ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra mikroelektroniky

Bakalářská práce

Programovatelný logický kontrolér (PLC) a jeho zapojení do automatizovaného procesu

Studijní obor: Elektrotechnika, elektronika a komunikační technika

Autor: Pavel Gregor

Vedoucí práce: prof. Ing. Miroslav Husák, CSc.

Praha 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Gregor** Jméno: **Pavel** Osobní číslo: **440260**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektrotechnika, elektronika a komunikační technika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Programovatelný logický kontrolér (PLC) a jeho zapojení do automatizovaného procesu

Název bakalářské práce anglicky:

Programmable Logic Controller (PLC) and Its Integration Into an Automated Process

Pokyny pro vypracování:

1. Poveďte analýzu současného stavu Programovatelného logického kontroléru a jeho využití pro automatizované procesy, vysvětlete stručně princip činnosti PLC.
2. Navrhněte demonstrační program PLC pro řízení modelu továrny. Model ovládejte několika funkčními tlačítky. Programové bloky budou obsahovat funkce a data, popište instrukce a strukturu programu.
3. Zhodnoťte dosažené výsledky a navrhněte možné úpravy pro jejich zlepšení.

Seznam doporučené literatury:

1. Stephen Philip Tubbs: Programmable Logic Controller (PLC) Tutorial, naklad. Stephen P. Tubbs, 2007
2. Anniss, Matthew: Understanding Programming and Logic, Megaknihy
3. Petruzella: Programmable Logic Controllers, Mcgraw Hill Higher Education, 2016, ISBN13 (EAN): 9780073373843

Jméno a pracoviště vedoucí(ho) bakalářské práce:

prof. Ing. Miroslav Husák, CSc., katedra mikroelektroniky FEL

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

Datum zadání bakalářské práce: **18.09.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **19.02.2022**

prof. Ing. Miroslav Husák, CSc. podpis vedoucí(ho) práce	prof. Ing. Pavel Hazdra, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

21.5.2021
Datum převzetí zadání

Podpis studenta

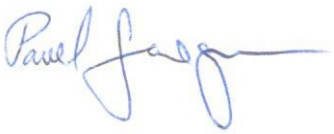
Poděkování

Chtěl bych poděkovat panu prof. Ing. Miroslavovi Husákovi, CSc, za motivaci a odborné vedení při tvorbě závěrečné práce. Dále děkuji svému zaměstnavateli AGC Automotive Czech za zapůjčení potřebného HW a SW. Také děkuji své manželce Oldřišce za psychickou podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 21. 5. 2021



.....

podpis

Anotace

Bakalářská práce seznamuje s programovatelnými logickými kontroléry, jejich využití a výhody. Stručným výčtem instrukcí a vyřešením několika jednoduchých úloh jsou vysvětleny základy programování převážně v jazyku žebříkových schémat. Hlavním tématem je popis postupu zprovoznění modelu továrny, na kterém by si mohl kdokoli zkusit základy programování, a být tak schopen následně vytvořit další projekty.

Klíčová slova: PLC, žebříčkové schema, model továrny

Annotation

The Bachelor thesis introduces programmable logic controllers, their use and benefits. A brief list of instructions and the solution of a few simple tasks explains the basics of programming, mainly in the language of ladder diagrams. The main topic is a description of the process of making the factory model work, on which anyone could try the basics of programming and then be able to create other projects.

Key words: PLC, ladder diagram, factory model

Obsah

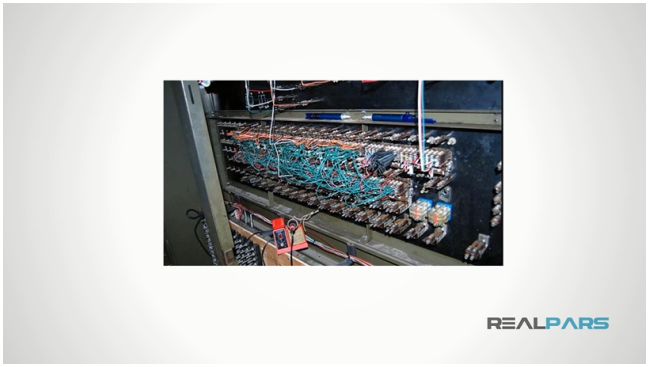
1. Úvod

1.1 Co je PLC?

Zkratka PLC znamená „programovatelný logický kontrolér“ z angl. *Programmable Logic Controler*. V češtině se můžeme setkat také s označením „programovatelný automat“ popř. „programovatelný logický automat“ (zkratky PA, PLA). Jedná se o zařízení, které se dnes běžně používá především ve výrobních procesech. Jeho hlavními úkoly je monitorování stavů jednotlivých zařízení (vstupů do PLC), jejich řízení (výstupů z PLC) na základě vnitřní logiky a zajistit jejich bezpečné fungování.

Úloha č. 1 – Od relé k PLC

Představme si několik skříní rozvaděče, který obsahuje převážně elektromechanická relé, časovače a čítače. Řekněme že těchto několik stovek elektromechanických prvků řídí pouze logiku konkrétního výrobního procesu. Vzhledem ke cvakání mechanických prvků se můžeme setkat s označením „klik klak logika“. Tyto prvky jsou propojeny vodiči, které samy o sobě jsou součástí logiky. Uvědomme si kolik vodičů potřebujeme připojit k jednomu relé. Dva vodiče pro cívku a dva vodiče ke každému dostupnému a využívanému kontaktu (minimálně jeden spínací a jeden rozpínací). Dostáváme se k několika tisícům vodičů.



Obrázek 1: Reléová logika v rozvaděči, převzato z [1]

Jedním z cílů automatizace je zjednodušení, a tak se dostáváme k programovatelnému relé. Tento prvek se skládá z relé, časovače a čítače. Navíc obsahuje i programovou paměť pro několik řádků jednoduchého programu (blokové schéma, žebříčkový diagram) – k programování se dostaneme později. Program pro toto relé může být napsán přímo pomocí displeje a tlačítek na něm umístěných nebo v počítači a poté nahrán. Displej s tlačítky představuje snadnější obsluhu.
Zjednodušeně řečeno bychom počet prvků v našem smyšleném rozvaděči jejich vhodným výběrem snížili na polovinu až třetinu. Dále by se snížil i počet nutných vodičů.

Obrázek 2: Programovatelné relé Zelio Logic SR2/SR3 výrobce Schneider Electric, převzato z [2]

Pojďme nyní prvky v tomto rozvaděči nahradit PLC. Zatím předpokládejme pouze jednoduché digitální a analogové vstupy a výstupy. Na jedné DIN liště se bude nacházet CPU jednotka s několika vstupy a výstupy a k ní připojené vstupní a výstupní moduly. Obsahově se může jednat až o stovky vstupů a výstupů. V tomto jednom PLC se bude nacházet také program znázorňující celou logiku

výrobního procesu. Můžeme tvrdit, že celý náš původně vymyšlený rozvaděč se nyní nachází v jednom menším rozvaděči spolu s dalšími řídícími prvky, které už neobstarávají logiku.

Obrázek 3: PLC Simatic řady S7-300 výrobce Siemens, převzato z [3]

1.2 Výhody PLC

Vznik PLC můžeme považovat za přirozený vývoj automatizační techniky. Díky němu zvládneme jednodušeji řídit složitější procesy. Řekněme, že jádro použití PLC je právě logika řízení. Jedná se o software, nabízí se tedy otázka, zda není vhodnější použít PC vzhledem k široké softwarové kompatibilitě. Z pohledu procesoru a datové a programové paměti je PLC o poznání horší.

Standard CPUs	CPU 1511-1 PN	CPU 1513-1 PN	CPU 1515-2 PN	CPU 1516-3 PN/DP	CPU 1517-3 PN/DP	CPU 1518-4 PN/DP
Use Case	Small to medium applications	Medium size applications	Medium to high end applications	Advanced applications and additional communication tasks	Demanding applications and additional communication tasks	High-performance applications and shortest possible response times
Program/Data memory	150 KB / 1 MB	300 KB / 1,5 MB	500 KB / 3 MB	1 MB / 5 MB	2 MB / 8 MB	4 MB / 20 MB

Obrázek 4: Porovnání CPU řady 1500 výrobce Siemens, převzato z [4]

Hlavní výhodou PLC je stabilita systému. PC je náchylné k různým pádům systému, PLC není zatíženo velkým počtem souběžných procesů. Po hardwarové stránce je PLC robustní a vhodné do prostředí s těžkými provozními podmínkami, odolné vůči přepětí a jiným nežádoucím jevům.
Ve většině výrobních procesů jde především o bezpečnost. Díky výše popsaným výhodám je použití PLC místo PC nesporně lepším rozhodnutím, i když má PC v automatizačních procesech své nezastupitelné místo (programování, design, vývoj, SCADA systémy atd.).

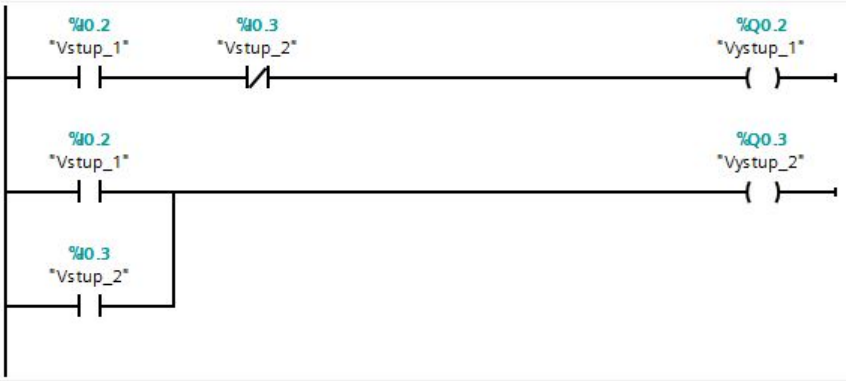
Obrázek 5: Screenshot PC programu PM-MAINT - správce řízení údržby (Siemens), převzato z [5]

Program PLC se jednoduše píše a je čitelný nejen programátorem. Porovnejme tuto vlastnost například s mikrokontroléry. V dalších kapitolách se budeme programování PLC věnovat obšírněji a zjistíme, že není nutná velká znalost programovacích jazyků.

2. Programovací jazyky PLC

2.1 Žebříčkové schema

Reléová logika byla zakreslena liniovým schematem, a tak byl vymyšlen programovací jazyk pro PLC, který z těchto schemat vychází. Jedná se o žebříčkové schema (LD – z angl. *Ladder diagram*). Tento jazyk je tedy univerzální a čitelný i obsluhou, např. elektrikáři. Všechny instrukce a výpisy převzaty ze SW [1].

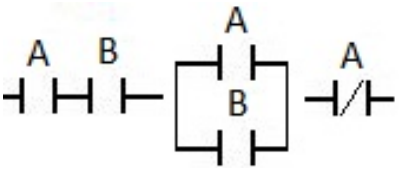


Obrázek 6: Příklad LD

Ve většině případů se na řádek píšou vlevo podmínky a vpravo jejich vyhodnocení. Program je vykonáván v nekonečném počtu cyklů, zleva doprava a řádek po řádku. Jeden cyklus trvá přibližně 10 ms. Během jednoho cyklu PLC přečte všechny vstupy, vyhodnotí napsaný software a nakonec aktualizuje všechny výstupy.

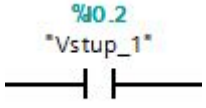
2.1.1 Instrukce bitové logiky

Booleova logika
Nejpoužívanější datový typ používaný pro programování PLC je obecně známý bool. Jde o bit a může nabývat pouze dvou hodnot – log. 1 nebo log. 0, resp. true nebo false. Na řádcích ať už liniového schematu nebo LD se objevují v podstatě logické funkce AND, OR, NOT atd. Instrukce PLC přebírají názvosloví z elektrotechniky.



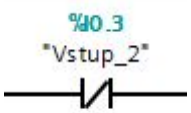
Obrázek 7: Zleva logické operace A AND B, A OR B, NOT A, SW [3]

NO kontakt – z angl. *Normally open contact* je v klidovém stavu rozepnutý, tj. nevede signál.



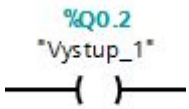
Obrázek 8: NO kontakt

NC kontakt – z angl. *Normally closed contact* je v klidovém stavu sepnutý, tj. vede signál.



Obrázek 9: NC kontakt

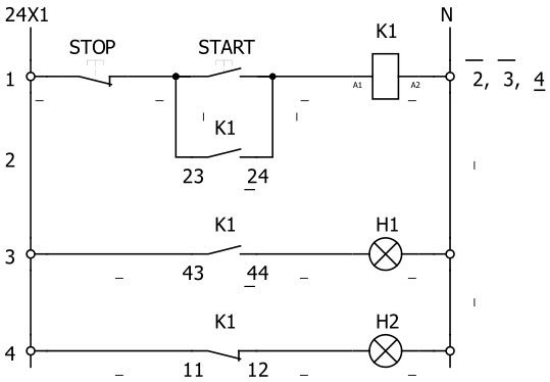
Cívka – z angl. *Coil* přiřazuje danému bitu log. 1. Dnes se setkáme také s označením přiřazení (z angl. *Assignment*).



Obrázek 10: Cívka

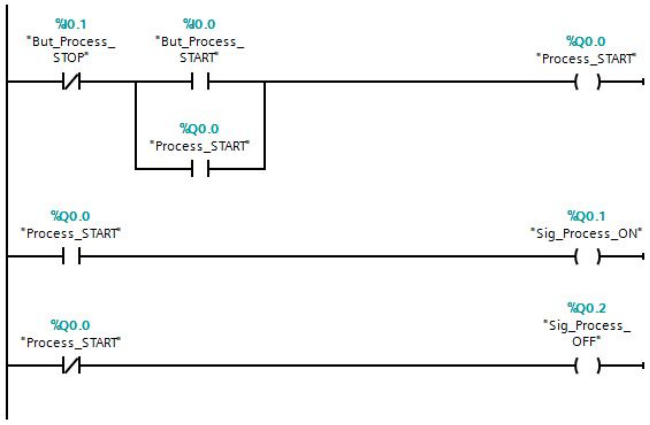
Úloha č. 2 – Ahoj Světe

Nyní si na příkladu ukážeme rozdíl mezi bitovými instrukcemi v LD a značkami v liniovém schematu. Zatímco ve vyšších programovacích začínáme úlohou „Ahoj Světe“, tj. vypsání textu na obrazovku PC, v LD chceme sepnout nějaký výstup na základě sepnutého vstupu.
Řekněme, že liniové schema obsahuje tlačítka START a STOP, relé K1 a žárovky H1 a H2. Jedná se o klasický případ spínání nějakého konkrétního procesu. Tlačítko START a relé tento proces spouští, tlačítko STOP přerušuje, žárovka H1 signalizuje jeho běh a žárovka H2 jeho přerušení. V liniovém schematu je nutné označit svorky připojovaných prvků. Všimněme si dvou použitých NO kontaktů. Relé K1 má fyzicky omezený počet kontaktů. Z hlediska zatížitelnosti a také estetiky zapojení plní každý kontakt jinou funkci. V tomto případě je jeden tzv. přídržný kontakt (svorky 23, 24) a druhý spíná žárovku H1 (svorky 11, 12). NC kontakt (svorky 11, 12) spíná žárovku H2.



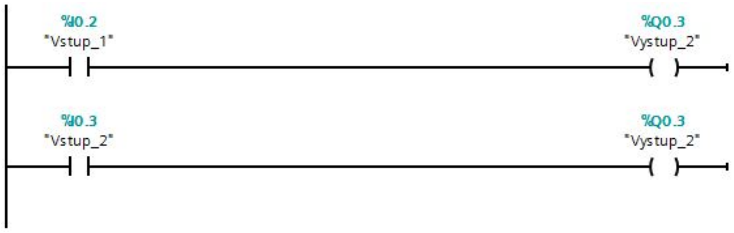
Obrázek 11: Liniové schema spuštění procesu, SW [2]

Nyní převeďme toto liniové schema do LD. Namísto spínacího relé K1 nám stačí jeden digitální výstup Q0.0. Žárovku H1, resp. H2, přiveďme na výstup Q0.1, resp. Q0.2. Tlačítko START, resp. STOP přiveďme na vstupy I0.0, resp. I0.1. Na první pohled je zapojení totožné, což je jedna z výše zmíněných výhod PLC.



