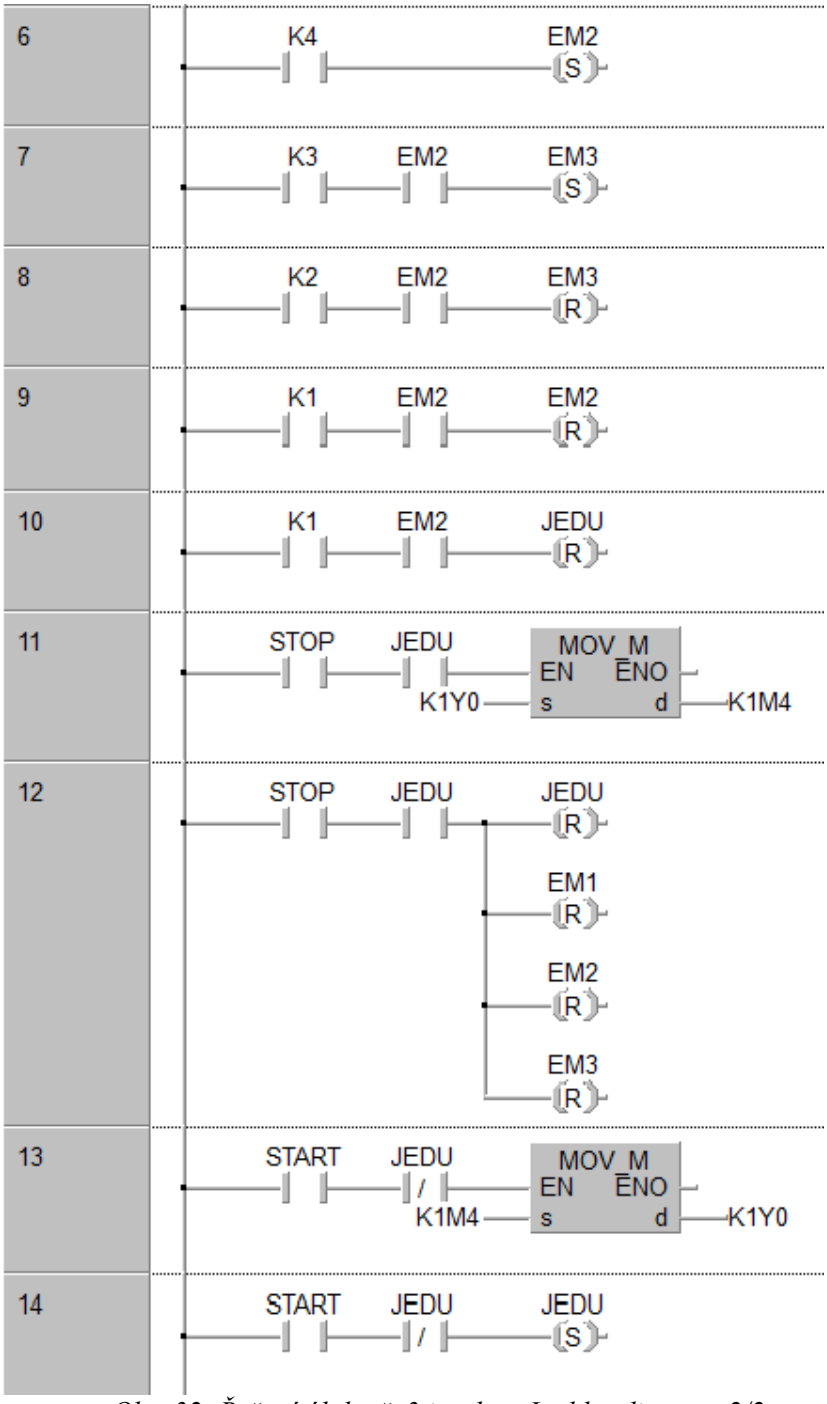


Obr. 30: Řešení úlohy č. 3 pomocí jazyka FBD 2/2

Výsledné řešení úlohy č. 3 jazykem Ladder diagram:



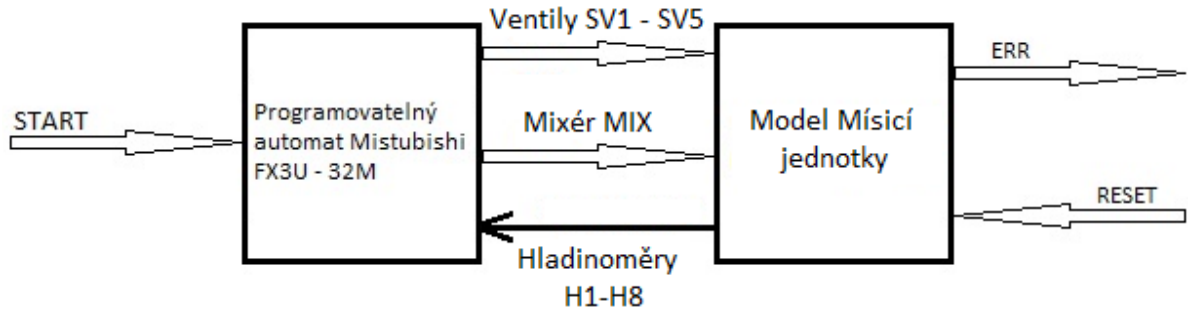
Obr. 31: Řešení úlohy č. 3 jazykem Ladder diagram 1/2



Obr. 32: Řešení úlohy č. 3 jazykem Ladder diagram 2/2

5.2 Mísicí jednotka

Hlavní funkce této jednotky již byly popsány v kapitole 2.2. Pro vypracování řízení zadaných úloh budeme potřebovat jedno externí tlačítko START, model mísicí jednotky, programovatelný automat Mitsubish FX3U-32M a počítač s nainstalovaným softwarem GX IEC Developer 7.01. Níže je zobrazeno blokové schéma komunikace automatu s modelem mísicí jednotky.



Obr. 33: Blokové schéma komunikace Mísicí jednotky s automatem

V následující tabulce je uvedeno jak jsou propojeny jednotlivé vstupní a výstupní proměnné s automatem Mitsubishi FX3U-32M podle tabulky zapojení vodičů na boku modelu hydraulické posuvové jednotky.

Vstupní proměnné	Adresa	Popis
H1	X7	Hladinoměr H1
H2	X6	Hladinoměr H2
H3	X5	Hladinoměr H3
H4	X4	Hladinoměr H4
H5	X3	Hladinoměr H5
H6	X0	Hladinoměr H6
H7	X1	Hladinoměr H7
H8	X2	Hladinoměr H8
START	X10	Tlačítko START

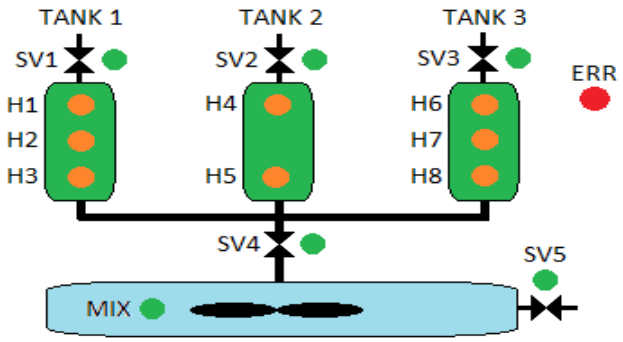
Výstupní proměnné	Adresa	Popis
SV1	Y0	Ventil SV1
SV2	Y1	Ventil SV2
SV3	Y2	Ventil SV3
SV4	Y5	Ventil SV4
SV5	Y4	Ventil SV5
MIX	Y6	Mixér MIX

Tab. 10: Propojení vstupních a výstupních proměnných s automatem

5.2.1 Zadání 1. úlohy

Naprogramujte řízení mísicí jednotky tak, aby po stlačení tlačítka START se otevřely ventily SV1, SV2, SV3 a naplnili všechny 3 tanky po hladiny H2, H4 a H7. Řízení rozdělte na 2 podprogramy, kde v 1. budou ošetřeny počáteční podmínky tak, že se bude čekat na stlačení tlačítka START, po jeho stlačení budou vyresetovány všechny výstupy, abychom měli jistotu že je vše uzavřeno. Následně bude spuštěn 2. podporgram ve kterém se naplní všechny 3 tanky. Po jejich naplnění se ventily SV1, SV2 a SV3 uzavřou.

