UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

SET LABORATORNÍCH ÚLOH PRO LOGICKÉ ŘÍZENÍ

Jan Hatla

Bakalářská práce

2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Hatla**
Osobní číslo:	**I13056**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Řízení procesů**
Název tématu:	**Set laboratorních úloh pro logické řízení**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit skupinu robustních, uzavřených úloh pro trénování návrhu řídicího programu pro PLC Siemens Simatic S7-1200. Budou vytvořeny standardní úlohy typu semafor, dopravníkový pás, výtah, směšovač a ohřev, a další ve formě kompaktních krabiček s vizualizací jednotlivých stavů. K úlohám bude dodána kompletní dokumentace včetně laboratorního návodu.
Teoretická část: popis principů logického řízení, popis rodiny PLC a příslušenství řady Simatic, rešerše podobných publikovaných úloh.
Praktická část: návrh úloh, dokumentace k jednotlivým úlohám (včetně podrobných schémat zapojení), ukázkové algoritmy k řízení jednotlivých úloh, návody k jednotlivým laboratorním úlohám. Při řešení je třeba dbát na bezpečnost provozu a robustnost úloh.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

BALÁTĚ, J. Automatické řízení. Praha: BEN, 2003. 654 s. ISBN 80-7300-020-2.
KWASNIEWSKI, J. Programmable Logic Controllers. Cracow: ROMA-POL, 2002. ISBN 83-86320-45-1.

Vedoucí bakalářské práce: **Ing. Petr Doležel, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **13. prosince 2016**
Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 15. prosince 2016

Prohlášení

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice

V Pardubicích dne

Jan Hatla

Poděkování

Tímto bych chtěl poděkovat svým rodičům za jejich podporu. Dále děkuji Ing. Petru Doleželovi, Ph.D. za odbornou pomoc při realizaci mé práce.

V Pardubicích dne

Jan Hatla

ANOTACE

Bakalářská práce obsahuje několik laboratorních úloh pro trénování návrhu řídicího programu pro PLC Siemens Simatic S7-1200. První část bakalářské práce se zabývá seznámením čtenáře se základními dovednostmi logického řízení. Také seznámí čtenáře s PLC obecně a s programovatelným automatem Simatic S7-1200. Ve druhé části práce jsou konkrétní zadání a také řešení laboratorních úloh.

KLÍČOVÁ SLOVA

programovatelné automaty, PLC, logické řízení, laboratorní úloha

TITLE

LAB EXERCISES FOR LOGIC CONTROL

ANNOTATION

Bachelor thesis contains several laboratory exercises for the design training of the control program for PLC Siemens Simatic S7-1200. First part of bachelor thesis deals with introducing the reader to the basic skills of logical control. Thesis also introduces the reader to the PLC in general and programmable logic controller Siemens Simatic S7-1200. In the second part of bachelor thesis there are specific assignments of laboratory exercises and their solutions.

KEYWORDS

Programmable controllers, PLC, logic control, laboratory exercise

Obsah

Seznam zkratek

PLC	Programmable Logic Controller
HMI	Human–Machine Interface
TIA	Totally Integrated Automation
SCADA	Supervisory Control And Data Acquisition
PC	Personal Computer
CPU	Central Processing Unit
HW	Hardware
SW	Software
DPS	Deska plošného spoje

Seznam tabulek

Seznam obrázků

Úvod

Tato práce je rozdělena na dvě části. První část je teoretická, ve které je popsáno základní řízení pro logické obvody, jehož znalost je potřebná k pochopení této bakalářské práce. Do této oblasti patří seznámení se základními logickými operacemi, Booleovou algebrou a také obecné informace o PLC a informace o rodině PLC Siemens Simatic S7 a příslušenství řady Simatic. Ve druhé části práce se nachází seznámení s programováním PLC v programu TIA portal za použití programovacího jazyka LAD. K seznámení je zde detailně popsaný příklad s ovládáním motoru. Dále jsou konkrétní zadání laboratorních úloh a také jejich řešení. Zde řešené laboratorní úlohy slouží pro trénování, navrhování a realizaci řídicích programů pro PLC Siemens Simatic S7-1200.

1 LOGICKÉ ŘÍZENÍ

Logické řízení je činnost, při které se logickým obvodem zpracovávají informace o řízeném procesu a podle těchto informací se ovládají příslušná zařízení tak, aby se dosáhlo požadovaného výsledku.

Při logickém řízení se využívá dvouhodnotových veličin. Tyto veličiny mohou nabývat pouze dvou možných hodnot. Klasickým příkladem je ventil nebo spínač, které mohou nabývat dvou hodnot – ventil je otevřen / zavřen, spínač je sepnut / rozepnut, atd. Dvouhodnotové veličiny používáme i k určení stavu objektů – hladina vody dosáhla / nedosáhla určité hodnoty, atd. Dvouhodnotové veličiny většinou vyjadřujeme hodnotami 0 a 1. V technice to jsou nejčastěji vyskytující se logické veličiny (Švarc, 2002).

Tab. 1.1 – Často používané veličiny

Logické veličiny	0	1
Slovní vyjádření	NE	ANO
Vyjádření anglické	FALSE	TRUE
V elektrotechnice	0 V	5 V ÷ 12 V

1.2 Logické funkce

Logickou funkci lze definovat jako přiřazení hodnot 0 a 1 logické proměnné y ke kombinacím hodnot nezávislých logických (dvouhodnotových) proměnných x_1, x_2, ..., x_n. Logické funkce mohou být funkce jedné, dvou až n proměnných (Švarc, 2002).

$$y = f(x_1, x_2, \ldots, x_n)$$

Takto je definovaná logická funkce pomocí rovnice. Dále lze definovat logickou funkci pomocí pravdivostní tabulky. Tento zápis je velmi jednoduchý a názorný. Pravdivostní tabulka vyjadřuje závislost výstupních veličin na vstupních. Velikost tabulky závisí na počtu vstupů. Počet řádků je roven 2^m, kde m je počet vstupů. Do tohoto počtu můžeme přičíst jeden řádek pro popis sloupců. Počet sloupců je dán součtem počtu vstupů a počtu výstupů.

Tab. 1.2 – Pravdivostní tabulka pro dvě proměnné

x_1	x_2	y
0	0	
0	1	
1	0	
1	1	

1.3 Základní logické funkce

V této kapitole se zabývám několika nejzákladnějšími funkcemi, které jsem používal ve druhé části této práce.

1.3.1 Negace (NON)

V logickém řízení nám tato funkce mění hodnotu dvouhodnotové veličiny na opačnou. Negace je funkcí jedné proměnné. Může se zapsat takto:

$$y = \bar{x} \text{ nebo } y = \neg x.$$

Tab. 1.3 – Negace

x	y
0	1
1	0

1.3.2 Konjunkce a její negace (AND a NAND)

Konjunkce (AND) je funkce dvou nebo více proměnných, které se mezi sebou logicky vynásobí. Tedy funkční hodnota *y* nabývá hodnoty jedna pouze v případě, když obě proměnné x_1, x_2 jsou zároveň jedna. Jinak řečeno – logický součin je roven nule, právě když alespoň jedna proměnná je rovna nule (Švarc, 2002). Zapisuje se následovně:

$$y = x_1 \cdot x_2 \text{ nebo } y = x_1 \wedge x_2$$

Negace konjunkce (NAND) je funkce dvou nebo více proměnných a od konjunkce se liší pouze opačnou funkční hodnotou *y*. Tedy stačí výsledek konjunkce znegovat. Tato funkce je také známá jako Shefferova funkce (Švarc, 2002). Zapisuje se následovně:

$$y = \overline{x_1 \cdot x_2} \text{ nebo } y = \overline{x_1 \wedge x_2}$$

Tab. 1.4 – Logický součin a jeho negace

x_1	x_2	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

1.3.3 Disjunkce a její negace (OR a NOR)

Disjunkce (OR) je funkce dvou nebo více proměnných, které se mezi sebou logicky sečtou. Funkční hodnota *y* nabývá hodnoty nula pouze v případě, když obě proměnné x_1, x_2 jsou zároveň nula. Jinak řečeno – logický součet je roven jedné, právě když alespoň jedna proměnná je rovna jedné (Švarc, 2002). Zapisuje se následovně:

$$y = x_1 + x_2 \text{ nebo } y = x_1 \vee x_2$$

Negace disjunkce (NOR) je funkce dvou nebo více proměnných a od disjunkce se liší pouze opačnou funkční hodnotou *y*. Tedy stačí pouze znegovat výsledek disjunkce. Tato funkce je známá jako Piercova funkce (Švarc, 2002). Zapisuje se následovně:

$$y = \overline{x_1 + x_2} \text{ nebo } y = \overline{x_1 \vee x_2}$$

Tab. 1.5 – Logický součet a jeho negace

x_1	x_2	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

1.4 Logický klopný obvod RS a JK

Nejjednodušším a nejzákladnějším logickým klopným obvodem je právě RS. Má dva vstupy označené „R“ a „S“ a výstup, který se značí „Q“. Někdy se uvádí i negovaný výstup.

Tab. 1.6 – Klopný obvod RS

R	S	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	X

Q_n značí aktuální stav. Q_{n+1} značí stav následující. *X* značí kritický (nebo zakázaný) stav.

Vstup „R“ značí „Reset“. Přivedení logické jedničky na tento vstup způsobí, že na výstupu „Q“ bude logická nula. Vstup „S“ značí „Set“. Přivedení logické jedničky na tento vstup způsobí, že na výstupu „Q“ bude logická jednička. Pokud na oba vstupy přivedeme logickou nulu, pak na výstupu bude předcházející stav. Pokud na oba vstupy přivedeme logickou jedničku, pak na výstupu dojde k hazardnímu stavu. To znamená, že výstup není možné předem nijak definovat a tedy nelze určit, v jakém stavu bude výstup. Tomu je potřeba předejít.

Jedním způsobem jak předejít zakázanému stavu je použití klopného obvodu JK. Tento klopný obvod je téměř totožný jako je RS. Rozdíl je v tom, že tento obvod potřebuje k chodu hodinový signál a nemá hazardní stav.

Tab. 1.7 – Klopný obvod JK

K	J	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	$\neg Q_n$

Přivedením logické jedničku na vstup „J“ (jump) nastaví výstup na logickou jedničku. Přivedením logické jedničky na vstup „K“ (kill) nastaví výstup na logickou nulu. Pokud přivedeme logickou nulu na oba vstupy tak na výstupu bude předcházející stav. Pokud na oba vstupy přivedeme logickou jedničku tak na výstupu bude negovaný předcházející stav.

1.5 Booleova algebra

Při řešení logických úloh se často stává, že nestačí základní logické funkce. Jakmile se narazí na nějakou složitější funkci, je lepší ji minimalizovat a k tomu právě pomůže Booleova algebra. Booleova algebra využívá 3 základní funkce (negaci, konjunkci, disjunkci) a těmi lze vyjádřit libovolnou funkci. Také se dá použít k minimalizaci složitých funkcí, to je vyjádření funkce co nejmenším počtem základních logických funkcí. Díky čemuž se zjednoduší technická realizace, sníží se počet potřebných součástí, což bude ekonomičtější a také spolehlivější (Švarc, 2002).

K minimalizaci se používají základní pravidla Booleovy algebry, která jsou popsány níže. Tato pravidla popisují logické operace.

1.5.1 Zákon vyloučení třetího

Zákon: $x \vee \bar{x} = 1$

Důkaz: $0 \vee 1 = 1 \qquad 1 \vee 0 = 1$

1.5.2 Logický rozpor

Zákon: $x \cdot \bar{x} = 0$

Důkaz: $0 \cdot 1 = 0 \qquad 1 \cdot 0 = 0$

1.5.3 Zákon dvojité negace

Zákon: $\bar{\bar{x}} = x$

Důkaz: $\bar{\bar{0}} = \bar{1} = 0 \qquad \bar{\bar{1}} = \bar{0} = 1$

1.5.4 Zákon opakování

Zákon: $x \cdot x = x$

$x \vee x = x$

Důkaz: $1 \cdot 1 = 1 \qquad 0 \cdot 0 = 0$

$1 \vee 1 = 1 \qquad 0 \vee 0 = 0$

1.5.5 Komutativní zákony

Zákon: $x_1 \cdot x_2 = x_2 \cdot x_1$

$x_1 \vee x_2 = x_2 \vee x_1$

Důkaz: $1 \cdot 1 = 1 \cdot 1 = 1 \qquad 0 \cdot 1 = 1 \cdot 0 = 0$

$1 \vee 1 = 1 \vee 1 = 1 \qquad 1 \vee 0 = 0 \vee 1 = 1$

1.5.6 Asociativní zákon

Zákon: $x_1 \vee (x_2 \vee x_3) = x_1 \vee x_2 \vee x_3$

$x_1 \cdot (x_2 \cdot x_3) = x_1 \cdot x_2 \cdot x_3$

Důkaz: $1 \vee (1 \vee 0) = 1 \vee 1 \vee 0 = 1$

$1 \cdot (1 \cdot 0) = 1 \cdot 1 \cdot 0 = 0$

1.5.7 Distributivní zákon

Zákon: $x_1 \cdot (x_2 \vee x_3) = (x_1 \cdot x_2) \vee (x_1 \cdot x_3)$

$x_1 \vee (x_2 \cdot x_3) = (x_1 \vee x_2) \cdot (x_1 \vee x_3)$

Důkaz: $1 \cdot (1 \vee 0) = (1 \cdot 1) \vee (1 \cdot 0) = 1$

$1 \vee (1 \cdot 0) = (1 \vee 1) \cdot (1 \vee 0) = 1$

1.5.8 Absorpční zákon

Zákon: $x_1 \vee (x_1 \cdot x_2) = x_1$

$x_1 \vee (\overline{x_1} \cdot x_2) = x_1 \vee x_2$

$x_1 \cdot (x_1 \vee x_2) = x_1$

$x_1 \cdot (\overline{x_1} \vee x_2) = x_1 \cdot x_2$

Důkaz: $1 \vee (1 \cdot 0) = 1$

$1 \vee (0 \cdot 0) = 1 \vee 0 = 1$

$1 \cdot (1 \vee 0) = 1$

$1 \cdot (0 \vee 0) = 1 \cdot 0 = 0$

1.5.9 Neutrálnost

Zákon: $1 \cdot x = x$

$0 \vee x = x$

Důkaz: $1 \cdot 1 = 1$ $\quad 1 \cdot 0 = 0$

$0 \vee 1 = 1$ $\quad 0 \vee 0 = 0$

1.5.10 Agresivnost

Zákon: $0 \cdot x = 0$

$1 \vee x = 1$

Důkaz: $0 \cdot 1 = 0$ $\quad 0 \cdot 0 = 0$

$1 \vee 1 = 1$ $\quad 1 \vee 0 = 1$

1.5.11 De Morganovy zákony

Zákon: $\overline{x_1 \vee x_2} = \overline{x_1} \cdot \overline{x_2}$

$\overline{x_1 \cdot x_2} = \overline{x_1} \vee \overline{x_2}$

Důkaz: $\overline{1 \vee 0} = \bar{1} \cdot \bar{0} = 0$

$\overline{1 \cdot 0} = \bar{1} \vee \bar{0} = 1$

De Morganovy zákony jsou velmi důležité. S jejich využitím můžeme Booleovu algebru převádět na NAND nebo NOR algebru. Tyto dvě algebry jsou založeny na stejnojmenných logických obvodech. Díky NAND algebře a De Morganovým zákonům můžeme řešit různé logické úlohy pouze pomocí NAND funkce. Stejně tak platí i pro NOR algebru. De Morganovy zákony se většinou požívají v upraveném stavu, který se z původních rovnic dostane pomocí negace – kdy se zneguje pravá i levá strana rovnic (Švarc, 2002).

$$x_1 \vee x_2 = \overline{\overline{x_1} \cdot \overline{x_2}} \qquad x_1 \cdot x_2 = \overline{\overline{x_1} \vee \overline{x_2}}$$

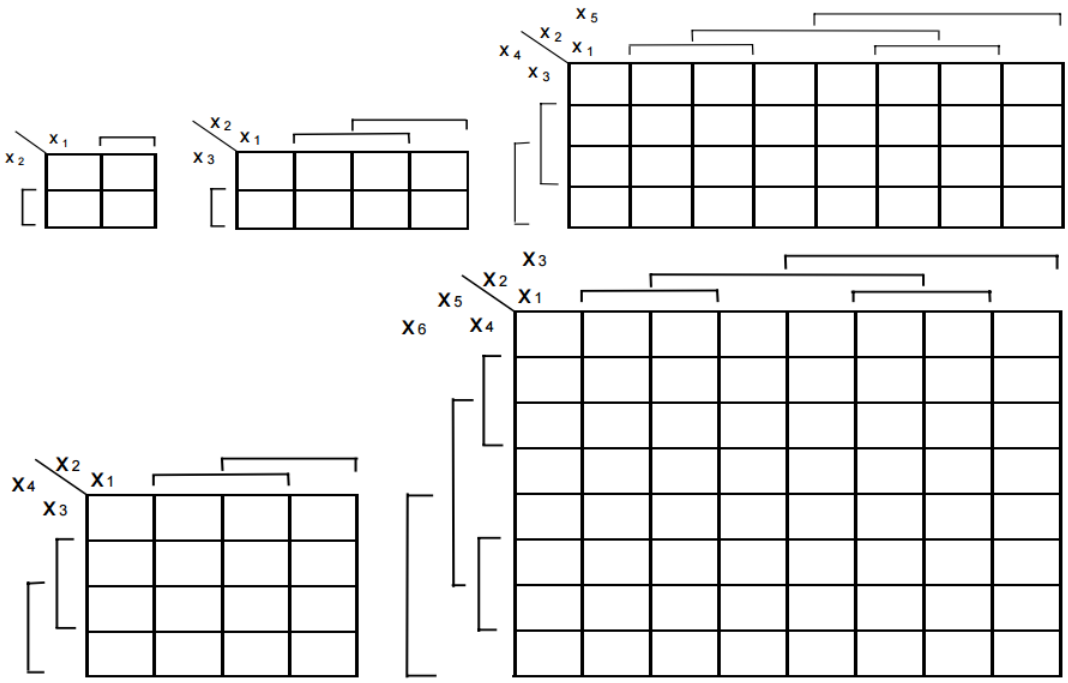
De Morganovy zákony se dají aplikovat i na více logických proměnných. Dále je napsán příklad pro tři proměnné.

$$x_1 \vee x_2 \vee x_3 = \overline{\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}} \qquad x_1 \cdot x_2 \cdot x_3 = \overline{\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}}$$

1.6 Karnaughova mapa

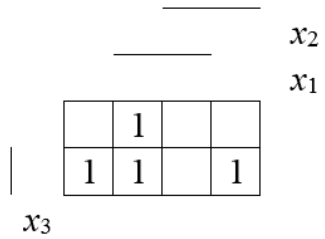
Maurice Karnaugh (narozen v roce 1924) je americký fyzik a matematik, který je znám zejména vytvořením metody „Karnaughova mapa“ (Wikipedia, 2017). Tato metoda se využívá v Booleově algebře, díky které se dá zapsat jakákoli Booleova funkce, ale také se touto metodou dají tyto funkce minimalizovat.