Obrázek 12: LD spuštění procesu

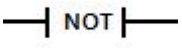
Podívejme se na kontakty výstupu Q0.0. V podstatě se těmito instrukcemi ptáme, zda je přímo výstup Q0.0 sepnutý nebo rozepnutý a můžeme se takto ptát na mnoha místech v programu, jelikož nejsme omezeni fyzickým počtem kontaktů. Avšak cívku Q0.0 nemůžeme použít na více místech, protože pro jeden bit platí jedna logická operace.



Obrázek 13: Musí být sepnuté oba vstupy, aby byl sepnutý výstup. Jedná se o STEJNOU cívku.

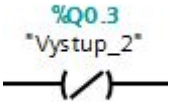
Další instrukce

Negace – obrací výsledek logické operace (RLO – z angl. *Result of logic operation*).



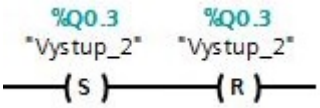
Obrázek 14: Negace

Negované přiřazení – rozepíná daný operand.



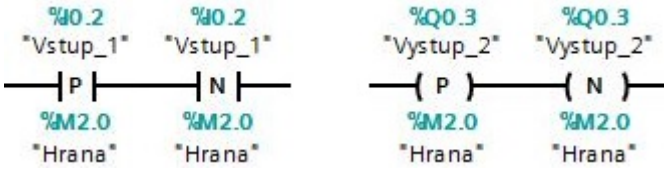
Obrázek 15: Negované přiřazení

Set – přiřadí operandu log. 1 a ten zůstává v log. 1, bez ohledu na dalším RLO.
Reset – přiřadí operandu log. 0 a ten zůstává v log. 0, bez ohledu na dalším RLO.



Obrázek 16: Zleva SET a RESET

Náběžné (P – positive) a sestupné (N – negative) hrany – čte hrany signálu (nějakého bitu) nebo RLO. Musíme použít další bit, ve kterém bude uložen výsledek předešlého čtení.



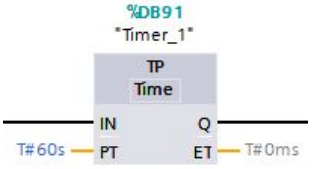
Obrázek 17: Čtení hrany zleva signálu a RLO

2.1.2 Instrukce časovačů

Ať už je PLC zapojeno do jakéhokoli reálného procesu, většinou se neobejdeme bez použití časovačů. Představíme si tři základní instrukce, resp. funkční bloky časovačů.

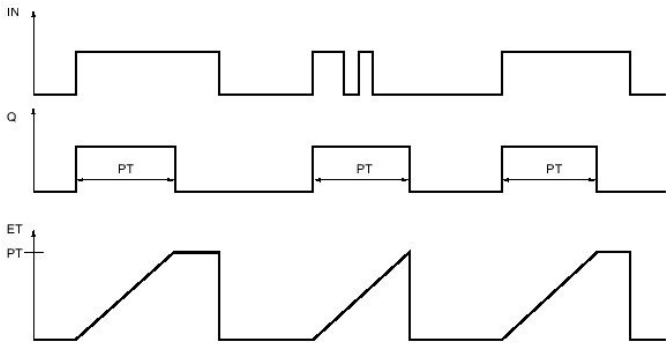
Rozhraní funkčních bloků časovačů

IN – vstup plní funkci zapnutí časovače.
Q – výstup je sepnut časovačem.
PT – přednastavený čas (z angl. *Preset time*) je nastaven programátorem.
ET – uběhlý čas (z angl. *Elapsed time*) je uběhlý čas.



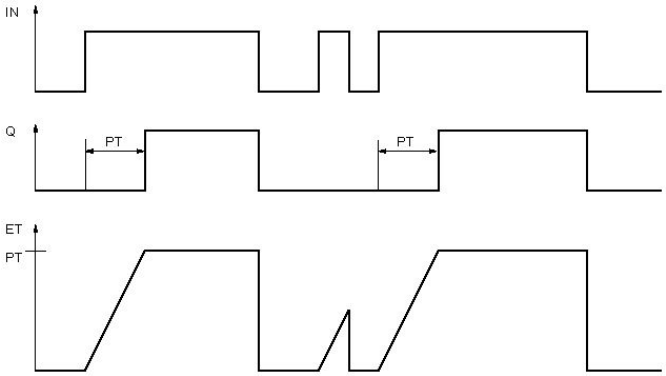
Obrázek 18: Funkční blok časovače

Generování pulzu – TP (z angl. *Time pulse*)
Při náběžné hraně IN časovač vygeneruje na Q pulz o pevné délce trvání PT. Tento pulz není přerušen případnými změnami IN. Další náběžná hrana IN je vyhodnocena až při rozepnutém Q.



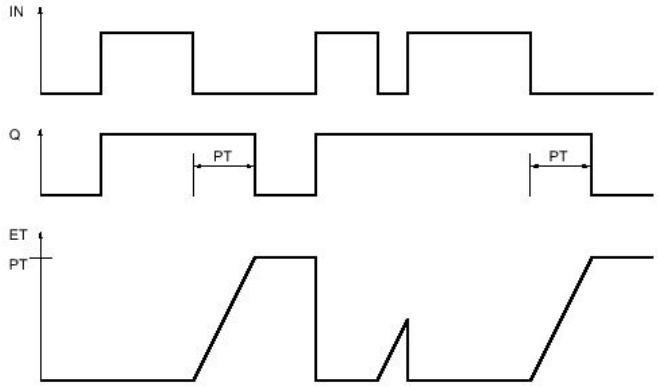
Obrázek 19: Generování pulzu, převzato z [6]

Zpožděné zapnutí – TON (z angl. *Timer ON*)
Pokud je IN log. 1, časovač počká dobu PT, a poté sepne Q. Hodnota IN log. 0 rozepíná Q, tj. pokud na IN přivedeme pulz o délce trvání kratší než PT, Q vůbec nesepneme.



Obrázek 20: Zpožděné zapnutí, převzato z [6]

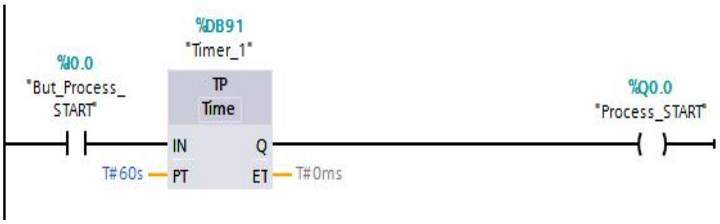
Zpožděné vypnutí – TOF (z angl. *Timer OFF*)
Pokud je IN log. 0, časovač počká dobu PT, a poté rozepne Q. Hodnota IN log. 1 spíná Q, tj. pokud IN rozepneme na dobu kratší než PT, Q vůbec nerozepneme.



Obrázek 21: Zpožděné vypnutí, převzato z [6]

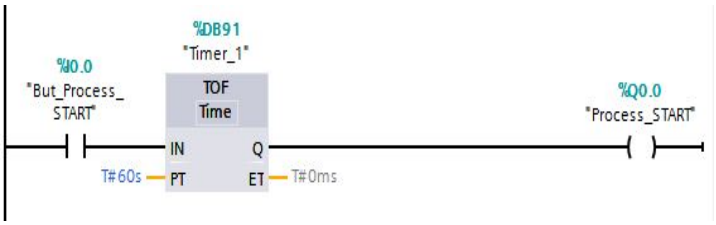
Úloha č. 3 – Chlazení

Tlačítkem budeme chtít spustit proces chlazení na dobu 60 s. Tlačítko START připojíme na vstup I0.0, výstup Q0.0 spíná proces chlazení. Nabízí se použít instrukci TP. Na IN přivedeme I0.0, na Q přivedeme Q0.0 a čas PT nastavíme na 60 s.



Obrázek 22: Zapnutí chlazení

Při jednom stisknutí tlačítka proběhne celý proces chlazení právě jednou. Pokud by operátor tlačítko přidržel na jakkoli dlouhou dobu, proces stále proběhne pouze jednou. Čekáme totiž na náběžnou hranu tlačítka. Při dalších stisknutích během spuštěného procesu, se chlazení nepřeruší ani znovu nezapne. Musí totiž proběhnout celé.
Pojďme nyní namísto instrukce TP použít TOF. Při jednom stisknutí tlačítka, proces v podstatě proběhne podle zadání. Problém nastane při přidržení tlačítka, popř. jeho několika stisknutích. Po tuto dobu je proces stále spuštěn. Čekáme totiž na sestupnou hranu tlačítka. Touto operací se délka procesu prodlužuje.



Obrázek 23: Zapnutí chlazení TOF

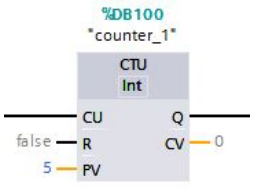
Pro zadanou úlohu je vhodnější použít instrukci TP. Záleží samozřejmě na dalších detailech zadání. Nemusí např. vadit, že operátor dalším stisknutím tlačítka chlazení prodlouží. Tato situace je málo pravděpodobná ale ne nereálná. Uživatelsky přijatelnější je zpřístupnit zápis hodnoty PT obsluze na HMI panelu.

2.1.3 Instrukce čítačů

Čítače slouží k inkrementaci či dekrementaci nějakého počtu (např. cyklů) při splnění určité podmínky a vyhodnocení dosažení přednastavené hodnoty tohoto počtu. Ukážeme si dvě základní instrukce.

Rozhraní funkčních bloků čítačů

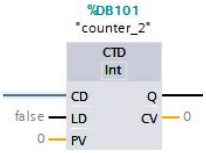
CU / CD – vstupní podmínka inkrementace (z angl. *Count up*) / dekrementace (z angl. *Count down*) čítače.
Q – výstup je sepnut čítačem po dosažení určité hodnoty.
PV – přednastavená hodnota (z angl. *Preset value*) je nastavena programátorem.
CV – aktuální hodnota čítače (z angl. *Current value*).
R / LD – podmínka pro reset / naplnění (z angl. *Load*) čítače.



Obrázek 24: Přičítání

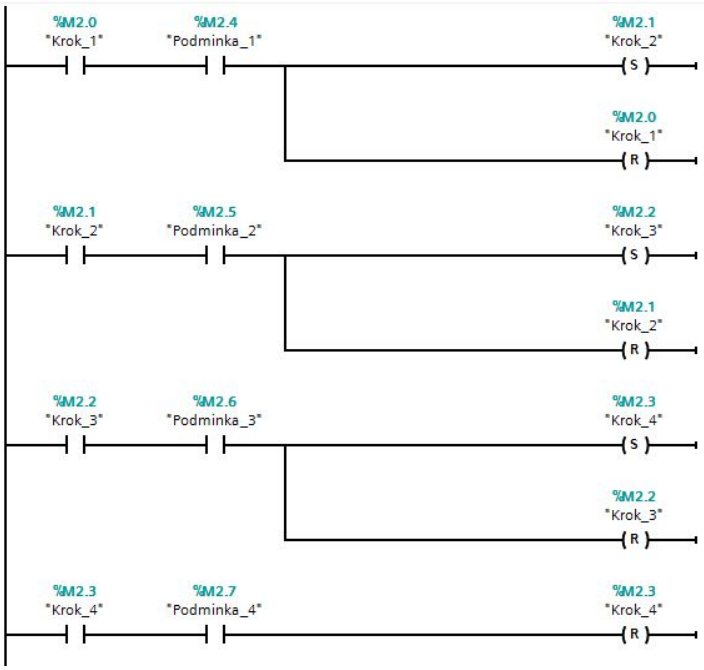
Přičítání – CTU
Při náběžné hraně vstupu CU je hodnota CV inkrementována o jedna. Výstup Q je sepnut, pokud hodnota CV je větší nebo rovna hodnotě PV. Při náběžné hraně vstupu R je hodnota CV resetována na nulu. Vstup CU čítač ignoruje, pokud je vstup R v log. 1, tj. neproběhne přičítání.

Odpočítávání – CTD
Při náběžné hraně vstupu CD je hodnota CV dekrementována o jedna. Výstup Q je sepnut, pokud hodnota CV je menší nebo rovna nule. Při náběžné hraně vstupu LD je hodnota CV nastavena na hodnotu PV. Vstup CD čítač ignoruje, pokud je vstup LD v log. 1, tj. neproběhne odečítání.



Obrázek 25: Odpočet

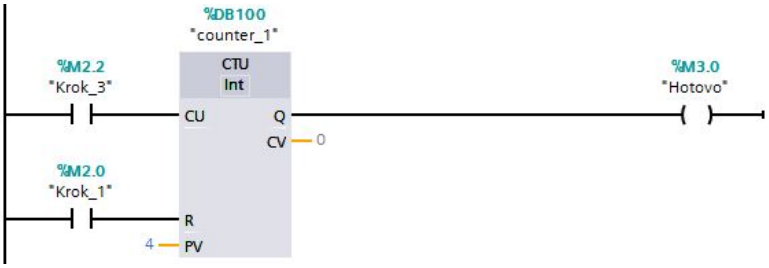
Úloha č. 4 – Sekvence a cyklus
Řekněme, že v nějaké sekvenci (tj. kroky procesu na sebe postupně navazují) máme 4 kroky. Tato krátká sekvence může být napsána velice jednoduše použitím bitové logiky. Mějme 4 bity pro kroky a 4 bity pro podmínky ukončení jednotlivých kroků (paměťové bity[1] M2.0 – M2.7).



Obrázek 26: Sekvence o čtyřech krocích

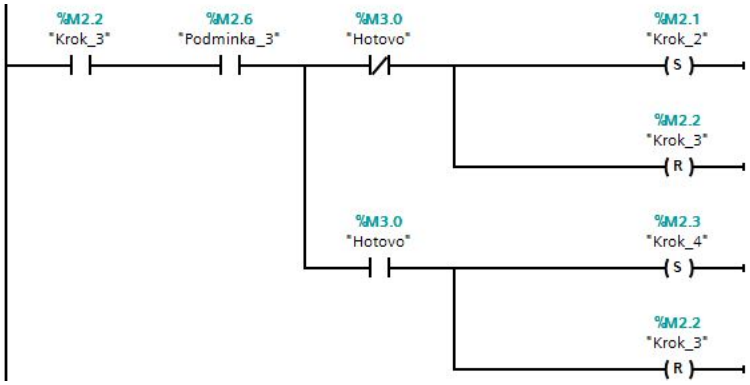
Nyní budeme chtít kroky 2 a 3 čtyřikrát zopakovat. Použijeme instrukci CTU, kde hodnotě PV přiřadíme hodnotu 4, na vstup CU přivedeme poslední krok, který chceme opakovat, tj. krok 3, na vstup R můžeme přivést např. krok 1 a stav čítače, tj. výstup Q budeme indikovat bitem M3.0.

1 Paměťové bity jsou vnitřní bity PLC, tj. nejsou to hardwarové vstupy ani výstupy.



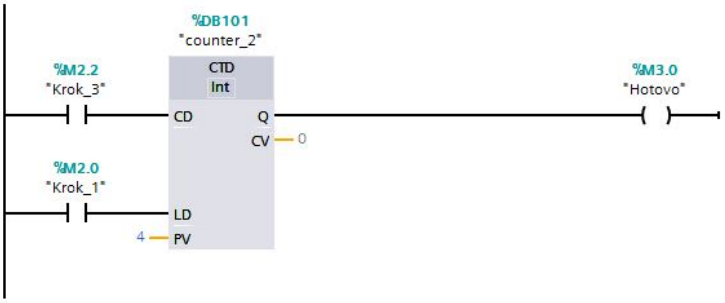
Obrázek 27: Opakování kroků 2 a 3

Ještě je zapotřebí vhodně upravit naší sekvenci. Změnu provedeme pouze v kroku 3, kde se bude rozhodovat o ukončení cyklu a pokračování sekvence.



