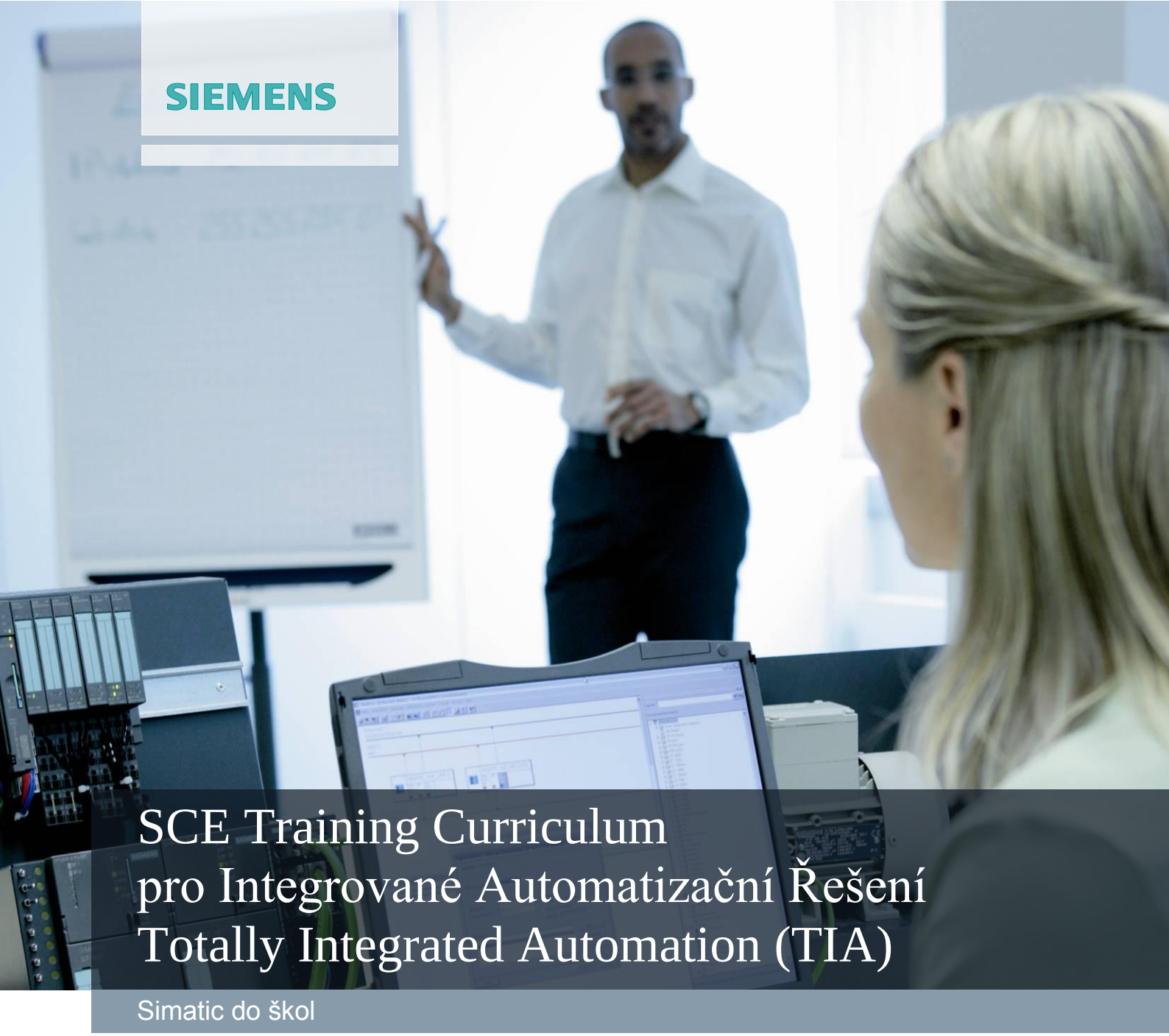


SCE Training Curriculum pro Integrované Automatizační Řešení Totally Integrated Automation (TIA)

Simatic do škol

TIA Portal Module 010-020

Blokové typy pro SIMATIC S7-1200

Cooperates with Education

Automation

SIEMENS

SCE tréninkové balíčky pro tyto manuály

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
  Order No.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
  Order No.: 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
  Order No.: 6ES7822-0AA01-4YE0

Přehled nabízených balíčků najdete na adrese: siemens.com/sce/tp

Další možnosti

Pro regionální Siemens SCE trénink kontaktujte osobu z vašeho regionu:
siemens.com/sce/contact

Další informace o SCE

siemens.com/sce

Informace o používání

Tento SCE training curriculum pro integrované automatizační řešení Totally Integrated Automation (TIA) byl připraven pro program "Siemens Automation Cooperates with Education (SCE)“ specielně pro výcvikové účely pro veřejnost a R&D. Siemens AG negarantuje obsah.

Tento dokument je pro úvodní trénink pro Siemens produkty/systémy; i.e., může být kopírován celý nebo po částech a předán těm kteří se zrovna zacvičují. Předávání a kopírování dokumentu stejně tak jako sdílení jeho obsahu je povoleno pro výcvikové účely.

Výjimky je třeba si vyžádat písemně od kontaktní osoby ze Siemens AG: Roland Scheuerer roland.scheuerer@siemens.com.

Porušení pravidel bude hnáno k zodpovědnosti. Všechna práva včetně překladu jsou rezervována, především při udílení patentů nebo registraci designu.

Použití pro kurzy průmyslových zákazníků je přímo zakázáno. Nechceme, aby byl dokument využít komerčně.

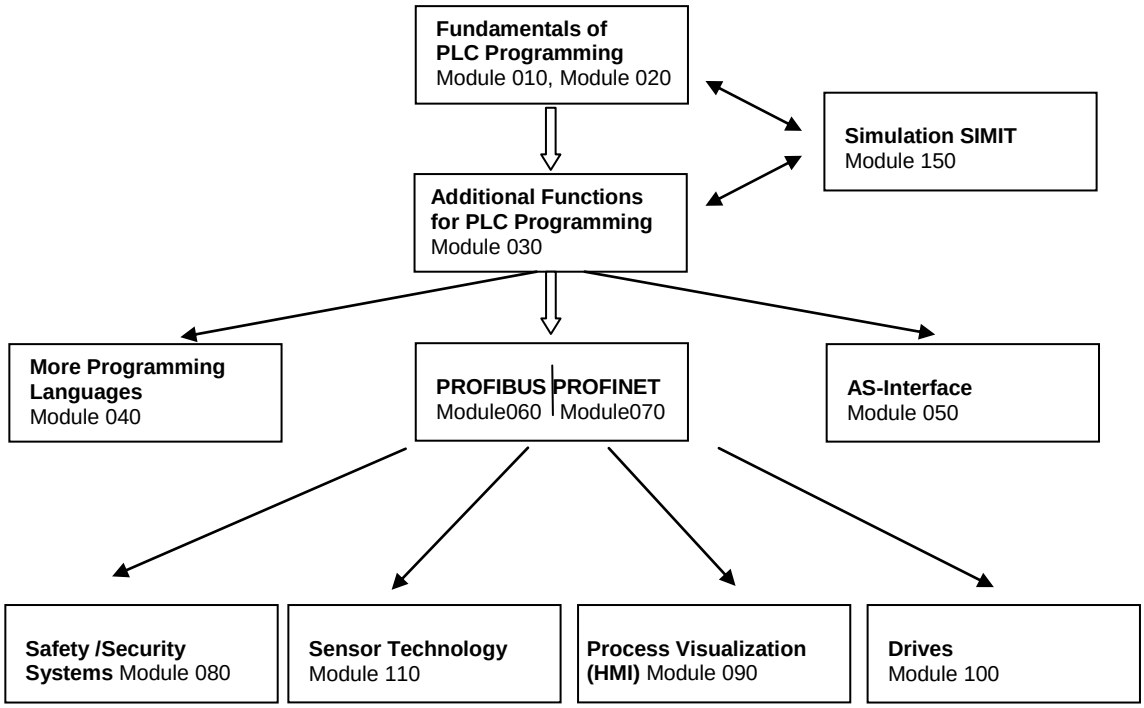
Děkujeme mnohokrát Michael Dziallas Engineering Corporation a všem ostatním zaintersovyným osobám za přípravu tohoto dokumentu.

PAGE

Obsah

1. Úvod

Vzhledem ke svému obsahu je modul SCE_CZ_010-020 součástí jednotky ‚**Základy PLC programování**' a je rychlým vstupním bodem pro programování SIMATIC S7-1200 pomocí TIA portálu.



Cíle tréninku:

V modulu SCE_CZ_010-020, se čtenář seznámí s různými bloky používanými pro programování SIMATIC S7-1200 v nástroji TIA portal. Modul vysvětluje různé blokové typy, a ukazuje v krocích níže uvedených jak generovat program pomocí funkčních bloků.

- Generování funkčního bloku
- Definování interních proměnných
- Programování pomocí interních proměnných v bloku
- Volání a parametrizace funkčního bloku v OB1

Prerekvizity:

Pro úspěšné absolvování této lekce je třeba znát:

- Práci s Windows operačním systémem
- Základy programování PLC s TIA portal

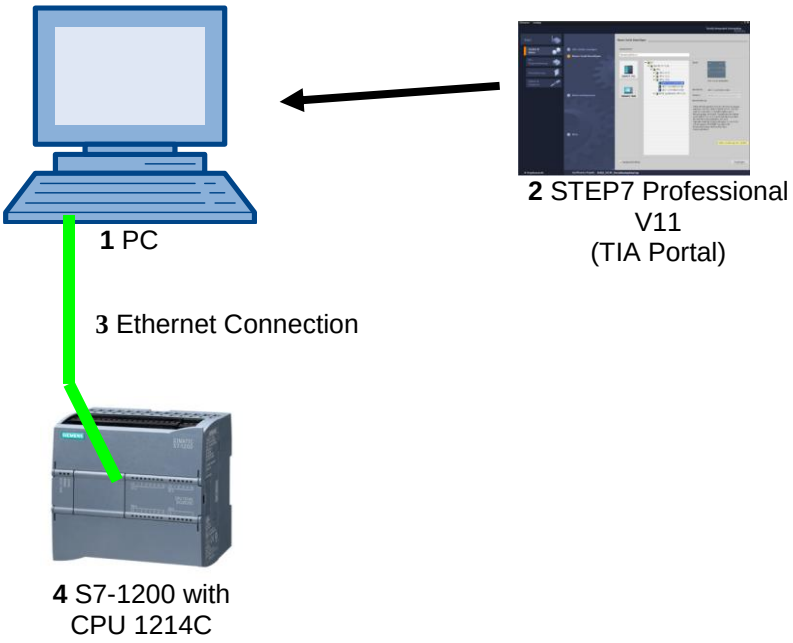
Vyžadovaný hardware a software:

1 PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB operating system Windows XP Professional SP3/Windows 7, Professional/Windows 7 Enterprise/Windows 7 Ultimate/Windows 2003 Server R2/Windows Server 2008 Premium SP1, Business SP1, Ultimate SP

2 Software STEP7 Professional V11 SP1 (Totally Integrated Automation (TIA) Portal V11)

3 Ethernet connection between PC and CPU 315F-2 PN/DP

4 PLC SIMATIC S7-1200; for example, CPU 1214C.
The inputs have to be brought out to a panel.



2. Poznámky k programování SIMATIC S7-1200

2.1 Automatizační systém SIMATIC S7-1200

Automatizační systém SIMATIC S7-1200 je modulární mikrokontrolér pro menší a střední automatizační úlohy.
Rozsáhlé spektrum modulů je k dostání pro optimální adaptaci na danou automatizační úlohu.
Kontrolér S7 se sestává ze zdrojového modulu, samotné řídící jednotky CPU a vstupně/výstupních modulů pro digitální a analogové signály.
Pokud je třeba, komunikační procesor a funkční moduly je možné upravit pro speciální úlohy jako třeba řízení krokového motoru.

S S7 programem, programovatelný logický automat (PLC) sleduje a řídí zařízení nebo proces, kde IO moduly jsou zapsány v S7 programu jako %I pro vstupy a %Q pro výstupy.

Celý systém je programován softwarem STEP 7.

2.2 Programovací software STEP 7 professional 11 (TIA portal V11)

Software STEP 7 Professional V11 (TIA Portal V11) je programovací nástroj pro následující automatizační systémy.