„Mapa je tabulka, která má tolik políček, kolik je kombinací proměnných vyšetřované funkce. Funkci s *n* proměnnými tedy vyjadřujeme mapou s 2^n políčky. Každé políčko odpovídá jedné z možných kombinací a zapisujeme do něj odpovídající funkční hodnotu. Podle kódu, kterým přiřazujeme políčka jednotlivým kombinacím proměnných, rozlišujeme různé mapy. Nejznámější je Karnaughova mapa. U ní se sousední políčka od sebe liší hodnotou jediné proměnné“ (Švarc, 2002).



Obr. 1.1 – Karnaughovy mapy pro logické funkce dvou až šesti proměnných (Švarc, 2002)

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



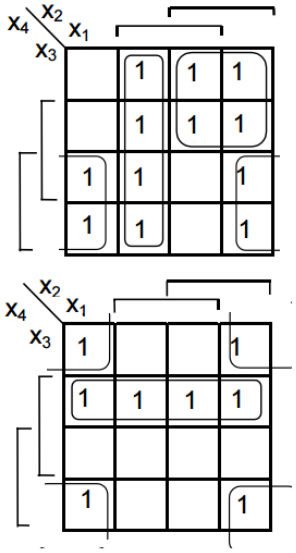
Obr. 1.2 – Příklad vyplnění Karnaughovy mapy

1.6.1 Minimalizace funkce pomocí Karnaughovy mapy

Nyní známe Karnaughovu mapu jako způsob zápisu logické funkce. Jejím hlavním účelem a nesmírnou výhodou, při řešení logických úloh, je velmi jednoduchá aplikace pro minimalizaci logických funkcí.

Postup při minimalizaci je opačný než při sestavování mapy a to nalezením algebraického tvaru funkce zadané mapou. Budeme postupovat tak, že sousední políčka mapy obsahující jedničku budeme sdružovat do dvojic, čtveřic, osmic, atd. Sdružením dvou jedniček vyloučíme jednu proměnnou, sdružením čtveřice jedniček vyloučíme dvě proměnné, sdružením osmice jedniček vyloučíme tři proměnné, atd. (Švarc, 2002). Dvojice mohou být tvořeny jako sloupec (2×1) nebo jako řádek (1×2). Čtveřice mohou být tvořeny jako sloupec (4×1) nebo řádek (4×1) nebo do tvaru čtverce (2×2). Osmice mohou nabývat tvarů: 1×8; 8×1; 2×4; 4×2.

Důležitou věcí při minimalizaci je porozumění sousednosti políček v Karnaughových mapách. Sousedními políčky jsou třeba políčka na protilehlých okrajích mapy. Pro představu je možné si mapu pomyslně zaoblit a spojit tak, že bude sousedit levý a pravý okraj a současně horní okraj se spodním. Příklady všech možných čtveřic v Karnaughových mapách jsou vyznačeny na obr. 1.3.



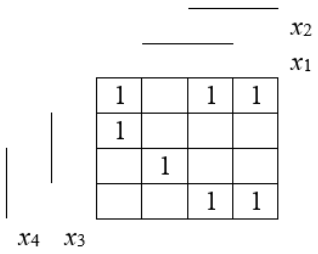
Obr. 1.3 – Příklad možných čtveřic v Karnaughových mapách (Švarc, 2002)

Z takto vyznačených skupin jedniček by se rovnou mohly psát algebraické funkce. Pro jednoduchost vysvětlím postup minimalizace na příkladu.

1.6.2 Příklad

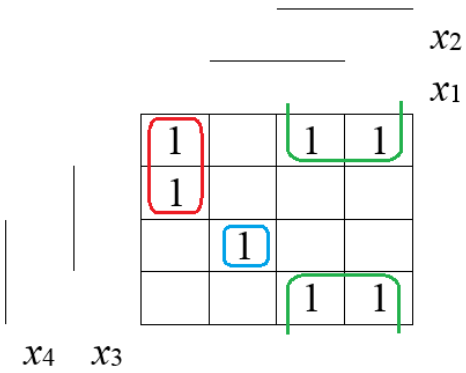
Zde si uvedeme jednoduchý příklad. Na obr. 1.4 máme zadání v tabulce, kde vidíme čtyři různé vstupní proměnné (x_1, x_2, x_3, x_4) a jednu výstupní proměnnou (y). Z této tabulky si vytvoříme Karnaughovu mapu tak abychom do ní mohli zapsat všechny kombinace z tabulky. Pro čtyři proměnné budeme tedy potřebovat mapu o velikosti 4×4. Mapu vyplníme pouze jedničkami a pouze tam kde pro kombinaci vstupních proměnných je výstupní proměnná rovna jedničce.

x_1	x_2	x_3	x_4	y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0



Obr. 1.4 – Zadání příkladu a vyplnění Karnaughovy mapy

Výpočet pomocí Booleovy algebry je možný, ovšem je zdlouhavý a v pozdějších úpravách se jednodušeji přehlédne chyba.



Obr. 1.5 – Sdružení jedniček

V Karnaughově mapě si sdružíme jedničky (jak je na obr. 1.5) a s takto sdruženými jedničkami můžeme rovnou psát algebraické funkce. Z těchto funkcí nám vypadnou některé proměnné. Tyto proměnné vypadnou v případě, že do námi zvolené dvojice, čtveřice, osmice, atd. zasáhnou pouze z poloviny. Např. pro červeně označenou dvojici na obr. 1.5 vypadne proměnná x_3, protože zasahuje pouze na spodní jedničku této dvojice. Pro zeleně označenou čtveřici vypadne z funkce proměnná x_1, protože zasahuje pouze na levé dvě jedničky a také vypadne proměnná x_4, protože ta zasahuje pouze na dvě spodní jedničky.

Při psaní algebraické funkce z Karnaughovy mapy píšeme proměnné podle toho, zda námi vybrané sdružení jedniček spadá do oblasti pro každou proměnnou. Když nám proměnná

pokryje celé jedno sdružení jedniček, tak ve výsledné funkci (pro toto sdružení) bude zapsána proměnná jako činitel. V případě, kdy nám proměnná nepokryje sdružení jedniček (ani z části), tak se do výsledné funkce zapíše jako negovaný činitel.

Výsledkem funkce pro sdružení jedniček je tedy součin vstupních proměnných. Ovšem výsledkem celé funkce z Karnaughovy mapy je součet všech funkcí pro jednotlivé sdružení jedniček. Řešení z obr. 1.5 bude tedy vypadat následovně:

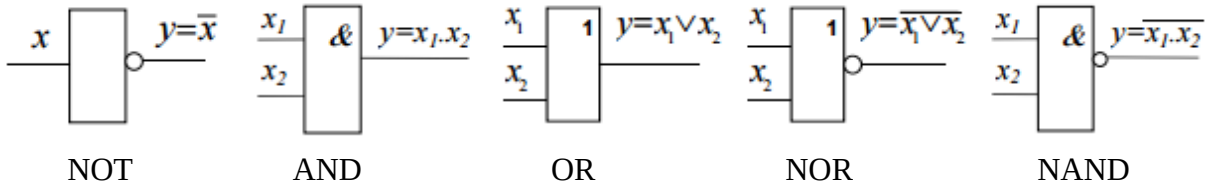
$$y = x_1 \cdot x_3 \cdot x_4 \cdot \overline{x_2} \vee \overline{x_1 \cdot x_2 \cdot x_4} \vee x_2 \cdot \overline{x_3}$$

Takto minimalizovaná funkce se již může začít realizovat.

1.6.3 Realizace logických obvodů

Samotná realizace spočívá ve správném sestavení logických prvků, které sestavujeme podle minimalizované rovnice.

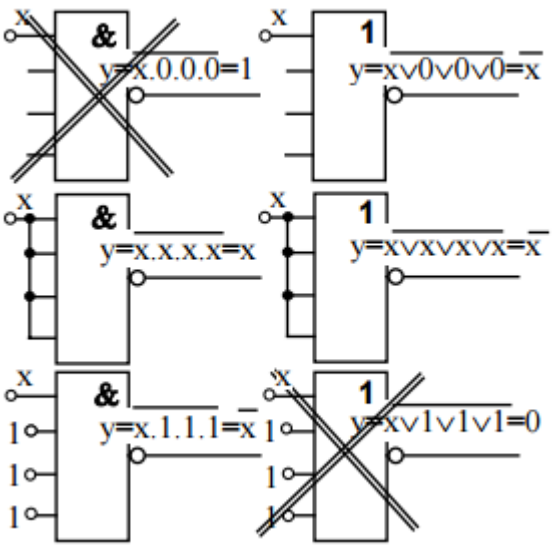
„K navrhování logických obvodů se často používají prvky NAND (negace logického součinu) a NOR (negace logického součtu), protože tyto prvky jsou snadno dostupné v širokém sortimentu a snadno se realizují. Výhodou oproti Booleovým prvkům je, že k realizaci používáme pouze jeden druh prvků, a to buď NAND anebo NOR“ (Švarc, 2002). Na obr 1.6 jsou znázorněny schématické značky pro pět základních funkcí.



Obr. 1.6 – Schématické značky základních funkcí (Švarc, 2002)

Při použití pouze prvků NAND nebo prvků NOR je dobré si ujasnit, jak se jimi realizuje funkce negace. Většinou tyto prvky mají tři nebo čtyři vstupy, které je možné:

- Ponechat volné (což je totožné jako připojit logickou nulu)
- Spojit se vstupní proměnou *x*
- Připojit na logickou jedničku



Obr. 1.7 – Realizace negace prvky NAND (vlevo) a NOR (vpravo) (Švarc, 2002)

Jak je vidět na obr. 1.7 tak při realizaci negace prvky NAND je třeba připojit na volné vstupy logickou jedničku. Možnost spojením všech vstupů vytvoří z prvku NAND pouze opakovač signálu. Připojením na volné vstupy logickou nulu (nebo ponecháním vstupů volných) získáme pouze zdroj logické jedničky (Švarc, 2002).

Zatím co při použití prvků NOR získáme funkci negace dvěma způsoby. První způsob je připojení na volné vstupy logickou nulu a druhým způsobem je propojení všech vstupů. Při připojení logické jedničky na volné vstupy získáme na výstupu pouze logickou nulu (Švarc, 2002).

Nyní si uvedeme krátký příklad na ukázku realizace Booleovými prvky, NAND prvky a NOR prvky. Nechť jsme po minimalizaci získali funkci:

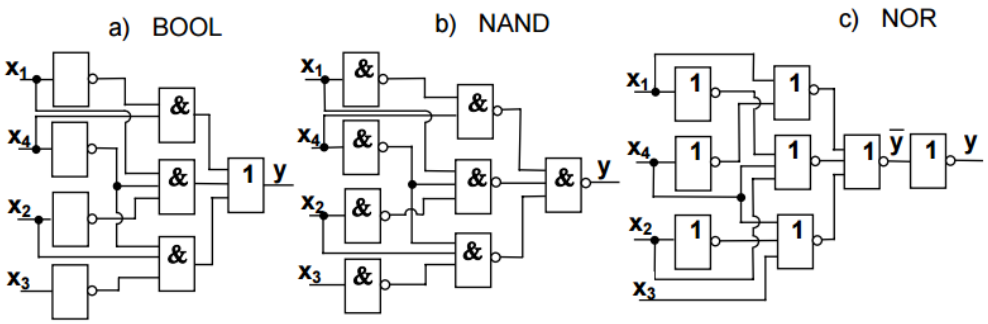
$$y = x_1\overline{x_2}\,\overline{x_4} \vee x_2\overline{x_3}\,\overline{x_4} \vee \overline{x_1}x_4$$

Z této funkce lze přímo nakreslit blokové schéma Booleovými prvky. Tuto funkci dále pak musíme upravit pomocí De Morganových zákonů na funkci vhodnou pro realizaci pomocí prvků NAND nebo prvků NOR. Výsledné funkce budou vypadat následovně (Švarc, 2002):

$$y = \overline{\overline{x_1}\,\overline{x_4}} \cdot \overline{x_2}\,\overline{x_3}\,\overline{x_4} \cdot \overline{x_1}\,\overline{x_2}\,\overline{x_4}$$

$$\overline{y} = \overline{x_1 \vee \overline{x_4}} \vee \overline{\overline{x_2} \vee x_3 \vee x_4} \vee \overline{\overline{x_1} \vee x_2 \vee x_4}$$

Vrchní funkce byla upravena pro použití prvků NAND a spodní funkce byla upravena pro použití prvků NOR. Při úpravě pro prvky NOR bylo zapotřebí znegovat levou i pravou stranu rovnice. Pokud budeme chtít pouze *y*, pak je třeba využít ještě jednoho prvku NOR a to pouze pro negaci (Švarc, 2002). Výsledná bloková schémata jsou na obr. 1.8.



Obr. 1.8 – Výsledná bloková schémata (Švarc, 2002)

Ze schémat na obr. 1.8 je vidět, že na realizaci pomocí Booleových prvků budeme potřebovat:

- Čtyři prvky pro negaci
- Tři součinové prvky
- Jeden sčítací prvek

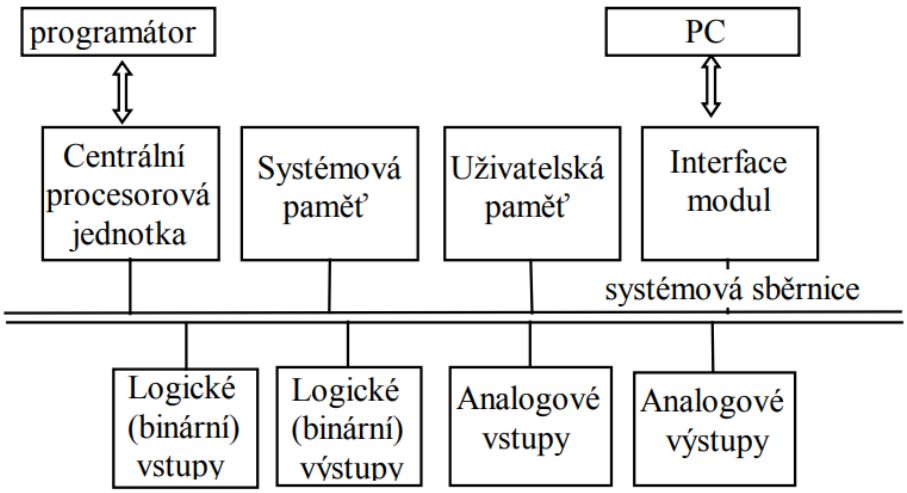
Zatím co pro realizaci pomocí prvků NAND budeme potřebovat osm prvků a pomocí prvků NOR bychom potřebovali také osm prvků. Veliká výhoda realizace pomocí těchto dvou metod spočívá v použití pouze jednoho druhu prvků (Švarc, 2002).

2 PROGRAMOVATELNÉ AUTOMATY

„Programovatelné automaty jsou programovatelné řídicí systémy umožňující řízení průmyslových a technologických systémů a procesů, u starších typů a u menších systémů specializované na úlohy převážně logického typu. Jsou známé pod označením PLC (Programmable Logic Controller). Menší typy bývají řešeny jako kompaktní celky, větší se zásadně konstruují jako modulární. V automatizační technice se programovatelné automaty používají zhruba od roku 1970. Původně byly určeny pro řízení strojů, jako náhrada za pevnou reléovou logiku. Postupně se jejich možnosti rozšiřovaly a dnes se s nimi můžeme setkat v nejrůznějších oborech, kde mnohdy vytlačují dříve používané přístroje. Jsou to nejenom tradiční strojírenské výrobní technologie včetně manipulační a dopravní techniky, ale i energetika (regulace v elektrárnách, v kotelnách, v klimatizačních jednotkách i chladicích zařízeních). Uplatnění mají programovatelné automaty rovněž i v chemických výrobách, farmacii, v zemědělských výrobnách atd.

Velkou předností programovatelných automatů je jejich univerzálnost. Již patří minulosti, že PLC řešily jen logické úlohy, zatímco k řízení spojitých veličin se používají spojité PID regulátory.

Programem PLC lze řešit i jinak velmi obtížné úlohy, kde jsou vazby mezi regulací různých veličin (např. teploty a vlhkosti), lze jím optimalizovat technologický proces a přizpůsobovat jej měnícím se podmínkám. Některé PLC mají zabudovanou i fuzzy logiku, a tím se rozšíří možnosti jejich použití i do dalších odvětví, např. do diagnostiky a zabezpečovací techniky“ (Švarc, 2002).



Obr. 2.1 – Vnitřní struktura PLC (Švarc, 2002)

„Vnitřní struktura PLC je znázorněna na Obr. 2.1. Pokud se jedná o modulární provedení, má pochopitelně variabilní počet vstupních i výstupních jednotek i dalších zařízení. Funkční bloky jsou propojeny prostřednictvím jedné nebo dvou sběrnic. Modulové jednotky běžně osazované v PLC jsou centrální procesorová jednotka, systémová paměť, uživatelská paměť, interface umožňující spojení s PC a množství modulů pro analogové, digitální a binární (logické) vstupy. Skutečnou sestavu volí uživatel tak, aby programovatelný automat co nejlépe vyhovoval řešeným úlohám. V krajních případech může mít PLC dvouhodnotové vstupy a výstupy a být vystavěn jako čistě binární (logický) systém anebo naopak může být koncipován jako analogový“ (Švarc, 2002).

Do binárních vstupů se často používají tlačítka, přepínače, různé další dvouhodnotové spínače (např. snímače tlaku, teploty, hladiny, atd.). Binární výstupy se často používají k buzení cívek relé, stykačů, elektromagnetických spojek, pneumatických a hydraulických převodníků, k ovládání signálek, ale i ke stupňovitému řízení pohonů a frekvenčních měničů (Švarc, 2002).

„K analogovým vstupům lze připojit například snímače teploty (obvykle odporové, polovodičové nebo termočlánky), snímače tlaku, vlhkosti, hladiny ale i většinu inteligentních přístrojů s analogovými výstupy. Prostřednictvím analogových výstupů lze ovládat spojité servopohony a frekvenční měniče, ale třeba i ručkové měřicí přístroje a jiné spojitě ovládané akční členy“ (Švarc, 2002).

2.1 Historie PLC

Český název programovatelné automaty (PA) vznikl v TESLE Kolín v 70. letech jako překlad anglického PLC – Programmable logic controller nebo německého SPS (Specherprogramierbare Steuerung). A to v době, kdy vlastním vývojem uvedla na trh NS-910, první řídicí systém v kategorii, která již tehdy měla ve světě své označení a některé psané i nepsané standardy. Ty psané standardy vyústily nakonec do světového standardu IEC-61131, který do svého systému harmonizovaných norem zahrnuje i EU jako EN 61131 a Česká republika jako ČSN EN 61131 (Trochu historie: Programovatelné automaty, 2009).

2.2 Rodina automatů Simatic S7

V roce 1994 byla na český trh uvedena nová řada programovatelných automatů Simatic S7. Rodina S7 byla členěna do tří kategorií podle potřebné výkonnosti.

Podle výkonnosti se daly automaty roztřídit na následující skupiny:

- Simatic S7-200 – mikro PLC
- Simatic S7-300 – dolní rozsah výkonnosti
- Simatic S7-400 – střední a nejvyšší rozsah výkonnosti

V roce 2009 Siemens přináší na trh řadu PLC Simatic S7-1200. O tři roky později pak firma Siemens představuje PLC S7-1500.