Obrázek 28: Větvení v kroku 3

Pro zajímavost můžeme zkusit použít také instrukci CTD. Nezmění se logika čítače ani kroku 3. Obě instrukce jsou zaměnitelné.



Obrázek 29: Výměna čítače

2.2 Schema funkčních bloků

Schema funkčních bloků (FBD – z angl. *Function block diagram*) můžeme přirovnat ke schematům s logickými hradly. Analogie je zřejmá a platí, že téměř každý program napsaný v LD lze převést na FBD a naopak.

Úloha č. 5 – Ahoj Světe v FBD

Převeďme naši úlohu do jazyka FBD.

Obrázek 30: Úloha Ahoj Světe analogicky v FBD

2.3 Seznam výroků

Jazyk STL (z angl. *Statement list*) se velice podobá jazyku symbolických instrukcí pro programování mikrokontrolérů. Do jazyka STL lze převést programy napsané v LD popř. FBD, naopak už můžeme narazit na problémy.

Úloha č. 6 – Ahoj Světe v STL

Převeďme naši úlohu do jazyka STL.

```
1     AN    "But_Process_STOP"          %I0.1
2     A(
3     O     "But_Process_START"         %I0.0
4     O     "Process_START"             %Q0.0
5     )
6     =     "Process_START"             %Q0.0
7
8     A     "Process_START"             %Q0.0
9     =     "Sig_Process_ON"            %Q0.1
10
11    AN    "Process_START"             %Q0.0
12    =     "Sig_Process_OFF"           %Q0.2
```

Obrázek 31: Úloha Ahoj Světe analogicky v STL

2.4 Strukturovaný jazyk řízení

Jazyk SCL (z angl. *Structured Control Language*) patří mezi vyšší programovací jazyky a podobá se Pascalu. Pokud neřešíme pouze logiku, ale potřebujeme také složitější matematické funkce, je vhodné použít jazyk SCL.

Úloha č. 7 – Ahoj Světe v SCL

Převeďme naši úlohu do jazyka SCL.

```
1  IF NOT("But_Process_STOP") AND ("But_Process_START" OR "Process_START") THEN
2      "Process_START" := TRUE;
3  END_IF;
4
5  IF "Process_START" THEN
6      "Sig_Process_ON" := TRUE;
7  END_IF;
8
9  IF NOT("Process_START") THEN
10     "Sig_Process_OFF" := TRUE;
11 END_IF;
```

Obrázek 32: Úloha Ahoj Světe analogicky v SCL

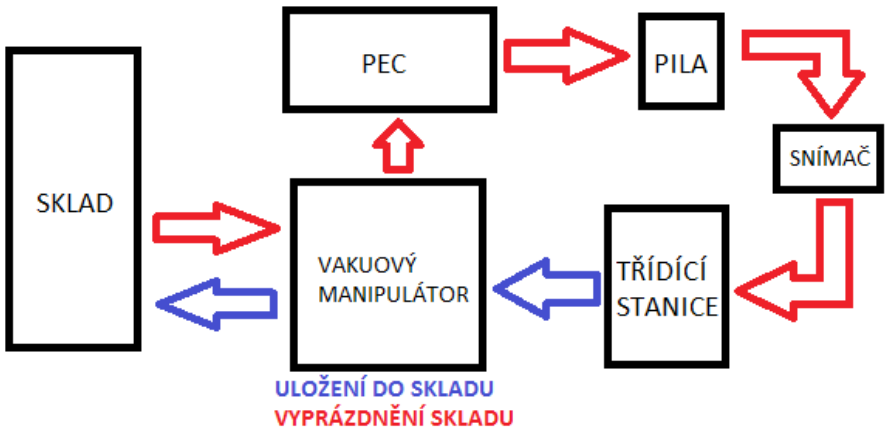
3. Model továrny

3. 1 Úvod

Využijme model továrny výrobce FischerTechnik (viz. Příloha A), abychom si ukázali základní principy programování PLC. Během vytváření projektu nenarazíme jen na základní instrukce, které jsme si ukázali v předchozích kapitolách, ale také na složitější situace. Naučíme se vytvářet vlastní funkční bloky a jejich volání, podíváme se na vstup s rychlým čtením pulzů, setkáme se s A/D převodníkem a napíšeme si krátkou funkci v jazyku SCL.

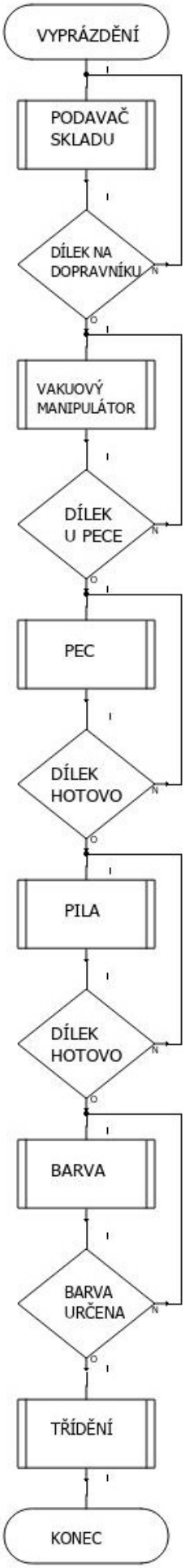
3. 2 Popis funkce celé továrny

Továrna obsahuje celkem čtyři stanice: sklad, vakuový manipulátor, výrobní stanici s pecí, třídící stanici. Řekněme si, jak by měl celý náš projekt fungovat.
Na začátku je sklad zaplněn dílky v bedničkách. Dílky nemusí být seřazené podle barvy. Podavač odebere dílek a odveze ho na dopravník. Dopravník dílek přiveze k vakuovému manipulátoru. Manipulátor odebere dílek z bedničky a odveze ho k peci. Dopravník mezitím odveze prázdnou krabičku zpět k podavači, který ji umístí na původní místo ve skladu.
Vakuový manipulátor odloží dílek na podavač pece, který poté zajede do pece. Probíhá pečení symbolizováno světelným signálem. Poté malý vakuový manipulátor odveze dílek na otočný stůl, který ho přemístí na pozici pod pilou. Opracování dílku pilou je opět pouze symbolické. Po opracování je dílek otočným stolem a pneumatickým vyhazovačem umístěn na dopravník.
Dopravník třídící stanice dílek proveze komorou se snímačem barvy. Za komorou je dílek umístěn jedním z vyhazovačů na pozici dle své barvy.
Takto můžeme vyprázdnit celý sklad. Následně můžeme již roztříděné dílky umístit zpět do skladu. Pokud jsou sklad a vakuový manipulátor naprogramovány na jednu sekvenci (např. po jedné barvě a po sloupcích), na konci procesu budou dílky ve skladu uspořádány. Možné cesty jednoho dílku znázorňuje obr. 33 a funkci celé továrny vývojové diagramy na obr. 35 a 34.



Obrázek 33: Schema procesů továrny, SW[3]

Obrázek 34: Vývojový diagram ULOŽENÍ, SW[2]



Obrázek 35: Vývojový diagram VYPRÁZDNĚNÍ, SW[2]

3. 3 HW konfigurace PLC a zapojení

Celý projekt napíšeme v programu TIA Portal 14. Začneme HW konfigurací PLC. K dispozici máme CPU 1215C DC/DC/DC výrobce Siemens a rozšiřující vstupně-výstupní modul výrobce Beckhoff (PROFINET bus BK9053, 5x digitální vstupy KL1408, 5x digitální výstupy KL2408). Nejdůležitější část pro naši orientaci v psaném programu je *přehled zařízení – Device overview*, kde vidíme adresy a typy vstupů a výstupů.
Dále podle zapojení konektorů na deskách jednotlivých stanic továrny vhodně připojíme celou továrnu k PLC a doplníme celý projekt o pět tlačítek, kterými budeme model řídit.
Podrobnosti viz Příloha B.

3. 4 Programování jednotlivých stanic

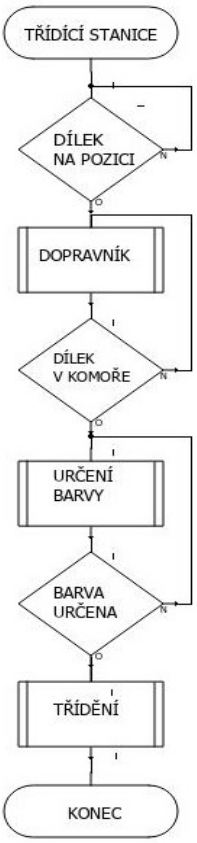
3. 4. 1 Rozvaha

Ze všeho nejdříve zavedeme tzv. štítky (angl. *tagy*), což je adresace daných vstupů a výstupů. Analogové vstupy a výstupy budeme muset vhodně nadefinovat. Dále promyslíme logiku řízení jednotlivých výstupů, kterou se budeme snažit oddělit od logiky samotného procesu. Proces si rozdělíme na dílčí podprocesy. K tomu můžeme využít networků (sítí). Ke každému funkčnímu bloku si podle potřeby zároveň vytvoříme alespoň jeden datový blok, kde nadefinujeme proměnné.
Podle vývojových diagramů naprogramujme dílčí funkce jednotlivých stanic. Jako první zvolíme třídící stanici z důvodu lehčího přístupu k programování. Postupně budeme funkce vylepšovat a s každým přestupem na další stanici si ukážeme další možnosti programování.
Celý SW v programu TIA Portal V14 je k dispozici na CD, jež je přílohou této práce. Popis všech proměnných použitých v daném výpisu viz Příloha C.

3. 4. 2 Třídící stanice

Funkce

Hlavním prvkem třídící stanice je dopravník řízený motorem, který převeze dílek od výrobní stanice (pily) přes komoru se snímačem barvy k vyhazovačům.
Snímač barvy určí, zda se jedná o dílek červené, modré nebo bílé barvy.
Pneumatické vyhazovače dílek správně zařadí podle barvy na jednu ze tří pozic.

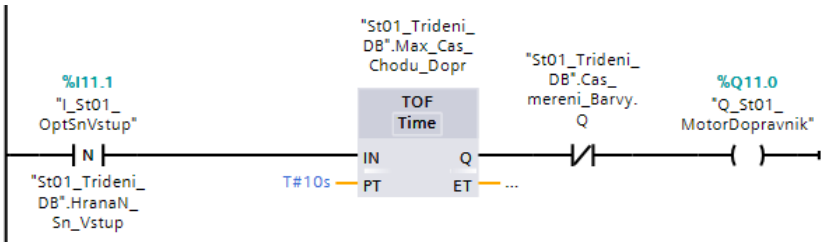


Obrázek 36: Vývojový diagram TŘÍDÍCÍ STANICE, SW[2]

Dopravník

Motor

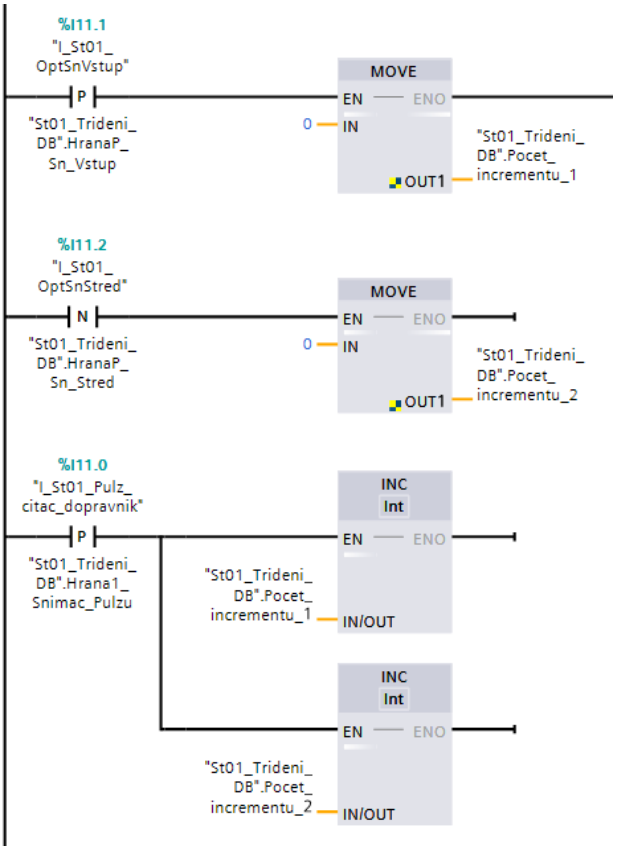
Dopravník se rozjede, pokud snímač na jeho vstupu detekuje přítomnost dílku. Uvědomme si, že v ideálním případě dílek na snímači nezůstává po celou dobu chodu dopravníku, a tak bychom mohli na výstup trvale zapsat log. 1, a poté vhodnou podmínkou znovu motor resetovat. Jako rozumnější variantu zvolíme nějaký vhodný časový limit chodu, např. v našem případě 10 s.



Obrázek 37: Motor dopravníku

Chod

Nejprve si zjistíme, kolik pulzů čítače dopravníku připadne na daný úsek dráhy. Poté budeme vhodně počítat pulzy. Potřebujeme počet pulzů pro vzdálenost od začátku dopravníku do komory snímače barvy a pro vzdálenost od snímače za komorou k danému pístu.

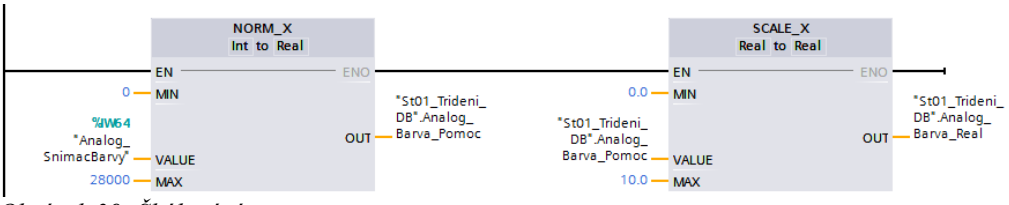


Obrázek 38: Chod dopravníku

Barva dílku

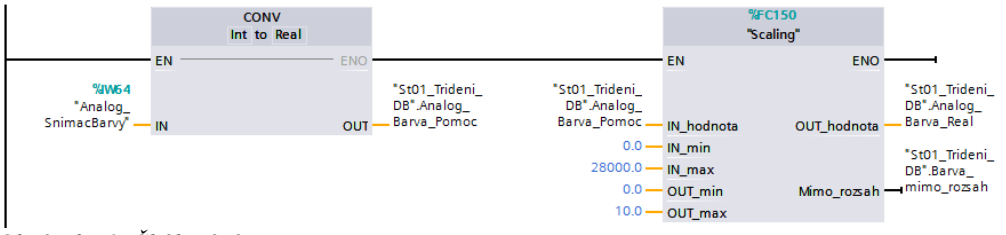
Měření barvy

Nejprve analogový vstup AI0 (adresa IW64) nadefinujeme pro měření napětí o rozsahu 0-10 V. Zjistíme si specifikaci A/D převodníku PLC a naměřenou hodnotu vhodně škálujeme. Škálovat můžeme vícero způsoby. Jednou z možností je vstupní hodnotu znormovat a následně škálovat. Výhodou je, že nemusíme nic vymýšlet a pouze použijeme instrukce, které nám program nabízí.



Obrázek 39: Škálování

Můžeme si také napsat jednoduchou funkci *Scaling* v jazyku SCL. Nesmíme však zapomenout na konverzi hodnoty na reálné číslo. Výhodou tohoto řešení je možnost definování různých podmínek během měření barvy, např. zjištění, že je barva mimo požadovaný rozsah. Představme si danou situaci jako detekování vadného dílku, a tedy jeho nepuštění dále do procesu. V našem případě tento problém vynecháme.