Schéma funkce mísicí jednotky je uvedeno v následujícím obrázku:



Obr. 34: Schéma funkce Mísicí jednotky

5.2.2 Řešení 1. úlohy (IL, ST, Melsec IL, FBD, LD)

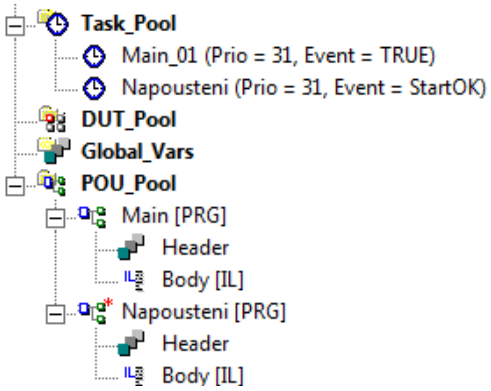
Pro řešení této úlohy si nejprve vyplníme tabulku Global_Vars, která bude stejná pro řešení úlohy všemi zmíněnými programovacími jazyky a to tímto způsobem:

VAR_GLOBAL	H1	X7	%IX7	BOOL	...	FALSE
VAR_GLOBAL	H2	X6	%IX6	BOOL	...	FALSE
VAR_GLOBAL	H3	X5	%IX5	BOOL	...	FALSE
VAR_GLOBAL	H4	X4	%IX4	BOOL	...	FALSE
VAR_GLOBAL	H5	X3	%IX3	BOOL	...	FALSE
VAR_GLOBAL	H6	X0	%IX0	BOOL	...	FALSE
VAR_GLOBAL	H7	X1	%IX1	BOOL	...	FALSE
VAR_GLOBAL	H8	X2	%IX2	BOOL	...	FALSE
VAR_GLOBAL	SV1	Y0	%QX0	BOOL	...	FALSE
VAR_GLOBAL	SV2	Y1	%QX1	BOOL	...	FALSE
VAR_GLOBAL	SV3	Y2	%QX2	BOOL	...	FALSE
VAR_GLOBAL	SV4	Y5	%QX5	BOOL	...	FALSE
VAR_GLOBAL	SV5	Y4	%QX4	BOOL	...	FALSE
VAR_GLOBAL	MIX	Y6	%QX6	BOOL	...	FALSE
VAR_GLOBAL	START	X10	%IX8	BOOL	...	FALSE
VAR_GLOBAL	StartOK	M0	%MX0.0	BOOL	...	FALSE

Tab. 11: Tabulka Global_Vars pro úlohu č. 1

V tabulce je uvedena i paměťová proměnná StartOK, která bude sloužit jako podmínka pro spuštění 2. podprogramu po splnění podmínek z podprogramu 1.

V následujícím obrázku je zobrazena struktura nového projektu, tato struktura je stejná pro všech 5 programovacích jazyků (IL, ST, FBD, LD, Melsec IL), pouze při vytváření projektu musíme, vybrat jaký z těchto jazyků budeme preferovat, označení tohoto jazyku bude potom zobrazeno v hranatých závorkách za Body v každé POU.



Obr. 35: Struktura nově vytvořeného projektu úlohy č. 1

Následuje řešení pomocí jazyka Instruction List (Instrukčního listu). V komentářích zdrojového kódu budou vždy popsány jednotlivé networky.

Obsah těla (Body [IL]) POU s názvem Main [PRG]:

```
Network 1:
LD M8000        // speciální pameť M8000, ve které je po celou dobu uložena
AND START             //logická 1 v kombinaci se stlačeným tlačitkem START
R SV1                                          // vyresetuje výstup SV1
R SV2                                          // vyresetuje výstup SV2
R SV3                                           // vyresetuje výstup SV3
R SV4                                           // vyresetuje výstup SV4
R SV5                                           // vyresetuje výstup SV5
R MIX                                           // vyresetuje výstup MIX
S StartOK     //  nastaví do StartOK logickou 1 pro spuštění POU Napouštění
```

Následuje obsah těla (Body [IL]) POU s názvem Napousteni [PRG]:

```
Network 1:
LD StartOK                                    // po splnění podmínky StartOK
ANDN H2                                         // a při prázdné hladině H1
S SV1                                             //  je otevřen ventil SV1

LD H2                                       // Po naplněni tanku do hladiny H1
R SV1                                          //   je   ventil   SV1   zavřen
Network 2:
LDN H7                        //  Pokud  není  tank  3  naplněn  po  hladinu  H7
S SV3                                          // je otevřen ventil SV3

LD H7                                     //  Po  naplněni  tanku  do  hladiny  H7
R SV3                                                 // je ventil SV3 zavřen
Network 3:
LDN H4                         //  Pokud  není  tank  3  naplněn  po  hladinu  H4
S SV2                                          // je otevřen ventil SV2

LD H4                                     //  Po  naplněni  tanku  do  hladiny  H4
R SV2                                          //  je ventil SV2 zavřen
```

```
R StartOK // vyresetován paměťový bit StartOK, aby mohl být použit v dalším
                                                                    // cyklu
```

Výsledné řešení pomocí jazyka Structured Text (Strukturovaný text) by mohlo vypadat takto. Obsah těla (Body [ST]) POU s názvem Main [PRG]:

```
IF M8000 AND START THEN        // Pro spuštění programu využijeme speciální
   SV1 := 0;         // pameť M8000 V kombinaci se stlačeným tlačítkem START
   SV2 := 0;                                    // Vyresetuje výstup SV1, SV2
   SV3 := 0;                                         // Vyresetuje výstup SV3
   SV4 := 0;                                         // Vyresetuje výstup SV4
   SV5 := 0;                                         // Vyresetuje výstup SV5
   MIX := 0;                                         // Vyresetuje výstup MIX
   StartOK := 1;         // A nastaví do StartOK logickou 1 pro spuštěni POU
                                                                  // Napouštění
END_IF;
```

Obsah těla (Body [ST]) POU s názvem Napousteni [PRG]:

```
IF StartOK AND NOT X6 THEN     // Po splnění podmínky StartOK a při prázdné
      SV1:=1;                            // hladině H2 je otevřen ventil SV1
END_IF;

IF X6 THEN                        // Pokud není tank 2 naplněn po hladinu H2
      SV1:=0;                                    // je otevřen ventil SV1
END_IF;

IF  NOT X1 THEN                          // Po naplněni tanku do hladiny H7
      SV3:=1;                                    // je ventil SV3 zavřen
END_IF;

IF  X1 THEN                              // Po naplněni tanku do hladiny H7
      SV3:=0;                                    // je ventil SV3 zavřen
END_IF;

IF NOT X4 THEN                    // Pokud není tank 3 naplněn po hladinu H4
      SV2:=1;                                    // je otevřen ventil SV2
END_IF;

IF X4 THEN                               // Po naplnění tanku do hladiny H4
      SV2:=0;                                    // je ventil SV2 zavřen
      StartOK:=0;            // a je vyresetován paměťový bit StartOK. Aby
                                          // mohl být použit v dalším cyklu
END_IF;
```

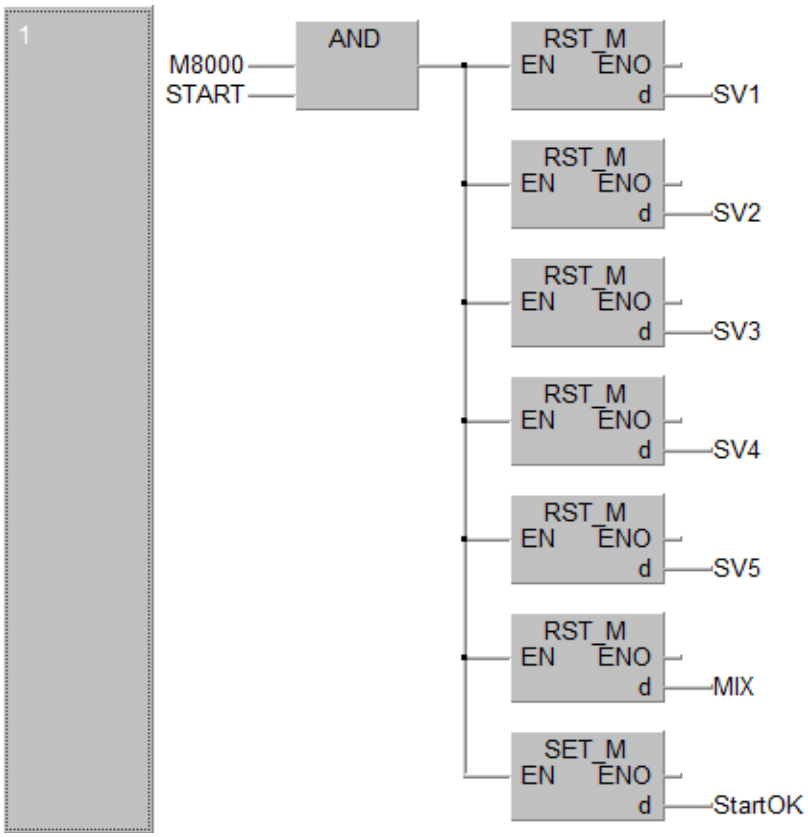
Výsledné řešení pomocí jazyka Melsec IL (Melsec instrukčního listu) by mohlo vypadat takto. Obsah těla (Body [Melsec IL]) POU s názvem Main [PRG]:

```
Network 1:
LD M8000        // specialní pameť M8000, ve které je po celou dobu uložena
AND START             // logická 1 v kombinaci se stlačeným tlačítkem START
RST SV1                                                 // vyresetuje výstup SV1
RST SV2                                                  // vyresetuje výstup SV2
RST SV3                                                 // vyresetuje výstup SV3
RST SV4                                                 // vyresetuje výstup SV4
RST SV5                                                 // vyresetuje výstup SV5
RST MIX                                                 // vyresetuje výstup MIX
SET StartOK   //  nastaví do StartOK logickou 1 pro spuštění POU Napouštění
```

