Jihočeská Univerzita v Českých Budějovicích

Přírodovědná fakulta

Bakalářská práce

Tvorba výukového panelu pro předmět Automatizace realizovaný s PLC firmy Amit

Autor: Tomáš Tunka

Vedoucí práce: Ing. Michal Šerý

České Budějovice 2020

Bibliografické údaje:

Tunka T., 2020, Tvorba výukového panelu pro předmět Automatizace realizovaný s PLC firmy Amit [Creation of a education panel for the subject Automation realized with PLC Amit company, Bc. Thesis, in Czech] – 47 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace:

Cílem bakalářské práce je navrhnout a sestavit výukový panel pro předmět Automatizace, který obsahuje PLC firmy Amit. K tomuto výukovému panelu je dále třeba vypracovat sadu úloh demonstrující vlastnosti PLC řídících systémů.

Klíčová slova:

PLC, Amit, Automatizace, výukový panel

Abstract:

The aim of the bachelor's thesis is to design and assemble an educational panel for the subject Automation, which contains a PLC from Amit company. In addition to this educational panel, it is necessary to develop a set of tasks demonstrating the properties of PLC control systems.

Keywords:

PLC, Amit, Automation, educational panel

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V českých Budějovicích dne 18.5.2020

……...

Tomáš Tunka

Obsah

1. Úvod

Cílem práce je navrhnout a sestavit výukový panel pro předmět Automatizace obsahující PLC firmy Amit a sepsat sadu demonstračních úloh. Panel má pomoci studentům zejména s porozuměním programování PLC systémů, a to jak se základy, tak s řešením pokročilejších problémů. V první řadě jsem se zaměřil na zvládnutí booleovy algebry, čítačů, časovačů a jednoduchých pamětí (např. klopných obvodů). Později jsem přidal práci s analogovými proměnnými, regulacemi a další.

Teoretická část obsahuje popis a funkce PLC systémů a jejich použití. Dále jsem psal o firmě Amit a o řídících systémech této firmy. Popisuji zde také softwary, které firma AMiT používá k programování jejich řídících systémů.

Teoretická část obsahuje:

- Charakteristika PLC řídících systémů
- Řídící systémy firmy AMiT
- Programovací software DetStudio a ViewDet

V praktické části popisuji funkce a části výukového panelu, které jsou pevně vestavěné v samotném panelu (displej, řídící terminál atd.). Dále jsem vymyslel a sepsal několik demonstračních úloh na kterých studenti řeší reálné problémy. U každé úlohy je také jedno z možných řešení s vysvětlením, jak mnou navrhnuté řešení funguje a jak se k němu dobrat.

Praktická část obsahuje:

- Konstrukce a komponenty výukového panelu
- Popis a obsah demonstračních úloh

2. Charakteristika PLC řídicích systémů

2.1 Úvod do PLC

Zařízení PLC (Programmable Logic Controler), v češtině často označované jako programovatelné automaty, patří již dlouhodobě k základům automatického měření a regulace různých aplikací a procesů. V současnosti již jde o jednoduché modulární a lehce programovatelné jednotky v podobě několika vzájemně propojených "krabiček" s mnoha vstupy a výstupy pro snadné připojení senzorů, displejů, spínačů a tlačítek, motorů a různých dalších přístrojů a zařízení. Funkce celého PLC i ovládání připojených prvků je řízeno uloženým programem, který lze snadno vytvořit pomocí výrobcem dodávaného grafického vývojového softwaru pro běžná PC a operační systém Windows. Ten umožňuje mimo programování i průběžnou grafickou simulaci a po připojení PLC k PC i reálné zkoušení a testování.

Zatímco při použití různých typů počítačů (ať obyčejných nebo průmyslových) pro potřeby regulace je vyžadována znalost některého z programovacích jazyků a struktury použitého procesoru, programování současných PLC se ve vývojovém softwaru provádí pomocí vkládání již připravených funkcí reprezentovaných ikonami na obrazovce a jejich propojování tažením myši bez nutné znalosti, co je uvnitř PLC. Mechanická instalace pak spočívá pouze u v jednoduchém nasazení na DIN lištu a připojení vodičů do šroubovacích svorek představujících jednotlivé vstupy a výstupy.

2.2 Historický vývoj PLC

V automatizační technice jsou PLC využívány již zhruba 30 let. Jejich typickou vlastností je programovatelnost na úrovni blízké mentalitě konstruktéra nebo projektanta. Původně byly PLC určeny pro řízení strojů, jako náhrada za pevnou reléovou logiku. Tomu odpovídal i programovací jazyk kontaktních (reléových) schémat. Jazyky prvních PLC disponovaly několika příkazy (typicky 8 nebo 16), které byly ekvivalentní spínacímu a rozpínacímu kontaktu, paralelnímu a sériovému řazení, cívce, obvodům paměti, čítače a časovače. Dnes je pro každý programovatelný automat k dispozici několik typů jazyků: kromě jazyka kontaktních schémat to bývá jazyk logických schémat, jazyk mnemokódů nebo jiný textový jazyk, nově i jazyk sekvenčního programování. Dnešní programovací jazyky jsou podstatně bohatší – bohužel se poněkud vzdálily mentalitě konstruktérů a projektantů, takže vznikla samostatná profese „programátor PLC“. Sjednocení programovacích jazyků a vývojových systémů pro PLC je cílem nové mezinárodní normy IEC 61131-3. Programovatelnost a variabilnost výstavby poskytuje PLC jejich pověstnou univerzálnost a přizpůsobivost. Již neplatí, že PLC řešil jen logické úlohy, zatímco ke zpracování analogových veličin se používaly specializované regulátory. PLC dnes zvládne oba typy úloh (a mnoho dalších).

2.3 Struktura a programování PLC

Struktura PLC se mírně liší od běžných počítačů. Jelikož byli PLC původně vytvořeny tak, aby programování bylo co nejjednodušší a intuitivní, je tomu přizpůsobena i struktura PLC.

2.3.1 Centrální jednotka

Poskytuje programovatelnému automatu inteligenci. Realizuje soubor instrukcí a systémových služeb, zajišťuje i základní komunikační funkce s vlastními i vzdálenými moduly, s nadřízeným systémem a s programovacím přístrojem. Paměťový prostor, který poskytuje uživateli je obvykle rozdělen na části. Prvá je určena pro uložení uživatelského program (PLC programu), datových bloků a tabulek. Její obsah se zadává v edičním režimu a během vykonávání programu se obvykle nemění. Druhá část je operační (zápisník). Jsou v ní lokalizovány uživatelské registry, čítače a časovače, obrazy vstupů a výstupů, komunikační, časové a jiné systémové proměnné (systémové registry). Obsah operační části se dynamicky mění působením uživatelského a systémového programu.

Centrální jednotky současných programovatelných automatů obsahují mikroprocesor, mikrořadič nebo specializovaný řadič, zaměřený na rychlé provádění instrukcí. Jeho programem (systémovým programem) jsou realizovány všechny funkce, které má uživatel k dispozici, tj. kompletní soubor instrukcí programovatelného automatu, jeho systémové služby, časové a komunikační funkce. Typické je použití instrukcí a příkazů jazyka programovatelného automatu, který je přizpůsoben převažujícím úlohám a způsobu myšlení typického uživatele a programátora PLC.

2.3.2 Soubor instrukcí PLC

Protože programovatelné automaty byly původně určeny k realizaci logických úloh a k náhradě pevné logiky, nechybějí v žádném PLC instrukce pro základní logické operace s bitovými operandy. Tj. sejmutí pravdivostní hodnoty adresovaného bitu, operace logického součtu a součinu, negace, výlučného součtu a jiných kombinačních logických funkcí, instrukce pro realizaci paměťových funkcí a klopných obvodů, pro zápis výsledku a mezivýsledku na adresované místo, ale i instrukce čítačů, časovačů apod.

Současné PLC nabízejí instrukční soubor podstatně bohatší. V souboru instrukcí vyspělých PLC obvykle nechybí ani instrukce pro aritmetiku a operace s čísly (někdy jen nejzákladnější, např. sčítání, odčítání a porovnávání, jindy kompletní knihovny pro výpočty s pevnou nebo plovoucí řádovou čárkou), logické instrukce s číselnými operandy (paralelní operace s operandem v délce byte, slova nebo delším) a přenosy dat. Obvyklé jsou instrukce pro organizaci programu (např. skoky v programu, volání podprogramů a návraty).

2.3.3 Specializované instrukce

Některé PLC poskytují i velmi výkonné instrukce pro komplexní operace, např. pro realizaci regulátorů a jejich automatické seřizování, pro fuzzy logiku a fuzzy regulaci, pro operace s daty a s datovými strukturami, pro realizaci ucelených funkčních bloků apod. Tyto specializované instrukce usnadňují programování (nabízejí již hotové ucelené funkce nebo jejich "prefabrikáty"), zvyšují však i výpočetní výkon PLC.

2.4 Vykonávání programu PLC

Program je soubor instrukcí, které jsou v řídícím systému postupně vykonávány za účelem realizace požadovaného algoritmu řízení.

2.4.1 Uživatelský program, cyklická aktivace

Program PLC je posloupnost instrukcí a příkazů jazyka. Typickým režimem jeho aktivace je cyklické vykonávání v programové smyčce. Na rozdíl od jiných programovatelných systémů se programátor PLC nemusí starat o to, aby po konci programu vrátil jeho vykonávání opět na začátek – zajistí to již systémový program. Naopak každé dlouhodobé setrvání programu v programové smyčce je "fatální chybou" a systém jej hlásí jako "překročení doby cyklu".

Cyklické vykonávání programu je schematicky znázorněno na obr. 1. Vždy po vykonání poslední instrukce uživatelského programu je předáno řízení systémovému programu, který provede tzv. otočku cyklu. V ní nejprve aktualizuje hodnoty výstupů a vstupů: hodnoty uložené dosud v paměti jako obrazy výstupů (registry Y) přepíše do registrů výstupních periferních modulů a hodnoty ze vstupních modulů okopíruje do paměťových obrazů vstupů (registry X). Dále aktualizuje časové údaje pro časovače a systémové registry, ošetří komunikaci a provede ještě řadu režijních úkonů. Po otočce cyklu je opět předáno řízení prvé instrukci uživatelského programu. Po spuštění programu jsou nejprve provedeny režijní operace systému, aktualizace systémových a časových proměnných, naplánována aktivace procesů pro další cyklus, ... apod. Poté jsou sejmuty aktuální hodnoty fyzických vstupů, které jsou pro celý následující cyklus konzervovány, jako obrazy vstupů X. Následně se vykoná požadovaný program a na závěr jsou na výstupy vyslány aktuálně vyčíslené hodnoty obrazů výstupů Y. Celý cyklus se stále opakuje.

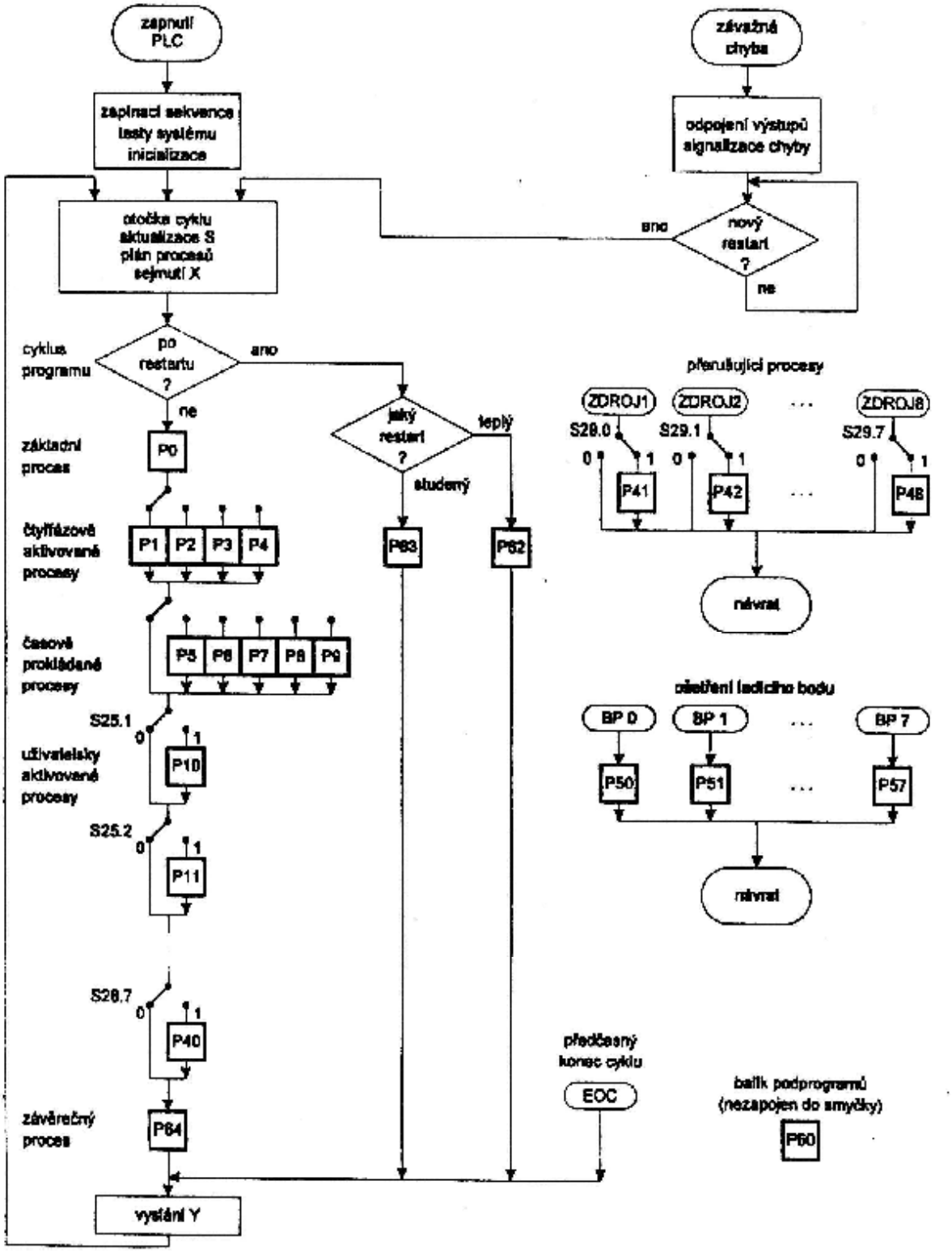
Obr. 1 Cyklické vykonávání programu

2.4.2 Obrazy vstupů a výstupů

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

2.4.3 Multiprogramování, vícesmyčkový režim

Některé systémy dovolují práci v určitém režimu multiprogramování nebo vícesmyčkové aktivace, případně práci v přerušovacím režimu. Na obr. 2 je příklad multiprogramové struktury PLC programu pro TECOMATy. Systém nabízí uživateli určitou "kostru", soubor procesů P0 až P64, pro které jsou dána pevná pravidla aktivace. Proces P0 se aktivuje vždy po otočce cyklu (úvodní proces), P64 naopak vždy před otočkou (závěrečný proces), procesy P1, P2, P3, P4 se v aktivaci cyklicky střídají (čtyřfázové procesy), P5 až P9 se aktivují v časových prioritách (každých 0,4 s; 3,2 s; 25,6 s; 3,4 min a 27,2 min). O aktivaci procesů P10 až P40 (uživatelské procesy) rozhoduje programátor tím, jak nastaví hodnoty aktivačních bitových proměnných v systémových registrech S25 až S29. Procesy P62 a P63 se aktivují jednorázově po zapnutí nebo po restartu systému (inicializace při teplém a studeném restartu). Procesy P41 až P44 jsou aktivovány jako odezva na přerušující událost (interval 10 ms, změna hodnoty na přerušujícím vstupu, při zjištění méně závažné chyby v programu, podle stavu čítače vnějších událostí). Je ponecháno na vůli programátora, jak dalece tuto strukturu aktivace procesů využije. Procesy, které nejsou vytvořeny, nebudou ani aktivovány. Pokud tedy všechny instrukce svého uživatelského programu zapíšeme do procesu P0 mezi instrukce P0 a E0, bude celý program aktivován v jediné nestrukturované programové smyčce. Nejsou-li vážné důvody pro složitější struktury, doporučujeme zůstat u tohoto nejjednoduššího způsobu aktivace.



Obr. 2 Schéma multiprogramové aktivace

2.5 Programovací jazyky PLC

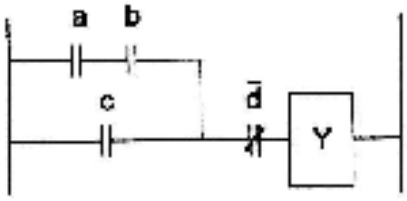
K programování nabízejí PLC systémy specializované jazyky. Jazyky systémů různých výrobců jsou podobné, nikoliv však stejné. Mezinárodní norma IEC 1131-3 však sjednocuje programovací jazyky pro PLC. Kodifikuje čtyři typy jazyků.

2.5.1 Jazyk mnemokódů

("Instructions List", "IL", v německé terminologii "Anweisungslist", "AWL") je obdobou assembleru u počítačů, a je také strojově orientován. To znamená, že každé instrukci PLC systému odpovídá stejně pojmenovaný příkaz jazyka. Jazyky mnemokódů poskytují i obvyklý "assemblerský komfort", tj. aparát symbolického označení návěští pro cíle skoků a volání, symbolická jména pro číselné hodnoty, pro pojmenování vstupních, výstupních a vnitřních proměnných a jiných objektů programu (datových bloků a tabulek, struktur a jejich prvků), pro automatické přidělování paměti pro uživatelské registry, pro jejich inicializaci (zadání počátečního obsahu), pro zadávání číselných hodnot v různých číselných soustavách. V současnosti se však tento jazyk používá jen zřídka.

2.5.2 Jazyk kontaktních (reléových) schémat

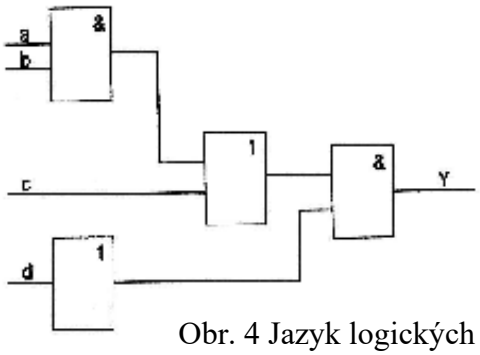
("Ladder Diagram", "LD", německy "Kontaktplan", "KOP" viz obr. 3.) je grafický. Program se základními logickými operacemi zobrazuje schéma ve formě obvyklé pro kreslení schémat při práci s reléovými a kontaktními prvky. Pouze symboly pro kontakty a cívky jsou zjednodušeny, aby mohly být vytvářeny semigraficky: spínací kontakty jako dvojice svislých čárek, rozpínací kontakt je navíc "přetržen" lomítkem, cívky jsou označovány dvojicí závorek. Funkční bloky (např. čítače, časovače) jsou kresleny jako obdélníkové značky. Instrukce, které nemají svou analogii v kontaktní symbolice (a těch bývá většina) se obvykle zobrazují jako dvojice závorek nebo obdélníková značka s vepsaným mnemokódem instrukce. Jazyk kontaktních schémat je výhodný při programování nejjednodušších logických úloh a v případech, kdy s ním pracuje personál, který nezná (a nechce znát) tradiční počítačové programování. Pokud ale v programu převažují složitější instrukce (třeba aritmetické instrukce, skoky a volání), pak je kontaktní schéma již násilné a postrádá svou názornost.



Obr. 3 Jazyk kontaktních (reléových) schémat

2.5.3 Jazyk logických schémat

(jazyk funkčních bloků, "Function Block Diagram" německy "FUP") - obr. 4 je opět grafický. Základní logické operace popisuje obdélníkovými značkami. Své značky mají i ucelené funkční bloky, např. čítače, časovače, posuvné registry, paměťové členy, ale i aritmetické a paralelní logické instrukce. Vychází vstříc uživatelům zvyklým na kreslení logických schémat pro zařízení s integrovanými obvody. Obdobný, ale obecnější, jazyk se využívá při popisu a programování systémů, zpracovávajících analogové proměnné, při programování regulačních a měřicích úloh. V moderních softwarech se tento jazyk může kombinovat s jazykem reléových schémat.



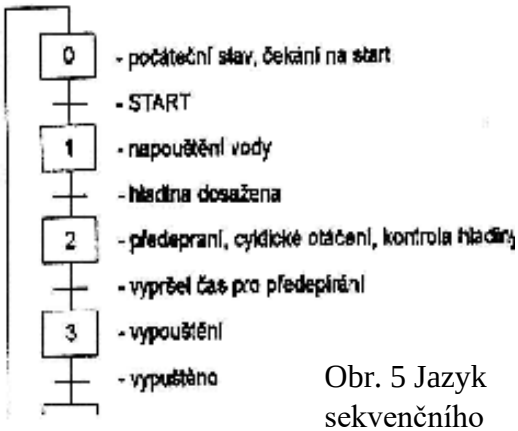
Obr. 4 Jazyk logických schémat

2.5.4 Jazyk strukturovaného textu

Je obdobou vyšších programovacích jazyků pro PC (např. Pascal nebo C). Umožňuje úsporný a názorný zápis algoritmů.

2.5.5 Grafické prostředí pro sekvenční programování

Nadstavbu nad popsanými jazyky tvoří grafické prostředí pro sekvenční programování – obr.5 (SFC, GRAFCET). Dovoluje stavový popis sekvenčních úloh v symbolice přechodového grafu konečných automatů a určité třídy Petriho sítí. K popisu struktury používá značky stavů, přechodů a větvení. Jazyk sekvenčního programování je velmi názorný a podporuje systémový přístup k programování. Programátor má malý prostor k vytváření chaoticky neuspořádaných programů, je nucen zamyslet se nad podstatou problému, má možnost systematicky ji popsat a realizovat. Většina řízených technologií je svou podstatou sekvenční. Poměrně náročným sekvenčním problémem bývá i vyhodnocení posloupnosti tlačítek a zásahů obsluhy.



Obr. 5 Jazyk sekvenčního programování

3. Řídící systémy firmy AMiT

3.1 O firmě AMiT, spol. s.r.o

Společnost AMiT je předním českým výrobcem řídicích systémů a elektroniky pro průmyslovou automatizaci a automatizaci budov**.** Významnou pozici na trhu dosáhli především díky úzkému propojení vývoje s moderní výrobou a díky bezkonkurenční technické podpoře. Vysoké nároky na kvalitu výroby a výsledné produkty, podpořené více než dvacetiletou praktickou zkušeností, jsou zárukou úspěšných realizací i u těch nejnáročnějších zákazníků a investorů. Kromě vlastních řídicích systémů a jejich elektronických prvků nabízí svým partnerům a zákazníkům vlastní know-how i v segmentu návrhových a ladicích programových prostředků, které podporují rychlé a spolehlivé uvedení automatizační techniky do provozu a její následné provozování a údržbu.

3.2 Stručná historie

1992 – Založení společnosti, orientace na zakázkový vývoj
1993 – První standartní řídící systémy pro průmyslovou automatizaci
1994 – Založení pobočky v Brně
1997 – Certifikace systému řízení jakosti dle normy ISO 9001:1994
2000 – Vlastní linka pro automatizovanou povrchovou montáž
2001 – Vlastní testovací pracoviště EMC
2005 – Změna sídla společnosti, nové výrobní prostory a modernizace výroby
2013 – Počet vlastních zaměstnanců překročil hranici 100, rozšiřování výroby
2015 – Zásadní investice do modernizace výroby
2016 – Rozšíření výrobních prostor a výrobního centra

3.3 Produkty

3.3.1 Volně programovatelné řídící systémy

Volně programovatelné řídicí systémy firmy AMiT jsou vhodné pro všechny aplikace tzv. malé a střední automatizace zejména v oblasti automatizace budov, řízení technologických celků, energetice nebo vytápění měst a obcí. Díky svým mnohdy unikátním vlastnostem představují tyto řídicí systémy ideální volbu z hlediska poměru cena/výkon. Vysokou spolehlivost našich řídicích systémů v praxi potvrzují tisíce realizovaných aplikací po celém světě.

Všechny řídicí systémy mohou být velmi snadno připojeny do informačního systému DB-Net/IP vyvinutého firmou AMiT. Pro integraci se systémy jiných výrobců lze s výhodou použít komunikaci MODBUS RTU a MODBUS TCP. Pro připojení dalších zařízení, regulátorů či snímačů k systému lze použít řadu podporovaných komunikačních protokolů případně využít možnost parametrizace uživatelského protokolu. Programování a parametrizace řídicích systémů je jednotné ve vlastně vyvinutém vývojovém prostředí DetStudio.

3.3.2 Operátorské panely a terminály (HMI)

Operátorské panely (terminály, HMI) jsou nedílnou součástí řídicích systémů. Díky těmto zařízením získává obsluha řízené technologie okamžité informace a stavu technologie a má možnost přímých zásahů nebo nastavení parametrů regulačních a řídicích algoritmů. V koncepci firmy AMiT však mnohé operátorské panely mohou plnit i funkci řídicích systémů. Lze je tedy volně programovat ve vývojovém prostředí DetStudio a začlenit je tak přímo do řídicí struktury s ostatními systémy s podporou komunikací DB-Net/IP nebo MODBUS.

3.3.3 Moduly vzdálených vstupů/výstupů

Moduly vzdálených vstupů a výstupů se používají pro rozšíření počtu vstupů a výstupů řídicího systému a pro připojení vzdálených signálů, čímž se významně šetří náklady na kabeláž. Připojením signálů k rozšiřujícím modulům v místě, kde signály vznikají, se navíc zvyšuje odolnost proti rušení především u analogových signálů – hodnoty se k řídicímu systému přenášejí zabezpečeným komunikačním protokolem a nemůže tedy dojít ke zkreslení. CPU jednotky řídicích systémů jsou zpravidla natolik výkonné, že mohou bez problémů zpracovávat mnohem více vstupů a výstupů, než je fyzicky možné k řídicímu systému připojit. Použitím rozšiřujících modulů, které jsou levnější než řídicí systémy, lze získat cenově optimálnější řešení. Firma AMiT nabízí řadu rozšiřujících modulů pod označením DM-xx (komunikační protokol ARION) a DMM-xx (protokol Modbus RTU). Moduly mohou být čistě jednodruhové (jeden typ V/V signálu) anebo kombinované.

3.3.4 Programovatelné regulátory AMREG

Nová generace volně programovatelných regulátorů pro automatizaci budov. Všechny regulátory řady AMREG umožňují naprogramovat vlastní ovládací algoritmus včetně návrhu obrazovek u jednotek s displejem ve vývojovém prostředí DetStudio. Každý regulátor je navržen pro konkrétní problematiku regulace (ovládání Fan Coil jednotek, ovládání tepelných zdrojů, ovládání topných větví apod.) avšak konkrétní použití a začlenění do komplexního řešení je jen na tvůrci uživatelského algoritmu bez jakýchkoliv omezení. K regulátorům AMREG je výhodné používat nástěnné programovatelné ovladače, které jsou založeny na stejné koncepci tvorby programů a komunikaci. Pro standardní řešení a běžně používané algoritmy jsou k dispozici volně ke stažení typová řešení aplikačních programů, která lze použít, jak jsou anebo se nechat inspirovat pro tvorbu vlastního originálního řešení.

3.3.5 Volně programovatelné nástěnné ovladače

Nová generace volně programovatelných nástěnných ovladačů pro automatizaci budov**.** U ovladačů řady AMREG lze ve vývojovém prostředí DetStudio naprogramovat jakoukoliv funkčnost a u ovladačů s grafickým displejem navíc vytvořit i vlastní design jednotlivých obrazovek ovládání. Všechny nástěnné ovladače řady AMREG podporují komunikační protokoly MODBUS RTU nebo ARION a umožňují snadnou integraci jak do sítě regulátorů řady AMREG, tak i do sítě systémů dalších výrobců.

3.3.6 Průmyslové počítače

Panelové a vestavné počítače firmy AMiT, postavené na platformě PC, podporují otevřené softwarové standardy a umožňují tak rychlou reakci na aktuální trendy v IT oblasti. Na těchto počítačích lze provozovat většinu dostupných operačních systémů. Standardně se počítače dodávají s předinstalovaným operačním systémem GNU/Linux nebo Windows Embedded případně s firemním monitorovacím a vizualizačním prostředím TouchDet.

3.3.7 Průmyslové komunikační převodníky

Komunikační převodníky rozšiřují funkční možnosti řídicích systémů firmy AMiT. Některé převodníky však vyžadují pro svou činnost systémovou podporu ze strany řídicích systémů nebo programovatelných regulátorů.