2.3 PLC Simatic S7-1200

„Nové modulární PLC SIMATIC S7-1200 je jádro nového automatizačního systému navrženého pro jednoduché automatizační úlohy. Společně s vizualizačními panely řady SIMATIC HMI Basic, navrženými pro bezproblémovou spolupráci s novým PLC a novým integrovaným softwarem, zaručují jednoduchou tvorbu aplikace, rychlé uvedení do provozu a precizní monitoring. Souhra mezi těmito produkty a jejich inovativní vlastnosti Vám zaručí bezprecedentní úroveň efektivnosti při řešení všech malých automatizačních úloh.

Nový SIMATIC S7-1200 je modulární, kompaktní, univerzální a bezpečná investice do řídicího systému širokého spektra automatizačních aplikací. Flexibilní design, komunikační rozhraní, které splňuje nejvyšší standardy průmyslové komunikace a široké spektrum integrovaných funkcí dělají toto PLC základní součástí automatizačních úloh“ (Siemens, 2012).

2.4 Jazyk pro programování PLC

Už od roku 1993 existuje standard známý jako „61131“, který vydala mezinárodní elektrotechnická komise (anglická zkratka IEC) a právě v tomto standardu najdeme část nazvanou „IEC 61131 – 3“. Tato část standardu obsahuje dokumenty týkající se jazyků PLC. Cílem bylo sjednocení již existujících jazyků (Marčan, 2018).

V tomto standardu se píše o pěti jazycích:

- Seznam instrukcí – IL (Instruction List)
- Strukturovaný text – ST / SCL (Structured Text / Structured Control Language)
- Kontaktní (reléová) schémata – LD / LAD (Ladder Diagrams)
- Funkční bloky – FBD (Function Block Diagram)
- Sekvenční funkční diagram – SFC (Sequential Function Chart)

Jazyky SCL, LAD a FBD používá SIMATIC S7 a jsou certifikovány podle standardu IEC 61131 – 3.

2.4.1 Jazyk IL (Instruction List)

Jedná se o textový jazyk nižší úrovně. Program je tvořen seznamem instrukcí a velmi připomíná assembler. Každá nová instrukce začíná na novém řádku. V dnešní době se používá málokdy, protože již existují přehlednější způsoby programování (Marčan, 2018).

```
      U(
      U(
      UN    "First_Scan"              M50.7
      SPBNB _00c
      L     100
      T     "MaxEngINT".Umid          DB5.DBW4
      SET
      SAVE
      CLR
_00c: U     BIE
      )
      SPBNB _00d
      L     "MaxEngINT".Umid          DB5.DBW4
      ITD
      T     "MaxEngDINT".Umid         DB6.DBD8
      SET
      SAVE
      CLR
_00d: U     BIE
      )
      SPBNB _00e
      L     "MaxEngDINT".Umid         DB6.DBD8
      DTR
      T     "MaxEngREAL".Umid         DB7.DBD8
_00e: NOP   0
```

Obr. 2.2 – Příklad programovacího jazyka IL (Marčan, 2018)

2.4.2 Jazyk ST (Structured Text)

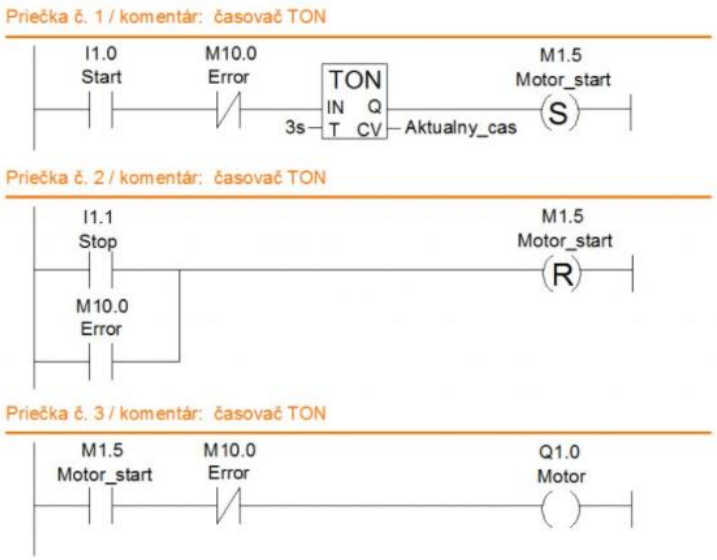
Jde opět o textový jazyk, ale vyšší úrovně. Tento programovací jazyk je srovnatelný s jazykem „C“ nebo Pascal. Na rozdíl od jazyku IL je možné psát více příkazů na jeden řádek a jednotlivé příkazy jsou zakončeny středníkem (Marčan, 2018).

```
        (* TEST CYCLE SETUP  *)
        cycle_In1 := tb_In1[testCycleNum];
        cycle_In2 := tb_In2[testCycleNum];
        cycle_In3 := tb_In3[testCycleNum];
        cycle_Out := tb_Out[testCycleNum];
        IF testCycleNum = 0 THEN
            (* INIT  *)
            PID_Subsystem(i0_PID_Subsystem, 0, cycle_In1, cycle_In2, cycle_In3, out_Out);
        END_IF;
        (* STEP  *)
        PID_Subsystem(i0_PID_Subsystem, 1, cycle_In1, cycle_In2, cycle_In3, out_Out);
        (* VERIFY  *)
        IF testVerify THEN
            IF cycle_Out = 0.0 THEN
                IF ABS(out_Out) > 9.9999997473787516E-5 THEN
                    testVerify := 0;
                END_IF;
            ELSIF ABS(out_Out - cycle_Out) > (9.9999997473787516E-5 * ABS(cycle_Out)) THEN
                testVerify := 0;
            END_IF;
        END_IF;
        testCycleNum := testCycleNum + 1;
    END_IF;
END_IF;
```

Obr. 2.3 – Příklad programovacího jazyka ST (Marčan, 2018)

2.4.3 Jazyk LD (Ladder Diagrams)

Jazyk kontaktních schémat (někdy nazýván jako jazyk reléových schémat) je jazyk grafický. Dnes velmi populární a využívaný jazyk. Je oblíbený díky své jednoduchosti, přehlednosti při zápisu, přehlednosti při čtení a také díky rychlému zjištění chyby v kódu.

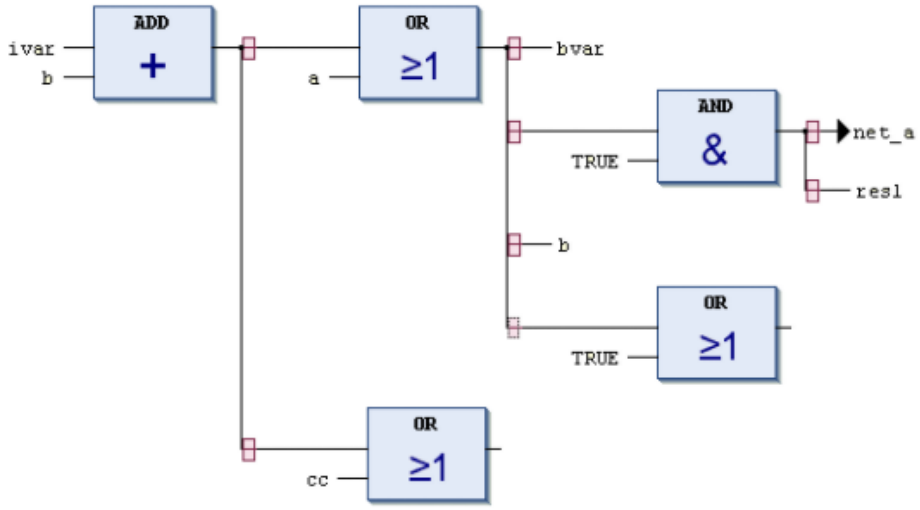


Obr. 2.4 – Příklad programovacího jazyka LD (Marčan, 2018)

Síť v tomto jazyce je ohraničena zleva i zprava svislou čarou. Tyto čáry znázorňují sběrnici. Mezi nimi je logické propojení, na které se vkládají různé spínací, rozpínací, časové, logické a jiné prvky. Propojení mezi jednotlivými prvky může být ve stavu zapnuto nebo vypnuto.

2.4.4 Jazyk FBD (Function Block Diagram)

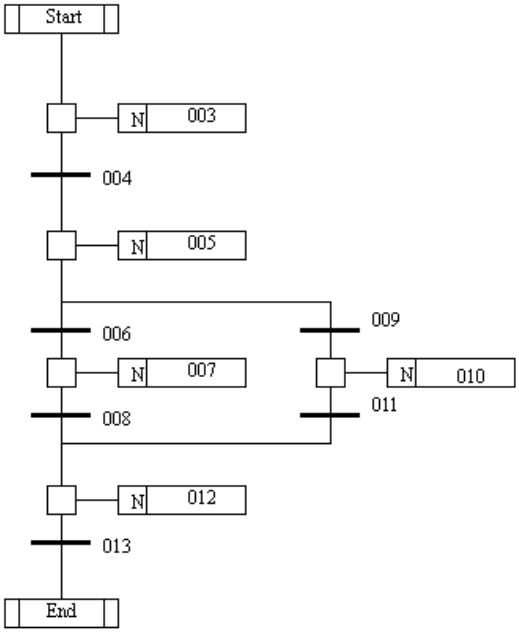
Druhým grafickým jazykem je FBD, ve kterém využíváme funkční bloky. Na tyto bloky se připojují vstupy a výstupy. Používají se standardní funkční bloky pro vyjádření logických funkcí, časovačů, matematických operací a dalších… Tyto bloky se propojí, což ucelí program, který je třeba předem definovat (Marčan, 2018).



Obr. 2.5 – Příklad programovacího jazyka FBD (Marčan, 2018)

2.4.5 Jazyk SFC (Sequential Function Chart)

Jazyk SFC je považován také za grafický. Tento typ programování je ideální, pokud chce programátor napsat program na nějaký sled událostí (Marčan, 2018). Programování je jednoduché a přehledné. Programují se stavy systému a akce, které se mají vykonávat v jednotlivých stavech. Dále se musí naprogramovat podmínky, které se musí splnit, aby program přešel z jednoho stavu do jiného.



Obr. 2.6 – Příklad programovacího jazyka FBD (Marčan, 2018)

Programování za použití grafického jazyka je jednodušší pro začátečníky. Je to převážně díky větší přehlednosti při psaní, ale i při čtení již napsaného programu. Také se v programu napsaném grafickým jazykem lépe hledají chyby a nedostatky. Dalším pozitivem je, že vývojové prostředí pro psaní programů mívá funkci pro lepší vizualizaci. Což jsou důvody, kvůli kterým jsem si při své práci zvolil právě grafický jazyk LAD. Tímto jazykem jsou napsány i úlohy v praktické části této práce.

3 TIA PORTAL

Totally Integrated Automation Portal, zkráceně TIA portal. Je vysoce komplexní softwarové prostředí s jednoduchým ovládáním od firmy Siemens. Umožňuje vývoj uživatelských aplikací pro PLC a decentrální periferie, projektování panelů HMI, rozsáhlých vizualizací SCADA, síťových komponent a komunikačních prvků i konfigurace a uvádění pohonů do provozu.

Výhodou TIA portalu je, že jeho součástí je prostředí Step 7 a WinCC. Zatím co v prostředí Step 7 se programují PLC, tak v prostředí WinCC se vytváří aplikace pro panely HMI, také nabízí řešení SCADA a vizualizaci na PC. To je pro uživatele velmi výhodné, protože může v jednom programu jednoduše a intuitivně programovat více věcí.

Na obrázku níže je znázorněna komplexnost TIA portalu: HMI, Network, Drivers, Safety, PLC, Diagnosis (Totally Integrated Automation Portal, 2017).

Obr. 3.1 – Komplexnost TIA portalu (Totally Integrated Automation Portal, 2017)

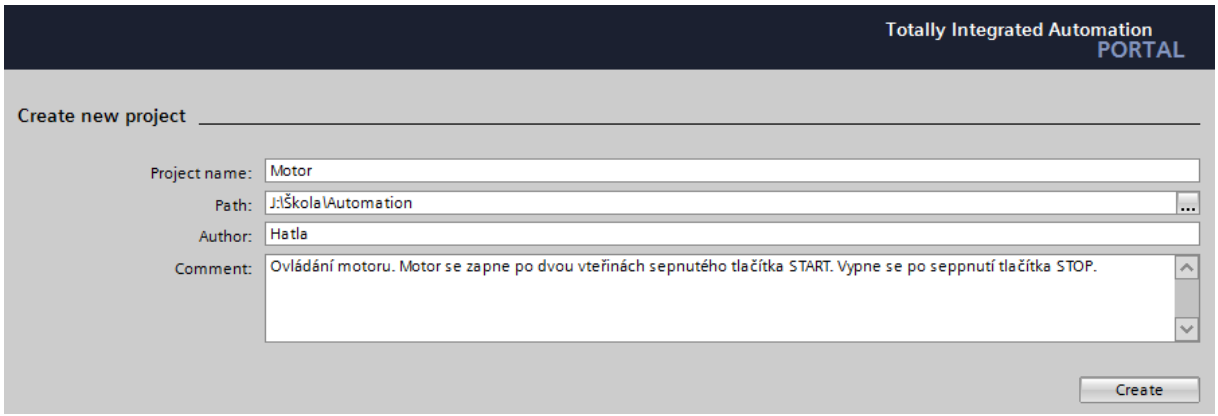
3.1 Začínáme s TIA portálem

V této podkapitole je jednoduchý příklad naprogramování ovládání motoru. K příkladu zní zadání takto: Naprogramujte ovládání motoru pomocí PLC tak, aby se motor zapnul po dvou sekundách stisknutého tlačítka START a zůstal zapnutý, dokud se nestiskne tlačítko STOP.

Ze zadání plyne, že budou potřeba dva vstupy (pro každé tlačítko jeden), jeden výstup pro motor a nějaké čekání nebo zdržení signálu.

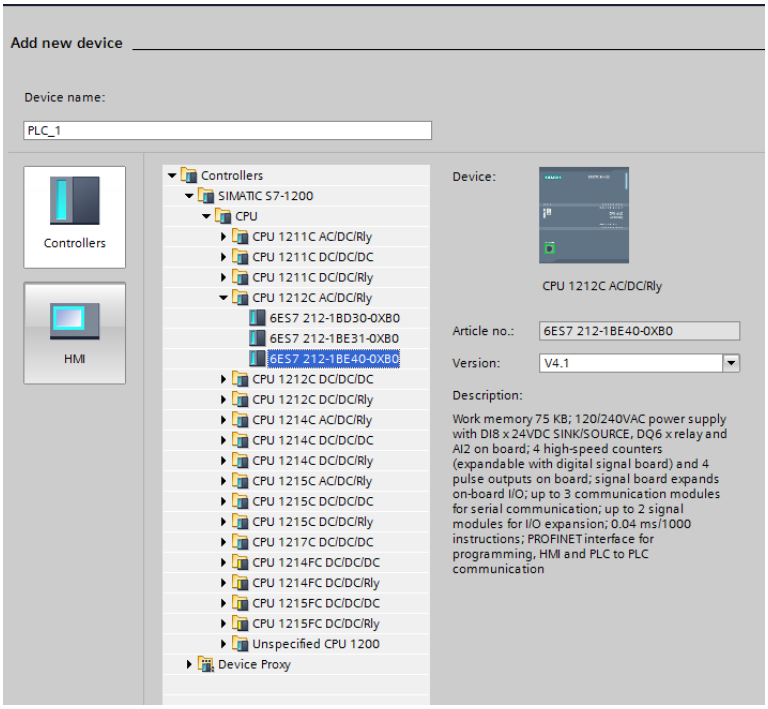
3.1.1 Založení projektu

Po spuštění programu TIA portal se na monitoru zobrazí úvodní nabídka, ve které lze otevřít existující projekt, vytvořit nový projekt, migrovat projekt a projekt zavřít. Vyberu tedy nabídku vytvoření nového projektu. Při vytváření projektu zadám název projektu, umístění na úložném zařízení, jméno autora a jako poslední je nepovinné pole pro komentář. Poté mohu vytvořit prázdný projekt, do kterého jako další krok přidám zařízení, která budu chtít použít.



Obr. 3.2 – Založení nového projektu

Přes tlačítko „Configurate a device“ nebo „Add new device“ přejdu k výběru zařízení. V mém případě vyberu zařízení, které je označené na obrázku níže.

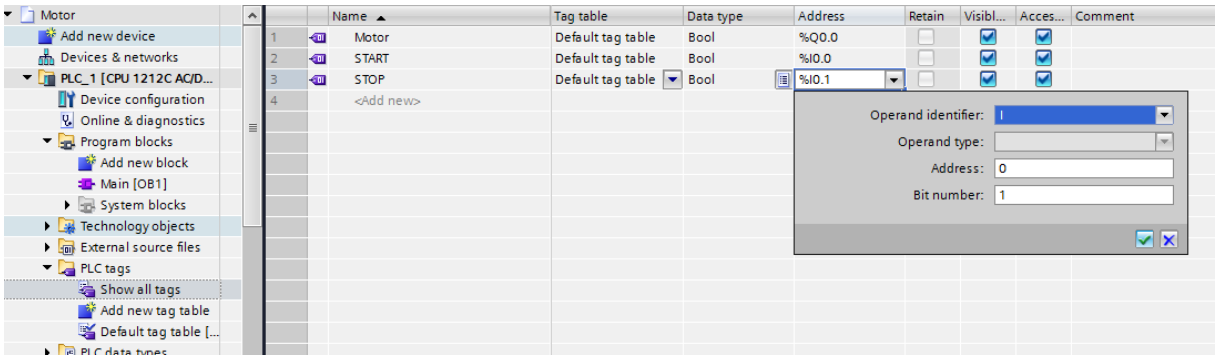


Obr. 3.3 – Výběr zařízení v TIA portal

Po výběru procesoru uvidím znázorněné PLC v programu. Jako další je potřeba definovat vstupy a výstupy.

3.1.2 Tagy

Z nabídky na levé straně vyberu složku „PLC tags“ a v ní „Show all tags“. Zde si dvojklikem na „<add new>“ mohu přidat „tag“ což je odkaz na určitý vstup či výstup na PLC, který si také definuji. Určím jeho název, o jaký jde typ (v mém případě použiji typ „bool“, což je dvouhodnotová (nebo také dvoustavová) veličina a v mém případě mi bude určovat, zda tlačítko stisknuté je nebo není a jestli motor zapnutý je nebo není) a poslední důležitá činnost je přiřazení adresy. Adresa zastupuje konkrétní kontakt na PLC. Označení adres (Q0.0, Q0.1, Q0.2, …., I0.0, I0.1, ….) je napsáno u každého kontaktu na PLC. Přidám celkem tři „tagy“ podle následujícího obrázku.