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

Se STEP 7 Professional V11, mohou být navrženy následující funkce pro automatizaci provozu:

- Konfigurace a parametrizace hardwaru
- Definování komunikace
- Programování
- Testování, prověření a servis s provozními/diagnostickými funkcemi
- Dokumentace
- Vytváření displejů pro SIMATIC basic panely s integrovaným WinCC Basic
- S doplňkovými WinCC balíčky, můžete připravit zobrazovací řešení pro PCs a ostatní panely

Všechny funkce jsou připraveny spolu s detailní online nápovědou.

3. Blokové typy pro SIMATIC S7-1200

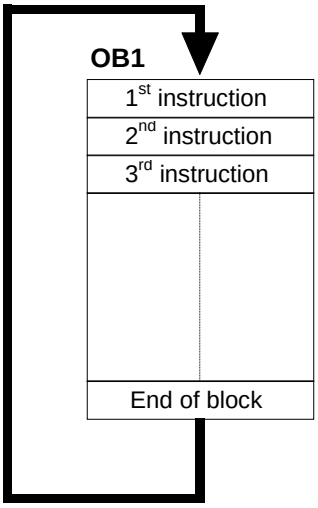
Pro SIMATIC S7-1200 je psán program v takzvaných blocích. V rámci standardizace je blok MAIN[OB1] vygenerován automaticky. Je to rozhraní do operačního systému CPU, je volán automaticky a prováděn v cyklech.

Pokud je automatizační úloha rozsáhlá, dělíme program do menších bloků, které jsou rozděleny podle funkce, a je snadné jim porozumět. Tyto bloky se nazývají organizační.

Na konci každého takového bloku se vracíme zpět do bloku, který ho vyvolal ihned na následující řádek za vyvolání.

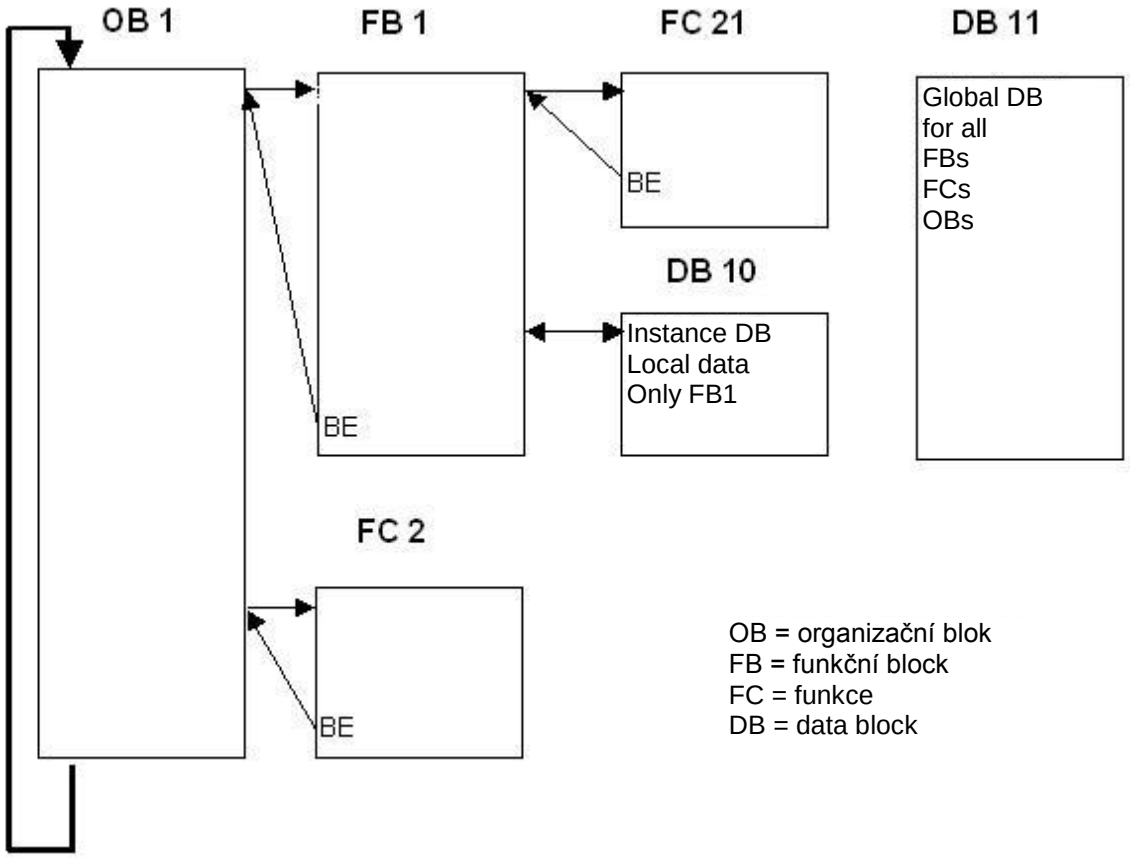
3.1 Lineární programování

Pro lineární programování, instrukce jsou uloženy v blocích a zpracovány v sekvenci v jaké jsou uloženy v programové paměti. Když program dojde na poslední krok, konec bloku, rozeběhne se zase od počátku.
Nazýváme to cyklické zpracování.
Čas, jaký zařízení potřebuje ke zpracování všech instrukcí, se nazývá čas cyklu.
Lineární programování je většinou používáno pro jednoduché kontrolní úlohy, které nejsou tolik rozsáhlé. Můžeme ho implementovat v jednom OB.



3.2 Strukturované programování

Pokud je řídící úloha moc rozsáhlá, rozdělíme programové bloky na menší rozdělené podle funkcí, které jsou snadno srozumitelné. Výhoda: můžeme program zkoušet po částech, a pokud fungují, můžeme je spojit do výsledné funkce.
Hlavní blok musí volat programové bloky. Pokud dojde proces na konec bloku (BE), program pokračuje ve zpracovávání následujícího volaného bloku.



3.3 Uživatelské bloky pro SIMATIC S7-1200

Pro strukturované programování je možné využít následující uživatelské bloky:

- OB (organization block):

OB jsou volány v cyklech operačním systémem a jsou rozhraním mezi uživatelským programem a operačním systémem. Z OB jednotka PLC získává informace o volání následujících bloků a zpracovává je.

- FB (function block):

Pro každou instanci potřebuje FB přidělenou oblast v paměti. Při volání FB můžeme přidělit například data block DB jako jeho novou instanci.
Přístup k datům v instanci DB je pak pomocí proměnných v FB.
Pokud voláme FB víckrát musíme mu přidělit různé oblasti paměti.
FC a FB můžeme vyvolávat ve funkčním bloku za sebou.

- FC (function):

FC nemá přidělenou oblast paměti. Lokální data funkce jsou smazána poté, co je funkce provedena.
FC a FB můžeme vyvolávat ve funkčním bloku za sebou.

- DB (data block):

DB poskytují paměť pro proměnné. DB je rozdělen do dvou tipů: global DB které mohou využít všechny OB, FB, a FC a instanční DB které jsou přiděleny k určité FB.

Poznámka:
Pokud při programování FC a FB použijeme jen interní proměnné, můžeme je využít několikrát ve formě standardního bloku. Mohou být volány, kolikrát chceme. Nicméně FB musíme přidělit oblast paměti, tak zvanou instanci (kupříkladu DB) pro každé vyvolání.

3.3.1 Organizační bloky

Organizační bloky OB jsou rozhraním mezi operačním systémem a programem uživatele. Jsou volány operačním systémem a řídí následující procesy:

- Cyklické zpracování programu
- Zpracování alarmů
- Zpracování chyb

Můžete programovat, OB jak potřebujete a pomocí toho řídit chování CPU.

Máte spousty možností jak využít OB ve vašem programu:

• **Startup OB, Cycle OB, Timing Error OB a Diagnosis OB:**
Stačí jen vložit OB do vašeho projektu, nemusíte mu přidělovat parametry ani ho volat.

• **Process Alarm OB a Time Interrupt OB:**
Tyto organizační bloky musí být parametrizovány. Navíc process alarm OB může být přidělen k vykonávané události v čase s využitím instrukce ATTACH, nebo od ní oddělen pomocí instrukce DETACH

• **Time Delay Interrupt OB:**
Time delay interrupt OB je možné programovat. Navíc je třeba ho volat v uživatelském programu pomocí instrukce SRT_DINT. Není třeba ho parametrizovat.

Startovní informace
V případě, že je nějaký OB spuštěn, operační systém si přečte informace, které mohou být vyhodnoceny v uživatelském programu. To může velice pomoci pro diagnostiku chyb. Jestli a jaká informace byla vyčtena je uvedeno v popisu OB.

3.3.2 Funkce

Funkce obsahuje program, který je proveden, pokud je funkce vyvolána z jiného bloku.

Funkce (FC) je blok kódu bez přidělené paměti. Data dočasných proměnných jsou ztracena poté, co je funkce provedena. Globální data bloky je možné využít k uložení dat z FC.

Funkci můžeme využít například k:

- Vrácení funkčních hodnot bloku, který vyvolává funkci. Například výpočet matematické funkce.
- Zpracování technologických funkcí, například řízení pomocí individuálních binárních operací.

Funkci můžeme volat několikrát na různých místech programu. To ulehčuje programování složitějších opakujících se funkcí.

3.3.3 Funkční bloky

Funkční bloky obsahují podprogramy, které jsou provedeny vždy, když je FB vyvolán.

FB jsou bloky kódu, které ukládají hodnoty svých proměnných do instančních DB, takže jsou tyto hodnoty dosažitelné i poté co je blok zpracován.
Uložte své vstupy, výstupy a in/out parametry permanentně do instančních DB. Budou dostupné i poté co blok proběhne. Proto jsou také nazývány bloky s pamětí.

Funkční bloky používáme tehdy, když nám na úlohu již nestačí funkce:

- Pokaždé když potřebujeme timer nebo counter. (více v modulu M3)
- Vždy když chceme uložit informaci v programu. Například když vybíráme operační mód pomocí tlačítka.

Funkční blok můžeme také volat víckrát z různých míst v programu. To zjednodušuje programování složitých a opakujících se funkcí.

Instance funkčního bloku
Volání funkčního bloku se nazývá instance.
Pro každou instanci funkčního bloku je přidělena oblast paměti, ta obsahuje data FB pro zpracování. Paměť je poskytnuta pomocí DB, který je systémem vytvořen automaticky. Je možné využít paměť pro více instancí v jednom DB jako multi-instanci.

3.3.4 Data bloky

Oproti blokům s kódem, datové bloky neobsahují instrukce, ale používají se k uložení dat uživatele. To znamená, že datové bloky obsahují hodnoty proměnných, které používá program uživatele pro práci.

Global data blocks ukládá data, které mohou být využity všemi ostatními bloky. Maximální velikost DB je různá, záleží na CPU. Strukturu DB můžeme specifikovat, jak potřebujeme.

Příklady použití:

- Ukládání informací o stavu skladiště: „Kde je který produkt uložen.“
- Ukládání informací o určitém produktu.

Každý funkční blok, funkce nebo organizační blok může číst nebo zapisovat data z a do globálního DB. Tyto data zůstanou uchována v DB, i když ho opustíme.

Volání FB je nazýváno instance. Pro každé volání funkčního bloku s přenosem parametru, je na server přidělen **instance data block** jako datové úložiště. V něm jsou vlastní data a parametry z funkčního bloku uloženy.
Maximální velikost instančního data bloku se mění, záleží na CPU. Proměnné deklarované ve funkčním bloku určuje strukturu instančního DB.

Globální DB a instanční DB mohou být používány zároveň.

4. Vzorové funkční bloky pro kontrolu pásového dopravníku

Pokud chceme vytvořit bloky, které pracují v programu jako „černá skříňka“, musíme je programovat s použitím proměnných. V tomto případě budeme následovat tyto pravidla: žádné absolutní adresování vstupů a výstupů, flagy a podobně. V blocích pouze proměnné a konstanty.

V příkladu níže, vytvoříme funkční blok, který má deklarované proměnné pro řízení pásového dopravníku pomocí nastavení operátorského módu.

Tlačítkem ‚S1‘ nastavíme mód ‚Manual‘, tlačítkem ‚S2‘ nastavíme mód ‚Automatic‘.
V módu ‚Manual‘ běží motor pouze, pokud je stisknuté tlačítko ‚S3‘, ale není stisknuté tlačítko ‚S4‘.
V módu ‚Automatic‘ je motor zapnutý stiskem tlačítka ‚S3‘ vypne se stiskem tlačítka ‚S4‘(rozpojení obvodu).