4 Programovací softwary firmy AMiT

4.1 DetStudio

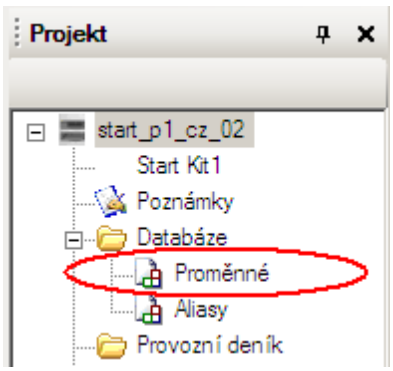
Vývojové prostředí DetStudio je určeno pro tvorbu uživatelských aplikací pro všechny standardní řídicí systémy a programovatelné regulátory firmy AMiT. V jediném prostředí lze vytvořit vlastní aplikaci, navrhnout a odsimulovat vzhled obrazovek zobrazovačů řídicích systémů, definovat chybová hlášení, on-line ladit běžící aplikaci, vytvořit dokumentaci vytvořeného programu.

V této kapitole jsou popsány základy zacházení a vlastnosti tohoto prostředí.

4.1.1 Funkce softwaru a Tvorba Aplikace

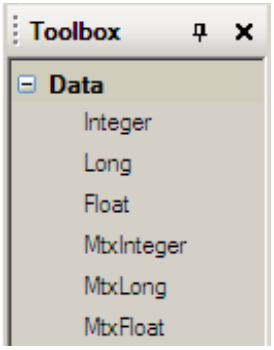
Při návrhu aplikace budeme vycházet z okna projektu, které má stromovou strukturu a je rozděleno do několika sekcí.

Tabulku nadefinovaných proměnných lze zobrazit kliknutím na položku Databáze/Proměnné v okně projektu.



Obr. 6 Okno Projekt položka Proměnné

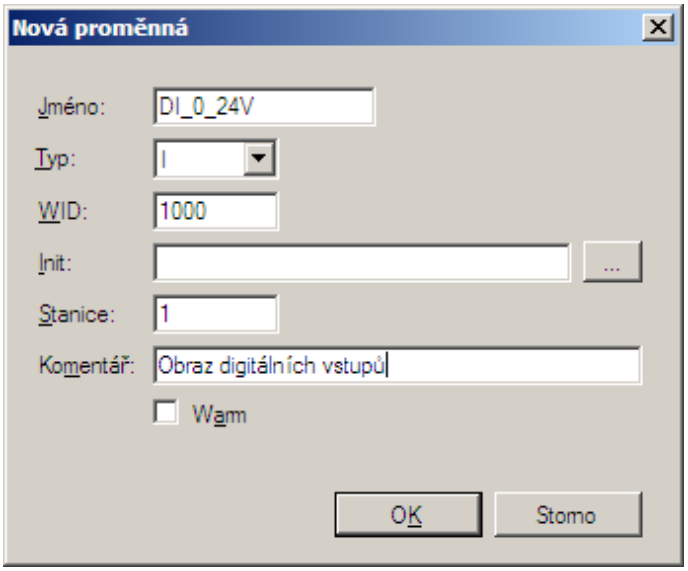
V pracovním okně projektu se otevře záložka „Proměnné“ a v Toolboxu se objeví rozvinovací položka s názvem „Data“. Kliknutím na položku „Data“ rozvineme nabídku proměnných, které lze definovat.



Obr. 7 Okno Toolbox – seznam proměnných

V DetStudiu je možné vytvořit mimo běžné proměnné také matice.

Založení proměnné



Obr. 8 Okno s definicí nové proměnné

Jméno – Jedinečné jméno v rámci řídicího systému.

Typ – Určuje datový typ proměnné. Může nabývat hodnot I, L, F, MI [x, y], ML [x, y], MF [x, y]. Písmena x, y určují rozměry matic.

WID – Číselný identifikátor. Toto číslo je používáno při přístupu k proměnné a musí být jedinečné v celé aplikaci (tedy i v síti řídicích systémů). Přidělování WID řeší DetStudio automaticky a nedoporučuje se jej editovat (až na výjimečné případy).

INIT – Pole, do kterého lze zadat inicializovanou (prvotní) hodnotu proměnné. Pokud bude vybrán maticový typ proměnné, lze pomocí tlačítka vyvolat okno „Inicializační hodnota proměnné“, ve kterém lze editovat jednotlivé buňky matice.

Stanice – Číslo řídicího systému, ve kterém je databázová proměnná umístěna.

Komentář – Uživatelský popisek funkce proměnné.

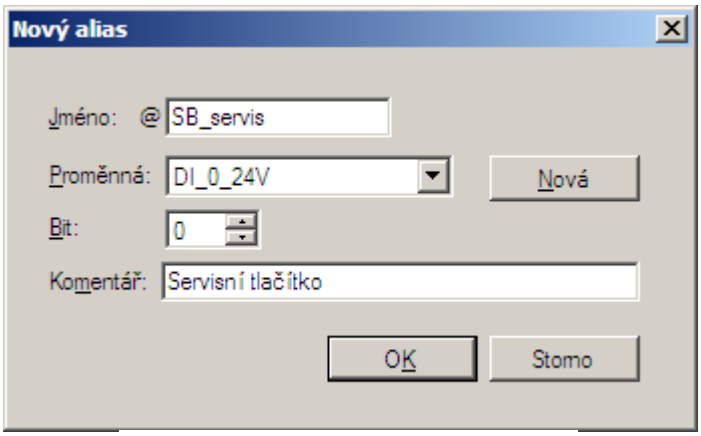
Warm – Příznak inicializace při teplém startu. Pokud není toto okénko zaškrtnuto, řídící terminál uloží do proměnné její poslední známou hodnotu namísto inicializované.

Proměnnou lze dodatečně editovat v tabulce proměnných.

Práce s Aliasy

K celočíselným proměnným typu I(MI)/L(ML) lze přistupovat nejen jako k číslům, ale také jako k šestnácti/třicetidvěma hodnotám True/False. Takovéto hodnoty se nazývají bity proměnné a jsou určeny číslem 0 až 15 respektive 0 až 31. Bity jsou pak identifikovány jménem proměnné a číslem bitu (jméno.číslo). Jednotlivé bity proměnných typu I a L lze také pojmenovat a odkazovat se na ně přímo pomocí těchto jmen. K tomu slouží alias jména. Alias jméno pak nahrazuje odkaz pomocí jména proměnné a čísla bitu: alias = jméno.číslo V zápisu je alias označován svým jménem, kterému předchází znak @.

Kliknutím na položku Databáze/Aliasy v okně projektu se zobrazí tabulka nadefinovaných aliasů.



Obr. 9 Okno vytvoření aliasu

IO konfigurace

Kliknutím na položku IO Konfigurace v okně projektu se zobrazí záložka „IO Konfigurace“ se seznamem dostupných kanálů

Tento seznam se liší v závislosti na typu použitého řídicího systému. Pokud je nastaven přepínač „Při založení nového projektu automaticky pojmenovat signály“ v dialogu Nástroje/Možnosti/Procesy/Obecné, pak jsou signály pojmenovány dle konvence.

DetStudio používá dva základní pojmy:

Fyzický kanál – Jedná se o skupinu signálů, která je fyzicky dostupná na řídicím systému (místo kam se fyzicky připojují vstupní či výstupní periferie).

Logický kanál – Tímto kanálem definujeme, jakým způsobem se přistupuje ke kanálům fyzickým. K jednomu fyzickému kanálu tak lze přistupovat z více kanálů logických. Jako příklad lze uvést analogové vstupy, ke kterým lze přistupovat jako k napěťovým a proudovým nebo k nim lze přistupovat jako ke vstupům pro čidla Ni1000.

Tato filozofie umožňuje zcela efektně přecházet na různé typy řídicích systémů se stejným aplikačním programem, neboť fyzické rozdíly jsou potlačeny jednotným systémem logických kanálů. Při přechodu na jiný typ řídicího systému je pak nutné pouze překontrolovat přiřazení logických kanálů k fyzickým.

V editoru IO konfigurace lze předdefinovaná jména signálů změnit a přiřadit jim symbolické jméno, které je dále používáno v programu.

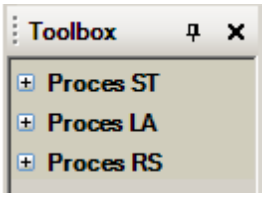
IO Arion

Každý řídicí systém firmy AMiT lze rozšířit o další vstupy/výstupy pomocí tzv. vzdálených vstupů/výstupů. Jedná se o moduly s určitým počtem vstupů nebo výstupů bez vlastní inteligence, které komunikují s řídicím systémem prostřednictvím linky RS485 pomocí protokolu ARION. Pro parametrizaci sítě ARION slouží v okně „Projekt“ sekce „IO Arion“.

Procesy

Činnost řídicího systému probíhá sekvenčně – je rozdělena do tzv. procesů. Každý proces je část programu, která pracuje relativně samostatně a nezávisle na ostatních procesech. Zejména u jednodušších řídicích systémů je výhodné popsat jedním procesem jeden regulační nebo měřicí okruh. Tím je zajištěna správná časová součinnost a vazba všech prvků okruhu a nezávislost na dalších okruzích.

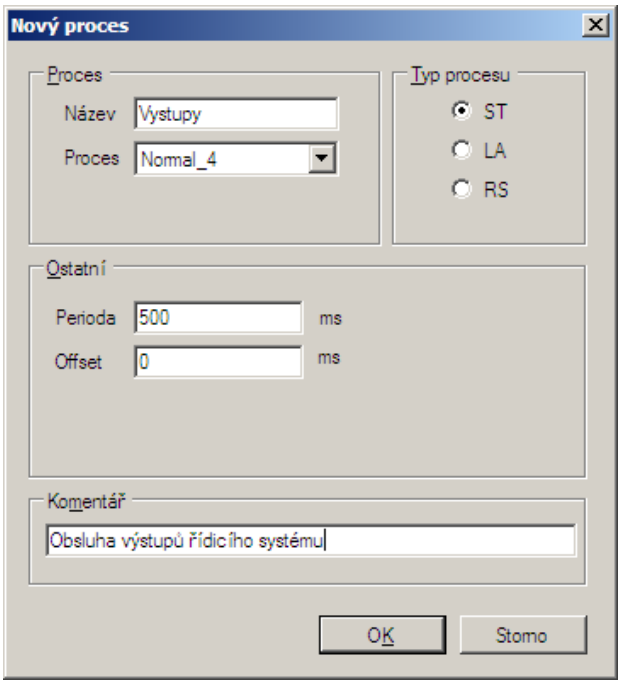
V okně „Toolbox“ se objeví seznam procesů, které lze v řídicích systémech definovat dle programovacího jazyka.



Obr. 10 Okno Toolbox – seznam procesů dle programovacího jazyka

- Proces ST (jedná se o klasický strukturovaný text).
- Proces LA (práce s vrcholem zásobníku, jazyk podobný assembleru).
- Proces RS (programování pomocí reléových schémat).

Založení nového procesu



Obr. 11 Okno Nový proces

Proces – Nastavení priority procesu. Pokud je třeba vykonat proces s vyšší prioritou než aktuálně probíhající proces, je probíhající proces pozastaven a bude pokračovat až po vykonání procesu s vyšší prioritou.

Perioda – Interval, po které se proces pravidelně vykonává.

Offset – Zpoždění procesu po uplynutí periody.

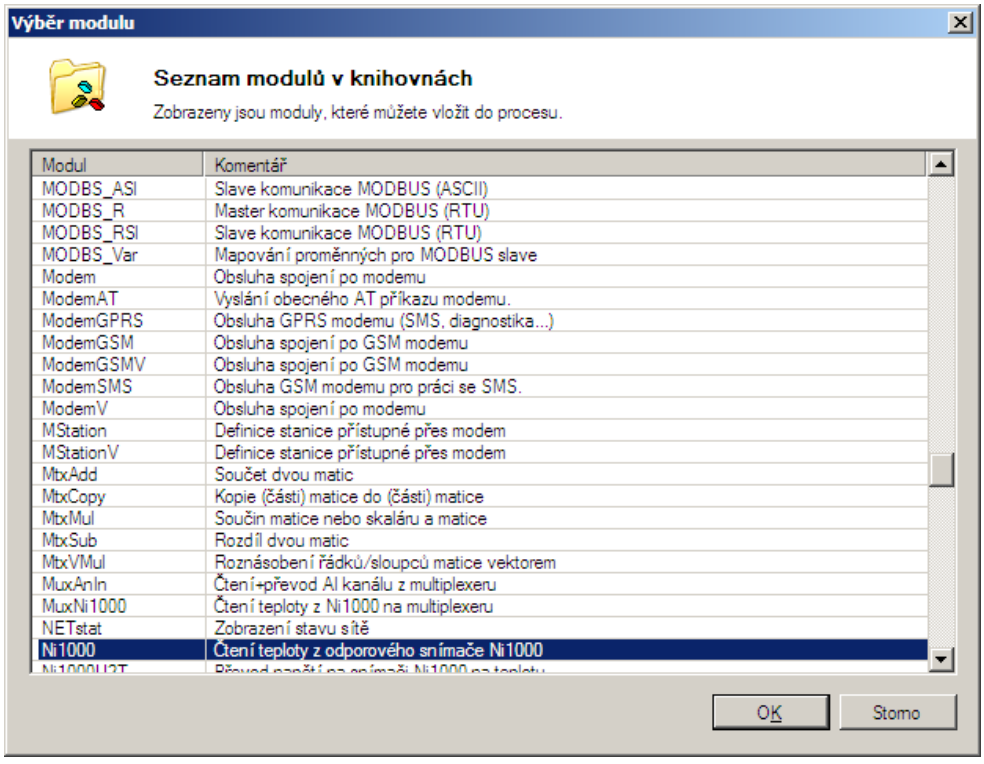
Proces ST a RS – Programování

Programování ST spočívá především ve vkládání funkčních modulů a jejich parametrizaci. Na každý řádek lze vložit pouze jeden modul.

Příklad modulu:

Ni1000 #Teplota, Tepl_Aktual, 6180

Modul se vkládá ze seznamu (který je možno zobrazit pomocí kláves Ctrl+l), který současně obsahuje jeho popis. Po rozkliknutí se zobrazí podrobná nápověda, jak modul nastavit.



Obr. 12 Okno Výběr modulu

Počet reléových modulů, které lze v jazyce RS využít, je omezen a nedosahuje tolik možností jako klasický strukturovaný text. Z toho důvodu lze jazyk RS a strukturovaný text kombinovat (stisknutím ikony ST v nástrojové liště RS procesu).

Obr. 13 Nástrojová lišta RS procesu

Podprogramy

Podprogram je speciální typ procesu, který se vykoná pouze při jeho zavolání (např. modulem Call). Lze definovat až 900 podprogramů. Seznam podprogramů lze zobrazit dvojklikem na položku Podprogramy v okně projektu.

Funkční bloky

V DetStudiu lze také definovat funkční bloky. Funkční blok je prvek, jehož vnitřní struktura je definovaná uživatelem, ale navenek se chová jako jakýkoli jiný modul DetStudia. Pro práci s funkčními bloky slouží položka Funkční Bloky.

Obrazovky

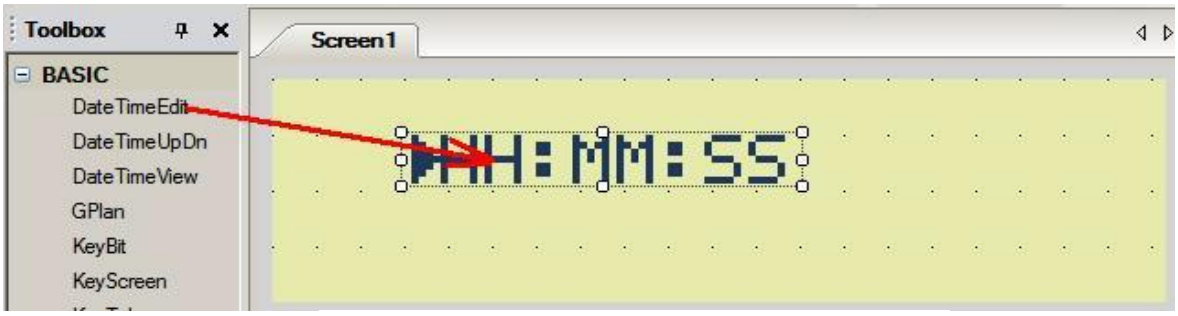
Princip tvorby obrazovek spočívá v umísťování ovládacích prvků na vytvářené obrazovky a v jejich parametrizaci. Zobrazení v editoru obrazovky odpovídá skutečnému zobrazení na displeji. Seznam nadefinovaných obrazovek lze zobrazit kliknutím na položku Obrazovky v okně projektu.

Obrazovka Global – Je určena pouze pro umístění neviditelných prvků. Vždy se vykonají nejdříve prvky v globální obrazovce a teprve potom v právě zobrazené obrazovce. Do globální obrazovky se typicky umísťuje taková funkčnost aplikace, která musí být prováděna vždy, nezávisle na právě zobrazované obrazovce. Příkladem může být signalizace alarmu blikající diodou, reakce na konkrétní klávesy v celé aplikaci apod.

Standartní obrazovky – Jsou to všechny obrazovky, mimo obrazovku „Global“. Na takovéto obrazovky lze umístit jak viditelné, tak neviditelné prvky dostupné v okně „Toolbox“. Jejich velikost je závislá na typu zvoleného terminálu.

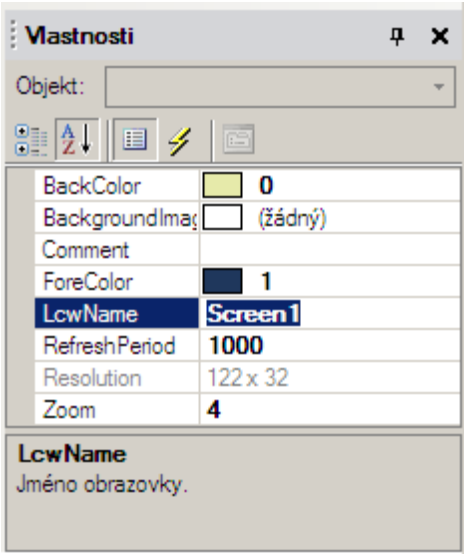
Návrh obrazovky

Vložení prvku – Po otevření obrazovky se v pracovním okně objeví displej (rozměrově dle zvoleného typu řídicího systému/terminálu), do kterého lze umisťovat požadované prvky.



Obr. 14 Přetažení prvku z okna Toolbox

Vlastnosti obrazovky a prvků se pak editují v okně vlastnosti.



Obr. 15 Okno Vlastnosti

Některé ovládací prvky nemají grafické rozhraní (například prvek KeyScreen). Takové prvky jsou pak umísťovány automaticky do spodní části pracovní plochy obrazovky oddělené šedou čarou.

Inspektor

K ladění je možné využít sledování obsahu vybraných procesních proměnných. K tomu slouží v DetStudiu Inspektor a v případě schémat RS tzv. Watch režim.

Pomocí inspektoru lze:

- Zobrazovat proměnné v různých formátech.
- Editovat hodnoty proměnných.
- Vytvářet vlastní funkce pro přepočet zobrazované proměnné.
- Zobrazovat grafický průběh proměnné.
- Ukládat hodnoty proměnné do souboru.
- Načíst proměnné z připojeného řídicího systému.

Inspektor lze otevřít pomocí hlavního menu Ladění/Inspektor X, kde X je z rozsahu 1 až 4 a určuje konkrétní okno inspektora.

Archivace

V řídicích systémech firmy AMiT lze přímo definovat archívy, které lze vyčítat, ukládat a následně zpracovávat v PC. Archívy jsou uloženy v zálohované paměti RAM. Pro práci s archívy slouží prvek SyncArch, případně prvek SyncMark.

Provozní deník

Řídicí systém vždy automaticky obsahuje provozní deník s hloubkou 50 hlášení. Slouží k evidenci chyb a jiných hlášení, které se vyskytnou při vlastním provozu řídicího systému. Je to kruhový buffer. Tento provozní deník se nazývá systémový a je umístěn v operačním systému NOS. Systémový provozní deník lze prohlížet pouze na terminálu připojeném k řídicímu systému. Aby mohl být provozní deník přístupný i po komunikační síti a mohl se tak zpracovávat speciálními programy na stanicích PC, musí se v aplikaci definovat tzv. aplikační provozní deník.

4.2 ViewDet

ViewDet je samostatný servisní nástroj. Doplňuje a rozšiřuje možnosti vývojového prostředí DetStudio v oblasti sledování, ladění a nastavení aplikace v řídicím systému. V omezené míře lze program použít jako jednoduchou vizualizaci.

Základní vlastností je možnost čtení a zápisu jednotlivých hodnot databázových proměnných a aliasů. Navíc ViewDet umí číst, zobrazovat, tisknout a exportovat archivy a provozní deník z řídicího systému včetně pamatování si jejich historie.

Archivy lze zobrazit buď v tabulce nebo formou grafu. Další možností je tvorba archivů a zobrazování archivů vznikajících přímo na PC periodickým čtením určitých hodnot z procesní stanice. Zobrazované proměnné lze umístit libovolně v tzv. „scéně“ s nadefinovaným obrázkem na pozadí – takto lze vytvořit velmi jednoduchou vizualizaci procesů. Pomocí zámku se zajistí, aby uživatel nechtěně nezměnil proměnné a parametry zobrazování ve ViewDet. ViewDet umožňuje kompletní editaci IP konfigurace stanic připojených k průmyslovému Ethernetu.

Mezi hlavní benefity programu ViewDet patří:

- možnost uskupení zobrazovacích objektů ve scénách, rychlé přepínání mezi scénami
- pozadí ve scénách
- statický text a proměnná
- snadná editaci časových plánů grafickou metodou
- nastavitelná perioda komunikace jež lze nastavit individuálně ke každému prvku
- zavedení režimu user a admin – ochrana proti nechtěným změnám
- podpora aliasů
- tisky scén a prvků
- exporty zaarchivovaných dat
- download uživatelské aplikace do řídicího systému včetně možné zálohy a obnovení obsahu proměnných
- přehledné okno návrhu projektu
- vícenásobné komunikační profily se snadným přepínáním
- hromadná synchronizace času řídicích systémů

4.2.1 Scény

Tvoří hlavní funkci programu ViewDet. Provádějí se zde prakticky veškeré operace a funkce, které tento program nabízí.

Každou ze scén si lze představit jako volnou pracovní plochu, na kterou lze vkládat různé, předdefinované, zobrazovací/editační prvky. Při práci s programem je možné mít otevřeno více scén v jednom okamžiku a libovolně se mezi nimi přepínat.

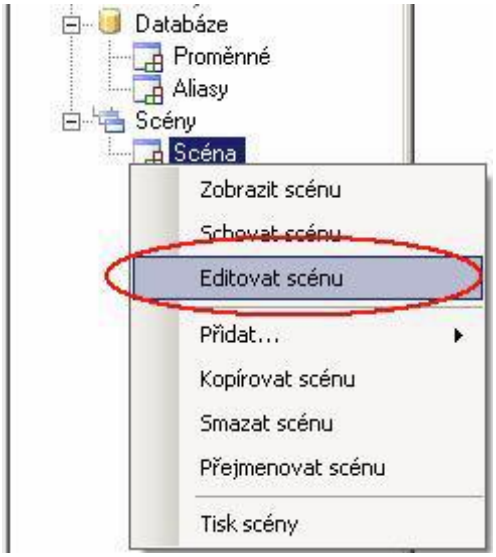
Vytvoření scény

Scény lze v projektu ViewDet vytvořit v kontextovém menu vyvolaným na položce Scény v okně Projekt

Obr. 16 Vytvoření scény z kontextového menu

Parametry scény

Editace parametrů scény probíhá prostřednictvím okna Oprava parametrů scény, které lze vyvolat z kontextového menu vyvolaného v okně Projekt na požadované scéně.



Obr. 17 Editace scény z kontextového menu

Okno Oprava parametrů scény má tři záložky.

Jméno – Aby byla možná lepší orientace v jednotlivých scénách, je možné každou scénu pojmenovat.

Parametry – Pomocí volby Použij periodu z nastavení projektu zvolíme, zda se mají vyčítat data pro prvky umístěné na scéně s tzv. globální periodou nebo zda mají být data vyčítána s jinou periodou, zadanou do položky Perioda [ms].

Pozadí – V záložce Pozadí se dané scéně definuje pozadí (například schéma řízeného systému). Jako pozadí lze vkládat obrázky typu JPG, BMP, GIF, PNG, TIF apod. Následně se zobrazí scéna s nadefinovaným obrázkem na pozadí, do které lze vkládat jednotlivé scénické prvky.

Scénické prvky

Na každou scénu je možné vložit libovolné množství scénických prvků. K dispozici jsou tyto scénické prvky:

Inspektor – prvek, ve kterém je možné zobrazit větší počet proměnných s jejich hodnotami.

Matice – prvek zobrazující hodnoty matice definované v řídicím systému.

Archiv – prvek zobrazující archiv vzorků uložený v řídicím systému.

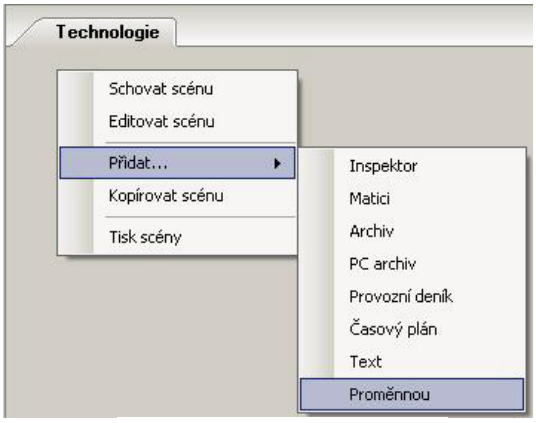
PC archiv – prvek umožňující archivaci načtených vzorků v PC.

Provozní deník – prvek spravující provozní deníky.

Časový plán – prvek umožňující komfortní správu časových plánů.

Text – textový popisek definované barvy popředí a pozadí.

Proměnnou – prvek, který zobrazuje jednoduché okno obsahující hodnotu definované proměnné.



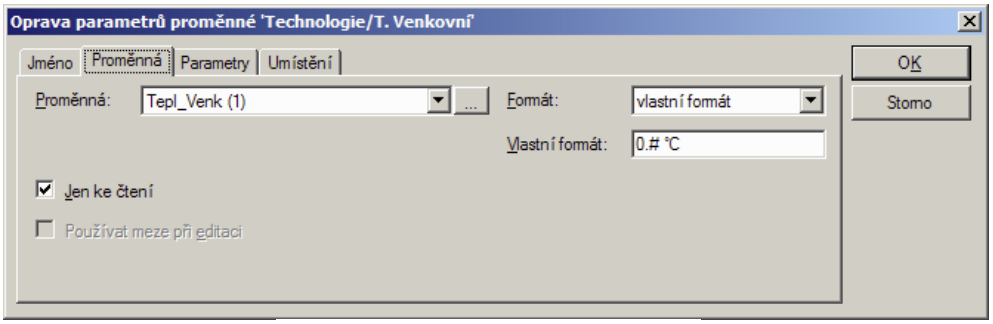
Obr. 18 Vložení prvku

Návrh scény

Prvek proměnná

Prvek Proměnná slouží pro zobrazení/editaci hodnot proměnných a aliasů.

Záložka proměnná – Kliknutím na ikonu dojde k rozvinutí nabídky proměnných, které lze do prvku Proměnná vložit. Budeme chtít zobrazit hodnotu venkovní teploty. V seznamu vyberte proměnnou Tepl_Venk. V případě, že vyžadujeme, aby prvek Proměnná zobrazoval hodnotu včetně textového řetězce (např. včetně znaku °C), zvolí se v při definici prvku Proměnná formát zobrazení jako „Vlastní formát“ a položka Vlastní formát se vyplní dle následujícího obrázku.



Obr. 19 Nastavení proměnné

Záložka umístění – V záložce Umístění lze zadat přesné souřadnice, kde má být prvek na scéně umístěn. V případě, že se zadá hodnota 0, bude se šířka prvku automaticky přizpůsobovat počtu míst zobrazovaného čísla.