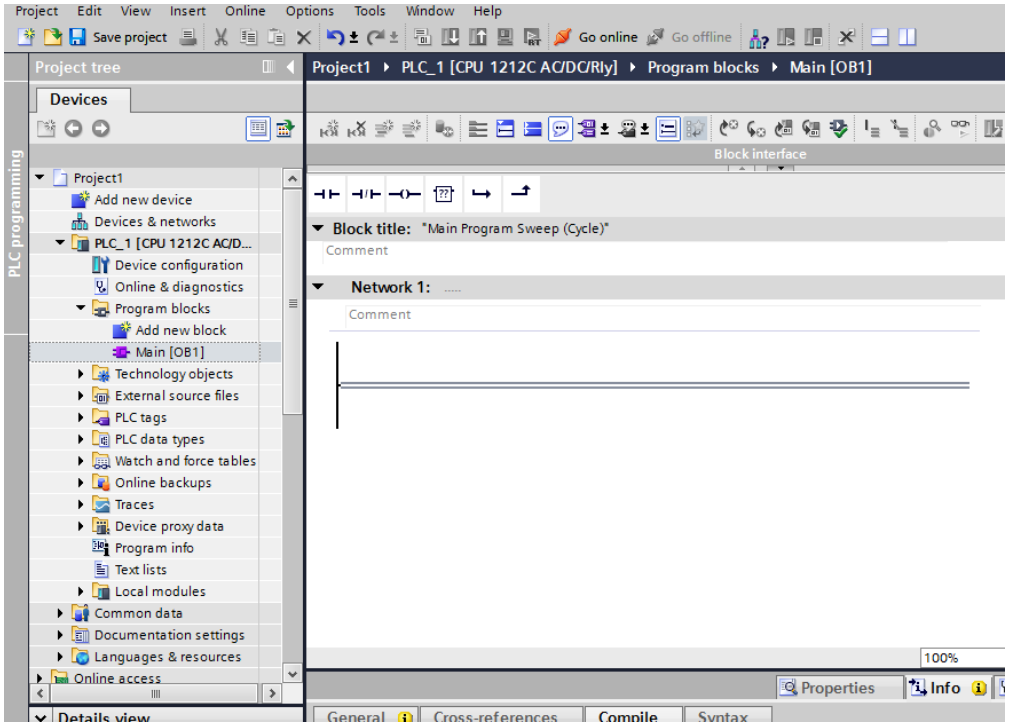
Obr. 3.4 – Nastavení „tagů“

Toto nastavení lze slovně vyjádřit následovně:

- Proměnná *Motor* je výstupní, dvouhodnotová veličina s adresou Q0.0
- Proměnná *START* je vstupní, dvouhodnotová veličina s adresou I0.0
- Proměnná *STOP* je vstupní, dvouhodnotová veličina s adresou I0.1

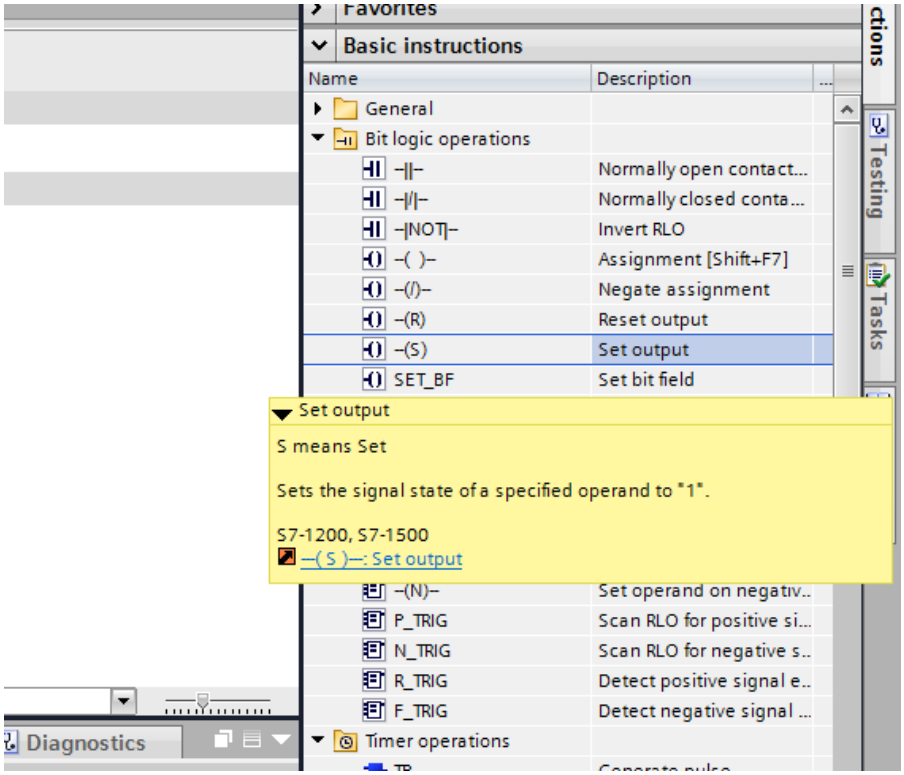
3.1.3 Step 7

Teď už mohu přejít k samotnému programu. Přejdu tedy do složky „Program block“ a tam se již nachází vytvořený program s názvem „Main“. Tento program je výchozí a je automaticky vytvořen při založení nového projektu. Po otevření tohoto programu (otevřeme dvojklikem) uvidím prázdný program v jazyce LAD, jak je vidět na obrázku níže:



Obr. 3.5 – Výchozí program v TIA portalu

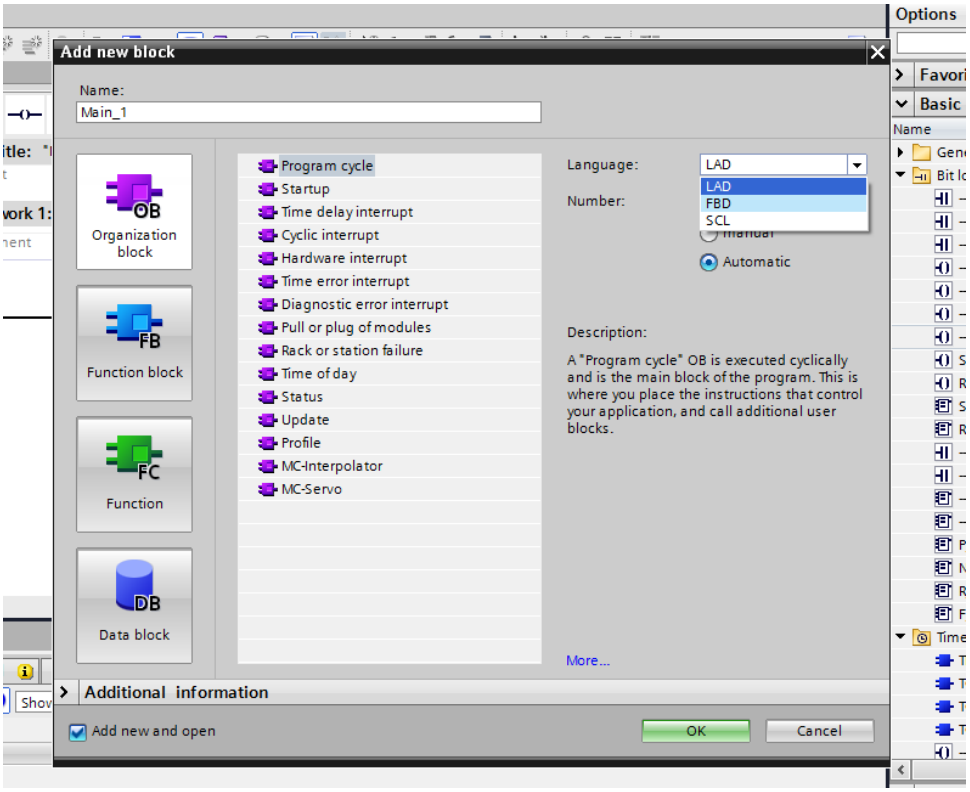
Na pravé straně jsou složky s instrukcemi, ve kterých jsou různé logické, časové, matematické, přesouvající, porovnávající a další operace. Ty je možné jednoduše přesunout do programu. Po vybrání nějaké operace se po chvíli zobrazí krátký popis operace. Viz obrázek níže. Pro detailnější popis operace lze vybrat určitou operaci a zmáčknout klávesu F1 na klávesnici.



Obr. 3.6 – Operace s krátkým popisem

3.1.4 Uživatelský program

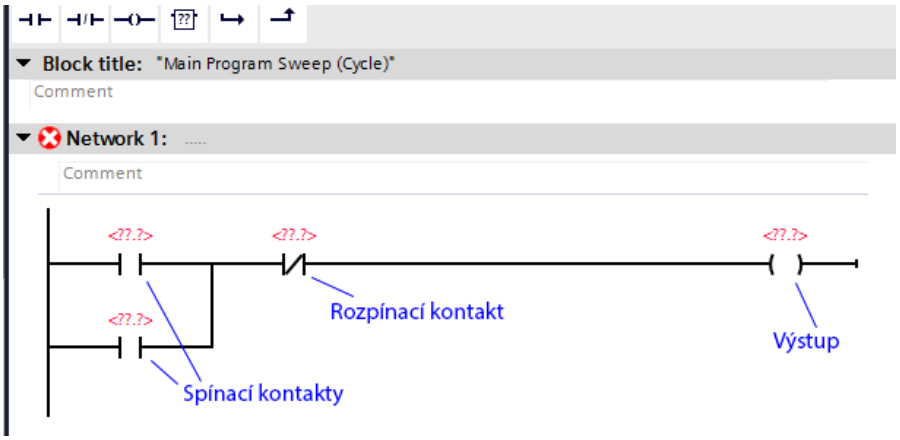
K založení nového bloku programu, funkce nebo datového bloku slouží tlačítko „Add new block“. Po dvojkliku na toto tlačítko se zobrazí okno s výběrem různých bloků a různých druhů programů, u kterých je napsaný popis k čemu slouží. Dále je zde možnost si vybrat jiný jazyk pro nový program. Pro zadání s motorem bude stačit základní a již vytvořený výchozí program.



Obr. 3.7 – Přidání nového bloku

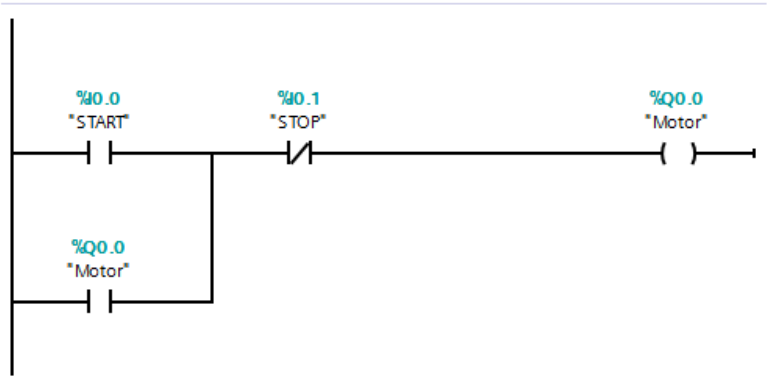
Ve vytvořeném programu (s programovacím jazykem LAD) vyberu a přesunu jisté operace tak, aby program vypadal jako na obr. 3.8. Vše potřebné najdu mezi šesti tlačítky, které se nachází na horní straně obr 3.8.

Použijí se dva spínací kontakty zapojené paralelně (logický součet) a k nim do série jeden rozpínací kontakt (logický součin). Za ten se následně umístí výstup. Pro větvení programu se použijí šipky z vrchní nabídky šesti operací.



Obr. 3.8 – Struktura programu pro ovládání motoru

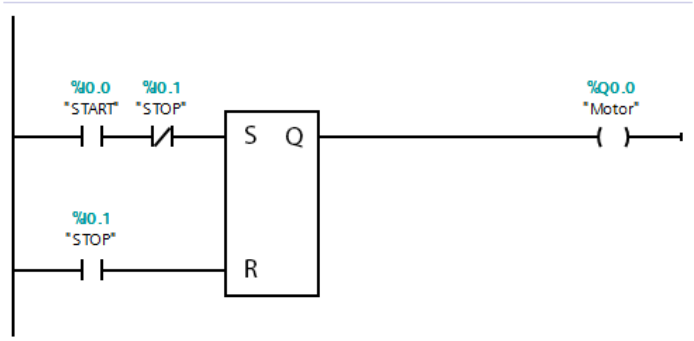
Dále ke kontaktům a výstupu přiřadím „tagy“, které jsem si dříve nastavil. Přiřadím je tím, že kliknu na červené otazníky u každé operace a začnu psát název nějakého „tagu“. Při tom se také objeví tabulka s nabídkou „tagů“ kde mohu nějaký vybrat. Tímto způsobem přidám „tagy“ ke všem kontaktům a výstupu tak, aby program vypadal stejně jako na obrázku níže.



Obr. 3.9 – Přidání „tagů“ do programu

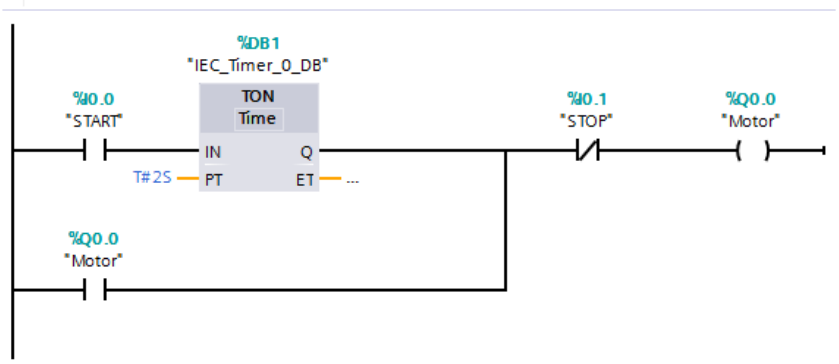
K jednomu spínacímu kontaktu jsem přiřadil stejný „tag“ jako k výstupu. To znamená, že tento kontakt se sepne, když bude na stejně označeném výstupu logická jednička. Tento způsob nám zajistí, že při puštění tlačítka „START“ se motor nevypne. Toto je jedno z mnoha řešení.

Podobné řešení by mohlo být například s logickým obvodem RS, kdy se zapojí spínací kontakt „START“ na vstup „S“. Na vstup „R“ se zapojí spínací kontakt „STOP“ a na výstup „Q“ se zapojí „Motor“. Při tomto zapojení může dojít k hazardnímu stavu a je nutné ho nějak ošetřit. Jedno řešení je dát rozpínací kontakt „STOP“ do série ke spínacímu kontaktu „START“. Při zapojení jako je na obrázku níže, nemůže dojít k hazardnímu stavu, protože při stisknutí obou tlačítek současně signál neprojde přes rozepnutý kontakt „STOP“ (tedy se nedostane na vstup „S“) a výstup klopného obvodu nastaví na logickou nulu a na výstupu pro motor bude též logická nula.



Obr. 3.10 – Znázorněné zapojení pomocí klopného obvodu RS

Dále budu pokračovat v příkladu bez klopného obvodu RS. Při tomto zapojení není splněna podmínka ze zadání a to, že se má motor zapnout po dvou sekundách sepnutého tlačítka „START“. To vyřeším tak, že za spínací kontakt „START“ přidám operaci, která je nazvána „TON“. Při přidávání tohoto bloku se objeví okno, ve kterém si mohu zvolit časovač, který má tento blok použít. Pokud zadám název časovače, který ještě v programu není použitý, tak se vytvoří nový. Když se každé operaci využívající nějaký časovač přidělí originální časovač, tak je menší pravděpodobnost udělat v programu chybu. Když na vstupu (IN) do bloku „TON“ bude logická jednička, tak se spustí přidělený časovač a ten běží do doby „PT“ (zapisováno v milisekundách) a pak na výstupu (Q) bude logická jednička. Jakmile se na vstup (IN) připojí logická nula, tak bude okamžitě i na výstupu (Q). Pro více informací o různých operacích doporučuji nápovědu v TIA portalu pod klávesou F1. Já tedy nastavím na vstup „PT“ dva tisíce milisekund. Program bude vypadat jako na dalším obrázku.

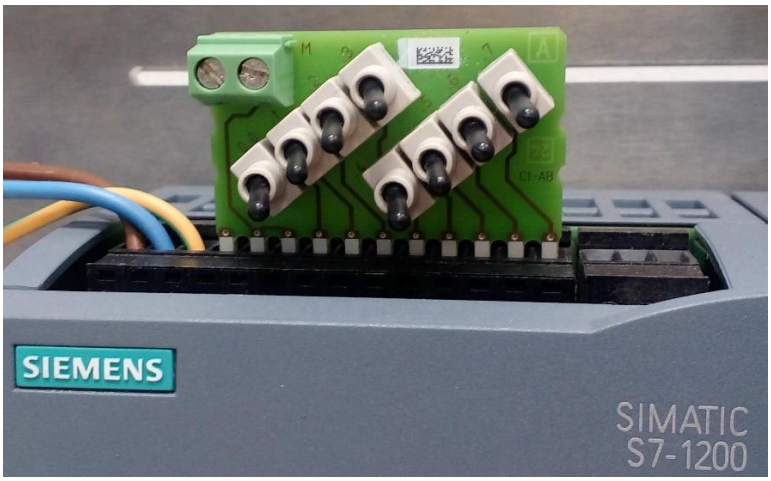


Obr. 3.11 – Přidání zpoždění do programu

Toto zapojení řeší problematiku se zmáčknutými tlačítky „START“ a „STOP“ současně. Při zmáčknutí tlačítka „STOP“ se rozepne kontakt před motorem a signál k motoru nedojde (motor se tedy nezapne) a to nezávisle na tlačítku „START“.

3.1.5 Simulace vstupů

V tomto stavu je již program kompletní a může projít kompilací a následně se může nahrát do PLC. Před kompilací programu na vstupní kontakty PLC připevním desku plošného spoje s vypínači, které použiji jako tlačítka „START“ a „STOP“. Připojení desky s vypínači je na obrázku níže.

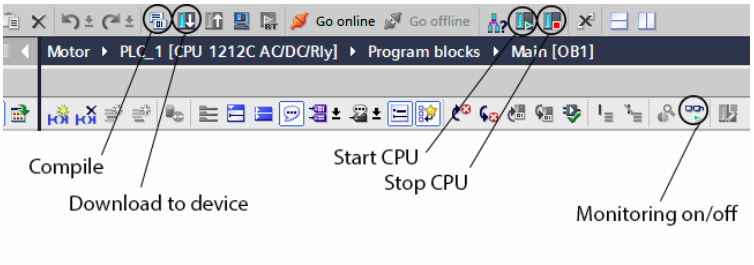


Obr. 3.12 – Připojení vypínačů k PLC

Při takto zapojených vypínačích odpovídá číslo u vypínače číslu bitu vstupního kontaktu. Tedy vypínač 0 je připojen na vstup I0.0 a vypínač 1 je připojen na vstup I0.1 atd. Vypínač 0 tedy reprezentuje tlačítko „START“ a vypínač 1 reprezentuje tlačítko „STOP“.

3.1.6 Přenos mezi PC a PLC

Jako další krok je nutné připojit PLC k počítači přes síťový kabel. Dále je nutné připojit PLC do elektrické sítě a tím ho zapnout. Poté už je vše nachystané na nahrání programu. Pro kompilaci, nahrání, spuštění a testování napsaného programu mi poslouží tlačítka znázorněna na následujícím obrázku.