Obsah těla (Body [Melsec IL]) POU s názvem Napousteni [PRG]:

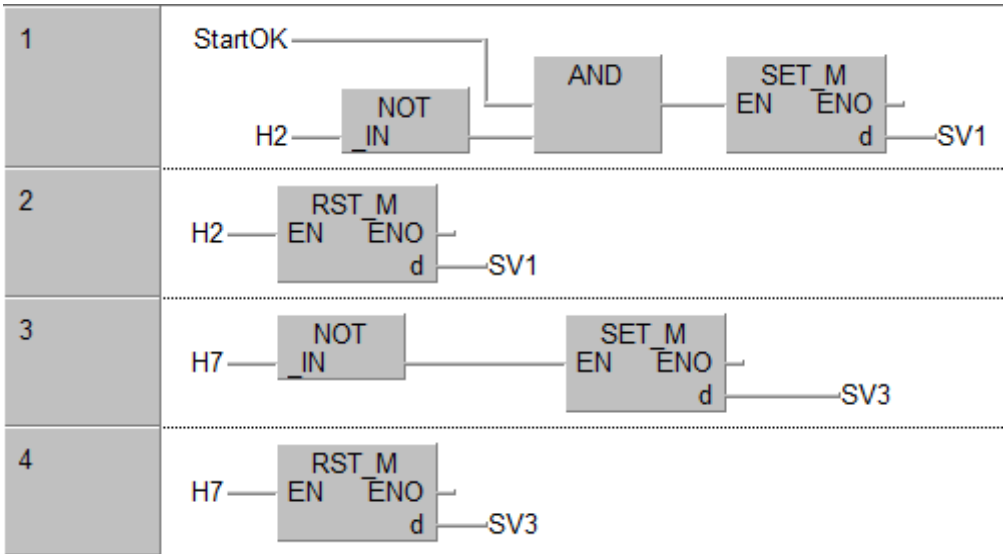
```
Network 1:
LD StartOK                                        // po splnění podmínky StartOK
ANI X6                                               // a při prázdné hladině H1
SET SV1                                                //  je otevřen ventil SV1

LD X6                                          // Po naplnění tanku do hladiny H1
RST SV1                                                  // je ventil SV1 zavřen
Network 2:
LDI X1                                 // Pokud není tank 3 naplněn po hladinu H7
SET SV3                                                 // je otevřen ventil SV3

LD X1                                        // Po naplnění tanku do hladiny H7
RST SV3                                            //   je   ventil   SV3   zavřen
Network 3:
LDI X4                                 // Pokud není tank 3 naplněn po hladinu H4
SET SV2                                                 // je otevřen ventil SV2

LD X4                                          // Po naplnění tanku do hladiny H4
RST SV2                                                  //  je ventil SV2 zavřen
RST StartOK      // vyresetován paměťový bit StartOK, aby mohl být použit v
                                                                  // dalším cyklu
```

Výsledné řešení pomocí jazyka FBD (Diagram funkčních bloků) by mohlo vypadat takto. Obsah těla (Body [FBD]) POU s názvem Main [PRG]:

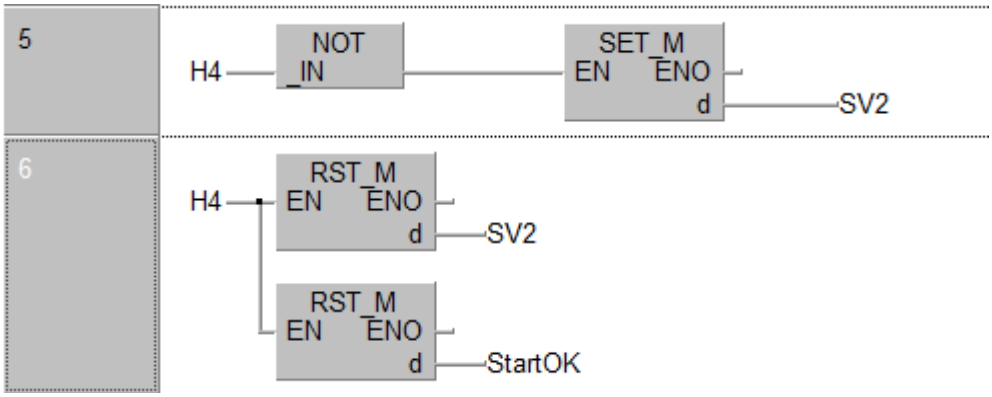


Obr. 36: Obsah těla POU Main v FBD

Obsah těla (Body [FBD]) POU s názvem Napousteni [PRG]:

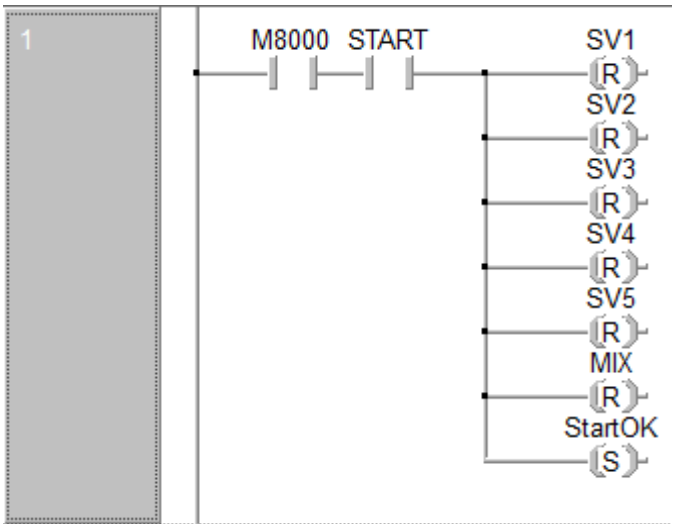


Obr. 37: Obsah těla POU Napousteni v FBD 1/2



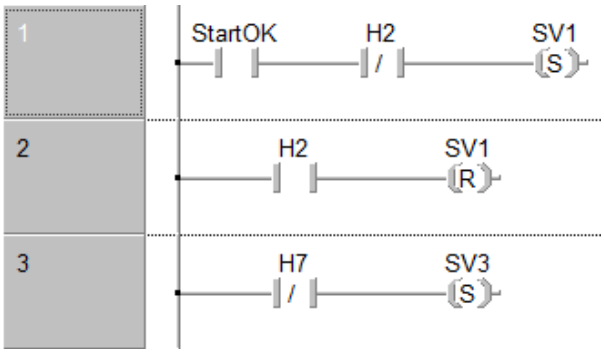
Obr. 38: Obsah těla POU Napousteni v FBD 2/2

Výsledné řešení pomocí jazyka LD (Kontaktních schémat) by mohlo vypadat takto. Obsah těla (Body [LD]) POU s názvem Main [PRG]:

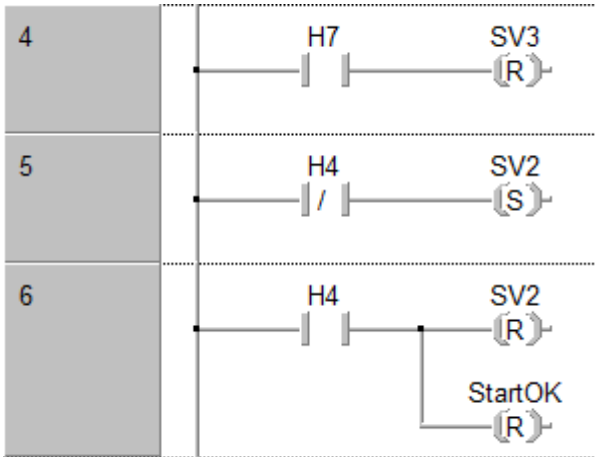


Obr. 39: Výsledné řešení POU Main v LD

Obsah těla (Body [LD]) POU s názvem Napousteni [PRG]:



Obr. 40: Výsledné řešení POU Napousteni v LD 1/2



Obr. 41: Výsledné řešení POU Napousteni v LD 2/2

5.2.3 Zadání 2. úlohy

Zadání 2. úlohy je rozšířením 1. úlohy a to tím způsobem, že po napuštění všech 3 tanků otevřeme ventil SV4, zároveň s ním bude spuštěn mixér Mix. Po vypuštění obsahů tanků do mísicí nádoby, bude ventil SV4 uzavřen. Následně bude otevřen ventil SV5 při stále spuštěném Mixu po dobu 15 vteřin. Po uplynutí této doby bude ventil SV5 uzavřen a Mix vypnut. Následně se bude čekat na opětovné stlačení tlačítka START.