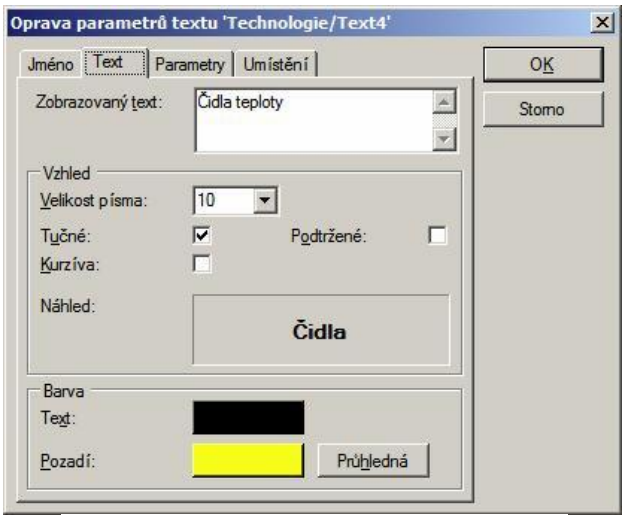
Tím dojde ke vložení prvku Proměnná do scény.

Obr. 20 Prvek proměnná

Prvek text

Slouží k zobrazování krátkých textů na scéně. Je možné u něj definovat vzhled písma, barvu popředí (inkoustu) a pozadí (papíru).

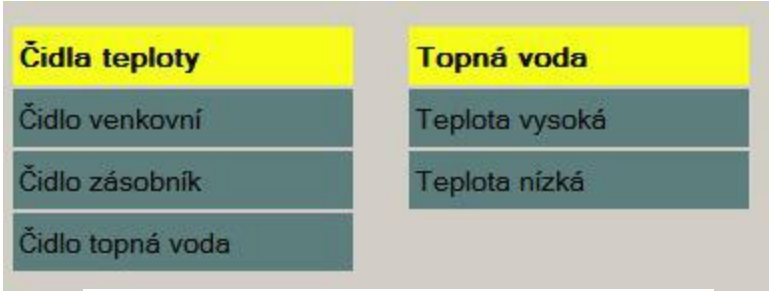
Záložka text – Zde se nastavují vlastnosti a obsah textu, a to včetně barvy textu a pozadí.



Obr. 21 Nastavení vkládaného textu

Záložka parametry – Obsahuje volbu zámek, která zabrání běžnému uživateli měnit parametry tomuto prvku.

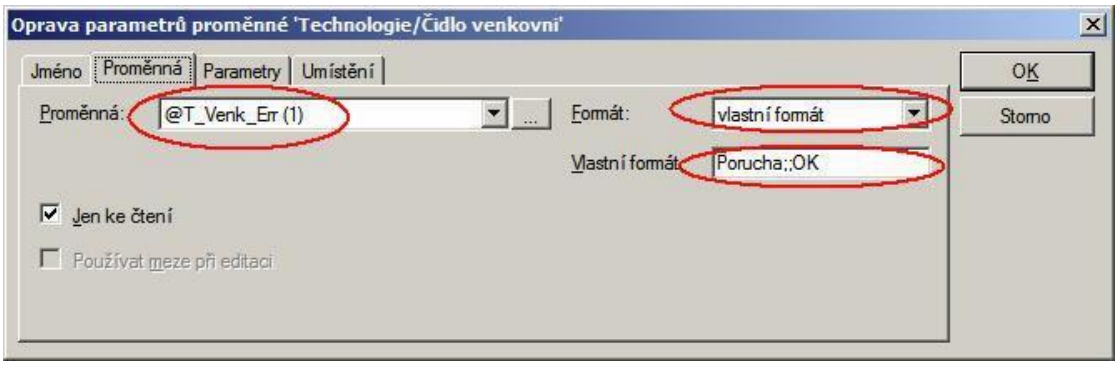
Výsledné prvky mohou vypadat následovně.



Obr. 22 Příklad prvků Text na obrazovce

Prvek proměnná (zobrazení aliasů)

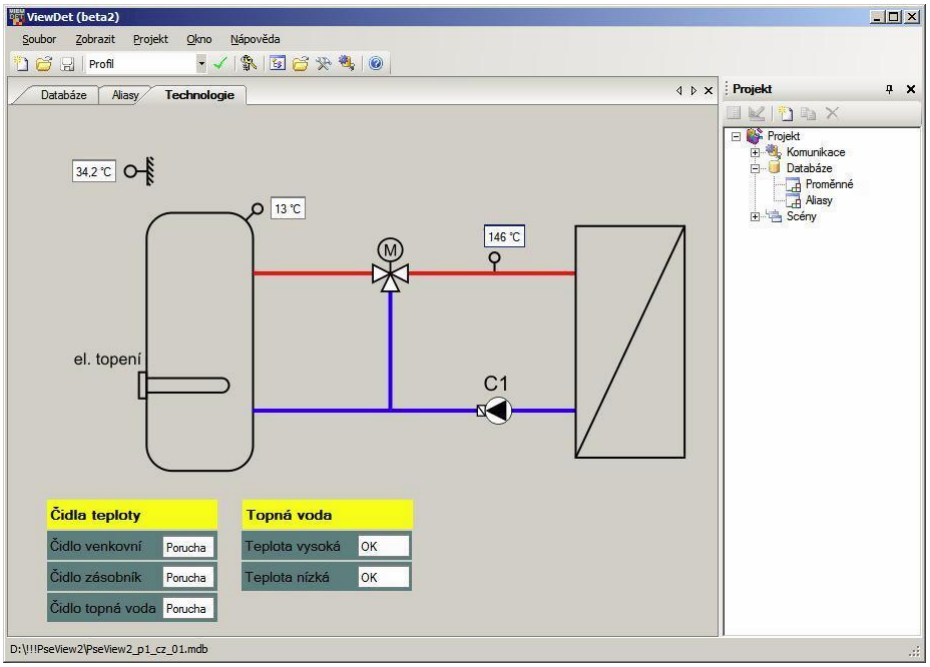
Pomocí prvku Proměnná lze zobrazovat i hodnoty aliasů. V tomto případě lze i nastavit textové řetězce, které se v prvku můžou měnit podle hodnoty aliasu (vhodné pro zobrazení upozornění). Toho docílíme nastavením položky Formát na hodnotu „Vlastní formát“. Do pole Vlastní formát pak zadejte řetězec.



Obr. 23 Nastavení upozornění

Aby prvek neměnil rozměr podle délky textu, je vhodné nastavit pevnou hodnotu (různou od 0) Šířka v záložce umístění.

Výsledná scéna pak může vypadat například takto.



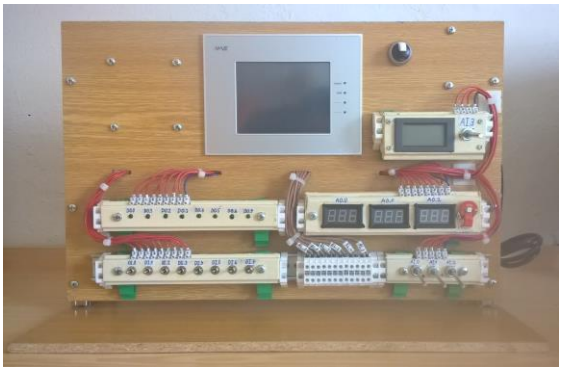
Obr. 24 Příklad výsledné scény

Prvek Archiv a PC archiv

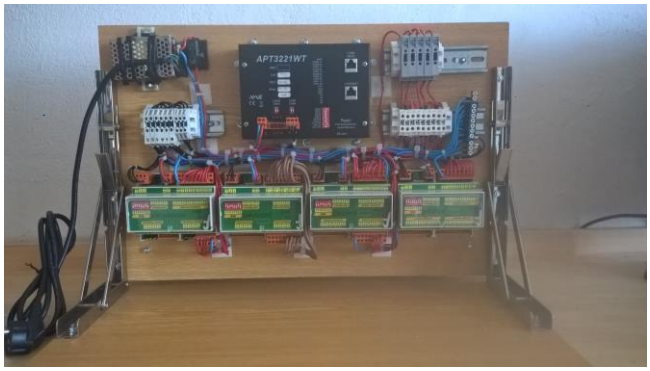
Pomocí těchto prvků se můžou průběžně nahrávat a vypisovat hodnoty vybraných proměnných uložených v paměti RAM řídícího systému (pokud byla v DetStudiu nastavena funkce Archivace). Proměnné lze zobrazovat jak ve formě tabulky, tak i ve formě grafu, které se dají kombinovat i s prvky Inspektor, Matice, nebo Časový plán (pro nastavení časů ukládání hodnot).

5. Konstrukce a komponenty výukového panelu

První částí bakalářské práce bylo navrhnout a sestrojit výukový panel obsahující řídící systém firmy Amit.



Obr. 25 Výukový panel – přední strana



Obr. 26 Výukový panel – zadní strana

5.1 Opěrná Konstrukce

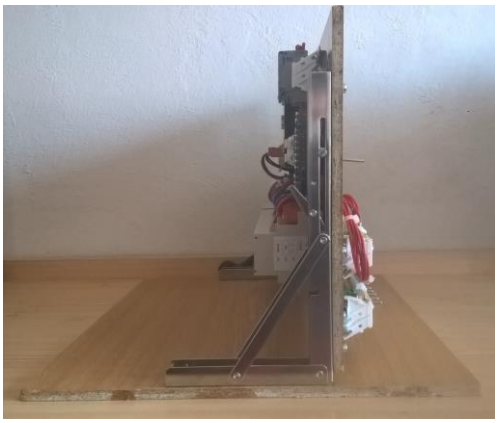
Základní opěrnou konstrukci tvoří dvě dřevěné desky spojené nerezovým sklápěcím držákem firmy Talamex.

První deska slouží pro upevnění jednotlivých komponentů jako je řídící terminál, moduly vzdálených vstupů a výstupů, zdroj napětí apod. Tato deska má rozměry 550x350x10 mm a jsou do ní vytvořené otvory pro uložení řídícího terminálu, vypínacího tlačítka šroubků a vedení vodičů mezi přední a zadní částí panelu. Ostatní komponenty jsou uchyceny buď pomocí DIN lišty (35 mm), nebo plastovými příchytkami (v případě že daný předmět nepodporuje uchycení DIN lištou).

Tloušťka desky 10 mm byla zvolena pro větší pevnost a stabilitu konstrukce. S většími rozměry však nastává problém s větší hmotností celého výukového panelu. Ta totiž ztěžuje přenášení a jinou podobnou manipulaci s výukovým panelem.

Druhá deska pak slouží zejména jako podstavec pro zajištění stability celého výukového panelu. Tato deska má rozměry 560x450x10 mm a je do ní vyvrtaných 6 otvorů pro šroubky k uchycení k ocelovému držáku. Rozměry desky sice zajišťují výukovému panelu více než dostatečnou stabilitu, ale opět přidávají i nadměrnou hmotnost, která je z pohledu přemísťování panelu nežádoucí.

Obě desky jsou pak spojeny nerezovým sklápěcím držákem firmy Talamex. Ten drží výchozí úhel 90°, avšak po menších úpravách ho lze nastavit i do libovolného úhlu mezi 0° a 90°. V tomto případě jsem zvolil úhel 40°, neboť tento úhel umožňuje pohodlnější práci na panelu. Zároveň je to nejnižší možný úhel, který lze nastavit a nepoškodit při tom některé komponenty umístěné na zadní straně. Při nižším úhlu než 40° se již můžou některé komponenty opírat do desky tvořící podstavec a držák tak ztrácí svoji funkčnost.



Obr. 27 Výukový panel – Svislá poloha

Obr. 28 Výukový panel – Sklon 40°

5.2 Komponenty

Řídící terminál

Stěžejní částí celého výukového panelu je řídící terminál APT3221WT firmy AMIT. Tento terminál zajišťuje zejména výpočetní výkon pro celý PLC systém a zpracovává program, který je v něm uložen. Terminál také obsahuje dotykovou obrazovku, kterou je rovněž možno naprogramovat a přizpůsobit si vlastním potřebám. Obrazovka pak může zobrazovat údaje jako například hodnoty na vstupech a výstupech, a pomocí dotykového displeje lze některé hodnoty i upravovat. Tento terminál podporuje komunikaci s okolím buď sériovou linkou (RS 232 nebo RS 485), nebo síťí Ethernet.

Obr. 29 Řídící terminál – Přední strana

Obr. 30 Řídící terminál – Zadní strana

Moduly vzdálených vstupů/výstupů

Terminál však neobsahuje žádné fyzické vstupy a výstupy do kterých by se připojily vstupní či výstupní zařízení. Proto výukový panel obsahuje tzv. moduly vzdálených vstupů/výstupů. Výhoda těchto modulů je v tom, že na jeden řídící terminál se dá připojit několik desítek těchto modulů dle vlastní volby. Nevýhoda jsou pak větší rozměry celé sestavy a obtížnější instalace. Výukový panel obsahuje čtyři různé moduly:

DM-DI24 – modul obsahující 24 digitálních vstupů s galvanickým oddělením po osmi (0 a 24 V ss./stř.)

DM-DO18 – modul obsahující 18 digitálních výstupů, s galvanickým oddělením po osmi (0 a 24 V ss./stř.)

DM-RDO12 – modul obsahující 12 reléových výstupů (maximální spínané napětí až 250 V ss./stř., maximální spínaný proud 6 A ss./stř.)

DM-UI8AO8U – obsahuje 8 univerzálních vstupů (0 – 5 V ss./stř., 0 – 10 V ss./stř., 0 – 20 mA ss./stř.) a 8 analogových výstupů (0 – 10 V ss./stř.)

Obr. 31 Moduly vzdálených vstupů/výstupů

Řídící terminál i moduly vzdálených vstupů/výstupů jsou napájeny napětím 24 V ss. a komunikují spolu sériovou linkou RS485 s protokolem ARION.

Napájení a sériové linky

O napájení se stará spínaný zdroj 230/24 V s výkonem 25 W od výrobce Mean Well RS-25-24. Tento zdroj dodává napětí 24 V a maximální proud 1.1 A což je pro výukový panel dostačující. Na vstupu zdroje se nachází tří žilový kabel s vidlicí na konci pro připojení do zásuvky 230 V. Každý z vodičů v tomto kabelů je lankového typu a má průřez 1,5 mm^2. Z výstupu tohoto zdroje jsou vyvedeny lankové vodiče červené a modré barvy s průřezem 1,5 mm^2 (podle norem

ISO), jenž přivádějí napájecí napětí do řídícího terminálu a modulů vzdálených vstupů a výstupů. Každý z těchto komponent je pak samostatně jištěn válcovou pojistkou umístěnou v pojistkové svorce. U modulů vzdálených vstupů a výstupů má pojistka jmenovitý proud 250 mA a v případě řídícího terminálu 2 A. Další komponenty (jako například periférie pro řízení vstupů a vizualizaci výstupů na přední straně panelu) jsou pak jištěny stejnou pojistkou jako modul ke kterému jsou připojeny. Tyto komponenty jsou kvůli malým proudům napájeny vodiči s průřezem pouze 0,75 mm^2 (což je stále více než dostačující). Výjimkou je modul DM_RDO12 (modul reléových výstupů), jehož vodiče na výstupech mají opět průřez 1,5 mm^2 a jsou hnědé barvy. Důvodem je dimenzování spínacích kontaktů tohoto modulu. Ty jsou určeny na 250 V ss./stř. a 6 A ss./stř., čemuž musí odpovídat i větší průřez. Celý výukový panel lze vypnout pomocí tlačítka EATON M22-K10 s ovládací hlavicí EATON 216867 M22-WRK, které odpojí celý výukový panel od zdroje napětí.

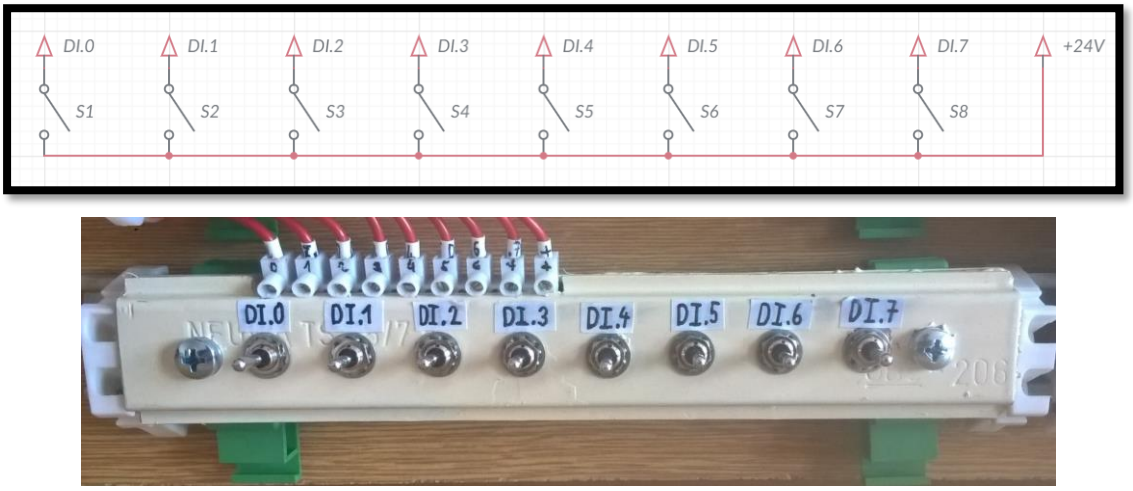
Komunikace mezi řídícím terminálem a moduly vzdálených vstupů a výstupů je provedeno sériovou linkou RS 485. Ta je vedena lankovými vodiči černé barvy s průřezem 0,5 mm^2, které jsou propojeny v řadových svorkách, aby bylo možné komunikovat se všemi zařízeními najednou.

Ovládání vstupů a vizualizace výstupů

Pro pohodlnou manipulaci se vstupy a zobrazování hodnot na výstupech jsem sestrojil několik druhů periférií, které lze připojit na vstupy a výstupy jednotlivých modulů vzdálených vstupů/výstupů a sledovat na nich chod programu. Tyto části jsou narozdíl od předchozích komponentů (kromě obrazovky řídícího terminálu), umístěny na přední straně výukového panelu a připevněny na DIN lištu pomocí speciálních příchytek. Kdyby tedy mnou navržené periférie nevyhovovaly, nebo se porouchaly, lze je z DIN lišty snadno odejmout a nahradit například klasickými řadovými svorkami. Proto také tyto části nejsou trvale připojeny k modulům vzdálených vstupů/výstupů ale jsou vybaveny svorkovnicí pro snadné odpojení vodičů. Kryty těchto periférií jsou tvořeny převážně z plechu a případné mezery jsou zaplněny dřevěnými částmi, aby se minimalizovalo znečištění či vniknutí cizích těles k elektrickým obvodům, které by je mohly poškodit či narušit jejich funkčnost. Všechny kryty jsou nakonec opatřeny ochranným nátěrem slonovinové barvy, kvůli ochraně před korozí a estetické působnosti. Tyto periférie jsou však vyrobeny podomácku, což se projevuje zejména jejich vzhledem. Výukový panel obsahuje následující periférie:

Periférie pro ovládání digitálních vstupů

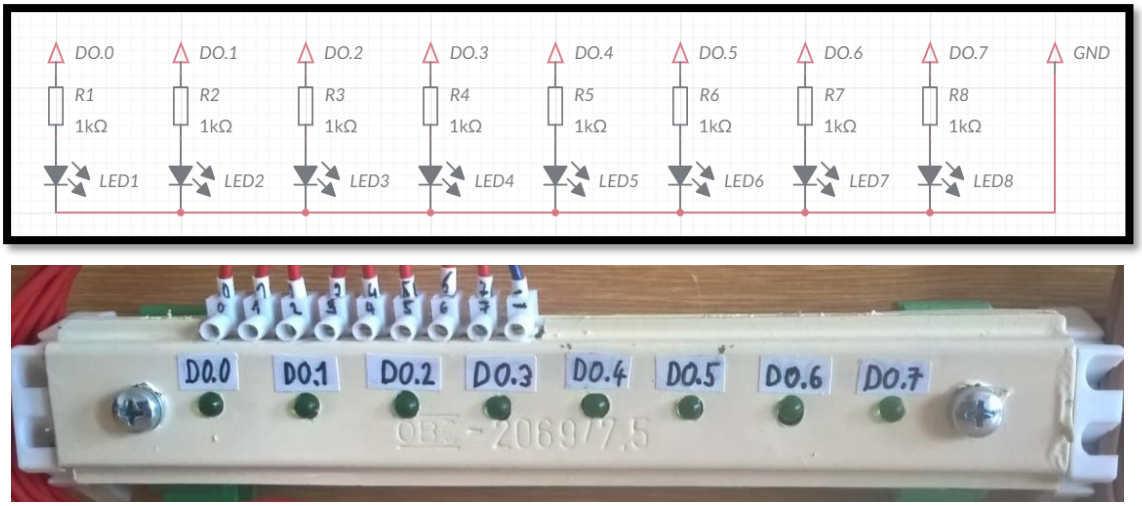
Tato periférie obsahuje 8 přepínačů typu ON-OFF-(ON). Tyto přepínače mají tedy i polohu ve které se chovají jako tlačítka (po uvolnění se samy vracejí do polohy OFF), což je pro některé úlohy velice praktické. Těmito přepínači je pak řízen přívod napětí na jednotlivých vstupech modulu DM_DI24 (modul obsahující digitální vstupy), které znázorňuje logické nuly a jedničky na těchto vstupech.



Obr. 32 Periférie pro řízení dig. vstupů

Periférie pro zobrazování digitálních výstupů

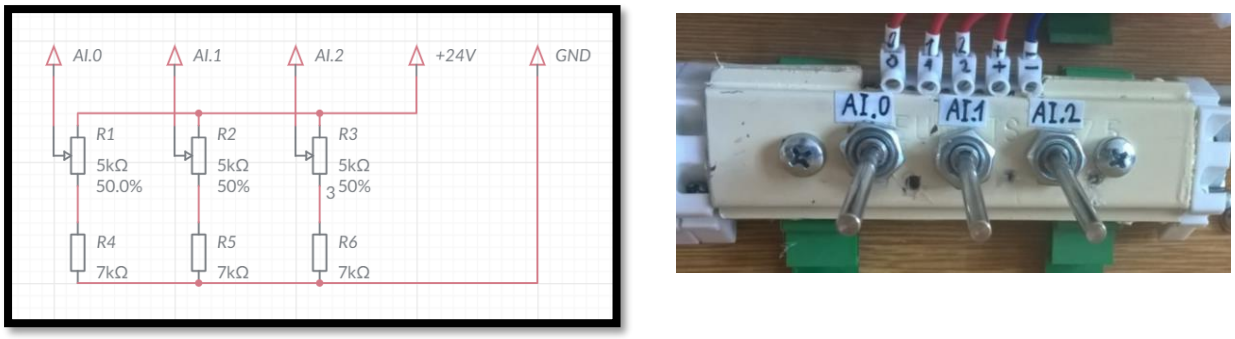
Tato periférie obsahuje 8 LED diod zelené barvy o velikosti 5 mm. Každá z těchto LED diod je připojena na výstup modulu DM_DO18 (modul obsahující digitální výstupy) a rozsvítí se tak, když je na daném výstupu napětí 24 V (log. 1). To umožňuje pohodlně sledovat hodnoty na digitálních výstupech. Jelikož LED diody potřebují ke svícení cca. 4 V a 20 mA a modul má na výstupech 24 V (při log. 1) je třeba před každou diodu sériově připojit rezistor s odporem 1 kΩ.



Obr. 33 Periférie pro zobrazování dig. výstupů

Periférie pro ovládání analogových vstupů

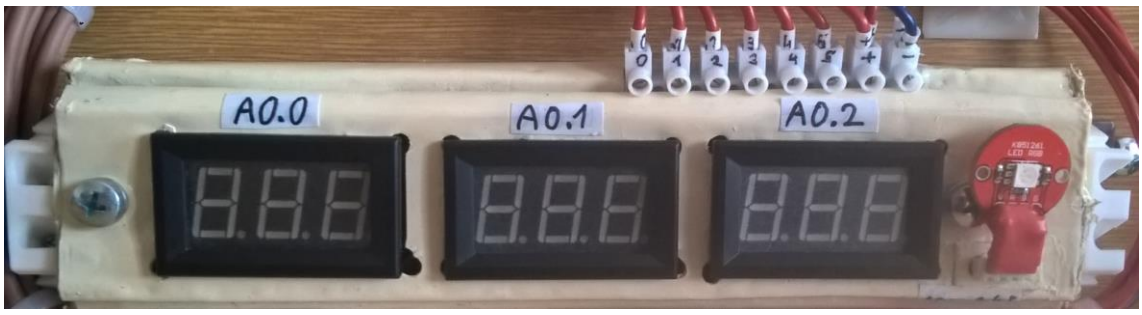
Tato periférie obsahuje 3 potenciometry s odporem v rozmezí 0-5 kΩ. Ke každému potenciometru je sériově připojen rezistor s odporem 7 kΩ, který v kombinaci s potenciometrem tvoří napěťový dělič. Pomocí tohoto napěťového děliče pak lze otáčením potenciometru měnit napětí na jednotlivých analogových vstupech modulu DM_UI8AO8U (modul univerzálních vstupů a analogových výstupů) v rozmezí 0-10 V.



Obr. 34 Periférie pro řízení an. vstupů

Periférie pro zobrazování analogových výstupů

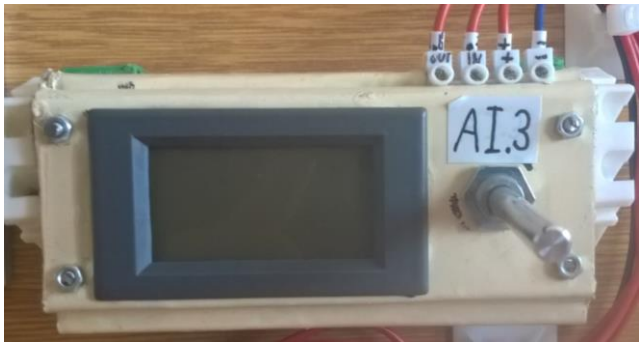
Tato periférie obsahuje 3 panelové voltmetry s rozsahem 0-30 V a RGB LED diodu (pro lepší znázornění úlohy 3.3). Voltmetry jsou připojeny na jednotlivé výstupy modulu DM_UI8AO8U (modul univerzálních vstupů a analogových výstupů) a zobrazují tak hodnoty na analogových výstupech. Vstupy RGB LED diody jsou opět připojeny na jednotlivé výstupy modulu DM_UI8AO8U tak, aby se každá barva diody mohla ovládat nezávisle a barvy se tak daly libovolně kombinovat. Z podobného důvodu jako v případě prvku digitálních výstupů je každý vstup RGB LED diody sériově připojen na rezistor s odporem 1 kΩ.



Obr. 35 Periférie pro zobrazování an. výstupů

Periférie pro simulaci regulované soustavy

Tato periférie je určená speciálně pro úlohu 3.5 (PID regulátor). Úloha spočívá v tom, že při změně elektrického odporu, PID regulátor musí udržet konstantní proud. To provede změnou napětí (podle Ohmova zákona I = U/R). Periférie obsahuje panelový ampérmetr s rozsahem 0-999 mA a potenciometr s odporem v rozmezí 0-1 kΩ. Tento potenciometr je připojen mezi jedním ze vstupů a jedním z výstupů modulu DM_UI8AO8U tak, aby se otáčením potenciometru měnil odpor, a tedy i proud na vstupu modulu a regulátor tedy musel upravit napětí na výstupu, aby tuto změnu kompenzoval. Aby se elektrický odpor pohyboval v rozumných mezích je k němu sériově připojen ještě rezistor s odporem 800 Ω (celkový odpor se tedy pohybuje v rozmezí 800–1800 Ω). Pro kontrolu, že regulátor pracuje správně, je zde umístěn panelový ampérmetr, který zobrazuje hodnotu elektrického proudu mezi vstupem a výstupem.



Obr. 36 Periférie pro úlohu 3.5 (PID regulátor)

Pro modul DM_RDO12 se mi nepovedlo vymyslet vhodnou periférii. která by poskytovala dobrou představu o funkci tohoto modulu. Proto jsem zde připevnil pouze řadové svorky pro připojení externích zařízení (například motorů, pro které jsou také určeny navržené úlohy).

Obr. 37 Řadové svorky pro připojení externího zařízení k modulu DM_RDO12

6. Demonstrační úlohy

Dalším cílem této práce je sestavit sadu demonstračních úloh, na kterých by studenti procvičili své dovednosti z PLC programování. Úlohy jsou navrženy tak, aby v nich byly obsaženy všechny základní prvky a jejich využití v různých situacích. Po vyřešení všech úloh by tedy měl mít student dobrou představu o PLC programování a měl by být schopen řešit reálné řídící problémy.