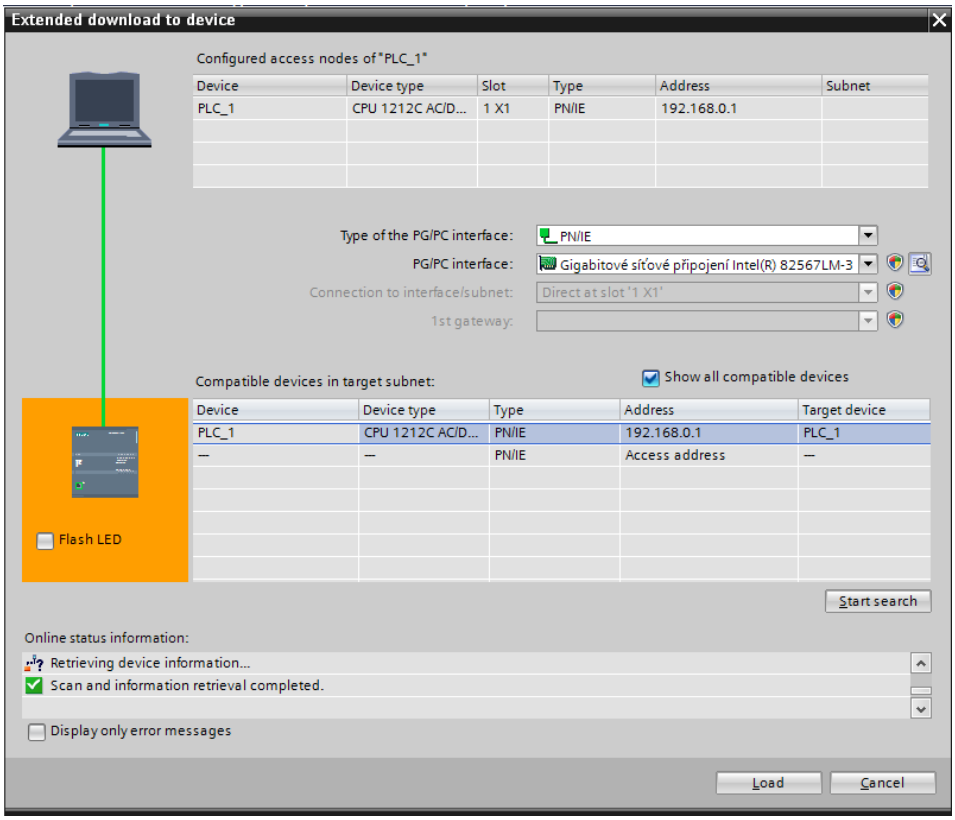


Obr. 3.13 – Popis tlačítek v TIA portal

- „Compile“ zkompiluje program
- „Download to device“ slouží k nahrání zkompilovaného programu do PLC
- „Start CPU“ spustí program nahraný v procesoru
- „Stop CPU“ zastaví běžící program v procesoru
- „Monitoring on/off“ je nástroj pro vizualizaci průběhu programu
- „Go online“ přepnutí do „online“ režimu

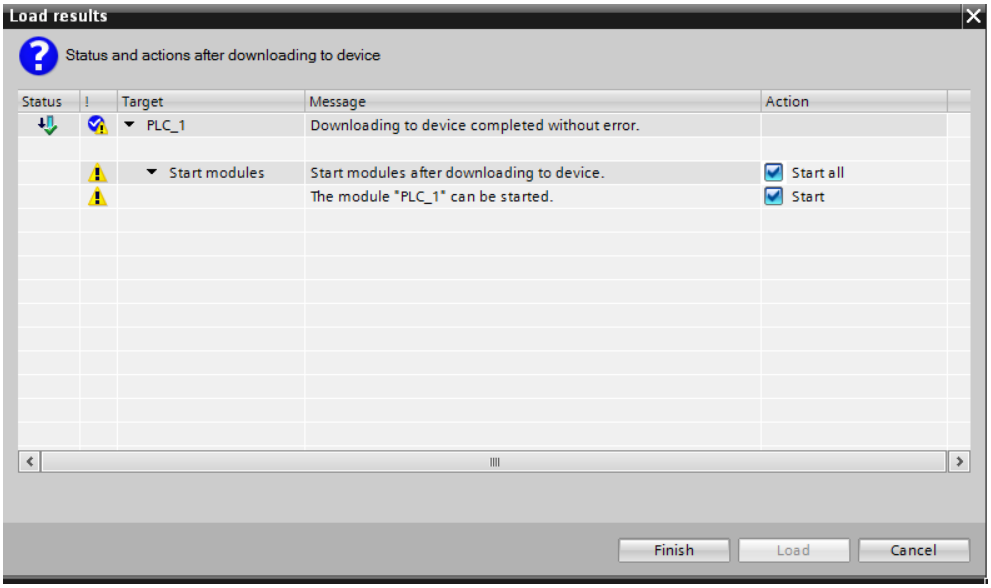
„Online“ režim slouží k testování uživatelských programů, k zobrazování a změnám provozního režimu CPU, zobrazení informací o modulech, zobrazení a nastavení data a času CPU, k diagnostice HW a dalším (Siemens, 2014).

Teď už je vše připravené a dám tedy kompilovat program (tlačítko „Compile“ lze stisknout, pouze pokud má uživatel otevřené a aktivní okno s programem, který chce kompilovat). Potom dám nahrát zkompilovaný program do zařízení přes tlačítko „Download to device“. Jelikož ještě nebyla navázána komunikace mezi počítačem a PLC tak se zobrazí okno zobrazené na následujícím obrázku.



Obr. 3.14 – Extended download to device

Zde vyberu tlačítko „Start search“ a chvíli počkám. Pokud je vše správně zapojené, tak se po chvíli zobrazí řádek s připojeným PLC, které kurzorem vyberu a kliknu na tlačítko „Load“. Pokud je vše v pořádku, tak se zobrazí následující okno.

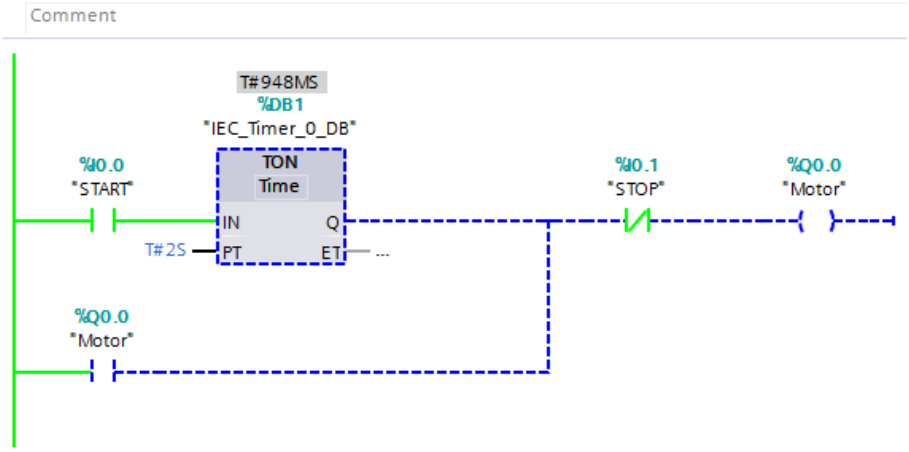


Obr. 3.15 – Load results

V tomto okně je mimo výsledků nahrání mého programu ještě možnost výběru akce, která se má vykonat po nahrání programu do zařízení. Při zaškrtnutí nabízené možnosti (zobrazené na předchozím obrázku) se nahraný program ihned spustí. Spuštěný program lze kdykoliv vypnout nebo zapnout pomocí tlačítek „Start CPU“ a „Stop CPU“. Nechám tedy program, aby se spustil. Dále stisknu tlačítko „Go online“ a tím přejdu do „online“ režimu. V TIA portal v okně „Project tree“ se zobrazí zelené značky, které informují o tom, že položka v mém projektu v počítači se zelenou značkou se shoduje s tím, co je v PLC. Například: Zelené kolečko u programového bloku značí to, že v PLC je nahraný stejný program jako je v otevřeném projektu. Pokud by něco nesouhlasilo, tak by se objevily oranžové značky. Ty se mohou objevit třeba, když uživatel upraví svůj program a nenahraje nově upravený a zkompilovaný program do PLC.

3.1.7 Test uživatelského programu

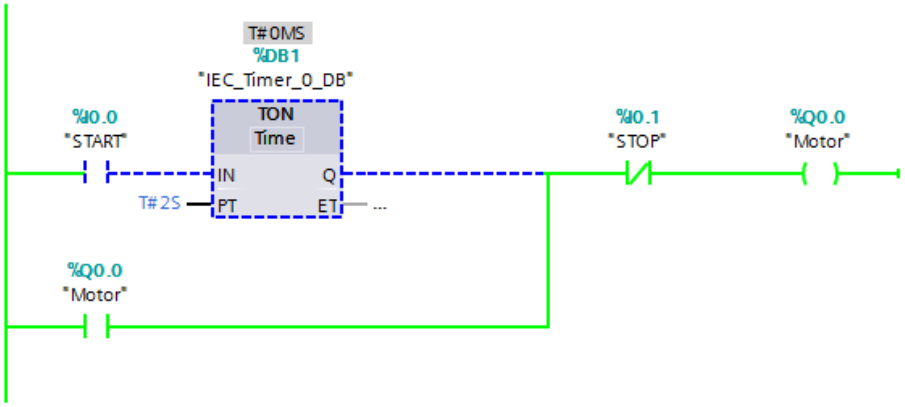
Když mám všechny značky zelené, tak je třeba otestovat program, jestli funguje správně a splňuje zadání. K tomu slouží tlačítko „Monitoring on/off“. Po jeho stisknutí se TIA portal přepne do monitorovacího režimu. Trochu se změní uživatelský program, který už není tvořen černou plnou čarou, ale je tvořen zelenou plnou a modrou přerušovanou čarou. Zelená čára značí logickou jedničku a modrá čára přerušovaná značí logickou nulu.



Obr. 3.16 – Spuštěný program 1

Na obr. 3.16 je spuštěný program. Na něm je vidět, že je sepnuté tlačítko „START“ a časovač v bloku „TON“ běží již 0,948 s. Tlačítko „STOP“ není stisknuté a tedy je rozpínací kontakt sepnutý. Na výstupu pro motor je logická nula, motor tedy není zapnutý.

Z následujícího obrázku je patrné, že na výstupu pro motor je signál, tudíž motor běží a není stisknuté žádné tlačítko. Po stisknutí tlačítka „STOP“ (přepnutí vypínače na pozici 1) se přeruší signál na výstup a rozepne se spínací kontakt „Motor“ a motor se zase zapne až při stisknutém tlačítku „START“ po dobu dvou sekund a nestisknutém tlačítku „STOP“.



Obr. 3.17 – Spuštěný program 2

3.1.8 Závěr příkladu

Po otestování programu mohu prohlásit, že jsem splnil zadání, které znělo takto: Naprogramujte ovládání motoru pomocí PLC, tak aby se motor zapnul po dvou sekundách stisknutého tlačítka START a zůstal zapnutý, dokud se nestiskne tlačítko STOP.

Na tomto jednoduchém příkladu s ovládáním motoru jsem ukázal a vysvětlil základní funkce, možnosti a postupy v programu TIA portal. Také jsem ukázal, že může existovat více správných řešení.

4 NÁVRH DPS

Ke každé úloze bude jedna deska plošného spoje se signalizací stavu. Na každé desce budou dvě, tři nebo čtyři LED diody podle toho kolik výstupů z PLC bude mít daná úloha. Diody na jedné desce budou mít společnou katodu.

Obr. 4.1 – Schéma zapojení diody

4.1 Výpočet hodnot

Výpočet teoretických hodnot s požadovaným napětím na diodě 2,5 V a proudem diodou 20 mA.

$$U_R = U_v - U_{LED} = 24 - 2{,}5 = 21{,}5\ \text{V}$$

kde U_R – napětí na odporu, V,

U_v – vstupní napětí, V,

U_{LED} – napětí na LED diodě, V.

$$R = \frac{U_R}{I_{LED}} = \frac{21{,}5}{0{,}02} = 1075\ \Omega$$

kde R – odpor před diodou, Ω,

U_R – napětí na odporu, V,

I_{LED} – proud procházející odporem a diodou, I.

$$P_R = U_R \times I_{LED} = 21{,}5 \times 0{,}02 = 0{,}43\ \text{W}$$

kde P_R – ztrátový výkon odporu R, W,

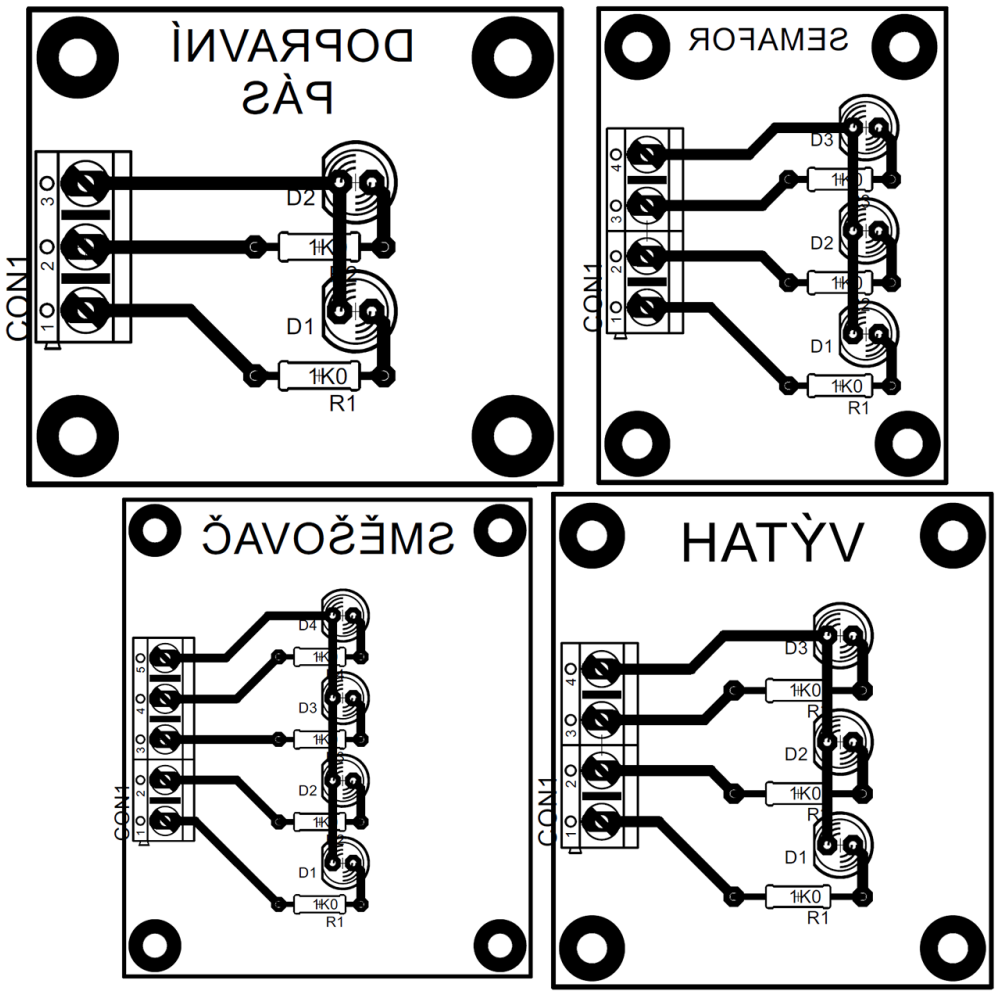
U_R – napětí na odporu, V,

I_{LED} – proud procházející odporem a diodou, I.

Při této hodnotě ztrátového výkonu je třeba zvolit odpor dimenzovaný od 0,5 W.

4.2 Eagle

DPS jsou navrženy v programu Eagle.



Obr. 4.2 – Návrhy DPS pro laboratorní úlohy

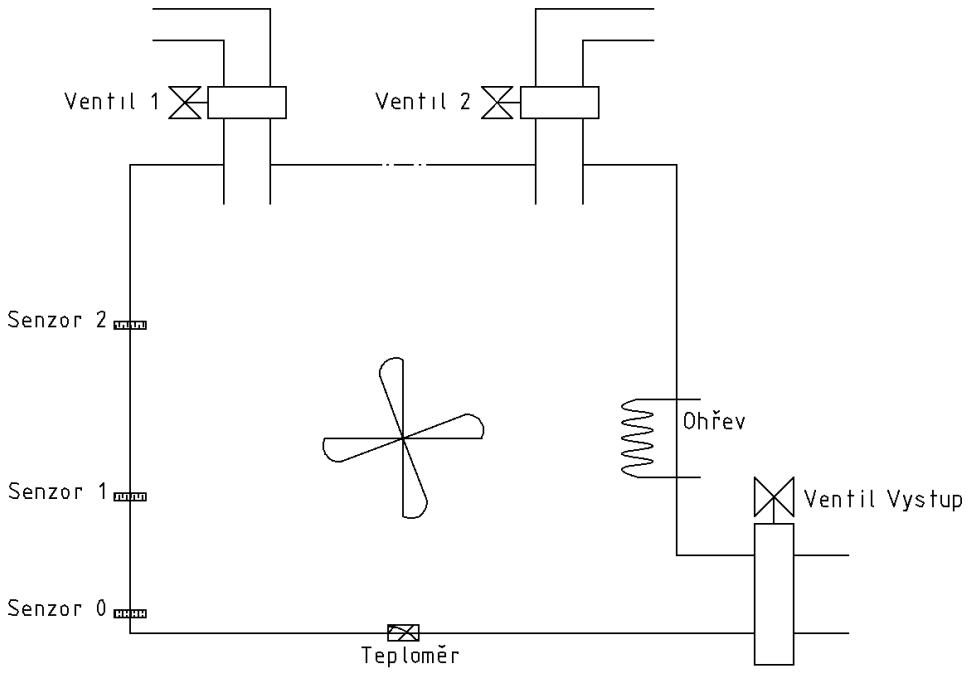
5 LABORATORNÍ ÚLOHY

V této kapitole jsou zadání pro čtyři laboratorní úlohy a také postupy pro jejich vypracování. Ke zhotovení těchto úloh je potřeba znalostí z předchozích kapitol. Úlohy mají tyto názvy:

- Směšovač
- Semafor
- Dopravní pás
- Výtah

5.1 Směšovač

Směšovač si lze představit jako veliký sud. Má v sobě tři otvory a v každém je připojená hadice s elektrickým ventilem. Dvěma hadicemi se napouští dvě různé tekutiny, které je potřeba smíchat a následně ohřát. Třetí ventil slouží k vypouštění smíchané a ohřáté tekutiny. Ve směšovači je na dně zabudovaný dvoustavový elektrický teploměr, který je nastavený na požadovanou teplotu. Když teplota stoupne nad nastavenou teplotu, tak na výstupu teploměru bude napětí (tedy logická jednička). Dále Směšovač obsahuje tři dvoustavová čidla hladiny, která dávají informaci o tom, zda čidlo je pod hladinou (logická jednička) nebo není (logická nula) a také je ve směšovači vrtule pro rozmíchání kapalin a topné těleso k jejich ohřevu.



Obr. 5.1 – Náčrt Směšovače

5.1.1 Zadání

Napište program pro PLC, který bude ovládat ventily a ohřev směšovače, tak aby při prázdném směšovači se spustil „Ventil 1“, který napustí nádrž až po „Senzor 1“ v tom momentu se uzavře „Ventil 1“ a otevře se „Ventil 2“, který napustí nádrž po „Senzor 2“ a pak se uzavře. Potom se spustí ohřev a směšování, jakmile stoupne teplota na určenou hodnotu tak se nádrž vypustí a začne se postup opakovat.