5.2.4 Řešení 2. úlohy (IL, FBD, LD, ST, Melsec IL)

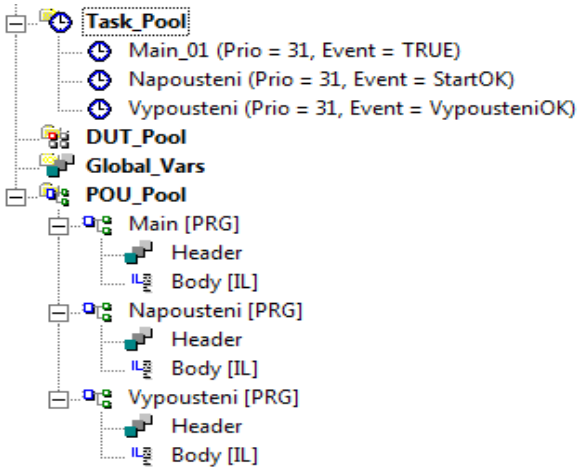
Pro řešení této úlohy si nejprve doplníme do tabulky Global_Vars, která bude stejná pro řešení úlohy jazyky Instruction List, FBD a Ladder diagram.

16	VAR_GLOBAL	VypousteniOK	M1	%MX0.1	BOOL	...	FALSE
17	VAR_GLOBAL	TIMER1C	TC0	%MX5.0	BOOL	...	FALSE

Tab. 12: Doplněné proměnné do tabulky Global_Vars z úlohy č. 1

V tabulce je uvedena nově doplněná paměťová proměnná VypousteniOK, která bude sloužit jako podmínka pro spuštění 3. podprogramu (POU Vypousteni) po splnění podmínek z podprogramu 2. (POU Napousteni). Dále je přidána ještě proměnná TIMER1C na adrese TC0, jedná se o 10ms časovač.

V následujícím obrázku je zobrazeno, jak byla rozšířena struktura projektu o novou POU s názvem Vypousteni, tato struktura je stejná pro všech 5 programovacích jazyků (IL, ST, FBD, LD, Melsec IL), pouze při vytváření projektu musíme vybrat, jaký z těchto jazyků budeme preferovat, označení tohoto jazyku bude potom zobrazeno v hranatých závorkách za Body v každé POU.



Tab. 13: Struktura projektu pro úlohu č. 2

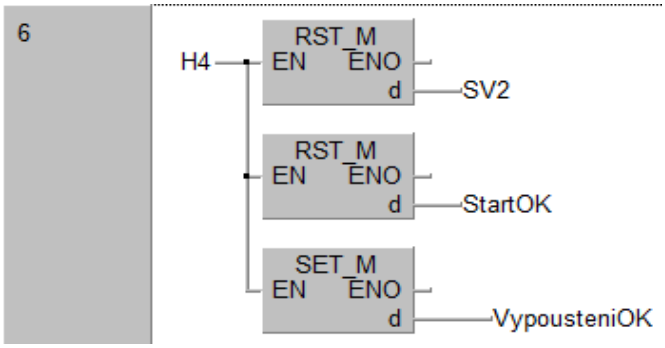
Následuje řešení úlohy č. 2 pomocí jazyka Instuction List. V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napousteni došlo k jedné změně a to, že tam byla přidána proměnná VypousteniOK, která bude spouštět novou POU vypouštění, níže je uvedeno jakým způsobem:

```
Network 3:
LDN H4                          // Pokud není tank 3 naplněn po hladinu H4
S SV2                                   //   je   otevřen   ventil   SV2.
LD H4                            //  Po  naplnění  tanku  do  hladiny  H4
R SV2                                                 //  je ventil SV2 zavřenR
StartOK         // vyresetována proměnná StartOK, pro požití v dalším cyklu
S VypousteniOK          //  nastaví do VypouštěníOK logickou 1 pro spuštění
                                                              // POU Vypoušťění
```

Následuje obsah těla (Body [IL]) POU s názvem Vypousteni[PRG]:

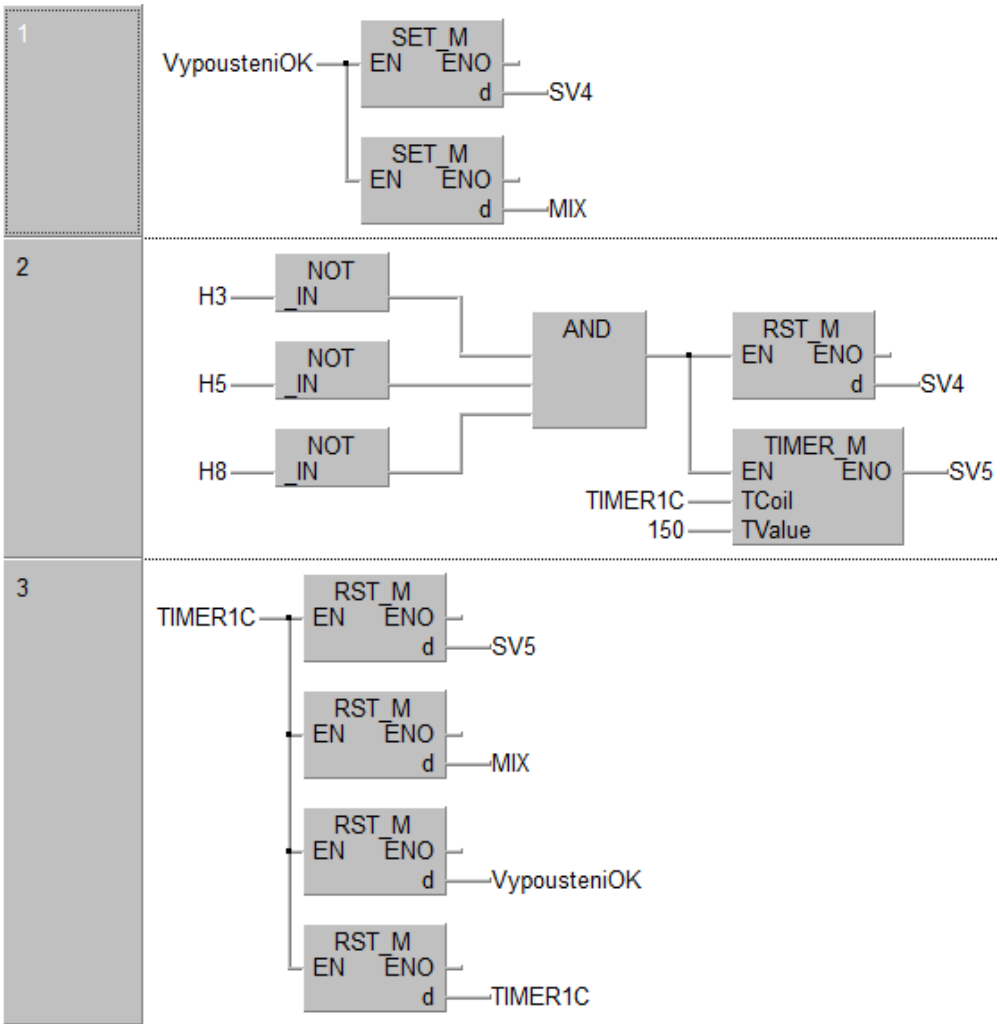
```
Network 1:
LD VypousteniOK                         // Po splnění podmínky VypouštěníOK
S SV4                                             // je otevřen ventil SV4
S MIX                                                   // je spuštěn MIX
Network 2:
LDN H3                                        // Po vyprázdnění všech tanků
ANDN H5
ANDN H8
R SV4                                              // je zavřen ventil SV4
TIMER_M     TIMER1C,150     // je spuštěn 10ms časovač TIMER1C s nastavenou
                                                          // hodnotou 150ms
S SV5                                             // je otevřen ventil SV5
Network 3:
LD TIMER1C                                    // po uplynuti nastavené doby
R SV5                                              // je zavřen ventil SV5
R MIX                                                      // vypnut Mix
R TIMER1C        // vyresetován TIMER1C, aby mohl být použit v dalším cyklu
R VypousteniOK          // vyresetována proměnná VypouštěníOK pro použití v
                                                          // dalšm cyklu
```