Úlohy jsou sepsány ve zvláštní příručce přiložené k této práci. Kvůli její délce a struktuře jsem však usoudil, že je vhodnější ji přidat v podobě přílohy. Tato příručka obsahuje jednak všechny úlohy vytvořené pro výukový panel, ale i stručný návod, jak se orientovat v programu DetStudio a jak nastavit vstupy/výstupy před začátkem samotného programování. I když DetStudio poskytuje více jazyků programování, úlohy v příručce jsou z drtivé části řešeny v reléových schématech, neboť tento způsob programování je velice intuitivní a v praxi i nejpoužívanější.

Každá úloha má následující strukturu:

Zadání

Zde je popsán spíše jen příklad, k čemu by mohl být výsledný program použit v praxi. Používají se zde někdy slangové výrazy, a ne každý může rozumět co od programu přesně očekávat. Studenti by však měli získat alespoň hrubou představu o funkci programu.

Specifikace

Zde jsou již popsány detailnější požadavky, jak by měl program fungovat. Je tu popsáno, jak by měl program reagovat na změnu jednotlivých vstupů či jakých hodnot můžou nabývat čítače, časovače apod. Na studentech je pak vymyslet, jakým způsobem tyto požadavky zrealizovat.

Doporučené moduly

Jelikož se v DetStudiu programuje pomocí tzv. modulů, kterých je zde skutečně požehnaně, vytvořil jsem u každé úlohy krátký výpis modulů, které bych při řešení použil já. Chtěl jsem tak zjednodušit vyhledávání modulů z dlouhého seznamu, které DetStudio nabízí. Zároveň tento výpis slouží jako menší nápověda, jak úlohu řešit.

Řešení

Zde je popsán podrobný postup, jak úlohu vyřešit. Každá úloha může mít samozřejmě mnoho možných řešení, zde jsem se však snažil vymyslet to nejjednodušší. Pokud však student vymyslí odlišný způsob, jak úlohu vyřešit, je to samozřejmě v pořádku. V každém řešení je tedy krok po kroku rozepsán a vysvětlen postup tak, aby se v něm každý dobře orientoval a pochopil, jak daný program funguje. Jelikož jsem se snažil postup popsat velmi detailně, je tato část taky nejobsáhlejší. Jedno řešení může obsáhnout i několik stránek, což se taky odráží na délce celé příručky. Tato část se navíc nachází vždy na další stránce než předešlé části (Zadání, Specifikace, Doporučené moduly), aby student náhodou neviděl řešení dříve, než vůbec začne úlohu řešit.

Všechna mnou navržená řešení k těmto úlohám jsou vyzkoušena a mohu i dodat soubory s fungujícími programy, které si studenti mohou detailně prohlédnout. Abych studentům ulehčil rutinní části, připravil jsemi soubor nazvaný Přednastvení_úloh.DSO, který obsahuje již přednastavenou komunikaci, vstupy, výstupy a naprogramovanou obrazovku. Studenti se tedy mohou pustit rovnou do řídící části a ušetřit tak čas.

Celá příručka pak má následující obsah.

Nastavení komunikace v síti ARION nastavení vstupů/výstupů

Tato kapitola popisuje základy softwaru DetStudio. Obsahuje návod, jak krok po kroku založit a nastavit nový projekt, proměnné, aliasy a procesy. Jelikož se v DetStudiu programování neprovádí obvyklým způsobem, je zde popsán i způsob vkládání tzv. modulů jejíž prostřednictvím se zde tvoří program. Výukový panel využívá tzv. moduly vzdálených vstupů/výstupů, proto je zde popsán stručný návod, jak nastavit komunikaci mezi nimi a řídícím terminálem.

Návrh obrazovky

Kapitola obsahuje rychlý tutoriál, jak navrhnout a naprogramovat obrazovku řídícího terminálu. Je zde zejména ukázáno, jak zobrazit hodnoty na vstupech a výstupech, což může nahradit některé periférie.

Po této kapitole již následují demonstrační úlohy.

Úlohy jsou děleny do tří skupin: Úlohy s digitálními vstupy a výstupy (8 úloh), úlohy s modulem DM-RDO12 (2 úlohy) a úlohy s analogovými vstupy a výstupy (5 úloh). Celkový počet úloh je tedy 15.

Úlohy s digitálními vstupy a výstupy

Tyto úlohy se zaměřují zejména na zvládnutí práce s digitálními hodnotami a booleovy logiky. Této části by se měla věnovat největší pozornost, neboť právě práce s digitálními hodnotami tvoří základ PLC programování. Jsou zde proto v první řadě zastoupeny úlohy na procvičení logických funkcí (AND, OR, NOT, …), poté jsou do úloh postupně přidávány prvky jako klopné obvody, čítače, časovače atd. Porozumění všem těmto prvkům a jejich využití je velice důležité pro další pokračování v PLC programování.

V úlohách se používají pouze digitální vstupy a výstupy, které obsahují moduly vzdálených vstupů/výstupů DM-DI24 a DM-DO18. Tyto vstupy a výstupy jsou pak připojené na sadu přepínačů a LED diod (periférie na přední straně panelu), na kterých lze odzkoušet a sledovat, jestli program pracuje správně.

Úlohy s modulem DM-RDO12 (spínací relé 250 V)

Na tyto úlohy jsem se zaměřil pouze okrajově, neboť je lze snadno nahradit klasickými digitálními výstupy. V praxi jsou však situace, kdy použití tohoto modulu je praktičtější a méně nákladnější. Navíc je dobré znát princip řízení pomocí spínání a rozpínání reléových kontaktů.

Úlohy s analogovými vstupy a výstupy

V současné době, kdy se PLC řízení používá ve stále více a více komplexnějších případech, se již neobejdeme pouze s digitálními hodnotami, neboť by v některých případech mohlo být řízení pomocí digitálních vstupů a výstupů nesmírně složité, nebo dokonce nemožné. Proto je třeba umět zpracovat i analogové hodnoty. Analogové a digitálními vstupy a výstupy jsou v úlohách většinou kombinovány, pro snadnější pochopení jednotlivých prvků a funkcí.

Pro tyto úlohy se používá zejména modul vzdálených vstupů/výstupů DM-UI8AO8U, který obsahuje analogové vstupy a výstupy. Analogové vstupy jsou připojeny na 5kΩ potenciometry s 7, kΩ rezistory, které fungují jako napěťový dělič a umožňují tak měnit napětí na vstupech modulu v rozmezí 0 - 10 V. Analogové výstupy jsou pak připojeny na digitální panelové voltmetry, které měří napětí na výstupech modulu.

Každá z těchto třech částí obsahuje krátký úvod. Jedná se rychlý souhrn, na co jsou úlohy zaměřeny a co je v nich obsaženo. Dále je v nich popsáno jakési "přednastavení" celého projektu. To obnáší převážně rutinní části, které je potřeba vykonat na začátku každého projektu. Patří mezi ně například nastavení komunikace a vstupů/výstupů, které jsou pro každou úlohu téměř identické.

Úlohy využívají jen zlomek vstupů a výstupů, než je obsaženo v periféríích na přední straně výukového panelu. Například periférie pro řízení dig. vstupů obsahuje 8 přepínačů, zatímco v úlohách jsou využívány nanejvýš 4. Větší počet vstupů a výstupů byl vytvořen pro případ, že by na výukovém panelu měli být zkoušeny jiné úlohy než mnou navržené, nebo se některé vstupy/výstupy porouchaly.

7. Závěr

Cílem bakalářské práce bylo navrhnout a sestrojit výukový panel pro předmět Automatizace obsahující PLC firmy Amit. K tomuto výukovému panelu se dále měla vypracovat sada demonstračních úloh.

Snažil jsem se, aby konstrukce výukového panelu byla pevná a odolná vůči poškození. To se však neblaze projevilo na její hmotnosti, která stěžuje případné přemísťování výukového panelu.

Mnou navržené periférie na přední straně výukového panelu považuji za velmi užitečné. Lze s nimi pohodlně a přehledně řídit a sledovat chod programu, bez nutnosti dalších zařízení či naprogramované obrazovky řídícího terminálu. Bez těchto periférií by mohlo být odzkoušení programu velmi obtížné či komplikované.

Při testování se ukázalo že periférie pro úlohu 3.5 (PID regulátor) má technický problém. Konkrétně se jedná o panelový ampérmetr, který významně zkresluje měřený proud na vstupu. Tento problém se mi nepodařilo odstranit a byl jsem nucen ampérmetr odpojit. Úlohu je stále možné s řešit pouze s potenciometrem obsaženým v této periférii, ale el. proud v obvodu je nutné sledovat na obrazovce řídícího terminálu, nebo na počítači pomocí softwaru DetStudio či ViewDet.

I když jsou řešení v úlohách podrobně rozepsané, množství úloh se mi zdá stále nedostačující k tomu, aby studenti plně porozuměli všem možnostem PLC řídících systémů. V úlohách jsou sice zastoupeny všechny možné prvky tvořící základ PLC programování, je ale třeba PLC programování potrénovat na více úlohách, znázorňujících různé situace. Více úloh by ovšem znamenalo rozšířit obsah už takhle dlouhé příručky.

Řešení v úlohách jsem se snažil udělat co nejpodrobnější, aby bylo každému zřejmé, jak jsem k danému výsledku došel. Přesto se obávám, že se stále mohou vyskytnout studenti, kteří mu dostatečně neporozumí. I když se PLC programování jeví jako velmi intuitivní, stále vyžaduje určitou schopnost logického uvažování a talent, který nemusí každý vlastnit.

8. Zdroje

Seznam použité literatury

1. VORÁČEK Rudolf. Automatizace a automatizační technika II, Praha: Computer Press, 2000. ISBN 80-7226-247-5.
2. KOLÁŘ Josef, PROKOPOVÁ Zuzana, ŠMEJKAL Ladislav. Programování PLC
3. Co se skrývá pod označením PLC
   https://automatizace.hw.cz/co-se-skryva-pod-oznacenim-plc
4. PLC program – Úvod
   http://plc-automatizace.cz/knihovna/program.htm
5. DetStudio – Průvodce první aplikací, Návod na obsluhu
   https://amitomation.cz/produkt/software/detstudio-vyvojove-prostredi/#tab2
   - detstudio_g_cz_104.pdf
6. ViewDet – Průvodce první aplikací, Návod na obsluhu
   https://amitomation.cz/produkt/software/viewdet-servisni-program/#tab2
   - viewdet_g_cz_100.pdf
7. Amit, spol. s.r.o. – Automation
   https://amitomation.cz/firma/o-nas/
8. Historie firmy Amit v několika bodech
   https://amitomation.cz/firma/historie/
9. Produkty společnosti Amit
   https://amitomation.cz/produkty/

Seznam obrázků

Přílohy (Sada úloh)

Následuje sada úloh, kterou jsem kvůli její délce a struktuře dodal v podobě přílohy.

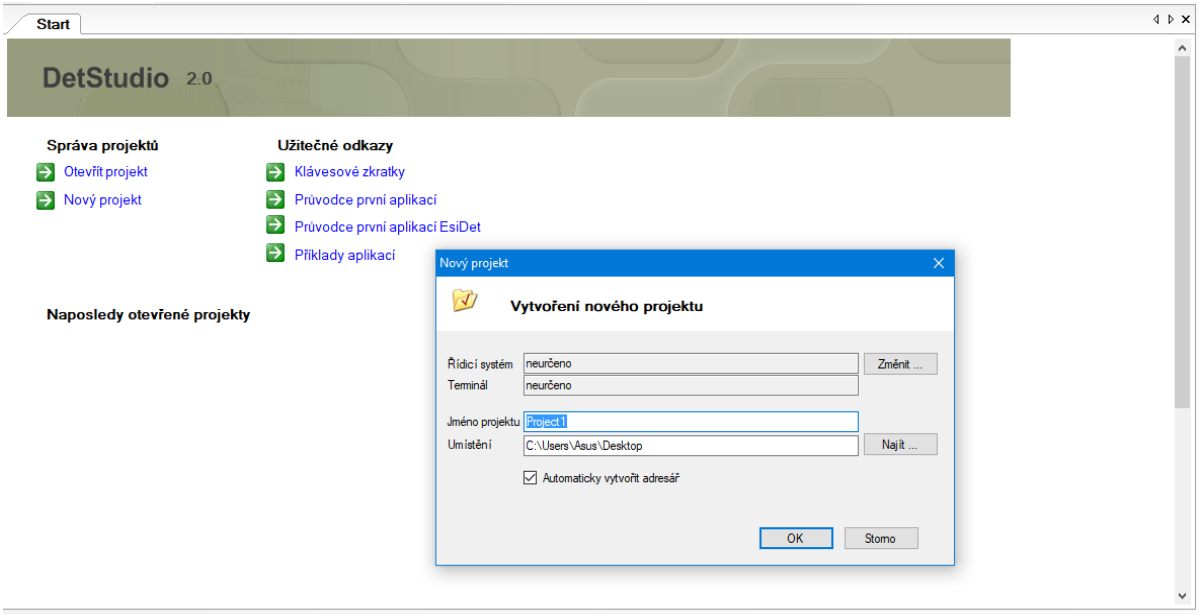
Sada úloh k PLC firmy Amit

Obsah

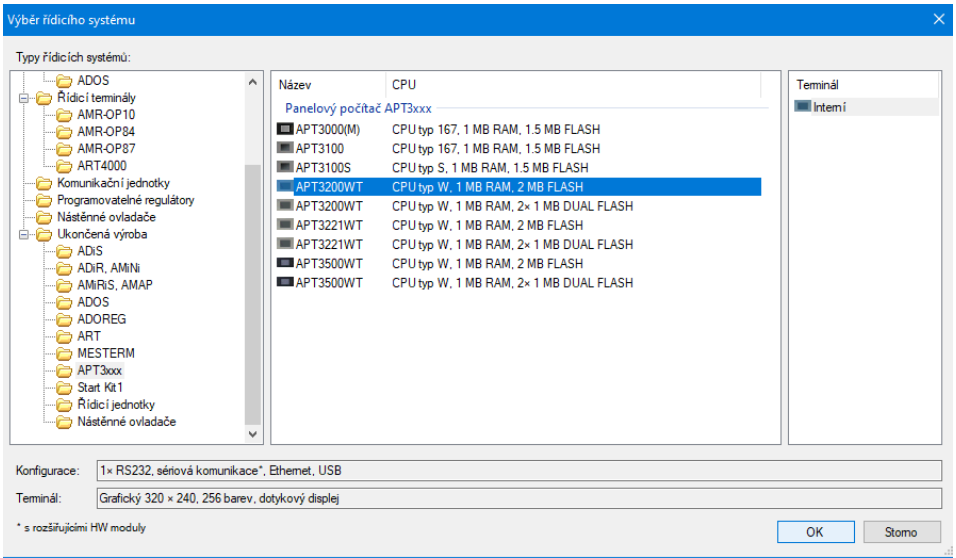
Nastavení komunikace v síti ARION a definování vstupů/výstupů

Nyní si ukážeme základy ovládání programu DetStudio a provedeme přípravy před zahájením programování.

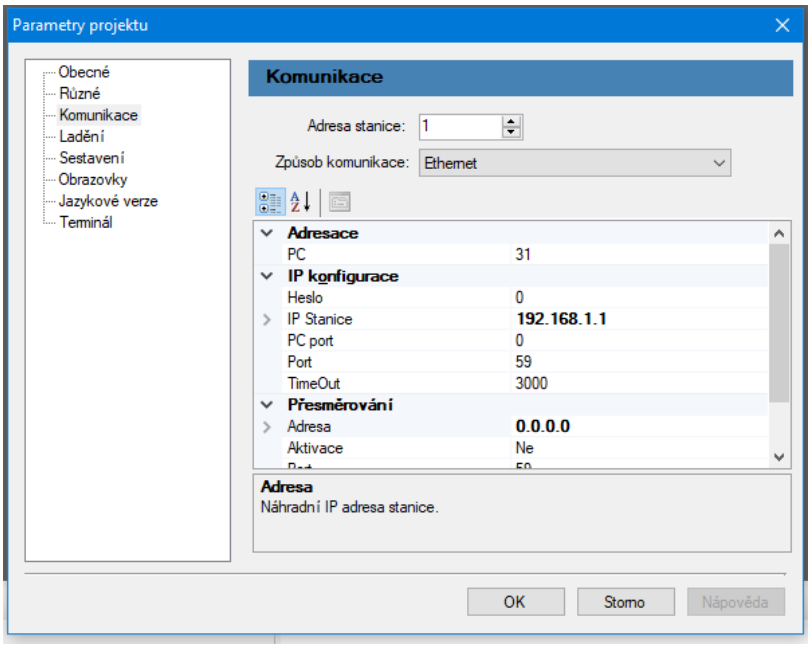
Po otevření DetStudia klikněte na ikonu Nový projekt. Otevře se následující okno.



Zde je nutné vyplnit Jméno projektu, Umístění a používaný Řídící systém. Po kliknutí na **Změnit** u řídícího terminálu najděte v záložce nalevo cestu **Ukončená výroba/APT3xxx** a vyberte terminál **APT3200WT** (některé řídící systémy jsou k dispozici ve dvou režimech FLASH paměti, použijte klasický režim 2 MB FLASH).



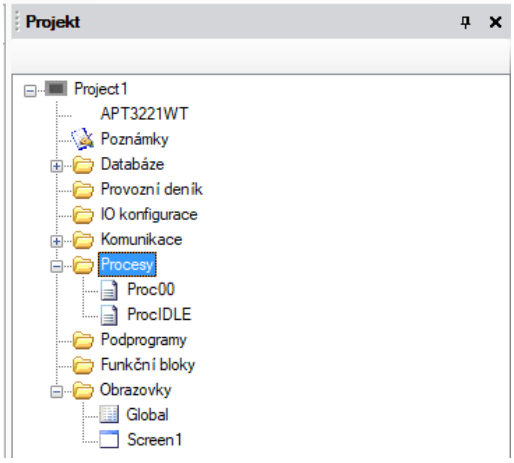
Po vytvoření projektu se otevře okno s jeho nastavením. Zde v záložce Komunikace nastavte Způsob komunikace na **Ethernet** a upravte požadované parametry. Tímto je nastavena komunikace s PC.



Tímto jsou nastaveny základní parametry a můžeme se začít seznamovat s prostředím DetStudia.

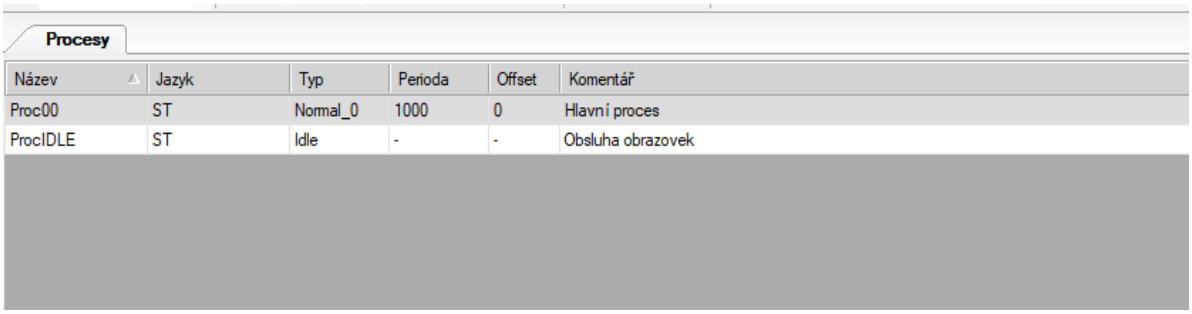
Na pravé straně obrazovky se nachází okno **Projekt**. V něm se nachází většina funkcí, které DetStudio nabízí. Jelikož využíváme moduly vzdálených vstupů a výstupů, tak si první řadě nastavíme komunikaci mezi těmito moduly a řídícím systémem.

V okně Projekt rozklikněte ikonu **Procesy**.



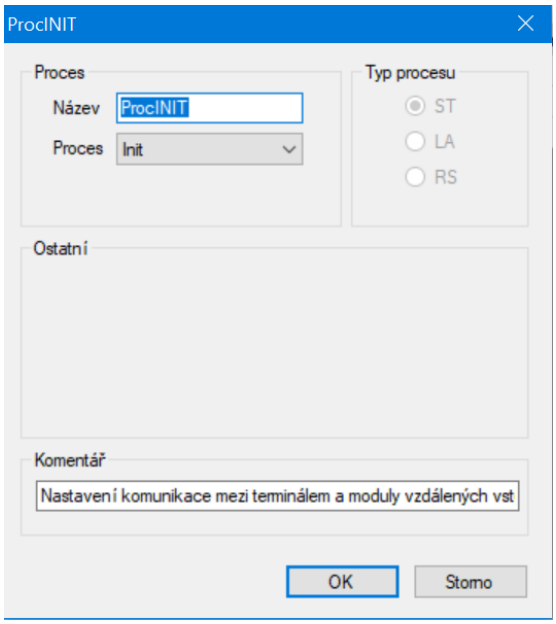
Procesy tvoří stěžejní prvek DetStudia. V procesech se uskutečňuje programování řídícího systému. Každý proces lze jednotlivě naprogramovat na různé funkce (načtení vstupů, zápis výstupů, řízení, komunikace, …) a skládat dohromady v jeden celek.

Po otevření se zobrazí okno s existujícími procesy (při vytvoření projektu se automaticky vytvoří dva procesy).

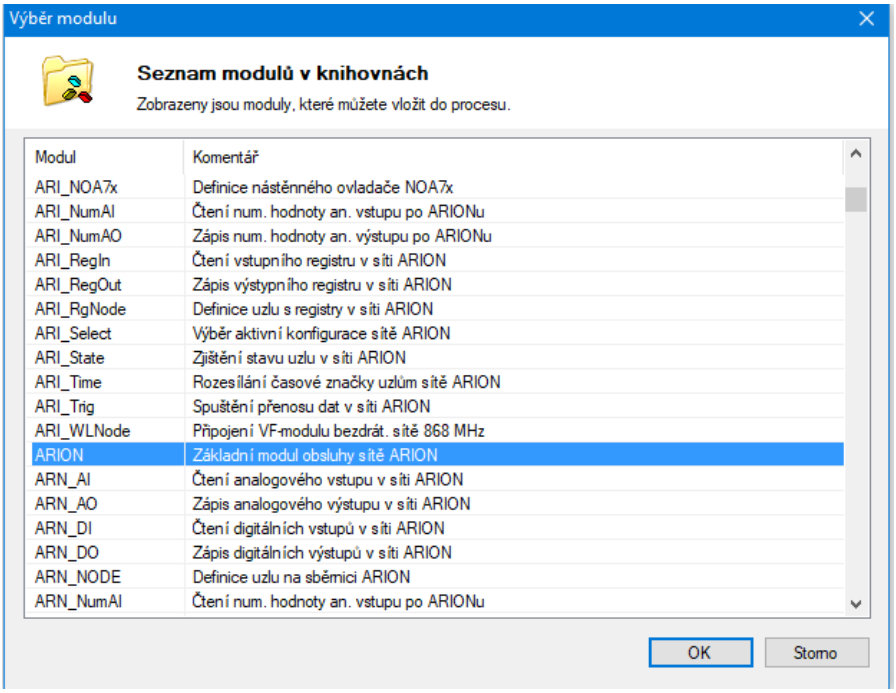


Nyní vytvoříme proces, který bude sloužit pro nastavení komunikace. Nový proces lze založit buď v **Toolboxu** na levé straně obrazovky, klávesou **Insert** nebo volbou **Přidat nový proces** z nabídky vyvolané pravým tlačítkem myši (případně lze editovat Proc00, který se vytvořil automaticky při vzniku projeku).

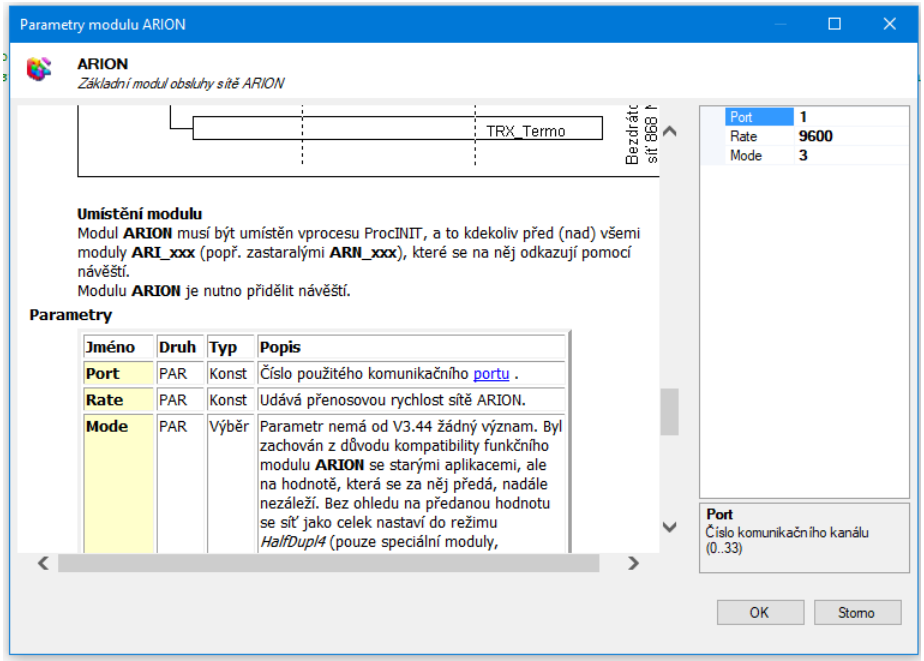
Proces pojmenujeme **ProcINIT** a bude typu **ST** (strukturovaný text). Tento proces bude sloužit k inicializaci komunikační sítě mezi terminálem a moduly vzdálených vstupů/výstupů, proto je nutné přiřadit mu prioritu Init.



Programování se v DetStudiu provádí pomocí vkládání tzv. modulů. Modul se vkládá z nabídky **Toolbox**, nebo pomocí klávesové zkratky **Ctrl+i**, kdy se u každého modulu zobrazí, k čemu slouží (doporučeno). Tímto způsobem nyní vložíme modul **ARION**.



Po rozkliknutí se zobrazí okno pro nastavení parametrů modulu a podrobná nápověda, co daný modul dělá a jak jej nastavit. Nastavte: **Port=1, Rate=19200**.



Tímto je položen základ komunikace sítě ARION. Jelikož může být v systému sítí více, je ještě třeba ji nějak odlišit. To uděláme pomocí tzv. návěstí, které bohužel není v základu modulu a je potřeba ho dopsat ručně. Na začátek řádku s modulem proto napište např.:1234. Řádek pak bude vypadat nějak takto.

```
:1234 ARION 1, 19200, 3
```

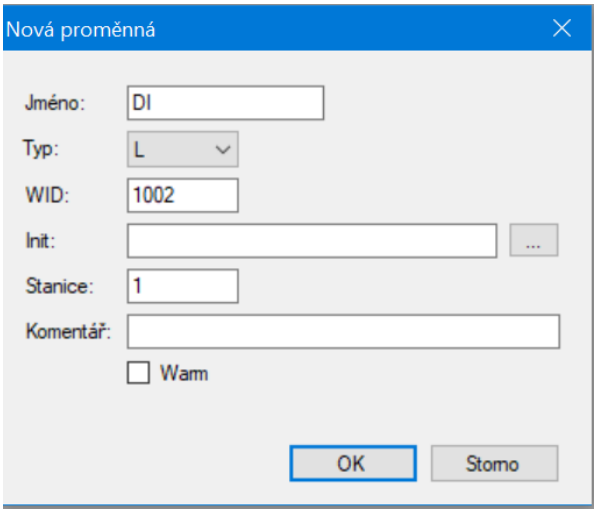
Podobným způsobem do této sítě připojíme moduly vzdálených vstupů a výstupů. To si ukážeme na modulu DM-DI24 (dig. vstupy).

Vložíme proto do samého procesu modul **ARI_DINode** a nastavíme: **Arion=1234 (návěstí sítě), Address= 11 (*adresa nastavená na připojeném zařízení)*, SigCount=24 (*počet vstupů)*** (ostatní parametry necháme být). Proces pak bude vypadat takto.

```
:1234 ARION 1, 19200, 3

ARI_DINode :1234, 11, 100, 2000, 24
```

Nyní si ukážeme, jak vstupy/výstupy ukládat do proměnných/aliasů a jak s nimi poté pracovat.