5.1.2 Příprava

Ze zadání a náčrtu plyne, že budou potřeba čtyři výstupní veličiny (pro každý ventil a ohřev se směšováním) a čtyři vstupní veličiny (pro každý senzor hladiny a teploměr). Nejprve si ze zadání a popisu směšovače vyplním pravdivostní tabulku se všemi možnostmi.

Pro jednoduchost zavedu proměnné: *S0* pro senzor 0; *S1* pro senzor 1; *S2* pro senzor 2; *t* pro teploměr; *V1* pro ventil 1; *V2* pro ventil 2; *VV* pro ventil výstupní; *OS* pro ohřev a směšování.

Tab. 5.1 – Pravdivostní tabulka pro směšovač

Vstupy				Výstupy			
S0	S1	S2	t	V1	V2	VV	OS
0	0	0	0	1			
0	0	0	1	1			
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0	1			
1	0	0	1			1	
1	0	1	0				
1	0	1	1				
1	1	0	0		1		
1	1	0	1			1	
1	1	1	0				1
1	1	1	1			1	

Každý řádek v tab 5.1 říká že: „Při této kombinaci vstupů bude právě taková kombinace výstupů“. V řádcích, ve kterých je pravá strana proškrtnuta, jsou kombinace vstupů, které za normálního provozu nemohou nastat. Pokud nastanou, tak se jedná o chybu.

Nyní si pro každou výstupní proměnnou udělám Karnaughovu mapu a minimalizuji funkce, o kterých jsem psal v kapitole 1.6. Další možností je minimalizace pomocí zákonů Booleovy algebry. Tato možnost je ovšem složitější a jednodušeji se v ní udělá chyba.

Tab. 5.2 – Mapa pro Ventil 1

t \ S2	S1 S0 = 00	01	11	10
0	1	1		
0				
1				
1	1			

Tab. 5.3 – Mapa pro Ventil 2

t \ S2	S1 S0 = 00	01	11	10
0			1	
0				
1				
1				

Tab. 5.4 – Mapa pro výstupní ventil

t \ S2	S1 S0 = 00	01	11	10
0				
0				
1			1	
1		1	1	

Tab. 5.5 – Mapa pro ohřev a směšování

t \ S2	S1 S0 = 00	01	11	10
0				
0			1	
1				
1				

Za pomocí minimalizace si z Karnaughových map vypíši následující funkce:

$$V_1 = \bar{T} \cdot \bar{S_2} \cdot \bar{S_1} + \bar{S_2} \cdot \bar{S_1} \cdot \bar{S_0}$$

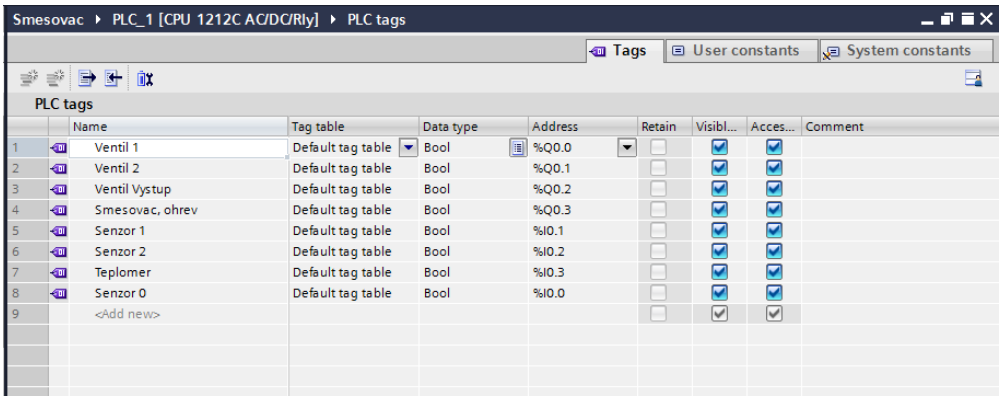
$$V_2 = S_0 \cdot S_1 \cdot \bar{S_2} \cdot \bar{T}$$

$$V_v = T \cdot \bar{S_2} \cdot S_0 + T \cdot S_1 \cdot S_0$$

$$OS = S_0 \cdot S_1 \cdot S_2 \cdot \bar{T}$$

5.1.3 TIA portal

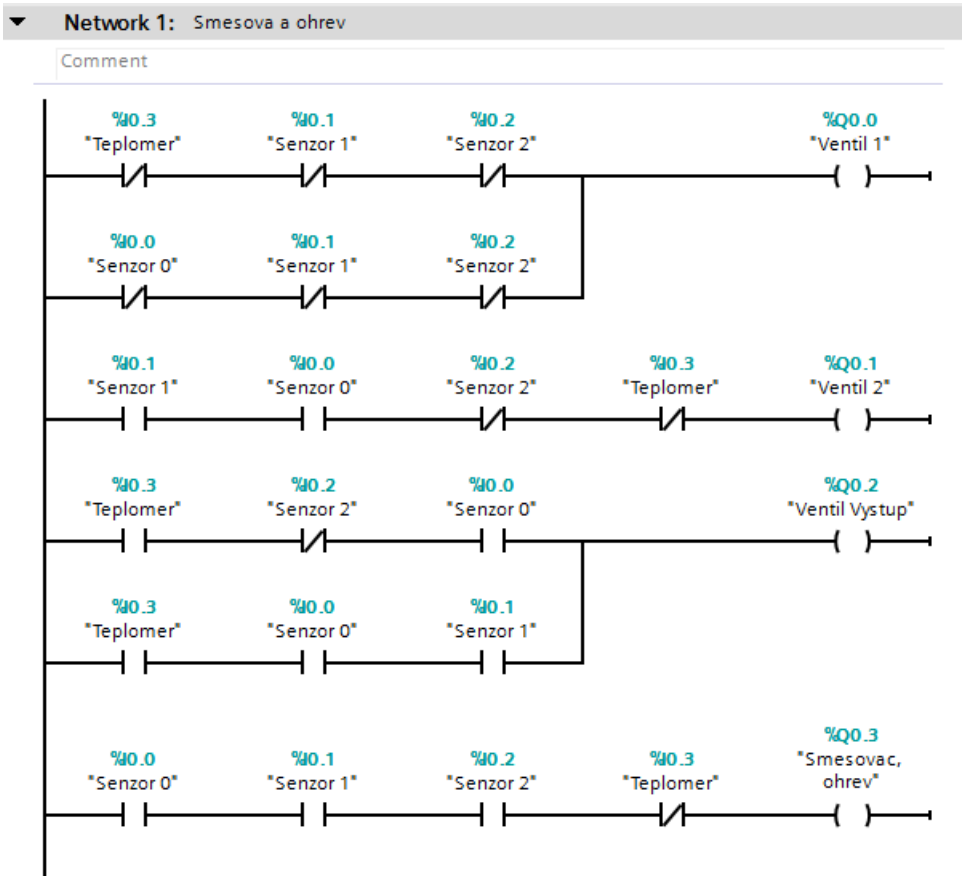
Z výše uvedených funkcí už lze přímo psát program pro PLC. V TIA portalu založím nový projekt s názvem „Směšovač“. Vyberu procesor „6ES7 212 – 1BE40 – 0XB0“ verze 4.1. Jako další krok budu definovat vstupy a výstupy na PLC do tabulky s „tagy“.



	Name	Tag table	Data type	Address	Retain	Visibl...	Acces...	Comment
1	Ventil 1	Default tag table	Bool	%Q0.0		☑	☑	
2	Ventil 2	Default tag table	Bool	%Q0.1		☑	☑	
3	Ventil Vystup	Default tag table	Bool	%Q0.2		☑	☑	
4	Smesovac, ohrev	Default tag table	Bool	%Q0.3		☑	☑	
5	Senzor 1	Default tag table	Bool	%I0.1		☑	☑	
6	Senzor 2	Default tag table	Bool	%I0.2		☑	☑	
7	Teplomer	Default tag table	Bool	%I0.3		☑	☑	
8	Senzor 0	Default tag table	Bool	%I0.0		☑	☑	
9	<Add new>					☑	☑	

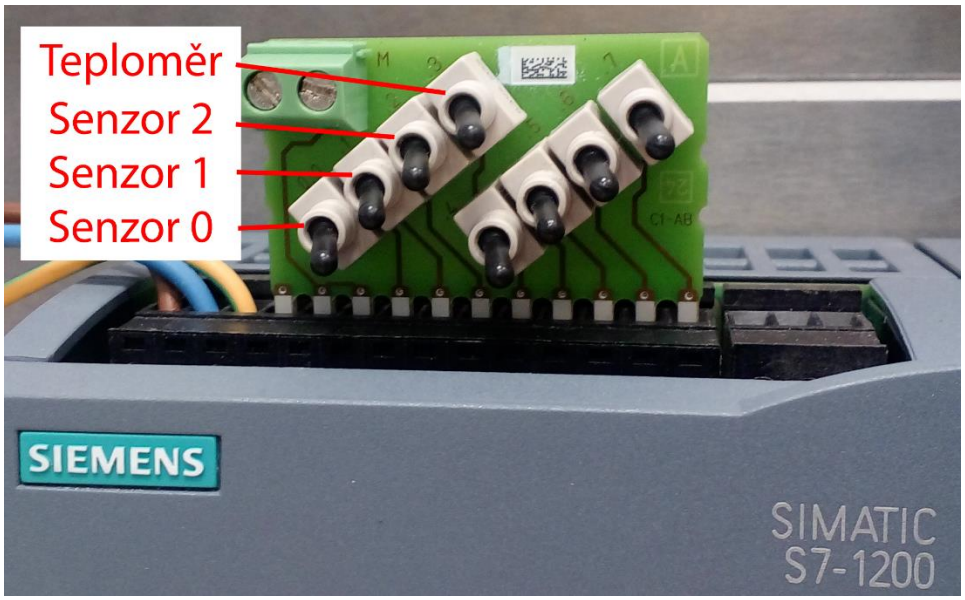
Obr. 5.2 – „Tagy“ pro Směšovač

Po vyplnění tabulky „tagů“ přejdu k psaní programu do programového bloku s jazykem LAD a funkce získané z Karnaughových map přepíšu tímto jazykem. Což bude vypadat jako na obr. 5.3.



Obr. 5.3 – Program pro ovládání směšovače

Po zhotovení programu je potřeba ho vyzkoušet. Pro testování a ověření funkčnosti programu připojím k výstupním kontaktům krabičku s diodami pro signalizaci stavu. Každá dioda symbolizuje nějaký ze tří ventilů nebo směšování s ohřevem. Pokud dioda svítí tak je příslušný ventil otevřený nebo je spuštěné směšování s ohřevem. Dále na vstupní kontakty PLC připojím destičku s vypínači. Těmito vypínači budu simulovat signál, který by šel od senzorů hladiny a teploty. Na obr 5.4 je znázorněné zastoupení signálů vypínači.



Obr. 5.4 – Simulace signálů pro směšovač

5.1.4 Závěr

Po ozkoušení různých vstupních kombinací a s předpokladem, že kapalina tekoucí skrz „Ventil 1“ velmi rychle sníží teplotu uvnitř směšovače a tím klesne teplota pod nastavenou hladinu na teploměru, tak mohu prohlásit, že jsem splnil zadání. Při skutečném zapojení směšovače se místo desek plošných spojů připojí senzory a ventily přímo ze směšovače. Při reálném zapojení je třeba zkontrolovat vstupní a výstupní napětí a proud všech připojených zařízení. Například se dá předpokládat, že signál pro zapnutí ohřevu bude veden z PLC na relé, které bude spínat větší proudy, než dokáže PLC poskytnout.

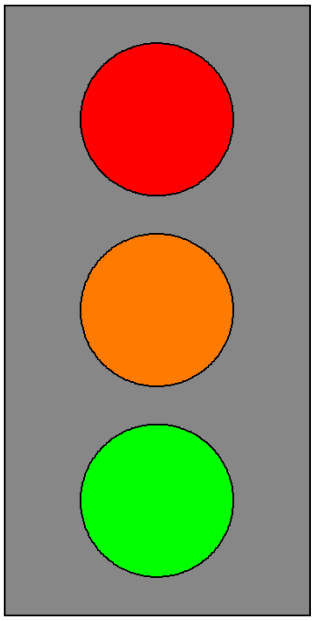
Za použití kombinační logiky tato úloha prověřuje znalosti získané v kapitole 1 Logické řízení.

Otázka k zamyšlení: Jak by se dal vylepšit tento program pro ovládání směšovače?

V tab 5.1 jsou vidět řádky, které mají škrtnutou pravou stranu. Jak bylo řečeno, tak jsou tyto vstupní kombinace chybné. Jedním z možných vylepšení je ošetření těchto kombinací. Do programu by se dopsala další výstupní proměnná, která by signalizovala, že nastala nějaká chybná kombinace. Pokud by nastala, mohla by se rozsvítit kontrolka pro ohlášení chyby a směšovač by se zastavil do doby, než bude chyba opravena. Dále by se mohl přidat vypínač, který by zapnul chod směšovače atd.

5.2 Semafor

Světelné signalizační zařízení známé jako semafor zná asi každý z nás. Tři světla, tři barvy obvykle ovládané časově. V dnešní době nezbytná a efektivní věc pro řízení provozu na větších nebo velmi využívaných pozemních komunikacích.



Obr. 5.5 – Semafor

5.2.1 Zadání

Napište program pro PLC, který bude časově ovládat jeden semafor pro automobilovou dopravu. Semafor musí signalizovat všechny čtyři pokyny pro řidiče za normálního provozu jako skutečný semafor. Časové nastavení pro každý pokyn:

- 3 s pro signalizaci „Stůj!“
- 3 s pro signalizaci „Volno“
- 1 s pro signalizaci „Připrav se k jízdě“
- 1 s pro signalizaci „Pozor!“

5.2.2 Příprava

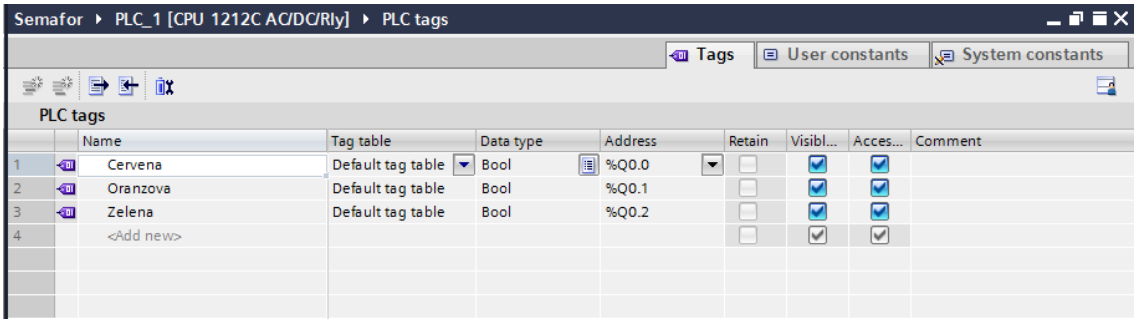
Nejdřív si rozvrhnu, jak reálně vypadá cyklus barev na semaforu:

1. červená
2. červená a oranžová
3. zelená
4. oranžová

Semafor bude ovládaný časově, tedy nebudou potřeba žádné vstupní proměnné. Výstupní proměnné budou tři. Pro každou barvu na semaforu bude jeden výstup z PLC. Dále bude potřeba vytvořit nějaký cyklus, který se použije v programu. Zvolím si vnitřní proměnnou, která mi bude určovat, která světla se mají rozsvítit.

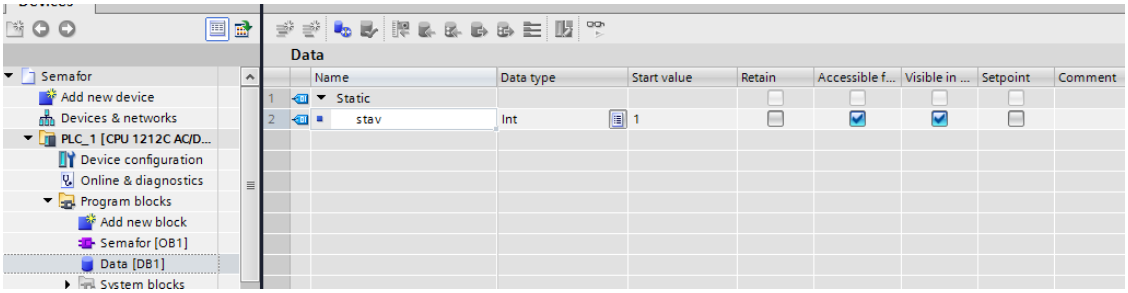
5.2.3 TIA portal

V TIA portalu založím nový projekt s názvem „Semafor“. Vyberu procesor „6ES7 212 – 1BE40 – 0XB0“ verze 4.1. Jako další krok budu definovat výstupy na PLC do tabulky s „tagy“.



Obr. 5.6 – „Tagy“ pro Semafor

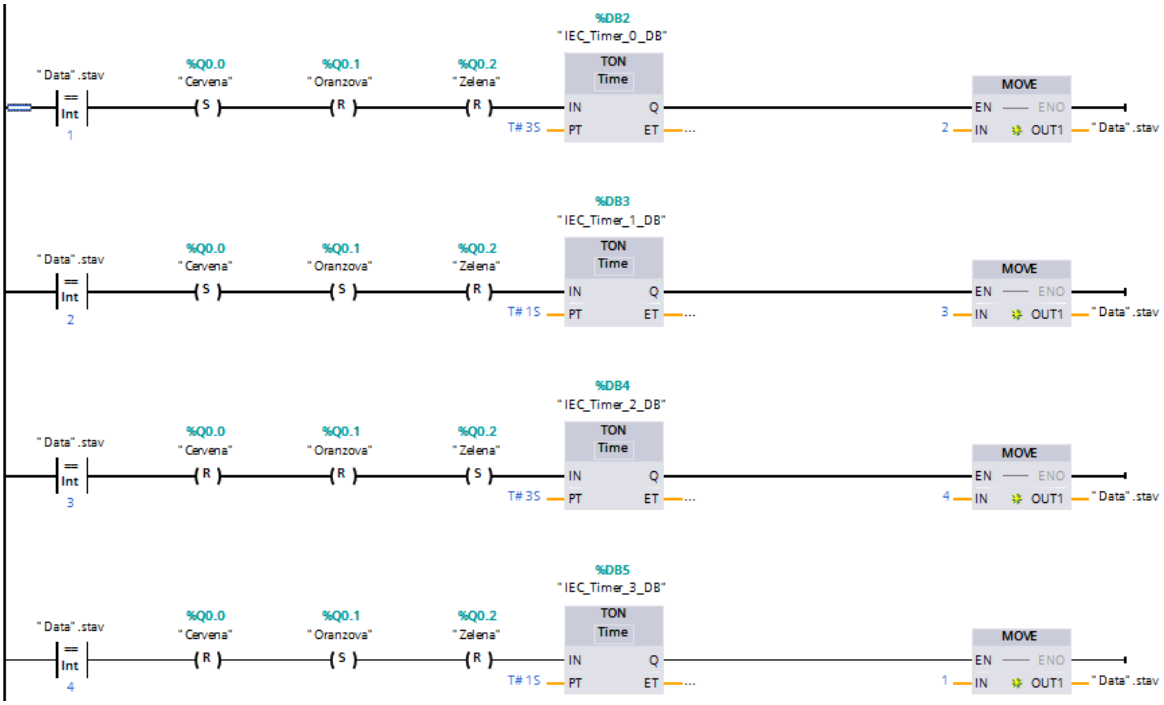
Po nastavení výstupů založím nový datový blok a do něho přidám proměnnou na určení stavu mého programu.