Seznam úloh:

Adresa	Symbol	Komentář
%I 0.0	S1	Tlačítko pro mód Manual S1 NO
%I 0.1	S2	Tlačítko pro mód Automatic S2 NO
%I 0.2	S3	On tlačítko S3 NO
%I 0.3	S4	Off tlačítko S4 NC
%Q 0.2	M1	Motor dopravníku M1

Poznámka:
Off tlačítko S4 je rozpojení obvodu, aby byla zajištěna bezpečnost při přerušení vodiče. To znamená: pokud je zde přerušen obvod systém se automaticky zastaví. Jinak by nešel zastavit při přerušení obvodu. Z tohoto důvodu jsou v řídící technice všechny stop spínače, off tlačítka apod. navrženy jako rozpojení obvodu.

5. Programování řízení pásového dopravníku SIMATIC S7-1200

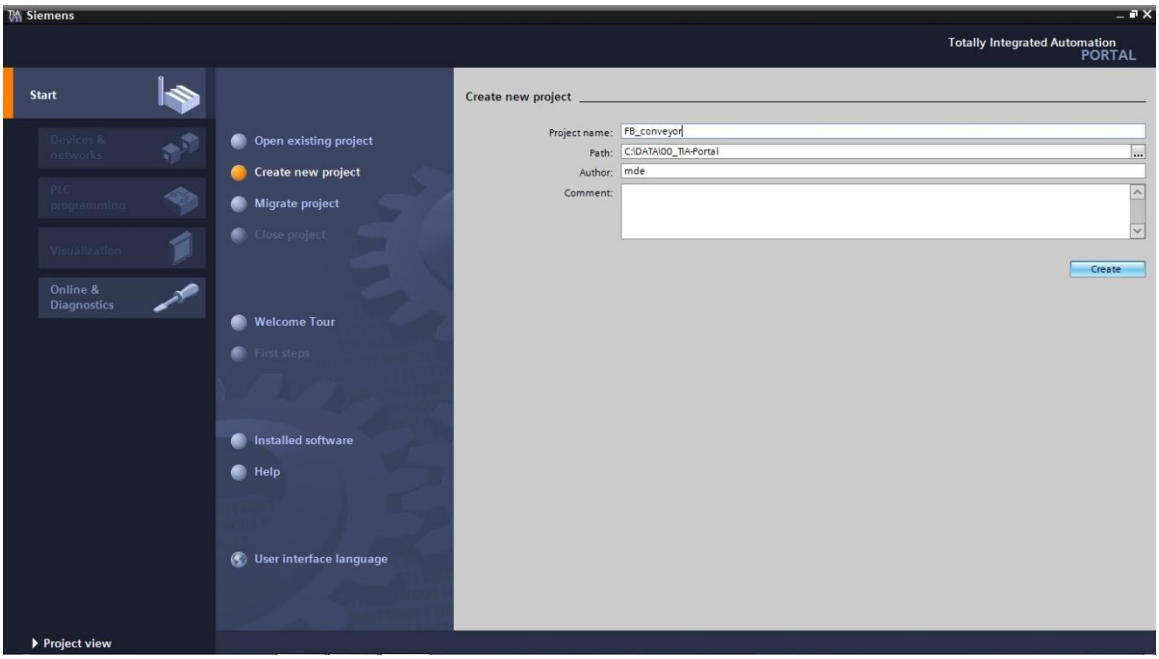
Projekt je vytvořen a programován v software ‘Totally **Integrated Automation Portal’**.

Zde pod společným rozhraním, jsou komponenty jako kontrolér, vizualizační panely a síťování automatizačních řešení nastaveny a programovány. Můžeme využít také online modu pro diagnostiku.

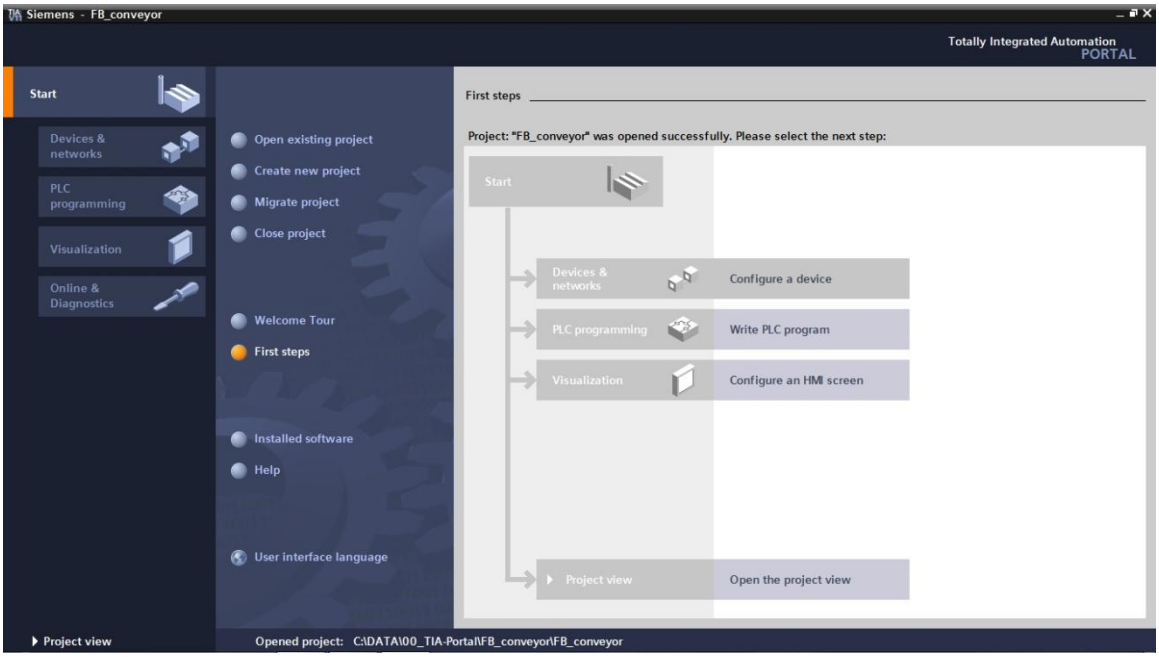
V krocích níže nastavíme projekt pro SIMATIC S7-1200 a naprogramujeme řešení úlohy:

1. Centrální nástroj je **’Totally Integrated Automation Portal’**, který spustíme dvojklikem (→ Totally Integrated Automation Portal V11)

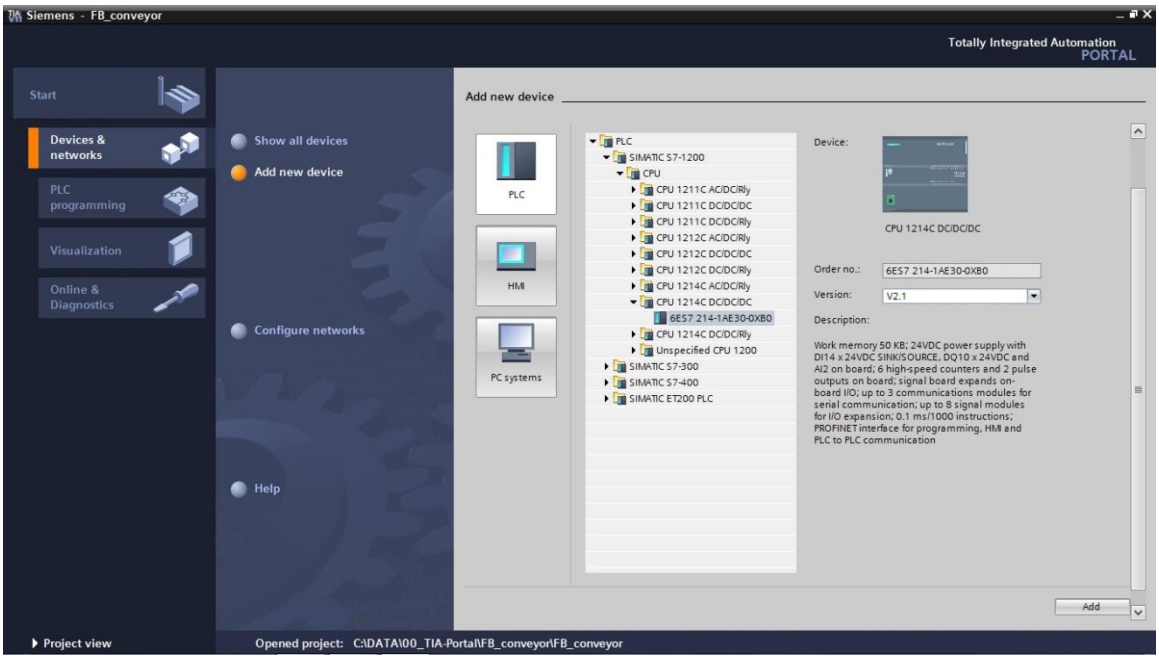
2. Program pro SIMATIC S7-1200 je spravován v portal view. (→ Create new project → FB_conveyor → Create)



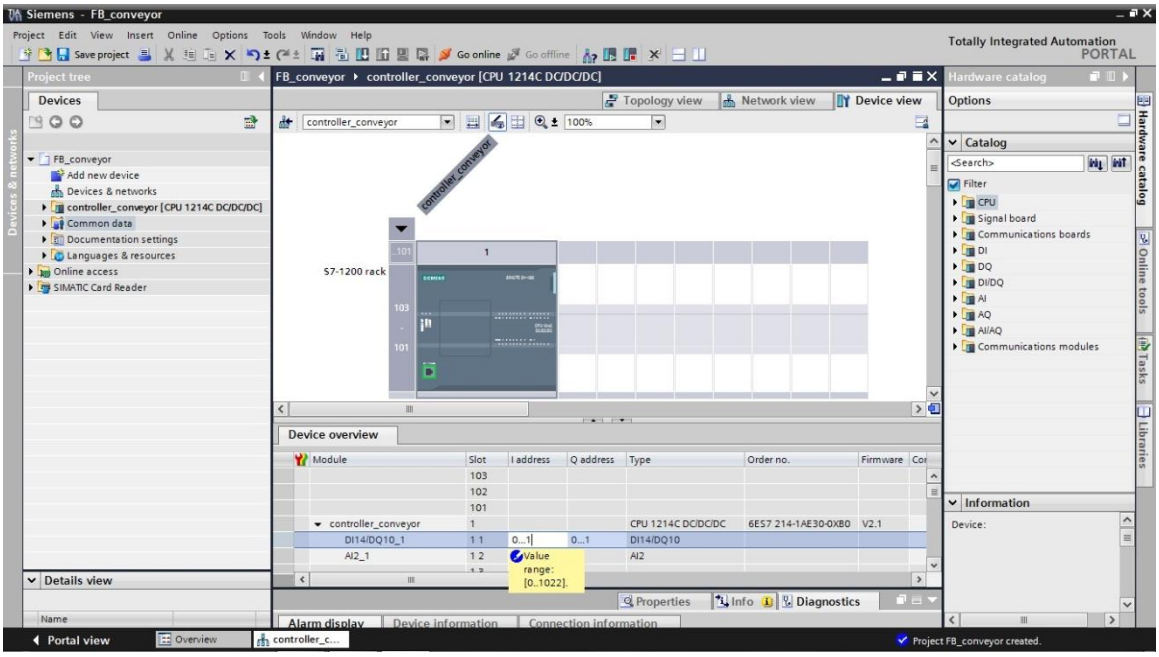
3. **'First Steps'** jsou navrženy s ohledem na konfiguraci. Nejdříve chceme **'Configure a device'.** (→ First Steps → Configure a device)



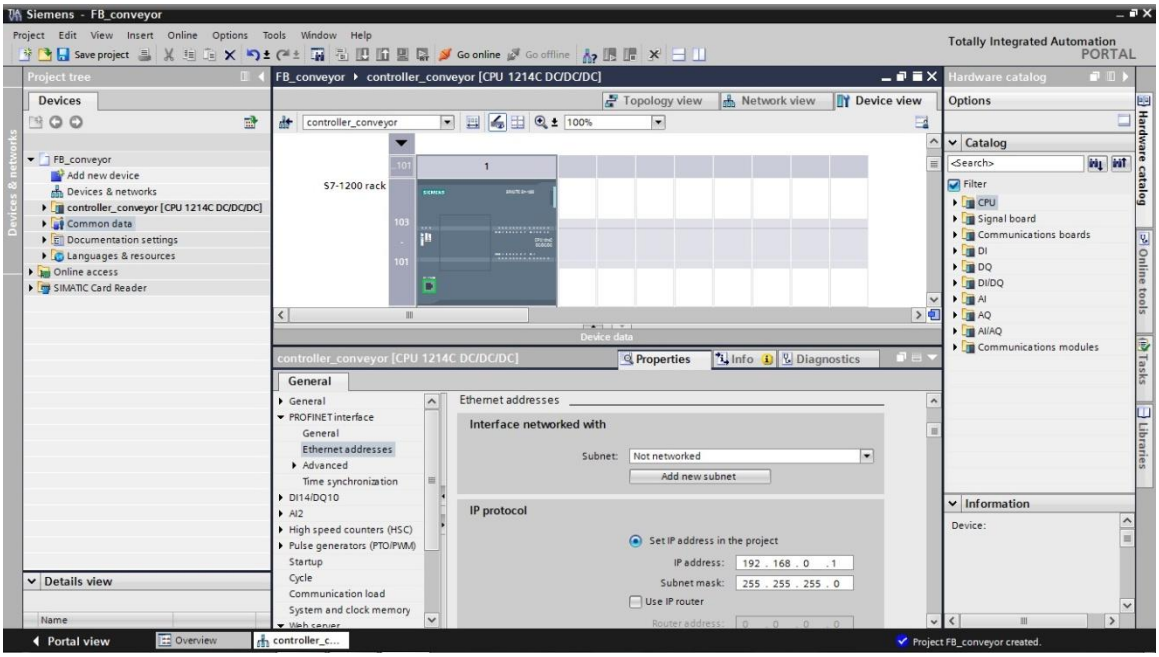
4. Dále ‘Add **new device’** s ‘Device **name conveyor control’**. Poté vybereme z katalogu **’CPU1214C’** se shodným objednacím číslem. (→ Add new device → conveyor control → CPU1214C → 6ES7 ……. → Add)