V okně Projekt otevřeme **Proměnné**. Z nabídky **Toolboxu**, nebo klávesou **F12** vytvoříme novou proměnnou typu **Long** (nazveme jí např. **DI**). Do této proměnné budeme ukládat stav vstupů z modulu vzdálených vstupů.



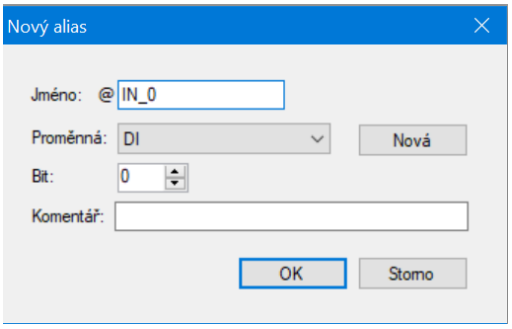
Parametr **WID** je identifikační číslo proměnné. Generuje se automaticky a nedoporučuje se ho přenastavovat. Parametr **Init.** je výchozí hodnota proměnné (pokud nebude parametr zadán, bude nastaven na 0). Parametr **Stanice** je adresa zařízení které s proměnou pracuje (ponecháme 1). Parametr **Warm** resetuje proměnou do výchozího stavu v případě výpadku nebo restartu řídícího systému (když tato volba není zaškrtnutá systém si po restartu bude pamatovat poslední známou hodnotu).

Nyní vytvoříme **proces** typu **ST** (pojmenovaný např. **Vstupy**), který bude sloužit na načtení vstupů uloží je do proměnné (Procesu ponecháme prioritu **Normal_0** a nastavíme periodu **100 ms**). Do procesu vložíme modul **ARI_DigIn** (načtení dig. vstupů ze sítě ARION). Nastavíme **Address=11**, **Variable=DI** (pokud jste proměnou pojmenovali jinak, zadejte váš název).

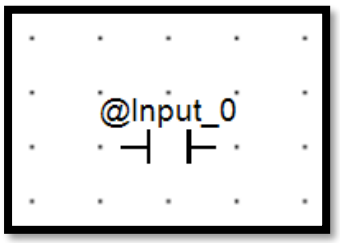
```
ARI_DigIn 11, 0, DI, 0x00000000
```

Proměnná do které jsme nahráli vstupy má 32 bitů (typ Long). Modul DM-DI24 má 24 fyzických dig. vstupů. Tento řádek nám uloží každý vstup na jeden bit proměnné (jelikož má proměnná více bitů než je potřeba, tak zbývající bity zůstanou nevyužité). Abychom mohli pohodlně manipulovat s každým bitem zvlášť a načítat tak z ní vstupy jednotlivě, je vhodné využívat tzv. Aliasy.

Alias je jakýsi odkaz na jeden bit v určité proměnné. **Aliasy** otevřeme v okně **Projekt**. Nový Alias založíme z **Toolboxu** nebo klávesovou zkratkou **F+12**. Nyní vytvoříme Alias (nazvaný např. @In_0), který bude sloužit na načítání vstupu ukládaném na “nultém“ bitu proměnné DI.



Nyní vytvoříme **proces** typu **RS** (reléové schéma), ve kterém budeme programovat řídící část. V procesu vytvoříme modul **LD** (načítání bitu) a do parametru **Value** vložíme Alias **Input_0**. Nyní máme vytvořený prvek, který bude reagovat na změnu vstupu modulu DM-DI24 (modul vzdálených vstupů).

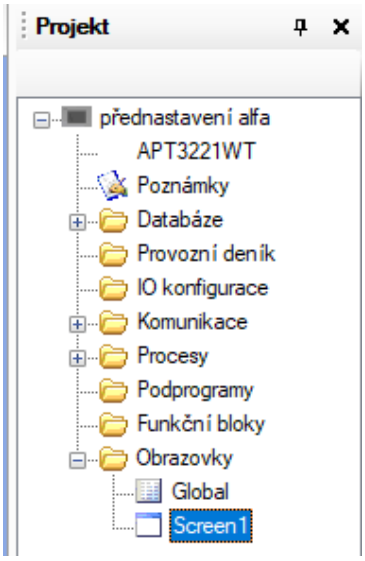


Tímto bych ukončil úvodní část a pustil bych se do řešení úloh.

Návrh obrazovky

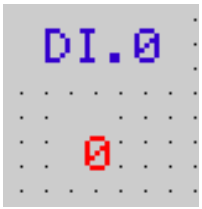
Nyní si ještě ukážeme, jak v DetStudiu naprogramovat obrazovku.

V okně Projekt rozklikněte ikonu **Obrazovky/Screen1**.

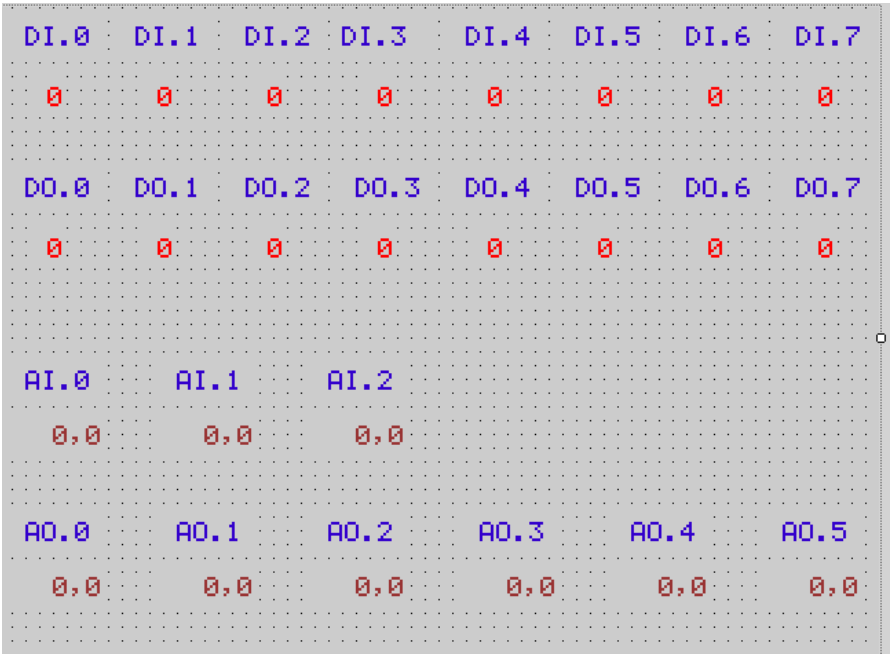


Zobrazí se plocha reprezentující obraz obrazovky. Do ní nyní budeme vkládat prvky z Toolboxu. Začneme prvkem Label. V okně vlastnosti se zobrazí parametry tohoto prvku. Upravíme tedy parametr text a napíšeme do něj například DI.0.

Nyní vložíme prvek BitSwitchKey a do parametru Variable nastavíme Bit proměnné (Alias) ve které se nám ukládá hodnota ze vstupu (v mém případě DI.0). Do parametrů TextFalse a TextTrue napíšeme 0 a 1. Pro leší přehlednost si můžeme upravit i barvy ForeColourTrue a ForeColourFalse.



Na obrazovce nyní můžeme sledovat hodnotu na vstupu DI.0. To můžeme udělat se všemi vstupy a výstupy a sledovat tak chod celého programu na obrazovce. Konečná obrazovka pak může vypadat takto.



Pokud chceme využívat dotykouvou obrazovku, můžeme použít i prvky jako ToogleButton či BitSwitchButton.

Pozor: Pokud nechcete obrazovku programovat, smažte ikonu Screen1 v okně Project. Pokud ponecháte obrazovku nenaprogramovanou, můžou se vyskytnout problémy při kompilaci.

Tímto bych ukončil úvodní část a pustil bych se do řešení úloh.

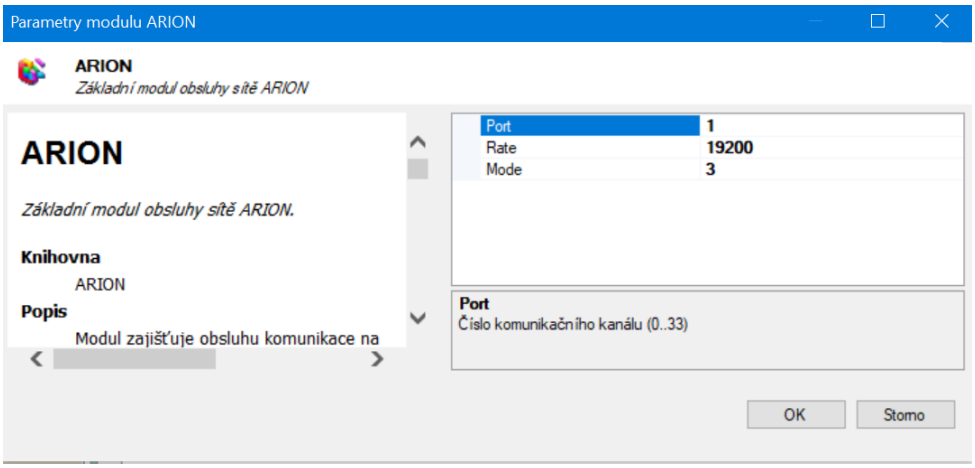
1. Práce s digitálními vstupy a výstupy

Tyto úlohy jsou zaměřené zejména na použití boolean logiky, časovačů, čítačů a klopných obvodů. Tyto prvky tvoří základ PLC programování a je velice důležité jim porozumět, proto jim také věnujeme největší pozornost.

Příprava

Nyní si ukážeme část programu, která je totožná u každé následující úlohy. Jedná se především o konkrétní nastavení komunikace, vstupů a výstupů.

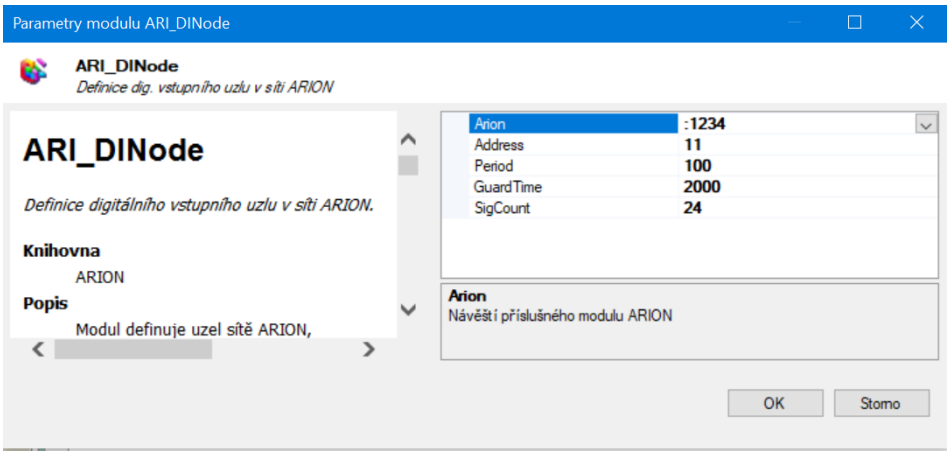
V první řadě do procesu ProcINIT (viz. nastavení komunikace v síti ARION a definování vstupů a výstupů), přidáme uzly (moduly vzdálených vstupů/výstupů) s digitálními vstupy a výstupy. Vložíme tedy do procesu modul ARION a nastavíme následovně.

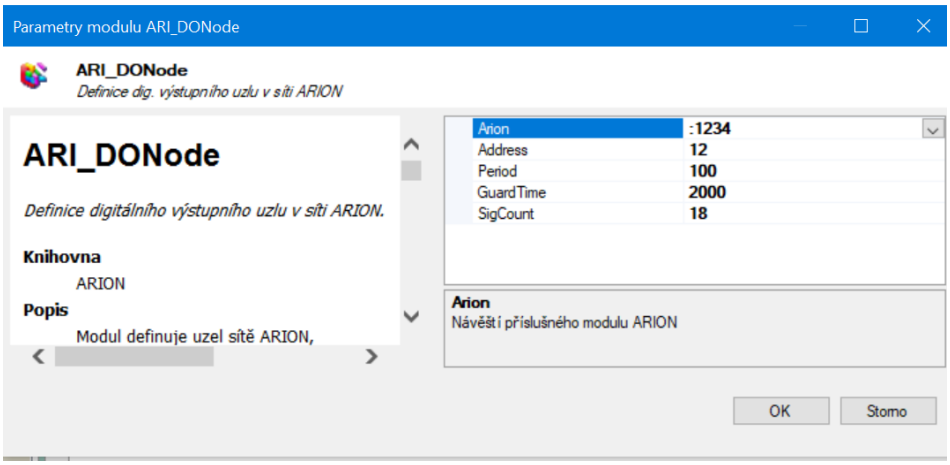


Po vytvoření modulu mu přidáme návěstí (např.:1234).

```
:1234 ARION 1, 19200, 3
```

Poté přidáme modul ARI_DINode a ARI_DONode, které inicializují dig. vstupy a výstupy. Nastavíme je následovně.

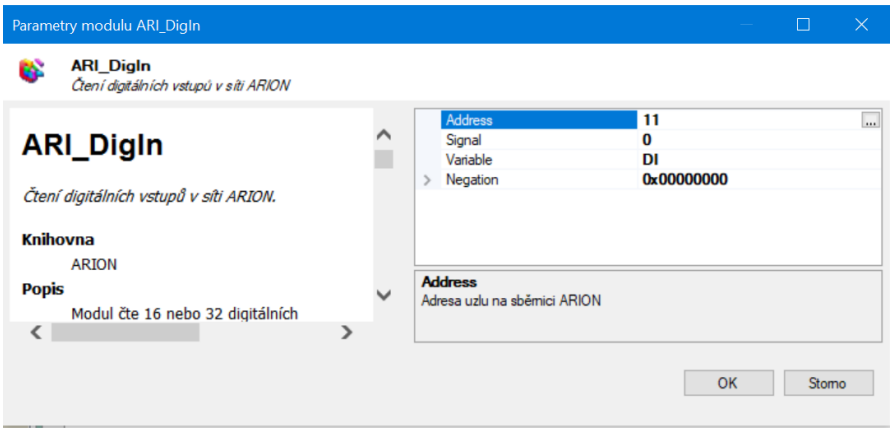




Proces pak ve výsledku bude vypadat takto.

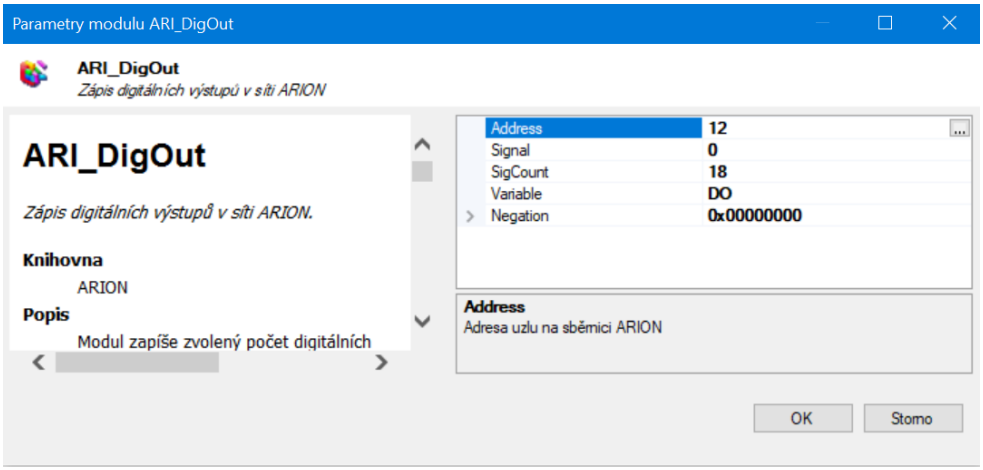
```
:1234 ARION 1, 19200, 3

ARI_DINode :1234, 11, 100, 2000, 24
ARI_DONode :1234, 12, 100, 2000, 18
```

Dále vytvoříme proces typu ST s prioritou Normal_0, který nám bude načítat vstupy a ukládat je do proměnné. Do tohoto procesu (nazvaného například Vstupy) vložíme modul ARI_DigIn a nastavíme následovně.



Do tohoto modulu je třeba vložit proměnnou do které se budou ukládat hodnoty ze vstupů. Vytvoříme proto novou proměnnou typu Long (nazvanou například DI) a použijeme jí v parametru Variable.

Obdobným způsobem vytvoříme proces, který nám bude hodnoty z proměnné (nazvané například DO) nahrávat na výstupy. Vytvoříme tedy nový proces typu ST s prioritou Normal_2 (nazvaný například Vystupy) a vložíme do něj modul ARI_DigOut. Ten bude nastaven následovně.



Nyní máme proměnné (DI a DO), ve kterých se budou ukládat hodnoty ze vstupů a výstupů. Bity z těchto proměnných nyní budeme využívat v procesu typu RS s prioritou Normal_1 (nazvaném například RS_proces), ve kterém budeme programovat řídící část programu. Veškeré následující úlohy se tedy budou programovat už pouze v tomto procesu.

Úloha 1.1

Zadání: Pomocí logických funkcí zrealizujte následující logické výrazy.

Specifikace: Využijte vstupy DI.0, DI.1, DI.2 a DI.3 (pro lepší přehlednost jsou ve výrazech označené jako A, B, C, D) k řízení výstupu DO.0, tak aby závislost výstupu na vstupech odpovídala následujícím výrazům.

Doporučené moduly: LD, ST, AND, OR, NOT

a) $A+B$

b) $\overline{A}x\overline{B}x\overline{C}$

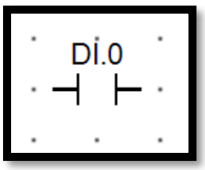
c) $AxB+\overline{A}x\overline{B}$

d) $\overline{AxB+CxD}$

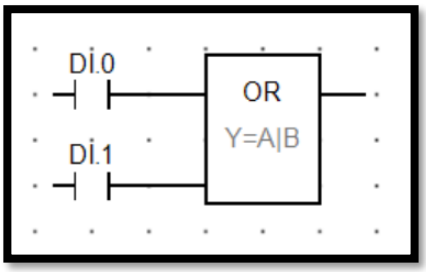
e) $(\overline{\overline{A}x\overline{B}}+C)x\overline{D}$

Řešení:

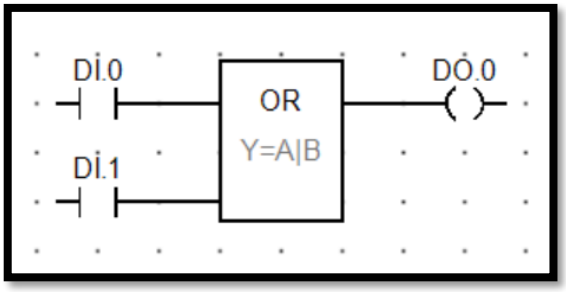
a) Pomocí modulu LD načteme vstupy DI.0 (ve výrazu označen jako A) a DI.1 (ve výrazu označen jako B).



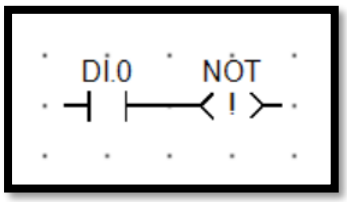
Ty spojíme s modulem OR (logický součet).



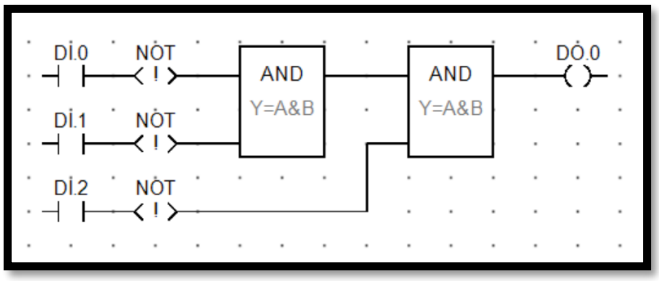
Podobně jako vstup, tak i výstup je třeba nahrát pomocí modulu. Tentokrát modulu ST. Ten připojíme na výstupní část modulu OR.



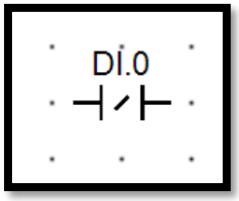
b) Horní podtržítko nad písmeny či logickými operátory značí negaci. Ta se provádí modulem NOT.



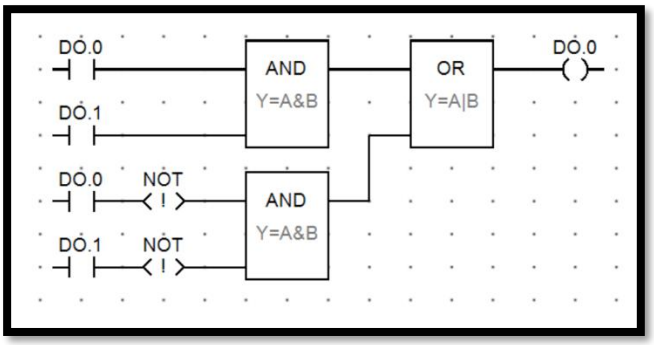
Jelikož na modul AND lze připojit jen dva vstupy, musíme v tomto případě zkombinovat více modulů za sebe.



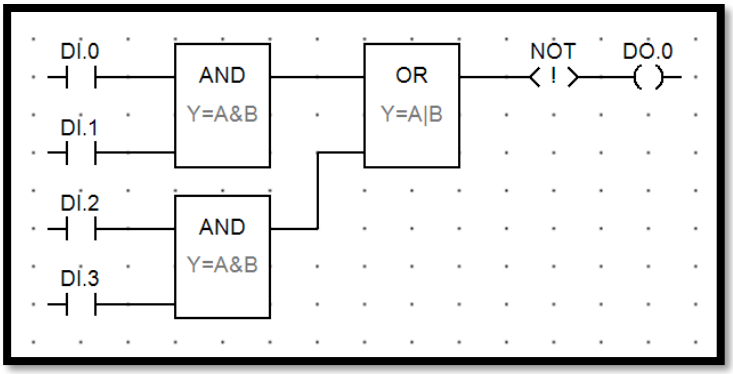
Pozn.: Negaci v tomto případě lze provést i s modulem LDN, který načítá vstup a zároveň ho i neguje (spojuje moduly LD a NOT dohromady).



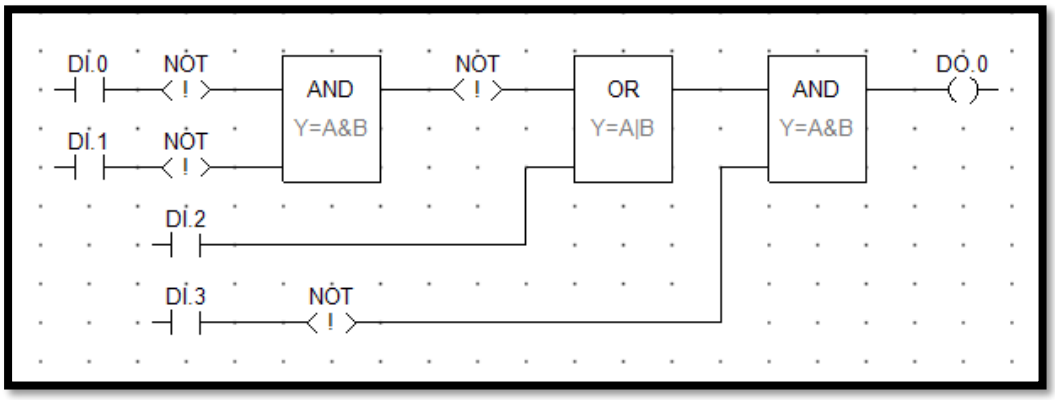
c) Stejně jako v artimetických tak i v logických operacích má násobení přednost před sčítaním, a proto nejdříve provedeme log. součiny (AND) a teprve potom log. součet (OR).



d) V tomto případě je negovaný celý výraz. Proto stačí přidat funkci OR na konec schématu.

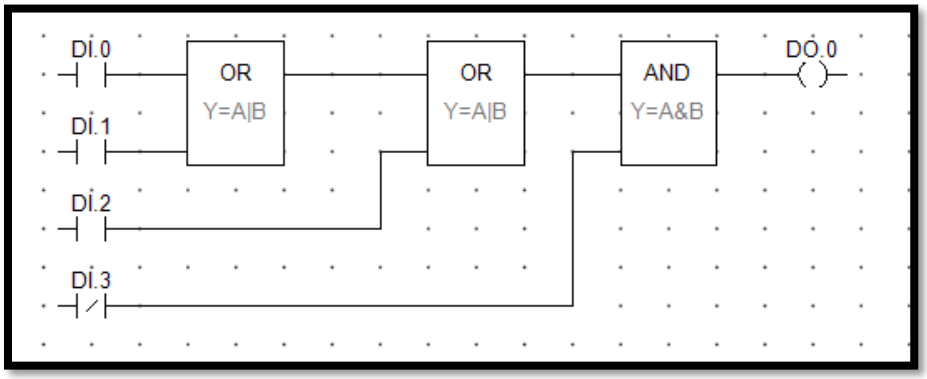


e) Zde můžete opět postupovat jako v předchozích případech. Schéma pak bude vypadat takto.



Jste-li seznámeni s pravidly pro úpravu log. rovnic, lze výraz upravit do tvaru:

$(\overline{\overline{A}x\overline{B}}+C)x\overline{D} = (A+B+C)x\overline{D}$



Úloha 1.2

Zadání: Sestavte program, při kterém se na výstupu objeví logická jednička tehdy, když je logická jednička alespoň na dvou vstupech.

Specifikace: Využijte vstupy DI.0, DI.1, DI.2 a DI.3 k řízení výstupu DO.0 tak, aby na výstupu byla log. 1 právě tehdy, když je na dvou a více vstupech log. 1.

Doporučené moduly: LD, ST, AND, OR, NOT

Řešení:

V případech kdy není řešení na první pohled zcela zřejmé, je dopručeno začít pravdivostní tabulkou z které vytvoříme logický výraz. Pravdivostní tabulka musí obsahovat všechny možné kombinace hodnot na vstupech a u každé kombinace musí být zapsán stav na výstupu/výstupech. Podle zadání musí být alespoň na dvou vstupech log. 1, aby mohla být log. 1 na výstupu. Tabulka tedy bude vypadat takto.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Nyní si vybereme ty řádky, kde je na výstupu log. 1.

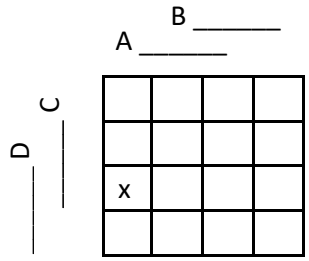
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Nyní by jsme již mohli vytvořit logický výraz, tak že vezmeme vybrané řádky a stavy vstupů vložíme do funkce AND. Vezmeme například první řádek.

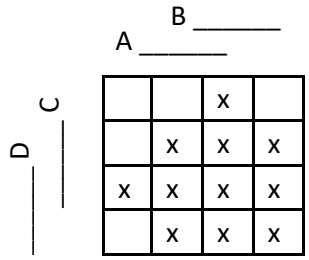
A	B	C	D	Y
0	0	1	1	1

Výraz pro tento řádek by vypadal takto: $\overline{A}x\overline{B}xCxD$ (jelikož jsou na vstupech A a B nuly, budou ve výrazu negovány). Toto by jsme provedli s každým řádkem a řádky spojily funkcí OR ($\overline{A}x\overline{B}xCxD + \overline{A}xBx\overline{C}xD +\ldots$). Takto vzniklý výraz by sice fungoval, ale byl by neskutečně dlouhý. Proto je dobré výraz upravit. To je možné například pomocí tzv. Karnaughových map.

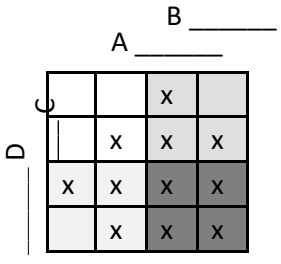
Vytvoříme si tabulku 4x4 do které budeme z pravdivostní tabulky vypisovat kombinace vstupů s log. 1 na výstupu. Opět si to ukážeme na výrazu $\overline{A}x\overline{B}xCxD$. Negované A odpovídá pravému a levému krajnímu sloupci, negovné B pak levému krajnímu a druhému sloupci zleva. C se pak nachází ve dvou prostředních řádcích a D ve spodních řádcích. Všechny tyto prvky budou mít průnik v jednom políčku.