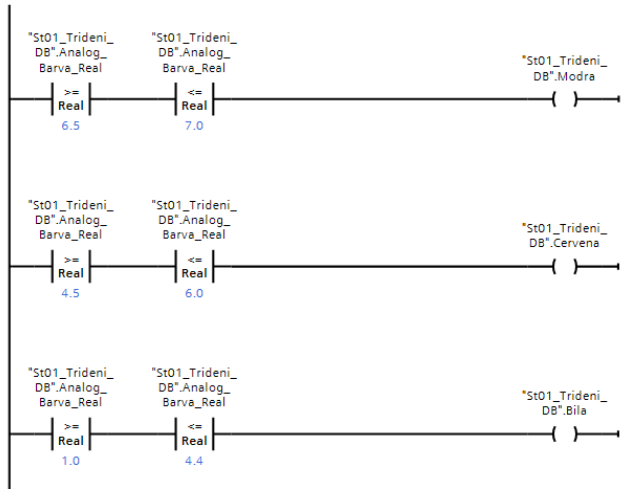


Obrázek 40: Škálování

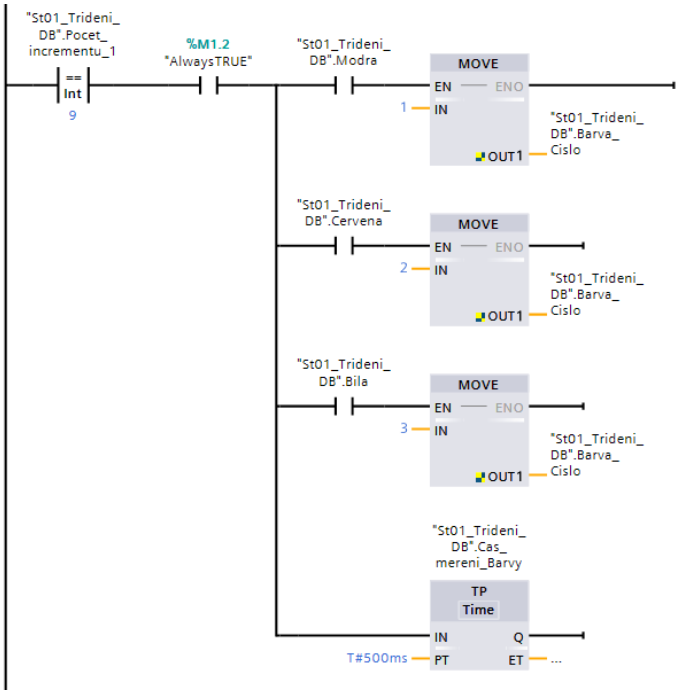
```
Scaling:
#Rozsah_IN := #IN_max - #IN_min;
#Rozsah_OUT := #OUT_max - #OUT_min;
#Nasobitel := #Rozsah_OUT / #Rozsah_IN;
#OUT_hodnota := (#Nasobitel * (#IN_hodnota - #IN_min)) + #OUT_min;
#Mimo_rozsah := FALSE;
IF      #IN_hodnota < #IN_min THEN
        #OUT_hodnota:= #OUT_min;
        #Mimo_rozsah := TRUE;
END_IF;
IF      #IN_hodnota > #IN_max THEN
        #OUT_hodnota := #OUT_max;
        #Mimo_rozsah := TRUE;
END_IF;
```

Určení barvy

Po škálování určíme o jakou barvu se jedná. Výsledek uložíme do proměnné až ve chvíli, kdy se dílek nachází v komoře, což určíme počtem pulzů čítače dopravníku. V tuto chvíli si uvědomme, že dopravník by měl během snímání barvy krátkou dobu stát. Využijeme tedy blok časovače a vhodně tento čas reflektujeme v logice řízení motoru dopravníku.



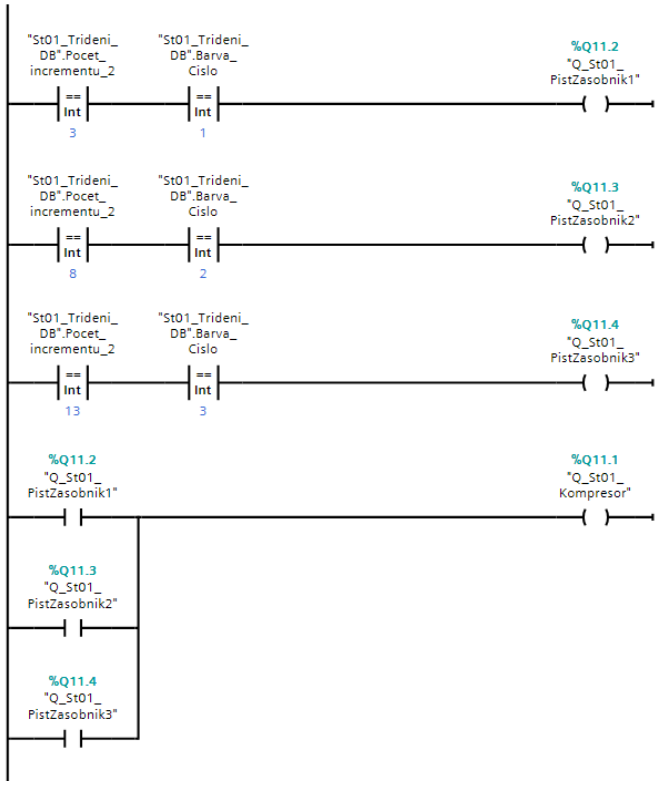
Obrázek 41: Určení barvy



Obrázek 42: Určení barvy

Roztřídění

Podle barvy a vzdálenosti aktivujeme daný vyhazovač. Potřebujeme vzduch, a tak určíme podmínky běhu kompresoru.



Obrázek 43: Roztřídění

3. 5. 3 Pec s pilou

Funkce

Stanice čeká, až vakuový manipulátor přenese dílek na podavač pece. Poté se otevřou dveře do pece a podavač zajede dovnitř. Pečení je signalizováno LED diodou.

Stanice obsahuje malý vakuový manipulátor, který přenese hotový dílek z podavače pece na otočný stůl. Otočný stůl převeze dílek k pile řízené motorkem a následně k pneumatickému vyhazovači, který dílek vyhodí na dopravník výrobní stanice. Tento dopravník je vhodně přistaven k dopravníku třídící stanice, a tak převeze dílek k třídění.

Tuto stanici začneme programovat krokově. Dále vytvoříme funkce zvlášť pro pec a pro pilu s transferem.

Ukážeme a vysvětlíme si jen nejdůležitější funkce a nebudeme vypisovat všechny kroky vzhledem k jejich podobnosti.

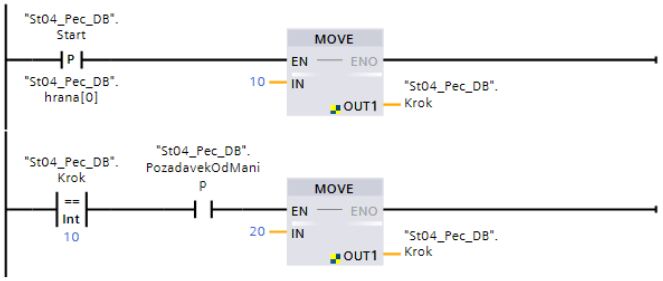
Nakonec určíme v jakých krocích budou spuštěny jednotlivé výstupy stanice (motory, vakuum, kompresor).

Obrázek 44: Vývojový diagram PEC S PILOU, SW[2]

Pec

Start sekvence

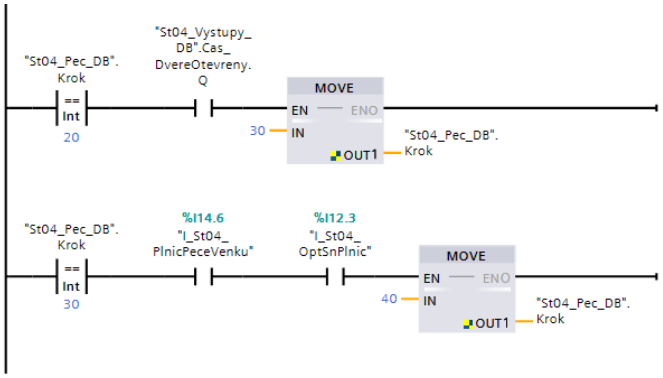
Startovací signál spustí krokování programu, na rozdíl od třídící stanice, kde jsme situaci, kdy není operátorem spuštěn celý cyklus neřešili. V kroku 70 pec čeká na signál od manipulátoru. Tímto způsobem budou psány v podstatě všechny kroky.



Obrázek 45: Start sekvence pece

Otevření dveří pece a vyjetí plniče

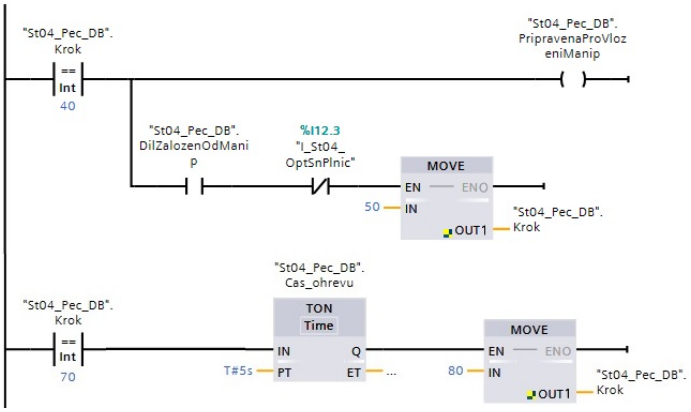
V kroku 20 se ujistíme, že se dveře opravdu otevřely vhodným časovým zpožděním, poté v kroku 30 vyjede plnič a optický snímač zjistí, zda nebyl v peci dílek. Tato informace je jen pro naše ujištění a nebudeme zabývat chybovými stavy.



Obrázek 46: Kroky pece 20 a 30

Založení dílku a pečení

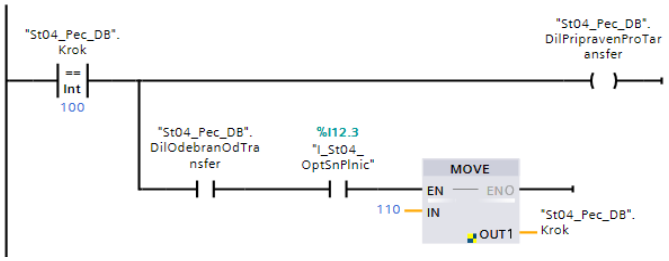
Manipulátor založí dílek na podavač. Stejně jako pec čekala na signál od manipulátoru, tak i on čekal na signál od pece, zda je připravena v kroku 40. Ve stejném kroku se znovu ujistíme, že dílek je opravdu na podavači. Poté podavač zajede dovnitř, dveře se zavřou (znovu volíme vhodné zpoždění) a na určitou dobu se v kroku 70 spustí pečení.



Obrázek 47: Kroky pece 40 a 70

Odebrání dílku

Po pečení dílek znovu vyjede ven z pece a transfer v kroku 100 pece reaguje na signál a dílek převezme. Ujistíme se, že dílek již na podavači není. Po odebrání podavač zajede zpět do pece a dveře se zavřou.



Obrázek 48: Krok pece 100

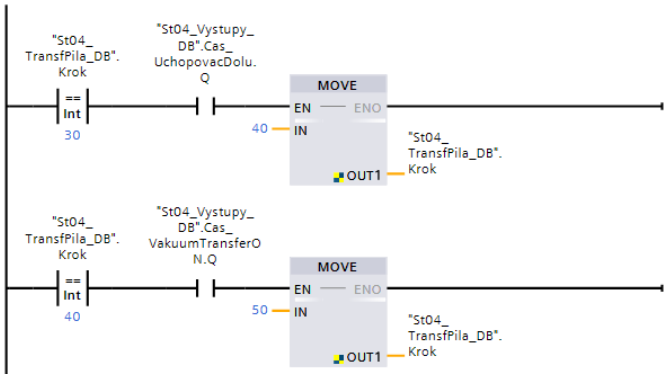
Pila s transferem

Start a příjezd transferu

Spuštění krokování řešíme stejně jako u pece. Transfer čeká na signál od pece, a poté k ní přijede.

Uchopení dílku

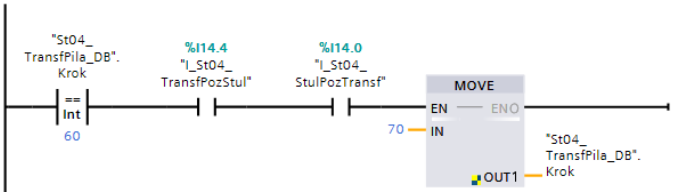
Píst transferu v kroku 30 sjede dolů, odebere dílek a vyjede nahoru. Znovu si uvědomme, že pracujeme s malými pneumatickými písty, a tak volíme vhodná časová zpoždění při jejich ovládání. V kroku 40 spustíme vakuum.



Obrázek 49: Kroky pece 30 a 40

Odložení dílku na stůl

Transfer v kroku 60 přijede ke stolu, u kterého zároveň kontrolujeme, zda je v pozici u transferu. Odložení probíhá analogicky k uchopení.



Obrázek 50: Krok pece 60

Opracování dílku

Stůl se otočí k pile, která běží vhodnou dobu.

Vyhození dílku

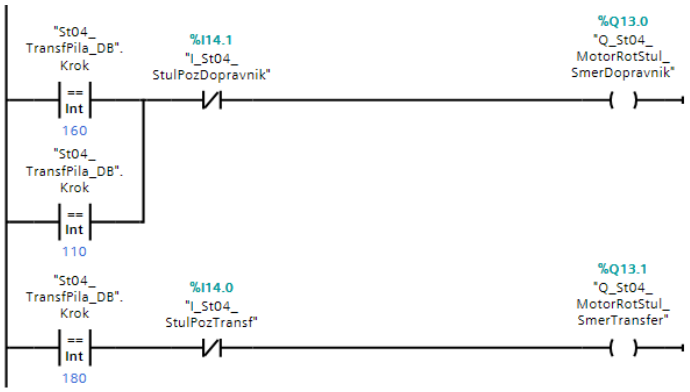
Stůl se otočí k dopravníku, kam vyhazovač vyhodí dílek, a poté se stůl vrátí do pozice u transferu.

Výstupy výrobní stanice

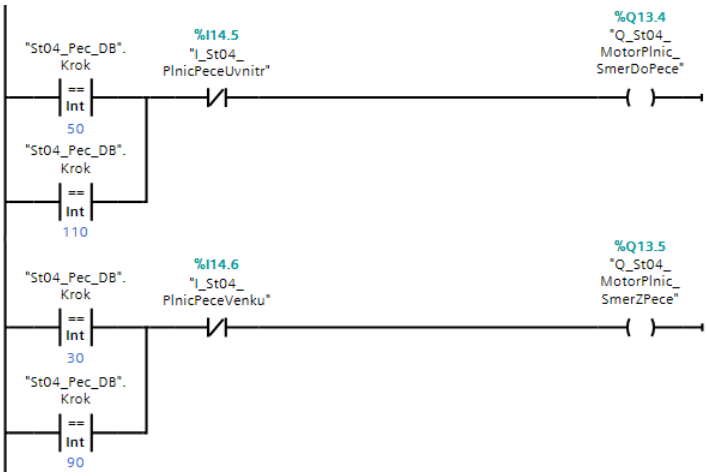
Výstupy budou spínány v jednotlivých krocích. Podle toho bude vypadat logika jejich ovládání.

Motory rotačního stolu, podavače pece a transferu

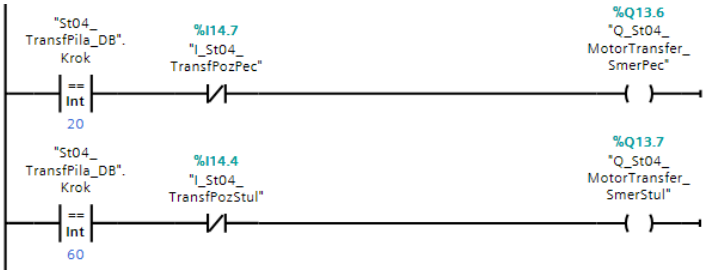
Hlídáme si jejich koncové spínače.