5. Nyní se TIA portal automaticky přepne na projekt view a otevře hardware konfiguraci. Můžeme zde také přidat přídavné moduly z hardware katalogu v **'Device overview'** a nastavit vstupní a výstupní adresy. Integrované vstupy CPU, mají adresy od %I0.0 do %I1.5 a integrované výstupy mají adresy od %Q0.0 do %Q1.1 (→ Device overview → DI14/DO10 → 0…1)



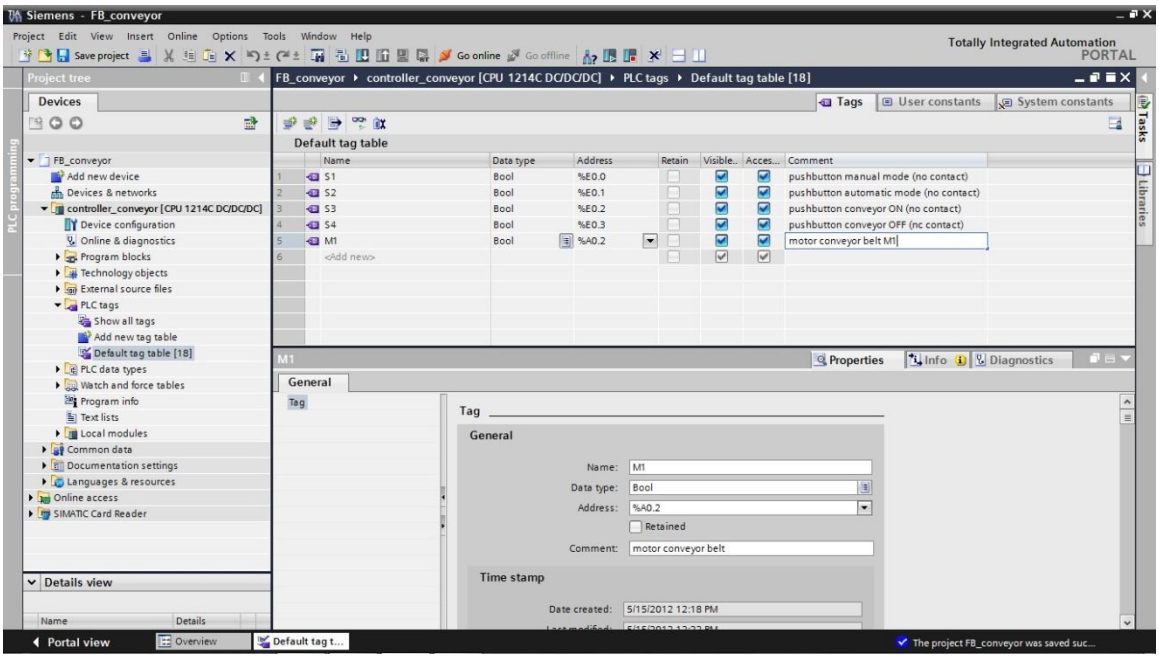
6. Aby TIA portal nahrál program do správného CPU, je nyní nutné nastavit správnou IP adresu a masku podsítě.
(→ Properties → General → PROFINET interface → IP address: 192.168.0.1 → Subnet mask: 255.255.255.0)



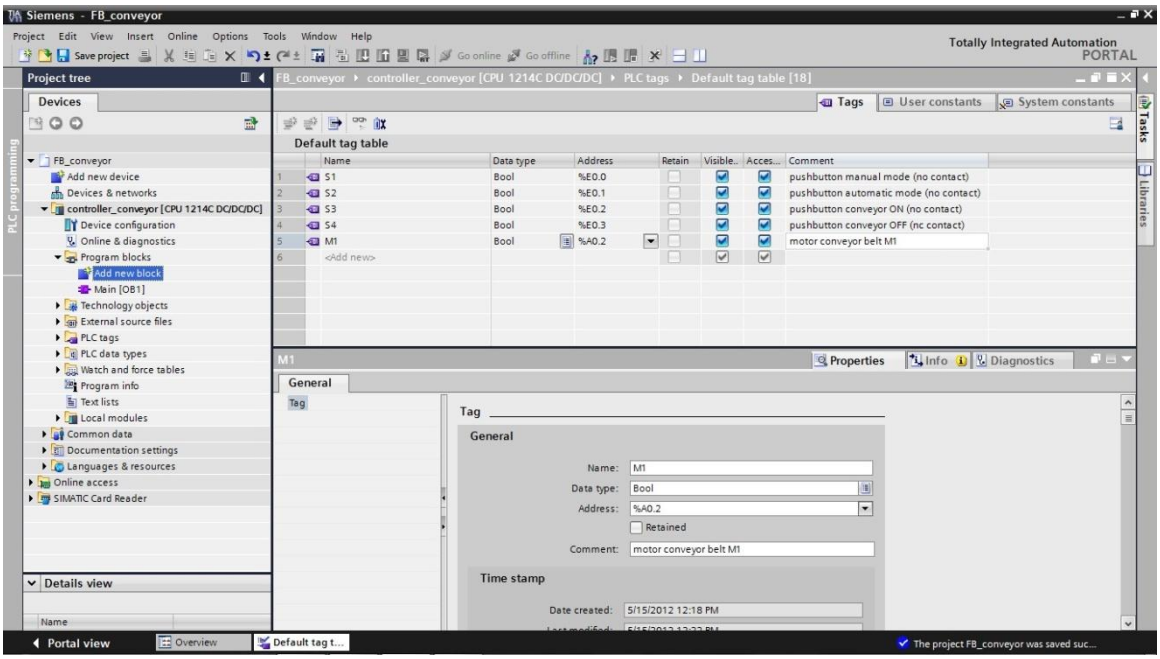
7. Protože se v moderním programování nevyužívá absolutní adresování, ale variabilní musíme zde nastavit **globální PLC tagy**.

Globální PLC tagy jsou jmény s komentáři pro vstupy a výstupy, které jsou použité v programu. Později, během programování, jsou tyto jména využita pro přístup ke globálním tagů. Tyto tagy je možné použít v celém programu ve všech blocích.

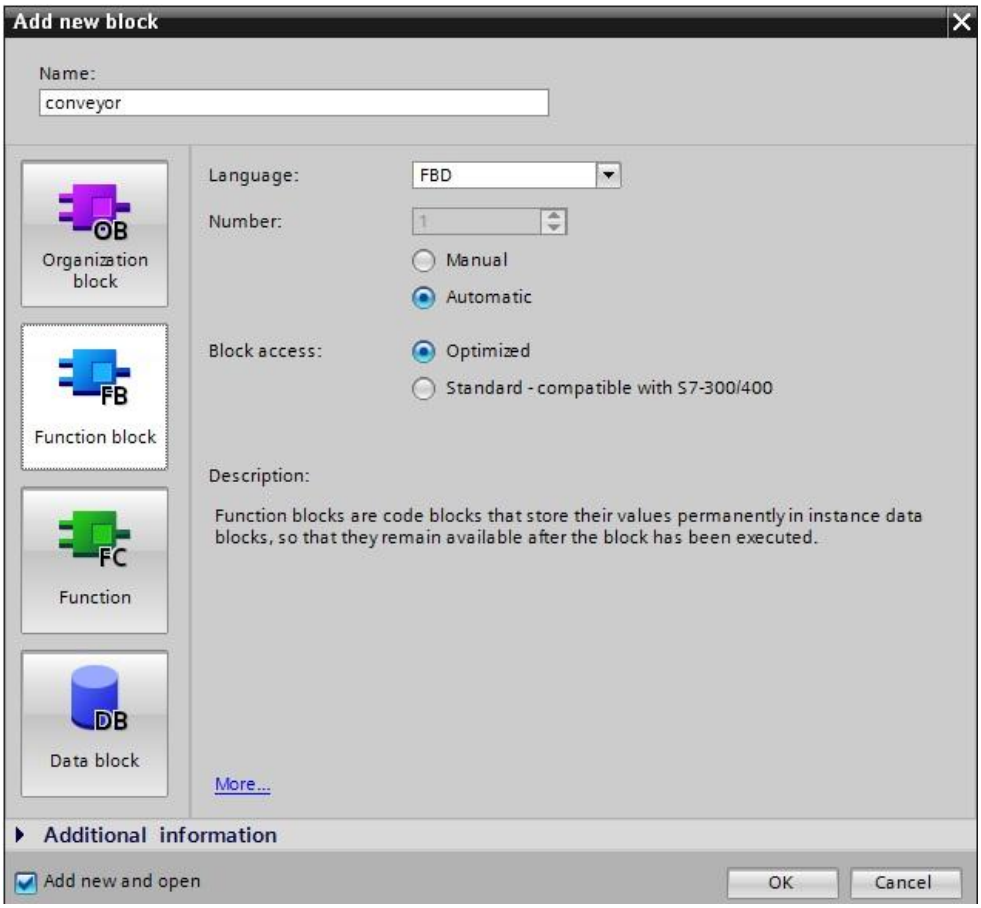
Nyní vyberte v Project Navigation **'controller_conveyorCPU1214C DC/DC/DC]'** poté 'PLC **tags'**. Otevřete tabulku **'PLC tags'** dvojklikem a zadejte jména vstupů a výstupů jak je ukázáno na obrázku níže. (→ controller_conveyor[CPU1214C DC/DC/DC]' → PLC tags→ PLC tags)



8. Pro vytvoření funkčního bloku FB1 nejdříve vybereme **'controller_conveyor[CPU1214C DC/DC/DC]'** a poté **'Program blocks'**. Dvojklik na **'Add new block'** (→ controller_conveyor[CPU1214C DC/DC/DC]' → Program blocks → Add new block)



9. Zde vybereme **'Function block (FB)'** a přiřadíme mu jméno **'conveyor'**. Jako programovací jazyk vybereme funkční blokový diagram **'FBD'**. Číslování proběhne automaticky. Protože je FB1 později vždy volán svým symbolickým jménem není toto číslo důležité. Potvrďte **'OK'.** (→ Function block (FB1) → conveyor → FBD → OK)



10. Blok **'conveyor[FB1]'** se otevře automaticky, ale než začneme psát program musíme nadefinovat rozhraní.
Když, deklarujeme rozhraní musíme ještě nadefinovat interní proměnné, použité jen uvnitř tohoto bloku.

Proměnné jsou rozdělené do dvou skupin:

- Blokové parametry, které vytvářejí rozhraní bloku pro volání v programu.

Typ	Jméno	Funkce	Dostupné v
Vstupní parametr	Input	Parametr, jehož hodnoty blok čte.	Functions, function blocks a některé tipy organization blocks
Výstupní parametr	Output	Parametr, jehož hodnoty blok zapisuje	Functions a function blocks
In/out parametr	InOut	Parametr, jehož hodnoty blok, čte, když je zavolán a po provedení zapisuje na stejné místo.	Functions a function blocks

- Lokální proměnné použité pro ukládání okamžitých výsledků.