Následuje řešení úlohy č. 2 pomocí jazyka FBD. V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napousteni došlo k jedné změně a to, že tam byla přidána proměnná VypousteniOK, která bude spouštět novou POU vypouštění, níže je uvedeno jakým způsobem:



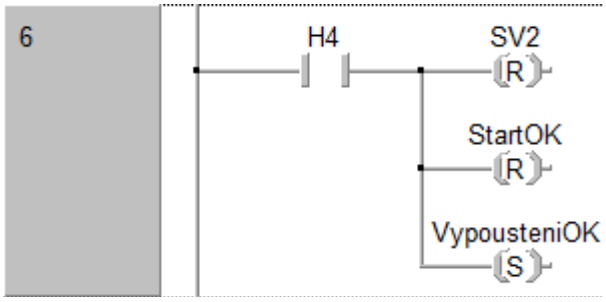
Obr. 42: Obsah těla POU Napousteni v FBD

Následuje obsah těla (Body [FBD]) POU s názvem Vypousteni[PRG]:



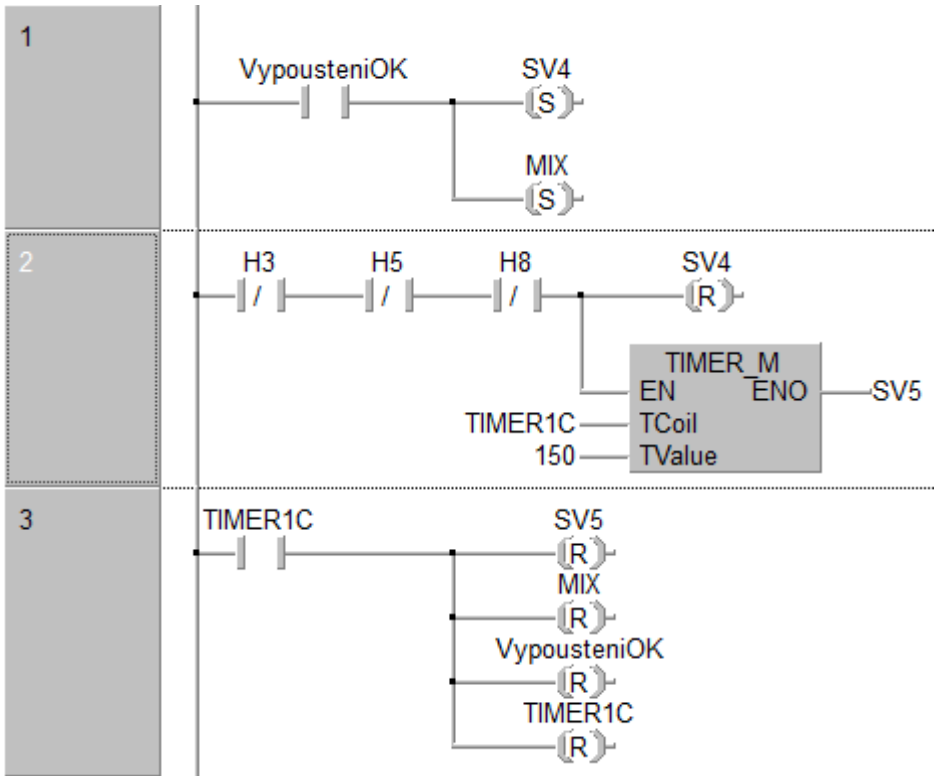
Obr. 43: Obsah těla POU Vypousteni v FBD

Následuje řešení úlohy č. 2 pomocí jazyka LD. V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napousteni došlo k jedné změně a to, že tam byla přidána proměnná VypousteniOK, která bude spouštět novou POU vypouštění, níže je uvedeno jakým způsobem:



Obr. 44: Obsah těla POU Napousteni v LD

Následuje obsah těla (Body [LD]) POU s názvem Vypousteni[PRG]:



Obr. 45: Obsah těla POU Vypousteni v LD

Následuje řešení úlohy č. 2 pomocí jazyka ST. Pro řešení této úlohy jsme přidali do tabulky Global_Vars z 1. úlohy následující proměnné:

16	VAR_GLOBAL	VypousteniOK	M1	%MX0.1	BOOL	...	FALSE
17	VAR_GLOBAL	TimerSet	M2	%MX0.2	BOOL	...	FALSE

Tab. 14: Doplněné proměnné do tabulky Global_Vars z úlohy č. 1

V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napousteni došlo k jedné změně a to, že tam byla přidána proměnná VypousteniOK, která bude spouštět novou POU vypouštění, níže je uvedeno jakým způsobem:

```
IF StartOK AND NOT X6 THEN      // Po splnění podmínky StartOK a při prázdné
      SV1:=1;                           // hladině H2 je otevřen ventil SV1
END_IF;

IF X6 THEN                          // Pokud není tank 2 naplněn po hladinu H2
      SV1:=0;                                    // je otevřen ventil SV1
END_IF;

IF  NOT X1 THEN                            // Po naplněni tanku do hladiny H7
      SV3:=1;                                    // je ventil SV3 zavřen
END_IF;

IF  X1 THEN                                // Po naplněni tanku do hladiny H7
      SV3:=0;                                    // je ventil SV3 zavřen
END_IF;

IF NOT X4 THEN                      // Pokud není tank 3 naplněn po hladinu H4
      SV2:=1;                                    // je otevřen ventil SV2
END_IF;

IF X4 THEN                                 // Po naplnění tanku do hladiny H4
      SV2:=0;                                    // je ventil SV2 zavřen
      StartOK:=0;             // a je vyresetován paměťový bit StartOK. Aby
                                          // mohl být použit v dalšim cyklu
      VypousteniOK:=1;  //  nastaví do VypouštěníOK logickou 1 pro spuštění
                                            // POU Vypouštění
END_IF;
```

Pro vyřešení POU Napousteni musíme doplnit do její hlavičky (Header) následující:

0	VAR	Timer	TON	...	
1	VAR_CONSTANT	Cas	TIME	...	T#15s

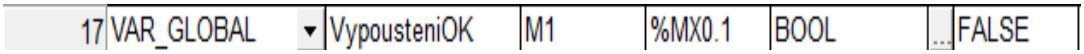
Tab. 15: Tabulka Header v POU Vypousteni

Následuje obsah těla (Body [ST]) POU s názvem Vypousteni[PRG]:

```
IF VypousteniOK THEN                      // Po splnění podmínky VypouštěníOK
      SV4:=1;                                    // je otevřen ventil SV4
      MIX:=1;                                          // je spuštěn MIX
END_IF;

IF NOT X2 AND NOT X3 AND NOT X5 THEN          // Po vyprázdnění všech tanků
   SV4:=0;                                       // je uzavřen ventil SV4
   SV5:=1;                                       // je otevřen ventil SV5
   TimerSet:=1;             // je nastavena logická 1 do proměnné TimerSet,
                                            // pro spuštění Timeru
END_IF;

   Timer(IN:=TimerSet,PT:=Cas);          // po splněni podminky TimerSet je
                                    // spuštěn časovač na nastavenou dobu
   IF Timer.Q THEN                 // pokud v časovači uplyne nastavená doba
```

```
   SV5:=0;                                                    // zavřen ventil SV5
   MIX:=0;                                                          // vypnut Mix
   VypousteniOK:=0;                        // vyresetována proměnná VypouštěníOK
   TimerSet:=0;// vyresetována proměnná TimerSet, pro vyresetování časovače
END_IF;
```

Následuje řešení úlohy č. 2 pomocí jazyka Melsec IL. Pro řešení této úlohy jsme přidali do tabulky Global_Vars z 1. úlohy následující proměnnou:



Obr. 46: Doplněné proměnné do tabulky Global_Vars z úlohy č. 1

V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napousteni došlo k jedné změně a to, že tam byla přidána proměnná VypousteniOK, která bude spouštět novou POU vypouštění, níže je uvedeno jakým způsobem:

```
Network 3:
LDI X4                              // Pokud není tank 3 naplněn po hladinu H4
SET SV2                                                 // je otevřen ventil SV2

LD X4                                        // Po naplnění tanku do hladiny H4
RST SV2                                                 //  je ventil SV2 zavřen
RST StartOK      // vyresetován paměťový bit StartOK, aby mohl být použit v
                                                                      // dalším cyklu
SET VypousteniOK     // nastaví do VypouštěníOK logickou 1 pro spuštění POU
                                                                    // Vypouštění
```