Obrázek 51: Motor stolku



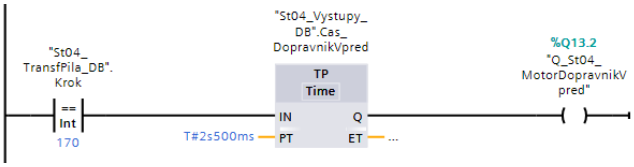
Obrázek 52: Motor podavače



Obrázek 53: Motor transferu

Motory dopravníku a pily

Zvolíme vhodnou dobu chodu motorů.



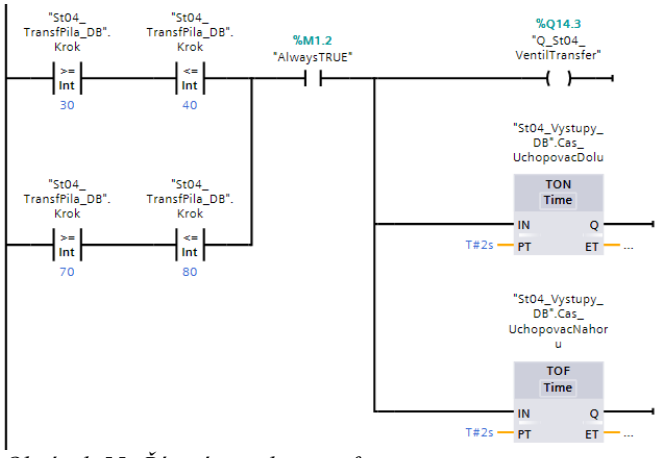
Obrázek 54: Motor dopravníku

Topení

Topení spustíme v kroku 70 pece.

Kompresor a ovládání pneumatických funkcí

Pro vakuum, manipulátor, dveře pece a vyhazovač stanovíme vhodná časová zpoždění. Bit AlwaysTRUE je stále v log. 1 a slouží pouze pro vizuální přehlednost.

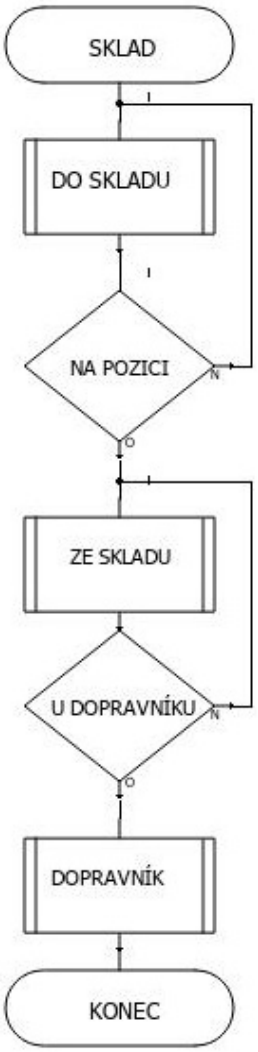


Obrázek 55: Řízení ventilu transferu

3. 5. 4 Sklad

Funkce

Horizontální a vertikální motory manipulátoru skladu obsahují enkodéry pro určení pozice, resp. počtu otáček daného motoru. Díky nim určíme absolutní pozice ve skladu a uložíme je do datového bloku.
Nebudeme vypisovat celou funkci, pouze dílčí funkce. Ať už bude manipulátor dílky zakládat do skladu nebo vykládat ze skladu, bude provádět vždy stejné kroky ve stejném pořadí, tzn. přijede na pozici ve skladu, vezme bedýnku, přijede k dopravníku, kterému bedýnku předá, poté bedýnku vrátí na stejnou pozici ve skladu a tyto kroky opakuje pro další pozici ve skladu. Během celého cyklu vhodně pracujeme s požadovanými pozicemi (přičítáme i odečítáme).



Obrázek 56: Vývojový diagram SKLAD, SW[2]

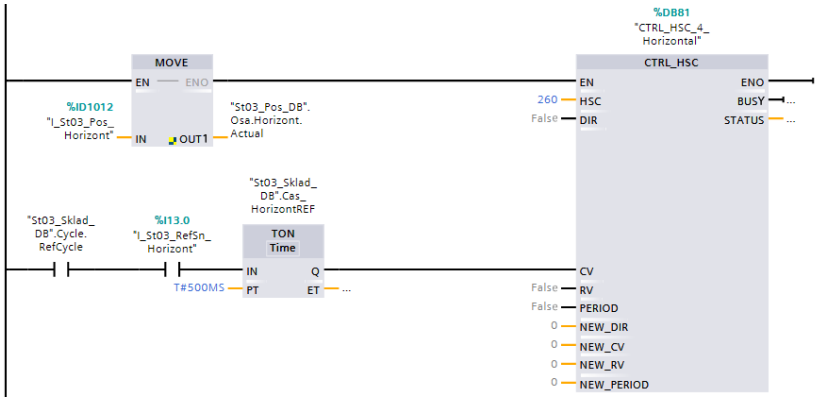
Měření pozic

Digitální vstupy I0.6 a I0.7 definujeme jako vysokorychlostní čítač HSC (adresa ID1012) pro signály A a B enkodéru horizontálního motoru. Analogicky také vstupy I1.0 a I1.1 u enkodéru vertikálního motoru (adresa ID1016).

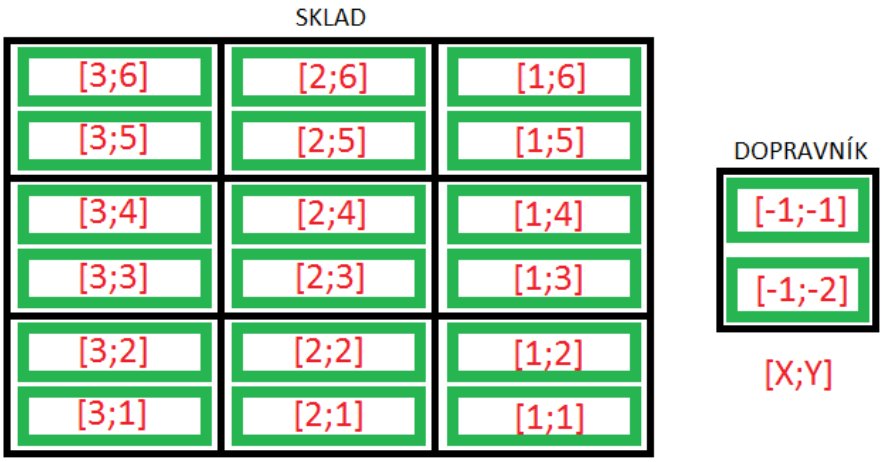
Pro pořádek budeme hodnoty z enkodérů ukládat do proměnné. Řídící bloky CTRL_HSC využijeme k nulování této hodnoty. Když manipulátor dojede na koncový spínač (referenční), po vhodném zpoždění dojde k nulování hodnoty. Zpoždění je nutné, protože při sepnutí se manipulátor nezastaví okamžitě, ale s mírným časovým zpožděním, a tak enkodér načte několik dalších pulzů.

Dále si uvědomme, že referovat (nulovat) chceme pouze během referenčního cyklu. Referenční cyklus v našem popisu vynecháme.

Pro požadované pozice nadefinujeme souřadnice X, Y (viz obr. 58) a tyto požadované pozice nadefinujeme (používáme zatím pouze absolutní hodnoty X a Y změřené pomocí enkodérů a navíc je vnímáme jako rozsah MIN a MAX).



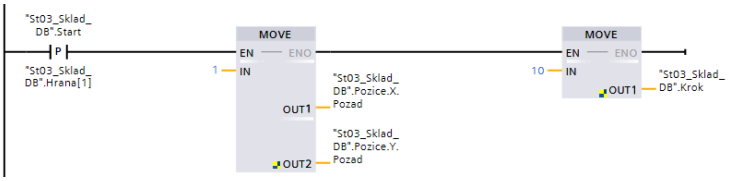
Obrázek 57: Měření polohy horizontálního motoru



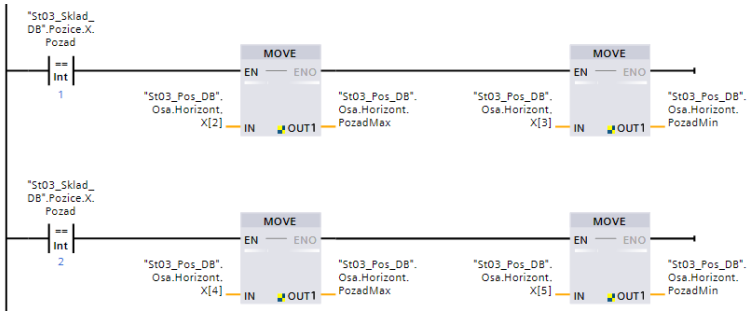
Obrázek 58: Nadefinování souřadnic pro pozice ve skladu a výstupního dopravníku, SW[3]

Start

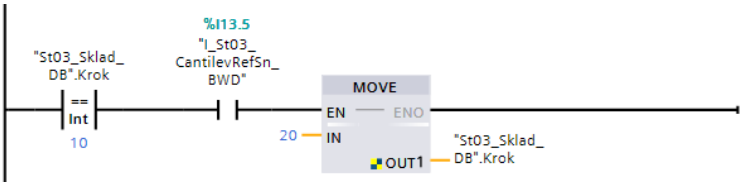
Startovací signál spustí krokování programu. Požadovanou pozici nastavíme X,Y = 1,1. Na tyto požadované pozice se ptáme analogicky i dále a určujeme horní a dolní mez enkodérů. Na úplném začátku v kroku 10 zajede podavač.



Obrázek 59: Start cyklu skladu



Obrázek 60: Příklad určení horní a dolní meze pro požadované pozice X

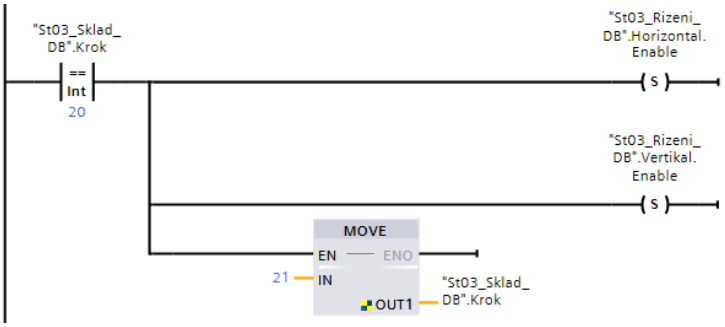


Obrázek 61: Zajetí podavače v kroku 10

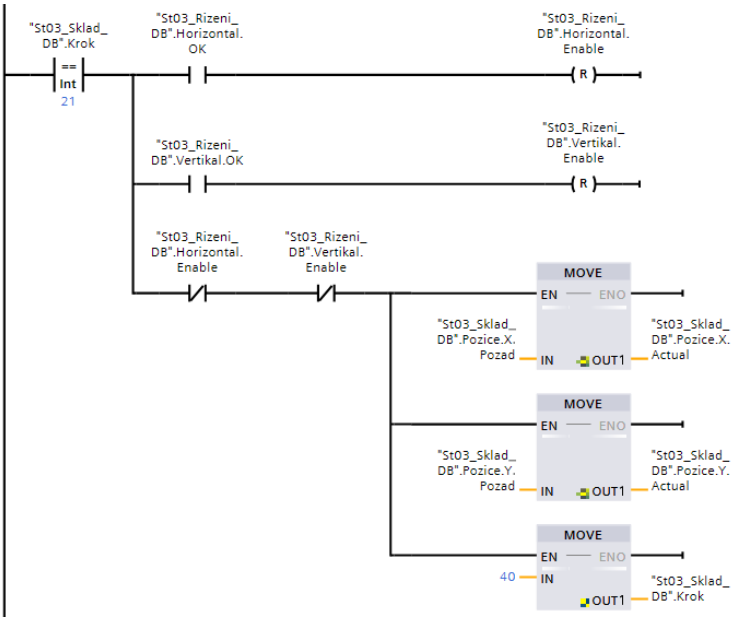
Do skladu

Příjezd na pozici a uložení aktuální polohy

Nejprve v kroku 20 povolíme pohyb v osách, a poté pohyb zakážeme v kroku 21, až bude manipulátor na požadované pozici. Na začátku cyklu ukládáme aktuální pozici ve skladu.



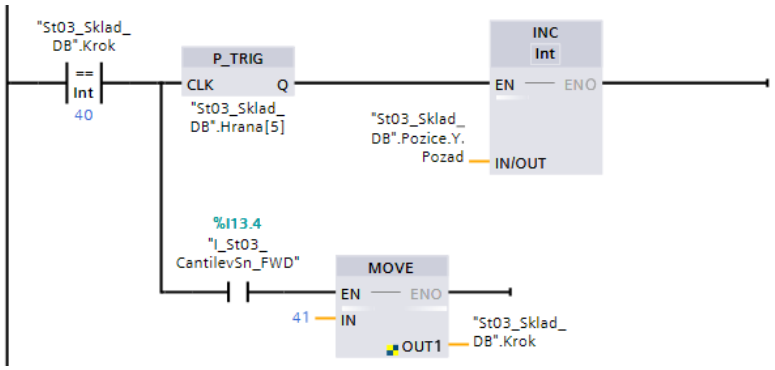
Obrázek 62: Povolení motorů v kroku 20



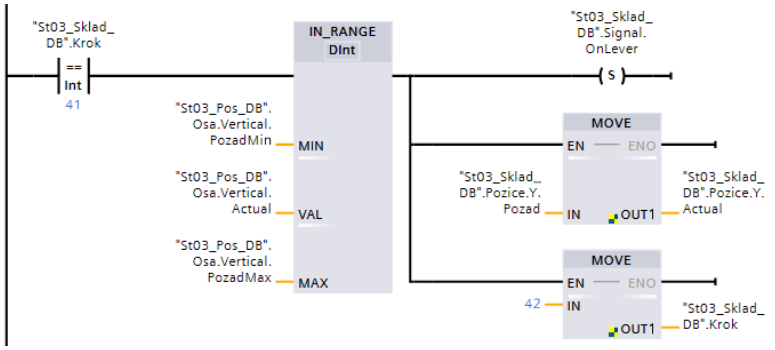
Obrázek 63: Uložení aktuální pozice v kroku 21

Odebrání bedýnky ze skladu

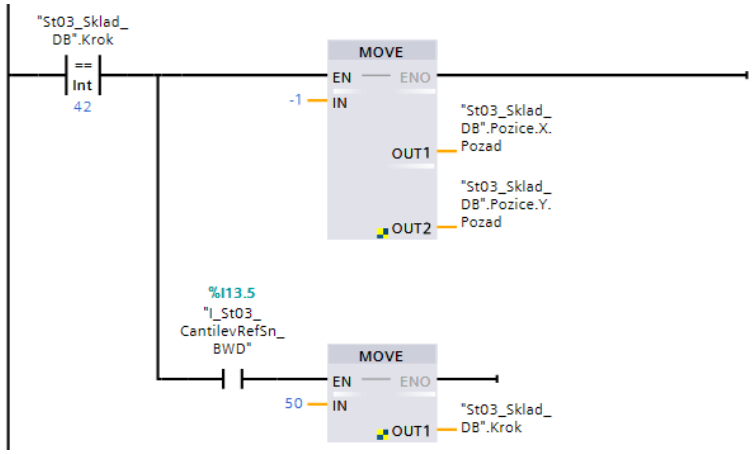
Při odebrání podavač vyjede do nižší polohy požadované pozice a pro posun manipulátoru do vyšší polohy použijeme blok IN_RANGE (v rozsahu). Tuto vyšší polohu uložíme (manipulátor se sem bude vracet). Signalizujeme přítomnost bedýnky na podavači. Během zajíždění podavače nastavíme požadovanou pozici na dopravník.