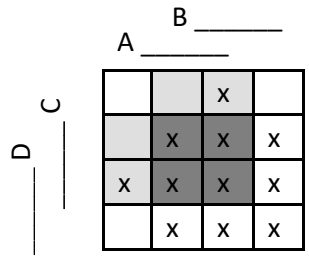
To provedeme s každým řádkem pravdivostní tabulky kde je na výstupu log. 1. Tabulka bude ve výsledku vypadat takto.



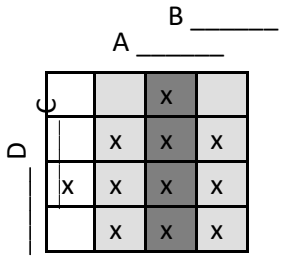
Nyní budeme označená políčka spojovat do větších čtverců nebo obdelníků, které lze popsat jednoduším log. výrazem.



BxD

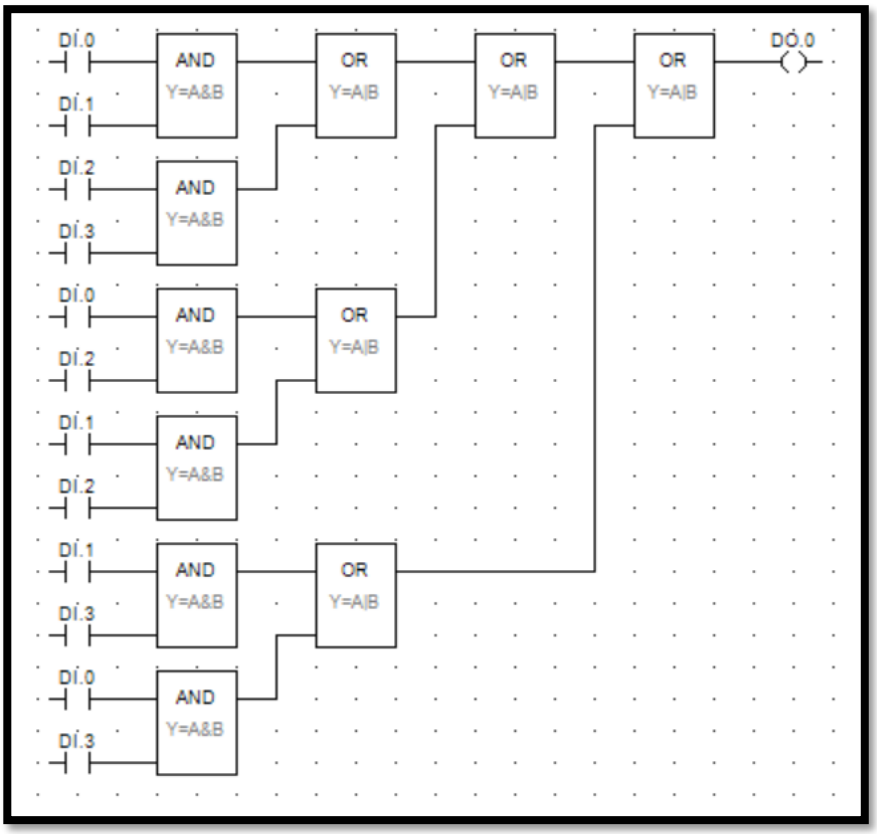


AxC



AxB

Takto pokračujte dokud nepoužijete každé označené pole alespoň jednou. Hledejte vždy co největší možné seskupení. Jednotlivé výrazy pak spojte funkcí OR. Finální výraz pak bude: AxB + CxD + AxC + BxC + BxD + AxD.



Úloha 1.3

Zadání: Sestavte program pro ovládání světla ze tří míst, kdy přepínače (nebo tlačítka) budou rozsvicovat a zhasínat nezávisle na sobě.

Specifikace: Využijte vstupy DI.0, DI.1 a DI.2 pro řízení výstupu DO.0 tak, aby po změně hodnoty na jakémkoliv z těchto vstupů došlo ke změně hodnoty na výstupu (v případě, že se rozhodnete používat tlačítka, tak jeden impulz způsobí na výstupu log.1 a druhý impulz log.0).

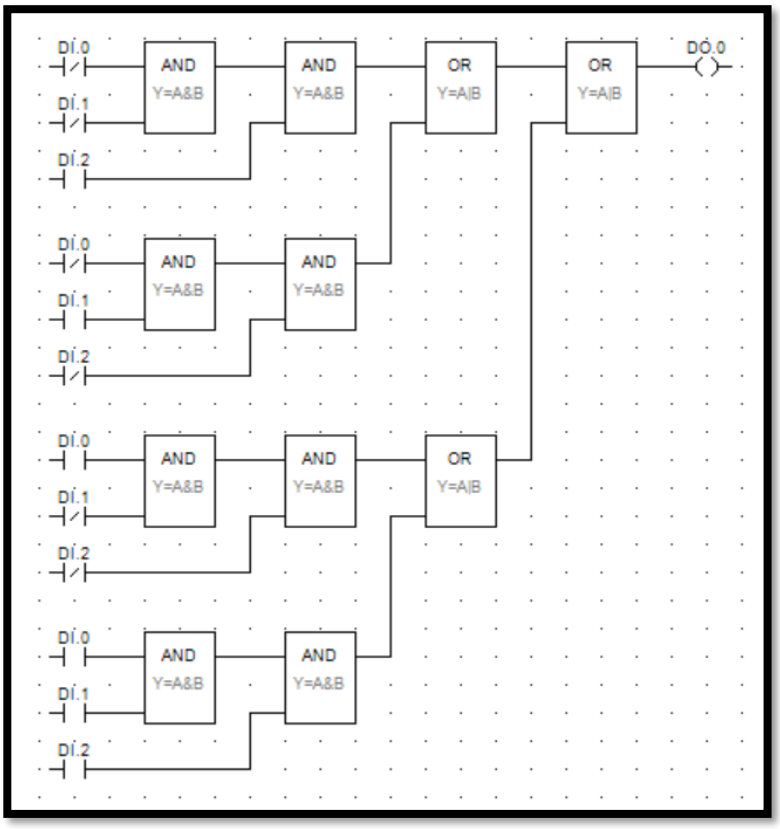
Doporučené moduly: LD, ST, AND, OR, XOR

Řešení:

Princip spočívá v tom, že první přepnutí libovolného přepínače světlo rozsvítí a druhé přepnutí přepínače (ať už stejného nebo odlišného) světlo vypne. Jedná se tedy o to že log.1 musí být na lichém počtu vstupů.

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Nyní můžeme postupovat jako v předešlé úloze. Logický výraz pak bude: $\overline{A}xB\overline{B}xC + \overline{A}xBx\overline{C} + Ax\overline{B}x\overline{C} + AxBxC$ (nejde již více upravit). Schéma bude vypadat takto.

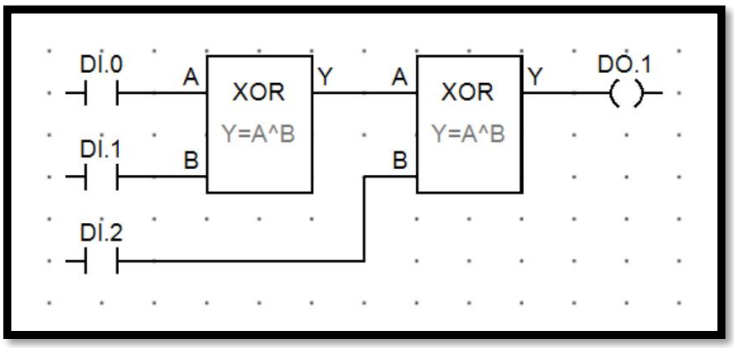


Druhou možností je použít funkci XOR (obsažený v modulu XOR). Tato funkce má na výstupu log.1, právě tehdy když je na vstupech lichý počet log. 1 (obsahuje výraz $Ax\overline{B}$

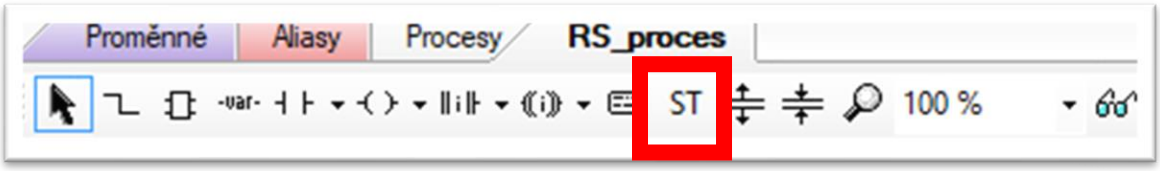
+ $\bar{A}$xB) což je přesně to co pro tuto úlohu potřebujeme. Jedná se tedy o podstatné zjednodušení schémata výše.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

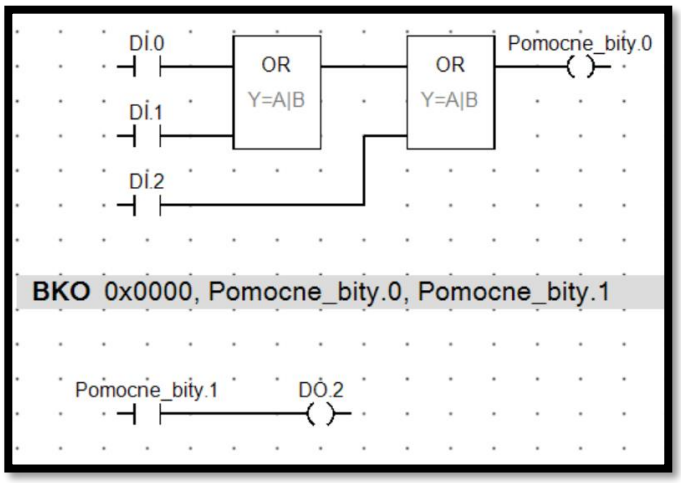
Schéma pak bude vypadat takto.



Pokud místo přepínačů použijeme tlačítka budeme potřebovat nějaký prvek, který udrží hodnotu na výstupu i potom, co impulz vyvolaný tlačítkem zmizí. K tomu můžeme využít bistabilní klopný obvod, který po krátkém impulzu na vstupu, trvale změní hodnotu na výstupu. Tento klopný obvod je zastoupen modulem BKO. Ten však není klasickou součástí procesu RS (reléových schémat) a tak ho vložíme pomocí ikony ST v horní liště.



Jelikož je tento modul textový, nejde zapojit do reléového schématu, a tak musíme založit novou proměnnou (nazveme jí například Pomocne_bity) jejíž bity využijeme k přenesení hodnot mezi moduly.



Úloha 1.4

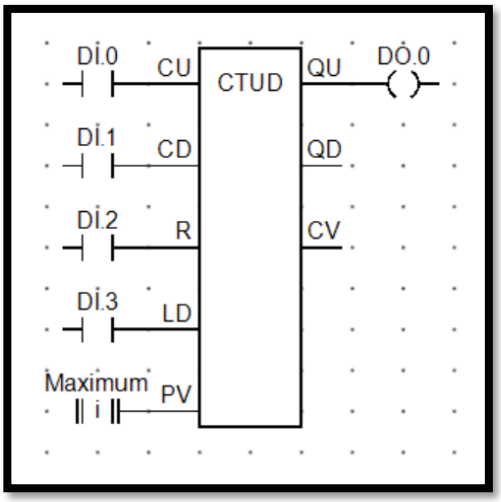
Zadání: Sestavte program, který bude počítat auta na parkovišti. Počet aut se bude určovat podle závory, která bude přičítat auta, která přijedou a odčítat auta, která odjedou. V momentě, kdy bude parkoviště plné, se rozsvítí nápis PLNÉ PARKOVIŠTĚ a závora přestane pouštět další auta.

Specifikace: Použijte vstup DI.0, který (po krátkém impulzu) bude na čítači přičítat 1 (auto přijelo na parkoviště). Dále použijte vstup DI.1, který bude na čítači odčítat 1 (auto odjelo z parkoviště). Použijte výstup DO.0, na kterém se objeví log. 1, když čítač dosáhne hodnoty 5 (parkoviště je plné). Vymyslete způsob, jak zabránit čítači dále přičítat, když dosáhne hodnoty 5 (zabránit dalším autům přijíždět, když je parkoviště plné).

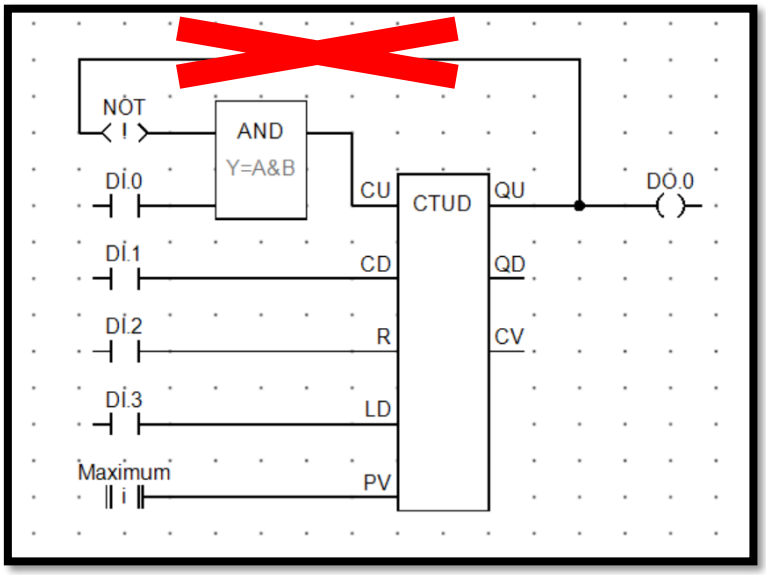
Doporučené moduly: LD, ST, LDi, CTUD, AND, NOT

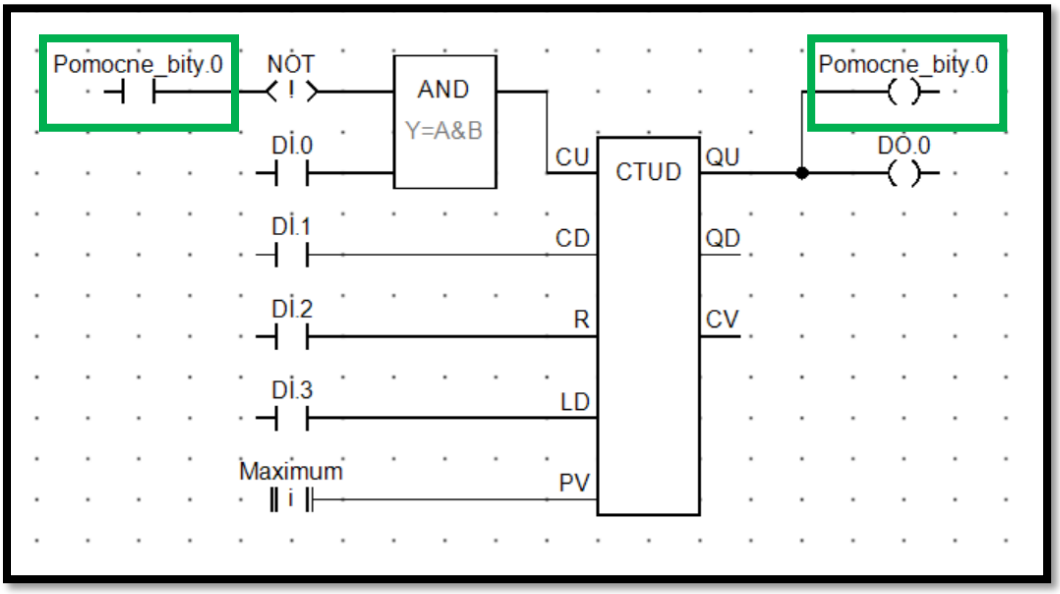
Řešení:

Vložíme modul CTUD, který obsahuje čítač a podle nápovědy DetStudia připojíme vstupy a výstupy (pro lepší zacházení s čítačem můžeme použít i vstup R). Jelikož na vstupu PV má být hodnoty typu integer, založíme proto novou proměnnou (nazveme jí třeba Maximum) a inicializujeme jí hodnotu 5. Tuto proměnou připojíme na čítač modulem LDi. Jelikož DetStudio odmítne program zkompilovat, dokud nebudou všechny vstupy čítače zapojené, připojíme k nim nevyužité vstupy, nebo pomocné bity.



Nyní musíme ještě zastavit přičítání, když čítač dosáhne hodnoty 5. To provedeme přidáním funkcí AND a NOT, tak aby signál z výstupu čítače QU řídil vstup CU. K tomu musíme použít pomocné bity, neboť kdybychom spojili QU a CU pouhou čarou, vznikla by smyčka, kterou DetStudio odmítne zkompilovat. Založíme tedy novou proměnnou (nazvanou například Pomocne_bity), jejíž bit použijeme k přenosu hodnoty z výstupu čítače na jeho vstup.


NOT
AND
Y=A&B
DI.0
CU
CTUD
QU
DO.0
DI.1
CD
QD
DI.2
R
CV
DI.3
LD
Maximum
PV


Pomocne_bity.0
NOT
AND
Y=A&B
Pomocne_bity.0
DI.0
CU
CTUD
QU
DO.0
DI.1
CD
QD
DI.2
R
CV
DI.3
LD
Maximum
PV

Úloha 1.5

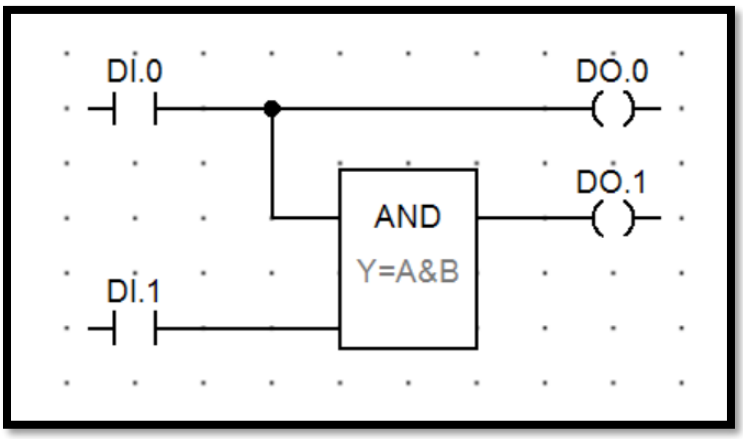
Zadání: Sestavte program pro postupné zapínání dvou motorů, tak že není možné spustit Motor 2 dříve než Motor 1. V případě že dojde k vypnutí Motoru 1, musí se zastavit i Motor 2.

Specifikace: Použijte vstupy DI.0, DI.1 k řízení výstupů DO.0, DO.1 (každý vstup ovládá jeden výstup). Na výstupu DI.1 nesmí být log.1, pokud již není i na výstupu DO.0.

Doporučené moduly: LD, ST, AND

Řešení:

Podmínkou je, že k zapnutí Motoru 2 (DO.1) musí být přepínač (DI.1) v poloze zapnuto a zároveň musí běžet Motor 1. Tuto podmínku uskutečníme funkcí AND.



Úloha 1.6

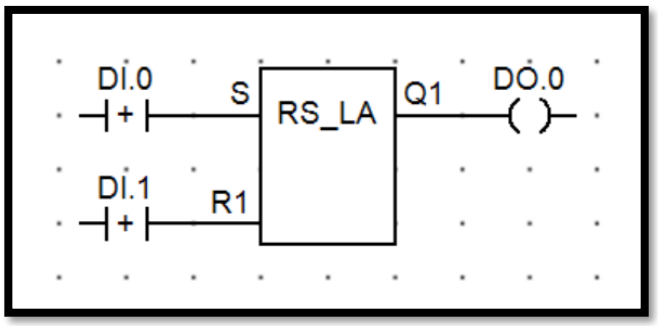
Zadání: Vytvořte program pro ovládání motoru dvěma tlačítky (START a STOP)

Specifikace: Použijte vstup DI.0 na přivádění log.1 na výstup DO.0 (další manipulace s tímto vstupem ale již nemůže přivést na výstup log. 0) a vstup DI.1 na přivádění log. 0.

Doporučené moduly: LD, ST, RS_LA

Řešení:

Nejjednodušší je použít klasický RS klopný obvod (obsažený v modulu RS_LA). Pokud na vstup S přivedeme log. 1 (stačí i krátký impulz vyvolaný stisknutím tlačítka), na výstupu Q zůstane log. 1 do doby, než přivedeme log. 1 na vstup R. Doporučuji nahradit moduly LD za moduly LD_R, které vygenerují krátký impulz, aby nedocházelo k možným kolizím.



Úloha 1.7

Zadání: Vytvořte program, který bude řídit blikání a) jedné diody b) tří diod

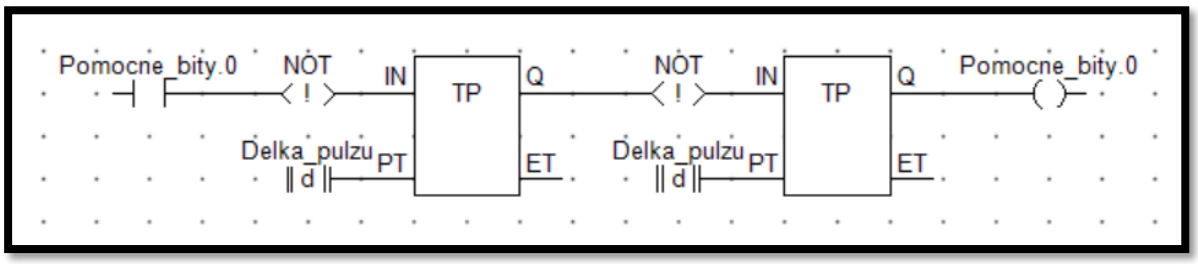
Specifikace: a) Použijte vstup DI.0 ke spuštění programu (tento vstup je spíše doporučený a není nutně potřeba), který přivede na výstup DO.0 log. 1 po dobu jedné vteřiny. Poté na výstup přivede log. 0 opět po dobu jedné vteřiny. Tento cyklus se bude sám neustále opakovat (bez vnějšího zásahu).

Doporučené moduly: LD, LD_R, LDd, ST, TP, NOT, AND, RS_LA

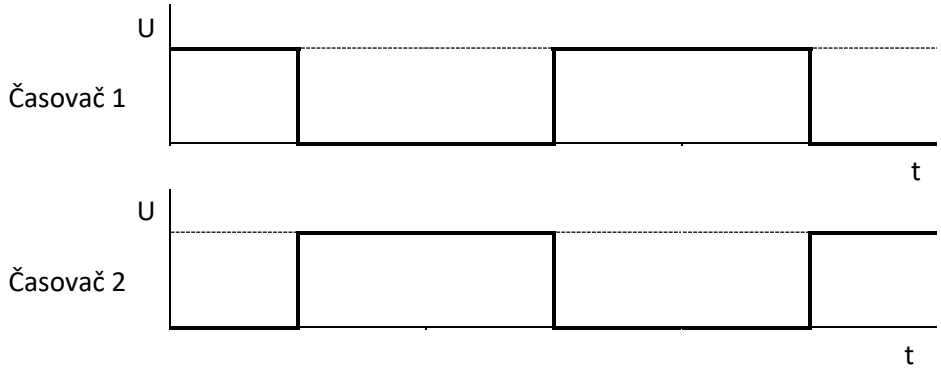
Řešení:

K této úloze budeme potřebovat časovače (konkrétně časovač obsažený v modulu TP, který generuje pulz, jehož délku lze nastavit). Vložíme tedy do schématu modul TP, na jehož vstup PT připojíme modul LDd s proměnou typu integer (nazvanou například Delka_pulzu), jenž bude mít inicializovanou hodnotu 1000 (čas v [ms]).

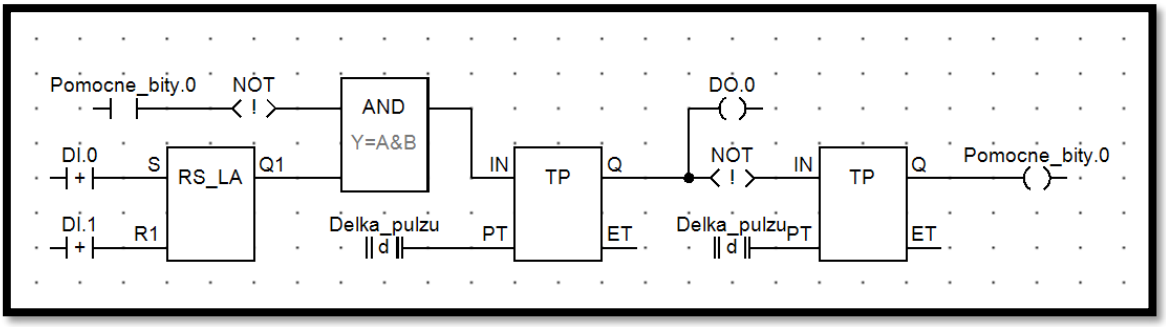
Nyní máme časovač, který rozsvítí diodu po dobu jedné sekundy. Potřebujeme ale také druhý časovač, který po druhé sekundě (kdy je dioda zhaslá) znovu spustí první časovač. Jelikož vstup časovače reaguje na vzestupnou hranu, musíme vstup druhého časovače znegovat, aby se spustil v momentě, kdy pulz z prvního časovače doběhne do konce. Konec druhého časovače pak musíme opět připojit na znegovaný vstup prvního časovače. To provedeme bitem z pomocné proměnné (nazvané například Pomocne_bity).



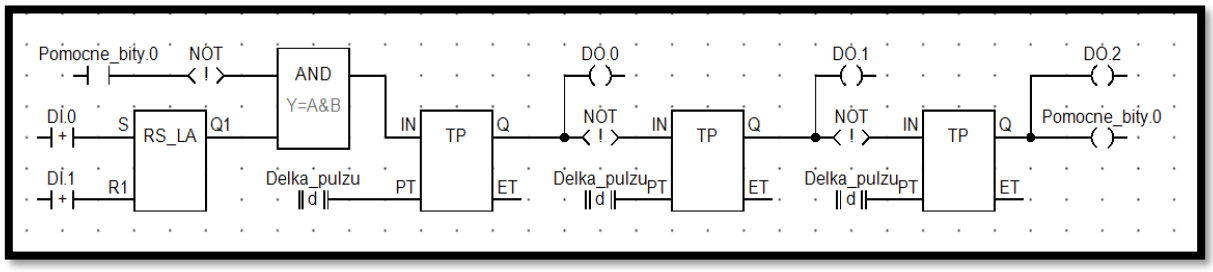
V momentě, kdy puls na jednom časovači doběhne do konce jeho sestupná hrana spustí druhý časovač a naopak. To způsobí nekonečnou smyčku, která rozbliká naši diodu.



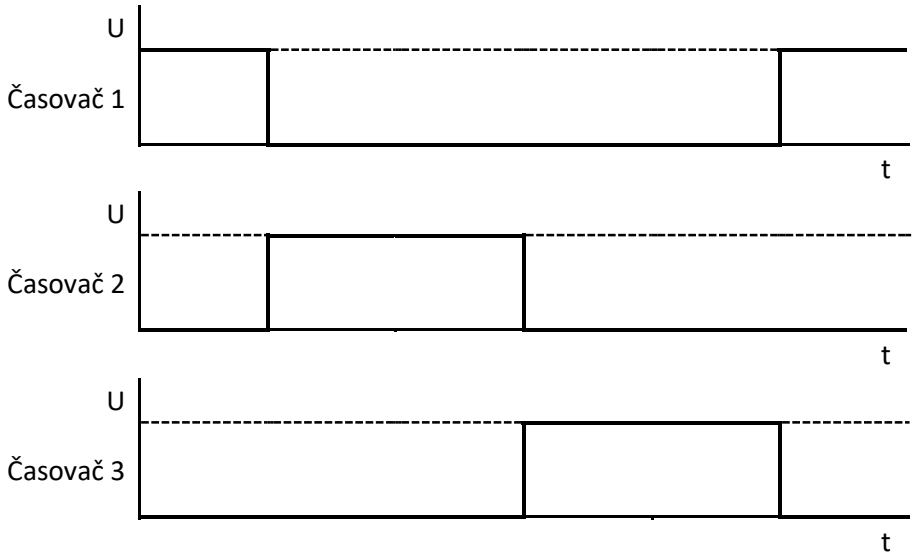
Nyní už jen potřebujeme prvotní impulz, který tento cyklu odstartuje (k tomu můžeme použít vstup DI.0) a připojit výstup jednoho z časovačů na LED diodu (DO.0). Mezi vstup DI.0 a časovač je ještě dobré vložit klopný obvod a funkci AND. Bez těchto prvků by byla na vstupu časovače log. 1 již při startu programu a časovač by nereagoval na stisk tlačítka na vstupu DI.0. Klopný obvod zároveň zamezí možným kolizím při opakovaném stisku tlačítka na vstupu DI.0, které by mohlo narušit cyklus. Konečné schéma pak bude vypadat takto.