Typ	Jméno	Funkce	Dostupné v
Dočasná lokální data	Temp	Proměnná použitá k ukládání dočasných okamžitých výsledků. Dočasná data se udržují jen po jeden cyklus.	Functions, function blocks a organization blocks
Statická lokální data	Static	Proměnné použité k ukládání statických okamžitých výsledků. Data jsou uložena, dokud nejsou znovu přepsána.	Function blocks

11. Pro náš příklad potřebujeme následující lokální proměnné.

Vstupy:

manual	Signál pro výběr pracovního módu Manual
automatic	Signál pro výběr pracovního módu Automatic
on	Startovní signál
off	Signál zastavení

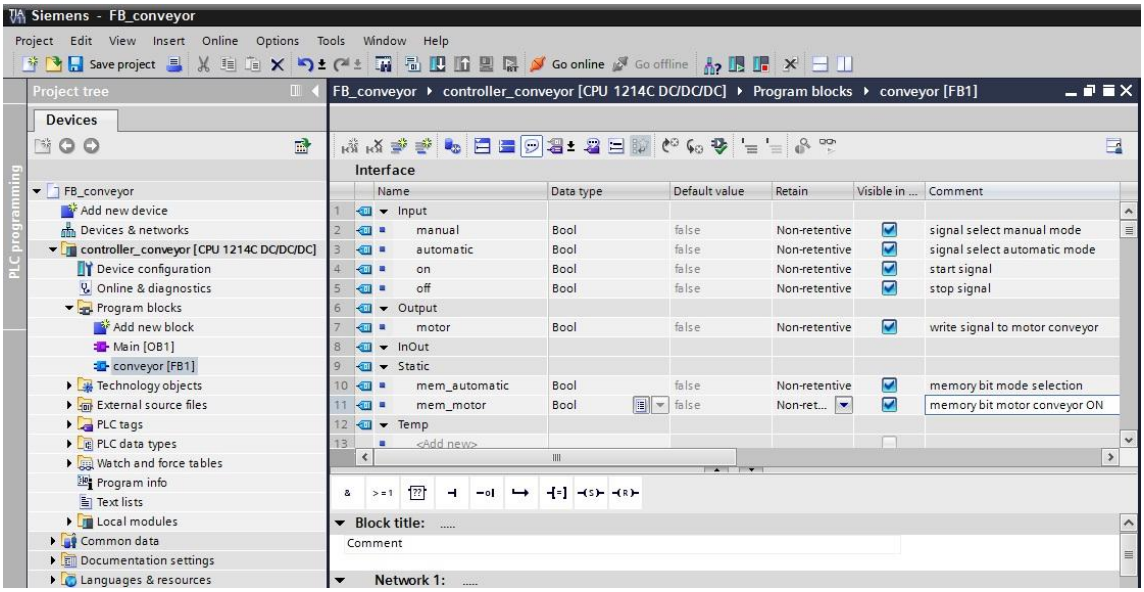
Výstup:

motor	Signál pro motor pásového dopravníku

Static (existují jen ve funkčních blocích FB):

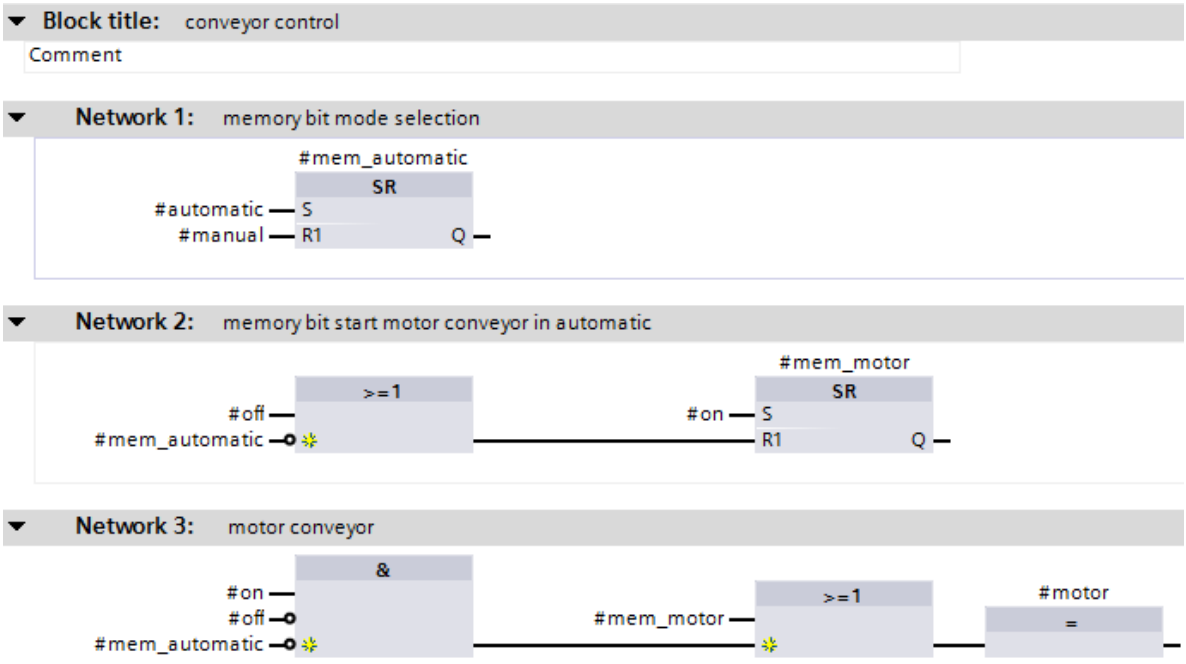
memory_automatic	Uložení předvoleného pracovního módu
memory_motor	Uložení kdy byl motor spuštěn v automatic módu

Všechny proměnné jsou typu ‚bool' to znamená binární hodnoty ‚0' a ‚1'
V tomto případě je důležité, aby status proměnných 'memory_automatic' a 'memory_motor' byl uložen po delší čas. Z tohoto důvodu musíme využít proměnnou typu **'Static'**. Tento typ proměnné existuje jen ve funkčním bloku FB.
Pro srozumitelnost by měli být všechny lokální proměnné opatřeny dostatečným komentářem.

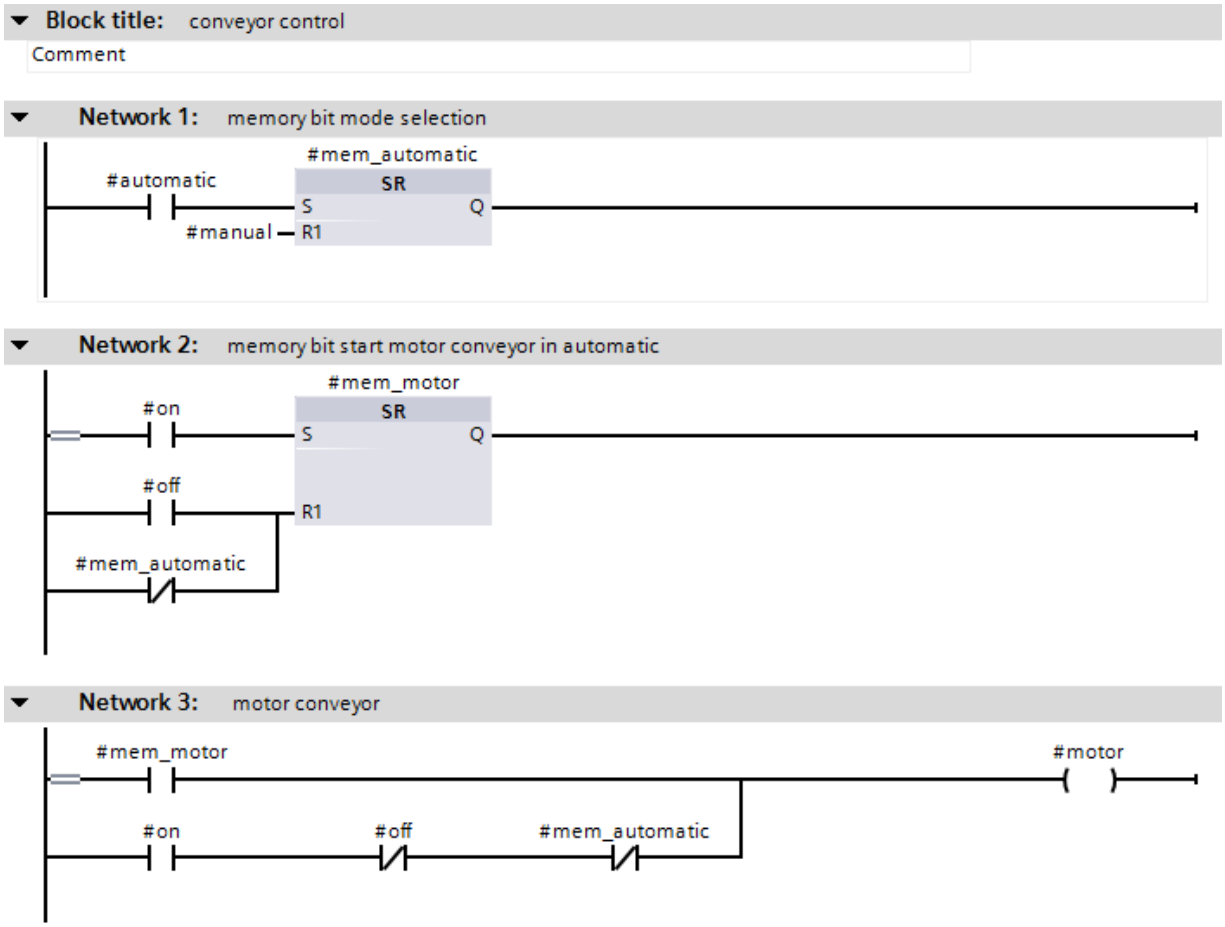


12. Poté co deklarujeme proměnné můžeme vytvořit samotný program. S použitím jmen proměnných, proměnné identifikujeme pomocí symbolu ‚#'. Náš program v FBD bude pak vypadat nějak takhle:

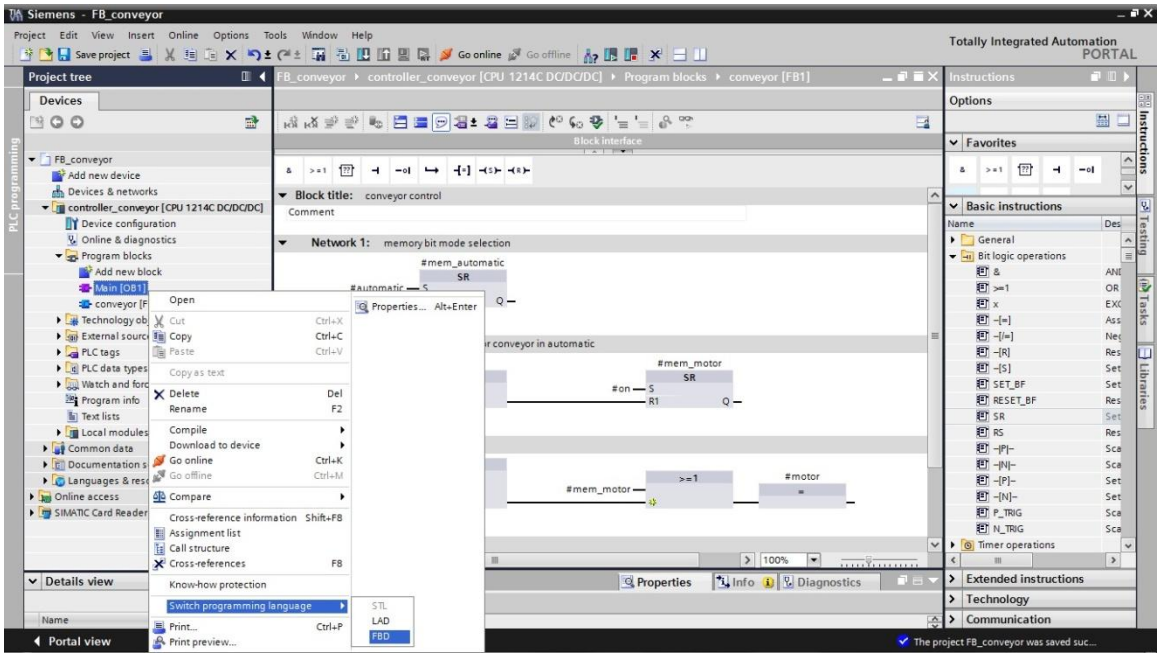
Program ve funkčním blokovém diagramu (FBD):



Program v žebříkovém diagramu (LAD):