Následuje obsah těla (Body [Melsec IL]) POU s názvem Vypousteni[PRG]:

```
Network 1:
LD VypousteniOK                           // Po splnění podmínky VypouštěníOK
SET SV4                                                 // je otevřen ventil SV4
SET MIX                                                       // je spuštěn MIX

LDI X2                                          // Po vyprázdnění všech tanků
ANI X3
ANI X5
RST SV4                                                 // je uzavřen ventil SV4
SET SV5                                                 // je otevřen ventil SV5

LD SV5                                             // je-li otevřen ventil SV5
OUT T0 K150        // spustí se 10ms časovač T0 s nastavenou hodnotou 150ms

LD T0                                          // po uplynutí doby v časovači
RST SV5                                                  // je zavřen ventil SV5
RST MIX                                                        // je vypnut Mix
RST VypousteniOK                    // je vyresetována proměnná VypouštěníOK
RST T0                                            // je vyresetován časovač T0
```

5.2.5 Zadání 3. úlohy

Zadání 3. úlohy je rozšířením 2. úlohy a to tím způsobem, že během cyklu vypouštění, který trvá 15 vteřin, bude rovnou spuštěno napouštění, tak aby se neztrácel čas a celý koloběh fungoval automaticky. Úlohu si rozdělíme na 4 podprogramy, a to Main, ve kterém budou po spuštění vyresetovány všechny výstupy, dále Napouštění, Mix a nakonec Vypouštění.

5.2.6 Řešení 3. úlohy (IL, FBD, LD, ST, Melsec IL)

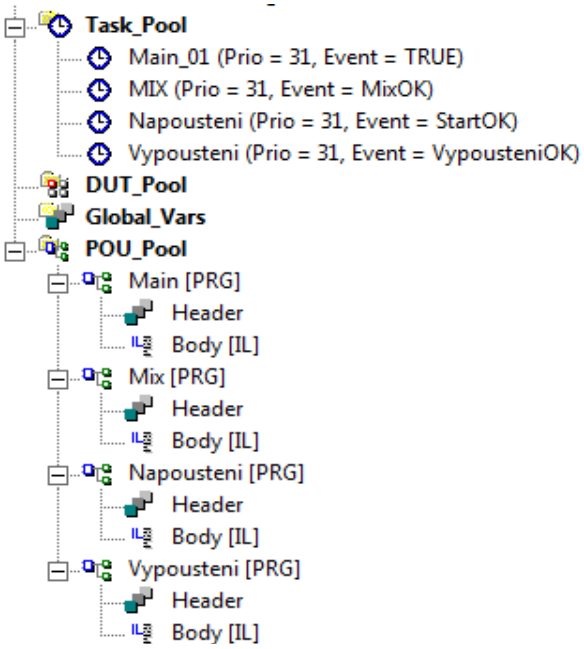
Pro řešení této úlohy si nejprve doplníme do tabulky Global_Vars, která bude stejná pro řešení úlohy jazyky Instruction List, FBD a Ladder diagram.

18	VAR_GLOBAL	MixOK	M2	%MX0.2	BOOL	FALSE

Tab. 16: Doplněné proměnné do tabulky Global_Vars z úlohy č. 2

V tabulce je uvedena nově doplněná paměťová proměnná MixOK, která bude sloužit jako podmínka pro spuštění podprogramu (POU Mix) po splnění podmínek z podprogramu (POU Napousteni).

V následujícím obrázku je zobrazeno, jak byla rozšířena struktura projektu o novou POU s názvem Mix, tato struktura je stejná pro všech 5 programovacích jazyků (IL, ST, FBD, LD, Melsec IL), pouze při vytváření projektu musíme vybrat, jaký z těchto jazyků budeme preferovat, označení tohoto jazyku bude potom zobrazeno v hranatých závorkách za Body v každé POU.



Obr. 47: Struktura projektu pro úlohu č. 3

Následuje řešení úlohy č. 3 pomocí jazyka Instuction List.

V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napousteni došlo k jedné změně a to, že tam byla nahrazena proměnná VypousteniOK proměnnou MixOK, která bude spouštět novou POU Mix, níže je uvedeno jakým způsobem:

```
Network 3:
LDN H4                              // Pokud není tank 3 naplněn po hladinu H4
S SV2                                                  // je otevřen ventil SV2
LD H4                                          // Po naplnění tanku do hladiny H4
R SV2                                                  //  je ventil SV2 zavřen
R StartOK        // vyresetován paměťový bit StartOK, aby mohl být použit v
                                                                    // dalším cyklu
S MixOK                //  nastaví do MixOK logickou 1 pro spuštění POU Mix
```

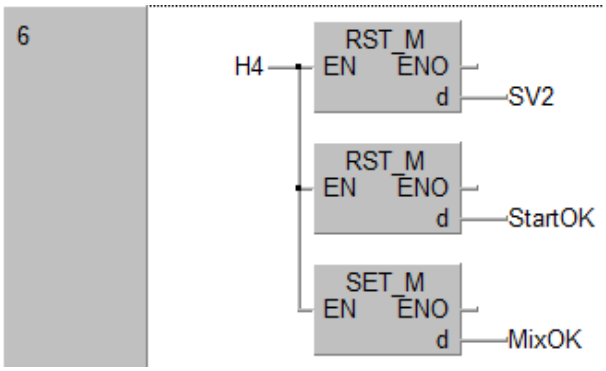
Následuje obsah těla (Body [IL]) POU s názvem Mix [PRG]:

```
Network 1:
LD MixOK                                              // Po splněni podmínky MixOK
S SV4                                                     // je otevřen ventil SV4
S MIX                                                              // spuštěn Mix

LDN H3                                               // Po vyprázdnění všech tanků
ANDN H5
ANDN H8
R SV4                                                  // je ventil SV4 uzavřen
R MixOK                        // vyresetuje MixOK pro použití v dalším cyklu
S VypousteniOK  // do VypouštěníOK je nastavena logická 1, pro spuštění POU
                                                                  // Vypouštění
```

Následuje obsah těla (Body [IL]) POU s názvem Vypouštění [PRG]:

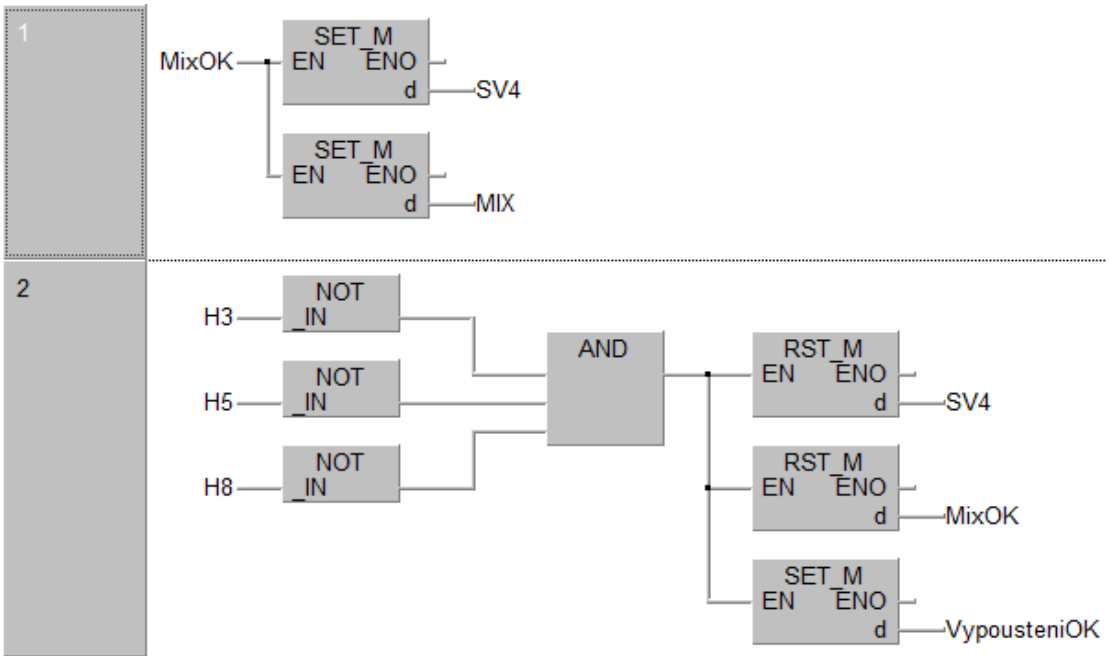
```
Network 1:
LD VypousteniOK                           // Po splnění podmínky VypouštěníOK
S StartOK                                 // je znovu spuštěna POU Napouštění
TIMER_M     TIMER1C,150     // je spuštěn 10ms časovač TIMER1C s nastavenou
                                                          // hodnotou 150ms
S SV5                                                  // je otevřen ventil SV5
Network 2:
LD TIMER1C                                           // po uplynuti nastavené doby
R SV5                                                        // zavren ventil SV5
R MIX                                                         // je vypnut Mix
R TIMER1C        // vyresetován TIMER1C, aby mohl být použit v dalším cyklu
R VypousteniOK         // vyresetován paměťový bit VypouštěníOK pro použití
                                                                // v dalším cyklu
```