V druhé části úlohy (rozblikání tří diod) budeme postupovat podobně. Rozdíl bude v tom, že do tohoto cyklu přidáme ještě třetí časovač a každý časovač připojíme na jinou LED diodu.



V momentě, kdy vyprchá impulz na jednom z časovačů, sestupná hrana impulzu spustí další časovač v pořadí. Tímto způsobem můžeme rozblikat libovolný počet diod na libovolných frekvencí.


U
Časovač 1
t
U
Časovač 2
t
U
Časovač 3
t

Úloha 1.8

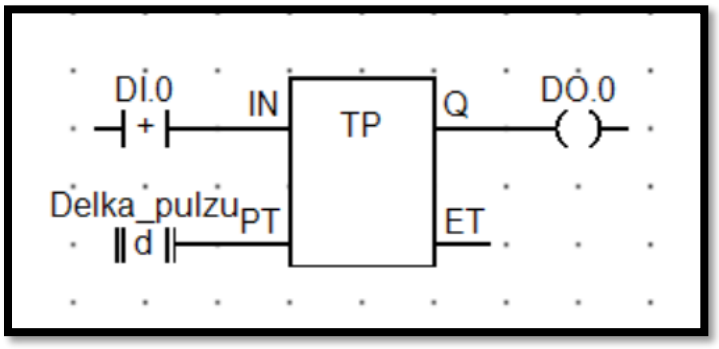
Zadání: Vytvořte program, který po stisku tlačítka START spustí mikrovlnku po určitou dobu.

Specifikace: Použijte vstup DI.0 ke spuštění časovače, který přivede log. 1 na výstup DO.0 po dobu 5 sekund.

Doporučené moduly: LD_R, LDd, ST, TP

Řešení:

Tuto úlohu lze vyřešit modulem TP. Ten generuje pulz, jehož délku trvání lze libovolně nastavit. Vložíme tedy modul TP a na jeho vstup PT připojíme modul LDd. Do modulu LDd vložíme proměnnou typu integer (nazvanou například Delka_pulzu), které inicializujeme hodnotu 5000 (čas v [ms]). Na vstup IN pak standartně vložíme modul LD nebo LD_R s bitem DI.0 a na výstup Q připojíme modul ST s bitem DO.0.

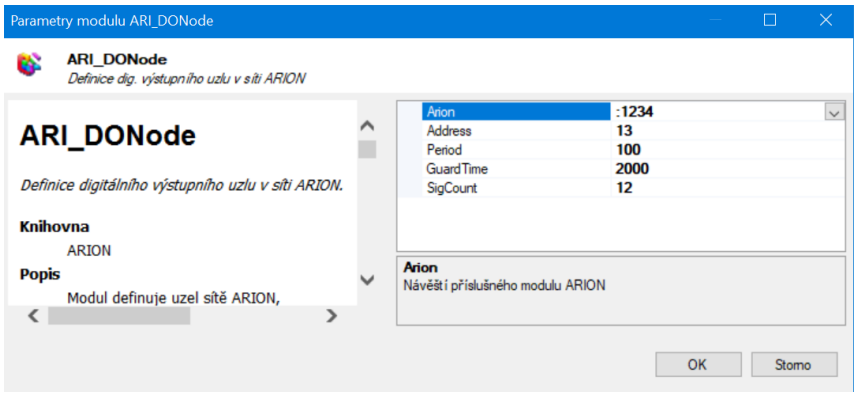


2. Práce s modulem DM-RDO12 (spínací relé 250 V/6 A)

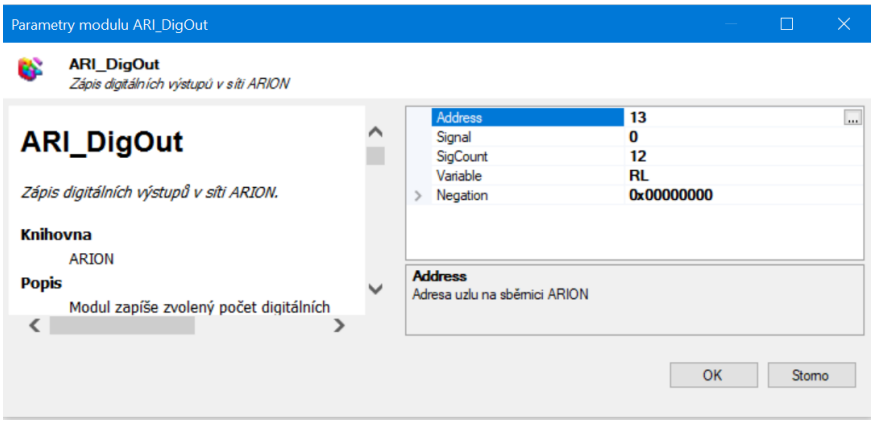
Tyto úlohy pracují s modulem vzdálených vstupů/výstupů DM-RDO12, které nahrazuje klasické digitální výstupy spínacími relé. Tyto relé se sice dají snadno nahradit, v některých případech je však jejich použití praktičtější a levnější. Z těchto důvodů si s nimi ukážeme alespoň pár úloh.

Příprava

Postup bude téměř stejný jako v předchozím případě. Rozdíl bude v procesu ProcINIT, kde v modulu ARI_DONode změníme parametry Address a SigCount.



Dále v procesu Vystupy, změníme modulu ARI_DigOut parametry Address, SigCount a Variable.



Hodnoty výstupů budeme ukládat v proměnné RL (kterou nahradíme proměnou DO).

Úloha 2.1

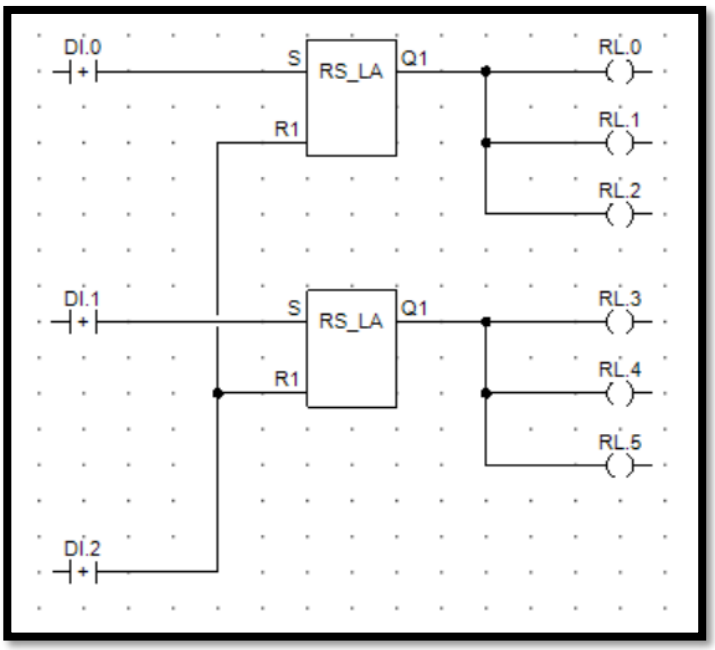
Zadání: Sestavte program na ovládání třífázového motoru. Tlačítko DOPRAVA roztočí motor na jednu stranu a Tlačítko DOLEVA na druhou. Tlačítko STOP motor zastaví. Pokud již běží motor na jednu stranu, není možné ho pustit na druhou (vzájemná blokace).

Specifikace: Použijte vstup DI.0 k přivedení log.1 na výstupy RL.0, RL.1 a RL.2 (třífázový motor), dále použijte vstup DI.1 k přivedení log.1 na výstupy RL.3, RL.4 a RL.5. Vstup DI.2 použijte na přivedení log. 0 na všechny výstupy. Pokud je log. 1 na výstupech RL.0, RL.1 a RL.2, nemůže již být i na RL.3, RL.4 a RL.5 (a opačně).

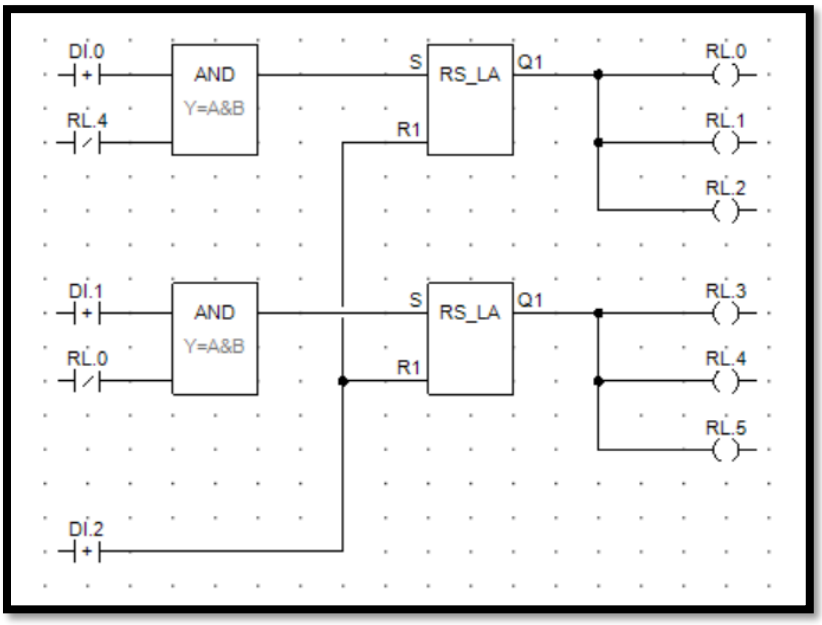
Doporučené moduly: LD_R, ST, RS_LA, LDN

Řešení:

Použijeme modul RS_LA (klopný obvod) k ovládání motorů pomocí tlačítek.



Nyní ještě musím přidat vzájemné blokování, aby nebylo možné pustit motor do obou směrů současně. To provedeme tak, že vezmeme jeden z výstupů z každé "větve", znegujeme ho a použijeme ve funkci AND v protější "větvi". V případě že motor běží v jednom směru se přeruší signál, který by se pokoušel spustit motor v druhém směru.



Úloha 2.2

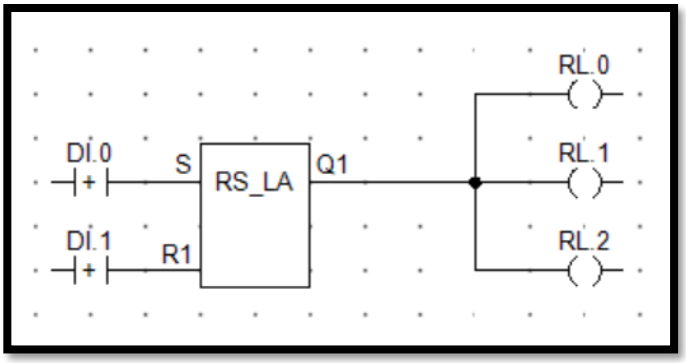
Zadání: Vytvořte program na ovládání trojfázového motoru. Tlačítko START spustí motor v zapojení do hvězdy. Po třech sekundách se automaticky přepne do zapojení trojúhelník. Tlačítko STOP motor zastaví.

Specifikace: Použijte vstup DI.0 k přivedení log.1 na výstupy RL.0, RL.1 a RL.2 (třífázový motor). Po třech sekundách se na těchto výstupech objeví log. 0 a na výstupech RL.3, RL.4 a RL.5 se objeví log.1. Vstup DI.1 musí kdykoliv přivést log. 0 na všechny výstupy.

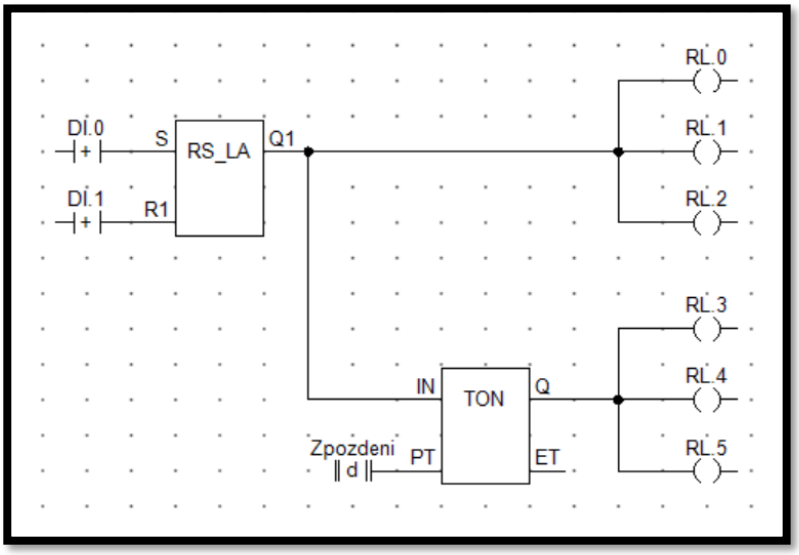
Doporučené moduly: LD_R, LDd, ST, RS_LA, AND, NOT, TON

Řešení:

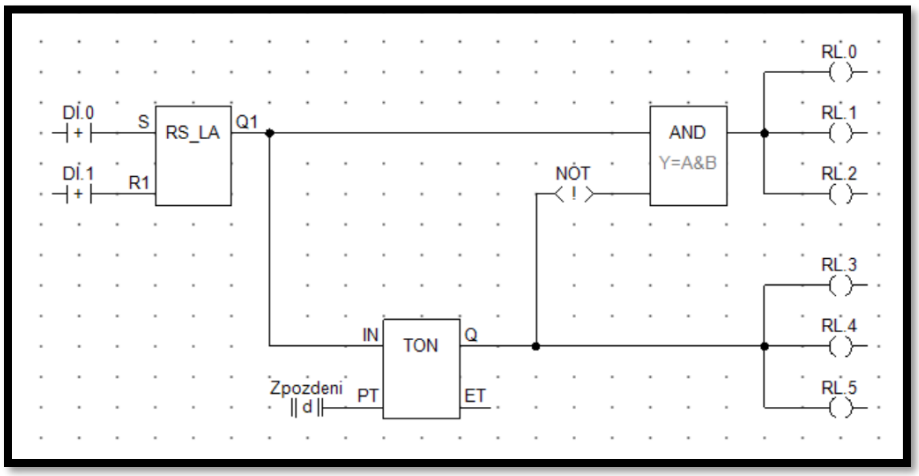
Nejdříve si uděláme zapojení do hvězdy.



Nyní vytvoříme časovač (obsažený v modulu TON), který po třech sekundách přepne motor do trojúhelníku. Časovač TON zpožďuje náběžnou hranu. Po startu motoru tedy začne časovač odpočítávat tři vteřiny. Po této době vyšle signál, který aktivuje zapojení do trojúhelníku. Čas nastavíme tak, že založíme novou proměnnou typu integer a inicializujeme jí hodnotu 3000 (čas v [ms]). Tuto proměnnou pak připojíme na vstup PT pomocí modulu LDd.



V tom samém momentě se ale musí také odpojit zapojení do hvězdy, a proto bude finální schéma vypadat takto.



3. Práce s analogovými vstupy a výstupy

V současné době si již často nevystačíme pouze s digitálními hodnotami, neboť by některé komplexnější úlohy bylo příliš obtížné s řešit pouze s těmito hodnotami. Proto je dobré mít i povědomí o práci s analogovými hodnotami s kterými jsou úkony jako aritmetické operace či regulace mnohem snazší.

Příprava

Opět budeme postupovat jako v úlohách “Práce s digitálními vstupy a výstupy“. Zde ale navíc přidáme do procesu ProcINIT moduly ARI_AINode a ARI_AONode. Oba tyto moduly budou mít parametry Address = 14 a SigCount = 8. Proces pak bude vypadat takto.

```
:1234 ARION 1, 19200, 3

ARI_DINode :1234, 11, 1000, 2000, 24
ARI_DONode :1234, 12, 1000, 2000, 18
ARI_AINode :1234, 14, 1000, 2000, 8, 0x000C
ARI_AONode :1234, 14, 1000, 2000, 8, 0x000C
```

Dále v procesu Vstupy vytvoříme 4 moduly ARI_AnIn a do každého z nich vložíme proměnou typu float (nazvané například AI_0, …, AI_3). Ostatní parametry nastavte tak, aby proces vypadal takto.

```
ARI_DigIn 11, 0, DI, 0x00000000
ARI_AnIn 14, 0, 1, AI_0, NONE[0,0], 10.000, 0.000, 10.000, 0.000, 10.000
ARI_AnIn 14, 1, 1, AI_1, NONE[0,0], 10.000, 0.000, 10.000, 0.000, 10.000
ARI_AnIn 14, 2, 1, AI_2, NONE[0,0], 10.000, 0.000, 10.000, 0.000, 10.000
ARI_AnIn 14, 3, 1, AI_3, NONE[0,0], 20.000, 0.000, 20.000, 0.000, 20.000
```

Podobně budeme postupovat i v procesu Vystupy do kterého vložíme 7 modulů ARI_AnOut. Zbylé parametry opět nastavte podle obrázku níže.

```
ARI_DigOut 12, 0, 18, DO, 0x00000000
ARI_AnOut 14, 0, 1, AO_0, NONE[0,0], 10.000, 0.000, 10.000, 0.000, 10.000
ARI_AnOut 14, 1, 1, AO_1, NONE[0,0], 10.000, 0.000, 10.000, 0.000, 10.000
ARI_AnOut 14, 2, 1, AO_2, NONE[0,0], 10.000, 0.000, 10.000, 0.000, 10.000
ARI_AnOut 14, 3, 1, AO_3, NONE[0,0], 5.000, 0.000, 5.000, 0.000, 5.000
ARI_AnOut 14, 4, 1, AO_4, NONE[0,0], 5.000, 0.000, 5.000, 0.000, 5.000
ARI_AnOut 14, 5, 1, AO_5, NONE[0,0], 5.000, 0.000, 5.000, 0.000, 5.000
ARI_AnOut 14, 6, 1, AO_6, NONE[0,0], 10.000, 0.000, 10.000, 0.000, 10.000
```

Úloha 3.1

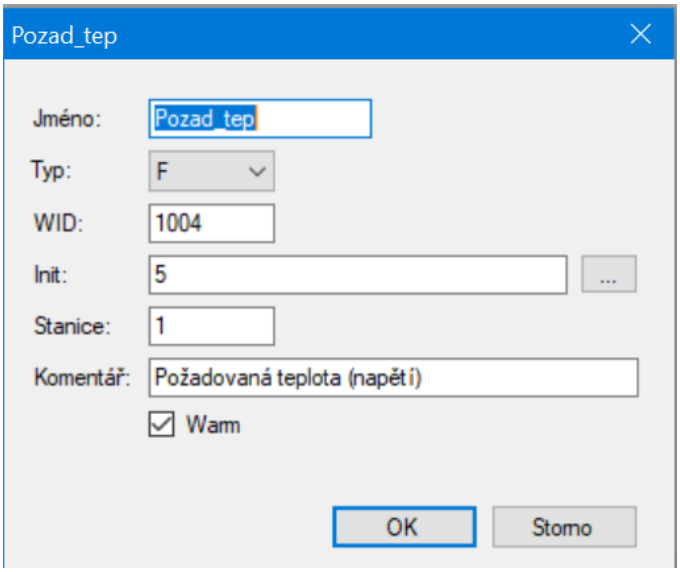
Zadání: Vytvořte program pro termostat, který a) po dosažení určité teploty vypne vytápění a po snížení teploty pod tento bod opět zapne b) upravte předchozí program, tak aby se vytápění zapínalo při nižší teplotě, než při které se vypíná.

Specifikace: Použijte vstup AI.0 k řízení výstupu DO.0 tak, aby se při překročení 5 V na vstupu se na výstupu DO.0 objevila log. 0 a po snížení napětí pod 5 V se na výstupu objevila log. 1. Poté tento program upravte tak, aby se na výstupu objevovala log. 1 při snížení napětí pod 3 V (log. 0 při 5 V zůstává).

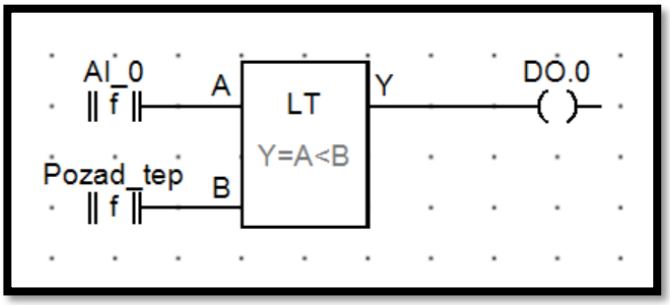
Doporučené moduly: LDf, ST, LT, GE, RS_LA

Řešení:

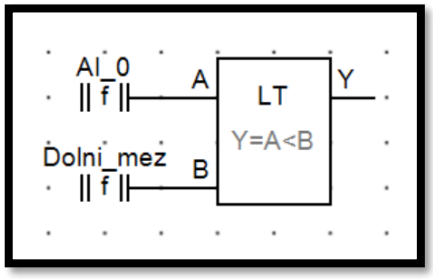
Použijeme modul LT k porovnávaní, zdali měřené napětí (hodnota na vstupu AI.0) nekleslo pod požadované (5 V). Na vstup A přivedeme měřené napětí (ukládané v proměnné AI_0) pomocí modulu LDf (který načte proměnou typu float). Na vstup B přivedeme proměnou typu float, ve které bude uložené požadované napětí (tato proměnná bude mít inicializovanou hodnotu 5).



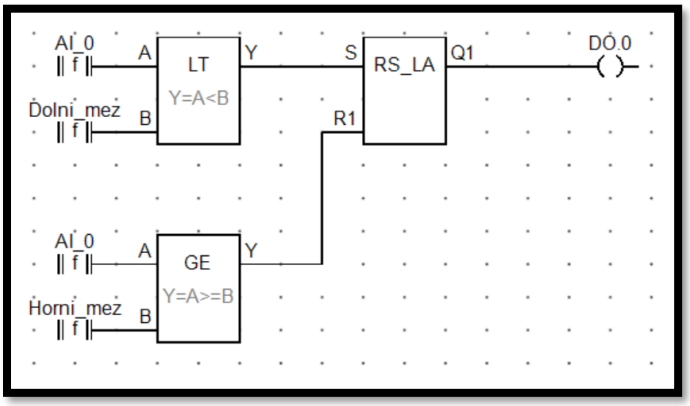
Pokud bude nyní měřené napětí menší, než požadované modul LT vyšle log. 1. Nyní už jen stačí přidat modul ST s výstupem DO.0.



V druhé části úlohy musíme pracovat s dvěma napětími. Vytvoříme proto dvě proměnné typu float (pojmenujeme je např.: Spodni_mez a Horni_mez) a inicializujeme jim hodnoty 3 a 5. Když napětí klesne pod 3 V objeví se na výstupu DO.0 log. 1 (vytápění se zapne a teplota se bude zvyšovat, dokud nedosáhne 20 °C).



Naopak, když napětí stoupne nad 5 V objeví se na výstupu log. 0 (vytápění se vypne a teplota bude opět klesat, dokud nedosáhne 15 °C). Tyto dvě větve pak připojíme na RS klopný obvod (obsažený v modulu RS_LA). Výsledné schéma pak bude vypadat takto.



Úloha 3.2

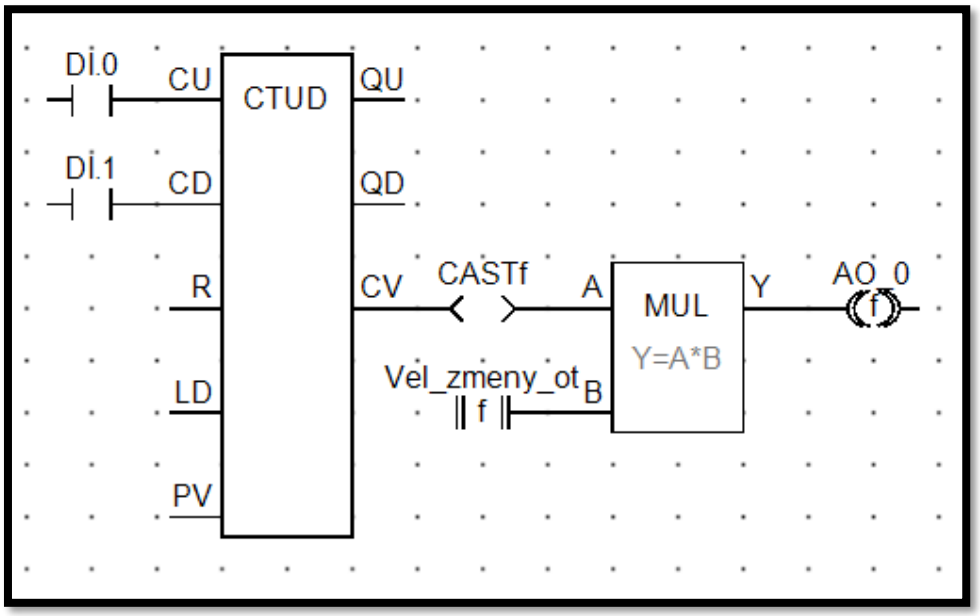
Zadání: Vytvořte program, který bude pomocí dvou tlačítek stupňovitě řídit otáčky stejnosměrného motoru od 0 % do 100 % (100 % odpovídá 5 V).

Specifikace: Použijte vstup DI.0 pro zvyšovaní napětí na výstupu AO.0 o 0,5V (10 %) a vstup DI.1 pro jeho snižování. Pokud bude výstupní napětí na 0 V (0 %) nebo 5 V (100 %), nesmí se již napětí dále snižovat, resp. zvyšovat.

Doporučené moduly: CTUD, LD, LDi, LDf, ST, STf, MUL, CASTf, NOT, AND

Řešení:

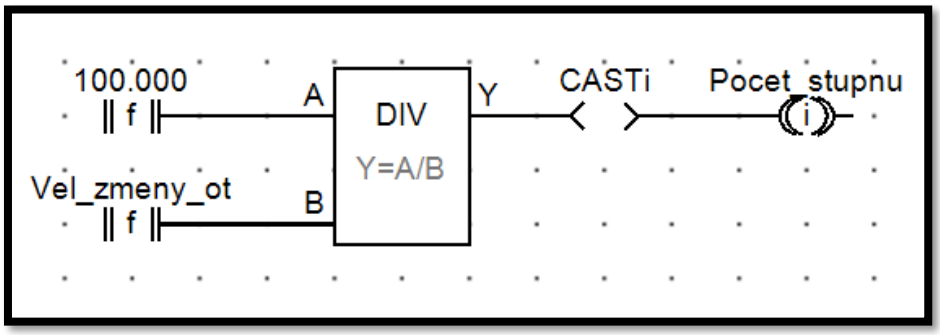
Použijeme čítač (obsažený v modulu CTUD) k počítání impulzů ze vstupů DI.0 a DI.1. Jelikož chceme, aby se po stisknutí některého z tlačítek změnily otáčky o 10 % (0,5 V) musíme tímto číslem vynásobit výstup čítače CV. To provedeme modulem MUL, na jehož vstupy připojíme výstup z čítače a modul STf do kterého vložíme proměnnou (pojmenujeme jí např. Vel_zmeny_ot), které inicializujeme hodnotu, o kterou chceme měnit otáčky po jednom stisku tlačítka. Jelikož čítač vyžaduje na výstupu typ integer a my potřebujeme typ float, použijeme modul CASTf, který nám integer přetypuje na float.



Proměnnou Vel_zmeny_ot můžeme nastavit dvěma způsoby. Buď si můžeme sami spočítat, že 10 % z 5 Voltů je 0,5 Voltu (což je ale nepraktické, pokud bychom chtěli v budoucnu hodnoty měnit), nebo můžeme hodnotu zadat v procentech. V druhém případě musíme ale upravit nastavení výstupu tak, aby přepočítával procenta na volty. Otevřeme proto proces Vystupy a v modul ARI_AnOut přenastavíme následujícím způsobem.