Obrázek 64: Vysunutí podavače v kroku 40



Obrázek 65: Posun manipulátoru v kroku 41



Obrázek 66: Zajetí podavače a nová pozice v kroku 42

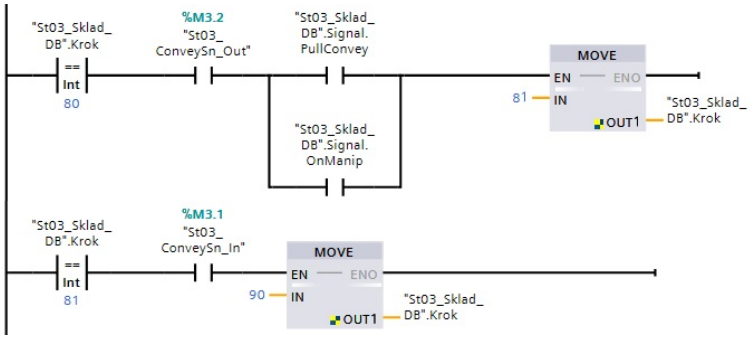
K dopravníku

Příjezd na pozici a odložení bedýnky

Probíhá stejně jako odebrání ze skladu, jen při odložení sjedeme z vyšší pozice do nižší a čekáme, až se bedýnka na dopravníku vrátí.

Dopravník

Dopravník odveze bedýnku na základě snímače přítomnosti bedýnky na jednom svém konci k vakuovému manipulátoru a čeká na signál od něj (buď dílek založil do bedýnky nebo jej z ní vyndal). Poté bedýnku přiveze zpět k podavači.



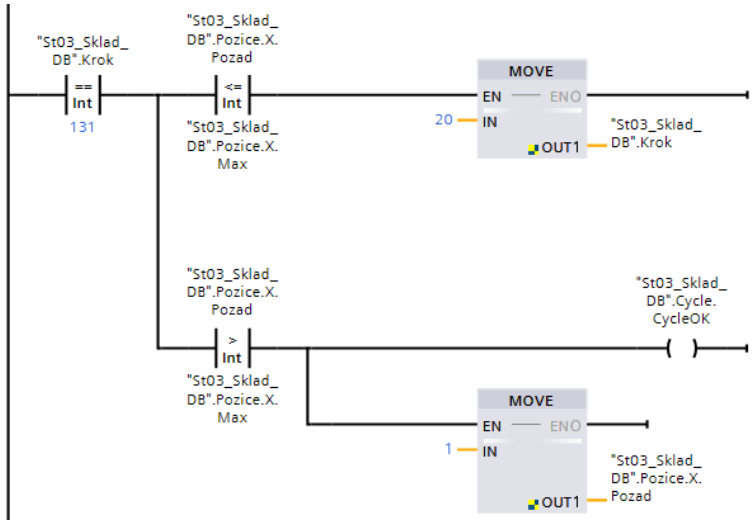
Obrázek 67: Funkce dopravníku v krocích 80 a 81

Návrat do skladu

Odebrání bedýnky z dopravníku, příjezd manipulátoru do skladu a odložení bedýnky zpět na původní pozici je analogické k předchozím funkcím a nebudeme je znovu vypisovat.

Kontrola cyklu

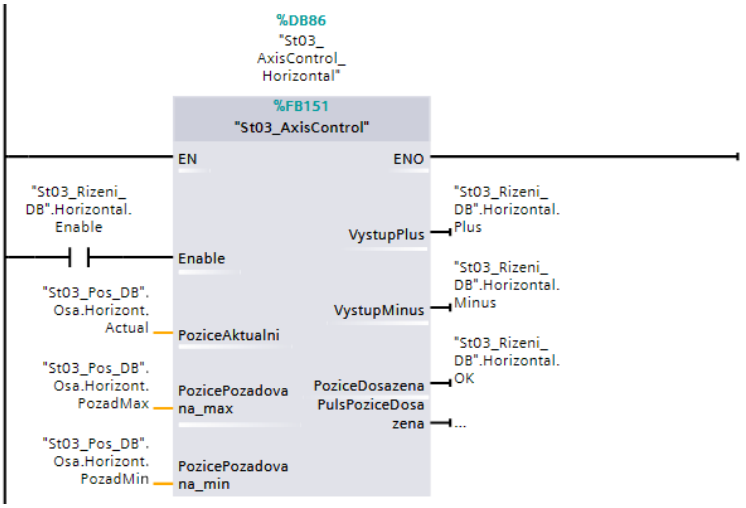
Porovnáváme vhodnými výpočty, zda bychom nebyli mimo sklad. Buď opakujeme kroky na další pozici nebo zakončíme vhodným signálem.



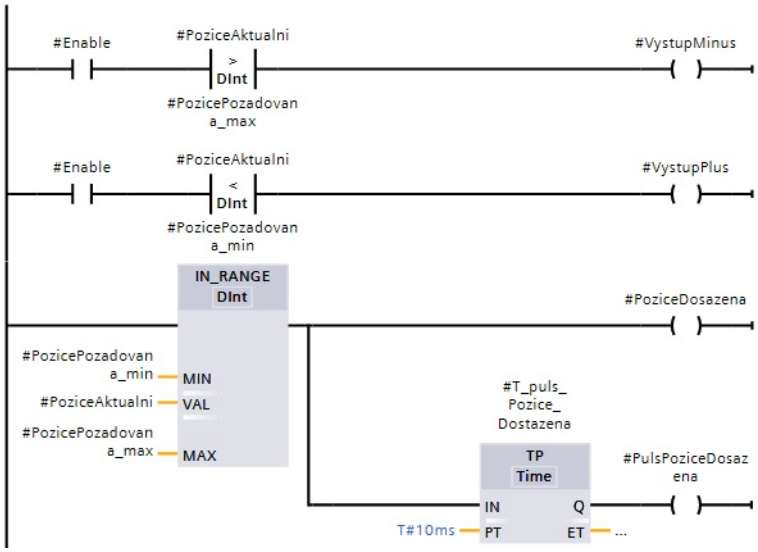
Obrázek 68: V kroku 131 hlídáme pozici X, v případě přesahu nastavíme znovu 1

Řízení horizontálního a vertikálního motoru manipulátoru

Vytvoříme si blok řízení motoru St03_AxisControl. Do bloku vstupují aktuální a požadovaná pozice a z bloku vystupují signály pro kladný (+) nebo záporný (-) pohyb motoru. Blok zjišťuje kde se manipulátor nachází a podle toho spustí daný motor. Také signalizuje, zda se již manipulátor nachází v požadované pozici.



Obrázek 69: Volání funkčního bloku AxisControl horizontálního motoru

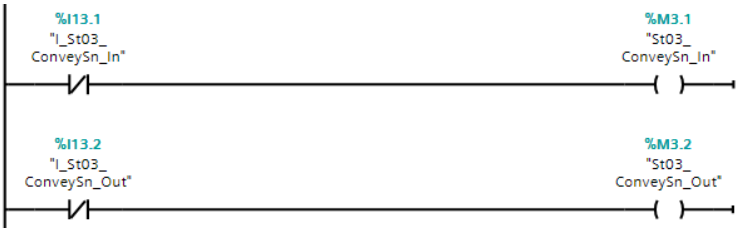


Obrázek 70: Výpis bloku AxisControl

Vstupy a výstupy automatizovaného skladu

Fotobraiéry

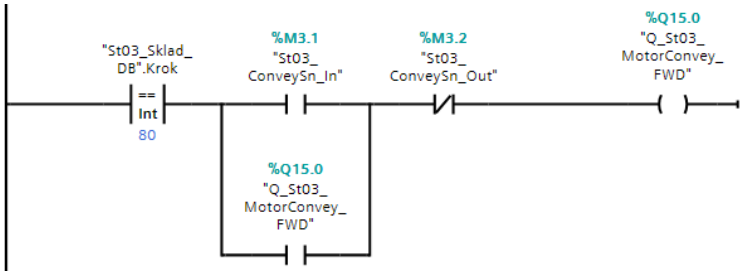
Můžeme obrátit logiku fotobariér (log. 0 = bez bedýnky, log. 1 = bedýnka).



Obrázek 71: Převrácená logika vstupů fotobariér

Motor dopravníku

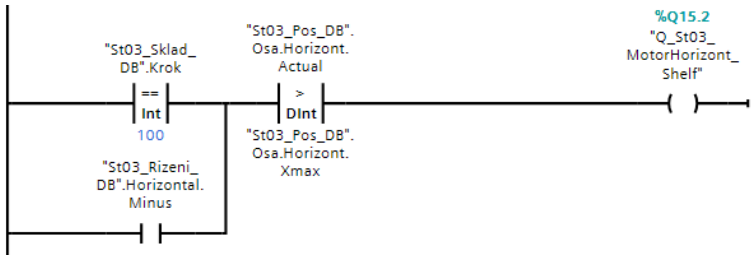
Dopravník jede, pokud se na něm opravdu nachází bedýnka. Aby jel dál i poté, co bedýnka odjede ze snímače, můžeme použít přídržný kontakt. Jakmile bedýnka dorazí na opačnou pozici, dopravník se zastaví.



Obrázek 72: Dopravník k vakuovému manipulátoru

Motor horizontální osy X a vertikální osy Y a motor podavače

Zápornou stranu osy X, resp. kladnou stranu osy Y, hlídáme vhodně změřenou koncovou polohou, jelikož na modelu se nenachází druhý koncový spínač.

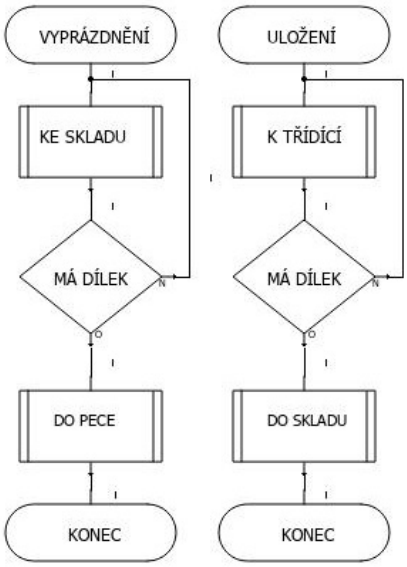


Obrázek 73: Hlídání polohy manipulátoru

3. 5. 5 Vakuový manipulátor

Funkce

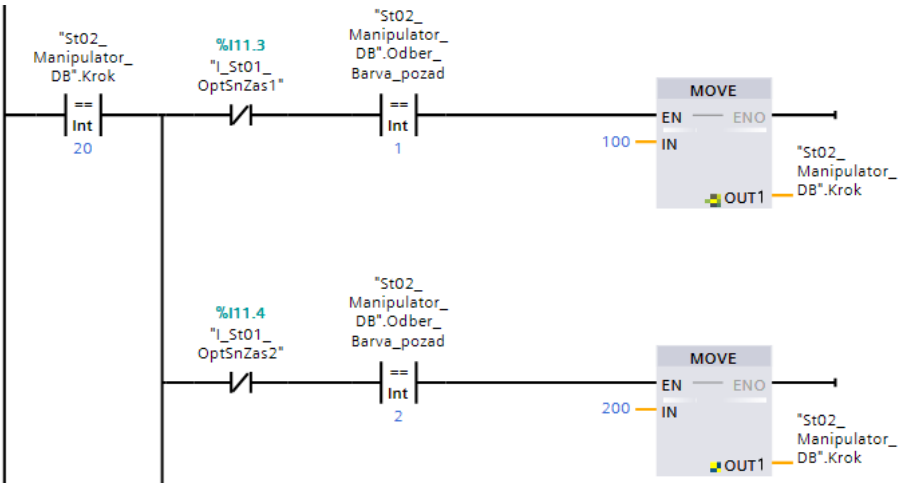
Manipulátor přenáší dílek ze skladu do pece a z třídící stanice zpět do skladu.
Řízení vertikální a horizontální osy, resp. měření pozice je analogické funkcím skladu a nebudeme se jím podrobně zabývat.
Pro řízení manipulátoru přibyla ještě osa rotace.
Ukážeme si pouze několik užitečných funkcí.



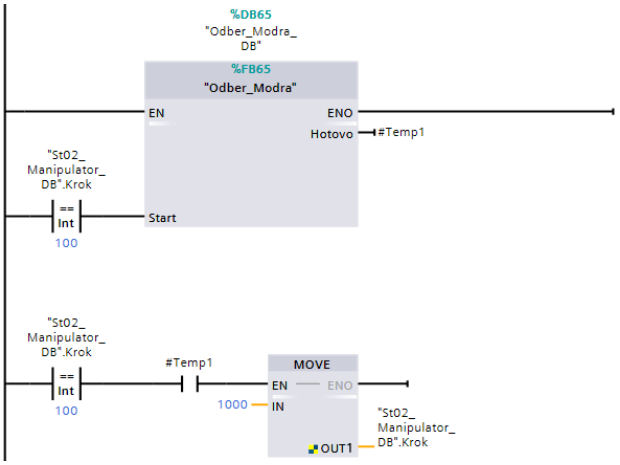
Obrázek 74: Vývojové diagramy pohybů manipulátoru, SW[2]

Přítomnost roztříděných dílků a jejich odběr

Barvě odpovídá číselná hodnota (modrá – 1, červená – 2, bílá – 3). Tato hodnota se nastaví podle přítomnosti dílku. Program postupně kontroluje přítomnost modré, červené a bílé. V případě nepřítomnosti jedné z barev kontroluje další v pořadí. Poté se krok nastaví podle požadované barvy. Odběr funguje stejně pro všechny tři barvy. Zavolá se příslušný blok. Bloky jsou shodné, liší se pouze hodnotami pozic. Manipulátor jede nejprve do pozice nad dílek, poté až k dílku a nakonec zpět nad dílek. V odběrové pozici zapneme vakuum.



Obrázek 75: Nastavení kroku manipulátoru podle požadované barvy



Obrázek 76: Volání bloku OdberModra

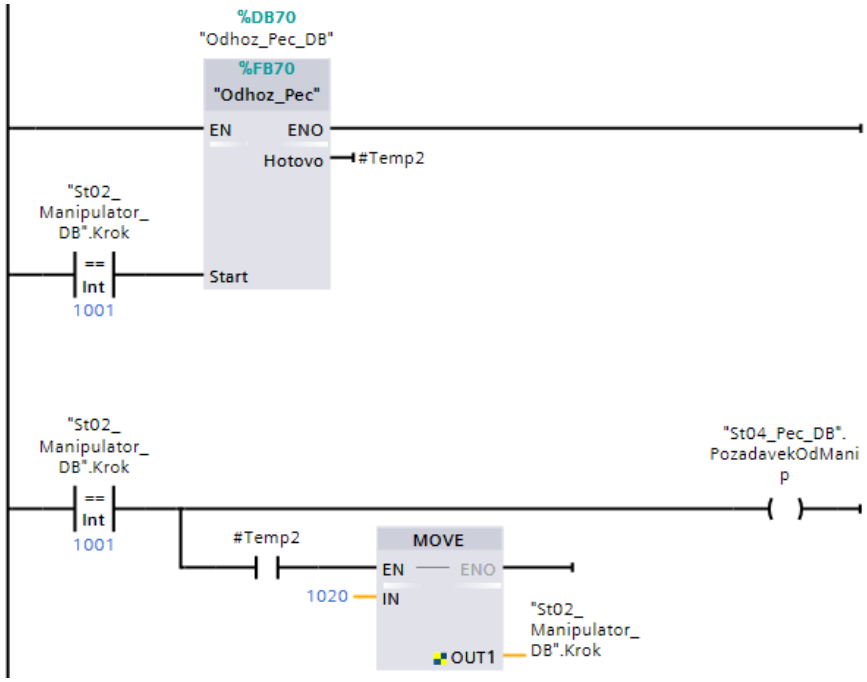
Odhození dílku na dopravník a odběr z něj

Manipulátor pracuje analogicky jako při odběru. Rozdíl je samozřejmě ve vypnutí vakua. Navíc čeká nad dopravníkem, než přijede bedýnka. Manipulátor také signalizuje, že dílek odhodil (signál PullConvey).