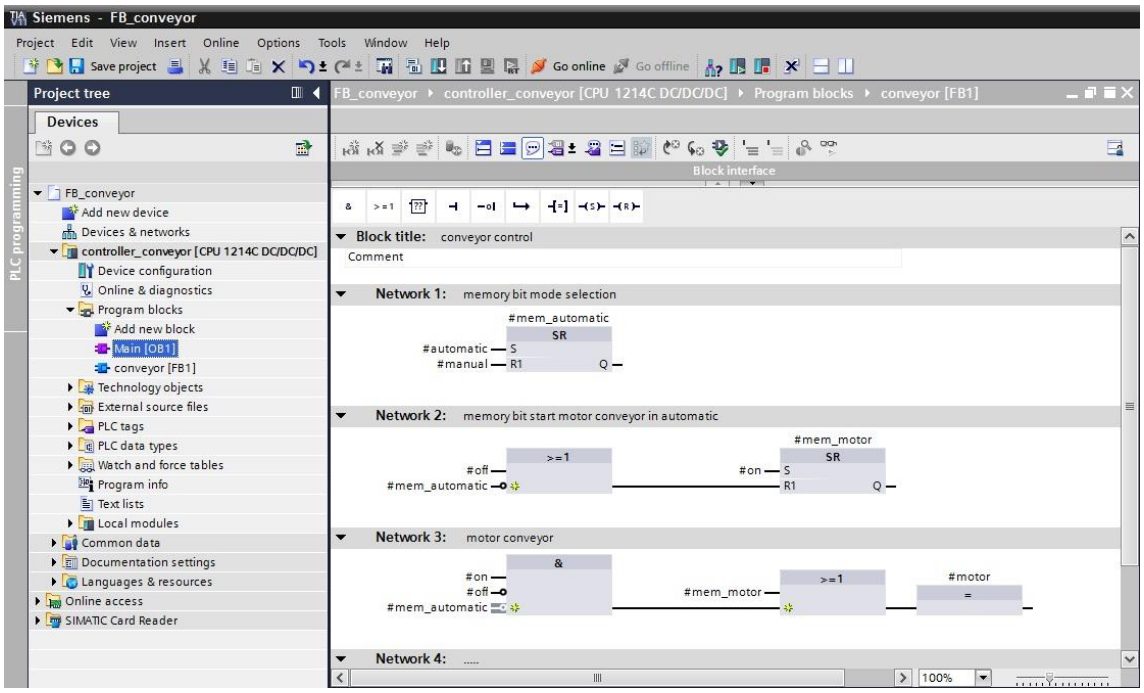


13. Poté pravý klik na **'Main[OB1]'**.
A pod **'Switch programming language'**, vybereme **'FBD'**.

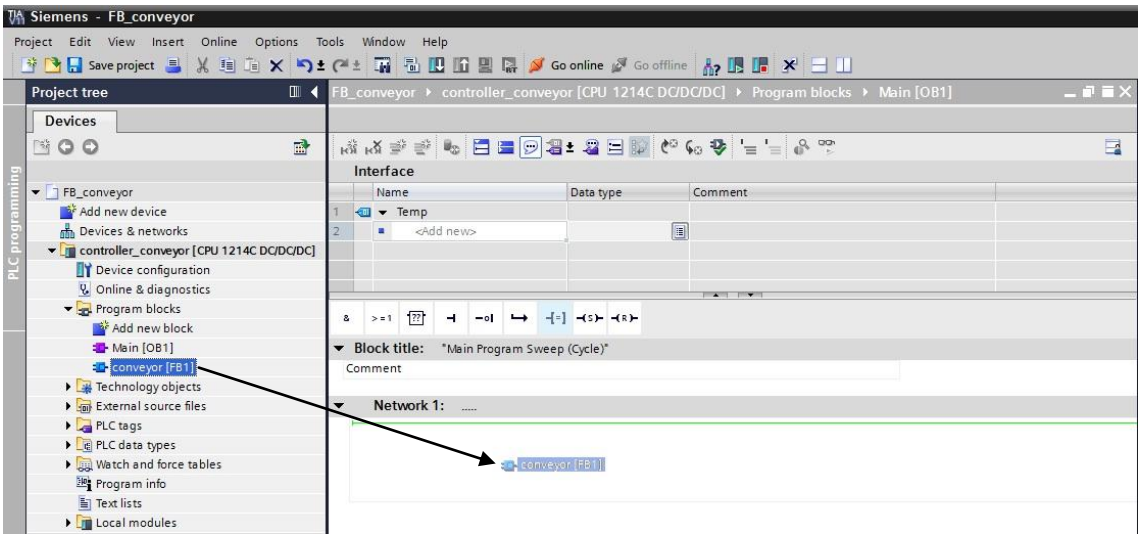


14. Nyní, blok “conveyor” musí být volán z programového bloku Main[OB1]. Jinak nebude blok proveden.

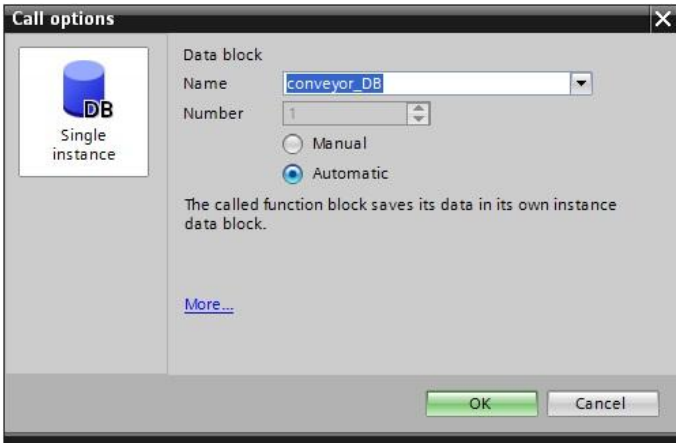
Dvojklik na **‘Main[OB1]’** a otevřeme tento blok. (→ Main[OB1])



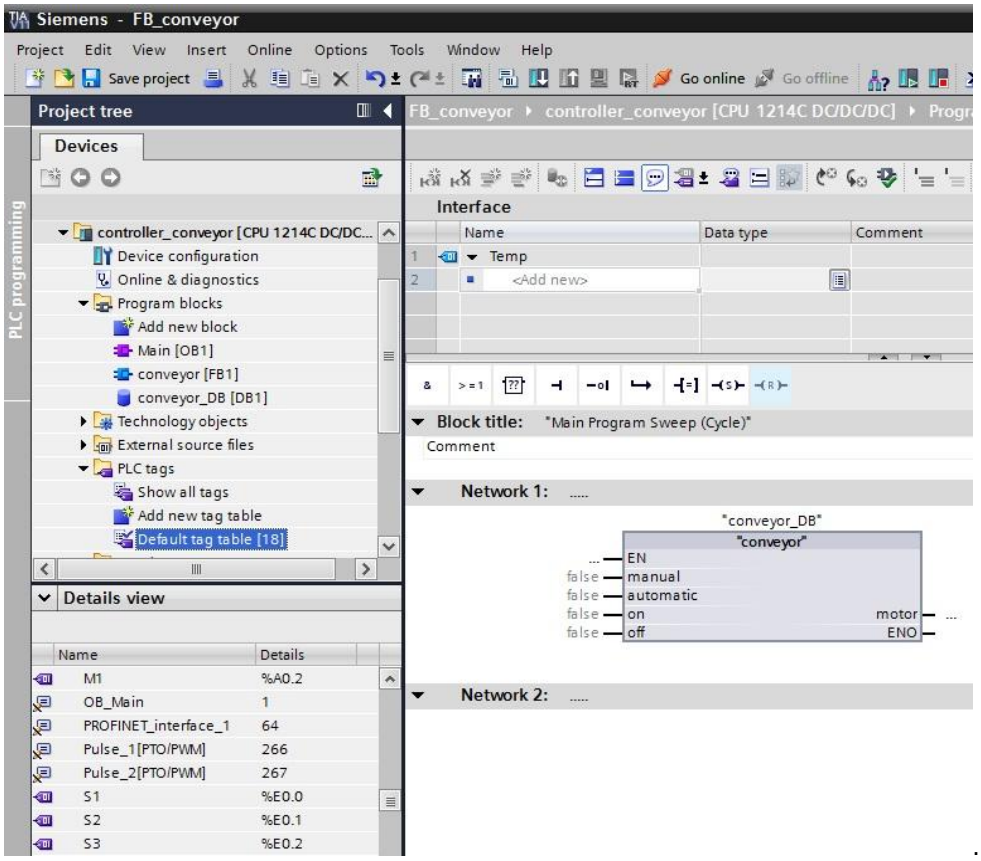
15. Nyní můžete přetáhnout blok **"conveyor[FB1]“** do sítě Network 1 v bloku Main[OB1]. (→ conveyor[FB1])



16. Protože je to funkční blok, musí mu být poskytnuta paměť. V SIAMTIC S7-1200, jsou jako paměť použity data bloky DB nazývané **Instance Data block**.
Zde je specifikován a vytvořen automaticky **'automatically'.** (→ Automatic→ OK)

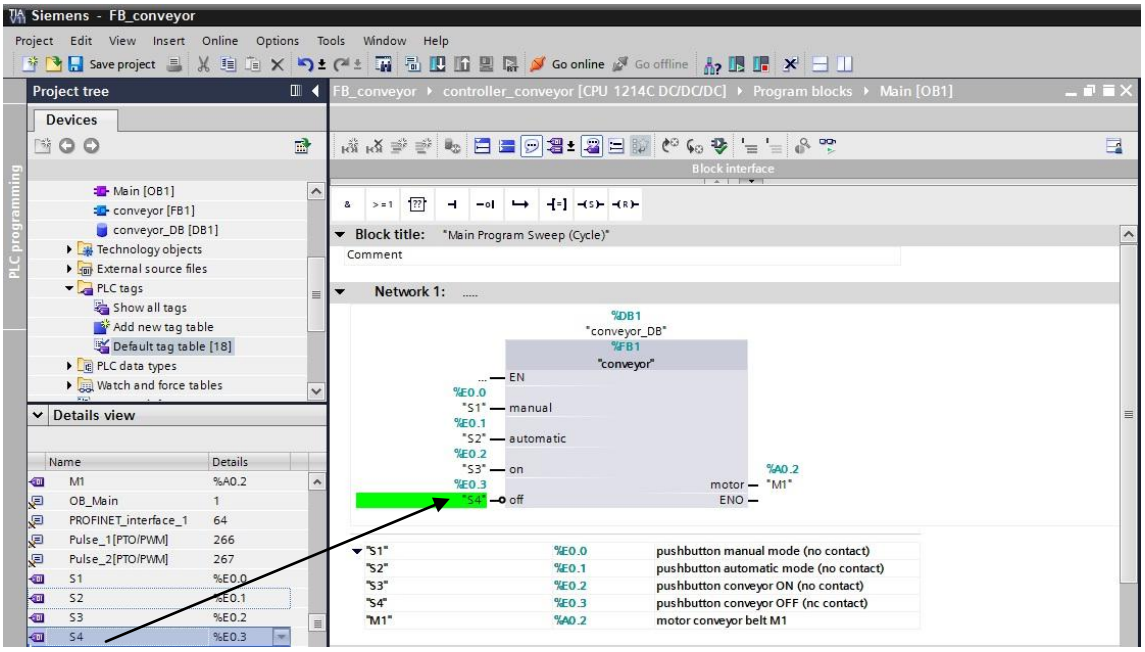


Zvýrazníme základní tabulku tagů.



17. V OB1, nyní propojíme vstupní a výstupní proměnné s PLC tagy zde zobrazenými. Stačí jen přetáhnout PLC tagy na blokové proměnné.