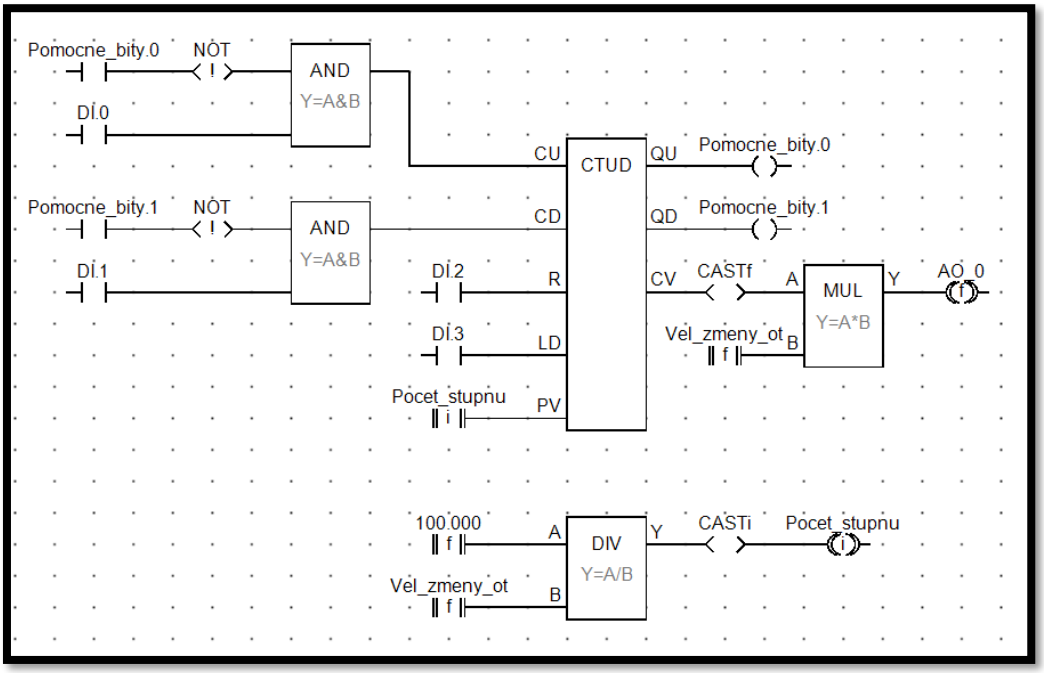
Address	14
Signal	0
SigCount	8
Value	AO_0
Conversion	NONE[0.0]
Range	5.000
ElMin	0.000
ElMax	5.000
PhysMin	0.000
PhysMax	100.000

Nyní ještě musíme zamezit tomu, aby výstupní napětí nepřekračovalo povolené meze (0 V a 5 V). Využijeme k tomu výstupy čítače QU a QD, které vyšlou log. 1, když je hodnota čítače 0, nebo hodnota nastavená na vstupu PV. Na Vstup PV proto připojíme modul LDi a vložíme do něj proměnnou (nazvanou např. Vel_stupne), jejíž hodnota bude maximální hodnotou čítače (jelikož chceme řídit otáčky od 0 % do 100 % a po stisknutí tlačítka je chceme změnit o 10 %, bude maximální hodnota 100/10 = 10). Tento výpočet můžeme také naprogramovat, abychom si ulehčili práci s případnými budoucími úpravami.



Výstupy QU a QD pak připojíme na vstupy CU a CD tak, aby při log. 1 na těchto výstupech přestal čítač dále přičítat, resp. odčítat.

Jelikož DetStudio odmítá kompilovat program, dokud nebudou všechny vstupy čítače zapojené, připojíme na ně nevyužívané bity (např. DI.2 a DI.3). Výsledné schéma pak bude vypadat takto.



Úloha 3.3

Zadání: Vytvořte program, který bude fungovat jako 3bitový D/A převodník (hodnoty z digitálních vstupů bude převádět na analogový výstup).

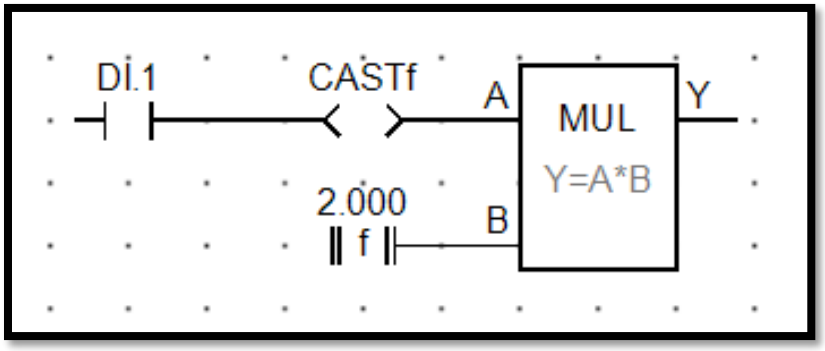
Specifikace: Použijte vstup DI.0 tak, aby při log. 1 přičetl “1“ na výstupu AO.0. Dále použijte vstup DI.1 tak, aby na výstupu přičítal “2“ a vstup DI.2 k přičítání “4“ (maximální hodnota na výstupu bude tedy “7“).

Pravdivostní tabulka			
DI.2 "4"	**DI.1** "2"	**DI.0** "1"	**AO.0**
0	**0**	**0**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**2**
0	**1**	**1**	**3**
1	**0**	**0**	**4**
1	**0**	**1**	**5**
1	**1**	**0**	**6**
1	**1**	**1**	**7**

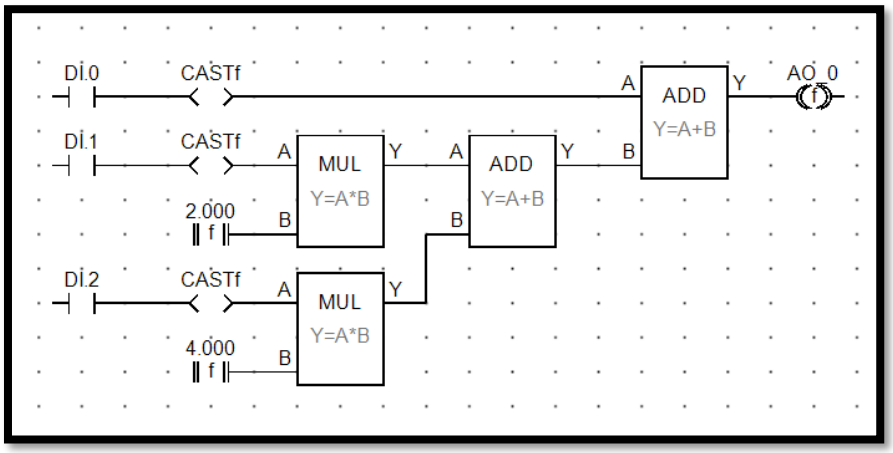
Doporučené moduly: LD, LDf, STf, ADD, MUL, CASTf

Řešení:

Jelikož digitální hodnoty obsahují pouze hodnoty “0“ a “1“, musíme je vynásobit modulem MUL abychom dostali hodnoty “0“ a “2“ nebo “0“ a “4“. Tento modul však vyžaduje, aby všechny vstupy i výstupy měly stejný datový typ, a proto použijeme modul CASTf jímž bit přetypujeme na float.



Toto provedeme se vstupy DI.1 a DI.2 (vstup DI.0 má mít hodnotu “1“ a proto není nutné ho násobit modulem MUL). Nyní už jen stačí sčítat hodnoty na vstupech modulem ADD.



Úloha 3.4

Zadání: Vytvořte program na řízení RGB LED diody tak aby:

a) jedna barva se pozvolna rozsvěcovala a po dosažení maximální svítivosti okamžitě zhasla.

b) jedna barva se pozvolna rozsvěcovala a po dosažení maximální svítivosti pozvolna zhasínala.

c) dvě nebo tři barvy se mezi sebou pozvolna střídaly.

Specifikace: a) Vytvořte program, který bude postupně zvyšovat napětí na výstupu AO.3 od 0 V do 4 V. Po dosažení 4 V napětí okamžitě klesne na 0 V. Poté se cyklus opakuje.

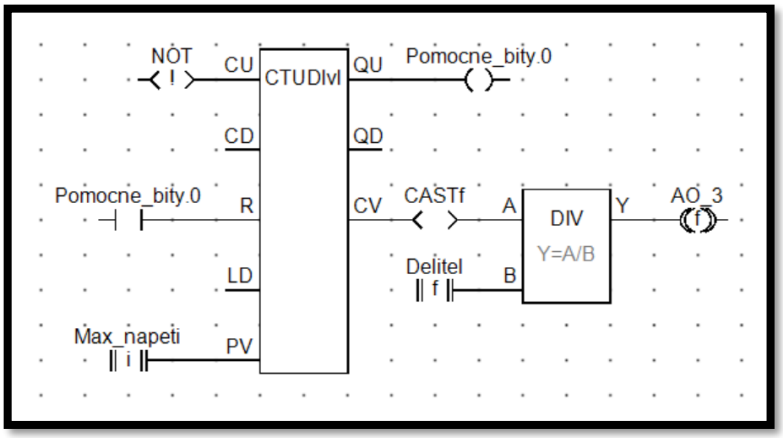
b) Upravte předchozí program tak, aby po dosažení 4V napětí postupně klesalo do hodnoty 0 V.

c) Rozšiřte předchozí program o druhou (nebo dokonce i třetí) barvu na výstupu AO.4 (nebo AO.5) tak, aby se barvy mezi sebou plynule střídaly (v momentě kdy na výstupu AO.3 bude hodnota 4 V, na výstupu AO_4 bude 0 V a naopak).

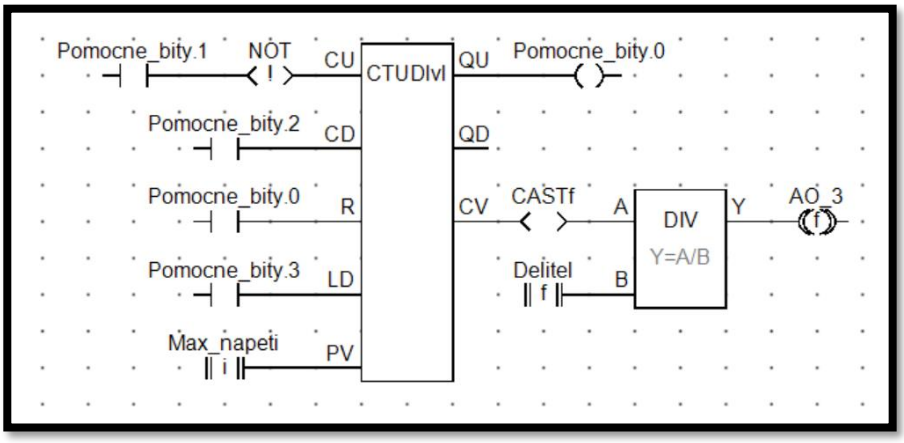
Doporučené moduly: LD, LDi, LDf, LDd, ST, STf, CTUDlvl, RS_LA, DIV, CASTf, NOT, TON

Řešení:

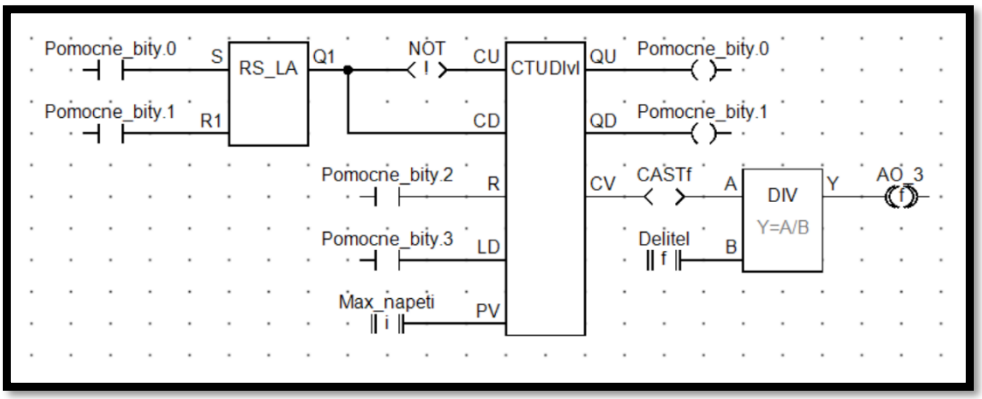
a) Použijeme hladinový čítač (obsažený v modulu CTUDlvl), který nepřičítá pouze na náběžnou hranu, ale při každém běhu procesu, kdy je na vstupu log. 1 (máme například periodu procesu nastavenou na 100 ms, během 500 ms tedy proces proběhne 5krát a čítač se dostane na hodnotu 5, tímto způsobem lze hladinový čítač použít i jako časovač). V první řadě musíme čítač nastavit tak, aby se po dosažení určité hodnoty sám resetoval (tím dosáhneme toho, že napětí na výstupu po dosažení 4 V klesne na 0 V). K tomu použijeme výstup čítače QU, na kterém se objeví log. 1, když čítač dosáhne hodnoty nastavené na vstupu PV. Na vstup PV tedy připojíme modul STi a vložíme do něj proměnnou (nazvanou např. Max_napeti), která má inicializovanou hodnotu 4. Pro plynulejší rozsvěcování je však vhodnější použít větší hodnotu jako třeba 40 a tu pak na výstupu čítače CV vydělit 10 modulem DIV. Tím dosáhneme toho, že napětí na výstupu nebude růst po celých voltech, ale po desetinách. Poté připojíme pomocným bitem výstup QU na vstup R. Neustálé přičítání zajistíme modulem NOT připojeném ke vstupu CU.



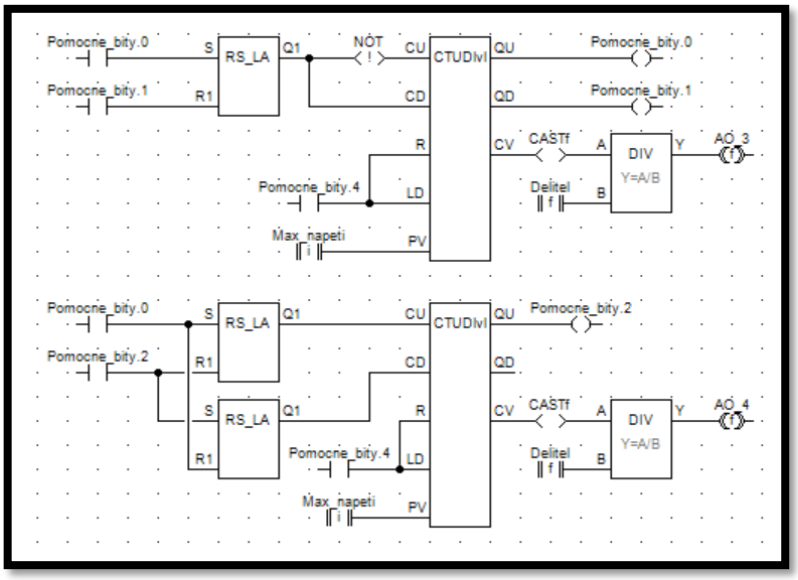
Jelikož DetStudio odmítá program zkompilovat, dokud nebudou všechny vstupy zapojené, připojíme na ně nevyužité pomocné bity.



b) V druhé části úlohy použijeme klopný obvod (v modulu RS_LA), který bude přepínat mezi vstupy CU a CD. V první řadě chceme, aby čítač začal přičítat hned po startu programu, proto na vstup CU připojíme modul NOT. Jakmile čítač dosáhne hodnoty nastavené na vstupu PV, klopný musí zastavit přičítání a spustit odčítání. Jakmile hodnota čítače dojde k nule, klopný obvod se resetuje a cyklus se opakuje. Nakonec opět připojíme nevyužité pomocné bity na zbývající vstupy čítače.



c) Ve třetí části použijeme druhý hladinový čítač, který bude řídit druhou barvu RGB LED diody. Ten však bude částečně řízen prvním čítačem. Jakmile hodnota na prvním čítači dosáhne maxima, spustí se druhý čítač. Tím zařídíme že jeden čítač bude vždy přičítat, zatímco druhý bude odčítat. U druhého čítače již nepoužijeme modul NOT, neboť by se čítač spustil ihned při startu programu. Použijeme radši druhý klopný obvod.



Popis programu: Při startu programu začne první čítač přičítat (druhý zatím zůstává nečinný). Jakmile první čítač dosáhne maximální hodnoty, spustí tím druhý čítač. Druhý čítač nyní začne přičítat, zatímco první již odečítá. Nyní se již budou čítače pravidelně střídat (jeden bude přičítat a druhý odečítat), což se projeví na střídání barev RGB LED diody.

Chceme-li přidat i třetí barvu, musíme použít časovač (v modulu TON), který nám bude zpožďovat náběžnou hranu na vstupu čítače CU a pozdrží tak start čítače.

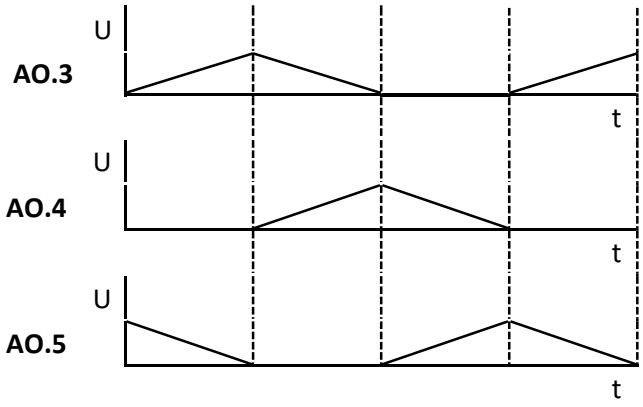
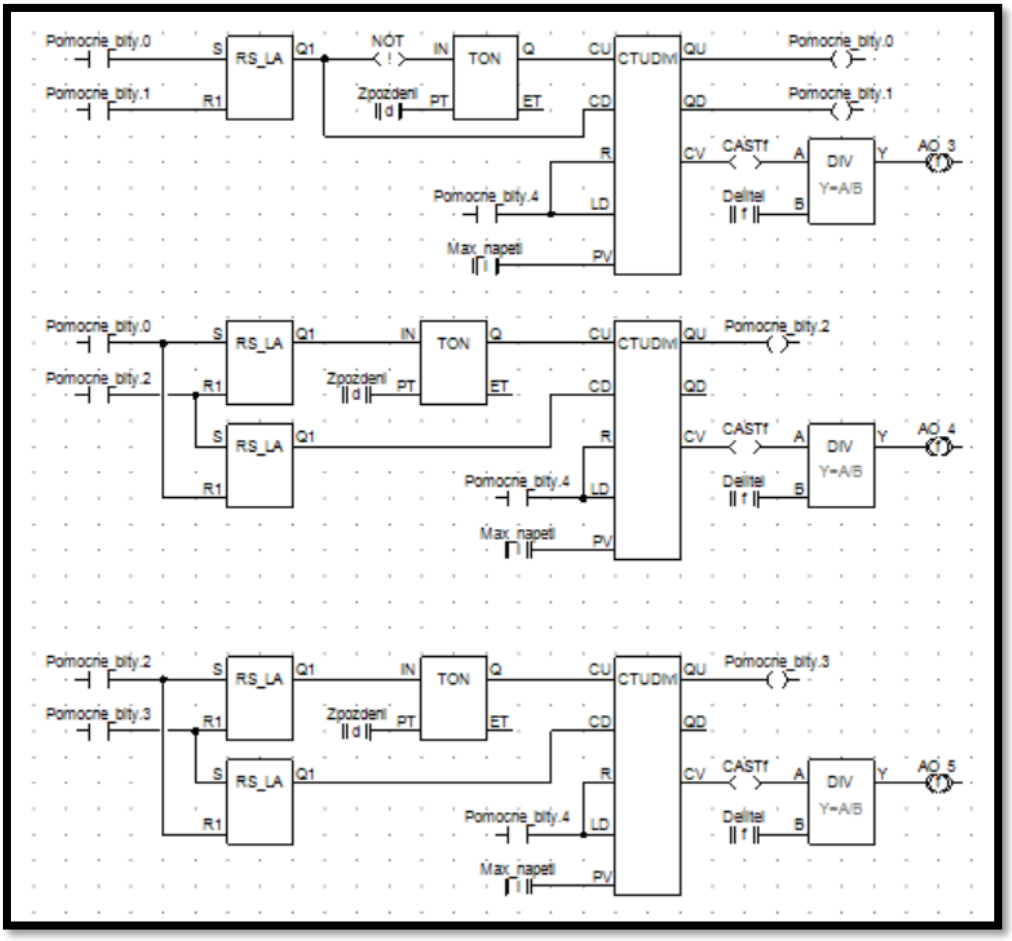


Schéma pak bude vypadat takto. Volné vstupy je opět třeba nějak zaplnit nevyužitými bity.



Úloha 3.5

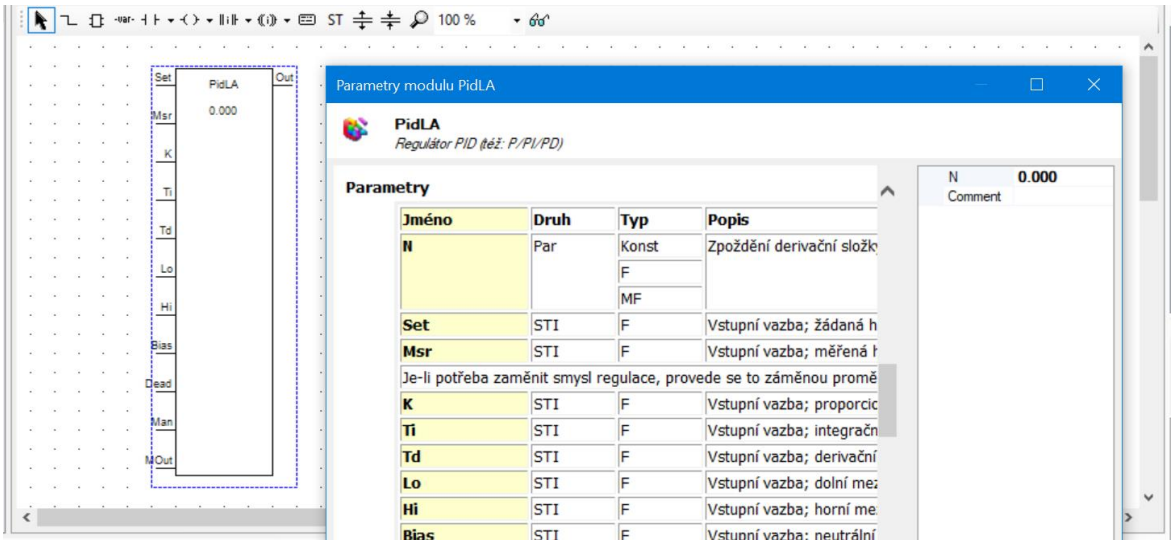
Zadání: Vytvořte program s PID regulátorem, který bude udržovat v obvodu konstantní el. proud nezávisle na el. odporu.

Specifikace: Použijte PID regulátor, který bude měřit na vstupu AI.3 el. proud a pokud se bude lišit od hodnoty 5 mA, regulátor upraví napětí na výstupu AO.6 tak, aby odchylku od této hodnoty kompenzoval. Pro kontrolu použijte potenciometr zapojený mezi vstupem AI.3 a výstupem AO.6 k měnění el. odporu v obvodu a sledujte, zdali regulátor skutečně udržuje stálý el. proud. Jako parametry regulátoru (proporcionální, integrační a derivační) použijte hodnoty 1, 5 a 0.

Doporučené moduly: LD, LDf, STf, PidLa

Řešení:

V první řadě musíme do schématu vložit PID regulátor (obsažený v modulu PidLa). Poté se prakticky stačí jen řídit nápovědou DetStudia.



Nyní tedy vytvoříme několik proměnných typu float, které pak budeme připojovat k jednotlivým vstupům a výstupům modulu PidLa.

První proměnnou nazveme třeba Poz_hodnota a použijeme inicializujeme jí hodnotu 5. Tuto proměnnou připojíme na vstup Set pomocí modulu LDf. Tato proměnná nyní obsahuje hodnotu, kterou chceme udržovat na výstupu regulátoru (v našem případě 5 mA).

Ke vstupu Msr připojíme proměnnou AI_3, což je hodnota naměřená na vstupu regulátoru.

Nyní na vstupech K, Ti a Td nastavíme parametry regulátoru. vytvoříme proto tři proměnné (nazvané například P_par, I_par a D_par) a inicializujeme jim hodnoty 1, 5 a 0. Tímto jsme nastavili proporcionální, integrační a derivační složku regulátoru.

Vstupy Hi a Lo slouží jako meze, ve kterých se může pohybovat hodnota na výstupu Out (akční zásah). Jelikož kvůli technickým omezením nemůžeme na výstupu přesáhnout hodnotu 10 V, vytvoříme proměnnou (nazvanou například Horni_mez) a inicializujeme jí hodnotu 10. Na dolní mez nemáme zvláštní požadavky, a proto vytvoříme proměnnou (nazvanou například Dolni_mez) a nechame ji hodnotu 0. Tyto proměnné opět připojíme pomocí modulů LDf na vstupy Hi a Lo.

Dalším vstupem je vstup Bias, který udává neutrální stav akčního členu (posunutí nuly). Je to v podstatě hodnota, která se bude přičítat na výstup Out k akčnímu zásahu. Jelikož tuto funkci pro tuto úlohu nepotřebujeme, připojíme k tomuto vstupu novou proměnnou (nazvanou například Posun_nuly) a ponecháme jí hodnotu 0.

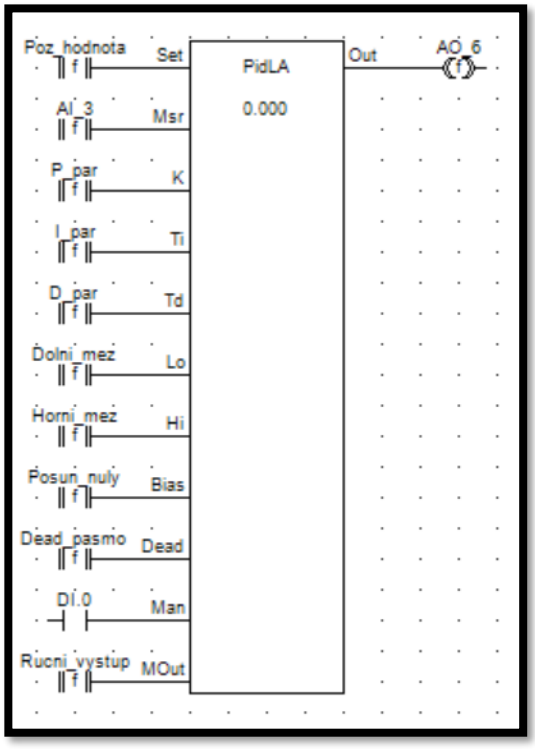
Vstup Dead označuje velikost odchylky od požadované hodnoty, ve které regulátor zůstává nečinný. Touto funkcí lze například zamezit zbytečným drobným pohybům akčního členu, které ho můžou rychleji opotřebovat. Vytvoříme tedy novou proměnnou (nazvanou například Dead_pasmo) a inicializujeme jí hodnotu 0,5. Pokud se nyní bude el. proud na vstupu regulátoru pohybovat mezi hodnotami 4,5 - 5,5 mA, regulátor nebude upravovat akční zásah.

Vstup Man umožňuje přepnutí na ruční ovládání. Pokud je na tomto vstupu log.1, regulátor přeruší svoji činost na výstup Out se bude zapisovat hodnota ze vstupu MOut. Budeme například chtít, aby v tomto režimu přestal téct obvodem el. proud (například kvůli provedení údržby). Na vstup Man tedy můžeme přivést dig. vstup DI.0 a na vstup MOut novou proměnnou (nazvanou například Rucni_vstup), které ponecháme hodnotu 0. Nyní při log. 1 na vstupu DI.0 přestane téct obvodem el. proud a regulátor bude nečinný.

Posledním krokem je připojit na výstup Out proměnnou AO_6 (jejíž hodnota je zapisována na fyzický výstup modulu vzdálených vstupů a výstupů). Seznam všech použitých proměnných je následující.

Jméno	Typ	Řádků	Sloupců	WID	Warm	Init hodnota
AI_3	F			1000	☐	
AO_6	F			1005	☐	
D_par	F			1011	☑	1
Dead_pasmo	F			1006	☑	0.5
DI	I			1008	☐	
Dolni_mez	F			1002	☑	
Horni_mez	F			1003	☑	10
I_par	F			1010	☑	1
P_par	F			1009	☑	1
Posun_nuly	F			1004	☑	
Poz_hodnota	F			1001	☑	5
Rucni_vystup	F			1007	☑	

Výsledné schéma pak bude vypadat takto.



Pozn.: Pro lepší regulaci je dobré přidat ještě filtr, který “vyhladí“ vstupní proud. Otevřeme tedy proces Vstupy a vložíme d něj modul Filtr1R (filtr prvního řádu) na jehož vstup umístíme proměnou AI_3.