Obr. 5.7 – Datový blok pro semafor

Použiji datový typ „Int“, který může nabývat hodnot od −32768 až po 32767. Tento datový typ zabírá v paměti 2 B (Siemens, 2014).

Nyní přejdu k psaní programu do programového bloku s jazykem LAD. Zde si v jedné síti udělám 4 příčky a na začátek každé vložím porovnávací člen, přes který projde signál, pouze pokud se hodnota proměnné *stav* bude rovnat zadané hodnotě. Potom vložím požadované kombinace výstupů pro každý stav a za ně přidám člen, který zdrží signál zadanou dobu. Jako poslední operaci přidám „MOVE“, která přepíše proměnnou *stav*. Program bude vypadat jako na obrázku níže.



Obr. 5.8 – Program pro ovládání semaforu

Pro výstupy jsou zde použity operace s písmeny R (reset) a S (set). Pokud na tuto operaci s písmenem S přijde signál tak přiřazený výstup nastaví na logickou jedničku. Pokud na tuto operaci s písmenem R přijde signál tak přiřazený výstup nastaví na logickou nulu.

Po napsání programu je čas pro ověření zda program splňuje zadání. Připojím tedy k výstupním kontaktům PLC příslušnou desku plošného spoje a zkontroluji, zda je PLC připojené do elektrické sítě a síťovým kabelem k počítači. Pak dám zkompilovat program a nahraji do procesoru v PLC. Pro kontrolu v TIA portalu zapnu funkci „Monitoring“ a spustím program.

5.2.4 Závěr

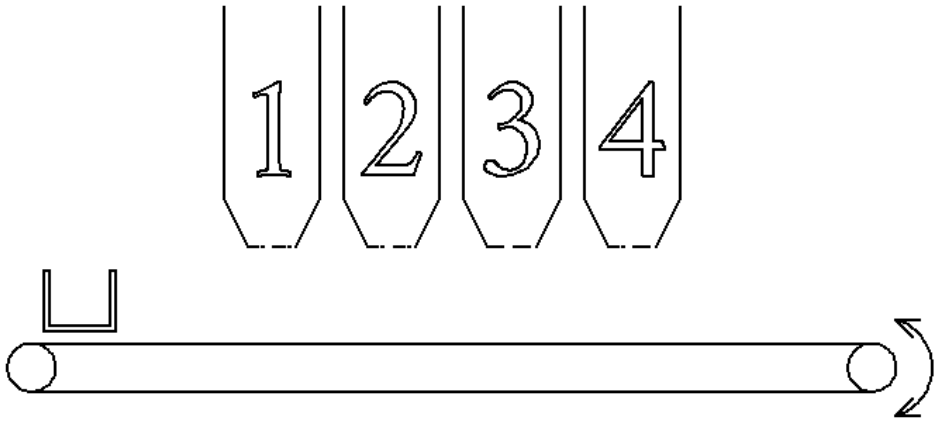
Při spuštěném programu se kombinace diod přepínají stejně jako na skutečném semaforu. Pouze časy na jednotlivých stavech jsou kratší než je v dopravě běžné, ale tyto časy odpovídají zadání. V této úloze jsou ukázány další možnosti, které se mohou využít při programování.

Otázka k zamyšlení: Jak by se dal rozšířit tento program pro ovládání semaforu?

Semafor při běžném provozu zde mám naprogramovaný, ale je pouze jeden. Rozšířit by se tedy program dal o další výstupy pro další semafor buď pro chodce, nebo pro vozidla jedoucí z jiného směru. Dále by se program dal rozšířit o přepnutí semaforu do stavu, kdy neplatí světelná signalizace a bliká jen oranžové světlo.

5.3 Dopravní pás

Důležitý dopravní prostředek pro spoustu výrobních firem. Slouží pro přepravu různých materiálů po výrobním závodě. Dopravní pás ovládaný motorem usnadňuje a urychluje práci dělníkům a operátorům ve výrobě. V tomto případě se jedná o pás, který převáží kontejner pod čtyřmi sýpkami, ze kterých se převážený kontejner naplňuje. Motor pohybující s pásem má dva vstupy. Signál do prvního vstupu roztočí motor na levou stranu a signál do druhého vstupu roztočí motor na stranu pravou. Dále pod každou sýpkou je senzor, který detekuje kontejner. Na tomto páse může být pouze jeden kontejner.



Obr. 5.9 – Náčrt dopravního pásu

5.3.1 Zadání

Napište program pro PLC, který bude ovládat dopravní pás. Na výrobní lince někdo popletl sýpky a materiály jsou ve špatných sýpkách. Je třeba, aby do kontejneru byly nasypány materiály ze všech čtyř sýpek, ale ve správném pořadí. Napište tedy program, který dopraví kontejner pod každou sýpku v následujícím pořadí:

1. Sýpka 3
2. Sýpka 1
3. Sýpka 2
4. Sýpka 4

Další požadavek je, že kontejner musí pod každou sýpkou (ze které se má sypat materiál do kontejneru) být tři sekundy. Nový (prázdný) kontejner přijíždí z levé strany a plný musí odjed z pásu doprava. Po odjetí plného kontejneru se bude cyklus opakovat.

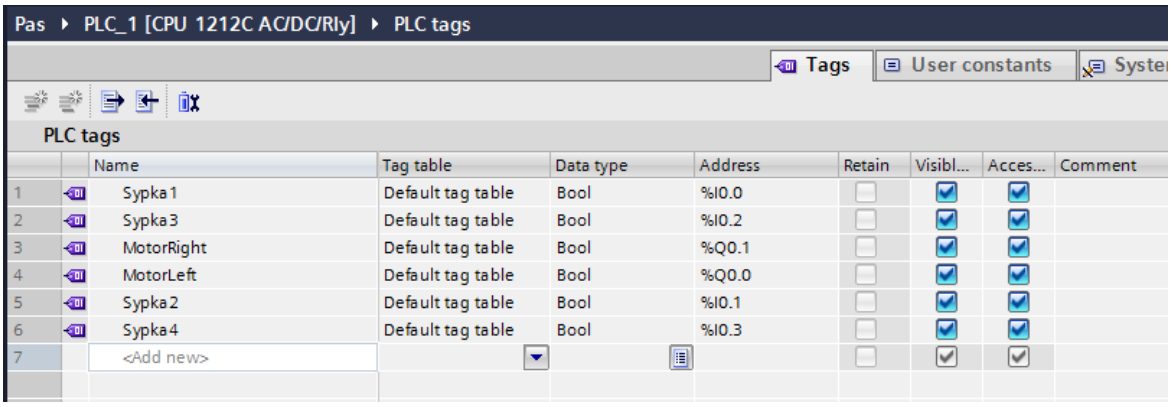
5.3.2 Příprava

Z popisu a zadání je jasné, že budu potřebovat čtyři vstupy (jeden vstup pro senzor pod každou sýpkou) a dva výstupy pro motor (pro pohyb pásu doleva a doprava). Pokud nebude žádný signál na motor, tak předpokládám, že se pás nehýbe.

V programu bude proměnná *stav*, která mi bude určovat, v jaké fázi se program nachází. Podle toho v jaké fázi program bude, tak bude nastaven výstup pro motor a čekání na sepnutí senzoru pod sýpkou.

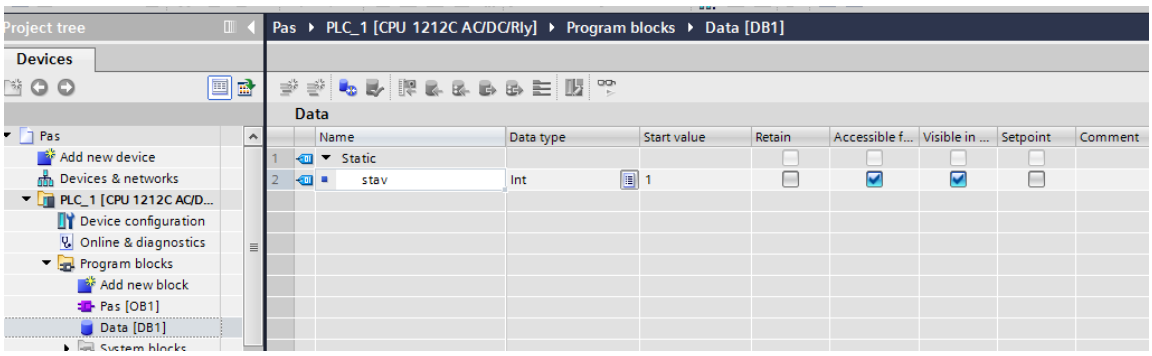
5.3.3 TIA portal

V TIA portalu založím nový projekt s názvem „Pás“. Vyberu procesor „6ES7 212 – 1BE40 – 0XB0“ verze 4.1. Jako další krok budu definovat vstupy a výstupy na PLC do tabulky s „tagy“.



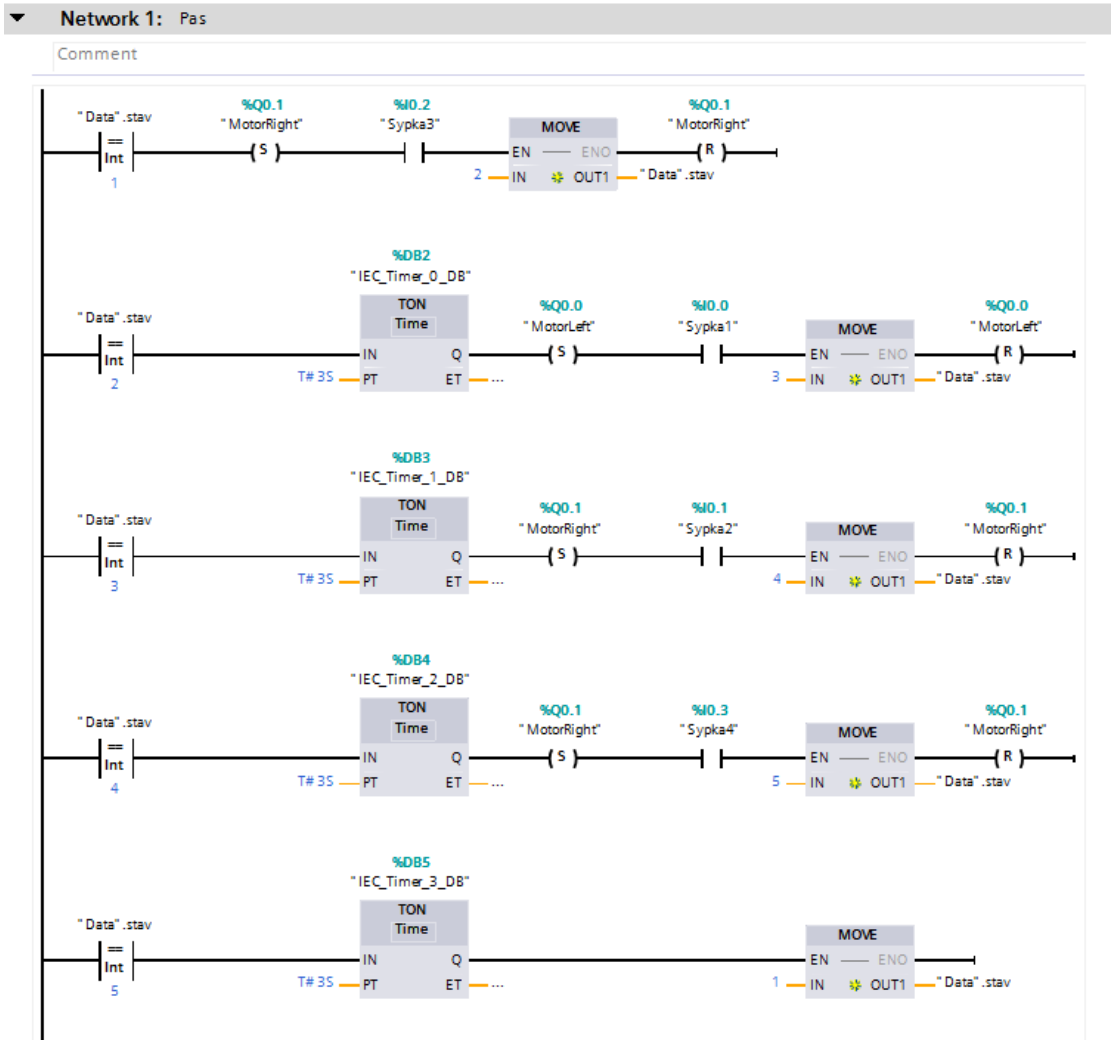
Obr. 5.10 – „Tagy“ pro dopravní pás

Po nastavení vstupů a výstupů založím datový blok a do něho vložím proměnnou k určování stavu programu.



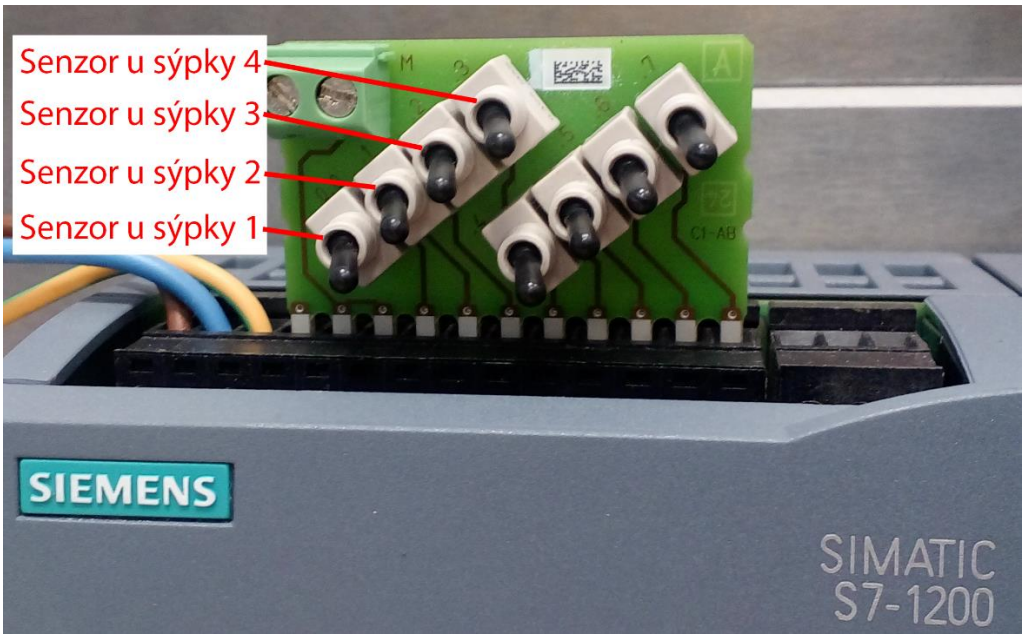
Obr. 5.11 – Datový blok pro dopravní pás

Pro vytvoření programu použiji opět jazyk LAD. V programu udělám pět příček, každá bude pro jeden stav, takže na začátek každé vložím operaci pro porovnání proměnné *stav*. Za porovnávací operace vložím nastavení výstupu pro motor a zdržení signálu, které nastavím na tři sekundy podle zadání. Za nastavení výstupů pro motor vložím spínací operaci, které přiřadím vstup do PLC, který bude odpovídat senzoru pod určitou sýpkou. Dále vložím operaci „MOVE“, která mi přepíše proměnnou *stav* a tedy změní stav programu. Po přepsání proměnné ještě resetuji vstup na motor, aby se motor zastavil.



Obr. 5.12 – Program pro ovládání dopravního pásu

Po napsaném programu je třeba ho otestovat. Zkontroluji připojení PLC do elektrické sítě a připojení síťovým kabelem k počítači. Dále připojím příslušnou desku plošného spoje se signalizací stavu na příslušné výstupní kontakty PLC a desku plošného spoje s vypínači na vstupní kontakty PLC. Zkompiluji program, nahraji zkompilovaný program do procesoru v PLC spustím „Monitoring“ a otestuji program, zda splňuje zadání.



Obr. 5.13 – Simulace signálů pro dopravní pás

5.3.4 Závěr

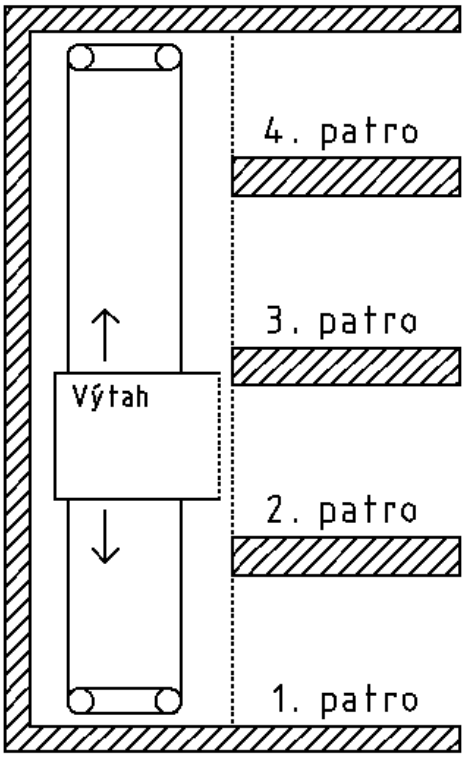
Po spuštění programu jede pás doprava a program čeká na signál od senzoru pod třetí sýpkou. Při signálu od senzoru třetí sýpky program změní stav, tři vteřiny čeká a pak pás jede na levou stranu a čeká na signál od senzoru u první sýpky. Podobně program pokračuje ke druhé a čtvrté sýpce, když dojede ke čtvrté tak program vyčká tři sekundy a pás jede doprava a čeká na nový kontejner, který má dojet k sýpce číslo tři. Tento postup a tedy i program splňuje zadání.

Otázka k zamyšlení: Co může nastat za problémy, se kterými program nepočítá?

Při psaní programu se předpokládalo, že na pásu může být pouze jeden kontejner, ale pokud by na pásu byly dva a více kontejnerů tak by mohl nastat problém. Stejně tak by mohl nastat problém, když se porouchá nějaký senzor. Také může nastat situace, kdy bude potřeba akutně zastavit pohyb pásu a to by vyžadovalo další vstup, který by rozpojil signál do motoru. Pro použití ve skutečném výrobním závodě by bylo potřeba rozšířit program o bezpečnostní prvky.

5.4 Výtah

Zařízení usnadňující práci a přepravu všem lidem. Tažená nebo tlačená plošina je dnes již nepostradatelné zařízení v téměř každé vícepatrové budově. Pro chod výtahu jsou v každém patře senzory pro určení polohy výtahu a také tlačítka pro přivolání výtahu. Na každé patro připadá jeden senzor a jedno tlačítko. Aby si cestující v kabině výtahu mohl zvolit, kam chce jet, tak je v kabině tlačítko pro každé patro.