Následuje řešení úlohy č. 3 pomocí jazyka FBD. V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napousteni došlo k jedné změně a to, že tam byla nahrazena proměnná VypousteniOK proměnnou MixOK, která bude spouštět novou POU Mix, níže je uvedeno jakým způsobem:



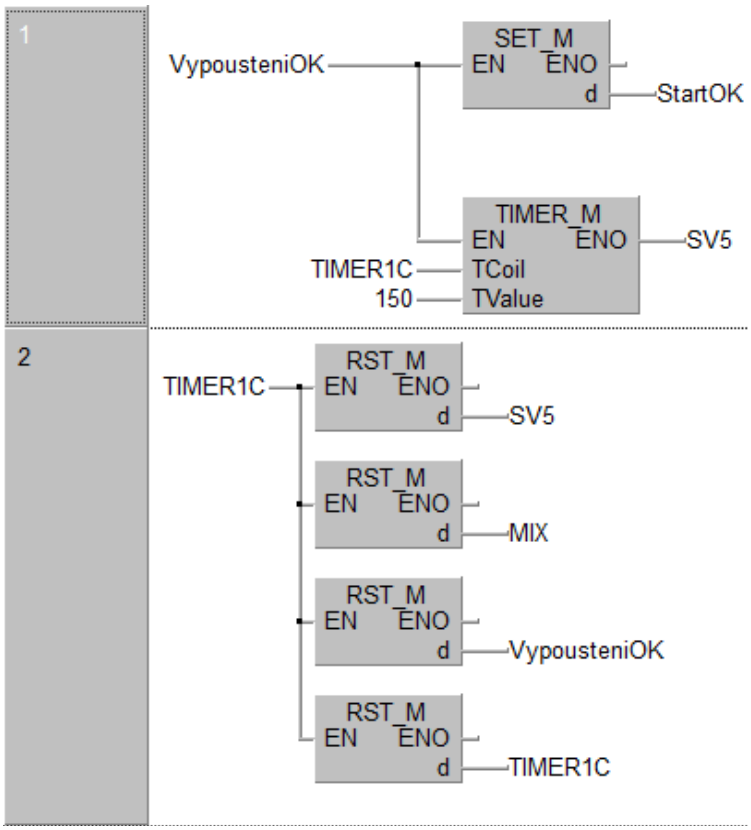
Obr. 48: Nahrazení proměnné VypousteniOK proměnnou MixOK

Následuje obsah těla (Body [FBD]) POU s názvem Mix [PRG]:



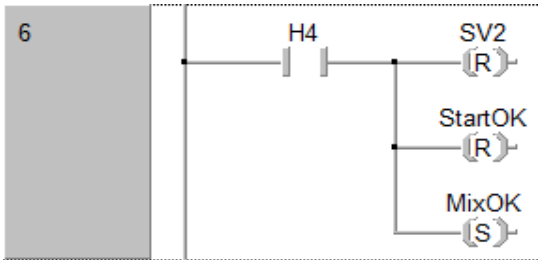
Obr. 49: Obsah těla POU Mix v FBD

Následuje obsah těla (Body [FBD]) POU s názvem Vypouštění [PRG]:



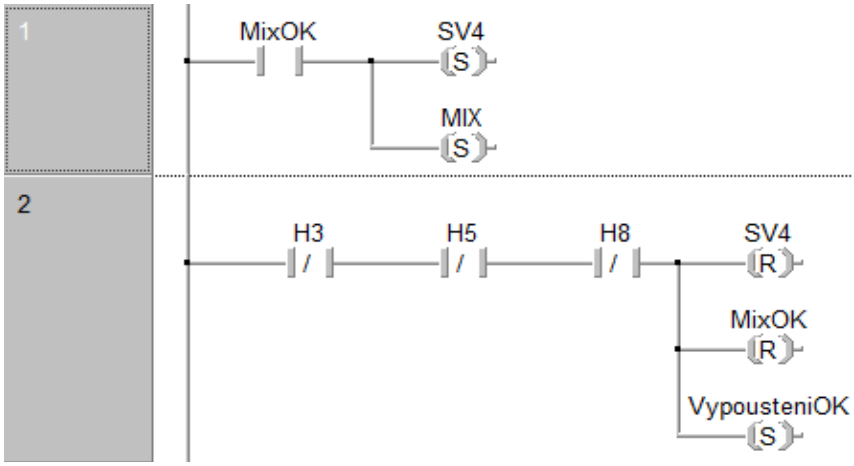
Obr. 50: Obsah těla POU Vypouštění v FBD

Následuje řešení úlohy č. 3 pomocí jazyka Ladder diagram. V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napousteni došlo k jedné změně a to, že tam byla nahrazena proměnná VypousteniOK proměnnou MixOK, která bude spouštět novou POU Mix, níže je uvedeno jakým způsobem:



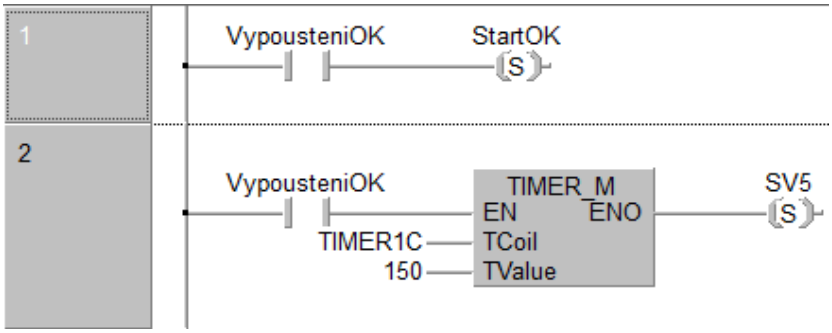
Obr. 51: Nahrazení proměnné VypouštěníOK proměnnou MixOK

Následuje obsah těla (Body [LD]) POU s názvem Mix [PRG]:

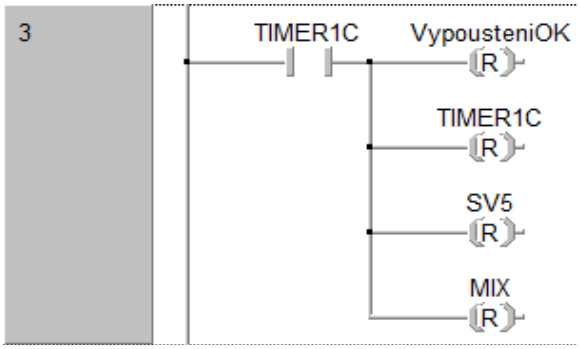


Obr. 52: Obsah těla POU Mix v LD

Následuje obsah těla (Body [LD]) POU s názvem Vypouštění [PRG]:



Obr. 53: Obsah těla POU Vypouštění v LD 1/2



Obr. 54: Obsah těla POU Vypouštění v LD 2/2

Následuje řešení úlohy č. 3 pomocí jazyka ST. Pro toto řešení jsme přesunuli časovač Timer z hlavičky POU Vypouštění do tabulky globálních proměnných Global_Vars.

18	VAR_GLOBAL	MixOK	M2	%MX0.2	BOOL	...	FALSE
19	VAR_GLOBAL	Timer			TON	...	

Obr. 55: Doplněné proměnné do tabulky Global_Vars z úlohy č. 2

Dále jsme do hlavičky (Header) POU Vypouštění přidali proměnnou PocitadloCyklu.