Při odběru manipulátor čeká než přijede bedýnka a signalizuje, že dílek odebral (signál OnManip). Znovu je většina kroků shodných jako při odhození.

Odhození dílku na podavač pece

V tomto kroku manipulátor generuje požadavek peci. Funguje analogicky jako odhození na dopravník.

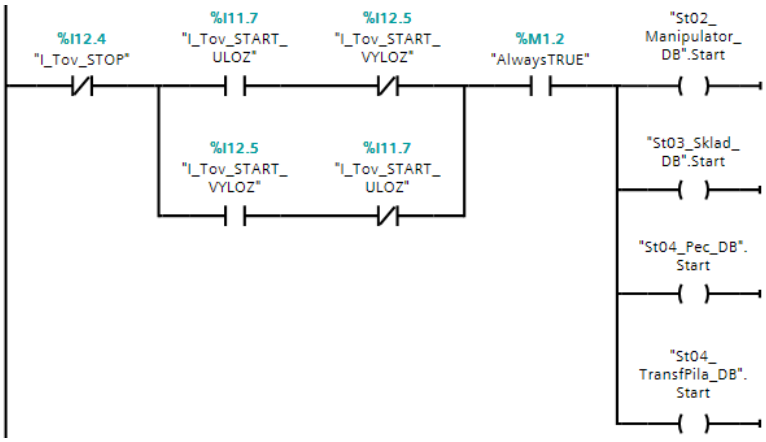


Obrázek 77: Volání bloku OdhozPec a generování signálu

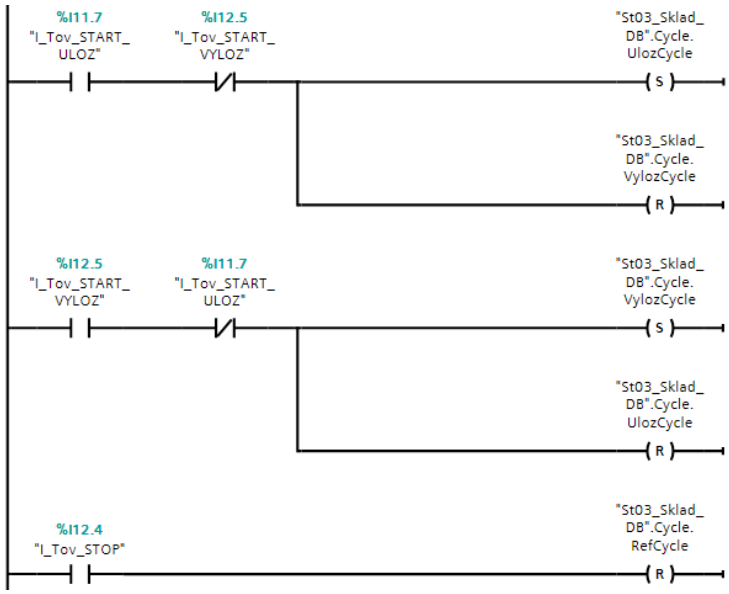
3. 5. 6 Řízení továrny tlačítky

Start

Stanice se spustí v daném cyklu. Zamezíme spuštění při stisku obou tlačítek a také chodu cyklů ULOŽ a VYLOŽ současně.

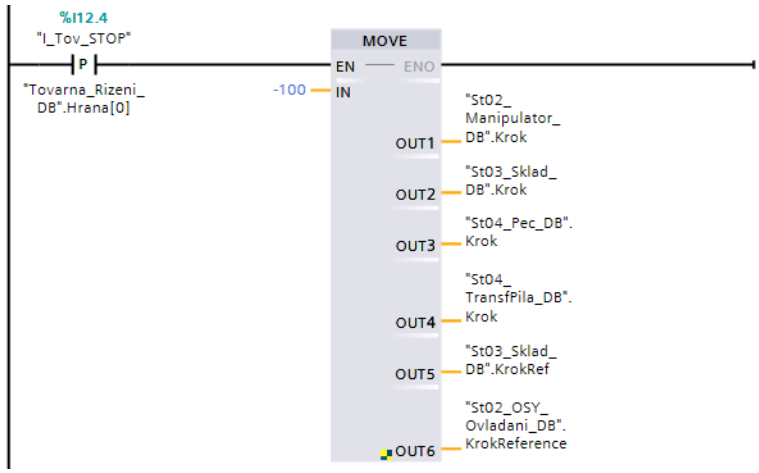


Obrázek 78: Spuštění továrny



Obrázek 79: Ulož vylož, STOP

Reset kroků



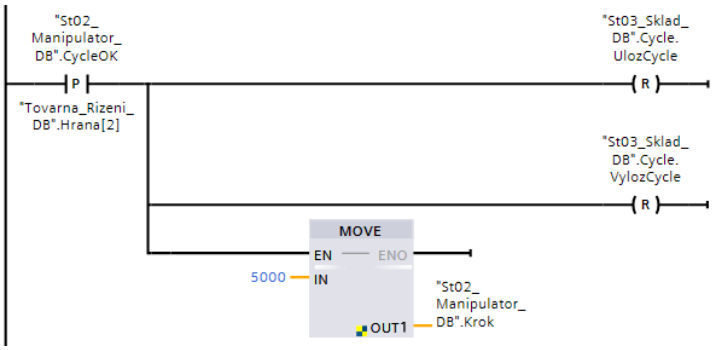
Obrázek 80: Reset kroků

Zamezení pohybu

Obrázek 81: Zamezení pohybu

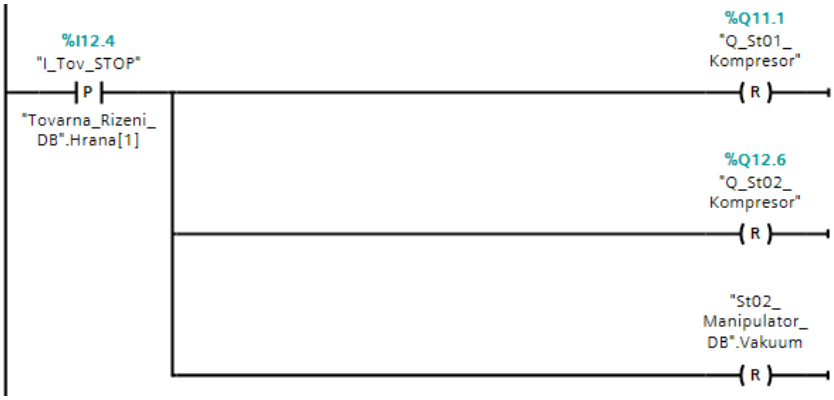
Cykly OK

Pokud manipulátor úspěšně dokončí svůj cyklus, resetujeme cykly ULOŽ a VYLOŽ a manipulátor nastavíme do pozice Home.



Obrázek 82: Reset cyklů

Vypnutí kompresorů



Obrázek 83: Vypnutí kompresorů

4. Závěr

Při vytváření projektu řízení modelu továrny se čtenář postupně seznamoval s principy programování PLC. Tyto vědomosti může posléze využít k vytvoření dalších stejně jednoduchých ale i složitějších projektů.

Takto zprovozněný model továrny a vytvořený projekt lze používat např. ve školících střediscích PLC programátorů, jelikož stále nabízí prostor pro vylepšení. Dále lze model využívat jako demonstraci automatizovaného procesu.

Do stávajícího projektu lze zařadit také čtení čárového kódu, které obstarává snímač na dopravníku skladu. Na každou bedýnku pro dílek lze nalepit čárový kód. Mohli bychom takto adresovat konkrétní bedýnku.

Dále se nabízí možnost připojit k PLC základní HMI panel a vhodně vizualizovat model. Jeho řízení se tak pro operátora stane přívětivějším.

Zdroje

[1] „Who is the Father of PLC and why was it invented“, Kevin Cope, 2018, https://realpars.com/father-of-the-plc/ [21-5-2021]

[2] Zelio Logic SR2/SR3, online katalog Schneider Electric, https://www.se.com/ww/en/product-range/531-zelio-logic-sr2-sr3/ [21-5-2021]

[3] Siemens S7 EAL Level 3 Unit (601/5326/ X Programmable Logic Controllers 3), Equinox Training Solutions, https://www.plc-training.co.uk/courses/eal-courses/siemens-s7-eal-level-3-unit/ [21-5-2021]

[4] Comparison of standard CPUs, SIEMENS, https://new.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-1500/cpus.html [21-5-2021]

[5] PM-MAINT - Maintenance Management System, SIEMENS, https://new.siemens.com/global/en/products/automation/industry-software/automation-software/scada/pm-add-ons.html [21-5-2021]

[6] „SIMATIC STEP 7 and WinCCEngineering V15.1, System manual“, Siemens, 2018, https://support.industry.siemens.com/cs/document/109755202/simatic-step-7-basic-professional-v15-1-and-simatic-wincc-v15-1?dti=0&lc=en-WW [21-5-2021]

[7] „Extended technical documentation, Factory Simulation 24V“, Fischer Technik, https://www.fischertechnik.de/en/products/simulating/training-models/536634-sim-factory-simulation-24v-simulation

Seznam použitého softwaru

[1] SIMATIC STEP 7 (TIA Portal) V14 – Siemens, všechny výpisy programu

[2] QElectroTech V 0.80c – kolektiv autorů, https://qelectrotech.org/

[3] MS Paint – Microsoft

Seznam příloh

tovarna.zap14 – archiv projektu TIA Portal V14, na CD

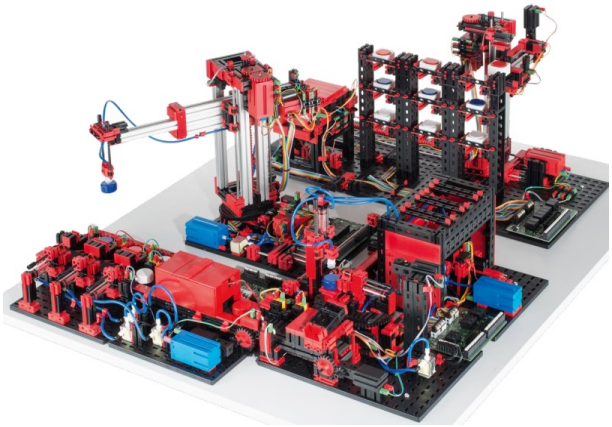
Příloha A – Popis modelu továrny

Příloha B – Hardwarová konfigurace PLC a připojení modelu k PLC

Příloha C – Tagy a datové bloky jednotlivých stanic

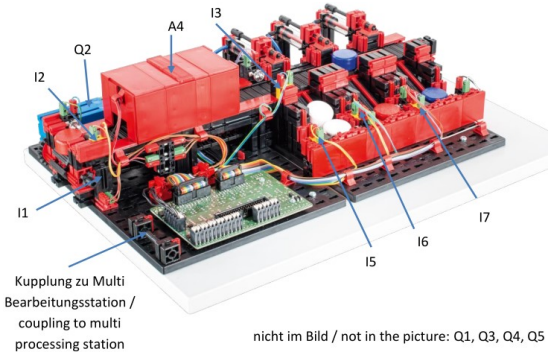
Příloha A – Popis modelu továrny

Výrobce: FischerTechnik
Název modelu: 536634 Factory Simulation 24 V



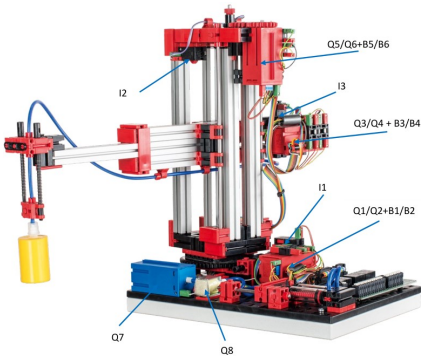
Obrázek 84: Model celé továrny, převzato z [7]

Model se skládá celkem ze čtyř stanic:
Třídící stanice: 536633



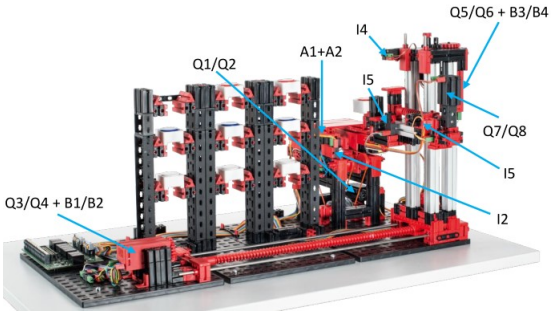
Obrázek 85: Třídící stanice, převzato z [7]

Vakuový manipulátor: 536630



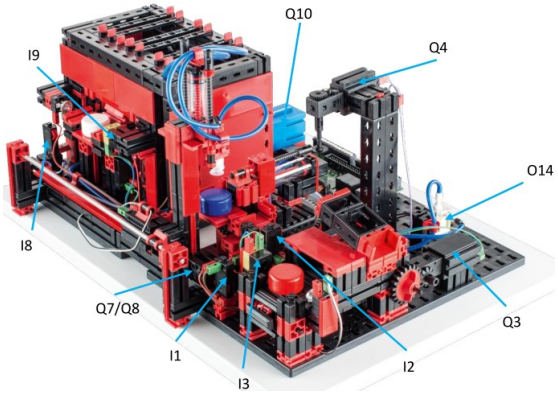
Obrázek 86: Vakuový manipulátor, převzato z [7]

Sklad: 536631



Obrázek 87: Sklad, převzato z [7]

Pec s pilou: 536632



Obrázek 88: Pec s pilou, převzato z [7]

Příloha B – Hardwarová konfigurace PLC a připojení modelu k PLC

KL1408										KL2408									
.0	.1	.0	.1	.0	.1	.0	.1	.0	.1	.0	.1	.0	.1	.0	.1	.0	.1	.0	.1
.2	.3	.2	.3	.2	.3	.2	.3	.2	.3	.2	.3	.2	.3	.2	.3	.2	.3	.2	.3
.4	.5	.4	.5	.4	.5	.4	.5	.4	.5	.4	.5	.4	.5	.4	.5	.4	.5	.4	.5
.6	.7	.6	.7	.6	.7	.6	.7	.6	.7	.6	.7	.6	.7	.6	.7	.6	.7	.6	.7
I11		I12		I13		I14		I15		Q11		Q12		Q13		Q14		Q15	

Obrázek 89: Adresace pro KL1408 a KL2408, SW [3]

Device overview

Module	Slot	I address	Q address	Type
PLC_1	1			CPU 1215C DC/DC/DC
DI 14/DQ 10_1	1 1	0...1	0...1	DI 14/DQ 10
AI 2/AQ 2_1	1 2	64...67	64...67	AI 2/AQ 2
	1 3			
PoziceVertikal	1 16	1000...1003		HSC
PoziceHorizontal	1 17	1004...1007		HSC
PoziceRotace	1 18	1008...1011		HSC
St03_Enc_Horizontal	1 19	1012...1015		HSC
St03_Enc_Vertical	1 20	1016...1019		HSC
HSC_6	1 21	1020...1023		HSC
Pulse_1	1 32		1000...1001	Pulse generator (PTO/P...
Pulse_2	1 33		1002...1003	Pulse generator (PTO/P...
Pulse_3	1 34		1004...1005	Pulse generator (PTO/P...
Pulse_4	1 35		1006...1007	Pulse generator (PTO/P...
PROFINET interface_1	1 X1			PROFINET interface