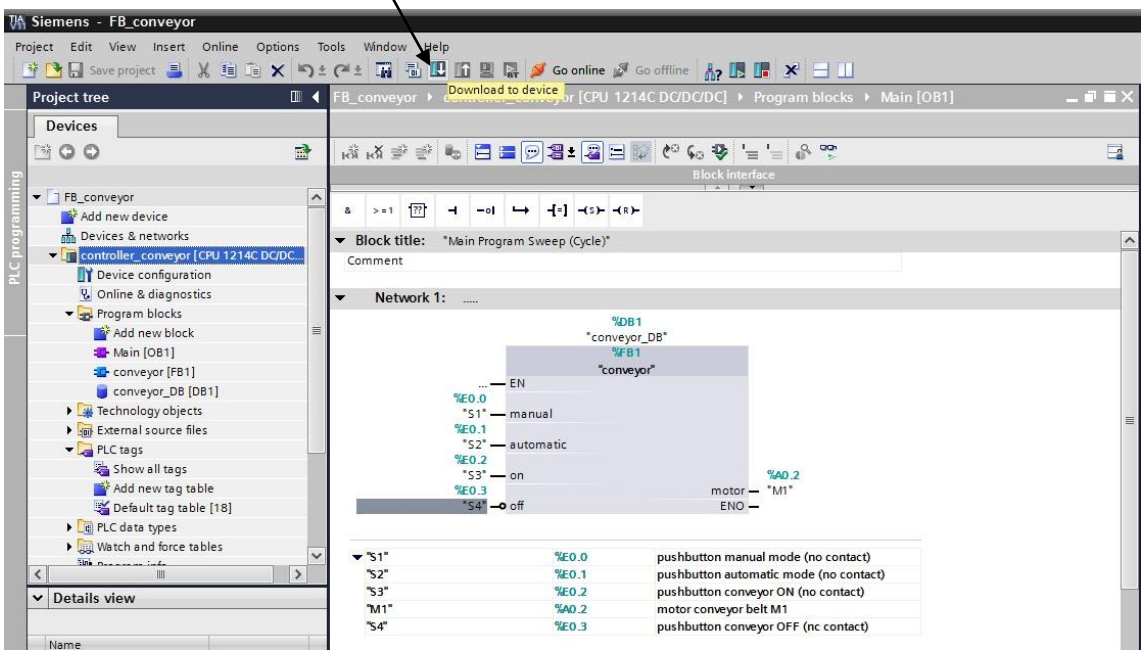
Kliknutím na Save project, je projekt uložen. (→ "S1“ → "S2“ → "S3“ → "S4“ → "M1“ → Save project)



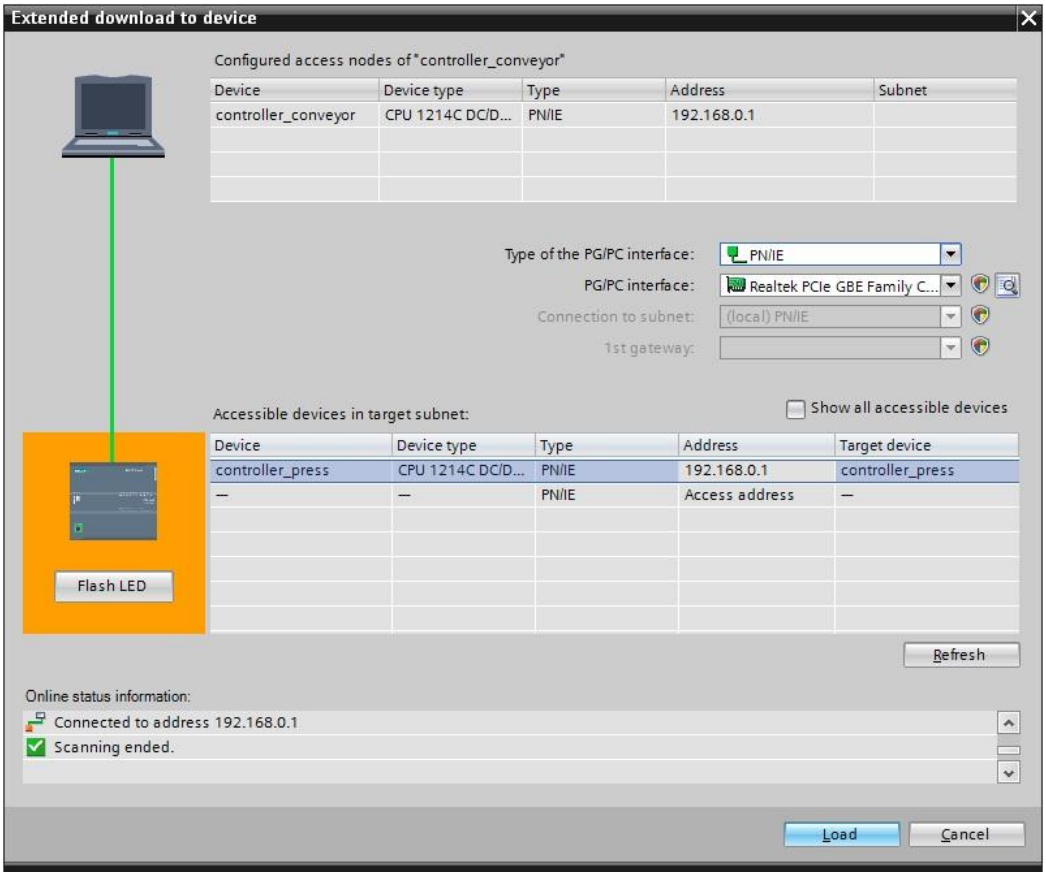
Důležité!

Off tlačítko S4 je rozpojení obvodu (NC) a musí být negované na bloku během spojování. Tzn. Off blok je funkční při stisku tlačítka Off a na vstupu %I0.3 není žádný signál.

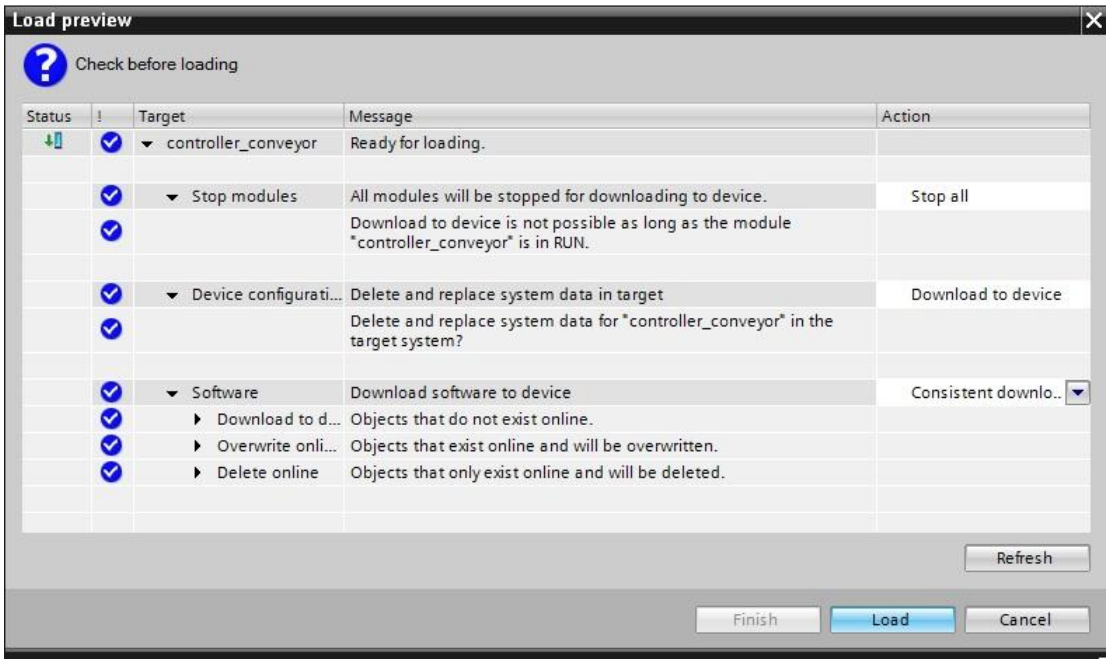
18. Pro nahrání celého programu do CPU, nejdříve zvýrazníme složku **'controller conveyor'** a poté stiskem nahrajeme do zařízení. (→ controller_conveyor →)



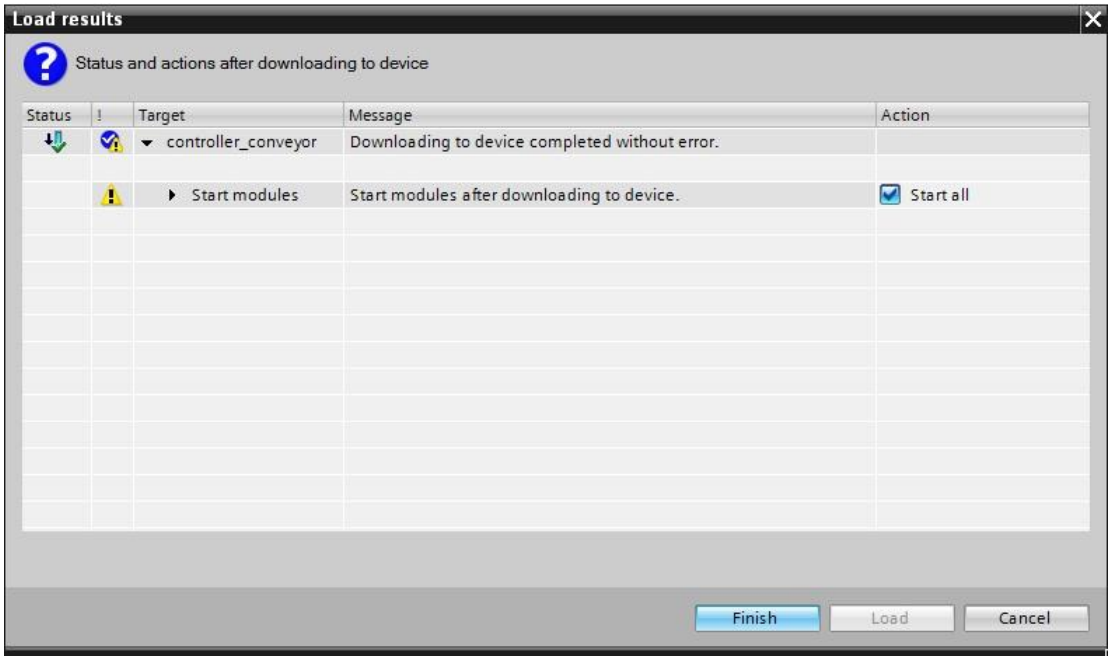
19. Pokud jste zapomněli specifikovat rozhraní PG/PC, zobrazí se okno, ve kterém to teď můžete udělat.
(→ PG/PC interface for loading → Load)



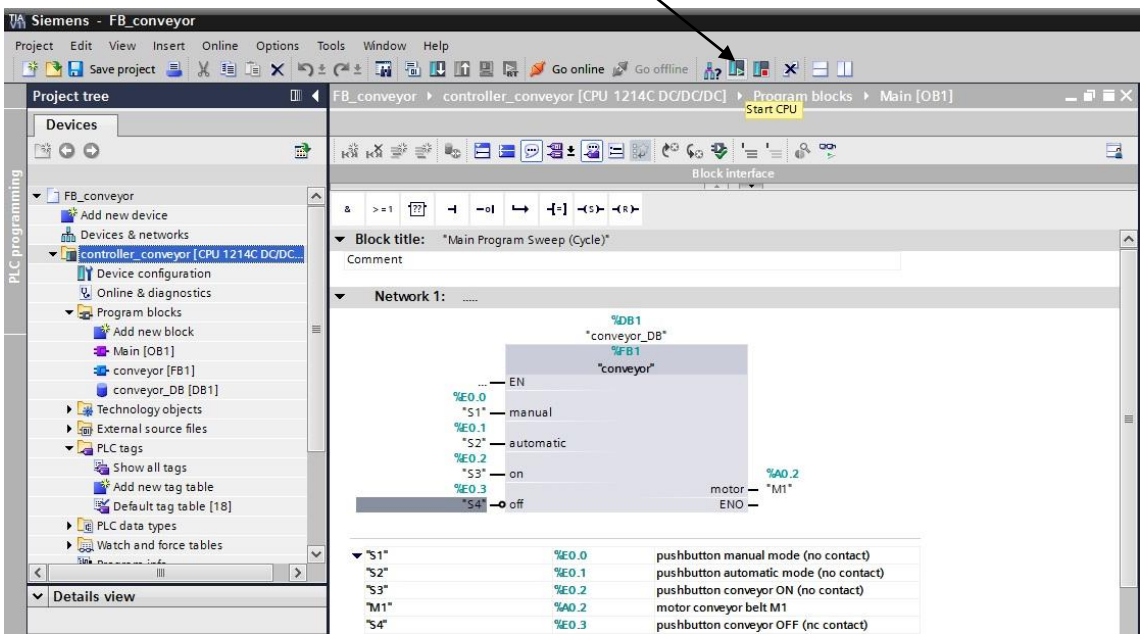
20. Klik na 'Load' ještě jednou. V okně je zobrazen během nahrávání status. (→ Load)



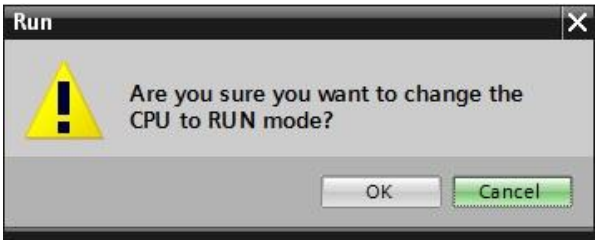
21. Pokud bylo nahrávání úspěšné, zobrazí se to v okně. Klik na **'Finish'.** (→ Finish)