Obr. 5.14 – Náčrt výtahu

5.4.1 Zadání

Napište program pro PLC, který bude ovládat výtah ve čtyřpatrové budově. Výtah musí být schopen přejet z kteréhokoli patra do jiného nejkratší možnou cestou. K motoru výtahu musí být přivedeny tři signály a ty jsou:

- Má-li výtah jet nahoru
- Má-li výtah jet dolů
- Má-li výtah stát

5.4.2 Příprava

Z popisu výtahu a zadání si mohu vypsat, že budu potřebovat tři výstupy pro motor, čtyři vstupy pro senzory polohy, jeden vstup pro každé tlačítko z každého patra (celkem čtyři) a další čtyři tlačítka umístěná v kabině výtahu. To dává dohromady tři výstupy a dvanáct vstupů. Takový počet vstupních kontaktů ovšem není k dispozici na použitém PLC. To vyřeším tak, že spojím signál z kabiny pro přivolání výtahu do určitého patra se signálem z tlačítka na přivolání výtahu v určitém patře. Tedy zmáčknutí tlačítka „1“ v kabině výtahu vyvolá v programu stejnou reakci jako zmáčknutí tlačítka pro přivolání výtahu v prvním patře. Zavedením tohoto předpokladu zredukuji potřebný počet vstupů na osm. Tento počet vstupů už je možný připojit k PLC.

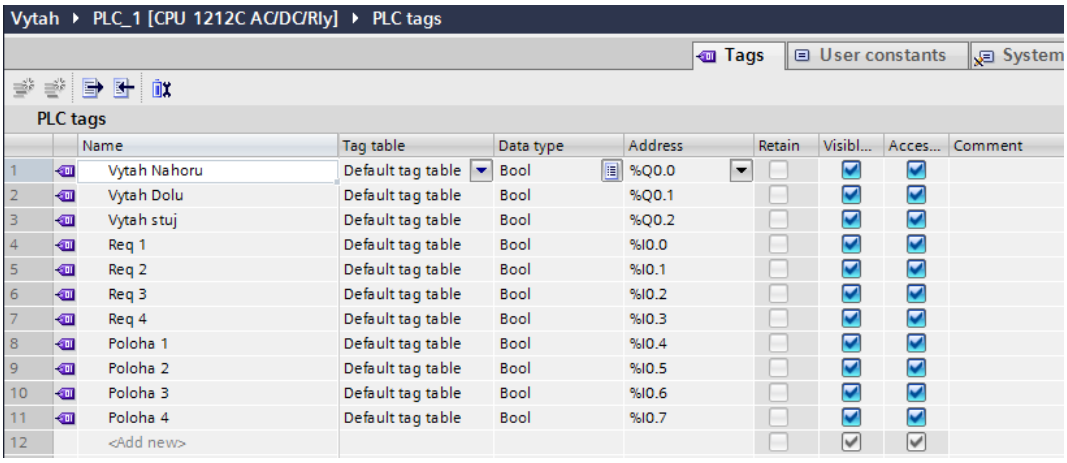
Program budu mít rozložen na tři části:

- Určení polohy výtahu
- Určení požadavku na patro
- Určení pohybu výtahu

Pro každou tuto část použiji jednu samostatnou síť. Dále budu potřebovat dvě proměnné, které se budou měnit podle polohy výtahu a přivolání výtahu na nějaké patro. Ty pak mezi sebou budu porovnávat a podle polohy výtahu a požadavku na výtah v nějakém patře se bude motor točit doprava, doleva nebo bude stát.

5.4.3 TIA portal

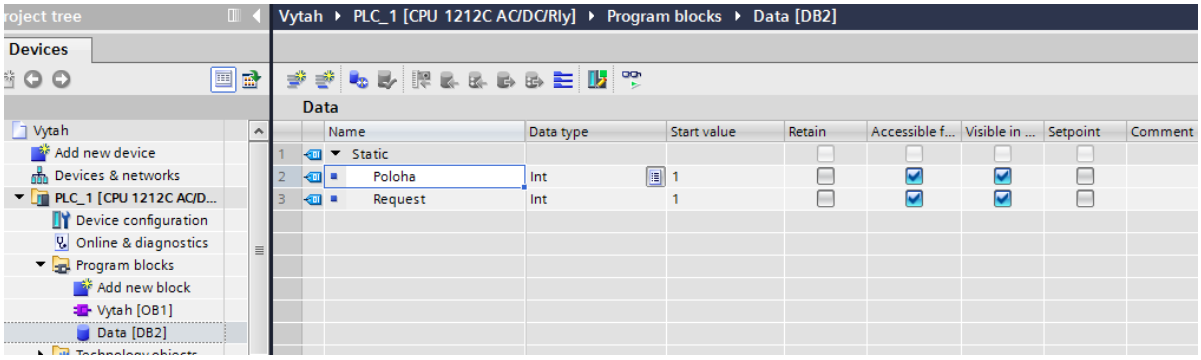
V TIA portalu založím nový projekt s názvem „Výtah“. Vyberu procesor „6ES7 212 – 1BE40 – 0XB0“ verze 4.1. Jako další krok budu definovat vstupy a výstupy na PLC do tabulky s „tagy“.



	Name	Tag table	Data type	Address	Retain	Visibl...	Acces...	Comment
1	Vytah Nahoru	Default tag table	Bool	%Q0.0		☑	☑	
2	Vytah Dolu	Default tag table	Bool	%Q0.1		☑	☑	
3	Vytah stuj	Default tag table	Bool	%Q0.2		☑	☑	
4	Req 1	Default tag table	Bool	%I0.0		☑	☑	
5	Req 2	Default tag table	Bool	%I0.1		☑	☑	
6	Req 3	Default tag table	Bool	%I0.2		☑	☑	
7	Req 4	Default tag table	Bool	%I0.3		☑	☑	
8	Poloha 1	Default tag table	Bool	%I0.4		☑	☑	
9	Poloha 2	Default tag table	Bool	%I0.5		☑	☑	
10	Poloha 3	Default tag table	Bool	%I0.6		☑	☑	
11	Poloha 4	Default tag table	Bool	%I0.7		☑	☑	
12	<Add new>					☑	☑	

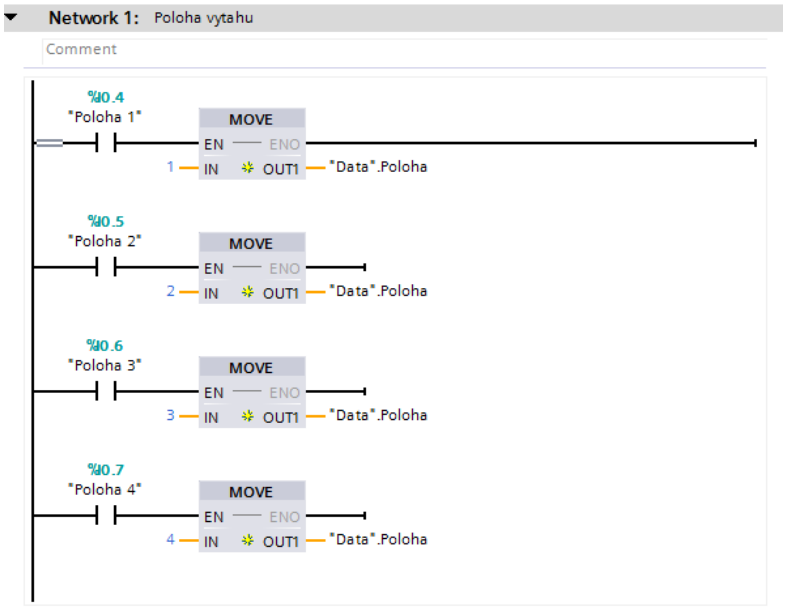
Obr. 5.15 – „Tagy“ pro ovládání výtahu

Označení „Req“ v tabulce s „tagy“ znamená „request“ tedy požadavek výtahu na nějaké patro. Na tyto vstupy se přivede signál z tlačítek v kabině a z tlačítek v každém patře. Po vytvoření „tagů“ a přidělení adres potřebuji vytvořit datový blok a do něj přidat dvě proměnné.



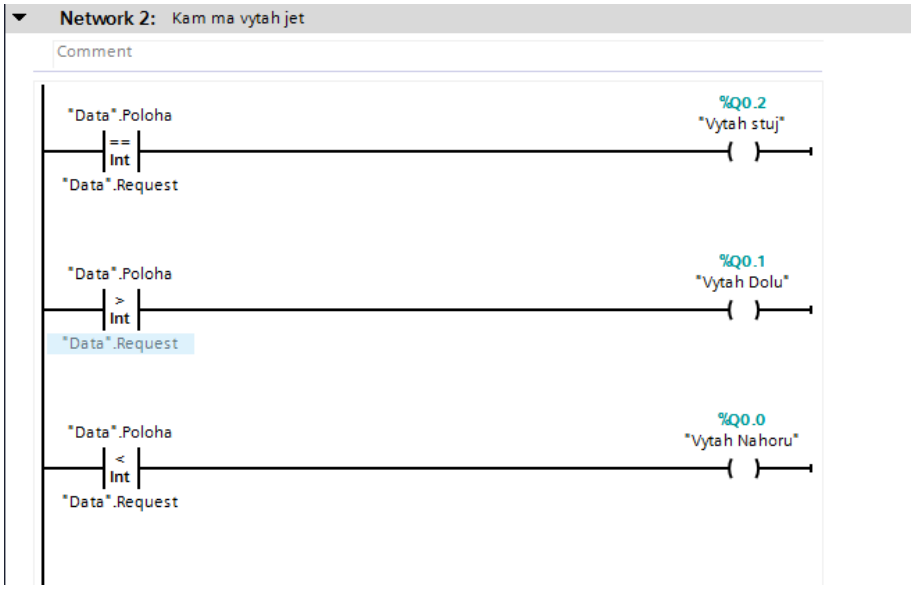
Obr. 5.16 – Datový blok pro výtah

Jak jsem již psal v přípravě, tak rozdělím program na tři části. V první části se bude přepisovat proměnná *Poloha*. Tedy udělám čtyři příčky (pro každý senzor polohy jednu) a na každou příčku vložím operace, které přepíšou proměnnou *Poloha* podle toho, který senzor polohy vyšle signál.



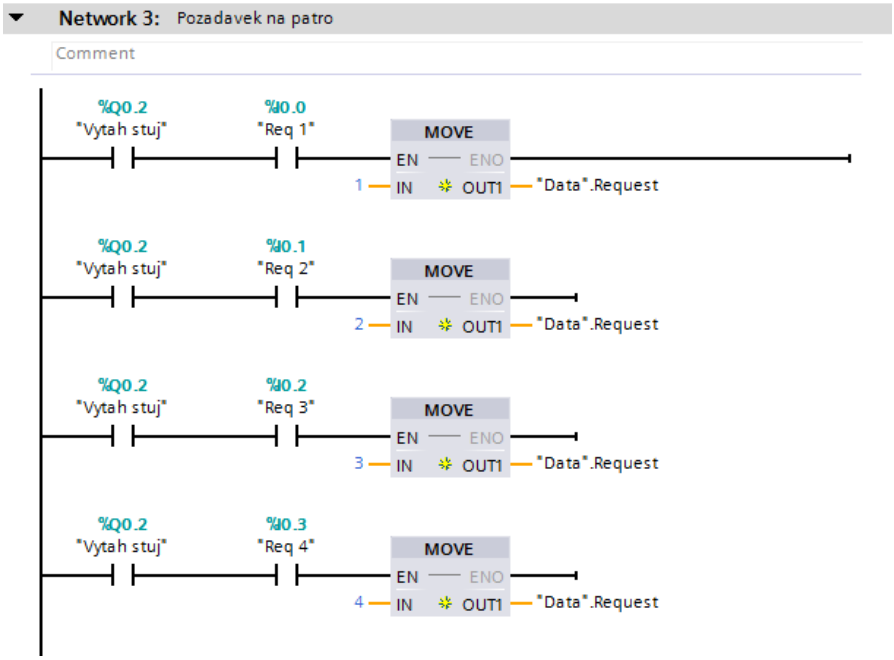
Obr. 5.17 – Nastavení polohy výtahu

Druhou část programu začnu psát ve druhé síti. Zde přidám porovnávací operace, které určí pohyb výtahu a pustí logickou jedničku na výstupní kontakty PLC.



Obr. 5.18 – Určení pohybu výtahu

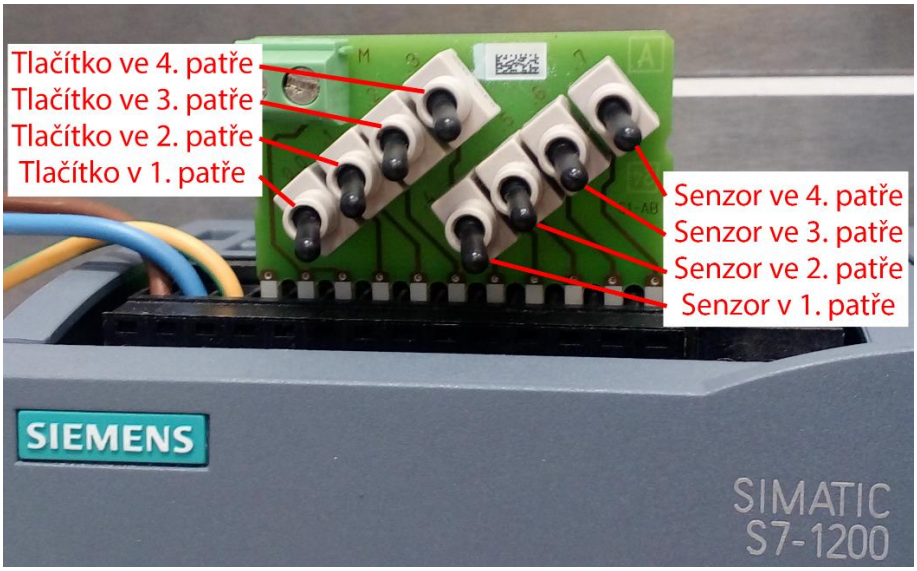
Třetí a také poslední část napíši do třetí sítě, ve které se přepíše proměnná *Request* podle toho, které tlačítko se zrovna stiskne. Téměř stejně jako v první části programu. S tím rozdílem, že ještě přidám operaci, která zabrání změně požadavku, když se výtah pohybuje.



Obr. 5.19 – Nastavení požadavku výtahu

To zaručí, že výtah dojede na požadované místo a až potom co výtah bude stát, tak se může poslat další požadavek na výtah.

Po zhotovení programu připojím příslušnou desku plošného spoje na výstupní kontakty PLC a desku plošného spoje s vypínači připojím na vstupní kontakty. Přepínám vypínačů, budu simulovat signály, které by při reálném zapojení šly od senzorů polohy a také od tlačítek pro přivolání výtahu do určitého patra. Dále připojím PLC do elektrické sítě a k počítači. Napsaný program pro výtah zkompiluji a nahraji do procesoru v PLC a pomocí funkce „Monitoring“ a připojenými vypínači zkontroluji funkčnost programu.



Obr. 5.20 – Simulace signálů pro výtah

5.4.4 Závěr

Po otestování programu pro výtah mohu říct, že s předpokladem spojení signálů na požadavek výtahu do určitého patra (popsáno v kapitole 4.4.2) program splňuje požadavky a zadání. Tato úloha má ukázat práci s více proměnnými a možnost využití více sítí pro přehlednost programu.

Otázka k zamyšlení: Co programu chybí, aby mohl být použit v reálném prostředí?

Zde mohou být dvě možnosti. Přidat další elektroniku, která by ovládala pouze funkce kabiny a nějak komunikovala s touto elektronikou. Nebo rozšířit HW o další vstupy a výstupy a rozšířit program o funkce pro ovládání kabiny. Tyto funkce by se týkaly třeba: ovládání dveří, bezpečnostních senzorů, váhových senzorů, zavolání technika při zaseknutém výtahu, atd.

6 ZÁVĚR

V mojí práci jsem shrnul informace o základech logického řízení, které jsou potřeba k pochopení programování PLC. Dále jsem v práci popsal postupy potřebné ke zhotovení a testování uživatelského programu v prostředí Step7. V tomto prostředí pak jsou řešené čtyři úlohy.

Řešené laboratorní úlohy v této práci procvičují a trénují logické uvažování a programování. Tyto úlohy nejsou složité, mohou být napsány více způsoby a dávají prostor pro zlepšení a rozšíření, o kterých píši v závěru u každé úlohy. Každý programátor po napsání programu by nejen měl otestovat svůj program, zda splňuje požadavky a zadání, ale také by se měl zamyslet nad různými situacemi, které mohou nastat. Bude program bezpečně fungovat i v případě, že se porouchá nějaký senzor? Program by měl správně reagovat na všechny možnosti, které mohou nastat. Převážně se jedná o bezpečnostní rizika.

Díky této práci je možné navrhnout a realizovat základní řídicí programy pro PLC. Programovatelný logický automat je velmi efektivní nástroj používaný pro automatizaci nejrůznějších procesů. PLC jsou velmi rozšířené, ale ve škole jsem se s nimi nesetkal. To jsou důvody, proč jsem si vybral právě toto téma své bakalářské práce. Tato práce může poskytnout materiál pro výuku programování PLC.

Literatura

Maurice Karnaugh. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017 [cit. 2017-04-25]. Dostupné z: https://en.wikipedia.org/wiki/Maurice_Karnaugh

MARČAN, Peter. Programovanie PLC: Programovacie jazyky PLC. Daily automation [online]. Kysucké Nové Mesto: Daily Automation, 2018 [cit. 2018-04-18]. Dostupné z: http://dailyautomation.sk/01-programovacie-jazyky-plc/

Siemens, s.r.o. *Simatic S7-1200* [online]. Praha: © Siemens, 2012 [cit. 2017-05-04]. Dostupné z: http://obchod.axima.cz/files/download-attachment/eshop/znacky/siemens/brochure-s7-1200-2012-cz.pdf/

Siemens. *STEP 7 Basic V13 SP1: System manual*. Norimberk, 2014. Dostupné také z: http://www1.siemens.cz/ad/current/content/data_files/automatizacni_systemy/mikrosystemy/simatic_s71200/manualy/gsg_step7-basic-v10-5_2014-12_en.pdf

ŠVARC, Ivan. *Základy automatizace – Učební texty pro kombinovanou formu bakalářského studia* [online]. [cit. 2017-4-25]. Dostupné z: http://autnt.fme.vutbr.cz/svarc/ZakladyAutomatizace.pdf

Totally Integrated Automation Portal. *Siemens* [online]. Praha: Siemens, 2017 [cit. 2017-08-08]. Dostupné z: http://stest1.etnetera.cz/ad/current/index.php?vw=0&ctxnh=2416f2e791&ctxp=home&cpa

Trochu historie: Programovatelné automaty. *TECO advanced automation: Průmyslová automatizace, Inteligentní budovy, Smart Grid* [online]. Kolín: Teco, 2009 [cit. 2017-08-07]. Dostupné z: http://www.tecomat.com/clanek_673_trochu-historie.html

Přílohy

A – CD

Příloha k bakalářské práci

SET LABORATORNÍCH ÚLOH PRO LOGICKÉ ŘÍZENÍ

Jan Hatla

CD

Obsah