1	VAR	PocitadloCyklu	INT	...	0

Obr. 56: Doplněné proměnné do tabulky Header v POU Vypousteni

V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napoušteni došlo k jedné změně a to, že tam byla nahrazena proměnná VypousteniOK proměnnou MixOK, která bude spouštět novou POU Mix, níže je uvedeno jakým způsobem:

```
IF X4 THEN                                  // Po naplnění tanku do hladiny H4
      SV2:=0;                                          // je ventil SV2 zavřen
      StartOK:=0;    // a je vyresetována proměnná StartOK pro dalším cyklus
      MixOK:=1;         //  nastaví do MixOK logickou 1 pro spuštění POU Mix
END_IF;
```

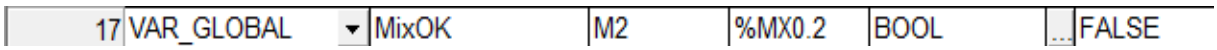
Následuje obsah těla (Body [ST]) POU s názvem Mix [PRG]:

```
IF MixOK THEN                                    // Po splnění podmínky MixOK
      MIX:=1;                                               // je spuštěn Mix
      SV4:=1;                                          // je otevřen ventil SV4
END_IF;

IF NOT X5 AND NOT X3 AND NOT X2 THEN          // Po vyprázdnění všech tanků
   SV4:=0;                                             // je ventil SV4 uzavřen
   MIX:=0;                                               // je vypnut Mixér MIX
   MixOK:=0;                                // je vyresetován paměťový bit MixOK
   VypousteniOK:=1;
END_IF;
```

Následuje obsah těla (Body [ST]) POU s názvem Vypouštění [PRG]:

```
Timer(IN:=TimerSet,PT:=Cas); // po splnění podminky TimerSet je spuštěn
                                                  // časovač na nastavenou dobu
IF VypousteniOK THEN                        // po splnění podmínky VypouštěníOK
      StartOK:=1;                                 // je spuštěna POU Napouštění
      SV5:=1;                                          // je otevřen ventil SV5
      MIX:=1;                                                // je zapnut Mix
      TimerSet:=1;           // do proměnné TimerSet je nastavena logická 1
END_IF;

IF TimerSet AND Timer.Q THEN  // je-li v TimerSet logická 1 a doběhl-li čas
      SV5:=0;                                 // v časovači je zavřen ventil SV5
      MIX:=0;                                                // je vypnut Mix
      TimerSet:=0;                         // je vyresetována proměnná TimerSet
      PocitadloCyklu:=PocitadloCyklu+1;  // do PocitadlaCyklu je k aktuální
END_IF;                                                  //  hodnotě přičtena 1

IF PocitadloCyklu >= 1 THEN   // je-li v PocitadleCyklu hodnota větší než 1
      PocitadloCyklu:=0;                  // je toto počítadlo vyresetováno
      VypousteniOK:=0;                      // je vyresetováno VypouštěníOK
END_IF;
```

Následuje řešení úlohy č. 3 pomocí jazyka Melsec IL. Pro toto řešení jsme museli do tabulky globálních proměnných Global_Vars následující proměnnou:



Obr. 57: Doplněné proměnné do tabulky Global_Vars z úlohy č. 2

V POU Main nedošlo k žádné změně, je tedy naprosto stejná s POU Main z 1. úlohy. V POU Napousteni došlo k jedné změně a to, že tam byla nahrazena proměnná VypousteniOK proměnnou MixOK, která bude spouštět novou POU Mix, níže je uvedeno jakým způsobem:

```
Network 3:
LDI X4                              // Pokud není tank 3 naplněn po hladinu H4
SET SV2                                          // je otevřen ventil SV2
LD X4                                //  Po  naplněni  tanku  do  hladiny  H4
RST SV2                                          //  je ventil SV2 zavřen
RST StartOK      // vyresetován paměťový bit StartOK, aby mohl být použit v
                                                                // dalším cyklu
SET MixOK               // nastaví do MixOK logickou 1 pro spuštění POU Mix
```

Následuje obsah těla (Body [Melsec IL]) POU s názvem Mix [PRG]:

```
Network 1:
LD MixOK                                          // Po splněni podmínky MixOK
SET SV4                                              // je otevřen ventil SV4
SET MIX                                                       // spuštěn Mix

LDI X5                                           // Po vyprázdnění všech tanků
ANI X3
ANI X2
RST SV4                                              // je ventil SV4 uzavřen
RST SV5                                                 // zavren ventil SV5
SET VypousteniOK //do VypouštěníOK nastavena 1, pro spuštění POU Vypousteni
```

Následuje obsah těla (Body [Melsec IL]) POU s názvem Vypouštění [PRG]:

```
Network 1:
LD VypousteniOK                          // Po splnění podmínky VypouštěníOK
SET StartOK                              // je znovu spuštěna POU Napouštění
SET SV5                                          // je otevřen ventil SV5

LD SV5                                        // je-li otevřen ventil SV5
OUT T0 K150                        // je spuštěn 10ms časovač T0 po dobu 15s
LD T0                                     // po uplynutí času v časovači
RST SV5                                       // je zavřen ventil SV5
RST MIX                                             // je vypnut Mix
RST T0                                       // vyresetován časovač T0
RST VypousteniOK     // vyresetována proměnná VypouštěníOK pro další cyklus
```

ZÁVĚR

Cílemi této bakalářské práce bylo seznámit se s modely soustav EDU-mod, programovatelným automatem Mitsubishi FX3U-32M, vývojovým prostředím GX IEC-Developer 7.01 a následně pro tyto modely navrhnout úlohy a jejich řešení realiozvat pomocí výše uvedených pomůcek. Všechny úlohy měly být řešeny 5 programovacími jazyky a to: Instruction List (jazyku instrukcí), Melsec IL (Melsec jazyku instrukcí), Sructured text (strukturovaném textu), Function Block Diagram (diagramu funkčních bloků) a Ladder Diagramu (jazyku kontaktních schémat).

V první kapitole s názvem Modely soustav EDU-mod byly popsány jednotlivé modely soustav, kterými se tato práce zabývala. Jednalo se o model Hydraulické posuvové jednotky a model Mísicí jednotky. U obou modelů byly stručně popsány vstupní a výstupní proměnné a také jejich celková funkce.

V druhé kapitole s názvem Mitsubishi FX3U-32M byl popsán automat, který byl používán pro řízení modelů soustav EDU-mod. Ve specifikaci jsou popsány hardwarové parametry automatu. Dále je zde umístěn nákres základní jednotky FX3U s popisem jenotlivých částí.

Třetí kapitola je zameřena na vývojové prostředí GX IEC-Developer 7.01, v této části jsou popsány jednotlivé prvky panelu nástrojů, které se při programování nejvíce využívají. Následně je zde popsáno jak se vytváří nový projekt včetně popisu jendotlivých kroků. Dále jsou popsány jednotlivé části ve vytvořené struktuře projektu. Je zde ukázáno, jak se dá využívat doplněk tohoto prostředí a to virtuální automat v GX Simulatoru, jak lze z tohoto prostředí měnit režim PLC a na závěr této kapitoly jsou zde uvedeny jednotlivé prvky, které byly použity při programování výsledných úloh.

Poslední kapitola je již zaměřena na podstatu této bakalářské práce, kterou bylo navrhnout pro každý z uvedených modelů 3 úlohy, které budou mít různou obtížnost a každá další úloha bude rozšířením předchozí úlohy. V této kapitole jsou tedy uvedeny výsledná zadání úloh s jejich podrobným popisem. Dále je zde popsáno, jak jsou jednotlivé modely propojeny s automatem a také popis jednotlivých vstupů a výstupů. Za zadáním každé úlohy je výsledné řešení ve všech 5 programovacích jazycích, které jsou zmíněny výše. Jejich funkčnost byla ověřena řízením pomocí PLC a proběhla úspěšně.

SEZNAM POUŽITÉ LITERATURY

[1] Ing. Kohout, Luděk. Edumat.cz – Autorizované kurzy Teco a.s., pomůcky do odborných učeben a laboratoří [online].[cit. 15. prosince 2009]
Dostupné z: <http://www.edumat.cz/produkty.php?produkt=edumod>

[2] Ing. Kohout, Luděk. Edumat.cz – Autorizované kurzy Teco a.s., pomůcky do odborných učeben a laboratoří [online].[cit. 15. prosince 2009]
Dostupné z: <http://www.edumat.cz/produkty.php?produkt=suport>

[3] Ing. Kohout, Luděk. Edumat.cz – Autorizované kurzy Teco a.s., pomůcky do odborných učeben a laboratoří [online].[cit. 16. prosince 2009]
Dostupné z: <http://www.edumat.cz/produkty.php?produkt=mixer>

[4] Programmable logic controller [online].[cit. 3. února 2010]
Dostupné z: <http://en.wikipedia.org/wiki/Programmable_logic_controller>

[5] Mitsubishi. Mitsubishi electric [online].[cit. 4. ledna 2010]]
Dostupné z: <http://www.mitsubishi-automation-cz.com/service/download.html>

SEZNAM PŘÍLOH

Příloha č. 1 – Tento dokument v elektronické podobě.
Příloha č. 2 – Zdrojové kódy výsledných řešení úloh.