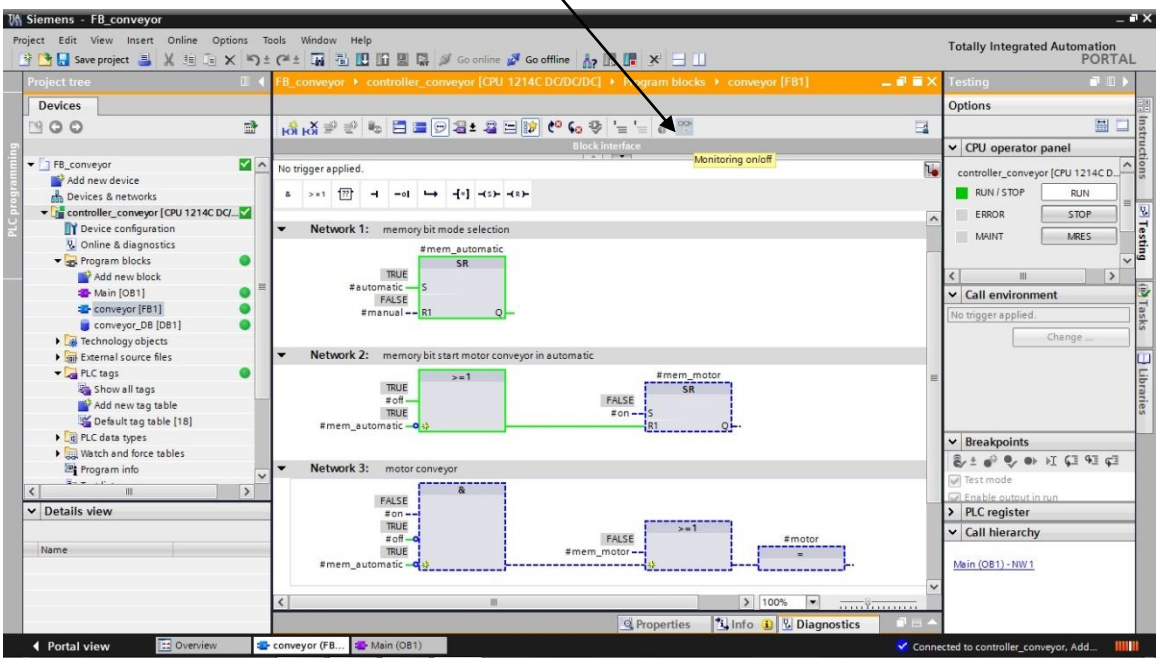
22. Nyní spustíme, CPU kliknutím na symbol  .



23. Kliknutím na **'OK'** potvrdíme, že chceme opravdu spustit CPU. (→ OK)



24. Kliknutím na  spustíme nebo vypneme monitoring. Během testování programu, můžeme kliknutím na blok “conveyor“ monitorovat stav vstupů a výstupů, ale také běh programu v bloku “conveyor“. (→ conveyor[FB1] → )



25. Díky tomu, že blok “conveyor“ byl vytvořen podle pravidel pro tvorbu standardních bloků (žádné globální proměnné v bloku!!!!!), můžeme ho volat a využít vícekrát.

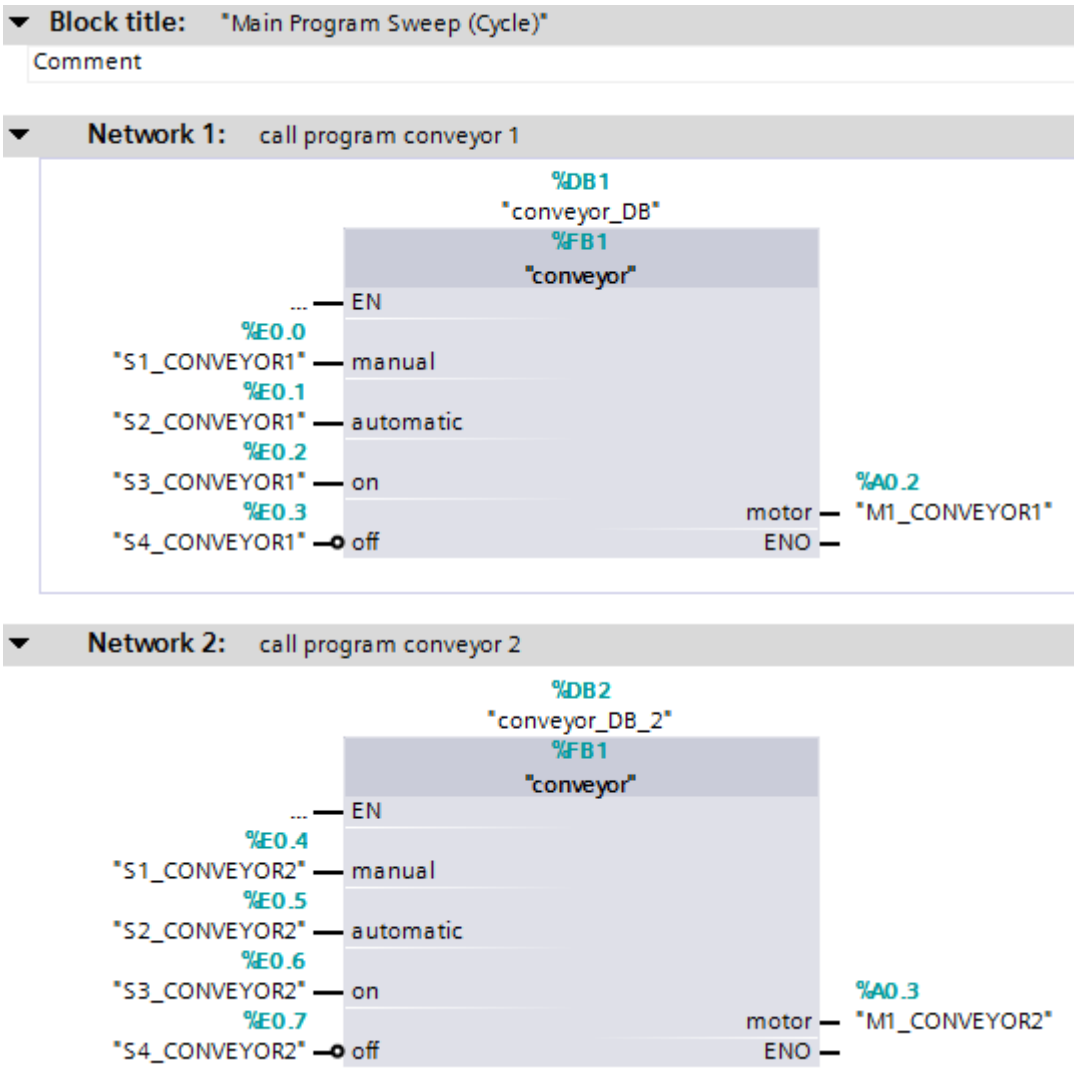
Níže vidíte otevřenou tabulku PLC tagů jsou v ní uvedeny vstupy a výstupy pro dva dopravníky.

Default tag table

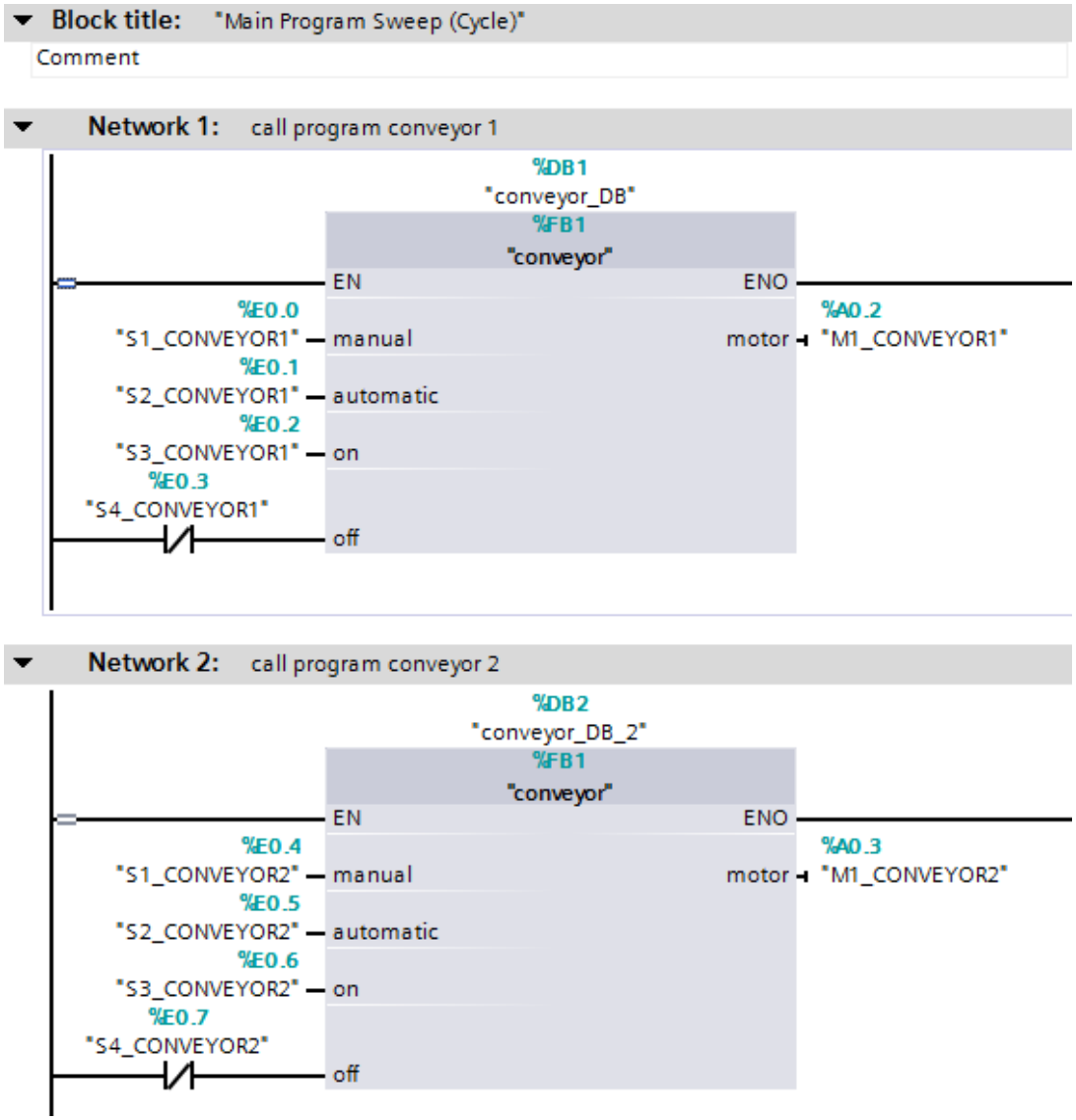
	Name	Data type	Address	Retain	Visible..	Acces...	Comment
1	S1_CONVEYOR1	Bool	%E0.0	☐	☑	☑	conveyor1 pushbutton manual mode (no contact)
2	S2_CONVEYOR1	Bool	%E0.1	☐	☑	☑	conveyor1 pushbutton automatic mode (no contact)
3	S3_CONVEYOR1	Bool	%E0.2	☐	☑	☑	conveyor1 pushbutton conveyor ON (no contact)
4	S4_CONVEYOR1	Bool	%E0.3	☐	☑	☑	conveyor1 pushbutton conveyor OFF (nc contact)
5	M1_CONVEYOR1	Bool	%A0.2	☐	☑	☑	conveyor1 motor conveyor belt M1
6	S1_CONVEYOR2	Bool	%E0.4	☐	☑	☑	conveyor2 pushbutton manual mode (no contact)
7	S2_CONVEYOR2	Bool	%E0.5	☐	☑	☑	conveyor2 pushbutton automatic mode (no contact)
8	S3_CONVEYOR2	Bool	%E0.6	☐	☑	☑	conveyor2 pushbutton conveyor ON (no contact)
9	S4_CONVEYOR2	Bool	%E0.7	☐	☑	☑	conveyor2 pushbutton conveyor OFF (nc contact)
10	M1_CONVEYOR2	Bool	%A0.3	☐	☑	☑	conveyor2 motor conveyor belt M1
11	<Add new>			☐	☑	☑	

26. Nyní můžeme blok **"conveyor“** volat v OB1 dvakrát, pokaždé s jiným zapojením. Pro každé volání je nastaven vlastní instanční data blok.

Program ve funkčním blokovém diagramu (FBD):



Program v žebříkovém diagramu (LAD):



Dva dopravníkové pásy můžeme nyní řídit pomocí stejného bloku. Pouze instanční data blok musí být pro každý dopravník zvlášť.