Obrázek 90: Přehled PLC

Device overview

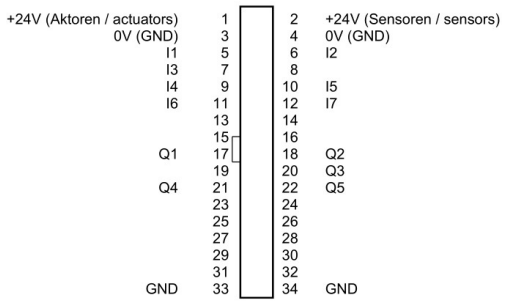
Module	Rack	Slot	I address	Q address	Type
bk9053	0	0	68...71	68...71	BK9053...
BK9053 V2.3 (at least FW...	0	0 X1			bk9053
Port 1	0	0 X1 P1			Port 1
Kx1xx8_1	0	1	11		Kx1xx8
Kx1xx8_2	0	2	12		Kx1xx8
Kx1xx8_3	0	3	13		Kx1xx8
Kx1xx8_4	0	4	14		Kx1xx8
Kx1xx8_5	0	5	15		Kx1xx8
Kx2xx8_1	0	6		11	Kx2xx8
Kx2xx8_2	0	7		12	Kx2xx8
Kx2xx8_3	0	8		13	Kx2xx8
Kx2xx8_4	0	9		14	Kx2xx8
Kx2xx8_5	0	10		15	Kx2xx8

Obrázek 91: Přehled rozšiřující I/O

Obrázek 92: Propojení PLC a rozšiřující I/O

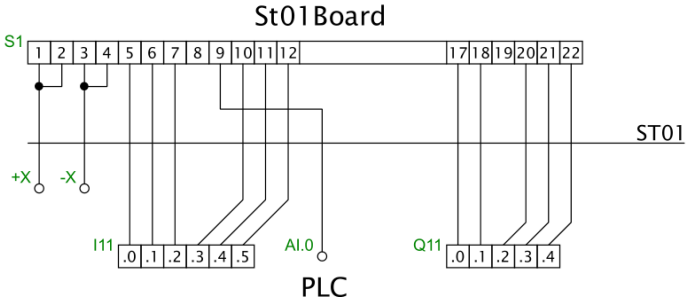
Stanice 1 -Třídící stanice

Zapojení na desce modelu



Obrázek 93: Konektor modelu třídící stanice, převzato z [7]

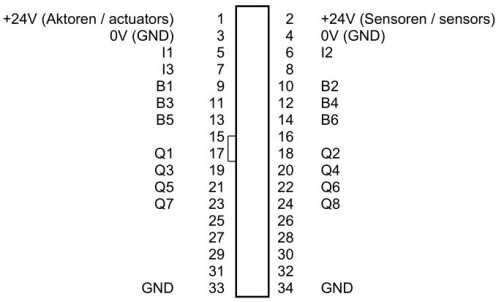
Připojení k PLC



Obrázek 94: Schema připojení třídící stanice, SW[2]

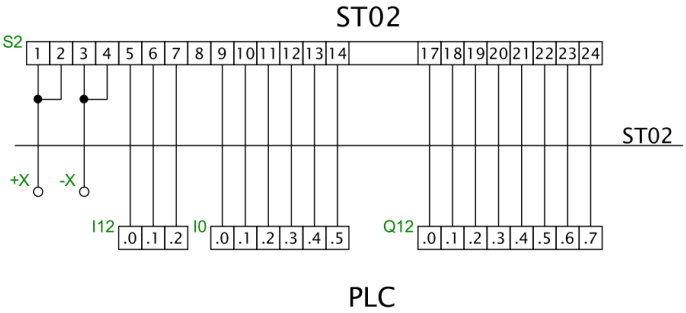
Stanice 2 – Vakuový manipulátor

Zapojení na desce modelu



Obrázek 95: Konektor modelu vakuového manipulátoru, převzato z [7]

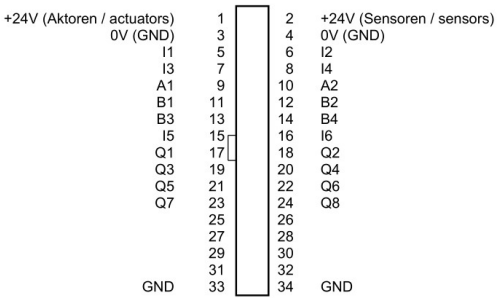
Připojení k PLC



Obrázek 96: Schema připojení vakuového manipulátoru, SW[2]

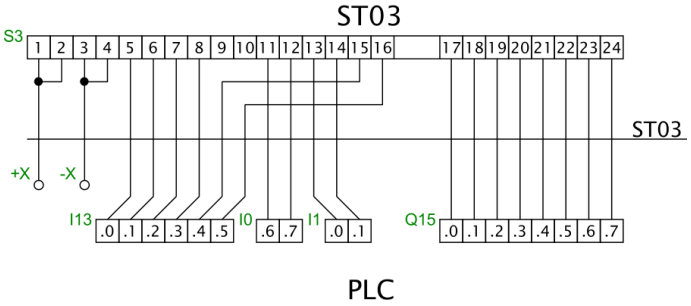
Stanice 3 – Sklad

Zapojení na desce modelu



Obrázek 97: Konektor modelu skladu, převzato z [7]

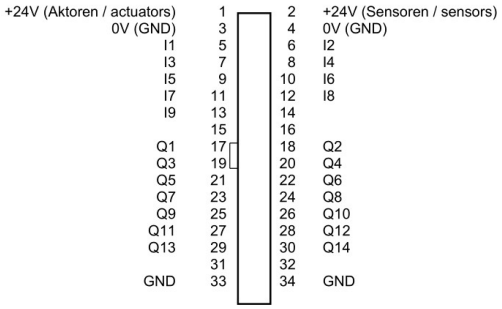
Připojení k PLC



Obrázek 98: Schema připojení skladu, SW[2]

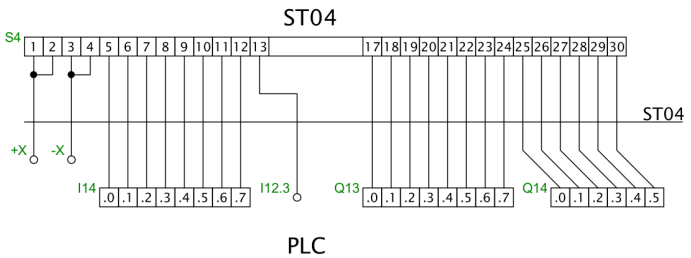
Stanice 4 – Pec s pilou

Zapojení na desce modelu



Obrázek 99: Konektor modelu pece s pilou, převzato z [7]

Připojení k PLC



Obrázek 100: Schema připojení pece s pilou, SW[2]

Zapojení tlačítek

Funkce tlačítka	**Adresa vstupu PLC**
Referování skladu	I11.6
Uložení do skladu	I11.7
STOP	I12.4
Vyložení ze skladu	I12.5
Referování manipulátoru	I12.6

Příloha C – Tagy a datové bloky jednotlivých stanic

Třídící stanice

Tagy

Název	Dat. typ	Adresa	Komentář
I_St01_Pulz_citac_dopravnik	Bool	%I11.0	Pulzní čítač dopravníku
I_St01_OptSnVstup	Bool	%I11.1	Přítomnost díu před snímáním barvy
I_St01_OptSnStred	Bool	%I11.2	Přítomnost díu za snímáním barvy
I_St01_OptSnZas1	Bool	%I11.3	Přítomnost dílu v zásobníku 1
I_St01_OptSnZas2	Bool	%I11.4	Přítomnost dílu v zásobníku 2
I_St01_OptSnZas3	Bool	%I11.5	Přítomnost dílu v zásobníku 3
Q_St01_MotorDopravnik	Bool	%Q11.0	Motor dopravníku
Q_St01_Kompresor	Bool	%Q11.1	Kompresor
Q_St01_PistZasobnik1	Bool	%Q11.2	Píst zásobníku 1
Q_St01_PistZasobnik2	Bool	%Q11.3	Píst zásobníku 2
Q_St01_PistZasobnik3	Bool	%Q11.4	Píst zásobníku 3
Analog_SnimacBarvy	Word	%IW64	Snímač barvy

Datový blok

Název	Dat. typ
HranaP_Sn_Vstup	Bool
HranaN_Sn_Vstup	Bool
HranaP_Sn_Stred	Bool
HranaN_Sn_Stred	Bool
Max_Cas_Chodu_Dopr	IEC_TIMER
Cas_mereni_Barvy	IEC_TIMER
Analog_Barva_INT	Int
Analog_Barva_Pomoc	Real
Analog_Barva_Real	Real
Barva_mimo_rozsah	Bool
Modra	Bool
Cervena	Bool
Bila	Bool
Barva_Cislo	Int
Pocet_incrementu_1	Int
Pocet_incrementu_2	Int
Hrana1_Snimac_Pulzu	Bool
Hrana2_Snimac_Pulzu	Bool
Hrana_Cas_Mereni	Bool

Pec s pilou

Tagy

Název	Dat. typ	Adresa	Komentář
I_St04_StulPozTransf	Bool	%I14.0	Stůl - pozice transfer
I_St04_StulPozDopravnik	Bool	%I14.1	Stůl - pozice dopravník
I_St04_OptSnKonec_Dopr	Bool	%I14.2	Snímač - dopravník
I_St04_StulPoz_Pila	Bool	%I14.3	Stůl - pozice pila
I_St04_TransfPozStul	Bool	%I14.4	Transfer - pozice stůl
I_St04_PlnicPeceUvnitr	Bool	%I14.5	Plnič pece – uvnitř
I_St04_PlnicPeceVenku	Bool	%I14.6	Plnič pece - venku
I_St04_TransfPozPec	Bool	%I14.7	Transfer - pozice pec
I_St04_OptSnPlnic	Bool	%I12.3	Snímač plniče - dílek na plniči
Q_St04_MotorRotStul_SmerDopravnik	Bool	%Q13.0	Motor stolu - k dopravníku
Q_St04_MotorRotStul_SmerTransfer	Bool	%Q13.1	Motor stolu - k transferu
Q_St04_MotorDopravnikVpred	Bool	%Q13.2	Motor dopravníku
Q_St04_MotorPila	Bool	%Q13.3	Motor pily
Q_St04_MotorPlnic_SmerDoPece	Bool	%Q13.4	Motor plniče - do pece
Q_St04_MotorPlnic_SmerZPece	Bool	%Q13.5	Motor plniče - z pece
Q_St04_MotorTransfer_SmerPec	Bool	%Q13.6	Motor transferu - k peci
Q_St04_MotorTransfer_SmerStul	Bool	%Q13.7	Motor transferu - ke stolu
Q_St04_TopeniPec	Bool	%Q14.0	Topení
Q_St04_Kompresor	Bool	%Q14.1	Kompresor
Q_St04_VentilVakuum	Bool	%Q14.2	Vakuum
Q_St04_VentilTransfer	Bool	%Q14.3	Píst - transfer
Q_St04_VentilPecDvere	Bool	%Q14.4	Píst – dveře pece
Q_St04_VentilRotStul	Bool	%Q14.5	Píst – vyhazovač

Datový blok – Pec

Název	Dat. typ
Start	Bool
Krok	Int
PozadavekOdManip	Bool
DilZalozenOdManip	Bool
PripravenaProVlozeniManip	Bool
DilPripravenProTransfer	Bool
DilOdebranOdTransfer	Bool
hrana	Array[0..5] of Bool
Cas_ohrevu	IEC_TIMER

Datový blok – Pila s transferem

Název	Dat. typ
Start	Bool
Krok	Int
hrana	Array[0..5] of Bool

Datový blok – Časovače

Název	Dat. typ
Cas_DvereOtevreny	IEC_TIMER
Cas_DvereZavreny	IEC_TIMER
Cas_DilVysunout	IEC_TIMER
Cas_DopravnikVpred	IEC_TIMER
Cas_UchopovacDolu	IEC_TIMER
Cas_UchopovacNahoru	IEC_TIMER
Cas_VakuumTransferON	IEC_TIMER
Cas_VakuumTransferOFF	IEC_TIMER
Cas_PilaChod	IEC_TIMER

Sklad

Tagy

Název	Dat. typ	Adresa	Komentář
I_St03_Ref_Button	Bool	%I11.6	Button - Reference
I_St03_RefSn_Horizont	Bool	%I13.0	Snímač - Horizont - REF
I_St03_ConveySn_In	Bool	%I13.1	Snímač - Dopravník - Sklad
I_St03_ConveySn_Out	Bool	%I13.2	Snímač - Dopravník - Gripper
I_St03_RefSn_Vertical	Bool	%I13.3	Snímač - Vertikal - REF
I_St03_CantilevSn_FWD	Bool	%I13.4	Snímač - Podavač – Vyjetý
I_St03_CantilevRefSn_BWD	Bool	%I13.5	Snímač - Podavač – Zajetý
I_St03_Pos_Horizont	DInt	%ID1012	Enkodér - Horizont
I_St03_Pos_Vertical	DInt	%ID1016	Enkodér - Vertikal
Q_St03_MotorConvey_FWD	Bool	%Q15.0	Motor - Dopravník - Gripper
Q_St03_MotorConvey_BWD	Bool	%Q15.1	Motor - Dopravník - Sklad
Q_St03_MotorHorizont_Shelf	Bool	%Q15.2	Motor - Horizont - Sklad
Q_St03_MotorHorizont_Convey	Bool	%Q15.3	Motor - Horizont - Dopravník
Q_St03_MotorVertical_Down	Bool	%Q15.4	Motor - Vertikal – Dolů
Q_St03_MotorVertical_Up	Bool	%Q15.5	Motor - Vertikal - Nahoru
Q_St03_MotorCantilev_FWD	Bool	%Q15.6	Motor - Podavač – Vpřed
Q_St03_MotorCantilev_BWD	Bool	%Q15.7	Motor - Podavač - Vzad
St03_ConveySn_In	Bool	%M3.1	Snímač - Dopravník - Sklad - LOG
St03_ConveySn_Out	Bool	%M3.2	Snímač - Dopravník - Gripper - LOG

Datový blok – AxisControl

Název	Dat. typ
Input	
- Enable	Bool
- PoziceAktualni	DInt
- PozicePozadovana_max	DInt
- PozicePozadovana_min	DInt
Output	
- VystupPlus	Bool
- VystupMinus	Bool
- PoziceDosazena	Bool
- PulsPoziceDosazena	Bool
Static	
- Pozadavek_max	DInt
- Pozadavek_min	DInt
T_puls_Pozice_Dosazena	IEC_TIMER

Vakuový manipulátor

Tagy

Název	Dat. typ	Adresa	Komentář
I_St02_RefSnimac_Vertikal	Bool	%I12.0	Snímač - Vertikal - REF
I_St02_RefSnimac_Horizontal	Bool	%I12.1	Snímač - Horizont - REF
I_St02_RefSnimac_Rotace	Bool	%I12.2	Snímač - Rotace - REF
I_St02_StartRef	Bool	%I12.6	Start REF
I_St02_Pozice_Vertikal	DInt	%ID1000	Enkodér - Vertikal
I_St02_Pozice_Horizontal	DInt	%ID1004	Enkodér - Horizontal
I_St02_Pozice_Rotace	DInt	%ID1008	Enkodér - Rotace
Q_St02_Motor_Vertikal_UP	Bool	%Q12.0	Motor - Vertikal - Nahoru
Q_St02_Motor_Vertikal_DN	Bool	%Q12.1	Motor - Vertikal – Dolů
Q_St02_Motor_Horizont_FWD	Bool	%Q12.2	Motor - Horizont – Vpřed
Q_St02_Motor_Horizont_BWD	Bool	%Q12.3	Motor - Horizont - Vzad
Q_St02_Motor_Rotace_CW	Bool	%Q12.4	Motor - Rotace - Po směru
Q_St02_Motor_Rotace_CCW	Bool	%Q12.5	Motor - Rotace - Proti směru
Q_St02_Kompresor	Bool	%Q12.6	Kompresor
Q_St02_Vakuum	Bool	%Q12.7	Vakuum

Datový blok - Manipulátor

Název	**Dat. typ**
Start	Bool
Krok	Int
MoveEnableAll	Bool
PoziceDosazenaAll	Bool
CycleOK	Bool
Odber_Barva_pozad	Int
Hrana	Array[0..50] of Bool
Vakuum	Bool
T_Konec	IEC_TIMER